Logic for Knowledge Representation, Learning, and Inference

Luciano Serafini

serafini@fbk.eu

Version August 28, 2023

Contents

CHAPTER 1

Knowledge Respersentation, Learning and Inference

1. Artificial Agents

Artificial intelligence has the main objective of developing artificial agents that are capable of (supporting humans in) taking decisions about what to achive some or more (human) goal in a given environment.

An agent can directly operate in the environment, e.g. robots, the stock marked autonomous agents, or they can suggest actions to be performed by other agents, usually humans. e.g., in recommended systems, or they can provide the necessary information for another agent, usually humans, to take decisions, e.g., agents that perform simulations about some environment.

We use the term "environment" in a very broad sense. It indicates all the relevant aspects of the context in which an agent makes decisions and operates. Examples of environments are: the ambient in which a physical robot is operating, or the worldwide Covid pandemic, a social network, or a smart city.

The environment where the agent operates is dynamic in the sense that it can change both by the effect of the actions executed by the agent or because of some factor external and independent from the agent. The environment is unpredictable in the sense that, at the moment in which we (as engineers) are designing and implementing the agent doesn't know all the possible situations in which the agent will operate.

According to the most famous book on AI Russell and Norvig 2010

> *An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.*

To make effective decisions in a specific situation, the agent needs sufficient knowledge about the the current situation and it has to accumulate sufficient experience on the effects of its actions, and the general laws obeyed by the environments. All this knowledge is not available when the agent is created, but it is the agent itself that should be capable to *learn knowledge by interacting with the environment*, *represent the learned knowledge in some internal structure (model)*, *perform inference (reasoning) on these models to support its decisions.*

EXAMPLE 1.1. *Consider the example of a robotic agent that has to operate in a simple environment that contains four coloured blocks. With its arm, the agent can move around the blocks, stacking them one on top of the other; it can move around the environment to get close to some blocks. Furthermore it perceives the environment through an RGB camera. This simple scenario is represented in the Figure 1.*

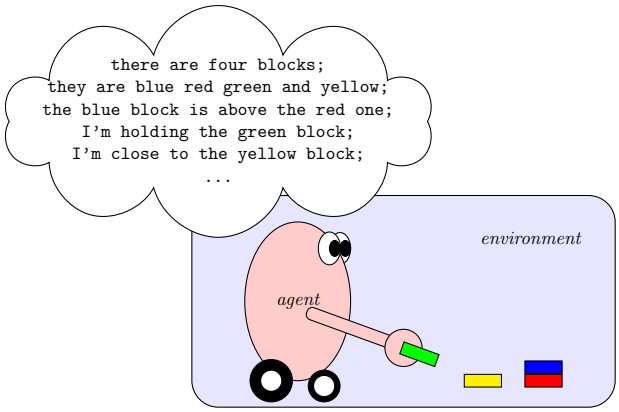


FIGURE 1.

In the following chapters, we will consider a set of methods for knowledge representation and the different types of inference/reasoning and learning methods in each type of model.

2. Models for Knowledge

Different models for representing knowledge. We concentrate on three models, Logical models, probabilistic models, and neural models, and the integration of the three.

2.1. Logical models. A logical model of an agent's knowledge of the environment is a *set of sentences* of a logical language that encodes a set of propositions about the environment that the agent is supposed to be true. There are many types of logical languages, e.g., propositional language, first-order language, modal language, temporal language, logic programs,... The father of logical model for AI agents is John McCarty, who, in McCarthy 1959, proposes to uses declarative logical language *"[...] to make programs*[1] *that learn from their experience as effectively as humans do"*. Successively he observed that *"In order for a program to be capable of learning something it must first be capable of being told it."*

A good summary of many different logical models is provided in the Handbook of Knowledge Representation Lifschitz, Porter, and Van Harmelen 2008. Logical models provide information of what is true, what is false and what logically follows from some premises. Minker Minker 2012 offers an overview of how logical models can be built and how they can be used for inference, decision making and planning.

2.2. Probabilistic models. Probabilistic models are particularly designed to represent uncertainty about the state of a domain. They allow associating a measure of the likelihood of a state of the world. A probabilistic model describes a domain in terms of *random variables* that can take values is some abstract domain, e.g. a finite set $\{1.2.3, \ldots, n\}$, the infinite set of natural or real numbers. The different configurations of the world are represented by the assignments of the random variables with a value in their domains. Probabilistic models associate a

[1]One can read "program" as "agents".

measure of the probability to encounter such a situation (joint probability distribution). A probabilistic model provides a formal language that allows specifying the joint probability distribution. Examples of probabilistic models are explicit probabilistic models, e.g., discrete models (Uniform, Binomial, Bernoulli, Poisson) and continuous models (Normal (or Gaussian), Exponential, Dirichlet, ...), directed/undirected) graphical models Maathuis et al. 2018, (e.g, Markov random fields, Bayesian networks,...).

Inference in probabilistic models has the objective of calculating the probability that a (set of) random variables takes a (set of) values possible conditioned from some facts about the domain (observations or priors) that are known to be true. These algorithms are strictly connected to the language used to specify the probabilistic model

Learning in probabilistic models has the objective of inducing some (or all) parameters of the probabilistic model starting from a set of observations of states of the world. As for the case of inference, learning algorithms strongly depend on the formalism used to specify the probabilistic model.

2.3. Artificial Neural Networks. [From wikipedia] Artificial neural networks (ANNs), usually simply called neural networks (NNs) or neural nets are computing systems inspired by the biological neural networks that constitute animal brains.

An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron receives signals then processes them and can signal neurons connected to it. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold.

Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer) to the last layer (the output layer), possibly after traversing the layers multiple times.

3. Integrating Logical models with other models

In this course, we are interested also in how logical models can be integrated with probabilistic models and neural networks. There are many reasons why one wants to combine logical models with neural networks and/or probabilities.

3.1. Integrating logic and probabilistic models. Logical models provide a compact representation of a world of objects which have properties and are in relation. In other words, logical languages are very good to express structured worlds. Logical language allows the imposition of restrictions on the structure of the world. I.e., logical language can express the fact that a certain state of the world is possible, or impossible, with no intermediate degree. In probabilistic models, instead, the world is described by a flat (unstructured) set of random variables. The possible states of the world are described by an assignment to the variables,

and therefore there is no structure in such a world. On the other hand, probabilistic models allow expressing that a certain configuration of the world is more likely than another.

The integration of logic and probabilistic models has the main the objective of taking the goods from both models, namely allowing probabilistic inference on structured domains. There is a large set of attempts to combine logic and probability in artificial intelligence. In this course we will introduce two of them: Probabilistic Logic Programming Lukasiewicz 1998; Kimmig et al. 2011; Gutmann, Thon, and De Raedt 2011 and Markov Logic Networks Richardson and Domingos 2006; Lowd and Domingos 2007

3.2. Integrating logic and neural networks. Logical language allows them to provide precise and crisp definitions of concepts. For instance the first-order logic formula $\forall x(\mathit{MasterStudent}(x) \leftrightarrow \mathit{Student}(x) \wedge \exists y(\mathit{MasterCourse}(y) \wedge \mathit{Enrolled}(x, y)))$ provides a precise definition of the concept of master student as a student who is enrolled in some master course. This definition is very useful when we want to find all the master students of a school by querying the database of such a school. However, the definition is not usable if we want to recognize if a person is a master student from his/her social media profile, or from some pictures of him/her. For this second task, it is much more effective to train a neural network with positive and negative examples. Combining the two types of models would offer the possibility to combine the two advantages described above.

There are many other strengths and weaknesses of neural models and logical models that justify a combination of the two approaches. They are summarized in Table 1. Let us describe them shortly.

- Neural networks are very effective to learn, from a set of positive and negative examples, how to classify an object in a certain category starting from its features from positive and negative examples. Logical models instead have some difficulties in learning general definitions from examples.
- Neural networks are good at processing objects which are described in terms of high-dimensional features. For instance, images, while in logic objects are described as atomic (abstract) entities and it is not easy to take into account numeric features associated with the objects.

Strength and Weakness of	**neural network**	**logical models**
Automatic learning from raw data	⋆⋆⋆⋆⋆	⋆⋆
Dealing with hight througput data	⋆⋆⋆⋆⋆	⋆⋆
Robust to exception	⋆⋆⋆⋆	⋆
Size of training data	⋆⋆	⋆⋆⋆⋆
Explanability of inference	⋆	⋆⋆⋆
Adding commonsense knowledge	⋆	⋆⋆⋆⋆⋆
Complexity of inference	⋆⋆⋆⋆⋆	⋆⋆

TABLE 1. Comparing logical and neural models

- Neural network is robust to exceptions and outlier data. In logic instead, if you state that a property holds *for all* objects, then a single exception makes the model inconsistent. So it is not very adaptable to deal with exceptions.
- Neural networks are data-hungry. I.e., to train a neural network you need to provide a sufficiently large amount of training examples usually manually annotated. The process of manual annotation is very costly. In logic instead one can express a general property with just one simple formula, and even if this requires the intervention of human knowledge this does not constitute a long process.
- Inference in a neural network is a black box. It is very difficult to find an explanation of why a neural network reaches a certain conclusion. In logic instead, the inference is obtained by applying inference rules and every conclusion is associated with a *deduction or proof* (i.e., a sequence of application of inference rules) that provide the explanation of the conclusion.
- Adding commonsense knowledge, such as for instance "a cat has one tail which is opposite to his/her head") is very difficult, since it needs to be provided either through examples or by changing the architecture of the network. In logical models, instead, commonsense knowledge can be added by using a single formula that encodes such knowledge.
- Inference in neural networks is usually very fast (especially on GPUs) since it is just a matter of computing compositions of linear and non-linear functions. Inference in logic instead might be exponentially complex since it involves the search for a proof or a deduction or the desired conclusion.

In the last few years, there has been a number of proposals for combining neural networks and logical models. In this course we will present two of them: Logic Tensor Networks Badreddine et al. 2022 and Knowledge Enhanced Neural networks Daniele and Serafini 2019.

3.3. Integrating logic, neural networks and probabilities. Finally, there are methods that combine all the three types of models, neural networks, probabilistic models, and logical models. One examples that will be described in this course is DeepProbLog Manhaeve et al. 2018 that combines probabilistic logic programming with neural networks, However, there are many other approaches De Raedt et al. 2020 provides a survey.

CHAPTER 2

Propositional Logic

1. What is a proposition

According to Stanford Encyclopedia of Philosophy[1] , the term "proposition" has a broad use in contemporary philosophy. It is used to refer to some or all of the following: the primary bearers of truth-value, the objects of belief and other "propositional attitudes" (i.e., what is believed, doubted, known, etc.), the referents of that-clauses, and the meanings of sentences.

Propositions can be expressed by sentences of a natural language (English, Italia, etc.) Examples of propositions are "John is a teacher", "John is rich' and "John is a rock singer". Notice that there is a difference between "sentence" and "proposition". A sentence is a string that *expresses a proposition*. I.e., the meaning of a sentence is a proposition. In natural langue we say that two sentences are equivalent, when they expresses the same proposition. E.g.,

(1) the brother of my mam is blond;
(2) my uncle is blond;

One of the most important characteristic of proposition is that they can take a truth value. IN classical propositional logic, there are only two truth values *true* and *false*. Notice that, one can imagine a situation in which a proposition is not completely true or completely false, or it is both true and false. To treat these situations there are other logics (we will see fuzzy logic). So, don't believe that the world of logic is limited to "true and false".

Complex propositions can be build by combining simpler (atomic) propositions. For instance the sentence "Paolo is painting and Renzo is playing piano" expresses a proposition which is the *conjunction* of the propositions expressed by the sentences "Paola is painting" and "Pietro is playing piano". There are other ways to compose complex proposition from simpler ones. For instance one could "disjunct" two propositions. E.g., the proposition "Paolo is painting or Renzo is playing piano" is the *disjunction* of the two simpler propositions. Another example is negation. E.g., the proposition rexpressed by the sentence "Renzo is not playing the piano" is the result of negating the proposition expressed by the sentence "Renzo is playing the piano".

Notice that the truth value of the simpler proposition determine the truth value of the complex proposition. This property is sometimes referred as *truth-functionality* and has been formally characterised by the Polish logician Alfred Tarski in 1933.

[1]see `https://plato.stanford.edu/entries/propositions/`

2. The language of propositional logic

The language of propositional logic allows to "speak about" propositions. The basic components of a propositional language is the set of *propositional variables* $\mathcal{P}$. This can be a finite or an infinite set (indeed the set of all propositions in general might be infinite).

$$\mathcal{P} = \{p_1, p_2, p_3, \dots\}$$

When we stud When we want to use propos y the general propoerties of PL, we don't care about which proposition is denoted by each single p_i in $\mathcal{P}$; we only assume that each p_i refers to some proposition. This is the reason why p_i is called *variable*. Instead, when we want to use PL to represent some concrete scenario, we have to specify the specific proposition which is denoted by each propositional variable. For instance if we want to use PL to describe what Paola and Renzo are doing. we can define the set of propositional variables:

$$\mathcal{P} = \{p, r\}$$

where

- p is the proposition expressed by the sentence "Paola is painting";
- r is the proposition expressed by the sentence "Renzo is playing the piano".

The language of PL allows to express also complex proposition which are the result of negating a proposition, and combining two propositions with conjunction, disjunction, implication, and equivalence. To allow this PL introduces the following set of symbols called *propositional connectives*

- $\neg$ for negation (not);
- $\wedge$ for cojunction (and);
- $\vee$ for disjunction (or);
- $\rightarrow$ for implication (if ... then ...);
- $\leftrightarrow$ for equivalence (if and only if).

In the literature one can find alternative symbols for implication and equivalence, which are $\supset$ and $\equiv$. We will use both the symbols. Two additional symbols "(" and ")" are also added to PL, which allows one to express the correct separation of simple propositions that occour in a complex propositions. Parenthesis in PL play the similar role played by punctuation symbols (e.g., ".", ";", ",", ...) in natural language.

In summary, the *alphabet* of a propositional language is composed by

- a set $\mathcal{P}$ of propositional variables (also called non-logical symbols);
- the set $\neg$, $\wedge$, *vee*, $\rightarrow$, and $\leftrightarrow$ of propositional connectives (also called logical symbols);
- parenthesis symbols (and).

From this alphabet the set of *well formed formula* are defined by induction as follows:

Definition 2.1 (Well formed formulas). *Given a set $\mathcal{P}$ of propositional variables, the set of* well formed formulas *is inductively defined as follows:*

(1) every $p \in \mathcal{P}$ is a well formed formula;
(2) if ϕ is a well formed formula then $\neg\phi$ is a well formed formula;

(3) if ϕ and ψ are well formed formula, then $\phi \circ \psi$ is a well formed formula for $\circ \in \{\wedge, \vee, \to, \leftrightarrow\}$;
(4) nothing else is a well formed formula.

Each element $p \in \mathcal{P}$ is called also atomic formula. *The set* wffs($\mathcal{P}$) *is the set of well formed formulas that contains only the propositions in $\mathcal{P}$*

An additional symbol, that is often use in propositional logic is the $\bot$ symbol. Intuitively $\bot$ is a constant that represents the proposition that is always false.

The term "formula" us used as a short version of "well formed formulas" when there is no confusion. We also shorten well formed formula with "wff".

EXAMPLE 2.1 (Formulas and non formulas).

Formulas	**Non formulas**
$p \to q$	pq
$p \to (q \to r)$	$(p \to \wedge((q \to r)$
$(p \wedge q) \to r$	$p \wedge q \to \neg r \neg$

Notice that the formulas on the left can be build by applying the rules of Definition 2.1. For instance the formula $(p \wedge q) \to r$ can be buld as follows

(1) p is a wff for item 1;
(2) q is a wff for item 1;
(3) $p \wedge q$ is a wff for item 3;
(4) $(pwedgeq) \to r$ is a wff for item 3.

Notice that in the last step we use parenthesis, in order to specify how the formula is build. Indeed if one does not use parenthesis, the formula would look as

$$p \wedge q \to r$$

The above formula however can be build in two ways.

- *first build $p \wedge q$ and then $p \wedge q \to r$ (as we indeed do)*
- *first buld $q \to r$ and then $p \wedge q \to r$*

To distinguish the two ways of constructing the formula we use parentes. obtaining

$$(p \wedge q) \to r \tag{1}$$

$$p \wedge (q \to r) \tag{2}$$

Notice that the two formulas above represents different propositions.

EXAMPLE 2.2. *Suppose that*

- *p stands for "Paola is painting";*
- *q stands for "outside it's raining";*
- *r stands for "Renzo is playing the piano".*

we have that the two formulas refers to the propositions expressed by

(3)
If Paola is painting and outside it is raining, then Renzo is playing the piano;

(4)
Paola is painting and, if outside it is raining, then Renzo is playing the piano

Notice that the proposition expressed by the two sentences above are different. So they are the corresponding propositional formula.

In some cases, the use of parenthesis will result in an awkward notation. In order to minimize the usage of parenthesis only when they are necessary, PL provide a "default" order of construction in case there are no parenthesis. This default order of application of rule of construction is specified by associating a *priority ordering* between connectives. $\neg$ has the highest priority, then $\wedge$, $\vee$, $\rightarrow$ and $\equiv$.

Symbol	Priority
$\neg$	1
$\wedge$	2
$\vee$	3
$\rightarrow$	4
$\leftrightarrow$	5

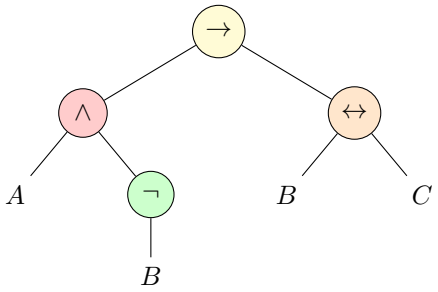
In absence of parenthesis the above priorities are applied. Therefore the formula $p \wedge q \rightarrow r$ is considered to be $(p \wedge q) \rightarrow r$. If we want to refer to the other formula we have to explicitly add parenthesis.

A formula can be seen as a tree. The leaves of the tree are propositional variables contained in the formula and the intermediate (non-leaf) nodes are associated to connectives that are used in order to build the complex formula.

EXAMPLE 2.3 (Tree of the formula). *The tree of the formula*

$$A \wedge \neg B \rightarrow (B \leftrightarrow C)$$

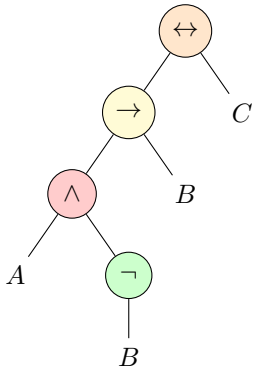
is the following:



Notice that, in order to force that the $\leftrightarrow$ connective should be applied before the $\rightarrow$ connective we have to use parenthesis. Indeed the tree of the formula without parenthesis, i.e.,

$$A \wedge \neg B \rightarrow B \leftrightarrow C$$

is the following:



Informally, a subformula is a part of a wff which is itself a wff. More formally the best approach is via the idea of the constructional history for a wff, with the subformulae being the wffs that appear in the history. Any decent textbook will explain this, and explain how you set out constructional histories as trees.

The set of formulas which need to be constructed in order to build a complex formula ϕ are called the *subformulas of* ϕ. The trees of the subformulas of a formula ϕ are the sub-trees of the tree of ϕ. A formal definition of sub-formula of ϕ is the following:

DEFINITION 2.2.

- *A is a* subformula *of itself*
- *A and B are* subformulas *of $A \wedge B$, $A \vee B$ $A \supset B$, e $A \equiv B$*
- *A is a subformula of $\neg A$*
- *if A is a subformula of B and B is a subformula of C, then A is a subformula of C.*
- *A is a* proper subformula *of B if A is a subformula of B and A is different from B.*

EXAMPLE 2.4. *The subformulas of $(p_0 \vee \neg p_1) \rightarrow (p_2 \wedge p_1)$ are: $(p_0 \vee \neg p_1) \rightarrow (p_2 \wedge p_1)$, $p_0 \wedge \neg p_1$, $p_2 \wedge p_1$, p_0, $\neg p_1$, p_2 and p_1.*

NOTATION 2.1. *Unfortunately, books on mathematical logic use widely varying notation for the Boolean operators; furthermore, the operators appear in programming languages with a different notation from that used in mathematics textbooks. The following table shows some of these alternate notations.*

Connective	*Alternative*	*programming languages*
$\neg$	$\sim$	`~ ! -`
$\wedge$	$\&$	`& && ^`
$\vee$	$\|\|$	`| ||`
$\rightarrow$	$\supset$ $\Rightarrow$	`=>, ->`
$\leftrightarrow$	$\Leftrightarrow$ $\equiv$	`<=>, <->, = %`

3. Interpretation of propositional formulas

Informally speaking an interpretation of a propositional language represents a state of affairs that allows one to establish for every proposition if it *true* or *false*, or equivalently if it *holds* or it *does not hold*. So for instance if we have the set of $\mathcal{P}$ are $\{p, q, r\}$ and they denote the propositions as described in Example 2.2, an interpretation correspond to a specific situation in which for instance Paola is painting, outside it is not raining, and Renzo is not playing piano. Since in PL we have that the truth value of complex propositions is fully determined by the truth value of the simplest (aka atomic) propositions, an interpretation can be specified by saying if p is holds or does not hold for every $p \in \mathcal{P}$. Let's now define this notion formally:

DEFINITION 2.3. *An* interpretation *of a propositional language on the set of propositional variables $\mathcal{P}$, is a function $\mathcal{I} : \mathcal{P} \rightarrow \{True, False\}$.*

Alternative notation for *True* and *False* that we will unse in this notes and are used in other books are 0 and 1, $\top$ and $\bot$. An alternative and equivalent definition of interpretation is the following

DEFINITION 2.4. *An* interpretation *of the set of propositional variables $\mathcal{P}$ is any subset $\mathcal{I} \subseteq \mathcal{P}$.*

The two definitions are equivalent under the correspondence

$$\mathcal{I}(p) = \mathit{True} \text{ if and only if } p \in \mathcal{I} \mathcal{I}(p) = \mathit{False} \text{ if and only if } p \notin \mathcal{I}$$

We will use alternatively both the definition depending on convenience. Notice that if $\mathcal{P}$ contains n propositional variables, (i.e, $|\mathcal{P}| = n$) then there are 2^n distinct interpretations. This corresponds to the fact that a set $\mathcal{P}$ that contains n elements has 2^n distinct subsets.

EXAMPLE 2.5. *Suppose that the set $\mathcal{P}$ of propositional variaables is equal to $\{p, q, r\}$, then there are $2^3 = 8$ propositional interpretations of $\mathcal{P}$. The following table reports all of them in the functional form $\mathcal{I} : \mathcal{P} \to \{\mathit{True}, \mathit{False}\}$ and in the setwise form $\mathcal{I} \subseteq \mathcal{I}$.*

	Functional form			*Set theoretic*
	p	q	r	*form*
$\mathcal{I}_1$	*True*	*True*	*True*	$\{p, q, r\}$
$\mathcal{I}_2$	*True*	*True*	*False*	$\{p, q\}$
$\mathcal{I}_3$	*True*	*False*	*True*	$\{p, r\}$
$\mathcal{I}_4$	*True*	*False*	*False*	$\{p\}$
$\mathcal{I}_5$	*False*	*True*	*True*	$\{q, r\}$
$\mathcal{I}_6$	*False*	*True*	*False*	$\{q\}$
$\mathcal{I}_7$	*False*	*False*	*True*	$\{r\}$
$\mathcal{I}_8$	*False*	*False*	*False*	$\{\}$

The truth value for propositional letters assigned by an interpretation $\mathcal{I}$ one can define when a formula is true (or holds) in the interpretation $\mathcal{I}$.

DEFINITION 2.5 ($\mathcal{I}$ satisfies a formula A, $\mathcal{I} \models A$). *A formula A is* satisfied by *an interpretation $\mathcal{I}$, in symbols*

$$\mathcal{I} \models A$$

according to the following inductive definition:

- *If $p \in \mathcal{P}$, $\mathcal{I} \models p$ if $\mathcal{I}(p) = \mathit{True}$;*
- *$\mathcal{I} \models \neg A$ if not $\mathcal{I} \models A$ (also written $\mathcal{I} \not\models A$);*
- *$\mathcal{I} \models A \wedge B$ if, $\mathcal{I} \models A$ and $\mathcal{I} \models B$;*
- *$\mathcal{I} \models A \vee B$ if $\mathcal{I} \models A$ or $\mathcal{I} \models B$;*
- *$\mathcal{I} \models A \to B$ if either $\mathcal{I} \not\models A$ or $\mathcal{I} \models B$;*
- *$\mathcal{I} \models A \leftrightarrow B$ if $\mathcal{I} \models A$ iff $\mathcal{I} \models B$.*

If $\mathcal{I}$ is an interpretation and A a formula, following expressions has the same meaning

- $\mathcal{I} \models A$;
- $\mathcal{I}$ satisfies A;
- A is true in $\mathcal{I}$;
- A holds in $\mathcal{I}$;
- A is satisfied by $\mathcal{I}$;
- $\mathcal{I}(A) = \mathit{True}$.

Notice that, if we have to check if a formula A is true in an interpretation $\mathcal{I}$, we have to take into account only the assignments that $\mathcal{I}$ does to the propositional variables that occours in A. This fact isi reflected by the following property:

PROPOSITION 2.1. *For every pair of interpretations $\mathcal{I}$ and $\mathcal{I}'$, if $\mathcal{I}(p) = \mathcal{I}'(p)$ for all the propositional variables p of a formula A, then $\mathcal{I} \models A$ iff $\mathcal{I}' \models A$*

In other words if $\mathcal{I}$ and $\mathcal{I}'$ differs only on the propositional variables that do not appear in A, then the truth values of A in $\mathcal{I}$ and $\mathcal{I}'$ are equal. As a consequence, to check if $\mathcal{I} \models A$ it is enough to consider the truth values that $\mathcal{I}$ assigns to the propositional variables appearing in A.

3.1. Material Implication. Some discussion is necessary about the semantics of implication $\rightarrow$. The operator $\rightarrow$ is called *material implication*; p is the *antecedent* and q is the *consequent*. Material implication does not claim causality; that is, it does not assert there the antecedent causes the consequent (or is even related to the consequent in any way). A material implication merely states that if the antecedent is true the consequent must be true; so it can be falsified only if the antecedent is true and the consequent is false. This definition of the conditions under which $a \rightarrow b$ is true is easily acceptable when a (the premise of the implication) is true, but when a is false, according to the definition, we have that $a \rightarrow b$ is true. In other words according to the condition given above we have that

$$\mathcal{I} \models a \rightarrow b \text{ if and only if } \mathcal{I} \models \neg a \vee b$$

so in other words $a \rightarrow b$ and $\neg a \vee b$ are equivalent propositions. This is sometimes very unintuitive. Consider for instance the statement:

the moon is made of cheese implies that the earth is flat

which can be represented by the propositional formula

$$c \rightarrow f$$

with c representing the proposition "the moon is made of cheese" and f the proposition "the earth is flat". According to the formal sematnics we have that $c \rightarrow f$ is true in the current state of the world (as we know that the premise is false), however the sentence in natural language suggests some cauation between the premise c and the conclusion f, which is not formalized by the implication. Other paradixical formulas are the following. They are true in all interpretation, however they are not intuitively acceptable:

- $(\neg p \wedge p) \rightarrow q$: p and its negation imply q. This is the paradox of entailment.
- $p \rightarrow (q \rightarrow p)$: if p is true then it is implied by every q.
- $\neg p \rightarrow (p \rightarrow q)$: if p is false then it implies every q. This is referred to as 'explosion'. In these cases, the statement $p \rightarrow q$ is said to be vacuously true.
- $p \rightarrow (q \vee \neg q)$: either q or its negation is true, so their disjunction is implied by every p.
- $(p \rightarrow q) \vee (q \rightarrow r)$: if p, q and r are three arbitrary propositions, then either p implies q or q implies r. This is because if q is true then p implies it, and if it is false then q implies any other statement. Since r can be p, it follows that given two arbitrary propositions, one must imply the other, even if they are mutually contradictory. For instance, "Nadia is in Barcelona implies Nadia is in Madrid, or Nadia is in Madrid implies Nadia is in Barcelona."is always true. This sounds like nonsense in ordinary discourse.

- $\neg(p \to q) \to (p \wedge \neg q)$: if p does not imply q then p is true and q is false. N.B. if p were false then it would imply q, so p is true. If q were also true then p would imply q, hence q is false. This paradox is particularly surprising because it tells us that if one proposition does not imply another then the first is true and the second false.

Suggestion: As a practice in order to avoid these types of paradoxes it is better to read $a \to b$ directly as $\neg a \vee b$.

3.2. The low of excluded middle. The formula $p \vee \neg p$ is valid (or equivalently it is a tautology). Indeed independently from the truth value assigned to p by an interpretation $\mathcal{I}$, we have that $\mathcal{I} \models p \vee \neg p$.

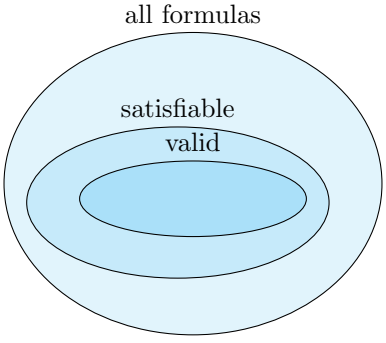
This tautology, called the *law of excluded middle*, is a direct consequence of our basic assumption that a proposition is a statement that is either true or false. Thus, the logic we will discuss here, so-called Aristotelian logic, might be described as a "2-valued" logic, and it is the logical basis for most of the theory of modern mathematics, at least as it has developed in western culture. There is, however, a consistent logical system, known as constructivist, or intuitionistic, logic which does not assume the law of excluded middle. This results in a "3-valued" logic in which one allows for a third possibility, namely,"other". In this system proving that a statement is "not true" is not the same as proving that it is "false", so that indirect proofs, which we shall soon discuss, would not be valid. If you are tempted to dismiss this concept you should be aware that there are those who believe that in many ways this type of logic is much closer to the logic used in computer science than Aristotelian logic. You are encouraged to explore this idea: there is plenty of material to be found in your library or through the worldwide web.

3.3. Valid, Satisfiable, and unsatisfiable formulas.

DEFINITION 2.6. *A formula A is*

- Valid *if* for all interpretations $\mathcal{I}$, $\mathcal{I} \models A$. *The notation $\models A$ (with no interpretation in the front) stands for "A is valid";*
- Satisfiable *if* there is an interpretations $\mathcal{I}$ *s.t.,* $\mathcal{I} \models A$
- Unsatisfiable *if* for no interpretations $\mathcal{I}$, $\mathcal{I} \models A$

Validity, satisfiable, and unsatisfiability are not independent concepts, they are related one another. This relation is highlighted in the following diagram



Stating that

$$\text{Valid formulas} \subset \text{Satisfiable formulas} \subset \text{Well formed formulas}$$

This implies that if a formula is valid it is also satisfiable, and if a formula it is unsatisfiable (not satisfiable) it is also not valid. Alternative terms for valid, satisfiable and unsatisfiable formula are the following:

- *tautology* is a synonym of *valid*;
- *contingency* is a synonym of *satisfiable*;
- *contradiction* is a synonym of *unsatisfiable*.

EXAMPLE 2.6.

Satisfiable	$A \to A$	*Valid*
	$A \vee \neg A$	
	$\neg\neg A \equiv A$	
	$\neg(A \wedge \neg A)$	
	$A \wedge B \to A$	
	$A \to A \vee B$	
	$p \vee q$	*Non Valid*
	$p \to q$	
	$\neg(p \vee q) \to r$	
Uatisfiable	$A \wedge \neg A$	
	$\neg(A \to A)$	
	$A \equiv \neg A$	
	$\neg(A \equiv A)$	

Prove that the blue formulas are valid, that the magenta formulas are satisfiable but not valid, and that the red formulas are unsatisfiable.

In the following we provide the proof for some formulas and leave the others by exercize.

- *$A \to A$ is valid. By Definition 2.5, $\mathcal{I} \models A \to A$ if and only if either $\mathcal{I} \models A$ or $\mathcal{I} \not\models A$, which holds for every $\mathcal{I}$.*
- *$A \vee \neg A$ is valid. By Definition 2.5, $\mathcal{I} \models A \vee \neg A$ if and only if $\mathcal{I} \models A$ or $\mathcal{I} \models \neg A$. Furhtermore, by Definition 2.5, $\mathcal{I} \models \neg A$ if and only if $\mathcal{I} \not\models A$. So $\mathcal{I} \models A \vee \neg A$ iff $\mathcal{I} \models A$ or $\mathcal{I} \not\models A$, which is true for every $\mathcal{I}$.*
- *$\neg\neg A \equiv A$ is valid. Suppose by contradiction that there is an $\mathcal{I}$ such that $\mathcal{I} \models \neg\neg A$ and $\mathcal{I} \not\models A$. By Definition 2.5, If $\mathcal{I} \models \neg\neg A$ then $\mathcal{I} \not\models \neg A$. Furtherore, again by Definition 2.5, $\mathcal{I} \not\models A$ then $\mathcal{I} \models \neg A$. So by assuming that there is an interpretation $\mathcal{I}$ that satisfies $\neg\neg A$ and does not satisfy A we reach a contraddiction that $\mathcal{I} \models \neg A$ and $\mathcal{I} \not\models \neg A$. Therefore there is no such an $\mathcal{I}$. This implies that for all interpretations $\mathcal{I} \models \neg\neg A$ if and obnly if $\mathcal{I} \models A$, and due to Definition 2.5, $\mathcal{I} \models \neg\neg A \equiv a$ for all $\mathcal{I}$.*
- *$p \vee q$ is satisfiable. To show that a formula is satisfiable it is sufficient to find an interpretation that makes it true. Let $\mathcal{I}$ be such that $\mathcal{I}(p) = True$, then (independently on the interpretation of q) we have that according to Definition 2.5 $\mathcal{I} \models p \vee q$.*
- *$p \vee q$ is not valied. To show that a formula is not valid we have to find an interpretation that does not satisfy it. Consider the interpretation $\mathcal{I}$ with $\mathcal{I}(p) = False$ and $\mathcal{I}(q) - False$. We have that by Definitin 2.5 $\mathcal{I} \not\models p \vee q$.*
- *$A \wedge \neg A$ is unsatisfiable. We have to prove that for all interpretation s $\mathcal{I}$, $\mathcal{I} \not\models A \wedge \neg A$. Suppose by contraddiction that there is an interpretation such that $\mathcal{I} \models A|wedge\neg A$, then by Definition 2.5 $\mathcal{I} \models A$ and $\mathcal{I} \models \neg A$. The last fact implies that $\mathcal{I} \not\models A$. Since by assuming that there is an $\mathcal{I}$ that satisfies $A \wedge \neg A$ we reach a contraddiction that $\mathcal{I} \models A$ and $\mathcal{I} \models \neg A$,*

we can conclude that there is no such $\mathcal{I}$, and therefore that $A \wedge \neg A$ is unsatisfiable.

DEFINITION 2.7 (Models of a formula). *For every formula A the set* $\text{models}(A)$, the models of A, *is the set $\{\mathcal{I} \mid \mathcal{I} \models A\}$, i.e., the set of truth assignments (interpretations) to the propositional variables* $\text{prop}(A)$ *that satisfy A;*

The sef of models of a complex formula can be computed in terms of the models of its direct subformulas. However, in computing the models of the subformulas we have to take into account the assignments to the proposition of the entire formulas. For this reason we introduce a generalization of the above definition

DEFINITION 2.8. *If $\mathcal{P}$ is a set of propositional variables that contains* $\text{prop}(A)$ *then* $\text{models}_{\mathcal{P}}(A)$ *is the set of assignments to $\mathcal{P}$ that satisfies A.*

Notice that the following facts holds

- if A is satisfiable $\text{models}(A) \neq \emptyset$;
- if A is valide $\text{models}(A) = 2^{\text{prop}(A)}$, i.e., the set of all interpretations of $\text{prop}(A)$;
- if A is unsatisfiable then $\text{models}(A) = \emptyset$.
- $\text{models}(\neg A) = 2^{\text{prop}(A)} \setminus \text{models}(A)$;
- $\text{models}(A \wedge B) = \text{models}_{\text{prop}(A \wedge B)}(A) \cap \text{models}_{\text{prop}(A \wedge B)}(B)$;
- $\text{models}(A \vee B) = \text{models}_{\text{prop}(A \vee B)}(A) \cup \text{models}_{\text{prop}(A \vee B)}(B)$;
- $\text{models}(A \to B) = \text{models}_{\text{prop}(A \to B)}(\neg A) \cup \text{models}_{\text{prop}(A \to B)}(B)$
- $\text{models}(A \equiv B = \text{models}_{\text{prop}(A \equiv B)}(A) \cap \text{models}_{\text{prop}(A \equiv B)}(B) \cup \models (\neg A) \cap \models$

Validity, satisfiability, and unsatisfiability of a formula A is also related to validity, satisfiability and unsatisfiability of its negation, i,e. $\neg A$.

PROPOSITION 2.2.

(1) A is valid if and only if $\neg A$ is unsatisfiable;
(2) A is satisfiable if and only if $\neg A$ is not valid;
(3) A is not valid if $\neg A$ is satisfiable;
(4) A is unsatisfiable if $\neg A$ is valid.

PROOF. We prove the first two points and left the other two by exercize.

(1) A is valid then $\text{models}(A) = 2^{\mathcal{P}}$. Since $\text{models}(\neg A) = 2^{\mathcal{P}} \setminus \text{models}(A)$ we have that $\text{models}(\neg A) = \emptyset$, and therefore $\neg A$ is unsatisfiable.
(2) A is satisfiable then $\text{models}(A) \neq \emptyset$. and therefore $\text{models}(\neg A) = 2^{\mathcal{P}} \setminus \text{models}(A) \neq 2^{\mathcal{P}}$ which means that $\neg A$ is not valid.

□

The definition of satisfiability, validity, and unsatisfiability can be extended also to sets of formulas as follows:

DEFINITION 2.9. *A set of formulas Γ is*

- Valid *if for all interpretations $\mathcal{I}$, $\mathcal{I} \models A$* for all formulas $A \in \Gamma$
- Satisfiable *if there is an interpretations $\mathcal{I}$, $\mathcal{I} \models A$* for all $A \in \Gamma$
- Unsatisfiable *if for no interpretations $\mathcal{I}$,, s.t. $\mathcal{I} \models A$* for all $A \in \Gamma$

PROPOSITION 2.3. *For any* finite set *of formulas Γ, (i.e., $\Gamma = \{A_1, \ldots, A_n\}$ for some $n \geq 1$), Γ is valid (resp. satisfiable and unsatisfiable) if and only if $A_1 \wedge \cdots \wedge A_n$ is valid (resp, satisfiable and unsatisfiable).*

We leave the proof of the previous proposition by exercize.

4. Logical consequence

Logical consequence is one of the key notion of every logic. It is the base of correct inference. Intuitively a proposition, called consequence, logically follows from (or equivalently, is a logical consequence of) a set of propositions, called premises or hypothesis, if from the assumption that the hypothesis are true we can conclude with cartainty that the consequence is also true. For instance if $x > y$ and $y > z$ we can conclude that $x - z > 0$. Logical consequence is strictly connected with the definition of truth of a formula in an interpretation. One of the main motivation of using logic is to make rigorous what is a "valid argument", i.e., when one fact follows form some other facts. Intuitively a "valid argument" is the one that derive true facts from true facts. To this aim, we use the notion of *logical consequence* (Some books refer to logical implication and entailment.)

DEFINITION 2.10. *A formula A is a logical consequnece of a set of formulas Γ, in symbols $\Gamma \models A$, if for all interpretations $\mathcal{I}$, if $\mathcal{I} \models C$ for all $C \in \Gamma$ then $\mathcal{I} \models B$. If Γ contains is a finite set of formulas $\{A_1, \ldots, A_n\}$, then we use the notation*

$$A_1, \ldots, A_n \models A$$

to denote that A is a logical consequence of Γ.

In other words, $\Gamma \models A$ means that whenever all the wffs in Γ are true, then A must also be true. Notice that this definition does not say anything about interpretations under which one or more of the wffs in Γ are false. In this case, we don't care whether A is true or false.

EXAMPLE 2.7. *q is a logical consequence of $\{p \to q, \neg p \to q\}$. In symbols $p \to q, \neg p \to q \models q$. To check this, since we have to look to all the interpretations we can build a need a truth table.*

	p	q	$p \to q$	$\neg p \to q$
$\mathcal{I}_1$	*True*	*True*	*True*	*True*
$\mathcal{I}_2$	*True*	*False*	*False*	*True*
$\mathcal{I}_3$	*False*	*True*	*True*	*True*
$\mathcal{I}_4$	*False*	*False*	*True*	*False*

$p \to q, \neg p \to q \models q$ holds since all the interpretations that satisfy $p \to q$ and $\neg p \to \neg q$ (i.e.,$\mathcal{I}_1$ and $\mathcal{I}_3$) also satisfy q.

NOTATION 2.2. *Given a non empty finite set of formulas $\Gamma = \{A_1, \ldots, A_n\}$*

$$\bigwedge \Gamma \text{ or equivalently } \bigwedge_{i=1}^{n} A_i$$

which stands for $A_1 \wedge A_2 \wedge \cdots \wedge A_n$

$$\bigvee \Gamma \text{ or equivalently } \bigvee_{i=1}^{n} A_i$$

which stands for $A_1 \vee A_2 \vee \cdots \vee A_n$

When Γ is empty we extend the notation as follows

$$\bigwedge \Gamma \text{ denotes } \top$$

and

$$\bigvee \Gamma \text{ denotes } \bot$$

4.1. Properties of propositional logical consequence.

PROPOSITION 2.4. *If Γ and Σ are two sets of propositional formulas and A and B two formulas, then the following properties hold:*

Reflexivity:: $\{A\} \models A$
Monotonicity:: *If* $\Gamma \models A$ *then* $\Gamma \cup \Sigma \models A$
Cut:: *If* $\Gamma \models A$ *and* $\Sigma \cup \{A\} \models B$ *then* $\Gamma \cup \Sigma \models B$
Deduction theorem:: *If* $\Gamma, A \models B$ *then* $\Gamma \models A \to B$
Refutation principle": $\Gamma \models A$ *iff* $\Gamma \cup \{\neg A\}$ *is unsatisfiable*

PROOF.

Reflexifity: For all $\mathcal{I}$ if $\mathcal{I} \models \{A\}$ then $\mathcal{I} \models A$.
Monotonicity: For all $\mathcal{I}$ if $\mathcal{I} \models \Gamma \cup \Sigma$, then $\mathcal{I} \models \Gamma$, by hypothesis ($\Gamma \models A$) we can infer that $\mathcal{I} \models A$, and therefore that $\Gamma \cup \Sigma \models A$
Cut:: For all $\mathcal{I}$, if $\mathcal{I} \models \Gamma \cup \Sigma$, then $\mathcal{I} \models \Gamma$ and $\mathcal{I} \models \Sigma$. The hypothesis $\Gamma \models A$ implies that $\mathcal{I} \models A$. Since $\mathcal{I} \models \Sigma$, then $\mathcal{I} \models \Sigma \cup \{A\}$. The hypothesis $\Sigma \cup \{A\} \models B$, implies that $\mathcal{I} \models B$. We can therefore conclude that $\Gamma \cup \Sigma \models B$.
Deduction theorem: Suppose that $\mathcal{I} \models \Gamma$. If $\mathcal{I} \not\models A$, then $\mathcal{I} \models A \to B$. If instead $\mathcal{I} \models A$, then by the hypothesis $\Gamma, A \models B$, implies that $\mathcal{I} \models B$, which implies that $\mathcal{I} \models B$. We can therefore conclude that $\mathcal{I} \models A \to B$.
Refutation principle: ($\Longrightarrow$) Suppose by contradiction that $\Gamma \cup \{\neg A\}$ is satisfiable. This implies that there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \Gamma$ and $\mathcal{I} \models \neg A$, i.e., $\mathcal{I} \not\models A$. This contradicts that fact that for all interpretations that satisfies Γ, they satisfy A ($\Longleftarrow$) Let $\mathcal{I} \models \Gamma$, then by the fact that $\Gamma \cup \{\neg A\}$ is unsatisfiable, we have that $\mathcal{I} \not\models \neg A$, and therefore $\mathcal{I} \models A$. We can conclude that $\Gamma \models A$.

□

The above property has an important impact in using propositinal logic for representing the knowledge of an artificial agent. In particular the *monotonicity* property, states that by adding new knowledge you never "delete" the old knowledge. For instance if an agent represent the fact that all birds flies, with the implication $\mathsf{bird} \to \mathsf{flies}$, and the fact that a penguin is a bird with the implication $\mathsf{penguin} \to \mathsf{bird}$, then this automatically implies that $\mathsf{penguin} \to \mathsf{flies}$. But we know that penguins do not fly, Humans adjust this problem by providing the additional knowledge that penguins are exceptions and therefore they don't fly. This is not possible in propositional logic, since if we add the fact that $\mathsf{penguin} \to \mathsf{exceptionalBird}$, and $\mathsf{exceptionalBird} \to \neg\mathsf{fly}$, we don't delete the fact that a penguin is a bird, and therefore we still derive that penguins fly. In order to cope with this type of representation problem, researchers in knowledge representation in AI introduces "non monotonic" logics Brewka 1989.

The *deduction theorem* states that logical consequence that involves a finite set of formulas can be "internalized" in an implications. Indeed an immediate consequence of the deduction theroem is that $A_1, \ldots, A - n \models A$ implies that $A_1 \to (A_2 \to \ldots \to (A_{n-1} \to A_n) \ldots))$ is a valid formulas.

The *Refutation principle* is also a very important property, that allows to transform the problem of checking logical consequence in to the problem of checking satisfiability of a set of formulas. We will explain this method in the next chapter that is dedicated to algorithms for checking satisfiability.

Another inportan property of the logical consequence in propositional logic is the so called *compactness theorem*n

THEOREM 2.1 (Compactness). *If* $\Gamma \models A$ *then* $\Gamma_0 \models A$ *for some finite subset* $\Gamma_0 \subseteq \Gamma$.

We report one proof of the compactness theorem at the end of this chapter.

5. Logical equivalence

DEFINITION 2.11. *Two formulas* A *and* B *are* logically equivalent *if and only if they are true under the same set of interpretations. Alternatively* A *and* B *are equivalent if* $[[A]] = [[B]]$.

With an abuse of notation, we write $A \equiv B$ to state that A is logically equivalent to B. Let us now analyse the different type of equivalence formulas we have in PL.

5.0.1. *Absorption of* $\top$ *and* $\bot$. Remember that $\top$ and $\bot$ are usually added to the language of propositional logic, and they are mapped to *True* and *False* respectively by all the interpretations. The appearance of these constants in a formula can collapse the formula so that the binary operator is no longer needed; it can even make a formula become a constant whose truth value no longer depends on the non-constant sub-formula.

$$\begin{aligned} A \vee \top &\equiv \top & A \vee \bot &\equiv A \\ A \wedge \top &\equiv A & A \wedge \bot &\equiv \bot \\ A \to \top &\equiv \top & A \to \bot &\equiv \neg A \\ \top \to A &\equiv A & \bot \to A &\equiv \top \\ A \leftrightarrow \top &\equiv A & A \leftrightarrow \bot &\equiv \neg A \end{aligned}$$

5.0.2. *Identical Operands.* Collapsing can also occur when both operands of an operator are the same or one is the negation of another.

$$\begin{aligned} A &\equiv \neg\neg A \\ A \wedge A &\equiv A \\ A \vee A &\equiv A \\ A \wedge \neg A &\equiv \bot \\ A \vee \neg A &\equiv \top \\ A \to A &\equiv \top \\ A \leftrightarrow A &\equiv \top \end{aligned}$$

5.0.3. *Commutativity, Associativity and Distributivity.* The binary Boolean operators are commutative, except for implication.

$$\begin{aligned} A \vee B &\equiv B \vee A \\ A \wedge B &\equiv B \wedge A \\ A \leftrightarrow B &\equiv B \leftrightarrow A \end{aligned}$$

If negations are added, the direction of an implication can be reversed:

$$A \to B \equiv \neg B \to \neg A$$

The formula $\neg B \to \neg A$ is called *the contrapositive* of $A \to B$. Disjunction, conjunction, equivalence are associative.

$$A \vee (B \vee C) \equiv (A \vee B) \vee C$$
$$A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C$$
$$A \leftrightarrow (B \leftrightarrow C) \equiv (A \leftrightarrow B) \leftrightarrow C$$

Disjunction and conjunction distribute over each other

$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$$
$$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$$

5.0.4. *De Morgan laws.* Negating a conjunction results in a disjunction of the negated conjuncts, and viceversa

$$\neg(A \wedge B) \equiv \neg A \vee \neg B$$
$$\neg(A \vee B) \equiv \neg A \wedge \neg B$$

5.0.5. *Distribution of implication.*

$$A \to B \vee C \equiv (A \to C) \vee (B \to C)$$
$$A \to B \wedge C \equiv (A \to C) \wedge (B \to C)$$
$$A \to (B \to C) \equiv (A \to B) \to (A \to C)$$

5.0.6. *The Relationship Between $\leftrightarrow$ and logical equivalence.* Equivalence, $\leftrightarrow$, is a Boolean operator in propositional logic and can appear in formulas of the logic. Logical equivalence, instead, is a relations between formulas. There is potential for confusion because we are using a similar vocabulary both for the object language, in this case the language of propositional logic, and for the metalanguage that we use reason about the object language. Equivalence and logical equivalence are, nevertheless, closely related as shown by the following theorem:

THEOREM 2.2. *A is logically equivalent to B if and only if $A \leftrightarrow B$ is valid*

PROOF. if A is logically equivalent to B if and only if $[[A]] = [[B]]$ which is true if and only if for all $\mathcal{I}$, $\mathcal{I} \models A$ whenever $\mathcal{I} \models B$ and viceversa, and i.e., $\mathcal{I} \models A \leftrightarrow B$ for all $\mathcal{I}$, which holds iff $A \leftrightarrow B$ is valid. □

6. Truth tables

Truth tables is a simple method for explicitly enumerating all the interpretations of the propositional variables of a formula A and for each interpretation it reports the corresponding truth value of A. The truth table for a propositional formula is a table containint as many columns as the propositional variables occurring in A plus the number of subformulas of A. The truth table of A contains one row for every interpretation of $\mathrm{prop}(A)$, where $\mathrm{prop}(A)$ is the set of propositional variables that occours in A. Therefore it contains $2^{|\mathrm{prop}(A)|}$ rows (where $|X|$ denote thor e cardinality of a set X, i.e., the number of elements that belongs to X). A raw of a truth table corresponds to an interpretation $\mathcal{I}$ i.e., an assignment of the propositional variables of A. The first n elements of the raws are the assignments of the propositional variables, while the last element is the value of $\mathcal{I}(A)$.

EXAMPLE 2.8. *the truth table of the formula* $p \to (q \vee \neg r)$ *is the following*[2]

p	q	r	p	$\to$	$($	q	$\vee$	$\neg$	r	$)$
True	*True*	*True*	*True*	*True*		*True*	*True*	*False*	*True*	
True	*True*	*False*	*True*	*True*		*True*	*True*	*True*	*False*	
True	*False*	*True*	*True*	*False*		*False*	*False*	*False*	*True*	
True	*False*	*False*	*True*	*True*		*False*	*True*	*True*	*False*	
False	*True*	*True*	*False*	*True*		*True*	*True*	*False*	*True*	
False	*True*	*False*	*False*	*True*		*True*	*True*	*True*	*False*	
False	*False*	*True*	*False*	*True*		*False*	*False*	*False*	*True*	
False	*False*	*False*	*False*	*True*		*False*	*True*	*True*	*False*	

The truth table contains the columns corresponding to the propositinal variables of the formula $p \to (q \vee \neg r)$*, i.e.,* $\{p, q, r\}$ *and one column for every subformula of* $p \to (q \vee \neg r)$*, which are* p*,* q*,* r*,* $\neg r$*,* $q \vee \neg r$*, and* $p \to (q \vee \neg r$*. The truth values of the subformulas are computed starting from the simplest one to the most complex until you compute the truth value of the entire formula. (marked in red)*

With the truth table for A is is possible to check if A valid, satisfiable, or unsatisfiable. If all the values in the columns of A are True, then A is valied, if there is at least one true then A is satisfiable, and all the values are False, then A is unsatisfiable. Truth tables are computationally very expensive since they enumerate the interpretations of a formula, which are esponentially large w.r.t., the size of the formula. Therefore they are only theoretical and pedagocical objects, in practice, (in real application where the number of propositional variables are large) you will never explicitly compute a truth table.

It is possible to build a truth table for more than one formula, by simply adding on the right one column for each formula. Sometimes it is also convenient to add columns corresponding to the subformulas of a complex formula. For instance if we have to compute the truth table of the formula

$$(F \vee G) \wedge \neg(F \wedge G)$$

we build a truth table for all its subformulas, as follows:

F	G	$F \vee G$	$F \wedge G$	$\neg(F \wedge G)$	$(F \vee G) \wedge \neg(F \wedge G)$
True	*True*	*True*	*True*	*False*	*False*
True	*False*	*True*	*False*	*True*	*True*
False	*True*	*True*	*False*	*True*	*True*
False	*False*	*False*	*False*	*True*	*False*

Exercise 1:
Use the truth tables method to determine whether

$$(p \to q) \vee (p \to \neg q)$$

is valid.

Solution

[2]To generate the truth table automatically I have used the web application availabe at `https://mrieppel.net/prog/truthtable.html`

p	q	$p \to q$	$\neg q$	$p \to \neg q$	$(p \to q) \vee (p \to \neg q)$
True	*True*	*True*	*False*	*False*	*True*
True	*False*	*False*	*True*	*True*	*True*
False	*True*	*True*	*False*	*True*	*True*
False	*False*	*True*	*True*	*True*	*True*

The formula is valid since it is satisfied by every interpretation. □

7. Propositional Theories

In the commonsense a theory is a system that allows to describe must be true in a certain domain of interests. An example of theory, which we have studied in high school, is euclidean geometry. A more sophisticate theory is the quantum mechanics. etc. Theories have been developed by humans in order to describe precisely a phenomena, and to allow to perform correct inference in order to deduce the truth of "unknown" propositions starting from a handful of principles (axioms) that are accepted to be necessarily true. In a theory there are propositions that are known to be true, i.e., somebody manage to show that truth by providing a deduction, but there are also propositions that are unknown to be true. For instance in numbner theory that are many so called "open problems" [3] In artificial intellighence, one can use a logical theory to represent the knowledge of an agent about a particular domain.by means of a logical theory, and use automatic deduction in order to infer what is true and what is false in such a domain. This was one of the original proposal of one of the founders of AI (John Mc Carthy McCarthy 1959.

A logical theory is nothing more than a set of sentences that expressews the propositions that must be true in all the configurations of the domain of interest that we believe to be possible.

Definition 2.12 (Propositional theory). *A theory is a set of propositional formulas on a set of propositional variables $\mathcal{P}$ closed under the logical consequence relation. I.e. A set of formulas T is a theory $T \models A$ implies that $A \in T$.*

An alternative and equivalent definition of theory is the following.

Definition 2.13 (Propositional theory). *A theory is a set of propositional formulas on a set of propositional variables $\mathcal{P}$ that are true in a set of interpretations of $\mathcal{P}$.*

Example 2.9. *Let $\mathcal{P}$ be the set of propositional variables. The set T of valid formulas on the propositional variables $\mathcal{P}$, i.e., $T = \{A \in \text{wffs}(\mathcal{P}) \mid A \text{ is valid}\}$. This is equivalent to say T is the set of fromulas that are true in all the interpretations of $\mathcal{P}$. For instance if p, q, r are propositional variables of $\mathcal{P}$, The formulas $p \vee \neg p$, $q \vee \neg q$, $p \wedge q \to p$, $r \to r$ belongs to T. While the formulas $p \to q$ does not. Notice that T is closed under logical consequence. Indeed suppose that $A_1, \ldots, A_n \models A$, and $A_1, \ldots, A_n \in T$. Then we have that each A_i is valid, and therefore it is true in every models, The fact that $A_1, \ldots, A_n \models A$ entails that A is also treue in every interpretation, and therefore A is valied, hwnce it belongs to T.*

[3] An example of open problem in number theory is connected to the Erdös–Moser equation:

$$1^k + 2^k + \cdots + m^k = (m+1)^k,$$

where m and k are positive integers. The only known solution is $1^1 + 2^1 = 3^1$, and Paul Erd ös conjectured that no further solutions exist.

EXAMPLE 2.10. *is the set of formulas which are true in the the following three interpretations of* $\mathcal{P}$, $\{\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3\}$, *where* $\mathcal{I}_1 = \{p, q\}$, $\mathcal{I}_2 = \{q, r\}$ *and* $\mathcal{I}_2 = \{p, r\}$. *Notice that this theory contains all the valid formulas since they are true in all the interpretations, and therefore also in* $\mathcal{I}_1, \mathcal{I}_2$ *and* $\mathcal{I}_3$. *However* T *contains formulas which are not valid as for instance* $p \vee q$, $\neg p \to q \wedge r$. *Notice that* T *can be defined as the set of formulas that are logical consequences of the formula* $A = (p \wedge q) \vee (p \wedge r) \vee (q \wedge r)$. *Indeed if* B *is a logical consequence of* A, *it is true in all the models that satisfies* A. *Since* $\mathcal{I}_1, \mathcal{I}_2$ *and* $\mathcal{I}_3$ *satisfies* B, *then* $B \in T$. *We way that* A *is an* axiomatization *of* T, *because from the axiom* A *all the formulas of* T *follows logically.*

EXAMPLE 2.11. *Let* T *be the set of formulas that are true in a single interpretation* $\mathcal{I}$. T *is a theory since it is closed under logical consequence. Indeed if* $A_i \in T$ *we have that* $\mathcal{I} \models A_i$ *we have that* $\mathcal{I} \models A_i$, *and if* $A_1, \ldots, A_n \models A$, *we have that* $\mathcal{I} \models A$ *and therefore* $A \in T$. *This theory is* complete *in the sense that for every formula* A, *either* $A \in T$ *or* $\neg A \in T$.

The three examples of theories provided above, range from the weakest theory to the strongest one.

A propositional theory contains an infinite set of formulas. Indeed every theory contains all the propositional tautoloties, which are infinite. However thee infinite set of formulas could be defined as the logical consequences of a smaller set of formulas (possibily but not necessarily finite) such that all the formulas of the theory logically follows. These set of formulas are called *axioms* of a theory, They are the basic principles of the theory from which everything else that is true in the theory follows logically. So the set of axioms constitute a base for the theory and characterizes it.

DEFINITION 2.14. *A set* S *of formulas is a set of* axioms *(or equivalently an* axiomatization*) of a theory* T *if* $T = \{A \in \text{wffs}(\mathcal{P}) \mid S \models A\}$.

An important property for a set of axioms S of a theory T is that they are minimal, in the sense that no formulas in S is a logical consequences of other formulas in S. This property can be formulated also as follows: there is no $S' \subset S$ that is an axiomatization of T.

EXAMPLE 2.12. *The axiomatization of the theory of Example 2.9 is the empty set. An axiomatization of the theory of Example 2.11 is the set* $\{p \mid \mathcal{I} \models p\} \cup \{\neg p \mid \mathcal{I} \not\models p\}$

8. The Compactness Theorem

In this section we prove a fundamental result about propositional logic called the *Compactness Theorem*. This will play an important role in the second half of the course when we study predicate logic. This is due to our use of Herbrand's Theorem to reduce reasoning about formulas of predicate logic to reasoning about infinite sets of formulas of propositional logic. Before stating and proving the Compactness Theorem we need to introduce one new piece of terminology.

Recall that a set of formulas Γ is satisfiable if there is an assignment that satisfies every formula in Γ. For example, the set of formulas

$$\Gamma = \{p_1 \vee p_2, \neg p_2 \vee \neg p_3, p_3 \vee p_4, \neg p_4 \vee p_5, \ldots\}$$

on the infinite set of propositional variables $\mathcal{P} = \{p_1, p_2, p_2, \dots\}$ is satisfied by the assignment $\mathcal{I}$ defined as

$$\mathcal{I}(p_i) = \begin{cases} \text{True} & \text{if } i \text{ is odd} \\ \text{False} & \text{if } i \text{ is even} \end{cases}$$

DEFINITION 2.15. *A set of formula is* finitely satisfiable *if all its finite subsets are satisfiable.*

THEOREM 2.3 (Compactness Theorem). *A set of formulas Γ is satisfiable if and only if it is finitely satisfiable.*

Notice that the formulation above of the compactness theorem is is different fron the one introduced by Theorem 2.1. Nevertheless, if we proove Theorem 2.3, we can easily prove the original compactness theorem by combining the refutation principle, monotonicity and the the new formulaiton of the compactness theorem.

- $\Gamma \models A$ if and only if $\Gamma \cup \{\neg A\}$ is not satisfiable (by the refutation principle)
- $\Gamma \cup \{\neg A\}$ is not satisfiable if and only if there is a finite subset Γ_0 of $\Gamma \cup \{\neg A\}$ which is not satisfiable (new formulation of the compactness theorem).
- This implies that $\Gamma_0 \cup \{\neg A\}$ is not satisfiable, by monotonicity, and therefore by refutation principle that $\Gamma_0 \models A$.

To get an idea of what says the Compactness theorem consider the following intuitive example

EXAMPLE 2.13. *Suppose that you have a logical language in which you can express by means of formula K, P_i and S the following proposition:*

K	*there is a cake of finite size*
P_i	*The i-th person has a piece of cake for $i = 1, 2, 3, \dots s$*
S	*The pieces of cake have all the same non zero dimension*

The formula $K \wedge S \to P_i$ formalizes the fact that if there is a cake and it is equally divided then the ith person gets it's piece of cake. Notice consider any finite subset of the set $\Gamma = \{K \wedge S \to P_1, K \wedge S \to P_2, K \wedge S \to P_3, \dots\}$. i.e., for every finite set of natural numbers $I \subset \mathbb{N}$ let

$$\Gamma_I = \{K \wedge S \to P_i \mid i \in I\}$$

Notice that the whole Γ is not satisfiable, since you cannot cut a finite cake in an infinite set of slices of finite and constant size. However each Γ_I for every finite $I \subset \mathbb{N}$ is satisfiable. Since the compactness theorems holds in propositional logic, we can conclude that such a scenario cannot be formalized in propositional logic.

To prove the compactness theorems we first need to prove the following lemma:

LEMMA 2.1. *If Γ is finitely satisfiable then either $\Gamma \cup \{\phi\}$ or $\Gamma \cup \{\neg\phi\}$ is finitely satisfiable, for every formula ϕ.*

PROOF.
- Suppose the conclusion of the lemma does not hold: Both $\Gamma \cup \{\phi\}$ and $\Gamma \cup \{\neg\phi\}$ are not finitely satisfiable.
- Hence, there are two finite subsets Γ_1 and Γ_2 of Γ such that both $\Gamma_1 \cup \{\phi\}$ and $\Gamma_2 \cup \{\neg\phi\}$ are not satisfiable.
- Let us show that $\Gamma_1 \cup \Gamma_2$ does not have models, i.e., it is unsatisfiable.

- If $\mathcal{I}$ is a model of Γ_1, than it cannot be a model of ϕ, therefore it is a model of $\neg\phi$. Otherwise $\Gamma_1 \cup \{\phi\}$ would be satisfiable. Therefore, $\mathcal{I} \models \neg\phi$.
- Since $\Gamma_2 \cup \{\neg\phi\}$ is not satisfiable, then $\mathcal{I}$ cannot be a model of Γ_2.
- This implies that every model of Γ_1 is not a model of Γ_2 and therefore, $\Gamma_1 \cup \Gamma_2$ is not satisfiable.
- Since $\Gamma_1 \cup \Gamma_2$ is finite and it is a subset of Γ, then Γ cannot be finitely satisfiable. This contraddicts the hypothesis of the lemma, and therefore the lemma is proved.

□

Let us now prove the compactness theorem:

Of theorem 2.3. If Γ is satisfiable, then every subset of Γ is satisfiable, and therefore Γ is finitely satisfiable. The prove of the opposite direction is more complex. We have to show that if Γ is finitely satisfiable then the whole Γ is satisfiable, i.e., there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \Gamma$.

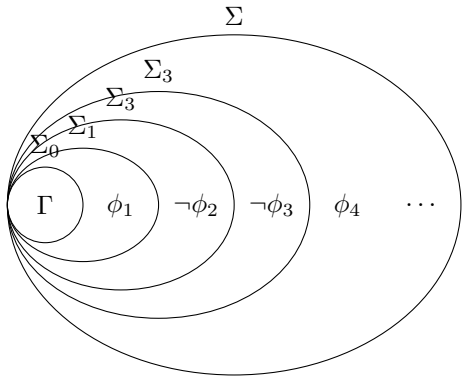
- let enumerate all the formulas $\phi_1, \phi_2, \phi_3, \dots$ of the language of Γ.
- let us define the sequence $\Sigma_0, \Sigma_1, \Sigma_2, \dots$ as follows

$$\Sigma_0 = \Gamma \qquad \Sigma_n = \begin{cases} \Sigma \cup \{\phi_n\} & \text{if } \Sigma_{n-1} \cup \{\phi_n\} \text{ is fin. sat.} \\ \Sigma \cup \{\neg\phi_n\} & \text{if } \Sigma_{n-1} \cup \{\neg\phi_n\} \text{ is fin. sat.} \end{cases}$$

- By induction, using previous lemma, Σ_i is finitely satisfiable;
- Let

$$\Sigma = \bigcup_{n \geq 0} \Sigma_i$$

The construction of Σ is shown in the following picture:



- By construction Σ is finitely satisfiable. Furthermore
  (1) For every formula ϕ either $\phi \in \Sigma$ or $\neg\phi \in \Sigma$ but not both.
  (2) For every $p \in \mathcal{P}$, $p \in \Sigma$ or $\neg p \in \Sigma$ but not both.
  (3) We define the interpretation $\mathcal{I}(p) = \begin{cases} \text{True} & \text{if } p \in \Sigma \\ \text{False} & \text{if } \neg p \in \Sigma \end{cases}$
  (4) Let us show that $\mathcal{I} \models \phi$ for all $\phi \in \Sigma$.
  (5) Consider the finite set Σ_i that contains ϕ and either p or $\neg p$ for all propositional variable p that occours in ϕ. Since ϕ contains only a finite set of propositional variables, such an finite Σ_i exists.

(6) Since Σ_i is finite, and Σ is finitely satisfiable, there is an interpretation $\mathcal{I}'$ that satisfies Σ_i, and therefore $\mathcal{I}' \models \phi$
(7) Furthermore $\mathcal{I}' \models p$ of $p \in \Sigma_i$ or $\mathcal{I}' \models \neg p$ if $\neg p$ in Σ_i.
(8) However by construction of $\mathcal{I}$, we have that $\mathcal{I}'$ and $\mathcal{I}$ agree on all the interpretations of all the propositional variables of ϕ and therefore $\mathcal{I} \models \phi$.
(9) Hence, $\mathcal{I} \models \Sigma$.
(10) Since $\Gamma \subset \Sigma$, then $\mathcal{I} \models \Gamma$.

□

9. Exercises

9.1. Formulas, subformulas, and other syntactic objects.

Exercise 2:
Decide which of the following phrases expresses a proposition

(1) The dog of my syster
(2) Is mario Happy?
(3) A lion in the forest
(4) A tiger is walking in the forest
(5) the sooner the better
(6) No more food please!
(7) Have a nice week end!
(8) The hause in which I was born
(9) Closed door
(10) The door is closed

Exercise 3:
Knowing that the precedence relations batween the propositional connectives is

$$\text{“}\neg\text{”} \prec \text{“}\wedge\text{”} \prec \text{“}\vee\text{”} \prec \text{“}\to\text{”} \prec \text{“}\leftrightarrow\text{”}$$

add the parenthesis to specify the correct parsing of the following formulas:

(1) $(\neg p \vee q) \wedge (q \to (\neg r \wedge \neg p)) \wedge (p \vee r)$
(2) $((\neg p \vee q) \to q \wedge (q \to r) \wedge \neg r) \to p$
(3) $\neg p \wedge (\neg q \vee r) \wedge (\neg p \to q \wedge \neg r)$

Solution

$$(\neg p \vee q) \wedge (q \to (\neg r \wedge \neg p)) \wedge (p \vee r)$$
$$((\neg p) \vee q) \wedge (q \to ((\neg r) \wedge (\neg p))) \wedge (p \vee r)$$

$$((\neg p \vee q) \to q \wedge (q \to r) \wedge \neg r) \to p$$
$$(((\neg p) \vee q) \to (q \wedge (q \to r) \wedge (\neg r))) \to p$$

$$\neg p \wedge (\neg q \vee r) \wedge (\neg p \to q \wedge \neg r)$$
$$(\neg p) \wedge ((\neg q) \vee r) \wedge ((\neg p) \to (q \wedge (\neg r)))$$

□

Exercise 4:
When we have two connectives which are the same, then we give precedence to the right one. I.e., $a \circ b \circ c$ reads as $a \circ (b \circ c)$ for every binary connective $\circ \in \{\to, \wedge, \vee, \leftrightarrow\}$. Put the right parentesist on the following formulas

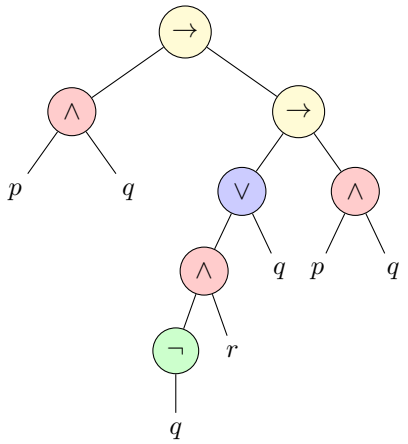
(1) $a \to b \wedge \neg c \to d$
(2) $a \to b \to c \to d$
(3) $a \leftrightarrow b \leftrightarrow c \wedge d \leftrightarrow e$

Exercise 5:
Draw the formula tree of the following formula:

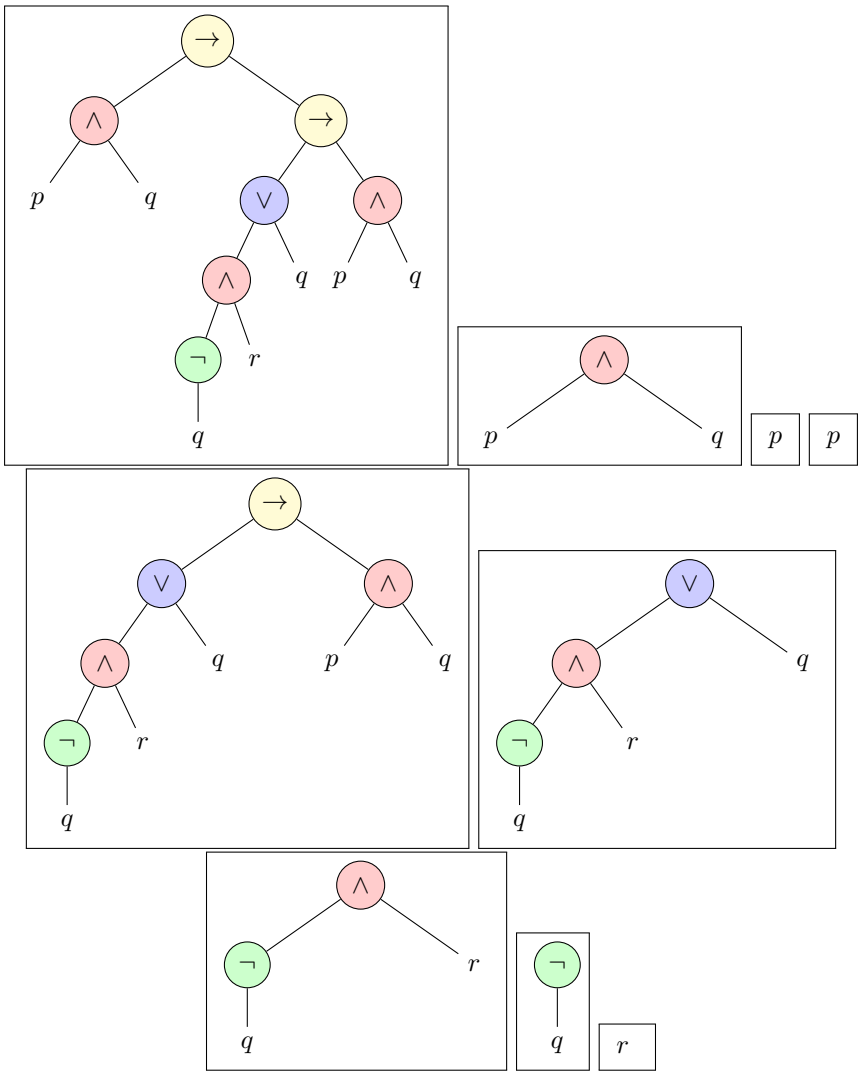
$$(p \wedge q) \to \neg q \wedge r \vee q \to (p \wedge q)$$

Rimember that the expression $a \to b \to c$ is parsed as $a \to (b \to c)$, and $a \leftrightarrow b \leftrightarrow c$ is parsed as $a \to (b \to c)$. Count all the sub-formulas of ϕ.

Solution The formula ϕ is parsed as $p \wedge q \to (((\neg q \wedge r) \vee q) \to q \wedge p)$ where parentesis are made explicit. The tree of ϕ is



In general a formula has as many subformulas as many distinct subtrees of its formula tree, including the entire tree and the leavers. Therefore ϕ has 9 subformula corresponding to the following subtrees:



□ **Exercise 6:**

Draw the formula tree for the following formulas and count the number of subformulas:

(1) $a \to b \to c \leftrightarrow a \wedge b \vee \neg a \wedge \neg b \vee \neg a$
(2) $\neg(a \leftrightarrow b \leftrightarrow c)$
(3) $\neg(a \wedge \neg(b \wedge \neg c)))$.
(4) $a \wedge b \vee b \wedge a$

Exercise 7:

For each of the following formula draw the formula tree and the form with minimal number of parenthesis:

(1) $(p \wedge q) \to (\neg(q \to r \wedge \neg r))$
(2) $((p \leftrightarrow (\neg\neg q)) \vee r \wedge q$

Exercise 8:

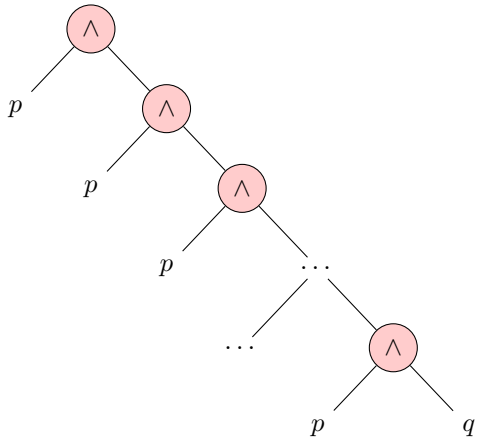
Produce a formula that contains only 2 propositional variables and $n = 1, 2, 3, 4$ binary connective $\wedge$, which has the maximum number of subformulas. Can you generalize this process to any n?

Solution If $n = 1$ then the formula $p \wedge q$ has 3 subformulas, and this is the only formula that satisfies the requirements of containing two propositinal variables and one binary connective.

With $n = 2$, then the formula must be of the form $\star \wedge (\star \wedge \star)$ or $(\star \wedge \star) \wedge \star$ where $\star$ can be replaced either with p or q. This implies that the atomic subformulas must be at most 2. Since the fromula contains 2 coonectives and every coonective corresponds to a subformula, then we have that the formulas has 4 subformulas.

if $n = 3$ the possible shapes is $\star \wedge (\star \wedge (\star \wedge \star))$, where $\star$ can be replaced by p or q. Furthermore notice that all the subformulas that has $\wedge$ as main connective are different. From this we can conclude that the we can produce a formula with 3 connectives that has 3 subformulas + 2 atomic formulas which in total is equal to 5. Notice that we can construct formulas that contains 3 occurrences of $\wedge$ that has less subformulas. For instance $(p \wedge q) \wedge (p \wedge q)$, contains only 4 subformulas, since the subformula $p \wedge q$ occurs twice and therefore it contributes only for 1 to the total number of subformulas.

In the general case we have that the formula $(\star \wedge \star) \wedge \star \wedge \star$



contains $n + 2$ subformulas. □

Exercise 9:

Suppose that a formula ϕ contains n occurrences of $\wedge$, m occurrences of $\vee$ and p occurrences of $\neg$.

(1) What is the maximum number s of subformulas of ϕ?
(2) Explain how you get this result.
(3) Provide an example of a formula with s subformulas;
(4) Provide an example of a formula with less then s subformulas. maximum number.

Solution

(1) For every occurrence of $\vee$ and $\wedge$ the formula has two subformulas (they can be different provided that we have anought propositional variables),

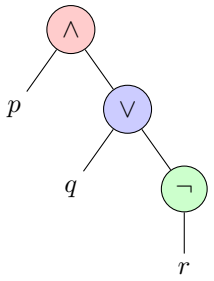
for every occurrence of $\neg$ there is one subformula. In total therefore we have $s = 2^{n+m} + p + 1$

(2) If you imagine the formula tree, and every node labelled with n there two subtrees, and for every nodel labelled with $\neg$ there is only one subtree. In both case we have to consider also the tree as a formula is a subformula of itself. If we make sure that all the subtrees are different, by using diffferent set of propositinal variables for each subtree, we can guarantee that all the subtrees are different.

(3) For instance $p \wedge (q \vee negr)$ is a formula that contains 1 $\wedge$, 1 occurrence of $\vee$ and 1 occurrence of $\neg$ and has $2^{1+1} + 1 + 1 = 6$ subformulas, which are

$$\begin{array}{ccc} p \wedge (q \vee \neg r) & p & q \vee \neg r \\ q & \neg r & r \end{array}$$

The formula tree is the following:



which has exactly 6 nodes.

(4) To obtain less subformulas than s we have to make sure that two subformulas are the same. This is the case for instance in the simple formula

$$p \wedge p$$

that contains 1 conjunction connective. Therefore $s = 2^1 + 1 = 3$, but the number of subformulas are 2, i.e., $p \wedge p$ and p. This is due to the fact that p is a subformula of both branches of $\wedge$.

□

Exercise 10:

If a formula ϕ contains n connectives, what is the minimum and the maximum number of subformulas?

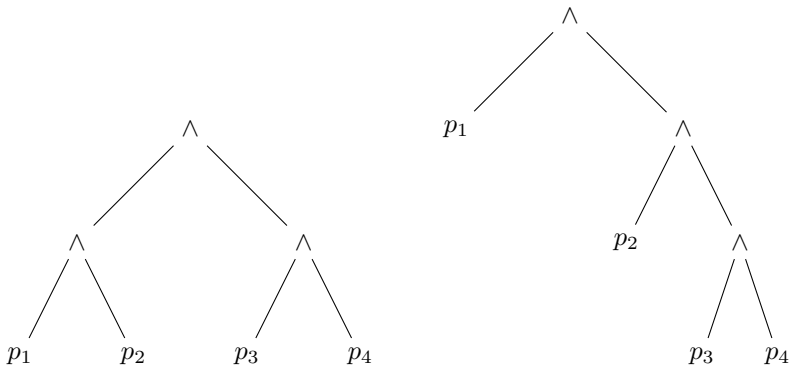
Solution First of all notice that the connective $\neg$ applies only to one subformula, while all the other connectives are binary, i.e., they apply to two subformulas. To minimize the number of subformulas therefore we can suppose that all the n connectives are $\neg$, i.e., the formula is of the form:

$$\overbrace{\neg(\neg(\neg(\dots(\neg}^{\times n} p)\dots))$$

In this case the formula has $n+1$ subformulas. Instead to maximize the number of subformulas one should use binary connectives since for each binary connective we can potentially have 2 subformulas. The maximum number of subformulas using n connectives can be obtained by using only binary connectives and making sure that the two subformulas of all the binary connectives are distinct. This can be

obtained by introducing enough propositional variables to make all subformulas for every connective different from the other. This implies that the occurrence of every connective corresponds to two subformulas. Therefore the maximum number of subformulas are $2n + 1$

Suppose for instance that $k = 2$, then the following are two examples of formulas with binary connectives ($\wedge$ in this case):



The number of subformuals are $2 \cdot 3 + 1 = 7$. More in general the maximum number of subformulas are $2n - 1$. □

Exercise 11:

Prove by structural induction that a propositional formula that contains n occurrences of $\neg$ and m occurrences of binary connectives has at most $n + 2m + 1$ subformulas. **Solution** Let ϕ be a formula that contains n occurrences of $\neg$ and m occurrences of binary propositional connectives. We prove the theorem by induction on $n + m$

Base case: Suppose that $n + m = 0$, then ϕ does not contain any propositoinal connective, and therefore it is an atomic formula p, which has exactly 1 subformula, i.e., ϕ itself. Therefore the number of subformulas of ϕ are $1 = n + 2m + 1$.

Step case: Suppose that the property holds for all the formulas that contains m' occurrences of $\neg$ and n' occurrences of binary connectives with $m' \leq m$ and $n' \leq n$ and $m' + n' < m + n$, and let us prove that it holds also for $m + n$. We consider two cases.

(1) ϕ is of the form $\neg\phi_1$. Then ϕ_1 contains $n-1$ occurrences of $\neg$ and m occurrences of binary connectives. By induction hypothesis we can infer that ϕ_1 has at most $n-1+2m+1 = n+2m$ subformulas. Since the subformulas of $\neg\phi_1$ are the subformulas of ϕ_1 plus the formula $\neg\phi_1$ itself, then the maximum number of subformulas of $\neg\phi_1$ are $n + 2m + 1$.

(2) ϕ is of the form $\phi_1 \circ \psi_2$ for some binary connective $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$. Then we have that each ϕ_i with $i = 1, 2$ contains n_i occurrences of $\neg$ and m_i occurrences of binary connectives. with $m_i \leq m - 1$, and $n_i \leq n$. We therefore have that $n_i + m_i < n + m$. By induction on ϕ_i we have that the maximum number of subformulas of ϕ_i are $n_i + 2m_i + 1$. Since the subformulas of $\phi_1 \circ \phi_2$ are the subformulas of ϕ_i for $i = 1, 2$ plus $\phi_1 \circ \phi_1$ itself, we have that $\phi_1 \circ \phi_2$ has at most

$n_1 + 2m_1 + 1 + n_2 + 2m_2 + 1 + 1 = (n_1 + n_2) + 2(m_1 + m_2 + 1) + 1$ subformulas. Notice that $n_1 + n_2 = n$ since every occurrence of $\neg$ in ϕ is either an occurrence in ϕ_1 or in ϕ_2 but not in both. Furthermore $m_1 + m_2 = m - 1$ because an occurrence of a binary connective is either in ϕ_1 or in ϕ_2 or it is $\circ$, the main connective of $\phi_1 \circ \phi_2$. This allows us to infer that $\phi_1 \circ \phi_2$ has at most $n + 2m + 1$ subformulas.

□

Exercise 12:

Define when a formula is valid, satisfiable, unsatisfiable, and not valid, and the relations between these concepts.

Exercise 13:

List the pair of items from each one of the two lists (i.e, a list of (number),(letter) pairs) that corresponds to equivalent statements.

(1) A is satisfiable	(a) $\neg A$ is satisfiable
(2) A is valid	(b) $\neg A$ is valid
(3) A is unsatisfiable	(c) $\neg A$ is unsatisfiable
(4) A is not valid	(d) $\neg A$ is not valid
(5) A is satisfiable and B is satisfiable	(e) $A \wedge B$ is satisfiable
(6) A is valid and B is valid	(f) $A \wedge B$ is valid
(7) A is unsatisfiable and B is unsatisfiable	(g) $A \wedge B$ is unsatisfiable
(8) A is not valid and B is not valid	(h) $A \wedge B$ is not valid
(9) A is satisfiable or B is satisfiable	(i) $A \vee B$ is satisfiable
(10) A is valid or B is valid	(j) $A \vee B$ is valid
(11) A is unsatisfiable or B is unsatisfiable	(k) $A \vee B$ is unsatisfiable
(12) A is not valid or B is not valid	(l) $A \vee B$ is not valid

Solution Let us recall the definiton of *satisfiable*, *valid*, *Unsatisfiabe*, and *not valid* formula.

A is satisfiable	if **there is** an interpretation $\mathcal{I}$ that $\mathcal{I} \models A$
A is valid	if **for every** interpretation $\mathcal{I}$, $\mathcal{I} \models A$
A is unsatisfiabe	if **for every** interpretation $\mathcal{I}$, $\mathcal{I} \not\models A$
A is not valid	if **there is** an interpretation $\mathcal{I}$ such that $\mathcal{I} \not\models A$

Where $\mathcal{I} \models A$ means that A is true in $\mathcal{I}$ and $\mathcal{I} \not\models A$ means that A is false in $\mathcal{I}$.

Using the above definition we can find the following connections between the items of the two lists.

	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	(l)
(1)				$\Leftrightarrow$					$\Rightarrow$			
(2)			$\Leftrightarrow$	$\Rightarrow$					$\Rightarrow$	$\Rightarrow$		
(3)	$\Rightarrow$	$\Leftrightarrow$					$\Rightarrow$	$\Rightarrow$			$\Leftarrow$	
(4)	$\Leftrightarrow$	$\Leftarrow$						$\Rightarrow$				$\Leftarrow$
(5)				$\Rightarrow$	$\Leftarrow$	$\Leftarrow$			$\Rightarrow$			
(6)			$\Rightarrow$	$\Rightarrow$	$\Rightarrow$	$\Leftrightarrow$			$\Rightarrow$	$\Rightarrow$		
(7)	$\Rightarrow$	$\Rightarrow$				$\Rightarrow$	$\Rightarrow$				$\Leftrightarrow$	$\Leftarrow$
(8)	$\Rightarrow$							$\Rightarrow$			$\Leftarrow$	$\Leftrightarrow$
(9)			$\Leftarrow$	$\Leftarrow$	$\Leftarrow$	$\Leftarrow$			$\Leftrightarrow$	$\Leftarrow$		
(10)						$\Leftarrow$			$\Rightarrow$	$\Rightarrow$		
(11)		$\Leftarrow$					$\Rightarrow$	$\Rightarrow$				
(12)	$\Leftarrow$		$\Leftarrow$				$\Leftarrow$	$\Leftrightarrow$			$\Leftarrow$	$\Leftarrow$

From the above tabel we can extract the pairs of equivalent statements (shown in red).

$$(1,d) \quad (2,c) \quad (3,b) \quad (4,a) \quad (6,t) \quad (7,k) \quad (8,l) \quad (9,i) \quad (12,h)$$

□

Exercise 14:

Explain the difference between the following statements

(1) $\models A \vee B$ ($A \vee B$ is valid)
(2) $\models A$ or $\models B$ (A is valid or B is valid)

which one is the strongest?

Solution Let us expland 1. and 2. according to the definition of valid formula:

(1) $\models A \vee B$ means that for every interpretation $\mathcal{I}$, $\mathcal{I} \models A \vee B$, which means that either $\mathcal{I} \models A$ or $\mathcal{I} \models B$
(2) $\models A$ or $\models B$ instead means that either for every interpretation $\mathcal{I}$, $\mathcal{I} \models A$ or for every interpretation $\mathcal{I}$, $\mathcal{I} \models B$.

To highlight the difference between 1. and 2. you can write their definition by using a more formal notation,

$$\models A \vee B \iff \forall \mathcal{I}, (\mathcal{I} \models A \text{ or } \mathcal{I} \models B) \tag{5}$$

$$\models A \text{ or } \models B \iff (\forall \mathcal{I}, \mathcal{I} \models A) \text{ or } (\forall \mathcal{I}, \mathcal{I} \models B) \tag{6}$$

An example that shows the difference can be constructed by taking A equal to the atomic formula p and B the negated atomic formula $\neg p$. You have that $\models p \vee \neg p$, but neither $\models p$ nor $\models \neg p$ Clearly 2. is a stronger statement than 1. □

Exercise 15:

Find three formula A, B, and C such that $A \wedge B \wedge C$ is unsatisfiable and such that the conjunction of any pair of them is satisfiable. I.e., $A \wedge B$, $A \wedge C$ and $B \wedge C$ are satisfiable.

Exercise 16:

Suppose that A and B contains two disjoint set of propositional variables. Show that $A \wedge B$ is satisfiable if and only if A is satisfiable and B is satisfiable.

Exercise 17:
Show that if A is satisfiable then $A \wedge p$ or $A \wedge \neg p$ is satisfiable.

Exercise 18:
Prove that if A and B contain a disjoint set of propositional variable then $A \vee B$ is valid if and only if A is valid or B is valid.

Exercise 19:
Let ϕ be a propositional formula built only with the operators $\vee$, $\wedge$ and $\neg$. An occurrence of a propositional variable p in ϕ is *positive* if it is in the scope of an *even* number of $\neg$ operators, and it is negative if it is not positive. Provide an explanation (or better a proof) of the fact that if ϕ does not contains two occurrences of the same propositional variable, which are one positive and one negative, then ϕ is satisfiable.

Solution If you transform ϕ in NNF (negated normal form) then for every propositional variable p, either all its occurrences in NNF(ϕ) will not be negated (if all the occurrences of p in ϕ are positive, then an even number negations cancel out) or all of them will have a $\neg$ in front of them (i.e, when they are negative occurrences, an odd number of negations reduce to a single negation). The assignment that maps all positive p into true and the negative p into false, satisfies NNF(ϕ) and therefore it satisfies also ϕ. □

Exercise 20:
For each of the following formulas, construct a truth table and state whether it is valid, satisfiable, or unsatisfiable.

(1) $p \wedge \neg p$
(2) $p \vee \neg p$
(3) $(p \vee \neg q) \to q$
(4) $(p \vee q) \to (p \wedge q)$
(5) $(p \to q) \leftrightarrow (\neg q \to \neg p)$
(6) $(p \to q) \leftrightarrow (q \to p)$

Exercise 21:
Give a truth-table definition of the ternary boolean operation "if p then q else r", and write a propositional formula using only the connectives $\to$ and $\neg$ that is equivalent to such an operator.

Solution One possible intuitive reading of "if p then q else r" is that when p is true then q should be also true and we don't know anything about r, and when p is false then q should be true, and we don't know anything about p. This is represented

by the following table:

p	q	r	if p then q else r
T	T	T	T
T	T	F	T
T	F	T	F
T	F	F	F
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	F

Such a ternary connective can be formalized by the propositional formula

$$(p \to q) \wedge (\neg p \to r)$$

□

Exercise 22:
Given the truth table for an arbitrary n-ary boolean function

$$f : \{0,1\}^n \to \{0,1\}$$

describe how one can build a formula ϕ using only n propositional variables $p_1, \ldots, p_n$ such that the following holds:

$$f(x_1, \ldots, x_n) = 1 \text{ if and only if } \mathcal{I} \models \phi$$

where $\mathcal{I}$ is an assignment such that $\mathcal{I}(p_i) = x_i$

Solution For every $\boldsymbol{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ let us define the formula $\phi_{\boldsymbol{x}}$ as follows:

$$\phi_{\boldsymbol{x}} = \bigwedge_{\substack{i=1 \\ x_i=1}}^{n} p_i \wedge \bigwedge_{\substack{i=1 \\ x_i=0}}^{n} \neg p_i$$

Notice that the formula $\phi_{\boldsymbol{x}}$ is satisfied only by a single assignment, i.e., the assignment in which $\mathcal{I}(p_i) = x_i$ for all the propositional variables p_i. Let us define ϕ as the disjunction of all the $\phi_{\boldsymbol{x}}$ such that $f(\boldsymbol{x}) = 1$.

$$\phi = \bigvee_{\substack{\boldsymbol{x} \in \{0,1\}^n \\ f(\boldsymbol{x})=1}} \phi_{\boldsymbol{x}}$$

For every interpretation $\mathcal{I}$, if $\mathcal{I} \models \phi$ then for some $\boldsymbol{x}$ for which $f(\boldsymbol{x}) = 1$, $\mathcal{I} \models \phi_{\boldsymbol{x}}$. By the construciton of $\phi_{\boldsymbol{x}}$ $\mathcal{I}$ is the assignment that assigns x_i to each p_i. □

Exercise 23:
Are the following formulae satisfiable, valid, unsatisfiable, or not valid?

(1) $(\neg p \vee q) \wedge (q \to (\neg r \wedge \neg p)) \wedge (p \vee r)$
(2) $((\neg p \vee q) \to q \wedge (q \to r) \wedge \neg r) \to p$
(3) $\neg p \wedge (\neg q \vee r) \wedge (\neg p \to q \wedge \neg r)$

Solution Let us build the truth tables of the three formulas:

p	q	r	$(((\neg$	$(p$	$\vee$	$q))$	$\wedge$	$(q$	$\rightarrow$	$((\neg$	$r)$	$\wedge$	$(\neg$	$p))))$	$\wedge$	$(p$	$\vee$	$r))$
T	T	T	F	T	T	T	F	T	F	F	T	F	F	T	**F**	T	T	T
T	T	F	F	T	T	T	F	T	F	T	F	F	F	T	**F**	T	T	F
T	F	T	F	T	T	F	F	F	T	F	T	F	F	T	**F**	T	T	T
T	F	F	F	T	T	F	F	F	T	T	F	F	F	T	**F**	T	T	F
F	T	T	F	F	T	T	F	T	F	F	T	F	T	F	**F**	F	T	T
F	T	F	F	F	T	T	F	T	T	T	F	T	T	F	**F**	F	F	F
F	F	T	T	F	F	T	F	F	T	F	T	F	T	F	**T**	F	T	T
F	F	F	T	F	F	F	T	F	T	T	F	T	T	F	**F**	F	F	F

p	q	r	$((((\neg$	$p)$	$\vee$	$q)$	$\rightarrow$	$((q$	$\wedge$	$(q$	$\rightarrow$	$r))$	$\wedge$	$(\neg$	$r)))$	$\rightarrow$	$p)$
T	T	T	F	T	T	T	F	T	T	T	T	T	F	F	T	**T**	T
T	T	F	F	T	T	T	F	T	F	T	F	F	F	T	F	**T**	T
T	F	T	F	T	F	F	T	F	F	F	T	T	F	F	T	**T**	T
T	F	F	F	T	F	F	T	F	F	F	T	F	F	T	F	**T**	T
F	T	T	T	F	T	T	F	T	T	T	T	T	F	F	T	**T**	F
F	T	F	T	F	T	T	F	T	F	T	F	F	F	T	F	**T**	F
F	F	T	T	F	T	F	F	F	F	F	T	T	F	F	T	**T**	F
F	F	F	T	F	T	F	F	F	F	F	T	F	F	T	F	**T**	F

p	q	r	$(((\neg$	$p)$	$\wedge$	$((\neg$	$q)$	$\vee$	$r))$	$\wedge$	$((\neg$	$p)$	$\rightarrow$	$(q$	$\vee$	$(\neg$	$r))))$
T	T	T	F	T	F	F	T	T	T	**F**	F	T	T	T	T	F	T
T	T	F	F	T	F	F	T	F	F	**F**	F	T	T	T	T	T	F
T	F	T	F	T	F	T	F	T	T	**F**	F	T	T	F	F	F	T
T	F	F	F	T	F	T	F	T	F	**F**	F	T	T	F	T	T	F
F	T	T	T	F	T	F	T	T	T	**T**	T	F	T	T	T	F	T
F	T	F	T	F	F	F	T	F	F	**F**	T	F	T	T	T	T	F
F	F	T	T	F	T	T	F	T	T	**F**	T	F	F	F	F	F	T
F	F	F	T	F	T	T	F	T	F	**T**	T	F	T	F	T	T	F

(1) The first formula is not valid since there are truth assignments that evaluates it to false. It is satisfiable since for the truth assignment $\mathcal{I}(p) = False$, $\mathcal{I}(q) = False$ and $\mathcal{I}(r) = True$ (one but last line of the first truth table) the formula is evaluated to True. Consequenclty the formula is not unsatisfiable.
(2) The second formula is valid since it is true for all the assignments. It is also satisfiable since there are assignments that makes it true.
(3) The third formula, like the first one is not valid but it is satisfiable.

□

Exercise 24:

Suppose that ϕ contains only the $\leftrightarrow$ operator. Prove that if every propositional variable p occours an evenq number of times, then ϕ is valid. **Solution** The proof is based on the fact that $\leftrightarrow$ operator is associative and commutative. let us prove these two properties

Associativity of $\leftrightarrow$: We build the truth tables of $A \leftrightarrow (B \leftrightarrow C)$ and $(A \leftrightarrow B) \leftrightarrow C$ and see that the two formulas takes identical truth values.

A	B	C	A	↔	(B	↔	C)	(A	↔	B)	↔	C
T	T	T	T	T	T	T	T	T	T	T	T	T
T	T	F	T	F	T	F	F	T	T	T	F	F
T	F	T	T	F	F	F	T	T	F	F	F	T
T	F	F	T	T	F	T	F	T	F	F	T	F
F	T	T	F	F	T	T	T	F	F	T	F	T
F	T	F	F	T	T	F	F	F	F	T	T	F
F	F	T	F	T	F	F	T	F	T	F	T	T
F	F	F	F	F	F	T	F	F	T	F	F	F

Commutativity of ↔: We use the same procedure to prove that $A \leftrightarrow B$ is equivalent to $B \leftrightarrow A$.

A	B	A	↔	B	B	↔	A
T	T	T	T	T	T	T	T
T	F	T	F	F	F	F	T
F	T	F	F	T	T	F	F
F	F	F	T	F	F	T	F

Associativity and commutativity imply that every formula ϕ in which all the propositional variables appears an even number of ties can be rearranged in the following form

$$(p_1 \leftrightarrow p_1) \leftrightarrow (p_2 \leftrightarrow p_2) \leftrightarrow (p_3 \leftrightarrow p_3), \dots$$

Which is equivalent to $\top \leftrightarrow \top \leftrightarrow \top \dots$ which is always true. □

Exercise 25:

Find a formula ϕ that has the following truth table, and explain the method you have followed to find it.

p	q	r	ϕ
T	T	T	T
T	T	F	F
T	F	T	F
T	F	F	F
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	F

Solution A possible way to proceed is based on three main steps (1) building is by associating a conjunction of literals that fully describes each interpretation that satisfies ϕ; (2) put them in a disjunction and (3) simplify the resulting disjunction as much as possible.

(1) For every interpretation $\mathcal{I}$ on the set of proposition $\mathcal{P}$ we can define the conjunction of the literals that are satisfied by $\mathcal{I}$

$$\psi_{\mathcal{I}} \triangleq \bigwedge_{\substack{l \in Lit \\ \mathcal{I} \models l}} l$$

where Lit is the set of literals on the propositional variables in $\mathcal{P}$. Notice that $\psi_{\mathcal{I}}$ is satisfied by $\mathcal{I}$ and $\mathcal{I}$ is the only model that satisfies $\psi_{\mathcal{I}}$.

(2) Let us put in disjunction all the $\psi_{\mathcal{I}}$ such that $\mathcal{I} \models \phi$.

$$(p \wedge q \wedge r) \vee (\neg p \wedge q \wedge r) \vee (\neg p \wedge \neg q \wedge r) \tag{7}$$

Notice taht the above formula has exactly the same models than ϕ since is it the disjunction of the formulas that are true in each model of ϕ.

(3) We can then simplify it. First notice that r is true in all the disjunction, so r must be true, We therefore obtain:

$$((p \wedge q) \vee (\neg p \wedge q) \vee (\neg p \wedge \neg q)) \wedge r \tag{8}$$

Furthermore notice that $(p \wedge q) \vee (\neg p \wedge q) \vee (\neg p \wedge \neg q)$ is equialent to $\neg(p \wedge \neg q)$ which is equivalent to $p \rightarrow q$

$$(p \rightarrow q) \wedge r$$

□

9.2. Logical consequence. Exercise 26:

Prove the following logical consequences:

(1) $p \models p \vee q$
(2) $q \vee p \models p \vee q$
(3) $p \vee q, p \rightarrow r, q \rightarrow r \models r$
(4) $p \rightarrow q, p \models q$
(5) $p, \neg p \models q$

Solution

(1) Suppose that $\mathcal{I} \models p$, then by definition $\mathcal{I} \models p \vee q$.
(2) Suppose that $\mathcal{I} \models q \vee p$, then either $\mathcal{I} \models q$ or $\mathcal{I} \models p$. In both cases we have that $\mathcal{I} \models p \vee q$.
(3) Suppose that $\mathcal{I} \models p \vee q$ and $\mathcal{I} \models p \rightarrow r$ and $\mathcal{I} \models q \rightarrow r$. Then either $\mathcal{I} \models p$ or $\mathcal{I} \models q$. In the first case, since $\mathcal{I} \models p \rightarrow r$, then $\mathcal{I} \models r$, In the second case, since $\mathcal{I} \models q \rightarrow r$, then $\mathcal{I} \models r$.
(4) Suppose that $\mathcal{I} \models \neg p$, then not $\mathcal{I} \models p$, which implies that there is no $\mathcal{I}$ such that $\mathcal{I} \models p$ and $\mathcal{I} \models \neg p$. This implies that all the interpretations that satisfy p and $\neg p$ (actually none) satisfy also q.
(5) ...
(6) ...

□

Exercise 27:

Show that if $\Gamma \models A$ and $\Gamma \models \neg A$, then Γ is not satisfiable.

Exercise 28:

Prove that $q \rightarrow p$ is not a logical consequence of $p \rightarrow q$.

Exercise 29:

Prove that $\neg pimp \neg q$ is not a logical consequence of $p \rightarrow q$.

Exercise 30:

Prove that $\neg q \rightarrow \neg p$ is a logical consequence of $p \rightarrow q$.

Exercise 31:
Prove the following logical consequences

(1) $p \models p \vee q$
(2) $q \vee p \models p \vee q$
(3) $p \vee q, p \to r, q \to r \models r$
(4) $p \to q, p \models q$
(5) $p, \neg p \models q$

Solution

(1) Suppose that $\mathcal{I} \models p$, then by definition $\mathcal{I} \models p \vee q$.
(2) Suppose that $\mathcal{I} \models q \vee p$, then either $\mathcal{I} \models q$ or $\mathcal{I} \models p$. In both cases we have that $\mathcal{I} \models p \vee q$.
(3) Suppose that $\mathcal{I} \models p \vee q$ and $\mathcal{I} \models p \to r$ and $\mathcal{I} \models q \to r$. Then either $\mathcal{I} \models p$ or $\mathcal{I} \models q$. In the first case, since $\mathcal{I} \models p \to r$, then $\mathcal{I} \models r$, In the second case, since $\mathcal{I} \models q \to r$, then $\mathcal{I} \models r$.
(4) Suppose that $\mathcal{I} \models \neg p$, then not $\mathcal{I} \models p$, which implies that there is no $\mathcal{I}$ such that $\mathcal{I} \models p$ and $\mathcal{I} \models \neg p$. This implies that all the interpretations that satisfy p and $\neg p$ (actually none) satisfy also q.

□

9.3. Logical equivalence. Exercise 32:

Show that $\wedge$ is associative, i.e., $a \wedge (b \wedge c)$ is equivalent to $a \wedge (b \wedge c)$, and therefore one can write $a \wedge b \wedge c \wedge d \wedge \dots$ without parentesis.

Solution Let us generate the truth table for both]

a	b	c	a	∧	(	b	∧	c	)	(	a	∧	b	)	∧	c
T	T	T	T	T		T	T	T			T	T	T	T	T	
T	T	F	T	F		T	F	F			T	T	T	F	F	
T	F	T	T	F		F	F	T			T	F	F	F	T	
T	F	F	T	F		F	F	F			T	F	F	F	F	
F	T	T	F	F		T	T	T			F	F	T	F	T	
F	T	F	F	F		T	F	F			F	F	T	F	F	
F	F	T	F	F		F	F	T			F	F	F	F	T	
F	F	F	F	F		F	F	F			F	F	F	F	F	

The truth values of the fomrulas are shown in the two columns blue and red. Notice that the two columns are identical. Therefore the two formulas are equivalent. □

Exercise 33:
Show that that $\to$ is a non associative operator, i.e., thjat $a \to (b \to c)$ is not equivalent to $(a \to b) \to c$.

Solution Let us build the truth table for the two formulas.

a	b	c	a	→	(	b	→	c	)	(	a	→	b	)	→	c
T	T	T	T	T		T	T	T			T	T	T	T	T	
T	T	F	T	F		T	F	F			T	T	T	F	F	
T	F	T	T	T		F	T	T			T	F	F	T	T	
T	F	F	T	T		F	T	F			T	F	F	T	F	
F	T	T	F	T		T	T	T			F	T	T	T	T	
F	T	F	F	T		T	F	F			F	T	T	F	F	
F	F	T	F	T		F	T	T			F	T	F	T	T	
F	F	F	F	T		F	T	F			F	T	F	F	F	

Notice that there are two assignments (highlighted in red), to a, b, c for which the truth value of the two formulas are not the same. The two formulas therefore are not equivalent. This implies that the expression $a \to b \to c$ is ambigous, and one should add the parenthesis in order to specify the correct parsing. In absence of parentesis the parsing $a \to (b \to c)$ is usually taken as correct. □

Exercise 34:
Repeat the previous exercise for the connective $\vee$.

Exercise 35:
Show that the $\leftrightarrow$ is commutative and associative.

Exercise 36:
Rewrite the following formulas by using only $\wedge$ and $\neg$.

(1) $p \vee q$
(2) $p \to q$
(3) $p \leftrightarrow q$
(4) $p \to (q \to r)$
(5) $p \leftrightarrow (q \vee r)$

Exercise 37:
Rewrite the following formulas by using only $\vee$ and $\neg$.

(1) $p \wedge q$
(2) $p \to q$
(3) $p \leftrightarrow q$
(4) $p \to (q \to r)$
(5) $p \leftrightarrow (q \wedge r)$

Exercise 38:
Among the following 5 formulas find one formula that is valid and one that is not valid. For the first one prove its validity with your preferred method, for the second one provide a counter-model. (remember that $a \to b \to c$ stand for $a \to (b \to c)$)

(1) $(P \wedge Q) \to P \to Q$
(2) $(P \to Q) \vee (P \wedge \neg Q)$
(3) $(P \to Q \to R) \to P \to R$
(4) $(P \to Q \vee R) \to P \to R$

(5) $\neg(P \vee Q) \to R \to \neg R \to Q$

Solution For each formula, we can construct a truth table that lists all possible truth assignments to the propositional variables. For n variables a truth table has 2^n rows. We can evaluate the formula for variable assignment and check whether it is true or false. If the result is true for every assignment, then the formula is valid. Otherwise, there is an assignment where the formula evaluates to false. This is the counterexample. We obtain the following results: 1, valid; 2. valid; 3. not valid, counterexample: $R = \bot$, $P = \top$, $Q = \bot$; 4. Not valid, counterexample: $R = \bot$, $P = \top$, $Q = \bot$; 5. valid. □

Exercise 39:
Consider the following binary connective $\circ$ and the corresponding truth table:

x	y	$x \circ y$
0	0	1
0	1	1
1	0	1
1	1	0

Express the formulas $\neg a$, $a \wedge b$, and $a \vee b$ using only the $\circ$ connective.

Solution

(9) $$\neg a = a \circ a$$

(10) $$a \wedge b = (a \circ b) \circ (a \circ b)$$

(11) $$a \vee b = (a \circ a) \circ (b \circ b)$$

□

CHAPTER 3

Modelling in Propositional Logic

1. Logic Based Problem-solving

One of the most important applications of logic in artificial intelligence is in providing a general method for solving problems by modeling the problem in terms of logical formulas and finding the solution by applying some form of logical inference. This role of logic has been clearly identified by Adnan Darwiche in Darwiche 2020.

At an abstract level, a problem is specified by providing a set of *hypothesis* (e.g., input data, set of hypothesis, context, background knowledge, set of rules, "'...) and a *query*, whose answer should be inferred from the hypothesis. The main task in logic-based problem-solving is in *modeling* hypothesis in terms of a set of logical formulas so that the answer of the *query* can be obtained by some logical inference from such a set of formulas. The main schema is shown in Figure 1. A real world problem can be seen seen as a question to be answered given a set of data. For instance one would like to know who is the murder between a group of suspected persons and a set of clues. The data are the fact that the murdered is one among the suspected people and all the cues, the query is "who is the the murderer?", the (correct) answer will identify the person who actually committed the murder. Perhaps an example closer to real application, is the problem of finding the shortest path from one point to the other of a town. The hypothesis (data, background knowledge) are the street connections, the query is "find the shortest path from

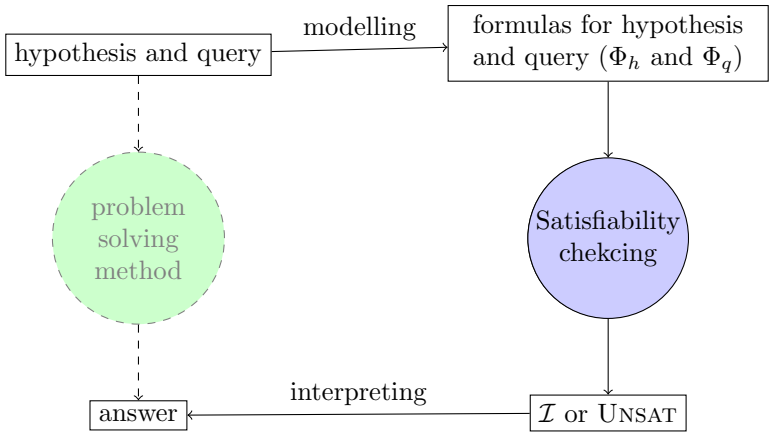


FIGURE 1. Schema of logic based problem solving method.

point a to point b" the answer is a sequence of connected streets that connets a and b with minimal length.

A general method for solving these classes of problems is by modelling the bypothesis and the query in a set of logic formulas Φ_h and Φ_q respectively and then apply some generic inference algorithm on Φ_h and Φ_q. Such an algorithm will provide an answer which is described in logical terms. Such an anwer need to be *iterpreted* to provide an answer to the real prolem.

The simplext logical inference task is *satisfiability*. I.e., check if a set of formula Φ is satisfiable by sone assignment to its propositional variables. In other words seach for an assignment $\mathcal{I}$ to the propositional variables of ϕ such that $\mathcal{I} \models \phi$. Algorithm for satisfiability provides two types of ansewrs when they are called with input Φ. The first answer is that Φ is or is not *unsatisfiable*; the second type of answer is an interpretation $\mathcal{I}$ that satisfies Φ (in case it is satisfiable). Therefore satisfiability algorithms can be used to answer two types of queries: *boolean queries* and *search queries*. Boolean queryies are queries of the form

"is it the case that a certain proposition is true/false?"

Example of these queries are "is John the murderer?", "is the murdered male or female". The answer to a boolean query is yes/no (true/false, 0/1, this is why they are called *boolean*). Search queries are queries of the form

"find an opbject that satisfies a certain proposition.

Examples of this type of queries are: "who is the mardered?". "find a path that connects location a to b", "find a path from a to b that passes through c". The answer to this type of queries is (the description of) a specific object.

2. Formalizing natural language (english) sentences

Natural [1] langauge is one of the most common way in which a problem can be specified; in the section, we discuss how to translate a variety of English statements into the language of propositional logic. From the viewpoint of sentential logic, there are five standard connectives – 'and', 'or', 'if...then', 'if and only if', and 'not'. corresponding to the connectives $\wedge$, $\vee$, $\rightarrow$, $\equiv$ and $\neg$. In addition to these standard connectives, there are in English numerous of other connectives, including 'unless', 'only if', 'neither...nor', among others.

To translate the description of a problem given in natural langauge text into a set of (propositional) logical formula we have to perform three basic steps.

(1) provide a set propositional variables corresponding to the simplest sentences of the text;
(2) compose the propositional variables in formula using the logical connectives in accordance to the natural language connectives that combine the atomic sentences

It is therefore of crucial importance to provide a correct way to tranlate the connectives in natural langauge, such as "not", "and", "although", ... into a suitable combination of the logical connectives $\wedge$, $\vee$,....

[1]The content of this section is a summary of a class by Gary Michael Hardegree, professor of Philosopy

2.1. Conjunction. Conjunction in enclish can be expressed by the connective "and"; there are however a set of alternative conjunctions that can be used. "but", "yet", "although", "though", "even though", "moreover", "furthermore", , "however", and "whereas" are all connectives that express some conjunctive information. Although these expressions have different connotations, they are all truthfunctionally equivalent to one another. For instance the sentences

(S1) it is raining, but I am happy
(S2) although it is raining, I am happy
(S3) it is raining, yet I am happy
(S4) it is raining and I am happy

are all true in the situations in which it is raining *and* i'm happy. Therefore they are truth-value equivalent (they capture the same proposition). They are all translated in $R \wedge H$, where H is the propositional variable that represent the proposition "its raining" and H the propositional variable corresponding to the proposition "I'm happy". This does not mean that they convey the same information. Indeed, for instance (S1) and (S2) convey some contrastive relation between being happy and raining, which is not present in (S4). However, this additional information is not directly related to the truth value of the formula itself. Since propositional logic captures only the truth-functions of connective, such additional information cannot be captured by propositional logic formulas.

Conjunctive information can be provided also in additional form: The template

A and B are C

where A and B are individuals and C is a common name describing a quality, corresponds to the conjunction

A is C *and* B is C

So for instance "Cesare and Caligola are emperors" can be paraphrased in "Cesare is an emperor *and* Caligola is an emperor". Similarly sentences that respect the pattern

A is B and C

where A is an individual and B and C describe some quality, can be paraphrased in

A is B *and* A is C

For instance "JS Bach is a composer and a musician" can be paraphrased in "JS Bac is a composer and JS Bach is a musician". There are many other ways to express conjunctive information about the same individual, for instance by using relative pronouns like "who". Often in this form the "and" is omitted and we have the pattern $AisaBC$, as for instance in

Charles Dickens is an English writer

meaning that Charles Dickens is English and Charles Dickens is a writer.

The pattern "A and B are C" can be used also in case in which C expresses some relation between the individuals A and B. In this case we cannot paraphrased the sentence as a conjunction of "A is C" and "B is C". Consider for instance the sentence "Pierre and Marie are married", as the intended meaning, if nothing else is added, is that "Pierre and Marie are married *each-other*".

2.2. Disjunction. See slides

2.3. Implication. See slides

2.4. Negation. See slides

3. Formalizing constraints on possible worlds

In many situation we are in front of the problem of finding a set of formulas that "must" be true in all the possible configuration of a "world", so that you restrict to consider the interpretations that corresponds to the "possible worlds", and exclude the "impossible" worlds.

EXAMPLE 3.1. *Suppose that a robot can move around a flat that is composed of 25 cells, some of them are occupied by other objects and the robot cannot move into them. This situation is graphically represented in Figure 2 and it is called semantic grid occupancy map.*

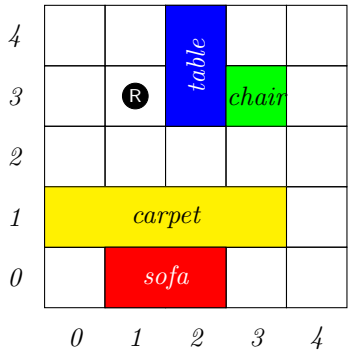


FIGURE 2. An example of occupancy 2D map

To model situation like the one shown in FIgure 2 you have to proceed follow two main steps (as in the case of translation from english)

(1) define the set of "atomic propositions" that are necessary to describe all the configurations (both the possible and the impossible worlds);
(2) write the formula that are true only in the "possible worlds" and are false in the impossible worlds".

EXAMPLE 3.2 (Cont'd). *The key aspect of scenario shown in Figure 2 is the fact that a certain object/robot occupies a cell.*

3.1. Graph coloring. Graph coloring problem is one of the basic problem in graph thery and it has a lot of aplications. In the following we will define the problem, describe it's formulation in propositional logic, and motivate it by means of an application.

DEFINITION 3.1. *A graph G is an ordered pair (V, E), where V is a finite set and $E \subset V \times V$, such that $(v, w) \in E$ implies that $v \neq w$. The set $f\, V$ is called the set of* vertices *and E is called the set of* edges *of G. G is* undirected *if $(v, w) \in V$ then $(w, v) \in V$. If $(v, w) \in E$ we way that v and w are adjacent vertices.*

DEFINITION 3.2. *A graph G is said to be k-colourable if each vertex can be assigned one of k colours so that adjacent vertices get different colours.*

An important problme is the following: given an undirected graph $G = (V, E)$ find the smallest k such that G is k-colourable. One possible method to solve this problem is to cast the problem of checking if a certain graph is k-colorable in a set of propositional formulas so that if they are satisfiable then the graph is k-colorable. In other words we have to define a propositional language and a set of axioms that formalize the graph k-coloring problem of a graph n nodes.

Let us first define the set of propositions that we need to axiomatize the graph k-coloring problem for a graph with n vertices.

- For each node $1 \leq i \leq n$ and color $1 \leq c \leq k$, color_{ic} is a propositional variable that represents the fact that the i-th node is colored with c-th color;
- For each pair of distinct nodes i, j such that $1 \leq i < j \leq n$, edge_{ij} is a propositional variable that represents the fact that there is an edge from node i to node j. Notice that we use only edge_{ij} for $i < j$ and we don't introduce edge_{ij} with $j < i$ since we have that the edges are symmetric and if htere is an edge from i to j there must be also an edge from j to i. This implies that the proposition edge_{ij} would be equivalent to edge_{ji}. Therefore, we need only one of the two.

Let us now introduce a set of axioms that imposes that the graph is crrectly colored.

- we first have to codigy the structure of the graph. for every $(i, j) \in V$ with $i < j$ we add edge_{ij}; furthermore for every $(i, j) \notin E$ with $i < j$ we add $\neg\mathsf{edge}_{ij}$;
- for each $1 \leq i \leq n$ we add the formula
$$\bigvee_{c=1}^{k} \mathsf{color}_{ic}$$
that formalizes the fact that each node has at least one color;
- for each $1 \leq i \leq n$ and $1 \leq c < c' \leq k$, we add the formula
$$\mathsf{color}_{ic} \rightarrow \neg\mathsf{color}_{ic'}$$
, which formalizes that every node has at most 1 color;
- for each $1 \leq i < j \leq n$ and $1 \leq c \leq k$ we add the formula
$$\mathsf{edge}_{ij} \rightarrow \neg(\mathsf{color}_{ic} \wedge \mathsf{color}_{jc})$$
that formalizes that adjacent nodes do not have the same color.

To conclude let's introducing a family of applications that involve avoiding some sort of clash, i.e. where some configuations shouldn't be allowed to happen in a world. A prototypical example is the following

EXAMPLE 3.3. *Suppose that a group of ministers serve on committees as described below:*

Committee	***Members***
Culture, Media & Sport	*Alexander, Burt, Clegg*
Defence	*Clegg, Djanogly, Evers*
Education	*Alexander, Gove*
Food & Rural Affairs	*Djanogly,Featherstone*
Foreign Affairs	*Evers, Hague*
Justice	*Burt, Evers, Gove*
Technology	*Clegg, Featherstone, Hague*

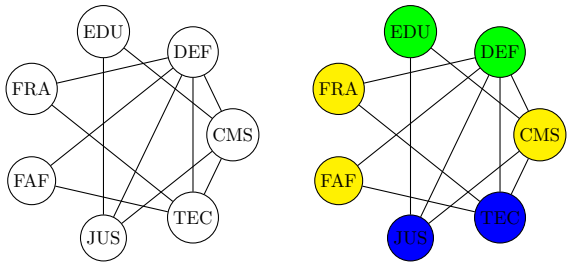


FIGURE 3. The graph at left has vertices labelled with abbreviated committee names and edges given by shared members. The graph at right is isomorphic, but has been redrawn for clarity and given a three-colouring, which turns out to be optimal.

What is the minimum number of time slots needed so that one can schedule meetings of these committees in such a way that the ministers involved have no clashes?

One can turn this into a graph-colouring problem by constructing a graph whose vertices are committees and whose edges connect those that have members in common: such committees can't meet simultaneously, or their shared members will have clashes. A suitable graph appears at left in Figure 3 where, for example, the vertex for the Justice committee (labelled JUS) is connected to the one for the Education committee (EDU) because Gove serves on both. The version of the graph at right in Figure 3 shows a three-colouring and, as the vertices CMS, EDU and JUS form a clique (i.e., totally connected graph). THerefore 3 is this is the smallest number of colours one can possibly use and so the chromatic number of the committee-and-clash graph is 3. This means that we need at least three time slots to schedule the meetings. To see why, think of a vertex's colour as a time slot: none of the vertices that receive the same colour are adjacent, so none of the corresponding committees share any members and thus that whole group of committees can be scheduled to meet at the same time. There are variants of this problem that involve, for example, scheduling exams so that no student will be obliged to be in two places at the same time or constructing sufficiently many storage cabinets in a lab so that chemicals that would react explosively if stored together can be housed separately.

4. Modelling Cardinality constraints

A very common class of constraints that we can encounter in modelling problems are the so called *cardinality constraints*, which impose limit on the number of proositional variables that are true. Let us consider the following simple example

EXAMPLE 3.4 (Crowded room). *Suppose that in a classroom there are k chairs, but there are $n > k$ students that attends the lecture. Suppose that you want to represent the fact that each single student stands or has found a seat. In this situation you have to impose that the maximum numner of students that seat are k. If seat_{ij} stands for the j-th students seats in place j, we have to impose that at most k propositional variables seat_{ij} are true, or equivalently at least $n-k$ propositional variables seat_{ij} are false. If, we want to impose that all the chairs are occupied then*

we need to require that exactly k variables seat_{ij} *are true, or equivalently, exactly* $n-k$ *variables* seat_{ij} *are false.*

let us see how cardinality constraints can be encoded in propositional formuls.

4.1. At least k**.** Given a set of boolean variables $X = \{x_1, x_2, \ldots, x_n\}$, the constraint " *at least k propositional variables in X are true* is formalized by

$$\bigvee_{\substack{I\subseteq[n]\\|I|=k}} \bigwedge_{i\in I} x_i \qquad \text{where } [n] = \{1, 2, 3, \ldots n\} \tag{12}$$

In words, we consider all the possible subsets of $\{x_1, \ldots, x_n\}$ that contains k elements and require that for at least one of such subset (formalized by the outer disjunction) all the variables are true (formalized by the inner conjunction).

Example 3.5. *At least 2 among* $X = \{a, b, c, d\}$*. The subsets of X that contains two elements are* $\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}$*, and* $\{c, d\}$*. We have therefore that formula* (12) *becomes*

$$(a \wedge b) \vee (a \wedge c) \vee (a \wedge d) \vee (b \wedge c) \vee (b \wedge d) \vee (c \wedge d)$$

An alternative formulation of at least k among $X = \{x_1, \ldots, x_n\}$ variables, can be obtained using the following arguments. If I select $n-k+1$ variables in the set X then at least one of them must be true. We can put this in a propositional formula obtaining

$$\bigwedge_{\substack{I\subseteq[n]\\|I|=n-k+1}} \bigvee_{i\in I} x_i \tag{13}$$

For reasons that will be clarified later, the form (13) with ouyter conjunction and inner disjunction is preferrable then the form (12). The two forms anyway are equivalent. (prove it by exercize)

Example 3.6. *The at least 2 among X costaint in the form* (13) *is the following*

$$(a \vee b \vee c) \wedge (a \vee b \vee d) \wedge (b \vee c \vee d)$$

4.2. At most k**.** The constraint *at most k propositional variables in X are true* can be rephrased as *it is not the case that at least* $k+1$ *variables in X are true*. Therefore the *at most k* constraint can be formalized as the negation of *at least* $k+1$.

$$\neg\left(\bigvee_{\substack{I\subseteq[n]\\|I|=k+1}} \bigwedge_{i\in I} x_i\right) \qquad \text{which is equivalent to} \qquad \bigwedge_{\substack{I\subseteq[n]\\|I|=k+1}} \bigvee_{i\in I} \neg x_i \tag{14}$$

Example 3.7. *At most 2 among* $X = \{a, b, c, d\}$ *are true, can be formalized using the right formula of* (4.2) *as:*

$$(\neg a \vee \neg b \vee \neg c) \wedge (\neg a \vee \neg b \vee \neg d) \ \wedge (\neg a \vee \neg c \vee \neg d) \wedge (\neg b \vee \neg c \vee \neg d)$$

4.3. Exactly k. The constraint *exactly k propositional variables in X are true* can be rephrased as the conjunction of *at least k and at most k variables in X are true*. Therefore it can be formalized by the conjunction the of the formulas (13) and (4.2) obtaining

$$\bigwedge_{\substack{I\subseteq[n]\\|I|=n-k+1}} \bigvee_{i\in I} x_i \wedge \bigwedge_{\substack{I\subseteq[n]\\|I|=k+1}} \bigvee_{i\in I} \neg x_i \tag{15}$$

EXAMPLE 3.8 (Exactly k among $X = \{a, b, c, d\}$).

$$(a \vee b \vee c) \wedge (a \vee b \vee d) \wedge (b \vee c \vee d) \wedge (\neg a \vee \neg b \vee \neg c) \wedge (\neg a \vee \neg b \vee \neg d) \wedge (\neg a \vee \neg c \vee \neg d) \wedge (\neg b \vee \neg c \vee \neg d)$$

A special case of *exactly k* cardinality constraint is whenpp $k = 1$. The cardinality constraint *exactly* 1 *among the variables in X are true* is formalized using (**??**) by the following formula

$$\bigvee_{i=1}^{n} x_i \wedge \bigwedge_{1\leq i<j\leq n} (\neg x_i \vee \neg x_j) \tag{16}$$

This formula can be read as: at least one variable among X must be true (first part of the conjunction) and for every pair ij with $i < j$ ar least one must be false. Notice that it is enough to consider $i < j$ because the case of $i > j$ will result in the same formula. (since $\vee$ is commutative).

4.4. Efficient representation of *exactly* k. Notice that the length of the formula that codifies the *exactly k* cardinality constaint is $(n - k + 1)\binom{n}{k-1} + (k + 1)\binom{n}{k+1}$, where $\binom{n}{k} = \frac{n!}{k!(n-k!)}$ is called the binomial coefficient (read "n choose k) is the number of distinct subsets of k elements choosen from a set of n elements. This can be done by encoding a procedure to produce the sets I. We can think that the selection of I is done by the following algorithm

Algorithm 1 Select k elements from X

```
Y ← ∅
for o ≤ i ≤ k do
    y ← select an element from X
    Y ← I ∪ {y}
    X ← I \ {y}
end for
return Y
```

The efficient encoding is obtained by simulating the behavious of the algorithm 1 withing a set of propositional formulas, exploiting only the *exactly 1* cardinality constraint. For every $1 \leq i \leq k$ and for every $1 \leq j \leq n$ add the variable y_{ij} with the following intuitive meaning:

(1) x_j is true if it is selected in some iteration;
(2) At every iteration you select exactly 1 x_j
(3) If x_j is selected at one iteration it cannot be selected in the other iterations

The above statements can be formalized by the following formulas

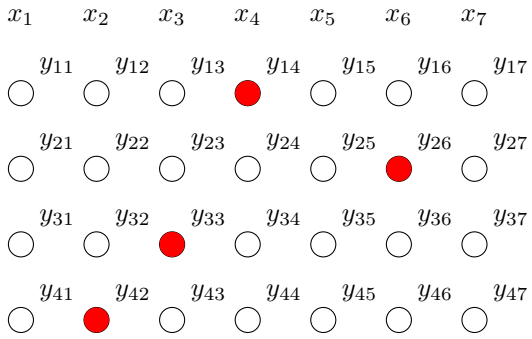


FIGURE 4. Every row represent an iteration, for every raw the selected variables is highighted in red.

(1) x_j is true if it is selected at least in one iteration

$$x_j \equiv \bigvee_{i=1}^{k} y_{ij}$$

(2) exactly 1 x_j is selected at every iteration i:

$$\bigwedge_{i=1}^{k} \left(\bigwedge_{j<j'=1}^{n} \neg(y_{ij} \wedge y_{ij'}) \wedge \bigvee_{j=1}^{n} y_{ij} \right)$$

(3) at most 1 x_j is selected in all the iterations:

$$\bigwedge_{j=1}^{n} \bigwedge_{i<i'=1}^{k} \neg(y_{ij} \wedge y_{i'j})$$

A graphical representation of what is happening is shown in Figure 4

5. exercises

Exercise 40:
Formalize the following english statements

(1) f Davide comes to the party then Bruno and Carlo come too
(2) Carlo comes to the party only if Angelo and Bruno do not come
(3) If Davide comes to the party, then, if Carlo doesn't come then Angelo comes

Solution We define the following propositional variables corresponding to the simple propositions.

A : Angela goes to the party

B : Bruno goes to the party

C : Carlo goes to the party

D : Davide goes to the party

(1) If Davide comes to the party then Bruno and Carlo come too

$$D \to B \wedge C$$

(2) Carlo comes to the party only if Angelo and Bruno do not come

$$C \to \neg A \wedge \neg B$$

(3) If Davide comes to the party, then, if Carlo doesn't come then Angelo comes

$$D \to (\neg C \to A)$$

□

Exercise 41:
Formalize the following english statements

(1) Carlo comes to the party provided that Davide doesn't come, but, if Davide comes, then Bruno doesn't come
(2) A necessary condition for Angelo coming to the party, is that, if Bruno and Carlo aren't coming, Davide comes
(3) Angelo, Bruno and Carlo come to the party if and only if Davide doesn't come, but, if neither Angelo nor Bruno come, then Davide comes only if Carlo comes

Solution We define the following propositional variables corresponding to the simple propositions.

A : Angela goes to the party

B : Bruno goes to the party

C : Carlo goes to the party

D : Davide goes to the party

(1) Carlo comes to the party provided that Davide doesn't come, but, if Davide comes, then Bruno doesn't come

$$(C \to \neg D) \wedge (D \to \neg B)$$

(2) A necessary condition for Angelo coming to the party, is that, if Bruno and Carlo aren't coming, Davide comes

$$A \to (\neg B \wedge \neg C \to D)$$

(3) Angelo, Bruno and Carlo come to the party if and only if Davide doesn't come, but, if neither Angelo nor Bruno come, then Davide comes only if Carlo comes

$$(A \wedge B \wedge C \leftrightarrow \neg D) \wedge (\neg A \wedge \neg B \to (D \to C))$$

□

Exercise 42:

Formalize the following constraing on the binary strings of n bits $(x_1, \ldots, x_n)$

(1) every sequence of k 1's is followed by a sequence of k 0's, with $k \geq 1$;
(2) the k-th digit is the product of the previous two digits (for $k \geq 2$;
(3) the k-th digit is the product of all the previous digits (the product of 0 digits is 1);
(4) the sequence is a palindrom;
(5) With n even: the second half of the string is a permutation of the first half;
(6) the string contains an even number of 0's.

Exercise 43:

Five friends (Abby, Heather, Kevin, Randy and Vijay) have access to an on-line chat room. We know the following are true:

(1) Either K or H or both are chatting.
(2) Either R or V but not both are chatting.
(3) If A is chatting, then R is chatting.
(4) V is chatting if and only if K is chatting.
(5) If H is chatting, then both A and K are chatting.

Represent the above facts in CNF (set of clauses) Notice that there are sentences that correspond to more than one clause.

Solution

(1) $K \vee H$,
(2) $R \vee V$, $\neg R \vee \neg V$,
(3) $\neg A \vee B$,
(4) $\neg V \vee K$, $V \vee \neg K$,
(5) $\neg H \vee A$, $\neg H \vee K$,

□

Exercise 44:

Translate each of the following statements into the language of sentential logic. Use the suggested abbreviations (capitalized words), if provided; otherwise, devise an abbreviation scheme of your own. In each case, write down what atomic statement each letter stands for, making sure it is a complete sentence. Letters should stand for positively stated sentences, not negatively stated ones; for example, the negative sentence 'I am not hungry' should be symbolized as '¬H' using 'H' to stand for 'I am hungry'.

(1) Although it is RAINING, I plan to go JOGGING this afternoon.
(2) It is not RAINING, but it is still too WET to play.
(3) JAY and KAY are Sophomores.
(4) It is DINNER time, but I am not HUNGRY.
(5) Although I am TIRED, I am not QUITTING.
(6) Jay and Kay are roommates, but they hate one another.
(7) Jay and Kay are Republicans, but they both hate Nixon.
(8) KEEP trying, and the answer will APPEAR.
(9) GIVE him an inch, and he will TAKE a mile.
(10) Either I am CRAZY or I just SAW a flying saucer.
(11) Either Jones is a FOOL or he is DISHONEST.
(12) JAY and KAY won't both be present at graduation.
(13) JAY will win, or KAY will win, but not both.
(14) Either it is RAINING, or it is SUNNY and COLD.
(15) It is RAINING or OVERCAST, but in any case it is not SUNNY.
(16) If JONES is honest, then so is SMITH.
(17) If JONES isn't a crook, then neither is SMITH.
(18) Provided that I CONCENTRATE, I will not FAIL.
(19) I will GRADUATE, provided I pass both LOGIC and HISTORY.
(20) I will not GRADUATE if I don't pass both LOGIC and HISTORY.
(21) Neither JAY nor KAY is able to attend the meeting.
(22) Although I have been here a LONG time, I am neither TIRED nor BORED.
(23) I will GRADUATE this semester only if I PASS intro logic.
(24) KAY will attend the party only if JAY does not.
(25) I will SUCCEED only if I WORK hard and take RISKS.
(26) I will go to the BEACH this weekend, unless I am SICK.
(27) Unless I GOOF off, I will not FAIL intro logic.
(28) I won't GRADUATE unless I pass LOGIC and HISTORY.
(29) In order to ACE intro logic, it is sufficient to get a HUNDRED on every exam.
(30) In order to PASS, it is necessary to average at least FIFTY.
(31) In order to become a PHYSICIAN, it is necessary to RECEIVE an M.D. and do an INTERNSHIP.
(32) In order to PASS, it is both necessary and sufficient to average at least FIFTY.
(33) Getting a HUNDRED on every exam is sufficient, but not necessary, for ACING intro logic.
(34) TAKING all the exams is necessary, but not sufficient, for ACING intro logic.
(35) In order to get into MEDICAL school, it is necessary but not sufficient to have GOOD grades and take the ADMISSIONS exam.
(36) In order to be a BACHELOR it is both necessary and sufficient to be ELIGIBLE but not MARRIED.
(37) In order to be ARRESTED, it is sufficient but not necessary to COMMIT a crime and GET caught.
(38) If it is RAINING, I will play BASKETBALL; otherwise, I will go JOGGING.

(39) If both JAY and KAY are home this weekend, we will go to the BEACH; otherwise, we will STAY home.
(40) JONES will win the championship unless he gets INJURED, in which case SMITH will win.
(41) We will have DINNER and attend the CONCERT, provided that JAY and KAY are home this weekend.
(42) If neither JAY nor KAY can make it, we should either POSTPONE or CANCEL the trip.
(43) Both Jay and Kay will go to the beach this weekend, provided that neither of them is sick.
(44) I'm damned if I do, and I'm damned if I don't.
(45) If I STUDY too hard I will not ENJOY college, but at the same time I will not ENJOY college if I FLUNK out.
(46) If you NEED a thing, you will have THROWN it away, and if you THROW a thing away, you will NEED it.
(47) If you WORK hard only if you are THREATENED, then you will not SUCCEED.
(48) If I do not STUDY, then I will not PASS unless the prof ACCEPTS bribes.
(49) Provided that the prof doesn't HATE me, I will PASS if I STUDY.
(50) Unless logic is very DIFFICULT, I will PASS provided I CONCENTRATE.
(51) Unless logic is EASY, I will PASS only if I STUDY.
(52) Provided that you are INTELLIGENT, you will FAIL only if you GOOF off.
(53) If you do not PAY, Jones will KILL you unless you ESCAPE.
(54) If he CATCHES you, Jones will KILL you unless you PAY.
(55) Provided that he has made a BET, Jones is HAPPY if and only if his horse WINS.
(56) If neither JAY nor KAY comes home this weekend, we shall not stay HOME unless we are SICK.
(57) If you MAKE an appointment and do not KEEP it, then I shall be ANGRY unless you have a good EXCUSE.
(58) If I am not FEELING well this weekend, I will not GO out unless it is WARM and SUNNY.
(59) If JAY will go only if KAY goes, then we will CANCEL the trip unless KAY goes.
(60) If KAY will come to the party only if JAY does not come, then provided we WANT Kay to come we should DISSUADE Jay from coming.
(61) If KAY will go only if JAY does not go, then either we will CANCEL the trip or we will not INVITE Jay.
(62) If JAY will go only if KAY goes, then we will CANCEL the trip unless KAY goes.
(63) If you CONCENTRATE only if you are INSPIRED, then you will not SUCCEED unless you are INSPIRED.
(64) If you are HAPPY only if you are DRUNK, then unless you are DRUNK you are not HAPPY.
(65) In order to be ADMITTED to law school, it is necessary to have GOOD grades, unless your family makes a large CONTRIBUTION to the law school.

(66) I am HAPPY only if my assistant is COMPETENT, but if my assistant is COMPETENT, then he/she is TRANSFERRED to a better job and I am not HAPPY.
(67) If you do not CONCENTRATE well unless you are ALERT, then you will FLY an airplane only if you are SOBER; provided that you are not a MANIAC.
(68) If you do not CONCENTRATE well unless you are ALERT, then provided that you are not a MANIAC you will FLY an airplane only if you are SOBER.
(69) If you CONCENTRATE well only if you are ALERT, then provided that you are WISE you will not FLY an airplane unless you are SOBER.
(70) If you CONCENTRATE only if you are THREATENED, then you will not PASS unless you are THREATENED – provided that CONCENTRATING is a necessary condition for PASSING.
(71) If neither JAY nor KAY is home this weekend, we will go to the BEACH; otherwise, we will STAY home.

Solution

(1) $R \wedge J$
(2) $\neg R \wedge W$
(3) $J \wedge K$
(4) $D \wedge \neg H$
(5) $T \wedge \neg Q$
(6) $R \wedge (J \wedge K)$
    R: Jay and Kay are roommates
    J: Jay hates Kay
    K: Kay hates Jay
(7) $(J \wedge K) \wedge (H \wedge N)$
    J: Jay is a Republican; K: Kay is a Republican
    H: Jay hates Nixon; N: Kay hates Nixon
(8) $K \rightarrow A$
(9) $G \rightarrow T$
(10) $C \vee S$
(11) $F \vee D$
(12) $\neg(J \wedge K)$
(13) $(J \vee K) \wedge \neg(J \wedge K)$
(14) $R \vee (S \wedge C)$
(15) $(R \vee O) \wedge \neg S$
(16) $J \rightarrow S$
(17) $\neg J \rightarrow \neg S$
(18) $C \rightarrow \neg F$
(19) $(L \wedge H) \rightarrow G$
(20) $\neg(L \wedge H) \rightarrow \neg G$
(21) $\neg J \wedge \neg K [or : \neg(J \vee K)]$
(22) $L \wedge (\neg T \wedge \neg B) [or : L \wedge \neg(T \vee B)]$
(23) $\neg P \rightarrow \neg G$
(24) $\neg\neg J \rightarrow \neg K [J \rightarrow \neg K]$
(25) $\neg(W \wedge R) \rightarrow \neg S$
(26) $\neg S \rightarrow B$

(27) $\neg G \to \neg F$
(28) $\neg(L \wedge H) \to \neg G$
(29) $H \to A$
(30) $\neg F \to \neg P$
(31) $\neg(R \wedge I) \to \neg P$
(32) $(\neg F \to \neg P) \wedge (F \to P)$
(33) $(H \to A) \wedge \neg(\neg H \to \neg A)$
(34) $(\neg T \to \neg A) \wedge \neg(T \to A)$
(35) $(\neg(G \wedge A) \to \neg M) \wedge \neg[(G \wedge A) \to M]$
(36) $(\neg(E \wedge \neg M) \to \neg B) \wedge [(E \wedge \neg M) \to B]$
(37) $((C \wedge G) \to A) \wedge \neg[\neg(C \wedge G) \to \neg A]$
(38) $(R \to B) \wedge (\neg R \to J)$
(39) $((J \wedge K) \to B) \wedge [\neg(J \wedge K) \to S]$
(40) $(\neg I \to J) \wedge (I \to S)$
(41) $(J \wedge K) \to (D \wedge C)$
(42) $(\neg J \wedge \neg K) \to (P \vee C)$
(43) $(\neg S \wedge \neg T) \to (J \wedge K)$
S: Jay is sick; T: Kay is sick;
J: Jay will go to the beach; K: Kay will go to the beach.
(44) $(A \to D) \wedge (\neg A \to D)$
A: I do (what ever action is being discussed);
D: I am damned.
(45) $(S \to \neg E) \wedge (F \to \neg E)$
(46) $(N \to T) \wedge (T \to N)$
(47) $(\neg T \to \neg W) \to \neg S$
(48) $\neg S \to (\neg A \to \neg P)$
(49) $\neg H \to (S \to P)$
(50) $\neg D \to (C \to P)$
(51) $\neg E \to (\neg S \to \neg P)$
(52) $I \to (\neg G \to \neg F)$
(53) $\neg P \to (\neg E \to K)$
(54) $C \to (\neg P \to K)$
(55) $B \to [(W \to H) \wedge (\neg W \to \neg H)]$
(56) $(\neg J \wedge \neg K) \to (\neg S \to \neg H)$
(57) $(M \wedge \neg K) \to (\neg E \to A)$
(58) $\neg F \to [\neg(W \wedge S) \to \neg G]$
(59) $(\neg K \to \neg J) \to (\neg K \to C)$
(60) $(\neg\neg J \to \neg K) \to (W \to D)$
(61) $(\neg J \to \neg K) \to (C \vee \neg I)$
(62) $(\neg K \to \neg J) \to (\neg K \to C)$
(63) $(\neg I \to \neg C) \to (\neg I \to \neg S)$
(64) $(\neg D \to \neg H) \to (\neg D \to \neg H)$
(65) $\neg C \to (\neg G \to \neg A)$
(66) $(\neg C \to \neg H) \wedge (C \to [T \wedge \neg H])$
(67) $\neg M \to [(\neg A \to \neg C) \to (\neg S \to \neg F)]$
(68) $(\neg A \to \neg C) \to [\neg M \to (\neg S \to \neg F)]$
(69) $(\neg A \to \neg C) \to [W \to (\neg S \to \neg F)]$
(70) $(\neg C \to \neg P) \to [(\neg T \to \neg C) \to (\neg T \to \neg P)]$

(71) $((\neg J \wedge \neg K) \rightarrow B) \wedge [\neg(\neg J \wedge \neg K) \rightarrow S]$

□

Exercise 45:

Translate each of the following statements into propositional logic. You have to specify the propositional variables you are using and their corresponding proposition in english (e.g., the translation of "I'm happy if Bob is at the party" is $B \rightarrow H$, where H stand for "I am happy", B stands for "Bob is at the party")

(1) If Ann will go to the party only if she has not to work, then, if John helps Ann in finishing the work Ann will come to the party;
(2) If neither Ann nor Bob is home this weekend, then we will go to the beach otherwise, we will stay home.
(3) I will be happy if at the end of the semester I will pass at least 2 exams among Deep Learning, Databases, and Probability.

Solution

(1)

$$A = \text{Ann will go to the party}$$
$$W = \text{Ann has to work}$$
$$H = \text{John Helps Ann in finishing the work}$$

$$(A \rightarrow \neg W) \rightarrow (H \rightarrow A)$$

A very common error is to translate "Ann will go to the party only if she has not to work" with the formula $\neg W \rightarrow A$. However this encodes the proposition "If Ann has not to work then she will go to the parti", Notice that the fact that Ann has not to work and She will go to the cinema, is consistent with the proposition "Ann will go to the party only if she has not to work".

(2)

$$A = \text{Ann stays at homw this weekend}$$
$$B = \text{Bob stays at homw this weekend}$$
$$G = \text{we will go to the beach}$$
$$H = \text{we will stay home}$$

$$(\neg A \wedge \neg B \rightarrow G) \wedge (\neg(\neg A \wedge \neg B) \rightarrow H)$$

which is equivalent to

$$(\neg A \wedge \neg B \rightarrow G) \wedge (A \vee B \rightarrow H)$$

(3)

$$H = \text{I will be happy}$$
$$D = \text{I will pass Deep Learning}$$
$$B = \text{I will pass Data Bases}$$
$$P = \text{I will pass Probability}$$

$$(D \wedge B) \vee (D \wedge F) \vee (B \wedge P) \rightarrow H$$

Some student proposed the more complex but equivalent translation.

$$(D \wedge B \wedge \neg P) \vee (D \wedge P \wedge \neg B) \vee (B \wedge P \wedge \neg D) \vee (D \wedge B \wedge P) \rightarrow H$$

This is ok, but in general it is a good practice to use the *simplest* formalization. □

5.1. Problem Solving with Propositional Logic. Exercise 46:
Determine the validity or invalidity of the following argument:
If Alice is elected class-president, then either Betty is elected vice-president, or Carol is elected treasurer but not both. Betty is elected vice-president. Therefore if Alice is elected class-president, then Carol is not elected treasurer. **Solution**

We use the following propositional variables for each atomic sentence.

A	-	Alice is elected class-president
B	-	Betty is elected vice - president
C	-	Carol is elected treasurer

The translation of each complex sentence of the argument is the following:

$A \rightarrow ((B \wedge \neg C) \vee (\neg B \vee C))$	If Alice is elected class-president, then either Betty is elected vice- president, or Carol is elected treasurer but not both.
B	Betty is elected vice-president
$A \rightarrow \neg C$	if Alice is elected class-president, then Carol is not elected treasurer.

The logical consequence corresponding to the argument is

$$A \rightarrow ((B \wedge \neg C) \vee (\neg B \vee C)), B \models A \rightarrow \neg C \tag{17}$$

In order to see if (17) holds, we can try to find an intepretation $\mathcal{I}$ for A,B and C that satisfies the premises and falsifies the conclusion Looking at the truth table of we have:

A	B	C	A	→	((	B	∧	¬	C	)	∨	(	¬	B	∧	C	))	B	A	→	¬	C
T	T	T	T	F		T	F	F	T	F		F	T	F	T			T	T	F	F	T
T	T	F	T	T		T	T	T	F	T		F	T	F	F			T	T	T	T	F
T	F	T	T	T		F	F	F	T	T		T	F	T	T			F	T	F	F	T
T	F	F	T	F		F	F	T	F	F		T	F	F	F			F	T	T	T	F
F	T	T	F	T		T	F	F	T	F		F	T	F	T			T	F	T	F	T
F	T	F	F	T		T	T	T	F	T		F	T	F	F			T	F	T	T	F
F	F	T	F	T		F	F	F	T	T		T	F	T	T			F	F	T	F	T
F	F	F	F	T		F	F	T	F	F		T	F	F	F			F	F	T	T	F

From the above truth table one can see that every time the two premises are true (highlighted in red background) the consequence is also true. This means that the logical argument is valid. □

Exercise 47:
Test the validity of the following arguments.

> James is either a policeman or a footballer (but not both). If he is a policeman, then he has big feet. James has not got big feet so he is a footballer.

Solution We use the following propositional variables for each atomic sentence.

p - James is a policeman
f - James is a footballer
b - James has big feet.

Then the argument is formalized by the following logical consequence:

$$(p \vee f) \wedge (\neg p \vee \neg f) \wedge (p \rightarrow b) \wedge \neg b \models f$$

Let us check the validity of the first argument by building a truth table for

b	f	p	p	$\vee$	f	$\neg$	p	$\vee$	$\neg$	f	p	$\rightarrow$	b	$\neg$	b	f
T	T	T	T	T	T	F	T	F	F	T	T	T	T	F	T	T
T	T	F	F	T	T	T	F	T	F	T	F	T	T	F	T	T
T	F	T	T	T	F	F	T	T	T	F	T	T	T	F	T	F
T	F	F	F	F	F	T	F	T	T	F	F	T	T	F	T	F
F	T	T	T	T	T	F	T	F	F	T	T	F	F	T	F	T
F	T	F	F	T	T	T	F	T	F	T	F	T	F	T	F	T
F	F	T	T	T	F	F	T	T	T	F	T	F	F	T	F	F
F	F	F	F	F	F	T	F	T	T	F	F	T	F	T	F	F

From the above truth table one can see that every time the two premises are true (highlighted in red background) the consequence is also true. This means that the logical argument is valid. □

Exercise 48:

Let p stand for the proposition "I bought a lottery ticket" and q for "I won the jackpot". Express the following as natural English sentences:

(1) $\neg p$
(2) $p \vee q$
(3) $p \wedge q$
(4) $p \rightarrow q$
(5) $\neg p \rightarrow \neg q$
(6) $\neg p \vee (p \wedge q)$

Exercise 49:

Formalise the following in terms of the propositional variables r, b, and w, first expressing in english that proposition they are intended to represent.

(1) Berries are ripe along the path, but rabbits have not been seen in the area.
(2) Rabbits have not been seen in the area, and walking on the path is safe, but berries are ripe along the path.
(3) If berries are ripe along the path, then walking is safe if and only if rabbits have not been seen in the area.
(4) It is not safe to walk along the path, but rabbits have not been seen in the area and the berries along the path are ripe.
(5) For walking on the path to be safe, it is necessary but not sufficient that berries not be ripe along the path and for rabbits not to have been seen in the area.
(6) Walking is not safe on the path whenever rabbits have been seen in the area and berries are ripe along the path.

Exercise 50:

Formalise these statements and determine (with truth tables or otherwise) whether they are consistent (i.e. if there are some truth assignment to the propositional variables that make all them true):

> *The system is in a multiuser state if and only if it is operating normally. If the system is operating normally, the kernel is functioning. Either the kernel is not functioning or the system is in interrupt mode. If the system is not in multiuser state, then it is in interrupt mode. The system is not in interrupt mode.*

Solution Let us find the propositions in the text and declare the propositional variables to represent them

> *The system is in a multiuser state if and only if it is operating normally.*
> *If the system is operating normally, the kernel is functioning.*
> *Either the kernel is not functioning or the system is in interrupt mode.*
> *If the system is not in multiuser state, then it is in interrupt mode.*
> *The system is not in interrupt mode.*

$$\begin{aligned} M &= \text{The system is in a multiuser state} \\ N &= \text{The system is operating normally} \\ K &= \text{Kernel is functioning} \\ I &= \text{The system is in interrupt mode} \end{aligned}$$

Let us now translate the above sentences in propositional logic:

- The system is in a multiuser state if and only if it is operating normally.

$$M \leftrightarrow N$$

- If the system is operating normally, the kernel is functioning.

$$N \rightarrow K$$

- Either the kernel is not functioning or the system is in interrupt mode.

$$\neg K \vee I$$

- If the system is not in multiuser state, then it is in interrupt mode.

$$\neg M \rightarrow I$$

- The system is not in interrupt mode.

$$\neg I$$

I	K	M	N	M	↔	N	N	→	K	¬	K	∨	I	¬	M	→	I	¬	I
T	T	T	T	T	T	T	T	T	T	F	T	T	T	F	T	T	T	F	T
T	T	T	F	T	F	F	F	T	T	F	T	T	T	F	T	T	T	F	T
T	T	F	T	F	F	T	T	T	T	F	T	T	T	T	F	T	T	F	T
T	T	F	F	F	T	F	F	T	T	F	T	T	T	T	F	T	T	F	T
T	F	T	T	T	T	T	T	F	F	T	F	T	T	F	T	T	T	F	T
T	F	T	F	T	F	F	F	T	F	T	F	T	T	F	T	T	T	F	T
T	F	F	T	F	F	T	T	F	F	T	F	T	T	T	F	T	T	F	T
T	F	F	F	F	T	F	F	T	F	T	F	T	T	T	F	T	T	F	T
F	T	T	T	T	T	T	T	T	T	F	T	F	F	F	T	T	F	T	F
F	T	T	F	T	F	F	F	T	T	F	T	F	F	F	T	T	F	T	F
F	T	F	T	F	F	T	T	T	T	F	T	F	F	T	F	F	F	T	F
F	T	F	F	F	T	F	F	T	T	F	T	F	F	T	F	F	F	T	F
F	F	T	T	T	T	T	T	F	T	F	T	T	F	F	T	T	F	T	F
F	F	T	F	T	F	F	F	T	T	F	T	T	F	F	T	T	F	T	F
F	F	F	T	F	F	T	T	F	F	T	F	T	F	T	F	F	F	T	F
F	F	F	F	F	T	F	F	T	F	T	F	T	F	T	F	F	F	T	F

Notice that there is no assignment that satisfy all the five formulas. Thererfore the set of formulas is inconsistent.

A shorter way to prove incinsistency would have been by propositional resolution. We first have to transform the four formulas in clauses, obtaining:

$$\{\neg M, N\}, \{\neg N, M\}, \{\neg N, K\}, \{\neg K, I\}, \{M, I\}, \{\neg I\}$$

By repeated applications of unit-propagation we can obtaine the empty clause

$$\begin{aligned}
&\{\neg M, N\}, \{\neg N, M\}, \{\neg N, K\}, \{\neg K, I\}, \{M, I\}, \{\neg I\} \\
&\{\neg M, N\}, \{\neg N, M\}, \{\neg N, K\}, \{\neg K\}, \{M\} && \text{By unit propagation on } \{\neg I\} \\
&\{N\}, \{\neg N, K\}, \{\neg K\} && \text{By unit propagation on } \{M\} \\
&\{K\}, \{\neg K\} && \text{By unit propagation on } \{N\} \\
&\{\} && \text{By unit propagation on } \{K\}
\end{aligned}$$

Since we derive the empty clauses the initial set of clauses must be inconsistent. □

Exercise 51:

Five friends (Abby, Heather, Kevin, Randy and Vijay) have access to an on-line chat room. We know the following are true:

(1) Either K or H or both are chatting.
(2) Either R or V but not both are chatting.
(3) If A is chatting, then R is chatting.
(4) V is chatting if and only if K is chatting.
(5) If H is chatting, then both A and K are chatting.

Represent the above facts in CNF (set of clauses) Notice that there are sentences that correspond to more than one clause.

Solution

(1) $K \vee H$,
(2) $R \vee V$, $\neg R \vee \neg V$,
(3) $\neg A \vee B$,

(4) $\neg V \vee K$, $V \vee \neg K$,
(5) $\neg H \vee A$, $\neg H \vee K$,

□

Exercise 52:

Imagine that a logician puts four cards on the table in front of you. Each card has a number on one side and a letter on the other. On the uppermost faces, you can see E,K,4,and 7. He claims that if a card has a vowel on one side, then it has an even number on the other. Which cards do you have to turn over to check this? Explain why.

Solution To check that the logician states the truth we can check it for every single card The statemen is an implication

$$wowel \rightarrow \text{even}$$

which is true if either theq premise is false or the conclusion is true. In the first card we see a vawel, therefore the premise is true (it is not false) and therefore we have to turn the card to check if the back is even. The second card shows a consonanto, which is not a vawel, which guarantees that $wowel \rightarrow$ even. independently from what there is on the back. The third card shows a even number, this implies tha the conseuqence of $wowel \rightarrow$ even is true,. This guarantee that $wowel \rightarrow$ even holds without any further information. Finally, the forth card shows an odd number, therefore the conclusion of $wowel \rightarrow$ even is false, to chek that the implication is true we have to see if the premise is also falsw, which means that we have to turn the card. □

Solution(alternative) To check if the formula $vowel \rightarrow even$ is true you have to check that in all the possible models, given what you know, the formula is true. If there are models in which the formula is false, you have to turn some card in order to acquire some knowledge that will exclude such a model.

Let V_i and E_i for $i \in \{1, 2, 3, 4\}$ be the propositional variables that express the fact that the i-th card has a vowel in one side and an even number on the other side.

The statement of the logician can be formalized by the formula:

$$\bigwedge_{i=1}^{4} V_i \rightarrow E_i$$

What you know on the basis on what you see on the table is

$$V_1 \wedge \neg V_2 \wedge E_3 \wedge \neg E_4 \tag{18}$$

The models that satisfies your knowledge i,e, (18) are:

V_1	E_1	V_2	E_2	V_3	E_3	V_4	E_4	$V_1 \to E_1$	$V_2 \to E_2$	$V_3 \to E_3$	$V_4 \to E_4$
1	0	0	0	0	1	0	0	0	1	1	1
1	0	0	0	0	1	1	0	0	1	1	0
1	0	0	0	1	1	0	0	0	1	1	1
1	0	0	0	1	1	1	0	0	1	1	0
1	0	0	1	0	1	0	0	0	1	1	1
1	0	0	1	0	1	1	0	0	1	1	0
1	0	0	1	1	1	0	0	0	1	1	1
1	0	0	1	1	1	1	0	0	1	1	0
1	1	0	0	0	1	0	0	1	1	1	1
1	1	0	0	0	1	1	0	1	1	1	0
1	1	0	0	1	1	0	0	1	1	1	1
1	1	0	0	1	1	1	0	1	1	1	0
1	1	0	1	0	1	0	0	1	1	1	1
1	1	0	1	0	1	1	0	1	1	1	0
1	1	0	1	1	1	0	0	1	1	1	1
1	1	0	1	1	1	1	0	1	1	1	0

From the truth table above, one can see that there are models that satisfies (18) (the knowledge we have) in which $V_1 \to E_1$ and $V_4 \to E_4$ are both true and false. This means that we are uncertain about the truth value of these formulas. since neither (18) $\models V_1 \to E_1$ nor (18) $\models \neg(V_1 \to E_1)$ (and the same for $V_4 \to E_4$).

If we would know the truth value of E_1 and V_4, we would be certain about the truth value of $V_1 \to E_1$ and $V_4 \to E_4$. Indeed notice that

- $V_1 \to E_1$ is true if and only if E_1 is true;
- $V_4 \to E_4$ is true if and only if V_4 is false.

Therefore to know if the logician said the truth we have to turn the first and the forth card.

An intuitive reasoning is the following. A statement $V_i \to E_i$ is true whenever the premise is true then the conclusion is also true, or equivalently that whenever the conclusion is false, also the premise is false. Therefore, to check that the logician says the truth, you have to turn the E (vowel is true) and the 7 (even is false) □

Exercise 53:

Formalize the following puzzle in a set Γ of propositional formulas and show that the answer is a formula ϕ that logically follows from Γ.

> A very special island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet two inhabitants: Marge and Zoey.
> (1) Marge says, "Zoey and I are both knights or both knaves."
> (2) Zoey claims, "Marge and I are the same."
> Can you determine who is a knight and who is a knave?

Solution We first define the propositional variables to represent the proposition we need to formalize the puzzle

M	Marge is a knight
Z	Zoey is a knight

Clearly since in the island if one person is not a knight it must be a knave, we have that $\neg M$ means that Marge is a knave and $\neg Z$ means that Zoey is a knave. Now we can formalize the knowledge encoded in the two sentences. We don't know if the two statements said by Marge and Zoey are true or false, but we know that if the speaker is a knight the sentence is true, and if the speaker is a knave the sentence must be false. So from the two sentences we can get the following facts:

$$\begin{aligned} M &\rightarrow (M \wedge Z) \vee (\neg M \wedge \neg Z) \\ \neg M &\rightarrow \neg((M \wedge Z) \vee (\neg M \wedge \neg Z)) \\ Z &\rightarrow (M \leftrightarrow Z) \\ \neg Z &\rightarrow \neg(M \leftrightarrow Z) \end{aligned}$$

We can simplify the above four formulas by reducing them in CNF

$$\{\neg M, Z\} \tag{19}$$

$$\{M, Z\} \tag{20}$$

$$\{\neg Z, M\} \tag{21}$$

$$\{Z, M\} \tag{22}$$

By applying the resolution rule to (19) and (20) we derive the clause $\{Z\}$, and then by unit propagation on (21) we obtain the clause $\{M\}$. WHich implies that both Marge and Zoey are knights. □

Exercise 54:

Consider the set $S = \{0, 1, 2, 3\}^8$ of all the strings of length equal to 8 composed of $0, 1, 2, 3$, and the following subset $T \subset S$:

$$T = \left\{ \begin{array}{c} 00000000, 00000011, 00001111, 00001122, \\ 00111111, 00111122, 00112222, 00112233 \end{array} \right\}$$

(1) Define a set of propositional variable $\mathcal{P}$ such that every truth assigment $\mathcal{I}$ (interpretation) of $\mathcal{P}$ is one-to-one mapped into a string $s(\mathcal{I}) \in S$;
(2) What is the cardinality of $\mathcal{I}$ and the cardinality of S?
(3) Using the set of propositional variables in $\mathcal{P}$, write a formula ϕ such that $\mathcal{I} \models \phi$ if and only if $m(\mathcal{I}) \in T$.

Solution

(1) We have various possibilities here. A first alternative is to add the set of propositional variables:

$$p_{ij} \qquad \text{for } 1 \leq i \leq 8 \text{ and } 0 \leq j \leq 3$$

and interpret p_{ij} as "the digit j is in position i". and add the axioms

$$\bigwedge_{i=1}^{8} \bigvee_{j=0}^{3} p_{ij} \qquad \text{At every place } i \text{ there is at least one digit} \tag{23}$$

$$\bigwedge_{i=1}^{8} \bigwedge_{j=0}^{3} \bigwedge_{k=j+1}^{3} \neg(p_{ij} \wedge p_{ik}) \qquad \text{At every place } i \text{ there is at most one digit} \tag{24}$$

In this case the set $\mathbb{I}$ of interpretations must be restricted to the ones that satisfies (23) and (24).

Alternatively, we can introduce less propositions

$$p_i, q_i \qquad \text{for } 1 \leq i \leq 8$$

and interpred p_i as "in position i that is a digit j such that $j \div 2 = 1$ q_i as "in position i that is a digit j such that $i\%2 = 1$. Notice that with this interpretation we have that we can represent the fact that in position i there is a digit j as follows:

$\neg p_i \wedge \neg q_i$	In position i there is a 0
$\neg p_i \wedge q_i$	In position i there is a 1
$p_i \wedge \neg q_i$	In position i there is a 2
$p_i \wedge q_i$	In position i there is a 3

Since the above 4 formulas are exclusive (i.e. one and only one can be true in any interpretation) we don't need to add any additional axiom. In this case all the interpretations can be considered.

(2) Since we have to define a one to one mapping between $\mathbb{I}$ and S their cardinality must be the same, i.e, 4^8.

(3) If we use the first language we can represent the set T with the formula

$$\bigvee_{\boldsymbol{t} \in T} \bigwedge_{i=1}^{8} p_{it_i}$$

where t_i is the i-th digit of $\boldsymbol{t}$. If we use the second langauge we can represent the set T with the formula

$$\bigvee_{\boldsymbol{t} \in T} \bigwedge_{i=1} \circ_{t_i \div 2} p_i \wedge \circ_{t_i \% 2} q_i$$

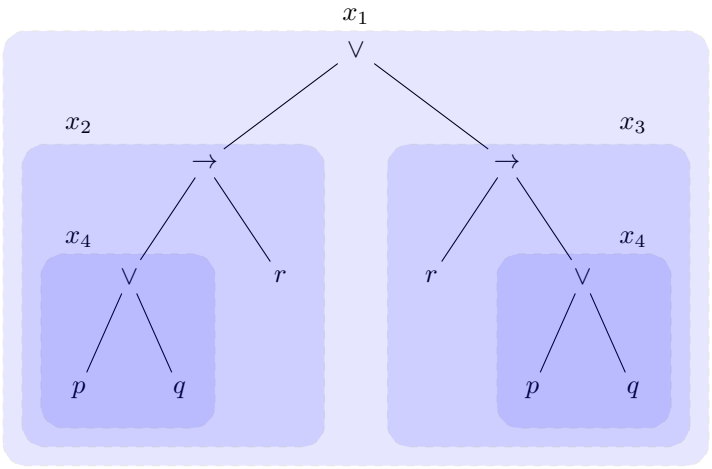
where $\circ_0$ is equal to the emtpy string $\circ_1$ is equal to $\neg$.

□

Exercise 55:
Provide the Tseitin's Transformation of the following formula:

$$((p \vee q) \rightarrow r) \vee (r \rightarrow (p \vee q))$$

Solution The Tseitin's tranformation introduces one new propositional variable for all the non atomic subformula of the original formula. The set of non atomic subformulas of $((p \vee q) \rightarrow r) \vee (r \rightarrow (p \vee q))$ (including itself) with the associated new propositional variables are:



x_1	$((p \vee q) \rightarrow r) \vee (r \rightarrow (p \vee q))$
x_2	$((p \vee q) \rightarrow r)$
x_3	$(r \rightarrow (p \vee q))$
x_4	$(p \vee q)$

We then define the follwong clauses

$$x_1 \equiv x_2 \vee x_3$$
$$x_2 \equiv x_4 \rightarrow r$$
$$x_3 \equiv r \rightarrow x_4$$
$$x_4 \equiv p \vee q$$

and tranform them in clausal form

$$(\neg x_1, x_2, x_3), (\neg x_2, x_1), (\neg x_3, x_1)$$
$$(\neg x_2, \neg x_4, r), (x_4, x_2), (\neg r, x_2)$$
$$(\neg x_3, \neg r, x_4), (r, x_3), (\neg x_4, x_2)$$
$$(\neg x_4, p, q), (\neg p, x_4), (\neg q, x_4)$$

□

Exercise 56:

Alice (F), Bob (M), Craig (M), and Donna (F) are four friends that want to take a tour with their motorbikes. Everybody can decide either to go with somebody else or to ride a bike alone. Use propositional logic to formulate the problem of finding all the possible configurations for the tour.

Solution We introduce the following propositional variables

- $ride_X$ for X drives a bike for $X \in \{A, B, C, D\}$
- $pass_{XY}$ for X give a pass to Y, for $X, Y \in \{A, B, C, D\}$ nd $X \neq Y$.

We add the following axioms:

$$\bigwedge_X \left(ride_X \vee \bigvee_{Y \neq X} pass_{YX} \right) \quad \text{Each friend, either rides or get a lift}$$

$$\bigwedge_{X \neq Y} (pass_{XY} \rightarrow ride_X) \quad \text{If } Y \text{ get a lift from } X \text{ then } X \text{ rides the bike}$$

$$\bigwedge_{X \neq Y} (rides_X \rightarrow \neg pass_{YX}) \quad \text{If } X \text{ rides he/she does not get a lift}$$

$$\bigwedge_{X \neq X' \neq Y} (pass_{XY} \rightarrow \neg pass_{X'Y}) \quad Y \text{ can get a lift from at most one person}$$

$$\bigwedge_{X \neq Y \neq Y'} (pass_{XY} \rightarrow \neg pass_{XY'}) \quad X \text{ can give a lift to at most one person}$$

A python program that computes all the solutions is the following

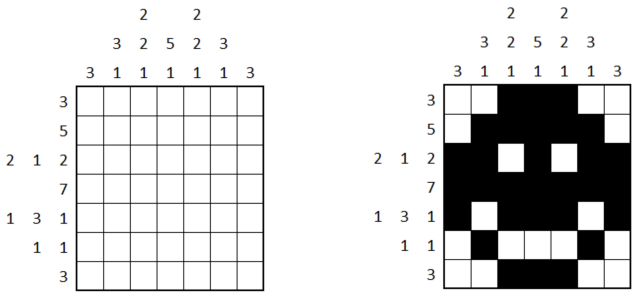
LISTING 3.1. Pysat code that compute all the configurations

```
from sympy import *

Friends = ["A","B","C","D"]

Rides = {X:Symbol(f"Rides_{X}") for X in Friends}
Pass = {(X,Y):Symbol(f"Pass_{X}{Y}") for X in Friends for Y in Friends if X != Y}

phi1 = And(*[Rides[X] | Or(*[Pass[Y,X] for Y in Friends if Y != X]) for X in Friends])
phi2 = And(*[Pass[X,Y] >> Rides[X] for X in Friends for Y in Friends if X != Y])
phi3 = And(*[Pass[X,Y] >> ~Rides[Y] for X in Friends for Y in Friends if X != Y])
phi4 = And(*[Pass[X,Y] >> ~Pass[X1,Y] for X in Friends for X1 in Friends for Y in Friends
             if X1 != X and Y != X1 and Y != X])
phi5 = And(*[Pass[X,Y] >> ~Pass[X,Y1] for X in Friends for Y in Friends for Y1 in Friends
             if X != Y and X != Y1 and Y != Y1])

for m in satisfiable(phi1 & phi2 & phi3 & phi4 & phi5, all_models=True):
    print(m)
```

□

Exercise 57:

A nonogram is a grid with a series of numbers on the left of each row and above each column of the grid. Each of these numbers represents a consecutive run of shaded spaces in the corresponding row or column. Each consecutive run is separated from other runs by at least one empty space. The puzzle is complete when all of the numbers have been satisfied. See for instance the example belwo and the corresponding solution



Describe a possible encoding of a generic configuration of a nanogram in propositional logic so that the solution can be obtained by running a sat solver. Show how you encode the clue contained in the second column of the above example.

Solution Let m, n be the size of the grid of a nanogram. We consider $n \times m$ propositional variables p_{rc} for $1 \leq r \leq n$ and $1 \leq c \leq m$. p_{rc} represents the proposition that the cell at row r and column c is black. A clue $n_1, \ldots, n_k$ associated to the r-th row can be modelled by adding $k \cdot m$ propositional variables $rclue(r, j)_c$ for $j = 1, \ldots, k$ and $c = 1, \ldots, m$. $rclue(r, j)_c$ is true when the run associated to the j-th element of the clue of r starts at column c. Using this set of proposition, we can model the constraitns that must be satisfied by a solution of a nanogram. For every row r if $n_1, \ldots, n_k$ is the associated clue, we add the following axioms:

$\sum_{c=1}^{m-n_j} rclue(r, j)_c = 1$	The j-th run of the clue of the r-th row starts at exaclty one column between 1 and n_j
$\bigwedge_{c+n_i \geq c'} rclue^r_{ic} \rightarrow \neg rclue^r_{(i+1)c'}$	The $j+1$-run at least one step after the end of the i-th run
$rclue(r, j)_c \rightarrow \bigwedge_{c'=c}^{c+n_j-1} p_{rc'}$	the next n_j cells after the start of the j-th run must be black
$rclue(r, j)_c \wedge rclue(r, j+1)_{c'} \rightarrow \bigwedge_{c''=c+n_i}^{c'-1} \neg p_{rc''}$	The cells between the end of the j-th run and the beginning of the $j+1$-th run are not black
$rclue^r_{1c} \rightarrow \bigwedge_{c'=1}^{c-1} \neg p_{rc'}$	The cells before the initial run are not black
$rclue^r_{kc} \rightarrow \bigwedge_{c'=c+n_k}^{m} \neg p_{rc'}$	The cells after the last run are not black

The clues for the columns can be formalized in an analogours using the propositional variables $cclue(c, j)_r$. □

Exercise 58:

Consider an undirected graph $G = (V, E)$ composed of a set of nodes $V = \{1, \ldots, n\}$ and a set of undirected edges between them. Formulate the problem of finding a path that visits all the nodes starting from a node s and ending in a node e without passing twice the same edge.

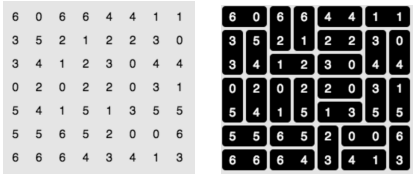
Solution Since the graph is undirected we have that $(i,j) \in E$ iff $(j,i) \in E$. Let $m = \frac{|E|}{2}$ the number or arcs in G. For every node $i \in V$ and for every $t \leq m$ we add the proposition p_i^t that means that i has been visited at time t Notice that $i \leq m$, since if we traverse more than m arcs it means that we have traversed one arc twice. We also add the "defined" proposition p_{ij}^t for every $(i,j) \in E$ that states that at time t we move from node i to node j. We use the propositional variable introduced above to formulate the requirements of the problem:

p_s^0		We start from s
p_e^m		We end at e
$\bigvee_{t=0}^{m} p_i$	$1 \leq i \leq n$	Every node is eventually visited
$\bigwedge_{1 \leq i < j \leq n} \neg(p_i^t \wedge p_j^t)$	$1 \leq t \leq m$	At every time we can be in at most one node
$p_i^t \rightarrow p_i^t \vee \bigvee_{(i,j) \in E} p_j^{t+1}$	$1 \leq t < m$	At every time stamp we can either stay in the node or move to a connected node
$p_{ij}^t \leftrightarrow (p_i^t \wedge p_j^{t+1} \vee p_i^{t+1} \wedge p_j^t)$	$(i,j) \in E$	we define p_{ij}^t as the proposition stating that at time t we traversed the edge i,j in either direction
$\bigwedge_{1 \leq t < t' < m} \neg(p_{ij}^t \wedge p_{ij}^{t'})$	$(i,j) \in E$	We cannot traverse the same arc twice

□

Exercise 59:

In the domino reconstruction problem you are given with board that shows only the numbers configuration of the domino tiles $(0,0),(0,1),\ldots,(6,5),(6,6)$ arranged in a 7×6 board (see figure on the left) and you have to reconstruct the position of each tile as shown in the the right picture below.



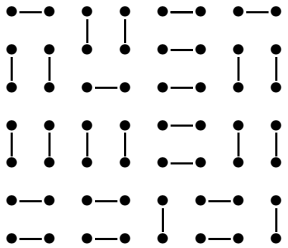
input reconstruction

In this exercise you have to propose an encoding of the domino reconstruction puzzle. In particular:

(1) define a set of propositional variables and say which property they encode.
(2) define a (set of) formula(s) that encodes the input (e.g., the picture on the left)
(3) define a (set of) formula(s) that imposes the constraints that every solution must satisfy
(4) explain how you can reconstruct the solution (picture on the right) from the interpretation returned by the sat solver.

Solution One can see the configuration shown on the right picture as a labelled graph, where the nodes are labelled with digits and the arc connects two digits

if they belong to the same tail. for instance the cofiguration shown on the right picture can be represented by the following graph:



To represent this configuration we can use $n = 6 \times 7$ we can use the proposition n_{ijkl} to denote that the node in position i, j is connected with the node in position k, l, with $1 \leq i, k \leq 6$ and $1 \leq j, l \leq 7$. We can impose the following constraints:

$$
\begin{aligned}
n_{ijkl} &\to n_{klij} && \text{symmetric connection} \\
\neg n_{ijkl} & && \text{for all } i,j,k,l \text{ such that } |i-k|+|j-l| \neq 1 \\
n_{ijkl} &\to \neg n_{ijk'l'} && \text{for all } i,j,k,l,k',l' \text{ with } k \neq k' \text{ and } l \neq l' \\
\bigvee_{|i-k|+|j-l|} & n_{ijkl} && \text{for all } i,j \text{ there is at least one connection}
\end{aligned}
$$

A second group of axioms will represent the labelling, We can use the proposition l_{ijn} to state that the node ij is labelled with the digit n. We have to impose that there are exactly 7 nodels labelled with the same digit

$$\sum_{i,j} l_{ijn} = 7 \qquad \text{for } 0 \leq n \leq 6$$

and that a node labelled with n should be connected with exactly one labelled with m for every n and m

$$\sum_{ijkl} (n_{ijkl} \wedge l_{ijn} \wedge l_{klm}) = 1 \tag{25}$$

□

CHAPTER 4

Decision Procedures in Propositional Logic

In the previous chapter we have presented four problems that involves propositional formulas and interpretations. In this chapter we analize them in details. We will see that the problems of checking the validity of a formula, and the problem of cheking that a formula is a logical consequence of a set of formulas can be rewritten in a problem of checking satisfiability of a set of formulas. Therefore, after a first part in which we will consider the problem of model checking, we will dedicate most of the chapter to the problem of satisfiability.

1. Model checking

The first and simplest problem that one has to solve in propositional logic is to compute the truth value of a formula with respecrt to a given interpretation $\mathcal{I}$.

Definition 4.1 (Model checking Problem). *Given a formula ϕ on a set of primitive propositions $\mathcal{P}$, and an interpretation (truth assignment) $\mathcal{I}$ of $\mathcal{P}$, the* model checking problem *is the problem of checking if*

$$\mathcal{I} \models \phi$$

An alternative formulation of the model cheking problem, is the problem of computing the truth value of a formula ϕ w.r.t., an interpretation $\mathcal{I}$ of the propositional variables of ϕ.

1.1. Decision procedure. A model checking decision procedure, MCDP is an algorithm that checks if a formula ϕ is satisfied by an interpretation $\mathcal{I}$. Namely

$$\begin{aligned} \text{MCDP}(\phi, \mathcal{I}) &= \textit{true} \quad \text{if and only if} \quad \mathcal{I} \models \phi \\ \text{MCDP}(\phi, \mathcal{I}) &= \textit{false} \quad \text{if and only if} \quad \mathcal{I} \not\models \phi \end{aligned}$$

The procedure of model checking returns for all inputs either *true* or *false* since for all models $\mathcal{I}$ and for all formulas ϕ, we have that either $\mathcal{I} \models \phi$ or $\mathcal{I} \not\models \phi$.

A simple way to check if $\mathcal{I} \models \phi$ is the following:

(1) Replace each occurrence of a propositional variables in ϕ with the truth value assigned by $\mathcal{I}$. I.e. replace each p with $\top$ is $\mathcal{I}(p) = \text{True}$ and with $\bot$ is $\mathcal{I}(p) = \text{False}$;
(2) Recursively apply the following rewriting rules:

$$\begin{array}{lllll} \neg\top \Rightarrow \bot & \top \wedge \top \Rightarrow \top & \top \vee \top \Rightarrow \top & \top \rightarrow \top \Rightarrow \top & \top \equiv \top \Rightarrow \top \\ \neg\bot \Rightarrow \top & \top \wedge \bot \Rightarrow \bot & \top \vee \bot \Rightarrow \top & \top \rightarrow \bot \Rightarrow \bot & \top \equiv \bot \Rightarrow \bot \\ & \bot \wedge \top \Rightarrow \bot & \bot \vee \top \Rightarrow \top & \bot \rightarrow \top \Rightarrow \top & \bot \equiv \top \Rightarrow \bot \\ & \bot \wedge \bot \Rightarrow \bot & \bot \vee \bot \Rightarrow \bot & \bot \rightarrow \bot \Rightarrow \top & \bot \equiv \bot \Rightarrow \top \end{array}$$

The complexity of this procedure is linear in the number of the connectives that appear in the formula ϕ. More specifically, the algorithm terminates in n steps where n is the number of connectives that appear in ϕ.

EXAMPLE 4.1. *Let us consider the formula* ϕ

$$\phi = p \vee (q \to r)$$

and the interpretation $\mathcal{I}$ *with*

$$\mathcal{I}(p) = \textit{False} \qquad \mathcal{I}(q) = \textit{False} \qquad \mathcal{I}(r) = \textit{True}$$

To check if $\mathcal{I} \models \phi$ *replace,* p*,* q*, and* r *in* ϕ *with* $\mathcal{I}(p)$*,* $\mathcal{I}(q)$ *and* $\mathcal{I}(r)$*, obtaining*

$$\bot \vee (\bot \to \top)$$

and then recursively apply the reduction rules

$$\begin{array}{l} \bot \vee (\bot \to \top) \\ \bot \vee \top \\ \top \end{array}$$

Since we obtain $\top$ *we can conclude that* $\mathcal{I} \models \phi$*.*

This method is not the most efficient one. One can notice that in many rewriting rules if one for the binary connectives $\wedge$,$\vee$, and $\to$, if one knows the value of one of the arguments it is not necessary to evaluate the other. For instance in evaluating $\phi \vee \psi$ if we already know that ϕ is true, we don't need to evaluate ψ, since we already know that the disjunction is true. Similarly in $\phi \wedge \psi$, if we have evaluated ϕ to be false, then we do not need to evaluate ψ since we already know, that independently from the evaluation of ψ the conjunction will be false. Notice that, this simplification is not possible for the $\equiv$ connective, since to evaluate the truth value of $\phi \equiv \psi$ knowing the truth value of ϕ is not sufficient to predict the truth value of the entire formula; we indeed have to evaluate ψ and check that the two truth values coincides.

The rewriting rules for $\wedge$, $\vee$ and $\to$ therefore can be rewritten as follows:

$$\begin{array}{lll} \top \wedge \top \Rightarrow \top & \top \vee * \Rightarrow \top & * \to \top \Rightarrow \top \\ * \wedge \bot \Rightarrow \bot & * \vee \top \Rightarrow \top & \top \to \bot \Rightarrow \bot \\ \bot \wedge * \Rightarrow \bot & \bot \vee \bot \Rightarrow \bot & \bot \to * \Rightarrow \top \end{array}$$

where "$*$" denotes either $\top$ or $\bot$ and therefore the corresponding expression does not need to be evaluated.

1.2. Formulas as boolean functions. A propositional formula ϕ on the set of propositional variables $\mathcal{P}$ specifies a boolean function

$$f_\phi : \{0,1\}^{|\mathcal{P}|} \to \{0,1\}$$

f_ϕ is a function that takes a vector of n 0-1 values (corresponding to a truth assignment) where $n = |\mathcal{P}|$ and returns either 1 or 0 depending of the fact that the corresponding truth assignment satisfies or does not satisfies the formula. More formally, if $\mathcal{P} = \{p_1, \ldots, p_n\}$, for every vector $\boldsymbol{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ we define

$\mathcal{I}_{\boldsymbol{x}}$ as the interpretation of $\mathcal{P}$ that assigns $\mathcal{I}_{\boldsymbol{x}}(p_i) = x_i$. With this convention the function f_ϕ can be defined

$$f_\phi(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \mathcal{I}_{\boldsymbol{x}} \models \phi \\ 0 & \text{if } \mathcal{I}_{\boldsymbol{x}} \not\models \phi \end{cases}$$

Notice that, if ϕ is equivalent to ψ then f_ϕ and f_ψ are the same functions.

PROPOSITION 4.1. *If $|\mathcal{P}| = n$ then there are at most 2^{2^n} boolean functions that can be defined in terms of a formulas with propositinal variables in $\mathcal{P}$.*

PROOF. The number of functions from a finite set M to another finite set N are n^m, where m and n are the number of elements (the cardinality) of the sets M and N respectively. Indeed for every element $x \in M$ whe have n different options to define $f(x)$ since $f(x) \in N$. So we have n options for the first element of M, other n options for the second elements, and so on. In total we have n^m alternatives. Therefore the number of distinct functions form $\{0,1\}^n$ to $\{0,1\}$ are equal to [1]

$$|\{0,1\}|^{|\{0,1\}^n|} = 2^{2^n}$$

Notice that $\{0,1\}$ contains 2 elements, and $\{0,1\}^n$ contains 2^n elements. □

The second question one should answer is the following: given an arbitrary function $f : \{0,1\}^n \to \{0,1\}$ is there a formula on the set $\mathcal{P}$ of n propositional fariables such that f_ϕ is equal to f?. The answer is yes, and it is stated in the following proposition.

PROPOSITION 4.2. *For every function $f : \{0,1\}^n \to \{0,1\}$, there is a formula ϕ with at most n propositional variables, such that f_ϕ is equal to f.*

PROOF. Let $p_1, \ldots, p_n$ be n distinct propositional variables. We define the formula ϕ as follows: For every $\boldsymbol{x} \in \{0,1\}^n$ let us define the formula $\phi_{\boldsymbol{x}}$ as the conjunction of p_i if $x_i = 1$ and $\neg p_i$ if $x_i = 0$. We can then define the formula

$$\phi = \bigvee_{f(\boldsymbol{x})=1} \phi_{\boldsymbol{x}}$$

Notice that the formula ϕ is satisfied by the assignments $\mathcal{I}_{\boldsymbol{x}}$ such that $f(\boldsymbol{x}) = 1$ and it is not satisfied by the assignments in which $f(\boldsymbol{x}) = 0$. Given the definition of f_ϕ we can easily see that f_ϕ and f coincides. □

2. Satisfiability checking

The propositional satisfiability problem (often called SAT) is the problem of determining whether a set of sentences in Propositional Logic is satisfiable. The problem is significant both because the question of satisfiability is important in its own right and because many other questions in Propositional Logic can be reduced to that of propositional satisfiability. In practice, many automated reasoning problems in Propositional Logic are first reduced to satisfiability problems and then by using a satisfiability solver. Today, SAT solvers are commonly used in hardware design, software analysis, planning, mathematics, security analysis, and many other areas.

[1] For every set S, $|S|$ indicates the number of elements of S, aka, the *cardinality* of S

2.1. Truth tables.

EXAMPLE 4.2. *Let*

$$\Gamma = \{p \vee q, p \vee \neg q, \neg p \vee q, \neg p \vee \neg q \vee \neg r, \neg p \vee r\}$$

We want to determine whether Γ *is satisfiable. So, we build a truth table for this case.*

p	q	r	$p \vee q$	$p \vee \neg q$	$\neg p \vee q$	$\neg p \vee \neg q \vee \neg r$	$\neg p \vee r$	Γ
0	0	0	0	1	1	1	0	0
0	0	1	0	1	1	1	1	0
0	1	0	1	0	1	1	1	0
0	1	1	1	0	1	1	1	0
1	0	0	1	1	0	1	0	0
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	0	0
1	1	1	1	1	1	0	1	0

In a truth table there is one row for each possible truth assignment. For each truth assignment, each of the sentences in Γ *is evaluated. If any sentence evaluates to 0, then* Γ *as a whole is not satisfied by the truth assignment. If a satisfying truth assignment is found, then* Γ *is determined to be satisfiable. If no satisfying truth assignment is found, then* Γ *is unsatisfiable. In this example, every row ends with* Γ *not satisfied. So the truth table method concludes that* Γ *is unsatisfiable (not satisfiable).*

The truth table method is complete because every truth assignment is checked. However, the method is impractical for all but very small problem instances. In our example with 3 propositional variables, there are $2^3 = 8$ rows. For a problem instance with 10 propositional variables, there are $2^{10} = 1024$ rows - still quite small for a modern computer. But as the number of propositions grow, the number of rows quickly overwhelms even the fastest computers. A more efficient method is needed.

3. Normal forms

The methods to check satisfiability that are more efficient that truth tables require that the input (set of) formula(s) have a specific structure. The description of such a structure is also called *Normal Form*. A normal form is a pattern for the structure of a formulas. A formula is in a specific normal form if its structure matches the pattern of the normal form. An important property of normal forms is that every formula can be transformed in an equivalent formula which is in normal form. This property is important since the semantics of the formula should be preserved by the transformation in normal form. There are many possible normal forms, e.g., negated normal form, conjunctive normal form, disjunctive normal form, deterministic disjunctive normal form ..., each of which is suitable for solving a specific problem. In this chapter, we introduce two normal forms, namely: Negated Normal Form and Conjunctive Normal Form, which are normal forms suitable for checking satisfiability. Further normal forms will be introduced in successive chapter where we will consider other problems such as for instance model counting and weighted model counting.

Let us first introduce some terminology. A *literal* is either an atomic formula or the negation of an atomic formula. Examples of literals are p, q, $\neg p$, $\neg q$. A literal is positive if it is an antom; a literal is negative if it is the negation of an atom.

3.1. Negation Normal Form (NNF).

DEFINITION 4.2 (Negation Normal Form). *A formula is in* negation normal form *(NNF) if negation connective ($\neg$) occurs only in literals; or equivalently if the negation connective occurs only in front of atomic formulas.*

EXAMPLE 4.3. *$p \wedge \neg q$ is in NNF, $\neg(p \wedge q)$ is not in NNF.*

Any formula can be transformed into an equivalent formula in NNF. by pushing the negation operator inwards, applying the following transformations that preserves the semantics.

$$\begin{aligned} \neg\neg\phi &\Rightarrow \phi \\ \neg(\phi \wedge \psi) &\Rightarrow \neg\phi \vee \neg\psi \\ \neg(\phi \vee \psi) &\Rightarrow \neg\phi \wedge \neg\psi \\ \neg(\phi \to \psi) &\Rightarrow \phi \wedge \neg\psi \\ \neg(\phi \leftrightarrow \psi) &\Rightarrow (\phi \vee \psi) \wedge (\neg\phi \vee \neg\psi) \end{aligned} \tag{26}$$

PROPOSITION 4.3. *If ϕ' is obtained by applying any sequence of transofmration rules in* (26) *to ϕ, then ϕ is equivalent to ϕ'.*

PROOF. Recall that ϕ is equivalent to ϕ' if and only if for every ionterpretation $\mathcal{I}$ of the propositional variables in ϕ and ϕ', $\mathcal{I} \models \phi$ if and only if $\mathcal{I} \models \phi'$. Let us start with the first rule:

$$\mathcal{I} \models \neg\neg\phi \text{ iff } \mathcal{I} \not\models \neg\phi \text{ iff } \mathcal{I} \models \phi$$

$$\begin{aligned} \mathcal{I} \models \neg(\phi \wedge \psi) &\text{ iff } \mathcal{I} \not\models \phi \wedge \psi \text{ iff } \text{ either } \mathcal{I} \not\models \phi \text{ or } \mathcal{I} \not\models \psi \\ &\text{iff } \text{ either } \mathcal{I} \models \neg\phi \text{ or } \mathcal{I} \models \neg\psi \text{ iff } \mathcal{I} \models \neg\phi \vee \neg\psi \end{aligned}$$

$$\begin{aligned} \mathcal{I} \models \neg(\phi \vee \psi) &\text{ iff } \mathcal{I} \not\models \phi \wedge \psi \text{ iff } \mathcal{I} \not\models \phi \text{ and } \mathcal{I} \not\models \psi \\ &\text{iff } \mathcal{I} \models \neg\phi \text{ and } \mathcal{I} \models \neg\psi \text{ iff } \mathcal{I} \models \neg\phi \wedge \neg\psi \end{aligned}$$

$$\begin{aligned} \mathcal{I} \models \neg(\phi \to \psi) &\text{ iff } \mathcal{I} \not\models \phi \to \psi \text{ iff } \mathcal{I} \models \phi \text{ and } \mathcal{I} \not\models \psi \\ &\text{iff } \mathcal{I} \models \phi \text{ and } \mathcal{I} \models \neg\psi \text{ iff } \mathcal{I} \models \phi \wedge \neg\psi \end{aligned}$$

$$\begin{aligned} \mathcal{I} \models \neg(\phi \leftrightarrow \psi) &\text{ iff } \mathcal{I} \not\models \phi \equiv \psi \\ &\text{iff either } \mathcal{I} \models \phi \text{ and } \mathcal{I} \not\models \psi \text{ or } \mathcal{I} \not\models \phi \text{ and } \mathcal{I} \models \psi \\ &\text{iff either } \mathcal{I} \models \phi \text{ and } \mathcal{I} \models \neg\psi \text{ or } \mathcal{I} \models \neg\phi \text{ and } \mathcal{I} \models \psi \\ &\text{iff either } \mathcal{I} \models \phi \wedge \neg\psi \text{ or } \mathcal{I} \models \neg\phi \wedge \psi \\ &\text{iff } \mathcal{I} \models (\phi \wedge \neg\psi) \vee (\neg\phi \wedge \psi) \\ &\text{iff } \mathcal{I} \models (\phi \vee \psi) \wedge (\neg\phi \vee \neg\psi) \end{aligned}$$

□

Notice that the presence of the $\leftrightarrow$ connective in the formula makes its negated normal form to to exponentially increase in the size. For instance the NNF of

$$\neg(p \leftrightarrow \neg(q \leftrightarrow r))$$

is equal to

$$(\neg p \wedge (p \leftrightarrow r)) \vee (p \wedge ((q \wedge \neg r) \vee (\neg q \wedge r)))$$

One final observation about Negated Normal Form concerns the fact that the "negative information" present in a NNF formula is not encoded exclusively by negative literals. For instance, an implication of the form $p \rightarrow q$ encodes some negative information despite the absence of negative literals. Indeed, when q is evaluated to false, p should also be false. This can be also seen by the fact that $p \rightarrow q$ is equivalent to $\neg p \vee q$. For this reason NNF is usally combined with other normal forms which remove the $\rightarrow$ and $\leftrightarrow$ connective by restriting $\phi \rightarrow \psi$ as $\neg\phi \vee \psi$ and $\phi \leftrightarrow \psi$ as $(\neg\phi \vee \psi) \wedge (\phi \vee \neg\psi)$.

3.2. Conjunctive Normal Form (CNF). Conjunctive Normal Form (CNF) is one of the most common form taken in input by algorithm for satisfiability checking. CNF is more specific than NNF (therefore every formula in CNF is also in NNF) and uses only the connectives $\vee$, $\wedge$ and $\neg$. We introduce the notion of clause.

A *clause* is a formula of the form $l_1 \vee l_2 \vee \cdots \vee l_n$ where l_i are literals. An example of clause is

$$a \vee \neg b \vee c$$

An interpretation $\mathcal{I}$ satisfies a clause $l_1 \vee \cdots \vee l_n$ if *there is* a literal l_i that is satisfied by $\mathcal{I}$. We also admit clauses that are composed of zero literals, which is called *empty clause*. By applying the same criteria for satisfiability of a clause, we have that an interpretation *never* satisfies the empty clause. For this reason the empty clasue can be consistently denoted by the $\bot$ symbol, representing the proposition that is always false. The clauses that contains only one literal nare called *unit clauses*. An interpretation satisfies a unit clause if and only if it assign to true (resp. false) the positive (resp. negative) literal it contains. Empty clauses and unit clauses play an important role in the satisfiability checking procedure, this is why they received a special name.

Definition 4.3 (Conjunctive normal form). *A formula is in* conjunctive normal form *if if is the conjunction clauses.*

Example 4.4. *The formula*

$$(p \vee \neg q \vee r) \wedge (q \vee \neg r) \wedge (\neg p \vee \neg q) \wedge r$$

is in CNF; it is composed by the following four clauses:

$$C_1 = p \vee \neg q \vee r$$
$$C_2 = q \vee \neg r$$
$$C_3 = \neg p \vee \neg q$$
$$C_4 = r$$

that contains 3, 3, 2, and 1 literal respectively.

The following is a procedure for reducing a formula in CNF can be obtained by applying
the following steps

(1) transform the implications $\rightarrow$ and the equivalence $\leftrightarrow$ with equivalent formulas using $\neg$, $\wedge$ and $\vee$

$$A \rightarrow B \Longrightarrow \neg A \vee B$$
$$A \leftrightarrow B \Longrightarrow (\neg A \vee B) \wedge (\neg B \vee A)$$

(2) push the negation operator $\neg$ in front of the atomic propositions (NNF)

$$\neg(A \vee B) \Longrightarrow \neg A \wedge \neg B$$
$$\neg(A \wedge B) \Longrightarrow \neg A \vee \neg B$$

(3) Distribute disjunction $\vee$ over conjunction $\wedge$

$$A \vee (B \wedge C) \Longrightarrow (A \vee B) \wedge (A \vee C)$$
$$(A \wedge B) \vee C) \Longrightarrow (A \vee C) \wedge (B \vee C)$$

FIGURE 1. CNF reduction rules.

Due to the fact that $\phi \vee \phi$ and $\phi \wedge \phi$ are equivalent to ϕ (identical operand) and the fact that $\phi \vee \psi$ and $\phi \wedge \psi$ are equivalent to $\psi \vee \phi$ $\psi \wedge \phi$ (commutativity) we can consider a clause as a *set of literals*, and a CNF formula as a *set of sets of literals*. For instance the previous example can be seen as the set that contains four sets (clauses)

$$\{\{p, \neg q, r\}, \{q, \neg r\}, \{\neg p, \neg q\}, \{r\}\}$$

PROPOSITION 4.4. *Every formula can be rewritten in an equivalent formula in CNF.*

As for the case of NNF we provide a set of rewriting rules, shown in Figure 1 that preserves the semantics (i.e., a formula is rewritten in an equivalent formula). To reduce a formula in CNF we therefore define a recursive algorithm CNF that computes the CNF of a formula ϕ by, first recursively compute the CNF of the sub-formulas of ϕ and then combine the results

EXAMPLE 4.5. *Let us transform the following formula in CNF*

$$\begin{aligned}
(p \rightarrow q) \wedge \neg q \rightarrow \neg p &\Longrightarrow ((p \rightarrow q) \wedge \neg q) \rightarrow \neg p && \textit{explicit the scope of connectives}\\
&\Longrightarrow \neg((\neg p \vee q) \wedge \neg q) \vee \neg p && \textit{rewrite } \rightarrow \textit{ in terms of } \neg \textit{ and } \vee\\
&\Longrightarrow ((p \wedge \neg q) \vee q) \vee \neg p && \textit{move } \neg \textit{ in front of atomic formulas}\\
&\Longrightarrow ((p \vee \neg q) \wedge (\neg q \vee q) \vee \neg p && \textit{distribute } \vee \textit{ over } \wedge\\
&\Longrightarrow ((p \vee \neg q \vee \neg p) \wedge (\neg q \vee q \vee \neg p) && \textit{distribute } \vee \textit{ over } \wedge
\end{aligned}$$

PROPOSITION 4.5. *CNF terminates for every input ϕ.*

PROOF. Notice that each rules (1), (2) and (3) can be applied only a finite number of times. □

After proving termination we also have to show that the transformation in CNF do not change the semantics of the formula, i.e., CNF(ϕ) and ϕ are equivalent.

PROPOSITION 4.6. *ϕ is equivalent to CNF(ϕ).*

PROOF. This is a consequence of the fact that every transformation applied by the procedure for reduction to CNF tranform a formula into an equivalent formula. □

An interpretation $\mathcal{I}$ satisfies a formula in CNF, or equivalently, a set of clauses $\{C_1, \ldots, C_n\}$, if $\mathcal{I}$ satisfies at least on literal $l_{ij} \in C_i$ for every clause C_i. This means that to check satisfiability of a set of clauses we have to find a truth assignment such that for every clauses it select one literal negative and assign 0 or 1 to the corresponding propositional variables depending on the fact that the literal is negative or positive.

EXAMPLE 4.6. *Consider the previous set of clauses*

$$\{\{p, \neg q, r\}, \{q, \neg r\}, \{\neg p, \neg q\}, \{r\}\} \tag{27}$$

An interpretation that satisfies (27) *should satisfy at least one literal per clause. Since the last clause contains only one literal (i.e. r) there is no other choice than satisfying it by setting $\mathcal{I}(r) = True$. This choice would also be a good one for the first clause, since it contains also r which is satisfied by $\mathcal{I}$, therefore we have to take care only of the two remaining clauses:*

$$\{q, \neg r\}, \{\neg p, \neg q\}$$

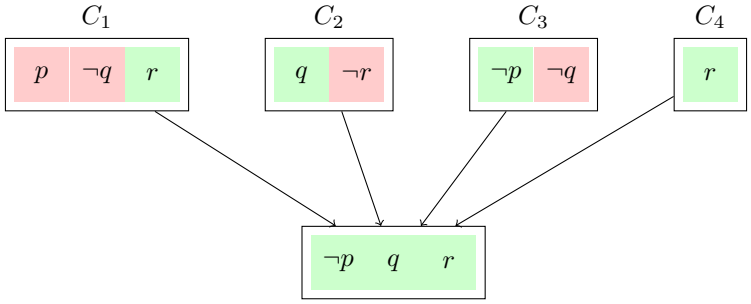
Considering the first of the above two clauses, one can see that the literal $\neg r$ is already falsified by our previous (obliged) choice of $\mathcal{I}(r) = True$. Therefore, to satisfy the first clause, the only choice is to set $\mathcal{I}(q) = True$. We ore now left with one last clause:

$$\{\neg p, \neg q\}$$

Since $\mathcal{I}(q) = True$, the second literal cannot be satisfied; therefore we should satisfy $\neg p$ by setting $\mathcal{I}(p) = False$. In conclusion, we have the interpretation

$$\mathcal{I}(p) = False \qquad \mathcal{I}(q) = True \qquad \mathcal{I}(r) = True$$

A graphical representation of the interpretation that we have found is the following:



literals satisfied by $\mathcal{I}$

3.3. Tseytin Transformation. Checking satisfiability/validity of a formula in CNF is easier. But there is a price in terms of the size of formulas. Indeed due the restriction imposed by CNF, representing some complex proposition in a CNF form can require an exponentially larger formula than by representing it with the full propositional language. As an intuitive example consider the formula $p \leftrightarrow q$ which

in CNF double its size becoming $(\neg p \vee q) \wedge (\neg q \vee p)$. Another example, happens when the formul ais a disjunction of conjunct (instead of a conjuction of disjunct) e.g., the formula $(a \wedge b) \vee (c \wedge d)$ expands into $(a \vee c) \wedge (a \vee d) \wedge (b \vee c) \wedge (b \vee d)$ which is twice the size of the original formula. When these patterns are nested, you can easily immagine that the explosion is of the order of 2^n where n iis the maximum nesting. Let us consider an example.

EXAMPLE 4.7 (Exponential explosion). *The CNF of the formula* $p \leftrightarrow (q \leftrightarrow (r \leftrightarrow s))$ *is composed by the following clauses*

$$\begin{array}{cccc} (\neg r, \neg s, \neg q, p) & (r, s, \neg q, p) & (\neg s, r, q, p) & (\neg r, s, q, p) \\ (\neg r, \neg s, q, \neg p) & (r, s, q, \neg p) & (\neg s, r, \neg q, \neg p) & (\neg r, s, \neg q, \neg p) \end{array}$$

If we add an extra equivalence and compute the CNF of

$$p \leftrightarrow (q \leftrightarrow (r \leftrightarrow (s \leftrightarrow t)))$$

we obtain twice the number of clauses

$$\begin{array}{cccc} (\neg s, \neg t, \neg r, \neg q, p) & (s, t, \neg r, \neg q, p) & (\neg t, s, r, \neg q, p) & (\neg s, t, r, \neg q, p) \\ (\neg s, \neg t, r, q, p) & (s, t, r, q, p) & (\neg t, s, \neg r, q, p) & (\neg s, t, \neg r, q, p) \\ (\neg s, \neg t, \neg r, q, \neg p) & (s, t, \neg r, q, \neg p) & (\neg t, s, r, q, \neg p) & (\neg s, t, r, q, \neg p) \\ (\neg s, \neg t, r, \neg q, \neg p) & (s, t, r, \neg q, \neg p) & (\neg t, s, \neg r, \neg q, \neg p) & (\neg s, t, \neg r, \neg q, \neg p) \end{array}$$

The exponential explosion of the CNF transformation can be contrasted by avoiding to do multiple time the same CNF expansion. To this aim, G. Tseytin in Tseytin 1966 proposed the following procedure:

(1) for a subformula ψ of ϕ which is not a literal, introducing new propositional variables p;
(2) (ii) replace all the occurrences of ψ with the p, and
(3) add a new conjunction stating that p and ϕ are equivlent i.e $\psi \leftrightarrow p$.

In short we apply this transformation:

$$\phi \Rightarrow \phi[\psi/p] \wedge p \leftrightarrow \psi \tag{28}$$

where p is a "fresh" propositional variable, i.e., a propositional variable that do not appear in ϕ, and the notation $\phi[\psi/p]$ indicates the formula ϕ obtained by replacing each occurrences of the subformula ψ with the propositional variable p. This transformation has the advantage that the exponential blow up of the "CNF-ization" of ψ happens only onece instead the number of occurrences of ψ in ϕ.

The transformation (28) however does not preserve the semantics of the formula, i.e., the resulting formula is *not equivalent* to the original formula. For instance the formula $(a \to b) \to (c \to (a \to b))$ is not equivalent to the formula $(p \to (c \to p)) \wedge (p \leftrightarrow (a \to b))$, obtained by replacing the subformula $a \to b$ with p. Indeed there are interpretations of the first formulas which do not satisfy the second. For instance the interpretation with $\mathcal{I}(p) =$ true and $\mathcal{I}(a) =$ True and $\mathcal{I}(b) =$ False satisfies the first formula but not the second. This does not constitute a serious problem, since we are interesting in checking satisfiability. For this task it is sufficient that the original formula is satisfiable if and only if the resulting formula is also satisfiable.

We therefore inroduce a weaker notion than equvalence, which is enough for checking satisfiability.

DEFINITION 4.4 (Equi-satisfiable formulas). *Two formulas ϕ and ϕ' are* equi-satisfiable *if, ϕ is satisfiable if and only if ϕ' is satisfiable*

Equi-satisfiability is weaker than equivalence; indeed it is easy to find pairs of equi-satisfiable formulas that are not equivalent. For instance any pair of atomic formulas p and q are equisatisfiable (indeed they are both satisfiable) but they are not equivalent, Since they are satisfied by different interpretations. Let us now prove that the transformation 28 preserves satisfiability. One can see the difference between equivalent and equi-satisfiability by looking at their formal definition:

$$\begin{array}{cc} \phi \text{ and } \psi \text{ are equivalent} & \forall \mathcal{I}.\ \mathcal{I} \models \phi \Leftrightarrow \mathcal{I} \models \psi \\ \phi \text{ and } \psi \text{ are equi-satisfiable} & \exists \mathcal{I}.\ \mathcal{I} \models \phi \Leftrightarrow \exists \mathcal{J}.\ \mathcal{J} \models \psi \end{array}$$

PROPOSITION 4.7. *For every formula ϕ and avery propositional variable p not occurring in ϕ, ϕ and $\phi[\psi/p] \wedge p \leftrightarrow \psi$ are equisatisfiable.*

PROOF. Suppose that ϕ is satisfiable by an interpretation $\mathcal{I}$ let $\mathcal{I}'$ be the interpretation obtained from $\mathcal{I}$ by setting

$$\mathcal{I}'(p) = \begin{cases} \text{True} & \text{if } \mathcal{I} \models \psi \\ \text{False} & \text{Otherwise} \end{cases}$$

By construction $\mathcal{I}' \models p \leftrightarrow \psi$, To show that $\mathcal{I}' \models \phi[\psi/p]$ we have to proceed by induction.

- **Base case:** ϕ is equal to ψ. Then $\mathcal{I} \models \phi$ implies that $\mathcal{I}' \models p$ and since $\phi[\psi/p]$ is equal to p, we have that $\mathcal{I}' \models \phi[\psi/p]$. If instead ψ is not a subformula of ϕ then $\phi[\psi/p] = \phi$ and since ϕ does not contain p, it has the same truth value w.r.t., $\mathcal{I}'$ and $\mathcal{I}$, and therefore $\mathcal{I}' \models \phi[\psi/p]$.
- **Step case:** Suppose that ϕ is $\phi_1 \vee \phi_2$, We have that $\mathcal{I} \models \phi_i$ for some $i = 1, 2$. By the induction hypothesis we have that $\mathcal{I}' \models \phi_i[\psi/p]$ Since $\phi[\psi/p]$ is equal to $\phi_1[\psi/p] \vee \phi_2[\psi/p]$, we can conclude that $\mathcal{I}' \models \phi[\psi/p]$.
- ... Do the other cases by exercize.

□

The Tseytin transformation introduced in Tseytin 1966 applies the (28) rule in a sistematic way, This correspond to the following procedure that can be applied to any formula ϕ (not necessariliy in CNF).

(1) let $\psi_1, \ldots, \psi_n$ all the subformulas of ϕ, including ϕ itself and excluding those that are literals[2]: make sure that if ψ_i is a subformula of ψ_j then $j \geq i$;
(2) let $x_1, \ldots, x_n$ a set of propositional variables not occurring in ϕ.
(3) for $i = 1, \ldots, n$ apply the following transformation:

$$\phi \Rightarrow \phi[\psi_i/x_i] \wedge (x_i \leftrightarrow \psi_i)$$

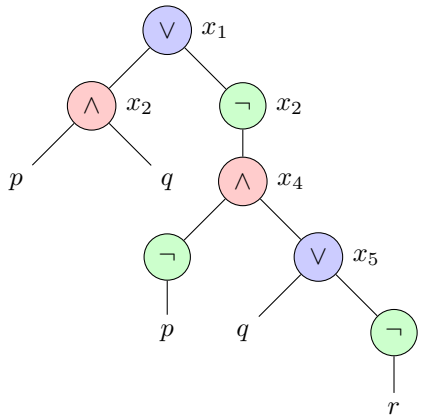
(4) apply CNF to the resulting formula.

EXAMPLE 4.8. *Let us consider the formula*

$$(p \wedge q) \vee \neg(\neg p \wedge (q \vee \neg r))$$

With the following parse tree and subformulas

[2]In the original formulation also negative literal were included.



The Tseytin's transformation introduces 5 additional variable $x_1, \ldots, x_5$ since the formula has 5 subformuls which are not literals. In the following table, we provide the definition of each x_i in terms of it's direct subformulas, and the corresponding transformation in CNF.

$$
\begin{aligned}
x_1 &\leftrightarrow x_2 \vee x_3 && \leftrightarrow \{\neg x_1, x_2, x_3\}, \{\neg x_2, x_1\}, \{\neg x_3, x_1\} \\
x_2 &\leftrightarrow p \wedge q && \leftrightarrow \{\neg x_2, p\}, \{\neg x_2, q\}, \{\neg p, \neg q, x_2\} \\
x_3 &\leftrightarrow \neg x_4 && \leftrightarrow \{\neg x_3, \neg x_4\}, \{x_3, x_4\} \\
x_4 &\leftrightarrow x_5 \wedge \neg p && \leftrightarrow \{\neg x_4, x_5\}, \{\neg x_5, x_4\} \\
x_5 &\leftrightarrow q \vee \neg r && \leftrightarrow \{\neg x_5, q, \neg r\}, \{\neg q, x_5\}, \{r, x_5\}
\end{aligned}
$$

To check if the initial formula is satisfiable it is sufficent to check if x_1 (which is the auxiliary variable that encodes the formula itself) and all the CNF provided above are satisfiable.

Tseytin's tranformation transform a formula ϕ into a set of clauses that contains at most three literals as all the formula that are given in input to CNF are of the form $x_i \leftrightarrow (l_1 \circ l_2)$ or $x_i \leftrightarrow \neg x_j$, where $\circ$ is some binary connective in $\{\wedge, \vee, \rightarrow, \leftrightarrow\}$. The connective that generates the largest number of clauses is $\leftrightarrow$, and they are 4. Indeed, CNF$(a \leftrightarrow (b \leftrightarrow c))$ contains four clauses, namely: $\{a, b, c\}$, $\{a, \neg b, \neg c\}$, $\{\neg a, b, \neg c\}$ and $\{\neg a, \neg b, c\}$. This means that if a formula ϕ contains n connectives its Tseytin tranformed contains at most $4 \times n$ clauses each of which contains at most 3 literals. Therefore the size of the CNF obtained with the Tseytin tranformation is linear w.r.t., the number of connectives of the original formula. This is clearly a good news for the deciding satisfiability but this reduction comes with the price of introducing n new propositional variable, for which the satisfiability d decision procedure has to produce a truth assignment.

4. Davis-Putnam-Logemann-Loveland Procedure (DPLL)

The Davis-Putnam-Logemann-Loveland (DPLL) algorithm M. Davis and Putnam 1960; M. Davis, Logemann, and Loveland 1962 is a procedure that combines search and deduction to decide satisfiability of CNF formulas. This algorithm underlies most modern SAT solvers. While the basic procedure itself has been developed in the 60's, practical DPLL-based SAT solvers only started to appear from the mid 1990s as a result of enhancements such as clause learning, non-chronological

backtracking, branching heuristics, restart strategies, and lazy data structures. The DPLL algorithm is based around backtrack search for a satisfying valuation.

Before introducing the DPLL algorithm, we need to provide some notation and a key subruting exploited by DPLL, called Unit Propagation that deals with the unit clauses. A *partial interpretation* is a truth assignment for a subset of the propositional variables of a formula. As we have seen in the section of model checking, partial assignments in some cases can fully determine the truth value of some formulas. For instance if p is assigned to True by a partial interpretation then any formula of the form $p \vee \phi$ will be evaluated to true, independently from the truth value of ϕ. A partial evaluation can be seen as a set of literals. We will use the same notation for interpretations and partial interpretations. $p \in \mathcal{I}$ means that $\mathcal{I}(p) = \text{True}$ $\neg p \in \mathcal{I}$ means that $\mathcal{I}(p) = \text{False}$. We also use the notation $\bar{l}$ to denote the opposite of the literal l. I.e., $\bar{p}$ is $\neg p$ and $\overline{\neg p}$ is p. Similar definition can be defined also in sets of literals C. $\bar{C} = \{\bar{l} \mid l \in C\}$.

Given a partial assignment $\mathcal{I}$, we can attempt to evaluate a clause C.

- A clause C is true under $\mathcal{I}$ if one of its literals is true, i.e if $\mathcal{I} \cap C \neq \emptyset$
- A clause C is false under $\mathcal{I}$ if all its literal are set to false by $\mathcal{I}$, i.e. $C \subseteq \bar{\mathcal{I}}$.
- therwise C is undefined (or "unresolved") in $\mathcal{I}$.

When a clause is unresolved in an assignment $\mathcal{I}$ we can simplify it by evaluating the literals that are assigned by $\mathcal{I}$, For a clause C and a literal l the semplification of C w.r.t., l, denoted by $C|_l$ denotes the clause obtained by evaluating C w.r.t., the partial evaluation that assigns the literal l to true (i.e. $l \in \mathcal{I}$). $C|_l$ is obtained from C by the following two operations:

- removing all the clauses that contains l;
- remove $\bar{l}$ (if present) from C,

For any CNF formula ϕ, $\phi|_l = \{C|_l \mid C \in \phi\}$. For a set of literals $\{l_1, \ldots, l_n\}$, $C_{l_1,\ldots,l_n}$ is equal to $(\ldots (C|_{l_1})\,|_{l_2} \ldots)\,|_{l_n}$ Notice that the order and the repetitions does not affect the result , and therefore we can use the notation $\phi|_{\mathcal{I}}$ where $\mathcal{I}$ is a set of literal for denotign $\phi|_{l_1,\ldots,l_n}$, where $\mathcal{I} = \{l_1, \ldots, l_n\}$.

The subruting that is used by DPLL is called unit propagation and is applied when a CNF formula contains some unit clause. If ϕ contains unit clause $\{l\}$ then, to satisfy ϕ we have to satisfy $\{l\}$ and therefore the literal l must be evaluated to True. As a consequence ϕ can be simplified using the procedure called UnitPropagation. The procedure is shown in algorithm **??**.

Algorithm 2 UnitPropagation(ϕ : CNF, $\mathcal{I}$: Partial assignment)

```
1: while φ contains a unit clause {l} do
2:     I, φ ← I ∪ {l}, φ|_l
3:     if {} ∈ φ then
4:         return I, φ
5:     end if
6: end while
7: return I, φ
```

The basic DPLL algorithm is shown in Figure 1. DPLL algorithm incrementally build a a partial interpretation that satisfies the input CNF-formula by depth-first search. At any time the state of the algorithm is a pair $(\mathcal{I}, \phi)$, where $\mathcal{I}$ is a partial

truth assignment and ϕ is the set of clauses that are undecided by $\mathcal{I}$. This means that $\mathcal{I}$ must be extended in order to satisfy all the remaining clauses which are in ϕ.

Algorithm 3 DPLL(ϕ : CNF, $\mathcal{I}$: Partial assignment)

```
1: I, φ ← UnitPropagation(I, φ)
2: if {} ∈ φ then
3:     return Unsatisfiable
4: end if
5: if φ = {} then
6:     return I
7: else
8:     select a l from some clause C ∈ φ
9:     I = DPLL(φ|_l, I ∪ {l})
10:    if I ≠ Unsatisfiable then
11:        return I
12:    else
13:        I ← DPLL(φ|_l̄, I ∪ {l̄})
14:        return I
15:    end if
16: end if
```

Example 4.9. *Let us consider a concrete example and look at how DPLL behaves on the following CNF ϕ.*

$$\{A, B\} \tag{29}$$
$$\{B, C\} \tag{30}$$
$$\{\neg A, \neg X, Y\} \tag{31}$$
$$\{\neg A, X, Z\} \tag{32}$$
$$\{\neg A, \neg Y, Z\} \tag{33}$$
$$\{\neg A, X, \neg Z\} \tag{34}$$
$$\{\neg A, \neg Y, \neg Z\} \tag{35}$$

Figure 2 shows the execution trace of DPLL algorithm on the set of clauses (29)–(35)*. The input set of clauses do not contain a unit clauses therefore unit propagation is not applied, furthermore $\mathcal{I}$ is not Unsatisfiable and and the set of clauses ϕ are not empty. Therefore DPLL goes to line 3 and select the literal A. Then it recursively apply DPLL to $\phi|_A$ and $\mathcal{I} = \{A\}$. The recursive calls proceeds similarly by selecting first the literal B and then X; At this point DPLL is called with input $\phi|_{A,B,X}$ and $\mathcal{I} = \{A, B, x\}$. Since $\phi_{A,B,X}$ contains a unit clause $\{Y\}$ then Y then the unit propagation is called. This procedure executes two steps by first propagating Y and then propagating Y, returning the set of clauses that containt the empty clause. At this point DPLL returns* Unsatisfiable *and backtrack (shown in dashed lines) at the most recent choice, which is on the literal X. Therefore, it analize the case of the negative literal $\neg X$, calling DPLL on $\phi_{A,B,\neg X}$ and $\mathcal{I} = \{A, B, \neg X\}$. But also in this case DPLL after calling unit propagation returns* Unsatisfiable*, Therfore, DPLL backtrack to the upper literal that has been choosen which is B*

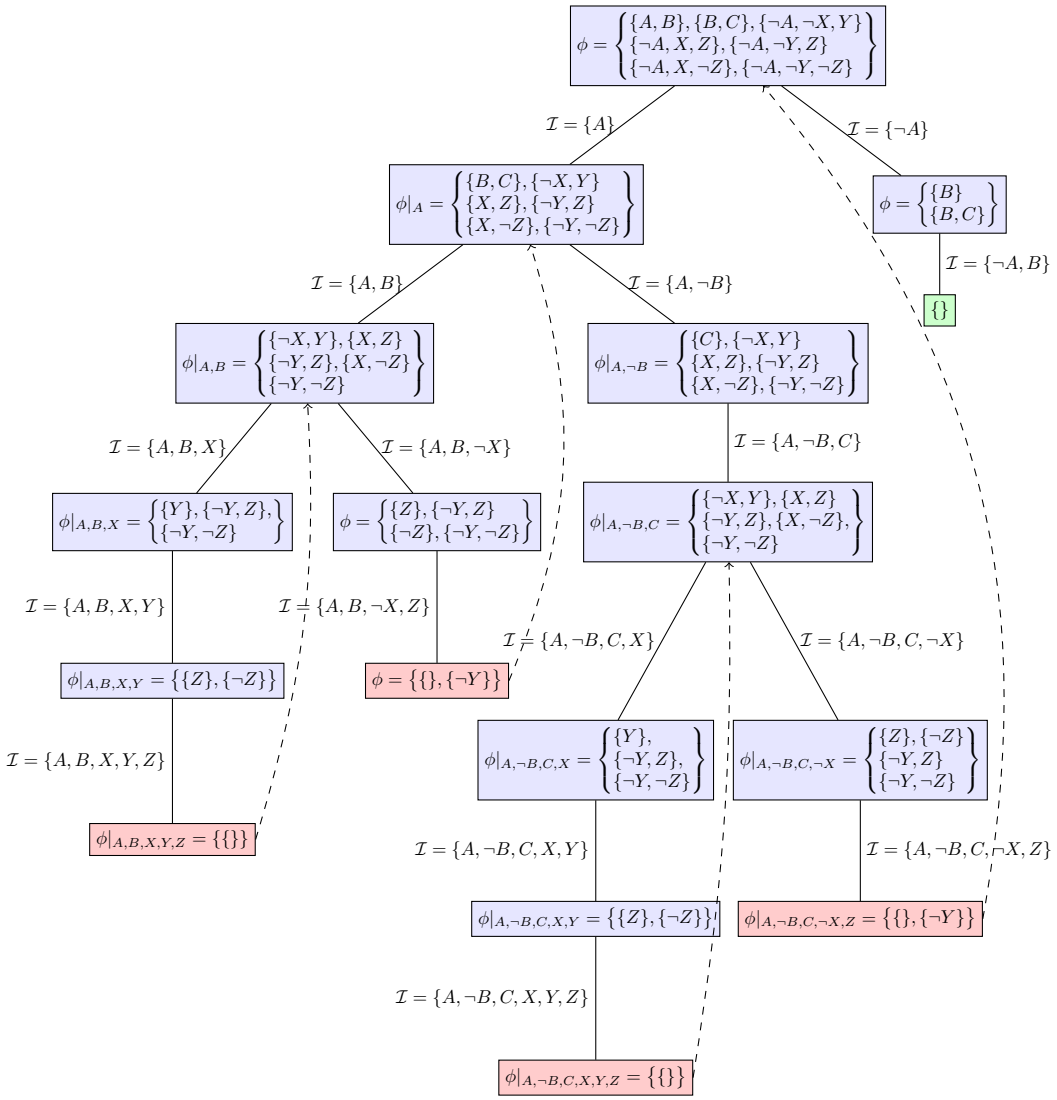


FIGURE 2. The execution trace of DPLL algorithm on the set of clauses (29)–(35).

.

and explores the case with $\neg B$. As one can see from the trace also in this case all the branches leads to a set of unsatisfiable clauses (empty clause). Therfore DPLL backtrack to the initial choiche, which was on the literal A, and attemst with the assignment $\mathcal{I} = \{\neg A\}$. This leads after a unit propoatation to the emtpy set of clauses, and tDPLL returns the partial assignment $\{\neg A, B\}$, and therefore the initial set of clauses are satisfiable, by all the assignments that agree with the partial assignment returned by DPLL.

5. Sat Solvers

Most of of the satate-of-the-art implementation of sat solvers are available through Python interface. In this section we summarize the main features offered by `PySAT`. Before introducing the library, we introduce a format for representation of CNF, called `DIMACS`.

5.1. `DIMACS` CNF. The `DIMACS` CNF format is a textual representation of a formula in conjunctive normal form. A formula in conjunctive normal form is a conjunction (logical and) of a set of clauses. Each clause is a disjunction (logical or) of a set of literals. A literal is a variable or a negation of a variable. `DIMACS` CNF uses positive integers to represent variables and their negation to represent the corresponding negated variable. This convention is also used for all textual input and output in Varisat.

There are several variations and extensions of the `DIMACS` CNF format. Varisat tries to accept any variation commonly found. Currently no extensions are supported.

`DIMACS` CNF is a textual format. Any line that begins with the character c is considered a comment. Some other parsers require comments to start with c and/or support comments only at the beginning of a file. Varisat supports them anywhere in the file.

A `DIMACS` file begins with a header line of the form p cnf ¡variables¿ ¡clauses¿. Where ¡variables¿ and ¡clauses¿ are replaced with decimal numbers indicating the number of variables and clauses in the formula.

Varisat does not require a header line. If it is missing, it will infer the number of clauses and variables. If a header line is present, though, the formula must have the exact number of clauses and may not use variables represented by a number larger than indicated.

Following the header line are the clauses of the formula. The clauses are encoded as a sequence of decimal numbers separated by spaces and newlines. For each clause the contained literals are listed followed by a 0. Usually each clause is listed on a separate line, using spaces between each of the literals and the final zero. Sometimes long clauses use multiple lines. Varisat will accept any combination of spaces and newlines as separators, including multiple clauses on the same line.

EXAMPLE 4.10. *As an example the formula* $(x \vee y \vee \neg z) \wedge (\neg y \vee z)$ *could be encoded as this:*

```
p cnf 3 2
1 2 -3 0
-2 3 0
```

The first line "`p cnf 3 2`" states that this file specifies a `p`roblem in `cnf` with `3` propositional variables and `2` clauses. The other two lines specify the clauses (every clause ends with a `0`).

5.2. Problem solving with Sat Solvers. In this section we describe how to compute a solution of a problem that you have specified in terms of a set of propositional formula Γ that uses the set of propositional variables $\mathcal{P}$. Usually $\mathcal{P}$ contains propositional variable with some "human readable" name. For instance we can use the the propositional variable happy(John) to remind us that it represents the proposition that John is happy.

5.2.1. *Encoding and Decoding propositinal variables.* The first operation that you have to perfomr is to define an function (encoding) that maps each $p \in \mathcal{P}$ into a natural number. i.e, the function

$$\texttt{Encode} : \mathcal{P} \longrightarrow [n]$$

where $[n] = \{1, 2, \ldots, n\}$ and n is the number of peropositional variables which are present in $\mathcal{P}$. The function `Encode` cannot associate the same integer to two different propositional variables, i.e., `Encode` is injective. This property allows us to define the `Decode` function from $[n]$ to $\mathcal{P}$ as the inverse of `Encode`.

$$\texttt{Decode} : [n] \longrightarrow \mathcal{P}$$

is such that $\texttt{Decode}(i) = \texttt{Encode}^{-1}(i)$.

5.2.2. *Generating the* `DIMACS` *code.* To generate the `DIMACS` code we have first to generate the first line, which is

$$\texttt{p CNF } |\mathcal{P}| \ |\Gamma|$$

that means that this is a cnf problem on $|\mathcal{P}|$ proposition and Γ clauses. Then for every clause $\gamma = \{l_1, \ldots, l_k\} \in \Gamma$ we cenerate the line $\texttt{Encode}(p)$ if l_i is a positive literal p and $-\texttt{Encode}(p)$ if l_i is a negative literal $\neg p$.

5.2.3. *Decoding solution.* We then call the sat solver on the encoding that we have generated. The sat solver can return either `UNSATISFIABLE` or `SATISFIABLE`. In the latter case we can ask the solver to return a mnodel. The sat solver return a model codified in a sequence of positive and negative integers.

$$r = [\pm 1, \pm 2, \ldots, \pm n]$$

where n is the total number of propositional variables. The list encodes the following interpretation: for every $p \in \mathcal{P}$

$$\mathcal{I}_r(p) = \begin{cases} \mathit{True} & \text{if } \texttt{Encode}(p) \in r \\ \mathit{False} & \text{if } -\texttt{Encode}(p) \in r \end{cases}$$

5.2.4. *Additional solutions.* If we want to obtain additional models we have to add to the initial set of clauses that fact that we want a model different from $\mathcal{I}_r$. A model of Γ is different from $\mathcal{I}_r$ if at least one proposition takes a truth value different from the one assigned by $\mathcal{I}_r$. We can therefore add the clause

$$\bigvee_{\mathcal{I}_r(p)=\mathit{True}} \neg p \vee \bigvee_{\mathcal{I}_r(p)=\mathit{False}} p$$

which can be codified in the sequence $-r = [-i \mid i \in r]$.

6. PySAT

see website `https://pysathq.github.io/`

7. Examples of problems solved in SAT

7.1. Numbermind in `PySAT`. The game Number Mind is a variant of the well known game Master Mind. Instead of colored pegs, you have to guess a secret sequence of digits. After each guess you're only told in how many places you've guessed the correct digit. So, if the sequence was 1234 and you guessed 2036, you'd be told that you have one correct digit; however, you would NOT be told that

you also have another digit in the wrong place. E.g. the feedbacks with the secret sequence `39542` are:

Guess	;	**feedback**
903	;	2 correct
70794	;	0 correct
39458	;	2 correct
34109	;	1 correct
51545	;	2 correct
12531	;	1 correct

How can we use `PySAT` to generate the guesses that takes into account the feedbacks?

The first step is to define the set of propositinal variables and extablish what they encode. The only relevant proposition in the NumberMind world is the fact that a certain number is in a certain position in a sequence. Therefore, for every position $p \in \{1, \dots, p_{\max}$ and every number $n \in \{0, \dots, 9\}$ we have the propositional variable

$$\mathsf{In}(n, p) \qquad \text{there is an } n \text{ in position } p$$

The second step is to provide encoding and decoding function for the `DIMACS` format. One simple encoding/decoding function can be the following.

$$\begin{aligned} \texttt{Encode}(\mathsf{In}(n,p)) &= p \cdot 10 + n + 1 \\ \texttt{Decode}(k) = \mathsf{In}(n,p) \text{ where } \quad & n = (k-1) \mod 10 \\ & p = (k-1) \div 10 \end{aligned}$$

In the next step we have to formalize the constraints of the game. In NumberMind constrains comes from two sources. The first is the fact that in the sequence only one digic prosition is admitted. The second soruce of constraints is the feedback provided when the player proposes a guess. These constraints are added every time the player proposes a guess. These constraints are both cardinality constraints:

- at every position there is exaclty one digit

$$\bigvee_{n=0}^{9} \mathsf{In}(n,p) \wedge \bigwedge_{0 \le n < n' \le 9} \neg(\mathsf{In}(n,p) \wedge \mathsf{In}(n',p)) \tag{36}$$

- there are k correct digits in given a guess $G = \{\mathsf{In}(n_1, 1), \mathsf{In}(n_2, 2), \dots, \mathsf{In}(n_{p_{\max}}, p_{\max})\}$

$$\bigwedge_{\substack{I \subseteq G \\ |I| = p_{\max} - k + 1}} \bigvee_{\mathsf{In}(n,p) \in I} \mathsf{In}(n,p) \wedge \bigwedge_{\substack{I \subseteq G \\ |I| = k+1}} \bigvee_{\mathsf{In}(n,p) \in I} \neg \mathsf{In}(n,p) \tag{37}$$

We are now ready to define the main cycle of the NumberMind game, which is reported in Algorithm 4

7.2. MineSweeper. ...

Algorithm 4 Numbermind main cycle

$s \leftarrow$ random secret sequence
$S \leftarrow \{\mathsf{In}(n,p) \mid$ the p-th element of s is $n\}$
$\Gamma \leftarrow$ (36) for every position p with $1 \leq p \leq p_{\max}$
while $k < n$ **do**
    $G \leftarrow SAT(\Gamma)$
    $k_G \leftarrow |\{\mathsf{In}(n,p) \in S \mid G \models \mathsf{In}(n,p)\}|$
    add (37) to Γ for $k = k_G$
end while
return $\{\mathsf{In}(n,p) \in S \mid G \models \mathsf{In}(n,p)\}$

8. Exercises

Exercise 60:
Explain why $\phi \models \psi$ holds if and only if $\{\phi, \neg\psi\}$ is not satisfiable.

Exercise 61:
How many conjunctive normal forms formulas can you build with n propositional variables? Remember that a conjunctive normal form can be seen as a set of set of literals.

Solution A CNF formula is a set of clauses. A clause is a set of literals. With n propositional variables we can build $2n$ literals, i.e., the positive literals, which coincide with the propositional variables, and the negative literals, which are the negation of propositional variables. A clause is any set of literals, Therefore we can build 2^{2n} clauses. A CNF formula is a set of clauses, therefore we can build $2^{2^{2n}}$ CNF formulas. □

Exercise 62:
Transform the following formula in CNF

$$((\neg((p \to q) \wedge (p \vee q \to r)) \to (p \to r)))$$

Exercise 63:
Explain the difference between the following two facts: "ϕ and ψ are equivalent"and "ϕ and ψ are equisatisfiable"

Solution ϕ is equivalent to ψ if and only if for all the interpretations $\mathcal{I}$, $\mathcal{I} \models \phi$ if and onlyif $\mathcal{I} \models \psi$.

ϕ and ψ are equisatisfiable if ϕ is satisfiable if and only if ψ is satisfiable. □

Exercise 64:
Consider the following two formulas:

$$\phi = P \to ((Q \to R) \wedge (Q \vee R))$$
$$\psi = (\neg P \to Q) \to R$$

Does ϕ logically follow from ψ? or viceversa? Prove it via truth tables.

Solution Let us compute the truth tables for both formulas:

	P	Q	R	P	→	((	Q	→	R	)	∧	(	Q	∨	R	))	(	¬	P	→	Q	)	→	R
1	T	T	T	T	T		T	T	T		T		T	T	T			F	T	T	T	T	T	
2	T	T	F	T	F		T	F	F		F		T	T	F			F	T	T	T	F	F	
3	T	F	T	T	T		F	T	T		T		F	T	T			F	T	T	F	T	T	
4	T	F	F	T	F		F	T	F		F		F	F	F			F	T	T	F	F	F	
5	F	T	T	F	T		T	T	T		T		T	T	T			T	F	T	T	T	T	
6	F	T	F	F	T		T	F	F		F		T	T	F			T	F	T	T	F	F	
7	F	F	T	F	T		F	T	T		T		F	T	T			T	F	F	F	T	T	
8	F	F	F	F	T		F	T	F		F		F	F	F			T	F	F	F	T	F	

Notice that in all the cases in which ϕ is true ϕ is also true, this implies that ϕ logically follows from ψ. On the other hand, since there is an assignment for which ϕ is true but ψ is false (assignment number 6) ψ is not a logical consequence of ϕ. □

Exercise 65:
Convert the following formula

$$p \vee q \rightarrow p \wedge \neg r$$

into an equisatisfiable CNF formula using the Tseitin's Transformation.

Exercise 66:
Check the following facts via DPLL (in the exam there could be one of this).

(1) $\models (p \rightarrow q) \wedge \neg q \rightarrow \neg p$
(2) $\models (p \rightarrow q) \rightarrow (p \rightarrow \neg q)$
(3) $\models (p \vee q \rightarrow r) \vee p \vee q$
(4) $\models (p \vee q) \wedge (p \rightarrow r \wedge q) \wedge (q \rightarrow \neg r \wedge p)$
(5) $\models (p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r))$
(6) $\models (p \vee q) \wedge (\neg q \wedge \neg p)$
(7) $\models (\neg p \rightarrow q) \vee ((p \wedge \neg r) \equiv q)$
(8) $\models (p \equiv q) \vee (p \equiv \neg q)$
(9) $\models (p \rightarrow (q \vee r)) \vee (r \rightarrow \neg p)$
(10) $\models \neg(p \equiv q) \vee \neg(p \equiv \neg q)$
(11) $\models (p \equiv q) \vee (p \equiv \neg q)$

SolutionThe solution of some cases are the following:

(1) The expression $\models (p \rightarrow q) \wedge \neg q \rightarrow \neg p$ means that $(p \rightarrow q) \wedge \neg q \rightarrow \neg p$ is valid. To check the validity of a formula ϕ with DPLL we have to check (un)satisfiability of the negated of ϕ i.e. $\neg\phi$. Since DPLL requires CNF we first have to transform in CNF $\neg\phi$.

$$\begin{aligned}\mathrm{CNF}(\neg((p \rightarrow q) \wedge \neg q \rightarrow \neg p)) &= \mathrm{CNF}((p \rightarrow q) \wedge \neg q) \wedge \mathrm{CNF}(\neg\neg p)) \\ &= \mathrm{CNF}((p \rightarrow q)) \wedge \mathrm{CNF}(\neg q) \wedge \mathrm{CNF}(p)) \\ &= \{\neg p, q\}, \{\neg q\}, \{p\}\end{aligned}$$

Now we apply DPLL to $\{\neg p, q\}, \{\neg q\}, \{p\}$.
(a) since $\{\neg p, q\}, \{\neg q\}, \{p\}$ contains unit clauses, we apply UNITPROPAGATION.

$\{\{\neg p, q\}, \{\neg q\}, \{p\}\}$	$\mathcal{I} = \{\}$	Unit propagation on $\{p\}$
$\{\{\neg p, q\}, \{\neg q\}\}\|_p$	$\mathcal{I} = \{p\}$	
$\{\{q\}\{\neg q\}\}$	$\mathcal{I} = \{p\}$	Unit propagation on $\{q\}$
$\{\{\neg q\}\}\|_q$	$\mathcal{I} = \{p, q\}$	
$\{\{\}\}$	$\mathcal{I} = \{p, q\}$	We have obtained the empty clause, and we return UNSAT

Since we have obtained the empty clauses we return UNSAT. Since $\neg((p \rightarrow q) \wedge \neg q \rightarrow \neg p)$ is not satisfiable, it must be that $(p \rightarrow q) \wedge \neg q \rightarrow \neg p$ is valid.

(2) We first compute the CNF of the negation of the formula.

$$\begin{aligned}\mathrm{CNF}(\neg((p \to q) \to (p \to \neg q))) &= \mathrm{CNF}(p \to q) \wedge \mathrm{CNF}(\neg(p \to \neg q)) \\ &= \mathrm{CNF}(p \to q) \wedge \mathrm{CNF}(p) \wedge \mathrm{CNF}(\neg\neg q) \\ &= \{\neg p, q\}, \{p\}, \{q\}\end{aligned}$$

We can now apply DPLL

$\{\{\neg p, q\}, \{p\}, \{q\}\}$	$\mathcal{I} = \{\}$	Unit propagation on $\{q\}$
$\{\{\neg p, q\}\}\|_q, \{p\}\|_q\{q\}\|_q$	$\mathcal{I} = \{q\}$	
$\{\{p\}\}$	$\mathcal{I} = \{q\}$	Unit propagation on $\{p\}$
$\{\{p\}\}\|_p$	$\mathcal{I} = \{q, p\}$	
$\{\}$	$\mathcal{I} = \{q, p\}$	

Since we obtaine the empty set of clauses (all the clauses are eliminated), then we have that the set of input clauses are satisfiable, I.e., that $\neg((p \to q) \to (p \to \neg q))$ is satisfiable, and therefore $(p \to q) \to (p \to \neg q)$ is not valid. A counter-model, i.e. a model that falsifies it, is returned by the DPLL and it is equal to $\mathcal{I}(p) = True$ and $\mathcal{I}(q) = True$.

6. Let's compute the CNF of $\neg((p \vee q) \wedge (\neg q \wedge \neg p))$

$$\begin{aligned}\mathrm{CNF}(\neg((p \vee q) \wedge (\neg q \wedge \neg p))) &= \mathrm{CNF}(\neg(p \vee q)) \otimes \mathrm{CNF}(\neg(\neg q \wedge \neg p)) \\ &= (\mathrm{CNF}(\neg p) \wedge \mathrm{CNF}(\neg q)) \otimes \mathrm{CNF}(q \vee p) \\ &= (\neg p \wedge \neg q) \otimes (q \vee p) \\ &= (\neg p \vee q \vee p) \wedge (\neg q \vee q \vee p) \\ &= \{\neg p, q, p\}, \{\neg q, q, p\}\end{aligned}$$

We can now apply DPLL: Since there is no unit clauses, we skip that block and select a literal

$\{\{\neg p, q, p\}, \{\neg q, q, p\}\}$	$\mathcal{I} = \{\}$	select the literal p
$\{\{\neg p, q, p\}, \{\neg q, q, p\}\}\|_p$	$\mathcal{I} = \{p\}$	
$\{\}$	$\mathcal{I} = \{p\}$	We have eliminated all the clauses therefore we return SAT

Since the negated of the formula is satisfiable, then the initial formula cannot be valid.

10. We compute the CNF of the negated formula

$$\begin{aligned}\mathrm{CNF}(\neg(\neg(p \equiv q) \vee \neg(p \equiv \neg q))) &= \mathrm{CNF}((p \equiv q) \wedge (p \equiv \neg q)) \\ &= \mathrm{CNF}((p \equiv q)) \wedge \mathrm{CNF}(p \equiv \neg q) \\ &= (\mathrm{CNF}(p) \otimes \mathrm{CNF}(\neg q)) \wedge (\mathrm{CNF}(\neg p) \otimes \mathrm{CNF}(q)) \wedge \\ &\quad (\mathrm{CNF}(p) \otimes \mathrm{CNF}(\neg\neg q)) \wedge (\mathrm{CNF}(\neg p) \otimes \mathrm{CNF}(\neg q)) \\ &= \{p, \neg q\}, \{\neg p, q\}, \{p, q\}, \{\neg p, \neg q\}\end{aligned}$$

Then we apply the DPLL algorithm

$\{\{p, \neg q\}, \{\neg p, q\}, \{p, q\}, \{\neg p, \neg q\}\}$	$\mathcal{I} = \{\}$	No unit clause. Choose literal p (*)
$\{\{p, \neg q\}, \{\neg p, q\}, \{p, q\}, \{\neg p, \neg q\}\}\|_p$	$\mathcal{I} = \{p\}$	
$\{\{q\}, \{\neg q\}\}$	$\mathcal{I} = \{p\}$	Apply unit propagation for clause $\{q\}$
$\{\{q\}, \{\neg q\}\}\|_q$	$\mathcal{I} = \{p, q\}$	
$\{\{\}\}$	$\mathcal{I} = \{p, q\}$	We have derived the empty clause. So we have backtrack to the last choosen literal (*) and choose the opposite one
$\{\{p, \neg q\}, \{\neg p, q\}, \{p, q\}, \{\neg p, \neg q\}\}$	$\mathcal{I} = \{\}$	Choose literal $\neg p$
$\{\{p, \neg q\}, \{\neg p, q\}, \{p, q\}, \{\neg p, \neg q\}\}\|_{\neg p}$	$\mathcal{I} = \{p\}$	
$\{\{\neg q\}, \{q\}\}$	$\mathcal{I} = \{\neg p\}$	
$\{\{q\}, \{\neg q\}\}$	$\mathcal{I} = \{\neg p\}$	Apply unit propagation for clause $\{q\}$
$\{\{q\}, \{\neg q\}\}\|_q$	$\mathcal{I} = \{\neg p, q\}$	
$\{\{\}\}$	$\mathcal{I} = \{\neg p, q\}$	We have derived the empty clause. Since no backtrack are possible, we stop and return UNSAT

Since the negation of the formula is unsatisfiable, then the formula must be valid.

□

Exercise 67:
Check the following facts via DPLL

$$\models ((\neg A \vee B) \rightarrow C) \vee ((\neg B \rightarrow A) \rightarrow C)$$

Solution To check that a formula ϕ is valid (i.e., $\models \phi$) with DPLL we have to tranform $\neg\phi$ in CNF and then try to derive the emplty clause. So let's start with transforming

$$\neg(((\neg A \vee B) \rightarrow C) \vee ((\neg B \rightarrow A) \rightarrow C))$$

in CNF obtaining the following clauses

$$\{\neg A, B\}, \{A, B\}, \{\neg C\}$$

By unit propagation we have that $\mathcal{I}(C) = False$, and we have to compute $\{A, B\}|_{\neg C}, \{A, \neg B\}|_{\neg C}$ which remain unchanged. We can therefore choose an assignment $\mathcal{I}(B) = True$ and we obtaine the empty set of clauses. WHich implies that any assignment with $\mathcal{I}$ with $\mathcal{I}(C) = False$ and $\mathcal{I}(B) = True$ falsifies the intitial formula, and therefore the intitial formula is not valid. □

Exercise 68:
Use DPLL to find an assignment that satisfies the clauses of exercise 1.

Solution Let ϕ be the set of clauses of exercise 1.

(1) no unit propagation (there are not unit clauses in ϕ)
(2) select the literal K of the first clasue and assign $\mathcal{I}(K) = True$

(3) compute $\phi|_K$

$$R \vee V$$
$$\neg R \vee \neg V$$
$$\neg A \vee R$$
$$V$$
$$\neg H \vee A$$

(4) apply unit propagation for the clause V setting $\mathcal{I}(V) = True$:

$$\neg R$$
$$\neg A \vee R$$
$$\neg H \vee A$$

(5) apply unit propagation for the clause $\neg R$ setting $\mathcal{I}(R) = False$:

$$\neg A$$
$$\neg H \vee A$$

(6) apply unit propagation for the clause $\neg A$ setting $\mathcal{I}(A) = False$:

$$\neg H$$

(7) apply unit propagation for the clause $\neg H$ setting $\mathcal{I}(H) = False$:
(8) we obtain the empty set of clauses, this implies DPLL exits with SAT and returns the assignment: $\mathcal{I}(K) = True$, $\mathcal{I}(V) = True$, $\mathcal{I}(R) = False$, $\mathcal{I}(A) = False$, $\mathcal{I}(H) = False$

that corresponds to the situation where K and V are the only one who are chatting. Notice that DPLL tells us that this is a situation compatible with the constraints, but does not guarantee that this is the only one. There could be others. To check if there are other assighment try to start by assigning false to K and continue with the DPLL algorithm. □

Exercise 69:

There are N towns each one having a local radio station. We have to assing a radio frequency out of Q available ones to each radio station. To avoid interferences, towns closer than $20Km$ can not use the same frequency. We have a function $Distance(i, j)$ indicating the distance between town i and j. Is it possible to assign the frequencies? Express the problem in CNF.

Solution We need the set of propositions $\{freq(q, t) \mid t \in T,\ q \in Q\}$, where $freq(q, t)$ means that the frequency q is assigned to the town t.

- Every town need one frequency

$$\bigwedge_{t \in T} \bigvee_{q \in Q} freq(q, t)$$

- the same frequency cannot be assigned to closed towns

$$\bigwedge_{\substack{q, q' \in Q \\ q \neq q'}} \bigwedge_{\substack{t, t' \in T \\ 0 < Distance(t, t') \leq 20Km}} \neg freq(q, t) \vee \neg freq(q, t')$$

- only one frequency is assigned for every town

$$\bigwedge_{t \in T} \bigwedge_{\substack{q, q' \in Q \\ q \neq q'}} \neg freq(q, t) \vee \neg freq(q', t)$$

□

Exercise 70:

Suppose that ϕ has k subformulas $\phi_1, \ldots, \phi_k$. How many new propositional variables are introduced by the Tseitin's transformation? **Solution** The Tseitin's transformation introduces one propositional variable for each non atomic subformula. Therefore, if ϕ contains m propositional variables, the Tseitin's trasformation introduces $n - m$ new propositional variables. □

Exercise 71:

Check if the following formula is valid using DPLL. If it is not valid provide a counter-model, i.e., an assignment that falsify it.

$$(p \wedge q) \vee r \equiv (p \rightarrow \neg q) \rightarrow r$$

Solution To show the validity we have to show that the negated of the formula is not satisfiable. Therefore let us consider the negated of the formula, which is

$$\neg(((p \wedge q) \vee r) \equiv ((p \rightarrow \neg q) \rightarrow r))$$

and transform it in CNF. We use the trnasfromation $\neg(A \equiv B)$ is equivalent ot $(A \vee B) \wedge (\neg A \vee \neg B)$, therefore we obtain the following two formulas that can be transofrmed in CNF independently.

$$(((p \wedge q) \vee r) \vee ((p \rightarrow \neg q) \rightarrow r))$$
$$(\neg((p \wedge q) \vee r) \vee \neg((p \rightarrow \neg q) \rightarrow r)$$

If we tranfrom the first formula we obtain the following clauses

$$\{p, r\}$$
$$\{q, r\}$$

If we transform the second formula we obtain the following clauses

$$\{\neg r\}$$
$$\{\neg p, \neg q\}$$

Therefore the total set of clauses are:

$$\{p, r\}$$
$$\{q, r\}$$
$$\{\neg r\}$$
$$\{\neg p, \neg q\}$$

We can apply unit propagation obtaining

$$\{p\}$$
$$\{q\}$$
$$\{\neg p, \neg q\}$$

and by other two applications of unit propagation we derive the empty clause $\{\}$. Since we have done no branching, then there is no backtrack possible, and DPLL terminates returning UNSAT. Therefore the initial formula was Valid, □

Exercise 72:
Check the following formulas are valid with DPLL.

(1) $(p \to q) \models \neg p \to \neg q$
(2) $(p \to q) \wedge \neg q \models \neg p$
(3) $p \to q \wedge r \models (p \to q) \to r$
(4) $p \vee (\neg q \wedge r) \models q \vee \neg r \to p$
(5) $\neg(p \wedge q) \equiv \neg p \vee \neg q$
(6) $(p \wedge q) \vee r \equiv (p \to \neg q) \to r$
(7) $(p \vee q) \wedge (\neg p \to \neg q) \equiv p$
(8) $((p \to q) \to q) \to q \equiv p \to q$

Exercise 73:
Define a method to transform a propositional formula in a set of formulas of the form

$$\bigwedge_{i=1}^{n} a_i \to \bigvee_{j=1}^{m} b_j \tag{38}$$

where $\bigwedge_{i=1}^{0} a_i = \top$ and $\bigvee_{j=1}^{0} B_j = \bot$.

Solution Notice that, since $A \to B$ is equivalent to $\neg A \vee B$, the formula (38) is equivalent to

$$\neg \bigwedge_{i=1}^{n} a_i \vee \bigvee_{j=1}^{m} b_j$$

by pushing the $\neg$ operator inside the $\bigwedge$, we obtain the following equivalent formula

$$\bigvee_{i=1}^{n} \neg a_i \vee \bigvee_{j=1}^{m} b_j \tag{39}$$

But this form is a clause that contains n negative literals and m positive literals. Therefore to transform a formula ϕ in the form (38) one can first transform ϕ in CNF, and then transform all the clause of the form (39) in the form (38) by applying the inverse transformation that has been shown above. If a cluase does not contain negative literals then it is of the form

$$\bigvee_{i=1}^{m} b_i$$

which is equivalent to

$$\top \to \bigvee_{i=1}^{m} b_i$$

with the convention that $\bigwedge_{i=1}^{0} a_i$ correspond to $\top$ then a clause with no negative literals is transformed in:

$$\bigwedge_{i=1}^{0} a_i \to \bigvee_{i=1}^{m} b_i$$

which is of the form (38). Analogous argument can be done for the clauses that do not contain positive literals. □

Exercise 74:

Provide the Tseitin's Transformation of the following formula:

$$((p \vee q) \rightarrow r) \vee (r \rightarrow (p \vee q))$$

Solution The Tseitin's tranformation introduces one new propositional variable for all the non atomic subformula of the original formula. The set of non atomic subformulas of $((p \vee q) \rightarrow r) \vee (r \rightarrow (p \vee q))$ (including itself) with the associated new propositional variables are:

$$\begin{array}{ll} x_1 & ((p \vee q) \rightarrow r) \vee (r \rightarrow (p \vee q)) \\ x_2 & ((p \vee q) \rightarrow r) \\ x_3 & (r \rightarrow (p \vee q)) \\ x_4 & (p \vee q) \end{array}$$

We then define the follwong clauses

$$\begin{aligned} x_1 &\equiv x_2 \vee x_3 \\ x_2 &\equiv x_4 \rightarrow r \\ x_3 &\equiv r \rightarrow x_4 \\ x_4 &\equiv p \vee q \end{aligned}$$

and tranform them in clausal form

$$\begin{aligned} &(\neg x_1, x_2, x_3), (\neg x_2, x_1), (\neg x_3, x_1) \\ &(\neg x_2, \neg x_4, r), (x_4, x_2), (\neg r, x_2) \\ &(\neg x_3, \neg r, x_4), (r, x_3), (\neg x_4, x_3) \\ &(\neg x_4, p, q), (\neg p, x_4), (\neg q, x_4) \end{aligned}$$

□

Exercise 75:
Let us consider the formula

$$\phi = ((p \vee q) \wedge r) \rightarrow \neg s \tag{40}$$

Solution The subformulas of ϕ involved in the tranformation and the corresponding fresh propositional variables are the following:

$$\begin{array}{ll} ((p \wedge q) \vee r) \rightarrow \neg s & x_1 \\ (p \wedge q) \vee r & x_2 \\ p \wedge q & x_3 \end{array}$$

we then apply the sequence of rule (40).

$$\begin{aligned} &x_1 \wedge (x_1 \leftrightarrow ((p \wedge q) \vee r) \rightarrow \neg s) \\ &x_1 \wedge (x_1 \leftrightarrow (x_2 \rightarrow \neg s)) \wedge (x_2 \leftrightarrow ((p \wedge q) \vee r)) \\ &x_1 \wedge (x_1 \leftrightarrow (x_2 \rightarrow \neg s)) \wedge (x_2 \leftrightarrow (x_3 \vee r)) \wedge (x_3 \leftrightarrow (p \wedge q)) \end{aligned}$$

We successifely have to tranform the obtained formula in CNF Obtaining the following set of clauses

$$\begin{aligned}
\mathrm{CNF}(x_1) &= \{x_1\} \\
\mathrm{CNF}(x_1 \leftrightarrow (x_2 \rightarrow \neg s)) &= \{\neg x_1, \neg x_2, \neg s\}, \{x_2, x_1\}, \{s, x_1\} \\
\mathrm{CNF}(x_2 \leftrightarrow (x_3 \vee r)) &= \{\neg x_2, x_3, r\}, \{\neg x_3, x_2\}, \{\neg r, x_2\} \\
\mathrm{CNF}(x_3 \leftrightarrow (p \wedge q)) &= \{x_3, p\}, \{x_3, q\}, \{\neg p, \neg q, x_3\}
\end{aligned}$$

Notice that we have not introduced a new propositional variable for hte subformula $\neg s$. This is not strictly necessary since this would mean to add $x_4 \equiv \neg p$, which are transformed in the two clauses $(x_4, p), (\neg x_4, \neg p$ and not reducing any of the other clauses. $\square$

CHAPTER 5

Maximum Satisfiability

So far we have considered the interpretations of a propositinal language as a plain set. However, in many situations it is important to represent relations between propositional interpretations, i.e, to impose some structure on the set of interpretations of a propositional language. A tipical, and very important example of structure definable on the set of propositional interpretation is the one that states the fact that some interpretations are "better" then others.

EXAMPLE 5.1. *Suppose that you want to build a team of four people to develop a project that requires competences in machine learning (M), knowledge representation (K) vision (V), and human computer interaction (H). You can select the team between 6 people who have following degree of experteese for each of the competence.*

Person	*gender*	M	K	V	H
Alice	*f*	*1*	*1*	*1*	*1*
Bea	*f*	*3*	*0*	*2*	*0*
Celine	*f*	*1*	*3*	*0*	*0*
Dania	*f*	*1*	*0*	*0*	*3*
Enrico	*m*	*1*	*0*	*3*	*0*
Felix	*m*	*2*	*1*	*0*	*0*

The hard constraints on the formation of the team is that you have to select four people, and each competence should be in the team. These constraints can be expressed in terms of propositional logic formulas. For instance, the fact that you want all the four competences in the team (independently from the competence level) can be formalized by requiring that $M \wedge K \wedge V \wedge H$ is true and by stating which person can provide the various competences (independently from the expertese level) using the following implications:

$$\begin{array}{ll} M & \rightarrow A \vee B \vee C \vee D \vee E \vee F \\ K & \rightarrow A \vee C \vee F \\ V & \rightarrow A \vee B \vee E \\ H & \rightarrow A \vee D \end{array} \tag{41}$$

The constraint about the team size can be expressed by the cardinality constraints, exactly four among $A, \ldots, F$. I.e.,

$$A + B + C + D + E + F = 4 \tag{42}$$

There are many choices that satisfy this constraints, they correspond to the assignments to $A, \ldots, F$ that satisfy the above formulas. All the interpretations that satisfy formulas (41) *and* (42) *are shown in Figure 1 For every interpretation $\mathcal{I}_i$ that satisfy the hard constraint you can extract the corresponding team. However, you would also like to be able to express some preference on the teams as, for instance, you prefer teams with gender and competence balance, but also the higher*

$$\mathcal{I}_1 = \{A, B, C, D, M, K, V, H\} \qquad \mathcal{I}_8 = \{A, C, D, F, M, K, V, H\}$$
$$\mathcal{I}_2 = \{A, B, C, E, M, K, V, H\} \qquad \mathcal{I}_9 = \{A, C, E, F, M, K, V, H\}$$
$$\mathcal{I}_3 = \{A, B, C, F, M, K, V, H\} \qquad \mathcal{I}_{10} = \{A, D, E, F, M, K, V, H\}$$
$$\mathcal{I}_4 = \{A, B, D, E, M, K, V, H\} \qquad \mathcal{I}_{11} = \{B, C, D, E, M, K, V, H\}$$
$$\mathcal{I}_5 = \{A, B, D, F, M, K, V, H\} \qquad \mathcal{I}_{12} = \{B, C, D, F, M, K, V, H\}$$
$$\mathcal{I}_6 = \{A, B, E, F, M, K, V, H\} \qquad \mathcal{I}_{13} = \{B, D, E, F, M, K, V, H\}$$
$$\mathcal{I}_7 = \{A, C, D, E, M, K, V, H\} \qquad \mathcal{I}_{14} = \{C, D, E, F, M, K, V, H\}$$

FIGURE 1. The models of the formulas (41) and (42), In red the people of the team.

		Competence level			
Team	*gb*	*M*	*K*	*V*	*H*
$Team_1 = \{A, B, C, D\}$	0.0	6	4	3	4
$Team_2 = \{A, B, C, E\}$	0.5	6	4	6	1
$Team_3 = \{A, B, C, F\}$	0.5	7	5	3	1
$Team_4 = \{A, B, D, E\}$	0.5	6	1	6	4
$Team_5 = \{A, B, D, F\}$	0.5	7	2	3	4
$Team_6 = \{A, B, E, F\}$	1.0	7	2	6	1
$Team_7 = \{A, C, D, E\}$	0.5	4	4	4	4
$Team_9 = \{A, C, D, F\}$	0.5	5	5	1	4
$Team_9 = \{A, C, E, F\}$	1.0	5	5	4	1
$Team_{10} = \{A, D, E, F\}$	1.0	5	2	4	4
$Team_{11} = \{B, C, D, E\}$	0.5	6	3	5	3
$Team_{12} = \{B, C, D, F\}$	0.5	7	4	2	3
$Team_{13} = \{B, D, E, F\}$	1.0	7	1	5	3
$Team_{14} = \{C, D, E, F\}$	1.0	5	4	3	3

FIGURE 2. Ranking of the teams w.r.t, the different criteria.

the total amount of each competence the better. In summary you can rank all the different teams that satisfy the hard criteria according to some preference criteria. We can for instance consider the gender balance criteria $gb = 1 - \frac{|\#male - \#female|}{4}$ *and the crieria of the sum of the competence level for each competence. The evaluation of the four teams (interpretations) according to tese 5 criteria are shown in Figure* **??**.

There is no team that maximizes all the criteria, hoever one could decide to give priority to the gender balance, preferring $team_2$ *and* $team_4$ *and then to the uniform distribution among competence. which implies that* $team_2$ *is seleccted.*

EXAMPLE 5.2. *Consider the situation in which you have to build a team of* n *people with a good gender balance. You will prefer teams with gender balance degree, defined as* $b = 1 - \frac{|\#male - \#female|}{n}$ *is close to* 1. *The best option would be a team with an even number of male and female, with* $b = 1$. *Formalizing it in propositional logic, if* p_i *represents the proposition that the* i*-th member of the team is a female,*

you prefer the interpretations in which the fraction of p_i set to true and those set to false are closer. This amount in introducing an order in the set of interpretations as shown in Figure3 following picture that shows how interpretations of $p_1, \ldots, p_4$ (denoted by a sequence of four 0/1) can be orderred according to their preference.

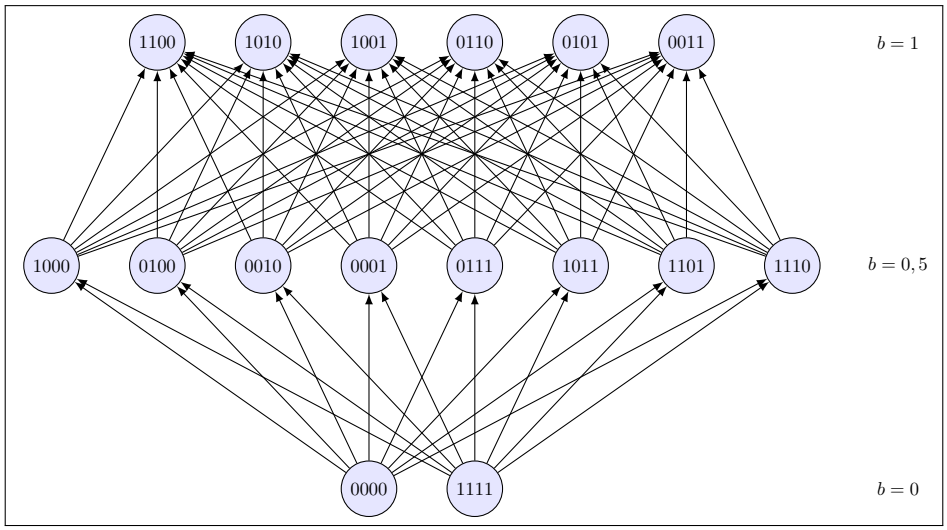


FIGURE 3.

Maximum satisfiability problem focus on this type of relation, which formally correspond to partial orders. In particular maximum satisfiabiliy focus on the problem of finding the "best" interpretation that satisfy a certain set of formulas. However, Before proceeding with the definition of the problem of maximum satisfiability and the relative solution methods let us discuss how it is possible to impose an ordering structure on the set of interpretations of a propositional language.

1. Ordering interpretations

To state that an interpretation is "better" than another, of that we "prefer" an interpretation w.r.t., another we should be able to order the set of interpretations from the less preferred to the most preferred. The mathematical notion that can be used for this aim is the notion of *preorder*

1.1. Partial and total orders. A *preorder* is a structure $(S, \precsim)$ where S is a set and $\precsim$ is a binary relation on S i.e., $\precsim \subseteq S \times S$ that satisfies the following properties: v

Reflexivity:: $s \precsim s$, for all $s \in S$
Transitivity:: $s \precsim t$ and $t \precsim u$ implies that $s \precsim u$.

The preorder is said to be *total* if

Totality:: $s \precsim s$ or $t \precsim s$ for all $s, t \in S$

We say that s and t are *equi-preferrable*, in symbol $s \sim t$ if $a \precsim t$ and $t \precsim s$. Finally we say that t is *strictly preferable* to s, in symbol $s \prec t$ if $s \precsim t$ but not $s \sim t$. One of the simplest way to specify a total preorder on the set $\mathcal{S}$ is by defining a function

$w : \mathcal{S} \to \mathbb{R}$, often called *weight function*, that associates to each element $s \in S$ a real number $w(s)$. The order on S is defined as $s \precsim t$ if and only if $w(s) \leq w(t)$. In the following we will use only total preorders specified by some weight function.

Let $\mathcal{P}$ be a set of propositional variables, and $\mathbb{I}$ be the set of interpretations of $\mathcal{P}$, a *weight function* for $\mathcal{P}$ is a function $w : \mathbb{I} \to \mathbb{R}$, that associates to each interpretation $\mathcal{I}$ of $\mathcal{P}$ a real number $w(\mathcal{I})$. Such a function defines the following total preorder on the models of ϕ

$$\mathcal{I} \precsim \mathcal{J} \text{ if and only if } w(\mathcal{I}) \leq w(\mathcal{J})$$

It is easy to show that this definition satisfies the property that defines a total pre-order (by exercise).

EXAMPLE 5.3. *Consider the example of forming a gender balanced team introduced above. Let $\mathcal{P} = \{p_1, \ldots, p_n\}$. If we define the weight function as:*

$$w(\mathcal{I}) = -\left|\textstyle\sum_i \mathcal{I}(p_i) - \frac{n}{2}\right| \tag{43}$$

we have that $w(\mathcal{I})$ reaches it's maximum equal to 0, when there is an even number of variables set to true and false (if n is even), or $-\frac{1}{2}$, which is reached when the number of variables assigned to true and false differs of one unit, in case n is odd.

Notice that, the nominal value of the weight of an interpretation is not really important, what matters is the order between the weights that determines the order of the interpretations. In the above exaple, for instance, if we define the weight function as $w(\mathcal{I}) = -(\sum_i \mathcal{I}(p_i) - \frac{n}{2})^2$ we obtain exactly the same order betwen the interpretations.

1.2. Specifying weight function with weighted formulas. In the most general case, the specification of $w : \mathbb{I} \to \mathbb{R}$ could involce the specification of $2^n - 1$ parameters. The "-1" is due to the fact that, without loss of generality we can suppose that the "worse" interpretation is weightd $-\infty$. However there are more compact and easy to interpret ways to specify a weight functionf on interpretations, one of this is by associating weights to formulas.

Let $F = \{w_i : \phi_i\}_{i=1}^n$ be a multiset[1] of of n propositional formulas the set of propositional variables $\mathcal{P}$ each of which is assigned a real number, called *weight*. We can use F to define a weight function on the set of truth assignments of $\mathcal{P}$ as follows:

$$w_F(\mathcal{I}) = \sum_{w:\phi \in F} w \cdot \mathcal{I}(\phi) \tag{44}$$

which implies that the ordering $\prec_F$ is defined as

$$\mathcal{I} \prec_F \mathcal{J} \Leftrightarrow \sum_{w:\phi \in F} w \cdot \mathcal{I}(\phi) \leq \sum_{w:\phi \in F} w \cdot \mathcal{J}(\phi)$$

One important intuition to bear in mind is the following: For every weighted formula $w : \phi \in F$

- if $w_i > 0$ we prefer interpretations that satisfy ϕ;
- if $w_i < 0$ we prefer interpretations that do not satisfy ϕ;
- if $w_i = 0$ we are indifferent about the truth value of γ.

[1]A multiset is a set that can contain multiple copies of the same elements. For instance $\{1, 2, 3, 5, 2, 1\}$ is a multiset, which is equal to $\{1, 1, 2, 2, 3, 5\}$.

1.3. Properties of weight function. In the MaxSAT problem the nominal weight of a model in not important, what matters is the ordering that a certain weight function induces on a set of iunterpretations. As a consequence the same ordering can be obtained by different weight functions. In the following we define the notion of *equivalence* between weight functions, that intuitively means that they define the same partial order on a set of interpretations. We also introduce the notion of a weight formula being the *opposite* of another weight formula with the intuitive meaning that the order defined by the two weight funciton are one the inverse of the other.

DEFINITION 5.1. *Two sets of weighted formulas on the propositional variables $\mathcal{P}$, F_1 and F_2 are* equivalent *if they define the same order. I.e., if for all interpretations $\mathcal{I}, \mathcal{J}$ of $\mathcal{P}$*

$$w_1(\mathcal{I}) < w_1(\mathcal{J}) \text{ if and only if } w_2(\mathcal{I}) < w_2(\mathcal{J})$$

Two sets of weighted formulas F_1 and F_2 are opposite *if*

$$w_1(\mathcal{I}) < w_1(\mathcal{J}) \text{ if and only if } w_2(\mathcal{J}) < w_2(\mathcal{I})$$

PROPOSITION 5.1. *(1) F is equivalent to $a \cdot F = \{a \cdot w : \phi \mid w : \phi \in F\}$ for $a > 0$;*
(2) F is opposite of $a \cdot F = \{a \cdot w : \phi \mid w : \phi \in F\}$ for $a < 0$;
(3) $F \cup \{w : \phi\}$ is equivalent to $F \cup \{-w : \neg\phi\}$
(4) If $\models \phi \leftrightarrow \psi$, then $F \cup \{w : \phi\}$ is equivalent to $F \cup \{w : \psi\}$;
(5) $F \cup \{w_1 : \phi,\ w_2 : \phi\}$ is equivalent to $F \cup \{w_1 + w_2 : \phi\}$

PROOF. Let use w_F to denote the weight function defined by the set of weighted formulas F. Notice that $w_{a\cdot F}(\mathcal{I}) = a \cdot w_F(\mathcal{I})$. Indeed

$$\begin{aligned} w_{a\cdot F}(\mathcal{I}) &= \sum_{w:\phi\in F} a \cdot w \cdots \mathcal{I}(\phi) \\ &= a \cdot \sum_{w:\phi\in F} w \cdots \mathcal{I}(\phi) = a \cdot w_F(\mathcal{I}) \end{aligned}$$

Property (1) and (2) directly derives from this fact. For property (3) we have that

$$\begin{aligned} w_{F\cup\{-w:\neg\phi\}} &= w_F(\mathcal{I}) - w \cdot (1 - \mathcal{I}(\phi)) \\ &= w_F(\mathcal{I}) - w + w \cdot \mathcal{I}(\phi) \\ &= w_{F\cup\{w:\phi\}}(\mathcal{I}) - w \end{aligned}$$

This implies that

$$\begin{aligned} w_{F\cup\{w:\phi\}}(\mathcal{I}) < w_{F\cup\{w:\phi\}}(\mathcal{J}) &\Leftrightarrow w_{F\cup\{w:\phi\}}(\mathcal{I}) - w < w_{F\cup\{w:\phi\}}(\mathcal{J}) - w \\ &\Leftrightarrow w_{F\cup\{-w:\neg\phi\}}(\mathcal{I}) < w_{F\cup\{-w:\neg\phi\}}(\mathcal{J}) \end{aligned}$$

The proof of properties (1) and (2) are left by exercize. □

Property (1) of Proposition 5.1 states that, if we re-scale the weights of a positive factor, the order on the interpretations does not change. Instead property (2) says that if we rescale with a negative factor then we obtaing the opposite ordering. Property refitem:F-phi-equiv-F–not-phi states that the weight function obtained by inverting the weight associated to the formula and negating the formulas differs from a constant from the original weight. This implies that optimizing the two weight function will lead to the same result. This property guarantees that,

without loss of generality we can assume that all the weights are positive. Indeed every negatively weighted formula $w : \phi$ can be replaced by the positively weighted formula $-w : \neg\phi$, without changing the order between the interpretations. Property (1) implies that, if two formulas are logically equivalent, then adding one with a weight or the other with the same weight has the same effect. In other words the specification of the weight function using weighted formulas is independent from the syntactic spefication of the formula but depends only from the semantics of the formula.

2. The MaxSAT problem

There are various versions of MaxSAT problems

- *Basic MaxSAT:* There are no hard clauses and all the soft clauses have the same weight (equal to 1). The solution of this problem is the assignment that satisfies the maximize number of soft clauses, or equivalently minimize the number of unsatisfied soft clauses.
- *Partial MaxSAT:* the set of hard clauses could be not empty and the soft clauses have the same weight (equal to 1). The solution need to satisfy them and to minimize the number of unsatisfied soft clauses
- *Weighted MaxSAT:* No hard clauses and different weights associated with soft clauses The solution has to minimize the sum of weights of unsatisfied soft clauses.
- *Weighted Partial MaxSAT:* The set of hard clauses could be non empty and they need to be satisfied by the solution. The soft clauses can be associated with different weight, and the solution has to minimize the sum of the weeight of the soft clauses that are not satisfieed.

The last version os the most general version and it. If no specification is given with the terms MaxSAT, we refer to this general formulation. In the following we provide formal definitions of the different versions.

A general definition of the maximum satiisfiability problem is the following:

Definition 5.2 (General maximum satisfiability problem). *Given a partial order $(\mathbb{I}, \prec)$ defined on the interpretations of a set of propositional variables $\mathcal{P}$, and a formula ϕ, the* maximum satisfiability problem *is the problem of finding a model $\mathcal{I}^*$ of ϕ such that:*

$$\mathcal{I}^* \in \sup_{\prec}(\{I \in \mathbb{I} \mid \mathcal{I} \models \phi\}) \tag{45}$$

When $\prec$ is a total order, then it can be specified by a weight function $w : \mathbb{I} \to \mathbb{R}$, then the problem of maximum satisfiability can be rewritten as the problem of finding the maximum of the weighted formula, i.e.

$$\mathcal{I}^* = \operatorname*{argmax}_{\mathcal{I} \models \phi} w(\mathcal{I}) \tag{46}$$

The literature contains multiple definition and variants of the MaxSat problem, that are specific cases, or can be reformulated in terms of (46). All the approaches to MaxSAT assumes that weighted and hard formulas are specified in CNF. In the following we report the various definitons. of the different MaxSAT problems

Definition 5.3 (Unweighted MaxSat). *Given a set of clauses $C_1, \ldots, C_n$, the* unweighted maximum satisfiability problem *is the problem of finding an assignment*

that maximizes the total number of satisfied clauses:, i.e.,

$$\mathcal{I}^* = \operatorname*{argmax}_{\mathcal{I}} \sum_i \mathcal{I}(C_i) \tag{47}$$

Much studied in Theoretical Computer Science are dedicated to the MaxSat formulation (47). This formulation can is a special case of the general definition (45) where ϕ is $\top$ and the weight function speficied by the weighted formulas $1 : C_1$, $\dots 1 : C_n$. It has been proved that unweighted MaxSat is NP-complete. Even Max2Sat, the restriction to instances in which each clause has at most two literals in it, is NP-complete.

DEFINITION 5.4 (Weighted MaxSat). *Given a set of weighted clauses* $w_1 : C_1, \dots, w_n : C_n$*, the* weighted maximum satisfiability problem *is the problem of finding an assignment that maximizes the sum of the weights of the clauses satisfied by the assignment.*

$$\mathcal{I}^* = \operatorname*{argmax}_{\mathcal{I}} \sum_i w_i \cdot \mathcal{I}(C_i) \tag{48}$$

DEFINITION 5.5 (Partial MaxSat). *Given a set of clauses* $C_1, \dots, C_n$*, called* soft clauses*, and a second set of clauses* $D_1, \dots, D_n$*, called* hard clauses*, the* partial maximum satisfiability problem *is the problem of finding an assignment that satisfies the hard clauses and that maximizes the number of satisfied soft clauses:*

$$\mathcal{I}^* = \operatorname*{argmax}_{\mathcal{I} \models D_1 \dots, D_n} \sum_i \mathcal{I}(C_i) \tag{49}$$

DEFINITION 5.6 (Partial weighte MaxSat). *Given a set of weighted clauses* $C_1, \dots, C_n$*, called* soft clauses*, and a second set of clauses* $D_1, \dots, D_n$*, called* hard clauses*, the* partial maximum satisfiability problem *is the problem of finding an assignment that satisfies the hard clauses and that maximizes the sum of the weight of the satisfied soft clauses:*

$$\mathcal{I}^* = \operatorname*{argmax}_{\mathcal{I} \models D_1 \dots, D_n} \sum_i w_i \cdot \mathcal{I}(C_i) \tag{50}$$

Inspite of the different definition each of the formulation can be rewritten in terms of the others and in particular a general MaxSat problem can be rewritten (in polinomial time) in an equivalent unweighted MaxSat problem.

This can be donw in the following way:

PROPOSITION 5.2. *Let* $\{D_i\}$ *and* $\{w_i : C_i\}$ *be a set of hard and soft clauses respectively.*

1. *Let* δ *be the the smallest value* $|w_i - w_j|$ *different from* 0*, otherwise let* $\delta = w_i$*;*
2. *Let* $v_i = \lceil \frac{w_i}{\delta} \rceil$
3. *let* $v^* = \sum_i v_i + 1$
4. *let* $\mathcal{C}$ *be the multiset of clauses that contains* v^* *copies of each hard clause and* v_i *copies of each clause* v_i*.*

I is the solution of the partial weighted maximum satisfiability problem (50) *if and only if* $\mathcal{I}$ *is the solution of the unweighted maximum satisfiability problem* (47) *on* $\mathcal{C}$ *and* $\mathcal{I} \models \boldsymbol{D}$ *where* $\boldsymbol{D} = \bigwedge_i D_i$*.*

PROOF. By exercize. Hint. First show that if $\mathcal{I} \not\models \boldsymbol{D}$ and $j \models \boldsymbol{D}$ then $w(\mathcal{I}) < w(\mathcal{J})$. Then show that Then show that if $\mathcal{I} \models \boldsymbol{D}$ and $\mathcal{J} \models \boldsymbol{D}$, then $w(\mathcal{I}) < w(J)$ if and only if the number of clauses in $\mathcal{C}$ satisfied by $\mathcal{I}$ are less then the number of clauses satisfied by $\mathcal{J}$. □

3. MaxSAT exact algorithms

There are different approaches to solve the MaxSAT problem. They can be classified in exact algorithms, which provide an exact solution to the problem, and approximated algorithms, which guarantees only sub-optimal solutions. All the algorithms assumes that weighted formulas are clauses. In this section we will present some of the basic algorithms of the first categori. Inside this category we can distinguish three main apporaches:

- Branch and bound algorithns;
- Transformation into Integer Programming;
- Algorithms that use SAT as oracle;
- Algorithms based on implicit hitting sets.

In the following section we introduce the basis of each of the above categories.

3.1. Branch and Bound. Branch and Bound (B&B) algorithms explore the search tree of all partial assignments for the soft and hard clauses, in a depth-first manner, in order to find the interpretation thas satisfy all the herd clauses and maximizes the weight of the satisfied soft clauses. For every interpretation, $\mathcal{I}$ let *loss*($\mathcal{I}$) is the sum of the weights of the clauses that are not satisfied by $\mathcal{I}$. Solving the MaxSAT problems coincides to find the interpretation $\mathcal{I}$ that minimizes *loss*($\mathcal{I}$).

Let us start by introducing a base algorithm that performs an exhaustive search of all the assignments that satisfies the hard clauses and select the one with minimal loss (or equivalently maximal weigh). The search lgorithm for MasSAT can be obtained by modifying the DPLL decision procedure, which searches for *any* assignment that satisfies a formula ϕ, so that it does not stop when one model of ϕ is found, but it continues to search other models of ϕ. The pseudocode is shown in Algorithm 5.

EXAMPLE 5.4. *Let us see with a simple example how the algorithm works. Consider the following sets of hard and soft clauses:*

$$\phi = \left\{ \begin{array}{c} \{A, B, C\} \\ \{\neg A, \neg B, \neg C\} \end{array} \right\} \qquad \psi \left\{ \begin{array}{c} 2{:}\{A, \neg B\} \\ 3{:}\{\neg A, C\} \\ 4{:}\{B, \neg C\} \end{array} \right\}$$

Aa possible expansion of the search tree of the max-DPLL procedure is showin in Figure 4. Notice that the best assignment, is the one with minimal loss which is equal to 2. In the tree we choose the literals $\neg A$, $\neg B$, and $\neg C$ in this order.

Algorithm 5 MAX-DPLL(ϕ : CNF, ψ : weighted CNF, $\mathcal{I}$: Partial assignment)

```
1: I, φ, ψ ← UnitPropagation(I, φ, ψ)
2: if {} ∈ φ then
3:     return I, ∞ c
4: end if
5: if φ = {} and ψ contains only empty weighted clauses then
6:     return I, Σ_{(w:C)∈ψ} w
7: else
8:     select a l from a clause in φ or in ψ
9:     I, loss ← Max-DPLL(φ|_l, ψ|_l, I ∪ {l})
10:    I', loss' ← Max-DPLL(φ|_l̄, ψ|_l̄, I ∪ {l̄})
11:    if loss ≤ loss' then
12:        return I, loss
13:    else
14:        return I', loss'
15:    end if
16: end if
```

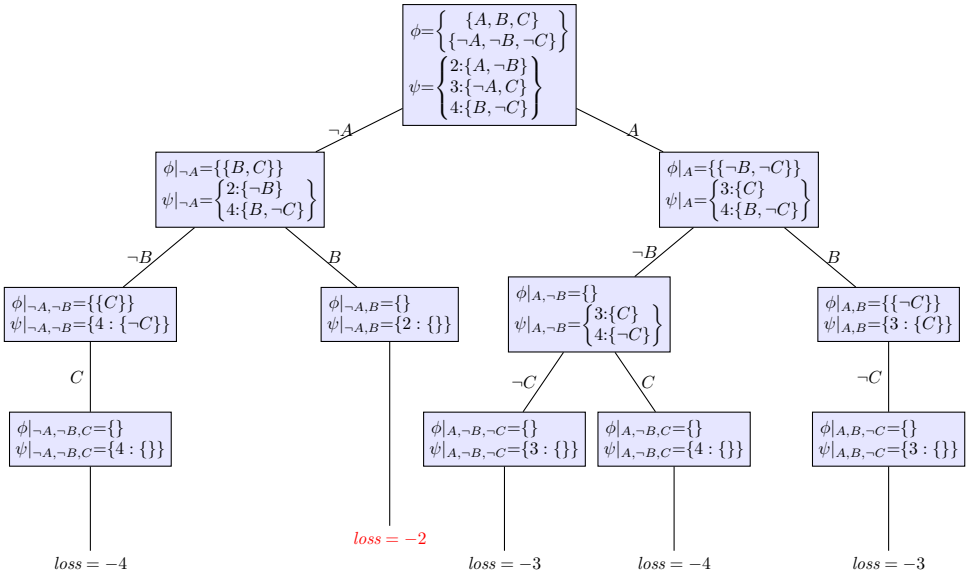


FIGURE 4. The search tree of the max-DPLL procedure

The MAX-DPLL algorithm recursively performs a depth-first visit of the tree of the partial assignments to the propositional variables of the input, searching for the assignment that minimizes the loss. MAX-DPLL takes in input a partial assignment, which is empty in the first call, a set of hard clauses ϕ and a set of soft clauses with the associated weights. MAX-DPLL starts by applying unit propagation (line 1) only on hard clauses, This means that, for all $\{l\} \in \phi$ the current partial assignment $\mathcal{I}$ is extended with $\{l\}$ and ϕ is set to $\phi|_l$ and ψ to $\psi|_l$. UNITPROPAGATION repeatedly applies this reduction until ϕ does not contain

any unit clause. Then, if the set of hard clauses contains an empty clause (i.e., the current assignment does not satisfy the hard clauses), then MAX-DPLL returns the current partial assignment $\mathcal{I}$ with infinite loss (line 3), In this situation the infinite loss has the effect that this solution will be the worse possible from the minimisation point of vies, and any other solution with finite cost would be preferred to it. If instead (line 5) the set of hard clauses are empty (i.e., all of them are satisfied by the current partial interpretation $\mathcal{I}$) and the soft clauses contains only empty (i.e.,, unsatisfiable) clauses, then the MAXSAT-DPLL returns the current interpretation with its cost, which is the sum of the weight of the soft clauses, which are all empty, and therefore they are not satisfied by $\mathcal{I}$. Otherwise, i.e., if there are hard clauses that need to be satisfied, and soft clauses that could be satisfied, (i,e, if $C \in \phi$ for some non empty C and $w : D \in \psi$ for some non empty D), the algorithm computes the solutions that contains the current assignment extended with l, (line 9) and with $\bar{l}$ (line 10) for some literal l, and choose the one of minimal cost (lines 11–14)

The MAX-DPLL algoritm is not very useful in practice as it potentially visits the entire tree of all the partial interpretations, which is exponentially large, w.r.t., the size of the input clauses. One should find some criteria for early stopping the search when there is some evidence that this will not lead to any better solution. Suppose that MAX-DPLL has already found a solution $\mathcal{I}$ with loss equal to x. Then every assignment with loss greater than x are not solutions of the maxSat problem. This means that x acts as an upper bound on the cost of the solutions, let denote x with UB. If MAX-DPLL is expanding anothe partial interpretation $\mathcal{I}'$. Let LB be the minimal loss of all the assignments which are expansions of $\mathcal{I}'$. A naïve wey to find a value for LB is by summing the weights of the soft-clauses not satisfied by $\mathcal{I}'$. If $LB \geq x$, then the loss of any extension of $\mathcal{I}'$ will have cost larger or equal to x, and they will not be better than the solution $\mathcal{I}$ already found which means that we can stop expanding $\mathcal{I}'$ and look to other alternative partial interpretations. This idea is implemented in the *Branch and Bound* algorithm shown in Algorithm 6. Algorithm B&B takes in input two additional parameters than MAX-DPLL which

Algorithm 6 B&B(ϕ: CNF, ψ: Weighted CNF, $\mathcal{I}$: Partial assignment, $\mathcal{I}_{UB}$: Best previously found solution, UB: Cost of $\mathcal{I}_{UB}$)

```
1: I, φ, ψ ← UnitPropagation(I, φ, ψ)
2: LB ← lowerBound(φ, ψ)
3: if {} ∈ φ or UB ≤ LB then
4:     return I_UB, UB
5: end if
6: if φ = {} and ψ contains only empty weighted clauses then
7:     return I, Σ_{(w:D)∈ψ} w
8: else
9:     select a l from some clause in φ or in ψ
10:    I, UB ← B&B(φ|_l, ψ|_l, I ∪ {l}, I_UB, UB)
11:    I', UB' ← B&B(φ|_l̄, ψ|_l̄, I ∪ {l̄}, I_UB, UB)
12:    return I', UB'
13: end if
```

are the best solution found sofar $\mathcal{I}_{UB}$ and its loss UB. B&B proceeds as MAX-DPLL with the only exception that it first computes a lower bound of the current

partial assignment (line **??**) and checks it it is larger than than the loss of the best solution found until now (line **??**). In this case the previous solution is returned. The first call of B&B is done with the empty interpretation $\mathcal{I}$, and $\mathcal{I}_{UB}$ and the infinite cost $UB = \infty$.

The simplest way to compute the lower bound of a partial interpretation is by summing the weights of the empty clauses in ψ. i.e.,

$$\text{LOWWERBOUND}(\phi, \psi) = \sum_{w:\{\}\in\psi} w \tag{51}$$

Later we will see more sophisticated algorithm that compute higher lower bounds and therefore that prevents B&B to explore larger parts of the search space. For the time being, let us see an example of B&B with this simple method of estimating lower bound.

EXAMPLE 5.5. *Consider the following set of hard and soft clauses with relative weight (hard clauses are labelled with infinite weight).*

$$\begin{array}{lll} (\neg a \vee b \vee c : \infty) & (a:1) & (c:3) \\ (\neg b \vee d : \infty) & (b:2) & (d:2) \\ (\neg d \vee \neg a : \infty) & & \end{array}$$

3.1.1. *Lower bound computation.* One step forward w.r.t., (51) can be done by incorporating an underestimation of the sum of the weights of clauses that will become unsatisfied if the current partial assignment is extended to a complete assignment.

EXAMPLE 5.6. *Suppose that ψ contains the two weighted unary clauses $w : \{x\}$ and $v : \{\neg x\}$. This implies that every assignment for ϕ, ψ will falsify one of the two clauses. Therefore, the loss for ϕ, ψ will not be less then* $\min(w, v)$. *A little more complicated example is when ψ contains multiple pairs of contradicting unary weighted clauses.*

$$\left\{ \begin{array}{c} w_1 : \{x_1\}, v_1 : \{\neg x_1\} \\ w_2 : \{x_2\}, v_2 : \{\neg x_2\} \\ \cdots \\ w_n : \{x_n\}, v_n : \{\neg x_n\} \end{array} \right\} \subseteq \psi$$

then every assignment for ϕ, ψ will falsify one of the two unary clause for every pair. Therefore a lowerbound for the loss is therefore equal to:

$$\sum_{i=1}^{n} \min(w_i, v_i)$$

Finally suppose that ϕ contains the hard clause $\{a, b\}$ and ψ contains the two weighted unary clauses $w : \{\neg a\}$ and $v : \{\neg b\}$ Then every assignment that satisfy ϕ will not satisfy one of the two weighted unary clauses, which implies that the loss will be at least $\min(w, v)$.

The following method to find a lowerbound for the loss of a maxSAT problem ϕ, ψ is a generalization of the previous example and uses a SAT solver as an oracle.

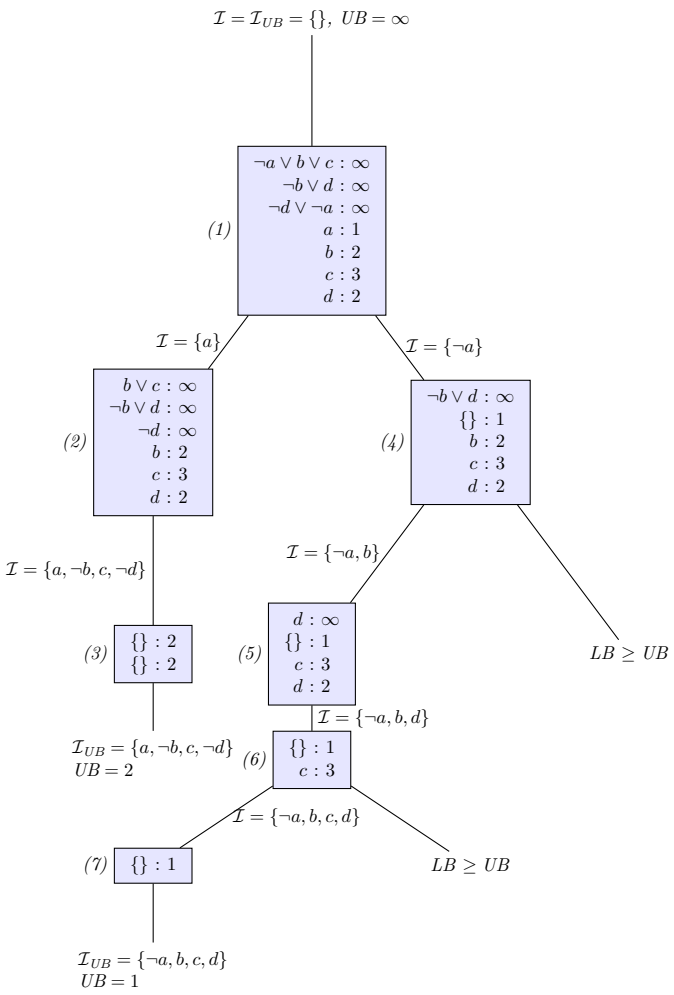


FIGURE 5. Exploration tree of the B&B algorithm for the hard and soft clauses of Example 5.5. Nodes are labelled in order of expansion. B&B find the first solution at node (3), the cost of this solution is 4. Therefore $\mathcal{I}_{UB}$ is set to be this solution and the UB (upper bound cost) is set to be 4. Then the algorithm find a second solution at node (7) whose cost is equal to 1 smaller than UB. Therefore $\mathcal{I}_{UB}$ is set to this new solution and UB is set to 1. Successively the B&B reaches the node (8), which does not correspond to a solution. but the solutions that are derivable from the partial assignment at this node will have a cost higher or equal to 3 (the sum of the costs of the two empty soft clauses) which is higher than the current UB and therefore B&B stops and returns $\mathcal{I}_{UB}$ and UB.

Suppose that ψ contains n disjoint subsets $\psi_1, \ldots, \psi_n$ and each $\{C \mid w : C \in \psi_i\} \cup \phi$

is not satisfiable, then

$$\text{lowerBound}(\phi,\psi)=\sum_{i=1}^{n}\min_{w:C\in\psi_i} w \tag{52}$$

Indeed notice that every assignment that satisfy ϕ will not satisfy at least one clause in each ϕ_i. Since we don't know which of the clauses of ϕ_i will be falsified, we can only infer that the loss will be increased with the minimum weights among those of the clauses in ϕ_i. Notice that (51) is a special case of (52).

Example 5.7. *Consider the set of hard and soft clauses:*

$$\phi=\left\{\begin{array}{c}\{a,b\}\\ \{\neg a,\neg b\}\end{array}\right\} \qquad \psi=\left\{\begin{array}{ll}1{:}\{a,\neg c\} & 2{:}\{\neg a,c\}\\ 3{:}\{b,\neg c\} & 4{:}\{\neg b,c\}\\ 5{:}\{c\} & 6{:}\{\neg c\}\end{array}\right\}$$

ψ contains the following two disjoint sets of weighted clauses

$$\psi_1=\left\{\begin{array}{l}1{:}\{a,\neg c\}\\ 3{:}\{b,\neg c\}\\ 5{:}\{c\}exercize\end{array}\right\} \qquad \psi_2=\left\{\begin{array}{l}2{:}\{\neg a,c\}\\ 4{:}\{\neg b,c\}\\ 6{:}\{\neg c\}\end{array}\right\}$$

such that, if we extend their unweighted version with ϕ, we obtain the following two sets of clauses

$$\left\{\begin{array}{c}\{a,b\}\\ \{\neg a,\neg b\}\\ \{a,\neg c\}\\ \{b,\neg c\}\\ \{c\}\end{array}\right\} \qquad \left\{\begin{array}{c}\{a,b\}\\ \{\neg a,\neg b\}\\ \{\neg a,c\}\\ \{\neg b,c\}\\ \{\neg c\}\end{array}\right\}$$

which are both unsatisfiable. This means that every assignment does not satisfy at least one clause in ψ_1 and one in ψ_2. This implies that the minimum value of the loss of any assignments for ϕ,ψ is $1+2=3$ which is the sum of the minimum weights of the clauses in ψ_1 and ψ_2.

3.1.2. *Literal selection.* One of the key aspect of the B&B algorithm is the literal selection performed on line 9. Selecting the "right" literal will avoid the expansion of many parts of the search tree and go straight to the maxSAT solution. For instance in figure 5 if the algorithm would have selected $\neg a$ instead of a in the first branching it would have reached the solution straightaway without expanding the subtree on the left (the one under the selection of a). Most of the exact MaxSAT solvers incorporate variable selection heuristics that take into account the number of literal occurrences in such a way that each occurrence has an associated weight that depends on the length of the clause that contains the literal. MaxSAT heuristics give priority to literals occurring in binary clauses instead of literals occurring in unit clauses as SAT heuristics do. Things are getting to technical and detailed here. So I decided to stop here with this topic. For those interested in the argument you can check works like Mohamedou and Planes 2009,

3.2. Core based algorithm. Core based algorithms for maximum satisfiability uses a SAT solver as subrutine. The main idea behind the core based approaches is the following:

If ϕ,ψ is satisfiable then any interpretation $\mathcal{I}$ that satisfies ϕ and ψ is a solution of the MaxSAT problem with cost equal to 0. If ϕ,ψ the ideas is to weakening the

soft clauses ψ (by adding literals to some of the clauses in ψ) but at the same time impose some additional constraints by extending the hard clauses, requiring that the added literals will be minimally true. Then check the satisfiability of the new MaxSAT problem. Let us see a concrete example on how this works:

EXAMPLE 5.8. *Consider the MaxSAT instance:*

$$\phi = \begin{Bmatrix} \{p, q\} \\ \{\neg p, \neg r\} \\ \{\neg q, \neg r\} \end{Bmatrix} \qquad \psi = \begin{Bmatrix} 1{:}\{p\} \\ 1{:}\{q\} \\ 1{:}\{r\} \end{Bmatrix}$$

Notice that the union of ϕ with (the unweighted version of) ψ is not satisfiable. Which implies that a solution of this simple MaxSAT should not satisfy at least one of the soft clauses. The idea is to weaken the soft clauses so that they become satisfiable and add a constraint that states that only one of them can be falsified.

To weaken a soft clause C we can add a fresh atom b obtaining $C \cup \{b\}$. In our example we have

$$\psi_1 = \begin{Bmatrix} 1{:}\{p, b_1\} \\ 1{:}\{q, b_2\} \\ 1{:}\{r, b_3\} \end{Bmatrix}$$

and then add the constraint that at most one among b_1, b_2 and b_3 can be true. I.e., $\sum_{i=1}^{3} b_i$. The new MaxSAT problem becomes:

$$\phi^{(1)} = \begin{Bmatrix} \{p, q\} \\ \{\neg p, \neg r\} \\ \{\neg q, \neg r\} \\ \sum_{i=1}^{3} b_i \leq 1 \end{Bmatrix} \qquad \psi^{(1)} = \begin{Bmatrix} 1{:}\{p, b_1\} \\ 1{:}\{q, b_2\} \\ 1{:}\{r, b_3\} \end{Bmatrix}$$

Notice that every interpretation $\mathcal{I}$ that satisfies $\phi^{(1)}$ and the unweighted version of $\psi^{(1)}$ must satisfy at least one b_i. Otherwise the initial $\phi \cup \psi$ would have been satisfiable. In other words b_i being true indicates that the initial i-th clause can be falsified. Furthermore it $\mathcal{I}$ is a solution of the MaxSAT problem $\phi^{(1)}, \psi^{(1)}$ it will also be a solution of the initial MaxSAT problem with a cost augmented of one unit.

In this simple case we have that $\phi^{(1)} \cup \psi^{(1)}$ is satisfiable, by the assignment:

$$p = 1 \qquad q = 0 \qquad r = 0 \qquad b_1 = 0 \qquad b_2 = 0 \qquad b_3 = 1$$

which contains a solution of the initial MaxSAT problem, with cost/loss equal to 1.

One can minimize the number of soft clauses to weaken by adding new variable, by considering only the minimal set of weak clause ψ' such that $\phi \cup \psi'$ is inconsistent. In the example we have two alternatives such a minimal sets.

$$\psi' = \begin{Bmatrix} 1{:}\{p\} \\ 1{:}\{r\} \end{Bmatrix} \qquad \psi'' = \begin{Bmatrix} 1{:}\{q\} \\ 1{:}\{r\} \end{Bmatrix}$$

We can consider one of the two subsets and proceed as before by adding two new variables b_1 and b_2 (instead of 3) obtaining the set of hard and soft clasuses:

$$\phi^{(1)} = \begin{Bmatrix} \{p, q\} \\ \{\neg p, \neg r\} \\ \{\neg q, \neg r\} \\ \sum_{i=1}^{2} b_i \leq 1 \end{Bmatrix} \qquad \psi^{(1)} = \begin{Bmatrix} 1{:}\{p, b_1\} \\ 1{:}\{r, b_2\} \end{Bmatrix}$$

The pair $\phi^{(1)} \cup \psi^{(1)}$ is satisfiable with the same assignment as before, with the exception of the assignment to b_3.

Now let us generalize the above example. We first define the notion of *core*. Given a set ϕ of unsatisfiable clauses a core is a subset ϕ' of ϕ such $\phi' \setminus \{C\}$ for every $C \in \phi'$ is consistent. In other words, a core is a minimally inconsistent set of formulas. such that if we remore one formula from it we are left with a consistent set of formulas.

Core of a set of clauses (Exercize 6).

EXAMPLE 5.9. *The core sets of the set of clauses*

$$\{\{a, b\}, \{\neg a, \neg b\}, \{a\}, \{b\}, \{\neg a\}, \{\neg b\}\}$$

are the following:

$$\{\{a\}, \{\neg a\}\} \qquad \{\{b\}, \{\neg b\}\} \quad \{\{a, b\}, \{\neg a\}, \{\neg b\}\} \qquad \{\{\neg a, \neg b\}, \{a\}, \{b\}\}$$

Notice that all the above subsets of the given clauses are not satisfiable, but removing one clause from each of them results in a satisfiable set of clauses.

SAT solvers are such that, when they are called with a a set of clauses ϕ which are inconsistent, they return UNSAT, and in addition they return a core set, i.e. a minimally inconsistent subset of ϕ. The Fu and Malik MaxSAT algorithm exploit this feature of SAT solver in order to solve the MaxSAT problem.

The Fu and Malik algorithm for MaxSAT, originally proposed by Fu and Malik 2006, uses the SAT subrutine; when it is called with a set of clauses ϕ, it returns a pair x, y where $x = SAT$ and $y = \mathcal{I}$ if ϕ is satisfiable and $\mathcal{I}$ is an interpretation of ϕ. or the pair $x = UNSAT$ and ϕ' is a core set. In the following, we consider the special case in which the weights of the soft clauses are equal to 1. Extensions of the algorithm for more general formulations of MaxSAT can be found in Manquinho, Marques-Silva, and Planes 2009.

Algorithm 7 Fu and Malik MaxSAT algorithm

Require: ϕ set of hard clauses; ψ set of soft clauses with weight = 1
1: $x, y \leftarrow \text{SAT}(\phi)$;
2: **if** $x = \text{UNSAT}$ **then**
3: **return** ∞, None
4: **end if**
5: $cost \leftarrow 0$
6: **while** True **do**
7: $x, y \leftarrow \text{SAT}(\phi \cup \psi)$
8: **if** $x = \text{SAT}$ **then**
9: **return** $cort, y$
10: **else** ▷ $x = \text{UNSAT}$, y is a core for $\phi \cup \psi$
11: $B = \emptyset$
12: **for** $C \in y \cap \psi$ **do** ▷ C is a soft clause that appears in the core y
13: $b \leftarrow$ new propositional variable
14: $C \leftarrow C \cup \{b\}$
15: $B \leftarrow B \cup \{b\}$
16: **end for**
17: **end if**
18: $\phi \leftarrow \phi \cup \{\sum_{b \in B} b \leq 1\}$
19: $cost \leftarrow cost + 1$
20: **end while**

EXAMPLE 5.10. *Consider the set of hard and soft clauses ϕ and ψ.*

$$\phi = \{\neg x_1, \neg x_2\}, \{\neg x_1, \neg x_3\}, \{\neg x_1, \neg x_4\}, \{\neg x_2, \neg x_3\}, \{\neg x_2, \neg x_4\}, \{\neg x_3, \neg x_4\}$$
$$\psi = 1 : \{x_1\},\ 1 : \{x_2\},\ 1 : \{x_3\},\ 1 : \{x_4\}$$

- *Since $\phi \cup \psi$ is not satisfiable, then the call to $SAT(\phi \cup \psi)$ returns $x = UNSAT$ and y a core set of $\phi \cup \psi$;*
- *Suppose that the returned core set is the following:*

$$y = \{\neg x_1, \neg x_2\}, \{x_1\}, \{x_2\}$$

  there are other core sets, but the result of the algorithm is independend from which core is returned by the sat solver.
- *The two waighted clauses $\{x_1\}$, $\{x_2\}$ are extended with two new propositions b_1 and b_2 respectively and the clause $b_1 + b_2 \leq 1$ (which is equivalent to $\{\neg b_1, \neg b_2\}$ is added.*
- *the cost is set to 1.*
- *Therefore we ontain the following new set of hard and soft clauses*

$$\phi = \{\neg x_1, \neg x_2\}, \{\neg x_1, \neg x_3\}, \{\neg x_1, \neg x_4\}, \{\neg x_2, \neg x_3\}, \{\neg x_2, \neg x_4\}, \{\neg x_3, \neg x_4\}, \{\neg b_1, \neg b_2\}$$
$$\psi = 1 : \{x_1, b_1\},\ 1 : \{x_2, b_2\},\ 1 : \{x_3\},\ 1 : \{x_4\}$$

- *The new set of clauses $\phi \cup \psi$ is also inconsistent, and a possible core is $\{\neg x_2, \neg x_4\}, \{x_3\}, \{x_4\}$.*
- *the cost is set to 2.*
- *We proceed as before obtaining the set of hard and soft clauses*

$$\phi = \{\neg x_1, \neg x_2\}, \{\neg x_1, \neg x_3\}, \{\neg x_1, \neg x_4\}, \{\neg x_2, \neg x_3\}, \{\neg x_2, \neg x_4\}, \{\neg x_3, \neg x_4\}, \{\neg b_1, \neg b_2\}, \{\neg b_3, \neg b_4\}$$
$$\psi = 1 : \{x_1, b_1\},\ 1 : \{x_2, b_2\},\ 1 : \{x_3, b_3\},\ 1 : \{x_4, b_4\}$$

- *the new set of clauses are not satisfiable and the only core is the set itself.*
- *the cost is set to 3;*
- *Every soft clause is therefore extended with a new variable. Since there are four soft clauses in the core, we introduce four new variables $b_5, \ldots, b_8$. And we also add the hard clause $b_5 + \cdots + b_8 \leq 1$ obtaining the following set of clauses:*

$$\phi = \{\neg x_1, \neg x_2\}, \{\neg x_1, \neg x_3\}, \{\neg x_1, \neg x_4\}, \{\neg x_2, \neg x_3\}, \{\neg x_2, \neg x_4\}, \{\neg x_3, \neg x_4\}, \{\neg b_1, \neg b_2\}, \{\neg b_3, \neg b_4\}$$
$$\psi = 1 : \{x_1, b_1, b_5\},\ 1 : \{x_2, b_2, b_6\},\ 1 : \{x_3, b_3, b_7\},\ 1 : \{x_4, b_4, b_8\}$$

  which is satisfiable.
- *Therefore the algorithm terminates.*

We have to prove that the algorithm terminates.

PROPOSITION 5.3. *The algorithm Fu and Malik terminates for every input of* ϕ, ψ

PROOF. to do □

3.3. Pseudo Boolean Optimizatiom. A Pseudo-Boolean function is a function $f0,1^n \to \mathbb{R}$. This type of function have beenn tudied since the '60s in operations research in integer linear programming Boros and Hammer 2002. One can consider also restricted versions, where f is represented in a polynomial of a given degree; f is represented as linear form, polinomial of degree 1;

A pseudo-boolean constraint is a constraint defined on a pseudo-boolean function f.

EXAMPLE 5.11. *Let* $f(x_1, \ldots, x_n)$ *be the linear pseudo-boolean function* $\sum_{i=1}^{n} x_i$. *An example of constraint on* f *is*

$$\sum_{i=1}^{n} x_i \leq k0$$

for some integer k, *which corresponds to the cardinality constraint "at most* k*" introduced in the previous chapters.*

DEFINITION 5.7 (Pseudo-Boolean optimization). *A pseudo-Boolean optimization is the problem of finding a boolean assignment to the variables* $x_1, \ldots, x_n$, *that satisfy the following:*

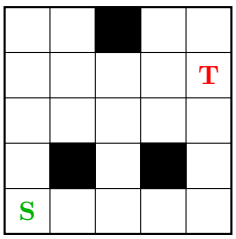
$$\begin{aligned} &\textit{Minimize} && \sum_{i=1}^{n} c_i x_i && \\ &\textit{Subject to} && \sum_{j=1}^{n} a_{ij} x_j \leq b_i && \forall i = 1, \ldots, m \end{aligned}$$

where $a_i, b_i, c_i \in \mathbb{Z}$.

[Section to be completed]

4. Solving problems with MaxSAT

4.1. Minimal path. Find shortest path in a grid with horizontal/vertical moves. Travel from **S** to **T** without enter in the black squares



$$p_{0,5,5} : \infty \tag{53}$$

$$\bigvee_{i=1}^{22} p_{i,2,5} : \infty \tag{54}$$

$$\bigwedge_{\substack{r,c=1 \\ (r,c)\neq(2,5)}}^{5} \left(p_{i,r,c} \rightarrow \bigvee_{\substack{r'c'=1 \\ |(r,c)-(r',c')|=1}}^{5} p_{i+1,r',c'} \right) : \infty \tag{55}$$

$$\bigwedge_{i=1}^{22} (\neg p_{i,1,3} \wedge \neg p_{i,4,1} \wedge \neg p_{i,4,4}) : \infty \tag{56}$$

$$\neg p_{i,r,c} : 1 \tag{57}$$

4.2. Optimal correlation clustering. The problem of optimal correlation clustering can be formulatecd as follows: Given a set of n points $V = \{v_1, \ldots, v_n\}$ and a symmetric similarity function $s : V \times V \rightarrow \{0, 1\}$ (such that $s(v_i, v_j) = 1$ (resp. 0) means that v_i is similar (resp. dissimilar) to v_j), the problem of *optimal correlation clustering* is the problem of partitioning V in a set of cluster $\mathbb{C} = C_1, \ldots, C_k$ for some (unknown) $k \geq 1$ such that the global correlation $G(\mathbb{C})$ is minimized:

$$G(\mathbb{C}) = \sum_{\substack{v_i \neq v_j \in V \\ cl(v_i)=cl(v_j)}} (1 - s(v_i, v_j)) + \sum_{\substack{v_i \neq v_j \in V \\ cl(v_i)\neq cl(v_j)}} s(v_i, v_j)$$

where $cl(v) = i$ means that $v \in C_i$.

A MaxSAT formulation of the optimal correlation clustering problem is based on a set of indicator variables x_{ij} , where $i < j$, with the interpretation that x_{ij} is true if points v_i and v_j are put in the same cluster. Using this variables we can formulate the following hard and soft clauses

Hard clauses:: for every $i < j < k$

$$\neg x_{ij} \vee \neg x_{jk} \vee x_{ik} \qquad\qquad \neg x_{ij} \vee \neg x_{ik} \vee x_{jk}$$

Soft clauses:: for every $i < j$

$$x_{ij} : 1 \qquad \text{If } s(v_i, v_j) = 1$$

$$\neg x_{ij} : 1 \qquad \text{If } s(v_i, v_j) = 0$$

The first hard clause states that, if v_i and v_j are put in the same cluster and v_j and v_k are also put in the same cluster then v_i and v_k must be put in the same cluster. This clause is the CNF form of transidivity: $x_{ij} \wedge x_{jk} \rightarrow x_{ik}$. The second hard clause has similar meaning; it corresponds to the euclidean property of a relation $x_{ij} \wedge x_{ik} \rightarrow x_{jk}$. Notice that clustering is a special type of equivalence relation and therefore it is both transitive and euclidean. We don't need symmetry and reflexivity because for $i < j$ x_{ji} and x_{ii} are not propositional variables. Soft clauses mimic what happens in the cost function $G(\mathbb{V})$. Notice that if v_i and v_j are symilar (i.e., $s(v_i, v_j) = 1$) and they are assigned with different clusters, i.e., x_{ij} is false, then the cost of the interpretation is increased by 1. This corresponds to the term $s(v_i, v_j)$ of the second summation of $G(\mathbb{V})$. Vice-versa, if two points v_i and v_j which are dissimilar (i.e., $v(v_i, v_j) = 0$) are put in the same cluster, i.e., x_{ij} is true then the cost of the interpretation will increase by 1, These corresponds to the term $1 - s(v_i, v_j)$ (which is equal to 1) of the first summations of $G(\mathbb{V})$

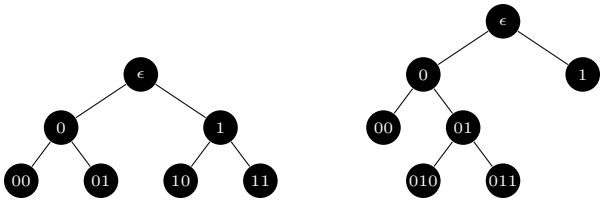


FIGURE 6. Two examples of binary trees with 7 nodes. Notice that the maximum depth of a binary tree with 7 nodes, is $\frac{7-1}{2} = 3$. Therefore we define B all the 0/1 strings of length less then or equal to 3. B indicates all the potential nodes of a binary tree.

Notice that the solution $\mathcal{I}$ of the MaxSAT problem does not provide directly the mapping of the v_i to the clusters. One has to derive such a clustering from the truth assignments of x_{ij}. This can be easily extracted by the following procedure; Let $\mathbb{C} = \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}$ be the initial clustering, and $x_{1,2}, \ldots, x_{1,n}, x_{2,3}, \ldots, x_{n-1,n}$ be an enumeration of the propositional variables, Then iterate on the elements of such a list, and at any step revise $\mathbb{C}$ as follows: If you are analyzing $x_{i,j}$ and $\mathcal{I}(x_{i,j}) = 1$, and $v_i \in C_q$ and $v_j \in C_r$ in the current clustering and $r \neq q$, then rmove C_q and C_r from $\mathbb{V}$ and add $C_q \cup C_r$.

5. MaxSAT for machine learning

5.1. Learning optimal decision tree. Let $\left\{\boldsymbol{x}^{(i)}, y^{(i)}\right\}_{i=1}^{d}$ be a dataset that we use to train a classifier for a class C. Suppose that $\boldsymbol{x}^{(i)}$ is a vector $(x_1, \ldots, x_k)$ of boolean features and that $y^{(i)}$ is equal to 0 if the i-th individual doesn't belong to C and 1 otherwise. Our goal is to construct a binary decision tree for the class C with at most n nodes. To cast this task in a MaxSat problem, we proceed in two steps, First, we formalize binary trees with at most n nodes in propositional logic. Successively, we associate to the internal node of the tree the corresponding features.

Let us start by formalizing binary trees in propositional logic. Before doing this let us see how a binary tree with n nodes looks like. Figure **??** shows two examples of binary trees with 7 nodes. To identify all possible nodes of a binary tree we use 0/1 strings, as follows. Suppose we want to consider a binary tree with at most n nodes. The maximum height of a binary tree with with n nodes. is $H = \frac{n-1}{2}$. Let B be the set $B = \bigcup_{h=0}^{H} \{0,1\}^h$. Intuitively B is all the potential nodes of a binary tree with n nodes. However we have to select only n elements of B. A binary tree can be seen as a non enpty subset of $T \subseteq B$ closed under substring. More formally $T \subseteq B$ is a tree iff

(1) $\epsilon \in T$ where ϵ is the empty string
(2) For all $\boldsymbol{b}i \in B$, $\boldsymbol{b}i \in T \Rightarrow \boldsymbol{b} \in T$

The above conditions can be easily formalized in propositional logic. Let $v_{\boldsymbol{b}i}$ for $\boldsymbol{b} \in B$ and $i \in \{0,1\}$ be a propositional variable indicating that the node $\boldsymbol{b}$ belongs to a binary tree T. For unifromity of formalization we add an extra depth step $H+1$ but we require that nodes with labels of length $H+1$ never belong to T.

The above conditions can be formalized as follows:

$$v_\epsilon \tag{58}$$

$$v_{\boldsymbol{b}i} \to v_{\boldsymbol{b}} \qquad \text{for } \boldsymbol{b} \in B \text{ and } i \in \{0,1\} \tag{59}$$

$$\neg v_{\boldsymbol{b}i} \qquad \text{for } \boldsymbol{b} \in B \text{ and } i \in \{0,1\} \text{ with } |\boldsymbol{b}| = H \tag{60}$$

Since we want a tree to be perfectly binary we have to guarantee that every node has either no children or two children. This can be formalized as:

$$v_{\boldsymbol{b}0} \leftrightarrow v_{\boldsymbol{b}1} \tag{61}$$

We can now pass to the second step in which we associate to each node $\boldsymbol{b} \in T$ one of the k features. To represent which feature is associated to which node we introduce the set of propositional variables $v_{\boldsymbol{b}}^f$ that codifies the fact that at node $\boldsymbol{b}$ we take the decision looking at feature f. We also want that only one feature is associated to internal nodes. This is formalizable by the axiom:

$$v_{\boldsymbol{b}} \wedge v_{\boldsymbol{b}i} \to \sum_f v_{\mathbf{b}}^f = 1 \tag{62}$$

Every propositional assignment that satisfies the above axioms identifies a unique binary decision tree with n nodes. We are only remained with the problem of formalizing in propositional logic how an item $\boldsymbol{x}^i$ is classified by such a binary tree. To this aim we introduce the propositional variabels $x_{\boldsymbol{b}}^{(i)}$ that indicates that the i-th item is classified in a subtree of the node $\boldsymbol{b}$. We can formalize the decision taken at each node of the tree, by the following set of formulas:

$x_\epsilon^{(i)}$	Every element is classified under the root node
$(x_{\boldsymbol{b}}^{(i)} \wedge v_{\boldsymbol{b}}^f) \leftrightarrow x_{\boldsymbol{b}x_f}^{(i)}$	If an is classified in $\boldsymbol{b}$ and the value of the feature associated to is i, then such item is classified under $\boldsymbol{b}i$

At this point you have to maximize the following measure

$$(x_{\boldsymbol{b}}^{(i)} \wedge \neg v_{\boldsymbol{b}i}) \wedge (C_{\boldsymbol{b}} \leftrightarrow y^{(i)})$$

5.2. Training Binary Neural Networks. [to be done]

6. Exercises

Exercise 76:

Given a list of numbers $a_1, \ldots, a_n$, formalize in propositional logic all the possible ways to split them into two sets. Then define a weight function that is maximal when the sums of the numbers in each set are as close as possible.

Exercise 77:

Define the weight functions that realizes the following total order on the interpretations of the propositinal variables $\mathcal{P} = \{A, B\}$

(1) $\{\} \prec \{A\} \prec \{B\} \prec \{A, B\}$
(2) $\{\} \prec \{B\} \prec \{A\} \prec \{A, B\}$
(3) $\{\} \prec \{A, B\} \prec \{A\} \prec \{B\}$
(4) $\{A, B\} \prec \{A\} \prec \{\} \prec \{B\}$
(5) $\{A\} \prec \{B\} \prec \{A, B\} \prec \{\}$

Exercise 78:

Prove the following facts:

(1) If $\models \phi \leftrightarrow \psi$, then $F \cup \{w : \phi\}$ is equivalent to $F \cup \{w : \psi\}$;
(2) $F \cup \{w_1 : \phi,\ w_2 : \phi\}$ is equivalent to $F \cup \{w_1 + w_2 : \phi\}$

Exercise 79:

Prove that $F \cup \{w : \phi \vee \psi\}$ is equivalent to $F \cup \{w : \phi \wedge \psi, w : \neg\phi \wedge \psi, w : \phi \wedge \neg\psi\}$

Exercise 80:

Suppose that $w < v$, Prove that $F \cup \{w : \phi, v : \neg\phi\}$ is equivalent to $F \cup \{v - w : \neg\psi\}$

Exercise 81:

Let $\mathcal{P}$ be a set of n propositinal variables, and let $w : 2^{\mathcal{P}} \to \mathbb{R}$ be a generic weight function. Define an algorithm that extract a set of weighted clauses F such that w_F is equivalent to w

Exercise 82:

Consider the set of strings that you can build with the letter A, B, C, and D. Define a propositional language such that everty interpretation correspond to a string and define a weight function that orders the strings (or equivalently the corresponding interpretations) lexicographically.

Solution Let us first define a set of propositional variables that we can use toi describe the finite strings composed of the letters A, B, C, and D. For every natural number $n \in \mathbb{N}$, we introduce the propositional variables A_n, B_n, C_n and D_n, where x_n for $x \in \{A, B, C, D\}$ means that the letter the n-th letter of the string is an x.

Then we have to add axioms that restricts the set of interpretations to those corresponding to strings.

(1) There must be at most one character at position n. This can be encoded either by a cardinality constraint AtMost$(1, \{A_n, B_n, C_n, D_n\})$ or by explicit representation:

$$\neg(A_n \wedge B_n) \quad \neg(A_n \wedge C_n) \quad \neg(A_n \wedge D_n) \quad \neg(B_n \wedge C_n) \quad \neg(B_n \wedge D_n) \tag{63}$$

(2) The second constraint states that, if in position n there is a character then there should be a character also in position $n-1$ i.e., we doin't have strings with "blanks" in the middle. This cen be formulated wiith:

$$(A_{n+1} \vee B_{n+1} \vee C_{n+1} \vee D_{n+1}) \rightarrow (A_n \vee B_n \vee C_n \vee D_n) \tag{64}$$

Notiche that every string $x_1x_2, \ldots, x_n$, with $x_i \in \{A, B, C, D\}$ can be mapped to an interpretation that makes x_i true and y_i false for eveyr $y \neq x$, and y_j false for every $y \in \{A, B, C, D\}$ and $j \geq n$. Therefore for instance, the string $ABAAC$ corresponds to the interpretation $\mathcal{I}_{ABAAC} = \{A_1, B_2, A_3, A_4, C_5\}$. Notice that the interpretation $\mathcal{I}_{x_1x_2\ldots x_n}$ corresponding to the string $x_1x_2\ldots x_n$ satisfies the axioms (63) and (64). Furtermore, every interpretation that satisfies these axioms corresponds to a (possibly infinite) string.

Let us now define a weight function that allow to order the interpretations that satisfies (63) and (64) lexicographically. I.e., $w(\mathcal{I}_{x_1x_2\ldots x_n\ldots}) < w(\mathcal{I}_{y_1y_2\ldots y_m\ldots})$ if and only if there is a k such that $x_k < y_k$ and $x_h = y_h$ for all $h < k$.

We define our weight function by associating a weight to each propositional variables and defining $w(\mathcal{I}) = \sum_{p \in \mathcal{I}} w(p)$

$$w(A_n) = 1 \cdot 10^{-n} \quad w(B_n) = 2 \cdot 10^{-n} \quad w(C_n) = 3 \cdot 10^{-n} \quad w(D_n) = 4 \cdot 10^{-n}$$

For instance we have the following weights:

$$\begin{aligned} w(\mathcal{I}_A) &= 0.1 \\ w(\mathcal{I}_{AB}) &= 0.1 + 0.02 = 0.12 \\ w(\mathcal{I}_B) &= 0.2 \\ w(\mathcal{I}_{BA}) &= 0.2 + 0.01 = 0.21 \\ w(\mathcal{I}_C) &= 0.3 \\ w(\mathcal{I}_D) &= 0.4 \\ w(\mathcal{I}_{DDDDDDDD\ldots}) &= 0.4 + 0.04 + 0.004 + \ldots = 0.\bar{4} \end{aligned}$$

□

Exercise 83:
Find all the cores of the following set of clauses.

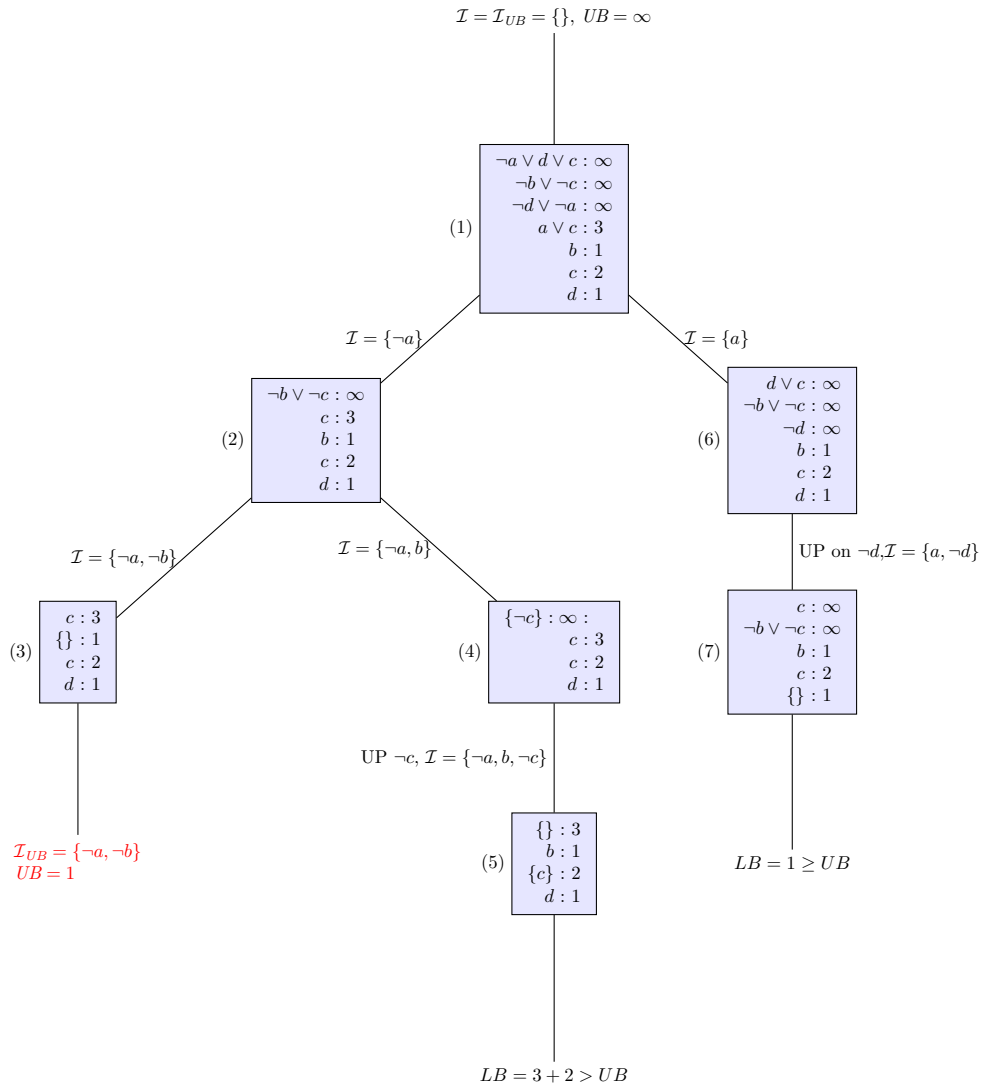
(1) $\{a, b, c\}, \{a, b, d\}, \{\neg a\}, \{\neg b\}, \{\neg c\}, \{\neg d\}$;
(2) $\{\neg a, b\}, \{\neg b, c\}, \{\neg c, d\}, \{\neg d, e\}, \{a, \neg b, \neg d\}$;
(3) $\{a\}, \{\neg a\}, \{b\}, \{\neg b\}, \{\neg a, b\}, \{a, \neg b\}$;
(4) $\{\neg a, b, c\}, \{\neg b, \neg d\}, \{\neg c, \neg d\}, \{a, d\}$;
(5) $\{x_i \mid i \in I\}$, $\{x_j \mid j \in J\}$, for some $I, J \subseteq \{1, \ldots, n\}$, with $I \cap J = \emptyset$ and the set of clauses $\{\neg x_i, \neg x_j\}$ for every $i < j \in \{1, \ldots, n\}$.

Exercise 84:

Use B&B algorithm to solve the following maxSat problem

$$(\neg a \vee d \vee c : \infty) \quad (a \vee c : 3) \quad (c : 2)$$
$$(\neg b \vee \neg c : \infty) \quad (b : 1) \quad (d : 1)$$
$$(\neg d \vee \neg a : \infty)$$

Solution



□

Exercise 85:

In the Fu and Malik algorithm for MaxSat, explain what is a minimally inconsistent set of clauses. And provide an example of four clauses, which has a minimally inconsistent set of three clauses.

Solution Given a set of clauses $S = \{C_1, C_2, \ldots, C_n\}$ a minimally inconsistent subset of clauses of S is any subset S' of S such that S' is inconsistent, and for all $C \in S'$, $S' \setminus \{C\}$ is consistent.

For instance consider the set of clauses

$$S = \{\{A, B\}, \{\neg A, \neg B\}, \{A, C\}, \{B, C\}, \{\neg A, \neg C\}, \{\neg B, \neg C\}, \{C\}\}$$

Then the subset

$$S' = \{\{A, B\}, \{\neg A, \neg C\}, \{\neg B, \neg C\}, \{C\}\}$$

is a minimally inconsistent subset of S. Indeed if we remove from S' any clause we can find always an interpretation. Indeed notice that

$$\begin{aligned}
S' \setminus \{\{A, B\}\} &= \{\{\neg A, \neg C\}, \{\neg B, \neg C\}, \{C\}\} && \text{Is satisfied by } A = F, B = F, C = T \\
S' \setminus \{\{\neg A, \neg C\}\} &= \{\{A, B\}, \{\neg B, \neg C\}, \{C\}\} && \text{Is satisfied by } A = T, B = F, C = T \\
S' \setminus \{\{\neg B, \neg C\}\} &= \{\{A, B\}, \{\neg A, \neg C\}, \{C\}\} && \text{Is satisfied by } A = F, B = T, C = T \\
S' \setminus \{\{C\}\} &= \{\{A, B\}, \{\neg A, \neg C\}, \{\neg B, \neg C\}\} && \text{is satisfied by } A = T, B = T, C = F
\end{aligned}$$

□

Exercise 86:

Consider the following set of formulas

weight : formula
$\infty : \phi_1 = A \leftrightarrow (X \wedge Y \wedge Z)$
$\infty : \phi_2 = B \leftrightarrow (Y \wedge T \wedge W)$
$\infty : \phi_3 = C \leftrightarrow (T \wedge V)$
$\infty : \phi_4 = \neg A \wedge \neg B \wedge \neg C$
$1 : X$
$2 : Y$
$3 : Z$
$4 : W$
$5 : V$
$6 : T$

Find an assignment that minimizes the cost. Remember that the cost of an assignment is the sum of the weights of the clauses that are not satisfied by the assignment

Solution

var	A	B	C	X	Y	Z	W	V	T	ϕ_1	ϕ_2	ϕ_3	ϕ_4
$\mathcal{I}(var)$	0	0	0	0	1	1	1	1	0	1	1	1	1
$cost$	0	0	0	1	0	0	0	0	6	0	0	0	0

$cost(\mathcal{I}) = 1 + 6 = 7$ A different model with the same cost is:

var	A	B	C	X	Y	Z	W	V	T	ϕ_1	ϕ_2	ϕ_3	ϕ_4
$\mathcal{I}'(var)$	0	0	0	1	0	1	1	0	1	1	1	1	1
$cost$	0	0	0	0	2	0	0	5	0	0	0	0	0

$cost(\mathcal{I}') = 2 + 3 = 7$ □ **Exercise 87:**

Suppose that you have to place m queens in an $n \times n$ chess board, with $m \leq n^2$. Every queen i can be placed in $p(i) = (x_i, y_i)$ with $1 \leq x_i, y_i \leq n$, so that it cannot

be "eaten" by any other queen. Encode the problem of finding the configuration that maximizes the total distance between the items, i.e.,

$$\sum_{1 \leq i,j \leq m} (x_i - x_j)^2 + (y_i - y_j)^2$$

.

Exercise 88:

Thransform the following MaxSat problem in and Integer Programming problem

$$\begin{aligned} \infty &: a \leftrightarrow \neg b \\ 2 &: a \wedge \neg b \rightarrow r \\ 3 &: b \wedge \neg a \rightarrow r \end{aligned}$$

Solution Since we have two weighted formulas we introduce two new propositional variables b_1 and b_2 and transform the clauses in the following integer constraints:

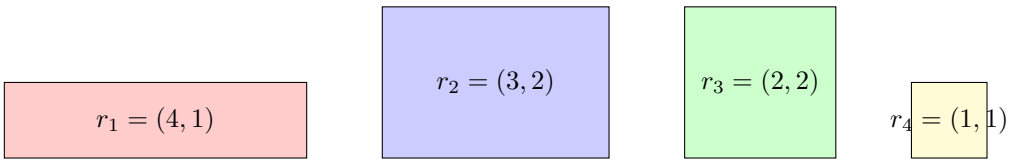
$$\begin{aligned} 1 \leq a \leq 1 \\ 1 \leq b \leq 1 \\ 1 \leq r \leq 1 \\ a + b = 1 \\ 0 \leq b_1 \leq 1 \\ 0 \leq b_2 \leq 1 \\ b_1 + (1 - a) + b + r \geq 1 \\ b2 + (1 - b) + a + r \geq 1 \end{aligned}$$

with the cost function: $2b_1 + 3b_2$. □

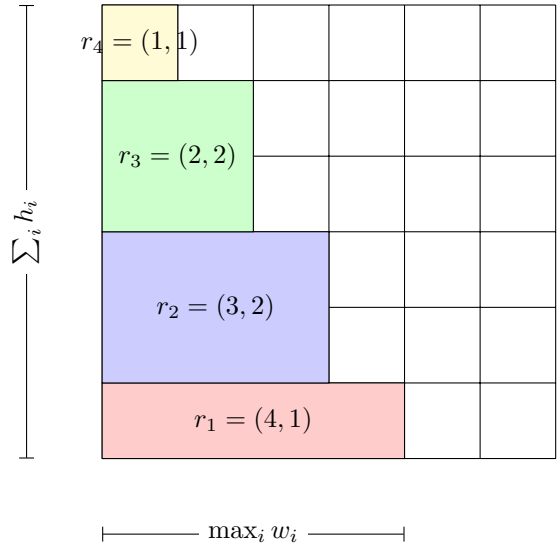
Exercise 89:

Suppose that you hae R rectangles $r_1, \ldots, r_R$, where each rectangle r_i has dimensions (w_i, h_i), with w_i and h_i natural numbers. Suppose that you have to arrange them inside an $n \times n$ square so that they don't overlap. Provide a MaxSat formulation of the problem of funding the smallest n for which theis is possible.

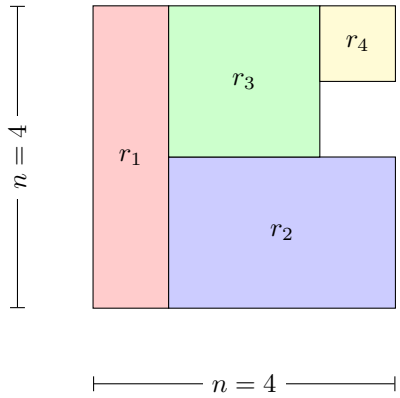
Solution To explain the general solution, let us consider an example with the following four rectangles:



Without loss of generality we can assume that $w_i \leq h_i$. Let $N = \max(\sum_i h_i, \max_j w_j)$. Notice that it is always possible to arrange the rectangles in an $N \times N$ square by stacking them, as shown in the following picture.



However this is not the optimal placement. An optimal placement would be, for instance, the following:



Let us formulate the problem to find such an optimal placement as a MaxSat problem. n We first provide the set of hard constraints that must be satisfied by every solution of the problem (not only the optimal ones). We use the propositions $H(i, j, k)$ and $V(i, j, k)$ to state that the i-th rectangle has been positioned horizontally or vertically respectivelly, in the position j, k where the position refers to the bottom left corner of the rectangle.

For instance the positioning of the first picture is the following:

$$H(1, 0, 0),\ H(2, 0, 1),\ H(3, 0, 3),\ H(4, 0, 5)$$

while that of the second picture is the following:

$$V(1, 0, 0),\ H(2, 1, 0),\ H(3, 1, 2),\ H(4, 3, 3)$$

Let us formulate the hard constraints:

(1) Every rectangle should be positioned either vertically or horizontally inside the $N \times N$ square. For every $i \in \{1, \ldots, R\}$ we add

$$\bigvee_{j=0}^{N-w_i} \bigvee_{k=0}^{N-h_i} H(i,j,k) \veebar V(i,k,j)$$

where $\veebar$ is the disjunctive or.

(2) Rectangles cannot overlap. If a rectangle is positioned in j, k then the other rectangles cannot be positioned in the slots occupied by the rectangle.

$$H(i,j,k) \rightarrow \bigwedge_{j'=0}^{w_i-1} \bigwedge_{k'=0}^{h_i-1} \bigwedge_{i' \neq i} \neg H(i', j+j', k+k') \wedge \neg V(i', j+j', k+k')$$

$$V(i,j,k) \rightarrow \bigwedge_{j'=0}^{h_i-1} \bigwedge_{k'=0}^{w_i-1} \bigwedge_{i' \neq i} \neg H(i', j+j', k+k') \wedge \neg V(i', j+j', k+k')$$

We then have to add some minimization criteria. The main idea is that if the top right corner of a rectangle covers the position j, k you will pay a cost of $(R+1)^{\max(j,k)}$. This can be obtained by adding the following weighted formula, one for every rectangle

$$H(i,j,k) : (R+1)^{\max(j+w_i, k+h_i)}$$
$$V(i,j,k) : (R+1)^{\max(j+h_i, k+w_i)}$$

Why we choose such a cost function. Because the cost of placing one single rectangle outside the $n \times n$ square will be larger than the cost of putting all the rectangles inside the $n \times n$ square. Indeed, if all the rectangles are inside the $n \times n$ square, the maximal cost that you will pay is

$$R(R+1)^n$$

If instead you put one single rectangle which goes out the $n \times n$ square you will pay a cost larger than

$$(R+1)^{n+1}$$

which it is larger than $R(R+1)^n$. This will force the system to search the minimal n. □

Exercise 90:
Prove that the solutions to the max sat problem

$$MaxSAT((A:1),(B:1),(C:1),(D:\infty))$$

when $\{A, B, D\}$ is not satisfiable, are equal to the solutions of the following reduced max sat problem

$$MaxSAT((A \vee a:1),(B \vee b:1),(C:1),(D:\infty),(a+b=1:\infty))$$

Notice that A, B, C, D are clauses that might contain common propositional variables.

Solution If the set $\{A, B, D\}$ is not satisfiable then the solution of the first MaxSat problem should falsify at least one of the soft clauses. A, B. In the second problem, the solution will make one of the proposition a and b. In the first case

the MaxSat problems become $(B, 1), (C, 1), (D; \infty)$ in the second case it becomes $(A, 1), (C, 1), (D; \infty)$. Solving the two problems is the same as solving the problem where A or B are falsified. $\square$

CHAPTER 6

Model counting

1. Introduction

A propositional formula ϕ partition the set of interpretations into two disjoint subsets. those that satisfy ϕ and those that do not.

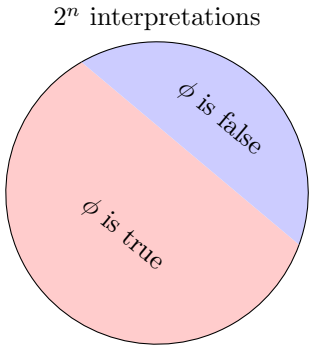


FIGURE 1. Visualisation of Model counting problem. The circle contains all the interpretations of the propositional variables of ϕ, which are 2^n. Model counting has to count the size of the red area.

Propositional model counting or #SAT Gomes, Sabharwal, and Selman 2009 is the problem of computing the number of models of a given propositional formula, i.e., the number of distinct truth assignments to propositional variables of a formula ϕ which satisfy the formula. Let us define the problem more precisely.

DEFINITION 6.1 (Model counting problem). *For a propositional formula ϕ let $\mathcal{P}(\phi)$ be the set of propositional variables occurring in ϕ. the* model counting of ϕ $\#\text{SAT}(\phi)$ *is the problem of finding the number of truth assignments to the propositional variables of ϕ that satisfies ϕ.*

$$\#\text{SAT}(\phi) = |\{\mathcal{I} : \mathcal{P}(\phi) \to \{0,1\} \mid \mathcal{I} \models \phi\}|$$

EXAMPLE 6.1. *A naive method to solve the model counting problem is via truth tables. For instance let us compute the model counting of the following formulas: $p \wedge q$, $p \vee q$, $p \to q$, $p \equiv q$, $\neg p$ and $p \wedge \neg p$*

p	q	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$	$\neg p$	$p \wedge \neg p$	$p \vee \neg p$
1	1	1 1 1	1 1 1	1 1 1	1 1 1	0 1	1 0 0 1	1 1 0 1
1	0	1 0 0	1 1 0	1 0 0	1 0 0	0 1	1 0 0 1	1 1 0 1
0	1	0 0 1	0 1 1	0 1 1	0 0 1	1 0	0 0 1 0	0 1 1 0
0	0	0 0 0	0 0 0	0 1 0	0 1 0	1 0	0 0 1 0	0 1 1 0
#SAT		1	3	3	2	2	0	4

Notice that in the above truth table the first four formulas contains all the propositional variables of the truth table, p and q in this case. while the last three formulas contains only one variables of the two. If we had computed model counting of the last three formulas separately, we would have obtained a different result:

p	$\neg p$	$p \wedge \neg p$	$p \vee \neg p$
1	0 1	1 0 0 1	1 1 0 1
0	1 0	0 0 1 0	0 1 1 0
	1	0	2

The difference is due to the fact that, in the first truth table, we compute model counting in the context *of the language of the propositional variables p and q, even if the formulas contain only one propositional variable (p in this case). This implies that we consider four possible truth assignments. In the second truth table, instead, we computee model counting in the context of the langauge that contains only one single variable, i.e., the one that appears in the formulas; in this case the number of interpretations are only two.*

The previous example highlights the fact that, to be precise, #SAT should also take also an extra parameter that is the language (set of propositional variables) of the interpretations. For this reason, when necessary we make explicit the langaue in which we compute the model counting, by writing $\#\text{SAT}(\phi, \mathcal{P})$, where $\mathcal{P}$ is a set of propositional variables such that $\mathcal{P}(\phi) \subseteq \mathcal{P}$. When we write the simpler notation $\#\text{SAT}(\phi)$ we actually mean $\#\text{SAT}(\phi, \mathcal{P}(\phi))$.

An alternative and equivalent formulation of the model counting problem, which will be useful when we want to extend the problem to *weighted model counting* is the following:

$$\#\text{SAT}(\phi, \mathcal{P}) = \sum_{\mathcal{I}:\mathcal{P}\rightarrow\{0,1\}} \mathcal{I}(\phi) \tag{65}$$

where $\mathcal{I}(\phi) = 1$ if $\mathcal{I} \models \phi$ and 0 otherwise.

The[1] model counting problem presents fascinating challenges for practitioners and poses several new research questions. Effcient algorithms for this problem will have a significant impact on many application areas that are inherently beyond SAT ('beyond' under standard complexity theoretic assumptions), such as probabilistic reasoning Chavira and Darwiche 2008a; Holtzen, Van den Broeck, and Millstein 2020 For example, various probabilistic inference problems, such as Bayesian net reasoning, can be effectively translated into model counting problems.Another application is in the study of hard combinatorial problems, such as combinatorial designs, where the number of solutions provides further insights into the problem. Even finding a single solution can be a challenge for such problems; counting the number of solutions is much harder. Not surprisingly, the largest formulas we

[1]This paragraph is an excerpt of the introductory section of Gomes, Sabharwal, and Selman 2009

can solve for the model counting problem with state-of-the-art model counters are orders of magnitude smaller than the formulas we can solve with the best SAT solvers. Generally speaking, current exact counting methods can tackle problems with a couple of hundred variables, while approximate counting methods push this to around 1,000 variables.

A rough intuition on the relation among model counting and probabilistic reasoning is as follows. If we consider an assignment to a set of variables as the outcome of an experiment and the formula ϕ a measure on this outcome, which is 1 if the outocme satisfy ϕ and 0 otherwise, then the higher $\#\text{SAT}(\phi)$ the higher the probability of observing a 1 in the measure.

As mentioned before the solution of the model counting problem is a very complex task and therefore one could also consider more efficient algirhtms that that does not guarante an exact solution, but provides an approximated one. Therefore, we will divide practical model counting techniques into two main categories: *exact counting* and *approximate counting*. Within exact counting, we will distinguish between methods based on DPLL-style exhaustive search, and those based on *knowledge compilation* or conversion of the formula into certain normal forms for which model counting is efficient (polinomial) Within approximate counting, we will distinguish between methods that provide fast estimates without any guarantees and methods that provide lower or upper bounds with a correctness guarantee.

2. Basic properties of model counting

As shown by the introductory example, the model counting of the same formula depends from the set of propositional variables that we consider (which should include those of the formula itself), Such a dependency is clarified in the following property:

PROPOSITION 6.1. *If ϕ is a propositinal formula that contains propositional variables in $\mathcal{P}$, i.e., $\mathcal{P}(\phi) \subseteq \mathcal{P}$, then $\#\text{SAT}(\phi, \mathcal{P}) = \#\text{SAT}(\phi) \cdot 2^{|\mathcal{P} \setminus \mathcal{P}(\phi)|}$.*

PROOF. Every interpretation $\mathcal{I}$ in $\mathcal{P}(\phi)$ that satisfies ϕ can be extended to an interpretation $\mathcal{I}'$ in the language of $\mathcal{P}$ by assigning any value in $\{0, 1\}$ to the variables in $\mathcal{P} \setminus \mathcal{P}(\phi)$. This implies that there are $2^{|\mathcal{P} \setminus \mathcal{P}(\phi)|}$ estensions. Since the truth value of ϕ is independent from the assignment to the variables not in $\mathcal{P}(\phi)$, we have that $\mathcal{I} \models \phi$ if and only if $\mathcal{I}' \models \phi$. Therefore for every models of ϕ in $\mathcal{P}(\phi)$ we have $2^{|\mathcal{P} \setminus \mathcal{P}(\phi)|}$ distinct models of ϕ in $\mathcal{P}$. Furthermore, every interpretation $\mathcal{I}'$ in $\mathcal{P}$ that satisfies ϕ, can be restricted to an interpretation $\mathcal{I}$ in $\mathcal{P}(\phi)$ by dropping the assignments to $\mathcal{P} \setminus \mathcal{P}(\phi)$. This guarantees that $\#\text{SAT}(\phi) \cdot 2^{|\mathcal{P} \setminus \mathcal{P}(\phi)|} \leq \#\text{SAT}(\phi, \mathcal{P})$. To show that $\#\text{SAT}(\phi) \cdot 2^{|\mathcal{P} \setminus \mathcal{P}(\phi)|} \geq \#\text{SAT}(\phi, \mathcal{P})$, notice that any pair of distinct interpretations $\mathcal{I}$ and $\mathcal{J}$ in $\mathcal{P}(\phi)$ which are extended into $\mathcal{I}'$ and $\mathcal{J}'$ interpretations of $\mathcal{P}$, will be such that $\mathcal{I}'$ is different from $\mathcal{J}'$. □

The previous proposition states that the model counting of a fomrula ϕ can be obtained multiplying the model counting of the formula w.r.t., the set of variables it contains times the number of assignments to the variables not containing in ϕ.

Other important properties of #SAT are the following:

PROPOSITION 6.2. *(1) If ϕ is valid, $\#\text{SAT}(\phi)$ is equal to $2^{|\mathcal{P}(\phi)|}$*
(2) If ϕ is unsatisfiable $\#\text{SAT}(\phi)$ is equal to 0
(3) $\#\text{SAT}(\neg\phi) = 2^{|\mathcal{P}(\phi)|} - \#\text{SAT}(\phi)$

(4) *If* $\phi \models \psi$ *then* $\#\text{SAT}(\phi, \mathcal{P}) \leq \#\text{SAT}(\psi, \mathcal{P})$*, where* $\mathcal{P}$ *is the set of propositional variables of* ϕ *and* ψ*.*
(5) *if* ϕ *is equivalent to* ψ*, then* $\#\text{SAT}(\phi, \mathcal{P}) = \#\text{SAT}(\psi, \mathcal{P})$ *and (by Property 6.1)* $\#\text{SAT}(\phi) \cdot 2^{\mathcal{P} \setminus \mathcal{P}(\psi)} = \#\text{SAT}(\phi) \cdot 2^{\mathcal{P} \setminus \mathcal{P}(\phi)}$

PROOF. (1) If ϕ is valied then every truth assignment of the propositional variables in ϕ will satisty ϕ. Since there are $2^{\mathcal{P}(\phi)}$ interpretation then $\#\text{SAT}(\phi) = 2^{\mathcal{P}(\phi)}$.
(2) If ϕ is unsatisfiable then ϕ is not satisfied by any interpretation and therefore $\#\text{SAT}(\phi) = 0$;
(3) Notice that $\mathcal{I} \models \neg\phi$ is and only if $\mathcal{I} \not\models \phi$. Since the total number of interpretations of the language of ϕ is $2^{\mathcal{P}(\phi)}$, we have that $\#\text{SAT}(\neg\phi) = 2^{\mathcal{P}(\phi)} - \#\text{SAT}(\phi)$.
(4) If $\phi \models \phi$ then every interpretation in the language of ϕ and ψ that satisfies ϕ also satisfies ψ.

□

A second set of properties concerns the relationship between the model counting of a formula and the model counting of its direct sub-formulas. We concentrate on the propositional connectives $\neg$, $\wedge$ and $\vee$ since, all the other connectives can be rewritten in ters of these three connectives.

2.1. #sat of negation. Since we have the low of excluded middle, i.e. every assignment $\mathcal{I}$ is such that either $\mathcal{I} \models \phi$ or $\mathcal{I} \models \neg\phi$, to count the models of $\neg\phi$, we can subtract the number of models of ϕ from the total set of assignments to the variables of ϕ which is $2^{|\mathcal{P}(\phi)|}$.

PROPOSITION 6.3. $\#\text{SAT}(\neg\phi) = 2^n - \#\text{SAT}(\phi)$ *where* n *is the number of propositinal variables that appears in* ϕ*.*

2.2. #sat of conjunction. In this section we consider how we can count the models of $\phi \wedge \phi$ by separately counting the models of ϕ and ψ or some derived (and simpler) formulas. If ϕ and ψ do not share propositional variables, then then the assignment to the propositional variables of ϕ does not interfere with the assignment to the propositional variables of ϕ. Therefore an assignment that satisfies $\phi \wedge \psi$ can be obtained by selecting any pair composed of a models of ϕ and a model of ψ. And therefore the number of models n of $\phi \wedge \psi$ is the product of the models of ϕ and the models ψ.

PROPOSITION 6.4. *If* ϕ *and* ψ *do not share propositional variables, then* $\#\text{SAT}(\phi \wedge \psi) = \#\text{SAT}(\phi) \cdot \#\text{SAT}(\psi)$*.*

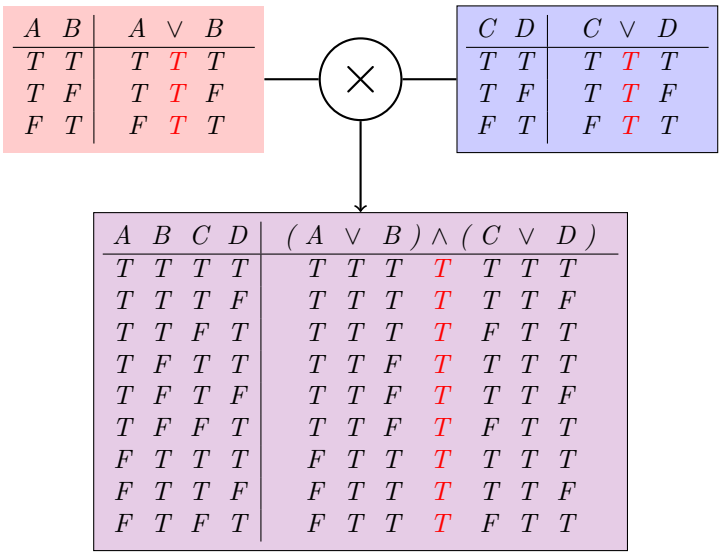
PROOF. Let $n = \#\text{SAT}(\phi)$ and $m = \#\text{SAT}(\psi)$. For every pair $\mathcal{I}$ and $\mathcal{J}$ of models of ϕ and ψ respectively we can define the assignment $\mathcal{I} \otimes \mathcal{J}$ on the propositinal variables of $\phi \wedge \psi$ defined as

$$\mathcal{I} \otimes \mathcal{J}(p) = \begin{cases} \mathcal{I}(p) & \text{if } p \in \mathcal{P}(\phi) \\ \mathcal{J}(p) & \text{if } p \in \mathcal{P}(\psi) \end{cases}$$

Since there is no overlap between the variables of ϕ and ψ the definition of $\mathcal{I} \otimes \mathcal{J}$ is well founded, Furthermore we have that $\mathcal{I} \otimes \mathcal{J} \models \phi \wedge \psi$ if and only if $\mathcal{I} \models \phi$ and $\mathcal{J} \models \psi$. Viceversa any model $\mathcal{I}$ of $\phi \wedge \psi$ can be decomposed of a model of ϕ and

a model of ψ by restricting it to the propositional variables that appear in the two formulas. Therefore the number of models of $\phi \wedge \psi$ is equal to $n \times m$. □

EXAMPLE 6.2. *Let us consider the formula* $(A \vee B) \wedge (C \vee D)$ *In the following picture we show how the assignments that satisfy this formula can be obtained by the composition of the of any models of* $(A \vee B)$ *and a model of* $(C \vee D)$.



What happens if ϕ and ψ contains common variables? We can combine a model of ϕ with a model of ψ only if the two models agree on the assignment of the shared propositional variables. Therefore for every assignment of the propositional variables we can combine the models that of ϕ and ψ that agree with this assignment. This is stated in the following proposition.

PROPOSITION 6.5. *If* $\mathcal{Q} = \mathcal{P}(\phi) \cap \mathcal{P}(\psi)$ *is the set of propositional variables that appears in both* ϕ *and* ψ*, then* $\#\text{SAT}(\phi \wedge \psi) = \sum_{\mathcal{I}:\mathcal{Q}\to\{0,1\}} \#\text{SAT}(\phi|_{\mathcal{I}}) \cdot \#\text{SAT}(\psi|_{\mathcal{I}})$ *where* $\phi|_{\mathcal{I}}$ *is the formula obtained by replacing* p *with* $\top$ *in* ϕ *if* $\mathcal{I}(p) = 1$ *and with* $\bot$ *if* $\mathcal{I}(p) = 0$.

Clearly Proposition 6.4 is a special case of Proposition 6.5.

EXAMPLE 6.3. *Let us compute the number of models of* $(p \vee q) \wedge (\neg q \vee r)$.

$$\begin{aligned}\#\text{SAT}((p \vee q) \wedge (\neg q \vee r)) &= \#\text{SAT}((\top \vee q) \wedge (\neg\top \vee r)) + \#\text{SAT}((\bot \vee q) \wedge (\neg\bot \vee r)) \\ &= \#\text{SAT}(\top \vee q) \cdot \#\text{SAT}(\neg\top \vee r) + \#\text{SAT}(\bot \vee q) \cdot \#\text{SAT}(\neg\bot \vee r)\end{aligned}$$

Notice that if ϕ and ψ share k propositional variables, the summations over all the possible interpretations of the k shared variable will contains 2^k addends.

2.3. #sat of disjunction. Let us now see how we can count the models of $\phi \vee \psi$. We know that a $\mathcal{I}$ is a model of $\phi \vee \psi$ if it is a model of ϕ or it is a model of ψ, So a first attempt would be to sum the models of ϕ and the models of ψ. However, in this way we counting twice the interpretations that satisfy both ϕ and ψ Therefore to fix this we have to subtract the models of $\phi \wedge \psi$.

PROPOSITION 6.6. *Let* $\mathcal{P}$ *be the set of propositional variables that occour in* $\phi \vee \psi$*. The following properties holds:*

(1) $\#\text{SAT}(\phi \vee \psi) = \#\text{SAT}(\phi, \mathcal{P}) + \#\text{SAT}(\psi, \mathcal{P}) - \#\text{SAT}(\phi \wedge \psi)$;
(2) *If* $\phi \wedge \psi$ *is unsatisfiable then* $\#\text{SAT}(\phi \vee \psi) = \#\text{SAT}(\phi, \mathcal{P}) + \#\text{SAT}(\psi, \mathcal{P})$.
(3) *If* ϕ *and* ψ *contain the same set of propositional variables then* $\#\text{SAT}(\phi \vee \psi) = \#\text{SAT}(\phi) + \#\text{SAT}(\psi) - \#\text{SAT}(\phi \wedge \psi)$.
(4) *If* ϕ *and* ψ *contain the same set of propositional variables and* $\phi \wedge \psi$ *is unsatisfiable then* $\#\text{SAT}(\phi \vee \psi) = \#\text{SAT}(\phi) + \#\text{SAT}(\psi)$.

PROOF. We prove only property (1) since all the other properties are corollaries. The set models($\phi \vee \psi$) of the models of $\phi \vee \psi$ can be partitioned in three disjoint subsets models($\phi \wedge \neg\psi$), models($\neg\phi \wedge \psi$), and models($\phi \wedge \psi$) Which implies that

$$\#\text{SAT}(\phi \vee \psi) = |\text{models}(\phi \wedge \neg\psi)| + |\text{models}(\neg\phi \wedge \psi)| + |\text{models}(\phi \wedge \psi)| \tag{66}$$

The set of assignments to $\mathcal{P}$ that satisfy ϕ can be partitioned in the two subsets models($\phi \wedge \neg\psi$) and models($\phi \wedge \psi$). From which we have that

$$\#\text{SAT}(\phi, \mathcal{P}) = |\text{models}(\phi \wedge \neg\psi)| + |\text{models}(\phi \wedge \psi)|$$

Similarly, we have that

$$\#\text{SAT}(\psi, \mathcal{P}) = |\text{models}(\neg\phi \wedge \psi)| + |\text{models}(\phi \wedge \psi)| \tag{67}$$

And therefore,

$$\begin{aligned}&\#\text{SAT}(\phi, \mathcal{P}) + \#\text{SAT}(\psi, \mathcal{P})\\ &\quad = |\text{models}(\phi \wedge \neg\psi)| + |\text{models}(\neg\phi \wedge \psi)| + 2 \cdot |\text{models}(\phi \wedge \psi)|\end{aligned}$$

From which we have that

$$\begin{aligned}&\#\text{SAT}(\phi, \mathcal{P}) + \#\text{SAT}(\psi, \mathcal{P}) - \text{models}(\phi \wedge \psi)|\\ &\quad = |\text{models}(\phi \wedge \neg\psi)| + |\text{models}(\neg\phi \wedge \psi)| + |\text{models}(\phi \wedge \psi)|\end{aligned} \tag{68}$$

By combining (66) and (68), we obtain property (1). □

PROPOSITION 6.7. *If* p *is a propositional variable of* ϕ,, *then* $\#\text{SAT}(\phi) = \#\text{SAT}(\phi|_p) + \#\text{SAT}(\phi|_{\neg p})$

PROOF. The proposition is a direct consequence of property (4) of Proposition 6.6. Indeed $\phi|_p$ and $\phi|_{\neg p}$ contain the same set of propositional variables; furthermore, since $\phi|_p$ and $\phi|_{\neg p}$ are equivalent to $\phi \wedge p$ and $\phi \wedge \neg p$ respectively, there is no interpretation that satisfies both formulas. This implies that property (4) of Proposition± 6.6 is applicable. Notice that the fact that p occurs in ϕ is an essential assumption, otherwise $\phi|_p$ and $\phi_{\neg p}$ would be the same formula □

We have seen that a naïve method for counting the models of a propositional formula is by computing the whole truth table. Since the truth table for a formula ϕ with n propositional variables contains 2^n lines, this method is very costly as it takes exponential time on the number of propositional variables. There are two possible direction in construct more efficient model counting algorithms.

The first one, consists in exploiting the properties seen in the previous section in order to take shortcuts, parallelise, and decompose the #SAT problem. Following this direction, for instance the fact that $\#\text{SAT}(\phi \wedge \psi) = \#\text{SAT}(\phi) \cdot \#\text{SAT}(\psi)$ when ϕ and ψ do not share propositinal variables, can be used to reduce the complexity, form 2^n to $2^{\max(n_1, n_2)}$ where n_1 and n_2 is the number of propositional variables occurring in ϕ and ψ respectively. This direction falls under the name of "exact

model counting" as the algorithms are guaranteed to return the correct value for $\#\text{SAT}(\phi)$. Since, as we will see in the following, exact algorithms are anyway very complex, the second direction aims to develop efficient algorithm that return an approximation of $\#\text{SAT}(\phi)$. In this section we describe exact algorithms while an example of approximate algorithms is described in the next section.

3. DPLL-Based Model Counting

In counting models of a propositional formula ϕ, one can see that [2] the models of ϕ can be split in two disjoint subsets by selecting a propositional variable p and counting the models of $\phi \wedge p$ and $\phi \wedge \neg p$. Counting the models of $\phi \wedge p$, is the same as counting the model of $\phi|_p$ which is the formula obtained by replacing p with $\top$ in ϕ. Similarly for $\phi|_{\neg p}$, which is obtained from ϕ by replacing p with $\bot$. With this simple rule we reduce $\#\text{SAT}(\phi)$ to $\#\text{SAT}(\phi|_p) + \#\text{SAT}(\phi|_{\neg p})$. Furthermore it is possible that ϕ_p and $\phi_{\neg p}$ are equivalent to formulas in which many other propositional letters have been removed. For instance if $\phi = (p \vee q) \wedge (r \vee s)$ then $\phi|_p$ is equivalent to $r \wedge s$. Therefore, by property 5 of Proposition 6.2 we can compute $\#\text{SAT}(\phi)$ by summing $\#\text{SAT}(\text{simplify}(\phi|'_p)) \cdot 2^n$ and $\#\text{SAT}(\text{simplify}(\phi|_{\neg p})) \cdot 2^m$ where n and m are the number of propositional variables that has been eliminated by simplifying $\phi|_p$ and $\phi|_{\neg p}$ respectively.

The property just described constitute the base for algorithm CDP (Counting Decision Procedure), shown in Algorithm 8. In particular the CPD algorithms exploits the property that for every ϕ in CNF

$$\#\text{SAT}(\phi, \mathcal{P}) = \#\text{SAT}(\phi|_p, \mathcal{P} \setminus \{p\}) + \#\text{SAT}(\phi|_{\neg p}, \mathcal{P} \setminus \{p\})$$

Notice that $\#\text{SAT}(\phi, \mathcal{P}) = \#\text{SAT}(\phi) \cdot 2^m$ where $m = |\mathcal{P} \setminus \mathcal{P}(\phi)|$. We could therefore replace $\mathcal{P}$ with n where n is the cardinality of $\mathcal{P}$, Since, $\mathcal{P}(\phi) \subseteq \mathcal{P}$ whe have that $m = n \setminus |\mathcal{P}(\phi)|$. The above property therefore can be rephrased in

$$\#\text{SAT}(\phi, n) = \#\text{SAT}(\phi|_p, n-1\}) + \#\text{SAT}(\phi|_{\neg p}, n-1)$$

CPD(ϕ, n) computes $\#\text{SAT}(\phi, \mathcal{P})$ for a formula ϕ in conjunctive normal form w.r.t. a set $\mathcal{P}$ of n propositional variables.

EXAMPLE 6.4. *Let* $\phi = \{\{p, q\}, \{\neg r, \neg p\}\}$. *To compute* $\#\text{SAT}(\phi)$ *we call CDP*$(\phi, 3)$ *since* ϕ *contains* 3 *propositional variables. The execution of* $CP(\phi, 3)$ *is shown in Figure 2.*

3.1. Literal selection. As for the case of DPLL the choice of the propositional variable to expand (line 8 of algorithm 8) has great influence on the efficiency of the algorithm. According to the above analisys the choice of p will generates two subproblems of expected complexity of $T(m_1, n-1)$ and $T(m_2, n-1)$ where m_1 is the set of clauses in $\phi|_p$ and m_2 the number of clauses in
$phi|_{\neg p}$. TO maximize this reduction we should choose the split that minimize $\max(m_1, m_2)$ where

3.2. Caching. If a set of clauses ϕ is encountered more than one time in the CPD algorithm, it would clearly be beneficial to cash the result of the first computation of $\#\text{SAT}(\phi)$ and to be able to efficiently recognize it in the next steps and reuse previous results.

[2] See Birnbaum and Lozinskii 1999.

Algorithm 8 CPD(ϕ, n)

Require: ϕ a propositional formula in CNF
Require: n an integer larger han $|\mathcal{P}(\phi)|$
if $\phi = \{\}$ **then** ▷ ϕ is the empty set of clauses
  return 2^n
end if
if $\{\} \in \phi$ **then** ▷ ϕ contains an empty clause
  return 0
end if
if $\{l\} \in \phi$ **then** ▷ Unit propagation
  return CDP$(\phi|_l, n-1)$
else
  $p \leftarrow$ select a propositinal variable of ϕ
  return CDP$(\phi|_p, n-1)$ + CDP$(\phi|_{\neg p}, n-1)$
end if

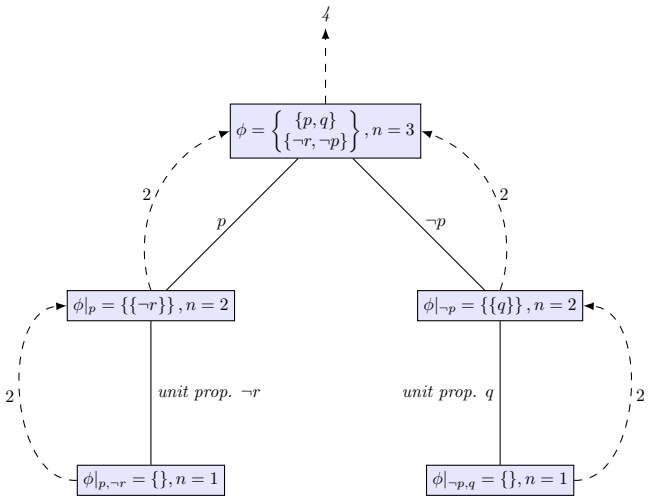


FIGURE 2. The execution tree of the function $CPD(\phi, 3)$. The total number of models returned by this procedure is 4, which is the sum of the two recursive calls.

3.2.1. *Complexity of CPD.* In the worse case the CDP decision procedure will generate all the possible assignments and therefore runs for 2^n steps. We way that CPD is *worse case exponential*. However we can provide a probabilistic estimation of the complexity of the CDP procedure. Birnbaum and Lozinskii 1999 provide such a result, and we report it in the following.

Assume that ϕ contains m clauses on n propositional variable. Assume also that the literals have the same probability p to appear in each clause ϕ. Let $T(m, n)$ denote the average running time of $CDP(\phi, n)$ We have the following theorem:

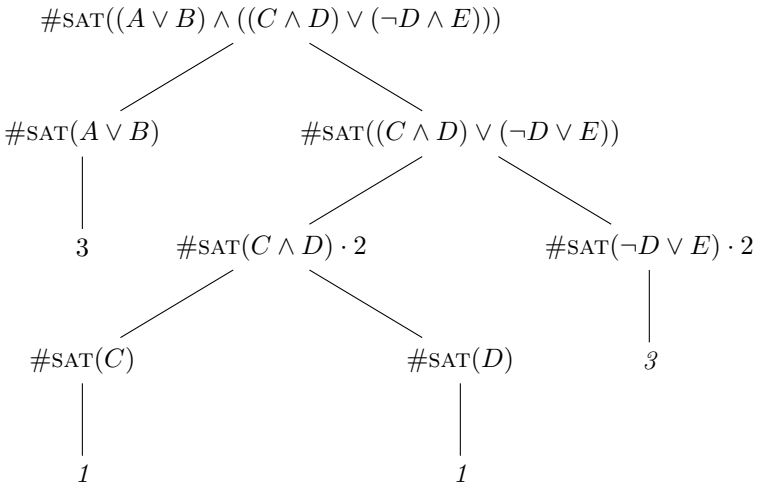
THEOREM 6.1. *$T(m, n) = O(m^d \cdot n)$ where $d = \lceil \frac{-1}{\log_2(1-p)} \rceil$*

The assumption of $p = \frac{1}{3}$ is commonly adopted in probabilistic analysis of algorithms handling CNF or DNF formulas Franco and Paull 1983 which means that for each variable, its occurrence in a clause with or without negation or non-occurrence, all have the same probability. Under this assumption we have that $T(m, n) = O(m^2 \cdot n)$.

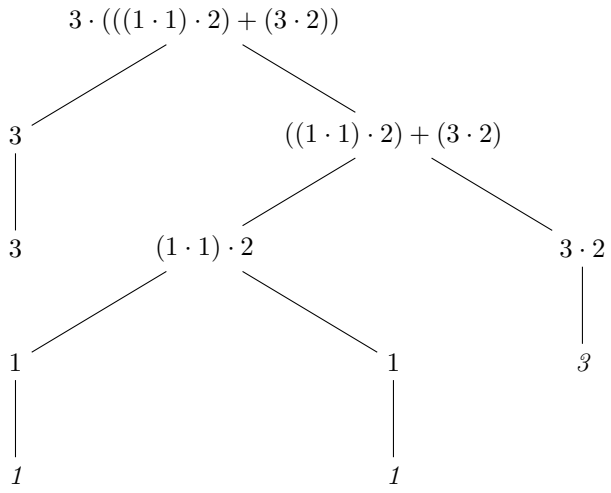
4. Model counting via Knowledge Compilation

The second method for exact model counting of a propositional formula is based on the transformation of ϕ is an equivalent formula for which model counting is decomposable using the properties introduced at the beginning of this chapter. Consider the following example:

EXAMPLE 6.5. *In computing* $\#\text{SAT}((A \vee B) \wedge ((C \wedge D) \vee (\neg D \wedge E)))$ *we can recursively apply the properties of model counting. This allow to decompose the problem of counting the models of a complex formula in the problem of counting the models of its subformulas and aggregating the results properly. The following tree shows how the* #SAT *ofr such a formula can be decomposed.*



The above tree show that to compute $\#\text{SAT}((A \vee B) \wedge ((C \wedge D) \vee (\neg D \wedge E)))$ *we compute* $\#\text{SAT}(A \vee B) = 3$, $\#\text{SAT}(C) = 1$, $\#\text{SAT}(D) = 1$, *and* $\#\text{SAT}(\neg D \vee E = 3$. *We then aggregates these results by seeing the decomposition tree as a algebraic expression:*

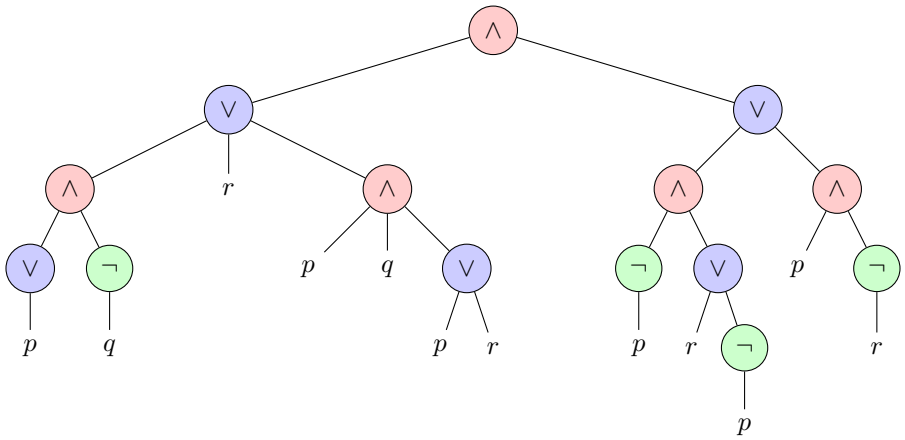


Knowledge compilation is the transformation of a formula ϕ in a form such that the decompositions shown in the previous example are possible. To this purpose we define a new normal form of propositional formulas and the rewriting rules that are necessary to transform every formula in such a form.

Before doing this let us recall what is Negated Normal Form (NNF).

Definition 6.2 (NNF). *A formula is in NNF (Negated Normal Form) if it contains only $\wedge$, $\vee$ and $\neg$ connectives, and the $\neg$ connective occours only in front of propositional variables*

Example 6.6. *In the following we show the parse tree of a formula in NNF. Notice that every branch contains an alternation of $\wedge$ and $\vee$ and ends with either an atom or the negation of the atom. aAA*



A formula is in NNF can be written as

$$\bigwedge_{i_1=1}^{n} \bigvee_{i_2=1}^{n_{i_1}} \bigwedge_{i_3=1}^{n_{i_1 i_2}} \cdots \bigvee_{i_k=1}^{n_{i_1 \ldots i_k}} l_{i_1,\ldots,i_k} \tag{69}$$

where $l_{i_1,\ldots,i_k}$ are literals and k is the depth (= maximum branch length) of the tree. CNF is a special kind of NNF with $k = 2$.

DEFINITION 6.3 (DNNF). *A propositional formula ϕ is in* Decomposable negation normal form (DNNF) *it is in Negated Normal Form (NNF) and for each conjunction $\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n$ $\mathcal{P}(\phi_i) \cap \mathcal{P}(\phi_j) = \emptyset$.*

Notice that in a DNNF formula in order to compute the model counting of a sub-formula that is a disjunction $\phi \wedge \psi$ we can apply Proposition 6.4 by computing the model counting of ϕ and ψ separately and then multiply the results.

Let us define a form that guarantee a similar property for disjunction.

DEFINITION 6.4 (d-DNNF). *A formula is in d-DNNF (deterministic DNNF) if it is in DNNF and for each disjunction $\phi_1 \vee \phi_2 \vee \cdots \vee \phi_n$ occurring in the formula, there is at most one $\mathcal{I}$ such that $\mathcal{I} \models \phi_i$ for every ϕ_i.*

As in the case of CNF we can define a set of rules that allows to tranform a formula ϕ in an equivalent formula in d-DNNF. The key tranformation to is called *Shannon's expansion*. The Shannon's expansion is a transformation that at the same time remove shared variables in a conjunction $\phi \wedge \psi$ and introduces a deterministic disjunction. The Shannon's expansion of ϕ is equal to

$$(p \wedge \phi|_p) \vee (\neg p \wedge \phi|_{\neg p}) \tag{70}$$

- Notice that, if ϕ is a conjunction $\phi_1 \wedge \phi_2$, then $p \wedge \phi|_p = p \wedge \phi_1|_p \wedge \phi_2|_p$, which is a conjunction of three formulas that do not share the variable p since p has been removed in $\phi|_p$ and $\phi_{\neg p}$. Similar observation holds for $\neg p \wedge \phi|_{\neg p}$. Furthermore, the disjunction introduced by the Shannon expansion is deterministic, since it is not possible that $p \wedge \phi|_p$ and $\neg p \wedge \phi|_{\neg p}$ are both true in an interpretation.
- If instead ϕ is the disjunction $\phi_1 \vee \phi_2$ and p occours either in ϕ_1 or in ϕ_2 or in both, then $p \wedge \phi|_p \vee \neg p \wedge \phi|_{\neg p}$ is equal to

  $$(p \wedge (\phi_1|_p \vee \phi_2|_p)) \vee (\neg p \wedge (\phi_1|_{\neg p} \vee \phi_2|_{\neg p}))$$

  is a deterministic disjunction, and the internal disjunctions $\phi_1|_p \vee \phi_2|_p$ and $\phi_1|_{\neg p} \vee \phi_2|_{\neg p}$ are less non-deterministic since they are interpreted on a smaller set of propositional variables.

DEFINITION 6.5 (Circuit for a d-DNNF formula). *Given a d-DNNF formula ϕ, the circuit for ϕ is the aritmetic expression circuit(ϕ) that computes $\#\text{SAT}(\phi)$ recursively defined as follows:*

- *if ϕ is a literal p or $\neg p$ then circuit(l) is 1*
- *if ϕ is $\phi_1 \wedge \phi_2$, then circuit(ϕ) = circuit(ϕ_1) $\cdot$ circuit(ϕ_2)*
- *if ϕ is $\phi_1 \vee \phi_2$, then circuit(ϕ) = circuit(ϕ_1) $\cdot 2^{n_2}$ + circuit(ϕ_2) $\cdot 2^{n_1}$, where n_1 (resp. n_2) is the number of propositional variables that occour in ϕ_1 (resp. ϕ_2) but not in ϕ_2 (resp. phi$_1$),*

EXAMPLE 6.7. *Let us transform $\phi = (A \vee B) \wedge (C \vee D) \wedge (\neg D \vee E)$ in d-DNNF. First notice that ϕ is the conjunction of two formulas ϕ_1 and ϕ_2 that do not share common variables. So the main conjunciton does not require any transformation, and we need to transform in d-DNNF the two sub-formula ϕ_1 and ϕ_2.*

$$\phi_1 = A \vee B$$
$$\phi_1 = (C \vee D) \wedge (\neg D \vee E)$$

ϕ_2 is not deterministic, since there is an interpretation that satisfies both disjunct A and B (i.e., the interpretation that makes both A and B true) so we have to apply

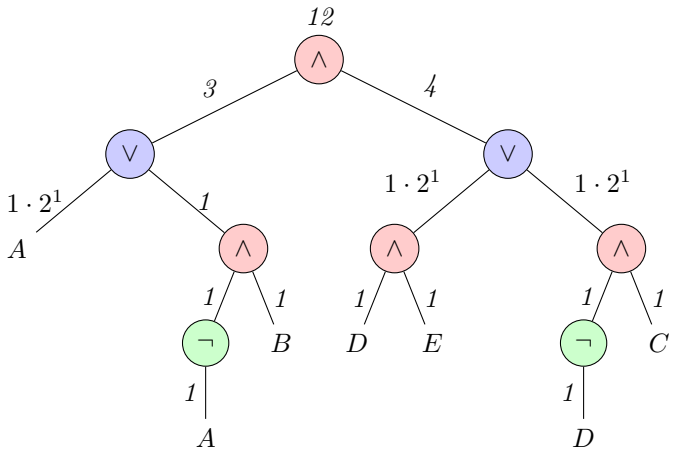


FIGURE 3. The formula tree of the d-DNNF formula $(A \vee (\neg A \wedge B)) \wedge ((D \wedge E) \vee (\neg D \wedge C))$. The numbers on the arcs represent the model counting of the corresponding subformula. To compute the model counting of the entire formula is is sufficient to start from the bottom, assigning a 1 to every proposition, and propagate up $\wedge$ nodes by multiplying the model counting of the subformulas (using property (6.4)) and propagating up $\vee$ nodes by summing the mc of the subformulas multiplied by 2^m where m is the number of variable occouring in the other subformula composing the or (property (2) of Proposition 6.6)

Shannon's expansion on some proposition of ϕ_1. Let us consider A. By Shannon's expansion we obtain

$$(A \wedge (\top \vee B)) \vee (\neg A \wedge (\bot \vee B))$$

and simplifying we obtain:

$$\phi_1' = A \vee (\neg A \wedge B)$$

Notice that ϕ_1' is deterministic, since every interpretation either falsify A or $\neg A$, with implies that there is no interpretation that simultaneously satisfies both the disjuncts. Furthermore every conjunction in ϕ_1' is such that the two conjuncts ($A \wedge \top$ and $\neg A \wedge B$) do not share propositional variables. So ϕ_1' is in d-DNNF.

Let us not tranform ϕ_2 in d-DNNF. Notice that ϕ_2 is the conjunciton of two formulas that share the propositonal variable D. Therefore we have to apply Shannon expansion on ϕ_2 w.r.t. the propositional variable D. We obvtain:

$$(D \wedge (C \vee \top) \wedge (\bot \vee E) \vee (\neg D \wedge (C \vee \bot) \wedge (\top \vee E))$$

which is equivalent to

$$\phi_2' = (D \wedge E) \vee (\neg D \wedge C)$$

Therefore the original formula ϕ is equivalent to $\phi' = \phi_1' \wedge \phi_2'$ which is in d-DNNF. The formula tree of ϕ' and how model counting can be obtained from the associated circuit is shown in Figure 3.

5. Approximate algorithm for model counting

In this section we describe one of the most basic algorithm for approximate model counting. The algorithm called ApproxCount has beed introduced in Wei and Selman 2005. The algorithm is based on a method called SampleSat for uniformly sampling models of a formula ϕ. Let us first introduce the SampleSat algorithm.

5.1. SampleSat. SampleSat algorithm is based on random walk strategies. It requires in input a formula ϕ in CNF, i.e., a set of clauses $\{C_1, \ldots, C_n\}$. Random walk (RW) strategy to search for a truth assignment that satisfies a formula ϕ starts from a random truth assignment. At every iteration the if the current assignment satisfies a formula then it is returned; otherwise one unsatisfied clause C_i is chosen uniformly at random from ϕ and a variable in the clause is chosen by some heuristic. The value of the variable is flipped. The algorithm repeats these steps until a satisfying assignment is reached.

Algorithm 9 SampleSat

```
1: I ← random assignments to the variabels of φ
2: while true do
3:     if I ⊨ φ then
4:         return I
5:     else
6:         C ← random clause C ∈ φ such that I ⊭ C
7:         p ← SelectVar(C);
8:         I(p) ← 1 − I(p)                    ▷ Flip the truth value of p in I
9:     end if
10: end while
```

The function SelectVar(C) select a literal from the clause C. A random selection of the literal when clauses are longer than 2 does not result in an unbias choice. It has been shown in Selman, H. A. Kautz, Cohen, et al. 1993 that using ϵ-greedy strategy provide a better behaviour (but other stratecies are also possible and could lead to better results).

For an assignment $\mathcal{I}$, the *break degree* of a propositional variable p is the number of clauses that are true in $\mathcal{I}$ and become false if we change the truth value of p.

$$break(p, \mathcal{I}) = |\{C' \in \phi \mid \mathcal{I} \models C' \text{ and } \mathcal{I}_{flip(p)} \not\models C'\}$$

The SelectVar(C) method proposed by Selman, H. A. Kautz, Cohen, et al. 1993 selects a variable with break degree equal to 0 if there exists one, otherwise it selects the variable of C with lower break degree with probability $1-\epsilon$ and a random literal with probability ϵ. (for some small ϵ). Though this strategy does not guarantee unbiased sampleing it is an improvement w.r.t., random sampling. The algorithm is shown in Algorithm 10. Advanced method are described in Selman, H. A. Kautz, Cohen, et al. 1993.

Let us now describe the ApproxCount algorithm, which is shown ,in Algorithm 11, The algorithm is based on the following intuition. Let S be a set of models of ϕ and S_p the subset of S that assigns p to true, I.e, $S_p = \{\mathcal{I} \in S \mid I(p) = True\}$.

Algorithm 10 SELECTVAR$(C, \mathcal{I})$

if there is a p in C with $break(p, \mathcal{I}) = 0$ **then**
    return p
else
    with probability $1 - \epsilon$ **return** $\text{argmin}_{p \in C}\, break(p, \mathcal{I})$
    with probability ϵ **return** a random propositional variable of C
end if

If S is a good sampling of the models of ϕ then we can estimate

$$\frac{\#\text{SAT}(\phi|_p)}{\#\text{SAT}(\phi)} \approx \frac{|S_p|}{|S|}$$

From which we have that

$$\#\text{SAT}(\phi) = \frac{|S|}{|S_p|} \cdot \#\text{SAT}(\phi_p)$$

Which means that we can appriximate the model counting of a formula ϕ by multiplying the model counting of a simpler formula ϕ_p by the factor $\frac{|S|}{|S_p|}$. The same resoning can be done by considering the set $S_{\neg p} = \{I \in S \mid \mathcal{I} \models \neg p\}$, obtaining a second approximation of $\#\text{SAT}(\phi)$:

$$\#\text{SAT}(\phi) = \frac{|S|}{|S_{\neg p}|} \cdot \#\text{SAT}(\phi_{\neg p})$$

For every literal l we call $\frac{|S|}{|S_l|}$ the *muliplying factor* associated to the literal l. Let $l_1, \ldots, l_k$ be a set of non contraddictory literals, then by iterating the above argument we obtain the approximation

$$\#\text{SAT}(\phi) = \#\text{SAT}(\phi|_{l_1,\ldots,l_k}) \prod_{i=1}^{k} \frac{|S^{(i)}|}{|S^{(i)}_{l_i}|} \tag{71}$$

Where $S^{(i)}$ is the assignments that satisfies $\phi|_{l_1\ldots,L_{i-1}}$. Notice that at eery iteration we sample different models. The APPROXCOUNT algorithm uses this method to approximate $\#\text{SAT}(\phi)$. Finally let us see a small example on how APPROXCOUNT.

EXAMPLE 6.8. *Let us ppply* APPROXCOUNT *to estimate the number of models of the following formulas and compare the the exact solution.*

$$(p \wedge r) \vee (q \wedge \neg s)$$

Run APPROXCOUNT *with input* $\phi = (p \wedge r) \vee (q \wedge \neg s)$ *and* $k = 2$

(1) Repeatetly all SAMPLESAT *to obtain a set* S *of* n *models for* ϕ*. You can decide the number* n *of models that you want. The larger* n *the better the approximation.*
(2) suppose that $S = \{1010, 1100, 0110, 1110, 0100\}$*, (an interpretation on* p, q, r, s *is represented with the 4-bit value* $\mathcal{I}(p)\mathcal{I}(q)\mathcal{I}(r)\mathcal{I}(s)$*);*
(3) select a propositional variable $x \in \{p, q, r, s\}$ *to split in order to minimize* $(|S_x| - |S_{\neg x}|)^2$*. The proposiitional variables that minimize this difference are* p *and* r*. Suppose that we select* p
(4) $S_p = \{1010, 1100, 1110\}$
(5) $S_{\neg p} = \{0110, 0100\}$

Algorithm 11 ApproxCount ϕ in CNF

$m \leftarrow 1$ ▷ m is called the multiplier factor
while $|props(\phi)| \geq k$ **do** ▷ When ϕ is small we apply exact method
  $S \leftarrow$ SampleSat(ϕ) n times ▷ Select n models for ϕ the larger the better
  $p \leftarrow$ Select a propositional variable of ϕ ▷ Heuristic: Choose p that maximizes $(|S_p| - |S_{\neg p}|)^2$
  $S_p \leftarrow \{\mathcal{I} \in S \mid \mathcal{I}(p) = True\}$
  $S_{\neg p} \leftarrow \{\mathcal{I} \in S \mid \mathcal{I}(p) = False\}$
  if $|S_p| \geq |S_{\neg p}|$ **then**
    $m \leftarrow m \cdot \frac{|S|}{|S_p|}$
    $\phi \leftarrow \phi|_p$
  else
    $m \leftarrow m \cdot \frac{|S|}{|S_{\neg p}|}$
    $\phi \leftarrow \phi|_{\neg p}$
  end if
end while
return $m \cdot \#\text{SAT}(\phi)$ ▷ $\#\text{SAT}(\phi)$ is computed with an exact method

(6) *since* $|S_P| > |S_{\neg p}|$, *we choose to multiplication fuctor* $\frac{|S|}{|S_p|} = \frac{5}{3}$.
(7) $m = \frac{5}{3}$.
(8) $\phi = \phi|_p = (r \vee (q \wedge \neg s)$
(9) *repeatetly call* SampleSat *in order to sample a new set of models for* ϕ
(10) *suppose that* $S = \{100, 010, 110\}$
(11) *the propositional variable that minimies* $(|S_x| - |S_{\neg x}|)^2$ *are* q *and* r. *Suppose that we select* q
(12) $S_q = \{100, 110\}$
(13) $S_{\neg q} = \{010\}$
(14) *since* $|S_q| > |S_{\neg q}|$ *we select the multiplicative factor* $\frac{|S|}{|S_q|} = \frac{3}{2}$
(15) $m = m \cdot \frac{3}{2} = \frac{5}{2}$
(16) $\phi = \phi|_q = (r \vee \neg s)$
(17) *since* ϕ *contains* $2 \leq k$ *variables, we exit the wile and return* $m \cdot \#\text{SAT}(r \vee \neg s) = \frac{5}{2} \cdot 3 = 7.5$

To understand quality of the result and measure the error let us compare the output of ApproxCount *and the exact value of* #sat *computed via truth table:*

p	q	r	s	$(p \wedge r) \vee (q \wedge \neg s)$
1	1	1	1	1
1	1	1	0	1
1	1	0	1	0
1	1	0	0	1
1	0	1	1	1
1	0	1	0	1
1	0	0	1	0
1	0	0	0	0
0	1	1	1	0
0	1	1	0	1
0	1	0	1	0
0	1	0	0	1
0	0	1	1	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	0
				7

The error is around 7%*.*

5.2. Graph Neural Networks for Propositional Model Counting. Saveri and Bortolussi 2022 TO DO

6. Exercises

Exercise 91:

Explain the difference between $\#\text{SAT}(\phi)$ and $\#\text{SAT}(\phi, \mathcal{P})$ when $\mathcal{P}$ is a set of propositional variables larger than the set of propositional variables that appears in ϕ.

Exercise 92:

Find an example in which $\phi \models \psi$ but $\#\text{SAT}(\phi) > \#\text{SAT}(\psi)$.

Exercise 93:

Prove that if ϕ contains p, then $\phi|_p$ is equivalent to $\phi \wedge p$. Provide a counteresample of this property when p does not occour in ϕ.

Exercise 94:

Given a set of propositional variables $p_1, \ldots, p_n$, describe a method to produce a formula that has exactly k models for every $k \leq 2^n$.

Solution Let $\boldsymbol{b} = b_1 \ldots b_n$ be the bitwise representation of any the integer h between 0 and $2^n - 1$ (included). Let us define

$$\phi_h = \bigwedge_{\substack{i=1 \\ b_i=1}}^{n} p_i \wedge \bigwedge_{\substack{i=1 \\ b_i=0}}^{n} \neg p_i$$

Notice that the formula ϕ_h is satisfied by a single interpretation, i.e., the interpretation $\mathcal{I}$, such that $\mathcal{I}(p_i) = True$ if and only if $b_i = 1$. Furthermore, for every $g \neq h$, we have that there is no model that satisfies both ϕ_g and ϕ_h. We can therefore define the formula

$$\phi_{\leq k} = \bigvee_{0=1}^{k-1} \phi_h$$

which will be satisfied by exaclty k models. □

Exercise 95:

Use CPD to count the models of the formulas $(A \rightarrow C) \wedge (B \rightarrow C)$ and the formula $(A \vee B) \rightarrow C$. **Solution** The two formulas need to be tranformed in CNF.

They both are transformed in the following set of clauses:

$$(\neg A, B\}, \{\neg B, C\}$$

Let us run CPD on them.

CPD($\phi = \{\{\neg A, B\}, \{\neg B, C\}\}, n = 0$)
  unit propagation is not applicable since there are no unit clauses
  select the literal $\neg A$
  $\phi_{\neg A} = \{\{\neg B, C\}\}$
  CPD($\phi = \{\{\neg B, C\}\}, n = 1$)
    unit propagation is not applicable since there are no unit clauses
    select the literal $\neg B$
    $\phi_{\neg B} = \{\}$

CPD($\phi = \{\}, n = 2$)
return $2^{|\mathcal{P}|-n} = 2^{3-2} = 2$ ($\mathcal{P} = A, B, C\}$
select the literal B
$\phi_B = \{\{C\}\}$
CPD($\phi = \{\{C\}\}, n = 2$)
apply unit propagation obtaining $\{\}$ and $n = 3$
return $2^{|\mathcal{P}|-n} = 2^0 = 1$
return $2 + 1 = 3$
select the literal A
$\phi_{\neg A} = \{\{B\}, \{\neg B, C\}\}$
CPD($\phi = \{\{B\}\{\neg B, C\}\}, n = 1$)
apply unit propagation obtaining $\{\}$ and $n = 3$
return $2^{|\mathcal{P}|-n} = 2^0 = 1$
return $3 + 1 = 4$

□

Exercise 96:

Use CPD to count the models of the formula $(A \to B) \wedge (B \to C) \wedge (C \to D)$

Solution CPD require the formula in CNF. So we first need to transform $(A \to B) \wedge (B \to C) \wedge (C \to D)$ in CNF, which results in the forllowing clauses:

$$\{\neg A, B\}, \{\neg B, C\}, \{\neg C, D\}$$

CPD($\phi = \{\{\neg A, B\}, \{\neg B, C\}, \{\neg C, D\}\}, n = 0$)
unit propagation is not applicable since there are no unit clauses
select the literal $\neg A$
$\phi_{\neg A} = \{\{\neg B, C\}, \{\neg C, D\}\}$
CPD($\phi = \{\{\neg B, C\}, \{\neg C, D\}\}, n = 1$)
unit propagation is not applicable since there are no unit clauses
select the literal $\neg B$
$\phi_{\neg B} = \{\{\neg C, D\}\}$
CPD($\phi = \{\{\neg C, D\}\}, n = 2$)
unit propagation is not applicable since there are no unit clauses
select the literal $\neg C$
$\phi_{\neg C} = \{\}$
CPD($\phi = \{\}, n = 3$)
return $2^{|\mathcal{P}|-n} = 2^1 = 2$ ($\mathcal{P} = A, B, C, D\}$
select the literal C
$\phi_C = \{D\}$
CPD($\phi = \{D\}, n = 3$)
apply unit propagation which returns $\phi = \{\}$ and $n = 4$
return $2^{|\mathcal{P}|-n} = 2^0 = 1$
return $2 + 1 = 3$
select the literal B
$\phi_B = \{\{C\}, \{\neg C, D\}\}$
CPD($\phi = \{\{C\}, \{\neg C, D\}\}, n = 2$)
apply unit propagation obtaining $\{\}$ and $n = 4$
return $2^{|\mathcal{P}|-n} = 2^0 = 1$

return $3+1=4$
select the literal A
$\phi_A = \{\{B\}\{\neg B, C\}, \{\neg C, D\}\}$
CPD($\phi = \{\{B\}\{\neg B, C\}, \{\neg C, D\}\}, n = 1$)
apply unit propagation obtaining $\{\}$ and $n = 4$
return $2^{|\mathcal{P}|-n} = 2^0 = 1$
return $4+1=5$

To check the correctness of the result let us compute the truth table explicitly:

A	B	C	D	(A	→	B)	∧	((B	→	C)	∧	(C	→	D))
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	0	1	0	0
1	1	0	1	1	1	1	0	1	0	0	0	0	1	1
1	1	0	0	1	1	1	0	1	0	0	0	0	1	0
1	0	1	1	1	0	0	0	0	1	1	1	1	1	1
1	0	1	0	1	0	0	0	0	1	1	0	1	0	0
1	0	0	1	1	0	0	0	0	1	0	1	0	1	1
1	0	0	0	1	0	0	0	0	1	0	1	0	1	0
0	1	1	1	0	1	1	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	0	1	1	1	0	1	0	0
0	1	0	1	0	1	1	0	1	0	0	0	0	1	1
0	1	0	0	0	1	1	0	1	0	0	0	0	1	0
0	0	1	1	0	1	0	1	0	1	1	1	1	1	1
0	0	1	0	0	1	0	0	0	1	1	0	1	0	0
0	0	0	1	0	1	0	1	0	1	0	1	0	1	1
0	0	0	0	0	1	0	1	0	1	0	1	0	1	0
8	8	8	8	8	12	8	**5**	8	12	8	8	8	12	8

□

Exercise 97:
Count the models that satisfy the formula $(A \to B \vee C) \wedge (C \to D \vee E)$

Solution The formula is a conjunction of two formulas that share the propositional C variable. We can therefore we can apply the Shannon reduction on C obtaining the formula:

$$C \wedge (A \to B \vee \top) \wedge (\top \to D \vee E) \vee$$
$$\neg C \wedge (A \to B \vee \bot) \wedge (\bot \to D \vee E)$$

which is equivalent to

$$C \wedge (A \to B \vee \top) \wedge (D \vee E) \vee$$
$$\neg C \wedge (A \to B) \wedge (\bot \to D \vee E)$$

Since the disjunction is deterministic (since only one of the disjunct holds), we can compute separately the number of models of the formulas and then sum up.

The number of models of $C \wedge (A \to B \vee \top) \wedge (D \vee E)$ is $1 \cdot 4 \cdot 3 = 12$ The number of models of $\neg C \wedge (A \to B) \wedge (\bot \to D \vee E)$ is $1 \cdot 3 \cdot 4 = 12$ In total we have $12 + 12 = 24$ models.

Alternatively one can fill the truth table

A	B	C	D	E	(A	→	(B	∨	C))	∧	(C	→	(D	∨	E))
T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
T	T	T	T	F	T	T	T	T	T	T	T	T	T	T	F
T	T	T	F	T	T	T	T	T	T	T	T	T	F	T	T
T	T	T	F	F	T	T	T	T	T	F	T	F	F	F	F
T	T	F	T	T	T	T	T	T	F	T	F	T	T	T	T
T	T	F	T	F	T	T	T	T	F	T	F	T	T	T	F
T	T	F	F	T	T	T	T	T	F	T	F	T	F	T	T
T	T	F	F	F	T	T	T	T	F	T	F	T	F	F	F
T	F	T	T	T	T	T	F	T	T	T	T	T	T	T	T
T	F	T	T	F	T	T	F	T	T	T	T	T	T	T	F
T	F	T	F	T	T	T	F	T	T	T	T	T	F	T	T
T	F	T	F	F	T	T	F	T	T	F	T	F	F	F	F
T	F	F	T	T	T	F	F	F	F	F	F	T	T	T	T
T	F	F	T	F	T	F	F	F	F	F	F	T	T	T	F
T	F	F	F	T	T	F	F	F	F	F	F	T	F	T	T
T	F	F	F	F	T	F	F	F	F	F	F	T	F	F	F
F	T	T	T	T	F	T	T	T	T	T	T	T	T	T	T
F	T	T	T	F	F	T	T	T	T	T	T	T	T	T	F
F	T	T	F	T	F	T	T	T	T	T	T	T	F	T	T
F	T	T	F	F	F	T	T	T	T	F	T	F	F	F	F
F	T	F	T	T	F	T	T	T	F	T	F	T	T	T	T
F	T	F	T	F	F	T	T	T	F	T	F	T	T	T	F
F	T	F	F	T	F	T	T	T	F	T	F	T	F	T	T
F	T	F	F	F	F	T	T	T	F	T	F	T	F	F	F
F	F	T	T	T	F	T	F	T	T	T	T	T	T	T	T
F	F	T	T	F	F	T	F	T	T	T	T	T	T	T	F
F	F	T	F	T	F	T	F	T	T	T	T	T	F	T	T
F	F	T	F	F	F	T	F	T	T	F	T	F	F	F	F
F	F	F	T	T	F	T	F	F	F	T	F	T	T	T	T
F	F	F	T	F	F	T	F	F	F	T	F	T	T	T	F
F	F	F	F	T	F	T	F	F	F	T	F	T	F	T	T
F	F	F	F	F	F	T	F	F	F	T	F	T	F	F	F

and counth the number of assignments that satisfy the formula. They are 24. □

Exercise 98:

Check if the following formulas are in d-DNNF form and if they are not explain why.

(1) $\neg(p \vee q) \vee (p \vee r)$
(2) $(p \wedge \neg q) \vee (q \wedge (r \vee (\neg r \wedge s)))$
(3) $(p \vee q) \wedge (r \vee (\neg r \wedge s))$
(4) $((p \wedge (\neg p \vee (p \wedge r))$

Solution

(1) $\neg(p \vee q) \vee (p \vee r)$ is not in d-DNNF because it is not in NNF since the negation occurs in front of a complex formula, while in NNF negation can occur only in front of atomic formulas.
(2) $(p \wedge \neg q) \vee (q \wedge (r \vee (\neg r \wedge s)))$ Is in d-DNNF.

(3) $(p \vee q) \wedge (r \vee (\neg r \wedge s))$ Is not in d-DNNF because $(p \vee q)$ is not deterministic since there is an interpretation that satisfies both p and q.
(4) $((p \wedge (\neg p \vee (p \wedge r))$ Is not in d-DNNF because it is not in DNNF since the sub formulas p and $(\neg p \vee (p \wedge r))$ contains both the atom p, and therefore the formula is not in DNNF.

□

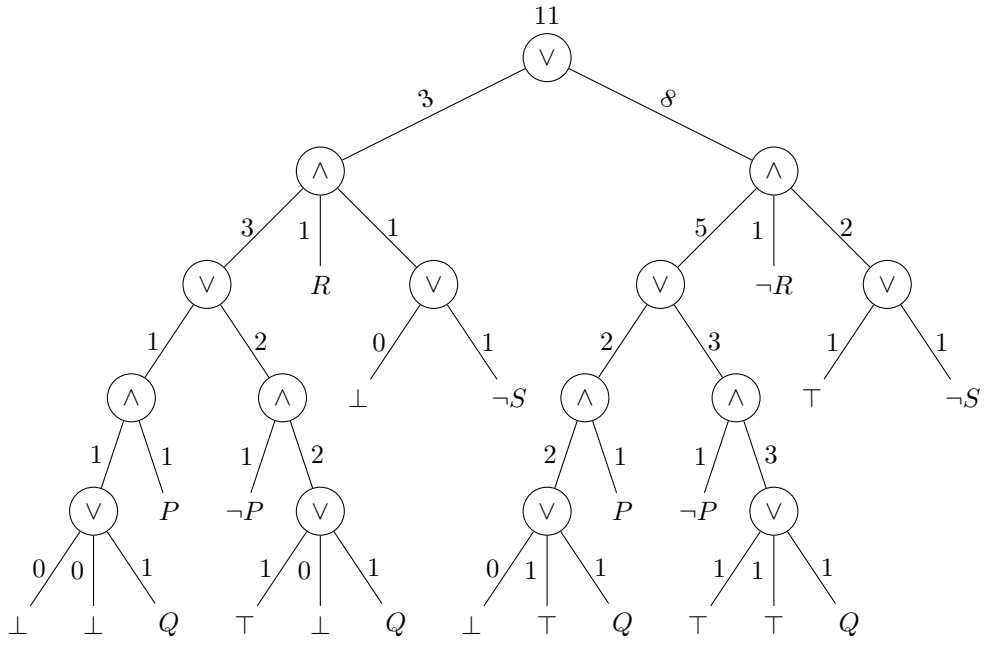
Exercise 99:
Transform the following formula in d-DNNF.

$$(P \vee Q) \wedge (R \vee S) \wedge (\neg S \vee T)$$

Exercise 100:
Compute $\#SAT((P \wedge R \rightarrow Q) \wedge (\neg R \vee \neg S))$ using knowledge compilation method.

Solution We first have to transform the formula in deterministic decomposable negated normal form (d-DNNF).

$$\begin{aligned}
(P \wedge R \rightarrow Q) \wedge (\neg R \vee \neg S) \equiv \\
(\neg P \vee \neg R \vee Q) \wedge (\neg R \vee \neg S) \equiv \\
((\neg P \vee \bot \vee Q) \wedge (\bot \vee \neg S) \wedge R) \vee ((\neg P \vee \bot R \vee Q) \wedge (\top \vee \neg S) \wedge \neg R) \equiv \\
(((\bot \vee \bot \vee Q) \wedge P) \vee ((\top \vee \bot \vee Q) \wedge \neg P) \wedge (\bot \vee \neg S) \wedge R) \vee \\
(((\bot \vee \top \vee Q) \wedge P) \vee ((\top \vee \top \vee Q) \wedge \neg P) \wedge (\top \vee \neg S) \wedge \neg R)
\end{aligned}$$



Let us verify the correctenss of the result by compiling the truth table

P	Q	R	S	((P	∧	R)	→	Q)	∧	(¬	R	∨	¬	S)
T	T	T	T	T	T	T	T	T	F	F	T	F	F	T
T	T	T	F	T	T	T	T	T	T	F	T	T	T	F
T	T	F	T	T	F	F	T	T	T	T	F	T	F	T
T	T	F	F	T	F	F	T	T	T	T	F	T	T	F
T	F	T	T	T	T	T	F	F	F	F	T	F	F	T
T	F	T	F	T	T	T	F	F	F	F	T	T	T	F
T	F	F	T	T	F	F	T	F	T	T	F	T	F	T
T	F	F	F	T	F	F	T	F	T	T	F	T	T	F
F	T	T	T	F	F	T	T	T	F	F	T	F	F	T
F	T	T	F	F	F	T	T	T	T	F	T	T	T	F
F	T	F	T	F	F	F	T	T	T	T	F	T	F	T
F	T	F	F	F	F	F	T	T	T	T	F	T	T	F
F	F	T	T	F	F	T	T	F	F	F	T	F	F	T
F	F	T	F	F	F	T	T	F	T	F	T	T	T	F
F	F	F	T	F	F	F	T	F	T	T	F	T	F	T
F	F	F	F	F	F	F	T	F	T	T	F	T	T	F

□

Exercise 101:

Codify in $\#SAT$ the problem of counting how many different k-tuples $(i_1, \ldots, i_5)$ of integers ≤ 8 such that $i_1 \leq \imath_2 \leq \cdots \leq i_5$.

Solution Let p_{ij}, for $1 \leq i \leq 5$ and $1 \leq j \leq 8$, be a propositional variable that stands for "there is a j in position i". The set of strings that satisfy the conditions of the exercise are those that satisfy the following formulas:

$$\bigwedge_{i=1}^{5} \bigvee_{j=1}^{8} p_{ij}$$ in every position there is at least one digit

$$\bigwedge_{i=1}^{5} \bigwedge_{j<k=1}^{8} \neg(p_{ij} \wedge p_{ik})$$ in every position there is at most one digit

$$\bigwedge_{i=1}^{4} \bigwedge_{j=1}^{8} \left(p_{ij} \rightarrow \bigvee_{k=j}^{8} p_{i+1,k} \right)$$ In the next position can occour only number larger or equal to the one present in the current position

□

Exercise 102:

Codify in $\#SAT$ the problem of counting how many different strings can be made by reordering the letters of the word "SUCCESS"?

Solution The reordering of the string "SUCCESS" is any string that contains three "S"s, two "C"s, one "U" and one "E". Let us introduce the following propositional variables

S_i	$1 \leq i \leq 7$	there is an "S" in position i
C_i	$1 \leq i \leq 7$	there is an "C" in position i
U_i	$1 \leq i \leq 7$	there is an "U" in position i
E_i	$1 \leq i \leq 7$	there is an "E" in position i

We have the following axioms:

$$\bigwedge_{i=1}^{7} S_i \vee C_i \vee U_i \vee E_i \qquad \text{In each position there is at least one letter}$$

$$\bigwedge_{i \neq j} \neg(U_i \wedge U_j) \qquad \text{At most one "U"}$$

$$\bigwedge_{i \neq j} \neg(E_i \wedge E_j) \qquad \text{At most one "E"}$$

$$\bigwedge_{i \neq j \neq k} \neg(C_i \wedge C_j \wedge C_k) \qquad \text{At most two "C"}$$

$$\bigwedge_{i \neq j \neq k \neq h} \neg(S_i \wedge S_j \wedge S_k \neg S_h) \qquad \text{At most tree "S"}$$

□

Exercise 103:

To buy a computer system, a customer can choose one of 4 monitors, one of 2 keyboards, one of 4 computers and one of 3 printers. But only certain combinations are allowed. Provide some example of how you can express constraints on possible combinations with propositional formulas, and codify the problem of determining the number of possible systems that a customer can choose from as model counting.

Exercise 104:

Suppose you have three coins: the faces of the first coin are black and white, the faces of the second coin are yellow and green, and the faces of the third coin are blue and red. In an experiment you toss the first coin; if you obtain a black you toss the second coin otherwise you toss the third coin. What are the number of possible outcomes? Encode the problem of counting the possible outcomes of this simple experiment in the problem of model counting a set of formulas

Exercise 105:

Transform the formula $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg D \vee E)$ in d-DDNF form, and use it for model counting. **Solution** Split in

(1) $(A \vee B) \wedge (\neg B \vee C)$
(2) $(\neg D \vee E)$

since they don't have common variables, and treat them separately

(1) let us consider $(A \vee B) \wedge (\neg B \vee C)$

$(A \vee B) \wedge (\neg B \vee C)$ Apply Shannon's expansion on B

$(B \wedge (A \vee \top) \wedge (\bot \vee C)) \vee (\neg B \wedge (A \vee \bot) \wedge (\top \vee C))$ in d-DDNF

The last formula has:

$$(1 \cdot (1+1) \cdot (0+1)) + (1 \cdot (1+0) \cdot (1+1) = 4$$

models

(2) Let us now consider $(\neg D \vee E)$

$\neg D \vee E$	Apply Shannon's expansion on D
$(D \wedge (\bot \vee E)) \vee (\neg D \wedge (\top \vee E))$	in d-DDNF

The last formula has:

$$(1 \cdot (0 + 1)) + (1 \cdot (1 + 1)) = 3$$

The total number of models is $4 \cdot 3 = 12$. □

Exercise 106:
Transform the following formula in d-DNNF.

$$(P \vee Q) \wedge (R \vee S) \wedge (\neg S \vee T)$$

Solution Notice that the formula is the conjunction of two formulas that do not have common propositional variables. We can therefore proceed to transform each subformula in d-DNNF.

$(P \vee Q)$ is transformed (via Shannon's expansion) in

$$P \wedge (\top \vee Q) \vee \neg P \wedge (\bot \vee Q)$$

$(R \vee S) \wedge (\neg S \vee T)$ instead is the conjunciton of two formulas that have one propositional variable in common (i.e., S). Therefore we have to consider the two cases in which S is true and S is false. We therefore rewrite the formula by using the Shannon's expansion

$$(S \wedge (R \vee \top) \wedge (\bot \vee T)) \vee (\neg S \wedge (R \vee \bot) \wedge (\top \vee T))$$

THe result d-DNNF transofmraiton is the conjunction of the two resutls:, i.e.,

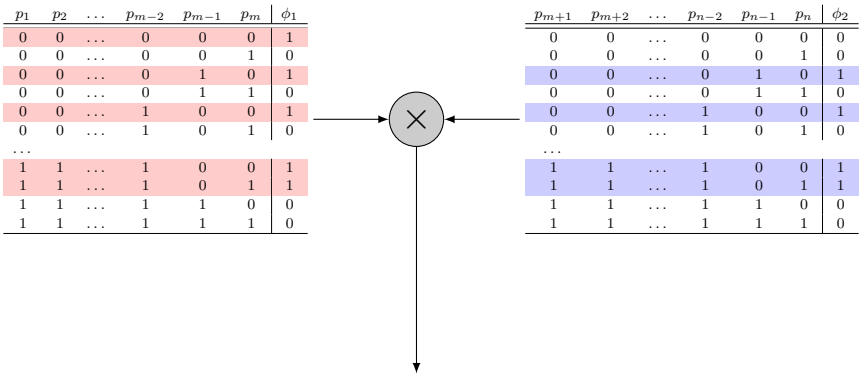
$$P \wedge (\top \vee Q) \vee \neg P \wedge (\bot \vee Q) \wedge$$
$$(S \wedge (R \vee \top) \wedge (\bot \vee T)) \vee (\neg S \wedge (R \vee \bot) \wedge (\top \vee T))$$

□

Exercise 107:
Explain in at most 10 lines, why if ϕ_1 and ϕ_2 don't share propositional variabels then $\#SAT(\phi_1 \wedge \phi_2) = \#SAT(\phi_1) \cdot \#SAT(\phi_2)$

Solution A model $\mathcal{I}$ of $\phi_1 \wedge \phi_2$ is an assignment of all the variables in ϕ_1 and ϕ_2 that satisfies both ϕ_1 and $\phi_{@}$. Since ϕ_1 and ϕ_2 do not have variables in common, we can split the assignment $\mathcal{I}$ in two assignments $\mathcal{I}_1$ and $\mathcal{I}_2$ where $\mathcal{I}_1$ assigns the variables in ϕ_1 and $\mathcal{I}_2$ the variables in ϕ_2. Furthermore, we have that $\mathcal{I}_1 \models \phi_1$ and $\mathcal{I}_2 \models \phi_2$. Therefore all the models of $\phi_1 \wedge \phi_2$ corresponds to all the pairs $\mathcal{I}_1$ and $\mathcal{I}_2$ where $\mathcal{I}_i$ is a model of ϕ_i for $i = 1, 2$.

Graphically:



p_1	p_2	…	p_{m-2}	p_{m-1}	p_m	ϕ_1
0	0	…	0	0	0	1
0	0	…	0	0	1	0
0	0	…	0	1	0	1
0	0	…	0	1	1	0
0	0	…	1	0	0	1
0	0	…	1	0	1	0
…						
1	1	…	1	0	0	1
1	1	…	1	0	1	1
1	1	…	1	1	0	0
1	1	…	1	1	1	0

p_{m+1}	p_{m+2}	…	p_{n-2}	p_{n-1}	p_n	ϕ_2
0	0	…	0	0	0	0
0	0	…	0	0	1	0
0	0	…	0	1	0	1
0	0	…	0	1	1	0
0	0	…	1	0	0	1
0	0	…	1	0	1	0
…						
1	1	…	1	0	0	1
1	1	…	1	0	1	1
1	1	…	1	1	0	0
1	1	…	1	1	1	0

p_1	p_2	…	p_{m-2}	p_{m-1}	p_m	p_{m+1}	p_{m+2}	…	p_{n-2}	p_{n-1}	p_n	$\phi_1 \wedge \phi_2$
0	0	…	0	0	0	0	0	…	0	0	0	0
0	0	…	0	0	0	0	0	…	0	0	1	0
0	0	…	0	0	0	0	0	…	0	1	0	1
0	0	…	0	0	0	0	0	…	0	1	1	0
0	0	…	0	0	0	0	0	…	1	0	0	1
0	0	…	0	0	0	0	0	…	1	0	1	0
0	0	…	0	0	0	…						
0	0	…	0	0	0	1	1	…	1	0	0	1
0	0	…	0	0	0	1	1	…	1	0	1	1
0	0	…	0	0	0	1	1	…	1	1	0	0
0	0	…	0	0	0	1	1	…	1	1	1	0
0	0	…	0	0	1	0	0	…	0	0	0	0
0	0	…	0	0	1	0	0	…	0	0	1	0
0	0	…	0	0	1	0	0	…	0	1	0	0
0	0	…	0	0	1	0	0	…	0	1	1	0
0	0	…	0	0	1	0	0	…	1	0	0	0
0	0	…	0	0	1	0	0	…	1	0	1	0
0	0	…	0	0	1	…						
0	0	…	0	0	1	1	1	…	1	0	0	0
0	0	…	0	0	1	1	1	…	1	0	1	0
0	0	…	0	0	1	1	1	…	1	1	0	0
0	0	…	0	0	1	1	1	…	1	1	1	0
0	0	…	0	1	0	0	0	…	0	0	0	0
0	0	…	0	1	0	0	0	…	0	0	1	0
0	0	…	0	1	0	0	0	…	0	1	0	1
0	0	…	0	1	0	0	0	…	0	1	1	0
0	0	…	0	1	0	0	0	…	1	0	0	1
0	0	…	0	1	0	0	0	…	1	0	1	0
0	0	…	0	1	0	…						
0	0	…	0	1	0	1	1	…	1	0	0	1
0	0	…	0	1	0	1	1	…	1	0	1	1
0	0	…	0	1	0	1	1	…	1	1	0	0
0	0	…	0	1	0	1	1	…	1	1	1	0

□

Exercise 108:

Explain in at most 10 lines, why if ϕ_1 and ϕ_2 don't share propositional variabels then $\#SAT(\phi_1 \vee \phi_2) = \#SAT(\phi_1) \cdot 2^{|\mathcal{P}_2|} + \#SAT(\phi_2) \cdot 2^{|\mathcal{P}_1|}$, where $\mathcal{P}_1$ ad $\mathcal{P}_2$ are the set of propositional variables that occours in ϕ_1 and ϕ_2 respectively.

Exercise 109:

Find a formula for $\#SAT(\phi_1 \leftrightarrow \phi_2)$ under the ussumption that ϕ_1 and ϕ_2 don't share any propositional variable.

Solution Notice that $\phi_1 \leftrightarrow \phi_2$ is equivalent to $(\phi_1 \wedge \phi_2) \vee (\neg\phi_1 \wedge \neg\phi_2)$. Furthermore the set of models that satisfy $(\phi_1 \wedge \phi_2)$ is disjoint from the set of models that satisfy $(\neg\phi_1 \wedge \neg\phi_2)$. This implies that

$$\#SAT(\phi_1 \veebar \phi_2) = \#SAT(\phi_1 \wedge \phi_2) + \#SAT(\neg\phi_1 \wedge \neg\phi_2)$$

Since ϕ_1 and ϕ_2 don't share any propositional variable, then we have that

$$\#SAT(\phi_1 \veebar \phi_2) = \#SAT(\phi_1) \cdot \#SAT(\phi_2) + \#SAT(\neg\phi_1) \cdot \#SAT(\neg\phi_2)$$

Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be the set of propositional variables occurring in $\phi_!$ and ϕ_2 respectively, then we have that $\#SAT(\neg\phi_i) = 2^{|\mathcal{P}_i)} - \#SAT(\phi_i)$ for $i = 1, 2$. We can therefore conclude that :

$$\begin{aligned}\#SAT(\phi_1 \veebar \phi_2) &= \#SAT(\phi_1) \cdot \#SAT(\phi_2) + (2^{|\mathcal{P}_1|} - \#SAT(\phi_1)) \cdot (2^{|\mathcal{P}_2|} - \#SAT(\phi_2)) \\ &= 2^{|\mathcal{P}_1 \cup \mathcal{P}_2|} - 2^{|\mathcal{P}_1|}\#SAT(\phi_2) - 2^{|\mathcal{P}_2|}\#SAT(\phi_1) \\ &+ 2 \cdot \#SAT(\phi_1) \cdot \#SAT(\phi_2)\end{aligned}$$

□

Solution(alternative) According to the definition of exclusive or, we have that $\phi_1 \veebar \phi_2$ is equivsalent to

$$(\phi_1 \wedge \neg\phi_2) \vee (\neg\phi_1 \wedge \phi_2)$$

The above formula has the following "nice" properties. The disjunciton is deterministic, which means that there is no interpretations that satisfies both the formulas of the dijunction. I.e., the models of $(\phi_1 \wedge \neg\phi_2)$ are not models of $(\neg\phi_1 \wedge \phi_2)$ and viceversa. Furthermore the disjunciton is also smooth, i.e, the set of propositional variables in the two disjuncts are the same. The third "nice"property derives from the fact that ϕ_1 and ϕ_2 do not share any propositional variable, which means that, we can obtain the model counting of the conjunction as the product of the model counting fo the conjuncts. We can therefore conclude that

$$\#\mathrm{SAT}(\phi_1 \veebar \phi_2) = s_1 \cdot (2^{n_2} - s_2) + s_2 \cdot (2^{n_1} - s_1)$$

where $s_i = \#\mathrm{SAT}(\phi_i)$ and n_i is the number of propositional variables that appear in ϕ_i for $i = 1, 2$. □

Exercise 110:

Let $\veebar$ be the connective for exclusive or, i.e., $p \veebar q$ is equivalent to $(p \wedge \neg q) \vee (\neg p \wedge q)$. Express $\#SAT(\phi_1 \veebar \phi_2)$ in terms of $\#SAT(\phi_1)$ and $\#SAT(\phi_2)$ under the hypothesis that ϕ_1 and ϕ_2 don't share any propositional variable. Explain your solution.

Exercise 111:

Show that if $\models \phi_1 \equiv \phi_2$, then

$$\#SAT(\phi_1) = \#SAT(\phi_2) \cdot 2^{|\mathcal{P}_2 \setminus \mathcal{P}_1| - |\mathcal{P}_1 \setminus \mathcal{P}_2|}$$

where $\mathcal{P}_i$ is the set of propositional variables occourring in ϕ_i for $i = 1, 2$.

Solution Let us consider the special case in which $\mathcal{P}_2 \subseteq \mathcal{P}_1$, We prove that

$$\#SAT(\phi_1) = \#SAT(\phi_2) \cdot 2^{|\mathcal{P}_1 \setminus \mathcal{P}_2|}$$

Let $\mathcal{I}_2$ be an assignment to $\mathcal{P}_2$ that satisfies ϕ_2, and let $\mathcal{I}_1$ be an estension of $\mathcal{I}_2$ for the variables in $\mathcal{P}_1 \setminus \mathcal{P}_2$. We have that $\mathcal{I}_1 \models \phi$ and since $\models \phi_1 \equiv \phi_2$ then $\mathcal{I}_2 \models \phi_1$. Since there are $2^{|\mathcal{P}_1 \setminus \mathcal{P}_2|}$ different extensions of $\mathcal{I}_2$, we have that

$$\#SAT(\phi_1) \geq \#SAT(\phi_2) \cdot 2^{|\mathcal{P}_1 \setminus \mathcal{P}_2|}$$

. SInce we also have that $\models \neg\phi_1 \equiv \neg\phi_2$, for the same argument we can show that

$$\#SAT(\neg\phi_1) \geq \#SAT(\neg\phi_2) \cdot 2^{|\mathcal{P}_1 \setminus \mathcal{P}_2|}$$

. Which implies that

$$2^{|\mathcal{P}_1|} - \#SAT(\phi_1) \geq (2^{|\mathcal{P}_2} - \#SAT(\phi_2)) \cdot 2^{|\mathcal{P}_1 \setminus \mathcal{P}_2|}$$

Since $\mathcal{P}_2 \subseteq \mathcal{P}_2$ we can conlcude that

$$\#SAT(\phi_1) \leq \#SAT(\phi_2) \cdot 2^{|\mathcal{P}_1 \setminus \mathcal{P}_2|}$$

And therefore we can have the following lemma:

$$\begin{aligned} &\text{forall } \phi_1 \text{ and } \phi_2 \text{ if } \models \phi_1 \equiv \phi_2 \text{ and } \mathcal{P}_2 \subseteq \mathcal{P}_2 \\ &\qquad \text{then } \#SAT(\phi_1) = \#SAT(\phi_2) \cdot 2^{|\mathcal{P}_1 \setminus \mathcal{P}_2|} \end{aligned} \tag{72}$$

We can now use lemma (72) to show the main result. First notice that if $\models \phi_1 \equiv \phi_2$, then $\models \phi_i \equiv \phi_1 \wedge \phi_2$ for $i = 1, 2$. By applying lemma (72) we have that

$$\#SAT(\phi_1 \wedge \phi_2) = \#SAT(\phi_1) \cdot 2^{|\mathcal{P}_1 \cup \mathcal{P}_2 \setminus \mathcal{P}_1|}$$

$$\#SAT(\phi_1 \wedge \phi_2) = \#SAT(\phi_2) \cdot 2^{|\mathcal{P}_1 \cup \mathcal{P}_2 \setminus \mathcal{P}_2|}$$

From which one can derive

$$\begin{aligned} \#SAT(\phi_1) \cdot 2^{|\mathcal{P}_1 \cup \mathcal{P}_2 \setminus \mathcal{P}_1|} &= \#SAT(\phi_2) \cdot 2^{|\mathcal{P}_1 \cup \mathcal{P}_2 \setminus \mathcal{P}_2|} \\ \#SAT(\phi_1) \cdot 2^{|\mathcal{P}_1 \setminus \mathcal{P}_1|} &= \#SAT(\phi_2) \cdot 2^{|\mathcal{P}_1 \setminus \mathcal{P}_2|} \\ \#SAT(\phi_1) &= \#SAT(\phi_2) \cdot 2^{|\mathcal{P}_1 \setminus \mathcal{P}_2| - |\mathcal{P}_1 \setminus \mathcal{P}_1|} \end{aligned}$$

□

CHAPTER 7

Weighted model counting

Perhaps, before introducing weightd model counting, this is a good time to summarise the inference tasks on propositional logic that we have described sofa including the one that we are going to describe in this section. They are listed in Table 7.

Task Name	Input	Result	Description
Model checking:	ϕ, $\mathcal{I}$	$\mathcal{I}(\phi)$	Compute the truth value that the interpretation $\mathcal{I}$ assigns to the formula ϕ, or equivalently check if ϕ is satisfied or not satisfied by $\mathcal{I}$.
Satisfiability:	ϕ	$\max_{\mathcal{I}} \mathcal{I}(\phi)$	Search for an assignment $\mathcal{I}$ to the propositional variables of ϕ that satisfies ϕ. If such an assignment does not exist ϕ is unsatisfiable otherwise it is satisfiable.
Maximum Satisfiability:	ϕ, w	$\max_{\mathcal{I}} \mathcal{I}(\phi) \cdot w(\mathcal{I})$	Search for an assignment $\mathcal{I}$ that satisfies the formula ϕ and maximize a weight function (or minimize a cost function) defined on the interpretations of the propositional variables in ϕ.
Model counting:	ϕ	$\sum_{\mathcal{I}} \mathcal{I}(\phi)$	Count how many assignments to the propositional variables of ϕ are models of (or equivalently satisfy) ϕ,
Weighted Model counting:	ϕ, w	$\sum_{\mathcal{I}} \mathcal{I}(\phi) \cdot w(\mathcal{I})$	Compute the weighted sum of the models of ϕ according to the weight function w.

1. Introduction

Weighted model counting is a generalisation of model counting where models have different weight, which is usually a positive numbers. Model counting is a special case of weighted model counting where each model weight is equal to 1. The most widespread application of weighted model counting is in probabilistic inference. Indeed, a recent and effective approach to probabilistic inference can be

obtained by reducing the problem to one of weighted model counting Chavira and Darwiche 2008b.

The problem of *Weighted model counting* can be formulated as follows: Given a propositional formula ϕ containing propositinal variables in $\mathcal{P}$ and a weight function w that assigns a non-negative weight to every truth assignment to propositional variables in $\mathcal{P}$, weighted model counting (henceforth WMC) concerns summing weights of the assignments that satisfy ϕ. formula. If every assignment has weight 1, the corresponding problem boils down to model counting.

DEFINITION 7.1 (Weighted model counting). *Foer every set of propositinal variables $\mathcal{P}$ and weight function $w : 2^{\mathcal{P}} \to \mathbb{R}^+$. the* weighted model counting *of a propositinal formula ϕ with propositinal variables in $\mathcal{P}$ w.r.t, w, is defined as:*

$$\text{WMC}(\phi, w) = \sum_{\mathcal{I}:\mathcal{P}\to\{0,1\}} \mathcal{I}(\phi) \cdot w(\mathcal{I}) \tag{73}$$

Weighted model counting and unweighted model counting (or simply model counting) share a lot. Indeed as it will be clear in this section, many of the techniques that have been developed for model counting can be equally applied or generalized for weighed model counting. There are also approaches that reduces weighted model counting to unweighted model counting Chakraborty et al. 2015. Concerning the application of weighted model counting to probabilistic reasoning we will describe them in details in this chapter.

In (unweighted) model counting each model of a formula counts 1; in weighted model counting some models are more important/probable/preferrable than others. To measure how much a model is important, it makes sense to associate a positive weight $w(\mathcal{I}) \geq 0$ to each interpretation $\mathcal{I}$. Why positive? Well this is not strictly necessary, however this is what happens. Furthermore, positive weights makes more direct the connection with probability. In weighted model counting each model $\mathcal{I}$ of a formula counts for its weight $w(\mathcal{I})$. The eight $w(\mathcal{I})$ associated to the model $\mathcal{I}$ can be interpreted in probabilistically; i.e. the weight is proportional to the likelihood of this model

EXAMPLE 7.1. *Suppose that a supermarked is selling item categories a, b, ... f, g. From the fidelity cards of the customers we observe the following records:*

#	*Itemsets*						
4	a	b	c	d			
1	a	b			e	f	
7	a	b	c				
3	a		c	d		f	
2							g
1				d			
4				d			g

Every row of the above table reports the number of customers that have bought a set of items that contains at least one item for each category listed in the row, (we dont count how many items of the category he/she has bought). Notice that there are $2^7 = 128$ combinations of itemssets, and the table reports only the combinations which has been observed at least once. Therefore, if a combination is not present then, it means that no customer has ever bought items of that combination of types.

Every combination of items can be seen as an interpretation on the set of propositions $a, b, \dots g$, where a means "the customer has bought at least one item of type a", similarly for b, dots g. We can consider the number of times we observe a combination as the weight of the corresponding interpretation. In other words we will have the follwoing weight function

			$\mathcal{I}$				
a	b	c	d	e	f	g	$w(\mathcal{I})$
1	1	1	1	0	0	0	4
1	1	0	0	1	1	0	1
1	1	1	0	0	0	0	7
1	0	1	1	0	1	0	3
0	0	0	0	0	0	1	2
0	0	0	0	1	0	0	1
0	0	0	0	1	0	1	4

We have 2^7 possible itemsets (interpretations $\mathcal{I}$), and we can assigns to each a weight equal to $w(\mathcal{I})$ where $w(\mathcal{I})$ is the number of times an itemset has been observed. Given this for every formula in the language of $a, \dots, g$ the weighted model counting returns the number of customer that bought items with tipes that respect the formula. For instance.

$\text{WMC}(a \wedge (b \vee c)) = 4 + 1 + 7 + 3 = 15$ *The number of times that a customer buys at least an item of type a and at least one of type b or c*

$\text{WMC}(a \wedge \neg g) = 0$ *No customer buys at the same time an item of type a and an item of type g.*

$\#\text{SAT}(a \to b) = 4 + 1 + 7 + 2 + 1 + 4 = 19$ *It counts the number of times a customer buys an item of type b. Notice that in the computation we also take into account all the cases in which the customer does not buy a. This is due to the "material implication" of propositional logicm that interprets $a \to b$ as $\neg a \vee b$.*

2. From #sat to wmc

Most of the results of model counting can be generalized to weight model counting.

2.1. The partition function.

PROPOSITION 7.1. *If ϕ is valid, then* $\text{WMC}(\phi, w)$ *is equal to* $\sum_{\mathcal{I}:\mathcal{P}\to\{0,1\}} w(\mathcal{I})$

The quantity $\sum_{\mathcal{I}:\mathcal{P}\to\{0,1\}} w(\mathcal{I})$ is the sum of the weights of all the interpretations w.r.t., the weight function w. When $\mathcal{P}$ is clear from the context we use the notation $\sum_{\mathcal{I}} w(\mathcal{I})$. This quantity has an important role in many formalizations

and algorithms. Therefore it has a special name: i.e., the *partition function of* w. It is usually denoted by $Z(w)$, or simply by Z, when w is clear from the context.

$$Z(w) = \sum_{\mathcal{I}} w(\mathcal{I}) \tag{74}$$

In many cases in which we have to perform weighted model counting and probabilistic reasoning we have to estimate $Z(w)$. This is usually a source of complexity. Indeed in the most general case computing (74) amounts in computing $w(\mathcal{I})$ for all the interpretations ı, wich means that we have to do 2^n calculations where n is the number of propositional variables in $\mathcal{P}$. Notice thta by seeing #SAT as a special case of WMC, where $w(\mathcal{I}) = 1$, computing the partition function $Z(w)$ is not problematinc since it is equal to 2^n.

Let us now see other propoerties of WMC.

PROPOSITION 7.2.

(1) If ϕ is unsatisfiable $\text{WMC}(\phi, w)$ *is equal to* 0*;*
(2) $\#\text{SAT}(\neg\phi) = Z(w) - \text{WMC}(\phi, w)$*;*
(3) If $\phi \models \psi$ *then* $\text{WMC}(\phi, w) \leq \text{WMC}(\psi, w)$*;*
(4) if ϕ is equivalent to ψ, then $\text{WMC}(\phi, w) = \text{WMC}(\psi, w)$*;*

PROOF. The proof is immediate. However it is worth noticing that property (3) is tightly connected to the fact tht the weight function is positive for all the interpretations. □

2.2. wmc and conjunction. Let us see how WMC behaves w.r.t, conjunction and dijunction. The property that allow to factorize model counting of a conjunction $\phi \wedge \phi$ where ϕ and ψ do not share propositional variables can be reformulated as follows: Let $\phi \wedge \psi$ be a formula on a set of propositional variables $\mathcal{P}$ such that $\mathcal{P}(\phi) \cap \mathcal{P}(\psi) = \emptyset$, and let w be a weight function on $\mathcal{P}$. Let us consider the propoerty

$$\text{WMC}(\phi \wedge \psi, w) = \text{WMC}(\phi, w_{\mathcal{P}(\phi)}) \cdot \text{WMC}(\psi, w_{\mathcal{P}(\psi)}) \tag{75}$$

where $w_{\mathcal{Q}}$ is way to restrict w to a given subset of propositions $\mathcal{Q} \subset \mathcal{P}$. In the general case property (75) does not hold. Consider the following example.

EXAMPLE 7.2. *Let $\mathcal{P} = \{p, q\}$. Consider the following two weight functions:*

$$w_1(\mathcal{I}) = 2 \cdot \mathcal{I}(p) + 3 \cdot \mathcal{I}(q)$$
$$w_3(\mathcal{I}) = 2^{\mathcal{I}(p)} \cdot 3^{\mathcal{I}(q)}$$

In the following two tables we show the weight funcitons on the left, and the weights of the formulas on the right.

p	q	$w_1(\mathcal{I})$	$w_2(\mathcal{I}))$
0	0	1	1
0	1	2	3
1	0	3	2
1	1	4	6

	p	q	$p \wedge q$
w_1	7	6	4
w_2	8	9	6

Let us now see how we can generalize the property of decomposition for model counting to weighted model counting:

PROPOSITION 7.3. *Let w be a weight function defined on a set of propositional variables $\mathcal{P} = \{p_1, \ldots, p_n\}$. For every $\mathcal{Q} \subseteq \mathcal{P}$ let $w_\mathcal{Q}$ be a weight function defined on the set $\mathcal{Q}$, such that $w_\mathcal{P} = w$. If, for every pair of formulas ϕ and ψ that do not share propositionla variables, it is true that:*

$$\text{WMC}(\phi \wedge \psi, w) = \text{WMC}(\phi, w_{\mathcal{P}(\phi)}) \cdot \text{WMC}(\psi, w_{\mathcal{P}(\psi)}) \tag{76}$$

if and only if w can be specified in the following form:

$$w(\mathcal{I}) = \begin{cases} \exp\left(\sum_{i=1}^{n} \left(v_i^+ \cdot \mathcal{I}(p_i) + v_i^- \cdot \mathcal{I}(\neg p_i)\right)\right) & \text{If } \mathcal{I} \models \phi \\ 0 & \text{Otherwise} \end{cases} \tag{77}$$

for some formula ϕ and $v_i^+, v_i^- \in \mathbb{R}$

PROOF. Let $\mathbb{I}$ be the set of interpretations such that $w(\mathcal{I}) \neq 0$, then let ϕ be be a propositional formula that is true if and only if $\mathcal{I} \in \mathbb{I}$. For every $\mathcal{I} \in \mathcal{I}$, let $\mathcal{P}^+$ the set of proposiitional variables true in $\mathcal{I}$ and $\mathcal{P}^-$ those that are false. Let $\phi_\mathcal{I} = \bigwedge \mathcal{P}^+ \wedge \bigwedge \neg P^-$. Since $\mathcal{I}$ is the ontly interpretation that satisfies $\phi_\mathcal{I}$ we have that $w(\mathcal{I}) = w(\phi_\mathcal{I})$. From (76) we have that

$$w(\mathcal{I}) = \prod_{p \in \mathcal{P}^+} w_{\{p\}}(p) \cdot \prod_{p \in \mathcal{P}^-} w_{\{p\}}(\neg p)$$

If we define $v_i^+ = \log(w_{\{p_i\}}(p_i)$ and $v_i^- = \log(w_{\{p_i\}}(\neg p_i)$, the previous expression can be rewritten as

$$\exp\left(\sum_{i=1}^{n} \left(v_i^+ \cdot \mathcal{I}(p_i) + v_i^- \cdot \mathcal{I}(\neg p_i)\right)\right)$$

The proof of the opposite direction is left by exercise. □

Not all the weight function can be expressed in the exponential form (76). Consider for instance the weight function w_1 of Example 7.1. If w_i were expressible in exponential form there should exists four values v_p^+, v_p^-, v_q^+, and v_q^- which are solutions of the following syste of equations:

$$\begin{cases} v_p^- + v_q^- = 0 \\ v_p^- + v_q^+ = \log 2 \\ v_p^+ + v_q^- = \log 3 \\ v_p^+ + v_q^+ = \log 4 \end{cases} \tag{78}$$

However, the system does not have a solution. Indeed, by summing the first and the fourth equation and subtracting the other two we obtain that $\log 4 = -\log 3 - \log 2$ which is false.

2.3. wmc and disjunction. Let us now see how weighted model counting behaves with disjunction. The key property for decomposing model counting of disjunction is determinism. By this property if $\phi \wedge \psi$ is unsatisfiable we have that $\#\text{SAT}(\phi \vee \psi) = \#\text{SAT}(\phi) \cdot 2^m + \#\text{SAT}(\psi) \cdot 2^n$ where m (resp. n) isthe number of propositional variables that occours in ψ but not in ϕ (resp, occour in ϕ but not in ψ). When $n = m = 0$, i.e., ϕ and ψ contains the same set of propositional variables, then we have that $\#\text{SAT}(\phi \vee \psi) = \#\text{SAT}(\phi) + \#\text{SAT}(\psi)$. This property is also true in weighted model counting

A disjunctive formula $\phi \vee \psi$ is called *smooth* if ϕ and ψ contains the same set of propositional variables. A formula ϕ is in sd-DNNF (smooth deterministic

decomposable negated normal form) if it is in d-DNNF and every disjunction is smooth.

PROPOSITION 7.4. *If $\phi \vee \psi$ is deterministic and smooth then*

$$\text{WMC}(\phi \vee \psi, w) = \text{WMC}(\phi, w_{\mathcal{P}(\phi)}) + \text{WMC}(\psi, w_{\mathcal{P}(\psi)}) \tag{79}$$

PROOF.

$$\begin{aligned}
\text{WMC}(\phi \vee \psi, w) &= \sum_{\mathcal{I} \models \phi \vee \psi} w(\mathcal{I}) \\
&= \sum_{\mathcal{I} \models \phi} w(\mathcal{I}) + \sum_{\mathcal{I} \models \psi} w(\mathcal{I}) && \text{By determinism} \\
&= \text{WMC}(\phi, w) + \text{WMC}(\psi, w) \\
&= \text{WMC}(\phi, w_{\mathcal{P}(\phi)}) + \text{WMC}(\psi, w_{\mathcal{P}(\psi)}) && \text{By smoothness}
\end{aligned}$$

□

To transform a d-DNNF formula into an sd-NNF formula we can apply the following rule:

- Smoothing left: For subformula $\phi \vee \psi$ with $p \in \mathcal{P}(\psi) \setminus \mathcal{P}(\phi)$ apply this transformation

$$\phi \wedge (p \vee \neg p) \vee \psi$$

- Smoothing right: For subformula $\phi \vee \psi$ with $p \in \mathcal{P}(\phi) \setminus \mathcal{P}(\psi)$ apply this transformation

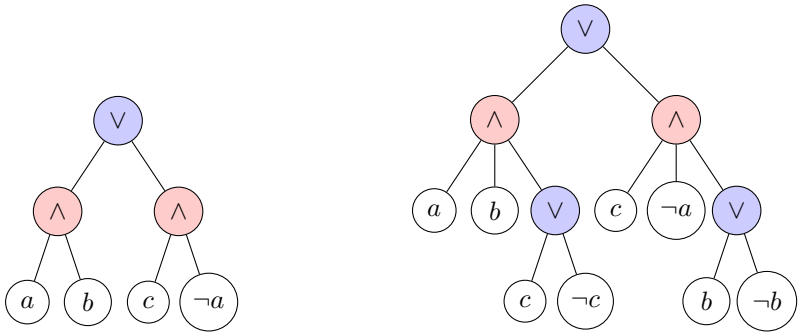
$$\phi \vee \psi \wedge (p \vee \neg p)$$

This results in:

$$\left(\phi \wedge \bigwedge_{p \in \mathcal{P}(\psi) \setminus \mathcal{P}(\phi)} (p \vee \neg p)\right) \vee \left(\psi \wedge \bigwedge_{q \in \mathcal{P}(\phi) \setminus \mathcal{P}(\psi)} (q \vee \neg q)\right)$$

EXAMPLE 7.3. *Smoothing $(a \wedge b) \vee (c \wedge \neg a)$ results in*

$$(a \wedge b \wedge (c \vee \neg c)) \vee ((c \wedge \neg a) \wedge (b \vee \neg b))$$



Notice that the conjunctions that are introduced by the smoothing rule are decomposable. Furthermore the disjunctions introduced by the smoothing rules are deterministic. Therefore by smooting an d-DNNF formula we will not loos the

fact the d-DNNF'ness. However, if we want to transform a generic formula in sd-NNF it is better to first transform it into d-DNNF and then apply smoothing in order to obtain an sd-DNNF formula.

Proceeding in the opposite direction would not be optimal. Consider the following example

EXAMPLE 7.4. *consider the formula* $(a \wedge b) \vee c$*, This formula is neither smooth nor deterministic. Should we try to first smooth it and then make it deterministic by applying Shannon's expansion? or should we proceed in the opposite direction? Let's analize the two cases:*

Smooth *then* determinism

$(a \wedge b) \vee c$	*Smoothing*
$((a \wedge b) \wedge (c \vee \neg c)) \vee (c \wedge (a \vee \neg a) \wedge (b \vee \neg b))$	*Shannon's expansion*
$(c \wedge ((a \wedge b) \vee (a \vee \neg a) \wedge (b \vee \neg b))) \vee (\neg c \wedge (a \wedge b))$	*the red disjunction is not deterministic. So we should apply again Shannon's expansion.*

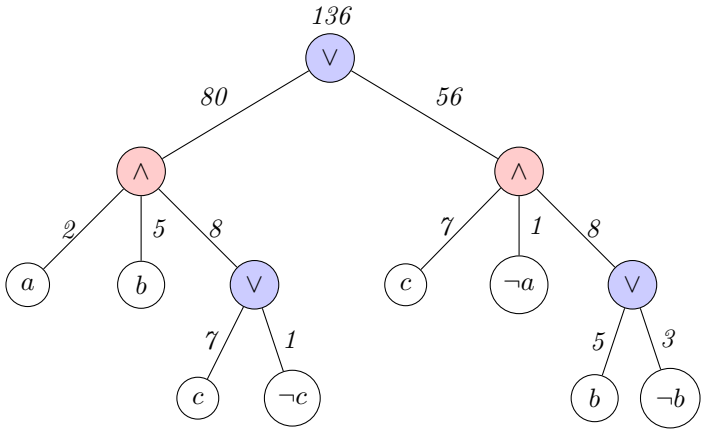
Instead if we apply first Shannon's expansion for determinism and then smoothing we proveed as follows:

$(a \wedge b) \vee c$	*Shannon's exp. on a*
$((b \vee c) \wedge a) \vee (c \wedge \neg a)$	*Shannon's exp. on b*
$((b \vee (c \wedge \neg b)) \wedge a) \vee (c \wedge \neg a)$	*Smoothing*
$((b \vee (c \wedge \neg b)) \wedge a) \vee (c \wedge \neg a \wedge (b \vee \neg b)$	*Smoothing*
$(((b \wedge (c \vee \neg c)) \vee (c \wedge \neg b)) \wedge a) \vee (c \wedge \neg a \wedge (b \vee \neg b))$	

2.4. Weighted model counting by knowledge compilation. When weight function has the exponential form, i.e., the weights are associated with literals weighted model counting can be performed on sd-DNNF formulas by transforming into a sum-product circuit. In particular, an sd-DNNF formula can be transformed a sum-product circuit as follows:

- Every leaf (literal) is associated with its weight;
- at every $\wedge$-node we perform the product of the child nodes;
- at every $\vee$-node we perform the sum of the child nodes.

EXAMPLE 7.5. *Consider the following weighted literals:* $w(a) = 2$*,* $w(\neg a) = 1$*,* $w(b) = 5$*,* $w(\neg b) = 3$*,* $w(c) = 7$ *and* $w(\neg c) = 1$*.*



3. Weights and probabilities

In this section we recall the formal relationship between weighted model coungint in propositional logic and probability. Let us first introduce the basic definition of probability measure.

3.1. Basic probability. A probability[1] space or event space is a set Ω together with a probability measure P on it. Ω is called the *sample space* and every element of ω is the otucome of some esperiment. Any subset of the sample space is called *event*.

Example 7.6.

(1) *Tossing a coin. The sample space is* $\Omega = \{H, T\}$, $= \{H\}$ *is an event.*
(2) *Tossing a die. The sample space is* $\Omega = \{1, 2, \ldots, 6\}$, $A = \{2, 4, 6\}$ *is an event, which can be described in words as "the result of the draw is an even number".*
(3) *Tossing a coin twice. The sample space is* $\Omega = \{HH, HT, TH, TT\}$. $A = \{HH, HT\}$ *is an event, which can be described in words as "the frst toss results in a Head".*
(4) *Tossing a die twice. The sample space is* $\Omega = \{(1,1), (1,2), \ldots, (6,6)\}$, *which contains 36 elements. "The sum of the results of the two toss is equal to 10" is an event* $A = \{(i,j) \mid i + j = 10\}$.

Example 7.7. *The set* $\mathbb{I} = \{\mathcal{I} \mid \mathcal{I} : \mathcal{P} \to \{0,1\}\}$ *of assignments of a set of propositional variable* $\mathcal{P}$ *is a sample space, and the set of interpretations that satisfies a formula* ϕ *is an event.*

P associates a number in $[0,1]$ to every subset of Ω. A subset A of Ω is also called an *event*. This means that Pr associates the probability to each event $A \subseteq \Omega$

$$\Pr(A) = \text{probability of } A$$

with

$$0 \leq \Pr(A) \leq 1 \tag{80}$$

[1]This is not the most general definition of probability space, but it is sufficiend for our purposes.

The probability of the whole space Ω is normalized to be $\Pr(\Omega) = 1$ and the probability of the empty set is 0 i.e, $\Pr(\emptyset) = 0$. For an element $\omega \in \Omega$ we may call $\{\omega\}$ an atomic event, and write $\Pr(\omega)$ instead of the more precise notation $\Pr(\{\omega\})$ to denote it's probability.

For two disjoint subsets A and B of Ω

$$\Pr(A \cup B) = \Pr(A) + \Pr(B) \tag{81}$$

In this case we say that A and B are disjoint events.

When Ω is finite, every event A is equal to the union of the atomic events that are contained in A. More precisely, if $A = \{\omega_1, \ldots, \omega_k\}$, then A can be expressed the union of the atomic events,

$$A = \{\omega_1\} \cup \{\omega_2\} \cup \cdots \cup \{\omega_k\}$$

Notice that every pair $\{\omega_i\}$ and $\{\omega_j\}$ with $i \neq j$ $\{\omega_i\} \cap \{\omega_j\} = \emptyset$. This implies, that we can apply property (81), obtaining:

$$\Pr(A) = \Pr(\omega_1) + \Pr(\omega_2) + \cdots + \Pr(\omega_k)$$

In conclusion, if Ω is finite, the probability of every event can be obtained by summing the probability the atomic events that belong to the event A. In other words, if we know $\Pr(\omega)$ for every $\omega \in \Omega$ then we can compute the probability of every event A by

$$\Pr(A) = \sum_{\omega \in A} \Pr(\omega)$$

The requirment that $\Pr(\Omega) = 1$ imposes also the restution that

$$\sum_{\omega \in \Omega} \Pr(\omega) = 1$$

i.e, that Pr is normalized to 1.

An important notion in probability theory is that of *conditional probability*. Given two event A and B in Ω the probability of A conditioned by B, denoted by $\Pr(A \mid B)$ is defined as:

$$\Pr(A \mid B) = \frac{\Pr(A \cap B)}{\Pr(B)} \tag{82}$$

EXAMPLE 7.8. *Suppose that you draw a dice and you know that the result is larger than* 3*, what is the probability that it is an odd number? Conditional probability $P(A \mid B)$ provide the answer to this question. In this case, the conditioning event is "the result of the toss is > 3 (B in the formula) and the conditioned event is "the result of the toss is odd", A in the formula. If we are in presence of a fair dice, we have that*

$$Pr(A \mid B) = \frac{Pr(A \cap B)}{Pr(B)} = \frac{1}{3}$$

3.2. From weights to probabilities. As shown in Example 7.7, one can see the set of all interpretations $\mathbb{I}$ of a set of propositional variables $\mathcal{P}$ as a sample space, where each interpretation is a single outcome of an experiment. This sample space contains $2^{|\mathcal{P}|}$ elements, and therefore it is finite. The weight function w maps every

interpretation of $\mathcal{P}$ into a positive integer, we could therefore define the probability of an interpretation as the normalized weight:

$$\Pr(\mathcal{I}) = \frac{w(\mathcal{I})}{\sum_{\mathcal{I}\in\mathbb{I}} w(\mathcal{I})} \tag{83}$$

Notice that if $w(\mathcal{I}) \geq 0$ for every $\mathcal{I}$ then Pr is a probability measure. Indeed we have that The probability of the set of all interpretations is equal to 1:

$$\Pr(\mathbb{I}) = \sum_{\mathcal{I}\in\mathbb{I}} \Pr(\mathcal{I}) = \sum_{\mathcal{I}\in\mathbb{I}} \frac{w(\mathcal{I})}{\sum_{\mathcal{I}\in\mathbb{I}} w(\mathcal{J})} = \frac{\sum_{\mathcal{I}\in\mathbb{I}} w(\mathcal{I})}{\sum_{\mathcal{I}\in\mathbb{I}} w(\mathcal{I})} = 1$$

Every formula ϕ defines a set $\mathbb{I}_\phi = \{\mathcal{I} \in \mathbb{I} \mid \mathcal{I} \models \phi\}$ of interpretations, which contains all the interpretations that satisfies ϕ. This means that every ϕ corresponds to the event $\mathbb{I}_\phi \subseteq \mathbb{I}$. Given this corrispondence, we can identify the event $\mathbb{I}_\phi$ with a formula ϕ, which allows us to talk about a "probability of a formula ϕ" denoted by $\Pr(\phi)$, and defined by the following equation:

$$\Pr(\phi) = \sum_{\mathcal{I}\models\phi} \Pr(\mathcal{I}) \tag{84}$$

if we plug in (84) the definition of $\Pr(\mathcal{I})$ given in (83) we obtain

$$\Pr(\phi) = \sum_{\mathcal{I}\models\phi} \frac{w(\mathcal{I})}{\sum_{\mathcal{I}\in\mathbb{I}} w(\mathcal{I})} = \frac{\sum_{\mathcal{I}\models\phi} w(\mathcal{I})}{\sum_{\mathcal{I}\models\top} w(\mathcal{I})}$$

which is equal to

$$\Pr(\phi \mid w) = \frac{1}{Z(w)} \text{WMC}(\phi, w) \tag{85}$$

Weighted model counting allows also to define conditional probability. WHich is $\Pr(\phi \mid \psi)$. Let us apply the definition:

$$\Pr(\phi \mid \psi) = \frac{\Pr(\phi \cap \psi)}{\Pr(\psi)}$$

What is the event $\phi \cap \psi$?, In our convention ϕ denotes the event $\mathbb{I}_\phi$ of the propositional assignments that satisfies ϕ and similarly $\mathbb{I}_\psi$. Therefore $\phi \cap \psi$ denotes $\mathbb{I}_\phi \cap \mathbb{I}_\psi$, which is the set of propositional interpretations that satisfy both ϕ and ψ. This coincides with the event $\mathbb{I}_{\psi\wedge\psi}$, also written as $\phi \wedge \psi$. We therefore have that

$$\Pr(\phi \mid \psi) = \frac{\Pr(\phi \wedge \psi)}{\Pr(\psi)}$$

If we replace the definition of probability of a formula in terms of weighted model counting provided in (85) we obtain

$$\Pr(\phi \mid \psi) = \frac{\frac{\text{WMC}(\phi\wedge\psi)}{\text{WMC}(\top)}}{\frac{\text{WMC}(\psi)}{\text{WMC}(\top)}} = \frac{\text{WMC}(\phi \wedge \psi)}{\text{WMC}(\psi)} \tag{86}$$

4. wmc for inference in Bayesian Networks

Bayesian networks (BNs) are graphical models (graphical in the sense that they are based on a graph) for probabilistic reasoning. The basic structure of a Bayesian networks is a directed acyclic graph where the nodes represent variables (discrete or continuous) and edges represent direct probabilistic connections between them. These direct connections are often causal connections. In addition, BNs model the quantitative strength of the connections between variables, allowing probabilistic beliefs about them to be updated automatically as new information becomes available.

4.1. Boolean Bayesian networks. In this chapter we consider only bayesian network which are based only on boolean random variables, though bayesian networks are defined on any type of random variables.

A directed acyclic graph $C = (V, E)$ is a graph on the set of vertices V and with directed edges $E \subseteq V \times V$ that does not contains cycles. A cycle is a sequence $(v_0, v_1), (v_1, v_2), \ldots, (v_{n-1}, v_n)$ where $v_0 = v_n$ and for all $0 \leq i < n$ $(v_i, v_{i+1}) \in E$. Given a directed graph $G = (V, E)$ for every vertex $v \in V$, we define $\mathrm{par}(v) = \{v' \mid (v', v) \in E\}$.

DEFINITION 7.2 (Bayesian Network). *A* Bayesian network *on a set of random variables $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ is a pair $\mathcal{B} = (G, Pr)$ is a pair composed of a directed acyclic graph $G = ([n], E)$ (where $[n] = \{1, \ldots, n\}$) and Pr specifies the conditional probabilities*

$$Pr(X_i = x_i \mid \boldsymbol{X}_{par(i)} = \boldsymbol{x}_{par(i)})$$

for every $X_i \in \boldsymbol{X}$. $\mathcal{B}$ uniquely define the join distribution on $\boldsymbol{X}$

$$Pr(\boldsymbol{X} = \boldsymbol{x}) = \prod_{i=1}^{n} Pr(X_i = x_i \mid \boldsymbol{X}_{par(i)} = \boldsymbol{x}_{par(i)}) \tag{87}$$

EXAMPLE 7.9. *The following simple Bayesian Netsork*

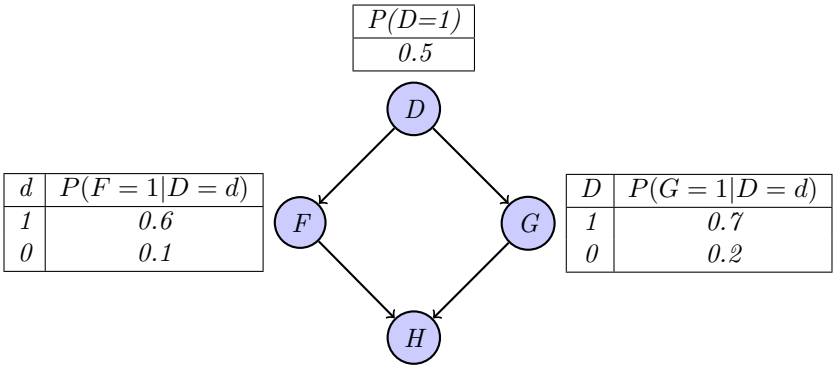
$Pr(A) = 1$
0.3

a	$Pr(B = 1 \mid A = a)$
0	0.4
1	0.9

specifies the joint probability distribution $P(A, B) = P(A) \cdot P(B \mid A)$ shown in the following table

a	b	$P(A = a, B = b)$
0	0	0.42
0	1	0.28
1	0	0.03
1	1	0.27

EXAMPLE 7.10. *As a second example consider the Bayesian network shown in this picture taken from Sang, Beame, and H. Kautz 2005*



d	$P(F=1\|D=d)$
1	*0.6*
0	*0.1*

D	$P(G=1\|D=d)$
1	*0.7*
0	*0.2*

f	g	$P(H=1\|F=f,G=g)$
1	*1*	*1.0*
1	*0*	*0.5*
0	*1*	*0.4*
0	*0*	*0.0*

The above Bayesian Network specifies a probability distribution on four boolean random variables, D, F, G, and H. Since they are boolean, they can take two values 0 and 1. As in propositional logic, boolean variables are used to express propositions, In this example we have that the propositional/random variables D, E, F and G stand for the following propositions:

D :	*John is Doing some work*
F :	*John has Finished his work*
G :	*John is Getting tired*
H :	*John Has a rest*

Every node of the graph is associated with a table that expresses the probability of the variables conditioned to the values of the parents. Theset tables are called conditional probabuility table (CPT). For instance the table associated to the node F states that:

$$Pr(F = 1 \mid D = 1) = 0.6$$
$$P(F = 1 \mid D = 0) = 0.1$$

The table specifies only the conditional probability for one of the two values of the boolean variable, since the value of the other can be obtained by difference.

$$Pr(F = 0 \mid D = 1) = 1 - Pr(F = 1 \mid D = 1) = 0.5$$
$$Pr(F = 0 \mid D = 1\text{‘}) = 1 - Pr(F = 1 \mid D = 0) = 0.9$$

The above Bayesian Network specifies the following joint distribution on D, F, G, H

d	f	g	h	$Pr(D, F, G, H = d, f, g, h)$
0	0	0	0	$0.5 \cdot 0.9 \cdot 0.8 \cdot 1.0 = 0.360$
0	0	0	1	$0.5 \cdot 0.9 \cdot 0.8 \cdot 0.0 = 0.000$
0	0	1	0	$0.5 \cdot 0.9 \cdot 0.2 \cdot 0.6 = 0.054$
0	0	1	1	$0.5 \cdot 0.9 \cdot 0.2 \cdot 0.4 = 0.036$
0	1	0	0	$0.5 \cdot 0.1 \cdot 0.8 \cdot 0.6 = 0.024$
0	1	0	1	$0.5 \cdot 0.1 \cdot 0.8 \cdot 0.4 = 0.016$
0	1	1	0	$0.5 \cdot 0.1 \cdot 0.2 \cdot 0.0 = 0.000$
0	1	1	1	$0.5 \cdot 0.1 \cdot 0.2 \cdot 1.0 = 0.010$
1	0	0	0	$0.5 \cdot 0.4 \cdot 0.3 \cdot 1.0 = 0.060$
1	0	0	1	$0.5 \cdot 0.4 \cdot 0.3 \cdot 0.0 = 0.000$
1	0	1	0	$0.5 \cdot 0.4 \cdot 0.7 \cdot 0.6 = 0.084$
1	0	1	1	$0.5 \cdot 0.4 \cdot 0.7 \cdot 0.4 = 0.056$
1	1	0	0	$0.5 \cdot 0.6 \cdot 0.3 \cdot 0.5 = 0.045$
1	1	0	1	$0.5 \cdot 0.6 \cdot 0.3 \cdot 0.5 = 0.045$
1	1	1	0	$0.5 \cdot 0.6 \cdot 0.7 \cdot 0.0 = 0.000$
1	1	1	1	$0.5 \cdot 0.6 \cdot 0.7 \cdot 1.0 = 0.210$
				1.000

From the previous example it should be clear that a Bayesian Network $\mathcal{B}$ with boolean variables expresses a probility distribution on a set of inbterpretations on the propositional variables corresponding to the random variables of $\mathcal{B}$. Therefore, An assignment to the random variables corresponds to a propositional interpretation. To highlight this fact from now on we use $\mathcal{P}$ to denot the set of random boolean variables (propositional variables) and $\mathcal{I}$ to denote an assignment to them.

There are many possible tasks that can be done with a given Bayesian network $\mathcal{B} = (G, Pr)$ on a set of propositional variables $\mathcal{P}$. The task we consider here is *Conditional Probability Queries.*

DEFINITION 7.3 (Conditional Probability Queries). *Given a Bayesian Network $\mathcal{B}$ on a set of propositional variables $\mathcal{P}$ a conditional probability queries is an expression of the form $Pr_{\mathcal{B}}(\phi \mid \psi)$, where ψ is called the evidence, and ϕ the query. The anwer to this query is*

$$\frac{\Pr_{\mathcal{B}}(\phi \wedge \psi)}{\Pr_{\mathcal{B}}(\phi)} = \frac{\sum_{\mathcal{I} \models \phi \wedge \psi} \Pr_{\mathcal{B}}(\mathcal{I})}{\sum_{\mathcal{I} \models \psi} \Pr_{\mathcal{B}}(\mathcal{I})} \tag{88}$$

In the previous example, If we want to know the probability that the work is finished (F) given the fact that we see John at the sun having a rest (H), we have to evaluate $Pr(F \mid H)$; in this case F is the query and H is the evidence.

4.2. Answering Conditional Probability Queries via wmc. Weighted model counting can be used to compute the value $\Pr_{\mathcal{B}} \phi \mid \psi$. To this purpose we exploit the relation between weighted model counting and probability, i.e., that

$$Pr(\phi|\psi) = \frac{\text{WMC}(\phi \wedge \psi, w)}{\text{WMC}(\psi, w)}$$

Therefore, the problem become to define a proper weight function w corresponding to a given Bayesian Network. The method is based on the following steps

(1) Extend the set of propositional variables $\mathcal{P}$ of the Bayesian Network, which are called *state variables* with the following new variables called *chance variables* For every state variable p that has parents $k > 0$ parent variables introduce 2^k new propositional variables $p_{\boldsymbol{b}}$ for every $\boldsymbol{b} \in \{0,1\}^k$. In the previous example we introduce G_0 and G_1, F_0 and F_1, and H_{00}, H_{01}, H_{10}, and H_{11}.

(2) To each of the introduced variables associate the weight specified in the corresponding line of the CPT. I.e., $w(p_{\boldsymbol{b}}) = \Pr(p = 1 \mid \text{par}(p) = \boldsymbol{b})$. In the above example we have the following weights:

$$\begin{aligned} w(D) &= 0.5 \\ w(F_0) &= 0.1 & w(F_1) &= 0.5 \\ w(G_0) &= 0.2 & w(G_1) &= 0.7 \\ w(H_{00}) &= 0.0 & w(H_{01}) &= 0.4 \\ w(H_{10}) &= 0.5 & w(H_{11}) &= 1.0 \end{aligned}$$

The weight of $\neg p_{\boldsymbol{b}}$ is set to $1 - w(p_{\boldsymbol{b}})$, and the weights of all the other literals are set to 1.

(3) The intuition of the choice variable $p_{\boldsymbol{b}}$ is that it will be true of the parent variables of p will take the values $\boldsymbol{b}$. So for instance F_0 it means that the only parent of F, which is D will be false. The third step is to connect the new variables to the variables in the graph; For every change variable $p_{\boldsymbol{b}}$ correspinding to a line $Pr(p = 1 \mid \text{par}(p) = \boldsymbol{b})$, we add the following formula, where $\text{par}(p) = \{p_1, \ldots, p_k\}$

$$p_{\boldsymbol{b}} \leftrightarrow p \wedge \left(\bigwedge_{\substack{i=1 \\ b_i=1}}^{k} p_i \wedge \bigwedge_{\substack{i=1 \\ b_i=0}}^{k} \neg p_i \right)$$

In our example we add the following formulas:

$$F_0 \leftrightarrow F \wedge \neg D \qquad F_1 \leftrightarrow F \wedge D \qquad G_0 \leftrightarrow G \wedge \neg D \qquad G_1 \leftrightarrow G \wedge D$$
$$H_{00} \leftrightarrow H \wedge \neg F \wedge \neg G \quad H_{01} \leftrightarrow H \wedge \neg F \wedge G \quad H_{10} \leftrightarrow H \wedge F \wedge \neg G \quad H_{11} \leftrightarrow H \wedge F \wedge G$$

(4) A further obtimization consistes in replacing literals with weight equal to 0 with formulas. In particular, if $w(l) = 0$ add the unweighted formula $\neg l$. and remove the weight of l. In the above example for instance we remove the weights for H_{00} and add $\neg H_{00}$.

Let $\Phi_{\mathcal{B}}$ and $w_{\mathcal{B}}$ be the conjunction of the formulas and the weight function obtained by applying the previous steps to Bayesian network $\mathcal{B}$, We then define the weight function

$$w_{\mathcal{B}}(\mathcal{I}) = \begin{cases} \prod_{\mathcal{I} \models p_{\boldsymbol{b}}} w(p_{\boldsymbol{b}}) \cdot \prod_{\mathcal{I} \not\models p_{\boldsymbol{b}}} w(\neg p_{\boldsymbol{b}}) & \text{if } \mathcal{I} \models \Phi_{\mathcal{B}} \\ 0 & \text{Otherwise} \end{cases}$$

Notice that w is more than a weight function. Indeed it is a probability distribution on the models of $\Phi_{\mathcal{B}}$, since the partition function $Z(w_{\mathcal{B}}) = \sum_{\mathcal{I}} w(\mathcal{I}) = \sum_{\mathcal{I} \models \Phi_{\mathcal{B}}} w(\mathcal{I}) = 1$, i.e., the weight of all the models of $\Phi_{\mathcal{B}}$ sum up to 1. Indeed we have the following proposition:

PROPOSITION 7.5. *Let $\mathcal{B}$ be a Bayesian networks on the boolean random variables $X_1, \ldots, X_n$ that defines the joint probability distribution $Pr(X_1, \ldots, X_n)$.*

- *for every assignment* $\boldsymbol{x} = (x_1, \ldots, x_n)$ *to the variables* $X_1, \ldots, X_n$. *there is a unique interpretaiton* $\mathcal{I}_{\boldsymbol{x}}$ *that satisfies* $\Phi_{\mathcal{B}}$ *and such that* $\mathcal{I}(X_i) = x_i$
- *For every* $\mathcal{I}$ *that satisfies* $\Phi_{\mathcal{B}}$

$$w_{\mathcal{B}}(\mathcal{I}) = Pr(X_1 = \mathcal{I}(X_1), \ldots, X_n = \mathcal{I}(X_n))$$

The answer to the conditional probability query $Pr(\phi \mid \psi)$ w.r.t, the Bayesian network $\mathcal{B}$ can be computed via weighted model counting.

$$\frac{\text{WMC}(\Phi_{\mathcal{B}} \wedge \phi \wedge \psi, w_{\mathcal{B}})}{\text{WMC}(\Phi_{\mathcal{B}} \wedge \psi, w_{\mathcal{B}})} \tag{89}$$

To this purpose we can use for instance knowledge compilation of $\Phi_{\mathcal{B}}$ in an sd-DNNF formula in order to compute the circuit. Notice that if the evidence ψ is a conjunction of literals we can set the weights of the opposite literal to 0 in the circuit of $\Phi_{\mathcal{B}}$ and we immediately obtain a circuit for $\Phi \wedge \psi$ Similarly, if the query ϕ is a conjunction of literals. THerefore in case of conjunctive conditional probability queryies (i.e., queries in which the evidence and the query are conjunctions of literals) once we have computed the circuit for $\Phi_{\mathcal{B}}$ we can easily computeall the answers of conditional probability conjunctive queries.

EXAMPLE 7.11. *The sd-DNNF of the* $\Phi_{\mathcal{B}}$ *for the previous example is*

$$\begin{aligned}
&D \wedge (F \wedge F_1 \wedge (G \wedge G_1 \wedge (H \wedge H_{11} \vee \neg H \wedge \neg H_{11})) \vee \\
&\qquad (\neg G \wedge \neg G_1 \wedge (H \wedge H_{10} \vee \neg H \wedge \neg H_{10}))) \vee \\
&\qquad (\neg F \wedge F_1 \wedge (G \wedge G_1 \wedge (H \wedge H_{01} \vee \neg H \wedge \neg H_{01})) \vee \\
&\qquad (\neg G \wedge \neg G_1 \wedge (H \wedge H_{00} \vee \neg H \wedge \neg H_{00}))) \vee \\
&\neg D \wedge (F \wedge F_0 \wedge (G \wedge G_0 \wedge (H \wedge H_{11} \vee \neg H \wedge \neg H_{11})) \vee \\
&\qquad (\neg G \wedge \neg G_0 \wedge (H \wedge H_{10} \vee \neg H \wedge \neg H_{10}))) \vee \\
&\qquad (\neg F \wedge F_0 \wedge (G \wedge G_0 \wedge (H \wedge H_{01} \vee \neg H \wedge \neg H_{01})) \vee \\
&\qquad (\neg G \wedge \neg G_0 \wedge (H \wedge H_{00} \vee \neg H \wedge \neg H_{00})))
\end{aligned} \tag{90}$$

Formula (90) *can be converted in a sum-product circuit.* $C_{\mathcal{B}}$ *that cam be used to compute the probability of any conjunctive formula. For instance to compute the probability of* $H \wedge \neg G$, *it is sufficient to set the weights of the literals* $\neg H$ *and* G *to* 0. *and compute the circuit. Therefore if we want to answer the conditional conjunctive query* $Pr(D \mid H \wedge \neg G)$ *we use the circuit to compute* $Pr(D \wedge H \wedge \neg G)$ *and* $Pr(H \wedge \neg G)$. *The anwer will be the fraction of the two results i.e.,*

$$\frac{Pr(D \wedge H \wedge \neg G)}{Pr(H \wedge \neg G)}$$

5. Learning weights

Until now, we have assumed that the weights associated to literals are given. In this last section, we describe a basic method to automatically learn the weights from a set of observations.

First, we have to define what is an observation. Since we want to learn a weight function (or a probability distribution) on the set of interpretations of a set of propositional variables $\mathcal{P}$ then the observations must be instances of such assugnments. Suppose we have a set of observations which are represented as a multi-set (i.e., a set with repeated objects) of interpretations $\mathbb{I} = \mathcal{I}^{(1)}, \mathcal{I}^{(2)}, \ldots, \mathcal{I}^{(d)}$ where d the the size of the observations. The criteria used to learn the weights is that of the *maximum likelihood* that requires to maximize the probability of

observing the data. Formally, we want to find a set of parameters (weights) $\boldsymbol{w}$ such that

$$\text{Likelihood}(\mathbb{I} \mid \boldsymbol{w}) = \Pr(\mathbb{I} \mid \boldsymbol{w}) \tag{91}$$

is maximal. We make the following simplifying hypothesis .The first one is that the observations are i.i.d. (independent and identically distributed). This hypothesis allows us to factorize the probability of all the observations as a product of the probability of each single observation.

$$\text{Likelihood}(\mathbb{I} \mid \boldsymbol{w}) = \prod_{i=1}^{d} \Pr(\mathcal{I} \mid \boldsymbol{w})$$

The second hypothesis concerns the form of $Pr(\cdot \mid \boldsymbol{w})$. We assume that this probability is specified via weighted model counting, i.e., that

$$Pr(\mathcal{I} \mid \boldsymbol{w}) = \frac{1}{Z(\boldsymbol{w})} w(\mathcal{I} \mid \boldsymbol{w})$$

The third and final hypothesis is that $w(\mathcal{I} \mid \boldsymbol{w})$ is specified by an exponential form w.r.t a given set of formulas $\phi_1, \ldots, \phi_n$. This means that

$$w(\mathcal{I} \mid \boldsymbol{w}) = \exp\left(\sum_{i=1}^{n} w_i \mathcal{I}(\phi_i)\right)$$

The problem consists in finding a tuple of weights $\boldsymbol{w} = (w_1, \ldots, w_n)$ that best fits the observed data. Putting everything toghether we are interested in the vector of real number $\boldsymbol{w} = (w_1, \ldots, w_n)$ that meximizes

$$\text{Likelihood}(\mathbb{I} \mid \boldsymbol{w}) = \frac{1}{Z(\boldsymbol{w})^d} \exp\left(\sum_{i=1}^{d}\sum_{j=1}^{n} w_j \mathcal{I}^{(i)}(\phi_j)\right)$$

$$Z(\boldsymbol{w}) = \sum_{\mathcal{I}} \exp\left(\sum_{j=1}^{n} w_j \mathcal{I}(\phi_j)\right)$$

It is convenient to pass to the logarithmic space. Indeed due to the monotonicity of the logarithmic function, we have that maximinzing a function $f(x)$ is equivalent to maximizing the logarithm of the function i.e. maximizing $\log(f(x))$. We therefore want to maximize the logarithm of the likelihood, also known as loglikelihood

$$\begin{aligned}
\text{LogLik}(\mathbb{I} \mid \boldsymbol{w}) &= \log(\text{Likelihood}(\mathbb{I} \mid \boldsymbol{w})) \\
&= \sum_{i=1}^{d}\sum_{j=1}^{n} w_j \cdot \mathcal{I}^{(i)}(\phi_j) - d \cdot \log(Z(\boldsymbol{w})) && (92) \\
&= \sum_{j=1}^{n}\sum_{i=1}^{d} w_j \cdot \mathcal{I}^{(i)}(\phi_j) - d \cdot \log(Z(\boldsymbol{w})) \\
&= \sum_{j=1}^{n} w_j \cdot n_j - d \cdot \log(Z(\boldsymbol{w})) && (93)
\end{aligned}$$

where n_j is the number of observations $\mathcal{I}^{(i)}$ for which the fornula ϕ_j is true. We can try to maximize (93) with gradient ascent approach, by putting to zeros the

partial derivatives of the log likelihood w.r.t. the parameters w_j.

$$\begin{aligned}
\frac{\partial \text{logLik}(\mathbb{I} \mid \boldsymbol{w})}{\partial w_i} &= 0 \\
\frac{\partial \left(\sum_{j=1}^{n} w_j \cdot n_j - d \cdot \log(Z(\boldsymbol{w})) \right)}{\partial w_j} &= 0 \\
n_j - \frac{d \cdot \sum_{\mathcal{I}} \mathcal{I}(\phi_j) \cdot \exp\left(\sum_{j=1}^{n} w_j \cdot \mathcal{I}(\phi_j) \right)}{Z(\boldsymbol{w})} &= 0
\end{aligned}$$

requires exponential amount of time. We can use an approximation by learning the weight of each formulas separately. I.e., we assume that it is the only formula that we use to cmpute the weight function. We therefore consider the case where we have a unique formula ϕ_1 and compute the derivative w.r.t. the only parameter w_1.

$$\begin{aligned}
n_1 - \frac{d \cdot \sum_{\mathcal{I}} \mathcal{I}(\phi_1) \cdot \exp(w_1 \cdot \mathcal{I}(\phi_1))}{Z(\boldsymbol{w})} &= 0 \\
n_1 - \frac{d \cdot \sum_{\mathcal{I} \models \phi_1} \cdot \exp(w_1)}{Z(\boldsymbol{w})} &= 0 \\
n_1 - \frac{d \cdot \#\text{SAT}(\phi_1) \cdot \exp(w_1)}{Z(\boldsymbol{w})} &= 0 \\
n_1 - \frac{d \cdot \#\text{SAT}(\phi_1) \cdot \exp(w_1)}{\sum_{\mathcal{I} \models \phi_1} \exp(w_1) + \sum_{\mathcal{I} \not\models \phi_1} \exp(0)} &= 0 \\
\frac{d \cdot \#\text{SAT}(\phi_1) \cdot \exp(w_1)}{\#\text{SAT}(\phi_1) \cdot \exp(w_1) + \#\text{SAT}(\neg\phi_1)} &= n_1 \\
n_1 \cdot \exp(w_1) \cdot \#\text{SAT}(\phi_1) + n \cdot \#\text{SAT}(\neg\phi_1) &= d \cdot \exp(w_1) \cdot \#\text{SAT}(\phi_1) \\
\frac{n_1 \cdot \#\text{SAT}(\neg\phi_1)}{(d - n_1)\#\text{SAT}(\phi_1)} &= \exp(w_1) \\
w_1 &= \log\left(\frac{n_1 \cdot \#\text{SAT}(\neg\phi_1)}{(d - n_1) \cdot \#\text{SAT}(\phi_1)} \right)
\end{aligned}$$

IN the following we summarize how weighted model counting can be used to perform probabilistic prediction starting from a set of observations. Suppose that you are interested in doing some predictions which are expressible with a propositional formula Q starting from a set of evidences which are expressed in terms of a propositional formula E. In synthesis you want learn a set of parameters $\boldsymbol{w}$ that allows you to answer the query $Pr(Q \mid E, \boldsymbol{w})$.

(1) collect a set of d observations, i.e., $\mathcal{I}^{(1)}, \ldots, \mathcal{I}^{(d)}$ from which you want to extract knowledge that can be used to answer your query.
(2) select a set set of propositional formulae $\phi_1, \ldots, \phi_n$ that you want to use to describe the properties of your data. How to choose this formula is a matter of design. A possible criteria is to conisder formulas such that their truth value has some impact on the answer to the query $\Pr(Q \mid E, \boldsymbol{w})$.
(3) learn the weights $w_1, \ldots, w_k$, separately using formula:

$$w_j = \log\left(\frac{n_j \cdot \#\text{SAT}(\neg\phi_j)}{(d - n_j) \cdot \#\text{SAT}(\phi_j)} \right) \tag{94}$$

(4) apply inference i.e., compute $\Pr(Q \mid E, \boldsymbol{w})$ via weighted model counting. i.e.

$$\Pr(Q \mid E, \boldsymbol{w}) = \frac{\text{WMC}(Q \wedge E, \boldsymbol{w})}{\text{WMC}(E, \boldsymbol{w})}$$

EXAMPLE 7.12. *Suppose that we have* $\mathbb{I} = \mathcal{I}^{(1)}, \ldots, \mathcal{I}^{(22)}$ *are summarized in the following table:*

#	*Itemsets*
4	a b c d
1	a b e f
7	a b c
3	a c d f
2	g
1	d
4	d g

Suppose that we are interested in answering query of the form $Pr(x \mid y \wedge \neg z)$ *for every* $x, y, z \in \{a, b, c, d, e, f, g\}$. *We can learn the weights of a set of formulas. See for instance the following (randomly choosen)*

$$a \qquad w = \log\left(\frac{15 \cdot 2^6}{7 \cdot 2^6}\right) \approx 0.76$$

$$\neg a \qquad w = \log\left(\frac{7 \cdot 2^6}{15 \cdot 2^6}\right) \approx -0.76$$

$$e \qquad w = \log\left(\frac{1 \cdot 2^6}{21 \cdot 2^6}\right) \approx -3.04$$

$$\neg e \qquad w = \log\left(\frac{21 \cdot 2^6}{1 \cdot 2^6}\right) \approx 3.04$$

$$a \wedge b \qquad w = \log\frac{12 \cdot (2^7 - 2^5)}{10 \cdot 2^5} \approx 8.21$$

$$c \wedge d \qquad w = \log\frac{7 \cdot (2^7 - 2^5)}{15 \cdot 2^5} \approx 7.27$$

$$e \wedge f \qquad w = \log\frac{1 \cdot (2^7 - 2^5)}{21 \cdot 2^5} \approx 4.99$$

$$a \rightarrow b \qquad w = \log\frac{19 \cdot (2^7 - 3 \cdot 2^5)}{3 \cdot 3 \cdot 2^5} \approx 0.75$$

$$a \wedge b \wedge c \wedge \neg e \wedge \neg f \rightarrow g \qquad w = \log\left(\frac{11}{11 \cdot (2^7 - 2)}\right) \approx -4.84$$

$$a \wedge b \wedge \neg c \wedge \neg d \wedge e \wedge f \wedge \neg g \qquad w = \log(21 \cdot (2^7 - 1)) \approx 7.89$$

And then estimate for instance $Pr(a \mid b, \neg c)$ *via weighted model counting.*

6. Exercises

Exercise 112:

Given a set of propositional variable $\mathcal{P}$, let $w(\mathcal{I}) = |\{p \in \mathcal{P} \mid \mathcal{I} \models p\}|$, i.e., $w(\mathcal{I})$ is the number of propositional variables that are true in $\mathcal{I}$. Compute WMC($\top$).

Exercise 113:

Given a set of propositional variable $\mathcal{P}$, let $w(\mathcal{I}) = x^{|\{p \in \mathcal{P} | \mathcal{I} \models p\}|}$ for some $x \neq 0$. Compute WMC($\top$).

Exercise 114:

Let $\mathcal{P} = \{p, q\}$ be a set of propositional variables. Check if the following two weight functions on the interpretations of $\mathcal{P}$ can be expressed in terms of a weight function on the literals of $\mathcal{P}$.

(95)

$\mathcal{I}$	p	q	$w(\mathcal{I})$
$\mathcal{I}_1$	0	0	0
$\mathcal{I}_2$	0	1	1
$\mathcal{I}_3$	1	0	2
$\mathcal{I}_4$	1	1	3

Solution A weight funciton on the interpretation can be expressed in terms of a weight function on the literals, if the weight of the interpretation is equal to the product of the weight of the literals that are true in the interpretation, i.e., if

$$w(\mathcal{I}) = \prod_{p \in \mathcal{P}} w(p)^{\mathcal{I}(p)} w(\neg p)^{\mathcal{I}(\neg p)}$$

Therefore, to be expressed in term of a weight funciton on literals the weight function (95) should be such that

$$\begin{aligned} w(\neg p) \cdot w(\neg q) &= 0 \\ w(\neg p) \cdot w(q) &= 1 \\ w(p) \cdot w(\neg q) &= 2 \\ w(p) \cdot w(q) &= 3 \end{aligned}$$

THe first equation implies that either $w(\neg p) = 0$ or $w(\neg q) = 0$; but the second equation implies that $w(\neg p) \neq 0$ and the third equation implies that $w(\neq q) \neq 0$, which is impossible. Therefore the weight function (95) cnanot be expressed in terms of weight funciton on literals. □

Exercise 115:

Suppose you have three coins: the faces of the first coin are black and white, the faces of the second coin are yellow and green, and the faces of the third coin are red and green. In an experiment you toss the first coin; if you obtain a black you toss the second coin otherwise you toss the third coin.

(1) Model this experiment in propositional logic and
(2) use model counting to determine what are the number of possible outcomes?

(3) Let p, q and r be the probability of obtaining a black, yellow, and red faces when tossing the first, second and third coin respectively. Compute the probability of obtaining an outcome which is either red or green.

Solution We can use the language B, W, R, G, Y to state that in the outcome there is a coin with a black, white, red, green, and yellow faces respectively. Notice that this is possible since there is no possibility to have outcomes with two coins with the same color face. We can now formalize the constraints of the game in terms of the following formulas:

$B + W = 1$	The toss of the first coin can have only one result among black and white
$R + Y + G = 1$	The toss of the second or third coin can have only one result since only one coin among the two is tossed
$B \rightarrow Y \vee G$	If you have a black then you can have only one among yellow and green since you toss the second coin
$W \rightarrow R \vee G$	If you have a white then you can have only one among red and green since you toss the third coin

The models that satisfies all the formulas are 4

$$\{B, Y\} \qquad \{B, G\} \qquad \{W, R\} \qquad \{W, G\}$$

If we associate the following weights:

$$w(B) = p \quad w(W) = 1 - p \quad w(Y) = q \quad w(G) = 1 - q \quad w(R) = r \quad w(G) = 1 - r$$

In this wey however we assign two weights to the same atom G. Indeed we have to distinguish when G is obtained by the second or by the third coin. To this purpose we introduce two new atoms

$$G_2 \leftrightarrow B \wedge G \qquad\qquad G_3 \leftrightarrow W \wedge G$$

Adding these new propositions does not change the number of models since they are fully defined in terms of the previous propositions. We still have 4 models by they are:

$$\{B, Y\} \qquad \{B, G, G_2\} \qquad \{W, R\} \qquad \{W, G, G_3\}$$

and update the weights for G to

$$w(G_2) = 1 - q \qquad\qquad w(G_3) = 1 - r$$

$$w(\{B, Y\}) = pq \quad w(\{B, G, G_2\}) = p(1 - q) \quad w(\{W, R\}) = (1 - p)r \quad w(\{W, G, G_3\}) = (1 - p))(1 - r)$$

Notice that the weight of all the models sum up to 1, and therefore they can be considered to be probabilities of the outcomes. Finally, the probability of in the result we have either red or green, is equal to the probability of the formula $R \vee G$ which can be computed by the sum of the probabilities of the models that satisfies $R \vee G$, i.e., $p(1 - q) + (1 - p)r + (1 - p)(1 - r) = p(1 - q) + 1 - p = 1 - pq$. □

Exercise 116:

Consider the set $\mathcal{P} = \{p, q\}$ of propositional variables and the following weight functions:

$\mathcal{I}$	p	q	$w_1(\mathcal{I})$
$\mathcal{I}_1$	0	0	1
$\mathcal{I}_2$	0	1	2
$\mathcal{I}_3$	1	0	3
$\mathcal{I}_4$	1	1	4

$\mathcal{I}$	p	q	$w_2(\mathcal{I})$
$\mathcal{I}_1$	0	0	4
$\mathcal{I}_2$	0	1	3
$\mathcal{I}_3$	1	0	8
$\mathcal{I}_4$	1	1	6

Check if they can be expressed in terms of a weight function $w : \mathcal{P} \to \mathbb{R}^+$.

Solution A weight function maps every interpretation of the propositional variables into a positive real number. This mapping can be specified explicilty, by providing the explicit weight for every interpretation (we have to specify 2^n numbers, where n is the number of propositional variables) This is how we have specified the two weight functions w_1 and w_2 of the exercise. Alternatively, and more compactly, the weight function can be specified indirectly, by associating a weight to every literal, and then define the weight of an assignment as the product of the weight of the literals that are satisfied by the assignment. i.e.,

$$w(\mathcal{I}) = \prod_{\substack{l \in Literals \\ \mathcal{I} \models l}} w(l)$$

With this method instead of specifying 2^n numbers we have to provide at most $2n$ parameters (corresponding to the number of literals). However, this is not always possible, i.e., there are weight functions that cannot be specified in terms of a weight function on the literals. The exercise asks if the weight functions w_1 and w_2 can or cannot be specified by a weight function defined on literals.

Let's start with w_1. If w_1 can be specified with the weight of the literals, then we have that

$$\begin{cases} w_1(\mathcal{I}_1) = 1 = w(\neg p) \cdot w(\neg q) \\ w_1(\mathcal{I}_2) = 2 = w(\neg p) \cdot w(q) \\ w_1(\mathcal{I}_3) = 3 = w(p) \cdot w(\neg q) \\ w_1(\mathcal{I}_4) = 4 = w(p) \cdot w(q) \end{cases}$$

To find the weight function for the literals, we have therefore to solve the following system of equations, where we have replaced $w(\neg p)$, $w(\neg q)$, $w(q)$, and $w(p)$, with a, b, c and d.

$$\begin{cases} a \cdot b = 1 \\ a \cdot c = 2 \\ d \cdot b = 3 \\ d \cdot c = 4 \end{cases} \Rightarrow \begin{cases} a = \frac{1}{b} \\ c = 2b \\ d = \frac{3}{b} \\ d = \frac{2}{b} \end{cases}$$

which does not have a solution. This implies that the weight funciton w_1 cannot be expressed in terms of weight function on literals. Let us now consider w_2. We proceed in the same way:

$$\begin{cases} a \cdot b = 4 \\ a \cdot c = 3 \\ d \cdot b = 8 \\ d \cdot c = 6 \end{cases} \Rightarrow \begin{cases} b = \frac{4}{a} \\ c = \frac{3}{a} \\ d = 2a \end{cases}$$

which has infinite solution for $a \neq 0$, for instance:

$$w(\neg p) = \frac{1}{2} \qquad w(\neg q) = 8 \qquad w(p) = 1 \qquad w(q) = 6$$
$$w(\neg p) = 1 \qquad w(\neg q) = 4 \qquad w(p) = 2 \qquad w(q) = 3$$

□

Exercise 117:
Consider the following weighted formulas

literal	A	$\neg A$	B	$\neg B$	C	$\neg C$
weight	1	2	1	2	1	2

Compute the weight and the probabilities of the formulas

$$(A \vee B) \rightarrow (B \vee C)$$

.

Solution Let us compute the weights of all the interpretations.

	A	B	C	$w(\mathcal{I})$
$\mathcal{I}_1$	0	0	0	8
$\mathcal{I}_2$	0	0	1	4
$\mathcal{I}_3$	0	1	0	4
$\mathcal{I}_4$	0	1	1	2
$\mathcal{I}_5$	1	0	0	4
$\mathcal{I}_6$	1	0	1	2
$\mathcal{I}_7$	1	1	0	2
$\mathcal{I}_8$	1	1	1	1

Notice that the models of $\phi = (A \vee B) \rightarrow (B \vee C)$ are all but $\mathcal{I}_5$. therefore the weight of the fomrula is the total weight (i.e., the weight of $\top$) minus the weight of $\mathcal{I}_5$. IN summary

$$w(\top) = \sum_{i=1}^{8} w(\mathcal{I}_i) = 27$$
$$w(\phi) = w(\top) - w(\mathcal{I}_5) = 23$$
$$Pr(\phi) = \frac{w(\phi)}{w(\top)} = \frac{23}{27}$$

□

Exercise 118:
Consider the following weighted formulas

$weight$	:	$literal$
1	:	A
2	:	$\neg A$
1	:	B
2	:	$\neg B$
1	:	C
2	:	$\neg C$

Compute the weight and the probabilities of the formulas

$$(A \vee B) \rightarrow (B \vee C)$$

Remember that in weighted model counting you multiply the weights of the literal that are true. **Solution**

A	B	C	w	$A \vee B \rightarrow B \vee C$
0	0	0	$2^3 = 8$	1
0	0	1	$2^2 = 4$	1
0	1	0	$2^3 = 4$	1
0	1	1	$2^1 = 2$	1
1	0	0	$2^2 = 4$	0
1	0	1	$2^1 = 2$	1
1	1	0	$2^1 = 2$	1
1	1	1	$2^0 = 1$	1

the weighted model counting of the $A \vee B \rightarrow B \vee C$ is equal to 23, and the probability is $\frac{23}{27} \approx 0.85$. □

Exercise 119:

Provide a weight function $W : \mathcal{P} \rightarrow \mathbb{R}^+$, that is equivalent to weight function defined in the previous exercise. Explain why such a weight function does not exist for the weight defined in Exercise 6.

Exercise 120:

Given the following observations on the items bought by people.

#	*Itemsets*						
4	a	b	c	d			
1		b			e	f	g
7	a	b	c				
3	a		c		e	f	
2							g
1		b			e		
4	a		c	d			g

Learn the weights of the following formulas:

(1) $a \wedge b \rightarrow c$
(2) $b \wedge c \rightarrow d \vee f$

Exercise 121:

Explain the relation between weight function and probability distribution on the set of interpretation.

Exercise 122:

Compute the weight for the following formula: $(b \rightarrow \neg c) \vee (d \leftrightarrow f)$ from the following itemset:

#	*Itemsets*					
4		b	c	d		
2	a			e	f	
6	a	b	c			
1	a		c	d	f	
3	a		c	e		g
5				d		
9		b		d	e	g

Solution We have to apply the formula for computing the weight given a set of observations, i.e.,

$$w = \ln\left(\frac{n \cdot \#SAT(\neg\phi)}{(d-n)\#SAT(\phi)}\right)$$

where n is the number of observations for which the fornula ϕ is true and d is the total number of observations. Let us first compute $\#\text{SAT}((b \rightarrow \neg c) \vee (d \leftrightarrow f))$. We do it by truth table

b	c	d	f	(	b	→	¬	c	)	∨	(	d	↔	f	)
1	1	1	1		1	0	0	1		1		1	1	1	
1	1	1	0		1	0	0	1		0		1	0	0	
1	1	0	1		1	0	0	1		0		0	0	1	
1	1	0	0		1	0	0	1		1		0	1	0	
1	0	1	1		1	1	1	0		1		1	1	1	
1	0	1	0		1	1	1	0		1		1	0	0	
1	0	0	1		1	1	1	0		1		0	0	1	
1	0	0	0		1	1	1	0		1		0	1	0	
0	1	1	1		0	1	0	1		1		1	1	1	
0	1	1	0		0	1	0	1		1		1	0	0	
0	1	0	1		0	1	0	1		1		0	0	1	
0	1	0	0		0	1	0	1		1		0	1	0	
0	0	1	1		0	1	1	0		1		1	1	1	
0	0	1	0		0	1	1	0		1		1	0	0	
0	0	0	1		0	1	1	0		1		0	0	1	
0	0	0	0		0	1	1	0		1		0	1	0	
										14					

Since we have to take into consideration also other three propositional variables not appearing in the formula we have that we have that

$$\#\text{SAT}((b \rightarrow \neg c) \vee (d \leftrightarrow f)) = 14$$
$$\#\text{SAT}(\neg((b \rightarrow \neg c) \vee (d \leftrightarrow f))) = 2^7 - \#\text{SAT}((b \rightarrow \neg c) \vee (d \leftrightarrow f)) = 16 - 14 = 2$$
$$d = 4 + 2 + 6 + 1 + 3 + 5 + 9 = 30$$
$$n = 2 + 6 + 1 + 3 + 5 + 9 = 26$$

By replacing this values in the formula we obtain

$$w = \ln\left(\frac{26 \cdot 2}{4 \cdot 14}\right) \approx \log(0.928) \approx -0.074$$

□

Exercise 123:
Prove that the formulas

$$\forall y(P(y) \wedge \exists x Q(x)) \qquad \forall y P(y) \wedge \exists x Q(x)$$

are equivalent. (Suggestion: you have to show that every interpretation satisfies the first formula if and only if it satisfies the second one).

Solution

$$\begin{aligned}
\mathcal{I} \models \forall y(P(y) \wedge \exists x Q(x)) &\iff \text{for all } d \in \Delta^{\mathcal{I}},\ \mathcal{I} \models P(x) \wedge \exists y P(y)[a_{x \leftarrow d}] \\
&\iff \text{for all } d \in \Delta^{\mathcal{I}},\ \mathcal{I} \models P(x)[a_{x \leftarrow d}] \text{ and } \mathcal{I} \models \exists y P(y)[a_{x \leftarrow d}] \\
&\iff \text{for all } d \in \Delta^{\mathcal{I}},\ \mathcal{I} \models P(x)[a_{x \leftarrow d}] \text{ and } \mathcal{I} \models \exists y P(y) \\
&\iff \mathcal{I} \models \forall x P(x) \text{ and } \mathcal{I} \models \exists y P(y) \\
&\iff \mathcal{I} \models \forall x P(x) \wedge \exists y P(y)
\end{aligned}$$

□

Exercise 124:
Let $\mathcal{P} = \{p_1, \ldots, p_n\}$ be a set of propositional variables and $\prec$ be a total order[2] on the set $\mathbb{I}$ of interpretations of $\mathcal{P}$. Consider the problem of ginding a weight function $w : \mathcal{L} \to \mathbb{R}^+$, where $\mathcal{L}$ is the set of literals on $\mathcal{P}$, such that

$$\mathcal{I} \prec \mathcal{J} \quad \text{ if and only if } \quad w(\mathcal{I}) \leq w(\mathcal{J})$$

(1) make a simple exempla with $|\mathcal{P}| = 2$,
(2) discusso on the fact if the problem has always a positive solution or not.
(3) Outline a method to find the solution.

Solution

(1) Let us consider the following two orders of the interpretations of $\{p, q\}$, where ij represents the interpretation $\mathcal{I}(p) = 1$ and $\mathcal{I}(q) = j$.

$$00 \prec 01 \prec 10 \prec 11 \tag{96}$$

$$00 \prec 11 \prec 10 \prec 01 \tag{97}$$

In the order (96) we can define $w(\neg p) = w(\neg q) = 1$ and $w(p) = 3$, and $w(q) = 2$, we obtain that

$$w(00) = 1 < w(01) = 2 < w(10) = 3 < w(11) = 6$$

In the order (97) instead we can write the following system of inequalities:

$$\begin{cases} w(00) < w(10) \\ w(11) < w(01) \\ w(00) < w(11) \end{cases}$$

[2]A total order $\prec$ on a set S is a binary relation on S such that (a) $s \not\prec s$, (b) if $s \neq t$ then $s \prec t$ or $t \prec s$, and (c) $s \prec t \prec u$ implies $s \prec u$. An example of total order is the usual order $<$ on integers.

which can be rewritten as

$$\begin{cases} w(\neg p)w(\neg q) < w(p)w(\neg q) \\ w(p)w(q) < w(\neg p)w(q) \end{cases} \Rightarrow \begin{cases} w(\neg p) < w(p) \\ w(p) < w(\neg p) \end{cases}$$

which does not have any solution.

(2) From the previous example one can see that the problem does not always have a solution.

(3) A method for solving this problem is to writhe explicitly the system of inequalities for every $\mathcal{I} \prec \mathcal{J}$

$$\prod_{i=1}^{n} w(p_i)^{\mathcal{I}(p_i)} \cdot w(\neg p_i)^{\mathcal{I}(\neg p_i)} < \prod_{i=1}^{n} w(p_i)^{\mathcal{J}(p_i)} \cdot w(\neg p_i)^{\mathcal{J}(\neg p_i)}$$

and try to solve it.

□

Exercise 125:
Compute the weighted model counting of the formula

$$(A \to B) \wedge (B \to (C \vee D))$$

via knowledge compilation with the following weight function:

lit	A	$\neg A$	B	$\neg B$	C	$\neg C$	D	$\neg D$
$w(lit)$	3	1	1	3	3	0.5	4	2

You can check your result by computing WMC using truth table (this is not strictly necessary for the exercize).

Solution We compute the WMC of the formula Φ by compiling it in the sd-DNNF form and then transform it in a computational circuit

$$\begin{aligned}
(A \to B) \wedge (B \to (C \vee D)) & \quad \text{to NNF} \\
(\neg A \vee B) \wedge (\neg B \vee C \vee D) & \quad \text{to DNNF with shannon expansion on } B \\
(B \wedge (C \vee D)) \vee (\neg B \wedge \neg A) & \quad \text{to d-DNNF} \\
(B \wedge (C \vee (\neg C \wedge D))) \vee (\neg B \wedge \neg A) & \quad \text{to sd-DNNF} \\
(B \wedge (C \wedge (D \vee \neg D) \vee (\neg C \wedge D)) \wedge (A \vee \neg A)) \vee & \\
(\neg B \wedge \neg A \wedge (C \vee \neg C) \wedge (D \vee \neg D)) & \quad \text{To circuit} \\
(1 \cdot (3 \cdot (4 + 2) + (0.5 \cdot 4)) \cdot (3 + 1)) + & \\
(3 \cdot 1 \cdot (3 + 0.5) \cdot (4 + 2)) = & \\
80 + 63 = 143 &
\end{aligned}$$

□

CHAPTER 8

First Order Logic

1. Introduction

First-order logic can be understood as an extension of propositional logic. In propositional logic the atomic formulas have no internal structure—they are propositional variables that are either true or false. In First-order logic the atomic formulas assert a property of an element of the domain of interest or the existence of a relationship among multiple elements. To emphasise this fact First-Order Logic is also called *predicate logic*. First-order logic, though not as expressive as full spoken language such as English or Italian, introduces a more structured syntax to express propositions. In particular, it allows us to express the fact that a certain object has some property by introducing symbols for objects and symbols for properties, and a way to combine the two in order to obtain the proposition that states that the property holds for the object. FOL not only allow to state property of objects but also relationships between objects as well as functions on objects. Form the first-order logic perspective a "world" is described as a set of objects (aka domain, or universe) and a set of functions and relations on these objects.

To understand what we gain by using FOL w.r.t .propositional logic consider this simple example.

EXAMPLE 8.1. *Suppose that we want to express the four propositions expressed by the following four English statements:*

(1) Mary is a person
(2) John is a person
(3) Mary is mortal
(4) Mary and John are siblings

Since the four statements express four different propositions, in propositional logic, we introduce four different propositional variables, say p, q, r, and s, with the following intuitive meaning.

- *p that stands for Mary is a person*
- *q that stands for John is a person*
- *r that stands for Mary is mortal*
- *s that stands for Mary and John are siblings*

By doing so, we lose much information about the relationship of the four propositions. For instance, we don't represent the fact that proposition (1) and (3) are about the same person, namely Mary; that proposition (1) and (2) states the same property about two individual; that proposition (4) states a relationship between the two individuals who are involved in propositions (1) and (2). All this relationship between propositions is lost since propositional variables are atomic, i.e., they are not built of simpler components.

The language of FOL instead introduces symbols to describe the entities and symbols for describing properties. For instance two symbols, e.g., M and J are introduced in order to denote the two entities John and Mary, and a symbol P is introduced to denote the property of being a person. Using these symbols in FOL we can build the formula $P(M)$ that represents the proposition that "the property P holds for the entity M" and the formula $P(J)$ that represents the proposition "the property P holds for the entity J" In a similar fashion FOL introduces a symbol S to represent the sibling relation and uses the formula $S(J, M)$ to state the property that John and Mary are siblings.

Entities of a domain can be specified by providing a direct name, e.g., John, Mary, ... but we can also refer to an entity in terms of how it relates to another entity. For instance, even if we don't have a specific symbol to denote the father of Mary, in natural language we can refer to him with the *definite description* "the father of Mary". In this case "the father of" is a construct that allows building the description of an entity starting from a description of another entity. We can apply this construction as many times as we want, for instance obtaining definite descriptions such as "the father of the father of Mary". The language of first-order logic introduces symbols to denote these constructors of description, they are called *functional symbols*. For instance, if F is a functional symbol corresponding to the definite description constructor "the father of", in FOL we can build the formulas $P(F(A))$ that states that the father of Alice is a Person.

Not only FOL allows to have expressions to denote specific objects, FOL also allows a set of symbols to denot any (not specific) object. This is similar to the work "object" or "entity" or "thing" or "element" in english. This are very generic terms that can be used to denote any specific element. They are different from proper names since by composing a proer name with a predicate you obtain a proposition, like in "John is a person", instead by composing a variables with a predicate you obtain "the object is a person" which is not a proposition, since we don't know to which objects it refers to. Individual variables are important since they are necessary to introduce quantification. This is similar to what happens in englis when we use "somebody" or "everybody", and "something" or "everything" Every individual variable x can be *quantified* either universlly (like in "everybody") or existentially (like in somebody) So we can say "somebody is a person" or "everybody is a person" or "every obejct is a person". For this purpose the language of FOL provides two new logical symbols (in addition to the boolean connectives) called *quantifiers* $\forall x$ and $\exists x$ for every individual variable x that stands for "for all individual" and "there is an indivudual".

In summary, while propositional logic describe a world in terms of a set of propositions which are true and false, in FOL describe the worldss in terms of a set of objects, which constitute the so called *interpretation domain* and a set of properties and relationship between them.

2. Syntax of FOL

The language of FOL logic is defined relative to a signature. A signature Σ consists of a set of constant symbols $c_1, c_2, \ldots$, a set of function symbols $f_1, f_2, \ldots$ and a set of predicate symbols $p_1, p_2, \ldots$. Each function and predicate symbol has an arity $k > 0$. We will often refer to predicates as relations. Predicates and function with arity equal to k are called k-ary predicate and k-ary functions.

We also suppose that Σ contains an infinite set of variables $x_0, x_1, \ldots$. To denote variables we also use the last letters of the alphabet x,y, ..., possibly with indices. For constants instead we use the first letters of the alphabet a, b, ... possibly with indices.

Not every seq1uence of symbols in Σ are legal expressioons. In FOL we provide a grammar for two types of expressions called *terms*, which are used to denote objects in the domain of interests, and *formulas*, which are intended to denote propositions about the domain of interests, which can be either true or false.

2.1. Terms in FOL. Terms are expression built starting from constants and individual variables, by composing them with funciton symbols. As clarified above they are descriptions of the objects of the domain of interest. We provide an inductive definition of the set of terms for a signaturee Σ.

DEFINITION 8.1. *Given a signature Σ, the set of terms (more precisely Σ-terms) is defined by the following set:*

- *a constant c_i is a term*
- *a variable x_i is a term*
- *if $t_1, \ldots, t_k$ are terms and f is a k-ary function symbol, then $f(t_1, \ldots, t_k)$ is a term*
- *nothing else is a term.*

Intuitively terms are used to denote objects in the domain of the world we want to describe with our FOL language

EXAMPLE 8.2 (Terms).

- *x_i: denotes a generic object of the domain; Variables are supposed to range over all the objects of the domain and they can be instantiated with any one of them. This is why they are also referred as "individual variables", to keep them distinguished from "propositional variables" which instead ranges over propositions.*
- *c_i; a constant denotes one specific element of the domain.*
- *$f_i(x_j, c_k)$; complex term. This is similar to what we say in English when we refer to a person in terms of "functions" of another preson, e.g., "the father of John", the person that stands between John and Mary*
- *$f(g(x, y), h(x, y, z), y)$; a more complex terms. e.g. the father of the person that stands between John and Mary*

A term is *ground* or *closed* if it does not contain individual variables. Ground terms are descriptions of some specific object in the domain. If there are no constant symbol in the siguature then there is no ground terms. If there are no function symbol then the set of ground terms coincides with the set of constants. If we have at least one constant and one function symbol then the set of ground terms is infinite. For instance if we have the constant Mary and the function motherOf then

we have the following infinite set of terms:

$$\begin{aligned} &\mathsf{Mary}\\ &\mathsf{motherOf(Mary)}\\ &\mathsf{motherOf(motherOf(Mary))}\\ &\mathsf{motherOf(motherOf(motherOf(Mary)))}\\ &\dots \end{aligned}$$

2.2. Formulas in FOL. The simplest FOL expression that specifies a proposition is called *atomic formula* and it states that a relation espressed by an n-ary predicate[1] p holds for an n-tuple of objects specified by the terms t_1, t_2,..., t_n.

DEFINITION 8.2 (Atomic formula). *An atomic formula on a signature Σ is an expression of the form $p(t_1, \dots, t_n)$ there p is an n-ary predicate of Σ and t_i are Σ terms.*

Roughly speaking atomic formulas in FOL correspond to propositional variables in propositional logic. We will make this correspondence more precise later. For now it is enough to think that the atomic formula $Happy(John)$ (where "John" is a constant symbol and "Happy" is a 1-ary (unary) predicate, corresponds to a propositional variables that we would have introduced in propositional logic to formalize the proposition that John is happy. In fact, sometimes in propositional logic we introduce the propositional variable "Happy(John)", but in spite of the fact that it looks the same as the first order atomic formula, it should be considered as a string and not as a structured object.

As in propositional logic complex formulas can be built starting from atomic formulas;

DEFINITION 8.3 (First order formulas). *A (first order) formula on a signature Σ is defined as follows:*

- *an atomic formula is a formula;*
- *if ϕ and ψ are formulas then $\neg\phi$, $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \to \psi$, and $\phi \leftrightarrow \psi$ are formulas;*
- *if ϕ is a formula and x and individual variable, the $\forall x.\phi$ and $\exists x.\phi$ are formulas;*
- *Nothing else is a formula.*

Notice that the definition of first order logic formula extends the definition of formula in propositional logic by adding a new rules for the operator $\forall x$ and $\exists x$. These operators are called *universal* and *existential* quantifier, respectively . The intuitive reading fo $\forall x.\phi$ is "for every x ϕ" and the intuitive reading of $\exists x\phi$ is "there is an x such as ϕ.

EXAMPLE 8.3. *Consider a signature with a single 2-ary (binary) relation symbol R. Since there are no constant symbols or function symbols, the only terms are variables x_1, x_2, ... The set of atomic formulas contains $R(x, x)$, $R(x, y)$,*

[1]An n-ary predicate, where n is a natural number, is a predicate with arity equal to n a predicate with arity equal to n.

$R(x_i, x_j)$ *Examples of formulas are*

$$\forall x.\forall y.\forall z.(R(x,y) \wedge R(y,z) \rightarrow R(x,z))$$
$$\forall x, \forall y.\exists z.(R(x,y) \rightarrow R(x,z) \wedge R(z,y)$$

This formulas expresse that R is a transitive relation and a dense relation, respectively.

Consider a signature with a constant symbol 0, unary function symbol s, and unary predicate symbol E. Terms over this signature include x, 0, $s(0)$, $s(x)$, $s(s(0))$, $s(s(s \dots s(0}$ $s(s(s \dots s(x) \dots)) \dots$. Atomic formulas are obtained by applying the unary predicate E to these terms, e.g. $E(x)$, $E(0)$, $E(s(0))$, $E(s(x))$, $E(s(s(0)))$, $E(s(s(s \dots s(0))$, $E(s(s(s \dots s(x) \dots)))$ $\dots$. An example of (non atomic) formula is the following:

$$E(0) \wedge \forall x(E(x) \leftrightarrow \neg E(s(x))$$

Sometimes we write function symbols and predicate symbols infix to improve readability:

Example 8.4. *Consider a signature with a constant symbol* 1*, binary function symbol* $+$*, and a binary relation symbol* $<$*, both written infix. Then* $x + 1$ *is a term (that is the infix notation of* $+(x, 1)$*) and* $\forall x.(x < (y + 1))$ *is a formula (corresponding to the infix notation of* $\forall x.(< (x, +(x, 1))))$*.*

As form propositional logic we should also clarify in case of ambiguity (because we miss the parentesis) what are the precedence of the quantifier w.r.t, connectives. We have that $\forall x$ and $\exists x$ have the same priority of $\neg$. This implies that they should be applied before all the other connectives.

Quantifiers add expressive power to first order logic w.r.t propositional logic, as they allow to state in a single formula a fact that holds for all the induvidual of the domain, without eplicitly enumerating the property of each of them. Notice that this is possible also when the domain contains an infinite set of objects (e.g., the natural numbers). In this case it would be impossible to write a propositional formula since it would have infinite length.

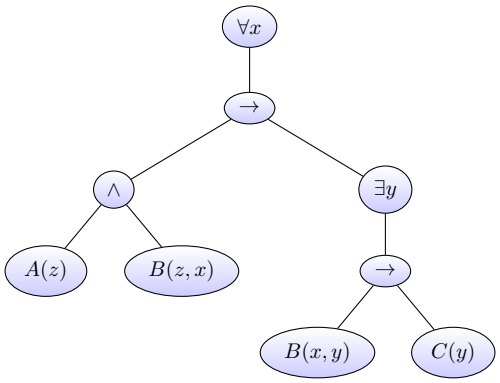
Example 8.5. *Consider a signature with a constant symbol* 0*, unary function symbol* s*, and unary predicate symbol* E*. As we have seen in the previous example with this signature we can build an infinite set of terms, wich might denote an infinite set of objects (e.g, the natural numbers). To express the same fact of the previous example, without the help of universal quantifier we should have written an infinite formula*

$$E(0) \wedge (E(0) \leftrightarrow \neg E(s(0))) \wedge (E(s(0)) \leftrightarrow \neg E(s(s(0)))) \wedge (E(s(s(0))) \leftrightarrow \neg E(s(s(s(0))))) \wedge \dots.$$

which is not possible in propositional logic.

For every first order formula ϕ we can define the formula tree that describe the structure of the formula. The formula tree expresses how the formula has been built using the rules of Definition 8.3. The definition is analogous to the formula tree for propositional formulas, with the addition of the universal and existential quantifier.

Example 8.6. *The tree of the formula* $\forall x(A(c) \wedge B(c, x) \rightarrow \exists y(B(x, y) \vee C(y)))$



2.3. Free and bound variables. If $\forall x\psi$ (resp. $\exists x\psi$) is a subformula of a formula ϕ, we way that ϕ is *the scope* of the that specific occurrenc of the quantifier $\forall x$ (resp. $\exists x$). Notice that the scope is defined for every occurrence of a quantifier, and that different occurrences of the same quantifier have different scope. For instance the scope of the forst occurrence of $\forall x$ in $\forall x P(x) \wedge \forall x(Q(x) \wedge \exists y.R(y))$ is $P(x)$, and the scope of the second occurrence of $\forall x$, is $Q(x) \wedge \exists y.R(y))$.

The notion of scope of the occurrence of a quantifier becomes rather clear if you think to the formula tree of a formula ϕ. Indeed an occurrence of a quantifier $\forall x$ of a formula ϕ corresponds to a node n of the tree of the formula ϕ labelled with $\forall x$; its scope is the sub-tree rooted at the only child of n.

Definition 8.4. *The occurrence of a variable x in a formula ϕ is* free *if it does not occour in the scope of a quantifier $\forall x$ or $\exists x$. A variable x is free in ϕ if there is at least one occurrence of x in ϕ that is free.*

Proposition 8.1. *Let $FV(\phi)$ denotes the set of free variables of ϕ, then*

(1) $FV(P(t_1, \ldots, t_n))$ is the set of variables that occour in some t_i;
(2) $FV(\neg\phi) = FV(\phi)$;
(3) $FV(\phi \circ \psi) = FV(\phi) \cup FV(\psi)$ for every connective $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$
(4) $FV(Qx.\phi) = -FV(\phi) \setminus \{x\}$ for $Q \in \{\forall, \exists\}$

If we want to make explicit the set of free variables of a formula ϕ, we write $\phi(x_1, \ldots, x_n)$ where $\mathrm{FV}(\phi) = \{x_1, \ldots, x_n)$. This is just notation; it should not interpreted as we add the expression $(x_1, \ldots, x_n)$ at the end of the formula. Actually, $\phi(x_1, \ldots, x_n)$ is the same formula as ϕ, they are two different way to denotes the same formula. In the second method we make explicit that in $\phi(x_1, \ldots, x_n)$ we made eplicit the set of free variables.

Example 8.7. *consider the formula*

$$\forall x(P(x, y) \wedge \exists z P(x, z) \rightarrow \exists x(Q(x, y, z, w))) \tag{98}$$

From the previous examples, it should be clear that a quantifier can occour also in the scope of another quantifier. We should clarify whaT happens when a quantifire of a variable x occours in the scope of another occurrence of a quantifier of the same wariable. Who has the precedence on the variable?

The occurrence of a variable x is *bounded* by a quantifier Qx (either $\forall x$ or $\exists x$) is the first node labelled with Qx, that occurs in the path from the leaf of the occurrence of x to the root of the formula tree. If there is no such a node we say that the occurrence of x is unbound or *free*.

EXAMPLE 8.8. *In the previous example, the occurrence of x in the nodes $B(z,x)$ and $B(x,y)$ is the root node $\forall x$; the occurrence of y in $B(x,y)$ is the node $\exists y$. Instead, the occurrences of z in $A(z)$ and $B(z,y)$ are unbound.*

An occurrence of a variable x in a formula ϕ is bound if that occurrence is bound by some quantifier. An occurrence that is not bound is said to be free. Note that different occurrences of the same variable in a given formula can be both bound and free, e.g., variable x occurs both bound and free in the formula $P(x) \wedge \exists x P(x)$. A formula with no free variables is said to be *closed* or a *sentence*. The formulas. If a formula ϕ contains a free occurrence of a variable x is denoted by $\phi(x)$; if ϕ contains at least one free occurrence for each variable $x_1 \ldots, x_n$ we write $\phi(x_1, \ldots, x_n)$.

EXAMPLE 8.9. *In the formula $\forall x(Q(x,y) \to R(x,y))$ the occurence of y is free while the occurence of x is bound, therefore y is free while x is bound. In the formula $\forall x(Q(x,y) \to \exists y R(x,y))$ the occurrence of y in $Q(x,y)$ is free while the occurence of y in $R(x,y)$ is bound. The two occurrences of x are bound. Therefore, the variable x is bound while the variable y is both free.*

2.4. Substitution of variables with terms. One of the most frequent operation that we have to do in a formula $\phi(x_1, \ldots, x_n)$ that contains the free variables $x_1, \ldots, x_n$ is to replace one, some, or all free variables with terms. This iperation intuitively means that you instantiate the variable, which intuitively denotes any element of the domain, with one specific element described by the corresponding term. The terms which replace variables can contains other variables, and as we will see later this might creeate some problem.

DEFINITION 8.5. *A* substitution *σ is a function that assigns to every individual variable x a term t. A substitution that replaces $x_1, \ldots, x_n$ with $t_1, \ldots, t_n$ respectively and leave every other variables unchanged is denoted by $x_1/t_1, \ldots, x_n/t_n$. The result of applying the substitution $x_1/t_1, \ldots, x_n/t_n$ to a term $t(x_1, \ldots, x_n)$ or to a formula $\phi(x_1, \ldots, x_n)$, denoted by $t(x_1, \ldots, x_n)[\sigma]$ and $\phi(x_1, \ldots, x_n)[\sigma]$ is the term or the formula obtained by replacing simultaneusly every free occurrence of every x_i, with t_i if this occurrence doees not occour in the scope of a quantifier of a variable of t_i.*

EXAMPLE 8.10. *Let $\phi(x,y) = R(x,y) \to \exists z R(y,z)$ and $\sigma = [x/a, y/f(z)]$ be a substitution. The application of σ to ϕ is $\phi(a, f(z))$ is $R(a, f(z)) \to \exists z R(y,z)$. Notice that the second occurrence of y in ϕ is not substituted because it occurs in the scope of the quantifier $\exists z$ and z occurs in the terms for which y is replaced.*

Some remarks are in order. The application of a substitution $x_1/t_1, \ldots, x_n/t_n$ to a term or a formula, should be done simultaneusly. This means that if some t_i contains a variable x_j this variable should not be substitute. For instance the application of the substitution $x/a, y/x$ to $P(x,y)$, i.e., $P(x,y)[x/a, y/x]$ is $P(a,x)$ and not $P(a,a)$.

The second remarks concerns the condition given at the end of Definition 8.5. According to this condition, we cannot replace an occurrence of x with a term $t(y)$ if x occurs in the scope of a quantifier on y. For instance the substitution $x/f(y)$ in the formula $P(x,y) \wedge \exists y R(x,y)$, is equal to $P(f(y),y) \wedge \exists y R(x,y)$. Notice that the occurrence of y in the scope of the existential quantifier $\exists y$ is not affected by the substitution, since y occurs in $f(y)$. For an intuiition on why this restriction is important consider the following example

EXAMPLE 8.11. *To represent in FOL any person x does not like some piece of music*

$$\forall x(\mathsf{person}(x) \to \exists y(\mathsf{music}(y) \wedge \neg \mathsf{likes}(x, y))) \tag{99}$$

If we believe that (99) *is true, then, intuitively, if we replace the variable x with any term t denoting some person, we should also believe in the result of the substitution. Suppose that the language contains the function symbols* composer *such that* author(x) *denotes the composer of apiece of music x. Consider the substitution $\sigma = x/\mathsf{composer}(y)$. Notice that there is one occurrence of x in* (99) *that is in the scope of a quantifier of y, which is a variable in the term by which x is replaced. If in the application of σ to* (99) *we ignore the condition given at the end of Definition 8.5 we obtain the formula*

$$\exists y(\mathsf{music}(y) \wedge \neg \mathsf{likes}(\mathsf{composer}(y), y))$$

which is a closed formula that states there is some piece of music which does not like to his/her composer. However, intuitively, this does not logically follows from (99).

DEFINITION 8.6. *A term t is free for x in ψ if x does not occour in the scope of some quantifier of a variable of t.*

From now onw we will consider only substitutions $[x_i/t_1, \ldots, x_n/t_n]$ for which t_i is free for x_i in ϕ for every $1 \leq i \leq n$.

Finally, we introduce an additional notation, which is rather intuitive for expressing the result of the application of a substitution $x_1/t_1, \ldots, x_n/t_n$ to a term $t(x_1, \ldots, x_n)$ or to a formula $\phi(x_1, \ldots, x_n)$. When the contest is clear, we replace the long notations $t(x_1, \ldots, x_n)[x_1/t_1, \ldots, x_n/t_n]$ and $\phi(x_1, \ldots, x_n)[x_1/t_1, \ldots, x_n/t_n]$, with $t(t_1, \ldots, t_n)$ and $\phi(t_1, \ldots, t_n)$.

3. Semantics of first order logic

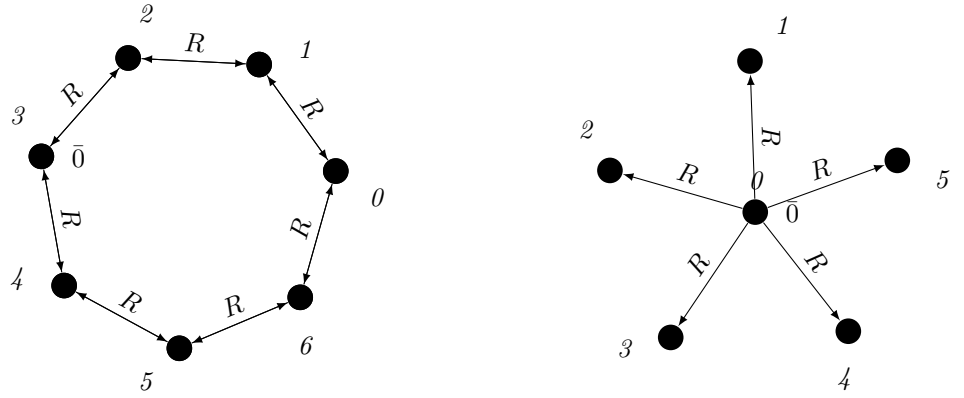
The semantics of a logic describes how the symbols and the expressions of such a logic can be interpreted in a mathematical structura that is intended to formalize a state of the world. The semantics of propositional language is just an assignment to propositional variables that state what is true and wat is false. The semantics of first order logic is more complex and is based on the notion of Σ-structure. The semantics of order logic with signature Σ is given in terms of a mathematical structure, called Σ-structure, or equivalently a Σ-interpretation.

DEFINITION 8.7 (Σ-interpretation). *A Σ-interpretation $\mathcal{I}$, or an interpretation of Σ, consistes of:*

- *A non-empty set $\Delta_{\mathcal{I}}$ called the universe or the domain of the structure $\mathcal{I}$;*
- *for each constant symbol c, an element $c^{\mathcal{I}} \in \Delta_{\mathcal{I}}$;*
- *for each k-ary function symbol f in a k-ary function, $f^{\mathcal{I}} : \Delta_{\mathcal{I}}^k \longrightarrow \Delta_{\mathcal{I}}$;*
- *for each k-ary predicate symbol P in a k-ary relation $P^{\mathcal{I}} \subseteq \Delta_{\mathcal{I}}^k$*

where $\Delta_{\mathcal{I}}^k$ denotes the set of k-tuple of elements in $\Delta_{\mathcal{I}}$.

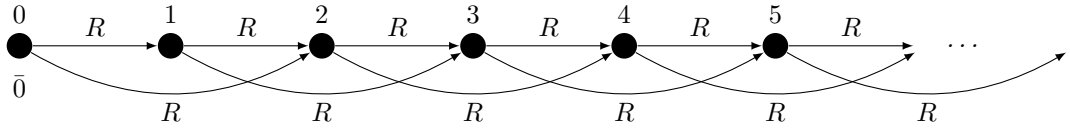
EXAMPLE 8.12. *Consider the signature $\Sigma = (\bar{0}, R)$ that contains only one cnstant symbol $\bar{0}$ and the binary predicate R Three examples of structures for this signature are shown in the following picture*



The domain of the Σ-structure on the left is equal to $\{0,1,2,3,4,5,6\}$ the constant $\bar{0}$ is interpreted in 3 and the predicate symbol R is interpreted in the set of pairs $\{(0,1),(1,0),(1,2),(2,1),\dots(5,6),(6,5),(6,1),(1,6)\}$. Notice the difference between $\bar{0}$ which is an element of Σ and the element 0 which is an element of the doain of the Σ-structure.

On the right we hav a Σ-structure which domain is $\{0,1,2,3,4,5,6,7\}$ the constant $\bar{0}$ is interpreted in 0 and the prtedicate symbol R is interpreted in the set of pairs $\{(0,1),(0,2),\dots,(0,7)\}$. The above example are finite *structures since the interpretation domain is finite, i.e., it conteins a finite number of elements.*

It is also possible to have infinite Σ-structures. For instance the following structure is infinite as it contains an infinite set of objects.



The domain of the above structure is the set of natural numbers $\mathbb{N} = \{0,1,2,3,4,\dots\}$, the constant $\bar{0}$ is interpreted in 0 and the relation R is the set of pairs $\{(n, n+1) \mid n \in \mathbb{N}\}$

Well studied structures in First order logic and in mathematics are $(\mathbb{N}, 0, <)$ wehre $\mathbb{N}$ is the set of natural numbers $<$ is the order relation between the natural numbers and 0 is the smalles natural number. This structure is very similar to the one shown above

EXAMPLE 8.13. *An undirected graph can be considered as a Σ-structure for the signature that contains the binary relation symbol E (for edge), where E is interpreted as the edge relation. For example, the graph shown in Figure1 can be represented by a structure $\mathcal{I}$ with universe $\Delta_{\mathcal{I}} = \{1,2,3,4\}$ and $E^{\mathcal{I}}$ is the irreflexive symmetric binary relation*

$$E^{\mathcal{I}} = \{(1,2),(2,3),(3,4),(4,1),(2,1),(3,2),(4,3),(1,4),(1,3),(3,1)\}$$

EXAMPLE 8.14. *TODO: Add an example of a sigma structure that contains also constants and functions*

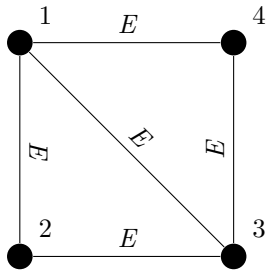


FIGURE 1.

The terms and the formulas that can be constructed from the signature Σ using the rules of First Order Logic are interpreted in a Σ-structure. However, a Σ-structure provides the "meaning" of all the symbols in Σ with the exception of the variables (which are not in Σ). Since terms and formulas contains also variable, in order to interpret terms and formulas in a Σ structure we also have to provide an assignment to the individual varialbes.

On could ask why the interpretation of individual variables is kept separated form the interpretation of the symbols in Σ. There is no technical reason, indeed to keed them separated, and in some textbook you can find that an interpretiaton also interpret the variabees. The main reason is that, in order to inverpret the unversa and existential quanfiers on a variable x we have to consider all (or some) instantiation of x, so we want to allow to modify the interpretation of variables in order to provide the meaning of the quantified formulas.

DEFINITION 8.8. *Given an interpretation $\mathcal{I}$ for the signature Σ (or equivalently a Σ-structure $\mathcal{I}$), an* assignment *is a function $a : X \to \Delta_{\mathcal{I}}$ form the set of individual variables X to the interpretation domain $\Delta_{\mathcal{I}}$.*

Given a Σ-structure $\mathcal{I}$ and an assignment a to the individual variables in $\Delta_{\mathcal{I}}$ we are now ready to define how terms and formulas are interpreted in $\mathcal{I}$.

DEFINITION 8.9. *Let $\mathcal{I}$ be a Σ-structure and a an assignment for the individual variables in $\Delta_{\mathcal{I}}$ the interpretation of a term t in $\mathcal{I}$ under the assignment a, denoted by $t^{\mathcal{I}}[a]$ is defined as follows:*

- *$c^{\mathcal{I}}[a] = c^{\mathcal{I}}$ for every constant symbol c;*
- *$x^{\mathcal{I}}[a] = a(x)$ for every individual variable symbol x;*
- *$f(t_1, \ldots, t_n)[a] = f^{\mathcal{I}}(t_1^{\mathcal{I}}[a], \ldots, t_n^{\mathcal{I}}[a])$ for every n-ary functional symbol n and n terms $t_1, \ldots, t_n$.*

Notice that the interpretation of a term, i.e., $t^{\mathcal{I}}[a]$ is "context free", i.e., the interpretation of a term t is independent from the context where it occurs.

REMARK 1. *The interpretation of a term w.r.t. an assignment a depends only on the value that a assigns to the variables that occurs in t. More formally, if a and a' agree on the assignment to the individual variables occurring in t (and might disagree on the assignment to other variables), then $t^{\mathcal{I}}[a] = t^{\mathcal{I}}[a']$.*

The next step is to provide a definition of when a formula is true or false in a Σ-structure $\mathcal{I}$ with respect to the assignment a. We start by defining wen an

atomic formula $P(t_1, \ldots, t_n)$ is true in $\mathcal{I}$ w.r.t., the assignment a, in symbols:

$$\mathcal{I} \models P(t_1, \ldots, t_n)[a]$$

DEFINITION 8.10. *An interprertation $\mathcal{I}$ satisfies (makes true) the atomic formula $P(t_1, \ldots, t_n)$ w.r.t. the assignment a, in symbols $\mathcal{I} \models P(t_1, \ldots, t_n)[a]$, if the n-tuple of elements of $\Delta_{\mathcal{I}}$ obtained by interpreting each t_i belongs to $P^{\mathcal{I}}$. In symbols:*

$$\mathcal{I} \models P(t_1, \ldots, t_n)[a] \quad \textit{if} \quad (t_1^{\mathcal{I}}[a], \ldots, t_n^{\mathcal{I}}[a]) \in P^{\mathcal{I}}$$

When Σ contains the equality binary predicate $=$, then it's interpretation is always the same and fixed to the set of pairs (d, d) with $d \in \Delta_{\mathcal{I}}$. As a consequence we have that $\mathcal{I} \models t_1 = t_2$ if and only if $t_1^{\mathcal{I}}[a]$ is equal to $t_2^{\mathcal{I}}[a]$.

In the next definition we use the notation $a_{x \mapsto d}$ for an assignment a an individual variable x and a domain element d, to denote the assignment obtained by modifying a so that $a(x) = d$.

DEFINITION 8.11 (Satisfiability of a formula w.r.t. an assignment). *An interpretation $\mathcal{I}$ satisfies a complex formula ϕ w.r.t. the assignment a according to the following rules:*

$$\begin{aligned}
\mathcal{I} \models \phi \wedge \psi[a] &\quad \textit{iff} \quad \mathcal{I} \models \phi[a] \textit{ and } \mathcal{I} \models \psi[a] \\
\mathcal{I} \models \phi \vee \psi[a] &\quad \textit{iff} \quad \mathcal{I} \models \phi[a] \textit{ or } \mathcal{I} \models \psi[a] \\
\mathcal{I} \models \phi \to \psi[a] &\quad \textit{iff} \quad \mathcal{I} \not\models \phi[a] \textit{ or } \mathcal{I} \models \psi[a] \\
\mathcal{I} \models \neg\phi[a] &\quad \textit{iff} \quad \mathcal{I} \not\models \phi[a] \\
\mathcal{I} \models \phi \equiv \psi[a] &\quad \textit{iff} \quad \mathcal{I} \models \phi[a] \textit{ iff } \mathcal{I} \models \psi[a] \\
\mathcal{I} \models \exists x \phi[a] &\quad \textit{iff} \quad \textit{there is a } d \in \Delta_{\mathcal{I}} \textit{ such that } \mathcal{I} \models \phi[a_{x \leftarrow d}] \\
\mathcal{I} \models \forall x \phi[a] &\quad \textit{iff} \quad \textit{for all } d \in \Delta_{\mathcal{I}}, \mathcal{I} \models \phi[a_{x \leftarrow d}]
\end{aligned}$$

When ϕ is a closed formula then $\mathcal{I} \models \phi[a]$ iff $\mathcal{I} \models \phi[a']$ for any assignment a', therefore the assignment does not play any role and we simplify the notaiton with $\mathcal{I} \models \phi$ skipping the assignment.

The notation $\phi[a]$ is very similar to the notation $\phi[\sigma]$ used for substitution where σ is a substitution of variables with terms. Though they are related, they should not be confused. a and σ are different concepts, since a maps variables into elmeents of the domain $\Delta_{\mathcal{I}}$, while σ maps variables in terms in the signature Σ. While $\phi[\sigma]$ is a formula, obtained by replacing each free occurrence of a variable x with $\sigma(x)$, $\phi[a]$ is just a formula ϕ and an assignmennt a, no transformation is einvolved in $\phi[a]$. One should not think that $\phi[a]$ is a formula obtained by replacing any variable x with $a(x) \in \Delta^{\mathcal{I}}$. Indeed the result of this replacement is not a formula since $\alpha[x)$ is en elment of the domian and not of the signature.

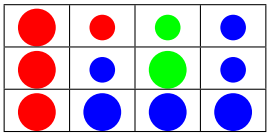
However there is a connection between the two expressions. This is stated by the followin proposition

PROPOSITION 8.2. *If $t^{\mathcal{I}}[a] = d$*

(1) for every term s, $s(x)^{\mathcal{I}}[a_{x \leftarrow d}] = s(t)^{\mathcal{I}}[a]$.
(2) then $\mathcal{I} \models \phi(t)$ iff $\mathcal{I} \models \phi(x)[a_{x \leftarrow d}]$.

PROOF. The proof is by induction on the terms and the formulas. □

EXAMPLE 8.15. *Consider*[2] *structures represented by a rectangular grid of "dots" Each dot has a color (red, blue, or green) and a size (small or large). An example of such a structure is shown in the following picture*



The logical language we use to describe our dot world has unary predicates red *,* green*,* blue*,* small *and* large*, which are interpreted in the obvious ways. The binary predicate* adj *is interpreted in the adiacency relation that contains the pairs of dots* (d, d') *contained in two slots with a wall in common. The binary predicates* sameColor*,* sameSize*,* sameRow*, and* sameColumn *are interpreted in the relation that contains the pairs of dots* (d, d') *that have the same color, the same size, they are on the same row, or on the same column respectively. Finally, the binary predicate* leftOf *is interpreted in the set of pair of dots* (d, d') *sich that* d *is left of* d' *regardless of what rows the dots are in. The interpretations of* rightOf*, above, and* below *are similar.*

Consider the following sentences:

(1) $\forall x(\mathsf{green}(x) \vee \mathsf{blue}(x))$
(2) $\exists x, y(\mathsf{adj}(x, y) \wedge \mathsf{green}(x) \wedge \mathsf{green}(y))$
(3) $\exists x((\exists z \mathsf{rightOf}(z, x)) \wedge (\forall y(\mathsf{leftOf}(x, y) \rightarrow \mathsf{blue}(y) \vee \mathsf{small}(y))))$
(4) $\forall x(\mathsf{large}(x) \rightarrow \exists y(\mathsf{small}(y) \wedge \mathsf{adj}(x, y)))$
(5) $\forall x(\mathsf{green}(x) \rightarrow \exists y(\mathsf{sameRow}(x, y) \wedge \mathsf{blue}(y)))$
(6) $\forall x, y(\mathsf{sameRow}(x, y) \wedge \mathsf{sameColumn}(x, y) \rightarrow x = y)$
(7) $\exists x \forall y(\mathsf{adj}(x, y) \rightarrow \neg \mathsf{sameSize}(x, y))$
(8) $\forall x \exists y(\mathsf{adj}(x, y) \wedge \mathsf{sameColor}(x, y))$
(9) $\exists y \forall x(\mathsf{adj}(x, y) \wedge \mathsf{sameColor}(x, y))$
(10) $\exists x(\mathsf{blue}(x) \wedge \exists y(\mathsf{green}(y) \wedge \mathsf{above}(x, y)))$

We can evaluate them in the model shown above: There they have the following truth values:

(1) false
(2) true
(3) true
(4) false
(5) true
(6) true
(7) false
(8) true
(9) false
(10) true

For each sentence, see if you can find a model that makes the sentence true, and another that makes it false. For an extra challenge, try to make all of the sentences true simultaneously. Notice that you can use any number of rows and any number of columns.

[2] `https://leanprover.github.io/logic_and_proof/semantics_of_first_order_logic.html`

REMARK 2. *Notice that, t does not contain any variable then in evaluating $t^{\mathcal{I}}[a]$ the assignmnet a does not play any role. Indeet it is easy to see that $t^{\mathcal{I}}[a] = t^{\mathcal{I}}[a']$ for every pair of assignments a and a'. We call such a term,* ground *term, and since we don't need the assignment to variables to evaluate it, we simplify $t^{\mathcal{I}}[a]$ with $t^{\mathcal{I}}$.*

REMARK 3. *Consider now a term t that contains the variables x, (we denote this with $t(x)$) and let a and a' be two assignments such that $a(x) = a'(x)$. One can see from the inductive definition that $t^{\mathcal{I}}[a] = t^{\mathcal{I}}[a']$, since the only variable that occours in t is x. This can be extended to terms that contains n variables and to formulas that contains n* free variables*.*

PROPOSITION 8.3.

(1) If $t(x_1, \ldots, x_n)$ is a term that contains the variables $x_1, \ldots, x_n$, then if $a(x_i) = a'(x_i)$ for every $1 \leq i \leq n$, then $t^{\mathcal{I}}[a] = t^{\mathcal{I}}[a']$.
(2) If $\phi(x_1, \ldots, x_n)$ is a formiula that contains the free variables $x_1, \ldots, x_n$, then if $a(x_i) = a'(x_i)$ for every $1 \leq i \leq n$, then $\mathcal{I} \models \phi(x_1, \ldots, x_n)[a]$ iff $\mathcal{I} \models \phi(x_1, \ldots, x_n)[a']$.

3.1. Satisfiability, Validity, and Logical Consequence. In every logical language the notions of satisfiable formula, valid formula, unsatisfiable fomrula, and the logical consequence relation is defined on the basis of the semantics of the logic. The semantics of a logic defines how the symbols of the logic can be interpreted in some structure called interpretation, and defines the satisfaction relation, usually denoted by $\models$ between interpretation and the formulas of the logical language. The definition of the above concepts is almost the same in all the logics. In the following we report the specific definition for the first order logic, but the student is invited to compare these definitions with the analogous definitions given for propositional logic, in order to recognize the analogies and the (small) differences.

DEFINITION 8.12 (Model, satisfiability and validity). *(1) An interpretation $\mathcal{I}$ of a signature Σ is a model of a first order formula ϕ in the signature Σ w.r.t. the assignment a, if $\phi[a]$ is evaluated true in $\mathcal{I}$, in symbols if*

$$\mathcal{I} \models \phi[a]$$

(2) A formula ϕ is satisfiable *if there is some interpretation $\mathcal{I}$ and some assignment a such that $\mathcal{I} \models \phi[a]$.*
(3) A formula ϕ is unsatisfiable *if it is not satisfiable.*
(4) A formula ϕ is valid *if every $\mathcal{I}$ and every assignment a $\mathcal{I} \models \phi[a]$*

REMARK 4. *Consider $\phi(x_1, \ldots, x_n)$ to be any first-order formula with free variables $x_1, \ldots, x_n$. We say that*

- *the sentence $\exists x_1, \ldots, \exists x_n.\phi(x_1, \ldots, x_n)$ is the existential closure of ϕ;*
- *the sentence $\forall x_1, \ldots, \forall x_n.\phi(x_1, \ldots, x_n)$ is the universal closure of ϕ.*

Satisfiability and validify of open formulas (i.e., formuilas with free variables) cen be reduced to satisfiability and validity of sentences (closed formulas = formulas without free variabels). This is stated in the following proposition.

PROPOSITION 8.4.

(1) $\phi(x_1, \ldots, x_n)$ is satisfiable iff $\exists x_1 \ldots x_n.\phi(x_1, \ldots, x_n)$ is satisfiable ;
(2) $\phi(x_1, \ldots, x_n)$ is valid iff $\forall x_1 \ldots x_n.\phi(x_1, \ldots, x_n)$ is valid.

PROOF. By exercize □

The relation of logical consequence, expresses the fact that one formula is true under the hypothesis that a set of formulas are true.

DEFINITION 8.13 (Logical Consequence). *A formula ϕ is a logical consequence of a set of formulas Γ, in symbols $\Gamma \models \phi$, if for all interpretations $\mathcal{I}$ and for all assignment a*

$$\mathcal{I} \models \Gamma[a] \quad \Longrightarrow \quad \mathcal{I} \models \phi[a]$$

where $\mathcal{I} \models \Gamma[a]$ means that $\mathcal{I}$ satisfies all the formulas in Γ under a.

3.2. Properties of quantifiers. The most important extension of FOL w.r.t., propositional logic are quantifiers. In the following let us see the most important properties of universal and existential quantifiers.

We start by showing that the formula $\forall x\phi(x)$, read as "the propoerty specified by the formula $\phi(x)$ is true for every x", implies $\phi(t)$, i.e., that ϕ holds for every closed term t.

PROPOSITION 8.5. *For every term t and formula $\phi(x)$ the formula*

$$\forall x\phi(x) \to \phi(t)$$

is valid.

PROOF. We have to prove that $\mathcal{I} \models \forall x\phi(x) \to \phi(t)[a]$ for every interpretation $\mathcal{I}$ and any assignment a. This is equivalent to prove that if $\mathcal{I} \models \forall x\phi(x)[a]$ then $\mathcal{I} \models \phi(t)[a]$.

$$\begin{aligned}\mathcal{I} \models \forall x\phi(x)[a] &\Longleftrightarrow \mathcal{I} \models \phi(x)[a_{x\leftarrow d}] \text{ for all } d \in \Delta^{\mathcal{I}} \\ &\Longrightarrow \mathcal{I} \models \phi(x)[a_{x\leftarrow d}] \text{ for } d = t^{\mathcal{I}}[a] \\ &\Longrightarrow \mathcal{I} \models \phi(t)[a] \text{ by Proposition 8.2}\end{aligned}$$

□

The opposite properrty holds for the existential quantifier

PROPOSITION 8.6. *For every term t and formula $\phi(x)$ the formula*

$$\phi(t) \to \exists x\phi(x)$$

n is valid.

A second important property is the duality of the two quantifiers, and the fact that they are definable one in terms of the other. This property is shown by the fact that for every formula $\phi(x)$ with one free variable x, we have that the following formulas are valid.

$$\forall x.\phi(x) \leftrightarrow \neg\exists x\neg\phi(x)$$
$$\exists x.\phi(x) \leftrightarrow \neg\forall x\neg\phi(x)$$

A third property concern the vacuous quantification. If a variable x is not free in ϕ then quantifying ϕ on x does not have any effect. Indeed the formulas

$$\forall x\phi \leftrightarrow \phi$$
$$\exists x\phi \leftrightarrow \phi$$

are valid when x is not free in ϕ.

Let us now see how quantifiers interacts with connectives. We start by noticing that $\forall x$ commutes with $\wedge$. Indeed thhe formula

$$\forall x(\phi \wedge \psi) \leftrightarrow \forall x\phi \wedge \forall x\psi$$

is valid. Differently we have that $\forall$ does not commute with $\vee$. Indeed the formula

$$\forall x(\phi \vee \psi) \leftrightarrow \forall x\phi \vee \forall x\psi$$

is not valid. Consider the following example.

EXAMPLE 8.16. *Let P be a unary predicate, we have that $\forall x(P(x) \vee \neg P(x))$ is true in every interpretation $\mathcal{I}$, as every of the domain $\Delta^{\mathcal{I}}$ is either in $P^{\mathcal{I}}$ or not in $P^{\mathcal{I}}$; while $\forall xP(x) \vee \forall x\neg P(x)$ is true only in the interpretation where P is interpreted either in the empty set of in the entire domain $\Delta^{\mathcal{I}}$.*

However, we have that this formula holds in one direction. indeed the formula

$$\forall x\phi \vee \forall x\psi \leftrightarrow \forall x(\phi \vee \psi)$$

is valid.

Let us now consider $\forall$ and $\neg$. In English saying "not all flowers are beautifull" it is not the same as saying "all flowers are not beautifull". This holds also in FOL. Inded the formula ¡

$$\forall x\neg\phi \leftrightarrow \neg\forall x\phi$$

is not valid.

The relationship between $\forall$ and $\rightarrow$. We have that the formula

$$\forall x\phi \rightarrow \forall x\psi \leftrightarrow \forall x(\phi \rightarrow \psi)$$

is not valid. Consider for instan the following interpretation.

EXAMPLE 8.17. *$\forall x\neg P(x) \rightarrow \forall xP(x)$ is satisfied by all the interpretations. where P is not interpreteed in the empty set. while $\forall x(\neg P(x) \rightarrow P(x))$ is satisfied only in the interpretations where P is interpreted in the emtore domain (since it is equivalent to $\forall xP(x)$).*

As happens for the disjunction we have that one direction is valid. Indeed

$$\forall x(\phi \rightarrow \psi) \rightarrow \forall x\phi \rightarrow \forall x\psi$$

is valid.

As a final property, we show that $\forall$ commutes with $\vee$ and $\rightarrow$ under some conditions. IN particular we have that if x is not free in ϕ we have that

$$\forall x(\phi \vee \psi) \leftrightarrow \phi \vee \forall x\psi$$
$$\forall x(\phi \rightarrow \psi) \leftrightarrow \phi \rightarrow \forall x\psi$$

are both valid.

Let us perform the same analisys for the existential quantifier. We summarize the results in the following table

$$\begin{aligned}
\exists x(\phi \wedge \psi) \leftrightarrow \exists x\phi \wedge \exists x\psi &\quad \text{is not valid} \\
\exists x(\phi \wedge \psi) \rightarrow \exists x\phi \wedge \exists x\psi &\quad \text{is valid} \\
\exists x(\phi \vee \psi) \leftrightarrow \exists x\phi \vee \exists x\psi &\quad \text{is valid} \\
\exists x\neg\phi \leftrightarrow \neg\exists x\phi &\quad \text{is not valid} \\
\neg\exists x\phi \rightarrow \exists x\neg\phi &\quad \text{is valid} \\
\exists x(\phi \rightarrow \psi) \leftrightarrow \exists x\phi \rightarrow \exists x\psi &\quad \text{is not valid} \\
(\exists x\phi \rightarrow \exists x\psi) \rightarrow \exists x(\phi \rightarrow \psi) &\quad \text{is valid} \\
\exists x(\phi \wedge \psi) \leftrightarrow \phi \wedge \exists x\psi &\quad \text{is valid if } x \text{ is not free in } \phi \\
\exists x(\phi \rightarrow \psi) \leftrightarrow \phi \rightarrow \exists x\psi &\quad \text{is valid if } x \text{ is not free in } \phi
\end{aligned}$$

Let us now see how quantifiers interacts one another. The formula

$$\forall x\phi \rightarrow \exists x\phi$$

is valid. The validity is guaranteed by the fact that the domain $\Delta^{\mathcal{I}}$ of any first order interpretation $\mathcal{I}$ is not empty. Indeed if $\mathcal{I} \models \forall x\phi$, ϕ is true for all the elements of the domain $\Delta^{\mathcal{I}}$, and since $\Delta^{\mathcal{I}}$ is not empty then there exists at least one element of the domain for which the property $\phi(x)$ is true. Obviously the converse

$$\exists x\phi \rightarrow \forall x\phi$$

Quantifiers of the same type can permute. Therefore

$$\forall x\forall y\phi \leftrightarrow \forall y\forall x\phi$$
$$\exists x\exists y\phi \leftrightarrow \exists y\exists x\phi$$

are valid formulas. Instead, quantifiers of different type do not permute Indeed

$$\forall x\exists y\phi \leftrightarrow \exists y\forall x\phi$$

is not valid. An intuitive counteressample of the previous equivalence can be obtained by a binary relation R. We have that $\forall x\exists yR(x, y)$ means that "exery x is related via R with some y, while $\exists y\forall xR(x, y)$ means that there is a y with which every x is related. A concrete example

$$\forall x\exists y\,\mathsf{supervisor}(y, x)$$

means that everybody has a supervisor. while the formula

$$\exists y\forall x\,\mathsf{supervisor}(y, x)$$

everybody has the same supervisor, or there is somebody who is supervising everybody.

4. Exercises

Exercise 126:
Draw the parse tree of the formula $\forall x\, P(x) \to \exists y \exists z\, Q(y,z) \wedge \neg \exists s\, R(x)$ respecting the precedence of operators.

Exercise 127:
For each the following formulas say which are the free and the bound occurrences of variables in the following formula. For the bound variables indicate the quantifier that bounds it.

(1) $\exists x(E(x,y) \wedge \exists y \neg E(y,x))$;
(2) $\forall x(\forall z A(x,z) \to \exists y(R(x,y) \wedge \forall x P(x,z)))$;
(3) $\forall z(A(x) \wedge B(x,y)) \to \forall z A(z,z)$.

Exercise 128:
Suppose that Σ contains the following symbols with the associated intuitive meaning:

T	Trento	
R	Rome	
I	Iitaly	
M	Marocco	
L	Lorenzo	
F	Francesca	
$major(x)$	The major of $xmotherOf(x)$	the mother of x
$fatherOf(x)$	the father of x	
$nationality(x)$	The nationality (country) of x	
$homeTown(x)$	The hometown of x	
$capital(x)$	the capital of x	
$route(x,y)$	the route from x to y	
$routeThroug(x,y,z)$	The route from x to y passing through z	

write the terms that corresponds to the following English description

(1) the capital of Italy;
(2) the hometown of Lorenzo;
(3) the route that connects the hometowns of Lorenzo and Francesca;
(4) the capital of the nationality of Francesca;
(5) the route from the hometown of the father of Francesca and the mother of Lorenzo passing through the capital of Marocco;
(6) the route going from the hometown of the major of the capital of Marocco to the hometown of his/her mother qspassing through the hometown of his/her grandfather.

Exercise 129:

Using the following symbols with the associated intuitive meaning

a	Adam
e	Eva
c	Cid
$\mathsf{B}(x)$	x is blond
$\mathsf{C}(x)$	x is a cat
$\mathsf{L}(x, y)$	x loves y
$\mathsf{T}(x, y)$	x is taller than y

Transcribe the following FOL sentences into English:

(1) $\mathsf{T}(\mathsf{c},\mathsf{e})$
(2) $\mathsf{L}(\mathsf{c},\mathsf{e})$
(3) $\neg\mathsf{T}(\mathsf{c},\mathsf{c})$
(4) $\mathsf{B}(\mathsf{c})$
(5) $\mathsf{T}(\mathsf{c},\mathsf{e}) \to \mathsf{L}(\mathsf{c},\mathsf{e})$
(6) $\mathsf{L}(\mathsf{c},\mathsf{e}) \vee \mathsf{L}(\mathsf{c},\mathsf{c})$
(7) $\neg(\mathsf{L}(\mathsf{c},\mathsf{e}) \wedge \mathsf{L}(\mathsf{c},\mathsf{a}))$
(8) $\mathsf{B}(\mathsf{c}) \leftrightarrow (\mathsf{L}(\mathsf{c},\mathsf{e}) \vee \mathsf{L}(\mathsf{c},\mathsf{c}))$

Transcribe the following English sentences into sentences first order logic;

(1) Cid is a cat.
(2) Cid is taller than Adam.
(3) Either Cid is a cat or he is taller than Adam.
(4) If Cid is taller than Eve then he loves her.
(5) Cid loves Eve if he is taller than she is.
(6) Eve loves both Adam and Cid.
(7) Eve loves either Adam or Cid.
(8) Either Adam loves Eve or Eve loves Adam, but both love Cid.
(9) Only if Cid is a cat does Eve love him.
(10) Eve is taller than but does not love Cid.

Exercise 130:

Transcribe the following english sentence in FOL

(1) Everyone loves Eve.
(2) Eve loves somebody.
(3) Eve loves everyone.
(4) Some cat loves some dog.
(5) Somebody is neither a cat nor a dog.
(6) Someone blond loves Eve.
(7) Some cat is blond.
(8) Somebody loves all cats.
(9) No cat is a dog.
(10) Someone loves someone.
(11) Everybody loves everyone.
(12) Someone loves everyone.
(13) Someone is loved by everyone.

(14) Everyone loves someone.
(15) Everyone is loved by somebody.

Exercise 131:

For each of the following formulas indicate: (a) the scope of the quantifiers (b) the free variables, and (c) whether it is a sentence (closed formula)

(1) $\forall x(A(x,y) \wedge B(x))$
(2) $\forall x(\forall y(A(x,y) \rightarrow B(x))$
(3) $\forall x(\forall y(A(x,y))) \rightarrow B(x)$
(4) $\exists x(: \forall y(A(x,y)))$
(5) $\forall x(A(x,y)) \wedge B(x)$
(6) $\forall x(A(x,x)) \wedge \forall y(B(y))$

Solution

□

Exercise 132:

Say if the term $f(x,y)$ is free for z in the following formulas:

(1) $\forall z(P(x,y,z) \rightarrow \exists y Q(x,y))$
(2) $\forall x(R(y,z) \wedge existszQ(z,y))$
(3) $P(x,y,z) \wedge \exists x Q(x,z)$

Exercise 133:

Apply the substitution $x/a, y/b, z/c$ to the formula $\forall z P(x,y,z) \rightarrow \forall y(Q(x,y) \wedge \exists x R(x,z))$

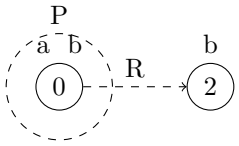
Exercise 134:

Find an interpretation that satisfies the formula

$$P(a) \wedge P(b) \wedge \forall x(P(x) \rightarrow \exists y.(R(x,y) \wedge \neg P(y)))$$

Solution

- $\Delta^{\mathcal{I}} = \{0,1\}$
- $a^{\mathcal{I}} = 0$
- $b^{\mathcal{I}} = 0$
- $P^{|} = \{0\}$
- $R^{\mathcal{I}} = \{(0,1)\}$



□

Exercise 135:

Transform in FOL the following sentences:

(1) If in a ceremony somebody seats between two people that are married, he or she must be their child;
(2) If a tree produces some apples then there must exist a farmer that takes care of it.
(3) In a personal exhibition all the paintings are done by a unique painter.

Solution

(1) If in a ceremony somebody seats between two people that are married, he or she must be their child;

$$\forall xyz(\mathsf{seatBetween}(x,y,z) \land \mathsf{married}(x,y) \rightarrow \mathsf{parent}(x,z) \land \mathsf{parent}(y,z) \tag{100}$$

(2) If a tree produces some apples then there must exist a farmer that takes care of it.

$$\forall x(\mathsf{tree}(x) \land \mathsf{produces}(x,y) \land \mathsf{fruit}(y) \rightarrow \exists z(\mathsf{farmer}(z) \land \mathsf{takesCare}(z,x)))$$

(3) In a personal exhibition all the paintings are done by a unique painter.

$$\forall x(\mathsf{personalExib}(x) \rightarrow \exists y(\mathsf{painter}(y) \land \forall z(\mathsf{exposes}(x,z) \rightarrow \mathsf{paints}(y,z))))$$

□

Exercise 136:
Translate the following sentences in FOL.

(1) Everything is bitter or sweet
(2) Either everything is bitter or everything is sweet
(3) There is somebody who is loved by everyone
(4) Nobody is loved by no one
(5) If someone is noisy, everybody is annoyed
(6) Frogs are green.
(7) Frogs are not green.
(8) No frog is green.
(9) Some frogs are not green.
(10) A mechanic likes Bob.
(11) A mechanic likes herself.
(12) Every mechanic likes Bob.
(13) Some mechanic likes every nurse.
(14) There is a mechanic who is liked by every nurse.

Solution

(1) Everything is bitter or sweet

$$\forall x(\mathsf{bitter}(x) \lor \mathsf{sweet}(x))$$

(2) Either everything is bitter or everything is sweet

$$\forall x\ \mathsf{bitter}(x) \lor \forall x\ \mathsf{sweet}(x)$$

(3) There is somebody who is loved by everyone

$$\exists x \forall y\ \mathsf{loves}(y,x)$$

(4) Nobody is loved by no one

$$\neg\exists x \neg\exists y\ \mathsf{loves}(y,x)$$

(5) If someone is noisy, everybody is annoyed

$$\exists x \ \mathsf{noisy}(x) \to \forall y \ \mathsf{annoyed}(y)$$

(6) Frogs are green.

$$\forall x \ (\mathsf{frog}(x) \to \mathsf{green}(x))$$

(7) Frogs are not green.

$$\forall x \ (\mathsf{frog}(x) \to \neg\mathsf{green}(x))$$

(8) No frog is green.

$$\neg\exists x \ (\mathsf{frog}(x) \wedge \mathsf{green}(x))$$

(9) Some frogs are not green.

$$\exists x \ (\mathsf{frog}(x) \wedge \neg\mathsf{green}(x))$$

(10) A mechanic likes Bob

$$\exists x(\mathsf{mechanic}(x) \wedge \mathsf{likes}(x, \mathsf{Bob})$$

(11) A mechanic likes herself.

$$\exists x(\mathsf{mechanic}(x) \wedge \mathsf{likes}(x, x))$$

(12) Every mechanic likes Bob.

$$\forall x(\mathsf{mechanic}(x) \to \mathsf{likes}(x, \mathsf{Bob}))$$

(13) Some mechanic likes every nurse.

$$\exists x(\mathsf{mechanic}(x) \wedge \forall y(\mathsf{nurse}(y) \to \mathsf{likes}(x, y)))$$

(14) There is a mechanic who is liked by every nurse.

$$\exists x(\mathsf{mechanic}(x) \wedge \forall y(\mathsf{nurse}(y) \to \mathsf{likes}(y, x)))$$

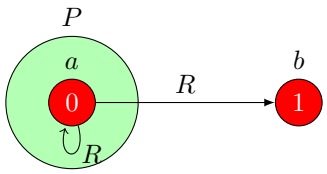
□

Exercise 137:

Given the FOL interpretation $\mathcal{I}$ defined on the domain $\Delta_{\mathcal{I}} = \{0, 1\}$ and the interpretation: $\mathcal{I}(P) = \{0\}$ and $\mathcal{I}(R) = \{(0, 0), (0, 1)\}$ $\mathcal{I}(a) = 0$ and $\mathcal{I}(b) = 1$. Verify whether the following formulas are true in M:

(1) $\forall x P(x)$
(2) $P(a)$
(3) $P(b)$
(4) $R(a, b)$
(5) $\neg R(a, a)$
(6) $\exists x R(a, x)$
(7) $\forall x R(x, b)$
(8) $\forall x R(x, x) \to P(x)$
(9) $\forall x \neg R(a, x) \to P(x)$
(10) $\forall x(P(x) \to \neg R(a, x))$

Solution The interpretation $\mathcal{I}$ can be graphically represented as follows:



Let us now check the truth value of the formulas

(1) $\forall x P(x)$ is false since thate is an element of the domain which is not in $\mathcal{I}(P)$;
(2) $P(a)$ is true since $\mathcal{I}(a) = 0 \in \mathcal{I}(P)$;
(3) $P(b)$ is false since $\mathcal{I}(b) = 1 \notin \mathcal{I}(P)$;
(4) $R(a, b)$ is true since $(\mathcal{I}(a), \mathcal{I}(b)) = (0, 1) \in \mathcal{I}(R)$;
(5) $\neg R(a, a)$ is false since $(\mathcal{I}(a), \mathcal{I}(a)) = (0, 0) \in \mathcal{I}(R)$, which makes $P(a, a)$ true and therefore $\neg P(a, a)$ false;
(6) $\exists x R(a, x)$ is true since if x is assigned to 0 we have that the pair $(\mathcal{I}(a), x) = (0, 0) \in \mathcal{I}(R)$;
(7) $\forall x R(x, b)$ is false because, if we assigne 0 to x we have that the pair $(x, \mathcal{I}(b)) = (0, 1) \notin \mathcal{I}(R)$;
(8) $\forall x R(x, x) \to P(x)$ is true. To show this we have to chrck the truth of $R(x, x) \to P(x)$ for all the possible assignments of x. If x is assigned to 0, $\mathcal{I} \models P(x)[x := 0]$ and therefore $\mathcal{I} \models R(x, x) \to P(x)[x := 0]$; If x is assigned to 1, then we have that $\mathcal{I} \not\models R(x, x)[x := 1]$, and therefore $\mathcal{I} \models R(x, x) \to P(x)[x := 1]$, Since $R(x, x) \to P(x)$ is true for all the assignments of x, we can conclude that $\mathcal{I} \models \forall x(R(x, x) \to P(x))$ is true..
(9) $\forall x \neg R(a, x) \to P(x)$ is true. As in the previous case we have to check for all the assignments of x. Since we have that $\mathcal{I} \models R(a, x)[x := 0]$ we have that $\mathcal{I} \models \neg R(a, x) \to P(x)[x := 0]$; We also have that $\mathcal{I} \models R(a, x)[x := 1]$, and therefore $\mathcal{I} \models \neg R(a, x) \to P(x)[x := 1]$. We can therefore conclude that $\mathcal{I} \models \forall x(\neg R(a, x) \to P(x)$
(10) $\forall x(P(x) \to \neg R(a, x))$ is false since if x is assignmet to 0 we have that $\mathcal{I} \not\models P(x) \to \neg R(a, x)[x = 0]$ as $0 \in \mathcal{I}(P)$ and $(0, 0) \in \mathcal{I}(R)$.

□

Exercise 138:
Describe all the models of the following set of formulas in the domain $\{1, 2, 3\}$.

(101) $\forall x \neg R(x, x)$
(102) $\forall x \forall y(R(x, y) \to R(y, x)) \wedge$
(103) $\forall x(A(x) \to \exists y(R(x, y) \wedge A(y)))$

Solution

$\mathcal{I}(A)$	$\mathcal{I}(R)$
$\emptyset$	any symmetric relation on $\{1,2,3\}$
$\{1,2\}$	$\{(1,2),(2,1)\}$
$\{1,3\}$	$\{(1,3),(3,1)\}$
$\{2,3\}$	$\{(2,3),(3,2)\}$
$\{1,2\}$	$\{(1,2),(2,1),(1,3),(3,1)\}$
$\{1,2\}$	$\{(1,2),(2,1),(2,3),(3,2)\}$
$\{1,2\}$	$\{(1,2).(2,1),(1,3),(3,1),(2,3),(3,2))\}$
$\{1,3\}$	$\{(1,3),(3,1),(1,2),(2,1)\}$
$\{1,3\}$	$\{(1,3),(3,1),(3,2),(2,3)\}$
$\{1,3\}$	$\{(1,2).(2,1),(1,3),(3,1),(2,3),(3,2))\}$
$\{2,3\}$	$\{(2,3),(3,2)\}$
$\{2,3\}$	$\{(2,3),(3,2),(1,2),(2,1)\}$
$\{2,3\}$	$\{(2,3),(3,2),(1,3),(3,1)\}$
$\{2,3\}$	$\{(1,2).(2,1),(1,3),(3,1),(2,3),(3,2))\}$
$\{1,2,3\}$	$\{(1,2),(2,1),(1,3),(3,1))\}$
$\{1,2,3\}$	$\{(1,2),(2,1),(2,3),(3,2))\}$
$\{1,2,3\}$	$\{(1,3),(3,1),(2,3),(3,2))\}$
$\{1,2,3\}$	$\{(1,2).(2,1),(1,3),(3,1),(2,3),(3,2))\}$

□

Exercise 139:
Consider the formula

$$\forall x \forall y (P(x,y) \wedge Q(y,x))$$

Give an interpretation that satisfies it in the domain $D = \{\text{John}, \text{Paul}, \text{Mary}\}$ **Exercise 140:**

Show the following equivalence:

$$(\forall x P(x) \wedge \forall x Q(x)) \leftrightarrow \forall x (P(x) \wedge Q(x)) \tag{104}$$

is valid

Solution Let us consider an interpretation $\mathcal{I}$ on the domain Δ, To prove that (104) is valid, we show that $\mathcal{I} \models \forall x P(x) \wedge \forall x Q(x)$ if and only if $\mathcal{I} \models \forall x (P(x) \wedge Q(x))$.

$$\begin{aligned}
\mathcal{I} \models \forall x (P(x) \wedge Q(x) &\Leftrightarrow \text{for all } d \in \Delta, \mathcal{I} \models P(x) \wedge Q(x)[a_{x \leftarrow d}] \\
&\Leftrightarrow \text{for all } d \in \Delta, \mathcal{I} \models P(x) \text{ and } \mathcal{I} \models Q(x)[a_{x \leftarrow d}] \\
&\Leftrightarrow \mathcal{I} \models \forall x P(x) \text{ and } \mathcal{I} \models \forall x Q(x) \\
&\Leftrightarrow \mathcal{I} \models \forall x P(x) \wedge \forall x Q(x)
\end{aligned}$$

□

Exercise 141:
Decide if

$$\forall x Q(x) \rightarrow \forall x P(x) \leftrightarrow \exists z \forall y (Q(y) \vee P(z))$$

is valid, satisfiable, non valid, or unsatisfiable.

Exercise 142:

Decide if

$$\forall x Q(x) \to \forall x P(x) \leftrightarrow \exists z \forall y (Q(y) \vee P(z))$$

is valid, satisfiable, non valid, or unsatisfiable, and explain your answer.

Solution One can try to rewrite the left part of the formula in order to obtain a formula equivalent to the right part.

$$\begin{aligned} \forall x Q(x) &\to \forall x P(x) \\ \neg \forall x Q(x) &\vee \forall x P(x) \\ \exists x \neg Q(x) &\vee \forall x P(x) \\ \exists z \neg Q(z) &\vee \forall y P(y) \\ \exists z \forall y (\neg Q(z) &\vee P(y)) \end{aligned}$$

One can now easily see that the obtained formula cannot be equivalent to $\exists z \forall y (\neg Q(z) \vee P(y))$. For instance the interpretation

$$\mathcal{I}(Q) = \{a, b\} \qquad\qquad \mathcal{I}(P) = \{a\}$$

satisfies $\exists x \forall y (Q(x) \vee P(y))$. Indeed $\mathcal{I} \models Q(a)$ and therefore $\mathcal{I} \models \forall y (Q(a) \vee P(y))$ and therefore $\mathcal{I} \models \exists x \forall y (Q(x) \wedge P(y))$. On the other hand $\mathcal{I} \not\models \exists x \forall y (\neg Q(x) \vee P(y))$ since there are no constants for which $Q(x)$ is true, and $Q(x)$ is not true for all the elements of the domain. Since we have foound an interpretation that satisfies the right hand side and do not satisfy the left hand side of the equivalence, the equivalence is not satisfied by an interpretation and therefore it is not valid.

The formula is satisfiable, since the interpretation that makes $\forall x P(x)$ always true, satisfies noth the left and the right hand side of the equivalence.

□

Exercise 143:

Show that the following formula is not valid

$$(\forall x P(x) \vee \forall x Q(x)) \leftrightarrow \forall x (P(x) \vee Q(x))$$

and provide an english sentence by instantiating the predicates P and Q in english adjectives with is intuitively false.

Solution Consider the first order interpretation $\mathcal{I}$ on the domain of natural numbers $\mathbb{N} = \{1, 2, 3, \dots\}$, where P is interpreted in the set of the even numbers, and Q in the set of odd numbers. We have that $\mathcal{I} \models \forall x (P(x) \vee Q(x)$ as every natural number is either odd or even. However $\mathcal{I} \not\models \forall x P(x)$ and $\mathcal{I} \not\models \forall x Q(x)$ since not all the numbers are even, or all the numbers are odd, and therefore $\mathcal{I} \not\models \forall x P(x) \vee \forall x Q(x)$.
□

Exercise 144:

For the following formula check if it is (a) valid, (b) satisfiable, (c) not valid, (d) unsatisfiable. Notice that more than one option is possible.

$$\forall x \exists y (A(x) \leftrightarrow R(x, y)) \leftrightarrow \forall x (A(x) \leftrightarrow \exists y R(x, y))$$

For each choice provide an argument that support your choice.

Solution The formula is not valid and satisfiable. Indeed we can provide a counterexample, i.e., an interpretation that does not satisfy the formula, and an example i.e, an interpretation that does satisfy it. Consider the two interpretations

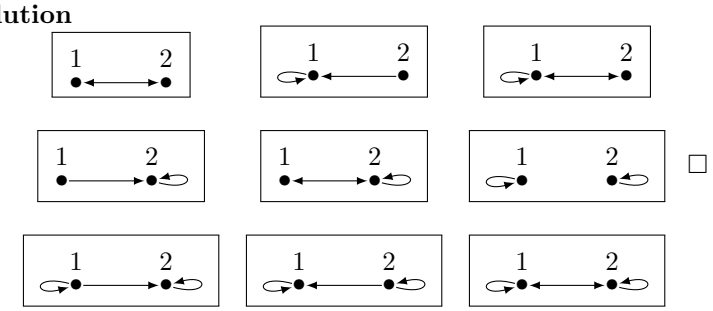
The leftmost interpretation does not satisfy the formula. Indeed $\exists y(A(x) \leftrightarrow R(x,y))$ is true for all the assignments to x, but $(A(x) \leftrightarrow \exists y R(x,y))$ is false when x is assigned to 0. Indeed, while $\exists y R(x,y)[x \leftarrow 0]$ is true, because $R(x,y)[x \leftarrow 0, y \leftarrow 2]$ is true, $A(x)[x \leftarrow 0$ is false, and therefore the equivalence $A(x) \leftrightarrow \exists y R(x,y)$ is false when x assigned to 0.

Instead, one can easily see that, the rightmost interpretation instead satisfies the formula. □

Exercise 145:

List all the models of $\forall x \exists y R(x,y)$ in the domain $\{1,2\}$.

Solution



□

Exercise 146:

List all the models of $\exists x \forall y R(x,y)$ in the domain $\{1,2\}$.

Exercise 147:

Find a formula ϕ which is true for some interpretation with infinite domain, but false for all interpretation with finite domains.

Solution Define ϕ to be the conjunciton of:

$$\forall xyz(p(x,y) \wedge p(y,z) \rightarrow p(x,z))$$
$$\forall x \neg p(x,x)$$
$$\forall x \exists y p(x,y)$$

We can verify that, for any finite interpretation, ϕ is false. On the other haqnd, ϕ is true under the interpretation $\mathcal{I}$ with $\Delta^{\mathcal{I}} = \mathbb{N}$ (the set of natural numbers and $\mathcal{I}(R) = \{(n,m) \in \mathbb{N}^2 \mid n < m\}$ □

CHAPTER 9

Knowledge Representation in First Order Logic

First order logic, as any other logic, can be used as language to specify knowledge about some particular domain. The type of knowledge that can be expressed in first order logic is that a certain proposiiton that can be specified by a first order sentence is true in all the possible configurations (worlds) of the domain we are considering.

In describing a domain we want to impose that (non logical) symbols have a specific semantics, so they cannot be interpreted deliberately. On the other hand FOL semantics alone does not impose any specific constraint on the interpretation of non logical symbol. For instance the constant "red" and "blue" can be interpreted in the same domain element, with the effect that the formula $red = blue$ is true. To constraint the semantics of non logical symbols of a signature Σ we have to limit the way in which such symbols are interpreted; in other words, we require that Σ is interpreted only in a subset of the entire set of Σ-structures. For instance, we want to consider only the Σ-structures in which *red* and *blue* are interpreted in two distinct individuals of the domain. Or equivalently the Σ-structures where the formula $\neg(red = blue)$ is true.

Another example is the following: Consider a signature Σ that contains two binary predicates Ancestor and Parent. We would like that the two relational symbols are interpreted according to the the usual (english) meaning of the corresponding words. I.e., that

(105) *a person is an anchestor of another preson if and only if there is a chain of parents between the two*

Put it formally the relational symbol Parent should be interpreted in the *transitive closure* of the relation Parent. While in the previous example we easily come up with a formula that captures the semantic condition, in this example coming up with a formula that "captures" the constraint expressed in (105) is not easy (actually it is impossible in FOL).

DEFINITION 9.1. *A* first order theory *on a signature Σ is a set Γ of first order sentences closed under logical consequence, i.e, if $\Gamma \models \phi$ then $\phi \in \Gamma$. The set of* axioms *of a theory Γ is a subset $\Delta \subseteq \Gamma$ such that and $\Delta \models \gamma$ for all $\gamma \in \Gamma$. Given a class S of Σ-structures, the* first order theory of S *is the set of sentences that are true in all the Σ-structures of S.*

One of the main motivation for the development of first order logic across the end of the 19th century and the beginning of the 20th century was the so called foundational crisis of mathemathics, that rises as a consequence of the discovery of several paradoxes or counter-intuitive results in mathematical theories. In the early 1920s, the German mathematician David Hilbert (1862–1943) proposed a

research program, called Hilbert's Program[1] that has the objective of describing all of mathematics in axiomatic form, together with a proof that this axiomatization of mathematics is consistent (i.e., not contraddictory). Stimulated by this ambitious program during the 20th century mathematical logics and in particular first order logic received a lot of attention from scientists, with the effect of developing a number of theories for different mathematical structures. These theories are of particular importance for artificial intelligence, since the formalization of knowledge about general concepts like time, quantities, space, ... can be mutuated from axiomatization of mathematial structures such as linar orders, partial orders, topologies, etc. In this chapter we report some example of axiomatic theories of the most important mathematical structures.

First order logics has also been used to specify ontological knowledge. An ontology is a formal, explicit, shared specification of a conceptualization of a domain Gruber 1993. A conceptualization describes the objects, the concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them. A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose.

First order logical theories have received also a lot of attention since the begining of Artificial Intelligence era. John Mc Carthy, one of the father of Artificial Intelligence, in the 60's proposed to use logical language to encode commonsense knowledge about the world with (first order) logic McCarthy 1959. Since then one of the most fruitflul field of artificial intelligence unde the label of Knowledge Representation and Reasoning developed logical theories and reasoning methods for (a subclass) of first order logical language. The paper E. Davis 2017 provides a large set of examples of formalizing commonsense knowledge by means of (first order) logical theores. Commonsense knowledge representation and reasoning is a central problem in artificial intelligence, if we want to build agents that are capable to operate in environments where humans can operate. Encoding commonsense knowledge with a set of (first order) logical formulas is an approach that has been pursued since the earliest days of the field of AI.

1. First order theories for algebraic structures

Some important algebraic structure are very useful to represent the knowledge and reason about certain phenomena. Having a first order logic axiomatization of such structures is useful, since one can infer automaticaly facts that are true in the structures. starting from the set of axioms.

1.1. Ordered sets. A set can be ordered in the sense that some elements comes before than others. An order on a set can be used to represent many real world aspects. For instance the answers obtained by a search engine are ordered by relevance, the the set of sets are ordered by containment relation, the set of cars can be ordered by price, the set of computer can be ordered according to the price, the memory size, the cpu size, ... Orders can be total or partial, in the sense that it is not necessary that for every pairs of elements of an ordered set one comes before than another. Let us provide the formal definition of ordered set,

DEFINITION 9.2. *A* partially ordered set (poset) *is a pair* $(S, \prec)$*, where* S *is a non empty set and* $\preceq \subseteq S \times X$ *is a binary relation which is*

[1] `https://plato.stanford.edu/entries/hilbert-program/#4`

(1) transitive *i.e., for all* $s,t,u \in S$ $s \prec t$ *and* $t \prec u$ *implies* $s \prec u$*;*
(2) symmetric *i.e.,* $s \prec s$ *for every* $s \in S$*, and*
(3) antisymmetric *i.e., it is not the case that* $s \prec s$*.*

where we use the notation $s \prec t$ *for* $(s,t) \in \prec$*.*

Notice that a poset is characterized by a set and a binary relation. Therefore it is Σ-structure where Σ contains only one binary predicate R (i.e., a predicate symbol with arity equal to 2). For simplicity let us call these structures R-structures. However notice that not all R-structures are poset. So we have to find a set of formulas that axiomatize the class of R-structures that are poset. This can be done by considering the first order theory on the signature $\{R\}$ that contains three formulas, corresponding to the first order "translation" of the three conditions of Definition 9.2. These formulas are

$$\forall x \forall y \forall z (R(x,y) \wedge R(y,z) \to R(x,z)) \tag{106}$$

$$\forall x \forall y (R(x,y) \to \neg R(y,x)) \tag{107}$$

$$\forall x \neg R(x,x) \tag{108}$$

The above formulas are one to one translation of the conditions on the order relation of a poset. A poset $(S, \prec)$ is *totally ordered* or is a *linear order* if for eveyr $s,t \in S$ which are different, either $s \prec t$ or $t \prec s$. R-structures that are total ordered can be axiomatized by adding the axiom

$$\forall x \forall y (R(x,y) \vee R(y,x) \vee x = y) \tag{109}$$

A partially ordered $(S, \prec)$ set is *dense* if for every $s \prec t$ there is a u such that $s \prec u$ and $u \prec t$. Dense orders can be axiomatized by adding the axiom

$$\forall x \forall y (R(x,y) \to \exists z (R(x,z) \wedge R(z,y))) \tag{110}$$

1.2. Equivalence relation. Equivalence relations are very important since it allows to partition a set of a set of equivalence classes. Therefore, it is a way to represent clustering of points.

DEFINITION 9.3. *An equivalence relation on a set* S *is a subset* $R \subseteq S \times S$ *that satisfies the following properties.*

(1) Reflexive: $(s,s) \in R$ *for all* $s \in S$*;*
(2) Symmetric: $(s,t) \in R$ *implies* $(t,s) \in R$*;*
(3) Transitive $(s,t) \in R$ *and* $(t,y) \in R$ *implies* $(s,u) \in R$

The theory of equivalence relation can be obtained by translating the above properties in first order logic:

$$\forall x R(x,x) \tag{111}$$

$$\forall x \forall y (R(x,y) \to R(y,x)) \tag{112}$$

$$\forall x \forall y \forall z (R(x,y) \wedge R(y,z) \to R(x,z)) \tag{113}$$

1.3. Peano arithmetic. One of the most important mathematical structure is the set of natural numbners $\mathbb{N} = \{0,1,2,\dots\}$ with the arithmetic operations for addition and product and the usual order relation. This is called the *standard model for arithmetic*. One important question that have been addressed by mathematical logicians is whether is is possible to provide a set of axioms A in a signature Σ

such that every Σ-structure that satisfies A is isomorphic to the standard model of arithmetic.

A (possible) signature Σ that allow to describe what is true in this important structure contains one constant $\underline{0}$ (we use $\underline{0}$ to distinguis this simbol from the natural number 0), one unary function s (for successor), two binary function $+$ and $\cdot$ (for sum and product, used with infix notation) and one binary predicate $<$ (for the ordering, also used with infix notation). This segnature provides terms for every natural number $0, 1, 2, \ldots$, They are $\underline{0}, s(\underline{0}), s(s(\underline{0})), \ldots$. Furthermore

It has been shown that there is no set of axioms which are true only in the Σ-structures isormorphic to the standard model of arithmetic. Therefore one could try to write a set of axioms that capture as much as possible all the formulas that are true in the standard model of arithmetic. This was provided by the Peano with the so called *Peano Arithmetic* which is the set of formulas that are logical consequence of the following (infinite) set of axioms

$$\forall x(\neg s(x) = \underline{0}) \tag{114}$$

$$\forall x \forall y(s(x) = s(y) \rightarrow x = y) \tag{115}$$

$$\forall x(x + \underline{0} = x) \tag{116}$$

$$\forall x \forall y(x + s(y) = s(x + y)) \tag{117}$$

$$\forall x(x \cdot \underline{0} = \underline{0}) \tag{118}$$

$$\forall x \forall y(x \cdot s(y) = (x \cdot y) + x) \tag{119}$$

$$\phi(\underline{0}) \wedge \forall x(\phi(x) \rightarrow \phi(s(x))) \rightarrow \forall x \phi(x) \tag{120}$$

(120) does not express a single formula but an infinite set of formulas for every instantiation of ϕ with a formula with the free variable x. All the logical consequence of the above axioms is called the *Peano Arithmetic*. The standard model of arithmetic is a model of the Peano Arithmetic but there are Σ-structures that satisfies Peano's axioms but are not isomorphic to the standard model of arithmetic. The reason why this is the case, and what is a formal proof of this fact is out of the scope of this lecture notes. It is the subject of a proper course in mathematical logic. Here it is enough to be aware that even with realitively simple structures like the standard model of arithmetic we cannot devise a set of formula that fully characterize it.

However, everything that one can infer from the set of Peano's Axioms will be true in the standard model of arithmetic.

2. First Order Theories for Labelled Graphs

A broad class of data, ranging from similarity networks, workflow networks to protein networks, can be modeled as graphs with data values as vertex labels. A graph is a mathematical structure which is whidely used to represent a set of objects which are connected one another in some way. Many data comes in the form of graphs, and therefore being able to represent knowledge about graphs in first order logic is important.

DEFINITION 9.4. *A graph is a pair $G = (V, E)$ where V is the set of vertices and E is the set of edges. An edge is a pair (v, v') of vertices with $v \neq v'$. A graph is directed if the edge (v, v') is considered different from the edge (v', v) and undirected if (v, v') and (v', v) are the same edge.*

To axiomatize graphs in first order logic we need only one binary predicate E. Directed graphs can be axiomatized by the only axioms

$$\forall x \neg E(x,x) \tag{121}$$

For undirected graphs we have to add also the fact that E is symmetric i.e. the axiom

$$\forall x \forall y (E(x,y) \rightarrow E(y,x)) \tag{122}$$

There are additional properties on graphs that can be axiomatized in terms of first order logic. Unfortunately the most important properties on graph structures, such as k-colorability, or connectivity cannot be axiomatized in First Order Logic. For these property one has to adopt an extension of FOL called *monadic second order logic* Gurevich 1985. Nevertheless, FOL formulas can be used to formalize properties on possible labelling of nodes and vertices of a graph. Let us first introduce the notion of labelled graph.

Definition 9.5.

In the above definition we consider the integers from 1 to n as possible labels from vertexes and nodes, however, any other set could be chosen. Notice that a graph labelled with a set of labels $\{1,\ldots,n\}$ is isomorphic to a Σ-structure $\{\Delta, \mathcal{I}\}$ where $\Delta = V$, Σ contains a unary predicate for p_i for every $i \in \{1,\ldots,n\}$ and a binary predicate r_i for every $i \in \{1,\ldots,n\}$. The formula $p_i(x)$ means that x is labelled with i and the formula $r_i(x,y)$ means that there is an edge from x to y and it is labelled with i.

To axiomatize the labelled graphs we have to add the following axioms that states that every node and edge has exactly one lable.

$$\forall x \left(\bigvee_{i=1}^{n} p_i(x) \right) \wedge \forall x \forall y \left(\bigwedge_{i<j=1}^{n} \neg (p_i(x) \wedge p_j(x)) \right) \tag{123}$$

$$\forall x \forall y \left(\bigwedge_{i<j=1}^{n} \neg (r_i(x,y) \wedge r_j(x,y)) \right) \tag{124}$$

Axiom (123) states that every vertex shuld be labelled with at least one lable. Instead, axiom (124) states that between every pair of nodes either there is no arc, if $\neg r_i(x,y)$ is false for all i, or there is one arc labelled with i, in case $r_i(x,y)$ is true. This axiom guarantees tha $r_i(x,y)$ is true for at most one i. Often graph labelling involves only vertexes. In this case we have only one binary relation r, where $r(x,y)$ means that there is an edge between x and y, and axiom (124) is vacuously true.

Additional logical formulas can be used to axiomatize other constrains on the labels of the graph. Consider the following example:

Example 9.1. *A* neighborhood constraint*Song et al. 2014 specifies label pairs that are allowed to appear on adjacent vertexes in the graph. A constraint graph $S = (L, C)$ is an undirected graph, where L is a finite set of labels and C is a set of edges on L. For every graph $G = (V, E)$, A labelling $\ell : V \rightarrow L$ of G with labels L satisfies the constrains specified in S if the following condition holds:*

$$(v, v') \in V \text{ implies } (\ell(v), \ell(v')) \in C \tag{125}$$

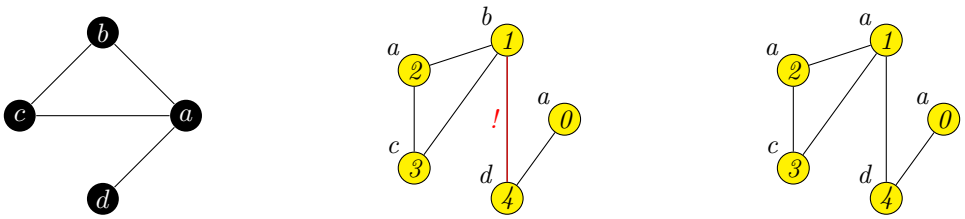


FIGURE 1. The graph on the left specifies the constraints that a labelling with a, b, c, d should satisfy. It is a graph whose nodes are the labels, and arcs represents admitted labelling of adjacent nodes. In the center, we show a graph on 5 nodes which are labelled with a, b, c, and d. This labelling, however, does not satisfy the constraints since the labelling of nodesl 1 and 4, which are adjacent, violates the constraint since (b, d) is not an arc in the constraints graph on the left. Instead, the labelling shown on the right respect all the constraints specified by the constraints graph.

The constraint (125) *can be easily expressed in first order logic.*

$$\forall x \forall y \left(r(x,y) \rightarrow \bigvee_{(i,j)\in C} (p_i(x) \wedge p_j(y) \vee p_j(x) \wedge p_i(y)) \right) \tag{126}$$

Therefore the class of graphs that satisfies (126) *are the labelled graphs which respects the constraint expressed by the constraint labelling graph* $S = (L, C)$.

3. First order theories for ontologies

An ontology is a formal, explicit, shared specification of a conceptualization of a domain Gruber 1993. A conceptualization is a formal description of the objects, the concepts, and the relations that are assumed to exist in some domain of knowledge. There are many example of ontologies that ranges from very general ontologies (called top-level or upper ontologies) to domain specific ontologies. Three well known and used examples of upper ontologies are Basic Formal Ontology (BFO) Arp, Smith, and Spear 2015 Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) Borgo and Masolo 2009, and Unified Foundational Ontology (UFO) Guizzardi 2015. In upper level ontologies concept like "state", "event", "process", are organized in a hiararchical structure and they high level relations like "being part of", "having a quality" are used to related these concepts. An exceprt of the concept hierarchy of DOLCE is shown in Figure 2.

Domain specific ontologies are ontologies which are specific for one particular domain. They have an important role in integration of heterogenous data Lenzerini 2002 and as a semantic interface for querying and accessing data Xiao et al. 2018. Large and important ontologies has been developed for instance in the domain of of medicine. For instance the SNOMED CT ontology is a systematically organized collection of medical terms. SNOMED CT concepts are organised into 19 distinct hierarchies, each of which cover different aspects of healthcare. An excerpt of one of the hierarchies of SNOMED CT is shown in Figure 3

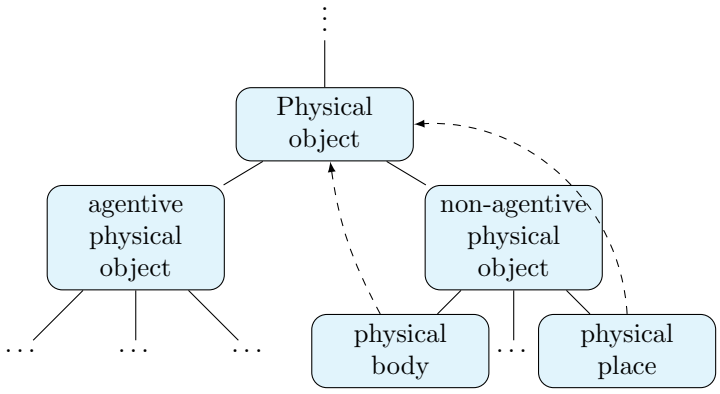


FIGURE 2. An excerpt of the concept hierarchy of DOLCE.

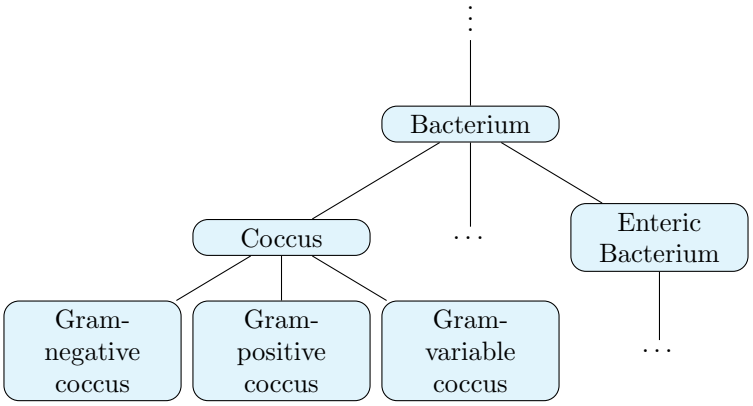


FIGURE 3. An excerpt of one of the concept hierarchies of SNOMED CT.

There are three main relationships that are formalized in an ontology. The concept hierarchy (also called is-a herarchy) the parthood-hierarchy (also called mereology) and the attributes properties. Let us see how they are usually formalized in first order logic.

3.1. Taxonomies in FOL. A taxonomy T is a DAG (directed acyclic graph) $T = (C, H)$ where C is the set of concepts and H (the hierarchy) are the edges between concepts that form a DAG on E. The arc $(c, d) \in H$ states the fact that the concept d is a specification of c. For instance in Figure 3 the arc from "Coccus" to "Gram negative Coccus", states that the concept of "gram negative coccus" is a specification of "coccus".

One possible way to formalize a taxonomy $T = (C, H)$ in FOL is to associate to every concept $c \in C$ a unary predicate $c(x)$, For instance in formalizing SNOMED CD in FOL we introduce the predicate $\mathsf{Bacterium}$, where the formula $\mathsf{Bacterium}(x)$ formalizes the fact that the object x is a bacterium, and the unary predicate Coccus,

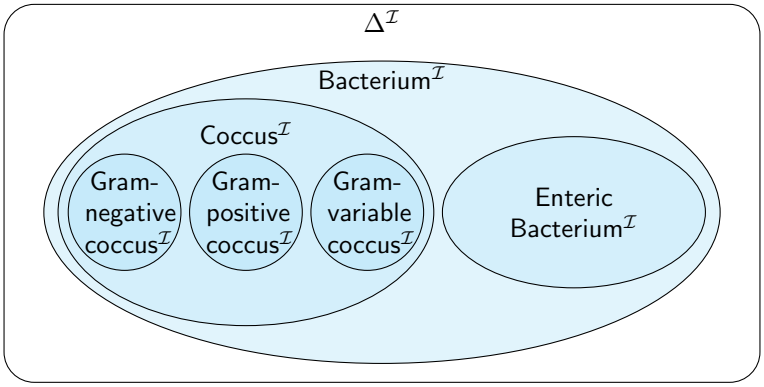


FIGURE 4. A visualization of the FOL semantics of the exceprt of SNOMED CT of Figure 3

where $\mathsf{Coccus}(x)$ means that x is a coccus. First order semantics associates to every unary predicate a subset of the domain element. Therefore, in the FOL formalization of a taxonomy, concepts are seen as sets of objects. We say that this is an *extensional semantics for concepts*.

To formalize the information contained in the hyerarchical part H of a taxonomy, we consider the fact that if a condept d is a specification of a concept c, then everyt instance of d must also be an instance of c. Therefore, we tranform the fact that $(c, d) \in H$ in the proposition "every element of the domain that is c must be also d" that in forst order logic can be rendered as:

$$\forall x(c(x) \to d(x)) \tag{127}$$

Consequently the FOL semantic of a Taxonomy $T = (C, H)$ is given in terms of set conainment. For instance the semantics of the part of the Snomer hierarchy shown in Figure 3 is shown in Figure 4 For every taxonomy $T = (C, H)$ let Γ_T the the theory obtained translatinc H in first order axioms according to (127) As a consequence of this modelling we obtain the intuitive fact that when $(c, d) \in H$ and $(d, e) \in H$. we have tha formula (127). As a consequence we have that if there is a path $c_1, c_2, \ldots, c_n$ that connects c_1 to c_n, we have that

$$\Gamma_T \models \forall x(c_n(x) \to c_1(x)) \tag{128}$$

In the SNOMED example we have that if $\Gamma_{\text{SNOMED-CT}}$ is the axiomaticazion of the SNOMED-CT hierarchies in FOL, then

$$\Gamma_{\text{SNOMED-CT}} \models \forall x(\mathsf{GramNegativeCoccus}(x) \to \mathsf{Bacterium}(x))$$

Formalizing a taxonomy $T = (C, H)$ in a first order theory allows to infer new specialization relations that are not explicitly stated in H which are the result of the transitive closure of H. These new specialization are shown in dashed lines in Figure 2 and 3.

A second aspect of a taxonimi that can be formalized in FOL is the relation between syblings. Usually, but not necessarily, the childrens of a concepts are disjoint concepts, I.e., concepts for which the concept obtained by the conjunction of two siblings is empty. This constraint can be explicitly formalized in FOL with

the axiom:

$$\forall x \left(\bigwedge_{\substack{(c,d),(c,e)\in H \\ d}} \neg(d(x) \wedge e(x)) \right) \tag{129}$$

For instance in the DOLCE taxonomy, we can add the axiom

$$\forall x \neg(\mathsf{agentivePhysicalObject}(x) \wedge \mathsf{nonAgentivePhysicalObject}(x))$$

The last important aspect of a Taxonomy concern the fact that the children concepts $d_1, \ldots, d_n$ cover the parent concept c. More precisely that every instance of the parent concept c is an instance of at least one child concept c_i. This can be easily modeledd in FOL with the axioms

$$\forall x \left(c(x) \rightarrow \bigvee_{(c,d)\in H} d(x) \right) \tag{130}$$

For example in the SNOMED-CT taxonomy we can formalize the fact that a coccus is either gram positive, or gram negative, or gram variable, by the axiom

$$\forall x \, (\mathsf{Coccus}(x) \rightarrow \mathsf{GramNegativeCoccus}(x) \vee \mathsf{GramPositiveCoccus}(x) \vee \mathsf{GramVariableCoccus}(x))$$

3.2. Axiomatizing concept relations. A second iportant conponent of an ontologies is consituite by the set of relations that connects concepts. For instance kThe most important ontological restrictions on relations concerns the domain and the range that states that a certain relations connects two concepts. To state that the *domain of a relation* r is the concept c, in FOL we can use the axioms

$$\forall x \forall y (r(x, y) \rightarrow c(x)) \tag{131}$$

$$\forall x (c(x) \rightarrow \exists y \, r(x, y)) \tag{132}$$

To state that the *range of a relation* r is the concept d, in FOL we can use the axioms

$$\forall x \forall y (r(x, y) \rightarrow d(y)) \tag{133}$$

$$\forall x (d(y) \rightarrow \exists x \, r(x, y)) \tag{134}$$

It is also often the case that in ontology we have constraints that involves both the domian and the range of a relation. For instnce one can state that the engine of an electric car is electring and the engine of a non electic care is a fossile fuel engine. This can be done with axioms of the form

$$\forall x (c(x) \rightarrow \forall y (r(x, y) \rightarrow d(y))) \tag{135}$$

In the car example this become:

$$\forall x (\mathsf{ElectricCar}(x) \rightarrow \forall y (\mathsf{hasEngine}(x, y) \rightarrow \mathsf{electricEngine}(y)))$$

$$\forall x (\mathsf{car}(x) \wedge \neg\mathsf{ElectricCar}(x) \rightarrow \forall y (\mathsf{hasEngine}(x, y) \rightarrow \mathsf{carbonFuelEngine}(y)))$$

3.3. Cardinality constraints on relations. In ontologies we often need to state that a concept instance is related with a (more than, less then or exactly a) number of instances with another concept. For example we want to state that a car has exactly four wheels, that a group has at least two members and that a car cannot have more than 7 seats. This can be done with the following FOL axiom

At least n

$$\forall x \left(c(x) \rightarrow \exists y_i \dots \exists y_n \left(\bigwedge_{i=1}^{n} \left(r(x, y_i) \wedge d(y_i) \wedge \bigwedge_{j=1}^{i-1} x_i \neq x_j \right) \right) \right) \tag{136}$$

At most n

$$\forall x \left(c(x) \rightarrow \forall y_1 \dots \forall y_{n+1} \left(\bigwedge_{i=1}^{n} (r(x, y_i) \wedge d(y_i)) \rightarrow \bigvee_{j=1}^{n} x_i = x_{n+1} \right) \right) \tag{137}$$

3.4. Concept definition in FOL. As a final aspect of ontologies we consider the notion of *defined concepts*. In an ontology one can distinguish between *primitive concepts* and *primitive relations* and *defined concepts* and *defined relations*.

For primitive concepts we don't need to provide a definition in terms of other concepts. For instance the concept of electic car can be defined as a car with one electic engine, hybrid car is a car that has one electric and one carbon fuel engine. As follows

$$\begin{aligned} \forall x(\mathsf{electricCar}(x) &\leftrightarrow \mathsf{car}(x) \wedge \forall y(\mathsf{hasEngine}(y) \rightarrow \mathsf{electricEngine}(y)) \\ \forall x(\mathsf{hybridCar}(x) &\leftrightarrow \mathsf{car}(x) \wedge \exists y(\mathsf{hasEngine}(y) \wedge \mathsf{electricEngine}(y)) \wedge \\ &\qquad \exists y(\mathsf{hasEngine}(y) \wedge \mathsf{carbonFuelEngine}(y))) \\ \forall x(\mathsf{carbonCar}(x) &\leftrightarrow \mathsf{car}(x) \wedge \forall y(\mathsf{hasEngine}(y) \rightarrow \mathsf{carbonFuelEngine}(y)) \end{aligned}$$

We also need to add the axiom that states that a car has at least one engine. otherwise a car without engine will be both an electic and a carbon car.

$$\forall x(\mathsf{car}(x) \rightarrow \exists y(\mathsf{hasEngine}(y) \wedge \mathsf{carbonFuelEngine}(y) \vee \mathsf{electicEngine}(y)))$$

3.5. Reasoning with FOL ontologies. to be finished

4. First order theories for commonsense

to be finished see `https://cs.nyu.edu/~davise/guide.html` Guide to Axiomatizing Domains in First-Order Logic

5. Exercises

Exercise 148:

Formalize in first order logic the following english statements using the following predicates

$B(x, y, z)$	y is between x and z
$S(x)$	x is a square
$T(x)$	x is a triangle
$C(x)$	x is a circle
$A(x, y)$	x is above y

(1) If a circle is between two squares then it is above some triangle
(2) two triangle cannot be one above the other unles one of them is between two squares;
(3) A circle must be always between two triangles;
(4) it is not possible that an object is between two distinct pairs of objects.

Solution

(1) If a circle is between two squares then it is above some triangle

$$\forall xyz(B(x, y, z) \wedge S(x) \wedge C(y) \wedge S(z) \rightarrow \exists w(A(y, w) \wedge T(w)))$$

(2) two triangle cannot be one above the other unless one of them is between two squares;

$$\forall xy(T(x) \wedge T(y) \wedge A(x, y) \rightarrow \exists zw(S(z) \wedge S(w) \wedge (B(z, x, w) \vee B(z, y, w))))$$

(3) A circle must be always between two triangles;

$$\forall x(C(x) \rightarrow \exists yz(T(y) \wedge T(z) \wedge B(y, x, z)))$$

(4) it is not possible that an object is between two distinct pairs of objects.

$$\forall xyz(B(x, y, z) \wedge B(x', y, z') \rightarrow x = x' \wedge z = z')$$

□

CHAPTER 10

Herbrand Theorem and Skolemization

To check satisfiability of a first order sentence ϕ on the signature Σ we have to produce an Σ-structure $\mathcal{I}$ that satisfies ϕ, i,e $\mathcal{I} \models \phi$. The naive procedure used for propositional logic, in which we check for all possible interpretations, is not working for FOL, since there are infinite many interpretations. Indeed we are free to choose the interpretation domain $\Delta^{\mathcal{I}}$ with any possibly infinite set, and therefore we have infinite possibilities to interpret the symbols of Σ.

The question, is whether there is a sistematic method to generate the Σ-strucutre for ϕ such that if ϕ is satisfiable, sooner or later we will encounter an interpretation that satisfies it.

The Herbrand's Theorem, called so after Jacques Herbrand (1908-1931), allows to reduces the problem of checking the satisfiability of a first-orde formula to the check of satisfiability of a set of propositional formulas. In this chapter we gradually introduce the theorem.

To keep the treatment simle in this chapter we consider only the case of First order language without equality.

1. Herbrand interpretation

Herbrand proposes the main idea to interpret terms in themselves. Notice that the definiton of Σ-structure $(\Delta^{\mathcal{I}}, \mathcal{I})$ $\Delta^{\mathcal{I}}$ can be any non empty set. Herbrand poposed to consider $\Delta^{\mathcal{I}}$ as the set of all ground terms that can be built from the signature Σ. Since $\Delta^{\mathcal{I}}$ must contain at least one elment, Herbrand required that Σ contains at least one constant symbol.

Definition 10.1 (Herbrand Universe). *The* Herbrand's universe *of a signature Σ that contains at least one constant symbol, is the set, denoted by $\Delta^{\mathcal{H}}$ of ground terms of Σ.*

Example 10.1. *If Σ contains two constants a and b and no function symbol then, the Herbrand's Universe of Σ is $\{a, b\}$ since a and b are the only ground terms that one can build in Σ. If, instead Σ contains a binary function symbo f then the set of ground terms, and therefore the Herbrand's Univers of Σ contains an infinite*

set of terms. i.e.,

a	b		
$f(a,a)$	$f(a,b)$	$f(b,a)$	$f(b,b)$
$f(a,f(a,a))$	$f(a,f(a,b))$	$f(a,f(b,a))$	$f(a,f(b,b))$
$f(b,f(a,a))$	$f(b,f(a,b))$	$f(b,f(b,a))$	$f(b,f(b,b))$
$f(f(a,a),a)$	$f(f(a,b),a)$	$f(f(b,a),a)$	$f(f(b,b),a)$
$f(f(a,a),b)$	$f(f(a,b),b)$	$f(f(b,a),b)$	$f(f(b,b),b)$
$f(f(a,a),f(a,a))$	$f(f(a,a),f(a,b))$	$f(f(a,a),f(b,a))$	$f(f(a,a),f(b,b))$
$\ldots$			

One can easily se that with one constant and a function symbol the Herbrand's Universe is infinite. Instead if there is no function symbols then the Herbrand universe has the same size of the number of constants in Σ.

An alternative way to define the Herbrand's Univers for Σ is by induction i.e., the herbrand universe $\Delta_\Sigma^{\mathcal{H}}$ for Σ is the smalles set that satisfies the following conditions

(1) Every constant of Σ belongs to $\Delta_\Sigma^{\mathcal{H}}$
(2) if $t_1, \ldots, t_n \in \Delta_\Sigma^{\mathcal{H}}$ and f is an n-ary function symbol of Σ, then $f(t_1, \ldots, t_n) \in \Delta_\Sigma^{\mathcal{H}}$.

Once we have defined the set $\Delta^{\mathcal{H}}$ to fully define an interpretation, we have to specify the interpretation function for the elements of Σ. The obvious way is to define the interpretation of constants and function symbols so that every terms is interpreted in itself, and every predicate with arity equal to n as a set of n-tuples of terms. i.e., in a subset of $\Delta_\Sigma^{\mathcal{H}}$.

DEFINITION 10.2. *An herbrand interpretation of a signature Σ is composed by the pair $(\Delta_\Sigma^{\mathcal{H}}, \mathcal{H})$, where*

(1) $\Delta_\Sigma^{\mathcal{H}}$ is the Herbrand's universe of Σ;
(2) $\mathcal{H}(c) = c$ for every constant symbol $c \in \Sigma$;
(3) $\mathcal{H}(f) : t_1, \ldots, t_n \mapsto f(t_1, \ldots, t_n)$ is the function that maps an n-tuple of terms of $\Delta_\Sigma^{\mathcal{H}}$ in a term of Δ_Σ^{H}, for every n-ary function symbol f;
(4) $\mathcal{H}(P) \subseteq (\Delta_\Sigma^{H})^n$ is a set of n-tuples of terms in $\Delta_\Sigma^{\mathcal{H}}$, for evert n-ary predicate symbol $P \in \Sigma$.

A simpler way to see an Herbrand interpretation is by seeing it as a mapping from ground atomic formulas to $\{0, 1\}$.

$$\mathcal{H} : \text{GroundAtoms}(\Sigma) \to \{0, 1\} \tag{138}$$

This definition is very close to the definition of propositional interpretation, where GroundAtoms(Σ) is the set of proposiional variabels. The set GroundAtoms(Σ) is called the *Herbrand's base for* Σ.

EXAMPLE 10.2. *The following is an example of an Herbrand Interpretation that satisfies the following set of formulas:*

$$\Gamma = \left\{ \begin{array}{c} \neg \mathit{friend}(x,x) \\ \mathit{friend}(x,y) \to \mathit{friend}(x,y) \\ \mathit{friend}(x,y) \to \mathit{knows}(x, \mathit{mother}(y)) \\ \mathit{friend}(\mathit{Mary}, \mathit{John}) \end{array} \right\}$$

$$\Delta_\Gamma^{\mathcal{H}} = \left\{ \begin{array}{l} \mathit{Mary}, \mathit{John}, \\ \mathit{mother}(\mathit{Mary}), \mathit{mother}(\mathit{John}), \\ \mathit{mother}(\mathit{mother}(\mathit{Mary})), \mathit{mother}(\mathit{mother}(\mathit{John})) \\ \mathit{mother}(\ldots \mathit{mother}(\mathit{Mary}) \ldots), \ \mathit{mother}(\ldots \mathit{mother}(\mathit{John}) \ldots), \ldots \end{array} \right\}$$

$$\mathcal{H} = \left\{ \begin{array}{l} \mathit{friend}(\mathit{John}, \mathit{Mary}), \ \mathit{friend}(\mathit{Mary}, \mathit{John}), \\ \mathit{knows}(\mathit{John}, \mathit{mother}(\mathit{Mary})), \\ \mathit{knows}(\mathit{Mary}, \mathit{mother}(\mathit{John})), \\ \mathit{knows}(\mathit{mother}(\mathit{Mary}), \mathit{mother}(\mathit{John})) \end{array} \right\}$$

2. Satisfiability in Herbrand's Interpretation

Satisfiability in Herbrand interpretations is defined as the problem of checking if a formula ϕ is satisfiable by an Herbrand's Interpretation on the signature of ϕ. One of the main version of the Herbrand's theorem states that satisfiability in general, can be reduced to satisfiability by an Herbrand's interpretation

PROPOSITION 10.1. *If $\mathcal{H}$ is an Herbrand interpretation then for every ground term t $\mathcal{H}(t) = t$.*

PROOF. By induction on the complexity of t. If t is the constant c then $\mathcal{H}(c) = c$ by definition. If $t = f(t_1, \ldots, t_n)$ then

$$\begin{array}{rll} \mathcal{H}(f(t_1, \ldots, t_n)) & = \mathcal{H}(f)(\mathcal{H}(t_1), \ldots, \mathcal{H}(t_n)) & \\ & = \mathcal{H}(f)(t_1, \ldots, t_n) & \text{By induction hypothesis} \\ & = f(t_1, \ldots, t_n) & \text{By definition } \mathcal{H}(f) \end{array}$$

□

PROPOSITION 10.2. *$\mathcal{H} \models \phi(x)[a_{x \leftarrow t}]$ iff $\mathcal{H} \models \phi(t)$*

PROOF. By induction on the complexity of ϕ (exercize) □

PROPOSITION 10.3. *$\mathcal{H} \models \forall x\, \phi(x)$ if and only if $\mathcal{H} \models \phi(t)$ for all ground term t.*

PROOF.

$$\begin{array}{r} \mathcal{H} \models \forall x\, \phi(x) \text{ fff } \mathcal{H} \models \phi(x)[a_{x \leftarrow t}] \text{ for all ground terms } t \\ \text{fff } \mathcal{H} \models \phi(t) \text{ for all ground terms } t \end{array}$$

□

DEFINITION 10.3 (quantifier-free formula). *A formula ϕ is quantifier-free if ϕ has no occurrence of either of the quantifiers $\forall$ or $\exists$.*

Notice that a quantifier-free formula is the combination of a set of atoms using the propositional connectives. Notice that all the individual variables that occours in a quantifier-free formula are free. Furthermore if a uantified free formula do not contains individual variables, then it is just a propositional formula.

EXAMPLE 10.3. *The following are examples of quantified free formulas.*

$$P(a) \vee Q(b, x) \rightarrow R(x, y, z)$$
$$R(a, b, f(c)) \vee R(b, a, g(a, b))$$

the second one does not contains individual variables, hence it is a propositional formula.

If we quantify universally the free variables of a quantified free formula we obtain a *universal sentence.*

DEFINITION 10.4 (Universal sentence). *A* universal sentence *is a sentence (closed formula of the form*

$$\forall x_1 \forall x_2 \dots \forall x_n \, \phi(x_1, \dots, x_n)$$

where $\phi(x_1, \dots, x_n)$ is a quantifier-free formula.

In other words universal sentences admit only universal quantifiers at the beginnin of the formula. We will see later that every formula can be transformed in an equi-satisfiable universal sentence. If we instantiate every variable of a universal sentence we obtain a propositional formula, that is called a ground ifnstance of the universal sentence.

DEFINITION 10.5 (Ground instance). *A* ground instance *of an universal sentence $\forall x_1 \dots \forall x_n . \phi(x_1, \dots, x_n)$ is a sentence $\phi(t_1, \dots, t_n)$ obtained by replacing each occurrence of x_i with a term t_i that does not contain variables.*

THEOREM 10.1 (Herbrand's Theorem). *A universal formula $\Phi = \forall x_1, \dots, \forall x_n \phi(x_1, \dots, x_n)$ is satisfiable if and only if it is true in an Herbrand interpretation in the signature of ϕ (if ϕ does not contain any constant we extend the signature with a constant a)*

PROOF (SKETCH). If Φ is satisfiable by an Herbrand interpretation then it is satisfiable. Let us proove the contrary. Suppose that Φ is satisfied by the interpretation $\mathcal{I}$. Starting from $\mathcal{I}$ we can build the following herbrand interpretation $\mathcal{H}$, on the domain of ground terms $\Delta_\Sigma^{\mathcal{H}}$ where Σ is the signature of Φ possibly extended with a constant a if Φ does not contain constant symbols. For every n-ary predicate p we define $\mathcal{H}(p)$

$$\mathcal{H}(p) = \{(t_1, \dots, t_n) \in \Delta_\Sigma^{\mathcal{H}} \mid \mathcal{I} \models p(t_1, \dots, t_n)\}$$

For every formula $\phi(x_1, \dots, x_n)$ we can prove by induction that the formula $\forall x_1, \dots, x_n \phi(x_1, \dots, x_n) \rightarrow \phi(t_1, \dots, t_n)$ is valid for every n-tuple of ground terms. This implies that

$$\mathcal{I} \models \phi(t_1, \dots, t_n) \tag{139}$$

and therefore that $\mathcal{H} \models \phi(t_1, \dots, t_n)$. This implies that

$$\mathcal{H} \models \phi(x_1, \dots, x_n)[a_{x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n}]$$

and therefore that $\mathcal{H} \models \Phi$. □

The immediate consequence of the Herbrand's theorem is that, to check if $\Phi = \forall x_1, \dots, x_n \, \phi(x_1, \dots, x_n)$ is satisfiable we can check if it is satisfiable only in the herbrand interpretations. If there is no herbrand interpretations that satisfies Φ then the formula is surely unsatisfiable.

A second, and related consequence, is that if Φ is unsatisfiable, then also the set

$$\text{Ground}(\Phi) = \{\phi(t_1, \dots, t_n) \mid t_i \in \Delta_\Sigma^{\mathcal{H}}\}$$

is not sartistiable. But one can notice that Ground(Φ) is a set of propositional formula, and therefore we can apply the main results of satisfiability in proposiitional

formula. In particulare, we use the compactenss theorem (Theorem **??**) that states that an infinite set of proposiitonal formula Γ is not satisfiable if and only if there is a finite subset Γ_0 of Γ that is not satisfiable. We can therefore conclude that Φ is unsatisfiable there is a finite set $G \subset \text{Ground}(\Phi)$ of groundings of Φ that is unsatisfiable.

If we find a way to enumerate $G_0, G_1, G_2, \dots$ (i.e., generate an infinite sequence) of all the finite subsets, of Ground(Φ) such that for every finite subset $G \subset \text{Ground}(\Phi)$ there is an i such that $C_i = C$ we could check at every iteration if G_i is satisfiable, and if Ground(Φ) is not satisfiable we enventuall find an i such that G_i is not satisfiable. This very naive idea is implemented in Algorithm 12. If

Algorithm 12 First Order Satisfiability,

Require: A universal formula $\Phi = \forall x_1, \dots, \forall x_n\, \phi(x_1, \dots, x_n)$
$\Sigma \leftarrow$ the signature of Φ
if Constants(Σ) $= \emptyset$ **then**
    $\Sigma \leftarrow \Sigma \cup \{a\}$
end if
$\Delta \leftarrow$ Constants(Σ)
while True **do**
    $G \leftarrow$ Ground(Φ, Δ)
    if PropositionalSat(G)=Unnsat **then**
        return Unsat
    end if
    $\Delta \leftarrow \Delta \cup \{f(t_1, \dots, t_n) \mid f \in n\text{-ary-Funct}(\Sigma),\ t_i \in \Delta\}$
end while

Φ is unsat, then by the Herbrand theorem we have that there is a finite subset of Grounding(Φ) that is unsat let k be the masimum dephth of the terms that appear in G, then at the k-th iteration the set G will be a subset of Grounding(Φ, Δ) which sill be inconsistent, and therefore the algorithm terms returning Unsat

3. Prenex normal form

In the previous section we only consider universally quantified formulas. In this section we show how to extend this result to the entire set of first order formulas, that inculdes also existential quantified formulas.

Definition 10.6. *A formula is in* prenex normal form *if it is in the form of*

$$Q_1 x_1 \dots, Q_n x_n\, \phi(x_1, \dots, x_n) \tag{140}$$

where each Q_i is either $\exists$ or $\forall$ and $\phi(x_1, \dots, x_n)$ is a quantified free first order formula.

Every formula can be reduced in prenex normal form by using the following rewriting rules:

- rewrite the $\to$ and $\leftrightarrow$ in terms of $\neg$ and $\vee$ and $\wedge$;
- switch the $\neg$ and the quantifiers with the rule:

$$\neg\forall x\phi \Longrightarrow \exists x \neg\phi$$
$$\neg\exists x\phi \Longrightarrow \forall x \neg\phi$$

- switch the binary connectives $\wedge$ and $\vee$ and the quantifier with the rules under the hypothesis that x does not

$$\forall x\, \phi(x) \wedge \psi \Longrightarrow \forall x(\phi(x) \wedge \psi)$$
$$\forall x\, \phi(x) \vee \psi \Longrightarrow \forall x(\phi(x) \vee \psi)$$
$$\exists x\, \phi(x) \wedge \psi \Longrightarrow \exists x(\phi(x) \wedge \psi)$$
$$\exists x\, \phi(x) \vee \psi \Longrightarrow \exists x(\phi(x) \vee \psi)$$

  If x appears free in ψ we can rewrite $Qx\, \phi(x)$ into the equivalent formula $Qy\, \phi(y)$ for some new variable y before applying the rules.
- the following rules can also be used but not strictly necessary

$$\exists x\, \phi(x) \vee \exists x \psi(x) \Longrightarrow \exists x(\phi(x) \vee \psi(x))$$
$$\forall x\, \phi(x) \wedge \forall x \psi(x) \Longrightarrow \forall x(\phi(x) \wedge \psi(x))$$

- finally it is possible to switch the existential and universal quantifier with the following rule

$$\forall x \exists y(\phi(x) \circ \psi(y)) \Longrightarrow \exists y \forall x(\phi(x) \circ \psi(y))$$

  if x is not free in $\psi(y)$ and y is not free in $\phi(x)$. As it will be clearer later, moving the existential quantifier out of the scope of an universal quantifier can be convenient.

Let us see an example about how to rewrite a formula in prenex normal form

EXAMPLE 10.4. *Consider the formula*

$$(\forall x \exists y P(x, y) \to \exists x Q(x)) \vee \forall x Q(x)$$

We first rewrite the $\to$

$$(\neg \forall x \exists y P(x, y) \vee \exists x Q(x)) \vee \forall x Q(x)$$

Then we push the $\neg$ *in front of atoms*

$$(\exists x \forall y \neg P(x, y) \vee \exists x Q(x)) \vee \forall x Q(x)$$

Then we can apply the rule that commutes $\exists x$ *and* $\vee$ *on the first disjunct*

$$\exists x(\forall y \neg P(x, y) \vee Q(x)) \vee \forall x Q(x)$$

and push out the $\forall x$ *quantifier*

$$\exists x \forall y(\neg P(x, y) \vee Q(x)) \vee \forall x Q(x)$$

We can also push out the first existential quantifier since x *is not free in* $\forall x Q(x)$

$$\exists x \forall y(\neg P(x, y) \vee Q(x) \vee \forall x Q(x))$$

Now if we want to push out the quantifier $|forallx$ *since* x *is free in* $\neg P(x, y) \vee Q(x)$ *we have to rename the variablel obtaining*

$$\exists x \forall y(\neg P(x, y) \vee Q(x) \vee \forall z Q(z))$$

now we can apply the rule to obtain

$$\exists x \forall y \forall z(\neg P(x, y) \vee Q(x) \vee Q(z))$$

which is in prenex normal form.

4. Skolemization

Skolem normal form is named after the late Norwegian mathematician Thoralf Skolem.(1887–1963). Skolemization is the operator of replacing existential quantifiers either with constants (0-ary functions) or with functions, obtaining an equisatisfiable formula.

Before providing the general definition let us consider the following siple example of proposition in FOL. Consider the proposition "Every preogrammer has written at least one computer program", In FOL this can be formalized as

$$\forall x(\mathsf{Programmer}(x) \to \exists y(\mathsf{Program}(y) \wedge \mathsf{Author}(x, y))$$

If it is the case that for every programmer we can find a program written by him/her, there exists a function from programmers to programs that selects one program for every programmer, and such that the author of the program selected by this function for some programmer x is x him/herself. Notice that there might be more than one program written by the same programmer, however f will pik one of them. To formalize this line of reasoning, we can extend the signature with a new symbol f that intuitively represent the funciton that select one program for every programmer, and we use f in place of the existential quantifier, by rewriting the original formula in

$$\forall x(\mathsf{Programmer}(x) \to \mathsf{Program}(\mathsf{f}(x)) \wedge \mathsf{Author}(x, \mathsf{f}(x)))$$

The previous example can be generalized by rewriting any formula of tghe form $\forall x \exists y \phi(x, y)$ in $\forall x \phi(x, f(x))$ for some new function symbol f. This is also possible when there is no universal quantifier in front of $\exists$. I.e., The formula $\exists x \phi(x)$ can be rewritten in $\phi(a)$ for some new constant a. Generalizing even more the formula $\forall x \forall y \exists z \phi(x, y, z)$ can be rewritten in $\forall x \forall y \phi(x, y, f(x, y))$ for some new binary function symbol f. Let us make this process fully general.

Definition 10.7 (Skolemization). *Let Φ be a formula in prenex normal form that start with m universal quantifiers followed by an existential quantifier. I.e., Φ is in the form:*

$$\forall x_1 \forall x_2 \dots \forall x_m \exists x_{m+1} Q_{m+1} x_{m+2} \dots Q_n x_n \phi(x_1, \dots, x_n)$$

a formula in prenex normal form the Skolemization if the operation of introducing a new n-ary function symbol f and replace x_{m+1} with $f(x_1, \dots, x_m)$, and remove the existential quantifier. I.e., transforming the formula in

$$\forall x_1 \forall x_2 \dots \forall x_m Q_{m+2} x_{m+2} \dots Q_n x_n \phi(x_1, \dots, x_m, f(x_1, \dots, x_m), x_{m+2}, \dots, x_n)$$

Proposition 10.4. *Let Ψ be the Skolemization of a forula Φ Every model that satisfies Φ can be extended to an interpretation $\mathcal{I}$ by providing the interpretation of the skolem function f that satisfies Ψ.*

Proof. We prove the property for the special case where Φ is $\forall x \exists y R(x, y)$, The general proof looks the same. In this special case Ψ, the skolemization of Φ is $\forall x R(x, f(x))$. Let us show that Φ is satisfiable iff Ψ is satisfiable

($\Longrightarrow$) If $\forall x \exists y R(x, y)$ is satisfiable, then there is an interpretation $\mathcal{I}$, such that $\mathcal{I} \models \forall x \exists y\, R(x, y)$. This implies that, for every element $d \in \Delta^{\mathcal{I}}$, there is an elmenet $d' \in \Delta^{\mathcal{I}}$ such that $(d, d') \in \mathcal{I}(R)$. Let $\mathcal{I}'$ be the interpretation on the same domain of $\mathcal{I}$ with $\mathcal{I}(R) = \mathcal{I}'(R)$ and $\mathcal{I}'(f)$ is a function that maps d into a d' such that

$(d, d') \in \mathcal{I}(R)$. This implies that for every $d \in \Delta^{\mathcal{I}'}$, $\mathcal{I}' \models R(x, f(x))[x \leftarrow d]$; and therefore that $\mathcal{I}' \models \forall x R(x, f(x))$.

($\Longleftarrow$) If $\forall x R(x, f(x))$ is satisfiable, there is an interpretation $\mathcal{I}$ of R and f such that for every $d \in \Delta^{\mathcal{I}}$, $(d, \mathcal{I}(f)(d)) \in \mathcal{I}(R)$ and therefore for every $d \in \Delta^{\mathcal{I}}$ there is a d' (which is $\mathcal{I}(f)(d)$) such that $(d, d') \in \mathcal{I}$. This implies that $\mathcal{I} \models \forall x \exists y R(x, y)$. If we consider $\mathcal{I}'$ the restriction of $\mathcal{I}$ to the signature that contains only R, we have that $\mathcal{I}' \models \forall x \exists y R(x, y))$ and therefore that $\forall x \exists y\, R(x, y)$ is satisfiable. $\square$

5. Exercises

Exercise 149:

Let Σ be a signature that contains the constants a and b and no function symbols, List all the ground instances of the following two universal sentences

(1) $\forall x \forall y\, P(x, y) \to R(x)$
(2) $\forall x\, P(x, x) \to R(x)$

Exercise 150:

Consider the universal formulas

$$\forall x \forall y \phi(x, y))$$
$$\forall x \forall y \phi(g(x, y), f(x))$$

What is the relationship between the set of grounding of the two formulas?

Exercise 151:

Prove that the following reqriting rules transform a formula in another formula which is equivalent to the original formula. More formally for every rule $A \Longrightarrow B$ you have to show that $A \models B$ and $B \models A$

$$\begin{aligned}
\neg\forall x\phi &\Longrightarrow \exists x\phi \\
\neg\exists x\phi &\Longrightarrow \forall x\phi \\
\exists x\, \phi(x) \vee \exists x\psi(x) &\Longrightarrow \exists x(\phi(x) \vee \psi(x)) \\
\forall x\, \phi(x) \wedge \forall x\psi(x) &\Longrightarrow \forall x(\phi(x) \wedge \psi(x)) \\
\forall x\, \phi(x) \wedge \psi &\Longrightarrow \forall x(\phi(x) \wedge \psi) && (*) \\
\forall x\, \phi(x) \vee \psi &\Longrightarrow \forall x(\phi(x) \vee \psi) && (*) \\
\exists x\, \phi(x) \wedge \psi &\Longrightarrow \exists x(\phi(x) \wedge \psi) && (*) \\
\exists x\, \phi(x) \vee \psi &\Longrightarrow \exists x(\phi(x) \vee \psi) && (*) \\
\forall x \exists y(\phi(x) \circ \psi(y)) &\Longrightarrow \exists y \forall x(\phi(x) \circ \psi(y)) && (**)
\end{aligned}$$

where $(*)$ means that x is not free in ψ and $(**)$ that x is not free in $\psi(y)$ and y is not free in $\phi(x)$. For the rules that have application restrictions $(*)$ and $(**)$, find an example of a formula that does not satisfies the restriction and that is not equivalent to the formula obtained by applying the transformation.

Exercise 152:

Find the prenex normal form of

$$\forall x(\exists y R(x, y) \wedge \forall y \neg S(x, y) \to \neg(\exists y R(x, y) \wedge P))$$

Solution

$$\forall x \forall y_1 \exists y_2 \forall y_3(\neg R(x, y_1) \vee S(x, y_2) \vee \neg R(x, y_3) \vee \neg P)$$

□

Exercise 153:

Transform the following formula into prenex normal form:

$$\neg[\forall x \exists y F(u,x,y) \to \exists x(\neg \forall y G(y,v) \to H(x))]$$

Solution

$$\forall x \exists y \exists z [F(u,x,y) \wedge \neg G(z,v) \wedge \neg H(x)]$$

□

Exercise 154:

Which of the following expressions are in prenex normal form?

(1) $\forall x\, P(x) \vee \forall x\, Q(x)$
(2) $\forall x \forall y\, \neg(P(x) \to Q(y))$
(3) $\forall x \exists y R(x,y)$
(4) $R(x,y)$
(5) $\neg \forall x\, R(x,y)$

Exercise 155:

Let Σ be the signature of a formula $\phi(x,y)$, Explain how a Σ-structure $\mathcal{I}$ that satisfies $\forall x \exists y \phi(x,y)$ can be extended in a Σ'-structure $\mathcal{I}'$ on the signature Σ' obtained by extending Σ with a new unary function symbol f, such that $\mathcal{I}' \models \forall x\, \phi(x, f(x))$.

Exercise 156:

Consider the followin english sentences which that involves existential quantifiers. Translate them in FOL using universa dna existential quantifier; trasnform them in prenex normal form and then in skolemized prenex normal form.

(1) Every philosopher writes at least one book;
(2) All students of a philosopher read one of their teacher's books;
(3) There exists a philosopher with students.

Solution

(1) Every philosopher writes at least one book.

$$\forall x(\mathsf{Phil}(x) \to \exists y(\mathsf{Book}(y) \wedge \mathsf{Writes}(x,y)))$$

Rewrige Implication:

$$\forall x(\neg \mathsf{Phil}(x) \vee \exists y(\mathsf{Book}(y) \wedge \mathsf{Writes}(x,y)))$$

Transform in prenex normal form

$$\forall x \exists y(\neg \mathsf{Phil}(x) \vee (\mathsf{Book}(y) \wedge \mathsf{Writes}(x,y)))$$

Skolemize: substitute $\exists y$ with $g(x)$

$$\forall x(\neg \mathsf{Phil}(x) \vee (\mathsf{Book}(g(x)) \wedge \mathsf{Writes}(x, g(x))))$$

(2) All students of a philosopher read one of their teacher's books.

$$\forall x \forall y(\mathsf{Phil}(x) \wedge \mathsf{StudentOf}(y,x) \to \exists z(\mathsf{Book}(z) \wedge \mathsf{Writes}(x,z) \wedge \mathsf{Reads}(y,z)))$$

Eliminate Implication:

$$\forall x \forall y(\neg \mathsf{Phil}(x) \vee \neg \mathsf{StudentOf}(y,x) \vee \exists z(\mathsf{Book}(z) \wedge \mathsf{Writes}(x,z) \wedge \mathsf{Reads}(y,z)))$$

Transform in prenex normal form

$$\forall x \forall y \exists z (\neg \mathsf{Phil}(x) \vee \neg \mathsf{StudentOf}(y, x) \vee (\mathsf{Book}(z) \wedge \mathsf{Writes}(x, z) \wedge \mathsf{Reads}(y, z)))$$

Skolemize: substitute $\exists z$ by $h(x, y)$

$$\forall x \forall y (\neg \mathsf{Phil}(x) \vee \neg \mathsf{StudentOf}(y, x) \vee (\mathsf{Book}(h(x, y)) \wedge \mathsf{Writes}(x, h(x, y)) \wedge \mathsf{Reads}(y, h(x, y))))$$

(3) There exists a philosopher with students.

$$\exists x \exists y (\mathsf{Phil}(x) \wedge \mathsf{StudentOf}(y, x))$$

Skolemize: substitute $\exists x$ by a and $\exists y$ by b

$$\mathsf{Phil}(a) \wedge \mathsf{StudentOf}(b, a)$$

□

Exercise 157:

Trensform the formula $\neg(\exists x(P(x) \rightarrow \forall x P(x)))$ in skolemized prenex normal form.

points**Exercise 158:** (4 points Pt.)

Let Σ be the signature of a formula $\phi(x, y)$, Explain how a Σ-structure $\mathcal{I}$ that satisfies $\forall x \exists y \phi(x, y)$ can be extended in a Σ'-structure $\mathcal{I}'$ on the signature $\Sigma' = \Sigma \cup \{f/1\}$ obtained by extending Σ with a new unary function symbol f, such that $\mathcal{I}' \models \forall x \phi(x, f(x))$. How is it called this operation?

Solution The transofrmation described in the exercize is called Skolemization. Skolemization allow to eliminate quantifiers by introducing new constant or function symbolis. It is proved that Skolemizaiton perserves satifiability. I.e., the original formula is satifiable if and only if the SKolemized formula does.

Let us now solve the exercize. Our objective is to define an interpretation of f, i.e., $\mathcal{I}'(f)$ such that $\mathcal{I}' \models \forall x \phi(x, f(x))$. This means that for every element $d \in \Delta^{\mathcal{I}}$ we have to find a d' such that $\mathcal{I}'(f)(d) = d'$ and such that $\mathcal{I} \models \phi(x, y)[x \leftarrow d, y \leftarrow d']$. Notice that for every element d of the interpretation domain $\Delta^{\mathcal{I}}$, the fact that $\mathcal{I} \models \forall x \exists y \phi(x, y)$. implies that $\mathcal{I} \models \exists y \phi(x, y)[x \leftarrow d]$. This implies that there is a d' such that $\imath \models \phi(x, y)[x \leftarrow d, y \leftarrow d']$. Let us define the interpretation of the function symbol $\mathcal{I}'(f) : d \mapsto d'$. Then we have that $\mathcal{I}' \models \phi(x, f(x))[x \leftarrow d]$. If we repreat this reasoning for all the d of the domain we have a complete definition of $\mathcal{I}'(f)$, such that $\mathcal{I} \models \phi(x, f(x))[x \leftarrow d]$ for all individuals $d \in \Delta^{\mathcal{I}}$. From this we can conclude that $\mathcal{I}' \models \forall x \phi(x, f(x))$. □

CHAPTER 11

Satisfiability in First Order Logic

In this chapter we consider the problem of satisfiability in first order logic. I.e., we introduce an algorithm that check if a first order sentence ϕ is satisfiable. We consider the more general formulation of checking if a set of first order sentencs Γ is satisfiable. As for propositiona logic, the problem of checking validity and logical consequence in first order logic can be reduced to satisfiability. Therefore, a procedure for checking satisfiability can be used also for checking if a formula ϕ is valid, by checking satisfiabilty of $\neg\phi$ and if ϕ is a logical consequence of a set of formulas Γ by checking if $\Gamma \cup \{\neg\phi\}$ is not satisfiable. However, we start with a bad news. Indeed it has been proved that such an algorithm does not exist.

Satisfiability in first order logic is *semi-decidable*. Which means that any algorithm that checks satisfiabil8ty of a first order sentence will eventually terminate if the formula is not satisfiable but might run infinitely if the formula is satistiable. The proof (which is out of the scope of this notes) is based on the fact that the *halting problem* can be encoded in first order formula ϕ and if we have an algorithm that check satisfiability of ϕ this means that we have an algorithm that decided the halting problem.

However the fact that when a formula is not valid the algirhtm might not terminate, does not prevent us in searching for a countermodel of such a forula. If such a countermodel is found then we can conclude that the formula is not valid. However, we cannot avoit the fact that the search of such a ountermodel can go on forever.

The algorithm for checking satisfiability of first order sentences is deebly based on the Herbrand theorem presented in the previous chapter. The Herbrand theorems thells us that if a set of univerally quantified formulas Γ is not satisfiabile, then there is a finite subset of ground formulas of Γ which are not satisfied. Therefore our algorithm will search for such a finite subset of formulas. THe algorithms is based on two main stesp, called *resolution* and *unification*. We describe them separately in the following two sections, and in the final seciton we show how they can jointly use to chesk satisfiability in first order logic.

1. Finite Model Finding

Given a set of first order sentences to check its satisfiability, we can search a finite model, i.e., a Σ-structure on a finite domain. There are many mehods for performing finite model finding see for instance Reger, Suda, and Voronkov 2016; Torlak and Jackson 2007; McCune 2003; J. Zhang and H. Zhang 1996; Baumgartner et al. 2009

A basic method for finite model finding is to iteratively search for models with increasing domain size starting from 1. That is, the search is started for models of domain size 1 only. If that fails, the search is repeated for a model of size 2, then of

3, and so on. If a formula has a finite model in theory it will be eventually found. Clearly if the fomrula has only infinite models then the procedure will continue indefinitely. Despice it's simplicity most of the methods follows this approach. One of the first systems for FOL finite model finding is called MACE4McCune 2003.

MACE4 transform the problem of finding a model of size n of a first order sentence ϕ by doing the following three steps:

(i) transform ϕ in prenex univerally quantified conjunctive normal form (performing also Skolemization)
(ii) ground the obtained formula in the finite set of constant $\{1, \ldots, n\}$
(iii) Enrich the set of ground clauses with clauses for equality for every term that appear in the ground clauses which is different from $1, \ldots, n$
(iv) Search for a propoisitional assignment that satisfies the entire set of clauses.
(v) If such an assignments is found, extract the first order model from it.

EXAMPLE 11.1. *To generate a model of size 2 for the formula* $\forall x(P(c,x) \to \neg Q(c,x)) \wedge \forall x \forall y(\neg P(x,y) \to \exists z Q(x,z))$*, it is first translated in the clause clause*

$$\{\neg P(c,x), \neg Q(c,x)\} \qquad \{P(x,y), Q(f(x),y)\}$$

The clause is then grounded w.r.t. the domain $\{1,2\}$ *obtainining the set of ground clausees*

$$\begin{array}{lll} \{\neg P(c,1), \neg Q(c,1)\} & \{p(1,1), q(f(1),1)\} & \{p(1,2), q(f(1),2)\} \\ \{\neg P(c,2), \neg Q(c,2)\} & \{p(2,1), q(f(2),1)\} & \{p(2,2), q(f(2),2)\} \end{array}$$

We extend the set of clauses with the following clauses:

$$\begin{array}{lll} \{c = 1, c = 2\} & \{f(1) = 1, f(1) = 2\} & \{f(2) = 1, f(2) = 2\} \\ \{c \neq 1, c \neq 2\} & \{f(1) \neq 1, f(1) \neq 2\} & \{f(2) \neq 1, \neg f(2) \neq 2\} \end{array}$$

and the clauses

$$\begin{array}{ll} \{\neg P(c,1), c \neq 1, P(1,1)\} & \{\neg P(c,1), c \neq 2, P(2,1)\} \\ \{\neg Q(f(1),1), f(1) \neq 1, Q(1,1)\} & \{\neg Q(f(1),1), f(1) \neq 2, Q(2,1)\} \\ \{\neg Q(f(2),1), f(2) \neq 1, Q(1,1)\} & \{\neg Q(f(2),1), f(2) \neq 2, Q(2,1)\} \end{array}$$

where the atom $x \neq y$ *is a shortcut for* $\neg(x = y)$*.*

In MACE4 steps (iii), (iv), and (v) are performed with a specific algorithm. Here, to simplify the presentation we propose to rephrase the problem of searching a finite domain for a set of ground clauses in a problem of propositional satisfiability by adding a set of clauses that manages with equality between terms that appear in the grounding of the first order clauses. A naïve implementation of a model finder algorithm based on SAT is shown in Algorithm 13.

2. The resolution rule

The resolution rule is a simple, yet powerful, inference rules that allows to infer that a formula, called *resolvent* is true whenever two formulas are true. It can be presented in the following form

$$\frac{\phi \vee \psi \qquad \neg\psi \vee \chi}{\phi \vee \chi} \tag{141}$$

Algorithm 13 Finite Model Finder

Require: a first order formula ϕ

$n \leftarrow 1$
$\phi \leftarrow$ prenex skolemized clausal form of ϕ
while true **do**
  $\phi_n \leftarrow \text{Ground}(\phi, \{1, \dots, n\})$
  for Every term t that appears in ϕ, which is not $1, \dots, n$ **do**
    $\phi_n \cup \{\{t = 1, t = 2, \dots, t = n\}\}$
    for $i = 1, \dots, n-1$ **do**
      for $j = i+1, \dots, n$ **do**
        $\phi_n \cup \{\{t \neq i, t \neq j\}$
      end for
    end for
    for $P(\dots, t, \dots)$ that appear in ϕ_n **do**
      for $i = 1, \dots, n$ **do**
        $\phi_n \leftarrow \phi_n \cup \{\neg P(\dots, t, \dots), t \neq i, P(\dots, i, \dots)\}\}$
      end for
    end for
  end for
  $\mathcal{I} \leftarrow \text{SAT}(\phi_n)$
  if $\mathcal{I} \neq$ UNSAT **then**
    return $\mathcal{I}$
  else
    $n \leftarrow n+1$
  end if
end while

Most of the time, the resolutin rule is used when the presmises are clauses, and therefore it is written as

$$\frac{\{l_1, \dots, l_k, p\} \quad \{\neg p, l_{k+1}, \dots, l_n\}}{\{l_1, \dots, l_n\}} \tag{142}$$

EXAMPLE 11.2 (Applications of resolution).

$$\frac{\{p, q, \neg r\} \quad \{\neg q, \neg r\}}{\{p, \neg r\}} \qquad \frac{\{\neg p, q, \neg r\} \quad \{r\}}{\{\neg p, q\}} \qquad \frac{\{\neg p\} \quad \{p\}}{\{\}}$$

PROPOSITION 11.1. *The rules* (141) *and* (142) *are correct. I.e, the resolvent is a logical consequence of the premises.*

PROOF. We have to show that, for every interpretation $\mathcal{I}$ and for every assignment a to the free variables of ϕ, ψ and χ we ahve that

$$\text{if } \mathcal{I} \models (\phi \vee \psi) \wedge (\neg\psi \vee \chi)[a], \text{ then } \mathcal{I} \models \phi \vee \chi[a]$$

Suppose that $\mathcal{I} \models (\phi \vee \psi) \wedge (\neg\psi \vee \chi)$, then $\mathcal{I} \models \phi \vee \psi$ and $\mathcal{I} \models \neg\psi \vee \chi$ This implies that $\mathcal{I} \models \phi \vee \psi$, and therefore that either $\mathcal{I} \models \phi$ or $\mathcal{I} \models \psi$. If $\mathcal{I} \models \phi$, then $\mathcal{I} \models \phi \vee \chi$; If $\mathcal{I} \models \psi$, then from the fact that $\mathcal{I} \models \neg\psi \vee \chi$ we have that $\mathcal{I} \models \chi$. Which implies that $\mathcal{I} \models \phi \vee \chi$. □

The resolution inference rule is a very general inference pattern which includes many inference rules that involves propositional connectives. Indeed, many common

inference rules for propositional connective reduce to one or two application of the resolutin rule when the formula sare transformed in CNF. We summerize some of them in the following table.

Rule Name	Original form	CNF form
Modus Ponens	$\dfrac{p \quad p \to q}{q}$	$\dfrac{\{p\} \quad \{\neg p, q\}}{\{q\}}$
Modus tollens	$\dfrac{\neg q \quad p \to q}{\neg p}$	$\dfrac{\{\neg q\} \quad \{\neg p, q\}}{\{\neg p\}}$
Chaining	$\dfrac{p \to q \quad q \to r}{p \to r}$	$\dfrac{\{\neg p, q\} \quad \{\neg q, r\}}{\{\neg p, r\}}$
Reductio ad absurdum	$\dfrac{p \to q \quad p \to \neg q}{\neg p}$	$\dfrac{\{\neg p, q\} \quad \{\neg p, \neg q\}}{\{\neg p\}}$
Reasoning by case	$\dfrac{p \vee q \quad p \to r \quad q \to r}{r}$	$\dfrac{\dfrac{\{p, q\} \quad \{\neg p, r\}}{\{q, r\}} \qquad \{\neg q, r\}}{\{r\}}$
Tertium non datur	$\dfrac{p \quad \neg p}{\bot}$	$\dfrac{\{p\} \quad \{\neg p\}}{\{\}}$

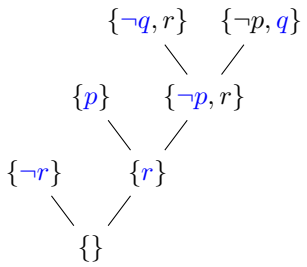
2.1. Satisfiability via resolution. Using resolution it is possible to build a decision procedure that decides if a set of clauses Γ are satisfiable. To check if a set of propositional formulas is satisfiable, you have transform them in a set Γ of clauses and then apply PROPOSITIONAL RESOCUTION algorithm. The simple

Algorithm 14 PROPOSITIONALA RESOLUTION

```
while C1 ∪ {p} ∈ Γ, C2 ∪ {¬p} ∈ Γ and C1 ∪ C2 ∉ Γ do
    Γ ← Γ ∪ {C1 ∪ C2}
    if {} ∈ Γ then
        return Unsat
    end if
end while
return Sat
```

algorithm Propositional Resolution, applies the resolution rule to all the possible pairs of clauses that contains opposite literals. Notice that the rules is applies also to the resolvent clauses obtained from the applicaiton of the resolution rule to other clauses. The algorithms terminates, since the number of clauses that can be build using the propositional variables occurring in Γ are finite, and since at every iteration a new clause is added to Γ, the while loop eventually terminates. However, differently from DPLL the propositional resolution decision procedure does not provide an assignmnet that satisfies the set of clauses Γ it they are satisfiable.

EXAMPLE 11.3. *Decide if the set of clauses* $\{\{\neg p, q\}, \{\neg q, r\}, \{p\}, \{\neg r\}\}$ *issatisfiable using* PROPOSITIONAL RESOLUTION. *The followint tree shows the applications of the resolution rules to the different clauses.*



By repeated applications of the resolution rule, we obtained the empty clause. THerefore, we can conclude that the set of initial clauses are not satisfiable. This is a consequence of Proposition 11.1. Indeed if there is an assignment that satisfies the initial set of clauses, this assignment should satisty also all the clauses that can be dreived via resolution from this initial set. Which implies that such an assignment should satisfy the empty clause. Since there is no assignment that satisfies the empty clause, we can conclude that there is also no assignment that satisfies the initial set of clauses.

Notice that resolution allows to prove (un)satisfiability of a set of clauses, Therefore, if we want to check validity of a formula ϕ we have to check the unsatisfiability of the formula $\neg\phi$. However, before applying Propositional Resolution we have to transform $\neg\phi$ is CNF

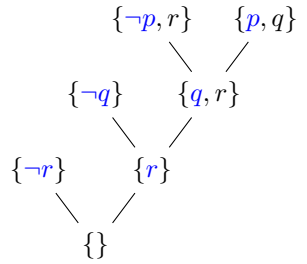
EXAMPLE 11.4. *To prove by resolution that the following formula is valide*

$$(p \vee q) \wedge (\neg p \vee r) \rightarrow q \vee r$$

we first have to transform the negation of the formula in CNF, obtaining the following set of clauses

$$\{p, q\} \qquad \{\neg p, r\} \qquad \{\neg q\} \qquad \{\neg r\}$$

By appltying resolution we can obtain the following clauses:



The fact that the negation of the CNF of the initial formula is not satisfiable allows us to infer that the negaciton of the formula is not satisfiable, and therefore that the formula itself is valid.

In a very similar method we can use propositional resolution to check if a formula ϕ is a logical consequence of a set of formulas Γ, i.e, that $\Gamma \models \phi$. To this aim we check if the CNF of $\Gamma \cup \{\neg\phi\}$ is satisfiable using propositional resolution, by exploiting the well known connection stating that $\Gamma \cup \{\neg\phi\}$ is unsat if and only if $\Gamma \models \phi$.

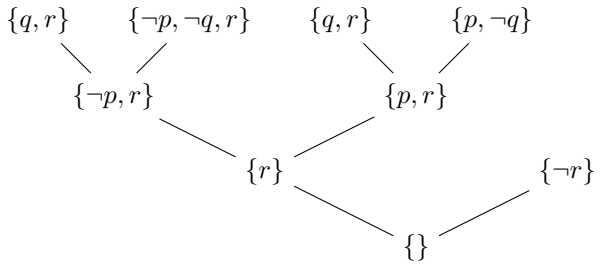
EXAMPLE 11.5. *To prove by propositional resolution that*

$$(q \to p), (q \vee r), (p \wedge q \to r) \models r$$

we have to transform the premises (the formulas on the left of the $\models$ symbol) and the negation of the conclusion (the formula on the right of the $\models$ symbol) in CNF, obtaining the following set of clauses:

$$\{p, \neg q\} \qquad \{q, r\} \qquad \{\neg p, \neg q, r\} \qquad \{\neg r\}$$

and then. by applying Propositional Resolution we can try to derive the empty clause:



The rule of propositional resolution is complete in the sense that if a set of clauses Γ are not satisfiable, then there is a finite number of applications of the resolution rule such that derives the empty clause.

THEOREM 11.1. *(Completeness of propositional resolution) If Γ is a set of unsatisfiable clauses then the algorithm Propositional Resolution terminates with "Unsat".*

PROOF. The proof is by induction on the number of propositional variable sthat aoccurs in Γ¿

Base case. If Γ contain 0 propositional variables then $\Gamma = \{\{\}\}$, and the PROPOSITIONAL RESOLUTION terminates with "unsat"

Inductive step. . Suppose that Γ contains $n+1$ variables and the theorem is true for the set of clauses that contains n variables. Let p a variable that occours in Γ Then $\Gamma|_p$ and $\Gamma|_{\neg p}$ are not satisfiable. Bt induction Propositional Resolution applied to $\Gamma|_p$ will generate the empty clause. By performing the same rule applications to the original clauses in Γ (from which we have removed $\neg p$), we obtain the clause $\{\neg p\}$. Following the same argument, since $\Gamma|_{\neg p}$ is inconsistent, we ahve that from Γ we can derive the clause $\{p\}$. Therefore, by a final application of the rule (142) we can derive the empty clasue, and therefore the algorithm will return "unsat". □

3. Unification

Theorem 11.1 states that the rule of resolution is sufficient for checking unsatisfiability (and therefore, satistiability, validity, and logical consequence) for propoistional formula. However, this rule is not sufficient to perfomr the same task for first order logic. For instance we know that $\forall x(P(x) \to Q(x)) \wedge P(a) \to Q(a)$ for some constant a is valid. To use the resolution method to prove it's validity, we transform it in clausal formu obtaining the two clauses $\{\neg P(x), Q(x)\}, \{P(a)\}, \{\neg Q(a)\}$ but we cannot apply the resolution rule to them, since $P(x)$ is different from $P(a)$. However, since every variable in a clause is implicitly univerally quantified, the $\{\neg P(x), Q(x)\}$ can be considered as a template for any clause $\{\neg P(t), Q(t)\}$ for every term t. This means that this clause also "contains" the clause $\{\neg P(a), Q(a)\}$.

Therefore, one can derive the clause $\{Q(a)\}$ and by resolution we can derive the empty clause $\{\}$.

TO FINISH

4. Deciding (un)satisfiability in FOL

TO DO

5. Exercises

Exercise 159:
Let $\theta = [x/f(y)]$,$\lambda = [y/z]$ and $\mu = [z/a]$. compute:

(1) $\theta \circ \lambda \circ \mu$
(2) $\theta \circ \mu \circ \lambda$
(3) $\lambda \circ \theta \circ \mu$
(4) $\lambda \circ \mu \circ \theta$
(5) $\mu \circ \theta \circ \lambda$
(6) $\mu \circ \lambda \circ \theta$

Exercise 160:
Find two substitutions α and β such that $\alpha \circ \beta \neq \beta \circ \alpha$.

Exercise 161:
Prove that the composition of substitutions is associative. I.e., that for every substitutions α, β, and γ:

$$(\alpha \circ \beta) \circ \gamma = \alpha \circ (\beta \circ \gamma)$$

Solution Let α, β and γ the following substitutions

$$\alpha = [x_1/t_1, \ldots, x_n/t_n]$$
$$\beta = [y_1/u_1, \ldots, y_n/u_n]$$
$$\gamma = [z_1/v_1, \ldots, z_n/n_n]$$

To prove the associativity property, we use the fact that for every substitution $\sigma = [x_1/t_1, \ldots, x_n/t_n]$, then, for every substitution θ, then

$$\sigma \circ \theta = [x_1/t_1\theta, \ldots, x_n/t_n\theta]$$

i.e., $\sigma \circ \theta$ is the substitution obtained by applying the substitution θ to all the terms t_i of the substitution σ. We therefore have that

$$\alpha \circ (\beta \circ \gamma) = [x_1/t_i(\beta \circ \gamma), \ldots, x_n/t_n\beta \circ \gamma)]$$

with

$$\beta \circ \gamma = [y_1/u_1\gamma, \ldots, y_n/y_n\gamma]$$

And therefore we have that

$$\alpha \circ (\beta \circ \gamma) = [x_1/t_i[y_1/u_1\gamma, \ldots, y_n/y_n\gamma] \ldots, x_n/t_n[y_1/u_1\gamma, \ldots, y_n/y_n\gamma]]$$

We also thave that

$$(\alpha \circ \beta) = [x_1/t_1[y_1/u_1 \ldots, y_n/u_n], \ldots x_n/t_n[y_1/u_1, \ldots, y_n/u_n]]$$

and therefore

$$\begin{aligned}(\alpha \circ \beta) \circ \gamma &= [x_1/t_1[y_1/u_1 \ldots, y_n/u_n]\gamma, \ldots x_n/t_n[y_1/u_1, \ldots, y_n/u_n]\gamma] \\ &= [x_1/t_1[y_1/u_1\gamma \ldots, y_n/u_n\gamma], \ldots x_n/t_n[y_1/u_1\gamma, \ldots, y_n/u_n\gamma]]\end{aligned}$$

□

Exercise 162:

Find the most general unifier (MGU) of the set of atoms $\{P(a,y), P(xf(b))\}$
Exercise 163:

Find a most general unifier for the set $\{P(a,x,f(g(y))).P(z,f(z),f(u))\}$ **Solution** $\theta = [z/a.x/f(a), u/g(y)]$ $\square$

Exercise 164:
Determine whether or not the set $W = \{Q(f(a),g(x)), Q(y,y)\}$ is unifiable. **Solution** W is not unifiable. $\square$

Exercise 165:
Determine whether each of the following sets of expressions are unifiable. If yes give a MGU:

(1) $\{Q(a,x,f(x)), Q(a,y,y)\}$
(2) $\{Q(x,y,z), Q(u,h(v,v),u)\}$

Exercise 166:
Transform the following formula in prenex Skolemized conjunctive normal form:

$\forall x \exists y \exists z(father(x,y) \wedge mother(x,z)) \wedge$
$\forall xyzw(father(x,z) \wedge mother(x,w) \wedge father(y,z) \wedge mother(y,w) \rightarrow sibling(x,y)$

Solution

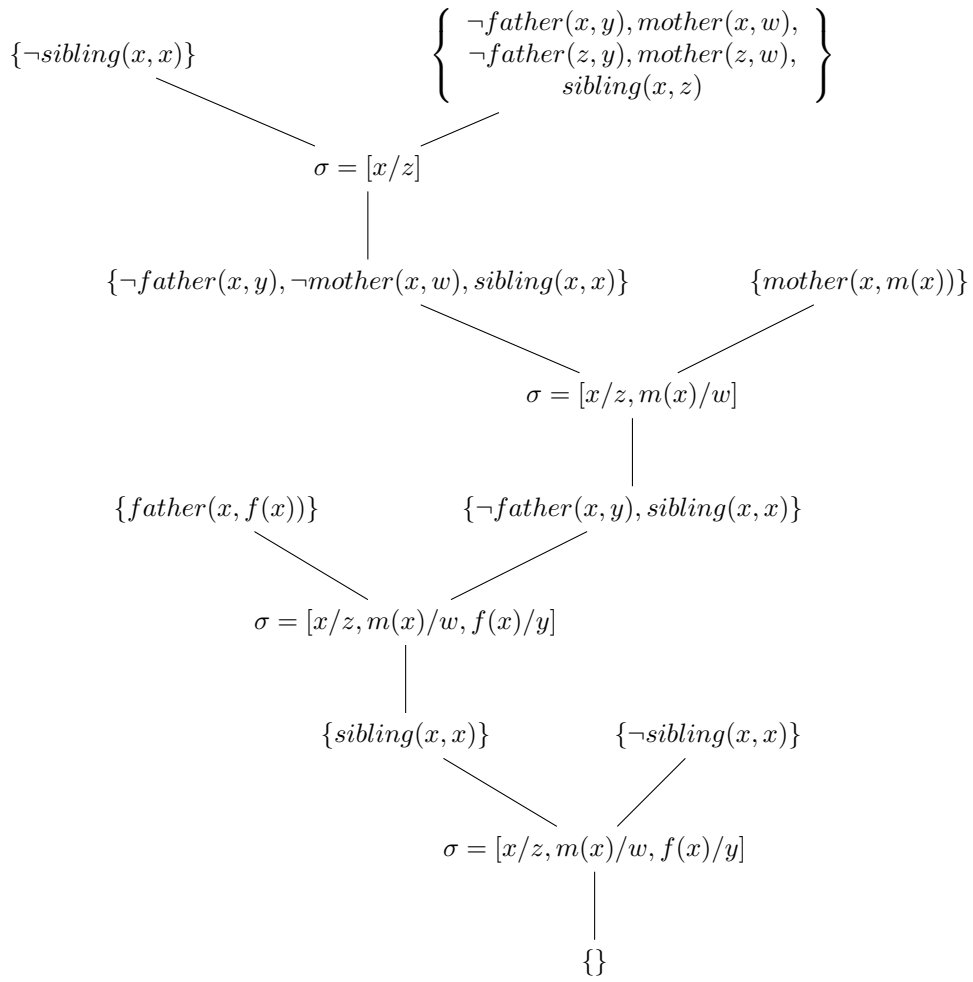
$father(x, f(x))$
$mother(x, m(x))$
$\neg father(x,z) \vee \neg mother(x,w) \vee \neg father(y,z) \vee \neg mother(y,w) \vee sibling(x,y)$

$\square$

Exercise 167:
From the clauses of the previous exercise prove that

$$sibling(x,x)$$

Solution



□ **Exercise 168:**

Find a most general unifier for the set

$$\{P(a,x,f(g(y))).P(z,f(z),f(u))\}$$

Solution

$$\sigma = [z/a, x/f(a), u/g(y)]$$

□ **Exercise 169:**

Apply the resolution and unification rule to the following clauses

$$\neg P(x,y) \vee \neg Q(x,b,y)$$
$$Q(a,z,f(z,w)) \vee m(w,b)$$

x, y, z, w are variables, and a, b are constants **Solution** The two clauses contains two opposite literals on the predicate Q that unify which are:

$$Q(x,b,y), Q(a,z,f(z,w))$$

Their most general unifier is

$$\sigma = [x/a, y/f(b,w), z/b]$$

By applying the resolution rule we obtain the clause

$$\neg P(a, f(b,w)) \vee m(w,b)$$

□

Exercise 170:

Find all resolvents (i.e., all the clauses that can be derived from the application of first order resolution) of the following two clauses:

$$\phi_1 = \{\neg P(x,y), \neg P(f(a), g(u,b)), Q(x,u)\}$$
$$\phi_2 = \{P(f(x), g(a,b)), \neg Q(f(a),b), \neg Q(a,b)$$

where x, y, and u are variables and a and b are constants. **Solution** Let us first rename the variables in order to be sure that there is no clashing. We rename the variable x of the second clause with z obtaining

$$\phi_2 = \{P(f(z), g(a,b)), \neg Q(f(a),b), \neg Q(a,b)$$

The two clauses contains four pairs of opposite literals that can bee unified. In the following table we report each pair of literal the most general unifier, and the corresponding resolvent

Lit. in ϕ_1	Lit. in ϕ_2	Unifier	resolvent
$\neg P(x,y)$	$P(f(z), g(a,b))$	$x/f(z), y/b$	$\neg P(f(a), g(u,b)), Q(f(z),u), \neg Q(f(a),b), \neg Q(a,b)$
$\neg P(f(a), g(u,b))$	$P(f(z), g(a,b))$	$z/a, u/a$	$\neg P(x,y), Q(x,a), \neg Q(f(a),b), \neg Q(a,b)$
$Q(x,y)$	$\neg Q(f(a),b)$	$x/f(a), y/b$	$\neg P(f(a),b), \neg P(f(a), g(u,b), P(z, g(a,b)), \neg Q(a,b)$
$Q(x,y)$	$\neg Q(a,b)$	$x/a, y/b$	$\neg P(a,b), \neg P(f(a), g(u,b)), P(f(z), g(a,b)), \neg Q(f(a),b)$

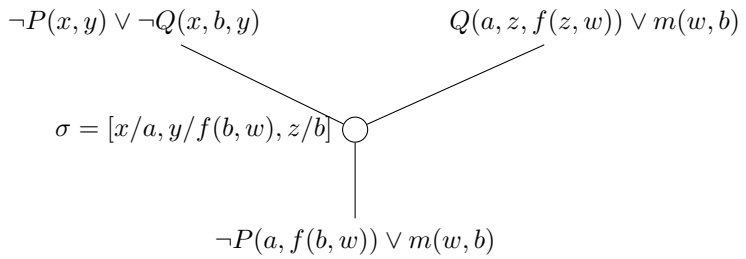
□

Exercise 171:

Apply the resolution and unification rule to the following clauses

$$\neg P(x,y) \vee \neg Q(x,b,y)$$
$$Q(a,z,f(z,w)) \vee m(w,b)$$

x, y, z, w are variables, and a, b are constants **Solution**



□

Exercise 172:

Conisder the following facts:

(1) Married people are humans;
(2) Every human has a mother;
(3) A parson is the mother in low of sombody, if she is the mother of his/her wife/husband;

Formalize them in a formula ϕ by using the predicates

- Human(x): x is a Human;
- Mother(x, y): x is the mother of y;
- MotherInLow(x, y): x is the mother in low of y;
- Married(x, y): x is merried with y.

and show by resolution that from ϕ is follows that every married person has a mother-in-low. **Solution**

(1) Married people are humans:

$$\forall xy(Married(x, y) \rightarrow Human(x) \wedge Human(y))$$

(2) Every human has a mother:

$$\forall x(Human(x) \rightarrow \exists y Mother(y, x))$$

(3) A parson is the mother in low of sombody, if she is the mother of his/her wife/husband:

$$\forall xyz.(Mother(x, y) \wedge Married(y, z) \rightarrow MotInLow(x, z))$$

We can transform in prenex CNF obtaining

$$\begin{aligned}
&\forall xy \neg Married(x, y) \vee Human(x)\\
&\forall xy \neg Married(x, y) \vee Human(y)\\
\forall x \exists y &\neg Human(x) \vee Mother(y, x)\\
\forall xyz &\neg Mother(x, y) \vee \neg Married(y, z) \vee MotInLow(x, z)
\end{aligned}$$

The third clause need to be scolemized by introducing a new function say f obtaining, the set of clauses

$$\neg Married(x, y) \vee Human(x) \tag{143}$$

$$\neg Married(x, y) \vee Human(y) \tag{144}$$

$$\neg Human(x) \vee Mother(f(x), x) \tag{145}$$

$$\neg Mother(x, y) \vee \neg Married(y, z) \vee MotInLow(x, z) \tag{146}$$

From this set of clauses we want to derive the fact that every married person has a mother-in-low, which can be translated into

$$\forall xz(Married(x, z) \rightarrow \exists y MotInLow(y, z))$$

We first need to negate and transform it in prenex CNF. obtaining

$$\exists x, z \forall y(Married(x, z) \wedge \neg MotInLow(y, z))$$

By applying skolemization we introduce two new constants a and b, and we obtain the clauses

$$Married(a, b) \tag{147}$$

$$\neg MotInLow(y, b) \tag{148}$$

We can now apply the following resolution and unification chain:

(149)	$Human(a)$	(147), (143), x/a, y/b
(150)	$Mother(f(a), a)$	(149), (145), x/a
(151)	$\neg Married(a, z) \vee MotInLow(f(a), z)$	(150), (146) x/a, $y/f(a)$
(152)	$\neg Married(a, b)$	(151),(148), z/b, $y/f(a)$
(153)	$\bot$	(152),(147)

Since we can derive the empty clause $\bot$ form the set of clauses and the negation of the conclusion, it means that the conclusion logicall follows from the initial clauses. □

Exercise 173:
Use resolution to decide if

$$\forall x Q(x) \to \forall x P(x) \leftrightarrow \exists z \forall y (Q(y) \vee P(z))$$

is valid, satisfiable, non valid, or unsatisfiable, and explain your answer.

Solution We have to negate the formula and then transform it into Skolemized negatd normal form: For the CNF transformation, We use the fact that $\neg(A \leftrightarrow B)$ is equivalent to $(A \vee B) \wedge (\neg A \vee \neg B)$.

$$\begin{gathered}\neg((\forall x Q(x) \to \forall x P(x)) \leftrightarrow (\exists z \forall y (Q(y) \vee P(z)))) \leftrightarrow \\ (\forall x Q(x) \to \forall x P(x)) \vee (\exists z \forall y (Q(y) \vee P(z))) \wedge \\ (\neg(\forall x Q(x) \to \forall x P(x)) \vee \neg(\exists z \forall y (Q(y) \vee P(z))))\end{gathered}$$

We treat the two conjuct separately. Let us start with the first one.

$$\forall x Q(x) \to \forall x P(x)) \vee (\exists z \forall y (Q(y) \vee P(z))$$

We rename variables.

$$\forall x Q(x) \to \forall w P(w)) \vee (\exists z \forall y (Q(y) \vee P(z))$$

We rewrite the $\to$ in terms of $\vee$.

$$\neg \forall x Q(x) \vee \forall w P(w)) \vee (\exists z \forall y (Q(y) \vee P(z))$$

and push the negation close to the atoms

$$(\exists x \neg Q(x) \vee \forall w P(w)) \vee (\exists z \forall y (Q(y) \vee P(z))$$

We apply Skolemization

$$(\neg Q(a) \vee \forall w P(w)) \vee (\forall y (Q(b) \vee P(z))$$

We mode the universal quantifiers into the front

$$\forall y w (\neg Q(a) \vee P(w) \vee Q(y) \vee P(b))$$

We therefore obtain the clause

$$\{\neg Q(a), P(w), Q(y), P(b)\}$$

Let us now conider the second conjunct

$$(\neg(\forall x Q(x) \to \forall x P(x)) \vee \neg(\exists z \forall y (Q(y) \vee P(z))))$$

We push the negation close to the atoms

$$(\forall x Q(x) \wedge \exists x \neg P(x)) \vee \forall z \exists y (\neg Q(y) \wedge \neg P(z))$$

We apply skolemization, introducting a new constant c (notice that you cannot use the constants used in the previous clauses) and the skolem function f.

$$(\forall x Q(x) \wedge P(c)) \vee \forall z(\neg Q(f(z)) \wedge \neg P(z))$$

We move the quantifiers in the front

$$\forall xz((Q(x) \wedge \neg P(c)) \vee (\neg Q(f(z)) \wedge \neg P(z)))$$

and transform the matrix in CNF obtaining the clauses

$$\{Q(x), \neg Q(f(z))\}.\{Q(x), \neg P(z)\}, \{\neg P(c), \neg Q(f(z))\}, \{\neg P(c), \neg P(z)\}$$

Summing up the set of clauses we have obtained are the following:

$$\{\neg Q(a), P(w), Q(y), P(b)\} \tag{154}$$

$$\{Q(x), \neg Q(f(z))\} \tag{155}$$

$$\{Q(x), \neg P(z)\} \tag{156}$$

$$\{\neg P(c), \neg Q(f(z))\} \tag{157}$$

$$\{\neg P(c), \neg P(z)\} \tag{158}$$

Notice that the set of clauses can be satisfied by the following interpretation:

$$\begin{aligned} \Delta^{\mathcal{I}} &= \{0\} \\ \mathcal{I}(a) = \mathcal{I}(b) &= \mathcal{I}(c) = 0 \\ \mathcal{I}(f) &= f(0) \mapsto 0 \\ \mathcal{I}(P) = \mathcal{I}(Q) &= \emptyset \end{aligned}$$

Since the negated of the formula is satisfiable, the formula is not valid. To check that the formula is satisfiable we have to build the an interpretation that satisfies it. To this purpopse it is convenient to transform it in CNF and then try to find an interpretation that satisfies each clause; This will show that the formula is satisfiable, Alternatively, we can try to derive the empty clause; In this case we would have proven that the formula is not satisfiable. Let us first transform the formula in CNF, We have that $A \leftrightarrow B$ is equivalent to the conjunction of $A \rightarrow B$ and $B \rightarrow A$; therefore let us transform each of the two part of the equivalence in CNF We start from the first equivalente, and we push the negation inside, and rewrite the implication

$$\begin{aligned} &(\forall x Q(x) \rightarrow \forall x P(x)) \rightarrow \exists z \forall y (Q(y) \vee P(z)) \leftrightarrow \\ &\neg(\forall x Q(x) \rightarrow \forall x P(x)) \vee \exists z \forall y (Q(y) \vee P(z)) \leftrightarrow \\ &(\forall x Q(x) \wedge \neg \forall x P(x)) \vee \exists z \forall y (Q(y) \vee P(z)) \leftrightarrow \\ &(\forall x Q(x) \wedge \exists x \neg P(x)) \vee \exists z \forall y (Q(y) \vee P(z)) \end{aligned}$$

Now let us rename the variables

$$(\forall x Q(x) \wedge \exists w \neg P(w)) \vee \exists z \forall y (Q(y) \vee P(z))$$

Distribute the and w.r.t., or

$$(\forall x Q(x) \vee \exists z \forall y (Q(y) \vee P(z))) \wedge (\exists w \neg P(w) \vee \exists z \forall y (Q(y) \vee P(z)))$$

Skolemization

$$(\forall x Q(x) \vee \forall y (Q(y) \vee P(a))) \wedge (\neg P(b) \vee \forall y (Q(y) \vee P(c)))$$

Prenex normal form

$$\forall xy((Q(x) \vee Q(y) \vee P(a)) \wedge (\neg P(b) \vee (Q(y) \vee P(c)))$$

and clausal form

$$\{Q(x), Q(y), P(a)\}$$
$$\{\neg P(b), Q(y), P(c)\}$$

Let us now consider the opposite implication

$$\exists z \forall y(Q(y) \vee P(z)) \rightarrow (\forall x Q(x) \rightarrow \forall x P(x)) \leftrightarrow$$
$$\neg(\exists z \forall y(Q(y) \vee P(z))) \vee (\neg \forall x Q(x) \vee \forall x P(x)) \leftrightarrow$$
$$\forall z \exists y(\neg Q(y) \wedge \neg P(z)) \vee (\exists x \neg Q(x) \vee \forall x P(x)) \leftrightarrow$$

Now let us rename the variables

$$\forall z \exists y(\neg Q(y) \wedge \neg P(z)) \vee (\exists x \neg Q(x) \vee \forall w P(w))$$

Skolemization and prenex normal form

$$\forall z(\neg Q(f(z) \wedge \neg P(z)) \vee (\neg Q(c) \vee \forall w P(w)) \leftrightarrow$$
$$\forall zw(\neg Q(f(z) \wedge \neg P(z)) \vee (\neg Q(c) \vee P(w)) \leftrightarrow$$
$$\forall zw(\neg Q(f(z)) \vee \neg Q(c) \vee P(w)) \wedge (\neg P(z) \vee \neg Q(c) \vee P(w))$$

and rewrite in CNF

$$\{\neg Q(f(z)), \neg Q(c), P(w)\}$$
$$\{\neg P(z), \neg Q(c), P(w)\}$$

Therefore the set of clauses are:

$$\{Q(x), Q(y), P(a)\}$$
$$\{\neg P(b), Q(y), P(c)\}$$
$$\{\neg Q(f(z)), \neg Q(c), P(w)\}$$
$$\{\neg P(z), \neg Q(c), P(w)\}$$

Notice that the interpretation that interpret $\mathcal{I}(P)$ in the entire domain, will satisfy all the four clauses. Therefore the initial formula is also satisfiable. □

Exercise 174:
Consider the following facts:

(1) John likes all kind of food.
(2) Anything anyone eats and not killed is food.
(3) Anil eats peanuts and she is still alive

Prove by resolution that John likes peanuts. (Suggestion: you have to perform the following steps: 1. Formalize the statements in first order logic, 2. trasform in CNF 3. negate the goal, 4 derive the empty clause by resolution and unification).

Solution Step-1: Conversion of Facts into FOL

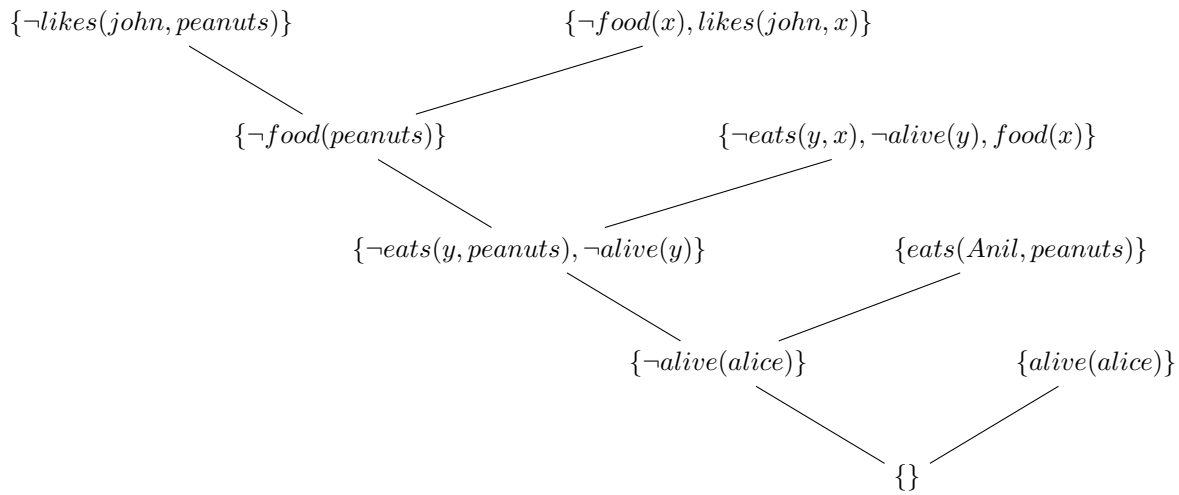
(1) $\forall x(food(x) \rightarrow likes(john, x))$
(2) $\forall x \forall y(eats(y, x) \wedge alive(y) \rightarrow food(x))$
(3) $eats(Anil, peanuts) \wedge alive(alice)$

Step-3: Conversion in CNF

(1) $\{\neg food(x), likes(john, x)\}$
(2) $\{\neg eats(y, x), \neg alive(y), food(x)\}$
(3) $\{eats(Anil, peanuts)\}$
(4) $alive(alice)$

Step-1: add negation of the goal

(1) $\{\neg food(x), likes(john, x)\}$
(2) $\{\neg eats(y, x), \neg alive(y), food(x)\}$
(3) $\{eats(Anil, peanuts)\}$
(4) $alive(alice)$
(5) $\neg likes(john, peanuts)$



Since we derived the empty clasue it means that the goal logically follows from the premises. □

Exercise 175:
Consider the following statements

A grandparent of a person is a parent of a parent of the person

Translate the above facts in FOL using the following symbols:

$P(x, y) = x$ is a parent of y, P is a binary predicate
$G(x, y) = x$ is a grandparent of y, G is a binary predicate

Then use resolution to show that if x and y have the same parents they also have the same grandparents. Only formulate the problem in clausal form without doing the resolution proof.

Solution The definition of grandparent can be obtrained by formalizing the sentence: "x is a granparent of y if there is a z that has x as parent and is the parent of z. In FOL

$$\forall x \forall y (G(x, y) \leftrightarrow \exists z P(x, z) \wedge P(z, y))$$

in clausal form after Skolemization:

$$\{\neg G(x,y), P(x,f(x,y))\}$$
$$\{\neg G(x,y), P(f(x,y),y)\}$$
$$\{\neg P(x,f(x,y)), \neg P(f(x,y),y), G(x,y)\}$$

The goal states that if two people has the same parents, then they have the same grandparents. Let us first formalize the sentence x and y has the same parents. This can be formalized by $\forall z(P(z,x) \leftrightarrow P(z,y))$ Similarly "x, y having the same grapdparents" can be formalized with the formula $\forall z(G(z,x) \leftrightarrow G(z,y))$. The entire statement is the implication between the two formulas for every x and y. i.e.,

$$\forall x \forall y(\forall z(P(z,x) \leftrightarrow P(z,y)) \rightarrow \forall z(G(z,x) \leftrightarrow G(z,y)))$$

Negate the goal and transform in CNF

$$\begin{aligned}
\neg\forall x\forall y(\forall z(P(z,x) \leftrightarrow P(z,y)) &\rightarrow \forall z(G(z,x) \leftrightarrow G(z,y))) \\
\exists x\exists y(\forall z(P(z,x) \leftrightarrow P(z,y)) &\wedge \exists z\neg(G(z,x) \leftrightarrow G(z,y))) \\
\forall z(P(z,a) \leftrightarrow P(z,b)) &\wedge \neg(G(c,a) \leftrightarrow G(c,b))) \\
\{\neg P(z,a), P(z,b)\}, \{\neg P(z,b), P(z,a)\}, &\{\neg G(c,a), \neg G(c,b)\}, \{G(c,a), G(c,b)\},
\end{aligned}$$

□

Exercise 176:
Prove by resolution that the following sets of clauses are not satisfiable.

$$\{\neg P(x), Q(x), R(x,f(x))\}, \quad \{\neg P(x), Q(x), S(f(x))\} \quad \{T(a)\}$$
$$\{P(a)\} \quad \{\neg R(a,z), T(z)\} \quad \{\neg T(x), \neg Q(x)\}$$
$$\{\neg T(y), \neg S(y)\}$$

Solution

(159)	$P(a)$	By assumption
(160)	$\{\neg P(x), Q(x), R(x, f(x))\}$	By assumption
(161)	$\{Q(a), R(a, f(a))\}$	from (159) and (160) with x/a
(162)	$\{\neg P(x), Q(x), S(f(x))\}$	By assumption
(163)	$\{Q(a), S(f(a))\}$	from (159) and (162) with x/a
(164)	$T(a)$	
(165)	$\neg T(x), \neg Q(x)$	By assumption
(166)	$\neg Q(a)$	from (164) and (165) with x/a
(167)	$\{R(a, f(a))\}$	from (166) and (161)
(168)	$\{\neg R(a, z), T(z)\}$	By assumption
(169)	$\{T(f(a))\}$	from (168) and (167) with $z/f(a)$
(170)	$\neg T(y), \neg S(y)$	By assumption
(171)	$\neg S(f(a))$	from (169) and (170) with x/a
(172)	$Q(a)$	from (171) and (163)
(173)	$\bot$	from (172) and (166)

□

CHAPTER 12

First Order Model Counting

In the same manner in which first-order logic is a generalization of propositional logic, one can ask him/herself how propositional model counting generalizes to first-order model counting. This amounts to making sense of the questions: how many Σ structures satisfy a first-order formula? Without further specifications, the answer is: a formula either has 0 models, if it is not satisfiable, or it has an infinite number of models if it is satisfiable. Indeed if a formula has one model $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$ then it has an infinite set of models with domains isomorphic to $\Delta^{\mathcal{I}}$. If we change our question to: is it possible to count the models of a first-order formula on a given domain? This makes more sense. However, if a formula is satisfiable by an infinite domain there are still infinite models. Indeed if $\Delta^{\mathcal{I}}$ is infinite and π is a one-to-one function from $\Delta^{\mathcal{I}}$ to $\Delta^{\mathcal{I}}$ we can define an interpretation $\mathcal{I}^{\pi}$ that satisfies the same formulas of $\mathcal{I}$ (proof by exercise). If $\Delta^{\mathcal{I}}$ is infinite then when $\mathcal{I} \models \phi$ there might be infinite $\mathcal{I}^{\pi}$ that satisfy ϕ. For this reason, we concentrate on first-order model counting on a finite domain that contains n elements. Without loss of generality, we concentrate on the domain of the first n integers $\{1, \ldots, n\}$ also denoted by $[n]$.

DEFINITION 12.1 (First order model counting). *The problem of* first order model counting *is the problem of computing the number of Σ-structures that satisfy a first-order sentence ϕ on a given finite domain of $n \geq 2$ elements. The problem is denoted as*

$$\text{FOMC}(\phi, n)$$

REMARK 5. *Notice that in first-order model counting, we are interested in the case in which the domain contains at least 2 elements. This is because if $n = 1$ the problem reduces to propositional model counting. Indeed, if $n = 1$ we have that the formula $\forall x \phi(x) \leftrightarrow \exists x \phi(x)$ and $Q_1 x_1, \ldots, Q_n x_n \phi(x_1, \ldots, x_n)$ is equivalent to $\forall x \phi(x, \ldots, x)$. This amounts in propositional model counting $\phi(a, \ldots, a)$ for some constant a.* $\text{FOMC}(\phi, [1]) = \#\text{SAT}(\mathit{Ground}(\phi, \{a\}))$.

1. Formalizing Counting Problems in fomc

First-order model counting provides a general methodology for solving a problem of counting a set of items w.r.t. some integer parameter n. Such a methodology is based on three main steps.

(1) Define a FOL signature Σ such that the items to be counted are mapped one-to-one is a set S of Σ-structures on a domain of n elements;
(2) Provide a complete axiomatization of S in terms of a finite set of first order formula $\phi_1, \ldots, \phi_k$. This means $\mathcal{I} \models \phi_1 \wedge \cdots \wedge \phi_k$ if and only if $\mathcal{I}$ corresponds to an element of S.
(3) Compute $\text{FOMC}(\phi_1 \wedge \cdots \wedge \phi_k, n)$.

In the following, we provide a few examples of the formalization of counting problems in FOMC.

EXAMPLE 12.1. *The number of undirected graphs with n nodes can be obtained by* $\text{FOMC}(UG, n)$

$$UG \triangleq \forall x \forall y (\neg R(x,x) \wedge (R(x,y) \leftrightarrow R(y,x)))$$

EXAMPLE 12.2. *The number of undirected graphs with n nodes with at most k arks can be obtained by* $\text{FOMC}(UG \wedge |R| \leq k, n)$ *were*

$$|R| \leq k \triangleq \forall x_1 \ldots \forall x_{k+1} \forall y_1 \ldots \forall y_{k+1} \left(\bigwedge_{i=1}^{k+1} R(x_i, y_i) \rightarrow \bigvee_{i<j=1}^{k+1} (x_i = x_j \wedge y_i = y_j) \right)$$

EXAMPLE 12.3. *The number of 3-coloured undirected graphs with n nodes is equal to* $\text{FOMC}(UG \wedge 3C, n)$

$$3C \triangleq \forall x \forall y ((C_1(c) \veebar C_2(x) \veebar C_3(x)) \wedge R(x,y) \rightarrow \bigwedge_{i=1}^{3} (\neg C_i(x) \wedge C_i(y)))$$

In the formula, we use the connective $\veebar$ for exclusive or, which is definable in terms of the other connectives. Namely, $a \veebar b \triangleq (a \vee b) \wedge \neg(a \wedge b)$.

EXAMPLE 12.4. *the number of graphs with n vertexes, and such that every pair of nodes are connected with a path with length $\leq k$. To encode this the problem we have to extend the signature with k new binary symbols $R_1 \ldots R_k$. Intuitively $R_i(x,y)$, x is connected with y with a path of length i. We can solve this counting problem my computing:* $\text{FOMC}(UG \wedge R_{\leq k}, n)$

$$\begin{aligned} R_{\leq k} \triangleq\ & \forall x \forall y\, R_{\leq k}(x,y) \\ & \wedge \forall x \forall y\, (R_{\leq 1}(x,y) \leftrightarrow R(x,y)) \\ & \wedge \forall x \forall y (\bigwedge_{i=2}^{k-1} (R_{\leq i}(x,y) \leftrightarrow (R_{\leq i-1}(x,y) \vee \exists z (R_{\leq i-1}(x,z) \wedge R(z,y))) \end{aligned}$$

EXAMPLE 12.5. *Compute the number of configurations of a group of n people composed of PhD students and professors knowing that every student has a supervisor that is a professor, every professor supervises at least one student. To formalize the problem, we introduce two unary predicates $\mathsf{Prof}/1$ and $\mathsf{Stud}/1$ that represents the professors and the students respectively, and a binary predicate $\mathsf{Super}/2$ that represent the relations between a student and his/her supervisor. To compute the number of the configuration described above we can compute* $\text{FOMC}(SP, n)$ *where SP is the following set of formulas:*

$$SP = \left\{ \begin{array}{l} \mathsf{Prof}(x) \veebar \mathsf{Stud}(x) \\ \mathsf{Super}(x,y) \rightarrow \mathsf{Stud}(x) \wedge \mathsf{Prof}(y) \\ \mathsf{Stud}(x) \rightarrow \exists y (\mathsf{Prof}(y) \wedge Super(x,y)) \end{array} \right\}$$

2. Solving FOMC for specific FOL formulas

Before considering a systematic and general enough method to solve $\text{FOMC}(\phi, n)$ for every formula ϕ in (a subclass of) first-order language. let us see some examples on the solution of $\text{FOMC}(\phi, n)$ for specific formulas ϕ.

EXAMPLE 12.6. *To compute* $\text{FOMC}(\phi, n)$ *when* ϕ *is*

$$\exists x \exists y (A(x) \wedge R(x,y) \wedge B(y))$$

We can reason as follows: if A *is interpreted in a subset of* a *elements and* B *is interpreted in a subset of* b *elements. then* R *cannot be interpreted in any subset of* $[n] \setminus \mathcal{I}(A) \times [n] \cup [n] \times [n] \setminus \mathcal{I}(B)$*. Since there fre* $2^{na} + 2^{nb} - 2^{ab}$ *such a subsets, we have that*

$$\text{FOMC}(\phi, n) = 2^{n^2+2n} - \sum_a \sum_b \binom{n}{a}\binom{n}{b} 2^{an+bn-ab}$$

EXAMPLE 12.7. *To count the models of* $\text{FOMC}(\forall x R(a,x), n)$ *we can proceed as follows. We have* n *possibilities to interpret the constant* a *and for all the other* $n-1$ *can be connected via* R *with any subset of* $[n]$ *which means that there are* $2n(n-1)$ *possible choices. We, therefore, have that*

$$\text{FOMC}(\forall x R(a,x)) = n 2^{n(n-1)} \tag{174}$$

EXAMPLE 12.8. *Formulas which are largely used in knowledge and ontology engineering have one of the following two forms:*

$$\phi_1 \triangleq \forall x (A(x) \rightarrow \forall y (R(x,y) \rightarrow B(y)))$$
$$\phi_2 \triangleq \forall x (A(x) \rightarrow \exists y (R(x,y) \wedge B(y)))$$

ϕ_1 *can be rewritten in* $\forall x \forall y (A(x) \wedge \neg B(y) \rightarrow \neg R(x,y))$*. If* A *is interpreted in* a *elements and* B *in* b *elements then* $\neg R$ *should contain* $\mathcal{I}(A) \times \mathcal{I}(\neg B)$ *plus some subset of* $\mathcal{I}(\neg A) \times \mathcal{I}(\neg B) \cup \mathcal{I}(\neg A) \times \mathcal{I}(B) \cup \mathcal{I}(A) \times \mathcal{I}(B)$*. Therefore there are* $2^{n^2-an+ab}$ *possible interpretation of* $\not{R}$ *and therefore also prossible interpretations of* R*. Therefore the total number of interpretations that satisfies* ϕ_2 *is*

$$\text{FOMC}(\phi_1, n) = \sum_a \sum_b \binom{n}{a}\binom{n}{b} 2^{n^2-an+ab}$$

About ϕ_2*, if* $\mathcal{I}(A)$ *contains* a *elements and* $\mathcal{I}(B)$ b *elements then, for every element in* $\mathcal{I}(A)$ *we have to select a non empty subset of* $\mathcal{I}(B)$ *and any subset of* $\mathcal{I}(\neg B)$*. Therefore for every element of* $\mathcal{I}(A)$ *we have* $(2^b - 1)2^{n-b}$ *possibilities. For the elements not in* $\mathcal{I}(A)$ *we can select any subset of the* n *element having* $2^{n(n-a)} = 2^{n^2-an}$ *possibilities.*

$$\text{FOMC}(\phi_2, n) = \sum_a \sum_b \binom{n}{a}\binom{n}{b} (2^b - 1)^a 2^{n^2-ab}$$

EXAMPLE 12.9. *Counting the number of transitive relation on a set of* n *elements has been the object of study in discrete mathematics. In first-order model counting terms this means finding a formula for* $\text{FOMC}(\mathit{Trans}(R), n)$ *where*

$$\mathit{Trans}(R) \triangleq \forall x \forall y \forall z (R(x,y) \wedge R(y,z) \rightarrow R(x,z))$$

Mala 2022 proves that any formula for the number of transitive relations on a set cannot be a polynomial. At the same time it provides some interesting recursive lower and upper bounds for $\text{FOMC}(\mathit{Trans}(R), n)$*. To have an idea of how*

FOMC($Trans(R), n$) *behaves w.r.t.* n *we report here the sequence reported by the On-Line Encyclopedia of Integer Sequences (OEIS) OEIS Foundation Inc. n.d.*

n	FOMC($Trans(R), n$)	n	FOMC($Trans(R), n$)
1	2	10	7307450299510288
2	13	11	3053521546333103057
3	171	12	1797003559223770324237
4	3994	13	1476062693867019126073312
5	154303	14	167923955814957022915680299 7
6	9415189	15	26282251741438573066236955766 71
7	878222530	16	56261758675137790587070060165929 54
8	122207703623	17	16388270713364863943791979866838296851
9	24890747921947	18	64662720846908542794678859718227127212465

3. FOMC via grounding

A first naïve idea to develop a general procedure to compute FOMC(ϕ, n) can be obtained by grounding the formula with n constants, which results in a propositional formula, and then apply propositional model counting. Since we are dealing with finite domains we can reduce first order formulas to equivalent propositional formulas by grounding quantifiers:

DEFINITION 12.2. *For every First Order sentence (= formula with no free variables)* ϕ *on a signature* Σ*, and set of constants* C*,* $Ground(\phi, C)$ *is recursively defined as follows:*

(1) $Ground(\phi, C) = \phi$ *if* ϕ *does not contain quantifiers;*
(2) $Ground(\forall x.\phi(x), C) = \bigwedge_{c \in C} Ground(\phi(c), C)$
(3) $Ground(\exists x.\phi(x), C) = \bigvee_{c \in C} Ground(\phi(c), C)$
(4) $Ground(\phi \circ \psi, C) = Ground(\phi, C) \circ Ground(\phi, C)$ *for every connective* $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$*;*
(5) $Ground(\neg\phi, C) = \neg Ground(\phi, C)$

In other words the operation of grounding a firt order formula w.r.t, a set of constants C replaces the universal quantifier $\forall x$ with a big conjunction where the variable x is replaced with each constant $c \in C$ and each existential quantifier $\exists x$ is replaced by a big disjunciton where c is replaced with each of the constant in C.

EXAMPLE 12.10. $Ground(\forall x(A(x) \rightarrow \exists y(R(x,y) \wedge B(y))), \{a, b\})$

$$A(a) \rightarrow (R(a,a) \wedge B(a)) \vee (R(a,b) \wedge B(b)) \wedge$$
$$A(b) \rightarrow (R(b,a) \wedge B(a)) \vee (R(b,b) \wedge B(b))$$

EXAMPLE 12.11. $Ground(\forall x, y.(R(x,y) \rightarrow R(y,x)), C) =$

$$\bigwedge_{c \in C} \bigwedge_{c' \in C} R(c,c') \rightarrow R(c',c)$$

Let us now shw the circumstances under which FOMC(ϕ, n) can be translated in propositional model counting.

PROPOSITION 12.1. *If* ϕ *is a first order sentence on a signature* Σ *containing only predicate symbols (i.e., no constant and function symbols), then*

$$\text{FOMC}(\phi, n) = \#\text{SAT}(Ground(\phi, \{c_1, \ldots, c_n\}), \mathcal{HB}_{\Sigma \cup \{c_1, \ldots, c_n\}})$$

where for every signature Σ, $\mathcal{HB}_\Sigma$ denotes the Herbrand's base (i.e. the set of ground atomc) that can be built from Σ.

PROOF OUTLINE. For every model $\mathcal{I}$ of ϕ on the domian of $\{1,\dots,n\}$ We define the following bijection:

$$\mathcal{I}_{FOL} \models p(x_1,\dots,x_n)[a_{x_1\leftarrow d_1,\dots,x_n\leftarrow d_n}] \text{ iff } \mathcal{I}_{PROP}(p(c_{d_1},\dots,c_{d_n})) = 1$$

One can easily show that this mapping is an isomorphism between the set fo FOL interrpetatins on $\{1,\dots,n\}$ and the propositional assignment $\mathcal{I}_{PROP}$ and thast $\mathcal{I}_{FOL} \models \phi$ if and only if $\mathcal{I}_{PROP} \models ground(\phi, \{c_1 \dots, c_n\})$ □

Notice that Proposition 12.1 requires that the formula does not contain neither constants nor functional symbols. If these symbols are there one have to provide also an interpretations of constants and function symbols. The rewriting is still possible but a bit more convoluted. Consider for instance the face that ϕ is the formula $\forall x R(a,x)$ for some constant a. The grounding is $R(a,c_1) \wedge \dots \wedge R(a,c_n)$. Notice that $\mathcal{HB}_{\{R,a,c_1,\dots,c_n\}}$ contains $(n+1)^2$ distinct propositional atoms, and only n of them occour in the grounding. This implies that $\#\text{SAT}(R(a,c_1) \wedge \dots \wedge R(a,c_n), \mathcal{HB}_{R,a,c_1,\dots,c_n}) = mc(R(a,c_1) \wedge \dots \wedge R(a,c_n)) \cdot 2^{((n+1)^2)-n} = 2^{n^2+n+1}$. The difference is due to the fact that on the domain of n element a is interpreted in one of the elements of the domain, and therefore $a = c_i$ is true for at least one c_i, Furthermore, when a is interpreted in the same element than c_i then the propositional variable $R(a,c_j)$ is equivalent to $R(c_i,c_j)$ therefore the two propositon cannot be interpreted independently.

The positive aspect of the metod of grounding in that it is a general method for FOMC which works for every first order sentence (without constants and function symbols). However, it has one major drawback, which is the fact that the grounding operation has the undesirable effect of exponentially exploding the formula. For instance the grounding of the formula $Q_1 x_1, \dots, Q_n x_k P(x_1,\dots,x_k)$ on a domain of n elements generates a conjunction of n^k formulas $\psi(c_1,\dots,c_k)$ where $n = |C|$. This conjunction will contain a polinomially large (in n) number of propositional variables and we know that model counting algorithm take exponential time in the number of propositional variables. This means that the complexity of this method will grow exponentially with the number of domain elements.

4. Liftability in FOMC

The notion of *liftability* has been introduced in Statistical Relational Learning models Poole 2003 as the capability of carry out probabilistic inference without grounding a probabilistic model to every single instance in the domain, assuming that objects are undistinguished. Since, one of the most important motivation for developing first order model counting is to develop liftable methods for probabilistic inference, the notion of liftability is very central in FOMC.

DEFINITION 12.3 (Liftable class of formulas). *A class $\mathcal{C}$ of first order formulas are* liftable *(for FOMC) if for every sentence $\phi \in \mathcal{C}$ there is an algorithm to compute* $\text{FOMC}(\phi, n)$ *that runs in time polynomial in n.*

The work Jaeger and Van den Broeck 2012 the authors provides a set of positive and negative results on liftability of certain classes of first order logic formulas. Here we concentrate with one of the most well known classes of first order logic formulas

for which FOMC has been shown to be liftable. This is the C^2 class. In the rest of the capter we concentrate on this class.

5. The Two-Variable Fragments: $\mathcal{L}^2$

FO^2 is the class of first order logical formulas that contains only two variables. Conventionally, these two variables are x and y. chapter we will mainlybe dealing with this fragment and its various extensions. From now on,

DEFINITION 12.4. *For every $k \geq 1$ the language $\mathcal{L}^k$ contains all the first order formulas that can be build using only k individual variables.*

EXAMPLE 12.12. *The following are formulas of FO^2;*

- $\forall x \exists y (R(x,y) \wedge A(x) \wedge B(y) \wedge \neg x = y)$
- $\exists x (A(x) \wedge \forall y (R(x,y) \rightarrow \exists x R(y,x) \wedge B(x))$

EXAMPLE 12.13. $\forall x, y, z . R(x,y) \wedge R(y,z) \rightarrow R(x,z)$ *is a formula in $\mathcal{L}^3$, that formalizes the fact that R is a transitive relation. Such a condition cannot be expressed in $\mathcal{L}^2$.*

From now on we will concentrate on $\mathcal{L}^2$.

5.1. Types and tables. In the following, we introduce the notion of 1-type. A 1-type describe one of the possible configuration of any element of a domain. A 1-type is a combination of all the unary properties an individual can have, e.g., "being red", "being italian", "not being male", ...

DEFINITION 12.5 (1-type). *Given a FOL signature Σ a 1-type is a conjunction of maximally consistent set of literals containing exactly one variable and no constants.*

EXAMPLE 12.14 (1-type). *Let $\Sigma = \{A/1, R/2, S/3\}$ (the notation X/n means that X is a predicate with arity equal to n) The set of 1-types of Σ are:*

$$A(x) \wedge R(x,x) \wedge S(x,x,x) \qquad A(x) \wedge R(x,x) \wedge \neg S(x,x,x)$$
$$A(x) \wedge \neg R(x,x) \wedge S(x,x,x) \qquad A(x) \wedge \neg R(x,x) \wedge \neg S(x,x,x)$$
$$\neg A(x) \wedge R(x,x) \wedge S(x,x,x) \qquad \neg A(x) \wedge R(x,x) \wedge \neg S(x,x,x)$$
$$\neg A(x) \wedge \neg R(x,x) \wedge S(x,x,x) \qquad \neg A(x) \wedge \neg R(x,x) \wedge \neg S(x,x,x)$$

Notice that in a 1-type we have also atoms with binary, ternary, and more in general n-ary predicates. The key point is that these predicates are applied only to a (n-tuple) of a single variable.

PROPOSITION 12.2. *If Σ contains n predicates there are 2^n 1-types.*

We use natural number $1(x)$, $2(x)$, ..., $u(x)$ to denote the 1-types. u is used to denot the last 1-type and the total number of 1-types. The notation $i(y)$, where $i(x)$ is 1-type and y a variable is the result of replacing x with y in $i(x)$. We use a similar notation for constants c where $i(c)$ denotes the replacement of x with c in the 1-type $i(x)$.

Analogously to 1-types, which describe the values of all the boolean properties of an individual, we want to have a similar notion that describes the type of relationship between two individuals. e.g "x is the boss of y", "x is older than y" "x is a friend of y", "x and y share the same office", ... Notice that x and y must

stay for two distinct individuals since the case in which x and y denotes the same individual is already part of the 1-type of x (e.g., "x is the boss of x"). THis is the notion of 2-table

DEFINITION 12.6 (2-table). *A* 2-table *of a FOL signature Σ is any conjunction of a maximally consistent set of literals containing exactly two distinct variables x, y and the literal $x \neq y$.*

EXAMPLE 12.15 (2-table). *Let* $\Sigma = \{R/2, S/2\}$

$$R(x,y) \wedge R(y,x) \wedge S(x,y) \wedge S(x,y) \wedge x \neq y$$
$$R(x,y) \wedge R(y,x) \wedge S(x,y) \wedge \neg S(y,x) \wedge x \neq y$$
$$R(x,y) \wedge R(y,x) \wedge \neg S(x,y) \wedge S(y,x) \wedge x \neq y$$
$$R(x,y) \wedge R(y,x) \wedge \neg S(x,y) \wedge \neg S(y,x) \wedge x \neq y$$
$$R(x,y) \wedge \neg R(y,x) \wedge S(x,y) \wedge S(y,x) \wedge x \neq y$$
$$R(x,y) \wedge \neg R(y,x) \wedge S(x,y) \wedge \neg S(y,x) \wedge x \neq y$$
$$R(x,y) \wedge \neg R(y,x) \wedge \neg S(x,y) \wedge S(y,x) \wedge x \neq y$$
$$R(x,y) \wedge \neg R(y,x) \wedge \neg S(x,y) \wedge \neg S(y,x) \wedge x \neq y$$
$$\neg R(x,y) \wedge R(y,x) \wedge S(x,y) \wedge S(y,x) \wedge x \neq y$$
$$\neg R(x,y) \wedge R(y,x) \wedge S(x,y) \wedge \neg S(y,x) \wedge x \neq y$$
$$\neg R(x,y) \wedge R(y,x) \wedge \neg S(x,y) \wedge S(x,y) \wedge x \neq y$$
$$\neg R(x,y) \wedge R(y,x) \wedge \neg S(x,y) \wedge \neg S(y,x) \wedge x \neq y$$
$$\neg R(x,y) \wedge \neg R(y,x) \wedge S(x,y) \wedge S(y,x) \wedge x \neq y$$
$$\neg R(x,y) \wedge \neg R(y,x) \wedge \neg S(x,y) \wedge S(y,x) \wedge x \neq y$$
$$\neg R(x,y) \wedge \neg R(y,x) \wedge \neg S(x,y) \wedge \neg S(y,x) \wedge x \neq y$$
$$\neg R(x,y) \wedge \neg \wedge S(x,y) \wedge \neg S(y,x) R(y,x) \wedge x \neq y$$

A special case of 2-tables happens when the signature contains only unary predicates. In this case there is only a single 2-table which is $x \neq y$.

Similarly to what we have done for 1-types, we use the notation $1(x,y), 2(x,y), \ldots b(x,y)$ to denote 2-table of a FOL signature Σ, and b denotes the number of the 2-tables.

PROPOSITION 12.3. *if Σ contains n_i predicates with arity equal to i, then there are* $2^{\sum_i n_i (2^i - 2)}$

We assume an arbitrary order on 1-types and 2-table. Finally we define 2-type that is a full description of the properties of two distinct domain elements and their relations.

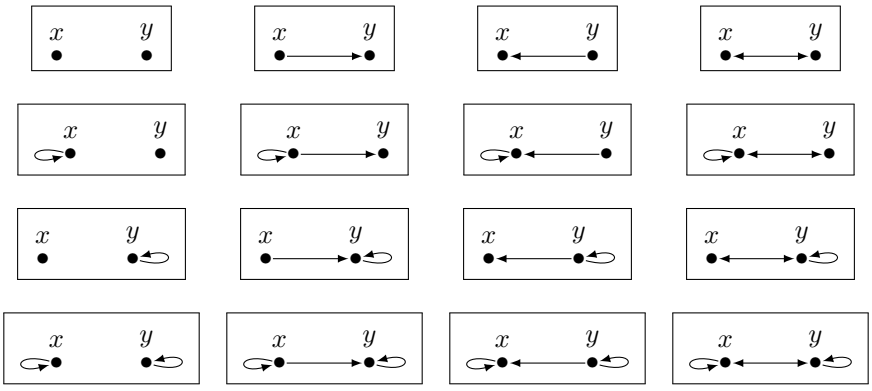
DEFINITION 12.7 (2-type). *Given a FOL signature Σ a* 2-type *is the conjunction of a maximally consistent set of literals containing at most two distinct variables x, y and no constants and the literal $x \neq y$.*

Notice that a 2-type is the conjunction of two one types one for x and another for y and a 2-table. Therefore we denote 2-types with three numbers $ijl(x,y)$ where i and j are the 1-types of x and y respectively and l is the 2-table of x and y. Formally we have that $ijl(x,y)$ is equal to $i(x) \wedge j(y) \wedge l(x,y)$.

EXAMPLE 12.16. *The set of 2-types of the FOL signature* $\Sigma = \{R/2\}$ *are*

$$R(x,x) \wedge R(y,y) \wedge R(x,y) \wedge R(y,x) \wedge x \neq y$$
$$R(x,x) \wedge R(y,y) \wedge R(x,y) \wedge \neg R(y,x) \wedge x \neq y$$
$$R(x,x) \wedge R(y,y) \wedge \neg R(x,y) \wedge R(y,x) \wedge x \neq y$$
$$R(x,x) \wedge R(y,y) \wedge \neg R(x,y) \wedge \neg R(y,x) \wedge x \neq y$$
$$R(x,x) \wedge \neg R(y,y) \wedge R(x,y) \wedge R(y,x) \wedge x \neq y$$
$$R(x,x) \wedge \neg R(y,y) \wedge R(x,y) \wedge \neg R(y,x) \wedge x \neq y$$
$$R(x,x) \wedge \neg R(y,y) \wedge \neg R(x,y) \wedge R(y,x) \wedge x \neq y$$
$$R(x,x) \wedge \neg R(y,y) \wedge \neg R(x,y) \wedge \neg R(y,x) \wedge x \neq y$$
$$\neg R(x,x) \wedge R(y,y) \wedge R(x,y) \wedge R(y,x) \wedge x \neq y$$
$$\neg R(x,x) \wedge R(y,y) \wedge R(x,y) \wedge \neg R(y,x) \wedge x \neq y$$
$$\neg R(x,x) \wedge R(y,y) \wedge \neg R(x,y) \wedge R(y,x) \wedge x \neq y$$
$$\neg R(x,x) \wedge R(y,y) \wedge \neg R(x,y) \wedge \neg R(y,x) \wedge x \neq y$$
$$\neg R(x,x) \wedge \neg R(y,y) \wedge R(x,y) \wedge R(y,x) \wedge x \neq y$$
$$\neg R(x,x) \wedge \neg R(y,y) \wedge R(x,y) \wedge \neg R(y,x) \wedge x \neq y$$
$$\neg R(x,x) \wedge \neg R(y,y) \wedge \neg R(x,y) \wedge R(y,x) \wedge x \neq y$$
$$\neg R(x,x) \wedge \neg R(y,y) \wedge \neg R(x,y) \wedge \neg R(y,x) \wedge x \neq y$$

The above 2-types can be visualised in the following 16 graph templates:



DEFINITION 12.8. *For every* Σ*-structure* $\mathcal{I}$

(1) a constant c realizes *a 1-type* i *if* $\mathcal{I} \models i(c)$*;*
(2) every set of two constants $\{c,d\}$ realizes *a 2-type* $ijl(x,y)$*, with* $i < j$ *if either* $\mathcal{I} \models ijl(c,d)$ *or* $\mathcal{I} \models ijl(d,c)$*;*
(3) every set of two constants $\{c,d\}$ realizes *a 2-type* $iil(x,y)$*, if* $c < d$ *implies that* $\mathcal{I} \models iil(c,d)$*.*

1-types and 2-types are exclusive in the sense that a domain element realizes one and only one 1-type; and a pair of domain elements realizes one and only one 2-type. This is formally stated by the following proposition:

PROPOSITION 12.4. *For every interpretation* $\mathcal{I}$*:*

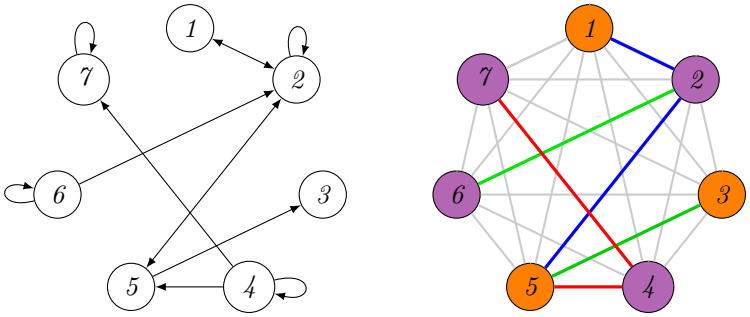


FIGURE 1. The left graph shows an interpretation on $\Sigma = \{R/2\}$, the graph on the right highlight with the corresponding colors q'the 1-type of every node and the 2-tables for every pair of nodes.

(1) *Every domain element realizes a single 1-type i;*
(2) *Every unordered pair of domain elements (i.e, any set of two domain elements) realizes a single 2-type ijl with $i \leq j$.*

PROOF. Let's start by proving that every domain element realizes a one and only une 1-type. For every n-ary predicate P and every constant we have that either $\mathcal{I} \models P(c, \ldots, c)$ or $\mathcal{I} \models \neg P(c, \ldots, c)$, but not both. Therefore $\mathcal{I} \models i(c)$ only for the 1-type

$$\bigwedge_{\mathcal{I} \models P(c,\ldots,c)} P(x, \ldots, x) \wedge \bigwedge_{\mathcal{I} \not\models P(c,\ldots,c)} \neg P(x, \ldots, x).$$

Similar reasoning can be done for 2-tables and 2-types. □

EXAMPLE 12.17. *Let $\Sigma = \{R/2\}$ then we have the following 1-types*

$$1(x) \triangleq R(x, x),$$
$$2(x) \triangleq \neg R(x, x),$$

and the following 2-tables

$$1(x, y) \triangleq R(x, y) \wedge R(y, x) \wedge x \neq y$$
$$2(x, y) \triangleq R(x, y) \wedge \neg R(y, x) \wedge x \neq y$$
$$3(x, y) \triangleq \neg R(x, y) \wedge R(y, x) \wedge x \neq y$$
$$4(x, y) \triangleq \neg R(x, y) \wedge \neg R(y, x) \wedge x \neq y$$

Suppose that we have an interpretation $\mathcal{I}$ as shown in the left part of Figure 1. This interpretation can be equivalently represented by associating to every element of the domin $\{1, \ldots, 7\}$ one specific 1-type and to every pair of elements a 2-table, as shown in the right graph of Figure 1.

In associating the 2-table one have to pay attention to the order of nodes, indeed if (c, d) realizes the 2-table $l(x, y)$ it is possible that d, c realizes a different 2-table. For instance we have that in the above example $\mathcal{I} \models 2(c, d)$ if and only if $\mathcal{I} \models 3(d, c)$. In order to maintain the fact that one pair of nodes realizes a single 2-table, we consider the following order:

- if $\mathcal{I} \models i(c) \wedge j(d)$ we consider the order (c,d) if $i < j$ and (d,c) if $j > i$;
- if $\mathcal{I} \models i(c) \wedge i(d)$ we consider (c,d) if $c < d$ otherwise we consider (d,c).

With this ordering every $\{c,d\}$ contained in the domain is associated with a single 2-type $ijl(x,y)$ with $i \leq j$.

5.2. Cardinality vectors. The *1-type cardinality vector* of a Σ-structure $\mathcal{I}$ is a vector $\boldsymbol{k} = (k_1, \ldots, k_u)$ where u is the number of 1-types of Σ. where k_i is the number of elements of the domain of $\mathcal{I}$ that realize the i-th 1-type. Since every element of the domain realizes one and only one 1-type $\sum_i k_i = n$ the size of the domain of $\mathcal{I}$.

A *2-table cardinality vector* of an interpretation is a vector $\boldsymbol{h} = (\boldsymbol{h}^{ij})_{i \leq j}$ of vectors, such that n for every pair of 1-types $i \leq j$ the vector of integers $\boldsymbol{h}^{ij} = (h_1^{ij}, \ldots, h_b^{ij})$ si such that $\boldsymbol{h}^{ij} = (h_1^{ij}, \ldots h_b^{ij})$, where b is the number of 2-tables of Σ and h_l^{ij} contains the number of pairs of domain elements that realize the 2-type ijl.

EXAMPLE 12.18. *Let us consider the interpretation shown in Figure 1. The 1-type cardinality vectors is*

$$\boldsymbol{k} = (4, 3)$$

Indeed we hae that 2, 4, 6 *and* 7 *realize the 1-type* $1(x)$*, and* 1, 3 *and* 5 *realize the 1-type* $2(x)$*. The 2-type cardinality vector is*

$$\boldsymbol{h}^{11} = (0, 1, 1, 4)$$
$$\boldsymbol{h}^{12} = (2, 1, 0, 9)$$
$$\boldsymbol{h}^{22} = (0, 0, 1, 2)$$

Let us summarise some equality about cardinality vectors.

- $\sum \boldsymbol{k} = \sum_{i=1}^{u} k_i = n$. This follows directly from the fact that every element of the domain realizes one and only one 1-type.
- $\sum \boldsymbol{h}^{ii} = \sum_l h_l^{ii} = \frac{k_i(k_i-1)}{2}$ this derives from the fact that if $\{c,d\}$ realizes iil then both c and d realizes i and therefore there are $\binom{k_i}{2=\frac{k_i(k_i-1)}{2}}$ subsets of two elements of a set of k_i elements.
- $\sum \boldsymbol{h}^{ij} = \sum_l h_l^{ij} = k_i \cdot k_j$ (if $i \neq j$) This is a consequence of the fact that every subset $\{a,b\}$ realizes one and only one 2-type ijl with $i \leq j$, and that a and b realize i and j respectively. Therefore there is a total of $k_i k_j$ sets that realizes some 2-table ijl for some l.
- $\sum \boldsymbol{h} = \sum_{i \leq j} \sum_{l=1}^{b} k_l^{ij} = \frac{n(n-1)}{2}$. This is the consequence of the fact that there are $\binom{n}{2}$ subsets of 2 elements of a set of n elements.

6. FOMC of universal formulas

In this section we provide a mathematical formula (a polinomial) that allows to compute the first order model counting of a restricted class of formulas of $\mathcal{L}^2$. They are universal formula that contains no constant and function symbols and only the two variables $x.y$. In other words they are formulas of the form

$$\forall x \forall y \phi(x,y) \tag{175}$$

We start by observing that, for every cardinality vector $(\boldsymbol{k}, \boldsymbol{h})$ there are

$$\binom{n}{\boldsymbol{k}} \prod_i \binom{\frac{k_i(k_i-1}{2}}{\boldsymbol{h}^{ii}} \prod_{i<j} \binom{k_i k_j}{\boldsymbol{h}^{ij}} \tag{176}$$

distinct interpretations that have the cardinality vector $(\boldsymbol{k}, \boldsymbol{h})$ where for every positive integers $a, b_1, \ldots, b_m$ with $\sum_i b_i = a$

$$\binom{a}{b_1, \ldots, b_m} = \frac{a!}{b_1! \cdot b_2! \cdots b_n!}$$

Suppose that for some cardinality vectors $(\boldsymbol{k}, \boldsymbol{h})$, we have that $h_l^{ij} \neq 0$, then every interpretation with this cardinality vector should have a pair $\{c, d\}$ that realizes ijl. If $\mathcal{I}$ satisfies also $\forall xy\phi(x, y)$, then $\mathcal{I}$ should also satisfy the grounding of $\phi(x, y)$ with $\{c, d\}$. I.e., $\phi(c, c) \wedge \phi(d, d) \wedge \phi(c, d) \wedge \phi(d, c)$. But this is only possible if the formula

$$\phi(c, c) \wedge \phi(d, d) \wedge \phi(c, d) \wedge \phi(d, c) \wedge i(c) \wedge j(d) \wedge l(c, d)$$

Let us introduce this notion formally

Definition 12.9. *A 2-type $ijl(x, y)$ is consistent with a universal formula $\forall x \forall y \phi(x, y)$, if and only if the propositional formula*

$$ijl(c, d) \wedge \mathit{Ground}(\phi(x, y), \{c, d\}) \tag{177}$$

for a pair of distinct constants c and d is satisfiable. $2t(\phi)$ denotes The set of 2-types consistent with $\forall x \forall y\, \phi(x, y)$.

Notice that the first part of formula (177) i.e., $ilj(c, d)$, is a conjunction of literals, and contains all the atoms that appears in $\phi(c, d)$. This implies that if (177) is consistent then the only assignment $\mathcal{I}$ that satisfies $ijl(c, d)$, satisfies $\phi(c, c)$, $\phi(c, d)$, $\phi(d, c)$ and $\phi(d, d)$.

A simple method for computing the set $2t(\phi)$ is via truth table

Example 12.19. *Consider the formula. $\forall x \forall y(R(x, x) \wedge x \neq y \wedge R(x, y) \rightarrow \neg R(y, x)))$ Let us compute the grounding of this formula w.r.t., the constants c, d. it is*

$$\begin{aligned} \mathit{Ground}(\forall x \forall y \phi(x, y), \{c, d\}) = {} & R(c, c) \wedge c \neq c \wedge R(c, c) \rightarrow \neg R(c, c) \\ & \wedge R(d, d) \wedge d \neq d \wedge R(d, d) \rightarrow \neg R(d, d) \\ & \wedge R(c, c) \wedge c \neq d \wedge R(c, d) \rightarrow \neg R(d, c) \\ & \wedge R(d, d) \wedge d \neq c \wedge R(d, c) \rightarrow \neg R(c, d) \end{aligned}$$

We have that $c \neq c$ is always false while $c \neq d$ is always true. This allow to simplify the above formula as follows:

$$(R(c, c) \wedge R(c, d) \rightarrow \neg R(d, c)) \wedge (R(d, d) \wedge R(d, c) \rightarrow \neg R(c, d))$$

2-type	$R(c,c)$	$R(d,d)$	$R(c,d)$	$R(d,c)$	$((R(c,c)\wedge R(c,d))\rightarrow\neg R(d,c))\wedge((R(d,d)\wedge R(d,c))\rightarrow\neg R(c,d))$										
$111(c,d)$	T	T	T	T	T	T	T	F	F	F	T	T	T	F	F
$112(c,d)$	T	T	T	F	T	T	T	T	T	T	T	F	F	T	F
$113(c,d)$	T	T	F	T	T	F	F	T	F	T	T	T	T	T	T
$114(c,d)$	T	T	F	F	T	F	F	T	T	T	T	F	F	T	T
$121(c,d)$	T	F	T	T	T	T	T	F	F	F	F	F	T	T	F
$122(c,d)$	T	F	T	F	T	T	T	T	T	T	F	F	F	T	F
$123(c,d)$	T	F	F	T	T	F	F	T	F	T	F	F	T	T	T
$124(c,d)$	T	F	F	F	T	F	F	T	T	T	F	F	F	T	T
$221(c,d)$	F	F	T	T	F	F	T	T	F	T	F	F	T	T	F
$222(c,d)$	F	F	T	F	F	F	T	T	T	T	F	F	F	T	F
$223(c,d)$	F	F	F	T	F	F	F	T	F	T	F	F	T	T	T
$224(c,d)$	F	F	F	F	F	F	F	T	T	T	F	F	F	T	T

Therefore the set $2t(\phi)$ contains the 3-types which evaluates the formula ϕ to be true (i.e.,they are consistent with the formula).

If ijl is not consistent with $\forall xy(\phi(x,y)$, i.e., if $ijl\notin t2(\phi)$, then any interpretation that contains at least one (c,d) that realizes the 2-type ijl should be excluded from the count of the model. All the remaining interpretations will be models of $\forall xy\phi(x,y)$. We can therefore modify equation (176) bu adding an indicator function that excludes these models from the summation.

PROPOSITION 12.5. *A pure universal formula $\forall x\forall y\,\phi(x,y)$ is equivalent to*

$$\forall x\forall y\left(x\neq y\rightarrow\bigvee_{i\leq j}\bigvee_{iji\in 2t(\phi)}ijl(x,y)\right)\tag{178}$$

on the class of models that contains at least 2 elements.

PROOF OUTLINE. Let $\mathcal{I}_1\ldots,\mathcal{I}_k$ be the models of $\forall x\forall y.\phi(x,y)$. For every $\mathcal{I}_j$, every $\{a,b\}\subseteq[n]$ realizes exaclty 1 2-type, ijl which implies that ijl is consistent with $\forall x\forall y\,\phi(x,y)$. Therefore $ijl\in 2t(\phi)$. This implies that $\mathcal{I}_i\models(178)$.

Viceversa suppose that $\mathcal{I}\not\models\forall x\forall y\,\phi(x,y)$. Then either $\mathcal{I}\not\models\phi(a,b)$ for $a\neq b$ (case (1)) or $\mathcal{I}\not\models\phi(a,a)$ for some a (case (2))

(1) If $\mathcal{I}\not\models\phi(a,b)$. Let ijl be the two type realized by c,d in $\mathcal{I}$, we have that $\phi(c,d)\wedge ijl(c,d)$ is not consistent and therefore $ijl(c,d)\notin 2t(\phi)$. Since $\{c,d\}$ can realize only a single 2-type we have that $\mathcal{I}\not\models\bigvee_{ijl\in 2t(\phi)}ijl(a,b)$ and therefore $\mathcal{I}\not\models(178)$
(2) If $\mathcal{I}\not\models\phi(a,a)$ let c be another element of the domain. This c exists since we have at least two elements. Suppose that $i\leq j$ (the proof of the other case is analogous) Let ijl be the 2-type realized by $\{a,c\}$ in $\mathcal{I}$ then we have that $\mathcal{I}\not\models\phi(a,a)\wedge\phi(c,c)\wedge\phi(c,a)\wedge\phi(a,c)\wedge ijl(a,b)$. We are now back to case (1).

□

EXAMPLE 12.20. *$\forall x\forall y(R(x,x)\wedge R(x,y)\rightarrow R(y,y))$ is equivalent to:*

$$\begin{aligned}\forall x\forall y\,(x\neq y\rightarrow\ & 111(x,y)\vee 112(x,y)\vee 113(x,y)\vee 114(x,y)\vee\\ & 123(x,y)\vee 114(x,y)\\ & 221(x,y)\vee 222(x,y)\vee 223(x,y)\vee 224(x,y)\vee\end{aligned}$$

Property 12.5 allow us to transform the problem of counting the models of $\forall x \forall y \phi(x,y)$ in the problem of counting the models of (178). Notice that an interpretation $\mathcal{I}$ with cardinality vectors $(\boldsymbol{k}, \boldsymbol{h})$ is a model of (178) iff for

$$h_l^{ij} \neq 0 \Rightarrow ijl \in 2t(\phi) \tag{179}$$

As a consequence all the models of $\forall x \forall y\, \phi(x,y)$ are those that have a cardinality vector that satisfies the condition f (179). Finally notice that condition (179) can be represented with

$$\mathbb{1}_{ijl \in 2t(\phi)}^{h_l^{ij}} = \begin{cases} 1 & \text{if } h_l^{ij} = 0 \text{ or } n_{ij} \neq 0 \\ 0 & \text{Otherwise} \end{cases}$$

where $\mathbb{1}_{ijl \in 2t(\phi)}$ is the indicator function for the set $2t(\phi)$. We can therefore conclude the following:

$$\begin{aligned} \text{FOMC}(\forall x, y.\phi(x,y), n) &= \sum_{\boldsymbol{k},\boldsymbol{h}} \binom{n}{\boldsymbol{k}} \prod_{i \leq j=1}^{u} \binom{\boldsymbol{k}(i,j)}{\boldsymbol{h}^{ij}} \prod_{l} \mathbb{1}_{ijl \in 2t(\phi)}^{h_l^{ij}} \\ &= \sum_{\boldsymbol{k}} \binom{n}{\boldsymbol{k}} \prod_{i \leq j=1}^{u} \sum_{\boldsymbol{k}} \binom{\boldsymbol{k}(i,j)}{\boldsymbol{h}^{ij}} \prod_{l} \mathbb{1}_{ijl \in 2t(\phi)}^{h_l^{ij}} \\ &= \sum_{\boldsymbol{k},\boldsymbol{h}} \binom{n}{\boldsymbol{k}} \prod_{i \leq j=1}^{u} \left(\sum_{l=1}^{b} \mathbb{1}_{ijl \in 2t(\phi)} \right)^{\boldsymbol{k}(i,j)} \\ &= \sum_{\boldsymbol{k},\boldsymbol{h}} \binom{n}{\boldsymbol{k}} \prod_{i \leq j=1}^{u} n_{ij}^{\boldsymbol{k}(i,j)} \end{aligned}$$

with

$$n_{ij} = \sum_{l=1}^{b} \mathbb{1}_{ijl \in 2t(\phi)}$$

THEOREM 12.1. *Let $\phi(x,y)$ a quantifier free formula that contains p predicate symbols and the two free variables x and y and no constant and functional symbolss;*

$$\text{FOMC}(\forall x, y.\phi(x,y), n) = \sum_{\boldsymbol{k}} \binom{n}{\boldsymbol{k}} \prod_{1 \leq i \leq j \leq u} n_{ij}^{\boldsymbol{k}(i,j)} \tag{180}$$

$$\text{FOMC}(\forall x, y.\phi(x,y), n) = \sum_{\boldsymbol{k},\boldsymbol{h}} \binom{n}{\boldsymbol{k}} \prod_{i \leq j=1}^{u} \binom{\boldsymbol{k}(i,j))}{\boldsymbol{h}^{ij}} \prod_{l} \mathbb{1}_{ijl \in 2t(\phi)}^{h_l^{ij}} \tag{181}$$

- $\boldsymbol{k} = (k_1, k_1, \ldots, k_u)$, *s.t.,* $\sum_{i=1}^{u} k_i = n$;
- $n_{ij} = \#\text{SAT}(Ground(\forall x \forall y \phi(x,y), [2]) \wedge i(1) \wedge j(2))$
- $\boldsymbol{k}(i,j) = \begin{cases} \frac{k_i \cdot (k_j - 1)}{2} & \text{if } i = j \\ k_i \cdot k_j & \text{Otherwise} \end{cases}$

Theore 12.1 provides two formulas for computing the first order model counting of a pure universal formula. The first formula require to consider only the cardinality vector for the 1-types (i.e., $\boldsymbol{k}$). This formula is simpler but as we will see later considering only the cardinality of the 1-types could not be enough to perform *weighted first order model counting*. The second and more complete formula consider also the cardinality vectors for the 2-tables (i.e., $\boldsymbol{h}$). Considering alse these

vectors will become essential when weights of the models are specified as weight on binary predicates.

EXAMPLE 12.21. *Consider the formula* $\Phi = \forall x(\neg R(x,x) \wedge (A(x) \wedge R(x,y) \rightarrow A(y))$. *Let us compute the first order model counting in the domain of 4 elements, i.e* $\text{FOMC}(\Phi, 4)$.

- *Let us first determine which are the 1-types and the 2-table for this formula*

1-types	**2-tables**
$1(x) = A(x) \wedge R(x,x)$	$1(x,y) = R(x,y) \wedge R(y,x)$
$2(x) = A(x) \wedge \neg R(x,x)$	$2(x,y) = R(x,y) \wedge \neg R(y,x)$
$3(x) = \neg A(x) \wedge R(x,x)$	$3(x,y) = \neg R(x,y) \wedge R(y,x)$
$4(x) = \neg A(x) \wedge \neg R(x,x)$	$4(x,y) = \neg R(x,y) \wedge \neg R(y,x)$

- *then we have to compute* $2t(\phi$ *i.e., the 2-types which are consistent with* ϕ. *For this we conside rthe formula*

$$\begin{aligned} \mathit{Ground}(\Phi, \{c,d\}) = \neg R(c,c) \wedge \neg R(d,d) \\ \wedge (A(c) \wedge R(c,c) \rightarrow A(c)) \\ \wedge (A(c) \wedge R(c,d) \rightarrow A(d)) \\ \wedge (A(d) \wedge R(d,c) \rightarrow A(c)) \\ \wedge (A(d) \wedge R(d,d) \rightarrow A(d)) \end{aligned}$$

That can be simplified in consistent with ϕ. *For this we conside rthe formula*

$$\neg R(c,c) \wedge \neg R(d,d) \wedge (A(c) \wedge R(c,d) \rightarrow A(d)) \wedge (A(d) \wedge R(d,c) \rightarrow A(c)) \tag{182}$$

For this we could compute the truth table for all the 2-types, however this will require a truth table with $6 \cdot 4 = 24$. *We can simplify this computation by observing that all the 1-types that contains* $R(x,x)$ *are not consistent with* (182). *So it is enoug to consider the 1-types* $2(x)$ *and* $4(x)$.

	$A(c)$	$A(d)$	$R(c,c)$	$R(d,d)$	$R(c,d)$	$R(d,c)$	(182)
$221(c,d)$	T	T	F	F	T	T	T
$222(c,d)$	T	T	F	F	T	F	T
$223(c,d)$	T	T	F	F	F	T	T
$224(c,d)$	T	T	F	F	F	F	T
$241(c,d)$	T	F	F	F	T	T	F
$242(c,d)$	T	F	F	F	T	F	F
$243(c,d)$	T	F	F	F	F	T	T
$244(c,d)$	T	F	F	F	F	F	T
$441(c,d)$	F	F	F	F	T	T	T
$442(c,d)$	F	F	F	F	T	F	T
$443(c,d)$	F	F	F	F	F	T	T
$444(c,d)$	F	F	F	F	F	F	T

Therefore we have that $2t(\phi) = \{221, 222, 223, 224, 243, 244, 441, 442, 443, 444\}$

- *from* $2t(\phi$ *we can compute* n_{ij} *which is the number of* $ijl \in 2t(\phi)$

$$n_{22} = 4 \qquad n_{24} = 2 \qquad n_{44} = 4$$

All the others are equal to 0*. We now have all the elements to compute formula* (180)*.*

$$\begin{aligned}\text{FOMC}(\Phi,4) = &\binom{4}{4,0,0,0}n_{11}^6\\ &+\binom{4}{3,1,0,0}n_{11}^3n_{12}^3\\ &+\dots\\ &+\binom{4}{0,4,0,0}n_{22}^6\\ &+\binom{4}{0,3,1,0}n_{22}^3n_{23}^3\\ &+\dots\end{aligned}$$

Notice however that all the n_{ij} where i or j are equal to 1 *or* 3 *are equal to* 0*, and therefore when k_i ore k_j is different form there the resulting term will be equal to* 0*, not contributing to the sum This means that we can concentrate only on k_2 and k_4. We can therefore simplify the formula in:*

$$\begin{aligned}\text{FOMC}(\Phi,4) &= \sum_{k_2+k_4=4}\binom{4}{k_2,k_4}n_{22}^{\frac{k_2(k_2-1)}{2}}n_{24}^{k_2k_4}n_{44}^{\frac{k_4(k_4-1)}{2}}\\ &= \sum_{k_2=0}\binom{4}{k_2}4^{\frac{k_2(k_2-1)}{2}}2^{k_2(4-k_2)}4^{\frac{(4-k_2)(3-k_2)}{2}}\\ &= \sum_{k_2=0}^{4}\binom{4}{k_2}2^{k_2(k_2-1)+k_2(4-k_2)+(4-k_2)(3-k_2)}\\ &= \sum_{k_2=0}^{4}\binom{4}{k_2}2^{k_2^2-4k_2+12}\\ &= 2^{12}+4\cdot 2^9+6\cdot 2^8+4\cdot 2^9+2^{12}\\ &= 3\cdot 2^{12}+6\cdot 2^8 = 13824\end{aligned}$$

In the above equations we use the simplified notation $\binom{n}{k_2}$ in place of $\binom{n}{k_1,k_2}$. This is the standard notation for the binomial coefficient where for every pair of integers $a \geq b$. $\binom{a}{b} = \binom{a}{b,a-b} = \frac{a!}{b!(a-b)!}$

7. Cardinality Constraints

A *cardinality constraint* is an arithmetic expression that imposes restrictions on the number of (pairs of) individual objects that belong to the interpretation of a certain predicate. In other words, a cardinality constraint imposes some restriction on the size of $\mathcal{I}(P_1),\dots\mathcal{I}(P_k)$ for some predicates $P_1,\dots,P_k$. A simple example of a cardinality constraint is $|A| = m$, for some unary predicate A and positive integer m. This cardinality constraint is satisfied by any interpretation $\mathcal{I}$ in which $\mathcal{I}(A)$ contains exactly m distinct individual objects. A more complex example of

a cardinality constraint could be: $|A| + |B| \leq |C|$, where A, B and C are some predicates in the language.

Notice that, the fact that an interpretation $\mathcal{I}$ satisfies a cardinality constraint γ depends only from its cardinality vector $(\boldsymbol{k}, \boldsymbol{h})$ of the interpretation. Indeed the cardinality of unary and binary predicates of an interpretation $\mathcal{I}$ can be directly computed starting from the cardinality vector of $\mathcal{I}$.

DEFINITION 12.10 (Satisfiability of a cardinality constraint). *For every predicate P we can compute $|\mathcal{I}(P)|$ from the cardinality vectors $\boldsymbol{k}, \boldsymbol{h}$ of $\mathcal{I}$ as follows, where A is a unary predicate and R a binary predicate.*

$$
\begin{aligned}
|\mathcal{I}(A)| = \boldsymbol{k}(A) &= \sum_{i=1}^{u} |\{A(x)\} \cap i(x)| \cdot k_i \\
\boldsymbol{k}(R) &= \sum_{i=1}^{u} |\{R(x,x)\} \cap i(x)| \cdot k_i \\
\boldsymbol{h}(R) &= \sum_{l=1}^{b} |\{R(x,y), R(y,x)\} \cap l(x)| \cdot h_l \\
|\mathcal{I}(R)| = (\boldsymbol{k}, \boldsymbol{h})(R) &= \boldsymbol{k}(R) + \boldsymbol{h}(R)
\end{aligned}
$$

where, for every 2-table l, $h_l = \sum_{i \leq j=1}^{u} h_l^{ij}$. If γ is a cardinality constraint then $\mathcal{I} \models \gamma$ holds if the expression obtained replacing $|A|$ with the value of $\boldsymbol{k}(A)$ and $|R|$ with the value of $(\boldsymbol{k}, \boldsymbol{h})(R)$ is true, where $\boldsymbol{k}, \boldsymbol{h}$ is the cartinality vectors of $\mathcal{I}$.

EXAMPLE 12.22. *Consider the formumrla $\Phi = \forall x \forall y\, A(x) \wedge R(x,y) \rightarrow A(y)$. This formula has the following 1-types and 2-tables:*

1-types	**2-tables**
$1(x) = A(x) \wedge R(x,x)$	$1(x,y) = R(x,y) \wedge R(y,x)$
$2(x) = A(x) \wedge \neg R(x,x)$	$2(x,y) = R(x,y) \wedge \neg R(y,x)$
$3(x) = \neg A(x) \wedge R(x,x)$	$3(x,y) = \neg R(x,y) \wedge R(y,x)$
$4(x) = \neg A(x) \wedge \neg R(x,x)$	$4(x,y) = \neg R(x,y) \wedge \neg R(y,x)$

With the following n_{ij}

$n_{11} = 4$	$n_{12} = 4$	$n_{13} = 2$	$n_{14} = 2$
	$n_{22} = 4$	$n_{23} = 2$	$n_{24} = 2$
		$n_{33} = 4$	$n_{34} = 4$
			$n_{44} = 4$

Suppose that we are interested in counting the models of Φ on a domain of 5 elements with the cardinality constraint $|A| = 3$, i.e., $\text{FOMC}(\Phi \wedge |A| = 3, 5)$. *Since the cardinality constraints involves only a unary predicate, we can adopt the formula* (180) *that sum over all possible cardinality vectors for unary predicates, and restrict*

the cardinality vectors that satisfies $\boldsymbol{k}(A) = 3$ *i.e.,* $k_1 + k_2 = 3$.

$$\begin{aligned}\text{FOMC}(\forall x, y.\Phi \wedge |A| = 3, 5) &= \sum_{\substack{k_1+k_2+k_3+k_4=5 \\ k_1+k_2=3}} \binom{5}{k_1, k_2, k_3, k_4} \prod_{1\le i\le j\le u} n_{ij}^{\boldsymbol{k}(i,j)} \\ &= \sum_{\substack{k_1+k_2=3 \\ k_3+k_4=2}} \binom{3}{k_1}\binom{2}{k_3} \prod_{1\le i\le j\le u} n_{ij}^{\boldsymbol{k}(i,j)}\end{aligned}$$

I f we want for instance to impose an additional cardinality constraint $|R| = 2$ *on the binary predicate* R*, then we have to consider the expanded version of the formula for FOMC, i.e., formula* (181) *and additionally restrict the* $\boldsymbol{h}$ *vector to satisfy* $\boldsymbol{k}, \boldsymbol{h}(R) = 2$

$$\begin{aligned}&\text{FOMC}(\forall x, y.\Phi \wedge |A| = 3 \wedge |R| = 2, 5) = \\ &\sum_{\substack{k_1+k_2=3 \\ k_3+k_4=2}} \binom{3}{k_1}\binom{2}{k_3} \sum_{\substack{\boldsymbol{h} \\ 2\cdot h_1+h_2+h_3+k_1+k_2=2}} \prod_{i\le j=1}^{u} \binom{\boldsymbol{k}(i,j)}{\boldsymbol{h}^{ij}} \prod_l \mathbb{1}_{ijl\in 2t(\phi)}^{h_l^{ij}}\end{aligned}$$

8. Dealing with Existential Quantifiers

In order to perform model counting of formulas that contain existential quantifier we suppose that the formula is on a special form called Scott's Normal form. In the following subsection we introduce such a form and show how every formula can be transformed in Scott's normal form which is counting-equivalent, i.e., the resultinf formula has the same number of models of the original formula.

8.1. Scott's Normal Form.

THEOREM 12.2 (Scott's Normal Form Scott 1962 and Kuusisto and Lutz 2018). *Every* FO^2 *sentence* Φ *in the signature* Σ *can be transformed in a formula*

$$\Phi' = \forall xy.\phi(x, y) \wedge \bigwedge_{i=1}^{m} \forall x\exists y.\psi_i(x, y) \tag{183}$$

where ϕ *and* ψ_i *are quantified free formulas in the signature* $\Sigma' = \Sigma \cup \{P_1, \ldots, P_m\}$ *for* m *new unary predicates* P_i*, such that every* Σ*-structure* $\mathcal{I}$ *that satisfies* Φ *can be extended in a unique way in a* Σ'*-structure that satisfies* Φ'.

PROOF OUTLINE. To transform a formula Φ in Scott's normal form you have to apply the following transformations until the formula does not contain subformulas of the form $Qy.\alpha(x, y)$ for some quantifier $Q \in \{\forall, \exists\}$

- If $Qy.\alpha(x, y)$ is a subformula of Φ and $\alpha(x, y)$ does not contain quantifiers, then replace it with a new predicate $P(x)$ and *define* $P(x)$ as $Qy.\alpha(x, y)$; Collect all the definition of the predicates $P(x)$ in Γ.

$$\begin{aligned}\Phi &\Longrightarrow \Phi[Qy.\alpha(x, y)/P(x)] \\ \Gamma &\Longrightarrow \Gamma \wedge \forall x.(A(x) \leftrightarrow Qy.\alpha(x, y))\end{aligned}$$

- Transform the formula $\forall x P(x) \leftrightarrow \forall y \alpha(x,y))$ that belongs to Γ in the following way

$$\forall x(P(x) \leftrightarrow Qy.\alpha(x,y)) \Longrightarrow \forall x(P(x) \rightarrow Qy.\alpha(x,y)) \wedge \forall x(\neg P(x) \rightarrow \bar{Q}y.\neg\alpha(x,y))$$

  where $\bar{Q}$ is the dual quantifier than Q, (i.e., if Q is $\forall$ then $\bar{Q}$ is $\exists$ and viceversa).
- then transform each implication as follows:

$$\begin{aligned}
\forall x(P(x) \rightarrow \forall y \alpha(x,y)) &\Longrightarrow \forall x \forall y.(P(x) \rightarrow \alpha(x,y)) \\
\forall x(\neg P(x) \rightarrow \exists y \neg\alpha(x,y)) &\Longrightarrow \forall x \exists y(\neg P(x) \rightarrow \neg\alpha(x,y)) \\
\forall x(P(x) \rightarrow \exists y \alpha(x,y)) &\Longrightarrow \forall x \exists y(P(x) \rightarrow \alpha(x,y)) \\
\forall x(\neg P(x) \rightarrow \forall y.\neg\alpha(x,y)) &\Longrightarrow \forall x \forall y(\neg P(x) \rightarrow \neg\alpha(x,y))
\end{aligned}$$

□

EXAMPLE 12.23. *Consider the formula*

$$\forall x(A(x) \rightarrow \exists y(R(x,y) \wedge \forall x(S(y,x) \rightarrow B(x)))) \tag{184}$$

We start by replacing the subformula $\forall x(S(y,x) \rightarrow B(x)$ with $P_1(y)$ and we add the definition of P_1 obtaining

$$\begin{aligned}
&\forall x(A(x) \rightarrow \exists y(R(x,y) \wedge P_1(y))) \\
&\quad \wedge \forall x(P_1(x) \leftrightarrow \forall y(S(x,y) \rightarrow B(y)))
\end{aligned}$$

then we replace the formula $\exists y(R(x,y) \wedge P_1(y))$ with $P_2(x)$ and add the definition of P_2 obtaining:

$$\begin{aligned}
&\forall x(A(x) \rightarrow P_2(x)) \\
&\quad \wedge \forall x(P_1(x) \leftrightarrow \forall y(S(x,y) \rightarrow B(y))) \\
&\quad \wedge \forall x(P_2(x) \leftrightarrow \exists y(R(x,y) \wedge P_1(y)))
\end{aligned}$$

Finally we replace the equivalence with the implication and move out the quantifiers obtaining

$$\begin{aligned}
&\forall x(A(x) \rightarrow P_2(x)) \\
&\quad \wedge \forall x \forall y(P_1(x) \rightarrow (S(x,y) \rightarrow B(y))) \\
&\quad \wedge \forall x \exists y(\neg P_1(x) \rightarrow \neg(S(x,y) \rightarrow B(y))) \\
&\quad \wedge \forall x \exists y(P_2(x) \rightarrow (R(x,y) \wedge P_1(y))) \\
&\quad \wedge \forall x \forall y(\neg P_2(x) \rightarrow \neg(R(x,y) \wedge P_1(y)))
\end{aligned}$$

8.2. Inclusion-Exclusion principle. In this section, we provide a proof for model counting of formulas in Scott's normal form by meking explicit use of the principle of inclusion-exclusion. A corollary of the principle of inclusion-exclusion that will be used for preforming FOMC is the following:

COROLLARY 1 (Wilf 2005 section 4.2). *Let Ω be a set of objects and let $\mathcal{S} = \{S_1, \ldots, S_m\}$ be a set of subsets of Ω. For every $\mathcal{Q} \subseteq \mathcal{S}$, let $N(\supseteq \mathcal{Q})$ be the count of objects in Ω that belong to all the subsets $S_i \in \mathcal{Q}$, i.e., $N(\supseteq \mathcal{Q}) = \left|\{\bigcap_{S_i \in Q} S_i\}\right|$.*

For every $0 \leq l \leq m$, *let* $s_l = \sum_{|\mathcal{Q}|=l} N(\supseteq \mathcal{Q})$ *and let* e_0 *be count of objects that* d *onot belong to any of the* S_i *in* $\mathcal{S}$, *then*

$$e_0 = \sum_{l=0}^{m} (-1)^l s_l \tag{185}$$

THEOREM 12.3. *For an* FO^2 *formula in Scott's Normal Form as given in* (183), *let* $\Phi' = \forall xy.(\Phi(x,y) \wedge \bigwedge_{i=1}^{q} P_i(x) \rightarrow \neg\Psi_i(x,y))$ *where* P_i*'s are fresh unary predicates, then:*

$$\text{FOMC}((183), n) = \sum_{(\boldsymbol{k},\boldsymbol{h})} (-1)^{\sum_i \boldsymbol{k}(P_i)} \text{FOMC}(\Phi', (\boldsymbol{k}, \boldsymbol{h})) \tag{186}$$

PROOF. TO BE REVISED Let Ω be the set of models of $\forall xy.\Phi(x,y)$ over the language of Φ and $\{\Psi_i\}$ (i.e., the language of Φ' excluding the predicates P_i) and on a domain Δ consisting of n elements. Let $\mathcal{S} = \{\Omega_{ci}\}_{c\in\Delta,\ 1\leq i\leq q}$ be the set of subsets of Ω where Ω_{ci} is the set of ω such that $\omega \models \forall y.\neg\Psi_i(c,y)$. For every model ω of (183), $\omega \not\models \forall y\neg\Psi_i(c,y)$ for any pair of i and c i.e. ω is not in any Ω_{ci}. Also, for every $\omega \in \Omega$, if $\omega \notin \Omega_{ci}$ for any pair of i and c, then $\omega \models \exists y.\Psi_i(c,y)$ for all i and for all $c \in \Delta$ i.e., $\omega \models \bigwedge_{i=1}^{q} \forall x\exists y.\Psi_i(x,y)$. Hence, $\omega \models (183)$ if and only if $\omega \notin \Omega_{ci}$ for all c and i. Therefore, the count of models of (183) is equal to the count of models in Ω which do not belong to any Ω_{ci}. Hence, If we are able to compute s_l (as introduced in Corollary 1), then we could use Corollary 1 for computing cardinality of all the models which do not belong to any Ω_{ci} and hence $\text{FOMC}((183), n)$.

For every $0 \leq l \leq n \cdot q$, let us define

$$\Phi'_l = \Phi' \wedge \sum_{i=1}^{q} |P_i| = l \tag{187}$$

We will now show that s_l is exactly given by $\text{FOMC}((187), n)$.

Every model of Φ'_l is an extension of an $\omega \in \Omega$ that belongs to at least l elements in $\mathcal{S}$. In fact, for every model ω of $\forall xy.\Phi(x,y)$ i.e. $\omega \in \Omega$, if $\mathcal{Q}'$ is the set of elements of $\mathcal{S}$ that contain ω, then ω can be extended into a model of Φ'_l in $\binom{|Q'|}{l}$ ways. Each such model can be obtained by choosing l elements in Q' and interpreting $P_i(c)$ to be true in the extended model, for each of the l chosen elements $\Omega_{ci} \in Q'$. On the other hand, recall that $s_l = \sum_{|\mathcal{Q}|=l} N(\supseteq Q)$. Hence, for any $\omega \in \Omega$ if $\mathcal{Q}'$ is the set of elements of $\mathcal{S}$ that contain ω, then there are $\binom{|\mathcal{Q}'|}{l}$ distinct subsets $\mathcal{Q} \subseteq \mathcal{Q}'$ such that $|\mathcal{Q}| = l$. Hence, we have that ω contributes $\binom{|\mathcal{Q}'|}{l}$ times to s_l. Therefore, we can conclude that

$$s_l = \text{FOMC}(\Phi'_l, n) = \sum_{|\mathcal{Q}|=l} N(\supseteq Q)$$

and by the principle of inclusion-exclusion as given in Corollary 1, we have that :

$$\begin{aligned}
\text{FOMC}((183), n) = e_0 &= \sum_{l=0}^{n\cdot q}(-1)^l s_l \\
&= \sum_{l=0}^{n\cdot q}(-1)^l \text{FOMC}(\Phi'_l, n) \\
&= \sum_{l=0}^{n\cdot q}(-1)^l \sum_{(\boldsymbol{k},\boldsymbol{h})\models\sum_i |P_i|=l} \text{FOMC}(\Phi', (\boldsymbol{k}, \boldsymbol{h})) \\
&= \sum_{(\boldsymbol{k},\boldsymbol{h})} (-1)^{\sum_i \boldsymbol{k}(P_i)} \text{FOMC}(\Phi', (\boldsymbol{k}, \boldsymbol{h}))
\end{aligned}$$

□

9. Exercises

Exercise 177:
Show that if $|\Delta^{\mathcal{I}}| = 1$ then $\mathcal{I} \models \forall x\phi(x) \leftrightarrow \exists x\phi(x)$

Exercise 178:
Prove that $\text{FOMC}(\phi, [1]) = \#\text{SAT}(\text{Ground}(\phi, \{1\}))$.

Exercise 179:
List all the models of the formula

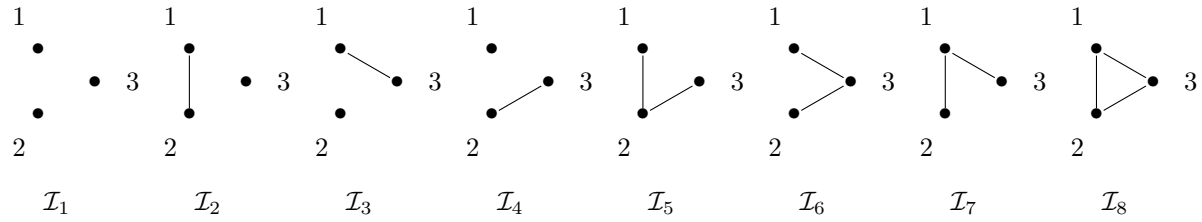
$$\forall x \neg R(x, x) \wedge \forall xy(R(x, y) \rightarrow R(y, x))$$

in the domain $\{1, 2, 3\}$.

Solution The intuitive reading of the formula is as follows:

- every object is not related with itself. (i.e., R is not reflexive)
- R is symmetric.

This means that we are interested in all the undirected graphs on the three edges 1, 2, and 3. Therefore for every subset of pairs of objects, there is a model. Which implies that the number of models are $2^{\binom{3}{2}} = 2^3 = 8$. A graphical representation of the models are shown in the following:a



□

Exercise 180:
List all the models of the formula $\forall x(A(x) \rightarrow B(x))$ on the interpretation domain $\{1, 2, 3\}$.

Exercise 181:
Let $\mathcal{I}$ be a first order interpretation and $\pi : \Delta^{\mathcal{I}} \rightarrow \Delta^{\mathcal{I}}$ be a isomorphism. Show that the interpretation $\mathcal{I}^{\pi}$ where

$$\mathcal{I}^{\pi}(a) \triangleq \pi(\mathcal{I}(a))$$
$$\mathcal{I}^{\pi}(f) \triangleq (d_t \dots d_n) \mapsto \pi(\pi^{-1}(d_1), \dots, \pi^{-1}(d_n))$$
$$\mathcal{I}^{\pi}(R) \triangleq \{(\pi(d_1), \dots, \pi(d_n) \mid (d_1, \dots, d_n) \in \mathcal{I}(R)\}$$

is such that $\mathcal{I} \models \phi$ if and only if $\mathcal{I}^{\pi} \models \phi$ for every first order sentence ϕ.

Exercise 182:
Provide an explicit mathematical formula to compute the number of models of $\forall x(A(x) \rightarrow B(x))$ in the domain $\{1, 2, 3, \dots, n\}$.

Solution The models of $forallx.A(x) \to B(x)$ are those in which B is interpreted in a subset of the interpretation of A. Notice that we have $\binom{n}{k}$ possible ways of interpreting A in a set of k objects. For any such interpretation B can take any subset of the interpretation of A, i.e, 2^k. therefore the set of interpretations are

$$\sum_{k=0}^{n} \binom{n}{k} 2^k = (1+2)^n = 3^n$$

□

Exercise 183:

For the following formuls ϕ describe all the models in the domain $\{1,2,3\}$ and find an explicit mathematical formula that computes $\text{FOMC}(\phi, n)$

$$\forall x(A(x) \leftrightarrow \forall y R(x,y))$$

Solution Notice that, in the above formula, the interpretation of R fully determines the interpretation of A, and furthermore R can be freely interpreted in any subset of pairs of elements of $\{1,2,3\}$. This means that the number of interpretations that satisfies the above formula coincides with the number of interpretations of R, which are $2^{3\cdot 3} = 2^9$. □

Exercise 184:

Provide an explicit mathematical formula to compute the number of models of the formula of the previous exercise in the domain $\{1,2,3,\ldots,n\}$.

Exercise 185:

For each formuls ϕ in the following list, describe all the models in the domain $\{1,2,3\}$ and find an explicit mathematical formula that computes $\mathsf{fomc}(\phi, n)$

(1) $\exists x A(x)$
(2) $\exists x \neg A(x)$
(3) $\neg \forall x \neg A(x)$
(4) $\forall x \exists y R(x,y)$
(5) $\exists x \forall y \neg R(x,y)$
(6) $\forall x(A(x) \to \exists y R(x,y))$
(7) $\forall x \exists y R(x,y) \wedge \forall xyz R(x,y) \wedge R(x,z) \to y = z$
(8) $\forall xy(R(x,y) \to A(x) \wedge \neg A(y))$

Solution

(1) $\exists x A(x)$: the set of models interpret A in a non empty subset of $\{1,2,3\}$. The number of non empty subsets of $\{1,2,3\}$ are $2^3 - 1$.
(2) $\exists x \neg A(x)$ the set of models interpret A in a set different from the entire domain. The number of such sets is $2^3 - 1$.
(3) $\neg \forall x \neg A(x)$: This formula is equivalent to $\exists x A(x)$. See item 1.
(4) $\forall x \exists y R(x,y)$. The set of models of such a formula are the interpretations that associates to every element $d \in \{1,2,3\}$ at least one element $d' \in \{1,2,3\}$ such that $(d,d') \in R^{\mathcal{I}}$. Therefore a model of this formula associates to every element $d \in \{1,2,3\}$ a non empty subset $D \subseteq \{1,2,3\}$ such that $(d,d') \in R^{\mathcal{I}}$ for all $d' \in D$. Since the number of non empty

subsets of $\{1,2,3\}$ is 2^3-1 we have that the number of models of the formula is equal to $(2^3-1)^3 = 7^3 = 343$.

(5) $\exists x \forall y \neg R(x,y)$. This formula is equivalent to $\neg \forall x \exists y R(x,y)$. Therefore the number of models of this formula is the total number of interpretations (which is equal to 2^9) minus the numbr of models of $\forall x \exists y \neg R(x,y)$ which is equal to $(2^3-1)^3$ (see previous point). This means that the number of models of the formula is equal to $2^9 - (2^3-1)^3$.

(6) $\forall x(A(x) \rightarrow \exists y R(x,y))$. An interpretation can associate to A any subset of $\{1,2,3\}$. Given an interpretation of A, for every element in the interpretation of A, R should associate a non empty set of elements of $\{1,2,3\}$. This means that, if A contains k elements, we have 2^3-1 possibilities. If A contains k elements there are $(2^3-1)^k$ possibilities for the other $n-k$ element R can be interpreted freely, allowing $2^{3(n-k)}$. Therefore in total there are $(2^3-1)^k 2^{3(3-k)}$ models where A is interpreted in a set of k elements. Since there are $\binom{3}{k}$ possible interpretations of A that contains k elements, the total number of interpretations are

$$\sum_{k=0}^{3} \binom{3}{k} ((2^3-1)^k + 2^{3(3-k)})$$

(7) $\forall x \exists y R(x,y) \wedge \forall xyz R(x,y) \wedge R(x,z) \rightarrow y = z$ This formula states that R is a total function on the domain $\{1,2,3\}$ i.e., for every element $d \in \{1,2,3\}$ it associates one and only one element $d' \in \{1,2,3\}$ such that $(d,d') \in R^{\mathcal{I}}$. The number of functions on a set of n elements are n^n, in this case we have $3^3 = 27$ models.

(8) $\forall xy(R(x,y) \rightarrow A(x) \wedge \neg A(y))$. The formula state that the interpretation of R must be a subset of $A^{\mathcal{I}} \times (\neg A)^{\mathcal{I}}$. If A is interpreted in a set of k k then R can be interpreted any subset of $k(n-k)$ pairs. So there are $2^{k(n-k)}$ possible interpretations of R. Since there are $\binom{n}{k}$ interpretations of A that contains k elements, the total number of models of the formula are:

$$\sum_{k=0}^{3} \binom{n}{k} 2^{k(n-k)}$$

□

Exercise 186:

Provide an explicit mathematical formula to compute the number of models of $\Phi \triangleq \forall x(A(x) \rightarrow B(f(x)))$ in the domain $\{1,2,3,\ldots,n\}$. To find this formula you first have to find (and describe in text) a procedure on how you can build a model for Φ.

Solution For every i in A, we have to find an element j in B such that $f(i) = j$. if B contains b elements we have b possibilities. So if we have a elements in a we have b^a possibilities. For any other element j which is not in A we can fix $f(j)$ to be any element of the domain. so we have n possibilites. If A contains a elements we have n^{n-a} possibilites. Considering all the possible cardinalities of A and B, we therefore have

$$\text{FOMC}(\Phi, n) = \sum_{a=0}^{n} \binom{n}{a} \sum_{b=0}^{n} \binom{n}{b} b^a n^{n-a}$$

□

Exercise 187:
Ground the formula $\forall x \exists y R(x, y)$ in the domain $\{a, b, c\}$.

Solution

$$\bigwedge_{x \in \{a,b,c\}} \bigvee_{y \in \{a,b,c\}} R(x, y)$$

□

Exercise 188:
Ground the formula $\forall x(A(x) \rightarrow \exists x R(x, x))$ in the domain $\{a, b, c\}$. **Solution**

$$\bigwedge_{d \in \{a,b,c\}} (A(d) \rightarrow \bigvee_{e \in \{a,b,c\}} R(d, e))$$

Extensively written is:

$$\begin{aligned} & (A(a) \rightarrow R(a, a) \vee R(a, b) \vee R(a, c)) \\ \wedge\ & (A(b) \rightarrow R(b, a) \vee R(b, b) \vee R(b, c)) \\ \wedge\ & (A(c) \rightarrow R(c, a) \vee R(c, b) \vee R(c, c)) \end{aligned}$$

□

Exercise 189:
Let $\mathcal{L}$ be a first-order language that contains the unary predicate A and the binary predicate R. Compute directly (without using the general formula) the number of models of one of the following formulas on a domain of n elements.

(1) $\forall x, y.(R(x, y) \rightarrow A(x) \wedge \neg A(y))$
(2) $\forall x, y.(A(x) \wedge A(y) \rightarrow R(x, y))$
(3) $\forall x.(A(x) \rightarrow \exists y R(x, y))$

Solution

(1) $\forall x, y.(R(x, y) \rightarrow A(x) \wedge \neg A(y))$

$$\sum_{k=1}^{n} \binom{n}{k} 2^{k(n-k)} \tag{188}$$

(2) $\forall x, y.(A(x) \wedge A(y) \rightarrow R(x, y))$

$$\sum_{k=1}^{n} \binom{n}{k} (2^n - 2^{k(n-k)}) \tag{189}$$

(3) $\forall x.(A(x) \rightarrow \exists y R(x, y))$

$$\sum_{k=1}^{n} \binom{n}{k} k(2^n - 1)(n - k)2^n \tag{190}$$

□

Exercise 190:
How many ground atoms occour in a formula that contains $A(x)$, $B(y)$ and $R(x, y)$ when it is grounded with the set of constants $\{c_1, \ldots, c_n\}$?

Exercise 191:
Compute the grounding of $\forall x(\exists y A(x,y) \to \forall z \neg B(z,w))$

Exercise 192:
Find a formula for computing $\text{FOMC}(\forall x R(a,x), n)$ where a is a constant.

Exercise 193:
Find a formula for computing $\text{FOMC}(\forall x(A(x) \to A(f(x)), n)$ where f is a function symbol.

Exercise 194:
Find a formula that computes $\text{FOMC}(\forall xy(R(x,y) \to \exists z S(x,y,z)), n)$

Solution Suppose that $\mathcal{I}(R)$ contains r pairs then for every pair $(a,b) \in \mathcal{I}(R)$ we can associate a non empty subset $S_{a,b}$ such that $(a,b,c) \in \mathcal{I}(S)$ for all $c \in S_{a,b}$. This results in:

$$\sum_{r=0}^{n^2} \binom{n^2}{r} (2^n - 1)^r (2^n)^{n^2 - r} = (2^n - 1 + 2^n)^{n^2} = (2^{n+1} - 1)^{n^2}$$

□

Exercise 195:
List all the 1-types of the signature $\Sigma = \{A/1, B/1, R/2\}$.

Exercise 196:
List all the 1-types, and 2-tables of the signature $\Sigma = \{A/1, B/1, R/2, S/2\}$.

Exercise 197:
What is the cardinality vectors $\boldsymbol{k}, \boldsymbol{h}$ of the following Σ-structure on the domain [6] where $\Sigma = \{A/1, B/1, R/2)$:

$$\mathcal{I}(A) = \{1,3,5\}$$
$$\mathcal{I}(B) = \{3,4,5\}$$
$$\mathcal{I}(R) = \{(2,2),(3,3),(3,5),(6,4),(5,3),(3,2)\}$$

Exercise 198:
Compute the set $2t(\phi)$ for the following formulas:
(1) $\forall x \forall y(A(x) \wedge R(x,y) \to A(y))$;
(2) $\forall x \forall y(R(x,y) \to A(x) \wedge B(y))$;
(3) $\forall x \forall y(A(x) \wedge B(y) \to x \neq y)$.

Exercise 199:
In the formula

$$\text{FOMC}(\forall x, y.\phi_0(x,y), n) = \sum_{\boldsymbol{k}} \binom{n}{\boldsymbol{k}} \prod_{0 \leq i \leq 2^p - 1} n_{ij}^{k(i,j)}$$

first order model counting if $\Phi_0(x,y)$ is $A(x) \wedge R(x,y) \rightarrow \neg A(y)$, specify the values of:

(1) the length of $\boldsymbol{k}$
(2) the number of 1-types
(3) n_{ij} for every pair of 1-type $i \leq j$

Exercise 200:

Compute $\text{FOMC}(\Phi, 4)$ for the following formulas using the formula (180) (in parenthesis the result)

(1) $\forall x \forall y(R(x,y) \rightarrow A(x) \wedge \neg A(y))$ (162)
(2) $\forall x \forall y(R(x,y) \vee R(y,x))$ (729)
(3) $\forall x \forall y(A(x) \wedge B(y) \leftrightarrow R(x,y))$ (256)

Exercise 201:

Using the formula for first order model counting of universal formulas in $\mathcal{L}^2$ compute $\text{FOMC}(\Phi, 3)$ where Φ is the following fomrula:

(1) $\forall x \forall y(R(x,y) \rightarrow R(y,x))$;
(2) $\forall x \forall y(R(x,y) \rightarrow \neg R(y,x))$;
(3) $\forall x \forall y(R(x,y) \rightarrow \neg R(x,x))$;
(4) $\forall x \forall y(R(x,x) \rightarrow (R(x,y) \rightarrow R(y,z)))$;

Solution

(1) $\forall x \forall y(R(x,y) \rightarrow R(y,x))$; Let us first compute the n_{ij} using the truth table

$R(x,x)$	$R(y,y)$	$(R(x,x) \rightarrow R(x,x)) \wedge R(x,y) \rightarrow R(y,x))$ $\wedge\ R(y,x) \rightarrow R(x,y)) \wedge R(y,y) \rightarrow R(y,y))$
0	0	$n_{00} = 2$
0	1	$n_{01} = 2$
1	0	$n_{10} = 2$
1	1	$n_{11} = 2$

Notice that n_{ij} is the number of models of the formula $(R(x,x) \rightarrow R(x,x)) \wedge$ $R(x,y) \rightarrow R(y,x)) \wedge R(y,x) \rightarrow R(x,y))$ when $R(x,x)$ is interpreted in i and $R(y,y)$ is interpreted in j. Here we have 1 unary predicate R which is applied to x (obtaining $R(x,x)$ and to y obtaining $R(y,y)$). This means that the dimension of $\boldsymbol{k}$ is equal to $2^p = 2$, i.e, $\boldsymbol{k} = (k_0, k_1)$. If we expan

the fomula we obtain

$$\sum_{\boldsymbol{k}\in\{(0,3),(1,2),(2,1),(3,0)\}} \binom{3}{\boldsymbol{k}} n_{00}^{\boldsymbol{k}(0,0)} n_{01}^{\boldsymbol{k}(0,1)}, n_{11}^{\boldsymbol{k}(1.1)}$$

$$= \sum_{\boldsymbol{k}\in\{(0,3),(1,2),(2,1),(3,0)\}} \binom{3}{\boldsymbol{k}} 2^{\boldsymbol{k}(0,0)+\boldsymbol{k}(0,1)+\boldsymbol{k}(1.1)}$$

$$=\binom{3}{0}2^{\frac{0(0-1)}{2}+0\cdot 3+\frac{3(3-1)}{2}} + \binom{3}{1}2^{\frac{1(1-1)}{2}+1\cdot 2+\frac{2(2-1)}{2}}$$

$$+\binom{3}{2}2^{\frac{2(2-1)}{2}+2\cdot 1+\frac{1(1-1)}{2}} + \binom{3}{3}2^{\frac{3(3-1)}{2}+3\cdot 0+\frac{0(0-1)}{2}}$$

$$=2^3 + 3\cdot 2^3 + 3\cdot 2^3 + 2^3 = 2^3 2^3 = 2^6$$

(2) $\forall x \forall y (R(x,y) \rightarrow \neg R(y,x))$; Let us compute first n_{ij}.

$R(x,x)$	$R(y,y)$	$(R(x,x) \rightarrow \neg R(x,x)) \wedge R(x,y) \rightarrow \neg R(y,x))$ $\wedge R(y,x) \rightarrow R\neg(x,y)) \wedge R(y,y) \rightarrow \neg R(y,y))$
0	0	$n_{00} = 2$
0	1	$n_{01} = 0$
1	0	$n_{10} = 0$
1	1	$n_{11} = 0$

The dimentions are the same as in the previous formula since we have only the predicate R.

$$\sum_{\boldsymbol{k}\in\{(0,3),(1,2),(2,1),(3,0)\}} \binom{3}{\boldsymbol{k}} n_{00}^{\boldsymbol{k}(0,0)} n_{01}^{\boldsymbol{k}(0,1)}, n_{11}^{\boldsymbol{k}(1.1)}$$

$$= \sum_{\boldsymbol{k}\in\{(0,3),(1,2),(2,1),(3,0)\}} \binom{3}{\boldsymbol{k}} 2^{\boldsymbol{k}(0,0)} 0^{\boldsymbol{k}(0,1)+\boldsymbol{k}(1.1)}$$

$$=\binom{3}{0}2^{\frac{0(0-1)}{2}} 0^{0\cdot 3+\frac{3(3-1)}{2}} + \binom{3}{1}2^{\frac{1(1-1)}{2}} 0^{1\cdot 2+\frac{2(2-1)}{2}}$$

$$+\binom{3}{2}2^{\frac{2(2-1)}{2}} 0^{2\cdot 1+\frac{1(1-1)}{2}} + \binom{3}{3}2^{\frac{3(3-1)}{2}} 0^{+3\cdot 0+\frac{0(0-1)}{2}}$$

$$=1\cdot 0 + 3\cdot 0 + 3\cdot 0 + 1\cdot 2^3 = 8$$

(3) $\forall x \forall y (R(x,y) \rightarrow \neg R(x,x))$;
(4) $\forall x \forall y (R(x,x) \rightarrow (R(x,y) \rightarrow R(y,z)))$;

□

Exercise 202:

The formula for first order model counting of formulas that contain exitantial quantifiers is.

$$\text{FOMC}\left(\forall x, y.\phi(x,y) \wedge \forall x \exists y \psi(x,y), n\right) =$$

$$\sum_{\boldsymbol{k}} \binom{n}{\boldsymbol{k}} (-1)^{\boldsymbol{k}(P)} \prod_{0\leq i\leq j\leq 2^{p+1}-1} n_{ij}^{k(i,j)}$$

Answer the following questions about the elements of the above mathematical formula when: $\phi(x,y)$ be $R(x,y)$ and $\psi(x,y)$ equal to $Q(x,y)$

(1) what is P?
(2) what is the value of p
(3) what is the lenght of $\boldsymbol{k}$
(4) on which formula do you compute n_{ij}

Exercise 203:

Using the formula for first order model counting of formulas that contains exitantial quantifiers, compute $\text{FOMC}(\Phi, n)$ where Φ is one of the following fomrula:

- $\forall x.\exists y.R(x.y)$
- $\forall x.\exists y.(R(x,y) \vee R(y,x))$
- $\forall x, y.(R(x,y) \rightarrow R(y,x)) \wedge \forall x \neg R(x,x) \wedge \forall x.\exists y.R(x.y)$

Exercise 204:

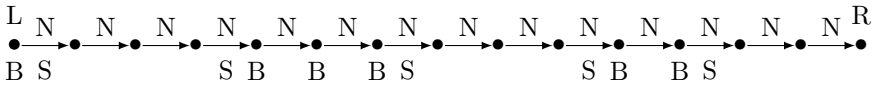
Formalize the following problem in FOL and formulate the solution in terms of FOMC (you don't need to actually compute the solution).

> Suppose that 6 boys and 9 girls line up in a row. Let S be the number of places in the row where a boy and a girl are standing next to each other. For example, for the row GBGGB-BGBBGGGBGG we have S = 8. The average value of S (if all possible orders of these 15 people are considered) is closest to.

Solution Let Φ be the conjunction of the following formulas.

$$\begin{aligned} L(x) &\leftrightarrow \forall y \neg N(y,x) \\ R(x) &\leftrightarrow \forall x \neg N(x,y) \\ \forall x(\neg R(x) &\rightarrow \exists y N(x,y)) \\ S(x,y) &\leftrightarrow N(x,y) \wedge (B(x) \leftrightarrow \neg B(y)) \\ |L| = |R| &= 1 \\ |B| &= 6 \\ |N| &= 14 \end{aligned}$$

Any model of Φ on the domain of 15 elements has the following structure



where the six labels B can be randomly assignet to any of the elements of the domain. $FOMC(\Phi, 15)$ therefore count how many of such configurations exists. For every $\boldsymbol{k}$ we can compute the cardinality of S, denoted by $\boldsymbol{k}(S)$. Therfore the problem can be solved by computing

$$\frac{\sum_{\boldsymbol{k}} \boldsymbol{k}(S) \binom{15}{\boldsymbol{k}} \prod_{i \leq j} n_{ij}^{\boldsymbol{k}(i,j)}}{\sum_{\boldsymbol{k}} \binom{15}{\boldsymbol{k}} \prod_{i \leq j} n_{ij}^{\boldsymbol{k}(i,j)}}$$

□

Exercise 205:

Formalize the following problem in FOL and formulate the solution in terms of FOMC (you don't need to actually compute the solution).

A mission to Mars will consist of 4 astronauts selected from 14 available. Exactly 5 of the 14 are trained in exobiology. If the mission requires at least 2 trained in exobiology, how many different crews can be selected?

Solution We can easily formulate the problem as $FOMC(\Phi, 14)$ where Φ is the following formula.

$$|E| = 5 \wedge |M| = 4 \wedge |M \cap E| = 2$$

Since there are no FOL formulas, and no binary predicates, we have that have that $n_{ij} = 1$ for all ij which implies that the $FOMC$ formula reduces to

$$\sum_{\substack{\boldsymbol{k}(E)=2, \boldsymbol{k}(M)=5 \\ \boldsymbol{k}(M\cap E)=2}} \binom{14}{\boldsymbol{k}}$$

which is equal to

$$\binom{14}{7,3,2,2}$$

However this number include also all the possible choices of the experts in exobiology, which is known. We have therefore to devide it by all the possible subset of 5 expers among the 14 astronauts, i.e., $\binom{14}{5}$. The final result therefore is

$$\frac{\binom{14}{7,3,2,2}}{\binom{14}{5}}$$

□

Exercise 206:
Using the formula for FOMC

$$\text{FOMC}(\forall x, y.\phi(x, y), n) = \sum_{\boldsymbol{k}} \binom{n}{\boldsymbol{k}} \prod_{0 \le i \le 2^p - 1} n_{ij}^{k(i,j)}$$

compute $\text{FOMC}(\forall x, y(A(x) \wedge R(x, y) \rightarrow A(y)), 3)$

Solution We first have to compute the n_{ij}. Notice that we have 2 unary predicates $A(x)$ and $R(x, x)$. THerefore we have that $0 \le i, j \le 2^2 - 1 = 3$

$A(x)$	$R(x,x)$	$A(y)$	$R(y,y)$	n_{ij}
0	0	0	0	$n_{00} = 4$
0	0	0	1	$n_{01} = 4$
0	0	1	0	$n_{02} = 2$
0	0	1	1	$n_{03} = 2$
0	1	0	1	$n_{11} = 4$
0	1	1	0	$n_{12} = 2$
0	1	1	1	$n_{13} = 2$
1	0	1	0	$n_{22} = 4$
1	0	1	1	$n_{23} = 4$
1	1	1	1	$n_{33} = 4$

The expansion of the formula for $\text{FOMC}(\phi, n)$ for $n = 3$ is the following.

$$\begin{array}{lllllllll}
\binom{3}{3,0,0,0} n_{00}^3 & + & \binom{3}{2,1,0,0} n_{00} n_{01}^2 & + & \binom{3}{2,0,1,0} n_{00} n_{02}^2 & + \\
\binom{3}{2,0,0,1} n_{00} n_{03}^2 & + & \binom{3}{1,2,0,0} n_{01}^2 n_{11} & + & \binom{3}{1,1,1,0} n_{01} n_{02} n_{12} & + \\
\binom{3}{1,1,0,1} n_{01} n_{03} n_{13} & + & \binom{3}{1,0,2,0} n_{02}^2 n_{22} & + & \binom{3}{1,0,1,1} n_{02} n_{03} n_{23} & + \\
\binom{3}{1,0,0,2} n_{22} n_{23}^2 & + & \binom{3}{0,3,0,0} n_{11}^3 & + & \binom{3}{0,2,1,0} n_{11} n_{12}^2 & + \\
\binom{3}{0,2,0,1} n_{11} n_{13}^2 & + & \binom{3}{0,1,2,0} n_{12}^2 n_{22} & + & \binom{3}{0,1,1,1} n_{12} n_{13} n_{23} & + \\
\binom{3}{0,1,0,2} n_{13}^3 n_{33} & + & \binom{3}{0,0,3,0} n_{22}^3 & + & \binom{3}{0,0,2,1} n_{22} n_{23}^2 & + \\
\binom{3}{0,0,1,2} n_{22}^3 n_{33} & + & \binom{3}{0,0,0,3} n_{33}^3 & & &
\end{array}$$

The result can be obtaine by replacing the value of n_{ij}, in the above expression. □

Exercise 207:
Use the formula for first order model counting to compute:

$$\text{FOMC}(R(x,y) \to P(y,x), 4)$$

Solution Let us recall the formula for first order model counting for unviversally quantified formulas

$$\text{FOMC}(\forall x, y.\phi_0(x,y), n) = \sum_{\boldsymbol{k}} \binom{n}{\boldsymbol{k}} \prod_{0 \leq i \leq 2^p - 1} n_{ij}^{k(i,j)}$$

- $\boldsymbol{k} = (k_0, k_1, \ldots, k_{2^p-1})$, s.t., $\sum_{i=1}^{2^p-1} k_i = n$;
- $\binom{n}{\boldsymbol{k}} = \frac{n!}{k_0! \cdot k_1! \cdots k_{2^p-1}!}$
- $n_{ij} = \#\text{SAT}(Ground(\phi_0(x,y) \wedge \alpha_i(x) \wedge \alpha_j(y), \{a, b\})$
- $\alpha_i(x) = \bigwedge_{\substack{b=1 \\ i_b=0}}^{p} \neg A_b(x) \wedge \bigwedge_{\substack{b=1 \\ i_b=1}}^{p} A_b(x)$
- $k(i,j) = \begin{cases} \frac{k_i \cdot (k_j - 1)}{2} & \text{if } i = j \\ k_i \cdot k_j & \text{Otherwise} \end{cases}$

Let us determine all the quantities contained in the formulas $p = 2$, since we have the unary atoms $R(x,x)$ and $P(x,x)$. Therefore we have to compute n_{ij} for $0 \leq i \leq j \leq 3$. Let $\Phi(a,b)$ be the grounding of $R(x,y) \to P(y,x)$ in the domain $\{a,b\}$, i.e.,

$$\begin{array}{c}
\Phi(a,b) = (R(a,a) \to P(a,a)) \wedge (R(b,b) \to P(b,b)) \\
(R(a,b) \to P(b,a)) \wedge (R(b,a) \to P(a,b)))
\end{array}$$

then the n_{ij} are the following:

$$
\begin{aligned}
n_{00} &= \#\text{SAT}(\neg R(a,a) \wedge \neg R(b,b) \wedge \neg P(a,a) \wedge \neg P(b,b) \wedge \Phi(a,b)) && = 9\\
n_{01} &= \#\text{SAT}(\neg R(a,a) \wedge \neg R(b,b) \wedge \neg P(a,a) \wedge P(b,b) \wedge \Phi(a,b)) && = 9\\
n_{02} &= \#\text{SAT}(\neg R(a,a) \wedge \neg R(b,b) \wedge P(a,a) \wedge \neg P(b,b) \wedge \Phi(a,b)) && = 9\\
n_{03} &= \#\text{SAT}(\neg R(a,a) \wedge \neg R(b,b) \wedge P(a,a) \wedge P(b,b) \wedge \Phi(a,b)) && = 9\\
n_{11} &= \#\text{SAT}(\neg R(a,a) \wedge R(b,b) \wedge \neg P(a,a) \wedge P(b,b) \wedge \Phi(a,b)) && = 9\\
n_{12} &= \#\text{SAT}(\neg R(a,a) \wedge R(b,b) \wedge P(a,a) \wedge \neg P(b,b) \wedge \Phi(a,b)) && = 0\\
n_{13} &= \#\text{SAT}(\neg R(a,a) \wedge R(b,b) \wedge P(a,a) \wedge P(b,b) \wedge \Phi(a,b)) && = 9\\
n_{22} &= \#\text{SAT}(R(a,a) \wedge \neg R(b,b) \wedge P(a,a) \wedge \neg P(b,b) \wedge \Phi(a,b)) && = 9\\
n_{23} &= \#\text{SAT}(R(a,a) \wedge \neg R(b,b) \wedge P(a,a) \wedge P(b,b) \wedge \Phi(a,b)) && = 9\\
n_{33} &= \#\text{SAT}(R(a,a) \wedge R(b,b) \wedge P(a,a) \wedge P(b,b) \wedge \Phi(a,b)) && = 9
\end{aligned}
$$

We can now replace the elements in the general formula obtaining

$$
\begin{aligned}
\sum_{\boldsymbol{k}} \binom{4}{\boldsymbol{k}} \prod_{\substack{0\leq i\leq j\leq 3\\(i,j)\neq(1,2)}} 9^{k(i,j)} &= \sum_{\boldsymbol{k}} \binom{4}{\boldsymbol{k}} 9^{6-k_1k_2}\\
&= \sum_{\substack{\boldsymbol{k}\\k_1k_2=0}} \binom{4}{\boldsymbol{k}} 9^6 + \sum_{\substack{\boldsymbol{k}\\k_1k_2=1}} \binom{4}{\boldsymbol{k}} 9^5 + \sum_{\substack{\boldsymbol{k}\\k_1k_2=2}} \binom{4}{\boldsymbol{k}} 9^4 + \sum_{\substack{\boldsymbol{k}\\k_1k_2=3}} \binom{4}{\boldsymbol{k}} 9^3 + \sum_{\substack{\boldsymbol{k}\\k_1k_2=4}} \binom{4}{\boldsymbol{k}} 9^2
\end{aligned}
$$

□

Exercise 208:
Using the formula in the slides compute the first order model counting for $\forall xy(R(x,y) \rightarrow \neg R(x,x) \wedge \neg R(y,y))$ in the domain of 3 elmenets. **Solution**

We have 1 unary predicate which is $R(x,x)$ (it is binary but applied to the same variable it becomes like a unary predicate). Therefore we have to compute n_{00} n_{01} and n_{11}. Each n_{ij} indicates the number of assignments to unary and binary atoms

that makes true the grounding of the formula with two elements a, b.

$R(a,a)$	$R(b,b)$	$R(a,b)$	$R(b,a)$	$R(a,a) \to \neg R(a,a) \wedge \neg R(a,a) \wedge$ $R(a,b) \to \neg R(a,a) \wedge \neg R(b,b) \wedge$ $R(b,a) \to \neg R(b,b) \wedge \neg R(a,a) \wedge$ $R(b,b) \to \neg R(b,b) \wedge \neg R(b,b)$
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Therefore $n_{00} = 4$ and $n_{01} = n_{11} = 0$. The formula for first order model counting is

$$\sum_{\boldsymbol{k}} \binom{n}{\boldsymbol{k}} \prod_{i \leq j} n_{ij}^{k(i,j)}$$

where n is the size of the domain and $\boldsymbol{k}$ is a vector of 2^u positive integers that sum to n, where u is the number of unary predicates. In our case $n = 3$, $u = 1$, and therefore $\boldsymbol{k}$ is a vector containing two elements that sum to 3. We can therefore use the binomial coefficient $\binom{n}{k_0}$ instead of the multinomial $\binom{n}{k_0, k_1}$, as they are equivalent. The expansion of the fomula is as follows:

$$\sum_{k_0=0}^{3} n_{00}^{\frac{k_0(k_0-1)}{2}} n_{01}^{k_0(3-k_0)} n_{11}^{\frac{(3-k_0)(3-k_0)}{2}}$$

$$= \sum_{k_0=0}^{3} 4^{\frac{k_0(k_0-1)}{2}} 0^{k_0(3-k_0)} 0^{\frac{(3-k_0)(3-k_0)}{2}}$$

Notice that if $k \neq 3$ then the product is equal to 0 and therefore it does not contributes to the sum. We have only to consider the case in which $k_0 = 2$, obtaining the following expression

$$4^{\frac{3 \cdot 2}{2}} = 4^3 = 2^6$$

□

Exercise 209:

Using the formula for first order model counting compu¯te the number of models of the FO^2-formula

$$\forall xy(R(x,x) \to (R(x,y) \to R(y,x)))$$

Solution Let us first determine which are the 1-types and the 2-tables. The

1-types are

$$1(x) \triangleq R(x,x),$$
$$2(x) \triangleq \neg R(x,x),$$

and the following 2-tables

$$1(x,y) \triangleq R(x,y) \wedge R(y,x) \wedge x \neq y$$
$$2(x,y) \triangleq R(x,y) \wedge \neg R(y,x) \wedge x \neq y$$
$$3(x,y) \triangleq \neg R(x,y) \wedge R(y,x) \wedge x \neq y$$
$$4(x,y) \triangleq \neg R(x,y) \wedge \neg R(y,x) \wedge x \neq y$$

Now let us compute n_{11}, n_{12} and n_{22}. To do so we have to do the grounding of the formula obtaining:

$$(R(c,c) \to (R(c,c) \to R(c,c))) \wedge$$
$$(R(c,c) \to (R(c,d) \to R(d,c))) \wedge$$
$$(R(d,d) \to (R(d,c) \to R(c,d))) \wedge$$
$$(R(d,d) \to (R(d,d) \to R(d,d)))$$

which can be simplified in

$$(R(c,d) \to (R(c,d) \to R(d,c))) \wedge (R(d,d) \to (R(d,d) \to R(c,d))) \tag{191}$$

Let us now construct the truth table

2-type	$R(c,c)$	$R(d,d)$	$R(c,d)$	$R(d,c)$	(191)
$111(c,d)$	T	T	T	T	T
$112(c,d)$	T	T	T	F	F
$113(c,d)$	T	T	F	T	F
$114(c,d)$	T	T	F	F	T
$121(c,d)$	T	F	T	T	T
$122(c,d)$	T	F	T	F	T
$123(c,d)$	T	F	F	T	T
$124(c,d)$	T	F	F	F	T
$221(c,d)$	F	F	T	T	T
$222(c,d)$	F	F	T	F	T
$223(c,d)$	F	F	F	T	T
$224(c,d)$	F	F	F	F	T

from which we have that $n_{11} = 2$, $n_{12} = 3$ and $n_{22} = 4$ We can then replace in the formula for FOMC

$$\sum_{k=1}^{4} \binom{4}{k} n_{11}^{\frac{k(k-1)}{2}} n_{12}^{k(4-k} n_{22}^{\frac{(4-k)(3-k)}{2}} = 4^6 + 4(3^3 4^3) + 6(2^1 3^4 4^1) + 4(2^3 3^3) + 2^6$$

□

Bibliography

Arp, Robert, Barry Smith, and Andrew D Spear (2015). *Building ontologies with basic formal ontology*. Mit Press.

Badreddine, Samy et al. (2022). "Logic tensor networks". In: *Artificial Intelligence* 303, p. 103649.

Baumgartner, Peter et al. (2009). "Computing finite models by reduction to function-free clause logic". In: *Journal of Applied Logic* 7.1, pp. 58–74.

Birnbaum, Elazar and Eliezer L Lozinskii (1999). "The good old Davis-Putnam procedure helps counting models". In: *Journal of Artificial Intelligence Research* 10, pp. 457–477.

Borgo, Stefano and Claudio Masolo (2009). "Foundational choices in DOLCE". In: *Handbook on ontologies*. Springer, pp. 361–381.

Boros, Endre and Peter L Hammer (2002). "Pseudo-boolean optimization". In: *Discrete applied mathematics* 123.1-3, pp. 155–225.

Brewka, Gerhard (1989). "Nonmonotonic Logics–A Brief Overview". In: *AI Communications* 2.2, pp. 88–97.

Chakraborty, Supratik et al. (2015). "From weighted to unweighted model counting". In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Chavira, Mark and Adnan Darwiche (2008a). "On probabilistic inference by weighted model counting". In: *Artificial Intelligence* 172.6-7, pp. 772–799.

— (2008b). "On probabilistic inference by weighted model counting". In: *Artificial Intelligence* 172.6, pp. 772–799. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2007.11.002. URL: https://www.sciencedirect.com/science/article/pii/S0004370207001889.

Daniele, Alessandro and Luciano Serafini (2019). "Knowledge enhanced neural networks". In: *PRICAI 2019: Trends in Artificial Intelligence: 16th Pacific Rim International Conference on Artificial Intelligence, Cuvu, Yanuca Island, Fiji, August 26–30, 2019, Proceedings, Part I 16*. Springer, pp. 542–554.

Darwiche, Adnan (2020). "Three modern roles for logic in AI". In: *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pp. 229–243.

Davis, Ernest (2017). "Logical formalizations of commonsense reasoning: a survey". In: *Journal of Artificial Intelligence Research* 59, pp. 651–723.

Davis, Martin, George Logemann, and Donald Loveland (1962). "A machine program for theorem proving". In: *Communications of the ACM* 5.7, pp. 394–397.

Davis, Martin and Hillary Putnam (1960). "A computing procedure for quantification theory". In: *Journal of ACM* 7, pp. 201–215.

De Raedt, Luc et al. (2020). "From statistical relational to neuro-symbolic artificial intelligence". In: *arXiv preprint arXiv:2003.08316*.

Franco, John and Marvin Paull (1983). "Probabilistic analysis of the Davis Putnam procedure for solving the satisfiability problem". In: *Discrete Applied Mathematics* 5.1, pp. 77–87.

Fu, Zhaohui and Sharad Malik (2006). "On solving the partial MAX-SAT problem". In: *International Conference on Theory and Applications of Satisfiability Testing*. Springer, pp. 252–265.

Gomes, Carla P., Ashish Sabharwal, and Bart Selman (2009). "Model Counting". In: *Handbook of Satisfiability*, pp. 633–654.

Gruber, Thomas R (1993). "A translation approach to portable ontology specifications". In: *Knowledge acquisition* 5.2, pp. 199–220.

Guizzardi, RS (2015). "Towards ontological foundations for conceptual modeling: the unified foundational ontology (UFO) story Appl". In: *Ontol* 10, pp. 3–4.

Gurevich, Yuri (1985). "Chapter XIII: Monadic second-order theories". In: *Model-theoretic logics* 8, pp. 479–506.

Gutmann, Bernd, Ingo Thon, and Luc De Raedt (2011). "Learning the parameters of probabilistic logic programs from interpretations". In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011. Proceedings, Part I 11*. Springer, pp. 581–596.

Holtzen, Steven, Guy Van den Broeck, and Todd Millstein (2020). "Scaling exact inference for discrete probabilistic programs". In: *Proceedings of the ACM on Programming Languages* 4.OOPSLA, pp. 1–31.

Jaeger, Manfred and Guy Van den Broeck (2012). "Liftability of probabilistic inference: Upper and lower bounds". In: *Proceedings of the 2nd international workshop on statistical relational AI*.

Kimmig, Angelika et al. (2011). "On the implementation of the probabilistic logic programming language ProbLog". In: *Theory and Practice of Logic Programming* 11.2-3, pp. 235–262.

Kuusisto, Antti and Carsten Lutz (2018). "Weighted model counting beyond two-variable logic". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*. Ed. by Anuj Dawar and Erich Grädel. ACM, pp. 619–628. DOI: 10.1145/3209108.3209168. URL: https://doi.org/10.1145/3209108.3209168.

Lenzerini, Maurizio (2002). "Data integration: A theoretical perspective". In: *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 233–246.

Lifschitz, Vladimir, Bruce Porter, and Frank Van Harmelen (2008). *Handbook of Knowledge Representation*. Elsevier.

Lowd, Daniel and Pedro Domingos (2007). "Efficient weight learning for Markov logic networks". In: *Knowledge Discovery in Databases: PKDD 2007: 11th European Conference on Principles and Practice of Knowledge Discovery in Databases, Warsaw, Poland, September 17-21, 2007. Proceedings 11*. Springer, pp. 200–211.

Lukasiewicz, Thomas (1998). "Probabilistic Logic Programming." In: *ECAI*, pp. 388–392.

Maathuis, Marloes et al. (2018). *Handbook of graphical models*. CRC Press.

Mala, Firdous Ahmad (2022). "On the number of transitive relations on a set". In: *Indian Journal of Pure and Applied Mathematics* 53.1, pp. 228–232.

Manhaeve, Robin et al. (2018). "Deepproblog: Neural probabilistic logic programming". In: *advances in neural information processing systems* 31.

Manquinho, Vasco, Joao Marques-Silva, and Jordi Planes (2009). "Algorithms for weighted boolean optimization". In: *International conference on theory and applications of satisfiability testing*. Springer, pp. 495–508.

McCarthy, John (1959). "Programs with Common Sense". In: pp. 77–84.

McCune, William (2003). "Mace4 reference manual and guide". In: *arXiv preprint cs/0310055*.

Minker, Jack (2012). *Logic-based artificial intelligence*. Vol. 597. Springer Science & Business Media.

Mohamedou, Nouredine Ould and Jordi Planes (2009). "Solver MaxSatz in MaxSAT Evaluation 2009". In: *SAT 2009 competitive events booklet: preliminary version*, p. 155.

OEIS Foundation Inc. (n.d.). *The On-Line Encyclopedia of Integer Sequences*. Published electronically at `http://oeis.org`.

Poole, David (2003). "First-order probabilistic inference". In: *IJCAI*. Vol. 3, pp. 985–991.

Reger, Giles, Martin Suda, and Andrei Voronkov (2016). "Finding finite models in multi-sorted first-order logic". In: *International Conference on Theory and Applications of Satisfiability Testing*. Springer, pp. 323–341.

Richardson, Matthew and Pedro Domingos (2006). "Markov logic networks". In: *Machine learning* 62, pp. 107–136.

Russell, Stuart and Peter Norvig (2010). *Artificial Intelligence: A Modern Approach*. 3rd ed. Prentice Hall.

Sang, Tian, Paul Beame, and Henry Kautz (2005). "Solving Bayesian networks by weighted model counting". In: *Proc.AAAI-05*. Vol. 1, pp. 475–482.

Saveri, Gaia and Luca Bortolussi (2022). "Graph Neural Networks for Propositional Model Counting". In: *arXiv preprint arXiv:2205.04423*.

Scott, D. (1962). "A decision method for validity of sentences in two variables". In: *Journal of Symbolic Logic* 27, p. 377.

Selman, Bart, Henry A Kautz, Bram Cohen, et al. (1993). "Local search strategies for satisfiability testing." In: *Cliques, coloring, and satisfiability* 26, pp. 521–532.

Song, Shaoxu et al. (2014). "Repairing vertex labels under neighborhood constraints". In: *Proceedings of the VLDB Endowment* 7.11, pp. 987–998.

Torlak, Emina and Daniel Jackson (2007). "Kodkod: A relational model finder". In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, pp. 632–647.

Tseytin, Grigori (1966). *On the complexity of derivation in propositional calculus*. Presented at the Leningrad Seminar on Mathematical Logic. URL: `http://www.decision-procedures.org/handouts/Tseitin70.pdf`.

Wei, Wei and Bart Selman (2005). "A new approach to model counting". In: *International Conference on Theory and Applications of Satisfiability Testing*. Springer, pp. 324–339.

Wilf, Herbert S (2005). *Generatingfunctionology*. CRC press.

Xiao, Guohui et al. (2018). "Ontology-based data access: A survey". In: International Joint Conferences on Artificial Intelligence.

Zhang, Jian and Hantao Zhang (1996). "System description generating models by SEM". In: *Automated Deduction—Cade-13: 13th International Conference on Automated Deduction New Brunswick, NJ, USA, July 30–August 3, 1996 Proceedings 13*. Springer, pp. 308–312.