Logic for knowledge representation, learning, and inference

Luciano Serafini

Contents

CHAPTER 1

Knowledge Representation in First Order Logic

Contents

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

(1) *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 (1) is not easy (actually it is impossible in FOL).

DEFINITION 1.1. *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] https://plato.stanford.edu/entries/hilbert-program/#4

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 1.2. *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) 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 1.2. These formulas are

$$\forall x \forall y \forall z (R(x, y) \wedge R(y, z) \to R(x, z)) \tag{2}$$

$$\forall x \forall y (R(x, y) \to \neg R(y, x)) \tag{3}$$

$$\forall x \neg R(x, x) \tag{4}$$

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

$$\forall x \forall y (R(x, y) \vee R(y, x) \vee x = y) \tag{5}$$

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

$$\forall x \forall y (R(x, y) \to \exists z (R(x, z) \wedge R(z, y))) \tag{6}$$

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 1.3. *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:

$$\forall x R(x, x) \tag{7}$$

$$\forall x \forall y (R(x, y) \to R(y, x)) \tag{8}$$

$$\forall x \forall y \forall z (R(x, y) \wedge R(y, z) \to R(x, z)) \tag{9}$$

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 $0, 1, 2, \dots$, They are $\underline{0}, s(\underline{0}), s(s(\underline{0})), \dots$. 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

$$\forall x (\neg s(x) = \underline{0}) \tag{10}$$

$$\forall x \forall y (s(x) = s(y) \to x = y) \tag{11}$$

$$\forall x (x + \underline{0} = x) \tag{12}$$

$$\forall x \forall y (x + s(y) = s(x + y)) \tag{13}$$

$$\forall x (x \cdot \underline{0} = \underline{0}) \tag{14}$$

$$\forall x \forall y (x \cdot s(y) = (x \cdot y) + x) \tag{15}$$

$$\phi(\underline{0}) \wedge \forall x (\phi(x) \to \phi(s(x))) \to \forall x \phi(x) \tag{16}$$

(16) 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 1.4. *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

$$\forall x \neg E(x, x) \tag{17}$$

For undirected graphs we have to add also the fact that E is symmetric i.e. the axiom

$$\forall x \forall y (E(x, y) \rightarrow E(y, x)) \tag{18}$$

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 1.5.

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.

$$\forall x \left(\bigvee_{i=1}^{n} p_i(x) \right) \wedge \forall x \forall y \left(\bigwedge_{i<j=1}^{n} \neg (p_i(x) \wedge p_j(x)) \right) \tag{19}$$

$$\forall x \forall y \left(\bigwedge_{i<j=1}^{n} \neg (r_i(x, y) \wedge r_j(x, y)) \right) \tag{20}$$

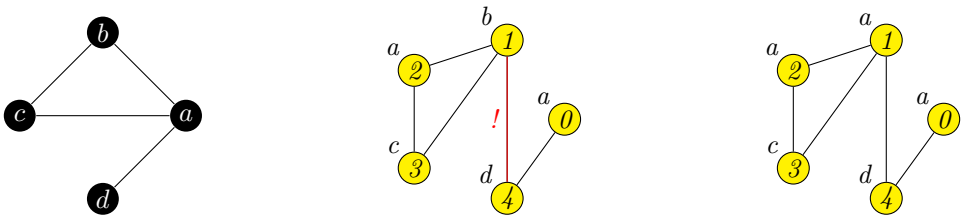

FIGURE 1. The graph on the left specifies the constraints that a labelling with a, b, c, d should satisfy. It is a graph whose nodes are the labels, and arcs represents admitted labelling of adjacent nodes. In the center, we show a graph on 5 nodes which are labelled with a, b, c, and d. This labelling, 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.

Axiom (19) states that every vertex shuld be labelled with at least one lable. Instead, axiom (20) 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 (20) is vacuously true.

Additional logical formulas can be used to axiomatize other constrains on the labels of the graph. Consider the following example:

EXAMPLE 1.1. *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 $\ell : V \to L$ of G with labels L satisfies the constrains specified in S if the following condition holds:*

$$(v, v') \in V \text{ implies } (\ell(v), \ell(v')) \in C \tag{21}$$

The constraint (21) *can be easily expressed in first order logic.*

$$\forall x \forall y \left(r(x, y) \to \bigvee_{(i,j) \in C} (p_i(x) \wedge p_j(y) \vee p_j(x) \wedge p_i(y)) \right) \tag{22}$$

Therefore the class of graphs that satisfies (22) *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)

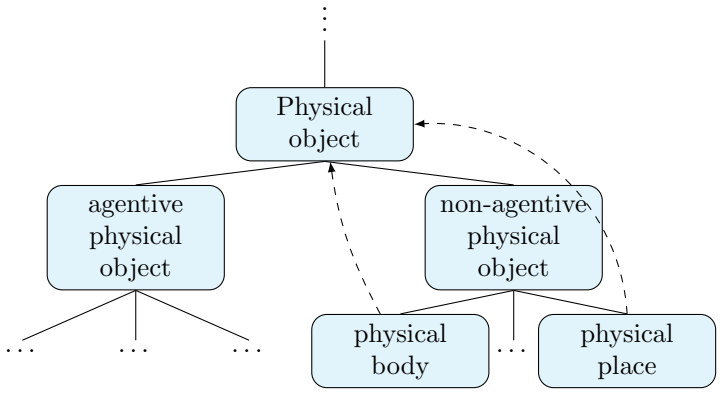

FIGURE 2. An excerpt of the concept hierarchy of DOLCE.

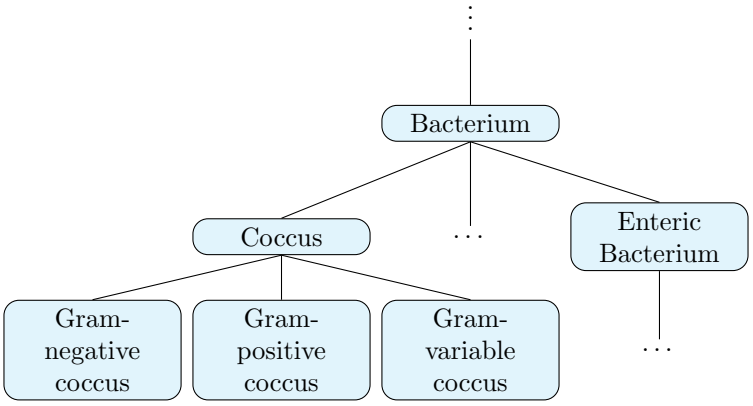

FIGURE 3. An excerpt of one of the concept hierarchies of SNOMED CT.

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

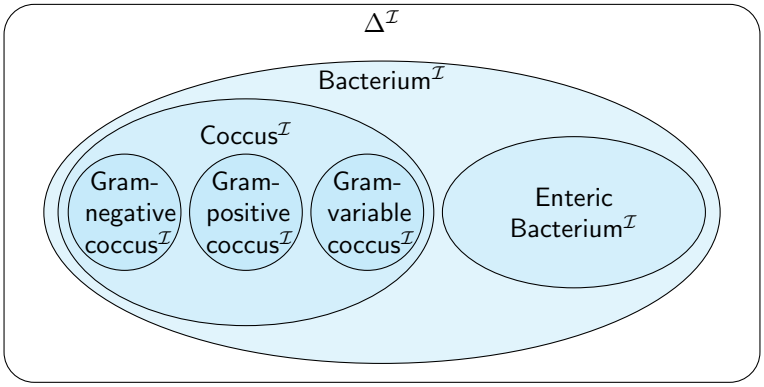

FIGURE 4. A visualization of the FOL semantics of the exceprt of SNOMED CT of Figure 3

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, 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:

$$\forall x(c(x) \to d(x)) \tag{23}$$

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 (23) 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 (23). 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

$$\Gamma_T \models \forall x(c_n(x) \to c_1(x)) \tag{24}$$

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:

$$\forall x \left(\bigwedge_{\substack{(c,d),(c,e)\in H \\ d}} \neg(d(x) \wedge e(x)) \right) \tag{25}$$

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

$$\forall x \left(c(x) \to \bigvee_{(c,d)\in H} d(x) \right) \tag{26}$$

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

$$\forall x \, (\mathsf{Coccus}(x) \to \mathsf{GramNegativeCoccus}(x) \vee \mathsf{GramPositiveCoccus}(x) \vee \mathsf{GramVariableCoccus}(x))$$

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

$$\forall x \forall y(r(x, y) \to c(x)) \tag{27}$$

$$\forall x(c(x) \to \exists y \, r(x, y)) \tag{28}$$

To state that the *range of a relation* r is the concept d, in FOL we can use the axioms

$$\forall x \forall y(r(x, y) \to d(y)) \tag{29}$$

$$\forall x(d(y) \to \exists x \, r(x, y)) \tag{30}$$

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

$$\forall x(c(x) \rightarrow \forall y(r(x,y) \rightarrow d(y))) \tag{31}$$

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

$$\forall x \left(c(x) \rightarrow \exists y_i \ldots \exists y_n \left(\bigwedge_{i=1}^{n} \left(r(x,y_i) \wedge d(y_i) \wedge \bigwedge_{j=1}^{i-1} x_i \neq x_j \right) \right) \right) \tag{32}$$

At most n

$$\forall x \left(c(x) \rightarrow \forall y_1 \ldots \forall y_{n+1} \left(\bigwedge_{i=1}^{n} (r(x,y_i) \wedge d(y_i)) \rightarrow \bigvee_{j=1}^{n} x_i = x_{n+1} \right) \right) \tag{33}$$

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

$$\forall x(\mathsf{electricCar}(x) \leftrightarrow \mathsf{car}(x) \wedge \forall y(\mathsf{hasEngine}(y) \rightarrow \mathsf{electricEngine}(y))$$

$$\forall x(\mathsf{hybridCar}(x) \leftrightarrow \mathsf{car}(x) \wedge \exists y(\mathsf{hasEngine}(y) \wedge \mathsf{electricEngine}(y)) \wedge \exists y(\mathsf{hasEngine}(y) \wedge \mathsf{carbonFuelEngine}(y)))$$

$$\forall x(\mathsf{carbonCar}(x) \leftrightarrow \mathsf{car}(x) \wedge \forall y(\mathsf{hasEngine}(y) \rightarrow \mathsf{carbonFuelEngine}(y))$$

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

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.

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.

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.

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.

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.

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.

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*.

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.

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.

Xiao, Guohui et al. (2018). "Ontology-based data access: A survey". In: International Joint Conferences on Artificial Intelligence.