Ontologies in Computer Science

Olivier Bellemans

University of Antwerp, Belgium
olivier.bellemans@student.uantwerpen.be

Abstract

The aim of this text is to give an introduction to ontologies and the means of expressing them. The formal foundations of ontology languages are emphasized, various applications in the field of computer science are discussed.

Keywords: Ontologies, Formal Logic, Description Logics, Semantic Web

1 Introduction

Ontologies are descriptions of the concepts and relationships in a particular domain. Such a description may take the form of informal prose, or it may be embedded in a more formal language with precise semantics. Formal logics are well suited for this task as they allow one to mechanically infer implicit knowledge based on what has already been stated about the domain.

We describe ontologies in more detail and provide some examples in Section 2. Section 3 briefly covers the syntax and semantics of predicate logic and description logics, and demonstrates how ontologies can be expressed as statements in these logics. We look at some standardized ontology languages and their applications in Section 4. Finally, we conclude in Section 5.

2 Ontologies

Ontologies capture knowledge about a domain. This knowledge may serve to share understanding among different groups, improve interoperabil-

ity of software systems [1], reason about properties of models [2] or any other goal which benefits from explicitly capturing knowledge.

To make the discussion more concrete we will focus on ontologies of one specific domain: family. A family has concepts such as `Person`, `Woman` or `Husband`. The instances of these concepts can be related to each other by relations such as `parent-of` or `sibling-of`. Furthermore, the concepts themselves may be related, we can say that every `Woman` is also a `Person`, for example.

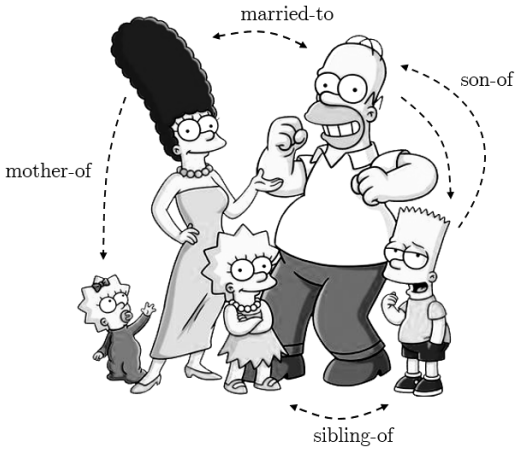

Figure 1: A family of individuals and how they are related. Ontologies can be used to capture general knowledge that is true of all families.

Most software systems contain ontologies. Class hierarchies contained in object-oriented computer programs are a form of ontology, as they describe the types and relationships of objects in a particular domain. Similarly, a database schema can be considered an ontology. However, the main purpose of a database schema is efficient storage and querying of data, not understanding. The formalisms we examine in Sections 3 and 4 have been designed with the explicit goal of understanding and reasoning.

3 Formal Logic

This section illustrates how formal logics can be used to express ontologies. The syntax and semantics are not covered in detail. For a full treatment of propositional logic and predicate logic see [3] and [4]. A good overview of description logics is given in [5].

3.1 Introduction

Consider following argument: 'Whenever it rains it is cloudy, it is not cloudy, therefore it does not rain'. It turns out that indeed it can not rain if the first two statements are true. This can be verified by translating the sentences to propositional logic, exposing their structure.

$$\begin{gathered} rain \rightarrow cloudy, \neg cloudy \models \neg rain \\ p \rightarrow q, \neg q \models \neg p \end{gathered} \tag{1}$$

Constructing a truth table reveals that indeed, the truth of the first two statements implies the truth of the third. Curiously, the fact we're talking about the weather is irrelevant, we can relabel the atomic propositions to construct an argument "blueprint" as in Eq. 1. This form of argument is best known as *modus tollens*, it can be applied in any context.

The ability to mechanically verify and derive statements based on a set of existing statements makes formal logics an attractive choice to express ontologies. A "knowledge base" can be built out of logic statements, describing things you know to be true about a domain. Any implicit knowledge can then be derived, for example the knowledge that any individual that is the parent of a parent is a grandparent.

3.2 Predicate Logic

Propositional logic is not expressive enough to capture the structure of many common statements. Sentences like 'All men are people' can only be encoded as atomic propositions. Predicate logic extends propositional logic with quantifiers, predicates and functions. Quantifiers allow one to express the fact that a property holds for all individuals or for at least one

individual. Unary and binary predicates can be used to model concepts and relationships, respectively. The universal quantifier $\forall$ combined with unary predicates lets us express subclass relations between concepts, as in Eq. 2, which states `Man` and `Woman` are subclasses of `Person`.

$$\begin{aligned} \forall x[Man(x) \rightarrow Person(x)] \\ \forall x[Woman(x) \rightarrow Person(x)] \end{aligned} \tag{2}$$

The existential quantifier $\exists$ can be combined in various ways to express cardinality constraints. The formula in Eq. 3 defines a "crazy cat lady" as a woman who owns two or more cats. Surely the threshold is higher than two, but you can imagine what would happen to our formula.

$$\begin{aligned} \forall x[CrazyCatLady(x) \leftrightarrow Woman(x) \wedge \\ \exists y[owns\text{-}cat(x,y) \wedge \exists z[z \neq y \wedge owns\text{-}cat(x,z)]]] \end{aligned} \tag{3}$$

These examples demonstrate that predicate logic is a rich formalism for expressing ontologies. However, the notation is verbose, concepts and relations need to be stated in a roundabout way. Furthermore, the validity of statements in predicate logic is undecidable. This is undesirable if we want to write computer programs to reason about our ontologies.

3.3 Description Logics

Description logics are a family of logics in which concepts and relations can be expressed in a more direct manner. The subclass statements of Eq. 2 can be rewritten as in Eq. 4.

$$\begin{aligned} Man \sqsubseteq Person \\ Woman \sqsubseteq Person \end{aligned} \tag{4}$$

In description logics concepts can be composed by using operators such as *conjunction* ($\sqcap$) and *disjunction* ($\sqcup$). This makes it straightforward to define new concepts in terms of others. Eq. 5 contains the same definition of "crazy cat lady" as in Eq. 3. Here, $\geq_2$ *owns-cat* describes a concept whose individuals all own two or more cats.

$$CrazyCatLady \equiv Woman \sqcap \geq_2 owns\text{-}cat \tag{5}$$

Aside from conjunction and disjunction, most description logics contain concept constructions such as *negation* ($\neg$), *existential restriction* ($\exists r.C$), and *value restriction* ($\forall r.C$). These last two can be used to define concepts whose instances are related to instances of another concept. For example, an individual belongs to $\exists parent\text{-}of.Child$ if he has at least one child. Those who belong to $\forall parent\text{-}of.Woman$ have only female children.

A collection of statements in a description logic is called a *knowledge base*. It consists of a terminology section called the *TBox* whose statements are about concepts, and an assertional part called the *ABox* which asserts facts about individuals. These statements may be true or false, depending on how we interpret the symbols of concepts, relations and individuals. An interpretation $\mathcal{I}$ in which all statements are true is called a *model* of a knowledge base $\mathcal{K}$, which we denote $\mathcal{I} \models \mathcal{K}$.

Interesting questions we can pose about a knowledge base is whether it is *consistent* (i.e. whether interpretations can exist that satisfy all statements) and whether all concepts are satisfiable. A knowledge base can also be queried. We could, for example, ask to enumerate all individuals that are instances of a given, possibly complex, concept.

4 Ontology Languages and Applications

4.1 OWL Web Ontology Language

We've seen that description logics are well suited for expressing ontologies. However, "bare" description logics were not designed for sharing knowledge between software systems. A standardized language that specifies a data format is needed to apply ontologies at large scale. One such language is RDF, the *Resource Description Framework*, a standard of the *World Wide Web Consortium* (W3C). Its specification includes an XML serialization format.

RDF is a fairly limited language that can not express, for example, cardinality constraints. The *OWL Web Ontology Language* is another W3C standard that is more expressive. It is based on a description logic called $\mathcal{SHOIN}(D)$, which is an almost-acronym that describes which features this

particular description logic posesses. The $\mathcal{S}$ stands for a set of "core" features which are included in many description logics. This includes conjunction, disjunction, complement and restricted forms of existential and universal quantification, which we saw in Section 3.3. The $\mathcal{H}$ stands for role **h**ierarchies (e.g. all those related by r are also related by s), $\mathcal{O}$ for n**o**minals (concept "literals"), $\mathcal{I}$ for **i**nverse roles (if a is a $parent\text{-}of\ b$, then b is a $child\text{-}of$ a), $\mathcal{N}$ for **n**umber restrictions (e.g. $\geq_2 owns\text{-}cat$). The D in parenthesis indicates that some concrete datatype has been integrated.

4.2 Semantic Web

A driving force behind the development of RDF and the OWL Web Ontology Language is the vision of the "Semantic Web". The Web today is essentially a graph of linked documents, consisting mainly of unstructured text. Although computer programs can examine the terms that occur in these documents, they do not have a deep understanding of its contents or its relationships. As a result, answering complex queries is difficult and typically can't be done in a fully automated fashion.

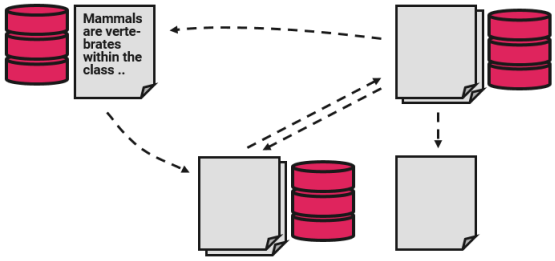

Figure 2: The web exposes and links documents of unstructed text. Documents are typically generated using data from a data source which stores the data in a more structured form, such as a relational database.

In many cases the data used to generate these documents is stored in a structured form, such as tables in a relational database. Given the right standards, we could expose this data and its associated structure to the outside world more directly, and link the data together just like we link documents together today. Ontology languages are a key component for making this vision a reality.

4.3 Model Driven Engineering

Another promising use of ontologies is in model driven engineering [2]. To determine whether a model of a railway system is an instance of the ontological type "Safe", for example, several different properties need to be satisfied. One of these is the property "CollisionFree", trains should not collide. Different orthogonal semantic mappings from the railway model onto models in suitable formalisms can be used. The models in the target formalisms should each satisfy relevant properties. Colored Petri Nets, for example, are appropriate to verify the "CollisionFree" property.

5 Conclusion

Ontologies describe the concepts and relationships of a particular domain. Formal logics are suitable hosts for expressing ontologies, as implicit knowledge can be mechanically derived. Furthermore, ontologies can be checked for consistency to make sure that no contradictions were introduced by mistake. Predicate logic can be used but suffers from verbose syntax and undecidability. Description logics are a family of logics which allow you to compose concepts in a more direct way.

Ontology languages such as the OWL Web Ontology Language are based on description logics and provide a standardized data format, allowing software systems to exchange ontologies. One of the major applications of ontology languages is the Semantic Web, which is an ambitious vision where the world's data is exposed in a structured format and linked together, as documents are linked together on the World Wide Web today.

References

[1] S. Grimm, A. Abecker, J. Völker, and R. Studer, Handbook of Semantic Web Technologies , 509 (2011), arXiv:1011.1669v3.

[2] B. Barroca, T. Kühne, and H. Vangheluwe, CEUR Workshop Proceedings **1237**, 77 (2014).

[3] S. S. Epp, *Discrete Mathematics with Applications*, 4th ed. (Brooks/Cole Publishing Co., Pacific Grove, CA, USA, 2010).

[4] M. Huth and M. Ryan, *Logic in Computer Science: Modelling and Reasoning About Systems* (Cambridge University Press, New York, NY, USA, 2004).

[5] F. van Harmelen, F. van Harmelen, V. Lifschitz, and B. Porter, *Handbook of Knowledge Representation* (Elsevier Science, San Diego, USA, 2007).

[6] G. Antoniou, E. Franconi, and F. Van Harmelen, Reasoning Web , 1 (2005).