

Description Logics and Time

Jie Zhang
School of Computer Science
University of Waterloo
j44zhang@cs.uwaterloo.ca

Abstract

In this survey paper, we begin by reviewing brief history of Description Logics developed from classical logics, semantic network and frames in demand of knowledge representation. We then introduce the logic language \mathcal{ALC} which is the basic form of description logic and some of its possible extensions. The usefulness of description logics for database will be briefly pointed out as well. Finally, we focus on the temporal extensions of \mathcal{ALC} , \mathcal{ALCT} . The dimension of point-based temporal logics and interval-based temporal logics will be introduced, respectively. And, the decidability and complexity of concept satisfiability problem in \mathcal{ALCT} will be briefly discussed.

1 Brief History

Knowledge representation plays an important role in knowledge-based systems [9], such as rule-based systems, case-based reasoning systems, and etc. A rule based system uses “rules” as the knowledge representation for knowledge coded into the system. In case-based reasoning systems, expertise is embodied in a library of past cases, rather than being encoded in classical rules. Each case typically contains a description of the problem, plus a solution and/or the outcome. The knowledge and reasoning process used by an expert to solve the problem is not recorded, but is implicit in the solution. In brief, the tasks of knowledge representation include representing knowledge base of an application domain, (a.k.a the “world”) and reasoning over it to infer implicitly represented knowledge from the knowledge that is explicitly contained in the knowledge base [5].

Two classes of approaches to knowledge representation having been developed include logic-based formalisms and non-logic-based representations [10]. The logic-based formalisms are mainly based on first-order logic, where knowledge

is represented by first-order logic predicate calculus and reasoning amounts to verifying logical consequence. Note the reason of not using propositional logic is that in propositional logic atomic sentences have no internal structure, and therefore we cannot access the structure of atomic sentences and . Review of classic logics is not a focus in this paper, and related readings can be found in [8]. The non-logic-based approaches represent knowledge based on the use of graphical interfaces. Two typical representations are semantic networks and frames, which are all called network-based structures. A simple example of network-based structure shown in Figure 1 represents knowledge concerning persons, employees, students and professors. Nodes in the graph represent concepts (for example, “Person” is a concept) and arcs represent relationships between concepts (for example, a “IS-A” relationship between students and persons indicates that a student is a person).

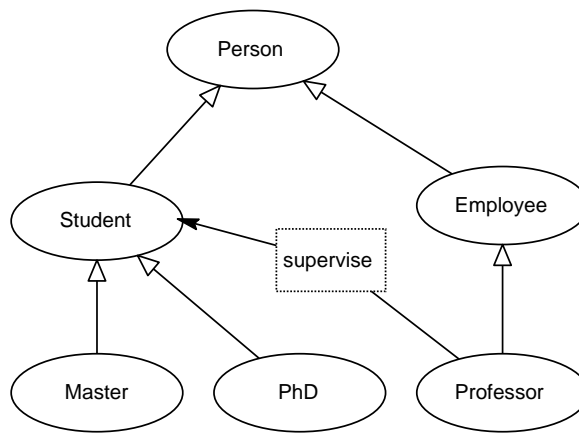


Figure 1: A Simple Example of Network

The network-based structures to represent knowledge is very straight forward, effective and more appealing. The problem of this representation is that they lack precise semantic characterization. Specifically, arcs can represent different kinds of relations between nodes [7]. For example, instead of a “IS-A” relationship, in Figure 1 the arc between professors and students indicate that professors supervise students. The consequence of this problem is that systems developed using network-based structures behave differently. Another problem is that reasoning on it is usually inefficient. The problems have been addressed by giving a semantics to frames based on first-order logic. However, first-order logic does not capture the constraints of semantic networks and frames. The structure of the knowledge is therefore lost. The expressive power of first-order logic is too high for having good computational properties and efficient procedures. Semantic networks and

frames do not require all the machinery of first-order logic. Another problem of directly using first-order logic is that the inference power of it may be too low for expressing interesting, but still decidable theories. This realization brings the term of “Description Logics”, which is a fragment of first-order logic. In the next section, we are going to look at very basic forms of “Description Logics”, and its syntax, semantics and related reasoning issues. We will also give examples of how to transform a network into “Description Logics”, and mention briefly its relationship with database development. In addition, we will discuss its possible extensions and focus specially on its temporal extensions.

2 Description Logics

The very basic types of a concept language are concepts, roles, and individuals (concept names) [4]. A concept is a description of a collection of individuals with common properties, for example, “Student” in Figure 1 is a concept. Inter-relationships between these individuals are represented by roles. For example, “supervise” represents relationships between individuals in “Student” and “Professor”. Individuals denote constants of concepts, for instance, “Jack” is a constant of “Student”.

The very basic description language is the \mathcal{FL}^- language introduced by Brachman and Levesque in 1984. This language contains concept conjunction, universal role quantification and limited existential quantification. After that, more powerful language are then defined by adding other constructors to this basic language [7]. A minimal language that is of practical interest is the language \mathcal{AL} introduced by Schmidt-Schauß and Smolka in 1991 [10]. The \mathcal{AL} language adds negation constructor on the \mathcal{FL}^- language. The \mathcal{FL} language disallows the limited existential quantification. The \mathcal{AL} language has been extended to form a family of \mathcal{AL} languages. For instance, \mathcal{ALC} extends \mathcal{AL} by including full negation and disjunction. The \mathcal{ALCF} language extends \mathcal{ALC} with functions [3]. The \mathcal{ALEN} language is the extension of \mathcal{AL} by full existential quantification and number restrictions [10]. For the purpose of simplicity, we focus only on the \mathcal{ALC} language.

2.1 Syntax

The grammar for the \mathcal{ALC} language is as follows:

$$C, D \rightarrow A | C \sqcap D | C \sqcup D | \neg C | \forall R.C | \exists R.C, \quad (1)$$

where A is an atomic concept, R is an atomic role, and C and D are concepts. Here are some examples of the \mathcal{ALC} language to transform the semantic network

presented in Figure 1. $Person \sqcap Student$ represents those persons who are also students. $Person \sqcap \forall supervise.PhD$ denotes those persons all of whom supervise PhD students. $Person \sqcap \exists supervise.Student$ denotes those persons at least one of whom supervises students.

2.2 Semantics

Intuitively, concepts represent classes, sets of individuals, roles represent relationships between pairs of individuals, and atomic concepts are the names of primitive concepts. “ \sqcap ” represents conjunctions of concepts. Semantics of the \mathcal{ALC} language can be formally defined in Table 1. This table presents formal (extensional) semantics of \mathcal{ALC} and FOL semantics of \mathcal{ALC} . In this table, $\Delta^{\mathcal{I}}$ represents a nonempty set (domain). The function $\cdot^{\mathcal{I}}$ is an interpretation function that maps every concept to a subset of $\Delta^{\mathcal{I}}$, every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and every individual to an element of $\Delta^{\mathcal{I}}$. To map syntax to FOL semantics, a concept C and a role R correspond to the FOL open formulae $F_C(x)$ and $F_R(x, y)$, respectively [4].

Table 1: Semantics of \mathcal{ALC}

Syntax	Formal Semantics	FOL semantics
A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	$F_A(x)$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	$F_C(x) \wedge F_D(x)$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	$F_C(x) \vee F_D(x)$
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	$\neg F_C(x)$
$\forall R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \forall b.(a, b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$	$\forall z.F_R(x, z) \rightarrow F_C(z)$
$\exists R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b.(a, b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$	$\exists z.F_R(x, z) \wedge F_C(z)$

2.3 Reasoning Services

One feature of Description Logics is reasoning as a central service. A variety of reasoning services are provided, including concept satisfiability, subsumption, satisfiability, instance checking, retrieval, and realization. Checking satisfiability of concepts is a key inference. A concept C is satisfiable with respect to \mathcal{T} if there exists a model \mathcal{I} of \mathcal{T} such that $C^{\mathcal{I}} \neq \emptyset$ [4]. The reasoning of subsumption tells us whether some concept is more general than other ones. A concept C is subsumed by a concept D with respect to \mathcal{T} if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model \mathcal{I} of \mathcal{T} . Note that concept satisfaction is a special case of subsumption, with the subsumer being the empty concept, meaning that a concept is not satisfiable.

On one hand, we define a description language that is expressive. On another hand, the reasoning problems of the defined language should be decidable, which

means that there exists an algorithm of solving the problems in a finite number of steps [1]. And, design of an effective and efficient algorithm for reasoning is a challenge. An algorithm should be sound and complete. As firstly argued by Brachman and Levesque that there is a trade-off between the expressivity of a representation language and the difficulty of reasoning on the representations built using that language [7]. In other words, the more expressive the language, the harder the reasoning. However, the efficiency of reasoning in Description Logics is measured by using worst-case complexity. This brings more careful study of computational complexity of reasoning in Description Logics, for example, the analysis of a general technique relying on a form of tableau calculus.

2.4 DL for Database Management

Here, we will mention little bit about Description Logics used for database management. Description Logics and database are strongly related. First, many knowledge-based systems contain both DataBase Management Systems (DBMS) and Description Logics Knowledge Representation System (DL-KRS). Second, Description Logics can be used to address many issues in database management. It can be used to check concept consistency to verify at design time whether an entity can have at least one instance. Description Logics can be used for database when there is a need of query optimization and/or information integration.

2.5 Temporal Extensions

The directions of extending Description Logics include adding to DL languages some representational features that were common in frame systems or that are relevant for certain classes of applications, and adding reasoning services that are useful in the development of knowledge bases. One of the possible extensions is temporal extension. Temporal extensions of Description Logics have been treated as a special kind of modal extension. The first proposal for handling time in a DL framework was originated in the context of the DL system Back. Later, following the standard approaches in the representation of time, both interval-based and point-based approaches have been studied, specifically focusing on the decidability and complexity of the reasoning problems [2]. In the next section, we will focus on temporal extensions of the \mathcal{ALC} language.

3 Temporal Description Logics

There are a variety of ways for temporal extension of Description Logics. They are differ in three dimensions [2]. The first dimension is that they are different on the

notion of time. Specifically, Description Logics are extended with a point-based notion of time or interval-based notion of time. They also differ on the way of adding the temporal dimension, either implicitly (a temporal information is only implicit in the language) or explicitly (an explicit notion of time is adopted). In the case of an explicit representation of time, there is a further distinction between an external and an internal point of view.

In this section we focus on only two extensions of Description Logics with a point-based notion of time and interval-based notion of time.

3.1 Point-based Temporal Description Logics

Schild first combines the Description Logic \mathcal{ALC} with a point-based notion of time [3]. The new language is called \mathcal{ALCT} . The syntax of \mathcal{ALC} is presented in Section 2.1. \mathcal{ALCT} is the extension of \mathcal{ALC} with the temporal operators \mathcal{U} (Until) and \mathcal{S} (Since) and other temporal connectives, sometime in the future, \diamond , always in the future, \square , sometime in the past, \blacklozenge , and always in the past, \blacksquare [2, 6]. The following rules then need to be added to the syntax presented in Equation 1:

$$\begin{aligned}
C, D \rightarrow \quad & C\mathcal{U}D \mid (C \text{ until } D) \text{ (until)} \\
& C\mathcal{S}D \mid (C \text{ since } D) \text{ (since)} \\
& \diamond C \mid \text{sometime in the future} \\
& \square C \mid \text{always in the future} \\
& \blacklozenge C \mid \text{sometime in the past} \\
& \blacksquare C \mid \text{always in the past}
\end{aligned}$$

Note that the connective operators can be defined in terms of Since and Until temporal operators as follows [6]:

$$\begin{aligned}
\diamond C &\doteq \top \mathcal{U} C \\
\blacklozenge C &\doteq \top \mathcal{S} C \\
\square C &\doteq \neg \diamond \neg C \\
\blacksquare C &\doteq \neg \blacklozenge \neg C
\end{aligned}$$

We define a temporal structure $\mathcal{T} = (\mathcal{P}, <)$, where \mathcal{P} is a set of time points and $<$ is a strict linear order on \mathcal{P} . An \mathcal{ALCT} temporal interpretation over \mathcal{T} is a pair $\mathcal{M} \doteq \langle \mathcal{T}, \mathcal{I} \rangle$, where \mathcal{I} is a function associating to each $t \in \mathcal{P}$ a standard non-temporal \mathcal{ALC} interpretation, $\mathcal{I}(t) \doteq \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}(t)} \rangle$. The semantics of \mathcal{ALCT} is then extended by adding two equations to the interpretation function \mathcal{I} satisfying the standard semantic definitions of \mathcal{ALC} defined in Table 1 for each t in \mathcal{T} :

$$(CUD)^{I(t)} = \{x \in \Delta^I \mid \exists v.(v > t) \wedge D^{I(v)}(x) \wedge \forall w.(t < w < v) \rightarrow C^{I(w)}(x)\}$$

$$(CSD)^{I(t)} = \{x \in \Delta^I \mid \exists v.(v > t) \wedge D^{I(v)}(x) \wedge \forall w.(v < w < t) \rightarrow C^{I(w)}(x)\}$$

Consider the examples based on the semantic network presented in Figure 1, a PhD student can be defined as a person who is current a student and will be a professor in the future. This definition can be expressed by using the temporal operators as follows:

$$PhD \doteq Person \sqcap (Student \mathbf{U} \blacksquare Professor) \quad (2)$$

3.2 Interval-based Temporal Description Logics

Schmiedel was the first to propose an extension of Description Logics with an interval-based notion of time [3]. The extension is based on the Allen's interval relationships [4] presented in Figure 2.

Relation	Abbr.	Inverse	<i>i</i>	<i>j</i>
before(<i>i</i> , <i>j</i>)	b	a		
meets(<i>i</i> , <i>j</i>)	m	mi		
overlaps(<i>i</i> , <i>j</i>)	o	oi		
starts(<i>i</i> , <i>j</i>)	s	si		
during(<i>i</i> , <i>j</i>)	d	di		
finishes(<i>i</i> , <i>j</i>)	f	fi		

Figure 2: The Allen's Interval Relationships

We have defined a linear and unbounded temporal structure \mathcal{T} in the last section. Then, the interval set of \mathcal{T} , \mathcal{T}^* can be defined as a set of all closed proper intervals $[t_1, t_2] \doteq \{t \in \mathcal{P} \mid t_1 \leq t \leq t_2\}$. An \mathcal{ALCT} temporal interpretation over \mathcal{T}^* is a pair $\mathcal{M} \doteq \langle \mathcal{T}, \mathcal{I} \rangle$, where \mathcal{I} is a function associating to each $i = [t_1, t_2] \in \mathcal{T}^*$ a standard non-temporal \mathcal{ALC} interpretation. The semantics of \mathcal{ALCT} is then extended by adding two equations to the interpretation function \mathcal{I} satisfying the standard semantic definitions of \mathcal{ALC} defined in Table 1 for each t in \mathcal{T} [4]:

$$(\langle A \rangle C)^{I(i)} = \{x \in \Delta^I \mid \exists j. \alpha(j, i) \wedge C^{I(j)}(x)\}$$

$$([A]C)^{I(i)} = \{x \in \Delta^I \mid \forall j. \alpha(j, i) \rightarrow C^{I(j)}(x)\}$$

where α is one of the relations defined in Figure 2.

The example of “PhD Student” can be redefined as a person who is student at the time interval and will be a professor at some other time interval met by the previous time interval. Therefore the Equation 2 can be rewritten as follows:

$$PhD \doteq Person \sqcap (Student \sqcap \langle mi \rangle Professor)$$

3.3 Decidability and Complexity

In this section, we briefly mention the decidability and complexity problems of concept satisfiability. As defined earlier in Section 2.3, a concept C is satisfiable if there exists \mathcal{M} and t such that $\mathcal{M}, t \models \neg(C \doteq \perp)$, which means that there exists an interpretation \mathcal{I} such that $C^{I(t)} \neq \emptyset$ for some t [4]. A problem is decidable if there exists a computational process that solves the problem in a finite number of steps [1]. As can be seen from Sections 3.1 and 3.2, we did not define temporal roles for \mathcal{ALCT} . It is because that \mathcal{ALCT} with global roles is undecidable in any unbounded linear order [4]. The concept satisfiability in \mathcal{ALCT} without global roles is *PSPACE – complete* [4].

4 Conclusions

In this survey paper, we first briefly introduce the history of Description Logics motivated by expressiveness and reasonability in demand of knowledge representation. Its temporal extensions are then surveyed in terms of the dimension of point-based temporal logics and interval-based temporal logics.

References

- [1] Enrico Franconi’s lecture notes: Foundations of first order logic. <http://www.inf.unibz.it/franconi/dl/course/>.
- [2] Alessandro Artale and Enrico Franconi. Introducing temporal description logics. In *TIME*, pages 2–5, 1999.

- [3] Alessandro Artale and Enrico Franconi. A survey of temporal extensions of description logics. *Annals of Mathematics and Artificial Intelligence*, 30(1-4):171–210, 2000.
- [4] Alessandro Artale and Enrico Franconi. Temporal description logics. In Dov Gabbay, Michael Fisher, and Lluís Vila, editors, *Handbook of Time and Temporal Reasoning in Artificial Intelligence*. MIT Press, 2000.
- [5] F. Baader and W. Nutt. Basic description logics. In F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors, *Description Logic Handbook*, pages 47–100. Cambridge University Press, 2002.
- [6] Jan Chomicki and David Toman. Temporal logic in information systems. In Jan Chomicki and Gunter Saake, editors, *Logics for Databases and Information Systems*, pages 31–70. Kluwer Academic publishers, 1998.
- [7] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Reasoning in description logics. In Gerhard Brewka, editor, *Principles of Knowledge Representation*, pages 191–236. CSLI Publications, Stanford, California, 1996.
- [8] J. Kelly. *The Essence of Logic*. Prentice Hall, 1997.
- [9] H. J. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence journal*, 3:78–93, 1987.
- [10] D. Nardi and R. J. Brachman. An introduction to description logics. In F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors, *Description Logic Handbook*, pages 5–44. Cambridge University Press, 2002.