Description logics, ontologies, and automated reasoning: an introduction

Logic Colloqium 2018, Udine, Italy, July 2018, Day 1

Relation with other logics, ontologies, examples and exercises

Uli Sattler

Previous and Next Steps

- So far: syntax, semantics, and basics of the DL $\mathcal{ALC}$:
  - **syntax**: $\mathcal{ALC}$ concepts, axioms, assertions, TBox, ABox, ontology
  - **semantics**: interpretations, models
  - **reasoning problems**: entailment, satisfiability, consistency,
    ...and relationships between reasoning problems
- Next: relationships between
  - Description Logics
  - First Order Logic
  - Modal Logic
  - OWL — so that we can use Protégé 5 for exercises

Where Description Logics come from

Description Logics were

- developed as logical formalisation of semantic networks in the late 80s
- discovered to have close relationships with FOL, ML in the early 90s
- investigated widely in the last 25+ years:
  - trade-off between expressive power and computational complexity of reasoning
  - model theory
  - ...
- used as the logical basis of the Web Ontology Language, OWL

Relationship with First Order Logic

The following is not hard to see:
if we view concept names A as unary predicates and roles r as binary predicates, then

FOL:

- each interpretation $\mathcal{I}$ can be seen as a FOL structure
- each $\mathcal{ALC}$ concept C can be translated into a FOL formula $t_x(C)(x)$ (in which x is a free variable) such that

$$e \in C^{\mathcal{I}} \text{ iff } \mathcal{I} \models t_x(C)[x/e]$$

Relationship with First Order Logic II

Here is the translation $t_x()$ from $\mathcal{ALC}$ concepts into FOL formulae in one free variable

$$
\begin{aligned}
t_x(A) &= A(x), & t_y(A) &= A(y),\\
t_x(\neg C) &= \neg t_x(C), & t_y(\neg C) &= \ldots,\\
t_x(C \sqcap D) &= t_x(C) \wedge t_x(D), & t_y(C \sqcap D) &= \ldots,\\
t_x(C \sqcup D) &= \ldots, & t_y(C \sqcup D) &= \ldots,\\
t_x(\exists r.C) &= \exists y.r(x,y) \wedge t_y(C), & t_y(\exists r.C) &= \ldots,\\
t_x(\forall r.C) &= \ldots, & t_y(\forall r.C) &= \ldots.
\end{aligned}
$$

- Fill in the blanks
- Why are $t_x(C)$, $t_y(C)$ formulas in one free variable?

Relationship with First Order Logic III

Here is the translation $t_x()$ from $\mathcal{ALC}$ concepts into FOL formulae in one free variable

$$
\begin{aligned}
t_x(A) &= A(x), & t_y(A) &= A(y),\\
t_x(\neg C) &= \neg t_x(C), & t_y(\neg C) &= \neg t_y(C),\\
t_x(C \sqcap D) &= t_x(C) \wedge t_x(D), & t_y(C \sqcap D) &= t_y(C) \wedge t_y(D,\\
t_x(C \sqcup D) &= t_x(C) \vee t_x(D), & t_y(C \sqcup D) &= t_y(C) \vee t_y(D),\\
t_x(\exists r.C) &= \exists y.r(x,y) \wedge t_y(C), & t_y(\exists r.C) &= \exists x.r(y,x) \wedge t_x(C),\\
t_x(\forall r.C) &= \forall y.r(x,y) \Rightarrow t_y(C), & t_y(\forall r.C) &= \forall x.r(y,x) \Rightarrow t_x(C)
\end{aligned}
$$

- ...
- Why are $t_x(C)$, $t_y(C)$ formulas in one free variable?

Relationship with First Order Logic – Ontologies

Translate an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ using $t()$ as follows:

$$
\begin{aligned}
t(\mathcal{O}) &= t(\mathcal{T}) \cup t(\mathcal{A}) \\
t(\mathcal{T}) &= \{\forall x.(t_x(\neg C) \vee t_x(D)) \mid C \sqsubseteq D \in \mathcal{T}\} \\
t(\mathcal{A}) &= \{t_x(C)[x/a] \mid a\colon C \in \mathcal{A}\} \cup \\
&\quad\ \{r(a,b) \mid (a,b)\colon r \in \mathcal{A}\}
\end{aligned}
$$

As a consequence, we have that

Theorem

1. e is an instance of C in $\mathcal{I}$ iff $\mathcal{I} \models t_x(C)[x/e]$
2. C is satisfiable iff $t_x(C)$ is satisfiable
3. C is satisfiable w.r.t. $\mathcal{O}$ iff $\{t_x(C)[x/e]\} \cup t(O)$ is satisfiable
4. C is subsumed by D iff $\forall x.t_x(C) \Rightarrow t_x(D)$ is valid
5. $\mathcal{O} \models C \sqsubseteq D$ iff $t(\mathcal{O}) \models \forall x.t_x(C) \Rightarrow t_x(D)$

Relationship with First Order Logic (ctd)

Observations:

- $t_x(C)$ only uses two variables

  $\Rightarrow$ $\mathcal{ALC}$ is a fragment of the **2-variable fragment of FOL** known to be decidable

- $t_x(C)$ only uses **guarded quantification**

  $\Rightarrow$ $\mathcal{ALC}$ is a fragment of the **guarded fragment of FOL** known to be decidable

Relationship with Modal Logic

Easy if only 1 role used, e.g.:

$(DL)\ \ A \sqcap \exists r.(A \sqcap B)$	$(ML)\ \ A \wedge \Diamond(A \wedge B)$
$(DL)\ \ A \sqcap \forall r.(A \sqcap B)$	$(ML)\ \ A \wedge \Box(A \wedge B)$
$(DL)\ \ A \sqcap \exists r.A \sqcap \forall r.B$	$(ML)\ \ A \wedge \Diamond A \wedge \Box B$
$(DL)\ \ A \sqcap \exists r.A \sqcap \forall r.\neg A$	$(ML)\ \ A \wedge \Diamond A \wedge \Box \neg A$

Need to switch to **Multi Modal Logic** for the general case, e.g.,:

$(DL)\ \ A \sqcap \exists r.A \sqcap \forall s.(\neg A \sqcap \exists t.B)$ $\quad$ $(ML)\ \ A \wedge \langle r \rangle A \wedge [s](\neg A \wedge \langle t \rangle B)$

I.e., extend syntax to parametrised boxes & diamonds, and
semantics to several accessibility relations R_s, e.g.,
$\mathcal{M}, w \models [s]\phi$ if, for every $v \in W$, $(w, v) \in R_s$ implies $\mathcal{M}, s \models \phi$

Relationship with Modal Logic: ontologies

In Modal Logic, we are mainly concerned with a single formula.

There is no equivalent to TBoxes or ABoxes, but (for $\tilde{C}$ the ML version of C):

TBox: if we have a **universal modality** u, we can translate

$$C \sqsubseteq D \quad \text{into} \quad [u](\neg\tilde{C} \vee \tilde{D})$$

ABox: if we have **nominals**, we can translate

$$a : C \quad \text{into} \quad @_a(\tilde{C})$$
$$(a, b) : r \quad \text{into} \quad @_a\langle r\rangle b$$

Relationship with Modal Logic – Harvest

We can use the

- Modal Logic algorithms (MLAs) to **decide**
  satisfiability of and subsumption between $\mathcal{ALC}$ concepts.
- soundness & completeness proof of MLA to show that $\mathcal{ALC}$ has FMP:

  C is satisfiable iff C is satisfiable in a finite interpretation.[1]

  soundness & completeness proof of MLA to show that $\mathcal{ALC}$ has TMP:

  C is satisfiable iff C is satisfiable in a tree interpretation.[2]

  soundness & completeness proof of MLA to show that $\mathcal{ALC}$ has FTMP:

  C is satisfiable iff C is satisfiable in a finite tree interpretation.[3]
- complexity results on ML to learn that

  satisfiability of $\mathcal{ALC}$ concepts is PSpace-complete.

[1] A finite interpretation is one with a finite domain.
[2] A tree interpretation is one whose domain has a tree structure.
[3] A finite tree interpretation is one that is finite and tree-shaped.

OWL and DLs

OWL:

- is the Web Ontology Language, now OWL 2 – but we use 'OWL'
- starting apoint: `www.w3.org/TR/owl2-overview/`
- has various syntaxes, e.g., RDF/XML, OWL/XML, and Manchester Syntax
- comes with import mechanisms, annotations, etc.
- **logical underpinning** through DLs:
  - an OWL ontology corresponds to a $\mathcal{SROIQ}(\mathcal{D})$ ontology
  - where $\mathcal{SROIQ}(\mathcal{D})$ is an extension of $\mathcal{ALC}$ with inverse roles, cardinality restrictions, transitive roles, ...
  - some OWL ontologies corresponds to an $\mathcal{ALC}$ ontology
  - we can express an $\mathcal{ALC}$ ontology in OWL
- ontology IDEs e.g., Protégé 5 help us to edit $\mathcal{O}$ and interact with reasoner
  - download Protégé 5: `owlapi.sourceforge.net/`
  - write your (first) $\mathcal{ALC}$ or OWL ontology

OWL and DLs (ctd)

To write an $\mathcal{ALC}$ or OWL ontology, you can use

- pen and paper
- a text editor and a typesetting system such as LaTex
- a "logic" IDE: e.g., Protégé 5

  In **Reasoner** menu, on choosing **Start Reasoner** for $\mathcal{O}$, the chosen reasoner

  – tests the ontology for **consistency** and **coherence**

  – for each pair of A, B of concept/classe **names**, determines whether $\mathcal{O} \models A \sqsubseteq B$ or $\mathcal{O} \models B \sqsubseteq A$

  ...and displays the results $\Rightarrow$ let's see how this works...

Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The Making of a Web Ontology Language. J. of Web Semantics, 1(1):7-26, 2003.

Suggested Exercise

So, for tomorrow, you are cordially invited to

- pick a domain of your choice and expertise (football, fashion, food, fish, ...)
- design your first ontology, in $\mathcal{ALC}$, with
  - TBox, to introduce/define relevant concepts and roles
  - ABox, to populate your TBox
  - say 20 concepts/role names, 8 individuals
- ideally in OWL, via Protégé 5, so that you can make use of a reasoner (they come with Protégé 5)

Links:

- for Protégé 5, go to `owlapi.sourceforge.net/`
- for OWL from a logics perspective, have a look at `http://owl.cs.manchester.ac.uk/about/orientation/a-logics-perspective/`