

**An Introduction to Description Logics:  
Techniques, Properties, and Applications**

**NASSLLI, Day 2, Part 1**

**Relation with other logics, ontologies, examples and exercises**

**Uli Sattler**

## Previous and Next Steps

- So far: syntax, semantics, and basics of the DL *ALC*:
  - where they come from
  - **syntax**: *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
  - Modal Logic
  - First Order 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

$$t_x(A) = A(x),$$

$$t_y(A) = A(y),$$

$$t_x(\neg C) = \neg t_x(C),$$

$$t_y(\neg C) = \dots,$$

$$t_x(C \sqcap D) = t_x(C) \wedge t_x(D), \quad t_y(C \sqcap D) = \dots,$$

$$t_x(C \sqcup D) = \dots, \quad t_y(C \sqcup D) = \dots,$$

$$t_x(\exists r.C) = \exists y.r(x, y) \wedge t_y(C), \quad t_y(\exists r.C) = \dots,$$

$$t_x(\forall r.C) = \dots, \quad t_y(\forall r.C) = \dots$$

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

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

$$t(\mathcal{O}) = t(\mathcal{T}) \cup t(\mathcal{A})$$

$$t(\mathcal{T}) = \{\forall x.t_x(\neg C) \vee t_x(D) \mid C \sqsubseteq D \in \mathcal{T}\}$$

$$t(\mathcal{A}) = \{t_x(C)[x/a] \mid a : C \in \mathcal{A}\} \cup \\ \{r(a, b) \mid (a, b) : r \in \mathcal{A}\}$$

As a consequence, we have that

- Theorem 1**
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  $\{t_x(C)[x/e]\} \cup t(\mathcal{O})$  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  
⇒  $\mathcal{ALC}$  is a fragment of the 2-variable fragment of FOL known to be decidable
- $t_x(C)$  only uses guarded quantification  
⇒  $\mathcal{ALC}$  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.:

$(DL) A \sqcap \exists r.(A \sqcap B)$	$(ML) A \wedge \diamond(A \wedge B)$
$(DL) A \sqcap \forall r.(A \sqcap B)$	$(ML) A \wedge \square(A \wedge B)$
$(DL) A \sqcap \exists r.A \sqcap \forall r.B$	$(ML) A \wedge \diamond A \wedge \square B$
$(DL) A \sqcap \exists r.A \sqcap \forall r.\neg A$	$(ML) A \wedge \diamond A \wedge \square \neg A$

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  $\mathcal{M}, v \models \phi$



## 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

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

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

$$\begin{aligned} a : C & \text{ into } @_a(\tilde{C}) \\ (a, b) : r & \text{ into } @_a\langle r \rangle b \end{aligned}$$

A little exercise: take the following  $\mathcal{ALC}$  concept  $C$ :

$$A \sqcap \exists r.(A \sqcap \exists s.B \sqcap \exists s.C) \sqcap \\ \exists r.B \sqcap \\ \forall r.(\exists s.A \sqcap \forall s.C)$$

- translate  $C$  into a modal logic formula  $\phi$
- translate  $C$  into a first order logic formula  $\phi'$

We can use the

- Modal Logic algorithms (MLAs) to **decide** satisfiability of and subsumption between  $\mathcal{ALC}$  concepts.
- soundness & completeness proof of the MLA to show that  $\mathcal{ALC}$  has FMP:  
 $C$  is satisfiable iff  $C$  is satisfiable in a finite interpretation.<sup>1</sup>
- soundness & completeness proof of the MLA to show that  $\mathcal{ALC}$  has TMP:  
 $C$  is satisfiable iff  $C$  is satisfiable in a tree interpretation.<sup>2</sup>
- soundness & completeness proof of the MLA to show that  $\mathcal{ALC}$  has FTMP:  
 $C$  is satisfiable iff  $C$  is satisfiable in a finite tree interpretation.<sup>3</sup>
- complexity results on ML to learn that satisfiability of  $\mathcal{ALC}$  concepts is PSpace-complete.

<sup>1</sup>A finite interpretation is one with a finite domain.

<sup>2</sup>A tree interpretation is one whose domain has a tree structure.

<sup>3</sup>A finite tree interpretation is one that is finite and tree-shaped.

### OWL:

- is the Web Ontology Language, now OWL 2 – but we use 'OWL'
- starting point: [www.w3.org/TR/owl2-overview/](http://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  $SR\mathcal{OIQ}(\mathcal{D})$  ontology
  - where  $SR\mathcal{OIQ}(\mathcal{D})$  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 such as Protégé 5 help us to edit these and interact with reasoner
  - download Protégé 5: [owlapi.sourceforge.net/](http://owlapi.sourceforge.net/)
  - write your (first) OWL ontology

## OWL and DLs – a snapshot

- concept in DL – class in OWL
- role in DL – property in OWL

Abstract Syntax	DL Syntax	Semantics
Descriptions ( $C$ )		
$A$ (URI reference)	$A$	$A^I \subseteq \Delta^I$
<code>owl:Thing</code>	$\top$	$\text{owl:Thing}^I = \Delta^I$
<code>owl:Nothing</code>	$\perp$	$\text{owl:Nothing}^I = \{\}$
<code>intersectionOf(<math>C_1 C_2 \dots</math>)</code>	$C_1 \sqcap C_2$	$(C_1 \sqcap C_2)^I = C_1^I \cap C_2^I$
<code>unionOf(<math>C_1 C_2 \dots</math>)</code>	$C_1 \sqcup C_2$	$(C_1 \sqcup C_2)^I = C_1^I \cup C_2^I$
<code>complementOf(<math>C</math>)</code>	$\neg C$	$(\neg C)^I = \Delta^I \setminus C^I$
<code>oneOf(<math>o_1 \dots</math>)</code>	$\{o_1, \dots\}$	$\{o_1, \dots\}^I = \{o_1^I, \dots\}$
<code>restriction(<math>R</math> someValuesFrom(<math>C</math>))</code>	$\exists R.C$	$(\exists R.C)^I = \{x \mid \exists y. (x, y) \in R^I \text{ and } y \in C^I\}$
<code>restriction(<math>R</math> allValuesFrom(<math>C</math>))</code>	$\forall R.C$	$(\forall R.C)^I = \{x \mid \forall y. (x, y) \in R^I \rightarrow y \in C^I\}$
<code>restriction(<math>R</math> hasValue(<math>o</math>))</code>	$R : o$	$(R : o)^I = \{x \mid (x, o^I) \in R^I\}$

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.

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 **Classify Ontology** 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

$$\mathcal{O} \models A \sqsubseteq B \text{ or } \mathcal{O} \models B \sqsubseteq A$$

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

## Homework

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 *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/](http://owlapi.sourceforge.net/)
- for OWL from a logics perspective, have a look at <http://owl.cs.manchester.ac.uk/about/orientation/a-logics-perspective/>