Description Logics

Giuseppe De Giacomo

Dipartimento di Informatica e Sistemistica

Università di Roma "La Sapienza"

Seminari di Ingegneria del Software: Integrazione di Dati e Servizi

Parte 4 - Ontologie

A.A. 2005/06

What are Description Logics?

In modeling an application domain we typically need to represent a situation in terms of

- objects
- classes
- relations (or associations)

and to reason about the representation

Description Logics are logics specifically designed to represent and reason on

- objects
- classes – called concepts in DLs
- (binary) relations – called roles in DLs

Origins of Description Logics

Knowledge Representation is a subfield of Artificial Intelligence

Early days KR formalisms (late '70s, early '80s):

- Semantic Networks: graph-based formalism, used to represent the meaning of sentences
- Frame Systems: frames used to represent prototypical situations, antecedents of object-oriented formalisms

Problems: no clear semantics, reasoning not well understood

Description Logics (a.k.a. Concept Languages, Terminological Languages) developed starting in the mid '80s, with the aim of providing semantics and inference techniques to knowledge representation systems

Current applications of DLs

DLs have evolved from being used "just" in KR

Found applications in:

- Databases:
  - schema design, schema evolution
  - query optimization
  - integration of heterogeneous data sources, data warehousing
- Conceptual modeling
- Ontologies
- ...

Ingredients of a DL

A Description Logic is characterized by:

1. A description language: how to form concepts and roles

   $\textbf{Human} \sqcap \textbf{Male} \sqcap (\exists \textbf{child}) \sqcap \forall \textbf{child}.(\textbf{Doctor} \sqcup \textbf{Lawyer})$

2. A mechanism to specify knowledge about concepts and roles (i.e., a TBox)

   gg$\mathcal{K} \;=\; \{\; \textbf{Father} \;\equiv\; \textbf{Human} \sqcap \textbf{Male} \sqcap (\exists \textbf{child}),$

   $\qquad \textbf{HappyFather} \;\sqsubseteq\; \textbf{Father} \sqcap \forall \textbf{child}.(\textbf{Doctor} \sqcup \textbf{Lawyer}) \;\}$

3. A mechanism to specify properties of objects (i.e., an ABox)

   $\mathcal{A} \;=\; \{\; \textbf{HappyFather}(\mathsf{JOHN}), \quad \textbf{child}(\mathsf{JOHN}, \mathsf{MARY}) \;\}$

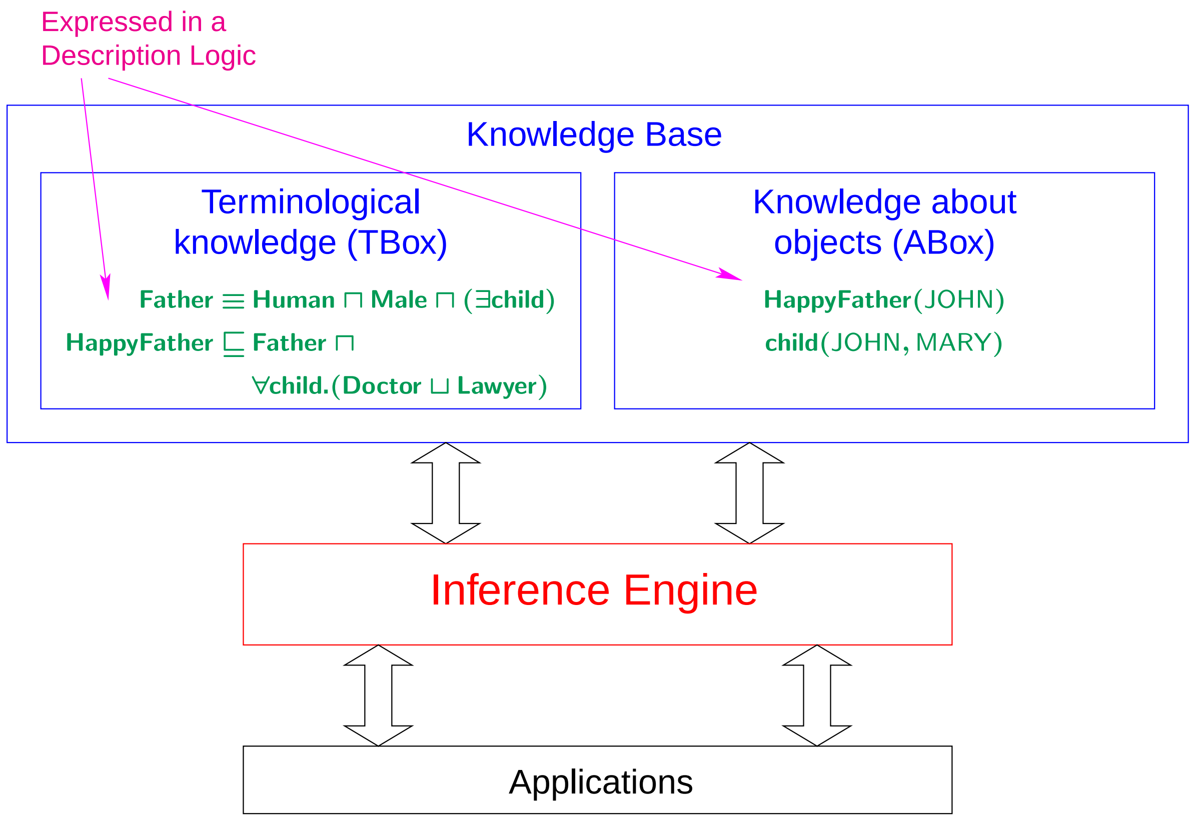
4. A set of inference services: how to reason on a given knowledge base

   $\mathcal{K} \;\models\; \textbf{HappyFather} \sqsubseteq \exists \textbf{child}.(\textbf{Doctor} \sqcup \textbf{Lawyer})$

   $\mathcal{K} \cup \mathcal{A} \;\models\; (\textbf{Doctor} \sqcup \textbf{Lawyer})(\mathsf{MARY})$

Note: we will consider ABoxes only later, when needed; hence, for now, we consider a knowledge base to be simply a TBox

Architecture of a DL system



Description language

A description language is characterized by a set of constructs for building complex concepts and roles starting from atomic ones:

- concepts represent classes: interpreted as sets of objects
- roles represent relations: interpreted as binary relations on objects

Semantics: in terms of interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where

- $\Delta^{\mathcal{I}}$ is the interpretation domain
- $\cdot^{\mathcal{I}}$ is the interpretation function, which maps
  - each atomic concept A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$
  - each atomic role P to a subset $P^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

The interpretation function is extended to complex concepts and roles according to their syntactic structure

Syntax and semantics of $\mathcal{AL}$

$\mathcal{AL}$ is the basic language in the family of $\mathcal{AL}$ languages

Construct	Syntax	Example	Semantics
atomic concept	A	Doctor	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
atomic role	P	child	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
atomic negation	$\neg A$	¬Doctor	$\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
conjunction	$C \sqcap D$	Hum ⊓ Male	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
(unqual.) exist. res.	$\exists R$	∃child	$\{ a \mid \exists b. (a,b) \in R^{\mathcal{I}} \}$
value restriction	$\forall R.C$	∀child.Male	$\{a \mid \forall b. (a,b) \in R^{\mathcal{I}} \supset b \in C^{\mathcal{I}}\}$

(C, D denote arbitrary concepts and R an arbitrary role)

Note: $\mathcal{AL}$ is not propositionally closed (no full negation)

The $\mathcal{AL}$ family

Typically, additional constructs w.r.t. those of $\mathcal{AL}$ are needed:

Construct	$\mathcal{AL}\cdot$	Syntax	Semantics
disjunction	$\mathcal{U}$	$C \sqcup D$	$C^\mathcal{I} \cup D^\mathcal{I}$
qual. exist. res.	$\mathcal{E}$	$\exists R.C$	$\{ a \mid \exists b. (a,b) \in R^\mathcal{I} \wedge b \in C^\mathcal{I} \}$
(full) negation	$\mathcal{C}$	$\neg C$	$\Delta^\mathcal{I} \setminus C^\mathcal{I}$
number restrictions	$\mathcal{N}$	$(\geq k\, R)$	$\{ a \mid \#\{b \mid (a,b) \in R^\mathcal{I}\} \geq k \}$
		$(\leq k\, R)$	$\{ a \mid \#\{b \mid (a,b) \in R^\mathcal{I}\} \leq k \}$
qual. number restrictions	$\mathcal{Q}$	$(\geq k\, R.C)$	$\{ a \mid \#\{b \mid (a,b) \in R^\mathcal{I} \wedge b \in C^\mathcal{I}\} \geq k \}$
		$(\leq k\, R.C)$	$\{ a \mid \#\{b \mid (a,b) \in R^\mathcal{I} \wedge b \in C^\mathcal{I}\} \leq k \}$
inverse role	$\mathcal{I}$	P^-	$\{ (a,b) \mid (b,a) \in P^\mathcal{I} \}$

We also use: $\bot$ for $A \sqcap \neg A$ (hence $\bot^\mathcal{I} = \emptyset$)
$\top$ for $A \sqcup \neg A$ (hence $\top^\mathcal{I} = \Delta^\mathcal{I}$)

The $\mathcal{AL}$ family – Examples

- Disjunction
  $\forall$child.(Doctor $\sqcup$ Lawyer)
- Qualified existential restriction
  $\exists$child.Doctor
- Full negation
  $\neg$(Doctor $\sqcup$ Lawyer)
- Number restrictions
  $(\geq 2\,$child$)$ $\sqcap$ $(\leq 1\,$sibling$)$
- Qualified number restrictions
  $(\geq 2\,$child.Doctor$)$ $\sqcap$ $(\leq 1\,$sibling.Male$)$
- Inverse role
  $\forall$child$^-$.Doctor

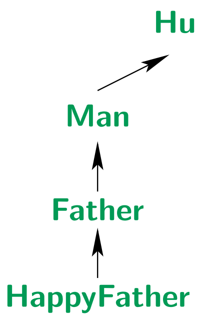
Reasoning on concept expressions

An interpretation $\mathcal{I}$ is a model of a concept C if $C^{\mathcal{I}} \neq \emptyset$

Basic reasoning tasks:

1. Concept satisfiability: does C admit a model?
2. Concept subsumption: does $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ hold for all interpretations $\mathcal{I}$?
   (written $C \sqsubseteq D$)

Subsumption used to build the concept hierarchy:



(1) and (2) are mutually reducible if DL is propositionally closed

Reasoning on concept expressions – Technique

Techniques are based on tableau algorithms: for satisfiability of C_0

1. Aims at building a tree representing a model of C_0
   - nodes represent objects of $\Delta^{\mathcal{I}}$, labeled with subconcepts of C_0
   - edges represent role successorship between objects
2. Concepts are first put in negation normal form (negation is pushed inside)
3. Tree initialized with single root node, labeled with $\{C_0\}$
4. Rules (one for each construct) add new nodes or concepts to the label
   - deterministic rules: for $\sqcap$, $\forall P.C$, $\exists P.C$, $(\geq k\, P)$
   - non-deterministic rules: for $\sqcup$, $(\leq k\, P)$
5. Stops when:
   - no more rule can be applied, or
   - a clash (obvious contradiction) is detected

Reasoning on concept expressions – Technique (Cont'd)

Properties of tableaux algorithms (must be proved for the various cases):

1. Termination: since quantifier depth decreases going down the tree
2. Soundness: if there is a way of terminating without a clash, then C_0 is satisfiable
   - construct from the tree a model of C_0
3. Completeness: if C_0 is satisfiable, there is a way of applying the rules so that the algorithm terminates without a clash
   - if $\mathcal{I}$ is a model of T, then there is a rule s.t. $\mathcal{I}$ is also a model of the tree obtained by applying the rule to T

Tableaux algorithms provide optimal decision procedures for concept satisfiability (and subsumption)

Reasoning on concept expressions – Complexity

Complexity of concept satisfiability

PTIME	$\mathcal{AL}$, $\mathcal{ALN}$
NP-complete	$\mathcal{ALU}$, $\mathcal{ALUN}$
coNP-complete	$\mathcal{ALE}$
PSPACE-complete	$\mathcal{ALC}$, $\mathcal{ALCN}$, $\mathcal{ALCI}$, $\mathcal{ALCQI}$

Observations:

- two sources of complexity
  - union ($\mathcal{U}$) of type NP
  - existential quantification ($\mathcal{E}$) of type coNP

  When they are combined, the complexity jumps to PSPACE
- number restrictions ($\mathcal{N}$) do not add to the complexity

Structural properties vs. asserted properties

We have seen how to build complex concept expressions, which allow to denote classes with a complex structure

However, in order to represent complex domains one needs the ability to assert properties of classes and relationships between them (e.g., as done in UML class diagrams)

The assertion of properties is done in DLs by means of knowledge bases

DL knowledge bases

A DL knowledge base consists of a set of inclusion assertions on concepts:

$$C \sqsubseteq D$$

- when C is an atomic concept, the assertion is called primitive
- $C \equiv D$ is an abbreviation for $C \sqsubseteq D,\ D \sqsubseteq C$

Example:

$$\begin{aligned} \mathcal{K} = \{ \ & \textsf{Father} \equiv \textsf{Human} \sqcap \textsf{Male} \sqcap (\exists \textsf{child}), \\ & \textsf{HappyFather} \sqsubseteq \textsf{Father} \sqcap \forall \textsf{child}.(\textsf{Doctor} \sqcup \textsf{Lawyer}) \ \} \end{aligned}$$

Semantics: An interpretation $\mathcal{I}$ is a model of a knowledge base $\mathcal{K}$ if

$$C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \quad \text{for every assertion } C \sqsubseteq D \text{ in } \mathcal{K}$$

Reasoning on DL knowledge bases

Basic reasoning tasks:

1. Knowledge base satisfiability
   Given $\mathcal{K}$, does it admit a model?

2. Concept satisfiability w.r.t. a KB — denoted $\mathcal{K} \not\models C \equiv \bot$
   Given C and $\mathcal{K}$, do they admit a common model?

3. Logical implication — denoted $\mathcal{K} \models C \sqsubseteq D$
   Given C, D, and $\mathcal{K}$, does $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ hold for all models $\mathcal{I}$ of $\mathcal{K}$?

Again, logical implication allows for classifying the concepts in the KB w.r.t. the knowledge expressed by the KB

Relationship among reasoning tasks

The reasoning tasks are mutually reducible to each other, provided the description language is propositionally closed:

(1) to (3) $\mathcal{K}$ satisfiable iff not $\mathcal{K} \models \top \sqsubseteq \bot$ iff $\mathcal{K} \not\models \top \equiv \bot$
(i.e., $\top$ satisfiable w.r.t. $\mathcal{K}$)

(3) to (2) $\mathcal{K} \models C \sqsubseteq D$ iff not $\mathcal{K} \not\models C \sqcap \neg D \equiv \bot$
(i.e., $C \sqcap \neg D$ unsatisfiable w.r.t. $\mathcal{K}$)

(2) to (1) $\mathcal{K} \not\models C \equiv \bot$ iff $\mathcal{K} \cup \{ \top \sqsubseteq \exists P_{new} \sqcap \forall P_{new}.C \}$ satisfiable
(where P_{new} is a new atomic role)

Relationship with First Order Logic

Most DLs are well-behaved fragments of First Order Logic

To translate $\mathcal{ALC}$ to FOL:

1. Introduce: a unary predicate $A(x)$ for each atomic concept A
   a binary predicate $P(x, y)$ for each atomic role P
2. Translate complex concepts as follows, using translation functions t_x, for any variable x:

$$\begin{aligned} t_x(A) &= A(x) \\ t_x(C \sqcap D) &= t_x(C) \wedge t_x(D) \\ t_x(C \sqcup D) &= t_x(C) \vee t_x(D) \\ t_x(\exists P.C) &= \exists y.\, P(x, y) \wedge t_y(C) && \text{with } y \text{ a new variable} \\ t_x(\forall P.C) &= \forall y.\, P(x, y) \supset t_y(C) && \text{with } y \text{ a new variable} \end{aligned}$$

3. Translate a knowledge base $\mathcal{K} = \bigcup_i \{ C_i \sqsubseteq D_i \}$ as a FOL theory

$$\Gamma_{\mathcal{K}} = \bigcup_i \{ \forall x.\, t_x(C_i) \supset t_x(D_i) \}$$

Relationship with First Order Logic (Cont'd)

Reasoning services:

C is consistent	iff	its translation $t_x(C)$ is satisfiable
$C \sqsubseteq D$	iff	$t_x(C) \supset t_x(D)$ is valid
C is consistent w.r.t. $\mathcal{K}$	iff	$\Gamma_{\mathcal{K}} \cup \{ \exists x.\, t_x(C) \}$ is satisfiable
$\mathcal{K} \models C \sqsubseteq D$	iff	$\Gamma_{\mathcal{K}} \models \forall x.\, (t_x(C) \supset t_x(D))$

Relationship with First Order Logic – Exercise

Translate the following $\mathcal{ALC}$ concepts into FOL formulas:

1. $\mathsf{Father} \sqcap \forall \mathsf{child}.(\mathsf{Doctor} \sqcup \mathsf{Manager})$
2. $\exists \mathsf{manages}.(\mathsf{Company} \sqcap \exists \mathsf{employs}.\mathsf{Doctor})$
3. $\mathsf{Father} \sqcap \forall \mathsf{child}.(\mathsf{Doctor} \sqcup \exists \mathsf{manages}.(\mathsf{Company} \sqcap \exists \mathsf{employs}.\mathsf{Doctor}))$

Solution:

1. $\mathsf{Father}(x) \land \forall y.\, (\mathsf{child}(x,y) \supset (\mathsf{Doctor}(y) \lor \mathsf{Manager}(y)))$
2. $\exists y.\, (\mathsf{manages}(x,y) \land (\mathsf{Company}(y) \land \exists w.\, (\mathsf{employs}(y,w) \land \mathsf{Doctor}(w))))$
3. $\mathsf{Father}(x) \land \forall y.\, (\mathsf{child}(x,y) \supset (\mathsf{Doctor}(y) \lor$
   $\exists w.\, (\mathsf{manages}(y,w) \land (\mathsf{Company}(w) \land \exists z.\, (\mathsf{employs}(w,z) \land \mathsf{Doctor}(z))))))$

DLs as fragments of First Order Logic

The above translation shows us that DLs are a fragment of First Order Logic

In particular, we can translate complex concepts using just two translation functions t_x and t_y (thus reusing the same variables):

$$
\begin{array}{rclrcl}
t_x(A) &=& A(x) & t_y(A) &=& A(y)\\
t_x(C \sqcap D) &=& t_x(C) \land t_x(D) & t_y(C \sqcap D) &=& t_y(C) \land t_y(D)\\
t_x(C \sqcup D) &=& t_x(C) \lor t_x(D) & t_y(C \sqcup D) &=& t_y(C) \lor t_y(D)\\
t_x(\exists P.C) &=& \exists y.\, P(x,y) \land t_y(C) & t_y(\exists P.C) &=& \exists x.\, P(y,x) \land t_x(C)\\
t_x(\forall P.C) &=& \forall y.\, P(x,y) \supset t_y(C) & t_y(\forall P.C) &=& \forall x.\, P(y,x) \supset t_x(C)
\end{array}
$$

$\leadsto$ $\mathcal{ALC}$ is a fragment of L2, i.e., FOL with 2 variables, known to be decidable (NEXPTIME-complete)

Note: FOL with 2 variables is more expressive than $\mathcal{ALC}$ (tradeoff expressive power vs. complexity of reasoning)

DLs as fragments of First Order Logic – Exercise

Translate the following $\mathcal{ALC}$ concepts into L2 formulas (i.e., into FOL formulas that use only variables x and y):

1. $\mathsf{Father} \sqcap \forall\mathsf{child}.(\mathsf{Doctor} \sqcup \mathsf{Manager})$
2. $\exists\mathsf{manages}.(\mathsf{Company} \sqcap \exists\mathsf{employs}.\mathsf{Doctor})$
3. $\mathsf{Father} \sqcap \forall\mathsf{child}.(\mathsf{Doctor} \sqcup \exists\mathsf{manages}.(\mathsf{Company} \sqcap \exists\mathsf{employs}.\mathsf{Doctor}))$

Solution:

1. $\mathsf{Father}(x) \land \forall y.\,(\mathsf{child}(x,y) \supset (\mathsf{Doctor}(y) \lor \mathsf{Manager}(y)))$
2. $\exists y.\,(\mathsf{manages}(x,y) \land (\mathsf{Company}(y) \land \exists x.\,(\mathsf{employs}(y,x) \land \mathsf{Doctor}(x))))$
3. $\mathsf{Father}(x) \land \forall y.\,(\mathsf{child}(x,y) \supset (\mathsf{Doctor}(y) \lor$ $\exists x.\,(\mathsf{manages}(y,x) \land (\mathsf{Company}(x) \land \exists y.\,(\mathsf{employs}(x,y) \land \mathsf{Doctor}(y))))))$

DLs as fragments of First Order Logic (Cont'd)

Translation can be extended to other constructs:

- For inverse roles, swap the variables in the role predicate, i.e.,

$$t_x(\exists P^-.C) = \exists y.\,P(y,x) \land t_y(C) \qquad \text{with } y \text{ a new variable}$$
$$t_x(\forall P^-.C) = \forall y.\,P(y,x) \supset t_y(C) \qquad \text{with } y \text{ a new variable}$$

  $\leadsto$ $\mathcal{ALCI}$ is still a fragment of L2

- For number restrictions, two variables do not suffice;
  but, $\mathcal{ALCQI}$ is a fragment of C2 (i.e, L2+counting quantifiers)

Relationship with Modal and Dynamic Logics

In understanding the computational properties of DLs a correspondence with Modal logics and in particular with Propositional Dynamic Logics (PDLs) has been proved essential

PDLs are logics specifically designed for reasoning about programs

PDLs have been widely studied in computer science, especially from the point of view of computational properties:

- tree model property
- small model property
- automata based reasoning techniques

Relationship with Modal Logics

$\mathcal{ALC}$ is a syntactic variant of $\mathbf{K}_m$ (i.e., multi-modal $\mathbf{K}$):

$$\begin{array}{rcl} C \sqcap D & \Leftrightarrow & C \wedge D \\ C \sqcup D & \Leftrightarrow & C \vee D \\ \neg C & \Leftrightarrow & \neg C \end{array} \qquad \begin{array}{rcl} \exists P.C & \Leftrightarrow & \Diamond_P C \\ \forall P.C & \Leftrightarrow & \Box_P C \end{array}$$

- no correspondence for inverse roles
- no correspondence for number restrictions

$\leadsto$ Concept consistency, subsumption in $\mathcal{ALC}$ $\Leftrightarrow$ satisfiability, validity in $\mathbf{K}_m$

To encode inclusion assertions, axioms are used

$\leadsto$ Logical implication in DLs corresponds to “global logical implication” in Modal Logics

Relationship with Propositional Dynamic Logics

$\mathcal{ALC}$ and $\mathcal{ALCI}$ can be encoded in Propositional Dynamic Logics (PDLs)

$$\begin{array}{rcl@{\qquad}rcl} C \sqcap D & \Leftrightarrow & C \wedge D & \exists R.C & \Leftrightarrow & \langle R \rangle C \\ C \sqcup D & \Leftrightarrow & C \vee D & \forall R.C & \Leftrightarrow & [R]C \\ \neg C & \Leftrightarrow & \neg C & & & \end{array}$$

Universal modality (or better "master modality") can be expressed in PDLs using reflexive-transitive closure:

- for $\mathcal{ALC}$ / PDL: $u = (P_1 \cup \cdots \cup P_m)^*$
- for $\mathcal{ALCI}$ / conversePDL: $u = (P_1 \cup \cdots \cup P_m \cup P_1^- \cup \cdots \cup P_m^-)^*$

Universal modality allows for internalizing assertions:

$$C \sqsubseteq D \quad \Leftrightarrow \quad [u](C \supset D)$$

Relationship with Propositional Dynamic Logics (Cont'd)

$\leadsto$ Concept satisfiability w.r.t. a KB (resp., logical implication) reduce to PDL (un)satisfiability:

$$\bigcup_i \{ C_i \sqsubseteq D_i \} \not\models C \equiv \bot \quad \Leftrightarrow \quad C \wedge \bigwedge_i [u](C_i \supset D_i) \text{ satisfiable}$$

$$\bigcup_i \{ C_i \sqsubseteq D_i \} \models C \sqsubseteq D \quad \Leftrightarrow \quad C \wedge \neg D \wedge \bigwedge_i [u](C_i \supset D_i) \text{ unsatisfiable}$$

Correspondence also extended to other constructs, e.g., number restrictions:

- polynomial encoding when numbers are represented in unary
- technique more involved when numbers are represented in binary

Note: there are DLs with non first-order constructs, such as various forms of fixpoint constructs. Such DLs still have a correspondence with variants of PDLs

Consequences of correspondence with PDLs

- PDL, conversePDL, DPDL, converseDPDL are EXPTIME-complete
  $\leadsto$ Logical implication in $\mathcal{ALCQI}$ is in EXPTIME

- PDLs enjoy the tree-model property: every satisfiable formula admits a model that has the structure of a (in general infinite) tree of linearly bounded width
  $\leadsto$ A satisfiable $\mathcal{ALCQI}$ knowledge base has a tree model

- PDLs admit optimal reasoning algorithms based on (two-way alternating) automata on infinite trees
  $\leadsto$ Automata-based algorithms are optimal for $\mathcal{ALCQI}$ logical implication

DL reasoning systems

Systems are available for reasoning on DL knowledge bases:

- FaCT [University of Manchester]
- Racer [University of Hamburg]
- Pellet [University of Maryland]

Some remarks on these systems:

- the state-of-the-art DL reasoning systems are based on tableaux techniques and not on automata techniques
  + easier to implement
  – not computationally optimal (NEXPTIME, 2NEXPTIME)
- the systems are highly optimized
- despite the high computational complexity, the performance is surprisingly good in real world applications:
  - knowledge bases with thousands of concepts and hundreds of axioms
  - outperform specialized modal logics reasoners

Summary on Description Logics

- Description Logics are logics for class-based modeling:
  - can be seen as a fragment of FOL with nice computational properties
  - tight relationship with Modal Logics and Propositional Dynamic Logics
- For reasoning over concept expressions, tableaux algorithms are optimal
- For most (decidable) DLs, reasoning over KBs is EXPTIME-complete:
  - tight upper bounds by automata based techniques
  - implemented systems exploit tableaux techniques, are suboptimal, but perform well in practice

Lets go back to our questions on reasoning on UML class diagrams

1. Can we develop sound, complete, and **terminating** reasoning procedures for reasoning on UML Class Diagrams?

   To answer this question we polynomially encode UML Class Diagrams in DLs

   $\leadsto$ reasoning on UML Class Diagrams can be done in EXPTIME

2. How hard is it to reason on UML Class Diagrams in general?

   To answer this question we polynomially reduce reasoning in EXPTIME-complete DLs to reasoning on UML class diagrams

   $\leadsto$ reasoning on UML Class Diagrams is in fact EXPTIME-hard

We start with point (2): EXPTIME lower bound established by encoding satisfiability of a concept w.r.t. an $\mathcal{ALC}$ KBs into consistency of a class in an

UML class diagram [AIJ2005].

Upper bound for reasoning on UML class diagrams

EXPTIME upper bound established by encoding UML class diagrams in DLs

What we gain by such an encoding

- DLs admit decidable inference
  $\leadsto$ decision procedure for reasoning in UML
- (most) DLs are decidable in EXPTIME
  $\leadsto$ EXPTIME method for reasoning in UML (provided the encoding in polynomial)
- exploit DL-based reasoning systems for reasoning in UML
- interface case-tools with DL-based reasoners to provide support during design (see demo on Monday)

Encoding of UML class diagrams in DLs

We encode an UML class diagram $\mathcal{D}$ into an $\mathcal{ALCQI}$ knowledge base $\mathcal{K}_{\mathcal{D}}$:

- classes are represented by concepts
- attributes and association roles are represented by roles
- each part of the diagram is encoded by suitable inclusion assertions

$\rightsquigarrow$ Consistency of a class in $\mathcal{D}$ is reduced to consistency of the corresponding concept w.r.t. $\mathcal{K}_{\mathcal{D}}$, similarly for the other reasoning tasks

Encoding of classes and attributes

- An UML class C is represented by an atomic concept C
- Each attribute a of type T for C is represented by an atomic role a
  - To encode the typing of a for C:

    $$C \sqsubseteq \forall a.T$$

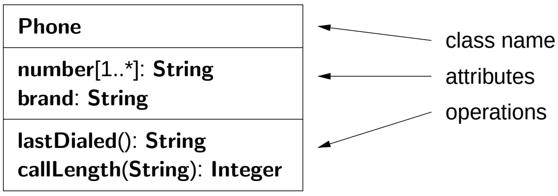
    This takes into account that other classes may also have attribute a
  - To encode the multiplicity $[i..j]$ of a:

    $$C \sqsubseteq (\geq i\, a) \sqcap (\leq j\, a)$$

    - ∗ when j is $*$, we omit the second conjunct
    - ∗ when the multiplicity is $[0..*]$ we omit the whole assertion
    - ∗ when the multiplicity is missing (i.e., $[1..1]$), the assertion becomes:

    $$C \sqsubseteq \exists a \sqcap (\leq 1\, a)$$

Encoding of classes and attributes – Example



- To encode the class **Phone**, we introduce a concept **Phone**
- Encoding of the attributes: **number** and **brand**

$$\textbf{Phone} \sqsubseteq \forall \textbf{number}.\textbf{String} \sqcap \exists \textbf{number}$$
$$\textbf{Phone} \sqsubseteq \forall \textbf{brand}.\textbf{String} \sqcap \exists \textbf{brand} \sqcap (\leq 1\ \textbf{brand})$$

- Encoding of the operations: $\textbf{lastDialed}()$ and $\textbf{callLength}(\textbf{String})$
  see later

Encoding of associations

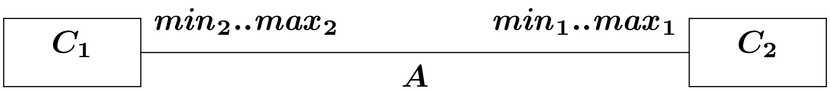
The encoding depends on:

- the presence/absence of an association class
- the arity of the association

	without association class	with association class
binary	via $\mathcal{ALCQI}$ role	via reification
non-binary	via reification	via reification

Note: for simplicity in the following we will only consider binary association without association classes role

Encoding of associations



- A is represented by an $\mathcal{ALCQI}$ role A, with:

$$\top \sqsubseteq \forall A.C_2 \sqcap \forall A^-.C_1$$

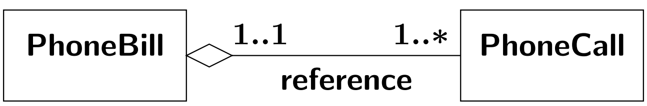
- To encode the multiplicities of A:
  - each instance of C_1 is connected through A to at least min_1 and at most max_1 instances of C_2:

$$C_1 \sqsubseteq (\geq min_1\ A) \sqcap (\leq max_1\ A)$$

  - each instance of C_2 is connected through A^- to at least min_2 and at most max_2 instances of C_1:

$$C_2 \sqsubseteq (\geq min_2\ A^-) \sqcap (\leq max_2\ A^-)$$

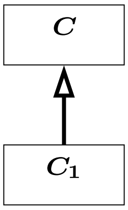
Associations – Example



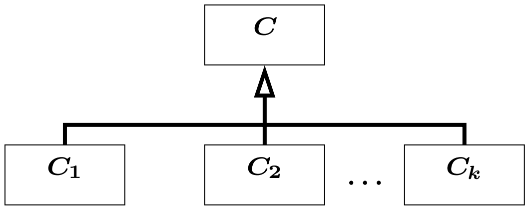
$$\top \sqsubseteq \forall \textsf{reference}.\textsf{PhoneCall} \sqcap \forall \textsf{reference}^-.\textsf{PhoneBill}$$
$$\textsf{PhoneBill} \sqsubseteq (\geq 1\ \textsf{reference})$$
$$\textsf{PhoneCall} \sqsubseteq (\geq 1\ \textsf{reference}^-) \sqcap (\leq 1\ \textsf{reference}^-)$$

Note: an aggregation is just a particular kind of binary association without association class

Encoding of ISA and generalization



$$C_1 \sqsubseteq C$$



$$C_1 \sqsubseteq C$$
$$\vdots$$
$$C_k \sqsubseteq C$$

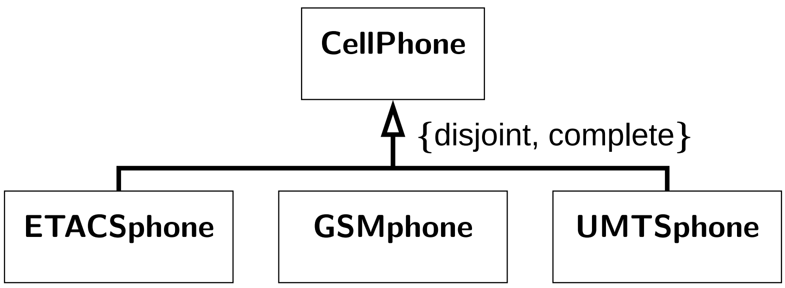
- When the generalization is disjoint

$$C_i \sqsubseteq \neg C_j \qquad \text{for } 1 \leq i < j \leq k$$

- When the generalization is complete

$$C \sqsubseteq C_1 \sqcup \cdots \sqcup C_k$$

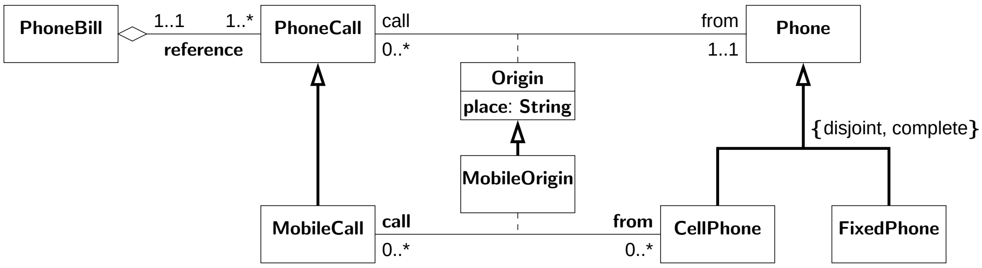
ISA and generalization – Example



ETACSphone	$\sqsubseteq$	CellPhone	ETACSphone	$\sqsubseteq$	$\neg$GSMPhone
GSMSphone	$\sqsubseteq$	CellPhone	ETACSphone	$\sqsubseteq$	$\neg$UMTSPhone
UMTSSphone	$\sqsubseteq$	CellPhone	GSMphone	$\sqsubseteq$	$\neg$UMTSPhone

CellPhone $\sqsubseteq$ ETACSphone $\sqcup$ GSMphone $\sqcup$ UMTSPhone

Encoding of UML in DLs – Example



$$
\begin{array}{rcl}
\top & \sqsubseteq & \forall \mathsf{reference}.\mathsf{PhoneCall} \sqcap \forall \mathsf{reference}^-.\mathsf{PhoneBill} \\
\mathsf{PhoneBill} & \sqsubseteq & (\geq 1\, \mathsf{reference}) \\
\mathsf{PhoneCall} & \sqsubseteq & (\geq 1\, \mathsf{reference}^-) \sqcap (\leq 1\, \mathsf{reference}^-) \\
\mathsf{Origin} & \sqsubseteq & \forall \mathsf{place}.\mathsf{String} \sqcap \exists \mathsf{place} \sqcap (\leq 1\, \mathsf{place}) \\
\mathsf{Origin} & \sqsubseteq & \exists \mathsf{call}.\mathsf{PhoneCall} \sqcap (\leq 1\, \mathsf{call}) \sqcap \exists \mathsf{from}.\mathsf{Phone} \sqcap (\leq 1\, \mathsf{from}) \\
\mathsf{MobileOrigin} & \sqsubseteq & \exists \mathsf{call}.\mathsf{MobileCall} \sqcap (\leq 1\, \mathsf{call}) \sqcap \exists \mathsf{from}.\mathsf{CellPhone} \sqcap (\leq 1\, \mathsf{from}) \\
\mathsf{PhoneCall} & \sqsubseteq & (\geq 1\, \mathsf{call}^-.\mathsf{Origin}) \sqcap (\leq 1\, \mathsf{call}^-.\mathsf{Origin}) \\
\mathsf{MobileOrigin} & \sqsubseteq & \mathsf{Origin} \\
\mathsf{MobileCall} & \sqsubseteq & \mathsf{PhoneCall} \\
\mathsf{CellPhone} & \sqsubseteq & \mathsf{Phone} \\
\mathsf{FixedPhone} & \sqsubseteq & \mathsf{Phone} \sqcap \neg \mathsf{CellPhone} \\
\mathsf{Phone} & \sqsubseteq & \mathsf{CellPhone} \sqcup \mathsf{FixedPhone}
\end{array}
$$

Encoding of UML in DLs – Exercise 1

Si faccia riferimento al diagramma delle classi UML mostrato all'inizio della lezione precedente

Translate the above UML class diagram into an $\mathcal{ALCQI}$ knowledge base

Encoding of UML in DLs – Solution of Exercise 1

Encoding of classes and attributes

$$
\begin{array}{rcl}
\mathsf{Scene} & \sqsubseteq & \forall \mathsf{code}.\mathsf{String} \sqcap \exists \mathsf{code} \sqcap (\leq 1\, \mathsf{code}) \\
\mathsf{Scene} & \sqsubseteq & \forall \mathsf{description}.\mathsf{Text} \sqcap \exists \mathsf{description} \sqcap (\leq 1\, \mathsf{description}) \\
\mathsf{Internal} & \sqsubseteq & \forall \mathsf{theater}.\mathsf{String} \sqcap \exists \mathsf{theater} \sqcap (\leq 1\, \mathsf{theater}) \\
\mathsf{External} & \sqsubseteq & \forall \mathsf{night\_scene}.\mathsf{Boolean} \sqcap \exists \mathsf{night\_scene} \sqcap (\leq 1\, \mathsf{night\_scene}) \\
\mathsf{Take} & \sqsubseteq & \forall \mathsf{nbr}.\mathsf{Integer} \sqcap \exists \mathsf{nbr} \sqcap (\leq 1\, \mathsf{nbr}) \\
\mathsf{Take} & \sqsubseteq & \forall \mathsf{filmed\_meters}.\mathsf{Real} \sqcap \exists \mathsf{filmed\_meters} \sqcap (\leq 1\, \mathsf{filmed\_meters}) \\
\mathsf{Take} & \sqsubseteq & \forall \mathsf{reel}.\mathsf{String} \sqcap \exists \mathsf{reel} \sqcap (\leq 1\, \mathsf{reel}) \\
\mathsf{Setup} & \sqsubseteq & \forall \mathsf{code}.\mathsf{String} \sqcap \exists \mathsf{code} \sqcap (\leq 1\, \mathsf{code}) \\
\mathsf{Setup} & \sqsubseteq & \forall \mathsf{photographic\_pars}.\mathsf{Text} \sqcap \exists \mathsf{photographic\_pars} \sqcap (\leq 1\, \mathsf{photographic\_pars}) \\
\mathsf{Location} & \sqsubseteq & \forall \mathsf{name}.\mathsf{String} \sqcap \exists \mathsf{name} \sqcap (\leq 1\, \mathsf{name}) \\
\mathsf{Location} & \sqsubseteq & \forall \mathsf{address}.\mathsf{String} \sqcap \exists \mathsf{address} \sqcap (\leq 1\, \mathsf{address}) \\
\mathsf{Location} & \sqsubseteq & \forall \mathsf{description}.\mathsf{Text} \sqcap \exists \mathsf{description} \sqcap (\leq 1\, \mathsf{description})
\end{array}
$$

Encoding of UML in DLs – Solution of Exercise 1 (Cont'd)

Encoding of hierarchies

$$
\begin{array}{rcl}
\mathsf{Internal} & \sqsubseteq & \mathsf{Scene} \\
\mathsf{External} & \sqsubseteq & \mathsf{Scene} \\
\mathsf{Scene} & \sqsubseteq & \mathsf{Internal} \sqcup \mathsf{External} \\
\mathsf{Internal} & \sqsubseteq & \neg \mathsf{External}
\end{array}
$$

Encoding of associations

$$
\begin{array}{rcl}
\top & \sqsubseteq & \forall \mathsf{stp\_for\_scn}.\mathsf{Setup} \sqcap \forall \mathsf{stp\_for\_scn}^{-}.\mathsf{Scene} \\
\mathsf{Scene} & \sqsubseteq & (\geq 1\, \mathsf{stp\_for\_scn}) \\
\mathsf{Setup} & \sqsubseteq & (\geq 1\, \mathsf{stp\_for\_scn}^{-}) \sqcap (\leq 1\, \mathsf{stp\_for\_scn}^{-}) \\
\top & \sqsubseteq & \forall \mathsf{tk\_of\_stp}.\mathsf{Take} \sqcap \forall \mathsf{tk\_of\_stp}^{-}.\mathsf{Setup} \\
\mathsf{Setup} & \sqsubseteq & (\geq 1\, \mathsf{tk\_of\_stp}) \\
\mathsf{Take} & \sqsubseteq & (\geq 1\, \mathsf{tk\_of\_stp}^{-}) \sqcap (\leq 1\, \mathsf{tk\_of\_stp}^{-}) \\
\top & \sqsubseteq & \forall \mathsf{located}.\mathsf{Location} \sqcap \forall \mathsf{located}^{-}.\mathsf{External} \\
\mathsf{External} & \sqsubseteq & (\geq 1\, \mathsf{located}) \sqcap (\leq 1\, \mathsf{located})
\end{array}
$$

Encoding of operations

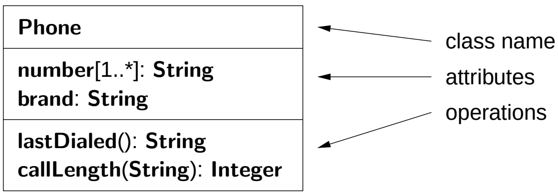
Operation $f(P_1, \ldots, P_m) : R$ for class C corresponds to an $(m+2)$-ary relation that is functional on the last component

- Operation $f() : R$ without parameters directly represented by an atomic role $\mathtt{P}_{f()}$, with:

$$C \sqsubseteq \forall \mathtt{P}_{f()}.R \sqcap (\leq 1\, \mathtt{P}_{f()})$$

- Operation $f(P_1, \ldots, P_m) : R$ with one or more parameters cannot be expressed directly in $\mathcal{ALCQI}$ $\leadsto$ we make use of reification (see [AIJ2005]

Encoding of operations – Example



- Encoding of the attributes: **number** and **brand**

$$\mathsf{Phone} \sqsubseteq \forall \mathsf{number}.\mathsf{String} \sqcap \exists \mathsf{number}$$
$$\mathsf{Phone} \sqsubseteq \forall \mathsf{brand}.\mathsf{String} \sqcap \exists \mathsf{brand} \sqcap (\leq 1\, \mathsf{brand})$$

- Encoding of the operations: **lastDialed()** and **callLength(String)**

$$\mathsf{Phone} \sqsubseteq \forall \mathtt{P}_{\mathsf{lastDialed()}}.\mathsf{String} \sqcap (\leq 1\, \mathtt{P}_{\mathsf{lastDialed()}})$$
$$\mathtt{P}_{\mathsf{callLength(String)}} \sqsubseteq \exists \mathtt{r}_0 \sqcap (\leq 1\, \mathtt{r}_0) \sqcap \exists \mathtt{r}_1 \sqcap (\leq 1\, \mathtt{r}_1) \sqcap \exists \mathtt{r}_2 \sqcap (\leq 1\, \mathtt{r}_2)$$
$$\mathtt{P}_{\mathsf{callLength(String)}} \sqsubseteq \forall \mathtt{r}_1.\mathsf{String}$$
$$\mathsf{Phone} \sqsubseteq \forall \mathtt{r}_0^-.(\mathtt{P}_{\mathsf{callLength(String)}} \Rightarrow \forall \mathtt{r}_2.\mathsf{Integer})$$

Correctness of the encoding

The encoding of an UML class diagram into an $\mathcal{ALCQI}$ knowledge base is correct, in the sense that it preserves the reasoning services over UML class diagrams

Proof idea: by showing a correspondence between the models of (the FOL formalization of) $\mathcal{D}$ and the models of $\mathcal{K}_{\mathcal{D}}$

Complexity of reasoning on UML class diagrams

All reasoning tasks on UML class diagrams can be reduced to reasoning tasks on $\mathcal{ALCQI}$ knowledge bases

From

- EXPTIME-completeness of reasoning on $\mathcal{ALCQI}$ knowledge bases
- the fact that the encoding in polynomial

we obtain:

Reasoning on UML class diagrams can be done in EXPTIME