Description Logics for Conceptual Data Modeling in UML

Giuseppe De Giacomo

Dipartimento di Informatica e Sistemistica

SAPIENZA Università di Roma

Description Logics

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
- Foundation for the semantic web
- ...

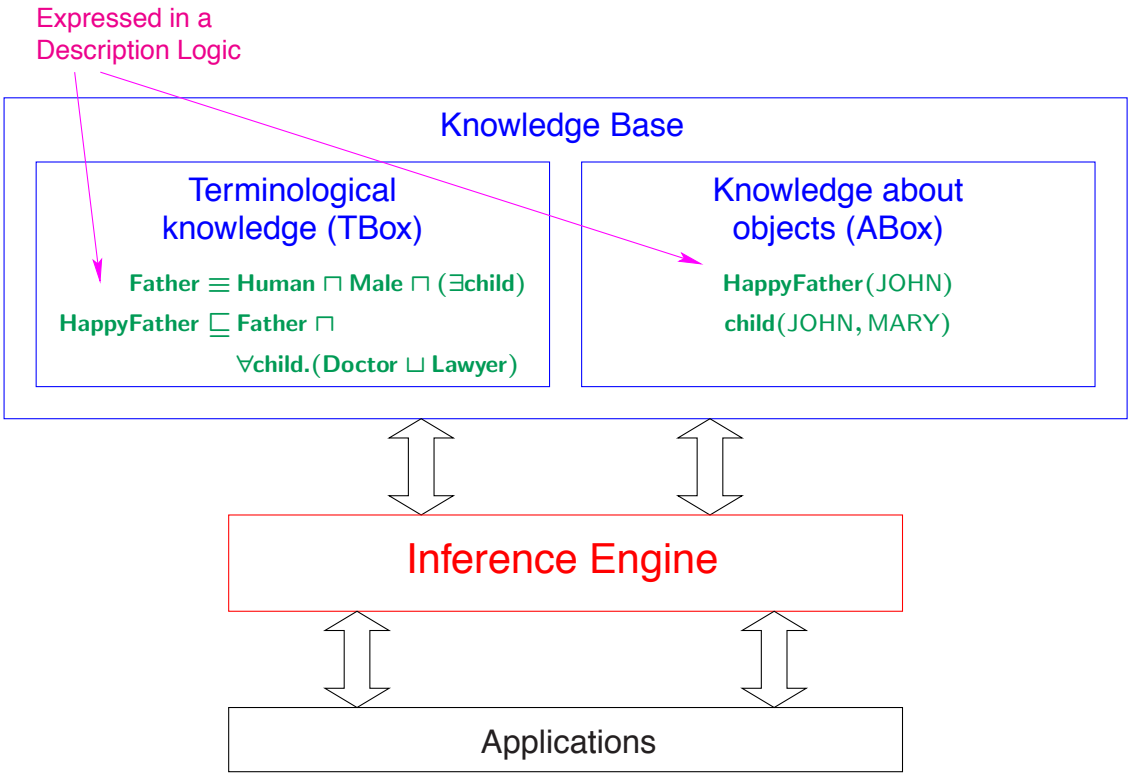
Ingredients of a DL

A Description Logic is characterized by:

1. A description language: how to form concepts and roles
   $\mathbf{Human} \sqcap \mathbf{Male} \sqcap (\exists \mathbf{child}) \sqcap \forall \mathbf{child}.(\mathbf{Doctor} \sqcup \mathbf{Lawyer})$
2. A mechanism to specify knowledge about concepts and roles (i.e., a TBox)
   $\mathcal{K} = \{ \mathbf{Father} \equiv \mathbf{Human} \sqcap \mathbf{Male} \sqcap (\exists \mathbf{child}),$
   $\mathbf{HappyFather} \sqsubseteq \mathbf{Father} \sqcap \forall \mathbf{child}.(\mathbf{Doctor} \sqcup \mathbf{Lawyer}) \}$
3. A mechanism to specify properties of objects (i.e., an ABox)
   $\mathcal{A} = \{ \mathbf{HappyFather}(\mathrm{JOHN}), \quad \mathbf{child}(\mathrm{JOHN}, \mathrm{MARY}) \}$
4. A set of inference services: how to reason on a given knowledge base
   $\mathcal{K} \models \mathbf{HappyFather} \sqsubseteq \exists \mathbf{child}.(\mathbf{Doctor} \sqcup \mathbf{Lawyer})$
   $\mathcal{K} \cup \mathcal{A} \models (\mathbf{Doctor} \sqcup \mathbf{Lawyer})(\mathrm{MARY})$

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

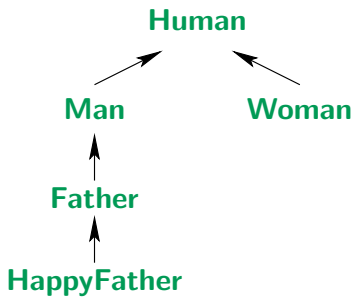
Reasoning on concept expressions

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 tableaux 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), but not for Knowledge Base reasoning (see later).

Reasoning on concept expressions – Technique (Cont'd)

For gentle introduction on how tableaux for $\mathcal{ALC}$ work without Knowledge base please refer to Ian Horrocks & Uli Sattler's slides:

`http://www.cs.man.ac.uk/~horrocks/Slides/IJCAR-tutorial/Print/p2-algorithmsI.pdf`

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:

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

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

$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every assertion $C \sqsubseteq D$ 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)

Tableaux procedures for reasoning with DLs knowledge bases

For details on tableaux based reasoning procedures for $\mathcal{ALC}$ knowledge bases, please refer refer to Ian Horrocks & Uli Sattler's slides:

`http://www.cs.man.ac.uk/~horrocks/Slides/IJCAR-tutorial/Print/p2-algorithmsII.pdf`

For resources on Description Logics, please refer to the Description Logics Site:

- `http://dl.kr.org/`

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

Solution:

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

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):

$$
\begin{array}{rcl}
t_x(A) & = & A(x) \\
t_x(C \sqcap D) & = & t_x(C) \wedge t_x(D) \\
t_x(C \sqcup D) & = & t_x(C) \vee t_x(D) \\
t_x(\exists P.C) & = & \exists y.\, P(x,y) \wedge t_y(C) \\
t_x(\forall P.C) & = & \forall y.\, P(x,y) \supset t_y(C)
\end{array}
\qquad
\begin{array}{rcl}
t_y(A) & = & A(y) \\
t_y(C \sqcap D) & = & t_y(C) \wedge t_y(D) \\
t_y(C \sqcup D) & = & t_y(C) \vee t_y(D) \\
t_y(\exists P.C) & = & \exists x.\, P(y,x) \wedge t_x(C) \\
t_y(\forall P.C) & = & \forall x.\, P(y,x) \supset t_x(C)
\end{array}
$$

$\rightsquigarrow$ $\mathcal{ALC}$ 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. $\mathbf{Father} \sqcap \forall \mathbf{child}.(\mathbf{Doctor} \sqcup \mathbf{Manager})$
2. $\exists \mathbf{manages}.(\mathbf{Company} \sqcap \exists \mathbf{employs}.\mathbf{Doctor})$
3. $\mathbf{Father} \sqcap \forall \mathbf{child}.(\mathbf{Doctor} \sqcup \exists \mathbf{manages}.(\mathbf{Company} \sqcap \exists \mathbf{employs}.\mathbf{Doctor}))$

Solution:

1. $\mathbf{Father}(x) \wedge \forall y.\, (\mathbf{child}(x,y) \supset (\mathbf{Doctor}(y) \vee \mathbf{Manager}(y)))$
2. $\exists y.\, (\mathbf{manages}(x,y) \wedge (\mathbf{Company}(y) \wedge \exists x.\, (\mathbf{employs}(y,x) \wedge \mathbf{Doctor}(x))))$
3. $\mathbf{Father}(x) \wedge \forall y.\, (\mathbf{child}(x,y) \supset (\mathbf{Doctor}(y) \vee \exists x.\, (\mathbf{manages}(y,x) \wedge (\mathbf{Company}(x) \wedge \exists y.\, (\mathbf{employs}(x,y) \wedge \mathbf{Doctor}(y))))))$

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.,

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

  $\rightsquigarrow \mathcal{ALCI}$ 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 $\mathbf{K}$):

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

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

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

To encode inclusion assertions, axioms are used

$\rightsquigarrow$ 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)

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

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

Universal modality allows for internalizing assertions:

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

Relationship with Propositional Dynamic Logics (Cont'd)

$\rightsquigarrow$ 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
  $\rightsquigarrow$ 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
  $\rightsquigarrow$ A satisfiable $\mathcal{ALCQI}$ knowledge base has a tree model

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

DL reasoning systems

Systems are available for reasoning on DL knowledge bases:

- FaCT/Fact++ [University of Manchester]
- Pellet [University of Maryland, Clark&Parsia]
- Racer/RacerPro [University of Hamburg, Racer Systems]

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

Identification constraints

Identification constraints (aka keys) are well-studied in

- relational databases
- conceptual data models (Entity-Relationship model, UML class diagrams)

Examples of keys:

- a student is identified by its id,
  i.e., no two students have the same id
- a company is identified by its telephone number,
  i.e., given a telephone number there is a unique company which owns it
  (although a company may own more than one telephone number)
- a person is identified by its name and surname,
  i.e., no two persons have the same name and surname

Keys in $\mathcal{ALCQI}$

Limited forms of keys can be expressed in $\mathcal{ALCQI}$ using number restrictions

Examples:

- a student is identified by its id

$$\mathbf{StudentId} \sqsubseteq \forall \mathbf{hasId}^{-}.\mathbf{Student} \sqcap (\leq 1\, \mathbf{hasId}^{-})$$

  and has a unique id, i.e., the student identifies the id

$$\mathbf{Student} \sqsubseteq \forall \mathbf{hasId}.\mathbf{StudentId} \sqcap (= 1\, \mathbf{hasId})$$

- a company is identified by its telephone number

$$\top \sqsubseteq (\leq 1\, \mathbf{telephone}^{-}.\mathbf{Company})$$

In $\mathcal{ALCQI}$ only unary keys can be expressed

Keys in $\mathcal{ALCQI}_{id}$

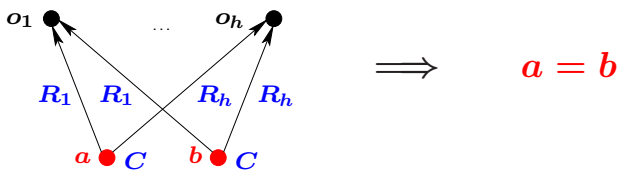
$\mathcal{ALCQI}_{id}$ KBs extend $\mathcal{ALCQI}$ KBs by key assertions:

$$(\mathrm{id}\ C\ R_1, \ldots, R_h)$$

A key assertion acts as a constraint, rather than denoting a set of objects

Semantics of a key assertion:

no two instances of C agree on the participation to $R_1, \ldots, R_h$



Example: a person is identified by its name and surname

$$(\mathrm{id}\ \mathbf{Person\ name, surname})$$

Reasoning in $\mathcal{ALCQI}_{id}$

Important observations:

- $\mathcal{ALCQI}$ knowledge bases have the tree-model property
- On tree-models, non-unary keys are trivially satisfied

Theorem: let $\mathcal{K}$ be a set of inclusion assertions, and
$\mathcal{F}$ be a set of non-unary key assertions

$$\mathcal{K} \cup \mathcal{F} \text{ satisfiable} \quad \text{iff} \quad \mathcal{K} \text{ satisfiable}$$

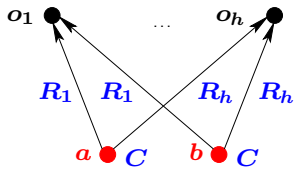
Since logical implication of inclusion assertions and concept satisfiability w.r.t. a KB can be reduced to KB satisfiability, we also have:

$$C \text{ satisfiable w.r.t. } \mathcal{K} \cup \mathcal{F} \quad \text{iff} \quad C \text{ satisfiable w.r.t. } \mathcal{K}$$
$$\mathcal{K} \cup \mathcal{F} \models C \sqsubseteq D \quad \text{iff} \quad \mathcal{K} \models C \sqsubseteq D$$

$\rightsquigarrow$ Key assertions do not interact with reasoning on inclusion assertions

Logical implication of keys in $\mathcal{ALCQI}_{id}$

To check $\mathcal{K} \cup \mathcal{F} \models (\mathbf{id}\ C\ R_1, \ldots R_h)$, reduce it to unsatisfiability of $\mathcal{K} \cup \mathcal{F} \cup \mathcal{A}$, where $\mathcal{A}$ is an ABox violating the key assertion:



To check satisfiability of $\mathcal{K} \cup \mathcal{F} \cup \mathcal{A}$, it is sufficient to check the key assertions in $\mathcal{F}$ on the objects of the ABox:

1. guess a saturation $\mathcal{A}_s$ of $\mathcal{A}$, i.e., a way of completing the knowledge about objects in $\mathcal{A}$ regarding concepts and roles in $\mathcal{F}$ ($\mathcal{A}_s$ is polynomial)
2. check that $\mathcal{A}_s$ satisfies $\mathcal{F}$ (polynomial)
3. check that $\mathcal{K} \cup \mathcal{A} \cup \mathcal{A}_s$ is satisfiable (exponential)

$\rightsquigarrow$ Logical implication in $\mathcal{ALCQI}_{id}$ is EXPTIME-complete

Reasoning on DL knowledge bases – Lower bounds

We have seen that reasoning on DL knowledge bases can be done in EXPTIME (e.g., by exploiting automata based techniques)

Are such techniques optimal for DL reasoning?
What is the intrinsic complexity of reasoning on DL knowledge bases?

Theorem: Logical implication in $\mathcal{ALC}$ (and hence concept satisfiability w.r.t. an $\mathcal{ALC}$ KB) is EXPTIME-hard

Reasoning on DL knowledge bases – Lower bounds (Cont'd)

The lower bound for logical implication in DLs can be strengthened

Theorem: concept satisfiability w.r.t. an $\mathcal{AL}$ KB and logical implication in $\mathcal{AL}$ are EXPTIME-hard

Proof: by reducing concept satisfiability w.r.t. an $\mathcal{ALC}$ KB in various steps to concept satisfiability w.r.t. an $\mathcal{AL}$ KB:

1. Reduce to satisfiability of an atomic concept w.r.t. a KB with primitive inclusion assertions only
2. Eliminate nesting of constructs in right hand side by introducing new assertions
3. Encode away qualified existential quantification
4. Encode away disjunction

1. Simplify assertions and concept

Reduce to satisfiability of an atomic concept w.r.t. a KB $\mathcal{K}$ with primitive inclusion assertions only:

$$C \text{ satisfiable w.r.t. } \bigcup_i \{ C_i \sqsubseteq D_i \}$$

iff

$$A_T \sqcap C \text{ satisfiable w.r.t. } \{ A_T \sqsubseteq \bigsqcap_i (\neg C_i \sqcup D_i) \sqcap \bigsqcap_P \forall P.A_T \}$$

iff

$$A_C \text{ satisfiable w.r.t. } \left\{ \begin{array}{l} A_T \sqsubseteq \bigsqcap_i (\neg C_i \sqcup D_i) \sqcap \bigsqcap_P \forall P.A_T, \\ A_C \sqsubseteq A_T \sqcap C \end{array} \right\}$$

with A_T and A_C new atomic concepts

2. Eliminate nesting of constructs in right hand side

Proceed as follows:

1. Push negation inside
2. Replace assertions as follows:

$$\begin{array}{lcl} A \sqsubseteq C_1 \sqcap C_2 & \Rightarrow & A \sqsubseteq C_1, \quad A \sqsubseteq C_2 \\ A \sqsubseteq C_1 \sqcup C_2 & \Rightarrow & A \sqsubseteq A_1 \sqcup A_2, \quad A_1 \sqsubseteq C_1, \quad A_2 \sqsubseteq C_2 \\ A \sqsubseteq \forall P.C & \Rightarrow & A \sqsubseteq \forall P.A_1, \quad A_1 \sqsubseteq C \\ A \sqsubseteq \exists P.C & \Rightarrow & A \sqsubseteq \exists P.A_1, \quad A_1 \sqsubseteq C \end{array}$$

with A_1, A_2 new atomic concepts for each replacement

Let $\mathcal{K}'$ be obtained from $\mathcal{K}$ by (1) and (2) above. We have

$$A_C \text{ satisfiable w.r.t. } \mathcal{K} \quad \text{iff} \quad A_C \text{ satisfiable w.r.t. } \mathcal{K}'$$

3. Encode away qualified existential quantification

Proceed as follows:

1. For each $\exists P.A$ appearing in $\mathcal{K}$, introduce a new atomic role P_A
2. Replace assertions as follows:

$$A \sqsubseteq \exists P.A' \quad \Rightarrow \quad A \sqsubseteq \exists P_{A'} \sqcap \forall P_{A'}.A'$$
$$A \sqsubseteq \forall P.A' \quad \Rightarrow \quad A \sqsubseteq \forall P.A' \sqcap \bigsqcap_{P_{A_i}} \forall P_{A_i}.A'$$

Let $\mathcal{K}''$ be obtained from $\mathcal{K}'$ by (1) and (2) above. We have

A_C satisfiable w.r.t. $\mathcal{K}'$ iff A_C satisfiable w.r.t. $\mathcal{K}''$

$\rightsquigarrow$ Concept satisfiability w.r.t. a (primitive) $\mathcal{ALU}$ KB is EXPTIME-hard

4. Encode away disjunction

Replace assertions as follows:

$$A_1 \sqsubseteq A_2 \sqcup A_3 \quad \Rightarrow \quad \neg A_2 \sqcap \neg A_3 \sqsubseteq \neg A_1$$

The two assertions are logically equivalent

$\rightsquigarrow$ Concept satisfiability w.r.t. an $\mathcal{AL}$ KB is EXPTIME-hard

Concept satisfiability w.r.t. an $\mathcal{AL}$ KB can be reduced to logical implication in $\mathcal{AL}$:

C satisfiable w.r.t. $\mathcal{K}$ iff not $\mathcal{K} \models C \sqsubseteq \bot$

$\rightsquigarrow$ Logical implication in $\mathcal{AL}$ is EXPTIME-hard

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
- Techniques can be extended to deal also with key constraints

EXPTIME-Hardness of Reasoning on UML Class Diagrams

We are now ready to answer our initial questions

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

   $\rightsquigarrow$ 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

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

We start with point (2)

Reasoning tasks on UML class diagrams

1. Consistency of the whole class diagram
2. Class consistency
3. Class subsumption
4. Class equivalence
5. $\cdots$

Obviously:

- Consistency of the class diagram can be reduced to class consistency
- Class equivalence can be reduced to class subsumption

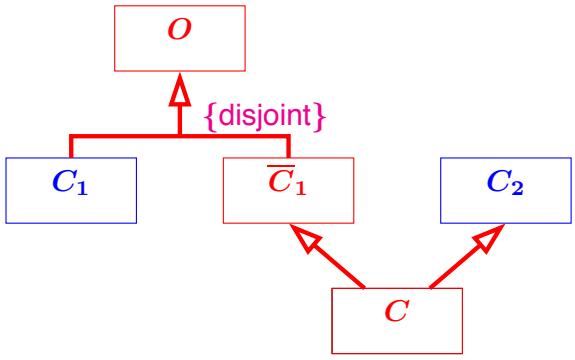
We show that also class consistency and class subsumption are mutually reducible

This allows us to concentrate on class consistency only

Reducing class subsumption to class consistency

To check whether a class C_1 subsumes a class C_2 in a class diagram $\mathcal{D}$:

1. Add to $\mathcal{D}$ the following part, with O, C, and $\overline{C}_1$ new classes

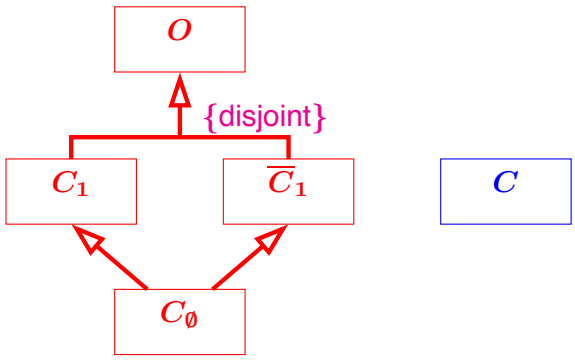


2. Check whether C is inconsistent in the resulting diagram

Reducing class consistency to class subsumption

To check whether a class C is inconsistent in a class diagram $\mathcal{D}$:

1. Add to $\mathcal{D}$ the following part, with O, C_1, $\overline{C}_1$, and $C_\emptyset$ new classes



2. Check whether $C_\emptyset$ subsumes C in the resulting diagram

Lower bound for reasoning on UML class diagrams

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

We exploit the reductions in the hardness proof of reasoning over $\mathcal{AL}$ KBs:

- By step (1) it suffices to consider satisfiability of an atomic concept w.r.t. an $\mathcal{ALC}$ knowledge base with primitive inclusion assertions only, i.e., of the form

$$A \sqsubseteq C$$

- By step (2) it suffices to consider concepts on the right hand side that contain only a single construct, i.e., assertions of the form

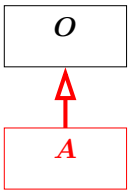
$$A \sqsubseteq B \qquad A \sqsubseteq \neg B \qquad A \sqsubseteq B_1 \sqcup B_2 \qquad A \sqsubseteq \forall P.B \qquad A \sqsubseteq \exists P.B$$

Note: by step (3) it would suffice to encode $A \sqsubseteq \exists P$ *instead of* $A \sqsubseteq \exists P.B$

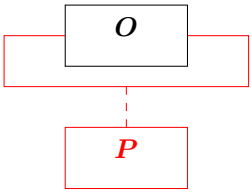
UML class diagram corresponding to an $\mathcal{ALC}$ KB - Optional

Given an $\mathcal{ALC}$ knowledge base $\mathcal{K}$ of the simplified form above, we construct an UML class diagram $\mathcal{D}_\mathcal{K}$:

- we introduce in $\mathcal{D}_\mathcal{K}$ a class O, intended to represent the whole domain
- for each atomic concept A in $\mathcal{K}$, we introduce in $\mathcal{D}_\mathcal{K}$ a class A

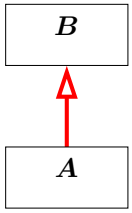


- for each atomic role P in $\mathcal{K}$, we introduce in $\mathcal{D}_\mathcal{K}$ a binary association P with related association class

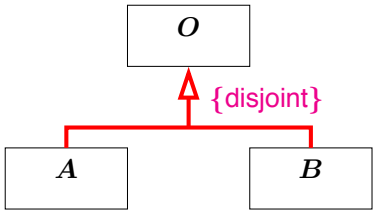


Encoding of $\mathcal{ALC}$ assertions - Optional

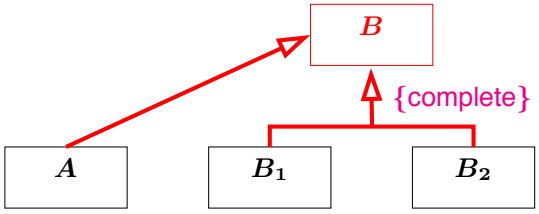
$A \sqsubseteq B$



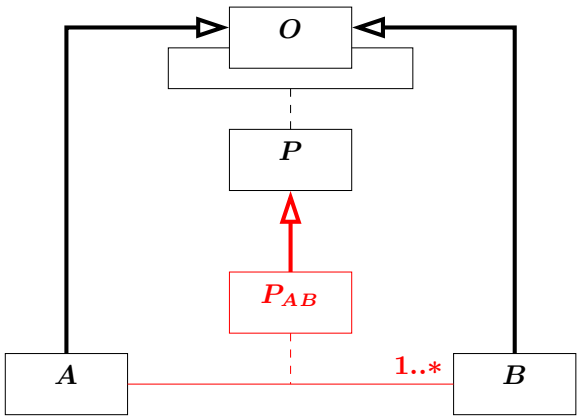
$A \sqsubseteq \neg B$



$A \sqsubseteq B_1 \sqcup B_2$

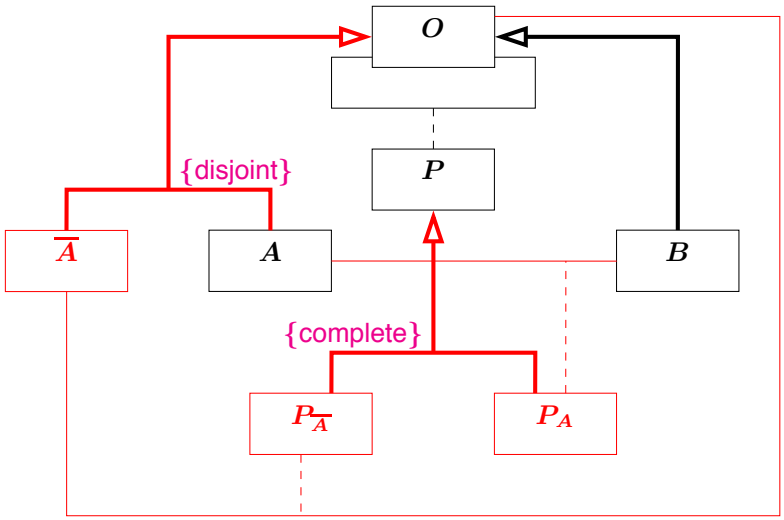


Encoding of $\mathcal{ALC}$ assertions (Cont'd) - Optional



$A \sqsubseteq \exists P.B$

Encoding of $\mathcal{ALC}$ assertions (Cont'd) - Optional



$$A \sqsubseteq \forall P.B$$

Correctness of the encoding - Optional

The encoding of an $\mathcal{ALC}$ knowledge base (of the simplified form) into an UML class diagram is correct, in the sense that it preserves concept satisfiability

Theorem:

An atomic concept A is satisfiable w.r.t. an $\mathcal{ALC}$ knowledge base $\mathcal{K}$
if and only if
the class A is consistent in the UML class diagram $\mathcal{D}_{\mathcal{K}}$ encoding $\mathcal{K}$

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

Lower bound for reasoning on UML class diagrams

The UML class diagram $\mathcal{D}_{\mathcal{K}}$ constructed from an $\mathcal{ALC}$ knowledge base $\mathcal{K}$ is of polynomial size in $\mathcal{K}$

From

- EXPTIME-hardness of concept satisfiability w.r.t. an $\mathcal{ALC}$ knowledge base
- the fact that the encoding in polynomial

we obtain:

Reasoning on UML class diagrams is EXPTIME-hard

Reasoning on UML Class Diagrams using Description Logics

Reasoning on UML class diagrams using DLs

We can use DLs to polynomially encode UML class diagrams: this gives us EXPTIME upper bound on reasoning with UML class diagrams.

More precisely from such encoding we get

- 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

Encoding of UML class diagrams in DLs

We encode an UML class diagram $\mathcal{D}$ into an $\mathcal{ALCQI}_{id}$ 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
- the properties of association classes are encoded trough suitable key assertions

$\leadsto$ Consistency of a class in $\mathcal{D}$ is reduced to consistency of the corresponding concept w.r.t. $\mathcal{K}_{\mathcal{D}}$

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_C
  - To encode the typing of a for C:

$$\exists a_C \sqsubseteq C \qquad \exists a_C^- \sqsubseteq T$$

We use a_C as name of the role to take into account that that the attribute a is local to the class C. Sometimes, for simplicity, we directly use a instead of a_C.

  - To encode the multiplicity $[i..j]$ of a:

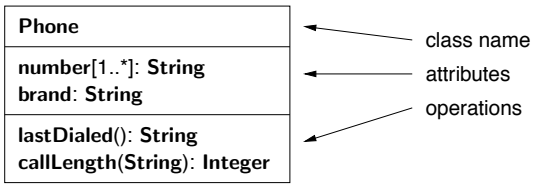
$$C \sqsubseteq (\geq i\, a_C) \sqcap (\leq j\, a_C)$$

    - 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:

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

Encoding of classes and attributes – Example



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

$$\exists \mathbf{number} \sqsubseteq \mathbf{Phone} \qquad \exists \mathbf{brand} \sqsubseteq \mathbf{Phone}$$
$$\exists \mathbf{number}^- \sqsubseteq \mathbf{String} \qquad \exists \mathbf{brand}^- \sqsubseteq \mathbf{String}$$
$$\mathbf{Phone} \sqsubseteq \exists \mathbf{number} \qquad \mathbf{Phone} \sqsubseteq \exists \mathbf{brand} \sqcap (\leq 1\, \mathbf{brand})$$

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

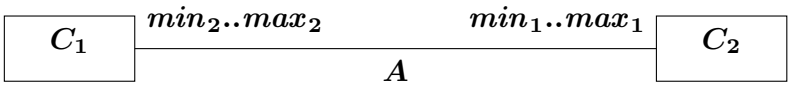
Encoding of associations

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

Encoding of binary associations without association class



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

$$\exists A \sqsubseteq C_1 \qquad \exists A^- \sqsubseteq C_2$$

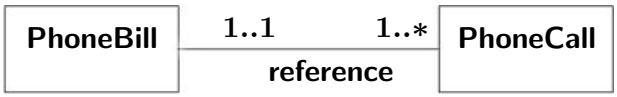
- To encode the multiplicities of A:
  - each instance of C_1 is connected through A to at least min_1 and at most max_1 instances of C_2:

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

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

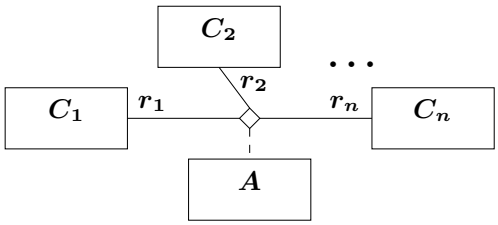
$$C_2 \sqsubseteq (\geq min_2\ A^-) \sqcap (\leq max_2\ A^-)$$

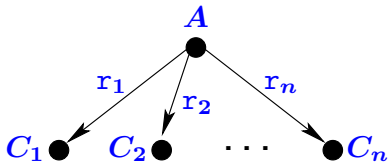
Binary associations without association class – Example



$$\exists \mathsf{reference} \sqsubseteq \mathsf{PhoneBill}$$
$$\exists \mathsf{reference}^- \sqsubseteq \mathsf{PhoneCall}$$
$$\mathsf{PhoneBill} \sqsubseteq \exists \mathsf{reference}$$
$$\mathsf{PhoneCall} \sqsubseteq \exists \mathsf{reference}^- \sqcap (\leq 1\ \mathsf{reference}^-)$$

Encoding of associations via reification



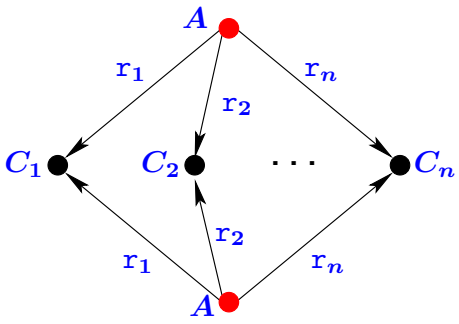


- Association A is represented by a concept A
- Each instance of the concept represents a tuple of the relation
- n (binary) roles $\mathtt{r}_{A,1}, \ldots, \mathtt{r}_{A,n}$ are used to connect the object representing a tuple to the objects representing the components of the tuple
- To ensure that the instances of A correctly represent tuples:

$$\exists \mathtt{r}_{A,i} \sqsubseteq A \qquad \exists \mathtt{r}_{A,i}^{-} \sqsubseteq C_i \qquad i = 1, \ldots, n$$
$$A \sqsubseteq \exists \mathtt{r}_{A,1} \sqcap \cdots \sqcap \exists \mathtt{r}_{A,n} \sqcap (\leq 1\, \mathtt{r}_{A,1}) \sqcap \cdots \sqcap (\leq 1\, \mathtt{r}_{A,n})$$

Encoding of associations via reification

We have not ruled out the existence of two instances of A representing the same tuple of association A:



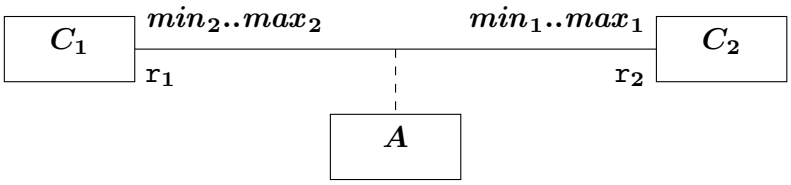
To rule out such a situation we could add a key assertion:

$$(\mathtt{id}\ A\ \mathtt{r}_1, \ldots, \mathtt{r}_n)$$

Note: in a tree-model the above situation cannot occur

$\rightsquigarrow$ Since in reasoning on an $\mathcal{ALCQI}$ KB we can restrict the attention to tree-models, we can ignore the key assertions

Multiplicities of binary associations with association class



To encode the multiplicities of A we need qualified number restrictions:

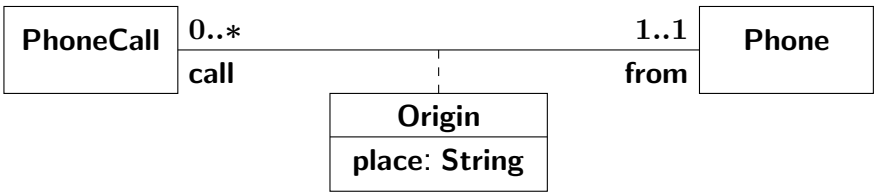
- each instance of C_1 is connected through A to at least min_1 and at most max_1 instances of C_2:

$$C_1 \sqsubseteq (\geq min_1\ \mathtt{r}^-_{A,1}) \sqcap (\leq max_1\ \mathtt{r}^-_{A,1})$$

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

$$C_2 \sqsubseteq (\geq min_2\ \mathtt{r}^-_{A,2}) \sqcap (\leq max_2\ \mathtt{r}^-_{A.2})$$

Associations with association class – Example



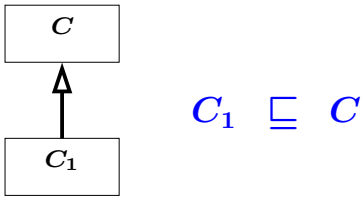
$\exists \mathsf{call} \sqsubseteq \mathsf{Origin} \qquad \exists \mathsf{call}^- \sqsubseteq \mathsf{PhoneCall}$

$\exists \mathsf{from} \sqsubseteq \mathsf{Origin} \qquad \exists \mathsf{from}^- \sqsubseteq \mathsf{Phone}$

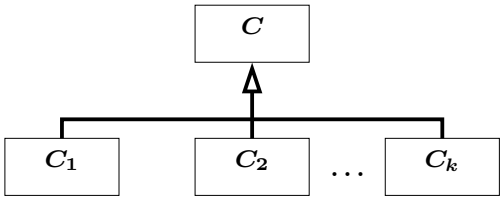
$\mathsf{Origin} \sqsubseteq \exists \mathsf{call} \sqcap (\leq 1\ \mathsf{call}) \sqcap \exists \mathsf{from} \sqcap (\leq 1\ \mathsf{from})$

$\mathsf{PhoneCall} \sqsubseteq (\geq 1\ \mathsf{call}^-) \sqcap (\leq 1\ \mathsf{call}^-)$

Encoding of ISA and generalization



$C_1 \sqsubseteq C$



$C_1 \sqsubseteq C$

$\vdots$

$C_k \sqsubseteq C$

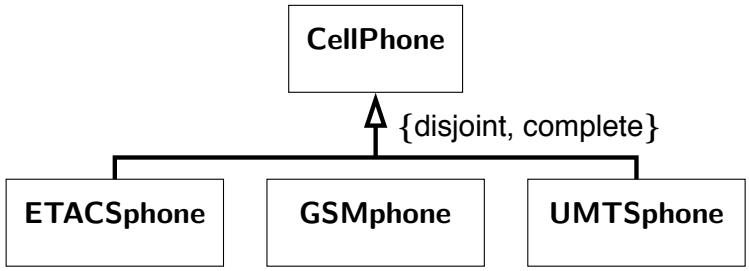
- When the generalization is disjoint

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

- When the generalization is complete

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

ISA and generalization – Example



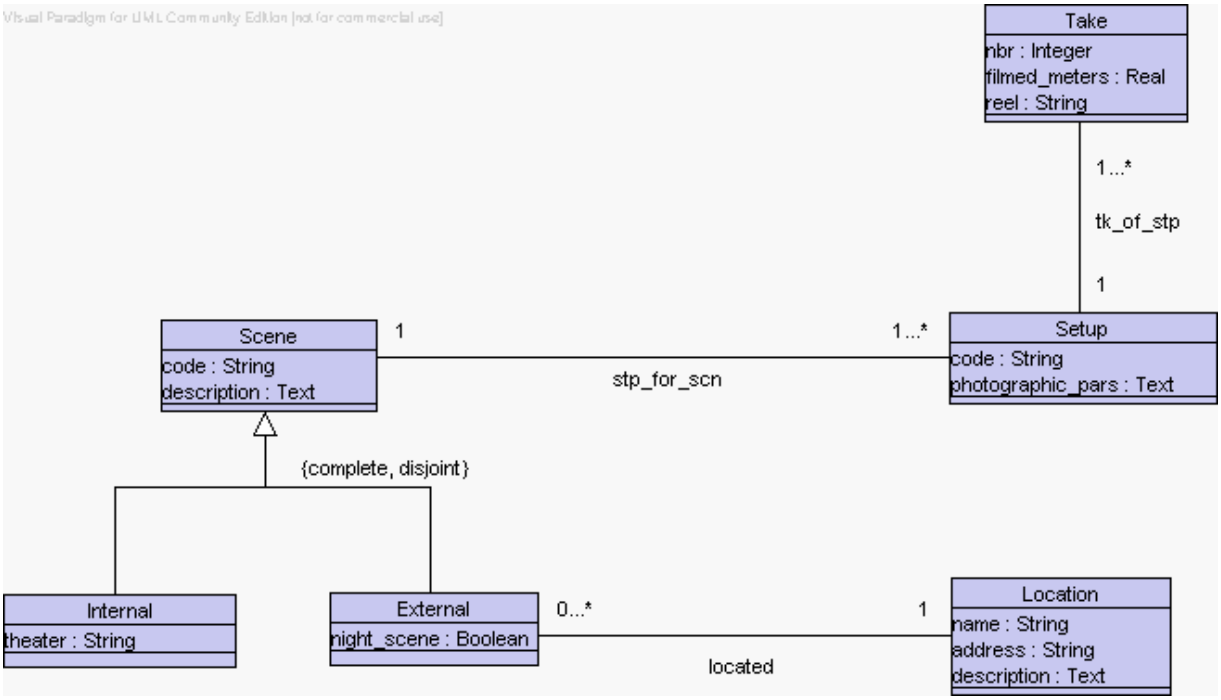
$\mathbf{ETACSphone} \sqsubseteq \mathbf{CellPhone}$ $\quad$ $\mathbf{ETACSphone} \sqsubseteq \neg\mathbf{GSMPhone}$

$\mathbf{GSMSphone} \sqsubseteq \mathbf{CellPhone}$ $\quad$ $\mathbf{ETACSphone} \sqsubseteq \neg\mathbf{UMTSPhone}$

$\mathbf{UMTSSphone} \sqsubseteq \mathbf{CellPhone}$ $\quad$ $\mathbf{GSMphone} \sqsubseteq \neg\mathbf{UMTSPhone}$

$\mathbf{CellPhone} \sqsubseteq \mathbf{ETACSphone} \sqcup \mathbf{GSMphone} \sqcup \mathbf{UMTSPhone}$

Encoding of UML in DLs – Exercise 1



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

$$\exists \mathbf{code}_{\mathbf{Scene}} \sqsubseteq \mathbf{Scene}$$
$$\exists \mathbf{code}_{\mathbf{Scene}}^{-} \sqsubseteq \mathbf{String}$$
$$\mathbf{Scene} \sqsubseteq \exists \mathbf{code}_{\mathbf{Scene}} \sqcap (\leq 1\ \mathbf{code}_{\mathbf{Scene}})$$

$$\exists \mathbf{description}_{\mathbf{Scene}} \sqsubseteq \mathbf{Scene}$$
$$\exists \mathbf{description}_{\mathbf{Scene}}^{-} \sqsubseteq \mathbf{Text}$$
$$\mathbf{Scene} \sqsubseteq \exists \mathbf{description}_{\mathbf{Scene}} \sqcap (\leq 1\ \mathbf{description}_{\mathbf{Scene}})$$

$$\exists \mathbf{theater} \sqsubseteq \mathbf{Internal}$$
$$\exists \mathbf{theater}^{-} \sqsubseteq \mathbf{String}$$
$$\mathbf{Internal} \sqsubseteq \exists \mathbf{theater} \sqcap (\leq 1\ \mathbf{theater})$$

$$\exists \mathbf{nightScene} \sqsubseteq \mathbf{External}$$
$$\exists \mathbf{nightScene}^{-} \sqsubseteq \mathbf{Boolean}$$
$$\mathbf{External} \sqsubseteq \exists \mathbf{nightScene} \sqcap (\leq 1\ \mathbf{nightScene})$$

$$\exists \mathbf{name}_{\mathbf{Location}} \sqsubseteq \mathbf{Location}$$
$$\exists \mathbf{name}_{\mathbf{Location}}^{-} \sqsubseteq \mathbf{String}$$
$$\mathbf{Location} \sqsubseteq \exists \mathbf{name}_{\mathbf{Location}} \sqcap (\leq 1\ \mathbf{name}_{\mathbf{Location}})$$

$$\exists \mathbf{address}_{\mathbf{Location}} \sqsubseteq \mathbf{Location}$$
$$\exists \mathbf{address}_{\mathbf{Location}}^{-} \sqsubseteq \mathbf{String}$$
$$\mathbf{Location} \sqsubseteq \exists \mathbf{address}_{\mathbf{Location}} \sqcap (\leq 1\ \mathbf{address}_{\mathbf{Location}})$$

$$\exists \mathbf{description}_{\mathbf{Location}} \sqsubseteq \mathbf{Location}$$
$$\exists \mathbf{description}_{\mathbf{Location}}^{-} \sqsubseteq \mathbf{Text}$$
$$\mathbf{Location} \sqsubseteq \exists \mathbf{description}_{\mathbf{Location}} \sqcap (\leq 1\ \mathbf{description}_{\mathbf{Location}})$$

$$\ldots$$

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

Encoding of hierarchies

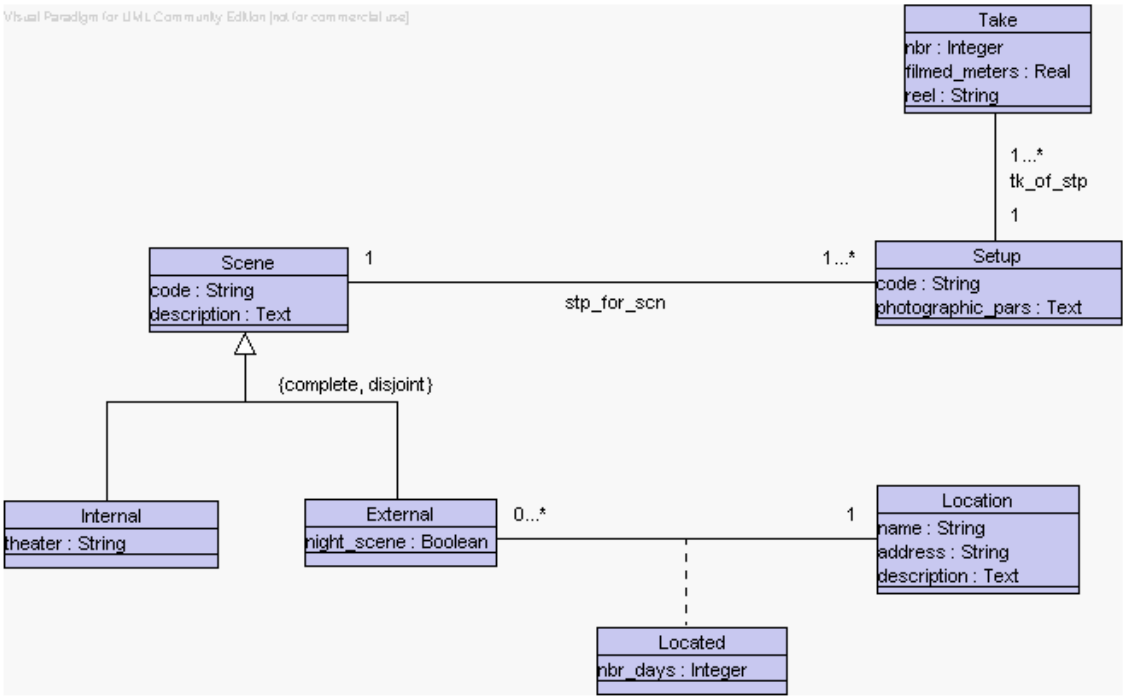
$$\begin{aligned} \textbf{Internal} &\sqsubseteq \textbf{Scene} \\ \textbf{External} &\sqsubseteq \textbf{Scene} \\ \textbf{Scene} &\sqsubseteq \textbf{Internal} \sqcup \textbf{External} \\ \textbf{Internal} &\sqsubseteq \neg\textbf{External} \end{aligned}$$

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

Encoding of associations

$$\begin{aligned} \exists\textbf{stp\_for\_scn} &\sqsubseteq \textbf{Scene} \\ \exists\textbf{stp\_for\_scn}^- &\sqsubseteq \textbf{Setup} \\ \textbf{Scene} &\sqsubseteq \exists\textbf{stp\_for\_scn} \\ \textbf{Setup} &\sqsubseteq \exists\textbf{stp\_for\_scn}^- \sqcap (\leq 1\ \textbf{stp\_for\_scn}^-) \\ \exists\textbf{tk\_of\_stp} &\sqsubseteq \textbf{Setup} \\ \exists\textbf{tk\_of\_stp}^- &\sqsubseteq \textbf{Take} \\ \textbf{Setup} &\sqsubseteq \exists\textbf{tk\_of\_stp} \\ \textbf{Take} &\sqsubseteq \exists\textbf{tk\_of\_stp}^- \sqcap (\leq 1\ \textbf{tk\_of\_stp}^-) \\ \exists\textbf{located} &\sqsubseteq \textbf{External} \\ \exists\textbf{located}^- &\sqsubseteq \textbf{Location} \\ \textbf{External} &\sqsubseteq (\geq 1\ \textbf{located}) \sqcap (\leq 1\ \textbf{located}) \end{aligned}$$

Encoding of UML in DLs – Exercise 2



How does the translation change w.r.t. the one for Exercise 1?

Encoding of UML in DLs – Solution of Exercise 2

The change is in the encoding of the association **located**, which now must be reified into a concept **Located**, i.e.,

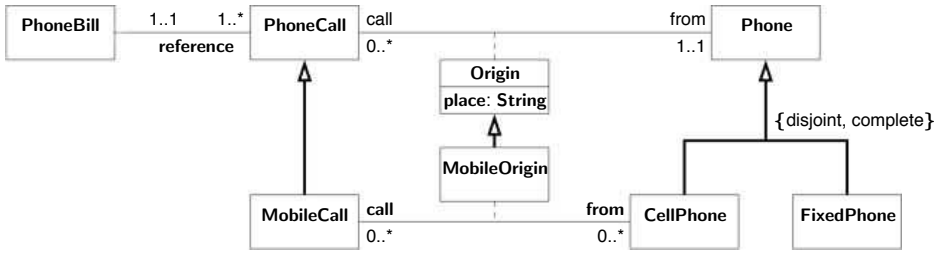
replace

$$\begin{aligned} \exists \mathbf{located} &\sqsubseteq \mathbf{External} \\ \exists \mathbf{located}^- &\sqsubseteq \mathbf{Location} \\ \mathbf{External} &\sqsubseteq (\geq 1\ \mathbf{located}) \sqcap (\leq 1\ \mathbf{located}) \end{aligned}$$

with

$$\begin{aligned} \exists \mathbf{r_i} &\sqsubseteq \mathbf{Located} \qquad i = 1, 2 \\ \exists \mathbf{r_1}^- &\sqsubseteq \mathbf{External} \\ \exists \mathbf{r_2}^- &\sqsubseteq \mathbf{Location} \\ \mathbf{Located} &\sqsubseteq \exists \mathbf{r_1} \sqcap (\leq 1\ \mathbf{r_1}) \sqcap \exists \mathbf{r_2} \sqcap (\leq 1\ \mathbf{r_2}) \\ \mathbf{External} &\sqsubseteq \exists \mathbf{r_1}^- \sqcap (\leq 1\ \mathbf{r_1}^-) \\ \exists \mathbf{nbr\_days} &\sqsubseteq \mathbf{Located} \\ \exists \mathbf{nbr\_days}^- &\sqsubseteq \mathbf{Integer} \\ \mathbf{Located} &\sqsubseteq \exists \mathbf{nbr\_days} \sqcap (\leq 1\ \mathbf{nbr\_days}) \end{aligned}$$

Encoding of UML in DLs – Excercise 3



$$
\begin{array}{rcl}
\exists \text{call} & \sqsubseteq & \text{Origin} \\
\exists \text{call}^- & \sqsubseteq & \text{PhoneCall} \\
\exists \text{from} & \sqsubseteq & \text{Origin} \\
\exists \text{from}^- & \sqsubseteq & \text{Phone} \\
\text{Origin} & \sqsubseteq & \exists \text{call} \sqcap (\leq 1\ \text{call}) \sqcap \exists \text{from} \sqcap (\leq 1\ \text{from}) \\
 & & \\
\text{MobileOrigin} & \sqsubseteq & \text{Origin} \\
\text{MobileOrigin} & \sqsubseteq & \forall \text{call}.\text{MobileCall} \sqcap \forall \text{from}.\text{CellPhone} \\
\text{MobileCall} & \sqsubseteq & \text{PhoneCall} \\
\text{CellPhone} & \sqsubseteq & \text{Phone} \\
\text{FixedPhone} & \sqsubseteq & \text{Phone} \sqcap \neg \text{CellPhone} \\
\text{Phone} & \sqsubseteq & \text{CellPhone} \sqcup \text{FixedPhone} \\
 & & \\
 & & \ldots
\end{array}
$$

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

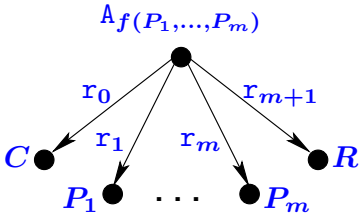
- Operation $f() : R$ without parameters directly represented by an atomic role $\mathtt{P}_{f()}$, with:

$$\exists \mathtt{P}_{f()} \sqsubseteq C \qquad \exists \mathtt{P}_{f()}^- \sqsubseteq R \qquad C \sqsubseteq (\leq 1\, \mathtt{P}_{f()})$$

- Operation $f(P_1, \ldots, P_m) : R$ with one or more parameters can be cannot expressed in $\mathcal{ALCQI}_{id}$ through reification:
  - relation is reified by using a concept $\mathtt{A}_{f(P_1,\ldots,P_m)}$
  - each instance of the concept represents a tuple of the relation
  - (binary) roles $\mathtt{r}_0, \ldots, \mathtt{r}_{m+1}$ connect the object representing a tuple to the objects representing the components of the tuple

Reification of operations

To represent operation $f(P_1, \ldots, P_m) : R$ for class C:



$$A_{f(P_1,\ldots,P_m)} \sqsubseteq \exists r_0 \sqcap \cdots \sqcap \exists r_{m+1} \sqcap (\leq 1\, r_0) \sqcap \cdots \sqcap (\leq 1\, r_{m+1}) \quad (1)$$

$$\exists r_i \sqsubseteq A_{f(P_1,\ldots,P_m)} \qquad \exists r_i^- \sqsubseteq P_i \qquad i = 1, \ldots, m \quad (2)$$

$$\exists r_o \sqsubseteq A_{f(P_1,\ldots,P_m)} \qquad \exists r_i^- \sqsubseteq C \quad (3)$$

$$\exists r_{m+1} \sqsubseteq A_{f(P_1,\ldots,P_m)} \qquad \exists r_{m+1}^- \sqsubseteq R$$

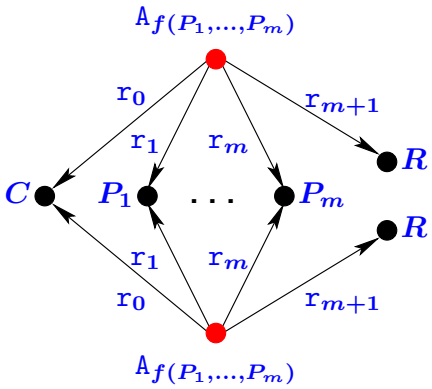
(1) ensures that the instances of $A_{f(P_1,\ldots,P_m)}$ represent tuples

(2) ensures that the parameters of the operation have the correct types

(3) ensures that, when the operation is applied to an instance of C, then the result is an instance of R

Reification of operations (Cont'd)

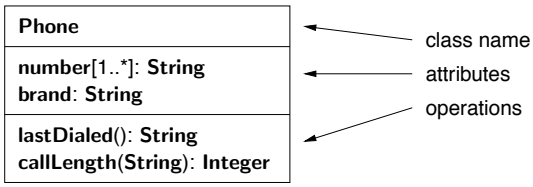
Again, we have not ruled out two instances of $A_{f(P_1,\ldots,P_m)}$ representing two applications of the operation with identical parameters but different result:



To rule out such a situation we could add a key assertion:
$(\mathbf{id}\ A_{f(P_1,\ldots,P_m)}\ r_0, r_1, \ldots, r_m)$

Again, by the tree-model property of $\mathcal{ALCQI}$, we can ignore the key assertion for reasoning

Encoding of operations – Example



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

$$\exists \mathrm{P}_{\mathsf{lastDialed()}} \sqsubseteq \mathbf{Phone}$$

$$\exists \mathrm{P}^{-}_{\mathsf{lastDialed()}} \sqsubseteq \mathbf{String}$$

$$\mathbf{Phone} \sqsubseteq (\leq 1\, \mathrm{P}_{\mathsf{lastDialed()}})$$

$$\mathrm{P}_{\mathsf{callLength(String)}} \sqsubseteq \exists \mathrm{r}_0 \sqcap (\leq 1\, \mathrm{r}_0) \sqcap \exists \mathrm{r}_1 \sqcap (\leq 1\, \mathrm{r}_1) \sqcap \exists \mathrm{r}_2 \sqcap (\leq 1\, \mathrm{r}_2)$$

$$\exists \mathrm{r}_i \sqsubseteq \mathrm{P}_{\mathsf{callLength(String)}} \qquad i = 0, 1, 2$$

$$\exists rc_0^- \sqsubseteq \mathbf{Phone}$$

$$\exists rc_1^- \sqsubseteq \mathbf{String}$$

$$\exists rc_2^- \sqsubseteq \mathbf{Integer}$$

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

Theorem:

A class C is consistent in an UML class diagram $\mathcal{D}$
if and only if
the concept C is satisfiable in the $\mathcal{ALCQI}$ knowledge base $\mathcal{K}_{\mathcal{D}}$ encoding $\mathcal{D}$

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

Conclusions

- We have formalized UML class diagrams in logics, which gives us the ability to reason on them so as to detect and deduce relevant properties
- We have provided an encoding in the DL $\mathcal{ALCQI}$ thus showing that:
  1. Reasoning on UML class diagrams is decidable, and in fact EXPTIME-complete, and thus can be automatized
  2. We can perform such automated reasoning using state-of-the-art DL reasoning systems

The above results lay the foundation for advanced CASE tools with integrated automated reasoning support