Description Logics for Conceptual Data Modeling in UML

Diego Calvanese, Giuseppe De Giacomo

Dipartimento di Informatica e Sistemistica

Università di Roma "La Sapienza"

ESSLLI 2003
Vienna, August 18–22, 2003

Part 2

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 (see ESSLLI'03 course by Horrocks & Sattler next week)
- $\cdots$

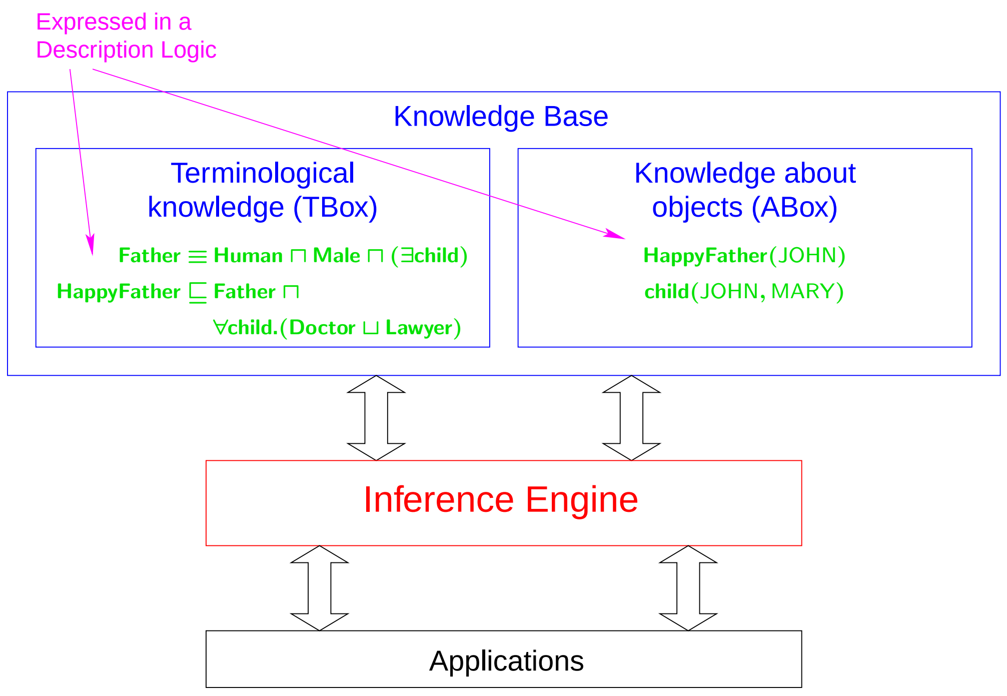
Ingredients of a DL

A Description Logic is characterized by:

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

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


Expressed in a Description Logic
Knowledge Base
Terminological knowledge (TBox)
Father ≡ Human ⊓ Male ⊓ (∃child)
HappyFather ⊑ Father ⊓ ∀child.(Doctor ⊔ Lawyer)
Knowledge about objects (ABox)
HappyFather(JOHN)
child(JOHN, MARY)
Inference Engine
Applications

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	$\neg A$	$\neg$Doctor	$\Delta^\mathcal{I} \setminus A^\mathcal{I}$
conjunction	$C \sqcap D$	Hum $\sqcap$ Male	$C^\mathcal{I} \cap D^\mathcal{I}$
(unqual.) exist. res.	$\exists R$	$\exists$child	$\{\, a \mid \exists b.\, (a,b) \in R^\mathcal{I} \,\}$
value restriction	$\forall R.C$	$\forall$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	$\mathcal{N}$	$(\geq k\, R)$	$\{\, a \mid \#\{b \mid (a,b) \in R^\mathcal{I}\} \geq k \,\}$
restrictions		$(\leq k\, R)$	$\{\, a \mid \#\{b \mid (a,b) \in R^\mathcal{I}\} \leq k \,\}$
qual. number	$\mathcal{Q}$	$(\geq k\, R.C)$	$\{\, a \mid \#\{b \mid (a,b) \in R^\mathcal{I} \wedge b \in C^\mathcal{I}\} \geq k \,\}$
restrictions		$(\leq k\, R.C)$	$\{\, a \mid \#\{b \mid (a,b) \in R^\mathcal{I} \wedge b \in C^\mathcal{I}\} \leq k \,\}$
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
  $\forall\textsf{child}.(\textsf{Doctor} \sqcup \textsf{Lawyer})$
- Qualified existential restriction
  $\exists\textsf{child}.\textsf{Doctor}$
- Full negation
  $\neg(\textsf{Doctor} \sqcup \textsf{Lawyer})$
- Number restrictions
  $(\geq 2\,\textsf{child}) \sqcap (\leq 1\,\textsf{sibling})$
- Qualified number restrictions
  $(\geq 2\,\textsf{child}.\textsf{Doctor}) \sqcap (\leq 1\,\textsf{sibling}.\textsf{Male})$
- Inverse role
  $\forall\textsf{child}^{-}.\textsf{Doctor}$

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:

$$\begin{tikzcd}
& \textsf{Human} & \\
\textsf{Man} \arrow[ur] & & \textsf{Woman} \arrow[ul] \\
\textsf{Father} \arrow[u] & & \\
\textsf{HappyFather} \arrow[u] & &
\end{tikzcd}$$

(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:

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

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 $\mathbf{K}$):

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

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

$\leadsto$ 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)

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

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:

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

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]

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

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

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

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

- a company is identified by its telephone number

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

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

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

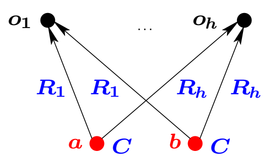
$\mathcal{ALCQI}_{key}$ KBs extend $\mathcal{ALCQI}$ KBs by key assertions:

$$(\text{key } C \mid R_1, \ldots, R_h)$$

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 $R_1, \ldots, R_h$



$$\Longrightarrow \qquad a = b$$

Example: a person is identified by its name and surname

$$(\text{key } \mathsf{Person} \mid \mathsf{name}, \mathsf{surname})$$

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

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

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

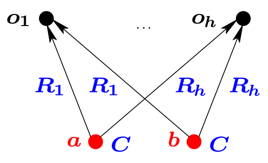
Since logical implication of inclusion assertions and concept satisfiability w.r.t. a KB can be reduced to KB satisfiability, we also have:

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

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

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

To check $\mathcal{K} \cup \mathcal{F} \models (\text{key } C \mid R_1, \ldots R_h)$, 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)

$\leadsto$ Logical implication in $\mathcal{ALCQI}_{key}$ 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)

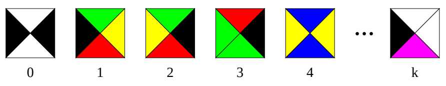
Are such techniques optimal for DL reasoning?
What is the intrinsic complexity of reasoning on DL knowledge bases?

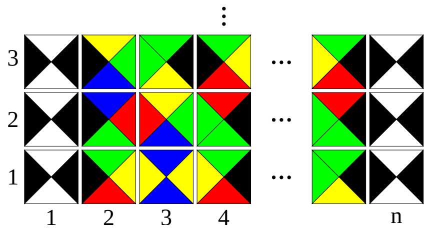
Theorem: Logical implication in $\mathcal{ALC}$ (and hence concept satisfiability w.r.t. an $\mathcal{ALC}$ KB) is EXPTIME-hard

We look at a proof based on encoding the two player corridor tiling problem

EXPTIME-hardness of $\mathcal{ALC}$ (1)

Two player corridor tiling game:

- Tiling system $\mathcal{T}$: finite set of tile types 

- 

A corridor tiling is a tiling of a corridor of width n with tiles of $\mathcal{T}$ respecting adjacency conditions

- ∀lice and ∃lias alternatively place a tile, row by row, from left to right, respecting adjacency conditions
- ∃lias wins if
  - he can place a special “winning tile” in the second position of a row, or
  - he can play in such a way that ∀lice can no longer place a tile

  (i.e., ∃lias loses if he cannot place a tile, or if the game goes on forever)

EXPTIME-hardness of $\mathcal{ALC}$ (2)

Two player corridor tiling problem

Given:

- a tiling system expressed as $\mathcal{T} = (k, H, V)$, where
  - $0, 1, \ldots, k$ are the tile types, with k being the winning tile
  - $H \subseteq [0..k] \times [0..k]$ is the horizontal adjacency relation
  - $V \subseteq [0..k] \times [0..k]$ is the vertical adjacency relation
- an initial row of tiles $t_1 t_2 \cdots t_n$ of length n

does ∃lias have a winning strategy?

I.e., for every move ∀lice makes, is there a move ∃lias can counter with in such a way that he wins?

Two player corridor tiling is EXPTIME-complete

EXPTIME-hardness of $\mathcal{ALC}$ (3)

Encoding of two player corridor tiling in $\mathcal{ALC}$:

- The intention is to represent each placed tile by an object;
  the object carries the information about the last n moves made
- We introduce an atomic role $next$ connecting individuals representing successive tiles
- We introduce the following atomic concepts:
  - R_i^t, one for each $i \in [1..n]$ and each $t \in [0..k]$, denoting that the last tile placed in column i has been tile t
  - Q_i, for $i \in [1..n]$, denoting that the next tile will be placed in column i
  - A, denoting that it is ∀lice's turn to place the next tile
  - W, denoting that ∃lias wins
- We construct an $\mathcal{ALC}$ knowledge base $\mathcal{K}_{\mathcal{T}}$ as follows

EXPTIME-hardness of $\mathcal{ALC}$ (4)

We introduce in $\mathcal{K}_\mathcal{T}$ the following inclusion assertions to ensure that tilings are correctly represented:

- To encode that each column has exactly one tile last placed into it:

$$\begin{array}{rcll} \top & \sqsubseteq & R_i^0 \sqcup \cdots \sqcup R_i^k & \text{for } i \in [1..n] \\ R_i^t & \sqsubseteq & \neg R_i^{t'} & \text{for } i \in [1..n], \quad t, t' \in [0..k], \quad t \neq t' \end{array}$$

- To encode that each move occurs in exactly one column in the corridor:

$$\begin{array}{rcll} \top & \sqsubseteq & Q_1 \sqcup \cdots \sqcup Q_n & \\ Q_i & \sqsubseteq & \neg Q_j & \text{for } i, j \in [1..n], \quad i \neq j \end{array}$$

- To encode that the tiles are placed in the correct left-to-right order:

$$\begin{array}{rcll} Q_i & \sqsubseteq & \forall next.Q_{i+1} & \text{for } i \in [1..n{-}1] \\ Q_n & \sqsubseteq & \forall next.Q_1 & \end{array}$$

EXPTIME-hardness of $\mathcal{ALC}$ (5)

We introduce in $\mathcal{K}_\mathcal{T}$ the following inclusion assertions to encode the adjacency conditions:

- To encode the vertical adjacency relation V:

$$Q_i \sqcap R_i^t \ \sqsubseteq\ \forall next.(\textstyle\bigsqcup_{t' \mid (t,t') \in V} R_i^{t'}) \qquad \text{for } i \in [1..n], \quad t \in [0..k]$$

- To encode the horizontal adjacency relation H:

$$Q_i \sqcap R_{i-1}^t \ \sqsubseteq\ \forall next.(\textstyle\bigsqcup_{t' \mid (t,t') \in H} R_i^{t'}) \qquad \text{for } i \in [2..n], \quad t \in [0..k]$$

- To encode that in columns where no move is made nothing changes:

$$\begin{array}{rcll} \neg Q_i \sqcap R_i^t & \sqsubseteq & \forall next.R_i^t & \text{for } i \in [1..n], \quad t \in [0..k] \\ \neg Q_i \sqcap \neg R_i^t & \sqsubseteq & \forall next.\neg R_i^t & \text{for } i \in [1..n], \quad t \in [0..k] \end{array}$$

EXPTIME-hardness of $\mathcal{ALC}$ (6)

We introduce in $\mathcal{K}_\mathcal{T}$ the following inclusion assertions to encode the game:

- To encode the existence of all possible moves in the game tree, provided ∃lias hasn't already won:

$$\neg R_2^k \sqcap Q_i \sqcap R_{i-1}^t \sqcap R_i^{t'} \quad \sqsubseteq \quad \bigsqcap_{t'' \mid (t,t'')\in H \,\wedge\, (t',t'')\in V} \exists next.R_i^{t''}$$
$$\text{for } i \in [2..n], \quad t,t' \in [0..k]$$

- To encode the alternation of moves:

$$\begin{aligned} A &\sqsubseteq \forall next.\neg A \\ \neg A &\sqsubseteq \forall next.A \end{aligned}$$

- To encode the winning of ∃lias:

$$W \quad \equiv \quad (A \sqcap R_2^k) \sqcup (A \sqcap \forall next.W) \sqcup (\neg A \sqcap \exists next.W)$$

EXPTIME-hardness of $\mathcal{ALC}$ (7)

Observations:

- if ∃lias cannot move when it is his turn, then W is false for the object representing that tile
- if ∀lice can force the game to go on forever, then there will be models of $\mathcal{K}_\mathcal{T}$ in which W is false

Theorem

∃lias has a winning strategy for tiling system $\mathcal{T}$ with initial row $t_1 \cdots t_n$

iff

$$\mathcal{K}_\mathcal{T} \models A \sqcap Q_1 \sqcap R_1^{t_1} \sqcap \cdots \sqcap R_n^{t_n} \sqsubseteq W$$

The size of $\mathcal{K}_\mathcal{T}$ is polynomial in $\mathcal{T}$ and n

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

$\rightsquigarrow$ 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 satisfiable w.r.t. $\mathcal{K}$ iff A_C 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:

$$\begin{array}{lcl} A \sqsubseteq \exists P.A' & \Rightarrow & A \sqsubseteq \exists P_{A'} \sqcap \forall P_{A'}.A' \\ A \sqsubseteq \forall P.A' & \Rightarrow & A \sqsubseteq \forall P.A' \sqcap \bigsqcap_{P_{A_i}} \forall P_{A_i}.A' \end{array}$$

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:

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

The two assertions are logically equivalent

$\leadsto$ 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 $\mathcal{AL}$:

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

$\leadsto$ 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