Description Logics
Old results and new problems

Franz Baader
Theoretical Computer Science
RWTH Aachen
Germany

- Introduction to Description Logics (terminological KR languages, concept languages, KL-ONE-like KR languages, ...)
- Research in DL (historical overview)
- Connection with (simple) conceptual graphs
- New inference problems: unification and matching of concepts
Description logics

- Descend from structured inheritance networks [Brachman 78].
- Tried to overcome ambiguities in semantic networks and frames that were due to their lack of a formal semantics.
- Restriction to a small set of "epistemologically adequate" operators for defining concepts.
- Importance of well-defined basic inference procedures: subsumption and instance problem.
- First realization: system KL-ONE [Brachman & Schmolze 85], many successor systems (Classic, Crack, Fact, Flex, Kris, Loom, ...).
- First application: natural language processing; now also other domains (configuration of technical systems, databases, chemical engineering, medical terminology, ...).
Description Logic Systems

TBox
- defines terminology of the application domain

ABox
- states facts about a specific "world"

reasoning component
- derive implicitly represented knowledge (e.g., subsumption)
- "practical" algorithms

description language
- constructors for building complex concepts and roles out of atomic concepts and roles
- formal, logic-based semantics

knowledge base
Description language

examples of typical constructors:

\[ C \sqcap D, \neg C, \forall r . C, \exists r . C, (\geq n r) \]

A man
that is married to a doctor, and
has at least 5 children,
all of whom are professors.

Human \sqcap \neg Female \sqcap
\exists married-to . Doctor \sqcap
(\geq 5 has-child) \sqcap
\forall has-child . Professor

TBox

definition of concepts

Happy-man = Human \sqcap ...

ABox

properties of individuals

Happy-Man(John)
mARRIED-TO(John,Mary)
Formal semantics

An interpretation $I$ associates

- concepts $C$ with sets $C^I$ and
- roles $r$ with binary relations $r^I$

such that the semantics of the constructors is respected; e.g.,

- $(C \cap D)^I = C^I \cap D^I$
- $(\geq n \ r)^I = \{ d \mid \# \{ e \mid (d,e) \in r^I \} \geq n \}$
- $(\forall r . C)^I = \{ d \mid \forall e: (d,e) \in r^I \Rightarrow e \in C^I \}$
- $\ldots$

$I \models A = C$ iff $A^I = C^I$

$I \models C(a)$ iff $a^I \in C^I$

$I \models r(a,b)$ iff $(a^I,b^I) \in r^I$
Reasoning makes implicitly represented knowledge explicit, is provided as system service by the DL system, e.g.:

**Satisfiability**  Is a concept description C non-contradictory?

- C is satisfiable  iff  there is an I such that $C^I \neq \emptyset$.

**Subsumption**  Is C a subconcept of D?

- $C \sqsubseteq D$  iff  $C^I \subseteq D^I$ for all interpretations I.

**Instantiation**  Is e an instance of C w.r.t. the given ABox $\mathcal{A}$?

- $\mathcal{A} \models C(e)$  iff  $e^I \in C^I$ for all models I of $\mathcal{A}$.
Focus of DL research

- decidability/complexity of reasoning
- requires restricted description languages
- systems and theoretical results available for various combinations of constructors
- application relevant concepts must be definable
- specific application domains may require specific language extensions
- new decidability/complexity results?

Reasoning feasible versus Expressivity sufficient
DL research

historical overview

- **until 1985**: mostly system development;
  expressive description languages, but no disjunction, negation, exist. quant.;
  use of so-called structural subsumption algorithms.

- **1985-1987**: introduction of logic-based semantics;
  first complexity results (NP-hardness) by Levesque and Brachman;
  incompleteness of structural algorithms.

- **1988**: Schmidt-Schauß and Smolka describe the first complete (tableau-based) subsumption algorithm for a non-trivial language;
  $\mathcal{ALC}$: propositionally closed (disjunction, negation, existential restrictions);
  subsumption as logical inference problem, reduced to satisfiability;
  complexity result: subsumption in $\mathcal{ALC}$ is PSPACE-complete.
1989: three papers [Patel-Schneider; Schild; Schmidt-Schauß] show undecidability of subsumption for description languages used in implemented DL systems.

since 1989: development of tableau-based algorithms for a great variety of description languages (DFKI, Germany; University of Rome I, RWTH Aachen, ...); extended to the instance problem for ABoxes.

1989 - 1991: exact worst-case complexity of satisfiability and subsumption for various description languages (DFKI, Germany; University of Rome I).

1991: Schild notices a close connection between DLs and modal logics; ALC is just a syntactic variant of propositional multi-modal K; algorithms, complexity results from modal logics carry over.

1992-1995: development of very expressive Description Logics based on decidable extensions of K (University of Rome I); e.g., used to express semantic data models (ER, OO, ...).
1991-1998: close connection between DLs and decidable sub-classes of first-order logic:

- **ALC** can be expressed within L2, i.e., first-order logic with two variables:
  - decidable [Mortimer 75], NEXPTIME-complete [Grädel, V., K. 97]
- **number restrictions** can be expressed in C2, i.e., the extension of L2 by counting quantifiers:
  - decidable [Grädel, O., R. 97], in 2-NEXPTIME [Pacholski, S., T. 97]

1992-1998: optimization of DL systems based on complete (tableau-like) algorithms:

- try to avoid explicit calls of subsumption algorithm during classification [Baader et. al 92, 94]; similar to techniques employed in CG systems [Ellis 91; Levinson 92].
- optimization of subsumption algorithms [Giunchiglia & Sebastiani 96, 98; Horrocks 98; Patel-Schneider 98].
Connection with Conceptual Graphs

- Conceptual graphs have the "full power of first-order logic" [Sowa 84]. Thus, most of the description languages considered in DL can be expressed.

- What about reasoning? Does this connection provide us with graph-based reasoning methods for DL?
  - Possible way of finding and/or explaining incomplete (subsumption) algorithms?
  - Sub-class of CGs for which graph-based methods yield decision procedures, and thus complete algorithms?
Simple Conceptual Graphs

can express concept descriptions built using conjunction (\( \sqcap \)) and existential restriction (\( \exists r . C \)).

\[
\text{Person } \sqcap \\
\exists \text{ has-child} . \text{Male } \sqcap \\
\exists \text{ has-child} . \text{Female}
\]

\[\text{Person:dummy} \quad \text{has-child} \quad \text{Male:*} \]
\[\text{Person:dummy} \quad \text{has-child} \quad \text{Female:*} \]

\[\Rightarrow \text{ Subsumption of descriptions corresponds to subsumption of SGs.}\]

\[\Rightarrow \text{ Subsumption of SGs characterized by existence of projection.}\]

\[\Rightarrow \text{ Testing for existence of projection is NP-complete.}\]

\[\Rightarrow \text{ Subsumption of descriptions is polynomial since translation yields SGs that are trees.}\]
New inference problems*

- Until recently, DL research concentrated on the traditional inference problems subsumption and instantiation.
- Building and maintaining larger knowledge bases requires support by new types of inference methods, e.g.:
  - unification of concepts: detect redundancies in KB
  - matching of concepts: prune large concept descriptions before printing them
- In the rest of the talk:
  - unification and matching in the simple DL $\mathcal{FL}_0$:
    - conjunction $C \sqcap D$, value restriction $\forall r . C$
  - extension to larger languages

* Joint work with A. Borgida, R. Küsters, D. McGuinness, P. Narendran
Situation

very large terminology is built by several knowledge engineers over a long time period (our application: process engineering)

Testing for equivalence is not sufficient to find out whether two concept descriptions describe the same concept: different knowledge engineers

- introduce different concept names for the same (intuitive) concept:
  - Masculine instead of Male

- model on different levels of granularity:
  - Man as atomic concept name
  - Human ⊑ Male as a concept term expressing the same concept
  - Human ⊑ Male ⊑ ∀drinks. Beer Bavarian knowledge engineer
Set of concept names is partitioned into concept variables and concept constants:

- concept patterns may contain variables
- concept descriptions not

- substitution replaces concept variables by concept descriptions
- unifier of two concept patterns C and D: substitution $\sigma$ such that

$$\sigma(C) \equiv \sigma(D)$$

i.e., $\sigma(C)^I = \sigma(D)^I$ for all interpretations $I$. 

Unification of concepts

Definition
Example

Might the following concept descriptions denote the same concept?

\( \forall \text{child} \cdot \forall \text{child} \cdot \text{Rich} \sqsubseteq \forall \text{child} \cdot \text{RMR} \)

\( \text{RMR} \leftrightarrow \sqsubseteq \forall \text{spouse} \cdot \text{Rich} \)

\( \forall \text{child} \cdot \forall \text{child} \cdot \text{Rich} \sqsubseteq \forall \text{child} \cdot (\text{Rich} \sqsubseteq \forall \text{spouse} \cdot \text{Rich}) \)

All grandchildren are rich and all children are rich and married rich.

\( \text{ACR} \leftrightarrow \forall \text{child} \cdot \text{Rich} \)

\( \text{ACR} \sqsubseteq \forall \text{child} \cdot \text{ACR} \sqsubseteq \forall \text{child} \cdot \forall \text{spouse} \cdot \text{Rich} \)
unification of $\mathcal{FL}_0$ concept patterns

direct translation possible

solving linear equations in semiring

semiring elements represented by finite trees

axiomatization of equivalence

results from unification theory

[Baader 89, Nutt 90, Baader&Nutt 91]

unification modulo equational theory

emptiness problem for tree automata

Problem reduction

ACUIh

Problem reduction

ACUIh

Problem reduction

ACUIh

Problem reduction

ACUIh

Problem reduction

ACUIh
The semiring corresponding to $\mathcal{FL}_0$

**Elements:** finite sets of words over alphabet of role names
   e.g., $\emptyset$, {$c, cs, ccs$}, {$s$}, ...

**Addition:** set union
   \[ \{c, cs, ccs\} \cup \{s\} = \{c, cs, ccs, s\} \]

**Multiplication:** element-wise concatenation
   \[ \{s\}\{c, cs, ccs\} = \{sc, scs, sccs\} \]

**Linear equations**
   $S_i, T_i$ coefficients, $X_i$ variables

   \[ S_0 \cup S_1 X_1 \cup \ldots \cup S_n X_n = T_0 \cup T_1 X_1 \cup \ldots \cup T_n X_n \]
unification of $\mathcal{FL}_0$ concept patterns

direct translation:

- Normal form for concept descriptions and patterns
- Characterization of equivalence of concept descriptions in normal form
- Translate this characterization into linear equations

solving linear equations in semiring
In $\mathcal{FL}_0$, equality of value-restriction sets characterizes equivalence.

Value-restriction sets are finite sets of words over the alphabet of role names, i.e., elements of the semiring.
Translation of unification problem into linear equations over finite sets of words

\[ C \equiv \forall K_1 . A_1 \cap \ldots \cap \forall K_n . A_n \cap \forall L_1 . X_1 \cap \ldots \cap \forall L_k . X_k \]

\[ D \equiv \forall M_1 . A_1 \cap \ldots \cap \forall M_n . A_n \cap \forall N_1 . X_1 \cap \ldots \cap \forall N_k . X_k \]

Equation \((A_i)\)

\[ K_i \cup L_1 X_{1,i} \cup \ldots \cup L_k X_{k,i} = M_i \cup N_1 X_{1,i} \cup \ldots \cup N_k X_{k,i} \]

Theorem

The unification problem \(C \equiv^? D\) is solvable \iff the formal language equations \((A_1), \ldots, (A_n)\) are each solvable.
unification of concept patterns

∀c.∀c.R ∩ ∀c.X \equiv \exists Y \ni ∀c.Y \ni ∀c.∀s.R

linear equation (R)

{cc} \cup \{c\}X = \{cs\} \cup \{ε, c\}Y

solution

X = \{ε, s\}, Y = \{c\}

yields solution set \{cc, c, cs\}
Finite sets of words over an n-element alphabet can be represented by n-ary finite trees:

\[
\{cc\} \cup X\{c\} = \{sc\} \cup Y\{\varepsilon, c\}
\]

Top-down tree automaton tests for existence of solution set:

- "guesses" the elements of the variables \(X_i\)
- makes sure that the concatenation with the coefficients is realized
Theorem [Baader\&Narendran ECAI'98]

Unification of $\mathcal{FL}_0$-concept patterns is decidable.

Complexity

- Reduction to tree automata yields EXPTIME decision procedure:
  - size of tree automaton exponential in size of system of equations
  - emptiness problem for tree automata is polynomial
- Decision problem is EXPTIME-hard:
  - emptiness of intersection of $m$ deterministic top down tree automata used for reduction
First results for matching in DL

- Matching modulo subsumption [Borgida & McGuinness, KR'96]
  - DL containing most of the CLASSIC constructs
  - Polynomial matching algorithm
  - Restriction on the syntactic form of patterns

- Matching modulo equivalence [Baader & Narendran, ECAI'98]
  - As special case of unification in $\mathcal{FL}_0$
  - Unlike unification, matching is polynomial for $\mathcal{FL}_0$
  - No restriction on the syntactic form of patterns
Extension of results to larger description language allowing for \( \bot \), atomic negation, number restrictions

- Reduction of unification and matching problems to (extended) linear equations over finite sets of words still possible.
  - Main technical problem: appropriate treatment of inconsistency in the characterization of equivalence of concept descriptions.
- How to test the resulting linear equations for solvability?
  - Unification: open problem even for \( \mathcal{FL}_0 + \bot \).
    The approach based on tree automata cannot work!
  - Matching: polynomial for \( \mathcal{FL}_0 + \bot + \text{atomic negation} + \text{number restrictions} \).
    Idea: compute largest "solution candidate" and test whether it is a solution.
Concept-centered NF in $\mathcal{FL}_0 + \bot$

\[
\forall r. \bot \sqcap \forall r. (B \sqcap \forall r. A) \quad \forall r. (B \sqcap \forall r. B) \sqcap \forall r. \bot
\]

- Distribute $\forall r$ over conjunction

\[
\forall r. \bot \sqcap \forall r. B \sqcap \forall r. \forall r. A \quad \forall r. B \sqcap \forall r. \forall r. B \sqcap \forall r. \bot
\]

- Collect value-restriction set for each concept

\[
\forall \{r\}. \bot \sqcap \forall \{rr\}. A \sqcap \forall \{r\}. B \quad \forall \{r\}. \bot \sqcap \forall \emptyset. A \sqcap \forall \{r, rr\}. B
\]

In contrast to the situation for $\mathcal{FL}_0$, equality of value-restriction sets is no longer sufficient to characterize equivalence.
Concept-centered NF characterization of equivalence in $\mathcal{FL}_0 + \bot$

$C \equiv \forall L_0 . \bot \cap \forall L_1 . A_1 \cap \ldots \cap \forall L_n . A_n$

$D \equiv \forall M_0 . \bot \cap \forall M_1 . A_1 \cap \ldots \cap \forall M_n . A_n$

$L_i, M_i$ finite sets of words over the alphabet $\Sigma$ of role names

Theorem

$C \equiv D$ iff $L_0 \Sigma^* = M_0 \Sigma^*$ and for $i = 1, \ldots, n$

$L_i \cup L_0 \Sigma^* = M_i \cup M_0 \Sigma^*$
Translation of matching problem into linear equations over finite sets of words

\[ C \equiv \forall L_0. \bot \cap \forall L_1. A_1 \cap \ldots \cap \forall L_n. A_n \]

\[ D \equiv \forall M_0. \bot \cap \forall M_1. A_1 \cap \ldots \cap \forall M_n. A_n \cap \forall N_1. X_1 \cap \ldots \cap \forall N_k. X_k \]

Equation (⊥)

\[ L_0 \Sigma^* = M_0 \Sigma^* \cup N_1 X_{1,0} \Sigma^* \cup \ldots \cup N_k X_{k,0} \Sigma^* \]

Equation (A_i)

\[ L_i \cup L_0 \Sigma^* = M_i \cup N_1 X_{1,i} \cup \ldots \cup N_k X_{k,i} \cup L_0 \Sigma^* \]

X_{i,j} variables for finite sets of words
Theorem

The matching problem $C \equiv ? D$ is solvable iff

the formal language equations ($\perp$), ($A_1$), ..., ($A_n$) are each solvable.

How to test solvability of ($\perp$), ($A_1$), ..., ($A_n$) ?

⇒ Compute largest "solution candidate".

⇒ Test whether this candidate is indeed a solution.

⇒ Both steps only require "easy" computations on finite sets of words (polynomial).
Conclusion

- Standard inference problems (subsumption, instantiation) well-investigated.
  - Decidability and complexity results for a great variety of description languages, including very expressive ones.
  - Efficient implementations of decision procedures available.

- Research on non-standard inference problems (unification, matching, ...) is just beginning:
  - Unification: decidability result only for the small language $\mathcal{FL}_0$; high complexity; unification in larger languages might be easier!?
  - Matching: polynomial for language that is expressive enough for applications.
  - Other applications for matching and/or unification, e.g., integration of heterogeneous databases?