Description
Logics
What Are Description Logics?

- A family of logic based KR formalisms
  - Descendants of semantic networks and KL-ONE
  - Describe domain in terms of concepts (classes), roles (relationships) and individuals

- Distinguished by:
  - Formal semantics (typically model theoretic)
    - Decidable fragments of FOL
    - Closely related to Propositional Modal & Dynamic Logics
  - Provision of inference services
    - Sound and complete decision procedures for key problems
    - Implemented systems (highly optimized)
Description Logics

- Major focus of KR research in the 1980’s
  - Led by Ron Brachman (AT&T Labs)
  - Grew out of early network-based KR systems like semantic networks and frames

- Major systems and languages
  - 80s: KL-ONE, NIKL, KANDOR, BACK, CLASSIC, LOOM
  - 90s: FACT, RACER, ...
  - 00s: DAML+OIL, OWL, Pellet, Jena, FACT++, ...
  - 10s: HermiT, ELK, ...

- Basis for semantic web language OWL
DL Paradigm

- **Description Logic** characterized by a set of constructors that allow one to build complex *descriptions* or *terms* out of *concepts* and *roles* from atomic ones
  - **Concepts**: classes interpreted as sets of objects,
  - **Roles**: relations interpreted as binary relations on objects
- Set of axioms for asserting **facts** about concepts, roles and **individuals**
Definitions of Terminology

Assertions about individuals

Division into TBox and ABox has no logical significance, but is made for conceptual and implementation convenience.
DL defines a family of languages

- The expressiveness of a description logic is determined by the **operators** that it uses
  - Adding or removing operators (e.g., $\neg$, $\cup$) increases or decreases the kinds of statements expressible
  - Higher expressiveness usually means higher reasoning complexity
- **AL** or **Attributive Language** is the base and includes just a few operators
- Other DLs are described by the additional operators they include
### AL: Attributive Language

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic concept</td>
<td>C</td>
<td>Human</td>
</tr>
<tr>
<td><em>atomic</em> negation</td>
<td>~ C</td>
<td>~ Human</td>
</tr>
<tr>
<td>atomic role</td>
<td>R</td>
<td>hasChild</td>
</tr>
<tr>
<td>conjunction</td>
<td>C ∧ D</td>
<td>Human ∧ Male</td>
</tr>
<tr>
<td>value restriction</td>
<td>R.C</td>
<td>Human ∃ hasChild.Blond</td>
</tr>
<tr>
<td>existential rest. (lim)</td>
<td>∃ R</td>
<td>Human ∃ hasChild</td>
</tr>
<tr>
<td>Top (univ. conc.)</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>bottom (null conc.)</td>
<td>⊥</td>
<td>⊥</td>
</tr>
</tbody>
</table>

*for concepts C and D and role R*
**ALC**

**ALC** is the smallest DL that is propositionally closed (i.e., includes full negation and disjunction) and include booleans (and, or, not) and restrictions on role values.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic concept</td>
<td>C</td>
<td>Human</td>
</tr>
<tr>
<td>negation</td>
<td>~ C</td>
<td>~ (Human V Ape)</td>
</tr>
<tr>
<td>atomic role</td>
<td>R</td>
<td>hasChild</td>
</tr>
<tr>
<td>conjunction</td>
<td>C ^ D</td>
<td>Human ^ Male</td>
</tr>
<tr>
<td>disjunction</td>
<td>C V D</td>
<td>Nice V Rich</td>
</tr>
<tr>
<td>value restrict.</td>
<td>∃ R.C</td>
<td>Human ∃ hasChild.Blond</td>
</tr>
<tr>
<td>existential restrict.</td>
<td>∃ R.C</td>
<td>Human ∃ hasChild.Male</td>
</tr>
<tr>
<td>Top (univ. conc.)</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>bottom (null conc)</td>
<td>⊥</td>
<td>⊥</td>
</tr>
</tbody>
</table>
Examples of ALC concepts

- **Person** ∧ ∀ hasChild.Male (everybody whose children are all male)
- **Person** ∧ ∀ hasChild.Male ∧ ∃ hasChild.T (everybody who has a child and whose children are all male)
- **Living_being** ∧ ¬ Human_being (all living beings that are not human beings)
- **Student** ∧ ¬ ∃ interested in. Mathematics (all students not interested in mathematics)
- **Student** ∧ ∀ drinks.tea (all students who only drink tea)
- ∃ hasChild.Male V ∀ hasChild. ⊥ (everybody who has a son or no child)
## Other Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number restriction</td>
<td>$\geq n , R$</td>
<td>$\geq 7 \text{ hasChild}$</td>
</tr>
<tr>
<td></td>
<td>$\leq n , R$</td>
<td>$\leq 1 \text{ hasmother}$</td>
</tr>
<tr>
<td>Inverse role</td>
<td>R-</td>
<td>haschild-</td>
</tr>
<tr>
<td>Transitive role</td>
<td>R*</td>
<td>hasChild*</td>
</tr>
<tr>
<td>Role composition</td>
<td>R $\circ$ R</td>
<td>hasParent $\circ$ hasBrother</td>
</tr>
<tr>
<td>Qualified # restric.</td>
<td>$\geq n , R.C$</td>
<td>$\geq 2 \text{ hasChild.Female}$</td>
</tr>
<tr>
<td>Singleton concepts</td>
<td>${&lt;\text{name}&gt;}$</td>
<td>${\text{Italy}}$</td>
</tr>
</tbody>
</table>
Special names and combinations


- **S** = ALC + transitive properties
- **H** = role hierarchy, e.g., rdfs:subPropertyOf
- **O** = nominals, e.g., values constrained by enumerated classes, as in owl:oneOf and owl:hasValue
- **I** = inverse properties
- **N** = cardinality restrictions (owl:cardinality, maxCardonality)
- **(D)** = use of datatypes properties
- **R** = complex role axioms (e.g. (ir)reflexivity, disjointedness)
- **Q** = Qualified cardinality (e.g., at least two female children)

→ **OWL-DL** is **SHOIN**(D)

→ **OWL 2** is **SROIQ**(D)

Note: **R->H** and **Q->N**
Complexity of reasoning in Description Logics

Note: the information here is (always) incomplete and updated often

Base description logic: Attributive Language with Complements

\[ ALC ::= \bot \cup T \cup A \cup \neg C \cup C \cap D \cup C \cup D \cup \exists R.C \cup \forall R.C \]

Concept constructors:

- \( \neg \) – functionality: \((\leq 1 \ R)\)
- \( \forall \) – (unqualified) number restrictions: \((\geq n \ R), (\leq n \ R)\)
- \( \exists \) – qualified number restrictions: \((\exists n \ R.C), (\leq n \ R.C)\)
- \( O \) – nominals: \(\{a\}\) or \(\{a_1, \ldots, a_n\}\) ("one-of")
- \( \mu \) – least fixpoint operator: \(\mu X.C\)
- \( X \) – complex roles in number restrictions

Role constructors:

- \( R^{-} \) – role inverse
- \( \cap \) – role intersection
- \( \cup \) – role union
- \( \neg \) – role complement
- \( o \) – role chain (composition)
- \( * \) – reflexive-transitive closure
- \( id \) – concept identity

TBox (concept axioms):

- empty TBox
- acyclic TBox: \(A \equiv C\) if \(A\) is a concept name; no cycles
- general TBox: \(C \subseteq D\) for arbitrary concepts \(C\) and \(D\)

RBox (role axioms):

- \( s^{-} \) – role transitivity
- \( H^{-} \) – role hierarchy
- \( \forall \) – complex role inclusions: \(R \circ S \subseteq R, R \circ S \subseteq S\)
- \( s \) – some additional features

You have selected a Description Logic: \( ALC \)

Complexity of reasoning problems:

<table>
<thead>
<tr>
<th>Reasoning problem</th>
<th>Complexity</th>
<th>Comments and references</th>
</tr>
</thead>
</table>
| Concept satisfiability  | PSpace-complete | • Hardness for \( ALC \): see [80].  
                           |             | • Upper bound for \( ALCQ \): see [12, Theorem 4.6]. |
| ABox consistency        | PSpace-complete | • Hardness follows from that for concept satisfiability.  
                           |             | • Upper bound for \( ALCQO \): see [17, Appendix A]. |

Important properties of the description logic

<table>
<thead>
<tr>
<th>Property</th>
<th>Validity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finite model property</td>
<td>Yes</td>
</tr>
<tr>
<td>Tree model property</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Maintained by: Evgeny Zolin
Please see the list of updates
Any comments are welcome: eZolin@cs.man.ac.uk

Notes:

1. The letters \( Q, I \), and \( \Omega \) are customary written in various orders, e.g., \( ALCQI \), and \( SIQO \). Here we do not reflect this tradition, but rather use a uniform naming scheme.
OWL as a DL

- OWL-DL is SHOIN$^D$
- We can think of OWL as having three kinds of statements
  - Ways to specify classes
    - the intersection of humans and males
  - Ways to state axioms about those classes
    - Humans are a subclass of apes
  - Ways to talk about individuals
    - John is a human, john is a male, john has a child mary
Subsumption: $D \subseteq C$?

- Concept $C$ subsumes $D$ iff on every interpretation $I$
  \[ I(D) \subseteq I(C) \]
- This means the same as $\forall(x)(D(x) \rightarrow C(x))$ for complex statements $D$ & $C$
- Determining whether one concept *logically* contains another is called the *subsumption problem*.
- Subsumption is *undecidable* for reasonably expressive languages
  - e.g.; for FOL: does one FOL sentence imply another
- and non-polynomial for fairly restricted ones
Other reasoning problems

These problems can be reduced to subsumption (for languages with negation) and to the satisfiability problem

- **Concept satisfiability** is $C$ (necessarily) empty?
- **Instance Checking** $\text{Father(john)}$?
- **Equivalence** $\text{CreatureWithHeart} \equiv \text{CreatureWithKidney}$
- **Disjointness** $C \cap D$
- **Retrieval** $\text{Father}(X)$? $X = \{\text{john, robert}\}$
- **Realization** $X(\text{john})$? $X = \{\text{Father}\}$
A **definition** is a description of a concept or a relationship.

It is used to assign a meaning to a term.

In description logics, definitions use a specialized logical language.

Description logics are able to do limited reasoning about concepts defined in their logic.

One important inference is **classification** (computation of subsumption).
Necessary vs. Sufficient

- **Necessary** properties of an object are common to all objects of that type
  - Being a man is a **necessary** condition for being a father

- **Sufficient** properties allow one to identify an object as belonging to a type and need not be common to all members of the type
  - Speeding is a **sufficient** reason for being stopped by the police

- **Definitions** typically specify both **necessary** and **sufficient** properties
Subsumption

• Meaning of Subsumption

A more general concept or description **subsumes** a more specific one. Members of a subsumed concept are necessarily members of a subsuming concept

• Two ways to formalize meaning of subsumption
  - Using logic: satisfying a subsumed concept implies that the subsuming concept is satisfied also
    E.g., if john is a person, he is also an animal
  - Using set theory: instances of subsumed concept are necessarily a subset of subsuming concept’s instances
    E.g., the set of all persons is a subset of all animals
How Does Classification Work?

A sick animal is defined as something that is both an animal and has at least one thing that is a kind of a disease.
A rabid dog is defined as something that is both a dog and has at least one thing that is a kind of a rabies
We can easily prove that a rabid dog is a kind of sick animal.
Defining “rabid animal”

A rabid animal is defined as something that is both an animal and has at least one thing that is a kind of a rabies disease.
DL reasoners places concepts in hierarchy

Note: we can remove the subclass link from rabid animal to animal because it is redundant. We don’t need to. But humans like to see the simplest structure and it may be informative for agents as well.

We can easily prove that a rabid dog is a kind of rabid animal
Primitive versus Structured (Defined)

- Description logics reason with definitions
  - They prefer to have *complete* descriptions
  - A complete definition includes both necessary conditions and sufficient conditions

- Often impractical or impossible, especially with **natural kinds**

- A “primitive” definition is an incomplete one
  - Limits amount of classification that can be done automatically

- Example:
  - Primitive: a Person
  - Defined: Parent = Person with at least one child
Classification is a powerful kind of reasoning that is very useful

Many expert systems can be usefully thought of as doing “heuristic classification”

Logical classification over structured descriptions and individuals is also quite useful

But... can classification ever deduce something about an individual other than what classes it belongs to?

And what does *that* tell us?
Incidental properties

If we allow incidental properties (e.g., ones that don’t participate in the definitions mechanism) then these can be deduced via classification

- E.g., red cars have been observed to have a high accident rate by insurance companies
- Birds weighing more than 25kg cannot fly