Description

Logics
What is **Description Logic**?

- 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*) based on a *decidable* fragments of FOL
  - Provision of inference services
    - Sound and complete decision procedures for key problems
    - Implemented systems (highly optimized)

- Formal basis for OWL (DL profile)
Informally, What is **Description Logic**?

- We define a concept using a simple noun phrase in a human language like English
  - A red car
  - A tall person who works for IBM
  - A tall person who works for a Bay-area Technology company

- E.g., we don’t do this, using a set of rules

- Natural languages have multiple ways of attaching modifiers to a simple concept
  - E.g. adjectives, propositional phrases, clausal modifiers, connectives (and, or, not)

- Description logics, like OWL-DL, designed to define concepts in a similar way
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
father = man ∏ E has.child X
human = mammal ∏ biped
...

john = human ∏ father
john has.child mary

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., ¬, ∪) 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** $\land$ $\forall$hasChild.Male (everybody whose children are all male)
- **Person** $\land$ $\forall$hasChild.Male $\land$ $\exists$hasChild.T (everybody who has a child and whose children are all male)
- **Living_being** $\land$ $\neg$Human_being (all living beings that are not human beings)
- **Student** $\land$ $\neg$\existsinterestedIn.Mathematics (all students not interested in mathematics)
- **Student** $\land$ $\forall$drinks.tea (all students who only drink tea)
- $\exists$hasChild.Male $\lor$ $\forall$hasChild.⊥ (everybody who has a son or no child)
Other Constructors

The general DL model has additional 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


- \( S = \text{ALC} + \text{transitive properties} \)
- \( H = \text{role hierarchy}, \text{e.g.,} \ rdfs:subPropertyOf \)
- \( O = \text{nominals}, \text{e.g.,} \ \text{values constrained by enumerated classes} \) (e.g., days of week), as in owl:oneOf and owl:hasValue
- \( I = \text{inverse properties} \)
- \( N = \text{cardinality restrictions} \) (owl:cardinality, maxCardinality)
- \( (D) = \text{use of datatypes properties} \)
- \( R = \text{complex role axioms} \) (e.g. (ir)reflexivity, disjointedness)
- \( Q = \text{Qualified cardinality} \) (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**

\[ ALC ::= ⊤ \mid \bot \mid A \mid ¬C \mid C_1 \land C_2 \mid C_1 \lor C_2 \mid ∃R.C \mid ∀R.C \]

#### Concept constructors:
- \( F \) – functionality: \( (≤1 \ R) \)
- \( \mathcal{N} \) – (unqualified) number restrictions: \( (≥n \ R), (≤n \ R) \)
- \( Q \) – qualified number restrictions: \( (≥n \ R.C), (≤n \ R.C) \)
- \( O \) – nominals: \( \{a\} \) or \( \{a_1, ..., a_n\} \) ("one-of")
- \( μ \) – least fixpoint operator: \( μX.C \)

#### Role constructors:
- \( f \) – role inverse: \( R^- \)
- \( ∩ \) – role intersection: \( R \land S \)
- \( \cup \) – role union: \( R \lor S \)
- \( ¬ \) – role complement: \( ¬R \)
- \( o \) – role chain (composition): \( R \circ S \)
- \( * \) – reflexive-transitive closure: \( R^* \)
- \( id \) – concept identity: \( id(C) \)

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

### RBox (role axioms):
- \( s \) – role transitivity: \( Tr(R) \)
- \( R \) – role hierarchy: \( R \sqsubseteq S \)
- \( R \) – complex role inclusions: \( R \circ S \sqsubseteq R, R \circ S \sqsubseteq S \)
- \( s \) – some additional features (check it to see)

---

**Complexity of reasoning problems**

<table>
<thead>
<tr>
<th>Reasoning problem</th>
<th>Complexity</th>
<th>Comments and references</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept satisfiability</td>
<td>PSpace-complete</td>
<td>- Hardness for ( ALC ), see [80].&lt;br&gt;- Upper bound for ( ALCQ ), see [12, Theorem 4.6].</td>
</tr>
<tr>
<td>ABox consistency</td>
<td>PSpace-complete</td>
<td>- Hardness follows from that for concept satisfiability.&lt;br&gt;- Upper bound for ( ALCQ ), see [17, Appendix A].</td>
</tr>
</tbody>
</table>

**Important properties of the description logic**

- **Finite model property**
  - Yes
  - \( ALC \) is a notational variant of the multi-modal logic \( K_m \) (cf. [77]), for which the finite model property can be found in [4, Sect. 2.3].

- **Tree model property**
  - Yes
  - \( ALC \) is a notational variant of the multi-modal logic \( K_m \) (cf. [77]), for which the tree model property can be found in [4, Proposition 2.15].

---

**Maintained by:** Evgeny Zolin

Please see the list of updates.

Any comments are welcome: tZolin@cs.man.ac.uk

---

**Notes:**

1. The letters \( O, L \) and \( Q \) are customary written in various orders, e.g. \( ALCQLO \) but \( SQLOQ \). Here we do not reflect this tradition, but rather use a uniform naming scheme.

---

http://www.cs.man.ac.uk/~ezolin/dl/
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, a male, and 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, subsumption means “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**  
  \[ \text{Father}(\text{john})? \]
- **Equivalence**  
  \[ \text{CreatureWithHeart} \equiv \text{CreatureWithKidney} \]
- **Disjointness**  
  \[ C \cap D \]
- **Retrieval**  
  \[ \text{Father}(X)? \quad X = \{\text{john, robert}\} \]
- **Realization**  
  \[ X(\text{john})? \quad X = \{\text{Father}\} \]
Definitions

• 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 (but there are others!)
- **Definitions** typically specify both **necessary and sufficient** properties
Subsumption

● Meaning of Subsumption
  
  A more general concept/description subsumes a more specific one. Members of a subsumed concept are necessarily members of a subsuming concept

● Example: Animal subsumes Person; (aka IS-A, rdfs:subClassOf)

● 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.
The rabid dog concept is defined as something that is both a dog and has rabies.
We can easily prove that a rabid dog is a kind of sick animal.
Defining “rabid animal”

The rabid animal concept is defined as something that is both an animal and has rabies.
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 s 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 very useful

- 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?
Example: Blood Pressure

A Non-Critical Blood Pressure is “a Systolic B.P. between 85 and 160.”
Example: Blood Pressure

Normal Systolic B.P. is “a Systolic B.P. between 90 and 140.”
If Joe’s BP is Normal is it also Non-Critical?
Non-Critical Systolic BP

Systolic Blood Pressure

Normal Systolic BP

- \( \geq 90 \)
- \( \geq 85 \)
- \( \leq 140 \)
- \( \leq 160 \)

Concept Classification Infers Normal BP is Subsumed by Non-Critical BP
Systolic Blood Pressure

- pressure $\geq 90$
- pressure $\geq 85$
- pressure $\leq 140$
- pressure $\leq 160$

With Classified Concepts the Answer is Easy to Compute
We can consider properties that are not part of any definition to be incidental.

Classification based on non-incidental properties allow the inference of incidental properties.

Examples:

- E.g., **red cars** have been observed to have a high accident rate by insurance companies.
- **Birds weighing more than 25kg** can not fly.
- People with **non-critical blood pressure** require no medication.
Description logic was the model for OWL reasoning

More expressive than rule-based systems without being undecidable or intractable

It can reason over general statements (e.g., a dog with rabies is a sick animal), unlike most rule-based systems

It still has limitations tho...