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 80’s
  - Led by Ron Brachman – (AT&T Labs)

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

• Used as the basis for the Semantic web languages DAML+OIL and OWL

• Some commercial systems
Description Logics

Thought to be well-suited for the representation of and reasoning about

- ontologies
- terminological knowledge
- Configurations and configuration problems
- database schemata
  - schema design, evolution, and query optimization
  - source integration in heterogeneous databases/data warehouses
  - conceptual modeling of multidimensional aggregation
Example of Network KR

- Person, Female, etc. are concepts
- hasChild is a property of Person
  - hasChild relates Parent to Person
  - Nil means infinity. A Parent is a Person with between 1 and infinity children
- Large arrows are “IS-A” links
  - A Mother is a (specialization of a) Parent
- Concepts either primitive or definitions
  - Primitive concepts have only necessary properties
  - Defined concepts have necessary and sufficient conditions

Graphical notation introduced by KL-ONE
**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 often specify both necessary and sufficient properties
A **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** correspond to classes
  - and are interpreted as sets of objects,
- **Roles** correspond to relations
  - and are interpreted as binary relations on objects

**Set of axioms for asserting facts** about concepts, roles and **individuals**
Basic Concepts of a DL

- Individuals are treated exactly the same as constants in FOL
  - \textit{john}

- Concepts are exactly the same as Unary Predicates in FOL
  - \textit{Person(john)}

- Roles are exactly the same as Binary Predicates in FOL
  - \textit{has\_mother(john, mary)}
Like FOL, we are dealing with (ultimately) sets of individuals and relations between them.

Basic unit of semantic significance is a Description.

"We are describing sets of individuals."

Description logics differ in allowed operators.

If a "happy father" is a man with both a son and daughter and all of whose children are either rich or happy, we describe it in DL as:

\[
\text{HappyFather} = \text{Man} \cap \exists \text{hasChild.Female} \cap \exists \text{hasChild.Male} \cap \forall \text{hasChild.}(\text{Rich} \cup \text{Happy})
\]
Typical Architecture

The division into TBox and ABox doesn’t have a logical significance, but is made for conceptual and implementation convenience.

father = man \land E \, has\_child \, X
human = mammal \land \text{biped}
...

father= man \land E \, has\_child \, X
human= mammal \land \text{biped}
...

John = human \land \text{father}
John \, has\_child \, Mary

Definitions of Terminology

Assertions about individuals
A family of languages

- The expressiveness of a description logic is determined by the operators that it uses
  - Add or eliminate operators (e.g., ¬, ∪), and the statements that can be expressed are increased/reduced in number
  - Higher expressiveness implies higher complexity
- \textit{AL} or \textit{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>$\sim C$</td>
<td>$\sim$ Human</td>
</tr>
<tr>
<td>atomic role</td>
<td>$R$</td>
<td>hasChild</td>
</tr>
<tr>
<td>conjunction</td>
<td>$C \land D$</td>
<td>Human $\land$ Male</td>
</tr>
<tr>
<td>value restriction</td>
<td>$R.C$</td>
<td>Human $\exists$ hasChild.Blond</td>
</tr>
<tr>
<td>existential rest. (lim)</td>
<td>$\exists R$</td>
<td>Human $\exists$ hasChild</td>
</tr>
<tr>
<td>Top (univ. conc.)</td>
<td>$T$</td>
<td>$T$</td>
</tr>
<tr>
<td>bottom (null conc)</td>
<td>$\bot$</td>
<td>$\bot$</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 boolean (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 rest.</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>
## Other Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number restriction</td>
<td>&gt;= n R</td>
<td>&gt;= 7 hasChild</td>
</tr>
<tr>
<td></td>
<td>&lt;= n R</td>
<td>&lt;= 1 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 ◦ R</td>
<td>hasParent ◦ hasBrother</td>
</tr>
<tr>
<td>Qualified # restric.</td>
<td>&gt;= n R.C</td>
<td>&gt;= 2 hasChild.Female</td>
</tr>
<tr>
<td>Singleton concepts</td>
<td>{&lt;name&gt;}</td>
<td>{Italy}</td>
</tr>
</tbody>
</table>
∀ and ∃ deserve special attention.

- Note that they only can come before a Role:
  \[
  \forall \text{HasChild.Girl} \quad \exists \text{isEmployedBy.Farmer}
  \]

- Remember, they describe sets of individuals

- \(\forall \text{HasChild.Girl would be interpreted as:}\)
  The set \(\{ x \mid \forall (y)( \text{HasChild}(x,y) \rightarrow \text{Girl}(y) ) \} \)
  Note the conditional: Are you in that set?

- \(\exists \text{isEmployedBy.Farmer would be:}\)
  The set \(\{ x \mid \exists (y)( \text{isEmployedBy}(x,y) \land \text{Farmer}(y) ) \} \)
Special names and combinations


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

$\Rightarrow \text{OWL-DL is SHOIN}^{(D)}$

$\Rightarrow \text{OWL 2 is SROIQ}^{(D)}$
Complexity of reasoning in Description Logics

Base description logic: Attributive Language with Complements

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

Concept constructors:
- \( \forall \) - functionality\(^2\) (\( \leq 1 \) \( R \))
- \( \forall^\subseteq \) (unqualified) number restrictions: (\( \geq 1 \) \( R \)), (\( \leq 1 \) \( R \))
- \( \forall^\exists \) qualified number restrictions: (\( \exists n R.C \)), (\( \leq n R.C \))
- \( \forall^\text{nominal} \) nominals: \( \{ a \} \) or \( \{ a_1, \ldots, a_n \} \) ("one-of")
- \( \mu \) - least fixpoint operator: \( \mu X.C \)

Role constructors:
- \( \forall \) - role inverse: \( R^- \)
- \( \forall \cap \) - role intersection\(^3\): \( R \cap S \)
- \( \forall \cup \) - role union: \( R \cup S \)
- \( \forall \to \) - role complement: \( \neg R \)
- \( \forall \circ \) - role chain (composition): \( R \circ S \)
- \( \forall \ast \) reflexive-transitive closure\(^4\): \( R^\ast \)
- \( \forall \text{id} \) - concept identity: \( \text{id}(C) \)

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

RBox (role axioms):
- \( \forall \to \) - role transitivity: \( \text{Tr}(R) \)
- \( \forall \) - role hierarchy: \( R \subseteq S \)
- \( \forall \ast \) - complex role inclusions: \( R \circ S \subseteq R, R \circ S \subseteq S \)
- \( \forall \circ \) - some additional features (check it to see)

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

Complexity of reasoning problems\(^7\)

<table>
<thead>
<tr>
<th>Reasoning problem</th>
<th>Complexity(^8)</th>
<th>Comments and references</th>
</tr>
</thead>
</table>
| Concept satisfiability | PSpace-complete | • Hardness for \( ALC \): see \([80]\).  
• Upper bound for \( ALCQ \): see \([12, \text{Theorem 4.6}]\). |
| ABox consistency | PSpace-complete | • Hardness follows from that for concept satisfiability.  
• Upper bound for \( ALCQO \): see \([17, \text{Appendix A}]\). |

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 \([4, \text{Sect. 2.3}]\). |
| 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 \([4, \text{Proposition 2.15}]\). |

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

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

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, john is a male, john has a child mary
# OWL Class Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>DL Syntax</th>
<th>Example</th>
<th>(Modal Syntax)</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>$C_1 \sqcap \ldots \sqcap C_n$</td>
<td>Human $\sqcap$ Male</td>
<td>$C_1 \wedge \ldots \wedge C_n$</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \sqcup \ldots \sqcup C_n$</td>
<td>Doctor $\sqcup$ Lawyer</td>
<td>$C_1 \vee \ldots \vee C_n$</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
<td>\neg Male</td>
<td>$\neg C$</td>
</tr>
<tr>
<td>oneOf</td>
<td>${x_1 \ldots x_n}$</td>
<td>{john, mary}</td>
<td>$x_1 \vee \ldots \vee x_n$</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>$\forall P.C$</td>
<td>$\forall$hasChild.Dr doctor</td>
<td>$[P]C$</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists P.C$</td>
<td>$\exists$hasChild.Lawyer</td>
<td>$\langle P\rangle C$</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\leq nP$</td>
<td>$\leq 1$hasChild</td>
<td>$[P]_{n+1}$</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq nP$</td>
<td>$\geq 2$hasChild</td>
<td>$\langle P\rangle_n$</td>
</tr>
</tbody>
</table>

- **XMLS datatypes** as well as classes in $\forall P.C$ and $\exists P.C$
  - E.g., $\exists$hasAge.$\text{nonNegativeInteger}$
- Arbitrarily complex **nesting** of constructors
  - E.g., $\text{Person} \sqcap \forall$hasChild.($\text{Doctor} \sqcup \exists$hasChild.$\text{Doctor}$)
## OWL Axioms

<table>
<thead>
<tr>
<th>Axiom</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>subClassOf</td>
<td>$C_1 \sqsubseteq C_2$</td>
<td>Human $\sqsubseteq$ Animal $\sqcap$ Biped</td>
</tr>
<tr>
<td>equivalentClass</td>
<td>$C_1 \equiv C_2$</td>
<td>Man $\equiv$ Human $\sqcap$ Male</td>
</tr>
<tr>
<td>disjointWith</td>
<td>$C_1 \sqsubseteq \neg C_2$</td>
<td>Male $\sqsubseteq$ $\neg$Female</td>
</tr>
<tr>
<td>sameIndividualAs</td>
<td>${x_1} \equiv {x_2}$</td>
<td>${\text{President}<em>\text{Bush}} \equiv {\text{G}</em>\text{W}_\text{Bush}}$</td>
</tr>
<tr>
<td>differentFrom</td>
<td>${x_1} \sqsubseteq \neg {x_2}$</td>
<td>${\text{john}} \sqsubseteq \neg {\text{peter}}$</td>
</tr>
<tr>
<td>subPropertyOf</td>
<td>$P_1 \sqsubseteq P_2$</td>
<td>hasDaughter $\sqsubseteq$ hasChild</td>
</tr>
<tr>
<td>equivalentProperty</td>
<td>$P_1 \equiv P_2$</td>
<td>cost $\equiv$ price</td>
</tr>
<tr>
<td>inverseOf</td>
<td>$P_1 \equiv P_2^-$</td>
<td>hasChild $\equiv$ hasParent$^-$</td>
</tr>
<tr>
<td>transitiveProperty</td>
<td>$P^+ \sqsubseteq P$</td>
<td>ancestor$^+$ $\sqsubseteq$ ancestor</td>
</tr>
<tr>
<td>functionalProperty</td>
<td>$\top \sqsubseteq \leq 1 P$</td>
<td>$\top \sqsubseteq \leq 1 \text{hasMother}$</td>
</tr>
<tr>
<td>inverseFunctionalProperty</td>
<td>$\top \sqsubseteq \leq 1 P^-$</td>
<td>$\top \sqsubseteq \leq 1 \text{hasSSN}^-$</td>
</tr>
</tbody>
</table>

**I satisfies** $C_1 \sqsubseteq C_2$ iff $C_1^T \sqsubseteq C_2^T$; satisfies $P_1 \sqsubseteq P_2$ iff $P_1^T \sqsubseteq P_2^T$

**I satisfies** ontology $\mathcal{O}$ (is a model of $\mathcal{O}$) iff satisfies every axiom in $\mathcal{O}$
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, as well

- **Concept satisfiability** is $C$ (necessarility) empty?
- **Instance Checking** $\text{Father}(\text{john})$?
- **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 often specify both *necessary and sufficient* properties
Subsumption

- Meaning of Subsumption
  
  A *more general concept or description is said to subsume* a *more specific one. Members of a subsumed concept are necessarily members of a subsuming concept*

- Two ways to formalize the 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
    
    The instances of subsumed concept are necessarily a subset of the subsuming concept’s instances
    
    E.g., the set of all persons is a subset of all animals
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.
Defining a “rabid dog”

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.
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
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

- This is often impractical or impossible, especially with natural kinds.

- A “primitive” definition is an incomplete definition
  - This limits the amount of classification that can be done automatically

- Example:
  - Primitive: a Person
  - Defined: Parent = Person with at least one child
Intentional versus Extensional Semantics

- **Extensional Semantics** are a model-theoretic idea. They define the meaning of a description by enumerating the set of objects that satisfy the description.

- **Intensional Semantics** defines the meaning of a description based on the intent or use of the description.

- Example:
  - Morning-Star    Evening-Star
    - **Extensional:** Same object, namely Venus
    - **Intensional:** Different objects, one meaning Venus seen in the morning and one in the evening.
Definition vs. Assertion

- A **definition** describes *intrinsic* properties of an object. The parts of a description have meaning as a part of a composite description of an object.

- An **assertion** is used to describe an *incidental* property of an object. Asserted facts have meaning on their own.

- Example: “a black telephone”
  Could be either a description or an assertion, depending on the meaning and import of “blackness” on the concept telephone.
Definition versus Assertion

- In English, “a black telephone” is ambiguous
  1. A black telephone is a common sight in an office
  2. A black telephone is on the corner of my desk
- KR languages should not be ambiguous so typically distinguish between descriptions of classes and descriptions of individuals
- KR languages often allow additional assertions to be made that are not part of the definition (e.g., annotation properties in OWL)
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?
Concept Classification Infers Normal BP is Subsumed by Non-Critical BP

- Systolic Blood Pressure
  - pressure $\geq 90$
  - pressure $\geq 85$
  - pressure $\leq 140$
  - pressure $\leq 160$

- Non-Critical Systolic BP
  - pressure $\leq 160$
  - pressure $\geq 85$

- Normal Systolic BP
  - pressure $\leq 140$
  - pressure $\geq 90$
With Classified Concepts the Answer is Easy to Compute

Non-Critical Systolic BP

Systolic Blood Pressure

<= 160 pressure

>= 85 pressure

<= 140 pressure

Normal Systolic BP

Pressure >= 90

Joe’s BP

With Classified Concepts the Answer is Easy to Compute
Incidental properties

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

See http://en.wikipedia.org/wiki/Description_logic

- **CEL**, free (for non-commercial use), LISP
- **Cerebra Engine**, commercial, C++
- **FaCT++**, free, open-source, C++
- **KAON2** free (for non-commercial usage), Java
- **MSPASS** free, open-source, C
- **Pellet** free, open-source, Java
- **RacerPro** commercial, LISP

**DIG** is a standard interface to a DL reasoner that predates RDF and today uses XML

**Protégé** uses DIG and can thus use any of several DL reasoners that have a DIG interface
Dig API: http://dig.sourceforge.net/