Description logics

LESSON 6: SYNTAX AND SEMANTICS, DECISION PROBLEMS, INFERENCE
Categories and objects [AIMA, Cap 12]

- Most of the reasoning takes place at the level of **categories** rather than on individuals.
- If we organize knowledge in categories and subcategories (in a **hierarchy**) it is enough to classify an object, according to its perceived properties, in order to infer properties of the categories to which it belongs.
- **Inheritance** is a common form of inference.
- **Ontologies** will play a crucial role, providing a source of shared and precisely defined terms that can be used in meta-data of digital objects and real world objects.
Domain ontologies

- In the 80’s we assist to a formalization of the ideas coming from *semantic networks* and *frames* resulting in *specialized logics*.
- These logics, called *terminological logics* and later *description logics* find an important application in describing “domain ontologies” and represent the theoretical foundations for adding reasoning capabilities to the *Semantic web*.
- **Ontology**: a formal model of an application domain (a conceptualization)
- Subclass relations are important in defining the terminology and serve to organize knowledge in hierarchical taxonomies (like in botany, biology, in library sciences ... but also electronic commerce, cultural heritage ...)

25/10/17
The Semantic Web

- The Semantic Web is the vision of Tim Berners-Lee (1998) to gradually develop alongside the “syntactic web” (or web of documents), for communication among people, a “semantic web” (or web of data) for communication among machines.

- The semantic web is a huge distributed network of linked data which can be used by programs as well, provided their semantics is shared and made clear (this is the role of formal ontologies).

- These data comply with standard web technologies: Unicode encoding, XML, URI, HTTP web protocol.
The technological stack of Semantic Web
The technologies of the Semantic Web

- Unicode and URI (Universal Resource Identifier)
- XML for syntactic interoperability
- RDF (Resource Description Framework): a language to describe binary relations between resources \((subject, predicate, object)\)
- RDF schema (RDFS): to define classes, relations between classes, to constrain domains and co-domains of relations. This is basic language for ontologies.
- OWL: the web ontology language, one among many description logics elected as standard by the W3C.

The **Web Semantic** course will tell you more about all this.
Description logics

Can be seen as:

1. Logical counterparts of **network** knowledge representation schema, **frames** and **semantic networks**.
   - In this formalization effort, **defaults** and exceptions are lost
   - The ideas and terminology (concept, roles, inheritance hierarchies) are very similar (to KLOne in particular).

2. Contractions of first order logic (FOL), investigated to obtain better computational properties.
   - Attention to computational complexity/decidability of the inference mechanisms
Example

The following is a typical proposition, expressed in the syntax of DL ("paper3 has exactly two authors"):

\[(\text{and} \quad \text{Paper} \quad \text{(atmost} \quad 2 \quad \text{hasAuthor}) \quad \text{(atleast} \quad 2 \quad \text{hasAuthor})) \quad [\text{paper3}]\]

Alternative, mathematical, notation (that we will use):

\[\text{paper3: Paper} \quad \prod \quad (\leq \quad 2 \quad \text{hasAuthor}) \quad \prod \quad (\geq \quad 2 \quad \text{hasAuthor})\]

Corresponding in FOL:

\[\text{Paper(paper3)} \quad \land \]
\[\exists x \quad \text{hasAuthor(paper3, x)} \quad \land \]
\[\exists y \quad \text{hasAuthor(paper3, y)} \quad \land \quad x \neq y \quad \land \]
\[\text{hasAuthor(paper3, z)} \quad \Rightarrow \quad (z = x) \quad \lor \quad (z = y)\]
Concepts, roles, individuals

Each DL is characterized by operators for the construction of terms.

**Terms** are of three types:

- **Concepts**, corresponding to unary relations
  with operators for the construction of complex concepts: and (\(\Pi\)), or (\(\cup\)), not (\(\neg\)), all (\(\forall\)), some (\(\exists\)), atleast (\(\geq n\)), atmost (\(\leq n\)), ...

- **Roles**, corresponding to binary relations
  possibly together with operators for construction complex roles

- **Individuals**: only used in assertions

**Assertions** are kept separate and can be only of two types:

- \(i : C\), where \(i\) an individual and \(C\) is a concept
- \((i, j) : R\), where \(i\) and \(j\) are individuals and \(R\) is a role
A KB based on description logic

KB

T-BOX

journalist ⇒ author
article ≡
(and (a-not book)
  (all author journalist))
author ⇒ creator

A-BOX

author[Eco, l1]
author[Biagi, l2]
journalist[Biagi]
(and book
  (all author journalist))[a2]
The logic $\mathcal{AL}$: the syntax of terms

<concept> → $A$

| $\top$                (top, universal concept) |
| $\bot$               (bottom)                 |
| $\neg A$             (atomic negation)       |
| $C \cap D$           (intersection)          |
| $\forall R . C$      (value restriction)     |
| $\exists R . \top$   (weak existential)      |

<role> → $R$

$A, B$ primitive concepts

$R$ primitive role

$C, D$ concepts

Examples:

- Person $\cap$ Female
- Person $\cap$ $\neg$Female
- Person $\cap$ $\exists$ hasChild. $\top$
- Person $\cap$ $\forall$ hasChild . Female
- Person $\cap$ $\forall$ hasChild . $\bot$
Semantics of $\mathcal{AL}$

$\Delta^I$ interpretation domain, a set of individuals

$I$ interpretation function, assigning:

- Atomic concepts $A$: $A^I \subseteq \Delta^I$
- Atomic roles $R$: $R^I \subseteq \Delta^I \times \Delta^I$
- Individual constants $a$: $a^I \in \Delta^I$

$\top^I = \Delta^I$ the interpretation domain

$\bot^I = \emptyset$ the empty set

$(-A)^I = \Delta^I \setminus A^I$ the complement of $A^I$ to $\Delta$

$(C \cap D)^I = C^I \cap D^I$ the intersection of the sets

$(\forall R. C)^I = \{a \in \Delta^I | \forall b . (a, b) \in R^I \rightarrow b \in C^I\}$

$(\exists R. T)^I = \{a \in \Delta^I | \exists b . (a, b) \in R^I\}$
Examples

1. **Persons with a child**
   \[ \text{Person} \sqcap \exists \text{hasChild} . \top \]

2. **Persons with only female children**
   \[ \text{Person} \sqcap \forall \text{hasChild} . \text{Female} \]

3. **All articles that have at least one authors and whose authors are all journalists.**
   \[ \text{Article} \sqcap \exists \text{hasAuthor} . \top \sqcap \forall \text{hasAuthor} . \text{Journalist} \]
More expressive logics

\( \mathcal{U} \): union, \( (C \sqcup D)^I = (C^I \cup D^I) \)

\( \mathcal{E} \): full existential

\( (\exists R \cdot C)^I = \{ a \in \Delta^I \mid \exists b. (a, b) \in R^I \land b \in C^I \} \)

\( \mathcal{N} \): numerical restrictions

\( (\geq n R)^I = \{ a \in \Delta^I \mid |\{ b \mid (a, b) \in R^I\}| \geq n \} \) (atleast)

\( (\leq n R)^I = \{ a \in \Delta^I \mid |\{ b \mid (a, b) \in R^I\}| \leq n \} \) (atmost)

\( n \), integer number \| \set \) cardinality

\( \mathcal{C} \): full complement, \( (\neg C)^I = \Delta^I \setminus C^I \)
Different description logics are obtained by adding other costructors to $\mathcal{AL}$.

$\mathcal{AL}[U][E][N][C]$  

Not all of them are distinct.

$\mathcal{ALUE} = \mathcal{ALC}$ given that $(C \sqcup D) \equiv \neg(\neg C \cap \neg D)$
and $\exists R. C \equiv \neg \forall R. \neg C$

$\mathcal{ALCN} = \mathcal{ALUEN}$
The language for T-BOX terminology

- **Terminological axioms** $\mathcal{T}$
  
  $C \subseteq D$  inclusion of concepts, $C^I \subseteq D^I$
  $R \subseteq S$  inclusion of roles $R^I \subseteq S^I$
  $C = D$  equality of concepts, $C^I = D^I$
  $R = S$  equality of roles, $R^I = S^I$

**Definitions**: equalities introducing a symbol on the left

  Mother $\equiv$ Woman $\sqcap$ hasChild.Person

**Terminology**: symbols appear on the left not more than once

**Primitive symbols**: appear only on the right

**Defined symbols**: may appear also on the left

We assume acyclic $\mathcal{T}$.  


An acyclic terminology

Woman ≡ Person ⊓ Female
Man ≡ Person ⊓ ¬ Woman
Mother ≡ Woman ⊓ ∃ hasChild. Person
Father ≡ Man ⊓ ∃ hasChild. Person
Parent ≡ Father ⊔ Mother
Grandmother ≡ Mother ⊓ ∃ hasChild. Parent
MotherWithManyChildren ≡ Mother ⊓ ⩾ 3 hasChild
MotherWithoutDaughter ≡ Mother ⊓ ∀ hasChild. ¬ Woman
Wife ≡ Woman ⊓ ∃ hasHusband. Man
Expansion of a terminology $\mathcal{T}$

If a terminology is acyclic, it can be expanded by substituting to defined symbols their definitions.

In the case of acyclic terminologies the process converges and the expansion $\mathcal{T}^e$ is unique.

Properties of $\mathcal{T}^e$:

- in $\mathcal{T}^e$ each equality has the form $C \equiv D^e$ where $D^e$ contains only primitive symbols
- $\mathcal{T}^e$ contains the same primitive and defined symbols of $\mathcal{T}$
- $\mathcal{T}^e$ is equivalent to $\mathcal{T}$
Expanded terminology

**Woman** $\equiv$ Person $\sqcap$ Female

**Man** $\equiv$ Person $\sqcap$ $\neg$(Person $\sqcap$ Female)

**Mother** $\equiv$ (Person $\sqcap$ Female) $\sqcap$ $\exists$ hasChild.Person

**Father** $\equiv$ (Person $\sqcap$ $\neg$(Person $\sqcap$ Female)) $\sqcap$ $\exists$ hasChild.Person

**Parent** $\equiv$ ((Person $\sqcap$ $\neg$(Person $\sqcap$ Female)) $\sqcap$ $\exists$ hasChild.Person)

**Grandmother** $\equiv$ ((Person $\sqcap$ Female) $\sqcap$ $\exists$ hasChild.Person)

$\sqcap$ $\exists$ hasChild.(((Person $\sqcap$ $\neg$(Person $\sqcap$ Female))

$\sqcap$ $\exists$ hasChild.Person)

$\sqcup$ ((Person $\sqcap$ Female)

$\sqcap$ $\exists$ hasChild.Person))

**MotherWithManyChildren** $\equiv$ ((Person $\sqcap$ Female) $\sqcap$ $\exists$ hasChild.Person) $\sqcap$ $\geq$ 3 hasChild

**MotherWithoutDaughter** $\equiv$ ((Person $\sqcap$ Female) $\sqcap$ $\exists$ hasChild.Person)

$\forall$ hasChild.($\neg$(Person $\sqcap$ Female))

**Wife** $\equiv$ (Person $\sqcap$ Female)

$\sqcap$ $\exists$ hasHusband.($\neg$(Person $\sqcap$ Female))
Specializations

Inclusion axioms are called **specializations**. For example:

\[
\text{Woman} \sqsubseteq \text{Person}
\]

A **generalized terminology** [with inclusion axioms], if acyclic, can be transformed in an **equivalent** terminology with just equivalence axioms:

\[
A \sqsubseteq C \rightarrow A \equiv A' \sqcap C
\]

where \(A'\) is a new primitive symbol

This also means that specializations do not add expressive power to the language, at least in the case of acyclic terminologies.
The language of assertions: A-BOX

An A-BOX is a set of assertions of the following type:

- \( a : C \), assertion over concepts, meaning \( a^I \in C^I \)
- \((b, c) : R\), assertions over su roles, meaning \((b^I, c^I) \in R^I\)

\(a, b, c, d\) ... are individuals

In description logic we make an assumption that different individual constants refer to different individuals: the **Unique Name Assumption** (UNA)

A-Box example:

- Mary: Mother
- (Mary, Peter): hasChild
- (Mary, Paul): hasChild
- Peter: Father
- (Peter, Harry): hasChild
DL are a contraction of FOL

It is always possible to translate DL assertions in FOL.

We define a translation function \( t(C, x) \) which returns a FOL formula with \( x \) free:

\[
t(C, x) \mapsto C(x)
\]

Translation rules for assertions:

- \( t(C \sqsubseteq D) \mapsto \forall x. \ t(C, x) \Rightarrow t(D, x) \)
- \( t(a : C) \mapsto t(C, a) \)
- \( t((a, b) : R) \mapsto R(a, b) \)

Translation rules for terms:

- \( t(\top, x) \mapsto true \)
- \( t(\bot, x) \mapsto false \)
- \( t(A, x) \mapsto A(x) \)
- \( t(C \sqcap D, x) \mapsto t(C, x) \land t(D, x) \)
- \( t(C \sqcup D, x) \mapsto t(C, x) \lor t(D, x) \)
- \( t(\neg C, x) \mapsto \neg t(C, x) \)
- \( t(\exists R . C, x) \mapsto \exists y . R(x, y) \land t(C, y) \)
- \( t(\forall R . C, x) \mapsto \forall y . R(x, y) \Rightarrow t(C, y) \)
Translation examples

\[ t(\text{HappyFather} \sqsubseteq \text{Man} \sqcap \exists \text{hasChild} \cdot \text{Female}) = \]
\[ \forall x. \ t(\text{HappyFather}, x) \Rightarrow t(\text{Man} \sqcap \exists \text{hasChild} \cdot \text{Female}, x) = \]
\[ \forall x. \ \text{HappyFather}(x) \Rightarrow t(\text{Man}, x) \land t(\exists \text{hasChild} \cdot \text{Female}, x) = \]
\[ \forall x. \ \text{HappyFather}(x) \Rightarrow \text{Man}(x) \land t(\exists \text{hasChild} \cdot \text{Female}, x) = \]
\[ \forall x. \ \text{HappyFather}(x) \Rightarrow \text{Man}(x) \land \exists y. \ \text{hasChild}(x, y) \land \text{Female}(y) \]

\[ t(a: \text{Man} \sqcap \exists \text{hasChild} \cdot \text{Female}) = \text{Man}(a) \land (\exists y. \ \text{hasChild}(a, y) \land \text{Female}(y)) \]
Alternative syntax (Lisp like)

\[\begin{align*}
\top & \rightarrow \text{*top*} \\
\bot & \rightarrow \text{*bottom*} \\
\neg C & \rightarrow (\text{not } C) \\
C \cap D & \rightarrow (\text{and } C \ D) \\
C \cup D & \rightarrow (\text{or } C \ D) \\
\exists R. C & \rightarrow (\text{some } R \ C) \\
\forall R. C & \rightarrow (\text{all } R \ C) \\
(\geq n \ R) & \rightarrow (\text{at-least } n \ R) \\
(\leq n \ R) & \rightarrow (\text{at-most } n \ R) \\
(\geq n \ R. C) & \rightarrow (\text{at-least } n \ R \ C) \\
(\leq n \ R. C) & \rightarrow (\text{at-most } n \ R \ C) \\
\{a\} & \rightarrow (\text{one-of } a) \\
R^- & \rightarrow (\text{inv } R) \\
C \sqsubseteq D & \rightarrow (\text{implies } C \ D) \\
A = C & \rightarrow (\text{define-concept } A \ C) \\
R \sqsubseteq P & \rightarrow (\text{implies-role } R \ P) \\
\text{fun}(f) & \rightarrow (\text{functional } f) \\
\text{trans}(R) & \rightarrow (\text{transitive } R) \\
a:C & \rightarrow (\text{instance } a \ C) \\
(a, b):R & \rightarrow (\text{related } a \ b \ R)
\end{align*}\]
The knowledge base in description logics

\( \mathcal{K} = (\mathcal{T}, \mathcal{A}) \)

\( \mathcal{T} \) (T-BOX), terminological component
\( \mathcal{A} \) (A-BOX), assertional component

An interpretation \( I \) satisfies \( \mathcal{A} \) and \( \mathcal{T} \) (therefore \( \mathcal{K} \)) iff it satisfies any assertion in \( \mathcal{A} \) and definition in \( \mathcal{T} \) (\( I \) is a model of \( \mathcal{K} \)).
Reasoning services for description logics

Design and management of ontologies
- Consistency checking of concepts and support for the creation of hierarchies

Ontology integration
- Relations between concepts of different ontologies
- Consistency of integrated hierarchies

Queries
- Determine whether facts are consistent wrt ontologies
- Determine if individuals are instances of concepts
- Retrieve individuals satisfying a query (concept)
- Verify if a concept is more general than another (subsumption)
Basic decision problems in DL

Classical decision problems

- **Satisfiability** of a KB: \( \text{KBS}(\mathcal{K}) \) is there a model for \( \mathcal{K} = (\mathcal{T}, \mathcal{A}) \)?

- **Logical consequence** of a KB: \( \mathcal{K} \models a : C \) also called **instance checking**

Typical decision problems

- **Concept satisfiability** \([\text{CS}(c)]\): is there an interpretation different from the empty set?
  - (father), a primitive concept, is satisfiable
  - (father \( \sqcap \lnot \text{father} \)) is unsatisfiable

- **Subsumption**: \( \mathcal{K} \models C \sqsubseteq D \) (\( D \) subsumes \( C \)) if for every model \( \mathcal{I} \) of \( \mathcal{T} \), \( C \sqsubseteq D \)
  - *structural subsumption*: person **subsumes** (person \( \sqcap \exists \text{hasChild}.T \))
  - *hybrid subsumption*:
    - person \( \sqcap \exists \text{hasChild}.T \) **subsumes** student \( \sqcap \exists \text{hasChild}.T \) if student \( \sqsubseteq \) person \( \in \) T-BOX

- **Concept equivalence**: \( \mathcal{K} \models C \equiv D \)
Other inferential services

- **Disjointness**: $C^I \cap D^I = \emptyset$ for any model $I$ of $\mathcal{T}$

- **Retrieval**: find all individuals which are instances of $C$, i.e. compute the set
  \[ \{a \mid \mathcal{K} \models a : C\} \]

- **Most Specific Concept** (MSC)
  Given a set of individuals, find the most specific concept of which they are instances. Used for classification.

- **Least Common Subsumer** (LCS)
  Given a set of concepts, find the most specific concept which subsumes all of them. Used for classification.
Reduction between decision problems

Decision problems are not independent.

- Structural subsumption can be reduced to concept satisfiability

- $C$ is unsatisfiable iff $C$ is subsumed by $ot$

- $C$ and $D$ sono disjoint iff $C \sqcap D$ is not consistent.

- ...

All problems can be reduced to KB satisfiability.

1. Concept consistency: $C$ is satisfiable iff $\mathcal{K} \cup \{a : C\}$ is satisfiable with a new individual constant. Note: $\{a : C\}$ is added to $\mathcal{A}$.

2. Subsumption: $\mathcal{K} \models C \sqsubseteq D$ (D subsumes C) iff $\mathcal{K} \cup \{a : C \sqcap \lnot D\}$ is unsatisfiable

3. Equivalence: $\mathcal{K} \equiv C$ iff $\mathcal{K} \not\models C \sqsubseteq D$ and $\mathcal{K} \not\models D \sqsubseteq C$

4. Instance checking: $\mathcal{K} \not\models a : C$ iff $\mathcal{K} \cup \{a : \lnot C\}$ is unsatisfiable
Examples of problem reduction

1. Are rich people happy?
   - Happy subsumes Rich? $\mathcal{K} \models \text{Rich} \sqsubseteq \text{Happy}
   - $\mathcal{K} \cup \{a: \text{Rich} \land \neg \text{Happy}\}$ is unsatisfiable?

2. Being rich and healthy is enough to be happy?
   - $\mathcal{K} \models \text{Rich} \land \text{Healthy} \sqsubseteq \text{Happy}
   - $\mathcal{K} \cup \{a: \text{Rich} \land \text{Healthy} \land \neg \text{Happy}\}$ is unsatisfiable?

3. Given that: To be happy one needs to be rich and healthy (and it is not enough)
   Can a rich person be unhappy?
   - T-BOX: Happy $\sqsubseteq$ Rich $\land$ Healthy
   - $(\text{Rich} \land \neg \text{Happy})$ is satisfiable?
   - $\mathcal{K} \cup \{a: \text{Rich} \land \neg \text{Happy}\}$ is satisfiable?
Deductive systems for DL

Algorithm for structural subsumption
- Used for not very expressive languages (without negation) → Your turn

The most used method is a technique for verifying **satisfiability of a KB** (KBS).
- It is technique of constraint propagation, a variant of a method for natural deduction, called **semantic tableaux**
- *Basic idea:* each formula in KB is a constraint on interpretations for them to be models of KB
- Complex constraints are decomposed in simpler constraints by means of propagation rules until we obtain, in a finite number of steps, **atomic constraints**, which cannot further decomposed.
- If the set of atomic constraints contains an evident contradiction then the KB is not satisfiable, otherwise a model has been found.
- The technique is simple, modular, useful for evaluating complexity of decision algorithm.
The logic $\mathcal{ALC}$

We will apply the technique to $\mathcal{ALC} = \mathcal{AL} +$ full complement (and union).

$\langle\text{concept}\rangle \rightarrow A$

| $\top$       | (top, universal concept) |
| $\bot$      | (bottom)                  |
| $\neg C$    | (atomic negation)         |
| $C \sqcap D$| (intersection)            |
| $C \sqcup D$| (union)                   |
| $\forall R.C$| (value restriction)      |
| $\exists R.T$| (weak existential)       |

$\langle\text{role}\rangle \rightarrow R$

$A \quad$ primitive concepts

$R \quad$ primitive role

$C, D \quad$ concepts
Preliminary steps before KBS

1. **Terminology expansion**: a preliminary step consisting in resolving specializations, getting rid of the terminology by substituting defined concepts in $\mathcal{A}$ with their definitions.
   
   This results in a $\mathcal{K} = (\{ \}, \mathcal{A}')$ with assertions only.

2. **Normalization**: assertions are transformed in *negation normal form*, by applying the following rules until every occurrence of negation is in front of a primitive concept.
   
   These transformed assertions constitute the initial set of constraints for the KBS algorithm.
Constraint propagation algorithm

A constraint is an assertion of the form \(a : C\) or \((b, c) : R\), where \(a, b\) and \(c\) are constants (distinct individuals) or variables \((x, y \ldots)\) referring to individuals but not necessarily distinct ones.

A constraint set \(\mathcal{A}\) is satisfiable \textit{iff} there exists an interpretation satisfying all the constraints in \(\mathcal{A}\).

Each step of the algorithm decomposes a constraint into a simpler one until we get a set of elementary constraints, or a contradiction (\textbf{clash}) is found.

For \(\mathcal{ALC}\) a \textit{clash} is one of the following types:

- \(\{a : C, a : \neg C\}\)
- \(\{a : \bot\}\)
Completion trees

Completion forest: data structures for supporting the execution of the algorithm
For each individual $a$ appearing in assertions in $\mathcal{A}$, a labelled tree is initialized.

- if $\mathcal{A}$ contains $a : C$, we add the constraint $C$ to the label of $a$
- if $\mathcal{A}$ contains $(a, b) : R$, we create a successor node of $a$ for $b$ to represent their $R$ relation

\[
\begin{array}{c}
a \\
L(a) = \{ C \} & \text{label of } a \\
\end{array}
\]

\[
\begin{array}{c}
R \\
b
\end{array}
\]
Rules for $\mathcal{ALC}$

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\cap)$ if 1. $(C_1 \cap C_2) \in \mathcal{L}(x)$ and 2. $(C_1, C_2) \not\subseteq \mathcal{L}(x)$ then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup {C_1, C_2}$</td>
<td></td>
</tr>
<tr>
<td>$(\cup)$ if 1. $(C_1 \cup C_2) \in \mathcal{L}(x)$ and 2. $(C_1, C_2) \cap \mathcal{L}(x) = \emptyset$ then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup {C}$ for some $C \in {C_1, C_2}$</td>
<td>neither one is in $\mathcal{L}(x)$</td>
</tr>
<tr>
<td>$(\exists)$ if 1. $\exists R.C \in \mathcal{L}(x)$ and 2. $x$ has no $R$-successor $y$ with $C \in \mathcal{L}(y)$ then create a new node $y$ with $\mathcal{L}((x, y)) = {R}$ and $\mathcal{L}(y) = {C}$</td>
<td></td>
</tr>
<tr>
<td>$(\forall)$ if 1. $\forall R.C \in \mathcal{L}(x)$ and 2. $x$ has an $R$-successor $y$ with $C \not\in \mathcal{L}(y)$ then $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup {C}$</td>
<td>not both in $\mathcal{L}(x)$</td>
</tr>
</tbody>
</table>
Comments about rules

Most rules are deterministic
The rule for disjunction is \textbf{non deterministic}: its application results in alternative constraints sets.: we have a fork in the proof.
\( \mathcal{A} \) is satisfiable \textit{iff} at least one of the resulting constraints set is satisfiable.
\( \mathcal{A} \) is unsatisfiable \textit{iff} all the alternatives end up with a clash.
Example 1

All the children of John are girls. Mary is a child of John.
Tim is a friend of professor Blake. Prove that Mary is a girl.

$$\mathcal{A} = \{\text{john : } \forall \text{hasChild}.\text{girl}, (\text{john, mary}) : \text{hasChild},
(\text{blake, tim}) : \text{hasFriend}, \text{blake} : \text{professor}\}$$

Prove that: $$\mathcal{A} \models \text{mary : girl}$$ or equivalently that $$\mathcal{A} \cup \text{mary : } \neg \text{girl}$$ is unsatisfiable.
Example 2

\[ \mathcal{A} = \{ x : \exists R . C \land \forall R . (\neg C \cup \neg D) \land \exists R . D \} \text{ satisfiable?} \]

\[ \mathcal{L}(x) = \{ \exists R . C, \ \forall R . (\neg C \cup \neg D), \ \exists R . D \} \]

\[ \mathcal{L}(y_1) = \{ C, \neg D, C \cup \neg D \} \]

\[ \mathcal{L}(y_2) = \{ D, C, C \cup \neg D \} \]

A is satisfiable

Model found: \( \Delta^I = \{ x, y_1, y_2 \} \quad C^I = \{ y_1 \} \quad D^I = \{ y_2 \} \quad R^I = \{(x, y_1), (x, y_2)\} \)
Example 3

\[ \mathcal{A} = \{ x : \exists R.C \land \forall R.\neg C \} \text{ satisfiable?} \]

\[ L(x) = \{ \exists R.C , \forall R.\neg C \} \]

\[ CLASH \]

\[ L(y_1) = \{ C, \neg C \} \]

\[ x \]

\[ y_1 \]

- \( \mathcal{A} \) is not satisfiable
- There are no models
Correctness and completeness of KBS

1. The result is invariant with respect to the order of application of the rules.

2. **Correctness**: if the algorithm terminates with at least one primitive constraint set and no *clashes*, then $\mathcal{A}$ is satisfiable and from the constraints we can derive a model.

2. **Completeness**: if a knowledge base $\mathcal{A}$ is satisfiable, then the algorithm terminates producing at least a finite model without clashes.

3. KBS is **decidable** for $\mathcal{ALC}$ and also for $\mathcal{ALCN}$. 
**Additional constructors**

\( H \): inclusion between roles

\[ R \subseteq S \iff R^I \subseteq S^I \]

\( Q \): qualified numerical restrictions

\[ (\geq n \ R.C)^I = \{a \in \Delta^I \mid \{b \mid (a, b) \in R^I \land b \in C^I\} \geq n\} \]

\[ (\leq n \ R.C)^I = \{a \in \Delta^I \mid \{b \mid (a, b) \in R^I \land b \in C^I\} \leq n\} \]

\( O \): nominals (singletons) \( \{a\}^I = \{a^I\} \)

\( I \): inverse roles, \( (R^{-})^I = \{(a, b) \mid (b, a) \in R^I\} \)

\( F \): functional roles

\[ \text{fun}(F) \iff \forall x, y, z \ (x, y) \in F^I \land (x, z) \in F^I \Rightarrow y = z \]

\( R^+ \): transitive role

\[ (R^+)^I = \{(a, b) \mid \exists c \ such \ that \ (a, c) \in R^I \land (c, b) \in R^I\} \]

\( S \): \( \mathcal{ALC} + R^+ \)
OWL – Ontology Web Language

**OWL-DL** is equivalent to

\[ \text{SHOIN} = \]

\[
S : \quad \text{ALC} + \text{transitive roles } R_+
\]

\[
H : \quad \text{roles specialization}
\]

\[
O : \quad \text{nominals/singletons}
\]

\[
I : \quad \text{inverse roles}
\]

\[
N : \quad \text{numerical restrictions}
\]

**OWL-Lite** is equivalent to

\[ \text{SHIF} = \]

\[
S : \quad \text{ALC} + \text{transitive roles } R_+
\]

\[
H : \quad \text{roles specialization}
\]

\[
O : \quad \text{nominals/singletons}
\]

\[
I : \quad \text{inverse roles}
\]

\[
F : \quad \text{functional roles}
\]
### OWL syntax

<table>
<thead>
<tr>
<th>Constructor</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (URI)</td>
<td>$A$</td>
<td>Conference</td>
</tr>
<tr>
<td>thing</td>
<td>$\top$</td>
<td></td>
</tr>
<tr>
<td>nothing</td>
<td>$\bot$</td>
<td></td>
</tr>
<tr>
<td>intersectionOf</td>
<td>$C_1 \sqcap \ldots \sqcap C_n$</td>
<td>Reference $\sqcap$ Journal</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \sqcup \ldots \sqcup C_n$</td>
<td>Organization $\sqcup$ Institution</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
<td>$\neg$MasterThesis</td>
</tr>
<tr>
<td>oneOf</td>
<td>${x_1} \sqcup \ldots \sqcup {x_n}$</td>
<td>$\forall$date.Date $\exists$date.{2005} ($\leq 1$ location) ($\geq 1$ publisher)</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>$\forall P. C$</td>
<td></td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists P. C$</td>
<td></td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\leq n P$</td>
<td></td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq n P$</td>
<td></td>
</tr>
</tbody>
</table>
# 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>${\text{President}<em>\text{Bush}} \equiv {\text{G.W.}</em>\text{Bush}}$</td>
</tr>
<tr>
<td>differentFrom</td>
<td>${x_1} \sqsubseteq \neg {x_2}$</td>
<td>${john} \sqsubseteq \neg {peter}$</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>$\top \sqsubseteq \leq 1P$</td>
<td>$\top \sqsubseteq \leq 1\text{hasMother}$</td>
</tr>
<tr>
<td>inverseFunctionalProperty</td>
<td>$\top \sqsubseteq \leq 1P^-$</td>
<td>$\top \sqsubseteq \leq 1\text{hasSSN}^-$</td>
</tr>
</tbody>
</table>
XML syntax

E.g., Person ⊑ ∀hasChild(Doctor ⊑ ∃hasChild.Doctor)

```xml
<owl:Class>
  <owl:intersectionOf rdf:parseType="collection">
    <owl:Class rdf:about="#Person"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild"/>
      <owl:toClass>
        <owl:unionOf rdf:parseType="collection">
          <owl:Class rdf:about="#Doctor"/>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasChild"/>
            <owl:hasClass rdf:resource="#Doctor"/>
          </owl:Restriction>
        </owl:unionOf>
        <owl:toClass>
        </owl:Restriction>
      </owl:toClass>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```
Complexity and decidability for DL’s

<table>
<thead>
<tr>
<th>Tractable</th>
<th>Decidable</th>
<th>Undecidable</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALN</td>
<td>ALCNR</td>
<td>FOL</td>
</tr>
<tr>
<td>AL</td>
<td>ALC</td>
<td>KL-One</td>
</tr>
<tr>
<td>ALNO</td>
<td>ALR</td>
<td>NIKL</td>
</tr>
<tr>
<td>PROP</td>
<td>OWL-Lite</td>
<td>OWL-DL</td>
</tr>
</tbody>
</table>

\[ P \subseteq \text{tractability threshold} \]
\[ \text{NP} \subseteq \text{decidability threshold} \]
\[ \text{PSpace} \subseteq \text{EXPTIME} \subseteq \text{NEXPTIME} \]
Conclusions

- Complexity studies on DL’s allowed to explore a wide spectrum of possibilities in the search of the best compromise between expressivity and computational complexity.
- They promoted the implementation of systems which are both efficient and expressive (even if from the theoretical point of view they have worst-case exponential complexity)
- The semantic web is laid on solid theoretical foundations.
Your turn

✓ Structural subsumption algorithm for Description logic (from the handbook).
✓ Complexity results for Description Logics
✓ Reasoning systems based on Description Logics (LOOM, BACK, KRIS, FaCT, DLP, Racer ...)

25/10/17
References

 ✓ Franz Baader, Werner Nutt, Handbook of Description Logics PDF Ch 2