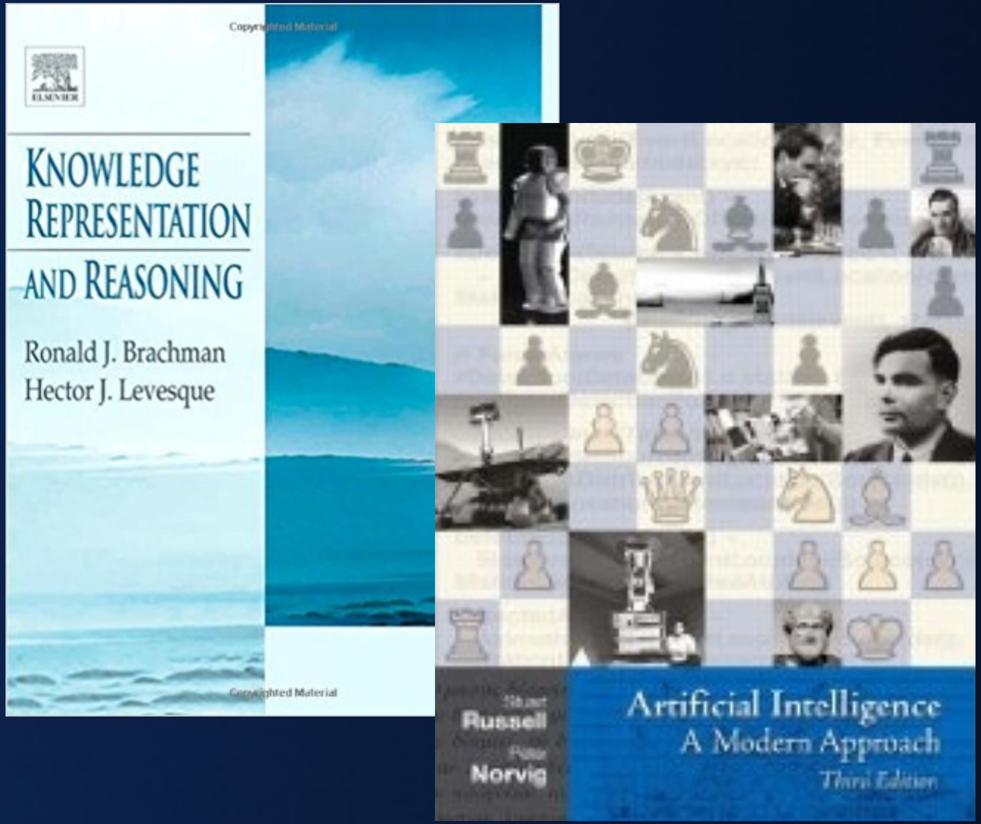


AI Fundamentals: Knowledge Representation and Reasoning

Maria Simi

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.

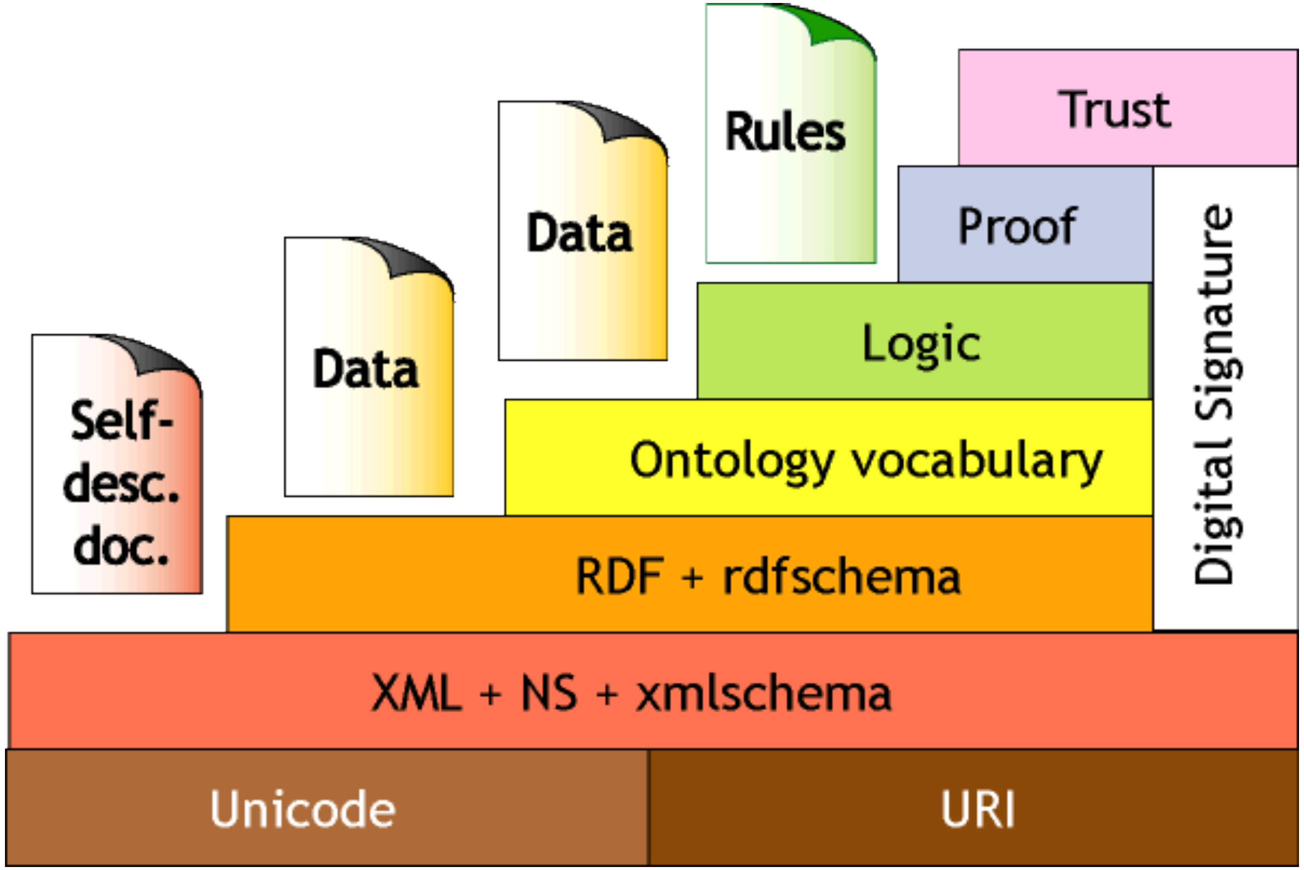
Domain ontologies

- 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 …)

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*"):

(**and** Paper (**atmost** 2 hasAuthor)
(**atleast** 2 hasAuthor)) [paper3]

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

$$\text{paper3: Paper} \sqcap (\leq 2\ \text{hasAuthor}) \sqcap (\geq 2\ \text{hasAuthor})$$

Corresponding in FOL:

$$\text{Paper(paper3)} \land$$
$$\exists x\, \text{hasAuthor(paper3}, x) \land$$
$$\exists y\, \text{hasAuthor(paper3}, y) \land x \neq y \land$$
$$\text{hasAuthor(paper3}, z) \Rightarrow (z = x) \lor (z = y)$$

Concepts, roles, individuals

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

Terms are of three types:

- *Concepts,* corresponding to unary relations

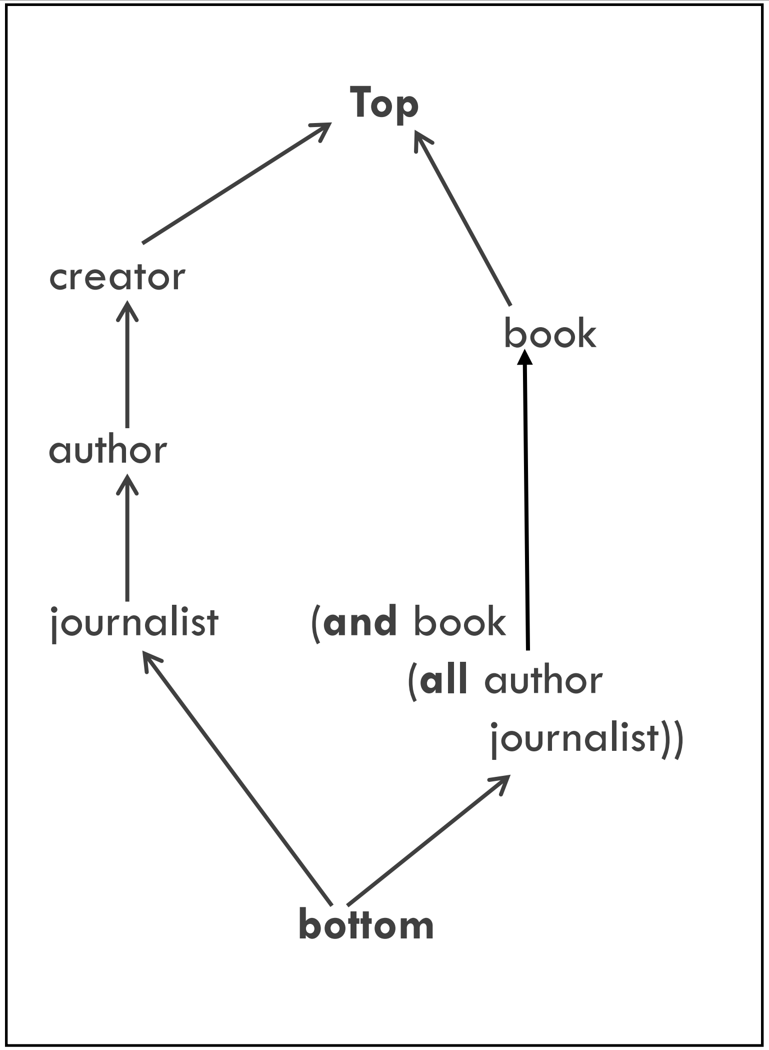
  with operators for the construction of complex concepts: and ($\sqcap$), or ($\sqcup$), not ($\neg$), all ($\forall$), some ($\exists$), atleast ($\geq n$), atmost ($\leq n$), …
- *Roles,* corresponding to binary relations

  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, I1]

author[Biagi, I2]

journalist[Biagi]

(**and** book
(**all** author
journalist))[a2]

The logic $\mathcal{AL}$: the syntax of terms

<concept> $\rightarrow A$

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

<role> $\rightarrow R$

A, B *primitive concepts*

R *primitive role*

C, D *concepts*

Examples:

Person ⊓ Female

Person ⊓ ¬Female

Person ⊓ ∃ hasChild. ⊤

Person ⊓ ∀ hasChild . Female

Person ⊓ ∀ hasChild . ⊥

Semantics of $\mathcal{AL}$

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

$\mathcal{I}$ interpretation function, assigning:

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

$\top^{I} = \Delta^{I}$ *the interpretation domain*

$\bot^{I} = \emptyset$ *the empty set*

$(\neg A)^{I} = \Delta^{I} \setminus A^{I}$ *the complement of A^{I} to Δ*

$(C \sqcap D)^{I} = C^{I} \cap D^{I}$ *the intersection of the sets*

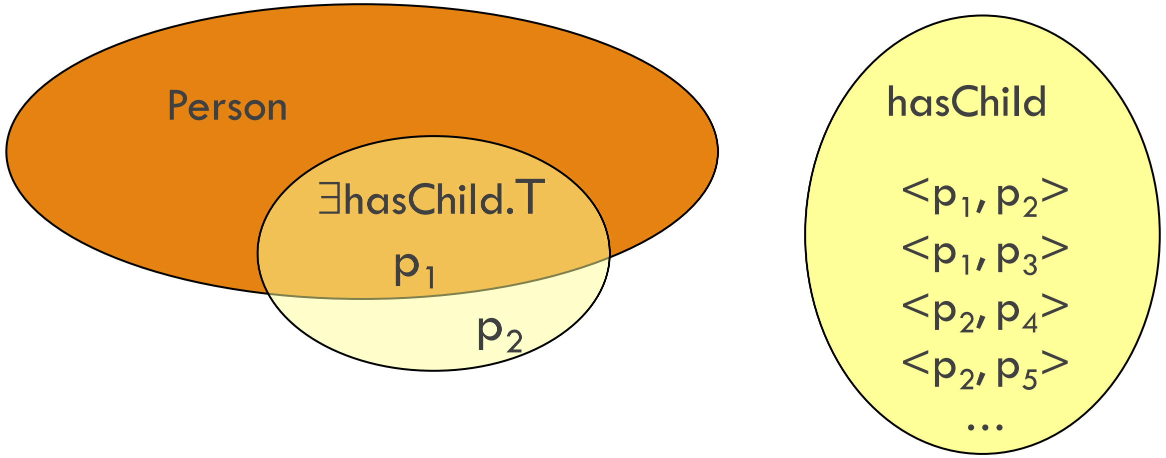
$(\forall R . C)^{I} = \{a \in \Delta^{I} \mid \forall b . (a, b) \in R^{I} \rightarrow b \in C^{I}\}$

$(\exists R . \top)^{I} = \{a \in \Delta^{I} \mid \exists b . (a, b) \in R^{I}\}$

Examples

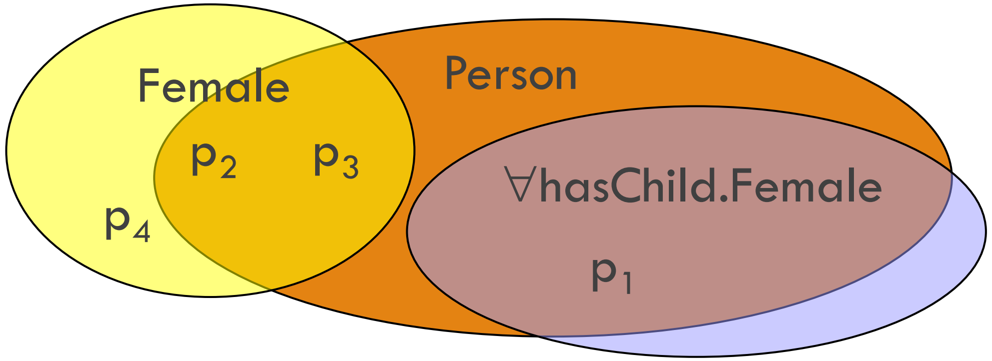
1. *Persons with a child*

   $\text{Person} \sqcap \exists\, \text{hasChild} . \top$



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

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

$\mathcal{N}$: numerical restrictions

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

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

n, integer number $\quad |\,.\,|$ set cardinality

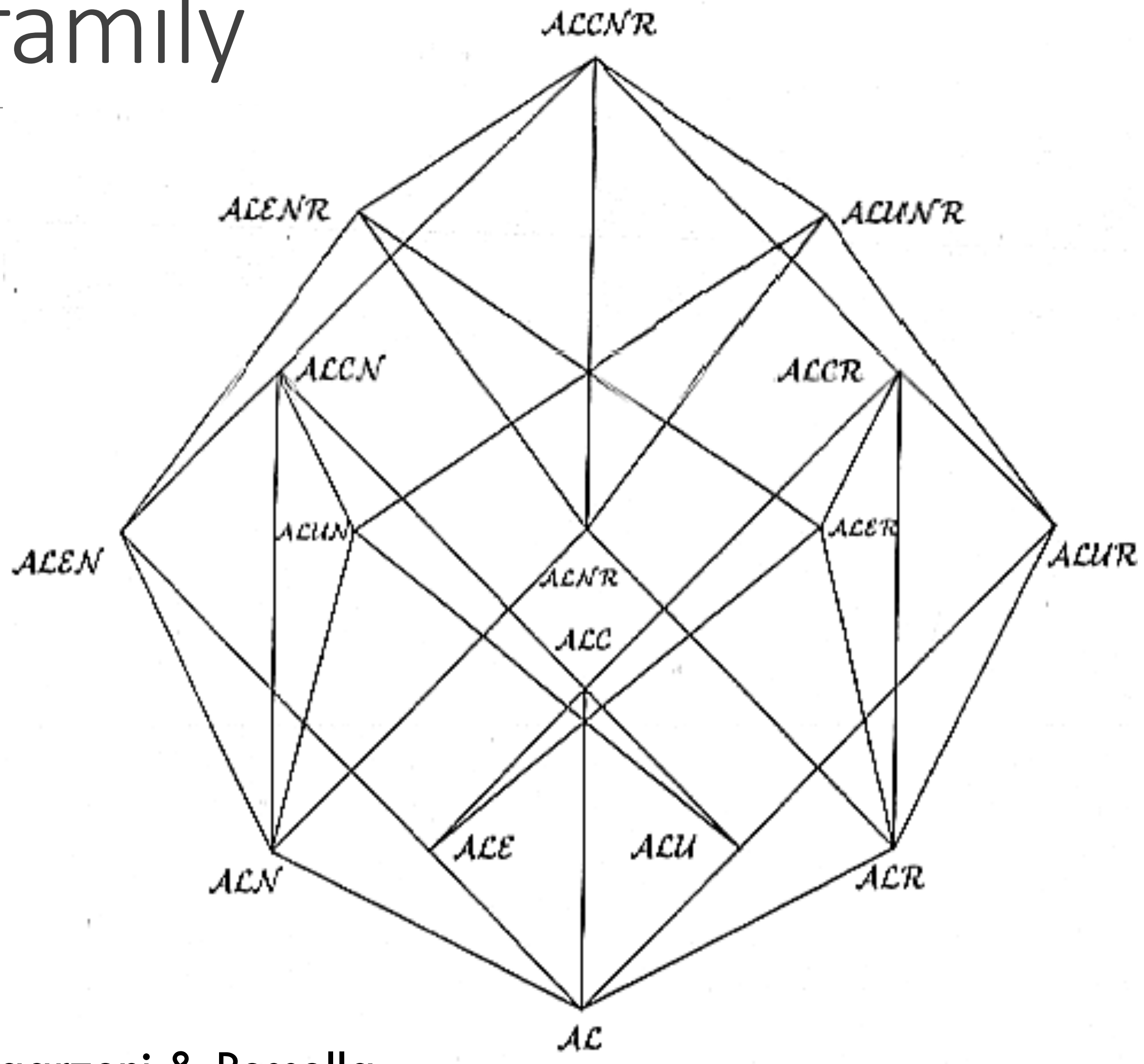
$\mathcal{C}$: full complement, $(\neg C)^I = \Delta^I \setminus C^I$

The lattice of the $\mathcal{AL}$ family

- Different description logics are obtained by adding other costructors to $\mathcal{AL}$

  $\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{N}][\mathcal{C}]$
- Not all of them are distinct
- $\mathcal{ALUE} = \mathcal{ALC}$ given that $(C \sqcup D) \equiv \neg(\neg C \sqcap \neg D)$

  and $\exists R.C \equiv \neg\forall R.\neg C$
- $\mathcal{ALCN} = \mathcal{ALUEN}$



The language for T-BOX terminology

- Terminological axioms $\mathcal{T}$

  $C \sqsubseteq D$ *inclusion of concepts,* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

  $R \sqsubseteq S$ *inclusion of roles* $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$

  $C \equiv D$ *equality of concepts,* $C^{\mathcal{I}} \equiv D^{\mathcal{I}}$

  $R \equiv S$ *equality of roles,* $R^{\mathcal{I}} \equiv S^{\mathcal{I}}$

Definitions: equalities introducing a symbol on the left

$$\text{Mother} \equiv \text{Woman} \sqcap \text{hasChild.Person}$$

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

$$
\begin{aligned}
\text{Woman} &\equiv \text{Person} \sqcap \text{Female} \\
\text{Man} &\equiv \text{Person} \sqcap \neg\, \text{Woman} \\
\text{Mother} &\equiv \text{Woman} \sqcap \exists \text{hasChild}.\text{Person} \\
\text{Father} &\equiv \text{Man} \sqcap \exists \text{hasChild}.\text{Person} \\
\text{Parent} &\equiv \text{Father} \sqcup \text{Mother} \\
\text{Grandmother} &\equiv \text{Mother} \sqcap \exists \text{hasChild}.\text{Parent} \\
\text{MotherWithManyChildren} &\equiv \text{Mother} \sqcap \geqslant 3\, \text{hasChild} \\
\text{MotherWithoutDaughter} &\equiv \text{Mother} \sqcap \forall \text{hasChild}.\neg\, \text{Woman} \\
\text{Wife} &\equiv \text{Woman} \sqcap \exists \text{hasHusband}.\text{Man}
\end{aligned}
$$

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

$$
\begin{array}{rcl}
\mathsf{Woman} & \equiv & \mathsf{Person} \sqcap \mathsf{Female} \\
\mathsf{Man} & \equiv & \mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female}) \\
\mathsf{Mother} & \equiv & (\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person} \\
\mathsf{Father} & \equiv & (\mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female})) \sqcap \exists \mathsf{hasChild}.\mathsf{Person} \\
\mathsf{Parent} & \equiv & ((\mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female})) \sqcap \exists \mathsf{hasChild}.\mathsf{Person}) \\
 & & \sqcup ((\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person}) \\
\mathsf{Grandmother} & \equiv & ((\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person}) \\
 & & \sqcap \exists \mathsf{hasChild}.(((\mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female})) \\
 & & \qquad \sqcap \exists \mathsf{hasChild}.\mathsf{Person}) \\
 & & \qquad \sqcup ((\mathsf{Person} \sqcap \mathsf{Female}) \\
 & & \qquad\quad \sqcap \exists \mathsf{hasChild}.\mathsf{Person})) \\
\mathsf{MotherWithManyChildren} & \equiv & ((\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person}) \sqcap \geqslant 3\, \mathsf{hasChild} \\
\mathsf{MotherWithoutDaughter} & \equiv & ((\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person}) \\
 & & \sqcap \forall \mathsf{hasChild}.(\neg(\mathsf{Person} \sqcap \mathsf{Female})) \\
\mathsf{Wife} & \equiv & (\mathsf{Person} \sqcap \mathsf{Female}) \\
 & & \sqcap \exists \mathsf{hasHusband}.(\mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female}))
\end{array}
$$

Specializations

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

Woman $\sqsubseteq$ 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 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:

$$t(\top, x) \mapsto true$$

$$t(\bot, x) \mapsto false$$

$$t(A, x) \mapsto A(x)$$

$$t(C \sqcap D, x) \mapsto t(C, x) \land t(D, x)$$

$$t(C \sqcup D, x) \mapsto t(C, x) \lor t(D, x)$$

$$t(\neg C, x) \mapsto \neg t(C, x)$$

$$t(\exists R \,.\, C, x) \mapsto \exists y \,.\, R(x, y) \land t(C, y)$$

$$t(\forall R \,.\, C, x) \mapsto \forall y \,.\, R(x, y) \Rightarrow t(C, y)$$

SKIPPED

Translation examples

$t\,(\text{HappyFather} \sqsubseteq \text{Man} \sqcap \exists \text{hasChild} \,.\, \text{Female}) =$

$\forall x \,.\, t\,(\text{HappyFather}, x) \Rightarrow t\,(\text{Man} \sqcap \exists \text{hasChild} \,.\, \text{Female}, x) =$

$\forall x \,.\, \text{HappyFather}(x) \Rightarrow t\,(\text{Man}, x) \land t\,(\exists \text{hasChild} \,.\, \text{Female}, x) =$

$\forall x \,.\, \text{HappyFather}(x) \Rightarrow \text{Man}(x) \land t\,(\exists \text{hasChild} \,.\, \text{Female}, x) =$

$\forall x \,.\, \text{HappyFather}(x) \Rightarrow \text{Man}(x) \land \exists y \,.\, \text{hasChild}(x, y) \land \text{Female}(y)$

$t\,(a : \text{Man} \sqcap \exists \text{hasChild} \,.\, \text{Female}) = \text{Man}(a) \land (\exists y \,.\, \text{hasChild}(a, y) \land \text{Female}(y))$

Alternative syntax (Lisp like)

$\top$	$\rightarrow$	`*top*`	$(\leq n\, R.C)$	$\rightarrow$	`(at-most n R C)`
$\bot$	$\rightarrow$	`*bottom*`	$\{a\}$	$\rightarrow$	`(one-of a)`
$\neg C$	$\rightarrow$	`(not C)`	R^{-}	$\rightarrow$	`(inv R)`
$C \sqcap D$	$\rightarrow$	`(and C D)`	$C \sqsubseteq D$	$\rightarrow$	`(implies C D)`
$C \sqcup D$	$\rightarrow$	`(or C D)`	$A = C$	$\rightarrow$	`(define-concept A C)`
$\exists R.C$	$\rightarrow$	`(some R C)`	$R \sqsubseteq P$	$\rightarrow$	`(implies-role R P)`
$\forall R.C$	$\rightarrow$	`(all R C)`	$fun(f)$	$\rightarrow$	`(functional f)`
$(\geq n\, R)$	$\rightarrow$	`(at-least n R)`	$trans(R)$	$\rightarrow$	`(transitive R)`
$(\leq n\, R)$	$\rightarrow$	`(at-most n R)`	$a{:}C$	$\rightarrow$	`(instance a C)`
$(\geq n\, R.C)$	$\rightarrow$	`(at-least n R C)`	$(a, b){:}R$	$\rightarrow$	`(related a b R)`

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 $\mathcal{I}$ satisfies $\mathcal{A}$ and $\mathcal{T}$ (therefore $\mathcal{K}$) iff it satisfies any assertion in $\mathcal{A}$ and definition in $\mathcal{T}$ ($\mathcal{I}$ 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: KBS($\mathcal{K}$) if there is a model for $\mathcal{K} = (\mathcal{T}, \mathcal{A})$?
- **Logical consequence** of a KB: $\mathcal{K} \vDash a : C$ also called **instance checking**

Typical decision problems

- **Concept satisfiability** [CS(c)]: is there an interpretation different from the empty set?
  - (father), a primitive concept, is satisfiable
  - (father $\sqcap$ ¬father) is unsatisfiable
- **Subsumption**: $\mathcal{K} \vDash C \sqsubseteq D$ (*D subsumes C*) if for every model $\mathcal{I}$ di $\mathcal{T}$, $C^I \subseteq D^I$
  - *structural subsumption:* person ***subsumes*** (person $\sqcap$ ∃hasChild.T)
  - *hybrid subsumption:*
  - person $\sqcap$ ∃hasChild.T ***subsumes*** student $\sqcap$ ∃hasChild.T if student $\sqsubseteq$ person ∈ T-BOX
- **Concept equivalence**: $\mathcal{K} \vDash C \equiv D$

Other inferential services

- **Disjointness**: $C^I \cap D^I = \emptyset$ for any model $\mathcal{I}$ of $\mathcal{T}$
- **Retrieval**: find all individuals which are instances of C, i.e. compute the set $\{ a \mid \mathcal{K} \vDash a : C \}$
- **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 $\bot$
- C and D are 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**: $\mathcal{K} \vDash C \sqsubseteq D$ (D *subsumes* C) *iff* $\mathcal{K} \cup \{a : C \sqcap \neg D\}$ is unsatisfiable
3. **Equivalence:** $\mathcal{K} \vDash D \equiv C$ *iff* $\mathcal{K} \vDash C \sqsubseteq D$ *and* $\mathcal{K} \vDash D \sqsubseteq C$
4. **Instance checking**: $\mathcal{K} \vDash a : C$ *iff* $\mathcal{K} \cup \{a : \neg C\}$ is unsatisfiable

Examples of problem reduction

1. *Are rich people happy?*
   - Happy *subsumes* Rich? $\mathcal{K} \models \text{Rich} \sqsubseteq \text{Happy}$
   - $\mathcal{K} \cup \{a: \text{Rich} \sqcap \neg\text{Happy}\}$ is unsatisfiable?
2. *Being rich and healthy is enough to be happy?*
   - $\mathcal{K} \models \text{Rich} \sqcap \text{Healthy} \sqsubseteq \text{Happy}$
   - $\mathcal{K} \cup \{a: \text{Rich} \sqcap \text{Healthy} \sqcap \neg\text{Happy}\}$ 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: $\text{Happy} \sqsubseteq \text{Rich} \sqcap \text{Healthy}$
   - $(\text{Rich} \sqcap \neg\text{Happy})$ is satisfiable?
   - $\mathcal{K} \cup \{a: \text{Rich} \sqcap \neg\text{Happy}\}$ 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 a 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).

<*concept*> $\rightarrow A$

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

<*role*> $\rightarrow R$

A *primitive concepts*

R *primitive role*

C, D *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

$$
\begin{aligned}
\neg\top &\mapsto \bot \\
\neg\bot &\mapsto \top \\
\neg\neg C &\mapsto C \\
\neg(C_1 \sqcap C_2) &\mapsto \neg C_1 \sqcup \neg C_2 \\
\neg(C_1 \sqcup C_2) &\mapsto \neg C_1 \sqcap \neg C_2
\end{aligned}
$$

$$
\begin{aligned}
\neg(\exists R.C) &\mapsto \forall R.\neg C \\
\neg(\forall R.C) &\mapsto \exists R.\neg C
\end{aligned}
$$

Constraint propagation algorithm

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

A constraint set $\mathcal{A}$ is satisfiable *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 (**clash**) is found.

For $\mathcal{ALC}$ a *clash* is one of the following types:

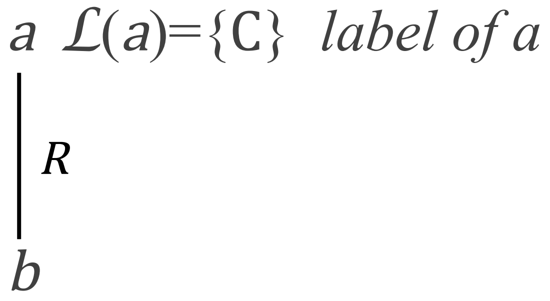
- $\{a : C, a : \neg C\}$
- $\{a : \bot\}$

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



Rules for $\mathcal{ALC}$

Rule		Description
$(\sqcap)$	if 1.	$C_1 \sqcap C_2 \in \mathcal{L}(x)$ and
	2.	$\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$
	then	$\mathcal{L}(x) \to \mathcal{L}(x) \cup \{C_1, C_2\}$
$(\sqcup)$	if 1.	$C_1 \sqcup C_2 \in \mathcal{L}(x)$ and
	2.	$\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
	then	$\mathcal{L}(x) \to \mathcal{L}(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$
$(\exists)$	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 $\mathcal{L}(\langle x, y\rangle) = \{R\}$ and $\mathcal{L}(y) = \{C\}$
$(\forall)$	if 1.	$\forall R.C \in \mathcal{L}(x)$ and
	2.	x has an R-successor y with $C \notin \mathcal{L}(y)$
	then	$\mathcal{L}(y) \to \mathcal{L}(y) \cup \{C\}$

not both in $\mathcal{L}(x)$

neither one is in $\mathcal{L}(x)$

Comments about rules

Most rules are deterministic

The rule for disjunction is **non deterministic**: its application results in alternative constraints sets.: we have a fork in the proof.

$\mathcal{A}$ is satisfiable *iff* at least one of the resulting constraints set is satisfiable.

$\mathcal{A}$ is unsatisfiable *iff* all the alternatives end up with a clash.

Example 1

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

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

Prove that: $\mathcal{A} \vDash \text{mary} : \text{female}$ or equivalently that $\mathcal{A} \cup \text{mary} : \neg\text{female}$ is unsatisfiable

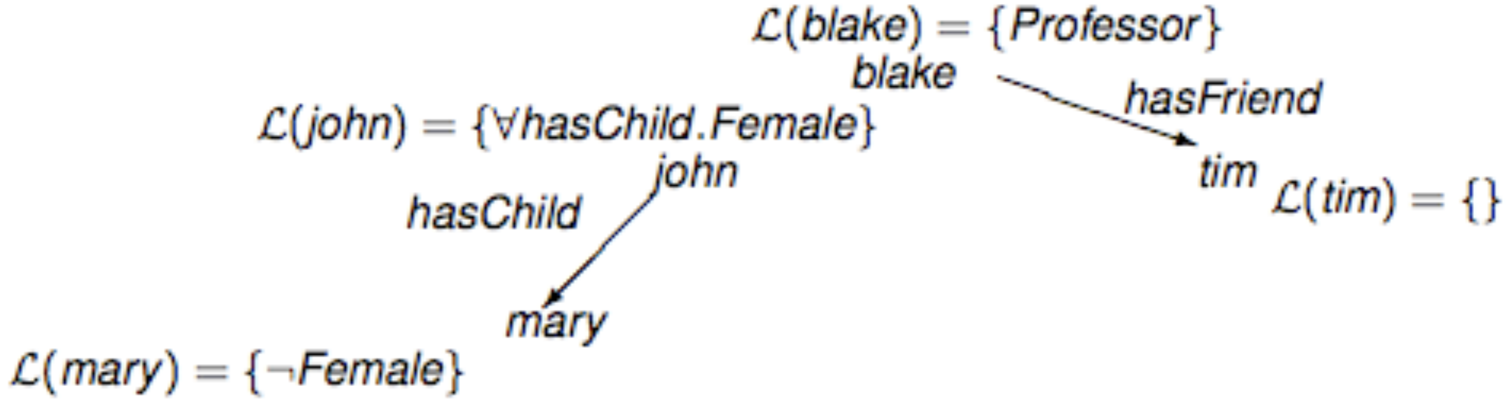
Example 1

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

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

Prove that: $\mathcal{A} \vDash \text{mary} : \text{female}$ or equivalently that $\mathcal{A} \cup \text{mary} : \neg\text{female}$ is unsatisfiable

Completion forest



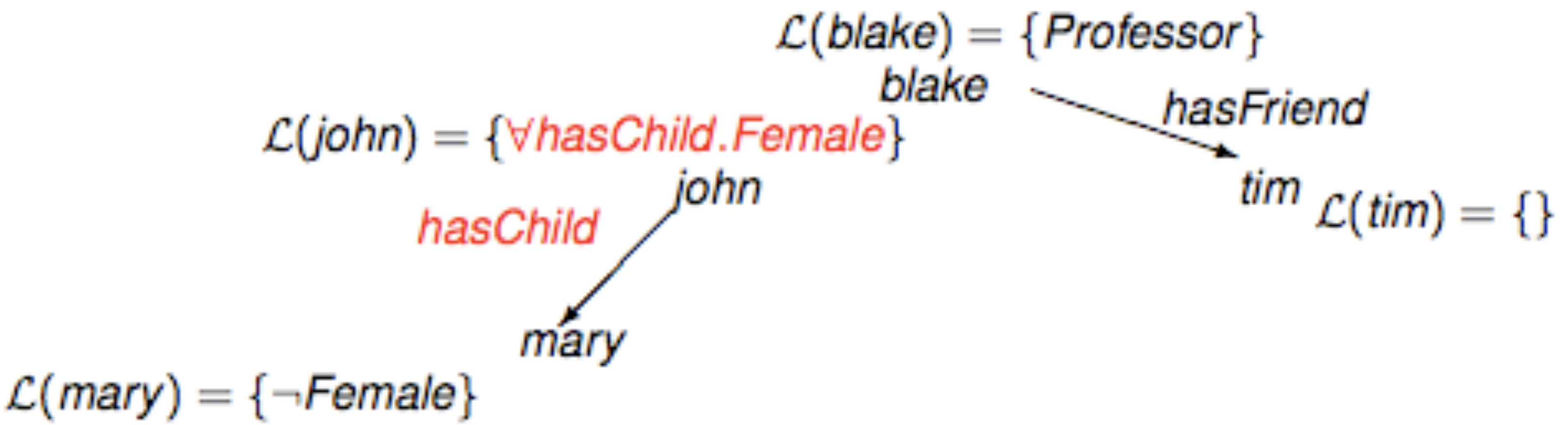
Example 1

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

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

Prove that: $\mathcal{A} \models \text{mary} : \text{female}$ or equivalently that $\mathcal{A} \cup \text{mary} : \neg\text{female}$ is unsatisfiable

Completion forest



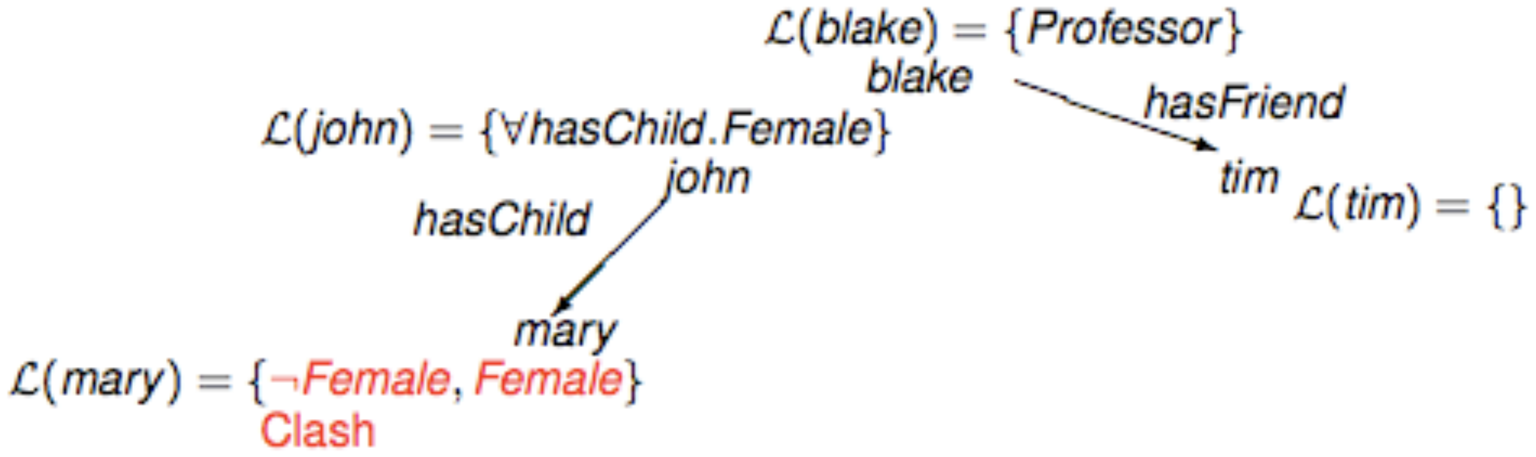
Example 1

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

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

Prove that: $\mathcal{A} \models \text{mary} : \text{female}$ or equivalently that $\mathcal{A} \cup \text{mary} : \neg\text{female}$ is unsatisfiable

Completion forest



Example 2

$\mathcal{A} = \{x : \exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D\}$ satisfiable?

$$\mathcal{L}(x) = \{\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D\}$$

x

Example 2

$\mathcal{A} = \{x : \exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D\}$ satisfiable?

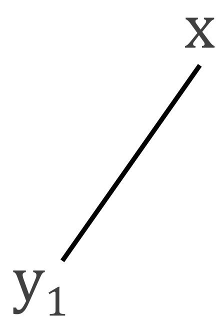
$$\mathcal{L}(x) = \{\exists R.C, \quad \forall R.(\neg C \sqcup \neg D), \quad \exists R.D\}$$

x

Example 2

$\mathcal{A} = \{x : \exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D\}$ satisfiable?

$$\mathcal{L}(x) = \{\exists R.C,\ \ \forall R.(\neg C \sqcup \neg D),\ \ \exists R.D\}$$



$$\mathcal{L}(y_1) = \{C\}$$

Example 2

$\mathcal{A} = \{x : \exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D\}$ satisfiable?

$\mathcal{L}(x) = \{\exists R.C, \ \forall R.(\neg C \sqcup \neg D), \ \exists R.D\}$

x

$\mathcal{L}(y_1) = \{C, \neg C \sqcup \neg D\}$ y_1

Example 2

$\mathcal{A} = \{x : \exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D\}$ satisfiable?

$\mathcal{L}(x) = \{\exists R.C, \ \forall R.(\neg C \sqcup \neg D), \ \exists R.D\}$

x

$\mathcal{L}(y_1)=\{C, \neg C, \neg C \sqcup \neg D\}$ y_1

Clash!

Example 2

$\mathcal{A} = \{x : \exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D\}$ satisfiable?

$\mathcal{L}(x) = \{\exists R.C, \ \forall R.(\neg C \sqcup \neg D), \ \exists R.D\}$

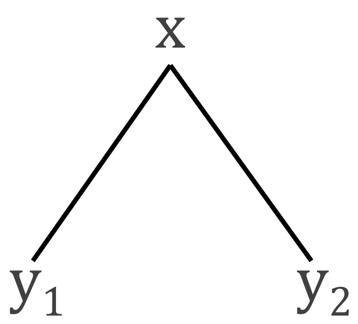
x

$\mathcal{L}(y_1) = \{C, \neg D, \neg C \sqcup \neg D\}$ y_1

Example 2

$\mathcal{A} = \{x : \exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D\}$ satisfiable?

$$\mathcal{L}(x) = \{\exists R.C,\ \ \forall R.(\neg C \sqcup \neg D),\ \ \exists R.D\}$$



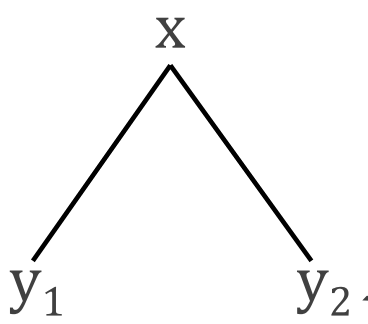
$\mathcal{L}(y_1) = \{C, \neg D, \neg C \sqcup \neg D\}$

$\mathcal{L}(y_2) = \{D\}$

Example 2

$\mathcal{A} = \{x : \exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D\}$ satisfiable?

$\mathcal{L}(x) = \{\exists R.C, \ \forall R.(\neg C \sqcup \neg D), \ \exists R.D\}$



$\mathcal{L}(y_1) = \{C, \neg D, \neg C \sqcup \neg D\}$

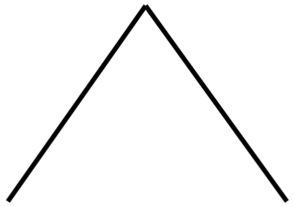
$\mathcal{L}(y_2) = \{D, \neg C \sqcup \neg D\}$

Example 2

$\mathcal{A} = \{x : \exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D\}$ satisfiable?

$$\mathcal{L}(x) = \{\exists R.C,\ \forall R.(\neg C \sqcup \neg D),\ \exists R.D\}$$

x



$\mathcal{L}(y_1)=\{C, \neg D, \neg C \sqcup \neg D\}$ y_1 y_2 $\mathcal{L}(y_2)=\{D, \neg C, \neg C \sqcup \neg D\}$

Example 2

$\mathcal{A} = \{x : \exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D\}$ satisfiable?

$\mathcal{L}(x) = \{\exists R.C,\ \forall R.(\neg C \sqcup \neg D),\ \exists R.D\}$

x

$\mathcal{L}(y_1)=\{C, \neg D, \neg C \sqcup \neg D\}$ y_1

y_2 $\mathcal{L}(y_2)=\{D, \neg C, \neg C \sqcup \neg D\}$

A is satisfiable

Model found: $\Delta^{\mathcal{I}} = \{x, y_1, y_2\}$ $C^{\mathcal{I}} = \{y_1\}$ $D^{\mathcal{I}} = \{y_2\}$ $R^{\mathcal{I}} = \{(x, y_1),(x, y_2)\}$

Example 3

$\mathcal{A} = \{x : \exists R.C \sqcap \forall R.\neg C\}$ satisfiable?

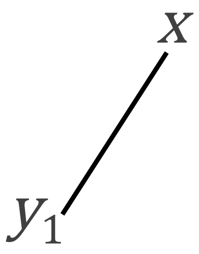
$\mathcal{L}(x) = \{\exists R.C, \ \forall R.\neg C\}$

x

Example 3

$\mathcal{A} = \{x : \exists R.C \sqcap \forall R.\neg C\}$ satisfiable?

$\mathcal{L}(x) = \{\exists R.C, \ \forall R.\neg C\}$



$\mathrm{L}(y_1) = \{C\}$

Example 3

$\mathcal{A} = \{x : \exists R.C \sqcap \forall R.\neg C\}$ satisfiable?

$\mathcal{L}(x) = \{\exists R.C, \ \forall R.\neg C\}$

x

$\mathrm{L}(y_1) = \{C, \neg C\}$ y_1

Clash!

Example 3

$\mathcal{A} = \{x : \exists R.C \sqcap \forall R.\neg C\}$ satisfiable?

$\mathcal{L}(x) = \{\exists R.C, \ \forall R.\neg C\}$

x

$L(y_1) = \{C, \neg C\}$ y_1

Clash!

- $\mathcal{A}$ 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 knoweldge 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

$\mathcal{H}$: inclusion between roles

$R \sqsubseteq S$ *iff* $R^I \subseteq S^I$

$\mathcal{Q}$: qualified numerical restrictions

$(\geq n\ R.C)^I = \{a \in \Delta^I \mid |\{b \mid (a, b) \in R^I \wedge b \in C^I\}| \geq n\}$

$(\leq n\ R.C)^I = \{a \in \Delta^I \mid |\{b \mid (a, b) \in R^I \wedge b \in C^I\}| \leq n\}$

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

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

$\mathcal{F}$: functional roles

$fun(F)$ *iff* $\forall x, y, z\ (x, y) \in F^I \wedge (x, z) \in F^I \Rightarrow y = z$

$\mathcal{R}^+$: transitive role

$(R^+)^I = \{(a, b) \mid \exists c \text{ such that } (a, c) \in R^I \wedge (c, b) \in R^I\}$

$\mathcal{S}$: $\mathcal{ALC} + \mathcal{R}^+$

SKIPPED

OWL – Ontology Web Language

OWL-DL is equivalent to

$\mathcal{SHOIN}$ =

$\mathcal{S}$: $\mathcal{ALC}$ + *transitive roles* $\mathcal{R}_+$

$\mathcal{H}$: *roles specialization*

$\mathcal{O}$: *nominals/singletons*

$\mathcal{I}$: *inverse roles*

$\mathcal{N}$: *numerical restrictions*

OWL-Lite is equivalent to

$\mathcal{SHIF}$ =

$\mathcal{S}$: $\mathcal{ALC}$ + *transitive roles* $\mathcal{R}_+$

$\mathcal{H}$: *roles specialization*

$\mathcal{I}$: *inverse roles*

$\mathcal{F}$: *functional roles*

SKIPPED

OWL syntax

Constructor	DL Syntax	Example
A (URI)	A	Conference
thing	$\top$	
nothing	$\bot$	
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Reference $\sqcap$ Journal
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Organization $\sqcup$ Institution
complementOf	$\neg C$	$\neg$MasterThesis
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{WISE, ISWC, ...}
allValuesFrom	$\forall P.C$	$\forall$date.Date
someValuesFrom	$\exists P.C$	$\exists$date.{2005}
maxCardinality	$\leqslant nP$	($\leq$ 1 location)
minCardinality	$\geqslant nP$	($\geq$ 1 publisher)

SKIPPED

OWL axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg${peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+ \sqsubseteq$ ancestor
functionalProperty	$\top \sqsubseteq \leqslant 1 P$	$\top \sqsubseteq \leqslant$1hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leqslant 1 P^-$	$\top \sqsubseteq \leqslant$1hasSSN$^-$

SKIPPED

XML syntax

E.g., Person ⊓ ∀hasChild.(Doctor ⊔ ∃hasChild.Doctor)

```
<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:intersectionOf>
</owl:Class>
```

SKIPPED

Complexity and decidability for DL's

Tractable	Decidable	Undecidable
$\mathcal{ALN}$	$\mathcal{ALCNR}$	FOL
$\mathcal{AL}$	$\mathcal{ALC}$	KL-One
	$\mathcal{ALR}$	NIKL
$\mathcal{ALNO}$	$\mathcal{ALE}$	
	PROP OWL-Lite OWL-DL	

P $\subseteq$ NP $\subseteq$ PSPACE $\subseteq$ EXPTIME $\subseteq$ NEXPTIME

tractability threshold

decidability threshold

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 …)

References

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