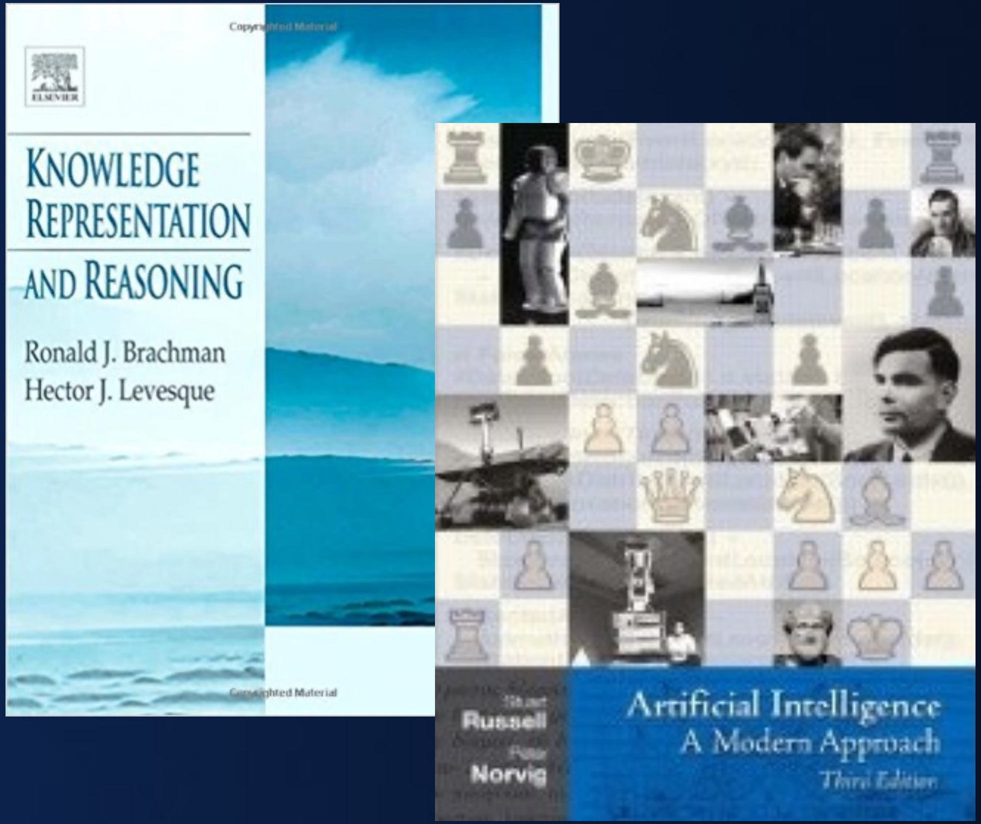


AI Fundamentals: Knowledge Representation and Reasoning

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, which exploits structure
- **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.

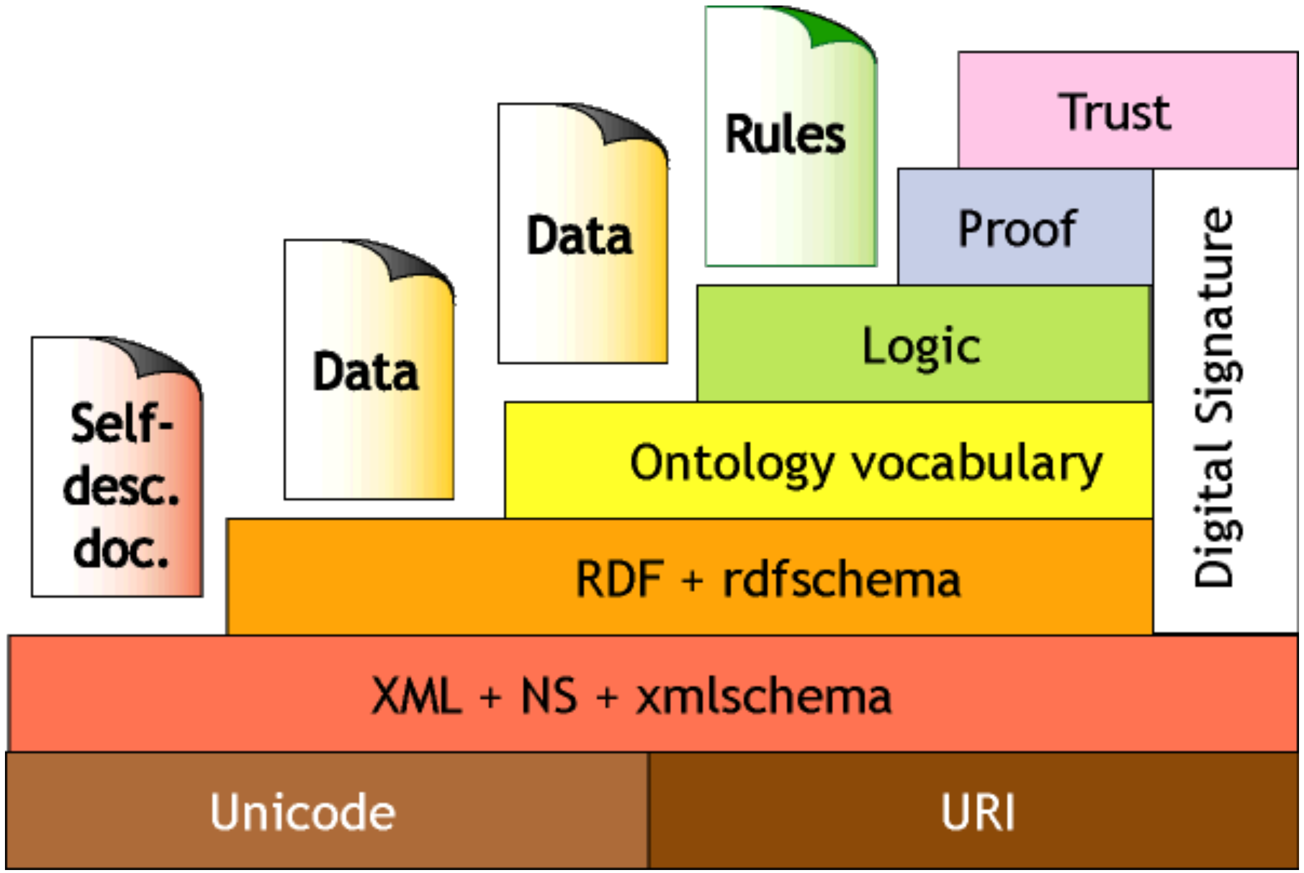
Domain ontologies

- 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 …)
- Shared ontologies are the basis for the semantic web.

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



Description logics

Can be seen as:

1. Logical counterparts of **network** knowledge representation schemas, *frames* and *semantic networks*.
   - In this formalization effort, **defaults** and exceptions are abandoned
   - 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):

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

Corresponding in FOL:

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

Concepts, roles, individuals

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

Terms (descriptions) are of three types:

- *Concepts,* corresponding to unary relations

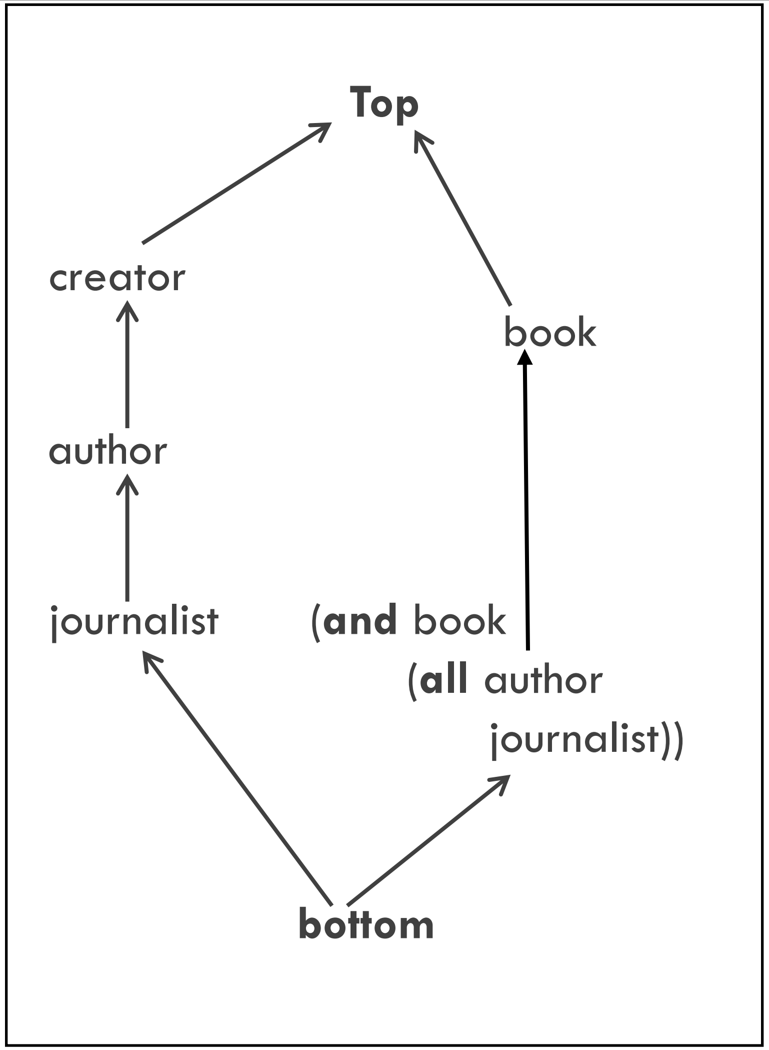
  with operators for the construction of complex concepts: and ($\sqcap$), or ($\sqcup$), not ($\neg$), all ($\forall$), some ($\exists$), atleast ($\geq n$), atmost ($\leq n$), …
- *Roles,* corresponding to binary relations

  possibly together with operators for construction of 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*)
$\mid \exists R \,.\, \top$	($\top$ for top, *weak existential*)

<role> $\rightarrow R$

A, B *primitive concepts*

R *primitive role*

C, D *concepts*

Examples:

Person $\sqcap$ Female

Person $\sqcap$ $\neg$Female

Person $\sqcap$ $\exists$ hasChild. $\top$

Person $\sqcap$ $\forall$ hasChild . Female

Person $\sqcap$ $\forall$ hasChild . $\bot$

Semantics of $\mathcal{AL}$

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

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

- Atomic concepts A: $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ set of individuals
- Atomic roles R: $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ set of pairs of individuals
- Individual constants a: $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ a specific individual

$\top^{I} = \Delta^{I}$ *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 denoted sets*

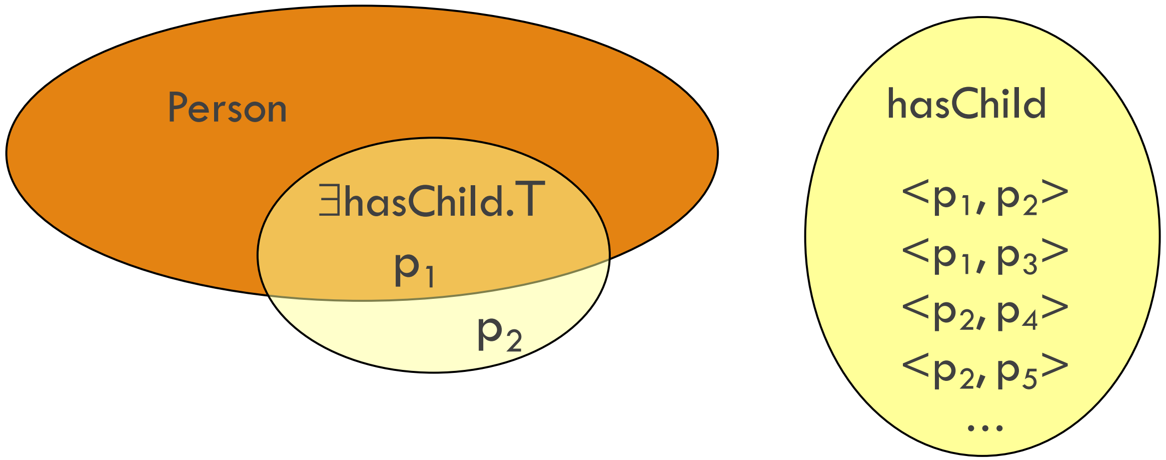
$(\forall R . C)^{I} = \{a \in \Delta^{I} \mid \forall b . (a, b) \in R^{I} \rightarrow b \in C^{I}\}$

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

Examples

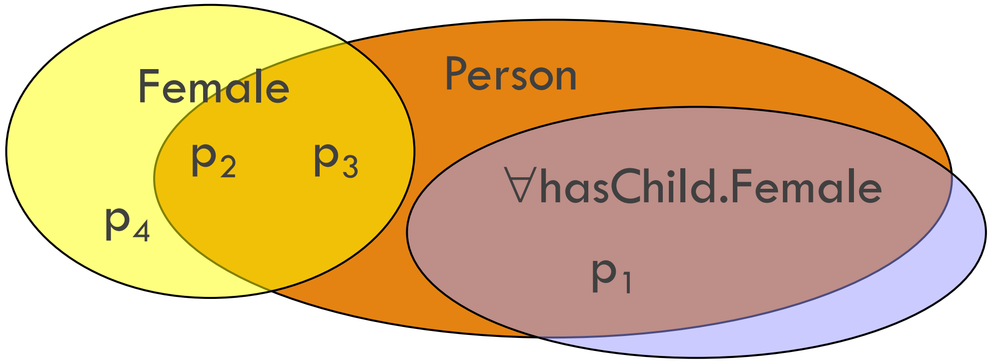
1. *Persons with a child*

   Person ⊓ ∃ hasChild . ⊤



2. *Persons with only female children*

   Person ⊓ ∀ hasChild . Female



3. *All articles that have at least one authors and whose authors are all journalists.*

   Article ⊓ ∃ hasAuthor .⊤ ⊓ ∀ hasAuthor . Journalist

More expressive logics

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

$\mathcal{E}$: 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

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

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

Example: ($\geq$ 3 *HasChild*) the set of individuals that have at least 3 children

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

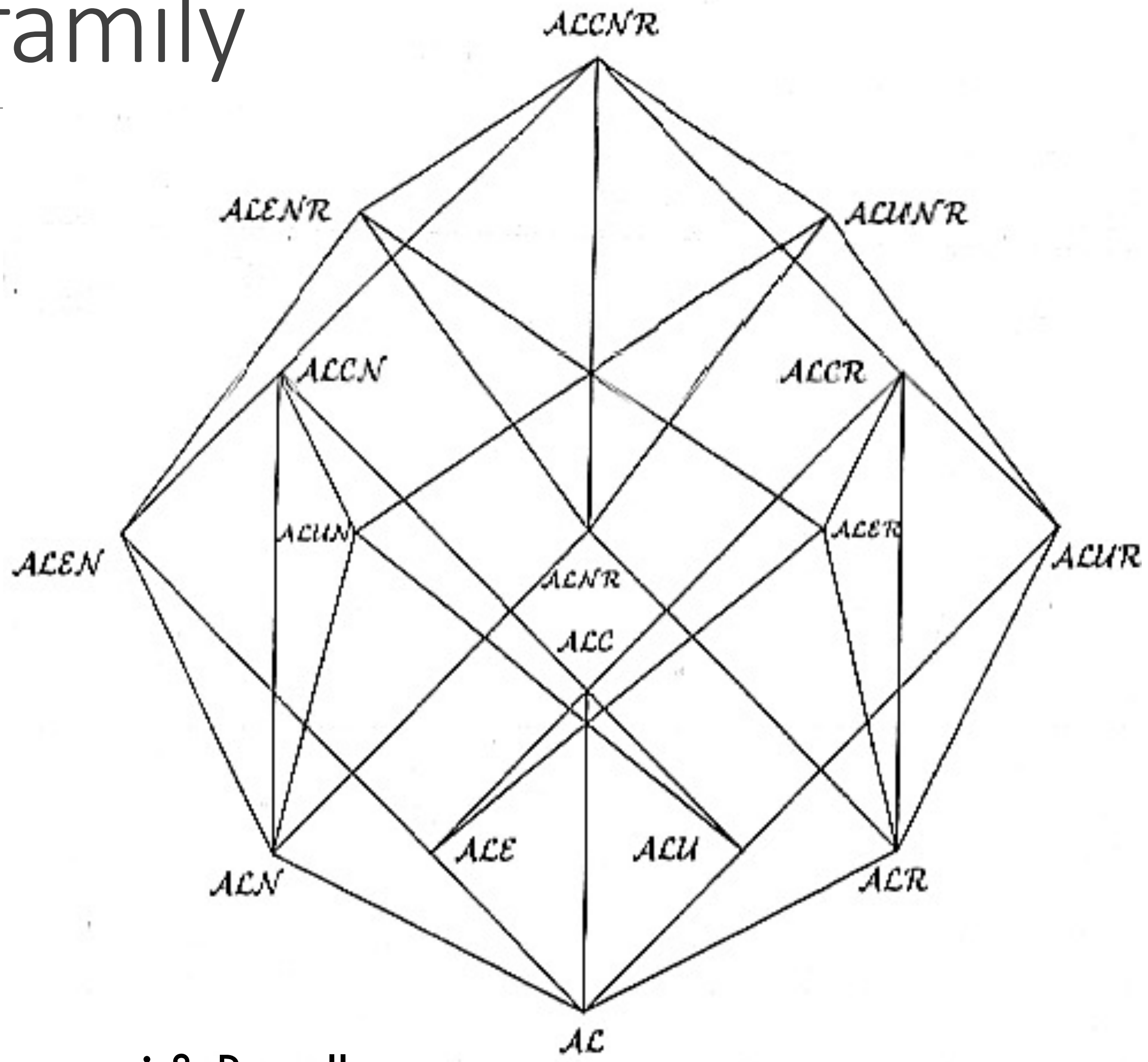
$\mathcal{C}$: full complement, $(\neg C)^I = \Delta^I \setminus C^I$

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

- Different description logics are obtained by adding other constructors 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 $\exists R.C \equiv \neg\forall R.\neg C$
- $\mathcal{ALCN}$ = $\mathcal{ALUEN}$



© Paolo Buongarzoni & Rossella

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: appear also on the left

We assume acyclic $\mathcal{T}$.

An acyclic terminology

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

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

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

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 $a^{\mathcal{I}} \in C^{\mathcal{I}}$

(b, c): *R*, assertions over roles, meaning $(b^{\mathcal{I}}, c^{\mathcal{I}}) \in R^{\mathcal{I}}$

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 into 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:

$t(\top, x)$	$\mapsto$	$true$
$t(\bot, x)$	$\mapsto$	$false$
$t(\mathrm{A}, x)$	$\mapsto$	$\mathrm{A}(x)$
$t(\mathrm{C} \sqcap \mathrm{D}, x)$	$\mapsto$	$t(\mathrm{C}, x) \wedge t(\mathrm{D}, x)$
$t(\mathrm{C} \sqcup \mathrm{D}, x)$	$\mapsto$	$t(\mathrm{C}, x) \vee t(\mathrm{D}, x)$
$t(\neg \mathrm{C}, x)$	$\mapsto$	$\neg t(\mathrm{C}, x)$
$t(\exists \mathrm{R.C}, x)$	$\mapsto$	$\exists y \,.\, R(x, y) \wedge t(\mathrm{C}, y)$
$t(\forall \mathrm{R.C}, x)$	$\mapsto$	$\forall y \,.\, R(x, y) \Rightarrow t(\mathrm{C}, y)$

Translation examples

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

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

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

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

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

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

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 or description)
- 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: $\mathcal{K} \vDash a : \mathrm{C}$ also called **instance checking**

Typical DL decision problems

- **Concept satisfiability** [CS(c)]: is there an interpretation different from the empty set?

  (father), a primitive concept, is satisfiable

  (father $\sqcap$ $\neg$father) is unsatisfiable

- **Subsumption**: $\mathcal{K} \vDash C \sqsubseteq D$ (*D subsumes C*) if for every model ***I* of** $\mathcal{T}$, $C^{I} \subseteq D^{I}$

  structural subsumption: person ***subsumes*** (person $\sqcap$ $\exists$hasChild.T)

  hybrid subsumption:

  person $\sqcap$ $\exists$hasChild.$\top$ ***subsumes*** student $\sqcap$ $\exists$hasChild.$\top$ if student $\sqsubseteq$ person $\in$ T-BOX

- **Concept equivalence**: $\mathcal{K} \vDash C \equiv D$

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.

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 temporarily added to $\mathcal{A}$.
2. **Subsumption**: $\mathcal{K} \vDash C \sqsubseteq D$ (D *subsumes* C) *iff* $C \sqcap \neg D$ is inconsistent *iff* $\mathcal{K} \cup \{a : C \sqcap \neg D\}$ is unsatisfiable
3. **Equivalence:** $\mathcal{K} \vDash C \equiv D$ *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?

An inference system for deciding KBS in DL

The most used method is a technique for verifying **satisfiability of a KB** (KBS).

- It is a variant of a method for natural deduction, called *semantic tableaux.* It applies **constraint propagation.**
- *Basic idea:* each formula in KB is a constraint on possible interpretations.
- 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**)
$\mid \neg C$	(*full negation*)
$\mid C \sqcap D$	(*intersection*)
$\mid C \sqcup D$	(*union*)
$\mid \forall R.C$	(*value restriction*)
$\mid \exists R.C$	(*full 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.

The label is a set of the constraints that apply to a .

- 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 the R relation between them

$$
\begin{array}{l}
a \quad \mathcal{L}(a)=\{\mathrm{C}\} \quad \textit{label of } a \\
\big| \; R \\
b
\end{array}
$$

Rules for constraint propagation in $\mathcal{ALC}$

Rule		Description	
$(\sqcap)$	if 1.	$C_1 \sqcap C_2 \in \mathcal{L}(x)$ and	
	2.	$\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$	*not both in* $\mathcal{L}(x)$
	then	$\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$	
$(\sqcup)$	if 1.	$C_1 \sqcup C_2 \in \mathcal{L}(x)$ and	
	2.	$\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$	*neither one is in* $\mathcal{L}(x)$
	then	$\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C\}$ 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	$\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{C\}$	

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 constraint 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: $\mathcal{A} \models \text{mary} : \text{female}$ or equivalently that

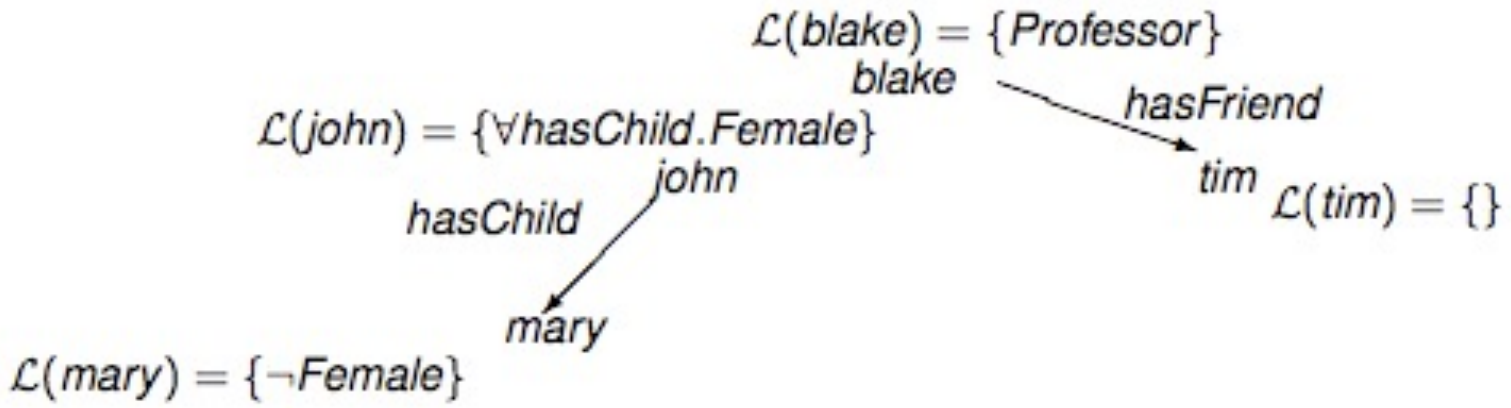
$\mathcal{A} \cup \{\text{mary} : \neg\text{female}\}$ is unsatisfiable

Example 1

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

Prove: $\mathcal{A} \cup \{\text{mary} : \neg\text{female}\}$ is unsatisfiable

Completion forest

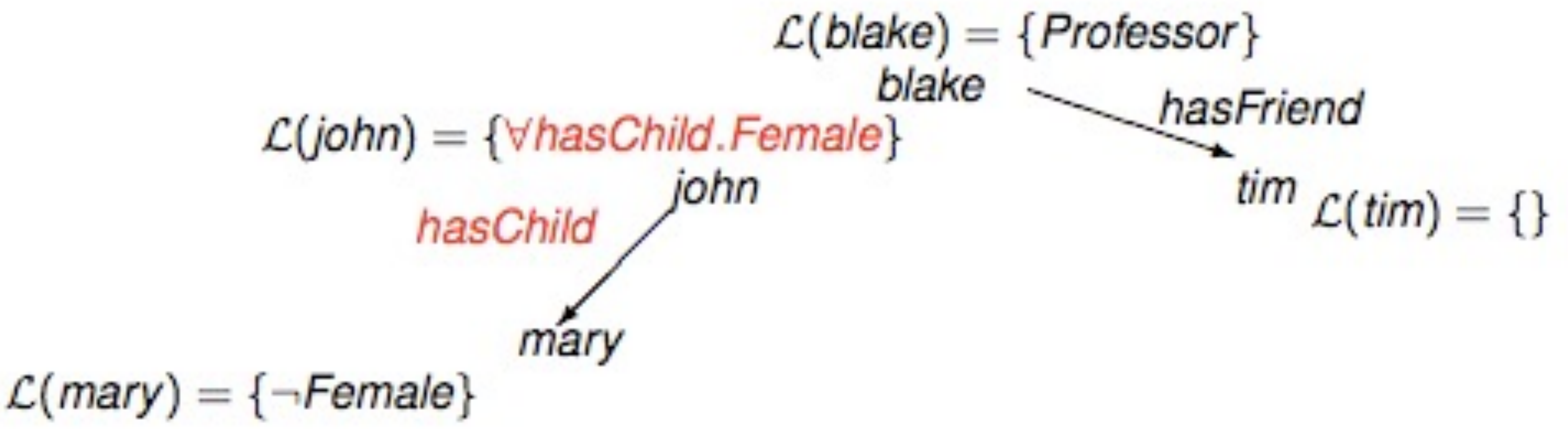


Example 1

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

Prove: $\mathcal{A} \cup \{\text{mary} : \neg\text{female}\}$ is unsatisfiable

Completion forest

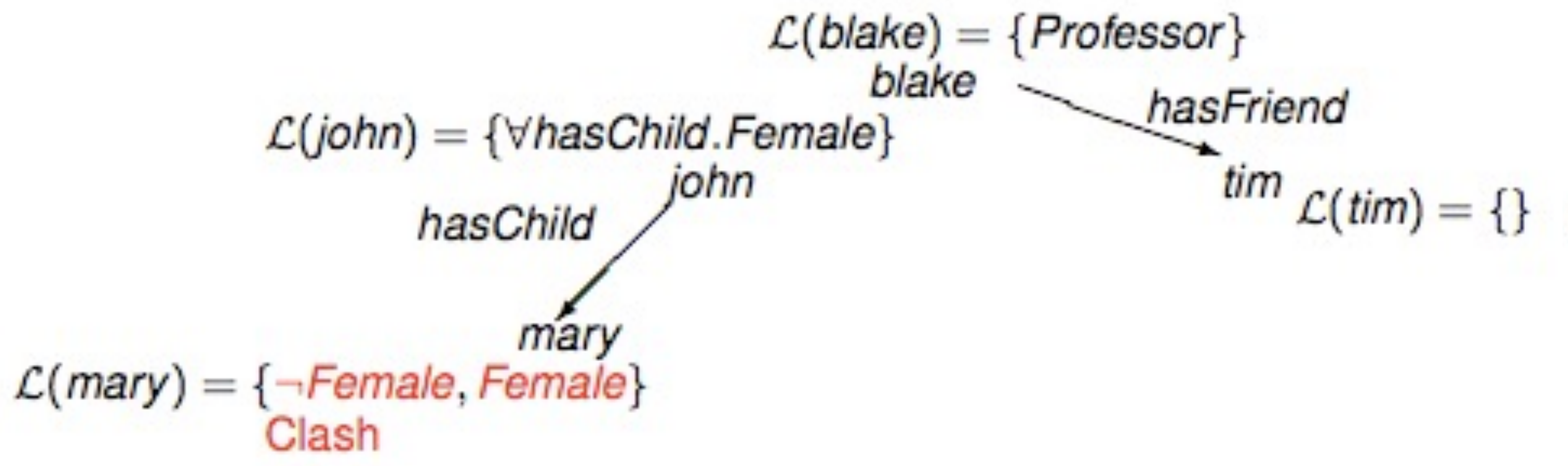


Example 1

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

Prove: $\mathcal{A} \cup \{\text{mary} : \neg\text{female}\}$ is unsatisfiable

Completion forest



Example 2

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

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

x

Example 2

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

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

x

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

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

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

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

$$x$$

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

Example 2

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

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

$$\begin{tikzcd}
 & x \arrow[dl, dash] \arrow[dr, dash] & \\
\mathcal{L}(y_1)=\{C, \neg D, \neg C \sqcup \neg D\}\ \ y_1 & & y_2\ \mathcal{L}(y_2)=\{D, \neg C \sqcup \neg D\}
\end{tikzcd}$$

Example 2

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

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

x

$\mathcal{L}(\mathrm{y}_1)=\{C, \neg D, \neg C \sqcup \neg D\}$ y_1

y_2 $\mathcal{L}(\mathrm{y}_2)=\{D, \neg C, \neg C \sqcup \neg D\}$

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

$\mathcal{L}(\mathrm{y}_1)=\{C, \neg D, \neg C \sqcup \neg D\}$ y_1 $\qquad$ y_2 $\mathcal{L}(\mathrm{y}_2)=\{D, \neg C, \neg C \sqcup \neg D\}$

A is satisfiable

Model found: $\Delta^{\mathcal{I}} = \{x,\ y_1,\ y_2\}$ $\quad$ C$^{\mathcal{I}}$ = $\{y_1\}$ $\quad$ D$^{\mathcal{I}}$ = $\{y_2\}$ $\quad$ 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?

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

x

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

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

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

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

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

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*

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	$\neg C$	¬MasterThesis
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{WISE, ISWC, ...}
allValuesFrom	$\forall P.C$	∀date.Date
someValuesFrom	$\exists P.C$	∃date.{2005}
maxCardinality	$\leqslant nP$	(≤ 1 location)
minCardinality	$\geqslant nP$	(≥ 1 publisher)

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$1hasSSN$^-$

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>
```

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.

References

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