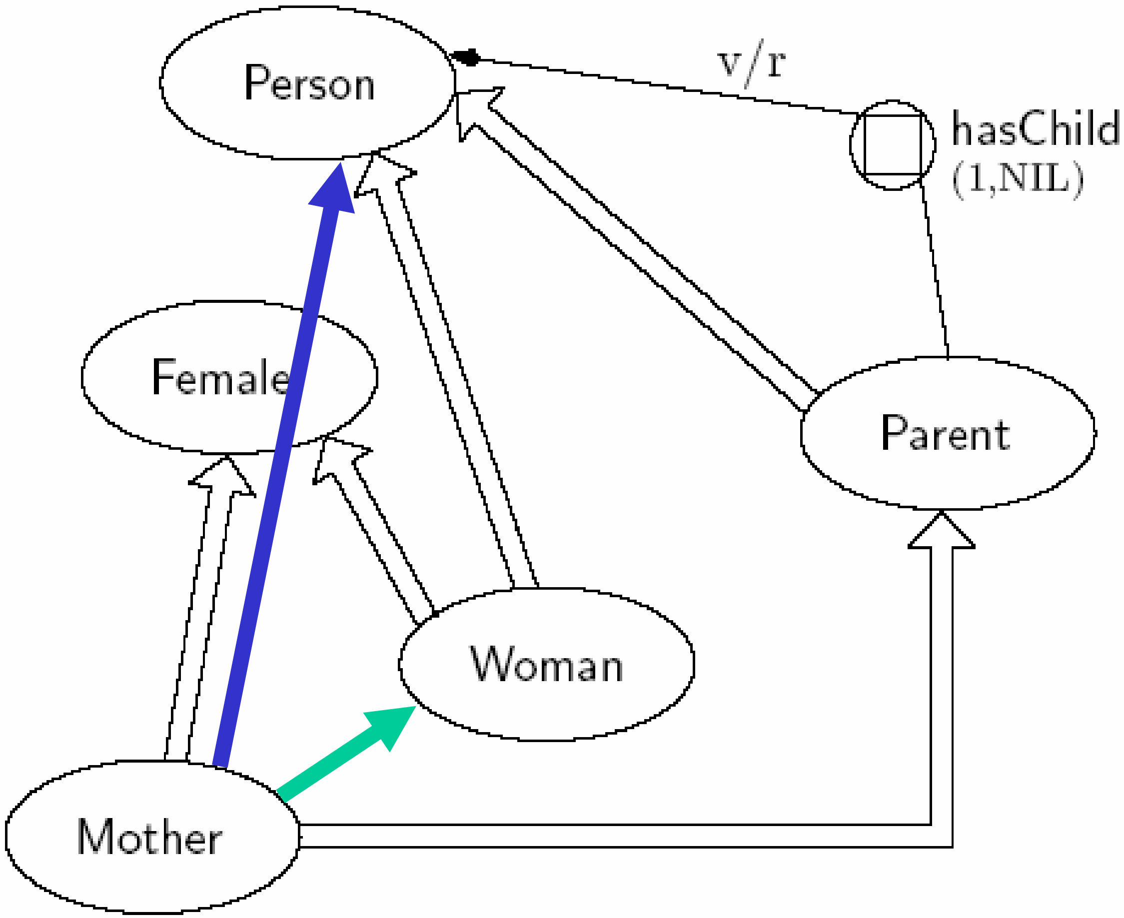
Part I : Description Logics

Naouel KARAM
karam@isima.fr

- Introduction
- Concept descriptions
- Knowledge bases
- Reasoning
- Non standard reasoning

DL origins

- Semantic Networks



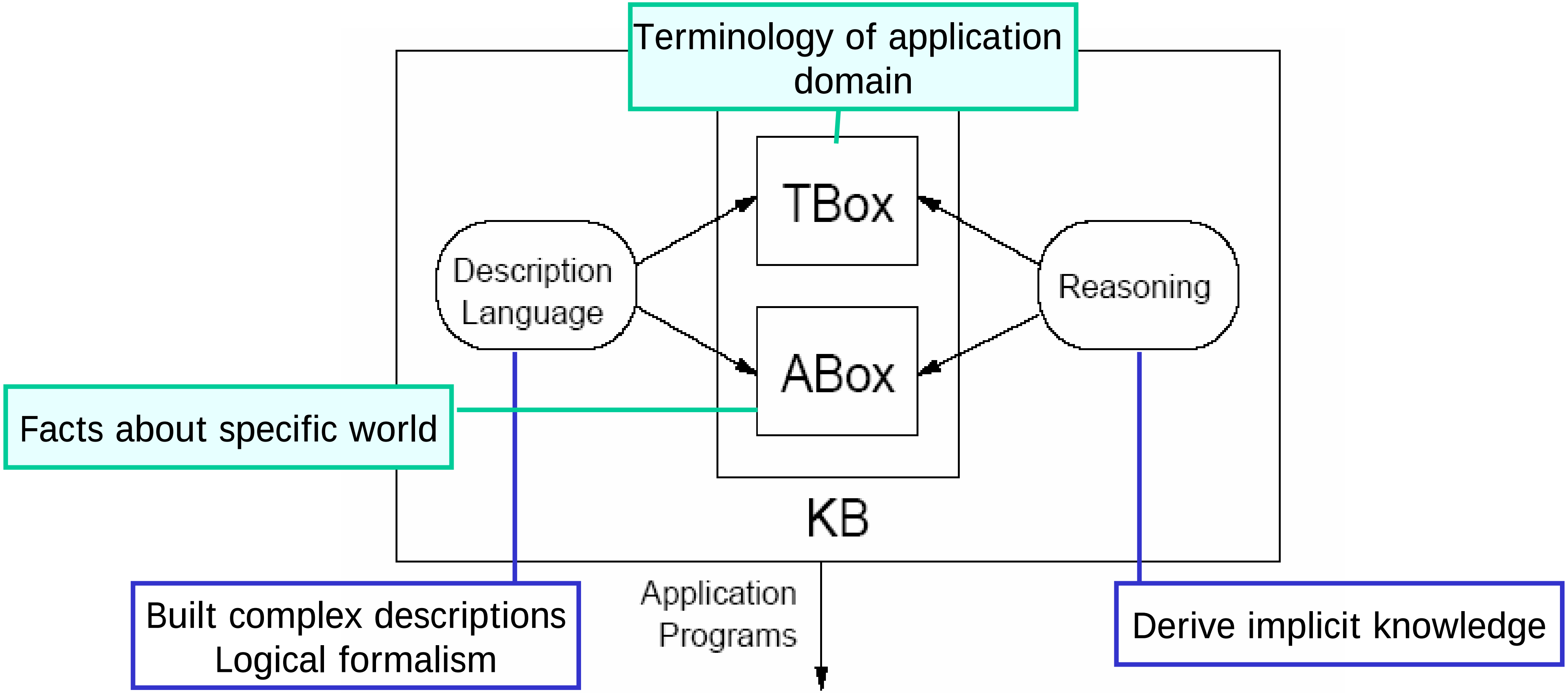
- Problem: missing semantics (complex networks)
- Solution: use a logical formalism rather than a network

DL definition

- Descendents of semantics networks, frame-based systems, and KL-ONE

- Family of logic-based knowledge representation (KR) formalisms well-suited for the representation of and reasoning about
  - terminological knowledge
  - ontologies
  - database schemata
  - ...

Architecture of a DL system



Overview of the tutorial

- Introduction
- Concept descriptions
- Knowledge bases
- Reasoning
- Non standard reasoning

Concept descriptions

- The conceptual knowledge of an application domain is represented by:
  - **Concepts** : interpreted as a set of individuals
  - **Roles** : interpreted as relations between individuals

- Complex concept descriptions can be built from atomic ones using concept constructors ($\sqcap$, $\sqcup$, $\forall$, $\exists$,...) :

$$\text{Person} \sqcap \text{Male} \sqcap \exists \text{hasChild}.\text{Person}$$

concept names assign a name to a set of individuals

role names assign a name to relations between individuals

concept constructors connect concept names and role names

The basic description language AL

- Concept descriptions are formed according to the following syntax rules:

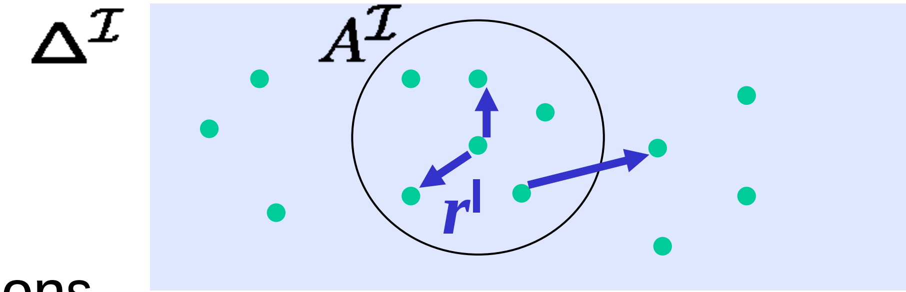
$C, D \rightarrow$	$\top \mid$	top concept
	$\bot \mid$	bottom concept
	$A \mid$	atomic concept
	$\neg A \mid$	atomic negation
	$C \sqcap D \mid$	conjunction
	$\forall r.C \mid$	value restriction
	$\exists r.\top$	limited existential quantification

- Examples of AL-concept descriptions

Person $\sqcap$ $\exists$hasChild.$\top$	persons that have at least one child
Person $\sqcap$ $\forall$hasChild.$\neg$Male	persons all of whose children are not male
Person $\sqcap$ $\forall$hasChild.$\bot$	persons without a child

Formal semantics for AL-concept descriptions

- Semantics based on **interpretation** $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$
  - A non empty set $\Delta^{\mathcal{I}}$ (the domain of the interpretation)
  - An interpretation function $\cdot^{\mathcal{I}}$
    - an atomic concept A: a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
    - an atomic role r: a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$



- Inductive extension to concept descriptions

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \sqcap D^{\mathcal{I}} \\
(\forall r.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \forall y : (x,y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\} \\
(\exists r.\top)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \exists y : (x,y) \in r^{\mathcal{I}}\}
\end{aligned}
$$

The family of AL-languages

- More expressive languages can be obtained by adding further constructors
  - Union of concepts (**U**)

    written $C \sqcup D$

    interpreted as $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$

  - Full existential quantification (**E**)

    written $\exists r.C$

    interpreted as $(\exists r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$

  - Negation (**C**)

    written $\neg C$

    interpreted as $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$

The family of AL-languages

– Number restrictions (**N**)

written $\geq n\ r$ (at-least restriction)

$\leq n\ r$ (at-most restriction)

interpreted as

$$
\begin{aligned}
(\geq n\ r)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \in \Delta^{\mathcal{I}} \mid (x, y) \in r^{\mathcal{I}}\} \geq n\} \\
(\leq n\ r)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \in \Delta^{\mathcal{I}} \mid (x, y) \in r^{\mathcal{I}}\} \leq n\}
\end{aligned}
$$

- Extending AL by any subset of the above operators yields a particular language identified by a string of the form

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

The family of AL-languages

Concept constructors	$\mathcal{AL}$	$\mathcal{ALN}$	$\mathcal{ALE}$	$\mathcal{ALEN}$	$\mathcal{ALC}$
$\top$	×	×	×	×	×
$\bot$	×	×	×	×	×
$\neg A$	×	×	×	×	×
$\neg C$					×
$C \sqcap D$	×	×	×	×	×
$C \sqcup D$					×
$\forall r.C$	×	×	×	×	×
$\exists r.\top$	×	×	×	×	×
$\exists r.C$			×	×	×
$\geq n\ r$		×		×	
$\leq n\ r$		×		×	

The family of $\mathcal{AL}$-languages

- Based on their semantics, prove the equivalence between the languages:

$\mathcal{ALC}$ and $\mathcal{ALUE}$

$\mathcal{ALCN}$ and $\mathcal{ALUEN}$

Union and full existential quantification can be expressed using negation, because of the equivalences:

$$C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$$
$$\exists r.C \equiv \neg \forall r.\neg C$$

Overview of the tutorial

- Introduction
- Concept descriptions
- Knowledge bases
- Reasoning
- Non standard reasoning

DL knowledge bases

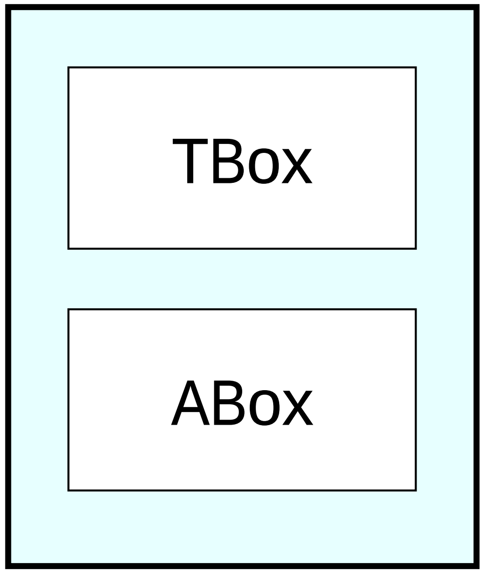
- Formed by two components: The intentional one, called **TBox** and the extensional one called **ABox**.

- TBox (T)
  - Schema describing the concepts of the application domain, their properties and the relations between them.

- ABox (A)
  - Partial instantiation of the schema describing assertions on individuals.

- A knowledge base is noted

$$\Sigma = (\mathcal{T}, \mathcal{A})$$



KB

Intentional knowledge

- A TBox is a set of terminological axioms having one of the forms:

Primitive concept
necessary conditions → $A \dot{\preceq} C$ Primitive Concept specification

Defined concept
necessary and sufficient conditions → $A \doteq C$ Concept definition

Concepts not appearing in the left-hand side of any terminological axiom are called atomic concepts

- A more general kind of TBox, called *free-TBox* is obtained by admitting terminological axioms of the form: $C \dot{\preceq} D$ and $C \doteq D$

- An example of a TBox from the family domain

$$\begin{aligned}
\text{Man} &\doteq \text{Human} \sqcap \text{Male} \\
\text{Parent} &\doteq \text{Human} \sqcap \exists \text{hasChild}.\text{Human} \\
\text{Father} &\doteq \text{Man} \sqcap \text{Parent} \\
\text{HappyFather} &\doteq \text{Father} \sqcap \forall \text{hasChild}.\neg \text{Male}
\end{aligned}$$

Cycles

- A concept name A directly uses a concept B in a TBox $\mathcal{T}$ if B appears on the right-hand side of the definition of A.
- We call uses the transitive closure of the relation directly uses.
- $\mathcal{T}$ is called acyclic iff there does not exist a concept name in $\mathcal{T}$ that uses itself.

A cyclic TBox:

$$\begin{array}{lcl} A_1 & \doteq & A_2 \sqcap \exists r.A_4 \\ A_2 & \doteq & \exists r.A_3 \sqcap A_5 \\ A_3 & \doteq & A_1 \end{array}$$

- Expansion of an acyclic TBox

$$\begin{array}{rcl} \mathsf{Man} & \doteq & \mathsf{Human} \sqcap \mathsf{Male} \\ \mathsf{Parent} & \doteq & \mathsf{Human} \sqcap \exists \mathsf{hasChild}.\mathsf{Human} \\ \mathsf{Father} & \doteq & \mathsf{Man} \sqcap \mathsf{Parent} \\ \mathsf{HappyFather} & \doteq & \mathsf{Father} \sqcap \forall \mathsf{hasChild}.\neg \mathsf{Male} \end{array}$$

$$\begin{array}{rcl} \mathsf{Father} & \doteq & \mathsf{Human} \sqcap \mathsf{Male} \sqcap \exists \mathsf{hasChild}.\mathsf{Human} \\ \mathsf{HappyFather} & \doteq & \mathsf{Human} \sqcap \mathsf{Male} \sqcap \exists \mathsf{hasChild}.\mathsf{Human} \sqcap \forall \mathsf{hasChild}.\neg \mathsf{Male} \end{array}$$

The expansion contains only atomic concepts in the right-hand side of each definition

TBoxes with primitive specifications

- Primitive specifications are used when we are unable to define completely a concept.
- For example, if the concept *Man* could not be defined in detail, one can require that every man is a human with the primitive specification:

$$\text{Man} \stackrel{.}{\preceq} \text{Human}$$

- A TBox $\mathcal{T}$ containing primitive specifications can be transformed into a regular TBox $\hat{\mathcal{T}}$ with only definitions by adding to primitive specifications a concept standing for the absent part of the definition.

$$\text{Man} \doteq \text{Human} \sqcap \overline{\text{Man}}$$

Qualities that distinguish a man among humans

- $\hat{\mathcal{T}}$ is called the normalization of $\mathcal{T}$

Semantics

- An interpretation I satisfies the terminological axiom:

$$
\begin{aligned}
A \,\dot{\preceq}\, C \quad &\text{if} \quad A^{\mathcal{I}} \subseteq C^{\mathcal{I}} \\
A \doteq C \quad &\text{if} \quad A^{\mathcal{I}} = C^{\mathcal{I}} \\
C \,\dot{\preceq}\, D \quad &\text{if} \quad C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \\
C \doteq D \quad &\text{if} \quad C^{\mathcal{I}} = D^{\mathcal{I}}
\end{aligned}
$$

- An interpretation I is a model of a TBox T iff it satisfies each terminological axiom in T.

Extensional knowledge

- An ABox is a set of assertions having one of the forms:

$C(a)$	concept assertion
$r(a,b)$	role assertion

- An example of an ABox from the family domain

  Man(PETER)
  Man(MARC)
  hasChild(PETER, MARC)

- Semantics
  - Extend interpretations to individual names: an interpretation I maps an individual name a to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
  - An interpretation I satisfies the assertion:

$$C(a) \text{ if } a^{\mathcal{I}} \in C^{\mathcal{I}}$$

$$r(a,b) \text{ if } (a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$$

  - An interpretation I is a model of an ABox A if it satisfies each assertion in A

Individual names in the description language

- Individual names can appear in the TBox
  - The *one-of* constructor $(\mathcal{O})$

    written $\{a_1, \ldots, a_n\}$

    interpreted as $\{a_1, \ldots, a_n\}^{\mathcal{I}} = \{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$

    example: $\{CHINA, FRANCE, RUSSIA, UK, USA\}$
  - In a language with the union constructor, a constructor for singleton sets adds sufficient expressiveness to describe arbitrary sets as

    $\{a_1, \ldots, a_n\}$ is equivalent to $\{a_1\} \sqcup \ldots \sqcup \{a_n\}$

  - The *fills* constructor

    written $r : a$

    interpreted as $(r : a)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid (d, a^{\mathcal{I}}) \in r^{\mathcal{I}}\}$
  - In a language with singleton sets and full existential quantification "*fills*" does not add anything new as

    $r : a$ is equivalent to $\exists r.\{a\}$

Overview of the tutorial

- Introduction
- Concept descriptions
- Knowledge bases
- Reasoning

Reasoning tasks for TBoxes

- Concept satisfibility (written $\mathcal{T} \not\models C \equiv \bot$)
  - A concept C is satisfiable with respect to T if there exists a model I of T such that $C^{\mathcal{I}}$ is nonempty.

- Subsumption (written $\mathcal{T} \models C \sqsubseteq D$ or $C \sqsubseteq_{\mathcal{T}} D$)
  - A concept C is subsumed by a concept D with respect to T if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model I of T.
  - Example: Parent subsume Father

- Equivalence (written $\mathcal{T} \models C \equiv D$ or $C \equiv_{\mathcal{T}} D$)
  - Two concepts C and D are equivalent with respect to T if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model I of T.

- Disjointness
  - Two concepts C and D are disjoint with respect to T if $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ for every model I of T.

Reductions

- Reduction to subsumption

(i) C is unsatisfiable $\Leftrightarrow$ C is subsumed by $\bot$;
(ii) C and D are equivalent $\Leftrightarrow$ C is subsumed by D and D is subsumed by C;
(iii) C and D are disjoint $\Leftrightarrow$ $C \sqcap D$ is subsumed by $\bot$.

- Reduction to satisfiability (systems allowing negation)

(i) C is subsumed by D $\Leftrightarrow$ $C \sqcap \neg D$ is unsatisfiable;
(ii) C and D are equivalent $\Leftrightarrow$ both $(C \sqcap \neg D)$ and $(\neg C \sqcap D)$ are unsatisfiable;
(iii) C and D are disjoint $\Leftrightarrow$ $C \sqcap D$ is unsatisfiable.

Reasoning tasks for ABoxes

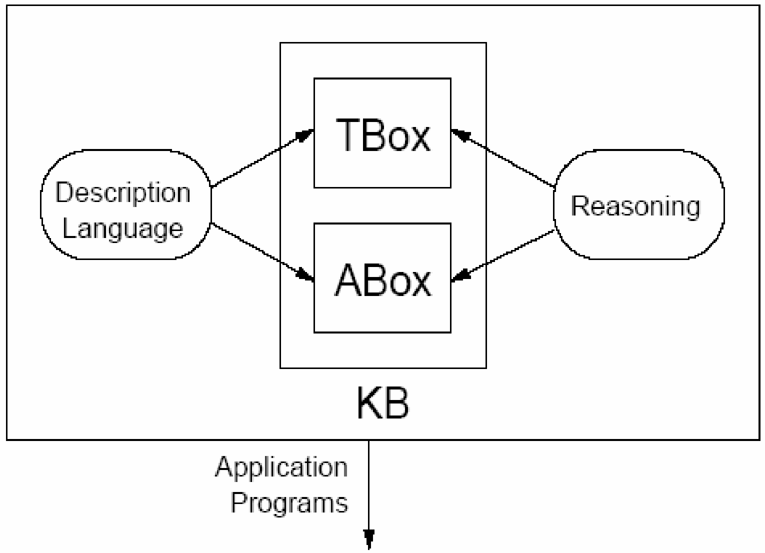
- Consistency (written $\Sigma \not\models$)
  - The problem of checking whether Σ is satisfiable, i.e. it has a model
- Instance checking (written $\Sigma \models C(a)$)
  - The problem of checking whether the assertion $C(a)$ is satisfied in every model of Σ.

- Reduction of instance checking to consistency

$$\Sigma \models C(a) \Leftrightarrow \Sigma \cup \{\neg C(a)\} \models$$

LIMOS NEOS

Reasoning tasks of a DL system



- Terminological
  - Classification

    compute the subsumption hierarchy

- Assertional
  - Realisation

    return the most specific concepts, w.r.t. the subsumption relation, of which a concept a is an instance
  - Retrieval

    return all instances of C.

Reasoning algorithms

- Two types of algorithms are employed to decide inference problems:
  - Structural subsumption algorithms
  - Tableau-based algorithms

only applicable for DLs not allowing for disjunction and full negation, useful for solving non-standard inferences (c.f. Part II)

state of the art technique to decide inferences for a great variety of very expressive DLs

- Illustrate the underlying idea for both approach
  - Running example

$$C_{ex} := \exists r.P \sqcap \forall r.Q \sqcap \forall r.Q',$$
$$D_{ex} := \exists r.(P \sqcap Q) \sqcap \forall r.Q',$$

Structural subsumption algorithms

- Two phases:
  - Turn the given potential subsumee into a normal form (making the implicit knowledge contained in the description explicit),
  - syntactically compare the (potential) subsumer with the normal form of the (potential) subsumee.
- Normalization
  - Uses a set of normalization rules
  - For our example we need the following rules:

$$\forall r.E \sqcap \forall r.F \;\rightarrow\; \forall r.(E \sqcap F),$$
$$\exists r.E \sqcap \forall r.F \;\rightarrow\; \exists r.(E \sqcap F) \sqcap \forall r.F.$$

  - We obtain $C'_{ex} := \exists r.(P \sqcap Q \sqcap Q') \sqcap \forall r.(Q \sqcap Q').$

$$\sqsubseteq \qquad \sqsubseteq$$

$$D_{ex} := \exists r.(P \sqcap Q) \sqcap \forall r.Q',$$

check if for all names and restrictions in the subsumer there exists more specific expressions in the normal form of the subsumee

Normalization rules for ALE

$$\forall r.C \sqcap \forall r.D \rightarrow \forall r.(C \sqcap D) \quad (1)$$

$$\forall r.C \sqcap \exists r.D \rightarrow \forall r.C \sqcap \exists r.(C \sqcap D) \quad (2)$$

$$\forall r.\top \rightarrow \top \quad (3)$$

$$C \sqcap \top \rightarrow C \quad (4)$$

$$P \sqcap \neg P \rightarrow \bot, \text{for all } P \in N_C \quad (5)$$

$$\exists r.\bot \rightarrow \bot \quad (6)$$

$$C \sqcap \bot \rightarrow \bot \quad (7)$$

Tableau algorithms

- Employed for DLs that allow for negation, the subsumption is reduced to deciding satisfiability of concepts: $C \sqsubseteq D \Leftrightarrow C \sqcap \neg D$ is unsatisfiable.

$$C_{ex} \sqcap \neg D_{ex} \;=\; \exists r.P \sqcap \forall r.Q \sqcap \forall r.Q' \sqcap \neg(\exists r.(P \sqcap Q) \sqcap \forall r.Q')$$

$$\equiv\; \exists r.P \sqcap \forall r.Q \sqcap \forall r.Q' \sqcap (\forall r.(\neg P \sqcup \neg Q) \sqcup \exists r.\neg Q') =: E_{ex}$$

Negation normal form

- Build $\mathcal{I}$ with $E_{ex}^{\mathcal{I}} \neq \emptyset$

$a_0 \in E_{ex}^{\mathcal{I}}$

a_1 with $(a_0, a_1) \in r^{\mathcal{I}}$ and $a_1 \in P^{\mathcal{I}}$

$a_1 \in P^{\mathcal{I}} \cap Q^{\mathcal{I}} \cap Q'^{\mathcal{I}}$

$a_0 \in (\forall r.(\neg P \sqcup \neg Q) \sqcup \exists r.\neg Q')^{\mathcal{I}}$

$a_1 \in (\neg P \sqcup \neg Q)^{\mathcal{I}}$ ✸

$a_0 \in (\exists r.\neg Q')^{\mathcal{I}}$ Backtrack

a_2 with $(a_0, a_2) \in r^{\mathcal{I}}$ and $a_2 \in \neg Q'^{\mathcal{I}}$

$a_2 \in (\neg Q')^{\mathcal{I}} \cap Q^{\mathcal{I}} \cap Q'^{\mathcal{I}}$ ✸

E_{ex} is unsatisfiable $\Rightarrow C_{ex} \sqsubseteq D_{ex}$

A tableau algorithm for ALCN

$\mathcal{A}$	rule	$\mathcal{A}'$
$(C_1 \sqcap C_2)(x)$	$\rightarrow_\sqcap$	$C_1(x)$, $C_2(x)$
$(C_1 \sqcup C_2)(x)$	$\rightarrow_\sqcup$	$C(x)$ where $C \in \{C_1, C_2\}$
$(\exists r.C)(x)$	$\rightarrow_\exists$	$C(y), r(x,y)$ where y not occurring in $\mathcal{A}$
$(\forall r.C)(x), r(x,y)$	$\rightarrow_\forall$	$C(y)$
$(\geq r)(x)$	$\rightarrow_\geq$	$\{r(x,y_i) \mid 1 \leq i \leq n\} \cup$ $\{y_i \not\doteq y_j \mid 1 \leq i \leq j \leq n\}$ where $y_1, ..., y_n$ not occurring in $\mathcal{A}$
$(\leq r)(x)$, $r(x,y_1), ..., r(x,y_{n+1})$	$\rightarrow_\leq$	$[y_i/y_j]$(renaming)

A tableau algorithm for ALCN

- Test the satisfiability of an ALCN-concept in negation normal form

$$
\begin{aligned}
\neg\neg C &\rightarrow C \\
\neg(C \sqcap D) &\rightarrow \neg C \sqcup \neg D \\
\neg(\exists r.C) &\rightarrow \forall r.\neg C \\
\neg(\forall r.C) &\rightarrow \exists r.\neg C \\
\neg(\leq n r) &\rightarrow (\geq n+1 r) \\
\neg(\geq 0 r) &\rightarrow \bot \\
\neg(\geq n r) &\rightarrow (\leq n+1 r) \text{ for } n > 0
\end{aligned}
$$

- Start with ABox

$$\mathcal{A}_0 = \{C_0(x_0)\}$$

- Apply propagation rules until
  - no more rule apply

    $\mathcal{A}_0$ is consistent, C_0 satisfiable
  - A contradiction (called clash) occurs

    $\mathcal{A}_0$ is inconsistent, C_0 insatisfiable

Clashes (i) $\{\bot(x)\} \subseteq \mathcal{A}$;

(ii) $\{A(x), \neg A(x)\} \subseteq \mathcal{A}$;

(iii) $\{(\leq n r)(x)\} \cup \{r(x, y_i) \mid 1 \leq i \leq n+1\} \cup \{y_i \neq y_j \mid 1 \leq i < j \leq n+1\} \subseteq \mathcal{A}$.

An example

- Verify the validity of the subsumption:

$$(\geq 3r) \sqcap \exists r.(P \sqcap Q) \sqsubseteq (\geq 2r) \sqcap \exists r.P$$

$$((\geq 3r) \sqcap \exists r.(P \sqcap Q) \sqcap ((\leq 1r) \sqcup \forall r.\neg P))(x)$$

$\rightarrow_\sqcap$	$(\geq 3r)(x)$ $(\exists r.(P \sqcap Q))(x)$ $((\leq 1r) \sqcup \forall r.\neg P)(x)$
$\rightarrow_\exists$	$r(x, y_1)$ $(P \sqcap Q)(y_1)$
$\rightarrow_\sqcap$	$P(y_1)$ $Q(y_1)$
$\rightarrow_\geq$	$r(x, y_2)$ $r(x, y_3)$ $y_1 \neq y_2$ $y_1 \neq y_3$ $y_2 \neq y_3$
$\rightarrow_\sqcup$	$(\leq 1r)(x)$ Clash
$\rightarrow_\sqcup$	$(\forall r.\neg P)(x)$
$\rightarrow_\forall$	$\neg P(y_1)$ $\neg P(y_2)$ $\neg P(y_3)$ Clash

A philosophical question

- The link between structural subsumption and tableau algorithms