Description Logics

Propositional Description Logics

Enrico Franconi

franconi@cs.man.ac.uk
http://www.cs.man.ac.uk/~franconi

Department of Computer Science, University of Manchester

Summary: where we stand

- Description Logics as a formalization of O-O languages
- Description Logics as a *predicate level* language
  - Concepts
  - Roles
- Reasoning in Description Logics
  - Subsumption
- $\mathcal{FL}^-$: the simplest structural description logic

Why Description Logics?

If predicate logic is directly used without some kind of restriction, then

- the structure of the knowledge/information is lost (no variables, concepts as classes, and roles as properties),
- the expressive power is too high for having good computational properties and efficient procedures.

Axioms, Disjunctions and Negations

$$\texttt{Teaching-Assistant} \sqsubseteq \neg\texttt{Undergrad} \sqcup \texttt{Professor}$$

$$\forall x.\ \texttt{Teaching-Assistant}(x) \rightarrow \neg\texttt{Undergrad}(x) \vee \texttt{Professor}(x)$$

A necessary condition in order to be a teaching assistant is to be either not undergraduated or a professor. Clearly, a graduated student being a teaching assistant is not necessarily a professor; moreover, it may be the case that some professor is not graduated.

$$\texttt{Teaching-Assistant} \doteq \neg\texttt{Undergrad} \sqcup \texttt{Professor}$$

$$\forall x.\ \texttt{Teaching-Assistant}(x) \leftrightarrow \neg\texttt{Undergrad}(x) \vee \texttt{Professor}(x)$$

When the left-han side is an atomic concept, the "$\sqsubseteq$" symbol introduces a *primitive definition* – giving only necessary conditions – while the "$\doteq$" symbol introduces a real definition – with necessary and sufficient conditions.

In general, it is possible to have complex concept expressions at the left-hand side as well.

$\mathcal{ALC}$: the simplest propositional DL

A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	primitive concept
R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$	primitive role
$\top$	$\Delta^{\mathcal{I}}$	top
$\bot$	$\emptyset$	bottom
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	complement
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	conjunction
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	disjunction
$\forall R.C$	$\{x \mid \forall y.\, R^{\mathcal{I}}(x,y) \rightarrow C^{\mathcal{I}}(y)\}$	universal quant.
$\exists R.C$	$\{x \mid \exists y.\, R^{\mathcal{I}}(x,y) \ \wedge C^{\mathcal{I}}(y)\}$	existential quant.

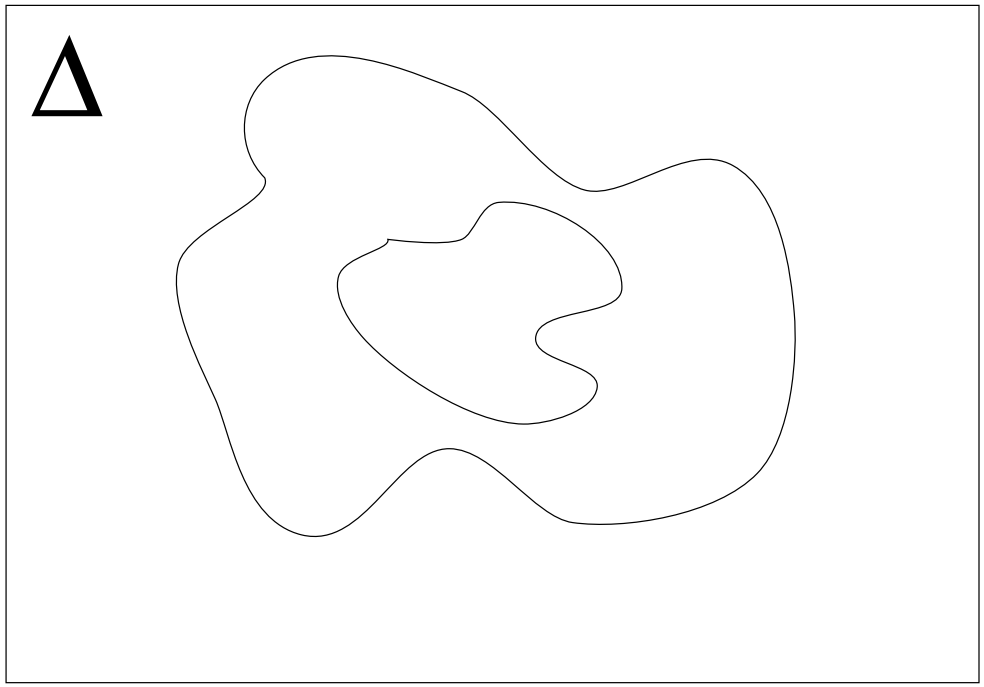
Closed Propositional Language

- **Conjunction** is interpreted as *intersection* of sets of individuals.
- **Disjunction** is interpreted as *union* of sets of individuals.
- **Negation** is interpreted as *complement* of sets of individuals.

- $\exists R.\top \iff \exists R.$
- $\neg(C \sqcap D) \iff \neg C \sqcup \neg D$
- $\neg(C \sqcup D) \iff \neg C \sqcap \neg D$
- $\neg(\forall R.C) \iff \exists R.\neg C$
- $\neg(\exists R.C) \iff \forall R.\neg C$

Negating Universal formulæ

- $\neg(\forall R.C) \Longrightarrow \exists R.\neg C$
- $\neg(\exists R.C) \Longrightarrow \forall R.\neg C$



(Compare with $\mathcal{FL}^-$ expressivity)

Formal Semantics

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of:

- a nonempty set $\Delta^{\mathcal{I}}$ (the *domain*)
- a function $\cdot^{\mathcal{I}}$ (the *interpretation function*) that maps
  - every *concept* to a subset of $\Delta^{\mathcal{I}}$
  - every *role* to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
  - every *individual* to an element of $\Delta^{\mathcal{I}}$

An interpretation function $\cdot^{\mathcal{I}}$ is an **extension** function if and only if it satisfies the semantic definitions of the language.

Knowledge Bases

$$\boxed{\Sigma = \langle \mathsf{TBox}, \mathsf{Abox} \rangle}$$

- **Terminological Axioms:** $C \sqsubseteq D$, $C \doteq D$
  - $\mathtt{Student} \doteq \mathtt{Person} \sqcap \exists \mathtt{NAME}.\mathtt{String} \sqcap \exists \mathtt{ADDRESS}.\mathtt{String} \sqcap \exists \mathtt{ENROLLED}.\mathtt{Course}$
  - $\mathtt{Student} \sqsubseteq \exists \mathtt{ENROLLED}.\mathtt{Course}$
  - $\exists \mathtt{TEACHES}.\mathtt{Course} \sqsubseteq \neg \mathtt{Undergrad} \sqcup \mathtt{Professor}$
- **Membership statements:** $C(a), R(a, b)$
  - $\mathtt{Student}(\mathtt{john})$
  - $\mathtt{ENROLLED}(\mathtt{john}, \mathtt{cs415})$
  - $(\mathtt{Student} \sqcup \mathtt{Professor})(\mathtt{paul})$

TBox: descriptive semantics

Different semantics have been proposed for the TBox, depending on the fact whether cyclic statements are allowed or not.

We consider now the descriptive semantics, based on classical logics.

- An interpretation $\mathcal{I}$ *satisfies* the statement $C \dot{\sqsubseteq} D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
- An interpretation $\mathcal{I}$ *satisfies* the statement $C \doteq D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$.

An interpretation $\mathcal{I}$ is a *model* for a TBox $\mathcal{T}$ if $\mathcal{I}$ satisfies all the statements in $\mathcal{T}$.

ABox

If $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is an interpretation,

- $C(a)$ is satisfied by $\mathcal{I}$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$.
- $R(a, b)$ is satisfied by $\mathcal{I}$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

A set $\mathcal{A}$ of assertions is called an ABox.

An interpretation $\mathcal{I}$ is said to be a *model* of the ABox $\mathcal{A}$ if every assertion of $\mathcal{A}$ is satisfied by $\mathcal{I}$. The ABox $\mathcal{A}$ is said to be *satisfiable* if it admits a model.

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is said to be a *model* of a knowledge base Σ if every axiom of Σ is satisfied by $\mathcal{I}$.
A knowledge base Σ is said to be *satisfiable* if it admits a model.

Logical Implication

$\Sigma \models \varphi$ if every model of Σ is a model of φ

Example:

TBox:

$$\exists \texttt{TEACHES.Course} \sqsubseteq \neg \texttt{Undergrad} \sqcup \texttt{Professor}$$

ABox:

$$\texttt{TEACHES}(\texttt{john}, \texttt{cs415}), \texttt{Course}(\texttt{cs415}), \texttt{Undergrad}(\texttt{john})$$

$\Sigma \models \texttt{Professor}(\texttt{john})$

Logical Implication

What if:

TBox:

$$\exists \texttt{TEACHES.Course} \sqsubseteq \texttt{Undergrad} \sqcup \texttt{Professor}$$

ABox:

$$\texttt{TEACHES}(\texttt{john}, \texttt{cs415}), \texttt{Course}(\texttt{cs415}), \texttt{Undergrad}(\texttt{john})$$

$$\Sigma \stackrel{?}{\models} \texttt{Professor}(\texttt{john})$$

$$\Sigma \stackrel{?}{\models} \neg\texttt{Professor}(\texttt{john})$$

Reasoning Services

- **Concept Satisfiability**

$\Sigma \not\models C \equiv \bot$ $\qquad$ $\texttt{Student} \sqcap \neg\texttt{Person}$

the problem of checking whether C is satisfiable w.r.t. Σ, i.e. whether there exists a model $\mathcal{I}$ of Σ such that $C^{\mathcal{I}} \neq \emptyset$

- **Subsumption**

$\Sigma \models C \sqsubseteq D$ $\qquad$ $\texttt{Student} \sqsubseteq \texttt{Person}$

the problem of checking whether C is subsumed by D w.r.t. Σ, i.e. whether $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in every model $\mathcal{I}$ of Σ

- **Satisfiability**

$\Sigma \not\models$ $\qquad$ $\texttt{Student} \doteq \neg\texttt{Person}$

the problem of checking whether Σ is satisfiable, i.e. whether it has a model

- **Instance Checking**

$\Sigma \models C(a)$ $\qquad$ $\texttt{Professor}(\texttt{john})$

the problem of checking whether the assertion $C(a)$ is satisfied in every model of Σ

Reasoning Services *(cont.)*

- **Retrieval**

  $\{a \mid \Sigma \models C(a)\}$ $\quad$ $\texttt{Professor} \Rightarrow \texttt{john}$

- **Realization**

  $\{C \mid \Sigma \models C(a)\}$ $\quad$ $\texttt{john} \Rightarrow \texttt{Professor}$

Reduction to satisfiability

- Concept Satisfiability

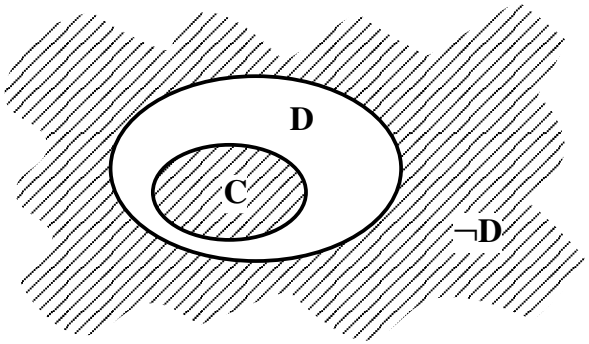
  $\Sigma \not\models C \equiv \bot \quad \leftrightarrow$

  exists x s.t. $\Sigma \cup \{C(x)\}$ has a model

- Subsumption

  $\Sigma \models C \sqsubseteq D \quad \leftrightarrow$

  $\Sigma \cup \{(C \sqcap \neg D)(x)\}$ has no models

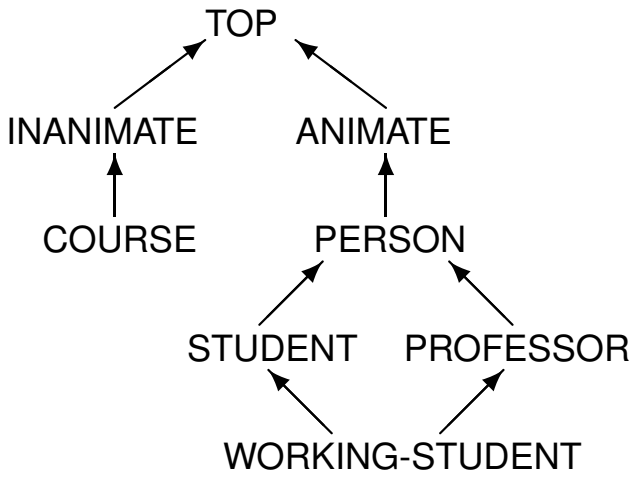


- Instance Checking

  $\Sigma \models C(a) \quad \leftrightarrow$

  $\Sigma \cup \{\neg C(a)\}$ has no models

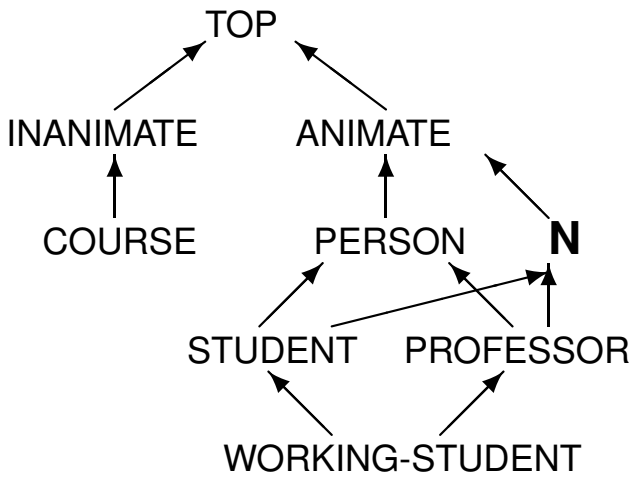
The Taxonomy



- Subsumption is a *partial ordering* relation in the space of concepts.
- If we consider only *named* concepts, subsumption induces a taxonomy where only direct subsumptions are explicitly drawn.
- A taxonomy is the minimal relation in the space of named concepts such that its rlflexive-transitive closure is the subsumption relation.

The Taxonomy

$$\mathtt{N} \doteq \mathtt{ANIMATE} \sqcap (\mathtt{STUDENT} \sqcup \mathtt{PROFESSOR})$$



- Subsumption is a *partial ordering* relation in the space of concepts.
- If we consider only *named* concepts, subsumption induces a taxonomy where only direct subsumptions are explicitly drawn.
- A taxonomy is the minimal relation in the space of named concepts such that its rlflexive-transitive closure is the subsumption relation.

Classification

- Given a concept C and a TBox $\mathcal{T}$, for all concepts D of $\mathcal{T}$ determine whether D subsumes C, or D is subsumed by C.
- Intuitively, this amounts to finding the "right place" for C in the taxonomy implicitly present in $\mathcal{T}$.
- *Classification* is the task of inserting new concepts in a taxonomy. It is *sorting* in partial orders.

Reasoning procedures

- Terminating, efficient and complete algorithms for deciding **satisfiability** – and all the other reasoning services – are available.
- Algorithms are based on tableaux-calculi techniques.
- Completeness is important for the usability of description logics in real applications.
- Such algorithms are efficient for both average and real knowledge bases, even if the problem in the corresponding logic is in PSPACE or EXPTIME.

Tableaux Calculus

The Tableaux Calculus is a decision procedure solving the problem of satisfiability.

If a formula is satisfiable, the procedure will constructively exhibit a model of the formula.

The basic idea is to incrementally build the model by looking at the formula, by decomposing it in a top/down fashion. The procedure exhaustively looks at all the possibilities, so that it can eventually prove that no model could be found for unsatisfiable formulas.

Tableaux Calculus

1. Syntactically transform a theory Σ in a *Constraint System* S – also called *tableaux*.
   Every formula of Σ is transformed into a *constraint* in S.

2. Add constraints to S, applying specific *completion rules*.
   Completion rules are either deterministic – they yield a uniquely determined constraint system – or nondeterministic – yielding several possible alternative constraint systems (*branches*).

3. Apply the completion rules until either a contradiction (a *clash*) is generated in every branch, or there is a *completed* branch where no more rule is applicable.

4. The completed constraint system gives a model of Σ; it corresponds to a particular branch of the tableaux.

The FOL example

$$\frac{\phi \wedge \psi}{\begin{array}{c}\phi\\ \psi\end{array}} \qquad \frac{\phi \vee \psi}{\phi \mid \psi} \qquad \frac{\forall x.\ \phi}{\phi\{X/t\}} \qquad \frac{\exists x.\ \phi}{\phi\{X/Z\}}$$

$$\boxed{\exists y.\ (p(y) \wedge \neg q(y)) \wedge \forall z.\ (p(z) \vee q(z))}$$

$$\exists y.\ (p(y) \wedge \neg q(y))$$

$$\forall z.\ (p(z) \vee q(z))$$

$$p(\bar{y}) \wedge \neg q(\bar{y})$$

$$p(\bar{y})$$

$$\neg q(\bar{y})$$

$$p(\bar{y}) \vee q(\bar{y})$$

$$\begin{array}{cc} p(\bar{y}) & q(\bar{y}) \\ < \texttt{COMPLETED} > & < \texttt{CLASH} > \end{array}$$

The formula is satisfiable. The devised model is $\Delta^{\mathcal{I}} = \{\bar{y}\},\ p^{\mathcal{I}} = \{\bar{y}\},\ q^{\mathcal{I}} = \emptyset$.

Negation Normal Form

Recall that the above completion rules for FOL work only if the formula has been translated into Negation Normal Form, i.e., all the negations have been pushed down.

In the same way, we can transform any $\mathcal{ALC}$ formula into an equivalent one in Negation Normal Form, so that negation appears only in front of atomic concepts:

- $\neg(C \sqcap D) \Longleftrightarrow \neg C \sqcup \neg D$
- $\neg(C \sqcup D) \Longleftrightarrow \neg C \sqcap \neg D$
- $\neg(\forall R.C) \Longleftrightarrow \exists R.\neg C$
- $\neg(\exists R.C) \Longleftrightarrow \forall R.\neg C$

Completion Rules: the AND rule

The propagation rules come straightforwardly from the semantics of constructors.

If in a given interpretation $\mathcal{I}$, whose domain contains the element a, we have that $a \in (C \sqcap D)^{\mathcal{I}}$, then from the semantics we know that such element a should be in the intersection of $C^{\mathcal{I}}$ and $D^{\mathcal{I}}$, i.e. it should be in both $C^{\mathcal{I}}$ and $D^{\mathcal{I}}$.

Since this must be true for any interpretation, we can abstract from interpretations and their elements, and say that if in a generic interpretation we have a generic element x that is in the interpretation of the concept $C \sqcap D$ (denote this by $x : (C \sqcap D)$) then the element x should belong both to the interpretation of C and to the interpretation of D.

The AND rule

Suppose now we want to construct a generic interpretation S such that the set corresponding to the concept $C \sqcap D$ contains at least one element. We can state this initial requirement as the constraint $x : (C \sqcap D)$.

Following the semantics, we know that S must be such that the constraints $x : C$ and $x : D$ must hold, hence we can add these new constraints to S, knowing that if S will ever satisfy them then it will also satisfy the first constraint.

These considerations lead to the following propagation rule:

$$S \rightarrow_{\sqcap} \{x : C,\ x : D\} \cup S$$

if 1. $x : C \sqcap D$ is in S,

2. $x : C$ and $x : D$ are not both in S

The SOME rule

If in a given interpretation $\mathcal{I}$, whose domain contains the element a, we have that $a \in (\exists R.C)^{\mathcal{I}}$, then from the semantics we know that there must be an element b (not necessarily distinct from a) such that $(a, b) \in R^{\mathcal{I}}$, and $b \in C^{\mathcal{I}}$.

Since this must be true for any interpretation, we can abstract from interpretations and their elements, and say that if in a generic interpretation we have a generic element x that is in the interpretation of the concept $\exists R.C$ (denote this by $x : \exists R.C$) then there must be a generic element y such that x and y are in relation through R (denote it xRy) and y belongs to the interpretation of C (denoted as $y : C$).

The SOME rule *(cont.)*

These considerations lead to the following propagation rule:

$$S \;\rightarrow_{\exists}\; \{xRy,\; y\colon C\} \cup S$$

if 1. $x\colon \exists R.C$ is in S,

2. y is a new variable,

3. there is no z such that

   both xRz and $z\colon C$ are in S

Completion rules for $\mathcal{ALC}$

$S \rightarrow_{\sqcap} \{x : C,\ x : D\} \cup S$

if 1. $x : C \sqcap D$ is in S,
2. $x : C, x : D$ are not both in S

$S \rightarrow_{\sqcup} \{x : E\} \cup S$

if 1. $x : C \sqcup D$ is in S,
2. neither $x : C$ nor $x : D$ is in S,
3. $E = C$ or $E = D$

$S \rightarrow_{\forall} \{y : C\} \cup S$

if 1. $x : \forall R.C$ is in S,
2. xRy is in S,
3. $y : C$ is not in S

$S \rightarrow_{\exists} \{xRy,\ y : C\} \cup S$

if 1. $x : \exists R.C$ is in S,
2. y is a new variable,
3. there is no z such that
both xRz and $z : C$ are in S

Clash

While building a constraint system, we can look for evident contradictions to see if the constraint system is not satisfiable. We call these contradictions **clashes**.

A *clash* is a constraint system having the form:

$\{x : A,\ x : \neg A\}$, where A is a concept name.

A clash is evidently an unsatisfiable constraint system, hence any constraint system containing a clash is obviously unsatisfiable.

An Example of tableaux

Satisfiability of the concept:

$$\boxed{\big((\forall \texttt{CHILD}.\texttt{Male}) \sqcap (\exists \texttt{CHILD}.\neg\texttt{Male})\big)}$$

$$\big((\forall \texttt{CHILD}.\texttt{Male}) \sqcap (\exists \texttt{CHILD}.\neg\texttt{Male})\big)(\texttt{x}) \qquad$$

$$(\forall \texttt{CHILD}.\texttt{Male})(\texttt{x}) \qquad \sqcap\textit{-rule}$$

$$(\exists \texttt{CHILD}.\neg\texttt{Male})(\texttt{x}) \qquad \text{“}$$

$$\texttt{CHILD}(x, y) \qquad \exists\textit{-rule}$$

$$\neg\texttt{Male}(\boldsymbol{y}) \qquad \text{“}$$

$$\texttt{Male}(\boldsymbol{y}) \qquad \forall\textit{-rule}$$

$$\langle CLASH \rangle$$

An Example of tableaux - *constraint* syntax -

$$\boxed{\big((\forall \texttt{CHILD}.\texttt{Male}) \sqcap (\exists \texttt{CHILD}.\neg\texttt{Male})\big)}$$

$$x : \big((\forall \texttt{CHILD}.\texttt{Male}) \sqcap (\exists \texttt{CHILD}.\neg\texttt{Male})\big)$$

$$x : (\forall \texttt{CHILD}.\texttt{Male}) \qquad \sqcap\textit{-rule}$$

$$x : (\exists \texttt{CHILD}.\neg\texttt{Male}) \qquad \text{“}$$

$$x\ \texttt{CHILD}\ y \qquad \exists\textit{-rule}$$

$$y : \neg\texttt{Male} \qquad \text{“}$$

$$y : \texttt{Male} \qquad \forall\textit{-rule}$$

$$\langle CLASH \rangle$$

Another example

$$\boxed{((\forall \texttt{CHILD.Male}) \sqcap (\exists \texttt{CHILD.Male}))}$$

$x : ((\forall \texttt{CHILD.Male}) \sqcap (\exists \texttt{CHILD.Male}))$

$x : (\forall \texttt{CHILD.Male})$ $\sqcap$-*rule*

$x : (\exists \texttt{CHILD.Male})$ "

$x \ \texttt{CHILD} \ y$ $\exists$-*rule*

$y : \texttt{Male}$ "

$y : \texttt{Male}$ $\forall$-*rule*

$\langle COMPLETED \rangle$

Exercise: find a model.

Tableaux with individuals

Check the satisfiability of the ABox:

$(\texttt{Parent} \sqcap \forall \texttt{CHILD}.\texttt{Male})(\texttt{john})$

$\neg \texttt{Male}(\texttt{mary})$

$\texttt{CHILD}(\texttt{john}, \texttt{mary})$

$\texttt{john} : \texttt{Parent} \sqcap \forall \texttt{CHILD}.\texttt{Male}$

$\texttt{mary} : \neg \texttt{Male}$

$\texttt{john}\ \texttt{CHILD}\ \texttt{mary}$

$\texttt{john} : \texttt{Parent}$ $\sqcap$-*rule*

$\texttt{john} : \forall \texttt{CHILD}.\texttt{Male}$ "

$\texttt{mary} : \texttt{Male}$ $\forall$-*rule*

$\langle CLASH \rangle$

The knowledge base is inconsistent.

Soundness of the Tableaux for $\mathcal{ALC}$

The calculus does not add unnecessary contradictions.

That is, deterministic rules always preserve the Satisfiability of a constraint system, and nondeterministic rules have always a choice of application that preserves Satisfiability.

Termination of the Tableaux for $\mathcal{ALC}$

A constraint system is *complete* if no propagation rule applies to it. A complete system derived from a system S is also called a *completion* of S. Completions are reached when there is no infinite chain of applications of rules.

Intuitively, this can be proved by using the following argument: all rules but $\longrightarrow_\forall$ are never applied twice on the same constraint; this rule in turn is never applied to a variable x more times than the number of the *direct successors* of x, which is bounded by the length of a concept; finally, each rule application to a constraint $y : C$ adds constraints $z : D$ such that D is a strict subexpression of C.

Completeness of the Tableaux for $\mathcal{ALC}$

If S is a completion of $\{x : C\}$ and S contains no clash, then it is always possible to construct an interpretation for C on the basis of S, such that $C^{\mathcal{I}}$ is nonempty.

The proof is a straightforward induction on the length of the concepts involved in each constraint.

Interpretations as graphs

An interpetation can be viewed as a labeled directed graph.

- Each node is a generic element of the interpretation domain.
- Labels on nodes are concepts which include that specific element in the interpretation.
- Each arc is labeled by a relationship (i.e., a role) among elements of the interpretation domain that must hold.

Exponential models

$\exists R.C_1 \sqcap \exists R.C_2 \sqcap \forall R.$ $\boxed{(\exists R.C_1 \sqcap \exists R.C_2 \sqcap \forall R.\boxed{\ldots})}$

$x : \exists R.C_1 \sqcap \exists R.C_2 \sqcap \forall R.(\exists R.C_1 \sqcap \exists R.C_2 \sqcap \forall R.(\ldots))$

$\boxed{\begin{array}{l} xRx_1,\ x_1 : C_1 \\ \quad x_1 : \exists R.C_1 \sqcap \exists R.C_2 \sqcap \forall R.(\ldots) \\ \quad\quad \boxed{\begin{array}{l}\ldots\\ \ldots\end{array}} \\ xRx_2,\ x_2 : C_2 \\ \quad x_2 : \exists R.C_1 \sqcap \exists R.C_2 \sqcap \forall R.(\ldots) \\ \quad\quad \boxed{\begin{array}{l}\ldots\\ \ldots\end{array}} \end{array}}$

2^n generated variables!

Exercise: depict the model as a graph.

Complexity of reasoning

Expressivity		$\models C \sqsubseteq D$	$\models C(a)$
$C \sqcap D$ $\forall R.C$ $\exists R$	$\mathcal{FL}^-$	P	P
$\neg A$	$\mathcal{AL}$	P	P
$\exists R.C$	$\mathcal{ALE}$	NP	PSPACE
$\neg C$	$\mathcal{ALC}$	PSPACE	$\Longleftarrow$!!
$\{a_1 \ldots\}$	$\mathcal{ALCO}$	PSPACE	
	$\mathcal{SHIQ}$	EXPTIME	
	KL-ONE	undecidable	

Traces

- In order to obtain a polynomial space algorithm for $\mathcal{ALC}$, we should exploit the property of independency between *traces* of a satisfiability proof.
- A *completed constraint system* can be partitioned into traces, where the computation can be performed *independently* – i.e. an inconsistency can be generated only by a clash belonging to a single trace.
- Since a completed constraints system denotes a model, it can be regarded as a graph: traces correspond to paths from the starting node to a leaf.
- A clash at a leaf node can only be generated by the application of rules from the trace it belongs to. It is impossible that a clash is generated by rules applied at some other trace. This is because completed constraint systems are trees.
- A trace has polynomial size!

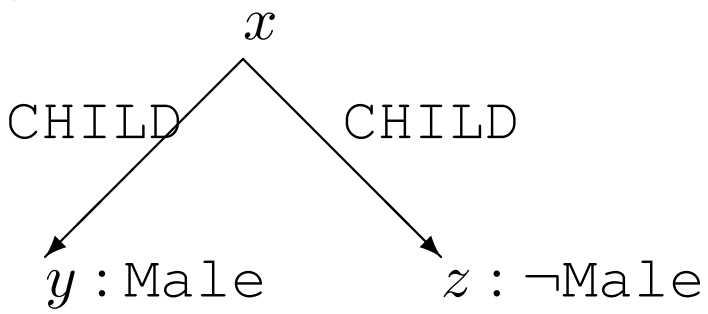
Functional Algorithms

- Nodes in a constraint system are only generated by the completion rule for the *existential constraint* "$\longrightarrow_\exists$".
- In order to exploit traces (which are paths in the model), we force a depth-first strategy in the generation of new nodes in the constraint system.
  - Apply the "$\longrightarrow_\exists$" rule only if no other rule is applicable;
  - If the "$\longrightarrow_\exists$" rule is applicable to more that one constraint, choose the constraint with the most recently generated variable.

Example

$$\boxed{((\exists \texttt{CHILD}.\texttt{Male}) \sqcap (\exists \texttt{CHILD}.\neg\texttt{Male}))}$$

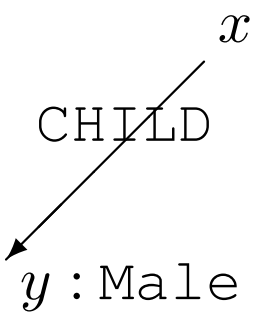
$x : ((\exists \texttt{CHILD}.\texttt{Male}) \sqcap (\exists \texttt{CHILD}.\neg\texttt{Male}))$	
$x : (\exists \texttt{CHILD}.\texttt{Male})$	$\sqcap$-*rule*
$x : (\exists \texttt{CHILD}.\neg\texttt{Male})$	"
$x\ \texttt{CHILD}\ y$	$\exists$-*rule*
$y : \texttt{Male}$	"
$x\ \texttt{CHILD}\ z$	$\exists$-*rule*
$z : \neg\texttt{Male}$	"
$\langle COMPLETED \rangle$	



Example with traces

$x : \big((\exists \texttt{CHILD}.\texttt{Male}) \sqcap (\exists \texttt{CHILD}.\neg\texttt{Male})\big)$

$x : (\exists \texttt{CHILD}.\texttt{Male})$	$\sqcap$-*rule*
$x : (\exists \texttt{CHILD}.\neg\texttt{Male})$	"
$x\ \texttt{CHILD}\ y$	$\exists$-*rule*
$y : \texttt{Male}$	"

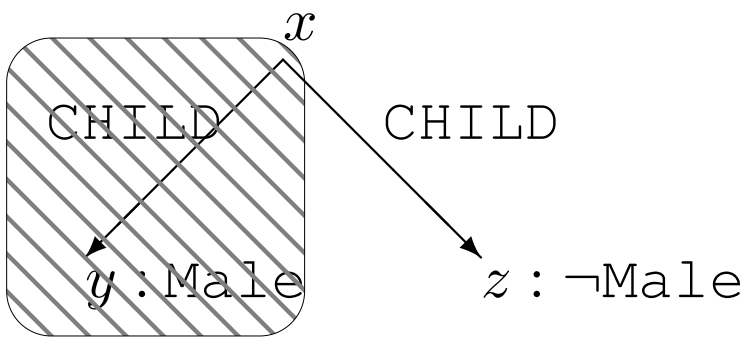


Example with traces

$x : \big((\exists \texttt{CHILD}.\texttt{Male}) \sqcap (\exists \texttt{CHILD}.\neg\texttt{Male})\big)$

$x : (\exists \texttt{CHILD}.\texttt{Male})$	$\sqcap$-*rule*
$x : (\exists \texttt{CHILD}.\neg\texttt{Male})$	"
$x\ \texttt{CHILD}\ y$	$\exists$-*rule*
$y : \texttt{Male}$	"
$x\ \texttt{CHILD}\ z$	$\exists$-*rule*
$z : \neg\texttt{Male}$	"

$\langle COMPLETED \rangle$



The Functional Algorithms for $\mathcal{ALC}$

sat(S) = `if` S includes a clash

`then false`

`elseif` $C \sqcap D \in S$ `and` $C \notin S$ `or` $D \notin S$

`then` $sat(S \cup \{C, D\})$

`elseif` $C \sqcup D \in S$ `and` $C \notin S$ `and` $D \notin S$

`then` $sat(S \cup \{C\})$ `or` $sat(S \cup \{D\})$

`else forall` $\exists R.C \in S$

$sat(\{C\} \cup \{D \mid \forall R.D \in S\})$

Sources of Complexity

Such a deterministic version of the tableaux calculus can be seen as a depth-first exploring of an AND-OR tree:

- AND-branching corresponds to the (independent) check of all *successors* of a node;
- OR-branching corresponds to the choices of application of the non-deterministic rule.

The exponential-time behaviour of the calculus has two origins:

- AND-branching – leading to constraint systems of exponential size (with an exponential number of possible clashes to be searched through);
- OR-branching – leading to an exponential number of possible constraint systems (like in propositional calculus).

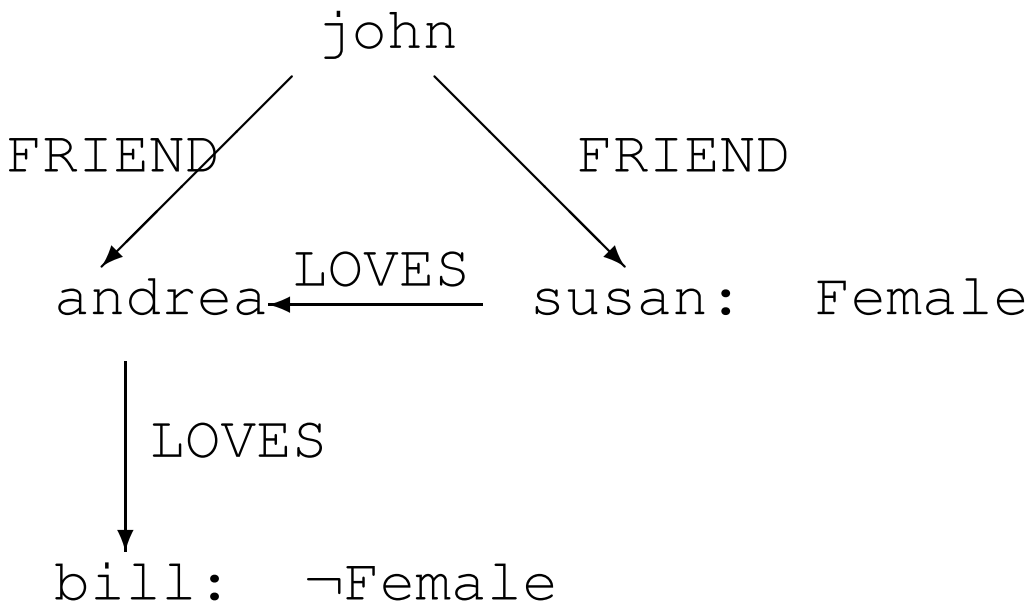
Sources of Complexity - II

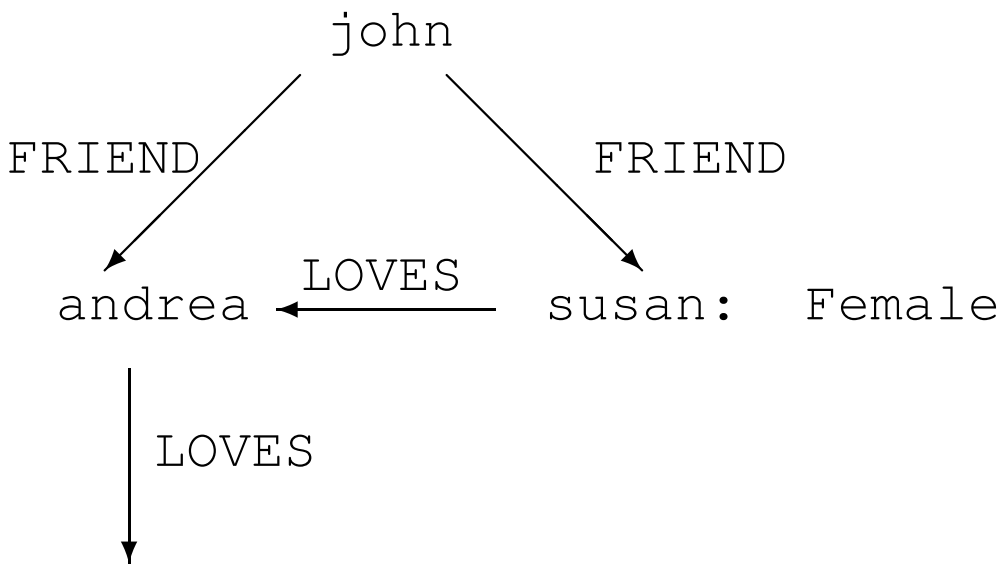
Differently from *databases* and, in general, from *static data structures*, description logics do not handle only ground and complete knowledge but perform also reasoning on incomplete knowledge and case analysis:

- Existential quantification ($\mathcal{ALE}$)
- Disjunction ($\mathcal{ALC}$)
- Enumerated types ($\mathcal{ALCO}$)
- Terminological axioms ($\mathcal{SHIQ}$)

An example

$$\Sigma = \text{FRIEND(john,susan)} \wedge$$
$$\text{FRIEND(john,andrea)} \wedge$$
$$\text{LOVES(susan,andrea)} \wedge$$
$$\text{LOVES(andrea,bill)} \wedge$$
$$\text{Female(susan)} \wedge$$
$$\neg\text{Female(bill)}$$





$\Sigma =$

Does John have a female friend loving a male (i.e. not female) person?

$$\exists X, Y.\ \texttt{FRIEND}(\texttt{john}, X) \wedge \texttt{Female}(X) \wedge \texttt{LOVES}(X, Y) \wedge \neg\texttt{Female}(Y)$$

$$\Sigma \overset{?}{\models} \big(\exists \texttt{FRIEND}.(\texttt{Female} \sqcap (\exists \texttt{LOVES}.\neg\texttt{Female}))\big)(\texttt{john})$$

$\Longrightarrow$ **Answer: YES**

Exercise

- Reduce the problem into a satisfiability problem
- Solve it using plain tableaux calculus
- Solve it using the functional algorithm (is there any difference?)
- Comment on the sources of complexity in finding the solution

Some extensions of $\mathcal{ALC}$

Constructor	Syntax	Semantics
concept name	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
top	$\top$	$\Delta^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction ($\mathcal{U}$)	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
negation ($\mathcal{C}$)	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
universal	$\forall R.C$	$\{x \mid \forall y : R^{\mathcal{I}}(x,y) \rightarrow C^{\mathcal{I}}(y)\}$
existential ($\mathcal{E}$)	$\exists R.C$	$\{x \mid \exists y : R^{\mathcal{I}}(x,y) \wedge C^{\mathcal{I}}(y)\}$
cardinality ($\mathcal{N}$)	$\geqslant n\ R$	$\{x \mid \sharp\{y \mid R^{\mathcal{I}}(x,y)\} \geq n\}$
	$\leqslant n\ R$	$\{x \mid \sharp\{y \mid R^{\mathcal{I}}(x,y)\} \leq n\}$
qual. cardinality ($\mathcal{Q}$)	$\geqslant nR.C$	$\{x \mid \sharp\{y \mid R^{\mathcal{I}}(x,y) \wedge C^{\mathcal{I}}(y)\} \geq n\}$
	$\leqslant nR.C$	$\{x \mid \sharp\{y \mid R^{\mathcal{I}}(x,y) \wedge C^{\mathcal{I}}(y)\} \leq n\}$
enumeration ($\mathcal{O}$)	$\{a_1 \ldots a_n\}$	$\{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$
selection ($\mathcal{F}$)	$f : C$	$\{x \in \mathsf{Dom}(f^{\mathcal{I}}) \mid C^{\mathcal{I}}(f^{\mathcal{I}}(x))\}$

($\mathcal{ALC}$ has same expressivity as $\mathcal{ALCUE}$)

Cardinality Restriction

Role quantification *cannot* express that a woman has *at least 3* (or *at most 5*) children.

Cardinality restrictions can express conditions on the number of fillers:

- $\texttt{Busy}-\texttt{Woman} \doteq \texttt{Woman} \sqcap (\geqslant 3\ \texttt{CHILD})$
- $\texttt{Conscious}-\texttt{Woman} \doteq \texttt{Woman} \sqcap (\leqslant 5\ \texttt{CHILD})$

$$\boxed{(\geqslant 1\ R) \Longleftrightarrow (\exists R.)}$$

Cardinality Restriction

- $\texttt{Busy-Woman} \doteq \texttt{Woman} \sqcap (\geqslant 3\ \texttt{CHILD})$
- $\texttt{Conscious-Woman} \doteq \texttt{Woman} \sqcap (\leqslant 5\ \texttt{CHILD})$

$\texttt{Busy-Woman}(mary)$

busy-woman : Woman,   CHILD : $_{\geq 3}$ Person

mary : Woman,   CHILD : $john$,   CHILD : sue,   CHILD : $karl$

$\models \texttt{Conscious-Woman}(mary)$ **?**

Roles as Functions

- A role is *functional* is the filler functionally depends on the individual, i.e., the role can be considered as a function: $R(x, y) \Leftrightarrow f(x) = y$.
- For example, the roles `CHILD` and `PARENT` are not functional, while the roles `MOTHER` and `AGE` are functional.
- If a role is functional, we write:
  $\exists f.C \equiv f : c$ *(selection operator)*

Individuals

In every interpretation different individuals are assumed to denote different elements, i.e. for every pair of individuals a, b, and for every interpretation $\mathcal{I}$, if $a \neq b$ then $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.

This is called the *Unique Name Assumption* and is usually assumed in database applications.

Example:

How many children does this family have?

`Family(f), Father(f,john), Mother(f,sue),`
`Son(f,paul), Son(f,george), Son(f,alex)`

$\models (\geqslant 3\, \mathtt{Son})(\mathtt{f})$

Enumeration Type (one-of)

- $\texttt{Weekday} \doteq \{\texttt{mon}, \texttt{tue}, \texttt{wed}, \texttt{thu}, \texttt{fri}, \texttt{sat}, \texttt{sun}\}$
- $\texttt{Weekday}^{\mathcal{I}} = \{\texttt{mon}^{\mathcal{I}}, \texttt{tue}^{\mathcal{I}}, \texttt{wed}^{\mathcal{I}}, \texttt{thu}^{\mathcal{I}}, \texttt{fri}^{\mathcal{I}}, \texttt{sat}^{\mathcal{I}}, \texttt{sun}^{\mathcal{I}}\}$
- $\texttt{Citizen} \doteq (\texttt{Person} \sqcap \forall \texttt{LIVES}.\texttt{Country})$
- $\texttt{French} \doteq (\texttt{Citizen} \sqcap \forall \texttt{LIVES}.\{\texttt{france}\})$

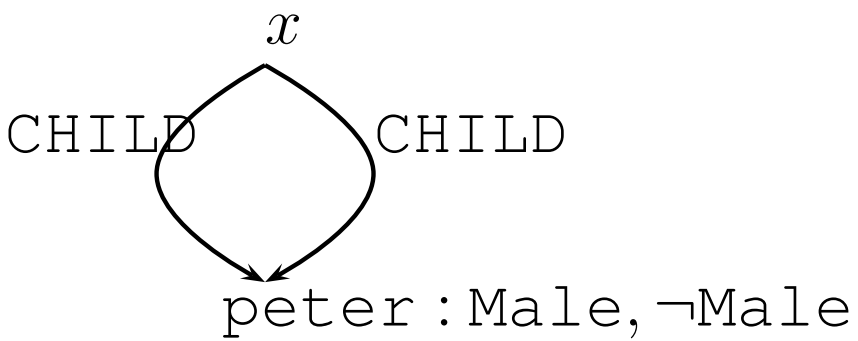
Trace-based satisfiability algorithm for one-of

Expressive languages may not have the trace-independence property: enumerated types introduce interactions between traces, even if the satisfiability problem is still in PSPACE.

Example:

$$\exists \texttt{CHILD}.(\texttt{Male} \sqcap \{\texttt{peter}\}) \sqcap$$
$$\exists \texttt{CHILD}.(\neg\texttt{Male} \sqcap \{\texttt{peter}\})$$

The two traces generated by the two existential quantifications on CHILD are independently satisfiable, but are globally unsatisfiable, since both existential variables should be co-referenced to the individual peter.



Adequacy

Student

Person	
name:	[String]
address:	[String]
enrolled:	[Course]

$$\begin{aligned}\texttt{Student} \doteq\ & \texttt{Person} \sqcap \\ & \texttt{NAME} : \texttt{String} \sqcap \\ & \forall \texttt{ADDRESS}.\texttt{String} \sqcap \\ & \geqslant 1\ \texttt{ADDRESS} \sqcap \\ & \exists \texttt{ENROLLED}.\texttt{Course}\end{aligned}$$

Some constructors for role expressions

Constructor	Syntax	Semantics
role name	P	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
conjunction	$R \sqcap S$	$R^{\mathcal{I}} \cap S^{\mathcal{I}}$
disjunction	$R \sqcup S$	$R^{\mathcal{I}} \cup S^{\mathcal{I}}$
negation	$\neg R$	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus R^{\mathcal{I}}$
inverse	R^{-}	$\{(x, y) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (y, x) \in R^{\mathcal{I}}\}$
composition	$R \circ S$	$\{(x, y) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \exists z.\ (x, z) \in R^{\mathcal{I}} \wedge (z, y) \in S^{\mathcal{I}}\}$
range	$R\|_C$	$\{(x, y) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
product	$C \times D$	$\{(x, y) \in C^{\mathcal{I}} \times D^{\mathcal{I}}\}$

Extending Description Logics

- Defaults and Beliefs
- Probability- and similarity-based reasoning
- Epistemic statements
- Closed world assumption
- Plural entities: records, sets, collections, aggregations
- Concrete domains
- Ontological primitives
  - time and action
  - space
  - parts and wholes