Propositional Description Logics

(Acknowledgement: Enrico Franconi)

Extending the set of concept forming operators with
Disjunction and **Negation**

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

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

The "$\sqsubseteq$" symbol introduces a *partial definition*: **necessary** conditions.

The "$\doteq$" symbol introduces a *complete definition* with **necessary** and **sufficient** conditions:

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

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

Syntax and Semantics of $\mathcal{ALC}$ — The Simplest Propositional DL

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

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.

Equivalences

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

Knowledge Bases

$\boxed{\Sigma = \langle \text{TBox, ABox} \rangle}$

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

Terminology

A terminology is a set $\mathcal{T}$ of terminological axioms s.t.

- each concept symbol occurs at most once on the lhs of a terminological axiom
- the definitions are not recursive (no cycles).

TBox: Descriptive Semantics

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

- An interpretation $\mathcal{I}$ *satisfies* the statement $C \sqsubseteq D$ 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* statements in $\mathcal{T}$.

ABox

If $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ 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 $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ 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:

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

ABox:

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

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

Reasoning Services

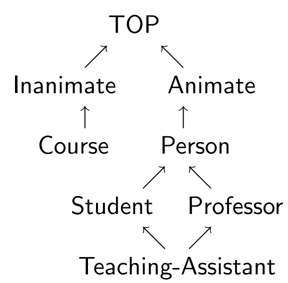
- **Concept Satisfiability**
  $\Sigma \not\models C \equiv \bot$ $\quad$ $\texttt{Student} \sqcap \neg\texttt{Person}$
  The problem of checking whether C is satisfiable w.r.t. Σ, i.e. whether there exists a model $\mathcal{I}$ of Σ such that $C^{\mathcal{I}} \neq \emptyset$
- **Subsumption**
  $\Sigma \models C \sqsubseteq D$ $\quad$ $\texttt{Student} \sqsubseteq \texttt{Person}$
  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**
  $\Sigma \not\models$ $\quad$ $\texttt{Student} \doteq \neg\texttt{Person}$
  The problem of checking whether Σ is satisfiable, i.e. whether it has a model

- **Instance Checking**
  $\Sigma \models C(a)$ $\quad$ $\texttt{Professor}(\texttt{john})$
  The problem of checking whether the assertion $C(a)$ is satisfied in every model of Σ
- **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
  $\Sigma \not\models C \equiv \bot \quad \longleftrightarrow \quad$ exists x s.t. $\Sigma \cup \{C(x)\}$ has a model
- Subsumption
  $\Sigma \models C \sqsubseteq D \quad \longleftrightarrow \quad \Sigma \cup \{(C \sqcap \neg D)(x)\}$ has no models
- Instance Checking
  $\Sigma \models C(a) \quad \longleftrightarrow \quad \Sigma \cup \{\neg C(a)\}$ has no models

Example Taxonomy



- The subsumption relation is a partial ordering relation in the space of concepts.
- If we consider only *named* concepts, subsumption induces a taxonomy — i.e. a generalization/specialization hierarchy — where only direct subsumptions are explicitly drawn.
- A taxonomy is the minimal relation in the space of named concepts such that its reflexive-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.

Example

$$\texttt{Woman} \sqsubseteq \texttt{Person}$$

$$\texttt{Man} \doteq \texttt{Person} \sqcap \neg\texttt{Woman}$$

$$\texttt{Parent} \doteq \texttt{Person} \sqcap \exists\texttt{CHILD}.\top \sqcap \forall\texttt{CHILD.Person}$$

$$\texttt{Mother} \doteq \texttt{Parent} \sqcap \texttt{Woman}$$

$$\texttt{Father} \doteq \texttt{Parent} \sqcap \neg\texttt{Mother}$$

$$\texttt{Grandparent} \doteq \texttt{Person} \sqcap \exists\texttt{CHILD.Parent} \sqcap \forall\texttt{CHILD.Person}$$

$$\texttt{Grandmother} \doteq \texttt{Grandparent} \sqcap \texttt{Woman}$$

$$\begin{array}{ccccc}
 & & \text{Person} & & \\
 & \nearrow & \uparrow & \nwarrow & \\
\text{Man} & & \text{Parent} & & \text{Woman} \\
 & \nearrow & \uparrow & \nwarrow & \\
\text{Father} & & \text{Grandparent} & & \text{Mother} \\
 & & \uparrow & \nwarrow & \\
 & & \bot & & \text{Grandmother}
\end{array}$$

Reasoning Procedures

- Terminating, efficient and complete algorithms for deciding *satisfiability* — and all the other reasoning services — are available.
- Algorithms are based on tableau calculus 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.

Checking Subsumption

"Standard method" for checking subsumption between two concepts C and D in a terminology (TBox) $\mathcal{T}$, $C \preceq_{\mathcal{T}} D$:

Instead of testing $C \preceq_{\mathcal{T}} D$, check whether $C \sqcap \neg D$ is inconsistent

- Replace all definitions $A \sqsubseteq X$ by $A \doteq A^* \sqcap X$, where A^* is a "primitive component" (a new syntactic category) which is completely undefined in $\mathcal{T}$, representing the primitive part of the concept's meaning.
- Replace all defined symbols in $C \sqcap \neg D$ by their definitions $\Longrightarrow E$.
- Transform E into negation normal form, i.e. only atomic concepts are negated $\Longrightarrow F$.
- Check with the tableau method whether F is inconsistent.

Tableau Calculus

1. Transform a theory Σ syntactically into a *constraint system* S — i.e. a *tableau*. 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 — yielding 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 rules are applicable.
4. The completed constraint system gives a model of Σ; it corresponds to a particular branch of the tableau.

Negation Normal Form

The completions rules require (as in FOL) that the formulae have been translated into Negation Normal Form, i.e., all the negations have been pushed down.

We can transform any $\mathcal{ALC}$ formula into an equivalent one in Negation Normal Form, such 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 for $\mathcal{ALC}$

1. $S \to_{\sqcap} \{x : C, x : D\} \cup S$, if

   (a) $x : C \sqcap D$ is in S,
   (b) $x : C$ and $x : D$ are not both in S.

2. $S \to_{\sqcup} \{x : E\} \cup S$, if

   (a) $x : C \sqcup D$ is in S,
   (b) neither $x : C$ nor $x : D$ is in S,
   (c) $E = C$ or $E = D$.

3. $S \to_{\forall} \{y : C\} \cup S$, if

   (a) $x : \forall R.C$ is in S,
   (b) xRy is in S,
   (c) $y : C$ is not in S.

4. $S \to_{\exists} \{xRy, y : C\} \cup S$, if

   (a) $x : \exists R.C$ is in S,
   (b) y is a new variable,
   (c) there is no such z such that both xRz and $z : C$ are in S.

Tableau Example — Constraint Syntax

Satisfiability of:

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

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

$x : (\forall \texttt{CHILD}.\texttt{Male})$	$\sqcap$-rule
$x : (\exists \texttt{CHILD}.\neg\texttt{Male})$	$\sqcap$-rule
x CHILD y	$\exists$-rule
$y : \neg\texttt{Male}$	$\exists$-rule
$y : \texttt{Male}$	$\forall$-rule
⟨*CLASH*⟩	

CLASH: unsatisfiable constraint system $\{x : A, x : \neg A\}$

Another Tableau Example

Satisfiability of:

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

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

$x : (\forall \texttt{CHILD}.\texttt{Male})$	$\sqcap$-rule
$x : (\exists \texttt{CHILD}.\texttt{Male})$	$\sqcap$-rule
x CHILD y	$\exists$-rule
$y : \texttt{Male}$	$\exists$-rule
$y : \texttt{Male}$	$\forall$-rule
⟨*COMPLETED*⟩	

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 rule applications.

Intuitively, this can be proved by using the following argument:

All rules but $\rightarrow_\forall$ 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 interpretation 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) between elements of the interpretation domain that must hold.

Complexity with Non-Cyclic Terminologies

Expressivity		$\Sigma \models C \sqsubseteq D$	$\Sigma \not\models$	$\Sigma \models C(a)$
$C \sqcap D$ $\forall R.C$ $\exists R$	$\mathcal{FL}^-$	P (*)	P	P (*)
$\neg A$	$\mathcal{AL}$	P (*)	P (*)	P (*)
$\exists R.C$	$\mathcal{ALE}$	NP	coNP	PSPACE
$\neg C$	$\mathcal{ALC}/K(n)$		PSPACE !!	
$\{a_1 \ldots\}$	$\mathcal{ALCO}$		PSPACE	
	$\mathcal{PDL}$		EXPTIME	
	KL-ONE		undecidable	

Traces

To obtain a PSPACE algorithm, exploit independency between *traces* of a satisfiability proof:

- A *completed 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 particular trace.
- A completed constraint system denotes a model, so traces correspond to paths in the graph from the starting node.
- A trace has polynomial size!
- Nodes in a constraint system are only generated by the completion rule for the *existential constraint*.

Sources of Complexity

A deterministic version of the tableau calculus can be seen as a depth-first exploration of an AND-OR tree:

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

Exponential-time behaviour of the calculus has two origins:

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

Sources of Complexity (2)

In contrast to databases (and, in general, to static data structures), description logics do not only handle ground and complete knowledge, but perform also reasoning on incomplete knowledge and case analyses:

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

Extensions of $\mathcal{ALC}$

Constructor	Syntax	Semantics
cardinality ($\mathcal{N}$)	$\exists^{\leq n} R$	$\{x \mid \|\{y \mid R^{\mathcal{I}}(x,y)\}\| \leq n\}$
	$\exists^{\geq n} R$	$\{x \mid \|\{y \mid R^{\mathcal{I}}(x,y)\}\| \geq n\}$
enumeration ($\mathcal{O}$)	$\{a_1, \ldots, a_n\}$	$\{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$
selection ($\mathcal{O}$)	$f : C$	$\{x \in \mathrm{dom} f^{\mathcal{I}} \mid C^{\mathcal{I}}(f^{\mathcal{I}}(x))\}$
coreference restr.	$P \downarrow Q$	$\{x \mid P^{\mathcal{I}}(x) = Q^{\mathcal{I}}(x)\}$

In terminological systems it is not assumed that a given terminology and domain description are complete (open world assumption). Given a specification that e.g. a certain person has only sons, it is not concluded that this person has no daughters.

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:

- $\mathtt{Busy-Woman} \doteq \mathtt{Woman} \sqcap (\exists^{\geq 3}\mathtt{CHILD})$
- $\mathtt{Conscious-Woman} \doteq \mathtt{Woman} \sqcap (\exists^{\leq 5}\mathtt{CHILD})$

$$\boxed{(\exists^{\geq 1}) \Longleftrightarrow (\exists R)}$$

Roles as Functions

- A role is *functional* if the filler depends functionally on the individual, i.e., the role can be considered as a function:
  $R(x,y) \Leftrightarrow f(x) = y$.
  Those roles are also called (genuine) *attributes* or *features*.
- 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 and Enumeration Types

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*; it is usually assumed in database applications.

Enumeration Types

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

Extending Description Logics

- Aggregation and abstraction operators
- Epistemic queries
- Closed world assumption
- Negation as failure
- Default values
- Beliefs
- Probability and similarity-based reasoning
- Generalized quantifiers and plural entities
- Ontological primitives
  time, events, space, topology, parts and wholes (mereology)

Decidability and Complexity Results

Name	Concept Forming Op.	Role-Forming Op.	Complexity of Subsumption
$\mathcal{FL}$	$C \sqcap D, \forall R.C,$ $\exists R$	$R\|_C$	coNP-hard, without $R\|_C$ in P [Brachman, Levesque 84]
$\mathcal{ALE}$	$C \sqcap D, \forall R.C,$ $\exists R.C$		NP-complete [Donini et al. 90]
BACK (KANDOR)	$C \sqcap D, \forall R.C,$ $\exists^{\geq n} R, \exists^{\leq n} R$	$R \sqcap R'$	coNP-hard, even without $R \sqcap R'$ but $\exists^{\geq n} R.C$ [Nebel 88]
$\mathcal{ALE}$	$C \sqcap D, C \sqcup D,$ $\neg C, \forall R.C, \exists R.C$		PSPACE-complete [Schmidt-Schauss, Smolka 88]
$\mathcal{R}$		$R \sqcap R', \neg R,$ $(R_1 \ldots R_n)$	undecidable [Schild 88]
KL-ONE	$C \sqcap D, \forall R.C,$ $P \downarrow Q$		undecidable [Schmidt-Sch. 89] [Patel-Schneider 89]