OWL

- Three species of OWL
  - OWL full is union of OWL syntax and RDF (Undecidable)
  - OWL DL restricted to FOL fragment (decidable in NEXPTIME)
  - OWL Lite is “easier to implement” subset of OWL DL (decidable in EXPTIME)
- Semantic layering
  - OWL DL within Description Logic (DL) fragment
- OWL DL is based on $\mathcal{SHOIN}(D_n)$ DL
- OWL Lite is based on $\mathcal{SHIF}(D_n)$ DL
- OWL 2 is based on $\mathcal{SROIQ}(D_n)$ DL

Description Logics (DLs)

- The logics behind OWL-DL and OWL-Lite, `http://dl.kr.org/`.
- Concept/Class: names are equivalent to unary predicates
  - In general, concepts equiv to formulae with one free variable
- Role or attribute: names are equivalent to binary predicates
  - In general, roles equiv to formulae with two free variables
- Taxonomy: Concept and role hierarchies can be expressed
- Individual: names are equivalent to constants
- Operators: restricted so that:
  - Language is decidable and, if possible, of low complexity
  - No need for explicit use of variables
    - Restricted form of $\exists$ and $\forall$
  - Features such as counting can be succinctly expressed

The DL Family

- A given DL is defined by set of concept and role forming operators
- Basic language: $\mathcal{ALC}$($\mathcal{A}$ttributive $\mathcal{L}$anguage with $\mathcal{C}$omplement)

Syntax				*Semantics*	*Example*
C, D	$\rightarrow$	$\top$	$\mid$	$\top(x)$	
		$\bot$	$\mid$	$\bot(x)$	
		A	$\mid$	$A(x)$	*Human*
		$C \sqcap D$	$\mid$	$C(x) \wedge D(x)$	*Human* $\sqcap$ *Male*
		$C \sqcup D$	$\mid$	$C(x) \vee D(x)$	*Nice* $\sqcup$ *Rich*
		$\neg C$	$\mid$	$\neg C(x)$	$\neg$*Meat*
		$\exists R.C$	$\mid$	$\exists y.R(x, y) \wedge C(y)$	$\exists$*has_child*.*Blond*
		$\forall R.C$		$\forall y.R(x, y) \Rightarrow C(y)$	$\forall$*has_child*.*Human*
	$C \sqsubseteq D$			$\forall x.C(x) \Rightarrow D(x)$	*Happy_Father* $\sqsubseteq$ *Man* $\sqcap$ $\exists$*has_child*.*Female*
	a:*C*			$C(a)$	*John*:*Happy_Father*

Toy Example

$$
\begin{aligned}
\mathit{Sex} &= \mathit{Male} \sqcup \mathit{Female} \\
\mathit{Male} \sqcap \mathit{Female} &\sqsubseteq \bot \\
\mathit{Person} &\sqsubseteq \mathit{Human} \sqcap \exists \mathit{hasSex}.\mathit{Sex} \\
\mathit{MalePerson} &= \mathit{Person} \sqcap \exists \mathit{hasSex}.\mathit{Male} \\
&\quad \mathit{functional}(\mathit{hasSex})
\end{aligned}
$$

$$\mathit{umberto}{:}\mathit{Person} \sqcap \exists \mathit{hasSex}.\neg \mathit{Female}$$

$$\mathit{KB} \models \mathit{umberto}{:}\mathit{MalePerson}$$

Note on DL Naming

$\mathcal{AL}$: $C, D \longrightarrow \top \mid \bot \mid A \mid C \sqcap D \mid \neg A \mid \exists R.\top \mid \forall R.C$

$\mathcal{C}$: Concept negation, $\neg C$. Thus, $\mathcal{ALC} = \mathcal{AL} + \mathcal{C}$

$\mathcal{S}$: Used for $\mathcal{ALC}$ with transitive roles $\mathcal{R}_+$

$\mathcal{U}$: Concept disjunction, $C_1 \sqcup C_2$

$\mathcal{E}$: Existential quantification, $\exists R.C$

$\mathcal{H}$: Role inclusion axioms, $R_1 \sqsubseteq R_2$, e.g. *is_component_of* $\sqsubseteq$ *is_part_of*

$\mathcal{N}$: Number restrictions, $(\geq n\ R)$ and $(\leq n\ R)$, e.g. $(\geq 3$ *has_Child*$)$ (has at least 3 children)

$\mathcal{Q}$: Qualified number restrictions, $(\geq n\ R.C)$ and $(\leq n\ R.C)$, e.g. $(\leq 2$ *has_Child*.*Adult*$)$ (has at most 2 adult children)

$\mathcal{O}$: Nominals (singleton class), $\{a\}$, e.g. $\exists$*has_child*.$\{$*mary*$\}$. **Note**: a:C equiv to $\{a\} \sqsubseteq C$ and (a, b):R equiv to $\{a\} \sqsubseteq \exists R.\{b\}$

$\mathcal{I}$: Inverse role, R^-, e.g. $isPartOf = hasPart^-$

$\mathcal{F}$: Functional role, f, e.g. *functional*(*hasAge*)

$\mathcal{R}_+$: transitive role, e.g. *transitive*(*isPartOf*)

For instance,

$$\begin{array}{rcll} \mathcal{SHIF} & = & \mathcal{S} + \mathcal{H} + \mathcal{I} + \mathcal{F} = \mathcal{ALCR}_+\mathcal{HIF} & \text{OWL-Lite} \\ \mathcal{SHOIN} & = & \mathcal{S} + \mathcal{H} + \mathcal{O} + \mathcal{I} + \mathcal{N} = \mathcal{ALCR}_+\mathcal{HOIN} & \text{OWL-DL} \\ \mathcal{SROIQ} & = & \mathcal{S} + \mathcal{R} + \mathcal{O} + \mathcal{I} + \mathcal{Q} = \mathcal{ALCR}_+\mathcal{ROIN} & \text{OWL 2} \end{array}$$

Semantics of Additional Constructs

$\mathcal{H}$: Role inclusion axioms, $\mathcal{I} \models R_1 \sqsubseteq R_2$ iff ${R_1}^{\mathcal{I}} \subseteq {R_1}^{\mathcal{I}}$

$\mathcal{N}$: Number restrictions,
$(\geq n\ R)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} : |\{y \mid \langle x, y\rangle \in R^{\mathcal{I}}\}| \geq n\}$,
$(\leq n\ R)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} : |\{y \mid \langle x, y\rangle \in R^{\mathcal{I}}\}| \leq n\}$

$\mathcal{Q}$: Qualified number restrictions,
$(\geq n\ R.C)^{\mathcal{I}} = \{x \in |\{y \mid \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}| \geq n\}$,
$(\leq n\ R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} : |\{y \mid \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}| \leq n\}$

$\mathcal{O}$: Nominals (singleton class), $\{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}$

$\mathcal{I}$: Inverse role, $(R^{-})^{\mathcal{I}} = \{\langle x, y\rangle \mid \langle y, x\rangle \in R^{\mathcal{I}}\}$

$\mathcal{F}$: Functional role, $I \models \mathit{fun}(f)$ iff $\forall z \forall y \forall z$ if $\langle x, y\rangle \in f^{\mathcal{I}}$ and $\langle x, z\rangle \in f^{\mathcal{I}}$ the $y = z$

$\mathcal{R}_{+}$: transitive role,
$(R_{+})^{\mathcal{I}} = \{\langle x, y\rangle \mid \exists z$ such that $\langle x, z\rangle \in R^{\mathcal{I}} \wedge \langle z, y\rangle \in R^{\mathcal{I}}\}$

Basics on Concrete Domains

- Concrete domains: reals, integers, strings, ...

  $(tim, 14)$*:hasAge*
  $(sf, \text{“}SoftComputing\text{”})$*:hasAcronym*
  $(source1, \text{“}ComputerScience\text{”})$*:isAbout*
  $(service2, \text{“}InformationRetrievalTool''\text{”})$*:Matches*
  $Minor = Person \sqcap \exists hasAge.\ \leq_{18}$

- Semantics: a clean separation between “object” classes and concrete domains
  - $D = \langle \Delta_D, \Phi_D \rangle$
  - Δ_D is an interpretation domain
  - Φ_D is the set of concrete domain predicates d with a predefined arity n and fixed interpretation $d^D \subseteq \Delta_D^n$
  - Concrete properties: $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta_D$
- Notation: (D). E.g., $\mathcal{ALC}(D)$ is $\mathcal{ALC}$ + concrete domains

- Example: assume $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ such that

$$\begin{aligned} \Delta^{\mathcal{I}} &= \{a, b, c, d, e, f, 1, 2, 4, 5, 8\} \\ \mathit{Person}^{\mathcal{I}} &= \{a, b, c, d\} \end{aligned}$$

- Consider the following concrete domain with of some unary predicates ($n = 1$) over reals
  - $\Delta_D = \mathbb{R}$,
  - $\Phi_D = \{=_m, \geq_m, \leq_m, >_m, <_m \mid m \in \mathbb{R}\}$
  - the fixed interpretation of the predicates is

$$\begin{aligned} (=_m)^D &= \{m\} & \qquad (>_m)^D &= \{k \mid k > m\} \\ (\geq_m)^D &= \{k \mid k \geq m\} & (<_m)^D &= \{k \mid k < m\} \\ (\leq_m)^D &= \{k \mid k \leq m\} \end{aligned}$$

  - Concrete properties: $\mathit{hasAge}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \mathbb{R}$

$$\mathit{hasAge}^{\mathcal{I}} = \{\langle a, 9\rangle, \langle c, 20\rangle, \langle b, 12\rangle\}$$

  - What is the interpretation of $\mathit{Person} \sqcap \exists \mathit{hasAge}.\ \leq_{18}$?

$$\begin{aligned} &(\mathit{Person} \sqcap \exists \mathit{hasAge}.\ \leq_{18})^{\mathcal{I}} \\ &\quad = \mathit{Person}^{\mathcal{I}} \cap \{x \mid \exists y \in \mathbb{R} \text{ such that } \langle x, y\rangle \in \mathit{hasAge}^{\mathcal{I}} \wedge y \in (\leq_{18})^{\mathcal{I}}\} \\ &\quad = \{a, b, c, d\} \cap \{x \mid \exists y. \langle x, y\rangle \in \{\langle a, 9\rangle, \langle c, 20\rangle, \langle b, 12\rangle\} \wedge y \leq 18\} \\ &\quad = \{a, b, c, d\} \cap \{a, b\} \\ &\quad = \{a, b\} \end{aligned}$$

DL Knowledge Base

- A DL Knowledge Base is a pair $KB = \langle \mathcal{T}, \mathcal{A} \rangle$, where
  - $\mathcal{T}$ is a TBox
    - containing general inclusion axioms of the form $C \sqsubseteq D$,
    - concept definitions of the form $A = C$
    - primitive concept definitions of the form $A \sqsubseteq C$
    - role inclusions of the form $R \sqsubseteq P$
    - role equivalence of the form $R = P$
  - $\mathcal{A}$ is a ABox
    - containing assertions of the form $a{:}C$
    - containing assertions of the form $(a, b){:}R$
    - containing (in) equality Axioms of the form $a = b$ and $a \neq b$
- An interpretation $\mathcal{I}$ is a model of KB, written $\mathcal{I} \models KB$ iff $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$, where
  - $\mathcal{I} \models \mathcal{T}$ ($\mathcal{I}$ is a model of $\mathcal{T}$) iff $\mathcal{I}$ is a model of each element in $\mathcal{T}$
  - $\mathcal{I} \models \mathcal{A}$ ($\mathcal{I}$ is a model of $\mathcal{A}$) iff $\mathcal{I}$ is a model of each element in $\mathcal{A}$

OWL DL as Description Logic

Concept/Class constructors:

Abstract Syntax	DL Syntax	Example
Descriptions (C)		
A (URI reference)	A	`Conference`
`owl:Thing`	$\top$	
`owl:Nothing`	$\bot$	
`intersectionOf`(C_1 C_2 ...)	$C_1 \sqcap C_2$	`Reference` $\sqcap$ `Journal`
`unionOf`(C_1 C_2 ...)	$C_1 \sqcup C_2$	`Organization` $\sqcup$ `Institution`
`complementOf`(C)	$\neg C$	$\neg$ `MasterThesis`
`oneOf`(o_1 ...)	$\{o_1, \ldots\}$	`{"WISE","ISWC",...}`
`restriction`(R `someValuesFrom`(C))	$\exists R.C$	$\exists$`parts.InCollection`
`restriction`(R `allValuesFrom`(C))	$\forall R.C$	$\forall$`date.Date`
`restriction`(R `hasValue`(o))	$\exists R.\{o\}$	$\exists$`date.{2005}`
`restriction`(R `minCardinality`(n))	$(\geq n\, R)$	$(\geqslant 1$ `location`$)$
`restriction`(R `maxCardinality`(n))	$(\leq n\, R)$	$(\leqslant 1$ `publisher`$)$
`restriction`(U `someValuesFrom`(D))	$\exists U.D$	$\exists$`issue.integer`
`restriction`(U `allValuesFrom`(D))	$\forall U.D$	$\forall$`name.string`
`restriction`(U `hasValue`(v))	$\exists U.=_v\}$	$\exists$`series.=`"`LNCS`"
`restriction`(U `minCardinality`(n))	$(\geq n\, U)$	$(\geqslant 1$ `title`$)$
`restriction`(U `maxCardinality`(n))	$(\leq n\, U)$	$(\leqslant 1$ `author`$)$

Note: R is an abstract role, while U is a concrete property of arity two.

Axioms:

Abstract Syntax	DL Syntax	Example
Axioms		
`Class`(A `partial` $C_1 \ldots C_n$)	$A \sqsubseteq C_1 \sqcap \ldots \sqcap C_n$	$Human \sqsubseteq Animal \sqcap Biped$
`Class`(A `complete` $C_1 \ldots C_n$)	$A = C_1 \sqcap \ldots \sqcap C_n$	$Man = Human \sqcap Male$
`EnumeratedClass`($A\ o_1 \ldots o_n$)	$A = \{o_1\} \sqcup \ldots \sqcup \{o_n\}$	$RGB = \{r\} \sqcup \{g\} \sqcup \{b\}$
`SubClassOf`($C_1\ C_2$)	$C_1 \sqsubseteq C_2$	
`EquivalentClasses`($C_1 \ldots C_n$)	$C_1 = \ldots = C_n$	
`DisjointClasses`($C_1 \ldots C_n$)	$C_i \sqcap C_j = \bot, i \neq j$	$Male \sqcap Female \sqsubseteq \bot$
`ObjectProperty`(R `super` $(R_1) \ldots$ `super` (R_n)	$R \sqsubseteq R_i$	$HasDaughter \sqsubseteq hasChild$
`domain`(C_1) ... `domain`(C_n)	$(\geq 1\ R) \sqsubseteq C_i$	$(\geq 1\ hasChild) \sqsubseteq Human$
`range`(C_1) ... `range`(C_n)	$\top \sqsubseteq \forall R.C_i$	$\top \sqsubseteq \forall hasChild.Human$
[`inverseof`(P)]	$R = P^-$	$hasChild = hasParent^-$
[`symmetric`]	$R \sqsubseteq R^-$	$similar = similar^-$
[`functional`]	$\top \sqsubseteq (\leq 1\ R)$	$\top \sqsubseteq (\leq 1\ hasMother)$
[`Inversefunctional`]	$\top \sqsubseteq (\leq 1\ R^-)$	
[`Transitive`])	$Tr(R)$	$Tr(ancestor)$
`SubPropertyOf`($R_1\ R_2$)	$R_1 \sqsubseteq R_2$	
`EquivalentProperties`($R_1 \ldots R_n$)	$R_1 = \ldots = R_n$	$cost = price$
`AnnotationProperty`(S)		

Abstract Syntax	DL Syntax	Example
DatatypeProperty(U super (U_1) ... super (U_n) domain(C_1) ...domain(C_n) range(D_1) ...range(D_n) [functional])	$U \sqsubseteq U_i$ $(\geq 1\ U) \sqsubseteq C_i$ $\top \sqsubseteq \forall U.D_i$ $\top \sqsubseteq (\leq 1\ U)$	 $(\geq 1\ hasAge) \sqsubseteq Human$ $\top \sqsubseteq \forall hasAge.posInteger$ $\top \sqsubseteq (\leq 1\ hasAge)$
SubPropertyOf($U_1\ U_2$)	$U_1 \sqsubseteq U_2$	$hasName \sqsubseteq hasFirstName$
EquivalentProperties($U_1 \ldots U_n$)	$U_1 = \ldots = U_n$	
Individuals		
Individual(o type (C_1) ... type (C_n)) value($R_1\ o_1$) ...value($R_n o_n$) value($U_1\ v_1$) ...value($U_n v_n$)	$o{:}C_i$ $(o, o_i){:}R_i$ $(o, v_1){:}U_i$	$tim{:}Human$ $(tim, mary){:}hasChild$ $(tim, 14){:}hasAge$
SameIndividual($o_1 \ldots o_n$)	$o_1 = \ldots = o_n$	$president\_Bush = G.W.Bush$
DifferentIndividuals($o_1 \ldots o_n$)	$o_i \neq o_j, i \neq j$	$john \neq peter$
Symbols		
Object Property R (URI reference)	R	hasChild
Datatype Property U (URI reference)	U	hasAge
Individual o (URI reference)	U	tim
Data Value v (RDF literal)	U	“International Conference on Semantic W

XML representation of OWL statements

Example:

$Person \sqcap \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$

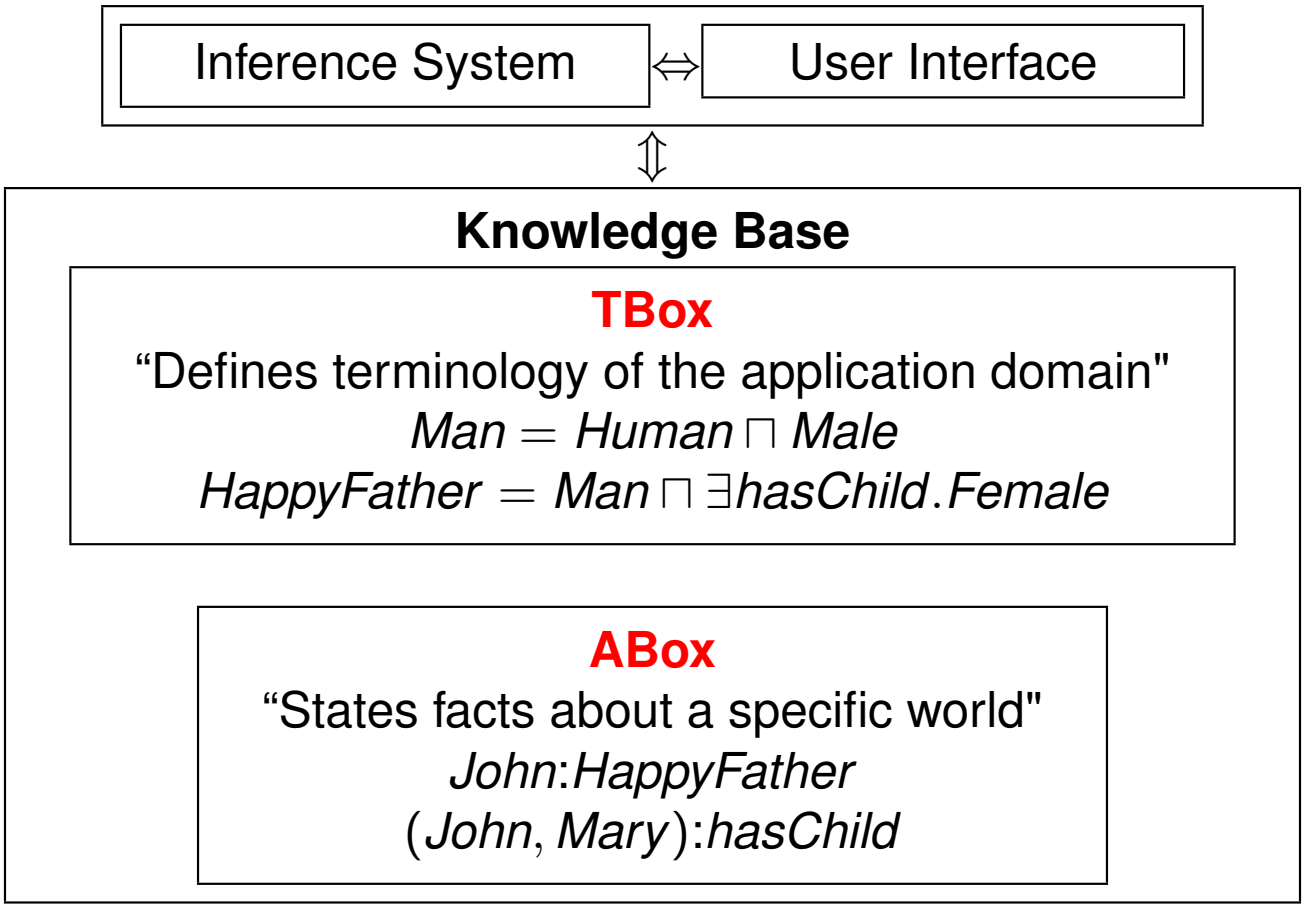
```
intersectionOf(
  Person
  restriction(hasChild
                allValuesFrom(unionOf(Doctor
                                      restriction(hasChild
                                                  someValuesFrom(Doctor)))))
```

```
<owl:Class>
  <owl:intersectionOf rdf:parseType=" collection">
    <owl:Class rdf:about="#Person"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild"/>
      <owl:allValuesFrom>
        <owl:unionOf rdf:parseType=" collection">
          <owl:Class rdf:about="#Doctor"/>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasChild"/>
            <owl:someValuesFrom rdf:resource="#Doctor"/>
          </owl:Restriction>
        </owl:unionOf>
      </owl:allValuesFrom>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

Description Logic Reasoning

- What can we do with it?
  - Design and maintenance of ontologies
    - Check class consistency and compute class hierarchy
    - Particularly important with large ontologies/multiple authors
  - Integration of ontologies
    - Assert inter-ontology relationships
    - Reasoner computes integrated class hierarchy/consistency
  - Querying class and instance data w.r.t. ontologies
    - Determine if set of facts are consistent w.r.t. ontologies
    - Determine if individuals are instances of ontology classes
    - Retrieve individuals/tuples satisfying a query expression
    - Check if one class subsumes (is more general than) another w.r.t. ontology
- How do we do it?
  - Use DLs reasoner (OWL DL = $\mathcal{SHOIN}(\mathrm{D})$)
    - Formal properties well understood (complexity, decidability)
    - Known practical reasoning algorithms
    - Implemented systems (highly optimised)

Description Logic System



Basic Inference Problems (Formally)

Consistency: Check if knowledge is meaningful

- Is KB satisfiability? $\mapsto$ Is there some model $\mathcal{I}$ of KB ?
- Is C satisfiability? $\mapsto$ $C^{\mathcal{I}} \neq \emptyset$ for some some model $\mathcal{I}$ of KB ?

Subsumption: structure knowledge, compute taxonomy

- $KB \models C \sqsubseteq D$? $\mapsto$ Is it true that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of KB ?

Equivalence: check if two classes denote same set of instances

- $KB \models C = D$? $\mapsto$ Is it true that $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all models $\mathcal{I}$ of KB ?

Instantiation: check if individual a instance of class C

- $KB \models a{:}C$? $\mapsto$ Is it true that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all models $\mathcal{I}$ of KB ?

Retrieval: retrieve set of individuals that instantiate C

- Compute the set $\{a \mid KB \models a{:}C\}$

Reduction to Satisfiability

Problems are all reducible to KB satisfiability

Subsumption: $KB \models C \sqsubseteq D$ iff $\langle \mathcal{T}, \mathcal{A} \cup \{a{:}C \sqcap \neg D\} \rangle$ not satisfiable, where a is a new individual

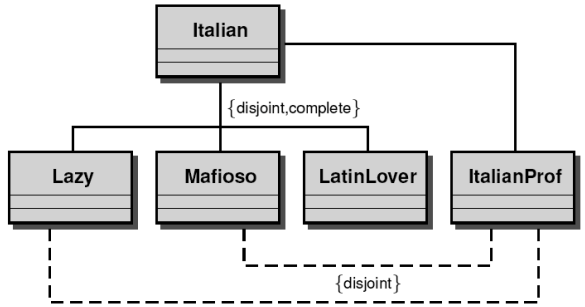
Equivalence: $KB \models C = D$ iff $KB \models C \sqsubseteq D$ and $KB \models D \sqsubseteq C$

Instantiation: $KB \models a{:}C$ iff $\langle \mathcal{T}, \mathcal{A} \cup \{a{:}\neg C\} \rangle$ not satisfiable

Retrieval: The computation of the set $\{a \mid KB \models a{:}C\}$ is reducible to the instance checking problem

Exercise

Problem: Consider the following conceptual schema



Encode it into Description logics and prove that $KB \models ItalianProf \sqsubseteq LatinLover$

Solution:

$$
\begin{aligned}
Lazy &\sqsubseteq Italian \\
Mafioso &\sqsubseteq Italian \\
LatinLover &\sqsubseteq Italian \\
Italian &\sqsubseteq (Lazy \sqcup Mafioso \sqcup LatinLover) \\
ItalianProf &\sqsubseteq Italian \\
Lazy &\sqsubseteq \neg Mafioso \\
Lazy &\sqsubseteq \neg LatinLover \\
Mafioso &\sqsubseteq \neg LatinLover \\
Mafioso &\sqsubseteq \neg ItalianProf \\
Lazy &\sqsubseteq \neg ItalianProf
\end{aligned}
$$

Reasoning in DLs: Basics

- Tableaux algorithm deciding satisfiability
- Try to build a tree-like model $\mathcal{I}$ of the KB
- Decompose concepts C syntactically
  - Apply tableau expansion rules
  - Infer constraints on elements of model
- Tableau rules correspond to constructors in logic ($\sqcap, \sqcup, \ldots$)
  - Some rules are nondeterministic (e.g., $\sqcup, \leq$)
  - In practice, this means search
- Stop when no more rules applicable or clash occurs
  - Clash is an obvious contradiction, e.g., $A(x), \neg A(x)$
- Cycle check (blocking) may be needed for termination

Negation Normal Form (NNF)

- We have to transform concepts into Negation Normal Form: push negation inside using de Morgan' laws

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

and

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

Completion-Forest

- This is a forest of trees, where
  - each node x is labelled with a set $\mathcal{L}(x)$ of concepts
  - each edge $\langle x, y \rangle$ is labelled with $\mathcal{L}(\langle x, y \rangle) = \{R\}$ for some role R (edges correspond to relationships between pairs of individuals)
- The forest is initialized with
  - a root node a, labelled $\mathcal{L}(x) = \emptyset$ for each individual a occurring in the KB
  - an edge $\langle a, b \rangle$ labelled $\mathcal{L}(\langle a, b \rangle) = \{R\}$ for each (a, b):R occurring in the KB
- Then, for each a:C occurring in the KB, set $\mathcal{L}(a) \longrightarrow \mathcal{L}(a) \cup \{C\}$
- The algorithm expands the tree either by extending $\mathcal{L}(x)$ for some node x or by adding new leaf nodes.
- Edges are added when expanding $\exists R.C$
- A completion-forest contains a clash if, for a node x, $\{C, \neg C\} \subseteq \mathcal{L}(x)$
- If nodes x and y are connected by an edge$\langle x, y \rangle$, then y is called a successor of x and x is called a predecessor of y. Ancestor is the transitive closure of predecessor.
- A node y is called an R-successor of a node x if y is a successor of x and $\mathcal{L}(\langle x, y \rangle) = \{R\}$.
- The algorithm returns “satisfiable" if rules can be applied s.t. they yield a clash-free, complete (no more rules can be applied) completion forest

$\mathcal{ALC}$ Tableau rules without GCI's

Rule		Description
$(\sqcap)$	if 1.	$C_1 \sqcap C_2 \in \mathcal{L}(x)$ and
	2.	$\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$
	then	$\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
$(\sqcup)$	if 1.	$C_1 \sqcup C_2 \in \mathcal{L}(x)$ and
	2.	$\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
	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	$\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{C\}$

Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

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

$$x$$

Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

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

x

Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

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

$$x$$

Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

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

$$x$$

Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

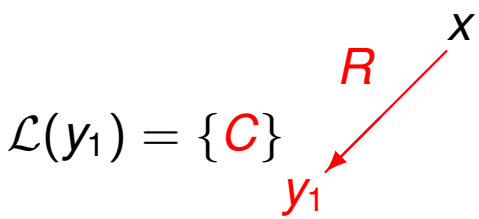
$$\mathcal{L}(x) = \{\exists R.C, \forall R.(\neg C \sqcup \neg D), \exists R.D\}$$

$$x$$

Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

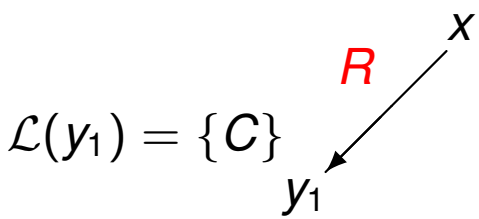
$$\mathcal{L}(x) = \{\exists R.C, \forall R.(\neg C \sqcup \neg D), \exists R.D\}$$



Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

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

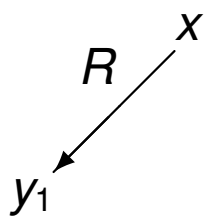


Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

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

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

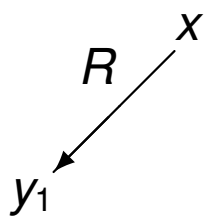


Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

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

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

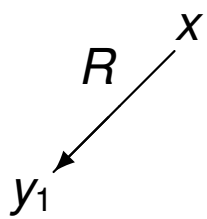


Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

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

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

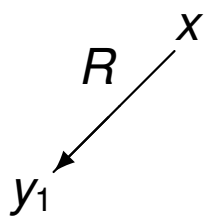


Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

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

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



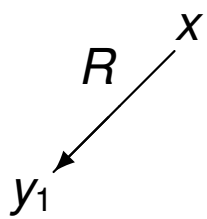
Clash

Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

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

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

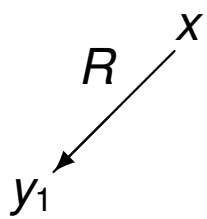


Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

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

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

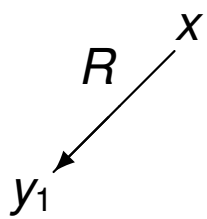


Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

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

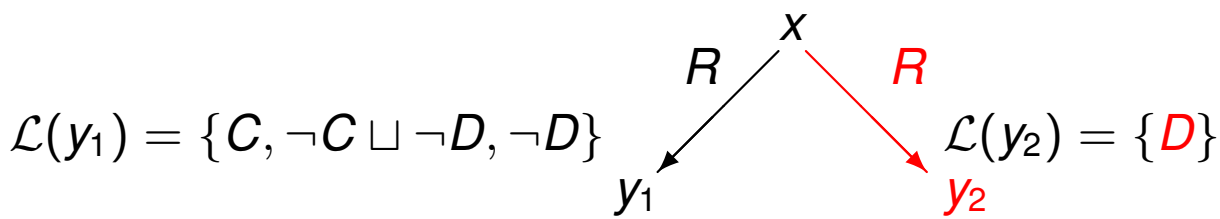
$$\mathcal{L}(y_1) = \{C, \neg C \sqcup \neg D, \neg D\}$$



Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

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

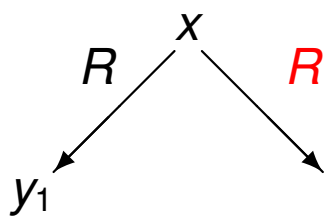


Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

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

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

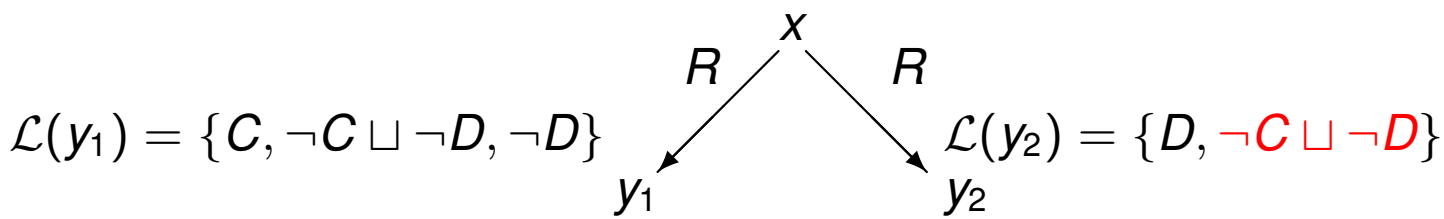


$\mathcal{L}(y_2) = \{D\}$

Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

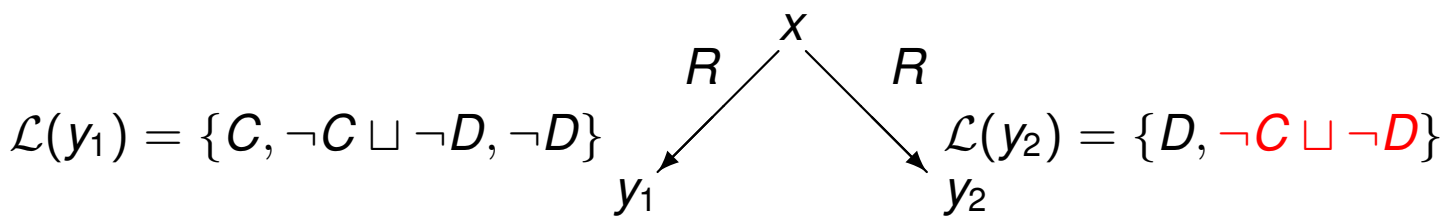
$$\mathcal{L}(x) = \{\exists R.C, \forall R.(\neg C \sqcup \neg D), \exists R.D\}$$



Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

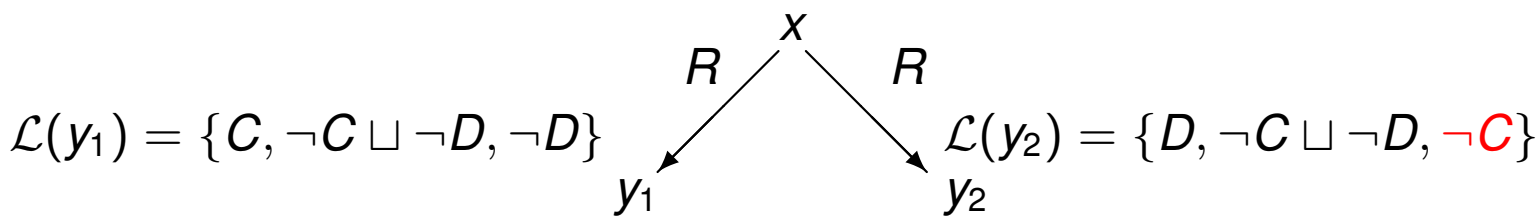
$$\mathcal{L}(x) = \{\exists R.C, \forall R.(\neg C \sqcup \neg D), \exists R.D\}$$



Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

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

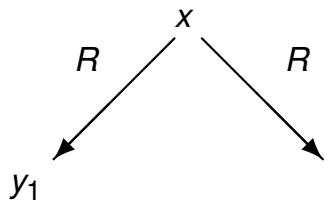


Example

Is $\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D$ satisfiable? Yes.

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

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



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

- Finished. No more rules applicable and the tableau is complete and clash-free
- Hence, the concept is satisfiable
- The tree corresponds to a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$
  - The nodes are the elements of the domain: $\Delta^{\mathcal{I}} = \{x, y_1, y_2\}$
  - For each atomic concept A, set $A^{\mathcal{I}} = \{z \mid A \in \mathcal{L}(z)\}$
    - $C^{\mathcal{I}} = \{y_1\}$, $D^{\mathcal{I}} = \{y_2\}$
  - For each role R, set $R^{\mathcal{I}} = \{\langle x, y\rangle \mid$ there is an edge labeled R from x to $y\}$
    - $R^{\mathcal{I}} = \{\langle x, y_1\rangle, \langle x, y_2\rangle\}$
  - It can be shown that $x \in (\exists R.C \sqcap \forall R.(\neg C \sqcup \neg D) \sqcap \exists R.D)^{\mathcal{I}} \neq \emptyset$

Example

Is $\exists R.C \sqcap \forall R.\neg C$ satisfiable? No.

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

x

Example

Is $\exists R.C \sqcap \forall R.\neg C$ satisfiable? No.

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

x

Example

Is $\exists R.C \sqcap \forall R.\neg C$ satisfiable? No.

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

x

Example

Is $\exists R.C \sqcap \forall R.\neg C$ satisfiable? No.

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

x

Example

Is $csomeR.C \sqcap \forall R.\neg C$ satisfiable? No.

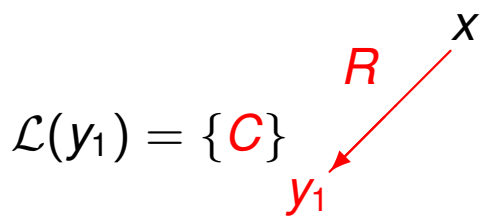
$$\mathcal{L}(x) = \{\exists R.C, \forall R.\neg C\}$$

x

Example

Is $\exists R.C \sqcap \forall R.\neg C$ satisfiable? No.

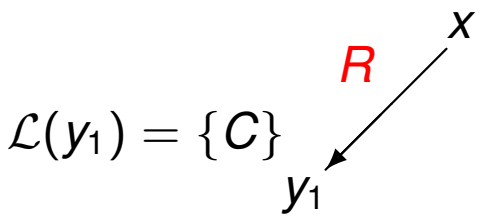
$$\mathcal{L}(x) = \{\exists R.C, \forall R.\neg C\}$$



Example

Is $\exists R.C \sqcap \forall R.\neg C$ satisfiable? No.

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

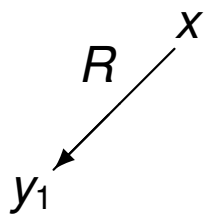


Example

Is $\exists R.C \sqcap \forall R.\neg C$ satisfiable? No.

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

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

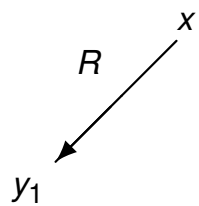


Example

Is $\exists R.C \sqcap \forall R.\neg C$ satisfiable? No.

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

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



Clash

- Finished. No more rules applicable and the tableau is complete, but not clash-free
- Hence, the concept is not satisfiable
- I.e. no model can be built, e.g.
  - $\Delta^{\mathcal{I}} = \{x, y_1\}$
  - $C^{\mathcal{I}} = \{y_1\}$
  - $R^{\mathcal{I}} = \{\langle x, y_1 \rangle\}$
  - is not a model because

$$(\exists R.C \sqcap \forall R.\neg C)^{\mathcal{I}} = (\exists R.C)^{\mathcal{I}} \cap (\forall R.\neg C)^{\mathcal{I}} = \{x\} \cap \emptyset = \emptyset$$

Soundness and Completeness

Theorem

Let $\mathcal{A}$ be an $\mathcal{ALC}$ ABox and F a completion-forest obtained by applying the tableau rules to $\mathcal{A}$. Then

1. *The rule application terminates;*
2. *If F is clash-free and complete, then F defines a (canonical) (tree) model for $\mathcal{A}$; and*
3. *If $\mathcal{A}$ has a model $\mathcal{I}$, then the rules can be applied such that they yield a clash-free and complete completion-forest.*

KBs with GCIs

- We have seen how to test the satisfiability of an ABox $\mathcal{A}$
- But, how can we check if a KB $KB = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable with $\mathcal{T} \neq \emptyset$?
- Basic idea: since $t(C \sqsubseteq D) \equiv \forall x.\neg t(C, x) \vee t(D, x)$
  - we use the rule: for each $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to every node
- But, termination is not guaranteed
  - E.g., consider $KB = \langle \mathcal{T}, \mathcal{A} \rangle$

$$\begin{aligned} \mathcal{T} &= \{Human \sqsubseteq \exists hasMother.Human\} \\ \mathcal{A} &= \{umberto{:}Human\} \end{aligned}$$

$\mathcal{L}(umberto) = \{Human, \neg Human \sqcup \exists hasMother.Human\}$ *umberto*

KBs with GCIs

- We have seen how to test the satisfiability of an ABox $\mathcal{A}$
- But, how can we check if a KB $KB = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable with $\mathcal{T} \neq \emptyset$?
- Basic idea: since $t(C \sqsubseteq D) \equiv \forall x.\neg t(C, x) \vee t(D, x)$
  - we use the rule: for each $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to every node
- But, termination is not guaranteed
  - E.g., consider $KB = \langle \mathcal{T}, \mathcal{A} \rangle$

$$\begin{aligned} \mathcal{T} &= \{Human \sqsubseteq \exists hasMother.Human\} \\ \mathcal{A} &= \{umberto{:}Human\} \end{aligned}$$

$\mathcal{L}(umberto) = \{Human, \neg Human \sqcup \exists hasMother.Human\}$ *umberto*

KB Satisfiability

- We have seen how to test the satisfiability of an ABox $\mathcal{A}$
- But, how can we check if a KB $KB = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable?
- Basic idea: since $t(C \sqsubseteq D) \equiv \forall x.\neg t(C, x) \vee t(D, x)$
  - we use the rule: for each $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to every node
- But, termination is not guaranteed
  - E.g., consider $KB = \langle \mathcal{T}, \mathcal{A} \rangle$

$$\begin{aligned} \mathcal{T} &= \{Human \sqsubseteq \exists hasMother.Human\} \\ \mathcal{A} &= \{umberto{:}Human\} \end{aligned}$$

$\mathcal{L}(umberto) = \{Human, \neg Human \sqcup \exists hasMother.Human\}$ *umberto*

KB Satisfiability

- We have seen how to test the satisfiability of an ABox $\mathcal{A}$
- But, how can we check if a KB $KB = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable?
- Basic idea: since $t(C \sqsubseteq D) \equiv \forall x.\neg t(C, x) \vee t(D, x)$
  - we use the rule: for each $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to every node
- But, termination is not guaranteed
  - E.g., consider $KB = \langle \mathcal{T}, \mathcal{A} \rangle$

$$\begin{aligned} \mathcal{T} &= \{Human \sqsubseteq \exists hasMother.Human\} \\ \mathcal{A} &= \{umberto{:}Human\} \end{aligned}$$

$\mathcal{L}(umberto) = \{Human, \neg Human \sqcup \exists hasMother.Human, \neg Human\}$ *umberto*

KB Satisfiability

- We have seen how to test the satisfiability of an ABox $\mathcal{A}$
- But, how can we check if a KB $KB = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable?
- Basic idea: since $t(C \sqsubseteq D) \equiv \forall x.\neg t(C, x) \vee t(D, x)$
  - we use the rule: for each $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to every node
- But, termination is not guaranteed
  - E.g., consider $KB = \langle \mathcal{T}, \mathcal{A} \rangle$

$$\begin{aligned} \mathcal{T} &= \{Human \sqsubseteq \exists hasMother.Human\} \\ \mathcal{A} &= \{umberto{:}Human\} \end{aligned}$$

$\mathcal{L}(umberto) = \{Human, \neg Human \sqcup \exists hasMother.Human, \neg Human\}$ *umberto*

Clash

KB Satisfiability

- We have seen how to test the satisfiability of an ABox $\mathcal{A}$
- But, how can we check if a KB $KB = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable?
- Basic idea: since $t(C \sqsubseteq D) \equiv \forall x.\neg t(C, x) \vee t(D, x)$
  - we use the rule: for each $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to every node
- But, termination is not guaranteed
  - E.g., consider $KB = \langle \mathcal{T}, \mathcal{A} \rangle$

$$\begin{aligned} \mathcal{T} &= \{Human \sqsubseteq \exists hasMother.Human\} \\ \mathcal{A} &= \{umberto{:}Human\} \end{aligned}$$

$\mathcal{L}(umberto) = \{Human, \neg Human \sqcup \exists hasMother.Human\}$ *umberto*

KB Satisfiability

- We have seen how to test the satisfiability of an ABox $\mathcal{A}$
- But, how can we check if a KB $KB = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable?
- Basic idea: since $t(C \sqsubseteq D) \equiv \forall x.\neg t(C, x) \vee t(D, x)$
  - we use the rule: for each $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to every node
- But, termination is not guaranteed
  - E.g., consider $KB = \langle \mathcal{T}, \mathcal{A} \rangle$

$$\begin{aligned} \mathcal{T} &= \{Human \sqsubseteq \exists hasMother.Human\} \\ \mathcal{A} &= \{umberto{:}Human\} \end{aligned}$$

$\mathcal{L}(umberto) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$ *umberto*

KB Satisfiability

- We have seen how to test the satisfiability of an ABox $\mathcal{A}$
- But, how can we check if a KB $KB = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable?
- Basic idea: since $t(C \sqsubseteq D) \equiv \forall x.\neg t(C, x) \vee t(D, x)$
  - we use the rule: for each $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to every node
- But, termination is not guaranteed
  - E.g., consider $KB = \langle \mathcal{T}, \mathcal{A} \rangle$

$$\begin{aligned} \mathcal{T} &= \{Human \sqsubseteq \exists hasMother.Human\} \\ \mathcal{A} &= \{umberto{:}Human\} \end{aligned}$$

$\mathcal{L}(umberto) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$ *umberto*

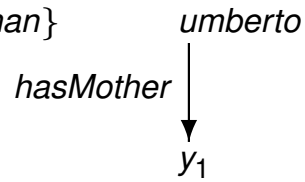
KB Satisfiability

- We have seen how to test the satisfiability of an ABox $\mathcal{A}$
- But, how can we check if a KB $KB = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable?
- Basic idea: since $t(C \sqsubseteq D) \equiv \forall x.\neg t(C, x) \vee t(D, x)$
  - we use the rule: for each $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to every node
- But, termination is not guaranteed
  - E.g., consider $KB = \langle \mathcal{T}, \mathcal{A} \rangle$

$$\begin{aligned} \mathcal{T} &= \{Human \sqsubseteq \exists hasMother.Human\} \\ \mathcal{A} &= \{umberto{:}Human\} \end{aligned}$$

$\mathcal{L}(umberto) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$

$\mathcal{L}(y_1) = \{Human, \neg Human \sqcup \exists hasMother.Human\}$



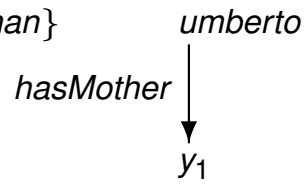
KB Satisfiability

- We have seen how to test the satisfiability of an ABox $\mathcal{A}$
- But, how can we check if a KB $KB = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable?
- Basic idea: since $t(C \sqsubseteq D) \equiv \forall x.\neg t(C, x) \vee t(D, x)$
  - we use the rule: for each $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to every node
- But, termination is not guaranteed
  - E.g., consider $KB = \langle \mathcal{T}, \mathcal{A} \rangle$

$$\begin{aligned} \mathcal{T} &= \{Human \sqsubseteq \exists hasMother.Human\} \\ \mathcal{A} &= \{umberto{:}Human\} \end{aligned}$$

$\mathcal{L}(umberto) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$

$\mathcal{L}(y_1) = \{Human, \neg Human \sqcup \exists hasMother.Human\}$



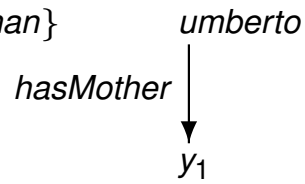
KB Satisfiability

- We have seen how to test the satisfiability of an ABox $\mathcal{A}$
- But, how can we check if a KB $KB = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable?
- Basic idea: since $t(C \sqsubseteq D) \equiv \forall x.\neg t(C, x) \vee t(D, x)$
  - we use the rule: for each $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to every node
- But, termination is not guaranteed
  - E.g., consider $KB = \langle \mathcal{T}, \mathcal{A} \rangle$

$$\begin{aligned} \mathcal{T} &= \{Human \sqsubseteq \exists hasMother.Human\} \\ \mathcal{A} &= \{umberto{:}Human\} \end{aligned}$$

$\mathcal{L}(umberto) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$

$\mathcal{L}(y_1) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$



KB Satisfiability

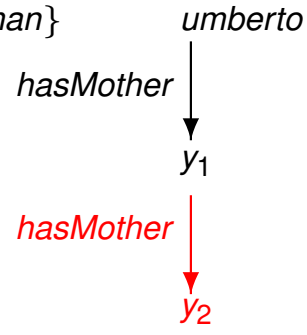
- We have seen how to test the satisfiability of an ABox $\mathcal{A}$
- But, how can we check if a KB $KB = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable?
- Basic idea: since $t(C \sqsubseteq D) \equiv \forall x.\neg t(C, x) \vee t(D, x)$
  - we use the rule: for each $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to every node
- But, termination is not guaranteed
  - E.g., consider $KB = \langle \mathcal{T}, \mathcal{A} \rangle$

$$\begin{aligned} \mathcal{T} &= \{Human \sqsubseteq \exists hasMother.Human\} \\ \mathcal{A} &= \{umberto{:}Human\} \end{aligned}$$

$\mathcal{L}(umberto) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$

$\mathcal{L}(y_1) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$

$\mathcal{L}(y_2) = \{Human, \neg Human \sqcup \exists hasMother.Human\}$



KB Satisfiability

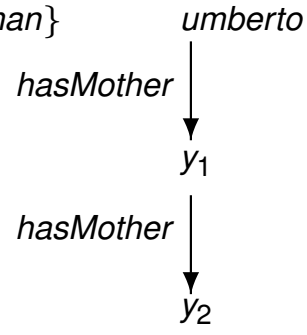
- We have seen how to test the satisfiability of an ABox $\mathcal{A}$
- But, how can we check if a KB $KB = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable?
- Basic idea: since $t(C \sqsubseteq D) \equiv \forall x.\neg t(C, x) \vee t(D, x)$
  - we use the rule: for each $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to every node
- But, termination is not guaranteed
  - E.g., consider $KB = \langle \mathcal{T}, \mathcal{A} \rangle$

$$\begin{aligned} \mathcal{T} &= \{Human \sqsubseteq \exists hasMother.Human\} \\ \mathcal{A} &= \{umberto{:}Human\} \end{aligned}$$

$\mathcal{L}(umberto) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$

$\mathcal{L}(y_1) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$

$\mathcal{L}(y_2) = \{Human, \neg Human \sqcup \exists hasMother.Human\}$



KB Satisfiability

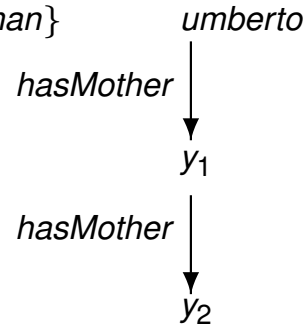
- We have seen how to test the satisfiability of an ABox $\mathcal{A}$
- But, how can we check if a KB $KB = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable?
- Basic idea: since $t(C \sqsubseteq D) \equiv \forall x.\neg t(C, x) \vee t(D, x)$
  - we use the rule: for each $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to every node
- But, termination is not guaranteed
  - E.g., consider $KB = \langle \mathcal{T}, \mathcal{A} \rangle$

$$\begin{aligned} \mathcal{T} &= \{Human \sqsubseteq \exists hasMother.Human\} \\ \mathcal{A} &= \{umberto{:}Human\} \end{aligned}$$

$\mathcal{L}(umberto) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$

$\mathcal{L}(y_1) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$

$\mathcal{L}(y_2) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$



KB Satisfiability

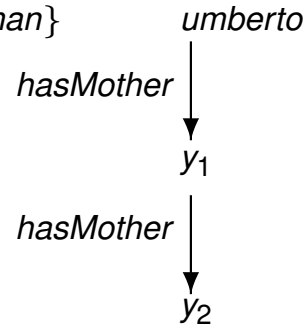
- We have seen how to test the satisfiability of an ABox $\mathcal{A}$
- But, how can we check if a KB $KB = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable?
- Basic idea: since $t(C \sqsubseteq D) \equiv \forall x.\neg t(C, x) \vee t(D, x)$
  - we use the rule: for each $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to every node
- But, termination is not guaranteed
  - E.g., consider $KB = \langle \mathcal{T}, \mathcal{A} \rangle$

$$\begin{aligned} \mathcal{T} &= \{Human \sqsubseteq \exists hasMother.Human\} \\ \mathcal{A} &= \{umberto{:}Human\} \end{aligned}$$

$\mathcal{L}(umberto) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$

$\mathcal{L}(y_1) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$

$\mathcal{L}(y_2) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$



KB Satisfiability

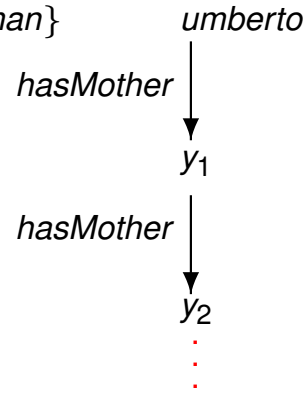
- We have seen how to test the satisfiability of an ABox $\mathcal{A}$
- But, how can we check if a KB $KB = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable?
- Basic idea: since $t(C \sqsubseteq D) \equiv \forall x.\neg t(C, x) \vee t(D, x)$
  - we use the rule: for each $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to every node
- But, termination is not guaranteed
  - E.g., consider $KB = \langle \mathcal{T}, \mathcal{A} \rangle$

$$\begin{aligned} \mathcal{T} &= \{Human \sqsubseteq \exists hasMother.Human\} \\ \mathcal{A} &= \{umberto{:}Human\} \end{aligned}$$

$\mathcal{L}(umberto) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$

$\mathcal{L}(y_1) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$

$\mathcal{L}(y_2) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$



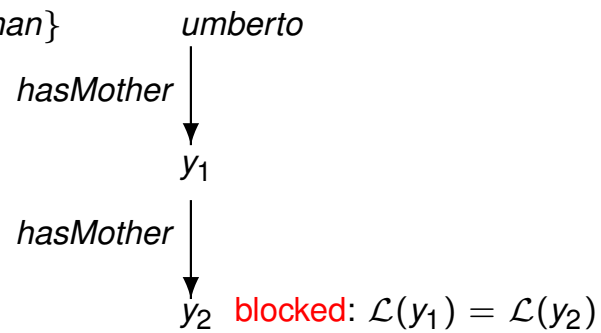
Node Blocking in $\mathcal{ALC}$

- When creating new node, check ancestors for equal label set
- If such a node is found, new node is blocked
- No rule is applied to blocked nodes

$\mathcal{L}(umberto) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$

$\mathcal{L}(y_1) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$

$\mathcal{L}(y_2) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$



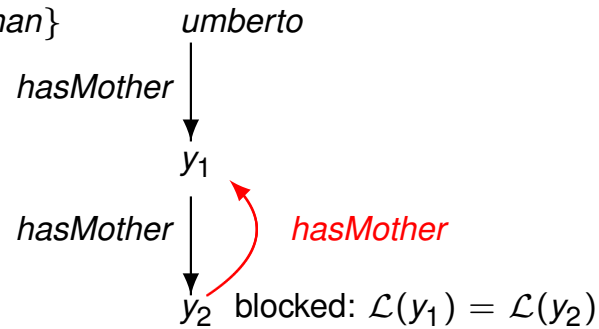
Node Blocking in $\mathcal{ALC}$

- When creating new node, check ancestors for equal label set
- If such a node is found, new node is blocked
- No rule is applied to blocked nodes

$\mathcal{L}(umberto) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$

$\mathcal{L}(y_1) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$

$\mathcal{L}(y_2) = \{Human, \neg Human \sqcup \exists hasMother.Human, \exists hasMother.Human\}$



- Block represents cyclical model
  - $\Delta^{\mathcal{I}} = \{umberto, y_1, y_2\}$
  - $Human^{\mathcal{I}} = \{umberto, y_1, y_2\}$
  - $hasMother^{\mathcal{I}} = \{\langle umberto, y_1\rangle, \langle y_1, y_2\rangle, \langle y_2, y_1\rangle\}$

Blocking in $\mathcal{ALC}$

- A non-root node x is blocked if for some ancestor y, y is blocked or $\mathcal{L}(x) = \mathcal{L}(y)$, where y is not a root node.
- A blocked node x is indirectly blocked if its predecessor is blocked, otherwise it is directly blocked.
- If x is directly blocked, it has a unique ancestor y such that $\mathcal{L}(x) = \mathcal{L}(y)$
- if there existed another ancestor z such that $\mathcal{L}(x) = \mathcal{L}(z)$ then either y or z must be blocked.
- If x is directly blocked and y is the unique ancestor such that $\mathcal{L}(x) = \mathcal{L}(y)$, we will say that y blocks x

$\mathcal{ALC}$ Tableau rules with GCI's

Rule		Description
$(\sqcap)$	if 1.	$C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not indirectly blocked and
	2.	$\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$
	then	$\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
$(\sqcup)$	if 1.	$C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not indirectly blocked and
	2.	$\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
	then	$\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$
$(\exists)$	if 1.	$\exists R.C \in \mathcal{L}(x)$, x is not blocked 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.	$\forall R.C \in \mathcal{L}(x)$, x is not indirectly blocked and
	2.	x has an R-successor y with $C \notin \mathcal{L}(y)$
	then	$\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{C\}$
$(\sqsubseteq)$	if 1.	$C \sqsubseteq D \in \mathcal{T}$, x is not indirectly blocked and
	2.	$\{nnf(\neg C), D\} \cap \mathcal{L}(x) = \emptyset$
	then	$\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{E\}$ for some
		$E \in \{nnf(\neg C), D\}$ ($nnf(\neg C)$ is NNF of $\neg C$)

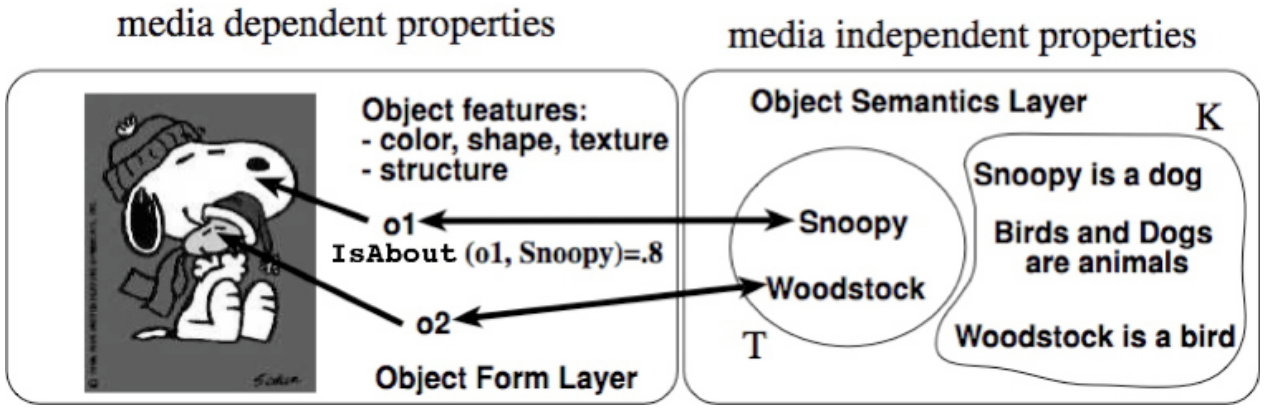
Soundness and Completeness

Theorem

Let KB be an $\mathcal{ALC}$ KB and F a completion-forest obtained by applying the tableau rules to KB. Then

1. *The rule application terminates;*
2. *If F is clash-free and complete, then F defines a (canonical) (tree) model for KB; and*
3. *If KB has a model $\mathcal{I}$, then the rules can be applied such that they yield a clash-free and complete completion-forest.*

Representing degrees in OWL-DL/OWL-Lite



- How can we represent degrees of uncertainty and vagueness in OWL-DL/OWL-Lite?
- Unfortunately, as for RDF, no standard exists yet
- We may make an encoding as for RDF

$$s1{:}\exists \mathit{hasSubject}.(\{o1\} \sqcap \exists \mathit{IsAbout}.\{\mathit{snoopy}\}) \sqcap \exists \mathit{hasDegree}.\ =_{0.8}$$

- But, again then such statements have to be appropriately be managed by the system according to the underlying uncertainty or vagueness theory
- A rather dangerous approach