# Contents

# What else is there in SICStus Prolog?

- Further constraint libraries:
  - CLPB – booleans
  - CLPQ/CLPR – linear inequalities on rationals/reals
  - Constraint Handling Rules: generic constraints
- Other features
  - “Traditional” built-in predicates, e.g. sorting, input/output, exception handling, etc.
  - Powerful data structures, e.g. AVL trees, multisets, heaps, graphs, etc.
  - Definite clause grammars, an extension of context-free grammars with Prolog terms
  - Interfaces to other programming languages, e.g. C/C++, Java, .NET, Tcl/Tk
  - Integrated development environment based on Eclipse (Spider)
  - Execution profiling
  - ...

# Some applications of (constraint) logic programming

- Boeing Corp.: Connector Assembly Specifications Expert (CASEy) – an expert system that guides shop floor personnel in the correct usage of electrical process specifications.
- Windows NT: `\WINNT\SYSTEM32\NETCFG.DLL` contains a small Prolog interpreter handling the rules for network configuration.
- Experian (one of the largest credit rating companies): Prolog for checking credit scores. Experian bought Prologia, the Marseille Prolog company.
- IBM bought ILOG, the developer of many constraint algorithms (e.g. that in `all_distinct`); ILOG develops a constraint programming / optimization framework embedded in C++.
- IBM uses Prolog in the Watson deep Question-Answer system for parsing and matching English text

# Part IV

## The Semantic Web

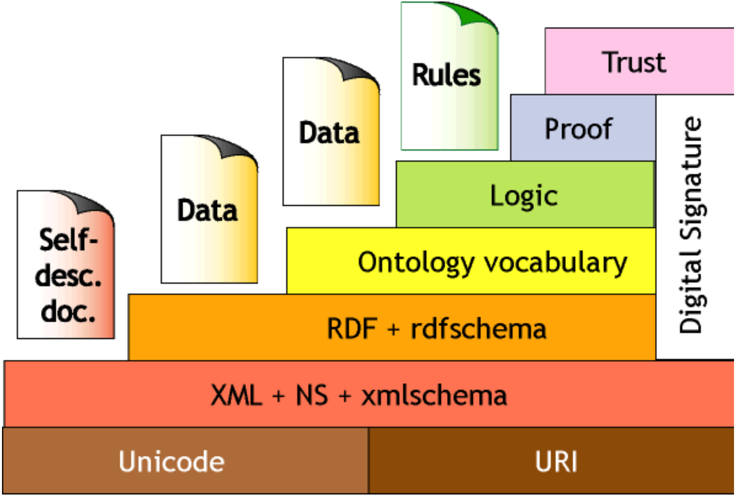
# Contents

# Semantic Technologies

- Semantics = meaning
- Semantic Technologies = technologies building on (formalized) meaning
- Declarative Programming as a semantic technology
  - A procedure definition describes its intended meaning
    - e.g. `intersect(L1, L2) :- member(X, L1), member(X, L2).`
      Lists `L1` and `L2` intersect if
      there exists an `X`, such that `X` is a member of both `L1` and `L2`.
  - The execution of a program can be viewed as a process of deduction
- The main goal of the Semantic Web (SW) approach:
  - make the information on the web processable by computers
  - machines should be able to understand the web, not only read it
- Achieving the vision of the Semantic Web
  - Add (computer processable) meta-information to the web
  - Formalize background knowledge – build so called ontologies
  - Develop reasoning algorithms and tools

# The vision of the Semantic Web

- The Semantic Web layer cake – Tim Berners-Lee



# The Semantic Web

- The goal: making the information on the web processable by computers
- Achieving the vision of the Semantic Web
  - Add meta-information to web pages, e.g.
    (*AIT* `hasLocation` *Budapest*)
    (*AIT* `hasTrack` *Track:Foundational-courses*)
    (*Track:Foundational-courses* `hasCourse` *Semantic-and-declarative...*)
  - Formalise background knowledge – build so called terminologies
    - hierarchies of notions, e.g.
      a *University* is a (subconcept of) *Inst-of-higher-education*,
      the `hasFather` relationship is a special case of `hasParent`
    - definitions and axioms, e.g.
      a *Father* is a *Male Person* `having` at least one `child`
  - Develop reasoning algorithms and tools
- Main topics
  - Description Logic, the maths behind the Semantic Web is the basis of Web Ontology Languages OWL 1 & 2 (W3C standards)
  - A glimpse at reasoning algorithms for Description Logic

# Contents

# First Order Logic

- Syntax:
  - non-logical (“user-defined”) symbols: predicates and functions, including constants (function symbols with 0 arguments)
  - terms (refer to individual elements of the universe, or interpretation), e.g. *fatherOf*(*Susan*)
  - formulas (that hold or do not hold in a given interpretation), e.g. $\varphi = \forall x.(\mathit{Optimist}(\mathit{fatherOf}(x)) \rightarrow \mathit{Optimist}(x))$
- Semantics:
  - determines if a closed formula $\varphi$ is true in an interpretation $\mathcal{I}$: $\mathcal{I} \models \varphi$ (also read as: $\mathcal{I}$ is a model of $\varphi$)
  - an interpretation $\mathcal{I}$ consists of a domain $\Delta$ and a mapping from non-logical symbols (e.g. *Optimist*, *fatherOf*, *Susan*) to their meaning
  - semantic consequence: $S \models \alpha$ means: if an interpretation is a model of all formulas in the set $S$, then it is also a model of $\alpha$ (note that the symbol $\models$ is overloaded)
- Deductive system (also called proof procedure): an algorithm to deduce a consequence $\alpha$ of a set of formulas $S$: $S \vdash \alpha$
  - example: resolution

# Soundness, completeness and decidability

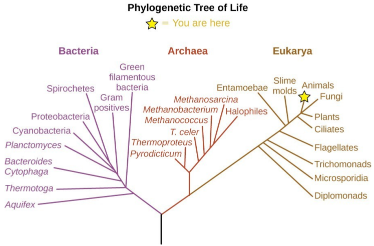
- Let $\alpha$ denote a single FOL statement, and $S$ a set of statements
- A deductive system is **sound** if $S \vdash \alpha$ implies $S \models \alpha$ (deduces only truths).
- A deductive system is **complete** if $S \models \alpha$ implies $S \vdash \alpha$ (deduces all truths).
- Kurt Gödel's original completeness theorem states that, given soundness, "$\models \implies \vdash$" holds: if $\alpha$ is true in all interpretations that satisfy $S$, i.e. if for all interpretations $\mathcal{I}$ s.t. $\mathcal{I} \models S$, $\mathcal{I} \models \alpha$ also holds, then $S \vdash \alpha$, i.e. $\alpha$ can be deduced from $S$
- The inverse statement "$\vdash \implies \models$" is trivially true, resulting in "$\models \equiv \vdash$", cf.

  **Association for Logic Programming**

- FOL is not decidable: there is no decision procedure for the question "does $S$ imply $\alpha$ ($S \vdash \alpha$)?" (Gödel's completeness theorem ensures that if the answer is "yes", then there exists a proof of $\alpha$ from $S$; but if the answer is "no", we have no guarantees – this is called semi-decidability)
- Developers of the Semantic Web strive for using decidable languages, i.e. for languages with a sound and complete proof procedure
- Semantic Web languages are based on Description Logics, which are decidable sublanguages of FOL, i.e. there is an algorithm that delivers a yes or no answer to the question "does $S$ imply $\alpha$?"

# Ontologies

- **Ontology**: computer processable description of knowledge
- Early ontologies include classification system (biology, medicine, books)




- Entities in the Web Ontology Language (OWL):
  - classes – describe sets of objects (e.g. optimists)
  - properties (attributes, slots) – describe binary relationships (e.g. has parent)
  - objects – correspond to real life objects (e.g. people, such as Susan, her parents, etc.)

# Knowledge Representation

- **Natural Language**:
  1. Someone having a non-optimist friend is bound to be an optimist.
  2. Susan has herself as a friend.
- **First order Logic** (unary predicate, binary predicate, constant):
  1. $\forall x.(\exists y.(\text{hasFriend}(x, y) \wedge \neg\text{opt}(y)) \rightarrow \text{opt}(x))$
  2. $\text{hasFriend}(\text{Susan}, \text{Susan})$
- **Description Logics** (concept, role, individual):
  1. $(\exists\text{hasFriend}.\neg\,\text{Opt}) \sqsubseteq \text{Opt}$ (GCI – Gen. Concept Inclusion axiom)
  2. $\text{hasFriend}(\text{Susan}, \text{Susan})$ (role assertion)
- **Web Ontology Language** (Manchester syntax)[5] (class, property, object):
  1. `(hasFriend some (not Opt)) SubClassOf: Opt`
     Those having some not Opt friends must be Opt
     (GCI – Gen. Class Inclusion axiom)
  2. `hasFriend(Susan,Susan)` (object property assertion)

[5] protegeproject.github.io/protege/class-expression-syntax

# A sample ontology to be entered into Protégé

1. There is a class of Animals, some of which are Male, some are Female.
2. No one can be both Male and Female.
3. Every Human is an Animal.
4. Every Optimist is a Human.
5. There is a relationship hasP meaning “has parent”. Relations hasFather and hasMother are sub-relations (special cases) of hasP.
6. Let’s define the class C1 as those who have an optimistic parent.
7. State that everyone belonging to C1 is Optimistic.
8. State directly that anyone having an Optimistic parent is Optimistic.
9. There is a relation hasF, denoting “has friend”. State that someone having a non-Optimistic friend must be Optimistic.
10. There are individuals: Susan, and her parents Mother and Father.
11. Mother has Father as her friend.

# The sample ontology in Description Logic and OWL/Protégé

| | English | Description Logic | OWL (Manchester syntax) |
|---|---|---|---|
| 1 | Male is a subclass of Animal. | Male $\sqsubseteq$ Animal | Male `SubClassOf:` Animal |
| | Female is a subclass of Animal. | Female $\sqsubseteq$ Animal | Female `SubClassOf:` Animal |
| 2 | Male and Female are disjoint. | Male $\sqsubseteq$ $\neg$ Female | Male `DisjointWith:` Female |
| 3 | Human is a subclass of Animal. | Human $\sqsubseteq$ Animal | Human `SubClassOf:` Animal |
| 4 | Optimist is a subclass of Human. | Opt $\sqsubseteq$ Human | Opt `SubClassOf:` Human |
| 5 | hasFather is a subprop. of hasP. | hasFather $\sqsubseteq$ hasP | hasFather `SubPropertyOf:` hasP |
| | hasMother is a subprop. of hasP. | hasMother $\sqsubseteq$ hasP | hasMother `SubPropertyOf:` hasP |
| 6 | C1 = those having an Opt parent. | C1 $\equiv$ $\exists$ hasP . Opt | C1 `EquivalentTo:` hasP `some` Opt |
| 7 | Everyone in C1 is Opt. | C1 $\sqsubseteq$ Opt | C1 `SubClassOf:` Opt |
| 8 | Children of Opt parents are Opt. | $\exists$ hasP . Opt $\sqsubseteq$ Opt | hasP `some` Opt `SubClassOf:` Opt |
| 9 | Those with a non-Opt friend are Opt. | $\exists$ hasF . $\neg$Opt $\sqsubseteq$ Opt | hasF `some not` Opt `SubClassOf:` Opt |
| 10 | Susan has parents Mother and Father. | hasP(Susan, Mother)<br>hasP(Susan, Father) | hasP(Susan, Mother)<br>hasP(Susan, Father) |
| 11 | Mother has Father as a friend. | hasF(Mother, Father) | hasF(Mother, Father) |

(In Protégé, select the “save as” format as “Latex syntax” to obtain DL notation.)

# Contents

# Description Logic (DLs) – overview

DL is a subset of FOL, providing the maths background of OWL

- Signature – relation and function symbols allowed in DL
  - concept name ($A$) – unary predicate symbol (cf. OWL class)
  - role name ($R$) – binary predicate symbol (cf. OWL property)
  - individual name ($a, \ldots$) – constant symbol (cf. OWL object)
  - No non-constant function symbols, no preds of arity $> 2$, no vars
- Concept names and concept expressions represent sets, e.g. ∃hasParent.Optimist – the set of those who have an optimist parent
- Terminological axioms (TBox) state background knowledge
  - A simple axiom using the DL language $\mathcal{ALE}$: ∃hasParent.Optimist ⊑ Optimist – the set of those who have an optimist parent is a subset of the set of optimists
  - Translation to FOL: $\forall x.(\exists y.(hasP(x, y) \wedge Opt(y)) \rightarrow Opt(x))$
- Assertions (ABox) state facts about individual names
  - Example: Optimist(JACOB), hasParent(JOSEPH, JACOB)
- A consequence of these TBox and ABox axioms is: Optimist(JOSEPH)
- DLs behind OWL 1 and OWL 2 are decidable: there are bounded time algorithms for checking if a set of axioms implies a statement.

# Some further examples of terminological axioms

(1) A Mother is a Person, who is a Female and who has(a)Child.

$$\text{Mother} \equiv \text{Person} \sqcap \text{Female} \sqcap \exists \text{hasChild}.\top$$

(2) A Tiger is a Mammal.

$$\text{Tiger} \sqsubseteq \text{Mammal}$$

(3) All children of an Optimist are Optimists, too.

$$\text{Optimist} \sqsubseteq \forall \text{hasChild}.\text{Optimist}$$

(alternatively:) $\exists \text{hasParent}.\text{Opt} \sqsubseteq \text{Opt}$

(4) Childless people are Happy.

$$\forall \text{hasChild}.\bot \sqcap \text{Person} \sqsubseteq \text{Happy}$$

(5) Those in the relation hasChild are also in the relation hasDescendant.

$$\text{hasChild} \sqsubseteq \text{hasDescendant}$$

(6) The relation hasParent is the inverse of the relation hasChild.

$$\text{hasParent} \equiv \text{hasChild}^{-}$$

(7) The hasDescendant relationship is transitive.

$$\textbf{Trans}(\text{hasDescendant})$$

# Description Logics – why the plural?

- These logic variants were progressively developed in the last two decades
- As new constructs were proved to be “safe”, i.e. keeping the logic decidable, these were added
- We will start with the very simple language $\mathcal{AL}$, extend it to $\mathcal{ALE}$, $\mathcal{ALU}$ and $\mathcal{ALC}$
- As a side branch we then define $\mathcal{ALCN}$
- We then go back to $\mathcal{ALC}$ and extend it to languages $\mathcal{S}$, $\mathcal{SH}$, $\mathcal{SHI}$ and $\mathcal{SHIQ}$ (which encompasses $\mathcal{ALCN}$)
- We briefly tackle further extensions $\mathcal{O}$, (**D**) and $\mathcal{R}$
- OWL 1, published in 2004, corresponds to $\mathcal{SHOIN}$(**D**)
- OWL 2, published in 2012, corresponds to $\mathcal{SROIQ}$(**D**)

# Contents

# Overview of the $\mathcal{ALCN}$ language

- In $\mathcal{ALCN}$ a statement (axiom) can be
  - a subsumption (inclusion), e.g. Tiger $\sqsubseteq$ Mammal, or
  - an equivalence, e.g. Woman $\equiv$ Female $\sqcap$ Person, Mother $\equiv$ Woman $\sqcap$ $\exists$hasChild.$\top$
- In general, an $\mathcal{ALCN}$ axiom can take these two forms:
  - subsumption: $C \sqsubseteq D$
  - equivalence: $C \equiv D$, where $C$ and $D$ are concept expressions
- A concept expression $C$ denotes a set of objects (a subset of the $\Delta$ universe of the interpretation), and can be:
  - an atomic concept (or concept name), e.g. Tiger, Female, Person
  - a composite concept, e.g. Female $\sqcap$ Person, $\exists$hasChild.Female
  - composite concepts are built from atomic concepts and atomic roles (also called role names) using some constructors (e.g. $\sqcap$, $\sqcup$, $\exists$, etc.)
- We first introduce language $\mathcal{AL}$, that allows a minimal set of constructors (all examples on this page are valid $\mathcal{AL}$ concept expressions)
- Next, we discuss richer extensions named $\mathcal{U}$, $\mathcal{E}$, $\mathcal{C}$, $\mathcal{N}$

# The syntax of the $\mathcal{AL}$ language

Language $\mathcal{AL}$ (Attributive Language) allows the following concept expressions, also called concepts, for short:

$A$ is an atomic concept, $C$, $D$ are arbitrary (possibly composite) concepts
$R$ is an atomic role

| DL concept | OWL class | Name | Informal definition |
|---|---|---|---|
| $A$ | $A$ (class name) | atomic concept | those in $A$ |
| $\top$ | `owl:Thing` | top | the set of all objects |
| $\bot$ | `owl:Nothing` | bottom | the empty set |
| $\neg A$ | `not` $A$ | atomic negation | those not in $A$ |
| $C \sqcap D$ | $C$ `and` $D$ | intersection | those in both $C$ and $D$ |
| $\forall R.C$ | $R$ `only` $C$ | value restriction | those whose all $R$s belong to $C$ |
| $\exists R.\top$ | $R$ `some owl:Thing` | limited exist. restr. | those having at least one $R$ |

Examples of $\mathcal{AL}$ concept expressions:

| | |
|---|---|
| Person $\sqcap$ $\neg$Female | Person `and not` Female |
| Person $\sqcap$ $\forall$hasChild.Female | Person `and (`hasChild `only` Female`)` |
| Person $\sqcap$ $\exists$hasChild.$\top$ | Person `and (`hasChild `some owl:Thing)` |

# The semantics of the $\mathcal{AL}$ language (as a special case of FOL)

- An interpretation $\mathcal{I}$ is a mapping:
  - $\Delta^\mathcal{I} = \Delta$ is the universe, the **nonempty** set of all individuals/objects
  - for each concept/class name $A$, $A^\mathcal{I}$ is a (possibly empty) subset of $\Delta$
  - for each role/property name $R$, $R^\mathcal{I} \subseteq \Delta \times \Delta$ is a binary relation on $\Delta$
- The semantics of $\mathcal{AL}$ extends $\mathcal{I}$ to composite concept expressions, i.e. describes how to "calculate" the meaning of arbitrary concept exprs:

$$
\begin{aligned}
\top^\mathcal{I} &= \Delta \\
\bot^\mathcal{I} &= \emptyset \\
(\neg A)^\mathcal{I} &= \Delta \setminus A^\mathcal{I} \\
(C \sqcap D)^\mathcal{I} &= C^\mathcal{I} \cap D^\mathcal{I} \\
(\forall R.C)^\mathcal{I} &= \{a \in \Delta | \forall b.(\langle a, b \rangle \in R^\mathcal{I} \to b \in C^\mathcal{I})\} \\
(\exists R.\top)^\mathcal{I} &= \{a \in \Delta | \exists b.\langle a, b \rangle \in R^\mathcal{I}\}
\end{aligned}
$$

- Finally we define how to obtain the truth value of an axiom:

$$
\begin{aligned}
\mathcal{I} \models C \sqsubseteq D \quad &\text{iff} \quad C^\mathcal{I} \subseteq D^\mathcal{I} \\
\mathcal{I} \models C \equiv D \quad &\text{iff} \quad C^\mathcal{I} = D^\mathcal{I}
\end{aligned}
$$

# A simple $\mathcal{AL}$ example

- An example TBox $\mathcal{T}$

$$\{$$
$$\text{FatherOfGirls} \equiv \text{Person} \sqcap \neg\text{Woman} \sqcap \forall\text{hasChild}.\text{Woman} \sqcap \exists\text{hasChild}.\top,$$
$$\text{FatherOfGirls} \sqsubseteq \text{Happy}$$
$$\}$$

- The First Order Logic (FOL) equivalent of the above TBox:

$$\forall x.\,(\text{FatherOfGirls}(x) \leftrightarrow \text{Person}(x) \land \neg\text{Woman}(x) \land \forall y.\,(\text{hasChild}(x,y) \rightarrow \text{Woman}(y)) \land \exists y.\text{hasChild}(x,y)) \land$$
$$\forall x.\,(\text{FatherOfGirls}(x) \rightarrow \text{Happy}(x))$$

# The $\mathcal{AL}$ language: limitations

## Recall the elements of the language $\mathcal{AL}$:

| DL concept | OWL class | Name | Informal definition |
|---|---|---|---|
| $A$ | $A$ (class name) | atomic concept | those in $A$ |
| $\top$ | owl:Thing | top | the set of all objects |
| $\bot$ | owl:Nothing | bottom | the empty set |
| $\neg A$ | not $A$ | atomic negation | those not in $A$ |
| $C \sqcap D$ | $C$ and $D$ | intersection | those in both $C$ and $D$ |
| $\forall R.C$ | $R$ only $C$ | value restriction | those whose all $R$s belong to $C$ |
| $\exists R.\top$ | $R$ some owl:Thing | limited exist. restr. | those having at least one $R$ |

## What is missing from $\mathcal{AL}$?

- We can specify the intersection of two concepts, but not the union, e.g. those who are either blue-eyed or tall.
- $\exists R.\top$ – we cannot describe e.g. those having a female child.
  Remedy: allow for full exist. restr., e.g. $\exists hasCh.Female$
- $\neg A$ – negation can be applied to atomic concepts only.
  Remedy: full negation, $\neg C$, where C can be non-atomic, e.g. $\neg(U \sqcap V)$

# The $\mathcal{ALCN}$ language family: extensions $\mathcal{U}$, $\mathcal{E}$, $\mathcal{C}$, $\mathcal{N}$

Further concept constructors, OWL equivalents shown in [`square brackets`]:

- Union: $C \sqcup D$, [$C$ `or` $D$] – those in either $C$ or $D$
$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}} \qquad (\mathcal{U})$$
- Full existential restriction: $\exists R.C$, [$R$ `some` $C$]
– those who have at least one $R$ belonging to $C$
$$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} | \exists b.\langle a, b \rangle \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \qquad (\mathcal{E})$$
- (Full) negation: $\neg C$, [`not` $C$] – those who do not belong to $C$
$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \qquad (\mathcal{C})$$
- Unqualified number restrictions: $(\leqslant n R)$, [$R$ `max` $n$ `owl:Thing`] and
$(\geqslant n R)$, [$R$ `min` $n$ `owl:Thing`]
– those who have at most/at least $n$ $R$-related objects
$$(\leqslant n\, R)^{\mathcal{I}} = \{\, a \in \Delta^{\mathcal{I}} \mid \; |\{b \mid \langle a, b \rangle \in R^{\mathcal{I}}\}| \le n \,\}$$
$$(\geqslant n\, R)^{\mathcal{I}} = \{\, a \in \Delta^{\mathcal{I}} \mid \; |\{b \mid \langle a, b \rangle \in R^{\mathcal{I}}\}| \ge n \,\} \qquad (\mathcal{N})$$

Example: $\text{Person} \sqcap ((\leqslant 1\, \text{hasCh}) \sqcup (\geqslant 3\, \text{hasCh})) \sqcap \exists \text{hasCh}.\text{Female}$

Person `and (`hasCh `max 1 or` hasCh `min 3) and (`hasCh `some` Female`)`

Note that qualified number restrictions, e.g., “those having at least 3 blue-eyed children” are not covered by the extension $\mathcal{N}$.

# Summary table of the $\mathcal{ALCUEN}$ language

| DL | OWL | Name | Informal definition | |
|---|---|---|---|---|
| $A$ | $A$ | atomic concept | those in $A$ | $\mathcal{AL}$ |
| $\neg A$ | not $A$ | atomic negation | those not in $A$ (cf. $\mathcal{C}$) | $\mathcal{AL}$ |
| $\top$ | owl:Thing | top | the set of all objects | $\mathcal{AL}$ |
| $\bot$ | owl:Nothing | bottom | the empty set | $\mathcal{AL}$ |
| $C \sqcap D$ | $C$ and $D$ | intersection | those in both $C$ and $D$ | $\mathcal{AL}$ |
| $\exists R.\top$ | $R$ some | existential restr. | those having an $R$ (cf. $\mathcal{E}$) | $\mathcal{AL}$ |
| $\forall R.C$ | $R$ only $C$ | value restriction | those whose all $R$s belong to $C$ | $\mathcal{AL}$ |
| $\neg C$ | not $C$ | full negation | those not in $C$ | $\mathcal{C}$ |
| $C \sqcup D$ | $C$ or $D$ | union | those in either $C$ or $D$ | $\mathcal{U}$ |
| $\exists R.C$ | $R$ some $C$ | existential restr. | those with an $R$ belonging to $C$ | $\mathcal{E}$ |
| $(\leqslant nR)$ | $R$ max $n$ o:T | unq. numb. restr. | those having at most $n$ $R$s | $\mathcal{N}$ |
| $(\geqslant nR)$ | $R$ min $n$ o:T | unq. numb. restr. | those having at least $n$ $R$s | $\mathcal{N}$ |

# Rewriting $\mathcal{ALCN}$ to first order logic

- Concept expressions map to predicates with one argument, e.g.
  Tiger $\Longrightarrow$ Tiger$(x)$ Mammal $\Longrightarrow$ Mammal$(x)$
  Person $\Longrightarrow$ Person$(x)$ Female $\Longrightarrow$ Female$(x)$
- Simple connectives $\sqcap$, $\sqcup$, $\neg$ map to boolean operations $\wedge$, $\vee$, $\neg$, e.g.
  Person $\sqcap$ Female $\Longrightarrow$ Person$(x) \wedge$ *Female*$(x)$
  Person $\sqcup \neg$Mammal $\Longrightarrow$ Person$(x) \vee \neg$Mammal$(x)$
- An axiom $C \sqsubseteq D$ is rewritten as $\forall x.(C(x) \rightarrow D(x))$, e.g.
  Tiger $\sqsubseteq$ Mammal $\Longrightarrow \forall x.($Tiger$(x) \rightarrow$ Mammal$(x))$
- An axiom $C \equiv D$ is rewritten as $\forall x.(C(x) \leftrightarrow D(x))$, e.g.
  Woman $\equiv$ Person $\sqcap$ Female $\Longrightarrow \forall x.(Woman(x) \leftrightarrow Person(x) \wedge Female(x))$
- Concept constructors involving a quantifier $\exists$ or $\forall$ are rewritten to an appropriate quantified formula, where a role name is mapped to a binary predicate (a predicate with two arguments), e.g.

  $\exists$hasParent.Opt$\sqsubseteq$Opt $\Longrightarrow \forall x.(\exists y.(\text{hasParent}(x, y) \wedge \text{Opt}(y)) \rightarrow \text{Opt}(x))$

# Rewriting $\mathcal{ALCN}$ to first order logic, example

- Consider $C = \text{Person} \sqcap ((\leqslant 1\, \text{hasCh}) \sqcup (\geqslant 3\, \text{hasCh})) \sqcap \exists \text{hasCh}.\text{Female}$
- Let's outline a predicate $C(x)$ which is true when $x$ belongs to concept $C$:

$$C(x) \leftrightarrow \quad Person(x) \wedge \\ (hasAtMost1Child(x) \vee hasAtLeast3Children(x)) \wedge \\ hasFemaleChild(x)$$

- Class practice:
  - Define the FOL predicates $hasAtMost1Child(x)$, $hasAtLeast3Children(x)$, $hasFemaleChild(x)$
  - Additionally, define the following FOL predicates:
    - $hasOnlyFemaleChildren(x)$, corresponding to the concept $\forall \text{hasCh}.\text{Female}$
    - $hasAtMost2Children(x)$, corresponding to the concept $(\leqslant 2\, \text{hasCh})$

# General rewrite rules $\mathcal{ALCN} \rightarrow$ FOL

Each concept expression can be mapped to a FOL formula:

- Each concept expression $C$ is mapped to a formula $\Phi_C(x)$ (expressing that $x$ belongs to $C$).
- Atomic concepts ($A$) and roles ($R$) are mapped to unary and binary predicates $A(x)$, $R(x, y)$.
- $\sqcap$, $\sqcup$, and $\neg$ are transformed to their counterpart in FOL ($\wedge, \vee, \neg$), e.g. $\Phi_{C \sqcap D}(x) = \Phi_C(x) \wedge \Phi_D(x)$
- Mapping further concept constructors:

$$\begin{aligned}
\Phi_{\exists R.C}(x) &= \exists y. (R(x, y) \wedge \Phi_C(y)) \\
\Phi_{\forall R.C}(x) &= \forall y. (R(x, y) \rightarrow \Phi_C(y)) \\
\Phi_{\geqslant n\, R}(x) &= \exists y_1, \ldots, y_n. \left( R(x, y_1) \wedge \cdots \wedge R(x, y_n) \wedge \bigwedge_{i<j} y_i \neq y_j \right) \\
\Phi_{\leqslant n\, R}(x) &= \forall y_1, \ldots, y_{n+1}. \left( R(x, y_1) \wedge \cdots \wedge R(x, y_{n+1}) \rightarrow \bigvee_{i<j} y_i = y_j \right)
\end{aligned}$$

# Equivalent languages in the $\mathcal{ALCN}$ family

- Language $\mathcal{AL}$ can be extended by arbitrarily choosing whether to add each of $\mathcal{UECN}$, resulting in $\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{C}][\mathcal{N}]$.
  Do these $2^4 = 16$ languages have different expressive power?
  Two concept expressions are said to be equivalent, if they have the same meaning, in all interpretations.
  Languages $\mathcal{L}_1$ and $\mathcal{L}_2$ have the same expressive power ($\mathcal{L}_1 \stackrel{e}{=} \mathcal{L}_2$), if any expression of $\mathcal{L}_1$ can be mapped into an equivalent expression of $\mathcal{L}_2$, and vice versa.
- As a preparation for discussing the above let us recall that these axioms hold in all models, for arbitrary concepts $C$ and $D$ and role $R$:

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

$$\neg\neg C \equiv C$$
$$\neg\top \equiv \bot$$
$$\neg\bot \equiv \top$$
$$\neg(C \sqcap D) \equiv \neg C \sqcup \neg D$$
$$\neg\exists R.\top \equiv \forall R.\bot$$
$$\neg\forall R.C \equiv \exists R.\neg C$$

# Equivalent languages in the $\mathcal{ALCN}$ family

Let us show that $\mathcal{ALUE}$ and $\mathcal{ALC}$ are equivalent:

- As $C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$ and $\exists R.C \equiv \neg\forall R.\neg C$, union and full existential restriction can be eliminated by using (full) negation. That is, to each $\mathcal{ALUE}$ concept expression there exists an equivalent $\mathcal{ALC}$ expression.
- The other way, each $\mathcal{ALC}$ concept can be transformed to an equivalent $\mathcal{ALUE}$ expression, by moving negation inwards, until before atomic concepts, and removing double negation; using the axioms from the right hand column on the previous slide
- Thus $\mathcal{ALUE}$ and $\mathcal{ALC}$ have the same expressive power, and so have the intermediate languages:
  $\mathcal{ALC}(\mathcal{N}) \stackrel{e}{=} \mathcal{ALCU}(\mathcal{N}) \stackrel{e}{=} \mathcal{ALCE}(\mathcal{N}) \stackrel{e}{=} \mathcal{ALCUE}(\mathcal{N}) \stackrel{e}{=} \mathcal{ALUE}(\mathcal{N})$.

Further remarks:

- As $\mathcal{U}$ and $\mathcal{E}$ is subsumed by $\mathcal{C}$, we will use $\mathcal{ALC}$ to denote the language allowing $\mathcal{U}$, $\mathcal{E}$ and $\mathcal{C}$
- It can be shown that any two of
  $\mathcal{AL}, \mathcal{ALU}, \mathcal{ALE}, \mathcal{ALC}, \mathcal{ALN}, \mathcal{ALUN}, \mathcal{ALEN}, \mathcal{ALCN}$ have different expressive power

# Another $\mathcal{ALC}$ example requiring case analysis

- Some facts about the Oedipus family (ABox $\mathcal{A}_{OE}$):

  hasChild(IOCASTE, OEDIPUS)
  hasChild(IOCASTE, POLYNEIKES)
  hasChild(OEDIPUS, POLYNEIKES)
  hasChild(POLYNEIKES, THERSANDROS)

  Patricide(OEDIPUS)

  (¬Patricide)(THERSANDROS)

- Let us call a person "special" if they have a child who is a patricide and who, in turn, has a child who is not a patricide:

  $$\text{Special} \equiv \exists \text{hasChild}.(\text{Patricide} \sqcap \exists \text{hasChild}.\neg\text{Patricide})$$

- Let TBox $\mathcal{T}_{OE}$ contain the above axiom only.
- Consider the instance check "Is Iocaste special?":
  $\mathcal{A}_{OE} \models_{\mathcal{T}_{OE}} \text{Special}(\text{IOCASTE})$?
- The answer is "yes", but proving this requires case analysis

# Contents

# A special case of ontology: definitional TBox

- $\mathcal{T}_{fam}$: a sample definitional TBox for family relationships

$$
\begin{aligned}
\text{Woman} &\equiv \text{Person} \sqcap \text{Female} \\
\text{Man} &\equiv \text{Person} \sqcap \neg\text{Woman} \\
\text{Mother} &\equiv \text{Woman} \sqcap \exists\text{hasChild}.\text{Person} \\
\text{Father} &\equiv \text{Man} \sqcap \exists\text{hasChild}.\text{Person} \\
\text{Parent} &\equiv \text{Father} \sqcup \text{Mother} \\
\text{Grandmother} &\equiv \text{Woman} \sqcap \exists\text{hasChild}.\text{Parent}
\end{aligned}
$$

- A TBox is definitional if it contains equivalence axioms only, where the left hand sides are distinct concept names (atomic concepts)
- The concepts on the left hand sides are called name symbols
- The remaining atomic concepts are called base symbols, e.g. in our example the two base symbols are Person and Female.
- In a definitional TBox the meanings of name symbols can be obtained by evaluating the right hand side of their definition

# Interpretations and semantic consequence

Recall the definition of assigning a truth value to TBox axioms in an interpretation $\mathcal{I}$:

$$\mathcal{I} \models C \sqsubseteq D \quad \text{iff} \quad C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$$
$$\mathcal{I} \models C \equiv D \quad \text{iff} \quad C^{\mathcal{I}} = D^{\mathcal{I}}$$

Based on this we introduce the notion of "semantic consequence" exactly in the same way as for FOL

- We can naturally extend the above $\mathcal{I} \models \alpha$ notation – where $\alpha$ is an axiom of the form $C \sqsubseteq D$ or $C \equiv D$ – to a TBox (i.e. a set of $\alpha$ axioms) $\mathcal{T}$
  - $\mathcal{I} \models \mathcal{T}$ ( $\mathcal{I}$ satisfies $\mathcal{T}$, $\mathcal{I}$ is a model of $\mathcal{T}$) iff for each $\alpha \in \mathcal{T}$, $\mathcal{I} \models \alpha$, i.e. $\mathcal{I}$ is a model of $\alpha$
- We now overload even further the " $\models$ " symbol: $\mathcal{T} \models \alpha$ (read axiom $\alpha$ is a semantic consequence of the TBox $\mathcal{T}$) iff
  - all models of $\mathcal{T}$ are also models of $\alpha$, i.e.
  - for all interpretations $\mathcal{I}$, if $\mathcal{I} \models \mathcal{T}$ holds, then $\mathcal{I} \models \alpha$ also holds

# TBox reasoning tasks

Reasoning tasks on TBoxes only (i.e. no ABoxes involved)

- A base assumption: the TBox is **consistent** (does not contain a contradiction), i.e. it has a model
- **Subsumption**: concept $C$ is subsumed by concept $D$ wrt. a TBox $\mathcal{T}$, iff $\mathcal{T} \models (C \sqsubseteq D)$, i.e. $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds in all $\mathcal{I}$ models of $\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$)
  e.g. $\mathcal{T}_{fam} \models (\text{Grandmother} \sqsubseteq \text{Parent})$ (recall that $\mathcal{T}_{fam}$ is the family TBox)
- **Equivalence**: concepts $C$ and $D$ are equivalent wrt. a TBox $\mathcal{T}$, iff $\mathcal{T} \models (C \equiv D)$, i.e. $C^{\mathcal{I}} = D^{\mathcal{I}}$ holds in all $\mathcal{I}$ models of $\mathcal{T}$ ($C \equiv_{\mathcal{T}} D$).
  e.g. $\mathcal{T}_{fam} \models (\text{Parent} \equiv \text{Person} \sqcap \exists \text{hasChild}.\text{Person})$
- **Disjointness**: concepts $C$ and $D$ are disjoint wrt. a TBox $\mathcal{T}$, iff $\mathcal{T} \models (C \sqcap D \equiv \bot)$, i.e. $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ holds in all $\mathcal{I}$ models of $\mathcal{T}$.
  e.g. $\mathcal{T}_{fam} \models (\text{Woman} \sqcap \text{Man}) \equiv \bot$
- Note that all these tasks involve two concepts, $C$ and $D$

# Reducing reasoning tasks to testing satisfiability

- We now introduce a simpler, but somewhat artificial reasoning task: checking the satisfiability of a concept
- **Satisfiability**: a concept $C$ is satisfiable wrt. TBox $\mathcal{T}$, iff there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}$ is non-empty (hence $C$ is non-satisfiable wrt. $\mathcal{T}$ iff in all $\mathcal{I}$ models of $\mathcal{T}$ $C^{\mathcal{I}}$ is empty)
- We will reduce each of the earlier tasks to checking non-satisfiability
- E.g. to prove: $\text{Woman} \sqsubseteq \text{Person}$, let's construct a concept $C$ that contains all counter-examples to this statement: $C = \text{Woman} \sqcap \neg\text{Person}$
- If we can prove that $C$ has to be empty, i.e. there are no counter-examples, then we have proven the subsumption
- Assume we have a method for checking satisfiability. Other tasks can be reduced to this method (usable in $\mathcal{ALC}$ and above):
  - $C$ is subsumed by $D \iff C \sqcap \neg D$ is not satisfiable
  - $C$ and $D$ are equivalent $\iff (C \sqcap \neg D) \sqcup (D \sqcap \neg C)$ is not satisfiable
  - $C$ and $D$ are disjoint $\iff C \sqcap D$ is not satisfiable
- In simpler languages, not supporting full negation, such as $\mathcal{ALN}$, all reasoning tasks can be reduced to subsumption

# Contents

# The $\mathcal{SHIQ}$ Description Logic language – an overview

- Expanding the abbreviation $\mathcal{SHIQ}$
  - $\mathcal{S} \equiv \mathcal{ALC}_{\mathcal{R}^+}$ (language $\mathcal{ALC}$ extended with transitive roles), i.e. one can state that certain roles (e.g. hasAncestor) are transitive.
  - $\mathcal{H} \equiv$ role hierarchies. Adds statements of the form $R \sqsubseteq S$, e.g. if a pair of objects belongs to the hasFriend relationship, then it must belong to the knows relationship too: hasFriend $\sqsubseteq$ knows (could be stated in English as: *everyone knows their friends*)
  - $\mathcal{I} \equiv$ inverse roles: allows using role expressions $R^-$ to denote the inverse of role $R$, e.g. hasParent $\equiv$ hasChild$^-$
  - $\mathcal{Q} \equiv$ qualified number restrictions (a generalisation of $\mathcal{N}$): allows the use of concept expressions $(\leqslant nR.C)$ and $(\geqslant nR.C)$ e.g. those who have at least 3 tall children : ($\geqslant$ 3 hasChild.Tall)

# $\mathcal{SHIQ}$ language extensions – the details

- Language $\mathcal{S} \equiv \mathcal{ALC}_{\mathcal{R}^+}$, i.e, $\mathcal{ALC}$ plus transitivity (cf. the index $_{R^+}$)
  - Concept axioms and concept expressions – same as in $\mathcal{ALC}$
  - An additional axiom type: **Trans**($R$) declares role $R$ to be transitive
- Extension $\mathcal{H}$ – introducing role hierarchies
  - Adds role axioms of the form $R \sqsubseteq S$ and $R \equiv S$
    ($R \equiv S$ can be eliminated, replacing it by $R \sqsubseteq S$ and $S \sqsubseteq R$)
  - In $\mathcal{SH}$ it is possible describe a weak form of transitive closure:

$$\mathbf{Trans}(\text{hasDescendant})$$
$$\text{hasChild} \sqsubseteq \text{hasDescendant}$$

  - This means that hasDescendant is a transitive role which includes hasChild
  - What we cannot express in $\mathcal{SH}$ is that hasDescendant is the smallest such role. (This property cannot be described in FOL either.)

# $\mathcal{SHIQ}$ language extensions – the details (2)

Extension $\mathcal{I}$ – adding inverse roles

- Our first role constructor is $^-$: $R^-$ is the inverse of role $R$
- Example: consider role axiom hasChild$^-$ $\equiv$ hasParent and:

$$\begin{aligned} \text{GoodParent} &\equiv \exists\text{hasChild}.\top \sqcap \forall\text{hasChild}.\text{Happy} \\ \text{MerryChild} &\equiv \exists\text{hasParent}.\text{GoodParent} \end{aligned}$$

  A consequence of the above axioms: MerryChild $\sqsubseteq$ Happy
- Multiple inverses can be eliminated: $(R^-)^- \equiv R, ((R^-)^-)^- \equiv R^-, \ldots$.

# $\mathcal{SHIQ}$ language extensions – the details (3)

- Extension $\mathcal{Q}$ – qualified number restrictions – generalizing extension $\mathcal{N}$:
  - $(\leqslant nR.C)$ – the set of those who have at most $n$ $R$-related individuals belonging to $C$, e.g.
    $(\leqslant 2\text{hasChild}.\text{Female})$ – those with at most 2 daughters
  - $(\geqslant nR.C)$ – those with at least $n$ $R$-related individuals belonging to $C$
- A role is simple if it is not transitive and does not have a transitive sub-role
- Important: roles appearing in number restrictions have to be simple.
  (This is because otherwise the decidability of the language would be lost.)
  - If the axiom **Trans**(hasDesc) is present, this means that role hasDesc is not simple, and so cannot be used in number restrictions
  - If we add further role axioms: hasAnc $\equiv$ hasDesc$^{-}$,
    hasAnc $\sqsubseteq$ hasBloodRelation, then hasBloodRelation is not simple, as
    - hasAnc is transitive because its inverse hasDesc is such
    - hasBloodRelation has the transitive hasAnc as its sub-role

# $\mathcal{SHIQ}$ syntax summary

## Notation

- $A$ – atomic concept, $C$, $C_i$, $D$ – concept expressions
- $R_A$ – atomic role, $R$, $R_i$ – role expressions,
  $R_S$ – simple role expression, i.e. a role with no transitive sub-role

## Concept expressions

| DL | OWL | Name | Informal definition | |
|---|---|---|---|---|
| $A$ | $A$ | atomic concept | those in $A$ | $\mathcal{AL}$ |
| $\top$ | `owl:Thing` | top | the set of all objects | $\mathcal{AL}$ |
| $\bot$ | `owl:Nothing` | bottom | the empty set | $\mathcal{AL}$ |
| $C \sqcap D$ | $C$ `and` $D$ | intersection | those in both $C$ and $D$ | $\mathcal{AL}$ |
| $\forall R.C$ | $R$ `only` $C$ | value restriction | those whose all $R$s belong to $C$ | $\mathcal{AL}$ |
| $C \sqcup D$ | $C$ `or` $D$ | union | those in either $C$ or $D$ | $\mathcal{U}$ |
| $\exists R.C$ | $R$ `some` $C$ | existential restr. | those with an $R$ belonging to $C$ | $\mathcal{E}$ |
| $\neg C$ | `not` $C$ | full negation | those not in $C$ | $\mathcal{C}$ |
| $(\leqslant nR_S)$ | $R_S$ `max` $n$ $C$ | qualif. num. restr. | those with at most $n$ $R_S$s in $C$ | $\mathcal{Q}$ |
| $(\geqslant nR_S)$ | $R_S$ `min` $n$ $C$ | qualif. num. restr. | those with at least $n$ $R_S$s in $C$ | $\mathcal{Q}$ |

# $\mathcal{SHIQ}$ syntax summary (2)

- The syntax of role expressions

| | | | | |
|---|---|---|---|---|
| $R \rightarrow$ | | $R_A$ | *atomic role* | $(\mathcal{AL})$ |
| | $\mid$ | $R^-$ | *inverse role* | $(\mathcal{I})$ |

- The syntax of terminological axioms

| | | | | |
|---|---|---|---|---|
| $T \rightarrow$ | | $C_1 \equiv C_2$ | *concept equivalence axiom* | $(\mathcal{AL})$ |
| | $\mid$ | $C_1 \sqsubseteq C_2$ | *concept subsumption axiom* | $(\mathcal{AL})$ |
| | $\mid$ | $R_1 \equiv R_2$ | *role equivalence axiom* | $(\mathcal{H})$ |
| | $\mid$ | $R_1 \sqsubseteq R_2$ | *role subsumption axiom* | $(\mathcal{H})$ |
| | $\mid$ | $\mathbf{Trans}(R)$ | *transitivity axiom* | $(\mathcal{R}^+)$ |

# $\mathcal{SHIQ}$ semantics (ADVANCED)

- The semantics of concept expressions

$$
\begin{array}{rcll}
\top^{\mathcal{I}} & = & \Delta^{\mathcal{I}} & \\
\bot^{\mathcal{I}} & = & \emptyset & \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} & \\
(C_1 \sqcap C_2)^{\mathcal{I}} & = & C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} & \\
(C_1 \sqcup C_2)^{\mathcal{I}} & = & C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}} & \\
(\forall R.C)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid & \forall b.\langle a, b \rangle \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid & \exists b.\langle a, b \rangle \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \\
(\geqslant n\, R.C)^{\mathcal{I}} & = & \{\, a \in \Delta^{\mathcal{I}} \mid & |\{b \mid \langle a, b \rangle \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}| \geq n \,\} \\
(\leqslant n\, R.C)^{\mathcal{I}} & = & \{\, a \in \Delta^{\mathcal{I}} \mid & |\{b \mid \langle a, b \rangle \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}| \leq n \,\}
\end{array}
$$

- The semantics of role expressions

$$
(R^{-})^{\mathcal{I}} = \{\langle b, a \rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \langle a, b \rangle \in R^{\mathcal{I}}\}
$$

# $\mathcal{SHIQ}$ semantics (2) (ADVANCED)

- The semantics of terminological axioms

$$
\begin{array}{lcl}
\mathcal{I} \models C_1 \equiv C_2 & \Leftrightarrow & C_1^{\mathcal{I}} = C_2^{\mathcal{I}} \\
\mathcal{I} \models C_1 \sqsubseteq C_2 & \Leftrightarrow & C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}} \\
\mathcal{I} \models R_1 \equiv R_2 & \Leftrightarrow & R_1^{\mathcal{I}} = R_2^{\mathcal{I}} \\
\mathcal{I} \models R_1 \sqsubseteq R_2 & \Leftrightarrow & R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}} \\
\mathcal{I} \models \mathbf{Trans}(R) & \Leftrightarrow & (\forall a, b, c \in \Delta^{\mathcal{I}}) \\
 & & (\langle a, b \rangle \in R^{\mathcal{I}} \wedge \langle b, c \rangle \in R^{\mathcal{I}} \rightarrow \langle a, c \rangle \in R^{\mathcal{I}})
\end{array}
$$

- Read $\mathcal{I} \models T$ as: "$\mathcal{I}$ satisfies axiom $T$" or as "$\mathcal{I}$ is a model of $T$"

# Negation normal form (NNF)

- Various normal forms are used in reasoning algorithms
- The tableau algorithms use NNF: only atomic negation allowed
- To obtain NNF, apply the following rules repeatedly until no subterm matching a left hand side can be found:

$$
\begin{array}{rcl}
\neg\neg C & \leadsto & C \\
\neg(C \sqcap D) & \leadsto & \neg C \sqcup \neg D \\
\neg(C \sqcup D) & \leadsto & \neg C \sqcap \neg D \\
\neg(\exists R.C) & \leadsto & \forall R.(\neg C) \\
\neg(\forall R.C) & \leadsto & \exists R.(\neg C) \\
\neg(\leqslant nR.C) & \leadsto & (\geqslant kR.C) \text{ where } k = n+1 \\
\neg(\geqslant 1R.C) & \leadsto & \forall R.(\neg C) \\
\neg(\geqslant nR.C) & \leadsto & (\leqslant kR.C) \text{ if } n > 1, \text{ where } k = n-1
\end{array}
$$

# Going beyond $\mathcal{SHIQ}$ – outline

- Extension $\mathcal{O}$ introduces nominals, i.e. concepts which can only have a single element. Example: {EUROPE} is a concept whose interpretation must contain a single element
  FullyEuropean $\equiv$ $\forall$hasSite.$\forall$hasLocation.{EUROPE}
- Extension (**D**): concrete domains, e.g. integers, strings etc, whose interpretation is fixed, cf. data properties in OWL
- The Web Ontology Language OWL 1 implements $\mathcal{SHOIN}$(**D**)
- OWL 2 implements $\mathcal{SROIQ}$(**D**)
- The main novelty in $\mathcal{R}$ wrt. $\mathcal{H}$ is the possibility to use role composition ($\circ$):
  hasParent $\circ$ hasBrother $\sqsubseteq$ hasUncle
  i.e. one's parent's brother is one's uncle
- To ensure decidability, the use of role composition is seriously restricted (e.g. it is not allowed to have $\equiv$ instead of $\sqsubseteq$ in the above example)

# Contents

# The notion of ABox

- The ABox contains assertions about individuals, referred to by individual names $a, b, c$ etc.
  Convention: concrete individual names are written in ALL_CAPITALS
  - concept assertions: $C(a)$, e.g. Father(ALEX), $(\exists$hasJob$.\top)$(BOB)
  - role assertions: $R(a, b)$, e.g. hasChild(ALEX, BOB).
- Individual names correspond to constant symbols of first order logic
- The interpretation function has to be extended:
  - to each individual name $a$, $\mathcal{I}$ assigns $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
- The semantics of ABox assertions is straightforward:
  - $\mathcal{I}$ satisfies a concept assertion $C(a)$ ($\mathcal{I} \models C(a)$), iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$,
  - $\mathcal{I}$ satisfies a role assertion $R(a, b)$ ($\mathcal{I} \models R(a, b)$), iff $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$,
  - $\mathcal{I}$ satisfies an ABox $\mathcal{A}$ ($\mathcal{I} \models \mathcal{A}$) iff $\mathcal{I}$ satisfies all assertions in $\mathcal{A}$, i.e. for all $\alpha \in \mathcal{A}$, $\mathcal{I} \models \alpha$ holds

# Reasoning on ABoxes

- ABox $\mathcal{A}$ is consistent wrt. TBox $\mathcal{T}$
  if and only if
  there is an interpretation $\mathcal{I}$ which satisfies both $\mathcal{A}$ and $\mathcal{T}$
  i.e. $\mathcal{I} \models \mathcal{A}$ and $\mathcal{I} \models \mathcal{T}$
- Is the ABox $\{\mathsf{Mother}(\mathsf{S}), \mathsf{Father}(\mathsf{S})\}$ consistent wrt. an empty TBox?
- Is this ABox consistent wrt. the family TBox (slide 385)?
- Assertion $\alpha$ is said to be a consequence of the ABox $\mathcal{A}$ wrt. TBox $\mathcal{T}$ ($\mathcal{A} \models_{\mathcal{T}} \alpha$):
  - whenever an interpretation $\mathcal{I}$ satisfies both the ABox $\mathcal{A}$ and the TBox $\mathcal{T}$ ($\mathcal{I} \models \mathcal{A}$ and $\mathcal{I} \models \mathcal{T}$)
  - $\alpha$ is bound to hold in interpretation $\mathcal{I}$ ($\mathcal{I} \models \alpha$)

# Reasoning on ABoxes – example

- Let $\mathcal{T}$ refer to the family TBox from slide 385:

$$
\begin{array}{rcl}
\text{Woman} & \equiv & \text{Person} \sqcap \text{Female} \\
\text{Man} & \equiv & \text{Person} \sqcap \neg\text{Woman} \\
\text{Mother} & \equiv & \text{Woman} \sqcap \exists\text{hasChild}.\text{Person} \\
\text{Father} & \equiv & \text{Man} \sqcap \exists\text{hasChild}.\text{Person} \\
\text{Parent} & \equiv & \text{Father} \sqcup \text{Mother} \\
\text{Grandmother} & \equiv & \text{Woman} \sqcap \exists\text{hasChild}.\text{Parent}
\end{array}
$$

- Consider the ABox $\mathcal{A}$:

| | | | |
|---|---|---|---|
| hasChild(SAM, SUE) | Person(SAM) | Person(SUE) | Person(ANN) |
| hasChild(SUE, ANN) | Female(SUE) | Female(ANN) | |

- Which of the assertions below is a consequence of $\mathcal{A}$ wrt. $\mathcal{T}$?
  1. Mother(SUE)
  2. Mother(SAM)
  3. ¬Mother(SAM)
  4. Father(SAM)
  5. (Mother⊔Father)(SAM)
  6. ($\leqslant$ 1 hasChild)(SAM)

# ABoxes and databases

- An ABox may seem similar to a relational database, but
  - Querying a database uses the closed world assumption (CWA): is the query true in the world (interpretation) where the given and only given facts hold?
  - Contrastingly, ABox reasoning uses logical consequence, also called open world assumption (OWA): is it the case that the query holds in all interpretations satisfying the given facts
- At first one may think that with CWA one can always get more deduction possibilities
- However, case-based reasoning in OWA can lead to deductions not possible with CWA (e.g. Susan being optimistic)

# Some important ABox reasoning tasks

- Instance check: Decide if assertion $\alpha$ is a consequence of ABox $\mathcal{A}$ wrt. $\mathcal{T}$.
  Example: Check if Mother(SUE) holds wrt. the example ABox $\mathcal{A}$ and the family TBox on slide 403.
- Instance retrieval:
  Given a concept expression $C$ find the set of all individual names $x$ such that $\mathcal{A} \models_{\mathcal{T}} C(x)$
  Example: Find all individual names known to belong to the concept Mother

# The optimists example as an ABox reasoning task

- Our earlier example of optimists:
  (1) If someone has an optimistic parent, then she is optimistic herself.
  (2) If someone has a non-optimistic friend, then she is optimistic.
  (3) Susan's maternal grandfather has her maternal grandmother as a friend.
- Consider the following TBox $\mathcal{T}$:

$$\exists \mathsf{hP}.\mathsf{Opt} \sqsubseteq \mathsf{Opt} \qquad (1)$$

$$\exists \mathsf{hF}.\neg\mathsf{Opt} \sqsubseteq \mathsf{Opt} \qquad (2)$$

- Consider the following ABox $\mathcal{A}$, representing (3):
  $\mathsf{hP}(\mathsf{S}, \mathsf{SM})$ $\quad$ $\mathsf{hP}(\mathsf{SM}, \mathsf{SMM})$ $\quad$ $\mathsf{hP}(\mathsf{SM}, \mathsf{SMF})$ $\quad$ $\mathsf{hF}(\mathsf{SMF}, \mathsf{SMM})$
- An instance retrieval task: find the set of all individual names $x$ such that $\mathcal{A} \models_{\mathcal{T}} \mathsf{Opt}(x)$

# Contents

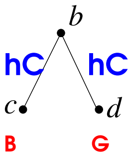
# Tableau algorithms

- Various TBox and ABox reasoning tasks have been presented earlier
- In $\mathcal{ALC}$ and above, any TBox task can be reduced to checking satisfiability
- Principles of the $\mathcal{ALCN}$ tableau algorithm
  - It checks if a concept is satisfiable, by trying to construct a model
  - Uses NNF, i.e. "$\neg$" can appear only in front of atomic concepts
  - The model is built through a series of transformations
- The data structure representing the model is called the tableau (state):
  - a directed graph
  - the vertices can be viewed as the domain of the interpretation
  - edges correspond to roles, each edge is labelled by a role
  - vertices are labelled with sets of concepts, to which the vertex is expected to belong
- Example: If a person has a green-eyed and a blonde child, does it follow that she/he has to have a child who is both green-eyed and blonde?
- Formalize the above task as a question in the Description Logic $\mathcal{ALC}$: Does the axiom $(\exists hC.B) \sqcap (\exists hC.G) \sqsubseteq \exists hC.(B \sqcap G)$ hold?[6]

[6] (hC = has child, B = blonde, G = green-eyed)

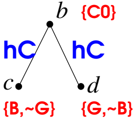
# An introductory example, using $\mathcal{ALC}$

- Question: Does the axiom $(\exists hC.B) \sqcap (\exists hC.G) \sqsubseteq \exists hC.(B \sqcap G)$ hold? (1)
- Transform to an **un**satisfiability task ($U \sqsubseteq V \Leftrightarrow U \sqcap \neg V$ is **not** satisfiable): $C = (\exists hC.B) \sqcap (\exists hC.G) \sqcap \neg(\exists hC.(B \sqcap G))$ is not satisfiable
- The neg. normal form of $C$ is: $C_0 = (\exists hC.B) \sqcap (\exists hC.G) \sqcap \forall hC.(\neg B \sqcup \neg G)$
- Goal: build an interpretation $\mathcal{I}$ such that $C_0^{\mathcal{I}} \neq \emptyset$. Thus we try to have a $b$ such that $b \in (\exists hC.B)^{\mathcal{I}}$, $b \in (\exists hC.G)^{\mathcal{I}}$, and $b \in (\forall hC.(\neg B \sqcup \neg G))^{\mathcal{I}}$.
- From $b \in (\exists hC.B)^{\mathcal{I}} \implies \exists c$ such that $\langle b, c \rangle \in hC^{\mathcal{I}}$ and $c \in B^{\mathcal{I}}$. Similarly, $b \in (\exists hC.G)^{\mathcal{I}} \implies \exists d$, such that $\langle b, d \rangle \in hC^{\mathcal{I}}$ and $d \in G^{\mathcal{I}}$.
- As $b$ belongs to $\forall hC.(\neg B \sqcup \neg G)$, and both $c$ and $d$ are hC relations of $b$, we obtain constraints: $c \in (\neg B \sqcup \neg G)^{\mathcal{I}}$ and $d \in (\neg B \sqcup \neg G)^{\mathcal{I}}$.
- $c \in (\neg B \sqcup \neg G)^{\mathcal{I}}$ means that either $c \in (\neg B)^{\mathcal{I}}$ or $c \in (\neg G)^{\mathcal{I}}$. Assuming $c \in (\neg B)^{\mathcal{I}}$ contradicts $c \in B^{\mathcal{I}}$. Thus we have to choose the option $c \in (\neg G)^{\mathcal{I}}$. Similarly, we obtain $d \in (\neg B)^{\mathcal{I}}$.
- We arrive at: $\Delta^{\mathcal{I}} = \{b, c, d\}$;
  $hC^{\mathcal{I}} = \{\langle b, c \rangle, \langle b, d \rangle\}$;
  $B^{\mathcal{I}} = \{c\}$ and $G^{\mathcal{I}} = \{d\}$.
  Here $b \in C_0^{\mathcal{I}}$, thus (1) does not hold.



# Extending the example to $\mathcal{ALCN}$

- Question: If a person having at most one child has a green-eyed and a blonde child, does it follow that she/he has to have a child who is both green-eyed and blonde?
- DL question: $(\leqslant 1 hC) \sqcap (\exists hC.B) \sqcap (\exists hC.G) \overset{?}{\sqsubseteq} \exists hC.(B \sqcap G))$ (2)
- Reformulation: "Is $C$ not satisfiable?", where
  $C = (\leqslant 1 hC) \sqcap (\exists hC.B) \sqcap (\exists hC.G) \sqcap \neg(\exists hC.(B \sqcap G))$
- Negation normal form:
  $C_0 = (\leqslant 1 hC) \sqcap (\exists hC.B) \sqcap (\exists hC.G) \sqcap \forall hC.(\neg B \sqcup \neg G))$
- We first build the same tableau as for (1):




- From $(\leqslant 1 hC)(b)$, $hC(b,c)$, and $hC(b,d)$ it follows that $c = d$ has to be the case. However merging $c$ and $d$ results in an object being both B and $\neg$B which is a contradiction (clash)
- Thus we have shown that $C_0$ cannot be satisfied, and thus the answer to question (2) is yes.

# Contents

# The $\mathcal{ALCN}$ tableau algorithm for empty TBoxes – outline

- "Is $C$ satisfiable?" $\Longrightarrow$ Let's build a model satisfying $C$, exhaustively.
- First, bring $C$ to negation normal form $C_0$.
- The main data structure, the tableau structure $T = (V, E, \mathcal{L}, I)$ where $(V, E, \mathcal{L})$ is a finite directed graph (more about $I$ later)
  - Nodes of the graph ($V$) can be thought of as domain elements.
  - Edges of the graph ($E$) represent role relationships between nodes.
  - The labeling function $\mathcal{L}$ assigns labels to nodes and edges:
    - $\forall x \in V$, $\mathcal{L}(x) \subseteq sub(C_0)$, the set of subexpressions of $C_0$
    - $\forall\langle x, y\rangle \in E$, $\mathcal{L}(\langle x, y\rangle)$ is a role within $C$ (in $\mathcal{SHIQ}$: set of roles)
  - The initial tableau has a single node, the root: $(\{x_0\}, \emptyset, \mathcal{L}, \emptyset)$, where $\mathcal{L}(x_0) = \{C_0\}$. Here $C_0$ is called the root concept.
- The algorithm uses transformation rules to extend the tableau
- Certain rules are nondeterministic, creating a choice point; backtracking occurs when a trivial clash appears (e.g. both $A$ and $\neg A \in \mathcal{L}(x)$)
- If a clash-free and complete tableau (no rule can fire) is reached $\Longrightarrow$ $C$ is satisfiable.
- When the whole search tree is traversed $\Longrightarrow$ $C$ is not satisfiable.

# Outline of the $\mathcal{ALCN}$ tableau algorithm (2)

- The tableau tree is built downwards from the root (edges are always directed downwards)
  - A node $b$ is called an $R$-successor (or simply successor) of $a$ iff there is an edge from $a$ to $b$ with $R$ as its label, i.e. $\mathcal{L}(\langle a, b \rangle) = R$
- Handling equalities and inequalities
  - To handle $(\leqslant n\,R)$ we need to merge (identify) nodes
  - In handling $(\geqslant n\,R)$ we will have to introduce $n$ $R$-successors which are pairwise non-identifiable ($x \dot{\neq} y$: $x$ and $y$ are not identifiable)
  - The component $I$ of the tableau data structure $T = (V, E, \mathcal{L}, I)$ is a set of inequalities of the form $x \dot{\neq} y$

# Transformation rules of the $\mathcal{ALCN}$ tableau algorithm (1)

## $\sqcap$-rule

***Condition:*** $(C_1 \sqcap C_2) \in \mathcal{L}(x)$ and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$

***New state T$'$:*** $\mathcal{L}'(x) = \mathcal{L}(x) \cup \{C_1, C_2\}$.

## $\sqcup$-rule

***Condition:*** $(C_1 \sqcup C_2) \in \mathcal{L}(x)$ and $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$.

***New state T$_1$:*** $\mathcal{L}'(x) = \mathcal{L}(x) \cup \{C_1\}$.

***New state T$_2$:*** $\mathcal{L}'(x) = \mathcal{L}(x) \cup \{C_2\}$.

## $\exists$-rule

***Condition:*** $(\exists R.C) \in \mathcal{L}(x)$, $x$ has no $R$-successor $y$ s.t. $C \in \mathcal{L}(y)$.

***New state T$'$:*** $V' = V \cup \{y\}$ ($y$ is a new node),

$E' = E \cup \{\langle x, y \rangle\}$, $\mathcal{L}'(\langle x, y \rangle) = R$, $\mathcal{L}'(y) = \{C\}$.

## $\forall$-rule

***Condition:*** $(\forall R.C) \in \mathcal{L}(x)$, $x$ has an $R$-successor $y$ s.t. $C \notin \mathcal{L}(y)$.

***New state T$'$:*** $\mathcal{L}'(y) = \mathcal{L}(y) \cup \{C\}$.

# Transformation rules of the $\mathcal{ALCN}$ tableau algorithm (2)

**$\geqslant$-rule**

***Condition:*** $(\geqslant n\, R) \in \mathcal{L}(x)$ and $x$ has no $n$ $R$-successors such that any two are non-identifiable.

***New state* T$'$*:*** $V' = V \cup \{y_1, \ldots, y_n\}$ ($y_i$ new nodes),

$E' = E \cup \{\langle\, x, y_1 \,\rangle, \ldots, \langle\, x, y_n \,\rangle\}$,

$\mathcal{L}'(\langle\, x, y_i \,\rangle) = R$, $\mathcal{L}'(y_i) = \emptyset$, for each $i = 1 \leq i \leq n$,

$I' = I \cup \{y_i \doteq\!\!\!\!/\; y_j \mid 1 \leq i < j \leq n\}$.

# Transformation rules of the $\mathcal{ALCN}$ tableau algorithm (3)

**$\leqslant$-rule**

***Condition:*** $(\leqslant n R) \in \mathcal{L}(x)$ and $x$ has $R$-successors $y_1, \ldots, y_{n+1}$ among which there are at least two identifiable nodes.

***For each*** $i$ and $j$, $1 \leq i < j \leq n+1$, where $y_i$ and $y_j$ are identifiable:

***New state*** $\mathbf{T}_{ij}$***:***

$$V' = V \setminus \{y_j\},\ \mathcal{L}'(y_i) = \mathcal{L}(y_i) \cup \mathcal{L}(y_j),$$

$$E' = E \setminus \{\langle x, y_j \rangle\} \setminus \{\langle y_j, u \rangle | \langle y_j, u \rangle \in E\} \cup \{\langle y_i, u \rangle | \langle y_j, u \rangle \in E\},$$

$$\mathcal{L}'(\langle y_i, u \rangle) = \mathcal{L}(\langle y_j, u \rangle),\ \forall u \text{ such that } \langle y_j, u \rangle \in E,$$

$I' = I[y_j \rightarrow y_i]$ (every occurrence of $y_j$ is replaced by $y_i$).

# The $\mathcal{ALCN}$ tableau algorithm – further details

- There is clash at some node $x$ of a tableau state iff
  - $\{\bot\} \subseteq \mathcal{L}(x)$; or
  - $\{A, \neg A\} \subseteq \mathcal{L}(x)$ for some atomic concept $A$; or
  - $(\leqslant nR) \in \mathcal{L}(x)$ and $x$ has $R$-successors $y_1, \ldots, y_{n+1}$ where for any two successors $y_i$ and $y_j$ it holds that $y_i \dot{\neq} y_j \in I$.
- A tableau state is said to be complete, if no transformation rules can be applied at this state (there is no rule the conditions of which are satisfied)

# The $\mathcal{ALCN}$ tableau algorithm

In this version the algorithm handles a set of tableau states, one for each yet unexplored subtree of the search space.

1. Intialise the variable `States` $= \{\mathbf{T}_0\}$ (a singleton set containing the initial tableau state)
2. If there is $\mathbf{T} \in$ `States` such that $\mathbf{T}$ contains a clash, remove $\mathbf{T}$ from `States` and continue at step 2
3. If there is $\mathbf{T} \in$ `States` such that $\mathbf{T}$ is complete (and clash-free), exit the algorithm, reporting satisfiability
4. If `States` is empty, exit the algorithm, reporting non-satisfiability
5. Choose an arbitrary element $\mathbf{T} \in$ `States` and apply to $\mathbf{T}$ an arbitrary transformation rule, whose conditions are satisfied[7] (don't care nondeterminism). Remove $\mathbf{T}$ from `States`, and add to `States` the `NewStates` resulting from the applied transformation, where `NewStates` $= \{\mathbf{T_1}, \mathbf{T_2}\}$ for the $\sqcup$-**rule**, `NewStates` $= \{\mathbf{T_{ij}} | \cdots\}$ for the $\leqslant$-**rule**, and `NewStates` $= \{\mathbf{T}'\}$ for all other (deterministic) rules. Continue at step 2

[7] Such a tableau state $\mathbf{T}$ and such a rule exist, because `States` is nonempty, and none of its elements is a complete tableau

# The $\mathcal{ALCN}$ tableau algorithm – an example

- Consider checking the satisfiability of concept $C_0$ (hC = has child, B = blonde):

$$
\begin{aligned}
C_0 &= C_1 \sqcap C_2 \sqcap C_3 \sqcap C_4 \\
C_1 &= (\geqslant 2\, \mathrm{hC}) \\
C_2 &= \exists \mathrm{hC}.\mathrm{B} \\
C_3 &= (\leqslant 2\, \mathrm{hC}) \\
C_4 &= C_5 \sqcup C_6 \\
C_5 &= \forall \mathrm{hC}.\neg \mathrm{B} \\
C_6 &= \mathrm{B}
\end{aligned}
$$

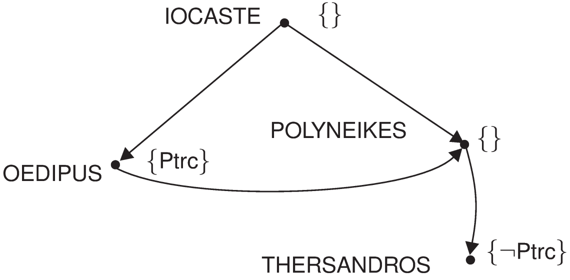
- The tableau algorithm completes with the answer: concept $C_0$ is satisfiable
- The interpretation constructed by the tableau algorithm: $\Delta^{\mathcal{I}} = \{b, c, d\}$; $\mathrm{hC}^{\mathcal{I}} = \{\langle b, c \rangle, \langle b, d \rangle\}$; $\mathrm{B}^{\mathcal{I}} = \{b, c\}$

# Extending the tableau algorithm to ABox reasoning

- To solve an ABox reasoning task (with no TBox), we transform the ABox to a graph, serving as the initial tableau state, e.g. for the IOCASTE family ABox:

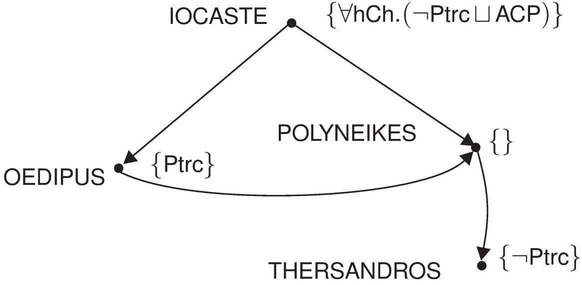
  hC(IOCASTE, OEDIPUS)
  hC(OEDIPUS, POLYNEIKES)
  Ptrc(OEDIPUS)
  hC(IOCASTE, POLYNEIKES)
  hC(POLYNEIKES, THERSANDROS)
  (¬ Ptrc)(THERSANDROS)

- Individual names become nodes of the graph, labelled by a set of concepts, and each role assertion generates an edge, labelled (implicitly) by hC:



# Handling ABox axioms in the tableau algorithm (ctd.)

- Given the Iocaste ABox, we want to prove that IOCASTE is special, i.e. she belongs to the concept $\exists$hC.(Ptrc $\sqcap$ $\exists$hC.$\neg$Ptrc)
- We do an indirect proof: assume that IOCASTE is not special, i.e. IOCASTE belongs to ($\forall$hC.($\neg$Ptrc $\sqcup$ $\forall$hC.Ptrc)) (1)
- Let's introduce an abbreviation: ACP $\equiv$ $\forall$hC.Ptrc
- To prove that Iocaste is special, we add concept (1) to the IOCASTE node:




- The tableau algorithm, with this initial state, will detect non-satisfiability

# Handling TBox axioms in the tableau algorithm

- An arbitrary $\mathcal{ALCN}$ TBox can be transformed to a set of subsumptions of the form $C \sqsubseteq D$ ($C \equiv D$ can be replaced by $\{C \sqsubseteq D, D \sqsubseteq C\}$)
- $C \sqsubseteq D$ can be replaced by $\top \sqsubseteq \neg C \sqcup D$
  cf. $(\alpha \rightarrow \beta)$ is the same as $(\neg\alpha \vee \beta)$
- An arbitrary TBox $\{C_1 \sqsubseteq D_1, C_2 \sqsubseteq D_2, \ldots, C_n \sqsubseteq D_n\}$ can be transformed to a single equivalent axiom: $\top \sqsubseteq C_{\mathcal{T}}$, where

$$C_{\mathcal{T}} = (\neg C_1 \sqcup D_1) \sqcap (\neg C_2 \sqcup D_2) \sqcap \cdots \sqcap (\neg C_n \sqcup D_n).$$

- Concept $C_{\mathcal{T}}$ is called the internalisation of TBox $\mathcal{T}$
- An interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ ($\mathcal{I} \models \mathcal{T}$) iff each element of the domain belongs to the $C_{\mathcal{T}}$ internalisation concept
  - This observation can be used in the tableaux reasoning algorithm, which tries to build a model
  - To build a model which satisfies the TBox $\mathcal{T}$ we add the concept $C_{\mathcal{T}}$ to the label of each node of the tableau

# Handling TBoxes in the tableau algorithm – problems

- Example: Consider the task of checking the satisfiability of concept Blonde wrt. TBox $\{\top \sqsubseteq \exists \text{hasFriend}.\text{Blonde}\}$
  - Concept $\exists$hasFriend.Blonde will appear in each node
  - thus the $\exists$-rule will generate an infinite chain of hasFriend successors
- To prevent the algorithm from looping the notion of blocking is introduced.

# Blocking

- Definition: Node $y$ is blocked by node $x$, if $y$ is a descendant of $x$ and the blocking condition $\mathcal{L}(y) \subseteq \mathcal{L}(x)$ holds (*subset blocking*).
- When $y$ is blocked, we disallow generator rules ($\exists$- and $\geqslant$-rules, creating new successors for $y$)
- This solves the termination problem, but raises the following issue
  - How can one get an interpretation from the tableau?
  - Solution (approximation, for $\mathcal{ALC}$ only): identify blocked node $y$ with blocking node $x$, i.e. redirect the edge pointing to $y$ so that it points to $x$. This creates a model, as
    - all concepts in the label of $y$ are also present in $x$
    - thus $x$ belongs to all concepts $y$ is expected to belong to
- Is Happy $\sqcap$ Blonde satisfiable wrt. TBox $\{\top \sqsubseteq \exists$hasFriend.Blonde$\}$ ?

$x$ o $\{$Happy, Blonde, $\exists$hasFriend.Blonde$\}$

hasFriend |

$y$ o $\{$Blonde, $\exists$hasFriend.Blonde$\}$

  - $x$ blocks $y$, the tableau is clash-free and complete
  - The model:
    $\Delta^{\mathcal{I}} = \{x\}$; $\text{Happy}^{\mathcal{I}} = \{x\}$; $\text{Blonde}^{\mathcal{I}} = \{x\}$; $\text{hasFriend}^{\mathcal{I}} = \{\langle x, x \rangle\}$