

# Ontologies (part 2)

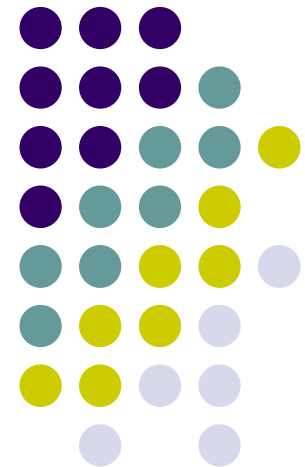
---

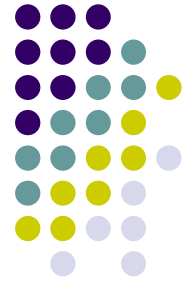
Lina Al-Jadir

UNIGE - SYINF

Cours Bases d'informations

Printemps 2007





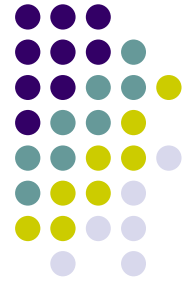
# Outline

- Introduction
  - Motivations
  - Definition
  - Classification
  - Requirements
- Ontology representations
  - Logic based models:
    - RDF(S)
    - DL
    - OWL
  - Database like models: Kaon
- Conclusion

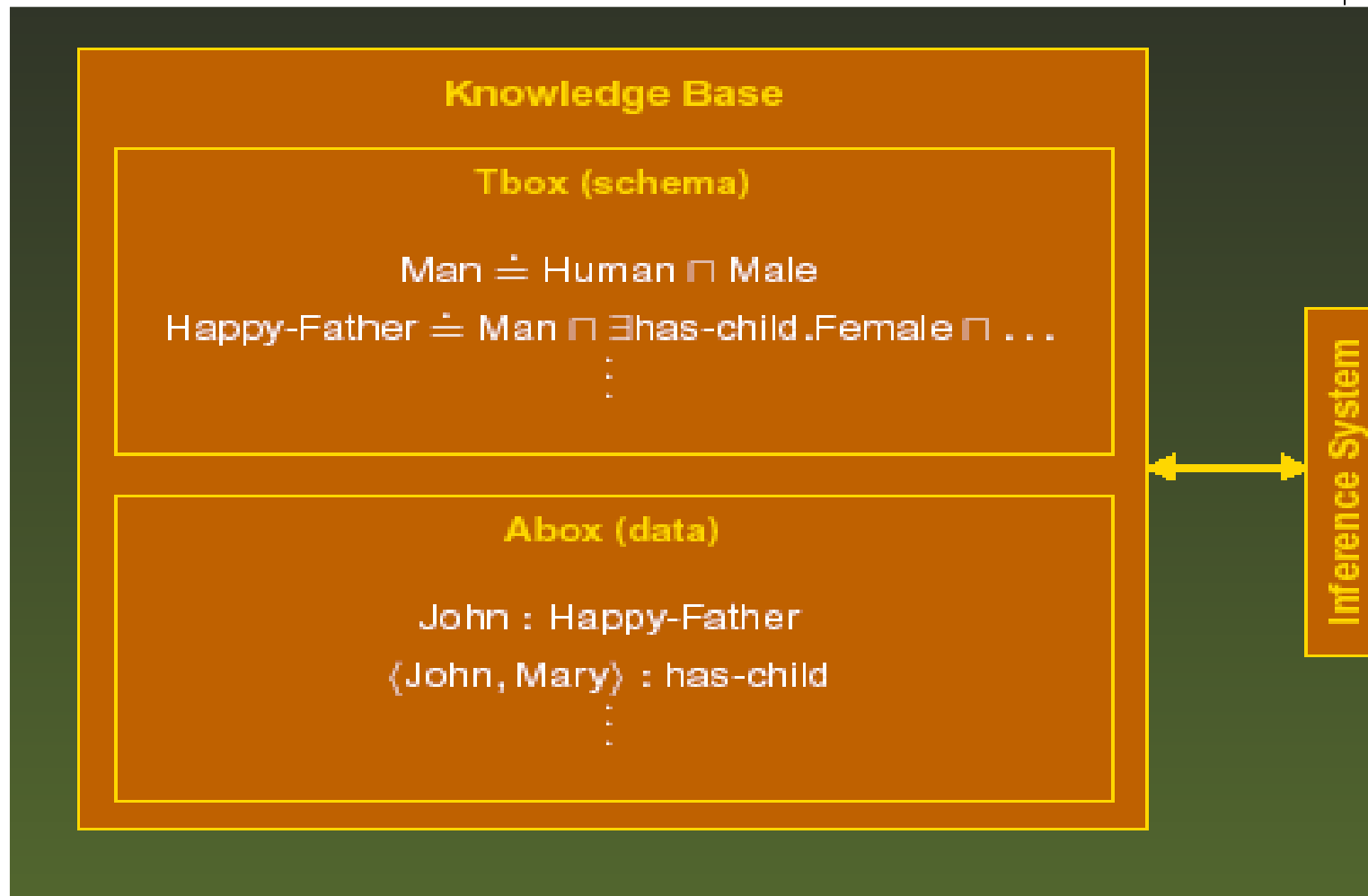


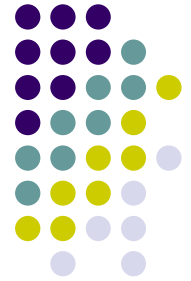
# Description Logics (DL)

- Description logics
  - Family of languages
  - Designed to represent knowledge
  - Based on formal semantics
- Core distinction between
  - Class definitions (TBox)
  - Instance definitions (ABox)



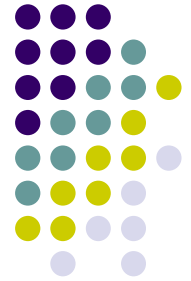
# Architecture of DL System





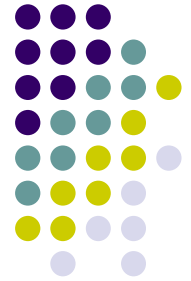
# DL Features

- Formalism of knowledge representation based on **concepts** and **roles**
- Complex concepts are defined thanks to **constructors**
- **Axioms** are used:
  - To name complex concepts
  - To state subsumption relationships between concepts



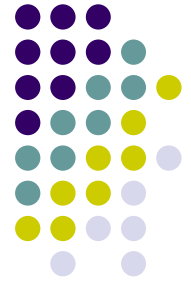
# DL Concepts and Roles

- **Concept (class):**
  - class of objects
- **Role (property):**
  - binary relationship between concepts
  - may hold cardinalities
- Complex concepts are defined by using constructors



# DL Constructors

- **Constructors** usually include (ALC Logic):
  - Intersection:  $\text{Human} \cap \text{Male}$
  - Union:  $\text{Man} \cup \text{Woman}$
  - Negation:  $\neg \text{Man}$
  - Existential restriction:  $\exists \text{hasChild.Male}$ 
    - Set of instances that have at least one child who is a male
  - Universal restriction:  $\forall \text{hasChild.Female}$ 
    - Set of instances whose children are all female



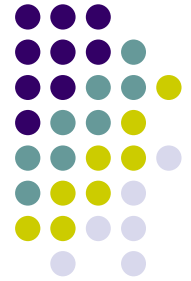
# DL Constructors

- May also include:
  - Number restrictions
    - $\leq 10$  hasChild
    - $\geq 3$  hasChild
    - $\geq 1$  hasMother  $\cap \leq 1$  hasMother
  - Inverse role
    - $\text{hasParent} \equiv \text{hasChild}^{-1}$
    - $\text{hasChild}(\text{elizabeth}, \text{charles}) \Rightarrow \text{hasParent}(\text{charles}, \text{elizabeth})$
  - Transitive role
    - $\text{hasAncestor}^*$
  - ...



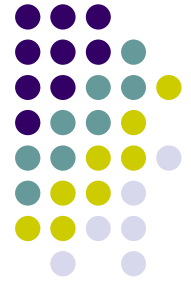
# DL Constructors

Constructor	DL Syntax	Example
intersectionOf	$C_1 \sqcap \dots \sqcap C_n$	Human $\sqcap$ Male
unionOf	$C_1 \sqcup \dots \sqcup C_n$	Doctor $\sqcup$ Lawyer
complementOf	$\neg C$	$\neg$ Male
oneOf	$\{x_1 \dots x_n\}$	{ Germany, France }
allValuesFrom	$\forall P.C$	$\forall$ hasParent.Human
someValuesFrom	$\exists P.C$	$\exists$ hasChild.Lawyer
hasValue	$\exists P.\{x\}$	$\exists$ citizenOf.{USA}
minCardinality	$\exists \geq n P$	$\exists \geq 1$ passport
maxCardinality	$\exists \leq n P$	$\exists \leq 2$ hasArm
cardinality	$\exists = n P$	$\exists = n$ head



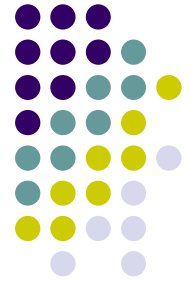
# Lisp-like syntax for DLs

$\top$	top or *top*
$\perp$	bottom or *bottom*
$\neg A$	(not A)
$C \cap D$	(and C D)
$C \cup D$	(or C D)
$\forall R.C$	(all R C)
$\exists R.T$	(some R top)
$\exists R.C$	(some R C)
$\exists_{\geq n} R$	( $\geq n$ R) or (at-least n R)
$\exists_{\leq n} R$	( $\leq n$ R) or (at-most n R)



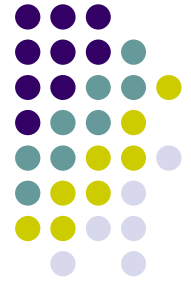
# TBox

- **TBox**: set of terminological axioms
- **Terminological axioms**:
  - $C \subseteq D$  inclusion axiom  
C is subsumed by D, or D subsumes C
  - $C \equiv D$  equivalence axiom  
C is equivalent to D
- **Definitions**
  - if the left hand side of an axiom is a name
    - $\text{Father} \equiv \text{Man} \cap \exists \text{hasChild.Human}$
    - $\text{Man} \subseteq \text{Person}$



# TBox Axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \subseteq C_2$	Human $\subseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
subPropertyOf	$P_1 \subseteq P_2$	hasDaughter $\subseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
sameAs	$\{x_1\} \equiv \{x_2\}$	{PresidentBush} $\equiv$ {GWBush}
disjointWith	$C_1 \subseteq \neg C_2$	Male $\subseteq \neg$ Female
differentIndividualAs	$\{x_1\} \subseteq \neg \{x_2\}$	{marc} $\subseteq \neg$ {hans}
inverseOf	$P_1 \equiv P_2^{-1}$	hasChild $\equiv$ hasParent <sup>-1</sup>
TransitiveProperty	$P^+ \subseteq P$	ancestor <sup>+</sup> $\subseteq$ ancestor



# More TBox Axioms

- Terminological **cycles**

- $\text{BinaryTree} \equiv \text{Tree} \cap (\leq 2 \text{ hasBranch}) \cap (\forall \text{hasBranch. BinaryTree})$

- General axioms

axioms whose LHS is not a concept name, or is **T** (thing, top)

- **sufficient condition** for concept grandma

$\text{woman} \cap \exists \text{hasChild}.\exists \text{hasChild}.\text{person} \subseteq \text{grandma}$

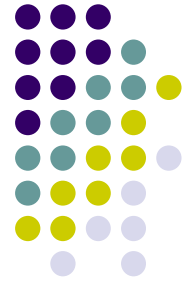
- **domain** for roles:  $\exists \text{hasChild}.\text{T} \subseteq \text{parent}$  or  $\text{T} \subseteq \forall \text{hasChild}^{\neg}.\text{parent}$
- **range** for roles:  $\text{T} \subseteq \forall \text{hasChild}.\text{person}$

# TBox Inference

[Haarslev, 05]



- Types of **inferences**:
  - **satisfiability** of concepts
    - is concept C satisfiable: will it be able to have instances?
  - **subsumption** of concepts
    - does concept C subsume concept D?
  - **equivalence** of concepts
    - is concept C equivalent to concept D?
  - **disjointness** of concepts
    - are concepts C and D disjoint?



# TBox Inference

- Concept satisfiability:

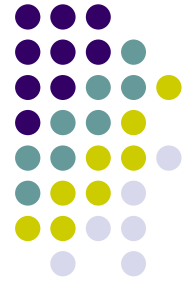
parent  $\equiv$  person  $\cap \exists \text{hasChild.person}$   
woman  $\equiv$  female  $\cap$  person  
mother  $\equiv$  female  $\cap$  parent

- The concepts **parent**, **woman**, **mother** are satisfiable
- $\neg \text{woman} \cap \text{mother}$  is unsatisfiable:
  - $\neg \text{woman} \cap \text{mother} \equiv$  (unfold)
  - $\neg(\text{female} \cap \text{person}) \cap \text{female} \cap \text{parent} \equiv$
  - $(\neg \text{female} \cup \neg \text{person}) \cap \text{female} \cap \text{parent} \equiv$
  - $(\neg \text{female} \cap \text{female} \cap \text{parent}) \cup (\neg \text{person} \cap \text{female} \cap \text{parent}) \equiv$
  - $\neg \text{person} \cap \text{female} \cap \text{parent} \equiv$
  - $\neg \text{person} \cap \text{female} \cap \text{person} \cap \exists \text{hasChild.person} \equiv \Phi$



# TBox Inference

- All concept inference services can be reduced to concept satisfiability:
  - let service **sat**(**C**, **T**), **C** a concept, **T** a TBox
  - **subsumes**(**C**, **D**, **T**)  $\equiv \neg \text{sat}(\neg \mathbf{C} \cap \mathbf{D}, \mathbf{T})$   
 $\mathbf{C} \supseteq \mathbf{D}$  holds  $\leftrightarrow \neg \mathbf{C} \cap \mathbf{D}$  unsatisfiable
  - **equivalence**(**C**, **D**, **T**)  $\equiv \text{subsumes}(\mathbf{C}, \mathbf{D}, \mathbf{T}) \wedge \text{subsumes}(\mathbf{D}, \mathbf{C}, \mathbf{T})$
  - **disjoint**(**C**, **D**, **T**)  $\equiv \neg \text{sat}(\mathbf{C} \cap \mathbf{D}, \mathbf{T})$



# TBox Inference

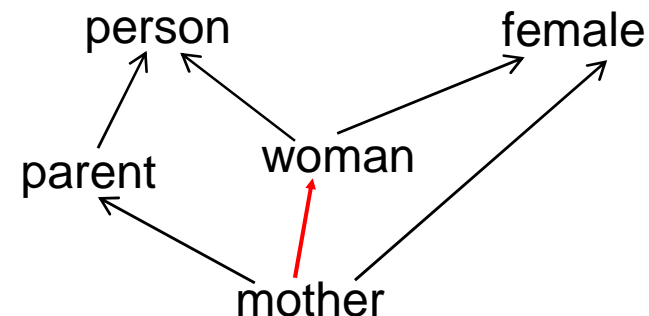
- Concept subsumption:

$\text{parent} \equiv \text{person} \sqcap \exists \text{hasChild}.\text{person}$

$\text{woman} \equiv \text{female} \sqcap \text{person}$

$\text{mother} \equiv \text{female} \sqcap \text{parent}$

- Does the concept **woman** subsume the concept **mother**?
- yes**





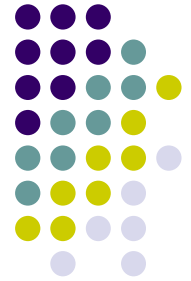
# TBox example

**Student**  $\equiv$  Person  $\cap \exists \text{register.University}$

**PublishedPaper**  $\equiv$  AcceptedPaper  $\cup$  Poster

**Poster**  $\equiv$  SubmittedPaper  
 $\cap \neg \text{AcceptedPaper}$   
 $\cap (\exists \text{hasReview.PositiveReview})$

**AcceptedPaper**  $\equiv$  SubmittedPaper  
 $\cap (\forall \text{hasReview.PositiveReview})$   
 $\cap (\geq 2 \text{ hasReview})$



# ABox

- **Abox**: set of assertional axioms
- **Assertional axioms**:
  - **concept assertions** for an individual  $a$ :  $C(a)$ 
    - $\text{Person}(\text{Paul})$
    - $(\text{Man} \sqcup \exists \text{hasChild.Male})(\text{John})$
  - **role assertions** for 2 individuals  $a, b$ :  $R(a, b)$ 
    - $\text{hasChild}(\text{John}, \text{Paul})$
- **Unique name assumption**
  - different names denote different individuals ( $a \neq b$ )
  - considered or not

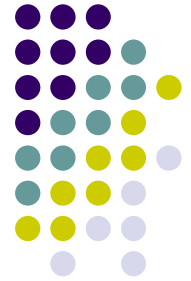
# ABox Inference

[Haarslev, 05]



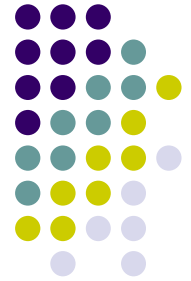
- Types of **inferences**:
  - ABox satisfiability
    - is the collection  $A$  of assertions satisfiable (wrt ABox  $A$  and TBox  $T$ )?
      - TBox:
        - $\text{Woman} \equiv \text{Person} \cap \text{Female}$
        - $\text{Man} \equiv \text{Person} \cap \neg \text{Woman}$
        - $\text{Mother} \equiv \text{Woman} \cap \exists \text{hasChild}.\text{Person}$
        - $\text{Father} \equiv \text{Man} \cap \exists \text{hasChild}.\text{Person}$
      - ABox:
        - $\text{Mother}(\text{Mary})$
        - $\text{Father}(\text{Mary})$

ok for ABox  
But inconsistent with TBox



# ABox Inference

- Instance checking:  $\text{instance?}(a, C, A)$ 
  - is  $a$  an instance of concept  $C$ ?
    - $\text{Woman}(\text{Chiara})$   
 $\text{hasChild}(\text{Chiara}, \text{Jeanne})$   
 $\text{Mother}(\text{Chiara})??$
- ABox realization
  - compute for all individuals in  $A$  their most-specific named concepts wrt TBox  $T$



# ABox Inference

- All inference services can be reduced to ABox satisfiability:
  - let service **asat**(*A*), *A* a ABox
  - instance checking:  
 $\text{instance?}(a, C, A) \equiv \neg \text{asat}(A \cup \{\neg C(a)\})$
  - concept satisfiability  
 $\text{sat}(C, T) \equiv \text{asat}(\{C(a)\})$
  - concept subsumption  
 $\begin{aligned} \text{subsumes}(C, D, T) &\equiv \neg \text{sat}(\neg C \cap D, T) \\ &\equiv \neg \text{asat}(\{(\neg C \cap D)(a)\}) \end{aligned}$



# ABox example

- woman(queen\_mum)  
hasChild(queen\_mum, elizabeth)  
woman(elizabeth)  
hasChild(elizabeth, charles)  
hasChild(elizabeth, anne)  
(parent  $\cap$  male) (charles)  
woman(anne)  
hasChild(charles, william)  
(person  $\cap$  male) (william)

# Closed vs Open-World Assumption



- DB
  - Schema
  - Instances
- What is not described in DB is **false**
- hasChild(Abdel, Mehdi)
  - Abdel has currently a single child

- DL
  - TBox
  - ABox
- What is not described in ABox is considered as **missing**
  - represented information is assumed incomplete
- hasChild(Abdel, Mehdi)
  - Abdel has a child but we can not state that he has currently a single child
  - we could add:  
( $\leq 1$  hasChild) (Abdel)

# Closed vs Open-World Assumption



- male(Mehdi)  
hasChild(Abdel, Mehdi)
  - Abdel has children (one) who are all male

- male(Mehdi)  
hasChild(Abdel, Mehdi)
  - we can not state that all the children of Abdel are male
  - we could add:  
( $\forall$ hasChild.male) (Abdel)
  - or we could assert that information about a 2<sup>nd</sup> child will not be added in the future, i.e. **close** a role for an individual  
( $\leq 1$  hasChild) (Abdel)



# Example (1)

## TBOX

$\text{Parent} = \text{Person} \cap \exists \text{ hasChild. Person}$

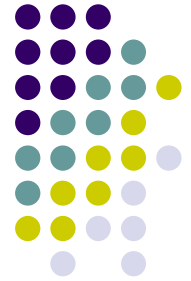
$\text{ParentWithSingleChild} = \text{Parent} \cap \leq 1 \text{ hasChild}$

## ABOX

$\text{Person}(\text{Paul})$

$\text{ParentWithSingleChild}(\text{Paul})$

*OK even if we do not specify a hasChild role for him  
(Parent)*



## Example (2)

### TBOX

Parent = Person  $\cap \exists$  hasChild.Person

ParentWithSingleChild = Parent  $\cap \leq 1$  hasChild

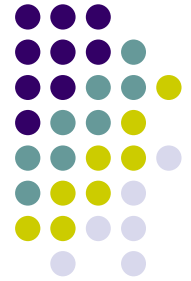
### ABOX

...

Parent(Marlyse)

(=1 hasChild) (Marlyse)

*Marlyse becomes an instance of ParentWithSingleChild*



## Example (3)

### TBOX

Parent = Person  $\cap \exists$  hasChild.Person

ParentWithSingleChild = Parent  $\cap \leq 1$  hasChild

### ABOX

...

ParentWithSingleChild(Abdel)

OK

hasChild(Abdel, Mehdi)

OK

hasChild(Abdel, Fadi)

Error! (*max card restr.*)



## Example (4)

### TBOX

Parent = Person  $\cap \exists$  hasChild.Person

ParentWithSingleChild = Parent  $\cap \leq 1$  hasChild

### ABOX

Person(Peter), Person(Harry)

hasChild(Peter, Harry)

- *Peter belongs to Parent,*
- *Peter does not belong to ParentWithSingleChild (OWA)*  
(=1 hasChild) (Peter)
- *Peter now belongs to ParentWithSingleChild*



# DL Reasoning

- TBox:
  - TBox coherence: list all unsatisfiable named concepts
  - Compute the subsumption hierarchy of named concepts
- ABox:
  - ABox consistency: consistent or not
  - ABox realization: compute for all individuals their most-specific named concepts



# DL Reasoning

- Reasoning useful at all stages of ontology life-cycle
  - **Ontology design and maintenance**
    - Check class consistency and (unexpected) implied relationships
    - Particularly important with large ontologies
  - **Ontology integration**
    - Assert inter-ontology relationships
    - Reasoner computes integrated class hierarchy/consistency
  - **Ontology deployment**
    - Determine whether ABox is consistent with ontology
    - Determine if individuals are instances of ontology classes

# Tableau Methods

[Haarslev, 05]



- How can we prove that C is satisfiable?
- By applying **tableau methods**
  - set of **completion rules** operating on constraint sets or tableaux
  - clash triggers
- **Proof procedure:**
  - transform all concepts into negation normal form:  
 $\neg(C \cap D) \rightarrow \neg C \cup \neg D, \neg \exists R.C \rightarrow \forall R.\neg C$
  - apply completion rules in arbitrary order as long as possible
  - application of rules:
    - stops in case of clash
    - terminates if no completion rule is applicable anymore
  - satisfiable iff a clash-free tableau can be derived

# Tableau Methods – Completion Rules (Logic ALC)



## clash trigger

$\{A(a), (\neg A)(a)\} \subseteq A$

## conjunction rule:

if 1.  $(C \cap D)(a) \in A$ , and  
2.  $\{C(a), D(a)\} \not\subseteq A$   
then  $A' = A \cup \{C(a), D(a)\}$

## disjunction rule:

if 1.  $(C \cup D)(a) \in A$ , and  
2.  $\{C(a), D(a)\} \cap A = \emptyset$   
then  $A' = A \cup \{C(a)\}$  or  
 $A' = A \cup \{D(a)\}$

## exists role restriction rule:

if 1.  $(\exists R.C)(a) \in A$ , and  
2.  $\neg \exists b \in O$  s.t.  $\{C(b), R(a,b)\} \subseteq A$   
then  $A' = A \cup \{C(b), R(a,b)\}$   
with  $b$  fresh in  $A$

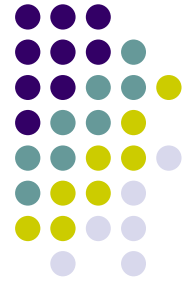
## universal role restriction rule:

if 1.  $(\forall R.C)(a) \in A$ , and  
2.  $\exists b \in O$  s.t.  $R(a,b) \in A$ , and  
3.  $C(b) \notin A$   
then  $A' = A \cup \{C(b)\}$



# Tableau Methods – Completion Rules

- Remarks:
  - Disjunction rule:
    - non-deterministic, 2 alternative ABoxes are explored
  - Existential role restriction rule:
    - $O$  is the set of all individual names.
    - the only rule that creates new individuals in ABox.
    - new individual (b) is considered anonymous.



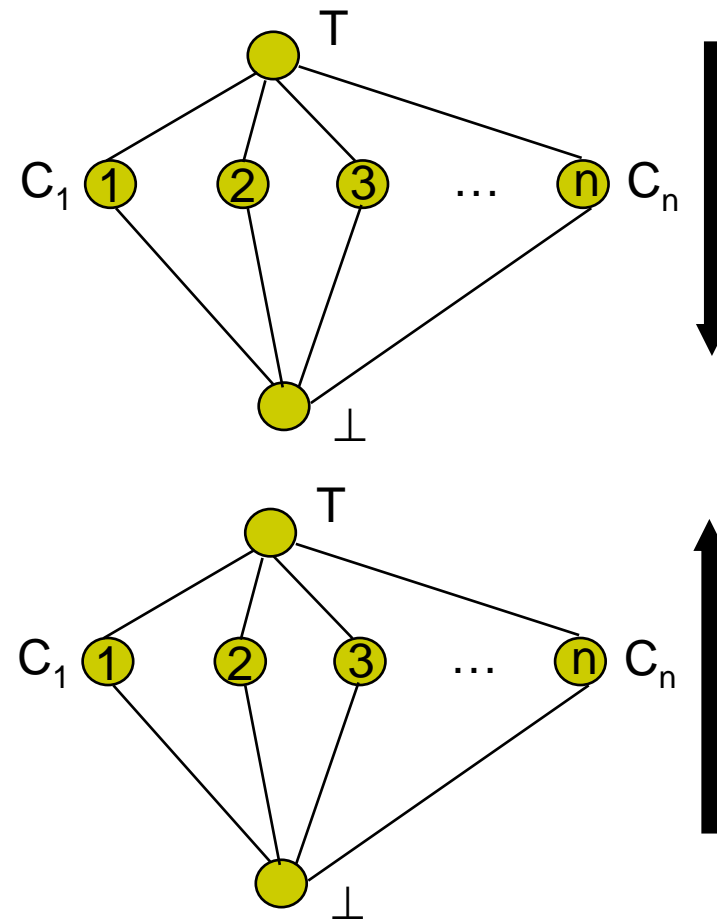
# Tableau Methods - example

- Does the concept **woman** subsume the concept **mother**?
- Is the concept  $\neg\mathbf{woman} \cap \mathbf{mother}$  unsatisfiable?
- Application of completion rules:
  - $A0 = \{(\neg\mathbf{woman} \cap \mathbf{mother})(a)\}$   
     $= \{(\neg(\mathbf{female} \cap \mathbf{person}) \cap \mathbf{mother})(a)\}$   
     $= \{((\neg\mathbf{female} \cup \neg\mathbf{person}) \cap \mathbf{female} \cap \mathbf{person} \cap \dots)(a)\}$
  - conjunction rule:  
     $A1 = \{(\neg\mathbf{female} \cup \neg\mathbf{person})(a), \mathbf{female}(a), \mathbf{person}(a), \dots\}$
  - disjunction rule:  
     $A2 = \{(\neg\mathbf{female})(a), (\neg\mathbf{female} \cup \neg\mathbf{person})(a), \mathbf{female}(a), \mathbf{person}(a), \dots\}$
  - clash between  $(\neg\mathbf{female})(a)$  and  $\mathbf{female}(a)$
  - disjunction rule:  
     $A2 = \{(\neg\mathbf{person})(a), (\neg\mathbf{female} \cup \neg\mathbf{person})(a), \mathbf{female}(a), \mathbf{person}(a), \dots\}$
  - clash between  $(\neg\mathbf{person})(a)$  and  $\mathbf{person}(a)$
- The concept  $\neg\mathbf{woman} \cap \mathbf{mother}$  is unsatisfiable  
     $\rightarrow$  **woman** subsumes **mother**

# Concept Classification



- Insert new concept D into existing subsumption hierarchy
- 1. **top-search phase:**
  - Traverse from top
  - **Determine parents of D**  
 $\text{sat}(\neg C_1 \cap D), \dots, \text{sat}(\neg C_n \cap D)$   
(if C3 not parent of D, do not consider children of C3)
- 2. **bottom-search phase:**
  - Traverse from bottom
  - **Determine children of D**  
 $\text{sat}(C_1 \cap \neg D), \dots, \text{sat}(C_n \cap \neg D)$





# DL Systems

- Systems implementing DLs
  - Current:
    - **Racer**:
      - reasoner for ABoxes and concept expressions renamed
      - based on sound and complete algorithms
      - optimized reasoner for ABoxes
    - **FaCT**
    - **Pellet**
  - Old:
    - Classic, LOOM, etc.



# DL Conclusion

- Expressive power determined by:
  - Kinds of constructors provided
  - Kinds of axioms allowed
- DL constructors/axioms restricted so that reasoning is decidable