Advanced Knowledge Based Systems
CS3411

Knowledge Bases in Description Logics

Enrico Franconi

http://www.cs.man.ac.uk/~franconi/teaching/1999/3411/

Understanding Knowledge Bases

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

- **Terminological Axioms:** $C \sqsubseteq D$
- **Assertional Axioms:** $C(a), R(a, b)$

- An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *satisfies* the statement $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
- $\mathcal{I}$ satisfies $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$.
- $\mathcal{I}$ satisfies $R(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

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

TBox statements

(1)	$A \sqsubseteq C$	Primitive concept definition
(2)	$A \doteq C$	Concept definition
(3)	$C \sqsubseteq D$	Concept inclusion
(4)	$C \doteq D$	Concept equation

Acyclic simple TBox

(1)	$A \sqsubseteq C$	Primitive concept definition
(2)	$A \doteq C$	Concept definition

Acyclic TBox: well-founded definitions.

A concept name A *directly uses* a concept name B in a TBox Σ iff the definition of A mentions B. A concept name A *uses* a concept name B_n iff there is a chain of concept names $\langle A, B_1, \ldots, B_n \rangle$ such that B_i directly uses B_{i+1}. A TBox is *acyclic* iff no concept name uses itself.

Acyclic simple TBox

Subsumtpion in acyclic simple TBoxes ($\Sigma \models C \sqsubseteq D$) can be reduced in subsumtpion in an empty TBox ($\models \widehat{C} \sqsubseteq \widehat{D}$).

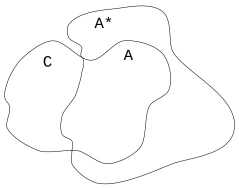
In order to get $\widehat{C}$ (and $\widehat{D}$):

1) Transform the TBox Σ into a new TBox Σ', by replacing every primitive concept definition in Σ of the form $A \sqsubseteq C$ with a concept definition $A \doteq C \sqcap A^*$ – where A^* is a freshly new generated concept name (called *primitive component* of A). Now Σ' contains only (acyclic) concept definitions.

2) Iteratively substitute every occurrence of any defined concept name in C (and D) by the corresponding definition in Σ'. Since Σ' is still acyclic, the process terminates in a finite number of iterations. This process is called *unfolding* or *expansion*.

Theorems

- For every interpretation of Σ there exists an interpretation of Σ' (and viceversa) such that $C^{\mathcal{I}} = C^{\mathcal{I}'}$ for every concept name C in Σ.



$$A \sqsubseteq C$$
$$A \doteq C \sqcap A^*$$

A^* denotes the *unexpressed* part of meaning implicitly contained in the primitive concept definition.

- $\Sigma \models C \sqsubseteq D$ iff $\Sigma' \models C \sqsubseteq D$
- $\Sigma' \models C \sqsubseteq D$ iff $\models \widehat{C} \sqsubseteq \widehat{D}$

Necessary and Sufficient conditions

- A primitive concept definition $A \sqsubseteq C$ states a necessary but not sufficient condition for membership in the class A. Having the property C is necessary for an object in order to be in the class A; however, this condition alone is not sufficient in order to conclude that the object is in the class A.

- A concept definition $A \doteq C$ states necessary and sufficient condition for membership in the class A. Having the property C is necessary for an object in order to be in the class A; moreover, this condition alone is sufficient in order to conclude that the object is in the class A.

Necessary and Sufficient conditions

When transforming primitive concept definitions into concept definitions we get necessary and sufficient conditions for membership in the primitive class A. However, the condition of being in the primitive component A^* can never be satisfied, since the concept name A^* can never be referred to by any other concept.

A concept is subsumed by a primitively defined concept if and only if it refers to its name in its (unfolded) definition.

Inheritance

Unfolding realizes what is usually called *inheritance* in Object-Oriented frameworks.

$$\texttt{Person} \doteq \exists \texttt{NAME}.\texttt{String} \sqcap \exists \texttt{ADDRESS}.\texttt{String}$$

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

$$\widehat{\texttt{Parent}} \doteq \exists \texttt{NAME}.\texttt{String} \sqcap \exists \texttt{ADDRESS}.\texttt{String} \sqcap \exists \texttt{CHILD}.(\exists \texttt{NAME}.\texttt{String} \sqcap \exists \texttt{ADDRESS}.\texttt{String})$$

$$\texttt{Female} \doteq \neg \texttt{Male}$$

$$\texttt{Man} \doteq \texttt{Person} \sqcap \forall \texttt{SEX}.\texttt{Male}$$

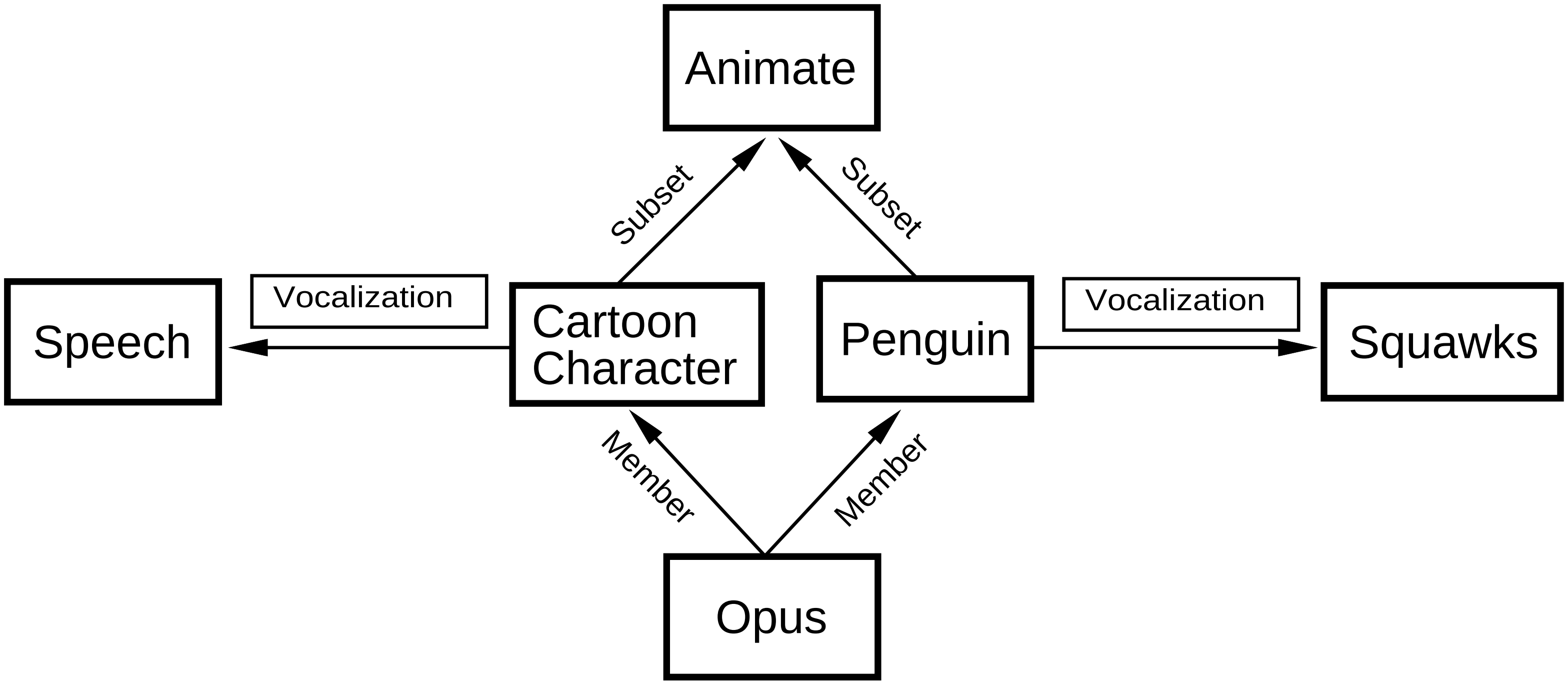
$$\texttt{Woman} \doteq \texttt{Person} \sqcap \forall \texttt{SEX}.\texttt{Female}$$

$$\texttt{Transexual} \doteq \texttt{Man} \sqcap \texttt{Woman}$$

$$\widehat{\texttt{Transexual}} \doteq \exists \texttt{NAME}.\texttt{String} \sqcap \exists \texttt{ADDRESS}.\texttt{String} \sqcap \forall \texttt{SEX}.\bot$$

Inheritance in O-O

Problems in O-O frameworks: overriding strategies for multiple inheritance.



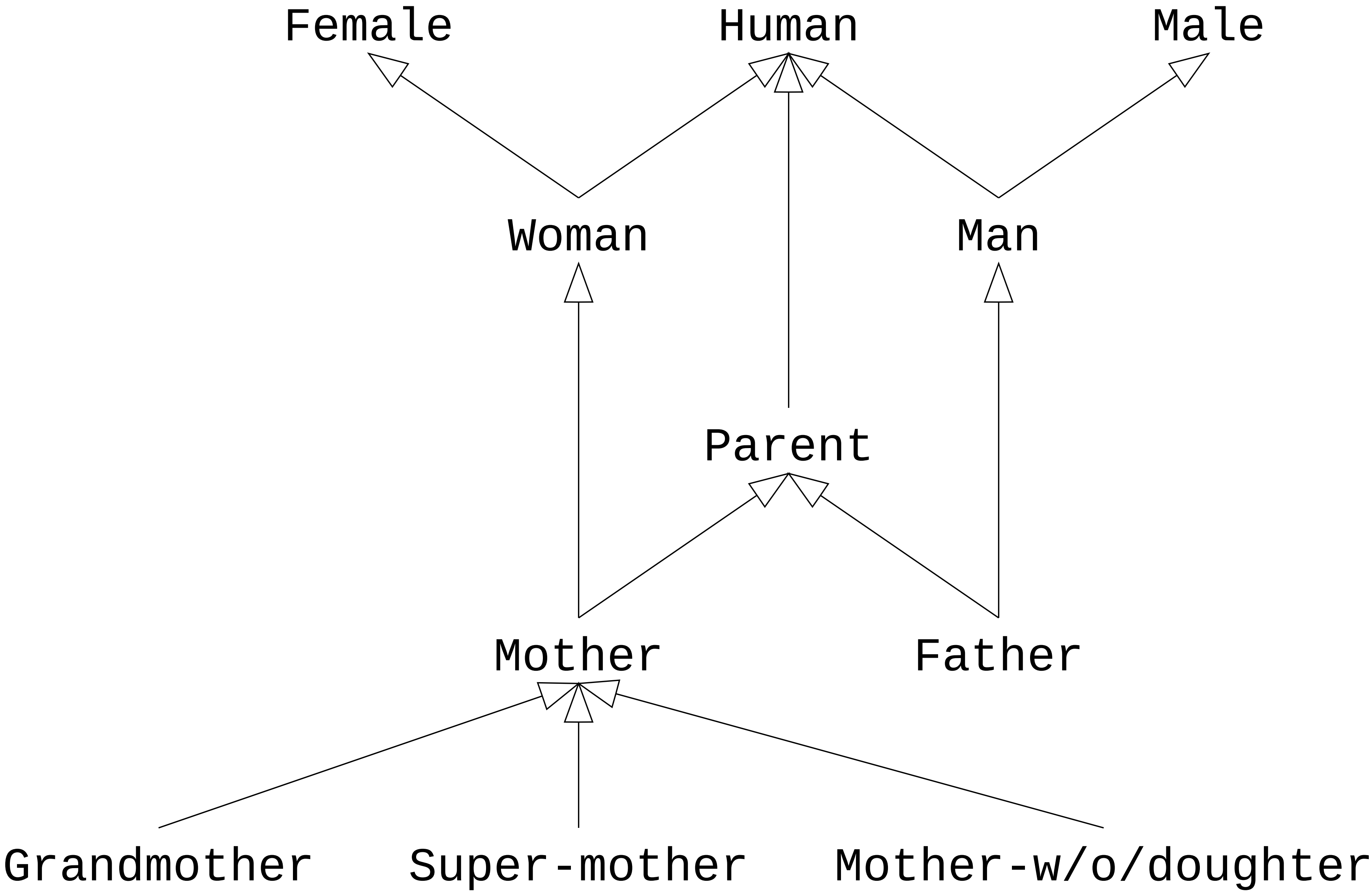
Using Knowledge Bases

The Royal Family

$$
\begin{aligned}
\mathtt{Male} &\doteq \neg\mathtt{Female} \\
\mathtt{Woman} &\doteq \mathtt{Human} \sqcap \mathtt{Female} \\
\mathtt{Man} &\doteq \mathtt{Human} \sqcap \mathtt{Male} \\
\mathtt{Mother} &\doteq \mathtt{Woman} \sqcap \exists\mathtt{CHILD.Human} \\
\mathtt{Father} &\doteq \mathtt{Man} \sqcap \exists\mathtt{CHILD.Human} \\
\mathtt{Parent} &\doteq \mathtt{Father} \sqcup \mathtt{Mother} \\
\mathtt{Grandmother} &\doteq \mathtt{Woman} \sqcap \exists\mathtt{CHILD.Parent} \\
\mathtt{Mother{-}w/o{-}doughter} &\doteq \mathtt{Mother} \sqcap \forall\mathtt{CHILD.Male} \\
\mathtt{Super{-}mother} &\doteq \mathtt{Mother} \sqcap \geqslant 3\mathtt{CHILD}
\end{aligned}
$$

```
Woman(elisabeth), Woman(diana),
Man(charles), Man(edward), Man(andrew),
Mother-w/o-doughter(diana),
CHILD(elisabeth,charles),
CHILD(elisabeth,edward),
CHILD(elisabeth,andrew),
CHILD(diana,william),
CHILD(charles,william)
```

Taxonomy



Female
Human
Male
Woman
Man
Parent
Mother
Father
Grandmother
Super-mother
Mother-w/o/doughter

Questions

- What happens if we add to the knowledge base:

  `CHILD(diana,margaret), Female(margaret)`

- $\Sigma \overset{?}{\models}$ `Super-mother(elisabeth)`
- $\Sigma \overset{?}{\models} \neg$`Female(william)`
- $\Sigma \overset{?}{\models}$ `Mother-w/o-doughter(elisabeth)`

- Which are the most specific concepts of which `elisabeth` is instance (realization problem)?

- Retrieve all the instances of `Male`.

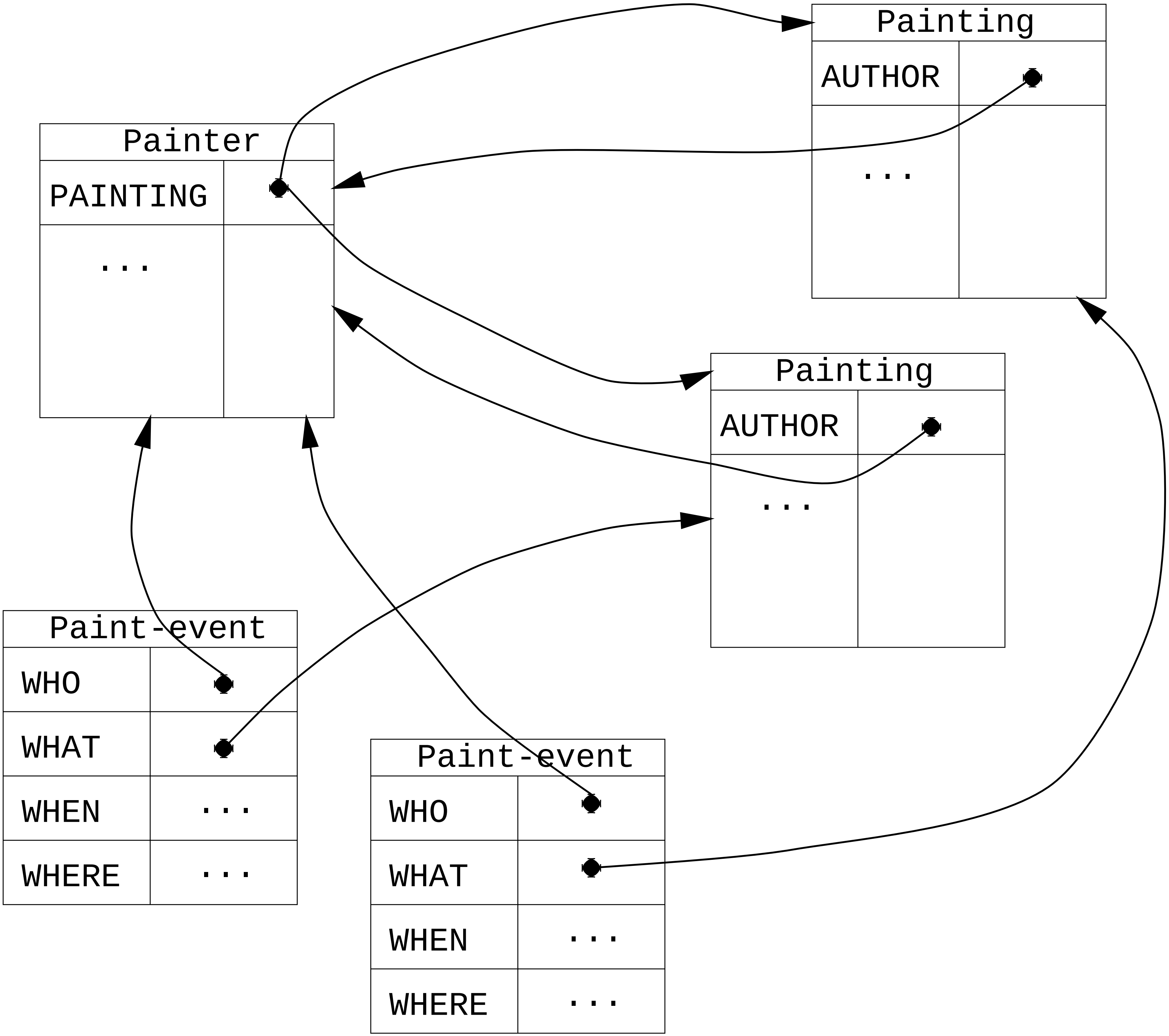
Inclusion Axioms

$$\begin{array}{rcl} \exists\texttt{TEACHES}.\texttt{Course} & \sqsubseteq & (\texttt{Student} \sqcap \exists\texttt{DEGREE}.\texttt{Bs}) \sqcup \texttt{Prof} \\ \texttt{Prof} & \sqsubseteq & \exists\texttt{DEGREE}.\texttt{Ms} \\ \exists\texttt{DEGREE}.\texttt{Ms} & \sqsubseteq & \exists\texttt{DEGREE}.\texttt{Bs} \\ \texttt{Ms} \sqcap \texttt{Bs} & \sqsubseteq & \bot \end{array}$$

TEACHES(john,cs156),
$(\leqslant 1\texttt{DEGREE})$(john),
Course(cs156)

$\Sigma \stackrel{?}{\models}$ Student(john)

Modeling a Museum


Painting
AUTHOR
...
Painter
PAINTING
...
Painting
AUTHOR
...
Paint-event
WHO
WHAT
WHEN
...
WHERE
...
Paint-event
WHO
WHAT
WHEN
...
WHERE
...

Concept as Role

- The painting *The Announcement* of Giotto's is in Florence.
- The painting *The Announcement*, painted by Giotto in 1285 in Venice, is in Florence.

$\texttt{Painter} \sqsubseteq \forall \texttt{PAINTING.Painting}$

$\texttt{Painting} \sqsubseteq \forall \texttt{AUTHOR.Painter}$

$\texttt{PaintEvent} \sqsubseteq \exists \texttt{WHO.Painter} \sqcap \exists \texttt{WHAT.Painting}$

- This TBox forces redundancy.
- This TBox does not reveal inconsistencies.
- This TBox is cyclic.

Redundant KB

$\texttt{Painter} \sqsubseteq \forall \texttt{PAINTING.Painting}$

$\texttt{Painting} \sqsubseteq \forall \texttt{AUTHOR.Painter}$

$\texttt{PaintEvent} \sqsubseteq \exists \texttt{WHO.Painter} \sqcap \exists \texttt{WHAT.Painting}$

```
Painter(giotto),

PAINTING(giotto, announcement),
Painting(announcement),
AUTHOR(announcement, giotto),

PAINTING(giotto, escape),
Painting(escape),
AUTHOR(escape, giotto),

PaintEvent(e1),
WHO(e1, giotto),
WHAT(e1, announcement),

PaintEvent(e2),
WHO(e2, giotto),
```

```
WHAT(e2, escape)
```

Reification

$$\forall xy.\mathtt{PAINTING}(x,y) \leftrightarrow \mathtt{AUTHOR}(y,x) \leftrightarrow \exists z.\mathtt{PaintEvent}(z) \wedge \mathtt{WHO}(z,x) \wedge \mathtt{WHAT}(z,y)$$

$$\mathtt{PaintEvent} \sqsubseteq \exists \mathtt{WHO}. \sqcap \leqslant 1 \mathtt{WHO} \sqcap \exists \mathtt{WHAT}. \sqcap \leqslant 1 \mathtt{WHAT}$$

$$\mathtt{Painter} \doteq \exists \mathtt{WHO}^{-}.\mathtt{PaintEvent}$$

$$\mathtt{Painting} \doteq \exists \mathtt{WHAT}^{-}.\mathtt{PaintEvent}$$

$$\mathtt{PAINTING} \doteq \mathtt{WHO}^{-}|_{\mathtt{PaintEvent}} \circ \mathtt{WHAT}$$

$$\mathtt{AUTHOR} \doteq \mathtt{WHAT}^{-}|_{\mathtt{PaintEvent}} \circ \mathtt{WHO}$$

```
PaintEvent(e1),
WHO(e1, giotto),
WHAT(e1, announcement)
```

$\models$

```
Painter(giotto),
PAINTING(giotto, announcement),
Painting(announcement),
AUTHOR(announcement, giotto)
```

```
PAINTING(giotto, announcement)

     Painter(giotto),
|=   Painting(announcement),
     AUTHOR(announcement, giotto)
```

$$\models \quad \begin{array}{l}\texttt{Painter(giotto),}\\ \texttt{Painting(announcement),}\\ \texttt{AUTHOR(announcement, giotto)}\end{array}$$

Building Real Knowledge Bases

Building Knowledge Bases

In order to build good KBs some choices must be done during its design. It is important to well understand some subtle distinctions:

- Primitive vs. Defined.
- Definitional vs. Incidental.
- Concept vs. Individual.
- Concept vs. Role.

When to Use Primitive Concepts?

- some concepts can not be completely defined (e.g. natural kinds);
- it can be not convenient/useful to completely define a concept;
- sooner or later we must end up with something not completely defined (*encyclopedic knowledge* cannot be given).

Thus, primitive concepts must be used when:

- there is no other way;
- even if it were defined, no (automatic) classification below it will be never required by the application.

Typically primitive concepts lie in the top region of the taxonomy.

When to Use Defined Concepts?

- ontological reason: it is easy and natural (in the context of the application) to give a complete definition of the concept;
- organization of the antecedents of rules;
- capturing complete descriptions used by rules for populating primitive concepts.

Definitional vs. Incidental

Are *incidental* all the properties that are contingent features for a concept, and thus must not be part of its definition.

Examples:

$(\forall \texttt{SUGAR}.\texttt{Dry})$

is incidental for the concept RED-BORDEAUX-WINE, while

$(\forall \texttt{COLOR}.\texttt{Red})$ and $(\forall \texttt{REGION}.\texttt{Bordeaux})$

are not.

$(\forall \texttt{INTELLIGENCE}.\texttt{Stupid})$

is incidental for CHICKEN, while

$(\forall \texttt{REPRODUCES}-\texttt{WITH}.\texttt{Egg})$

is not.

Concept vs. Individual

- the set of individuals is a countable, discrete set;
- the concept space is ideally continuous and infinite;
- each individual has a clear identity: even if two individual have the same properties, they are distinct;
- if two concepts have equivalent descriptions, they **denote the same** concept;
- individual descriptions can be modified;
- concept descriptions can not be modified;
- individual update does not (usually) change the concept hierarchy;
- rules applies only to individuals.

Nevertheless, it is not always easy to decide whether an object should be a concept or an individual. The main issue to deal with is the “granularity” level.

Example:

Consider the KB describing courses in a Computer Science department : is “Introduction To Data Structures And Algorithms (503)” course a concept or an individual?

Individual vs. Concept

Another example: if we have `Wine` and `White-wine`, what about:

- chardonay-wine
- forman-chardonay
- 1981-forman-chardonay
- 1981-forman-chardonay-from-vineyard32
- 1981-forman-chardonay-from-in-cask18
- 1981-forman-chardonay-bottle#1576

A key to solve the problem could be asking the domain/application expert: "how many wines do you have?", in order to understand the needed granularity.

Concept vs. Role

Is not always easy to decide what must be a concept and what a role.

E.g.:

- PERSON: it is a concept.
- MOTHER:
  - consider "Sue is a new mother" and "Sue is the mother of Tom"
  - `Mother` as a concept does not exist if we don't consider the "role she plays" in a parental relation, i.e., if "Sue is a new mother" she must be the `MOTHER` of somebody!

Thus:

$$\texttt{Mother} \doteq (\texttt{Woman} \sqcap (\exists \texttt{MOTHER}.\texttt{Person}))$$

- But when the role is not the only important component of the definition, this dual use is not so neat (consider `VINTAGE`, `GRAPE`).
- Another problem is the *reading direction*: in the above example the role could be, `MOTHER` or `CHILD`.
- A clear convention must be stated, possibly creating long, non ambiguous names for roles, as, e.g.: `HAS-CHILD, IS-THE-PARENT-OF, HAS-VINTAGE, HAS-GRAPE`.
- As an alternative, the adoption of long names for concepts can be also suggested: e.g., `WINE-GRAPE`.

How to design a KB in 12 steps

1. **Enumerate Object**. As a bare list of elements of the KB; they will became individuals, concepts, or role.

2. **Distinguish Concepts from Roles**. Make a first decision about what object must be considered role; remember that some could have a "natural" concept associated. The remaining objects will be concepts (or maybe individuals). Also, try to distinguish roles from attributes.

3. **Develop Concept Taxonomy**. Try to decide a classification of all the concepts, imagining their extensions. This taxonomy will be used as a first reference, and could be revised when definition will be given. It will be used also to check if definition meet our expectations (sometime, interesting, unforeseen (re)classifications are found).

4. **Devise partitions**. Try to make explicit all the disjointness and covering constraints among classes, and reclassify the concepts.

5. **Individuals**. Try to list as many as possible *generally* useful individuals . Some could have been already listed in step 1. Try to describe them (classify).

6. **Properties and Parts**. Begin to define the internal structure of concepts (this process

will continue in the next steps). For each concept list:

- *intrinsic* properties, that are part of the very nature of the concept;
- *extrinsic* properties, that are contingent or external properties of the object; they can sometime change during the time;
- *parts*, in the case of structured or collective objects. They can be physical (e.g., “the components of a car”, “the casks of a winery”, “the students of a class”, “the members of a group”, *“the grape of a wine”*) or abstract (e.g., *“the courses of a meal”*, “the lessons of a course”, *“the topics of a lesson”*).

In some cases some relationships between individuals of classes can be considered too accidental to be listed above (e.g., “the employees of a winery”; but the matter could change if we consider `Winery` as a subconcept of `Firm`.

In general, the above distinctions depend on the level of detail adopted.

Some of the listed roles will be later considered definitional, and some incidental.

After this and the next steps check/revision of step 3 could be necessary.

7. **Cardinality Restrictions.** For the relevant roles for each concept.
8. **Value Restriction.** As above. Also, chose the right restriction.
9. **Propagate Value Restrictions.** If some value restrictions stated in the previous step

does not correspond to already existing concepts, they must be defined.

10. **Inter-role Relationship.** Even if hardly definable in DL, they can be useful during the populating and debugging phases.

11. **Definitional and Incidental.** It is important distinguish between definitional and incidental properties, w.r.t. to the particular application.

12. **Primitive and Defined.** As above.

Basic Reference

- Brachman, R., McGuinness, D., Patel-Schneider, P., Resnick, L., Borgida, A.. Living with CLASSIC: When and how to use a KL-ONE-like language. In: Principles of Semantic Networks, edited by John Sowa, Morgan Kaufmann, 1991