

Terminological knowledge



Prof. Mauro Gaspari

Dipartimento di Informatica Scienze e Ingegneria
(DISI)

mauro.gaspari@unibo.it

Categories and Objects



- **The organization of objects into categories** is a vital part of knowledge representation.
- There is a large amount of reasoning that takes place at the level of categories.
- **Categories** can be represented in FOL:
 - Predicates: Professor(Mauro)
 - Objects (reification): Member(Mauro,Professor)
- Categories are useful for organizing the KB through **inheritance**, using a **subclass relationship** (semantics: set inclusion).
- Categories are organized as a hierarchy, **taxonomy**.

Definitions



- First-order logic makes it easy to state facts about categories, by:
 - **Instance relationship** relating objects to categories.
 - **Subclass relationship.**
 - **Quantifying over their members.**
- Moreover, other properties can be expressed:
 - We say that two or more categories are **disjoint** if they have no members in common.
 - A disjoint exhaustive decomposition is known as a **partition.**
- However, FOL does not support specific and efficient reasoning mechanisms for these operations.

Reasoning with Categories



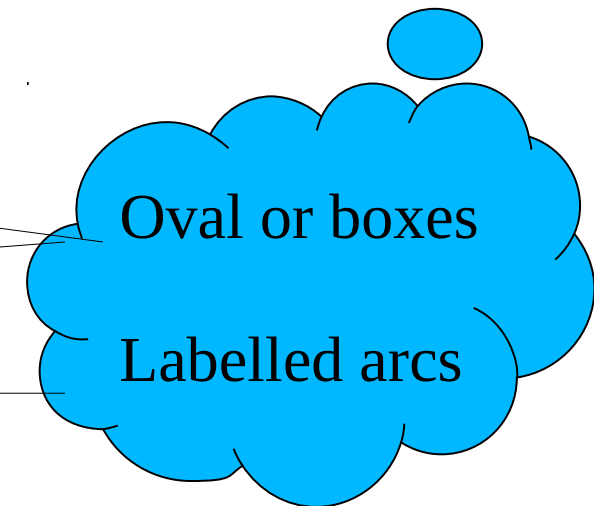
- Categories are the primary building blocks of any large scale knowledge based system.
- Categories are used for representing terminology.
- There are two closely related families of systems:
 - **Semantic Networks:** provide graphical aids for visualizing hierarchies of categories and efficient algorithms for inferring properties of objects.
 - **Description logic:** provides a formal language for constructing and combining category definitions and efficient algorithm for deciding subset and superset relationship.

Semantic Networks



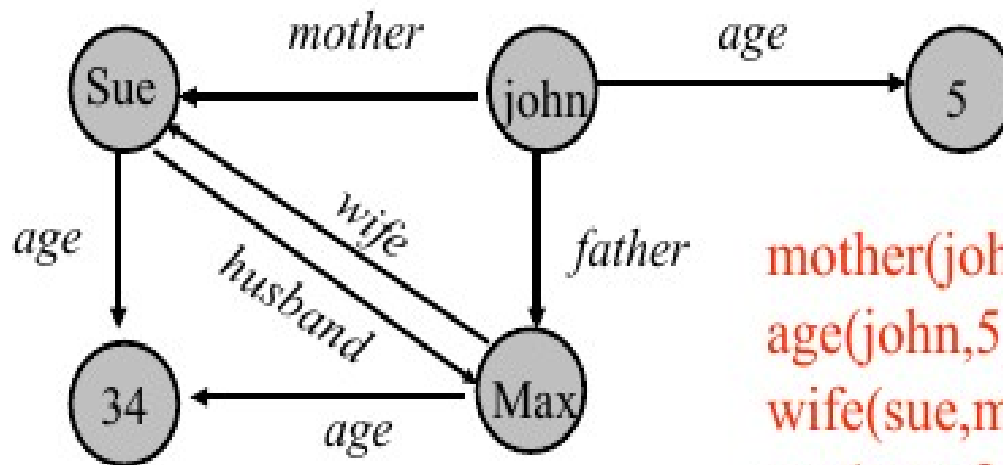
- In 1909 Charles Peirce proposed a graphical notation called existential graphs that he called "the logic of the future."
- Thus began a long-running debate between advocates of "logic" and advocates of "semantic networks", that obscured the underlying unity of the field (well-defined semantic networks are a form of logic).
- There are many variants of Semantic Networks, all of them represents:

- Individual Objects:
- Categories of Objects:
- Relations among objects:





Example



`mother(john,sue)`

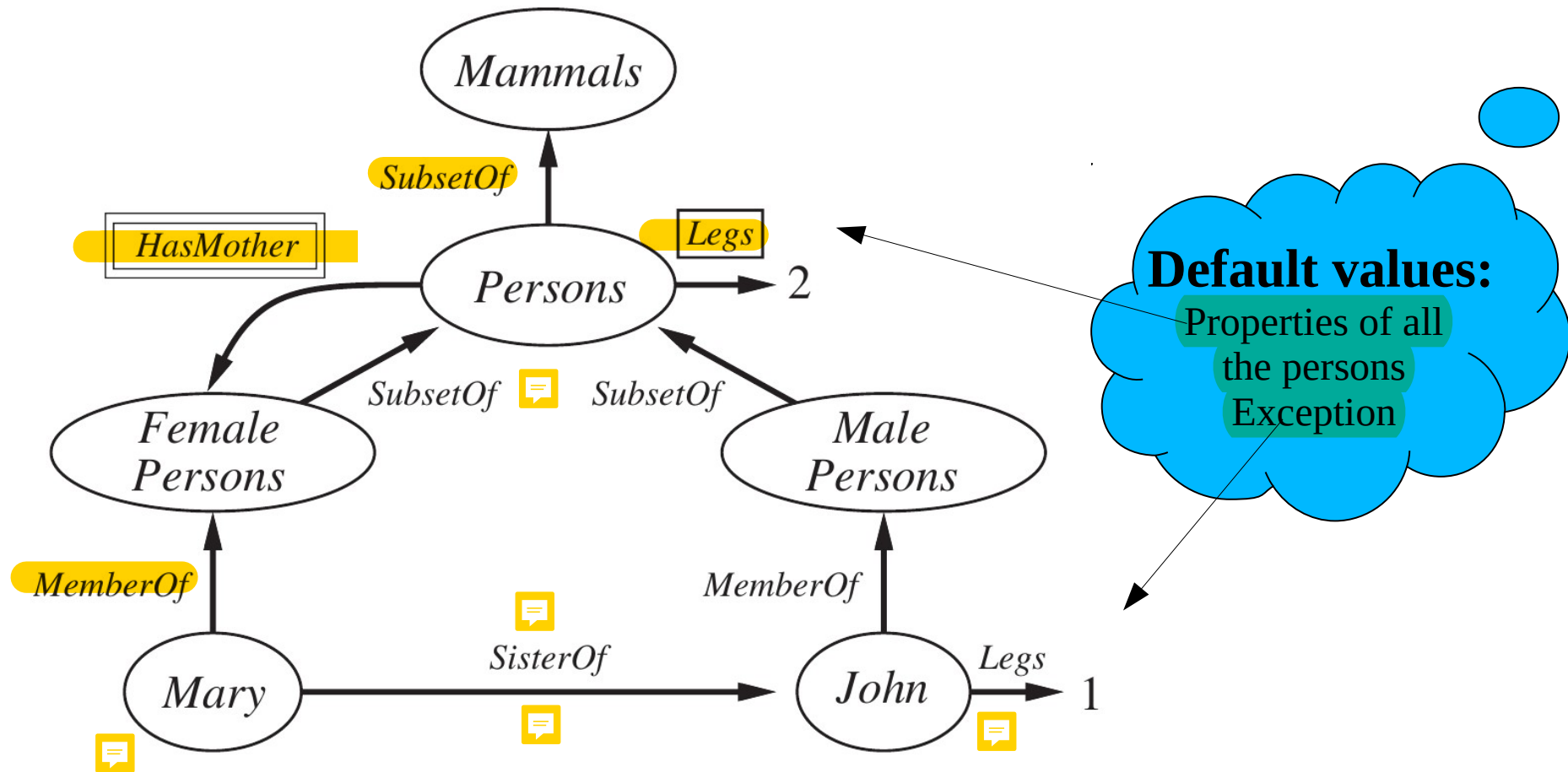
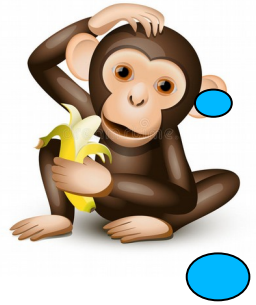
`age(john,5)`

`wife(sue,max)`

`age(max,34)`

... 

Example





Observations



- The MemberOf between Mary and FemalePersons corresponds to the logical assertion: $\text{Type}(\text{Mary}, \text{FemalePerson})$.
- Similarly the SisterOf link corresponds to: $\text{SisterOf}(\text{Mary}, \text{John})$.
- HasMother is represented as a double box because it is not a relation between the two categories, but between the elements of categories:
$$\forall x \text{ Type}(x, \text{Person}) \rightarrow [\forall y \text{ HasMother}(x, y) \rightarrow \text{Type}(y, \text{FemalePersons})]$$
- A single box is used for a property of all the elements of a category.
$$\forall x \text{ Type}(x, \text{Person}) \rightarrow \text{Legs}(x, 2)$$

Link Types



Link Type	Semantics	Example
$A \xrightarrow{\text{Subset}} B$	$A \subset B$	$Cats \subset Mammals$
$A \xrightarrow{\text{Member}} B$	$A \in B$	$Bill \in Cats$
$A \xrightarrow{R} B$	$R(A, B)$	$Bill \xrightarrow{Age} 12$
$A \xrightarrow{\boxed{R}} B$	$\forall x \ x \in A \Rightarrow R(x, B)$	$Birds \xrightarrow{\boxed{Legs}} 2$
$A \xrightarrow{\boxed{\boxed{R}}} B$	$\forall x \ \exists y \ x \in A \Rightarrow y \in B \wedge R(x, y)$	$Birds \xrightarrow{\boxed{\boxed{Parent}}} Birds$

Here we use \Rightarrow in place of \rightarrow

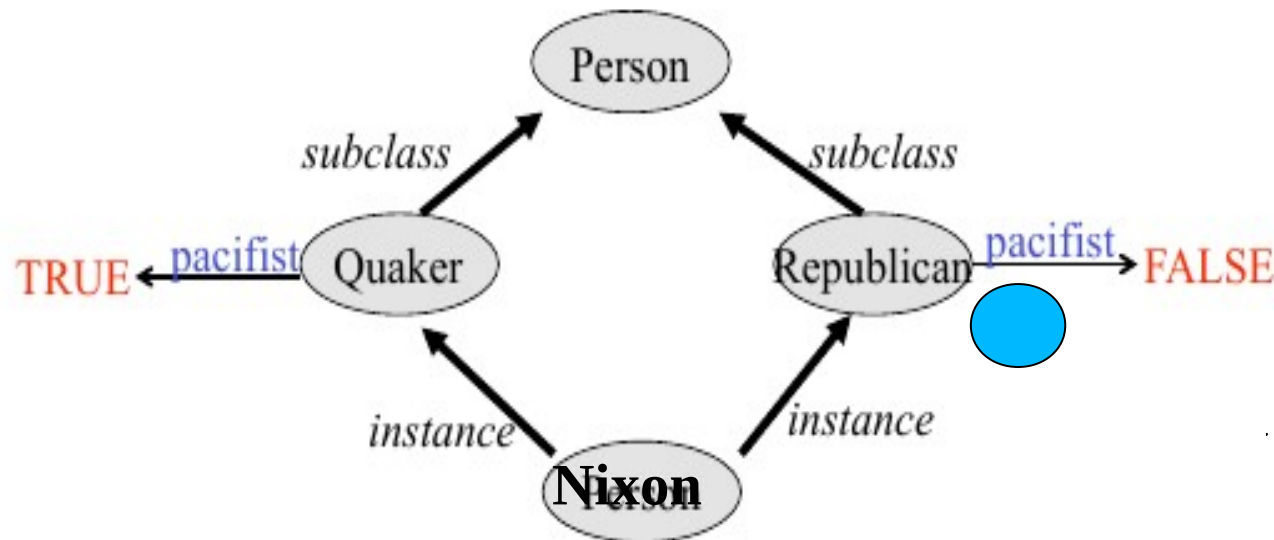
Taxonomic reasoning




- One of the main kinds of reasoning done in a semantic net is the inheritance of values following Subclass and MemberOf links.
- Semantic networks differ in how they handle the case of inheriting multiple different values:
 - All possible values are inherited
 - Only the “lowest” value or values are inherited



Multiple Inheritance



Here Taxonomic reasoning should report the conflict. 

I am a pacifist!

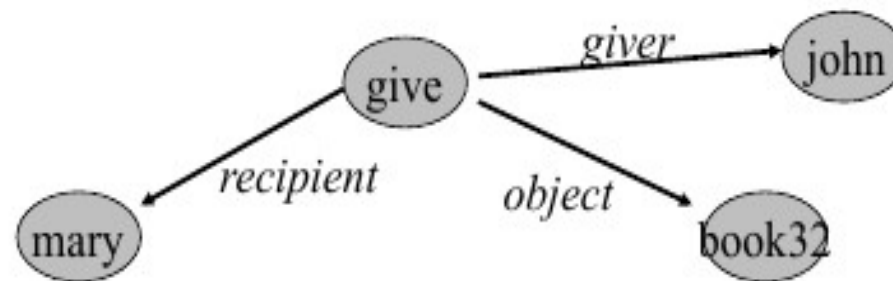
What about Nixon?

Is Trump a Quacker?

Reification



- Semantic Networks only support binary relationships.
- Non-binary relationships can be represented by **reification**: turning a proposition into an object.
- A generic give event involving three things: a giver, a recipient and an object: `give(john,mary,book32)`, can be represented as follows:



Inverse Link Reasoning



- HasSister is the inverse of SisterOf:
 $\forall x,y \text{ HasSister}(x,y) \leftrightarrow \text{SisterOf}(y,x)$
- However this information is not encoded in the semantic network. Thus taxonomic reasoning is not able to answer in an efficient way if HasSister for John holds. To answer the algorithm should consider all the female persons to see if one of them is linked to John with SisterOf.
- This problem can be solved by reification adding the following nodes in the network:



Observations

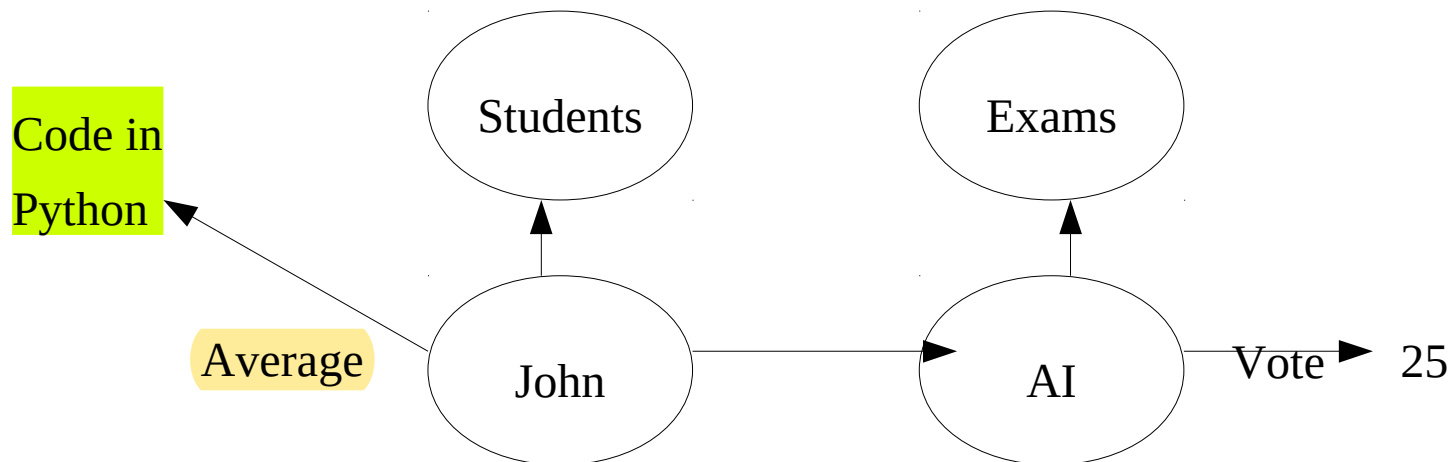


- Using reification we can represent every ground atomic sentence in FOL (without functional symbols).
- Some universally quantified sentences can be encoded with the box, double box, or inverse link.
- But the expressive power is still not that of FOL.
- What is missing?
 - Negation;
 - Disjunction
 - Nested function symbols
 - Existential quantification

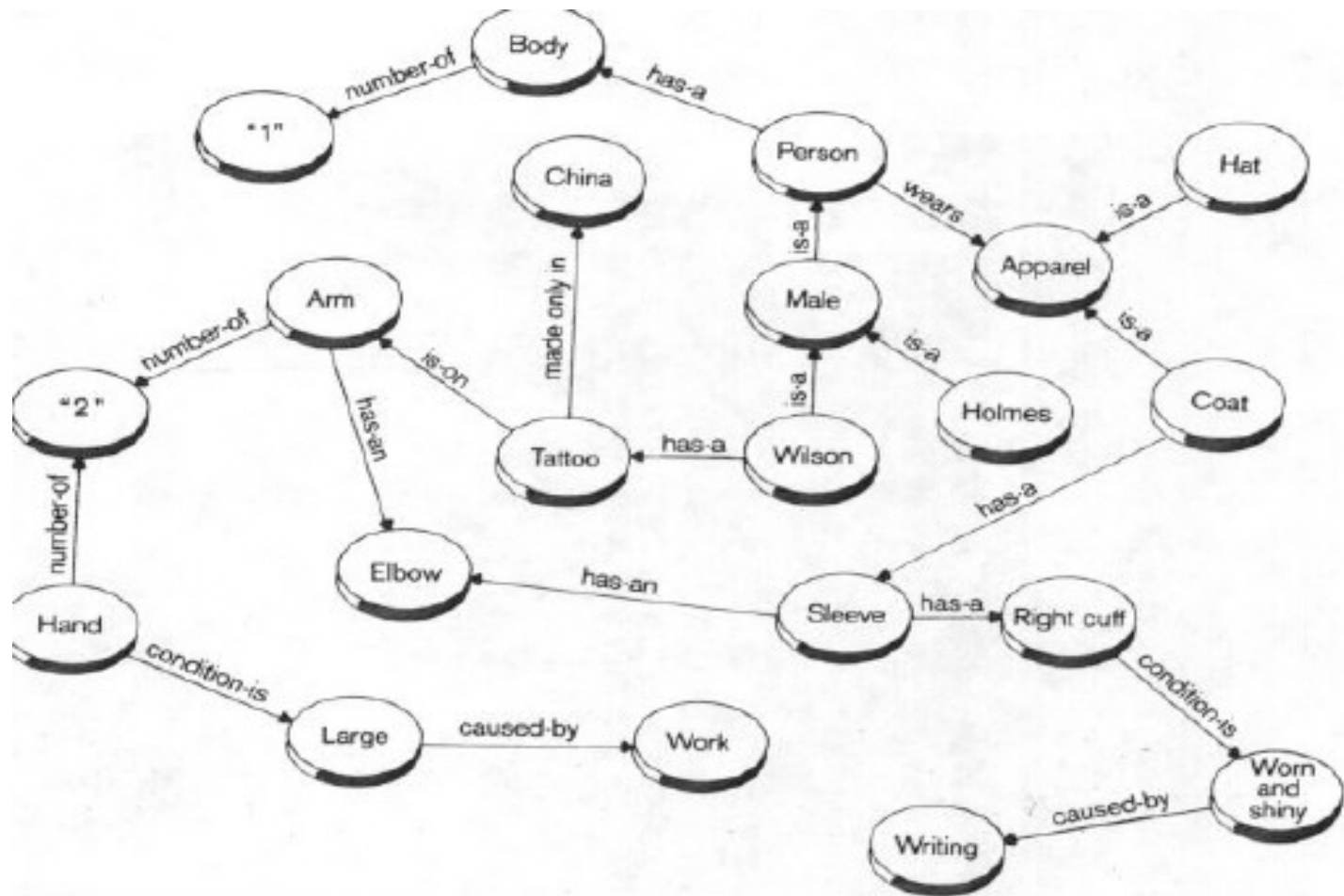
Procedural Attachment



- In principle it would be possible to extend the notation adding more expressive power to the network, however, in this way the simplicity (efficiency) and transparency of the reasoning process is lost.
- To maintain efficiency, in cases where the expressive power of the network is limited **procedural attachment** can be used.
- Using procedural attachment a call to a certain relationship results in a call to a specific procedure implemented in an high level language.



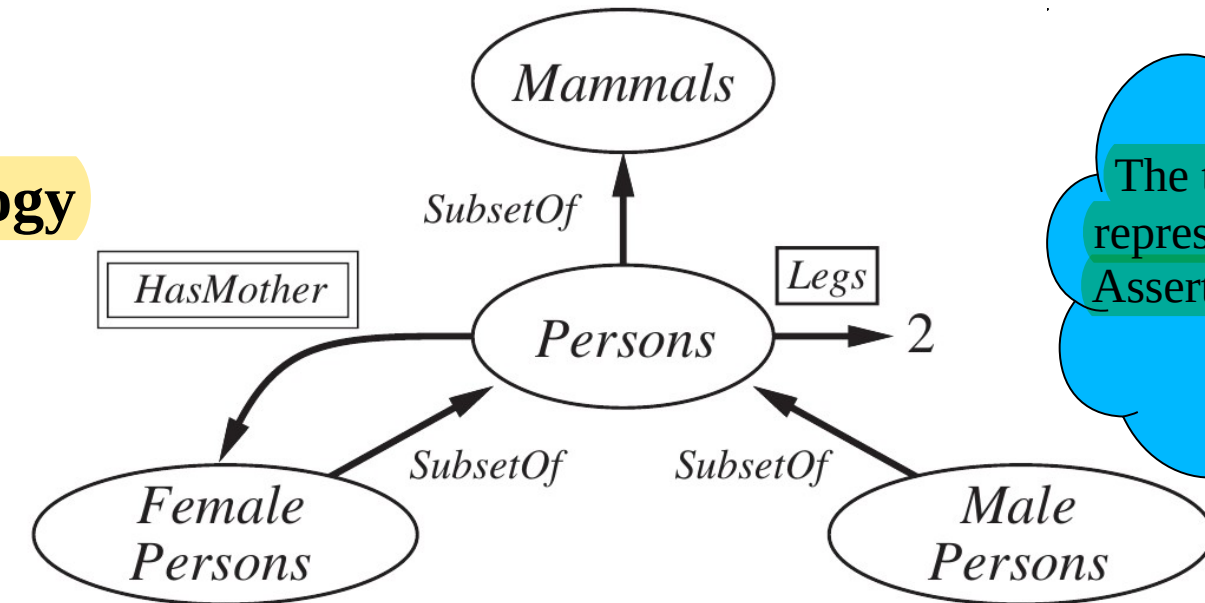
Scaling up



Terminology and Assertions



Terminology



Assertions



Description Logics



- **Description logics (DL):** are notations that are designed to make it easier to describe definitions and properties of categories.
- The objective of DL is formalize what a network means and studying the reasoning mechanisms.
- The principal inference tasks for DL are:
 - **Subsumption:** checking if one category is a subset of another comparing their definitions.
 - **Classification:** checking whether an object belongs to a category.
 - **Consistency** of a category definition: whether the membership criteria are logically satisfiable.

Features of DL



- Describe a domain in terms of **concepts** (categories), **roles** (properties, relationships) and **individuals** (objects).
- Syntax and Formal semantics (model based semantics).
 - Decidable fragments of FOL
 - Closely related to Propositional Modal, Hybrid & Dynamic Logics
- Efficient inference:
 - Decision procedures for key problems (satisfiability, subsumption, classification etc).
 - Highly optimised tools.




The DL Family



- The first DL was **KL-ONE** development as an attempt to design a language able to encode semantic networks.
- First incomplete systems: Back, **Classic**, **Loom**, **Omega** ...
 - Based on structural algorithms
 - Some of them like Classic and Looms were Lisp Based.
- Development of tableau algorithms and complexity results trying to find out the better compromise between expressive power and efficiency.
- DL family: a family of languages has been developed for studying DL. Simplest logic in this family is named **AL**.

AL






- ~~ALC~~ Syntax: 
 - Atomic Concepts: Person, Doctor, HappyParent, 
 - Roles (relationships): hasChild, loves
 - Individuals (nominals): John, Mary, Italy
- Concepts constructed using boolean operators:
 \sqcup, \neg

- Restricted quantifiers:
 \exists, \forall

AL Syntax




- Concepts:

C, D	\longrightarrow	A		(atomic concept)
		\top		(universal concept) 
		\perp		(bottom concept) 
		$\neg A$		(atomic negation)
		$C \sqcap D$		(intersection)
		$\forall R.C$		(value restriction)
		$\exists R.\top$		(limited existential quantification).
				

Examples:



- Person and Female are atomic concepts:
- $\text{Person} \sqcap \text{Female} \Rightarrow$ those persons that are female
- $\text{Person} \sqcap \neg \text{Female} \Rightarrow$ those persons that are not female.
- hasChild is an atomic role:
- $\text{Person} \sqcap \exists \text{hasChild} \Rightarrow$ those persons that have a child
- $\text{Person} \sqcap \forall \text{hasChild.Female} \Rightarrow$ those persons all of whose children are female.
- $\text{Person} \sqcap \forall \text{hasChild}.\perp \Rightarrow$ those persons without a child . 

Semantic of AL



- We consider interpretations I that consist of a non-empty set Δ^I (the domain of the interpretation) and an interpretation function, which assigns to every atomic concept A a set $A^I \subseteq \Delta^I$ and to every atomic role R a binary relation $R^I \subseteq \Delta^I \times \Delta^I$.
- The interpretation function is defined inductively as follows:

$$\begin{aligned}\top^I &= \Delta^I \\ \perp^I &= \emptyset \\ (\neg A)^I &= \Delta^I \setminus A^I \\ (C \sqcap D)^I &= C^I \cap D^I \\ (\forall R.C)^I &= \{a \in \Delta^I \mid \forall b. (a, b) \in R^I \rightarrow b \in C^I\} \\ (\exists R.\top)^I &= \{a \in \Delta^I \mid \exists b. (a, b) \in R^I\}.\end{aligned}$$

Equivalence Definitions



- We say that two concepts C , D are **equivalent**, and write $C \equiv D$, if $C^I = D^I$ for all interpretations I .
- For example:

$$\forall \text{hasChild.Female} \sqcap \forall \text{hasChild.Student} \equiv \forall \text{hasChild.}(\text{Female} \sqcap \text{Student})$$

AL vs ALC



- The basic description language AL is not expressive enough, indeed it does not support disjunction and provides limited forms of negation and existential quantifier only.
- Thus we start introducing an extension named ALC which has not the above limitations and has a more practical impact.
- ALC means Attributive (Concept) Language with Complements, in this language, unlike AL, the complement of any concept is allowed, not only the complement of atomic concepts.
- ALC is sufficiently expressive to support many fundamental constructors for representing terminological knowledge.
- ALC is a syntactic variant of the propositional modal logic K,

ALC



- ALC Syntax:
 - Atomic Concepts: Person, Doctor, HappyParent,
 - Roles (relationships): hasChild, loves
 - Individuals (nominals): John, Mary, Italy
- Concepts constructed using boolean operators:
 \sqcap, \sqcup, \neg
- Restricted quantifiers:
 \exists, \forall

ALC



Atomic types: concept names A, B, \dots (unary predicates)
role names R, S, \dots (binary predicates)

Constructors:

- $\neg C$ (negation)
- $C \sqcap D$ (conjunction)
- $C \sqcup D$ (disjunction)
- $\exists R.C$ (existential restriction)
- $\forall R.C$ (universal restriction)

For example: $\neg(A \sqcup \exists R.(\forall S.B \sqcap \neg A))$

Mammal $\sqcap \exists \text{bodypart.trunk} \sqcap \forall \text{color.Grey}$ 

Semantics



Semantics based on **interpretations** $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\cdot^{\mathcal{I}}$ maps

- each concept name A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$.
- each role name R to a binary relation $R^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$.

Semantics of complex concepts:

$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \quad (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \quad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$

$$(\exists R.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid (d, e) \in R^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I}}\}$$

$$(\forall R.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid (d, e) \in R^{\mathcal{I}} \text{ implies } e \in C^{\mathcal{I}}\}$$

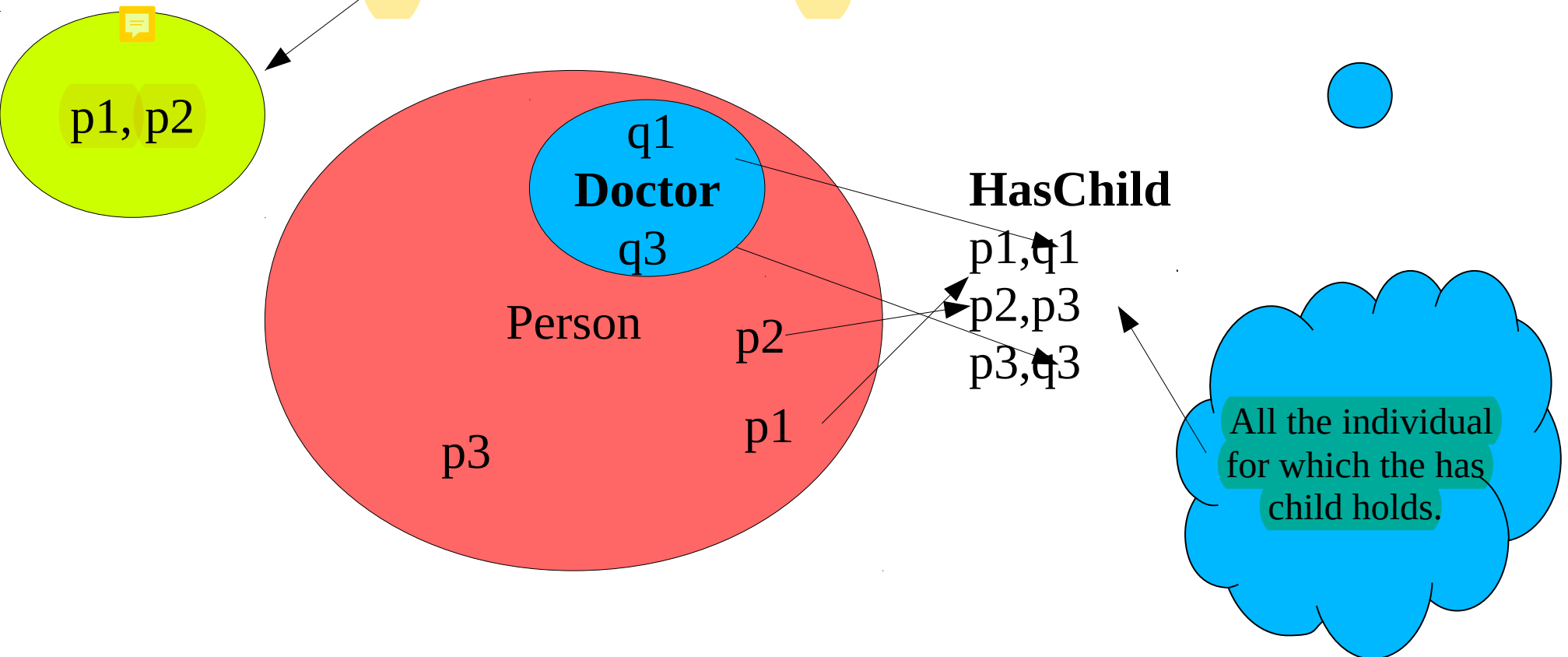
An interpretation \mathcal{I} is a **model** for a concept C if $C^{\mathcal{I}} \neq \emptyset$.



Example

- The category of Person whose children are either Doctors or have a child who is a Doctor:

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



DL Knowledge Bases



- A **TBox** is a set of “schema” axioms (sentences), e.g.:

$\{\text{Doctor} \sqsubseteq \text{Person},$
 $\text{HappyParent} \equiv \text{Person} \sqcap \forall \text{hasChild} . (\text{Doctor} \sqcup \exists \text{hasChild} . \text{Doctor})\}$

– i.e., a **background theory** (a set of **non-logical axioms**)

- An **ABox** is a set of “data” axioms (ground facts), e.g.:

$\{\text{John} : \text{HappyParent},$
 $\text{John hasChild Mary}\}$

– i.e., non-logical axioms including (restricted) use of nominals

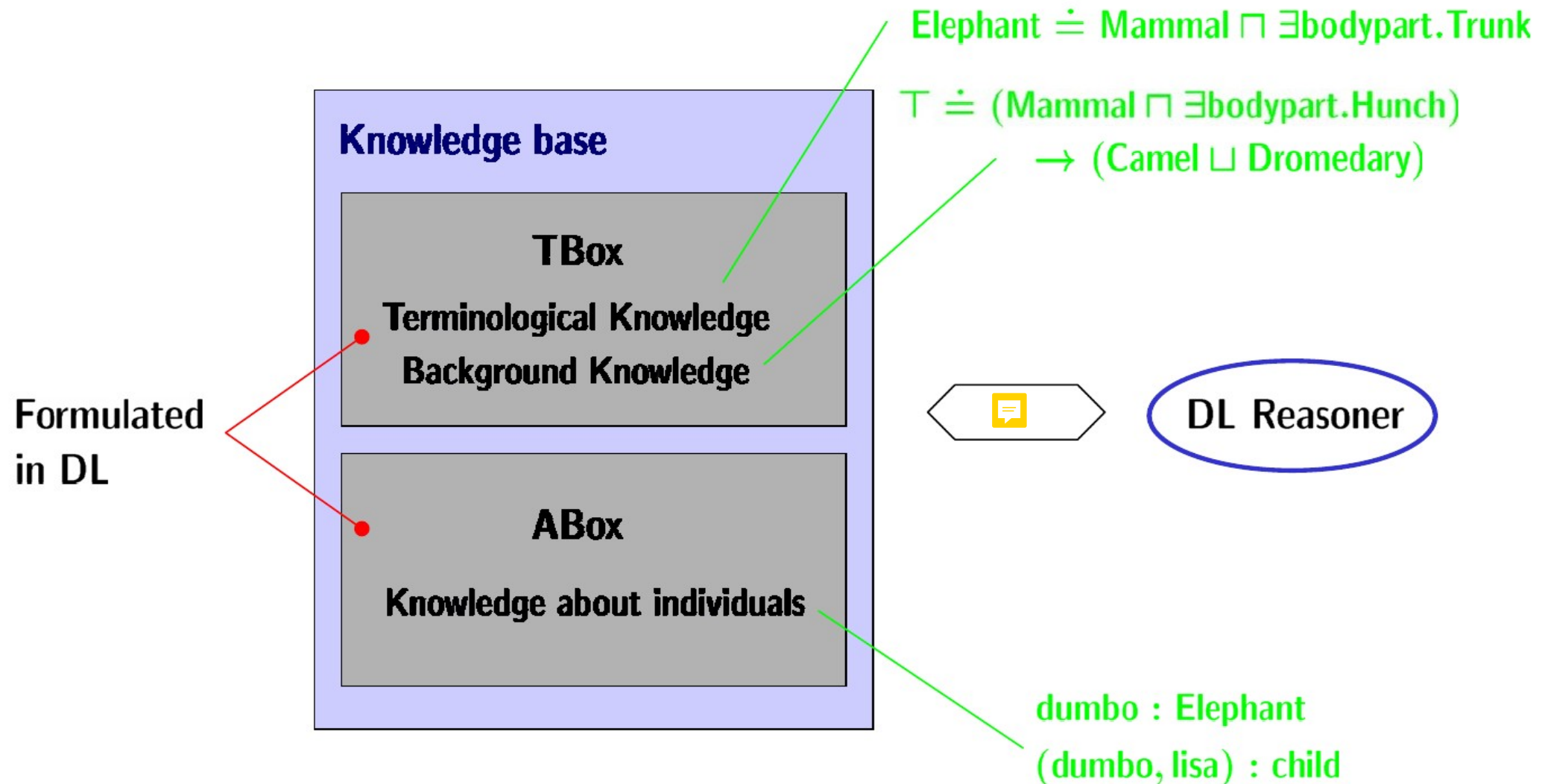
Tbox ALC Example



- Concepts about family relationships:

Woman	≡	Person \sqcap Female
Man	≡	Person \sqcap \neg Woman
Mother	≡	Woman \sqcap \exists hasChild.Person
Father	≡	Man \sqcap \exists hasChild.Person
Parent	≡	Father \sqcup Mother
Grandmother	≡	Mother \sqcap \exists hasChild.Parent
MotherWithoutDaughter	≡	Mother \sqcap \forall hasChild. \neg Woman
Wife	≡	Woman \sqcap \exists hasHusband.Man

DL Knowledge Base



Reasoning in DL

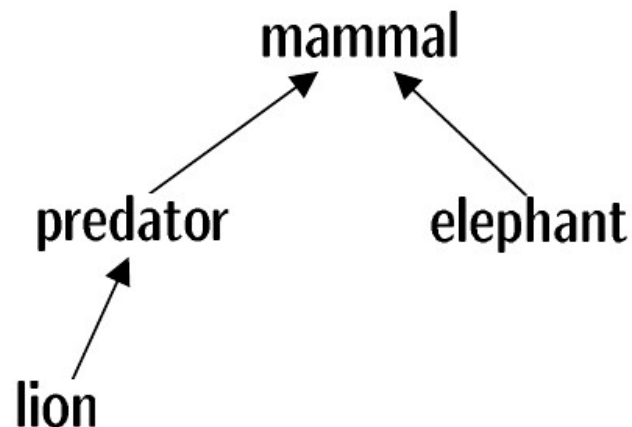


Two main reasoning tasks:

1. **Concept satisfiability** — does there exist a model of C ?
2. **Concept subsumption** — does $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ hold for all \mathcal{I} ?
(written $C \sqsubseteq D$)

Why subsumption?

⇒ Can be used to compute a concept hierarchy:



Extending ALC



In many cases, the expressive power of \mathcal{ALC} does not suffice:

- an elephant has precisely four legs
- every elephant has a bodypart which is a trunk and every trunk is a bodypart of an elephant

Many extensions of \mathcal{ALC} have been developed, for example:

- qualified number restrictions ($\leq n R C$) and ($\geq n R C$)
- inverse roles R^- to be used in existential and universal restriction

But: Increasing expressivity also increases computational complexity

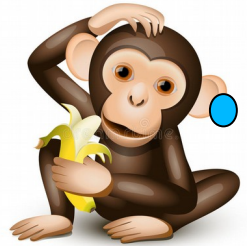
\Rightarrow **!! tradeoff between expressivity and computational complexity !!**

ALC extensions



- **S**: often used for ALC extended with transitive roles: e.g.,
transitive(Anccestor)
- Additional letters indicate other extensions:
 - **H** for **role hierarchy** (e.g., hasDaughter \sqsubseteq hasChild)
 - **O** for **nominals/singleton classes** (e.g., {Italy})
 - **I** for **inverse roles** (e.g., inverse(HasSister, SisterOf))
 - **Q** for **qualified number restrictions**: e.g.,
 ≤ 2 hasChild.Female
 ≥ 1 hasParent.Male
 - **N** for **number restrictions** (e.g., ≤ 2 hasChild.T)

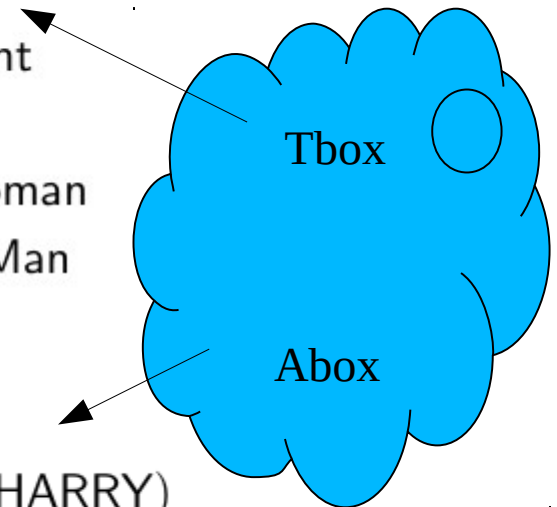
Example



Woman	\equiv	Person \sqcap Female
Man	\equiv	Person $\sqcap \neg$ Woman
Mother	\equiv	Woman $\sqcap \exists$ hasChild.Person
Father	\equiv	Man $\sqcap \exists$ hasChild.Person
Parent	\equiv	Father \sqcup Mother
Grandmother	\equiv	Mother $\sqcap \exists$ hasChild.Parent
MotherWithManyChildren	\equiv	Mother $\sqcap \geq 3$ hasChild
MotherWithoutDaughter	\equiv	Mother $\sqcap \forall$ hasChild. \neg Woman
Wife	\equiv	Woman $\sqcap \exists$ hasHusband.Man

MotherWithoutDaughter(MARY)
hasChild(MARY, PETER)
hasChild(MARY, PAUL)

Father(PETER)
hasChild(PETER, HARRY)



DL as a fragment of FOL



concept names A	\iff	unary predicates P_A
role names R	\iff	binary predicates P_R
concepts	\iff	formulas with one free variable

$$\begin{aligned}\varphi^x(A) &= P_A(x) \\ \varphi^x(\neg C) &= \neg \varphi^x(C) \\ \varphi^x(C \sqcap D) &= \varphi^x(C) \wedge \varphi^x(D) \\ \varphi^x(C \sqcup D) &= \varphi^x(C) \vee \varphi^x(D) \\ \varphi^x(\exists R.C) &= \exists y. P_R(x, y) \wedge \varphi^y(C) \\ \varphi^x(\forall R.C) &= \forall y. P_R(x, y) \rightarrow \varphi^y(C)\end{aligned}$$

φ^y symmetric
with x and y exchanged

- Note:
- two variables suffices (no "=", no constants, no function symbols)
 - formulas obtained by translation have "guarded" structure
 - not all DLs are purely first-order (transitive closure, etc.)

Lisp syntax for DL

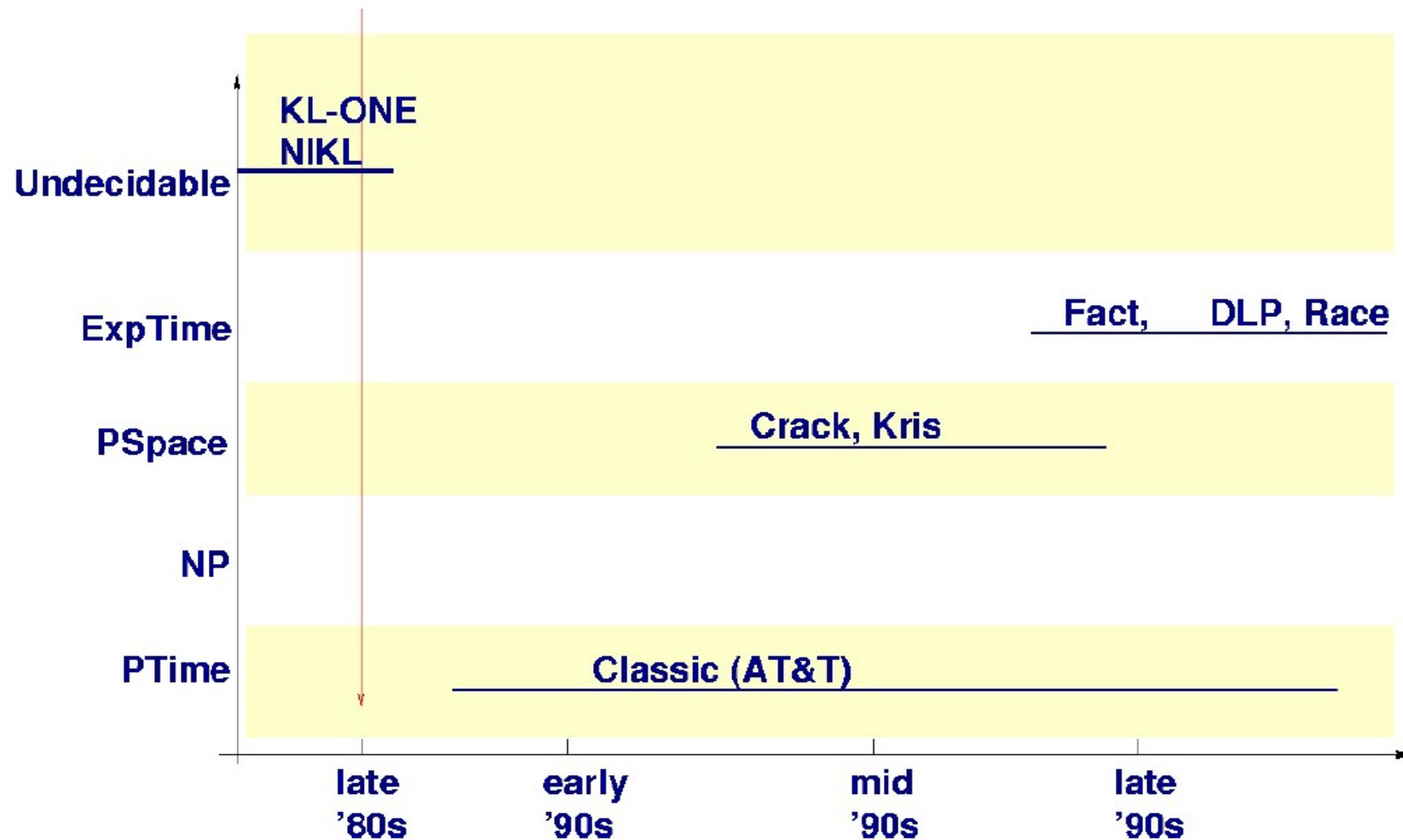


\top	top or *top*
\perp	bottom or *bottom*
$\neg A$	(not A)
$C \sqcap D$	(and C D)
$C \sqcup D$	(or C D)
$\forall R.C$	(all R C)
$\exists R.T$	(some R top)
$\exists R.C$	(some R C)
$\exists_{\geq n} R$	($\geq n$ R) or (at-least n R)
$\exists_{\leq n} R$	($\leq n$ R) or (at-most n R)

DL implementations



Description Logics should be decidable. But what complexity is “ok”?



References



- Brachman, Ronald J. and James G. Schmolze. “An Overview of the KL-ONE Knowledge Representation System.” Cognitive Science 9 (1985): 171-216.
- CLASSIC: A structural data model for objects A Borgida, RJ Brachman, DL McGuinness - ACM Sigmod, 1989.
- Living with CLASSIC: When and how to use a KL-ONE-like language RJ Brachman, DL McGuinness - Principles of semantic Networks, 1991.
- Inside the LOOM description classifier RM MacGregor - ACM Sigart Bulletin, 1991.
- Knowledge Embedding in the Description System Omega. C Hewitt, G Attardi, M Simi - AAAI, 1980.
- A description-oriented logic for building knowledge bases G Attardi, M Simi - Proceedings of the IEEE, 1986.
- The description logic handbook: Theory, implementation and applications F Baader, D Calvanese, D McGuinness- 2003.