Advanced Knowledge Based Systems
CS3411

Structural Description Logics

Enrico Franconi

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

Summary: why a logic

- Formalization of what is *true* by giving the *meaning* of logical expressions (i.e., the *representation*) with respect to the *structure* of the world.
- Formalization of the inter-relations among the *representation* and the *inference* processes.

Summary: what is a logic

Clear definitions for:

- the *formal language*
  - Semantics
  - Expressive Power
- the *reasoning problems*
  - Decidability
  - Computational Complexity
- the *problem solving procedures*
  - Soundness and Completeness
  - (Asymptotic) Complexity

The ideal computational Logic

- Expressive
- With decidable reasoning problems
- With sound and complete reasoning procedures
- With efficient reasoning procedures – possibly sub-optimal

Description Logics explore the “most” interesting expressive decidable logics with “classical” semantics, equipped with “good” reasoning procedures.

Structural Description Logics – OUTLINE

- Description Logics
  - The need for a formalism
    - O-O ambiguities
  - A structure to FOL
    - A predicate level language
- Examples from O-O
- $\mathcal{FL}^-$: the simplest structural description logics
  - Syntax
  - Semantics
  - Reasoning problems
  - Reasoning procedures

Description Logics

- A logical reconstruction and *unifying* formalism for the representation tools
  - Frame-based systems
  - Semantic Networks
  - Object-Oriented representations
  - Semantic data models
  - Type systems
  - Feature Logics
  - . . .
- A *structured* fragment of predicate logic
- Provide theories and systems for *expressing* structured information and for *accessing* and *reasoning* with it in a principled way.

Applications

Description logics based systems are currently in use in many applications.

- Configuration
- Conceptual Modeling
- Query Optimization and View Maintenance
- Natural Language Semantics
- I3 (Intelligent Integration of Information)
- Information Access and Intelligent Interfaces
- Formal Specifications in Engineering
- Terminologies and Ontologies
- Software Management
- Planning
- . . .

A formalism

- Description Logics formalize many *Object-Oriented* representation approaches.
- As such, their purpose is to disambiguate many imprecise representations.

Frames or Objects

- Identifier
- Class
- Instance
- Slot (attribute)
  - Value
    - Identifier
    - Default
  - Value restriction
    - Type
    - Concrete Domain
    - Cardinality
    - Encapsulated method

Ambiguities: classes and instances

Person : AGE : Number, SEX : M, F, HEIGHT : Number, WIFE : Person.

john : AGE : 29, SEX : M, HEIGHT : 76, WIFE : *mary*.

Ambiguities:
classes and instances (incomplete information)

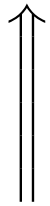
$\texttt{29'er} : \texttt{AGE} : 29,$
$\texttt{SEX} : M,$
$\texttt{HEIGHT} : \texttt{Number},$
$\texttt{WIFE} : \texttt{Person}.$

$\textit{john} : \texttt{AGE} : 29,$
$\texttt{SEX} : M,$
$\texttt{HEIGHT} : \texttt{Number},$
$\texttt{WIFE} : \texttt{Person}.$

Ambiguities:
is-a

Sub-class:

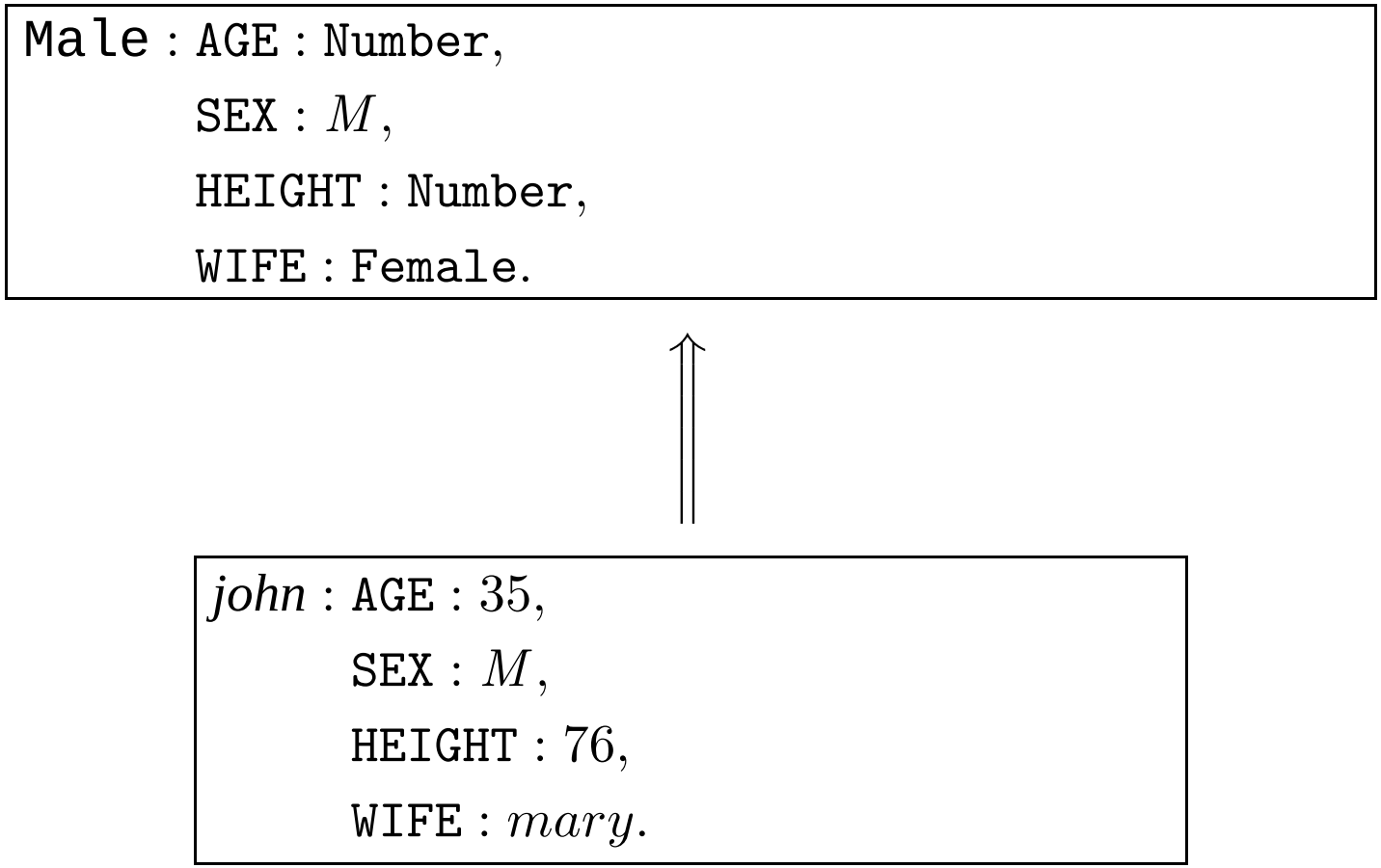
Person : AGE : Number,
SEX : M, F,
HEIGHT : Number,
WIFE : Person.



Male : AGE : Number,
SEX : M,
HEIGHT : Number,
WIFE : Female.

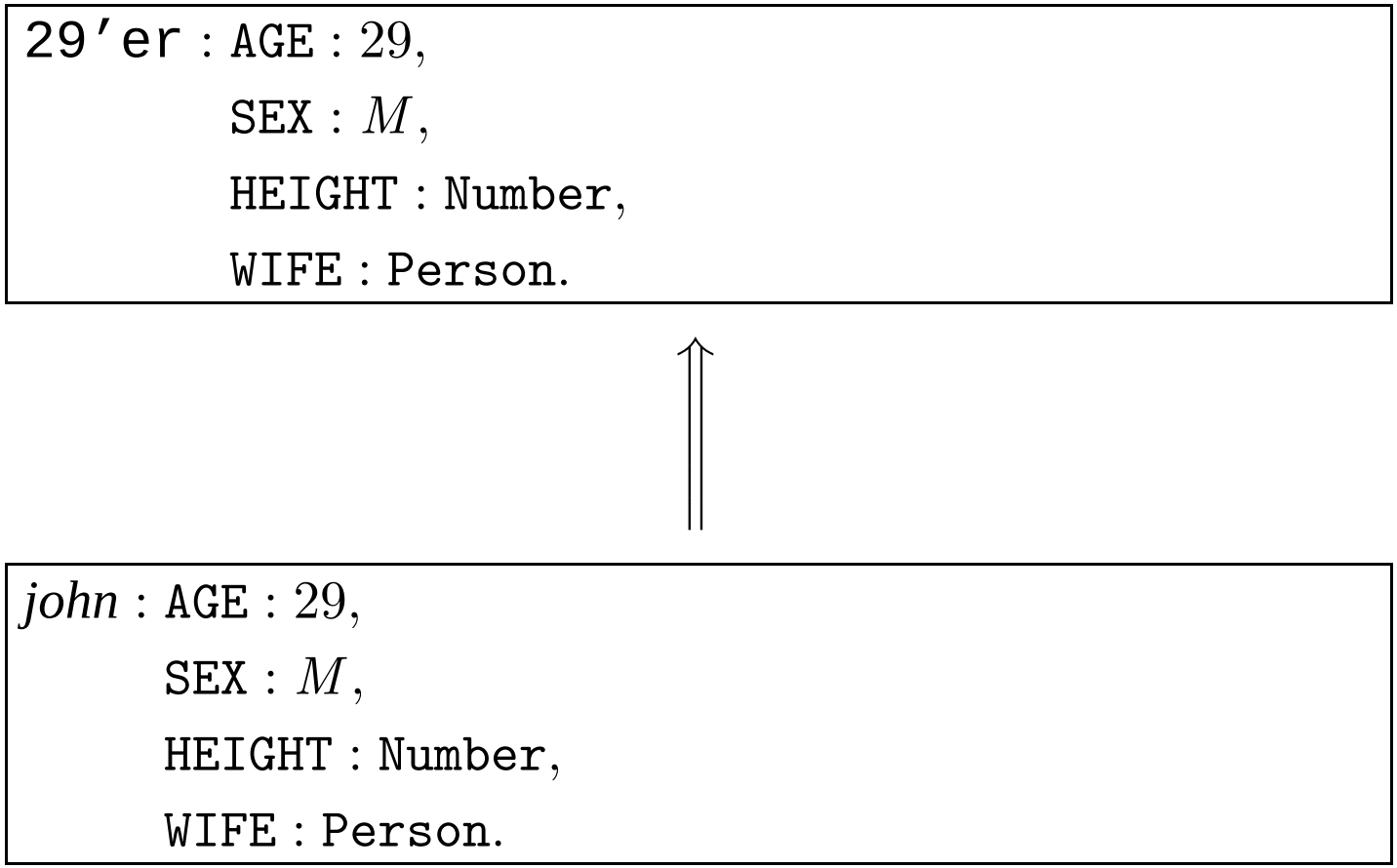
Ambiguities:
is-a

Instance-of:



Ambiguities:
is-a

Instance-of:



Ambiguities: relations

Implicit relation:

john : AGE : 35,
SEX : M,
HEIGHT : 76,
WIFE : $mary$.

mary : AGE : 32,
SEX : F,
HEIGHT : 59,
HUSBAND : $john$.

Ambiguities: relations

Explicit relation:

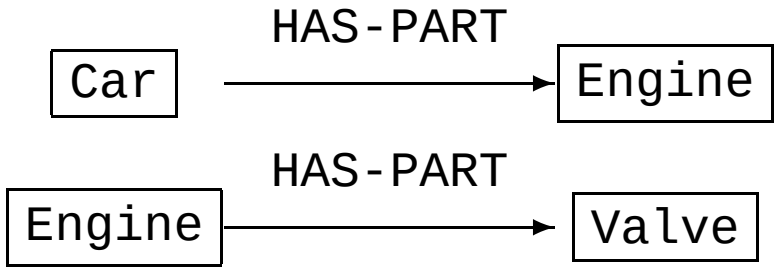
john : AGE : 35, SEX : M, HEIGHT : 76.

mary : AGE : 32, SEX : F, HEIGHT : 59.

m-j-family : WIFE : $mary$, HUSBAND : $john$.

Ambiguities: relations

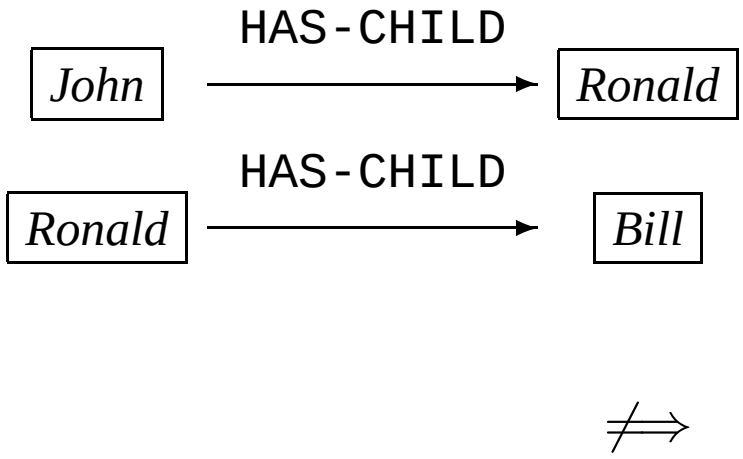
Special relation:



$\Longrightarrow$

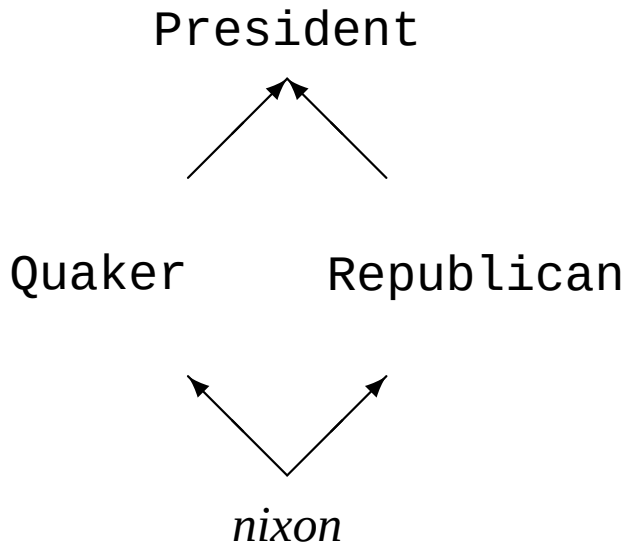
Ambiguities: relations

Normal relation:



Ambiguities: default

The *Nixon* diamond:



Quakers are pacifist, Republicans are not pacifist.

$\Longrightarrow$ Is Nixon pacifist or not pacifist?

Ambiguities: quantification

What is the exact meaning of:

Ambiguities: quantification

What is the exact meaning of:

- Every frog is just green

Ambiguities: quantification

What is the exact meaning of:

- Every frog is just green
- Every frog is also green

Ambiguities: quantification

What is the exact meaning of:

- Every frog is just green
- Every frog is also green
- Every frog is of some green

Ambiguities: quantification

What is the exact meaning of:

- Every frog is just green
- Every frog is also green
- Every frog is of some green
- There is a frog, which is just green

Ambiguities: quantification

What is the exact meaning of:

- Every frog is just green
- Every frog is also green
- Every frog is of some green
- There is a frog, which is just green
- . . .
- Frogs are typically green, but there may be exceptions

False friends

- The meaning of object-oriented representations is logically very ambiguous.
- The appeal of the graphical nature of object-oriented representation tools has led to forms of reasoning that do not fall into standard logical categories, and are not yet very well understood.
- It is unfortunately much easier to develop some algorithm that appears to reason over structures of a certain kind, than to *justify* its reasoning by explaining what the structures are saying about the domain.

A structured logic

- Any (basic) Description Logic is a fragment of FOL.
- The representation is at the *predicate level*: no variables are present in the formalism.
- A Description Logic theory is divided in two parts:
  - the definition of predicates (*TBox*)
  - the assertion over constants (*ABox*)
- Any (basic) Description Logic is a subset of $\mathcal{L}_3$, i.e. the function-free FOL using only at most *three* variable names.

Why not FOL

If FOL is directly used without additional restrictions then

- the structure of the knowledge is destroyed, and it can not be exploited for driving the inference;
- the expressive power is too high for obtaining decidable and efficient inference problems;
- the inference power may be too low for expressing interesting, but still decidable theories.

Structured Inheritance Networks: KL-ONE

- Structured Descriptions
  - corresponding to the complex relational structure of objects,
  - built using a restricted set of epistemologically adequate constructs
- distinction between conceptual (*terminological*) and instance (*assertional*) knowledge;
- central role of automatic classification for determining the subsumption – i.e., universal implication – lattice;
- strict reasoning, no defaults.

Types of the TBox Language

- **Concepts** – denote *entities*
  (unary predicates, classes)

  Example: `Student, Married`

  $\{x \mid \texttt{Student}(x)\}$,
  $\{x \mid \texttt{Married}(x)\}$

- **Roles**– denote *properties*
  (binary predicates, relations)

  Example: `FRIEND, LOVES`

  $\{\langle x, y\rangle \mid \texttt{FRIEND}(x, y)\}$,
  $\{\langle x, y\rangle \mid \texttt{LOVES}(x, y)\}$

Concept Expressions

Description Logics organize the information in classes – *concepts* – gathering homogeneous data, according to the relevant common properties among a collection of instances.

Example:

$\texttt{Student} \sqcap \exists \texttt{FRIEND}.\texttt{Married}$

$$\{x \mid \texttt{Student}(x) \wedge \exists y.\texttt{FRIEND}(x, y) \wedge \texttt{Married}(y)\}$$

A note on λ's

In general, λ is an explicit way of forming *names* of functions:

$\boxed{\lambda x.\ f(x)}$ is the function that, given input x, returns the value $f(x)$

The λ-conversion rule says that:

$$(\lambda x.\ f(x))(a) = f(a)$$

Thus, $\boxed{\lambda x.\ (x^2 + 3x - 1)}$ is the function that applied to 2 gives 9:

$$(\lambda x.\ (x^2 + 3x - 1))(2) = 9$$

We can give a name to this function, so that:

$$f_{231} \doteq \lambda x.\ (x^2 + 3x - 1)$$
$$f_{231}(2) = 9$$

λ to define predicates

Predicates are special case of functions: they are *truth* functions. So, if we think of a formula $P(x)$ as denoting a truth value which may vary as the value of x varies, we have:

$\lambda x.\ P(x)$ denotes a function from domain individuals to truth values.

In this way, as we have learned from FOL, P denotes exactly the set of individuals for which it is true. So, $P(a)$ means that the individual a makes the predicate P true, or, in other words, that a is in the extension of P.

For example, we can write for the *unary* predicate `Person`:

$$\texttt{Person} \doteq \lambda x.\ \texttt{Person}(x)$$

which is equivalent to say that `Person` denotes the *set* of persons:

$$\begin{gathered}\texttt{Person} \rightsquigarrow \{x \mid \texttt{Person}(x)\} \\ \texttt{Person}^{\mathcal{I}} = \{x \mid \texttt{Person}(x)\} \\ \texttt{Person}(john) \quad \text{IFF} \quad john^{\mathcal{I}} \in \texttt{Person}^{\mathcal{I}}\end{gathered}$$

In the same way for the *binary* predicate `FRIEND`:

$$\begin{gathered}\texttt{FRIEND} \doteq \lambda x, y.\ \texttt{FRIEND}(x, y) \\ \texttt{FRIEND}^{\mathcal{I}} = \{\langle x, y\rangle \mid \texttt{FRIEND}(x, y)\}\end{gathered}$$

The functions we are defining with the λ operator may be parametric:

$$\texttt{Student} \sqcap \texttt{Worker} = \lambda x.\ (\texttt{Student}(x) \wedge \texttt{Worker}(x))$$

$$(\texttt{Student} \sqcap \texttt{Worker})^{\mathcal{I}} = \{x \mid\ (\texttt{Student}(x) \wedge \texttt{Worker}(x)\}$$

$$(\texttt{Student} \sqcap \texttt{Worker})^{\mathcal{I}} = \texttt{Student}^{\mathcal{I}} \cap \texttt{Worker}^{\mathcal{I}}$$

(Verify as exercise)

Concept Expressions

$$(\texttt{Student} \sqcap \exists \texttt{FRIEND}.\texttt{Married})^{\mathcal{I}}$$

$$=$$

$$(\texttt{Student})^{\mathcal{I}} \cap (\exists \texttt{FRIEND}.\texttt{Married})^{\mathcal{I}}$$

$$=$$

$$\{x \mid \texttt{Student}(x)\} \cap$$
$$\{x \mid \exists y.\, \texttt{FRIEND}(x, y) \wedge \texttt{Married}(y)\}$$

$$=$$

$$\{x \mid \texttt{Student}(x) \wedge$$
$$\exists y.\, \texttt{FRIEND}(x, y) \wedge \texttt{Married}(y)\}$$

Objects: classes

Student

Person	
name:	[String]
address:	[String]
enrolled:	[Course]

$$\{x \mid \texttt{Student}(x)\} =$$
$$\{x \mid \texttt{Person}(x) \wedge$$
$$(\exists y.\ \texttt{NAME}(x, y) \wedge \texttt{String}(y)) \wedge$$
$$(\exists z.\ \texttt{ADDRESS}(x, z) \wedge \texttt{String}(z)) \wedge$$
$$(\exists w.\ \texttt{ENROLLED}(x, w) \wedge \texttt{Course}(w))\ \}$$

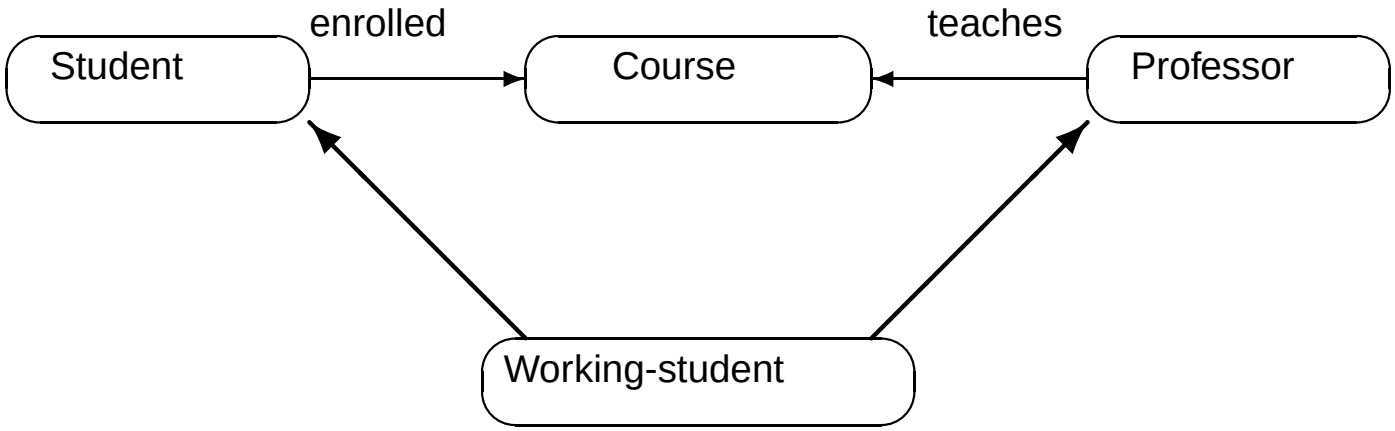
$$\texttt{Student} \doteq \texttt{Person} \sqcap$$
$$\exists \texttt{NAME}.\texttt{String} \sqcap$$
$$\exists \texttt{ADDRESS}.\texttt{String} \sqcap$$
$$\exists \texttt{ENROLLED}.\texttt{Course}$$

Objects: instances

s1: Student	
name:	“John”
address:	“Abbey Road. . .”
enrolled:	cs415

$$\texttt{Student}(\texttt{s1}) \wedge$$
$$\texttt{NAME}(\texttt{s1}, \text{“}\texttt{john}\text{”}) \wedge \texttt{String}(\text{“}\texttt{john}\text{”}) \wedge$$
$$\texttt{ADDRESS}(\texttt{s1}, \text{“}\texttt{abbey-road}\text{”}) \wedge \texttt{String}(\text{“}\texttt{abbey-road}\text{”}) \wedge$$
$$\texttt{ENROLLED}(\texttt{s1}, \texttt{cs415}) \wedge \texttt{Course}(\texttt{cs415})$$

Semantic Networks



$\forall x.\ \texttt{Student}(x) \rightarrow$
$\quad \exists y.\ \texttt{ENROLLED}(x, y) \wedge \texttt{Course}(y)$

$\forall x.\ \texttt{Professor}(x) \rightarrow$
$\quad \exists y.\ \texttt{TEACHES}(x, y) \wedge \texttt{Course}(y)$

$\forall x.\ \texttt{Working-student}(x) \rightarrow$
$\quad \texttt{Student}(x) \wedge \texttt{Professor}(x)$

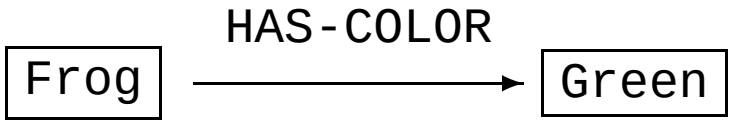
$\texttt{Student} \sqsubseteq \exists \texttt{ENROLLED.Course}$

$\texttt{Professor} \sqsubseteq \exists \texttt{TEACHES.Course}$

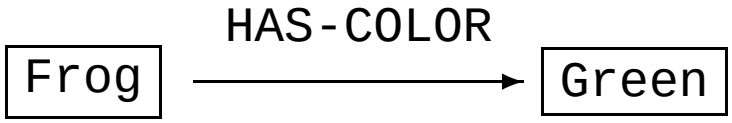
$\texttt{Working-student} \sqsubseteq \texttt{Student}$

$\texttt{Working-student} \sqsubseteq \texttt{Professor}$

Quantification



- $\texttt{Frog} \sqsubseteq \exists \texttt{HAS-COLOR}.\texttt{Green}$:
  Every frog is also green
- $\texttt{Frog} \sqsubseteq \forall \texttt{HAS-COLOR}.\texttt{Green}$:
  Every frog is just green
- $\texttt{Frog} \sqsubseteq \forall \texttt{HAS-COLOR}.\texttt{Green}$
  $\texttt{Frog}(x)$, $\texttt{HAS-COLOR}(x, y)$:
  There is a frog, which is just green



Every frog is also green

$\texttt{Frog} \sqsubseteq \exists \texttt{HAS-COLOR}.\texttt{Green}$

$\forall x.\ \texttt{Frog}(x) \rightarrow$
$\quad \exists y.\ (\texttt{HAS-COLOR}(x, y) \wedge \texttt{Green}(y))$

Exercise: is this a model?

Frog(*oscar*), Green(*green*),
HAS-COLOR(*oscar*,*green*),
Red(*red*),
HAS-COLOR(*oscar*,*red*).

Every frog is only green

$\texttt{Frog} \sqsubseteq \forall \texttt{HAS-COLOR}.\texttt{Green}$

$\forall x.\ \texttt{Frog}(x) \rightarrow$
$\quad \forall y.\ (\texttt{HAS-COLOR}(x, y) \rightarrow \texttt{Green}(y))$

Exercise: is this a model?

Frog(*oscar*), Green(*green*),
HAS-COLOR(*oscar*,*green*),
Red(*red*),
HAS-COLOR(*oscar*,*red*).

and this?

Frog(*sing*),
AGENT(*sing*,*oscar*).

Objects: adequacy

(Exercise) Check whether the found representation for `Student` is an adequate one.

Student

Person	
name:	[String]
address:	[String]
enrolled:	[Course]

$$\{x \mid \texttt{Student}(x)\} =$$
$$\{x \mid \texttt{Person}(x) \wedge$$
$$(\exists y.\ \texttt{NAME}(x,y) \wedge \texttt{String}(y)) \wedge$$
$$(\exists z.\ \texttt{ADDRESS}(x,z) \wedge \texttt{String}(z)) \wedge$$
$$(\exists w.\ \texttt{ENROLLED}(x,w) \wedge \texttt{Course}(w))\ \}$$

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

Another example

$$
\begin{array}{l}
(\texttt{Student} \sqcap \\
\quad \exists \texttt{dept.CS} \sqcap \\
\quad \geq 3\ \texttt{enrolled-course.} \\
\qquad (\texttt{Graduate-Course} \sqcap \exists \texttt{dept.CS-dept}))
\end{array}
$$

$$
\begin{array}{l}
\lambda x.\ [\texttt{Student}(x) \wedge \texttt{dept}(x, \texttt{CS}) \wedge \\
\quad \exists y_1 y_2 y_3 (y_1 \neq y_2 \wedge y_1 \neq y_3 \wedge y_2 \neq y_3 \wedge \\
\qquad \texttt{enrolled-course}(x, y_1) \wedge \texttt{Grad-Course}(y_1) \wedge \\
\qquad\quad \exists z (\texttt{dept}(y_1, z) \wedge \texttt{CS-dept}(z)) \wedge \\
\qquad \texttt{enrolled-course}(x, y_2) \wedge \texttt{Grad-Course}(y_2) \wedge \\
\qquad\quad \exists z (\texttt{dept}(y_2, z) \wedge \texttt{CS-dept}(z)) \wedge \\
\qquad \texttt{enrolled-course}(x, y_3) \wedge \texttt{Grad-Course}(y_3) \wedge \\
\qquad\quad \exists z (\texttt{dept}(y_3, z) \wedge \texttt{CS-dept}(z)))]
\end{array}
$$

CLUMSY!

Analytic reasoning

(Let's see this intuitively, by now.)

Person

subsumes

(Person **with every** male friend **is a** doctor)

subsumes

(Person **with every** friend **is a**
    (Doctor **with a** specialty **is** surgery))

Analytic reasoning

(Let's see this intuitively, by now.)

Person

subsumes

(Person **with every** male friend **is a** doctor)

subsumes

(Person **with every** friend **is a**
    (Doctor **with a** specialty **is** surgery))

(Person **with** $\geq$ 2 children)

subsumes

(Person **with** $\geq$ 3 male children)

Analytic reasoning

(Let's see this intuitively, by now.)

Person

subsumes

(Person **with every** male friend **is a** doctor)

subsumes

(Person **with every** friend **is a**
    (Doctor **with a** specialty **is** surgery))

(Person **with** $\geq$ 2 children)

subsumes

(Person **with** $\geq$ 3 male children)

(Person **with** $\geq$ 3 young children)

disjoint

(Person **with** $\leq$ 2 children)

Basic References

- Russell and Norvig *Artificial Intelligence: A Modern Approach*. *Chapter 10, section 6.*
- H. J. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. Computational Intelligence journal 3, 78-93 (1987).
- B. Nebel. Reasoning and Revision in Hybrid Representation Systems. Lecture Notes in Artificial Intelligence 422, Springer-Verlag, 1990. *Chapters 1 – 6.*
- Woods, W., Schmolze, J., 'The KL-ONE Family', Computers and Mathematics with Applications, special issue: Semantic Networks in Artificial Intelligence, Vol 2-5, pp 133-177, 1992.