A Sorted-Graph Unification Approach to the Semantic Web

Hassan Aït-Kaci

Senior Technical Staff Member

IBM Canada Ltd.

- Semantic Web formalisms
- Graphs as constraints
- $\mathcal{OSF}$ *vs.* $\mathcal{DL}$
- $\mathcal{LIFE}$—logically and functionally constrained sorted graphs
- Recapitulation

- Semantic Web formalisms
- Graphs as constraints
- $\mathcal{OSF}$ vs. $\mathcal{DL}$
- $\mathcal{LIFE}$—logically and functionally constrained sorted graphs
- Recapitulation

- $\mathcal{OWL}$ technology has become its *de facto* standard

- Everyone talks about $\mathcal{OWL}$ dialects!

  The whole World-Wide Web is abuzz with $\mathcal{OWL}$-*this* and $\mathcal{OWL}$-*that*, ...

- **However, *a lesser number understands them...***

  $\mathcal{SHIF}$, $\mathcal{SHIQ}$, $\mathcal{SHOQ}(\mathcal{D})$, $\mathcal{SHOIQ}$, $\mathcal{SRIQ}$, $\mathcal{SROIQ}$, ...

  are *not* alien species' tongues but dialects devised for $\mathcal{OWL}$ (W3C's Ontology Web Language) by some of the most prolific and influential SW's researchers.

What language(s) do $\mathcal{OWL}$'s speak?—a confusing growing crowd of strange-sounding languages and logics:

- $\mathcal{OWL}$, $\mathcal{OWL}$ Lite, $\mathcal{OWL}$ DL, $\mathcal{OWL}$ Full
- $\mathcal{DL}$, $\mathcal{DLR}$, ...
- $\mathcal{AL}$, $\mathcal{ALC}$, $\mathcal{ALCN}$, $\mathcal{ALCNR}$, ...
- $\mathcal{SHIF}$, $\mathcal{SHIN}$, $\mathcal{CIQ}$, $\mathcal{SHIQ}$, $\mathcal{SHOQ}(D)$, $\mathcal{SHOIQ}$, $\mathcal{SRIQ}$, $\mathcal{SROIQ}$, ...

Depending on whether the system allows:

- concepts, roles (inversion, composition, inclusion, ...)
- individuals, datatypes, cardinality constraints
- various combination thereof

For better or worse, the W3C has married its efforts to $\mathcal{DL}$-based reasoning systems:

- All the proposed $\mathcal{DL}$ Knowledge Base formalisms in the $\mathcal{OWL}$ family use *tableaux-based methods* for reasoning
- Tableaux methods work by building models explicitly via formula expansion rules
- This limits $\mathcal{DL}$ reasoning to finite (*i.e.*, decidable) models
- Worse, tableaux methods only work for small ontologies: they fail to scale up to large ontologies

Tableaux style $\mathcal{DL}$ reasoning ($\mathcal{ALCNR}$)

$(\mathcal{DL}_\sqcap)$ **CONJUNCTIVE CONCEPT:**

$$\left[\begin{array}{rl} \textbf{if} & x:(C_1 \sqcap C_2) \;\in\; S \\ \textbf{and} & \{x:C_1, x:C_2\} \not\subseteq S \end{array}\right] \qquad \frac{S}{S \cup \{x:C_1, x:C_2\}}$$

$(\mathcal{DL}_\sqcup)$ **DISJUNCTIVE CONCEPT:**

$$\left[\begin{array}{rl} \textbf{if} & x:(C_1 \sqcup C_2) \;\in\; S \\ \textbf{and} & x:C_i \;\notin\; S \;\; (i=1,2) \end{array}\right] \qquad \frac{S}{S \cup \{x:C_i\}}$$

$(\mathcal{DL}_\forall)$ **UNIVERSAL ROLE:**

$$\left[\begin{array}{rl} \textbf{if} & x:(\forall R.C) \;\in\; S \\ \textbf{and} & y \;\in\; R_S[x] \\ \textbf{and} & y:C \;\notin\; S \end{array}\right] \qquad \frac{S}{S \cup \{y:C\}}$$

$(\mathcal{DL}_{\exists})$ **EXISTENTIAL ROLE**:

$$\left[\begin{array}{ll} \textbf{if} & x : (\exists R.C) \in S \ \textbf{s.t.}\ R \stackrel{\text{DEF}}{=\!=} (\textstyle\prod_{i=1}^{m} R_i) \\ \textbf{and} & z : C \in S \Rightarrow z \notin R_S[x] \\ \textbf{and} & y \text{ is new} \end{array}\right] \quad \frac{S}{S \cup \{xR_i y\}_{i=1}^{m} \cup \{y : C\}}$$

$(\mathcal{DL}_{\geq})$ **MIN CARDINALITY**:

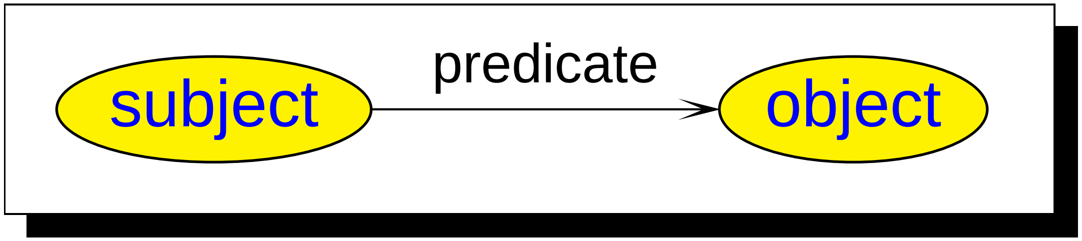
$$\left[\begin{array}{ll} \textbf{if} & x : (\geq n.R) \in S \ \textbf{s.t.}\ R \stackrel{\text{DEF}}{=\!=} (\textstyle\prod_{i=1}^{m} R_i) \\ \textbf{and} & |R_S[x]| \neq n \\ \textbf{and} & y_i \text{ is new } (0 \leq i \leq n) \end{array}\right] \quad \frac{S}{S \cup \{xR_i y_j\}_{i,j=1,1}^{m,n} \cup \{y_i \dot{\neq} y_j\}_{1 \leq i < j \leq n}}$$

$(\mathcal{DL}_{\leq})$ **MAX CARDINALITY**:

$$\left[\begin{array}{ll} \textbf{if} & x : (\leq n.R) \in S \\ \textbf{and} & |R_S[x]| > n \ \ \textbf{and}\ \ y, z \in R_S[x] \\ \textbf{and} & y \dot{\neq} z \notin S \end{array}\right] \quad \frac{S}{S \cup S[y/z]}$$

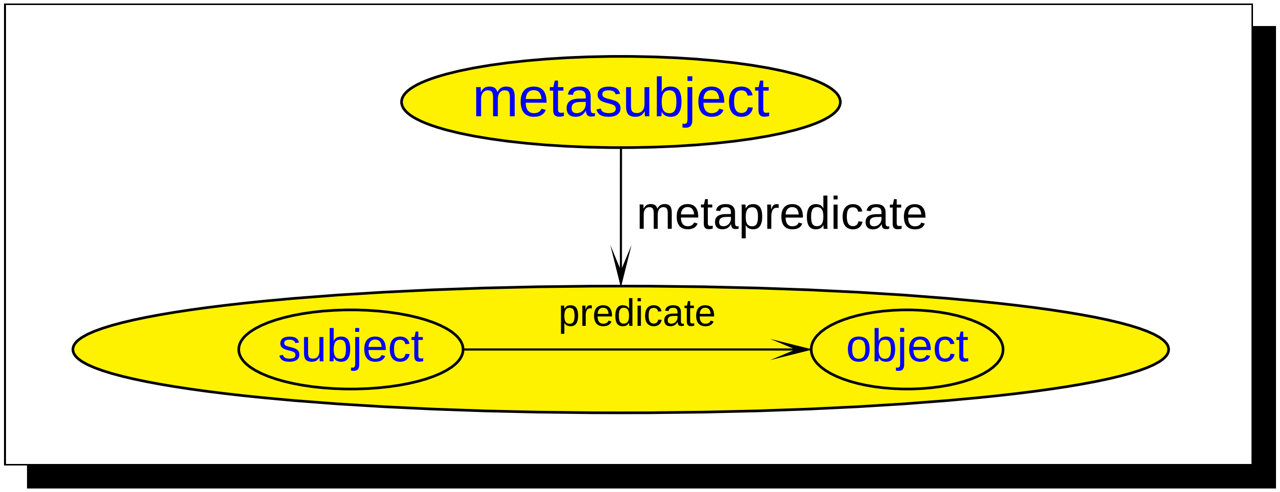
RDF is a notation for meta-description about data (metadata) using (edge- and node-) labeled graphs.

- Basic building block: "triple" labeled by "resources"—*i.e.*, URI's.
- A triple consists of a resource (the subject), linked through a resource (the predicate) to another resource (the object).
- A triple states that the subject has a property, denoted by the predicate, whose value is the object:



- The information carried by a triple is called a "statement."

- RDF statements can be reified and be denoted as resources—hence, RDF's metalinguistic nature:



- RDF uses XML for its serialized syntax.
- RDF enables the definition of vocabularies which can be shared over the Web thanks to XML namespaces (*e.g.*, Dublin Core).
- RDF Schema (RDFS) is a meta-description of RDF in RDF; it defines a vocabulary for RDF.

RDF triples may be expressed using several syntaxes:

- a (normative) XML syntax
- Notation 3 syntax (TBL)
- Turtle—TRTL: Terse RDF Triple Language (TBL, David Beckett)
- ...

Linked Data has become *de facto* standard for distributed information—***it does to*** $\mathcal{RDF}$ ***what HTML has done to text: it interconnects knowledge through the Internet***

```
<rdf:RDF
  xmlns:rdf
   ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:ex="http://w3.hak.org/school-ns#">

  <rdf:Description rdf:about="ID-6541">
    <ex:name>John Doe</ex:name>
    <ex:title>Assistant Professor</ex:title>
    <ex:age rdf:datatype="&xsd:integer">35</ex:age>
    <ex:teaches rdf:resource="#CS-100"/>
    <ex:teaches rdf:resource="#CS-345"/>
  </rdf:Description>
```

```
<rdf:Description rdf:about="CS-100">
  <ex:courseName>
    Introduction to Computer Programming
  </ex:courseName>
  <ex:courseTime>MTW/9:00-10:30</ex:courseTime>
  <ex:coursePlace>Wheston Hall 230</ex:coursePlace>
</rdf:Description>

<rdf:Description rdf:about="CS-200">
  <ex:courseName>Operating Systems</ex:courseName>
  <ex:courseTime>TTh/11:00-13:00</ex:courseTime>
  <ex:coursePlace>Dietrich Hall 34</ex:coursePlace>
</rdf:Description>
```

```
  <rdf:Description rdf:about="CS-345">
    <ex:courseName>
      Introduction to Compiler Design
    </ex:courseName>
    <ex:courseTime>MTW/9:00-10:30</ex:courseTime>
    <ex:coursePlace>Chetham Hall 130</ex:coursePlace>
    <ex:prerequisites>
      <rdf:bag>
        <rdf:_1 rdf:resource="#CS-100">
        <rdf:_2 rdf:resource="#CS-220">
      </rdf:bag>
    </ex:prerequisites>
  </rdf:Description>

</rdf:RDF>
```

Adding types to RDF nodes:

```
<rdf:Description rdf:about="CS-100">
  <rdf:type rdf:resource="ex:course"/>
  <ex:courseName>
    Introduction to Computer Programming
  </ex:courseName>
  <ex:courseInstructor rdf:resource="#ID-6541"/>
  <ex:courseTime>MTW/9:00-10:30</ex:courseTime>
  <ex:coursePlace>Wheston Hall 230</ex:coursePlace>
</rdf:Description>
```

Adding types to RDF nodes:

```
<rdf:Description rdf:about="ID-6541">
  <rdf:type rdf:resource="ex:instructor"/>
  <ex:name>John Doe</ex:name>
  <ex:title>Assistant Professor</ex:title>
  <ex:age rdf:datatype="&xsd:integer">35</ex:age>
  <ex:teaches rdf:resource="#CS-100"/>
  <ex:teaches rdf:resource="#CS-345"/>
</rdf:Description>
```

Simplified XML notation for RDF nodes:

1. Replace `rdf:Description` tag with the value of its `rdf:type` attribute if present
2. Replace a single leaf node by an attribute named as the node's tag with string value equal to the node's contents

```
<ex:instructor
  rdf:about="ID-6541"
  ex:name="John Doe"
  ex:title="Assistant Professor"/>

  <ex:age rdf:datatype="&xsd:integer">35</ex:age>
  <ex:teaches rdf:resource="#CS-100"/>
  <ex:teaches rdf:resource="#CS-345"/>
</ex:instructor>
```

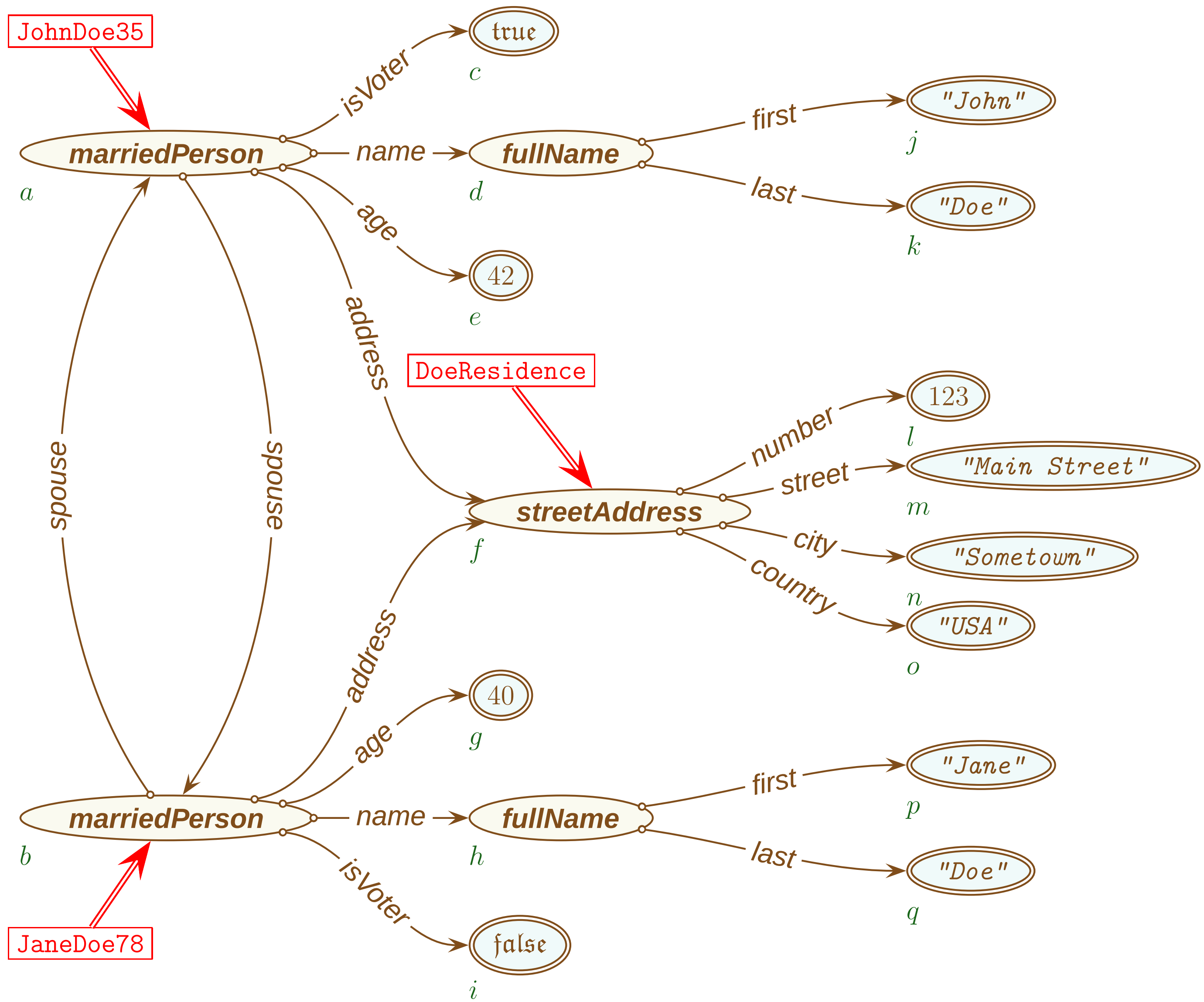
Semantic Web formalisms—*Objects are labelled graphs!*

```
JohnDoe35 : marriedPerson ( name    => fullName
                                       ( first => "John"
                                       , last  => "Doe" )
                          , age     => 42
                          , address => DoeResidence
                          , spouse  => JaneDoe78
                          , isVoter => true
                          )
```

Semantic Web formalisms—*Objects are labelled graphs!*

```
JaneDoe78 : marriedPerson ( name     => fullName
                                        ( first => "Jane"
                                        , last  =>  "Doe" )
                          , age      => 40
                          , address  => DoeResidence
                          , spouse   => JohnDoe35
                          , isVoter  => false
                          )

DoeResidence : streetAddress  ( number  => 123
                              , street  => "Main Street"
                              , city    => "Sometown"
                              , country => "USA"
                              )
```



JohnDoe35
marriedPerson
a
isVoter
true
c
name
fullName
d
first
"John"
j
last
"Doe"
k
age
42
e
address
DoeResidence
streetAddress
f
number
123
l
street
"Main Street"
m
city
"Sometown"
n
country
"USA"
o
spouse
spouse
address
age
40
g
marriedPerson
b
name
fullName
h
first
"Jane"
p
last
"Doe"
q
isVoter
false
i
JaneDoe78

$\mathcal{OSF}$ Graph Constraint Formalism—*Outline*

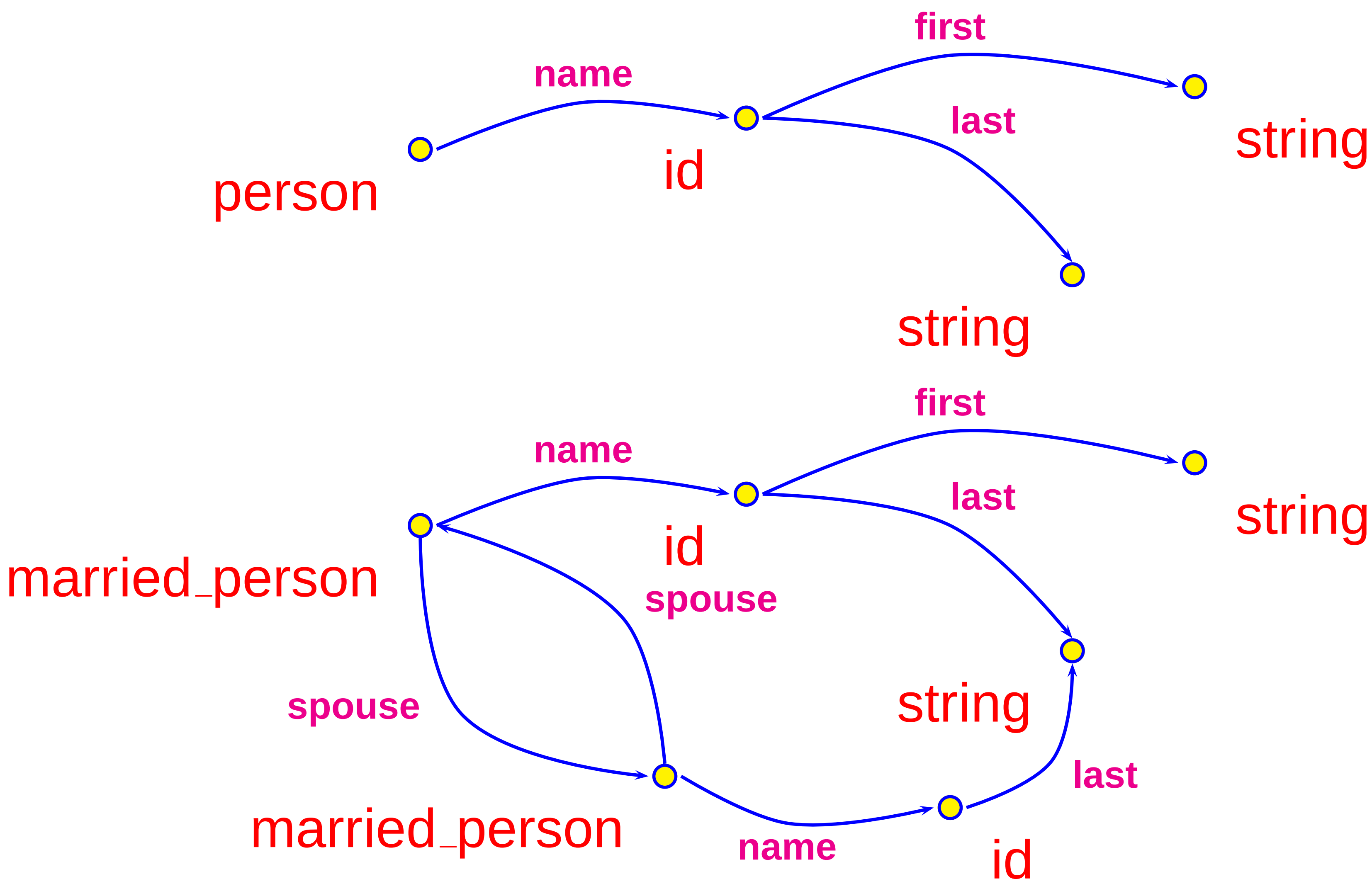
- Semantic Web formalisms
- Graphs as constraints
- $\mathcal{OSF}$ *vs.* $\mathcal{DL}$
- $\mathcal{LIFE}$—logically and functionally constrained sorted graphs
- Recapitulation

- **Proposal:** a formalism for representing objects that is: intuitive (objects as labelled graphs), expressive (“real-life” data models), formal (logical semantics), operational (executable), & efficient (constraint-solving)
- **Why?** *viz.*, ubiquitous use of labelled graphs to structure information naturally as in:
  - object-orientation, knowledge representation,
  - databases, semi-structured data,
  - natural language processing, graphical interfaces,
  - concurrency and communication,
  - XML, RDF, the “Semantic Web,” *etc.*, ...

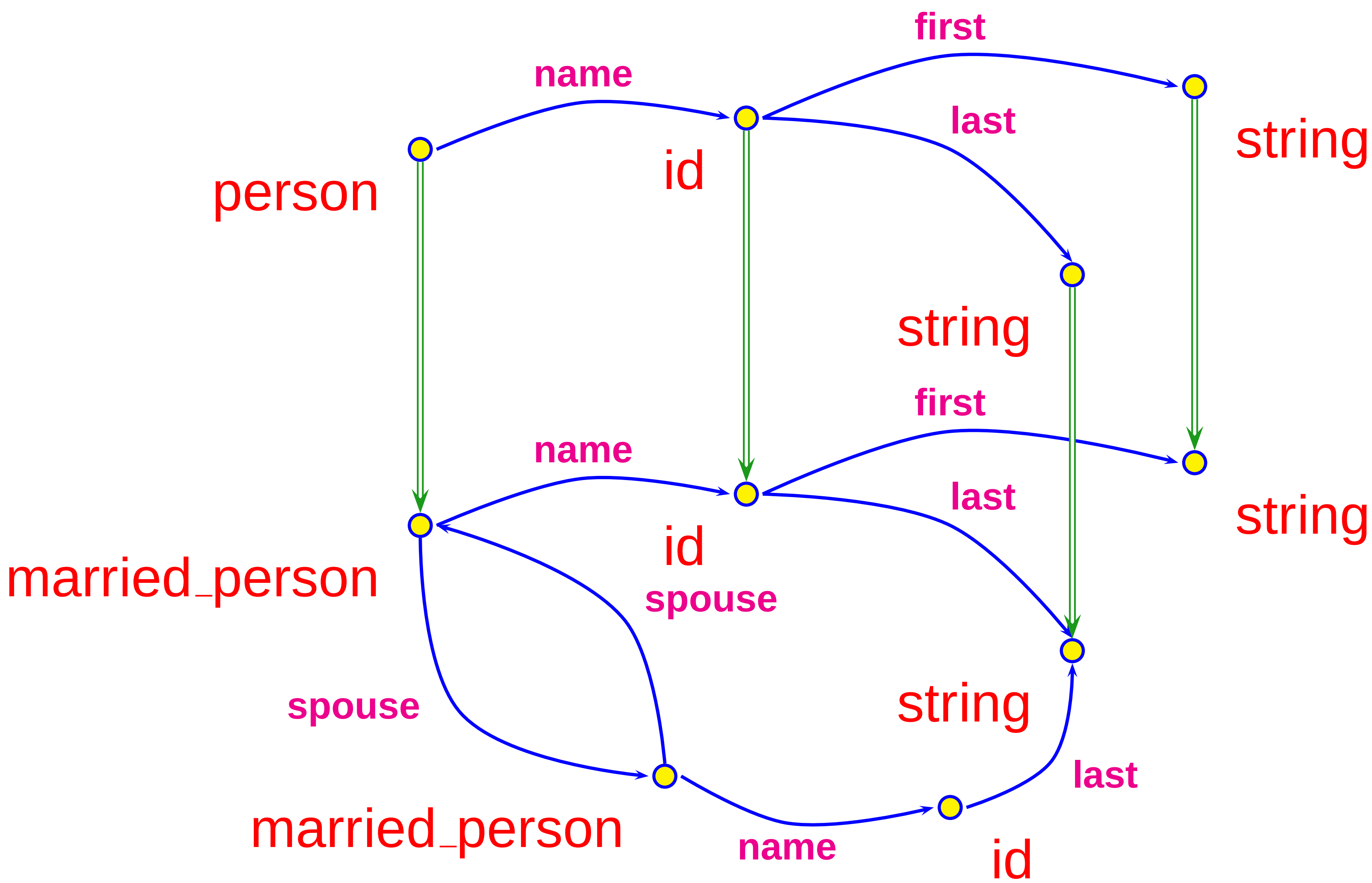
Viewing graphs as ***constraints*** stems from the work of:

- Hassan Aït-Kaci (since 1983)
- Gert Smolka (since 1986)
- Andreas Podelski (since 1989)
- Franz Baader, Rolf Backhofen, Jochen Dörre, Martin Emele, Bernhard Nebel, Joachim Niehren, Ralf Treinen, Manfred Schmidt-Schauß, Remi Zajac, . . .

Graphs as constraints—*Inheritance as graph endomorphism*



Graphs as constraints—*Inheritance as graph endomorphism*



Let $\mathcal{V}$ be a countably infinite set of variables.

An $\mathcal{OSF}$ term is an expression of the form:

$$X : s(\ell_1 \Rightarrow t_1, \ldots, \ell_n \Rightarrow t_n)$$

where:

- $X \in \mathcal{V}$ is the root variable
- $s \in \mathcal{S}$ is the root sort
- $n \geq 0$ (if $n = 0$, we write $X : s$)
- $\{\ell_1, \ldots, \ell_n\} \subseteq \mathcal{F}$ are features
- $t_1, \ldots, t_n$ are $\mathcal{OSF}$ terms

Graphs as constraints—$\mathcal{OSF}$ *term syntax example*

$$
\begin{aligned}
X : person(&name \Rightarrow N : \top(first \Rightarrow F : string),\\
&name \Rightarrow M : id(last \Rightarrow S : string),\\
&spouse \Rightarrow P : person(name \Rightarrow I : id(last \Rightarrow S : \top),\\
&\qquad\qquad\qquad\qquad spouse \Rightarrow X : \top)).
\end{aligned}
$$

Lighter notation:

$$
\begin{aligned}
X : person(&name \Rightarrow \top(first \Rightarrow string),\\
&name \Rightarrow id(last \Rightarrow S : string),\\
&spouse \Rightarrow person(name \Rightarrow id(last \Rightarrow S),\\
&\qquad\qquad\qquad\qquad spouse \Rightarrow X)).
\end{aligned}
$$

- $\mathcal{OSF}$ term $t = X : s(\ell_1 \Rightarrow t_1, \ldots, \ell_n \Rightarrow t_n)$
- $\mathcal{OSF}$ interpretation $\mathfrak{A}$
- $\mathfrak{A}$-valuation $\alpha : \mathcal{V} \mapsto D^{\mathfrak{A}}$

Denotation of t in $\mathfrak{A}$ under valuation α:

$$[\![t]\!]^{\mathfrak{A},\alpha} \stackrel{\text{DEF}}{=} \{\alpha(X)\} \cap s^{\mathfrak{A}} \cap \left(\bigcap_{1 \leq i \leq n} (\ell_i^{\mathfrak{A}})^{-1}([\![t_i]\!]^{\mathfrak{A},\alpha}) \right)$$

Denotation of t in $\mathfrak{A}$ under all possible valuations:

$$[\![t]\!]^{\mathfrak{A}} \stackrel{\text{DEF}}{=} \bigcup_{\alpha:\mathcal{V} \mapsto D^{\mathfrak{A}}} [\![t]\!]^{\mathfrak{A},\alpha}.$$

An $\mathcal{OSF}$ constraint is one of:

- $X : s$
- $X.\ell \doteq X'$
- $X \doteq X'$

where X (X') is a variable (*i.e.*, a node), s is a sort (*i.e.*, a node's type), and ℓ is a feature (*i.e.*, an arc).

An $\mathcal{OSF}$ clause is a conjunction of $\mathcal{OSF}$constraints—*i.e.*, a set of $\mathcal{OSF}$ constraints

$$\phi_1 \quad \& \quad \ldots \quad \& \quad \phi_n$$

Satisfaction of $\mathcal{OSF}$ constraints in an $\mathcal{OSF}$ algebra $\mathfrak{A}$ by a valuation $\alpha : \mathcal{V} \mapsto D^{\mathfrak{A}}$ is defined by:

$$
\begin{array}{lcl}
\mathfrak{A}, \alpha \models X : s & \Longleftrightarrow & \alpha(X) \in s^{\mathfrak{A}} \\
\mathfrak{A}, \alpha \models X \doteq Y & \Longleftrightarrow & \alpha(X) = \alpha(Y) \\
\mathfrak{A}, \alpha \models X.\ell \doteq Y & \Longleftrightarrow & \ell^{\mathfrak{A}}(\alpha(X)) = \alpha(Y) \\
\mathfrak{A}, \alpha \models \phi_1 \ \& \ \ldots \ \& \ \phi_n & \Longleftrightarrow & \mathfrak{A}, \alpha \models \phi_i \ \ \forall i = 1, \ldots, n
\end{array}
$$

An $\mathcal{OSF}$ term $t = X : s(\ell_1 \Rightarrow t_1, \ldots, \ell_n \Rightarrow t_n)$ is dissolved into an $\mathcal{OSF}$ clause $\phi(t)$ as follows:

$$\boxed{\begin{array}{lllllllll} \varphi(t) & \stackrel{\text{DEF}}{=\!=} & X : s & \& & X.\ell_1 \doteq X_1 & \& & \ldots & \& & X.\ell_n \doteq X_n \\ & & & \& & \varphi(t_1) & \& & \ldots & \& & \varphi(t_n) \end{array}}$$

where $X_1, \ldots, X_n$ are the root variables of $t_1, \ldots, t_n$.

Theorem: $$\boxed{\mathfrak{A}, \alpha \models \varphi(t) \iff [\![t]\!]^{\mathfrak{A},\alpha} \neq \emptyset}$$

Graphs as constraints—*Example of $\mathcal{OSF}$ term dissolution*

$$
\begin{aligned}
t \;=\; X : person(&name \Rightarrow N : \top(first \Rightarrow F : string),\\
&name \Rightarrow M : id(last \Rightarrow S : string),\\
&spouse \Rightarrow P : person(name \Rightarrow I : id(last \Rightarrow S : \top),\\
&\qquad\qquad\qquad\qquad\quad spouse \Rightarrow X : \top))
\end{aligned}
$$

$$
\begin{array}{rlllll}
\varphi(t) = X : person & \& & X\,.\,name \doteq N & \& & N : \top \\
& \& & X\,.\,name \doteq M & \& & M : id \\
& \& & X\,.\,spouse \doteq P & \& & P : person \\
& \& & N\,.\,first \doteq F & \& & F : string \\
& \& & M\,.\,last \doteq S & \& & S : string \\
& \& & P\,.\,name \doteq I & \& & I : id \\
& \& & I\,.\,last \doteq S & \& & S : \top \\
& \& & P\,.\,spouse \doteq X & \& & X : \top
\end{array}
$$

(1) Sort Intersection

$$\frac{\phi \ \& \ X : s \ \& \ X : s'}{\phi \ \& \ X : s \wedge s'}$$

(3) Variable Elimination

$$\frac{\phi \ \& \ X \doteq X'}{\phi[X'/X] \ \& \ X \doteq X'} \quad \text{if } X \neq X' \text{ and } X \in \textbf{\textit{Var}}(\phi)$$

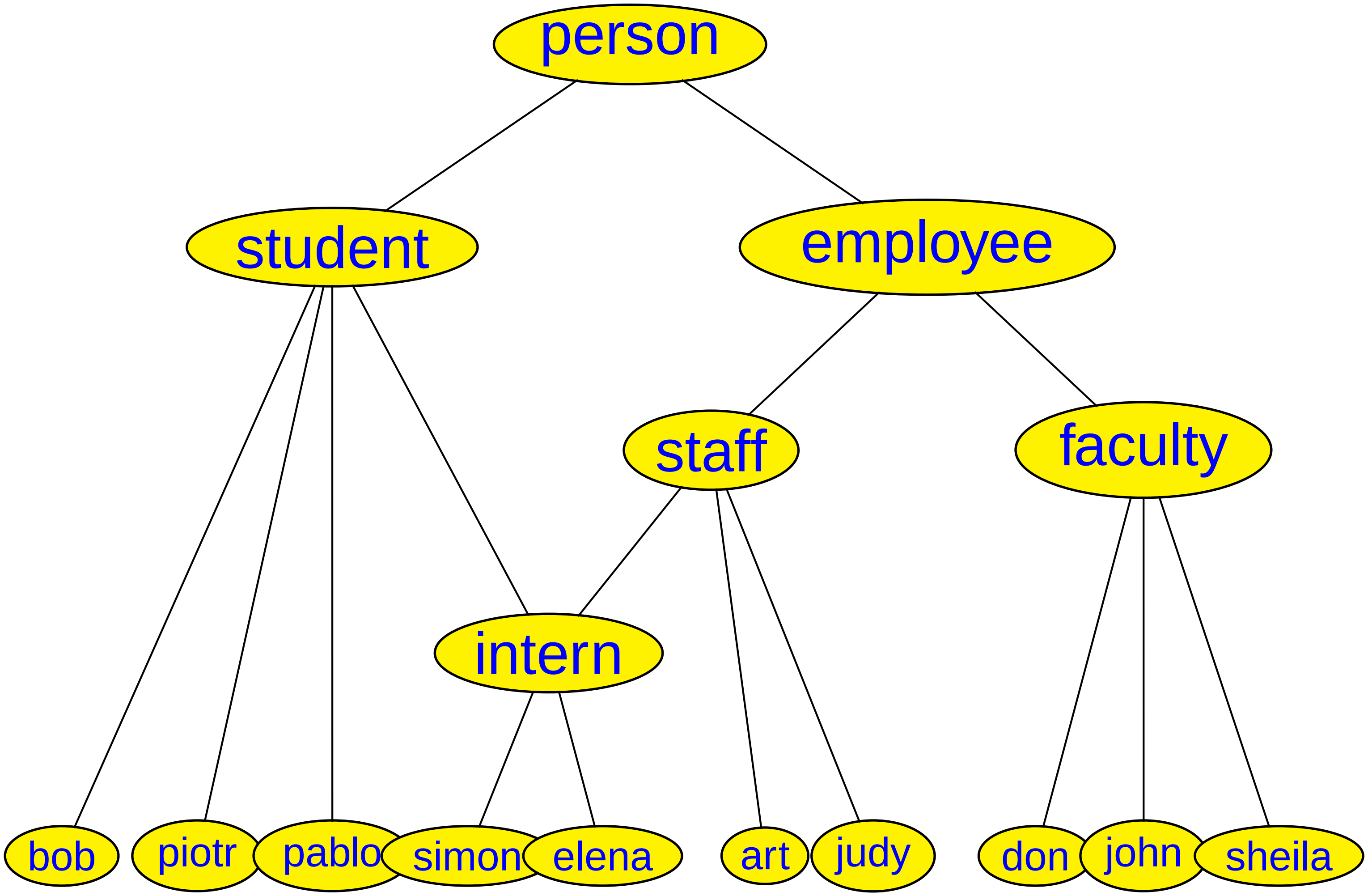
(2) Inconsistent Sort

$$\frac{\phi \ \& \ X : \bot}{X : \bot}$$

(4) Feature Functionality

$$\frac{\phi \ \& \ X.\ell \doteq X' \ \& \ X.\ell \doteq X''}{\phi \ \& \ X.\ell \doteq X' \ \& \ X' \doteq X''}$$

Graphs as constraints—$\mathcal{OSF}$ ***unification = $\mathcal{OSF}$ constraint normalization***



Graphs as constraints—$\mathcal{OSF}$ *unification* = $\mathcal{OSF}$ *constraint normalization*

```
X : student
    (roommate => person(rep => E : employee),
     advisor  => don(secretary => E))
```

&

```
Y : employee
    (advisor  => don(assistant => A),
     roommate => S : student(rep => S),
     helper   => simon(spouse => A))
```

&

```
X = Y
```

Graphs as constraints—$\mathcal{OSF}$ *unification* = $\mathcal{OSF}$ *constraint normalization*

```
X : intern
    (roommate => S : intern(rep => S),
     advisor => don(assistant => A,
                    secretary => S),
     helper => simon(spouse => A))
```

&

```
X = Y
```

&

```
E = S
```

Let $\Sigma \stackrel{\text{DEF}}{=} \biguplus_{n\in\mathbb{N}} \Sigma_n$ be a ranked signature.

The first-order (rational) terms in $\mathcal{T}_{\Sigma,\mathcal{V}}$ are $\mathcal{OSF}$ terms s.t.:

- $\mathcal{S} \stackrel{\text{DEF}}{=} \Sigma \cup \{\top, \bot\}$ is a flat lattice
- $\mathcal{F} \stackrel{\text{DEF}}{=} \mathbb{N} \setminus \{0\}$
- $\textbf{\textit{Arity}}(\top) \stackrel{\text{DEF}}{=} \emptyset$
- $\textbf{\textit{Arity}}(\bot) \stackrel{\text{DEF}}{=} \{i \in \mathbb{N}^* \mid i \leq \max\{n > 0 \mid \Sigma_n \neq \emptyset\}\}$
- $\forall f \in \Sigma_n : \quad \textbf{\textit{Arity}}(f) \stackrel{\text{DEF}}{=} \{1, \ldots, n\}$
- $\forall i \in \mathcal{F} : \quad \textbf{\textit{Dom}}(i) \stackrel{\text{DEF}}{=} \bigcup_{i \leq n} \Sigma_n$
- $\forall i \in \mathcal{F}, \forall f \in \Sigma : \quad \textbf{\textit{Ran}}_f(i) \stackrel{\text{DEF}}{=} \begin{cases} \top & \text{if } f \in \textbf{\textit{Dom}}(i) \\ \bot & \text{otherwise} \end{cases}$

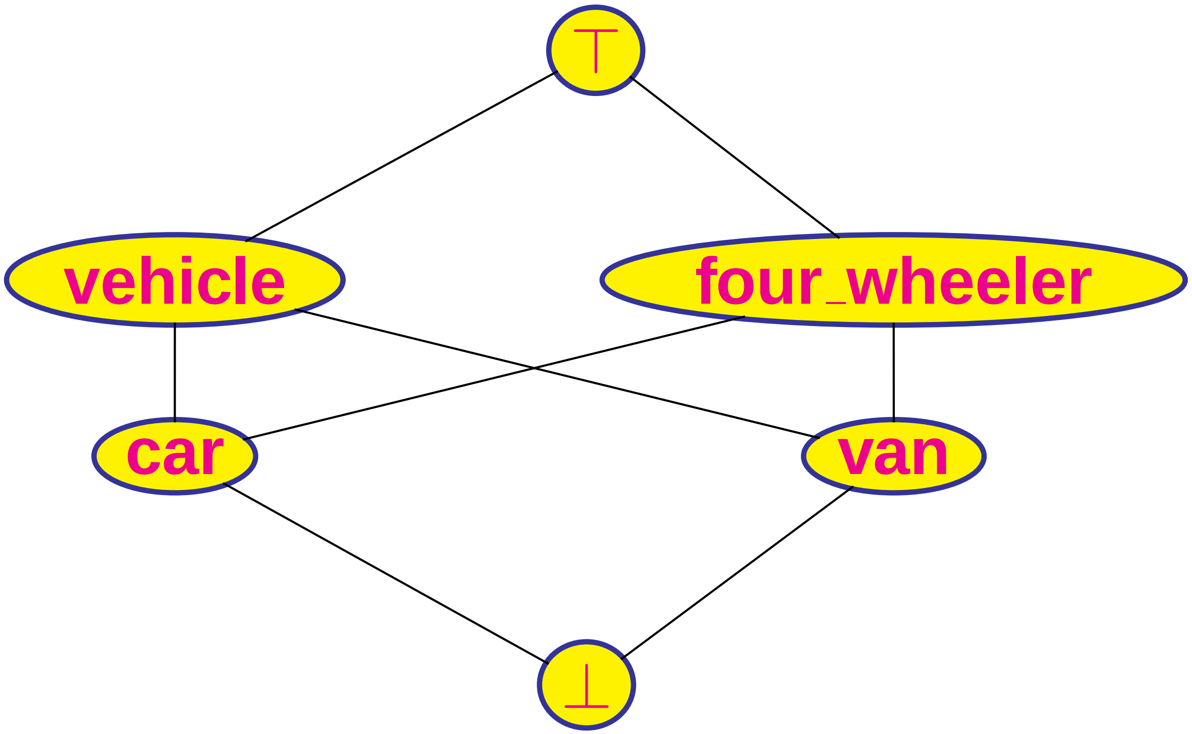
Basic $\mathcal{OSF}$ terms may be extended to express:

- Non-lattice sort signatures
- Disjunction
- Negation
- Partial features
- Extensional sorts (*i.e.*, denoting elements)
- Relational features (*a.k.a.*, “roles”)
- Regular-expression feature paths
- Aggregates (*à la* monoid comprehensions)
- Sort definitions (*a.k.a.*, “$\mathcal{OSF}$ theories”)

Graphs as constraints—*Extended $\mathcal{OSF}$ terms*

$$
\begin{array}{rcl}
\mathtt{OsfTerm} & ::= & [\ \mathtt{Variable:}\ ]\ \mathtt{Term} \\
\mathtt{Term} & ::= & \mathtt{ConjunctiveTerm} \\
 & \mid & \mathtt{DisjunctiveTerm} \\
 & \mid & \mathtt{NegativeTerm} \\
\mathtt{ConjunctiveTerm} & ::= & \mathtt{Sort}\ [\ (\ \mathtt{Attribute}^{+}\ )\ ] \\
\mathtt{Attribute} & ::= & \mathtt{Feature} \Rightarrow \mathtt{OsfTerm} \\
\mathtt{DisjunctiveTerm} & ::= & \{\ \mathtt{OsfTerm}\ [\ ;\ \mathtt{OsfTerm}\ ]^{*}\ \} \\
\mathtt{NegativeTerm} & ::= & \neg\ \mathtt{OsfTerm}
\end{array}
$$

Extended $\mathcal{OSF}$ constraints—*Non-lattice signatures, disjunction*



Non-unique GLBs are disjunctive sorts:

$$\textbf{vehicle} \wedge \textbf{four\_wheeler} \quad = \quad \{\textbf{car}; \textbf{van}\}$$

Syntax of disjunctive $\mathcal{OSF}$ terms:

$$\{\ t_1\ ;\ \ldots\ ;\ t_n\ \}$$

Semantics of disjunctive $\mathcal{OSF}$ terms:

$$[\![\{t_1; \ldots; t_n\}]\!]^{\mathfrak{A},\alpha} \stackrel{\text{DEF}}{=\!=} \bigcup_{1 \leq i \leq n} [\![t_i]\!]^{\mathfrak{A},\alpha}$$

Disjunctive $\mathcal{OSF}$ clauses:

$$\varphi(\{t_1; \ldots; t_n\}) \stackrel{\text{DEF}}{=\!=} \varphi(t_1) \parallel \ \ldots \ \parallel \varphi(t_n)$$

$\mathfrak{A}, \alpha \models \phi_1 \parallel \ \ldots \ \parallel \phi_n$ iff $\mathfrak{A}, \alpha \models \phi_i$ for some $i = 1, \ldots, n$

(5) Non-unique GLB

$$\frac{\phi \;\;\&\;\; X : s \;\;\&\;\; X : s'}{\phi \;\;\&\;\; (X : s_1 \parallel \ldots \parallel X : s_n)}$$

where $\{s_i\}_{i=0}^{n}$

$\stackrel{\text{DEF}}{=} \max_{\leq}\{t \in \mathcal{S} \mid t \leq s$ and $t \leq s'\}$

(6) Distributivity

$$\frac{\phi \;\;\&\;\; (\phi' \parallel \phi'')}{(\phi \;\;\&\;\; \phi') \;\parallel\; (\phi \;\;\&\;\; \phi'')}$$

(7) Disjunction

$$\frac{\phi \parallel \phi'}{\phi}$$

Extended $\mathcal{OSF}$ constraints—*Negation*

Syntax of negative $\mathcal{OSF}$ terms: $\neg t$

Semantics of negative $\mathcal{OSF}$ terms: $[\![\neg t]\!]^{\mathfrak{A}} \stackrel{\text{DEF}}{=} D^{\mathfrak{A}} \setminus [\![t]\!]^{\mathfrak{A}}$

Complemented sorts: $[\![\overline{s}]\!]^{\mathfrak{A}} \stackrel{\text{DEF}}{=} D^{\mathfrak{A}} \setminus [\![s]\!]^{\mathfrak{A}}$

Sorted variable simplification:

$$\varsigma(X : s) \stackrel{\text{DEF}}{=} X : s \qquad \text{if } s \in \mathcal{S}$$

$$\varsigma(X : \overline{\overline{s}}) \stackrel{\text{DEF}}{=} \varsigma(X : s)$$

$$\varsigma(X : \overline{\{s_1; \ldots; s_n\}}) \stackrel{\text{DEF}}{=} \varsigma(X : s_1) \ \& \ \ldots \ \& \ \varsigma(X : s_n)$$

Extended $\mathcal{OSF}$ constraints—*Negative $\mathcal{OSF}$ terms*

Dissolving negative $\mathcal{OSF}$ terms into $\mathcal{OSF}$ clauses eliminates negation:

$$
\begin{array}{rcl}
\varphi(\neg(\neg t)) & \stackrel{\text{DEF}}{=\!=} & \varphi(t) \\[2em]
\varphi(\neg\{t_1; \ldots ; t_n\}) & \stackrel{\text{DEF}}{=\!=} & \varphi(\neg t_1) \ \& \ \ldots \ \& \ \varphi(\neg t_n) \\[2em]
\varphi(\neg X : s(\ell_i \Rightarrow t_i)_{i=1}^{n}) & \stackrel{\text{DEF}}{=\!=} & \varsigma(X : \overline{s}) \\[1em]
& \| & X.\ell_1 \doteq X_1 \ \& \ \varphi(\neg t_1) \\
& \| & X.\ell_1 \doteq X_1' \ \& \ X_1' \dot{\neq} X_1 \ \& \ \varphi(t_1) \\
& & \ldots \\
& \| & X.\ell_n \doteq X_n \ \& \ \varphi(\neg t_n) \\
& \| & X.\ell_n \doteq X_n' \ \& \ X_n' \dot{\neq} X_n \ \& \ \varphi(t_n)
\end{array}
$$

(8) Variable Disequality

$$\frac{\phi \;\&\; X \dot{\neq} X}{\bot}$$

(9) Sort Complement

$$\frac{\phi \;\&\; X : \overline{s}}{\phi \;\&\; X : s'} \qquad \text{if } s' \in \max_{\leq}\{t \in \mathcal{S} \mid s \not\leq t \text{ and } t \not\leq s\}$$

Partial features have restricted domains:

$$\exists y, \quad y = \ell(x) \text{ only if } x \in \textbf{\textit{Dom}}(\ell)$$

Declaring partial feature domains:

$$\textbf{\textit{Dom}} : \mathcal{F} \mapsto \mathbf{2}^{\mathcal{S}}$$

s.t. $\textbf{\textit{Dom}}(\ell) \stackrel{\text{DEF}}{=\!=}$ set of maximal sorts where ℓ is defined. Can also declare a feature's range: $\textbf{\textit{Ran}}_s : \mathcal{F} \mapsto \mathcal{S}$ for $s \in \textbf{\textit{Dom}}(\ell)$.

(10) Partial Feature

$$\frac{\phi \;\&\; X.\ell \doteq X'}{\phi \;\&\; X.\ell \doteq X' \;\&\; X : s \;\&\; X' : s'} \quad \begin{array}{ll} \text{if} & s \in \textbf{\textit{Dom}}(\ell) \\ \text{and} & \textbf{\textit{Ran}}_s(\ell) = s' \end{array}$$

Extended $\mathcal{OSF}$ constraints—*Partial features (example)*

Assume $\{nil, cons, list\} \subseteq \mathcal{S}$ such that:

$$\begin{array}{lcl} nil & < & list \\ cons & < & list \end{array}$$

and $\{hd, tl\} \subseteq \mathcal{F}$ such that:

$$\begin{array}{lcl} \textbf{\textit{Dom}}(hd) & \stackrel{\text{DEF}}{=\!=} & \{cons\} \\ \textbf{\textit{Dom}}(tl) & \stackrel{\text{DEF}}{=\!=} & \{cons\} \end{array}$$

then:

$$\begin{array}{lcl} list(tl \Rightarrow X) & \leadsto & cons(tl \Rightarrow X) \\ int(tl \Rightarrow X) & \leadsto & \perp \end{array}$$

The fact that some sorts denote singletons (*e.g.*, numbers) is not part of our axioms so far!

i.e.,

$$f(a \Rightarrow 1, b \Rightarrow 1) \not\leq f(a \Rightarrow X, b \Rightarrow X)$$

because:

$$f(a \Rightarrow X : s, b \Rightarrow X' : s) \leq f(a \Rightarrow Y, b \Rightarrow Y) \quad \text{iff} \quad X = X'$$

A sort that denotes a singleton, whenever all its images by a specific set of features do, is called extensional.

Extensional sorts are element constructors.

Let $\mathcal{E} \subseteq Minimals(\mathcal{S})$ be the set of extensional sorts with rank function:

$$\boldsymbol{Arity} : \mathcal{E} \mapsto \mathbf{2}^{\mathcal{F}}$$

e.g.:

$$\begin{array}{lcl} \boldsymbol{Arity}(n) & = & \emptyset \quad \forall n \in \mathbb{N} \\ \boldsymbol{Arity}(nil) & = & \emptyset \\ \boldsymbol{Arity}(cons) & = & \{hd, tl\} \end{array}$$

Extensional sorts obey an axiom reminiscent of the axiom of functionality; *viz.*,

$$\begin{array}{ll} \text{if} & \textbf{\textit{Arity}}(f) = n \;\text{ and }\; X_i = Y_i \;\; (\forall i = 1, \ldots, n) \\ \text{then} & f(X_1, \ldots, X_n) \;=\; f(Y_1, \ldots, Y_n) \end{array}$$

(11) Weak Extensionality

$$\frac{\phi \;\; \& \;\; X : s \;\; \& \;\; X' : s}{\phi \;\; \& \;\; X : s \;\; \& \;\; X \doteq X'} \qquad \begin{array}{l} \text{if } s \in \mathcal{E} \text{ and } \forall \ell \in \textbf{\textit{Arity}}(s)\text{:} \\ \{X.f \doteq Y, X'.f \doteq Y\} \subseteq \phi \end{array}$$

The Weak Extensionality rule works, but not for cyclic terms; *viz.*:

$$\text{let } s \in \mathcal{E} \text{ and } \textit{\textbf{Arity}}(s) = \{\ell\}$$

$$\text{then } X : s(\ell \Rightarrow X) \;\&\; X' : s(\ell \Rightarrow X')$$

$$\text{or } X : s(\ell \Rightarrow X') \;\&\; X' : s(\ell \Rightarrow X)$$

are not reduced! So we need a stronger condition for cycles.

Proceed coinductively from roots to leaves carrying a context Γ, a set of pairs $s/\{X_1, \ldots, X_n\}$ s.t. $X_i \in \mathcal{V}$ $(i = 1, \ldots, n)$ and $s \in \mathcal{E}$ occurs at most once in Γ:

(12) Extensional Occurrence

$$\frac{\Gamma \uplus \{s/V, \ldots,\} \vdash \phi \ \& \ X : s}{\Gamma \uplus \{s/V \cup \{X\}, \ldots\} \vdash \phi \ \& \ X : s}$$

if $s \in \mathcal{E}$ and $X \notin V$
and $\forall f \in \textbf{\textit{Arity}}(s)$:
$\{X.f \doteq X', X' : s'\} \subseteq \phi$
with $s' \in \mathcal{E}$

(13) Strong Extensionality

$$\frac{\Gamma \uplus \{s/\{X, X', \ldots\}\} \vdash \phi}{\Gamma \uplus \{s/\{X, \ldots\}\} \vdash \phi \ \& \ X \doteq X'}$$

if $s \in \mathcal{E}$

Extended $\mathcal{OSF}$ constraints—*Relational features and aggregation*

Relational features are set-valued features:

$$\forall \langle x, y\rangle \in A \times B : \quad \langle x, y\rangle \in R \quad \text{iff} \quad y \in R[x] \quad \text{iff} \quad x \in R^{-1}[y]$$

Sets are a particular case of monoidal aggregates:

- the notation "$X : s$" is generalized to carry an optional value $e \in \mathcal{E}$
- "$X = e : s$" means "X has value e of sort s" $(X \in \mathcal{V},\ \ e \in \mathcal{E},\ \ s \in \mathcal{S})$
- the shorthand "$X = e$" means "$X = e : \top$"
- when the sort $s \in \mathcal{S}$ denotes a commutative monoid $\langle \star, \mathbf{1}_\star\rangle$, the shorthand "$X : s$" means "$X = \mathbf{1}_\star : s$."

The semantic conditions are thus extended:

$$\mathfrak{A}, \alpha \models X = e : s \text{ iff } e^{\mathfrak{A}} \in s^{\mathfrak{A}} \text{ and } \alpha(X) = e^{\mathfrak{A}}$$

(14) Value Aggregation

$$\frac{\phi \;\&\; X = e : s \;\&\; X = e' : s'}{\phi \;\&\; X = e \star e' : s \wedge s'}$$

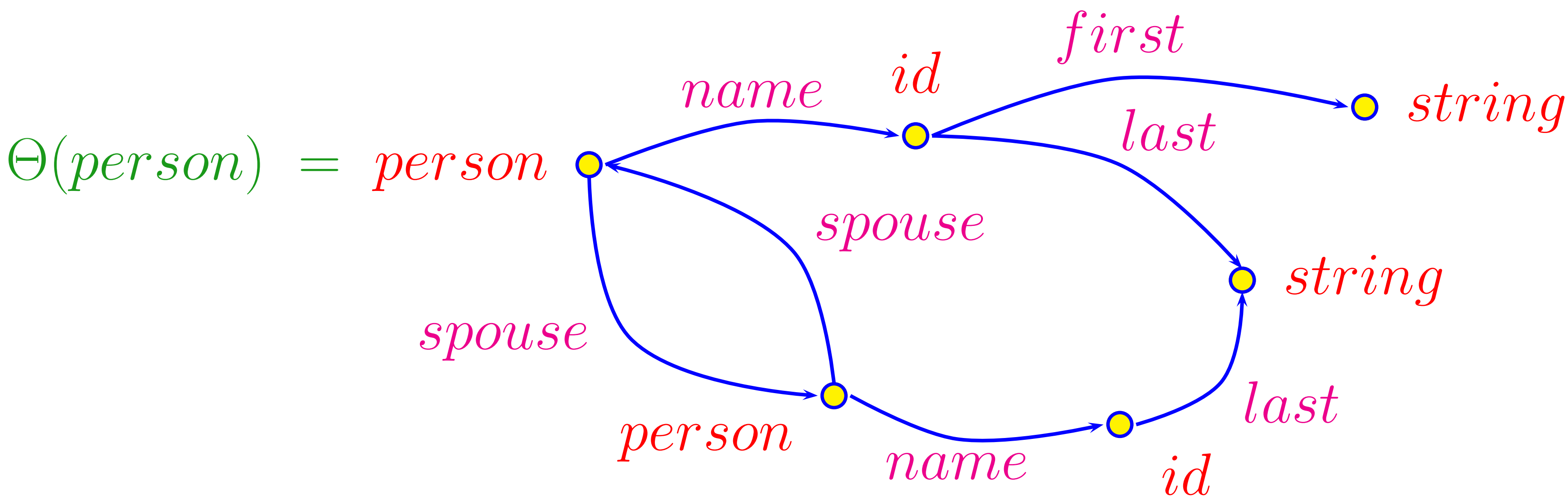
if s and s' are both subsorts of commutative monoid $\langle \star, \mathbf{1}_\star \rangle$

N.B.: This works for any commutative monoid—not just sets!

IDEA: Augment the sort ordering with constraints imposing:

- sorts of features
- coreference equations

e.g., define the sort person to abide by the structure:



An $\mathcal{OSF}$ theory is a function: $\Theta : \mathcal{S} \mapsto \Psi$

An $\mathcal{OSF}$ theory is order-consistent iff it is monotonic:

$$s \leq s' \;\Rightarrow\; \Theta(s) \leq \Theta(s')$$

$\mathcal{OSF}$ theory unification problem:

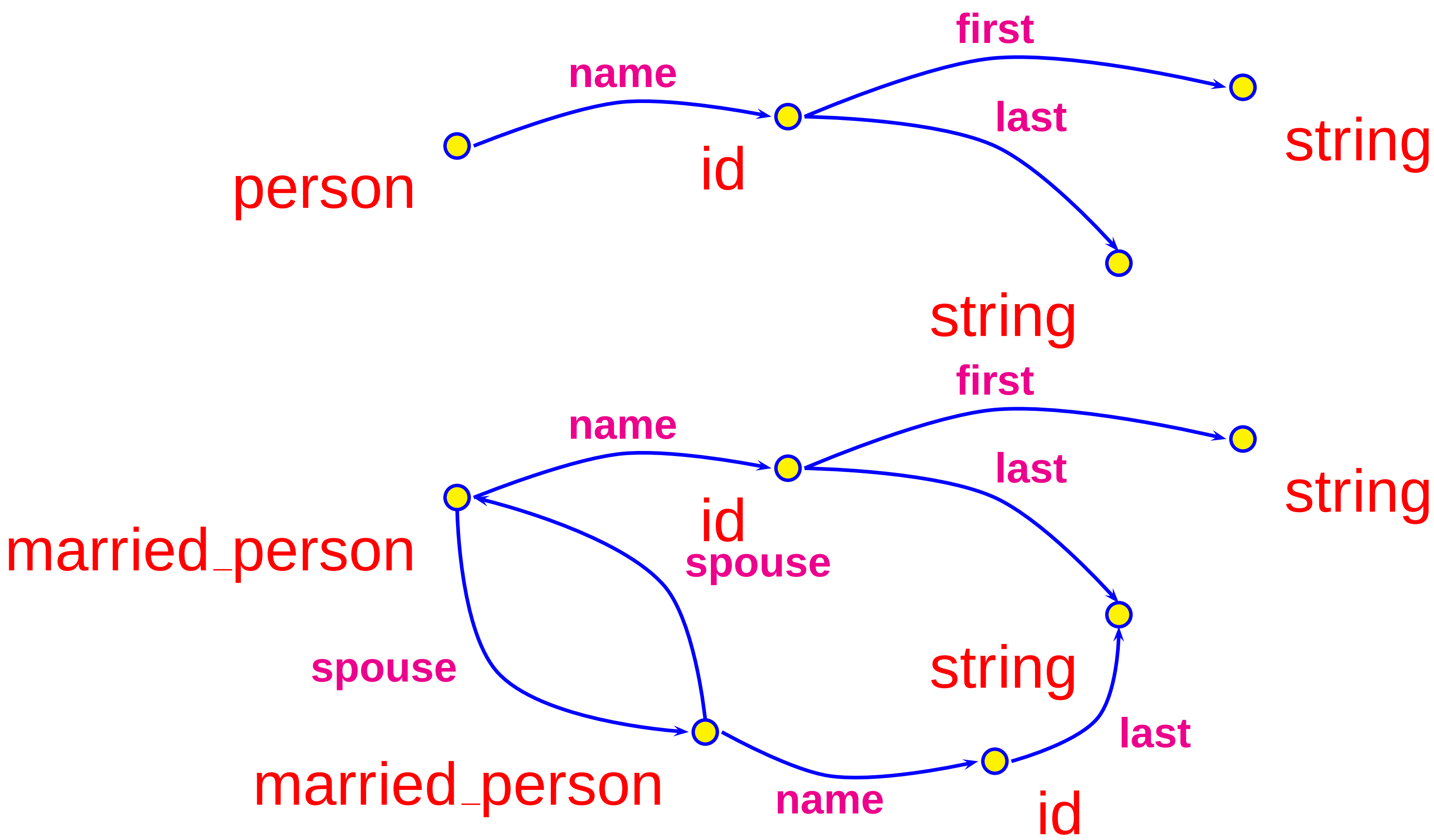
> Given an order-consistent $\mathcal{OSF}$ theory Θ, normalize any term of sort s taking into account the $\mathcal{OSF}$ constraints $\Theta(s)$.

Theorem $\mathcal{OSF}$ theory unification is undecidable.

However... there is an algorithm such that:

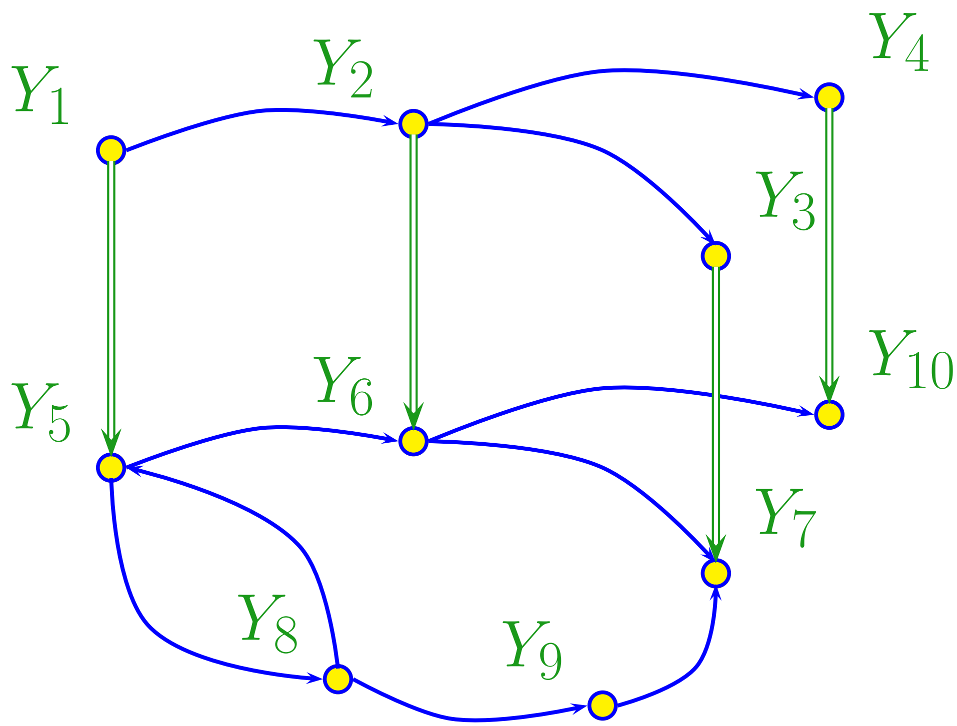
- inconsistent terms are always normalized to $\perp$ in finitely many steps;
- normalization can perform $\mathcal{OSF}$ constraint inheritance from the theory lazily;
- there is an efficient algorithm which is complete for a large class of $\mathcal{OSF}$ theories;
- only one rule completes it (and may cause divergence).

Extended $\mathcal{OSF}$ constraints—$\mathcal{OSF}$ *theory unification (example)*

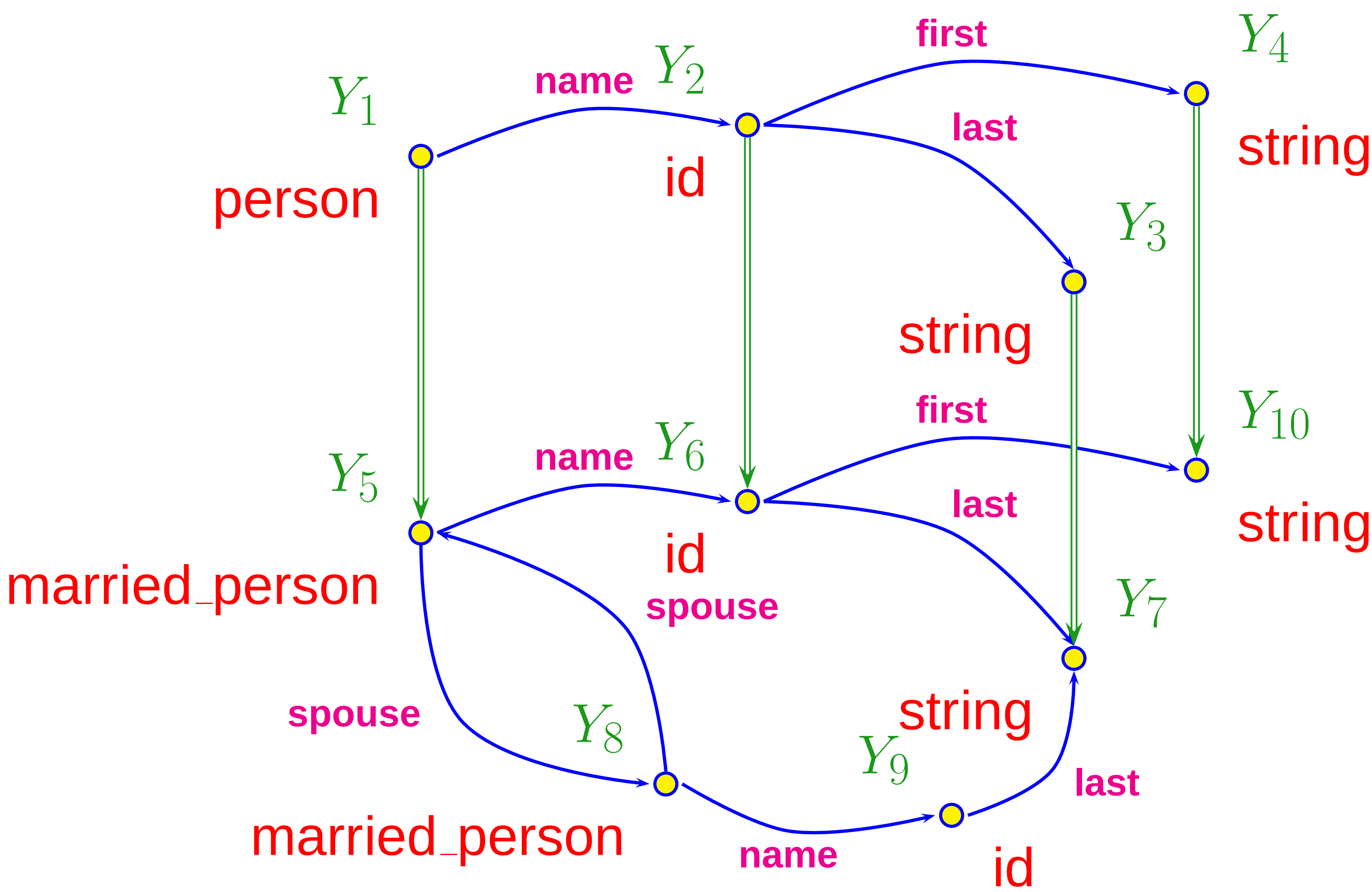


The fact that an $\mathcal{OSF}$ theory is order-consistent yields an endomorphic mapping of theory variables.

In particular, the sort ordering $\leq$ and the GLB operation $\wedge$ extend homomorphically to all theory variables.



Extended $\mathcal{OSF}$ constraints—$\mathcal{OSF}$ *theory unification*



Normalizing:

$$
\begin{array}{ll}
 & P : person(name \Rightarrow \top(last \Rightarrow \textit{"Smith"})) \\
\& & P : married\_person(spouse \Rightarrow Q) \\
\& & Q : person(name \Rightarrow id(last \Rightarrow S))
\end{array}
$$

yields, among other things:

$$
\begin{array}{ll}
 & P : married\_person \\
\& & Q : married\_person \\
\& & S : \textit{"Smith"} \ldots
\end{array}
$$

(0) Frame Allocation

$$\frac{\Gamma \qquad\qquad\qquad\qquad \vdash\ X : s\ \&\ \phi}{\Gamma \bigcup \{\{X\backslash Y_s\}\} \vdash\ X : s\ \&\ \phi} \qquad \text{if } \forall s' \in \mathcal{S},\ \forall F \in \Gamma :\ X\backslash Y_{s'} \notin F$$

(1) Sort Intersection

$$\frac{\Gamma \bigcup \{\{X \backslash Y_{s'}\} \cup F\} \quad \vdash \ X : s \ \& \ X : s' \ \& \ \phi}{\Gamma \bigcup \{\{X \backslash Y_{s \wedge s'}\} \cup F\} \vdash \ X : s \wedge s' \ \& \ \phi}$$

(2) Inconsistent Sort

$$\frac{\Gamma \bigcup \{\{X \backslash Y_{\perp}\} \cup F\} \vdash \ \phi}{\emptyset \qquad\qquad\qquad\qquad \vdash \ \perp}$$

(3) Variable Elimination

$$\frac{\Gamma \vdash X \doteq X' \;\&\; \phi}{\Gamma[X'/X] \vdash X \doteq X' \;\&\; \phi[X'/X]}$$

if $X \neq X'$
and $X \in \textbf{\textit{Var}}(\Gamma) \cup \textbf{\textit{Var}}(\phi)$

(4) Feature Functionality

$$\frac{\Gamma \vdash X.\ell \doteq X' \;\&\; X.\ell \doteq X'' \;\&\; \phi}{\Gamma \vdash X.\ell \doteq X' \;\&\; X' \doteq X'' \;\&\; \phi}$$

(5) Feature Inheritance (if $\ell(Y) = Y'$ and $X'\backslash Y' \notin F$)

$$\frac{\Gamma \bigcup \{\{X\backslash Y\} \cup F\} \quad \vdash \phi \;\&\; X.\ell \doteq X'}{\Gamma \bigcup \{\{X\backslash Y, X'\backslash Y'\} \cup F\} \vdash \phi \;\&\; X.\ell \doteq X' \;\&\; X' : \textbf{\textit{Sort}}(Y')}$$

(6) Frame Merging

$$\frac{\Gamma \bigcup \{\{X\backslash Y_s\} \cup F, \{X\backslash Y_{s'}\} \cup F'\} \vdash \phi}{\Gamma \bigcup \{\{X\backslash Y_{s \wedge s'}\} \cup F \cup F'\} \quad \vdash \phi}$$

(7) Frame Reduction

$$\frac{\Gamma \bigcup \{\{X \backslash Y, X \backslash Y'\} \cup F\} \ \vdash \ \phi}{\Gamma \bigcup \{\{X \backslash Y\} \cup F\} \qquad \vdash \ \phi} \qquad \text{if } Y \leq Y'$$

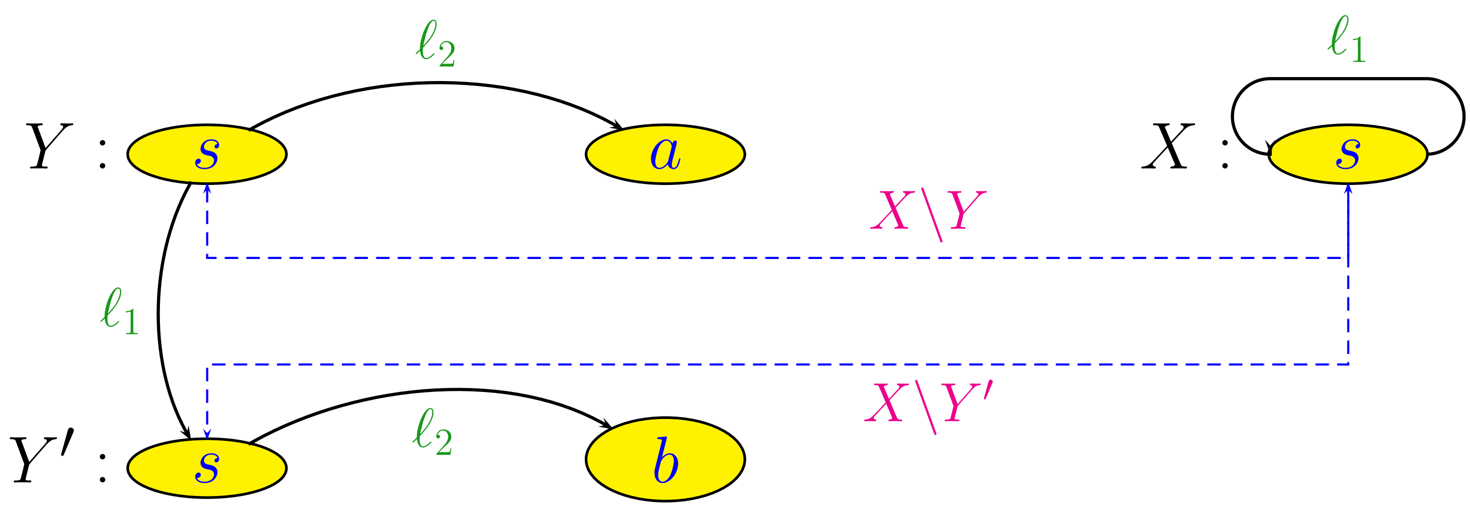
(8) Theory Coreference

$$\frac{\Gamma \bigcup \{\{X \backslash Y, X' \backslash Y\} \cup F\} \ \vdash \ \phi}{\Gamma \bigcup \{\{X \backslash Y\} \cup F\} \qquad \vdash \ \phi \ \& \ X \doteq X'}$$

(9) Theory Feature Completion

$$\frac{\Gamma \ \vdash \ \phi}{\Gamma \ \vdash \ X.\ell \doteq Z \ \& \ \phi}$$

if $X \backslash Y \in F$ for some $F \in \Gamma$
and $X \backslash Y' \in F'$ for some $F' \in \Gamma$
and both $\ell(Y)$, $\ell(Y')$ exist
and Z is new



Order-sorted featured graph constraints—*(Summary)*

We have overviewed a formalism of objects where:

- “real-life” objects are viewed as logical constraints
- objects may be approximated as set-denoting constructs
- object normalization rules provide an efficient operational semantics
- consistency extends unification (and thus matching)
- this enables rule-based computation (whether rewrite or logical rules) over general graph-based objects
- this yield a powerful means for effectively using ontologies

$\mathcal{OSF}$ Graph Constraint Formalism—*Outline*

- Semantic Web formalisms
- Graphs as constraints
- $\mathcal{OSF}$ *vs.* $\mathcal{DL}$
- $\mathcal{LIFE}$—logically and functionally constrained sorted graphs
- Recapitulation

Understanding $\mathcal{OWL}$ amounts to reasoning with knowledge expressed as $\mathcal{OWL}$ *sentences*. Its $\mathcal{DL}$ semantics relies on *explicitly* building models using *induction*.

ergo:

Inductive techniques are *eager* and (thus) *wasteful*

Reasoning with knowledge expressed as constrained ($\mathcal{OSF}$) *graphs* relies on *implicitly* pruning inconsistent elements using *coinduction*.

ergo:

Coinductive techniques are *lazy* and (thus) *thrifty*

- Semantic Web formalisms
- Graphs as constraints
- $\mathcal{OSF}$ vs. $\mathcal{DL}$
- $\mathcal{LIFE}$—logically and functionally constrained sorted graphs
- Recapitulation

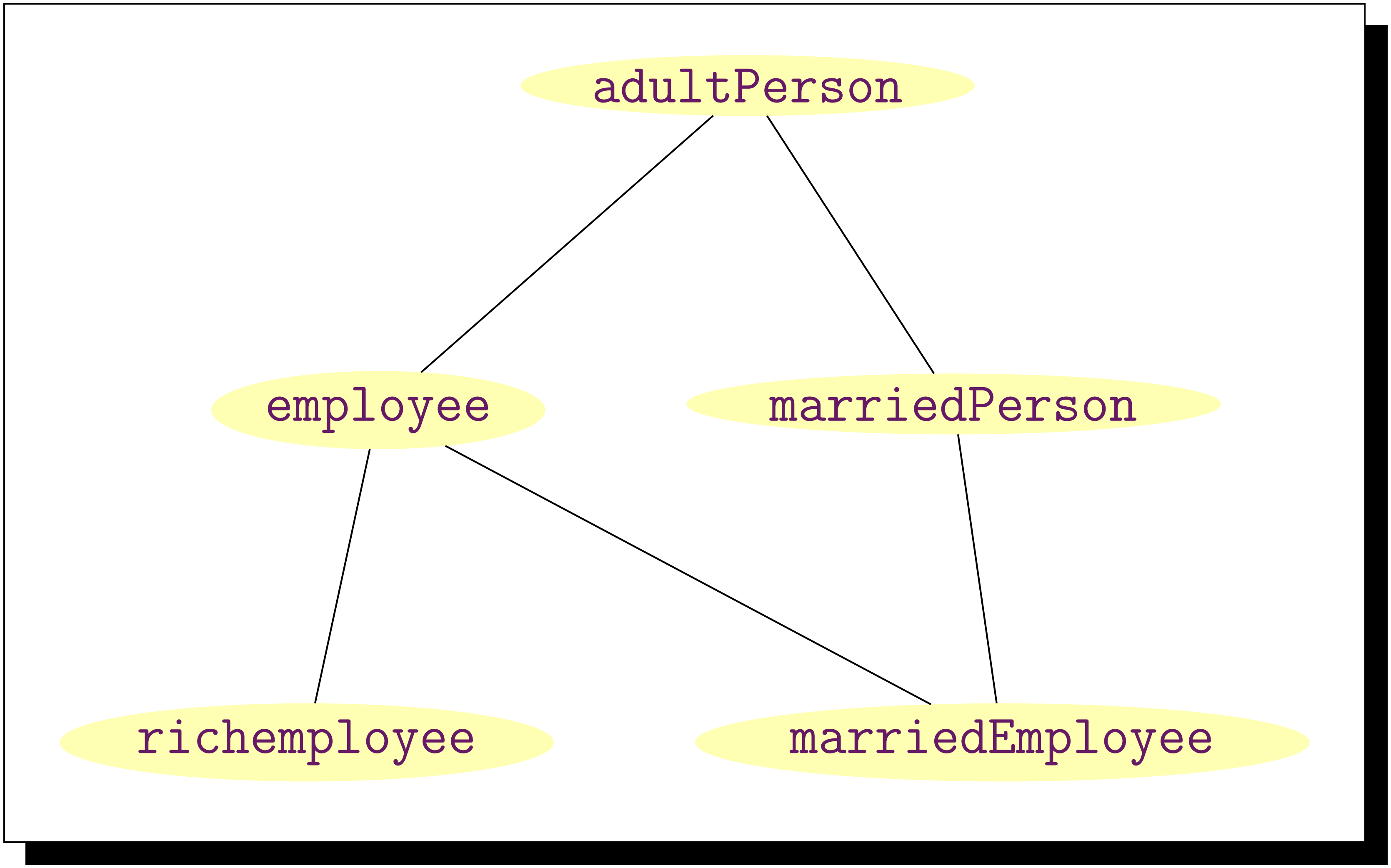
$\mathcal{LIFE}$—$\mathcal{L}$ogic, $\mathcal{I}$nheritance, $\mathcal{F}$unctions, and $\mathcal{E}$quations

$\mathcal{CLP}(\chi)$—$\mathcal{C}$onstraint, $\mathcal{L}$ogic, $\mathcal{P}$programming, parameterized over is a constraint system χ

$\mathcal{LIFE}$ is a $\mathcal{CLP}$ system over $\mathcal{OSF}$ constraints and functions over them (rewrite rules); namely:

$$\boxed{\mathcal{LIFE} = \mathcal{CLP}(\mathcal{OSF} + \mathcal{FP})}$$

$\mathcal{LIFE}$—logically and functionally constrained sorted graphs



A multiple-inheritance hierarchy

The same hierarchy in Java

```
interface adultPerson {
   name id;
   date dob;
   int age;
   String ssn;
}
interface employee extends adultPerson {
   title position;
   String institution;
   employee supervisor;
   int salary;
}
interface marriedPerson extends adultPerson {
   marriedPerson spouse;
}
interface marriedEmployee extends employee, marriedPerson {
}
interface richEmployee extends employee {
}
```

The same hierarchy in $\mathcal{LIFE}$

```
employee <: adultPerson.
marriedPerson <: adultPerson.
richEmployee <: employee.
marriedEmployee <: employee.
marriedEmployee <: marriedPerson.

:: adultPerson    ( id => name
                  , dob => date
                  , age => int
                  , ssn => string ).

:: employee       ( position => title
                  , institution => string
                  , supervisor => employee
                  , salary => int ).

:: marriedPerson ( spouse => marriedPerson ).
```

A relationally and functionally constrained $\mathcal{LIFE}$ sort hierarchy

$$
\begin{array}{lll}
:: \ P : \texttt{adultPerson} & (& \texttt{id} \Rightarrow \texttt{name} \\
 & , & \texttt{dob} \Rightarrow \texttt{date} \\
 & , & \texttt{age} \Rightarrow A : \texttt{int} \\
 & , & \texttt{ssn} \Rightarrow \texttt{string} \ ) \\
\multicolumn{3}{l}{\quad \mid \ A = \texttt{ageInYears}(P), \ A \geq 18.}
\end{array}
$$

$$
\begin{array}{lll}
:: \ \texttt{employee} & (& \texttt{position} \Rightarrow T : \texttt{title} \\
 & , & \texttt{institution} \Rightarrow \texttt{string} \\
 & , & \texttt{supervisor} \Rightarrow E : \texttt{employee} \\
 & , & \texttt{salary} \Rightarrow S : \texttt{int} \ ) \\
\multicolumn{3}{l}{\quad \mid \ \texttt{higherRank}(E.\texttt{position}, T) \ , \ E.\texttt{salary} \geq S.}
\end{array}
$$

A relationally and functionally constrained $\mathcal{LIFE}$ sort hierarchy

$$:: \; M : \texttt{marriedPerson} \; (\; \texttt{spouse} \Rightarrow P : \texttt{marriedPerson} \;)$$
$$\mid \; P.\texttt{spouse} = M.$$

$$:: \; R : \texttt{richEmployee} \; (\; \texttt{institution} \Rightarrow I \; , \; \texttt{salary} \Rightarrow S \;)$$
$$\mid \; \texttt{stockValue}(I) = V \; , \; \texttt{hasShares}(R, I, N) \; , \; S + N * V \geq 200000.$$

Proof "memoizing"

$\mathcal{OSF}$ constraints as syntactic variants of logical formulae:

Sorts are unary predicates: $X : s \iff [\![s]\!]([\![X]\!])$

Features are unary functions: $X.f \doteq Y \iff [\![f]\!]([\![X]\!]) = [\![Y]\!]$

Coreferences are equations: $X \doteq Y \iff [\![X]\!] = [\![Y]\!]$

So ...

Why not use (good old) logic proofs instead?

Proof "memoizing"

But: $\boxed{\textbf{model equivalence} \neq \textbf{proof equivalence!}}$

- $\mathcal{OSF}$-unification proves sort constraints by reducing them monotonically w.r.t. the sort ordering
- *ergo*, once $X : s$ has been proven, the proof of $s(X)$ is recorded as *the sort "s" itself!*
- if further down a proof, it is again needed to prove $X : s$, it is remembered as X's binding
- Indeed, $\mathcal{OSF}$ ***constraint proof rules ensure that:***

no type constraint is ever proved twice

Proof "memoizing"

$\mathcal{OSF}$ type constraints are incrementally "*memoized*" as they are verified:

sorts act as (instantaneous!) proof caches!

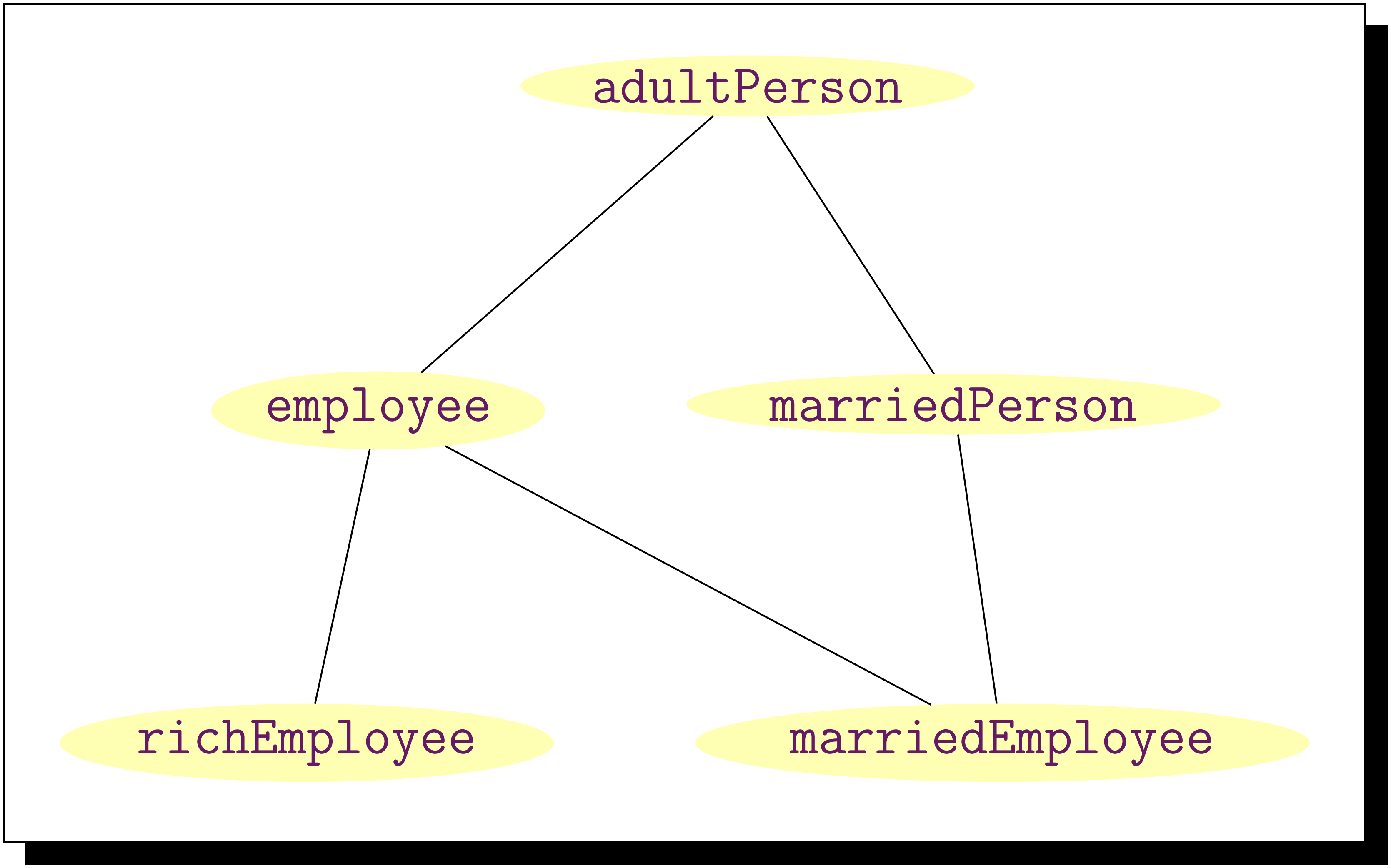
whereas in logic having proven $s(X)$ is not *"remembered"* in any way (*e.g.*, Prolog)

Example: consider the $\mathcal{OSF}$ constraint conjunction:

- $X : \texttt{adultPerson}(\texttt{age} \Rightarrow \texttt{25})$,
- $X : \texttt{employee}$,
- $X : \texttt{marriedPerson}(\texttt{spouse} \Rightarrow Y)$.

Notation: $\texttt{type}\#(\texttt{condition})$ means "*constraint* $\texttt{condition}$ *attached to sort* $\texttt{type}$"

Proof "memoizing"—Example hierarchy reminded



Proof "memoizing"

1. proving: $X : \texttt{adultPerson}(\texttt{age} \Rightarrow \texttt{25})$ ...
2. proving: $\texttt{adultPerson\#}(X.\texttt{age} \geq \texttt{18})$ ...
3. proving: $X : \texttt{employee}$ ...
4. proving: $\texttt{employee\#}(\texttt{higherRank}(E.\texttt{position}, P))$ ...
5. proving: $\texttt{employee\#}(E.\texttt{salary} \geq S)$ ...
6. proving: $X : \texttt{marriedPerson}(\texttt{spouse} \Rightarrow Y)$ ...
7. proving: $X : \texttt{marriedEmployee}(\texttt{spouse} \Rightarrow Y)$ ...
8. proving: $\texttt{marriedEmployee\#}(Y.\texttt{spouse} = X)$ ...

Therefore, ***all other inherited conditions coming from a sort greater than*** `marriedEmployee` ***(such as*** `employee` ***or*** `adultPerson`***) can be safely ignored!***

Proof "memoizing"

This "*memoizing*" property of $\mathcal{OSF}$ constraint-solving enables:

using rules over ontologies

as well as, *conversely*,

enhancing ontologies with rules

Indeed, with $\mathcal{OSF}$:

- ***concept ontologies may be used as constraints by rules*** for inference and computation
- ***rule-based conditions in concept definitions may be used*** to magnify expressivity of ontologies thanks to the ***proof-memoizing property of ordered sorts***

- Semantic Web formalisms
- Graphs as constraints
- $\mathcal{OSF}$ vs. $\mathcal{DL}$
- $\mathcal{LIFE}$: logically and functionally constrained sorted graphs
- Recapitulation

Recapitulation—*what you must remember from this talk. . .*

- Objects are ***graphs***
- Graphs are ***constraints***
- Constraints are ***good***: they provide both ***formal*** theory and ***efficient*** processing
- ***Formal Logic*** is ***not*** all there is
- even so: ***model*** theory $\neq$ ***proof*** theory
- indeed, due to its youth, much of W3C technology is often ***naïve*** in conception and design

  Ergo. . . it is condemned to reinventing [square!] wheels as long as it does not realize that such issues have been studied in depth for the past 50 years in theoretical CS!

Example of W3C's "reinvention of square wheels:"

- ***structure-sharing*** in trees and graphs
- ***WAM-style compilation*** of trees and graphs as triple-based machine instructions
- ***local/global name*** scoping management
- ***types*** as used in programming and logic

Example of W3C's "reinvention of square wheels"—ctd:

- ***syntax***: What's ***essential***?
  What's ***superfluous***?

  confusing notation : XML-based cluttered verbosity

  ok, not for human consumption—but still!

- ***semantics***: What's a ***model*** good for?
  What's (efficiently) ***provable***?
  decidable $\neq$ ***efficient***
  undecidable $\neq$ ***inefficient***

- . . .

- **Linked data** represents all information as interconnected sorted labelled $\mathcal{RDF}$ graphs—it has become a universal *de facto* knowledge model standard
- **Differences between $\mathcal{DL}$ and $\mathcal{OSF}$ can come handy**:
  - $\mathcal{DL}$ is expansive—therefore, expensive—and can only describe finitely computable sets; whereas,
  - $\mathcal{OSF}$ is contractive—therefore, efficient—and can also describe recursively-enumerable sets
- **$\mathcal{OSF}$-Constraint Solving enables practical KR**:
  - ***structural:*** objects, classes, inheritance
  - ***non-structural:*** path equations, relational constraints, type definitions

Recapitulation—*what you must remember from this talk. . .*

It is exciting to see the prospects of the W3C. . . so. . .
. . . it is time that the SW effort rely on mature science!

Whatever the situation may be, we shall live the consequences of this Darwinian survival of the fittest:

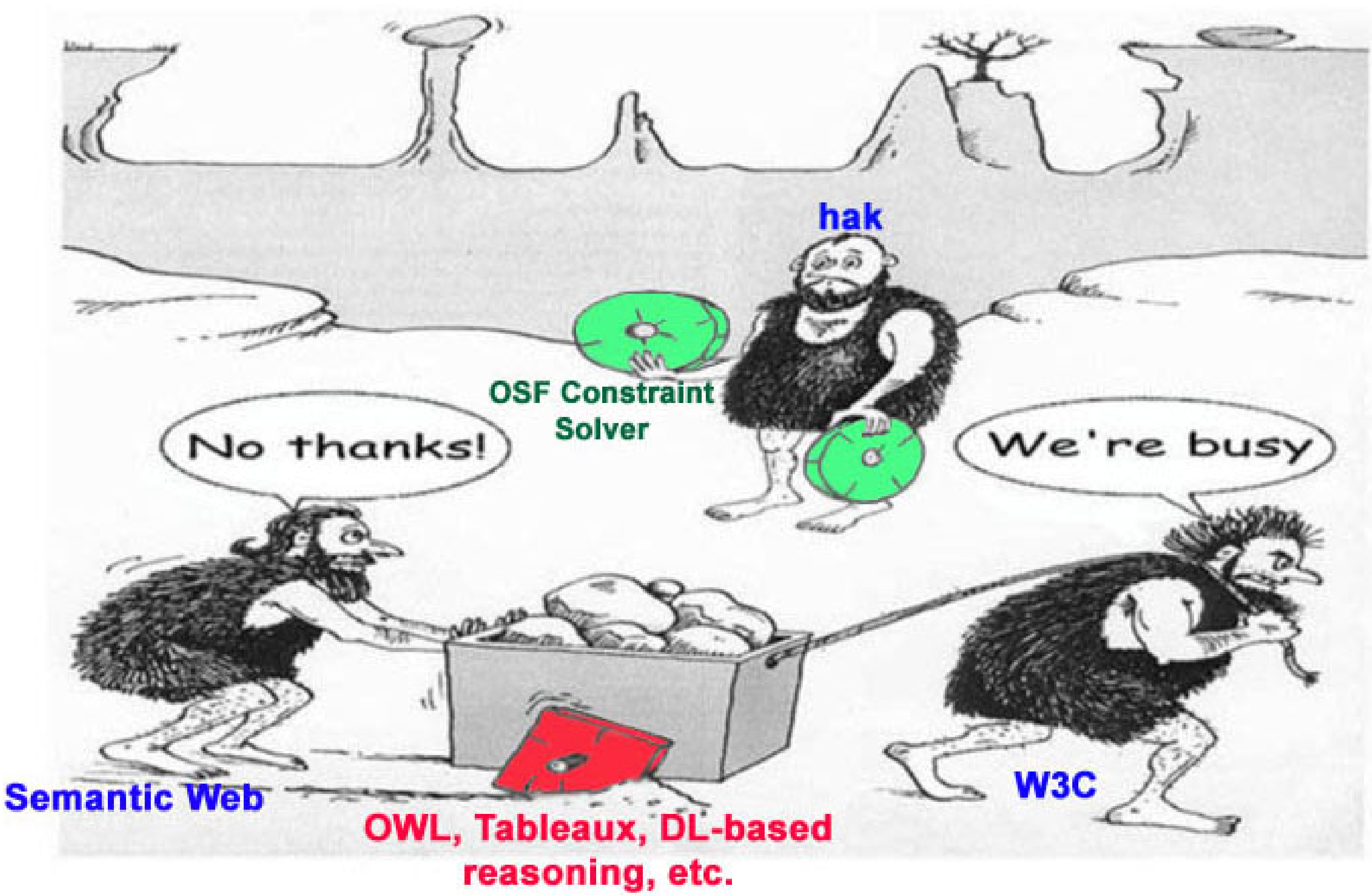
> ***May the most appropriate win!. . .***

The paradox of culture is that language [...] is too linear, not comprehensive enough, too slow, too limited, too constrained, too unnatural, too much a product of its own evolution, and too artificial. This means that [man] must constantly keep in mind the limitations language places upon him.

Edward T. Hall—*Beyond Culture*

Innovation takes courage. . . *(from Martin Wildberger's "Smarter Planet" Keynote, CASCON 2009)*

If I'd asked my customers what they wanted, they'd have said a faster horse!*—Henry Ford*



Thank You For Your Attention !

For more information:

- http://portal.acm.org/citation.cfm?id=1467247.1467250
- http://www.facebook.com/video/video.php?v=375500157155
- http://www.cs.brown.edu/people/pvh/CPL/Papers/v1/hak.pdf

- hak@ca.ibm.com
- http://www.linkedin.com/in/hak2007