

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
- ▶ *OSF* vs. *DL*
- ▶ *LIFE*—logically and functionally constrained sorted graphs
- ▶ Recapitulation

- ▶ Semantic Web formalisms
- ▶ Graphs as constraints
- ▶ *OSF* vs. *DL*
- ▶ *LIFE*—logically and functionally constrained sorted graphs
- ▶ Recapitulation

▶ *OWL* technology has become its *de facto* standard

▶ Everyone talks about *OWL* dialects!

The whole World-Wide Web is abuzz with *OWL-this* and *OWL-that*, ...

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

SHIF, SHIQ, SHOQ(D), SHOIQ, SRIQ, SROIQ, ...

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

Semantic Web formalisms—*OWL* speaks

What language(s) do *OWL*'s speak?—a confusing growing crowd of strange-sounding languages and logics:

- *OWL*, *OWL* Lite, *OWL* DL, *OWL* Full
- *DL*, *DLR*, ...
- *AL*, *ALC*, *ALCN*, *ALCNR*, ...
- *SHIF*, *SHIN*, *CIQ*, *SHIQ*, *SHOQ(D)*, *SHOIQ*, *SRIQ*, *SROIQ*, ...

Depending on whether the system allows:

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

Semantic Web formalisms—*DL* dialects

For better or worse, the W3C has married its efforts to *DL*-based reasoning systems:

- ▶ All the proposed *DL* Knowledge Base formalisms in the *OWL* family use *tableaux-based methods* for reasoning
- ▶ *Tableaux methods work* by building models explicitly *via* formula expansion rules
- ▶ This limits *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 ($\mathcal{ALCN}\mathcal{R}$)

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

$$\left[\begin{array}{l} \text{if } x : (C_1 \sqcap C_2) \in S \\ \text{and } \{x : C_1, x : C_2\} \notin S \end{array} \right] \frac{S}{S \cup \{x : C_1, x : C_2\}}$$

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

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

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

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

Semantic Web formalisms—Tableaux style \mathcal{DL} reasoning (\mathcal{ALCN}) - ctd.

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

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

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

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

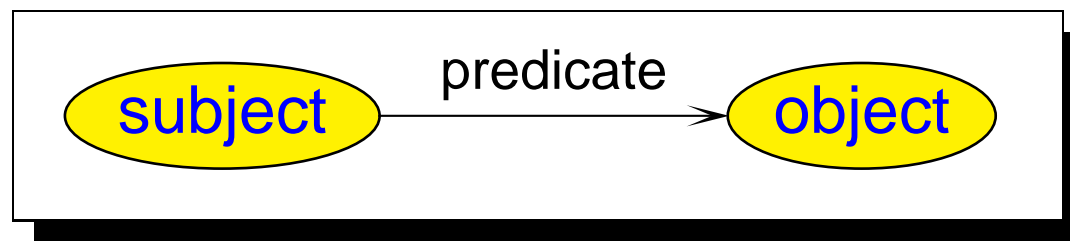
(\mathcal{DL}_{\leq}) MAX CARDINALITY:

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

Semantic Web formalisms—*RDF triples*

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

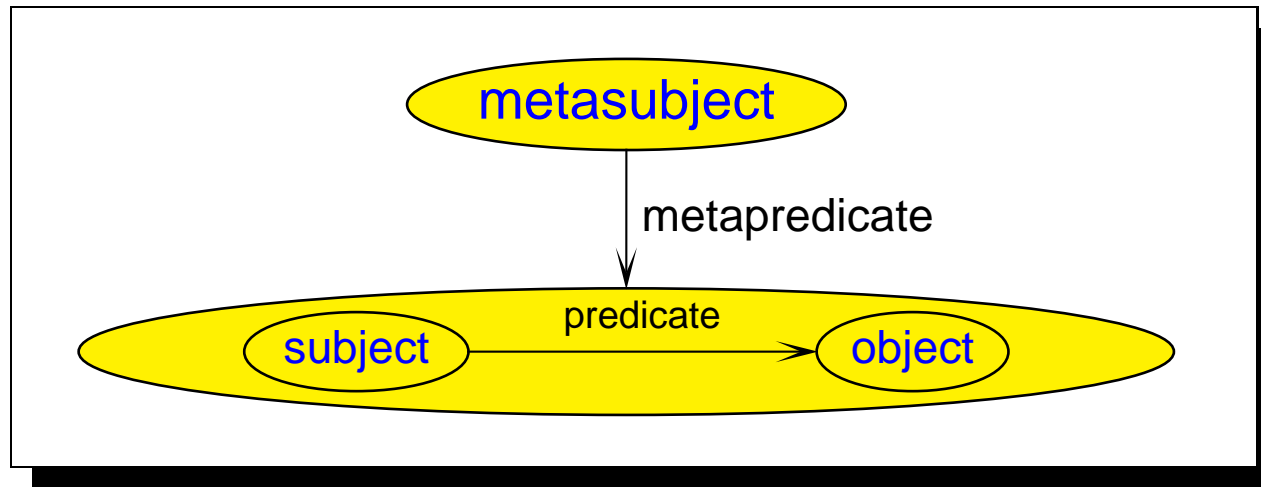
- ▶ Basic building block: “triple” labeled by “resources”—*i.e.*, **URI**'s.
- ▶ A **triple** consists of a resource (the **subject**), linked through a resource (the **predicate**) to another resource (the **object**).
- ▶ A triple states that the **subject** has a **property**, denoted by the **predicate**,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



- ▶ The information carried by a triple is called a “**statement**.”

Semantic Web formalisms—*RDF triples*

- ▶ 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**.

Semantic Web formalisms—*RDF* triples

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 *RDF* what *HTML* has done to text: it interconnects knowledge through the Internet*

Semantic Web formalisms—*RDF XML syntax*

`<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>`

Semantic Web formalisms—*RDF XML syntax*

```
<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>
```

Semantic Web formalisms—*RDF XML syntax*

```
<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>
```

Semantic Web formalisms—*RDF XML syntax*

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>
```

Semantic Web formalisms—*RDF XML syntax*

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>
```

Semantic Web formalisms—*RDF XML syntax*

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>
```

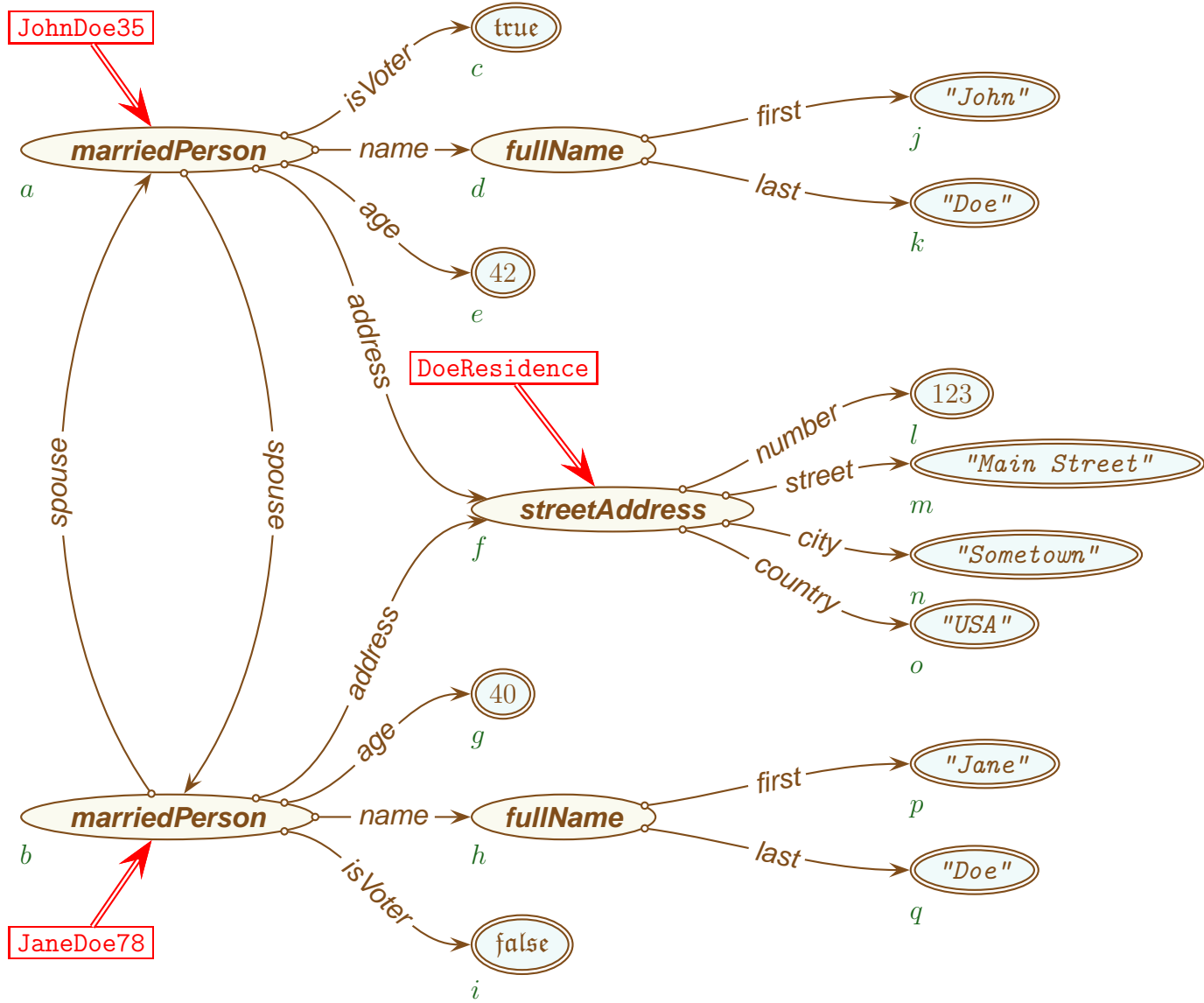
Semantic Web formalisms—*Objects are labelled graphs!*

```
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"
                               )
```



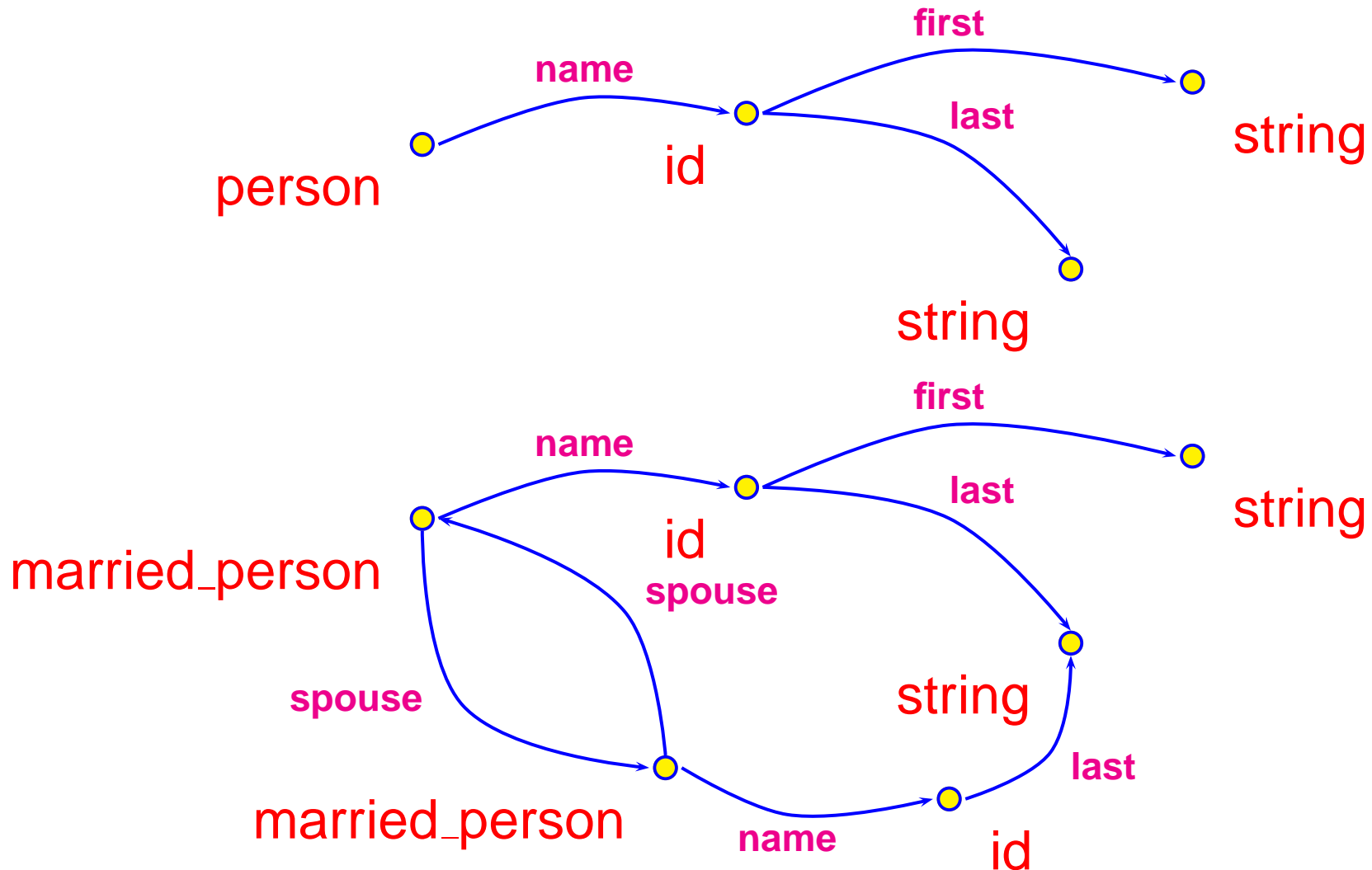
- ▶ Semantic Web formalisms
- ▶ Graphs as constraints
- ▶ *OSF* vs. *DL*
- ▶ *LIFE*—logically and functionally constrained sorted graphs
- ▶ Recapitulation

- ▶ **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.*, ...

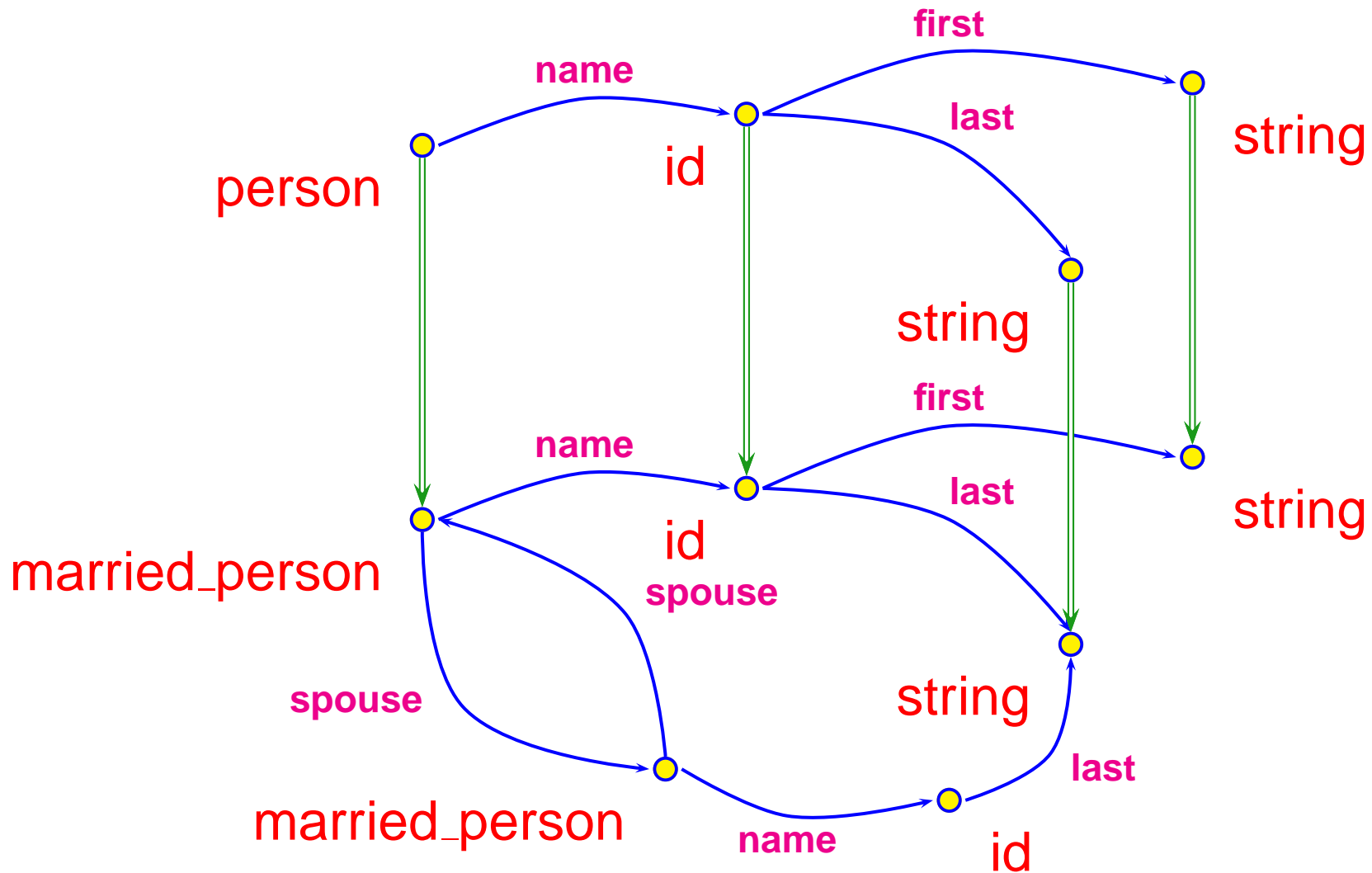
Viewing graphs as **constraints** stems from the work of:

- ▶ 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*



Graphs as constraints—*OSF* term syntax

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

An *OSF* **term** is an expression of the form:

$$X : s(l_1 \Rightarrow t_1, \dots, l_n \Rightarrow t_n)$$

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 $X : s$)
- ▶ $\{l_1, \dots, l_n\} \subseteq \mathcal{F}$ are features
- ▶ t_1, \dots, t_n are *OSF* terms

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

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

Lighter notation:

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

Graphs as constraints—*OSF* term semantics

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

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

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

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

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

Graphs as constraints—*OSF* clause syntax

An *OSF* constraint is one of:

- ▶ $X : s$
- ▶ $X.l \doteq X'$
- ▶ $X \doteq X'$

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 *OSF* clause is a **conjunction** of *OSF* constraints—*i.e.*, a **set of *OSF* constraints**

$$\phi_1 \ \& \ \dots \ \& \ \phi_n$$

Graphs as constraints—Semantics of OSF clauses

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

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

$$\mathfrak{A}, \alpha \models X \doteq Y \iff \alpha(X) = \alpha(Y)$$

$$\mathfrak{A}, \alpha \models X.l \doteq Y \iff \ell^{\mathfrak{A}}(\alpha(X)) = \alpha(Y)$$

$$\mathfrak{A}, \alpha \models \phi_1 \ \& \ \dots \ \& \ \phi_n \iff \mathfrak{A}, \alpha \models \phi_i \ \forall i = 1, \dots, n$$

Graphs as constraints—From OSF terms to OSF clauses

An OSF term $t = X : s(l_1 \Rightarrow t_1, \dots, l_n \Rightarrow t_n)$ is **dissolved** into an OSF clause $\phi(t)$ as follows:

$$\phi(t) \stackrel{\text{DEF}}{=} X : s \quad \& \quad X.l_1 \doteq X_1 \quad \& \quad \dots \quad \& \quad X.l_n \doteq X_n \\ \quad \quad \quad \& \quad \varphi(t_1) \quad \quad \quad \& \quad \dots \quad \quad \quad \& \quad \varphi(t_n)$$

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

Theorem:

$$\mathfrak{A}, \alpha \models \phi(t) \iff \llbracket t \rrbracket^{\mathfrak{A}, \alpha} \neq \emptyset$$

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

$$t = X : \text{person}(\text{name} \Rightarrow N : \top(\text{first} \Rightarrow F : \text{string}), \\ \text{name} \Rightarrow M : \text{id}(\text{last} \Rightarrow S : \text{string}), \\ \text{spouse} \Rightarrow P : \text{person}(\text{name} \Rightarrow I : \text{id}(\text{last} \Rightarrow S : \top), \\ \text{spouse} \Rightarrow X : \top))$$

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

Graphs as constraints—Basic *OSF* term normalization

(1) Sort Intersection

$$\phi \ \& \ X : s \ \& \ X : s'$$

$$\phi \ \& \ X : s \wedge s'$$

(2) Inconsistent Sort

$$\phi \ \& \ X : \perp$$

$$X : \perp$$

(3) Variable Elimination

$$\phi \ \& \ X \doteq X'$$

$$\phi[X'/X] \ \& \ X \doteq X'$$

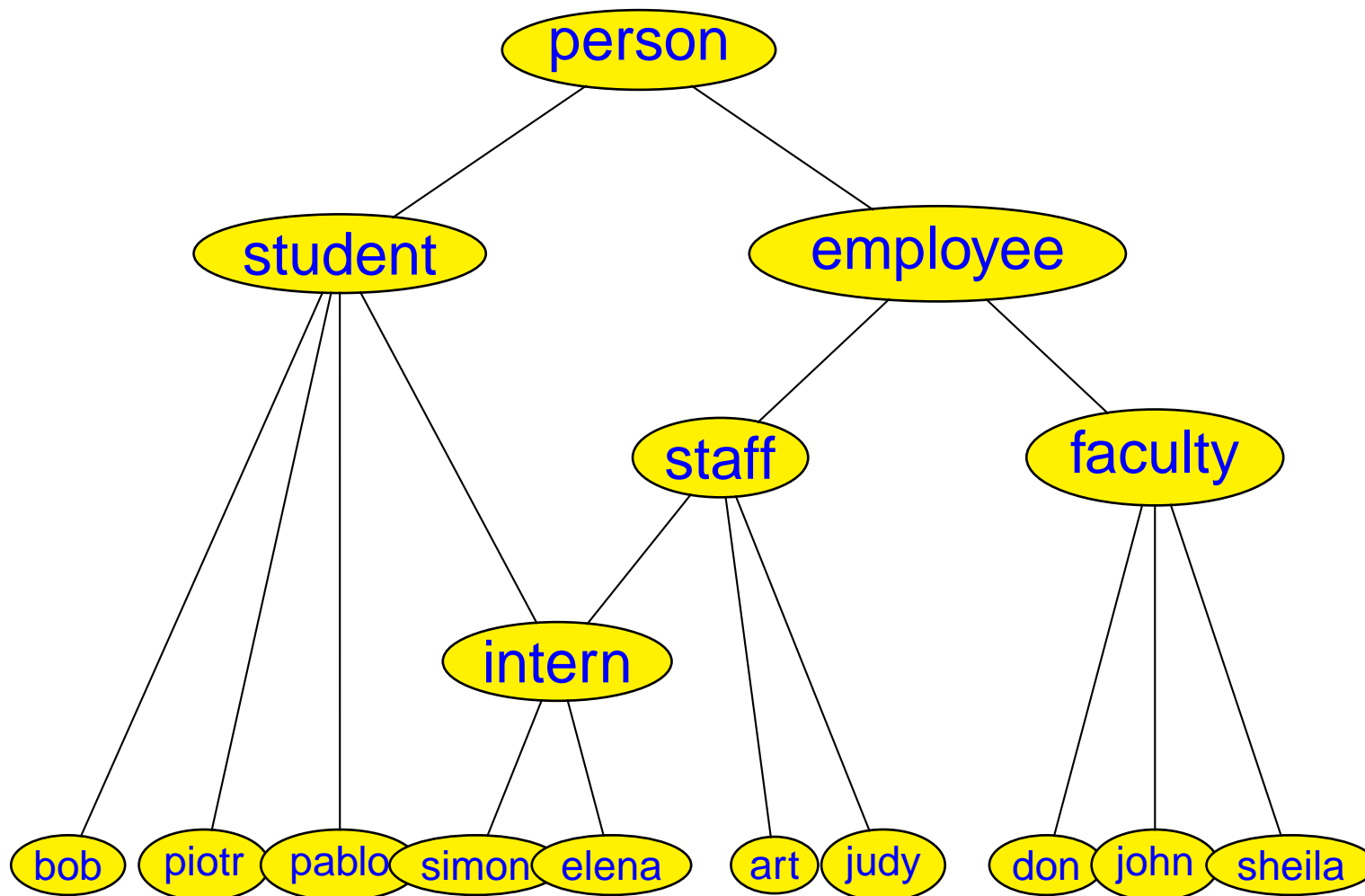
if $X \neq X'$
and $X \in \mathbf{Var}(\phi)$

(4) Feature Functionality

$$\phi \ \& \ X.l \doteq X' \ \& \ X.l \doteq X''$$

$$\phi \ \& \ X.l \doteq X' \ \& \ X' \doteq X''$$

Graphs as constraints—*OSF* unification = *OSF* constraint normalization



Graphs as constraints—*OSF* unification = *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—*OSF* unification = *OSF* constraint normalization

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

&

```
X = Y
```

&

```
E = S
```

Graphs as constraints—First-order terms as \mathcal{OSF} graph constraints

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

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

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

Basic 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.*, “ OSF theories”)

Graphs as constraints—*Extended OSF terms*

$\text{OsfTerm} ::= [\text{Variable:}] \text{Term}$

$\text{Term} ::= \text{ConjunctiveTerm}$

$| \text{DisjunctiveTerm}$

$| \text{NegativeTerm}$

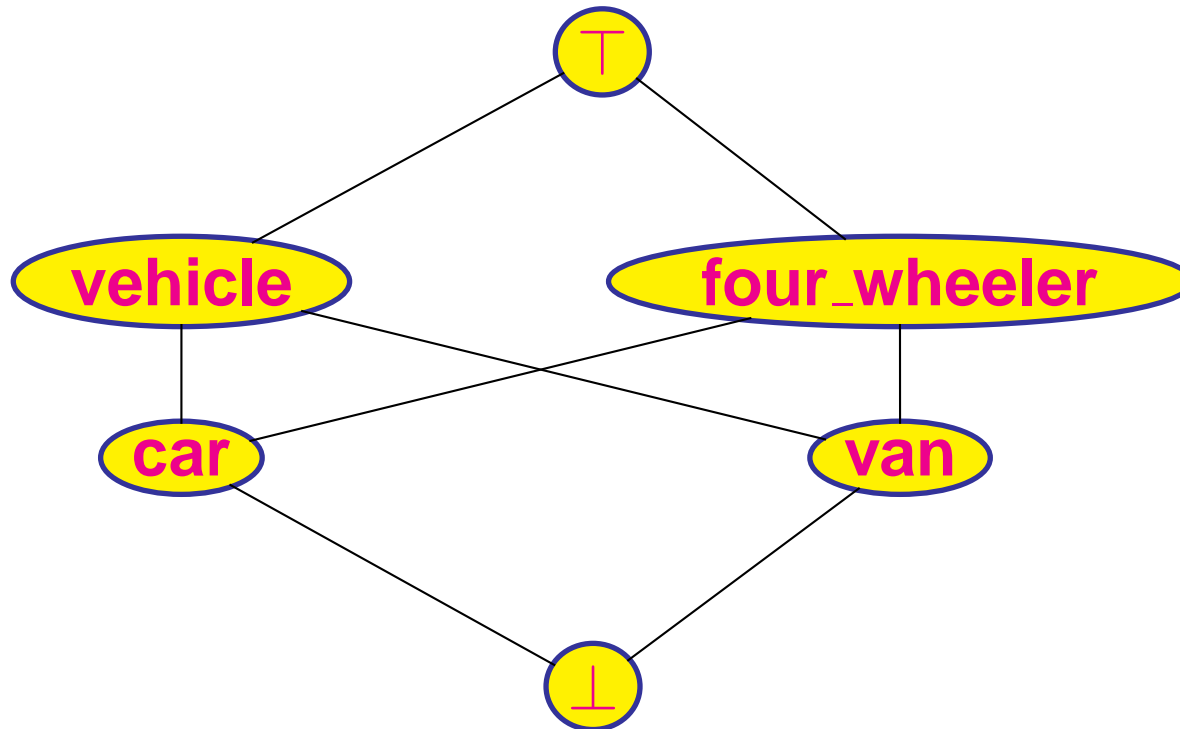
$\text{ConjunctiveTerm} ::= \text{Sort} [(\text{Attribute}^+)]$

$\text{Attribute} ::= \text{Feature} \Rightarrow \text{OsfTerm}$

$\text{DisjunctiveTerm} ::= \{ \text{OsfTerm} [; \text{OsfTerm}]^* \}$

$\text{NegativeTerm} ::= \neg \text{OsfTerm}$

Extended *OSF* constraints—*Non-lattice signatures, disjunction*



Non-unique GLBs are **disjunctive sorts**:

$$\mathbf{vehicle} \wedge \mathbf{four_wheeler} = \{\mathbf{car}; \mathbf{van}\}$$

Extended OSF constraints—Disjunctive OSF terms

Syntax of disjunctive OSF terms:

$$\{ t_1 ; \dots ; t_n \}$$

Semantics of disjunctive OSF terms:

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

Disjunctive OSF clauses:

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

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

Extended OSF constraints—Disjunctive OSF normalization

(5) Non-unique GLB

$$\phi \ \& \ X : s \ \& \ X : s'$$

$$\phi \ \& \ (X : s_1 \parallel \dots \parallel X : s_n)$$

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

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

(6) Distributivity

$$\phi \ \& \ (\phi' \parallel \phi'')$$

$$(\phi \ \& \ \phi') \parallel (\phi \ \& \ \phi'')$$

(7) Disjunction

$$\phi \parallel \phi'$$

$$\phi$$

Extended OSF constraints—Negation

Syntax of negative OSF terms: $\neg t$

Semantics of negative OSF terms: $\llbracket \neg t \rrbracket^{\mathcal{A}} \stackrel{\text{DEF}}{=} D^{\mathcal{A}} \setminus \llbracket t \rrbracket^{\mathcal{A}}$

Complemented sorts: $\llbracket \bar{s} \rrbracket^{\mathcal{A}} \stackrel{\text{DEF}}{=} D^{\mathcal{A}} \setminus \llbracket s \rrbracket^{\mathcal{A}}$

Sorted variable simplification:

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

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

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

Extended *OSF* constraints—Negative *OSF* terms

Dissolving negative *OSF* terms into *OSF* clauses eliminates negation:

$$\varphi(\neg(\neg t)) \stackrel{\text{DEF}}{=} \varphi(t)$$

$$\varphi(\neg\{t_1; \dots; t_n\}) \stackrel{\text{DEF}}{=} \varphi(\neg t_1) \ \& \ \dots \ \& \ \varphi(\neg t_n)$$

$$\varphi(\neg X : s(l_i \Rightarrow t_i)_{i=1}^n) \stackrel{\text{DEF}}{=} \varsigma(X : \bar{s})$$

$$\| \quad X.l_1 \doteq X_1 \ \& \ \varphi(\neg t_1)$$

$$\| \quad X.l_1 \doteq X'_1 \ \& \ X'_1 \neq X_1 \ \& \ \varphi(t_1)$$

...

$$\| \quad X.l_n \doteq X_n \ \& \ \varphi(\neg t_n)$$

$$\| \quad X.l_n \doteq X'_n \ \& \ X'_n \neq X_n \ \& \ \varphi(t_n)$$

(8) Variable Disequality

$$\phi \ \& \ X \neq X$$

$$\perp$$

(9) Sort Complement

$$\phi \ \& \ X : \bar{s}$$

$$\phi \ \& \ X : s'$$

if $s' \in \max_{\leq} \{t \in \mathcal{S} \mid s \not\leq t \text{ and } t \not\leq s\}$

Extended OSF constraints—Partial features

Partial features have restricted domains:

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

Declaring partial feature domains:

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

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

(10) Partial Feature

$$\phi \ \& \ X.l \doteq X'$$

$$\phi \ \& \ X.l \doteq X' \ \& \ X : s \ \& \ X' : s'$$

$$\text{if } s \in \mathbf{Dom}(\ell) \\ \text{and } \mathbf{Ran}_s(\ell) = s'$$

Extended OSF constraints—Partial features (example)

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

$$nil < list$$

$$cons < list$$

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

$$\mathbf{Dom}(hd) \stackrel{\text{DEF}}{=} \{cons\}$$

$$\mathbf{Dom}(tl) \stackrel{\text{DEF}}{=} \{cons\}$$

then:

$$list(tl \Rightarrow X) \rightsquigarrow cons(tl \Rightarrow X)$$

$$int(tl \Rightarrow X) \rightsquigarrow \perp$$

Extended OSF constraints—*Extensional sorts*

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**.

Extended OSF constraints—*Extensional sorts*

Extensional sorts are element constructors.

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

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

e.g.:

$$\mathit{Arity}(n) = \emptyset \quad \forall n \in \mathbb{N}$$

$$\mathit{Arity}(\text{nil}) = \emptyset$$

$$\mathit{Arity}(\text{cons}) = \{\text{hd}, \text{tl}\}$$

Extended OSF constraints—Extensional sorts

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

if $\mathit{Arity}(f) = n$ and $X_i = Y_i$ ($\forall i = 1, \dots, n$)

then $f(X_1, \dots, X_n) = f(Y_1, \dots, Y_n)$

(11) Weak Extensionality

$\phi \ \& \ X : s \ \& \ X' : s$

if $s \in \mathcal{E}$ and $\forall l \in \mathit{Arity}(s)$:

$\phi \ \& \ X : s \ \& \ X \doteq X'$

$\{X.f \doteq Y, X'.f \doteq Y\} \subseteq \phi$

Extended OSF constraints—*Extensional sorts*

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

let $s \in \mathcal{E}$ and $\mathbf{Arity}(s) = \{\ell\}$

then $X : s(\ell \Rightarrow X)$ & $X' : s(\ell \Rightarrow X')$

or $X : s(\ell \Rightarrow X')$ & $X' : s(\ell \Rightarrow X)$

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

Extended OSF constraints—Strong extensionality

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

(12) Extensional Occurrence

$$\Gamma \uplus \{s/V, \dots, \} \vdash \phi \ \& \ X : s$$

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

if $s \in \mathcal{E}$ and $X \notin V$

and $\forall f \in \mathit{Arity}(s)$:

$\{X.f \doteq X', X' : s'\} \subseteq \phi$
with $s' \in \mathcal{E}$

(13) Strong Extensionality

$$\Gamma \uplus \{s/\{X, X', \dots\} \vdash \phi$$

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

if $s \in \mathcal{E}$

Extended 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

$$\phi \ \& \ X = e : s \ \& \ X = e' : s'$$

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

$$\phi \ \& \ X = e \star e' : s \wedge s'$$

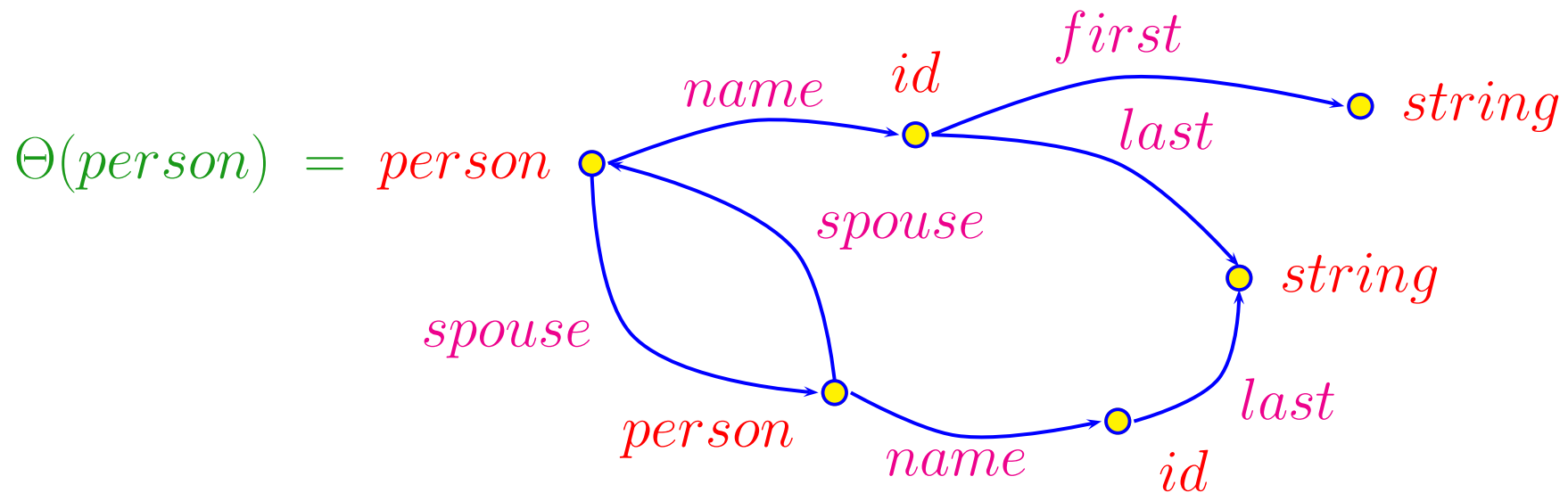
N.B.: This works for any commutative monoid—not just sets!

Extended OSF constraints— OSF theory unification

IDEA: Augment the sort ordering with constraints imposing:

- sorts of features
- coreference equations

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



Extended *OSF* constraints—*OSF* theory

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

An *OSF* theory is **order-consistent** iff it is **monotonic**:

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

OSF theory unification problem:

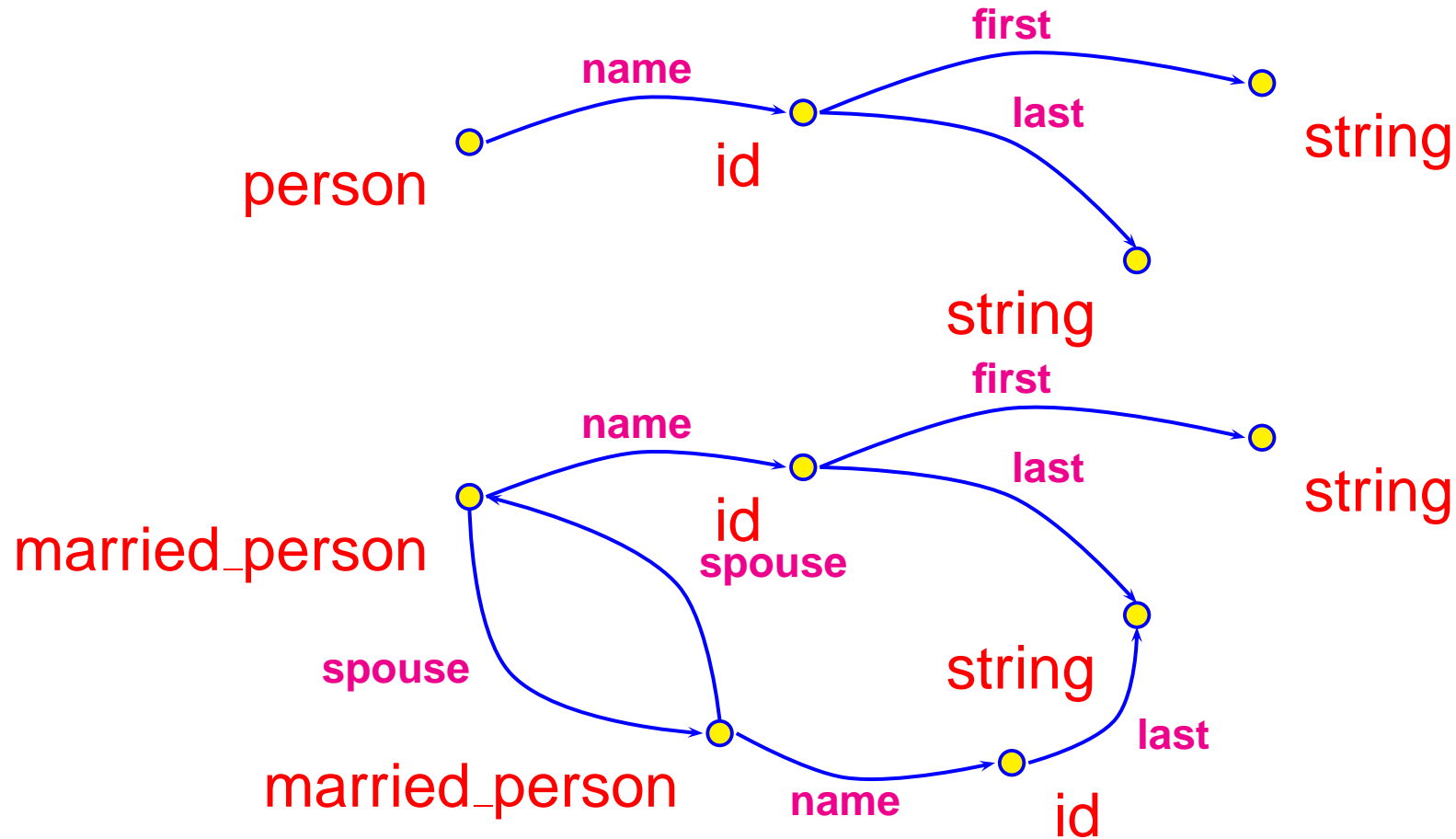
Given an order-consistent *OSF* theory Θ , normalize any term of sort s taking into account the *OSF* constraints $\Theta(s)$.

Theorem *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 *OSF* constraint inheritance from the theory lazily;
- ▶ there is an efficient algorithm which is complete for a large class of *OSF* theories;
- ▶ only one rule completes it (and may cause divergence).

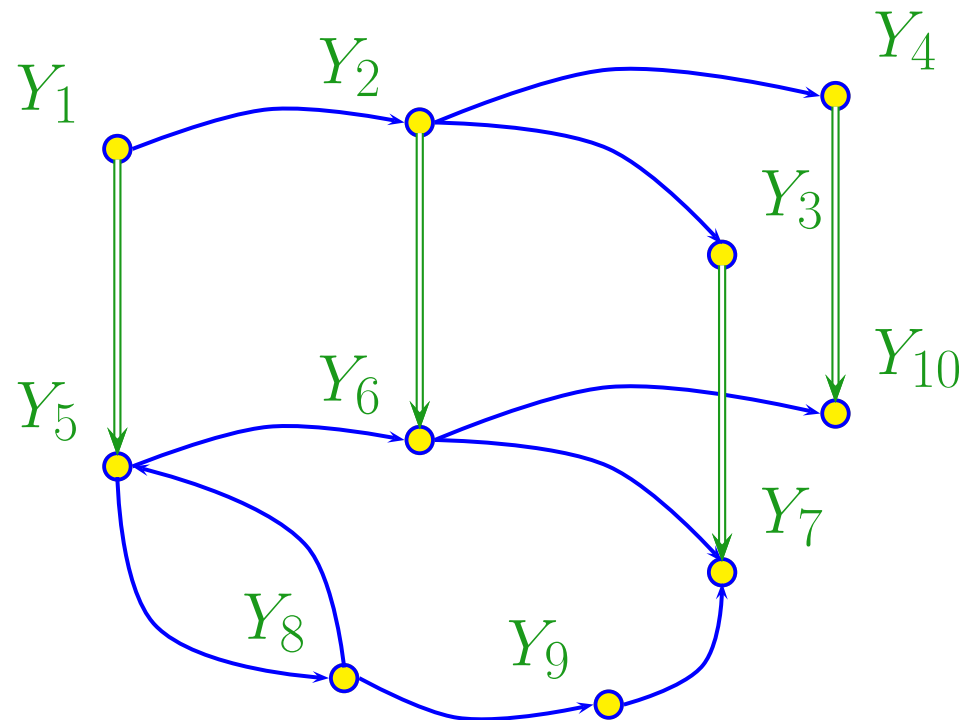
Extended *OSF* constraints—*OSF* theory unification (example)



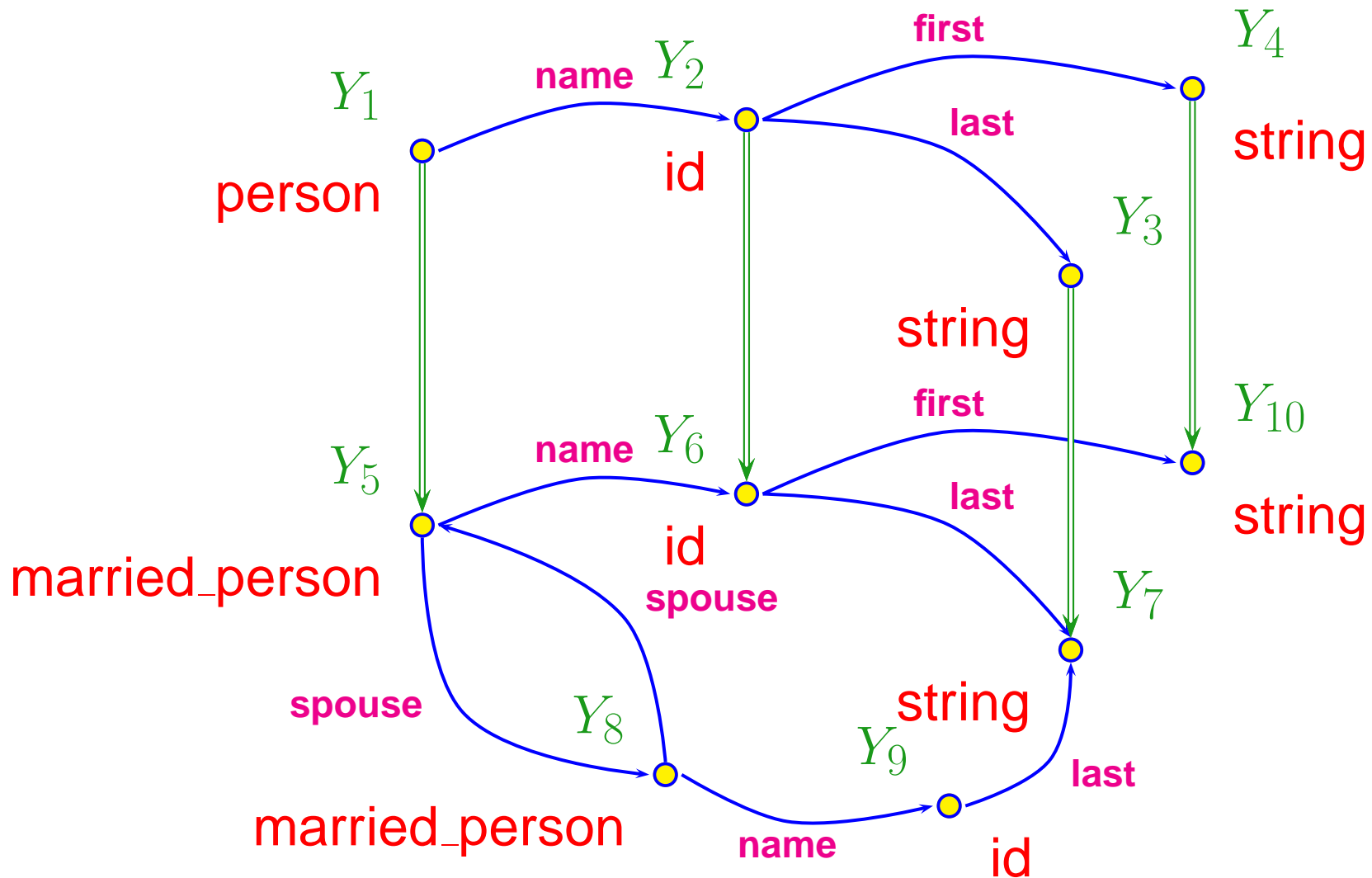
Extended OSF constraints— OSF theory unification

The fact that an OSF theory is order-consistent yields an **endomorphoric mapping** of **theory variables**.

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



Extended *OSF* constraints—*OSF* theory unification



Normalizing:

$P : person(name \Rightarrow \top(last \Rightarrow \text{“Smith”}))$
& $P : married_person(spouse \Rightarrow Q)$
& $Q : person(name \Rightarrow id(last \Rightarrow S))$

yields, among other things:

$P : married_person$
& $Q : married_person$
& $S : \text{“Smith”} \dots$

(0) Frame Allocation

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

(1) Sort Intersection

$$\Gamma \cup \{\{X \setminus Y_{s'}\} \cup F\} \vdash X : s \ \& \ X : s' \ \& \ \phi$$

$$\Gamma \cup \{\{X \setminus Y_{s \wedge s'}\} \cup F\} \vdash X : s \wedge s' \ \& \ \phi$$

(2) Inconsistent Sort

$$\Gamma \cup \{\{X \setminus Y_{\perp}\} \cup F\} \vdash \phi$$

$$\emptyset \vdash \perp$$

(3) Variable Elimination

$$\frac{\Gamma \quad \vdash X \doteq X' \ \& \ \phi}{\Gamma[X'/X] \vdash X \doteq X' \ \& \ \phi[X'/X]} \quad \begin{array}{l} \text{if } X \neq X' \\ \text{and } X \in \mathbf{Var}(\Gamma) \cup \mathbf{Var}(\phi) \end{array}$$

(4) Feature Functionality

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

Extended OSF constraints— OSF theory unification (non-empty theory)

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

$$\Gamma \cup \{X \setminus Y\} \cup F \quad \vdash \phi \ \& \ X.l \doteq X'$$

$$\Gamma \cup \{X \setminus Y, X' \setminus Y'\} \cup F \quad \vdash \phi \ \& \ X.l \doteq X' \ \& \ X' : \mathbf{Sort}(Y')$$

(6) Frame Merging

$$\Gamma \cup \{X \setminus Y_s\} \cup F, \{X \setminus Y_{s'}\} \cup F' \quad \vdash \phi$$

$$\Gamma \cup \{X \setminus Y_{s \wedge s'}\} \cup F \cup F' \quad \vdash \phi$$

(7) Frame Reduction

$$\frac{\Gamma \cup \{X \setminus Y, X \setminus Y'\} \cup F \vdash \phi}{\Gamma \cup \{X \setminus Y\} \cup F \vdash \phi} \quad \text{if } Y \leq Y'$$

(8) Theory Coreference

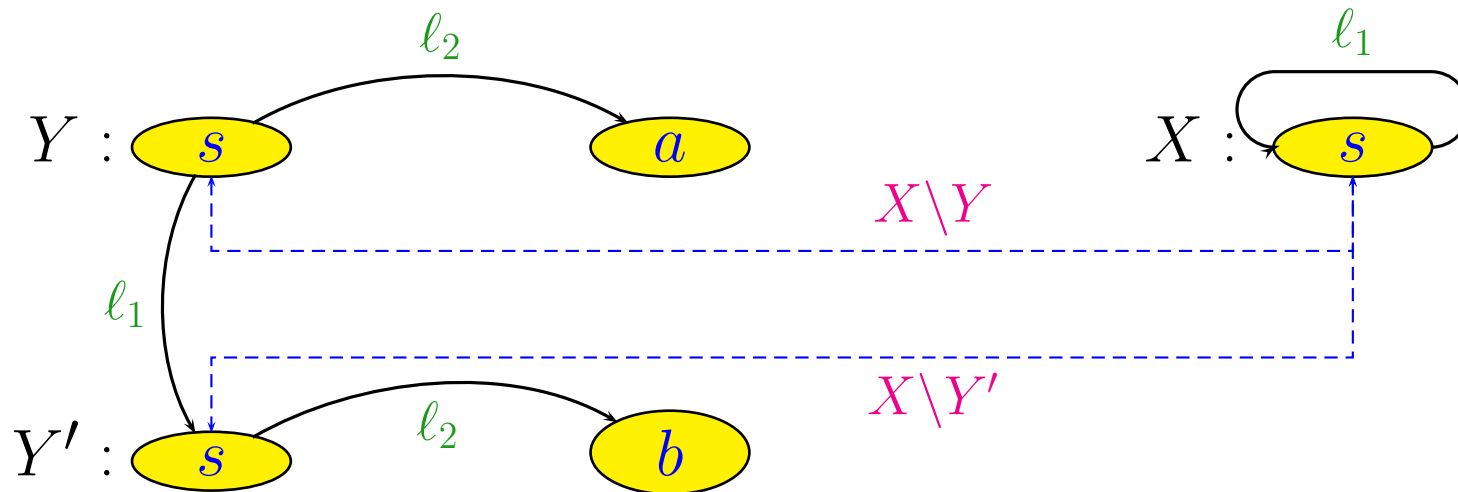
$$\frac{\Gamma \cup \{X \setminus Y, X' \setminus Y\} \cup F \vdash \phi}{\Gamma \cup \{X \setminus Y\} \cup F \vdash \phi \ \& \ X \doteq X'}$$

(9) Theory Feature Completion

$\Gamma \vdash \phi$

$\Gamma \vdash X.l \doteq Z \ \& \ \phi$

if $X \setminus Y \in F$ for some $F \in \Gamma$
 and $X \setminus Y' \in F'$ for some $F' \in \Gamma$
 and both $l(Y), l(Y')$ 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

OSF Graph Constraint Formalism—Outline

- ▶ Semantic Web formalisms
- ▶ Graphs as constraints
- ▶ *OSF* vs. *DL*
- ▶ *LIFE*—logically and functionally constrained sorted graphs
- ▶ Recapitulation

Understanding *OWL speak*—*OSF* vs. *DL*

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

ergo:

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

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

ergo:

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

OSF Graph Constraint Formalism—Outline

- ▶ Semantic Web formalisms
- ▶ Graphs as constraints
- ▶ *OSF* vs. *DL*
- ▶ *LIFE*—logically and functionally constrained sorted graphs
- ▶ Recapitulation

LIFE—logically and functionally constrained sorted graphs

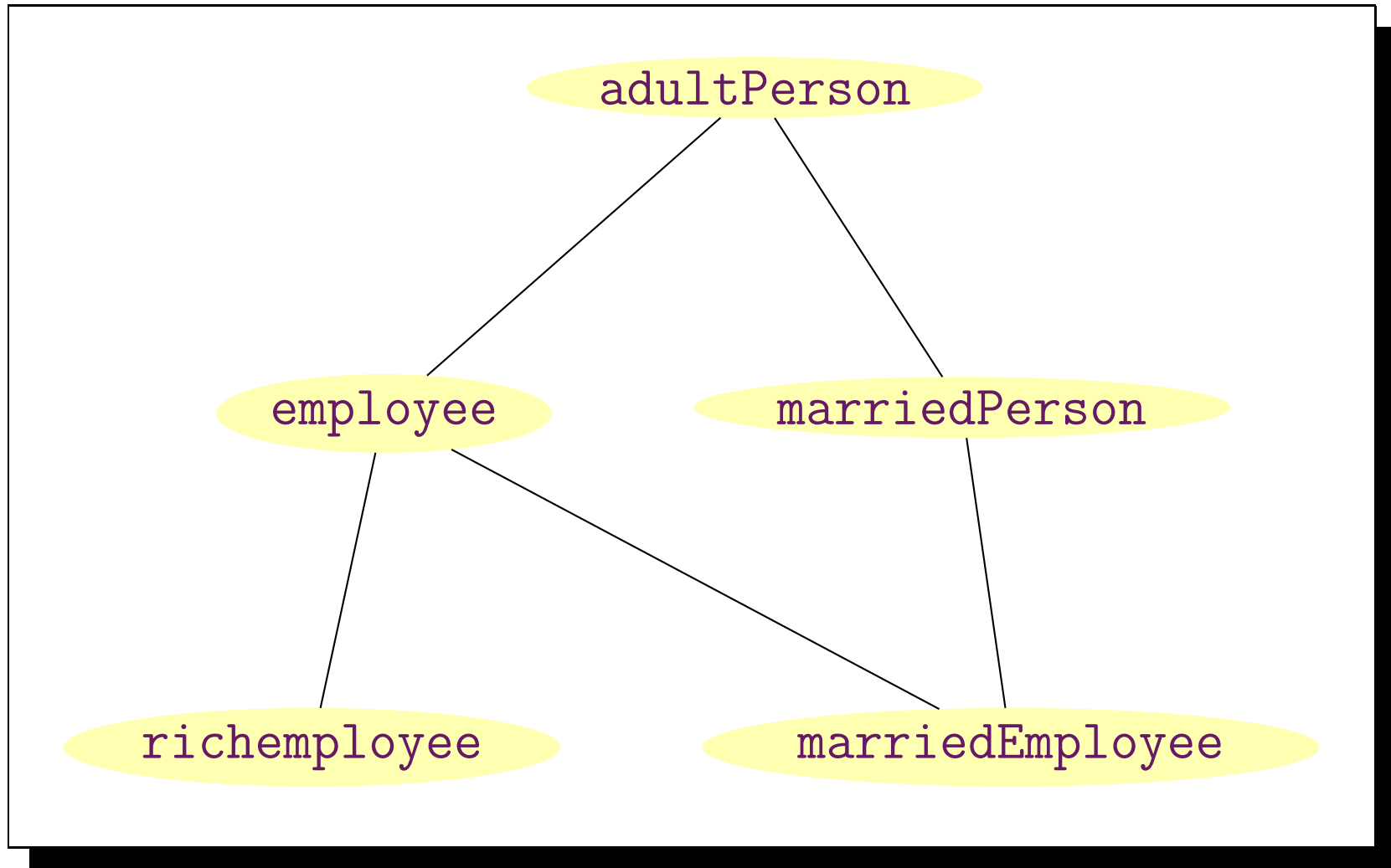
LIFE—*L*ogic, *I*nheritance, *F*unctions, and *E*quations

CLP(χ)—*C*onstraint, *L*ogic, *P*rogramming, parameterized over is a constraint system χ

LIFE is a *CLP* system over *OSF* constraints and functions over them (rewrite rules); namely:

$$LIFE = CLP(OSF + FP)$$

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 *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 *LIFE* sort hierarchy

```
:: P : adultPerson ( id ⇒ name
                    , dob ⇒ date
                    , age ⇒ A : int
                    , ssn ⇒ string )
```

```
| A = ageInYears(P), A ≥ 18.
```

```
:: employee        ( position ⇒ T : title
                    , institution ⇒ string
                    , supervisor ⇒ E : employee
                    , salary ⇒ S : int )
```

```
| higherRank(E.position, T) , E.salary ≥ S.
```

A relationally and functionally constrained *LIFE* sort hierarchy

```
:: M : marriedPerson ( spouse  $\Rightarrow$  P : marriedPerson )  
| P.spouse = M.
```

```
:: R : richEmployee ( institution  $\Rightarrow$  I  
                    , salary  $\Rightarrow$  S )  
| stockValue(I) = V , hasShares(R, I, N) , S + N * V  $\geq$  200000.
```

Proof “memoizing”

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: **model equivalence \neq proof equivalence!**

- ▶ *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, *OSF constraint proof rules ensure that:*

no type constraint is ever proved twice

Proof “memoizing”

OSF type constraints are incrementally “*memoized*” as they are verified:

sorts act as (instantaneous!) proof caches!

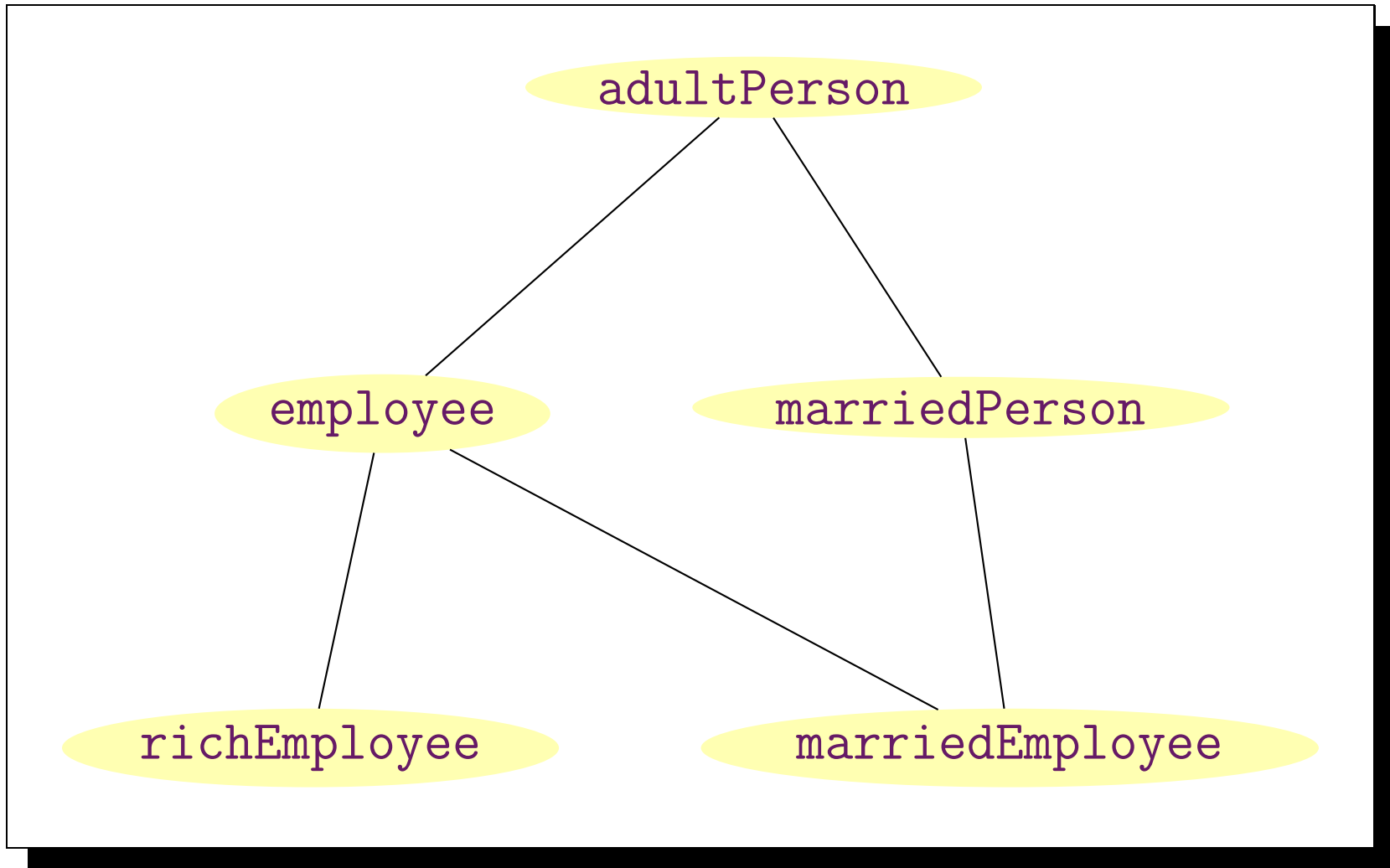
whereas in logic having proven $s(X)$ is not “*remembered*” in any way (e.g., Prolog)

Example: consider the *OSF* constraint conjunction:

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

Notation: $\text{type}\#(\text{condition})$ means “*constraint condition attached to sort type*”

Proof “memoizing”—Example hierarchy reminded



Proof “memoizing”

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

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 *OSF* constraint-solving enables:

using rules over ontologies

as well as, *conversely*,

enhancing ontologies with rules

Indeed, with *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
- ▶ *OSF* vs. *DL*
- ▶ *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!

Recapitulation—*what you must remember from this talk... (ctd)*

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

Recapitulation—*what you must remember from this talk... (ctd)*

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***
- ...

Recapitulation—*what you must remember from this talk... (ctd)*

- ▶ **Linked data** represents all information as interconnected sorted labelled *RDF* graphs—it has become a universal *de facto* knowledge model standard
- ▶ **Differences between *DL* and *OSF* can come handy:**
 - *DL* is **expansive**—therefore, **expensive**—and can only describe finitely computable sets; whereas,
 - *OSF* is **contractive**—therefore, **efficient**—and can also describe recursively-enumerable sets
- ▶ ***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>