TODAY'S SEMANTIC WEB

WHAT IT AMOUNTS TO; WHY IT IS ILL-CONCEIVED; HOW IT COULD BE FIXED

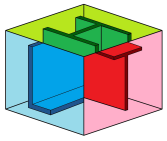
ICWR 2018 ***Keynote***

Tehran, Iran

April 25, 2018

Hassan Aït-Kaci

HAK Language Technologies



- In this presentation, **I will**:
  - **review standard Semantic Web formalisms**
  - **propose a constraint-based formalism to amend them**

- It is meant for **a technically mature audience**—familiar with elementary Logic Programming (Prolog), and common Web technology and terminology ($\mathcal{RDF}$, $\mathcal{XML}$, ...)

- Too **technical/mathematical?**—**not to worry**: **focus on the general ideas in my comments**

- **Technical contents** only **serve as examples to** illustrate **the points made in my comments**

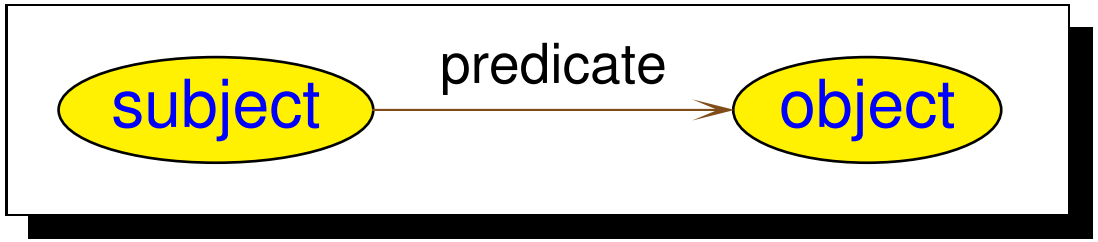
- Please **ask questions**; feel free to **propose discussions**

- Semantic Web formalisms
- Graphs as constraints
- $\mathcal{OSF}$ *vs.* $\mathcal{DL}$
- $\mathcal{LIFE}$: $\mathcal{L}$ogic $\mathcal{I}$nheritance $\mathcal{F}$unctions $\mathcal{E}$quations
- Recapitulation

- Semantic Web formalisms
- Graphs as constraints
- $\mathcal{OSF}$ *vs.* $\mathcal{DL}$
- $\mathcal{LIFE}$: $\mathcal{L}$ogic $\mathcal{I}$nheritance $\mathcal{F}$unctions $\mathcal{E}$quations
- Recapitulation

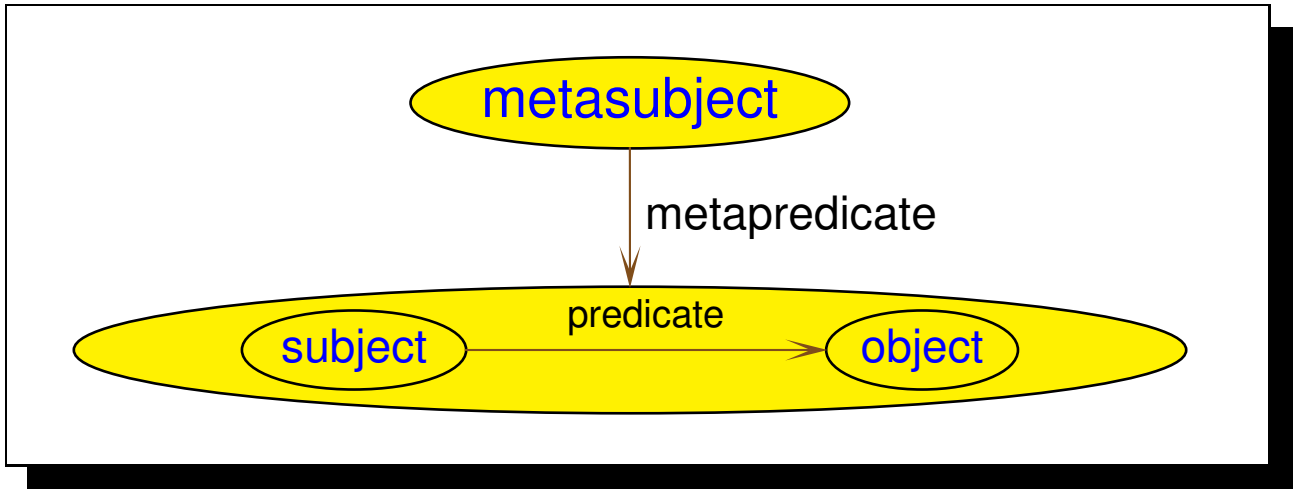
The Resource Decription Framework ($\mathcal{RDF}$) is a standard notation for describing connected data and metadata using (edge- and node-) labeled graphs.

- Basic building block: "triple" labeled by "resources"—*i.e.*, data objects or URIs and connections between resources.
- 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."

- $\mathcal{RDF}$ statements can be reified and be denoted as resources—hence, $\mathcal{RDF}$'s metalinguistic nature:



- $\mathcal{RDF}$ uses the eXtensible Markup Language $\mathcal{XML}$ for its serialized syntax.
- $\mathcal{RDF}$ enables the definition of **vocabularies** which can be shared over the Web thanks to standard $\mathcal{XML}$ namespaces (*e.g.*, Dublin Core).
- $\mathcal{RDF}$ **Schema** (RDFS) is a meta-description of $\mathcal{RDF}$ in $\mathcal{RDF}$ specified as a meta-vocabulary for $\mathcal{RDF}$; other sharable knowledge data models are expressible (*e.g.*, **S**imple **K**nowledge **O**rganization **S**ystem SKOS).

$\mathcal{RDF}$ triples may be expressed using several syntaxes:

- a (normative) $\mathcal{RDF}$ $\mathcal{XML}$ syntax
- Notation 3 syntax (Tim Berners-Lee, Dan Conolly)
- Turtle syntax—TRTL: Terse $\mathcal{RDF}$ Triple Language (David Beckett, Tim Berners-Lee)
- JSON—**J**ava**S**cript **O**bject **N**otation
- any other syntax you fancy as long as you can parse it into the normative $\mathcal{RDF}$ $\mathcal{XML}$ syntax . . .

JSON object

key/value map

term syntax

```
{ "menu" :
  { "id" : "file"
  , "value" : "File"
  , "popup" : { "menuitem":
                { "value" : "New"
                , "onclick" : "CreateNewDoc()"
                }
              , "menuitem":
                { "value" : "Open"
                , "onclick" : "OpenDoc()"
                }
              , "menuitem":
                { "value" : "Close"
                , "onclick" : "CloseDoc()"
                }
              }
  }
}
```

The same JSON object term expressed using $\mathcal{XML}$ syntax:

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

```
<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 $\mathcal{RDF}$ 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 $\mathcal{RDF}$ 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 $\mathcal{XML}$ notation for $\mathcal{RDF}$ 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>
```

- $\mathcal{OWL}$ is the W3C official standard formalism to use for the Semantic Web's knowledge representation and ontological reasoning

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

  The whole World-Wide Web is abuzz with $\mathcal{OWL}$-*this* and $\mathcal{OWL}$-*that*, . . . (*kn*$\underline{\mathcal{OWL}}$ *edge representation?*)

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

  $\mathcal{SHIN}$, $\mathcal{CIQ}$, $\mathcal{SHIQ}$, $\mathcal{SHOQ}(D)$, $\mathcal{SHOIN}$, $\mathcal{SHOIQ}$, $\mathcal{SRIQ}$, $\mathcal{SROIQ}$, . . . , are *not* alien species' tongues but dialects devised for $\mathcal{OWL}$ (W3C's Web Ontology 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:

- $\mathcal{OWL}$ species: $\mathcal{OWL}$ Lite, $\mathcal{OWL}$ DL, $\mathcal{OWL}$ Full
- which are varieties of Description Logics ($\mathcal{DL}$, $\mathcal{DLR}$, . . .)
- themselves categories of Attributive Logics ($\mathcal{ALC}$, $\mathcal{ALCN}$, $\mathcal{ALCNR}$, . . .)
- which gave rise to a proliferation of SW languages ($\mathcal{SHIN}$, $\mathcal{CIQ}$, $\mathcal{SHIQ}$, $\mathcal{SHOQ}(D)$, $\mathcal{SHOIN}$, $\mathcal{SHOIQ}$, $\mathcal{SRIQ}$, $\mathcal{SROIQ}$, . . .)

Naming conventions 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 using **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 — we verified!

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

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

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

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

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

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

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

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

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

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

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

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

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

- Semantic Web formalisms
- Graphs as constraints
- $\mathcal{OSF}$ vs. $\mathcal{DL}$
- $\mathcal{LIFE}$: $\mathcal{L}$ogic $\mathcal{I}$nheritance $\mathcal{F}$unctions $\mathcal{E}$quations
- Recapitulation

- **Proposal:** a formalism for representing structured objects that is: intuitive (objects as labeled graphs), expressive ("real-life" data models), formal (rigorous semantics), operational (executable), & efficient (constraint-solving)

- **Why?** *viz.*, ubiquitous use of labeled graphs to structure information naturally as in:
  - object-orientation, knowledge representation,
  - databases, semi-structured data,
  - natural language processing, graphical interfaces,
  - concurrency and communication,
  - $\mathcal{XML}$, $\mathcal{RDF}$, the "Semantic Web," *etc.*, ...

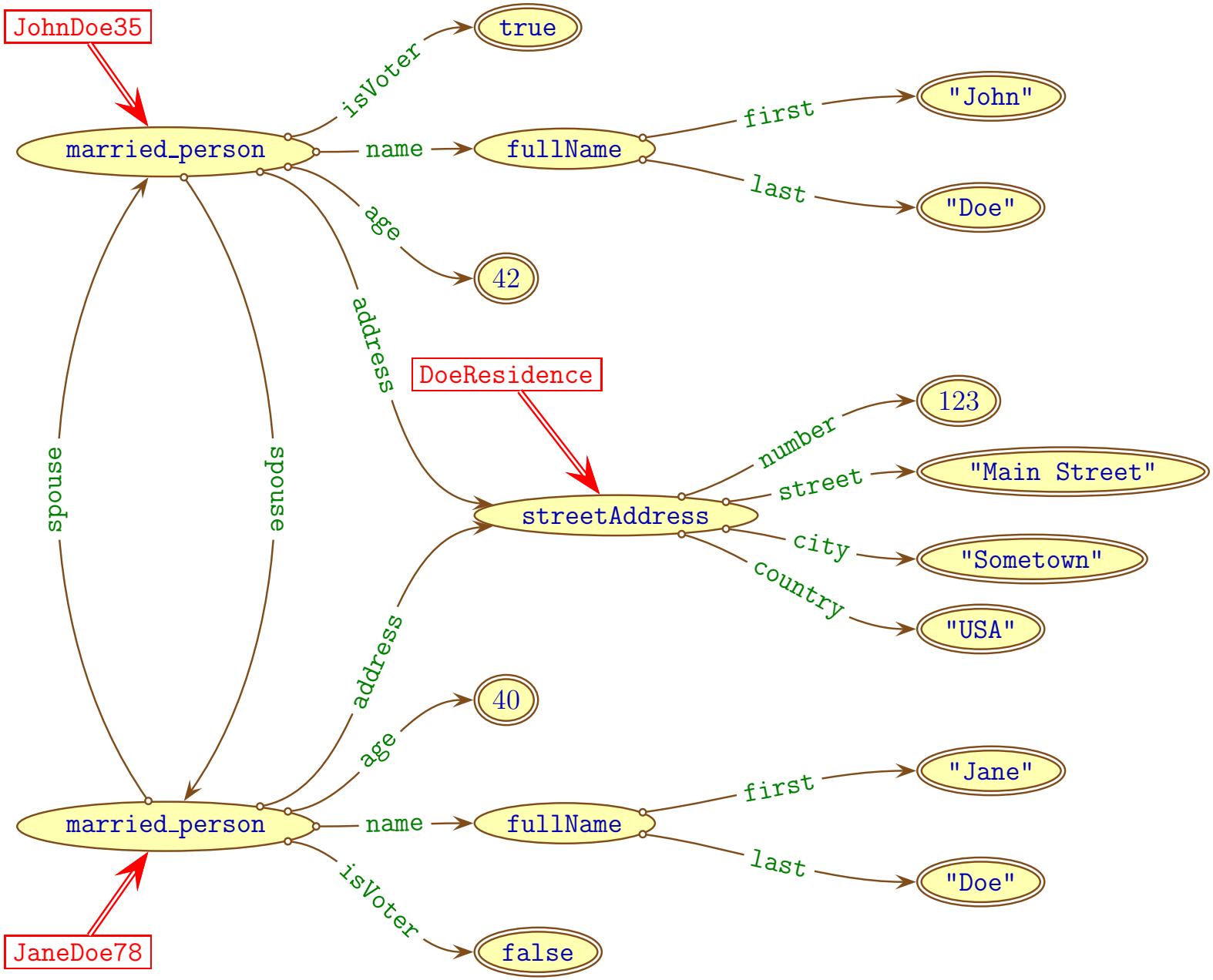
Elementary observation—*Web objects are key/value structures*

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

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

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

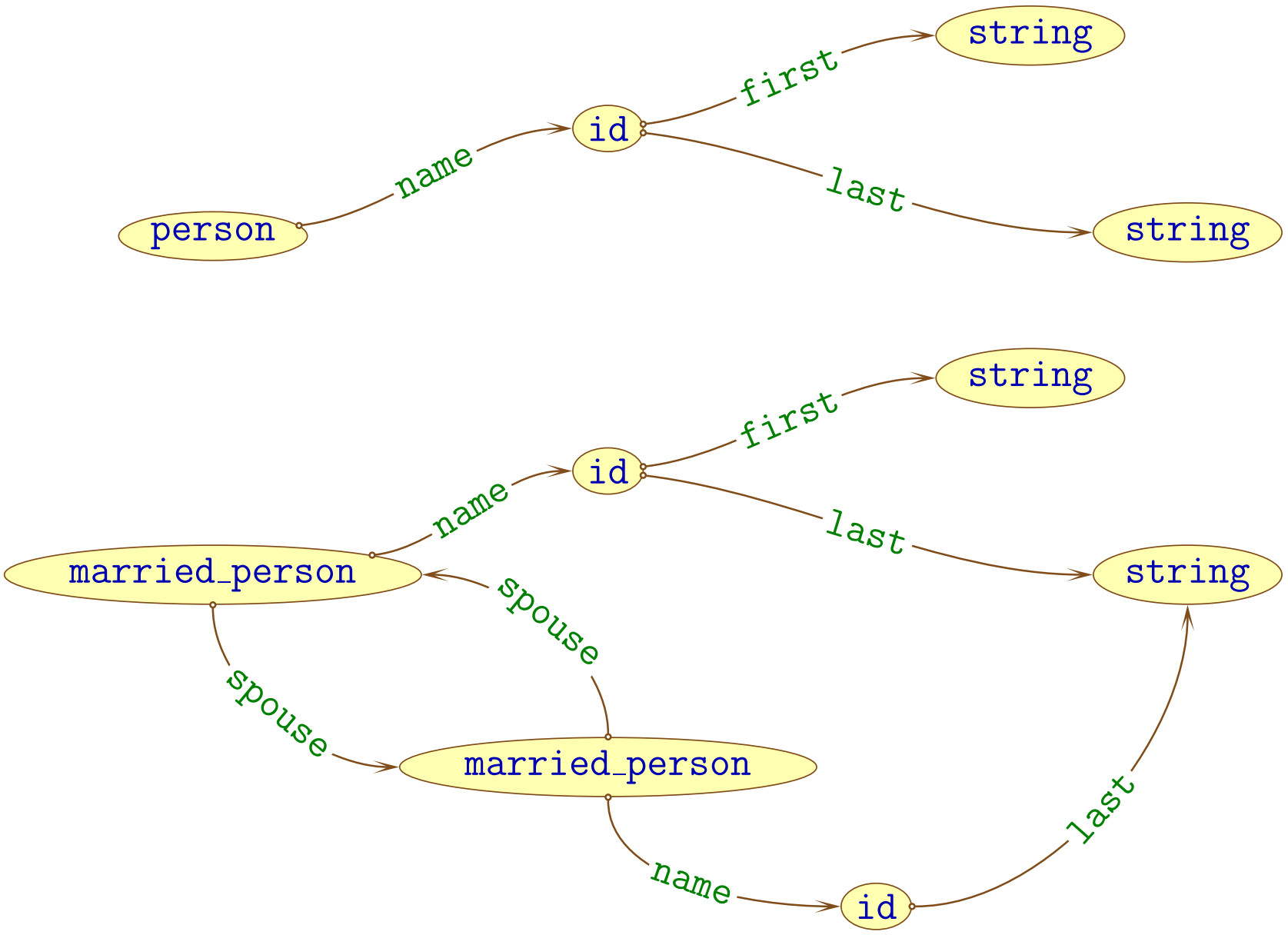
Elementary deduction—*Web objects are labeled graphs!*


JohnDoe35
married_person
isVoter
true
name
fullName
first
"John"
last
"Doe"
age
42
address
DoeResidence
streetAddress
number
123
street
"Main Street"
city
"Sometown"
country
"USA"
spouse
spouse
address
age
40
married_person
name
fullName
first
"Jane"
last
"Doe"
isVoter
false
JaneDoe78

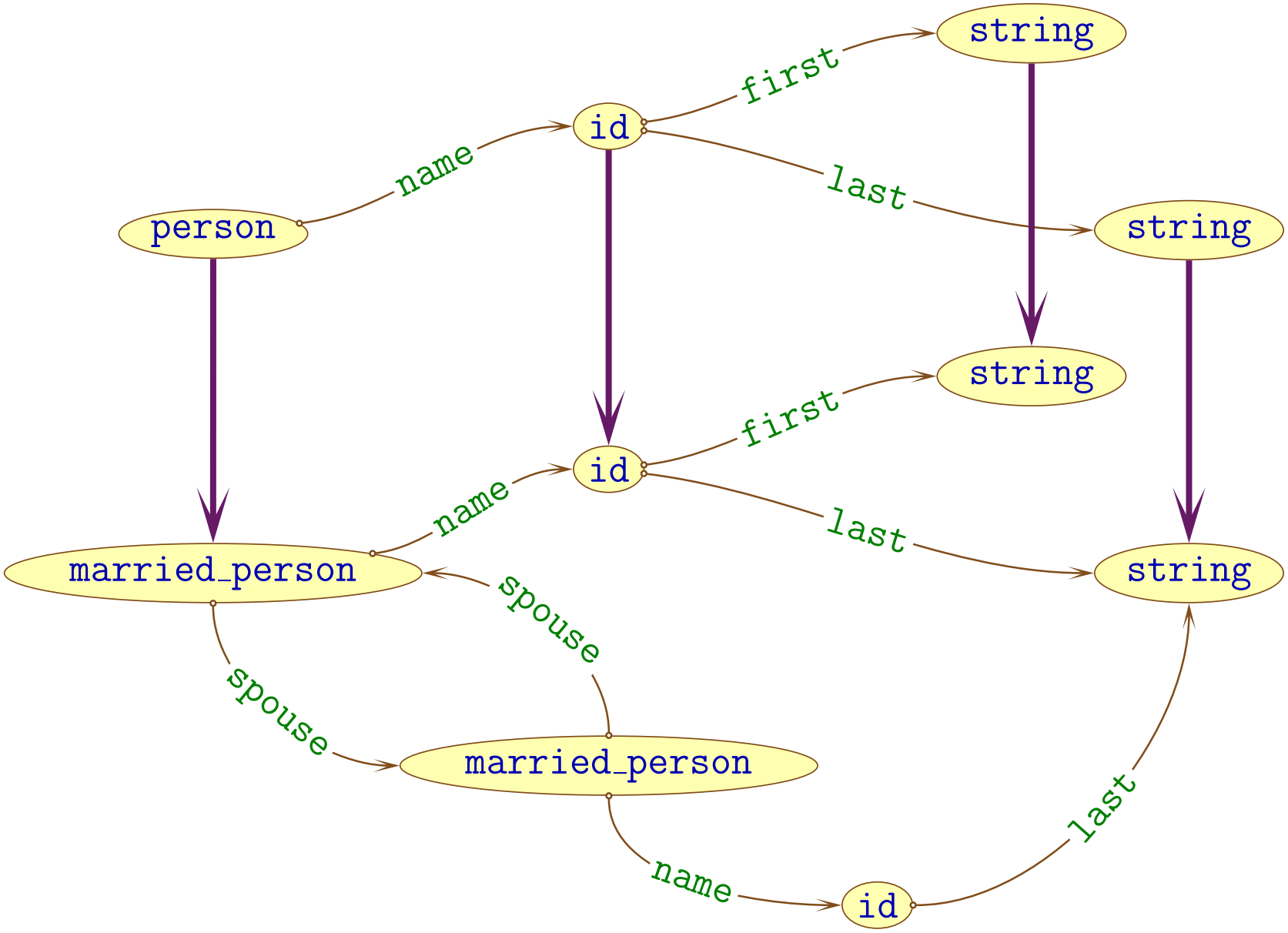
Viewing labeled graphs as **constraints** stems from the work of:

- 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* and $\mathcal{S}$ a set of *sorts*.

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

$$\mathtt{X} : \mathtt{s}(\mathtt{f}_1 \Rightarrow t_1, \ldots, \mathtt{f}_n \Rightarrow t_n)$$

where:

- $\mathtt{X} \in \mathcal{V}$ is the root variable
- $\mathtt{s} \in \mathcal{S}$ is the root sort
- $\{\mathtt{f}_1, \ldots, \mathtt{f}_n\} \subseteq \mathcal{F}$ are features
- $t_1, \ldots, t_n$ are $\mathcal{OSF}$ terms
- $n \geq 0$ — if $n = 0$, we simply write $\mathtt{X} : \mathtt{s}$

```
X : person(name ⇒ N : ⊤(first ⇒ F : string)
          ,name ⇒ M : id(last ⇒ S : string)
          ,spouse ⇒ P : person(name ⇒ I : id(last ⇒ S : ⊤)
                              ,spouse ⇒ X : ⊤ ))
```

Lighter notation for the same term by **erasing single tags**:

```
X : person(name ⇒ ⊤(first ⇒ string)
          ,name ⇒ id(last ⇒ S : string)
          ,spouse ⇒ person(name ⇒ id(last ⇒ S : ⊤)
                          ,spouse ⇒ X : ⊤ ))
```

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

- $\mathtt{X} : \mathtt{s}$
- $\mathtt{X.f} \doteq \mathtt{X}'$
- $\mathtt{X} \doteq \mathtt{X}'$

where $\mathtt{X}$ ($\mathtt{X}'$) is a variable (*i.e.*, a node), $\mathtt{s}$ is a sort (*i.e.*, a node's type), and $\mathtt{f}$ is a feature (*i.e.*, an arc).

An $\mathcal{OSF}$ **constraint clause** is a conjunctive set of atomic $\mathcal{OSF}$ constraints

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

An $\mathcal{OSF}$ term:

$$t = \mathtt{X} : \mathtt{s}(\mathtt{f}_1 \Rightarrow t_1, \ldots, \mathtt{f}_n \Rightarrow t_n)$$

is dissolved into an $\mathcal{OSF}$ clause $\varphi(t)$ as follows:

$$\begin{array}{lcllllllll} \varphi(t) & \stackrel{\text{DEF}}{=\!=} & \mathtt{X} : \mathtt{s} & \& & \mathtt{X}.\mathtt{f}_1 \doteq \mathtt{X}_1 & \& & \ldots & \& & \mathtt{X}.\mathtt{f}_n \doteq \mathtt{X}_n \\ & & & \& & \varphi(t_1) & \& & \ldots & \& & \varphi(t_n) \end{array}$$

where $\mathtt{X}_1, \ldots, \mathtt{X}_n$ are the root variables of $t_1, \ldots, t_n$

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

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

Sort Intersection

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

Variable Elimination

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

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

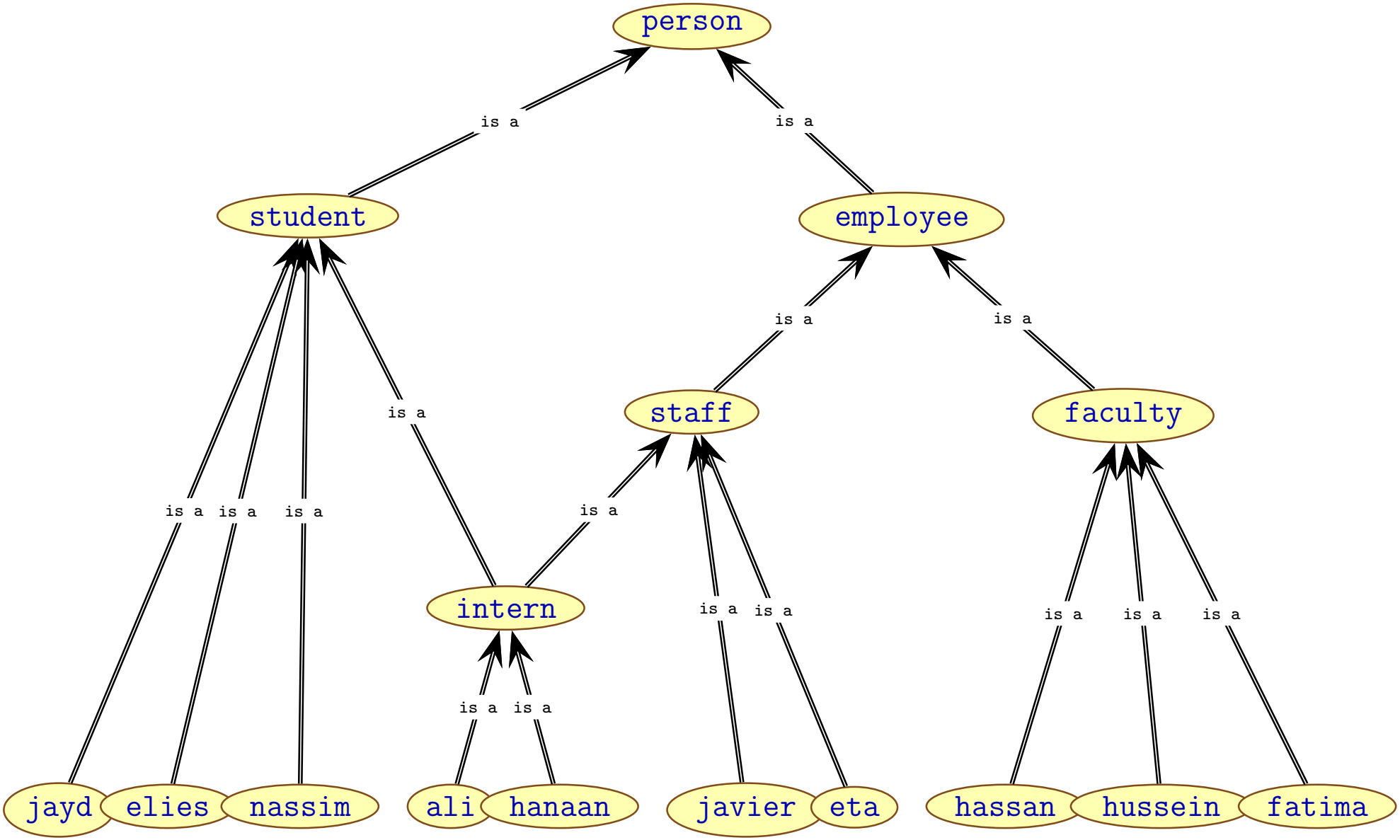
Inconsistent Sort

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

Feature Functionality

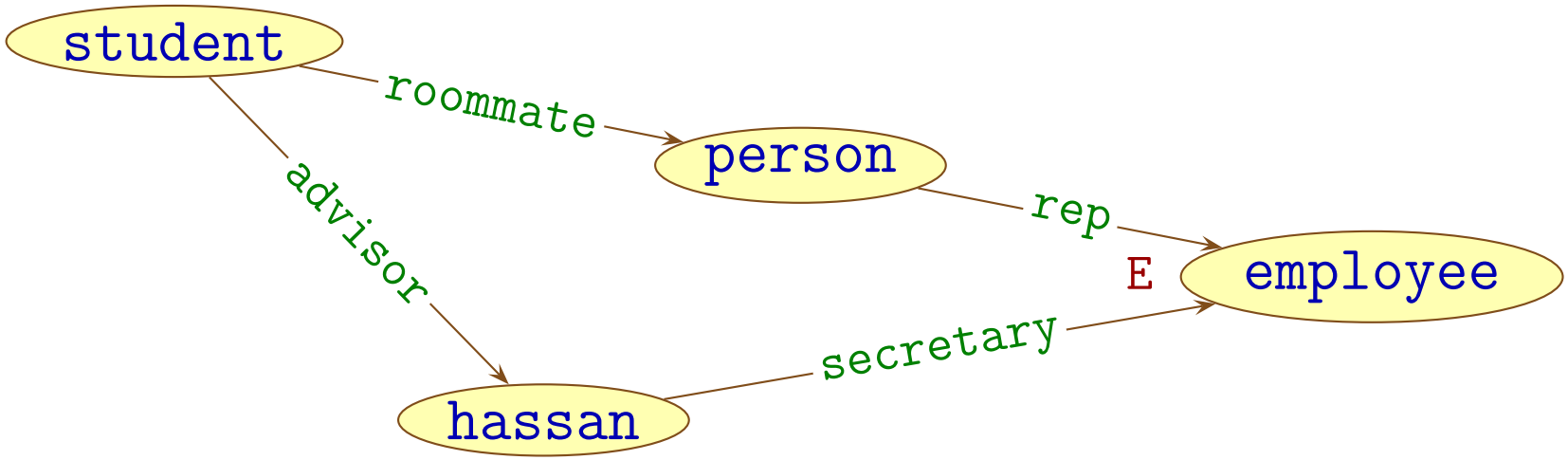
$$\frac{\phi \;\&\; \mathtt{X.f} \doteq \mathtt{X}' \;\&\; \mathtt{X.f} \doteq \mathtt{X}''}{\phi \;\&\; \mathtt{X.f} \doteq \mathtt{X}' \;\&\; \mathtt{X}' \doteq \mathtt{X}''}$$

Partially-Ordered Sort Signature



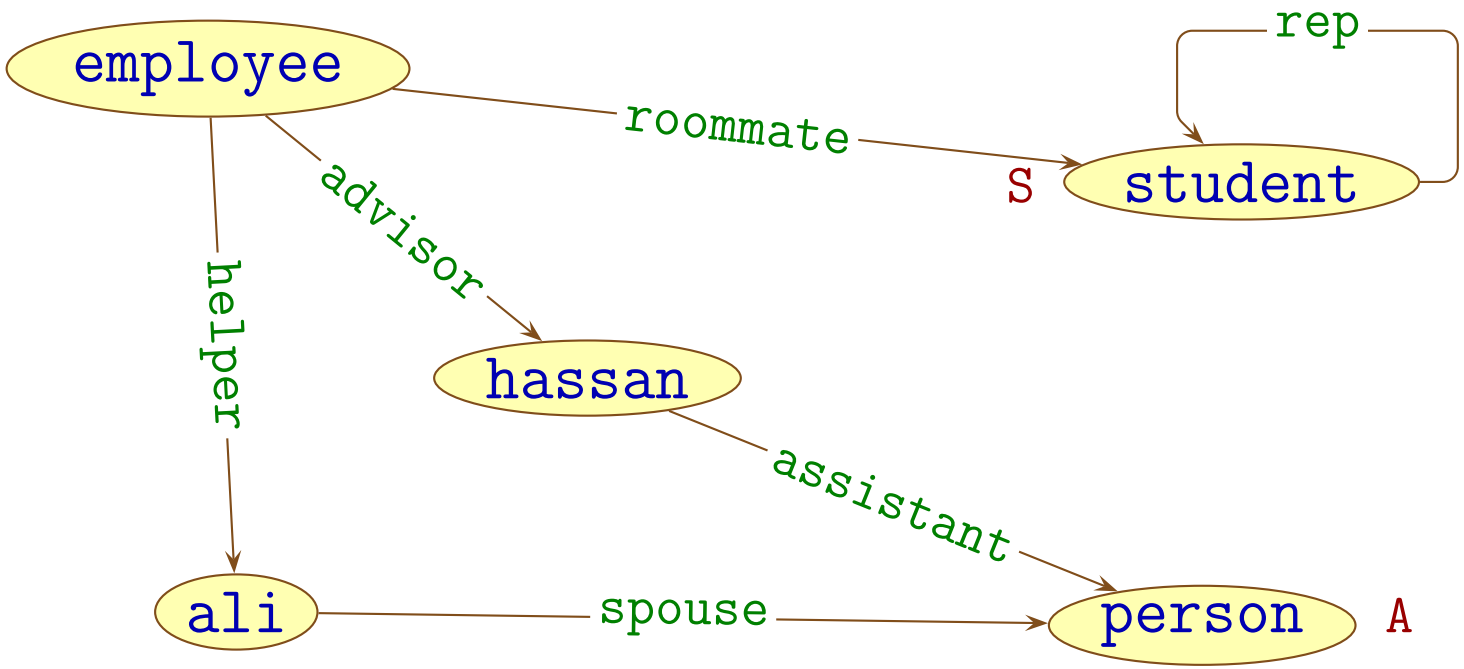
An $\mathcal{OSF}$ term and its $\mathcal{OSF}$ graph

$$\mathbf{t}_1 = \texttt{student}\ (\ \texttt{roommate} \Rightarrow \texttt{person}\ (\ \texttt{rep} \Rightarrow \texttt{E} : \texttt{employee}\ )\ , \texttt{advisor} \Rightarrow \texttt{hassan}\ (\ \texttt{secretary} \Rightarrow \texttt{E}\ )\ )$$

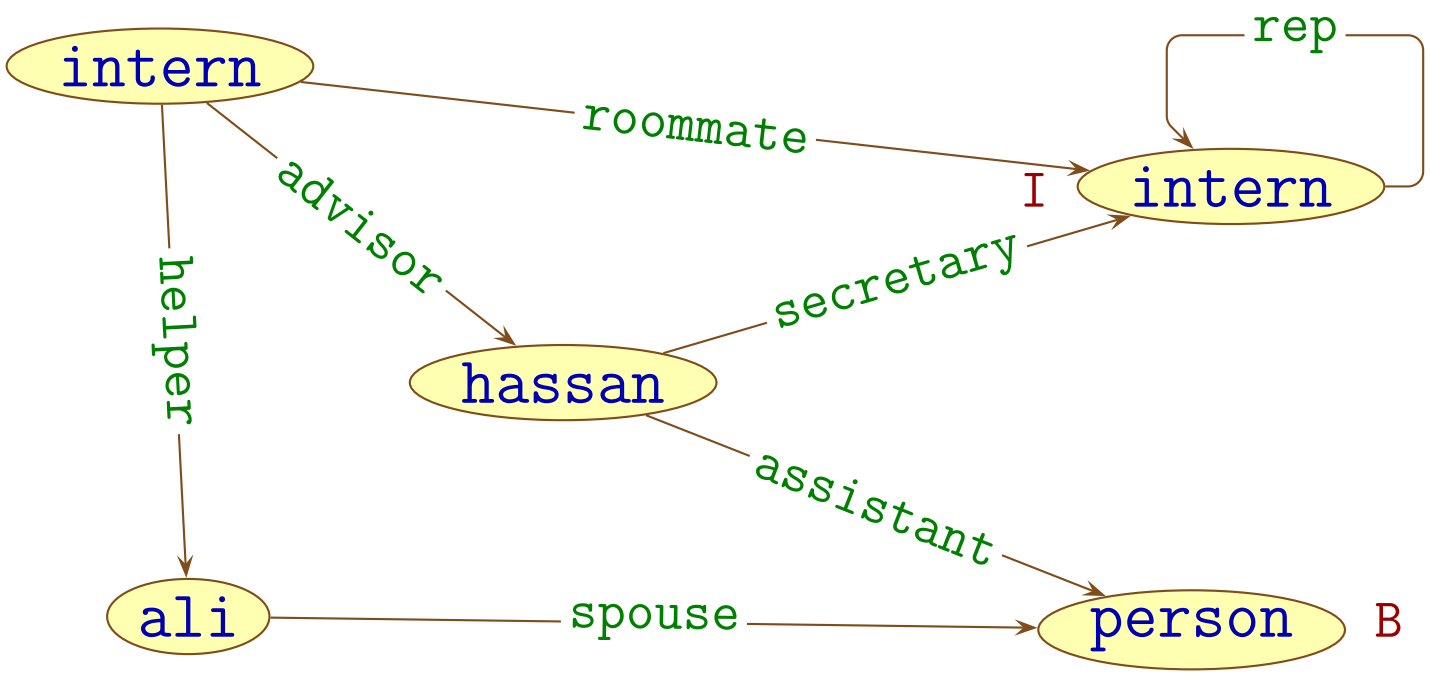


An $\mathcal{OSF}$ term and its $\mathcal{OSF}$ graph

$$
\begin{aligned}
\mathbf{t}_2 \;=\; & \texttt{employee} \\
& (\,\texttt{advisor} \Rightarrow \texttt{hassan}\,(\,\texttt{assistant} \Rightarrow \texttt{A}\,)\,) \\
& (\,\texttt{, roommate} \Rightarrow \texttt{S} : \texttt{student}\,(\,\texttt{rep} \Rightarrow \texttt{S}\,)\,) \\
& (\,\texttt{, helper} \Rightarrow \texttt{ali}\,(\,\texttt{spouse} \Rightarrow \texttt{A}\,)\,)
\end{aligned}
$$

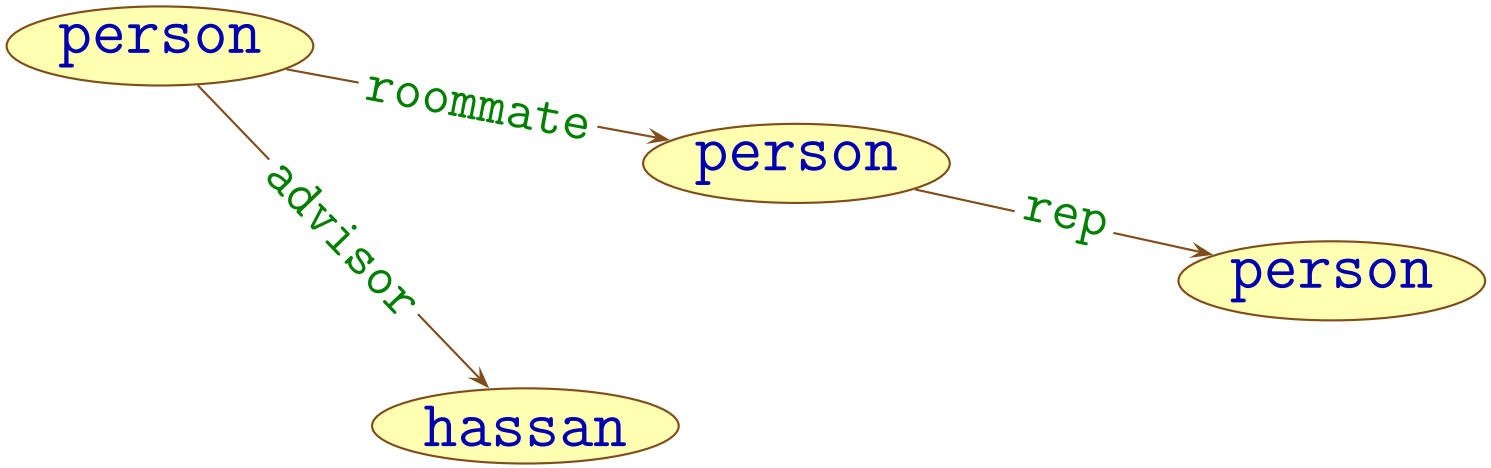


```
t = intern
    ( advisor ⇒ hassan ( assistant ⇒ B,
                         secretary ⇒ I )
    , helper ⇒ ali ( spouse ⇒ B )
    , roommate ⇒ I : intern ( rep ⇒ I ) )
```

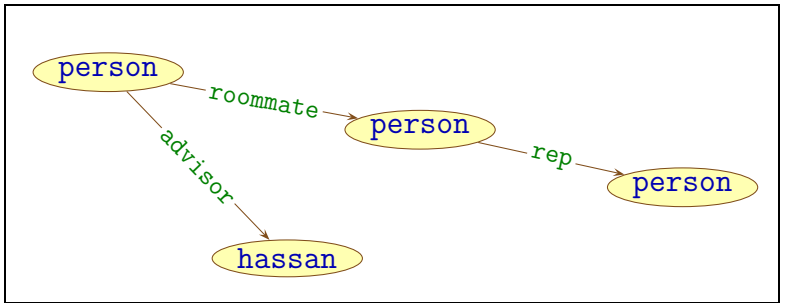


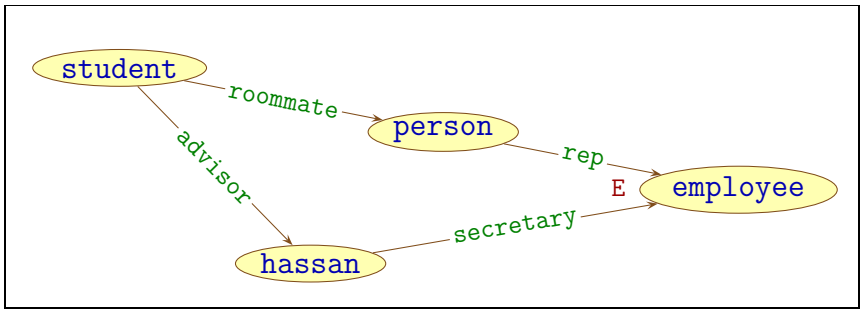
The Least Upper Bound—$\mathcal{OSF}$ Generalization: $\overline{\mathbf{t}} = \mathbf{t}_1 \vee \mathbf{t}_2$

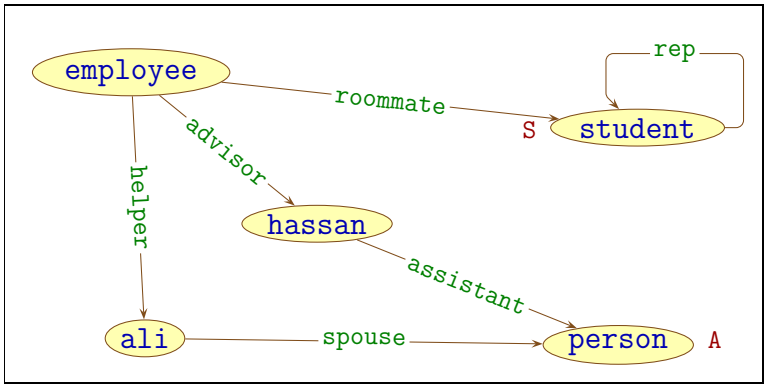
$$\overline{\mathbf{t}} = \texttt{person}\ (\ \texttt{roommate} \Rightarrow \texttt{person}\ (\ \texttt{rep} \Rightarrow \texttt{person}\ )\ ,\ \texttt{advisor} \Rightarrow \texttt{hassan}\ )$$

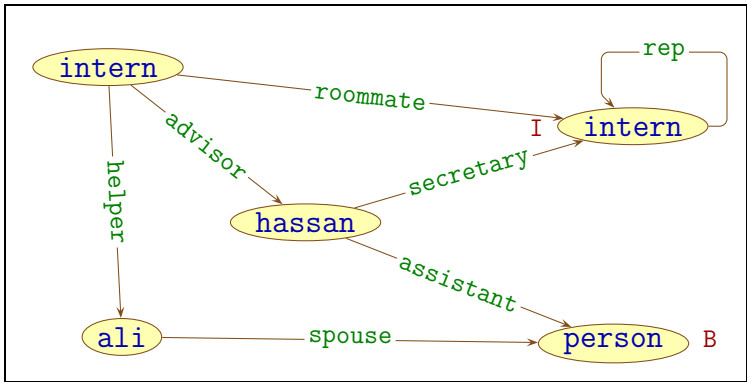


$\mathcal{OSF}$ Inheritance Lattice

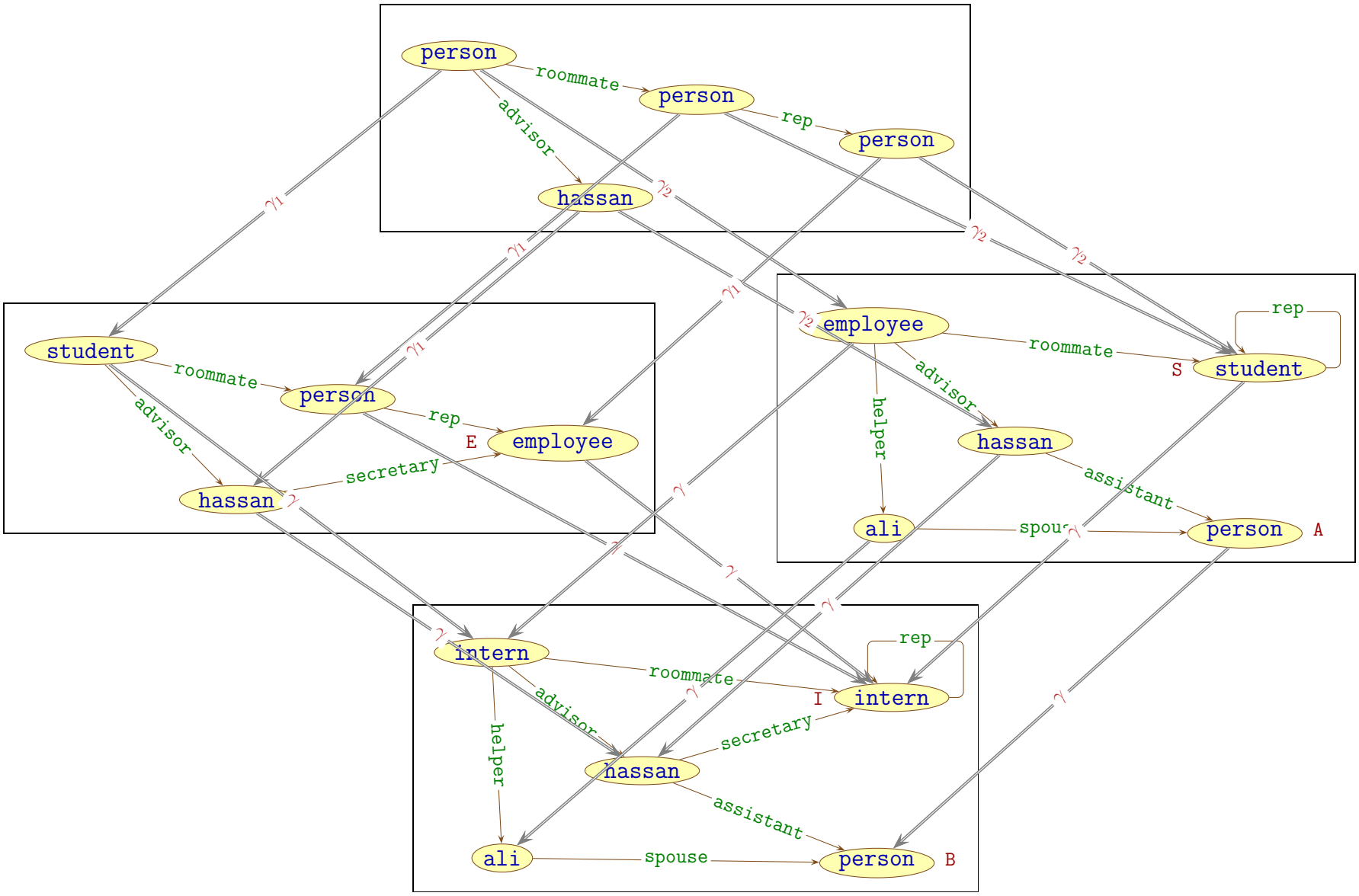








$\mathcal{OSF}$ Endomorphic Inheritance Lattice Diagram



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

- Non-lattice sort signatures, disjunctive sorts, complemented sorts (and actually gain in taxonomic reasoning efficiency!)
- Partial features and element-denoting sorts (see this article)
- Relational features ("roles;" *i.e.*, set-valued features)
- Infinite feature-composition paths (regular expressions)
- Aggregates (*à la* monoid comprehensions)
- Sort definitions (*a.k.a.*, "$\mathcal{OSF}$ theories")

- Semantic Web formalisms
- Graphs as constraints
- $\mathcal{OSF}$ vs. $\mathcal{DL}$
- $\mathcal{LIFE}$: $\mathcal{L}$ogic $\mathcal{I}$nheritance $\mathcal{F}$unctions $\mathcal{E}$quations
- Recapitulation

Reasoning with knowledge expressed as $\mathcal{OWL}$ *sentences* is based on its $\mathcal{DL}$ tableau-semantics *explicitly* ***building models*** by *inductive processing*.

however:

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

An object systematically materializes all its components. . .

Much work is done even if not needed!

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

this is great, because:

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

An object materializes only components that are requested...

No work is done unless needed!

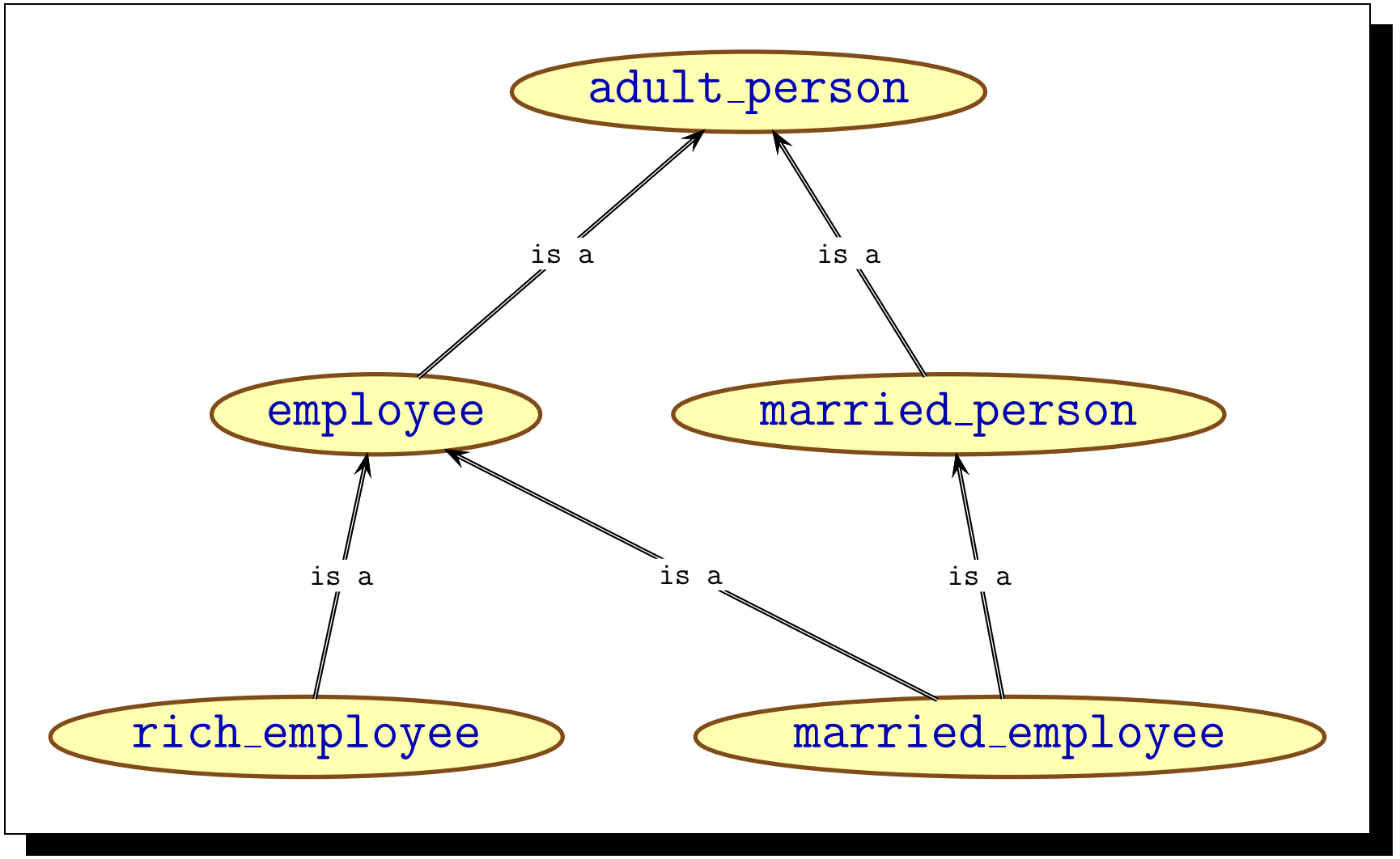
- Semantic Web formalisms
- Graphs as constraints
- $\mathcal{OSF}$ *vs.* $\mathcal{DL}$
- $\mathcal{LIFE}$: $\mathcal{L}$ogic $\mathcal{I}$nheritance $\mathcal{F}$unctions $\mathcal{E}$quations
- Recapitulation

Intuitively:

$\mathcal{LIFE}$: logically and functionally constrained $\mathcal{OSF}$ graphs

Formally:

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



A multiple-inheritance hierarchy

The same hierarchy in Java

```
interface adult_person {
    name id;
    date dob;
    int age;
    String ssn;
}
interface employee extends adult_person {
    Title position;
    String company;
    employee supervisor;
    int salary;
}
interface married_person extends adult_person {
    married_person spouse;
}
interface married_employee extends employee, married_person {
}
interface rich_employee extends employee {
}
```

The same hierarchy in $\mathcal{LIFE}$

```
employee <: adult_person.
married_person <: adult_person.
rich_employee <: employee.
married_employee <: employee.
married_employee <: married_person.

:: adult_person    ( id ⇒ name
                   , dob ⇒ date
                   , age ⇒ int
                   , ssn ⇒ string ).

:: employee        ( position ⇒ title
                   , company ⇒ string
                   , supervisor ⇒ employee
                   , salary ⇒ int ).

:: married_person ( spouse ⇒ married_person ).
```

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

```
:: P : adult_person ( id ⇒ name
                    , dob ⇒ date
                    , age ⇒ A : int
                    , ssn ⇒ string )
   | A = ageInYears(P), A ≥ 18.

---

:: employee         ( position ⇒ T : title
                    , company ⇒ string
                    , supervisor ⇒ E : employee
                    , salary ⇒ S : int )
   | higherRank(E.position, T) , E.salary ≥ S.
```

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

$$:: \mathtt{M} : \mathtt{married\_person} \ (\ \mathtt{spouse} \Rightarrow \mathtt{P} : \mathtt{married\_person} \ )$$

$$\mid \mathtt{P.spouse} = \mathtt{M}.$$

$$:: \mathtt{R} : \mathtt{rich\_employee} \quad (\ \mathtt{company} \Rightarrow \mathtt{I}$$
$$, \ \mathtt{salary} \Rightarrow \mathtt{S} \ )$$

$$\mid \mathtt{stockValue}(\mathtt{I}) = \mathtt{V} \ , \ \mathtt{hasShares}(\mathtt{R}, \mathtt{I}, \mathtt{N}) \ , \ \mathtt{S} + \mathtt{N} * \mathtt{V} \geq 200000.$$

Curious about $\mathcal{LIFE}$? Please check out:

- the $\mathcal{LIFE}$ Tutorial lecture slides
- the `WildLife 1.02` manual

Unfortunately, no $\mathcal{LIFE}$ implementation is available any longer

In any case, it should now be re-implemented with more mature implementation techniques (such as this, this, and this)

model equivalence $\neq$ 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

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

using rules to query ontologies

- ***concept ontologies may be used as constraints by rules*** for ***efficient knowldege-based inference***

as well as, *conversely*:

enhancing ontologies with rule-defined predicates

- ***rule-based conditions in concept definitions boost the expressive power of ontologies*** with ***ordered concepts acting as proof caches***

- Semantic Web formalisms
- Graphs as constraints
- $\mathcal{OSF}$ *vs.* $\mathcal{DL}$
- $\mathcal{LIFE}$: $\mathcal{L}$ogic $\mathcal{I}$nheritance $\mathcal{F}$unctions $\mathcal{E}$quations
- Recapitulation

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

- Structured objects are $\mathcal{OSF}$ **graphs**
- $\mathcal{OSF}$ graphs are conjunctive sets of simple **constraints**
- Constraints are **good**: they provide both **formal** theory and **efficient** processing (order is not important)
- **Formal Logic** is **not** all there is
  (Lattice Theory, Relational Algebra, Constraint Solving, *etc.*)
- even so: **model** theory $\neq$ **proof** theory

Essential questions:

- **syntax**: What's **essential**?
  What's **superfluous**?
  URI's cluttered verbosity makes **confusing notation** (ok, not for human consumption—but still!)

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

- . . .

It is exciting to see the prospects of the W3C. . .

however

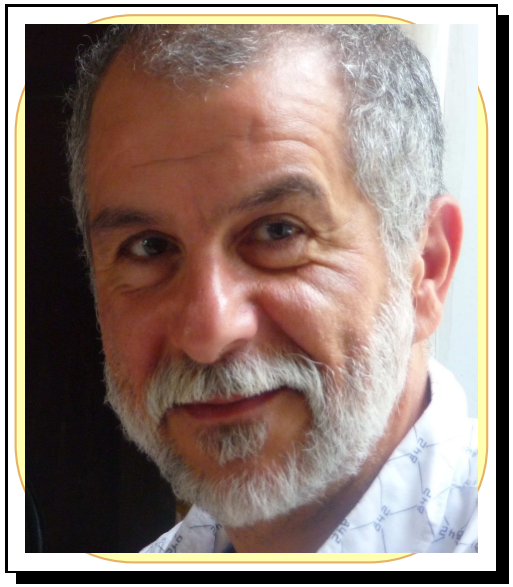
. . . a truly *semantic* web has yet to be achieved. . .

although

. . . we have all the tools to enable it!

Thank You For Your Attention !

Hassan Aït-Kaci



hak@acm.org