Data Systematics: The PSOA RuleML Metamodel Illustrated by Grailog Visualization of Wedding Atoms

(PDF version: ruleml.org/talks/PSOAMetamodelGrailogWedding.pdf**)**

Harold Boley
University of New Brunswick
Faculty of Computer Science
Fredericton, NB, Canada

Started: 13 July 2018

RULES: Logic and Applications (RULES 2019). NTUA, Athens, Greece, 16-17 December 2019

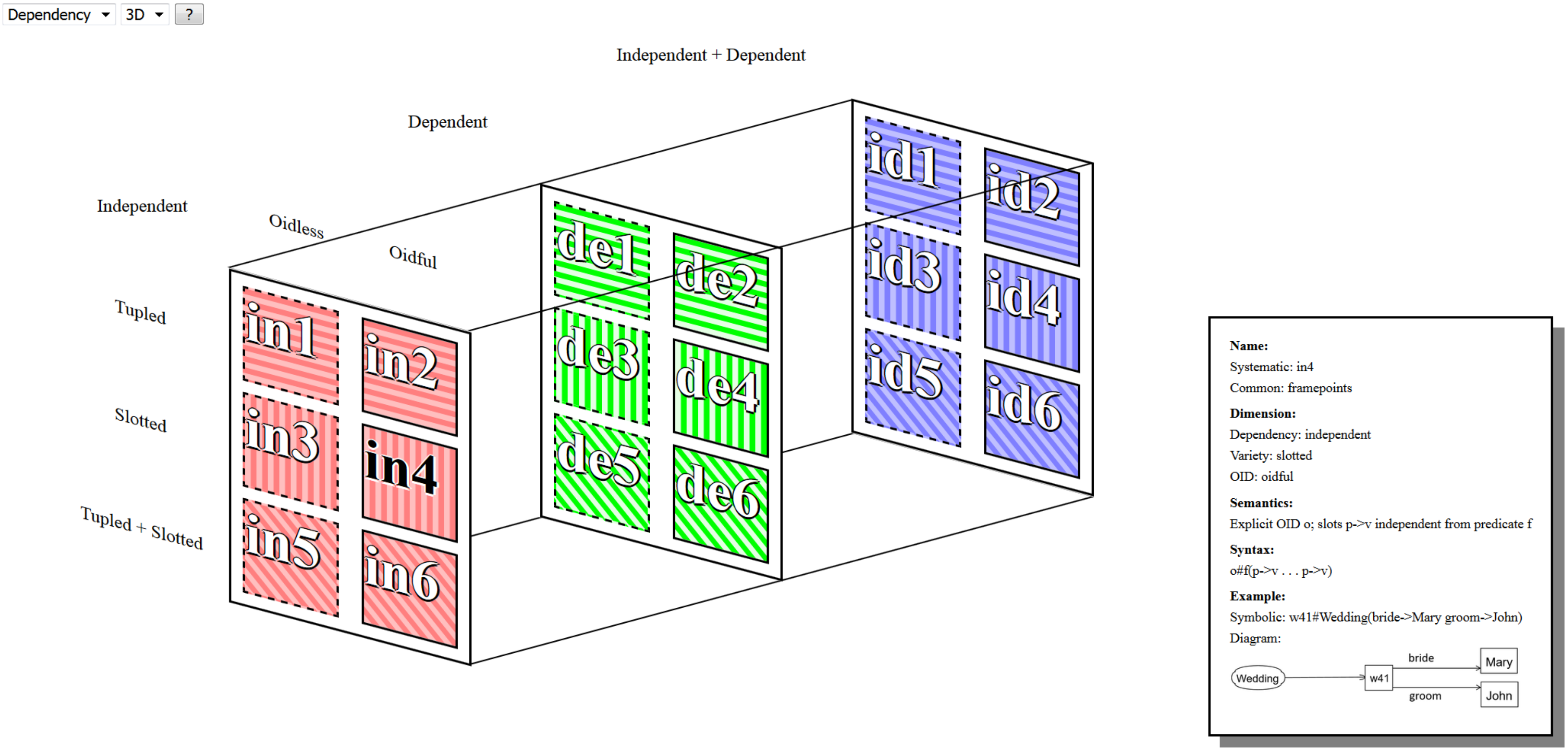
Introduction

- PSOA RuleML builds on a novel **data** systematics: Discover here its new *kinds* of **data**, via 3D metamodel and 2D abstract *visualization syntax* for **semantic intuition**
- Slicing and dicing the *PSOA metamodel cube* (from PSOAPerspectivalKnowledge, Appendix A)
- Exemplify with oidless/oidful, tupled/slotted/combined, independent/dependent/combined atoms (2*3*3 = 18)
- Illustrate all kinds of atoms by Grailog visualization, realized by concrete *(symbolic) presentation syntax* in PSOATransRun
- Informal syntax templates and English semantics (formal in PSOAPerspectivalKnowledge, Sections 4 and 5)
- Experience full metamodel dynamically by online PSOAMetaViz visualization, realized in JavaScript/JSON

Slicing and Dicing the PSOA Metamodel Cube

- The **full metamodel** cube, via 3 (orthogonal) dimensions, systematizes 18 kinds of atoms that are contained in 18 unit cubes (units) named in*j*, de*j*, id*j* (*j*=1,…,6)
- Choosing one of the reductions DVO, VDO, or OVD (s. below), users can slice and dice the cube, in a kind of (meta)OLAP, initially reducing its 3 dimensions to slices of 2 dimensions:
- DVO reduction, via Dependency dimension, to 3 slices, each with 6 units structured by Variety-row (tupled/slotted/combined) and OID-column (oidless/oidful) dimensions:
  - 6 **in**dependent units **in***j* (*j*=1,…,6) vs. 6 **de**pendent units **de***j* (*j*=1,…,6) vs. 6 combined **i**ndependent+**d**ependent units **id***j* (*j*=1,…,6)
- The **core metamodel** is an 8-unit subcube of the full metamodel cube, which can be reduced, DVO-style, to 2 Dependency slices: in1-in4 and de1-de4
  - Each includes a 'landmark' unit: ***framepoint*** (in4) and ***relationship*** (de1) atoms
- VDO reduction (e.g., for full metamodel), via Variety dimension, to 3 slices, each with 6 units structured by Dependency-row and OID-column dimensions:
  - 6 tupled+slotted units in*j*, de*j*, id*j* (*j*=5,6) vs. 6 slotted units in*j*, de*j*, id*j* (*j*=3,4) vs. 6 tupled units in*j*, de*j*, id*j* (*j*=1,2)
- OVD reduction (e.g., for full metamodel), via OID dimension, to 2 slices, each with 9 units structured by Variety-row and Dependency-column dimensions:
  - 9 oidful units in*j*, de*j*, id*j* (*j*=2,4,6) vs. 9 oidless units in*j*, de*j*, id*j* (*j*=1,3,5)

The PSOAMetaViz Cube with Current Selection of **Framepoint Atoms from Independent Slice**



Name:

Systematic: in4

Common: framepoints

Dimension:

Dependency: independent

Variety: slotted

OID: oidful

Semantics:

Explicit OID o; slots p->v independent from predicate f

Syntax:

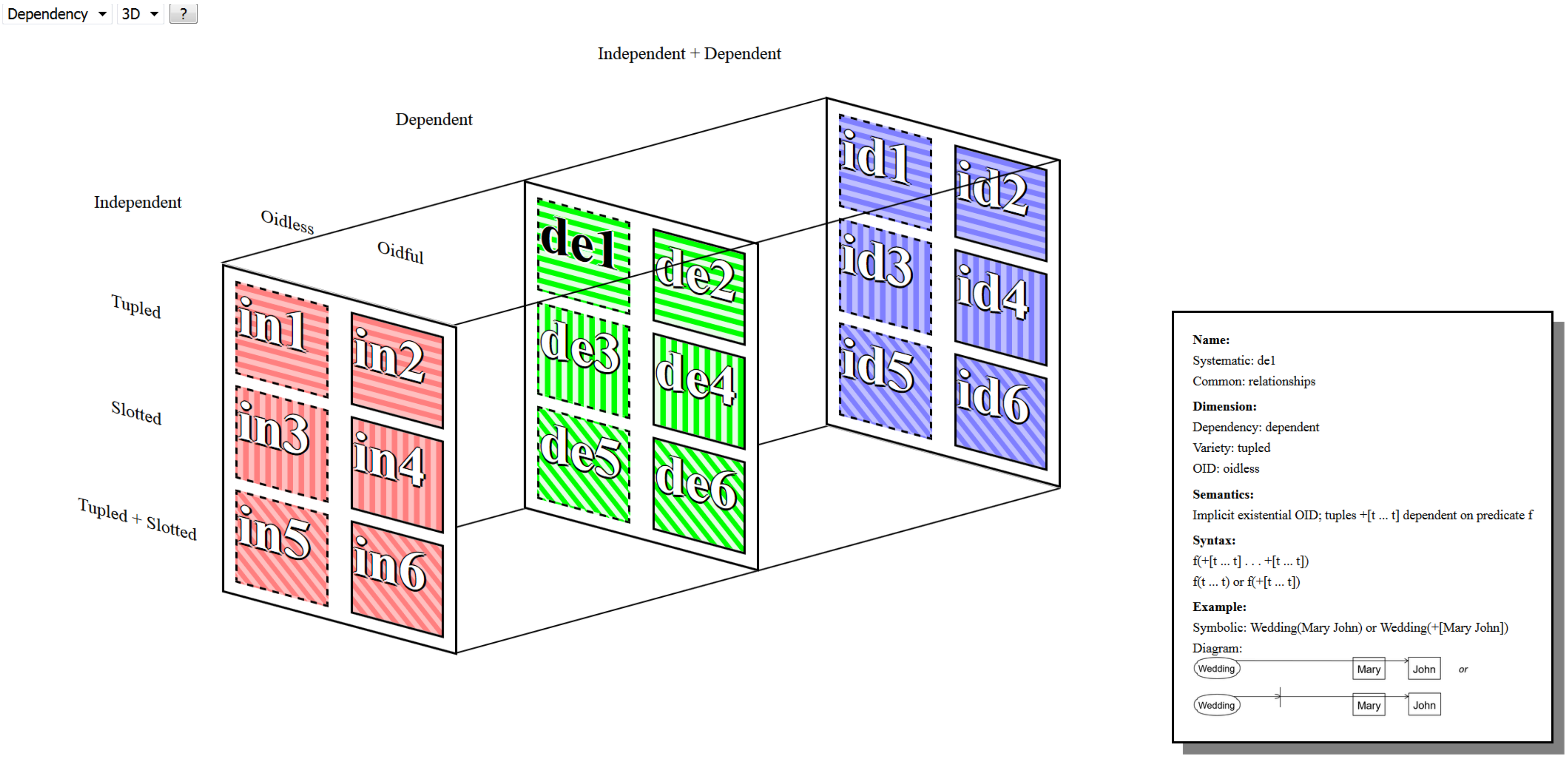
o#f(p->v . . . p->v)

Example:

Symbolic: w41#Wedding(bride->Mary groom->John)

Diagram:

The PSOAMetaViz Cube with Current Selection of **Relationship Atoms from Dependent Slice**



Name:

Systematic: de1

Common: relationships

Dimension:

Dependency: dependent

Variety: tupled

OID: oidless

Semantics:

Implicit existential OID; tuples +[t ... t] dependent on predicate f

Syntax:

f(+[t ... t] . . . +[t ... t])

f(t ... t) or f(+[t ... t])

Example:

Symbolic: Wedding(Mary John) or Wedding(+[Mary John])

Diagram:

Running Example

Wedding *events with*
bride *and* groom *roles*
etc.

Disambiguating “groom” using a **de**pendent slot (e.g., within ***pairpoints***):

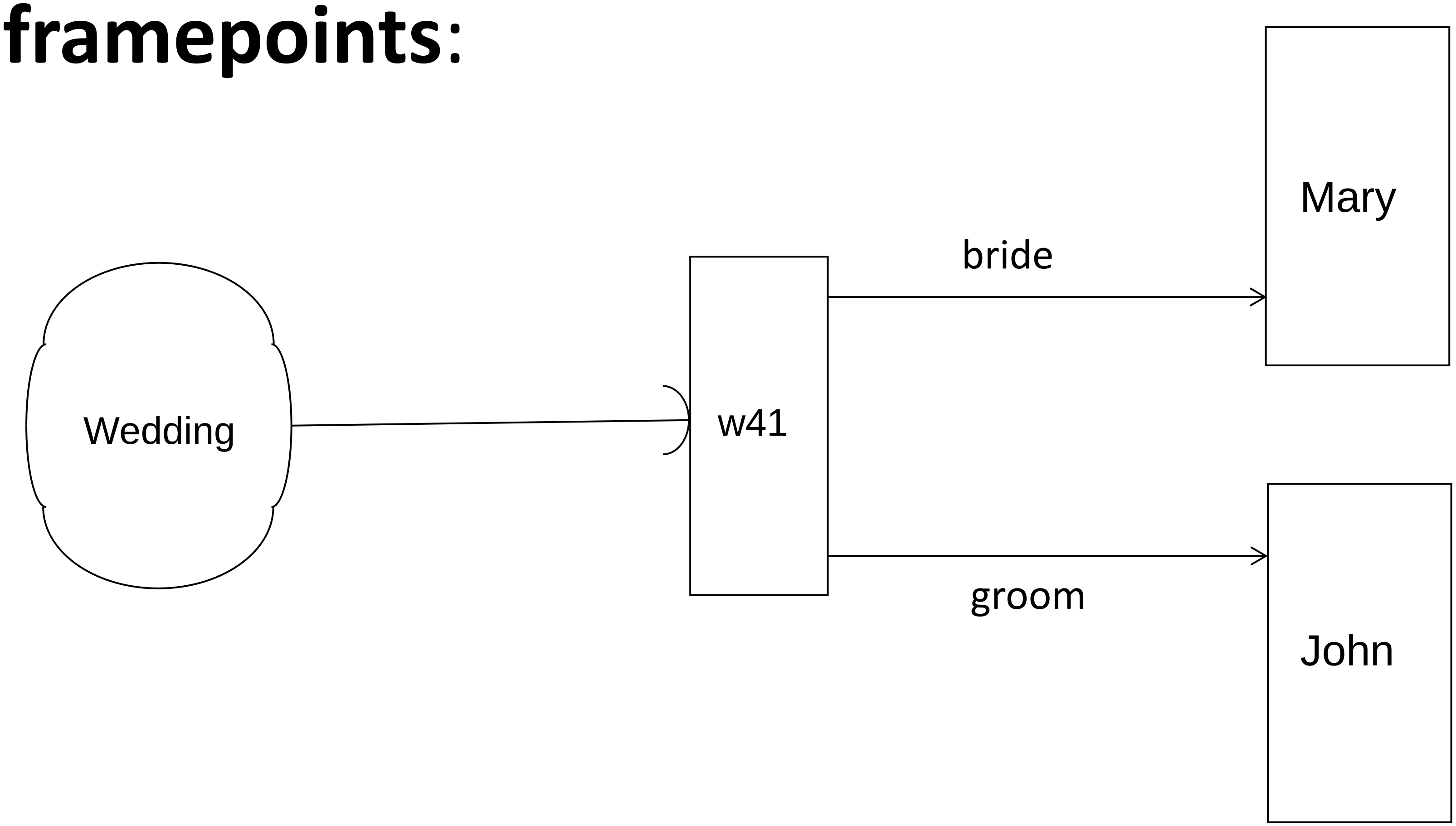
noun: **groom**

1. a person employed to take care of horses.
2. **a bridegroom**

https://www.google.com/search?q=groom

Move between *visualization syntax* …

framepoints:



… and *(symbolic) presentation syntax*

framepoints:

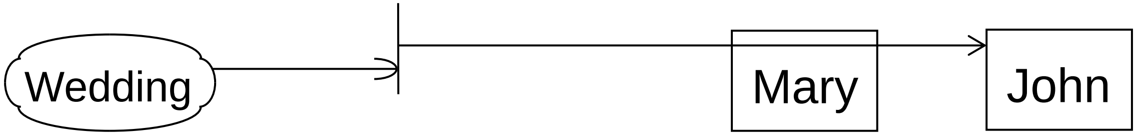
```
w41#Wedding(
                bride->Mary
                groom->John
               )
```

Exemplifying the Dependency Slices

Core oidless/oidful, tupled/slotted atoms that are **in**dependent:

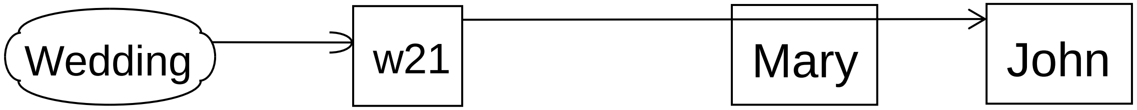
Grailog:

in1. for single-tuple: shelfships



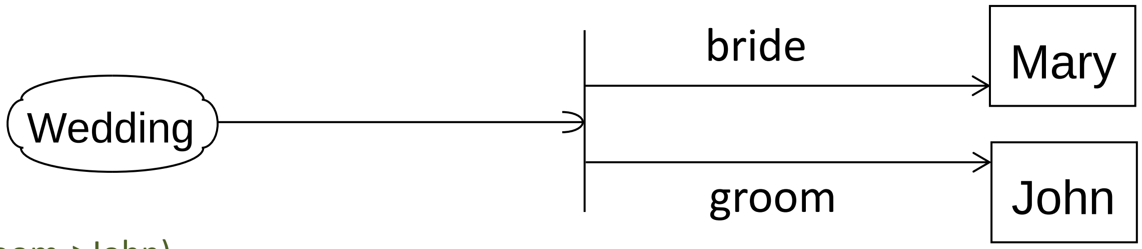
Wedding(-[Mary John])

in2. for single-tuple: shelfpoints



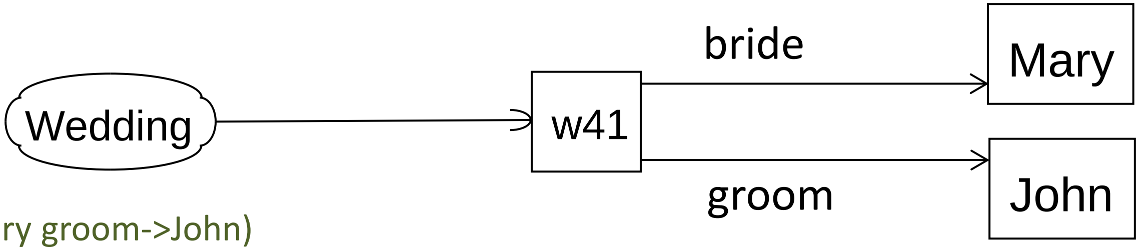
w21#Wedding(-[Mary John])

in3. frameships



Wedding(bride->Mary groom->John)

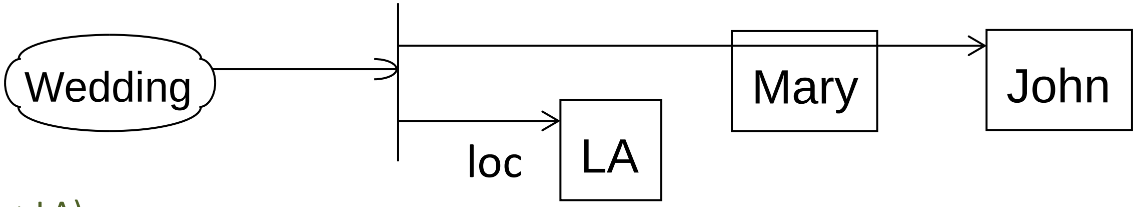
in4: **framepoints**



w41#Wedding(bride->Mary groom->John)

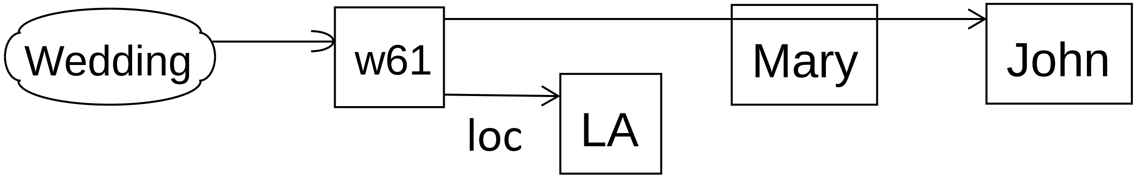
Extra oidless/oidful, combined tupled+slotted atoms that are **in**dependent:

in5. for single-tuple:
shelframeships



Wedding(-[Mary John] loc->LA)

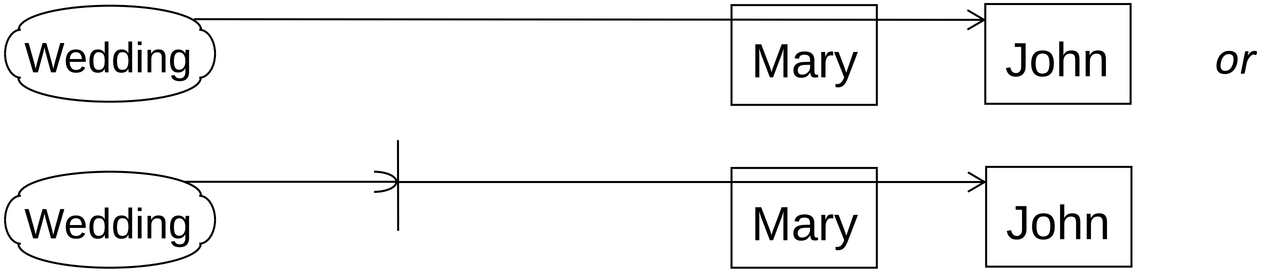
in6. for single-tuple:
shelframepoints



w61#Wedding(-[Mary John] loc->LA)

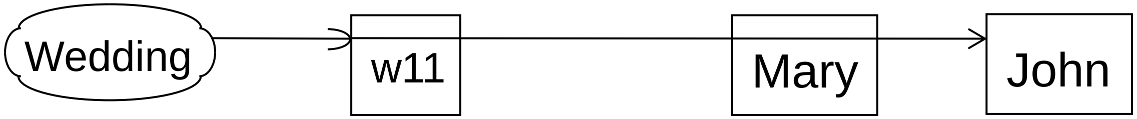
Core oidless/oidful, tupled/slotted atoms that are **de**pendent:

de1. for single-tuple: **relationships**



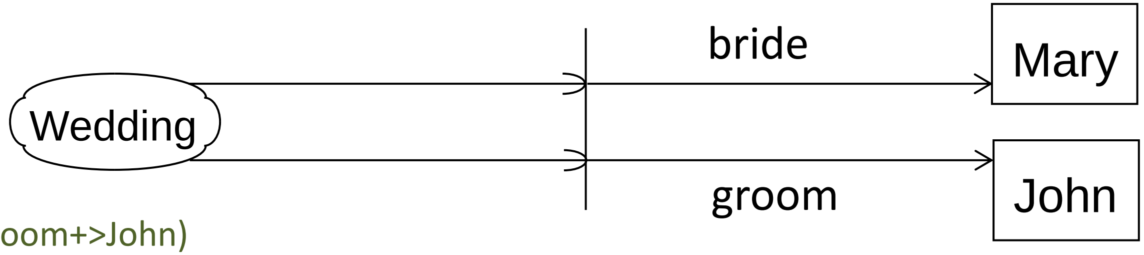
Wedding(Mary John) *or* Wedding(+[Mary John])

de2. for single-tuple: relationpoints



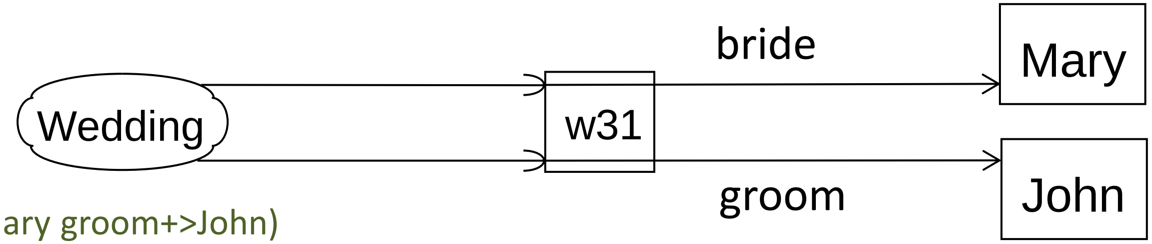
w11#Wedding(+[Mary John])

de3: pairships



Wedding(bride+>Mary groom+>John)

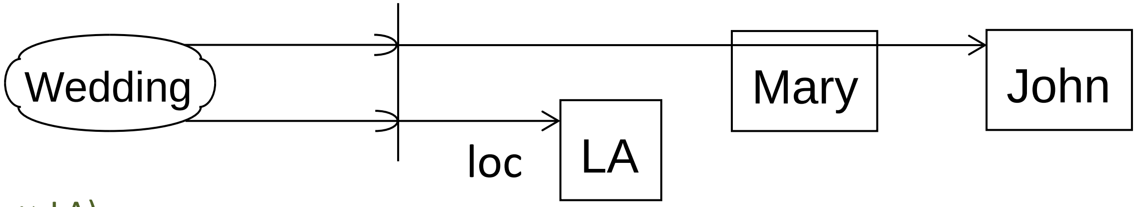
de4. pairpoints



w31#Wedding(bride+>Mary groom+>John)

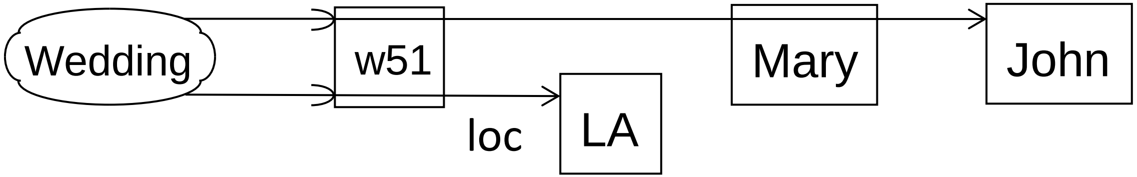
Extra oidless/oidful, combined tupled+slotted atoms that are **de**pendent:

de5. for single-tuple: relpairships



Wedding(+[Mary John] loc+>LA)

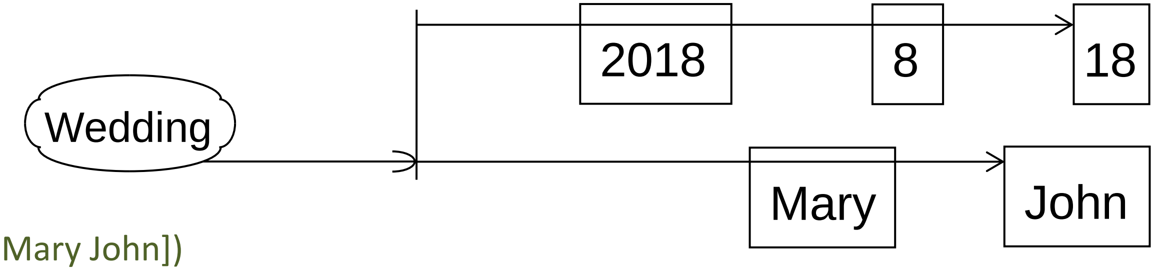
de6. for single-tuple: relpairpoints



w51#Wedding(+[Mary John] loc+>LA)

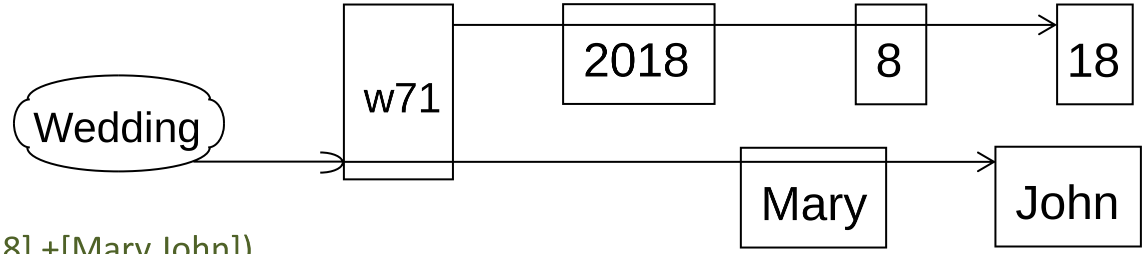
Adding oidless/oidful, tupled/slotted, combined **i**ndependent+**d**ependent atoms:

id1



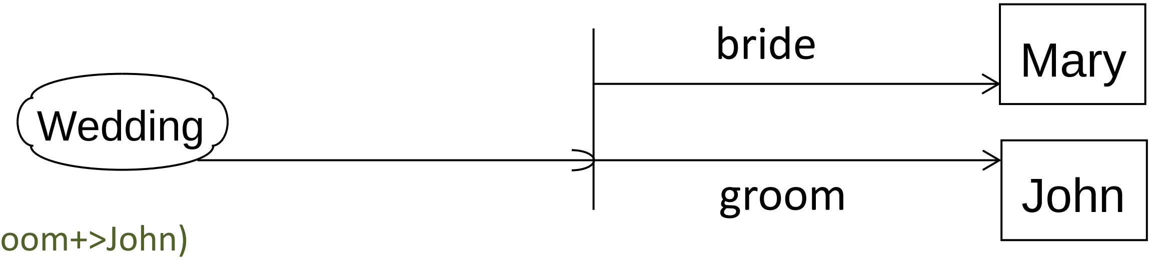
Wedding(-[2018 8 18] +[Mary John])

id2



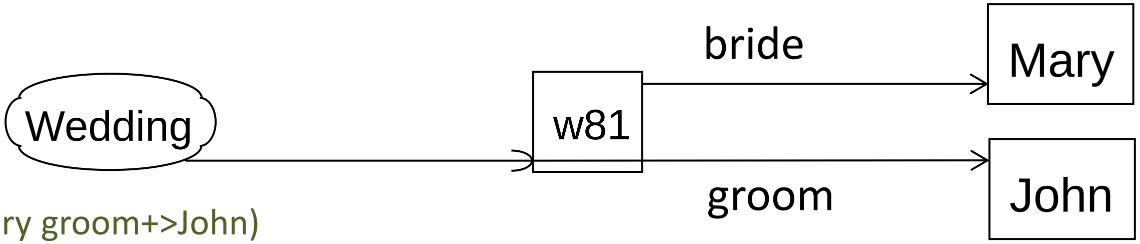
w71#Wedding(-[2018 8 18] +[Mary John])

id3



Wedding(bride->Mary groom+>John)

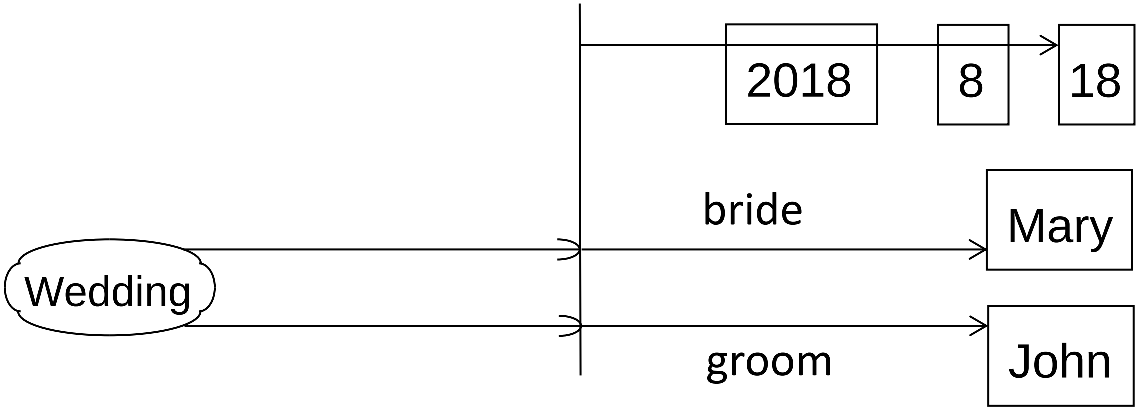
id4



w81#Wedding(bride->Mary groom+>John)

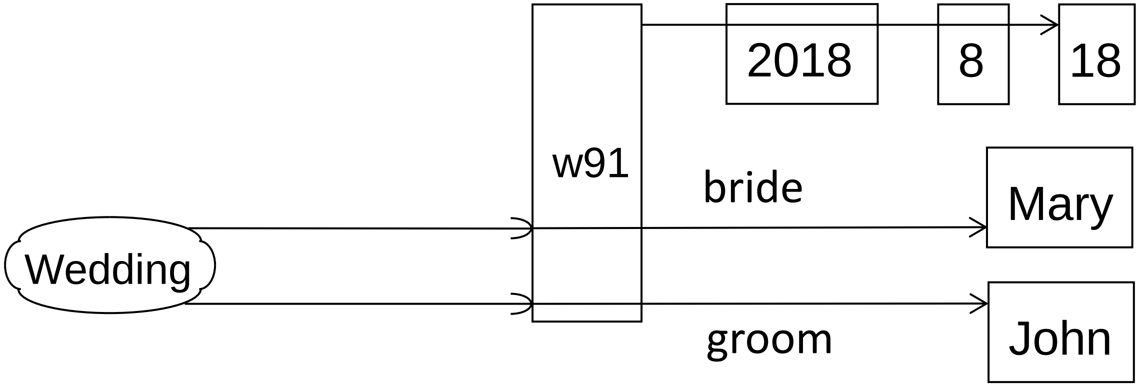
Also oidless/oidful, combined tupled+slotted, combined **i**ndependent+**d**ependent:

id5



Wedding(-[2018 8 18] bride+>Mary groom+>John)

id6



w91#Wedding(-[2018 8 18] bride+>Mary groom+>John)

Syntax and Semantics of Atoms

Core oidless/oidful, tupled/slotted atoms that are **in**dependent:

in1. for single-tuple:
shelfships

f(-[t ... t] . . . -[t ... t])

f(-[t ... t])

Implicit existential OID; tuples -[t ... t] independent from predicate f

in2. for single-tuple:
shelfpoints

o#f(-[t ... t] . . . -[t ... t])

o#f(-[t ... t])

Explicit OID o; tuples -[t ... t] independent from predicate f

in3: frameships

f(p->v . . . p->v)

Implicit existential OID; slots p->v independent from predicate f

in4: **framepoints**

o#f(p->v . . . p->v)

Explicit OID o; slots p->v independent from predicate f

Extra oidless/oidful, combined tupled+slotted atoms that are **in**dependent:

in5. for single-tuple: shelframeships

f(-[t ... t] . . . -[t ... t] p->v . . . p->v)

f(-[t ... t] p->v . . . p->v)

Implicit existential OID; descriptors independent from predicate f

in6. for single-tuple: shelframepoints

o#f(-[t ... t] . . . -[t ... t] p->v . . . p->v)

o#f(-[t ... t] p->v . . . p->v)

Explicit OID o; descriptors independent from predicate f

Core oidless/oidful, tupled/slotted atoms that are **de**pendent:

de1. for single-tuple:
relationships

f(+[t ... t] . . . +[t ... t])

f(+[t ... t]) *or* f(t ... t)

Implicit existential OID; tuples +[t ... t] dependent on predicate f

de2. for single-tuple:
relationpoints

o#f(+[t ... t] . . . +[t ... t])

o#f(+[t ... t]) *or* o#f(t ... t)

Explicit OID o; tuples +[t ... t] dependent on predicate f

de3: pairships

f(p+>v . . . p+>v)

Implicit existential OID; slots p+>v dependent on predicate f

de4: pairpoints

o#f(p+>v . . . p+>v)

Explicit OID o; slots p+>v dependent on predicate f

Extra oidless/oidful, combined tupled+slotted atoms that are **de**pendent:

de5. for single-tuple:
relpairships

f(+[t ... t] . . . +[t ... t] p+>v . . . p+>v)

Implicit existential OID; descriptors dependent on predicate f

f(+[t ... t] p+>v . . . p+>v) *or* f(t ... t p+>v . . . p+>v)

de6. for single-tuple:
relpairpoints

o#f(+[t ... t] . . . +[t ... t] p+>v . . . p+>v)

Explicit OID o; descriptors dependent on predicate f

o#f(+[t ... t] p+>v . . . p+>v) *or* o#f(t ... t p+>v . . . p+>v)

Adding oidless/oidful, tupled/slotted, combined **i**ndependent+**d**ependent atoms:

id1

f(+[t ... t] . . . +[t ... t] -[t ... t] . . . -[t ... t])	Implicit existential OID; both in/dependent tuples w.r.t. predicate f

id2

o#f(+[t ... t] . . . +[t ... t] -[t ... t] . . . -[t ... t])	Explicit OID o; both in/dependent tuples w.r.t. predicate f

id3

f(p+>v . . . p+>v p->v . . . p->v)	Implicit existential OID; both in/dependent slots w.r.t. predicate f

id4

o#f(p+>v . . . p+>v p->v . . . p->v)	Explicit OID o; both in/dependent slots w.r.t. predicate f

Also oidless/oidful, combined tupled+slotted, combined **i**ndependent+**d**ependent:

id5

```
f(+[t ... t] . . . +[t ... t]
  -[t ... t] . . . -[t ... t]
  p+>v . . . p+>v
  p->v . . . p->v)
```

Implicit existential OID; both in/dependent descriptors w.r.t. predicate f

id6

```
o#f(+[t ... t] . . . +[t ... t]
     -[t ... t] . . . -[t ... t]
     p+>v . . . p+>v
     p->v . . . p->v)
```

Explicit OID o; both in/dependent descriptors w.r.t. predicate f

Conclusions

- Full PSOA metamodel cube visualized dynamically by PSOAMetaViz, and atoms (e.g., data facts) in Grailog, to significantly facilitate learning PSOA RuleML
- Facts complemented by (interoperation) rules, including for core interoperation path de1-de3-de4-in4, e.g. abridged to one PSOA rule: http://wiki.ruleml.org/index.php/PSOA_RuleML_Bridges_Graph_and_Relational_Databases
- Core path augmented to roundtrip between wedding atoms: http://wiki.ruleml.org/index.php/Exploring_the_PSOA_RuleML_Space_of_Core_Atoms
- Use sample ground-atom facts, also augmented by rules, for ground- and non-ground-atom queries in PSOATransRun
- PSOA RuleML 1.03 being standardized by Relax NG schemas for XML-serialized facts and rules: http://wiki.ruleml.org/index.php/PSOA_RuleML#Syntaxes
- PSOA metamodel transferrable to other languages