Complex Schema Mapping and Linking Data: Beyond Binary Predicates

Jacobo Rouces
Aalborg University
jacobo@rouces.org

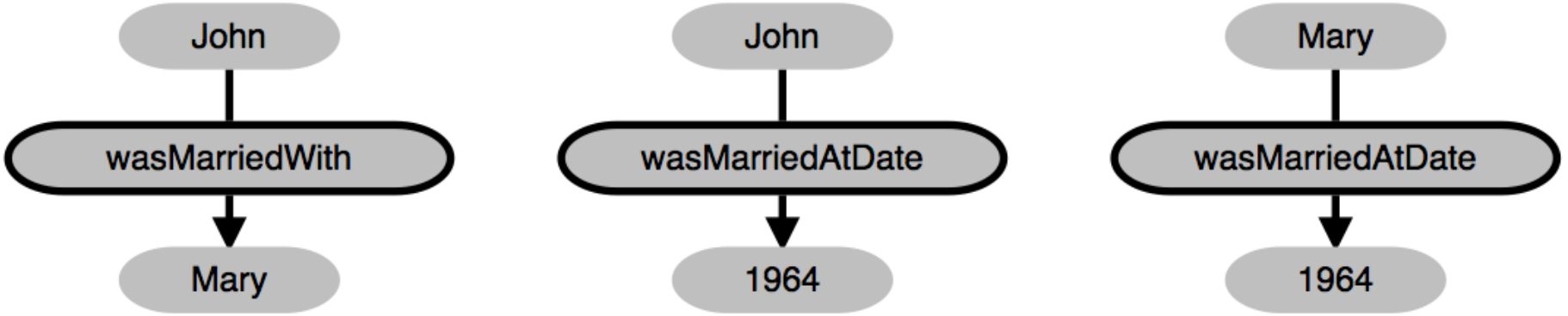
Gerard de Melo
Tsinghua University
gdm@demelo.org

Katja Hose
Aalborg University
khose@cs.aau.dk

Overview

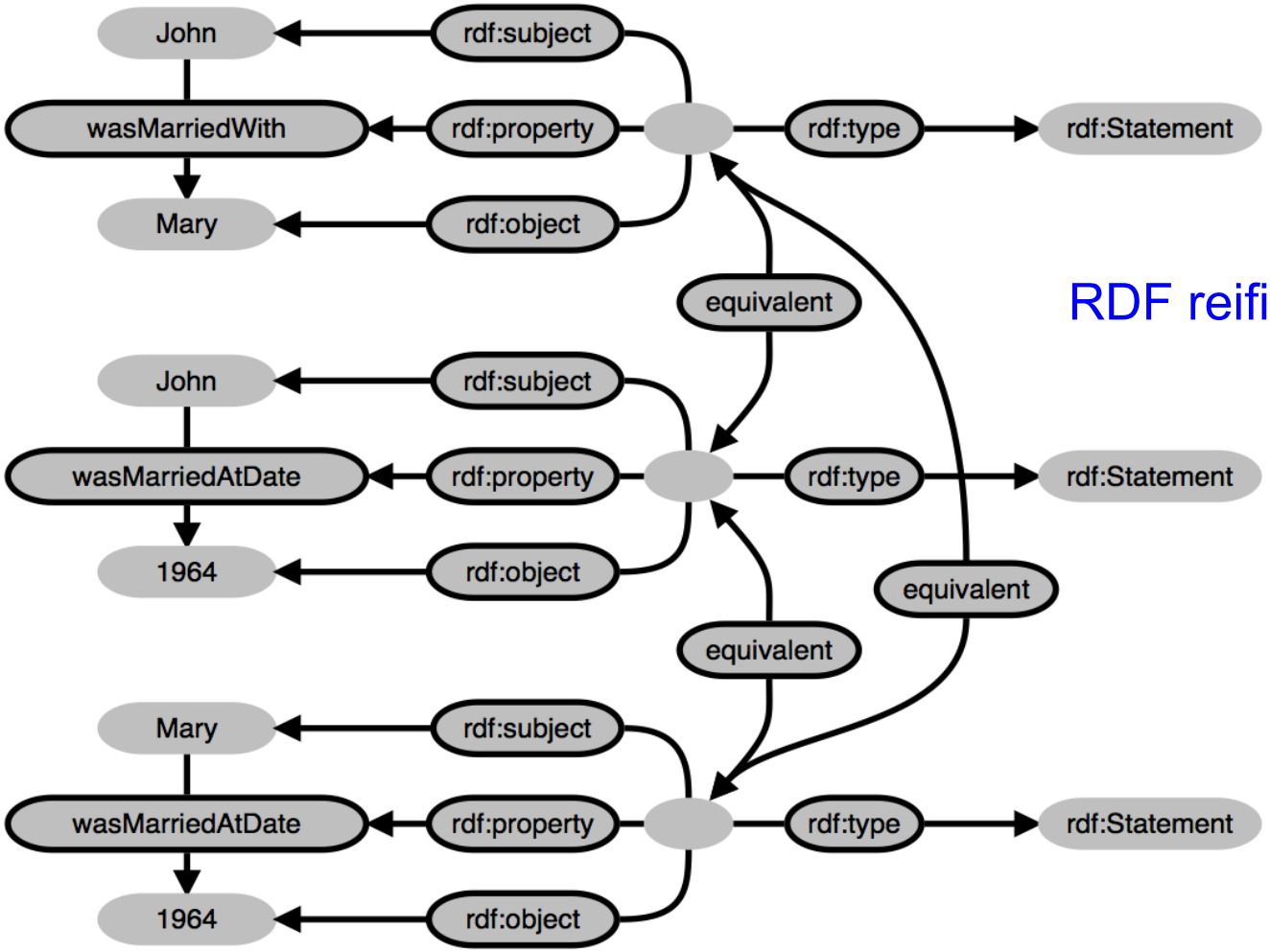
- Data Heterogeneity in the LOD Cloud
- FrameBase
- Creation of complex mappings with FrameBase hub
- Conclusion & Future Work

Data Heterogeneity in the LOD Cloud



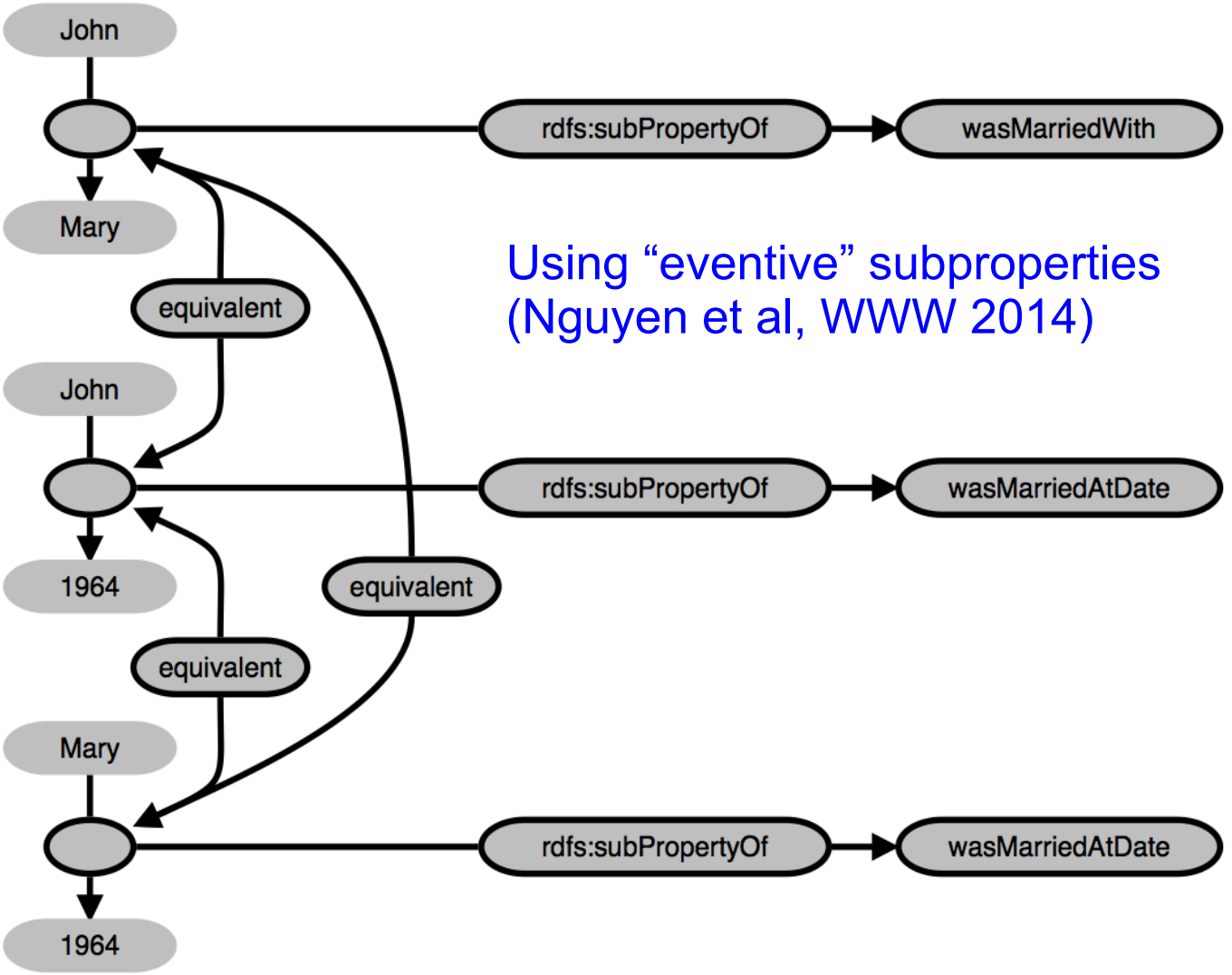
Using Direct Binary Relations *(used as "default" mode in most KBs)*

Data Heterogeneity in the LOD Cloud

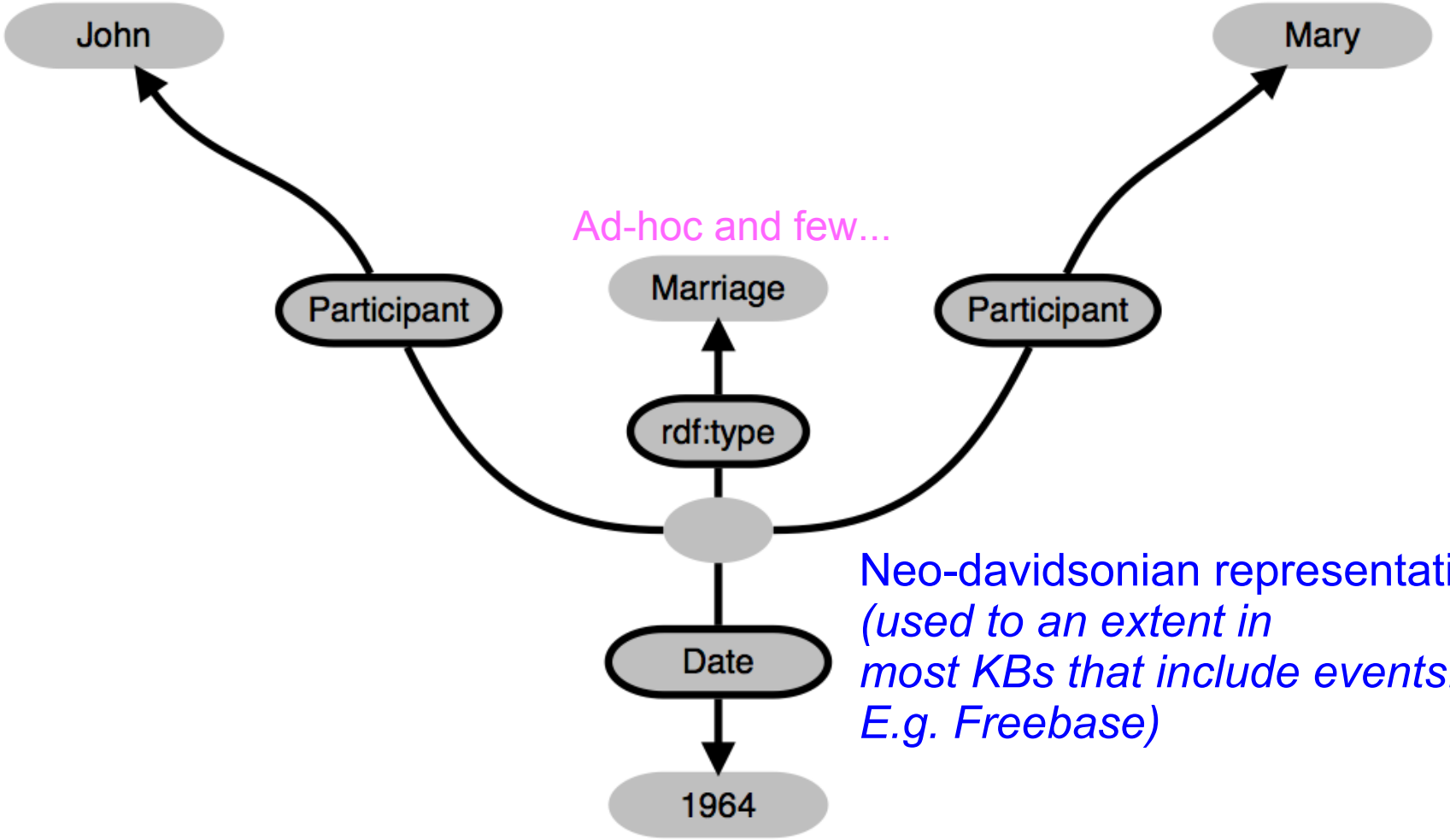


RDF reification *(YAGO)*

Data Heterogeneity in the LOD Cloud

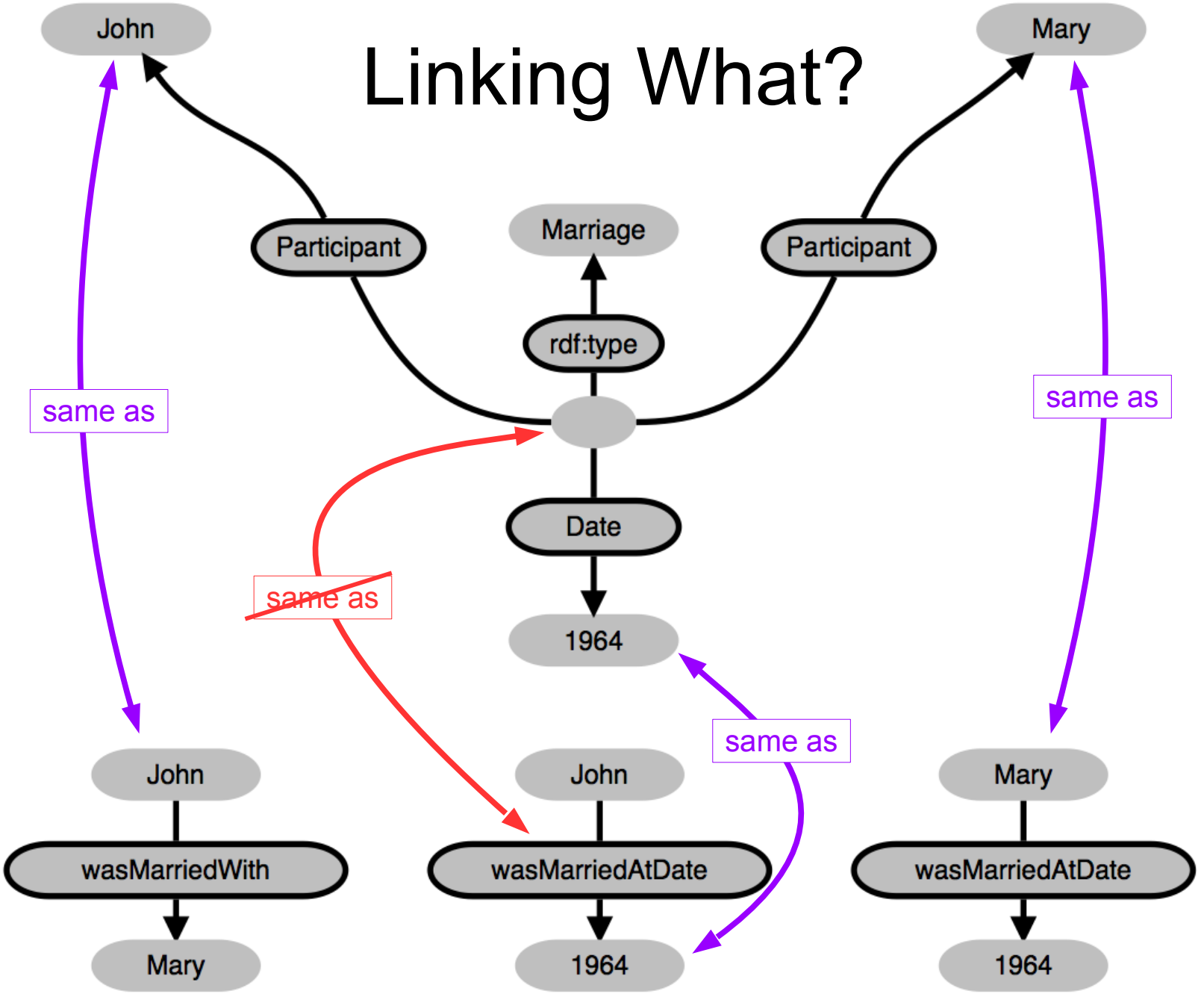


Data Heterogeneity in the LOD Cloud

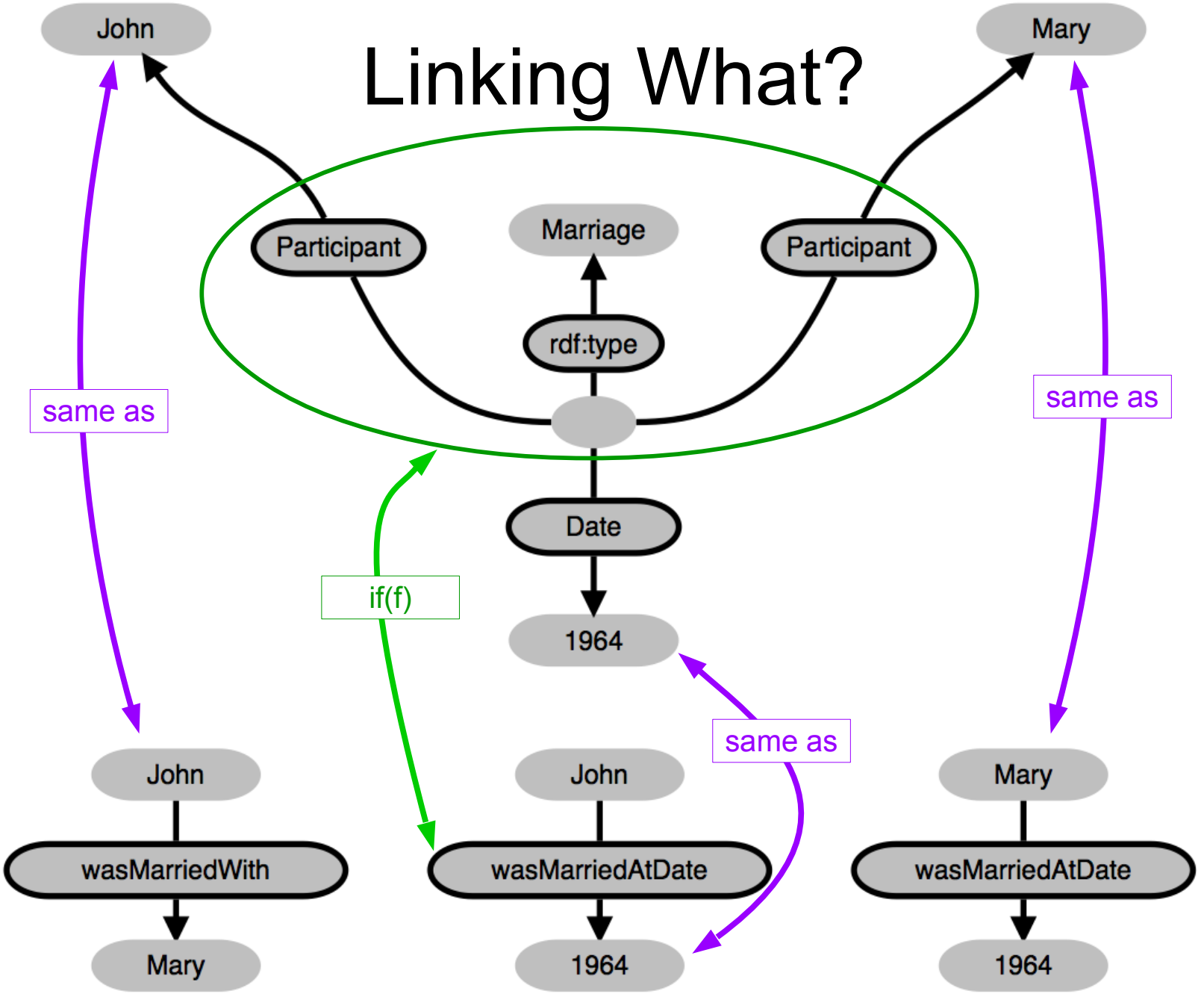


Neo-davidsonian representations
(used to an extent in most KBs that include events. E.g. Freebase)

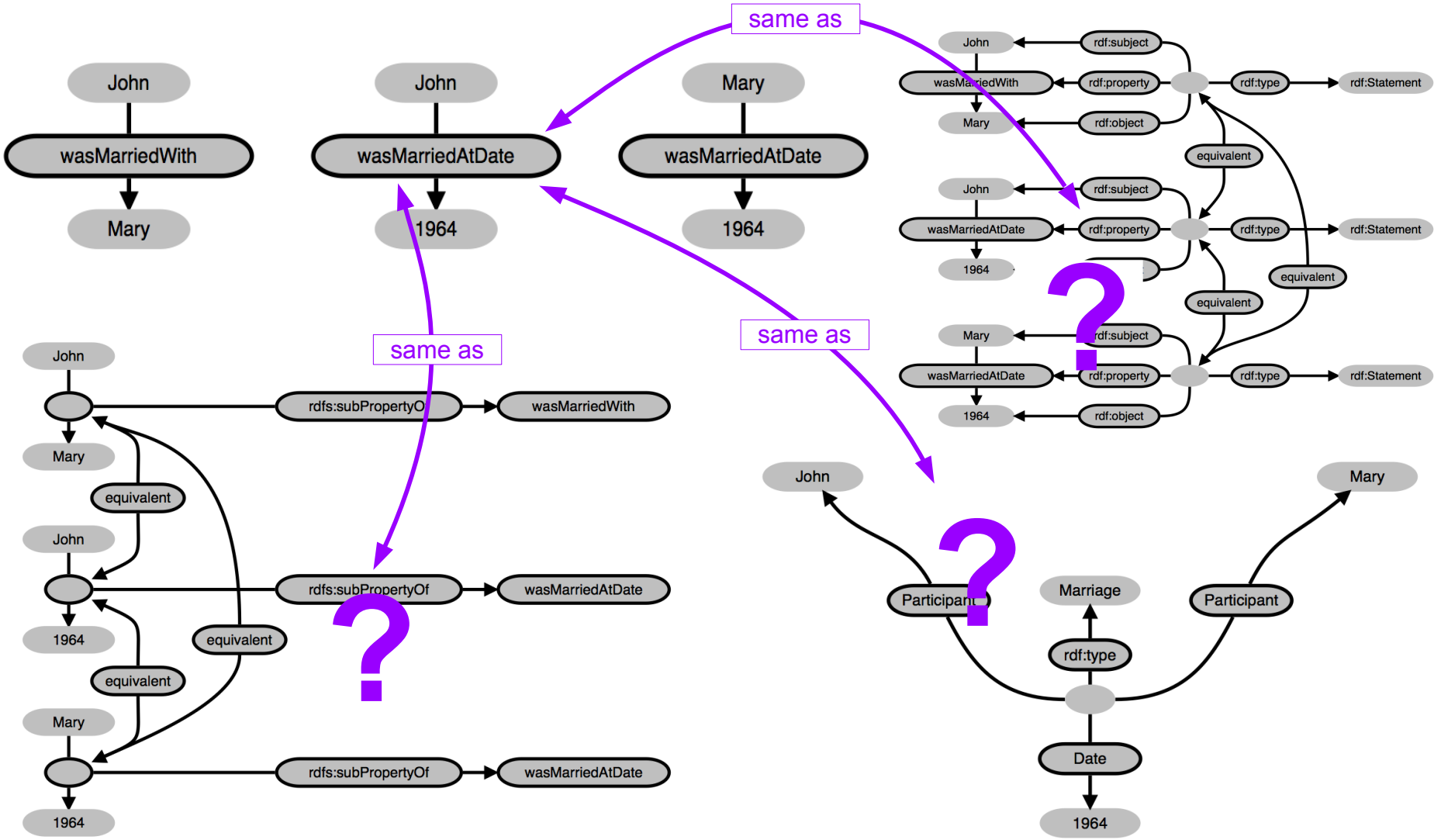
Linking What?



Linking What?



Linking What?

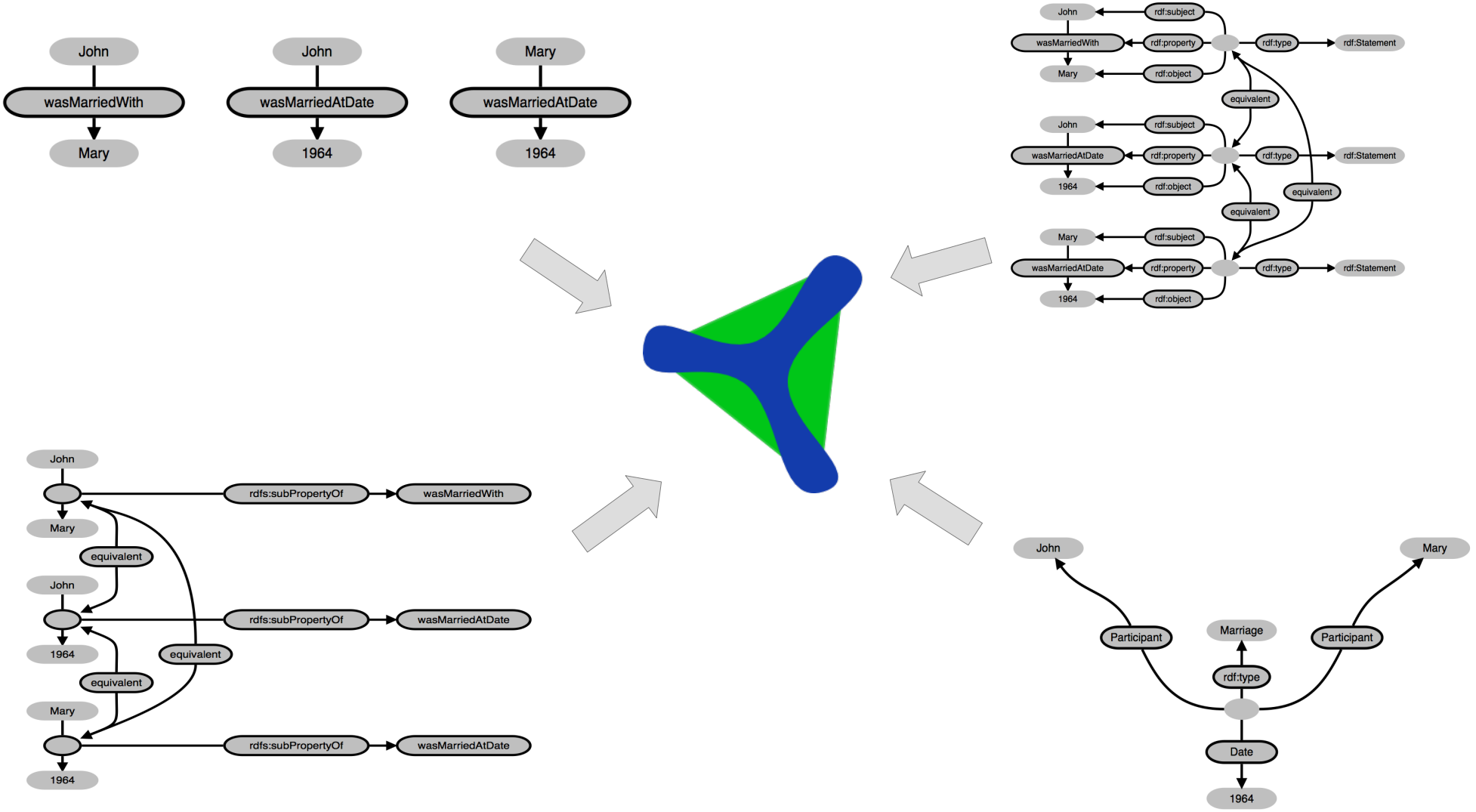


Linking What?

- Linking **data** is not linking **entities**
- Current efforts focus mostly on linking entities one to one

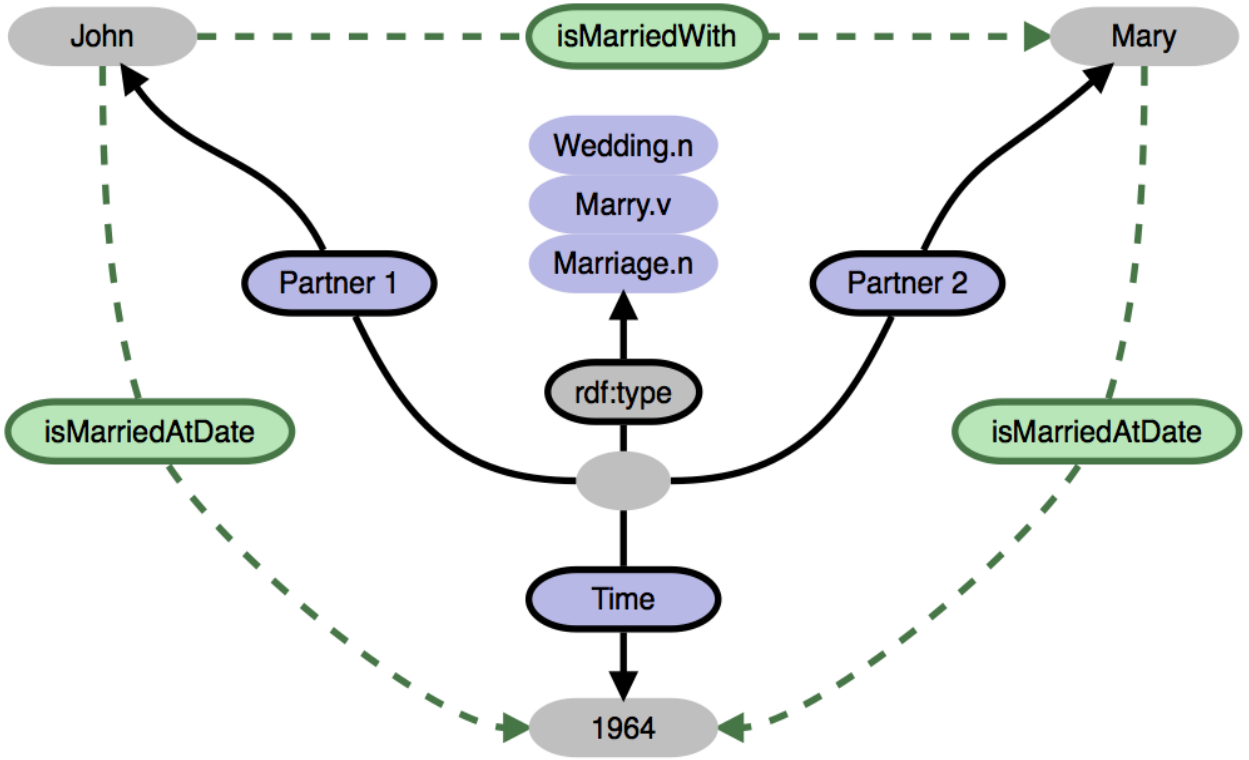
FrameBase

schema



FrameBase: schema

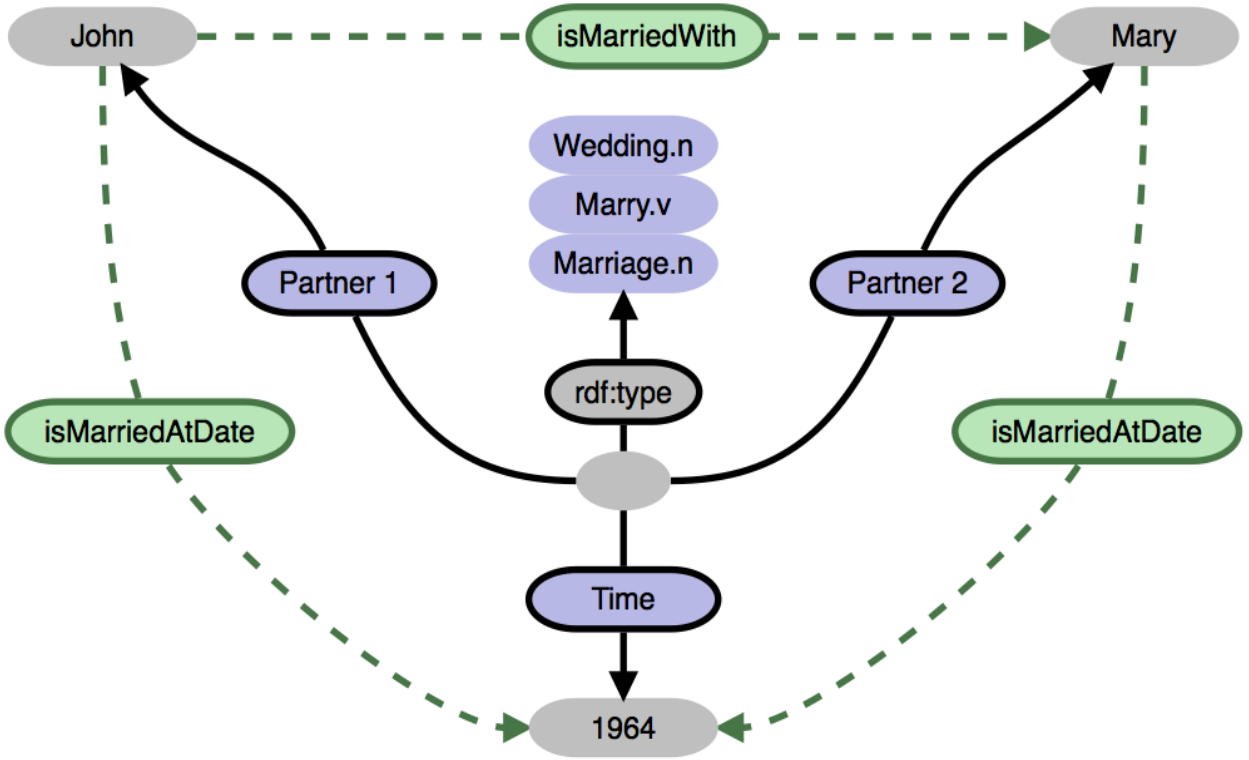
- Core: RDFS schema to represent knowledge using neo-Davidsonian approach with a wide and extensible vocabulary of
  - **Frames**. In a hierarchy. Frames are “events, situations, eventualities…”
  - **Frame Elements**. Outgoing properties representing frame-specific semantic roles



FrameBase

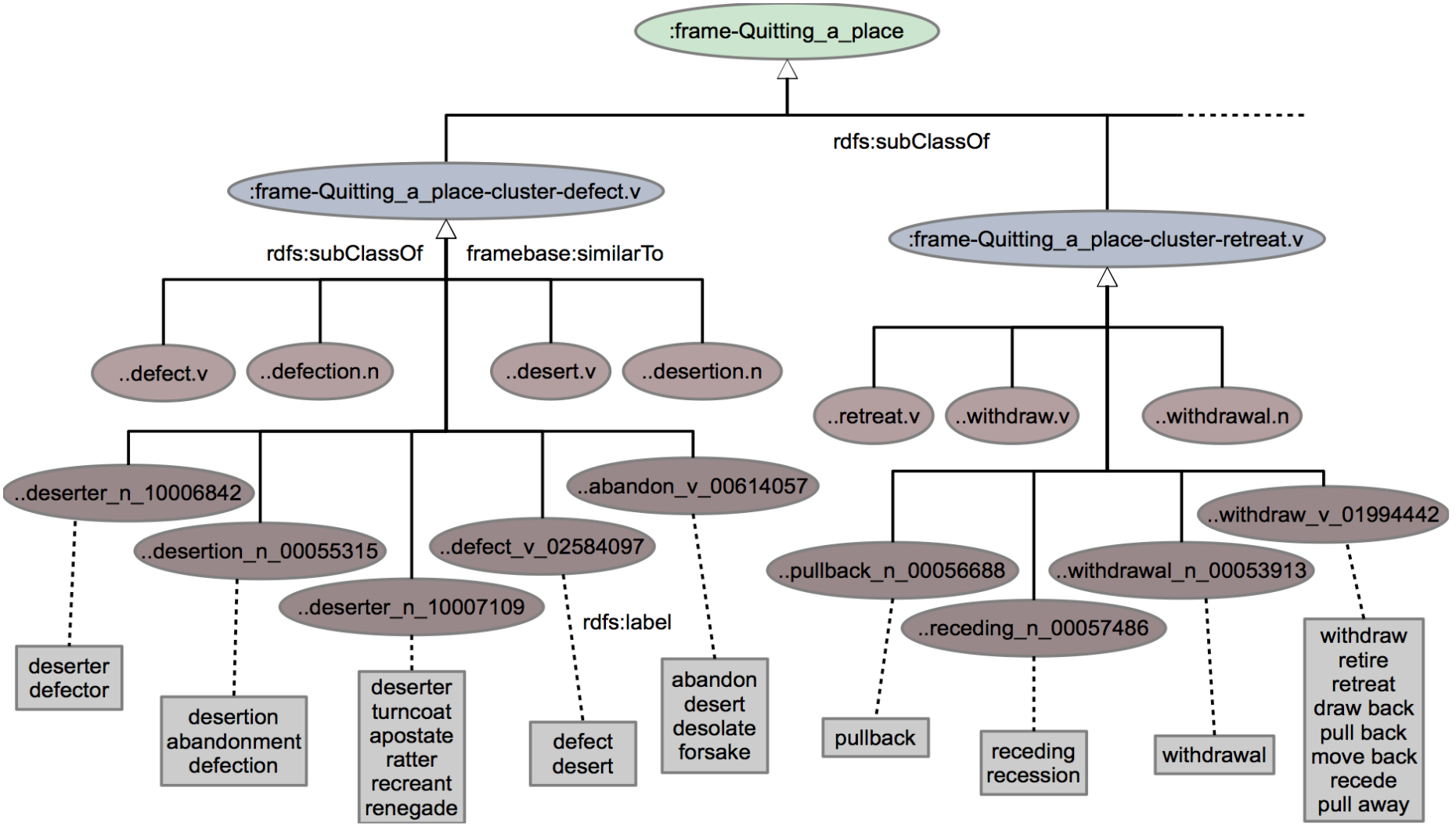
schema

- Vocabulary based on NLP resources (FrameNet+WordNet)
  - This provides connection with natural language and semantic role labeling systems. It clusters near-equivalents.



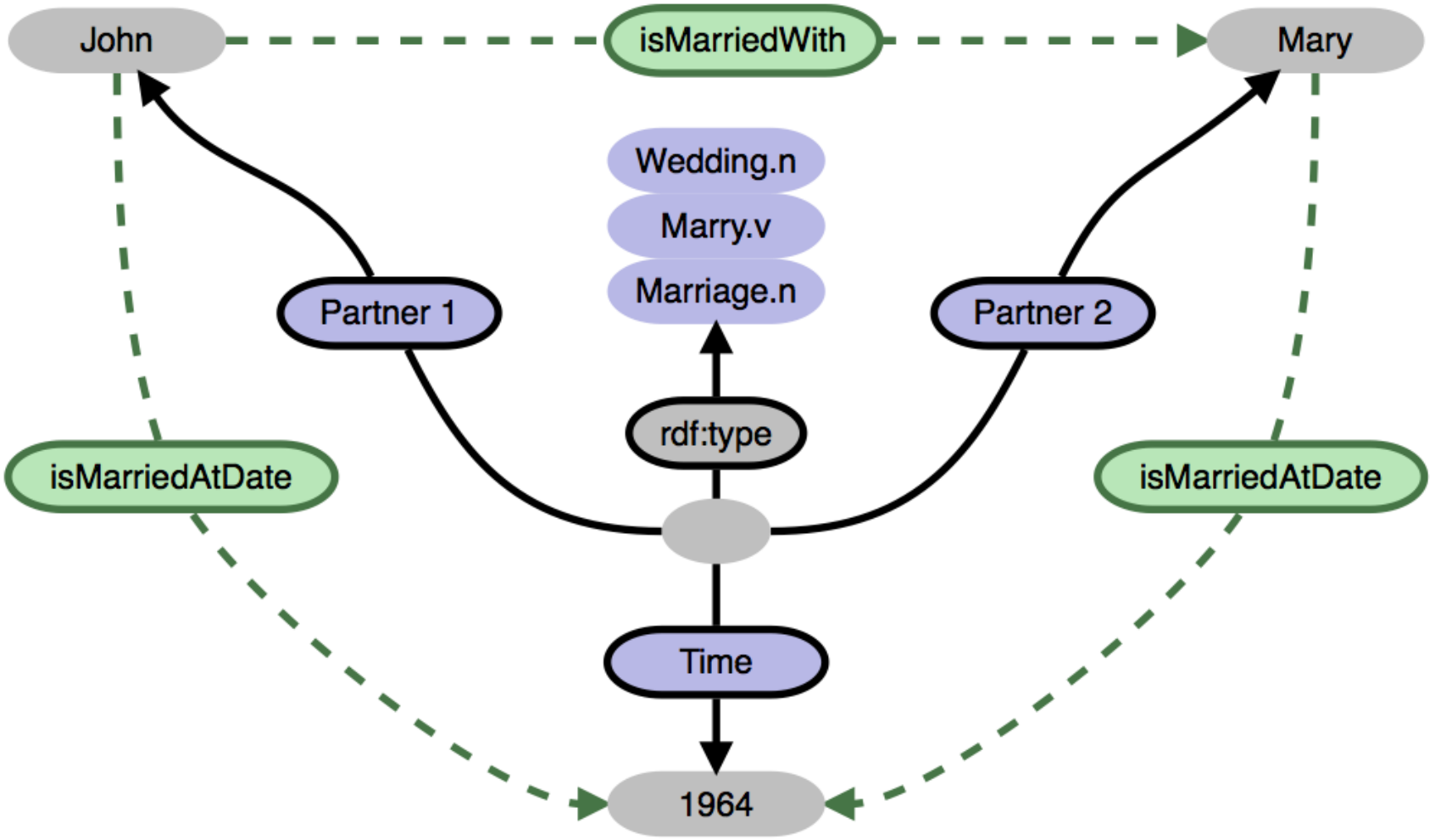
FrameBase

schema



FrameBase

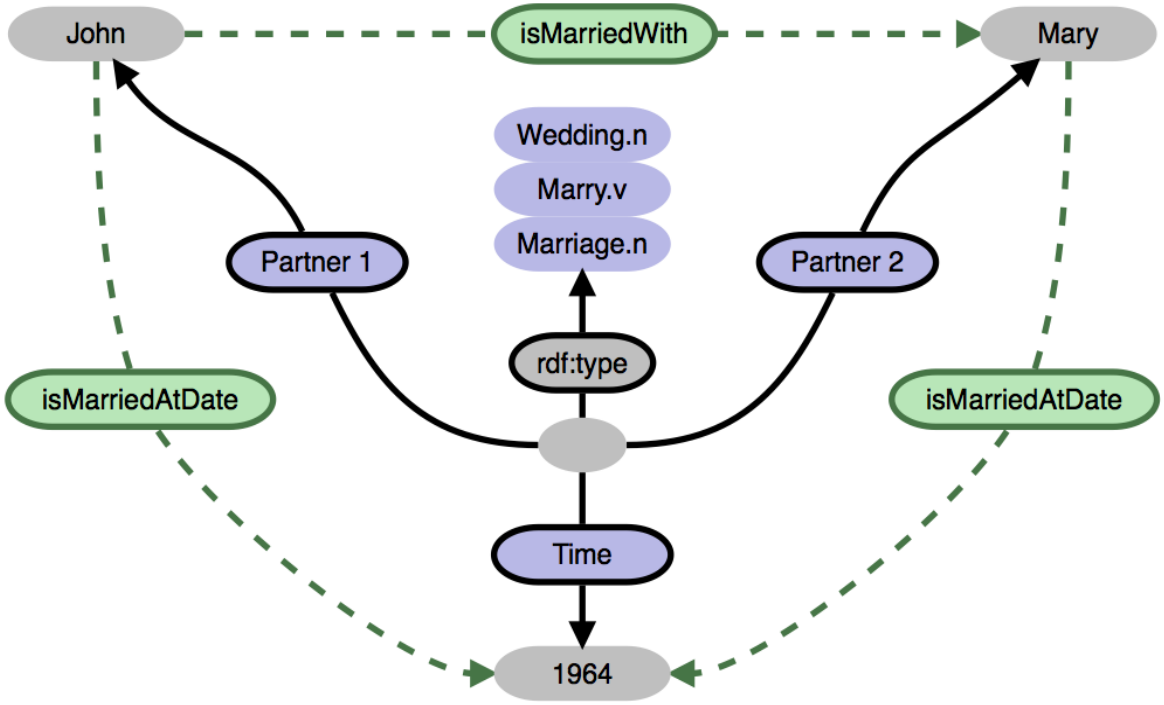
schema



FrameBase: *ReDer* rules

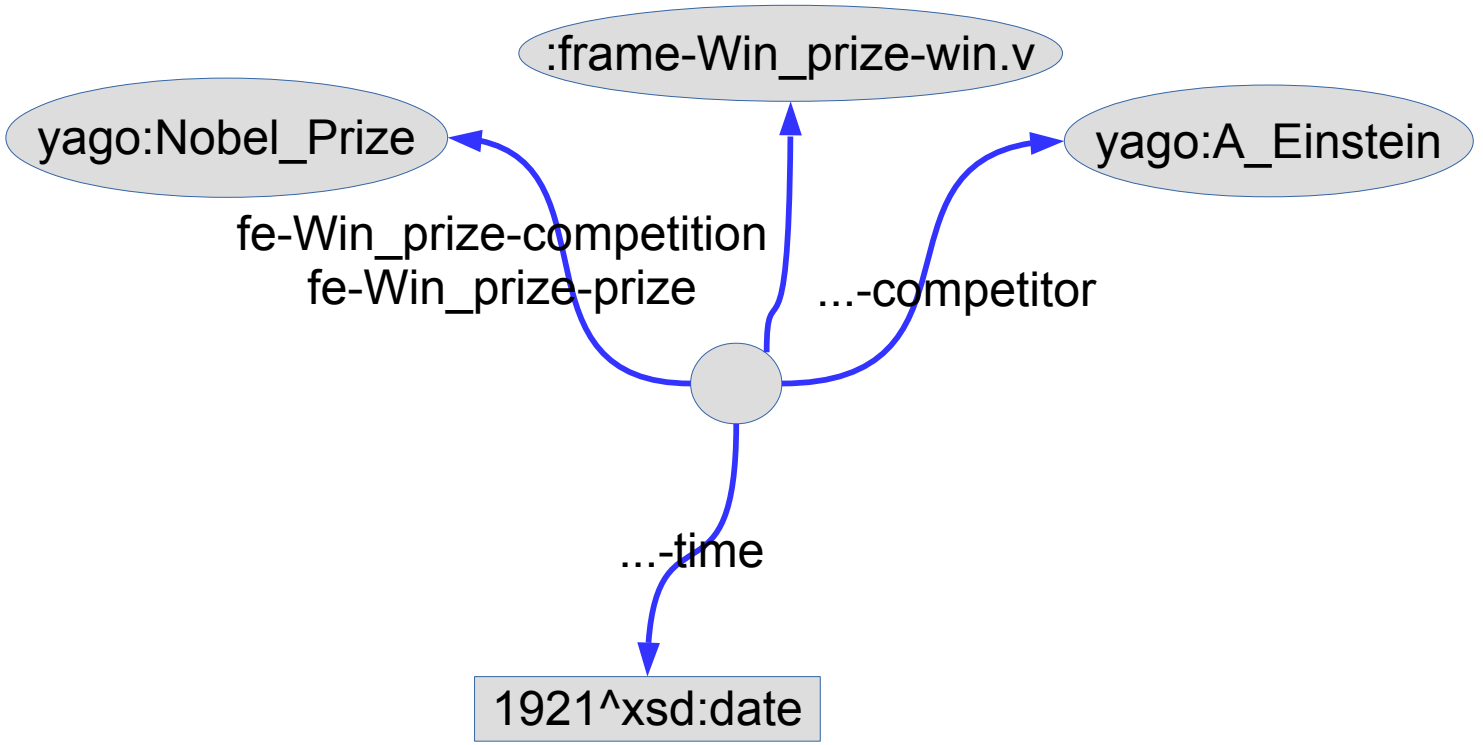
- Two-layered structure:
  - ☞ Create two levels of reification, and Reification-Dereification (**ReDer**) inference rules (horn clauses) that connect them.
    - Reified knowledge using frames and frame elements
    - Dereified knowledge using direct binary predicates (**DBPs**)
- Currently ~15000 rules/DBPs

```
?f a :frame-Separating-partition.v
AND
?f :fe-Separating-Whole ?s
AND
?f :fe-Separating-Parts ?o
IFF
?s ..-isPartitionedIntoParts ?o
```



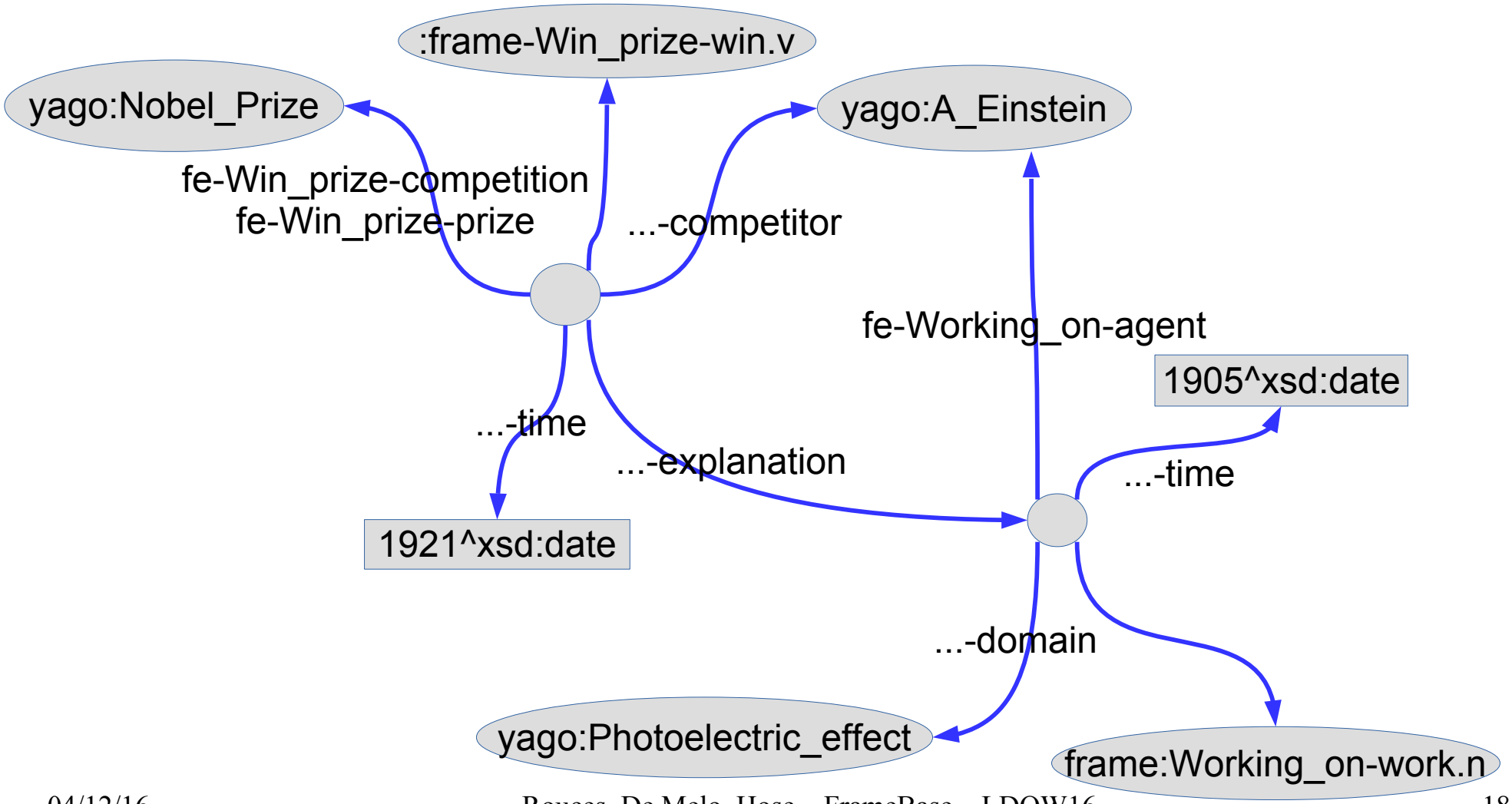
FrameBase

Example



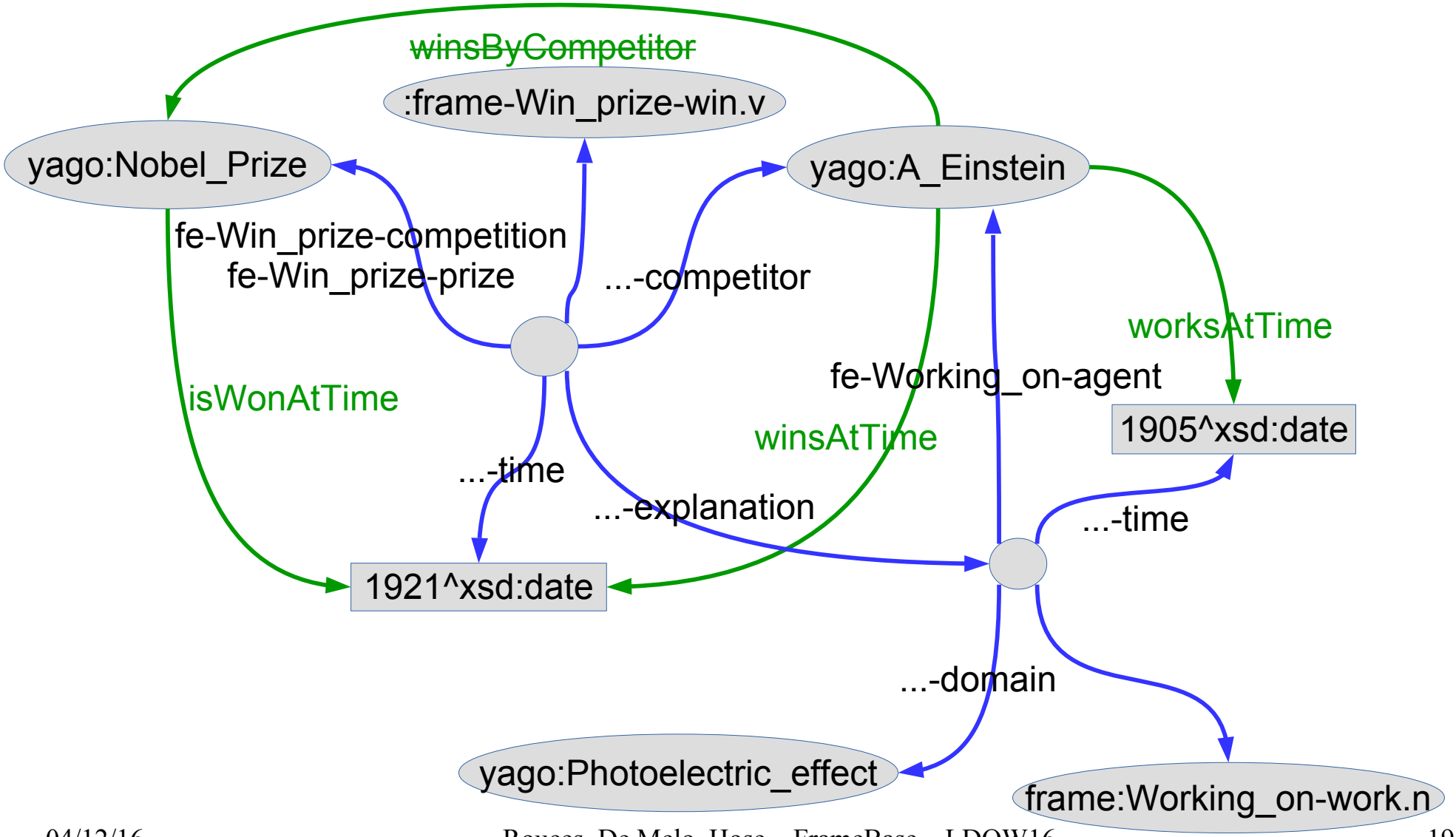
FrameBase

Example



FrameBase

Example



Creation of complex mappings

- Complex mappings between FrameBase and external KBs. Built in three steps:
  1. Creating ReDer rules and DBPs in FrameBase
  2. Canonicalizing predicate names from external Kbs
  3. Matching DBPs with external predicate names

```
?f a :frame-Separating-partition.v
AND ?f :fe-Separating-Whole ?s
AND ?f :fe-Separating-Parts ?o
IFF ?s ..-isPartitionedIntoParts ?o
```

```
somekb:splitInto
→
somekb:isSplitInto
```

```
sim('is split into',
    'is partitioned into parts')
```

```
?f a :frame-Separating-partition.v
AND ?f :fe-Separating-Whole ?s
AND ?f :fe-Separating-Parts ?o
IFF ?s somekb:splitInto ?o
```

Creation of complex mappings

Step 1. Creating DBPs in FrameBase

- In [1], DBPs are created with verbs and nouns as heads. We extend the approach to to deal with adjectives as well.
- We use syntactic annotations from FrameNet

Creation Rule: Copula+Adjective
Create DBP with name "is LU PREP FE-O" *if*
ISADJECTIVE(LU) AND phrase-type-o==PP[PREP] AND grammatical-function-s==Ext AND grammatical-function-o==Dep

```
?s dbp-Sound_level-isLoudToDegree ?o
↕
f type frame-Sound_level-loud.a
f fe-Sound_level-Entity ?s
f fe-Sound_level-Degree ?o
```

[1] J. Rouces, G. de Melo and K. Hose. *FrameBase: Representing N-ary Relations using Semantic* Frames. In: Proc. 12th Extended Semantic Web Conference (ESWC 2015) http://goo.gl/EDomXq

Creation of complex mappings

Step 1. Creating DBPs in FrameBase

- In [1], DBPs are created with verbs and nouns as heads. We extend the approach to to deal with adjectives as well.
- We use syntactic annotations from FrameNet

Creation Rule: Copula+Adjective
Create DBP with name "is LU PREP FE-O" *if*
ISADJECTIVE(LU) AND phrase-type-o==PP[PREP] AND grammatical-function-s==Ext AND grammatical-function-o==Dep

```
?s dbp-Sound_level-becomesLoudToDegree ?o
↕
f type frame-Sound_level-loud.a
f fe-Sound_level-Entity ?s
f fe-Sound_level-Degree ?o
f' type frame-Becoming-become.v
f' fe-Becoming-Entity ?s
f' fe-Becoming-Final_state f
```

```
?s dbp-Sound_level-seemsLoudToDegree ?o
↕
f type frame-Sound_level-loud.a
f fe-Sound_level-Entity ?s
f fe-Sound_level-Degree ?o
f' type frame-Appearance-seem.v
f' fe-Appearance-Phenomenon ?s
f' fe-Appearance-Inference f
```

Creation of complex mappings

Step 2. Canonicalizing predicate names from external Kbs

Apply a set of rules for name transformations:

- If the name p of a property is a past participle, it can be extended with the prefix “is” (without postfix “of”). *Ex: “created” → “**is** created”*
- If the name p of a property is a noun or a noun phrase, and a range is declared for the property, let X be a set containing p’s name and the hypernyms of all its word senses (obtained from WordNet). If for any element x in X, p is a substring of x or x is a substring of p, then p can be extended with the prefix “has”. *Ex: “creator” with range ”person” → “**has** creator”*
- The same rule as above, but using the domain instead of the range, which allows p to be extended with the prefix “is” and postfix “of”. *Ex: “creator” with domain “person” → “**is** creator **of**”*

Creation of complex mappings

Step 2. Canonicalizing predicate names from external Kbs

- If the property is symmetric, we can introduce extensions both with “has” and with “is”+ . . . +“of”. *Ex: “sibling” is owl:symmetric → “has sibling”, “is sibling of”,*
- For every property p corresponding to the pattern “is X of”, an inverse property can be created of the form “has X”. *Ex: “is mentor of” → ^“has mentor”*
- For every property p corresponding to the pattern “has X”, an inverse property can be created of the form “is X of”. *Ex: “has mentor” → ^“is mentor of”*

Creation of complex mappings

Step 3. Matching DBPs with external predicate names

For each canonicalized Source Dataset Property (SDP), maximize over all DBPs:

bag of words labels

bag of words domain, range, etc

"core" properties in reified pattern

$$w_1 \cos(v_1^{\text{SDP}}, v_1^{\text{DBP}}) + w_2 \cos(v_2^{\text{SDP}}, v_2^{\text{DBP}}) + w_3 c_1 + w_4 c_2$$

0.7 0.1 0.1 0.1

- 0.1's for resolving ties
- Difficult to use supervised ML: very low IA agreement for gold standards

Creation of complex mappings
Results

- Canonicalized properties from source KB (DBpedia)
  - Examples:

Source property	**Canonicalized**
currently run by	*is currently run by*
golden raspberry award	*has golden raspberry award*
statistic	*is statistic of*
link title	*has link title*
first leader	*has first leader*

Precision: 85%

Creation of complex mappings
Results

- Integration rules (DBpedia)
  - Examples:

Precision: 79%

```
CONSTRUCT {
 _:r a :frame-Education_teaching-school.v .
 _:r :fe-Education_teaching-Student ?S .
 _:r :fe-Education_teaching-Skill ?O .
} WHERE {
 ?S <http://dbpedia.org/property/schooledAt> ?O .
}
```

```
CONSTRUCT {
 _:r a :frame-Appearance-smell.v .
 _:r :fe-Appearance-Phenomenon ?S .
 _:r :fe-Appearance-Characterization ?O .
} WHERE {
 ?S <http://dbpedia.org/property/smellsLike> ?O .
}
```

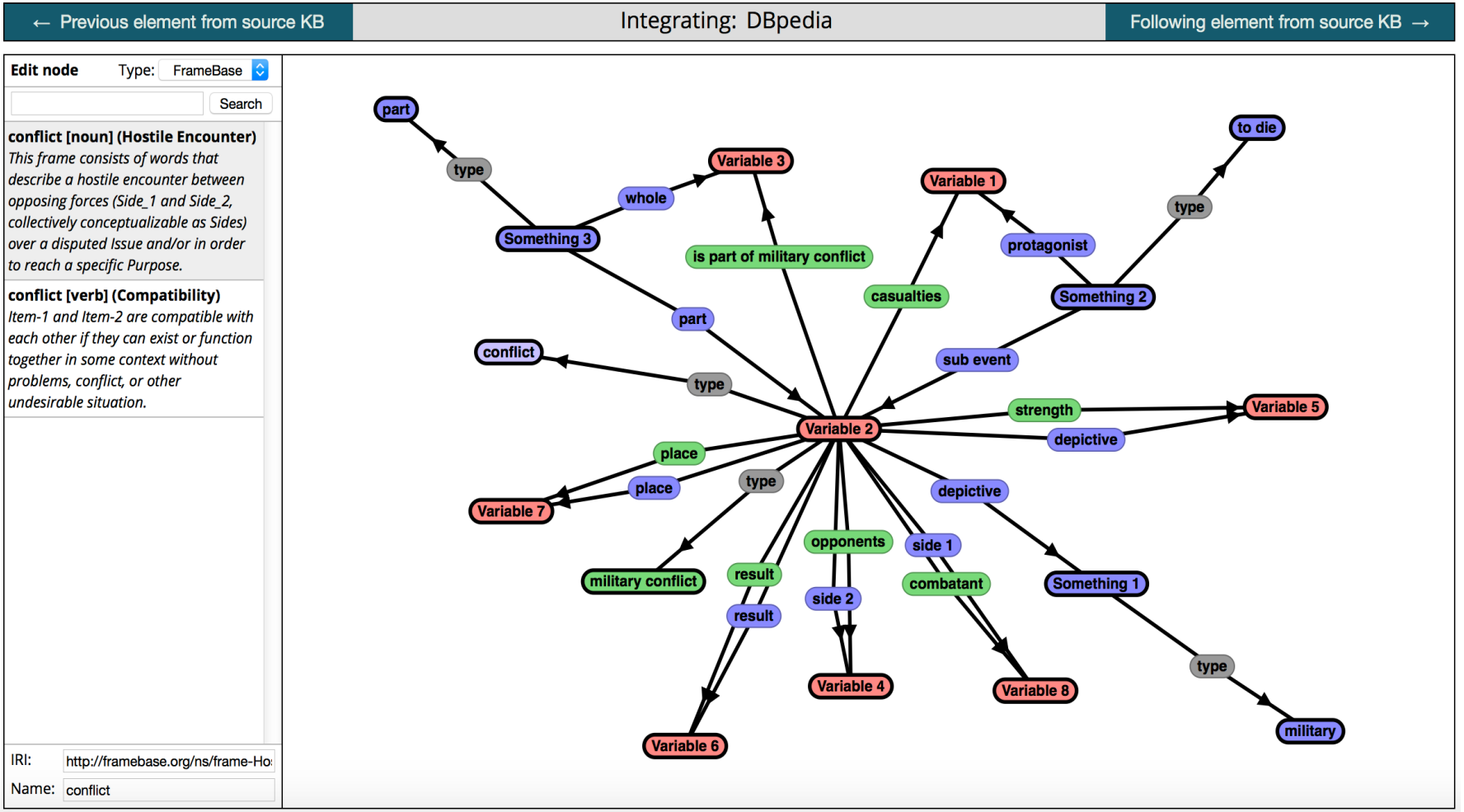
```
CONSTRUCT {
 _:r a :frame-Residence-reside.v .
 _:r :fe-Residence-Resident ?S .
 _:r :fe-Residence-Location ?O .
} WHERE {
 ?S <http://dbpedia.org/property/residesIn> ?O .
}
```

Conclusion & Future Work

- We create complex mappings between properties in external KBs and “reified” property-frame-property patterns in FrameBase.
- Future work:
  - Combining with traditional one-to-one mappers (class-class, property-property)
    - This produces transitive complex maps between arbitrary external KBs
  - More very-complex maps
    - (becomes/seems Adj → Noun → Verb)

Conclusion & Future Work

- Web interface for semi-automatic integration (IJCAI 16 demo)



Questions

More information at http://framebase.org