On Blank Nodes

A. Mallea @janoma M. Arenas A. Hogan A. Polleres @axelpolleres

School of Engineering — PUC Chile
DERI Galway — National University of Ireland

October 2011

How are blank nodes defined?

Blank nodes are defined in RDF

How are blank nodes defined?

Blank nodes are defined in RDF, used in:

- SPARQL
- OWL
- RDB2RDF
- RIF

How are blank nodes defined?

Blank nodes are defined in RDF, used in:

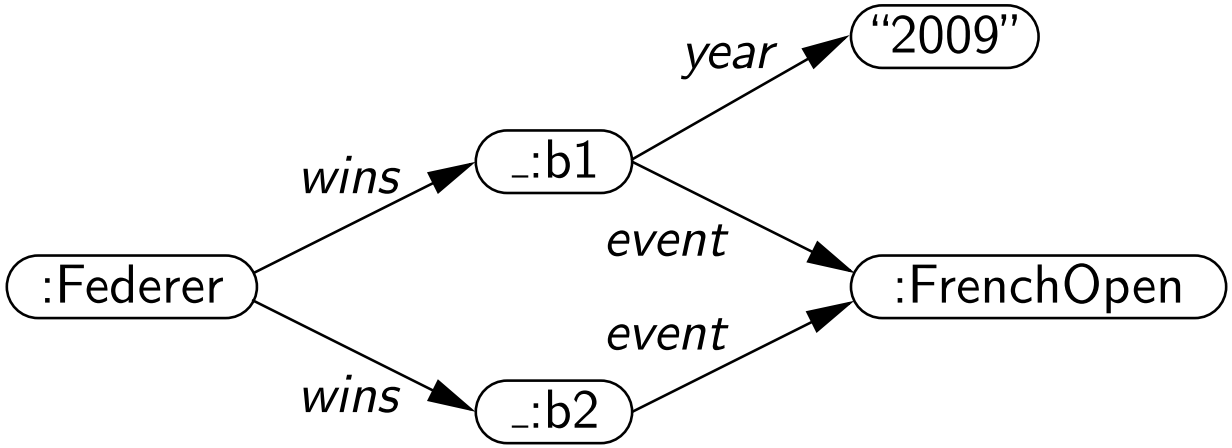
- SPARQL
- OWL
- RDB2RDF
- RIF
- data

How are blank nodes defined?

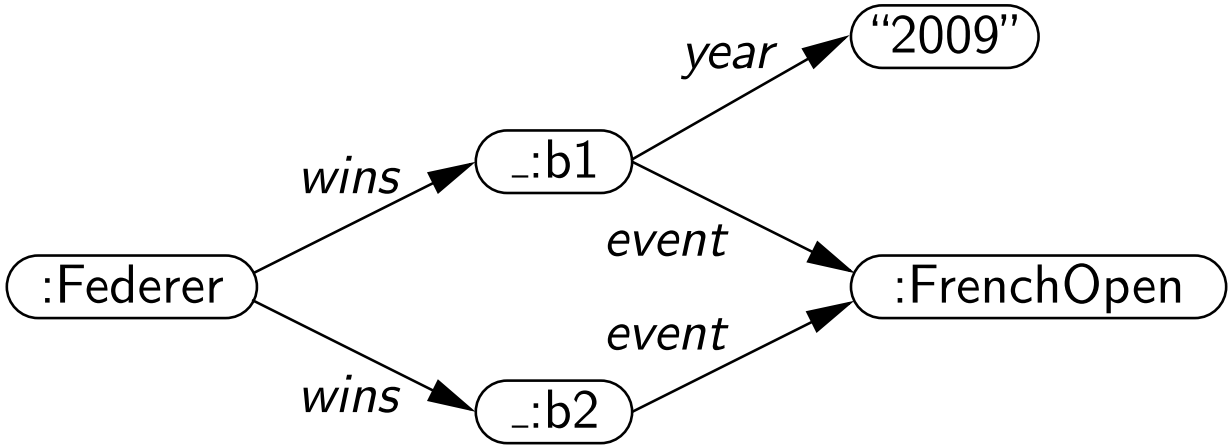
Blank nodes are defined in RDF, used in:

- SPARQL
- OWL
- RDB2RDF
- RIF
- data

How many times has Federer won the French Open?

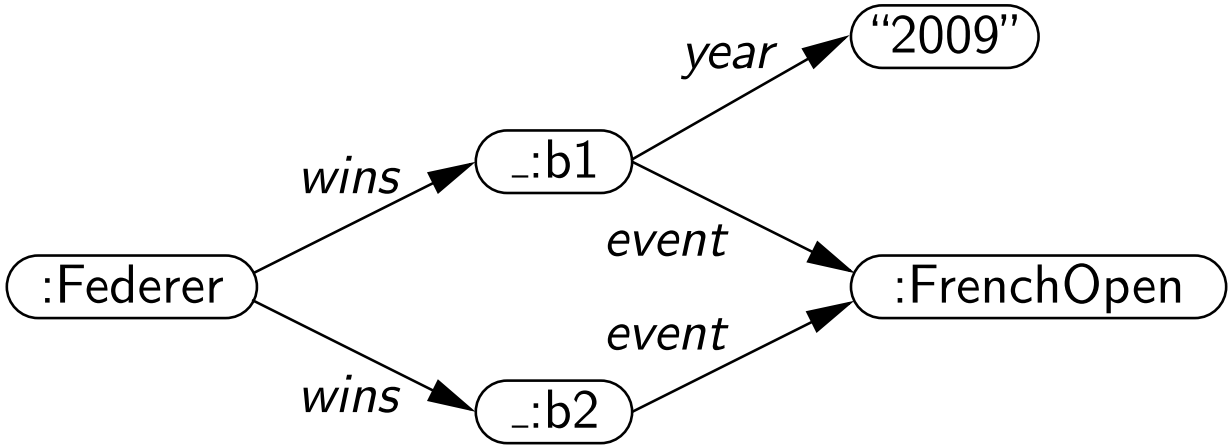


How many times has Federer won the French Open?




```
SELECT ?FO
WHERE {
  :Federer wins ?FO .
  ?FO event :FrenchOpen .
}
```

How many times has Federer won the French Open?




```
SELECT (COUNT(?FO) AS ?total)
WHERE {
  :Federer wins ?FO .
  ?FO event :FrenchOpen .
}
```

In RDF, blank nodes are anonymous existential variables

Blank nodes don't have names.

```
:Vader :fatherOf [ :name "Luke" ] , [ :name "Leia" ].
```

In RDF, blank nodes are anonymous existential variables

Blank nodes don't have names.

```
:Vader :fatherOf [ :name "Luke" ] , [ :name "Leia" ].
```

Labels are used in (some) serializations.

```
:Vader :fatherOf _:b1 .
:Vader :fatherOf _:b2 .
_:b1 :name "Luke" .
_:b2 :name "Leia" .
```

Careful with the terminology: names, labels and the like

In different normative documents, the following words are mixed in several ways:

- Name
- Label
- Identifier

In RDF, blank nodes are anonymous existential variables

Blank nodes indicate the existence of a thing.

?

```
:John :telephone _:b1 .
```

In RDF, blank nodes are anonymous existential variables

Blank nodes indicate the existence of a thing.

"John has a telephone number."

```
:John :telephone _:b1 .
```

In RDF, blank nodes are anonymous existential variables

Two blank nodes *could* represent the same thing.

?

```
:John :telephone _:b1 .
:John :telephone _:b2 .
```

In RDF, blank nodes are anonymous existential variables

Two blank nodes *could* represent the same thing.

"John has a telephone number."

```
:John :telephone _:b1 .
:John :telephone _:b2 .
```

In RDF, blank nodes are anonymous existential variables

A blank node is valid in a limited context.

Graph G_1

```
:Vader :fatherOf _:b1 .
```

Graph G_2

```
:John :telephone _:b1 .
```

In RDF, blank nodes are anonymous existential variables

A blank node is valid in a limited context.

Graph G_1

```
:Vader :fatherOf _:b1 .
```

Graph G_2

```
:John :telephone _:b1 .
```

- Not the same blank node!

In RDF, blank nodes are anonymous existential variables

A blank node is valid in a limited context.

Graph G_1

```
:Vader :fatherOf _:b1 .
```

Graph G_2

```
:John :telephone _:b1 .
```

- Not the same blank node!

Merge of G_1 and G_2

```
:Vader :fatherOf _:x .
:John :telephone _:y .
```

In RDF, blank nodes are anonymous existential variables

A blank node is valid in a limited context.

Graph G_1

```
:Vader :fatherOf _:b1 .
```

Graph G_2

```
:John :telephone _:b1 .
```

- Not the same blank node!

Merge of G_1 and G_2

```
:Vader :fatherOf [] .
:John :telephone [] .
```

Blank nodes are not used as NULLs

An SQL table

name	telephone
John	888-8888
Jane	NULL

Is this graph equivalent, *in some sense*?

```
_:b1 :name "John" ;
     :telephone "888-8888" .
_:b2 :name "Jane" ;
     :telephone _:b3 .
```

Blank nodes are not used as NULLs

An SQL table

name	telephone
John	888-8888
Jane	NULL

Is this graph equivalent, *in some sense*?

```
_:b1 :name "John" ;
     :telephone "888-8888" .
_:b2 :name "Jane" ;
     :telephone _:b3 .
```

Blank nodes are not used as NULLs

An SQL table

name	telephone
John	888-8888
Jane	NULL

Is this graph equivalent, *in some sense*?

```
_:b1 :name "John" ;
     :telephone "888-8888" .
_:b2 :name "Jane" .
     :telephone _:b3 . *
```

* Recently discussed by the RDB2RDF Working Group.

Blank nodes are not the same as NULLs

What about RDF2RDB?

An RDF graph

```
_:b1 :name "John" ;
     :telephone "888-8888" .
_:b2 :name "Jane" ;
     :telephone _:b3 .
```

Blank nodes are not the same as NULLs

What about RDF2RDB?

An RDF graph

```
_:b1 :name "John" ;
     :telephone "888-8888" .
_:b2 :name "Jane" ;
     :telephone _:b3 .
```

Is this an equivalent SQL table?

name	telephone
John	888-8888
Jane	NULL

Blank nodes are not the same as NULLs

What about RDF2RDB?

An RDF graph

```
_:b1 :name "John" ;
     :telephone "888-8888" .
_:b2 :name "Jane" ;
     :telephone _:b3 .
```

Is this an equivalent SQL table? No!

name	telephone
John	888-8888
Jane	NULL

In practice, blank nodes are used as constants. . .

. . . *different* constants.

"John has two telephone numbers"

```
:John :telephone _:b1 .
:John :telephone _:b2 .
```

In practice, blank nodes are used as constants. . .

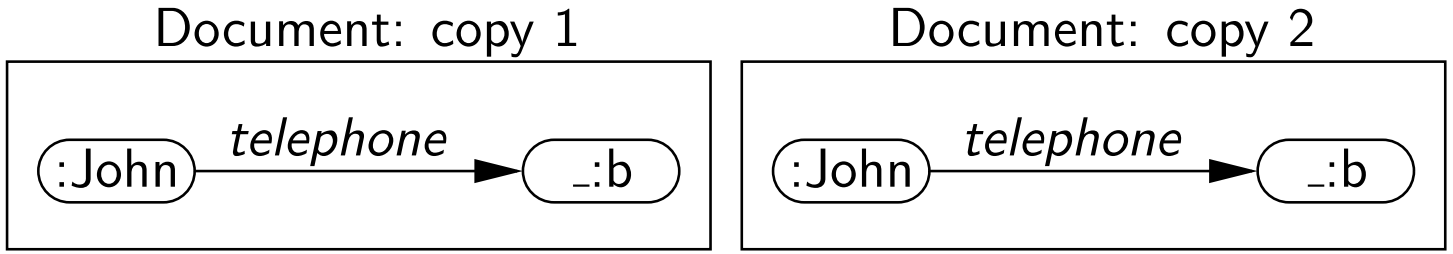
. . . *different* constants.

"John has two telephone numbers"

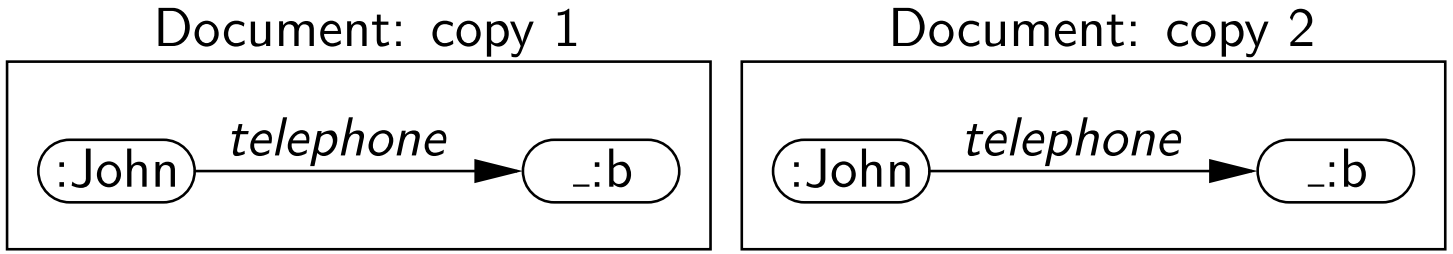
```
:John :telephone _:b1 .
:John :telephone _:b2 .
```

Not aligned with existential.

In practice, blank nodes are used as constants. . .



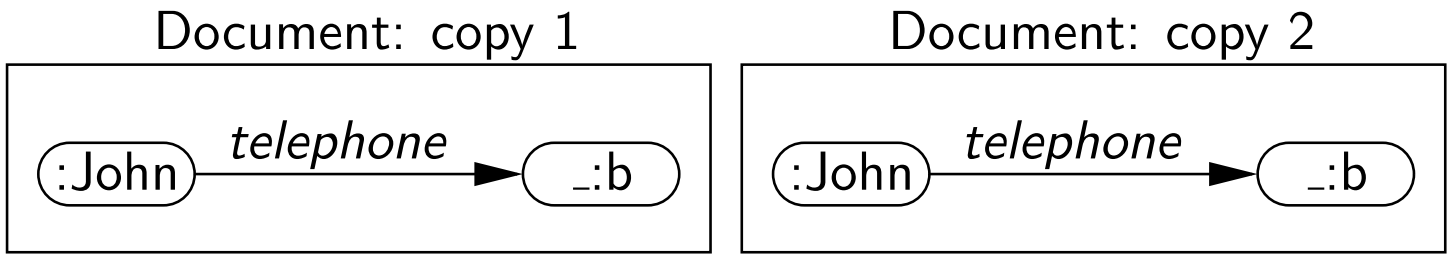
In practice, blank nodes are used as constants. . .



Merge of the two copies:

```
:John :telephone _:b1 .
:John :telephone _:b2 .
```

In practice, blank nodes are used as constants. . .



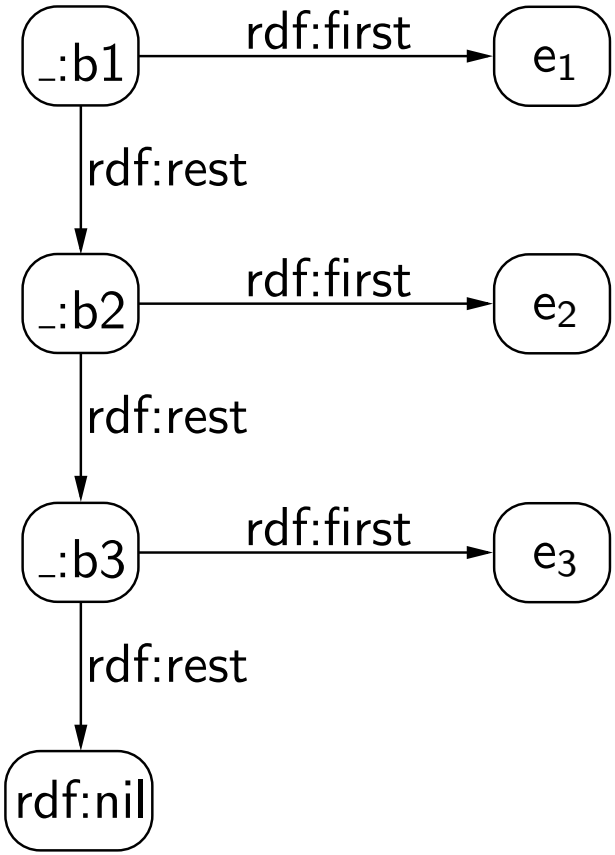
Merge of the two copies:

```
:John :telephone _:b1 .
:John :telephone _:b2 .
```

- Still two telephone numbers?

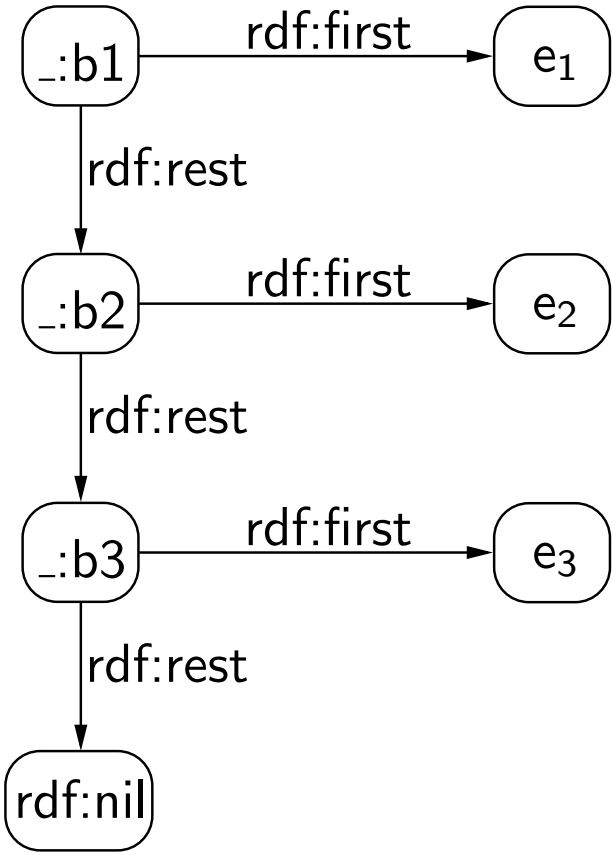
. . . as local/structural constructs in data structures. . .

. . . a.k.a. *syntactic sugar*.



. . . as local/structural constructs in data structures. . .

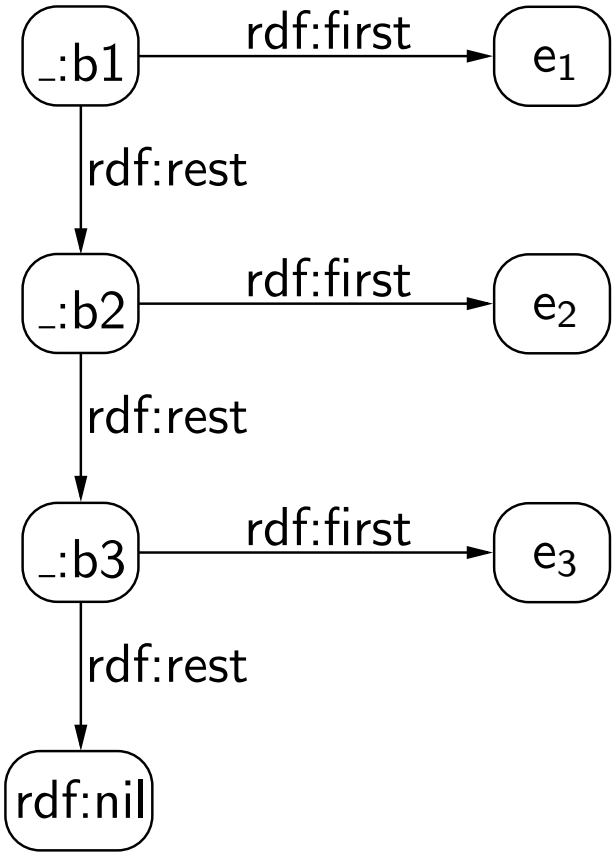
. . . a.k.a. *syntactic sugar*.

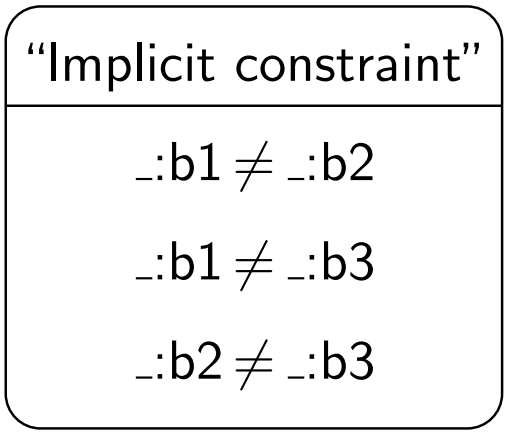


Existential does not work here.

. . . as local/structural constructs in data structures. . .

. . . a.k.a. *syntactic sugar*.





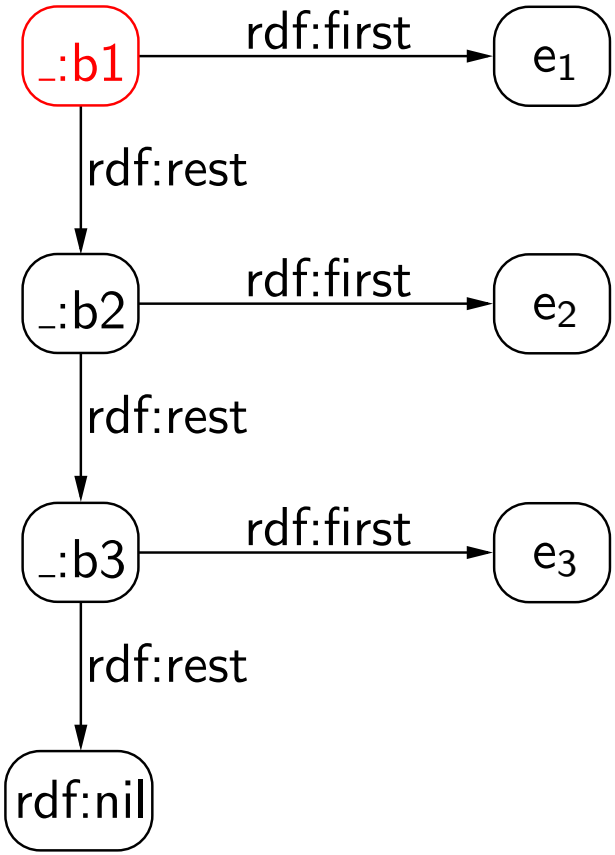
X

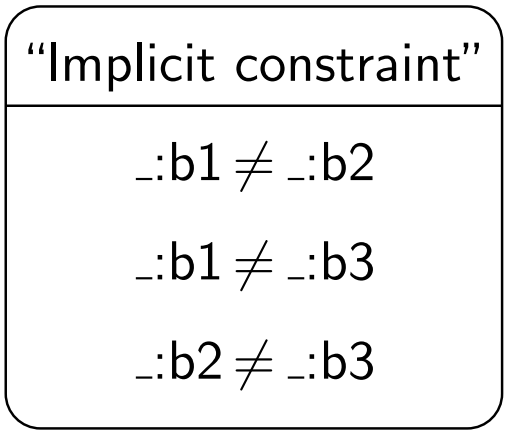
Existential does not work here.

These blank nodes are meant to be different.

. . . as local/structural constructs in data structures. . .

. . . a.k.a. *syntactic sugar*.





X

✓

Existential does not work here.

These blank nodes are meant to be different.

. . . as default names

. . . instead of URIs that I *should* generate:

`http://i.am.too.lazy/to/generate/a/URI`
`http://i.dont.want/to/generate/a/URI`

. . . as default names

. . . instead of URIs that I *should* generate:

~~`http://i.am.too.lazy/to/generate/a/URI`~~
~~`http://i.dont.want/to/generate/a/URI`~~

`_:faster`
`_:easier`

. . . as default names

. . . instead of URIs that I *should* generate:

~~`http://i.am.too.lazy/to/generate/a/URI`~~
~~`http://i.dont.want/to/generate/a/URI`~~

`_:faster`
`_:easier`

Meaning depends on the publisher's intention (if any).

Not all the normative uses are common in practice

We already asked:

- Informal poll sent to sem-web and LOD mailing lists.
- $\sim$90 responses (thanks!).
- `http://bit.ly/bnodes`

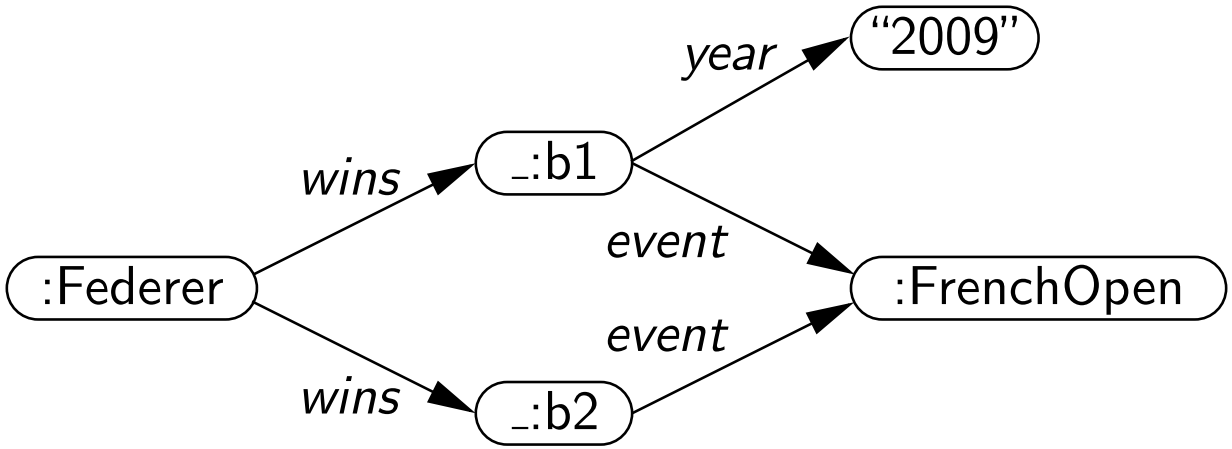
Not all the normative uses are common in practice

We already asked:

- Informal poll sent to sem-web and LOD mailing lists.
- ∼90 responses (thanks!).
- `http://bit.ly/bnodes`

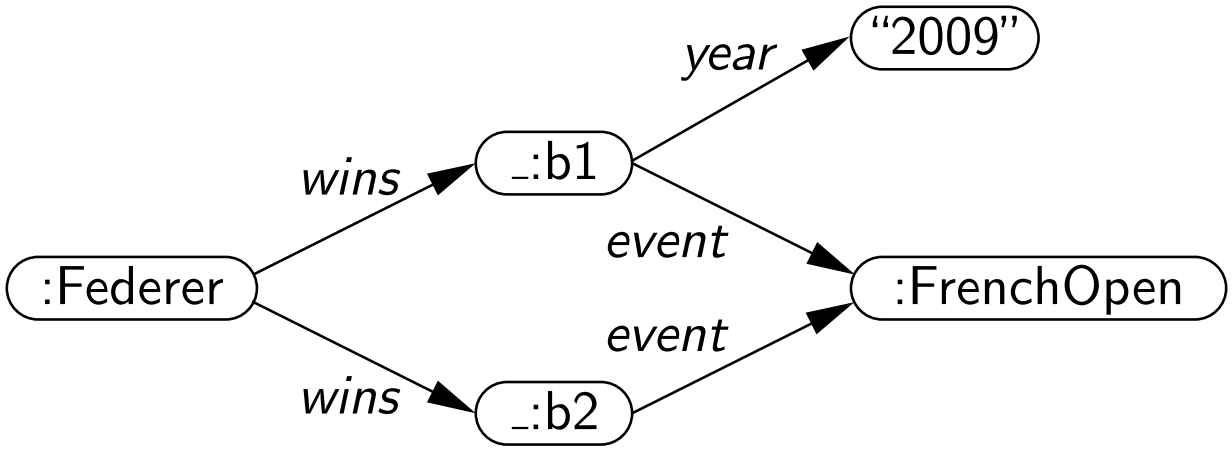
One clear conclusion: blank nodes are divisive.

SPARQL considers blank nodes as different constants




```
SELECT ?FO WHERE {
  :Federer wins ?FO .
  ?FO event :FrenchOpen .
}
```

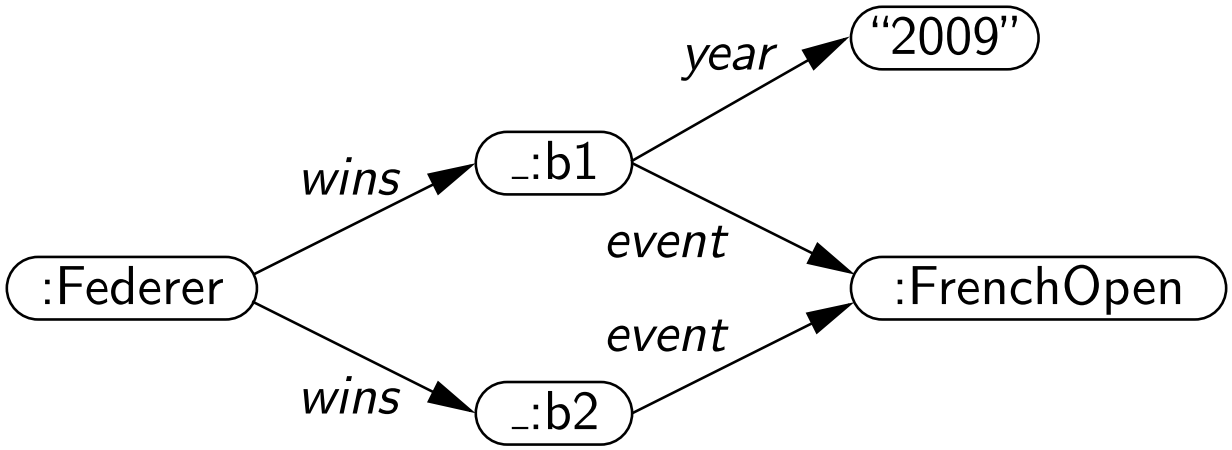
SPARQL considers blank nodes as different constants




```
SELECT ?FO WHERE {
  :Federer wins ?FO .
  ?FO event :FrenchOpen .
}
```

?FO
_:b1
_:b2

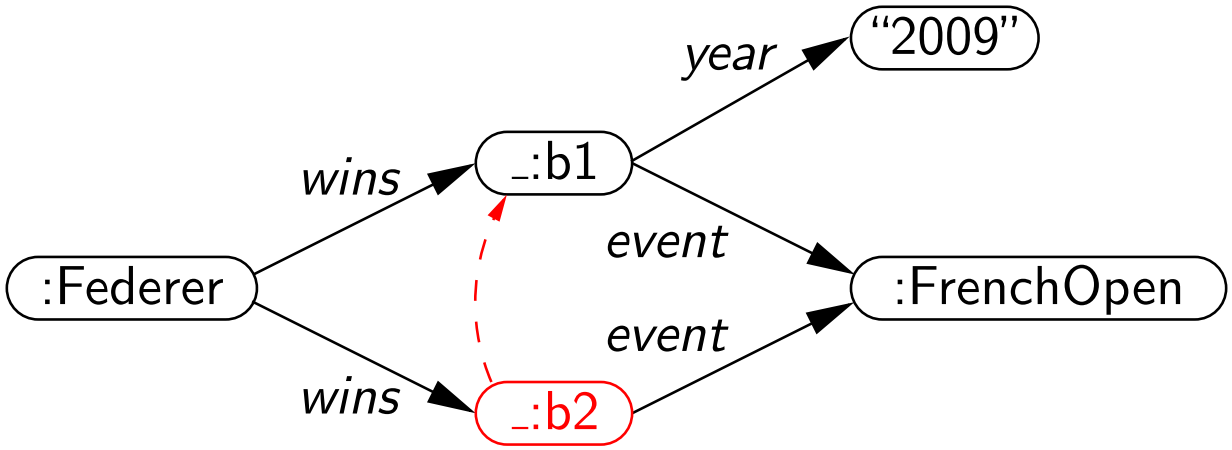
SPARQL considers blank nodes as different constants




```
SELECT DISTINCT ?FO WHERE {
   :Federer wins ?FO .
   ?FO event :FrenchOpen .
}
```

?FO
_:b1
_:b2

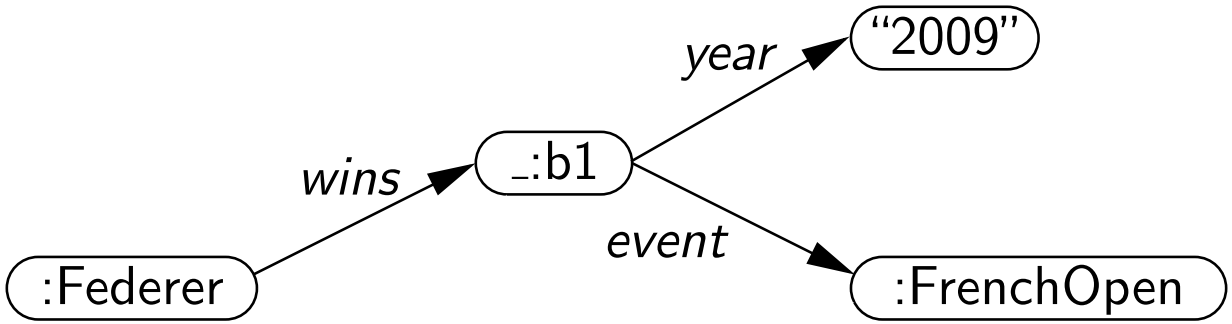
SPARQL considers blank nodes as different constants




```
SELECT DISTINCT ?FO WHERE {
   :Federer wins ?FO .
   ?FO event :FrenchOpen .
}
```

?FO
_:b1
_:b2

SPARQL considers blank nodes as different constants

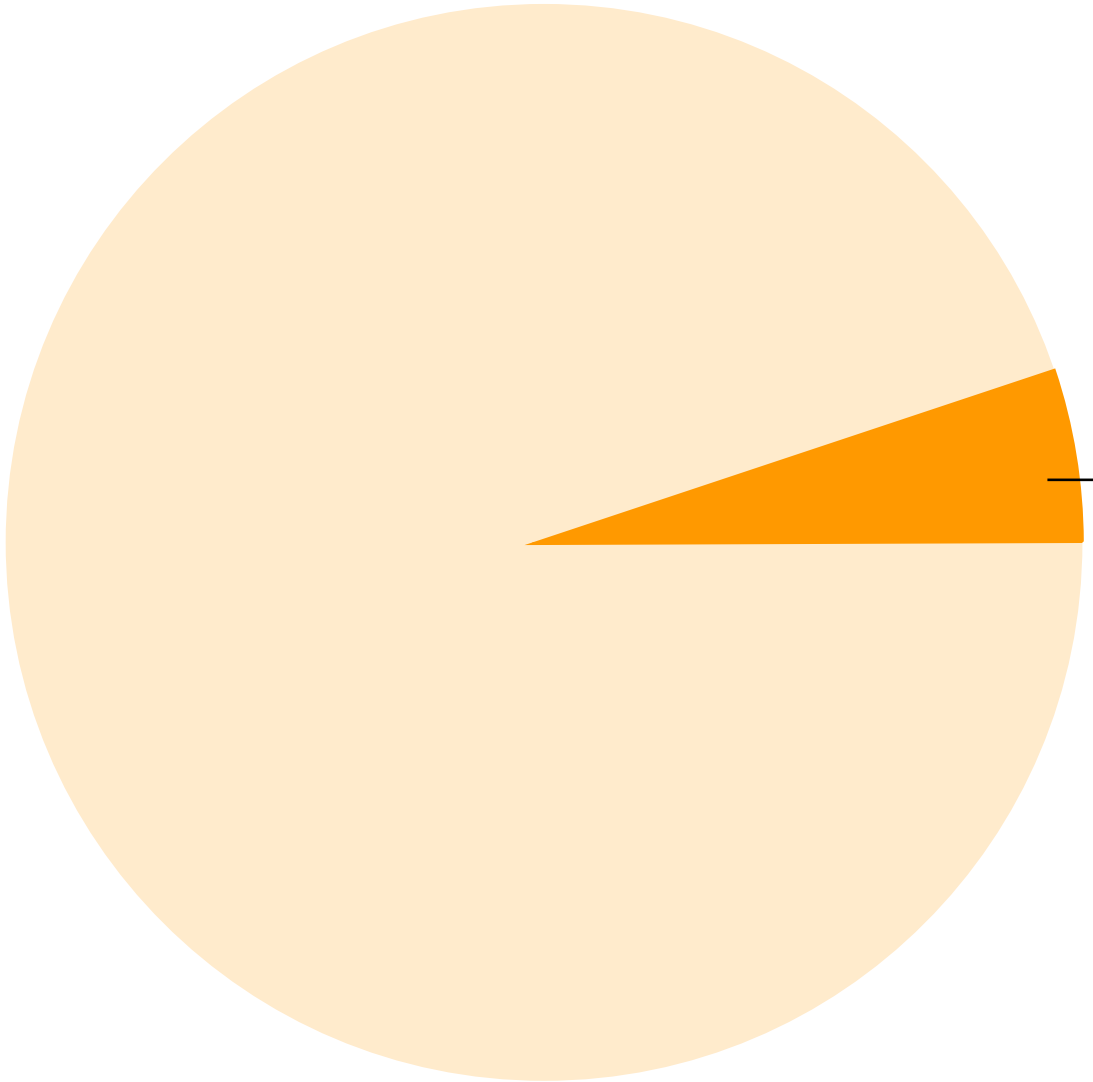



```
SELECT DISTINCT ?FO WHERE {
   :Federer wins ?FO .
   ?FO event :FrenchOpen .
}
```

?FO
_:b1

Blank nodes in The Wild: size

From the corpus of study in Aidan Hogan's PhD thesis:



783 domains

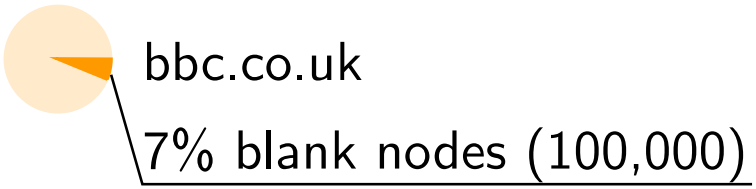
4M documents

965M triples

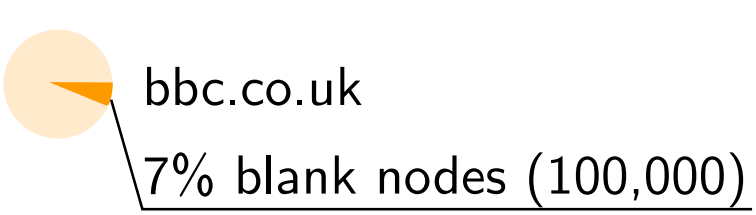
avg. 7.5% bnodes per domain

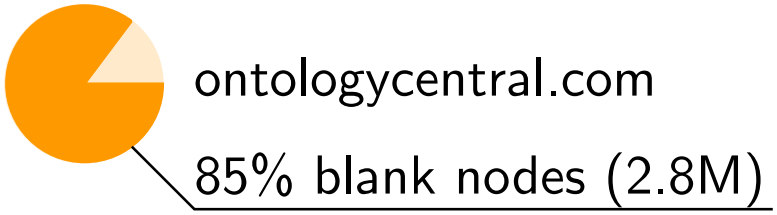
44% of domains did not publish any blank nodes

Blank nodes in The Wild: size

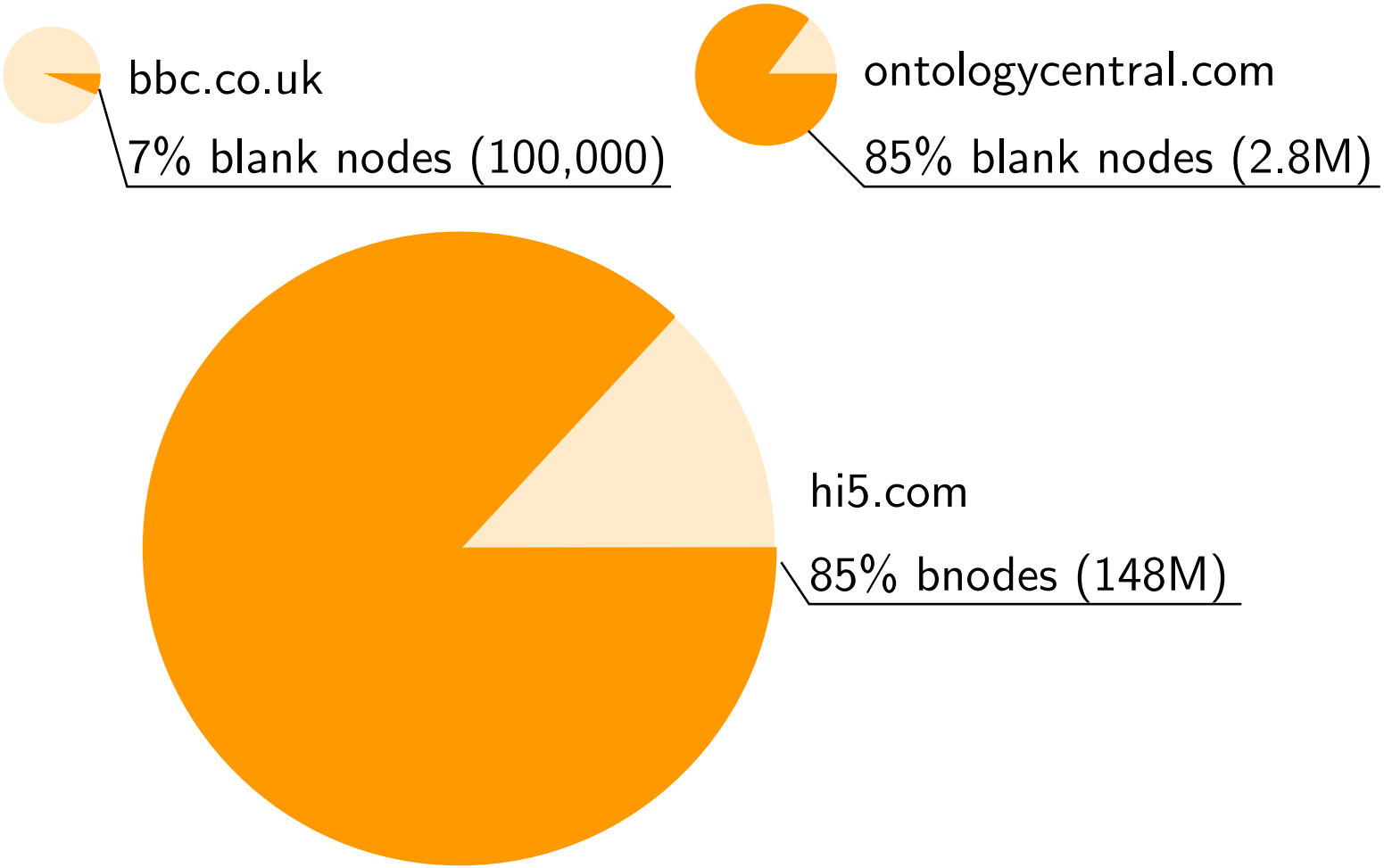


Blank nodes in The Wild: size



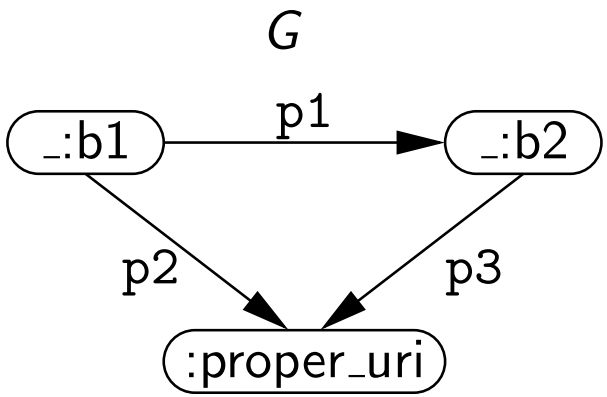


Blank nodes in The Wild: size



Blank nodes: structure

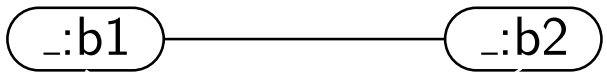
For an RDF graph G, blank(G) is the largest subgraph of G consisting only of blank nodes, seen as an undirected graph.



Blank nodes: structure

For an RDF graph G, blank(G) is the largest subgraph of G consisting only of blank nodes, seen as an undirected graph.

blank(G)



Blank nodes: structure

Treewidth is a measure of "cyclicity" of a graph.

Blank nodes: structure

Treewidth is a measure of "cyclicity" of a graph.

Why do we need this notion?

Blank nodes: structure

Treewidth is a measure of "cyclicity" of a graph.

Why do we need this notion?

- The higher the treewidth of blank(G), the harder entailment becomes [Pichler et al. 2008].

Blank nodes: structure

Treewidth is a measure of "cyclicity" of a graph.

Why do we need this notion?

- The higher the treewidth of blank(G), the harder entailment becomes [Pichler et al. 2008].
- Polynomial algorithm for *bounded treewidth:* $O(n^k)$, with k proportional to the treewidth of blank(G)

Blank nodes: structure

Treewidth is a measure of "cyclicity" of a graph.

Why do we need this notion?

- The higher the treewidth of blank(G), the harder entailment becomes [Pichler et al. 2008].
- Polynomial algorithm for *bounded treewidth:* $O(n^k)$, with k proportional to the treewidth of blank(G); NP-complete in general.

Blank nodes: structure

We studied the blank-node structure of the graphs in our corpus, looking to find trees of blank nodes.

Blank nodes: structure

We studied the blank-node structure of the graphs in our corpus, looking to find trees of blank nodes.

Why trees? Trees behave nicely.

- Entailment is efficient—low treewidth.
- Merging trees produces more trees.

Blank nodes in The Wild: plenty of "good" graphs

Good news #1

Not many graphs have complex blank node structure.

treewidth	# graphs
1	518,831
2	8,134
3	208
4	99
5	23
6	–
7	1

Treewidth in our corpus

The outlier: a graph with "blank treewidth" of 7

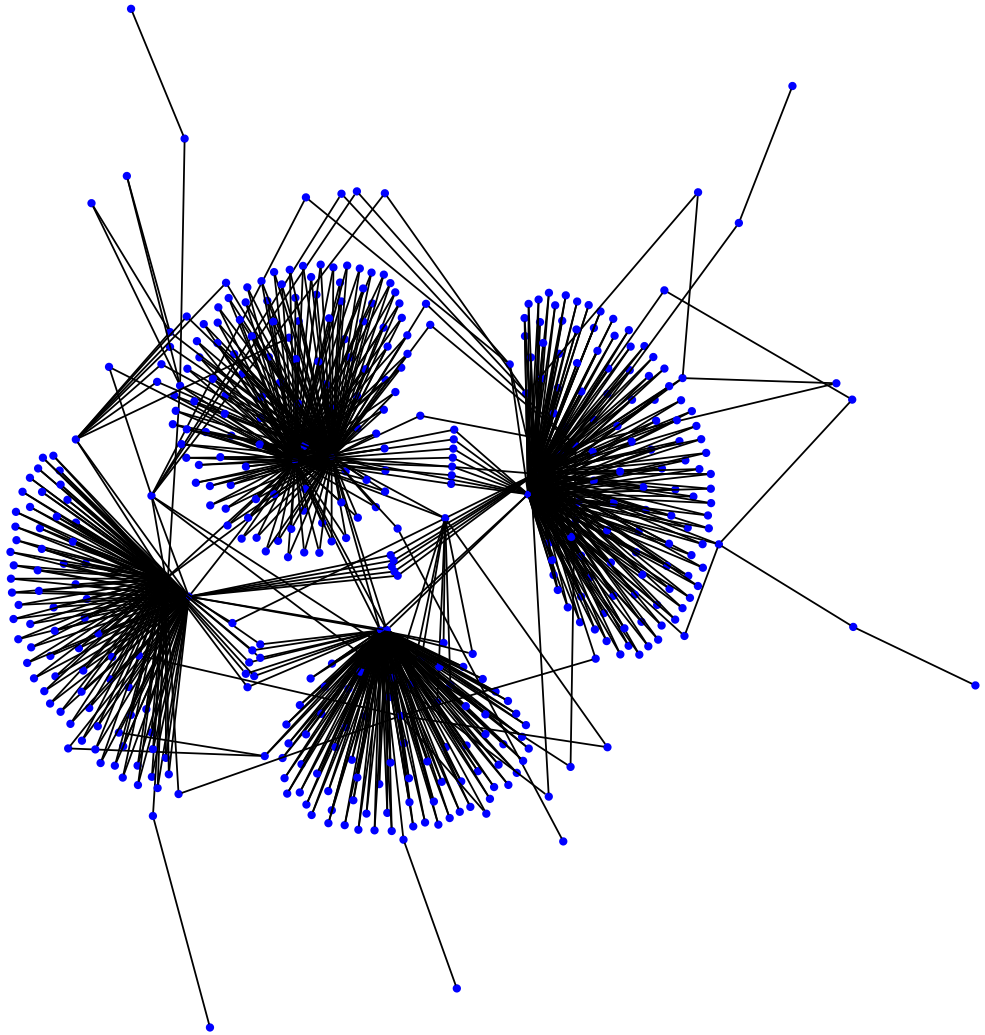
Found in
http://www.rdfabout.com/rdf/usgov/congress/people/B000084,
its "blank version" has treewidth 7.

The outlier: a graph with "blank treewidth" of 7

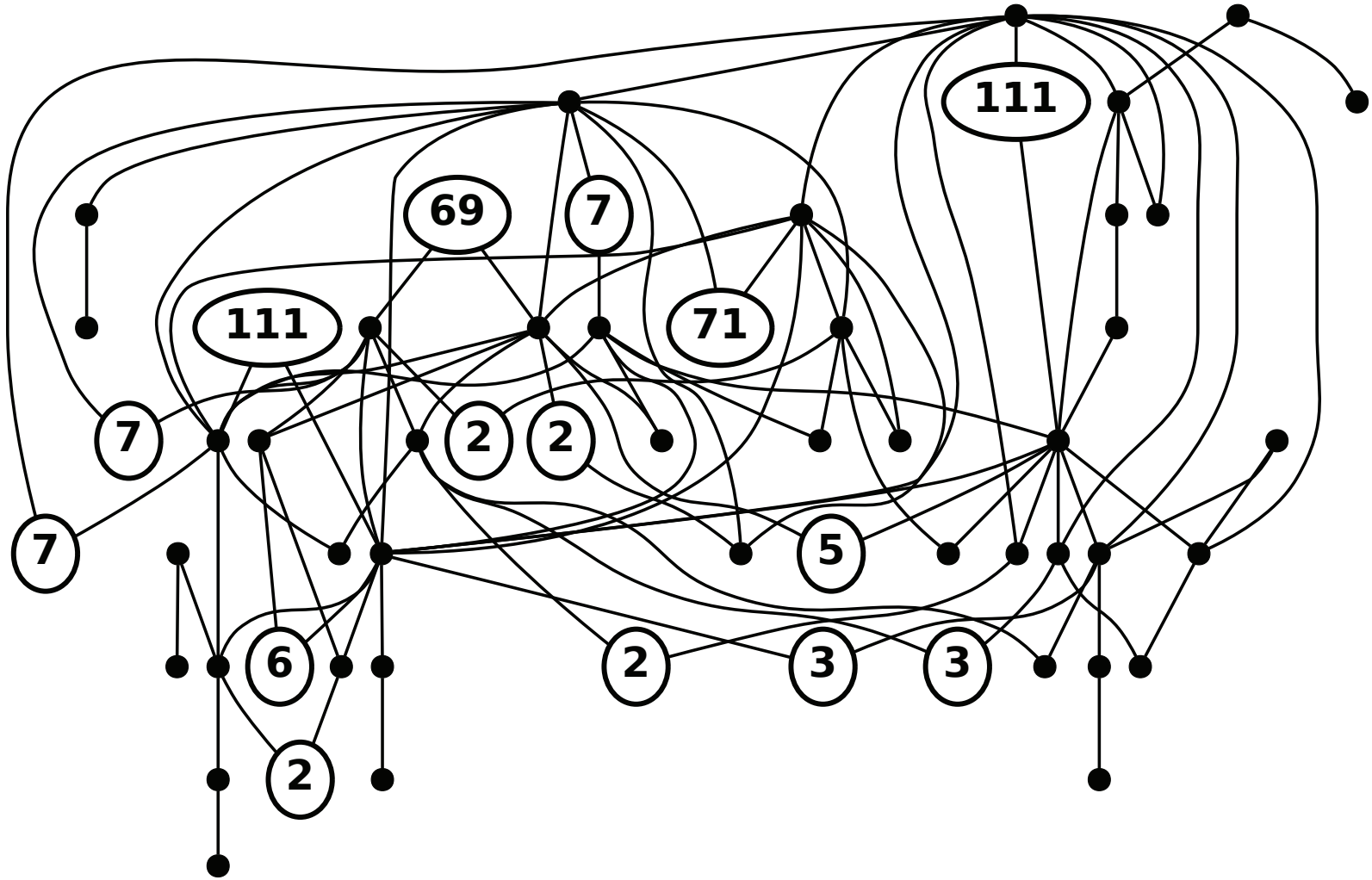
Found in
http://www.rdfabout.com/rdf/usgov/congress/people/B000084,
its "blank version" has treewidth 7.

- You should be impressed ☺

The outlier: a graph with "blank treewidth" of 7



The outlier: a graph with "blank treewidth" of 7



The outlier: a graph with “blank treewidth” of 7

Questions you might have right now:

- “What does this mean?”
- “Should I panic?”
- “How do I avoid this mess in my datasets?”

How to guarantee a tree-structure for blank nodes

Good news #2

When blank nodes are not labeled, we have trees.

How to guarantee a tree-structure for blank nodes

Good news #2

When blank nodes are not labeled, we have trees.

- Bracket syntax in Turtle

```
:Vader :fatherOf [ :name "Luke" ] .
```

How to guarantee a tree-structure for blank nodes

Good news #2

When blank nodes are not labeled, we have trees.

- Bracket syntax in Turtle

```
:Vader :fatherOf [ :name "Luke" ] .
```

- RDF/XML without `nodeID`

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:ex="http://example.org/">
  <rdf:Description rdf:about="http://example.org/Vader">
    <ex:fatherOf rdf:parseType="Resource">
      <ex:name>Luke</ex:name>
    </ex:fatherOf>
  </rdf:Description>
</rdf:RDF>
```

What if I want to get rid of blank nodes?

Good news #3: Skolemization

Process to replace blank nodes with fresh constants.

Original graph

```
:John :telephone _:b1 .
```

What if I want to get rid of blank nodes?

Good news #3: Skolemization

Process to replace blank nodes with fresh constants.

Original graph

```
:John :telephone _:b1 .
```

Skolemized graph

```
:John :telephone newConstant .
```

`newConstant` is a name that has not been used before, anywhere.

What if I want to get rid of blank nodes?

Good news #3: Skolemization

Process to replace blank nodes with fresh constants.

Original graph

```
:John :telephone _:b1 .
```

Skolemized graph

```
:John :telephone newConstant .
```

`newConstant` is a name that has not been used before, anywhere.

- If `newConstant` is a URI, round-tripping becomes possible

What if I want to get rid of blank nodes?

Good news #3: Skolemization

Process to replace blank nodes with fresh constants.

Original graph

```
:John :telephone _:b1 .
```

Skolemized graph

```
:John :telephone newConstant .
```

`newConstant` is a name that has not been used before, anywhere.

- If `newConstant` is a URI, round-tripping becomes possible (but we may lose locality!).

Standardizing the process with a Skolemization scheme

"So. . . how (on earth) do I generate these constants?"

Standardizing the process with a Skolemization scheme

"So. . . how (on earth) do I generate these constants?"

A Skolemization scheme is a set of guidelines for Skolemization.

Standardizing the process with a Skolemization scheme

"So. . . how (on earth) do I generate these constants?"

A Skolemization scheme is a set of guidelines for Skolemization.

- Defines a standard syntax for *Skolem constants*.

Standardizing the process with a Skolemization scheme

"So. . . how (on earth) do I generate these constants?"

A Skolemization scheme is a set of guidelines for Skolemization.

- Defines a standard syntax for *Skolem constants*.
- It is recommended as "best practice".

Standardizing the process with a Skolemization scheme

"So. . . how (on earth) do I generate these constants?"

A Skolemization scheme is a set of guidelines for Skolemization.

- Defines a standard syntax for *Skolem constants*.
- It is recommended as "best practice".
- It doesn't require changes in the semantics of RDF.

Towards a Skolemization scheme

Currently under discussion by the RDF Working Group:

Use well-known URIs (RFC 5785)

`http://my.domain/.well-known/bnode/abc123xyz789`

- Fixed string in well-known URIs
- Fixed string in this particular Skolemization scheme
  - Other options: `genid`, `skolem`
- Locally-unique identifier
- Globally-unique identifier

Towards a Skolemization scheme

Currently under discussion by the RDF Working Group:

Use well-known URIs (RFC 5785)

`http://my.domain/.well-known/bnode/abc123xyz789`

- Fixed string in well-known URIs
- Fixed string in this particular Skolemization scheme
  - Other options: `genid`, `skolem`
- Locally-unique identifier
- Globally-unique identifier

Towards a Skolemization scheme

Currently under discussion by the RDF Working Group:

Use well-known URIs (RFC 5785)

`http://my.domain/.well-known/bnode/abc123xyz789`

- Fixed string in well-known URIs
- Fixed string in this particular Skolemization scheme
  - Other options: `genid`, `skolem`
- Locally-unique identifier
- Globally-unique identifier

Towards a Skolemization scheme

Currently under discussion by the RDF Working Group:

Use well-known URIs (RFC 5785)

`http://my.domain/.well-known/bnode/abc123xyz789`

- Fixed string in well-known URIs
- Fixed string in this particular Skolemization scheme
  - Other options: `genid`, `skolem`
- Locally-unique identifier
- Globally-unique identifier

Towards a Skolemization scheme

Currently under discussion by the RDF Working Group:

Use well-known URIs (RFC 5785)

`http://my.domain/.well-known/bnode/abc123xyz789`

- Fixed string in well-known URIs
- Fixed string in this particular Skolemization scheme
  - Other options: `genid`, `skolem`
- Locally-unique identifier
- Globally-unique identifier

Summary

- There is confusion about the semantics of blank nodes across several standards. In the paper we discuss SPARQL, OWL and RDB2RDF. These uses are not necessarily “in line” with the RDF semantics.

Summary

- There is confusion about the semantics of blank nodes across several standards. In the paper we discuss SPARQL, OWL and RDB2RDF. These uses are not necessarily “in line” with the RDF semantics.
- The intent of publishers is often not aligned with the semantics of blank nodes in RDF.

Summary

- There is confusion about the semantics of blank nodes across several standards. In the paper we discuss SPARQL, OWL and RDB2RDF. These uses are not necessarily "in line" with the RDF semantics.
- The intent of publishers is often not aligned with the semantics of blank nodes in RDF.
- Blank nodes raise the complexity of simple entailment, but it's efficient for most published data (*i.e.*, with low treewidth).

Summary

- There is confusion about the semantics of blank nodes across several standards. In the paper we discuss SPARQL, OWL and RDB2RDF. These uses are not necessarily "in line" with the RDF semantics.
- The intent of publishers is often not aligned with the semantics of blank nodes in RDF.
- Blank nodes raise the complexity of simple entailment, but it's efficient for most published data (*i.e.*, with low treewidth).
- Skolemization is useful when exchanging blank nodes (*e.g.*, round-tripping).

Summary

- There is confusion about the semantics of blank nodes across several standards. In the paper we discuss SPARQL, OWL and RDB2RDF. These uses are not necessarily "in line" with the RDF semantics.
- The intent of publishers is often not aligned with the semantics of blank nodes in RDF.
- Blank nodes raise the complexity of simple entailment, but it's efficient for most published data (*i.e.*, with low treewidth).
- Skolemization is useful when exchanging blank nodes (*e.g.*, round-tripping).
- Furthermore, there should be a recommendation on how to Skolemize in the near future.

Summary

- There is confusion about the semantics of blank nodes across several standards. In the paper we discuss SPARQL, OWL and RDB2RDF. These uses are not necessarily "in line" with the RDF semantics.
- The intent of publishers is often not aligned with the semantics of blank nodes in RDF.
- Blank nodes raise the complexity of simple entailment, but it's efficient for most published data (*i.e.*, with low treewidth).
- Skolemization is useful when exchanging blank nodes (*e.g.*, round-tripping).
- Furthermore, there should be a recommendation on how to Skolemize in the near future.

Thanks to: participants in the poll, reviewers, and, in particular,

Stefan Decker, Richard Cyganiak and Michael Hausenblas for their feedback.