Minimal Deductive Systems for RDF

Sergio Muñoz, Jorge Pérez, Claudio Gutierrez

Department of Computer Science
Universidad de Chile
Pontificia Universidad Católica de Chile

Center for Web Research
http://www.cwr.cl

European Semantic Web Conference 2007

RDF Language and Model Theory looks a little bit complicated at first sight...

W3C

RDF Semantics

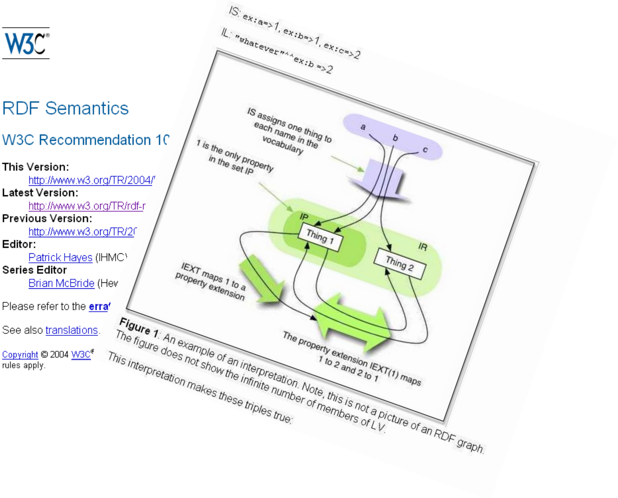
W3C Recommendation 10 February 2004

This Version:
http://www.w3.org/TR/2004/REC-rdf-mt-20040210/
Latest Version:
http://www.w3.org/TR/rdf-mt/
Previous Version:
http://www.w3.org/TR/2003/PR-rdf-mt-20031215/
Editor:
Patrick Hayes (IHMC)< phayes@ihmc.us>
Series Editor
Brian McBride (Hewlett Packard Labs)<bwm@hplb.hpl.hp.com>

Please refer to the **errata** for this document, which may include some norn

See also translations.

RDF Language and Model Theory looks a little bit complicated at first sight...



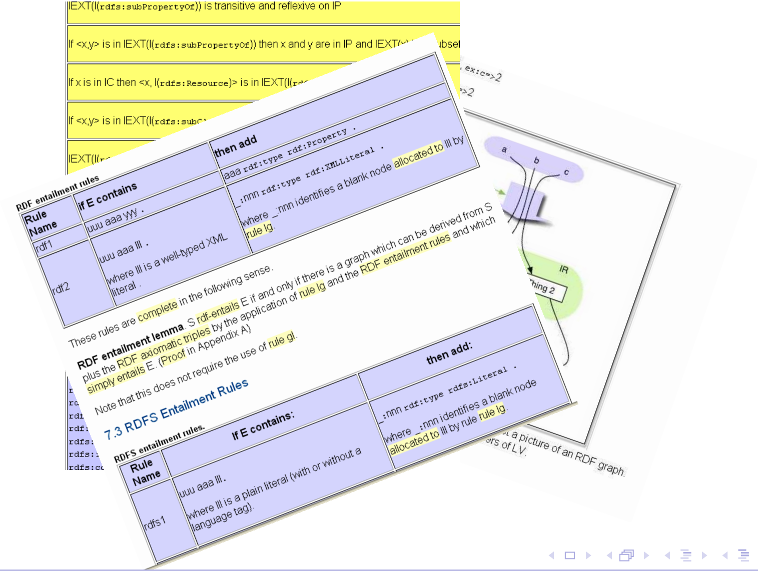
RDF Language and Model Theory looks a little bit complicated at first sight...

IEXT(I(`rdfs:subPropertyOf`)) is transitive and reflexive on IP
If <x,y> is in IEXT(I(`rdfs:subPropertyOf`)) then x and y are in IP and IEXT(x) is a subset
If x is in IC then <x, I(`rdfs:Resource`)> is in IEXT(I(`rdfs:subClassOf`))
If <x,y> is in IEXT(I(`rdfs:subClassOf`)) then x and y are in IC and ICEXT(x) is a subset c
IEXT(I(`rdfs:subClassOf`)) is transitive and reflexive on IC
If x is in ICEXT(I(`rdfs:ContainerMembershipProperty`)) then: < x, I(`rdfs:member`)> is in IEXT(I(`rdfs:subPropertyOf`))
If x is in ICEXT(I(`rdfs:Datatype`)) then <x, I(`rdfs:Literal`)> is in IEXT(I(`rdfs:subClass`

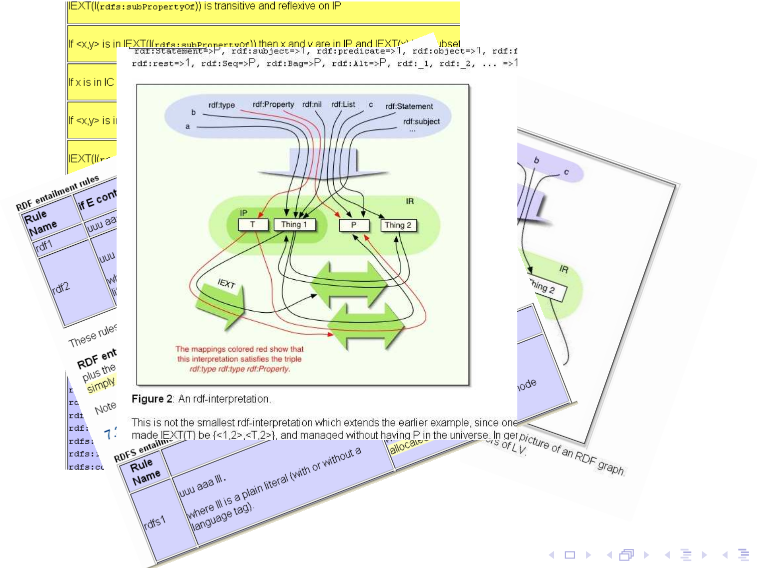
RDFS axiomatic triples.

```
rdf:type rdfs:domain rdfs:Resource .
rdfs:domain rdfs:domain rdf:Property .
rdfs:range rdfs:domain rdf:Property .
rdfs:subPropertyOf rdfs:domain rdf:Property .
rdfs:subClassOf rdfs:domain rdfs:Class .
rdf:subject rdfs:domain rdf:Statement .
rdf:predicate rdfs:domain rdf:Statement .
rdf:object rdfs:domain rdf:Statement .
rdfs:member rdfs:domain rdfs:Resource .
rdf:first rdfs:domain rdf:List .
rdf:rest rdfs:domain rdf:List .
rdfs:seeAlso rdfs:domain rdfs:Resource .
rdfs:isDefinedBy rdfs:domain rdfs:Resource .
rdfs:comment rdfs:domain rdfs:Resource .
```

RDF Language and Model Theory looks a little bit complicated at first sight...



RDF Language and Model Theory looks a little bit complicated at first sight...



Several dozens of reserved keywords...

rdfs:Resource [res]
rdf:Property [prop]
rdfs:Class [class]
rdfs:Literal [literal]
rdfs:Datatype [datatype]
rdf:XMLLiteral [xmlLit]
rdfs:Container [cont]
rdf:Statement [stat]
rdf:List [list]
rdf:Alt [alt]
rdf:Bag [bag]
rdf:Seq [seq]
rdfs:ContainerMembershipProperty [contMP]

rdf:type [type]
rdfs:domain [dom]
rdfs:range [range]
rdfs:subClassOf [sc]
rdfs:subPropertyOf [sp]
rdf:subject [subj]
rdf:predicate [pred]
rdf:object [obj]
rdfs:member [member]
rdf:first [first]
rdf:rest [rest]
rdfs:seeAlso [seeAlso]

rdfs:isDefinedBy [isDefined]
rdfs:comment [comment]
rdfs:label [label]
rdf:value [value]
rdf:nil [nil]
rdf:_1 [_1]
rdf:_2 [_2]
...
rdf:_i [_i]
...

...plus axiomatic triples...

(type,	type,	prop)
(subj,	type,	prop)
(pred,	type,	prop)
(obj,	type,	prop)
(first,	type,	prop)
(rest,	type,	prop)
(value,	type,	prop)
(_1,	type,	prop)
(_1,	type,	contMP)
(_2,	type,	prop)
(_2,	type,	contMP)
...		
(_i,	type,	prop)
(_i,	type,	contMP)
...		
(nil,	type,	prop)
(xmlLit,	type,	datatype)

...plus more axiomatic triples...

(type,	dom,	res)	(type,	range,	class)
(dom,	dom,	prop)	(dom,	range,	class)
(range,	dom,	prop)	(range,	range,	class)
(sp,	dom,	prop)	(sp,	range,	prop)
(sc,	dom,	class)	(sc,	range,	class)
(subj,	dom,	stat)	(subj,	range,	res)
(pred,	dom,	stat)	(pred,	range,	res)
(obj,	dom,	stat)	(obj,	range,	res)
(member,	dom,	res)	(member,	range,	res)
(first,	dom,	list)	(first,	range,	res)
(rest,	dom,	list)	(rest,	range,	list)
(seeAlso,	dom,	res)	(seeAlso,	range,	res)
(isDefined,	dom,	res)	(isDefined,	range,	res)
(comment,	dom,	res)	(comment,	range,	literal)
(label,	dom,	res)	(label,	range,	literal)
(value,	dom,	res)	(value,	range,	res)
(_1,	dom,	res)	(_1,	range,	res)
(_2,	dom,	res)	(_2,	range,	res)
...			...		
(_i,	dom,	res)	(_i,	range,	res)
...			...		

...plus more axiomatic triples...

$$
\begin{array}{lcr}
(\texttt{alt}, & \texttt{sc}, & \texttt{cont}) \\
(\texttt{bag}, & \texttt{sc}, & \texttt{cont}) \\
(\texttt{seq}, & \texttt{sc}, & \texttt{cont}) \\
(\texttt{contMP}, & \texttt{sc}, & \texttt{prop}) \\
(\texttt{xmlLit}, & \texttt{sc}, & \texttt{literal}) \\
(\texttt{datatype}, & \texttt{sc}, & \texttt{class})
\end{array}
$$

$$(\texttt{isDefined}, \quad \texttt{sp}, \quad \texttt{seeAlso})$$

...and on top of this a (slightly) non-standard model theory

- A notion of interpretation

$$(Res, Prop, Class, Lit, PExt, CExt, Int)$$

  including subsets of the universe denoting properties, classes and literals, and mapping defining their extensions.
- Notion of interpretation of blank nodes
- Definition of reflexivity, transitivity and semi-extensionality of subClass and subProperty
- Typing restrictions

But...we need a workable language to bring to reality the vision of the Semantic Web

Would like to:

- Have a simple user-language to be able to popularize RDF among Web users.
- Have a simple specification to allow sound development work.
- Have a language in streamlined form to make it easy to formalize and prove results about its properties.

What is to be done?: To simplify the language

What is to be done?: To simplify the language

There is a minimal fragment of the theory preserving the essential core of RDFS

What is to be done?: To simplify the language

There is a minimal fragment of the theory preserving the essential core of RDFS

- Basic idea: Separate user-language from features and constructors which define and specificy the language.
- Concentrate in vocabulary with non-trivial semantics.

Main contributions & Outline

- Identify a fragment of RDFS that covers the crucial vocabulary and preserves the original RDFS semantics.
- Study dependencies among vocabulary and develop sound and complete proof systems for each fragment.
- Present algorithms to modularize reasoning according to relevant vocabulary.

Main contributions & Outline

- Identify a fragment of RDFS that covers the crucial vocabulary and preserves the original RDFS semantics.
- Study dependencies among vocabulary and develop sound and complete proof systems for each fragment.
- Present algorithms to modularize reasoning according to relevant vocabulary.

Main contributions & Outline

- Identify a fragment of RDFS that covers the crucial vocabulary and preserves the original RDFS semantics.
- Study dependencies among vocabulary and develop sound and complete proof systems for each fragment.
- Present algorithms to modularize reasoning according to relevant vocabulary.

The core vocabulary

rdfs:Resource [res]	rdf:type [type]	rdfs:isDefinedBy [isDefined]
rdf:Property [prop]	rdfs:domain [dom]	rdfs:comment [comment]
rdfs:Class [class]	rdfs:range [range]	rdfs:label [label]
rdfs:Literal [literal]	rdfs:subClassOf [sc]	rdf:value [value]
rdfs:Datatype [datatype]	rdfs:subPropertyOf [sp]	rdf:nil [nil]
rdf:XMLLiteral [xmlLit]	rdf:subject [subj]	rdf:_1 [_1]
rdfs:Container [cont]	rdf:predicate [pred]	rdf:_2 [_2]
rdf:Statement [stat]	rdf:object [obj]	...
rdf:List [list]	rdfs:member [member]	rdf:_i [_i]
rdf:Alt [alt]	rdf:first [first]	...
rdf:Bag [bag]	rdf:rest [rest]	
rdf:Seq [seq]	rdfs:seeAlso [seeAlso]	

rdfs:ContainerMembershipProperty [contMP]

The core vocabulary

rdfs:Resource [res]
rdf:Property [prop]
rdfs:Class [class]
rdfs:Literal [literal]
rdfs:Datatype [datatype]
rdf:XMLLiteral [xmlLit]
rdfs:Container [cont]
rdf:Statement [stat]
rdf:List [list]
rdf:Alt [alt]
rdf:Bag [bag]
rdf:Seq [seq]
rdfs:ContainerMembershipProperty [contMP]

rdf:type [type]
rdfs:domain [dom]
rdfs:range [range]
rdfs:subClassOf [sc]
rdfs:subPropertyOf [sp]
rdf:subject [subj]
rdf:predicate [pred]
rdf:object [obj]
rdfs:member [member]
rdf:first [first]
rdf:rest [rest]
rdfs:seeAlso [seeAlso]

rdfs:isDefinedBy [isDefined]
rdfs:comment [comment]
rdfs:label [label]
rdf:value [value]
rdf:nil [nil]
rdf:_1 [_1]
rdf:_2 [_2]
...
rdf:_i [_i]
...

The core vocabulary

rdfs:Resource [res]
rdf:Property [prop]
rdfs:Class [class]
rdfs:Literal [literal]
rdfs:Datatype [datatype]
rdf:XMLLiteral [xmlLit]
rdfs:Container [cont]
rdf:Statement [stat]
rdf:List [list]
rdf:Alt [alt]
rdf:Bag [bag]
rdf:Seq [seq]
rdfs:ContainerMembershipProperty [contMP]

rdf:type [type]
rdfs:domain [dom]
rdfs:range [range]
rdfs:subClassOf [sc]
rdfs:subPropertyOf [sp]
rdf:subject [subj]
rdf:predicate [pred]
rdf:object [obj]
rdfs:member [member]
rdf:first [first]
rdf:rest [rest]
rdfs:seeAlso [seeAlso]

rdfs:isDefinedBy [isDefined]
rdfs:comment [comment]
rdfs:label [label]
rdf:value [value]
rdf:nil [nil]
rdf:_1 [_1]
rdf:_2 [_2]
...
rdf:_i [_i]
...

$$\rho\text{df} = \{\texttt{sp}, \texttt{sc}, \texttt{type}, \texttt{dom}, \texttt{range}\}$$

Blank node rule

$\frac{G}{H}$ if there is a homomorphism $\mu : H \rightarrow G$

Core rules

Subproperty (transitivity, definition)

$$\frac{(A,\mathrm{sp},B)\ \ (B,\mathrm{sp},C)}{(A,\mathrm{sp},C)} \qquad \frac{(A,\mathrm{sp},B)\ \ (X,A,Y)}{(X,B,Y)}$$

Subclass (transitivity, definition)

$$\frac{(A,\mathrm{sc},B)\ \ (B,\mathrm{sc},C)}{(A,\mathrm{sc},C)} \qquad \frac{(A,\mathrm{sc},B)\ \ (X,\mathrm{type},A)}{(X,\mathrm{type},B)}$$

Typing (domain, range)

$$\frac{(A,\mathrm{dom},B)\ \ (X,A,Y)}{(X,\mathrm{type},B)} \qquad \frac{(A,\mathrm{range},B)\ \ (X,A,Y)}{(Y,\mathrm{type},B)}$$

Implicit Typing (strange case...)

$$\frac{(A,\mathrm{dom},B)\ \ (C,\mathrm{sp},A)\ \ (X,C,Y)}{(X,\mathrm{type},B)} \qquad \frac{(A,\mathrm{range},B)\ \ (C,\mathrm{sp},A)\ \ (X,C,Y)}{(Y,\mathrm{type},B)}$$

Core rules

Subproperty (transitivity, definition)

$$\frac{(A,\mathrm{sp},B)\ \ (B,\mathrm{sp},C)}{(A,\mathrm{sp},C)} \qquad \frac{(A,\mathrm{sp},B)\ \ (X,A,Y)}{(X,B,Y)}$$

Subclass (transitivity, definition)

$$\frac{(A,\mathrm{sc},B)\ \ (B,\mathrm{sc},C)}{(A,\mathrm{sc},C)} \qquad \frac{(A,\mathrm{sc},B)\ \ (X,\mathrm{type},A)}{(X,\mathrm{type},B)}$$

Typing (domain, range)

$$\frac{(A,\mathrm{dom},B)\ \ (X,A,Y)}{(X,\mathrm{type},B)} \qquad \frac{(A,\mathrm{range},B)\ \ (X,A,Y)}{(Y,\mathrm{type},B)}$$

Implicit Typing (strange case...)

$$\frac{(A,\mathrm{dom},B)\ \ (C,\mathrm{sp},A)\ \ (X,C,Y)}{(X,\mathrm{type},B)} \qquad \frac{(A,\mathrm{range},B)\ \ (C,\mathrm{sp},A)\ \ (X,C,Y)}{(Y,\mathrm{type},B)}$$

Core rules

Subproperty (transitivity, definition)

$$\frac{(A,\mathtt{sp},B)\ \ (B,\mathtt{sp},C)}{(A,\mathtt{sp},C)} \qquad \frac{(A,\mathtt{sp},B)\ \ (X,A,Y)}{(X,B,Y)}$$

Subclass (transitivity, definition)

$$\frac{(A,\mathtt{sc},B)\ \ (B,\mathtt{sc},C)}{(A,\mathtt{sc},C)} \qquad \frac{(A,\mathtt{sc},B)\ \ (X,\mathtt{type},A)}{(X,\mathtt{type},B)}$$

Typing (domain, range)

$$\frac{(A,\mathtt{dom},B)\ \ (X,A,Y)}{(X,\mathtt{type},B)} \qquad \frac{(A,\mathtt{range},B)\ \ (X,A,Y)}{(Y,\mathtt{type},B)}$$

Implicit Typing (strange case...)

$$\frac{(A,\mathtt{dom},B)\ \ (C,\mathtt{sp},A)\ \ (X,C,Y)}{(X,\mathtt{type},B)} \qquad \frac{(A,\mathtt{range},B)\ \ (C,\mathtt{sp},A)\ \ (X,C,Y)}{(Y,\mathtt{type},B)}$$

Core rules

Subproperty (transitivity, definition)

$$\frac{(A,\mathtt{sp},B)\ \ (B,\mathtt{sp},C)}{(A,\mathtt{sp},C)} \qquad \frac{(A,\mathtt{sp},B)\ \ (X,A,Y)}{(X,B,Y)}$$

Subclass (transitivity, definition)

$$\frac{(A,\mathtt{sc},B)\ \ (B,\mathtt{sc},C)}{(A,\mathtt{sc},C)} \qquad \frac{(A,\mathtt{sc},B)\ \ (X,\mathtt{type},A)}{(X,\mathtt{type},B)}$$

Typing (domain, range)

$$\frac{(A,\mathtt{dom},B)\ \ (X,A,Y)}{(X,\mathtt{type},B)} \qquad \frac{(A,\mathtt{range},B)\ \ (X,A,Y)}{(Y,\mathtt{type},B)}$$

Implicit Typing (strange case...)

$$\frac{(A,\mathtt{dom},B)\ \ (C,\mathtt{sp},A)\ \ (X,C,Y)}{(X,\mathtt{type},B)} \qquad \frac{(A,\mathtt{range},B)\ \ (C,\mathtt{sp},A)\ \ (X,C,Y)}{(Y,\mathtt{type},B)}$$

Core rules

Subproperty (transitivity, definition)

$$\frac{(A,\texttt{sp},B)\quad(B,\texttt{sp},C)}{(A,\texttt{sp},C)} \qquad \frac{(A,\texttt{sp},B)\quad(X,A,Y)}{(X,B,Y)}$$

Subclass (transitivity, definition)

$$\frac{(A,\texttt{sc},B)\quad(B,\texttt{sc},C)}{(A,\texttt{sc},C)} \qquad \frac{(A,\texttt{sc},B)\quad(X,\texttt{type},A)}{(X,\texttt{type},B)}$$

Typing (domain, range)

$$\frac{(A,\texttt{dom},B)\quad(X,A,Y)}{(X,\texttt{type},B)} \qquad \frac{(A,\texttt{range},B)\quad(X,A,Y)}{(Y,\texttt{type},B)}$$

Implicit Typing (strange case...)

$$\frac{(A,\texttt{dom},B)\quad(C,\texttt{sp},A)\quad(X,C,Y)}{(X,\texttt{type},B)} \qquad \frac{(A,\texttt{range},B)\quad(C,\texttt{sp},A)\quad(X,C,Y)}{(Y,\texttt{type},B)}$$

Reflexivity rules

If "$(A, \texttt{type}, \mathit{Property})$" then $(A, \texttt{sp}, A)$

Subproperty Reflexivity

$$\frac{(X,A,Y)}{(A,\texttt{sp},A)} \qquad \frac{}{(p,\texttt{sp},p)} \text{ for } p \in \rho\text{df}$$

$$\frac{(A,\texttt{sp},B)}{(A,\texttt{sp},A) \;\; (B,\texttt{sp},B)} \qquad \frac{(A,\texttt{dom},X)}{(A,\texttt{sp},A)} \qquad \frac{(A,\texttt{range},X)}{(A,\texttt{sp},A)}$$

Subclass Reflexivity

$$\frac{(A,\texttt{sc},B)}{(A,\texttt{sc},A) \;\; (B,\texttt{sc},B)} \qquad \frac{(X,\texttt{range},A)}{(A,\texttt{sc},A)}$$

$$\frac{(X,\texttt{dom},A)}{(A,\texttt{sc},A)} \qquad \frac{(X,\texttt{type},A)}{(A,\texttt{sc},A)}$$

Reflexivity rules

If "$(A, \texttt{type}, \mathit{Property})$" then $(A, \texttt{sp}, A)$

Subproperty Reflexivity

$$\frac{(X,A,Y)}{(A,\texttt{sp},A)} \qquad \frac{}{(p,\texttt{sp},p)} \text{ for } p \in \rho\text{df}$$

$$\frac{(A,\texttt{sp},B)}{(A,\texttt{sp},A)\ \ (B,\texttt{sp},B)} \qquad \frac{(A,\texttt{dom},X)}{(A,\texttt{sp},A)} \qquad \frac{(A,\texttt{range},X)}{(A,\texttt{sp},A)}$$

Subclass Reflexivity

$$\frac{(A,\texttt{sc},B)}{(A,\texttt{sc},A)\ \ (B,\texttt{sc},B)} \qquad \frac{(X,\texttt{range},A)}{(A,\texttt{sc},A)}$$

$$\frac{(X,\texttt{dom},A)}{(A,\texttt{sc},A)} \qquad \frac{(X,\texttt{type},A)}{(A,\texttt{sc},A)}$$

Soundness and Completeness

Let $\models$ denote the standard RDFS entailment, and $\vdash_{\rho\mathrm{df}}$ a proof system based on the rules presented.

Theorem

Let G and H be graphs in ρdf then

$$G \models H \text{ if and only if } G \vdash_{\rho\mathrm{df}} H.$$

Blank Nodes Modularization

Blank nodes can be treated in an orthogonal form to ρdf vocabulary.

Theorem

Let G and H be graphs in ρdf and $G \models H$, then

there is a proof of H from G where the blank rule is used at most once and at the end.

The role of reflexivity

The only consequence of reflexivity of $\mathtt{sp}$ and $\mathtt{sc}$ in RDFS is the possible entailment of triples of the form $(x, \mathtt{sp}, x), (x, \mathtt{sc}, x)$.

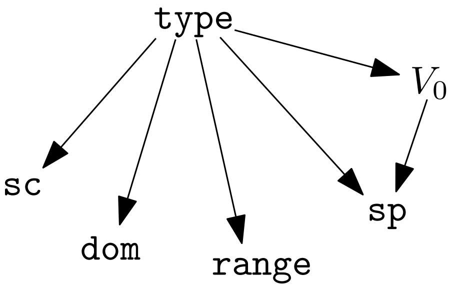
Theorem

Let G and H be ρdf graphs. Assume that H does not contain triples of the form $(x, \mathtt{sp}, x)$ and $(x, \mathtt{sc}, x)$. Then,

$$G \vdash_{\rho\mathrm{df}} H \text{ without using reflexivity rules.}$$

(Also, by not imposing reflexivity, axiomatic triples can be completely avoided.)

Dependence diagram among ρdf vocabulary



To determine $G \models H$ it is enough to test $G' \models H$ where G' is the subgraph of G which involves only nodes in voc(H) and their dependencies in the diagram.

It is possible to avoid the closure to test RDFS entailment

- A naive approach to test $G \models H$ is:
  - (pre-)compute the closure of G
  - check if H is contained in the closure of G.

Theorem

The size of the closure of G is $O(|G|^2)$, and this bound is tight.

It is possible to avoid the closure to test RDFS entailment

- A naive approach to test $G \models H$ is:
  - (pre-)compute the closure of G
  - check if H is contained in the closure of G.

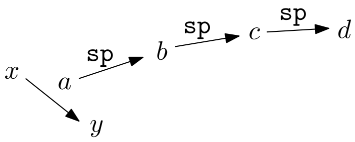
Theorem

The size of the closure of G is $O(|G|^2)$, and this bound is tight.

- Alternative: to use a goal oriented approach based on the dependencies diagram.

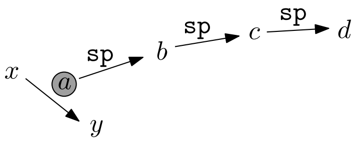
Goal oriented entailment algorithm

Does the graph entails (x, d, y)?
Look for triples of the form (x, a, y) and sp-paths from a to d.



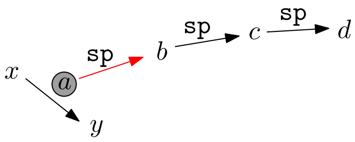
Goal oriented entailment algorithm

Does the graph entails (x, d, y)?
Look for triples of the form (x, a, y) and sp-paths from a to d.



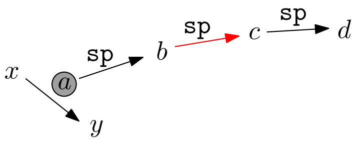
Goal oriented entailment algorithm

Does the graph entails (x, d, y)?
Look for triples of the form (x, a, y) and sp-paths from a to d.



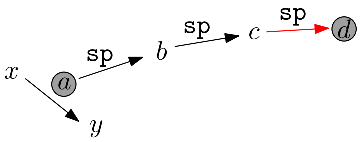
Goal oriented entailment algorithm

Does the graph entails (x, d, y)?
Look for triples of the form (x, a, y) and sp-paths from a to d.



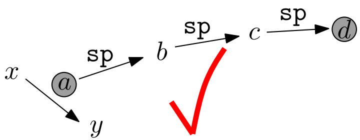
Goal oriented entailment algorithm

Does the graph entails (x, d, y)?
Look for triples of the form (x, a, y) and sp-paths from a to d.



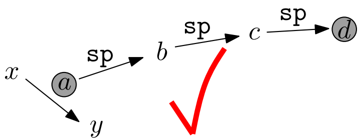
Goal oriented entailment algorithm

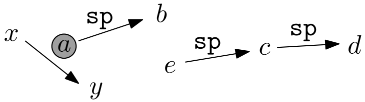
Does the graph entails (x, d, y)?
Look for triples of the form (x, a, y) and sp-paths from a to d.



Goal oriented entailment algorithm

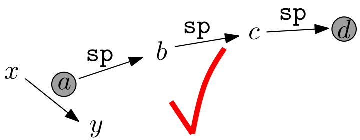
Does the graph entails (x, d, y)?
Look for triples of the form (x, a, y) and sp-paths from a to d.

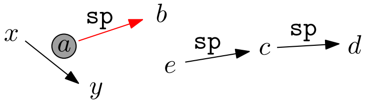




Goal oriented entailment algorithm

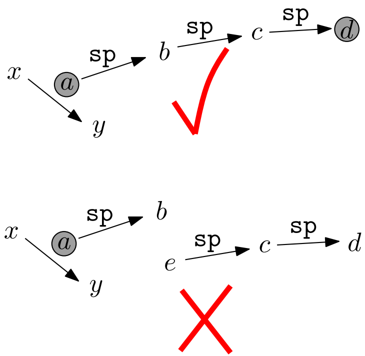
Does the graph entails (x, d, y)?
Look for triples of the form (x, a, y) and sp-paths from a to d.





Goal oriented entailment algorithm

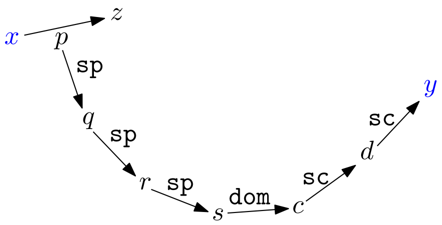
Does the graph entails (x, d, y)?
Look for triples of the form (x, a, y) and sp-paths from a to d.



Goal oriented entailment algorithm

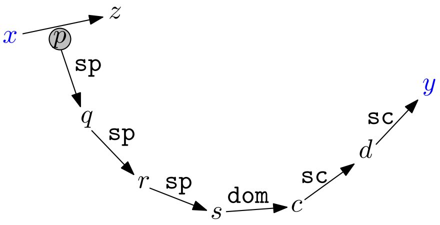
Let $G = \{\ldots, (x, p, z), (p, sp, q), (q, sp, r), (r, sp, s), (s, \texttt{dom}, c), (c, sc, d), (d, sc, y), \ldots\}$

Does the graph entails $(x, \texttt{type}, y)$?



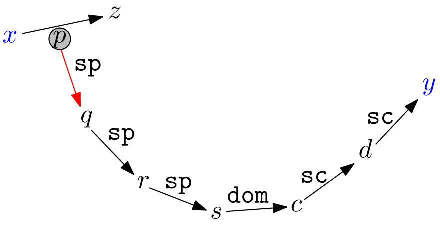
Goal oriented entailment algorithm

Let $G = \{\ldots, (x, p, z), (p, sp, q), (q, sp, r), (r, sp, s), (s, \texttt{dom}, c),$
$(c, sc, d), (d, sc, y), \ldots\}$
Does the graph entails $(x, \texttt{type}, y)$?



Goal oriented entailment algorithm

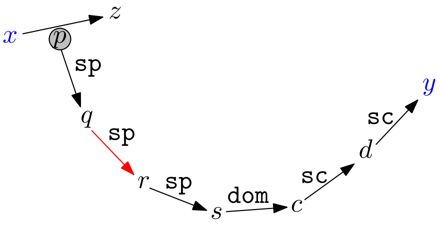
Let $G = \{\ldots, (x, p, z), (p, sp, q), (q, sp, r), (r, sp, s), (s, \texttt{dom}, c),$
$(c, sc, d), (d, sc, y), \ldots\}$
Does the graph entails $(x, \texttt{type}, y)$?



Goal oriented entailment algorithm

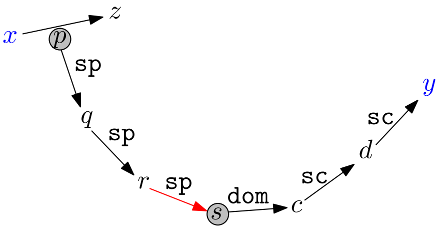
Let $G = \{\dots, (x, p, z), (p, sp, q), (q, sp, r), (r, sp, s), (s, \texttt{dom}, c), (c, sc, d), (d, sc, y), \dots\}$

Does the graph entails $(x, \texttt{type}, y)$?



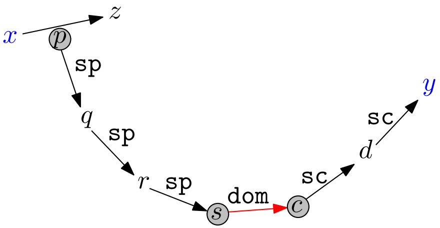
Goal oriented entailment algorithm

Let $G = \{\ldots, (x, p, z), (p, sp, q), (q, sp, r), (r, sp, s), (s, \texttt{dom}, c),$
$(c, sc, d), (d, sc, y), \ldots\}$
Does the graph entails $(x, \texttt{type}, y)$?



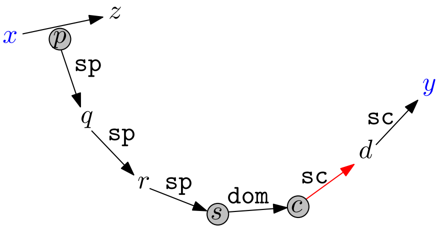
Goal oriented entailment algorithm

Let $G = \{\ldots, (x, p, z), (p, sp, q), (q, sp, r), (r, sp, s), (s, \texttt{dom}, c),$
$(c, sc, d), (d, sc, y), \ldots\}$
Does the graph entails $(x, \texttt{type}, y)$?

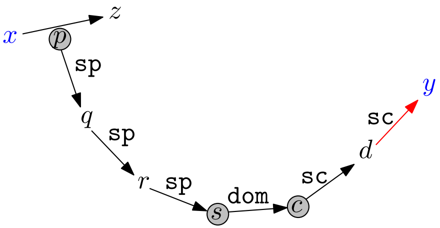


Goal oriented entailment algorithm

Let $G = \{\ldots, (x, p, z), (p, sp, q), (q, sp, r), (r, sp, s), (s, \texttt{dom}, c),$
$(c, sc, d), (d, sc, y), \ldots\}$
Does the graph entails $(x, \texttt{type}, y)$?

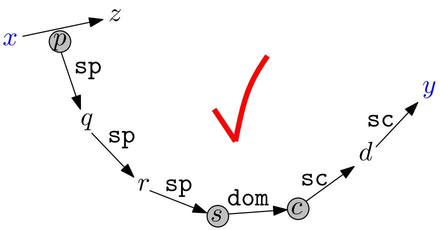


Let $G = \{\ldots, (x, p, z), (p, sp, q), (q, sp, r), (r, sp, s), (s, \texttt{dom}, c),$
$(c, sc, d), (d, sc, y), \ldots\}$
Does the graph entails $(x, \texttt{type}, y)$?



Goal oriented entailment algorithm

Let $G = \{\dots, (x, p, z), (p, sp, q), (q, sp, r), (r, sp, s), (s, \texttt{dom}, c),$
$(c, sc, d), (d, sc, y), \dots\}$
Does the graph entails $(x, \texttt{type}, y)$?



Entailment can be done in $O(n \log n)$ time

Theorem

The goal oriented algorithm takes $O(|G| \log |G|)$ time in testing the entailment $G \models t$.

- Correctness follows by the dependencies diagram.
- Essentially time $|G| \log |G|$ in constructing the necessary graph data-structures.
- Time $|G|$ in traversing these data-structures.

There is no more efficient approach to test $G \models t$!

Entailment can be done in $O(n \log n)$ time

Theorem

The goal oriented algorithm takes $O(|G| \log |G|)$ time in testing the entailment $G \models t$.

- Correctness follows by the dependencies diagram.
- Essentially time $|G| \log |G|$ in constructing the necessary graph data-structures.
- Time $|G|$ in traversing these data-structures.

There is no more efficient approach to test $G \models t$!

The $O(n \log n)$ bound is tight.

Theorem

Testing $G \models t$ *takes time* $\Omega(|G| \log |G|)$.

Idea: Coding the set-disjointness problem, which is $\Omega(n \log n)$

Conclusions

- Core fragment of RDFS well-behaved, representative, and simple.

Conclusions

- Core fragment of RDFS well-behaved, representative, and simple.
- Efficient algorithm to check entailment

Conclusions

- Core fragment of RDFS well-behaved, representative, and simple.
- Efficient algorithm to check entailment
- Suggestions
  - Add missing rules
  - Eliminate reflexivity
  - Treat bnodes orthogonal to rest

Conclusions

- Core fragment of RDFS well-behaved, representative, and simple.
- Efficient algorithm to check entailment
- Suggestions
  - Add missing rules
  - Eliminate reflexivity
  - Treat bnodes orthogonal to rest
- Next: Navigational language based on testing algorithm