Languages for Linked Data

Vladimiro Sassone

joint to various extent with Gabriel Ciobanu, Mariangiola Dezani, Ross Horne, Giuseppe Castagna, Giorgio Ghelli, Kim Nguyen, ...

21 June 2014

The Web of Linked Data

- The Web of Hypertext (1989[1]–): Emphasis on documents interlinked using URIs.
- The Semantic Web (2001[2]–2006[3]): Emphasis on deep ontologies classifying everything (we can learn from this as an AI winter).
- The Web of Linked Data (2006[4]–): Emphasis on raw data interlinked using URIs and delivered by simple data APIs.

[1] Berners-Lee. Information management: A proposal
[2] Berners-Lee, Lassila & Hendler. The Semantic Web
[3] Berners-Lee, Hall & Shadbolt. The Semantic Web revisited
[4] Berners-Lee. Linked Data — design issues

Four Principles of Linked Data

- Use URIs to identify resources.
- Use HTTP URIs to identify resources so we can look them up.
- When a URI is looked up, return data about the resource using the standards.
- Include URIs in the data, so they can also be looked up.

Dereferencing the URI *dbpedia:Kazakhstan*

- `curl -I -H "Accept:text/n3" http://dbpedia.org/resource/Kazakhstan`
- Request:

```
GET /resource/Kazakhstan HTTP/1.1
Host:  dbpedia.org
Accept:  text/n3
```

- Response:

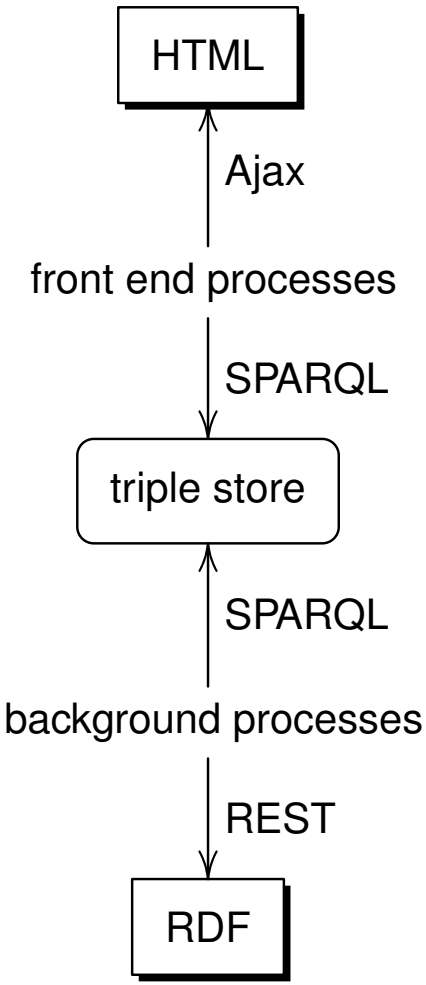
```
HTTP/1.1 303 See Other
Content-Type:  text/n3
Location:  http://dbpedia.org/data/Kazakhstan.n3
```

Dereferencing the URI *dbpedia:Kazakhstan*

```
curl -H "Accept:text/n3" http://dbpedia.org/data/Kazakhstan.n3
```

```
@prefix dbpprop: <http://dbpedia.org/property/> .
@prefix dbpedia: <http://dbpedia.org/resource/> .
dbpedia:Medeo dbpedia-owl:location dbpedia:Kazakhstan .
dbpedia:Zhetysu_Stadium dbpedia-owl:location dbpedia:Kazakhstan .
dbpedia:Astana_Arena dbpedia-owl:location dbpedia:Kazakhstan .
dbpedia:Kazakhstan_Sports_Palace dbpedia-owl:location dbpedia:Kazakhstan .
dbpedia:Munayshy_Stadium dbpedia-owl:location dbpedia:Kazakhstan .
dbpedia:Aral_Sea dbpedia-owl:location dbpedia:Kazakhstan .
dbpedia:Aral_Sea dbpedia-owl:country dbpedia:Kazakhstan .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ns4: <http://en.wikipedia.org/wiki/> .
ns4:Kazakhstan foaf:primaryTopic dbpedia:Kazakhstan .
dbpedia:Air_Kokshetau dbpprop:headquarters dbpedia:Kazakhstan .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix ns6: <http://data.nytimes.com/> .
ns6:N630326210260860620 91 owl:sameAs dbpedia:Kazakhstan .
dbpedia:Rakhimzhan_Qoshqarbaev dbpprop:placeOfBirth dbpedia:Kazakhstan .
@prefix yago-res: <http://mpii.de/yago/resource/> .
yago-res:Kazakhstan owl:sameAs dbpedia:Kazakhstan .
dbpedia:Regina_Kulikova dbpedia-owl:birthPlace dbpedia:Kazakhstan .
dbpedia:Almaty_International_School dbpprop:country dbpedia:Kazakhstan .
dbpedia:The_Gift_to_Stalin dbpedia-owl:country dbpedia:Kazakhstan .
dbpedia:Dmytro_Salamatin dbpedia-owl:birthPlace dbpedia:Kazakhstan .
dbpedia:Dungan_language dbpedia-owl:spokenIn dbpedia:Kazakhstan .
dbpedia:Kazakhstan dbpprop:currencyCode "KZT"@en .
dbpedia:Kazakhstan dbpedia-owl:percentageOfAreaWater "1.7"^^xsd:float .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix yago: <http://dbpedia.org/class/yago/> .
dbpedia:Kazakhstan rdf:type yago:StatesAndTerritoriesEstablishedIn1991 .
@prefix ns10: <http://umbel.org/umbel/rc/> .
dbpedia:Kazakhstan rdf:type ns10:Location_Underspecified .
dbpedia:Kazakhstan rdf:type dbpedia-owl:PopulatedPlace ,
dbpedia:Kazakhstan rdf:type yago:CentralAsianCountries ,
dbpedia:Kazakhstan rdf:type yago:LandlockedCountries .
@prefix ns11: <http://schema.org/> .
dbpedia:Kazakhstan rdf:type ns11:Country ,
dbpedia:Kazakhstan rdf:type dbpedia-owl:Country ,
dbpedia:Kazakhstan rdf:type yago:YagoGeoEntity ,
dbpedia:Kazakhstan rdf:type dbpedia-owl:Place ,
dbpedia:Kazakhstan rdf:type yago:Economy108366753 .
```

An Architecture for Linked Data Consumers



Front end: traditional Web architecture,
with SPARQL replacing SQL.

Triples store: graph based query and update, suited to
combining diverse data sources so they can be collectively queried.

Back end: pulls raw data from the Web,
using RESTful open data APIs.

What are the programming language problems?

- Building the front end is relatively easy. Web development platforms support SPARQL queries, to a decent level.
- Building the back end is new and non-trivial.
  - What makes a high-level domain specific language that supports scripts that consume data from heterogeneous RESTful data APIs and combine the data in a triple store?
  - What is an appropriate programming language and type system that can catch routine programming errors[5]?
  - How to provide typing support Language based support for database aspects?[6]
  - What is the operational semantics and derived algebra for languages that interact with triple stores?
    - Sequential consistency as assumed by labelled transition systems and bisimulation works[7]; but is really too strong a semantics when there are atomic transactions involving multiple triples in a distributed environment.
    - Causal consistency (e.g., Pomsets) is more accurate[8] for read-write transactions.
    - Perhaps still weaker models are required for non-blocking read-only transactions...
  - How to keep the local view of data up-to-date (using stochastic analysis of data sources[9])?
  - Language based support for provenance[10], trust, security, etc.

[5] Ciobanu, Horne & Sassone (2013)
[6] Castagna, Ghelli, Nguyen, Sassone (in progress)
[7] Horne & Sassone (2011)
[8] Ciobanu & Horne (2012)
[9] Cho, Garcia-Molina & Page (1998)
[10] Dezani, Horne & Sassone (2012)

Front end queries and background scripts.

A *front end* SPARQL query that discovers a URI for the capital of Kazakhstan:

```
select $x
from named dbpedia:Kazakhstan
where
    { graph dbpedia:Kazakhstan {dbpedia:Kazakhstan  dbp:capital  $x} }
limit 1
```

A *background* script that finds the URI for the capital of Kazakhstan, then loads dereferenced data into the triple store in a *named graph* identified by the discovered URI.

```
select $x
from named dbpedia:Kazakhstan
where
    graph dbpedia:Kazakhstan {dbpedia:Kazakhstan  dbp:capital  $x}
from named $x
```

Requirements of Type System for Linked Data

Must reflect the following W3C recommendations:

- XML Datatypes[11] are used for literals (*xsd:string*, *xsd:dateTime*, *xsd:decimal*, *xsd:integer* and *xsd:anyURI*).
- RDF[12] is the triple based data format.

$$\textit{dbpedia:Kazakhstan}\ \ \textit{dbp:capital}\ \ \textit{dbpedia:Astana}\ \ .$$

- From RDF Schema[13] use property ranges (using *rdfs:range*). e.g.

$$\mathcal{S}\,(\textit{rdfs:label}) = \textit{xsd:string}$$

- In OWL[14], *owl:ObjectProperty* corresponds to *range*(*xsd:anyURI*), and *owl:Thing* corresponds to *xsd:anyURI*.
- In SPARQL[15] datatypes are used in Boolean filters.

Furthermore, in Facebook's Open Graph protocol, "properties have 'types' which determine the format of their values,"[16] as in our work.

---

[11] XML Schema part 2: Datatypes Second Edition. 2004

[12] Resource Description Framework: Concepts and Abstract Syntax. 2004

[13] RDF Vocabulary Description Language 1.0: RDF Schema. 2004

[14] OWL2 Web Ontology Language Primer (Second Edition). 2012

[15] SPARQL 1.1 Query Language. 2013

[16] https://developers.facebook.com/docs/opengraph/property-types/ (accessed 27.3.2013)

Example of Typed Script: language tags

Assume that $\mathcal{S}$ (*rdfs:label*) = *xsd:string*.

```
do   select $g: xsd:anyURI, $x: xsd:anyURI, $y: xsd:string
     where
         graph $g {$x rdfs:label $y}
         langMatches($y, ru)
     from named $x
```

The script finds resources in any named graph that have a label in the Russian language. It then dereferences the resources. The script is iterated as many times as the implementation feels necessary, without revisiting data.

Example of Typed Script: regular expressions

Assume that $\mathcal{S}(\textit{rdfs:comment}) = \textit{xsd:string}$
and $\mathcal{S}(\textit{rdfs:label}) = \textit{xsd:string}$.

```
select $p: range(xsd:anyURI), $y: xsd:string, $z: xsd:anyURI
where
    {
       graph dbp: {$p rdfs:label $y}
       union
       graph dbp: {$p rdfs:comment $y}
    }
    graph dbpedia:Kazakhstan {$z $p dbpedia:Kazakhstan}
    regex($y, location) && langMatches($y, en)
from named $z
```

The above well typed script looks in two named graphs. In the named graph dbpedia:Kazakhstan it looks for properties with dbpedia:Kazakhstan as the object, and in the named graph dbp: it looks for properties that have either a label or comment that contains the string "location".

The Syntax of Typed Scripts

script ::=	`where` *query script*	*satisfy a query*
|	`from named` *term script*	*dereference a URI*
|	`select` *variable*`:` *type script*	*select a binding*
|	`do` *script*	*iterate script*
|	`success`	*successfully terminate*

datatype ::= *xsd:anyURI* | *xsd:string* | *xsd:decimal* | *xsd:dateTime* | *xsd:integer*

type ::= *datatype* | *range*(*datatype*)

term ::= *variable* | *uri* | *string* | *integer* | *decimal* | *dateTime*

expr ::= *term* | `now` | `str`(*expr*) | `abs`(*expr*) | *expr* + *expr* | *expr* − *expr* | ...

boolean ::= *boolean* || *boolean* | *boolean* && *boolean* | ¬*boolean*
| `regex` (*expr*, *regex*) | `langMatches`(*expr*, *lang-range*) | *expr* < *expr* | ...

triples ::= *term term term* | *triples triples* *data* ::= `graph` *term* {*triples*} | *data data*

query ::= *data* | *boolean* | *query query* | *query* `union` *query*

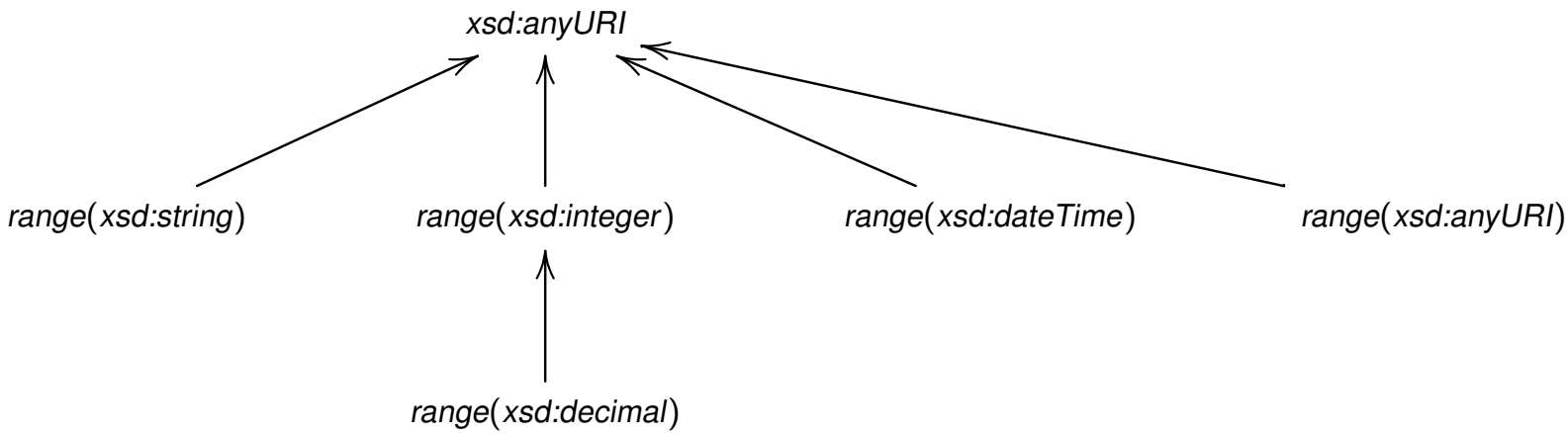
Subtyping

Subtype rules

$$\frac{}{\vdash \mathit{xsd{:}integer} \leq \mathit{xsd{:}decimal}} \qquad \frac{}{\vdash \mathit{type} \leq \mathit{type}}$$

$$\frac{\vdash \mathit{datatype}_1 \leq \mathit{datatype}_2}{\vdash \mathit{range}(\mathit{datatype}_2) \leq \mathit{range}(\mathit{datatype}_1)} \qquad \frac{}{\vdash \mathit{range}(\mathit{datatype}) \leq \mathit{xsd{:}anyURI}}$$

Partial order over types for URIs



Type System: scripts

$$\frac{\Gamma \vdash \mathit{query}_1 \quad \Gamma \vdash \mathit{query}_2}{\Gamma \vdash \mathit{query}_1\ \mathit{query}_2} \qquad \frac{\Gamma \vdash \mathit{query}_1 \quad \Gamma \vdash \mathit{query}_2}{\Gamma \vdash \mathit{query}_1\ \texttt{union}\ \mathit{query}_2} \qquad \frac{\Gamma \vdash \mathit{query} \quad \Gamma \vdash \mathit{script}}{\Gamma \vdash \texttt{where}\ \mathit{query}\ \mathit{script}}$$

$$\frac{\Gamma \vdash \mathit{term} : \mathit{xsd{:}anyURI} \quad \Gamma \vdash \mathit{script}}{\Gamma \vdash \texttt{from named}\ \mathit{term}\ \mathit{script}} \qquad \frac{\Gamma, \texttt{\$x} : \mathit{type} \vdash \mathit{script}}{\Gamma \vdash \texttt{select \$x}\ : \mathit{type}\ \mathit{script}}$$

$$\frac{\Gamma \vdash \mathit{script}}{\Gamma \vdash \texttt{do}\ \mathit{script}} \qquad \frac{}{\Gamma \vdash \texttt{success}}$$

Type System: terms and expressions

$$\frac{\vdash type_0 \leq type_1}{\Gamma, \$x : type_0 \vdash \$x : type_1} \qquad \frac{\vdash range(\mathcal{S}(uri)) \leq type}{\Gamma \vdash uri : type} \qquad \frac{\vdash xsd{:}integer \leq datatype}{\Gamma \vdash integer : datatype}$$

$$\frac{}{\Gamma \vdash decimal : xsd{:}decimal} \qquad \frac{}{\Gamma \vdash string : xsd{:}string} \qquad \frac{}{\Gamma \vdash dateTime : xsd{:}dateTime}$$

$$\frac{}{\Gamma \vdash \texttt{now} : xsd{:}dateTime} \qquad \frac{\Gamma \vdash expr_1 : datatype \quad \Gamma \vdash expr_2 : datatype \quad \vdash datatype \leq xsd{:}decimal}{\Gamma \vdash expr_1 + expr_2 : datatype}$$

$$\frac{\Gamma \vdash expr : datatype}{\Gamma \vdash \texttt{str}(expr) : xsd{:}string} \qquad \frac{\Gamma \vdash expr : datatype \quad \vdash datatype \leq xsd{:}decimal}{\Gamma \vdash \texttt{abs}(expr) : datatype}$$

$$\frac{\Gamma \vdash expr : xsd{:}string}{\Gamma \vdash \texttt{regex}\,(expr, regex)} \qquad \frac{\Gamma \vdash expr : xsd{:}string}{\Gamma \vdash \texttt{langMatches}(expr, lang\text{-}range)}$$

$$\frac{\Gamma \vdash expr_1 : datatype \quad \Gamma \vdash expr_2 : datatype}{\Gamma \vdash expr_1 = expr_2} \qquad \frac{\Gamma \vdash expr_1 : datatype \quad \Gamma \vdash expr_2 : datatype}{\Gamma \vdash expr_1 < expr_2}$$

$$\frac{\Gamma \vdash boolean_0 \quad \Gamma \vdash boolean_1}{\Gamma \vdash boolean_0 \;\&\&\; boolean_1} \qquad \frac{\Gamma \vdash boolean_0 \quad \Gamma \vdash boolean_1}{\Gamma \vdash boolean_0 \;||\; boolean_1} \qquad \frac{\Gamma \vdash boolean}{\Gamma \vdash !boolean}$$

Minimal Type Inference

- Suppose that we have the following untyped program:

```
⊢ do  select $g, $x, $y
      where
          graph $g {$x rdfs:label $y}
          langMatches($y, ru-*)
      from named $x
```

- Firstly, use the algorithmic type system to generate constraints:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{G \leq \mathit{xsd:anyURI}}}{\$g : G \vdash g : \mathit{xsd:anyURI}} \quad \dfrac{\dfrac{X \leq \mathit{xsd:anyURI}}{\$x : X \vdash \$x : \mathit{xsd:anyURI}} \quad \dfrac{Y \leq \mathcal{S}(\mathit{rdfs:label})}{\$y : Y \vdash \$x : \mathit{xsd:string}}}{\$x : X, \$y : Y \vdash \$x \ \mathit{rdfs:label} \ \$y}}{\$g : G, \$x : X, \$y : Y \vdash \texttt{graph}\ \$g\ \{\$x \ \mathit{rdfs:label} \ \$y\}} \quad \dfrac{\dfrac{}{Y \leq \mathit{xsd:string}}}{\$y : Y \vdash \$y : \mathit{xsd:string}}}{\$g : G, \$x : X, \$y : Y \vdash \texttt{graph}\ \$g\ \{\$x \ \mathit{rdfs:label} \ \$y\}\ \texttt{langMatches}(\$y, \texttt{ru-*})} \quad \dfrac{\dfrac{}{X \leq \mathit{xsd:anyURI}}}{\$x : X \vdash \$x : \mathit{xsd:anyURI}}}{\$x : X \vdash \texttt{from named}\ \$x}}{\$g : G, \$x : X, \$y : Y \vdash \texttt{where graph}\ \$g\ \{\$x \ \mathit{rdfs:label} \ \$y\}\ \texttt{langMatches}(\$y, \texttt{ru-*})\ \texttt{from named}\ \$x}}{\vdash \texttt{select}\ \$g : G, \$x : X, \$y : Y\ \texttt{where graph}\ \$g\ \{\$x \ \mathit{rdfs:label} \ \$y\}\ \texttt{langMatches}(\$y, \texttt{ru-*})\ \texttt{from named}\ \$x}}{\vdash \texttt{do select}\ \$g : G, \$x : X, \$y : Y\ \texttt{where graph}\ \$g\ \{\$x \ \mathit{rdfs:label} \ \$y\}\ \texttt{langMatches}(\$y, \texttt{ru-*})\ \texttt{from named}\ \$x}$$

Minimal Type Inference

- Secondly, find the most general solution to the generated constraints:

$$X \leq \textit{xsd:anyURI} \qquad Y \leq \textit{xsd:string} \qquad G \leq \textit{xsd:anyURI}$$

- Substituting type variables for the the most general solution results in a well typed annotated script:

$\vdash$ `do` `select` `$g`: *xsd:anyURI*, `$x`: *xsd:anyURI*, `$y`: *xsd:string*
`where`
`graph` `$g` {`$x` *rdfs:label* `$y`}
`langMatches`(`$y`, `ru-*`)
`from named` `$x`

A Larger Example

```
from named dbpedia:Almaty
select $almalat: xsd:decimal, $almalong: xsd:decimal
where
    graph dbpedia:Almaty {dbpedia:Almaty geo:lat $almalat}
    graph dbpedia:Almaty {dbpedia:Almaty geo:long $almalong}
from named dbpedia:Kazakhstan
do  select $loc: xsd:anyURI
    where
        graph dbpedia:Kazakhstan {$loc dbp:location dbpedia:Kazakhstan}
    from named $loc
    select $lat: xsd:decimal, $long: xsd:decimal
    where
        graph $loc {$loc geo:lat $lat}
        graph $loc {$loc geo:long $long}
        haversine($lat, $long, $almalat, $almalong) < 100
    do  select $person: xsd:anyURI
        where
            graph $loc {$person dbp:birthPlace $loc}
        from named $person
```

Dereference data about people born in places in Kazakhstan less than 100km from Almaty.

Algorithmic Typing, Subject Reduction and Type Safety

- **Algorithmic typing** proves that we have a deterministic type system hence our type inference algorithm terminates.

Theorem

The transitivity and subsumption rules are admissible.

- **Subject reduction** proves that a well typed system will still be well typed while it is being executed. Executions are analysed using an operational semantics.

Theorem

If $\vdash system_1$ *and* $system_1 \longrightarrow system_2$*, then* $\vdash system_2$*.*

- **Type safety** proves that a well type system will not throw basic runtime errors.

Theorem

If $\vdash$ *script and script* $\longrightarrow$ *script*$'$*, then script*$'$ *does not contain an error.*

A process calculus approach

$A :=$	$(a\ a\ v)$	triple
$\mid$	$\overline{A}$	complement
$\mid$	$A \otimes A$	tensor
$\mid$	A, A	par
$\mid$	$\bot$	nothing
$\mid$	success	true

$\phi :=$	success	true
$\mid$	0	false
$\mid$	$\phi \oplus \phi$	or
$\mid$	$\phi \otimes \phi$	and
$\mid$	$\neg\phi$	not
$\mid$	$\ldots$	etc.

a URI $\quad$ v literal or URI

x variable for URI or literal

$U :=$	A	label
$\mid$	ϕ	filter
$\mid$	$U \oplus U$	choice
$\mid$	$U \otimes U$	tensor
$\mid$	U, U	par
$\mid$	$\exists x.U$	exists
$\mid$	$*U$	iteration
$\mid$	τU	delay
$\mid$	$U \| U$	interleave

An example atomic commitment

Let *dbp:SoccerPlayer* $\leq$ *dbp:Athlete*.

$$\begin{array}{l}\overline{(\mathit{Armstrong}\ \mathit{foaf{:}name}\ \texttt{"Joe Armstrong"})\|} \\ \overline{(\mathit{Armstrong}\ \mathit{rdf{:}type}\ \mathit{dbp{:}SoccerPlayer})\|} \\ \exists a.\left(\begin{array}{l} \exists z.\left(\begin{array}{l} \left(\begin{array}{l} \exists x, y.\left(\begin{array}{l} \left|(a\ \mathit{foaf{:}givenName}\ x)\right| \\ \left|(a\ \mathit{foaf{:}familyName}\ y)\right| \\ (z = x + \texttt{" "} + y) \end{array}\right) \\ \oplus \\ \left|(a\ \mathit{foaf{:}name}\ z)\right| \end{array}\right) \\ (z \in \texttt{"J.* Armstrong"}) \end{array}\right) \\ \left(\begin{array}{l} \left|(a\ \mathit{rdf{:}type}\ \mathit{dbp{:}Athlete})\right| \\ \oplus \\ \left|(a\ \mathit{rdf{:}type}\ \mathit{dbp{:}Artist})\right| \end{array}\right) P \end{array}\right) \end{array}$$

The above process commits to the following process.

$$\left(\begin{array}{l} \overline{(\mathit{Armstrong}\ \mathit{foaf{:}name}\ \texttt{"Joe Armstrong"})\|} \\ \overline{(\mathit{Armstrong}\ \mathit{rdf{:}type}\ \mathit{dbp{:}SoccerPlayer})\|} \\ P\left\{{}^{\mathit{Armstrong}}/_{a}\right\} \end{array}\right)$$

Atomic commitments

(reflexivity)

$$P \rhd P$$

(tensor)

$$\frac{P, U \rhd P' \quad Q, V \rhd Q'}{P, Q, (U \otimes V) \rhd P' \otimes Q'}$$

(par)

$$\frac{P \rhd U \otimes P' \quad Q \rhd V \otimes Q'}{P, Q \rhd (U, V) \otimes P' \otimes Q'}$$

(complement)

$$\frac{P \rhd Q \otimes A}{P, \overline{A} \rhd Q}$$

(choose left)

$$\frac{P, U \rhd Q}{P, (U \oplus V) \rhd Q}$$

(choose right)

$$\frac{P, V \rhd Q}{P, (U \oplus V) \rhd Q}$$

(exists)

$$\frac{P, U\{^{V}/_{x}\} \rhd Q}{P, \exists x.U \rhd Q}$$

(filter)

$$\frac{\vDash \phi}{\phi \rhd \mathtt{success}}$$

(weakening)

$$*U \rhd \mathtt{success}$$

(dereliction)

$$\frac{P, U \rhd Q}{P, *U \rhd Q}$$

(contraction)

$$\frac{P, (*U \otimes *U) \rhd Q}{P, *U \rhd Q}$$

(interact)

$$\frac{P, Q, R \rhd S}{P, (Q \parallel R) \rhd S}$$

(left action)

$$\frac{P, (Q \otimes \tau R) \rhd S}{P, (Q \parallel R) \rhd S}$$

(right action)

$$\frac{P, (\tau Q \otimes R) \rhd S}{P, (Q \parallel R) \rhd S}$$

Definition (Bisimulation)

Let $P \xrightarrow{A} P' \equiv P \rhd A \otimes \tau P$.

If $P \sim Q$ and $P \xrightarrow{A} P'$ then there exists some Q' such that $Q \xrightarrow{A} Q'$ and $P' \sim Q'$.

Definition (Contextual equivalence)

Contextual equivalence, written $\simeq$, is the greatest symmetric, reduction closed, context closed relation.
A relation $\mathcal{R}$ is reduction closed iff $P \mathcal{R} Q$ and $P \to P'$ then there exists some Q' such that $Q \to Q'$ and $P' \mathcal{R} Q'$.
A relation $\mathcal{R}$ is context closed iff $P \mathcal{R} Q$ yields that $\mathcal{C}P \mathcal{R} \mathcal{C}Q$, for all contexts $\mathcal{C}$.

Lemma (Bisimulation is reduction closed)

If $P \xrightarrow{success} Q$ *then* $P \rhd Q$.

Lemma (Bisimulation is context closed)

If $P \sim Q$ *and* $\mathcal{C}$ *is a context, then* $\mathcal{C}P \sim \mathcal{C}Q$.

Theorem (Bisimulation is a contextual equivalence)

If $P \sim Q$ *then* $P \simeq Q$.

A sound algebra for queries

- $(P, \phi, *, \otimes, \oplus, \mathtt{success}, 0, \neg, \leq)$ forms a Kleene algebra with tests.
- Existential quantification is the least upper bound of substitutions for a variable.
- Iteration is the least upper bound of powers of processes.
- Least upper bounds distribute over tensor.
- $\left(A, \otimes, \mathbf{,}, \mathtt{success}, \bot, \overline{(.)}, \leq\right)$ is a model of multiplicative linear logic.
- $(P, \mathbf{,}, 0)$ is a commutative monoid.
- $(P, Q) \leq P \parallel Q \quad (P \otimes \tau Q) \leq P \parallel Q \quad (\tau P \otimes Q) \leq P \parallel Q \quad \tau(P \parallel Q) \leq \tau P \otimes \tau Q.$

Theorem (Soundness of the algebra)

If $U = V$ in the algebra, then $U \sim V$.

Facebook Open Graph protocol http://ogp.me/ns#

```
og:url a rdf:Property ;
  rdfs:label "url"@en-US ;
  rdfs:comment "The canonical URL of your object that will be used as its permanent ID in the graph, e.g., \"http://www.imdb.com
  rdfs:seeAlso dc:identifier,
    foaf:homepage ;
  rdfs:isDefinedBy og: ;
  rdfs:range ogc:url .
og:type a rdf:Property ;
  rdfs:label "type"@en-US ;
  rdfs:comment "The type of your object, e.g., \"movie\".  Depending on the type you specify, other properties may also be requi
  rdfs:seeAlso rdf:type ;
  rdfs:isDefinedBy og: ;
  rdfs:range ogc:string .
og:title a rdf:Property ;
  rdfs:label "title"@en-US ;
  rdfs:comment "The title of the object as it should appear within the graph, e.g.,  \"The Rock\"."@en-US ;
  rdfs:subPropertyOf rdfs:label ;
  rdfs:isDefinedBy og: ;
  rdfs:range ogc:string .
og:locale a rdf:Property ;
  rdfs:label "locale"@en-US ;
  rdfs:comment "A Unix locale in which this markup is rendered."@en-US ;
  rdfs:isDefinedBy og: ;
  rdfs:range ogc:string .
og:image a rdf:Property ;
  rdfs:label "image"@en-US ;
  rdfs:comment "An image URL which should represent your object within the graph."@en-US ;
  rdfs:seeAlso foaf:depiction ;
  rdfs:isDefinedBy og: ;
  rdfs:range ogc:url .
<http://ogp.me/ns#image:width> a rdf:Property ;
  rdfs:label "image width"@en-US ;
  rdfs:comment "The width of an image."@en-US ;
  rdfs:isDefinedBy og: ;
  rdfs:range ogc:integer_str .
<http://ogp.me/ns#image:height> a rdf:Property ;
  rdfs:label "image height"@en-US ;
  rdfs:comment "The height of an image."@en-US ;
  rdfs:isDefinedBy og: ;
  rdfs:range ogc:integer_str .
<http://ogp.me/ns#image:secure_url> a rdf:Property ;
  rdfs:label "image secure url"@en-US ;
```

Conclusion

- A high level domain specific language for processes that consume Linked Data.
- Syntax of language is familiar for programmers that use W3C recommendations.
- Simple type system reflects W3C recommendations.
- The type system mixes static and dynamic typing.
- The type system is algorithmic; hence can be used for type inference.
- Facebook's Open Graph protocol demands types at the level we deliver.
- Scope for further investigation: operational semantics that reflect weak consistency of triple stores, stochastic analysis of data sources to keep data relevant, implementing a verified scripting language. . .