A Formal Definition of RESTful Semantic Web Services

Antonio Garrote Hernández
María N. Moreno García
Outline

- Motivation
- Resources and Triple Spaces
- Resources and Processes
- RESTful Semantic Resources
- Example
- Conclusions and Future Work
Motivation

• Web development is becoming increasingly API-centric:
  - Rich Internet Applications
  - Mobile platforms
  - Mash-ups, OAuth, Microformats
Motivation

• Open issues in current web development:
  - Description of data: graph of social objects
  - Data interoperability
  - ...
Motivation

• Semantic Web technologies have the potential to solve some of these issues:
  - Open world semantics, monotonic reasoning
  - Description vocabulary (OWL, RDFS)
  - Data model (RDF, RDFa), query (SPARQL) ...
Problem

- Semantic web adoption in regular web development is almost non-existent
  - A pragmatic approach for leveraging semantic technologies is required
  - RESTful web services can serve as the foundation for such an approach
Proposal

• A model for RESTful semantic distributed computation

• A formal definition of the model
Outline

• Motivation
• Resources as Triple Spaces
  • Resources and Processes
  • RESTful Semantic Resources
  • Example
• Conclusions and Future Work
Resources as Triple Spaces

- Generative Communication (Linda), TS computing [Gelernter, Fensel]

- **Resource**: Triple encoded graph stored in shared memory known as **triple space**

- Triple space operations:
  - Data manipulation
  - coordination primitives
GET http://test.com/resource.rdf

σ = http://test.com/resource
Resources and Triple Spaces

• Extended triple space operations:
  - atomic \textit{swap} operation modeling PUT requests
  - \textit{notify} operation as an additional coordination primitive
Resources and Triple Spaces

• Formalization:

\[
P ::= 0 | T | P|P | !P | if T ? P.P | x ::= T
\]
\[
T ::= \text{rd}(\theta_i, p) | \text{in}(\theta_i, p) | \text{out}(\theta_i, v) | \text{swap}(\theta_i, p, v) | \text{rdb}(\theta_i, p) | \text{inb}(\theta_i, p) | \text{notify}(\theta_i, \rho, v)
\]

\[
\theta ::= \{ \text{triple spaces} \}
\]
\[
\rho ::= \{ \text{in, out} \}
\]
\[
\mu ::= \{ \text{URIs} \}
\]
\[
\lambda ::= \{ \text{literals} \}
\]
\[
p ::= \{ \text{patterns} \}
\]
\[
v ::= \{ \text{values} \} = \{ \mu \} \cup \{ \lambda \} \cup <p, v> \cup <p, \theta_i>
\]
Resources and Triple Spaces

• Formalization:

1. \frac{P \rightarrow P'}{P | Q \rightarrow P' | Q}
2. \frac{P \rightarrow P', \text{if } P \equiv Q \text{ and } P' \equiv Q'}{Q \rightarrow Q'}
3. !P.Q \rightarrow Q | P
4. \frac{rd(\theta_i,p).P}{rd(\theta_i,p).P \xrightarrow{<p,\theta_i>} P}
5. \frac{in(\theta_i,p).P}{rd(\theta_i,p).P \xrightarrow{<p,\theta_i>} P, \theta_i = \theta_i - <p,\theta_i>}
6. \frac{out(\theta_i,v).P}{out(\theta_i,v).P \xrightarrow{\overline{v}} P, \theta_i = \theta_i \cup v}
7. \frac{swap(\theta_i,p,v).P}{swap(\theta_i,p,v).P \xrightarrow{<p,\theta_i>, \overline{v}} P, \theta_i = \theta_i - <p,\theta_i> \cup v}
8. \frac{out(\theta_i,v).Q}{out(\theta_i,v).Q \xrightarrow{\overline{v}, <p,v>} P | Q}
9. \frac{notify(\theta_i, out,p).P | out(\theta_i,v).Q}{notify(\theta_i, out,q).P | in(\theta_i,p).Q \xrightarrow{<p,v>, <q, <p,v>}} P | Q}
10. if T.P.Q \xrightarrow{\theta}, if T.P.Q \xrightarrow{\theta} P
Resources and Triple Spaces

• TS useful for describing RESTful web resources as data manipulation

• Problems:
  - Creation, destruction of triple spaces (POST, DELETE operations)
  - Creation of new names
  - Blocking triple space operations
Outline

• Motivation
• Resources as Triple Spaces
• Resources as Processes
  • RESTful Semantic Resources
  • Example
• Conclusions and Future Work
Resources and Processes

- Pi-Calculus [Milner]
- HTTP request can be modeled as a message sent from a process (client) to another process (resource)
- URI can be modeled as a channel between processes
- The URIs inside the RDF graph returned as a response grants the process access to new communication channels
Resources and Processes

POST http://test.com/resources

201 http://test.com/resources/test
Resources and Processes

http://test.com/resources

{req,post,data}
{resp,201,rdf graph}

spawn!
Resources and Processes

• Formalization:

\[
P ::= 0 | T | M | P | P | !P | if T? P.P | x ::= T |
\]
\[
new \mu in P
\]
\[
M ::= \frac{\text{req}(\mu)[m, p, v]}{[m, p, v]\text{req}(\mu)} | \frac{\text{resp}(\mu)[c, v]}{[c, v]\text{resp}(\mu)}
\]
\[
m ::= \{\text{get, post, put, delete}\}
\]
\[
c ::= \{200, 201, 404, 401\}\]
Resources and Processes

- Formalization:

\[
\begin{align*}
\frac{\text{req}(\mu)[m,p,v]}{P \rightarrow P', Q} & , \frac{\text{resp}(\mu)[c,v]}{P \rightarrow P', Q} \\
\frac{\text{req}(\mu)[m,p,v]}{P} & , \frac{\text{resp}(\mu)[c,v]}{P} \\
\frac{\text{req}(\mu)[m,p,v]}{P} & , \frac{\text{resp}(\mu)[c,v]}{P} \\
\frac{[m,p,v]\text{req}(\mu)}{P} & , \frac{\text{resp}(\mu)[c,v]}{P}
\end{align*}
\]
Resources and Processes

\( \text{uri1} ::= \text{http://test.com/resources} \)
\( \text{uri2} ::= \text{http://test.com/resources/test} \)

\( \text{Ag} ::= \text{req(uri1)[post,0,Data]}. \)
\( \hspace{1cm} \[201, \{\text{Uri2,type,resource}\}]\text{resp(uri1)}. \)
\( \hspace{1cm} \text{req(Uri2),[get,*,0]. [200,Data]}\text{resp(Uri2)} \)

\( \text{Res} ::= \text{new uri2 in ([post,0,Data]}\text{req(uri1).Res2(uri2,Data)}!. \)
\( \hspace{1cm} \text{resp(uri1)[401,uri2]}).\text{Res} \)

\( \text{Res2(uri2, D)} ::= [\text{post, *,0]}\text{req(uri2).resp(uri2)[200,D]}\).\text{Res2} \)
Resources and Processes

- Message passing, channels and processes useful for modeling process view of HTTP computations

- Problems:
  - Modeling the state of the resource is less intuitive
Outline

• Motivation
• Resources as Triple Spaces
• Resources as Processes
• RESTful Semantic Resources
  • Example
  • Conclusions and Future Work
RESTful semantic resources

- Triple spaces and Pi-calculus are complementary formalisms
- Combination of both formalisms suitable for describing RESTful distributed computation
RESTful semantic resources

- Computation takes place in certain “computational places”
  - Nodes executing a web services API
  - A web browser
  - A mobile phone
RESTful semantic resources

• Inside each “computational place” a set of processes are executed and a certain number of triple spaces are shared

• Communication between processes inside a computational place is tuple space based
RESTful semantic resources
RESTful semantic resources

• Certain processes in “computational places” have an associated URI and process incoming messages according to REST semantics

• Communication between “computational places” is message passing based

• URIs can be transferred between triple spaces in different “computational places”
RESTful semantic resources
RESTful semantic resources

- RESTful semantic web resource:
  - Process being executed in a computational place
  - Associated URI
  - Receives HTTP messages through URI
  - Manipulates TS according to REST semantics
RESTful semantic resources

• Formalization:

\[
R_{REST}(\theta, \mu) ::= \ [m, v, p] req(\mu) \cdot \text{if } m = \text{get } ? R_{get}(\theta, \mu). \\
\text{if } m = \text{post } ? R_{post}(\theta, \mu). \\
\text{if } m = \text{put } ? R_{put}(\theta, \mu). \\
\text{if } m = \text{delete } ? R_{delete}(\theta, \mu). \\
\text{resp(\mu)[406, 0] \cdot R_{REST}(\theta, \mu)}
\]

\[
R_{get}(\theta, \mu) ::= \ x ::= rd(\theta, p). \text{resp(\mu)[200, x]} \cdot R_{REST}
\]

\[
R_{post}(\theta, \mu) ::= \ new \nu \ \text{in out(\theta, < p, \nu >)} \cdot !R(\theta, \nu). \\
\text{resp(\mu)[201, < p, \nu >] \cdot R_{REST}}
\]

\[
R_{put}(\theta, \mu) ::= \ swap(\theta, p, v). \text{resp(\mu)[200, v]} \cdot R_{REST}
\]

\[
R_{delete}(\theta, \mu) ::= \ in(\theta, p_\mu). \text{resp(\mu)[200, 0]} \cdot 0
\]
Outline

• Motivation
• Resources and Triple Spaces
• Resources and Processes
• RESTful Semantic Resources
• Example

• Conclusions and Future Work
Semantic Bespin

Twitter

(GET) (hRESTS)

Bespin JS app

POST

status

POST

Rails Backend

TS

(POST) (hRESTS)

notification

http://github.com/antoniogarrote/semantic_rest
Outline

• Motivation
• Resources and Triple Spaces
• Resources and Processes
• RESTful Semantic Resources
• Example
• Conclusions and Future Work
Conclusions

- Clear definition of data related and process related aspects of RESTful computations
- Introduces the notion of computational place as an aggregation of resources or processes with RESTful interfaces
- Modeling decoupled from actual implementation
Conclusions

• Benefits:
  - Composition of services trivially modeled as a sequence of messages in the calculus
  - It is possible to model complex interaction scenarios triggering blocking TS operations (notify, rdb, outb) as a side effect of a HTTP message
Conclusions

• Benefits:
  - Use of semantic metadata offers an uniform model for data shared among resources
  - Shared operations for querying and manipulating resources
  - Incremental description of resources
Future work

• Blocking operations
  - blocking communication primitives useful for coordination between agents and resources
  - avoid polling
  - restricted to triple space operations
  - extension to the HTTP interface
Future work

• Type system
  - Types can be assigned to resources based on the ontology primitives used in the description of the resource
  - OWL, RDFS, RDF entailment regimes
  - Importance for the discovery of resources
Future work

- Implementation
  - Experimental implementation with blocking operations
  - hRESTS, RabbitMQ, OpenSesame, Erlang OTP
  - http://github.com/antoniogarrotoe/Plaza