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 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 nonexistent
  - 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
Resources and Triple Spaces

GET http://test.com/resource.rdf

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

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

• Formalization:

\[
\begin{align*}
P & ::= \ 0 \mid T \mid P \mid P \mid P \mid P \mid P \mid x ::= T \\
T & ::= \ rd(\theta_i, p) \mid in(\theta_i, p) \mid out(\theta_i, v) \mid swap(\theta_i, p, v) \mid \\
& \quad rdb(\theta_i, p) \mid inb(\theta_i, p) \mid notify(\theta_i, \rho, v)
\end{align*}
\]

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

- Formalization:

(1) \[ \frac{P \rightarrow P'}{P \mid Q \rightarrow P' \mid Q} \]

(2) \[ \frac{P \rightarrow P'}{Q \rightarrow Q'} \text{ if } P \equiv Q \text{ and } P' \equiv Q' \]

(3) \[ !P.Q \rightarrow Q \mid P \]

(4) \[ \frac{\text{rd}(\theta_i,p),P}{\text{rd}(\theta_i,p),P \frac{<p,\theta_i>}{P}} \]

(5) \[ \frac{\text{in}(\theta_i,p),P}{\text{rd}(\theta_i,p),P \frac{<p,\theta_i>}{P}, \theta_i = \theta_i - <p,\theta_i>} \]

(6) \[ \frac{\text{out}(\theta_i,v),P}{\text{out}(\theta_i,v),P \frac{\overline{v}}{P, \theta_i = \theta_i \cup v}} \]

(7) \[ \frac{\text{swap}(\theta_i,p,v),P}{\text{swap}(\theta_i,p,v),P \frac{<p,\theta_i>,\overline{v}}{P, \theta_i = \theta_i - <p,\theta_i> \cup v}} \]

(8) \[ \frac{\text{out}(\theta_i,v),Q}{\text{notify}(\theta_i,\text{out},p),P \mid \text{out}(\theta_i,v),Q \frac{\overline{v},<p,v>}{P \mid Q}} \]

(9) \[ \frac{\text{in}(\theta_i,p),Q}{\text{notify}(\theta_i,\text{in},q),P \mid \text{in}(\theta_i,p),Q \frac{<p,v>,<q,<p,v>}{P \mid Q}} \]

(10) \[ \frac{\text{if } T.P.Q}{\overline{0}}, \frac{\text{if } T.P.Q}{\overline{v}} \]
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

{req, post, data}
http://test.com/resources
{resp, 201, rdf graph}

spawn!
Resources and Processes

• Formalization:

\[ P ::= 0 | T | M | P | P | !P | i f \ T ? P . P | x ::= T | \]
\[ \text{new } \mu \text{ in } P \]

\[ M ::= \text{req}(\mu)[m, p, v] | [m, p, v]\text{req}(\mu) | \text{resp}(\mu)[c, v] | [c, v]\text{resp}(\mu) \]

\[ m ::= \{\text{get, post, put, delete}\} \]
\[ c ::= \{200, 201, 404, 401\} \]
Resources and Processes

• Formalization:

\[ P \xrightarrow{\text{req}(\mu)[m,p,v]} P', Q \xrightarrow{[m,p,v]\text{req}(\mu)} Q' \]

\[ P \| Q \rightarrow P'|Q' \]

\[ P \xrightarrow{\text{resp}(\mu)[c,v]} P', Q \xrightarrow{[c,v]\text{resp}(\mu)} Q' \]

\[ P \| Q \rightarrow P'|Q' \]

\[ \frac{\text{req}(\mu)[m,p,v].P}{\text{req}(\mu)[m,p,v].P \xrightarrow{*}[c,v]\text{resp}(\mu).Q} \]

\[ \frac{[m,p,v]\text{req}(\mu).P}{[m,p,v]\text{req}(\mu).P \xrightarrow{*}\text{resp}(\mu)[c,v].Q} \]
Resources and Processes

\[ \text{uri1} ::= \text{http://test.com/resources} \]
\[ \text{uri2} ::= \text{http://test.com/resources/test} \]

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

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

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

- Message passing, channels and processes useful for modeling dynamic aspects 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 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

Diagram:
- TS
- POST uri1
- notify
- uri2
- PUT
- GET uri2
- rd
- out
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:

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

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

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

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

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

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

Twitter

(\texttt{hRESTS})

GET

TS

\textbf{Semantic Bespin}

http://github.com/antoniogarroton/semmatic_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/antoniogarroto/Plaza