

A Logic Programming Model for Web Resources

Giulio Piancastelli Andrea Omicini

`{giulio.piancastelli, andrea.omicini}@unibo.it`

ALMA MATER STUDIORUM—Università di Bologna

4th International Conference on Web Information
Systems and Technologies (WEBIST 2008)
Funchal (Madeira), May 4-7, 2008

Background

A proper description, understanding, formalisation and divulgement of the World Wide Web architectural principles and design criteria has been achieved only recently by

- the Representational State Transfer (REST) architectural style for distributed hypermedia systems
- the Resource-Oriented Architecture (ROA) as a REST-based set of guidelines and best practices to create web applications

Both REST and ROA

- focussed on the *resource* as the main data abstraction, defined as any conceptual target of a hypertext reference
- prescribed communication amongst resources to occur through a *uniform interface* by transferring a *representation* of a resource's current state.

However, a resource programming model is still missing

Motivation

Find a programming model for the World Wide Web abstractions recently described by REST and ROA

- by avoiding overused abstractions such as *page*, *controller*, *service*
- by founding the model on a different programming paradigm than the imperative paradigm subsumed by procedural and object-oriented languages

The Advantages of Logic Programming

We adopt the logic declarative paradigm to define our model for resource-oriented programming for two main reasons:

- 1 the mapping between logic programming elements and the World Wide Web computation model is natural and straightforward
- 2 the foundational idea of treating programs as data leads to resource representation that are directly executable
 - abstractions stay simple without reducing their own expressiveness and computational power
 - novel possibilities are opened for dynamic extension and modification of resource behaviour at runtime

From Resources to Web Logic Resources

Resource Definition

Any conceptual target of an hypertext reference

Resources have:

- a name, i.e. an URI (RFC 2396)
- data representing their state
- behaviour to perform actions such as changing their state, building up their representations, managing interaction with other resources

Web Logic Resources apply the following mapping:

names are mapped onto logic atoms, much as it happens for logic module names

data is represented by logic facts

behaviour is encoded in logic rules

Resource-Oriented Architecture Addressability

- URIs should be descriptive
- URIs should have a definite structure, and that structure should vary in predictable ways
- Each name encompasses other resource names, and ultimately the name of the resource associated with the domain at the URI root

Bookshelf Sharing example

The name of a book on the shelf of the jdoe user

`http://example.com/users/jdoe/shelf/5`

encompasses the names

`http://example.com/users/jdoe/shelf`

`http://example.com/users/jdoe`

`http://example.com/users`

`http://example.com`

From Resources to Contexts

- 1 The resource naming structure, e.g.

`http://example.com/sales/2004/Q4`

suggests that each resource does not exist in isolation, but lives in an information *context* composed by the resources associated to the names encompassed by the name of that resource

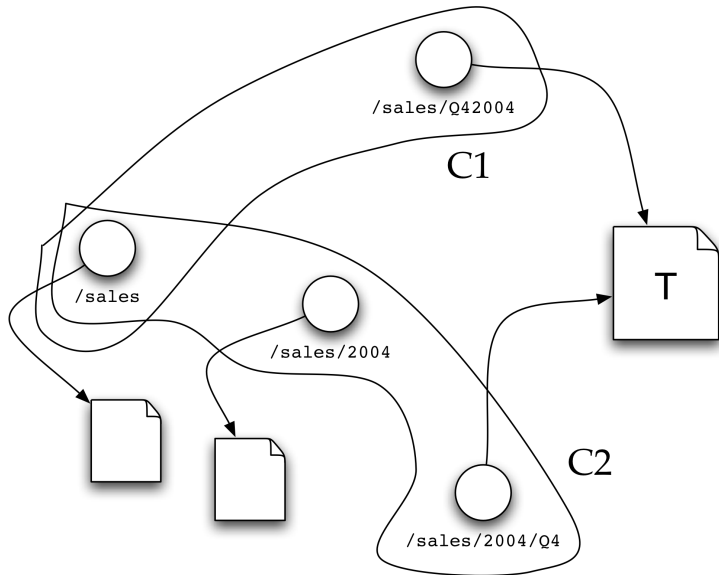
- 2 Since more than one name can identify the same resource, e.g.

`http://example.com/sales/2004/Q4`

`http://example.com/sales/Q42004`

the context of a resource has to be associated with its name rather than directly with the resource itself

`http://example.com/sales/Q42004`



`http://example.com/sales/2004/Q4`

Web Logic Resources, revisited

The locus of computation is no more the single logic program representing a resource: the notion of context is introduced in the World Wide Web domain

Definition

Given a resource R with a name $N(R)$ such that

$$N(R) \subseteq N(R_1) \subseteq \dots \subseteq N(R_n)$$

then, the associated context $C(R)$ is generated by the following composition

$$C(R) = T(R) \cdot T(R_1) \cdot \dots \cdot T(R_n)$$

The World Wide Web Computation Model

According to REST and ROA, the World Wide Web computation revolves around transactions in the HyperText Transfer Protocol

- HTTP (RFC 2616) is a document oriented protocol aimed at transferring *representations* of a resource current state
- HTTP requests contain two key computation elements:
 - method information** indicates how the sender expects the receiver to process the request
 - scope information** indicates on which part of the data set the receiver should operate the method
- HTTP responses typically contain the representation of the target resource (new) state as the result of the computation

Logic Programming Computation Model

A computation in logic programming is a deduction of consequences of a set of facts and rules defining relationships between entities.

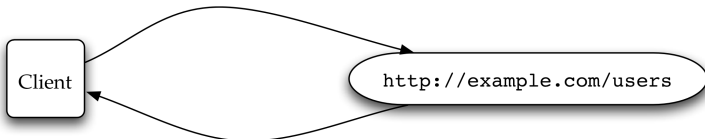
- Sets of facts and rules are called *logic theories*
- *Logic queries* are used to trigger the application of deduction rules on a theory

Web Logic Computation Model

- 1 Each HTTP request gets translated to represent a deduction by
 - retaining the *scope information* to indicate the target *logic theory*, and
 - mapping the *method information* onto a *logic goal*
- 2 The computation takes place in the *context* associated to the resource target of the request
- 3 The information resulting from goal solution is translated to a suitable *representation* in order to be sent back as the payload of the HTTP response

```
POST /users HTTP/1.1
Host: example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 15
```

```
name=jdoe&sex=f
```



```
HTTP/1.1 201 Created
Location: http://example.com/users/jdoe
```

Goal

```
'http://example.com/users' : post(Request,  
                                Response,  
                                View).
```

Computation

```
post(Request, Response, _) :-  
    create_user(Request),  
    param(Request, (name, User)),  
    user_url(User, Url),  
    header(Response, (location, Url)),  
    status(Response, (201, created)).
```

Context Traversal

Being the context $C(R)$ the composition of a number of theories, the computation is carried on so that the query G is asked in turn to each theory.

- The goal fails if no solution is found in any theory;
- the goal succeeds as soon as it is solved using the knowledge base contained in a theory $T(R_i)$.

When the goal G gets substituted by the subgoals $S_j(G)$ of the matching rule in the theory, the computation proceeds from the context of the resource R_i rather than being restarted from the original context.

```
GET /jdoe/shelf/biology HTTP/1.1  
Host: example.com
```



/jdoe/shelf/biology

```
get/3  
pick_biology_books/1
```

```
% ...  
pick_biology_books(Books),  
% ...
```

(1)

/jdoe/shelf/biology

```
get/3  
pick_biology_books/1
```

```
parent_id(Shelf),  
pick_books(Books, Shelf,  
            category(biology)).
```

(2)

/jdoe/shelf/biology

get/3
pick_biology_books/1

% ...
pick_books(Books, Shelf,
category(biology)).

/jdoe/shelf

/jdoe

/

pick_books/3
filter/3

/jdoe/shelf/biology

get/3
pick_biology_books/1

/jdoe/shelf

/jdoe

/

pick_books/3
filter/3

% ...
filter(AllBooks, C, Books).

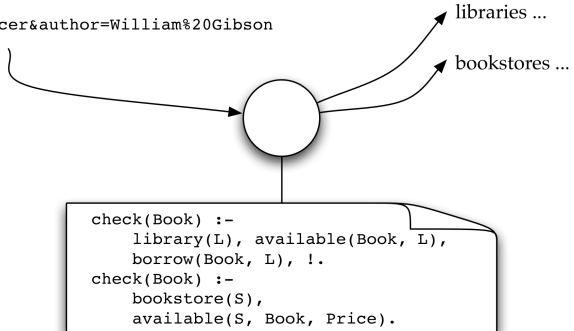
Dynamic Resource Behaviour

Resource behaviour can be regarded as dynamic under two independent aspects:

- ① Two or more URIs can be associated to the same resource at any point in time: thus, a resource may live in two different contexts at the same time and feature different behaviour according to the context where the computation takes place
- ② Behavioural rules are expressed as first-class abstractions in logic programming languages, where programs can be treated as data and vice versa: the HTTP protocol allows changing resource data by means of the PUT method, so that it becomes possible to imagine behavioural changes of a contextualised resource at runtime

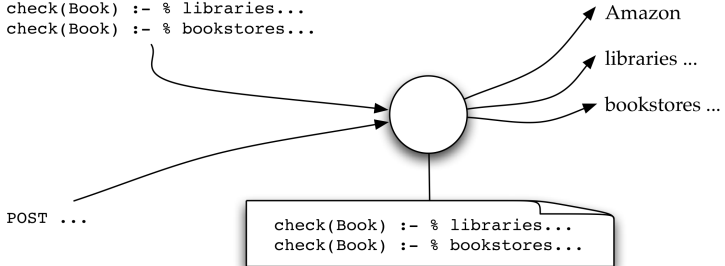
```
POST /users/jdoe/wishlist HTTP/1.1
Host: example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 41
```

```
title=Neuromancer&author=William%20Gibson
```



```
PUT /users/jdoe/wishlist HTTP/1.1
Host: example.com
Content-Type: text/plain
Content-Length: 164
```

```
check(Book) :-
    available(amazon, Book, Price),
    Price < Threshold,
    place_order(amazon, Book), !.
check(Book) :- % libraries...
check(Book) :- % bookstores...
```



```
PUT /users/jdoe/wishlist HTTP/1.1
Host: example.com
Content-Type: text/plain
Content-Length: 59
```

```
check(Book) :- % libraries...
check(Book) :- % bookstores...
```

POST ...

libraries ...

bookstores ...

```
check(Book) :- % Amazon...
check(Book) :- % libraries...
check(Book) :- % bookstores...
```