Prosper: A Framework for Extending Prolog Applications with a Web Interface

http://prospear.sourceforge.net/

Levente Hunyadi
Budapest University of Technology and Economics

document repository
server page documents within web server root

logic module repository
conventional Prolog modules in a dedicated directory; these are referenced by server page documents

web server

Prolog Server Pages core

- imports server page documents from repository
- converts documents into intermediate term representation
- evaluates documents
- generates output

expression language

extension to is/2 predicate
typed expressions
access to context data
extensible with custom functions

context module

- asserts request-specific context as dynamic predicates
- loads and saves persistent context (sessions)

extension infrastructure

predefined special elements

conditional evaluation
iteration, retrieving more solutions
assignment to local page variables
inserting values of expressions in output

user-defined elements

- defined by transformation rules (pre-processing-time and evaluation-time)
- no restrictions compared to predefined elements

FastCGI module

communicates with FastCGI stub in web server via FastCGI protocol (written in C)

Prolog Web Container

communicates with web server or user agent
manages worker threads
extracts HTTP context (e.g. GET or POST variables)

configuration
communication settings, workers configuration, extensions registration

Process of reply generation

server page document
regular XML in document repository

preprocessing

intermediate term
a syntax-verified representation

context

evaluation
unbound variables in intermediate term representation are instantiated based on context that derives from HTTP request

evaluated term
depends on context

content generation

output document
most commonly (X)HTML

Server page documents

Line 1
Link server page to logic module exporting *queens*/2. The second argument of *queens*/2 is a list of *piece*(Figure, Color, X, Y) terms.

Line 4
Bind elements in the XML namespace to a set of transformation rules. *board:chessboard* will correspond to the transformation rule *chessboard*/4 in module *boards*. The namespace to Prolog module binding is done in the configuration file.

Line 5 (left)
Introduce a local variable *N*. Local variables are assigned on instantiation and their scope is limited to descendant elements.

```
1 <html logic-module="queens"
2       xmlns="http://www.w3.org/1999/xhtml"
3       xmlns:psp="http://prosper.iit.bme.hu"
4       xmlns:board="http://prosper.iit.bme.hu/board">
5 <psp:assign var="N" expr="{atom_number(http_get(n))}">
6   <psp:for-each-else>
7     <psp:for-each function="queens(N)" iterator="L">
8       <p>Solution:</p>
9       <board:chessboard size="{N}" figures="L" />
10    </psp:for-each>
11    <psp:else>
12      <p>Sorry but there are no solutions.</p>
13    </psp:else>
14  </psp:for-each-else>
15 </psp:assign>
16 </html>
```

Line 5 (right)
Fetch the value of the HTTP GET context variable *n* and converts it into a number. Expression language terms like this are enclosed in {curly braces}.

Line 7
Iteration constructs are ideal to display repetitive content. The term in the attribute *function* is extended with an extra output argument when called. The local variable *L* gives access to the output argument.

Budapest University of Technology and Economics