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

Levente Hunyadi
Budapest University of Technology and Economics
Department of Automation and Applied Informatics

International Conference on Logic Programming
Porto, September 12, 2007

Goals

- Prolog: a general-purpose programming language
- but difficulty w.r.t. user interaction

Problem: no general method of web content generation with sufficient abstraction

Goals

Prolog
+
easy web integration
=
more intelligent web applications

We need:

- definition of user interface in a natural way possibly with a graphical editor
- possibility of reusing Prolog code with little or no modification
- integration with existing systems

Characteristics of the proposed framework

- *short development time*: hides the complexity of web protocols
- *ease of use*: does not require code to be written in an imperative language
- *maintainability*: separates presentation and application logic
- *GUI editor support*: describes presentation declaratively in XML files
- *open architecture*: is extensible in almost all aspects
- *integration*: flexible, cooperates with web servers

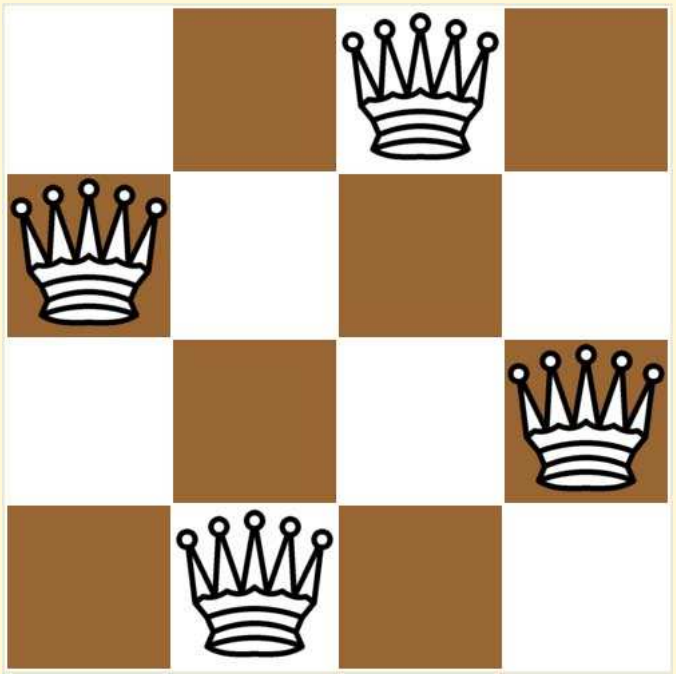
Motivating example

N-queens problem: how to place N queens on an $N \times N$ board so that they do not threaten one another

- Prolog implementation consists of 8 predicates
- task: create a web interface that presents the solutions

$\Rightarrow$ with Prosper this can be accomplished with a server page of 21 lines (using a board drawing extension)

Output for motivating example



Template for motivating example

```
<psp:assign var="N"
            expr="{atom_number(http_get(n))}">
  <psp:for-each-else>
    <psp:for-each function="queens(N)"
                  iterator="L">
      <board:chessboard size="{N}" figures="L" />
    </psp:for-each>
    <psp:else>
      <p>Sorry but there are no solutions.</p>
    </psp:else>
  </psp:for-each-else>
</psp:assign>
```

Template technology

- declarative and fits the Prolog language
- allows developing complex web applications
- extensible
- easy to learn, resembles other server-side technologies

Template technology

Two major constituents:

- application logic
- presentation layer

Application logic

Implemented in regular Prolog modules, no modification needed.

N-queens: `queens.pl` module constitutes the application logic.

Templates (server pages)

- define presentation in XML format:
  - special XML elements with semantics defined as Prolog *transformation rules*
- have a standard set of elements similar to other technologies:
  - conditionals (`psp:if`, `psp:if-else`)
  - branching (`psp:choose`)
  - iteration (`psp:for-each`)
  - more solutions (`psp:for-all`)
  - variables (`psp:assign`, `psp:insert`)

Templates (server pages)

- have single-assigned variables to store intermediate values
- feature a typed expression language (an extension to is/2) with special functions to access HTTP context
- access application logic predicates directly

Extensibility

Prosper = Prolog Server Pages *Extensible* Architecture

Standard set of elements is extensible with user-defined elements:

- elements referenced in XML in the form *<namespace>*:*<name>*
- namespaces bound to Prolog element implementor module in configuration file
- semantics:
  - *<name>*(VariableTypes, Attributes, Content, Terms) hook predicate for syntactic validation
  - *<name>*(. . . , Variables, Terms, Elements) hook predicate for evaluation and content generation

Summary

Prosper is a framework that

- facilitates developing web applications in Prolog re-using existing Prolog code
- separates application logic and presentation
- is efficient and flexible
- has an open architecture
- is easy to learn

Reference implementation:
http://prospear.sourceforge.net/