OMSE 531: Software Requirements Engineering

Spring 2012 Assignment 1
Due: May 5th, 2012
Instructor: Joe Maybee
jmaybee@pdx.edu

Overview

In this assignment you will build a framework for formal requirements that will generate natural language specifications from a formal representation of finite state automata.

The assignment is due no later than Saturday, May 5th, 2012. Assignments must be delivered via the D2L drop box.

Assignment: Requirements Framework

The representation of the requirements will consist of two separate files. The first file will consist of basic facts concerning the requirements and will be named '`ground.pl`'. The second file will consist of the requirements themselves and will be named '`req.pl`'.

In this assignment, you will provide a third file which will contain the necessary Prolog to validate that each requirement adheres to the facts in the '`ground.pl`' file, and also to generate natural language requirements from the requirements.

The basic definition of facts in the '`ground.pl`' file will define the valid states, events and responses for the requirements set.

Definitions in 'ground.pl' use the names '`state/1`', '`event/1`', and '`response/1`'. For example:

```
state(temp_hi).     % defines 'temp_hi' to be a valid state name

event(over_temp).   % defines 'over_temp' to be a valid event name

response(fan_on) .  % defines 'fan_on' to be a valid system response
```

Requirements are defined in the file '`req.pl`' using '`req/5`', and the individual terms correspond to the requirement identifier, the starting state, the event, the response to the event and the final state. A requirement would look like this:

```
%
% requirement id_21
%

req(id_21, temp_normal, over_temp, fan_on, temp_hi).
```

Requirements processing

The student will provide clauses in a separate file, along with any necessary rules to support them that will generate natural language from the requirements base named '`gen_nl_reqs/0`'.

The output for `gen_nl_reqs/0` for the requirement listed above would look like this:

```
Req id_21:  If the system is in a temp_normal state, and an over_temp even
occurs, the system shall respond with fan_on and move to the temp_hi state.
```

In addition, the student will also provide an additional clause '`valid_req_spec/1`', which will evaluate a given requirement for validity (terms for the requirement are valid):

```
?- valid_req_spec(id_21).
      true.
?-
```

Assignment Example and Evaluation Files

The instructor will provide some example files for '`ground.pl`' and '`req.pl`' to help get students started with their understanding of the inputs to the system. These example files will not be complete nor representative of the files used for evaluating the assignment

One week before the assignment due date, the instructor will provide 'ground.pl', and 'req.pl'

Grading

The artifacts will be judged on the following merits:

Correct: Does the code perform correctly?

Clarity: Are the code and comments easy to understand?

Conciseness / Succinctness: Is the code straight to the point?

Coherence: Does the code reflect the process as it was defined?

Completeness: Are all elements of the code present and easy to identify?