

Clayton School of Information Technology
CSE4213 Assignment 2
Course and Unit Specification
 Due Date: 12noon, 24 Apr 2007

1 Objectives

The objectives of this exercise are:

1. To develop robust specifications, and thereby understand the difference between preconditions and guards;
2. To use multiple machines in developing a specification, and thereby gain some understanding of the B design process;
3. To explore how to add further theory to the B Toolkit, to allow the discharge of proof obligations in particular specifications.

2 Making *Enrol1* Robust

Extend/alter your specification for *Enrol1* (from Assignment 1) to make a new specification *Enrol2* which includes a new operation (in addition to those in *Enrol1*):

OfferUnit(unit) which adds the parameter *unit* to the set of *offered* units. (This operation was left out of Assignment 1.)

Then you are to develop a new machine *Enrol2R* which INCLUDES your machine *Enrol2*, and makes all the operations of *Enrol2* robust, according to the discussion in lectures.

Enrol2R should draw its operation responses from a separate machine called *Responses*.

(4 marks)

3 The *UnitSpec2* Machine

Specify an abstract machine whose sole purpose is to define a set of units. If you feel brave, a complete list of FIT units is available on-line in the MUSO web pages for this unit, or at the URL:

`http://bendigo.infotech.monash.edu.au/
 ~ajh/teaching/cse4213/2007/assessment/assignment2/units.txt`

(4 marks)

4 The *CourseSpec2* Machine

The main task is to develop a specification of the unit and course framework, to identify in what units a student must be enrolled in order to complete the course. To simplify this specification, assume that a course is made up from a number of *components*. Each ‘component’ is a tuple (cross product) consisting of a natural number, and a set of *units*. The number defines how many of the units in the set must be completed. If the number equals $\text{card}(\text{units})$, then every unit must be completed, that is, is compulsory. If the number equals 1, then the set of units defines a single elective unit, chosen from within that set.

For example, ‘components’ could define year levels of a course, with one ‘component’ defining all the core units at that level, and another ‘component’ defining all the elective units at that year level.

1. Define an operation $NewCourse(name)$ that creates a new course with name $name$. $name$ should be drawn from a set defined by a parameter to the machine. (2 marks)
2. Define an operation $comp \leftarrow NewComponent(nn, units)$ that defines a component of a course, according to the explanation given above. $units$ must be a subset of the units defined in the $UnitSpec2$ machine. $comp$ is the returned new component. (2 marks)
3. Define an operation $AddComponent(name, comp)$ that adds component $comp$ to the course named $name$. (2 marks)

Submit the publication form of all the proof obligations and the proofs that show the discharge of all proof obligations.

(3 marks)

5 The $CourseSpec2R$ Machine

Define a new machine $CourseSpec2R$ that makes $CourseSpec2$ robust. Use the $Response$ machine to define any responses.

(3 marks)

6 Specific Tasks

1. Code up your specifications $UnitSpec2$, $CourseSpec2$ and $CourseSpec2R$ (and any utility machines), using the ASCII form, and use the B Toolkit to analyse them. Submit for assessment publication format listings of all machines.
2. Generate the proof obligations, and discharge them. Submit the publication form of all the proof obligations and the proofs that show the discharge of all proof obligations. You may need to add new theory to the proof methods to do this. Make sure you explain what steps you have taken in this process. For example, if you add a new inference rule, explain what that rule is saying and what it is valid in this context. If you need to introduce a lemma, and prove it off-line, show the proof in mathematical form, that is, with each simplification or rewriting you make, the theorem or justification for performing that simplification or rewrite. Use the method described below to build your submission.

7 \LaTeX Issues

Remember that all submissions must be in the form of soft files, encoded into a single tar file. Hence if you want to add explanations to your submitted work, you need to know how the \LaTeX files are encoded. Comments in machine specifications can be made in the form of C-style comments ($/* \dots */$), and comments in \LaTeX files may be included by starting a comment with a percent sign (%). The comment ends at the end of line.

Some explanation of handling the \LaTeX documents themselves may be in order. The BToolkit generates \TeX documents in the \TeX subdirectory, with (fairly) obvious names. you can copy these files out of that directory, and use them to build your own documentation. The main thing to note is that they require a suite of \TeX macros to work. These macros are defined in $BDefs.sty$ and $BToolkit.sty$ – see the resources page for downloading instructions, and on how to use them in a \LaTeX document.

You can then build your own L^AT_EX documents (like this one), including code fragments from the BTools:

```
MACHINE SimpleDate ( YEAR , MONTH , DAY )
CONSTRAINTS YEAR = 2000 .. 2099  $\wedge$  MONTH = 1 .. 12  $\wedge$  DAY = 1 .. 31
    et cetera ...
END
```

This fragment was produced with (cut and paste from a BTool generated tex file):

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% SimpleDate.mch.tex
%
\bsetindent
\begin{tabbing}
\setTabs
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MACHINE
%
{\bf MACHINE} \bhspace+\em SimpleDate\ ( {\em YEAR\} , {\em MONTH\} , {\em DAY\} )
-\label{SimpleDate}\index{SimpleDate}
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% CONSTRAINTS
%
\bbnl
{\bf CONSTRAINTS} \bhspace{\em YEAR\} $=$ {\em 2000\} $\upto$ {\em 2099\} $\wedge$
{\em MONTH\} $=$ {\em 1\} $\upto$ {\em 12\} $\wedge$
{\em DAY\} $=$ {\em 1\} $\upto$ {\em 31\}
%
\+\bbnl{\sf et cetera ...}\-
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% END
%
\bbnl
{\bf END}
\end{tabbing}
\resetindent
```