CSSE4603/7032 Models of Software Systems Semester 1, 2016

Assignment 3: MSMIE

Course coordinator: Graeme Smith **Due: 5pm, Fri 13 May**

The purpose of this assignment is to give you experience in developing a Z specification towards an implementation using data refinement. The case study is the Multiprocessor Shared-Memory Information Exchange (MSMIE) used in embedded software of nuclear safety systems.

Initial Z specification

MSMIE uses multiple shared memory locations to ensure that data is never overwritten by one process while it is being read by another. It has been designed so that neither reading nor writing processes should have to wait for a memory location to become available. In the simplification considered here it is assumed that there is a single writing processor and multiple reading processes.

For each shared variable, MSMIE uses three memory locations. At any time, one is reserved for writing, another may be reserved for reading (by one or more processes) and another may have just been written and contains the latest information, but is not yet being read. This results in 4 configurations

$$CONFIG ::= wii \mid wri \mid wni \mid wrn$$

where *w* denotes a memory location is reserved for writing, *r* a memory location is reserved for reading, *n* a memory location contains the newest information but is not being read, and *i* a memory location is idle (i.e., none of the preceding apply).

The following specification of a MSMIE shared variable does not model the processes nor the actual data. It is only concerned with modifying the statuses of the memory locations. The state comprises a configuration *config* and a set of reading processes *readers*. Each process is identified by a process ID of type *PID*.

$$[PID]$$

MSMIE

$config : CONFIG$
$readers : \mathbb{P}\, PID$

$config \in \{wii, wni\} \Leftrightarrow readers = \varnothing$

The invariant of the state schema ensures that the set of readers is empty precisely when no memory locations are being read.

Initially, there is one memory location reserved for writing and the other two memory locations are idle.

MSMIEInit

$MSMIE$

$config = wii$

The *Write* operation corresponds to a write occurring. The status of the memory location to which the write occurs is set to *n* and another memory location with status either *i* or *n* is set to *w*. The operation is defined in terms of a function *next_config* which returns the resulting configuration.

$$\begin{array}{|l}
next\_config : CONFIG \to CONFIG \\
\hline
\forall c : CONFIG \bullet \\
\qquad c \in \{wii, wni\} \Rightarrow next\_config(c) = wni \\
\qquad c \in \{wri, wrn\} \Rightarrow next\_config(c) = wrn
\end{array}$$

$$\begin{array}{|l}
\underline{\quad Write \quad} \\
\Delta MSMIE \\
\hline
config' = next\_config(config) \\
readers' = readers \\
\hline
\end{array}$$

Other operations corresponding to a process reading a memory location are not included here.

Towards an implementation

To implement MSMIE, we could have variables w, r and n which are each references (i.e., pointers) to a memory location. For example, we could define a type

$[Ref]$

to denote references to memory locations and a constant

$$\begin{array}{|l}
null : Ref
\end{array}$$

to denote the null reference. Then the state schema is

$$\begin{array}{|l}
\underline{\quad MSMIE1 \quad} \\
w, r, n : Ref \\
readers1 : \mathbb{P}\, PID \\
\hline
w \neq null \\
w \neq r \land w \neq n \land (r = n \Rightarrow r = n = null) \\
r = null \Leftrightarrow readers1 = \varnothing \\
\hline
\end{array}$$

The initial state schema is

$$\begin{array}{|l}
\underline{\quad MSMIEInit1 \quad} \\
MSMIE1 \\
\hline
r = n = null \\
\hline
\end{array}$$

and the operation corresponding to a write occurring is

$$\begin{array}{|l}
\underline{\quad Write1 \quad} \\
\Delta MSMIE1 \\
\hline
n' = w \land r' = r \\
readers1' = readers1 \\
\hline
\end{array}$$

Task description

Your task is to prove the second specification of MSMIE is a data refinement of the first. To do this, you must define a suitable function *current_config* and use the following retrieve relation.

$$
\begin{array}{|l}
\multicolumn{1}{l}{R} \\
\hline
\mathit{MSMIE} \\
\mathit{MSMIE}1 \\
\hline
\mathit{config} = \mathit{current\_config}(r, n) \\
\mathit{readers} = \mathit{readers}1 \\
\hline
\end{array}
$$

A skeleton of the function *current_config* and the required proofs is available as a LaTeX document on the Blackboard site.

A pdf file of your completed assignment must be submitted via Blackboard by the due date.

Instructions for writing proofs in LaTeX are in the file guide.pdf on the Blackboard site. Note that CZT cannot currently be used to type check proofs.

CSSE7032 students only: You will need to write a short reflection (a few paragraphs only) on your experience with using formal methods in this course. What were the most challenging aspects of using the techniques? What potential improvements can you think of which would make the techniques easier to use?

Marking criteria

The assignment is worth 20% of your final grade.

For CSSE4603 students:

You will be given

- a mark out of 2 based on the percentage of required predicates for the function *current_config* that are both syntactically and semantically correct, and
- a mark out of 18 based on the percentage of required proof steps that are both correct and appropriately justified.

A mark of 0 will be given for work with little or no academic merit.

For CSSE7032 students, the above multiplied by 16/20 plus:

A mark out of 4 for a clear and concise reflection.

School Policy on Student Misconduct:

You are required to read and understand the School Statement on Misconduct, available on the School's website at:

http://www.itee.uq.edu.au/itee-student-misconduct-including-plagiarism