CMU 17-651 | Models of Software Systems | Fall 2018

Homework #7: Invariants and Introduction to Z

Dario A Lencina Talarico | Due: 15 October 2018

Part 1: Invariants

Consider the Diverging Counter example of Chapter 10 of GWC09. Prove that $x + y = 0$ is an invariant of the *DivergingCounter* state machine.

DivergingCounter = (
$[x, y : \mathbb{Z}]$,
$\{s : [x, y : \mathbb{Z}] \mid s(x) = -s(y)\}$,
$\{poke(i : \mathbb{Z})\}$,
$\delta ==$

poke(i: $\mathbb{Z}$)
pre $i > 0$
post $x' = x + i \wedge y' = y - i$

).
1. Base case: show that θ holds in the initial state.
Here there's only one initial state:
$[x = 0;\ y = 0]$
Proof:
x + y = 0

= [initial state]
0 + 0 = 0
= [aritmetic]
0 = 0
2. Induction step on inc:

Show: $\theta(s), pre(s), post(s, s') \vdash \theta(s')$
That is, from $x' = x + i \wedge y' = y - i$

$\theta(s) == x + y = 0$
prove that $x' + y' = 0$
Proof:

y'

= [poscondition]
y - i
= [introduction hypothesis y + x = 0, or y = -x]
-x - i
= [factorizing]
-(x + i)
= [replacing definition of x']
-x'

y' = -x' is equivalent to y' + x' = 0
(Note: In your proof use style C (in Section 10.1.1) of reasoning about invariants and a similar degree of formalism as in the lecture on this topic.)

Part 2: Z

Note: For this part of the assignment you must format your answers using LaTeXand typecheck the answers using *fuzz*, Z-EVES, or the Community Z tools.

Write a Z specification of the following system. Your specification should include sufficient explanatory prose to make it easily understandable. (The prose is important—answers with little or no prose will receive a low grade.)

A teacher wants to keep a register of students in the class, and to record which students have completed their homework.

Let the given set *Student* represent the set of all students who might ever be enrolled in a class:

[*Student*]

Specify each of the following:

1. The state space for a register.

   Hint: use two sets of students:

   Register
   $enrolled : \mathbb{P}\, Student$
   $completed : \mathbb{P}\, Student$

   $completed \subseteq enrolled$

2. An operation to enroll a new student.

   Enroll
   $\Delta Register$
   $student? : Student$

   $student? \notin enrolled$
   $enrolled' = enrolled \cup \{\text{student?}\}$

3. The initial state(s) for your state space. The proposed spec initialized both sets as empty sets.

$$
\begin{array}{|l}
\hline
\multicolumn{1}{l}{\mathit{InitRegister}} \\
\mathit{Register} \\
\hline
\mathit{enrolled} = \emptyset \\
\mathit{completed} = \emptyset \\
\hline
\end{array}
$$

4. An operation to record that a student (already enrolled in class) has completed the homework. The proposed Z spec adds student? to the completed power set by definining a union.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\mathit{RegisterHomeworkCompletion}} \\
\Delta \mathit{Register} \\
\mathit{student}? : \mathit{Student} \\
\hline
\mathit{student}? \in \mathit{enrolled} \\
\mathit{completed}' = \mathit{completed} \cup \{\text{student?}\} \\
\hline
\end{array}
$$

5. An operation to inquire whether a student (who must be enrolled) has completed the homework (the answer is to be either ‘Yes’ or ‘No’). Defining type to answer:
Answer ::= Yes | No

$$
\begin{array}{|l}
\multicolumn{1}{l}{\mathit{HasCompletedTheHomeworkSuccess}} \\
\Xi \mathit{Register} \\
\mathit{student}? : \mathit{Student} \\
\mathit{answer}! : \mathit{Answer} \\
\hline
\mathit{student}? \in \mathit{completed} \\
\mathit{answer}! : \mathit{Yes} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\mathit{HasCompletedTheHomeworkErr}} \\
\Xi \mathit{Register} \\
\mathit{student}? : \mathit{Student} \\
\mathit{answer}! : \mathit{Answer} \\
\hline
\mathit{student}? \notin \mathit{completed} \\
\mathit{answer}! : \mathit{No} \\
\hline
\end{array}
$$

Only one of the expressions will meet the preconditions so $\vee$ is the right relation to use:

$\text{HasCompletedTheHomework} \doteq \mathit{HasCompletedTheHomeworkSuccess} \vee \mathit{HasCompletedTheHomeworkErr}$

6. A robust version of the system. (Be sure to use the schema calculus, as illustrated by the class lecture and the paper by Spivey on Z.)
Declaring type to return a result:
REPORT ::= ok | already-enrolled | not-enrolled

$$
\begin{array}{|l}
\multicolumn{1}{l}{\mathit{Success}} \\
\mathit{result}! : \mathit{REPORT} \\
\hline
\mathit{result}! : \mathit{ok} \\
\hline
\end{array}
$$

Used to capture the scenario where the student is not enrolled in the register:

$$\begin{array}{|l}
\hline \textit{NotEnrolled} \\
\Xi Register \\
student? : Student \\
result! : REPORT \\
\hline
student? \notin enrolled \\
result! : \text{not-enrolled} \\
\hline
\end{array}$$

$$\begin{array}{|l}
\hline \textit{AlreadyEnrolled} \\
\Xi Register \\
student? : Student \\
result! : REPORT \\
\hline
student? \in enrolled \\
result! : \text{already-enrolled} \\
\hline
\end{array}$$

Robust versions of the operations:
RREnroll $\widehat{=}$ $(Enroll \wedge Success) \vee AlreadyEnrolled$
RRegisterHomeworkCompletion $\widehat{=}$ $(RegisterHomeworkCompletion \wedge Success) \vee NotEnrolled$
HasCompletedTheHomework is already a robust method so no need to modify.