Rigorous Software Development
Event-B Exercise Sheet #1

Manuel Carro
manuel.carro@upm.es

November 30th, 2019

Due on Dec. 11th, 2019, 7pm CET

1 Calculate $a \times b$ using addition and bit-shifting

Given constants $a \in \mathbb{N}, b \in \mathbb{N}$, we want to calculate $r = a \times b$ using only bit shifts, the operators >> and << available in Java, C, Python and other programming languages.

Given variables $x \in \mathbb{N}, y \in \mathbb{N}, r \in \mathbb{N}$, the (sequential) events shown below:

```
Event INIT
  x,y,r := a,b,0
end
```

```
Event Finish
  when x = 0
  then skip
end
```

```
Event ProgressOdd
  when (x > 0 ∧ x mod 2 = 1)
  then
    r, x, y := r + y, x ÷ 2, y × 2
end
```

```
Event ProgressEven
  when (x > 0 ∧ x mod 2 = 0)
  then
    x, y := x ÷ 2, y × 2
end
```

where '÷' means integer division, are assumed to calculate `a * b`.

Note: In the division example we *defined* division in terms of multiplication and addition. In the current exercise we assume that we know what multiplication is, but we want to implement it with the simpler operations ×2 and ÷2 (that can be implemented using bitwise shifts).

1.1 Execution trace

Write a trace of the values taken by the variables for $a = 102$ and $b = 3$. As an example, for $a = 98, b = 7$ it would be

x	y	r
98	7	0
49	14	0
24	28	14
12	56	14
6	112	14
3	224	14
1	448	238
0	896	686

Note: writing a trace is not really necessary, but in some cases it may give you an intuition of the relationships among the variables during in the execution. That informal understanding may help in working out a reasonable invariant.

Hint: if the previous examples do not ring a bell, you may also try with $x = 2^n$ and $x = 2^n \pm 1$ for different (not too large) n

1.2 Invariant

The type axioms and invariants for our system are

$$\begin{array}{llll} A_1: & a \in \mathbb{N} & I_1: & x \in \mathbb{N} \\ A_2: & b \in \mathbb{N} & I_2: & y \in \mathbb{N} \\ & & I_3: & r \in \mathbb{N} \end{array}$$

Write an invariant $I_4(a, b, x, y, r)$ for the model such than when $x = 0$ (i.e., when the event `Finish` can be fired), it implies that $r = a \times b$:

$$I_4 \wedge x = 0 \Rightarrow r = a \times b$$

The aim of this invariant is to prove the correctness of the model w.r.t. its intended meaning.

1.3 Prove that the invariant I_4 is inductive.

Prove, using sequent-style proofs, that the invariant I_4 from the previous section is inductive, i.e., prove the family of sequents

$$A(\overline{c}), G_i(\overline{v}, \overline{c}), I_{1\ldots4}(\overline{v}, \overline{c}) \vdash I_4(E_i(\overline{v}, \overline{c}), \overline{c})$$

for all guards G_i and events E_i that change the state in the model (i.e., for `INITIALIZATION`, `ProgressEven`, and `ProgressOdd`).

2 Turning in the Homework

You may:

- Hand me the printed homework (handwritten or typeset; I do not mind as long as I can understand it) in the classroom, or
- Send me a **PDF**. Please do not send me documents in .doc / .docx / .rtf / … formats.

In any of theses cases I will only grade homework that has been sent to me by **December 11th, 2019, 7pm CET**, since I intend to use part of that session to explain / explore solutions to the homework.

Please remember to add a personal identification to what you turn in.