Sequential programs, refinement, and proof obligations[1]

Manuel Carro
manuel.carro@upm.es

Universidad Politécnica de Madrid &
IMDEA Software Institute

[1] Several slides, examples, borrowed from J. R. Abrial

All you ever wanted to know about installing Rodin...

...is at

https://wp.software.imdea.org/cbc/#tools

and

https://wp.software.imdea.org/cbc/rodin-installation-and-tips/

Sequential programs and Event B

- Sequential programs can be transpiled into Event B.
- Correctness, termination, etc. proven with Event B tools.
- However, underuse of Event B.
  Other approaches are very good at this.
- Better approach: design with Event B from the beginning.
- Apply to reactive and concurrent systems – strong points of Event B.
- For illustration: will develop several sequential programs.

Appetizer

Let us use Rodin with the Integer Division example.

```
INITIALISATION
   a, r := 0, b
END

EVENT Progress
   WHERE r >= c THEN
      r, a := r − c, a + 1
END

EVENT Finish
   WHERE r < c THEN
      skip
END
```

Two types of components in a Rodin project:

- Context(s) Contains constants and axioms.
- Machine(s) Variables, invariants, and events (and some other things). Machines *see* Contexts.

Switching to Rodin. The example I will type is available as part of the course material.

Specification of a sequential program

- Sequential programs are usually specified by means of:
  - A precondition
  - And a postcondition
- Represented with a Hoare triple

$$\{Pre\} \quad P \quad \{Post\}$$

Searching in an array

We are given as preconditions:

- A natural, non-zero number: $n \in \mathbb{N}1$.
- An array f of n elements of naturals: $f \in 1..n \rightarrow \mathbb{N}$.
- A value v known to be in the array: $v \in \mathrm{ran}(f)$.

We are looking for (postconditions):

- An index r in the array: $r \in \mathrm{dom}(f)$
- Such that $f(r) = v$

$$\left\{ \begin{array}{l} n \in \mathbb{N}1 \\ f \in 1..n \rightarrow \mathbb{N} \\ v \in \mathrm{ran}(f) \end{array} \right\} \quad \textbf{search} \quad \left\{ \begin{array}{l} r \in \mathrm{dom}(f) \\ f(r) = v \end{array} \right\}$$

Encoding a Hoare-triplet

Preconditions	Program	Postconditions
$\left\{ \begin{array}{l} n \in \mathbb{N}1 \\ f \in 1..n \rightarrow \mathbb{N} \\ v \in \mathrm{ran}(f) \end{array} \right\}$	**search**	$\left\{ \begin{array}{l} r \in \mathrm{dom}(f) \\ f(r) = v \end{array} \right\}$
Axioms Input parameters, constants		Guards, invariants Variables

- Ensuring (total) correctness:
  - post-condition implied by invariants and guard of (unique) final event: $Axioms, Invs, \neg Guard \vdash Post$.
  - Non-final events terminate.
  - Events are deterministic.
  - Events do not deadlock.
- We will see later how to formally express the last two properties.

Encoding search

$\left\{ \begin{array}{l} n \in \mathbb{N}1 \\ f \in 1..n \to \mathbb{N} \\ v \in \mathrm{ran}(f) \end{array} \right\}$ **search** $\left\{ \begin{array}{l} r \in \mathrm{dom}(f) \\ f(r) = v \end{array} \right\}$

Constants: n, f, v
Axiom 1: $n \in \mathbb{N}1$
Axiom 2: $f \in 1..n \to \mathbb{N}$
Axiom 3: $v \in \mathrm{ran}(f)$

$r :\in \mathrm{dom}(f)$ assigns to r a number randomly chosen from the set $dom(f)$.

VARIABLES r
INVARIANTS $r \in \mathrm{dom}(f)$
INIT
  $r :\in \mathrm{dom}(f)$
END

EVENT Finish
  WHERE $f(r) = v$
  THEN
    skip
END

EVENT Progress
  WHERE $f(r) \neq v$
  THEN
    $r :\in \mathrm{dom}(f)$
END

Encoding search (cont.)

- Does not capture a *good* computation method (Why?).
- Let us write it in Rodin.
- Entering symbols:

To enter...	type
$\in$	`:`
$:\in$	`::`
$\mathbb{N}$	`NAT`
$\to$	`-->`
$\neq$	`/=`

$f \in \mathbb{N} \to 1..n$ would be typed `f : NAT --> 1..n`

Open Rodin and let start typing it together.