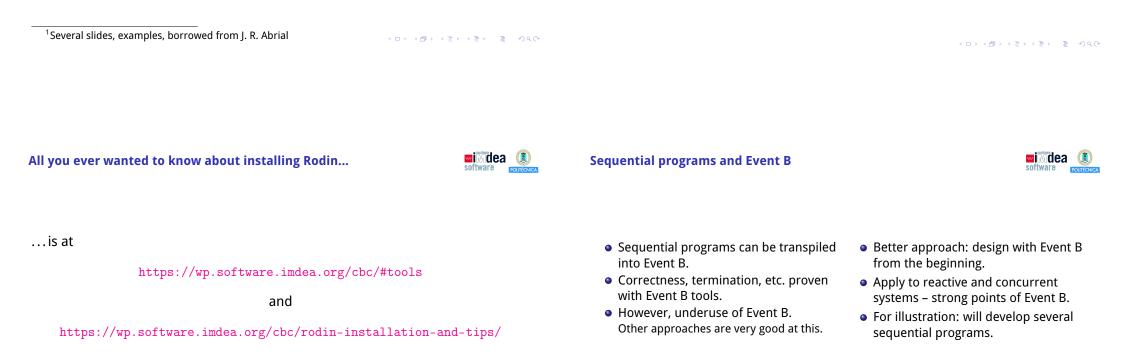


# Sequential programs, refinement, and proof obligations<sup>1</sup>

Manuel Carro manuel.carro@upm.es

#### Universidad Politécnica de Madrid & IMDEA Software Institute

Installing and using Rodins. 3	Refinement: the sorted array cases. 25
Sequential programs: specification and	Guard strengthenings. 32
propertiess. 4	Simulations. 38
Specification of searching in arrays. 7	Rodin and refinements. 41
Refinement of searchs. 12	Rodin proof of INVs. 43
Termination and correctnesss. 17	Reviewed hypothesess. 57
Well-definedness and feasibilitys. 20	Theoremss. 58



## Appetizer Let us use Rodin with the Integer Division example.



Specification of a sequential program



INITIALISATION<br/>a, r := 0, bTwo types of co<br/>project:ENDContext(s) Co<br/>axiEVENT Progress<br/>WHERE  $r \ge c$  THEN<br/>r, a := r - c, a + 1Machine(s) Val<br/>eventhat<br/>eventhat<br/>CoEVENT Finish<br/>WHERE r < c THEN<br/>skipSwitching to Ro

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.

◆□▶★舂▶★≧▶★≧▶ ≧ のへで

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

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

wi Mdea

#### Searching in an array

END



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:  $v \in ran(f)$ .

We are looking for (postconditions):

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

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

# Encoding a Hoare-triplet Preconditions Program Postconditions $(n \in \mathbb{N}1)$ )

 $\begin{cases} f \in 1..n \to \mathbb{N} \\ v \in \operatorname{ran}(f) \end{cases}$  search  $\begin{cases} r \in \operatorname{dom}(f) \\ f(r) = v \end{cases}$ Axioms
Guards, invariants
Variables

• Ensuring (total) correctness:

- post-condition implied by invariants and guard of (unique) final event: Axioms, Invs, ¬Guard ⊢ 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**

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

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

 $f \in 1..n \rightarrow \mathbb{N}$ 

r := dom(f) assigns to r a number randomly chosen from the set dom(f).

```
VARIABLES r
INVARIANTS r \in \text{dom}(f)
INIT
  r :\in \operatorname{dom}(f)
END
EVENT Finish
  WHERE f(r) = v
  THEN
     skip
END
EVENT Progress
  WHERE f(r) \neq v
  THEN
     r :\in \operatorname{dom}(f)
```

END

| ↓ □ ▶ ★ ፼ ▶ ★ 厘 ▶ ★ 厘 ▶ ↓ 厘 → りへで

Encoding search (cont.)



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

To enter	type
$\in$	:
:∈	::
$\mathbb{N}$	NAT
$\rightarrow$	>
$\neq$	/=

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

Open Rodin and let start typing it together.

・ロト・(部)・(目)・(目)・(日)・