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.

CONSTANTS b, c
AXIOMS $b \in \mathbb{N}, c \in \mathbb{N}_1$

VARIABLES a, r
INVARIANTS $a \in \mathbb{N}, r \in \mathbb{N}, a \times c + r = b$

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: $f \in 1..n \to \mathbb{N}$.
- 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 $f(r) = v$

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

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: $Axioms, Invs, Guard \vdash Post$.
  - 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

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

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 $dom(f)$.

(Actually, it just states r is in $\mathrm{dom}(f)$. Operational approximation: random assignment. Better approximation: "represents all executions with all possible elements in $\mathrm{dom}(f)$.")

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

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

EVENT Finish
  WHERE $f(r) = v$
  THEN
    skip
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`
$\rightarrow$	`-->`
$\neq$	`/=`

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

Open Rodin and let start typing it together.

Some Rodin conventions

- Every line has an identifier, used to refer to the line.

```
Search:  not extended ordinary
THEN
 ∘   act1:   r :∈ dom(f) ›
END
```

- Rodin generated proof obligations (but we have seen only INV).

```
▾ Proof Obligations
    INITIALISATION/inv1/INV
    INITIALISATION/act1/FIS
    Search/inv1/INV
    Search/act1/FIS
    Finish/grd1/WD
```

- Proof naming: EventName/Identifier/TypeOfProof
- FIS: prove operation can be applied (is there any element in $\mathrm{dom}(f)$?)
- WD: (sub)expression is well-defined (it can be evaluated)
- Some help from more powerful theorem provers may be needed.
- **Note:** (un)discharged proof obligations may differ across versions due to differences in theorem provers, and relative processor speed (timeouts involved). General ideas applicable, though.

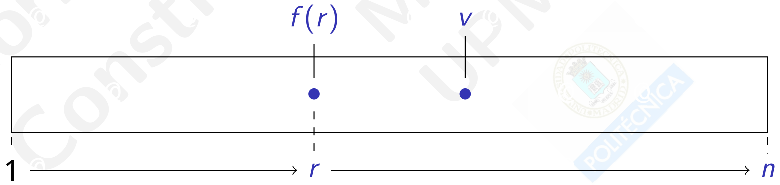
Refinement

Purposes of refinement

- Add more requirements, and/or
- Have a realizable design, and/or
- Increase performance.

Idea for this case

- Scan vector from left to right.



Refined events

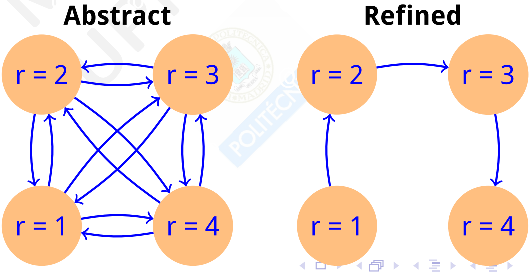
Right click on machine, select *Refine*, enter new name, change events to *non extended* for **INIT** and **Search**, edit $\Rightarrow$ SIM proof not discharged.

Event INITIALISATION
    $r := 1$
end

Event Progress
    where $f(r) \neq v$
then
    $r := r + 1$
end

Event Finish
    where $f(r) = v$
end

- We won't see SIM's formal definition at this moment.
- Intuitively:
  - Invariants of abstract models kept.
  - *Histories* of new, refined model must be subset of abstract model's.
  - No new behavior introduced $\Longrightarrow$ correctness preserved.



Refined events

Right click on machine, select *Refine*, enter new name, change events to *non extended* for **INIT** and **Search**, edit $\Rightarrow$ SIM proof not discharged.

Event INITIALISATION
    r := 1
end

Event Progress
    where f(r) $\neq$ v
then
    r := r + 1
end

Event Finish
    where f(r) = v
end

- Cannot prove SIM because $r \in \mathrm{dom}(f)$ cannot be proven. ($r \in \mathrm{dom}(f)$ invariant of previous model.)
- Can (refined) Progress transition to a state where (abstract) Progress cannot?

	Invariant	Guard	Action
Abstract	$r \in \mathrm{dom}(f)$	$f(r) \neq v$	$r' :\in \mathrm{dom}(f)$
Refined	$r \in \mathrm{dom}(f)$	$f(r) \neq v$	$r' := r + 1$

Refined events

Right click on machine, select *Refine*, enter new name, change events to *non extended* for **INIT** and **Search**, edit $\Rightarrow$ SIM proof not discharged.

```
Event INITIALISATION
    r := 1
end

Event Progress
    where f(r) ≠ v
then
    r := r + 1
end

Event Finish
    where f(r) = v
end
```

- Cannot prove SIM because $r \in \text{dom}(f)$ cannot be proven. ($r \in \text{dom}(f)$ invariant of previous model.)
- Can (refined) Progress transition to a state where (abstract) Progress cannot?

	Invariant	Guard	Action
Abstract	$r \in \text{dom}(f)$	$f(r) \neq v$	$r' :\in \text{dom}(f)$
Refined	$r \in \text{dom}(f)$	$f(r) \neq v$	$r' := r + 1$

- Can $r' :\in \text{dom}(f)$ be proven in refined model?

$$r \in dom(f), f \in 1..n \rightarrow \mathbb{N}, v \in ran(f), f(r) \neq v \vdash r + 1 \in dom(f)$$

Refined events

Right click on machine, select *Refine*, enter new name, change events to *non extended* for **INIT** and **Search**, edit $\Rightarrow$ SIM proof not discharged.

```
Event INITIALISATION
    r := 1
end

Event Progress
    where f(r) ≠ v
then
    r := r + 1
end

Event Finish
    where f(r) = v
end
```

- Cannot prove SIM because $r \in \mathrm{dom}(f)$ cannot be proven. ($r \in \mathrm{dom}(f)$ invariant of previous model.)
- Can (refined) Progress transition to a state where (abstract) Progress cannot?

	Invariant	Guard	Action
Abstract	$r \in \mathrm{dom}(f)$	$f(r) \neq v$	$r' :\in \mathrm{dom}(f)$
Refined	$r \in \mathrm{dom}(f)$	$f(r) \neq v$	$r' := r + 1$

- Can $r' :\in \mathrm{dom}(f)$ be proven in refined model?

$$r \in dom(f), f \in 1..n \to \mathbb{N}, v \in ran(f), f(r) \neq v \vdash r + 1 \in dom(f)$$

Simple update of the model?

Refined events

Patching events

Event INITIALISATION

$r := 1$

end

Event Progress

where $r < n \land f(r) \neq v$

then

$r := r + 1$

end

Event Finish

where $f(r) = v$

end

Refined events

Patching events

Event INITIALISATION
$r := 1$
end

Event Progress
where $r < n \wedge f(r) \neq v$
then
$r := r + 1$
end

Event Finish
where $f(r) = v$
end

- Update model in Rodin, check POs,

Refined events

Patching events

```
Event INITIALISATION
    r := 1
end

Event Progress
    where r < n ∧ f(r) ≠ v
then
    r := r + 1
end

Event Finish
    where f(r) = v
end
```

- Update model in Rodin, check POs,
  $r \in dom(f), f \in 1..n \rightarrow \mathbb{N}, v \in ran(f), r < n, f(r) \neq v \vdash r+1 \in dom(f)$
- But:
  - Guards don't ensure absence of deadlock.
  - There are admissible states with both guards false
    Counterexample?

Refined events

Patching events

Event INITIALISATION
    r := 1
end

Event Progress
    where $r < n \wedge f(r) \neq v$
then
    r := r + 1
end

Event Finish
    where $f(r) = v$
end

- Update model in Rodin, check POs,
  $r \in dom(f), f \in 1..n \rightarrow \mathbb{N}, v \in ran(f), r < n, f(r) \neq v \vdash r + 1 \in dom(f)$
- But:
  - Guards don't ensure absence of deadlock.
  - There are admissible states with both guards false
    Counterexample?
  - However, can we really reach a point where $r = n \wedge f(r) \neq v$?

Event INITIALISATION
    r := 1
end

Event Progress
    where $r < n \wedge f(r) \neq v$
then
    r := r + 1
end

Event Finish
    where $f(r) = v$
end

- Update model in Rodin, check POs,
  $r \in dom(f), f \in 1..n \to \mathbb{N}, v \in ran(f), r < n, f(r) \neq v \vdash r + 1 \in dom(f)$
- But:
  - Guards don't ensure absence of deadlock.
  - There are admissible states with both guards false
    Counterexample?
  - However, can we really reach a point where $r = n \wedge f(r) \neq v$?
- Our formulas don't "remember" the past.
- If $r = n - 1 \wedge f(r) \neq v$, executing $r := r + 1$ and not finding v is admissible (with existing information and inference methods).

Refined events
Patching events

Event INITIALISATION
    r := 1
end

Event Progress
    where r < n ∧ f(r) ≠ v
then
    r := r + 1
end

Event Finish
    where f(r) = v
end

- Update model in Rodin, check POs,
  $r \in dom(f), f \in 1..n \rightarrow \mathbb{N}, v \in ran(f), r < n, f(r) \neq v \vdash r + 1 \in dom(f)$
- But:
  - Guards don't ensure absence of deadlock.
  - There are admissible states with both guards false
    Counterexample?
  - However, can we really reach a point where $r = n \wedge f(r) \neq v$?
- Our formulas don't "remember" the past.
- If $r = n - 1 \wedge f(r) \neq v$, executing $r := r + 1$ and not finding v is admissible (with existing information and inference methods).
- We need more information to help $\Rightarrow$ stronger invariants!

Refined events
Adding invariants

INVARIANTS

???

Event INITIALISATION

r := 1

end

Event Progress

where $f(r) \neq v$

then

r := r + 1

end

Event Finish ...

- We know $v \in \mathrm{dom}(f)$.
- Express the idea that at some point we will find v.

Refined events
Adding invariants

INVARIANTS

???

Event INITIALISATION

r := 1

end

Event Progress

where $f(r) \neq v$

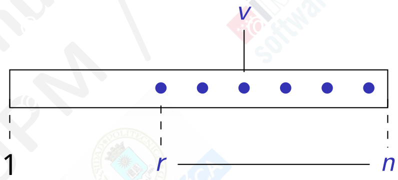
then

r := r + 1

end

Event Finish …

- We know $v \in \mathrm{dom}(f)$.
- Express the idea that at some point we will find v.
- Hint: *at all times, v is in position r or to "its right".*



Refined events
Adding invariants

INVARIANTS

???

Event INITIALISATION

r := 1

end

Event Progress

where f(r) $\neq$ v

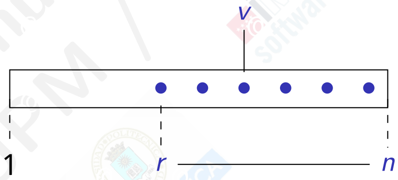
then

r := r + 1

end

Event Finish ...

- We know $v \in \mathrm{dom}(f)$.
- Express the idea that at some point we will find v.
- Hint: *at all times, v is in position r or to "its right".*



- Intuitively:
  $v \in \{f(r), f(r+1), \ldots, f(n-1), f(n)\}$

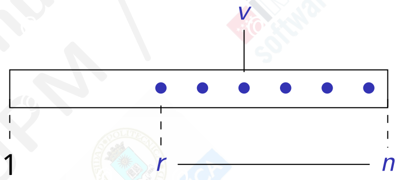
INVARIANTS
???

Event INITIALISATION
r := 1
end

Event Progress
where $f(r) \neq v$
then
r := r + 1
end

Event Finish ...

- We know $v \in \text{dom}(f)$.
- Express the idea that at some point we will find v.
- Hint: *at all times, v is in position r or to "its right".*



- Intuitively:
  $v \in \{f(r), f(r+1), \ldots, f(n-1), f(n)\}$
- More formally: $\exists i \cdot i \in r..n \wedge f(i) = v$

Refined events
Adding invariants

INVARIANTS

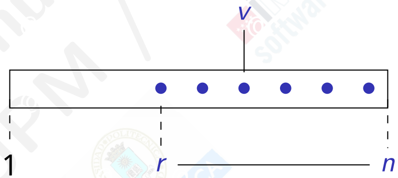
$\exists i \cdot i \in r..n \wedge f(i) = v$

Event INITIALISATION
    r := 1
end

Event Progress
    where f(r) $\neq$ v
then
    r := r + 1
end

Event Finish ...

- We know $v \in \mathrm{dom}(f)$.
- Express the idea that at some point we will find v.
- Hint: *at all times, v is in position r or to "its right".*



- Intuitively:
  $v \in \{f(r), f(r+1), \ldots, f(n-1), f(n)\}$
- More formally: $\exists i \cdot i \in r..n \wedge f(i) = v$

Refined events

Adding invariants to Rodin

INVARIANTS

$\exists\, i \cdot i \in r..n \land f(i) = v$

Event INITIALISATION

$r := 1$

end

Event Progress

where $f(r) \neq v$

then

$r := r + 1$

end

Event Finish ...

- Add invariant to Rodin.
  ($\land$ is written &, $\exists$ is #)
- Check proof obligations (left panel).
- In my case, *INITALIZATION/inv1/INV* and *Progress/act1/SIM* are not discharged.
  - YMMV: processor speed, tool version.
- *Prover view*: interact with theorem prover.
  - Navigate proof & applied inferences, add/remove hypotheses.
  - Invoke ext. th. provers (better, black box).
  - Check the *Proving* section of the web site.
  - For INV: PP works.
  - For SIM: simplify dom plus PP works.

Refined events

Adding invariants to Rodin

INVARIANTS

$v \in f[r..n]$

Event INITIALISATION

$r := 1$

end

Event Progress

where $f(r) \neq v$

then

$r := r + 1$

end

Event Finish ...

- Event B has a specific notation for $\exists i \cdot i \in 1..n \wedge f(i) = v$: $v \in f[r..n]$
- Substitute invariant in Rodin.
- Check Proof obligations.
- In my case, *Progress/inv1/INV* and *Progress/act1/SIM* are not discharged.
  - Again, YMMV.
- Double click on proof obligation, go to *Prover view*.
  - For Progress/inv1/INV: PP works.
  - For Progress/act1/SIM: simplify dom plus PP works.

INVARIANTS

$v \in f[r..n]$

Event INITIALISATION

$r := 1$

end

Event Progress

where $f(r) \neq v$

then

$r := r + 2$

end

Event Finish ...

- Introduce an error / mistake.
- Won't be able to discharge the POs for *Progress/inv1/INV, Progress/act1/SIM.*

Refined events

Termination

INVARIANTS

$v \in f[r..n]$

VARIANT

Event INITIALISATION

$r := 1$

end

Event Progress <convergent>

where $f(r) \neq v$

then

$r := r + 1$

end

Event Finish ...

- Termination is proven by defining an expression that:
  - Has a measure (an integer expression or a set with a well-defined size).
  - Has a lower bound.
  - Is reduced every time a *convergent* event is fired.

Refined events

Termination

INVARIANTS

$v \in f[r..n]$

VARIANT

Event INITIALISATION

$r := 1$

end

Event Progress <convergent>

where $f(r) \neq v$

then

$r := r + 1$

end

Event Finish ...

- Termination is proven by defining an expression that:
  - Has a measure (an integer expression or a set with a well-defined size).
  - Has a lower bound.
  - Is reduced every time a *convergent* event is fired.
- Used to:
  - Prove termination (in our case).
  - Prove absence of non-starvation / progress (concurrent systems).

Refined events

Termination

INVARIANTS

$v \in f[r..n]$

VARIANT

Event INITIALISATION

$r := 1$

end

Event Progress <convergent>

where $f(r) \neq v$

then

$r := r + 1$

end

Event Finish ...

- Termination is proven by defining an expression that:
  - Has a measure (an integer expression or a set with a well-defined size).
  - Has a lower bound.
  - Is reduced every time a *convergent* event is fired.
- Used to:
  - Prove termination (in our case).
  - Prove absence of non-starvation / progress (concurrent systems).
- Which expression could we use as variant?

Refined events

Termination

INVARIANTS

$v \in f[r..n]$

VARIANT

$n - r$

Event INITIALISATION

$r := 1$

end

Event Progress <convergent>

where $f(r) \neq v$

then

$r := r + 1$

end

Event Finish ...

- Termination is proven by defining an expression that:
  - Has a measure (an integer expression or a set with a well-defined size).
  - Has a lower bound.
  - Is reduced every time a *convergent* event is fired.
- Used to:
  - Prove termination (in our case).
  - Prove absence of non-starvation / progress (concurrent systems).
- Which expression could we use as variant?
- Add it to the model, check POs.

Formalized and proven

- The refinement is correct (no bugs introduced).
- Events maintain invariants.
- $v \in \mathrm{ran}(f) \Rightarrow$ **Progress** will always reach a position that contains v $\Rightarrow$ it is not enabled more than n times $\Rightarrow r \not> n \Rightarrow$ variant never becomes negative $\Rightarrow$ it is a natural number.
- Since **Progress** increases r, the variant decreases; as it has a lower bound, it will terminate.
- The model is deadlock free (Why?).
- The model is deterministic (Why?).

Sequential correctness

- Postcondition P must be true at the end of execution.
- End of execution associated to special event Finish:

$$A_{1\dots l}(c), I_{1\dots m}(v,c), G_{\mathsf{Finish}}(v,c) \vdash P(v,c)$$

correctness condition for integer division

$$\overbrace{b \in \mathbb{N}, c \in \mathbb{N}, c > 0}^{\text{Axioms}}, \overbrace{a \in \mathbb{N}, r \in \mathbb{N}, b = a \times c + r}^{\text{Invariants}}, \overbrace{r < c}^{\text{Guard}} \vdash \underbrace{b = a \times c + r \wedge r < c}_{\text{Postcond}}$$

- Note: in some cases there may be several Finish Events $\Rightarrow$ one sequent per event.
- Not applicable to non-terminating systems (other proofs required).
- $I_{1..n}$ and G_{Finish} related to postcondition; not necessarily identical.
- $I_{1\dots n}$ need to be *strong* enough.

Termination: formalization

- "*Postcondition P must be true **at the end** of execution*"
- General strategy: look for a *ranking function* that measures progress
- In Event B lingo: a *variant* $V(v, c)$
  - An expression V (with $V \in \mathbb{N}$ or $V \subseteq S$) that is reduced by each *non-terminating* event

$$A_{1\ldots l}(c), I_{1\ldots m}, G_i(v, c) \vdash V(v, c) > V(E_i(v, c), c)$$

  - We do not say how it is reduced: it has to be proven

Termination proof

$$\dfrac{\dfrac{}{c > 0 \vdash r > r - c}\ \text{Arith}}{b \in \mathbb{N}, c \in \mathbb{N}, c > 0, a \in \mathbb{N}, r \in \mathbb{N}, b = a \times c + r, r \geq c \vdash r > r - c}\ \text{Mon}$$

No deadlock, determinism

At least one guard must be true at any moment:

$$A_{1\ldots l}(v), I_{1\ldots m}(v,c) \vdash G_1(v,c) \lor G_2(v,c) \lor \ldots \lor G_m(v,c)$$

No two events can be active at the same time:

$$A_{1\ldots l}(v), I_{1\ldots m}(v,c) \vdash \bigwedge_{\substack{i,j=1\\ i\neq j}}^{n} \neg(G_i(v,c) \land G_j(v,c))$$

- In Rodin: add the RHS to the INVARIANTS.
- Usually, mark them as "theorem".
- A formula marked as theorem uses **only** the formulas (axioms, invariants, guards) in its scope that appear before it.
- Will see them with more detail later.

Well-definedness and feasibility

First machine (already seen)

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

- Proof Obligations
  - INITIALISATION/inv1/INV
  - INITIALISATION/act1/FIS
  - Search/inv1/INV
  - Search/act1/FIS
  - Finish/grd1/WD

- We (formally) know INV.
- Let us see WD and FIS in more detail.

WD (Well-Definedness)

- Ensuring that axioms, theorems, invariants, guards, actions, variants... are well-defined.
- I.e., all of their arguments "can be used". For example:

Expression	WD to prove
$f(E)$	$E \in dom(f)$
E/F	$F \neq 0$
$E \mod F$	$F \neq 0$
$card(S)$	$finite(S)$
$\min(S)$	$S \subseteq \mathbb{Z} \wedge \exists x \cdot x \in \mathbb{Z} \wedge (\forall n \cdot n \in S \Rightarrow x \leq n)$

- In our example: $v \neq f(r)$ needs $r \in dom(f)$.
- Formulas traversed to require WD of their components (with some special cases).

- Ensure that non-deterministic assignments $x :\in S$ are feasible.
- They are a particular case of *Before-After* predicates
- Relate values of variables before and after an action.

$$A_1(c), \ldots, A_m(c), I_1(c,v), \ldots, I_n(c,v), G(c,v) \vdash \exists v' \cdot BAP(v,c,v')$$

- v' becomes the next value of v after finishing the action.
- There has to be a value v' that makes the *BAP* true for any admissible value of v and c.
- Examples:

x := x - 1	is	$\exists x' \cdot x' = x - 1$
x :∈ S \ {x}	is	$\exists x' \cdot x' \in S \setminus \{x\}$

BAP and assignments

Notation: $\overline{v}$, $\overline{c}$ denote tuples of variables, constants.

- Before-after predicate:
  - $x :\in \{x | P(\overline{v}, \overline{c})\}$
    x one of the variables in $\overline{v}$.
  - $P(\overline{v}, \overline{c})$ needs to be true for some x.
  - Notation: $\overline{v}'$ is the "next value".
    $x :| \; x' = x + 7 \vee x' = x - 5$
- Non-deterministic assignment:
  - $x :\in S$ a shorthand for $x :| \; x' \in S$
  - S explicit, $S \neq \varnothing$
- Deterministic assignment:
  - $x := E(\overline{v}, \overline{c})$ a shorthand for
    $x :| \; x' = E(\overline{v}, \overline{c})$
  - E evaluates to a single value.

- More general invariant proof obligation:

$$A_{1..l}(\overline{c}), I_{1..m}(\overline{v}, \overline{c}), G_i(\overline{v}, \overline{c}), BAP(\overline{v}, \overline{c}, \overline{v}') \vdash I_j(\overline{v}', \overline{c})$$

For:

$x :| \; g(x') > 0$
$x :| \; g(x') > g(x)$

What are the WD and FIS POs?

Refinement: the sorted array case

Search in sorted array – specification

Preconditions

- A strictly positive number: $0 < n$.
- A sorted array f of n elements built on $\mathbb{N}$: $f \in 1..n \rightarrow \mathbb{N}$.
- A value v in the array: $v \in \mathrm{ran}(f)$.

$$n \in \mathbb{N}1$$
$$f \in 1..n \rightarrow \mathbb{N}$$
$$v \in \mathrm{ran}(f)$$

Postconditions

- r is an index of the array: $r \in dom(f)$.
- Such that $f(r) = v$.

To enter...	**type**
$\forall$	`!`
$\cdot$	`.`
$\mathbb{N}1$	`NAT1`

Q: *Sorted* charaterization

$$\forall i, j \cdot i \in 1..n \land j \in 1..n \land i \leq j \Rightarrow f(i) \leq f(j)$$

- Rodin: create `search-c1` extending `search-c0`, add axiom.

Variations on an invariant

We can write

$$\forall i, j \cdot i \in 1..n \land j \in 1..n \land i \leq j \Rightarrow f(i) \leq f(j) \tag{1}$$

But also

$$\forall i, j \cdot i \in 1..n \land j \in 1..n \land i < j \Rightarrow f(i) \leq f(j) \tag{2}$$

If $i = j$, of course $f(i) = f(j)$, so the $i = j$ case is superfluous. $i < j$ is stronger than $i \leq j$, because $i < j \Rightarrow i \leq j$. Which one is preferable?

Q: Which one should we prefer?

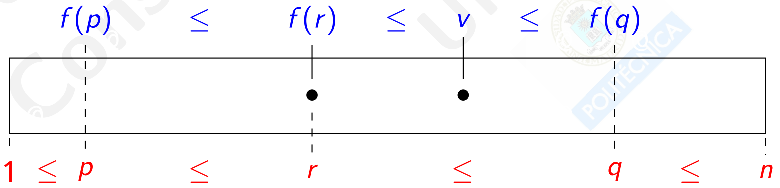
Both invariants are correct. But in general, we prefer stronger invariants. And (1) is stronger than (2)! They follow, resp., the scheme $A \Rightarrow C$ and $B \Rightarrow C$, and it happens that $B \Rightarrow A$. But the formula $(B \Rightarrow A) \Rightarrow ((A \Rightarrow C) \Rightarrow (B \Rightarrow C))$ is a tautology, while $(B \Rightarrow A) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C))$ is not. However, the latter case would happen if $i = j$ and $f(i) < f(j)$, which cannot happen in a function. So in reality, due to the semantics of functions (and with the help of suitable axioms), both will have the same power, but using the weaker version may require longer proofs.

Refinement

Add requirements (to the problem or how it is solved). The solution space shrinks. New models (rather, their states) must be contained in previous models.

Refining search

- First version: r randomly selected.
- Second version: r scans left-to-right.
- Refinement: narrow range of r around the position of v.
- Idea:
  - p and q ($p \leq q$) range so that $r \in p..q$, always.
  - r is chosen between p and q: $p \leq r \leq q$.
  - Depending on the position of $f(r)$ w.r.t. v, we update p or q.
  - Therefore we always keep $f(p) \leq f(r) \leq f(q)$ (remember $\forall i, j \cdot i \in \mathrm{dom}(f) \wedge j \in \mathrm{dom}(f) \wedge i \leq j \Rightarrow f(i) \leq f(j)$)



First Refinement

MACHINE BS_M1
REFINES BS_M0
SEES BS_C0
VARIABLES

r
p
q

INVARIANTS

inv1: $p \in 1\,..\,n$
inv2: $q \in 1\,..\,n$
inv3: $r \in p\,..\,q$
inv4: $v \in f[p\,..\,q]$

VARIANT

$q - p$

EVENTS

Initialisation
  begin
    act1: $p := 1$
    act2: $q := n$
    act3: $r :\in 1\,..\,n$
  end

Event final ⟨ordinary⟩ $\widehat{=}$
refines final
  when
    grd2: $f(r) = v$
  then
    skip
  end

Event inc ⟨convergent⟩ $\widehat{=}$
refines progress
  when
    grd1: $f(r) < v$
  then
    act2: $p := r + 1$
    act3: $r :\in r + 1\,..\,q$
  end

Event dec ⟨convergent⟩ $\widehat{=}$
refines progress
  when
    grd1: $f(r) > v$
  then
    act1: $q := r - 1$
    act2: $r :\in p\,..\,r - 1$
  end

END

convergent: VARIANT must decrease.

In RODIN: Do not mark events as "extended".

Q: Why does this model eventually find r?

If r not yet found, $q - p$ is decremented. Eventually, $q - p = 0$ and then $r = p = q$. At this moment, if the invariants hold, $f(r) = v$.

Proof Obligations

- INITIALISATION/inv1/INV
- INITIALISATION/inv2/INV
- INITIALISATION/inv3/INV
- INITIALISATION/inv4/INV

(Depending on the version of Rodin, of the theorem provers, and the speed of the computer)

- inc/grd1/WD
- inc/inv1/INV
- inc/inv3/INV
- inc/inv4/INV
- inc/grd1/GRD
- inc/act3/FIS
- inc/act1/SIM
- inc/VAR
- inc/NAT
- dec/grd1/WD
- dec/inv2/INV
- dec/inv3/INV
- dec/inv4/INV
- dec/grd1/GRD
- dec/act2/FIS
- dec/act1/SIM
- dec/VAR
- dec/NAT

Doing refinement right

The concrete model behaves as specified by the abstract model (i.e., concrete model does not exhibit any new behaviors)

To show this we have to prove that:

1. Transitions in the concrete model can not take place in states whose corresponding abstract state did not exhibit that transition (GRD).
2. Actions in concrete events cannot result in states that were not in the abstract model (SIM).

We will make these two conditions more precise and formalize them as proof obligations.

The Essence of GRD

Abstract model to (more) *concrete* model: details introduced

Abstract model

- Contains all correct states (at its level of description).
- Guards keep model from drifting into wrong states.

Concrete model: more details / more variables / richer state

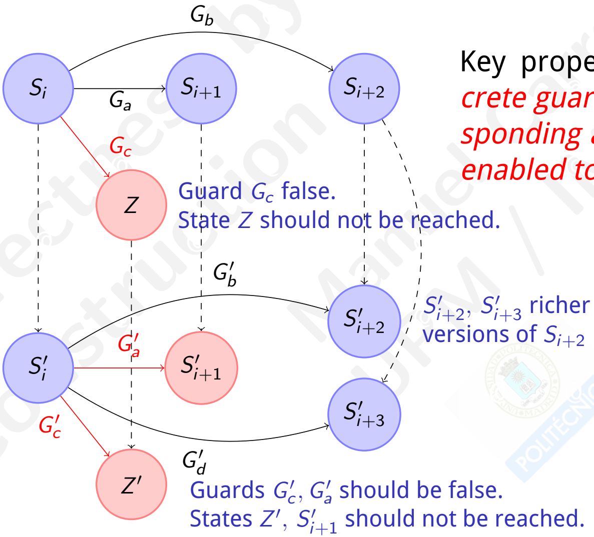
- Concrete and abstract states differ.
- A correspondence (“simulation”) must exist.
- Additional constraints may make some abstract states invalid in the concrete model: they must not be reachable (they disappear).
- Some abstract states *split* into several concrete states.

Initial model: r can move freely. Refinement: some histories removed.
All states / transitions in concrete model must exist in abstract model.

The Essence of GRD (Cont)

Abstract model

Concrete model (*primed* elements are concrete versions of abstract counterparts: S'_i **simulate** S_i)



Key property: *Whenever a concrete guard is enabled, the corresponding abstract guard must be enabled too, i.e.,* $G' \Rightarrow G$

The Essence of GRD (Cont)

$G'_b \Rightarrow G_b$ (and $G'_d \Rightarrow G_b$) A concrete transition was already valid in the abstract model (and $\top \Rightarrow \top$ is valid).

$G'_c \Rightarrow G_c$ A non-enabled concrete transition was not enabled in the abstract model (and $\perp \Rightarrow \perp$ is valid).

$G'_a \Rightarrow G_a$ A transition which was enabled in the abstract model cannot be taken any more because the destination state is not valid in the concrete model (and $\perp \Rightarrow \top$ is valid).

However, if G'_c were true in the concrete model, then $G'_c \Rightarrow G_c$ would be false, because $\top \Rightarrow \perp$ is not valid.

Non-reachable, incorrect states in abstract model would become reachable states in the concrete model – that is wrong.

- (Concrete) Guards in refining event stronger than guards in abstract event.
- Ensures that when concrete event enabled, so is the corresponding abstract event.
- For concrete “evt” and abstract guard “grd” in corresponding abstract event: **evt/grd/GRD**

Axioms Abstract Invariant Concrete Invariant Concrete Guard $\vdash$ Abstract Guard	$A(c)$ $I(c,v)$ $J(c,v,w)$ $H(c,w)$ $\vdash$ $G_i(c,v)$	GRD

Guard Strengthening Example

Event progress $\langle$anticipated$\rangle$ $\widehat{=}$
  when
    grd1: $f(r) \neq v$
  then
    act1: $r :\in dom(f)$
  end

Event inc $\langle$convergent$\rangle$ $\widehat{=}$
refines progress
  when
    grd1: $f(r) < v$
  then
    act2: $p := r + 1$
    act3: $r :\in r + 1 \,..\, q$
  end

- Is $f(r) < v$ more restrictive than $f(r) \neq v$?
- Yes: there are cases where $f(r) \neq v$ is true but $f(r) < v$ is not, and
- Whenever $f(r) < v$ is true, $f(r) \neq v$ is true as well.
- Therefore, $f(r) < v \Rightarrow f(r) \neq v$.

- Ensure that actions in concrete events *simulate* the corresponding abstract actions.
- Ensures that when the concrete event fires, it does not contradict the action of the corresponding abstract event.

(Ignore *witness predicate* W1, W2)

Axioms Abstract invariants and thms. Concrete invariants and thms. Concrete event guards witness predicate witness predicate Concrete before-after predicate $\vdash$ Abstract before-after predicate	$evt/act/$SIM

$$
\begin{array}{l}
A(s,c) \\
I(s,c,v) \\
J(s,c,v,w) \\
H(y,s,c,w) \\
W1(x,y,s,c,w) \\
W2(y,v',s,c,w) \\
BA2(w,w',\ldots) \\
\vdash \\
BA1(v,v',\ldots)
\end{array}
$$

SIM Example

$$
\begin{array}{l}
\textbf{Event} \ \ \text{progress} \ \langle\text{anticipated}\rangle \ \widehat{=} \\
\quad \textbf{when} \\
\qquad \texttt{grd1}: \quad f(r) \neq v \\
\quad \textbf{then} \\
\qquad \texttt{act1}: \ r :\in dom(f) \\
\quad \textbf{end}
\end{array}
$$

$$
\begin{array}{l}
\textbf{Event} \ \ \text{inc} \ \langle\text{convergent}\rangle \ \widehat{=} \\
\textbf{refines} \ \ \text{progress} \\
\quad \textbf{when} \\
\qquad \texttt{grd1}: \quad f(r') < v \\
\quad \textbf{then} \\
\qquad \texttt{act2}: \ p := r' + 1 \\
\qquad \texttt{act3}: \ r' :\in r' + 1 \mathrel{..} q \\
\quad \textbf{end}
\end{array}
$$

Are the states created by $r' :\in r' + 1..q$ *inside* the states created by $r :\in \mathrm{dom}(f)$?

- Yes. Intuitively: $p..q \subseteq \mathrm{dom}(f)$ deduced from invariant. Any choice made by $r' :\in p..q$ could also be done by $r \in \mathrm{dom}(f)$.

Rodin and the Second Refinement

Create new machine, input previous refinement, check what proofs are automatically discharged

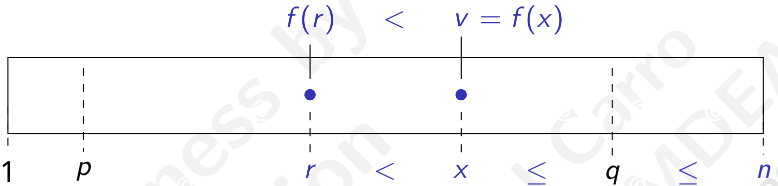
What theorem provers did (last time I tried :-):

inc/inv1/INV	Automatically discharged by PP (takes time)
inc/inv4/INV	Automatically discharged by PP
inc/act3/FIS	Needs interaction
dec/inv2/INV	Automatically discharged by PP
dec/inv4/INV	Needs interaction
dec/act2/FIS	Needs interaction

Reasons for differences:

- Timeout, different speed
- Different prover versions
- Provers use heuristics: slight changes may facilitate proofs (in general, keeping the set of hypotheses small is good)
- Rodin keeps cache of proofs, reuses (can be purged)

inc/inv1/INV



inv1 $p \in 1..n$

Action $p := r+1, r :\in r+1..q$

Goal (inv. after) $r+1 \in 1..n$ (with r the value **before** the action)

- We had $r \in 1..n$ before; just prove $r < n$.

Strategy $v \in ran(f)$; say $f(x) = v$. As $dom(f) = 1..n$, $1 \leq x \leq n$. Since $f(r) < v = f(x)$, $r < x$ (monotonically sorted array). Therefore $r < x \leq n$ and $r < n$.

Sketch of a Proof for inc/inv1/INV

Available hypotheses and goal

$$r \in dom(f)$$

$$\forall i, j \cdot (i \in dom(f) \land j \in dom(f) \land i \leq j) \Rightarrow f(i) \leq f(j)$$

$$f(r) < v$$

$$v \in ran(f)$$

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

$$\vdash r < n$$

LHS of sequent

Sketch of a Proof for inc/inv1/INV

Available hypotheses and goal

$r \in dom(f)$

$\forall i, j \cdot (i \in dom(f) \land j \in dom(f) \land i \leq j) \Rightarrow f(i) \leq f(j)$

$f(r) < v$

$v \in ran(f)$

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

$\vdash r < n$

LHS of sequent

$\forall i, j \cdot f(i) > f(j) \Rightarrow (i \notin dom(f) \lor j \notin dom(f) \lor i > j)$

Contraposition of implication

Sketch of a Proof for inc/inv1/INV

Available hypotheses and goal

$$r \in dom(f)$$

$$\forall i, j \cdot (i \in dom(f) \land j \in dom(f) \land i \leq j) \Rightarrow f(i) \leq f(j)$$

$$f(r) < v$$

$$v \in ran(f)$$

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

$$\vdash r < n$$

LHS of sequent

$$\forall i, j \cdot f(i) > f(j) \Rightarrow (i \notin dom(f) \lor j \notin dom(f) \lor i > j)$$

$$\forall i \cdot f(i) > f(r) \Rightarrow (i \notin dom(f) \lor r \notin dom(f) \lor i > r)$$

Instantiate j with r

Sketch of a Proof for inc/inv1/INV

Available hypotheses and goal

$$r \in dom(f)$$

$$\forall i, j \cdot (i \in dom(f) \land j \in dom(f) \land i \leq j) \Rightarrow f(i) \leq f(j)$$

$$f(r) < v$$

$$v \in ran(f)$$

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

$$\vdash r < n$$

LHS of sequent

$$\forall i, j \cdot f(i) > f(j) \Rightarrow (i \notin dom(f) \lor j \notin dom(f) \lor i > j)$$

$$\forall i \cdot f(i) > f(r) \Rightarrow (i \notin dom(f) \lor r \notin dom(f) \lor i > r)$$

$$\exists a \cdot a \in dom(f) \land f(a) = v$$

Since $v \in ran(f)$

Sketch of a Proof for inc/inv1/INV

Available hypotheses and goal

$$r \in dom(f)$$

$$\forall i, j \cdot (i \in dom(f) \land j \in dom(f) \land i \leq j) \Rightarrow f(i) \leq f(j)$$

$$f(r) < v$$

$$v \in ran(f)$$

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

$$\vdash r < n$$

LHS of sequent

$$\forall i, j \cdot f(i) > f(j) \Rightarrow (i \notin dom(f) \lor j \notin dom(f) \lor i > j)$$

$$\forall i \cdot f(i) > f(r) \Rightarrow (i \notin dom(f) \lor r \notin dom(f) \lor i > r)$$

$$\exists a \cdot a \in dom(f) \land f(a) = v$$

$$f(a) > f(r) \Rightarrow (a \notin dom(f) \lor r \notin dom(f) \lor a > r)$$

Instantiate i in universally quantified formula with a

Sketch of a Proof for inc/inv1/INV

Available hypotheses and goal

$$r \in dom(f)$$

$$\forall i, j \cdot (i \in dom(f) \land j \in dom(f) \land i \leq j) \Rightarrow f(i) \leq f(j)$$

$$f(r) < v$$

$$v \in ran(f)$$

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

$$\vdash r < n$$

LHS of sequent

$$\forall i, j \cdot f(i) > f(j) \Rightarrow (i \notin dom(f) \lor j \notin dom(f) \lor i > j)$$

$$\forall i \cdot f(i) > f(r) \Rightarrow (i \notin dom(f) \lor r \notin dom(f) \lor i > r)$$

$$\exists a \cdot a \in dom(f) \land f(a) = v$$

$$f(a) > f(r) \Rightarrow (a \notin dom(f) \lor r \notin dom(f) \lor a > r)$$

$$v > f(r) \Rightarrow (a \notin dom(f) \lor r \notin dom(f) \lor a > r)$$

Substitute $f(a)$ for v

Sketch of a Proof for inc/inv1/INV

Available hypotheses and goal

$$r \in dom(f)$$

$$\forall i, j \cdot (i \in dom(f) \land j \in dom(f) \land i \leq j) \Rightarrow f(i) \leq f(j)$$

$$f(r) < v$$

$$v \in ran(f)$$

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

$$\vdash r < n$$

LHS of sequent

$$\forall i, j \cdot f(i) > f(j) \Rightarrow (i \notin dom(f) \lor j \notin dom(f) \lor i > j)$$

$$\forall i \cdot f(i) > f(r) \Rightarrow (i \notin dom(f) \lor r \notin dom(f) \lor i > r)$$

$$\exists a \cdot a \in dom(f) \land f(a) = v$$

$$f(a) > f(r) \Rightarrow (a \notin dom(f) \lor r \notin dom(f) \lor a > r)$$

$$v > f(r) \Rightarrow (a \notin dom(f) \lor r \notin dom(f) \lor a > r)$$

$$a \notin dom(f) \lor r \notin dom(f) \lor a > r$$

Deduce consequent since antecedent is true

Sketch of a Proof for inc/inv1/INV

Available hypotheses and goal

$$r \in dom(f)$$

$$\forall i, j \cdot (i \in dom(f) \land j \in dom(f) \land i \leq j) \Rightarrow f(i) \leq f(j)$$

$$f(r) < v$$

$$v \in ran(f)$$

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

$$\vdash r < n$$

LHS of sequent

$$\forall i, j \cdot f(i) > f(j) \Rightarrow (i \notin dom(f) \lor j \notin dom(f) \lor i > j)$$

$$\forall i \cdot f(i) > f(r) \Rightarrow (i \notin dom(f) \lor r \notin dom(f) \lor i > r)$$

$$\exists a \cdot a \in dom(f) \land f(a) = v$$

$$f(a) > f(r) \Rightarrow (a \notin dom(f) \lor r \notin dom(f) \lor a > r)$$

$$v > f(r) \Rightarrow (a \notin dom(f) \lor r \notin dom(f) \lor a > r)$$

$$a \notin dom(f) \lor r \notin dom(f) \lor a > r$$

$$r \notin dom(f) \lor a > r$$

$a \notin dom(f)$ is not true by definition

Sketch of a Proof for inc/inv1/INV

Available hypotheses and goal

$r \in dom(f)$

$\forall i, j \cdot (i \in dom(f) \wedge j \in dom(f) \wedge i \leq j) \Rightarrow f(i) \leq f(j)$

$f(r) < v$

$v \in ran(f)$

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

$\vdash r < n$

LHS of sequent

$\forall i, j \cdot f(i) > f(j) \Rightarrow (i \notin dom(f) \vee j \notin dom(f) \vee i > j)$

$\forall i \cdot f(i) > f(r) \Rightarrow (i \notin dom(f) \vee r \notin dom(f) \vee i > r)$

$\exists a \cdot a \in dom(f) \wedge f(a) = v$

$f(a) > f(r) \Rightarrow (a \notin dom(f) \vee r \notin dom(f) \vee a > r)$

$v > f(r) \Rightarrow (a \notin dom(f) \vee r \notin dom(f) \vee a > r)$

$a \notin dom(f) \vee r \notin dom(f) \vee a > r$

$r \notin dom(f) \vee a > r$

$a > r$

$r \notin dom(f)$ is not true by definition

Sketch of a Proof for inc/inv1/INV

Available hypotheses and goal

$$r \in dom(f)$$

$$\forall i, j \cdot (i \in dom(f) \land j \in dom(f) \land i \leq j) \Rightarrow f(i) \leq f(j)$$

$$f(r) < v$$

$$v \in ran(f)$$

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

$$\vdash r < n$$

LHS of sequent

$$\forall i, j \cdot f(i) > f(j) \Rightarrow (i \notin dom(f) \lor j \notin dom(f) \lor i > j)$$

$$\forall i \cdot f(i) > f(r) \Rightarrow (i \notin dom(f) \lor r \notin dom(f) \lor i > r)$$

$$\exists a \cdot a \in dom(f) \land f(a) = v$$

$$f(a) > f(r) \Rightarrow (a \notin dom(f) \lor r \notin dom(f) \lor a > r)$$

$$v > f(r) \Rightarrow (a \notin dom(f) \lor r \notin dom(f) \lor a > r)$$

$$a \notin dom(f) \lor r \notin dom(f) \lor a > r$$

$$r \notin dom(f) \lor a > r$$

$$a > r$$

$$a \leq n$$

a is within the domain of f

Sketch of a Proof for inc/inv1/INV

Available hypotheses and goal

$$r \in dom(f)$$

$$\forall i, j \cdot (i \in dom(f) \land j \in dom(f) \land i \leq j) \Rightarrow f(i) \leq f(j)$$

$$f(r) < v$$

$$v \in ran(f)$$

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

$$\vdash r < n$$

LHS of sequent

$$\forall i, j \cdot f(i) > f(j) \Rightarrow (i \notin dom(f) \lor j \notin dom(f) \lor i > j)$$

$$\forall i \cdot f(i) > f(r) \Rightarrow (i \notin dom(f) \lor r \notin dom(f) \lor i > r)$$

$$\exists a \cdot a \in dom(f) \land f(a) = v$$

$$f(a) > f(r) \Rightarrow (a \notin dom(f) \lor r \notin dom(f) \lor a > r)$$

$$v > f(r) \Rightarrow (a \notin dom(f) \lor r \notin dom(f) \lor a > r)$$

$$a \notin dom(f) \lor r \notin dom(f) \lor a > r$$

$$r \notin dom(f) \lor a > r$$

$$a > r$$

$$a \leq n$$

$$r < n$$

Putting together both inequalities

Sketch of a Proof for inc/inv1/INV

Available hypotheses and goal

$$r \in dom(f)$$

$$\forall i, j \cdot (i \in dom(f) \wedge j \in dom(f) \wedge i \leq j) \Rightarrow f(i) \leq f(j)$$

$$f(r) < v$$

$$v \in ran(f)$$

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

$$\vdash r < n$$

LHS of sequent

$$\forall i, j \cdot f(i) > f(j) \Rightarrow (i \notin dom(f) \vee j \notin dom(f) \vee i > j)$$

$$\forall i \cdot f(i) > f(r) \Rightarrow (i \notin dom(f) \vee r \notin dom(f) \vee i > r)$$

$$\exists a \cdot a \in dom(f) \wedge f(a) = v$$

$$f(a) > f(r) \Rightarrow (a \notin dom(f) \vee r \notin dom(f) \vee a > r)$$

$$v > f(r) \Rightarrow (a \notin dom(f) \vee r \notin dom(f) \vee a > r)$$

$$a \notin dom(f) \vee r \notin dom(f) \vee a > r$$

$$r \notin dom(f) \vee a > r$$

$$a > r$$

$$a \leq n$$

$$r < n$$

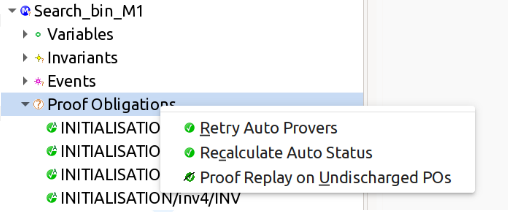
Goal achieved

Proving dec/inv4/INV in Rodin

- Double click on undischarged proof, switch to proving perspective.
- Show all hypothesis (click on search button).
- Select the hypothesis in the previous slide.
- Click on the **+** button in the tab of the 'Search hypotheses' window. They should now appear under 'Selected hypotheses'.
- Invert implication inside universal quantifier.
- Instantiate j to be r.
- Click on the `P0` button (*proof on selected hypothesis*) in the 'Proof Control' window.
  - This will try to prove the goal using only the selected hypotheses; it can then explore much deeper, since we are using only a subset of the existing hypotheses and we have fixed a value in the universal quantifier.
- Almost immediately, a green face should appear.
- Save the proof status (`Ctrl-s`) to update the proof status.

Notes and Hints on Discharging Proofs with RODIN

- Your results may differ.
  - Timeout-bound: non-decidable task.
  - Speed may cause differences.
  - Third-party theorem provers: versions may behave differently.
  - Search heuristics: sensitive to details (may open unneeded search paths).
- External theorem provers black boxes.
  - How they discharged proofs unknown to Rodin.
- Changing axioms, invariants, guards in principle invalidate proofs where they appear as hypotheses.
- Proofs saved and reused.
  - History may impact behavior.
  - Right-click on proofs to retry them:



Notes and Hints on Discharging Proofs with RODIN

- Labels (act2, inv1, etc.) depend on how model is written.
- From Atelier B: NewPP, PP, ML.
  - Other theorem provers available.
- Do **not** use `NewPP`: it's unsound.
- `PP` weak with WD: $\vdash b \in f^{-1}[\{f(b)\}]$ not discharged.
- It may not discharge easy proofs if unneeded hypothesis present.
- `ML` useful for arithmetic-based reasoning, weaker with sets.
- To test: **copy** project, work on copied project.
- Removing project: **select** `Delete from hard disk`.
- POs can be *accepted* with . Flagged *reviewed* to temporarily continue or because they were manually proved.

For more, useful information, please check:

- The Rodin and Proving sections of the course web site.
- https://www3.hhu.de/stups/handbook/rodin/current/html/atelier_b_provers.html
- https://www3.hhu.de/stups/handbook/rodin/current/html/proving_perspective.html

Reusing formulas

- Reusing formulas deducible from axioms is sometimes handy.
- In our examples we very often transformed

$$\forall i, j \cdot (i \in dom(f) \land j \in dom(f) \land i \leq j) \Rightarrow f(i) \leq f(j)$$

into the logically equivalent

$$\forall i, j \cdot f(i) < f(j) \Rightarrow (i \notin dom(f) \lor j \notin dom(f) \lor i < j)$$

- We can add the latter to the model to save clicks.
- It could be an **axiom**.
- But axioms should not be redundant.
  - If we update one axioms, but not one of its versions, the model could be inconsistent.

Theorems

- Rodin offers theorems: a formula that can be proven from others in the same class.

```
AXIOMS
  ∘  axm1:  ∀i,j·(i ∈ dom(f) ∧ j ∈ dom(f) ∧ i ≤ j ⇒ f(i) ≤ f(j)) not theorem ›
  ∘  axm2:  ∀i,j·(f(i) > f(j) ⇒ (i ∉ dom(f) ∨ j ∉ dom(f) ∨ i > j)) theorem ›
```

```
INVARIANTS
  ∘  inv1:  ∀n·n∈N ∧ n≠r ⇒ d(n)≤d(f(n)) not theorem ›
  ∘  inv2:  ∀n·n∈N ⇒ c(n)∈d(n)‥d(n)+1 not theorem ›
  ∘  thm1:  d(r) ≤ c(r) theorem ›
  ∘  thm2:  ∀n·n∈N ⇒ d(n)≤d(r) theorem ›
```

- Simplify proofs.
- Similar to lemmas in maths.
- Help provers (sometimes necessary).
- They need to be proved!

Proving theorems

- For a theorem “thm”, the name of its PO is **thm/THM**.
- Proved as usual.

Axioms $\vdash$ Theorem	thm/THM

$$\begin{array}{l} \quad A(s,c) \\ \vdash \\ \quad P(s,c) \end{array}$$

- For a theorem that requires an invariant: Axioms + Invariants
- Has to be placed after the axioms / invariants needed.

The strange case of the un-(well-defined) theorem

axm2 : $\forall i, j \cdot f(i) < f(j) \Rightarrow$
$(i \notin dom(f) \lor j \notin dom(f) \lor i < j)$

- Proof Obligations
  - axm1/WD
  - axm2/WD
  - axm2/THM

- Why? It is equivalent! Any idea?

The strange case of the un-(well-defined) theorem

$$\text{axm2} : \forall i,j \cdot f(i) < f(j) \Rightarrow$$
$$(i \notin dom(f) \lor j \notin dom(f) \lor i < j)$$

- Proof Obligations
  - axm1/WD
  - axm2/WD
  - axm2/THM

- Why? It is equivalent! Any idea?
- Proof explorer: is $f(i)$ valid?
- WD for implications *(ordered WD)*:
  $WD(P \Rightarrow Q) \equiv WD(P) \land P \Rightarrow WD(Q)$
- Treats P as a "domain" property.

- Workaround: instead of

$$\forall i,j \cdot (i \in dom(f) \land j \in dom(f) \land i \leq j)$$
$$\Rightarrow f(i) \leq f(j)$$

use

$$\forall i,j \cdot (i \in dom(f) \land j \in dom(f)) \Rightarrow$$
$$(i \leq j \Rightarrow f(i) \leq f(j))$$

Will that be equivalent?

The strange case of the un-(well-defined) theorem

$$\text{axm2} : \forall i, j \cdot f(i) < f(j) \Rightarrow$$
$$(i \notin dom(f) \lor j \notin dom(f) \lor i < j)$$

- Proof Obligations
  - axm1/WD
  - axm2/WD
  - axm2/THM

- Why? It is equivalent! Any idea?
- Proof explorer: is $f(i)$ valid?
- WD for implications *(ordered WD)*:
  $WD(P \Rightarrow Q) \equiv WD(P) \land P \Rightarrow WD(Q)$
- Treats P as a "domain" property.

- Workaround: instead of

$$\forall i, j \cdot (i \in dom(f) \land j \in dom(f) \land i \leq j)$$
$$\Rightarrow f(i) \leq f(j)$$

use

$$\forall i, j \cdot (i \in dom(f) \land j \in dom(f)) \Rightarrow$$
$$(i \leq j \Rightarrow f(i) \leq f(j))$$

Will that be equivalent?

- Contrapositive:

$$\forall i, j \cdot (i \in dom(f) \land j \in dom(f)) \Rightarrow$$
$$(f(i) > f(j) \Rightarrow i > j)$$