Sequential programs, refinement, and proof obligations[1]

Manuel Carro
manuel.carro@upm.es

IMDEA Software Institute &
Universidad Politécnica de Madrid

[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

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

Constants: n, f, v
Axiom 1: $n \in \mathbb{N}1$
Axiom 2: $f \in 1..n \rightarrow \mathbb{N}$
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 ∈ dom(f)
INIT
   r :∈ dom(f)
END

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

EVENT Progress
   WHERE f(r) ≠ v
   THEN
      r :∈ 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`
$\rightarrow$	`-->`
$\neq$	`/=`

$f \in \mathbb{N} \rightarrow 1..n$ 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 $\mathrm{dom}(f)$?)
- Some help from more powerful theorem provers 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.

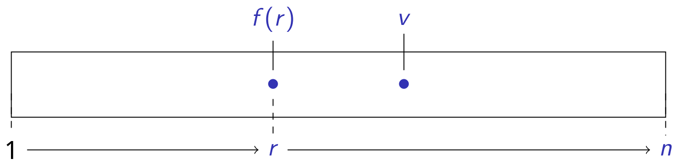
Refinement

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.



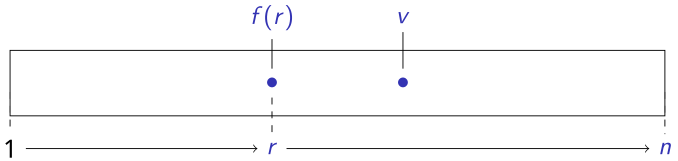
Refinement

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

```
Event INITIALISATION
    r := 1
end

Event Finish
    where f(r) = v
end

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


Refined events

```
Event INITIALISATION
    r := 1
end

Event Finish
    where f(r) = v
end

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


- *Histories* of refined model: subset of abstract's.
- No new behavior introduced $\Longrightarrow$ correctness preserved.
- Preservation of $r \in \mathrm{dom}(f)$ cannot be proven.
- Invariant too weak, it is true in forbidden states $\Longrightarrow$ strengthen them!
- $v \in f[r..n]$
- $f[p..q]$: image of f for the set $p..q$.
- **variant**: bounded expression that decreases for all convergent events.

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 won't be $> n \Rightarrow$ variant never becomes negative $\Rightarrow$ it is a natural number.
- Since **Progress** decreases the variant and it has a lower bound, it will terminate.
- Since guards are the negation of each other:
  - The model is deadlock free.
  - The events exclude each other (the model is deterministic)

Sequential correctness

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

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

Q: correctness condition for integer division

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

- Not applicable to non-terminating systems (other proofs required)
- $I_{1..n}$ and G_{Finish} related to P; not necessarily identical
- $I_{1\ldots n}$ need to be *strong* enough.

Termination

- "*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:

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

No two events can be active at the same time:

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

In Rodin: add the RHS in the INVARIANTS section, mark them as "theorem".

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 $f(r) \neq v$
  $r :\in \mathrm{dom}(f)$
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 "exist". For example:

Expression	WD to prove
$f(E)$	f is a partial function and $E \in dom(f)$
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).

FIS

Ensure that non-deterministic actions are feasible.

Axioms Invariants Guards of the event $\vdash$ $\exists v' \cdot$ Before-after predicate	$evt/act/$FIS

$$\begin{array}{l} A(s,c) \\ I(s,c,v) \\ G(x,s,c,v) \\ \vdash \\ \exists v' \cdot BAP(x,s,c,v,v') \end{array}$$

- In $G(x,s,c,v)$: x event parameters, s are carrier sets — not yet seen.
- $BAP(x,s,c,v,v')$: *Before-After* predicate (next).

BAP and non-deterministic assignments

- Simple assignment:
  - $v := E(v,c)$.
  - E evaluates to a single value.
- Non-deterministic assignment:
  - $v :\in S$
  - S explicit, $S \neq \varnothing$ — FIS PO.
  - E.g., $v :\in 1..n$ needs $n \geq 1$.
- Before-after predicate:
  - $x :\in \{x | P(v,c)\}$
    x one of the variables in v.
  - $P(v,c)$ needs to be true for some x.
  - Notation: v' is the "next value".
    $x :| \ x' = x + 7 \vee x' = x - 5$

- $BAP(v,v',c)$ generalizes $v := E(v,c)$.
- More general invariant proof obligation:

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

For:

$x :| \ g(x') > 0$
$x :| \ g(x') > g(x)$
$x :| \ g(x') > \frac{1}{g(x)}$

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 $\mathbb{N}: f \in 1..n \to \mathbb{N}$.
- A value v in the array: $v \in \mathrm{ran}(f)$.

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

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* invariant

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

Variations on an invariant

We can write

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

But also

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

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

Q: Which one should we prefer?

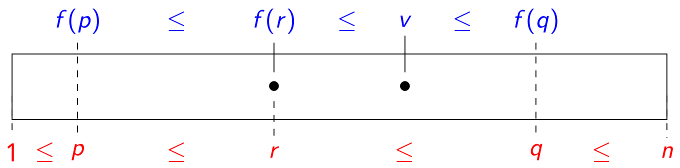
Both invariants are correct. But in general, we prefer stronger invariants. And (1) is stronger than (2)! They follow, resp., the scheme $a \Rightarrow c$ and $b \Rightarrow c$, and it happens that $b \Rightarrow a$. But the formula $(b \Rightarrow a) \Rightarrow ((a \Rightarrow c) \Rightarrow (b \Rightarrow c))$ is valid, while $(b \Rightarrow a) \Rightarrow ((b \Rightarrow c) \Rightarrow (a \Rightarrow c))$ is not.

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

- r "shoots" indiscriminately.
- 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 $\forall i, j \cdot i \in \mathrm{dom}(f) \land j \in \mathrm{dom}(f) \land i \leq j \Rightarrow f(i) \leq f(j)$)



First Refinement

```
MACHINE BS_M1
REFINES BS_M0
SEES BS_C0
VARIABLES
    r
    p
    q
INVARIANTS
    inv1: p ∈ 1 .. n
    inv2: q ∈ 1 .. n
    inv3: r ∈ p .. q
    inv4: v ∈ f[p .. q]
VARIANT
    q − p
EVENTS
Initialisation
    begin
        act1: p := 1
        act2: q := n
        act3: r :∈ 1 .. n
    end
Event final ⟨ordinary⟩ ≙
refines final
    when
        grd2: f(r) = v
    then
        skip
    end
Event inc ⟨convergent⟩ ≙
refines progress
    when
        grd1: f(r) < v
    then
        act2: p := r + 1
        act3: r :∈ r + 1 .. q
    end
Event dec ⟨convergent⟩ ≙
refines progress
    when
        grd1: f(r) > v
    then
        act1: q := r − 1
        act2: r :∈ 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 and the theorem provers)

- 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

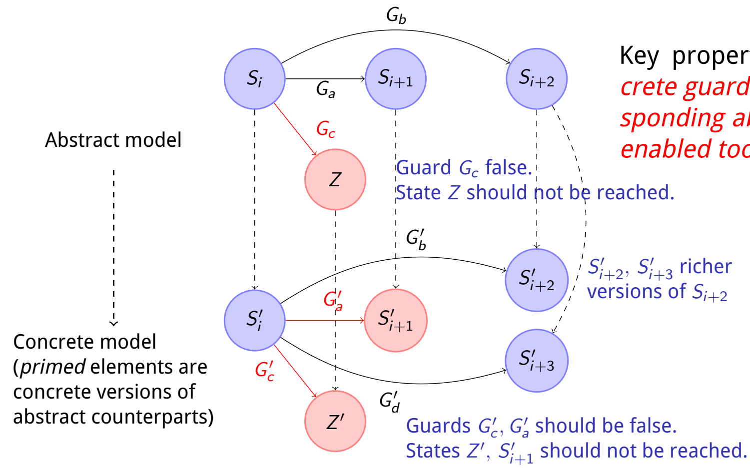
- Contains all correct states.
- Guards keep model from drifting into wrong states.

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: not all histories possible. But all states / transitions in refined model contained in abstract model.

The Essence of GRD (Cont)



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 be *transformed* into reachable states in the concrete model, which is wrong.

GRD

- (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 ⟨anticipated⟩ $\widehat{=}$
  when
    grd1: $f(r) \neq v$
  then
    act1: $r :\in dom(f)$
  end

Event inc ⟨convergent⟩ $\widehat{=}$
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, $f(r) < v \Rightarrow f(r) \neq v$.

SIM

- 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

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

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

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 $r \in \mathrm{dom}(f)$.

SIM, More Formally

$$
\begin{array}{l}
n \in \mathbb{N}1 \\
f \in 1..n \longrightarrow \mathbb{N} \\
r \in \mathrm{dom}(f) \\
p \in 1..n \\
q \in 1..n \\
r \in p..q \\
v \in f[p..q] \\
f(r) < v \\
\forall i, j \cdot i \in 1..n \wedge j \in 1..n \wedge i \leq j \Rightarrow f(i) \leq f(j) \\
r' \in r + 1..q \\
\vdash \\
r' \in \mathrm{dom}(f)
\end{array}
$$

- Can you find a proof (by contradiction)?

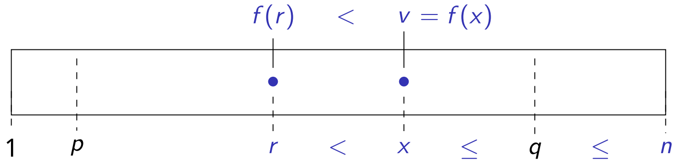
Rodin and the Second Refinement

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

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

inc/inv1/INV	PP, ML timeout: needs interaction
inc/inv4/INV	Automatically discharged by PP
inc/act3/FIS	Needs interaction
dec/inv2/INV	Needs interaction
dec/inv4/INV	Needs interaction
dec/act2/FIS	Needs interaction

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

$r \in dom(f)$

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

$f(r) < v$

$v \in ran(f)$

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

$\vdash r+1 \in 1..n$

Left: selected hypothesis and goal.

Right: rewritings of the LHS of the sequent.

Sketch of a Proof for inc/inv1/INV

$r \in dom(f)$

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

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

$f(r) < v$

$v \in ran(f)$

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

$\vdash r+1 \in 1..n$

Left: selected hypothesis and goal.

Right: rewritings of the LHS of the sequent.

Sketch of a Proof for inc/inv1/INV

$r \in dom(f)$

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

$f(r) < v$

$v \in ran(f)$

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

$\vdash r + 1 \in 1..n$

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

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

Left: selected hypothesis and goal.

Right: rewritings of the LHS of the sequent.

Sketch of a Proof for inc/inv1/INV

$r \in dom(f)$

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

$f(r) < v$

$v \in ran(f)$

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

$\vdash r + 1 \in 1..n$

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

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

$x \mapsto v \in f$

Left: selected hypothesis and goal.

Right: rewritings of the LHS of the sequent.

Sketch of a Proof for inc/inv1/INV

$r \in dom(f)$

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

$f(r) < v$

$v \in ran(f)$

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

$\vdash r + 1 \in 1..n$

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

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

$x \mapsto v \in f$

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

Left: selected hypothesis and goal.

Right: rewritings of the LHS of the sequent.

Sketch of a Proof for inc/inv1/INV

$r \in dom(f)$

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

$f(r) < v$

$v \in ran(f)$

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

$\vdash r + 1 \in 1..n$

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

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

$x \mapsto v \in f$

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

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

Left: selected hypothesis and goal.

Right: rewritings of the LHS of the sequent.

Sketch of a Proof for inc/inv1/INV

$r \in dom(f)$

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

$f(r) < v$

$v \in ran(f)$

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

$\vdash r + 1 \in 1..n$

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

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

$x \mapsto v \in f$

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

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

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

Left: selected hypothesis and goal.

Right: rewritings of the LHS of the sequent.

Sketch of a Proof for inc/inv1/INV

$r \in dom(f)$

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

$f(r) < v$

$v \in ran(f)$

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

$\vdash r + 1 \in 1..n$

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

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

$x \mapsto v \in f$

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

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

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

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

Left: selected hypothesis and goal.

Right: rewritings of the LHS of the sequent.

Sketch of a Proof for inc/inv1/INV

$r \in dom(f)$

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

$f(r) < v$

$v \in ran(f)$

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

$\vdash r + 1 \in 1..n$

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

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

$x \mapsto v \in f$

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

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

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

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

$x > r$

Left: selected hypothesis and goal.

Right: rewritings of the LHS of the sequent.

Sketch of a Proof for inc/inv1/INV

$r \in dom(f)$

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

$f(r) < v$

$v \in ran(f)$

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

$\vdash r + 1 \in 1..n$

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

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

$x \mapsto v \in f$

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

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

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

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

$x > r$

$x \leq n$

Left: selected hypothesis and goal.

Right: rewritings of the LHS of the sequent.

Sketch of a Proof for inc/inv1/INV

$r \in dom(f)$

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

$f(r) < v$

$v \in ran(f)$

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

$\vdash r + 1 \in 1..n$

Left: selected hypothesis and goal.

Right: rewritings of the LHS of the sequent.

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

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

$x \mapsto v \in f$

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

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

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

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

$x > r$

$x \leq n$

$r < n$

Sketch of a Proof for inc/inv1/INV

$r \in dom(f)$

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

$f(r) < v$

$v \in ran(f)$

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

$\vdash r + 1 \in 1..n$

Left: selected hypothesis and goal.

Right: rewritings of the LHS of the sequent.

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

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

$x \mapsto v \in f$

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

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

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

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

$x > r$

$x \leq n$

$r < n$

$r + 1 \leq n$

Proving inc/inv1/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 on Discharging Proofs with RODIN

- Different versions may behave differently.
- Search heuristics. Sensitive to details.
- Proof parts saved and reused. Behavior may change depending on history.
- Labels (act2, inv1, etc.) depend on how model is written.
- Do **not** use the `NewPP` prover: 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.
- See `https://www3.hhu.de/stups/handbook/rodin/current/html/atelier_b_provers.html` and `https://www3.hhu.de/stups/handbook/rodin/current/html/proving_perspective.html`.
- To test: **copy** project, work on copied project.
- When removing, tick on `Delete from hard disk`.

Reviewed Hypothesis

POs can be *accepted* with Ⓡ. Flagged *reviewed* to temporarily continue or because they were manually proved.

Reusing formulas

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

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

into the logically equivalent

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

- 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 but not a version of it, 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.
- 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 ⊢ Theorem	thm/THM

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

- 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

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 correct?

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?
- 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 correct?

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

Type checking and mathematical proofs

Types

- Determine types correct.
- Based on function types + typing rules.

$f(x : \mathbb{R}) : \mathbb{R}$
return $x * 3.5$

$g(x : \mathbb{R}) : \mathbb{N}$
return $x * 3.5$

Theorems

- Determine formula valid.
- Hypotheses + deduction rules.

$x \in \mathbb{R} \vdash x * 3.5 \in \mathbb{R}$

$x \in \mathbb{R} \vdash x * 3.5 \in \mathbb{N}$

Type checking and mathematical proofs

Types	Theorems
• Determine types correct.	• Determine formula valid.
• Based on function types + typing rules.	• Hypotheses + deduction rules.
$f(x: \mathbb{R}): \mathbb{R}$ return $x * 3.5$	$x \in \mathbb{R} \vdash x * 3.5 \in \mathbb{R}$
$g(x: \mathbb{R}): \mathbb{N}$ return $x * 3.5$	$x \in \mathbb{R} \vdash x * 3.5 \in \mathbb{N}$

- Traditional type checking: weak theorem proving.
- Type checking rules basically **same** as logic inference rules.
- Most type systems decidable, efficient.

Type checking and mathematical proofs

- Highly expressive type systems (Liquid Haskell, Agda, Idris):
  - More properties captured

    length(concat(a, b)) = length(a) + length(b)
  - Decidability can be challenged.
    - E.g., ML type system.
- Some frameworks give up.
- Others allow user intervention
  - Dafny, Coq: help adding invariants, lemmas
    - If found, proof is *black box*.
  - Event B:
    - In addition, user intervention at the proof level.
    - Full expressiveness in properties.