Sequential Binary Search

Manuel Carro
manuel.carro@upm.es

IMDEA Software Institute &
Universidad Politécnica de Madrid

Seen So Far

- Sequential case:
  - Termination (VAR).
  - (Partial) correctness.
  - (Total) correctness.
- Write code and verify
  - Code previously written
  - Check it works as intended
- **Many** approaches and tools
  - Floyd-Hoare logic / Dafny / . . .

Pitfalls

- Large programs
  - Large verification problem — may be intractable!
  - If issue discovered: hard to find problem source

Pitfalls

- Large programs
  - Large verification problem — may be intractable!
  - If issue discovered: hard to find problem source
- Concurrency
  - E.g., several events may be true simultaneously
  - Non-determinism

Pitfalls

- Large programs
  - Large verification problem — may be intractable!
  - If issue discovered: hard to find problem source
- Concurrency
  - E.g., several events may be true simultaneously
  - Non-determinism
- Non-terminating programs
  - Need to model environment behavior
  - No notion of final state / correctness
    - All correctness in invariants
  - Need to ensure no deadlock

Some Answers

- Large programs
  - Stepwise refinement
  - Model some requirements, prove relevant properties
  - Add requirements, ensure previous properties untouched
  - Additional properties need to be proved

Some Answers

- Large programs
  - Stepwise refinement
  - Model some requirements, prove relevant properties
  - Add requirements, ensure previous properties untouched
  - Additional properties need to be proved
- Concurrent programs
  - Non-determinism
    - Nothing to do; event model already covers that

Some Answers

- Large programs
  - Stepwise refinement
  - Model some requirements, prove relevant properties
  - Add requirements, ensure previous properties untouched
  - Additional properties need to be proved
- Concurrent programs
  - Non-determinism
    - Nothing to do; event model already covers that
- Non-terminating programs
  - Deadlock-freedom proofs
  - Correctness: completely invariant-based

Some Answers

- Large programs
  - Stepwise refinement
  - Model some requirements, prove relevant properties
  - Add requirements, ensure previous properties untouched
  - Additional properties need to be proved
- Concurrent programs
  - Non-determinism
    - Nothing to do; event model already covers that
- Non-terminating programs
  - Deadlock-freedom proofs
  - Correctness: completely invariant-based
- We will see:
  - Refinement for a sequential case
  - Refinement + concurrency + reactiveness (rest of Event-B part)
  - Tools.

Rodin Info

Install Rodin

Rodin: tool to write Event B models and, using semi-automatic theorem provers, discharge proof obligations.

- Instructions at `http://www.event-b.org/install.html`
- Read also `http://babel.ls.fi.upm.es/teaching/rsd/#rodin-tool`
- Download latest version from `https://sourceforge.net/projects/rodin-b-sharp/files/Core_Rodin_Platform/`
- Start it. Go to `Help` → `Install New Software` → choose `Atelier B Provers` from the `Work with` drop-down menu → select it → click `Next`. Follow instructions.[1]
- If it does not start: likely, issues with environment / java version.
- For example, for Ubuntu Linux 18.10 (and perhaps later):
  - Add `--add-modules=ALL-SYSTEM` to the end of `rodin.ini` in the installation.
  - Install (Oracle) Java v. 1.8 (required by line `-Dosgi.requiredJavaVersion=1.8` in `rodin.ini`).

[1] Collection of automatic theorem provers needed to automate some proofs.

Use Rodin

Using the theorem prover will be your next task.

Exercise (do it before trying anything else)!

1. Write and prove the "integer division" example.
2. Redo the examples in this set of slides.

Again, go to

`babel.ls.fi.upm.es/teaching/rsd/index.html#rodin-tool`

for pointers and information on:

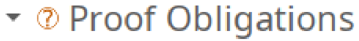
- The RODIN Handbook.
- How to set up a Rodin project (which we will see in short).
- Using the theorem provers inside Rodin.

Please read the relevant sections of the reference manual.
They are linked from the web page.

Practical Matters

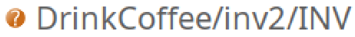
- Editing models: right-click on keywords for drop-down menu with elements to add.
- Most useful keybindings:
  - `Alt-G`: add *children* element e.g., if in THEN, add *action*.
  - `Alt-T`: add *sibling* element e.g., if finished with one guard, add another guard.
  - `Ctrl-S`: save model, run auto-provers, update proof status.
- Help $\rightarrow$ Show Active Keybindings.

- “Explorer window”: expand project, machine / context, “Proof Obligations” to check undischarged proofs:



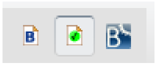
Proof Obligations

- Double click on undischarged proof:



DrinkCoffee/inv2/INV

- Switch to “Prover view” in icons under menu.



- Switch back to “Event-B view” to continue editing.

Proving Hints

www3.hhu.de/stups/handbook/rodin/current/html/proving_perspective.html

- Goal to prove is in "Goal" tab.
- Reasonable strategy:
  - "Lasso" operation: select hypothesis.
  - "P0" p0 : prove with selected hypothesis.
- Otherwise, maybe "PP" pp (*use all hypothesis*) works. If not:
  - Hit "Search Hypothesis" in proof control tab.
  - Select all hypothesis in "Search hypothesis" tab, add to current set (with the black '+' on green).
  - Remove those not related to goal.
  - Try with "P0".
- Quantifiers ($\forall$, $\exists$) may be difficult:
  - Instantiate variables to values useful for proof.
  - Implications: if necessary, reverse implication (right-click on '$\Rightarrow$' and select) so hypothesis appear in antecedent.

Proving Hints (Cont.)

- Innocent changes (e.g., renaming) may make discharged proofs disappear:
  - Rodin can reuse previous proofs.
  - Right click on project, “Proof replay”.
- Backtrack to previous state: right-button select node in “Proof Tree” tab, select “Prune”.
- Right-click on any red symbol to rewrite the expression.
  - Sometimes it is simplified and makes proving easier. E.g., if $r \in p..q$ and $r, p, q \in \mathbb{N}$, it simplifies to $r \leq q$.
- Click on ct to negate goal, move to opposite side of sequent, use proof by contradiction.

$$\frac{P, \neg Q \vdash \bot}{P \vdash Q}$$

Exporting and Importing Projects

(Needed to, for example, turn in the project)

Export (`https://www3.hhu.de/stups/handbook/rodin/current/html/sect0032.html`):

- Select project, right-click and `Export`.
- Select `General` ⟩ `Archive File`, click `Next`.
- Choose path and file name; `Finish`.

Import (`https://www3.hhu.de/stups/handbook/rodin/current/html/sect0033.html`):

- `File` ⟩ `Import` from menu.
- Select `General` ⟩ `Existing Projects into Workspace`, `Next`.
- Choose `Select archive file`, then `Browse`.
- Find zip file and `Finish`.
- Note: importing will fail if project names clash. Either:
  - Right-click and `Rename` before exporting (to save under a different name), or
  - Before importing (to change name of *target* project).

Search in a Sorted Array

Search in array – problem specification

Preconditions

- A natural number $n \in \mathbb{N}$, strictly positive: $0 < n$.
- A sorted array f of n elements built on a set $\mathbb{N}$: $f \in 1..n \to \mathbb{N}$.
- A value v in the array: $\exists i \cdot v = f(i)$.

Postconditions

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

Q: Axioms and invariants (1)

AXIOMS

axm1: $n \in \mathbb{N}_1$

axm3: $f \in 1\,..\,n \to \mathbb{N}$

axm4: $\forall i \cdot \forall j \cdot (i \in 1\,..\,n \land j \in 1\,..\,n \land i \leq j) \Rightarrow f(i) \leq f(j)$

axm5: $v \in ran(f)$

END

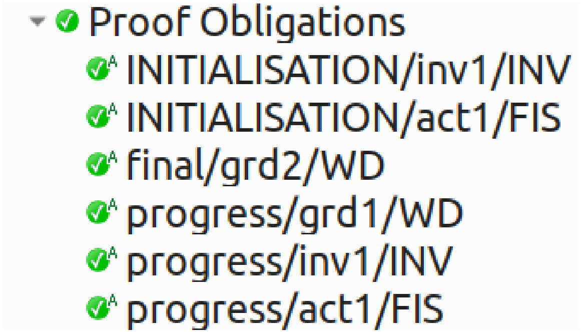
First Machine

MACHINE BS_M0
SEES BS_C0
VARIABLES
  r
INVARIANTS
  `inv1`: $r \in dom(f)$
EVENTS
Initialisation
  begin
    `act1`: $r :\in 1 \,..\, n$

Event final ⟨ordinary⟩ $\widehat{=}$
  when
    `grd2`: $f(r) = v$
  then
    skip
  end
Event progress ⟨anticipated⟩ $\widehat{=}$
  when
    `grd1`: $f(r) \neq v$
  then
    `act1`: $r :\in dom(f)$
  end

Anticipated event: must not increase variant (there is no variant now, but this is necessary to refine it later and add a variant).

(Automatically Proven) Proof Obligations


Proof Obligations
INITIALISATION/inv1/INV
INITIALISATION/act1/FIS
final/grd2/WD
progress/grd1/WD
progress/inv1/INV
progress/act1/FIS

WD (Well-Definedness)

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

$f(E)$	f is a partial function and $E \in dom(f)$
E/F	$F \neq 0$
$E \mod F$	$F \neq 0$
$card(S)$	$finite(S)$
$\min(S)$	$S \subseteq \mathbb{Z} \land \exists x \cdot x \in \mathbb{Z} \land (\forall n \cdot n \in S \Rightarrow x \leq n)$

- In our example: $v \neq f(r)$ needs $r \in dom(f)$, which is ensured by the invariant, which is assumed true (and independently discharged).

FIS

- Ensuring that each non-deterministic action is feasible.
- For an event "evt" and a non-deterministic action "act" in it, the name of this PO is **evt/act/FIS**

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 our case:

- In $r :\in \mathrm{dom}(f)$, is $\mathrm{dom}(f)$ non-empty?
- $f \in 1..n \longrightarrow \mathbb{N}$; since $n \in \mathbb{N}1$, $\mathrm{dom}(f) \neq \varnothing$.

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.

Task: make a round cake with a chocolate layer and creme on top.

1. Make a cake (cylinder shape) with anything.
2. Make a cake like in (1) with a chocolate layer.
3. Make a cake like in (2) with creme on top.

If we produce a square cake in (3), we will break requirement (1) and we will produce a cake which is not a refinement of (2).

Refining Search

- r "shoots" indiscriminately.
- Additional requirement: the range of r must get narrower 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 $\forall i, j \cdot (i \in 1..n \land j \in 1..n \land i \leq j) \Rightarrow f(i) \leq f(j)$)

First Refinement

```
MACHINE BS_M1
REFINES BS_M0
SEES BS_C0
VARIABLES
        r
        p
        q
INVARIANTS
        inv1:  $p \in 1 \,..\, n$
        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
```

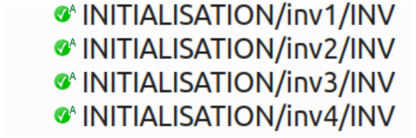
convergent: VARIANT must decrease.

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

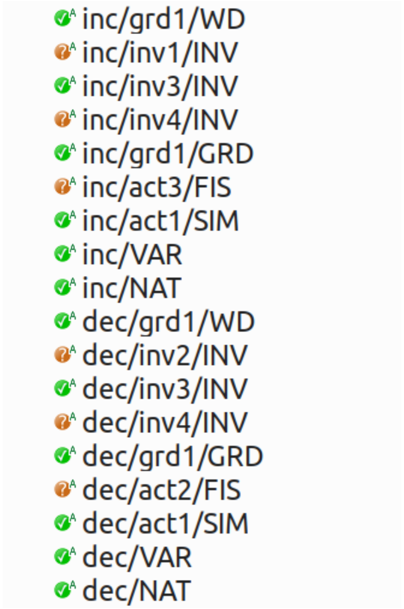
Q: Why does this model eventually find r? (2)

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



- ✓ 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 do not lead to states[2] that were not reachable in the abstract model (GRD).
2. Every concrete event is simulated by an abstract event (SIM).

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

[2]Being imprecise here: the relationship is between concrete states and their simulations.

The Essence of GRD

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

Abstract model

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

Concrete model: more details / more variables / richer state

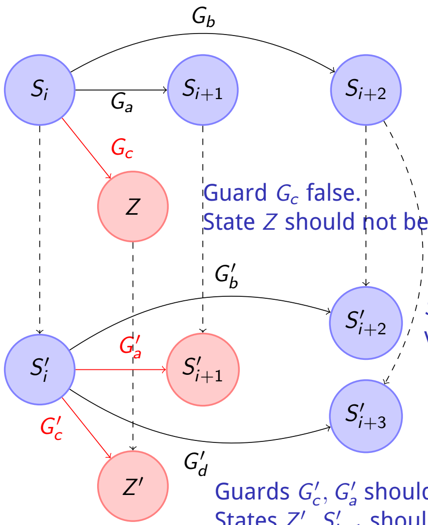
- Concrete and abstract states differ.
- A correspondence (“simulation”) must exist.
- Additional constraints may make some abstract states invalid in the concrete model: they must not be reachable (they disappear).
- Some abstract states *split* into several concrete states.

The Essence of GRD (Cont)

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

Abstract model

Concrete model (*primed* elements are concrete versions of abstract counterparts)



Guard G_c false.
State Z should not be reached.

S'_{i+2}, S'_{i+3} richer versions of S_{i+2}

Guards G'_c, G'_a should be false.
States Z', S'_{i+1} should not be reached.

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 $\bot \Rightarrow \bot$ 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 $\bot \Rightarrow \top$ is valid).

However, if G'_c were true in the concrete model, then $G'_c \Rightarrow G_c$ would be false, because $\top \Rightarrow \bot$ 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 abstract ones.
- Ensures that when concrete event enabled then so is the corresponding abstract one.
- 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⟩ ≙
    when
        grd1:  $f(r) \neq v$
    then
        act1:  $r :\in dom(f)$
    end
```

```
Event inc ⟨convergent⟩ ≙
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 event *simulate* corresponding abstract action
- 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

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

- Let's find a proof (by contradiction): when could it be that $r' \notin \mathrm{dom}(f)$.

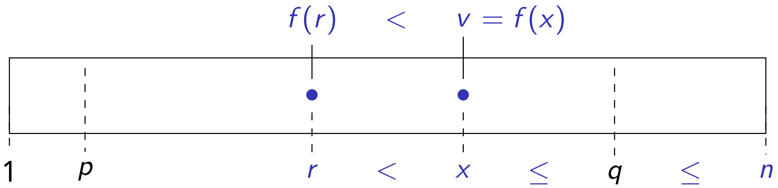
Rodin and the Second Refinement

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

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

inc/inv1/INV	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

$$r \in dom(f)$$

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

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

$$f(r) < v$$

$$v \in ran(f)$$

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

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

Left: selected hypothesis and goal.

Right: rewritings of the LHS of the sequent.

Sketch of a Proof for inc/inv1/INV

$$r \in dom(f)$$

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

$$f(r) < v$$

$$v \in ran(f)$$

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

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

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

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

Left: selected hypothesis and goal.

Right: rewritings of the LHS of the sequent.

Sketch of a Proof for inc/inv1/INV

$$r \in dom(f)$$

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

$$f(r) < v$$

$$v \in ran(f)$$

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

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

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

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

$$x \mapsto v \in f$$

Left: selected hypothesis and goal.

Right: rewritings of the LHS of the sequent.

Sketch of a Proof for inc/inv1/INV

$$r \in dom(f)$$

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

$$f(r) < v$$

$$v \in ran(f)$$

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

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

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

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

$$x \mapsto v \in f$$

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

Left: selected hypothesis and goal.

Right: rewritings of the LHS of the sequent.

Sketch of a Proof for inc/inv1/INV

$$r \in dom(f)$$

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

$$f(r) < v$$

$$v \in ran(f)$$

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

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

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

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

$$x \mapsto v \in f$$

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

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

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

$$r \in dom(f)$$

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

$$f(r) < v$$

$$v \in ran(f)$$

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

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

Left: selected hypothesis and goal.

Right: rewritings of the LHS of the sequent.

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

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

$$x \mapsto v \in f$$

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

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

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

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

$$x > r$$

$$x \leq n$$

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

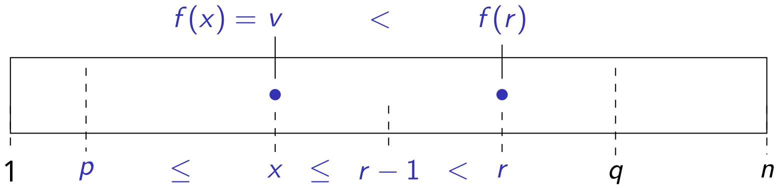
- Double click on undischarged proof, switch to proving perspective.
- Show all hypothesis (click on search button).
- Select the hypothesis in the previous slide.
- Click on the **+** button in the tab of the 'Search hypotheses' window. They should now appear under 'Selected hypotheses'.
- Invert implication 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`.

dec/act2/FIS

FIS In act2 $r :\in p..r-1$; need to ensure that $p \le r-1$.



$v \in f[p..q]$ from which $f(x) = v$ and $x \in p..q$, from which $x \ge p$. Since $v < f(r)$ and $f(x) = v$, $f(x) < f(r)$ and $x < r$; therefore $x \le r-1$. We have also $r \in p..q$, and then $r \le p$. Then $p \le x \le r-1$.

Reviewed Hypothesis

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

Additional Info: THM

- Sometimes reusing formulas deducible from axioms is handy.
- E.g., in our examples we used

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

  derived from axm2. This is a **theorem** in Rodin / Event-B.
- Theorems useful to simplify proofs.
- For a theorem “thm”, the name of its PO is **thm/THM**.
- Proved as usual.

Axioms $\vdash$ Theorem	thm/THM

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

Type Checking and Mathematical Proofs

Types

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

$$\begin{aligned} &f(x : \mathbb{R}) : \mathbb{R} \\ &\quad \text{return } x * 3.5 \\ &g(x : \mathbb{R}) : \mathbb{N} \\ &\quad \text{return } x * 3.5 \end{aligned}$$

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

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

$$
\begin{array}{ll}
f(x: & \mathbb{R}) : \mathbb{R} \\
 & \text{return } x * 3.5 \\
g(x: & \mathbb{R}) : \mathbb{N} \\
 & \text{return } x * 3.5
\end{array}
$$

Theorems

- Determine formula valid.
- Hypotheses + deduction rules.

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

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

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

Type Checking and Mathematical Proofs

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

    length(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.