Event-B: Introduction and First Steps[1]

Manuel Carro
manuel.carro@upm.es

IMDEA Software Institute &
Universidad Politécnica de Madrid

[1] Many slides borrowed from J. R. Abrial

Mundane matters 3
Landscape 6
Computation model 10
Integer division example 15
Invariants 19
Sequents and proofs 24
Inference rules 37
Invariant proofs 40
Formula strength 45
Termination and correctness 47

- Jean-Raymond Abrial.
  Faultless systems: Yes we can!
  IEEE Computer, 42(9):30–36, 2009.
- Jean-Raymond Abrial.
  Modeling in Event-B - System and Software Engineering.
  Cambridge University Press, 2010.
- Mordechai Ben-Ari.
  Mathematical Logic for Computer Science, 3rd Edition.
  Springer, 2012.
- Michael Huth and Mark Ryan.
  Logic in Computer Science: Modelling and Reasoning About Systems.
  Cambridge University Press, New York, NY, USA, 2004.
- Lawrence C. Paulson.
  Logic and Proof.
  Lecture notes, U. of Cambridge.

Take notes

Picture & headline ©*The Atlantic*
https://www.theatlantic.com/technology/archive/2014/05/to-remember-a-lecture-better-take-notes-by-hand/361478/

I will make notes / slides available *after* the lectures
I will ask you to work during the lectures

2020-2021 specific information

Sep. 23 – Oct. 28 Event B.

Nov. 4 – Dec. 9 Floyd-Hoare logic and Dafny; executable specifications (Maude, Prolog).

Dec. 16 Term project presentation.

Jan. 19 (2021) Final exam (if needed).

Note: in case of written exams, they must be f2f.

Lecture plan

- 2 × 50 min. sessions.

Project plan

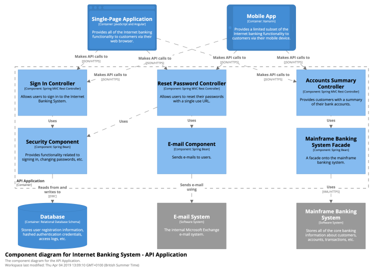
Project to be developed individually or in groups (depending on # students) to be presented to everyone.

Event B

An industry-oriented method, language, and set of supporting tools to describe systems of interacting, reactive software, hardware components, and their environment, and to reason about them.

Industrial systems: usual characteristics

- Functionality often not too complex.
  - Algorithms / data structures relatively simple.
  - Underlying maths of reasonable complexity.
- Requirements document usually poor.
- Reactive and concurrent by nature.
  - But often coarse: protecting (large) critical regions often enough.



- Many special cases.
- Communication with hardware / environment involved.
- Many details (≈ properties to ensure) to be taken into account.
- Large (in terms of LOCs).

Producing correct (software) systems hard — but not necessarily from a theoretical point of view.

The Event B approach

Complexity: Model Refinement

- System built incrementally, monotonically.
  - Take into account subset of requirements at each step.
  - Build model of a *partial* system.
  - Prove its correctness.
- **Add** requirements to the model, ensure correctness:
  - The requirements correctly captured by the new model.
  - New model preserves properties of previous model.

Details: Tool Support

- Tool to edit Event B models (Rodin).
- Generates *proof obligations*: theorems to be proved to ensure correctness.
- Interfaced with (interactive) theorem provers.
- Extensible.

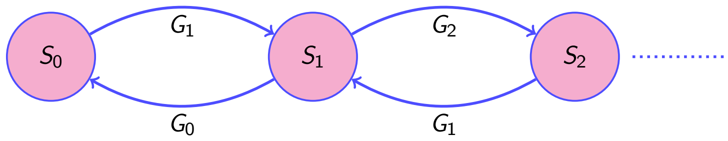
Basic ideas

Model: formal description of a discrete system.

- Formal: mechanism to decide whether some properties hold
- Discrete: can be represented as a transition system
- Formalization contains models of:
  - The future software components
  - The future equipments surrounding these components
- The overall model construction can be very complex.
- Refinement and decomposition key to master this complexity:
  - Build model gradually
  - Ordered sequence of more precise models
  - Each model refines its predecessor.

Models and states

A discrete model is made of states



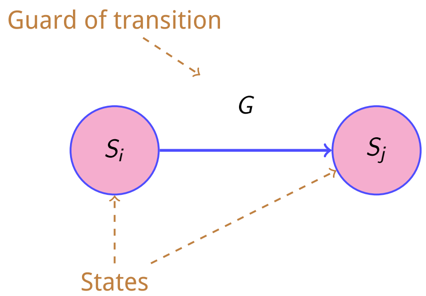
- States are represented by constants and variables

$$S_i = \langle c_1, \ldots, c_n, v_1, \ldots, v_m \rangle$$

- Relationships among constants and variables written using set-theoretic expressions

States and transitions

- Transitions between states are triggered by events
- An event is made of a guard and an action
  - The guard G denotes the enabling condition of the event
  - The action denotes the way the state is modified by the event
- Guards and actions are written using set-theoretic expressions (e.g., first-order, classical logic).



A simple example

Search for element `k` in array `f` of length `n`, assuming `k` is in `f`.

Constants / Axioms

```
CONST n ∈ ℕ
CONST k ∈ ℕ
CONST f ∈ 1..n ⟶ ℕ
```

Variables / Invariants

```
i ∈ 1..n
```

```
Event Search
  when
    i < n ∧ f(i) ≠ k
  then
    i := i + 1
  end
```

```
Event Found
  when
    f(i) = k
  then
    skip
  end
```

(initialization of `i` not shown for brevity)

Events

```
Event EventName
  when
    guard:  G(v, c)
  then
    action:  v := E(v, c)
  end
```

G(v, c): a predicate that must be true for EventName to be enabled

Informal operational interpretation

```
Initialize;
while (some events have true guards) {
   Choose one such event;
   Modify the state accordingly;
}
```

```
Event EventName
  when
    guard:  G(v, c)
  then
    action:  v := E(v, c)
  end
```

- An event execution takes no time.
  - No two events occur simultaneously.
- When all events have false guards, the discrete system stops.
- When some events have true guards, one of them is non-deterministically chosen and its action modifies the state.
- Previous phase is repeated if possible.
- Halting not necessary: a discrete system may run forever.

Running example (sequential code)

$$a = \left\lfloor \frac{b}{c} \right\rfloor$$

- We want to define division for natural numbers (without using division).

Q: division specification (1)

We assume:

$$b, c \in \mathbb{N} \wedge c > 0$$

We want:

$$b = a \times c + k \wedge a \in \mathbb{N} \wedge k \in \mathbb{N} \wedge k < c$$

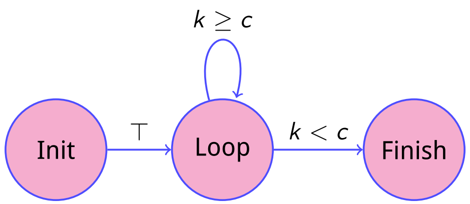
- Input and output?
- Variables and constants?
- Types?

Programming integer division

- We have addition and substraction
- We have a simple procedural language: variables, assignment, while loops, if-then-else, + & -, arithmetical operators, arithmetical comparison
- Think for a couple of minutes and let us write code

Q: integer division code (2)

```
a := 0
k := b
while k >= c
   k := k - c
   a := a + 1
```



Note

This step is not taken in Event B. We are writing this code only for illustration purposes.

Please copy the code and save it for later.

Towards events

```
Event EventName
  when
    G(v, c)
  then
    v := E(v, c)
  end
```

- Special initialization event (**INIT**).
- Sequential program (special case):
  - *Finish* event, *Progress* events
  - *Finish* event guard negate *Progress'*
  - Some guard is always true
  - Think for 5 min. Share the result.

Q: integer division events (3)

```
Event INIT            Event Progress                Event Finish
  a, k = 0, b           when                          when
end                       k >= c                        k < c
                        then                          then
                          k, a := k - c, a + 1          skip
                        end                           end
```

Categorizing elements

Constants	Axioms
Q: constants (4) b c	Q: axioms (5) $b \in \mathbb{N}$ $c \in \mathbb{N}$ $c > 0$
Variables	**Invariants**
Q: variables (6) a k	Later!

Invariants

- Invariant: formula true before and after event
- State *safety* conditions, prove correctness
  - What must always be true in a physical system
  - What must always be true in an algorithm
  - Necessary to prove (sequential) correctness
  - In non-terminating, reactive systems:
    capture conditions which must always hold (safety)
- Finding invariants: mixes art and science
- Hint: explore what happens with the variables as the code proceeds

Finding invariants

Constants and **variables**?

Which assertions are invariant in our model?

Q: model invariants (7)

I_1: $a \in \mathbb{N}$ *// Type invariant*
I_2: $k \in \mathbb{N}$ *// Type invariant*
I_3: $b = a \times c + k$

One formula which is an invariant for **any** Event-B model / loop.

Q: eternal invariant (8)

$\top$

Invariant preservation

- Proving invariant preservation: For all event i, invariant j

$$A(c), G_i(v,c), I_{1\ldots n}(v,c) \vdash I_j(E_i(v,c),c)$$

  - $A(c)$ axioms
  - $G_i(v,c)$ guard of event i
  - $I_j(v,c)$ invariant j
  - $I_{1\ldots n}(v,c)$ all the invariants
  - $E_i(v,c)$ result of action i

Sequent

$$\Gamma \vdash \Delta$$

Show that:
with assumptions Γ, I can prove Δ

Invariant preservation

If an invariant holds and the guards of an event are true and we execute the event's action, the invariant still holds.

- **INIT** case

invariant preservation for **INIT** (9)

$$A(c) \vdash I_j(E_{\text{init}}(v,c),c)$$

Invariant preservation proofs

- Invariant: mathematical assertion
- Mathematically proven from code and math axioms
- Three invariants & three events: nine proofs
- Named as e.g. $E_{\text{Progress}}/I_2/\text{INV}$
  - Other types of proofs will be necessary in due time

$E_{\text{INIT}} / I_1 / \text{INV}$

INIT I1 invariant proof (10)

$$\dfrac{\dfrac{}{\vdash 0 \in \mathbb{N}}\ \text{PO}}{b \in \mathbb{N}, c \in \mathbb{N}, c > 0 \vdash 0 \in \mathbb{N}}\ \text{MON}$$

$E_{\text{INIT}} / I_2 / \text{INV}$

INIT I2 invariant proof (11)

$$\dfrac{\dfrac{}{b \in \mathbb{N} \vdash b \in \mathbb{N}}\ \text{HYP}}{b \in \mathbb{N}, c \in \mathbb{N}, c > 0 \vdash b \in \mathbb{N}}\ \text{MON}$$

Invariant preservation proofs

$E_{\text{INIT}} / I_3 / \text{INV}$

INIT I3 invariant proof (12)

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash b = b}\ \text{EQL}}{\vdash b = 0 + b}\ \text{Arith}}{\vdash b = 0 \times c + b}\ \text{Arith}}{b \in \mathbb{N}, c \in \mathbb{N}, c > 0 \vdash b = 0 \times c + b}\ \text{MON}$$

Sequents

- Mechanize proofs
  - Humans "understand"; proving is tiresome and error-prone
  - Computers manipulate symbols
- How can we mechanically construct correct proofs?
  - Every step crystal clear
  - For a computer to perform
- Several approaches
- For Event B: sequent calculus
  - To read: [Pau] (available at course web page), at least Sect. 3.3 to 3.5 , 6.4, and 6.5. Note: when we use $\Gamma \vdash \Delta$, Paulson uses $\Gamma \Rightarrow \Delta$

Inference rule

- An inference rule is a tool to perform a formal proof.
- It is denoted by:

$$\frac{A}{C}\ R$$

- A is a (possibly empty) collection of sequents: the antecedents.
- C is a sequent: the consequent.
- R is the name of the rule.

The proofs of each sequent of A
——— together give you ———
a proof of sequent C

An example of inference rule

Note: not exactly the inference rules we will use. Only an intuitive example.

- A(lice) and B(ob) are siblings:

$$\frac{\text{C is mother of A} \qquad \text{C is mother of B}}{\text{A and B are siblings}}\ \text{Sibling-M}$$

$$\frac{\text{C is father of A} \qquad \text{C is father of B}}{\text{A and B are siblings}}\ \text{Sibling-F}$$

- Note: other possibilities may exist.

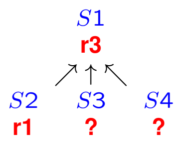
$$\frac{}{S2}\mathbf{r1} \quad \frac{S7}{S4}\mathbf{r2} \quad \frac{S2\ \ S3\ \ S4}{S1}\mathbf{r3} \quad \frac{}{S5}\mathbf{r4} \quad \frac{S5\ \ S6}{S3}\mathbf{r5} \quad \frac{}{S6}\mathbf{r6} \quad \frac{}{S7}\mathbf{r7}$$

$S1$
?

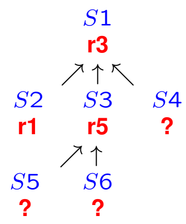
$$\frac{}{S2}\mathbf{r1} \quad \frac{S7}{S4}\mathbf{r2} \quad \frac{S2\ \ S3\ \ S4}{S1}\mathbf{r3} \quad \frac{}{S5}\mathbf{r4} \quad \frac{S5\ \ S6}{S3}\mathbf{r5} \quad \frac{}{S6}\mathbf{r6} \quad \frac{}{S7}\mathbf{r7}$$

$S1$
r3
$\nearrow \uparrow \nwarrow$
$S2$ $S3$ $S4$
? **?** **?**

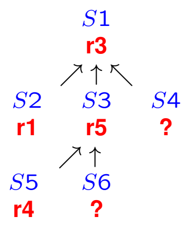
$\frac{}{S2}$**r1** $\frac{S7}{S4}$**r2** $\frac{S2 \quad S3 \quad S4}{S1}$**r3** $\frac{}{S5}$**r4** $\frac{S5 \quad S6}{S3}$**r5** $\frac{}{S6}$**r6** $\frac{}{S7}$**r7**



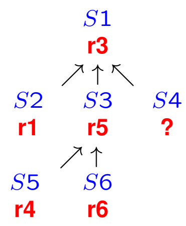
$\frac{}{S2}$**r1** $\frac{S7}{S4}$**r2** $\frac{S2 \quad S3 \quad S4}{S1}$**r3** $\frac{}{S5}$**r4** $\frac{S5 \quad S6}{S3}$**r5** $\frac{}{S6}$**r6** $\frac{}{S7}$**r7**



$\frac{}{S2}$**r1** $\frac{S7}{S4}$**r2** $\frac{S2 \quad S3 \quad S4}{S1}$**r3** $\frac{}{S5}$**r4** $\frac{S5 \quad S6}{S3}$**r5** $\frac{}{S6}$**r6** $\frac{}{S7}$**r7**

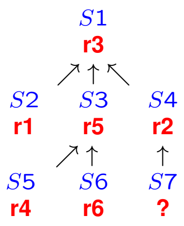


$\frac{}{S2}$**r1** $\frac{S7}{S4}$**r2** $\frac{S2 \quad S3 \quad S4}{S1}$**r3** $\frac{}{S5}$**r4** $\frac{S5 \quad S6}{S3}$**r5** $\frac{}{S6}$**r6** $\frac{}{S7}$**r7**



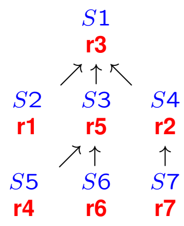
Proof of Sequent $S1$

$\frac{}{S2}$**r1** $\quad \frac{S7}{S4}$**r2** $\quad \frac{S2 \quad S3 \quad S4}{S1}$**r3** $\quad \frac{}{S5}$**r4** $\quad \frac{S5 \quad S6}{S3}$**r5** $\quad \frac{}{S6}$**r6** $\quad \frac{}{S7}$**r7**



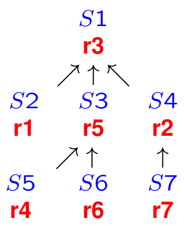
Proof of Sequent $S1$

$\frac{}{S2}$**r1** $\quad \frac{S7}{S4}$**r2** $\quad \frac{S2 \quad S3 \quad S4}{S1}$**r3** $\quad \frac{}{S5}$**r4** $\quad \frac{S5 \quad S6}{S3}$**r5** $\quad \frac{}{S6}$**r6** $\quad \frac{}{S7}$**r7**



Recording the Proof of Sequent $S1$

$\frac{}{S2}$**r1** $\quad \frac{S7}{S4}$**r2** $\quad \frac{S2 \quad S3 \quad S4}{S1}$**r3** $\quad \frac{}{S5}$**r4** $\quad \frac{S5 \quad S6}{S3}$**r5** $\quad \frac{}{S6}$**r6** $\quad \frac{}{S7}$**r7**



- The proof is a tree

Being more precise about sequents

- We supposedly have a predicate language
  - Not formally defined yet
  - We will assume it is first-order, classical logic
  - Recommended references: [Pau, HR04, Ben12]
- A sequent is denoted by the following construct: $H \vdash G$
- H is a (possibly empty) collection of predicates: the hypotheses
- G is a predicate: the goal

Objective

Show that, under the hypotheses of collection H, the goal G can be proven.

Basic inference rules

- There are three basic inference rules
- These rules are independent of the predicate Language

HYPothesis

$$\frac{}{H, P \vdash P}\ \text{HYP}$$

If the goal is among the hypothesis, we are done.

MONotony

$$\frac{H \vdash Q}{H, P \vdash Q}\ \text{MON}$$

If goal proven without hypothesis P, then can be proven with P.

shortCUT

$$\frac{H \vdash P \qquad H, P \vdash Q}{H \vdash Q}\ \text{CUT}$$

A goal can be proven with an intermediate deduction P. Nobody tells us what is P or how to come up with it. (*Cut Elimination Theorem*)

More Rules

- There are many other rules for:
  - Logic itself
    - Look at the slides / documents in the course web page
  - reasoning on arithmetic (Peano axioms),
  - reasoning on sets,
  - reasoning on functions,
  - ...
- We will not list all of them here (see online documentation)
- We will explain them as they appear
- But a mechanical prover has them as "inside knowledge" (plus tactics, strategies)

Previous (unexplained) rules

First Peano axiom

$$\frac{}{\vdash 0 \in \mathbb{N}}\ \text{P0}$$

Second Peano axiom

$$\frac{}{n \in \mathbb{N} \vdash n+1 \in \mathbb{N}}\ \text{P1}$$

Term substitution

$$\frac{Q(E), E = F \vdash R(E)}{Q(E), E = F \vdash R(F)}\ \text{EQ-LR}$$

Equality

$$\frac{}{\vdash E = E}\ \text{EQL}$$

Invariant preservation proofs

$E_{Progress}$ / I_1 / INV

Progress I1 invariant proof (13)

$$\frac{\dfrac{}{a \in \mathbb{N} \vdash a+1 \in \mathbb{N}}\ \text{P1}}{b \in \mathbb{N}, c \in \mathbb{N}, c > 0, k \geq c, k \in \mathbb{N}, b = a \times c + k, a \in \mathbb{N} \vdash a+1 \in \mathbb{N}}\ \text{MON}$$

More rules

$$\frac{H, Q \vdash R \qquad H, P \vdash R}{H, P \vee Q \vdash R}\ \text{OR-L}$$

A disjunction on the LHS needs both branches of the disjunction be discharged separately

$$\frac{H \vdash P}{H \vdash P \vee Q}\ \text{OR-R1} \qquad \frac{H \vdash Q}{H \vdash P \vee Q}\ \text{OR-R2}$$

A disjunction on the RHS only needs ***one*** *of the branches to be proven. That's why there are two rules: one to choose each of the branches.*

$$\frac{H \vdash Q \qquad H \vdash P}{H \vdash P \wedge Q}\ \text{AND-L}$$

A conjunction on the RHS only nees both branches of the conjunction be proven independently of each other.

Invariant preservation proofs

E_{Progress} / I_2 / INV

Progress I2 invariant proof (14)

$$\frac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash 0 \in \mathbb{N}}\ \text{P0}}{\vdash c - c \in \mathbb{N}}\ \text{Arith}}{c \in \mathbb{N}, c \in \mathbb{N} \vdash c - c \in \mathbb{N}}\ \text{MON}}{c \in \mathbb{N}, k = c, k \in \mathbb{N} \vdash k - c \in \mathbb{N}}\ \text{EQ-LR} \qquad \dfrac{\dfrac{\dfrac{}{c \in \mathbb{N}, k - c > 0, k \in \mathbb{N} \vdash k - c \in \mathbb{N}}\ \text{Arith}^*}{c \in \mathbb{N}, k - c > c - c, k \in \mathbb{N} \vdash k - c \in \mathbb{N}}\ \text{Simp-M-Minus}}{c \in \mathbb{N}, k > c, k \in \mathbb{N} \vdash k - c \in \mathbb{N}}\ \text{Arith-M-M-R}}{\dfrac{\dfrac{c \in \mathbb{N}, k > c \vee k = c, k \in \mathbb{N} \vdash k - c \in \mathbb{N}}{c \in \mathbb{N}, k \geq c, k \in \mathbb{N} \vdash k - c \in \mathbb{N}}\ \text{Arith}}{b \in \mathbb{N}, c \in \mathbb{N}, c > 0, k \geq c, a \in \mathbb{N}, b = a \times c + k, k \in \mathbb{N} \vdash k - c \in \mathbb{N}}\ \text{MON}}\ \text{OR-L}$$

Invariant preservation proofs

E_{Progress} / I_3 / INV

Progress I3 invariant proof (15)

$$\frac{\dfrac{\dfrac{\dfrac{\dfrac{}{b = a \times c + k \vdash b = a \times c + k}\ \text{HYP}}{b = a \times c + k \vdash b = a \times c + c + k - c}\ \text{Arith-M-Pl-Dist}}{b = a \times c + k \vdash b = (a+1) \times c + k - c}\ \text{Arith-M-Pl-Dist}}{b = a \times c + k \vdash b = (a+1) \times c + (k - c)}\ \text{Arith-Pl-M}}{b \in \mathbb{N}, c \in \mathbb{N}, c > 0, k \geq c, a \in \mathbb{N}, k \in \mathbb{N}, b = a \times c + k \vdash b = (a+1) \times c + (k - c)}\ \text{MON}$$

Invariant preservation proofs

Proofs for `Finish`

- $E_{\text{Finish}}/I_1/\text{INV}$
- $E_{\text{Finish}}/I_2/\text{INV}$
- $E_{\text{Finish}}/I_3/\text{INV}$

are trivial (`Finish` does not change anything)

Inductive and non-inductive invariants

- We want to prove

$$A(c) \vdash I_j(E_{init}(v,c),c)$$
$$A(c), G_i(v,c), I_{1...n}(v,c) \vdash I_j(E_i(v,c),c)$$

- I_i: *inductive invariant* (base case + inductive case)
- True invariants can be non-inductive if they cannot be proved from program

```
Event INIT          Event Loop
  a:  x := 1          a:  x := 2*x - 1
end                 end
```

- $x \geq 0$ is an invariant.
- It is not inductive (`Loop`: $x \geq 0 \vdash 2 * x - 1 \geq 0$?)
- $x > 0$ is inductive

- Strengthening (if $x \geq 0$ needed, we can use $x > 0$)

Formula strength and information

- If $A \vdash B$ or $A \Rightarrow B$, then A is stronger than B.
- Balance between extremes
  - Too weak: easy as invariant, maybe not enough information
  - You want invariants strong enough to prove the property one wants.
- What are the strongest and weakest possible formulæ?

Q: strongest / weakest formulæ. (16)

$$\bot \equiv Q \wedge \neg Q \qquad \text{(because } \bot \vdash R\text{)}$$
$$\top \equiv Q \vee \neg Q \qquad \text{(because } R \vdash \top\text{)}$$

- The *proof by contradiction* rule:

$$\frac{}{\bot \vdash P}\ \text{ECQ}$$

Sequential correctness

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

$$A(c), G_{Finish}(v,c), I_{1..n}(v,c) \vdash P(v,c)$$

Q: corr. cond. for example (17)

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

- Not applicable to non-terminating systems (other proofs required)
- $I_{1..n}$ and G_{Finish} related to P; not necessarily identical
- Nota that correct $I_{1...n}$ may not be *strong* enough: the stronger, the better.

Termination

- "*Postcondition P must be true **at the end** of execution*"
- General strategy: look for a *ranking function / progress measure*
- 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

$$A(v), I_{1...n}, G_i(v,c) \vdash V(v,c) > V(E_i(v,c),c)$$

Q: variant expression (18)

```
variant:  k
```

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

Termination proof (19)

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