Modelling Software-based Systems

Tutorial

Telecom Nancydocu

Dominique Méry
Telecom Nancy,Université de Lorraine

2 mai 2025(10:18 A.M.)

dominique.mery@loria.fr

General Summary

1. Invariant versus theorem in Event-B
2. Designing a C program for $\lambda x.x \times x$
3. The power function $\lambda x, y.x^y$

Current Summary

1. Invariant versus theorem in Event-B
2. Designing a C program for $\lambda x.x \times x$
3. The power function $\lambda x, y.x^y$

$x \leq 0$ is always true !

```
VARIABLES x
 INVARIANTS
  @inv1 x ∈ ℤ
  @theproperty x ≤ 0
   EVENT INITIALISATION
       then
     @act1 x := −1
   end
   EVENT event1
       where
     @grd1 x ≥ 0
       then
     @act1 x := x + 1
   end
   EVENT event2
       then
     @act1 x := x
   end
```

- $x \leq 0$ is always true.
- $x \leq 0 \wedge BA(event1)(x, x') \Rightarrow x' \leq 0$ is not true !

$x \leq 0$ is always true !

```
VARIABLES   x
 INVARIANTS
  @inv1 x  ∈  ℤ
  @theproperty x ≤ 0
   EVENT   INITIALISATION
        then
     @act1 x   :=   − 1
   end
   EVENT   event1
        where
     @grd1 x  ≥ 0
        then
     @act1 x  :=  x + 1
   end
   EVENT   event2
        then
     @act1 x  :=  x
   end
```

- $x \leq 0$ is always true.
- $x \leq 0 \wedge BA(event1)(x, x') \Rightarrow x' \leq 0$ is not true !
  - $x \leq 0 \wedge BA(event1)(x, x') \Rightarrow x' \leq 0$

$x \leq 0$ is always true !

```
VARIABLES x
INVARIANTS
 @inv1 x ∈ ℤ
 @theproperty x ≤ 0
 EVENT INITIALISATION
   then
    @act1 x := −1
 end
 EVENT event1
   where
    @grd1 x ≥ 0
   then
    @act1 x := x + 1
 end
 EVENT event2
   then
    @act1 x := x
 end
```

- $x \leq 0$ is always true.
- $x \leq 0 \wedge BA(event1)(x, x') \Rightarrow x' \leq 0$ is not true !
  - $x \leq 0 \wedge BA(event1)(x, x') \Rightarrow x' \leq 0$
  - $\vdash x \leq 0 \wedge BA(event1)(x, x') \Rightarrow x' \leq 0$

$x \leq 0$ is always true !

```
VARIABLES  x
 INVARIANTS
  @inv1 x ∈ ℤ
  @theproperty x ≤ 0
  EVENT  INITIALISATION
      then
    @act1 x := −1
  end
  EVENT  event1
      where
    @grd1 x ≥ 0
      then
    @act1 x := x + 1
  end
  EVENT  event2
      then
    @act1 x := x
  end
```

- $x \leq 0$ is always true.
- $x \leq 0 \land BA(event1)(x, x') \Rightarrow x' \leq 0$ is not true !
  - $x \leq 0 \land BA(event1)(x, x') \Rightarrow x' \leq 0$
  - $\vdash x \leq 0 \land BA(event1)(x, x') \Rightarrow x' \leq 0$
  - $x \leq 0 \land x \geq 0 \land x' = x + 1 \vdash x' \leq 0$

$x \leq 0$ is always true !

```
VARIABLES  x
 INVARIANTS
  @inv1 x ∈ ℤ
  @theproperty x ≤ 0
  EVENT  INITIALISATION
      then
    @act1 x := −1
  end
  EVENT  event1
      where
    @grd1 x ≥ 0
      then
    @act1 x := x + 1
  end
  EVENT  event2
      then
    @act1 x := x
  end
```

- $x \leq 0$ is always true.
- $x \leq 0 \wedge BA(event1)(x, x') \Rightarrow x' \leq 0$ is not true !
  - $x \leq 0 \wedge BA(event1)(x, x') \Rightarrow x' \leq 0$
  - $\vdash x \leq 0 \wedge BA(event1)(x, x') \Rightarrow x' \leq 0$
  - $x \leq 0 \wedge x \geq 0 \wedge x' = x + 1 \vdash x' \leq 0$
  - $x \leq 0 \wedge x \geq 0 \wedge x' = x + 1 \Rightarrow x' \leq 0$

$x \leq 0$ is always true !

```
VARIABLES x
 INVARIANTS
  @inv1 x ∈ ℤ
  @theproperty x ≤ 0
   EVENT INITIALISATION
        then
     @act1 x := −1
   end
   EVENT event1
        where
     @grd1 x ≥ 0
        then
     @act1 x := x + 1
   end
   EVENT event2
        then
     @act1 x := x
   end
```

- $x \leq 0$ is always true.
- $x \leq 0 \wedge BA(event1)(x,x') \Rightarrow x' \leq 0$ is not true !
  - $x \leq 0 \wedge BA(event1)(x,x') \Rightarrow x' \leq 0$
  - $\vdash x \leq 0 \wedge BA(event1)(x,x') \Rightarrow x' \leq 0$
  - $x \leq 0 \wedge x \geq 0 \wedge x' = x+1 \vdash x' \leq 0$
  - $x \leq 0 \wedge x \geq 0 \wedge x' = x+1 \Rightarrow x' \leq 0$
  - $x \leq 0, x \geq 0, x' = x+1 \vdash x' \leq 0$

$x \leq 0$ is always true !

```
VARIABLES x
INVARIANTS
 @inv1 x ∈ ℤ
 @theproperty x ≤ 0
 EVENT INITIALISATION
    then
   @act1 x := −1
 end
 EVENT event1
    where
   @grd1 x ≥ 0
    then
   @act1 x := x + 1
 end
 EVENT event2
    then
   @act1 x := x
 end
```

- $x \leq 0$ is always true.
- $x \leq 0 \wedge BA(event1)(x,x') \Rightarrow x' \leq 0$ is not true !
  - $x \leq 0 \wedge BA(event1)(x,x') \Rightarrow x' \leq 0$
  - $\vdash x \leq 0 \wedge BA(event1)(x,x') \Rightarrow x' \leq 0$
  - $x \leq 0 \wedge x \geq 0 \wedge x' = x+1 \vdash x' \leq 0$
  - $x \leq 0 \wedge x \geq 0 \wedge x' = x+1 \Rightarrow x' \leq 0$
  - $x \leq 0, x \geq 0, x' = x+1 \vdash x' \leq 0$
  - $x \leq 0, x \geq 0, x' = x+1 \vdash x+1 \leq 0$

$x \leq 0$ is always true !

```
VARIABLES x
INVARIANTS
 @inv1 x ∈ ℤ
 @theproperty x ≤ 0
 EVENT INITIALISATION
   then
    @act1 x := −1
 end
 EVENT event1
   where
    @grd1 x ≥ 0
   then
    @act1 x := x + 1
 end
 EVENT event2
   then
    @act1 x := x
 end
```

- $x \leq 0$ is always true.
- $x \leq 0 \wedge BA(event1)(x, x') \Rightarrow x' \leq 0$ is not true !
  - $x \leq 0 \wedge BA(event1)(x, x') \Rightarrow x' \leq 0$
  - $\vdash x \leq 0 \wedge BA(event1)(x, x') \Rightarrow x' \leq 0$
  - $x \leq 0 \wedge x \geq 0 \wedge x' = x + 1 \vdash x' \leq 0$
  - $x \leq 0 \wedge x \geq 0 \wedge x' = x + 1 \Rightarrow x' \leq 0$
  - $x \leq 0, x \geq 0, x' = x + 1 \vdash x' \leq 0$
  - $x \leq 0, x \geq 0, x' = x + 1 \vdash x + 1 \leq 0$
  - $x \leq 0, x \geq 0, x = 0, x' = x + 1 \vdash x + 1 \leq 0$

$x \leq 0$ is always true !

```
VARIABLES x
 INVARIANTS
  @inv1 x ∈ ℤ
  @theproperty x ≤ 0
  EVENT INITIALISATION
     then
    @act1 x := −1
  end
  EVENT event1
     where
    @grd1 x ≥ 0
     then
    @act1 x := x + 1
  end
  EVENT event2
     then
    @act1 x := x
  end
```

- $x \leq 0$ is always true.
- $x \leq 0 \wedge BA(event1)(x,x') \Rightarrow x' \leq 0$ is not true !
  - $x \leq 0 \wedge BA(event1)(x,x') \Rightarrow x' \leq 0$
  - $\vdash x \leq 0 \wedge BA(event1)(x,x') \Rightarrow x' \leq 0$
  - $x \leq 0 \wedge x \geq 0 \wedge x' = x+1 \vdash x' \leq 0$
  - $x \leq 0 \wedge x \geq 0 \wedge x' = x+1 \Rightarrow x' \leq 0$
  - $x \leq 0, x \geq 0, x' = x+1 \vdash x' \leq 0$
  - $x \leq 0, x \geq 0, x' = x+1 \vdash x+1 \leq 0$
  - $x \leq 0, x \geq 0, x = 0, x' = x+1 \vdash x+1 \leq 0$
  - $x \leq 0, x \geq 0, x = 0, x' = x+1 \vdash 1 \leq 0$!

$x \leq 0$ is always true !

```
VARIABLES x
 INVARIANTS
  @inv1 x ∈ ℤ
  @inv2 x = −1
  theorem @safety1 x ≤ 0
   EVENT INITIALISATION
       then
     @act1 x := −1
   end
   EVENT event1
       where
     @grd1 x ≥ 0
       then
     @act1 x := x + 1
   end
   EVENT event2
       then
     @act1 x := x
end
```

- $x \leq 0$ is always true.
- $x = -1 \wedge BA(event1)(x, x') \Rightarrow x' = -1$ is correct
- $x \leq 0$ is a theorem
- $x = -1$ is an inductive invariant.

Current Summary

1. Invariant versus theorem in Event-B
2. Designing a C program for $\lambda x.x \times x$
3. The power function $\lambda x, y.x^y$

Designing a C program for $\lambda x.x \times x$

Computing $\lambda x.x \times x$ with only addition

The problem is to derive a C program which is computing the function $\lambda x.x \times x$ using only addition.

```
#ifndef _A_H
#define _A_H
#include <limits.h>
/*@ axiomatic auxmath {
  @ axiom  rule1: \forall int n; n >0 ==> n*n == (n-1)*(n-1)+2*n-
  @ } */

/*@  requires  0 <= x;
     requires x <= INT_MAX;
     requires x*x <= INT_MAX;
   assigns \nothing;
     ensures \result == x*x;
*/
int power2(int x);
#endif
```

Context for computing $\lambda x.x \times x$

CONTEXT $power20$
CONSTANTS $n0\ v\ w\ s$
AXIOMS
$@axm1\ n0 \in \mathbb{N} // precondition$
$@axm2\ w \in \mathbb{N} \to \mathbb{Z}$
$@axm3\ w(0) = 0$
$@axm4\ \forall n.\ n \in \mathbb{N} \Rightarrow w(n+1) = w(n) + 2$
$@axm5\ v \in \mathbb{N} \to \mathbb{Z}$
$@axm6\ v(0) = 0$
$@axm7\ \forall n.\ n \in \mathbb{N} \Rightarrow v(n+1) = v(n) + w(n) + 1$
$@axm8\ s \in \mathbb{N} \to \mathbb{N} \land (\forall i.\ i \in \mathbb{N} \Rightarrow s(i) = i + 1)$
$@axm9\ \forall A.\ A \subseteq \mathbb{N} \land 0 \in A \land s[A] \subseteq A \Rightarrow \mathbb{N} \subseteq A$
$theorem\ @axm10\ \forall n.n \in \mathbb{N} \Rightarrow w(n) = 2 * n$
$theorem\ @axm11\ \forall n.n \in \mathbb{N} \Rightarrow v(n) = n * n$
$@axm12\ n0 \geq 3$
end

Machine power21 for stating the pre/post specification

MACHINE $power21$ SEES $power20$

VARIABLES $r\ ok\ n$
INVARIANTS
$@inv1\ r \in \mathbb{Z}$
$@inv2\ n \in \mathbb{Z}$
$@inv3\ ok \in BOOL$
$@inv4\ ok = TRUE \Rightarrow r = n0 \times n0$
$@inv5\ n = n0$

- Defining variables and invariant
- r is the variable for the result.
- ok is the boolean variable used for expressing the process termination.
- n is the variable containing the input of the process.

Machine power21 for stating the pre/post specification

EVENT $INITIALISATION$
then
$@act1\ r\ :\in\ \mathbb{Z}$
$@act2\ ok\ :=\ FALSE$
$@act3\ n\ :=\ n0$
end
EVENT $final$
where
$@grd1\ ok = FALSE$
then
$@act1\ r\ :=\ v(n)$
$@act2\ ok\ :=\ TRUE$
end

- INITIALISATION is setting variables especially $n = n0$
- final is observed and gets the value $v(n)$ which is sound since $v(n) = n \times n$.
- ok controls the obsedrvation of the event final at most one time.

Machine power21 for stating the pre/post specification

```
EVENT  INITIALISATION
   then
  @act1 r  :∈  ℤ
  @act2 ok  :=  FALSE
  @act3 n  :=  n0
 end
 EVENT  final
   where
  @grd1 ok = FALSE
   then
  @act1 r  :=  v(n)
  @act2 ok  :=  TRUE
 end
```

- INITIALISATION is setting variables especially $n = n0$
- final is observed and gets the value $v(n)$ which is sound since $v(n) = n \times n$.
- ok controls the obsedrvation of the event final at most one time.
- iteration is anticipating an hidden iteration.

Machine power21 for stating the pre/post specification

EVENT $INITIALISATION$
then
$@act1\ r\ :\in\ \mathbb{Z}$
$@act2\ ok\ :=\ FALSE$
$@act3\ n\ :=\ n0$
end
EVENT $final$
where
$@grd1\ ok = FALSE$
then
$@act1\ r\ :=\ v(n)$
$@act2\ ok\ :=\ TRUE$
end

- INITIALISATION is setting variables especially $n = n0$
- final is observed and gets the value $v(n)$ which is sound since $v(n) = n \times n$.
- ok controls the obsedrvation of the event final at most one time.
- iteration is anticipating an hidden iteration.

$anticipated$ EVENT $iteration$
then
$@act1\ r, ok, n\ :|\ (r' \in \mathbb{Z}\ \wedge\ (ok' = TRUE\ \Rightarrow\ r' = n0 * n0)\ \wedge\ n' = n0\ )$
end

Machine power22 for stating the computing process

MACHINE $power22$ REFINES $power21$ SEES $power20$
VARIABLES $r\ vv\ k\ ww\ ok\ n$
INVARIANTS
$@inv1\ vv \in \mathbb{N} \nrightarrow \mathbb{Z}$
$@inv2\ ww \in \mathbb{N} \nrightarrow \mathbb{Z}$
$@inv3\ k \in \mathbb{N}$
$@inv4\ \forall i.\ i \in dom(vv) \Rightarrow vv(i) = v(i)$
$@inv5\ \forall i.\ i \in dom(ww) \Rightarrow ww(i) = w(i)$
$@inv6\ dom(vv) = 0..k$
$@inv7\ dom(ww) = 0..k$
$@inv8\ k \leq n$
$theorem\ @safe1\ \forall i.\ i \in dom(vv) \Rightarrow vv(i) = i * i$
$theorem\ @safe2\ \forall i.\ i \in dom(ww) \Rightarrow ww(i) = 2 * i$
$@inv11\ k < n \Rightarrow ok = FALSE$

- Two new variables vv and ww are introduced for storing the two sequences v and w by iterating over k
- Condition of termination is that $n \in dom(vv)$
- $vv(i) = v(i)$ and $ww(i) = w(i)$ are expressing the relationship between computed values and mathematically defned values of the two sequences.

Machine power22 for stating the computing process

EVENT $INITIALISATION$
then
$@act1\ r\ :\in\ \mathbb{Z}$
$@act2\ vv\ :=\ \{0 \mapsto 0\}$
$@act3\ ww\ :=\ \{0 \mapsto 0\}$
$@act4\ k\ :=\ 0$
$@act5\ ok\ :=\ FALSE$
$@act6\ n\ :=\ n0$
end

- INITIALISATION is setting variables especially ww and vv
- Sequences v and w are used for intialisation.

Machine power22 for stating the computing process

EVENT $final$ REFINES $final$
where
@$grd1$ $n \in dom(vv)$
@$grd2$ $ok = FALSE$
then
@$act1$ $r := vv(n)$
@$act2$ $ok := TRUE$
end

$convergent$ EVENT $step-computing$
REFINES $iteration$
where
@$grd1$ $n \notin dom(vv)$
@$grd2$ $ok = FALSE$
then
@$act1$ $vv(k+1) := vv(k) + ww(k) + 1$
@$act2$ $k := k+1$
@$act3$ $ww(k+1) := ww(k) + 2$
end

VARIANT $n - k$

- the event final is controled by the condition $n \in dom(vv)$ meaning that we have finally reached the computing goal.
- SIM proof obligations are generated.
- the event step-computng is refining iteration and when it observed, the variant $n - k$ is decreasing.
- it refines iteration

Machine power23 for getting an algorithmic process

MACHINE $power23$
REFINES $power22$
SEES $power20$

VARIABLES $r\ vv\ k\ cv\ ww\ cw\ ok\ n$

INVARIANTS
$@inv1\ cv \in \mathbb{Z}$
$@inv2\ cv = vv(k)$
$@inv3\ cw \in \mathbb{Z}$
$@inv4\ cw = ww(k)$
$theorem\ @inv5\ k \in 0..n$
$theorem\ @inv6\ cw = 2 * k$
$theorem\ @inv7\ cv = k * k$
$theorem\ @inv8\ 4 * cv = cw * cw$

- Two new variables are introduced for storing really useful data namey the last computed values of the two sequences.
- Obviously, $cv = vv(k)$ and $cw = ww(k)$
- Previous properties of abstrcat variables are safety properties which are no more to be reproved, thanks to refienement.
- We can get extra properties that are relating the variables as $4 \times cv = cw \times cw$.

Machine power23 for getting an algorithmic process

EVENT $INITIALISATION$
then
$@act1\ r :\in \ \mathbb{Z}$
$@act2\ vv \ := \ \{0 \mapsto 0\}$
$@act3\ k \ := \ 0$
$@act4\ cv \ := \ 0$
$@act5\ ww \ := \ \{0 \mapsto 0\}$
$@act6\ cw \ := \ 0$
$@act7\ ok \ := \ FALSE$
$@act8\ n \ := \ n0$
end

- Initialisation of new variables according to the invariant.

Machine power23 for getting an algorithmic process

EVENT $final$ REFINES $final$
where
$@grd1\ k = n$
then
$@act1\ r\ := \ cv$
$@act2\ ok\ := \ TRUE$
end

$convergent$ EVENT $step - prealgo$
REFINES $step - computing$
where
$@grd1\ k\ <\ n$
then
$@act1\ vv(k+1)\ := \ vv(k) + ww(k) + 1$
$@act2\ k\ := \ k+1$
$@act3\ cv\ := \ cv + cw + 1$
$@act4\ ww(k+1)\ := \ ww(k) + 2$
$@act5\ cw\ := \ cw + 2$
end

VARIANT $n - k$

- The two events SIMulate the abstrcat events.
- However, the guards are strengthened and are made closer to an implmentation : $k < n$ implies $n \notin dom(vv)$ and $k = n$ implies that $n \in dom(vv)$.

Machine power24 for getting an algorithmic machine

MACHINE $power24$ REFINES $power23$
SEES $power20$

VARIABLES $r\ k\ cv\ cw\ ok\ n$

INVARIANTS
$theorem\ @th1\ cw = 2 * k$
$theorem\ @th2\ cv = k * k$
$theorem\ @inv1th3\ 4 * cv = cw * cw$

- The two variables vv and ww are now hidden and they disappear from the machine.
- They are playing the role of model variables as ghost variables.
- Invariants and safety properties are preserved through refinement.

Machine power24 for getting an algorithmic machine

EVENT $INITIALISATION$
then
$@act1\ r :\in \mathbb{Z}$
$@act5\ k := 0$
$@act8\ cv := 0$
$@act10\ cw := 0$
$@act11\ ok := FALSE$
$@act12\ n := n0$
end

- INITILISATION is the same event without vv and ww.

Machine power24 for getting an algorithmic machine

```
EVENT final REFINES final
  where
  @grd1 k = n
  then
  @act1 r := cv
  @act2 ok := TRUE
end

convergent EVENT step
 REFINES step − prealgo
  where
  @grd1 k < n
  then
  @act4 k := k + 1
  @act5 cv := cv + cw + 1
  @act7 cw := cw + 2
end
```

- Assignments of vv and ww are removed.

Translating the machine power24 to an algorithm

$$
\begin{array}{l}
begin \\
\ int\, r,k \ := \ 0, \ cv \ := \ 0, \ cw \ := \ 0, \ ok \ := \ FALSE, n \ := \ n0; \\
\ while\, k \ < \ n \ \{ \\
\ (\ k, cv, cv) \ := \ (k+1, cv+cw+1, cw \ := \ cw+2); \\
\ \} \ ; \\
\ r \ := \ cv; \\
\ ok \ := \ TRUE \\
\ end
\end{array}
$$

Translating the machine power24 to an algorithm

```
#include <limits.h>
#include "ppower2.h"

int power2(int x)
{int  r,k,cv,cw,or,ok,ocv,ocw;
  r=0;k=0;cv=0;cw=0;or=0;ok=k;ocv=cv;ocw=cw;
      /*@ loop invariant cv == k*k;
          @ loop invariant k <= x;
          @ loop invariant cw == 2*k;
          @ loop invariant 4*cv == cw*cw;
          @ loop assigns k,cv,cw,or,ok,ocv,ocw;
           @ loop variant x-k;
*/
  while (k<x)
        {
          ok=k;ocv=cv;ocw=cw;
          k=ok+1;
          cv=ocv+ocw+1;
          cw=ocw+2;

        }
  r=cv;
  return(r);
}
```

Current Summary

1. Invariant versus theorem in Event-B
2. Designing a C program for $\lambda x.x \times x$
3. The power function $\lambda x, y.x^y$

Contract for $\lambda x, y.x^y$

VARIABLES $r, done, a, b$

$<<<<<<<<<<<$ **REQUIRES** $\left(\begin{array}{l} a_0, b_0 \in \mathbb{N} \\ r_0 \in \mathbb{Z} \\ done_0 = FALSE \end{array}\right.$

ENSURES $\left(\begin{array}{l} r_f = p(a \mapsto b) \\ a_f = a_0 \wedge b_f = b_9 \end{array}\right.$

Steps

- Defining the context for the fuynction p.
- Stating the contract in the Event-B language@
- Introducing a computing process
- Making the computing process computable
- Transforming intro a programming labgauge
- Annotating the algorithm
- Checking the resulting annotated algorithm

POW0

CONTEXT $POW0$

CONSTANTS $a0\ b0\ p\ s$

AXIOMS

$@axm1\ a0 \in \mathbb{N} \land b0 \in \mathbb{N}$

$@axm2\ s \in \mathbb{N} \rightarrow \mathbb{N}$

$@axm3\ \forall\, n.\ n \in \mathbb{N} \Rightarrow s(n) = n + 1$

$@axm4\ p \in \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

$@axm5\ \forall\, a.\ a \in \mathbb{N} \Rightarrow p(a \mapsto 0) = 1$

$@axm6\ \forall\, a, b.\ a \in \mathbb{N} \land b \in \mathbb{N} \Rightarrow p(a \mapsto b + 1) = p(a \mapsto b) * a$

$@axm7\ \forall\, A.\ A \subseteq \mathbb{N} \land 0 \in A \land s[A] \subseteq A \Rightarrow \mathbb{N} \subseteq A$

$theorem\ @th\ \forall\, a, b.\ a \in \mathbb{N} \land b \in \mathbb{N} \Rightarrow p(a \mapsto b) = a^b$

end

MACHINE $POW1$ SEES $POW0$

VARIABLES $r\ done\ a\ b$

INVARIANTS

$@inv1\ r \in \mathbb{Z} \wedge done \in BOOL$

$@inv2\ done = TRUE \Rightarrow r = p(a0 \mapsto b0)$

$@inv3\ a = a0 \wedge b = b0$

EVENTS

EVENT $INITIALISATION$

then

$@act1\ r :\in \mathbb{Z}$

$@act2\ done := FALSE$

$@act3\ a := a0$

$@act4\ b := b0$

end

EVENT $computing1$

where

$@grd1\ done = FALSE$

then

$@act1\ r := p(a \mapsto b)$

$@act2\ done := TRUE$

end

end

MACHINE $POW2$
REFINES $POW1$
SEES $POW0$
VARIABLES $r\ done\ a\ b\ pp\ k$
INVARIANTS
$@inv1\ k \in 0..b0$
$@inv2\ pp \in \mathbb{N} \leftrightarrow \mathbb{N}$
$@inv3\ \forall i.\ i \in dom(pp)$
$\Rightarrow pp(i) = p(a0 \mapsto i)$
$@inv4\ dom(pp) = 0..k$
VARIANT $b0 - k$
EVENTS
EVENT $INITIALISATION$
then
$@act1\ r :\in \mathbb{Z}$
$@act2\ done := FALSE$
$@act3\ a := a0$
$@act4\ b := b0$
$@act5\ k := 0$
$@act6\ pp := \{0 \mapsto 1\}$
$@act7$
end

EVENT $computing2$ REFINES $computing1$
where
$@grd1\ done = FALSE$
$@grd2\ b0 \in dom(pp)$
then
$@act1\ r := pp(b0)$
$@act2\ done := TRUE$
end

$convergent$ EVENT $step2$
where
$@grd1\ done = FALSE$
$@grd2\ b0 \notin dom(pp)$
then
$@act1\ k := k+1$
$@act2\ pp(k+1) := pp(k) * a$
end

POW3

MACHINE $POW3$
REFINES $POW2$
SEES $POW0$
VARIABLES $r\ done\ a\ b\ pp\ k\ cp$
INVARIANTS
$@inv1\ cp \in \mathbb{Z}$
$@inv2\ cp = pp(k)$
$@inv3\ k < b0 \Rightarrow done = FALSE$
$@inv4\ done = TRUE \Rightarrow k = b0$
EVENTS
EVENT $INITIALISATION$
then
$@act1\ r :\in \mathbb{Z}$
$@act2\ done := FALSE$
$@act3\ a := a0$
$@act4\ b := b0$
$@act5\ k := 0$
$@act6\ pp := \{0 \mapsto 1\}$
$@act7\ cp := 1$
end

EVENT $computing3$ REFINES $computing2$
where
$@grd2\ k = b0$
$@grd3\ done = FALSE$
then
$@act1\ r := cp$
$@act2\ done := TRUE$
end

EVENT $step2$ REFINES $step2$
where
$@grd1\ done = FALSE$
$@grd2\ k < b0$
then
$@act1\ k := k + 1$
$@act2\ pp(k+1) := pp(k) * a$
$@act3\ cp := cp * a$
end

POW4

MACHINE $POW4$
REFINES $POW3$
SEES $POW0$
VARIABLES $r\ done\ a\ b\ k\ cp$
INVARIANTS
$theorem\ @inv1\ cp = p(a \mapsto k)$
$@inv2\ done = TRUE \ \Rightarrow\ cp = a^b$
$theorem\ @inv3\ a = a0\ \wedge\ b = b0$
EVENTS
EVENT $INITIALISATION$
then
$@act1\ r :\in\ \mathbb{Z}$
$@act2\ done\ :=\ FALSE$
$@act3\ a\ :=\ a0$
$@act4\ b\ :=\ b0$
$@act5\ k\ :=\ 0$
$@act7\ cp\ :=\ 1$
end

EVENT $computing4$ REFINES $computing3$
where
$@grd2\ k\ =\ b0$
$@grd3\ done = FALSE$
then
$@act1\ r\ :=\ cp$
$@act2\ done\ :=\ TRUE$
end

EVENT $step4$ REFINES $step2$
where
$@grd1\ done =\ FALSE$
$@grd2\ k\ <\ b0$
then
$@act1\ k\ :=\ k+1$
$@act3\ cp\ :=\ cp\ *\ a$
end

Deriving the algorithm

```
int power(int a0,int b0)
{
  int  a=a0,b=b0,r,k=0,cp=1;
   while (k<b)
    {
       cp=cp*a;
       k=k+1;
    }
  r=cp;
  return(r);
}
```