Almost certain termination and Rabin's Choice-Coordination algorithm

Annabelle McIver
Carroll Morgan
Thai Son Hoang
Zhendong Jin
Ken Robinson

May 30, 2003

1 Challenge and aims

- Implement termination with probability one into the B-Toolkit.
- Specify the Choice-Coordination problem and implement Rabin's solution.
- Generate and prove the obligations for the development.

2 What is termination with probability one?

Consider the following programs:

$n := 2$; WHILE $n \neq 0$ DO $\quad n := n - 1$ END	$n := 1$; WHILE $n \neq 0$ DO $\quad n := n - 1 \, [\!] \, SKIP$ END	$n := 1$; WHILE $n \neq 0$ DO $\quad n := n - 1 \; {}_{0.5}\!\oplus SKIP$ END
Program A	Program B	Program C

- Program A: *Absolute* correctness.
- Program B: *Demonic* incorrectness.
- Program C: *Almost-certain* correctness.

Consider Program D:

$n := 1$;
WHILE $n \neq 0$ DO
$\quad n := n - 1 \; {}_{p}\!\oplus SKIP$
END

Program D

Let t be the probability of termination:

$$
\begin{aligned}
& t = p + (1-p) \times t \\
\equiv \quad & p = p \times t \\
\equiv \quad & t = 1 \text{ provided that } p \neq 0
\end{aligned}
$$

3 Abstract probabilistic choice substitution

Program D will still terminate with probability one without knowing the actual probability p.

Consider Program E:

$n := 1;$
WHILE $n \neq 0$ DO
    $n := n - 1 \oplus SKIP$
END

Program E

$S \oplus T$ is the abstract probability choice substitution between S and T. The $\oplus$ should be implemented by "concrete" probabilistic choices ${}_p\oplus$ that are bounded away from 0 and 1.

4 Proof rules for loops

WHILE G DO
    S
END

with INVARIANT I and VARIANT V.

- Partial correctness condition: This can be proved using INVARIANT I. If the loop terminates, it is correct.
- Total correctness condition: The loop is partially correct and terminates. This can be proved using INVARIANT I and VARIANT V.

4.1 Original proof rules for loops

- Partial correctness condition: The INVARIANT I is maintained during the loop.
- Total correctness condition:
  - The VARIANT V is bounded below.
  - For every iteration of the loop, the VARIANT V strictly decreases.

4.2 New proof rules for loops

- Partial correctness condition: The INVARIANT I is maintained during the loop. *All abstract probabilistic choices $\oplus$ are interpreted demonically.*
- Total correctness condition:
  - The VARIANT V is bounded below.
  - *The VARIANT V is bounded above.*
  - For every iteration of the loop, the VARIANT V decreases *with a non-zero probability, i.e. treating all abstract probabilistic choices angelically.*

5 Choice-Coordination problem

Originally, the prolem was explained in terms of different processes try to decided on one possible outcome.

Rabin's algorithm provides a symmetric, distributed solution for the problem.

There is a group of tourists trying to decide between going to the church (which is on the "left") and the museum (which is on the "right"). Every tourist runs the same algorithm independently.

6 Rabin's algorithm

The actual algorithm as follows:

- Each tourist carries a notepad, on which he will write various numbers. Originally, number 0 appears on all the notepads.
- There are two noticeboard outside, "left" and "right", on which various messages will be written. Originally, number 0 appears on each board.

Each tourist (with number k on his pad) will alternate between the two places. Every time he goes to a place, if the noticeboard displays "here" then he goes inside, otherwise, it will display a number (K):

- if $k < K$ — The tourist writes K on his notepad in place of k, and goes to the other place.
- if $k > K$ — The tourist writes "here" on the noticeboard (erasing K), and goes inside.
- if $k = K$ — The tourist chooses $K' = K + 2$, and then flips a coin: if it comes up heads, he changes the value of K' to the "conjugate" value of K'. He then writes K' on the noticeboard and on his pad before going to the other place.

This algorithm terminates with probability 1. [1]

[1] *Note: Conjugate*(n) is $n + 1$ if n is even, and $n - 1$ if n is odd.

7 The specification

MACHINE *Rabin* (*maxtotal*)

CONSTRAINTS $maxtotal \leq 2147483646$

OPERATIONS

$lin , rin \longleftarrow$ **Decide** (*lout* , *rout*) $\widehat{=}$
  PRE $lout \in \mathbb{N} \wedge rout \in \mathbb{N} \wedge lout + rout \leq maxtotal$ **THEN**
    CHOICE $lin := lout + rout \;||\; rin := 0$
    OR $rin := lout + rout \;||\; lin := 0$
    END
  END

END

8 The refinement using bags

We will use bags to model the tourists inside and outside the two places. In the first refinement we will only be concerned with the number of items in each bag.

REFINEMENT *RabinR*

REFINES *Rabin*

SEES *FBag_ctx*

INCLUDES *lin . FBag* , *rin . FBag* , *lout . FBag* , *rout . FBag*

OPERATIONS

$lin , rin \longleftarrow$ **Decide** (*lout* , *rout*) $\widehat{=}$
  BEGIN
    ANY *flinbag* , *frinbag* , *floutbag* , *froutbag* **WHERE**
      $flinbag \in Bag \wedge frinbag \in Bag \wedge floutbag \in Bag \wedge froutbag \in Bag \wedge$
      $\mathrm{dom} (flinbag) \in \mathbb{F} (\mathbb{N}) \wedge \mathrm{dom} (frinbag) \in \mathbb{F} (\mathbb{N}) \wedge$
      $\mathrm{dom} (floutbag) \in \mathbb{F} (\mathbb{N}) \wedge \mathrm{dom} (froutbag) \in \mathbb{F} (\mathbb{N}) \wedge$
      $floutbag = \{\} \wedge froutbag = \{\} \wedge$
      $(bagSize (flinbag) = 0 \vee bagSize (frinbag) = 0) \wedge$
      $bagSize (flinbag) + bagSize (frinbag) = lout + rout$
    THEN
      lin . SetToBag (*flinbag*) || *rin . SetToBag* (*frinbag*) ||
      lout . SetToBag (*floutbag*) || *rout . SetToBag* (*froutbag*)
    END **;**
    $lin \longleftarrow lin . Size \;||\; rin \longleftarrow rin . Size$
  END

END

9 The implementation

IMPLEMENTATION *RabinRI*

REFINES *RabinR*

SEES *Bool_TYPE* , *FBag_ctx* , *Math*

IMPORTS *RabinChoice* (*maxtotal*)

OPERATIONS

```
lin , rin <-- Decide ( lout , rout )  ≙
  VAR  sizelout , sizerout  IN
    InitState ( lout , rout ) ;
    sizelout <-- loutSize ; sizerout <-- routSize ;
    WHILE  sizelout ≠ 0 ∨ sizerout ≠ 0  DO
      UpdatePad ;
      sizelout <-- loutSize ; sizerout <-- routSize
    BOUND  9 × total
    VARIANT
      rEqual ( LL , RR ) × 3 × total + 3 × total − (
        3 × ( bagSize ( linbag ) + bagSize ( rinbag ) ) +
        ( bagGreat ( loutbag , LL ) + bagGreat ( routbag , LL ) ) +
        ( bagGreat ( loutbag , RR ) + bagGreat ( routbag , RR ) ) )
    INVARIANT
      bagSize ( loutbag ) = sizelout ∧ bagSize ( routbag ) = sizerout ∧
      total = lout + rout
    END  ;
    lin <-- linSize ; rin <-- rinSize
  END
```

END

10 Supporting the implementation

```
MACHINE    RabinChoice ( maxtotal )

CONSTRAINTS    maxtotal ≤ 2147483646

SEES    Math , Bool_TYPE , FBag_ctx

INCLUDES    RabinState ( maxtotal )

PROMOTES    linSize , rinSize , loutSize , routSize , InitState

INVARIANT

   LL ↦ RR ∈ dom ( rEqual ) ∧
   ¬ ( Conjugate ( LL ) ∈ ran ( routbag ) ) ∧ ¬ ( Conjugate ( RR ) ∈ ran ( loutbag ) )

OPERATIONS

   UpdatePad  ≙
     PRE    bagSize ( loutbag ) ≠ 0 ∨ bagSize ( routbag ) ≠ 0    THEN
       SELECT    bagSize ( loutbag ) ≠ 0    THEN
         ANY    ll    WHERE    ll ∈ ran ( loutbag )    THEN
           SELECT    linbag = {} ∧ ll < LL    THEN    MoveToRight ( ll , LL )
           WHEN    bagSize ( linbag ) ≠ 0 ∨ ll > LL    THEN    MoveInLeft ( ll )
           WHEN    linbag = {} ∧ ll = LL    THEN
             LET    newLL    BE    newLL = LL + 2    IN
               ACHOICE    MoveToRight ( ll , newLL )
               OR    MoveToRight ( ll , Conjugate ( newLL ) )
               END
             END
           END
         END
       WHEN    bagSize ( routbag ) ≠ 0    THEN
         ANY    rr    WHERE    rr ∈ ran ( routbag )    THEN
           SELECT    bagSize ( rinbag ) = 0 ∧ rr < RR    THEN    MoveToLeft ( rr , RR )
           WHEN    bagSize ( rinbag ) ≠ 0 ∨ rr > RR    THEN    MoveInRight ( rr )
           WHEN    bagSize ( rinbag ) = 0 ∧ rr = RR    THEN
             LET    newRR    BE    newRR = RR + 2    IN
               ACHOICE    MoveToLeft ( rr , newRR )
               OR    MoveToLeft ( rr , Conjugate ( newRR ) )
               END
             END
           END
         END
       END
     END

END
```

MACHINE *RabinState* (*maxtotal*)

CONSTRAINTS $maxtotal \leq 2147483646$

SEES

Math , *FBag_ctx*

INCLUDES *lin* . *FBag* , *rin* . *FBag* , *lout* . *FBag* , *rout* . *FBag*

PROMOTES

lin . *Size* , *rin* . *Size* , *lout* . *Size* , *rout* . *Size* , *lout* . *Anyelem* , *rout* . *Anyelem*

VARIABLES

total , *LL* , *RR*

INVARIANT

$total \in \mathbb{N} \wedge$
$total = bagSize\ (\ linbag\ ) + bagSize\ (\ rinbag\ ) +$
$\quad (\ bagSize\ (\ loutbag\ ) + bagSize\ (\ routbag\ )\ ) \wedge$
$LL \in \mathbb{N} \wedge RR \in \mathbb{N} \wedge$
$maxInBag\ (\ linbag\ ) \leq RR \wedge maxInBag\ (\ loutbag\ ) \leq RR \wedge$
$maxInBag\ (\ rinbag\ ) \leq LL \wedge maxInBag\ (\ routbag\ ) \leq LL \wedge$
$(\ bagSize\ (\ linbag\ ) \neq 0 \Rightarrow maxInBag\ (\ linbag\ ) > LL\ ) \wedge$
$(\ bagSize\ (\ rinbag\ ) \neq 0 \Rightarrow maxInBag\ (\ rinbag\ ) > RR\ ) \wedge$
$3 \times total \geq$
$\quad 3 \times (\ bagSize\ (\ linbag\ ) + bagSize\ (\ rinbag\ )\ ) +$
$\quad (\ bagGreat\ (\ loutbag\ ,\ LL\ ) + bagGreat\ (\ routbag\ ,\ LL\ )\ ) +$
$\quad (\ bagGreat\ (\ loutbag\ ,\ RR\ ) + bagGreat\ (\ routbag\ ,\ RR\ )\ )$

INITIALISATION

$total := 0 \ \ || \ \ LL\ ,\ RR := 0\ ,\ 0$

OPERATIONS

InitState (*lout* , *rout*) $\widehat{=}$

PRE $lout \in \mathbb{N} \wedge rout \in \mathbb{N} \wedge lout + rout \leq maxtotal$ **THEN**

lin . $SetToBag$ ({}) || rin . $SetToBag$ ({}) ||

$lout$. $SetToBag$ ((1 .. $lout$) $\times$ { 0 }) || $rout$. $SetToBag$ ((1 .. $rout$) $\times$ { 0 }) ||

$total := lout + rout$ || $LL := 0$ || $RR := 0$

END **;**

MoveInLeft (*ll*) $\widehat{=}$

PRE $ll \in \mathsf{ran}\ (loutbag) \wedge (bagSize\ (linbag) \neq 0 \vee ll > LL)$ **THEN**

$lout$. $Takelem$ (ll) || lin . $Addelem$ (ll)

END **;**

MoveInRight (*rr*) $\widehat{=}$

PRE $rr \in \mathsf{ran}\ (routbag) \wedge (bagSize\ (rinbag) \neq 0 \vee rr > RR)$ **THEN**

$rout$. $Takelem$ (rr) || rin . $Addelem$ (rr)

END **;**

MoveToLeft (*rr* , *mm*) $\widehat{=}$

PRE $rr \in \mathsf{ran}\ (routbag) \wedge mm \in \mathbb{N}_1 \wedge bagSize\ (rinbag) = 0 \wedge RR \leq mm$ **THEN**

$rout$. $Takelem$ (rr) || $lout$. $Addelem$ (mm) || $RR := mm$

END **;**

MoveToRight (*ll* , *mm*) $\widehat{=}$

PRE $ll \in \mathsf{ran}\ (loutbag) \wedge mm \in \mathbb{N}_1 \wedge bagSize\ (linbag) = 0 \wedge LL \leq mm$ **THEN**

$lout$. $Takelem$ (ll) || $rout$. $Addelem$ (mm) || $LL := mm$

END **;**

$ll \longleftarrow$ **LLVal** $\widehat{=}$ $ll := LL$ **;**

$rr \longleftarrow$ **RRVal** $\widehat{=}$ $rr := RR$

END