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:

<table>
<thead>
<tr>
<th>$n := 2$</th>
<th>$n := 1$</th>
<th>$n := 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHILE $n \neq 0$ DO</td>
<td>WHILE $n \neq 0$ DO</td>
<td>WHILE $n \neq 0$ DO</td>
</tr>
<tr>
<td>$n := n - 1$</td>
<td>$n := n - 1 \parallel SKIP$</td>
<td>$n := n - 1.5 \parallel SKIP$</td>
</tr>
<tr>
<td>END</td>
<td>END</td>
<td>END</td>
</tr>
</tbody>
</table>

Program A | Program B | Program C

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

Consider Program D:

<table>
<thead>
<tr>
<th>$n := 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHILE $n \neq 0$ DO</td>
</tr>
<tr>
<td>$n := n - 1 \parallel SKIP$</td>
</tr>
<tr>
<td>END</td>
</tr>
</tbody>
</table>

Program D

Let $t$ be the probability of termination:
\[
\begin{align*}
  t &= p + (1 - p) \times t \\
  &\equiv p \times t \\
  &\equiv t = 1 \text{ provided that } p \neq 0
\end{align*}
\]

3 Abstract probabilistic choice substitution

Program D will still terminate with probability one without knowing the actual probability \( p \).
Consider Program E:

\[
\begin{align*}
  n &:= 1; \\
  \text{WHILE } n \neq 0 \text{ DO} \\
  &\quad n := n - 1 \oplus \text{SKIP} \\
\end{align*}
\]

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 problem 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 noticeboards 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 ≤ 2147483646

OPERATIONS

\[
\text{lin, rin} \leftarrow \text{Decide (lout, rout)} \equiv \\
\text{PRE} \quad \text{lout} \in \mathbb{N} \land \text{rout} \in \mathbb{N} \land \text{lout} + \text{rout} \leq \text{maxtotal} \quad \text{THEN} \\
\text{CHOICE} \quad \text{lin} := \text{lout} + \text{rout} \quad \text{|| rin} := 0 \\
\text{OR} \quad \text{rin} := \text{lout} + \text{rout} \quad \text{|| lin} := 0 \\
\text{END} \\
\text{END}
\]

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

\[
\text{lin, rin} \leftarrow \text{Decide (lout, rout)} \equiv \\
\text{BEGIN} \\
\text{ANY} \quad \text{flinbag, frinbag, ffloutbag, ffrontbag WHERE} \\
\quad \text{flinbag} \in \text{Bag} \land \text{frinbag} \in \text{Bag} \land \text{ffloutbag} \in \text{Bag} \land \text{ffrontbag} \in \text{Bag} \land \\
\quad \text{dom (flinbag)} \in \mathbb{F} (\mathbb{N}) \land \text{dom (frinbag)} \in \mathbb{F} (\mathbb{N}) \land \\
\quad \text{dom (ffloutbag)} \in \mathbb{F} (\mathbb{N}) \land \text{dom (ffrontbag)} \in \mathbb{F} (\mathbb{N}) \land \\
\quad \text{ffloutbag} = \{\} \land \text{ffrontbag} = \{\} \land \\
\quad (\text{bagSize (flinbag)} = 0 \lor \text{bagSize (frinbag)} = 0) \land \\
\quad \text{bagSize (flinbag)} + \text{bagSize (frinbag)} = \text{lout} + \text{rout} \land \\
\text{THEN} \\
\quad \text{lin} \cdot \text{SetToBag (flinbag)} \lor \text{rin} \cdot \text{SetToBag (frinbag)} \\
\quad \text{lout} \cdot \text{SetToBag (ffloutbag)} \lor \text{rout} \cdot \text{SetToBag (ffrontbag)} \\
\text{END ;} \\
\quad \text{lin} \leftarrow \text{lin} \cdot \text{Size} \lor \text{rin} \leftarrow \text{rin} \cdot \text{Size} \\
\text{END}
\]

END
9 The implementation

IMPLEMENTATION RobinRI
REFINES RobinR
SEES Bool.TYPE, FBAG.ctx, Math
IMPORTS RobinChoice ( maxtotal )

OPERATIONS
\[
\text{lin}, \text{rin} \leftarrow \text{Decide} (\text{lout}, \text{ rout}) \triangleq
\]
\begin{align*}
\text{VAR} & \quad \text{sizelout}, \text{sizerout} \ \text{IN} \\
\text{InitState} (\text{lout}, \text{ rout}) ; \\
\text{sizelout} \leftarrow \text{loutSize} ; \text{sizerout} \leftarrow \text{ routSize} ; \\
\text{WHILE} & \quad \text{sizelout} \neq 0 \lor \text{sizerout} \neq 0 \ \text{DO} \\
\text{UpdatePad} ; \\
\text{sizelout} \leftarrow \text{loutSize} ; \text{sizerout} \leftarrow \text{ routSize} \\
\text{BOUND} & \quad 9 \times \text{total} \\
\text{VARIANT} & \quad r\text{Equal} (\text{LL}, \text{RR}) \times 3 \times \text{total} + 3 \times \text{total} - ( \\
& \quad 3 \times (\text{bagSize (linbag)} + \text{bagSize (rinbag)}) + \\
& \quad (\text{bagGreat (loutbag, LL)} + \text{bagGreat (routbag, LL)}) + \\
& \quad (\text{bagGreat (loutbag, RR)} + \text{bagGreat (routbag, RR)}))
\end{align*}

INVARIANT
\[
\text{bagSize (loutbag)} = \text{sizelout} \land \text{bagSize (routbag)} = \text{sizerout} \land \\
\text{total} = \text{lout} + \text{rout}
\]

END ;
\[
\text{lin} \leftarrow \text{linSize} ; \text{rin} \leftarrow \text{rinSize}
\]

END

END
10 Supporting the implementation

MACHINE RabinChoice ( maxtotal )

CONSTRAINTS maxtotal \leq 2147483646

SEES Math, Bool TYPE, FBag, etc

INCLUDES RabinState ( maxtotal )

PROMOTES linSize, rinSize, loutSize, routSize, InitState

INARIANT

\[ LL \mapsto RR \in \text{dom} ( rEqual ) \land \\
\neg ( \text{Conjugate} ( LL ) \in \text{ran} ( routbag ) ) \land \\
\neg ( \text{Conjugate} ( RR ) \in \text{ran} ( loutbag ) ) \]

OPERATIONS

UpdatePad :=

PRE bagSize ( routbag ) \neq 0 \lor bagSize ( routbag ) \neq 0 THEN

SELECT bagSize ( routbag ) \neq 0 THEN

ANY \mbox{ } il WHERE il \in \text{ran} ( loutbag ) THEN

SELECT linbag = {} \land il < LL THEN MoveToRight ( il, LL )

WHEN bagSize ( linbag ) \neq 0 \lor il > LL THEN MoveInLeft ( il )

WHEN linbag = {} \land il = LL THEN

LET newLL BE newLL = LL + 2 IN

ACHOICE MoveToRight ( il, newLL )

OR MoveToRight ( il, Conjugate ( newLL ) )

END

END

END

WHEN bagSize ( routbag ) \neq 0 THEN

ANY \mbox{ } ir WHERE ir \in \text{ran} ( routbag ) THEN

SELECT bagSize ( rinbag ) = 0 \land ir < RR THEN MoveToLeft ( ir, RR )

WHEN bagSize ( rinbag ) \neq 0 \land ir > RR THEN MoveInRight ( ir )

WHEN bagSize ( rinbag ) = 0 \land ir = RR THEN

LET newRR BE newRR = RR + 2 IN

ACHOICE MoveToLeft ( ir, newRR )

OR MoveToLeft ( ir, Conjugate ( newRR ) )

END

END

END

END
MACHINE  \texttt{RabinState (maxtotal)}

CONSTRAINTS  \texttt{maxtotal} \leq 2147483646

SEES  
\texttt{Math, FBag.ctx}

INCLUDES  \texttt{lin . FBag, rin . FBag, lout . FBag, rout . FBag}

PROMOTES  

VARIABLES  
\texttt{total, LL, RR}

IN Variant

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

INITIALISATION

\texttt{total := 0 || LL, RR := 0, 0}
OPERATIONS

InitState ( out , rout ) ≜
  PRE  out ∈ \mathbb{N} \land rout ∈ \mathbb{N} \land out + rout ≤ maxtotal  THEN
  lin . SetToBag ( {} ) || rin . SetToBag ( {} ) ||
  out . SetToBag ( 1 .. out × \{ 0 \} ) || rout . SetToBag ( 1 .. rout × \{ 0 \} ) ||
  total := out + rout || LL := 0 || RR := 0
END ;

MoveInLeft ( ll ) ≜
  PRE  ll ∈ ran ( routbag ) \land ( bagSize ( linbag ) ≠ 0 \lor ll > LL )  THEN
  out . Takelem ( ll ) || lin . Addelem ( ll )
END ;

MoveInRight ( rr ) ≜
  PRE  rr ∈ ran ( routbag ) \land ( bagSize ( rinbag ) ≠ 0 \lor rr > RR )  THEN
  rout . Takelem ( rr ) || rin . Addelem ( rr )
END ;

MoveToLeft ( rr , mm ) ≜
  PRE  rr ∈ ran ( routbag ) \land mm ∈ \mathbb{N}_1 \land bagSize ( rinbag ) = 0 \land RR ≤ mm  THEN
  rout . Takelem ( rr ) || out . Addelem ( mm ) || RR := mm
END ;

MoveToRight ( ll , mm ) ≜
  PRE  ll ∈ ran ( routbag ) \land mm ∈ \mathbb{N}_1 \land bagSize ( linbag ) = 0 \land LL ≤ mm  THEN
  out . Takelem ( ll ) || rout . Addelem ( mm ) || LL := mm
END ;

ll ← LLVal ≜ ll := LL ;

rr ← RRVal ≜ rr := RR
END