

Formal Methods for Probabilistic Systems

Annabelle McIver
Carroll Morgan

- Source-level program logic
- Meta-theorems for loops
 - Introduction and example
 - Review of rules for standard loops
 - Rules for probabilistic loops
 - Analysis of an example
 - Probability-one termination
 - The Zero-One Law

Introduction and example

```
// Implement  $p \oplus$  using unbiased random bits only.
x := p;
b := true  $_{1/2} \oplus$  false;
do b →
  x := 2x;
  if x ≥ 1 then x := x-1 fi;
  b := true  $_{1/2} \oplus$  false;
od;

if x ≥ 1/2 // Variable x at least 1/2 with probability exactly p.
then ...
else ...
fi;
```

Example due to Joe Hurd (Cambridge, now Oxford).

CC Morgan. *Proof rules for probabilistic loops*. Proc. BCS-FACS 7th Refinement Workshop. Springer, 1996. ewic.bcs.org/conferences/1996/refinement/papers/paper10.htm

Proof rules for standard loops

```
x, b, e := 1, B, E;
do e ≠ 0 →
  if even e
  then b, e := b2, e ÷ 2
  else e, x := e-1, x × b
  fi
od
```

Set x to B^E in logarithmic time.

RW Floyd. *Assigning meanings to programs*. Proc. Symp. Appl. Math. Mathematical Aspects of Computer Science 19:19-32, JT Schwartz (ed.). American Math. Soc. 1967.

CAR Hoare, 1969.
EW Dijkstra, 1975.
Gries, Backhouse, Kaldewaij, Cohen...

Proof rules for standard loops

```
{ B > 0 and E ≥ 0 }
x, b, e := 1, B, E;
{ b > 0 and e ≥ 0 and BE = x × be }
do e ≠ 0 →
  { ... and e > 0 }
  if even e
  then { e ≥ 2 and even e ... b, e := b2, e ÷ 2 { BE = x × be }
        ... and BE = x × be }
  else { BE = x × be } e, x := e-1, x × b { BE = x × be }
  fi
  { BE = x × be }
od
{ BE = x × be and e = 0 }
{ x = BE }
```

RW Floyd. *Assigning meanings to programs*. Proc. Symp. Appl. Math. Mathematical Aspects of Computer Science 19:19-32, JT Schwartz (ed.). American Math. Soc. 1967.

Proof rules for standard loops

```

{ B > 0 and E ≥ 0 }
x, b, e := 1, B, E;
{ b > 0 and e ≥ 0 and BE = x × be }
do e ≠ 0 →
  { ... and e > 0 }
  if even e
  then { e ≥ 2 and even e ... b, e := b2, e ÷ 2 { BE = x × be }
        ... and BE = x × be }
  else { BE = x × be }
  fi
  { BE = x × be }
od
{ BE = x × be and e = 0 }
{ x = BE }

```

$e, x := e-1, x \times b \{ B^E = x \times b^e \}$

Purely local reasoning.

CAR Hoare. *An axiomatic basis for computer programming.* Comm. ACM 12(10); 576-580, 583, 1969.

Proof rules for standard loops

then { $e \geq 2$ and even $e \dots$ $b, e := b^2, e \div 2$ { $B^E = x \times b^e$ }
 ... and $B^E = x \times b^e$ }

$$\begin{aligned}
 & B^E = x \times b^e \\
 \bullet & \equiv (B^E = x \times b^e) \langle b, e := b^2, e \div 2 \rangle && wp.(b, e := b^2, e \div 2) \\
 & \equiv B^E = x \times (b^2)^{(e \div 2)} && \text{substitution} \\
 \Leftarrow & B^E = x \times b^e \quad \wedge \quad \text{even } e && \text{arithmetic}
 \end{aligned}$$

EW Dijkstra. *Guarded commands, nondeterminacy, and formal derivation of programs.* Comm. ACM 18(8):453-457, 1975.

... *A Discipline of Programming.* Prentice-Hall, 1976.

Proof rules for standard loops

```

{ pre }
init;
{ inv }
do G →
  { G ∧ inv }
  body
  { inv }
od
{ ¬G ∧ inv }

```

The *loop invariant* makes it unnecessary to reason about “last time” or “next time” or “how many times” in the loop.

No “fence-post problem”...
No “bananas”.

```

{ pre }
init;
{ inv }
do G →
  { G ∧ inv }
  body
  { inv }
od
{ ¬G ∧ inv }

```

Proof rules for standard loops

Proof rules for probabilistic loops

```

{pre}
init;
{inv}
do G →
  {[G] × inv}
  body
  {inv}
od
{[-G] × inv}

{pre}
init;
{inv}
do G →
  {G ∧ inv}
  body
  {inv}
od
{[-G] ∧ inv}
    
```

Proof rules for probabilistic loops: example

```

{?} ←
x := p;
b := true 1/2 ⊕ false;

do b →
  x := 2x;
  if x ≥ 1 then x := x-1 fi;
  b := true 1/2 ⊕ false;
od
{[x ≥ 1/2]}
    
```

What is the probability that x exceeds 1/2 on termination?

Proof rules for probabilistic loops: example

If we assume $0 \leq p \leq 1$, then it's clear that $0 \leq x \leq 1$ is a loop invariant...

...and we can therefore write the assignments to x in the loop body in the more convenient form

```
x := frac.(2x).
```

```

x := p;
b := true 1/2 ⊕ false;
{0 ≤ x ≤ 1}
do b →
  {b ∧ 0 ≤ x ≤ 1}
  x := frac.(2x);
  if x ≥ 1 then x := x-1 fi;
  b := true 1/2 ⊕ false;
  {0 ≤ x ≤ 1}
od
{[x ≥ 1/2]}
    
```

Proof rules for probabilistic loops: example

```

x := p;
b := true 1/2 ⊕ false;
do b →
  x := frac.(2x);
  b := true 1/2 ⊕ false;
  {frac.(2x) < b > int.(2x)}
od
{[x ≥ 1/2]}
    
```

$[x \geq 1/2]$
 $\Leftarrow [-b] \times (\text{frac.}(2x) \leq b \leq \text{int.}(2x))$
 $\text{frac.}(2x) \text{ if } b \text{ else } \text{int.}(2x)$

CAR Hoare. A couple of novelties in the Propositional Calculus. Zeitschr. für Math. Logik und Grundlagen der Math. 31(2):173-178, 1985.

Proof rules for probabilistic loops: example

$$\begin{aligned} \square &\equiv \text{frac.}(2x) \triangleleft b \triangleright \text{int.}(2x) \\ &\equiv (\text{frac.}(2x) + \text{int.}(2x))/2 \\ &\equiv 2x/2 \\ &\equiv x \end{aligned}$$

```

x:= p;
b:= true 1/2 ⊕ false;
do b →
  x:= frac.(2x);
  { x }
  b:= true 1/2 ⊕ false;
  { frac.(2x) < b > int.(2x) }
od
{ [x ≥ 1/2] }
    
```

Proof rules for probabilistic loops: example

Assignment;
loop initialisation;
and then we repeat the earlier step.

```

x:= p;
{ x }
b:= true 1/2 ⊕ false;
{ frac.(2x) < b > int.(2x) }
do b →
  { frac.(2x) }
  x:= frac.(2x);
  { x }
  b:= true 1/2 ⊕ false;
  { frac.(2x) < b > int.(2x) }
od
{ [x ≥ 1/2] }
    
```

Proof rules for probabilistic loops: example

And finally we see that the pre-expectation overall...

is just p .

```

{ p }
x:= p;
{ x }
b:= true 1/2 ⊕ false;
{ frac.(2x) < b > int.(2x) }
do b →
  { frac.(2x) }
  x:= frac.(2x);
  { x }
  b:= true 1/2 ⊕ false;
  { frac.(2x) < b > int.(2x) }
od
{ [x ≥ 1/2] }
    
```

Proof rules for probabilistic loops: example

The probability that the program establishes $x \geq 1/2$ is just p .

The loop invariant was

$$\{ \text{frac.}(2x) \triangleleft b \triangleright \text{int.}(2x) \},$$

“established” by the initialisation and “maintained” by the body.

```

{ p }
x:= p;
{ x }
b:= true 1/2 ⊕ false;
{ frac.(2x) < b > int.(2x) }
do b →
  { frac.(2x) }
  x:= frac.(2x);
  { x }
  b:= true 1/2 ⊕ false;
  { frac.(2x) < b > int.(2x) }
od
{ [x ≥ 1/2] }
    
```

Proof rules for probabilistic loops: *termination*

$\{ inv \}$ do $G \rightarrow$ $\{ [G] \times inv \}$ $body$ $\{ inv \}$ od $\{ [-G] \times inv \}$	$\{ inv \}$ do $G \rightarrow$ $\{ G \wedge inv \}$ $body$ $\{ inv \}$ od $\{ -G \wedge inv \}$
---	---

In addition, show that $inv \Rightarrow term$, where $term$ is the probability of termination ...

... in which case the conclusion $\{ inv \} \mathbf{do} \dots \mathbf{od} \{ [-G] \times inv \}$ expresses total — rather than just partial — correctness.

The $inv \Rightarrow term$ rule: a paradox?

Suppose $term$ is everywhere nonzero — but not necessarily one — and that the loop body is itself terminating.

Now $[true]$ is an invariant for the loop, which is just the everywhere-1 random variable. By *scaling* we therefore have also that p is invariant, for any non-negative constant p .

$\mathbf{do} \ G \rightarrow$
 $\{ [G] \times p \}$
 $body$
 $\{ p \}$
 \mathbf{od}

The $inv \Rightarrow term$ rule: a paradox?

Suppose $term$ is everywhere nonzero — but not necessarily one — and that the loop body is itself terminating.

Now $[true]$ is an invariant for the loop, which is just the everywhere-1 random variable. By *scaling* we therefore have also that p is invariant, for any non-negative constant p .

$\mathbf{do} \ G \rightarrow$
 $\{ [G] \times 1 \}$
 $body$
 $\{ 1 \}$
 \mathbf{od}

The $inv \Rightarrow term$ rule: a paradox?

Suppose $term$ is everywhere nonzero — but not necessarily one — and that the loop body is itself terminating.

Now $[true]$ is an invariant for the loop, which is just the everywhere-1 random variable. By *scaling* we therefore have also that p is invariant, for any non-negative constant p .

Choose nonzero $p \Rightarrow term$, and conclude
 $\{ p \} \mathbf{do} \dots \mathbf{od} \{ [-G] \times p \}$,

$\{ p \}$
 $\mathbf{do} \ G \rightarrow$
 $\{ [G] \times p \}$
 $body$
 $\{ p \}$
 \mathbf{od}
 $\{ [-G] \times p \}$

The $inv \Rightarrow term$ rule: a paradox?

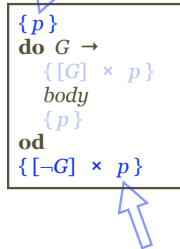
Suppose $term$ is everywhere nonzero — but not necessarily one — and that the loop body is itself terminating.

Now $[true]$ is an invariant for the loop, which is just the everywhere-1 random variable. By *scaling* we therefore have also that p is invariant, for any non-negative constant p .

Choose nonzero $p \Rightarrow term$, and conclude
 $\{p\} \text{ do } \dots \text{ od } \{[-G] \times p\}$,

whence — by scaling back again — we have
 $\{[true]\} \text{ do } \dots \text{ od } \{[-G]\}$.

Thus in fact the loop terminates with probability one everywhere.



The $inv \Rightarrow term$ rule: a paradox?

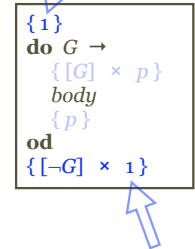
Suppose $term$ is everywhere nonzero — but not necessarily one — and that the loop body is itself terminating.

Now $[true]$ is an invariant for the loop, which is just the everywhere-1 random variable. By *scaling* we therefore have also that p is invariant, for any non-negative constant p .

Choose nonzero $p \Rightarrow term$, and conclude
 $\{p\} \text{ do } \dots \text{ od } \{[-G] \times p\}$,

whence — by scaling back again — we have
 $\{[true]\} \text{ do } \dots \text{ od } \{[-G]\}$.

Thus in fact the loop terminates with probability one everywhere.



The $inv \Rightarrow term$ rule: a paradox?

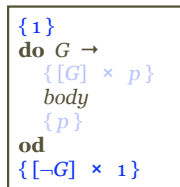
Suppose $term$ is everywhere nonzero — but not necessarily one — and that the loop body is itself terminating.

Now $[true]$ is an invariant for the loop, which is just the everywhere-1 random variable. By *scaling* we therefore have also that p is invariant, for any non-negative constant p .

Choose nonzero $p \Rightarrow term$, and conclude
 $\{p\} \text{ do } \dots \text{ od } \{[-G] \times p\}$,

whence — by scaling back again — we have
 $\{[true]\} \text{ do } \dots \text{ od } \{[-G]\}$.

Thus in fact the loop terminates with probability one everywhere.



The $inv \Rightarrow term$ rule: a paradox?

Suppose $term$ is everywhere nonzero — but not necessarily one — and that the loop body is itself terminating.

Now $[true]$ is an invariant for the loop, which is just the everywhere-1 random variable. By *scaling* we therefore have also that p is invariant, for any non-negative constant p .

Choose nonzero $p \Rightarrow term$, and conclude
 $\{p\} \text{ do } \dots \text{ od } \{[-G] \times p\}$,

whence — by scaling back again — we have
 $\{[true]\} \text{ do } \dots \text{ od } \{[-G]\}$.

Thus in fact the loop terminates with probability one everywhere.

It's not a paradox: it's a zero-one law proved entirely at the level of program logic.

Exercises

Ex. 1: Give an operational argument justifying the zero-one law.

Ex. 2: Find a version of the law that holds even in infinite state-spaces.