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

Proof rules for standard loops

\[ x, b, e := 1, B, E; \]
\[ \text{do } e \neq 0 \rightarrow \]
\[ \quad \text{if even } e \]
\[ \quad \quad b, e := b^2, e+2 \]
\[ \quad \text{else } e, x := e-1, x \times b \]
\[ \quad \text{fi} \]
\[ \text{od} \]

Set \( x \) to \( B^E \) in logarithmic time.

Proof rules for standard loops

\[ \{ B > 0 \text{ and } E \geq 0 \} \]
\[ x, b, e := 1, B, E; \]
\[ \{ b > 0 \text{ and } e \geq 0 \text{ and } B^E = x \times b^e \} \]
\[ \text{do } e \neq 0 \rightarrow \]
\[ \quad \{ \ldots \text{ and } e > 0 \} \]
\[ \quad \text{if even } e \]
\[ \quad \quad b, e := b^2, e+2 \quad \{ B^E = x \times b^e \} \]
\[ \quad \quad \quad \ldots \text{ and } B^E = x \times b^e \}
\[ \quad \text{else } \{ B^E = x \times b^e \} \]
\[ \quad e, x := e-1, x \times b \quad \{ B^E = x \times b^e \} \]
\[ \quad \text{fi} \]
\[ \{ B^E = x \times b^e \} \]
\[ \text{od} \]

\[ \{ B^E = x \times b^e \text{ and } e = 0 \} \]

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


e.g. /dev/random

**Proof rules for standard loops**

\[
\begin{align*}
\{ B > 0 \text{ and } E \geq 0 \} \\
x, b, e := 1, B, E; \\
\{ b > 0 \text{ and } e \geq 0 \text{ and } B^E = x \times b^e \} \\
do e ! 0 ! \\
if \text{ even } e \\
\{ \ldots \text{ and } e > 0 \} \\
then \\
\quad \{ e \geq 2 \text{ and even } e \ldots \} \\
\quad b, e := b^2, e + 2 \quad \{ B^E = x \times b^e \} \\
\quad \ldots \text{ and } B^E = x \times b^e \\
\quad e, x := e - 1, x \times b \quad \{ B^E = x \times b^e \} \\
else \\
\quad \{ B^E = x \times b^e \} \\
f_i \\
\{ B^E = x \times b^e \} \\
\od \\
\{ B^E = x \times b^e \text{ and } e = 0 \} \\
\{ x = B^E \} \\
\end{align*}
\]


---

**Proof rules for standard loops**

\[
\begin{align*}
\{ e \geq 2 \text{ and even } e \ldots \} \\
b, e := b^2, e + 2 \quad \{ B^E = x \times b^e \} \\
\ldots \text{ and } B^E = x \times b^e \\
\end{align*}
\]

\[
\begin{align*}
B^E & = x \times b^e \\
\overset{\text{wp.}}{=} (b^E = x \times b^e \langle b, e := b^2, e + 2 \rangle) \\
& = B^E = x \times (b^2)^{e+2} \\
& = B^E = x \times b^e \land \text{even } e \\
& \quad \text{substitution} \\
& \quad \text{arithmetic}
\end{align*}
\]

**Proof rules for standard loops**

\[
\begin{align*}
\{ \text{pre} \} \\
\text{init;} \\
\{ \text{inv} \} \\
do G \rightarrow \\
\quad \{ G \land \text{[inv]} \} \\
\quad \text{body} \\
\quad \{ \text{inv} \} \\
\od \\
\{ \neg G \land \text{[inv]} \}
\end{align*}
\]

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 "banananas".


... *A Discipline of Programming.* Prentice-Hall, 1976.
Proof rules for probabilistic loops

\{ \text{pre} \} \quad \text{init}; \quad \{ \text{inv} \}

\textbf{do}\ G! \quad \{ G \times \text{inv} \}

\text{body}

\{ \text{inv} \}

\textbf{od}

\{ \neg G \times \text{inv} \}

\begin{align*}
x &:= p; \\
b &:= \text{true}_{1/2} \oplus \text{false};
\end{align*}

\textbf{do}\ b\rightarrow

\begin{align*}
\{ b \land 0 \leq x \leq 1 \} \\
x &:= 2x; \\
\text{if } x \geq 1 \text{ then } x &:= x-1 \text{ fi;}
\end{align*}

\begin{align*}
b &:= \text{true}_{1/2} \oplus \text{false};
\end{align*}

\textbf{od}

\{ \{ x \geq 1/2 \} \}

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

\begin{align*}
x &:= \frac{x}{2}.
\end{align*}

Proof rules for probabilistic loops: example

\{ \text{?} \}

\begin{align*}
x &:= p; \\
b &:= \text{true}_{1/2} \oplus \text{false};
\end{align*}

\textbf{do}\ b\rightarrow

\begin{align*}
\{ b \land 0 \leq x \leq 1 \} \\
x &:= 2x; \\
\text{if } x \geq 1 \text{ then } x &:= x-1 \text{ fi;}
\end{align*}

\begin{align*}
b &:= \text{true}_{1/2} \oplus \text{false};
\end{align*}

\textbf{od}

\{ \{ x \geq 1/2 \} \}

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

\begin{align*}
x &:= p; \\
b &:= \text{true}_{1/2} \oplus \text{false};
\end{align*}

\textbf{do}\ b\rightarrow

\begin{align*}
\{ b \land 0 \leq x \leq 1 \} \\
x &:= \frac{x}{2}; \\
\text{if } b \text{ then } x &:= \text{int}(x) \text{ fi;}
\end{align*}

\begin{align*}
b &:= \text{true}_{1/2} \oplus \text{false};
\end{align*}

\textbf{od}

\{ \{ x \geq 1/2 \} \}

\begin{align*}
\{ x \geq 1/2 \} &\leftrightarrow \{ \neg b \times ( \frac{x}{2} \oplus b \geq \text{int}(2x) ) \} \\
&\leftrightarrow \{ \frac{x}{2} \oplus b \geq \text{int}(2x) \} \\
&\leftrightarrow \{ \frac{x}{2} \oplus b \geq \text{int}(2x) \}
\end{align*}

Proof rules for probabilistic loops: example

\[
\begin{align*}
\text{Proof rules for probabilistic loops: example} \\
\frac{\text{x:= p;}}{&} \\
\text{b:= true 1/2} \oplus \text{false;} \\
\text{do } b \rightarrow \\
\text{x:= frac.(2x);} \\
\text{\{} x \text{\} } \\
\text{b:= true 1/2} \oplus \text{false;} \\
\text{\{} frac.(2x) < b \triangleright \text{int.(2x)} \text{\} } \\
\text{od} \\
\text{\{} x \geq 1/2 \text{\} }
\end{align*}
\]

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

is just \(p\).

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

The loop invariant was 
\(\{ \text{frac.(2x) < b \triangleright \text{int.(2x)}} \} \), 
"established" by the initialisation and "maintained" by the body.
Proof rules for probabilistic loops: termination

\begin{align*}
\{ \text{inv} \} & \quad \{ \text{inv} \} \\
do \ G \rightarrow & \quad \do \ G \rightarrow \\
\{ \lbrack G \rbrack \times \text{inv} \} & \quad \{ G \land \text{inv} \} \\
\text{body} & \quad \text{body} \\
\{ \text{inv} \} & \quad \{ \text{inv} \} \\
\od & \quad \od \\
\{ \lbrack -G \rbrack \times \text{inv} \} & \quad \{ -G \land \text{inv} \} \\
\end{align*}

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

... in which case the conclusion \{ \text{inv} \} \od \ldots \od \{ \lbrack -G \rbrack \times \text{inv} \}
expresses total — rather than just partial — correctness.

The inv \Rightarrow term rule: a paradox?

Suppose \text{term} is everywhere nonzero — but not necessarily one — and that the loop body is itself terminating.

Now \{\text{true}\} is an invariant for the loop, which is just the everywhere-1 random variable. By scaling we therefore have also that \text{p is invariant}, for any non-negative constant \text{p}.

Choose nonzero \text{p} \Rightarrow \text{term}, and conclude \{ \text{p} \} \od \ldots \od \{ \lbrack -G \rbrack \times \text{p} \}.
Suppose \( \text{term} \) is everywhere nonzero — but not necessarily one — and that the loop body is itself terminating.

Now \([\text{true}]\) 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 \( p \Rightarrow \text{term} \), and conclude
\[
\{ p \} \text{ do } \cdots \text{ od } \{ [-G] \times p \},
\]
whence — by scaling back again — we have
\[
\{ [\text{true}] \} \text{ do } \cdots \text{ od } \{ [-G] \}.
\]
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.