Formal Methods for Probabilistic Systems

Annabelle McIver
Carroll Morgan

- Probabilistic temporal logic: qTL
- Probabilistic sequential-programming logic: pGCL
  - Origins of (this) program logic
  - Syntax and semantics of pGCL
  - Geometric interpretation (informal)
  - Metatheorems (for iteration)

Program logic

- What kind of Logic is it?
- Where did it come from?
- How does it fit in?

Inspired...

Floyd static annotations of flowchart programs

Program logic

- **Hoare logic of sequential programs**: \( \{ \text{pre} \} \text{prog} \{ \text{post} \} \)
- **Floyd static annotations of flowchart programs**
- **Relational model**: \( S \rightarrow S \)

Inspired by...
- **Floyd static annotations of flowchart programs**
- **Hoare logic of sequential programs**: \( \{ \text{pre} \} \text{prog} \{ \text{post} \} \)

C.A.R. Hoare.
*An axiomatic basis for computer programming.*
Comm. A.C.M. 12(10), 1969

Program logic

- **Dijkstra logic of weakest preconditions**: \( \text{pre} \Rightarrow \text{wp}(\text{prog}, \text{post}) \)
- **Transformer model**: \( \text{PS} \rightarrow \text{PS} \) (conjunctive)
- **Relational model**: \( S \rightarrow \text{PS} \)

E.W. Dijkstra.
*A Discipline of Programming.*
Prentice Hall, 1976
Program logic
Has a Galois connection between...
relational model: $S \rightarrow \mathbb{P}S$
(transform conjunctive)

Dijkstra logic of weakest preconditions:
$\text{pre} \rightarrow \text{wp}(\text{post}, \text{prog})$

Hoare logic of sequential programs:
$\{\text{pre}\} \text{prog} \{\text{post}\}$

Generalises to...
add non-determinism

Dijkstra logic of weakest preconditions:
$\text{pre} \rightarrow \text{wp}(\text{post}, \text{prog})$

Hoare logic of sequential programs:
$\{\text{pre}\} \text{prog} \{\text{post}\}$

Is modelled by...

relational model:
$S \rightarrow \mathbb{P}S$

transformer model:
$\mathbb{P}S \rightarrow \mathbb{P}S$

Kozen logic of probabilistic programs:
$\{\text{preE}\} \text{prog} \{\text{postE}\}$

D. Kozen.
Semantics of probabilistic programs.
Journal of Computer and System Sciences, 1981
A probabilistic PDL. Proc. 15th STOC, ACM, 1983

relational model:
$S \rightarrow \text{dist.S}$

transformer model:
$\text{fun.S} \rightarrow \text{fun.S}$
(linear)

relational model:
$S \rightarrow \text{dist.S}$

transformer model:
$\text{fun.S} \rightarrow \text{fun.S}$
(linear)
1983–96...


What is the probability that the program
\( coin:= \text{heads} \frac{1}{2} \# \text{tails} \)
establishes the postcondition \( coin = \text{heads} \)?

Probabilistic choice: \( \frac{1}{2} \) left; \( \frac{1}{2} \) right.

We can abbreviate “\( coin:= \text{heads} \frac{1}{2} \# \text{tails} \)” as just
\( coin:= \text{heads} \frac{1}{2} \# \text{tails} \)
because the left-hand sides “\( coin:= \)” are the same.

In the program logic we write
\[ \text{wp}(\text{coin:= heads} \frac{1}{2} \# \text{tails}), \text{coin = heads}] = \frac{1}{2} \]
to say that the probability is just \( \frac{1}{2} \).

We will look at these in turn: what we need to know for each type of program fragment \( prog \) is

What is \( \text{wp}.prog.B \) for arbitrary postcondition \( B \)?

The usual technique for setting this out is \textit{structurally} over the syntax of the programming language.

1. Assignment statements;
2. Probabilistic choice;
3. Conditionals;
4. Sequential composition;
5. Demonic choice.

— written in \textit{pGCL}.

**Interpretation: assignments**

\[ x := E \quad \text{Assign the value of expression } E \]  
\text{to the variable } x. 

---

\[ \text{wp.}(x := E).B = B[x := E] \]

Definition.

- **Syntactic substitution.**
- **Definition.**
- **Substitution.**
- **Arithmetic.**

**Example.**

\[ \text{wp.}(x := x + 1), [x = 3] \]

\[ x = 3 \]

\[ x := x + 1 \]

\[ \text{definition} \]

\[ [x = 3] \]

\[ \text{substitution} \]

\[ [x + 1 = 3] \]

\[ \text{arithmetic} \]

**Why are these here?**

---

**Interpretation: probabilistic choice**

\[ \text{if } G \text{ then } \text{prog}_1 \text{ fi} \]

Execute the left-hand side with probability \( p \), otherwise execute the right-hand side (probability \( 1 - p \)).

---

\[ \text{wp.}(\text{prog}_1 \ p \ @ \ \text{prog}_2), B = p \times \text{wp.}\text{prog}_1, B + (1 - p) \times \text{wp.}\text{prog}_2, B \]

---

\[ \text{wp.}(c := H_{1/2} @ T), [c = H] \]

\[ 1/2 \times \text{wp.}(c := H), [c = H] \]

\[ 1/2 \times \text{wp.}(c := T), [c = H] \]

\[ 1/2 \times [H = H] + 1/2 \times [T = H] \]

\[ 1/2 \times 1 + 1/2 \times 0 \]

\[ 1/2. \]

**Interpretation: deterministic choice**

\[ \text{if } G \text{ then } \text{prog}_1 \text{ else } \text{skip} \text{ fi} \]

If guard \( G \) holds, then execute \( \text{prog}_1 \); otherwise do nothing.

---

\[ \text{skip} \]

Do nothing.

---

\[ \text{if } G \text{ then } \text{prog}_1 \text{ else } \text{prog}_2 \text{ fi} \]

If guard \( G \) holds, then execute \( \text{prog}_1 \); otherwise execute \( \text{prog}_2 \).

---

\[ \text{if } G \text{ then } \text{prog}_1 \text{ else } \text{prog}_2 \text{ fi} \]

If \( G \) holds, then go left with probability \( 1 \), and vice versa.

\[ \text{wp.}(x := x+1), [x = 3] \]

\[ [x = 3] \]

\[ x := x + 1 \]

**Definition.**

**Substitution.**

**Arithmetic.**

The probability that \( x := x + 1 \) achieves \( x = 3 \) is one if \( x = 2 \) initially, and zero otherwise.

Thus \( \boxed{\text{•}} \) must be an embedding function that takes true to one and false to zero.

**Interpretation: embedding Booleans**

\[ \text{wp.}(x := x+1), [x = 3] \]

\[ x := x + 1 \]

**Definition.**

**Substitution.**

**Arithmetic.**

**Informal description.**

Why are these here?

**Syntactic substitution.**

**Interpretation: probabilistic choice**

**Interpretation: deterministic choice**
Interpretation: deterministic choice

\[
\text{wp} \cdot (\text{if } x \geq 1 \text{ then } x := x - 1 \text{ else } x := x + 2) \cdot [x \geq 2]
\]

\[
= \text{wp} \cdot (x := x - 1 \cdot [x \geq 1]) \times \text{wp} \cdot (x := x + 2) \cdot [x \geq 2]
\]

For a standard conditional, the reasoning is just “as usual”.

Interpretation: sequential composition

\[
\text{prog}_1; \text{prog}_2 \quad \text{Execute the first program; then execute the second.}
\]

\[
\text{wp} \cdot (\text{prog}_1; \text{prog}_2) \cdot B = \text{wp} \cdot \text{prog}_1 \cdot (\text{wp} \cdot \text{prog}_2 \cdot B)
\]

The expected value of the function \(1/2 \times [c=H] + 1/2 \times [c=T]\) over the distribution of states produced by the program is \(1/2\).

As a special case (from elementary probability theory) we know that the expected value of the function \([\text{pred}]\), for some Boolean \text{pred}, is just the probability that \text{pred} holds.

That’s why \text{wp} \cdot \text{prog} \cdot [\text{pred}] gives the probability that \text{pred} is achieved by \text{prog}. But, as we see above, we can be much more general if we wish.
Expected value of $[\text{pred}]$ is thus
\[ p \times 1 + (1-p) \times 0, \]
that is, just $p$ itself.

We note that the standard logic can be embedded in the probabilistic logic simply by converting all Booleans $\text{false}$, $\text{true}$ to the integers 0, 1 (a technique familiar to C programmers). The probabilistic $wp$-logic (greatest pre-expectations) extends the standard $wp$-logic (weakest preconditions) conservatively in this sense:

If we restrict ourselves to standard programs (i.e. do not use the probabilistic choice operator), then the theorems for those programs are exactly the same as before.

Mathematically this is expressed as follows:

For all standard programs $\text{prog}$, and Boolean postconditions $\text{post}$, we have
\[ wp.(\text{prog} \land \text{post}) \equiv wp.\text{prog}.\text{post} \]
where on the left the $wp$ is weakest precondition, and on the right it is greatest pre-expectation.

Interpretation: demonic choice

\begin{align*}
\text{wp.} \text{abort}. \text{post} & := 0 \\
\text{wp.} \text{skip}. \text{post} & := \text{post} \\
\text{wp.}(x := \text{expr}). \text{post} & := \text{post} \{ x \mapsto \text{expr} \} \\
\text{wp.}(\text{prog}; \text{prog'}). \text{post} & := \text{wp.} \text{prog}. \text{wp.} \text{prog}'. \text{post} \\
\text{wp.}(\text{prog} \lor \text{prog'}). \text{post} & := \text{wp.} \text{prog}. \text{post} \min \text{wp.} \text{prog}'. \text{post} \\
\text{wp.}(\text{prog} \land \text{prog'}). \text{post} & := p \times \text{wp.} \text{post} + \bar{p} \times \text{wp.} \text{prog}'. \text{post}
\end{align*}

Recall that $\bar{p}$ is the complement of $p$.

The expression on the right gives the greatest pre-expectation of $\text{post}E$ with respect to each $\mu GCL$ construct, where $\text{post}E$ is an expression of type $\mathbb{S}$ over the variables in state space $S$. (For historical reasons we continue to write $wp$ instead of $wp$.)

In the case of recursion, however, we cannot give a purely syntactic definition. Instead we say that
\[ (\mu x. x \cdot C) := \text{least fixed-point of the function } \text{cute.} \mathbb{S} \Rightarrow \mathbb{S} \]
defined so that $\text{cute}(wp.x).x = wp.C.$

**Figure 1.5.3. Probabilistic $wp$-semantics of $\mu GCL$**

Although the program might achieve $c=H$, the largest probability of that which can be guaranteed... is zero.
An unbiased coin.


Interpretation: a geometric view

A heads-biased coin.

Demonic choice between these

A biased coin, up to 1/6 either way...

... one refinement of which is an unbiased coin.
A possibly nonterminating coin... whose refinements include all three coins before.

Demonically, either of two possibly nonterminating coins.

Demonically, either of two possibly nonterminating coins.

\[(1/4, 1/3)\]
\[(1/3, 1/4)\]

What is the greatest guaranteed expected value of this program with respect to the post-expectation \[2[c=H] + [c=T] \] ?

It's not \(11/12\).
What is the greatest guaranteed expected value of this program with respect to the post-expectation $2[c=H] + [c=T]$?

It's $\frac{5}{6}$ again, because this time the demon goes to the other extreme.

$2x + y = \frac{2}{4} + \frac{1}{3} = \frac{5}{6}$

$2x + y = 0$

Metatheorems for iteration

// Implement $p^#$ using unbiased random bits only.

```
x := p;
b := true 1/2# false;
do   b  !
x := 2x - [x \geq \frac{1}{2}];
b := true 1/2# false;
od;
if  x  \geq \frac{1}{2} then prog_1  else prog_2
fi
```

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


Metatheorems for iteration

Metatheorems for iteration

begin
var x,b;
x := p;
b := true 1/2# false;
do  b  
x := 2x - [x \geq \frac{1}{2}];
b := true 1/2# false;
od;
if  x \geq \frac{1}{2} then prog_1  else prog_2
fi
end
Iteration: reminder of standard metatheorems

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

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

Standard metatheorems: invariants

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

Probabilistic metatheorems: invariants again

\[ \{ \text{pre} \} \]
\[ \text{init}; \]
\[ \{ \text{inv} \} \]
\[ \textbf{do} \ G \rightarrow \]
\[ \quad \{ G \times \text{inv} \} \]
\[ \quad \text{body} \]
\[ \quad \{ \text{inv} \} \]
\[ \textbf{od} \]
\[ \{ \neg G \times \text{inv} \} \]

\[ \{ \text{pre} \} \]
\[ \text{init}; \]
\[ \{ \text{inv} \} \]
\[ \textbf{do} \ G \rightarrow \]
\[ \quad \{ [G] \times \text{inv} \} \]
\[ \quad \text{body} \]
\[ \quad \{ \text{inv} \} \]
\[ \textbf{od} \]
\[ \{ [\neg G] \times \text{inv} \} \]

\[ \quad \text{body} \]
\[ \quad \{ \text{inv} \} \]
\[ \textbf{od} \]
\[ \{ \neg G \times \text{inv} \} \]
Iteration: probabilistic example

\[
\begin{align*}
\{ ? \} \\
\text{x} &:= p; \\
\text{b} &:= \text{true} \oplus \text{false}; \\
\text{do} & \ b \rightarrow \\
\text{x} &:= 2x - [x \geq 1/2]; \\
\text{b} &:= \text{true} \oplus \text{false}; \\
\text{od}
\end{align*}
\]

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

Example: iteration achieves its goal on termination

\[
\begin{align*}
x &:= p; \\
\text{b} &:= \text{true} \oplus \text{false}; \\
\text{do} & \ b \rightarrow \\
\left(x \geq 1/2\right) \\
\left(2x - [x \geq 1/2] \land \text{b} \lor [x \geq 1/2]\right)
\end{align*}
\]

... if \( \text{b} \) else ...

Example: iteration body preserves invariant

\[
\begin{align*}
\text{x} &:= p; \\
\text{b} &:= \text{true} \oplus \text{false}; \\
\text{do} & \ b \rightarrow \\
\left(x \geq 1/2\right) \\
\left(2x - [x \geq 1/2] \land \text{b} \lor [x \geq 1/2]\right)
\end{align*}
\]

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

Example: iteration properly initialised

\[
\begin{align*}
\text{x} &:= p; \\
\{ x \} \\
\text{b} &:= \text{true} \oplus \text{false}; \\
\{ 2x - [x \geq 1/2] \land \text{b} \lor [x \geq 1/2]\} \\
\text{do} & \ b \rightarrow \\
\{ x \geq 1/2\}
\end{align*}
\]
\[ x := p; \]
\[ b := \text{true} \] \( \frac{1}{2} \) \# \text{false};
\[ \text{do} \quad b \not\Rightarrow x := 2x - [x \text{ } \frac{1}{2}]; \]
\[ b := \text{true} \] \( \frac{1}{2} \) \# \text{false};
\[ \text{od} \]
\{ \{x \} \}
\{ \{x \} \}
\{ ... \}
\{ 2x - [x \text{ } \frac{1}{2}] \}
\{ ... \}
\{ 2x - [x \text{ } \frac{1}{2}] \}
\{ 2x - [x \text{ } \frac{1}{2}] \}

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

is just \( p \).

The loop invariant was

\[ 2x - [x \geq \frac{1}{2}] \not\Rightarrow [x \geq \frac{1}{2}] \]

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

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

... in which case the conclusion \( \{ \text{inv} \} \text{do} \cdots \text{od} \{ [-G] \times \text{inv} \} \)

expresses total — rather than just partial — correctness.

\[ \{ p \} \quad \{ \{x \} \} \]
\[ x := p; \]
\[ \{ \{x \} \} \]
\[ b := \text{true} \] \( \frac{1}{2} \not\Rightarrow \text{false}; \]
\[ \text{do} \quad b \not\Rightarrow x := 2x - [x \text{ } \frac{1}{2}]; \]
\[ b := \text{true} \] \( \frac{1}{2} \) \# \text{false};
\[ \text{od} \]
\{ \{x \geq \frac{1}{2} \} \}

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

The answers you get to Ex. 1 and Ex. 2 should differ.

Explain “in layman’s terms” why they do.

(Hint: Imagine an experiment with two people and two coins, in each case.)
Ex. 4: The nature of demonic choice

It is sometimes suggested that demonic choice can be regarded as an arbitrary but unpredictable probabilistic choice; this would simplify matters because there would then only be one kind of choice to deal with.

Use our logic to investigate this suggestion; in particular, look at the behaviour of

\[ c := H_{\frac{1}{2}} T; \quad d := H_{p} T \]

for arbitrary \( p \),

and compare it with the program of Ex. 1. Explain your conclusions in layman’s terms.

Ex. 5: Compositionality

Consider the two programs

\[ A: \quad \text{coin} := \text{edge} \land \left( \text{coin} := \text{heads} \frac{1}{2} \uplus \text{coin} := \text{tails} \right) \]
\[ B: \quad \left( \text{coin} := \text{edge} \land \text{coin} := \text{heads} \right) \frac{1}{2} \uplus \left( \text{coin} := \text{edge} \land \text{coin} := \text{tails} \right) , \]

which we will call \( A \) and \( B \). Say that they are similar because from any initial state they have the same worst-case probability of achieving any given postcondition. (This can be shown by tabulation: there are only eight possible postconditions.)

Find a program \( C \) such that \( A; C \) and \( B; C \) are not similar, even though \( A \) and \( B \) are. (Use the wp-definition of \( ; \).) Why is this a problem?

More generally, let \( A \) and \( B \) be any two programs that are not equal in our \( \text{wp} \) logic. Show that there is always a program \( C \) as above, i.e. such that \( A; C \) and \( B; C \) are not similar. What does that tell you about our quantitative logic in terms of its possibly being a “minimal complication”?