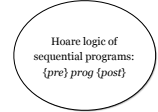


# Formal Methods for Probabilistic Systems

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)

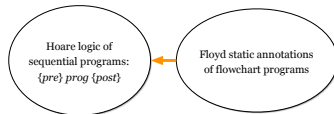
## Program logic



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

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

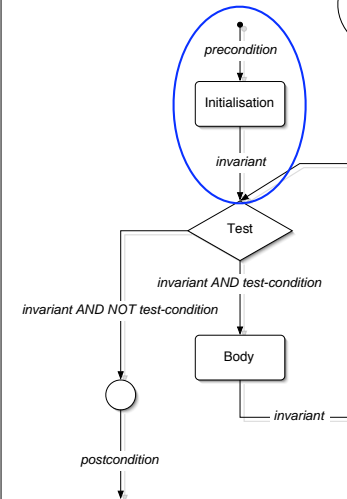
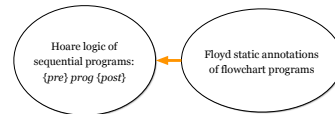
## Program logic



Inspired...

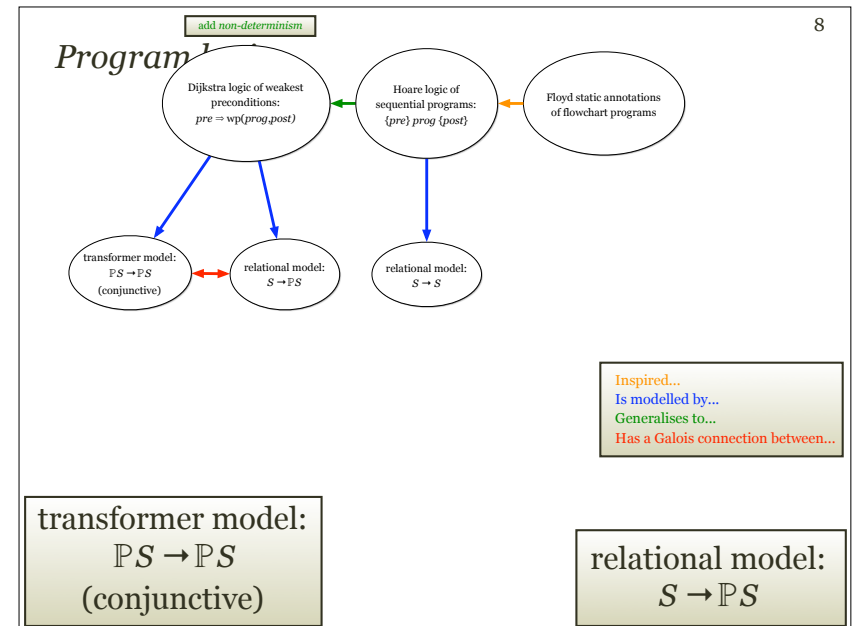
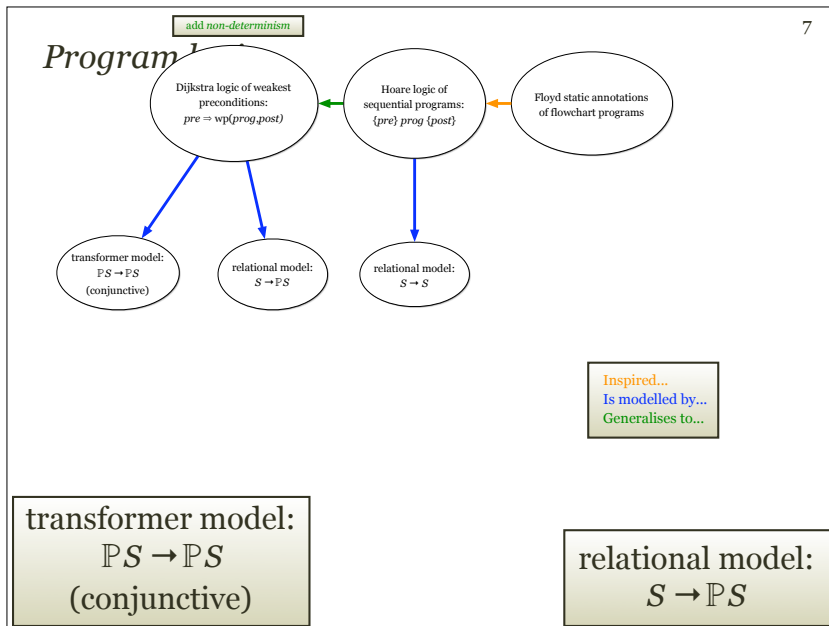
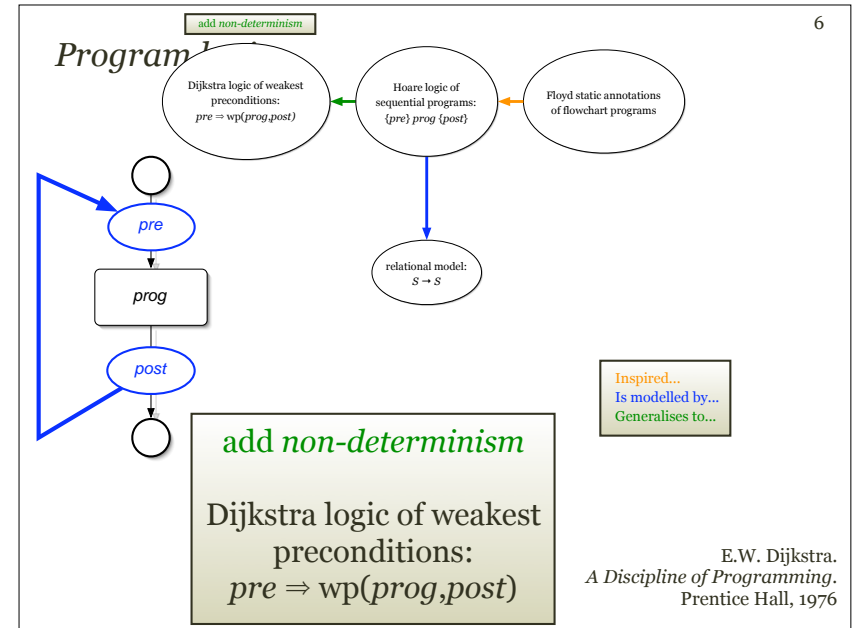
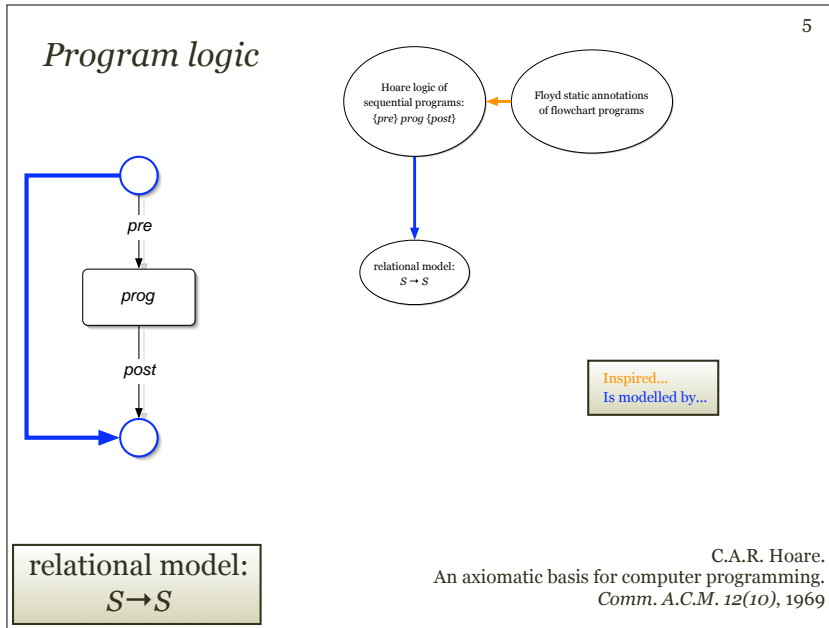
Floyd static annotations of flowchart programs

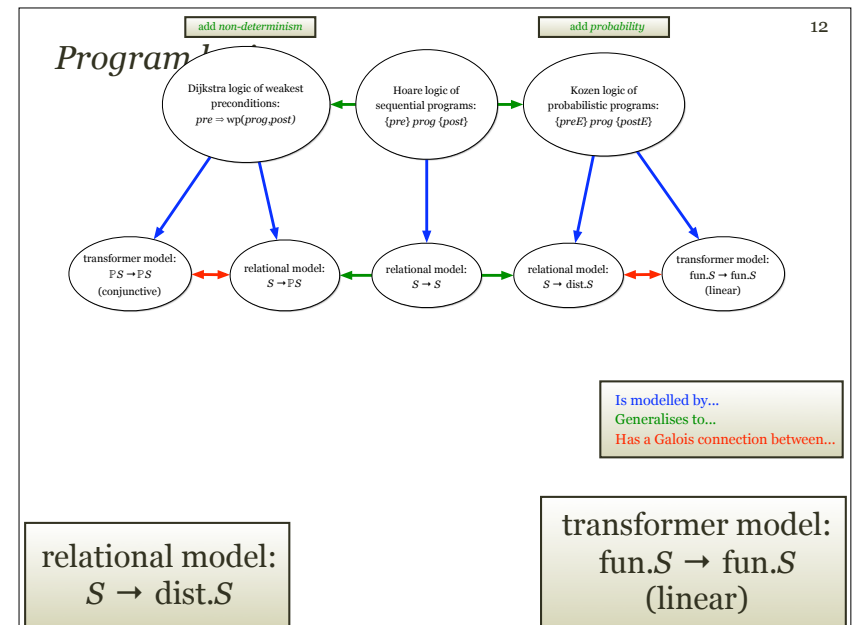
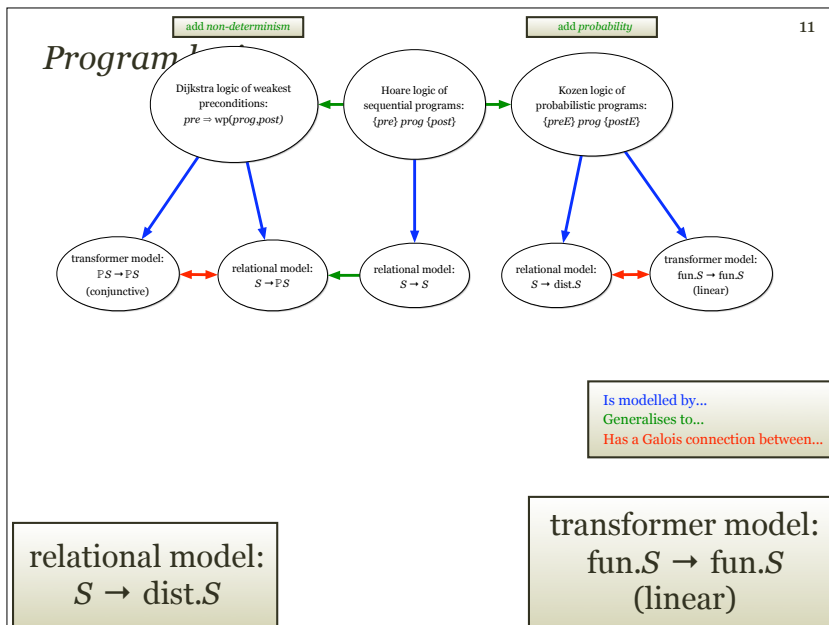
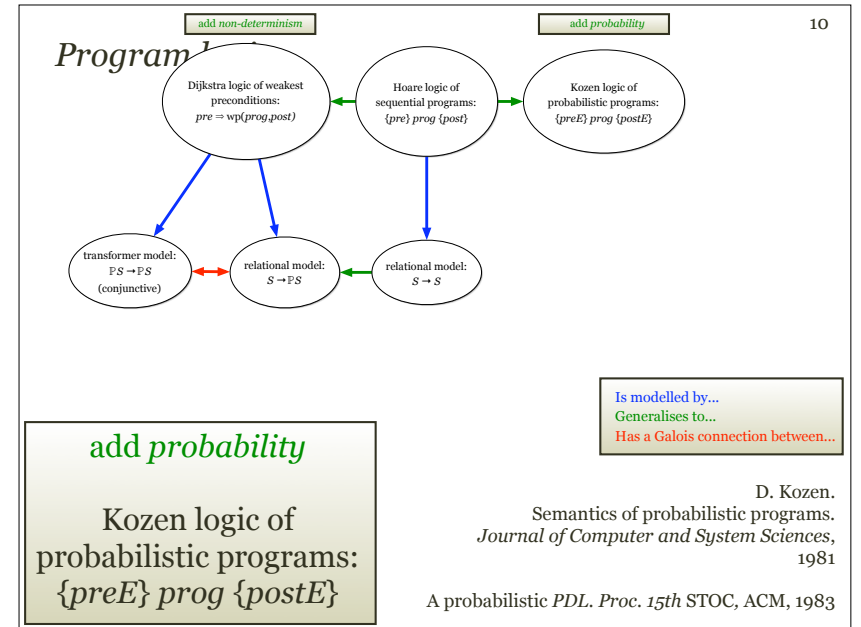
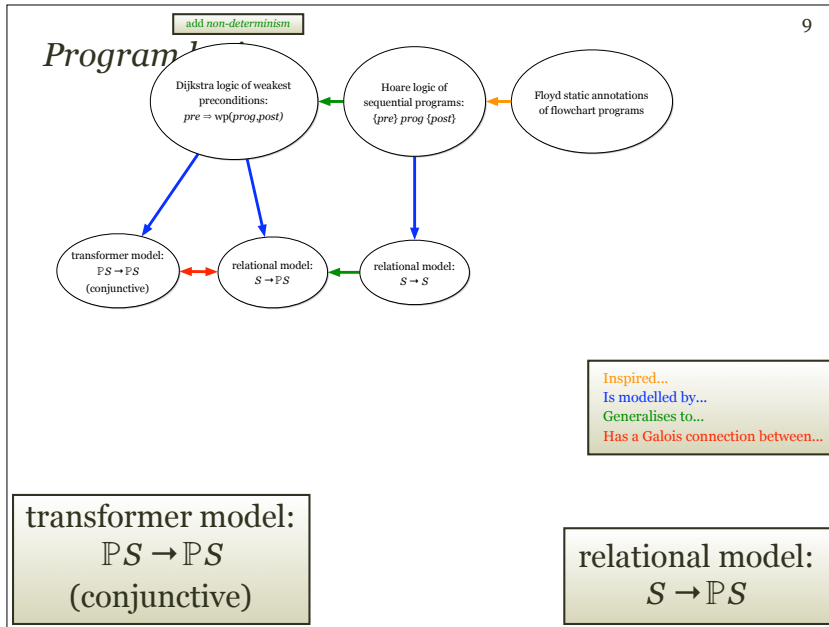
## Program logic



Inspired...

R.W. Floyd. Assigning meanings to programs.  
*Mathematical Aspects of Computer Science*,  
J.T. Schwartz (ed.)  
American Mathematical Society, 1967





1983–96...

Is modelled by...  
Generalises to...  
Has a Galois connection between...

C Jones. *Probabilistic nondeterminism*.  
Monograph ECS-LFCS-90-105 (PhD Thesis),  
University of Edinburgh, 1989.

C Jones and G Plotkin. *A probabilistic powerdomain of evaluations*.  
Proc. 4th IEEE LICS Symp., 168-195, 1989.

Combined logic of weakest  
pre-expectations:  
 $preE \Rightarrow wp(prog, postE)$

add non-determinism and  
probability

Is modelled by...  
Generalises to...  
Has a Galois connection between...

J. He, A. McIver, K. Seidel  
Probabilistic models for the  
guarded command language  
*Science of Computer Programming* 28,  
1997

Combined logic of weakest  
pre-expectations:  
 $preE \Rightarrow wp(prog, postE)$

C.C. Morgan, A. McIver, K. Seidel  
Probabilistic predicate transformers  
*ACM TOPLAS* 18(3)  
1996

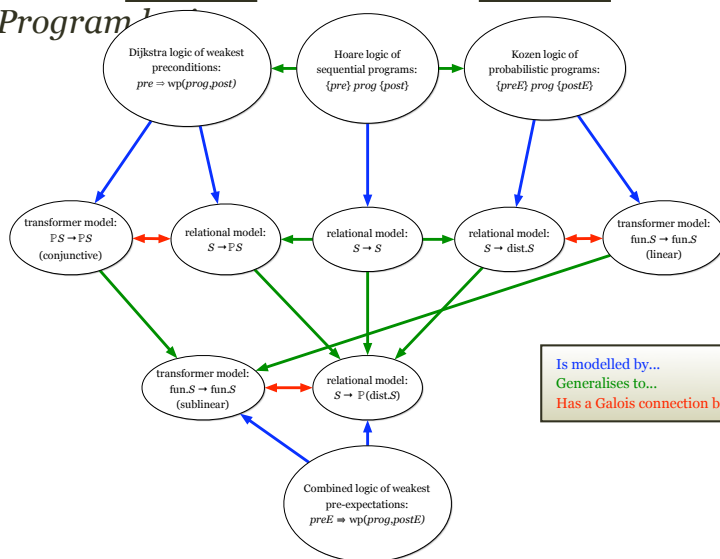
add non-determinism and probability

Program

add non-determinism

add probability

15



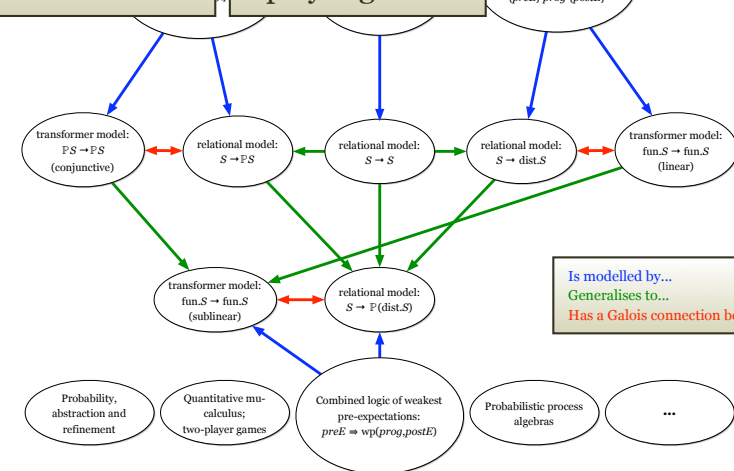
add non-determinism and probability

Probability,  
abstraction and  
refinement

Quantitative mu-  
calculus; two-  
player games

Probabilistic process  
algebras

probabilistic programs:  
{preE} prog {postE}



add non-determinism and probability

## Probabilistic-program logic: introduction

17

What is the probability that the program

$coin := heads_{1/2} \oplus tails$

establishes the postcondition  $coin = heads$ ?

Probabilistic choice:  $1/2$  left;  $(1-1/2)$  right.

We can abbreviate “ $coin := heads_{1/2} \oplus coin := tails$ ” as just

$coin := heads_{1/2} \oplus tails$

because the left-hand sides “ $coin :=$ ” are the same.

## Probabilistic-program logic: introduction

18

What is the probability that the program

$coin := heads_{1/2} \oplus tails$

establishes the postcondition  $coin = heads$ ?

In the program logic we write

$wp.(coin := heads_{1/2} \oplus tails).[coin = heads] \equiv 1/2$

to say that the probability is just  $1/2$ .

CC Morgan, AK McIver and K Seidel. *Probabilistic predicate transformers*.  
 TOPLAS 18(3):325-353, 1996.

D Kozen. *A probabilistic PDL*. J. Comp. & Sys. Sci. 30(2):162-178, 1985.

## Probabilistic-program logic: introduction

19

program (fragment)

non-negative real-valued  
 expression over program  
 variables

$wp.(coin := heads_{1/2} \oplus tails).[coin = heads] \equiv 1/2$

## Interpretation

20

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

— written in *pGCL*.

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

What is  $wp.prog.B$  for arbitrary postcondition  $B$ ?

The usual technique for setting this out is *structurally* over the syntax of the programming language.

CC Morgan, AK McIver. *pGCL: Formal reasoning for random algorithms*.  
 SACJ 22, 1999.



### Interpretation: assignments

$x := E$  Assign the value of expression  $E$  to the variable  $x$ . Informal description.

$wp.(x := E).B \equiv B[x := E]$  Definition.

Syntactic substitution.

Example.  
 $wp.(x := x+1).[x=3]$   
 $\equiv [x=3] \langle x := x+1 \rangle$  definition  
 $\equiv [(x+1)=3]$  substitution  
 $\equiv [x=2]$  arithmetic

Why are these here?

### Interpretation: embedding Booleans

$wp.(x := x+1).[x=3]$   
 $\equiv [x=3] \langle x := x+1 \rangle$  definition  
 $\equiv [(x+1)=3]$  substitution  
 $\equiv [x=2]$  arithmetic

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

Thus "[ $\cdot$ ]" must be an *embedding function* that takes *true* to one and *false* to zero.

### Interpretation: probabilistic choice

$prog_1 \text{ } p \oplus \text{ } prog_2$  Execute the left-hand side with probability  $p$ , otherwise execute the right-hand side (probability  $1-p$ ).

$wp.(prog_1 \text{ } p \oplus \text{ } prog_2).B \equiv p \times wp.prog_1.B + (1-p) \times wp.prog_2.B$

Example.  
 $wp.(c := H_{1/2} \oplus T).[c=H]$   
 $\equiv 1/2 \times wp.(c := H).[c=H] + (1-1/2) \times wp.(c := T).[c=H]$  definition  
 $\equiv 1/2 \times [H=H] + 1/2 \times [T=H]$  assignment  
 $\equiv 1/2 \times 1 + 1/2 \times 0$  embedding  
 $\equiv 1/2$  arithmetic

### Interpretation: deterministic choice

**if  $G$  then  $prog$  fi** If guard  $G$  holds, then execute the body  $prog$ ; otherwise do nothing.

**if  $G$  then  $prog$  else skip fi**  
 skip Do nothing.  $x := x$

**if  $G$  then  $prog_1$  else  $prog_2$  fi**  
 If guard  $G$  holds, then execute  $prog_1$ ; otherwise execute  $prog_2$ .  
 If  $G$  holds, then go left with probability 1, and vice versa.  $prog_1 [G] \oplus prog_2$

### Interpretation: deterministic choice

$$\begin{aligned}
 & wp.(\text{if } x \geq 1 \text{ then } x := x - 1 \text{ else } x := x + 2 \text{ fi}).[x \geq 2] \\
 \equiv & wp.(x := x - 1 \ [x \geq 1]^\oplus \ x := x + 2).[x \geq 2] && \text{"sugar"} \\
 \equiv & [x \geq 1] \times wp.(x := x - 1).[x \geq 2] && \text{prob. choice} \\
 & + [1 - [x \geq 1]] \times wp.(x := x + 2).[x \geq 2] \\
 \equiv & [x \geq 1] \times [(x-1) \geq 2] + [x < 1] \times [(x+2) \geq 2] && \text{assignment} \\
 \equiv & [x \geq 1] \triangleleft [x \geq 3] \oplus [x < 1] \triangleleft [x \geq 0] && \text{arithmetic} \\
 \equiv & [x \geq 1] \triangleleft x \geq 3 \oplus x < 1 \triangleleft x \geq 0 && \text{embedding} \\
 \equiv & [x \geq 3 \vee 0 \leq x < 1]. && \text{logic}
 \end{aligned}$$

For a *standard* conditional, the reasoning is just "as usual".

### Interpretation: sequential composition

$prog_1 ; prog_2$       Execute the first program;  
then execute the second.

---


$$wp.(prog_1 ; prog_2).B \equiv wp.prog_1.(wp.prog_2.B)$$


---


$$\begin{aligned}
 & wp.(c := H_{1/2} \oplus T ; d := H_{1/2} \oplus T).[c=d] \\
 \equiv & wp.(c := H_{1/2} \oplus T). (wp.(d := H_{1/2} \oplus T).[c=d]) && \text{definition} \\
 \equiv & wp.(c := H_{1/2} \oplus T). ( && \text{prob. choice; assignment} \\
 & \quad \frac{1}{2} \times [c=H] + \frac{1}{2} \times [c=T] && \\
 & ) && \\
 \equiv & \frac{1}{2} \times (\frac{1}{2} \times [H=H] + \frac{1}{2} \times [H=T]) && \text{prob. choice; assignment} \\
 & + \frac{1}{2} \times (\frac{1}{2} \times [T=H] + \frac{1}{2} \times [T=T]) \\
 \equiv & \frac{1}{4} + \frac{1}{4} && \text{embedding} \\
 \equiv & \frac{1}{2}. && \text{arithmetic}
 \end{aligned}$$

### Interpretation: sequential composition

$prog_1 ; prog_2$       Execute the first program;  
then execute the second.

---


$$wp.(prog_1 ; prog_2).B \equiv wp.prog_1.(wp.prog_2.B)$$


---


$$\begin{aligned}
 & wp.(c := H_{1/2} \oplus T ; d := H_{1/2} \oplus T).[c=d] \\
 \equiv & wp.(c := H_{1/2} \oplus T). (wp.(d := H_{1/2} \oplus T).[c=d]) && \text{definition} \\
 \equiv & wp.(c := H_{1/2} \oplus T). ( && \text{prob. choice; assignment} \\
 & \quad \frac{1}{2} \times [c=H] + \frac{1}{2} \times [c=T] && \\
 & ) && \\
 \equiv & \frac{1}{2} \times (\frac{1}{2} \times [H=H] + \frac{1}{2} \times [H=T]) && \text{prob. choice; assignment} \\
 & + \frac{1}{2} \times (\frac{1}{2} \times [T=H] + \frac{1}{2} \times [T=T]) \\
 \equiv & \frac{1}{4} + \frac{1}{4} && \text{embedding} \\
 \equiv & \frac{1}{2}. && \text{arithmetic}
 \end{aligned}$$

### Interpretation: a proper extension

$$\begin{aligned}
 & wp.(c := H_{1/2} \oplus T).(\frac{1}{2} \times [c=H] + \frac{1}{2} \times [c=T]) \\
 \equiv & \frac{1}{2} \times (\frac{1}{2} \times [H=H] + \frac{1}{2} \times [H=T]) \\
 & + \frac{1}{2} \times (\frac{1}{2} \times [T=H] + \frac{1}{2} \times [T=T]) \\
 \equiv & \frac{1}{2}.
 \end{aligned}$$

The *expected value* of the function  $\frac{1}{2} \times [c=H] + \frac{1}{2} \times [c=T]$  over the distribution of states produced by the program is  $\frac{1}{2}$ .

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

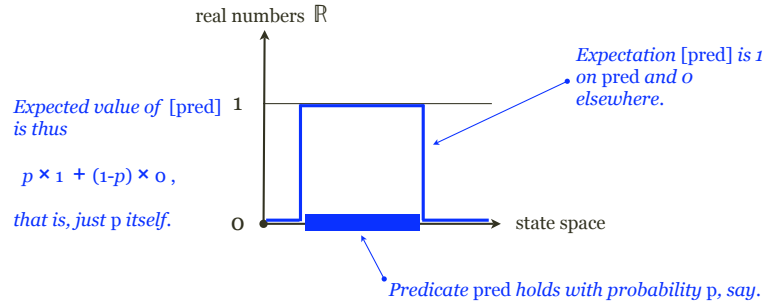
That's why  $wp.prog.[pred]$  gives the probability that  $pred$  is achieved by  $prog$ . But, as we see above, we can be much more general if we wish.

### Interpretation: a proper extension

The expression  $wp.prog.B$  gives, as a function of the initial state, the *expected value* of the “post-expectation”  $B$  over the distribution of final states that  $prog$  will produce from there.

We call it the *greatest pre-expectation* of  $prog$  with respect to  $B$ . When  $prog$  and  $B$  are standard (i.e. non-probabilistic), it is the same as the *weakest precondition*... except that it is 0/1-valued rather than Boolean.

As a “hybrid”, we have that  $wp.prog.[pred]$  is the probability that  $pred$  will be achieved.



### Interpretation: a conservative extension

We note that the standard logic can be embedded in the probabilistic logic simply by converting all Booleans *false*, *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  $prog$ , and Boolean postconditions  $post$ , we have

$$[wp.prog.post] \equiv wp.prog.[post] ,$$

where on the left the *wp* is weakest precondition, and on the right it is greatest pre-expectation.

$$\begin{aligned}
 wp.abort.postE &:= 0 \\
 wp.skip.postE &:= postE \\
 wp.(x := expr).postE &:= postE \langle x \mapsto expr \rangle \\
 wp.(prog; prog').postE &:= wp.prog.(wp.prog'.postE) \\
 wp.(prog \sqcap prog').postE &:= wp.prog.postE \min wp.prog'.postE \\
 wp.(prog \oplus prog').postE &:= p * wp.prog.postE + \bar{p} * wp.prog'.postE
 \end{aligned}$$

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

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

In the case of recursion, however, we cannot give a purely syntactic definition. Instead we say that

$$(\mu \text{ } xxx \cdot C) := \text{least fixed-point of the function } cntx: \mathbb{T}S \rightarrow \mathbb{T}S \text{ defined so that } cntx.(wp.xxx) = wp.C. \quad 40$$

Figure 1.5.3. PROBABILISTIC *wp*-SEMANTICS OF *pGCL*

### Interpretation: demonic choice

$prog_1 \sqcap prog_2$       Execute the left-hand side — or maybe execute the right-hand side. Whatever...

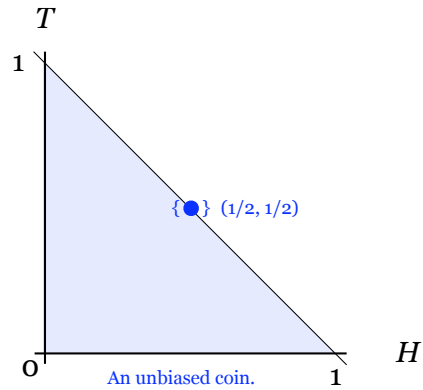
$$wp.(prog_1 \sqcap prog_2).B \equiv wp.prog_1.B \min wp.prog_2.B$$

$$\begin{aligned}
 & wp.(c := H \sqcap c := T).[c=H] \\
 \equiv & wp.(c := H).[c=H] \min wp.(c := T).[c=H] && \text{definition} \\
 \equiv & [H=H] \min [T=H] && \text{assignment} \\
 \equiv & 1 \min 0 && \text{embedding} \\
 \equiv & 0 . && \text{arithmetic}
 \end{aligned}$$

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

Interpretation: a geometric view

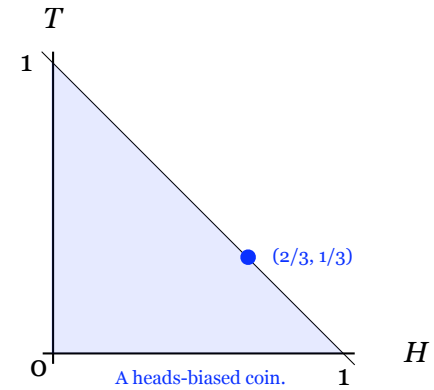
33



McIver and Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems*, Chapter 6. Springer Verlag, 2004.

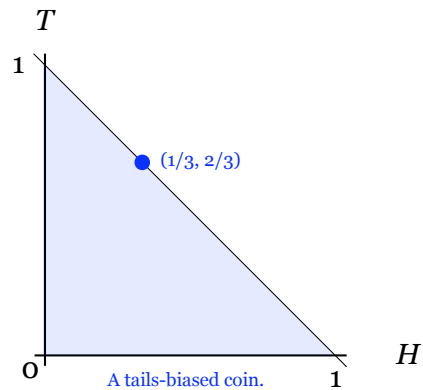
Interpretation: a geometric view

34



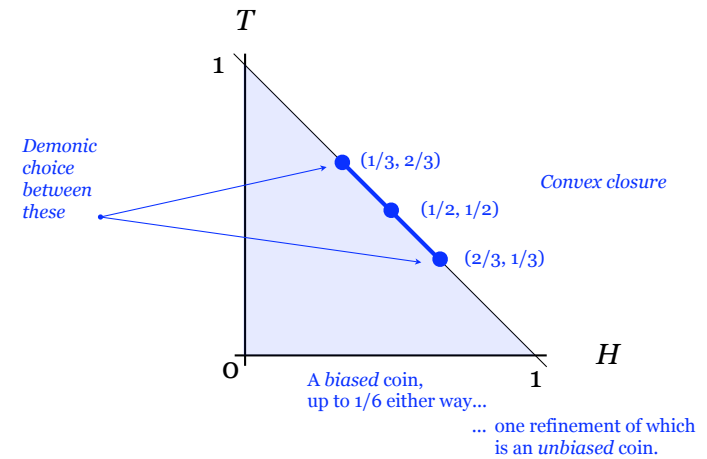
Interpretation: a geometric view

35



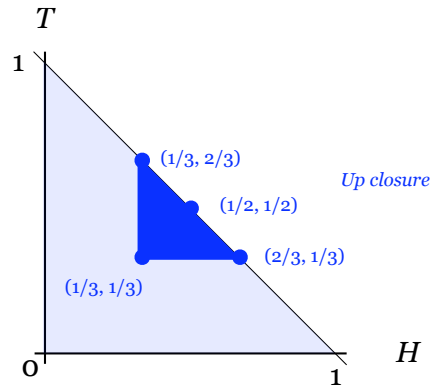
Interpretation: a geometric view

36



Interpretation: a geometric view

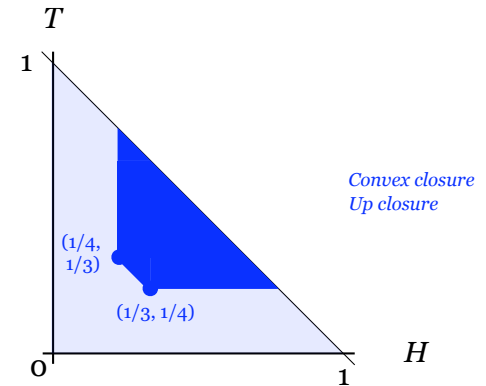
37



A possibly nonterminating coin... whose refinements include all three coins before.

Interpretation: a geometric view

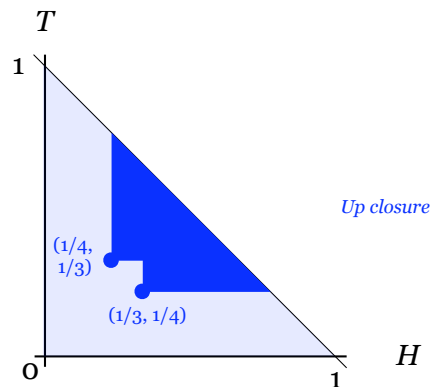
38



Demonically, either of two possibly nonterminating coins.

Interpretation: a geometric view

39

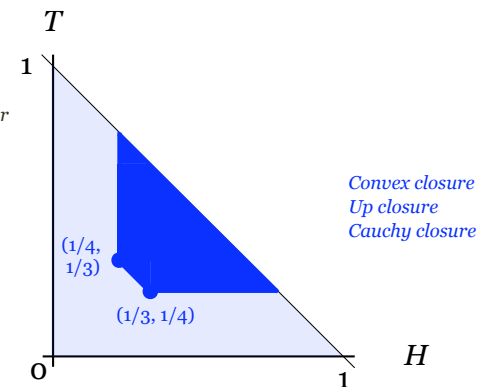


Demonically, either of two possibly nonterminating coins.

Interpretation: a geometric view

40

He, McIver and Seidel. Probabilistic models for the guarded command language. Sci. Comp. Prog. 28:171-192, 1997.

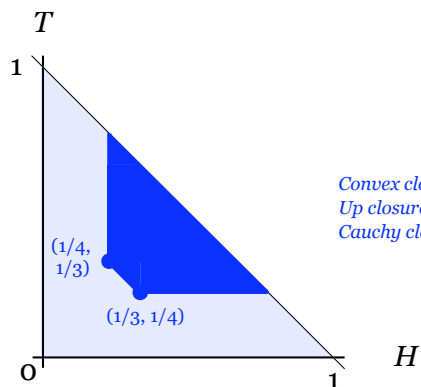


Demonically, either of two possibly nonterminating coins.

Interpretation: a geometric view ... but what's the connection with the programming logic?

He, McIver and Seidel. *Probabilistic models for the guarded command language*. Sci. Comp. Prog. 28:171-192, 1997.

Morgan, McIver and Seidel. *Probabilistic predicate transformers*. ACM TOPLAS 18(3): 325-353, 1996.



Convex closure  
Up closure  
Cauchy closure

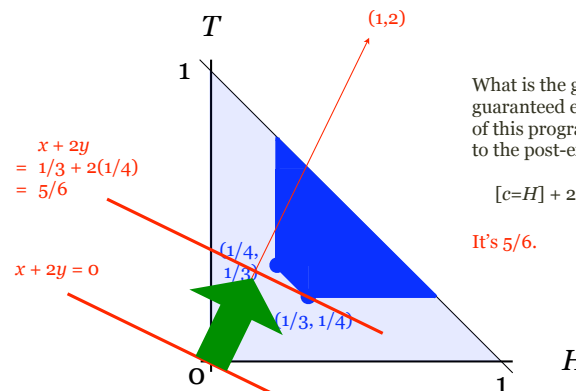
Demonically, either of two possibly nonterminating coins.

Interpretation: a geometric view ... but what's the connection with the programming logic?

What is the greatest guaranteed expected value of this program with respect to the post-expectation

$$[c=H] + 2[c=T] ?$$

It's 5/6.



$$x + 2y = 1/3 + 2(1/4) = 5/6$$

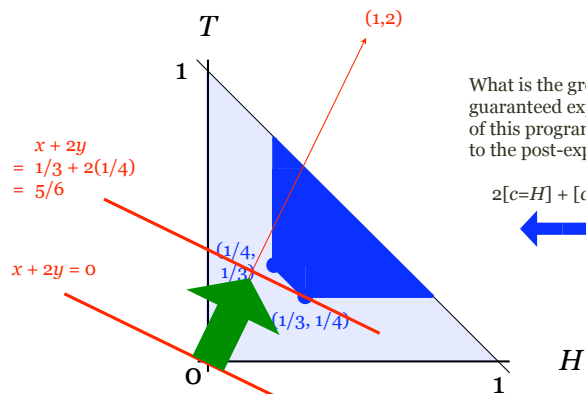
$$x + 2y = 0$$

Demonically, either of two possibly nonterminating coins.

Interpretation: a geometric view ... but what's the connection with the programming logic?

What is the greatest guaranteed expected value of this program with respect to the post-expectation

$$2[c=H] + [c=T] ?$$



$$x + 2y = 1/3 + 2(1/4) = 5/6$$

$$x + 2y = 0$$

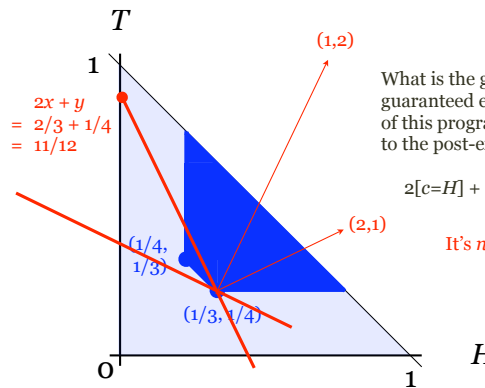
Demonically, either of two possibly nonterminating coins.

Interpretation: a geometric view ... but what's the connection with the programming logic?

What is the greatest guaranteed expected value of this program with respect to the post-expectation

$$2[c=H] + [c=T] ?$$

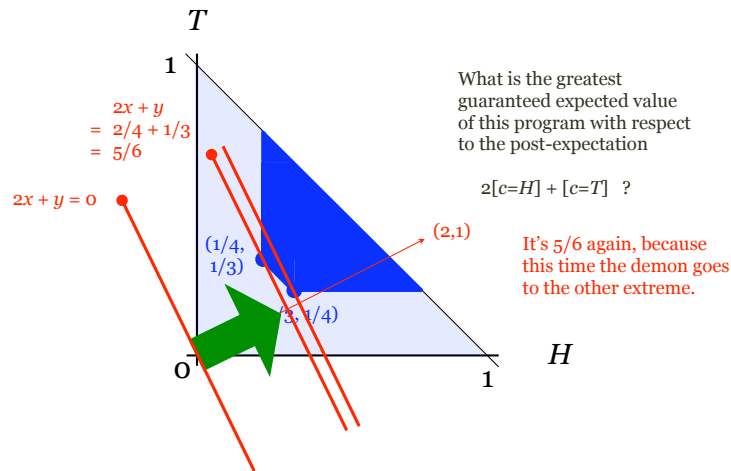
It's not 11/12.



$$2x + y = 2/3 + 1/4 = 11/12$$

Demonically, either of two possibly nonterminating coins.

Interpretation: a geometric view ... but what's the connection with the programming logic?



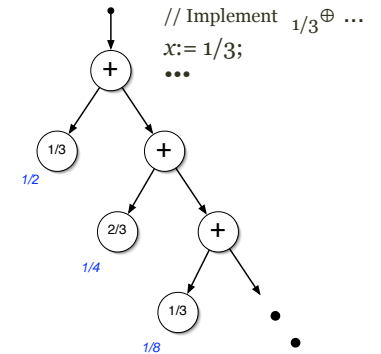
Metatheorems for iteration

// Implement  $p^\oplus$  using unbiased random bits only.

$x := p;$   
 $b := \text{true}_{1/2}^\oplus \text{false};$

**do**  $b \rightarrow$   
 $x := 2x - [x \geq 1/2];$   
 $b := \text{true}_{1/2}^\oplus \text{false};$

**od;**



Metatheorems for iteration

// Implement  $p^\oplus$  using unbiased random bits only.

$x := p;$   
 $b := \text{true}_{1/2}^\oplus \text{false};$

**do**  $b \rightarrow$   
 $x := 2x - [x \geq 1/2];$   
 $b := \text{true}_{1/2}^\oplus \text{false};$

**od;**

**if**  $x \geq 1/2$  // Variable  $x$  at least 1/2 with probability exactly  $p$ .

**then**  $prog_1$

**else**  $prog_2$

**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](http://ewic.bcs.org/conferences/1996/refinement/papers/paper10.htm)

Metatheorems for iteration

$prog_1 \quad p^\oplus \quad prog_2$

is implemented by

and on the average uses only two random bits.

```

begin
  var  $x, b;$ 

   $x := p;$ 
   $b := \text{true}_{1/2}^\oplus \text{false};$ 
  do  $b \rightarrow$ 
     $x := 2x - [x \geq 1/2];$ 
     $b := \text{true}_{1/2}^\oplus \text{false};$ 
  od;

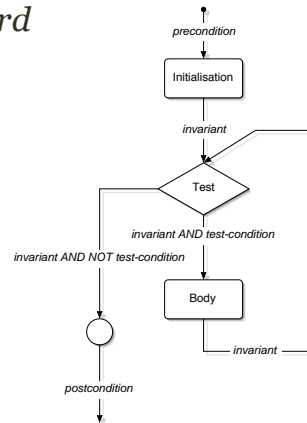
  if  $x \geq 1/2$ 
    then  $prog_1$ 
  else  $prog_2$ 
  fi
end
    
```

## Iteration: reminder of standard metatheorems

```

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.



49

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...

## Standard metatheorems: invariants

```

{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
{x = BE}
    
```

50

## Standard metatheorems: invariants

```

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

51

## Probabilistic metatheorems: invariants *again*

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

52

### Iteration: probabilistic example

```

{?} ←
x:= p;
b:= true 1/2 ⊕ false;
do b →
  x:= 2x - [x ≥ 1/2];
  b:= true 1/2 ⊕ false;
od
{ [x ≥ 1/2] }
    
```

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

### Example: iteration achieves its goal on *termination*

```

[x ≥ 1/2]
⇐ [-b] ×
  ( 2x - [x ≥ 1/2] ◁ b ▷ [x ≥ 1/2] )
  ... if b else ...

x:= p;
b:= true 1/2 ⊕ false;
do b →
  x:= 2x - [x ≥ 1/2];
  b:= true 1/2 ⊕ false;
  { 2x - [x ≥ 1/2] ◁ b ▷ [x ≥ 1/2] }
od
{ [x ≥ 1/2] }
    
```

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

### Example: iteration *body* preserves invariant

```

◻ ≡ ( 2x - [x ≥ 1/2] ◁ b ▷ [x ≥ 1/2] )
≡ ( 2x )/2
≡ x

x:= p;
b:= true 1/2 ⊕ false;
do b →
  x:= 2x - [x ≥ 1/2];
  { x }
  b:= true 1/2 ⊕ false;
  { 2x - [x ≥ 1/2] ◁ b ▷ [x ≥ 1/2] }
od
{ [x ≥ 1/2] }
    
```

### Example: iteration properly *initialised*

```

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

x:= p;
{ x }
b:= true 1/2 ⊕ false;
{ 2x - [x ≥ 1/2] ◁ b ▷ [x ≥ 1/2] }
do b →
  { 2x - [x ≥ 1/2] }
  x:= 2x - [x ≥ 1/2];
  { x }
  b:= true 1/2 ⊕ false;
  { 2x - [x ≥ 1/2] ◁ b ▷ [x ≥ 1/2] }
od
{ [x ≥ 1/2] }
    
```

### Example: correct overall

57

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

is just  $p$ .

```

{p}
x:= p;
{x}
b:= true  $\frac{1}{2} \oplus$  false;
{2x - [x  $\geq$  1/2]  $\triangleleft$  b  $\triangleright$ }
do b  $\rightarrow$  [x  $\geq$  1/2]
  {2x - [x  $\geq$  1/2]}
  x:= 2x - [x  $\geq$  1/2];
  {x}
  b:= true  $\frac{1}{2} \oplus$  false;
  {2x - [x  $\geq$  1/2]  $\triangleleft$  b  $\triangleright$ }
od
{x  $\geq$  1/2}
    
```

### Example: summary

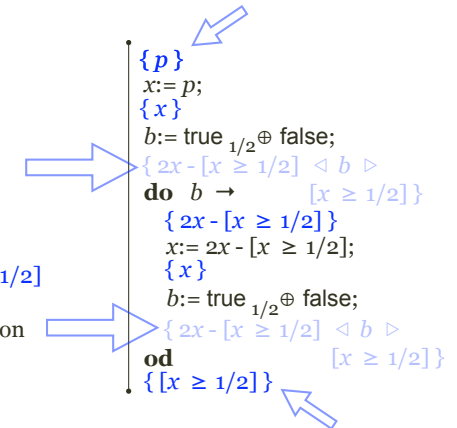
58

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

The loop invariant was

$$2x - [x \geq 1/2] \triangleleft b \triangleright [x \geq 1/2]$$

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



### Termination of probabilistic iterations

59

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

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

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

### Exercises

60

#### Ex. 1: Probabilistic then demonic choice

Calculate  $wp.(c := H \frac{1}{2} \oplus T; d := H \sqcap T).[c=d]$ .

#### Ex. 2: Demonic then probabilistic choice

Calculate  $wp.(d := H \sqcap T; c := H \frac{1}{2} \oplus T).[c=d]$ .

#### Ex. 3: Explain the difference

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

$$c := H_{1/2} \oplus T; \quad d := H_p \oplus T \quad \text{for arbitrary } p,$$

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

#### Ex. 5: Compositionality

Consider the two programs

$$A: \quad \text{coin} := \text{edge} \sqcap (\text{coin} := \text{heads})_{1/2} \oplus \text{coin} := \text{tails}$$

$$B: \quad \begin{array}{l} (\text{coin} := \text{edge} \sqcap \text{coin} := \text{heads}) \\ 1/2 \oplus (\text{coin} := \text{edge} \sqcap \text{coin} := \text{tails}), \end{array}$$

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 *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”?