

# *Formal Methods for Probabilistic Systems*

Annabelle McIver  
Carroll Morgan

- Source-level program logic
- Meta-theorems for loops
- Examples
- Relational operational model
- Almost-certain termination
- Mu-calculus, temporal logic and games

British *EPSRC* (Oxford),  
then Australian *ARC*  
(Macquarie/UNSW),  
*1994-continuing.*

Annabelle McIver  
Carroll Morgan  
Jeff Sanders  
Karen Seidel

`web.comlab.ox.ac.uk/  
oucl/  
research/  
areas/  
probs/`

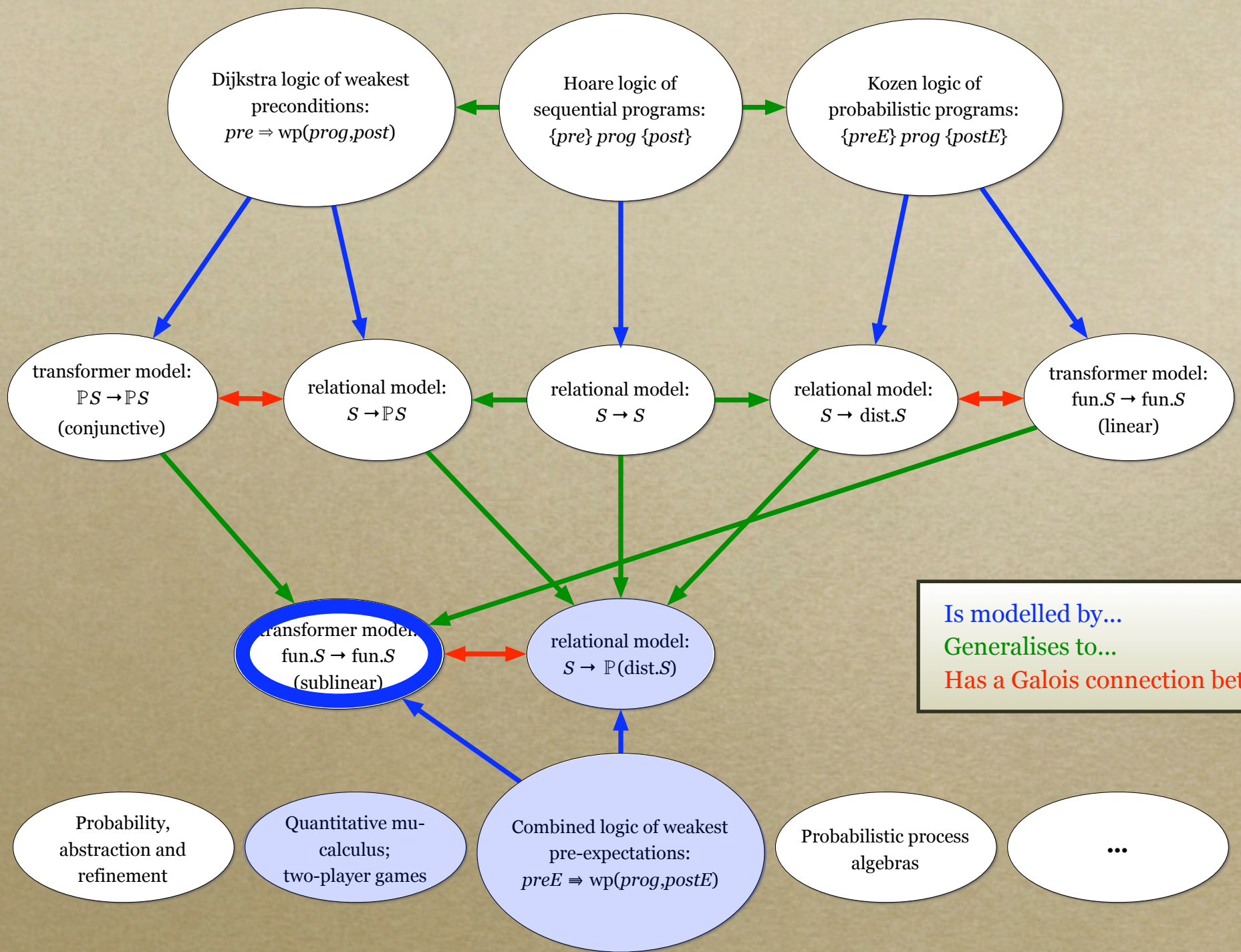
# *Formal Methods for Probabilistic Systems*

Annabelle McIver  
Carroll Morgan

- Source-level program logic
  - Introduction to probabilistic-program logic
  - Systematic presentation via structural induction
  - Layout of calculations in practice
  - Random variables and expected values
  - The impact of demonic choice

add non-determinism

add probability



# *Formal Methods for Probabilistic Systems*

Annabelle McIver  
Carroll Morgan

- Source-level program logic
  - Introduction to probabilistic-program logic
  - Systematic presentation via structural induction
  - Layout of calculations in practice
  - Random variables and expected values
  - The impact of demonic choice

When are two  
probabilistic  
programs equal?

## *Equality of programs — what is it?*

We *say* that two *standard* programs are equal iff from all initial states they have equal assurance of establishing equal postconditions, that is

if for state  $s$  and postcondition  $post$  the first program, started in  $s$  *initially*, is guaranteed to establish  $post$  *finally*... then so is the second program, and vice versa.

(This is so-called *functional equivalence*.)

## *Equality of programs — why is it important?*

One way of specifying what you want is to write a simple, high-level program that does the job...

...and then you say to the implementor “write me a program that works like this, but runs faster, or is distributed, or uses embedded hardware, or...”

Your acceptance, “sign-off” criterion is then

“is the program the implementor wrote *functionally the same* as the program that I wrote?”

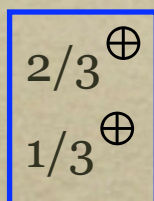
Without that notion of “the same”, there would be nothing you could do if you’re not happy with the delivered product.

## Equality – first guess for probability

Two *probabilistic* programs are equal iff from all initial states they have equal probability of establishing equal postconditions: for example,

$coin := heads$       $\frac{2}{3} \oplus$       $coin := tails$   
 and      $coin := tails$       $\frac{1}{3} \oplus$       $coin := heads$

are equal.



Probabilistic choice.

{}	0
{H}	2/3
{T}	1/3
{H,T}	1

What about

$coin := edge \sqcap (coin := heads \frac{1}{2} \oplus coin := tails)$

Nondeterministic choice.

and

$\frac{1}{2} \oplus (coin := edge \sqcap coin := heads)$   
 $\frac{1}{2} \oplus (coin := edge \sqcap coin := tails)$  .

Are they equal?

## Equality — first guess

$coin := edge \sqcap (coin := heads_{1/2} \oplus coin := tails)$

postcondition

probability

$\{\}$	0
$\{edge\}$	0 - 1
$\{heads\}$	0 - 1/2
$\{tails\}$	0 - 1/2
$\{edge, heads\}$	1/2 - 1
$\{edge, tails\}$	1/2 - 1
$\{heads, tails\}$	0 - 1
$\{edge, heads, tails\}$	1

# Equality — first guess

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

postcondition

probability

$\{\}$	0
$\{\text{edge}\}$	0 - 1
$\{\text{heads}\}$	0 - 1/2
$\{\text{tails}\}$	0 - 1/2
$\{\text{edge}, \text{heads}\}$	1/2 - 1
$\{\text{edge}, \text{tails}\}$	1/2 - 1
$\{\text{heads}, \text{tails}\}$	0 - 1
$\{\text{edge}, \text{heads}, \text{tails}\}$	1

postcondition

probability

$\{\}$	0
$\{\text{edge}\}$	0 - 1
$\{\text{heads}\}$	0 - 1/2
$\{\text{tails}\}$	0 - 1/2
$\{\text{edge}, \text{heads}\}$	1/2 - 1
$\{\text{edge}, \text{tails}\}$	1/2 - 1
$\{\text{heads}, \text{tails}\}$	0 - 1
$\{\text{edge}, \text{heads}, \text{tails}\}$	1

## Equality – first guess

Two (probabilistic) programs are equal iff from all initial states they have equal probability of establishing equal postconditions: for example,

$coin := heads$      $\frac{2}{3} \oplus$      $coin := tails$   
 and     $coin := tails$      $\frac{1}{3} \oplus$      $coin := heads$

are equal.

$\{\}$	0
$\{H\}$	$\frac{2}{3}$
$\{T\}$	$\frac{1}{3}$
$\{H, T\}$	1

What about

$coin := edge \sqcap (coin := heads \frac{1}{2} \oplus coin := tails)$

and

$\frac{1}{2} \oplus (coin := edge \sqcap coin := heads)$   
 $\frac{1}{2} \oplus (coin := edge \sqcap coin := tails)$  .

postcondition	probability
$\{\}$	0
$\{edge\}$	0-1
$\{heads\}$	0-1/2
$\{tails\}$	0-1/2
$\{edge, heads\}$	1/2-1
$\{edge, tails\}$	1/2-1
$\{heads, tails\}$	0-1
$\{edge, heads, tails\}$	1

Are they equal? **Apparently they are.**

# Equality — first guess, wrong guess

## But should they be?

No, they should not — and their indistinguishability in this simple probabilistic logic makes it *non-compositional*.

The logic we now present represents the “least extra effort” —in a sense that can be made precise— that regains compositionality.

What about

$$\text{coin} := \text{edge} \sqcap (\text{coin} := \text{heads} \text{ } \frac{1}{2} \oplus \text{coin} := \text{tails})$$

and

$$\frac{1}{2} \oplus (\text{coin} := \text{edge} \sqcap \text{coin} := \text{heads}) \\ \frac{1}{2} \oplus (\text{coin} := \text{edge} \sqcap \text{coin} := \text{tails}).$$

Are they equal? **Apparently they are.**

postcondition	probability
$\{\}$	0
$\{\text{edge}\}$	0 - 1
$\{\text{heads}\}$	0 - 1/2
$\{\text{tails}\}$	0 - 1/2
$\{\text{edge, heads}\}$	1/2 - 1
$\{\text{edge, tails}\}$	1/2 - 1
$\{\text{heads, tails}\}$	0 - 1
$\{\text{edge, heads, tails}\}$	1

# Probabilistic-program logic: introduction

What is the **probability** that the **program**

$coin := heads \boxed{\frac{1}{2} \oplus} tails$

establishes the **postcondition**  $coin = heads$  ?

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

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

$coin := heads \frac{1}{2} \oplus tails$

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

# Probabilistic-program logic

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 programs

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.

# Probabilistic programs: assignment statements

 $x := E$ 

Assign the value  
of expression  $E$   
to the variable  $x$ .

Informal  
description.

 $wp.(x := E).B$ 
 $\equiv$ 
 $B \langle x := E \rangle$ 

Definition.

Syntactic substitution.

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

definition

 $\equiv [(x+1)=3]$ 

substitution

 $\equiv [x=2]$ 

arithmetic

Example.

Why are these here?

# Probabilistic programs: embedded **Booleans**

$$\begin{aligned}
 & wp.(x := x+1).[x=3] \\
 \equiv & [x=3] \langle x := x+1 \rangle && \text{definition} \\
 \equiv & [(x+1)=3] && \text{substitution} \\
 \equiv & [x=2] && \text{arithmetic}
 \end{aligned}$$

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

Thus “[•]” must be an *embedding function* that takes *true* to one and *false* to zero.

# Probabilistic programs: probabilistic choice

$$prog_1 \stackrel{p}{\oplus} prog_2$$

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

---


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


---

$$\begin{aligned} & wp.(c := H \stackrel{1/2}{\oplus} T).[c=H] \\ \equiv & \quad 1/2 \times wp.(c := H).[c=H] \\ & + (1-1/2) \times wp.(c := T).[c=H] \\ \equiv & \quad 1/2 \times [H=H] + 1/2 \times [T=H] \\ \equiv & \quad 1/2 \times 1 + 1/2 \times 0 \\ \equiv & \quad 1/2. \end{aligned}$$

*definition*

*assignment*

*embedding*

*arithmetic*

# Probabilistic programs: syntactic “sugar”

**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.

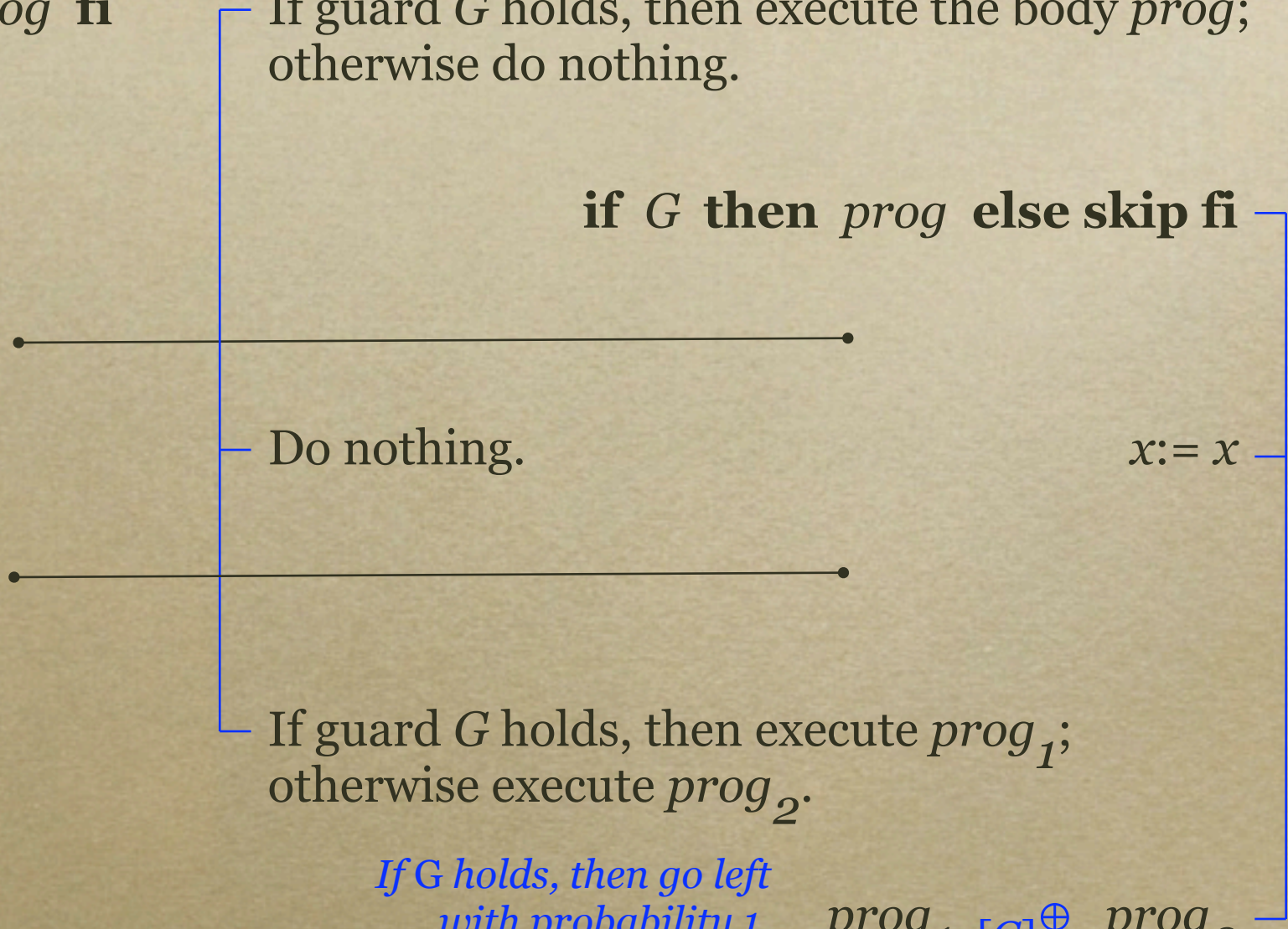
$x := x$

**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.*

$prog_1 [G]^\oplus prog_2$



# Probabilistic programs: conditional

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

$$\equiv wp.(x := x - 1 \ [x \geq 1]^\oplus \ x := x + 2).[x \geq 2]$$

“sugar”

$$\equiv \begin{aligned} & [x \geq 1] \times wp.(x := x - 1).[x \geq 2] \\ & + \boxed{(1 - [x \geq 1])} \times wp.(x := x + 2).[x \geq 2] \end{aligned}$$

prob. choice

$$\equiv [x \geq 1] \times [(x-1) \geq 2] + \boxed{[x < 1]} \times [(x+2) \geq 2]$$

assignment

$$\equiv [x \geq 1] \triangle \times [x \geq 3] \oplus [x < 1] \triangle \times [x \geq 0]$$

arithmetic

$$\equiv [x \geq 1] \triangle \wedge x \geq 3 \ \vee \ x < 1 \triangle \wedge x \geq 0$$

embedding

$$\equiv [x \geq 3 \ \vee \ 0 \leq x < 1].$$

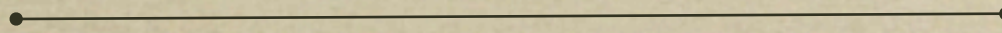
logic

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

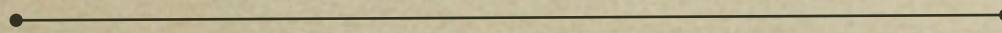
# Probabilistic programs: sequential composition

$prog_1 ; prog_2$

Execute the first program;  
then execute the second.



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



$wp.(c := H_{1/2} \oplus T ; d := H_{1/2} \oplus T).[c=d]$

$\equiv wp.(c := H_{1/2} \oplus T). (wp.(d := H_{1/2} \oplus T).[c=d])$  *definition*

$\equiv wp.(c := H_{1/2} \oplus T). ($  *prob. choice; assignment*

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

)

$\equiv 1/2 \times (1/2 \times [H=H] + 1/2 \times [H=T])$  *prob. choice; assignment*

$+ 1/2 \times (1/2 \times [T=H] + 1/2 \times [T=T])$

$\equiv 1/4 + 1/4$

*embedding*

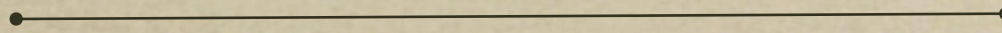
$\equiv 1/2 .$

*arithmetic*

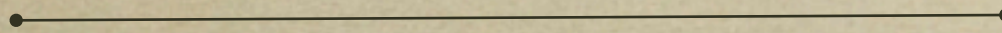
# Probabilistic programs: sequential composition

$prog_1 ; prog_2$

Execute the first program;  
then execute the second.



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



$wp.(c := H_{1/2} \oplus T ; d := H_{1/2} \oplus T).[c=d]$

$\equiv wp.(c := H_{1/2} \oplus T).(wp.(d := H_{1/2} \oplus T).[c=d])$

*definition*

$\equiv wp.(c := H_{1/2} \oplus T).($

*prob. choice; assignment*

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

)

$\equiv 1/2 \times (1/2 \times [H=H] + 1/2 \times [H=T])$

*prob. choice; assignment*

$+ 1/2 \times (1/2 \times [T=H] + 1/2 \times [T=T])$

$\equiv 1/4 + 1/4$

*embedding*

$\equiv 1/2 .$

*arithmetic*

*Probabilistic programs*: “proper” *probabilistic*  
postconditions

$$wp.(c := H \oplus_{1/2} T).(1/2 \times [c=H] + 1/2 \times [c=T])$$

$$\equiv \begin{aligned} & 1/2 \times (1/2 \times [H=H] + 1/2 \times [H=T]) \\ + & 1/2 \times (1/2 \times [T=H] + 1/2 \times [T=T]) \end{aligned}$$

$$\equiv 1/2 .$$

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  $[pred]$ , for some Boolean *pred*, is just the probability that *pred* holds.

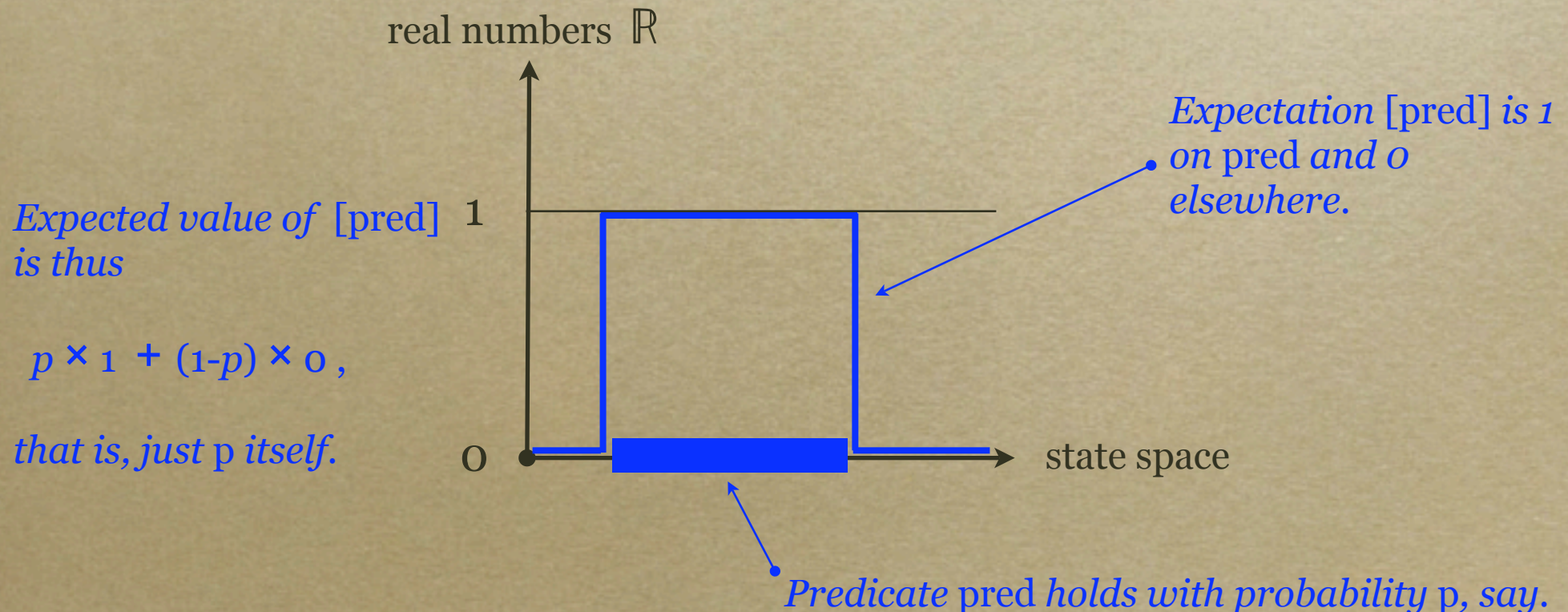
That’s why  $wp.prog.[pred]$  gives the probability that *pred* is achieved by *prog*. But, as we see above, we can be much more general if we wish.

## Probabilistic programs: proper *post-expectations*

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.



# Probabilistic programs: demonic choice

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

---


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


---

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

*definition*

*assignment*

*embedding*

*arithmetic*

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

# Exercises

*Ex. 1: Probabilistic then demonic choice*

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

*Ex. 2: Demonic then probabilistic choice*

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

*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

Recall programs

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

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

which we now 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. (We showed this by tabulation.)

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 “;” .) What does this tell you about the simple program logic we considered briefly at the beginning?

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 when compared to the simple logic?