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)

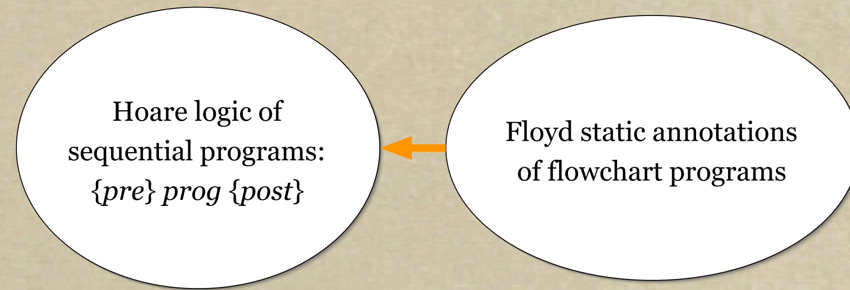
Program logic

Hoare logic of
sequential programs:
{*pre*} *prog* {*post*}

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

Hoare logic of
sequential programs:
{*pre*} *prog* {*post*}

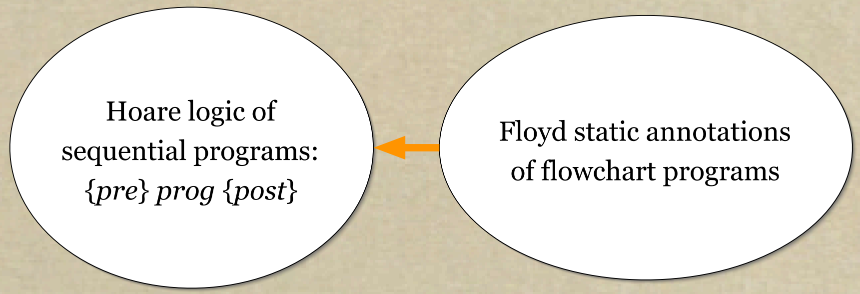
Program logic

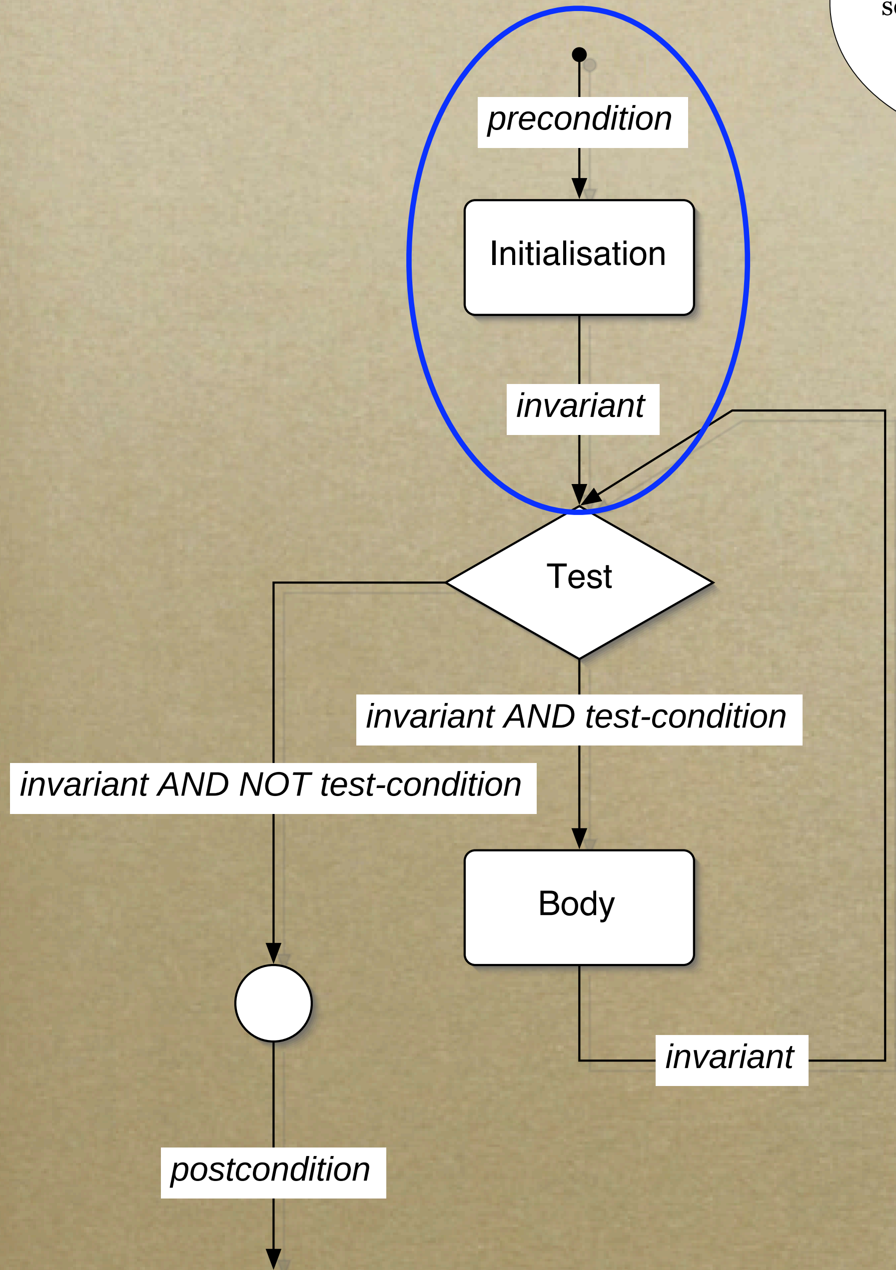


Inspired...

Floyd static annotations of flowchart programs

Program logic

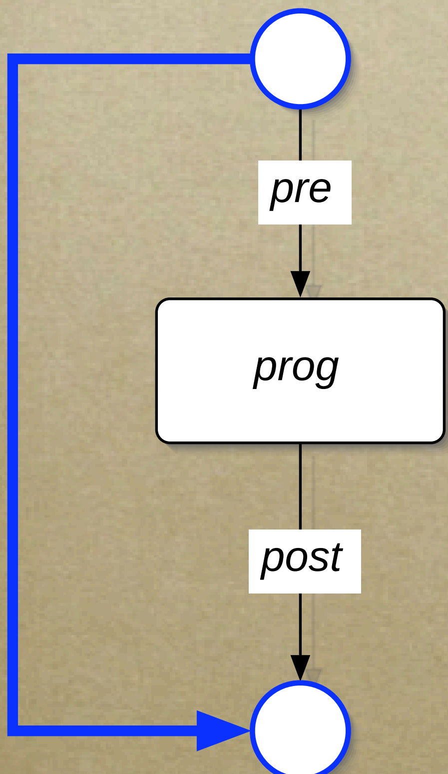


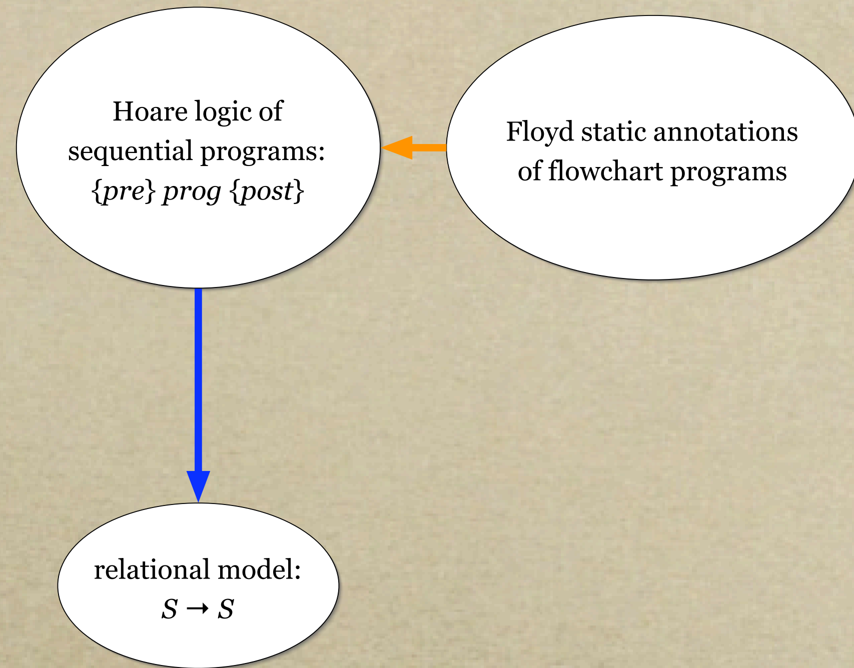


Inspired...

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

Program logic





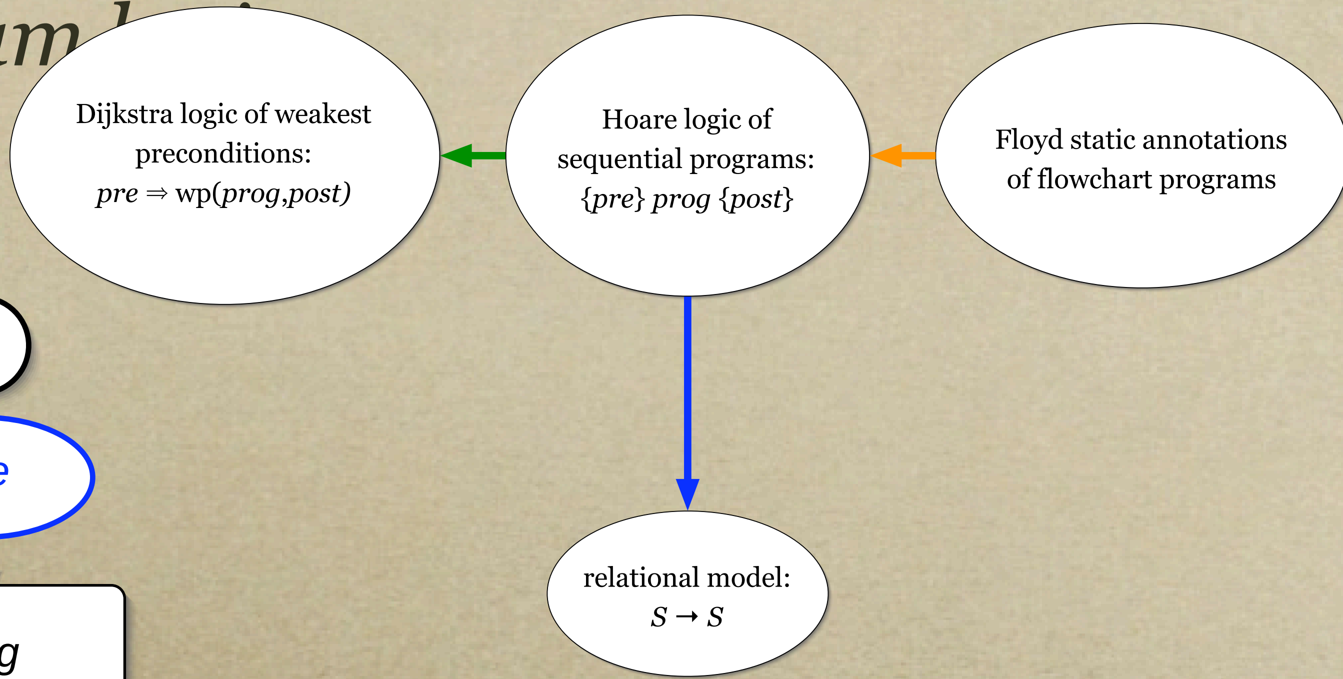
Inspired...
Is modelled by...

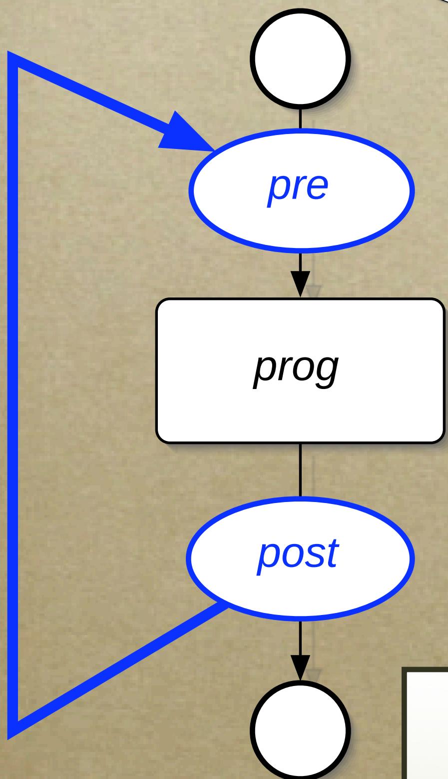
relational model:
$S \to S$

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

add *non-determinism*

Program l





Inspired...
Is modelled by...
Generalises to...

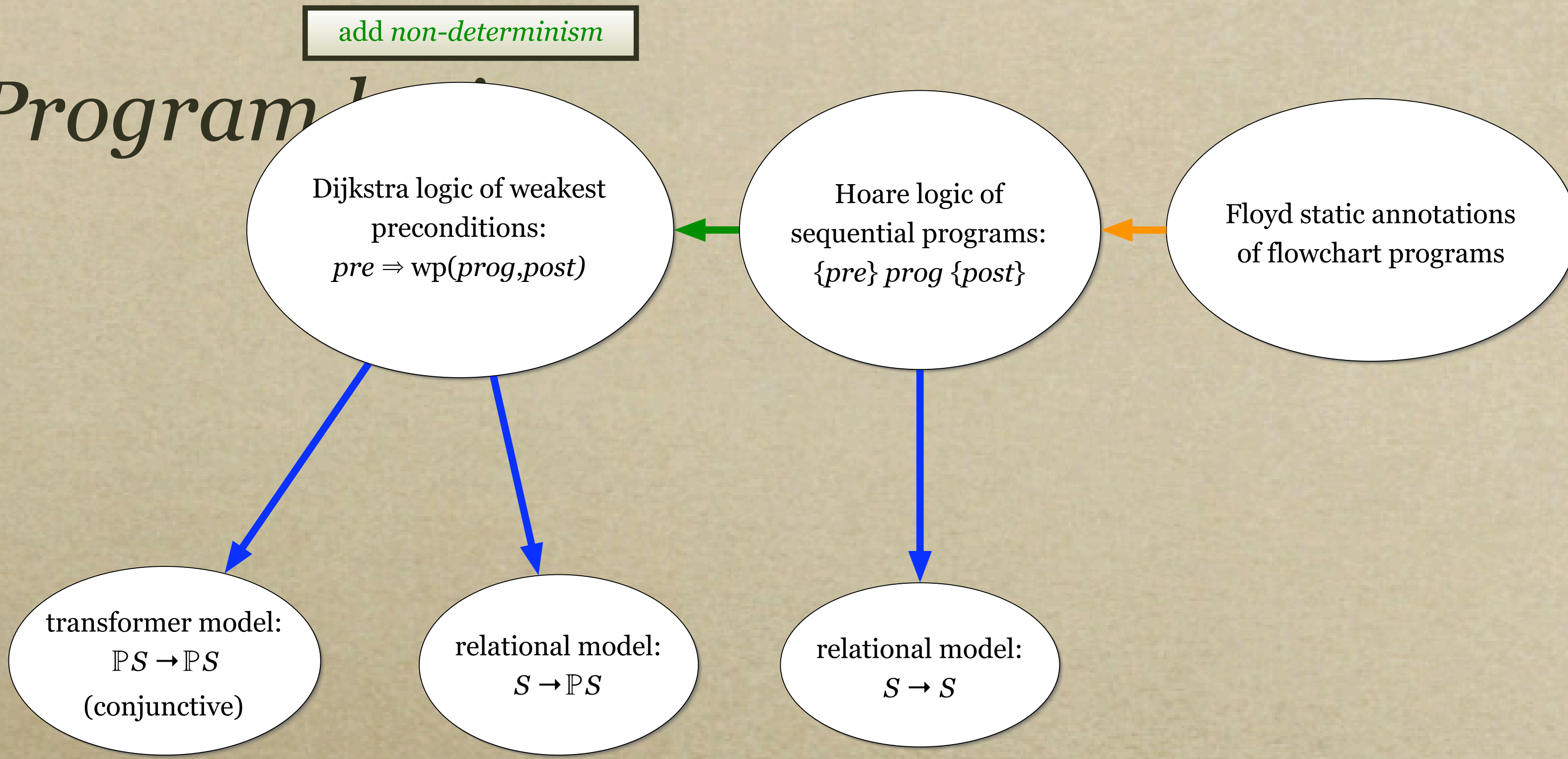
add *non-determinism*

Dijkstra logic of weakest preconditions:

pre ⇒ wp(*prog*,*post*)

E.W. Dijkstra.
A Discipline of Programming.
Prentice Hall, 1976

Program l



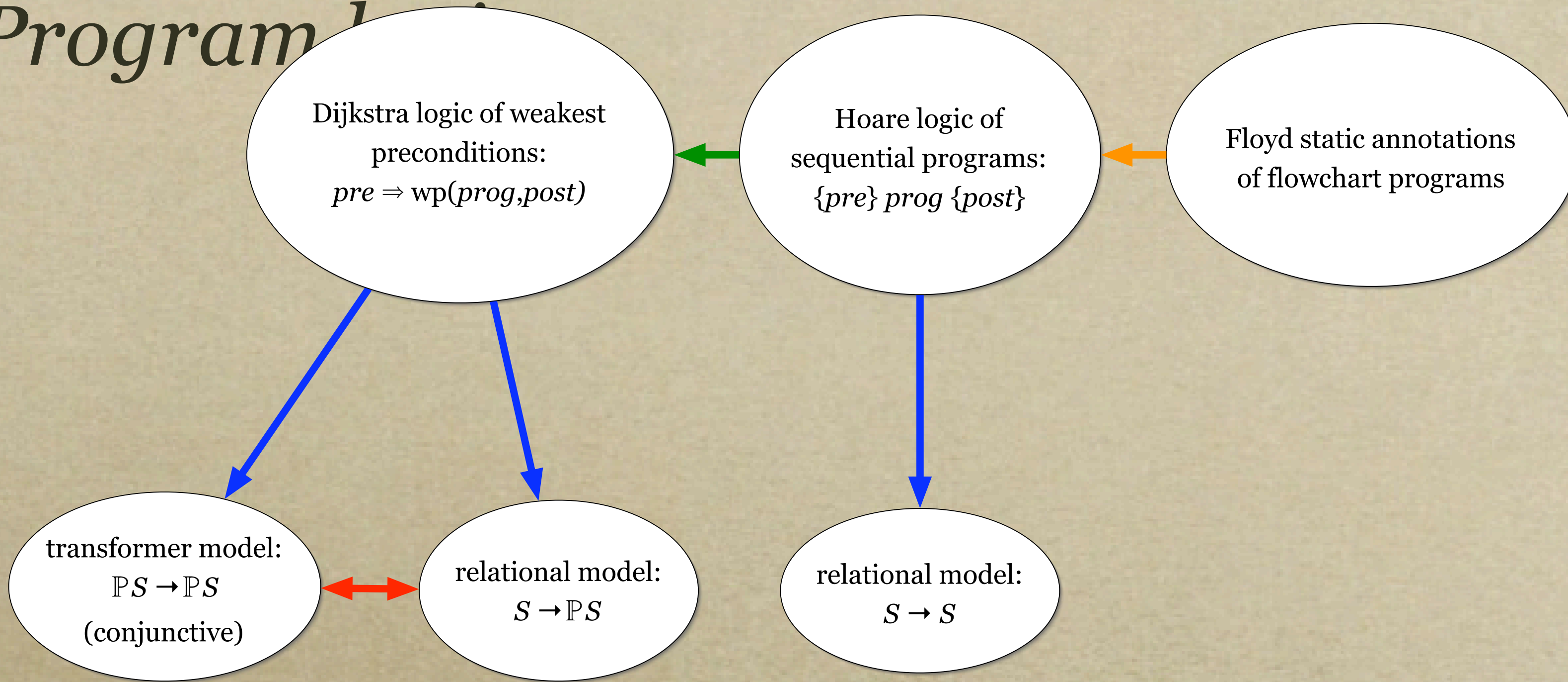
Inspired...
Is modelled by...
Generalises to...

transformer model:
$\mathbb{P}S \rightarrow \mathbb{P}S$
(conjunctive)

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

add *non-determinism*

Program



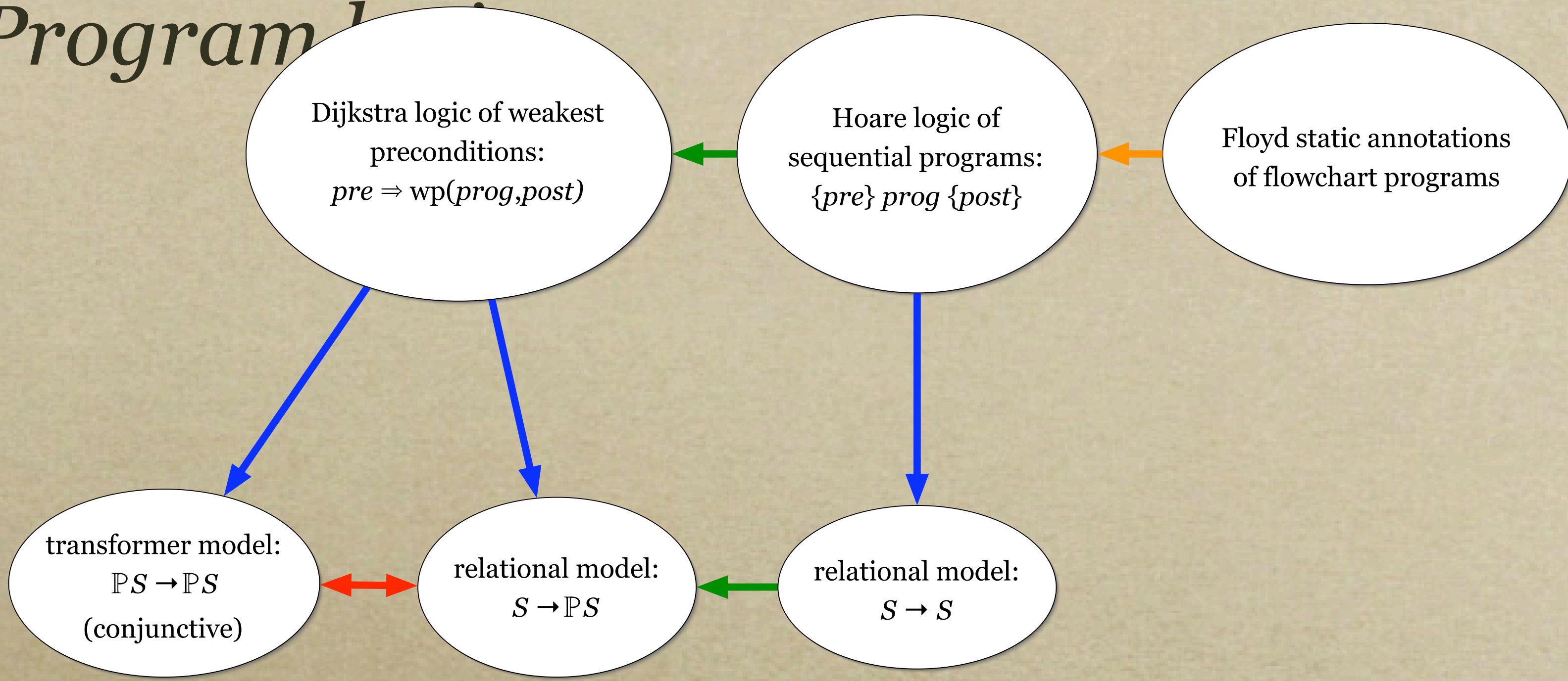
Inspired...
Is modelled by...
Generalises to...
Has a Galois connection between...

transformer model:
$\mathbb{P}S \to \mathbb{P}S$
(conjunctive)

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

add *non-determinism*

Program

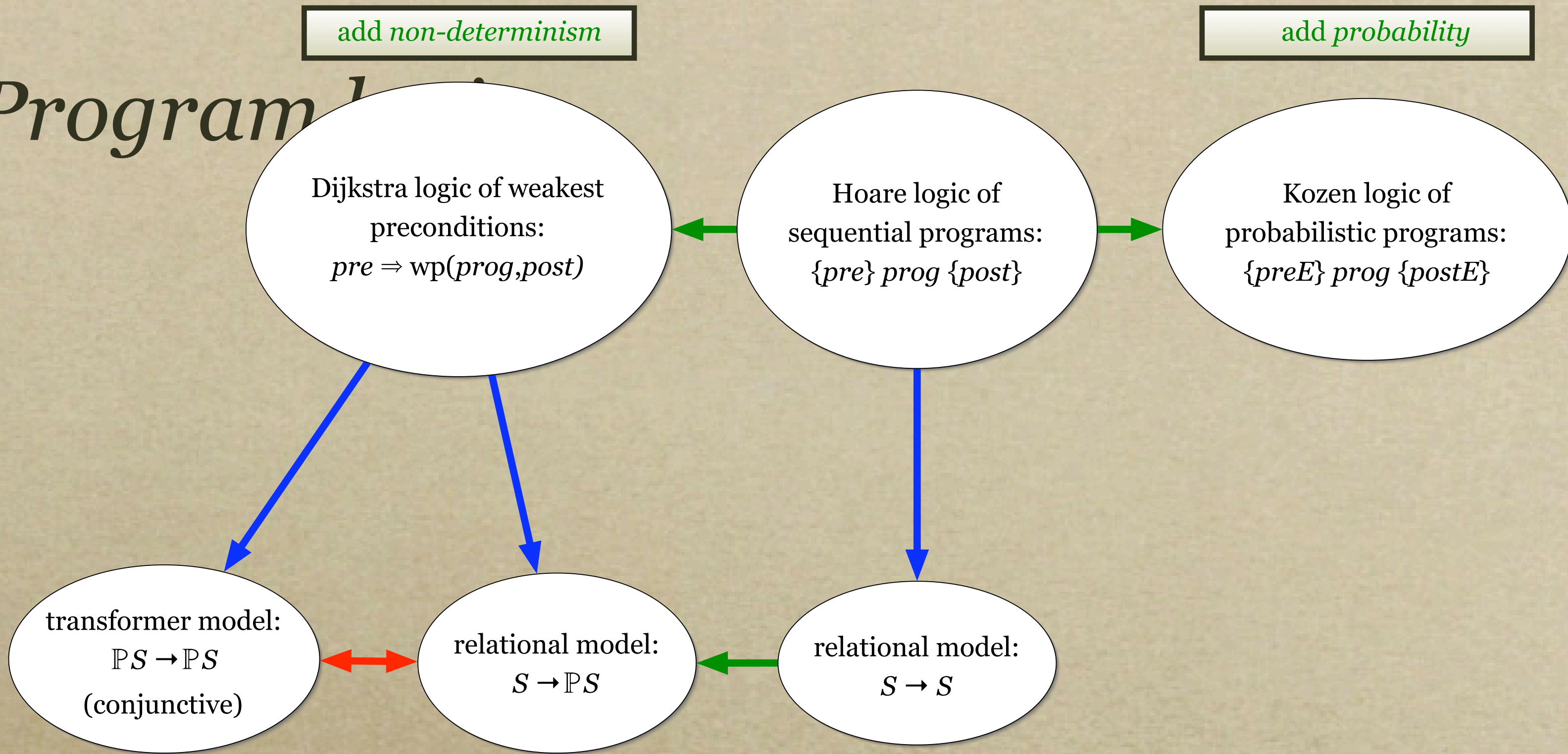


Inspired...
Is modelled by...
Generalises to...
Has a Galois connection between...

transformer model:
$\mathbb{P}S \to \mathbb{P}S$
(conjunctive)

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

Program



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

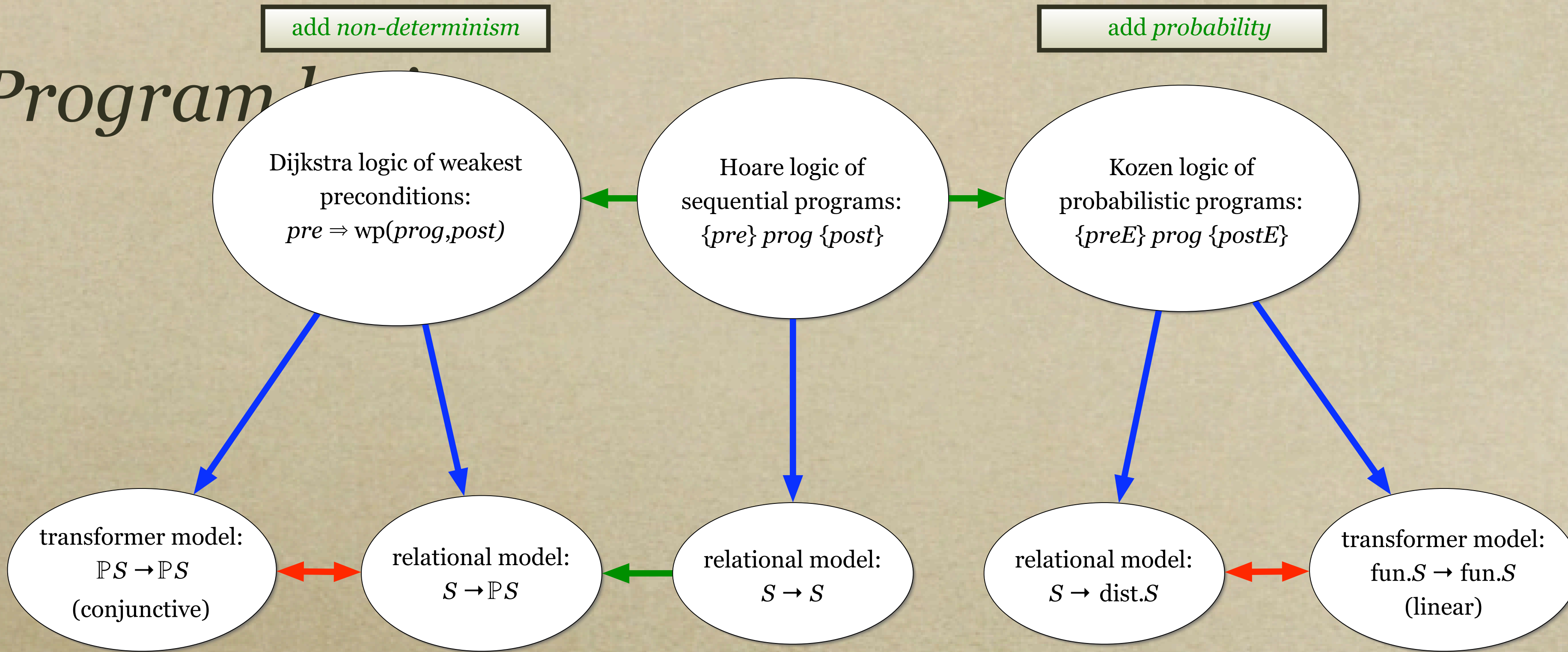
add *probability*

Kozen logic of probabilistic programs: {*preE*} *prog* {*postE*}

D. Kozen.
Semantics of probabilistic programs.
Journal of Computer and System Sciences,
1981

A probabilistic *PDL*. *Proc. 15th* STOC, ACM, 1983

Program

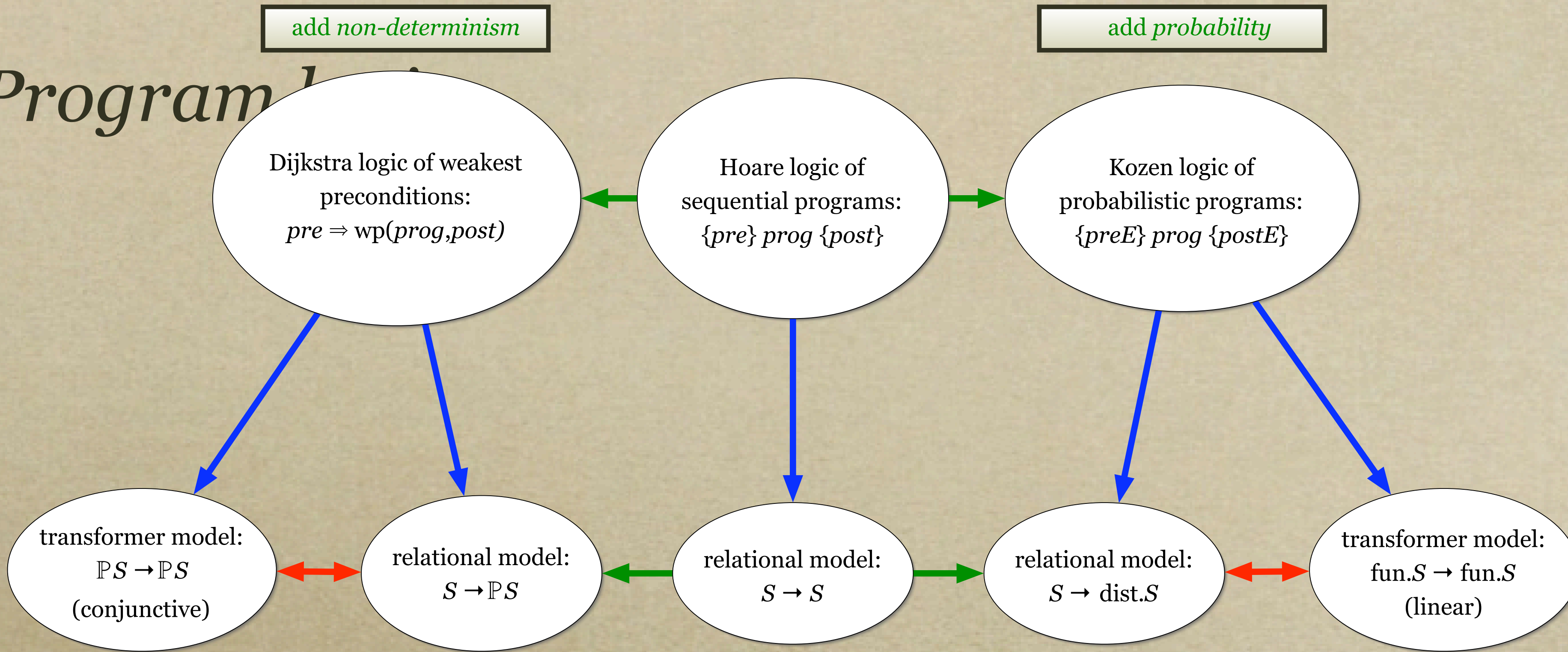


Is modelled by...
Generalises to...
Has a Galois connection between...

relational model:
S → dist.*S*

transformer model:
fun.*S* → fun.*S*
(linear)

Program l



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

relational model:
S → dist.S

transformer model:
fun.S → fun.S
(linear)

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.

Co bined logic of wea est
re-expectations
pre ⇛ wp(*prog*,*p* *tE*)

add *no eterm n and*
babi

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

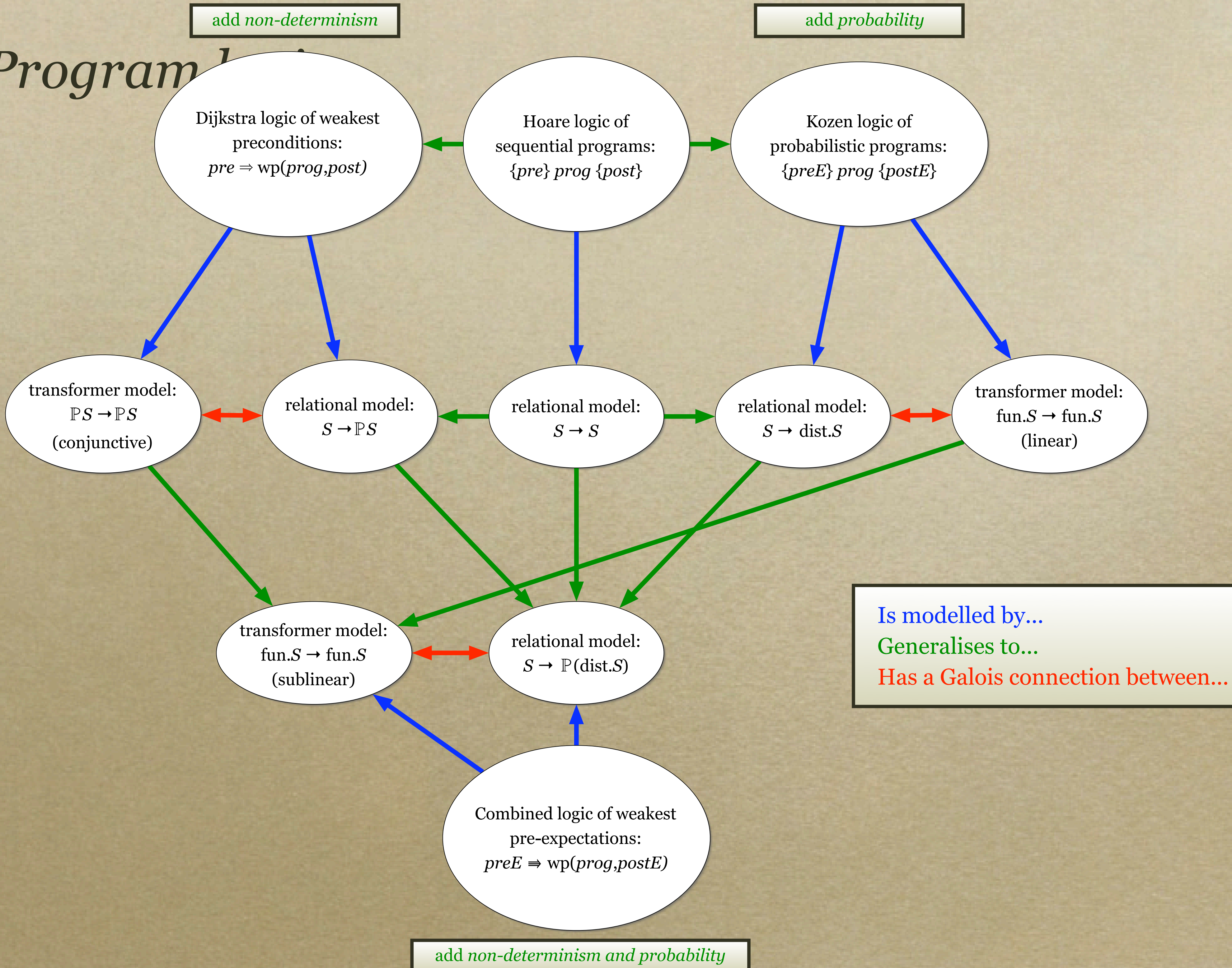
J. He, A. McIver. K. Seidel
Probabilistic models for the guarded command language
Science of Computer Programming 28, 1997

Combined logic of weakest pre-expectations:
preE ⇛ wp(*prog*,*postE*)

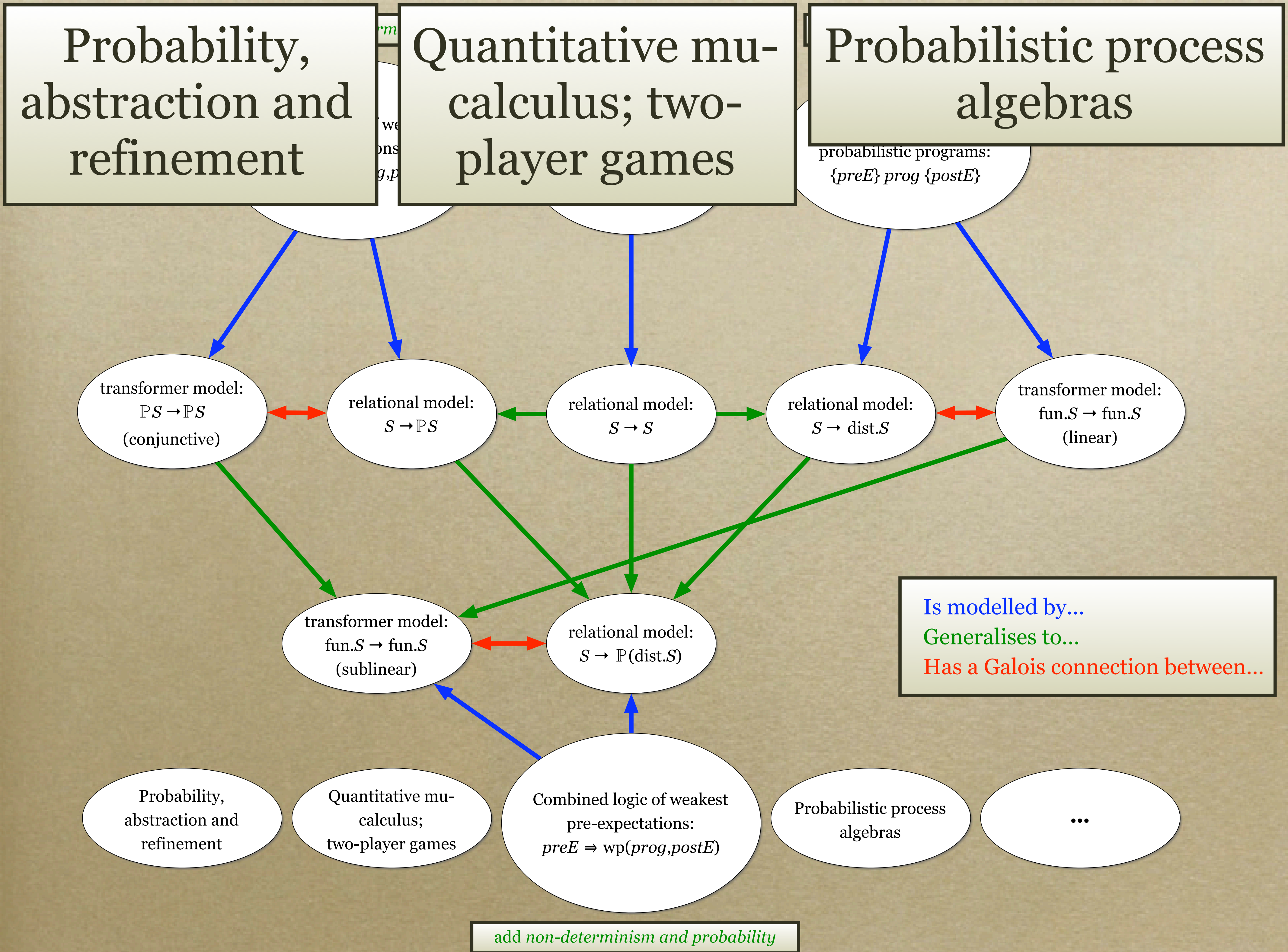
add *non-determinism and probability*

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

Program l



add non-determinism
add probability
Dijkstra logic of weakest preconditions: pre ⇒ wp(prog,post)
Hoare logic of sequential programs: {pre} prog {post}
Kozen logic of probabilistic programs: {preE} prog {postE}
transformer model: ℙS → ℙS (conjunctive)
relational model: S → ℙS
relational model: S → S
relational model: S → dist.S
transformer model: fun.S → fun.S (linear)
transformer model: fun.S → fun.S (sublinear)
relational model: S → ℙ(dist.S)
Combined logic of weakest pre-expectations: preE ⇛ wp(prog,postE)
add non-determinism and probability
Is modelled by...
Generalises to...
Has a Galois connection between...


Probability, abstraction and refinement
Quantitative mu-calculus; two-player games
Probabilistic process algebras
probabilistic programs:
{preE} prog {postE}
transformer model:
ℙS → ℙS
(conjunctive)
relational model:
S → ℙS
relational model:
S → S
relational model:
S → dist.S
transformer model:
fun.S → fun.S
(linear)
transformer model:
fun.S → fun.S
(sublinear)
relational model:
S → ℙ(dist.S)
Is modelled by...
Generalises to...
Has a Galois connection between...
Probability,
abstraction and
refinement
Quantitative mu-
calculus;
two-player games
Combined logic of weakest
pre-expectations:
preE ⇛ wp(prog,postE)
Probabilistic process
algebras
...
add non-determinism and probability

Probabilistic-program logic: introduction

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

What is the probability that the program

$$coin := heads\ {}_{1/2}\!\oplus\ tails$$

establishes the postcondition $coin = heads$?

In the program logic we write

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

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

program (fragment)

non-negative real-valued expression over program variables

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

Interpretation

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

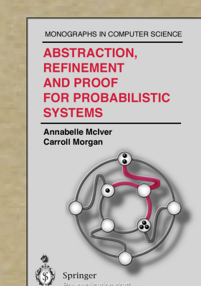
— written in *pGCL*.

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.

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

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

Interpretation: probabilistic choice

$prog_1 \;{}_{p}{\oplus}\; prog_2$

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

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

$$wp.(\, c{:=}\, H \;{}_{1/2}{\oplus}\; T\,).[c{=}H]$$

$\equiv \quad 1/2 \times wp.(c{:=}\,H).[c{=}H] + (1\text{-}1/2) \times wp.(c{:=}\,T\,).[c{=}H]$ *definition*

$\equiv \quad 1/2 \times [H{=}H] \;+\; 1/2 \times [T{=}H]$ *assignment*

$\equiv \quad 1/2 \times 1 \;+\; 1/2 \times 0$ *embedding*

$\equiv \quad 1/2\,.$ *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

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

$\equiv \quad wp.(x{:=}\ x - 1\ {}_{[x \geq 1]}\!\oplus\ x{:=}x + 2).[x \geq 2]$ *"sugar"*

$\equiv \quad [x \geq 1] \times wp.(x{:=}\ x - 1).[x \geq 2]$ *prob. choice*

$\quad + \ (1 - [x \geq 1]) \times wp.(x{:=}x + 2).[x \geq 2]$

$\equiv \quad [x \geq 1] \times [(x-1) \geq 2] + [x < 1] \times [(x+2) \geq 2]$ *assignment*

$\equiv \quad [x \geq 1] \times [x \geq 3] + [x < 1] \times [x \geq 0]$ *arithmetic*

$\equiv \quad [x \geq 1 \wedge x \geq 3 \ \vee \ x < 1 \wedge x \geq 0]$ *embedding*

$\equiv \quad [x \geq 3 \ \vee \ 0 \leq x < 1]\,.$ *logic*

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.

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

$$
\begin{array}{lll}
 & wp.(c{:=}\,H \;{}_{1/2}{\oplus}\; T\,;\; d{:=}\,H \;{}_{1/2}{\oplus}\; T).[c{=}d] & \\
\equiv & wp.(c{:=}\,H \;{}_{1/2}{\oplus}\; T).\,(\,\boxed{wp.(d{:=}\,H \;{}_{1/2}{\oplus}\; T).[c{=}d]}\,) & \textit{definition} \\
\equiv & wp.(c{:=}\,H \;{}_{1/2}{\oplus}\; T).\,(& \textit{prob. choice; assignment} \\
 & \quad \boxed{1/2 \times [c{=}H] \;+\; 1/2 \times [c{=}T]} & \\
 &) & \\
\equiv & \quad 1/2 \times (1/2 \times [H{=}H] + 1/2 \times [H{=}T]) & \textit{prob. choice; assignment} \\
 & + \; 1/2 \times (1/2 \times [T{=}H] + 1/2 \times [T{=}T]) & \\
\equiv & 1/4 + 1/4 & \textit{embedding} \\
\equiv & 1/2\,. & \textit{arithmetic}
\end{array}
$$

Interpretation: sequential composition

$prog_1 \,;\; prog_2$

Execute the first program; then execute the second.

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

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

$\equiv \quad wp.(c{:=}\, H \;_{1/2}{\oplus}\; T).\,(\,wp.(d{:=}\, H \;_{1/2}{\oplus}\; T).[c{=}d]\,)$ — *definition*

$\equiv \quad wp.(c{:=}\, H \;_{1/2}{\oplus}\; T).\,($ — *prob. choice; assignment*

$\qquad 1/2 \times [c{=}H] \;+\; 1/2 \times [c{=}T]$

$\quad)$

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

$\quad + \; 1/2 \times (1/2 \times [T{=}H] + 1/2 \times [T{=}T])$

$\equiv \quad 1/4 + 1/4$ — *embedding*

$\equiv \quad 1/2\,.$ — *arithmetic*

Interpretation: a proper extension

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

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

$$\equiv \quad 1/2\ .$$

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.

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.

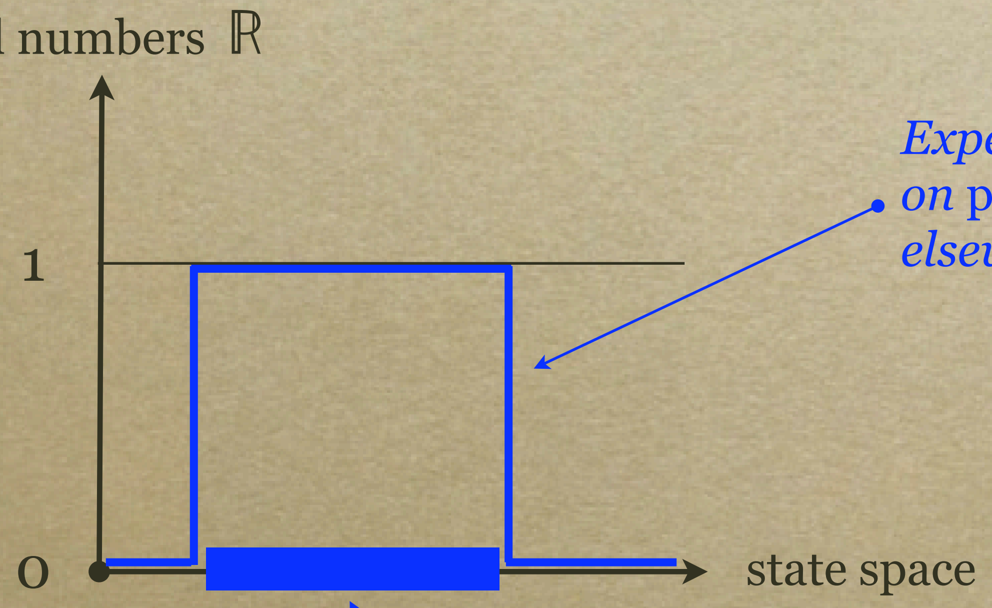
We call it the *greatest pre-expectation* of *prog* with respect to *B*. When *prog* and *B* are standard (*i.e.* non-probabilistic), it is the same as the *weakest precondition*... except that it is 0/1-valued rather than Boolean.

As a "hybrid", we have that *wp.prog.*[*pred*] is the probability that *pred* will be achieved.

Expected value of [pred] *is thus*

$p \times 1 + (1-p) \times 0$,

that is, just p *itself.*



Expectation [pred] *is 1 on* pred *and 0 elsewhere.*

Predicate pred *holds with probability* p*, say.*

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
>
> $$[wp.prog.post] \quad \equiv \quad wp.prog.[post] \;,$$
>
> where on the left the *wp* is weakest precondition, and on the right it is greatest pre-expectation.

$$
\begin{array}{rcl}
wp.\mathbf{abort}.postE & := & 0 \\
wp.\mathbf{skip}.postE & := & postE \\
wp.(x := expr).postE & := & postE\ \langle x \mapsto expr \rangle \\
wp.(prog; prog').postE & := & wp.prog.(wp.prog'.postE) \\
wp.(prog \sqcap prog').postE & := & wp.prog.postE \ \ \mathsf{min} \ \ wp.prog'.postE \\
wp.(prog\ {}_p\oplus\ prog').postE & := & p * wp.prog.postE \ \ + \ \ \overline{p} * wp.prog'.postE
\end{array}
$$

Recall that $\overline{p}$ 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

$$
(\mathbf{mu}\ xxx \bullet \mathcal{C}) \quad := \quad \text{least fixed-point of the function } cntx\colon \mathbb{T}S \to \mathbb{T}S \text{ defined so that } cntx.(wp.xxx) = wp.\mathcal{C}.
$$
[40]

Figure 1.5.3. Probabilistic wp-semantics of $pGCL$

Interpretation: demonic choice

$prog_1 \sqcap prog_2$

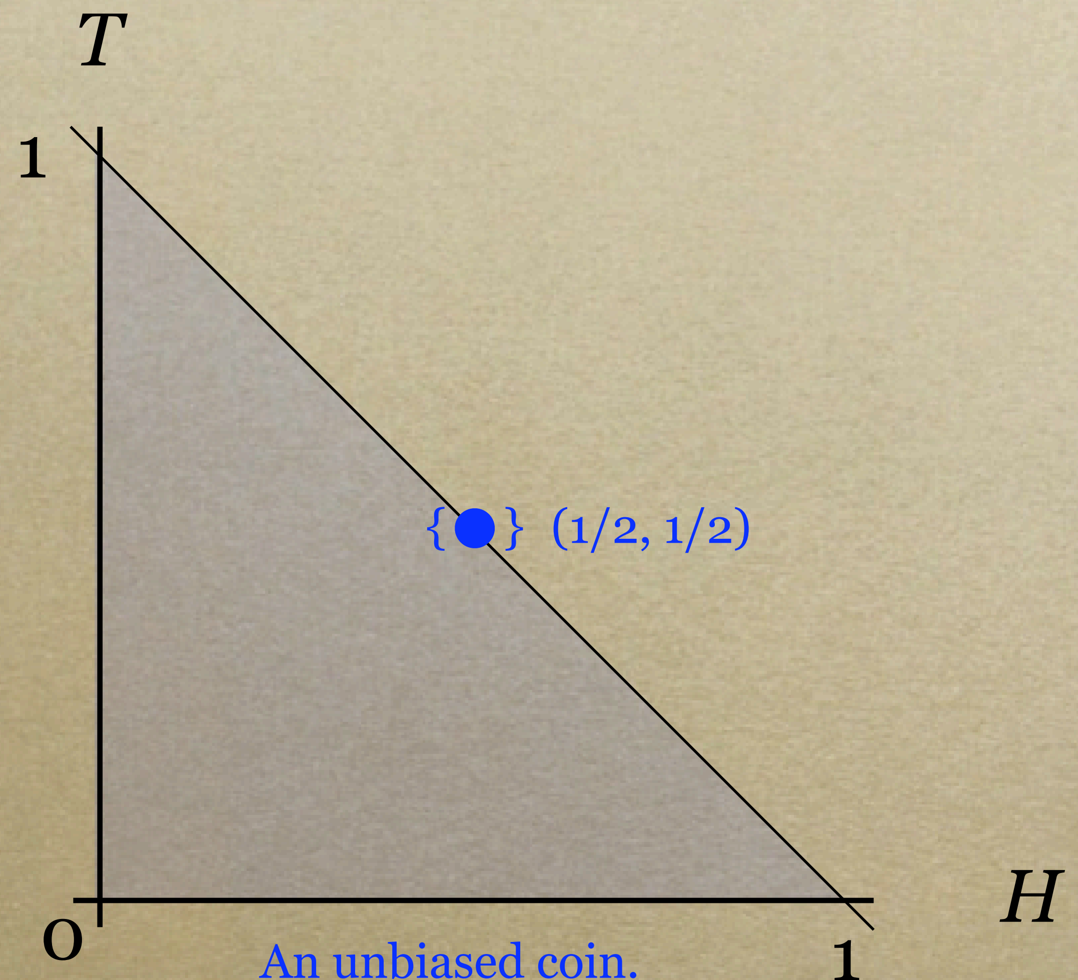
Execute the left-hand side — or maybe execute the right-hand side. *Whatever...*

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

$$
\begin{array}{lll}
 & wp.(c{:=}H \sqcap c{:=}T).[c{=}H] & \\
\equiv & wp.(c{:=}H).[c{=}H] \ \mathbf{min} \ wp.(c{:=}T).[c{=}H] & \textit{definition} \\
\equiv & [H{=}H] \ \mathbf{min} \ [T{=}H] & \textit{assignment} \\
\equiv & 1 \ \mathbf{min} \ 0 & \textit{embedding} \\
\equiv & 0\,. & \textit{arithmetic}
\end{array}
$$

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

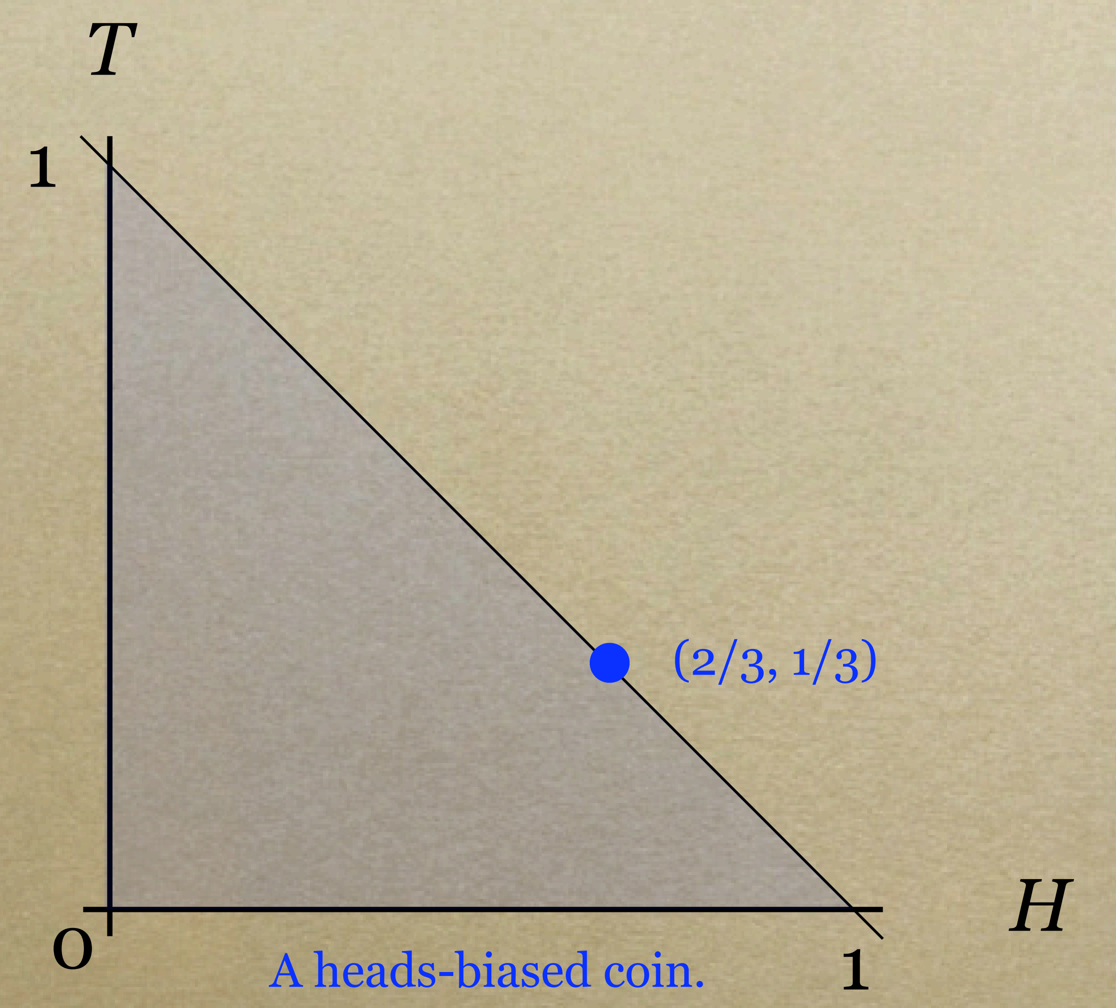
Interpretation: a geometric view



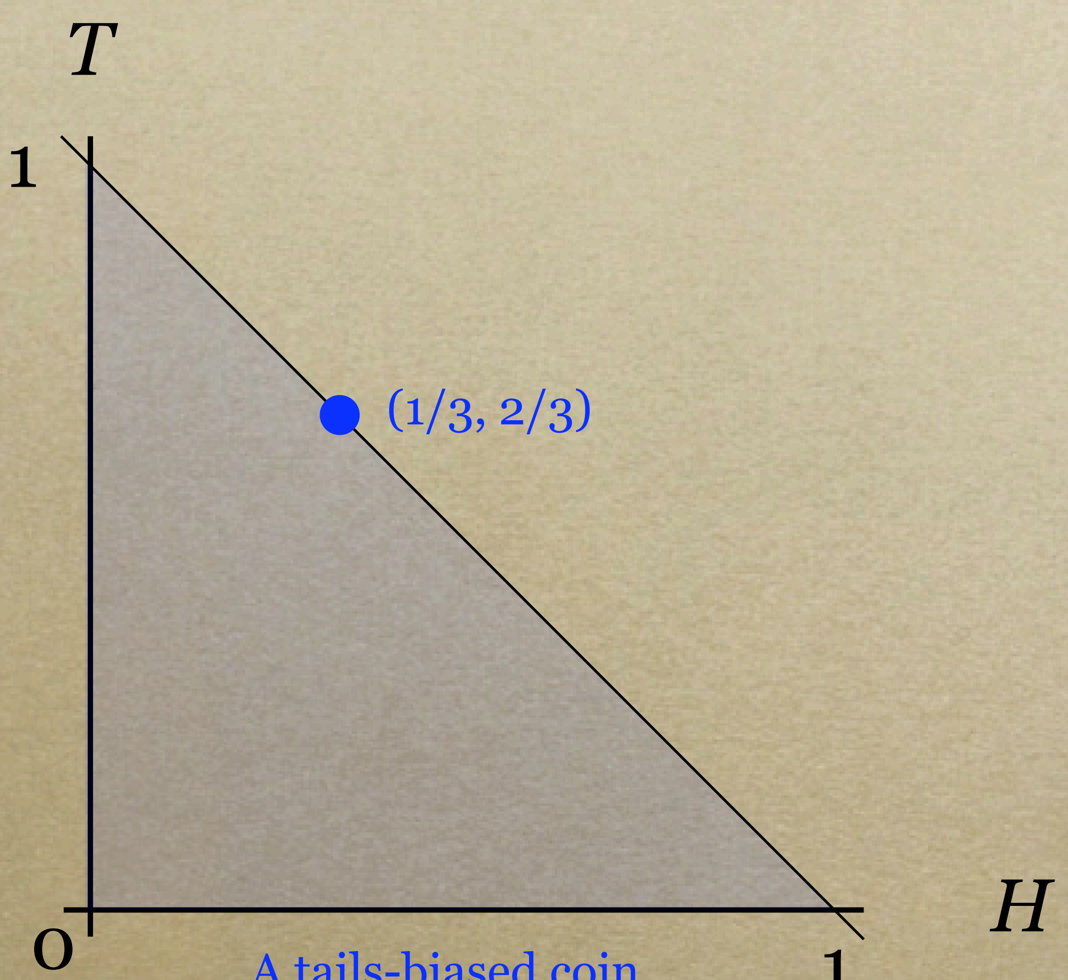
An unbiased coin.

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

Interpretation: a geometric view

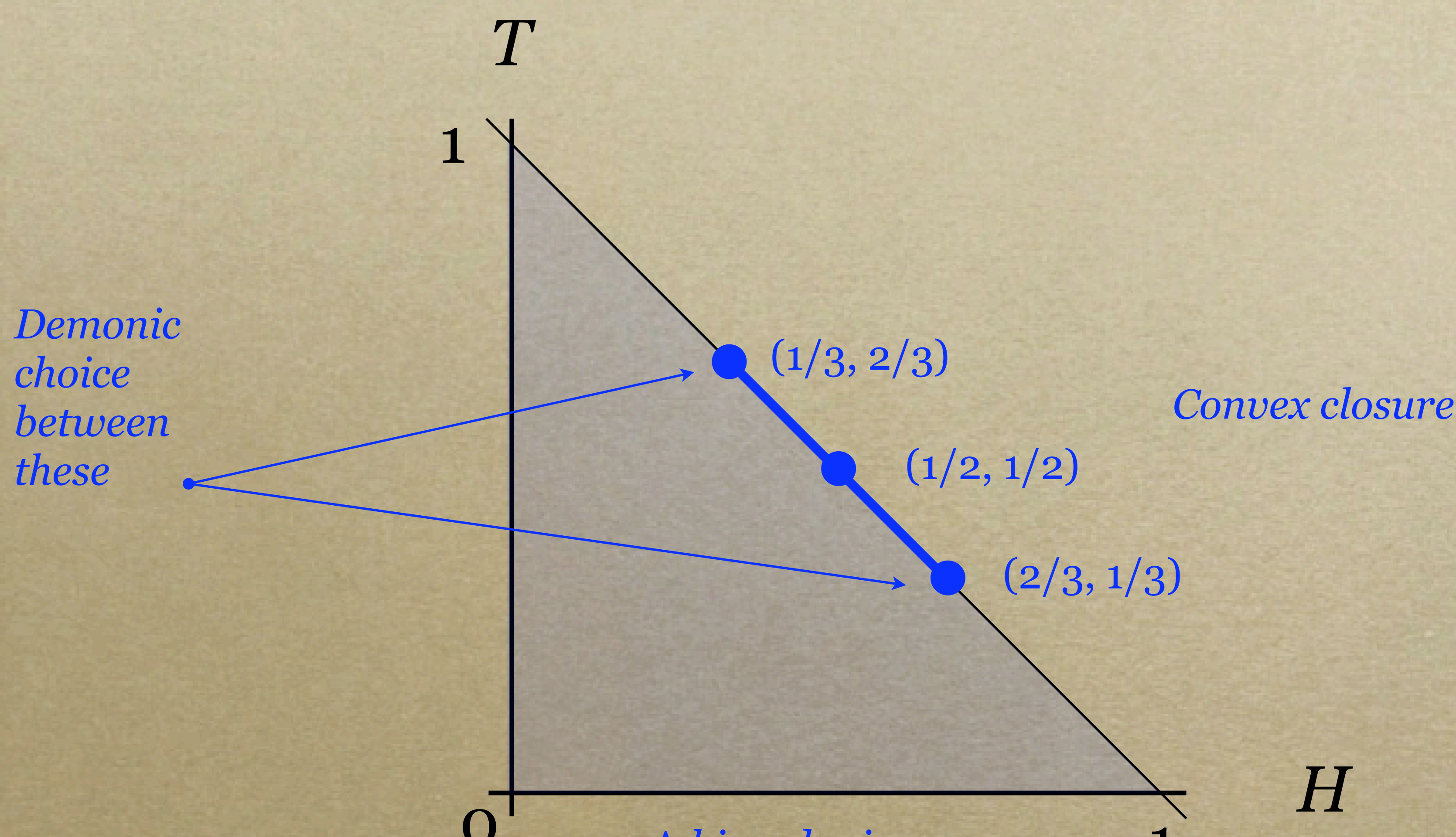


Interpretation: a geometric view



A tails-biased coin.

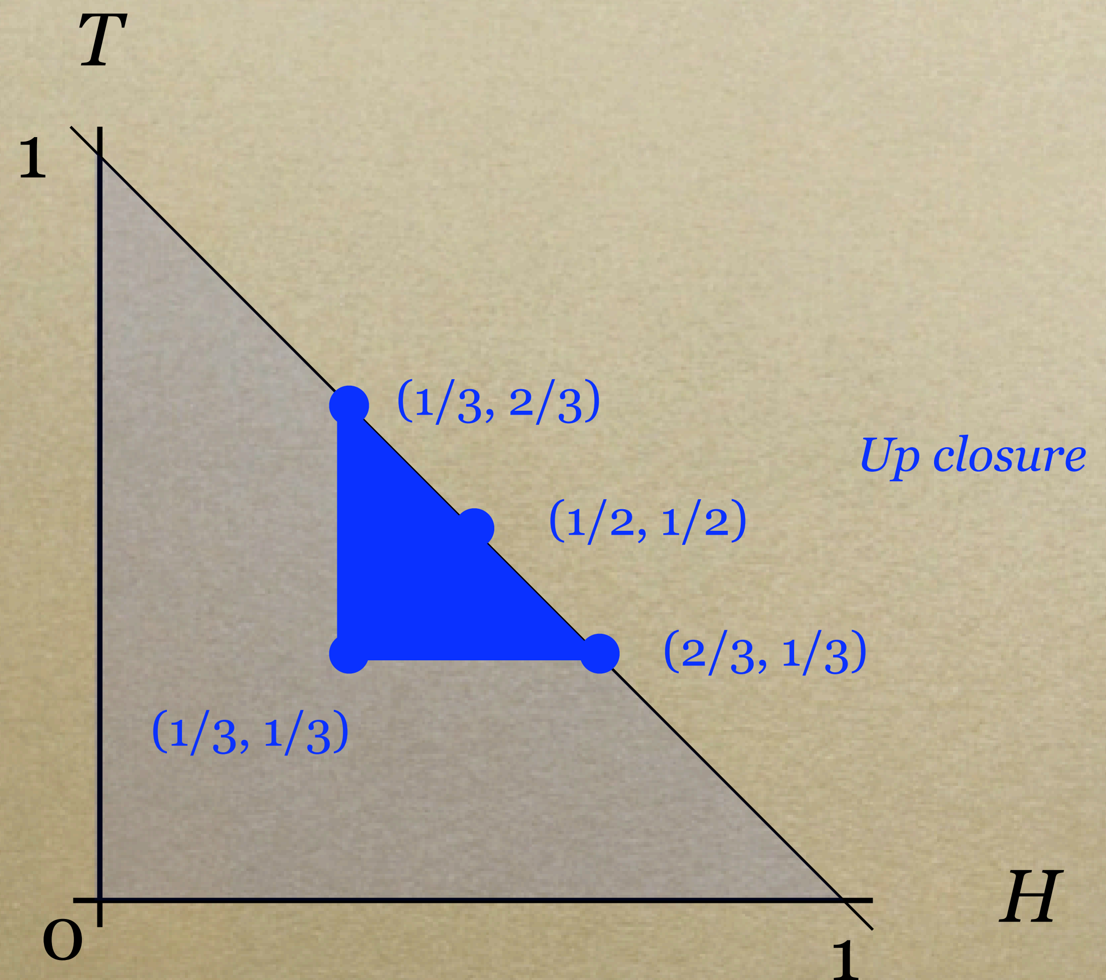
Interpretation: a geometric view



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

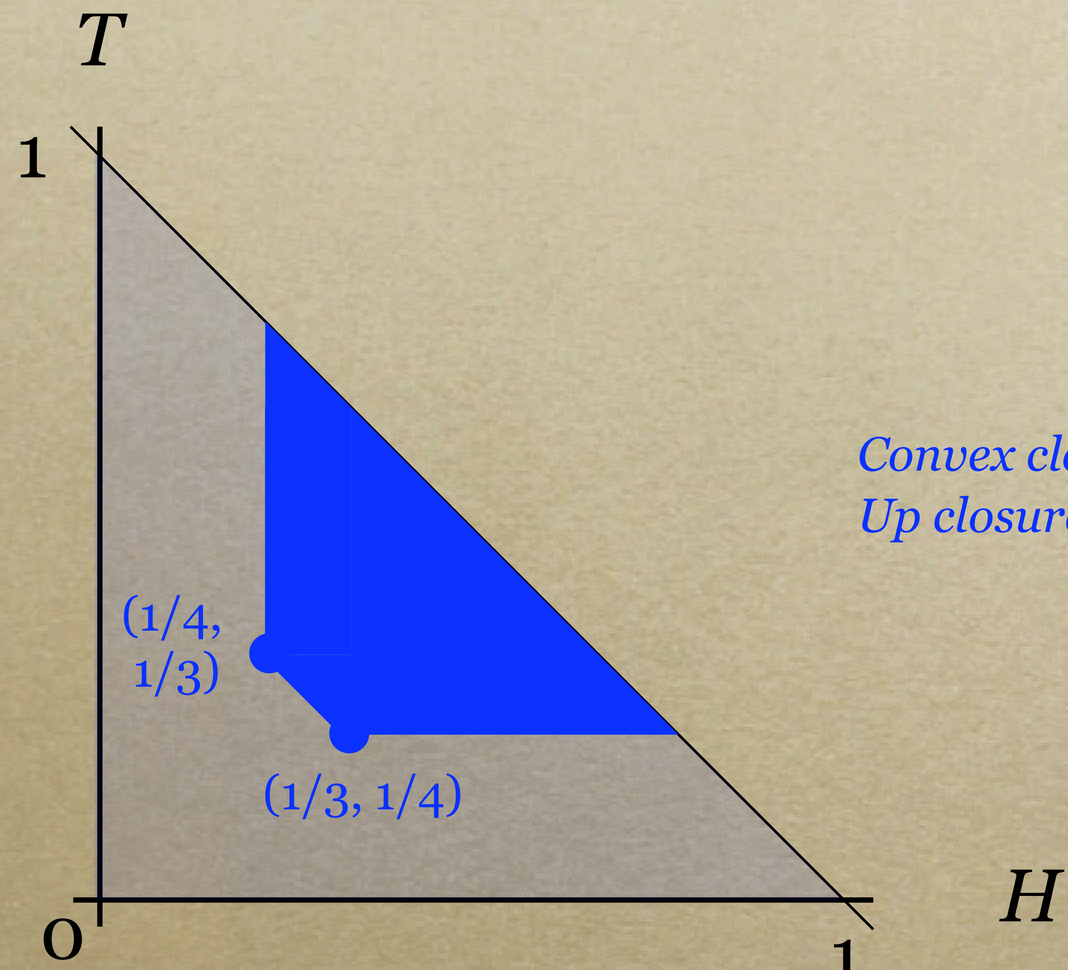
... one refinement of which
is an *unbiased* coin.

Interpretation: a geometric view



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

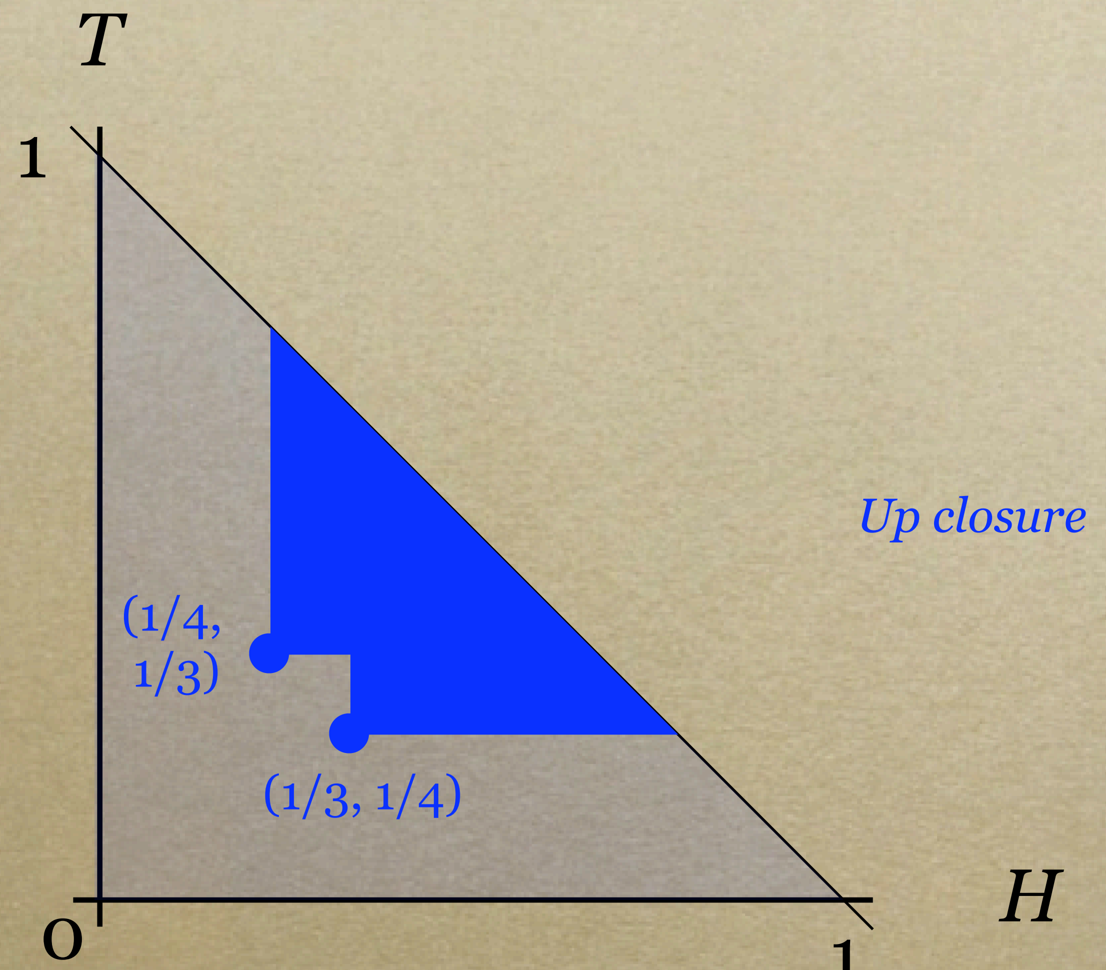
Interpretation: a geometric view



Convex closure
Up closure

Demonically, either of two possibly *nonterminating* coins.

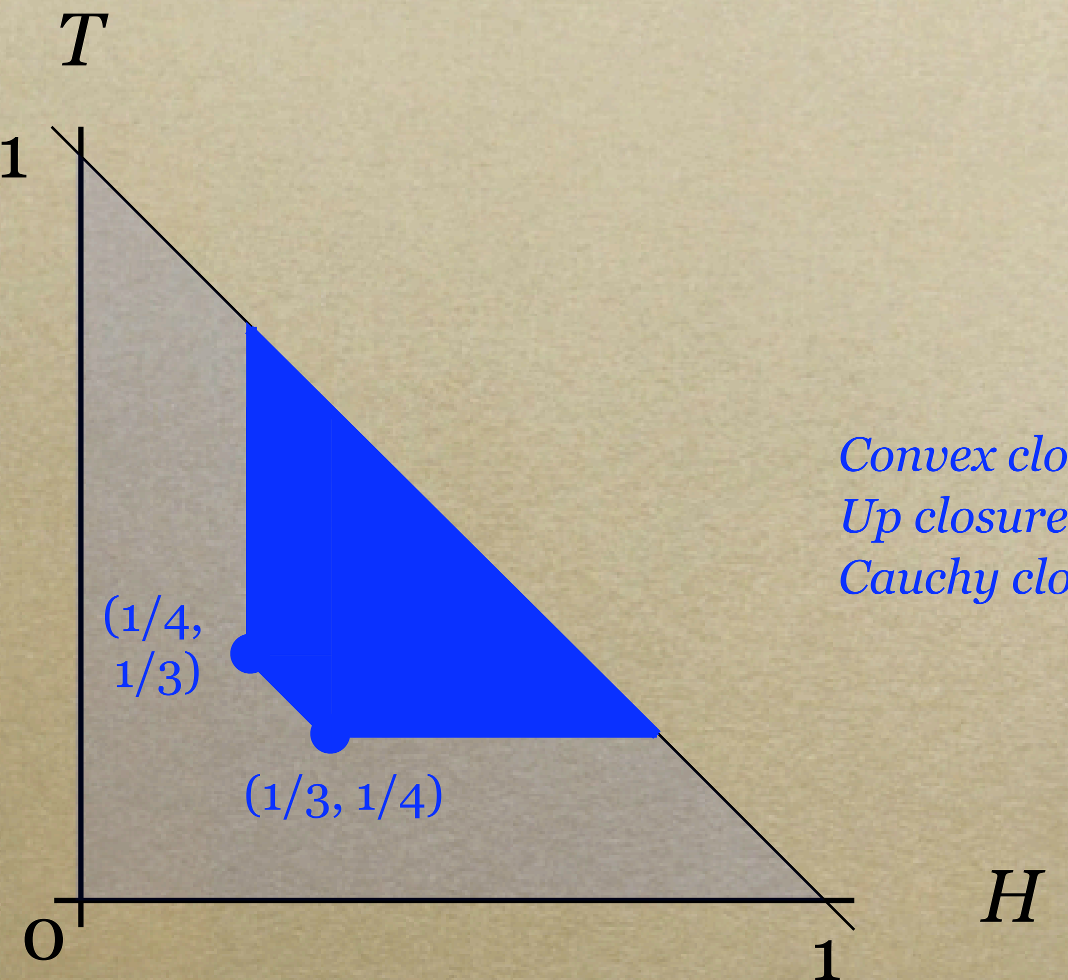
Interpretation: a geometric view



Demonically, either of two
possibly *nonterminating* coins.

Interpretation: a geometric view

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



Convex closure
Up closure
Cauchy closure

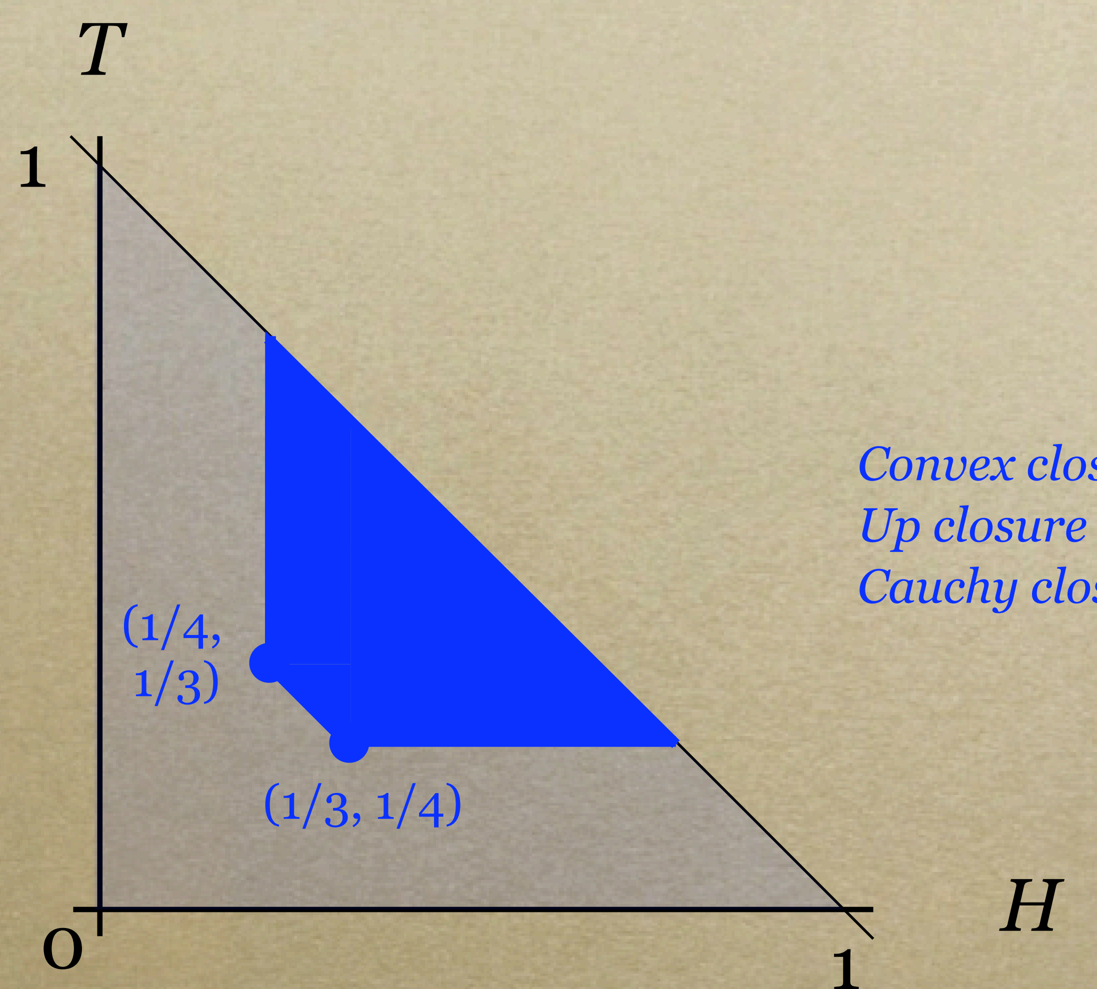
Demonically, either of two possibly *nonterminating* coins.

Interpretation: a geometric view

... but what's the connection with the programming logic?

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.

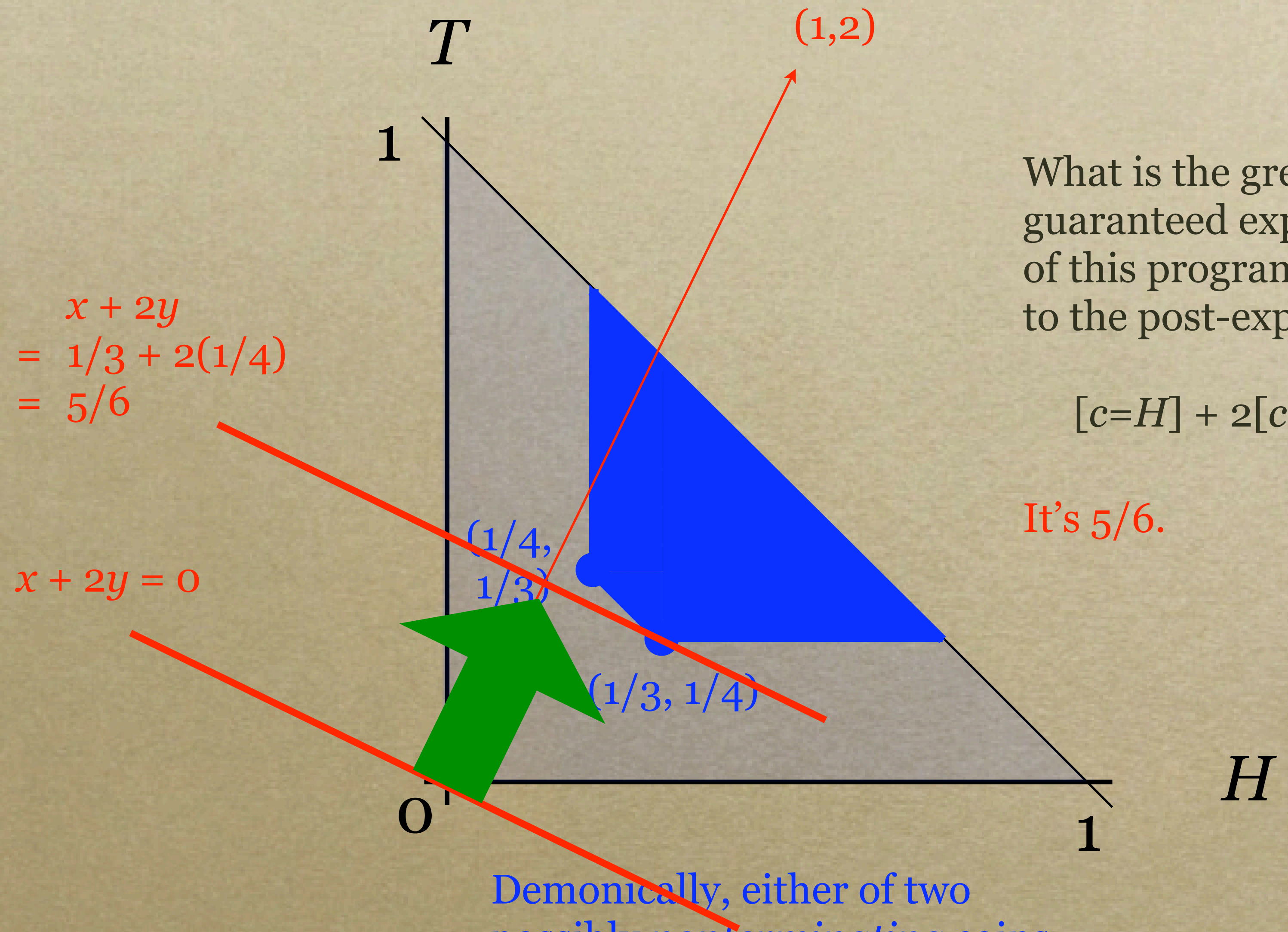


Convex closure
Up closure
Cauchy closure

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

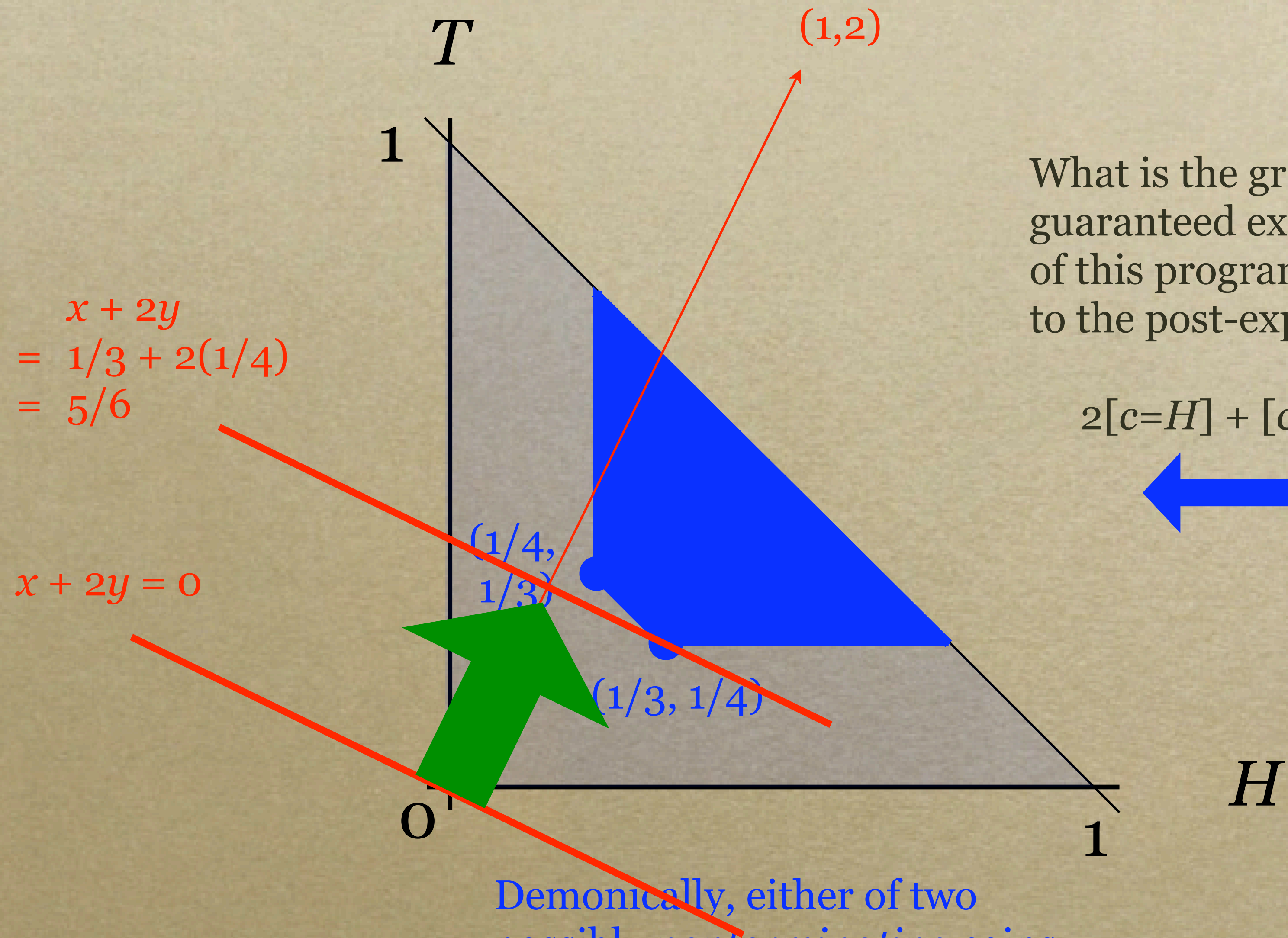
$[c{=}H] + 2[c{=}T]$?

It's 5/6.

Demonically, either of two possibly *nonterminating* coins.

Interpretation: a geometric view

... but what's the connection with the programming logic?



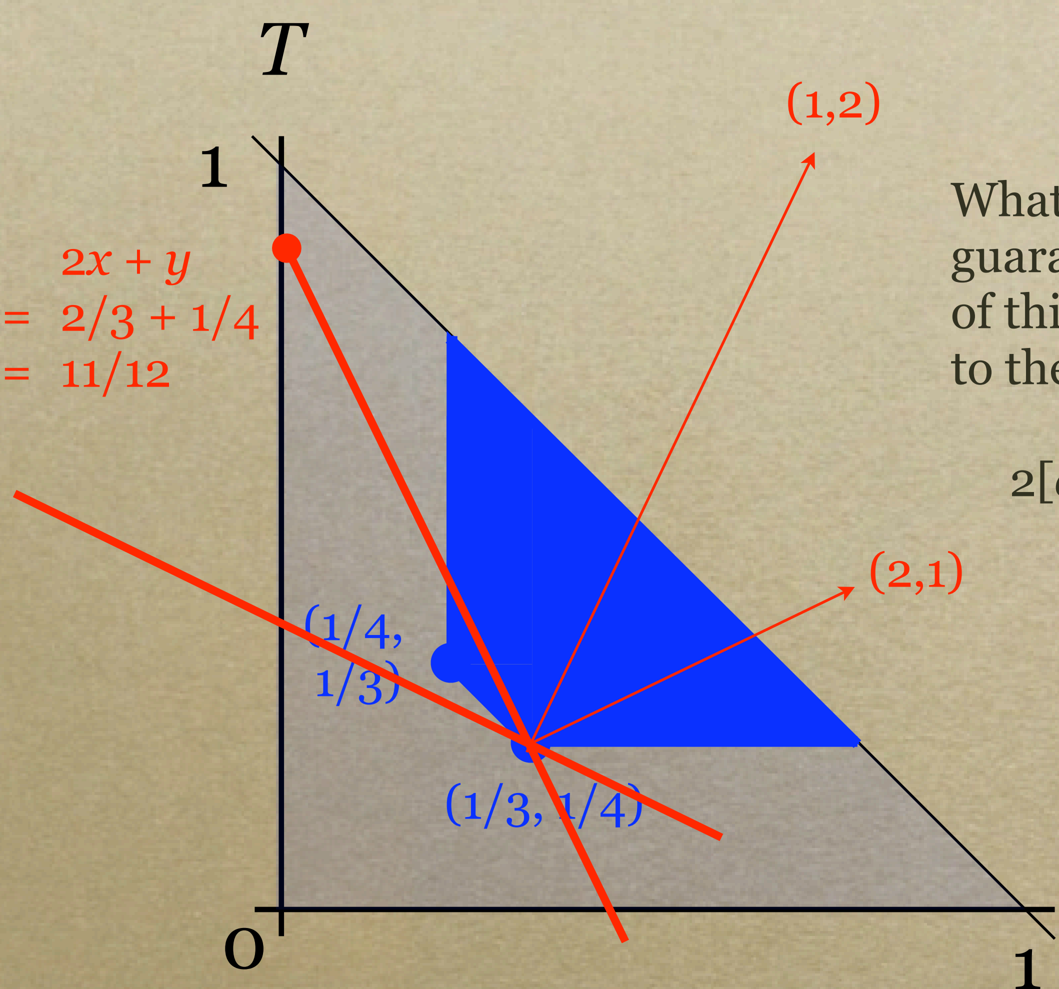
What is the greatest guaranteed expected value of this program with respect to the post-expectation

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

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

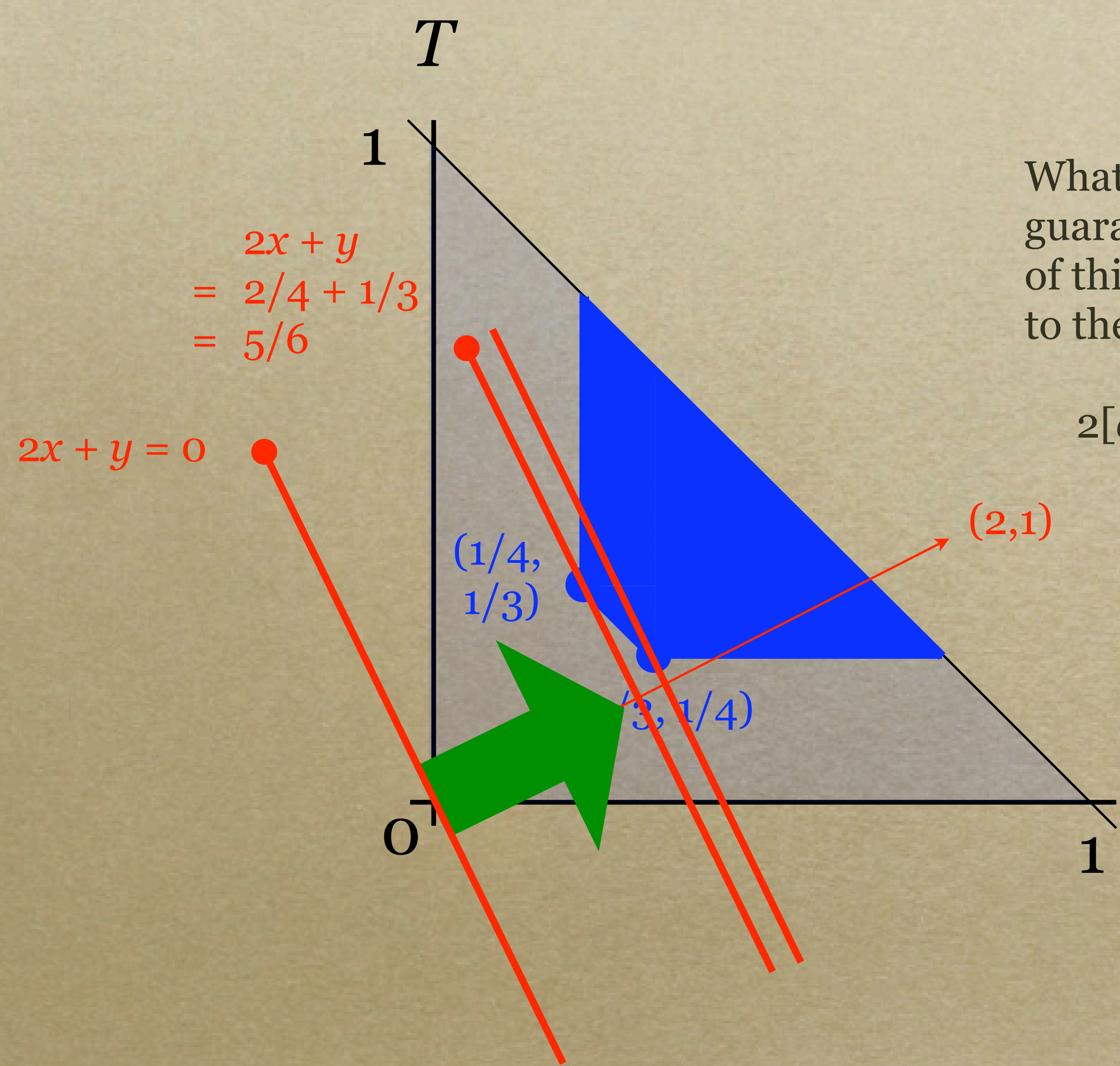
$2[c{=}H] + [c{=}T]$?

It's *not* 11/12.

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

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

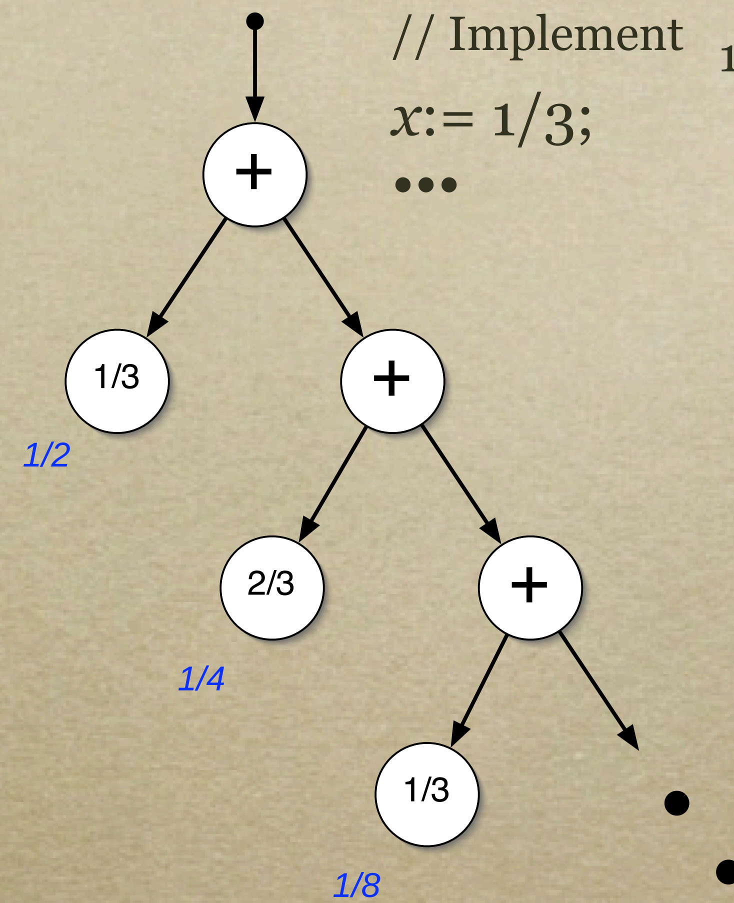
It's 5/6 again, because this time the demon goes to the other extreme.

Metatheorems for iteration

// Implement ${}_p\oplus$ using unbiased random bits only.

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

// Implement ${}_{1/3}\oplus$ …
x:= 1/3;
…



Metatheorems for iteration

// Implement ${}_p\oplus$ using unbiased random bits only.

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

e.g. /dev/random

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**

Metatheorems for iteration

$prog_1 \ {}_{p}\!\oplus \ prog_2$

is implemented by

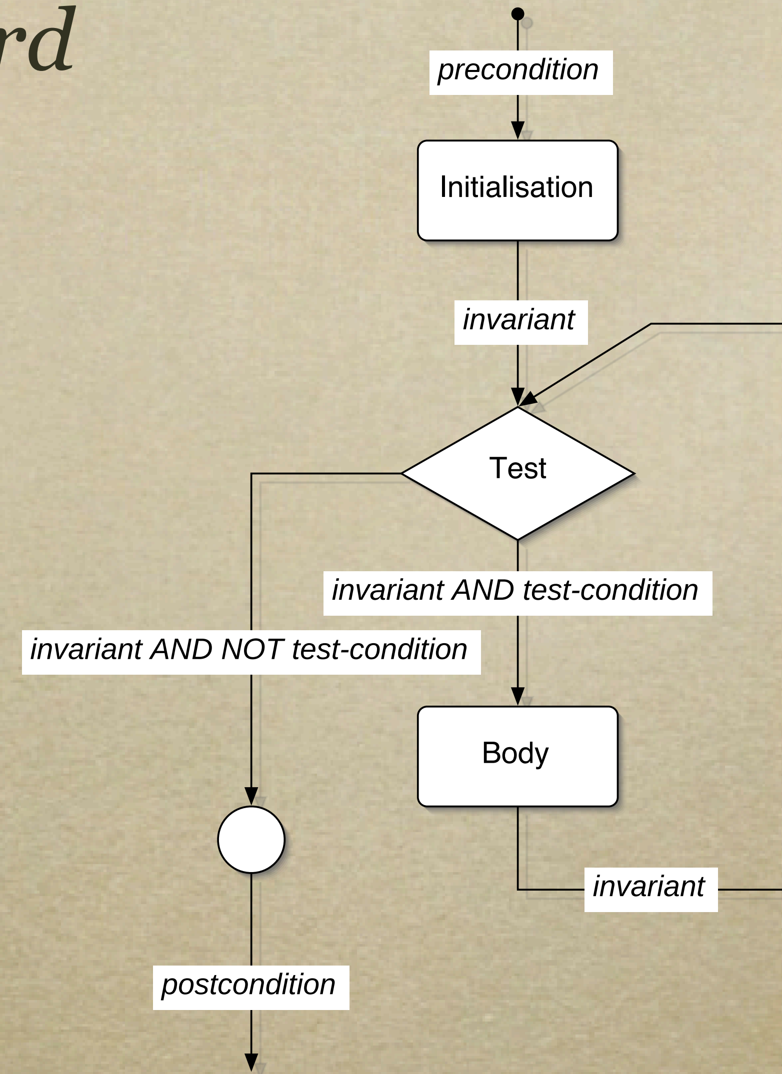
```
begin
  var x,b;

  x:= p;
  b:= true 1/2⊕ false;
  do  b →
    x:= 2x - [x ≥ 1/2];
    b:= true 1/2⊕ false;
  od;

  if x ≥ 1/2
    then prog1
    else prog2
  fi
end
```

and on the average uses only two random bits.

Iteration: reminder of standard metatheorems

$x, b, e := 1, B, E;$
do $e \neq 0 \rightarrow$
   if even e
      then $b, e := b^2,\ e \div 2$
      else $e, x := e-1,\ x \times b$
   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...

Standard metatheorems: invariants

$\{ B > 0$ and $E \geq 0 \}$
$x,b,e := 1,B,E;$
$\{ b > 0$ and $e \geq 0$ and $B^E = x \times b^e \}$
do $e \neq 0 \rightarrow$
$\{ \ldots$ and $e > 0 \}$
if even e

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

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

fi
$\{ B^E = x \times b^e \}$
od
$\{ x = B^E \}$

Standard metatheorems: invariants

$\{\ pre\ \}$
init;
$\{\ inv\ \}$
do G →
$\{\ G \wedge inv\ \}$
  body
  $\{\ inv\ \}$
od
$\{\ \neg G \wedge inv\ \}$

Probabilistic metatheorems: invariants again

$\{\, pre \,\}$

init;

$\{\, inv \,\}$

do $G \rightarrow$

$\{\, [G] \times inv \,\}$

body

$\{\, inv \,\}$

od

$\{\, [\neg G] \times inv \,\}$

$\{\, pre \,\}$

init;

$\{\, inv \,\}$

do $G \rightarrow$

$\{\, G \wedge inv \,\}$

body

$\{\, inv \,\}$

od

$\{\, \neg G \wedge inv \,\}$

Iteration: probabilistic example

$\{\ ?\ \}$

$x := p;$

$b :=$ true $_{1/2}\oplus$ false;

do $b \rightarrow$

$x := 2x - [x \geq 1/2];$

$b :=$ true $_{1/2}\oplus$ false;

od

$\{\ [x \geq 1/2]\ \}$

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

Example: iteration achieves its goal on termination

$[x \geq 1/2]$

$\Leftarrow \quad [\neg b] \times$

$(\, 2x - [x \geq 1/2] \; \boxed{\lhd} \; b \; \boxed{\rhd} \; [x \geq 1/2] \,)$

... **if** b **else** ...

$x := p;$

$b :=$ true $_{1/2}\oplus$ false;

do $b \rightarrow$

$\quad x := 2x - [x \geq 1/2];$

$\quad b :=$ true $_{1/2}\oplus$ false;

$\quad \{\, 2x - [x \geq 1/2] \lhd b \rhd [x \geq 1/2] \,\}$

od

$\{\, [x \geq 1/2] \,\}$

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

$$(2x - [x \geq 1/2] \lhd b \rhd [x \geq 1/2])$$

$$\boxed{\cdot} \equiv (2x)/2$$

$$\equiv x$$

$x := p;$

$b :=$ true $_{1/2}\oplus$ false;

do $b \rightarrow$

$x := 2x - [x \geq 1/2];$

$\{x\}$

$b :=$ true $_{1/2}\oplus$ false;

$\{2x - [x \geq 1/2] \lhd b \rhd [x \geq 1/2]\}$

od

$\{[x \geq 1/2]\}$

Example: iteration properly initialised

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

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

Example: correct overall

And finally we see that the pre-expectation overall…

is just p.

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

Example: summary

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.

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

Termination of probabilistic iterations

$\{ \mathit{inv} \}$
do $G \rightarrow$
$\quad \{ [G] \times \mathit{inv} \}$
$\quad \mathit{body}$
$\quad \{ \mathit{inv} \}$
od
$\{ [\neg G] \times \mathit{inv} \}$

$\{ \mathit{inv} \}$
do $G \rightarrow$
$\{ G \wedge \mathit{inv} \}$
$\quad \mathit{body}$
$\quad \{ \mathit{inv} \}$
od
$\{ \neg G \wedge \mathit{inv} \}$

In addition, show that $\mathit{inv} \Rrightarrow \mathit{term}$, where *term* is the probability of termination ...

... in which case the conclusion $\{ \mathit{inv} \}$ **do ••• od** $\{ [\neg G] \times \mathit{inv} \}$ expresses total — rather than just partial — correctness.

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

$$c := H \; {}_{1/2}\!\oplus \; T\,; \;\; d := H \; {}_{p}\!\oplus \; T \qquad \text{for arbitrary } p,$$

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

Ex. 5: Compositionality

Consider the two programs

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

B: $\quad (coin{:=}\ edge \sqcap coin{:=}\ heads)$

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

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 complification"?