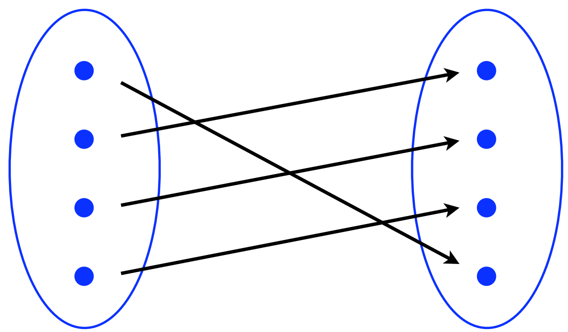
Formal Methods for Probabilistic Systems

Annabelle McIver
Carroll Morgan

- Source-level program logic
- Meta-theorems for loops
- Examples
- Relational operational model
  - Standard, deterministic, terminating .... *functions*
  - Standard, deterministic, non-terminating .... *functions with ⊥*
  - Standard, demonic, non-terminating .... *relations with ⊥*
  - Standard powerdomains .... *closure*
  - Probabilistic powerdomains .... *sub-distributions*
  - Demonic, probabilistic powerdomains .... *sets of...*
  - Examples .... *program geometry*

Standard, deterministic, terminating *relational semantics*



Standard, deterministic, terminating relational semantics

No probabilistic choice.

Output predictable from input.

No infinite loops or "run-time errors".

"Relates" initial states to final states, and so gives an "operational" view.

standard, deterministic, terminating...

A program f is a function of type state-to-state.

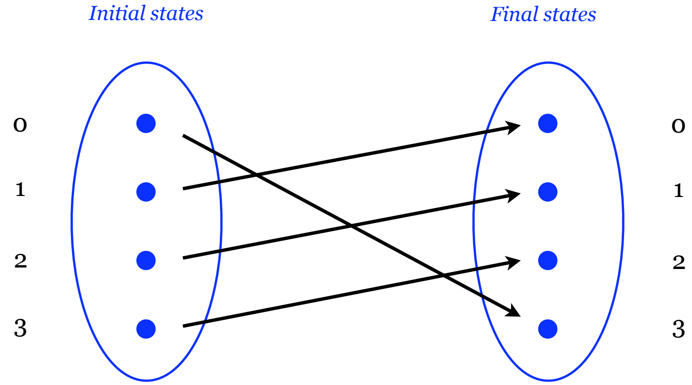
$$f\colon S \to S$$

Function f applied to initial state s gives final state s'.

$$f.s = s'$$

This is function application, that is f(s).

Standard, deterministic, terminating relational semantics



$$s\colon= (s-1)\ \mathbf{mod}\ 4$$ *syntax*

Function f, where $f.s = (s-1)\ \mathbf{mod}\ 4$. *relational semantics*

of type {0..3}→{0..3}

Standard, deterministic relational semantics

possibly nonterminating

A program f is now a function of type state to state-or-*bottom*.

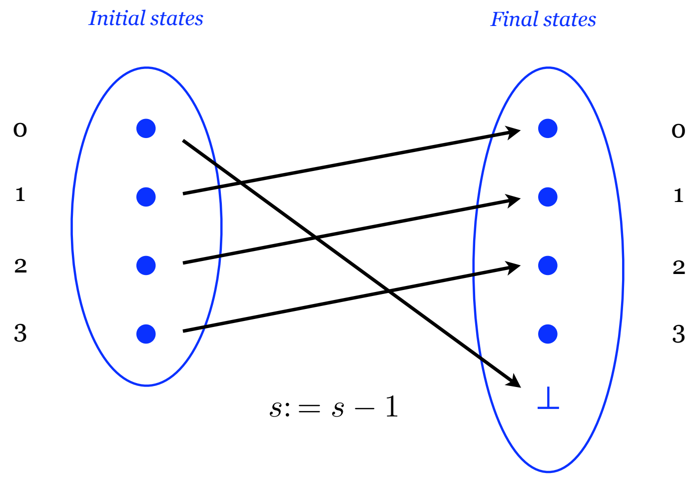
Function f applied to initial state s gives final state s'... or the special *nontermination* state $\perp$.

$$f : S \to S_\perp$$

($S_\perp$: $S \cup \{\perp\}$)

$$\begin{array}{lcll} f.s & = & s' & \text{if } f \text{ terminates, from } s\text{, at } s' \\ & = & \perp & \text{otherwise} \end{array}$$

Standard, deterministic relational semantics



$s := s - 1$

We suppose it is a "run-time error" to attempt to set s *to* -1.

Standard relational semantics

possibly nonterminating
possibly demonically nondeterministic

A program r is now a *relation* of type state to state-or-bottom.

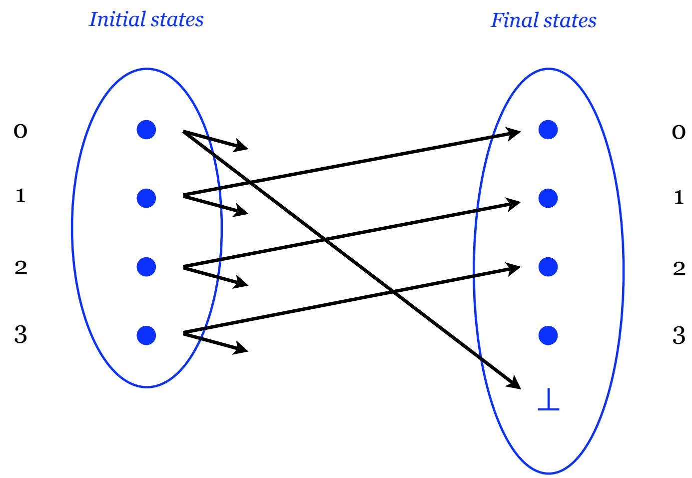
$$r : S \leftrightarrow S_\perp$$

Usually r is *total* ⟷ "Miracles" *are excluded.*
image-finite ⟷ Continuity *is required.*
and *up-closed.* ⟷ *If* r *can fail to terminate, then all (other) behaviours are deemed possible as well.*

$r.s.s'$ holds just when r can reach s' from s.

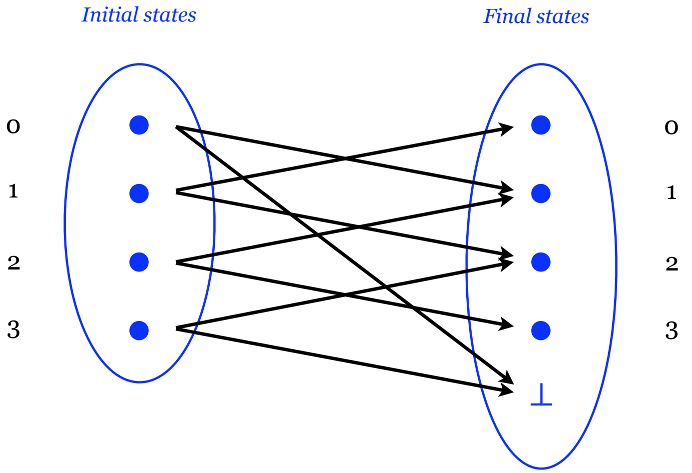
$r.s.s'$: *Boolean valued* — true *iff* $(s,s') \in r$.

Standard relational semantics



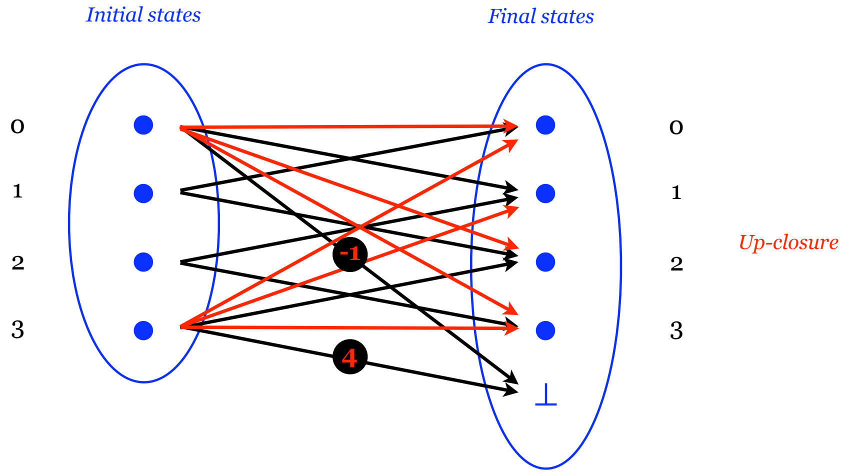
$s := s \pm 1$

Standard relational semantics



$s := s \pm 1$

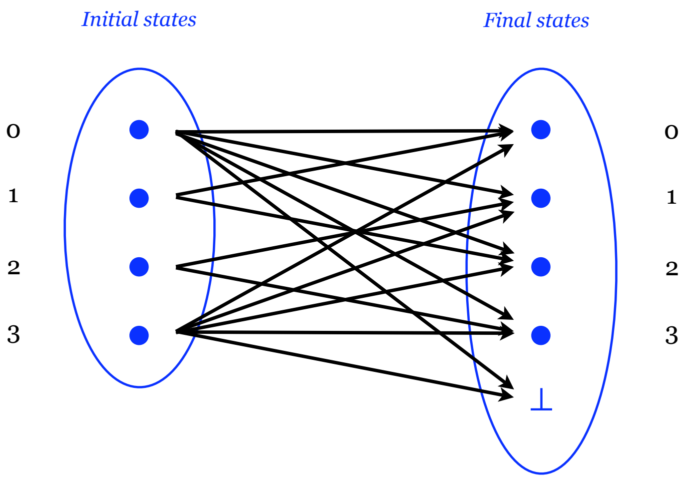
Standard relational semantics



$s := s \pm 1$

Remember that it is a "run-time error" to attempt to set s outside its type {0..3}.

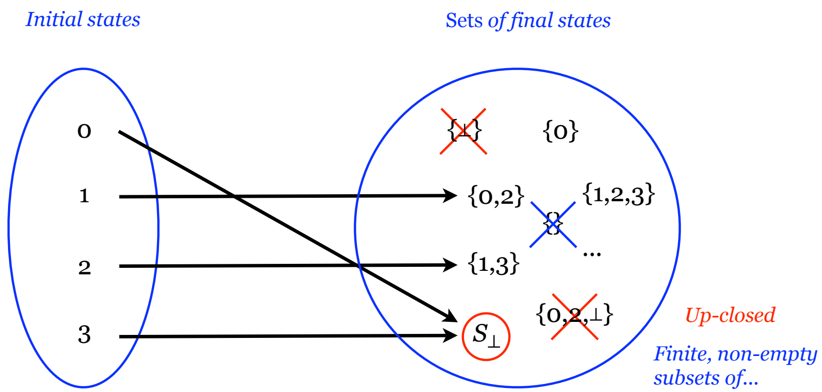
Standard relational semantics



Gets complicated...

$s := s \pm 1$ *...so we take an alternative view...*

Standard relational semantics

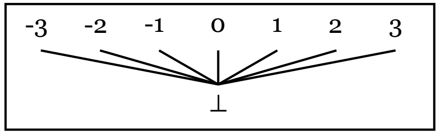


$s := s \pm 1$

A program *r* is a *relation* of type state to state-or-bottom... or equivalently a set-valued *function*.

$r\colon S \leftrightarrow S_\perp$
$r\colon S \rightarrow \mathbb{P}S_\perp$
$r\colon S \rightarrow \mathbb{F}^+S_\perp$

The significance of up-closure



A flat domain, based on the integers; for example $\bot \sqsubseteq 3$.

a proper outcome — *nontermination*

Informally, we regard 3 *as a "better" outcome than* ⊥. And we regard a program f_2 that delivers consistently better results than some other program f_1 as a "better" program overall:

If $(\forall s \bullet f_1.s \sqsubseteq f_2.s)$ then we say $f_1 \sqsubseteq f_2$

Program f_1 is *refined* by f_2 if some of f_1's nontermination is replaced by *proper outcomes* in f_2.

This in effect "promotes" the order ⊑ from an order on individual values to an order on functions resulting in those values.

For nondeterminism we seek a similar promotion, but this time to the *sets* of values that represent the demonic choice.

The significance of up-closure: *powerdomains*

Given two relational programs r_1 and r_2, we say that $r_1 \sqsubseteq r_2$ iff, for all initial states s, any outcome in the set $r_2.s$ can be justified by some outcome in the set $r_1.s$, that is *if every behaviour of the implementation* r_2 *is justified by the specification* r_1:

for all s, and $s_2 \in r_2.s$, there is an $s_1 \in r_1.s$ such that $s_1 \sqsubseteq s_2$

That is, if $r_2.s.s_2$ *holds, or equivalently* $(s,s_2) \in r_2$.

This is known as the Smyth *order.*

For subsets S_1, S_2 of the state space S, we say that $S_1 \sqsubseteq S_2$ iff

$$(\forall s_2 \mid s_2 \in S_2 \cdot (\exists s_1 \mid s_1 \in S_1 \cdot s_1 \sqsubseteq s_2)) .$$

The refinement order between relations is just the Smyth order on result-sets, lifted functionally as we saw before.

MB Smyth. *Power domains*. Jnl. Comp. Sys. Sci. 16: 23-36, 1978.

The significance of up-closure: *equivalence classes*

The Smyth *pre-order*.

For subsets S_1, S_2 of the state space S, we say that $S_1 \sqsubseteq S_2$ iff

$$(\forall s_2 \mid s_2 \in S_2 \cdot (\exists s_1 \mid s_1 \in S_1 \cdot s_1 \sqsubseteq s_2)) .$$

Notice that both $\{\bot, 2\} \sqsubseteq \{\bot, 3\}$ and $\{\bot, 3\} \sqsubseteq \{\bot, 2\}$. This means that ⊑ is not a partial order — rather it is a *pre-order*, satisfying reflexivity and transitivity *but not anti-symmetry*. We say that $\{\bot, 2\} \cong \{\bot, 3\}$, that they are *equivalent*.

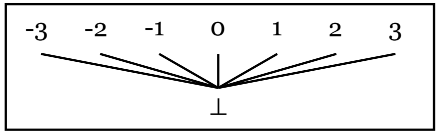
This is a nuisance, but a well known one with a well known solution: we form equivalence classes, and take a distinguished representative from each class. That representative happens to be the *up-closure* — that is, for subset S_1 of S, the set

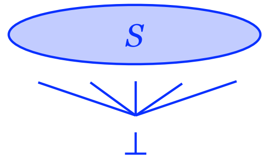
$$S_1\!\uparrow \quad \text{defined} \quad \{ s \mid s \in S \bullet (\exists s_1 \mid s_1 \in S_1 \bullet s_1 \sqsubseteq s) \}$$

Not only does this have the property $S_1 \cong S_2$ iff $S_1\!\uparrow = S_2\!\uparrow$, as we would expect from the equivalence-class-representative construction, but we have also that

$$S_1 \sqsubseteq S_2 \quad \text{iff} \quad S_1\!\uparrow \supseteq S_2\!\uparrow$$

Up-closure for a flat domain

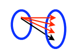




In the *flat domain* $S_\bot$ the definition of up-closure is particularly simple; it is

$$\begin{aligned} S_1\!\uparrow &= S_1 && \text{if } \bot \notin S_1 \\ &= S_\bot && \text{otherwise.} \end{aligned}$$

If the computation can reach ⊥, then we deem that it can reach every other state as well.



Although we have gone a long way to justify this easy construction, it is reassuring to know that it fits in with the general theory — and that will make things work much more smoothly later on.

Morgan's Rule: *If you're going to re-invent the wheel... at least make sure it's round.*

A probabilistic powerdomain

The powerdomain construction we have just seen took an underlying set of *values*, with a partial order representing "refinement", and from it constructed — in a very general way — a partial order over *sets* of those values, one which can be used to describe demonically nondeterministic programs.

A similar construction — though more complex — takes the underlying set of values, with its refinement order, to a partial order over *distributions* on those values.

That is what we shall use.

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.

A probabilistic powerdomain over a flat structure

- Take our underlying space $S_\perp$, with its "flat" *information order*, and generate the *Scott topology* on it.
- Carry on by generating the space of probabilistic *evaluations* over that topology.
- Then notice that the result is *isomorphic* to

  $\{ \Delta : \blacksquare \to [0,1] \mid (\Sigma_{s \in S} \ \Delta.s) \leq 1 \}$ ⟵ *Notice we do not need to refer to ⊥ explicitly.*

  with the order

  $\Delta_1 \sqsubseteq \Delta_2$ iff $\Delta_1.s \leq \Delta_2.s$ for all $s \in S$.

- These are called *discrete sub-probability measures*.

G Gierz et al. *A Compendium of Continuous Lattices*. Springer Verlag, 1980.

Jones and Plotkin. *Op. cit.*

(for example) D Kozen. *Semantics of probabilistic programs*.
Jnl. Comp. Sys. Sci. 22:328-350, 1981.

Discrete sub-probability measures

A *discrete* probability distribution assigns probabilities to individual points, *e.g.* the function $\{ H \mapsto 1/2, T \mapsto 1/2 \}$ that describes flipping an unbiased coin.

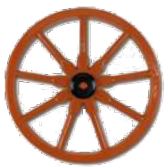
It gains the prefix "sub-" if it is not required to sum to one, as in the distribution $\{ H \mapsto 1/3, T \mapsto 1/3 \}$ for a coin that "might not terminate"— this implicitly includes a probability 1 - 1/3 - 1/3 = 1/3 of nontermination ⊥.

Such a coin is *refined* by another that terminates at least as often; and any extra termination can be assigned to either proper outcome; for example, we have

$$\{ H \mapsto 1/3, T \mapsto 1/3 \} \quad \sqsubseteq \quad \{ H \mapsto 2/5, T \mapsto 1/2 \}$$

in which the right-hand coin refines the left-hand coin, but still has probability 1/10 of failing to terminate.

Again we have used theoretical tools (information orders, Scott topology, Jones/Plotkin evaluations...) that in the end (via isomorphism) have produced something quite simple.



We have thus ensured that the wheel is "round" — and so it rolls very nicely!

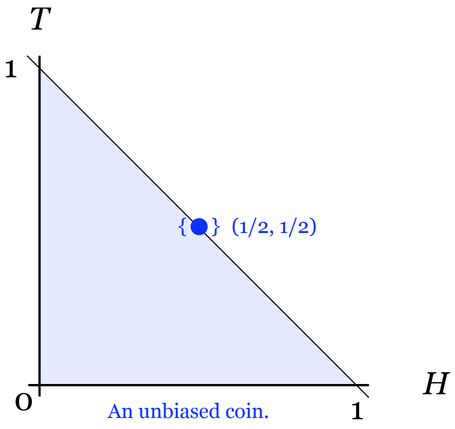
A demonic powerdomain over discrete sub-probability measures

To have both *demonic* and *probabilistic* choice available to us, we take the probabilistic powerdomain we have just constructed — discrete sub-probability measures — and apply our earlier "up-closure of sets" construction; the latter will add demonic choice, as it did before, but this time to elements that already model probability.

The flat domain over state space *S*.	$S_\perp$	
The probabilistic powerdomain over $S_\perp$.	$\overline{S}$	Discrete sub-probability measures.
Sets of discrete sub-probability measures, for demonic choice.	$\mathbb{P}\overline{S}$	Then up closure, *convex closure, Cauchy closure ?*
Closed sets of discrete sub-probability measures, for refinement.	$\mathbb{C}S$	$\mathbb{C}S \subseteq \mathbb{P}\overline{S}$
The relational model of demonic, probabilistic programs.	$S \to \mathbb{C}S$	$\mathbb{H}S$

A brief tour of ℂS

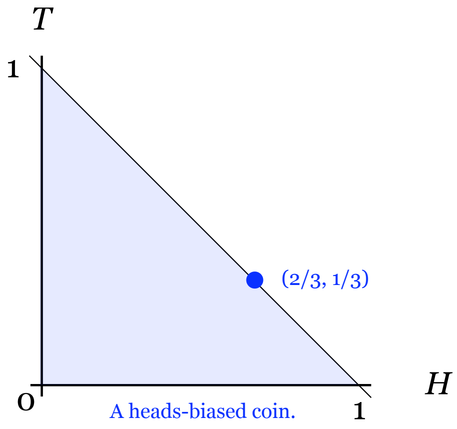
when S *is the two-element space* { *H*, *T* } *of coin-flip results*



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

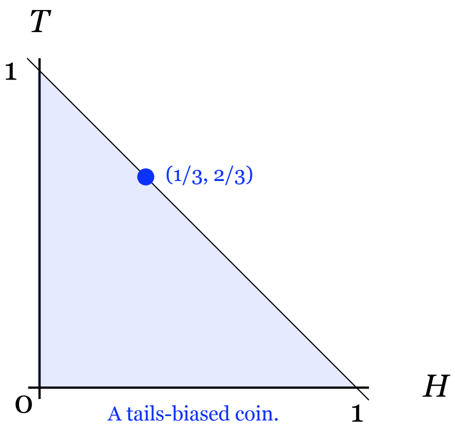
A brief tour of ℂS

when S *is the two-element space* { *H*, *T* } *of coin-flip results*



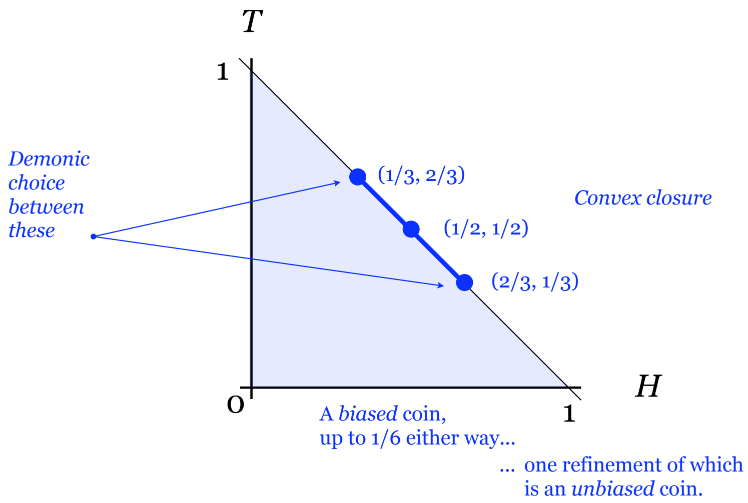
A brief tour of ℂS

when S *is the two-element space* { *H*, *T* } *of coin-flip results*



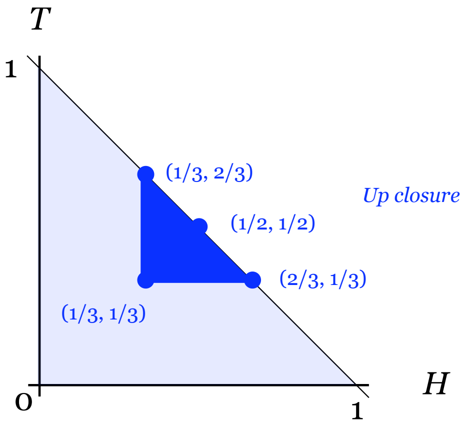
A brief tour of ℂS

when S *is the two-element space* { *H*, *T* } *of coin-flip results*



A brief tour of $\mathbb{C}$S

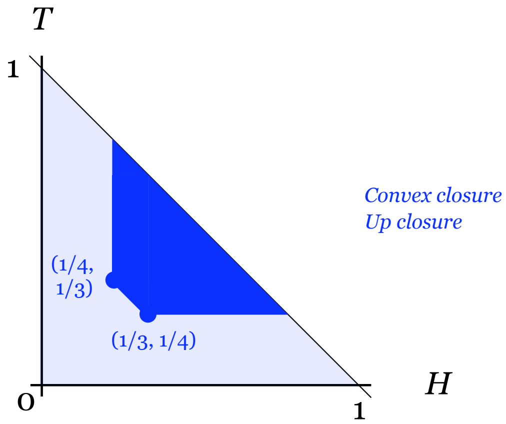
when S *is the two-element space* { *H*, *T* } *of coin-flip results*



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

A brief tour of $\mathbb{C}$S

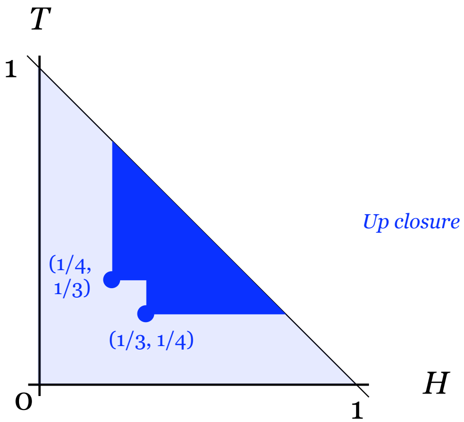
when S *is the two-element space* { *H*, *T* } *of coin-flip results*



Demonically, either of two possibly *nonterminating* coins.

A brief tour of $\mathbb{C}$S

when S *is the two-element space* { *H*, *T* } *of coin-flip results*



Demonically, either of two possibly *nonterminating* coins.

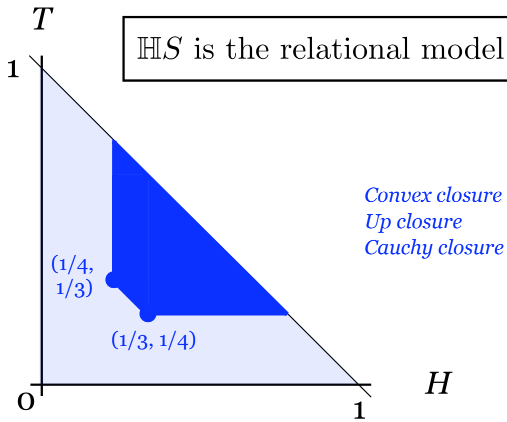
A brief tour of $\mathbb{C}$S

when S *is the two-element space* { *H*, *T* } *of coin-flip results*

$\mathbb{H}S$ is the relational model $S \to \mathbb{C}S$

$\mathbb{H}S$ for Jifeng $\mathbb{H}$e

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

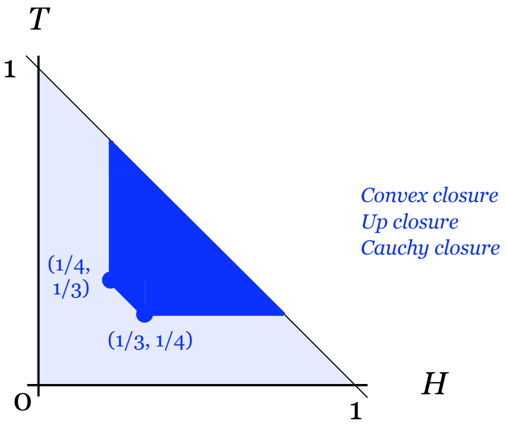


Demonically, either of two possibly *nonterminating* coins.

A brief tour of ℂS *concluded* ... *but what's the connection with the programming logic?*

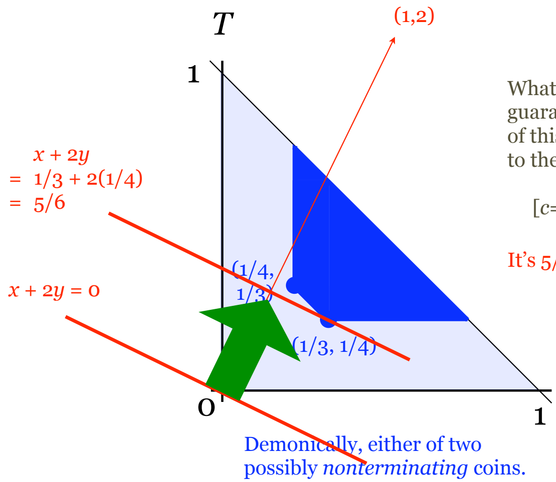
He, McIver and Seidel. *Probabilistic models for the guarded command language*. Sci. Comp. Prog. 28:171-192, 1997.

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



Demonically, either of two possibly *nonterminating* coins.

A brief tour of ℂS *concluded* ... *but what's the connection with the programming logic?*



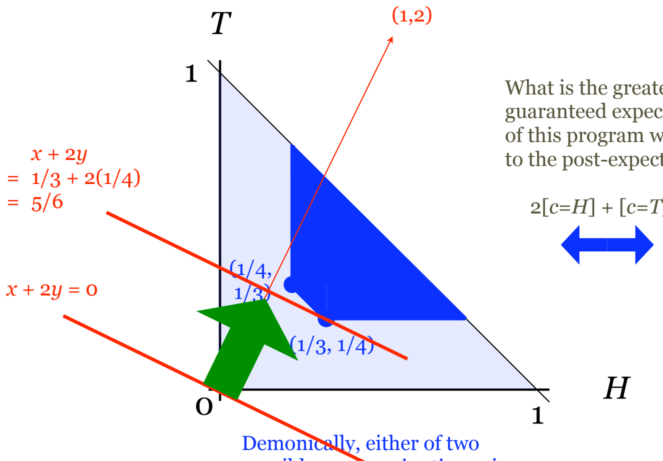
What is the greatest guaranteed expected value of this program with respect to the post-expectation

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

It's 5/6.

Demonically, either of two possibly *nonterminating* coins.

A brief tour of ℂS *concluded* ... *but what's the connection with the programming logic?*

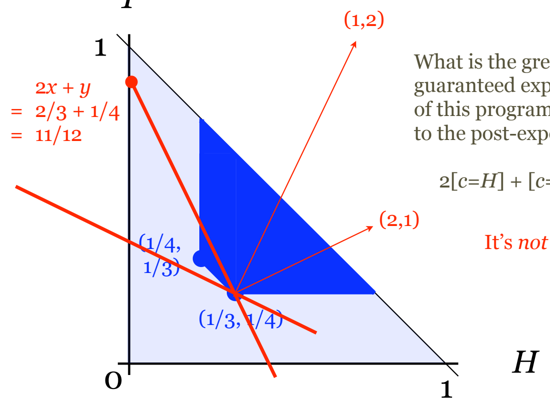


What is the greatest guaranteed expected value of this program with respect to the post-expectation

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

Demonically, either of two possibly *nonterminating* coins.

A brief tour of ℂS *concluded* ... *but what's the connection with the programming logic?*



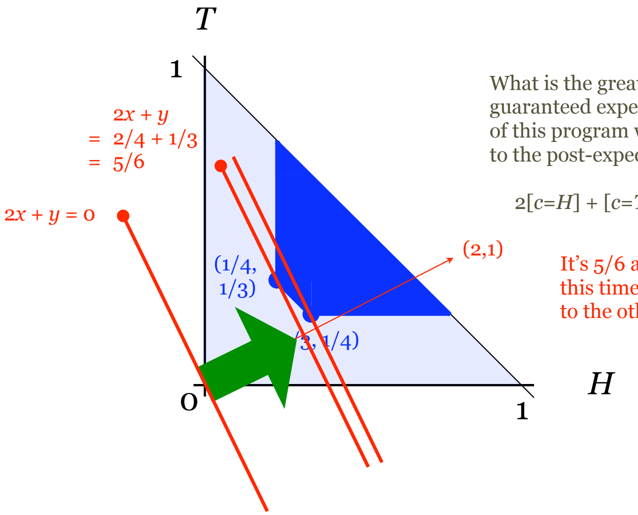
What is the greatest guaranteed expected value of this program with respect to the post-expectation

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

It's *not* 11/12.

Demonically, either of two possibly *nonterminating* coins.

A brief tour of ℂS concluded … *but what's the connection with the programming logic?*



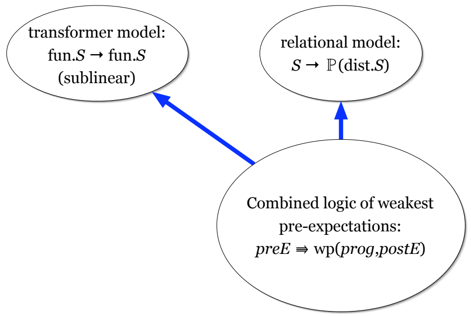
What is the greatest guaranteed expected value of this program with respect to the post-expectation

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

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

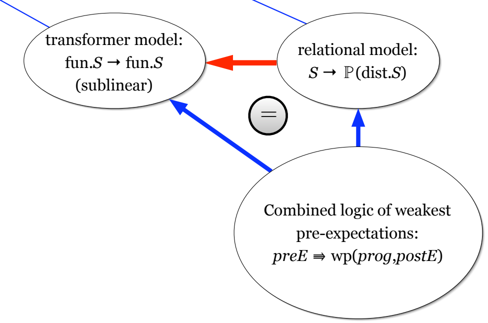
The two semantics are congruent

$preE \Rrightarrow$ wp.$prog$.$postE$
means
$(\forall s{:}S; \sigma{:}[\![prog]\!].s \bullet [\![preE]\!].s \leq \mathrm{exp}_{\sigma}.[\![postE]\!])$



The two semantics are congruent

wp.$prog$.$postE$.s
means
$(\sqcap \sigma{:}[\![prog]\!].s \bullet [\![preE]\!].s \leq \mathrm{exp}_{\sigma}.[\![postE]\!])$



Exercises

Ex. 1: *Experiment with the geometric presentation of the transformer semantics: what happens to the demonic coin with post-expectation*

just [c=H] ?
just [c=T] ?
just [true] ?

(The direction numbers for [true] *are (1,1), i.e. the grazing-line approaches at 45°. Which distribution-point does it touch first?)*

Ex. 2: *Why isn't the answer to the third item above just 1 again?*