Deriving Probabilistic Semantics

Objective

To construct a probabilistic model for an imperative language from the standard relational model using weakest completion method.

From a Basic theory to a Probabilistic one

(1) We extend the base type of the model by adding probabilistic information, and as part of that we give an explicit link between the orginal type and its probabilistic extension.
(2) From that link between the base and the extended type we then attempt to induce an embedding of *programs* over the base type.

Healthiness Conditions

The final stage is to examine the image, in the extended space of programs, of the original (probability-free) model and to determine what algebraic characteristics define it. Such characteristics, often called "healthiness conditions, both reflect the behaviour of "real" programs and allow us to formulate and prove further algebraic laws that are of practical use in program derivation.

Probabilistic program syntax

$P ::=$	$\bot$	primitive aborting program
	II	'skip' program
	$x := e$	assignment
	$P \lhd b \rhd P$	conditional
	$P \sqcap P$	demonic choice
	$P \,{}_r\!\oplus\, P$	probabilistic choice
	$P ; P$	sequential composition
	$(\mu X \bullet P(X))$	recursion

For ease of exposition we have included an explicit demonic choice operator $\sqcap$ and used a conventional 'if-then-else' conditional.

Relational semantics

A standard sequential program starts its execution in an initial state, and terminates (if it ever does) in one of a set of final states. We give the relational meaning of a program by a pair of predicates p, R, *design*, with this syntax and interpretation:

$$p(s) \vdash R(s, s') \quad =_{\text{df}} \quad (ok \wedge p) \Rightarrow (ok' \wedge R)$$

$$
\begin{aligned}
\bot &=_{\text{df}} \mathbf{false} \vdash \mathbf{true} \\
II &=_{\text{df}} \mathbf{true} \vdash (s' = s) \\
x := e &=_{\text{df}} \mathbf{true} \vdash (s' = s[e/x]) \\
P \sqcap Q &=_{\text{df}} P \vee Q \\
P \lhd b \rhd Q &=_{\text{df}} (b \wedge P) \vee (\neg b \wedge Q) \\
P ; Q &=_{\text{df}} P \circ Q \\
(\mu X \bullet P(X)) &=_{\text{df}} \bigvee \{Q \mid \forall ok, s, ok', s' \bullet (Q \Rightarrow P(Q))\}
\end{aligned}
$$

where $\circ$ stands for the (relational) composition.

Probabilistic semantics

We extend our standard states to probabilsitic states by replacing the final state s' with a final *distribution* which we call $prob'$. Maintaining the structure of designs in other respects, we say that a probabilistic program P can be identified as a design with ok, s, ok' and $prob'$ as free variables.

Definition (Probabilistic distributions and designs)
Let S be a state space. The set of distributions over S is the set of total functions from S into the closed interval of reals $[0, 1]$,

$$PROB \quad =_{\text{df}} \quad S \to [0, 1]$$

We insist further that for any member $prob$ of $PROB$ the probabilities must sum to 1:

$$\Sigma_{s \in S}\ prob(s)\ =\ 1\ .$$

For any subset X of S we define

$$prob(X)\ \ =_{\text{df}}\ \ \Sigma_{s \in X}\, prob(s)\ .$$

We use $\mathcal{P}$ to denote the set of *probabilistic designs* $p(s) \vdash R(s, prob')$.

Retraction

Definition (Retraction of states)
The design ρ relates a probabilistic state *prob* to a set of standard states s' according to the definition

$$\rho(ok, prob, ok', s') \quad =_{\text{df}} \quad \mathbf{true} \vdash prob(s') > 0 \ .$$

Note that we have expressed the retraction — though not properly a computation — in the design notation as well, complete with its implicit *ok* variables; that allows us to use design composition in the construction of our sub-commuting diagram below, from which we extract the embedding of designs.

Embedding

For any standard design D, we identify its embedding $\mathcal{K}(D)$, which will be a probabilistic design, as the *weakest solution* in X of the equation

$$X \circ \rho \quad = \quad D \ .$$

Thus define

$$\mathcal{K}(D) \ =_{\text{df}} \ D/\rho \ ,$$

where D/ρ is the *weakest pre-specification* of D through ρ

$$(X \circ \rho) \sqsupseteq D \qquad \textbf{iff} \qquad X \sqsupseteq D/\rho,$$

Properties of Embedding

Theorem

$$\mathcal{K}(p(s) \vdash R(s, s')) \quad = \quad p(s) \vdash (prob'(\{t \mid R(s,\, t)\}) = 1)$$

Theorem

$$\mathcal{K}(D) \circ \rho \quad = \quad D$$

Preservation of program structure

Theorem (Primitive probabilistic designs)

(1) $\mathcal{K}(\bot) \quad = \quad \bot.$

(2) $\mathcal{K}(x := e) \quad = \quad \mathbf{true} \vdash prob'(s[e/x]) = 1.$

Theorem (Conditional choice)

$\mathcal{K}(D0 \lhd b \rhd D1) \quad = \quad \mathcal{K}(D0) \lhd b \rhd \mathcal{K}(D1)$

Demonic choice

Theorem

$\mathcal{K}(D0 \sqcap D1) \quad =$

$\mathcal{K}(D0) \vee \mathcal{K}(D1) \vee \bigvee_{0<r<1}(\mathcal{K}(D0) \|_{M_r} \mathcal{K}(D1))$

where M_r is a 'coupling predicate'

$$(p0 \vdash R0) \|_{M_r} (p1 \vdash R1)$$

$$=_{\text{df}} \quad ((p0 \wedge p1) \vdash (R0(s, 0.prob') \wedge R1(s, 1.prob')) \circ M_r$$

and $M_r \quad =_{\text{df}} \quad \mathbf{true} \vdash prob' = r \times 0.prob + (1 - r) \times 1.prob.$

Sequential composition

For sequential composition we introduce $\uparrow$ to lift a design taking (ok, s) to $(ok', prob')$ to a design taking $(ok, prob)$ to $(ok', prob')$.

$$\uparrow (p \vdash R) =_{df}$$

$$(prob(p) = 1) \vdash \left(\begin{array}{l} \exists Q \in (S \to PROB) \bullet \\ \forall s \bullet prob(s) > 0 \Rightarrow R(s, Q(s)) \wedge \\ prob' = \Sigma_{t \in S}(prob(t) \times Q(t)) \end{array} \right)$$

Theorem

$$\mathcal{K}(D0; D1) \quad = \quad \mathcal{K}(D0) \circ \uparrow \mathcal{K}(D1)$$

New unit

Theorem

(1) $\uparrow \mathcal{K}(II) \quad = \quad \mathbf{true} \vdash (prob' = prob)$

(2) $\mathcal{K}(II) \circ \uparrow P \quad = \quad P$

Deriving probabilistic semantics

$$
\begin{aligned}
\bot &=_{\text{df}} \mathcal{K}(\mathbf{true}) \\
II &=_{\text{df}} \mathcal{K}(\mathbf{true} \vdash (s' = s)) \\
x{:=}\ e &=_{\text{df}} \mathcal{K}(\mathbf{true} \vdash (s' = s[e/x])) \\
P \lhd b \rhd Q &=_{\text{df}} (b \wedge P) \ \vee \ (\neg b \wedge Q) \\
P \sqcap Q &=_{\text{df}} P \ \vee \ Q \ \vee \bigvee_{0<r<1} (P \|_{M_r} Q) \\
P \ {}_r\!\oplus Q &=_{\text{df}} P \|_{M_r} Q, \qquad \text{when } 0 < r < 1 \\
P \ {}_1\!\oplus Q &=_{\text{df}} P \quad P \ {}_0\!\oplus Q \ =_{\text{df}} \ Q \\
P; Q &=_{\text{df}} P \circ \uparrow Q \\
(\mu X \bullet P(X)) &=_{\text{df}} \bigvee \{Q \mid Q \sqsupseteq P(Q)\}
\end{aligned}
$$

Predicate-transformer semantics

A retraction was used to induce an embedding of standard programs into a space of probabilistic programs. The principal tool was the 'weakest completion' of a sub-commuting diagram, for which we used the weakest pre- and post-specifications.
In the following we apply the weakest completion to probabilistic transformer (as opposed to relational) semantics, for which our starting point is standard predicate-transformer semantics.

Probabilistic predicate

A 'probabilistic predicate' A holds at s only with some probability: thus the type of A should be $S \to [0,1]$, which we write $\mathbb{E}S$.

Standard predicates in $\mathbb{P}S$ are injected into $\mathbb{E}S$ simply by converting them to characteristic functions: writing P^* for that injection of P, we note that P holds at s if the probability that P^* holds at s is 1, and that P does not hold at s if the probability that P^* holds at s is 0.

Embedding programs

Based on our injection $()^*$ of predicates, we now seek an embedding of programs: for standard transformer t in $\mathbb{P}S \to \mathbb{P}S$ we want a definition of its embedding t^*, which will be a probabilistic predicate transformer in $\mathbb{E}S \to \mathbb{E}S$.

We turn therefore to 'healthiness conditions' on $\mathbb{E}S \to \mathbb{E}S$, making explicit which conditions — like monotonicity — we will impose.

Healthiness conditions

(1) Continuity

$$t(\sqcup \mathcal{A}) \quad = \quad \sqcup \{t(A) \mid A \in \mathcal{A}\}$$

(2) Scaling: For all c in $[0, 1]$, probabilistic predicate A in $\mathbb{E}S$ and transformer u in $\mathbb{E}S \to \mathbb{E}S$ we have

$$u(c \times A) \quad = \quad c \times u(A)$$

Weakest completion

t^*, for t in $\mathbb{P}S \to \mathbb{P}S$, is defined as the least monotonic and scaling transformer in $\mathbb{E}S \to \mathbb{E}S$ that satisfies

$$t^*(P^*) \;=\; t(P)^*$$

Program structure is preserved

Theorem

1. $\bot^*(A) = \underline{0}$.
2. $(x := e)^*(A) = A \circ (\lambda s \bullet s[e/x])$.
3. $(t_1 \lhd B \rhd t_2)^* = t_1^* \lhd B \rhd t_2^*$.
4. $(t_1; t_2)^* = t_1^*; t_2^*$
5. Demonic choice, acting as conjuunction for standard transformers, becomes *minimum* for probabilistic transformers:

$$(t_1 \Box t_2)^*(A) = t_1^*(A) \sqcap t_2^*(A) \ ;$$

Conclusion

Choosing the 'right' semantic extension can be a tricky business, to some extent a lucky guess. Here we have isolated a principle that can reduce the guesswork.
In our two examples we postulated an extension of the 'base' type to include the desired new information — in this case probability. Although the base types differ — states on the one hand, and predicates (sets of states) on the other — the subsequent procedure was broadly the same in each case: to induce a corresponding embedding of programs, take the weakest completion of a certian sub-commuting diagram.

Preservation of algebraic laws

We have shown that the embedding is actually a homomorphism, *i.e.* it distributes over appropriately-defined programming operators. As a result, most of the algebraic laws, which were established in the original semantical framework and were used to capture the properties of the programming operators, remain valid in the enriched model.

Weakest completion and data refinement

The weakest completion approach has been used for in support of data refinement in VDM and other state-based development methods, where the link is designed to connect abstract data type with its concrete representation. It is also used for data abstraction in model checking by dramatically reducing the size of state space. This paper illustrates another way of using simulation to derive enriched semantics.