

Appendix A

Alternative approaches

A.1	Probabilistic Hoare-triples	313
A.2	A programming logic of distributions	316

A.1 Probabilistic Hoare-triples

This section explores the first of two alternative approaches to our use of expectations as the basis for a probabilistic program logic. The alternative turns out to be non-compositional when both probabilistic- and demonic choice are present.

Our point of departure is to generalise Hoare-triples *as a whole* from absolute to probabilistic judgements. That is, instead of changing the “raw material” of our logical statements, *i.e.* changing what they are about (about expectations rather than predicates), we change the nature of the statements themselves.

The standard view is that a precondition *guarantees* some program will establish a postcondition; we generalise that as follows. Continuing with standard predicates, we introduce probability via probabilistic judgements of the form

$$p \vdash \{pre\} prog \{post\} , \tag{A.1}$$

that mean “from any initial state in *pre* the program *prog* will with probability at least *p* reach a final state in *post*.” In general, probability *p* can be an expression over the initial state.

A typical Hoare-triple rule in the resulting system would be this one, for sequential composition: when probabilities p, q are *constant*, we have

$$\frac{p \vdash \{pre\} prog_0 \{mid\} \quad q \vdash \{mid\} prog_1 \{post\}}{p * q \vdash \{pre\} prog_0; prog_1 \{post\}}$$

for any programs $prog_0, prog_1$ and standard predicates $pre, mid, post$.¹

It relies on the probabilistic choices in $prog_0$ and $prog_1$ being independent, and on the monotonicity of multiplication. Indeed by defining

$$p \vdash \{pre\} prog \{post\} \quad := \quad p * [pre] \Rightarrow wp.prog.[post]$$

such statements become special cases within our current system, and the above sequential composition rule is easily proved from sublinearity.² That means that we can use rules like the above safely, if we find them more intuitive than the full expectation-based logic; it also means that the proposal adds no expressive power.

In fact, the problem is that probabilistic Hoare-triples are not expressive enough, and thus we cannot adopt this approach as the *sole* basis for our program logic: not only are the judgements (A.1) too weak, they are not compositional in general. Consider for example the two programs

$$\begin{aligned} prog_0 &:= n := 4 \sqcap (n := 5 \frac{1}{2} \oplus n := 6) \\ prog_1 &:= (n := 4 \sqcap n := 5) \frac{1}{2} \oplus (n := 4 \sqcap n := 6) . \end{aligned}$$

(They correspond to executing the game of Fig. 1.3.1 from initial squares 0 and 1 respectively.) In Fig. A.1.1 we set out all eight possible judgements of the form (A.1), showing that in this simpler system $prog_0$ and $prog_1$ would be identified. Are they therefore the same?

No they are not: define a further program

$$prog \quad := \quad (n := 5 \frac{1}{2} \oplus n := 6) \text{ if } n = 4 \text{ else skip} ,$$

and consider the sequential compositions $(prog_0; prog)$ and $(prog_1; prog)$ with respect to the postcondition $n = 5$: we have

$$\begin{aligned} &1/2 \vdash \{\text{true}\} prog_0; prog \{n = 5\} \\ \text{but } &1/2 \not\vdash \{\text{true}\} prog_1; prog \{n = 5\} , \end{aligned}$$

¹If q in particular were not constant, we would have to take account of its being evaluated over the *final* state of $prog_0$ (i.e. the initial state of $prog_1$) rather than the *initial* state of $prog_0$ as is the case for p . The resulting composite probability would then be

$$p * (\sqcap v \mid mid \bullet q) ,$$

where v is the vector of variables that $prog_0$ can assign to, since — taking the demonic view — we would have to assume that any choice inherent in postcondition mid for $prog_0$ would be exploited to make q as low as possible.

²Use its consequences *scaling* and *monotonicity*.

possible postcondition	$prog_0$ probability	$prog_1$ probability
false	0	0
$n = 4$	0	0
$n = 5$	0	0
$n = 6$	0	0
$n \neq 4$	0	0
$n \neq 5$	1/2	1/2
$n \neq 6$	1/2	1/2
true	1	1

Programs $prog_0$ and $prog_1$ cannot be distinguished with standard postconditions.

Figure A.1.1. COUNTER-EXAMPLE TO COMPOSITIONALITY

$postE$	$[n = 4] + 2[n = 5]$	$[n = 4] + 2[n = 6]$
$wp.prog_0.postE$	1	1
$wp.prog_1.postE$	1/2	1/2

Programs $prog_0, prog_1$ are distinguished by either of the two post-expectations $postE$.

Figure A.1.2. COMPOSITIONALITY REQUIRES FULL USE OF EXPECTATIONS

and in fact the strongest judgement we can make about $prog_1$ is

$$1/4 \vdash \{\text{true}\} prog_1; prog \{n = 5\} . \quad ^3$$

That is why standard postconditions are not expressive enough — if the programs $(prog_0; prog)$ and $(prog_1; prog)$ are different, then $prog_0$ and $prog_1$ cannot be the same.⁴ This lack of compositionality is why we do not use probabilistic Hoare-triples.

Fig. A.1.2 shows that $prog_0$ and $prog_1$ are indeed distinguished by the properly probabilistic post-expectations that we introduced in Chap. 1.

³For $(prog_0; prog)$ note that it doesn't matter how the initial nondeterministic choice is resolved, since the result is 1/2 either way. For $(prog_1; prog)$ however the probability of establishing $n = 5$ is $1/2 \sqcap 1 = 1/2$ for the left branch of the initial choice $\frac{1}{2} \oplus$, but $1/2 \sqcap 0 = 0$ for the right branch; thus overall it is only $(1/2 + 0)/2 = 1/4$.

⁴A further (but only informal) argument that $prog_0$ and $prog_1$ should be distinguished is the observation that $prog_0$ should terminate in states 5, 6 “with equal frequency,” however low or high that might be — but $prog_1$ does not have that property.

Demonic nondeterminism is to blame for the above effects. In Chap. 8 we saw from Thm. 8.3.5 that deterministic programs are linear. It is clear that Boolean postconditions are enough for those: over a finite state space at least, linearity determines general pre-expectations from the weakest pre-expectations with respect to the standard “point” postconditions that correspond to single states.

Thus it seems that any semantics for the probabilistic language of guarded commands — with its demonic nondeterminism — must be at least as powerful as the system we have proposed. He *et al.* give a more extensive discussion of alternative models [HSM97].

A.2 A programming logic of distributions

A second alternative to our approach is to “lift” the whole semantics, from states to *distributions* over states. We imagine that probabilistic programs move from distributions to distributions (rather than from states to states, as standard programs do), and we reconstruct the whole of the usual weakest-precondition apparatus above that, considering distributions now as “higher-order” states in their own right but with an internal, probabilistic structure.

Thus our pre- and postconditions will be formulae about *distributions*, containing (sub-)formulae like

$$\begin{array}{ll} \text{Pr.}(c = \text{heads}) \geq 1/2 & \text{the probability that } c \text{ is} \\ & \text{heads is at least } 1/2 \\ \\ \text{and } \text{Exp.}(n) \leq 3 & \text{the expected value of } n \text{ }^5 \\ & \text{is no more than } 3. \end{array}$$

These hold, or do not hold, over *distributions* of states containing variables like c and n . We would for example have the judgement

$$\{\text{true}\} \ c := \text{heads}_{1/2} \oplus \text{tails} \ \{\text{Exp.}[c = \text{heads}] \geq 1/2\} ,$$

about the behaviour of a fair coin c ; in the style of Chap. 1 (but as a Hoare triple) we would instead have written that as

$$\{1/2\} \ c := \text{heads}_{1/2} \oplus \text{tails} \ \{c = \text{heads}\} , \quad ^6$$

and in the notation of the previous alternative we would have written

$$1/2 \vdash \ \{\text{true}\} \ c := \text{heads}_{1/2} \oplus \text{tails} \ \{c = \text{heads}\} .$$

⁵In fact the use of Exp is the more general since, as we have seen, we can express probabilities via characteristic functions: the first formula above is equivalently $\text{Exp.}[c = \text{heads}] \geq 1/2$.

⁶Recall that to reduce clutter we omit embedding brackets $[\dots]$ immediately enclosed by assertion brackets $\{\dots\}$.

But again (as in Sec. A.1) we are in difficulty with demonic nondeterminism. Consider this example: if *Fair* stands for the predicate over distributions

$$\Pr.(c = \text{heads}) = \Pr.(c = \text{tails}) ,$$

then we have these two judgements about programs operating over a variable c representing a coin as above: both

$$\{Fair\} \quad \mathbf{skip} \quad \{Fair\}$$

$$\text{and } \{Fair\} \quad c := \bar{c} \quad \{Fair\} ,$$

hold, where $\overline{\text{heads}} = \text{tails}$ etc. The first program leaves the state unchanged, and the second permutes it in a way that does not change the given (uniform) distribution.

But we also have the general principle that if two programs satisfy the *same* specification then so does the demonic choice between them,⁷ and so from the above we would expect

$$\{Fair\} \quad \mathbf{skip} \sqcap c := \bar{c} \quad \{Fair\}$$

to hold as well — yet it does not. That demonic choice $\mathbf{skip} \sqcap c := \bar{c}$ is refined for example by the deterministic $c := \text{tails}$ which never establishes postcondition *Fair* at all, whether precondition *Fair* held initially or not.

Because there are several phenomena involved here — and all our preconceptions as well — we cannot point to any one of them and say “that causes the contradiction.” But one way of describing the situation is as follows.

Our treatment of demonic nondeterminism is the traditional one in which the imagined demon can resolve the choice, at runtime, with full knowledge of the state at the time the choice is to be made. That is inherent in our postulated refinement

$$\mathbf{skip} \sqcap c := \bar{c} \sqsubseteq c := \text{tails}$$

from above, in which we imagine the demon chooses the left-hand \mathbf{skip} when c is *tails*, and the right-hand $c := \bar{c}$ otherwise. In effect we are using the law

$$(\dots \sqcap \dots) \sqsubseteq (\dots \text{ if } G \text{ else } \dots) , \tag{A.2}$$

which holds for any Boolean G and for the test “ $c = \text{tails}$ ” in particular.⁸

⁷This is a general property of any approach that relates refinement \sqsubseteq and demonic choice \sqcap in the elementary way we prefer, that is as given by the simple rules for a partial order. See for example Law 6 in Sec. B.1.

⁸See Law 7 in Sec. B.1.

When we lift the whole semantic structure up to distributions, from states, the demonic choice “loses” the ability to see *individual* states: it can only see distributions. Equivalently, the choice \sqcap must be resolved “blind” although still arbitrarily, *i.e.* unpredictably but without looking at the state.

One way of doing that is to insist that all demonic choices are made in advance, as if the demon were required to write its future decisions down on a piece of paper before the program is begun. Once the program is running, the decisions “left now, or right” are carried out exactly, in sequence, and cannot be changed.⁹

There are circumstances in which such *oblivious* nondeterminism, as we call it, is the behaviour we are trying to capture — for example when we are dealing with concurrency or modularity in which separation of processes, or information hiding, can “protect” parts of the state from being read freely by other parts of the system.

For sequential programs, however, the use of laws like (A.2) on p. 317 is so pervasive that we consider it to be the deciding factor in this case.

⁹Making the choices in advance is in fact the usual semantic technique for dealing with nondeterminism when it is the principal object of study [Seg95]; we do just that in Sec. 11.6.1 when dealing with demonic, angelic and probabilistic choice all at once. In that case nondeterminism is controlled by whether the decisions made in advance are a sequence of simple Booleans, interpreted “go left” or “go right” (as suggested above: a very weak form of nondeterminism), or are a sequence of predicates over states (so called “memoriless” strategies that can see the current state but have no access to previous states: a stronger form), or are a sequence of predicates over “state histories,” which can resolve a nondeterministic choice using knowledge not only of the state the system is in now, but also of the states it has passed through to get there (a stronger form still, and the one used in this text). But this extra semantic power has a cost.¹⁰

¹⁰An advantage of including strategies explicitly in the mathematical model is that it is then possible to make fine adjustments, as above, to their power; and it is easier to discuss issues related to the strategies themselves. A great disadvantage for practical reasoning, however, is that such models often fail to be “fully abstract,” where *full abstraction* means that program fragments are identified in the model exactly when they are operationally interchangeable [Sto88].

Here one loses full abstraction because the strategy sequence contains “too much information,” in this case the order in which the strategy elements are used. The two programs

$$c := \text{heads} \sqcap \text{tails}; d := \text{heads} \sqcap \text{tails}$$

and

$$d := \text{heads} \sqcap \text{tails}; c := \text{heads} \sqcap \text{tails}$$

are equal in their observable behaviour; yet in their semantics — as functions of strategy sequences — they differ. That is, the first program’s assignment to c is controlled by the first element of the strategy sequence; but in the second program, the first element of the strategy sequence controls d .

When unwanted distinctions like that occur, it is necessary to use more elaborate techniques to prove algebraic equalities. Kozen rejected a similar sequence-of-choices model for (deterministic) probabilistic programs on just those grounds (among other reasons) [Koz81]; and it is for similar reasons (again, among others) that we use the model we have chosen, an extension of Kozen’s [HSM97].

In Chap. 11 we have it both ways, however: we prove the equivalence of two models, one with explicit strategies and one without. The explicit-strategy model is used to establish *e.g.* that memoriless and full-memory strategies are equivalent over finite state spaces (*i.e.* that the “stronger” and “stronger-still” options above are the same); and the implicit-strategy model — our main subject — can be used to formulate algebraic and logical laws (as we did in Chap. 10).

Appendix B

Supplementary material

B.1	Some algebraic laws of probabilistic programs	321
	B.1.1 A list of algebraic laws.	322
	B.1.2 . . . and an example of their use	325
B.2	Loop rule for <i>demonic</i> iterations	328
B.3	Further facts about probabilistic <i>wp</i> and <i>wlp</i>	331
B.4	Infinite state spaces	332
	B.4.1 Topological preliminaries; compactness of \bar{S}	332
	B.4.2 The Galois functions; continuity	334
	B.4.3 The Galois connection	337
	B.4.4 Healthiness conditions	338
	B.4.5 Demonic closure in the infinite case	338
	B.4.6 Angelic closure in the infinite case	340
B.5	Linear-programming lemmas	341
B.6	Further lemmas for <i>eventually</i>	342

B.1 Some algebraic laws of probabilistic programs

Many algebraic laws for programs have quite easy proofs from the weakest pre-expectation definitions of the operators involved, and subsequently allow reasoning about programs directly without appealing again to the logic. In effect, they provide a “third layer” of intellectual tools, above the logic which in turn lies above the semantics — and, in practice, we use the algebra if we can. If that fails we appeal to the logic; and in rare cases

we must push all the way down to the model, the principal reference from which all else is derived.

Although many of the laws we give apply to all demonic/probabilistic programs, in some cases we make restrictions as indicated by these naming conventions:¹

- General demonic/probabilistic programs ... *prog*
as in... $s := 0_{\geq p} \oplus 1$ for $[0, 1]$ -valued expression p ²
- Deterministic (but possibly probabilistic) programs ... *det*
 $s := 0_p \oplus 1$
- Standard (but possibly demonic) programs ... *std*
 $s := 0 \sqcap 1$
- Standard deterministic programs ... *stdet*
 $s := (0 \text{ if } G \text{ else } 1)$ for Boolean expression G

B.1.1 A list of algebraic laws...

We list a number of the laws, below; they are collected into groups based on the healthiness conditions (or other facts) that justify them. In some cases we include “conventional” laws (*i.e.* well-known from standard programming) also.³

- Laws following from basic arithmetic of the operators separately:
 1. Demonic choice is commutative, associative and idempotent.⁴
 2. Probabilistic choice is idempotent and *quasi-commutative*:

$$\begin{aligned} prog_p \oplus prog &= prog \\ prog_1 \oplus_p prog_2 &= prog_2 \oplus_{\bar{p}} prog_1 \end{aligned}$$

We will usually assume the above two laws without comment.

3. Probabilistic choice is *quasi-associative*: for $0 \leq p, q < 1$ and $p + q \leq 1$ we have

$$prog_1 \oplus_p (prog_2 \oplus_{q/\bar{p}} prog_3) = (prog_1 \oplus_{p/\bar{q}} \oplus prog_2) \oplus_q prog_3$$

¹We haven’t included laws here for angelic programs.

²Note that the probability p may be an *expression*, *i.e.* it may depend on the current state.

³A similar set of probabilistic laws was given by Jifeng He *et al.* in earlier work [HSM97]; they were shown in fact to be complete, but for a slightly different model.

⁴Recall that an operator \odot is IDEMPOTENT whenever $x \odot x = x$ for all x .

4. Probabilistic choice is *quasi-distributive*: for $0 \leq p, q, r \leq 1$ we have

$$(prog_1 \text{ }_p\oplus prog_2) \text{ }_q\oplus (prog_3 \text{ }_r\oplus prog_4) = \left| \begin{array}{l} prog_1 \quad @ \quad pq \\ prog_2 \quad @ \quad \bar{p}q \\ prog_3 \quad @ \quad \bar{q}r \\ prog_4 \quad @ \quad \bar{q}\bar{r} \end{array} \right.$$

5. (From Law 4; compare Laws 23 and 24 below.)

$$\begin{aligned} (prog_1 \text{ }_p\oplus prog_2) \text{ }_q\oplus (prog_1 \text{ }_p\oplus prog_3) &= prog_1 \text{ }_p\oplus (prog_2 \text{ }_q\oplus prog_3) \\ \text{and } (prog_1 \text{ }_p\oplus prog_2) \text{ }_{\geq q}\oplus (prog_1 \text{ }_p\oplus prog_3) &= prog_1 \text{ }_p\oplus (prog_2 \text{ }_{\geq q}\oplus prog_3) \end{aligned}$$

• Law relating demonic choice and refinement:

$$6. \quad prog_1 \sqsubseteq prog_2 \sqcap prog_3 \quad \text{iff} \quad \begin{array}{l} prog_1 \sqsubseteq prog_2 \\ \text{and } prog_1 \sqsubseteq prog_3 \end{array}$$

• Laws depending on the arithmetic of $\text{ }_p\oplus$ and \sqcap together:

7. For any probabilities $0 \leq p \leq q \leq 1$ and Boolean expression G , both possibly functions of the state, we have

$$prog_1 \sqcap prog_2 \sqsubseteq \left\{ \begin{array}{l} prog_1 \text{ }_{\geq p}\oplus prog_2 \\ prog_1 \text{ if } G \text{ else } prog_2 \end{array} \right. \sqsubseteq \left\{ \begin{array}{l} prog_1 \\ prog_1 \text{ }_p\oplus prog_2 \\ prog_1 \text{ }_{\geq q}\oplus prog_2 \end{array} \right.$$

$$8. \quad (prog_1 \sqcap prog_2) \text{ }_p\oplus prog_3 = (prog_1 \text{ }_p\oplus prog_3) \sqcap (prog_2 \text{ }_p\oplus prog_3)$$

$$9. \quad (prog_1 \sqcap prog_2) \text{ }_{\geq p}\oplus prog_3 = (prog_1 \text{ }_{\geq p}\oplus prog_3) \sqcap (prog_2 \text{ }_{\geq p}\oplus prog_3)$$

$$10. \quad (prog_1 \sqcap prog_2) \text{ }_{\leq p}\oplus prog_3 = (prog_1 \text{ }_{\leq p}\oplus prog_3) \sqcap (prog_2 \text{ }_{\leq p}\oplus prog_3)$$

$$11. \quad (prog_1 \text{ }_p\oplus prog_2) \sqcap prog_3 \sqsupseteq (prog_1 \sqcap prog_3) \text{ }_p\oplus (prog_2 \sqcap prog_3)$$

• Laws involving sequential composition that are consequences of the way wp is applied from right to left:

$$12. \quad (prog_1 \text{ if } G \text{ else } prog_2); prog_3 = prog_1; prog_3 \text{ if } G \text{ else } prog_2; prog_3$$

$$13. \quad (prog_1 \text{ }_p\oplus prog_2); prog_3 = prog_1; prog_3 \text{ }_p\oplus prog_2; prog_3$$

$$14. \quad (prog_1 \sqcap prog_2); prog_3 = prog_1; prog_3 \sqcap prog_2; prog_3$$

15. (From Laws 13, 14 and the definition of $\text{ }_{\geq p}\oplus$.)

$$(prog_1 \text{ }_{\geq p}\oplus prog_2); prog_3 = prog_1; prog_3 \text{ }_{\geq p}\oplus prog_2; prog_3$$

- Laws following from sublinearity (and its consequence, monotonicity):

16. (Compare Law 18 below.)

$$\text{prog}_1; (\text{prog}_2 \oplus_p \text{prog}_3) \sqsupseteq \text{prog}_1; \text{prog}_2 \oplus_p \text{prog}_1; \text{prog}_3$$

17. (Compare Law 19 below.)

$$\text{prog}_1; (\text{prog}_2 \sqcap \text{prog}_3) \sqsubseteq \text{prog}_1; \text{prog}_2 \sqcap \text{prog}_1; \text{prog}_3$$

- Laws for sequential composition when the left-hand side is restricted:

18. (Compare Law 16.) $\text{det}; (\text{prog}_1 \oplus_p \text{prog}_2) = \text{det}; \text{prog}_1 \oplus_p \text{det}; \text{prog}_2$

19. (Compare Law 17.) $\text{std}; (\text{prog}_1 \sqcap \text{prog}_2) = \text{std}; \text{prog}_1 \sqcap \text{std}; \text{prog}_2$

20. (From Laws 18, 19 and the definition of $\geq_p \oplus$.)

$$\text{stdet}; (\text{prog}_1 \geq_p \oplus \text{prog}_2) = \text{stdet}; \text{prog}_1 \geq_p \oplus \text{stdet}; \text{prog}_2$$

21. (From Laws 13, 18 and 4.)

$$(\text{det}_1 \oplus_p \text{det}_2); (\text{prog}_3 \oplus_q \text{prog}_4) = \begin{array}{l} \text{det}_1; \text{prog}_3 \ @ \ pq \\ \text{det}_1; \text{prog}_4 \ @ \ p\bar{q} \\ \text{det}_2; \text{prog}_3 \ @ \ \bar{p}q \\ \text{det}_2; \text{prog}_4 \ @ \ \bar{p}\bar{q} \end{array}$$

22. (From Laws 8, 13 and 19.)

$$\begin{aligned} & (\text{std}_1 \oplus_p \text{std}_2); (\text{prog}_3 \sqcap \text{prog}_4) \\ &= \begin{array}{l} \text{std}_1; \text{prog}_3 \oplus_p \text{std}_2; \text{prog}_3 \\ \sqcap \text{std}_1; \text{prog}_3 \oplus_p \text{std}_2; \text{prog}_4 \\ \sqcap \text{std}_1; \text{prog}_4 \oplus_p \text{std}_2; \text{prog}_3 \\ \sqcap \text{std}_1; \text{prog}_4 \oplus_p \text{std}_2; \text{prog}_4 \end{array} \end{aligned}$$

- Quasi-associative/distributive laws with inequalities:

23. (From Laws 7 and 8 and the definition of $\geq_p \oplus$; compare Law 5.)

$$\begin{aligned} & (\text{prog}_1 \oplus_p \text{prog}_2) \geq_q \oplus (\text{prog}_1 \geq_p \oplus \text{prog}_3) \\ \sqsupseteq & \text{prog}_1 \geq_p \oplus (\text{prog}_2 \geq_q \oplus \text{prog}_3) \\ \sqsupseteq & (\text{prog}_1 \geq_p \oplus \text{prog}_2) \geq_q \oplus (\text{prog}_1 \oplus_p \text{prog}_3) \\ = & (\text{prog}_1 \geq_p \oplus \text{prog}_2) \geq_q \oplus (\text{prog}_1 \geq_p \oplus \text{prog}_3) \end{aligned}$$

24. (From Law 8 and the definition of $\geq_p \oplus$; compare Law 5.)

$$(prog_1 \geq_p \oplus prog_2) \geq_q \oplus prog_3 \quad \sqsupseteq \quad \begin{array}{l} (prog_1 \geq_q \oplus prog_3) \\ \geq_p \oplus (prog_2 \geq_q \oplus prog_3) \end{array}$$

B.1.2 ... and an example of their use

As an example we show the following elementary equivalence, that a 99% reliability for global Boolean a to hold is achieved via two, lower 90% reliabilities for local Booleans b and c .⁵ Our first steps (set out in full detail) will be to remove the local variables: we have

$$\begin{aligned} & b: = \text{true} \geq_{.9} \oplus \text{false}; \\ & c: = \text{true} \geq_{.9} \oplus \text{false}; \\ & a: = b \vee c \\ = & \quad b: = \text{true} \geq_{.9} \oplus \text{false}; && \text{Law 15} \\ & \quad (c: = \text{true}; a: = b \vee c) \geq_{.9} \oplus (c: = \text{false}; a: = b \vee c) \\ = & \quad b: = \text{true} \geq_{.9} \oplus \text{false}; && \text{standard program algebra; } c \text{ local} \\ & \quad a: = \text{true} \geq_{.9} \oplus b \\ = & \quad \begin{array}{l} b: = \text{true}; a: = \text{true} \geq_{.9} \oplus b \\ \geq_{.9} \oplus b: = \text{false}; a: = \text{true} \geq_{.9} \oplus b \end{array} && \text{Law 15} \\ = & \quad \begin{array}{l} (b: = \text{true}; a: = \text{true}) \geq_{.9} \oplus (b: = \text{true}; a: = b) \\ \geq_{.9} \oplus (b: = \text{false}; a: = \text{true}) \geq_{.9} \oplus (b: = \text{false}; a: = b) \end{array} && \text{Law 20} \\ = & \quad \begin{array}{l} a: = \text{true} \geq_{.9} \oplus \text{true} \\ \geq_{.9} \oplus a: = \text{true} \geq_{.9} \oplus \text{false} \end{array} && \text{standard program algebra; } b \text{ local} \\ = & \quad a: = \text{true} \geq_{.9} \oplus (\text{true} \geq_{.9} \oplus \text{false}) . \end{aligned}$$

Now the local variables are gone and the structure of the program is clear. We finish off by continuing

$$\begin{aligned} & a: = \text{true} \geq_{.9} \oplus (\text{true} \geq_{.9} \oplus \text{false}) \\ = & \quad a: = \text{true} && \text{definition } \geq_p \oplus; \text{ Law 8} \\ & \quad \sqcap a: = \text{true} \geq_{.9} \oplus \text{true} \\ & \quad \sqcap a: = \text{true} \geq_{.9} \oplus (\text{true} \geq_{.9} \oplus \text{false}) \end{aligned}$$

⁵The example comes from a case study being carried out by Steve Schneider *et al.* on control-system reliability [SHRT04], using a probabilistic version $pAMN$ of *Event B* [Abr96b].

$$\begin{aligned}
&= a := \mathbf{true} \sqcap a := (\mathbf{true} \text{.}_9 \text{.}_{99} \oplus \mathbf{true}) \text{.}_{99} \oplus \mathbf{false} && \text{Law 3} \\
&= a := \mathbf{true} \sqcap a := \mathbf{true} \text{.}_{99} \oplus \mathbf{false} \\
&= a := \mathbf{true} \text{.}_{\geq 99} \oplus \mathbf{false} , && \text{definition } \text{.}_{\geq p} \oplus
\end{aligned}$$

which finally is our 99% reliability. The calculation we have just done suggests these further laws (thus from Laws 8, 3 and the definition of $\text{.}_{\geq p} \oplus$):

$$25. \text{ prog}_1 \oplus_{\leq pq} \text{ prog}_2 = \text{ prog}_1 \oplus_{\leq p} (\text{ prog}_1 \oplus_{\leq q} \text{ prog}_2)$$

$$26. \text{ prog}_1 \oplus_{\geq pq} \text{ prog}_2 = \text{ prog}_1 \oplus_{\geq p} (\text{ prog}_1 \oplus_{\geq q} \text{ prog}_2)$$

They might be useful if the two instances of prog_1 on the right were subsequently to be manipulated (*e.g.* refined) in different ways, such as in these laws (thus from Laws 25, 26 and monotonicity):

$$27. (\text{ prog}_1 \sqcap \text{ prog}_2) \oplus_{\leq pq} \text{ prog}_3 \sqsubseteq \text{ prog}_1 \oplus_{\leq p} (\text{ prog}_2 \oplus_{\leq q} \text{ prog}_3)$$

$$28. (\text{ prog}_1 \sqcap \text{ prog}_2) \oplus_{\geq pq} \text{ prog}_3 \sqsubseteq \text{ prog}_1 \oplus_{\geq p} (\text{ prog}_2 \oplus_{\geq q} \text{ prog}_3)$$

We finish with some remarks concerning the demonic choice “hidden” within $\text{.}_{\geq p} \oplus$, and its interaction with probabilistic choice. (Similar issues were explored with respect to (2.8) on p. 50.)

The statement $c := \mathbf{true} \text{.}_{\geq .9} \oplus \mathbf{false}$ above contains demonic choice resolved *after* the probabilistic choice carried out in $b := \mathbf{true} \text{.}_{\geq .9} \oplus \mathbf{false}$ immediately before it. But since the following statement is $a := b \vee c$, the “knowledge” that the c -demon has of the b -outcome is of no use: the best c -strategy for making a true is $c := \mathbf{true}$; and the best c -strategy for making a false is $c := \mathbf{true} \text{.}_9 \oplus \mathbf{false}$. Both are independent of the value of b , however determined, and in fact c 's strategy could be chosen in advance — *i.e.* before the program is run — without in any way reducing c 's power to influence the result in a .

But consider the related program

$$\begin{aligned}
b &:= \mathbf{true} \text{.}_{\geq 1/2} \oplus \mathbf{false}; && \text{(B.1)} \\
c &:= \mathbf{true} \text{.}_{\geq 1/2} \oplus \mathbf{false}; \\
a &:= (b \Leftrightarrow c)
\end{aligned}$$

in which we have replaced the disjunction “ \vee ” with equivalence “ \Leftrightarrow ”, an operator which (unlike disjunction) is not monotonic in c (or in b).⁶ Expressed

⁶At the same time we have used 1/2 instead of .9 for the probabilities, because it brings the issues closer to everyday experience, *e.g.* coin flipping.

in English, it suggests the question

if two coins are flipped, each with probability *at least* $1/2$ of giving heads, what is the probability that they will come up showing the same face? (B.2)

Note that our coins are not fair — rather they are “heads biased.”

An informal analysis would suggest that the worst case occurs when the first coin always gives heads, but the second gives heads only half the time (in other words, at the extremes of probability which separate the potential outcomes as much as possible). In that case the answer is that they will be the same with probability ⁷

$$\begin{aligned} & 1 * 1/2 && \leftarrow \text{for heads/heads} \\ + & 0 * 1/2 && \leftarrow \text{for tails/tails} \\ = & 1/2 . \end{aligned}$$

However for the program (B.1) the informal description above, and the subsequent analysis, is *quite wrong*.⁸ In fact we have a situation in which

one coin is flipped, with probability at least $1/2$ of giving heads, and then a second coin is flipped again with probability at least $1/2$ of giving heads *but which probability can be affected by the outcome of the first coin*.

What really is the probability that they will show the same face?

Suppose *b*'s strategy is to use probability $1/2$; subsequently *c* uses $1/2$ if *b* chose true, but uses 1 if *b* chose false. The probability that *a* will be true is now $1/2 * 1/2 + 1/2 * 0 = 1/4$, and that is borne out by calculation:

$$\begin{aligned} & b: = \text{true} \geq_{1/2} \oplus \text{false}; \\ & c: = \text{true} \geq_{1/2} \oplus \text{false}; \\ & a: = (b \Leftrightarrow c) \\ = & \quad b: = \text{true} \geq_{1/2} \oplus \text{false}; && \text{Law 15; } c \text{ local} \\ & \quad a: = b \geq_{1/2} \oplus \bar{b} \\ = & && \text{Laws 15, 20; } b \text{ local} \\ & \quad a: = (\text{true} \geq_{1/2} \oplus \text{false}) \geq_{1/2} \oplus (\text{false} \geq_{1/2} \oplus \text{true}) \\ = & \dots \end{aligned}$$

⁷More rigorously, if we pick probabilities $p, q \geq 1/2$ for *b, c*, then the probability the two coins show the same face is $pq + \bar{p}\bar{q}$, an expression whose least value is $1/2$ in the range given for p, q .

⁸Lynch *et al.* discuss this same issue [LSS94, Example 4.1].

$$\begin{aligned}
\dots &= && \text{definition } \geq_p \oplus; \text{ Law 8} \\
& && a := \text{true} \quad \leftarrow \text{true most likely} \\
& \sqcap && a := \text{true}_{1/2} \oplus \text{false} \\
& \sqcap && a := \text{true}_{1/2} \oplus \text{false} \\
& \sqcap && a := \text{true}_{1/2} \oplus (\text{false}_{1/2} \oplus \text{true}) \\
& \sqcap && a := (\text{true}_{1/2} \oplus \text{false})_{1/2} \oplus \text{false} \quad \leftarrow \text{true least likely} \\
& \sqcap && a := (\text{true}_{1/2} \oplus \text{false})_{1/2} \oplus (\text{false}_{1/2} \oplus \text{true}) \\
= & && \text{Law 7, i.e. } (\sqcap) \sqsubseteq ({}_p \oplus) \\
& && a := \text{true} \quad \leftarrow \text{true most likely} \\
& && \vdots \quad (\text{others subsumed}) \quad \quad \quad \vdots \\
& \sqcap && a := (\text{true}_{1/2} \oplus \text{false})_{1/2} \oplus \text{false} \quad \leftarrow \text{true least likely} \\
= & && a := \text{true}_{\geq 1/4} \oplus \text{false} . \quad \quad \quad \text{Law 3, definition } \geq_p \oplus
\end{aligned}$$

Thus we confirm that the c -demon's knowledge of b 's outcome is important in this case.

A program corresponding to the description (B.2) above would in fact be

$$\begin{aligned}
b, c &:= (\text{true}_{\geq 1/2} \oplus \text{false}), (\text{true}_{\geq 1/2} \oplus \text{false}); \\
a &:= (b \Leftrightarrow c) ,
\end{aligned}$$

in which the first statement generalises the syntactic sugar for $\geq_p \oplus$, i.e. is an abbreviation for

$$\begin{aligned}
& b, c := \text{true}, \text{true} \\
& \sqcap \quad b, c := \text{true}, (\text{true}_{1/2} \oplus \text{false}) \\
& \sqcap \quad b, c := (\text{true}_{1/2} \oplus \text{false}), \text{true} \\
& \sqcap \quad b, c := (\text{true}_{1/2} \oplus \text{false}), (\text{true}_{1/2} \oplus \text{false}) .
\end{aligned} \tag{B.3}$$

Running the assignments “in parallel” avoids the interaction of demonic choice in one with probabilistic choice in the other.

B.2 Loop rule for *demonic* iterations

Here we finish off our proof of the invariant-implies-termination loop rule Thm. 7.3.3, showing that it extends to demonic loops as well. The approach is to use Fact B.3.5 to replace the loop body by an appropriate deterministic refinement of it, as in the following lemma.

Lemma B.2.1 Let the program $dloop$ be defined

$$dloop \quad := \quad \mathbf{do} \ G \rightarrow \mathbf{det} \ \mathbf{od} \ ,$$

for any choice of standard predicate G (the loop guard) and deterministic program det (the loop body). Then for any $loop$ and post-expectation $postE$ there is a det such that $body \sqsubseteq det$ and

$$wp.dloop.postE \quad \equiv \quad wp.loop.postE \ . \quad ^9 \quad (\text{B.4})$$

Proof Define $preE := wp.loop.postE$, and use Fact B.3.5 to choose det so that $body \sqsubseteq det$ and

$$wp.body.preE \quad \equiv \quad wp.det.preE \ . \quad (\text{B.5})$$

Then we have

$$\begin{aligned} & wp.det.preE \quad \mathbf{if} \ G \ \mathbf{else} \quad postE \\ \equiv & \quad wp.body.preE \quad \mathbf{if} \ G \ \mathbf{else} \quad postE \quad \text{by construction (B.5)} \\ \equiv & \quad preE \ , \quad \text{definition } preE; \text{ definition (1.19) of iteration} \end{aligned}$$

so that $preE$ satisfies the (least) fixed-point equation (1.19) given for $wp.dloop.postE$, as well as (by construction) the equation given for $wp.loop.postE$. Hence $wp.dloop.postE \Rightarrow preE$ and, from $body \sqsubseteq det$ and monotonicity, we have

$$wp.dloop.postE \quad \equiv \quad wp.loop.postE$$

as required. □

With Lem. B.2.1 we have our theorem easily.

Theorem B.2.2 INVARIANT-IMPLIES-TERMINATION LOOP RULE

In the terminology of Sec. 7.3, if I is a wp -invariant of $loop$ and $I \Rightarrow T$ then

$$I \quad \Rightarrow \quad wp.loop.([\overline{G}] * I) \ .$$

Proof Use Lem. B.2.1 to choose deterministic refinement det of $body$ so that

$$wp.dloop.([\overline{G}] * I) \quad \equiv \quad wp.loop.([\overline{G}] * I) \ ,$$

and observe that since $body \sqsubseteq det$ we have I a wp -invariant of $dloop$ also.

The result is then immediate from Thm. 7.3.3. □

⁹A similar trick is required for the proof of (2.24) for intrinsically unbounded invariants (p. 72). If all deterministic $\text{det} \sqsubseteq \text{body}$ satisfy Condition (6) in Sec. 2.12 — that the expected value of the invariant I “while still iterating” tends to zero — then in particular the one satisfying (B.4) does, and we proceed as follows.

We suppose $[\overline{G}] * I \rightleftharpoons \text{post}E$ for some invariant I and, using the least fixed-point definition (1.19) of iteration and the linearity of wp.det , we have

$$\text{wp.dloop}([\overline{G}] * I) \equiv (\sqcup n: \mathbb{N} \cdot I_n),$$

where we define

$$\begin{aligned} I_0 &:= 0 \\ I_1 &:= [\overline{G}] * I \\ I_2 &:= [\overline{G}] * I + [G] * \text{wp.det}([\overline{G}] * I) \\ I_3 &:= [\overline{G}] * I + [G] * \text{wp.det}([\overline{G}] * I) + [G] * \text{wp.det}([G] * \text{wp.det}([\overline{G}] * I)) \\ &\vdots \end{aligned}$$

Now we formalise “the expected value of the invariant I after n iterations” as

$$E_n := (\lambda X \cdot [G] * \text{wp.det}.X)^n.I,$$

and since by Condition (6) we have $(\lim_{n \rightarrow \infty} E_n) = 0$, we can add the E_n ’s into the above limit, giving

$$\text{wp.dloop}([\overline{G}] * I) \equiv \left(\lim_{n \rightarrow \infty} I_n + E_n \right),$$

where we must write “lim” on the right because the terms are no longer necessarily increasing. Then we have for example

$$\begin{aligned} &I_3 + E_3 \\ \equiv &\left. \begin{aligned} &+ [\overline{G}] * I \\ &+ [G] * \text{wp.det}([\overline{G}] * I) \\ &+ [G] * \text{wp.det}([G] * \text{wp.det}([\overline{G}] * I)) \end{aligned} \right\} I_3 \\ &+ [G] * \text{wp.det}([G] * \text{wp.det}([G] * \text{wp.det}.I)) \quad \left. \right\} E_3 \\ \Leftarrow &\left. \begin{aligned} &+ [\overline{G}] * I \\ &+ [G] * \text{wp.det}([\overline{G}] * I) \\ &+ [G] * \text{wp.det}([G] * \text{wp.det}([\overline{G}] * I)) \\ &+ [G] * \text{wp.det}([G] * \text{wp.det}([G] * I)) \end{aligned} \right\} \text{invariance of } I \\ \equiv &\left. \begin{aligned} &+ [\overline{G}] * I \\ &+ [G] * \text{wp.det}([\overline{G}] * I) \\ &+ [G] * \text{wp.det}([G] * \text{wp.det}.I) \end{aligned} \right\} \text{linearity of } \text{det} \\ \Leftarrow &I, \quad \text{above two steps twice more} \end{aligned}$$

which inequality is easily verified for all n by induction. Thus we have

$$\text{wp.dloop}([\overline{G}] * I) \equiv (\sqcup n: \mathbb{N} \cdot I_n) \equiv \left(\lim_{n \rightarrow \infty} I_n + E_n \right) \Leftarrow I,$$

as required.

B.3 Further facts about probabilistic wp and wlp

Proofs of these facts can be constructed within a probabilistic wlp semantics [MM01b]; they are placed here in the appendix, rather than in the main text, because that theory has not been included in this volume.

Fact B.3.1 For standard program $prog$ and standard postcondition $post$ we have

$$wlp.prog.post \sqcap wp.prog.1 \quad \Rightarrow \quad wp.prog.post .$$

□

Fact B.3.2 *sub-distributivity of $\&$* For program $prog$ and post-expectations $postE_0, postE_1$ we have

$$wlp.prog.postE_0 \ \& \ wp.prog.postE_1 \quad \Rightarrow \quad wp.prog.(postE_0 \ \& \ postE_1) .$$

□

Fact B.3.3 *sub-distributivity of $+$* For any program $prog$ and post-expectations $postE_0, postE_1$ we have

$$wlp.prog.postE_0 + wlp.prog.postE_1 \quad \Rightarrow \quad wlp.prog.(postE_0 + postE_1) ,$$

with equality when $prog$ is deterministic.

□

Fact B.3.4 For any program $prog$ we have $wlp.prog.1 \equiv 1$.

□

Fact B.3.5 For any program $prog$ and post-expectation $postE$ there is a deterministic refinement of it — a deterministic det with $prog \sqsubseteq det$ — such that

$$wp.prog.postE \quad \equiv \quad wp.det.postE .$$

□

Fact B.3.5 is related to continuity, and is most easily understood by working beyond the logic as in the geometric argument given in Sec. 6.9.5.

Fact B.3.6 For any program $prog$ and post-expectation $postE$ we have

$$\lfloor wp.prog.postE \rfloor \quad \Rightarrow \quad wp.prog.\lfloor postE \rfloor .$$

□

B.4 Details of the extension to infinite state spaces

We now fill in the details behind the ideas introduced in Sec. 8.2 that allow us to extend our semantic models, and the results concerning them, to infinite state spaces. We begin by examining some properties of infinitary discrete probability distributions.

For emphasis, in this section only we use $\mathbb{E}_B S$ for the set of bounded-above expectations over S .

B.4.1 Topological preliminaries; compactness of \overline{S}

Our set of discrete sub-distributions \overline{S} is a subset of the function space $S \rightarrow \mathbb{R}_{\geq}$, which in turn can be regarded as Euclidean space \mathbb{R}_{\geq}^S but now possibly of infinite dimension. That is, each state s in S corresponds to a dimension: a point in the non-negative hyper-octant is determined by a function from dimension to non-negative co-ordinate.¹⁰

The *Euclidean topology* for the real line \mathbb{R}_{\geq} corresponds to the usual Euclidean metric, and the topology \mathcal{E}_S we use for \mathbb{R}_{\geq}^S is the product of the Euclidean topologies over \mathbb{R}_{\geq} for each dimension in S . A basis for \mathcal{E}_S is then the collection of sets of the form

$$N \times \mathbb{R}_{\geq}^{S-P} \tag{B.6}$$

(ignoring order in the product), for all finite subsets P of S and open subsets N of \mathbb{R}_{\geq}^P . For each dimension s in S we say that s is in the *support* of an open set in \mathcal{E}_S just when the projection of that set onto s is not all of \mathbb{R}_{\geq} . Similar notions of finiteness apply to expectations:

Definition B.4.1 FINITARY EXPECTATION An expectation α in $\mathbb{E}S$ is *finitary* if there is a finite subset P of S such that $\alpha.s \neq 0$ only for states s in P . (Note that such expectations are bounded by construction.)

The *support* of an expectation α is the set of states s such that $\alpha.s \neq 0$; thus an expectation is finitary iff it has finite support.

We write $\mathbb{F}S$ for the set of finite subsets of S , and $\mathbb{E}_f S$ for the set of finitary expectations over S .

For subset P of S , we define the *restriction* of α to P as

$$(\alpha \downarrow P).s \quad := \quad \begin{array}{ll} \alpha.s & \text{if } s \in P \\ 0 & \text{otherwise.} \end{array}$$

Thus $\alpha \downarrow P$ is finitary if P is finite. □

¹⁰A review of Chap. 6 makes this clear for finite state spaces. In particular note that points in the Euclidean space do not correspond to states of the program: rather they correspond to “tuples” of numbers, one for each state.

Our technique for extending our results will be to show that they continue to hold in infinite S provided we restrict our attention to finitary expectations; then continuity of the transformers involved will carry the equalities through to all expectations, since any expectation can be written as a directed \sqcup -limit of its finitary restrictions.

First we must investigate some properties of finitary expectations themselves; we begin by showing that finitary expectations can be used to define “closed half-spaces” in \mathbb{R}_{\geq}^S .

Lemma B.4.2 CLOSED FINITARY HALF-SPACE For (finitary) α in $\mathbb{E}_f S$ and r in \mathbb{R}_{\geq} , sets of the forms

$$\{\Delta: \mathbb{R}_{\geq}^S \mid \int_{\Delta} \alpha \leq r\} \quad \text{and} \quad \{\Delta: \mathbb{R}_{\geq}^S \mid \int_{\Delta} \alpha \geq r\}$$

are closed in the topology \mathcal{E}_S over \mathbb{R}_{\geq}^S .

Proof Let the support of α be \bar{P} , finite because α is finitary. Then in either case above the projection of the set is all of \mathbb{R}_{\geq} for each dimension outside of P ; and its projection into \mathbb{R}_{\geq}^P is the complement of an open set there. \square

For infinitary expectations, however, we have closure in one direction only.

Lemma B.4.3 CLOSED INFINITARY HALF-SPACE For any expectation α in $\mathbb{E}_B S$ and r in \mathbb{R}_{\geq} , the set of distributions¹¹

$$\{\Delta: \mathbb{R}_{\geq}^S \mid \int_{\Delta} \alpha \leq r\}$$

is closed in \mathbb{R}_{\geq}^S .

Proof We have

$$\begin{aligned} & \{\Delta: \mathbb{R}_{\geq}^S \mid \int_{\Delta} \alpha \leq r\} \\ = & \{\Delta: \mathbb{R}_{\geq}^S \mid \int_{\Delta} (\sqcup P: \mathbb{F}S \cdot \alpha \downarrow P) \leq r\} \\ = & \{\Delta: \mathbb{R}_{\geq}^S \mid (\sqcup P: \mathbb{F}S \cdot \int_{\Delta} \alpha \downarrow P) \leq r\} \\ & \text{bounded monotone convergence for } \int_{\Delta} \text{ [Jon90]} \\ = & \{\Delta: \mathbb{R}_{\geq}^S \mid (\forall P: \mathbb{F}S \cdot \int_{\Delta} \alpha \downarrow P \leq r)\} \\ = & (\cap P: \mathbb{F}S \cdot \{\Delta: \mathbb{R}_{\geq}^S \mid \int_{\Delta} \alpha \downarrow P \leq r\}), \end{aligned}$$

which is an intersection of a (P -indexed) collection of closed sets (Lem. B.4.2), since $\alpha \downarrow P$ is finitary for all P in $\mathbb{F}S$. \square

¹¹We really do mean “ $\mathbb{E}_B S$ ”, not “ $\mathbb{E}_f S$ ”. The former are the bounded expectations over the whole space S ; the latter are the expectations of finite support, bounded by construction. We use least upper bounds within $\mathbb{E}_f S$ to approximate elements of $\mathbb{E}_B S$ in these arguments.

That Lem. B.4.3 works in only one direction is because our expectations (finitary or not) take only non-negative values. To see that its dual does not hold in general, consider the half-space

$$\{\Delta: \mathbb{R}_{\geq}^S \mid \int_{\Delta} \underline{1} \geq 1\} . \tag{B.7}$$

The origin (Δ everywhere 0) does not lie within it, yet every open set in the basis (B.6) containing the origin also intersects the half-space — for example, take Δ in \mathbb{R}_{\geq}^S to be one at some point in $S-P$ and zero elsewhere. Thus the origin is a limit point of (B.7) but is not in it.

With the above we now have the compactness property that we need for our space of distributions.

Lemma B.4.4 PROBABILITY DISTRIBUTION SPACE IS COMPACT

The space \overline{S} of distributions over S is a compact subset of $(\mathbb{R}_{\geq}^S, \mathcal{E}_S)$.

Proof The set \overline{S} may be written

$$[0, 1]^S \cap \{\Delta: \mathbb{R}_{\geq}^S \mid \int_{\Delta} \underline{1} \leq 1\} .$$

But $[0, 1]^S$ is compact by Tychonoff’s Theorem,¹² and $\{\Delta: \mathbb{R}_{\geq}^S \mid \int_{\Delta} \underline{1} \leq 1\}$ is closed by Lem. B.4.3. Thus \overline{S} is the intersection of a compact set and a closed set, and is therefore compact. (It is also closed.) \square

B.4.2 The Galois functions; continuity

We now return to the connection (Thm. 5.7.7) between the relational space $\mathbb{H}S$ and the expectation-transformer space $\mathbb{T}S$: they are linked by the functions

$$\begin{array}{ll} rp: \mathbb{T}S \rightarrow \mathbb{H}S & \text{(Def. 5.7.1)} \\ \text{and } wp: \mathbb{H}S \rightarrow \mathbb{T}S & \text{(Def. 5.5.2)} \end{array} .$$

For well-definedness of rp in the finite case we showed that given any t in $\mathbb{T}S$ (for which rp is defined), and s in S , the set $rp.t.s$ was up-closed, convex and Cauchy closed: and they are just the conditions placed on result sets in $\mathbb{H}S$ by Def. 5.4.4.

In the infinite case rp is not so well behaved, although up closure and convexity follow as before. But Cauchy closure now requires an explicit appeal to the bounded continuity of t (Def. 5.6.6 on p.147): it can no longer be proved from sublinearity. The following technical lemma shows that when t is boundedly continuous we may restrict ourselves to finitary expectations in the use of Def. 5.7.1:

¹²TYCHONOFF’S THEOREM states that a product of compact sets is compact in the product topology.

Lemma B.4.5 For any boundedly-continuous t in $\mathbb{T}S$ and s in S we have

$$rp.t.s = \left\{ \Delta: \bar{S} \cdot (\forall \alpha: \mathbb{E}_f S \cdot t.\alpha.s \leq \int_{\Delta} \alpha) \right\}.$$

The only change from Def. 5.7.1 is the type of the universally quantified α — it is now drawn from $\mathbb{E}_f S$ rather than $\mathbb{E}S$.

Proof For arbitrary α in $\mathbb{E}_B S$ and Δ in \bar{S} we have

$$\begin{aligned} & (\forall P: \mathbb{F}S \cdot t.(\alpha \downarrow P).s \leq \int_{\Delta} \alpha \downarrow P) \\ \text{implies} \quad & (\sqcup P: \mathbb{F}S \cdot t.(\alpha \downarrow P).s) \leq (\sqcup P: \mathbb{F}S \cdot \int_{\Delta} \alpha \downarrow P) \end{aligned}$$

$$\text{iff} \quad t \text{ boundedly continuous; bounded monotone convergence} \\ t.(\sqcup P: \mathbb{F}S \cdot \alpha \downarrow P).s \leq \int_{\Delta} (\sqcup P: \mathbb{F}S \cdot \alpha \downarrow P)$$

$$\text{iff} \quad t.\alpha.s \leq \int_{\Delta} \alpha.$$

The result then follows directly from Def. 5.7.1, since any $\alpha \downarrow P$ is finitary. \square

With Lem. B.4.5 we can re-establish Cauchy closure in the infinitary case of $rp.t$ for boundedly-continuous t .

Lemma B.4.6 For any boundedly-continuous t in $\mathbb{T}S$ and state s in S , the set of distributions $rp.t.s$ is Cauchy closed in \bar{S} .

Proof We reason

$$\begin{aligned} & rp.t.s \\ = & \left\{ \Delta: \bar{S} \cdot (\forall \alpha: \mathbb{E}_f S \cdot t.\alpha.s \leq \int_{\Delta} \alpha) \right\} && \text{Lem. B.4.5} \\ = & (\cap \alpha: \mathbb{E}_f S \cdot \{F: \bar{S} \mid t.\alpha.s \leq \int_{\Delta} \alpha\}), \end{aligned}$$

which by Lem. B.4.2 is an intersection of Cauchy-closed sets. \square

To see that continuity is a necessary condition for Lem. B.4.6, define t in $\mathbb{T}S$ by

$$t.\alpha.s := \sqcap \alpha \tag{B.8}$$

for all s in S ; it is not continuous when S is infinite,¹³ and corresponds to the unboundedly demonic standard program **chaos** that chooses any final state in S . Now for all Δ in $rp.t.s$ we have $\sqcap \alpha \leq \int_{\Delta} \alpha$ by Def. 5.7.1, so that in particular $1 \leq \int_F \underline{1}$. But conversely $1 \leq \int_F \underline{1}$ implies

$$\sqcap \alpha \leq (\sqcap \alpha) \int_{\Delta} \underline{1} = \int_{\Delta} \sqcap \alpha \leq \int_{\Delta} \alpha,$$

so that distribution \int_{Δ} is in $rp.t.s$ iff $1 \leq \int_{\Delta} \underline{1}$. Reasoning similar to (B.7) above then shows that $rp.t.s = \{\Delta: \bar{S} \mid 1 \leq \int_{\Delta} \underline{1}\}$ is not Cauchy closed.

¹³To see that this t is not continuous, consider $\underline{1}$, the limit of the family of (finitary) expectations $\{\underline{1} \downarrow P \mid P: \mathbb{F}S\}$, and note that $t.\underline{1} \equiv \underline{1}$ whereas $t.(\underline{1} \downarrow P) \equiv \underline{0}$ for any P in $\mathbb{F}S$.

In fact the set $\{\Delta: \overline{S} \mid 1 \leq \int_{\Delta} 1\}$ is the convex closure of the set of all standard final states $\{s: S \cdot \overline{s}\}$ — that is why it represents the program “choose any final state in S .” Applying Cauchy closure as well would include the origin (by the argument at (B.7) again), whence up closure would include all of \overline{S} . Thus the only probabilistic and continuous program that can choose demonically from all of infinitely many final states must possibly fail to terminate as well. (See also Footnote 33 on p. 289.)

This is how the often-used restriction of “bounded nondeterminism” guarantees continuity in the probabilistic models.

Having shown boundedly-continuous t to yield Cauchy-closed $rp.t$, we turn to the converse: that Cauchy-closed r yields boundedly-continuous $wp.r$ — for infinite S it must be done directly, rather than from sublinearity.

Lemma B.4.7 For any relational program r in $\mathbb{H}S$, the corresponding expectation transformer $wp.r$ is boundedly continuous.

Proof Let \mathcal{A} be a \Rightarrow -directed and bounded subset of expectations in $\mathbb{E}_B S$; we show that for any $c > 0$ and state s in S

$$wp.r.(\sqcup \mathcal{A}).s \leq (\sqcup \alpha: \mathcal{A} \cdot wp.r.\alpha.s) + c,$$

which is sufficient for continuity since c may be arbitrarily small. (The other direction is given by monotonicity.)

Define $x := (\sqcup \alpha: \mathcal{A} \cdot wp.r.\alpha.s)$; then we have

$$\begin{array}{ll} & (\forall \alpha: \mathcal{A} \cdot wp.r.\alpha.s \leq x) & \text{definition of } r \\ \text{iff} & (\forall \alpha: \mathcal{A} \cdot (\sqcap \Delta: r.s \cdot \int_{\Delta} \alpha \leq x)) & \text{Def. 5.7.1} \\ \text{implies} & (\forall \alpha: \mathcal{A} \cdot (\exists \Delta: r.s \cdot \int_{\Delta} \alpha \leq x + c)) & c > 0 \\ \text{iff} & (\forall \alpha: \mathcal{A} \cdot \{\Delta: r.s \cdot \int_{\Delta} \alpha \leq x + c\} \neq \emptyset) & \\ \\ \text{implies} & \text{Lem. B.4.3; } r.s \text{ closed in } \overline{S}; \mathcal{A} \text{ directed; } \overline{S} \text{ compact — see } (\dagger) \text{ below} & \\ & (\sqcap \alpha: \mathcal{A} \cdot \{\Delta: r.s \cdot \int_{\Delta} \alpha \leq x + c\}) \neq \emptyset & \\ \\ \text{iff} & (\exists \Delta: r.s \cdot (\forall \alpha: \mathcal{A} \cdot \int_{\Delta} \alpha \leq x + c)) & \\ \text{iff} & (\exists \Delta: r.s \cdot \int_{\Delta} (\sqcup \mathcal{A}) \leq x + c) & \text{bounded monotone convergence} \\ \text{implies} & (\sqcap \Delta: r.s \cdot \int_{\Delta} (\sqcup \mathcal{A}) \leq x + c) & \\ \text{iff} & wp.r.(\sqcup \mathcal{A}).s \leq x + c. & \text{Def. 5.5.2} \end{array}$$

† For the deferred justification note that Lem. B.4.3 and Cauchy closure of $r.s$ imply Cauchy closure of each set $\{\Delta: r.s \cdot \int_{\Delta} \alpha \leq x + c\}$, and \mathcal{A} ’s being directed ensures that they have the finite-intersection property; we thus appeal to the Finite-Intersection Lemma for non-emptiness of their intersection. \square

The geometric-distribution program of Fig. 2.11.1 on p. 69 illustrates continuity for a transformer even where the set of possible final states is infinite. Although that program is “infinitely branching” — every natural number is reachable with nonzero probability — it is still \sqcup -continuous

since the branching is probabilistic, not demonic or angelic. Recall that in contrast infinitely \sqcap -branching programs are generally not \sqcup -continuous.

B.4.3 The Galois connection

Now we re-establish the partial Galois connection, and with it our characterisation of demonic programs, for infinite state spaces.

Let $\mathbb{T}_c S$ be the boundedly continuous expectation transformers in $\mathbb{T}S$. The fact that rp and wp form a partial Galois connection between $\mathbb{H}S$ and $\mathbb{T}_c S$ is not difficult to show: the inequalities

$$rp \circ wp \sqsubseteq \text{id} \quad \text{and} \quad \text{id} \sqsubseteq wp \circ rp$$

do not require finiteness of S . However our original proofs of the stronger results, namely the equality $rp \circ wp = \text{id}$ and that sublinearity characterises the wp images in $\mathbb{T}S$ (reported above as Lem. 5.7.2 and Lem. 5.7.6), did use finiteness.

Our new proof for the first is as follows.

Lemma B.4.8 For any r in $\mathbb{H}S$ we have

$$rp.(wp.r) = r .$$

Proof The proof for the finite case (p. 150) appealed to the *Separating-Hyperplane Lemma* Lem. B.5.1. Using Lem. B.5.3 instead allows the proof to go through for the infinite case. \square

To recover the second result we must strengthen its assumptions to include bounded continuity; then we have

Lemma B.4.9 If (boundedly-continuous) t in $\mathbb{T}_c S$ is sublinear, then

$$wp.(rp.t) = t .$$

Proof We are able to replay our earlier proof of Lem. 5.7.6 that established $wp.(rp.t).\alpha \equiv t.\alpha$, but need consider only expectations α with finite support. For by continuity of t , Lem. B.4.6 and Lem. B.4.7 we have continuity of $wp.(rp.t)$ also, and every element of $\mathbb{E}_B S$ is the limit of a directed subset of $\mathbb{E}_f S$.

Thus we proceed as in the proof of Lem. 5.7.4, but (less generally) for finitary $\alpha: \mathbb{E}_f S$ and using Lem. B.4.5 instead of Def. 5.7.1; thus we reach that

$$\begin{aligned} & (\cap \alpha': \mathbb{E}_f S \cdot \{ \Delta: \bar{S} \mid t.\alpha'.s \leq \int_{\Delta} \alpha' \}) \\ & \cap \{ \Delta: \bar{S} \mid -t.\alpha.s \leq (\int_{\Delta} -\alpha) \} \\ & = \emptyset . \end{aligned} \tag{B.9}$$

Again we argue by the *Finite-Intersection Lemma* that some finite sub-collection of the sets (B.9) has empty intersection.

Consider the union of the supports of all the finitary predicates α' or α generating that M -collection; it is a finite subset T of the state space S .

The distributions and predicates involved in (B.9) can then be restricted to \mathbb{R}_{\geq}^T , a finite dimensional space, and we are effectively considering an empty intersection of M sets based on

$$\begin{aligned} & (\cap \alpha': \mathbb{E}_f T \cdot \{\Delta: \mathbb{R}_{\geq}^T \mid t.\alpha'.s \leq \int_{\Delta} \alpha'\}) \\ \cap & \{\Delta: \mathbb{R}_{\geq}^T \mid -t.\alpha.s \leq (\int_{\Delta} -\beta)\} \\ \cap & \{\Delta: \mathbb{R}_{\geq}^T \mid -1 \leq \int_{\Delta} (-1)\} \\ = & \emptyset. \end{aligned}$$

The subsequent contradiction is achieved exactly as in Lem. 5.7.4. \square

B.4.4 Healthiness conditions

The results above show that in our characterisation of wp -images of programs in $\mathbb{H}S$, the already-established healthiness conditions of Fig. 5.6.7 are applicable to the infinite case as well. The first three conditions in Fig. 5.6.7 are consequences of sublinearity as shown earlier and, in the special case where the scalars are $\{0, 1\}$ -valued, all are generalisations of properties of predicate transformers.

The results of Sections 8.3–8.5 also remain valid within the space of boundedly-continuous expectation transformers. All our constructions preserve continuity, although the extra work in the case of demonic programs (Sec. 8.4) merits some extra explanation, given below.

For the angelic case (Sec. 8.5) we need only ensure that Def. 8.5.4 is applied to finitary expectations, and then Thm. 8.5.8 establishes any boundedly-continuous and semi-sublinear transformer as a supremum of sublinear ones. As summarised in this section, the details of our proofs where we must appeal to theorems from the theory of convex sets are reduced to reasoning over a finite projection of the state space, where those theorems are still valid.

B.4.5 Demonic closure in the infinite case

In Sec. 8.4 we remarked (Footnote 13 on p. 225) that in the infinite case we must take a supremum of finitary infima to construct an arbitrary continuous demonic transformer from pre-deterministic ones. We sketch the details here, to some extent working beyond the logic by relying on geometric arguments. (Refer Chap. 6.)

Since a clump of possible final distributions can have a “smooth” — *i.e.* not piecewise straight — boundary, we see immediately that a finitary infimum will not do. Instead we approach the clump of distributions via a nested sequence of enclosing polytopes (the supremum),¹⁴ each one formed as the intersection of finitely many closed upwards half-spaces — and each

¹⁴A POLYTOPE is a polygon generalised to dimensions other than two.

such intersection, as we see below, can be expressed as a finitary union (an infimum) of up-closures of a single distribution (the pre-deterministic programs).

Let our “target” demonic transformer be t . Take any finite subset P of the state space S , and fix some integer N . From P, N construct a set $\mathcal{A}_{N,P}$ of expectations (of size $\#P * (N+1)$, the set is therefore finite) by taking for each state $s \in P$ all possible values n/N for $0 \leq n \leq N$, and for each state $s \notin P$ the value zero.

‡ For each initial state $s \in P$ consider each (post-)expectation $\alpha \in \mathcal{A}_{N,P}$ in turn, and construct the half-space $\{\Delta: \bar{S} \mid t.\alpha.s \leq \int_{\Delta} \alpha\}$ — that will be the space above the hyperplane with normal α which is just touching the clump $t.\alpha.s$ from below. The intersection of all these α -hyperplanes (still holding s fixed) clearly contains $t.\alpha.s$.

To express that intersection as a finitary union, we refer to Fig. 6.6.1 where we can see (in two dimensions) a clump whose lower boundary is formed from three straight lines; the clump itself is therefore the intersection of the three half-planes lying above (the extensions of) those lines, restricted of course to the space of sub-distributions lying below the base $x + y = 1$ of the triangle. But that clump is also the *union* of the pre-deterministic programs given by the vertices of that clump, as shown in the smaller illustrations below it, of up-closure.

Thus in general (*i.e.* in any finite number of dimensions) we can express the finitary intersection of up half-spaces as the finitary union (with closure) of the pre-deterministic clumps determined by the intersection’s vertices. (Because the polytope has finitely many faces (the half-planes in $\mathcal{A}_{N,P}$), it will have only finitely many vertices.)

Holding N, P (and hence $\mathcal{A}_{N,P}$) fixed, we can do this for each s in P , and so construct an approximation $t_{N,P}$ (from below, or from “outside”) of our target program t . (For $s \notin P$ we take the whole space \bar{S} , *i.e.* behave as **abort** as we did explicitly with the assertion $\{n \leq N\}$ in the example of Sec. 8.4.) And each approximation is the finitary union of pre-deterministic programs.¹⁵

To conclude our argument, we must show that t itself is the supremum of all our continuous approximants $t_{N,P}$, *i.e.* that for all α, s we have

$$t.\alpha.s = (\bigsqcup N: \mathbb{N}; P: \mathbb{F}S \cdot t_{N,P}.\alpha.s) . \tag{B.10}$$

Informally we can say that as P increases in size, more and more of S is covered; and as N increases, the polytope encloses the target clump more and more closely. Formally, we proceed as follows.

¹⁵It is difficult to count how many exactly; but it is no more than the product over all $s \in P$ of the number of vertices of the polytope formed from t and $\mathcal{A}_{N,P}$ for each s by the construction above.

Fix *finitary* expectation α and state s in S ; it is clear from the construction that $t.\alpha.s \geq t_{N,P}.\alpha.s$ for any integer N and finite subset P of S . To show the limit is attained, pick arbitrary real $x > 0$: we show there is a pair N, P such that $t_{N,P}.\alpha.s \geq t.\alpha.s - x$.

Choose P so that it contains both the support of α and the state s ; we assume without loss of generality that α is not everywhere zero. Choose N large enough so that there is an $\alpha' \in \mathcal{A}_{N,P}$ with $\alpha' \ominus x/\sqcup\alpha \Rrightarrow \alpha/\sqcup\alpha \Rrightarrow \alpha'$ — we can do this “bracketing” of $\alpha/\sqcup\alpha$ because we have only finitely many differences $\alpha'.s - \alpha.s/\sqcup\alpha$ to consider, since P is finite and outside of P we know that α is zero. Now we reason as follows:

$$\begin{aligned}
 & t_{N,P}.\alpha.s \\
 = & \sqcup\alpha * t_{N,P}.\alpha/\sqcup\alpha.s && t_{N,P} \text{ scaling} \\
 \geq & \sqcup\alpha * t_{N,P}.\alpha' \ominus x/\sqcup\alpha.s && \text{choice of } \alpha' \\
 \geq & \sqcup\alpha * (t_{N,P}.\alpha'.s \ominus x/\sqcup\alpha) && \ominus \text{ sub-distribution} \\
 \geq & \sqcup\alpha * t_{N,P}.\alpha'.s - x && \text{arithmetic} \\
 = & \sqcup\alpha * t.\alpha'.s - x && t \text{ and } t_{N,P} \text{ agree here — see } (\dagger) \text{ below} \\
 \geq & \sqcup\alpha * t.\alpha/\sqcup\alpha.s - x && \text{choice of } \alpha' \\
 = & t.\alpha.s - x. && t \text{ scaling}
 \end{aligned}$$

† The reason that t and $t_{N,P}$ agree at α', s is that, because $s \in P$, the polytope determining $t_{N,P}$ at s used $\alpha' \in \mathcal{A}_{N,P}$ as one of its constructing faces, and that face was positioned precisely so that it touched the clump determined by t and s . (Recall (§) on p. 339.)

Thus we have shown that (B.10) holds for all finitary α ; but since both t (by assumption) and $t_{N,P}$ (by construction) are continuous, the result extends to all α .

B.4.6 Angelic closure in the infinite case

For the infinite angelic case we must show that if semi-sublinear t is continuous then it is the supremum of *continuous* sublinear t_α 's. Our first step is to re-work the proof of Thm. 8.5.8 as follows:

$$\begin{aligned}
 & t.\beta \\
 \equiv & (\sqcup\alpha: \mathbb{E}_f S \mid \alpha \Rrightarrow \beta \cdot t.\alpha) && t \text{ continuous} \\
 \equiv & (\sqcup\alpha: \mathbb{E}_f S \mid \alpha \Rrightarrow \beta \cdot t_\alpha.\alpha) && \text{Lem. 8.5.5} \\
 \Rightarrow & (\sqcup\alpha: \mathbb{E}_f S \mid \alpha \Rrightarrow \beta \cdot t_\alpha.\beta) && \alpha \Rrightarrow \beta \\
 \Rightarrow & (\sqcup\alpha: \mathbb{E}_f S \cdot t_\alpha.\beta) && \\
 \Rightarrow & t.\beta, && \text{Lem. 8.5.6}
 \end{aligned}$$

showing that continuous t is attainable as the supremum of t_α 's for finitary α . But, as we see now, if α is finitary then t_α is itself continuous.

Let P be the support of α , take any directed set \mathcal{B} of expectations, and for arbitrary $x > 1$ exploit the finiteness of P by choosing $\beta_x \in \mathcal{B}$ so that $(\sqcup\mathcal{B})\downarrow P \Rrightarrow x * \beta_x \downarrow P$. We then have

$$\begin{aligned}
 & t_\alpha.(\sqcup\mathcal{B}) \\
 \equiv & (\sqcup c, c': \mathbb{R}_\geq \mid c\alpha - c' \Rightarrow \sqcup\mathcal{B} \cdot c(t.\alpha.s) - c') && \text{definition } t_\alpha \\
 \Rightarrow & (\sqcup c, c': \mathbb{R}_\geq \mid c\alpha - c' \Rightarrow x * \beta_x \cdot c(t.\alpha.s) - c') && \alpha \text{ zero outside of } P \\
 \equiv & x * (\sqcup c, c': \mathbb{R}_\geq \mid c\alpha - c' \Rightarrow \beta_x \cdot c(t.\alpha.s) - c') && \text{arithmetic} \\
 \equiv & x * t_\alpha.\beta_x && \text{definition } t_\alpha \\
 \Rightarrow & x * (\sqcup\beta: \mathcal{B} \cdot t_\alpha.\beta) . && \beta_x \in \mathcal{B}
 \end{aligned}$$

Since x can be taken arbitrarily close to one, we have our result.

B.5 Linear-programming lemmas

The first two of these lemmas are well known in linear programming.

Lemma B.5.1 THE SEPARATING-HYPERPLANE LEMMA Let \mathcal{C} be a convex and Cauchy-closed subset of \mathbb{R}^N , and p a point in \mathbb{R}^N that does not lie in \mathcal{C} . Then there is a separating hyperplane S with p on one side of it and all of \mathcal{C} on the other.

Proof See for example a standard text on Linear Programming [Tru71, p.8] or Game Theory [Kuh03]. \square

Lemma B.5.2 FARKAS' LEMMA Let A be an $M \times N$ matrix, x an $N \times 1$ column-vector and r an $M \times 1$ column-vector, and suppose that A and r are so that the system of equations

$$A \cdot x \geq r \tag{B.11}$$

has no solution in x , where \cdot denotes matrix multiplication. Then there is a $1 \times M$ row-vector C of non-negative values such that

$$C \cdot A = 0 \quad \text{but} \quad C \cdot r > 0 . \tag{B.12}$$

Proof See Schrijver's text for example [Sch86, p.89], taking the contrapositive of Corollary 7.1e there. \square

Lem. B.5.2 can be motivated by considering its converse, also true but trivially so: if there is a C satisfying (B.12) then inequation (B.11) can have no solution—for if it did, we could reason

$$0 < C \cdot r \leq C \cdot A \cdot x = 0 \cdot x = 0 ,$$

a contradiction. Thus the lemma can be read “if (B.11) has no solution in x then there is a witness C to that fact.”

The connection between probability and linear programming is reported by Fagin *et al.* also [FHM90].

Lemma B.5.3 THE SEPARATING-HYPERPLANE LEMMA — INFINITE CASE
 Let \mathcal{C} be a convex subset of \mathbb{R}^S that is compact (hence closed) in the product \mathcal{E}_S of the Euclidean topologies over its constituent projections \mathbb{R} . If some p does not lie in \mathcal{C} , then there is a separating hyperplane with p on one side of it and all of \mathcal{C} on the other.

Proof If $p \notin \mathcal{C}$, then because \mathcal{C} is closed there is some neighbourhood N in the basis of \mathcal{E}_S with $p \in N$ and $N \cap \mathcal{C} = \emptyset$.

Let T in $\mathbb{F}S$ be the support of N . Writing $(\downarrow T)$ for projection onto T , we then have $p \downarrow T \in N \downarrow T$ and $N \downarrow T \cap \mathcal{C} \downarrow T = \emptyset$, because for the latter $N \downarrow (S - T) = \mathbb{R}^{S-T}$, and thus $p \downarrow T \notin \mathcal{C} \downarrow T$. Note that $\mathcal{C} \downarrow T$ is compact because \mathcal{C} is, hence closed; and it is convex also because \mathcal{C} is.

Applying the standard *Separating Hyperplane Lemma* for the finite dimensional space Lem. B.5.1 in the case of $p \downarrow T$ and $\mathcal{C} \downarrow T$, within the finite-dimensional \mathbb{R}^T , gives us a separating hyperplane there; its extension parallel to the remaining axes $S - T$ is then the hyperplane we seek in \mathbb{R}^S . □

B.6 Further lemmas for *eventually*

In this section we set out the proofs for a number of simple lemmas concerning the quantitative *eventually* operator \diamond .

Lemma B.6.1 PROBABILISTIC DOUBLE-EVENTUALLY 16

For all expectations A we have

$$\diamond \diamond A \equiv \diamond A .$$

Proof Replace \subseteq by \Rightarrow in the proof of Lem. 10.1.2 (p. 266). □

Lemma B.6.2 DATA REFINEMENT OF *eventually* \diamond

If for any monotonic transformer t we have

$$\circ(t.A) \Rightarrow t.(\circ A)$$

for all expectations A , then we have also

$$\diamond(t.A) \Rightarrow t.(\diamond A)$$

for all A .

¹⁶See Footnote 18 on p. 94 for an example of this lemma in use.

Proof We reason

$$\begin{array}{llll}
 & \diamond(t.A) \Rightarrow t.(\diamond A) & & \\
 \text{if} & t.A \sqcup \circ t.(\diamond A) \Rightarrow t.(\diamond A) & \diamond(t.A) \text{ least, Fig. 10.3.1} & \\
 \text{if} & t.A \sqcup t.(\circ \diamond A) \Rightarrow t.(\diamond A) & \text{assumption} & \\
 \text{if} & t.(A \sqcup \circ \diamond A) \Rightarrow t.(\diamond A) & t \text{ monotonic} & \\
 \text{iff} & t.(\diamond A) \Rightarrow t.(\diamond A) . & \text{Def. 9.3.2} &
 \end{array}$$

□

Lemma B.6.3 \diamond MONOTONICITY For all expectations A, B we have

$$A \Rightarrow B \text{ implies } \diamond A \Rightarrow \diamond B .$$

Proof Again use $\diamond A$ least (Fig. 10.3.1), and check that

$$\begin{array}{lll}
 & A \sqcup \circ \diamond B & \\
 \Rightarrow & B \sqcup \circ \diamond B & A \Rightarrow B \\
 \equiv & \diamond B , &
 \end{array}$$

as required. □

Lemma B.6.4 \diamond EXCLUDED MIRACLE For all expectations A we have

$$\diamond A \Rightarrow \underline{\sqcup A} .$$

Proof Use $\diamond A$ least, and check that

$$\begin{array}{lll}
 & A \sqcup \circ \underline{\sqcup A} & \\
 \Rightarrow & A \sqcup \underline{\underline{\sqcup A}} & \circ \text{ excluded miracle} \\
 \equiv & \underline{\sqcup A} . &
 \end{array}$$

□

Lemma B.6.5 \diamond SCALING For all expectations A and scalars p in $[0, 1]$ we have

$$\diamond(pA) \equiv p(\diamond A) .$$

Proof We prove $\diamond(pA) \Rightarrow p(\diamond A)$ first, using $\diamond(pA)$ least and checking

$$\begin{array}{lll}
 & pA \sqcup \circ(p(\diamond A)) & \\
 \equiv & pA \sqcup p(\circ \diamond A) & \circ \text{ scaling} \\
 \equiv & p(A \sqcup \circ \diamond A) & \\
 \equiv & p(\diamond A) . &
 \end{array}$$

For $\diamond(pA) \Leftarrow p(\diamond A)$ note first that it is trivial when $p = 0$. For $p > 0$ we prove equivalently $(\diamond(pA))/p \Leftarrow \diamond A$, where Lem. B.6.4 guarantees well-definedness of the left-hand side:

$$(\diamond(pA))/p \Rightarrow \underline{\sqcup(pA)}/p \Rightarrow \underline{p/p} \equiv \underline{1} .$$

Then using $\diamond A$ *least*, we check

$$\begin{array}{ll}
 & A \sqcup \circ((\diamond(pA))/p) \\
 \equiv & A \sqcup (p/p)(\circ((\diamond(pA))/p)) \qquad p \neq 0 \\
 \equiv & A \sqcup (\circ(p(\diamond(pA))/p))/p \qquad \circ \text{ scaling} \\
 \equiv & A \sqcup (\circ \diamond(pA))/p \\
 \equiv & (pA \sqcup \circ \diamond(pA))/p \qquad p \neq 0 \\
 \equiv & \diamond(pA)/p .
 \end{array}$$

□

Bibliography

- [ABL96] J.-R. Abrial, E. Börger, and H. Langmaack, editors. *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, volume 1165 of *LNCS*. Springer-Verlag, 1996.
- [Abr96a] J.-R. Abrial. *The B Book: Assigning Programs to Meanings*. Cambridge University Press, 1996.
- [Abr96b] J.-R. Abrial. Extending *B* without changing it (for developing distributed systems). In H. Habrias, editor, *First Conference on the B Method*, pages 169–190. Laboratoire LIANA, L’Institut Universitaire de Technologie (IUT) de Nantes, November 1996.
- [AH90] James Aspnes and M. Herlihy. Fast randomized consensus using shared memory. *J. Algorithms*, 11(3):441–61, 1990.
- [AL96] Christoph Andriessens and Thomas Lindner. Using FOCUS, LUSTRE, and probability theory for the design of a reliable control program. In Abrial *et al.* [ABL96], pages 35–51.
- [ASBSV95] A. Aziz, V. Singhal, F. Balarinand R.K. Brayton, and A.L. Sangiovanni-Vincentelli. It usually works: The temporal logic of stochastic systems. In *Computer-Aided Verification, 7th Intl. Workshop*, volume 939 of *LNCS*, pages 155–65. Springer-Verlag, 1995.
- [AZP03] T. Arons, L. Zuck, and A. Pnueli. Parameterized verification by probabilistic abstraction. In Andrew D. Gordon, editor, *FOSSACS 2003*, volume 2620 of *LNCS*, pages 87–102. Springer-Verlag, 2003.
- [Bac78] R.-J.R. Back. On the correctness of refinement steps in program development. Report A-1978-4, Department of Computer Science, University of Helsinki, 1978.

- [Bac88] R.-J.R. Back. A calculus of refinements for program derivations. *Acta Informatica*, 25:593–624, 1988.
- [BAPM83] M. Ben-Ari, A. Pnueli, and Z. Manna. The temporal logic of branching time. *Acta Informatica*, 20:207–26, 1983.
- [BB96] G. Brassard and P. Bratley. *Fundamentals of Algorithmics*. Prentice-Hall, 1996.
- [BdA95] Andrea Bianco and Luca de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Foundations of Software Technology and Theoretical Computer Science*, volume 1026 of *LNCS*, pages 499–512, December 1995.
- [BFL⁺99] B. Bérard, A. Finkel, F. Laroussine, A. Petit, L. Petrucci, Ph. Schoebelen, and P. McKenzie. *Systems and Software Verification: Model-Checking Techniques and Tools*. Springer-Verlag, 1999.
- [BKS83] R.-J.R. Back and R. Kurki-Suonio. Decentralisation of process nets with centralised control. In *2nd ACM SIGACT-SIGOPS Symp. Principles of Distributed Computing*, pages 131–42, 1983.
- [Boo82] H. Boom. A weaker precondition for loops. *ACM Transactions on Programming Languages and Systems*, 4:668–77, 1982.
- [BvW90] R.-J.R. Back and J. von Wright. Duality in specification languages: a lattice theoretical approach. *Acta Informatica*, 27:583–625, 1990.
- [BvW93] R.-J. Back and J. von Wright. Predicate transformers and higher-order logic. In J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *REX Workshop*, volume 666 of *LNCS*, pages 1–20. Springer-Verlag, 1993.
- [BvW96] R.-J.R. Back and J. von Wright. Interpreting nondeterminism in the refinement calculus. In He Jifeng, John Cooke, and Peter Wallis, editors, *Proceedings of the BCS-FACS 7th Refinement Workshop*, Workshops in Computing. Springer-Verlag, July 1996. www.springer.co.uk/ewic/workshops/7RW.
- [BvW98] R.-J.R. Back and J. von Wright. *Refinement Calculus: A Systematic Introduction*. Springer-Verlag, 1998.
- [Chr90] I. Christoff. Testing equivalences and fully abstract models for probabilistic processes. In *CONCUR '90*, volume 458 of *LNCS*, pages 126–40. Springer-Verlag, 1990.
- [CM88] K.M. Chandy and J. Misra. *Parallel Program Design: A Foundation*. Addison-Wesley, Reading, Mass., 1988.
- [CMA02] D. Cansell, D. Méry, and J.-R. Abrial. A mechanically proved and incremental development of the IEEE 1394 tree-identify protocol. *Formal Aspects of Computing*, 14(3):215–27, 2002.
- [Coh00] E. Cohen. Separation and reduction. In *Mathematics of Program Construction, 5th International Conference*, volume 1837 of *LNCS*, pages 45–59. Springer-Verlag, July 2000.
- [CY95] C. Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *JACM*, 42(4):857–907, 1995.

- [dA99] Luca de Alfaro. Computing minimum and maximum reachability times in probabilistic systems. In *Proceedings of CONCUR '99*, volume 1664 of *LNCS*, pages 66–81. Springer-Verlag, 1999.
- [dAH00] Luca de Alfaro and T. Henzinger. Concurrent ω -regular games. In *Proc. 15th IEEE Symp. Logic in Computer Science*, pages 141–54. IEEE, 2000.
- [dAM01] Luca de Alfaro and Rupak Majumdar. Quantitative solution of omega-regular games. In *Proc. STOC '01*, pages 675–83. ACM, 2001.
- [DFP01] M. DufLOT, L. Fribourg, and C. Picaronny. Randomized finite-state distributed algorithms as Markov chains. In *Int. Conf. on Distributed Computing (DISC'2001)*, volume 2180 of *LNCS*, pages 240–54, October 2001.
- [DFP02] M. DufLOT, L. Fribourg, and C. Picaronny. Randomized dining philosophers without fairness assumption. In *Proc. 2nd IFIP Int. Conf. Theoretical Computer Science*, volume 223 of *IFIP Conference Proceedings*, pages 169–80, August 2002.
- [DGJP02] José Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. The metric analogue of weak bisimulation for probabilistic processes. In *Proc. LICS '02*, pages 413–22. IEEE, 2002.
- [DGJP03] José Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. Approximating labelled Markov processes. *Information and Computation*, 184:160–200, 2003.
- [dHdV02] J.J. den Hartog and E.P. de Vink. Verifying probabilistic programs using a Hoare-like logic. *Int. J. Found. Comp. Sci.*, 13(3):315–40, 2002.
- [Dij71] E.W. Dijkstra. Hierarchical ordering of sequential processes. *Acta Informatica*, 1(2):576–80, 583, October 1971.
- [Dij76] E.W. Dijkstra. *A Discipline of Programming*. Prentice Hall International, Englewood Cliffs, N.J., 1976.
- [DIM83] S. Dolev, A. Israeli, and S. Moran. Analyzing expected time by scheduler-luck games. *IEEE Transactions on Software Engineering*, 21(5):429–39, 1983.
- [DP90] B.A. Davey and H.A. Priestly. *Introduction to Lattices and Order*. Cambridge Mathematical Textbooks. Cambridge University Press, 1990.
- [dRE98] W.-P. de Roever and K. Engelhardt. *Data Refinement: Model-Oriented Proof Methods and their Comparison*, volume 47 of *Cambridge Tracts in Computer Science*. Cambridge University Press, 1998.
- [Eda95] A. Edalat. Domain theory and integration. *Theoretical Computer Science*, 151(1):163–93, 1995.
- [EH86] E.A. Emerson and J.Y. Halpern. “Sometimes” and “not never” revisited: on branching vs. linear time temporal logic. *Journal of the ACM*, 33(1):151–78, 1986.

- [Eme90] E.A. Emerson. Temporal and modal logic. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 995–1072. Elsevier and MIT Press, 1990.
- [Eve57] H. Everett. Recursive games. In *Contributions to the Theory of Games III*, volume 39 of *Ann. Math. Stud.*, pages 47–78. Princeton University Press, 1957.
- [Far99] J. Farkas. Die algebraische Grundlage der Anwendungen des mechanischen Princips von Fourier. *Mathematische und Naturwissenschaftliche Berichte aus Ungarn*, 16:25–40, 1899.
- [Fel71] W. Feller. *An Introduction to Probability Theory and its Applications*, volume 2. Wiley, second edition, 1971.
- [FH84] Yishai A. Feldman and David Harel. A probabilistic dynamic logic. *J. Computing and System Sciences*, 28:193–215, 1984.
- [FHM90] R. Fagin, J.Y. Halpern, and N. Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87:78–128, 1990.
- [FHMV95] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.
- [Flo67] R.W. Floyd. Assigning meanings to programs. In J.T. Schwartz, editor, *Mathematical Aspects of Computer Science*, number 19 in Proc. Symp. Appl. Math., pages 19–32. American Mathematical Society, 1967.
- [Fra86] N. Francez. *Fairness*. Texts and Monographs in Computer Science. Springer-Verlag, 1986.
- [FS03] Colin Fidge and Carron Shankland. But what if I don't want to wait forever? *Formal Aspects of Computing*, 14(3):281–94, 2003.
- [FV96] J. Filar and O.J. Vrieze. *Competitive Markov Decision Processes: Theory, Algorithms, and Applications*. Springer-Verlag, 1996.
- [GM91] P.H.B. Gardiner and C.C. Morgan. Data refinement of predicate transformers. *Theoretical Computer Science*, 87:143–62, 1991. Reprinted in [MV94].
- [GM93] P.H.B. Gardiner and C.C. Morgan. A single complete rule for data refinement. *Formal Aspects of Computing*, 5(4):367–82, 1993. Reprinted in [MV94].
- [Gol03] R. Goldblatt. Mathematical modal logic: a view of its evolution. *Journal of Applied Logic*, 1(5–6):309–92, 2003.
- [GR97] Lindsay Groves and Steve Reeves, editors. *Formal Methods Pacific '97*. Discrete Mathematics and Computer Science. Springer-Verlag, 1997.
- [Gro] Probabilistic Systems Group. A quantified measure of security 1: a relational model. Available at [MMSS, key QMSRM].
- [GS92] G.R. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford Science Publications, second edition, 1992.
- [GW86] G.R. Grimmett and D. Welsh. *Probability: an Introduction*. Oxford Science Publications, 1986.

- [Heh89] E.C.R. Hehner. Termination is timing. In J.L.A. van de Snepscheut, editor, *Mathematics of Program Construction*, volume 375 of *LNCS*, pages 36–47. Springer-Verlag, 1989. Invited presentation.
- [Her90] T. Herman. Probabilistic self-stabilization. *Inf. Proc. Lett.*, 35(2):63–7, 1990.
- [Hes92] Wim H. Hesselink. *Programs, Recursion and Unbounded Choice*. Number 27 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, U.K., 1992.
- [Hes95] Wim H. Hesselink. Safety and progress of recursive procedures. *Formal Aspects of Computing*, 7:389–411, 1995.
- [HHS87] C.A.R. Hoare, Jifeng He, and J.W. Sanders. Prespecification in data refinement. *Inf. Proc. Lett.*, 25(2):71–6, May 1987.
- [HJ94] H. Hansson and B. Jonsson. A logic for reasoning about time and probability. *Formal Aspects of Computing*, 6(5):512–35, 1994.
- [HK94] Joseph Y. Halpern and Bruce M. Kapron. Zero-one laws for modal logic. *Annals of Pure and Applied Logic*, 69:157–93, 1994.
- [HK97] Michael Huth and Marta Kwiatkowska. Quantitative analysis and model checking. In *Proceedings of 12th Annual IEEE Symposium on Logic in Computer Science*, pages 111–22, 1997.
- [HMM04] Joe Hurd, A.K. McIver, and C.C. Morgan. Probabilistic guarded commands mechanised in HOL. To appear in *Proc. QAPL '04 (ETAPS)*, 2004.
- [Hoa69] C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–80, 583, October 1969.
- [Hon03] Ross Honsberger. *Mathematical Diamonds*, volume 26 of *Dolciani Mathematical Expositions*. Mathematical Society of America, 2003.
- [HP01] Oltea Mihaela Herescu and Catuscia Palamidessi. On the generalized dining philosophers problem. In *Proc. 20th Annual ACM Symposium on Principles of Distributed Computing (PODC'01)*, pages 81–9, 2001.
- [HP02] Joseph Y. Halpern and Riccardo Pucella. Reasoning about expectation. In *Proceedings of the Eighteenth Conference on Uncertainty in AI*, pages 207–15, 2002.
- [HS86] Sergiu Hart and Micha Sharir. Probabilistic propositional temporal logics. *Information and Control*, 70:97–155, 1986.
- [HSM97] Jifeng He, K. Seidel, and A.K. McIver. Probabilistic models for the guarded command language. *Science of Computer Programming*, 28:171–92, 1997. Available at [MMSS, key HSM95].
- [HSP83] S. Hart, M. Sharir, and A. Pnueli. Termination of probabilistic concurrent programs. *ACM Transactions on Programming Languages and Systems*, 5:356–80, 1983.
- [Hur02] Joe Hurd. A formal approach to probabilistic termination. In Víctor A. Carreño, César A. Muñoz, and Sofiène Tahar, editors, *15th International Conference on Theorem Proving in Higher Order Logics: TPHOLs 2002*, volume 2410 of *LNCS*, pages 230–45,

- Hampton, VA., August 2002. Springer-Verlag.
www.cl.cam.ac.uk/~jeh1004/research/papers.
- [Hut03] Michael Huth. An abstraction framework for mixed non-deterministic and probabilistic systems. In *Validation of Stochastic Systems*, Lecture Notes in Computer Science Tutorial Series. Springer-Verlag, 2003.
- [JHSY94] B. Jonsson, C. Ho-Stuart, and W. Yi. Testing and refinement for nondeterministic and probabilistic processes. In Langmaack, de Roever, and Vytupil, editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 863 of *LNCS*, pages 418–30. Springer-Verlag, 1994.
- [Jon86] C.B. Jones. *Systematic Software Development using VDM*. Prentice-Hall, 1986.
- [Jon90] C. Jones. Probabilistic nondeterminism. Monograph ECS-LFCS-90-105, Edinburgh University, 1990. (Ph.D. Thesis).
- [KB00] M. Kwiatkowska and C. Baier. Domain equations for probabilistic processes. *Mathematical Structures in Computer Science*, 10(6):665–717, 2000.
- [Koz81] D. Kozen. Semantics of probabilistic programs. *Jnl. Comp. Sys. Sciences*, 22:328–50, 1981.
- [Koz83] D. Kozen. Results on the propositional μ -calculus. *Theoretical Computer Science*, 27:333–54, 1983.
- [Koz85] D. Kozen. A probabilistic PDL. *Jnl. Comp. Sys. Sciences*, 30(2):162–78, 1985.
- [Kuh03] Harold W. Kuhn. *Lectures on the Theory of Games*. Princeton University Press, 2003. First appeared, as lecture notes, in 1952.
- [Lam80] L. Lamport. “Sometimes” is sometimes “not never”: on the temporal logic of programs. In *Proc. 7th ACM PoPL*, pages 174–85. ACM, 1980.
- [Lam83] L. Lamport. What good is temporal logic. In R.E.A. Mason, editor, *Proc. IFIP 9th World Congress*, pages 657–68. Elsevier, 1983.
- [LPZ85] O. Lichtenstein, A. Pnueli, and L.D. Zuck. The glory of the past. In *Proc. Logics of Programs Workshop*, volume 193 of *LNCS*, pages 198–218. Springer-Verlag, 1985.
- [LR94] D. Lehmann and M.O. Rabin. On the advantages of free choice: a symmetric and fully-distributed solution to the Dining Philosophers Problem. In Roscoe [Ros94], pages 333–52. An earlier version appeared in *Proc. 8th Ann. Symp. PoPL, 1981*.
- [LS82] D. Lehmann and S. Shelah. Reasoning with time and chance. *Information and Control*, 53(3):165–98, 1982.
- [LS89] Nancy Lynch and Roberto Segala. An introduction to I/O automata. *CWI Quarterly*, 2(3):219–46, 1989.
- [LS91] K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.

- [LSS94] Nancy Lynch, Isaac Saias, and Roberto Segala. Proving time bounds for randomized distributed algorithms. In *Proc. 13th ACM Symp. Prin. Dist. Comp.*, pages 314–323, 1994.
- [LvdS92] J.J. Lukkien and J.L.A. van de Snepscheut. Weakest preconditions for progress. *Formal Aspects of Computing*, 4:195–236, 1992.
- [McI01] A.K. McIver. A generalisation of stationary distributions, and probabilistic program algebra. In Stephen Brookes and Michael Mislove, editors, *Electronic Notes in Theo. Comp. Sci.*, volume 45. Elsevier, 2001.
- [McI02] A.K. McIver. Quantitative program logic and expected time bounds in probabilistic distributed algorithms. *Theoretical Computer Science*, 282(1):191–219, 2002.
- [MG90] C.C. Morgan and P.H.B. Gardiner. Data refinement by calculation. *Acta Informatica*, 27:481–503, 1990. Reprinted in [MV94].
- [MM] C.C. Morgan and A.K. McIver. Proofs for Chapter 11. Draft presentations of the full proofs can be found at [MMSS, key GAMES02].
- [MM97] C.C. Morgan and A.K. McIver. A probabilistic temporal calculus based on expectations. In Groves and Reeves [GR97], pages 4–22. Available at [MMSS, key PTL96].
- [MM99a] C.C. Morgan and A.K. McIver. An expectation-based model for probabilistic temporal logic. *Logic Journal of the IGPL*, 7(6):779–804, 1999. Available at [MMSS, key MM97].
- [MM99b] C.C. Morgan and A.K. McIver. *pGCL*: Formal reasoning for random algorithms. *South African Computer Journal*, 22, March 1999. Available at [MMSS, key pGCL].
- [MM01a] A.K. McIver and C.C. Morgan. Demonic, angelic and unbounded probabilistic choices in sequential programs. *Acta Informatica*, 37:329–54, 2001. Available at [MMSS, key PPT2].
- [MM01b] A.K. McIver and C.C. Morgan. Partial correctness for probabilistic programs. *Theoretical Computer Science*, 266(1–2):513–41, 2001. Available at [MMSS, key PCFPDP].
- [MM01c] C.C. Morgan and A.K. McIver. Cost analysis of games using program logic. In *Proc. of the 8th Asia-Pacific Software Engineering Conference (APSEC 2001)*, December 2001. Abstract only: full text available at [MMSS, key MDP01].
- [MM02] A.K. McIver and C.C. Morgan. Games, probability and the quantitative μ -calculus $q\mu$. In *Proc. LPAR*, volume 2514 of *LNAI*, pages 292–310. Springer-Verlag, 2002. Revised and expanded at [MM04b].
- [MM03] C.C. Morgan and A.K. McIver. Almost-certain eventualities and abstract probabilities in the quantitative temporal logic *qTL*. *Theoretical Computer Science*, 293(3):507–34, 2003. Available at [MMSS, key PROB-1]; earlier version appeared in CATS '01.
- [MM04a] A.K. McIver and C.C. Morgan. An elementary proof that Herman's Ring has complexity $\Theta(N^2)$. Available at [MMSS, key HR04], 2004.

- [MM04b] A.K. McIver and C.C. Morgan. Results on the quantitative μ -calculus $qM\mu$. To appear in *ACM TOCL*, 2004.
- [MMH03] A.K. McIver, C.C. Morgan, and Thai Son Hoang. Probabilistic termination in B. In D. Bert, J.P. Bowen, S. King, and M. Waldén, editors, *ZB 2003: Formal Specification and Development in Z and B*, volume 2651 of *LNCS*, pages 216–39. Springer-Verlag, 2003.
- [MMS96] C.C. Morgan, A.K. McIver, and K. Seidel. Probabilistic predicate transformers. *ACM Transactions on Programming Languages and Systems*, 18(3):325–53, May 1996.
doi.acm.org/10.1145/229542.229547.
- [MMS00] A.K. McIver, C.C. Morgan, and J.W. Sanders. Probably Hoare? Hoare probably! In J.W. Davies, A.W. Roscoe, and J.C.P. Woodcock, editors, *Millennial Perspectives in Computer Science*, Cornerstones of Computing, pages 271–82. Palgrave, 2000.
- [MMSS] A.K. McIver, C.C. Morgan, J.W. Sanders, and K. Seidel. Probabilistic Systems Group: Collected reports.
web.comlab.ox.ac.uk/oucl/research/areas/probs.
- [MMSS96] C.C. Morgan, A.K. McIver, K. Seidel, and J.W. Sanders. Refinement-oriented probability for CSP. *Formal Aspects of Computing*, 8(6):617–47, 1996.
- [MMT98] A.K. McIver, C.C. Morgan, and E. Troubitsyna. The probabilistic steam boiler: a case study in probabilistic data refinement. In J. Grundy, M. Schwenke, and T. Vickers, editors, *Proc. International Refinement Workshop, ANU, Canberra*, Discrete Mathematics and Computer Science, pages 250–65. Springer-Verlag, 1998. Available at [MMSS, key STEAM96].
- [Mon01] David Monniaux. *Analyse de programmes probabilistes par interprétation abstraite*. Thèse de doctorat, Université Paris IX Dauphine, 2001. Résumé étendu en français. Contents in English.
- [Mor87] J.M. Morris. A theoretical basis for stepwise refinement and the programming calculus. *Science of Computer Programming*, 9(3):287–306, December 1987.
- [Mor88a] C.C. Morgan. Auxiliary variables in data refinement. *Inf. Proc. Lett.*, 29(6):293–6, December 1988. Reprinted in [MV94].
- [Mor88b] C.C. Morgan. The specification statement. *ACM Transactions on Programming Languages and Systems*, 10(3):403–19, July 1988. Reprinted in [MV94].
- [Mor90] J.M. Morris. Temporal predicate transformers and fair termination. *Acta Informatica*, 27:287–313, 1990.
- [Mor94a] C.C. Morgan. The cuppest capjunctive capping, and Galois. In Roscoe [Ros94], pages 317–32.
- [Mor94b] C.C. Morgan. *Programming from Specifications*. Prentice-Hall, second edition, 1994.
- [Mor96] C.C. Morgan. Proof rules for probabilistic loops. In He Jifeng, John Cooke, and Peter Wallis, editors, *Proceedings of the BCS-FACS 7th*

- Refinement Workshop*, Workshops in Computing. Springer-Verlag, July 1996.
ewic.bcs.org/conferences/1996/...refinement/papers/paper10.htm.
- [Mor97] J.M. Morris. Nondeterministic expressions and predicate transformers. *Inf. Proc. Lett.*, 61(5):241–6, 1997.
- [MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- [MV94] C.C. Morgan and T.N. Vickers, editors. *On the Refinement Calculus*. FACIT Series in Computer Science. Springer-Verlag, Berlin, 1994.
- [NCI99] N. Narasimha, R. Cleaveland, and P. Iyer. Probabilistic temporal logics via the modal mu-calculus. In *Proceedings of the Foundation of Software Sciences and Computation Structures, Amsterdam*, number 1578 in LNCS, pages 288–305, 1999.
- [Nel89] G. Nelson. A generalization of Dijkstra’s calculus. *ACM Transactions on Programming Languages and Systems*, 11(4):517–61, October 1989.
- [PRI] PRISM. Probabilistic symbolic model checker.
www.cs.bham.ac.uk/~dxp/prism.
- [PZ93] A. Pnueli and L. Zuck. Probabilistic verification. *Information and Computation*, 103(1):1–29, March 1993.
- [Rab76] M. O. Rabin. Probabilistic algorithms. In J. F. Traub, editor, *Algorithms and Complexity: New Directions and Recent Results*, pages 21–39. Academic Press, 1976.
- [Rab82] M.O. Rabin. The choice-coordination problem. *Acta Informatica*, 17(2):121–34, June 1982.
- [Rao94] J.R. Rao. Reasoning about probabilistic parallel programs. *ACM Transactions on Programming Languages and Systems*, 16(3):798–842, May 1994.
- [Ros94] A.W. Roscoe, editor. *A Classical Mind: Essays in Honour of C.A.R. Hoare*. Prentice-Hall, 1994.
- [Roy68] H.L. Royden. *Real Analysis*. MacMillan, second edition, 1968.
- [Sch86] A. Schrijver. *Theory of Integer and Linear Programming*. Wiley, New York, 1986.
- [SD80] N. Saheb-Djahromi. CPO’s of measures for nondeterminism. *Theoretical Computer Science*, 12(1):19–37, 1980.
- [Seg95] Roberto Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, MIT, 1995.
- [Sha53] L. S. Shapley. Stochastic games. In *Proc. National Academy of Sciences USA*, volume 39, pages 1095–1100, 1953.
- [SHRT04] S. Schneider, T.S. Hoang, K.A. Robinson, and H. Treharne. Tank monitoring: a case study in *pAMN*. Technical Report CSD-TR-03-17, Royal Holloway College, 2004. In draft.

- [SL95] Roberto Segala and Nancy Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–73, 1995.
- [SMM] K. Seidel, C.C. Morgan, and A.K. McIver. Probabilistic imperative programming: a rigorous approach. Revises [SMM96]; available at [MMSS, key SMM96]; extended abstract appears in [GR97].
- [SMM96] K. Seidel, C.C. Morgan, and A.K. McIver. An introduction to probabilistic predicate transformers. Technical Report PRG-TR-6-96, Programming Research Group, February 1996. Available at [MMSS, key SMM96].
- [Smy78] M.B. Smyth. Power domains. *Jnl. Comp. Sys. Sciences*, 16:23–36, 1978.
- [Smy89] M.B. Smyth. Power domains and predicate transformers: a topological view. In *Automata, Languages and Programming 10th Colloquium, Barcelona, Spain*, volume 298 of *LNCS*, pages 662–75. Springer-Verlag, 1989.
- [SPH84] M. Sharir, A. Pnueli, and S. Hart. Verification of probabilistic programs. *SIAM Journal on Computing*, 13(2):292–314, May 1984.
- [Spi88] J.M. Spivey. *Understanding Z: a Specification Language and its Formal Semantics*. Cambridge University Press, 1988.
- [Sti92] C. Stirling. Modal and temporal logics. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science, Volume 2: Computational Structures*, pages 478–551. Clarendon Press, 1992.
- [Sti94] David Stirzaker. *Elementary Probability*. Cambridge University Press, 1994.
- [Sti95] C. Stirling. Local model checking games. In *CONCUR '95*, volume 962 of *LNCS*, pages 1–11. Springer-Verlag, 1995. Extended abstract.
- [Sto88] Allen Stoughton. *Fully Abstract Models of Programming Languages*. Research Notes in Theoretical Computer Science. Pitman/Wiley, 1988.
- [Sto96] N. Storey. *Safety-Critical Computer Systems*. Addison-Wesley, 1996.
- [Sut75] W. Sutherland. *An Introduction to Metric and Topological Spaces*. Oxford University Press, 1975.
- [SV03] M.I.A. Stoelinga and F.W. Vaandrager. A testing scenario for probabilistic automata. In *Proceedings ICALP '03*, volume 2719 of *LNCS*, pages 407–18. Springer-Verlag, 2003.
- [Tar55] A. Tarski. A lattice-theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.
- [Tru71] K. Trustrum. *Linear Programming*. Library of Mathematics. Routledge and Kegan Paul, London, 1971.
- [Tur84] R. Turner. *Logics for AI*. Artificial Intelligence. Ellis Horwood, 1984.

- [Van98] B. Van Roy. *Learning and Value Function Approximation in Complex Decision Processes*. PhD thesis, EECS, MIT, 1998.
- [Var85] M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proc. 26th IEEE Symp. on Foundations of Computer Science*, pages 327–38, Portland, October 1985.
- [Var01] M.Y. Vardi. Branching vs. linear time: Final showdown. In T. Margaria and Wang Yi, editors, *Seventh International Conference on Tools and Analysis of Systems, Genova*, number 2031 in LNCS, pages 1–22, April 2001.
- [vBMOW03] Franck van Breugel, Michael W. Mislove, Joel Ouaknine, and James Worrell. An intrinsic characterization of approximate probabilistic bisimilarity. In *Proceedings of FOSSACS 2003*, number 2620 in LNCS, pages 200–15. Springer-Verlag, 2003.
- [vGB98] A.J.G van Gasteren and A. Bijlsma. An extension of the program derivation format. In D. Gries and W.P. de Roever, editors, *PROCOMET*, volume 125 of *IFIP Conference Proceedings*, pages 167–85. Chapman & Hall, 1998.
- [vNM47] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, second edition, 1947.
- [War89] M. Ward. *Proving Program Refinements and Transformations*. PhD thesis, Programming Research Group, 1989.
- [Yin02] Mingsheng Ying. Bisimulation indexes and their applications. *Theoretical Computer Science*, 275(1–2):1–68, 2002.
- [Yin03] Minsheng Ying. Reasoning about probabilistic sequential programs in a probabilistic logic. *Acta Informatica*, 39(5):315–89, 2003.
- [YL92] W. Yi and K.G. Larsen. Testing probabilistic and nondeterministic processes. In R.J Linn and M. Ümit Uyar, editors, *Proceedings of 12th IFIP International Symposium on Protocol Specification, Testing and Verification, Florida, USA*, pages 47–61, 1992.
- [YW00] Mingsheng Ying and Martin Wirsing. Approximate bisimilarity. In Teodor Rus, editor, *AMAST*, volume 1816 of *LNCS*, pages 309–22. Springer-Verlag, 2000.

Index of citations

- [ABL96], 124, 345
- [Abr96a], 5, 108
- [Abr96b], 77, 325
- [AH90], 89
- [AL96], 124
- [ASBSV95], 263, 268
- [AZP03], 105, 216

- [Bac78], 5
- [Bac88], 5
- [BAPM83], 215, 246, 247, 249,
266, 268
- [BB96], 43, 47
- [BdA95], 180, 290, 291
- [BFL⁺99], 263
- [BKS83], 77
- [Boo82], 289
- [BvW90], 218, 227, 228, 238, 240
- [BvW93], 38
- [BvW96], 246, 262
- [BvW98], 5, 36, 110, 195, 242, 289

- [Chr90], 164
- [CM88], 77, 105, 215
- [CMA02], 216
- [Coh00], 242
- [CY95], 105

- [dA99], 295
- [dAH00], 78, 310
- [dAM01], 291, 309, 310
- [DFP01], 105
- [DFP02], 106
- [DGJP02], 6, 125
- [DGJP03], 164
- [dHdV02], 36, 77
- [Dij71], 88
- [Dij76], 4, 7, 19, 21, 28, 38, 71, 82, 83,
135, 145, 162, 182, 184, 198,
199, 218, 224, 249
- [DIM83], 310
- [DP90], 25
- [dRE98], 107

- [Eda95], 220
- [EH86], 247
- [Eme90], 246, 262
- [Eve57], 296, 310

- [Far99], 180
- [Fel71], 17, 73
- [FH84], 36, 77, 246
- [FHM90], 341
- [FHMV95], 245
- [Flo67], 5, 38, 77

- [Fra86], 105
 [FS03], 216
 [FV96], 164, 180, 263, 295, 304
- [GM91], 110
 [GM93], 242
 [Gol03], 245
 [GR97], xi, 351, 354
 [Gro], 110, 124
 [GS92], 16, 41, 54, 76, 100, 134, 284, 297
 [GW86], 52, 134, 222, 283, 284, 291, 303
- [Heh89], 64
 [Her90], 56
 [Hes92], 130, 142, 162, 163
 [Hes95], 252
 [HHS87], 108, 109
 [HJ94], 263
 [HK94], 78
 [HK97], 290, 310
 [HMM04], 162, 216, 291
 [Hoa69], 38, 77, 135, 184, 199
 [Hon03], 73
 [HP01], 93, 106
 [HP02], 18, 164
 [HS86], 216, 291
 [HSM97], 36, 137, 139, 164, 291, 316, 319, 322
 [HSP83], 78, 216
 [Hur02], 207, 210, 216
 [Hut03], 124, 164
- [JHSY94], 164
 [Jon86], 5
 [Jon90], 18, 36, 135, 137, 164, 196, 216, 238, 333
- [KB00], 164
 [Koz81], 4, 15, 18, 131, 135, 164, 319
 [Koz83], 239, 246, 252, 293, 310
 [Koz85], 4, 15, 36, 135
 [Kuh03], 174, 180, 262, 275, 341
- [Lam80], 247
 [Lam83], 247
 [LPZ85], 246
 [LR94], 89
 [LS82], 246
 [LS89], 164
 [LS91], 124, 164
 [LSS94], 164, 327
 [LvdS92], 252
- [McI01], 291
 [McI02], 88, 216, 268, 289
 [MG90], 124
 [MM01a], x
 [MM01b], x, 183, 184, 220, 238, 239, 297, 331
 [MM01c], 242, 295
 [MM02], x, 293, 304, 306, 307
 [MM03], 289, 295
 [MM04a], 58, 74, 215
 [MM04b], xi, 293, 351
 [MM97], x, 310
 [MM99a], x, 265
 [MM99b], x, 295
 [MMH03], 216, 235
 [MMS00], x
 [MMS96], ix, x, 129, 291
 [MMSS96], 138
 [MMSS], xi, 348, 349, 351, 352, 354
 [MMT98], ix, x, 107
 [MM], 306, 309
 [Mon01], 124
 [Mor87], 5, 138, 239
 [Mor88a], 113, 122
 [Mor88b], 5, 138, 239
 [Mor90], 252
 [Mor94a], 279
 [Mor94b], 5, 110
 [Mor96], x, 57
 [Mor97], 207
 [MR95], 35, 36, 47, 105
 [MV94], 348, 351, 352
- [NCI99], 310
 [Nel89], 138, 198, 239
- [PRI], 105, 304
 [PZ93], 105

- [Rab76], 36
[Rab82], 80, 295
[Rao94], 77, 105
[Ros94], 350, 352
[Roy68], 152

[Sch86], 341
[SD80], 36
[Seg95], 164, 318
[Sha53], 310
[SHRT04], 325
[SL95], 291
[SMM96], xi, 354
[SMM], x
[Smy78], 138
[Smy89], 138, 298
[SPH84], 268
[Spi88], 5
[Sti92], 246
[Sti94], 100
[Sti95], 246, 262, 294, 299, 300, 310
[Sto88], 319

[Sto96], 118, 124
[Sut75], 141
[SV03], 124

[Tar55], 38, 179
[Tru71], 341
[Tur84], 245

[Van98], 310
[Var01], 294, 309
[Var85], 78, 105, 164
[vBMOW03], 6, 125
[vGB98], 48
[vNM47], 262

[War89], 38

[Yin02], 125
[Yin03], 6, 36, 125
[YL92], 164
[YW00], 125

General index

Symbols are indexed in order of occurrence, and alphabetically if appropriate, referring to a spelled-out entry whose name may be sufficient to jog the memory.

Hierarchical topics are cross-referenced both up and down, with page-number references occurring at the tips. Thus “semantics” leads down to “expectation-transformer semantics” and then to “assignment,” where the expectation-transformer semantics of assignment is located on p. 7.

In the reverse direction, starting from “assignment” a reference leads back to “expectation-transformer semantics” (showing what other constructs also have that kind of semantics) and then up to “semantics” (showing what other kinds of semantics there are).

Bold page numbers indicate definitions; underlined numbers are named figures, definitions *etc.* A “qv” after a sub-item is a cross-reference to the main item beginning with those words.

$\{pred\}$, <i>see</i> assertion	$[\cdot]$, <i>see</i> characteristic function
$:\odot$, <i>see</i> assignment	$\bar{\cdot}$, <i>see</i> complement
$:=$, <i>see</i> assignment, definitions	$(Qx: set \cdot expr)$, <i>see</i> comprehension
\leq , <i>see</i> bag: bounded above	$(Qx: set pred)$, <i>see</i> comprehension
\square , <i>see</i> bag: empty	$(Qx: set pred \cdot expr)$, <i>see</i> comprehension
$\llbracket \cdot \rrbracket$, <i>see</i> bag: enumeration	$\{x: set pred \cdot expr\}$, <i>see</i> comprehension
$\#$, <i>see</i> bag: size	$\bar{\cdot}, \tilde{\cdot}$, <i>see</i> conjugate
$+$, <i>see</i> bag: sum	$\underline{0}, \underline{1}, \underline{\cdot}$, <i>see</i> constant function
\perp , <i>see</i> bottom element	
$\lceil \cdot \rceil$, <i>see</i> ceiling	

\sqcap, \in, \notin , *see* demonic choice
 $:\leq$, *see* demonic decrease
 \mathbb{D} , *see* deterministic program space
 $[\cdot]$, *see* embedding
 \overline{f} , *see* embedding of program
 \models , *see* entailment
 \mathbb{R}^N , *see* Euclidean space
 $\mathbb{E}, \mathbb{E}_B, \mathbb{E}_f$, *see* expectation
 $\alpha \downarrow P$, *see* expectation
 $\mathbb{T}, \mathbb{T}_r, \mathbb{T}_o, \mathbb{T}_\square, \mathbb{T}_s$, *see* expectation transformer space
 f , *see* expected value
 Exp, Exp_n , *see* expected value
 $!$, *see* factorial
 ω , *see* fixed point
 ν , *see* fixed point: greatest
 μ , *see* fixed point: least
 $\lfloor \cdot \rfloor$, *see* floor
 $f.x, f(x)$, *see* function application
 $f \circ g, f^n$, *see* functional composition
 $B \leftarrow A$, *see* function: reverse total
 $A \rightarrow B$, *see* function: total
 \mathcal{RS} , *see* gambling game
 \mathcal{C} , *see* gambling game: colour
 $\underline{\sigma}, \overline{\sigma}$, *see* gambling strategy
 $\phi_{\mathcal{G}}, \phi_{\overline{\mathcal{G}}}$, *see* gambling strategy
 $[[\phi]]$, *see* game tree
 wp , *see* greatest pre-expectation
 $p \vdash \{Q\} \text{ prog } \{R\}$, *see* Hoare triple
 \Rightarrow , *see* implication
 \sqcap_P , *see* infimum
 ∞ , *see* infinity
 \mathbb{Z} , *see* integers
 $[a..b)$, *see* interval
 \cdot^* , *see* Kleisli composition
 λ , *see* lambda notation
 \dagger, \ddagger , *see* local cross-reference
 $\sqcup\beta$, *see* maximum
 \sqcup , *see* maximum
 \min , *see* minimum
 \mathbb{N} , *see* natural numbers
 \perp , *see* nontermination state
 \overline{s} , *see* point distribution
 \mathbb{F}, \mathbb{F}^+ , *see* powerset
 \mathbb{P} , *see* powerset
 t^+, t^- , *see* predicate transformer embedding
 $@$, *see* probabilistic choice

${}_p\oplus, \oplus_p, \geq_p\oplus, {}_p\oplus_q$, *see* probabilistic choice
 \parallel , *see* probabilistic comprehension
 $\&$, *see* probabilistic conjunction
 $\Rightarrow, \equiv, \Leftarrow$, *see* probabilistic implication
 $\Rightarrow, \dashv\rightarrow$, *see* probabilistic implication
 \mathbb{HS} , *see* probabilistic relational semantics
 Δ , *see* probability distribution
 Pr, Pr_n , *see* probability distribution
 \sqsubseteq, \sqsubset , *see* probability distribution: refinement
 \prod , *see* product
 $\|\phi\|, \|\phi\|_\mu$, *see* quantitative modal μ -calculus
 α, β , *see* random variable
iid, *see* random variable
 $\mathbb{R}, \mathbb{R}_{\geq}$, *see* real numbers
 \mathbb{SS} , *see* relations
 \mathcal{C} , *see* sets of distributions
 $(\cdot \surd k)$, *see* shift by k
 S, S_\perp, S_a, S_b , *see* state space
 \overline{S} , *see* sub-distribution space
 $\cdot \langle \cdot \mapsto \cdot \rangle$, *see* substitution
 $\mathcal{C}(OP)$, *see* substitution: implicit
 \ominus , *see* subtraction
 \sum , *see* summation
 $\forall X, \exists X$, *see* temporal logic
 $\circ\circ$, *see* temporal logic right-association
 $\|\cdot\|_{\mathcal{V}}$, *see* temporal logic semantics
 \mathcal{V} , *see* valuation
 $\cdot \equiv$, *see* weakest precondition
 wp , *see* weakest precondition
 $0-1$, *see* Zero-One
 \square, \mathbf{G} , *see* *always*
 \diamond, \mathbf{F} , *see* *eventually*
 \circ, \mathbf{X} , *see* *next-time*
 \triangleright , *see* *unless*
 rp , *see* wp

A —

$\forall X$, *see* temporal logic

Åbo Akademi, ix

abort

is bottom (\perp) among programs, 131, 133

is (pre-)deterministic, 131

“jail” metaphor for, 12

- not normalised, 18
- not *wlp*-scaling, 184
- refined by any behaviour, 285
- syntax, **19**
- see also*
 - complementary... semantics,
 - expectation... semantics,
 - informal semantics,
 - liberal... semantics
- aborting behaviour
 - called “divergence”, 249
 - in multi-way choice @, 20
 - in *next-time* operator, 249, 251
 - indicated by one-deficit, 20, 112
 - removed by refinement, 172
- J.-R. Abrial, 5ff, 6, 77, 84, 114, 216
- abstract, probabilistic choice *qv*
- abstract interpretation, 124
- abstraction, 5, 36
 - complementary to refinement,
 - see* refinement
 - full *qv*
 - relation, 124
 - see also* demonic choice
- action systems, 77
- add n to b** , *see* bag
- adjoint, **279**
 - conjunction and implication, 279
 - see also* probabilistic conjunction,
 - probabilistic implication
- adversarial scheduling, *see* scheduler
- agent
 - angelic or demonic, **275**
 - maximising or minimising, 180
- L. de Alfaro, *see* de Alfaro
- algebra, 265ff
 - Borel, *see* σ -algebra
 - Kleene, viii
 - modal, 262
 - omega *qv*
 - program *qv*
 - σ - *qv*
 - temporal, *see* quantitative... logic
- algebraic properties of transformers, 241
- almost
 - certain, *see* termination
 - fair, *see* scheduler
 - impossible, **41**, 44, 263
 - none, 297
- α , *see* random variable
- α -component, **229**
- alternating fixed points, 277, 294
- alternation-free formulae,
 - see* temporal logic
- always* (\square, G), 248
 - as a game, 261ff
 - as iteration, 250
 - illustrated by coin flips, 273–4
 - informal standard semantics, 250
 - quantitative, **253**
 - special case of *unless*, 251
 - standard, **252**
- amplification, probabilistic, 46
- C. Andriessens, 124
- angelic
 - choice operator, **227**
 - interaction with demonic choice,
 - 246, 271
 - nondeterminism, 78, 242
 - programs not sublinear, 231
 - programs semi-sublinear, 232
 - transformers, 227–31, 247
- anti-refinement, 80
- anti-symmetric, **25**
- approximate, refinement *qv*
- J. Aspnes, 89
- assertion ($\{pred\}$)
 - as command, **110**, 225
 - as comment, 4ff, 209
- assignment
 - generalised ($:\odot$), **114**
 - syntax, **19**
 - written $:=$, **7**
 - see also*
 - expectation... semantics,
 - informal semantics
- associativity, quasi-, **322**
 - see also* probabilistic conjunction
- asynchronous network of processors, 60ff
- atomic action, 92
- automata
 - Büchi, 291
 - I/O, 164
- automated reasoning,
 - see* higher-order logic

- auxiliary invariant, *see* reasoning
 within an invariant
 average, 16, 255
 see also expected value
 Axiom of Choice, 25
 Axioms for standard branching-time
 temporal logic, 266
 A. Aziz, 263
- \mathbb{B} —
 The *B* Method, 216
 abstract-probabilistic (*qB*), 235
 probabilistic (*pAMN*), 325
 R.-J.R. Back, ix, x, 36, 38, 77, 195,
 218, 228, 238, 242, 246, 262
 bag (multiset)
 add n to b , **82**
 bounded above (\leq), **83**
 empty bag (\square), **82**
 enumeration ($\llbracket \cdot \rrbracket$), **81**
 maximum of (**max**), **84**
 size of (**#**), **82**
 sum (**+**), **82**
 take n from b , **82**
 C. Baier, 164
 basketball, 174
 M. Ben-Ari, 266, 280, 291
 B. Bérard, 263
 β , *see* random variable
 A. Bianco, 180
 bisimulation, 124ff, 164
 D. Bjørner, 5
 Borel algebra, *see* σ -algebra
 bottom element (\perp), 179
 of a partial order, **133**
 bound variables, scope always
 delimited explicitly, 16, 61, 72,
 131, 155, 194, 247
 bounded
 expectation *qv*
 invariant *qv*
 bounded monotone convergence, 335
 bounded nondeterminism,
 see continuity
BPP, *see* complexity class
 F. van Breugel, *see* van Breugel
 Büchi automata, 291
- \mathbb{C} —
 C, *see* gambling game: colour
 \mathbb{CS} , *see* sets of distributions
 Cambridge University,
 Computer Laboratory, x
 card-and-dice game
 operational semantics for *pGCL*, 14
 see also Programs
 D. Carrington, ix
 casino manager, *see* demonic choice
 Cauchy closure, **139**
 see also sets of distributions
 ceiling ($\lceil \cdot \rceil$), **99**, **193**
 Orieta Celiku, x
 chain, 298 *see also* complete partial
 order
 K.M. Chandy, 77
chaos, **289**, 335
 characteristic function ($\llbracket \cdot \rrbracket$), 13, 224
 converts to expectation, **7**
 expected value of, 13, 16, 254, 316
 see also embedding
 choice coordination, *see* Programs
 I. Christoff, 164
 classical logic, as opposed to
 temporal, 247
 closed set, *see* Euclidean space
 closure, *see* sets of distributions
 clump
 can be smooth, 338ff
 see also sets of distributions
 E. Cohen, 242
 coin-flipping
 fair *qv*
 illustration of *always* \square , 274
 illustration of *eventually* \diamond , 272
 illustration of *next-time* \circ , 270
 implementation of $p\oplus$, 210 *see also*
 Programs
 thin and fat coins, 268ff
 three coins, 72
 three-sided, 269
 two coins, 327ff
 colour (C)
 μ, ν -generated, 306ff
 see also gambling game
 commutativity, quasi-, **322**
 compactness, *see* Euclidean space
 complement ($\bar{\cdot}$)

- for Booleans, **7**
- for probabilities, **7, 26**
- complementary and consistent semantics, 154, 159ff, 160
- abort**, 179
- demonic choice, 156
- recursion, 178
- sequential composition, 154
see also semantics
- complete partial order, **25**, 148, 179, 185
 - abbreviated *cpo*, 25, 139
 - bottom element (\perp), **133**
 - chain in, **25**
 - CS is, 139
 - directed subset of, **25**, 139, **147**, 234, 235
 - DS is, 131
 - ES is if one-bounded, 183, 299
 - ES is not in general, 24, 25, 148
 - flat, **142**
 - HS is, 139
 - S_{\perp} is, 142
 - \bar{S} is, 131
 - top element, 183
 - TS is if one-bounded, 183
 - TS is not in general, 24, 148
- completeness of variant rule, 193
for probabilistic guards, 205
- complexity, expected, 64, 268
- complexity class
 - BPP, 47
 - ZPP, 47
- composition, functional *qv*
- compositionality
 - counter-example, 315
 - requires full use of expectations, 15, 313ff, 315
- comprehension
 - $(Qx: set \bullet expr)$, **61**
 - $(Qx: set \mid pred)$, **194**
 - $(Qx: set \mid pred \bullet expr)$, **155**, 229
 - $\{x: set \mid pred\}$, **72**
 - $\{x: set \mid pred \bullet expr\}$, **140**
 - probabilistic *qv*
- computability, *see* real numbers
- computational model, *see* semantics
- computations, unending, 21, 268
- concurrency, 77, 164, 310
see also bisimulation
- concurrent game, 78
see also concurrency
- conditional (**if, then, else, fi**)
 - defined in terms of $_p \oplus$, **19**
 - hybrid with then/else and guards combined, **82**
see also informal semantics
- confidence measure, *see* statistics
- conjugate ($\bar{\cdot}$)
 - in choice-coordination program, **82**
 - minimum of ($\bar{\cdot}$), **82**
- conjunctivity, 6, **28**, 145, 161, 162
 - generalised by sublinearity, 28, 31, 221
 - implied by sub-conjunctivity, 31, 237
 - in temporal logic, 249
 - is complete, 162ff
 - of modal algebra, 295
 - of standard transformers, 31, 237
 - of *next-time*, 262
 - positive, **142**, 161, 163, **249**
 - sub-, *see* probabilistic conjunction
see also healthiness conditions
- constant function ($\underline{\cdot}$), **17**
- continuity, 139, 145, **147**
 - and fixed points, 179
 - bounded, 147
 - bounded nondeterminism, 71, 220, 335–7
- chaos** is not, 289
- convenient assumption, 26, 289
- distributes limits, 147, 148
- ε - δ style, 147
- explicit assumption, 219ff
- fails for retraction $(\bar{\cdot})^-$, 235
- image finiteness, 21, 71
- implied by sublinearity, 147, 234
- its purpose, 148
- not imposed, 288, 289
- of $\sqcap_P(\bar{\cdot})$, 234
- of expectation transformers, 62, 147, 178, 333ff
- of predicate transformers, 143
- of transformer-transformers, 178
- picture of, 175

- preserved by suprema of finitary infima, **225**
 - topological- agreeing with \sqsubseteq -, **298**
 - unbounded nondeterminism, **71**, **220**, **335–7**
 - see also* healthiness conditions
 - convexity, **138**
 - picture of, **168**
 - see also* sets of distributions
 - countable
 - closure of σ -algebra, **297**
 - state space qv
 - coupling invariant,
 - see* data refinement
 - C. Courcoubetis, **105**
 - cowboys, *see* Duelling Cowboys
 - cpo*, *see* complete partial order
 - customer-oriented, data refinement qv
- D** —
- $\mathbb{D}S$, *see* deterministic program space
 - data refinement, **57**, **64**, **107–24**, **209**, **242**
 - as sub-commutation, **109**
 - customer- *vs.* supplier-oriented, **108**, **113**, **121**
 - defined, **109**
 - intermediate step, **122**
 - is transitive, **122**
 - of *eventually* \diamond , **342**
 - of *next-time*, **286**
 - proved by simulation, **109**
 - via coupling invariant, **122**, **209**, **210**
 - datatype, **107ff**, **108**
 - abstract, **108**, **110**
 - concrete, **108**, **110**
 - L. de Alfaro, **78**, **180**, **291**, **310**
 - W.-P. de Roever, **107**
 - E.P. de Vink, **36**, **77**
 - deadlock
 - dining philosophers, **88**
 - impossible in Rabin and Lehmann’s algorithm, **92**
 - defiance probability, **89**
 - definitions, written $:=$, **6**
 - Δ , *see* probability distribution
 - demonic choice (\sqcap), **4ff**
 - blind-choice metaphor, **12**
 - casino manager, **261**
 - from empty set, **21**
 - implemented by a demon, **9**, **26**
 - indicates abstraction, **5**, **9**, **36**, **45**
 - inherent in $\geq_p \oplus$, **50**, **275**, **326**
 - interaction with angelic, **246**, **271**
 - interaction with probabilistic choice, **36**, **51**, **164**, **254**, **326**
 - interaction with probabilistic choice, avoiding, **328**
 - means pre-expectation is only a lower bound, **23**, **65**
 - non-Markovian, **284**
 - not factored out, **284**
 - picture of, **168**
 - refined by probabilistic choice, **10**, **26**, **45**, **80**, **139**, **220**
 - syntax, **19**
 - tries to minimise pre-expectation, **144**
 - written \in , \notin , **21**
 - see also*
 - complementary... semantics, expectation... semantics, informal semantics, nondeterminism, probabilistic relational... , Programs
 - demonic decrease ($:\leq$), **114**
 - demonic expressions, **207**
 - demonic probabilistic model, **139**
 - Demonic program
 - over finite state space, **226**
 - sublinear but not additive, **226**
 - see also* Programs
 - J.J. den Hartog, **36**, **77**
 - J. Desharnais, **125**
 - deterministic
 - includes probabilistic, **24**, **27**
 - liberally, **130**
 - only pre- if nonterminating, **27**, **130**, **222**
 - deterministic program, **130–3**, **221–4**
 - additive, **224**
 - characterised by linearity, **28**, **66**, **221–4**
 - definition, **222**
 - embedded, **132**
 - example, *see* Programs
 - has repeatable behaviour, **12**, **135**

- is \sqsubseteq -maximal, 133
 - need only be tested at points, 112
 - picture of, 166–8
 - probabilistic, **131**
 - probabilistic depicted, **167**
 - space (\mathbb{D}), **131**
 - standard, 221
 - see also*
 - probabilistic relational. . . ,
 - standard relational. . .
 - E.W. Dijkstra, 4, 5, 6, 21, 38, 135ff, 145ff, 158, 162, 175, 199, 238
 - style reasoning, 4
 - dining-philosophers problem *qv*
 - see also*
 - Guarded Command Language
 - dining philosophers, **90**, 88–98
 - generalised, 106
 - in contention, **91**
 - variant function for, **98***see also* Programs
 - directed set,
 - see* complete partial order
 - discrete
 - sub-probability measure, 36, 131
 - see also* probability distribution
 - disjoint union, 213
 - disjunctivity, 28, 224 *see also* linearity
 - distribution, probability *qv*
 - distributive, quasi-, **323**
 - divergence, *see* aborting behaviour
 - do**, *see* iteration
 - do-it-yourself semantics,
 - see* semantics
 - S. Dolev, 310
 - domain equation, 164
 - Duelling Cowboys, **211**
 - see also* Programs
 - M. DufLOT, 105, 106
 - dyadic rational,
 - see* probabilistic choice
- \mathbb{E} —
- $\mathbb{E}, \mathbb{E}_B, \mathbb{E}_f$, *see* expectation
 - $\exists X$, *see* temporal logic
 - A. Edalat, 220
 - efficiency, increased in random
 - algorithms, v
 - see also* probabilistic amplification
 - else**, *see* conditional
 - embedding
 - [.] omitted, 209, 211, 316
 - [false], [true], **7**
 - \bar{f} of a standard deterministic program *f*, **132**
 - of \models is \Rightarrow , 20
 - predicates to expectations ([.]), 5, **19** *see also* characteristic function
 - relational-to-transformer, **144**
 - standard, **232**
 - K. Engelhardt, 107
 - entailment (\models), **20**
 - also written \Rightarrow , 39, 267
 - see also* probabilistic implication
 - environment, *see* valuation
 - equivalence (\equiv)
 - differs from $=$, 19
 - see also* probabilistic implication
 - equivalence of given-strategy games
 - and logic, **306**
 - equivalence of *qMu* and games, 308
 - Euclidean space (\mathbb{R}^N), 139ff, 149, 165–80
 - closed subset of, **139**
 - compact subset of, 141, 152, 179, 180, 221, 334
 - finitary half-space closed, **333**
 - infinitary half-space closed, **333**
 - infinitary half-space not closed, 334
 - metric for, 332
 - of infinite dimension, 130, 332
 - program states are dimensions, 332
 - topology for, 332ff
 - evaluation, 36 *see also* probability distribution
 - event, **16**
 - not every subset allowed, 16, 297
 - Event-B*, 77
 - eventually* (\diamond, F), 248, 342–4
 - as a game, 255ff
 - as iteration, 249
 - composition of several, 94, 95
 - double- equals single, 267, 273, 278
 - illustrated by coin flips, 271–3

- informal standard semantics, 249–50
 - quantitative, **253**
 - standard, **251**
 - ◇ excluded miracle, **343**
 - ◇ monotonicity, **343**
 - ◇ scaling, **343**
 - H. Everett, 296, 310
 - exact, invariant qv
 - existence of fixed point, **148**
 - exotic, expectation transformer qv
 - Exp, Exp_n , *see* expected value
 - expectation, 6, **7**
 - bounded above, 13, 25, 66, 68ff
 - bounded space (\mathbb{E}_B), **332**
 - called probabilistic predicate, 13
 - expressions *vs.* functions, 25, 130, 136, 183
 - finitary, 220, **332**
 - finitary restriction ($\alpha \downarrow P$), **220, 332**
 - finitary space (\mathbb{E}_f), **332**
 - “glue” between reasoning steps, 24, 271
 - greatest pre- qv
 - idiom for nonzero, 99
 - intuition behind, 22ff
 - is random variable, 17
 - mixed-sign, to be avoided, 70
 - negative, 100
 - non-negative, 24, 66, 68ff
 - not confused with distribution, 137
 - one-bounded, 31, 181–215, 247, 252, 296
 - post-, 13, **17**
 - pre-, 13, **17**
 - proper, **63**, 231, 253, 254, 256, 268, 269, 271, 273
 - “safe”, 34
 - space (\mathbb{E}), **24, 143**
 - standard, **40, 137**
 - support of, **332**
 - unbounded, 34, 216
 - see also* random variable
 - expectation transformer, 15
 - angelic space (\mathbb{T}_{\sqcap}), **227**
 - continuity not imposed qv
 - demonic space (\mathbb{T}_{\sqcap}), **225**
 - deterministic space (\mathbb{T}_{\circ}), **223**
 - does not distribute min, 30
 - exotic, **238**, 240, 242
 - hierarchy, 218, 221–38
 - infeasible, 138, 238, 250
 - linear, **222**
 - regular, 141–64, 219
 - regular space (\mathbb{T}_r), **145**, 219ff
 - semi-linear, **234**
 - semi-sublinear, **228**
 - space (\mathbb{T} , *PTS*), **24**, 157ff, 334ff
 - standard space (\mathbb{T}_s), **232**
 - standard-preserving, **235**
 - sub-conjunctive, **31**
 - sublinear, **28, 146**
 - expectation-transformer semantics, 142, 157
 - assignment $:=$, **7**
 - demonic choice \sqcap , **9**
 - iteration, **38**
 - of *pGCL*, including **abort**, **skip** and recursion, **26**
 - probabilistic choice $_p\oplus$, **7**
 - sequential composition, **15**
 - variants of $_p\oplus$, **20**
 - see also* expectation...space, semantics
 - expected, complexity qv
 - expected value (Exp), **16, 134**
 - explicit in formulae, 316ff
 - picture of, 173
 - same as average, 5, 13, 134
 - written \int , **134**
 - Exp_n , in proof of loop rules, **76**
 - see also* average, expectation
- \mathbb{F} —
- \mathbb{F} , *see* eventually
 - \bar{f} , *see* embedding of program
 - \mathbb{F}, \mathbb{F}^+ , *see* powerset
 - factorial (!), **51**
 - fail** command, in steam boiler, 119ff
 - fair coin (flipping), 316
 - nearly, 6
 - reasonably, 269
 - fairness, **89**, 93–5, 105ff, 263
 - almost, *see* scheduler
 - k -fairness, **60**
 - probabilistic, **61**
 - Farkas’ Lemma, 130, 180, **341**

- use of, 152
 - fat* coin, *see* coin-flipping
 - fault-tolerance, ν *see also* Programs:
 - steam boiler
 - faulty 2-skipper, [111](#)
 - faulty N -skipper, [111](#)
 - faulty factorial, [52](#) *see also* Programs
 - faulty skipper
 - for $p := 1/2$, [115](#)
 - see also* Programs
 - feasibility, [29](#), [147](#), [238](#)
 - eventually is*, [257](#)
 - example of use, [183](#), [187](#), [257](#), [296](#)
 - excludes miracles, [29](#)
 - generalises strictness, [29](#), [221](#)
 - in temporal logic, [249](#)
 - more general treatment without it, [144](#), [238](#)
 - of transformers, [147](#)
 - picture of, [175](#)
 - preserves one-boundedness, [183](#), [296](#)
 - preserving, [26](#), [148](#)
 - standard, [249](#)
 - two kinds for probability, [239](#)
 - wnp* is not, [250](#)
 - see also*
 - expectation transformer,
 - healthiness conditions,
 - miracle,
 - strictness
 - feature interactions, [195](#)
 - need semantics to avoid, [197](#)
 - Y.A. Feldman, [36](#), [77](#)
 - fi**, *see* conditional
 - Fibonacci numbers, [115](#)
 - C.J. Fidge, [216](#)
 - finite state space, *see* state space
 - Finite-Intersection Lemma, [152](#), [180](#), [336](#), [337](#)
 - Firewire*, *IEEE* protocol, [216](#)
 - first-order logic, [4](#)
 - vs.* second-order, [38](#)
 - fixed point, [77](#)
 - alternating qv
 - appeal to continuity, [148](#), [179](#)
 - avoids circular argument, [197](#)
 - general $f.x \geq x$ property, [185](#), [278](#)
 - general $f.x \leq x$ property, [102](#), [278](#)
 - greatest (ν), [183](#), [184](#), [185](#), [251](#), [300](#)
 - is a limit, [179](#), [259](#)
 - least (μ), [21](#), [26](#), [38](#), [61](#), [102](#), [177](#), [179](#), [251](#), [299](#), [300](#)
 - N - or ω -limit formulation, [178](#), [179](#), [288](#)
 - special proof for existence of, [148](#)
 - flat *cpo*, *see* complete partial order
 - flip-a-coin
 - generates $p \oplus$, [209](#)
 - see also* Programs
 - floor ($\lfloor \cdot \rfloor$), [193](#)
 - R.W. Floyd, [5](#), [6](#), [77](#)
 - four semantic models, [143](#)
 - free will, [275](#)
 - fruit machine, *see* poker machine
 - full abstraction, facilitates program algebra, [319](#)
 - function
 - characteristic qv
 - constant, [17](#)
 - generating, [291](#)
 - homogeneous, [135](#)
 - measurable, [305](#)
 - reverse total ($B \leftarrow A$), [24](#), [142](#)
 - strict, [133](#), [143](#)
 - total ($A \rightarrow B$), [130](#)
 - function application ($f.x$), [6](#), [248](#)
 - not written $f(x)$, [6](#), [53](#)
 - functional composition ($f \circ g$), [154](#)
 - n -fold (f^n), [179](#)
 - functional properties, refinement qv
 - futures market, example, [304](#)
- \mathbb{G} —
- G**, *see always*
 - Galois connection, [226](#), [279](#)
 - between \mathbb{HS} and \mathbb{TS} , [334](#)–[8](#)
 - see also* adjoint
 - gambling game, [6](#), [11](#)–[15](#), [246](#), [254](#)–[62](#), [293](#)–[309](#)
 - colours C used for fixed points, [299ff](#)
 - concurrent, [78](#)
 - maximin* value of, [304](#)
 - maximising player *Max*, [299ff](#)
 - minimax* value of, [262](#), [294](#), [300](#), [304](#), [306](#)

- minimising player *Min*, 299ff
- path, **299**
- payoff, **300**
- round (\mathcal{RS}), **296**
- scheduler-luck, 310
- tree, *see* game tree
- value of fixed-strategy, **305**
- zero-sum, 295, 300
- gambling strategy, 246, 255ff
 - decided in advance, 275, 305
 - existence of memoriless, 308
 - formalised as functions $(\underline{\sigma}, \bar{\sigma})$, **305**
 - memoriless $(\phi_{\underline{\sigma}}, \phi_{\bar{\sigma}}^*)$, 256, 284, 294, 304, 307–8, 318
 - optimal, 256, 257ff, 295, 303
 - optimal not unique, 261, 276
 - winning, 295
 - with full memory, 305
 - see also* martingale
- game
 - card-and-dice, *see* Programs
 - gambling *qv*
 - Monty-Hall, *see* Programs
 - path, *see* gambling game
 - three-up, *see* Programs
- game semantics, 11–15, 254–62, 268
 - see also* semantics
- game tree, probabilistic ($\llbracket \phi \rrbracket$), 305
- P.H.B. Gardiner, 124, 242
- GCL*, *see* Guarded Command Language
- generating function, 291
- geometric distribution, 69, 195ff, 208, 336
- Geometric interpretation of
 - sublinearity, 176
- glue, *see* expectation
- Golden Ratio, 115
- gp*, *see* greatest pre-expectation
- greatest fixed-point, *see* fixed point
- greatest liberal pre-expectation
 - (*wlp*), 183, 239 *see also* liberal...semantics
- greatest lower bound, 25
- greatest pre-expectation (*wp*), **13**
 - as a modal operator, 252
 - greatest guaranteed probability of win metaphor, 12
 - not written *gp*, 26
 - picture of, 174
 - see also* defiance probability, expectation, expectation...semantics, pre-expectation
- guard
 - iteration *qv*
 - probabilistic, *see* iteration
- Guarded Command Language (*GCL*), 4ff, 158
- \mathbb{H} —
- \mathbb{H} , *see* probabilistic relational semantics
- half-space, *see* Euclidean space
- J.Y. Halpern, 164, 18, 78
- Halting Problem, solved, 261
- H. Hansson, 263
- D. Harel, ix, 36, 77
- S. Hart, 78, 216, 291
- J.J. den Hartog, *see* den Hartog
- Hausdorff space, 141
- I.J. Hayes, ix, x
- Jifeng He, 36, 108, 137, 139, 316, 322
- healthiness conditions, 18, 28–34, 129, 145–9, **148**, 217, 218
 - apply for infinite state spaces, 221, 338
 - apply to temporal logic, 249
 - defined, **29**
 - derive from sublinearity, 146
 - for regular transformers, 148
 - justify program algebra, 161
 - not satisfied by *wlp*, 184
 - pictures of, 175–7
 - strictness, *see* feasibility
 - weakened by angelic choice, 291
 - see also*
 - \ominus -subdistributivity,
 - additivity,
 - conjunctivity,
 - continuity,
 - feasibility,
 - monotonicity,
 - scaling,
 - semi-sublinearity,
 - sub-additivity,
 - sublinearity
- E.C.R. Hehner, 43, 64

- T. Henzinger, 78, 310
 O.M. Herescu, 93, 106
 M. Herlihy, 89
 Herman's Graph, variant for, [59](#)
 Herman's Ring, [57](#), 215
 exact running time, 58, **74**
 generalisations, 59ff
 variant for, [58](#)
 see also Programs
 W.H. Hesselink, 252
 heuristics, for invariants *qv*
 hierarchy, *see* standard transformer,
 transformer
 higher-order logic
 Skolem function, 154
 theorem prover (*HOL*), x, 162,
 216, 291
 see also second-order logic
 Thai Son [Hoang](#), x, 211
 C.A.R. Hoare, 5ff, 6, 36, 77, 108,
 135ff, 199
 -style reasoning, *see* Hoare triple
 Hoare logic, 36ff, 77ff *see also* Hoare
 triple
 Hoare triple, 4, 313–16
 probabilistic ($p \vdash \{Q\} \text{ prog } \{R\}$),
 313
 sequential composition, **314**
 specification, 33
HOL, *see* higher-order logic
 J. Hurd, x, 162, 207, 210, 216, 291
 M. Huth, x, 124, 164, 290
 hybrid, *see* conditional
 hyper-
 octant, 152, 332
 plane, 150, 172, 176, 333ff, 341ff
 pyramid (tetrahedron), 152,
 166, 239 *see also* probability
 distribution space
- II —
- I/O automata, 164
 idempotent, **31**, **322**
 idiom
 nonzero expectation, 99
 restricted implication, 39
if, *see* conditional
 IFIP Working Groups 2.1/2.3, x, 88
iid, *see* random variable
 image-finite, 71, **142**, 145
 see also continuity,
 standard relational semantics
 implication (\Rightarrow)
 Boolean- differs from \Rightarrow , 19, 182
 idiom, 39
 standard-embedded, **278**
 see also probabilistic implication
 inaccessible, *see* Smyth topology
 independent, random variable *qv*
 infeasible, *see* feasibility
 infimum
 of empty set is infinite, 138, 239
 over a set (\sqcap_P), **232**
 infinite state space, *see* state space
 infinitely many final states
 still continuous, 72
 see also Programs: geometric
 distribution
 infinity (∞)
 adjoined to the reals, 25, 70, 134,
 138, 144, 150
 see also infimum, miracle
 informal semantics
abort, **26**
 assignment, **26**
 conditional, **19**
 demonic choice, **26**
 probabilistic choice $p \oplus$, **26**
 probabilistic loop-guard, **196**
 sequential composition, **26**
skip, **26**
 standard temporal operators,
 248–51
 see also semantics
 integers (\mathbb{Z}), **7**
 interval, closed-open ($[a..b)$), **56**
 intrinsically unbounded, invariant *qv*
 invariant, 6, 39ff, 184–91, 199ff
 auxiliary, *see* reasoning within an
 invariant
 bounded in loop rules, 71
 coupling, *see* data refinement
 exact, **67**, 69, 187, 208, 212, 213
 for iteration, **39**
 heuristics for, 40, 48, 210–14
 intrinsically unbounded, 71, 330
 of iteration, **200**
 partial correctness, 185

- principal reasoning tool, 77
- probabilistic, **39**
- reasoning within *qv*
- standard, 39, 43, 53, 66, 71, 203
- strong, **191**
- technique is complete, 200
- typical form, 40
- unbounded in loop rules, 71
- weak, **182**, 190, 203
- Invariant-implies-termination loop
 - rule, **188**, **202**, **329**
- iteration (**do**...**od**)
 - algebraic properties, *see* program algebra
 - alternative rules, 56
 - counter-example to rule, 71
 - demonic rules, 328ff
 - extended rules, 93
 - guard, **39**, **196**
 - invariant of *qv*
 - justification of rules, 74
 - loop rules, **43**, **203**
 - multiple guards, **83**, 187
 - only weak invariance necessary, 186
 - partial correctness, 199, 200
 - probabilistic guard, 40, 69, 195–214
 - semantics (as fixed point), **21**, **198**, 330
 - semantics for probabilistic guards, **198**
 - semantics for *always*, 250
 - semantics for *eventually*, 249
 - syntax, **21**
 - total correctness, 42, 43, 203
 - see also* variant,
 - liberal... semantics
- J —
 - jail, as metaphor for **abort** *qv*
 - Zhendong Jin, x
 - C. Jones, 18, 36, 135, 164, 196, 216
 - C.B. Jones, 5
 - B. Jonsson, 164, 263
 - jump, *see* random walk: general
- K —
 - k*-fairness, *see* fairness
 - B.M. Kapron, 78
 - Kleene algebra, viii
- Kleisli composition (\cdot^*)
 - of programs, **135**
- D. Kozen, 6, 18, 36, 131ff, 135, 164, 239, 246, 252, 294ff, 310, 319
 - logic for sequential probabilistic programs, 4, 15
- R. Kurki-Suonio, 77
- M.Z. Kwiatkowska, 105, 164, 290, 304
- L —
 - lambda notation,
 - for functions (λ), **131**
 - K.G. Larsen, 124, 164
 - Las-Vegas algorithm, **43**, 45
 - see also* Programs
 - lattice, **25**
 - Law of the Excluded Miracle, 29, 175
 - is strictness, **145**
 - relaxed, 239
 - laws, program algebra *qv*
 - layered variant, *see* variant
 - leadership election,
 - see* Programs: self-stabilisation
 - least fixed-point, *see* fixed point
 - least upper bound, 25
 - D. Lehmann,
 - dining-philosophers algorithm,
 - see* Programs
 - lexicographic order, **85**, 87
 - see also* variant
 - liberal expectation-transformer
 - semantics, 184ff
 - abort**, **184**
 - iteration, 184
 - recursion, 184
 - see also*
 - greatest liberal... , semantics
 - limit point, 334
 - T. Lindner, 124
 - linear programming, 152, 180, 310, 341
 - linearity, 66
 - characterises determinism,
 - 218, **221–4**
 - implies standard disjunctivity, 28
 - super-, **28**
 - livelock, in the dining philosophers, **88**

- local cross-reference (\dagger , \ddagger), **65**
- logic
- classical *qv*
 - higher-order, second-order, first-order *qv*
 - quantitative temporal *qv*
 - standard *qv*
 - temporal *qv*
 - three-valued *qv*
 - working beyond, 34, 177, 288, 331, 338
- logical constant,
- captures initial value, 99
- loop, *see* iteration
- LTL*, **309**
- J.J. Lukkien, 252
- N. Lynch, 164
- M** —
- A.K. McIver, 88, 137, 216, 291
- Macquarie University, x
- R. Majumdar, 291, 310
- Z. Manna, 266
- Markov
- chain, 73
 - decision process, 164, 180, 263
 - process, 106, 115, **284**, 291
 - see also* demonic choice
- E. Martin, x
- martingale
- gambling strategy, 44ff
 - program, 45
 - program revisited, 65
 - see also* Programs
- Max*, *see* gambling game
- max**, *see* maximum
- maximin*, *see* gambling game
- maximum (**max**), **29**
- as unary operator, **29** *see also* bag
 - of expectation ($\sqcup\beta$), **147**
 - written \sqcup , **227**
- measurable function, over σ -algebra, 305
- measure
- integration over, 134
 - not needed even over infinite state space, 219
 - see also* evaluation
- memoriless strategies suffice, 307
- meta-theorems, 196, 199–206
- vs.* theorems, 199
- metric space, 141 *see also* Euclidean space, topology
- Miller-Rabin primality test, 47, 50
- imprecise probabilities, 275
 - see also* Programs
- Min*, *see* gambling game
- min**, *see* minimum
- min-straggler, in a token graph, **59**
- minimax*, *see* gambling game
- Minimax defined for Kozen
- interpretation, 307
- minimum (**min**), **9**
- miracle, 242ff
- Law of Excluded-, 145, 175
 - requires ∞ , 150
 - see also* feasibility, program
- Jay Misra, 77
- modal μ -calculus, 246
- quantitative *qv*
- modal algebra, standard, 262
- modal logic, 245
- modality, angelic \exists , demonic \forall , 298, 302
- model checking, 78, 105, 124, 180, 216, 268, 287, 304
- modular reasoning, 32–4, 267
- D. Monniaux, 124
- monotonicity, 6, **29**, 145, 147
- example of use, 314
 - has no picture, 175
 - of transformers, 147
 - see also* healthiness conditions
- Monte-Carlo algorithm, **44**, 47
- nontermination allowed, 44, 235
 - see also* Programs
- Monty Hall
- game, 6, 21ff, 27–8
 - program, 22 *see also* Programs
- C.C. Morgan, 5, 88, 124, 242
- J.M. Morris, 5, 252
- moth, mites in the ears of, 80
- R. Motwani, 105
- μ , *see* fixed point: least
- mu**, *see* recursion
- μ -calculus, modal *qv*
- multiplication, symbol sometimes omitted, 53

- multiset, *see* bag
- \mathbb{N} —
 - \mathbb{N} , *see* natural numbers
 - $N_{b,c}$, used in martingale example, **45**
 - N. Narashima, 310
 - natural numbers (\mathbb{N}), **61**
 - G. Nelson, 198
 - network
 - asynchronous, 60ff
 - synchronous, 56ff
 - next-time* (\circ, X), 248
 - as a game, 255ff
 - excludes miracles, 278
 - illustrated by coin flips, 269–71
 - informal standard semantics, 249
 - is monotonic, 278
 - is scaling, 278
 - subdistributes \ominus , 278
 - subdistributes ${}_p\oplus$, 278
 - quantitative, **252**
 - standard, **251**
 - nondeterminism
 - bounded, *see* continuity
 - demonic, *see* demonic choice
 - oblivious, 318
 - nontermination
 - picture of, **171**
 - state (\perp), 111, **130**
 - G. Norman, 304
 - ν , *see* fixed point: greatest
- \mathbb{O} —
 - oblivious, nondeterminism, 318
 - octant
 - hyper- *qv*
 - negative, 174
 - positive, 166, 172, 173
 - od**, *see* iteration
 - omega algebra, viii, 242
 - ω -limit, *see* fixed point
 - one-bounded, *see* expectation
 - order, partial *qv*
 - Oxford University, ix
 - Oxford-style of Z , 5
- \mathbb{P} —
 - \mathbb{P} , *see* powerset
 - C. Palamidessi, 93, 106
 - pAMN*, *see* The B Method
 - Prakash Panangaden, 164
 - parentheses, *see* bound variables
 - partial correctness, **42**, 82ff, 184–6, 199–202, 216
 - of gambler’s reasoning, 44
 - partial order, **25**
 - payoff, *see* gambling game
 - P_C , used in martingale example, **46**
 - pCTL*, *pCTL**, 77, 263, 268, 290ff
 - pGCL*, 4, 6, 129
 - introduction, 7–15
 - semantics, 24–7 *see also*
 - expectation. . . semantics
 - syntactic space (*Syn*), 157ff
 - syntax, 18–22
 - see also*
 - Guarded Command Language
 - $\phi_{\mathbb{G}}$, $\phi_{\overline{\mathbb{G}}}$, *see* gambling strategy
 - \prod , *see* product
 - G. Plotkin, 36
 - A. Pnueli, 78, 105, 216, 266
 - point distribution (\overline{s}), **132**
 - picture of, 167
 - set of, 144
 - see also* probability distribution
 - poker-, fruit-, slot machine, 254
 - polytope, **338**
 - positive, conjunctivity *qv*
 - post-expectation
 - picture of, 172
 - see also* expectation
 - postcondition, 4, 17, 251
 - powerdomain, probabilistic *qv*
 - powerset (\mathbb{P}), **138**, **143**
 - finite (\mathbb{F}), **332**
 - non-empty finite (\mathbb{F}^+), **142**, **143**
 - Pr , Pr_n , *see* probability distribution
 - pre-deterministic, **27**, **130**, 222
 - body of iteration, 187
 - see also* deterministic
 - pre-expectation
 - greatest *qv*
 - sufficient, **43**
 - see also* expectation
 - precondition, 4, 17, 251
 - necessary *vs.* sufficient, 43
 - weakest *qv*
 - predicate

as Boolean expression, 19
 as set of states, 13
 probabilistic, *see* expectation
 predicate transformer, 13, 251
 embedding (t^+), **232**
 retraction (t^-), **234**
 primality test, *see* Miller-Rabin,
 Programs
 priority, of a processor, 60
PRISM, 105, 304
 probabilistic
 amplification, 47 *see also* Programs
 fairness qv
 powerdomain, 36, 164
 probabilistic wp -semantics of $pGCL$,
 26
 probabilistic always-eventually law,
 279
 probabilistic and nondeterministic
 gambling game, 15
 probabilistic choice (${}_p\oplus$), **4**
 abstract in qB , 235
 as a range (${}_p\oplus q$), **8**, **20**
 at least p ($\geq_p\oplus$), **21**, 50, 275, 328
 between distributions, **138**
 between expectations, **198**
 can depend on the state, 19,
 196, 210–14
 in loop guard, 196ff
 interaction with demonic choice,
 see demonic choice
 is not free will, 275
 limited to (dyadic) rationals, 210
 multi-way (@), **20** *see also*
 probabilistic comprehension
 picture of, 167
 refines demonic, *see* demonic choice
 reversed (\oplus_p), **20**
 syntax, **19**
 used in ordinary arithmetic, **63**
 see also
 expectation... semantics,
 informal semantics,
 probabilistic relational...
 probabilistic closure (\mathbb{C}),
 see sets of distributions
 probabilistic comprehension ($\llbracket \cdot \rrbracket$), **20**
 probabilistic conjunction ($\&$), **30**

adjoint, *see* probabilistic
 implication
 associative over $[0, 1]$, 31
 commutative, 61
 depicted, 33
 example of use, 181–215
 intuition for, 31ff
 is sub-distributive, 31
 modular reasoning qv
 not idempotent, 183
 related to joint-event probability,
 31
 probabilistic conjunctivity, implies
 standard conjunctivity, 31
 probabilistic deterministic
 transformers, 135–7
 probabilistic double-eventually, 342
 probabilistic generalisation of
 standard temporal axioms, 281
 probabilistic implication
 &-adjoint (\dashv), 277, **279**
 between Boolean predicates,
 39, 267
 differs from \leq , 19
 differs from \Rightarrow , 19, 182
 entailment (\Rightarrow), **9**, **19**, 24
 equivalence (\equiv), **19**
 $exp \Rightarrow exp'$ badly typed, 19
 “in *all* states”, 33
 in semantics of iteration, 38
 is embedding of \models , 20
 is super-distributive, 279
 looks like “ \geq ”, 9
 reverse (\Leftarrow), **19**
 standard-embedded (\Rightarrow), 277
 see also probabilistic conjunction
 probabilistic relational semantics,
 137–41, 157
 demonic choice, **140**
 deterministic program, **131**
 model for ($\mathbb{H}S$), **139**, 149, 334ff
 probabilistic choice ${}_p\oplus$, **140**
 sequential composition, **140**
 space (PRS), 157ff
 see also semantics
 probabilistic temporal logic,
 see quantitative temporal logic
 probabilistic temporal operators,
 basic properties, 278

- probabilistic transformer semantics,
 - see* expectation transformer...
- probabilistically guarded iteration,
 - syntax, **196**
- probability
 - defiance, 89
 - theory, 16ff
- probability distribution (Pr), **16**
 - continuous, 16, 220, 238, 297
 - discrete, **16**, 166, 219, 220, 238
 - discrete over infinite state space, 219, 220, 332
 - explicit in formulae, 316ff
 - generalised to *evaluation*, 18, 36
 - geometric, 70, 208 *see also* Programs
 - infinitary, 332
 - not confused with expectation, 137
 - order, **131**
 - point (\bar{s}), 132
 - Pr_n , in proof of loop rules, **76**
 - space (\bar{S}), **130**
 - space as (hyper-) pyramid, 166ff, 239
 - space is compact, **334**
 - stationary, 291
 - sub-, **130**
 - “sub-” omitted, 131
 - uniform over $[0, 1]$, 220
 - written Δ , 130, 219
 - see also* sets of distributions
- product (\prod), **62**
- program
 - atomic, **160**
 - complexity *qv*
 - depicted as a diamond, **169**
 - depicted as a line, **169**
 - least, *see* **abort**
 - maximal, *see* deterministic program
 - miraculous, 138 *see also* feasibility
 - relational semantics for, *see*
 - probabilistic relational...
 - standard relational...
 - transformer semantics for, *see*
 - expectation transformer...
 - standard transformer...
- program algebra, 6, 28ff, 161–3, 170
 - collection of laws, 321–8
 - example, 10, 11, 112, 114ff, 120, 325–6
 - iteration, 38, 187
 - justified by healthiness conditions, 161
 - of *wp/wlp*, 331
 - of recursion, 38
- program superposition, 64
- Programs
 - card-and-dice game, 11ff
 - choice coordination (Rabin’s tourists), 79–87
 - coin-flip implementation of $p \oplus$, 210
 - control system, 325–6
 - counter-example to loop rules, 71
 - demonic, 8, 226
 - demonic not additive, 226
 - demonic/probabilistic, 21ff
 - deterministic, 7
 - dining philosophers, 88–98
 - Duelling Cowboys, 210–14
 - faulty factorial, 51–3
 - faulty skipper, 111, 115
 - flip a coin, 6
 - geometric distribution, 69, 196, 208
 - Herman’s (token) Ring, 56–61
 - Las-Vegas, 44–6
 - martingale gambling, 44–6, 64ff
 - Monte-Carlo, 46–51
 - Monty-Hall game, 22, 27ff
 - not continuous, 335
 - primality testing (Miller-Rabin), 46–51
 - probabilistic amplification, 46–51
 - probabilistic termination example, 40
 - random walk, bounded
 - two-dimensional, *see* Three-up
 - random walk, general, 99–105
 - random walk, symmetric, 71
 - refinement example, 10
 - self-stabilisation, 56–61
 - software publishing, 301–4
 - square root, 5
 - standard variant fails, 55, 56
 - steam boiler (safety-critical, fault-tolerant), 117–23
 - Three-up game, 72ff
 - unboundedly nondeterministic, 335

- unboundedly probabilistic, *see*
 - geometric distribution program
 - uncertain termination example, 62ff
 - proper, states, *see* state space
 - PRS*, *see* probabilistic relational semantics
 - PTS*, *see* expectation-transformer semantics
 - R. Pucella, 18, 164
 - pyramid, hyper- *qv*
- Q —
- qB*, *see* B Method
 - qM μ* , *see* quantitative modal μ -calculus
 - qTL*, *see* quantitative temporal logic
 - quantitative modal μ -calculus (*qM μ*), 293–309
 - complementary interpretations, 304–8
 - game interpretation ($\int \dots \llbracket \phi \rrbracket$), 305
 - logical interpretation ($\llbracket \phi \rrbracket, \llbracket \phi \rrbracket$), **298, 305**
 - quantitative temporal logic (*qTL*)
 - algebra of, 265–91
 - called *qTL*, 262, 265
 - existential modalities, 291
 - introduction, 245–62
 - probabilistic- as a special case, 246
 - semantics ($\llbracket \cdot \rrbracket_{\nu}$), 252–4
 - special case of *qM μ* , 294, 309
 - quasi-associativity, 322
 - quasi-commutativity, 322
 - quasi-distributivity, 322
- R —
- RS*, *see* gambling game
 - $\mathbb{R}, \mathbb{R}_{\geq}$, *see* real numbers
 - Rabin and Lehmann’s dining-philosophers algorithm, 92
 - M.O. Rabin, 36
 - choice coordination, 83 *see also* Programs
 - dining-philosophers, *see* Programs
 - see also* Miller-Rabin primality test
 - P. Raghavan, 105
 - random behaviour, 4
 - random bit, v
 - random stumbler, *see* random walk
 - random variable, **16**, 133–5
 - called *expectation*, 137
 - definition, **134**
 - independent and identically distributed (*iid*), **100**
 - written α, β , **130, 219**
 - see also* expectation
 - random walk, 52, 291
 - as paradigm for termination, 215
 - bounded two-dimensional, 72
 - general, 99–105, 283–8
 - general program, 101
 - generalised symmetric, 105
 - (non-)homogeneous, 100, 284ff
 - requiring fully numeric reasoning, 267
 - symmetric, 71, 99, 105
 - symmetric- as counter-example, 71
 - tabulated, 75
 - unbounded, 74
 - with stumbling, 287–8
 - see also* Programs
 - J.R. Rao, 77, 105, 291
 - rational, dyadic, *see* probabilistic choice
 - ready
 - groups of philosophers, **94**
 - philosophers must act, 94
 - real numbers (\mathbb{R}), 134
 - computable, 209
 - non-negative (\mathbb{R}_{\geq}), **13**
 - reasoning within an invariant
 - auxiliary, **66**, 67ff
 - of a loop, 66, 211ff
 - see also* modular reasoning
 - recurrent, process, **99**
 - recursion
 - algebraic properties, *see* program algebra
 - consistent semantics, *see* complementary... semantics
 - defines iteration, 21, 198
 - simple example of termination, 41
 - syntax (**mu**), **19, 21**
 - tail, 38, 198

- see also*
 expectation... semantics,
 liberal... semantics
 referential transparency, 277
 refinement (\sqsubseteq), 5
 (\sqcap) \sqsubseteq ($_{p\oplus}$), 10, 26, 45, 80, **139**, 220
 approximate, vii
 approximate not necessary, 170
 between deterministic relational
 programs, **131**
 complementary to abstraction, 5, 9
 data *qv*
 in semantics of recursion, 38, 41
 is set inclusion, 170
 is “up-right” movement
 geometrically, 172
 of functional properties only, 107
 picture of, 170, 174
 probabilistic, **9**, **24**
 simple example, *see* Programs
 strict (\sqsubset), **133**
 Refinement Calculus, 5, 36
 reflexive, **25**
 regular transformer
 is sublinear, 146
see also expectation transformer
 Regulator
 implementation, 121
 refinement, 120
 specification, 119
 Reisz Representation Theorem, 17
 relational semantics
 probabilistic *qv*
 standard *qv*
 relations, continuous and feasible
 standard (\mathbb{S}), **232**
 reliability, quantified, 118
 retraction, 80
 for standard reasoning, 80
 standard, **234**
 transformer-to-relational (*rp*),
149, 251
see also continuity, *wp*
 \mathbb{R}^N , *see* Euclidean space
 K.A. Robinson, x
 W.-P. de Roever, *see* de Roever
 B. Van Roy, *see* Van Roy
 Royal Holloway College, x
rp, *see* retraction, *wp*
- \mathbb{S} —
 \mathbb{S} , *see* relations
 S, S_{\perp}, S_a, S_b , *see* state space
 \bar{S} , *see* sub-distribution space
 \bar{s} , *see* point distribution
 safety property, 60
 safety-critical system, example, *see*
 Programs
 N. Saheb-Djahromi, 36
 I. Saias, 164
 sample space, **16**
 J.W. Sanders, 108, ix
 scaling, **29**, 147
 allows manipulation of bounds, 183
 example of use, 34, 53, 67, 94, 190,
 191, 193, 314, 340
 of transformers, 147
 picture of, 175
see also healthiness conditions
 scheduler
 adversarial, 61, 83, **88**, 93ff
 almost fair, 93
 built-in to semantics, 291
 “real-world”, 61
 S. Schneider, x, 325
 J.L.A. van de Snepscheut, *see* van de
 Snepscheut
 second-order, *vs.* first-order logic, 38
 R. Segala, 164
 K. Seidel, ix, 137
 self-stabilisation, *see* Programs
 semantic models, all four, 143
 semantic structures, 158
 semantics
 as a card game, 14
 complementary and consistent *qv*
 computational model, 11ff
 do-it-yourself, 195–214
 expectation-transformer *qv*
 game *qv*
 informal *qv*
 liberal expectation-transformer *qv*
 liberal predicate-transformer, 182
 probabilistic relational *qv*
 probabilistic transformer, *see*
 expectation... semantics
 standard relational *qv*
 standard-transformer *qv*

- see also*
 - quantitative temporal logic,
 - standard temporal logic
- semi-linearity, **234**
 - characterises standard
 - transformers, 218, 237
 - see also* expectation transformer
- semi-sublinearity, **228**
 - characterises \sqcup -closure, 218, 230
 - see also* expectation transformer
- Separating-Hyperplane Lemma,
 - 337, 341
 - infinite case, 342
 - use of, 150
- sequential composition
 - consistent semantics, *see*
 - complementary... semantics
 - syntax, **19**
 - see also*
 - expectation... semantics,
 - Hoare triple,
 - informal semantics,
 - probabilistic relational...
- sets of distributions
 - called a *clump*, **168**
 - Cauchy-closed, **139**, 154, 334
 - convex, **138**, 164, 180
 - non-empty, 138, 239
 - probabilistically closed (C), **139**, **143**, 166ff
 - rationale for closure, 164
 - up-closed, **138**
- C. Shankland, 216
- L.S. Shapley, 310
- M. Sharir, 78, 216, 291
- shift by k ($\cdot \not\prec k$), **286**
- \sum , *see* summation
- $\sigma, \bar{\sigma}$, *see* gambling strategy
- σ -algebra, 16, 36, 41, **297**
 - Borel algebra, 36, **297**, 298
 - defined by tree, 297, 305
 - example, 297, 298
 - measurable function, 305
- simulation
 - for data refinement, **110**
 - see also* data refinement
- skip**
 - informal semantics, 26
 - syntax, **19**
- see also* expectation... semantics
- skipper, *see* Programs
- Skolemisation, **154**
- A. Skou, 124, 164
- slot machine, *see* poker machine
- smooth, *see* clump
- Smyth order, 138, 142
- Smyth topology, **298**
 - inaccessible, **298**
- soundness *vs.* completeness, 162ff, 202
- specification
 - debugging of, 287
 - Hoare-triple qv
 - of random stumbler, 288
 - of random walker, 285
- square-root program, *see* Programs
- SRS*, *see* standard relational...
 - standard
 - expectation qv
 - logic, **5**
 - means “non-probabilistic”, **7**, 247
 - transformer, *see* standard transformer
- standard demonic programs depicted, 168
- standard double-eventually, 266
- standard reasoning, via retraction qv
- standard relational semantics, 142ff
 - deterministic program, **130**
 - image-finite program, 142
 - space (*SRS*), 157ff
 - total program, 142
 - see also* semantics
- standard temporal logic
 - complete axioms for, 216, 266
 - semantics ($\|\cdot\|_{\nu}$), 251–2
- standard transformer
 - characterisation, 231–8 *see also*
 - semi-linearity
 - hierarchy, 218
 - semantics (*STS*), 142ff, 157ff *see also* semantics, standard temporal logic
- state, probabilistic, 135
- state space (S), 130
 - (un-)countable, 16, 209, 220, 297
 - final subset of (S_b), **136**

- finiteness exploited, 130, 133, 139, 142, 143, 147, 149ff, 166ff, 194ff, 205ff, 294ff
 - infinite, 218, 219–21, 332–41
 - initial subset of (S_a) , **136**
 - original, 72
 - proper elements, **132**
 - superposed, 72
 - with nontermination (S_\perp) , **130, 143**
 - statistics, confidence measure, 12, 221
 - steam boiler, 108, 117–23
 - changed water level, 118
 - see also* Programs
 - C. Stirling, 246, 310, 262
 - game, 294ff
 - M.I.A. Stoelinga, 124
 - straggler, in a token graph, **59**
 - strategic software development
 - example, 302
 - strategy, gambling *qv*
 - strict function, *see* function
 - strictness, 145
 - generalised by feasibility, 29, 221
 - see also* feasibility
 - strong invariant, *see* invariant
 - strongly connected graph, **59**
 - structural induction, 178ff
 - Structure of transformer spaces, 240
 - STS*, *see* standard transformer...
 - stumbler, *see* random walk
 - sub-additivity, **148, 221** *see also*
 - healthiness conditions
 - sub-conjunctivity
 - example of use, 34
 - see also* probabilistic conjunction
 - sub-distribution, **130**
 - order (\sqsubseteq) , **131**
 - space (\bar{S}) , **130**
 - see also* probability distribution
 - sub-probability measure, 164
 - discrete *qv*
 - see also* probability distribution
 - \ominus -subdistributivity, **148, 239**
 - picture of, 177
 - see also* healthiness conditions
 - Sublinear transformers are regular, 153
 - sublinearity, 18, 129, **145, 162, 267**
 - characterises \sqcap closure, 226
 - characterises regular transformers, 153, 218
 - equivalent to separate conditions, 148
 - example of use, 102, 314
 - generalises conjunctivity, 28, 31, 221
 - implies all other healthiness conditions, 146
 - not in *wlp*-semantics, 184
 - of *next-time*, 262
 - picture of, 176
 - variations on, 218
 - see also* expectation transformer, healthiness conditions
 - substitution
 - implicit $(\mathcal{C}(OP))$, **109**
 - written $\cdot \langle \cdot \mapsto \cdot \rangle$, **6**
 - subtraction, truncated (\ominus) , **28, 146**
 - sufficient, pre-expectation *qv*
 - summation (\sum) , **16**
 - super-additivity, 221
 - super-distributivity,
 - see* probabilistic implication
 - super-linearity, *see* linearity
 - superposition, 64 *see also* state space
 - supplier-oriented, data refinement *qv*
 - support
 - of an expectation, 332
 - of an open set, **332**
 - symmetric, random walk *qv*
 - symmetry breaking,
 - in distributed algorithms, v, 79
 - see also* Programs: distributed consensus, dining philosophers
 - Syn*, *see* *pGCL*
 - synchronous network of processors, 56ff
 - syntactic sugar, 19–21, 293, 328
 - can be ignored, 184
- T** —
- $\mathbb{T}, \mathbb{T}_r, \mathbb{T}_o, \mathbb{T}_\sqcap, \mathbb{T}_s$, *see* expectation transformer space
 - $T_{b,c}$, used in martingale example, **46**
 - t^+, t^- , *see* predicate transformer embedding

- Tabulation of 2-skipper program, 116
tail recursion, *see* recursion
take n from b , *see* bag
temporal logic, 245
 alternation-free formulae, 276–7
 branching-time, 291
 fixed set of variables (*Var*), **248**
 linear-time, 247, 309
 probabilistic, *see* quantitative
 temporal logic
 quantitative branching-time,
 252–62
 right-association of operators, 269
 semantic $\| \cdot \|$ omitted, 269
 semantics ($\| \cdot \|_v$), 248ff
 standard branching-time,
 247–52, 266
 universal \forall *vs.* existential \exists
 operators, 247, 249, 291
see also
 modal μ -calculus,
 quantitative temporal logic,
 standard temporal logic
- termination, 40–4, 53–63, 79–105
 absolute, **41**, 44
 almost-certain, **41**, 44, 54, 191–5,
 203–6, 246
 example of probabilistic-, *see*
 Programs
 picture of, 171
 proved by mathematical induction,
 61ff
 proved by variant, *see* variant
 uncertain, 61ff *see also* Programs
 Zero-One Law for qv
- testing equivalence, 164
tetrahedron, *see* hyper-pyramid
then, *see* conditional
theorems *vs.* meta-theorems, *see*
 meta-theorems
theory, of a logic, **199**
thin coin, *see* coin flipping
Three simple programs depicted, 167
Three-up game, 73 *see also* Programs
three-valued logic, 84 *see also*
 undefined values
- threshold probability, 263ff, 290
token ring, *see* Programs
top element, of a partial order, 183
topology, 36, 141, 221, 297–8, 332ff
 basis of, 332
 neighbourhood, 342
 network, 106
 product, 334, 342
 Smyth, 298
 see also Euclidean space
- total correctness, **42**, 184, 186–9,
 202–3, 216
 for probabilistic loops, 42, 203
 iteration, 42, 43, 203
- touching-plane geometry, 173
tourists, *see* Programs
transformer
 expectation qv
 hierarchy, 218, 221–38
 predicate qv
- transformer-to-relational, retraction
 qv
- Transition system justifying choice of
 operators, 282
- transitive, **25**
TUCS, ix
Tychonoff's Theorem, **334**
- type, of program variables, 21, 22, 28,
 33, 72, 213
- U —
- unbounded
 expectation qv
 invariant qv
- unbounded nondeterminism,
 see continuity
- uncertain, termination qv
- uncountable, state space qv
- undefined values, **84**, 100 *see also*
 three-valued logic
- unending computations, 21, 268
UNITY, 77, 105, 291
- The University of New South Wales
 (UNSW), x
- unless* (\triangleright), 248
 also called *weak until*, 250
 as a game, 261ff
 informal standard semantics, 250–1
 quantitative, **253**
 standard, **252**
- unreliability, quantified, 118
until, weak, *see unless*

- up closure, **138**
 - picture of, 171
 - standard, **142**
 - see also* sets of distributions
- US quarters, 269
- ∇ —
- \mathcal{V} , *see* temporal logic semantics, valuation
- F.W. Vaandrager, 124
- valuation (\mathcal{V}), **297**
 - incorporates environment, 298
- F. van Breugel, 125
- J.L.A. van de Snepscheut, 252
- B. Van Roy, 310
- Var*, *see* temporal logic
- M. Vardi, 78, 105, 164
- variant, 6, 54–61, 191–5, 203–6
 - alternative rule for iteration, **56**
 - complete for finite state spaces, 55, 89, 194, 205–6, 214
 - complete for standard termination, 55
 - defined, **55**
 - eventual* decrease of, 93
 - extended rule for iteration, **93**
 - incomplete for probabilistic termination, 85 *see also* Programs
 - increasing, 56
 - layered (lexicographic), 60, **85**, 89, 205
 - rule for loops, 55, 191
 - rule for probabilistic-guard loops, 204
 - standard fails, 85
- VDM development method, 5
- E.P. de Vink, *see* de Vink
- J. von Wright, 38, 195, 218, 228, 238, 242, 246, 262
- W —
- Wang Yi, 164
- M. Ward, 38
- weak invariant, *see* invariant
- weakest liberal precondition (*wlp*), 182ff, 250, 331
 - distinguished from *wp*, 182
 - for loop, **184**
 - not expressible with *wp* alone, 182
- weakest nonterminating precondition (*wnp*), 250
 - is not feasible, 250
- weakest pre-expectation, *see* greatest pre-expectation
- weakest precondition (*wp*), 7
 - guaranteed-win metaphor, 12
 - “syntactic” *vs.* “semantic” version, 136
 - written as reason in calculation ($\cdot \equiv$), **48**
 - see also*
 - weakest liberal precondition
- Martin Wirsing, 125
- wlp*, *see*
 - greatest liberal . . .
 - weakest liberal . . .
- wnp*, *see* weakest nonterminating precondition
- wp*
 - as embedding function, **144**, **219**, 334
 - demonic-enabled, 137
 - inverse *rp*, **148**, 334
 - is continuous, 179
 - semantics, *see*
 - expectation-transformer . . .
 - standard-transformer . . .
 - see also* greatest pre-expectation
- J. von Wright, *see* von Wright
- ℕ —
- X, *see* *next-time*
- x-ray vision, 14
- Υ —
- M. Yannakakis, 105
- Wang Yi, *see* Wang Yi
- Mingsheng Ying, 36, 125

\mathbb{Z} — \mathbb{Z} , *see* integers Z specification language, 5

Zero-One Law, 78

example of use, 56, 76, 93, 101,
104, 204, 214, 258for probabilistic processes, 54for termination, **54** qTL version, 289 \mathcal{ZPP} , *see* complexity class

L. Zuck, 105, 216