Appendix A
Alternative approaches

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

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

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

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

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

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

(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

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

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

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

[1] If 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.

[2] Use 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

$$1/4 \vdash \{\mathsf{true}\}\ prog_1; prog\ \{n = 5\}\ .$$ [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.

[3] For $(prog_0; prog)$ 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 $(1/2 + 0)/2 = 1/4$.

[4] A 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

$$\Pr.(c = \mathsf{heads}) \;\geq\; 1/2 \qquad \text{the probability that } c \text{ is heads is at least } 1/2$$

$$\text{and} \quad \mathrm{Exp}.(n) \;\leq\; 3 \qquad \text{the expected value of } n \text{ is no more than 3.}$$

[5]

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

$$\{\mathsf{true}\} \quad c{:=}\ \mathsf{heads}\ {}_{1/2}\!\oplus \mathsf{tails} \quad \{\,\mathrm{Exp}.[c = \mathsf{heads}] \;\geq\; 1/2\,\}\ ,$$

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

$$\{1/2\} \quad c{:=}\ \mathsf{heads}\ {}_{1/2}\!\oplus \mathsf{tails} \quad \{c = \mathsf{heads}\}\ , \qquad [6]$$

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

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

[5] In 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 $\mathrm{Exp}.[c = \mathsf{heads}] \;\geq\; 1/2$.

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

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

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

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

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

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

hold, where $\overline{\mathsf{heads}} = \mathsf{tails}$ *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

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

to hold as well — yet it does not. That demonic choice $\mathbf{skip} \sqcap c{:=}\ \overline{c}$ is refined for example by the deterministic $c{:=}\ \mathsf{tails}$ 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

$$\mathbf{skip} \ \sqcap \ c{:=}\ \overline{c} \quad \sqsubseteq \quad c{:=}\ \mathsf{tails}$$

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

$$(\cdots \sqcap \cdots) \quad \sqsubseteq \quad (\cdots \ \mathbf{if}\ G\ \mathbf{else}\ \cdots) \ , \tag{A.2}$$

which holds for any Boolean G and for the test "$c = \mathsf{tails}$" in particular.[8]

[7] This 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.

[8] See 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.

[9]Making 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]

[10] An 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

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

and $$d{:=}\ \mathsf{heads} \sqcap \mathsf{tails}\,;\ c{:=}\ \mathsf{heads} \sqcap \mathsf{tails}$$

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

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*

$s{:=}\ (0 \textbf{ if } G \textbf{ else } 1)$ 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*:

$$\begin{array}{rcl} prog \,{}_{p}\!\oplus\, prog & = & prog \\ prog_1 \,{}_{p}\!\oplus\, prog_2 & = & prog_2 \,{}_{\overline{p}}\!\oplus\, prog_1 \end{array}$$

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

$$prog_1 \,{}_{p}\!\oplus\, (prog_2 \,{}_{q/\overline{p}}\!\oplus\, prog_3) \quad = \quad (prog_1 \,{}_{p/\overline{q}}\!\oplus\, prog_2) \,{}_{q}\!\oplus\, prog_3$$

---

[1] We haven't included laws here for angelic programs.

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

[3] A 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.

[4] Recall 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

$$(prog_1 \ {}_p\!\oplus\ prog_2) \ {}_q\!\oplus\ (prog_3 \ {}_r\!\oplus\ prog_4) \quad = \quad \left| \begin{array}{ll} prog_1 & @\ pq \\ prog_2 & @\ \overline{p}q \\ prog_3 & @\ \overline{q}r \\ prog_4 & @\ \overline{q}\,\overline{r} \end{array} \right.$$

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

$$(prog_1 \ {}_p\!\oplus\ prog_2) \ {}_q\!\oplus\ (prog_1 \ {}_p\!\oplus\ prog_3) \quad = \quad prog_1 \ {}_p\!\oplus\ (prog_2 \ {}_q\!\oplus\ prog_3)$$
$$\text{and} \quad (prog_1 \ {}_p\!\oplus\ prog_2) \ {}_{\geq q}\!\oplus\ (prog_1 \ {}_p\!\oplus\ prog_3) \quad = \quad prog_1 \ {}_p\!\oplus\ (prog_2 \ {}_{\geq q}\!\oplus\ prog_3)$$

- Law relating demonic choice and refinement:

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

- Laws depending on the arithmetic of ${}_p\!\oplus$ 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

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

8. $(prog_1 \sqcap prog_2) \ {}_p\!\oplus\ prog_3 \quad = \quad (prog_1 \ {}_p\!\oplus\ prog_3) \ \sqcap\ (prog_2 \ {}_p\!\oplus\ prog_3)$
9. $(prog_1 \sqcap prog_2) \ {}_{\geq p}\!\oplus\ prog_3 \quad = \quad (prog_1 \ {}_{\geq p}\!\oplus\ prog_3) \ \sqcap\ (prog_2 \ {}_{\geq p}\!\oplus\ prog_3)$
10. $(prog_1 \sqcap prog_2) \ {}_{\leq p}\!\oplus\ prog_3 \quad = \quad (prog_1 \ {}_{\leq p}\!\oplus\ prog_3) \ \sqcap\ (prog_2 \ {}_{\leq p}\!\oplus\ prog_3)$
11. $(prog_1 \ {}_p\!\oplus\ prog_2) \ \sqcap\ prog_3 \quad \sqsupseteq \quad (prog_1 \sqcap prog_3) \ {}_p\!\oplus\ (prog_2 \sqcap prog_3)$

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

12. $(prog_1 \ \textbf{if}\ G\ \textbf{else}\ prog_2); prog_3 \quad = \quad prog_1; prog_3 \ \textbf{if}\ G\ \textbf{else}\ prog_2; prog_3$
13. $(prog_1 \ {}_p\!\oplus\ prog_2); prog_3 \quad = \quad prog_1; prog_3 \ {}_p\!\oplus\ prog_2; prog_3$
14. $(prog_1 \sqcap prog_2); prog_3 \quad = \quad prog_1; prog_3 \ \sqcap\ prog_2; prog_3$
15. (From Laws 13, 14 and the definition of ${}_{\geq p}\!\oplus$.)

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

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

16. (Compare Law 18 below.)

$$prog_1; (prog_2 \ {}_p\!\oplus\ prog_3) \quad \sqsupseteq \quad prog_1; prog_2 \ \ {}_p\!\oplus\ \ prog_1; prog_3$$

17. (Compare Law 19 below.)

$$prog_1; (prog_2 \sqcap prog_3) \quad \sqsubseteq \quad prog_1; prog_2 \ \sqcap\ prog_1; prog_3$$

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

18. (Compare Law 16.) $det; (prog_1 \ {}_p\!\oplus\ prog_2) \quad = \quad det; prog_1 \ {}_p\!\oplus\ det; prog_2$

19. (Compare Law 17.) $std; (prog_1 \sqcap prog_2) \quad = \quad std; prog_1 \sqcap std; prog_2$

20. (From Laws 18, 19 and the definition of ${}_{\geq p}\!\oplus$.)

$$stdet; (prog_1 \ {}_{\geq p}\!\oplus\ prog_2) \quad = \quad stdet; prog_1 \ \ {}_{\geq p}\!\oplus\ \ stdet; prog_2$$

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

$$(det_1 \ {}_p\!\oplus\ det_2);\ (prog_3 \ {}_q\!\oplus\ prog_4) \quad = \quad \left| \begin{array}{ll} det_1;\ prog_3 & @\ pq \\ det_1;\ prog_4 & @\ p\overline{q} \\ det_2;\ prog_3 & @\ \overline{p}q \\ det_2;\ prog_4 & @\ \overline{p}\,\overline{q} \end{array} \right.$$

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

$$(std_1 \ {}_p\!\oplus\ std_2);\ (prog_3 \sqcap prog_4)$$

$$\begin{array}{rl} = & \quad std_1; prog_3 \ \ {}_p\!\oplus\ \ std_2; prog_3 \\ \sqcap & \quad std_1; prog_3 \ \ {}_p\!\oplus\ \ std_2; prog_4 \\ \sqcap & \quad std_1; prog_4 \ \ {}_p\!\oplus\ \ std_2; prog_3 \\ \sqcap & \quad std_1; prog_4 \ \ {}_p\!\oplus\ \ std_2; prog_4 \end{array}$$

- Quasi-associative/distributive laws with inequalities:

23. (From Laws 7 and 8 and the definition of ${}_{\geq p}\!\oplus$; compare Law 5.)

$$\begin{array}{ll} & (prog_1 \ {}_p\!\oplus\ prog_2) \ \ {}_{\geq q}\!\oplus\ \ (prog_1 \ {}_{\geq p}\!\oplus\ prog_3) \\ \sqsupseteq & \quad prog_1 \ \ {}_{\geq p}\!\oplus\ \ (prog_2 \ {}_{\geq q}\!\oplus\ prog_3) \\ \sqsupseteq & \quad (prog_1 \ {}_{\geq p}\!\oplus\ prog_2) \ \ {}_{\geq q}\!\oplus\ \ (prog_1 \ \ {}_p\!\oplus\ prog_3) \\ = & \quad (prog_1 \ {}_{\geq p}\!\oplus\ prog_2) \ \ {}_{\geq q}\!\oplus\ \ (prog_1 \ {}_{\geq p}\!\oplus\ prog_3) \end{array}$$

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

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

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

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

[5] The 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].

$$
\begin{array}{lll}
= & a{:=}\ \mathsf{true} \ \sqcap\ a{:=}\ (\mathsf{true}\ {}_{.9/.99}\!\oplus \mathsf{true})\ {}_{.99}\!\oplus \mathsf{false} & \text{Law 3} \\
= & a{:=}\ \mathsf{true} \ \sqcap\ a{:=}\ \mathsf{true}\ {}_{.99}\!\oplus \mathsf{false} & \\
= & a{:=}\ \mathsf{true}\ {}_{\geq .99}\!\oplus\ \mathsf{false}\ , & \text{definition } {}_{\geq p}\!\oplus
\end{array}
$$

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 ${}_{\geq p}\!\oplus$):

25. $prog_1 \oplus_{\leq pq} prog_2 \quad = \quad prog_1 \oplus_{\leq p} (prog_1 \oplus_{\leq q} prog_2)$

26. $prog_1 \oplus_{\geq pq} prog_2 \quad = \quad prog_1 \oplus_{\geq p} (prog_1 \oplus_{\geq q} prog_2)$

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. $(prog_1 \sqcap prog_2) \oplus_{\leq pq} prog_3 \quad \sqsubseteq \quad prog_1 \oplus_{\leq p} (prog_2 \oplus_{\leq q} prog_3)$

28. $(prog_1 \sqcap prog_2) \oplus_{\geq pq} prog_3 \quad \sqsubseteq \quad prog_1 \oplus_{\geq p} (prog_2 \oplus_{\geq q} prog_3)$

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

The statement $c{:=}\ \mathsf{true}\,{}_{\geq .9}\!\oplus\mathsf{false}$ above contains demonic choice resolved *after* the probabilistic choice carried out in $b{:=}\ \mathsf{true}\,{}_{\geq .9}\!\oplus\mathsf{false}$ 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 $c{:=}\ \mathsf{true}$; and the best c-strategy for making a false is $c{:=}\ \mathsf{true}\ {}_{.9}\!\oplus \mathsf{false}$. 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

$$
\begin{array}{l}
b{:=}\ \mathsf{true}\ {}_{\geq 1/2}\!\oplus \mathsf{false}; \\
c{:=}\ \mathsf{true}\ {}_{\geq 1/2}\!\oplus \mathsf{false}; \\
a{:=}\ (b \Leftrightarrow c)
\end{array}
\qquad \text{(B.1)}
$$

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

[6] At 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]

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

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:

$$\begin{array}{lll} & b{:=}\ \mathsf{true}\ {}_{\geq 1/2}\!\oplus \mathsf{false}; & \\ & c{:=}\ \mathsf{true}\ {}_{\geq 1/2}\!\oplus \mathsf{false}; & \\ & a{:=}\ (b \Leftrightarrow c) & \\ \\ = & b{:=}\ \mathsf{true}\ {}_{\geq 1/2}\!\oplus \mathsf{false}; & \text{Law 15; } c \text{ local} \\ & a{:=}\ b\ {}_{\geq 1/2}\!\oplus \overline{b} & \\ \\ = & & \text{Laws 15, 20; } b \text{ local} \\ & a{:=}\ (\mathsf{true}\ {}_{\geq 1/2}\!\oplus \mathsf{false})\ {}_{\geq 1/2}\!\oplus\ (\mathsf{false}\ {}_{\geq 1/2}\!\oplus \mathsf{true}) & \\ = \ldots & & \end{array}$$

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

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

$\ldots =$ definition $_{\geq p}\oplus$; Law 8

$$\begin{array}{lll} & a{:=}\ \mathsf{true} & \leftarrow \textit{ true most likely} \\ \sqcap & a{:=}\ \mathsf{true}\ {}_{1/2}\oplus\ \mathsf{false} & \\ \sqcap & a{:=}\ \mathsf{true}\ {}_{1/2}\oplus\ \mathsf{false} & \\ \sqcap & a{:=}\ \mathsf{true}\ {}_{1/2}\oplus\ (\mathsf{false}\ {}_{1/2}\oplus\ \mathsf{true}) & \\ \sqcap & a{:=}\ (\mathsf{true}\ {}_{1/2}\oplus\ \mathsf{false})\ {}_{1/2}\oplus\ \mathsf{false} & \leftarrow \textit{ true least likely} \\ \sqcap & a{:=}\ (\mathsf{true}\ {}_{1/2}\oplus\ \mathsf{false})\ {}_{1/2}\oplus\ (\mathsf{false}\ {}_{1/2}\oplus\ \mathsf{true}) & \end{array}$$

$=$ Law 7, *i.e.* $(\sqcap) \sqsubseteq ({}_p\oplus)$

$$\begin{array}{lll} & a{:=}\ \mathsf{true} & \leftarrow \textit{ true most likely} \\ \vdots & \text{(others subsumed)} & \vdots \\ \sqcap & a{:=}\ (\mathsf{true}\ {}_{1/2}\oplus\ \mathsf{false})\ {}_{1/2}\oplus\ \mathsf{false} & \leftarrow \textit{ true least likely} \end{array}$$

$= \quad a{:=}\ \mathsf{true}\ {}_{\geq 1/4}\oplus\ \mathsf{false}\ .$ Law 3, definition $_{\geq p}\oplus$

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

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

in which the first statement generalises the syntactic sugar for $_{\geq p}\oplus$, *i.e.* is an abbreviation for

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

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

$$dloop \quad := \quad \textbf{do}\ G \to det\ \textbf{od}\ ,$$

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

$$wp.dloop.postE \quad \equiv \quad wp.loop.postE\ . \quad {}^{9} \tag{B.4}$$

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

$$wp.body.preE \quad \equiv \quad wp.det.preE\ . \tag{B.5}$$

Then we have

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

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 $wp.dloop.postE \Rrightarrow preE$ 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 $I \Rrightarrow T$ then

$$I \quad \Rrightarrow \quad wp.loop.(\lceil \overline{G} \rceil * I)\ .$$

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

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

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. □

[9] A similar trick is required for the proof of (2.24) for intrinsically unbounded invariants (p. 72). If all deterministic $det \sqsupseteq body$ 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 $\lceil \overline{G} \rceil * I \Rrightarrow postE$ for some invariant I and, using the least fixed-point definition (1.19) of iteration and the linearity of $wp.det$, we have

$$wp.dloop.(\lceil \overline{G} \rceil * I) \quad \equiv \quad (\sqcup n{:}\,\mathbb{N} \bullet\ I_n) \ ,$$

where we define

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

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

$$E_n \quad := \quad (\lambda X \bullet\ [G] * wp.det.X)^n.I \ ,$$

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

$$wp.dloop.(\lceil \overline{G} \rceil * I) \quad \equiv \quad (\ \lim_{n\to\infty} I_n + E_n\ ) \ ,$$

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

$$\begin{array}{lll} & I_3 + E_3 & \\ & & \\ \equiv & \left.\begin{array}{ll} & \lceil \overline{G} \rceil * I \\ + & [G] * wp.det.(\lceil \overline{G} \rceil * I) \\ + & [G] * wp.det.([G] * wp.det.(\lceil \overline{G} \rceil * I)) \end{array}\right\} I_3 & \\ & & \\ & \left.\begin{array}{ll} + & [G] * wp.det.([G] * wp.det.([G] * wp.det.I)) \end{array}\right\} E_3 & \\ & & \\ \Lleftarrow & \begin{array}{ll} & \lceil \overline{G} \rceil * I \\ + & [G] * wp.det.(\lceil \overline{G} \rceil * I) \\ + & [G] * wp.det.([G] * wp.det.(\lceil \overline{G} \rceil * I)) \\ + & [G] * wp.det.([G] * wp.det.([G] * I)) \end{array} & \text{invariance of } I \\ & & \\ \equiv & \begin{array}{ll} & \lceil \overline{G} \rceil * I \\ + & [G] * wp.det.(\lceil \overline{G} \rceil * I) \\ + & [G] * wp.det.([G] * wp.det.I) \end{array} & \text{linearity of } det \\ & & \\ \Lleftarrow & I \ , & \text{above two steps twice more} \end{array}$$

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

$$wp.dloop.(\lceil \overline{G} \rceil * I) \quad \equiv \quad (\sqcup n{:}\,\mathbb{N} \bullet\ I_n) \quad \equiv \quad (\ \lim_{n\to\infty} I_n + E_n\ ) \quad \Lleftarrow \quad I \ ,$$

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

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

□

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

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

□

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

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

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

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

□

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 $S \to \mathbb{R}_{\geq}$, 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. □

[10] A 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 $\mathbb{R}^S_{\geq}$.

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

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

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

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 $\mathbb{R}^P_{\geq}$ is the complement of an open set there. □

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 distributions[11]

$$\{\Delta{:}\,\mathbb{R}^S_{\geq} \mid \int_\Delta \alpha \leq r\}$$

is closed in $\mathbb{R}^S_{\geq}$.

Proof We have

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

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 $\mathbb{F}S$. □

[11] We 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

$$\{\Delta\colon \mathbb{R}^S_{\geq} \mid \int_\Delta \underline{1} \geq 1\} \ . \tag{B.7}$$

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 $\mathbb{R}^S_{\geq}$ 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 $(\mathbb{R}^S_{\geq}, \mathcal{E}_S)$.

Proof The set $\overline{S}$ may be written

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

But $[0,1]^S$ is compact by Tychonoff's Theorem,[12] and $\{\Delta\colon \mathbb{R}^S_{\geq} \mid \int_\Delta \underline{1} \leq 1\}$ 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.) □

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

$$\begin{array}{rrll} & rp\colon & \mathbb{T}S \to \mathbb{H}S & \text{(Def. 5.7.1)} \\ \text{and} & wp\colon & \mathbb{H}S \to \mathbb{T}S & \text{(Def. 5.5.2)} \quad . \end{array}$$

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:

[12] Tychonoff'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

$$rp.t.s \quad = \quad \{\Delta\colon \overline{S} \bullet (\forall \alpha\colon \mathbb{E}_f S \bullet t.\alpha.s \leq \int_\Delta \alpha)\} \ .$$

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 $\overline{S}$ we have

$$\begin{array}{ll} & (\forall P\colon \mathbb{F}S \bullet t.(\alpha{\downarrow}P).s \leq \textstyle\int_\Delta \alpha{\downarrow}P) \\ \text{implies} & (\sqcup P\colon \mathbb{F}S \bullet t.(\alpha{\downarrow}P).s) \leq (\sqcup P\colon \mathbb{F}S \bullet \ \textstyle\int_\Delta \alpha{\downarrow}P) \\ \\ \text{iff} & \qquad\qquad\qquad\qquad t \text{ boundedly continuous; bounded monotone convergence} \\ & t.(\sqcup P\colon \mathbb{F}S \bullet \alpha{\downarrow}P).s \leq \textstyle\int_\Delta (\sqcup P\colon \mathbb{F}S \bullet \alpha{\downarrow}P) \\ \\ \text{iff} & t.\alpha.s \leq \textstyle\int_\Delta \alpha \ . \end{array}$$

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

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 $\overline{S}$.

Proof We reason

$$\begin{array}{lll} & rp.t.s & \\ = & \{\Delta\colon \overline{S} \bullet (\forall \alpha\colon \mathbb{E}_f S \bullet t.\alpha.s \leq \textstyle\int_\Delta \alpha)\} & \text{Lem. B.4.5} \\ = & (\cap \alpha\colon \mathbb{E}_f S \bullet \{F\colon \overline{S} \mid t.\alpha.s \leq \textstyle\int_\Delta \alpha\}) \ , & \end{array}$$

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

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

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

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

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

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

[13] To see that this t is not continuous, consider $\underline{1}$, the limit of the family of (finitary) expectations $\{\underline{1}{\downarrow}P \mid P\colon \mathbb{F}S\}$, 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 $\{\Delta{:}\,\overline{S} \mid 1 \leq \int_\Delta \underline{1}\}$ is the convex closure of the set of all standard final states $\{s{:}\,S \bullet \overline{s}\}$ — 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 $\Rrightarrow$-directed and bounded subset of expectations in $\mathbb{E}_B S$; we show that for any $c > 0$ and state s in S

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

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

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

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

† For the deferred justification note that Lem. B.4.3 and Cauchy closure of $r.s$ imply Cauchy closure of each set $\{\Delta{:}\,r.s \bullet\ \int_\Delta \alpha \leq x + c\}$, 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. □

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

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

do not require finiteness of S. However our original proofs of the stronger results, namely the equality $rp \circ wp = \mathsf{id}$ 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

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

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. □

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

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

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

$$\begin{array}{rl} & (\cap \alpha' : \mathbb{E}_f S \bullet \{\Delta : \overline{S} \mid t.\alpha'.s \leq \textstyle\int_\Delta \alpha'\}) \\ \cap & \{\Delta : \overline{S} \mid -t.\alpha.s \leq (\textstyle\int_\Delta -\alpha)\} \\ = & \emptyset \ . \end{array} \tag{B.9}$$

Again we argue by the *Finite-Intersection Lemma* that some finite subcollection 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 $\mathbb{R}^T_{\geq}$, a finite dimensional space, and we are effectively considering an empty intersection of M sets based on

$$\begin{array}{rl} & (\cap \alpha' : \mathbb{E}_f T \bullet \{\Delta : \mathbb{R}^T_{\geq} \mid t.\alpha'.s \leq \int_\Delta \alpha'\}) \\ \cap & \{\Delta : \mathbb{R}^T_{\geq} \mid -t.\alpha.s \leq (\int_\Delta -\beta)\} \\ \cap & \{\Delta : \mathbb{R}^T_{\geq} \mid -1 \leq \int_\Delta (\underline{-1})\} \\ = & \emptyset\,. \end{array}$$

The subsequent contradiction is achieved exactly as in Lem. 5.7.4. □

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

[14] A 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 $\{\Delta{:}\,\overline{S} \mid t.\alpha.s \leq \int_\Delta \alpha\}$ — 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 $\overline{S}$, *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

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

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.

[15] It 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 $\alpha' \ominus \overline{x/\sqcup\alpha} \Rrightarrow \alpha/\sqcup\alpha \Rrightarrow \alpha'$ — 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:

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

† 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:

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

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

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

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]. □

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

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

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. □

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

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

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

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

Proof Replace $\subseteq$ by $\Rrightarrow$ 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

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

for all expectations A, then we have also

$$\diamond(t.A) \quad \Rrightarrow \quad t.(\diamond A)$$

for all A.

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

Proof We reason

$$\begin{array}{lll} & \Diamond(t.A) \quad \Rrightarrow \quad t.(\Diamond A) & \\ \text{if} & t.A \sqcup \circ t.(\Diamond A) \quad \Rrightarrow \quad t.(\Diamond A) & \Diamond(t.A) \textit{ least}, \text{ Fig. 10.3.1} \\ \text{if} & t.A \sqcup t.(\circ\Diamond A) \quad \Rrightarrow \quad t.(\Diamond A) & \text{assumption} \\ \text{if} & t.(A \sqcup \circ\Diamond A) \quad \Rrightarrow \quad t.(\Diamond A) & t \text{ monotonic} \\ \text{iff} & t.(\Diamond A) \quad \Rrightarrow \quad t.(\Diamond A)\ . & \text{Def. 9.3.2} \end{array}$$

□

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

$$A \;\Rrightarrow\; B \quad \text{implies} \quad \Diamond A \;\Rrightarrow\; \Diamond B\ .$$

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

$$\begin{array}{lll} & A \sqcup \circ\Diamond B & \\ \Rrightarrow & B \sqcup \circ\Diamond B & A \Rrightarrow B \\ \equiv & \Diamond B\ , & \end{array}$$

as required. □

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

$$\Diamond A \quad \Rrightarrow \quad \underline{\sqcup A}\ .$$

Proof Use $\Diamond A$ *least*, and check that

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

□

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

$$\Diamond(pA) \quad \equiv \quad p(\Diamond A)\ .$$

Proof We prove $\Diamond(pA) \Rrightarrow p(\Diamond A)$ first, using $\Diamond(pA)$ *least* and checking

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

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

$$(\Diamond(pA))/p \quad \Rrightarrow \quad \underline{\sqcup(pA)}/p \quad \Rrightarrow \quad \underline{p}/p \quad \equiv \quad \underline{1}\ .$$

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

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

□

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 *Dolcani 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 Vytopil, 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 qMu. 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 interprètation 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

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.
