Annabelle McIver
Carroll Morgan

Abstraction, Refinement and Proof for Probabilistic Systems

With 62 Figures

Springer
Preface

Probabilistic techniques in computer programs and systems are becoming more and more widely used, for increased efficiency (as in random algorithms), for symmetry breaking (distributed systems) or as an unavoidable artefact of applications (modelling fault-tolerance). Because interest in them has been growing so strongly, stimulated by their many potential uses, there has been a corresponding increase in the study of their correctness — for the more widespread they become, the more we will depend on understanding their behaviour, and their limits, exactly.

In this volume we address that last concern, of understanding; we present a method for rigorous reasoning about probabilistic programs and systems. It provides an operational model — “how they work” — and an associated program logic — “how we should reason about them” — that are designed to fit together. The technique is simple in principle, and we hope that with it we will be able to increase dramatically the effectiveness of our analysis and use of probabilistic techniques in practice.

Our contribution is a probabilistic calculus that operates at the level of the program text, and it is light-weight in the sense that the amount of reasoning is similar in size and style to what standard assertional techniques require. In the fragment at right, for example, each potential loop entry occurs with probability $1/2$; the resulting iteration establishes $x \geq 1/2$ with probability exactly $p$ for any $0 \leq p \leq 1$. It is thus an implementation of the general operation choose with probability $p$, but it uses only simple tests of unbiased random bits (to implement the loop guard). It should take only a little quantitative logic to confirm that claim, and indeed we will show that just four lines of reasoning suffice.

Economy and precision of reasoning are what we have come to expect for standard programs; there is no reason we should accept less when they are probabilistic.

---

The cover illustration comes from page 59.
The program fragment is adapted from Fig. 7.7.10 on page 210.
Preface

Scope and applicability

Methods for the analysis of probabilistic systems include automata, labelled transition systems, model checking and logic (e.g. dynamic or temporal). Our work falls into the last category: we overlay the Hoare-logic paradigm with probabilistic features imported from Markov processes, taking from each the essential characteristics required for a sound mathematical theory of refinement and proof. The aim is to accommodate modelling and analysis of both sequential and distributed probabilistic systems, and to allow — even encourage — movement between different levels of abstraction.

Our decision to focus on logic — and a proof system for it — was motivated by our experience with logical techniques more generally: they impose a discipline and order which promotes clarity in specifications and design; the resulting proofs can often be carried out, and checked, with astonishing conciseness and accuracy; and the calculation rules of the logic lead to an algebra that captures useful equalities and inequalities at the level of the programs themselves.

Although we rely ultimately on an operational model, we use it principally to validate the logic (and that, in turn, justifies the algebra) — direct reliance on the model’s details for individual programs is avoided if possible. (However we do not hesitate to use such details to support our intuition.) We feel that operational reasoning is more suited to the algorithmic methods of verification used by model checkers and simulation tools which can, for specific programs, answer questions that are impractical for the general approach that a logic provides.

Thus the impact of our approach is most compelling when applied to programs which are intricate either in their implementation or their design, or have generic features such as undetermined size or other parameters. They might appear as probabilistic source-level portions of large sequential programs, or as abstractions from the probabilistic modules of a comprehensive system-level design; we provide specific examples of both situations. In the latter case the ability to abstract modules’ properties has a significant effect on the overall verification enterprise.

Technical features

Because we generalise the well-established assertional techniques of specifications, pre- and postconditions, there is a natural continuity of reasoning style evident in the simultaneous use of the new and the familiar approaches: the probabilistic analysis can be deployed more, or less, as the situation warrants.

A major feature is that we place probabilistic choice and abstraction together, in the same framework, without having to factor either of them out for separate treatment unless we wish to (as in fact we do in Chap. 11). This justifies the abstraction and refinement of our title, and is what gives
us access to the stepwise-development paradigm of standard programming where systems are “refined” from high levels of abstraction towards the low levels that include implementation detail.

As a side-effect of including abstraction, we retain its operational counterpart demonic choice as an explicit operator \( \sqcap \) in the cut-down probabilistic programming language \( pGCL \) which we use to describe our algorithms — that is, the new probabilistic choice operator \( p \sqcup \) refines demonic choice rather than replacing it. In Chap. 8 we consider angelic choice \( \sqcup \) as well, which is thus a further refinement.

Probabilistic and demonic choice together allow an elementary treatment of the hybrid that selects “with probability at least \( p \)” (or similarly “at most \( p \)”), an abstraction which accurately models our unavoidable ignorance of exact probabilities in real applications. Thus in our mathematical model we are able to side-step the issue of “approximate refinement.”

That is, rather than saying “this coin refines a fair coin with probability 95%,” we would say “this coin refines one which is within 5% of being fair.” This continues the simple view that either an implementation refines a specification or it does not, which simplicity is possible because we have retained the original treatment in terms of sets of behaviours: abstraction is inclusion; refinement is reverse inclusion; and demonic choice is union. In that way we maintain the important relationship between the three concepts. (Section 6.5 on pp. 169ff illustrates this geometrically.)

Organisation and intended readership

The material is divided into three major parts of increasing specialisation, each of which can to a large extent be studied on its own; a fourth part contains appendices. We include a comprehensive index and extensive cross-referencing.

Definitions of notation and explanations of standard mathematical techniques are carefully given, rather than simply assumed; they appear as footnotes at their first point of use and are made visually conspicuous by using SMALL CAPITALS for the defined terms (where grammar allows). Thus in many cases a glance should be sufficient to determine whether any footnote contains a definition. In any case all definitions, whether or not in footnotes, may be retrieved by name through the index; and those with numbers are listed in order at page xvii.

Because much of the background material is separated from the main text, the need for more advanced readers to break out of the narrative should be reduced. We suggest that on first reading it is better to consult the footnotes only when there is a term that appears to require definition — otherwise the many cross-references they contain may prove distracting, as they are designed for “non-linear” browsing once the main ideas have already been assimilated.
Preface

Part I, *Probabilistic guarded commands*, gives enough introduction to the probabilistic logic to prove properties of small programs such as the one earlier, for example at the level of an undergraduate course for Formal-Methods-inclined students that explains “what to do” but not necessarily “why it is correct to do that.” These would be people who need to understand how to reason about programs (and why), but would see the techniques as intellectual tools rather than as objects of study in their own right.

We have included many small examples to serve as models for the approach (they are indexed under *Programs*), and there are several larger case studies (for example in Chap. 3).

Part II, *Semantic structures*, develops in detail the mathematics on which the probabilistic logic is built and with which it is justified. That is, whereas the earlier sections present and illustrate the new reasoning techniques, this part shows where they have come from, why they have the form they do and — crucially — why they are correct.

That last point is especially important for students intending to do research in logic and semantics, as it provides a detailed and extended worked example of the fundamental issue of proving reasoning techniques themselves to be correct (more accurately, “valid”), a higher-order concept than the more familiar theme of the previous part in which we presented the techniques *ex cathedra* and used them to verify particular programs.

This part would thus be suitable for an advanced final-year undergraduate or first-year graduate course, and would fit in well with other material on programming semantics. It defines and illustrates the use of many of the standard tools of the subject: lattices, approximation orders, fixed points, semantic injections and retractions *etc.*

Part III, *Advanced topics*, concentrates on more exotic methods of specification and design, in this case probabilistic temporal/modal logics. Its final chapter, for example, contains material only recently discovered and leads directly into an up-to-date research area. It would be suitable for graduate students as an introduction to this specialised research community.

Part IV includes appendices collecting material that either leads away from the main exposition — e.g. alternative approaches and why we have not taken them — or supports the text at a deeper level, such as some of the more detailed proofs.

It also contains a short list of algebraic laws that demonic/probabilistic program fragments satisfy, generated mainly by our needs in the examples and proofs of earlier sections. An interesting research topic would be a more systematic elaboration of that list with a view to incorporating it into probabilistic Kleene- or omega algebras for distributed computations.
Overall, readers seeking an introduction to probabilistic formal methods could follow the material in order from the beginning. Those with more experience might instead sample the first chapter from each part, which would give an indication of the scope and flavour of the approach generally.

**Original sources**

Much of the material is based on published research, done with our colleagues, in conference proceedings and journal articles; but here it has been substantially updated and rationalised — and we have done our best to bring the almost ten years’ worth of developing notation into a uniform state.

For self-contained presentations of the separate topics, and extra background, readers could consult our earlier publications as shown overleaf.

At the end of each chapter we survey the way in which our ideas have been influenced by — and in some cases adopted from — the work of other researchers, and we indicate some up-to-date developments.

**Acknowledgements**

Our work on probabilistic models and logic was carried out initially at the University of Oxford, together with Jeff Sanders and Karen Seidel and with the support of the UK’s *Engineering and Physical Sciences Research Council* (the EPSRC) during two projects led by Sanders and Morgan over the years 1994–2001.

Morgan spent sabbatical semesters in 1995–6 at the University of Utrecht, as the guest of S. Doaitse Swierstra, and at the University of Queensland and the Software Verification and Research Centre (SVRC), as the guest of David Carrington and Ian Hayes. The foundational work the EPSRC projects produced during that period — sometimes across great distances — benefited from the financial support of those institutions but especially from the academic environment provided by the hosts and by the other researchers who were receptive to our initial ideas [MMS96].

Ralph Back at Åbo Akademi hosted our group’s visit to Turku for a week in 1996 during which we were able to explore our common interests in refinement and abstraction as it applied to the new domain; that led later to a three-month visit by Elena Troubitsyna from the *Turku Center for Computer Science* (TUCS), to our group in Oxford in 1997, and contributed to what has become Chap. 4 [MMT98].

David Harel was our host for a two-week visit to Israel in 1996, during which we presented our ideas and benefited from the interaction with researchers there.
Chapters’ dependence on original sources

Chapter 1 see [MM99b, SMM, MMS00]
Chapter 2 see [Mor96, MMS00]
Chapter 3 see [MM99b]
Chapter 4 see [MMT98]
Chapter 5 see [MMS96]
Chapter 6 is new material
Chapter 7 see [Mor96, MM01b]
Chapter 8 see [MM01a]
Chapter 9 see [MM97]
Chapter 10 see [MM99a]
Chapter 11 see [MM02]

The sources listed opposite are in chronological order of writing, thus
giving roughly the logical evolution of the ideas.

Subsequently we have continued to work with Sanders and with Ken
Robinson, Thai Son Hoang and Zhendong Jin, supported by the Australian
Research Council (ARC) over the (coming) years 2001–8 in their Large
Grant and Discovery programmes, at the Universities of Macquarie and of
New South Wales.

Joe Harp from the Computer Laboratory at Cambridge University vis-
ited us in 2002, with financial assistance from Macquarie University; and
Orieta Celiku was supported by TUCS when she visited in 2003. Both
worked under McIver’s direction on the formalisation of pGCL, and its
logic, in the mechanised logic HOL.

Hoang, Jin and especially Eric Martin have helped us considerably with
their detailed comments on the typescript; also Ralph Back, Ian Hayes,
Michael Huth, Quentin Miller and Wayne Wheeler have given us good
advice. Section B.1 on the algebraic laws satisfied by probabilistic programs
has been stimulated by the work (and the critical eyes) of Steve Schneider
and his colleagues at Royal Holloway College in the U.K.

We thank the members of IFIP Working Groups 2.1 and 2.3 for their
many comments and suggestions.

LRI Paris  Annabelle McIver
May 2004  Carroll Morgan

In memoriam AJMcG
List of sources in order of writing


Contents

Preface \hspace{1cm} v

List of definitions etc. \hspace{1cm} xvii

Part I Probabilistic guarded commands \hspace{1cm} 1

1 Introduction to pGCL \hspace{1cm} 3
1.1 Sequential program logic \hspace{1cm} 4
1.2 The programming language pGCL \hspace{1cm} 7
1.3 An informal computational model for pGCL \hspace{1cm} 11
1.4 Behind the scenes: elementary probability theory \hspace{1cm} 16
1.5 Basic syntax and semantics of pGCL \hspace{1cm} 18
1.6 Healthiness and algebra for pGCL \hspace{1cm} 28
1.7 Healthiness example: modular reasoning \hspace{1cm} 32
1.8 Interaction of probabilistic- and demonic choice \hspace{1cm} 34
1.9 Summary \hspace{1cm} 35

Chapter notes \hspace{1cm} 36

2 Probabilistic loops: invariants and variants \hspace{1cm} 37
2.1 Introduction: loops via recursion \hspace{1cm} 38
2.2 Probabilistic invariants \hspace{1cm} 39
2.3 Probabilistic termination \hspace{1cm} 40
2.4 Invariance and termination together: the loop rule \hspace{1cm} 42
2.5 Three examples of probabilistic loops \hspace{1cm} 44
2.6 The Zero-One Law for termination \hspace{1cm} 53
2.7 Probabilistic variant arguments for termination \hspace{1cm} 54
2.8 Termination example: self-stabilisation .......................... 56
2.9 Uncertain termination .................................................. 61
2.10 Proper post-expectations ............................................. 63
2.11 Bounded v.s. unbounded expectations .......................... 68
2.12 Informal proof of the loop rule ................................. 74

Chapter notes ................................................................. 77

3 Case studies in termination ............................................
  3.1 Rabin’s choice coordination ........................................ 79
  3.2 The dining philosophers ............................................. 88
  3.3 The general random “jump” ........................................ 99

Chapter notes ................................................................. 105

4 Probabilistic data refinement: the steam boiler ..................
  4.1 Introduction: refinement of datatypes ......................... 107
  4.2 Data refinement and simulations ............................... 108
  4.3 Probabilistic datatypes: a worked example .................. 110
  4.4 A safety-critical application: the steam boiler ............ 117
  4.5 Summary ............................................................... 123

Chapter notes ................................................................. 124

Part II Semantic structures ............................................

5 Theory for the demonic model ....................................... 129
  5.1 Deterministic probabilistic programs ......................... 130
  5.2 The sample space, random variables and expectations ...... 133
  5.3 Probabilistic deterministic transformers ..................... 135
  5.4 Relational demonic semantics ..................................... 137
  5.5 Regular transformers ................................................ 141
  5.6 Healthiness conditions for probabilistic programs .......... 145
  5.7 Characterising regular programs ................................ 149
  5.8 Complementary and consistent semantics ..................... 154
  5.9 Review: semantic structures ...................................... 157

Chapter notes ................................................................. 164

6 The geometry of probabilistic programs ......................... 165
  6.1 Embedding distributions in Euclidean space ................. 166
  6.2 Standard deterministic programs ............................... 166
  6.3 Probabilistic deterministic programs ......................... 167
  6.4 Demonic programs ................................................ 168
  6.5 Refinement ........................................................... 169
Contents

6.6 Nontermination and sub-distributions ......................... 171
6.7 Expectations and touching planes .............................. 172
6.8 Refinement seen geometrically ................................ 174
6.9 The geometry of the healthiness conditions .................... 175
6.10 Sublinearity corresponds to convexity ......................... 176
6.11 Truncated subtraction ......................................... 177
6.12 A geometrical proof for recursion .............................. 177

Chapter notes ....................................................... 180

7 Proved rules for probabilistic loops ............................... 181

7.1 Introduction .................................................... 182
7.2 Partial loop correctness ........................................ 184
7.3 Total loop correctness .......................................... 186
7.4 Full proof of the loop rule ...................................... 189
7.5 Probabilistic variant arguments ................................. 191
7.6 Finitary completeness of variants ............................... 193
7.7 Do-it-yourself semantics ...................................... 195
7.8 Summary ......................................................... 214

Chapter notes ....................................................... 216

8 The transformer hierarchy .......................................... 217

8.1 Introduction ..................................................... 217
8.2 Infinite state spaces ............................................. 219
8.3 Deterministic programs ........................................ 221
8.4 Demonic programs .............................................. 224
8.5 Angelic programs ................................................ 227
8.6 Standard programs .............................................. 231
8.7 Summary .......................................................... 238

Chapter notes ....................................................... 242

Part III Advanced topics ........................................... 243

9 Quantitative temporal logic: an introduction .................... 245

9.1 Modal and temporal logics ...................................... 245
9.2 Standard temporal logic: a review ............................. 247
9.3 Quantitative temporal logic ...................................... 252
9.4 Temporal operators as games .................................. 254
9.5 Summary .......................................................... 262

Chapter notes ....................................................... 263
10 The quantitative algebra of $qTL$ 265
  10.1 The role of algebra ................................. 265
  10.2 Quantitative temporal expectations .................. 268
  10.3 Quantitative temporal algebra ........................ 277
  10.4 Examples: demonic random walkers and stumblers .... 283
  10.5 Summary ............................................ 289
Chapter notes ................................................ 291

11 The quantitative modal $\mu$-calculus $qM\mu$, and games 293
  11.1 Introduction to the $\mu$-calculus ....................... 293
  11.2 Quantitative $\mu$-calculus for probability ............ 295
  11.3 Logical formulae and transition systems ............... 295
  11.4 Two interpretations of $qM\mu$ ........................ 298
  11.5 Example ............................................. 301
  11.6 Proof of equivalence of interpretations ............... 304
  11.7 Summary ............................................. 308
Chapter notes ................................................ 310

Part IV Appendices, bibliography and indexes 311

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

B Supplementary material 321
  B.1 Some algebraic laws of probabilistic programs ......... 321
  B.2 Loop rule for demonic iterations ........................ 328
  B.3 Further facts about probabilistic wp and wdp .......... 331
  B.4 Infinite state spaces ................................... 332
  B.5 Linear-programming lemmas ............................. 341
  B.6 Further lemmas for eventually .......................... 342

Bibliography 345

Index of citations 357

General index 361


## Appendix A

Alternative approaches

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1</td>
<td>Probabilistic Hoare-triples</td>
<td>313</td>
</tr>
<tr>
<td>A.2</td>
<td>A programming logic of distributions</td>
<td>316</td>
</tr>
</tbody>
</table>

### 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

\[ p \vdash \{ \text{pre} \} \text{prog} \{ \text{post} \}, \]

(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

$$
p \vdash \{ \text{pre} \} \ \text{prog}_{0} \ \{ \text{mid} \} \quad q \vdash \{ \text{mid} \} \ \text{prog}_{1} \ \{ \text{post} \}
$$

$$
p \cdot q \vdash \{ \text{pre} \} \ \text{prog}_{0} ; \ \text{prog}_{1} \ \{ \text{post} \}
$$

for any programs $\text{prog}_{0}, \ \text{prog}_{1}$ and standard predicates $\text{pre, mid, post}$.\footnote{If $q$ in particular were not constant, we would have to take account of its being evaluated over the final state of $\text{prog}_{0}$ (i.e. the initial state of $\text{prog}_{1}$) rather than the initial state of $\text{prog}_{0}$ as is the case for $p$. The resulting composite probability would then be

$$
p \cdot \left[ \prod_{v \in \text{vars}} (v \cdot q) \right],
$$

where $v$ is the vector of variables that $\text{prog}_{0}$ can assign to, since — taking the demonic view — we would have to assume that any choice inherent in postcondition $\text{mid}$ for $\text{prog}_{0}$ would be exploited to make $q$ as low as possible.}

It relies on the probabilistic choices in $\text{prog}_{0}$ and $\text{prog}_{1}$ being independent, and on the monotonicity of multiplication. Indeed by defining

$$
p \vdash \{ \text{pre} \} \ \text{prog} \ \{ \text{post} \} \quad := \quad p \cdot \text{pre} \Rightarrow \text{wp} \ \text{prog} \ \{ \text{post} \}
$$

such statements become special cases within our current system, and the above sequential composition rule is easily proved from sublinearity.\footnote{Use its consequences scaling and monotonicity.} 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

$$
\text{prog}_{0} \quad := \quad n = 4 \ \land \ (n = 5 \ \land \ n = 6)
$$

$$
\text{prog}_{1} \quad := \quad (n = 4 \ \land \ n = 5) \ \lor \ (n = 4 \ \land \ n = 6).
$$

(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 $\text{prog}_{0}$ and $\text{prog}_{1}$ would be identified. Are they therefore the same?

No they are not: define a further program

$$
\text{prog} \quad := \quad (n = 5 \ \land \ n = 6) \ \textbf{if} \ n = 4 \ \textbf{else} \ \textbf{skip},
$$

and consider the sequential compositions $\text{prog}_{0} ; \ \text{prog}$ and $\text{prog}_{1} ; \ \text{prog}$ with respect to the postcondition $n = 5$: we have

$$
1/2 \ \vdash \ \{ \text{true} \} \ \text{prog}_{0} ; \ \text{prog} \ \{ n = 5 \}
$$

but

$$
1/2 \ \not\vdash \ \{ \text{true} \} \ \text{prog}_{1} ; \ \text{prog} \ \{ n = 5 \},
$$

Programs $\text{prog}_0$ and $\text{prog}_1$ cannot be distinguished with standard postconditions.

Figure A.1.1. COUNTER-EXAMPLE TO COMPOSITIONALITY

<table>
<thead>
<tr>
<th>possible postcondition</th>
<th>$\text{prog}_0$ probability</th>
<th>$\text{prog}_1$ probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$n = 4$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$n = 5$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$n = 6$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$n \neq 4$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$n \neq 5$</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>$n \neq 6$</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>true</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Programs $\text{prog}_0$, $\text{prog}_1$ are distinguished by either of the two post-expectations $\text{post}E$.

Figure A.1.2. COMPOSITIONALITY REQUIRES FULL USE OF EXPECTATIONS

and in fact the strongest judgement we can make about $\text{prog}_1$ is

$$1/4 \vdash \{\text{true}\} \text{prog}_1; \text{prog}\{n = 5\} \, . \tag{3}$$

That is why standard postconditions are not expressive enough — if the programs $(\text{prog}_0; \text{prog})$ and $(\text{prog}_1; \text{prog})$ are different, then $\text{prog}_0$ and $\text{prog}_1$ cannot be the same.\(^4\) This lack of compositionality is why we do not use probabilistic Hoare-triples.

Fig. A.1.2 shows that $\text{prog}_0$ and $\text{prog}_1$ are indeed distinguished by the properly probabilistic post-expectations that we introduced in Chap. 1.

\(^3\)For $(\text{prog}_0; \text{prog})$ note that it doesn’t matter how the initial nondeterministic choice is resolved, since the result is $1/2$ either way. For $(\text{prog}_1; \text{prog})$ however the probability of establishing $n = 5$ is $1/2 \times 1/2 = 1/4$ for the left branch of the initial choice, but $1/2 \times 1/2 = 1/4$ for the right branch; thus overall it is only $(1/2 + 0)/2 = 1/4$.

\(^4\)A further (but only informal) argument that $\text{prog}_0$ and $\text{prog}_1$ should be distinguished is the observation that $\text{prog}_0$ should terminate in states $5, 6$ “with equal frequency,” however low or high that might be — but $\text{prog}_1$ does not have that property.
Appendix A. Alternative approaches

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:

\[
\Pr(c = \text{heads}) \geq 1/2 \quad \text{the probability that } c \text{ is heads is at least } 1/2
\]

and

\[
\exp(n) \leq 3 \quad \text{the expected value of } n \text{ is no more than } 3.
\]

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

\[
\{ \text{true} \} \quad c := \text{heads} \quad \{ \exp(c = \text{heads}) \geq 1/2 \},
\]

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

\[
\{1/2\} \quad c := \text{heads} \quad \{ c = \text{heads} \},
\]

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

\[
\frac{1}{2} \vdash \{ \text{true} \} \quad c := \text{heads} \quad \{ c = \text{heads} \}.
\]

\footnote{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 \(\exp(c = \text{heads}) \geq 1/2\).}

\footnote{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 \( \text{Fair} \) stands for the predicate over distributions 

\[
\Pr(c = \text{heads}) = \Pr(c = \text{tails}),
\]

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

\[
\{ \text{Fair} \} \quad \text{skip} \quad \{ \text{Fair} \}
\]

and

\[
\{ \text{Fair} \} \quad c := \overline{c} \quad \{ \text{Fair} \},
\]

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

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

\[
\{ \text{Fair} \} \quad \text{skip} \sqcap c := \overline{c} \quad \{ \text{Fair} \}
\]

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

\[
\text{skip} \sqcap c := \overline{c} \sqsubseteq c := \text{tails}
\]

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

\[
(\cdots \sqcap \cdots) \sqsubseteq (\cdots \text{if } G \text{ else } \cdots),
\]

(A.2)

which holds for any Boolean \( G \) and for the test \( “c = \text{tails}” \) 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 $\cap$ 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.$^{10}$
A.2. A programming logic of distributions 319

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

\[ c = \text{heads} \cap \text{tails}; \quad d = \text{heads} \cap \text{tails} \]

and

\[ d = \text{heads} \cap \text{tails}; \quad c = \text{heads} \cap \text{tails} \]

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

When unwanted distinctions like that occur, it is necessary to use more elaborate techniques to prove algebraic equalities. Kozen rejected a similar sequence-of-choices model for (deterministic) probabilistic programs on just those grounds (among other reasons) [Koz81]; and it is for similar reasons (again, among others) that we use the model we have chosen, an extension of Kozen’s [HS97].

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).
Bibliography


[AL96] Christoph Andriessen 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.


[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].


[MM04a] A.K. McIver and C.C. Morgan. An elementary proof that Herman’s Ring has complexity Θ(N^2). Available at [MMSS, key HR04], 2004.
[MM04b] A.K. McIver and C.C. Morgan. Results on the quantitative 

termination in B. In D. Bert, J.P. Bowen, S. King, and M. Waldén, 
editors, ZB 2003: Formal Specification and Development in Z and 

transformers. ACM Transactions on Programming Languages and 
doi:acm.org/10.1145/229542.229547.

Hoare probably! In J.W. Davies, A.W. Roscoe, and J.C.P. Woodcock, 

Systems Group: Collected reports. 
web.comlab.ox.ac.uk/oucl/research/areas/prob.

Available at [MMS8], key STEAM86).

[Mon01] David Monniaux. Analyse de programmes probabilistes par in-
interprétation abstraite. Thèse de doctorat, Université Paris IX 

[Mor87] J.M. Morris. A theoretical basis for stepwise refinement and the 


[Mor88b] C.C. Morgan. The specification statement. ACM Transactions on 
Reprinted in [MV94].


[Mor94a] C.C. Morgan. The cuppest conjunctive capping, and Galois. In 
Roscoe [Ros94], pages 317–32.

[Mor94b] C.C. Morgan. Programming from Specifications. Prentice-Hall, 

Cooke, and Peter Wallis, editors, Proceedings of the BCS-FACS 7th

ewic.bcs.org/conferences/1996/...
...refinement/papers/paper10.htm


Bibliography


Index of citations

[ABL96], 124, 345
[Abr96a], 5, 108
[Abr96b], 77, 325
[AH90], 89
[AL96], 124
[ASH93], 263, 268
[AZP03], 105, 216
[Bac78], 5
[Bac88], 5
[BAPM83], 215, 246, 247, 249, 266, 268
[BB06], 43, 47
[BdA95], 180, 290, 291
[BFL+99], 263
[BKS83], 77
[Boo82], 289
[BvW00], 218, 227, 228, 238, 240
[BvW93], 38
[BvW96], 246, 262
[BvW98], 5, 36, 110, 195, 242, 289
[Chr90], 164
[CM88], 77, 105, 215
[CMA92], 216
[Coh00], 242
[CY95], 105
[dA99], 295
[dAH90], 78, 310
[dAM01], 291, 309, 310
[DFP01], 105
[DFP02], 100
[DGJO02], 6, 125
[DGJO03], 164
[dHDV02], 36, 77
[DjJ71], 88
[DjJ97], 4, 7, 19, 21, 28, 38, 71, 82, 83, 135, 145, 162, 182, 184, 198, 199, 218, 224, 249
[DIM83], 310
[DP90], 25
[dRE98], 107
[Eda95], 220
[EH96], 247
[Ene90], 246, 262
[Eve57], 296, 310
[Far99], 180
[Flt71], 17, 73
[FH84], 36, 77, 246
[FHM90], 341
[FHM95], 245
[Flo07], 5, 38, 77
Index of citations

[Fra86], 105
[FS03], 216
[FV06], 164, 180, 263, 295, 304

[GM91], 110
[GM93], 242
[Go03], 245
[GR97], xi, 351, 354
[Grp], 110, 124
[GS92], 16, 41, 54, 76, 100, 134, 284, 297
[GW86], 52, 134, 222, 283, 284, 291, 303

[He89], 64
[Her90], 56
[He92], 130, 142, 162, 163
[He95], 252
[HHS87], 108, 109
[HI94], 263
[HK94], 78
[HK97], 290, 310
[HM94], 162, 216, 291
[Hoa69], 38, 77, 135, 184, 199
[Hon03], 73
[HP01], 93, 106
[HP02], 18, 164
[HS86], 216, 291
[HSM97], 36, 137, 139, 164, 291, 316, 319, 322
[HSP83], 78, 216
[Hur02], 207, 210, 216
[Hut03], 124, 164

[JHSY94], 164
[Jon86], 5
[Jon90], 18, 36, 135, 137, 164, 196, 216, 238, 333

[KB00], 164
[Koe81], 4, 15, 18, 131, 135, 164, 319
[Koe83], 239, 246, 252, 293, 310
[Koe85], 4, 15, 36, 135
[Kuh03], 174, 180, 262, 275, 341

[Lam80], 247
[Lam83], 247
[LPZ85], 246
[LR94], 89
[LS82], 246
[LS89], 164
[LS91], 124, 164
[LS94], 164, 327
[LvdS92], 252

[Mc01], 291
[Mc02], 88, 216, 268, 289
[MG96], 124
[MM01a], x
[MM01b], x, 183, 184, 220, 238, 239, 297, 331
[MM01c], 242, 295
[MM02], x, 293, 304, 306, 307
[MM03], 289, 295
[MM04a], 58, 74, 215
[MM04b], xi, 293, 351
[MM05], x, 310
[MM99a], x, 265
[MM99b], x, 295
[MM99c], 216, 235
[MM99d], x
[MM99e], ix, x, 129, 291
[MMSS96], 138
[MMSS97], xi, 348, 349, 351, 352, 354
[MMT98], ix, x, 107
[MM], 306, 309
[Mon01], 124
[Mon87], 5, 138, 239
[Mon88a], 113, 122
[Mon88b], 5, 138, 239
[Mon90], 252
[Mon94a], 279
[Mon94b], 5, 110
[Mon96], x, 57
[Mon97], 207
[MR95], 35, 36, 47, 105
[MV94], 348, 351, 352

[NC99], 310
[Ne99], 138, 198, 239
[Pe99], 138, 198, 239
[PRI], 105, 304
[PZ93], 105
<table>
<thead>
<tr>
<th>Reference</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rab76</td>
<td>36</td>
</tr>
<tr>
<td>Rab82</td>
<td>80, 295</td>
</tr>
<tr>
<td>Rac94</td>
<td>77, 105</td>
</tr>
<tr>
<td>Ros94</td>
<td>350, 352</td>
</tr>
<tr>
<td>Roy68</td>
<td>152</td>
</tr>
<tr>
<td>Sch86</td>
<td>341</td>
</tr>
<tr>
<td>SD80</td>
<td>36</td>
</tr>
<tr>
<td>Seg95</td>
<td>164, 318</td>
</tr>
<tr>
<td>Sha53</td>
<td>310</td>
</tr>
<tr>
<td>SHRT04</td>
<td>325</td>
</tr>
<tr>
<td>SL95</td>
<td>291</td>
</tr>
<tr>
<td>SMM96</td>
<td>xi, 354</td>
</tr>
<tr>
<td>SMI</td>
<td>x</td>
</tr>
<tr>
<td>Smy78</td>
<td>138</td>
</tr>
<tr>
<td>Smy80</td>
<td>138, 298</td>
</tr>
<tr>
<td>SPH84</td>
<td>268</td>
</tr>
<tr>
<td>Spi88</td>
<td>5</td>
</tr>
<tr>
<td>Sti92</td>
<td>246</td>
</tr>
<tr>
<td>Sti94</td>
<td>100</td>
</tr>
<tr>
<td>Sti95</td>
<td>246, 262, 294, 299, 300, 310</td>
</tr>
<tr>
<td>Sto88</td>
<td>319</td>
</tr>
<tr>
<td>Sto96</td>
<td>118, 124</td>
</tr>
<tr>
<td>Sut75</td>
<td>141</td>
</tr>
<tr>
<td>SV03</td>
<td>124</td>
</tr>
<tr>
<td>Tar55</td>
<td>38, 179</td>
</tr>
<tr>
<td>Tru71</td>
<td>341</td>
</tr>
<tr>
<td>Tur84</td>
<td>245</td>
</tr>
<tr>
<td>Van98</td>
<td>310</td>
</tr>
<tr>
<td>Var01</td>
<td>294, 309</td>
</tr>
<tr>
<td>Var85</td>
<td>78, 105, 164</td>
</tr>
<tr>
<td>yBMOW03</td>
<td>6, 125</td>
</tr>
<tr>
<td>vGB98</td>
<td>48</td>
</tr>
<tr>
<td>vNM47</td>
<td>262</td>
</tr>
<tr>
<td>War89</td>
<td>38</td>
</tr>
<tr>
<td>Yin02</td>
<td>125</td>
</tr>
<tr>
<td>Yin03</td>
<td>6, 36, 125</td>
</tr>
<tr>
<td>YL92</td>
<td>164</td>
</tr>
<tr>
<td>YW00</td>
<td>125</td>
</tr>
</tbody>
</table>
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 “qu” after a sub-item is a cross-reference to the main item beginning with those words.

{pred}, see assertion
: ⊗, see assignment
:=, see assignment, definitions
≤, see bag: bounded above
☐, see bag: empty
[ ], see bag: enumeration
#, see bag: size
+, see bag: sum
⊥, see bottom element
⌈⌉, see ceiling

[ ], see characteristic function
¬, see complement
(Qx; set• expr), see comprehension
(Qx; set | pred), see comprehension
(Qx; set | pred• expr),

see comprehension
{x; set | pred• expr},

see comprehension
¬¬, see conjugate
0, see constant function
\( \Pi, \subseteq, \emptyset \), see demonic choice
\( \cup \), see demonic decrease
\( \Xi \), see deterministic program space
\( \mathcal{U} \), see embedding
\( f \), see embedding of program
\( \mathcal{E} \), see entailment
\( \mathbb{R}^N \), see Euclidean space
\( \mathcal{E}_P, \mathcal{E}_f \), see expectation
\( \alpha \mid P \), see expectation
\( \mathcal{T}, \mathcal{P}_i, T, F, P \), see expectation	
transformer space
\( f \), see expected value
\( \text{Exp}, \text{Exp}_P \), see expected value
\( !, \), see factorial
\( \varpi \), see fixed point
\( \nu \), see fixed point: greatest
\( \mu \), see fixed point: least
\( \lfloor \cdot \rfloor \), see floor
\( f, x, f(x) \), see function application
\( f \circ g, f' \), see functional composition
\( B \leftarrow A \), see function: reverse total
\( A \rightarrow B \), see function: total
\( \mathcal{RS} \), see gambling game
\( \mathcal{C} \), see gambling game: colour
\( \mathcal{G}, \mathcal{F}, \), see gambling strategy
\( \mathcal{G}_0, \mathcal{F}_0 \), see game tree
\( \wp, \), see greatest pre-expectation
\( P \vdash \{ Q \} \), see \( P \vdash \{ Q \} \)
\( \Rightarrow \), see implication
\( \Gamma \), see intuitionistic
\( \infty \), see infinity
\( \mathbb{Z} \), see integers
\( [a,b) \), see interval
\( \langle \cdot \rangle \), see Kleisli composition
\( \lambda : \), see lambda notation
\( \uparrow, \downarrow \), see local cross-reference
\( \uparrow \beta \), see maximum
\( \downarrow, \), see maximum
\( \min \), see minimum
\( \mathbb{N} \), see natural numbers
\( \perp \), see nontermination state
\( \pi \), see point distribution
\( F, F^+ \), see powerset
\( P, \), see powerset
\( t^+, \tau^- \), see predicate transformer embedding
\( \square, \), see probabilistic choice
\( \varnothing, \), see probabilistic choice
\( \mathcal{L}, \), see probabilistic comprehension
\( \& \), see probabilistic conjunction
\( \Rightarrow, \), see probabilistic implication
\( \vdash \), see probabilistic implication
\( \mathbb{R} \), see probabilistic relational semantics
\( \Delta \), see probability distribution
\( \Pr, \), see probability distribution
\( \square, \), see probability distribution: refinement
\( \prod \), see product
\( \Box \), see quantitative modal \( \mu \)-calculus
\( \alpha, \beta \), see random variable
\( \text{iid} \), see random variable
\( \mathbb{R}, \mathbb{R}_2 \), see real numbers
\( \mathcal{S}, \mathcal{S}^+ \), see relations
\( \mathcal{C} \), see sets of distributions
\( (\cdot k) \), see shift by \( k \)
\( S, S_1, S_A, S_b \), see state space
\( \mathcal{S} \), see sub-distribution space
\( \cdot \left( \leftarrow \right) \), see substitution
\( \mathcal{C}(OP) \), see substitution: implicit
\( \Box, \), see substitution
\( \sum \), see summation
\( \forall X, \exists X \), see temporal logic
\( 0, 0 \), see temporal logic right-association
\( \mid, \), see temporal logic semantics
\( \forall, \), see valuation
\( \cdot, \), see weakest precondition
\( \wp, \), see weakest precondition
\( 0-1 \), see Zero-One
\( \Box, \), see always
\( \diamond F \), see eventually
\( 0- \), see next-time
\( \triangleright, \), see unless
\( \triangleright, \), see wp
\( \mathcal{A} \), see
\( \forall X, \), see temporal logic
\( \hat{A} \), see
\( \text{Åbo Akademi}, ix \)
\textit{abort}

is bottom (\( \perp \)) among programs,
131, 133
is (pre)-deterministic, 131
"jail" metaphor for, 12
not normalised, 18
not up-scaling, 184
refined by any behaviour, 285
syntax, 19
see also
complementary...semantics,
expectation...semantics,
informal semantics,
literal...semantics
aborting behaviour
called “divergence”, 249
in multi-way choice ⊕, 20
in next-time operator, 249, 251
indicated by one-deficit, 20, 112
removed by refinement, 172
J.-R. Abrial, 5ff, 6, 77, 84, 114, 216
abstract, probabilistic choice qv
abstract interpretation, 124
abstraction, 5, 36
complementary to refinement,
see refinement
full qv
relation, 124
see also demonic choice
action systems, 77
add n to b, see bsg
adjoint, 279
conjunction and implication, 279
see also probabilistic conjunction,
probabilistic implication
adversarial scheduling, see scheduler
agent
angelic or demonic, 275
maximising or minimising, 180
L. de Alfaro, see de Alfaro
algebra, 265ff
Borel, see σ-algebra
Kleene, viii
modal, 262
omega qv
program qv
σ- qv
temporal, see quantitative...logic
algebraic properties of transformers, 241
almost
certain, see termination
fair, see scheduler
impossible, 41, 44, 263
none, 297
α, see random variable
α-component, 229
alternating fixed points, 277, 294
alternation-free formulae,
see temporal logic
always (G, G), 248
as a game, 261ff
as iteration, 250
illustrated by coin flips, 273-4
informal standard semantics, 250
quantitative, 253
special case of unless, 251
standard, 252
amplification, probabilistic, 46
C. Andriessen, 124
angelic
choice operator, 227
interaction with demonic choice,
246, 271
nondeterminism, 78, 242
programs not sublinear, 231
programs semi-sublinear, 232
transformers, 227-31, 247
anti-refinement, 80
anti-symmetric, 25
approximate, refinement qv
J. Aspnes, 80
assertion (\{pred\})
as command, 110, 225
as comment, 4ff, 209
assignment
generalised (\{\circ\}), 114
syntax, 19
written :=, 7
see also
expectation...semantics,
informal semantics
associativity, quasi, 322
see also probabilistic conjunction
asynchronous network of processors, 60ff
atomic action, 92
automata
Büchi, 291
I/O, 164
automated reasoning,
see higher-order logic
auxiliary invariant, see reasoning within an invariant
average, 16, 255
see also expected value
Axiom of Choice, 25
Axioms for standard branching-time
temporal logic, 266
A. Axiz, 263
B
The B Method, 216
abstract-probabilistic (qB), 235
probabilistic (pAMN), 325
R.-J.R. Back, ix, x, 36, 38, 77, 195, 218, 228, 238, 242, 246, 262
bag (multiset)
 \( \text{add} \ n \ \text{to} \ b, \ 82 \)
bounded above (\( \leq \)), 83
empty bag (\( \emptyset \)), 82
enumeration (\( \{ \} \)), 81
maximum of (max), 84
size of (\#), 82
sum (+), 82
take \( n \) from \( b \), 82
C. Baier, 164
basketball, 174
M. Ben-Ari, 206, 280, 291
B. Béard, 263
\( \beta \), see random variable
A. Blanco, 180
bisimulation, 124ff, 164
D. Björner, 5
Borel algebra, see \( \sigma \)-algebra
bottom element (\( \bot \)), 179
of a partial order, 133
bound variables, scope always delimited explicitly, 16, 61, 72,
131, 155, 194, 247
bounded
expectation \( q \varepsilon \)
bound monotone convergence, 335
bounded nondeterminism, see continuity

\( BPP \), see complexity class
F. van Breugel, see van Breugel
Büchi automata, 291

C
C, see gambling game: colour
C3, see sets of distributions
Cambridge University,
Computer Laboratory, x
card-and-dice game
operational semantics for \( pGCL \), 14
see also Programs
D. Carrington, ix
casino manager, see demonic choice
Cauchy closure, 139
see also sets of distributions
ceiling (\( \lceil \cdot \rceil \)), 99, 193
Orieta Celiku, x
chain, 298 see also complete partial order
K.M. Chandy, 77
chaos, 289, 335
characteristic function (\( \{ \cdot \} \)), 13, 224
converts to expectation, 7
expected value of, 13, 16, 254, 316
see also embedding
choice coordination, see Programs
I. Christoff, 164
classical logic, as opposed to
temporal, 247
closed set, see Euclidean space
closure, see sets of distributions
clump
\( \text{can be smooth, 338ff} \)
see also sets of distributions
E. Cohen, 242
coin-flipping
\( \text{fair \( q \varepsilon \)} \)
\( \text{illustration of always } \square, \ 274 \)
\( \text{illustration of eventually } \Diamond, \ 272 \)
\( \text{illustration of next-time } \Diamond, \ 270 \)
\( \text{implementation of } \mu \Diamond, \ 210 \) see also
Programs
thin and fat coins, 268ff
three coins, 72
triple six-sided, 269
two coins, 327ff
colour (\( C \))
\( \mu, \nu \)-generated, 306ff
see also gambling game
commutativity, quasi-, 322
compactness, see Euclidean space
complement (\( \neg \))
for Booleans, 7
for probabilities, 7, 26
complementary and consistent
semantics, 154, 159ff. 160
abort, 179
demonic choice, 156
recursion, 178
sequential composition, 154
see also semantics
complete partial order, 25, 148, 179, 185
abbreviated cpo, 25, 139
bottom element (⊥), 133
chain in, 25
CS is, 130
directed subset of, 25, 139, 147, 234, 235
DS is, 131
ES is if one-bounded, 183, 299
ES is not in general, 24, 25, 148
flat, 142
ES is, 139
S is, 142
S is, 131
top element, 183
TS is if one-bounded, 183
TS is not in general, 24, 148
completeness of variant rule, 193
for probabilistic guards, 205
complexity, expected, 64, 268
complexity class
BPP, 47
ZPP, 47
composition, functional qv
compositionality
counter-example, 315
requires full use of expectations, 15, 313ff. 315
comprehension
(Qx: set • expr), 61
(Qx: set | pred), 194
(Qx: set | pred • expr), 155, 229
{x: set | pred}, 72
{x: set | pred • expr}, 140
probabilistic qv
computability, see real numbers
computational model, see semantics
computations, unending, 21, 268
concurrency, 77, 164, 310
see also bisimulation
concurrent game, 78
see also concurrency
conditional (if, then, else, fi)
defined in terms of y^{fi}, 19
hybrid with then/else and guards combined, 82
see also informal semantics
confidence measure, see statistics
conjugate (·), 17
in choice-coordination program, 82
minimum of (·), 82
conjunctivity, 6, 28, 145, 161, 162
generalised by sub-linearity, 28, 31, 221
implied by sub-conjunctivity, 31, 237
in temporal logic, 249
is complete, 162ff
of modal algebra, 295
of standard transformers, 31, 237
of next-time, 262
positive, 142, 161, 163, 249
sub-, see probabilistic conjunction
see also healthiness conditions
constant function (·), 17
continuity, 139, 145, 147
and fixed points, 179
bounded, 147
bounded nondeterminism, 71, 220, 335–7
chaos is not, 289
convenient assumption, 26, 289
distributes limits, 147, 148
gδ style, 147
explicit assumption, 219ff
fails for retraction (·)−, 235
image finiteness, 21, 71
implied by sub-linearity, 147, 234
its purpose, 148
not imposed, 288, 289
of P (·), 234
of expectation transformers, 62, 147, 178, 333ff
of predicate transformers, 143
of transformer-transfomers, 178
picture of, 175
preserved by suprema of finitary infima, 225
topological—agreeing with ⊔, 298
unbounded nondeterminism, 71, 220, 335–7
see also healthiness conditions
convexity, 138
picture of, 168
see also sets of distributions
countable
closure of σ-algebra, 297
state space $q_S$
coupling invariant,
see data refinement
C. Courcoubetias, 105
cowboys, see Duelling Cowboys
$cpo$, see complete partial order
customer-oriented, data refinement $q_v$

$D$
$DS$, see deterministic program space
data refinement, 57, 64, 107–24, 209, 242
as sub-commutation, 109
customer—vs. supplier-oriented,
108, 113, 121
defined, 109
intermediate step, 122
is transitive, 122
of eventually $\bigcirc$, 342
of next-time, 296
proved by simulation, 109
via coupling invariant, 122, 209, 210
data type, 107ff., 108
abstract, 108, 110
concrete, 108, 110
L. de Alfaro, 78, 180, 291, 310
W.-P. de Roever, 107
E.P. de Vink, 36, 77
deadlock
dining philosophers, 88
impossible in Rabin and Lehmann’s algorithm, 92
defiance probability, 89
definitions, written $\geq$, 6
$\Delta$, see probability distribution
demonic choice (71), 4ff
blind-choice metaphor, 12
casino manager, 261
from empty set, 21
implemented by a demon, 9, 26
indicates abstraction, 5, 9, 36, 45
inherent in $\geq_P$, 50, 275, 326
interaction with angelic, 246, 271
interaction with probabilistic choice, 36, 51, 164, 254, 326
interaction with probabilistic choice, avoiding, 328
means pre-expectation is only a
lower bound, 23, 65
non-Markovian, 284
not factored out, 284
picture of, 168
refined by probabilistic choice, 10,
26, 45, 80, 139, 220
syntax, 19
tries to minimize pre-expectation,
144
written $\in \mathcal{F}$, 21
see also
complementary…semantics,
expectation…semantics,
informal semantics,
nondeterminism,
probabilistic relational…
Programs
demonic decrease ($\leq$), 114
demonic expressions, 207
demonic probabilistic model, 139
Demonic program
over finite state space, 226
sublinear but not additive, 226
see also Programs
J.J. den Hartog, 36, 77
J. Desharnais, 125
deterministic
includes probabilistic, 24, 27
literally, 130
only pre if nonterminating, 27,
130, 222
deterministic program, 130–3, 221–4
additive, 224
characterised by linearity, 28,
66, 221–4
definition, 222
embedded, 132
example, see Programs
has repeatable behaviour, 12, 135
is $\subseteq$-maximal, 133
need only be tested at points, 112
picture of, 166–8
probabilistic, 131
probabilistic depicted, 167
space (13), 131
standard, 221
see also
probabilistic relational . . . ,
standard relational . . .
E.W. Dijkstra, 4, 5, 6, 21, 38, 135ff, 143ff, 158, 162, 175, 199, 236
-first reasoning, 4
dining-philosophers problem qv
see also
Guarded Command Language
dining philosophers, 90, 88–98
generalised, 106
in contention, 91
variant function for, 98
see also Programs
directed set,
see complete partial order
discrete
sub-probability measure, 36, 131
see also probability distribution
disjoint union, 213
disjunctivity, 28, 224 see also linearity
distribution, probability qv
distributive, quasi- 323
divergence, see aborting behaviour
do, see iteration
do-it-yourself semantics,
see semantics
S. Dolev, 310
domain equation, 164
Dueling Cowboys, 211
see also Programs
M. Dufot, 105, 106
dyadic rational,
see probabilistic choice
$E$
$E_1$, $E_2$, $E_3$, see expectation
$\exists x$, see temporal logic
A. Edalat, 220
efficiency, increased in random
algorithms, v
see also probabilistic
amplification
else, see conditional
embedding
$[]$ omitted, 209, 211, 316
$[false], [true]$, 7
$\mathcal{F}$ of a standard deterministic
program $f$, 132
of $\mathcal{F}$ is $\mathcal{F}$, 20
predicates to expectations ([ ]),
5, 19 see also characteristic
function
relational-to-transformer, 144
standard, 232
K. Engelhardt, 107
entailment (\textit{\textit{\textquoteleft}}), 20
also written $\textit{\textit{\textquoteleft}}$, 39, 267
see also probabilistic implication
environment, see valuation
equivalence ($\equiv$)
differs from $\approx$, 19
see also probabilistic implication
equivalence of given-strategy games
and logic, 306
equivalence of $\mathcal{G}Ms$ and games, 308
Euclidean space ($\mathbb{R}^N$), 139ff,
149, 165–80
closed subset of, 139
compact subset of, 141, 152, 179,
180, 221, 334
finitary half-space closed, 333
infinitary half-space closed, 333
infinitary half-space not closed, 334
metric for, 332
of infinite dimension, 130, 332
program states are dimensions, 332
topology for, 332ff
evaluation, 36 see also probability
distribution
event, 16
not every subset allowed, 16, 297
$Event-B$, 77
eventually ($\textit{\textit{\textquoteleft}}$, $\mathcal{F}$), 248, 342–4
as a game, 255ff
as iteration, 249
composition of several, 94, 95
double equals single, 267, 273, 278
illustrated by coin flips, 271–3
informal standard semantics, 249–50
quantitative, 253
standard, 251
◊ excluded miracle, 343
◊ monotonicity, 343
◊ scaling, 243
H. Everett, 296, 310
exact, invariant \( qv \)
existence of fixed point, 148
exotic, expectation transformer \( qv \)
\( \text{Exp}, \text{Exp}_a \), see expected value
expectation, 6, 7
bounded above, 13, 25, 66, 68ff
bounded space \( (\mathbb{E}_b) \), 332
called probabilistic predicate, 13
expressions vs. functions, 25, 130, 136, 183
finite, 220, 332
finite restriction \( (\alpha \downarrow P) \), 220, 332
finite space \( (\mathbb{E}_f) \), 332
"glue" between reasoning steps, 24, 271
greatest \( qv \)
idiom for nonzero, 99
intuition behind, 22ff
is random variable, 17
mixed-sign, to be avoided, 70
negative, 100
non-negative, 24, 66, 68ff
not confused with distribution, 137
one-bounded, 31, 181–215, 247, 252, 296
post-, 13, 17
pre-, 13, 17
proper, 63, 231, 253, 254, 256, 268, 269, 271, 273
"safe", 34
space \( (\mathbb{E}) \), 24, 143
standard, 40, 137
support of, 332
unbounded, 34, 216
see also random variable
expectation transformer, 15
angelic space \( (\mathbb{T}_a) \), 227
continuity not imposed \( qv \)
demonic space \( (\mathbb{T}_d) \), 225
deterministic space \( (\mathbb{T}_d) \), 223
does not distribute \( \min, \) 30

exotic, 238, 240, 242
hierarchy, 218, 221–38
infeasible, 138, 238, 250
linear, 222
regular, 141–64, 219
regular space \( (\mathbb{T}_r) \), 145, 219ff
semi-linear, 234
semi-sublinear, 228
space \( (\mathbb{T}, \mathbb{PTS}) \), 24, 157ff, 334ff
standard space \( (\mathbb{T}_d) \), 232
standard-preserving, 235
sub-conjunctive, 31
sublinear, 28, 146
expectation-transformer semantics, 142, 157
assignment :=, 7
demonic choice \( ! \), 9
iteration, 38
of \( \mathbb{PCL} \), including \texttt{abort, skip}
and recursion, 26
probabilistic choice \( \nu \), 7
sequential composition, 15
variants of \( \nu \), 20
see also expectation… space, semantics
expected, complexity \( qv \)
expected value (\( \text{Exp} \)), 16, 134
explicit in formulae, 316ff
picture of, 173
same as average, 5, 13, 134
written \( \bar{\bar{f}} \), 134
\( \text{Exp}_a \), in proof of loop rules, 76
see also average, expectation

\( \mathbb{F} \)
—
\( \mathbb{F} \), see eventually
\( \mathbb{F} \), see embedding of program
\( \mathbb{F}, \mathbb{F}^h \), see powerset
factorial (!), 51
fail command, in steam boiler, 119ff
fair coin (flipping), 316
nearly, 6
reasonably, 209
fairness, 89, 93–5, 105ff, 263
almost, see scheduler
\( k \)-fairness, 60
probabilistic, 61
Farlas’ Lemma, 130, 180, 341
use of, 152
fault coin, see coin-flipping
fault-tolerance, v see also Programs:
steam boiler
faulty 2-skitter, 111
faulty N-skitter, 111
faulty factorial, 52 see also Programs
faulty skipper
for p = 1/2, 115
see also Programs
feasibility, 29, 147, 238
eventually is, 257
example of use, 183, 187, 257, 296
excludes miracles, 29
generalises strictness, 29, 221
in temporal logic, 249
more general treatment without it,
144, 238
of transformers, 147
picture of, 175
preserves one-boundedness,
183, 296
 preserving, 26, 148
standard, 249
two kinds for probability, 239
wp is not, 250
see also
expectation transformer,
healthiness conditions,
miracle,
strictness
feature interactions, 195
need semantics to avoid, 197
Y.A. Feldman, 36, 77
fl, see conditional
Fibonacci numbers, 115
C.I. Fidge, 216
finite state space, see state space
Finite-Intersection Lemma, 152, 180,
336, 337
Firewire, IEEE protocol, 216
first-order logic, 4
vs. second-order, 38
fixed point, 77
alternating qe
appeal to continuity, 148, 179
avoids circular argument, 197
general f, x ≥ x property, 185, 278
general f, x ≤ x property, 102, 278
greatest (r), 183, 184, 185,
251, 300
is a limit, 179, 259
least (µ), 21, 26, 38, 61, 102, 177,
179, 251, 299, 300
N- or w-limit formulation, 178,
179, 288
special proof for existence of, 148
flat qe, see complete partial order
flip-a-coin
generates E, 209
see also Programs
floor ([ ]), 193
R.W. Floyd, 5, 6, 77
four semantic models, 143
free will, 275
fruit machine, see polar machine
full abstraction, facilitates program
algebra, 319
function
characteristic qe
constant, 17
generating, 291
homogeneous, 135
measurable, 305
reverse total (B ← A), 24, 142
strict, 133, 143
total (A → B), 130
function application (f(x)), 6, 248
not written f(x), 6, 53
functional composition (f ◦ g), 154
n-fold (f^n), 179
functional properties, refinement qe
futures market, example, 304
−
G, see always
Galois connection, 226, 279
between HS and T S, 334–8
see also adjoint
gambling game, 6, 11–15, 246,
254–62, 293–309
colours C used for fixed points,
299ff
concurrent, 78
maximin value of, 304
maximising player Max, 299ff
minimax value of, 262, 294, 300,
304, 306
minimising player Min, 290ff
path, 299
payoff, 300
round (\(R^5\)), 296
scheduler-luck, 310
tree, see game tree
value of fixed-strategy, 205
zero-sum, 295, 300
gambling strategy, 246, 255ff
decided in advance, 275, 305
existence of memoryless, 308
formalised as functions (\(\sigma, \tau\)), 305
memoryless (\(\sigma, \tau\)), 256, 284, 294,
304, 307–8, 318
optimal, 256, 257ff, 295, 303
optimal not unique, 261, 276
winning, 295
with full memory, 305
see also martingale
game
card-and-dice, see Programs
gambling qv
Monty-Hall, see Programs
path, see gambling game
three-up, see Programs
game semantics, 11–15, 254–62, 268
see also semantics
game tree, probabilistic (\(\{0\}\)), 305
P.H.B. Gardiner, 124, 242
GCL, see Guarded Command Language
generating function, 291
general distribution, 69, 195ff, 308, 336
Geometric interpretation of
sublinearity, 176

glue, see expectation
Golden Ratio, 115
gp, see greatest pre-expectation
greatest fixed-point, see fixed point
greatest liberal pre-expectation
\((wp)\), 183, 239 see also
liberal...semantics
greatest lower bound, 25
greatest pre-expectation (gp), 13
as a modal operator, 252
greatest guaranteed probability of
win metaphor, 12
not written gp, 26
picture of, 174
see also defiance probability,
expectation,
expectation...semantics,
pre-expectation
guard
iteration qv
probabilistic, see iteration
Guarded Command Language (GCL),
4ff, 158

K —
see probabilistic relational
semantics
half-space, see Euclidean space
J.Y. Halpern, 164, 18, 78
Halting Problem, solved, 261
H. Hansson, 263
D. Harel, ix, 36, 77
S. Hart, 78, 216, 291
J.J. den Hartog, see den Hartog
Hausdorff space, 141
I.J. Hayes, ix, x
Jifeng He, 36, 108, 137, 139, 316, 322
healthiness conditions, 18, 28–34, 129,
145–9, 148, 217, 218
apply for infinite state spaces,
221, 338
apply to temporal logic, 249
defined, 29
derive from sublinearity, 146
for regular transformers, 148
justify program algebra, 161
not satisfied by wp, 184
pictures of, 175–7
strictness, see feasibility
weakened by angelic choice, 291
see also
\(\odot\)-subdistribution,
additivity,
conjunctivity,
continuity,
feasibility,
monotonicity,
scaling,
semi-sublinearity,
sub-additivity,
sublinearity
E.C.R. Helmer, 43, 64
T. Henzinger, 78, 310
O.M. Herescu, 93, 106
M. Herlihy, 89
Herman’s Graph, variant for, 59
Herman’s Ring, 27, 215
exact running time, 58, 74
generalisations, 59ff
variant for, 59
see also Programs
W.H. Hesselink, 252
heuristics, for invariants \( qv \)
hierarchy, see standard transformer, transformer
higher-order logic
Skolem function, 154
theorem prover (HOL), x, 162, 216, 291
see also second-order logic
Thai Son Hoang, x, 211
C.A.R. Hoare, 5ff, 6, 36, 77, 108, 133ff, 199
style reasoning, see Hoare triple
Hoare logic, 36ff, 77ff see also Hoare triple
Hoare triple, 4, 313–16
probabilistic \( (p \triangleright \{Q\} \text{ prog } \{R\}) \), 313
sequential composition, 314
specification, 33
HOL, see higher-order logic
J. Hurd, x, 162, 207, 210, 216, 291
M. Huth, x, 124, 164, 290
hybrid, see conditional
hyper-
octant, 152, 332
plane, 150, 172, 176, 333ff, 341ff
pyramid (tetrahedron), 152,
160, 230 see also probability distribution space

I

I/O automata, 164
idempotent, 31, 322
idom
nonzero expectation, 99
restricted implication, 39
\( \text{iid} \), see random variable

image-finite, 71, 142, 146
see also continuity,
standard relational semantics
implication \( (\Rightarrow) \)
Boolean differs from \( \Rightarrow \), 19, 182
idiom, 39
standard-embedded, 278, see also probabilistic implication inaccessible, see Smyth topology
independent, random variable \( qv \)
infimum
of empty set is infinite, 138, 239
over a set \( (\Pi_F) \), 232
infinite state space, see state space
infinitely many final states
still continuous, 72
see also Programs; geometric distribution
infinity \( (\infty) \)
adjoined to the reals, 25, 70, 134, 138, 144, 150
see also infimum, miracle
informal semantics
abort, 26
assignment, 26
conditional, 19
demonic choice, 26
probabilistic choice \( p \triangleright \), 26
probabilistic loop-guard, 196
sequential composition, 26
skip, 26
standard temporal operators, 248–51
see also semantics
integers \( (\mathbb{Z}) \), 7
interval, closed-open \((a,b)\), 56
intrinsicly unbounded, invariant \( qv \)
invariant, 6, 39ff, 184–91, 199ff
auxiliary, see reasoning within an invariant
bounded in loop rules, 71
coupling, see data refinement
exact, 67, 69, 187, 208, 212, 213
for iteration, 39
heuristics for, 40, 48, 210–14
intrinsicly unbounded, 71, 330
of iteration, 200
partial correctness, 185
principal reasoning tool, 77
probabilistic, 39
reasoning within *qe*
standard, 30, 43, 53, 66, 71, 203
strong, 191
technique is complete, 200
typical form, 40
unbounded in loop rules, 71
weak, 182, 190, 203

Invariant-implies-termination loop rule, 188, 202, 329
iteration (do...od)

algebraic properties, see program algebra
alternative rules, 56
counter-example to rule, 71
demonic rules, 328ff
extended rules, 93
guard, 39, 196
invariant of *qe*
justification of rules, 74
loop rules, 43, 203
multiple guards, 83, 187
only weak invariance necessary, 186
partial correctness, 199, 200
probabilistic guard, 40, 69, 195–214
semantics (as fixed point), 21, 198, 330
semantics for probabilistic guards, 195
semantics for *always*, 50
semantics for *eventually*, 249
syntax, 21
total correctness, 42, 43, 203
see also variant, liberal ...semantics

J —
jail, as metaphor for abort *qe*
Zhendong Jin, x
C. Jones, 18, 36, 135, 164, 196, 216
C.B. Jones, 5
B. Jonsson, 164, 263
jump, see random walk: general

K —
k-fairness, see fairness
B.M. Kapron, 78
Kleene algebra, viii

Kleisli composition (·*)
of programs, 135
D. Kozen, 6, 18, 36, 131ff, 135, 164,
230, 246, 252, 294ff, 310, 319
logic for sequential probabilistic
programs, 4, 15
R. Kurki-Suonio, 77
M.Z. Kwiatkowska, 105, 164, 290, 304

L —
lambda notation,
for functions (*λ*), 131
K.G. Larsen, 124, 164
Las-Vegas algorithm, 43, 45
see also Programs
lattice, 25
Law of the Excluded Miracle, 29, 175
is strictness, 145
relaxed, 239laws, program algebra *qe*
layered variant, see variant
leadership election,
see Programs; self-stabilisation
least fixed-point, see fixed point
least upper bound, 25
D. Lehmann, dining-philosophers algorithm,
see Programs
lexicographic order, 85, 87
see also variant
liberal expectation-transformer
semantics, 184ff
abort, 184
iteration, 184
recursion, 184
see also greatest liberal ..., sematics
limit point, 334
T. Lindner, 124
linear programming, 152, 180, 310, 341
linearity, 66
characterises determinism, 218, 221–4
implies standard disjunctivity, 28
super-, 28
live lock, in the dining philosophers, 88
local cross-reference (†,†), 65
logic
   classical $qv$
   higher-order, second-order,
       first-order $qv$
   quantitative temporal $qv$
   standard $qv$
   temporal $qv$
   three-valued $qv$
   working beyond, 34, 177, 288, 331, 338
logical constant,
   captures initial value, 99
loop, see iteration
$LTL$, 309
J.J. Lukkien, 252
N. Lynch, 164

M
A.K. McIver, 88, 137, 216, 291
Macquarie University, x
R. Majumdar, 291, 310
Z. Manna, 266
Markov
   chain, 73
decision process, 164, 180, 263
   process, 106, 115, 284, 291
   see also demonic choice
E. Martin, x
martingale
   gambling strategy, 44ff
   program, 45
   program revisited, 65
   see also Programs
Max, see gambling game
   max, see maximum
   maximin, see gambling game
maximum (max), 29
   as unary operator, 29
   see also bag of expectation ($\cup \beta$), 147
   written $\bar{U}$, 227
measurable function, over $\sigma$-algebra,
   305
measure
   integration over, 134
   not needed even over infinite state
   space, 219
   see also evaluation
memoriless strategies suffice, 307
meta-theorems, 196, 199–206
   vs. theorems, 199
metric space, 141 see also Euclidean
   space, topology
Miller-Rabin primality test, 47, 50
   imprecise probabilities, 275
   see also Programs
Min, see gambling game
   min, see minimum
min-straggler, in a token graph, 59
   minimax, see gambling game
Minimax defined for Koen
   interpretation, 307
   minimum (min), 9
m miracle, 242ff
   Law of Excluded-, 145, 175
   requires $\infty$, 150
   see also feasibility, program
Jay Misra, 77
modal $\mu$-calculus, 246
   quantitative $qv$
   modal algebra, standard, 262
   modal logic, 245
   modality, angelic $\exists$, demonic $\forall$
   298, 302
model checking, 78, 105, 124, 180,
   216, 268, 287, 304
   modular reasoning, 354, 267
D. Monniaux, 124
monotonicity, 6, 29, 145, 147
   example of use, 314
   has no picture, 175
   of transformers, 147
   see also healthiness conditions
Monte-Carlo algorithm, 44, 47
   nontermination allowed, 44, 235
   see also Programs
Monty Hall
   game, 6, 21ff, 27–8
   program, 22
   see also Programs
C.C. Morgan, 5, 88, 124, 242
J.M. Morris, 5, 252
moth, mites in the ears of, 80
R. Motwani, 105
$\mu$, see fixed point; least
   see also recursion
$\mu$-calculus, modal $qv$
multiplication, symbol sometimes
   omitted, 53
multiset, see bag

\[ N \]

\[ N \]  , see natural numbers
\[ N_{b,c} \]  used in martingale example, 45
N. Narashima, 310
natural numbers (\[ N \]), 61
G. Nelson, 198
network
asynchronous, 60ff
synchronous, 56ff
next-time (\( O(X) \), 248

as a game, 255ff
excludes miracles, 278
illustrated by coin flips, 269-71
informal standard semantics, 249
is monotonic, 278
is scaling, 278
subdistributes \( \odot \), 278
subdistributes \( p\otimes \), 278
quantitative, 252
standard, 251
non-determinism
bounded, see continuity
demonic, see demonic choice
oblivious, 318
non-termination
picture of, 171
state (\( L \)), 111, 130
G. Norman, 304
\( \nu \), see fixed point: greatest

\( \odot \)

oblivious, nondeterminism, 318
octant
hyper- \( qv \)
negative, 174
positive, 166, 172, 173
\( \odot \), see iteration
omega algebra, vi, 242
\( \omega \)-limit, see fixed point
one-bounded, see expectation
order, partial \( qv \)
Oxford University, ix
Oxford-style of Z, 5

\( \mathbb{P} \)

\( \mathbb{P} \), see powerset
C. Palamidessi, 93, 106
\( pAMN \), see The \( \mathbb{P} \) Method
Prakash Panangaden, 164
parentheses, see bound variables
partial correctness, 42, 82ff, 184-6, 190-202, 216
of gambler’s reasoning, 44
partial order, 25
payoff, see gambling game
\( P_i \), used in martingale example, 46
\( p\text{CTL}, p\text{CTL}^* \), 77, 263, 268, 290ff
\( p\text{GCL} \), 4, 6, 129
introduction, 7-15
semantics, 24-7 see also
equation . . . semantics
syntactic space (\( \text{Syn} \)), 157ff
syntax, 18-22
see also
Guarded Command Language
\( \phi_i, \phi \), see gambling strategy
\( \Pi \), see product
G. Plotkin, 36
A. Prugli, 78, 105, 216, 266
point distribution (\( \mathbb{X} \)), 132
picture of, 167
set of, 144
see also probability distribution
poker-, fruit-, slot machine, 254
polytope, 338
positive, conjunctivity \( qv \)
post-expectation
picture of, 172
see also expectation
post-condition, 4, 17, 251
powerdomain, probabilistic \( qv \)
powerset (\( \mathbb{P} \)), 138, 143
finite (\( \mathbb{F} \)), 332
non-empty finite (\( \mathbb{F}^+ \)), 142, 143
\( \Pr, \Pr_n \), see probability distribution
pre-deterministic, 27, 130, 222
body of iteration, 187
see also deterministic
pre-expectation
greatest \( qv \)
sufficient, 43
see also expectation
pre-condition, 4, 17, 251
necessary vs. sufficient, 43
weakest \( qv \)
predicate
## General index

as Boolean expression, 19
as set of states, 13
probabilistic, see expectation predicate transformer, 13, 251
embedding (\( f \)), 232
retraction (\( f^* \)), 234
primality test, see Miller-Rabin, Programs
priority, of a processor, 60
PRISM, 105, 304
probabilistic
amplification, 47 see also Programs
fairness *q* 4
powerdomain, 36, 164
probabilistic *a*-semantics of *pGCL*, 26
probabilistic always-eventually law, 272
probabilistic and non-deterministic gambling games, 15
probabilistic choice (\( p \odot q \)), 4
abstract in *qR*, 235
as a range (\( p \odot q \)), 8, 20
at least \( p \) (\( \geq p \)), 21, 50, 275, 328
between distributions, 138
between expectations, 198
can depend on the state, 19, 190, 210–14
in loop guard, 196ff
interaction with demonic choice, see demonic choice
is not free will, 275
limited to (dyadic) rationals, 210
multi-way (\( \otimes \)), 20 see also probabilistic comprehension picture of, 167
refines demonic, see demonic choice reversed (\( p \odot q \)), 20
syntax, 19
used in ordinary arithmetic, 63 see also expectation... semantics, informal semantics, probabilistic relational...
probabilistic closure (\( C \)), see sets of distributions
probabilistic comprehension (\( I \)), 20
probabilistic conjunction (\( k \)), 30
adjoint, see probabilistic implication
associative over \([0, 1] \), 31
commutative, 31
depicted, 31
example of use, 181–215
intuition for, 31ff
is sub-distributive, 31
modular reasoning *q* *p*
not idempotent, 183
related to joint-event probability, 31
probabilistic conjunctivity, implies standard conjunctivity, 31
probabilistic deterministic transforms, 135–7
probabilistic double-eventually, 342
probabilistic generalisation of standard temporal axioms, 281
probabilistic implication
\&-adjoint (\( \& \)), 277, 279
between Boolean predicates, 39, 267
differs from \( \leq \), 19
diffs from \( \Rightarrow \), 19, 182
entailment (\( \models \)), 9, 19, 24
equivalence (\( \equiv \)), 19
\( exp \Rightarrow exp \) badly typed, 19
“in all states”, 33
in semantics of iteration, 38
is embedding of \( \models \), 20
is super-distributive, 279
looks like “\( \geq \)”, 9
reverse (\( \not\equiv \)), 19
standard-embedded (\( \equiv \)), 277
see also probabilistic conjunction
probabilistic relational semantics, 137–41, 157
demonic choice, 140
deterministic program, 131
model for (\( \mathcal{L}S \)), 139, 140, 334ff
probabilistic choice \( p \odot q \), 140
sequential composition, 140
space (\( PRS \)), 157ff
see also semantics
probabilistic temporal logic, see quantitative temporal logic
probabilistic temporal operators, basic properties, 278
probabilistic transformer semantics, see expectation transformer...
probabilistically guarded iteration, syntax, 196

probability

defiance, 89
theory, 16ff

probability distribution (Pr), 16
continuous, 16, 166, 203, 203, 297
discrete, 16, 166, 219, 220, 238
discrete over infinite state space, 219, 229, 332
explicit in formula, 316ff
generalised to evaluation, 16, 36
geometric, 16, 196, 208 see also Programs

infinite, 332
not confused with expectation, 137
order, 131
point (5), 132

Pr(a) in proof of loop rules, 76
space (5), 130

space as (hyper-) pyramid, 166ff, 239

space is compact, 334
stationary, 291

sub, 130

“sub” omitted, 131

uniform over [0, 1], 220
written Δ, 130, 219

see also sets of distributions

product (∏), 62

program

atomic, 160
complexity qv

depicted as a diamond, 109

depicted as a line, 160

least, see abort

maximal, see deterministic program

miraculous, 138 see also feasibility

relational semantics for, see

probabilistic relational...,

standard relational...

 transformer semantics for, see

expectation transformer...

standard transformer...

program algebra, 6, 28ff, 161-3, 170

collection of laws, 321-8

example, 10, 11, 112, 114ff,

120, 325-6

iteration, 38, 187

justified by healthiness conditions,

161

of wp/wlp, 331

of recursion, 38

program superposition, 64

Programs

card-and-dice game, 11ff

choice coordination

(Rabin’s tourists), 79-87

coin-flip implementation of pr, 210

counter-example to loop rules, 71
demonic, 8, 226
demonic not additive, 226
demonic/probabilistic, 21ff
deterministic, 7
dining philosophers, 88-98

duelling cowboys, 210-14

default factorial, 51-3

default skipper, 111, 115

flip a coin, 6

geometric distribution, 69, 196, 208

Herman’s (token) ring, 56-61

Las Vegas, 44-6

martingale gambling, 44-6, 64ff

Monte-Carlo, 46-51

Monty-Hall game, 22, 27ff

not continuous, 335

primality testing (Miller-Rabin),

46-51

probabilistic amplification, 46-51

probabilistic termination example, 40

random walk, bounded

two-dimensional, see Three-up

random walk, general, 99-105

random walk, symmetric, 71

refinement example, 10

self-stabilisation, 56-61

software publishing, 301-4

square root, 5

standard variant fails, 55, 56

steam boiler (safety-critical,

fault-tolerant), 117-23

Three-up game, 72ff

unboundedly nondeterministic, 335
unboundedly probabilistic, see geometric distribution
program uncertain termination example, 62ff
proper, states, see state space
PRS, see probabilistic relational semantics
PTS, see expectation-transformer semantics
R. Pucella, 18, 164
pyramid, hyper- qe

—
qB, see B Method
qMµ, see quantitative modal µ-calculus
qTL, see quantitative temporal logic
quantitative modal µ-calculus (qMµ), 293–309
complementary interpretations, 304–8
game interpretation ([ · · · [ϕ] ]), 305
logical interpretation (∥ϕ∥, ∥φ∥), 298, 305
quantitative temporal logic (qTL) algebra of, 265–91
called qTL, 262, 265
existential modalities, 291
introduction, 245–62
probabilistic- as a special case, 246
semantics (∥ · ∥ψ), 252–4
special case of qMµ, 294, 309
quasi-associativity, 322
quasi-commutativity, 322
quasi-distributivity, 322

R
—
RS, see gambling game
R,R≥, see real numbers
Rabin and Lehmann’s dining-philosophers algorithm, 92
M.O. Rabin, 36
choice coordination, 83 see also Programs
dining-philosophers, see Programs
see also Miller-Rabin primality test
P. Raghavan, 105
random behaviour, 4
random bit, v
random stumbler, see random walk
random variable, 16, 133–5
called expectation, 137
definition, 134
independent and identically distributed (iid), 100
written α, β, 130, 219
see also expectation
random walk, 52, 291
as paradigm for termination, 215
bounded two-dimensional, 72
general, 99–105, 283–8
general program, 101
generalised symmetric, 105
(non-)homogeneous, 100, 284ff
requiring fully numeric reasoning, 267
symmetric, 71, 99, 105
tabulated, 25
unbounded, 74
with stumbling, 287–8
see also Programs
J.R. Rao, 77, 106, 291
rational, dyadic, see probabilistic choice
ready
groups of philosophers, 94
philosophers must act, 94
real numbers (R), 134
computable, 209
non-negative (R≥), 13
reasoning within an invariant auxiliary, 66, 67ff
of a loop, 26, 211ff
see also modular reasoning
recurrent, process, 99
recursion
algebraic properties, see program algebra
consistent semantics, see
complementary . . . semantics
defines iteration, 21, 198
simple example of termination, 41
syntax (μI), 19, 21
tail, 38, 198
see also
  expectation . . . semantics,
  liberal . . . semantics
refinement (⊆), 277
(7) ⊆ (⊔), 10, 26, 45, 80, 139, 220
approximate, vii
approximate not necessary, 170
between deterministic relational
programs, 131
complementary to abstraction, 5, 9
data qv
in semantics of recursion, 38, 41
is set inclusion, 170
is “up-right” movement
geometrically, 172
of functional properties only, 107
picture of, 170, 174
probabilistic, 9, 24
simple example, see Programs
strict (⊇), 133
Refinement Calculus, 5, 36
reflexive, 25
regular transformer
is sublinear, 146
see also expectation transformer
Regulator
implementation, 121
refinement, 130
specification, 119
Reise Representation Theorem, 17
relational semantics
probabilistic qv
standard qv
relations, continuous and feasible
standard (⊇), 232
reliability, quantified, 118
retraction, 80
for standard reasoning, 80
standard, 234
transformer-to-relational (rp), 149, 251
see also continuity, wp
R^n, see Euclidean space
K.A. Robinson, x
W.-P. de Roever, see de Roever
B. Van Roy, see Van Roy
Royal Holloway College, x
rp, see retraction, wp

S
SS, see relations
S, S', S', S', see state space
S, see sub-distribution space
S, see point distribution
safety properly, 60
safety-critical system, example, see
  Programs
N. Saheb-Djahromi, 36
J. Saisas, 164
sample space, 16
J.W. Sanders, 168, ix
scaling, 29, 147
allows manipulation of bounds, 183
example of use, 34, 53, 67, 94, 190,
  191, 193, 314, 340
of transformers, 147
picture of, 175
see also healthiness conditions
scheduler
adversarial, 61, 83, 88, 93ff
almost fair, 93
built-in to semantics, 291
"real-world", 61
S. Schneider, x, 325
J.-L.A. van de Snepscheut, see van de
  Schneepcheut
second-order, vs. first-order logic, 38
R. Segala, 164
K. Seidel, ix, 137
self-stabilisation, see Programs
semantic models, all four, 143
semantic structures, 158
semantics
as a card game, 14
complementary and consistent qv
computational model, 11ff
do-it-yourself, 195-214
expectation-transformer qv
game qv
informal qv
liberal expectation-transformer qv
liberal predicate-transformer, 182
probabilistic relational qv
probabilistic transformer, see
  expectation . . . semantics
standard relational qv
standard-transformer qv
see also
quantitative temporal logic, standard temporal logic
semi-linearity, 234
characterises standard transformers, 218, 237
see also expectation transformer
semi-sublinearity, 228
characterises weak closure, 218, 230
see also expectation transformer
Separating-Hyperplane Lemma, 337, 341
infinite case, 342
use of, 150
sequential composition
consistent semantics, see complementarity... semantics
syntax, 19
see also
expectation... semantics, Hoare triple, informal semantics, probabilistic relational...
sets of distributions
called a clump, 168
Cauchy-closed, 139, 154, 334
covex, 138, 154, 180
non-empty, 138, 239
probabilistically closed (C), 139, 143, 166ff
rational for closure, 164
up-closed, 138
C. Shankland, 216
L.S. Shapley, 310
M. Sharir, 78, 216, 291
shift by k (x/k), 286
∑, see summation
2, 7, see gambling strategy
σ-algebra, 16, 36, 41, 297
Borel algebra, 36, 297, 298
defined by tree, 297, 305
equal, 297, 298
measurable function, 305
simulation
for data refinement, 110
see also data refinement
skip
informal semantics, 26
syntax, 19
see also expectation... semantics
skipper, see Programs
Skolemisation, 154
A. Skou, 124, 164
slot machine, see poker machine
smooth, see chump
Smyth order, 138, 142
Smyth topology, 298
inaccessible, 298
soundness vs. completeness, 162ff, 202
specification
debugging of, 287
Hoare-triple qv
of random stumbler, 288
of random walker, 285
square-root program, see Programs
SRS, see standard relational...
standard
expectation qv
logic, 5
means “non-probabilistic”, 7, 247
transformer, see standard transformer
standard demonic programs depicted, 168
standard double-eventually, 266
standard reasoning, via retraction qv
standard relational semantics, 142ff
deterministic program, 130
image-finite program, 142
space (SRS), 157ff
total program, 142
see also semantics
standard temporal logic
characterisation, 231–8 see also semi-linearity
hierarchy, 218
semantics (STS), 142ff, 157ff see also semantics, standard temporal logic
state, probabilistic, 135
state space (S), 130
(un)countable, 16, 209, 220, 297
final subset of (S), 136
finiteness exploited, 130, 133, 139, 142, 143, 147, 149ff, 194ff, 205ff, 294ff
infinite, 218, 219-21, 332-41
initial subset of \( S_n \), 136
original, 72
proper elements, 132
superposed, 72
with nontermination \( S_L \), 130, 143

statistics, confidence measure, 12, 221
steam boiler, 168, 117-23
changed water level, 118
see also Programs

C. Stirling, 246, 310, 262
game, 294ff

M.I.A. Stoelinga, 124
straggler, in a token graph, 59
strategic software development
example, 392
strategy, gambling \( qv \)
strict function, see function
strictness, 145
generalised by feasibility, 29, 221
see also feasibility
strong invariant, see invariant strongly connected graph, 59
structural induction, 178ff
Structure of transformer spaces, 240
STS, see standard transformer... stubler, see random walk
sub-additivity, 148, 221 see also healthiness conditions
sub-conjunctivity
example of use, 34
see also probabilistic conjunction
sub-distribution, 130
order \( \subseteq \), 131
space \( \mathcal{S} \), 130
see also probability distribution
sub-probability measure, 164
discrete \( qv \)
see also probability distribution
\( \varepsilon \)-subdistribution, 148, 239
picture of, 177
see also healthiness conditions
Sublinear transformers are regular, 153
sublinearity, 18, 129, 145, 162, 267
characterises \( \cap \) closure, 226
characterises regular transformers, 153, 218

equivalent to separate conditions, 148
example of use, 102, 314
generalises conjunctivity, 28, 31, 221
implies all other healthiness conditions, 146
not in \( \mathcal{L} \) semantics, 184
of next-time, 262
picture of, 176
variations on, 218
see also expectation transformer,
healthiness conditions
substitution implicit \( (\mathcal{C}(\mathcal{O}P)) \), 109
written : \( \cdot \rightarrow \cdot \), 6
subtraction, truncated \( (\varepsilon) \), 28, 146
sufficient, pre-expectation \( qv \)
summation \( \sum \), 16
super-additivity, 221
super-distributivity,
see probabilistic implication
super-linearity, see linearity
superposition, 64 see also state space
supplier-oriented, data refinement \( qv \)
support of an expectation, 332
of an open set, 332
symmetric, random walk \( qv \)
symmetry breaking,
in distributed algorithms, \( \nu \), 79
see also Programs: distributed consensus,
dining philosophers

Syn, see pGCL
synchronous network of processors, 56ff
syntactic sugar, 19-21, 293, 328
can be ignored, 184

\( T \)

\( T, T_\mathcal{P}, T_\mathcal{Q}, T_\mathcal{R}, T_\mathcal{P}, T_\mathcal{Q} \), see expectation transformer space
\( T_\mathcal{P,C} \), used in martingale example, 46
\( t^+, t^- \), see predicate transformer
embedding
up closure, 138
picture of, 171
standard, 142
see also sets of distributions
US quarters, 269

\( \forall \) —
\( \forall \), see temporal logic semantics, valuation

F.W. Vaandrager, 124
valuation (\( \forall \)), 297
incorporates environment, 298
F. van Breugel, 125
J.L.A. van de Snepscheut, 252
B. Van Roy, 310
Var, see temporal logic
M. Vardi, 78, 105, 164
variant, 6, 54–61, 191–5, 203–6
alternative rule for iteration, 56
complete for finite state spaces, 55,
89, 194, 205–6, 214
complete for standard termination, 55
defined, 55
eventual decrease of, 93
extended rule for iteration, 93
incomplete for probabilistic
termination, 85 see also
Programs
increasing, 56
layered (lexicographic), 60, 85,
89, 205
rule for loops, 55, 191
rule for probabilistic-guard loops,
204
standard fails, 85
VDM development method, 5
E.P. de Vink, see de Vink
J. von Wright, 38, 195, 218, 228, 238,
242, 246, 262

\( \exists \) —
Wang Yi, 164
M. Ward, 38
weak invariant, see invariant
weakest liberal precondition (\( \exists \text{lp} \)),
182ff, 250, 331
distinguished from \( \exists \text{p} \), 182
for loop, 184
not expressible with \( \exists \text{p} \) alone, 182
weakest nonterminating precondition
(\( \exists \text{np} \)), 250
is not feasible, 250
weakest pre-expectation,
see greatest pre-expectation
weakest precondition (\( \exists \text{p} \)), 7
guaranteed-win metaphor, 12
"syntactic" vs. "semantic" version,
136
written as reason in calculation
(\( \vdash \)), 48
see also
weakest liberal precondition
Martin Wising, 125
\( \exists \text{lp} \), see
greatest liberal...
weakest liberal...
\( \exists \text{np} \), see weakest nonterminating
precondition
\( \exists \text{p} \)
as embedding function, 144,
219, 334
demonic-enabled, 137
inverse \( \exists \text{p} \), 148, 334
is continuous, 179
semantics, see
expectation-transformer..., standard-transformer ...
see also greatest pre-expectation
J. von Wright, see von Wright

\( X \) —
X, see next-time
x-ray vision, 14

\( Y \) —
M. Yannakakis, 105
Wang Yi, see Wang Yi
Mingsheng Ying, 36, 125
\[ \mathbb{Z} \]

- see integers

**Z** specification language, 5

Zero-One Law, 78

  - example of use, 56, 76, 93, 101, 104, 204, 214, 258
  - for probabilistic processes, 54
  - for termination, 54
  - qTL version, 289

**ZPP**, see complexity class

L. Zuck, 105, 216