1

Semantics of Probabilistic Programming: A Gentle Introduction

Fredrik Dahlqvist and Alexandra Silva
University College London
Dexter Kozen
Cornell University

Abstract: Reasoning about probabilistic programs is hard because it compounds the difficulty of classic program analysis with sometimes subtle questions of probability theory. Having precise mathematical models, or *semantics*, describing their behaviour is therefore particularly important. In this chapter, we review two probabilistic semantics. First an operational semantics which models the local, step-by-step, behaviour of programs, then a denotational semantics describing global behaviour as an operator transforming probability distributions over memory states.

1.1 Introduction

A *probabilistic program* is any program whose execution is probabilistic. This usually means that there is a source of randomness that allows weighted choices to be made during execution. Given an initial machine-state, in the event that the program halts, there will be a distribution describing the probability of output events. Any deterministic program is trivially a probabilistic program that does not make any random choices. The source of randomness is typically a *random number generator*, which is assumed to provide independent samples from a known distribution. In practice, these are often *pseudo-random number generators*, which do not provide true randomness, but only an approximation; however, it is possible to construct hardware random number generators that provide true randomness, for example by measuring a noisy electromagnetic process.

Reasoning about deterministic programs usually involves answering binary yes/no questions: *Is the postcondition always satisfied? Does this program halt on all inputs? Does it always halt in polynomial time?* On the other hand, reasoning about probabilistic programming usually involves more *quantitative* questions: *What is the probability that the postcondition is satisfied? What is the probability that this*

[a] From *Foundations of Probabilistic Programming*, edited by Gilles Barthe, Joost-Pieter Katoen and Alexandra Silva published 2020 by Cambridge University Press.

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

program halts? Is its expected halting time polynomial? In order to answer questions like these, the first step should be to develop a formal mathematical semantics for probabilistic programs, which will allow us to formalise such questions precisely. This is the main purpose of this chapter.

Reasoning about probabilistic programs is in general difficult because it compounds the difficulty of deterministic program analysis with questions of probability theory, which can sometimes be counterintuitive. We will use examples to illustrate all the main ideas presented in this chapter. We introduce these examples here and will return to them as we develop the semantics of probabilistic programs. We start with two examples involving *discrete probabilities* for which naive probability theory provides a sufficient framework for reasoning. We will then present two programs that involve *continuous* distributions for which a more general theory known as *measure theory* is needed. The requisite background for understanding these concepts is presented in Section 1.2.

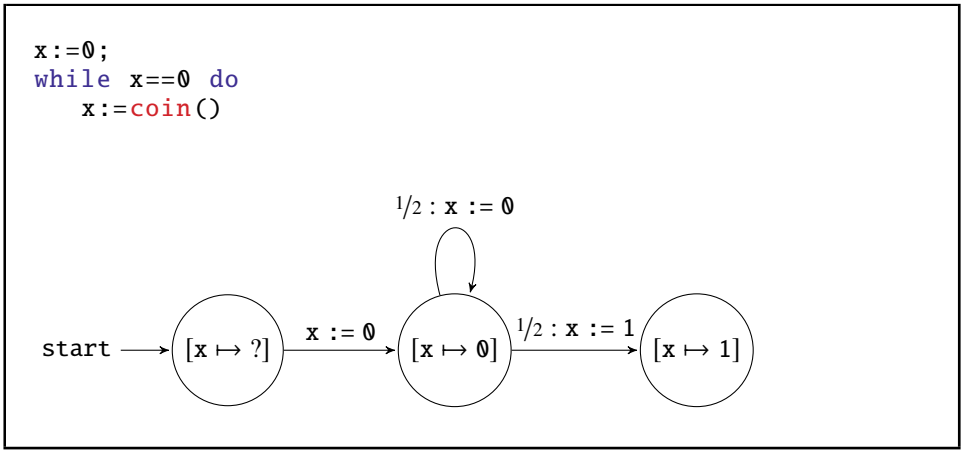

Figure 1.1 A simple coin-toss program

We start with the simple program of Fig. 1.1 displayed next to the small probabilistic automaton it implements. Here the construct `coin()` is our random number generator; each successive call returns 0 or 1, each with probability $1/2$, and successive calls are *independent*, which means that n successive calls will yield one of the 2^n possible sequences of n binary digits, each with probability 2^{-n}. A distribution on $\{0, 1\}$ that takes value 1 with probability p and 0 with probability $1 - p$ is called a *Bernoulli distribution with (success) parameter* p. Thus `coin()` is a Bernoulli distribution with success parameter $1/2$.

It is intuitively clear that this program eventually halts with probability 1. Looking at the automaton of Fig. 1.1, one can see that the probability of the program going

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

through n iterations of the body of the loop is 2^{-n}. Moreover, the expected number of iterations of the body of the loop is given by

$$\sum_{n=1}^{\infty} n2^{-n} = 2.$$

This type of simple probabilistic process involving repeated independent trials until some fixed "success" event occurs is called a *Bernoulli process*. If the probability of success in each trial is p, then the expected time until success is $1/p$. In this example, $p = 1/2$. We will show in Section 1.3 how the mathematical interpretation of this program (its *semantics*) can be constructed *compositionally*, that is to say line-by-line, and how it agrees with these simple observations.

Our second example is also discrete, but intuitively less obvious. The program of Fig. 1.2 implements a random walk on the two-dimensional grid $\mathbb{Z} \times \mathbb{Z}$. In each iteration of the body of the loop, the function `step` updates the current coordinates by moving left, right, down, or up, each with equal probability $1/4$.

```
main{
    u:=0;
    v:=0;
    step(u,v);
    while u!=0 || v!=0 do
        step(u,v)
}

step(u,v){
    x:=coin();
    y:=coin();
    u:=u+(x-y);
    v:=v+(x+y-1)
}
```

Figure 1.2 A random walk on a two-dimensional grid

The loop continues until the random walk returns to the origin. The first call to `step` outside the loop ensures that the program takes at least one step, so it does not halt immediately. The question of the halting probability is now much less obvious. The state space is infinite, and there is no constraint on how far the random walk can travel from the origin. Indeed, for any distance, there is a nonzero probability that it goes at least that far. However, it turns out that the probability that the program halts is 1. In the terminology of probability theory, we would say that the two-dimensional random walk is *recurrent* at every point. This example illustrates how the analysis of probabilistic programs can rely on results from probability theory

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

that are far from obvious. Indeed, the three-dimensional version is not recurrent; the probability that a random walk on $\mathbb{Z}^3$ eventually returns to the origin is strictly less than 1.

We now consider two programs that require *continuous* distributions. The semantics of such programs cannot be defined without the full power of *measure theory*, the mathematical foundation of probabilities and integration. The program of Fig. 1.3 approximates the constant π using *Monte Carlo integration*, a probabilistic integration method. The program works by taking a large number of independent, uniformly distributed random samples from the square $[0,1] \times [0,1]$ and counting the number that fall inside the unit circle. As the area of the square is 1 and the area of the part of the unit circle inside that square is $\pi/4$, by the law of large numbers we expect to see a $\pi/4$ fraction of sample points lying inside the circle.

```
i:=0;
n:=0;
while i<1e9 do
   x:=rand();
   y:=rand();
   if (x*x+y*y) < 1 then n:=n+1;
   i:=i+1
i:=4*n/1e9;
```

Figure 1.3 Probabilistic computation of π.

In this example, the random number generator `rand()` samples from the uniform distribution on the interval $[0,1]$. This distribution is often called *Lebesgue measure*. Here the state space $[0,1]$ is uncountable and the probability of drawing any particular $x \in [0,1]$ is zero. Such probability distributions are called *continuous*. The natural question to ask about this program is not whether it terminates (it clearly does) but whether it returns a good approximation of π with high probability. We will answer this question in Section 1.3.

Finally, the program in Fig. 1.4 generates a real number between $[0,1]$ whose expansion in base 3 does not contain any 1's. This program is not like the others in that it does not halt (nor is it meant to). The program generates a sample from a curious and in many respects counterintuitive distribution called the *Cantor distribution*. It cannot be described using discrete probability distributions (i.e. finite or countable weighted sums of point masses), although the program only uses a discrete fair coin as a source. The Cantor distribution is also an example of *continuous* probability distribution, which assigns probability zero to every element of the state space. It is also an example of a so-called *singular* distribution, since it can be shown that the set of all its possible outcomes—that is to say the set of

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

all real numbers whose base-3 expansion contains no 1's—has measure 0 in the Lebesgue measure on $[0,1]$.

```
x:=0;
d:=1;
while true do
   d:=d/3;
   x:=x+2*coin()*d
```

Figure 1.4 Cantor distribution program.

1.2 Measure theory: What you need to know

Measures are a generalization of the concepts of length, area, or volume of Euclidean geometry to other spaces. They form the basis of probability and integration theory. In this section, we explain what it means for a space to be a *measurable space*, we define *measures* on these spaces, and we examine the rich structure of *spaces of measures*, which will be essential to the semantics of probabilistic programs defined in Section 1.3.5. When not specified otherwise we use the word measure to refer to finite measures.

1.2.1 Some intuition

The concepts of length, area, and volume on Euclidean spaces are examples of *(positive) measures*. These are sufficient to illustrate most of the desired properties of measures and some pitfalls to avoid. For the sake of simplicity, let us examine the concept of *length*. Given an interval $[a,b] \subseteq \mathbb{R}$, its length is of course $\ell([a,b]) = b-a$. But the length function ℓ makes sense for other subsets of $\mathbb{R}$ besides intervals. So we will begin with two related questions:

(a) Which subsets of $\mathbb{R}$ can meaningfully be assigned a "length" consistent with the length of intervals? I.e., what should the domain of ℓ be?
(b) Which properties should the length function ℓ satisfy?

The answer to question (a) will give rise to the notion of *measurable space*, and the answer to question (b) will give rise to the notion of *measure*, both defined formally in Section 1.2.2.

Note that larger intervals have larger lengths: if $[a,b] \subseteq [c,d]$, then we have that $\ell([a,b]) = b-a \leq d-c = \ell([c,d])$. This intuitively obvious property is a general feature of all positive measures: they associate nonnegative real numbers to subsets monotonically with respect to set inclusion. Let us now take two disjoint intervals

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

$[a_1, b_1]$ and $[a_2, b_2]$ with $b_1 < a_2$. It is natural to define the length of $[a_1, b_1] \cup [a_2, b_2]$ as the sum of the length of the respective intervals, i.e.

$$\ell([a_1, b_1] \cup [a_2, b_2]) = \ell([a_1, b_1]) + \ell([a_2, b_2]) = (b_1 - a_1) + (b_2 - a_2).$$

We can draw two conclusions from this natural definition. First, if A, B are two disjoint subsets of $\mathbb{R}$ in the domain of ℓ, then their union should also belong to the domain of ℓ, and the measure of the union should be the sum of the measures. More generally, if A_i, $1 \leq i \leq n$, is any finite collection of pairwise disjoint sets in the domain of ℓ, then $\bigcup_{i=1}^{n} A_i$ should also be in the domain of ℓ, and the measure of the union should be the sum of the measures of the A_i; that is,

$$\ell\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} \ell(A_i). \tag{1.1}$$

A real-valued function on subsets satisfying (1.1) is called (finitely) *additive*. All measures will be finitely additive, and in fact more. Consider the countable collection of pairwise disjoint intervals $[n, n + 2^{-n}), n \in \mathbb{N}$. Generalising (1.1), it is natural to define ℓ on the union of these intervals as

$$\ell\left(\bigcup_{n=0}^{\infty} [n, n + 2^{-n})\right) = \sum_{n=0}^{\infty} 2^{-n} = 2.$$

Again, we can draw two conclusions from this natural definition. First, if A_i for $i \in \mathbb{N}$ is a *countable* collection of pairwise disjoint sets in the domain of ℓ, then $\bigcup_{i \in \mathbb{N}} A_i$ should be in the domain of ℓ; second, that (1.1) should be extended to such countable collections, so that

$$\ell\left(\bigcup_{i=0}^{\infty} A_i\right) = \sum_{i=0}^{\infty} \ell(A_i). \tag{1.2}$$

A function ℓ satisfying (1.2) is called *countably additive* or *σ-additive*. Every measure will be countably additive. The reader will now legitimately ask: what happens if the sum in (1.2) diverges? To deal with this behaviour, one simply allows ∞ as a possible length, that is to say the codomain of ℓ can be the extended real line $\mathbb{R}^+ \cup \{\infty\}$. In particular, this allows us to define the length of $\mathbb{R}$ via (1.2) as:

$$\ell(\mathbb{R}) = \ell\left(\bigcup_{n \in \mathbb{Z}} [n, n + 1)\right) = \infty.$$

However, for the purpose of semantics of probabilistic programs, we will not need measures taking the value ∞. A measure is called *finite* if it only assigns finite values in $\mathbb{R}$ to any set in its domain. *For the remainder of this chapter, the term "measure", otherwise unqualified, will refer to* finite *measures*.

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

Consider now subsets $A \subseteq B$ of $\mathbb{R}$ in the domain of ℓ such that $\ell(A) \leq \ell(B) < \infty$. From finite additivity, it would make sense to define $\ell(B \setminus A) = \ell(B) - \ell(A)$, since $B = A \cup (B \setminus A)$ is a partition of B. In other words, it would also be natural to require that if $A \subseteq B$ and A and B are in the domain of ℓ, then so should be $B \setminus A$, and $\ell(B \setminus A) = \ell(B) - \ell(A)$. Thus the domain of ℓ should be closed under complementation.

The reader may now be wondering: If the domain of ℓ contains all intervals and is closed under countable pairwise disjoint unions and complementation, that is already a very large set of subsets of $\mathbb{R}$. Is it possible that a length can be sensibly assigned to *all* subsets of $\mathbb{R}$? In other words, can we extend ℓ to domain $\mathcal{P}(\mathbb{R})$? Alas, it turns out that this is not possible. An important and desirable property of the length function ℓ is that it is *translation invariant*: given a set A with length $\ell(A)$ (for example an interval), if the entire set A is translated a fixed distance, say d, then its length should be unchanged; that is, $\ell(A) = \ell(\{x + d \mid x \in A\})$. Vitali (1905) constructed a countable set of subsets of the interval $[0, 1)$, called *Vitali sets*, which are pairwise disjoint, translates of each other (modulo 1), and whose union is $[0, 1)$. They would all have to have the same measure, which would break the countable additivity axiom (1.2). Vitali sets are examples of *non-measurable sets*. They provide an example of subsets of $\mathbb{R}$ which are incompatible with the basic assumptions of how the length function should behave. Thus the domain of the length function cannot be $\mathcal{P}(\mathbb{R})$, because it cannot contain the Vitali sets.

The length function ℓ described in the preceding paragraphs is called the *Lebesgue measure* on $\mathbb{R}$. We now turn our attention to axiomatizing the intuitive ideas presented thus far.

1.2.2 Measurable spaces and measures

We start by axiomatizing the closure properties of the domain of a measure (such as the length function) which we have described informally in the previous section.

A *σ-algebra* $\mathcal{B}$ on a set S is a collection of subsets of S containing the empty set $\varnothing$ and closed under complementation in S and countable union (hence also under countable intersection). A pair $(S, \mathcal{B})$, where S is a set and $\mathcal{B}$ is a σ-algebra on S, is called a *measurable space*. The elements of $\mathcal{B}$ are called the *measurable sets* of the space. In a probabilistic setting, elements of S and $\mathcal{B}$ are often called *outcomes* and *events*, respectively. The domain of a measure, for example the length function, will always be a σ-algebra. If the σ-algebra is obvious from the context, we simply say that S is a measurable space. The set of all subsets $\mathcal{P}(S)$ is a σ-algebra called the *discrete σ-algebra*, but as noted above, it may not be an appropriate choice since it may not allow the definition of certain measures. However, it is always an

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

acceptable choice for finite or countable sets, and we will always assume that finite and countable sets are equipped with the discrete σ-algebra.

If $\mathcal{F}$ is a collection of subsets of a set S, we define $\sigma(\mathcal{F})$, the σ-algebra *generated* by $\mathcal{F}$, to be the smallest σ-algebra containing $\mathcal{F}$. That is, $\sigma(\mathcal{F})$ is the smallest collection of subsets of S containing $\mathcal{F}$ and $\varnothing$ and closed under countable union and complement. Equivalently,

$$\sigma(\mathcal{F}) \triangleq \bigcap\{\mathcal{A} \mid \mathcal{F} \subseteq \mathcal{A} \text{ and } \mathcal{A} \text{ is a } \sigma\text{-algebra}\}.$$

Note that $\sigma(\mathcal{F})$ is well-defined, since the intersection is nonempty, as $\mathcal{F} \subseteq \mathcal{P}(S)$ and $\mathcal{P}(S)$ is a σ-algebra. If $(S, \mathcal{B})$ is a measurable space and $\mathcal{B} = \sigma(\mathcal{F})$, we say that the space is *generated* by $\mathcal{F}$.

Measurable functions. Let $(S, \mathcal{B}_S)$ and $(T, \mathcal{B}_T)$ be measurable spaces. A function $f\colon S \to T$ is *measurable* if the inverse image $f^{-1}(B) = \{x \in S \mid f(x) \in B\}$ of every measurable subset $B \in \mathcal{B}_T$ is a measurable subset of S. When $\mathcal{B}_T$ is generated by $\mathcal{F}$, then f is measurable if and only if $f^{-1}(B)$ is measurable for every $B \in \mathcal{F}$.

An example of a measurable function is $\chi_B\colon S \to \{0,1\}$, the *characteristic function* of a measurable set B:

$$\chi_B(s) = \begin{cases} 1, & s \in B, \\ 0, & s \notin B. \end{cases}$$

Here, $(S, \mathcal{B})$ is a measurable space, $B \in \mathcal{B}$, and $\{0,1\}$ is the discrete space.

Measures. A *signed (finite) measure* on $(S, \mathcal{B})$ is a countably additive map $\mu\colon \mathcal{B} \to \mathbb{R}$ such that $\mu(\varnothing) = 0$. Recall that *countably additive* means that if $\mathcal{A}$ is a countable set of pairwise disjoint events, then $\mu(\bigcup \mathcal{A}) = \sum_{A \in \mathcal{A}} \mu(A)$. Equivalently, if $A_0, A_1, A_2, \ldots$ is a countable chain of events (a countable collection of measurable sets such that $A_n \subseteq A_{n+1}$ for all $n \geq 0$), then $\lim_n \mu(A_n)$ exists and is equal to $\mu(\bigcup_n A_n)$.

A signed measure on $(S, \mathcal{B})$ is called *positive* if $\mu(A) \geq 0$ for all $A \in \mathcal{B}$. A positive measure on $(S, \mathcal{B})$ is called a *probability measure* if $\mu(S) = 1$ and a *subprobability measure* if $\mu(S) \leq 1$. A measurable set B such that $\mu(B) = 0$ is called a *μ-nullset*, or simply a *nullset* if there is no ambiguity. A property is said to hold *μ-almost surely* (μ-a.s.) or *μ-almost everywhere* (μ-a.e.) if the set of points on which it does *not* hold is contained in a nullset.

In probability theory, measures are sometimes called *distributions*. We will use the terms *measure* and *distribution* synonymously.

For $s \in S$, the *Dirac measure*, or *Dirac delta*, or *point mass* on s is the probability

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

measure

$$\delta_s(B) = \begin{cases} 1, & s \in B, \\ 0, & s \notin B. \end{cases}$$

A measure is *discrete* if it is a countable weighted sum of Dirac measures. In particular a convex sum of Dirac measures is a discrete probability measure. These are finite or countable sums of the form $\sum_{s \in C} a_s \delta_s$, where all $a_s \geq 0$ and $\sum_{s \in C} a_s = 1$.

A measure μ on a measurable set $(S, \mathcal{B})$ is called *continuous* if $\mu(\{s\}) = 0$ for all singleton sets $\{s\}$ in $\mathcal{B}$. The Lebesgue measures on $\mathbb{R}^n$ for $n \in \mathbb{N}$, that is, the lengths, areas, volumes, etc., are the best known examples of continuous measures.

Pushforward measure. Given $f\colon (S, \mathcal{B}_S) \to (T, \mathcal{B}_T)$ measurable and a measure μ on $\mathcal{B}_S$, one defines the *pushforward measure* $f_*(\mu)$ on $\mathcal{B}_T$ by

$$f_*(\mu)(B) = \mu(f^{-1}(B)), \quad B \in \mathcal{B}_T. \tag{1.3}$$

This measure is well defined: since f is measurable, f^{-1} maps measurable sets of $\mathcal{B}_T$ to measurable sets of $\mathcal{B}_S$.

Lebesgue integration. An important operation on measures and measurable functions is *Lebesgue integration*. Let $(S, \mathcal{B})$ be a measurable space. Given a measure $\mu\colon \mathcal{B} \to \mathbb{R}$ and bounded measurable function $f\colon S \to \mathbb{R}$, say bounded above by M and below by m, the *Lebesgue integral* of f with respect to μ, denoted $\int f \, d\mu$, is a real number obtained as the limit of finite weighted sums of the form

$$\sum_{i=0}^{n} f(s_i)\mu(B_i), \tag{1.4}$$

where $B_0, \ldots, B_n$ is a measurable partition of S, the value of f does not vary more than $(M-m)/n$ in any B_i, and $s_i \in B_i$, $1 \leq i \leq n$. The limit is taken over increasingly finer measurable partitions of the space. For the details of this construction, see for example (Dudley, 2002, Ch. 4) or (Aliprantis and Border, 1999, Ch. 11).

For a finite discrete space $n = \{1, 2, \ldots, n\}$, the integral reduces simply to a weighted sum: $\int f \, d\mu = \sum_{i=1}^{n} f(i)\mu(i)$.

The *bounded integral* $\int_B f \, d\mu$, where $B \in \mathcal{B}$, is obtained by integrating over the set B instead of all of S; equivalently,

$$\int_B f \, d\mu \triangleq \int \chi_B \cdot f \, d\mu, \tag{1.5}$$

where χ_B is the characteristic function of B and $\chi_B \cdot f$ is the pointwise product of real-valued functions.

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

Absolute continuity. Given two measures μ, ν, we say that μ is *absolutely continuous* with respect to ν and write $\mu \ll \nu$ if for all measurable sets B, if $\nu(B) = 0$, then $\mu(B) = 0$. Informally, if ν assigns no mass to B, then neither does μ. Although we will not need it, we cannot fail to mention the following theorem, which is one the pillars of probability theory.

Theorem 1.1 (Radon–Nikodym) *Let μ, ν be two finite measures on a measurable space $(S, \mathcal{B})$ and assume that μ is absolutely continuous with respect to ν. Then there exists a measurable function $f : S \to \mathbb{R}$ defined uniquely up to a μ-nullset such that*

$$\mu(B) = \int_B f \, d\nu.$$

The function f is called the Radon–Nikodym derivative *of μ with respect to ν.*

Radon–Nikodym derivatives are known in probability theory as *probability density functions*. For example, the standard Gaussian probability measure is absolutely continuous with respect to Lebesgue measure (the length function) on $\mathbb{R}$. Its Radon–Nikodym derivative with respect to Lebesgue measure is the Gaussian density function $f(t) = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}$.

Products. Given two measurable spaces $(S_1, \mathcal{B}_1)$ and $(S_2, \mathcal{B}_2)$, one can construct the *product space* $(S_1 \times S_2, \mathcal{B}_1 \otimes \mathcal{B}_2)$, where $S_1 \times S_2$ is the cartesian product and $\mathcal{B}_1 \otimes \mathcal{B}_2$ is the σ-algebra on $S_1 \times S_2$ generated by all *measurable rectangles* $B_1 \times B_2$ for $B_1 \in \mathcal{B}_1$ and $B_2 \in \mathcal{B}_2$. In other words,

$$\mathcal{B}_1 \otimes \mathcal{B}_2 \triangleq \sigma\left(\{B_1 \times B_2 \mid B_1 \in \mathcal{B}_1, B_2 \in \mathcal{B}_2\}\right). \tag{1.6}$$

The measurable rectangles $B_1 \times B_2$ are a generalisation of the case where $S_1 = S_2 = \mathbb{R}$ and B_1, B_2 are intervals. The product of two measurable spaces is thus the measurable space generated by the corresponding measurable rectangles.

A measure on the product space $(S_1 \times S_2, \mathcal{B}_1 \otimes \mathcal{B}_2)$ is sometimes called a *joint distribution*. Due to the inductive construction (1.6) of $\mathcal{B}_1 \otimes \mathcal{B}_2$ from measurable rectangles $B_1 \times B_2$, joint distributions are uniquely determined by their values on measurable rectangles. For details of this extension, see (Dudley, 2002, §4.4).

A special class of joint distributions are the *product measures* $\mu_1 \otimes \mu_2$ formed from a measure μ_1 on $(S_1, \mathcal{B}_1)$ and a measure μ_2 on $(S_2, \mathcal{B}_2)$, defined on measurable rectangles by

$$(\mu_1 \otimes \mu_2)(B_1 \times B_2) \triangleq \mu_1(B_1)\mu_2(B_2).$$

As mentioned, this extends uniquely to a joint distribution $\mu_1 \otimes \mu_2 : \mathcal{B}_1 \otimes \mathcal{B}_2 \to \mathbb{R}$. Product measures capture the idea of *independence*: sampling $\mu_1 \otimes \mu_2$ to obtain an element of $S_1 \times S_2$ is equivalent to independently sampling μ_1 on S_1 and μ_2 on S_2.

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

Markov Kernels. Let $(S, \mathcal{B}_S)$ and $(T, \mathcal{B}_T)$ be measurable spaces. A function $P\colon S \times \mathcal{B}_T \to \mathbb{R}$ is called a *Markov kernel* (also called a Markov transition, measurable kernel, stochastic kernel, or stochastic relation) if

- for fixed $A \in \mathcal{B}_T$, the map $\lambda s.P(s, A)\colon S \to \mathbb{R}$ is a measurable function on $(S, \mathcal{B}_S)$; and
- for fixed $s \in S$, the map $\lambda A.P(s, A)\colon \mathcal{B}_T \to \mathbb{R}$ is a probability measure on $(T, \mathcal{B}_T)$.

These properties allow integration on the left and right, respectively.

The measurable spaces and Markov kernels form a category, the *Kleisli category of the Giry monad*; see Panangaden (1998, 2009); Doberkat (2007); Giry (1982). In this context, we occasionally write $P\colon (S, \mathcal{B}_S) \to (T, \mathcal{B}_T)$ or just $P\colon S \to T$. Composition is given by integration: for $P\colon S \to T$ and $Q\colon T \to U$,

$$(P \mathbin{;} Q)(s, A) = \int_{t \in T} P(s, dt) \cdot Q(t, A). \tag{1.7}$$

Associativity of composition follows essentially from Fubini's theorem (see Chung, 1974, or Halmos, 1950). Markov kernels were introduced in Lawvere (1962) and were proposed as a model of probabilistic while programs in Kozen (1985).

The definition of the pushforward of a measure can be extended to Markov kernels as follows. Given a measure μ on $\mathcal{B}_S$, its *pushforward* under the Markov kernel $P\colon (S, \mathcal{B}_S) \to (T, \mathcal{B}_T)$ is the measure $P_*(\mu)$ on $\mathcal{B}_T$ defined by

$$P_*(\mu)(B) = \int_{s \in S} P(s, B)\ \mu(ds). \tag{1.8}$$

Any measurable map $f\colon (S, \mathcal{B}_S) \to (T, \mathcal{B}_T)$ determines a trivial Markov kernel $s \mapsto \delta_{f(s)}$, and under this definition the pushforward operation defined in Eq. (1.3) is just a special case of Eq. (1.8).

The reader will note that we changed notation in displaying the Lesbegue integrals in Equations (1.7) and (1.8) when compared to Eq. (1.5). This is standard notation in measure theory, in particular in the presence of Markov kernels, but for clarity we note that these could have been written as:

$$(P \mathbin{;} Q)(s, A) = \int_T Q(-, A)\ dP(s, -)$$
$$P_*(\mu)(B) = \int_S P(-, B)\ d\mu.$$

Recall that because P and Q are Markov kernels the functions $Q(-, A)$ and $P(-, B)$ are measurable and $P(s, -)$ is a probability measure.

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

1.2.3 Spaces of measures

The set of all (finite, signed) measures on a measurable set $(S, \mathcal{B})$ will be denoted $\mathcal{M}(S, \mathcal{B})$, or simply $\mathcal{M}S$ if $\mathcal{B}$ is understood. The spaces $\mathcal{M}S$ carry a very rich structure which lies at the heart of the denotational semantics described in Section 1.3.5. We now describe this structure.

Vector space structure. First, $\mathcal{M}S$ is always a *real vector space* whose addition operation and scalar multiplication are defined pointwise:

$$(\mu + \nu)(B) \triangleq \mu(B) + \nu(B) \qquad\qquad (a\mu)(B) \triangleq a\mu(B)$$

for $B \in \mathcal{B}$, $\mu, \nu \in \mathcal{M}S$, and $a \in \mathbb{R}$. It is easily argued that $\mu + \nu$ and $a\mu$ defined in this way are measures whenever μ, ν are.

The set of measures on a finite set $n = \{0, 1, \ldots, n-1\}$ is isomorphic as a real vector space to $\mathbb{R}^n$: the mass $\mu(i)$ of the element i corresponds to the i^{th} coordinate of a vector in $\mathbb{R}^n$ in the standard basis. This perspective is well established in the theory of Markov chains, where initial or stationary distributions are represented as row or column vectors depending on the convention (see e.g. Norris (1997)).

Normed space structure. The second important structure carried by $\mathcal{M}S$ is its *norm*. Combined with the vector space structure, this makes $\mathcal{M}S$ a *Banach space*. The norm $\|\mu\|$ of a measure μ is called the *total variation norm* and defined by

$$\|\mu\| \triangleq \sup \left\{ \sum_{i=1}^{n} |\mu(B_i)| : \{B_1, \ldots, B_n\} \text{ is a finite measurable partition of } S \right\}. \quad (1.9)$$

For a positive measure μ, the norm is always $\|\mu\| = \mu(S)$, and for a probability measure, $\|\mu\| = 1$. In other words, probability measures lie on the boundary of the unit ball of the space of measures. However, a general (signed) measure can assign positive mass to some regions of S and negative mass to others, hence the idea of partitioning the space and the presence of the absolute value in (1.9).

The total variation norm interacts with the vector space structure to make $\mathcal{M}S$ a normed vector space: $\|x\| \geq 0$, $\|x\| = 0$ iff $x = 0$, $\|a\mu\| = |a|\,\|\mu\|$, and $\|\mu + \nu\| \leq \|\mu\| + \|\nu\|$. Moreover, the normed vector space $\mathcal{M}S$ is *complete*, which means that all Cauchy sequences of measures converge to a limit in $\mathcal{M}S$. A complete normed vector space is called a *Banach space*.

In the case of a finite set n, using the vector space representation of $\mathcal{M}n$ described above, the norm of a finite measure $\mu \in \mathcal{M}n$ is simply the usual ℓ_1-norm, sometimes also called *Manhattan* or *taxicab norm* $\|\mu\| = \sum_{i=1}^{n} |\mu(i)|$. A probability measure on n is thus always a vector in $\mathbb{R}^n$ of ℓ_1-norm 1.

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

Order structure. The final piece of structure is a *partial order*. Measures have a natural pointwise order: $\mu \leq \nu$ if $\mu(B) \leq \nu(B)$ for every $B \in \mathcal{B}$.

Any pair of distinct probability measures are incomparable in this order, as $\mu(B) < \nu(B)$ for a some $B \in \mathcal{B}$ implies $\nu(B^c) < \mu(B^c)$, where B^c is the complement of B, since $\mu(B) + \mu(B^c) = 1 = \nu(B) + \nu(B^c)$. A measure μ is positive if $0 \leq \mu$. The set of all positive measures is called the *positive cone* of $\mathcal{M}S$ and denoted $(\mathcal{M}S)^+$. Probability measures are of course positive measures, so the probability measures in $\mathcal{M}S$ are precisely the positive measures of norm 1, and the subprobability measures are the positive measures of norm at most 1. In other words, the subprobability measures comprise the positive orthant of the unit ball of $\mathcal{M}S$.

The partial order is compatible with the vector space structure in the sense that

- if $\mu \leq \nu$, then $\mu + \rho \leq \nu + \rho$; and
- if $0 \leq a \in \mathbb{R}$ and $\mu \leq \nu$, then $a\mu \leq a\nu$.

We say that the operations of addition and multiplication by a positive scalar are *monotone*. Moreover, the partial order in fact defines a *lattice structure*, that is to say every pair of measures μ, ν have a least upper bound and a greatest lower bound with respect to the partial order, defined explicitly by

$$(\mu \vee \nu)(B) \triangleq \sup \{\mu(A \cap B) + \nu(A^c \cap B) \mid A \in \mathcal{B}\}$$
$$(\mu \wedge \nu)(B) \triangleq \inf \{\mu(A \cap B) + \nu(A^c \cap B) \mid A \in \mathcal{B}\}.$$

In particular, the *positive part* of a measure μ can be defined as $\mu^+ = \mu \vee 0$, and its negative part as $\mu^- = (-\mu) \vee 0$. Note that both μ^+ and μ^- are positive measures, and in fact every measure can be decomposed as the difference of two positive measures, since $\mu = \mu^+ - \mu^-$. Moreover, there is a measurable set C such that $\mu^+ = \mu_C$ and $\mu^- = -\mu_{C^c}$, where μ_C is the measure $\mu_C(A) = \mu(A \cap C)$; equivalently, $\mu(B) \geq 0$ for all measurable sets $B \subseteq C$ and $\mu(B) \leq 0$ for all measurable sets $B \subseteq C^c$. The set C is *essentially unique* in the sense that if C' is any other measurable set satisfying these properties, then all measurable subsets of the symmetric difference $C \triangle C'$ are μ-nullsets. This is known as the *Hahn-Jordan decomposition theorem*; see e.g. (Dudley, 2002, Th. 5.6.1). The sum of the positive and negative part is called the *modulus* of μ and is denoted $|\mu| = \mu^+ + \mu^-$.

The order is compatible with the norm in the sense that $|\mu| \leq |\nu|$ implies $\|\mu\| \leq \|\nu\|$. A Banach space with a lattice structure that is compatible with both the linear and normed structures in the sense detailed above is called a *Banach lattice*. Thus $\mathcal{M}S$ is always a Banach lattice.

In order to interpret `while` loops in Section 1.3.5, we will need one last order-theoretic concept. A Banach lattice is said to be *σ-order-complete* or *σ-Dedekind-complete* if every countable order-bounded set of measures in $\mathcal{M}S$ has a supremum

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

in $\mathcal{M}S$. From the perspective of theoretical computer science, this notion is similar to the completeness property for ω-complete partial orders (ω-CPOs) in domain theory, the key difference being that only sets of elements with a common upper bound are considered. In particular, the set $\mathcal{M}1 \cong \mathbb{R}$ is σ-order complete but is not an ω-CPO, because this would require adding a point at infinity, thereby losing the vector space structure. In fact, every space of measures $\mathcal{M}S$ is σ-order complete.

In the case of a finite set $\{1, \ldots, n\}$, the order is just the usual pointwise order on $\mathbb{R}^n$: $(x_1, \ldots, x_n) \leq (y_1, \ldots, y_n)$ if $x_i \leq y_i$, $1 \leq i \leq n$, and the lattice structure on $\mathcal{M}n \cong \mathbb{R}^n$ simplifies to $(\mu \vee \nu)(\{i\}) = \max\{\mu(\{i\}), \nu(\{i\})\}$ and $(\mu \wedge \nu)(\{i\}) = \min\{\mu(\{i\}), \nu(\{i\})\}$.

Operators. One of the advantages of working with spaces of measures is that, since they are vector spaces, we can do linear algebra on them. In the case of measure spaces over finite sets, i.e. spaces of the form $\mathbb{R}^n$, this means the usual matrix-based linear algebra. In the general case, we have the infinite-dimensional generalisation in terms of *linear operators*.

A *linear operator* (or simply an *operator*) $T \colon V \to W$ between two vector spaces over the reals is a map satisfying $T(x + y) = T(x) + T(y)$ and $T(ax) = aT(x)$ for $x, y \in V$ and $a \in \mathbb{R}$. In the finite-dimensional case, if we choose bases $(v_1, \ldots, v_m)$ and $(w_1, \ldots, w_n)$ for V and W respectively, T can be represented as an $n \times m$ matrix $\mathbf{T}$ whose ij^{th} component $\mathbf{T}_{ij}$ is the j^{th} coordinate of $T(v_i)$ in the basis of W.

We will be mostly interested in operators that send probability measures to subprobability measures. Recall that probability measures are precisely the positive measures of norm 1. A *positive operator* between Banach lattices is an operator that sends positive vectors to positive vectors, i.e. $Tv \geq 0$ whenever $v \geq 0$. A *stochastic operator* is a positive operator preserving the norm of positive vectors, i.e. such that $\|Tv\| = \|v\|$ whenever $v \geq 0$. Similarly, an operator sending probabilities to subprobabilities is characterised as a positive operator contracting the norm of positive vectors, i.e. such that $\|Tv\| \leq \|v\|$ whenever $v \geq 0$.

In the case of measures over finite spaces $\mathcal{M}n \cong \mathbb{R}^n$, we can represent stochastic operators as *(right) stochastic matrices*, matrices with nonnegative entries whose rows each sum to 1.[1] Indeed, any operator A sending probability measures $p = (p_1, \ldots, p_n)$ to probability measures must be stochastic, as the k^{th} row of A is the result of applying A to the Dirac delta δ_k:

$$(\delta_k A)_j = \sum_{i=1}^{n} \delta_k(i) A_{ij} = A_{kj}.$$

[1] We follow the convention adopted in the literature on Markov chains, where measures are represented as row vectors and operators are applied from the right (hence "right stochastic").

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

1.3 Semantics of a simple imperative probabilistic language

Now we are ready to give the formal semantics of a simple imperative programming language with two types of probabilistic operations: a function sampling from a Bernoulli distribution with parameter $p = 1/2$, and a function sampling from the uniform distribution on the interval $[0, 1]$. This simple language will cover the examples in Figs. 1.1, 1.2, 1.3, and 1.4.

1.3.1 Syntax

We start by defining the syntax of the language.
(i) Deterministic terms:

$d ::= a$	$a \in \mathbb{R}$, constants
$\mid x$	$x \in$ Var, a countable set of variables
$\mid d$ `op` d	`op` $\in \{+, -, *, \div\}$

(ii) Terms:

$t ::= d$	d a deterministic term
$\mid$ `coin()` $\mid$ `rand()`	sample in $\{0, 1\}$ and $[0, 1]$, respectively
$\mid t$ `op` t	`op` $\in \{+, -, *, \div\}$

(iii) Tests:

$b ::=$ `true` $\mid$ `false`	
$\mid d == d \mid d < d \mid d > d$	comparison of deterministic terms
$\mid b$ `&&` $b \mid b$ `||` $b \mid$ `!`b	Boolean combinations of tests

(iv) Programs:

$e ::=$ `skip`	
$\mid x$ `:=` t	assignment
$\mid e$ `;` e	sequential composition
$\mid$ `if` b `then` e `else` e	conditional
$\mid$ `while` b `do` e	`while` loop

Remark 1.2 We disallow probabilistic terms in tests for simplicity in the presentation. This restriction is without loss of generality, as one could consider such expressions as syntactic sugar; the probabilistic term can always be removed using auxiliary variables. For example, the right-hand program below is the de-sugared

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

version of the left-hand program using a fresh auxiliary variable x:

$$\texttt{if coin() == 1 then } e_1 \texttt{ else } e_2 \quad x := \texttt{coin() ; if } x == 1 \texttt{ then } e_1 \texttt{ else } e_2$$

1.3.2 Operational versus denotational semantics

As we outlined in the introduction, the main purpose of this chapter is to define a formal mathematical interpretation—a *semantics*—for probabilistic programs. Having a semantics provides the tools necessary to reason about the properties of programs, as we will show with a few concrete examples. However, the focus of this chapter will primarily be on defining the semantics itself.

We will present two classical types of semantics: *operational* and *denotational*. The purpose of operational semantics is to model the step-by-step executions of the program on a machine. It will model the evolution of the state of the machine described by its *memory-state*, the values assigned to each variable and to each random number generator in a program, together with a stack of instructions. On the other hand, the purpose of denotational semantics is to model the intended mathematical meaning of a probabilistic program in terms of probability distributions.

1.3.3 Operational semantics of probabilistic programs

The random number generator `coin()` is represented by an independent and identically distributed (i.i.d.) sequence of random variables distributed according to the Bernoulli distribution on $\{0,1\}$ with parameter $p = 1/2$. Similarly, `rand()` is represented by an i.i.d. sequence distributed according to the uniform distribution on $[0,1]$. When a program runs, we fix at the beginning two infinite streams $m_0 m_1 m_2 \cdots$ and $p_0 p_1 p_2 \cdots$, where each $m_i \in \{0,1\}$ is an independent sample from the Bernoulli distribution with $p = 1/2$ and each $p_i \in [0,1]$ is an independent sample from the uniform distribution on $[0,1]$. Intuitively, these sequences represent infinite stacks of random numbers that are available to the program. Each time the function `coin()` or `rand()` is called, the next random number is popped from the stack. We use the auxiliary head and tail functions $\mathsf{hd}(m_0 m_1 m_2 \cdots) = m_0$ and $\mathsf{tl}(m_0 m_1 m_2 \cdots) = m_1 m_2 m_3 \cdots$ to implement this. This deterministic behaviour reflects the behaviour of pseudo-random number generators which, given a seed, deterministically generate such sequences (or to be precise, seemingly random cyclical sequences with very long periods).

Given a program containing n variables $\{x_1, \ldots, x_n\}$, as described in Section 1.3.1, a *memory-state* is modelled as a triple (s, m, p) consisting of a store $s : n \to \mathbb{R}$ and a pair of infinite streams $m \in \{0,1\}^\omega$ and $p \in [0,1]^\omega$ representing the current streams of available random digits. A machine-state is a 4-tuple (e, s, m, p), where e corresponds to a stack of instructions and (s, m, p) is a memory-state.

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

We can now define the operational semantics of our language. The operational semantics is defined in terms of a single-step *reduction relation* $(e, s, m, p) \longrightarrow (e', s', m', p')$ between machine-states. The reduction relation describes the step-by-step evaluation of a program and its corresponding effect on the state of the machine. The reflexive transitive closure of this relation will be denoted $\overset{*}{\longrightarrow}$ and will be used to talk about multi-step transitions.

In order to define the reduction relation, we first define the semantics of terms. Each term t will be interpreted as a function

$$[\![t]\!] : \mathbb{R}^n \times \mathbb{N}^\omega \times \mathbb{N}^\omega \to \mathbb{R} \times \mathbb{N}^\omega \times \mathbb{N}^\omega.$$

Intuitively, the value of $[\![t]\!]$ on (s, m, p) is a triple (a, m', p'), where a is the real value of the term and m' and p' are the new stacks of random numbers. Formally, $[\![t]\!]$ is defined inductively:

$$\begin{aligned}
[\![r]\!] &: (s, m, p) \mapsto (r, m, p) \\
[\![x_i]\!] &: (s, m, p) \mapsto (s(i), m, p) \\
[\![\texttt{coin()}]\!] &: (s, m, p) \mapsto (\mathsf{hd}\, m, \mathsf{tl}\, m, p) \\
[\![\texttt{rand()}]\!] &: (s, m, p) \mapsto (\mathsf{hd}\, p, m, \mathsf{tl}\, p) \\
[\![t_1 \ \mathsf{op} \ t_2]\!] &: (s, m, p) \mapsto \begin{array}{l} \texttt{let}\ (a_1, m', p') = [\![t_1]\!](s, m, p)\ \texttt{in} \\ \texttt{let}\ (a_2, m'', p'') = [\![t_2]\!](s, m', p')\ \texttt{in} \\ (a_1\ \mathsf{op}\ a_2, m'', p'') \end{array}
\end{aligned}$$

where $\mathsf{op} \in \{+, -, *, \div\}$. In other words, we evaluate t_1 first, which produces a value a_1 but might consume some values from m and p, popping those stacks to leave m' and p'; these new stacks are then used in the evaluation of t_2, which in turn changes them to m'' and p'' and yields a value a_2. The values a_1 and a_2 are combined with the appropriate arithmetic operation, and that is returned with the stacks m'' and p''.

There is already an interesting question to consider. Note that we are evaluating $t_1\ \mathsf{op}\ t_2$ from left to right, i.e. starting with t_1. This is an arbitrary decision. Would evaluation from right to left be equally valid? It clearly would not always give the same value; e.g., the program `rand()` $\div$ `rand()` evaluated on $(s, m, (.2, .5, p))$ from left to right would give $(.4, m, p)$, but evaluated from right to left would give $(2.5, m, p)$. However, perhaps surprisingly, the two methods of evaluation are *probabilistically* equivalent, which means they give the same values with equal probability. The denotational semantics to be given below will allow us to reason more easily about such facts.

We now define the semantics of tests. Each test b will be interpreted as a function

$$[\![b]\!] : \mathbb{R}^n \times \mathbb{N}^\omega \times \mathbb{N}^\omega \to \{\texttt{true}, \texttt{false}\}$$

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

defined inductively by

$$[\![t_1 == t_2]\!] : (s,m,p) \mapsto \begin{cases} \texttt{true} & \text{if } [\![t_1]\!](s,m,p) = [\![t_2]\!](s,m,p) \\ \texttt{false} & \text{otherwise} \end{cases}$$

$$[\![t_1 < t_2]\!] : (s,m,p) \mapsto \begin{cases} \texttt{true} & \text{if } [\![t_1]\!](s,m,p) < [\![t_2]\!](s,m,p) \\ \texttt{false} & \text{otherwise} \end{cases}$$

$$[\![t_1 > t_2]\!] : (s,m,p) \mapsto \begin{cases} \texttt{true} & \text{if } [\![t_1]\!](s,m,p) > [\![t_2]\!](s,m,p) \\ \texttt{false} & \text{otherwise} \end{cases}$$

$$[\![b_1 \texttt{ \&\& } b_2]\!] : (s,m,p) \mapsto [\![b_1]\!](s,m,p) \wedge [\![b_2]\!](s,m,p)$$

$$[\![b_1 \texttt{ || } b_2]\!] : (s,m,p) \mapsto [\![b_1]\!](s,m,p) \vee [\![b_2]\!](s,m,p)$$

$$[\![\texttt{!}\, b]\!] : (s,m,p) \mapsto \neg [\![b]\!](s,m,p)$$

where $\wedge, \vee$ and $\neg$ are the usual Boolean operations on $\{\texttt{true}, \texttt{false}\}$. Note that in the definition of the three base cases above, m, p are arguments of both $[\![t_1]\!]$ *and* $[\![t_2]\!]$, unlike in the semantics of terms. This is because we only allow deterministic terms in tests, thus $[\![t_1]\!]$ does not consume any random numbers, leaving the stacks m, p unchanged for the evaluation of $[\![t_2]\!]$. For the same reason, it is unnecessary to include m, p in the output of $[\![b]\!]$.

We can now define the reduction relation $\longrightarrow$. The relation is given by the rules gathered in Table 1.1. We use the traditional notation $s[i \mapsto l]$ to denote the n-tuple defined exactly as s apart from at position i where value l is used instead. We will say that the execution of a program e *terminates* from the memory-state (s,m,p) if there exists a memory-state (s',m',p') such that

$$(e,s,m,p) \xrightarrow{*} (\texttt{skip}, s', m', p').$$

We will say that a program e *diverges* from a state (s,m,p) if it does not terminate.

1.3.4 Operational semantics through examples

To illustrate how the system described above works concretely, let us examine the operational semantics of the programs described in Section 1.1.

Example 1: A simple Markov chain

We start with the very simple program displayed in Fig. 1.1. Using the rule for assignments, the rule for sequential composition and the definition of the reflexive transitive closure $\xrightarrow{*}$ we get the following derivation, where for notational convenience we write e for `while` x == 0 `do` x `:=` `coin()`:

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

Assignment:

$$\frac{[\![t]\!](s,m,p) = (a,m',p')}{(x_i := t, s, m, p) \longrightarrow (\texttt{skip}, s[i \mapsto a], m', p')}$$

Sequential composition:

$$\frac{(e_1,s,m,p) \longrightarrow (e_1',s',m',p')}{(e_1 \texttt{ ; } e_2,s,m,p) \longrightarrow (e_1' \texttt{ ; } e_2,s',m',p')} \qquad \frac{}{(\texttt{skip ; } e,s,m,p) \longrightarrow (e,s,m,p)}$$

Conditional:

$$\frac{[\![b]\!](s,m,p) = \texttt{true}}{(\texttt{if } b \texttt{ then } e_1 \texttt{ else } e_2,s,m,p) \longrightarrow (e_1,s,m,p)}$$

$$\frac{[\![b]\!](s,m,p) = \texttt{false}}{(\texttt{if } b \texttt{ then } e_1 \texttt{ else } e_2,s,m,p) \longrightarrow (e_2,s,m,p)}$$

`while` loops:

$$\frac{}{(\texttt{while } b \texttt{ do } e,s,m,p) \longrightarrow (\texttt{if } b \texttt{ then } (e \texttt{ ; while } b \texttt{ do } e) \texttt{ else skip},s,m,p)}$$

Reflexive-transitive closure:

$$\frac{}{(e,s,m,p) \xrightarrow{*} (e,s,m,p)} \qquad \frac{(e_1,s_1,m_1,p_1) \longrightarrow (e_2,s_2,m_2,p_2)}{(e_1,s_1,m_1,p_1) \xrightarrow{*} (e_2,s_2,m_2,p_2)}$$

$$\frac{(e_1,s_1,m_1,p_1) \xrightarrow{*} (e_2,s_2,m_2,p_2) \qquad (e_2,s_2,m_2,p_2) \xrightarrow{*} (e_3,s_3,m_3,p_3)}{(e_1,s_1,m_1,p_1) \xrightarrow{*} (e_3,s_3,m_3,p_3)}$$

Table 1.1 *Rules of the operational semantics*

$$\frac{\dfrac{(x := 0,s,m,p) \longrightarrow (\texttt{skip},s[x \mapsto 0],m,p)}{(x := 0 \texttt{ ; } e,s,m,p) \longrightarrow (\texttt{skip ; } e,s[x \mapsto 0],m,p)} \qquad \dfrac{}{(\texttt{skip ; } e,s[x \mapsto 0],m,p) \longrightarrow (e,s[x \mapsto 0],m,p)}}{(x := 0 \texttt{ ; } e,s,m,p) \xrightarrow{*} (e,s[x \mapsto 0],m,p)}$$

We now turn our attention to $(e, s[x \mapsto 0], m, p)$. Using the rule for while loops and conditionals we get (using the same definition of e as above)

$$(e,s[x \mapsto 0],m,p) \xrightarrow{*} (x := \texttt{coin()} \texttt{ ; } e,s[x \mapsto 0],m,p) \tag{1.10}$$

since $[\![x == 0]\!](s[x \mapsto 0],m,p) = \texttt{true}$. We now encounter our first probabilistic behaviour:

$$(x := \texttt{coin()} \texttt{ ; } e,s[x \mapsto 0],m,p) \xrightarrow{*} (e,[s \mapsto \mathsf{hd}\, m],\mathsf{tl}\, m,p).$$

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

If $\mathsf{hd}\, m = 0$, then s does not change, and we are back at the left-hand state of (1.10), but with $\mathsf{tl}\, m$ instead of m. This same sequence of steps continues until encountering a suffix of m of the form $1m'$, at which time, combining the rule for while loops and conditionals, we get

$$(e, s[x \mapsto 1], m', p) \xrightarrow{*} (\mathtt{skip}, s[x \mapsto 1], m', p)$$

since $[\![x == 0]\!](s[x \mapsto 1], m', p) = \mathtt{false}$. If no such suffix $1m'$ exists, then (1.10) continues forever. We can conclude that

$$(x \mathtt{ := } 0 \mathtt{ ; } e, s, m, p) \xrightarrow{*} (\mathtt{skip}, s[x \mapsto 1], m', p)$$

for some m', that is, the program terminates in some state $(s[x \mapsto 1], m', p)$ when started in state (s, m, p), if and only if there exists a suffix of m of the form $1m'$, that is, there exist $k \geq 0$ and $m' \in \{0,1\}^\omega$ such that $m = 0^k 1 m'$. We can compute the probability of termination with a simple calculation:

$$\begin{aligned}
&\mathbb{P}\left[\exists m' \, (x \mathtt{ := } 0 \mathtt{ ; } e, s, m, p) \xrightarrow{*} (\mathtt{skip}, s[x \mapsto 1], m', p)\right] \\
&= \mathbb{P}\left[\exists k \geq 0 \, \exists m' \; m = 0^k 1 m'\right] \\
&= \sum_{k=1}^{\infty} 2^{-k} = 1
\end{aligned}$$

as claimed in the introduction.

Example 2: A random walk on $\mathbb{Z}^2$

We now turn our attention to the program of Fig. 1.2, which implements a random walk on the set $\mathbb{Z}^2$. We show fewer details, since the previous example already described some of the simplest derivations. The program is written with the function `step` for readability and notational convenience, but it is of course equivalent to the inline program where each instance of `step` in `main` is substituted by its definition. For any $s \in \mathbb{R}^4$ and $m, p \in \mathbb{N}$, there can be one of four actions:

$$(\mathtt{step}, s, 00m, p) \xrightarrow{*} (\mathtt{skip}, s[(\mathtt{u}, \mathtt{v}) \mapsto (0, -1), (\mathtt{x}, \mathtt{y}) \mapsto (0, 0)], m, p)$$
$$(\mathtt{step}, s, 01m, p) \xrightarrow{*} (\mathtt{skip}, s[(\mathtt{u}, \mathtt{v}) \mapsto (-1, 0), (\mathtt{x}, \mathtt{y}) \mapsto (0, 1)], m, p)$$
$$(\mathtt{step}, s, 10m, p) \xrightarrow{*} (\mathtt{skip}, s[(\mathtt{u}, \mathtt{v}) \mapsto (1, 0), (\mathtt{x}, \mathtt{y}) \mapsto (1, 0)], m, p)$$
$$(\mathtt{step}, s, 11m, p) \xrightarrow{*} (\mathtt{skip}, s[(\mathtt{u}, \mathtt{v}) \mapsto (0, 1), (\mathtt{x}, \mathtt{y}) \mapsto (1, 1)], m, p)$$

The probability of each of these outcomes is $1/4$, and in all cases the program enters the loop, since $[\![\,!(\mathtt{u} == \mathtt{0}) \;||\; !(\mathtt{v} == \mathtt{0})]\!] = \mathtt{true}$.

Now define the i.i.d. random variables $X_1, X_2, \ldots$ on $\mathbb{Z}^2$ such that each X_i takes

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

one of the values $(0,1),(0,-1),(1,0),(-1,0)$, each with probability $1/4$, as well as the random variables

$$S_n = \sum_{i=1}^{n} X_i$$

where the sum is the usual componentwise sum on $\mathbb{Z}^2$. With these definitions in place, and since the loop is entered at least once, the first few steps of the reduction relation give

$$\begin{aligned}&(\texttt{main}, s, m, p) \xrightarrow{*} \\ &(\texttt{while !(u == 0) || !(v == 0) do step(u,v)}, s[(\texttt{u},\texttt{v}) \mapsto (i,j)], \mathrm{tl}^4(m), p)\end{aligned}$$

where (i,j) is distributed according to S_2 and the $\mathrm{tl}^4(m)$ indicates that four random bits were consumed. The program will exit the loop if there exists n such that $S_{2n} = (0,0)$; the factor 2 is because a return to 0 is only possible after an even number of calls to the function `step`. If such an n exists, then the rule for `while` loops gives

$$(\texttt{main}, s, m, p) \xrightarrow{*} (\texttt{skip}, s[(\texttt{u},\texttt{v}) \mapsto (0,0)], \mathrm{tl}^{4n}(m), p).$$

It follows that to compute the probability that the program terminates amounts to computing

$$\begin{aligned}&\mathbb{P}\left[\exists n\, (\texttt{main}, s, m, p) \xrightarrow{*} (\texttt{skip}, s[(\texttt{u},\texttt{v}) \mapsto (0,0)], \mathrm{tl}^{4n}(m), p)\right] \\ &\quad = \mathbb{P}\left[\bigvee_{n=0}^{\infty} S_{2n} = (0,0)\right]\end{aligned} \tag{1.11}$$

In order for $S_{2n} = (0,0)$, there must exist m such that the walk performed m steps up, m steps down, $n-m$ steps left and $n-m$ steps right. Following Durrett (1996),

$$\begin{aligned}\mathbb{P}[S_{2n} = (0,0)] &= 4^{-2n} \sum_{m=0}^{n} \frac{(2n)!}{m!m!(n-m)!(n-m)!} \\ &= 4^{-2n} \binom{2n}{n} \sum_{m=0}^{n} \binom{n}{m}^2 \\ &= 4^{-2n} \binom{2n}{n}^2.\end{aligned} \tag{1.12}$$

Since the events $S_{2n} = (0,0)$ for $n \in \mathbb{N}$ are not disjoint—one can return to (0,0) twice in $2n$ steps if $n > 1$—one cannot simply sum these contributions to get the probability of termination as we did in the previous example. However, it can be

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

shown that

$$\sum_{n=0}^{\infty} \mathbb{P}\left[S_{2n} = (0,0)\right] = \sum_{m=0}^{\infty} \mathbb{P}\left[\bigvee_{n=0}^{\infty} S_{2n} = (0,0)\right]^m, \tag{1.13}$$

as both expressions describe the expected number of visits to the origin in an infinite random walk (Durrett, 1996, Ch. 3, Thm. 2.2). Moreover, using Stirling's approximation with (1.12), it can be shown that the sums in (1.13) diverge (Durrett, 1996, Ch. 2, Thm. 1.4), which occurs iff the probability (1.11) is 1; that is, the program of Fig. 1.2 terminates with probability one. However, the *expected time to* termination is infinite: in probabilistic terms, the random walk on $\mathbb{Z}^2$ is recurrent but not positively recurrent.

Example 3: Probabilistic computation of π

The program given in Fig. 1.3 is different from the previous two examples in several ways. First of all, it samples from a *continuous* distribution, therefore relies on the full power of measure theory. Secondly, it clearly terminates, so the question this time is not to determine the probability of termination, but rather to evaluate the correctness of the program. Considering the intended purpose of the program, which is to compute an approximation of π, the question we will answer is the following: Given an error tolerance $\varepsilon > 0$, what is the probability that the final numerical value of `pi` is within ε of π?

Let $N = 1e9$ (one billion), the number of iterations of the loop. It is not difficult to see that, starting from a state (s, m, p), the program halts in a state

$$(\texttt{prog}, s, m, p) \xrightarrow{*} (\texttt{skip}, s[\texttt{i} \mapsto 4n/N, \texttt{n} \mapsto n, \ldots], m, \mathrm{tl}^{2N}(p))$$

for some integer $0 \leq n \leq N$. The value n/N is the average of N samples of the random variable

$$Z = \begin{cases} 1 & \text{if } X^2 + Y^2 < 1 \\ 0 & \text{else} \end{cases}$$

where X and Y are independent random variables distributed uniformly on $[0, 1]$.

Let us compute the expected value of Z. First, it is easy to compute the density function for X^2 (and thus Y^2):

$$\mathbb{P}\left[X^2 \leq t\right] = \mathbb{P}\left[X \leq \sqrt{t}\right] = \int_0^{\sqrt{t}} \mathbb{1}_{[0,1]}(x)\, dx = \sqrt{t}$$

for $0 \leq t \leq 1$. The density of X^2 is thus given by

$$f(t) = \frac{\partial \mathbb{P}\left[X^2 \leq t\right]}{\partial t} = \frac{1}{2\sqrt{t}} \mathbb{1}_{[0,1]}(t)$$

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

It is well known that the density of the sum of two independent distributions is given by the convolution of their densities, i.e.

$$(f * f)(t) = \int_{-\infty}^{\infty} \frac{1}{2\sqrt{x}} \mathbb{1}_{[0,1]}(x) \frac{1}{2\sqrt{t-x}} \mathbb{1}_{[0,1]}(t-x)\, dx$$
$$= \begin{cases} \displaystyle\int_0^t \frac{1}{4\sqrt{x}\sqrt{t-x}}\, dx & \text{if } 0 \le t \le 1 \\ \displaystyle\int_{t-1}^1 \frac{1}{4\sqrt{x}\sqrt{t-x}}\, dx & \text{if } 1 < t \le 2 \end{cases}$$

Since we are only interested in computing $\mathbb{P}\left[X^2 + Y^2 \le 1\right]$, we need only compute the first expression above, which under the substitution $u = \sqrt{x/t}$ yields:

$$\int_0^t \frac{1}{4\sqrt{x}\sqrt{t-x}}\, dx = \int_0^1 \frac{1}{2\sqrt{1-u^2}}\, du = \frac{1}{2}(\sin^{-1}(1) - \sin^{-1}(0)) = \frac{\pi}{4}.$$

Thus

$$\mathbb{P}\left[X^2 + Y^2 \le 1\right] = \int_0^1 (f * f)(t)\, dt = \int_0^1 \frac{\pi}{4}\, dt = \frac{\pi}{4}.$$

We have therefore computed that Z is a Bernoulli variable with probability of success $\pi/4$. In particular, its variance is $\sigma^2 = \pi/4 - (\pi/4)^2$. We can now use Chebyshev's inequality to get:

$$\mathbb{P}\left[\left|\frac{n}{N} - \frac{\pi}{4}\right| > \varepsilon\right] \le \frac{\sigma^2}{N\varepsilon^2}.$$

For example, for $\varepsilon = 0.0005$ and $N = 1e9$, the program outputs an approximation of π which is correct up to 2 significant digits with probability greater than 99.9%. In terms of the operational semantics, we can write the probability of the informal Hoare triple

$$\mathbb{P}\left[\{\}\ (\texttt{prog}, s, m, p) \xrightarrow{*} (\texttt{skip}, s', m', p')\ \{s'(\texttt{i}) \in [3.139, 3.144]\}\right] \ge 0.999.$$

We will show in Section 1.3.6 that much tighter bounds can be obtained using a more sophisticated inequality than Chebyshev's inequality.

Example 4: The Cantor distribution

The final example given in Fig. 1.4 constructs a curious object know as the Cantor distribution. This is a distribution on $[0, 1]$ that has uncountable support (it is therefore continuous), yet this support has Lebesgue measure (that is to say length) zero. It is an example of a so-called *singular distribution*.

A simple look at the program given by Fig. 1.4 shows that it does not terminate. However, it is possible to give it an operational (and denotational) semantics.

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

Starting from an initial configuration (s, m, p) where $m = m_0 m_1 m_2 \cdots$, it is not difficult to see from the rules of the operational semantics that the program generates an the infinite sequence

$$\begin{aligned}
&(\mathtt{x} := \mathtt{0}; \mathtt{d} := 1; \mathtt{w}, s, m, p) \xrightarrow{*} \\
&(\mathtt{w}, s\,[\mathtt{x} \mapsto 0, \mathtt{d} \mapsto 1], m, p) \xrightarrow{*} \\
&(\mathtt{w}, s \left[\mathtt{x} \mapsto \frac{2m_0}{3}, \mathtt{d} \mapsto \frac{1}{3}\right], \mathsf{tl}(m), p) \xrightarrow{*} \\
&(\mathtt{w}, s \left[\mathtt{x} \mapsto \sum_{i=1}^{k} \frac{2m_{i-1}}{3^i}, \mathtt{d} \mapsto \frac{1}{3^k}\right], \mathsf{tl}^k(m), p) \xrightarrow{*} \cdots
\end{aligned} \tag{1.14}$$

The program can thus be understood as generating a random real number between 0 and 1 whose base-3 expansion does not contain any ones. After k iterations of the loop, the probability that the base-3 expansion of the number is given by a particular sequence $a_1 \cdots a_k$, where $a_i \in \{0, 2\}$, is 2^{-k}. We now show how this interpretation of the program relates to the Cantor measure.

We first look at a σ-algebra. For $a_1 \cdots a_k \in \{0, 1, 2\}^k$, each set

$$C_{a_1 \cdots a_k} := \{x \in [0, 1] \mid \text{the base-3 expansion of } x \text{ starts with } a_1 \cdots a_k\}$$

is a measurable set for the usual σ-algebra on $[0, 1]$. Moreover, it can be shown that the σ-algebra generated by all these sets is in fact the usual σ-algebra on $[0, 1]$.

With the σ-algebra in place, we can define a measure. Following the behaviour of the program above, it makes sense to define

$$\mu(C_{a_1 \cdots a_k}) = \begin{cases} 2^{-k} & \text{if } a_1 \cdots a_k \in \{0, 2\}^k \\ 0 & \text{otherwise.} \end{cases} \tag{1.15}$$

It can be shown that this definition extends to the entire σ-algebra on $[0, 1]$, thereby defining a *bona fide* measure (in the same way that the definition of the product measure on rectangles extends to the entire product σ-algebra, see Section 1.2.2). This measure is the *Cantor measure*, which we shall denote by κ. Note that for any $x \in [0, 1]$, the probability of any singleton must satisfy $\kappa(\{x\}) \leq 2^{-k}$ for all k, thus $\kappa(\{x\}) = 0$ and the Cantor measure is therefore continuous. If the base-3 expansion of x contains a 1, the inequality above is trivially satisfied. If it does not, it follows from the fact that $\{x\} \subseteq C_{a_1 \cdots a_k}$, where $a_1 \cdots a_k$ consists of the first k digits in the base-3 expansion of x. It can also be shown that the set

$$C = \{x \in [0, 1] \mid \text{the base-3 expansion of } x \text{ has no ones}\}$$

has Lebesgue measure 0, but Cantor measure 1. This set, called the *Cantor set*, is

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

clearly in one-to-one correspondence with the set of possible traces described by the operational semantics described by (1.14).

1.3.5 Denotational semantics of probabilistic programs

Consider the simple program $\mathtt{x} := \mathtt{coin()}$. As we have just seen, the operational semantics models the two possible machine-state transitions defined by this program, viz.

$$(\mathtt{x} := \mathtt{coin()}, s, m, p) \longrightarrow (\mathtt{skip}, s[\mathtt{x} \mapsto \mathtt{0}], \mathsf{tl}\, m, p)$$
$$(\mathtt{x} := \mathtt{coin()}, s, m, p) \longrightarrow (\mathtt{skip}, s[\mathtt{x} \mapsto \mathtt{1}], \mathsf{tl}\, m, p)$$

depending on the value of $\mathsf{hd}\, m$, each associated with the probability $1/2$. Thus two possible executions are explored *separately*. The denotational approach explores both possibilities *simultaneously*. It does so by changing the notion of state. Operationally, a state is a machine state, i.e. a mathematical representation of the memory state and of the stack of instructions. Denotationally, a state is a *probability distribution over memory states*. The two possible memory states $s[\mathtt{x} \mapsto \mathtt{0}]$ and $s[\mathtt{x} \mapsto \mathtt{1}]$, which correspond operationally to two distinct executions, are combined into a single state $\frac{1}{2}s[\mathtt{x} \mapsto \mathtt{0}] + \frac{1}{2}s[\mathtt{x} \mapsto \mathtt{1}]$. As a consequence, the program $\mathtt{x} := \mathtt{coin()}$ can be interpreted as the operation which associates to a state s the *distribution over states* $\frac{1}{2}s[\mathtt{x} \mapsto \mathtt{0}] + \frac{1}{2}s[\mathtt{x} \mapsto \mathtt{1}]$.

More generally, since a state s can be identified with the Dirac measure δ_s (see Section 1.2.2), the denotational semantics will view the program $\mathtt{x} := \mathtt{coin()}$ as an *operator* (we will justify this term in a moment) which maps the probability distribution δ_s to the probability distribution $\frac{1}{2}s[\mathtt{x} \mapsto \mathtt{0}] + \frac{1}{2}s[\mathtt{x} \mapsto \mathtt{1}]$. This is the essence of the denotational perspective: a program is interpreted as an operator mapping probability distributions to (sub)probability distributions. It follows that denotationally, a program has a single trace (or execution) for a given input, but this trace keeps track of all possible memory state transitions and their probabilities simultaneously. It also follows that an input state could be a nontrivial distribution, for example $\frac{1}{2}s[\mathtt{x} \mapsto \mathtt{0}] + \frac{1}{2}s[\mathtt{x} \mapsto \mathtt{1}]$ could be an *input* for the program $\mathtt{x} := \mathtt{coin()}$.

We now formalize the intuition given above following Kozen (1981). Given a program obeying the syntax of Section 1.3.1 and containing n variables $\{x_1, \ldots, x_n\}$ ranging over $\mathbb{R}$, a *state* will be modelled as a probability distribution μ on $\mathbb{R}^n$ and a program e will be interpreted as an operator $[\![e]\!] \colon \mathcal{M}\mathbb{R}^n \to \mathcal{M}\mathbb{R}^n$ called a *state transformer*. To define this interpretation formally, we start with the semantics of terms. Each term t denotes a map $[\![t]\!] \colon \mathbb{R}^n \to \mathcal{M}\mathbb{R}$ defined inductively as follows.

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

Here a_i represents the value of the variable x_i, $1 \leq i \leq n$.

$$\begin{aligned}
[\![r]\!](a_1,\ldots,a_n) &= \delta_r,\ r \in \mathbb{R} \\
[\![x_i]\!](a_1,\ldots,a_n) &= \delta_{a_i} \\
[\![\texttt{coin()}]\!](a_1,\ldots,a_n) &= \frac{1}{2}\delta_0 + \frac{1}{2}\delta_1 \\
[\![\texttt{rand()}]\!](a_1,\ldots,a_n) &= \lambda \\
[\![t_1 \ \mathsf{op}\ t_2]\!](a_1,\ldots,a_n) &= \mathsf{op}_*([\![t_1]\!](a_1,\ldots,a_n) \otimes [\![t_2]\!](a_1,\ldots,a_n))
\end{aligned}$$

where $\mathsf{op} \in \{+,-,\times,\div\}$ and λ is the Lebesgue (uniform) measure on $[0,1]$. Recall that $\mu \otimes \nu$ is the product of the measures μ and ν and that op_* denotes the pushforward operation (see Section 1.2.2). In this case, $[\![t_1]\!](a_1,\ldots,a_n) \in \mathcal{M}\mathbb{R}$ and $[\![t_2]\!](a_1,\ldots,a_n) \in \mathcal{M}\mathbb{R}$, $[\![t_1]\!](a_1,\ldots,a_n) \otimes [\![t_2]\!](a_1,\ldots,a_n) \in \mathcal{M}(\mathbb{R}^2)$, $\mathsf{op}\colon \mathbb{R}^2 \to \mathbb{R}$, and $\mathsf{op}_*\colon \mathcal{M}(\mathbb{R}^2) \to \mathcal{M}\mathbb{R}$. By definition, for B a measurable subset of $\mathbb{R}$,

$$\begin{aligned}
&\mathsf{op}_*([\![t_1]\!](a_1,\ldots,a_n) \otimes [\![t_2]\!](a_1,\ldots,a_n))(B) \\
&\quad = [\![t_1]\!](a_1,\ldots,a_n) \otimes [\![t_2]\!](a_1,\ldots,a_n)\left\{(u,v) \in \mathbb{R}^2 \mid u \ \mathsf{op}\ v \in B\right\}.
\end{aligned}$$

It follows almost immediately from this definition that:

Proposition 1.3 *The denotational semantics of any term is a Markov kernel* $\mathbb{R}^n \to \mathcal{M}\mathbb{R}$.

The denotational semantics of tests is given by *subsets of* $\mathbb{R}^n$ defined inductively as follows:

$$\begin{aligned}
[\![t_1 == t_2]\!] &= \{(a_1,\ldots,a_n) \in \mathbb{R}^n \mid [\![t_1]\!](a_1,\ldots,a_n) = [\![t_2]\!](a_1,\ldots,a_n)\} \\
[\![t_1 < t_2]\!] &= \{(a_1,\ldots,a_n) \in \mathbb{R}^n \mid [\![t_1]\!](a_1,\ldots,a_n) < [\![t_2]\!](a_1,\ldots,a_n)\} \\
[\![t_1 > t_2]\!] &= \{(a_1,\ldots,a_n) \in \mathbb{R}^n \mid [\![t_1]\!](a_1,\ldots,a_n) > [\![t_2]\!](a_1,\ldots,a_n)\} \\
[\![b_1 \ \&\&\ b_2]\!] &= [\![b_1]\!] \cap [\![b_2]\!] \\
[\![b_1 \ ||\ b_2]\!] &= [\![b_1]\!] \cup [\![b_2]\!] \\
[\![!\, b]\!] &= [\![b]\!]^c.
\end{aligned}$$

Note that since we limit ourselves to deterministic guards, the comparisons in the right-hand side of the first three cases above are between Dirac deltas, thus between elements of $\mathbb{R}^n$, as one would expect. It is not hard to show the following result:

Proposition 1.4 *The denotational semantics of any test is a measurable subset of* $\mathbb{R}^n$.

Each measurable subset $B \subseteq \mathbb{R}^n$ defines a linear operator $T_B\colon \mathcal{M}\mathbb{R}^n \to \mathcal{M}\mathbb{R}^n$ defined as

$$T_B(\mu)(C) = \mu(B \cap C). \tag{1.16}$$

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

In particular, every test defines such an operator. Note that the operator T_B does not in general send probability distributions to probability distributions, since the mass outside of B is lost. Thus T_B sends probability distributions to *sub-probability distributions*, that is to say measures whose total mass is *at most* 1.

We can now define the denotational semantics of programs. Programs will be interpreted as (linear) *operators* $\mathcal{M}\mathbb{R}^n \to \mathcal{M}\mathbb{R}^n$. The inductive definition is as follows:

(i) $[\![\texttt{skip}]\!] = \mathrm{Id}_{\mathcal{M}\mathbb{R}^n} : \mathcal{M}\mathbb{R}^n \to \mathcal{M}\mathbb{R}^n$, the identity operator $\mu \mapsto \mu$.

(ii) Assignments: Prop. 1.3 interprets terms t as Markov kernels $[\![t]\!] : \mathbb{R}^n \to \mathcal{M}\mathbb{R}$. Given such a term t and an index $1 \leq i \leq n$, we define the Markov kernel $F_t^i : \mathbb{R}^n \to \mathcal{M}\mathbb{R}^n$ as

$$F_t^i(x_1, \ldots, x_n) = \delta_{x_1} \otimes \ldots \otimes \delta_{i-1} \otimes [\![t]\!](x_1, \ldots, x_n) \otimes \delta_{x_{i+1}} \otimes \ldots \otimes \delta_{x_n}$$

Using this definition, we define $[\![x_i := t]\!]$ as the operator

$$[\![x_i := t]\!](\mu) = (F_t^i)_*(\mu) \tag{1.17}$$

where $(F_t^i)_*$ is the pushforward of Markov kernels defined in (1.8). It is useful to consider special cases of this formula. Consider first the expression $x_i := r$ for some constant $r \in \mathbb{R}$. The definition above becomes

$$\begin{aligned}
&[\![x_i := r]\!](\mu)(B_1 \times \cdots \times B_n) \\
&= \int_{\mathbb{R}^n} \delta_{x_1} \otimes \cdots \otimes \delta_{x_{i-1}} \otimes \delta_r \otimes \delta_{x_{i+1}} \otimes \cdots \otimes \delta_{x_n}(B_1 \otimes \cdots \otimes B_n)\, d\mu \\
&= \int_{\mathbb{R}^n} \delta_{x_1}(B_1) \cdots \delta_{x_{i-1}}(B_{i-1}) \delta_r(B_i) \delta_{x_{i+1}}(B_{i+1}) \cdots \delta_{x_n}(B_n)\, d\mu \\
&= \mu(B_1 \times \cdots \times B_{i-1} \times \mathbb{R} \times B_{i+1} \times \cdots \times B_n) \delta_r(B_i).
\end{aligned}$$

Similarly, for $x_i :=$ `coin()` we get

$$\begin{aligned}
&[\![x_i := \texttt{coin()}]\!](\mu)(B_1 \times \cdots \times B_n) \\
&\quad = \mu(B_1 \times \cdots \times B_{i-1} \times \mathbb{R} \times B_{i+1} \times \cdots \times B_n) \left(\frac{1}{2}\delta_0 + \frac{1}{2}\delta_1 \right)(B_i).
\end{aligned}$$

(iii) $[\![e_1 \texttt{ ; } e_2]\!] = [\![e_2]\!] \circ [\![e_1]\!]$, the usual composition of operators.

(iv) $\texttt{if } b \texttt{ then } e_1 \texttt{ else } e_2$ is the operator defined by

$$[\![\texttt{if } b \texttt{ then } e_1 \texttt{ else } e_2]\!] = [\![e_1]\!] \circ T_{[\![b]\!]} + [\![e_2]\!] \circ T_{[\![b]\!]^c} \tag{1.18}$$

where $T_{[\![b]\!]}$ and $T_{[\![b]\!]^c}$ are defined as in (1.16).

(v) To define the semantics of `while` loops, we use the equivalence of the following two programs,

$$\texttt{while } b \texttt{ do } e \qquad\qquad \texttt{if } b \texttt{ then } (e \texttt{ ; while } b \texttt{ do } e) \texttt{ else skip},$$

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

to write a fixpoint equation which will define the semantics of $[\![\texttt{while}\ b\ \texttt{do}\ e]\!]$. Formally, and following (1.18), the equivalence of programs above implies that

$$[\![\texttt{while}\ b\ \texttt{do}\ e]\!] = [\![\texttt{while}\ b\ \texttt{do}\ e]\!] \circ [\![e]\!] \circ T_{[\![b]\!]} + T_{[\![b]\!]^c}$$

We therefore *define* $[\![\texttt{while}\ b\ \texttt{do}\ e]\!]$ to be the least fixpoint of the transformation on operators defined by

$$\tau(S) = S \circ [\![e]\!] \circ T_{[\![b]\!]} + T_{[\![b]\!]^c} \tag{1.19}$$

This least fixpoint can be constructed explicitly by the familiar construction

$$[\![\texttt{while}\ b\ \texttt{do}\ e]\!] = \bigvee_{n\geq 0} \tau^n(0) = \sum_{k=0}^{\infty} T_{[\![b]\!]^c} \circ \left([\![e]\!] \circ T_{[\![b]\!]}\right)^k, \tag{1.20}$$

invoking the fact that countable norm-bounded directed sets have suprema and that τ preserves such suprema.

Theorem 1.5 *The denotational semantics of any program given by the syntax of Section 1.3.1 is a positive operator of norm at most one.*

A comment on non-terminating programs

As we will see when examining the Cantor program (see Fig. 1.4) in Section 1.3.6, the construction of the least fixpoint via Eq. (1.20) returns the constant linear operator 0 for non-terminating programs of the form

$$\texttt{while true do}\ e.$$

This is in complete agreement with what happens in the formalism of Kleene Algebras with Tests (Kozen, 1997) where the following equivalence holds:

$$\texttt{while true do}\ e \triangleq (\texttt{true}; e)^*; \texttt{false} = \texttt{false}.$$

Here, `false` is the program that always aborts corresponding to the constant linear operator 0.

Recall that we also saw in Section 1.3.4 that the Cantor program can be given a non-trivial operational semantics in terms of infinite traces. How can we reconcile these two seemingly conflicting semantics?

There are (at least) two ways. The first is to declare that infinite traces are not valid, and identify diverging executions with a crash behaviour. The second, mathematically much more interesting solution, is not to consider the *least* fixpoint solution to the equation $\tau(S) = S$ (where τ is defined in Eq. (1.19)), but another fixpoint given by the *mean ergodic theorem*. When the guard of the `while` loop is `true`, Eq. (1.19) simplifies to $\tau(S) = S \circ [\![e]\!]$. Now suppose for simplicity's sake that e does not contain any `while` loop, and note that by Theorem 1.5 $[\![e]\!]$:

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

$\mathcal{M}\mathbb{R}^n \to \mathcal{M}\mathbb{R}^n$ is a is a positive operator of norm at most one. It can be shown (see e.g. Dunford et al., 1971; Eisner et al., 2015) that $[\![e]\!]$ is *mean ergodic*, that is to say that for any measure $\mu \in \mathbb{R}^n$ the limit

$$P_{[\![e]\!]}(\mu) \triangleq \lim_{n\to\infty} \frac{1}{n} \sum_{j=0}^{n-1} [\![e]\!]^j(\mu) \tag{1.21}$$

exists. Moreover, $P_{[\![e]\!]}$ is a projection $\mathcal{M}\mathbb{R}^n \to \mathrm{fix}([\![e]\!])$, the subspace of $[\![e]\!]$-invariant measures, and satisfies the fixpoint equation defining the `while` loop:

$$\tau\left(P_{[\![e]\!]}\right) = P_{[\![e]\!]} \circ [\![e]\!] = P_{[\![e]\!]}.$$

This provides an alternative fixpoint semantics in the case of the non-terminating `while` true program, which is the denotational counterpart to the operational semantics in terms of infinite traces. In fact, it would be tempting to decompose the semantics of *any* `while` loop into its terminating component, defined via the least fixpoint construction of Eq. (1.20), and its non-terminating component, defined via the mean ergodic theorem as we have just described. We leave this possibility to be explored in future work.

1.3.6 Denotational semantics through examples

Example 1: Simple Markov chain

We start by looking at the very simple program of Fig. 1.1. Since it contains a single variable, its denotational semantics is given by an operator

$$[\![\texttt{x := 0 ; while x == 0 do x := coin()}]\!] : \mathcal{M}\mathbb{R} \to \mathcal{M}\mathbb{R}$$

which we compute *compositionally*, that is to say line-by-line. Following the definition of the denotational semantics of assignments given in the previous section, the first line of the program is interpreted as:

$$[\![\texttt{x := 0}]\!] \colon \mathcal{M}\mathbb{R} \to \mathcal{M}\mathbb{R} \qquad\qquad \mu \mapsto \mu(\mathbb{R})\delta_0,$$

since $[\![\texttt{0}]\!] = \delta_0$. Note how the presence of the term $\mu(\mathbb{R})$ is what makes this operator linear: the constant function $\mu \mapsto \delta_0$—which might be a tempting semantics for assignments—would not be linear; assignments must be weighted by the total mass of the input measure. Analogously, we have

$$[\![\texttt{x := coin()}]\!] \colon \mathcal{M}\mathbb{R} \to \mathcal{M}\mathbb{R} \qquad\qquad \mu \mapsto \mu(\mathbb{R})(\tfrac{1}{2}\delta_0 + \tfrac{1}{2}\delta_1).$$

Next, we evaluate the interpretation of the `while` loop using (1.20). It is easy to see from the definition that the denotation of the test $[\![\texttt{x == 0}]\!]$ is simply the measurable singleton $\{0\}$. Now consider the first two nonzero terms in the join of (1.20):

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

$$\begin{aligned}
\tau^0(0)(\mu) &= 0 \\
\tau^1(0)(\mu) &= T_{\{0\}^c}(\mu) = \mu(- \cap \{0\}^c) \\
\tau^2(0)(\mu) &= \left(T_{\{0\}^c} + T_{\{0\}^c} \circ [\![\texttt{x := coin()}]\!] \circ T_{\{0\}}\right)(\mu) \\
&= \mu(- \cap \{0\}^c) + \frac{\mu(\{0\})}{2}\delta_1.
\end{aligned}$$

One can show by induction that

$$\begin{aligned}
\tau^k(0)(\mu) &= \mu(- \cap \{0\}^c) + \sum_{i=1}^{k-1} \frac{\mu(0)}{2^i}\delta_1 \\
&= \mu(- \cap \{0\}^c) + (1 - 2^{-(k-1)})\mu(\{0\})\delta_1,
\end{aligned}$$

which for positive μ is an increasing sequence with limit $\mu(- \cap \{0\}^c) + \mu(\{0\})\delta_1$. It follows that $[\![\texttt{while x == 0 do x := coin()}]\!] \colon \mathcal{M}\mathbb{R} \to \mathcal{M}\mathbb{R}$ is the operator defined by

$$\mu \mapsto \mu(- \cap \{0\}^c) \cap \mu(\{0\})\delta_1 = \mu(\{0,1\})\delta_1 + \mu(- \cap \{0,1\}^c).$$

The interpretation of the entire program is now obtained by operator composition:

$$[\![\texttt{x := 0 ; while x == 0 do x := coin()}]\!] \colon \mathcal{M}\mathbb{R} \to \mathcal{M}\mathbb{R} \qquad \mu \mapsto \mu(\mathbb{R})\delta_1.$$

In other words, given any input measure, the program outputs the Dirac delta over 1 up to the scalar factor required to make the operator linear. In particular, if the input is a probability distribution, then the output is simply δ_1, which is clearly consistent with the behaviour of the Markov chain of Fig. 1.1.

Example 2: Random walk on $\mathbb{Z}^2$

We start by computing the semantics of `step`. Since there are four variables `u`, `v`, `x`, `y`, we will get an operator $\mathcal{M}\mathbb{R}^4 \to \mathcal{M}\mathbb{R}^4$ (assume that the variables are ordered alphabetically; i.e., `u` corresponds to the first component of $\mathbb{R}^4$, etc.) Working line-by-line, we get

(1) $[\![\texttt{x := coin()}]\!] \colon \mathcal{M}\mathbb{R}^4 \to \mathcal{M}\mathbb{R}^4$

$$\mu \mapsto \lambda(B_1 \times B_2 \times B_3 \times B_4).\mu(B_1 \times B_2 \times \mathbb{R} \times B_4)\left(\frac{1}{2}\delta_0 + \frac{1}{2}\delta_1\right)(B_3)$$

(2) $[\![\texttt{x := coin() ; y := coin()}]\!] \colon \mathcal{M}\mathbb{R}^4 \to \mathcal{M}\mathbb{R}^4$

$$\begin{aligned}
\mu \mapsto &\lambda(B_1 \times B_2 \times B_3 \times B_4). \\
&\mu(B_1 \times B_2 \times \mathbb{R} \times \mathbb{R})\left(\frac{1}{2}\delta_0 + \frac{1}{2}\delta_1\right)(B_3)\left(\frac{1}{2}\delta_0 + \frac{1}{2}\delta_1\right)(B_4)
\end{aligned}$$

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

(3) $[\![\texttt{x := coin() ; y := coin() ; u := u + (x - y)}]\!] \colon \mathcal{M}\mathbb{R}^4 \to \mathcal{M}\mathbb{R}^4$

$$\begin{aligned}\mu \mapsto \lambda(B_1 \times B_2 \times B_3 \times B_4).& \\ \int_{\mathbb{R}^4} \delta_{u+(x-y)}(B_1)\delta_v(B_2)\delta_x(B_3)\delta_y(B_4)\, d[\![\texttt{x := coin() ; y := coin()}]\!](\mu)& \\ = \mu(\mathbb{R}^4)\Big(\frac{1}{4}\mu(B_1 \times B_2 \times \mathbb{R}^2)\delta_0(B_3)\delta_0(B_4)& \\ + \frac{1}{4}\mu(B_1 - 1 \times B_2 \times \mathbb{R}^2)\delta_0(B_3)\delta_1(B_4)& \\ + \frac{1}{4}\mu(B_1 + 1 \times B_2 \times \mathbb{R}^2)\delta_1(B_3)\delta_0(B_4)& \\ + \frac{1}{4}\mu(B_1 \times B_2 \times \mathbb{R}^2)\delta_1(B_3)\delta_1(B_4)\Big)&\end{aligned}$$

(4) $[\![\texttt{x := coin() ; y := coin() ; u := u + (x - y) ; v := v + (x + y - 1)}]\!]$: $\mathcal{M}\mathbb{R}^4 \to \mathcal{M}\mathbb{R}^4$

$$\begin{aligned}\mu \mapsto \lambda(B_1 \times B_2 \times B_3 \times B_4).\mu(\mathbb{R}^4)\Big(\frac{1}{4}\mu(B_1 \times B_2 - 1 \times \mathbb{R}^2)\delta_0(B_3)\delta_0(B_4)& \\ + \frac{1}{4}\mu(B_1 - 1 \times B_2 \times \mathbb{R}^2)\delta_0(B_3)\delta_1(B_4)& \\ + \frac{1}{4}\mu(B_1 + 1 \times B_2 \times \mathbb{R}^2)\delta_1(B_3)\delta_0(B_4)& \\ + \frac{1}{4}\mu(B_1 \times B_2 + 1 \times \mathbb{R}^2)\delta_1(B_3)\delta_1(B_4)\Big)&\end{aligned}$$

where $B_1 + 1 = \{x + 1 \mid x \in B_1\}$ and similarly for the other combinations. It now follows that

$$\begin{aligned}&[\![\texttt{u := 0 ; v := 0 ; step(u,v)}]\!](\mu)(B_1 \times B_2 \times B_3 \times B_4)& \\ &= \frac{\mu(\mathbb{R}^4)}{4}\left(\delta_0(B_1)\delta_{-1}(B_2)\delta_0(B_3)\delta_0(B_4) + \delta_{-1}(B_1)\delta_0(B_2)\delta_0(B_3)\delta_1(B_4)\right.& \\ &\quad \left. + \ \delta_1(B_1)\delta_0(B_2)\delta_1(B_3)\delta_0(B_4) + \delta_0(B_1)\delta_1(B_2)\delta_1(B_3)\delta_1(B_4)\right), \qquad (1.22)&\end{aligned}$$

where each summand corresponds to one of the four possible execution paths. This will be the input distribution to the operator denoted by the `while` loop. Let us now compute this operator. For notational clarity, we write

$$E \triangleq [\![\texttt{!(u == 0) || !(v == 0)}]\!]^c = \{0\} \times \{0\} \times \mathbb{R} \times \mathbb{R}$$

As in the previous example, we look at the first terms of the formula (1.20).

$$T_E \qquad T_E + T_E \circ [\![\texttt{step}]\!] \circ T_{E^c}$$
$$T_E + T_E \circ [\![\texttt{step}]\!] \circ T_{E^c} + T_E \circ ([\![\texttt{step}]\!] \circ T_{E^c})^2$$

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

For notational clarity, we consider measurable rectangles of the form $A \times B$, where A, B are measurable subsets of $\mathbb{R}^2$ corresponding to events involving the variables $\mathtt{u}, \mathtt{v}$ and $\mathtt{x}, \mathtt{y}$, respectively. For such an $A \times B \subseteq \mathbb{R}^4$, we have

$$T_E(\mu)(A \times B) = \mu((A \times B) \cap E) = \delta_{(0,0)}(A)\mu(\{(0,0)\} \times B).$$

This is the probability that the while loop exits immediately. Similarly,

$$\begin{aligned}
&T_E \circ [\![\mathtt{step}]\!] \circ T_{E^c}(\mu)(A \times B) \\
&\quad = \delta_{(0,0)}(A)[\![\mathtt{step}]\!] \circ T_{E^c}(\mu)((0,0) \times B) \\
&\quad = \frac{\delta_{(0,0)}(A)}{4}(\mu((0,-1) \times \mathbb{R}^2)\delta_{(0,0)}(B) + \mu((-1,0) \times \mathbb{R}^2)\delta_{(0,1)}(B) \\
&\qquad\qquad + \mu((1,0) \times \mathbb{R}^2)\delta_{(1,0)}(B) + \mu((0,1) \times \mathbb{R}^2))\delta_{(1,1)}(B)
\end{aligned}$$

where we have omitted curly brackets around singletons for readability's sake, i.e. $(1,0)$ stands for $\{(1,0)\}$, etc. We can already see that μ is evaluated at the points of $\mathbb{Z}^2$ which can reach $(0,0)$ in exactly one step. Similarly, we have

$$\begin{aligned}
&T_E \circ [\![\mathtt{step}]\!] \circ T_{E^c} \circ [\![\mathtt{step}]\!] \circ T_{E^c}(\mu)(A \times B) \\
&\quad = \delta_{(0,0)}(A)[\![\mathtt{step}]\!] \circ T_{E^c} \circ [\![\mathtt{step}]\!] \circ T_{E^c}(\mu)((0,0) \times B) \\
&\quad = \frac{\delta_{(0,0)}(A)}{4}([\![\mathtt{step}]\!] \circ T_{E^c}(\mu)((0,-1) \times \mathbb{R}^2)\delta_{(0,0)}(B) \\
&\qquad\qquad + [\![\mathtt{step}]\!] \circ T_{E^c}(\mu)((-1,0) \times \mathbb{R}^2)\delta_{(0,1)}(B) \\
&\qquad\qquad + [\![\mathtt{step}]\!] \circ T_{E^c}(\mu)((1,0) \times \mathbb{R}^2)\delta_{(1,0)}(B) \\
&\qquad\qquad + [\![\mathtt{step}]\!] \circ T_{E^c}(\mu)((0,1) \times \mathbb{R}^2)\delta_{(1,1)}(B)) \\
&\quad = \frac{\delta_{(0,0)}(A)}{4}\left(\frac{\delta_{(0,0)}(B)}{4}\mu((0,-2) \cup (-1,-1) \cup (1,-1) \times \mathbb{R}^2)\right. \\
&\qquad\qquad + \frac{\delta_{(0,1)}(B)}{4}\mu((-1,-1) \cup (-2,0) \cup (-1,1) \times \mathbb{R}^2) \\
&\qquad\qquad + \frac{\delta_{(1,0)}(B)}{4}\mu((1,-1) \cup (2,0) \cup (1,1) \times \mathbb{R}^2) \\
&\qquad\qquad \left. + \frac{\delta_{(1,1)}(B)}{4}\mu((-1,1) \cup (1,1) \cup (0,2) \times \mathbb{R}^2)\right).
\end{aligned}$$

The expression above enumerates all the points that can reach $(0,0)$ in exactly two steps and keeps track of the last move via the terms $\delta_{(i,j)}(B)$. The operators T_{E^c} annihilate any combination of two steps corresponding to a path visiting $(0,0)$ in two or fewer steps. For example, the path "down followed by up" from $(0,0)$ also reaches $(0,0)$ in zero steps, so is excluded. Note also that there is some double counting, as some paths are more likely than others.

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

To describe the operator corresponding to the k^{th} iteration of the loop, we define $P(x, y, k, l)$ as the set of paths of length $k-1$ from (x, y) to $(1, 0)$ that do not visit $(0, 0)$. A single additional "left" (l) step thus defines a path of length k to $(0, 0)$ which only visits $(0, 0)$ at the last step, hence the notation. We similarly define the obvious corresponding sets where l is replaced by d for "down", u for "up" and r for "right". With this notation, we get for $k \geq 1$

$$\begin{aligned} & T_E \circ (\llbracket \texttt{step} \rrbracket \circ T_{E^c})^k (\mu)(A \times B) \\ &= \frac{\delta_{(0,0)}(A)}{4^k} \Bigg(\delta_{(0,0)}(B) \sum_{(x,y)\in\mathbb{Z}^2} \#P(x, y, k, l)\mu((x, y) \times \mathbb{R}^2) \\ &\qquad + \delta_{(0,1)}(B) \sum_{(x,y)\in\mathbb{Z}^2} \#P(x, y, k, d)\mu((x, y) \times \mathbb{R}^2) \\ &\qquad + \delta_{(1,0)}(B) \sum_{(x,y)\in\mathbb{Z}^2} \#P(x, y, k, u)\mu((x, y) \times \mathbb{R}^2) \\ &\qquad + \delta_{(1,1)}(B) \sum_{(x,y)\in\mathbb{Z}^2} \#P(x, y, k, r)\mu((x, y) \times \mathbb{R}^2) \Bigg). \end{aligned} \tag{1.23}$$

Of course, $P(x, y, k, l)$ is nonempty for only finitely many (x, y). The expression (1.23) is uniquely determined by its values on B replaced by one of:

$$\{(0, 0)\}, \{(0, 1)\}, \{(1, 0)\}, \text{ and } \{(1, 1)\}.$$

Since this holds for any k, it also holds for the full expansion

$$\sum_{k=0}^{\infty} T_E \circ (\llbracket \texttt{step} \rrbracket \circ T_{E^c})^k (\mu)(A \times B).$$

By plugging (1.23) into the above expression and regrouping the coefficients of each term $\mu((x, y) \times \mathbb{R}^2)$, we get for $B = \{(0, 0)\}$

$$\delta_{(0,0)}(A) \left(\mu((0, 0) \times \mathbb{R}^2) + \sum_{(x,y)\in\mathbb{Z}^2\setminus(0,0)} \mu((x, y) \times \mathbb{R}^2) \left(\sum_{k=1}^{\infty} \frac{\#P(x, y, k, l)}{4^k} \right) \right), \tag{1.24}$$

and similarly for $B = \{(0, 1)\}, \{(1, 0)\}, \{(1, 1)\}$. Since $\frac{\#P(x,y,k,l)}{4^k}$ is precisely the probability that the random walk reaches $(0, 0)$ from (x, y) in exactly k steps by avoiding $(0, 0)$ until the last step, it follows that (1.24) simplifies to

$$\sum_{(x,y)\in\mathbb{Z}^2} \mu((x, y) \times \mathbb{R}^2) = \mu(\mathbb{Z}^2 \times \mathbb{R}^2),$$

since the probability of reaching $(0, 0)$ from (x, y) in some number of steps is 1. We

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

conclude that the semantics of the entire loop is the operator which sends μ to

$$\mu(\mathbb{Z}^2 \times \mathbb{R}^2)\delta_{(0,0)} \otimes \frac{1}{4}\left(\delta_{(0,0)} + \delta_{(0,1)} + \delta_{(1,0)} + \delta_{(1,1)}\right).$$

Intuitively, if the starting point is in $\mathbb{Z}^2$, then $(0,0)$ is reached with probability one.

By applying the loop operator to the initialized measure described in (1.22), it now follows that the denotation of the entire program is simply the operator mapping a measure μ to the measure

$$\mu(\mathbb{R}^4)\delta_{(0,0)},$$

as expected from the operational semantics.

Example 3: Probabilistic computation of π

Before computing the denotational semantics of the program in Fig. 1.3, it is instructive as a warm-up exercise to compute the denotational semantics of the frequent loop-iterator pattern

$$[\![\texttt{while i} < N \texttt{ do i := i + 1}]\!],$$

where N is some arbitrary constant. There is a single variable, so the semantics will be given by a linear operator $\mathcal{M}\mathbb{R} \to \mathcal{M}\mathbb{R}$. It is not hard to see from the definitions that

$$[\![\texttt{i := i + 1}]\!](\mu)(B) = \mu(\{x \mid x + 1 \in B\}) = \mu(B - 1).$$

Let us compute the first few terms of (1.20) applied to $[\![\texttt{i := i + 1}]\!]$. For $n = 1$ we get

$$\tau^1(0)(\mu)(B) = \mu(B^{\geq N}) + \mu((B^{\geq N} - 1)^{<N})$$

where $X^{\geq N} \triangleq \{x \in X \mid x \geq N\}$, $X^{<N} \triangleq \{x \in X \mid x < N\}$, and

$$(B^{\geq N} - 1)^{<N} = \{x \in \mathbb{R} \mid N \leq x + 1 \in B\}^{<N} = \{x \mid x < N \leq x + 1 \in B\},$$

where $x < N \leq x + 1 \in B$ is shorthand for $x < N \leq x + 1$ and $x + 1 \in B$. Similarly,

$$\begin{aligned}\tau^2(0)(\mu)(B) &= \mu(B^{\geq N}) + \mu((B^{\geq N} - 1)^{<N}) + \mu(((B^{\geq N} - 1)^{<N}) - 1)^{<N}) \\ &= \mu(B^{\geq N}) + \mu(\{x \mid x < N \leq x + 1 \in B\}) \\ &\qquad + \mu(\{x \mid x + 1 < N \leq x + 2 \in B\}).\end{aligned}$$

More generally, for $n \geq 1$:

$$\tau^n(0)(\mu)(B) = \mu(B^{\geq N}) + \sum_{k=0}^{n} \mu(\{x \mid x + k < N \leq x + (k + 1) \in B\}).$$

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

It follows that the operator $[\![\texttt{while i} < N \texttt{ do i := i + 1}]\!]$ maps a measure μ to the measure

$$[\![\texttt{while i} < N \texttt{ do i := i + 1}]\!](\mu)(B) \\ = \mu(B^{\geq N}) + \sum_{k=0}^{\infty} \mu(\{x \mid x + k < N \leq x + (k+1) \in B\}).$$

The intuition behind this operator is as follows. Considering the interval $B = [N, N+1]$, it is clear that if $x + k \leq N < x + k + 1$, then it will hold that $x + k + 1 \in [N, N+1]$, and thus

$$[\![\texttt{while i} < N \texttt{ do i := i + 1}]\!](\mu)([N, N+1]) \\ = \mu([N, N+1]) + \sum_{k=0}^{\infty} \mu((N-k-1, N-k]) \\ = \mu\left((-\infty, N+1]\right).$$

In other words, all the μ-mass below N accumulates in $[N, N+1]$ because it corresponds to the program exiting the loop from an initial state not satisfying its guard. Conversely,

$$[\![\texttt{while i} < N \texttt{ do i := i + 1}]\!](\mu)((N+1, \infty)) = \mu((N+1, \infty)),$$

since this is the μ-mass of states that never enter the loop. It follows that $[\![\texttt{while i} < N \texttt{ do i := i + 1}]\!](\mu)([N, \infty)) = \mu(\mathbb{R})$, i.e. any measure gets mapped to a measure whose support is $[N, \infty)$, which makes good semantic sense. Note also that if the input distribution is a Dirac delta δ_x with $x < N$, then as expected, the definition above gives

$$[\![\texttt{while i} < N \texttt{ do i := i + 1}]\!](\delta_x) = \delta_{x + \lceil N - x \rceil},$$

where $\lceil N - x \rceil$ is the ceiling function applied to $N - x$, i.e. the smallest integer above $N - x$.

With this understanding of the interpretation of the common loop-iterator pattern, let us turn to the program in Fig. 1.3. We start by examining the body of the loop, namely:

```
x:=rand();
y:=rand();
if (x*x+y*y) < 1 then n:=n+1;
i:=i+1
```

which we will denote by `body`. By applying the rules defining the denotational

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

semantics, we see that the operator $[\![\texttt{body}]\!]\colon \mathcal{M}\mathbb{R}^4 \to \mathcal{M}\mathbb{R}^4$ is

$$\begin{aligned}&[\![\texttt{body}]\!](\mu)(B_{\texttt{i}} \times B_{\texttt{n}} \times B_{\texttt{x}} \times B_{\texttt{y}})\\ &\quad = \lambda(B_{\texttt{x}} \times B_{\texttt{y}} \cap D)\mu(B_{\texttt{i}} - 1 \times B_{\texttt{n}} - 1 \times \mathbb{R} \times \mathbb{R})\\ &\qquad + \lambda(B_{\texttt{x}} \times B_{\texttt{y}} \cap D^c)\mu(B_{\texttt{i}} - 1 \times B_{\texttt{n}} \times \mathbb{R} \times \mathbb{R}),\end{aligned}$$

where $D = \{(x, y) \mid x^2 + y^2 \leq 1\}$ is the unit disk and λ is the two-dimensional Lebesgue measure (area) restricted to $[0, 1]^2$, which is equivalently given by the product of two copies of the uniform distribution on $[0, 1]$ corresponding to the two occurrences of `rand()`.

We iteratively compute $[\![\texttt{while i} < N \texttt{ do body}]\!]$ by evaluating the terms in the monotone sequence (1.20). The computation is similar to the simple warmup loop-iterator pattern above, with the operator $[\![\texttt{body}]\!]$ replacing $[\![\texttt{i := i + 1}]\!]$ as body of the loop.

$$\begin{aligned}&\tau^1(0)(\mu)(B_{\texttt{i}} \times B_{\texttt{n}} \times B_{\texttt{x}} \times B_{\texttt{y}})\\ &\quad = \mu(B_{\texttt{i}}^{\geq N} \times B_{\texttt{n}} \times B_{\texttt{x}} \times B_{\texttt{y}})\\ &\qquad + \lambda(B_{\texttt{x}} \times B_{\texttt{y}} \cap D)\mu((B_{\texttt{i}}^{\geq N} - 1)^{<N} \times B_{\texttt{n}} - 1 \times \mathbb{R} \times \mathbb{R})\\ &\qquad + \lambda(B_{\texttt{x}} \times B_{\texttt{y}} \cap D^c)\mu((B_{\texttt{i}}^{\geq N} - 1)^{<N} \times B_{\texttt{n}} \times \mathbb{R} \times \mathbb{R}).\end{aligned}$$

Iterating once more, we get

$$\begin{aligned}&\tau^2(0)(\mu)(B_{\texttt{i}} \times B_{\texttt{n}} \times B_{\texttt{x}} \times B_{\texttt{y}})\\ &\quad = \tau^1(0)(\nu)(B_{\texttt{i}} \times B_{\texttt{n}} \times B_{\texttt{x}} \times B_{\texttt{y}})\\ &\qquad + \lambda(B_{\texttt{x}} \times B_{\texttt{y}} \cap D)\left[\frac{\pi}{4}\mu(((B_{\texttt{i}}^{\geq N} - 2)^{<N} - 1)^{<N} \times B_{\texttt{n}} - 2 \times \mathbb{R} \times \mathbb{R})\right.\\ &\qquad\qquad \left. + \left(1 - \frac{\pi}{4}\right)\mu(((B_{\texttt{i}}^{\geq N} - 1)^{<N} - 1)^{<N} \times B_{\texttt{n}} - 1 \times \mathbb{R} \times \mathbb{R})\right]\\ &\qquad + \lambda(B_{\texttt{x}} \times B_{\texttt{y}} \cap D^c)\left[\frac{\pi}{4}\mu(((B_{\texttt{i}}^{\geq N} - 1)^{<N} - 1)^{<N} \times B_{\texttt{n}} - 1 \times \mathbb{R} \times \mathbb{R})\right.\\ &\qquad\qquad \left. + \left(1 - \frac{\pi}{4}\right)\mu(((B_{\texttt{i}}^{\geq N} - 1)^{<N} - 1)^{<N} \times B_{\texttt{n}} \times \mathbb{R} \times \mathbb{R})\right]\end{aligned}$$

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

since $\lambda(D) = \pi/4$ and $\lambda(D^c) = 1 - \pi/4$. Generally,

$$\begin{aligned}
&\tau^{n+1}(0)(\mu)(B_{\mathtt{i}} \times B_{\mathtt{n}} \times B_{\mathtt{x}} \times B_{\mathtt{y}}) \\
&\quad = \tau^{n}(0)(\nu)(B_{\mathtt{i}} \times B_{\mathtt{n}} \times B_{\mathtt{x}} \times B_{\mathtt{y}}) \\
&\qquad + \ \lambda(B_{\mathtt{x}} \times B_{\mathtt{y}} \cap D) \left[\sum_{k=0}^{n} \binom{n}{k} \left(\frac{\pi}{4}\right)^{k} \left(1 - \frac{\pi}{4}\right)^{n-k} \right. \\
&\qquad\qquad \left. \mu(B_{\mathtt{i}}(n+1, N) \times B_{\mathtt{n}} - k - 1 \times \mathbb{R}^2) \right] \\
&\qquad + \ \lambda(B_{\mathtt{x}} \times B_{\mathtt{y}} \cap D^c) \left[\sum_{k=0}^{n} \binom{n}{k} \left(\frac{\pi}{4}\right)^{k} \left(1 - \frac{\pi}{4}\right)^{n-k} \right. \\
&\qquad\qquad \left. \mu(B_{\mathtt{i}}(n+1, N) \times B_{\mathtt{n}} - k \times \mathbb{R}^2) \right]
\end{aligned}$$

where we have defined $B_{\mathtt{i}}(n, N) \triangleq \{x \mid x + n - 1 < N \leq x + n \in B_{\mathtt{i}}\}$. The limit distribution can be seen as an infinite sum of disjoint cases indexed by n, the number of iterations of the loop. To each n corresponds a binomial distribution with parameters $(\frac{\pi}{4}, n)$ counting the number of times $(\mathtt{x} * \mathtt{x} + \mathtt{y} * \mathtt{y} < 1)$—and thus the increment $\mathtt{n} := \mathtt{n} + 1$—was realized.

The denotation simplifies considerably when we pre-compose with the two initialisation steps of the program, namely $\mathtt{i} := \mathtt{0}$; $\mathtt{n} := \mathtt{0}$. Assuming a probability distribution $\mu \in \mathcal{M}\mathbb{R}^4$ as input, the denotational semantics is then given by

$$\begin{aligned}
&[\![\mathtt{i := 0\ ;\ n := 0\ ;\ while\ i < 1e9\ do\ body}]\!](\mu)(B_{\mathtt{i}} \times B_{\mathtt{n}} \times B_{\mathtt{x}} \times B_{\mathtt{y}}) \\
&\quad = [\![\mathtt{while\ i < 1e9\ do\ body}]\!]\mu(\mathbb{R} \times \mathbb{R} \times B_{\mathtt{x}} \times B_{\mathtt{y}})\delta_0(B_{\mathtt{i}})\delta_0(B_{\mathtt{n}}) \\
&\quad = \lambda(B_{\mathtt{x}} \times B_{\mathtt{y}} \cap D) \left[\sum_{k=0}^{1e9} \binom{1e9}{k} \left(\frac{\pi}{4}\right)^{k} \left(1 - \frac{\pi}{4}\right)^{n-k} \delta_{1e9}(B_{\mathtt{i}})\delta_{k+1}(B_{\mathtt{n}}) \right] \\
&\qquad + \ \lambda(B_{\mathtt{x}} \times B_{\mathtt{y}} \cap D^c) \left[\sum_{k=0}^{1e9} \binom{1e9}{k} \left(\frac{\pi}{4}\right)^{k} \left(1 - \frac{\pi}{4}\right)^{n-k} \delta_{1e9}(B_{\mathtt{i}})\delta_{k}(B_{\mathtt{n}}) \right].
\end{aligned}$$

Finally, we compose with $[\![\mathtt{i := 4 * n/1e9}]\!]$. Since we are only interested in the register $\mathtt{i}$ containing the approximation of π, we compute the $\mathtt{i}$-marginal given by

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

$$
\begin{aligned}
&(\pi_{\mathtt{i}})_* \left([\![\mathtt{i := 0 \ ; \ n := 0 \ ; \ while \ i < 1e9 \ do \ body \ ; \ i := 4 * n/1e9}]\!]\right)(\mu)(B_{\mathtt{i}}) \\
&\quad = [\![\mathtt{i := 0 \ ; \ n := 0 \ ; \ while \ i < 1e9 \ do \ body \ ; \ i := 4 * n/1e9}]\!](\mu)(B_i \times \mathbb{R}^3) \\
&\quad = \lambda(D)\left[\sum_{k=0}^{1e9}\binom{1e9}{k}\left(\frac{\pi}{4}\right)^k\left(1-\frac{\pi}{4}\right)^{n-k}\delta_{k+1}(\{x \mid 4x/1e9 \in B_{\mathtt{i}}\})\right] \\
&\qquad + \ \lambda(D^c)\left[\sum_{k=0}^{1e9}\binom{1e9}{k}\left(\frac{\pi}{4}\right)^k\left(1-\frac{\pi}{4}\right)^{n-k}\delta_{k}(\{x \mid 4x/1e9 \in B_{\mathtt{i}}\})\right] \\
&\quad = \sum_{k=0}^{1e9+1}\binom{1e9+1}{k}\left(\frac{\pi}{4}\right)^k\left(1-\frac{\pi}{4}\right)^{n-k}\delta_{k}(\{x \mid 4x/1e9 \in B_{\mathtt{i}}\}) \\
&\quad = \text{Binomial}\left(\frac{\pi}{4}, 1e9+1\right)(\{x \mid 4x/1e9 \in B_{\mathtt{i}}\}).
\end{aligned}
$$

In other words, from a denotational standpoint, our program returns a distribution whose `i`-marginal is the pushforward under the rescaling map $x \mapsto 4x/1e9$ of the binomial distribution Binomial $(\pi/4, 1e9+1)$. Since the binomial distribution is just a sum of Bernoulli distributions with parameter $\pi/4$, the connection with the operational semantics of Section 1.3.4 is evident. However, note that the final output of the denotational semantics captures all possible branches of the operational semantics in one single object, namely the distribution Binomial $(\pi/4, 1e9+1)$.

As promised in Section 1.3.4, we will provide a tighter bound on the probability of getting a good approximation of π via the program of Fig. 1.3. Hoeffding's inequality (Hoeffding (1994)) with X a random variable distributed as Binomial (p, n) says that

$$\mathbb{P}\left[|X - pn| \leq \varepsilon n\right] \geq 1 - 2\exp(-2\varepsilon^2 n).$$

For example, with error tolerance $\varepsilon = 0.00007$, we get that

$$\text{Binomial}\left(\frac{\pi}{4}, 1e9+1\right)(\{x \mid 4x/1e9 \in [\pi-\varepsilon, \pi+\varepsilon]\}) \geq 0.999,$$

which is a much tighter bound than the one provided by the Chebyshev inequality in Section 1.3.4. This shows that proving probabilistic guarantees about a program such as

$$(\pi_{\mathtt{i}})_*[\![\mathtt{prog}]\!]([\pi-\varepsilon, \pi+\varepsilon]) \geq 0.999$$

can depend on the difficult and purely mathematical problem of finding sufficiently tight bounds to concentration of measure inequalities.

Example 4: The Cantor distribution

We finish by computing the denotational semantics of the program of Fig. 1.4. As mentioned at the end of Section 1.3.5, the least fixpoint denotational semantics of *any* program containing a `while true` loop is simply the constant operator to 0

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

since the program never halts. Formally, if we denote by `cantor` the program of Fig. 1.4, then for every $\mu \in \mathcal{M}\mathbb{R}^2$ (since the program contains two variables)

$$[\![\texttt{cantor}]\!](\mu) = 0.$$

However, as we also discussed above, the mean ergodic theorem provides an alternative semantics to non-terminating `while true` programs. We compute this semantics explicitly for the `cantor` program and, as expected, recover the Cantor measure which we encountered when we computing the operational semantics of `cantor` in Section 1.3.4.

Once again we start by examining the body of the loop. By unravelling the definition of the denotational semantics of terms, it is not hard to compute the semantics of the body of the loop which is given by the operator sending μ to the measure

$$\begin{aligned}[\![\texttt{body}]\!](\mu)(B_{\mathtt{x}} \times B_{\mathtt{d}}) = \frac{1}{2}\mu\left(\left\{(x,d) \mid x \in B_{\mathtt{x}}, \frac{d}{3} \in B_{\mathtt{d}}\right\}\right) \\ + \frac{1}{2}\mu\left(\left\{(x,d) \mid x + \frac{2d}{3} \in B_{\mathtt{x}}, \frac{d}{3} \in B_{\mathtt{d}}\right\}\right).\end{aligned}$$

It is much easier to reason about the semantics of `cantor` in terms of Markov kernels. The body of the loop in particular is the pushforward (defined in Eq. (1.8)) of the kernel $[\![\texttt{body}]\!]^{\mathrm{ker}} \colon \mathbb{R}^2 \to \mathcal{M}\mathbb{R}^2$ defined by

$$[\![\texttt{body}]\!]^{\mathrm{ker}}(x,d) \triangleq \frac{1}{2}\delta_{(x,\frac{d}{3})} + \frac{1}{2}\delta_{(x+\frac{2d}{3},\frac{d}{3})}$$

It is easy to check that $[\![\texttt{body}]\!]^{\mathrm{ker}}_* = [\![\texttt{body}]\!]$, and since the pushforward operation $(-)_*$ is functorial we can compose these kernels to get a kernel representation $\left([\![\texttt{body}]\!]^j\right)^{\mathrm{ker}}$ of the operator $[\![\texttt{body}]\!]^j$ for each $j \geq 1$:

$$\left([\![\texttt{body}]\!]^j\right)^{\mathrm{ker}}(x,d) = \frac{1}{2^j}\sum_{w \in \{0,2\}^j} \delta_{(x+\sum_{i=1}^{j}\frac{w_i d}{3^i},\frac{d}{3^j})}$$

Using this kernel representation of $[\![\texttt{body}]\!]^j$ it becomes relatively straightforward to compute the mean ergodic limit given by Eq. (1.21), viz.

$$\lim_{n\to\infty}\frac{1}{n}\sum_{j=0}^{n-1}[\![\texttt{body}]\!]^j \tag{1.25}$$

In order to compute this limit we start by defining the σ-algebra $\mathcal{B}_{x,d}$ generated by the sets

$$C_w^{x,d} = \left\{y \mid x + \sum_{i=1}^{|w|}\frac{w_i d}{3^i} \leq y < x + \sum_{i=1}^{|w|}\frac{w_i d}{3^i} + \frac{d}{3^{|w|}}\right\},$$

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

where w is a word of $\{0,1,2\}^*$, $|w|$ is the length of this word, and w_i is the i^{th} letter in w. The sets $C_w^{x,d}$ are variants of the sets $C_{a_1\cdots a_k}$ which we defined when discussing the operational semantics of the `cantor` program in Section 1.3.4, stretched and shifted to fit the intervals $[x, x+d]$. In particular, $C_{w_1\cdots w_k} = C_{w_1\cdots w_k}^{0,1}$. In exactly the same way, these sets generate the usual (Borel) σ-algebra on each interval $[x, x+d]$.

Consider now the Markov kernel on $\mathbb{R}^2$ defined at each (x,d) by the measure $\gamma(x,d)$ on $[x, x+d]$ specified uniquely by its values on the generators $C_w^{x,d}$ via

$$\gamma(x,d)(C_w^{x,d} \times B_{\mathtt{d}}) = \begin{cases} 2^{-|w|}\delta_{(0)}(B_{\mathtt{d}}) & \text{if } w \in \{0,2\}^*, \\ 0 & \text{otherwise} \end{cases}$$

where $\delta_{(0)}$ is a curious measure defined as follows:

$$\delta_{(0)}(B) = \begin{cases} 1 & \text{if } B \text{ contains an interval } (0,\epsilon) \\ 0 & \text{otherwise} \end{cases}$$

We will see in an instant how $\delta_{(0)}$ arises, but note first that, modulo the $\delta_{(0)}$ term, the measure $\gamma(x,d)$ is simply a stretched and shifted version of the Cantor measure defined in Eq. (1.15), Section 1.3.4. In particular, $\gamma(0,1)$ is simply the product $\kappa \otimes \delta_{(0)}$ of the Cantor measure κ with $\delta_{(0)}$. We claim that γ is the Markov kernel corresponding to the mean ergodic limit Eq. (1.25). To see this we simply compute that

$$\frac{1}{n}\sum_{j=0}^{n-1}\left(\llbracket \mathtt{body} \rrbracket^j\right)^{\ker}(x,d)(C_w^{x,d} \times B_{\mathtt{d}}) = \begin{cases} 0 & n < |w| \\ \frac{1}{2^{|w|}}\frac{1}{n}\sum_{i=|w|}^{-1n}\delta_{\frac{d}{3^i}} & \text{otherwise.} \end{cases}$$

It follows that for any rectangle $C_w^{x,d} \times B_{\mathtt{d}}$

$$\lim_{n\to\infty}\frac{1}{n}\sum_{j=0}^{n-1}\left(\llbracket \mathtt{body} \rrbracket^j\right)^{\ker}(x,d)(C_w^{x,d} \times B_{\mathtt{d}}) = \gamma(x,d)(C_w^{x,d} \times B_{\mathtt{d}}) \tag{1.26}$$

Indeed, the $C_w^{x,d}$ contribution to the product is the constant $\frac{1}{2^{|w|}}$ as soon as $n \geq |w|$, and it is not hard to convince oneself that $\lim_{n\to\infty}\frac{1}{n}\sum_{i=|w|}^{n}\delta_{\frac{d}{3^i}}(B_{\mathtt{d}})$, the $B_{\mathtt{d}}$ contribution, is precisely given by $\delta_{(0)}(B_{\mathtt{d}})$: if $B_{\mathtt{d}}$ contains an interval $(0,\epsilon)$ then there will exist an N such that $\frac{d}{3^n} \in (0,\epsilon) \subseteq B_{\mathtt{d}}$ for all $n \geq N$ and the limit $\frac{1}{n}\sum_{i=|w|}^{n-1}\delta_{\frac{d}{3^i}}(B_{\mathtt{d}})$ will thus converge to 1. In all other cases the limit converges to 0. We can then conclude that the measures on the LHS and RHS of Eq. (1.26) agree on the usual (Borel) product σ-algebra on $[x, x+d]\times\mathbb{R}$ (this follows from Dynkin's π-λ lemma (Aliprantis and Border, 1999, 4.11) because the collection of sets $C_w^{x,d}$ are closed under intersection, i.e. form a π-system).

https://doi.org/10.1017/9781108770750.002 Published online by Cambridge University Press

We can now conclude that the mean ergodic denotational semantics of the program $[\![\texttt{while true do body}]\!]$ is the operator given by the pushforward of the kernel γ. In particular, the semantics of the entire `cantor` program is given by

$$\begin{aligned}
[\![\texttt{cantor}]\!](\mu) &= [\![\texttt{x} := \texttt{0}; \texttt{d} := \texttt{1}; \texttt{while true do body}]\!](\mu) \\
&= \gamma_*([\![\texttt{x} := \texttt{0}; \texttt{d} := \texttt{1}]\!])(\mu) \\
&= \mu(\mathbb{R}^2)\gamma_*(\delta_{(0,1)}) \\
&= \mu(\mathbb{R}^2)\gamma(0,1) \\
&= \mu(\mathbb{R}^2)(\kappa \otimes \delta_{(0)})
\end{aligned}$$

and we recover the Cantor measure which we obtained from the operational semantics in Section 1.3.4, as expected.

References

Aliprantis, C., and Border, K. 1999. *Infinite Dimensional Analysis*. Springer.

Chung, K. L. 1974. *A Course in Probability Theory*. 2nd edn. Academic Press.

Doberkat, Ernst-Erich. 2007. *Stochastic Relations: Foundations for Markov Transition Systems*. Studies in Informatics. Chapman Hall.

Dudley, R.M. 2002. *Real Analysis and Probability*. Cambridge Studies in Advanced Mathematics. Cambridge University Press.

Dunford, N., Schwartz, J. T., Bade, W. G, and Bartle, R. G. 1971. *Linear Operators I*. Wiley-interscience New York.

Durrett, R. 1996. *Probability: Theory and Examples*. The Wadsworth & Brooks/-Cole Statistics/Probability Series. Duxbury Press.

Eisner, T., Farkas, B., Haase, M., and Nagel, R. 2015. *Operator Theoretic Aspects of Ergodic Theory*. Vol. 272. Springer.

Giry, Michele. 1982. A categorical approach to probability theory. Pages 68–85 of: *Categorical Aspects of Topology and Analysis*. Springer.

Halmos, P. R. 1950. *Measure Theory*. Van Nostrand.

Hoeffding, Wassily. 1994. Probability inequalities for sums of bounded random variables. Pages 409–426 of: *The Collected Works of Wassily Hoeffding*. Springer.

Kozen, Dexter. 1981. Semantics of probabilistic programs. *J. Comput. Syst. Sci.*, **22**(3), 328–350.

Kozen, Dexter. 1985. A probabilistic *PDL*. *J. Comput. Syst. Sci.*, **30**(2), 162–178.

Kozen, Dexter. 1997. Kleene algebra with tests. *ACM Trans. Programming Languages and Systems (TOPLAS)*, **19**(3), 427–443.

Lawvere, W. 1962. *The category of probabilistic mappings*. Manuscript, 12 pages.

Norris, J. R. 1997. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

Panangaden, Prakash. 1998. Probabilistic relations. Pages 59–74 of: *Proceedings of PROBMIV*.

Panangaden, Prakash. 2009. *Labelled Markov Processes*. Imperial College Press.

Vitali, Giuseppe. 1905. *Sul problema della misura dei Gruppi di punti di una retta: Nota*. Tip. Gamberini e Parmeggiani.