Termination and Semantics of Probabilistic Lambda Calculus

Andrew Kenyon-Roberts
DPhil Thesis

December 13, 2024

Contents

Abstract

This thesis gives a complete method for proving the almost sure termination of probabilistic programs, in a simply-typed higher-order language with continuous random variables and an explicit recursion construct. The method is based on supermartingales and ranking functions, which are a probabilistic version of a loop variant, as described in similar work by McIver, Morgan, Kaminski and Katoen in the setting of a first-order imperative language. The basic version of this method is extended in three ways. *Sparse ranking functions* permit more flexibility in how the ranking functions may be defined, so that a sparse ranking function only needs to be defined at a subset of program states, which makes providing ranking functions much more convenient. *Antitone ranking functions* have a weaker condition on the rate of decrease than the basic version of ranking functions, so that they can be applied to programs that terminate arbitrarily slowly. *Ranking functions with respect to alternative reduction strategies* allow the order in which the program is assumed to evaluate in the ranking function to deviate somewhat from the actual reduction strategy used, again making it more convenient to define ranking functions. All of these extensions may be combined, and it is proven that if a program has an antitone sparse ranking function with respect to any reduction strategy, it is almost surely terminating.

There is also an overview of probabilistic programming, with a particular focus on some of the many different ways of defining a formal semantics for probabilistic lambda calculus. The last extension of the ranking function method, relating to alternative reduction strategies, makes use of a novel variant on the operational trace-based semantics with a limited confluence result despite the random sampling. This confluent trace semantics may be of interest beyond just its application to proving almost sure termination.

Chapter 1

Introduction

1.1 Motivation

Probabilistic programming is a powerful method for using bayesian inference to understand the behaviour of systems which may be modelled but with some degree of uncertainty. In systems too complicated for the direct application of Bayes' Law to be feasible, there are a variety of algorithms that may be used, and probabilistic programming languages such as Church [25], Stan [9], Anglican [50], Gen [17], Pyro [7], Edward [51] and Turing [23] provide a highly flexible uniform way of specifying probabilistic models so that inference algorithms may be applied without having to re-make the inference algorithms for each application. They attain this flexibility by including in the model specification language all the features required to make a programming language Turing-complete.

This of course comes at the cost that not all programs will terminate. The appropriate notion of termination for probabilistic programs is in many cases either termination with probability 1 (known as almost-sure termination, AST) or finite expected runtime (known as positive almost-sure termination, PAST), and there are many algorithms and theorems which are only applicable to AST or PAST programs (e.g. [46, 25, 36], and via [36]'s result that AST programs have almost-everywhere differentiable density functions, also [56, 43, 35] which rely in turn on this differentiability condition). Proving AST is therefore useful, but since checking AST is Π^0_2-complete[30, 6], there is no algorithm for checking AST, and somewhat less automatic criteria, like Theorems 3.4, 3.7, 3.9, 3.10 and 4.4, must be used instead. There are various existing approaches to proving AST[31, 44, 30, 39, 22, 14, 12], but the theorems that may be applied vary somewhat depending on the features of the language being considered. This work (or rather, the paper produced in the course of the research leading up to this thesis, [32]) is the first such work I know of that is suitable for proving AST for a higher-order language with continuous distributions.

1.2 Approach to Proving AST

The approach used here to prove AST and PAST is based on ranking functions (Definition 3.2). Roughly speaking, a ranking function is a function from program states to non-negative real numbers such that as the program executes, the value of the ranking function decreases at a sufficiently fast rate to guarantee that the program eventually terminates. This is based on a very similar approach used in [22] and [40], the main difference being that here, it is applied to a functional language rather than an imperative one. Although the proof of the basic version of the ranking function theorem (Theorem 3.4, that any term in the language PPCF which has a ranking function is AST) is not all that essentially different from the imperative version, it turns out that in order for it to actually be practical to define ranking functions for anything more than a trivially complicated program, some extensions unique to the higher-order case are needed.

Since a functional language does not have distinct `while` loops the same way an imperative language does, the ranking functions are defined using the set of terms reachable from the given term as their domain (the set of program states), rather than the more convenient definition of ranking functions as a sort of loop variant in the imperative case. Defining the ranking function value for every single reachable state is terribly verbose and not really logically necessary, so the first extension, described in Section 3.2, is to a different sort of ranking function (*sparse ranking functions*) which can be defined on essentially an arbitrary subset of the reachable states, as the user prefers. This extension also improves the compatibility of the ranking function method with syntactic sugar, and allows the method to more easily integrate sub-proofs of termination in certain cases by other methods.

The second extension, to *antitone ranking functions*, described in Section 3.3, is not unique to the higher-order case, unlike the others (and is similar to the results of [40]), but is necessary to make the method more powerful. Only those terms which terminate sufficiently quickly (the Y-PAST terms, see Definition 3.4) have ranking functions, but by loosening the condition on the rate at which the ranking function must decrease, we can make it the case that every AST term[1] has an antitone ranking function (Theorem 3.12) while still ensuring that every term with an antitone ranking function is AST (Theorem 3.9). This extension is also compatible with the previous one, so antitone sparse ranking functions have the expected properties.

The third extension, described in Chapter 4, is by far the most complicated and the most interesting. When using a functional language, it is natural to think of it in terms of the equational reasoning permitted by the lambda calculus, where sub-terms may be rearranged and executed in various different orders, all of which reach the same result due to confluence. It is quite tricky to apply confluence results to the probabilistic lambda calculus, or even to define a confluence result that is true in this context. Most trivially, a program that

[1] assuming it is also a closed term, for uninteresting reasons

just takes a single random sample and returns it could reduce to either 0 or 1, immediately breaking confluence unless some effort it taken to ensure that the same samples are used in every run. Even after this, there are several more ways in which confluence fails. By defining a new version of the semantics of PPCF (the model language I am using) specifically to work around these issues, and also restricting it to only a subset of reduction orders to avoid some more essential failures of confluence, I prove a restricted confluence result (Theorem 4.1). This is then used to prove the third extension to the ranking function result, the extension to *ranking functions w.r.t. alternative reduction strategies*. Because of confluence (and the fact that the default reduction strategy terminates most often of all permitted reduction strategies, Theorem 4.3), a term which has a ranking function w.r.t. any reduction strategy is AST in the sense of the standard call-by-value reduction strategy (Theorem 4.4). This extension can also be combined with both of the previous two. Theorem 4.4 does apply to somewhat more terms than the basic ranking function theorem, but that is not the main point. Theorem 3.9 is already complete so Theorem 4.4 is not any more powerful than that. The point, as with sparse ranking functions, is to make the ranking functions easier to define.

The exploration of the limits of confluence for probabilistic programs, the way that these limits can be partially overcome, and the semantics used to prove these results are also interesting beyond just their application to proving AST.

1.3 Semantics

There is a wide variety of different ways that a higher-order probabilistic language may be represented in a formal semantics, with tradeoffs in complexity and ease of proving various properties. As in the case of non-probabilistic languages, there is the distinction of operational versus denotational semantics. Operational semantics corresponds more closely to how programs are actually executed on a real computer, and is particularly suitable for discussing properties such as running time and termination, whereas denotational semantics is better at representing program transformations and equivalences. There is also the question of how to represent probabilities specifically. A trace-based semantics, where something like a probability space is used, is more convenient for correlating program behaviour to other derived functions of the program's progress and the random samples, and is in a sense more fine-grained, whereas a distribution-based semantics is good for ignoring irrelevant details about when particular outcomes occur and how the random samples are used, in favour of just how likely each one is, which is generally the more important thing to know about a program. Both trace-based and distribution-based semantics can be combined with operational and denotational semantics (with particular complications in each version of denotational semantics), and within each of these categories there are various sub-types as well: big-step vs. little-step operational semantics, various different categories (in the category theory sense) in which a denotational semantics may be defined, infinite vs. finite traces, etc. Sections

2.4.2 and 2.5 provide a summary of the topic, with examples of each of these major categories, providing some points of comparison for the confluent trace semantics (a new variant which falls into the trace-based operational category).

1.4 Acknowledgements

Many of the results in this thesis were originally published in [32] (or its more complete arXiv version), which was written in collaboration with my first supervisor, Luke Ong. The parts we collaborated on are mostly some parts of Chapter 3, the rest being my work. I would also like to thank my second supervisor, Bartek Klin, who provided useful feedback and advice while I was writing this thesis. I was funded for this research by the UK's Engineering and Physical Sciences Research Council.

Chapter 2

Preliminaries

2.1 Probabilistic Programming

A probabilistic programming language is simply a language which includes a sampling construct for producing random (or pseudo-random) values and a conditioning construct. The first of these is available in any full-featured programming language, for example the `rand()` function from the C standard library. It is conditioning that makes probabilistic programming unique. Conditioning corresponds to evidence in bayesian statistics, where sampling gives the prior.

Example 2.1.1. Consider the following program.

```
x := flip_coin()
y := flip_coin()
observe x=Heads || y=Heads
```

The command `observe` excludes runs where its condition is not satisfied, with the probabilities of all of the remaining runs scaled up to compensate so they still add up to 1. In this example, there are three remaining possible outcomes. `x`, `y`, or both can be heads, and each of these outcomes, after rescaling, gets a probability of $\frac{1}{3}$.

Example 2.1.2. This sort of construct is known as hard conditioning, but evidence doesn't have to be all-or-nothing. Suppose a doctor runs a screening test for whether their patient has some disease. 80% of people with the disease give a positive result on the test, as do 1% of healthy people, and before seeing the test results, the doctor gives a 5% chance of the patient having the disease. The test comes up positive. This can be modeled with this probabilistic program.

```
ill := sample_bernoulli(0.05)
if (ill)
    score 0.8
else
    score 0.01
```

The `score` construct here, which does soft conditioning, is a generalisation of `observe` which weights the execution path it occurs in by its argument. In this case, the possibility that `ill` is true starts out with a probability of 0.05 which is multiplied by the likelihood of 0.8 to give 0.04, and the other option has a value 0.0095, so after normalising the total probability, the chance that `ill` is true is approximately 80.8%. If the argument of `score` is bounded above, as it is here, soft conditioning can be implemented in terms of hard conditioning.

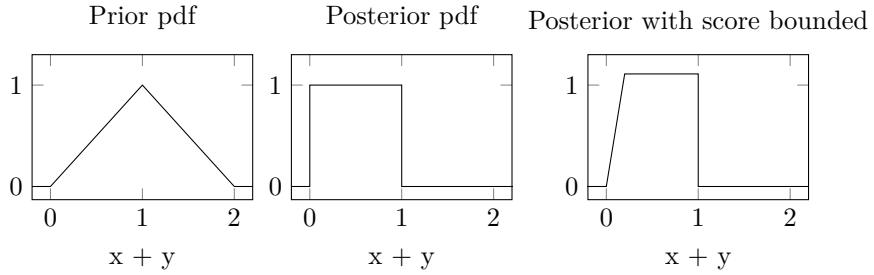
```
function score(x) {
    observe sample_bernoulli(x/bound)
}
```

In practice this may be less computationally efficient than having a built-in soft conditioning operator, but in theory it's equivalent provided the number of calls is constant across runs. Each run ends up having a probability `x/bound` of not being rejected, and since the posterior distribution is re-scaled to have total probability 1 anyway, the factors of `1/bound` cancel out. If there is no upper bound on the likelihood though, this translation fails because the parameter of the Bernoulli distribution has to be in the range 0 to 1. For example, consider this program.

Example 2.1.3.

```
x := sample_uniform(0,1)
y := sample_uniform(0,1)
observe x + y < 1
score 1/(x+y)
```

The `score` construct precisely compensates for the variation in probability density of `x + y` between 0 and 1 to give a posterior distribution of `x + y` that is uniform in this range. For any value of the bound used to simulate `score` with `observe`, it fails for sufficiently small values of `x + y`.



Although unbounded likelihoods are strange in the context of bayesian reasoning on discrete distributions, where they represent probabilities and are therefore never greater than 1, when reasoning about continuous distributions they are quite natural since probability density functions can be unbounded. Suppose a sample is drawn from a Gaussian distribution with mean 0 and unknown variance parameter and it is observed that its value is x. If we have some

prior P on the parameter of the distribution and we want to know what the posterior is, we can simulate the situation like this.

```
v := sample(P)
score gaussian_pdf(0,v,x)
```

It would not have been possible to phrase this as `observe sample_gaussian(0,v)=x` because the probability of the condition being true is always 0 unless $v = x = 0$. The lower `v` is, so long as it greater than about x^2, the higher the likelihood. In practice, probabilistic programming languages often have sampling and immediately conditioning on the value (like `observe sample_gaussian(0,v)=x` in the pseudocode notation I've been using) as a primitive construct so that it is possible, but this is usually equivalent to soft conditioning, and can be considered syntactic sugar for it.

2.2 Almost Sure Termination

All of the examples considered so far have been simple programs with no loops, but in order to encode interesting and non-trivial programs, loops are usually necessary, which brings the possibility of non-termination. Definitions of termination in the context of probabilistic programming must take into account that there is in general some probability of terminating rather than a certainty. One significant version of termination is almost sure termination (AST), the condition that the program terminates with probability 1. This is weaker than requiring that every computational path terminates, because it is possible that all the non-terminating paths, in the course of their infinitely long execution, have probabilities that tend to 0.

Example 2.2.1.

```
c := flip_coin()
while (c = Heads) {
    c = flip_coin()
}
```

The probability that this program terminates after exactly n flips for $n > 0$ is 2^{-n}, so the total probability that it terminates after any number of flips is $\sum_{n=1}^{\infty} 2^{-n} = 1$. There is nevertheless the possibility that every single flip returns tails, in which case the program fails to terminate, it's just that the probability of this happening is $\lim_{n\to\infty} 2^{-n} = 0$ (and crucially there are few enough such paths that the total probability of all of them remains 0 too).

2.2.1 Conditioning

In the context of soft conditioning, the definition of AST seems to get more complicated. If the program terminates, it is possible to assign posterior probabilities to (sufficiently well-behaved) subsets of the execution paths, but if some

paths don't terminate, these won't have well-defined probabilities. The total score for any path is the product of the parameters of each score construct that gets executed along that path. If the path is infinitely long, this is an infinite product which may fail to have a limit.

Example 2.2.2.

```
c := flip_coin()
n := 0
while (c = Heads) {
    n := n + 1
    score 3^n
    c := flip_coin()
    score 3^-n
}
```

This program is largely equivalent to the previous one if we ignore the variable `n` which does not affect the condition of the while loop. The `score 3^n` and `score 3^-n` cancel out, but if we take the program state immediately after an execution of `score 3^n`, the value of the unnormalised posterior up to that point for the path that hasn't terminated yet is $(\frac{3}{2})^n$ which is greater than the unnormalised posterior of any of the runs which have already terminated. If we take the normalized posterior of the program state up to a certain number of execution steps, it switches back and forth between being dominated by the runs that terminated quickly and the runs that are still going.

To avoid this issue, conditioning is ignored in the definition of AST, and only the priors are used. With this definition, all the necessary limits are well-defined, since the probability of an execution path of length n is equal to the total probability of all paths of length m of which it is a prefix, for any fixed $m \geq n$ (including those that terminate in fewer than n or m steps and don't do anything more as still being valid paths of those lengths). Hard conditioning does not have this issue, but also doesn't affect AST except in the cases where almost all runs are excluded, and these cases tend to be considered not well-defined anyway even if some of them intuitively do have sensible meanings. In any case, this definition of AST that doesn't consider conditioning turns out to be useful.

Example 2.2.3. This does have the effect that some transformations which would seem to leave the program's meaning unchanged actually change whether it's AST or not.

```
function step_plain() {
    if (sample_bernoulli(1/3))
        return 1
    else
        return -1
```

```
}

function step_score() {
    if (sample_bernoulli(2/3)) {
        score 1/2
        return 1
    } else {
        score 2
        return -1
    }
}

x := 10
while (x > 0) {
    x := x + step_plain()
}
```

This is a random walk biased towards 0, where it terminates. The function `step_score` is essentially equivalent to `step_plain`, as both give the same distribution of return values, but if we replace the use of `step_plain` in the loop by `step_score`, the program counterintuitively becomes not AST. This happens because without the conditioning operators, `step_score` becomes biased in the positive direction, so the random walk becomes biased away from 0.

2.2.2 Positive Almost Sure Termination

Another useful version of termination is positive almost sure termination (PAST). A program is PAST if the expected number of execution steps is finite. This is a stronger condition than AST, because if there's a non-zero probability that the execution time is infinite, the expected execution time is necessarily infinite, but some AST programs fail to be PAST. An unbiased random walk in one or two dimensions is a well-known example of this.[53]

Example 2.2.4. A more direct (and easier to prove) example is this.

```
x := sample_uniform(0,pi^2/6)
n := 0
s := 0
while (s < x) {
    n := n + 1
    s := s + 1/n^2
}
```

The value of `s` tends to $\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$, therefore eventually `s < x` becomes false unless `x` is exactly $\frac{\pi^2}{6}$, which has probability 0, therefore this program is AST, but the probability of the loop executing n times before terminating is $\frac{6}{\pi^2 n^2}$ so the expected number of steps is

$$\begin{aligned}
&\sum_{n=1}^{\infty}(4+3n)\frac{6}{\pi^2 n^2}\\
=&\sum_{n=1}^{\infty}(\frac{24}{\pi^2 n^2}+\frac{18}{\pi^2 n})\\
=&\infty
\end{aligned}$$

2.2.3 Tail Bounds

Another termination condition, more precise than almost sure termination but more general than PAST, is sets of tail bounds on the runtime, that is, functions $f : \mathbb{N} \to [0, 1]$ such that $\mathbb{P}[\text{runtime} > n] \leq f(n)$. The method of ranking functions can be used to give tail bounds for the runtime of terms (as in Corollary 3.2, with the obvious modification made to bound the runtime rather than Y-runtime), and ranking-function based conditions can be constructed that imply given tail bounds (using a construction similar to Theorem 3.12), but providing more precise bounds on runtime than it just being finite is not a focus of this thesis.

2.3 Probability Measures

In formalising the notion of probability, it is most convenient to enumerate all of the possible outcomes of all of the random events simultaneously in a set, Ω, the sample space, then consider all of the random variables as functions from this set. For example, if the situation is rolling two dice, we could take $\Omega = [1, 6]^2$, with the values of the first and second dice being the first and second projections $\pi_1, \pi_2 : [1, 6]^2 \to [1, 6]$. This formulation has the advantage that it makes all of the correlations and dependencies among the variables fully explicit. Probability, in this context, is a function $\mathbb{P}$ which takes subsets of Ω (events) and returns values between 0 and 1. As is customary when discussing probabilities, some syntactic sugar will be used to avoid explicitly mentioning the sets and elements of the sample space involved. If f is a function and $X_1, \dots$ are random variables, the random variable $x \mapsto f(X_1(x), \dots)$ will usually be denoted $f(X_1, \dots)$, and if ϕ is a predicate, $\mathbb{P}(\{x \mid \phi(X_1(x), \dots)\})$ will be denoted $\mathbb{P}[\phi(X_1, \dots)]$, for example $\mathbb{P}[X_1 = X_2]$ or $\mathbb{P}[X^2 + Y^2 < 1]$. If a proposition has probability 1, it is said to be "almost surely" or "a.s." true (like if it is almost surely true that a program terminates). In order to make sense as a probability, $\mathbb{P}$ must satisfy some axioms.

$$\mathbb{P}(\Omega) = 1$$
$$\mathbb{P}(X) \geq 0$$
$$\sum_{n=0}^{\infty} \mathbb{P}(X_n) = \mathbb{P}\left(\bigcup_{n=0}^{\infty} X_n\right) \quad \text{if } X_n \text{ are mutually disjoint}$$

For finite sample spaces like the dice example, this all works fine, and even for countably infinite spaces, the axioms are all satisfied by giving each element x of Ω some probability $\mathbb{P}(\{x\})$ such that $\sum_{x\in\Omega} \mathbb{P}(\{x\}) = 1$, then setting $\mathbb{P}(X) = \sum_{x\in X} \mathbb{P}(\{x\})$. If countable spaces are all we're considering, defining probabilities in this way with the probability function assigning a value to each element of Ω rather than subsets is simpler, but it does not extend to uncountable sets well.

First, note that the additivity axiom only applies to countable families of events. This is necessary because the union of uncountably many events each with probability 0 can have a non-zero probability in total. Take the uniform probability distribution on the subset $[0, 1)$ of the real numbers for example. We take $\Omega = [0, 1)$ and want $\mathbb{P}(\Omega) = 1$ of course. The defining feature of the uniform distribution is that $\mathbb{P}([a, b)) = b - a$ where $0 \leq a \leq b \leq 1$, so by the additivity and non-negativity axioms, $\mathbb{P}(\{x\}) < b - a$ for all $0 \leq a \leq x < b \leq 1$ i.e. $\mathbb{P}(\{x\}) = 0$, therefore

$$\sum_{0\leq x<1} \mathbb{P}(\{x\}) = 0 < 1 = \mathbb{P}\left(\bigcup_{0\leq x<1} \{x\}\right).$$

Even with the restriction that the additivity axiom only applies to countable families, there's still a problem. The uniform distribution ought to be translation invariant (modulo 1), i.e. $\mathbb{P}(X) = \mathbb{P}(\{x + c \mod 1 \mid x \in X\})$ for all c. Let W be a set containing one member of each equivalence class in $[0, 1)$ under equivalence modulo the rational numbers (by the axiom of choice). For all q, $\mathbb{P}(W) = \mathbb{P}(W+q)$, but $\bigcup_{0\leq q<1, q\in\mathbb{Q}}(W+q) = [0, 1)$ (where we're still taking sums mod 1 of course). The probability of W times $\aleph_0$ has to be 1, but this property does not hold for any real number. The standard solution is to refuse to assign W and other sets like it any probability at all. It is not the sort of set we would expect to want to know the probability of anyway, being a pathological case specified in a non-constructive way, the problem then is just to specify which events should get probabilities and which shouldn't.

2.3.1 Sigma Algebras

In any probability space (or, more generally, in any measure space, where the total measure need not add up to 1), those subsets which are given a well-defined value are called "measurable" or in probability spaces "events", with the set of

measurable subsets being a σ-algebra $\mathcal{F}$. To be a σ-algebra, $\mathcal{F}$ must satisfy some axioms.

$$\begin{aligned} \Omega &\in \mathcal{F} \\ \forall X \in \mathcal{F}.\, \Omega \setminus X &\in \mathcal{F} \\ \forall (X_n)_n \in \mathcal{F}^{\mathbb{N}}.\, \bigcup_{n \in \mathbb{N}} X_n &\in \mathcal{F} \end{aligned}$$

These are just the restrictions necessary to get the probability space axioms to work. With this definition, we can now define a probability space as a triple $(\Omega, \mathcal{F}, \mathbb{P})$ where $\mathcal{F}$ is a σ-algebra on Ω and $\mathbb{P}$ is a function $\mathcal{F} \to \mathbb{R}$ which satisfies the axioms given earlier. As usual for this sort of mathematical structure, in most cases a probability space will usually be identified with its sample space or just the sample space and probability function rather than spelling out all three components each time.

Every set can be equipped with the discrete σ-algebra, which is simply its powerset. For countable sets, this is usually the σ-algebra used, as those that are not themselves discrete differ from the discrete σ-algebra only in that some of their elements are indistinguishable. For the real numbers, there are a few options. The discrete σ-algebra is unsatisfactory because, as seen above, it cannot support some very natural continuous probability distributions. The Borel σ-algebra is the set of subsets of $\mathbb{R}$ which can be formed by taking complements and countable unions of open sets, or equivalently, the minimal σ-algebra which contains all of the open sets of $\mathbb{R}$. This is usually satisfactory, and is the σ-algebra I will be assuming $\mathbb{R}$ has if not otherwise specified. The set of Lebesgue measurable sets is a larger σ-algebra on $\mathbb{R}$ which is sometimes used and has some desirable properties the Borel σ-algebra lacks, but its definition is a bit more complicated.

Given an integrable non-negative function f on the reals, a measure μ can be defined on the reals such that for all $a < b$, $\mu([a, b]) = \int_a^b f(x)\ \mathrm{d}x$, which is said to have density function f. If $\int_{-\infty}^{\infty} f(x)\ \mathrm{d}x = 1$, it is a probability density function. The density function can be thought of as the ratio of the measure μ to the standard measure on $\mathbb{R}$ (the Lebesgue measure), and this can be generalised to be relative to other measures too, just replacing the integral in the definition with the integral with respect to that measure. In this general case, the density function is also known as the Radon-Nikodym derivative.

Another sort of uncountable measure space that will be important later is one which includes infinitely many independent random variables. These can be defined as products of families of measure spaces. If $(\mathcal{F}_i)_{i \in I}$ are σ-algebras on $(\Omega_i)_{i \in I}$, the product can be defined as the minimal σ-algebra containing

$$\{\{f \mid f(i) \in X\} \mid i \in I, X \in \mathcal{F}_i\}$$

and is a σ-algebra on $\Pi_{i \in I} \Omega_i$. This is the category-theoretic product if the morphisms between σ-algebras are taken to be measurable functions, functions

between the underlying sets such that the inverse image of each measurable set it measurable. The product of the probability measures can also be formed, as the unique measure consistent with $\mathbb{P}(\{f \mid f(i) \in X\}) = \mathbb{P}_i(X)$, although this construction only works for probability measures, not general measures, since in infinite products of numbers not bounded in the range 0 to 1 like probabilities, there are issues of convergence.

Sum, or coproduct, σ-algebras will also be useful. If $(\mathcal{F}_i)_{i \in I}$ are σ-algebras on $(\Omega_i)_{i \in I}$, then the sum σ-algebra is the minimal σ-algebra on $\biguplus_{i \in I} \Omega_i$ containing the image of X in $\biguplus_{i \in I} \Omega_i$, for each $i \in I$ and $X \in \mathcal{F}_i$.

2.3.2 Expectations

For the definition of almost sure termination, having defined probabilities suffices, but for PAST, ranking functions (which will prove to be useful for proving programs AST) and generally giving meaning to the results of programs, it is useful to also have the notion of expectations (informally, the averages values of random variables). It is defined as the Lebesgue integral:

$$\mathbb{E}[X] := \int_0^\infty \mathbb{P}[X > x] \, \mathrm{d}x - \int_{-\infty}^0 \mathbb{P}[X < x] \, \mathrm{d}x.$$

Equivalently, it can be defined as the unique function such that $\mathbb{E}[\mathbf{1}_A] = \mathbb{P}(A)$ where $\mathbf{1}_A$ is the indicator function, $\mathbb{E}[X + aY] = \mathbb{E}[X] + a\mathbb{E}[Y]$ and $|\mathbb{E}[X]| \leq \sup |X|$.

This gives overall average values, but the average value of one variable given some information about other parts of the random state are also useful, for example in the context of calculating the expectation of the result of a program that is already part way through its run. For discrete conditions, the conditional expectation of variable X given event A is just $\mathbb{E}[X \mid A] = \frac{\mathbb{E}[X \mathbf{1}_A]}{\mathbb{P}(A)}$. It would be nice to also be able to condition on the values of continuous random variables, but this definition doesn't work in that case because the probability of any continuous random variable taking any exact value is 0, which leads to the conditional expectation being $\frac{0}{0}$. A more complicated definition is therefore needed in this case.

Any state of partial information on the random state can be summarised as another σ-algebra $\mathcal{F}'$ which is a subset of $\mathcal{F}$. Two random states are considered indistinguishable given this information iff there is no set measurable in $\mathcal{F}'$ containing one of the elements but not the other. A σ-algebra which is a subset of another on the same underlying set Ω is called "coarser" (and the superset "finer"). The finest σ-algebra on a given set is the discrete σ-algebra, and the coarsest is $\{\Omega, \emptyset\}$. The relevance of this to conditional expectations is that these σ-algebras coarser than the one $\mathbb{P}$ is defined on can encode the information being conditioned on. If that information is the values of some random variables $X_1, \ldots$, the corresponding σ-algebra $\sigma(X_1, \ldots)$ is the coarsest such that all of those random variables are measurable with respect to it ("adapted

to it"), i.e. the σ-algebra generated by the inverse images of the random variables' codomains' measurable sets in the random variables. For example, take Ω to be $\{-2,-1,0,1,2\}$, and take X to be the squaring function, then $\sigma(X)$ contains $\{-2,2\}$, $\{-1,1\}$, and $\{0\}$ because they are the inverse images under X of $\{4\}$, $\{1\}$, and $\{0\}$ respectively (which are all measurable under the standard σ-algebra on $\mathbb{N}$, the discrete σ-algebra), and also contains all of their unions because σ-algebras are closed under countable unions and there are only finitely many of them, but $\sigma(X)$ does not contain $\{2\}$, because there is no way to distinguish 2 from -2 by their squares.

The conditional expectation $\mathbb{E}[X \mid \mathcal{F}']$ is defined as a random variable adapted to $\mathcal{F}'$ such that for all $A \in \mathcal{F}'$, $\mathbb{E}[\mathbf{1}_A \mathbb{E}[X \mid \mathcal{F}']] = \mathbb{E}[1_A X]$. This also implies that for any Y adapted to $\mathcal{F}'$, $\mathbb{E}[Y\mathbb{E}[X \mid \mathcal{F}']] = \mathbb{E}[YX]$. The expectation conditioning on some set of variables $\mathbb{E}[X \mid \sigma(Y_1,\dots)]$ is also denoted $\mathbb{E}[X \mid Y_1,\dots]$, and with this notation, $\mathbb{E}[f(Y_1,\dots)\mathbb{E}[X \mid Y_1,\dots]] = \mathbb{E}[f(Y_1,\dots)X]$ for any f. The conditional expectation as defined in this way is itself a random variable since it can take different values depending on part of the random information. It is in particular a random variable adapted to $\mathcal{F}'$, which means that for values of the random information which are not distinguished in $\mathcal{F}'$ (like 2 and -2 above), the conditional expectation takes the same value. Substituting values in like $\mathbb{E}[X \mid Y_1 = y_1,\dots]$ therefore gets a constant non-random result like non-conditional expectations.

Conditional expectations are not in general unique, in that if Z is a random variable adapted to $\mathcal{F}'$ which is almost surely 0, and Y is a conditional expectation for X w.r.t. $\mathcal{F}'$, then so is $Y + Z$. This means that individual values of the conditional expectation are often not well defined, even if the function as a whole is not allowed to vary much. In most cases, there is one obviously correct answer and infinitely many silly very discontinuous variants on it. For example, if X and Y are two independent random variables uniformly distributed in the range 0 to 1, clearly $\mathbb{E}[X \mid Y = y]$ should be $\frac{1}{2}$ for all values of y, even if technically it could take arbitrary other values for countably many cases. In cases such as this, or when conditioning on a program's execution up to a certain point, I will be using the obviously correct conditional expectations rather than arbitrary ones.

2.4 PPCF, SPCF, and their Semantics

The method of proving AST using ranking functions is applicable to a variety of languages, but for concreteness I will be using the language PPCF (Probabilistic Programming Computable Functions), an extension of PCF to include random sampling of real numbers from the interval $[0,1]$. It is a subset of SPCF (Statistical PCF)[52] as used in [36] which lacks conditioning (including conditioning would not substantially change any of the arguments). It is a call-by-value (CBV) simply-typed lambda calculus with real numbers, loops, and random sampling. Types and terms are defined as follows, where r is a real number, x

is a variable, and $f : \mathbb{R}^n \to \mathbb{R}$ is any measurable function:

$$\begin{aligned} \textit{types } A, B &:= \mathsf{R} \mid A \to B \\ \textit{values } V &:= \lambda x.M \mid \underline{r} \\ \textit{terms } M, N &:= V \mid x \mid M_1\, M_2 \mid \underline{f}(M_1, \dots, M_n) \mid \mathsf{Y}\, M \\ &\quad \mid \mathsf{if}(M < 0, N_1, N_2) \mid \mathsf{sample} \end{aligned}$$

Of course, in a real programming language there would only be some fixed set of primitive functions f to use, but simply including all of them will do for the purposes of this text. Terms are identified up to α-equivalence, as usual, except where otherwise specified. The set of all terms is denoted Λ, and the set of closed terms is denoted Λ^0. Some abbreviations will be used when writing PPCF terms: where infix operators like $M + N$ are used, it formally refers to terms like $\underline{+}(M, N)$, and an if statement with a boolean condition other than $M < 0$, $\mathsf{if}(\phi(M_1, \dots), N_1, N_2)$ is shorthand for $\mathsf{if}(\underline{\phi'}(M_1, \dots) < 0, N_1, N_2)$, where ϕ' is some real function which is negative where the condition ϕ is true.

Only those terms which are well-typed will be considered, with the types as given below in a natural deduction style. A judgement of the form $\Gamma \vdash M : A$ means that the term M has type A, assuming the free variables have the types given by the environment Γ, which is an unordered list of type assignments of varibles $x : B$. The set of judgements that are provable is the smallest set of judgements which contains the conclusion of any typing rule all of whose premises it contains. The rules are written with premises above the line, conclusion below the line and side-conditions (which constrain the values the meta-variables can take) to the side.

$$\frac{}{\Gamma; x : A \vdash x : A} \qquad \frac{\Gamma; x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B} \qquad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash M\, N : B}$$

$$\frac{\Gamma \vdash M : (A \to B) \to (A \to B)}{\Gamma \vdash \mathsf{Y}M : (A \to B)} \qquad \frac{\Gamma \vdash M : \mathsf{R} \quad \Gamma \vdash N_1 : A \quad \Gamma \vdash N_2 : A}{\Gamma \vdash \mathsf{if}(M < 0, N_1, N_2) : A}$$

$$\frac{}{\underline{r} : \mathsf{R}} \qquad \frac{}{\Gamma \vdash \mathsf{sample} : \mathsf{R}} \qquad \frac{\Gamma \vdash M_1 : \mathsf{R} \quad \dots \quad \Gamma \vdash M_n : \mathsf{R}}{\Gamma \vdash \underline{f}(M_1, \dots, M_n) : \mathsf{R}}\ (f : \mathbb{R}^n \to \mathbb{R})$$

Figure 2.1: Typing rules of PPCF

To define the CBV reduction relation, let *evaluation contexts* be of the form:

$$\begin{aligned} E &:= [\,] \mid E\, M \mid V\, E \mid \underline{f}(\underline{r}_1, \dots, \underline{r}_{k-1}, E, M_{k+1}, \dots, M_n) \\ &\quad \mid \mathsf{Y}E \mid \mathsf{if}(E < 0, M_1, M_2) \end{aligned}$$

then a term reduces if it is formed by substituting a redex in a context i.e. the reduction relation $\to$ consists of the cases

$$\begin{aligned}
E[(\lambda x.M)V] &\to E[M[V/x]] \\
E[\underline{f}(\underline{r}_1,\ldots,\underline{r}_n)] &\to E[\underline{f(r_1,\ldots,r_n)}] \\
E[\mathsf{Y}\lambda x.M] &\to E[\lambda z.M[(\mathsf{Y}\lambda x.M)/x]z] \quad \text{where } z \text{ is not free in } M \\
E[\mathsf{if}(\underline{r}<0, M_1, M_2)] &\to E[M_1] \text{ where } r<0 \\
E[\mathsf{if}(\underline{r}<0, M_1, M_2)] &\to E[M_2] \text{ where } r\geq 0 \\
E[\mathsf{sample}] &\to E[\underline{r}] \text{ where } r\in[0,1].
\end{aligned}$$

We write $\to^*$ for the reflexive, transitive closure of $\to$. Every closed term either is a value or reduces to another term.

The reduction rule for Y is a little counter-intuitive, since it is much more complicated than the simpler (η and β equivalent) rule $E[\mathsf{Y}M] \to E[M(\mathsf{Y}M)]$. The reason for the additional complexity is that, given the call-by-value reduction order, this simpler rule would never terminate. If the term M reduces to a value V, then $\mathsf{Y}M \to M(\mathsf{Y}M) \to^* V(\mathsf{Y}M) \to V(M(\mathsf{Y}M)) \to^* \cdots$. The η-expanded term $\lambda z.M[(\mathsf{Y}\lambda x.M)/x]z$ is always a value, so it is not reduced any further unless it is actually necessary. Ensuring this η-expansion is well-typed is the reason the type of M in the typing rule for $\mathsf{Y}M$ is $(A\to B)\to(A\to B)$ rather than simply $A\to A$. Also, by including the substitution of $\mathsf{Y}\lambda x.M$ for the variable x in the Y-reduction step rather than having that be a separate reduction step (i.e. $E[\mathsf{Y}M]\to E[\lambda z.M(\mathsf{Y}M)z]$), it avoids needing to reduce the $\mathsf{Y}\lambda x.M$ in the reduct at all in cases where it is deleted by being in the branch of an if construct that is not taken.

These two complications, the η-expansion and β-reduction built in to Y-reduction, are redundant as either one on its own would be sufficient to ensure the correct reduction order (with minor insignificant differences). In retrospect, including only the η-expansion would have simplified some of the proofs, but I will be sticking with the rule given above which is more similar to the one found in [36]. The simpler rule $\mathsf{Y}M\to M\mathsf{Y}M$ is used in [20], in contrast, but they have to include a let construct to deal with the non-termination this produces instead.

2.4.1 Call-By-Value vs. Call-By-Name

One significant choice to make in defining the semantics of a probabilistic lambda calculus is whether it is call-by-value or call-by-name. In pure lambda calculus, these may differ in that some terms may terminate in one but not the other, but for terms that do terminate, they reach the same values because of confluence (at least for terms of base type, since values of other types may not necessarily be normal forms). In probabilistic lambda calculus however, although there is a limited confluence result that applies when restricting only to a certain

subset of reduction strategies (Theorem 4.1), CBV and CBN semantics are still significantly different.

The differences in the semantics of CBN PPCF are that the forms of evaluation context VE and $\mathsf{Y}E$ are not allowed, the β-reduction rule $E[(\lambda x.M)V] \to E[M[V/x]]$ is replaced with $E[(\lambda x.M)N] \to E[M[N/x]]$, and the Y-reduction rule is replaced with $E[\mathsf{Y}M] \to E[M\mathsf{Y}M]$.

Example 2.4.1. Consider the term $(\lambda x.x - x)\ \mathsf{sample}$. Under the CBV semantics, this reduces as

$$\begin{aligned} & (\lambda x.x - x)\ \mathsf{sample} \\ \to\ & (\lambda x.x - x)\ \underline{r} \\ \to\ & \underline{r} - \underline{r} \\ \to\ & \underline{0} \end{aligned}$$

whereas under the CBN semantics, it reduces as

$$\begin{aligned} & (\lambda x.x - x)\ \mathsf{sample} \\ \to\ & \mathsf{sample} - \mathsf{sample} \\ \to\ & \underline{r} - \mathsf{sample} \\ \to\ & \underline{r} - \underline{t} \\ \to\ & \underline{r - t} \end{aligned}$$

where r and t are real numbers in the interval $[0, 1]$. The two versions are therefore inequivalent.

When a random term is duplicated, the same samples or independent samples may be used in the copies. With CBV, the same samples are used by default, but it is possible to get multiple independent samples by wrapping the random term in a λ, for example, $(\lambda x.\ x\underline{0} - x\underline{0})\ (\lambda y.\mathsf{sample})$ produces in the CBV semantics the same distribution as the term above does under CBN, because $\lambda y.\mathsf{sample}$ is already a value so it is substituted in to $\lambda x.\ x\underline{0} - x\underline{0}$ unchanged. Because CBV can imitate CBN's behaviour in this way, it is possible to translate any term in CBN PPCF into a term in CBV PPCF that behaves the same way. In CBN, the default behaviour is instead to use independent samples, and there is not even any way to avoid this. In CBN PPCF, as defined so far, it is impossible for multiple copies of the same sample to be used, which makes CBN less expressive than CBV.

For this reason, an extra construct is usually added to CBN probabilistic languages to make duplicating random values possible. It has the rules

$$\begin{aligned} \textit{terms}\ M, N &:= \cdots \mid \mathsf{let}\ x = M\ \mathsf{in}\ N \\ \textit{contexts}\ E &:= \cdots \mid \mathsf{let}\ x = E\ \mathsf{in}\ N \\ \frac{\Gamma \vdash M : A \quad \Gamma; x : A \vdash N : B}{\Gamma \vdash \mathsf{let}\ x = M\ \mathsf{in}\ N : B} & \\ E[\mathsf{let}\ x = V\ \mathsf{in}\ N] &\to E[N[V/x]]. \end{aligned}$$

With this addition, we can translate the CBV meaning of the term above as $\mathsf{let}\ x = \mathsf{sample}\ \mathsf{in}\ x - x$, and indeed, any term in CBV PPCF can be translated into CBN PPCF with this extension, making the two versions equally powerful.

While the two versions of PPCF are largely interchangeable, the CBV version has the same expressive power without the added complexity of including the $\mathsf{let}\ \ldots\ \mathsf{in}\ \ldots$ construct (at the cost of making Y's reduction rule more complicated), so I will be using the CBV version from here on except where otherwise noted.

2.4.2 Trace Semantics

The relation $\to$ allows sample to reduce to any number in $[0, 1]$. This defines which results are possible, but to specify the exact probabilities, an additional argument is needed to determine the outcome of random samples. Let $I := [0, 1] \subset \mathbb{R}$, and define $\mathbb{S} := I^{\mathbb{N}}$, with the σ-algebra $\Sigma_{\mathbb{S}}$ and probability measure $\mu_{\mathbb{S}}$ being the product of $\aleph_0$ copies of the Borel σ-algebra and the uniform distribution on I respectively. The maps $\pi_h : \mathbb{S} \to I$ (projecting to the first element) and $\pi_t : \mathbb{S} \to \mathbb{S}$ (popping the first element) are then measurable. Following [16], we call the probability space $(\mathbb{S}, \Sigma_{\mathbb{S}}, \mu_{\mathbb{S}})$ the *entropy space*, and elements of $\mathbb{S}$ *traces*.

Definition 2.1. Define a *skeleton* to be a term but, instead of having real constants $\underline{r}$, it has a placeholder X, so that each term M has a skeleton $Sk(M)$, and each skeleton S can be converted to a term $S[r]$ given a vector r of n real numbers to substitute in, where n is the number of occurrences of X in S. The notions of value, evaluation context, redex and so on all extend to skeletal versions in the obvious way.

$$\begin{aligned}
Sk(\lambda x.M) &:= \lambda x.Sk(M) \\
Sk(\underline{r}) &:= \mathsf{X} \\
Sk(x) &:= x \\
Sk(MN) &:= Sk(M)Sk(N) \\
Sk(\underline{f}(M_1, \ldots, M_n)) &:= \underline{f}(Sk(M_1), \ldots, Sk(M_n)) \\
Sk(\mathsf{Y}M) &:= \mathsf{Y}Sk(M) \\
Sk(\mathsf{if}(M < 0, N_1, N_2)) &:= \mathsf{if}(Sk(M) < 0, Sk(N_1), Sk(N_2)) \\
Sk(\mathsf{sample}) &:= \mathsf{sample}
\end{aligned}$$

Following [8, 48, 20], the σ-algebra on Λ is defined by identifying Λ with $\bigcup_{m \geq 0} \left(\mathsf{Sk}_m \times \mathbb{R}^m\right)$, where Sk_m is the set of skeletons containing m occurrences of X. Thus identified, we give Λ the sum σ-algebra of the product σ-algebras of the discrete σ-algebras on Sk_m and the Borel σ-algebras on $\mathbb{R}^m$. Note that the connected components of Λ have the form $\{S\} \times \mathbb{R}^m$, with S ranging over Sk_m, and m over $\mathbb{N}$.

The one-step reduction of the *trace-based* (or *sampling-style*) *operational*

semantics [34, 8, 16] is given by the function $\mathrm{red} : \Lambda^0 \times \mathbb{S} \to \Lambda^0 \times \mathbb{S}$ where

$$\mathrm{red}(M,s) := \begin{cases} (E[N], s) & \text{if } M = E[R],\ R \to N \text{ and } R \neq \mathsf{sample} \\ (E[\underline{\pi_h(s)}], \pi_t(s)) & \text{if } M = E[\mathsf{sample}] \\ (M, s) & \text{if } M \text{ is a value} \end{cases}$$

The result after n steps is then simply

$$\mathrm{red}^n(M,s) = \underbrace{\mathrm{red}(\ldots \mathrm{red}(}_{n} M, s) \ldots)$$

and the limit red^∞ can then be defined as a partial function as $\lim_{n\to\infty} \mathrm{red}^n(M,s)$ whenever that sequence becomes constant by reaching a value. A term M terminates for a trace s if the limit $\mathrm{red}^\infty(M,s)$ is defined.

The reduction function is measurable, and the set of values is measurable, therefore the set of s such that M terminates at s within n steps is measurable for any n, therefore $\{s \mid M \text{ terminates at } s\}$ is measurable [8, 36]. We say that a term M is *almost surely terminating* (AST) just if $\mu_{\mathbb{S}}(\{s \mid M \text{ terminates at } s\}) = 1$.

The example used earlier, of flipping a coin repeatedly until it lands on heads, can be encoded in PPCF as

$$(\mathsf{Y}\lambda f, x.\mathsf{if}(\mathsf{sample} - 0.5 < 0, \underline{0}, fx))\underline{0}$$

which terminates on the set $\mathbb{S} \setminus [0.5, 1]^{\mathbb{N}}$ which has measure 1, therefore this term is AST as expected.

2.5 Other styles of semantics

In addition to the distinction of CBV vs. CBN semanticses which are not equivalent, there are many options for how exactly to describe the semantics that are not such significant differences, but are just different equivalent ways of looking at the same thing. For one thing, using traces is not the only way to define the semantics of a probabilistic language. Having a probability space ($\mathbb{S}$) on which to define other random variables is convenient, but it is also possible to instead define the probabilities of outcomes directly, without referring to a probability space. This is the distribution-based semantics[48]. Its version of the reduction relation red_d is a measurable function from Λ to probability measures on Λ, defined by

$$\begin{aligned} \mathrm{red}_d(R)(A) &= \mathbf{1}_A(R') \text{ if } R \to R' \text{ and } R \text{ has a redex other than } \mathsf{sample} \\ \mathrm{red}_d(V)(A) &= \mathbf{1}_A(V) \\ \mathrm{red}_d(E[\mathsf{sample}])(A) &= \mu_{\mathbb{R}}(\{x \mid E[\underline{x}] \in A\}) \end{aligned}$$

Functions of this sort, from a measurable space to probability measures on another measurable space, are known as Markov kernels, and they can be

composed together using the formula $(f \cdot g)(x)(A) = \int f(y)(A)\ \mathrm{d}g(x)(y)$. The n-step function red_d^n can then be derived. If A is a set of values, $\mathrm{red}_d^n(M)(A)$ is monotonic increasing, and $\mathrm{red}_d^\infty(M)(A)$ can be defined as its limit. Terms which are not values are not included, even if their probabilities have well-defined limits.

The distribution-based semantics is equivalent to the trace-based semantics in the sense that for all $A \subset \Lambda$ measurable and $n \in \mathbb{N} \cup \{\infty\}$, $\mu_{\mathbb{S}}(\{s \mid \mathrm{red}^n(M, s) \in A\}) = \mathrm{red}_d^n(M)(A)$. This is proven for a somewhat different language in [8, Theorem 3] (most notably, their language is untyped and can include multiple primitive distributions), but the proof does not depend on the details of the reduction relation anyway, only that it is the same for the trace-based and distribution-based semantics, so the proof applies just as well to a variety of languages including PPCF.

The way probabilities are represented is not the only way in which the semantics can vary. There is also the distinction of operational versus denotational semantics.[48] Everything discussed so far has been operational, meaning the meaning of a program is defined by following its reduction step-by-step to a value (or forever if it's non-terminating). Denotational semantics instead defines the meaning of a program in terms of its component parts, by structural induction on the proof that it is well-typed. Each type A is assigned an interpretation $[\![A]\!]$, which is an object of a cartesian-closed category, meaning the category contains products $X \times Y$ and exponentials X^Y. This amount of structure in the category is sufficient to interpret the plain lambda calculus without probabilities, and the construction for this is standard[15]:

$$\begin{aligned} [\![A \to B]\!] &= [\![B]\!]^{[\![A]\!]} \\ \text{If } \Gamma \vdash M : A \to B, \Gamma \vdash N : A, \text{ then } [\![MN]\!] &= ev \circ \langle [\![M]\!], [\![N]\!] \rangle \\ \text{If } \Gamma; x : A \vdash M : B, \text{ then } [\![\lambda x.M]\!] &= \lambda([\![M]\!]) \end{aligned} \tag{2.1}$$

where ev and λ are the product-exponential adjunction required in a cartesian closed category. To interpret the other constructs in PPCF takes additional structure in the category. For some extensions of lambda calculus, it can be useful to define the semantics using a monad in the category that corresponds to the additional feature[42], analogously to the use of monads in languages like Haskell to implement extra language features that would not be possible in pure lambda calculus. Given a monad P with operations $\mu : P \circ P \to P$ and $\eta : I \to P$, strength t and costrength t', the denotation of a program satisfying the typing judgement $x_0 : A_0; \ldots; x_n : A_n \vdash M : B$ is then a morphism $\Pi_i [\![A_i]\!] \to P([\![B]\!])$, and the semantics of the plain lambda calculus features can be given by

$$\begin{aligned} [\![A \to B]\!] &= (P[\![B]\!])^{[\![A]\!]} \\ [\![MN]\!] &= \mu_{[\![B]\!]} \circ \mu_{P[\![B]\!]} \circ P^2 ev \circ P t_{[\![A \to B]\!], [\![A]\!]} \circ t'_{[\![A \to B]\!], P[\![A]\!]} \circ \langle [\![M]\!], [\![N]\!] \rangle \\ [\![\lambda x.M]\!] &= \eta_{[\![A \to B]\!]} \circ \lambda([\![M]\!]) \end{aligned} \tag{2.2}$$

then of course any additional language features' denotations will depend on the details of the category and monad. As described in [42], separating out language features as monads in this way permits more modularity in the language design than would otherwise be available in denotational semantics.

For a distribution-based denotational semantics, it would be natural to try using this construction with the Giry monad[24], which takes each object to the set of probability distributions on that object. In the category of measurable spaces, this makes sense as a functor because the set of probability distributions on a measurable space itself has a sensible σ-algebra. Unfortunately the category of measurable spaces and the corresponding Kleisli category, also known as the category of kernels, are not cartesian-closed so an alternative more complicated category is needed[20].

Various options for this exist. For example, [20] defines a denotational semantics using a category they define, $\mathbf{Cstab}_m$, consisting of measurable cones with stable measurable functions. Very roughly speaking, a cone is like a normed vector space where scalar multiplication only includes positive scalars, where all of the vectors are positive as well, and a stable function is one whose finite differences to all orders are non-negative (so it's weakly increasing, convex, etc.). For any measurable space, the set of measures forms a cone (since measures can be added together and multiplied by non-negative scalars). The basic lambda calculus constructs are then interpreted as 2.1 as usual, and the base type R can then be interpreted as the cone of measures on $\mathbb{R}$, then $[\![\mathsf{sample}]\!](\gamma)$ in this semantics is simply the uniform probability distribution on $[0, 1]$.

Unfortunately the way the cartesian-closed structure of $\mathbf{Cstab}_m$ works implies that the semantics is call-by-name. A $\mathsf{let}\ x = M\ \mathsf{in}\ N$ construct is included, but it is limited to cases where the type of M is the base type, R, which limits the expressiveness of this version of PPCF. The interpretation of a term $[\![\mathsf{let}\ x = M\ \mathsf{in}\ N]\!]_\gamma$ is defined as taking the value $\int [\![N]\!]_{\gamma;x=\delta_r}(U)\ [\![M]\!]_\gamma(\mathrm{d}r)$ on a measurable set U, where δ_r is a discrete distribution having a value of 1 at r only, and the notion of integration used here has to be specially defined for the context of $\mathbf{Cstab}_m$, and does not readily generalise to cases where the type of M (and thus the domain of integration) is anything more complicated than $\mathbb{R}$.

Other versions of denotational distribution-based semantics, such as [52], avoid the CBN issue, though that is for a somewhat different language in other ways.

It is also possible to define a trace-based version of denotational semantics, but it is not as simple as just using the reader monad $X \to X^S$ for some trace space S, as this would lead to incorrectly duplicated use of random values.

Example 2.5.1. In the program

$$(\lambda x.\ x\underline{0} - x\underline{0})(\lambda y.\ \mathsf{sample})$$

all of the other versions of the semantics so far would produce two separate samples, one for each copy of x, then subtract them, giving a random result in

the range $[-1, 1]$. If, however, $[\![\mathsf{sample}]\!]$ is defined to take a fixed sample from the sample space argument (perhaps using a different one for each occurrence of sample in the input program), the same sample will be used both times and the program will always return 0. Not only does this mean the semantics would not be equivalent to the other versions, it also makes the language considerably less powerful if the number of independent random samples any program can use is bounded by the size of the program.

One solution is to simply add an argument to the sampling construct that can be used as an address, but this is unsatisfactory because it requires additional unnecessary programmer effort and is likely to lead to bugs. A better solution, like that used in [55] and [3], is to label the samples automatically based on the structure of where they end up used. [55] gives a translation to programs with explicit labels rather than a denotational semantics though, and the semantics presented in [3] does not actually properly avoid re-using random variables (it fails in the way shown in Example 2.5.1, and the example program from Section 9 of [3] is incorrect for the same reason), so neither of these is satisfactory as a denotational trace semantics.

Another option is to label samples by the order in which they are used, as is done in [26]. [26] uses a more obscure structure for the sample space rather than simply a probability space, and also uses a rather different version of PPCF. The details of the structure they use instead of a probability space are not relevant here, but roughly speaking, their monad $v\mathbf{R}_1$ is similar to $X \mapsto (X \times (\mathbb{N} \cup \{\infty\}))^{\mathbb{S}}$, with monad operations, phrased in the language of lambda-calculus rather than category theory for convenience, $\eta = \lambda xs.(x, 0)$ and $\mu = \lambda fs.\mathsf{let}\ (f', n) = fs, (x, m) = f'd(n, s)\ \mathsf{in}\ (x, n + m)$ (and in the case where n is infinite, a sort of limit is used). Then if $[\![A]\!](s) = (x, n)$, the term A evaluated with trace s uses the first n samples and returns x.

There is something a little unnatural however about labelling the samples by the order in which they are used like this in the context of a denotational semantics, which complicates its use for proving program equivalences. Even quite a simple rearrangement of a term like $(\lambda xy.M)XY) \to (\lambda yx.M)YX$ leads to quite a complicated rearrangement of the order in which the samples are used, complicating the correspondence between the interpretations of these terms. It seems likely that there is a way to do better, labelling the samples in some way related to the structure of the term, but I am not aware of any such semanticses in the literature.

All four combinations of operational/denotational and trace/distribution-based can give equivalent descriptions of program semantics in different ways, but because the presence of a probability space in which additional random variables can be defined is convenient, the denotational trace semantics is particularly complicated, and the method of ranking functions requires the sequence of reduction steps that only operational semantics provides, I will be using the operational trace semantics as the canonical semantics for the rest of this thesis.

2.6 Inference Algorithms

In a language that has random sampling but no conditioning, there is no particular difficulty in executing the programs. An interpreter can simply generate a random number (from a pseudo-random number algorithm or hardware randomness source) whenever a random sample is taken. If the whole distribution of results is required, rather than just one sample, it can be approximated by running the program many times. A fully probabilistic language including a conditioning operator as well requires a more complicated approach.

2.6.1 Rejection Sampling

If hard conditioning is used, it is possible to use the very simple method of rejection sampling. The program is executed, and if one of the conditions fails, it gives up on the execution run and tries again from the start. This works, and conceptually it's very simple, but it's far from ideal. With no particular effort made to prefer runs with high success probability, it can end up rejecting an excessive number of runs, requiring more time to compute than more efficient algorithms. Also, it is only able to deal with hard conditioning, not soft conditioning. As explained in Section 2.1, it is possible to simulate soft conditioning using hard conditioning, but only if the likelihoods are bounded above, and the less tight the bound is, the worse the execution time.

Recall Example 2.1.3

```
x := sample_uniform(0,1)
y := sample_uniform(0,1)
observe x + y < 1
score 1/(x+y)
```

which uses a `score` construct with an unbounded argument. Even if the error in the posterior distribution caused by putting some artificial bound b on the likelihood is accepted, and `score p` is implemented as `observe sample_bernoulli(p/b)`, the probability of rejecting a run (assuming $b > 1$) becomes $\frac{1}{2}(1 + (1 - \frac{1}{b})^2) = 1 - O(\frac{1}{b})$. There is therefore a tradeoff, with more accurate results requiring higher values of b, but higher acceptance probability (and therefore better speed) requiring lower values. For more complicated programs involving conditioning multiple times, it becomes even worse.

2.6.2 Metropolis-Hastings Monte-Carlo

Thankfully, better algorithms exist. The Metropolis-Hastings algorithm is based on the fact that if one run of the program had a high likelihood, trying similar runs is likely to also give high likelihoods, and the chance of having to reject a run can be reduced. Metropolis-Hastings is useful for generating random samples in a variety of circumstances where the ratios of the probabilities of any two states can be computed, but the overall scale factor may be unknown,

for example in statistical mechanics for simulating microscopic physical systems at non-zero temperatures[41, 38]. The general algorithm has the form

$s \leftarrow$ an arbitrary state
loop
  $s' \leftarrow f(s)$
  if a random boolean with probability $a(s, s')$ **then**
    $s \leftarrow s'$
  end if
  output s
end loop

The loop can be allowed to continue until enough samples have been collected for whatever purpose they are needed for. The function f produces random values, and the probability that it produces an output s' given input s is the *transition probability* $t(s, s')$. The function a is the *acceptance probability*. The desired result is that the states have probabilities proportional to some function p, and in order to ensure that this is the case, the functions t and a should satisfy the detailed balance condition for all states s, s':

$$\frac{p(s')}{p(s)} = \frac{t(s, s')a(s, s')}{t(s', s)a(s', s)}. \tag{2.3}$$

Additionally, there is an ergodicity condition, that for any pair of possible states, there is some possible sequence of steps with non-zero probability that gets from one to the other (and also there is some minimum N such that it is possible to make this transition in any number of steps $n > N$, although this stronger ergodicity condition is in most cases implied by the first part of the ergodicity condition along with detailed balance). If both of these conditions are satisfied, the Markov chain of states thus generated tends towards a steady state with probabilities proportional to p. Even if the stronger version of the ergodicity condition is not satisfied, at worst the probability distribution at each step will cycle through multiple distributions, the average of which is proportional to p.

The fact that a probability distribution proportional to p is a fixed point can be easily seen from the detailed balance condition. If the probability $\mathbb{P}[S_n = s]$ that the state at step n is s is given for all s by $cp(s)$ for some constant c, then

the probability of that state at the next time step is

$$\begin{aligned}
&\mathbb{P}[S_{n+1} = s] \\
=&\mathbb{P}[S_n = s \text{ and the transition is rejected}] \\
&+ \sum_{s'} \mathbb{P}[S_n = s' \text{ and the transition } s' \to s \text{ occurs}] \\
=&\mathbb{P}[S_n = s] \left(1 - \sum_{s'} \mathbb{P}[\text{the transition } s \to s' \text{ occurs} \mid S_n = s]\right) \\
&+ P[S_n = s']t(s', s)a(s', s) \\
=&cp(s) \left(1 - \sum_{s'} t(s, s')a(s, s')\right) + cp(s')t(s', s)a(s', s) \\
=&cp(s) + c\sum_{s'} (p(s')t(s', s)a(s', s) - p(s)t(s, s')a(s, s')) \\
=&cp(s).
\end{aligned}$$

The Markov process can be expressed as a matrix of transition probabilities, where the (s, s')th element of the matrix is $\mathbb{P}[S_{n+1} = s' \mid S_n = s]$. The fixed point probability distributions are then eigenvectors of this matrix. The detailed balance condition implies that the transition matrix can be expressed as XP where X is some symmetric matrix and P is a diagonal matrix with entries $P_{ss} = p(s)$. If two vectors (probability distributions) u and v have distinct eigenvalues $XPu = \lambda_u u, XPv = \lambda_v v$, then $\lambda_u u^T Pv = (XPu)^T Pv = u^T PXPv = u^T P(\lambda_v v) = \lambda_v u^T Pv$ therefore eigenvectors with distinct eigenvalues satisfy the orthogonality condition $u^T Pv = 0$. Because of the ergodicity condition and the fact that probabilities (and therefore the entries of XP) are non-negative, for sufficiently large n all of the entries of $(XP)^n$ are strictly positive, therefore the vector with the largest eigenvalue has all positive entries (or equivalently all negative since it is only determined up to a scale factor). For any two positive vectors, $u^T Pv > 0$ therefore p is the only positive eigenvector therefore it has the greatest eigenvalue (and is non-degenerate). For any starting vector v, the vectors $(XP)^n v$, which correspond to the probability distributions after n steps, therefore converge to p (unless there is a large negative eigenvalue, but that is impossible because it would require $(XP)^n$ to have negative coefficients, and the case of eigenvalue -1 exactly is excluded by the ergodicity condition). This proof only applies in the case where there are finitely many states, and I have skipped over many of the details. A more thorough proof can be found in [47].

The algorithm generates not just a single sample, but a sequence of them. The samples from this sequence are correlated to each other, with the correlation decreasing the greater the number of iterations that pass between the two samples. It may be necessary to take these correlations into account in interpreting the results. If the distributions involved are not discrete, then $t(s, -)$ and p are instead probability density functions, and any differences in the mea-

sures with respect to which these densities are defined cancel out so long as the same definition is used for t and p, but the result is essentially the same.

The acceptance probability can be taken to be

$$a(s, s') = \begin{cases} 1 & \text{if } p(s')t(s', s) > p(s)t(s, s') \\ \frac{p(s')t(s',s)}{p(s)t(s,s')} & \text{otherwise} \end{cases}$$

which ensures that the detailed balance condition is satisfied whatever p and t are. It is merely necessary that the probability distribution $t(s, -)$ is possible to both sample from and compute the density of (or at least it is necessary to compute $\frac{t(s,s')}{p(s')}$ which may actually be simpler). Subject to this and the ergodicity constraint, t (and therefore f) may be taken to be whatever is convenient, the specific form affects only efficiency, not correctness.

In statistical mechanics, the states are the states of the physical system and the function p is $e^{-\beta E}$. In probabilistic program inference, the states are the execution runs of the program and p gives the weight of each run, determined both by the prior probabilities of the samples taken in that run and the conditioning constructs that occur. The runs can be parameterised by the random samples taken, since given the same random samples, the program will always execute the same way. This can be simply a list of all of the random samples used in order, or a more complicated method can be used to attempt to make sure that the same samples are used in the same ways on different runs.

To be more explicit, the Metropolis-Hastings algorithm as it applies to interpreting a probabilistic program P is as follows. First, run the program P, keeping track of the random samples used along the way (these are the trace, the whole of which is a single sample of MH) and the values used in any soft conditioning operators (the product of these is the weight, which corresponds roughly to the function p). If the weight is 0, try again until it isn't. Next, for the main loop of MH, vary the trace by some random amount (the transition t), and run P with this new trace, i.e. run P, but instead of picking a new random number each time it takes a sample, use one of the numbers recorded in the trace. If the new run is longer than the old one and the samples in the trace run out, new fresh random samples can be added to the end. Conceptually, the trace is infinite, but only a finite portion of it is ever stored at one time. The rest, having never been involved in the algorithm yet, has a probability distribution that takes a simple known form (perhaps the same as the initial distribution, depending on the implementation details). The weight of this new program run is computed and compared to the old weight, and based on this the transition is either accepted or rejected, then the process repeats starting with the new trace. Be careful to keep the states used in MH, which are entire traces or runs of P, distinct in your mind from the states that occur within the execution of P (its variables, program counter etc.), as they are entirely different.[1] There may

[1] Funnily enough, the same confusion can occur in the application of MH and similar algorithms in physics and chemistry, despite the term being different. There, the MH (or diffusion monte-carlo) states are sometimes called "particles", and physical particles are also involved in the system being simulated.

well be opportunities to re-use data from one run of P to the next, and various other optimisations, but this is the basic version of the algorithm.

Assuming the proposal distribution is reasonably well-behaved, with the distribution $f^n(i)$ having a finite propability density relative to i, the prior distribution, and further assuming that the program itself doesn't almost certainly return a weight of 0, the MH algorithm as described above is AST iff the program it's evaluating is AST (or at least, each step of MH is AST, as MH as a whole is designed to run forever), since the distribution of traces the program is run on at step n is a convex linear combination of $f^m(i)$ for $0 \leq i \leq n$. If the proposal function f violates this condition, the detailed balance condition can't be satisfied anyway, and if the program violates the condition of having a non-zero weight, its posterior distribution isn't even well defined, so in all reasonable cases, if a program is AST, it's possible to evaluate it with MH. If the weights of the program are such that taking them into account may seem to make the program non-terminating, this may just result in MH not converging.

In order for the inference algorithm to be efficient, it is crucial that the acceptance probability be kept high. Every step, whether or not it is accepted, requires executing the program (or at least part of it) in order to compute the probability of acceptance, so steps that are not accepted are wasted effort. Note that it is still necessary to count them as part of the output otherwise the probabilities would be biased, they're just less useful for calculating an accurate distribution of the outputs since they aren't independent samples. The acceptance probability being high requires that transitions mostly occur between states of similar p, which generally means they are chosen to be similar states. On the other hand, in order to get a good sample of the outputs it is necessary to sample a wide range of states, so the steps should not be taken to be too small either. The issue of acceptance probability is similar to the simpler case of rejection sampling, and indeed rejection sampling using a distribution $d(s)$ for the guesses is very similar to Metropolis-Hastings with transition probabilities $t(s, s') = d(s')$ that do not depend at all on the initial state and $a(s, s') = \frac{p(s')}{bd(s')}$ where b is the bound used on the ratio $\frac{p(s')}{d(s')}$ (which in the case of probabilistic inference corresponds to the weight of the run, given by all of the `score`s) although it is not exactly the same because each sample ends up getting counted a random number of times. Since it is not hard to pick a t much better than this, Metropolis-Hastings can be seen to be a considerable improvement on rejection sampling.

Although it is possible to simulate soft conditioning using hard conditioning, at least in some limited cases, doing so can have a negative effect on the efficiency of the inference algorithm. The additional random sample used generates additional noise, requiring additional program runs to average out. For this reason, it can be useful for a probabilistic language to have a dedicated soft conditioning construct even if hard conditioning would do.

2.6.3 Variational Inference

Another approach to inference that is sometimes used is variational inference. In this approach, given a complicated probabilistic program whose behaviour we want to know, a simpler program whose behaviour can be seen more directly is generated (for example, the simplified program might just be the original program but with any conditioning removed) and its parameters are optimised to make the distribution of its results as close as possible to the distribution of the original program. It is convenient to define "as close as possible" here using the K-L divergence $\sum_x P(s) \log(\frac{P(x)}{Q(x)})$, where Q is the true distribution and P is the estimate. Unlike the rejection sampling or Metropolis-Hastings algorithms, variational inference is deterministic, and its results are only approximately correct. M-H also has errors, but they are random errors, and can be made arbitrarily small simply by taking more samples, which is not the case for variational inference.

Chapter 3

Supermartingales and Ranking Functions

One approach to proving that a program terminates is to find a variant, some function from program states to numbers which is bounded below and decreases sufficiently quickly as the program runs that it is impossible for the program to keep running forever. In the context of probabilistic programming, the condition that the variant decreases is replaced by the condition that it decreases on average (with some additional restrictions on how fast it must decrease, to be elaborated on later, in Section 3.3). For imperative probabilistic languages, this approach to proving AST has been used in [22] and [40]. In the context of lambda-calculus based languages like PPCF, the program states are *reachable terms*. The set of reachable terms starting from a given term M is defined as $Rch(M) = \{N \in \Lambda \mid M \to^* N\}$, with the σ-algebra induced as a subset of Λ.

In order to ignore certain zero-probability outcomes, it will be necessary to use the concept of almost complete reachable sets.

Definition 3.1. An *almost complete reachable set* for a term M is a measurable set $C \subset Rch(M)$ such that the probability of reaching any term outside of C starting from M is 0 (formally, $\mu_{\mathbb{S}}(\{s \in \mathbb{S} \mid \exists n, c, s'.\, \mathrm{red}^n(M, s) = (c, s'), c \notin C\}) = 0$).[1]

An almost complete reachable set can roughly be thought of as a set of terms whose complement has measure 0, except for the fact that there is not actually such a measure on $Rch(M)$, just $\mathbb{S}$. The function $s \mapsto \pi_0(\mathrm{red}^n(M, s))$ induces a measure on $Rch(M)$ for each n, but these measures may overlap, and an almost complete reachable set must have measure 1 in all of these measures simultaneously.

In PPCF, the basic version of the definition of a variant (called a *ranking function* in its probabilistic form) is

[1] The fact that this set is measurable follows from red and C being measurable.

Definition 3.2. A ranking function on $M \in \Lambda^0$ is a measurable function $f : Rch(M) \to \mathbb{R}$ such that $f(N) \geq 0$ for all N,[2] and there is some almost complete reachable set C such that

1. $f(E[\mathsf{Y}\lambda x.N]) \geq 1 + f(E[\lambda z.N[(\mathsf{Y}\lambda x.N)/x]z)$ where z is not free in N
2. $f(E[\mathsf{sample}]) \geq \int_I f(E[\underline{x}])\,\mathrm{d}x$
3. $f(E[R]) \geq f(E[R'])$ for any other redex R with $R \to R'$

for all E, R, etc. such that the terms on the left hand sides of these inequalities are in C. We say that the ranking function f is *strict* if there exists $\varepsilon > 0$ such that for all E and $R \to R'$ (with $E[R] \in C$), $f(E[R]) \geq f(E[R']) + \varepsilon$.

Any closed term for which a ranking (respectively, strict ranking) function exists is called *rankable* (respectively, *strictly rankable*). For example, $(\mathsf{Y}\lambda x.x)\,\underline{0}$ is not rankable.

This differs from the original version published in [32] by the introduction of the almost complete reachable set, rather than simply requiring the conditions to be true for every single reachable term. This change was necessary because the original version of Theorem IV.5 (here Theorem 3.7) was incorrect.

It will be demonstrated later that for any rankable term M, if $(M_n)_{n\geq 0}$ is the reduction sequence starting from M (considered as a stochastic process), then $(f(M_n))_{n\geq 0}$ is a supermartingale, and M terminates almost surely, but first, some preliminaries about supermartingales (see e.g. [54, 19] for details).

3.1 Supermartingales

Fix a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and a *filtration* $(\mathcal{F}_n)_{n\geq 0}$, i.e. a sequence of σ-algebras $\mathcal{F}_n \subseteq \mathcal{F}$ such that $\mathcal{F}_n \subseteq \mathcal{F}_{n+1}$ for all n. Recall that a sub-σ-algebra can be used to represent a state of partial information, so a filtration represents information being revealed over time. A sequence of r.v.s $(Y_n)_{n\geq 0}$ is said to be *adapted to* $(\mathcal{F}_n)_{n\geq 0}$ if each Y_n is measurable with respect to the corresponding $\mathcal{F}_n$. A single r.v. T that takes values in $\mathbb{N}\cup\{\infty\}$ is called a *stopping time adapted to* $(\mathcal{F}_n)_{n\geq 0}$ just if $\{T = n\} \in \mathcal{F}_n$, for all n.

Definition 3.3.

1. A sequence of r.v.s $(Y_n)_{n\geq 0}$ adapted to a filtration $(\mathcal{F}_n)_{n\geq 0}$ is a *supermartingale* if for all $n \geq 0$, Y_n is integrable (i.e. $\mathbb{E}[|Y_n|] < \infty$), and $\mathbb{E}[Y_{n+1} \mid \mathcal{F}_n] \leq Y_n$ a.s. (i.e. for all $A \in \mathcal{F}_n$, $\int_A Y_{n+1}(\omega)\,\mathbb{P}(\mathrm{d}\omega) \leq \int_A Y_n(\omega)\,\mathbb{P}(\mathrm{d}\omega)$).
2. Let $\varepsilon > 0$. Given a stopping time T and a supermartingale $(Y_n)_{n\geq 0}$, both adapted to filtration $(\mathcal{F}_n)_{n\geq 0}$, we say that $(Y_n)_{n\geq 0}$ is a *ε-ranking supermartingale w.r.t. T* if for all n, $Y_n \geq 0$ and $\mathbb{E}[Y_{n+1} \mid \mathcal{F}_n] \leq Y_n - \varepsilon \cdot \mathbf{1}_{\{T>n\}}$ a.s.. A *ranking supermartingale w.r.t. T* is a ε-ranking supermartingale w.r.t. T for some $\varepsilon > 0$.

[2] The fact that the ranking function can be 0 in some cases before it reaches a value, which is different from how it works in some of the references, is necessary to get Theorem 3.5 to work neatly, and the restriction to an almost complete reachable set is necessary for Theorem 3.7.

This notion of ranking supermartingale is a slight generalisation of the original definition in [11, 22], which does not involve an arbitrary stopping time. In their version, T is just the minimum n at which $Y_n = 0$ (although this is using my notation rather than theirs). The separation of T from Y will be useful later, since in the use case of program termination, T will represent the time at which the term reaches a value (which may be infinite if it fails to terminate) and Y_n will represent the rank of the program after n steps of computation, and these are logically separate. Also, keeping them separate, which allows ranking functions to take the value 0 before the program terminates, will allow some theorems such as Theorem 3.5 and Theorem 3.7 to work more neatly. In a ε-ranking supermartingale, each computation step causes a strict decrease in rank of at least ε on average, provided the term in question is not a value, which means that $T > n \Leftrightarrow Y_n > 0$ in this case anyway, but there will be cases involving non-ε-ranking supermartingales where this equivalence does not hold.

Lemma 3.1. *Let $(Y_n)_{n\geq 0}$ be a ε-ranking supermartingale w.r.t. the stopping time T, then $T < \infty$ a.s., and $\mathbb{E}[T] \leq \frac{\mathbb{E}[Y_0]}{\varepsilon}$.*

(Lemma 3.1 is a slight extension of [22, Lem. 5.5], which asserts the same result but for the specific stopping time $T : \omega \mapsto \min\{n \mid Y_n(\omega) = 0\}$.)

Proof. We first prove (by induction on n): for all $n \geq 0$

$$\mathbb{E}[Y_n] \leq \mathbb{E}[Y_0] - \varepsilon\big(\textstyle\sum_{i=0}^{n-1} \mathbb{P}[T > i]\big).$$

The base case is trivial (it requires only that $\mathbb{E}[Y_0]$ exists, which is required in the definition of a supermartingale), and the inductive step follows quite directly from the definition.

$$\begin{aligned}
&\mathbb{E}[Y_{n+1}] \\
&= \mathbb{E}[\mathbb{E}[Y_{n+1} \mid \mathcal{F}_n]] \\
&\leq \mathbb{E}\left[Y_n - \varepsilon \cdot \mathbf{1}_{\{T>n\}}\right] \\
&= \mathbb{E}[Y_n] - \varepsilon\mathbb{P}[T > n] \\
&\leq \mathbb{E}[Y_0] - \varepsilon\big(\textstyle\sum_{i=0}^{n-1} \mathbb{P}[T > i] + \mathbb{P}[T > n]\big) \\
&= \mathbb{E}[Y_0] - \varepsilon\big(\textstyle\sum_{i=0}^{n} \mathbb{P}[T > i]\big)
\end{aligned}$$

It follows that $\sum_{i=0}^{\infty} \mathbb{P}[T > i]$ converges, since the partial sums increase monotonically and are bounded above by $\frac{\mathbb{E}[Y_0]}{\varepsilon}$; hence we have $\lim_{i\to\infty} \mathbb{P}[T > i] = 0$ and so $\mathbb{P}[T < \infty] = 1$. It then remains to observe: if $T < \infty$ a.s., then $\mathbb{E}[T] = \sum_{i=0}^{\infty} \mathbb{P}[T > i]$. □

Let T and T' be stopping times adapted to $(\mathcal{F}_n)_{n\geq 0}$. The σ-algebra $\mathcal{F}_T$ (which you can think of as measurable sets prior to T) is defined as

$$\mathcal{F}_T := \{A \in \mathcal{F} \mid \forall i \geq 0\,.\, A \cap \{T \leq i\} \in \mathcal{F}_i\}$$

and if $T \leq T'$, then $\mathcal{F}_T \subseteq \mathcal{F}_{T'}$.

The following is an iterated version of Doob's well-known Optional Sampling Theorem (see, e.g., [2, §6.7] and [22, Thm. 7.2]).

Theorem 3.1 (Optional Sampling)**.** *Let $(X_n)_{n\geq 0}$ be a supermartingale, and $(T_n)_{n\geq 0}$ a sequence of increasing stopping times, all adapted to filtration $(\mathcal{F}_n)_{n\geq 0}$, then $(X_{T_n})_{n\geq 0}$ is a supermartingale adapted to $(\mathcal{F}_{T_n})_{n\geq 0}$ if one of the following conditions holds:*

1. *each T_n is bounded i.e. $T_n < c_n$ where c_n is a constant*
2. *$(X_n)_{n\geq 0}$ is uniformly integrable.*

3.1.1 Ranking functions and supermartingales

Henceforth, fix the probability space $(\mathbb{S}, \Sigma_{\mathbb{S}}, \mu_{\mathbb{S}})$, and a closed PPCF term M (The fact that it is closed only matters because closed terms terminate at values, and it is more convenient to refer to values rather than general irreducible terms). For $n \geq 0$, define the random variables

$$M_n(s) := \pi_0(\mathrm{red}^n(M, s))$$
$$T_M(s) := \min\{n \mid M_n(s) \text{ is a value}\}$$

and the filtration $(\mathcal{F}_n)_{n\geq 0}$ where $\mathcal{F}_n := \sigma(M_1, \cdots, M_n)$, i.e. all the information of the samples revealed in executing up to n steps (the samples used can be derived from the terms reached, because after each sample reduction $E[\mathsf{sample}] \to E[r]$, the next term $E[r]$ contains the sample used, r). Thus T_M is the *runtime* of M (and M is AST iff $\mu_{\mathbb{S}}(T_M < \infty) = 1$); we say that M is *positively almost surely terminating* (*PAST*) if $\mathbb{E}[T_M] < \infty$.

Our first result is the following theorem.

Theorem 3.2 (Deriving supermartingales)**.** *If a PPCF term M is rankable (respectively, strictly rankable) by f then $(f(M_n))_{n\geq 0}$ is a supermartingale (respectively, ranking supermartingale w.r.t. stopping time T_M) adapted to $(\mathcal{F}_n)_{n\geq 0}$.*

Proof. For the non-strictly ranking case, take $\varepsilon = 0$. Fix some n. We want to show that the supermartingale condition at step n is satisfied. First, we must partition $\mathbb{S}$ into subsets U_i such that for each i, the sequence of skeletons $Sk(M_m(s))$ for $m \leq n+1$ is constant for all s in U_i (and U_i is also assumed to be maximal subject to this condition). This partition is finite, because although the reduction relation on skeletons is nondeterministic, each skeleton only reduces to at most two others (this being in the case of if-reduction). Call the finite indexing set $\mathcal{I}$. $U_i \in \mathcal{F}_n$, since for each U_i, either M_n has a sample-redex, in which case $Sk(M_{n+1})$ is constant even if M_{n+1} itself isn't, or it doesn't, in which case M_{n+1} follows deterministically from M_n. Also, $f(M_n(s))$ can be expressed as $\sum_{i\in\mathcal{I}}[s \in U_i]f(M_n(s))$ (using the Iverson bracket notation, i.e. $[P]$ is 1 if P is true, or 0 if it is false), therefore the supermartingale condition can be split

into a separate part for each U_i. Take some i, and let S be the common value of $Sk(M_n(s))$ for all $s \in U_i$, then the required condition is

$$\int_A f(M_{n+1}(s))[s \in U_i]\,\mathrm{d}\mu_s(s) \leq \int_A f(M_n(s))[s \in U_i]\,\mathrm{d}\mu_{\mathbb{S}}(s) - \varepsilon\mathbb{P}[M_n \text{ not a value}, s \in U_i \cap A]$$

for $A \in \mathcal{F}_n$, or equivalently, that

$$\int_A f(M_{n+1}(s))\,\mathrm{d}\mu_s(s) \leq \int_A f(M_n(s))\,\mathrm{d}\mu_{\mathbb{S}}(s) - \varepsilon\mu_s(A)[S \text{ is not a value}] \quad (3.1)$$

for $A \in \mathcal{F}_n$ restricted to $A \subset U_i$, for each i. (The integrability condition in this case follows from the non-increasing condition because $f(M_0(s))$ is constant and f is bounded below.)

Let $c = |\{k < n \mid \exists E : M_k(s) = E[\mathsf{sample}]\}|$, so that $\pi_1(\mathrm{red}^n(M, s)) = \overbrace{\pi_t(\cdots(\pi_t}^{c}(s))$ for all $s \in U_i$. Equation (3.1) can then be proven by a case analysis on S. It is of one of the forms V, $E[\mathsf{sample}]$, $E[\mathsf{Y}\lambda x.N]$, or $E[R]$ for some redex R of one of the other forms (β, if or $\underline{f}$). All of the cases except $E[\mathsf{sample}]$ follow simply from the fact that, for M rankable and M_n of one of these forms, $f(M_n) \geq f(M_{n+1})$ (the case that $M_n \notin$ the almost complete reachable set C may be ignored because it happens on a set of probability 0 and therefore does not contribute to the integral).

If $S = E[\mathsf{sample}]$, let $\sigma : U_i \to \mathbb{R}^c$ be a function such that $\forall s \in U_i.M_n(s) = E[\sigma(s)][\mathsf{sample}]$, then it follows that also $M_{n+1}(s) = E[\sigma(s)][s(c)]$. (The two square brackets here first take the skeletal evaluation context E to an evaluation context, then take that to a term.)

For any measurable $g : U_i \to \mathbb{R}$, if $g(s)$ only depends on the prefix of s of length l (denoted $s_{<l}$), a function $\widehat{g} : I^l \to \mathbb{R}$ can be defined such that $\widehat{g}(s_{<l}) = g(s)$, and from the definition of $\mu_{\mathbb{S}}$, it follows that for any $A \subset U_i, A \in \mathcal{F}_n$

$$\int_A g(s)\,\mathrm{d}\mu_{\mathbb{S}}(s) = \int_{A_{<l}} \widehat{g}(x)\,\mathrm{d}^l x.$$

For such an A, $A_{<c+1} = A_{<c} \times I$, since $\mathcal{F}_n = \sigma(M_0, \ldots, M_n)$, and the values of $M_0, \ldots, M_n$ do not depend on $s(c)$ so long as $s \in U_i$. Applying this to

Equation (3.1),

$$\begin{aligned}
&\int_A f(M_{n+1}(s))\ \mathrm{d}\mu_{\mathbb{S}}(s)\\
=&\int_A f(E[\sigma(s)][s(c)])\ \mathrm{d}\mu_{\mathbb{S}}(s)\\
=&\int_{A_{<c+1}} f(E[\widehat{\sigma}(x)][x(c)])\ \mathrm{d}^{c+1}x\\
=&\int_{A_{<c}}\int_I f(E[\widehat{\sigma}(x)][y])\ \mathrm{d}y\ \mathrm{d}^c x\\
\leq&\int_{A_{<c}} f(E[\widehat{\sigma}(x)][\mathsf{sample}]) - \varepsilon\ \mathrm{d}^c x\\
=&\int_A f(E[\sigma(s)][\mathsf{sample}]) - \varepsilon\ \mathrm{d}\mu_{\mathbb{S}}(s)\\
=&\int_A f(M_n(s))\ \mathrm{d}\mu_{\mathbb{S}}(s) - \varepsilon\mu_{\mathbb{S}}(A)
\end{aligned}$$

(Again, the case that $M_n \notin C$ may be ignored inside the integral.) □

In fact a slightly stronger condition can be proven, and will be needed later, because of the fact that $f(E[R]) \geq f(E[R']) + 1$ if $R \to R'$ is a Y-redex.

Corollary 3.1. *If a PPCF term M is rankable by f, then for all $n \geq 0$, $\mathbb{E}\left[f(M_{n+1}) \mid \mathcal{F}_n\right] \leq f(M_n) - \mathbf{1}_{\{\exists E,N.M_n = E[\mathsf{Y}\lambda x.N]\}}$ a.s..*

Proof. The proof is just the same as Theorem 3.2, except that in the case that $S = E[\mathsf{Y}\lambda x.N]$, the extra $+1$ in the Y case of the definition of rankability is used. □

3.1.2 Soundness of rankability

In the rest of this section it will be shown that if a PPCF term is rankable, then it is AST.

Take $M \in \Lambda^0$ and define random variables:

$$\begin{aligned}
T_{-1}(s) &:= -1\\
T_{n+1}(s) &:= \min\{k \mid k > T_n(s), M_k(s) \text{ a value or of form } E[\mathsf{Y}\lambda x.N]\}\\
Y_n(s) &:= f(M_{T_n(s)}(s))\\
T_M^{\mathsf{Y}}(s) &:= \min\left\{n \mid M_{T_n(s)}(s) \text{ is a value}\right\}
\end{aligned} \tag{3.2}$$

or equivalently, T_n is the time that the reduction sequence reaches its $(n+1)$th term[3] of the form $E[\mathsf{Y}\lambda x.N]$ or a value, and Y_n is the ranking function value at this time. We first state a useful property about r.v.s $T_0, T_1, T_2, \ldots$.

[3] $n+1$ because English's ordinals are badly named

Lemma 3.2. *$(T_n)_{n\geq 0}$ is an increasing sequence of stopping times adapted to $(\mathcal{F}_n)_{n\geq 0}$, and each T_i is bounded.*

Proof. The fact that they are stopping times follows from the definition, as the event $\{T_i \leq n\}$ depends only on M_k for $k \leq n$. The fact that they are bounded is proven by a reduction argument to the strong normalisation of the simply-typed nondeterministic lambda-calculus[27], where the values of real constants are ignored and conditionals are represented with nondeterminism.

Given a closed PPCF term M, we transform it to a term $\ulcorner M \urcorner$ of the nondeterministic simply-typed lambda-calculus as follows. (We may assume that the nondeterministic calculus is generated from a base type ι with a ι-type constant symbol X, and a function symbol $\bot : A$ for each function type A.)

$$\begin{aligned}
\ulcorner \mathsf{sample} \urcorner &:= (\lambda z.\mathsf{X})\mathsf{X} \\
\ulcorner \underline{r} \urcorner &:= \mathsf{X} \\
\ulcorner x \urcorner &:= x \\
\ulcorner \lambda x.N \urcorner &:= \lambda x.\ulcorner N \urcorner \\
\ulcorner M_1\, M_2 \urcorner &:= \ulcorner M_1 \urcorner \ulcorner M_2 \urcorner \\
\ulcorner \underline{f}(M_1, \cdots, M_n) \urcorner &:= (\lambda z_1 \cdots z_n.\mathsf{X}) \ulcorner M_1 \urcorner \cdots \ulcorner M_n \urcorner \\
\ulcorner \mathsf{if}(B, M_1, M_2) \urcorner &:= \big(\lambda z.(\ulcorner M_1 \urcorner + \ulcorner M_2 \urcorner)\big) \ulcorner B \urcorner \\
\ulcorner \mathsf{Y} N \urcorner &:= (\lambda y.\bot) \ulcorner N \urcorner
\end{aligned}$$

In the above, variables $z, z_1, \cdots, z_n$ are assumed to be fresh; f ranges over primitive functions and r over reals.

The transformation for the types is analogous, and $\ulcorner - \urcorner$ is constructed in such a way that if M is well-typed, so is $\ulcorner M \urcorner$. For any reduction sequence starting at M (with any trace) that does not include any Y-reductions, it is clear that there is a corresponding reduction sequence of $\ulcorner M \urcorner$. If the reduction sequence of $\ulcorner M \urcorner$ is of length $\leq l$, so is the reduction sequence of M (the transformation may lengthen reduction sequences, for example because of the way primitive functions are encoded, but never shorten them).

Thanks to the following

Theorem 3.3 (de Groote [27])**.** *The simply-typed nondeterministic lambda calculus is strongly normalising.* □

we have that the reduction sequence of $\ulcorner M \urcorner$ is finite and bounded, therefore so is the Y-free initial portion of the reduction sequence of M itself.

This suffices to show that T_0 is bounded. Given a bound l on T_n, there are finitely many skeletons $Sk(M_{T_n+1})$ and a fortiori finitely many possible values of $\ulcorner M_{T_n+1} \urcorner$. For each of these transformed terms, there is some bound l_i on its runtime, therefore T_{n+1} is bounded by $l + 1 + \max_i l_i$. □

The random variable T_M^{Y}, which is called the Y-*runtime* of M, can equivalently be defined as the number of Y-reduction steps in the reduction sequence of M. If $T_M^{\mathsf{Y}} < \infty$, clearly M terminates (with that trace), and conversely, if M

doesn't terminate, since each T_n is bounded but M_{T_n} is not a value for any n, T_M^{Y} is infinite. Combining these, $T_M^{\mathsf{Y}} < \infty$ almost surely iff M is AST.

Definition 3.4. M is called Y-PAST if $\mathbb{E}\left[T_M^{\mathsf{Y}}\right] < \infty$.

Y-positive almost sure termination is mostly of interest just because it implies AST but is weaker (and thus easier to prove) than PAST. The difference between Y-PAST and PAST is demonstrated by this example.

Example 3.1.1. The term

$$M = (\mathsf{Y}\lambda f\, c.\mathsf{if}(\mathsf{sample} - 0.5 < 0, c\,\underline{0}, f(\lambda x.c(c\,x)))) \;\; (\lambda x.x + 1)$$

which can be written in pseudocode as

```
c := λx. x+1
while (flip_coin() = Heads) {
    c := c ∘ c
}
c(0)
```

is Y-PAST and rankable, terminating with only 2 Y-reductions on average, i.e., $\mathbb{E}\left[T_M^{\mathsf{Y}}\right] = 2$, but applies the increment function 2^n times with probability 2^{-n-1} for $n \geq 0$, which diverges, i.e., $\mathbb{E}\left[T_M\right] = \infty$.

Lemma 3.3. *T_M^{Y} is a stopping time adapted to $(\mathcal{F}_{T_n})_{n\geq 0}$.*

Proof. To see $\left\{T_M^{\mathsf{Y}} = n\right\} \in \mathcal{F}_{T_n}$ for a given n, we need to show that $\left\{T_M^{\mathsf{Y}} = n\right\} \cap \{T_n \leq i\} \in \mathcal{F}_i$, for all $i \in \mathbb{N}$. Let $\mathcal{V}$ be the set of values in Λ^0 (which is measurable), and let $\mathcal{W}$ be the non-values. Then, it suffices to observe that $\left\{T_M^{\mathsf{Y}} = n\right\} \cap \{T_n \leq i\} = \bigcup_{l=n}^{i} \left(M_l^{-1}[\mathcal{V}] \cap M_{l-1}^{-1}[\mathcal{W}] \cap \{T_n = l\}\right)$, where each of $M_l^{-1}[\mathcal{V}]$, $M_{l-1}^{-1}[\mathcal{W}]$, and $\{T_n = l\}$ is in $\mathcal{F}_i$. □

Theorem 3.4 (Soundness of rankability)**.** *1. If a closed PPCF term M is rankable, then M is AST and Y-PAST.*

2. If a closed PPCF term M is strictly rankable, then M is PAST.

Proof. Let f be a ranking function on M. For part 1, since $(T_n)_{n\geq 0}$ is an increasing sequence of stopping times, each is adapted to $(\mathcal{F}_n)_{n\geq 0}$ and bounded (Lemma 3.2), and $(f(M_n))_{n\geq 0}$ is a supermartingale also adapted to $(\mathcal{F}_n)_{n\geq 0}$ (Theorem 3.2), it follows from the Optional Sampling Theorem 3.1 that $(Y_n)_{n\geq 0}$ is a supermartingale adapted to $(\mathcal{F}_{T_n})_{n\geq 0}$. Since M_{T_n} is of the form $E[\mathsf{Y}\lambda x.N]$ whenever $T_M^{\mathsf{Y}} > n$, it follows from Corollary 3.1 that $\mathbb{E}\left[M_{T_n+1} \mid \mathcal{F}_{T_n}\right] \leq M_{T_n} - \mathbf{1}_{\{T_M^{\mathsf{Y}}\}}$, then by Optional Sampling (again since T_{n+1} is bounded), $\mathbb{E}\left[M_{T_{n+1}} \mid \mathcal{F}_{T_n}\right] \leq M_{T_n} - \mathbf{1}_{\{T_M^{\mathsf{Y}}\}}$. The stopping time T_M^{Y} is also adapted to $(\mathcal{F}_{T_n})_{n\geq 0}$ (Lemma 3.3) therefore we have that $(Y_n)_{n\geq 0}$ is a 1-ranking supermartingale w.r.t. $(\mathcal{F}_{T_n})_{n\geq 0}$ and T_M^{Y}. Therefore, by Lemma 3.1, $T_M^{\mathsf{Y}} < \infty$ a.s. and $\mathbb{E}\left[T_M^{\mathsf{Y}}\right] < \infty$.

Statement 2 follows at once from Theorem 3.2 and Lemma 3.1. □

Thus the method of (strict) ranking function is sound for proving (positive) a.s. termination of PPCF programs. It is in fact also complete in a limited sense: if M is Y-PAST then M is rankable (Theorem 3.5).

3.1.3 Restrictions

The fact that if a term is rankable it is not just AST but Y-PAST implies that this proof method is incomplete, since there are plenty of terms which are AST but not Y-PAST. A one or two dimensional random walk, if implemented in the obvious way (Example 3.1.2 below), will take one Y step per iteration, therefore the expected Y-runtime is the expected number of moves to reach the origin, which is infinite.

You might wonder why it is necessary that the ranking function decreases by at least 1 for every Y reduction step, rather than merely decreasing by some amount. If this restriction could be lifted, more slowly-terminating terms like the random walks could be rankable. One way to partially lift this restriction will be shown in Section 3.3, but for now, I will just show why it can't be removed completely. The term $\Omega\ \underline{0} = \mathsf{Y}(\lambda wn.w(n+\underline{1}))\ \underline{0}$ is clearly non-terminating. Roughly speaking, it could be assigned the (weakened) ranking function $\Omega\ \underline{n} \mapsto 2^{-n}$ (ignoring the fact that $\Omega\ \underline{0}$ does not actually reduce to $\Omega\ \underline{1}$ directly, but rather reduces to $\Omega\ (\underline{0}+\underline{1})$ after 3 steps), which decreases by 2^{-n-1} for each Y reduction. Since this supposed ranking function decreases indefinitely but only approaches 0 asymtotically, it is unable to force almost sure termination. Of course, the decrease being at least 1 is arbitrary, it is merely necessary that it be bounded below.

Example 3.1.2. A possible implementation of the 1D unbiased random walk in PPCF is

$$\left(\mathsf{Y}\,\lambda fn.\,\mathsf{if}\left(n=0,0,f\,(n-\underline{1})\oplus_{1/2} f\,(n+\underline{1})\right)\right)10$$

Here some new syntactic sugar is used, $M \oplus_p N = \mathsf{if}\left(\mathsf{sample}-\underline{p}<0,M,N\right)$. It will be shown in Section 3.3.2 how to prove this term AST using an extension of the ranking function method, but as mentioned earlier, the theorem already proven Theorem 3.4 is not strong enough. It is possible to rewrite the program in such a way that it does still represent an unbiased random walk but it becomes rankable. In pseudocode,

$$\begin{aligned}&\mathrm{walk}(f,n) = f(\mathrm{walk}(f\circ f),n)\\&\mathrm{step}(c,n) = \mathsf{if}\left(n=0,0,c(n+1)\oplus_{1/2} c(n-1)\right)\\&\mathrm{walk}(\mathrm{step},10)\end{aligned}$$

or in actual PPCF,

$$W\,S\,\underline{10} = (Y\lambda wf.f(w(\lambda c.f(fc))))(\lambda cn.\mathsf{if}\left(n-\underline{1}<0,0,c(n+\underline{1})\oplus_{1/2} c(n-\underline{1})\right))\underline{10}.$$

The number of execution steps is not decreased, so this is still not PAST, but like in Example 3.1.1, an exponentially increasing number of logical steps occurs for every Y-reduction step, so $\mathbb{E}\left[T_W^{\mathsf{Y}}\right]$ is roughly $\mathbb{E}\left[\log_2(\text{the number of steps})\right]$, which is finite. Although the partial completeness result Theorem 3.5 implies that $W\,S\,\underline{10}$ is rankable, I'm not going to actually explicitly construct its ranking function yet, because it turns out constructing ranking functions in the basic form defined in Definition 3.2 is extremely tedious. Really, Theorem 3.4 is better thought of as a lemma on the way to better proof methods. See Example 4.5.3 for a sort of ranking function for this term that will be possible to construct with these better methods.

The issue is that the ranking function's value must be defined at every single reachable term (or at least an almost complete subset of them). That might be practical if the ranking functions are being generated by a computer, but not by hand. In an imperative language, variants only need to be defined at the state reached at the end of each iteration of a loop. The rigid execution order provides a natural set of checkpoints, and any non-recursive non-terminating program fails to terminate precisely by one of its loops (the identities of which are known in advance) running forever. In a functional language like PPCF, there are much more complicated and varied ways of failing to terminate. We know that any non-terminating program must execute infinitely many Y reduction steps by Lemma 3.2, but it is much harder to say what will be going on at each of those points. In a recursively defined function like $\lambda x.\mathsf{Y}\lambda y.xy$, the recursive variable y may be passed as an argument to arbitrary functions passed in as x by the caller, complicating the recursive structure at runtime.

For this reason, the applicability of the ranking function method for proving AST and PAST (including the extensions still to be shown) is somewhat limited to cases where it is, in a sense, the almost-sure bit that is the issue, not the termination. If a program's termination is nontrivial to prove simply because of the many complicated reachable terms, constructing a ranking function requires a brute-force enumeration of almost all of these cases anyway. It is only when the number of cases is naturally manageable and (at least a significant subset of) the reachable terms can be parameterised, with the probabilistic reductions between these cases providing the main difficulty in proving the program's termination, that I expect the ranking function method to be particularly useful.

3.2 Sparse Ranking Functions

Since ranking functions contain such an excessive amount of usually irrelevant detail, it stands to reason that much of it could be inferred rather than explicitly specified. This is the idea behind sparse ranking functions, which will be defined shortly, but since this requires deriving ranking functions, let's first take a look at the conditions under which they can be shown to exist in general. Y-PAST is a stronger condition than AST, and is necessary for rankability by Theorem 3.4. It turns out it is also sufficient.

Theorem 3.5. *Given a closed* Y*-PAST term* M*, the function* $f : Rch(M) \to \mathbb{R}$ *given by*

$$f(N) := \begin{cases} \mathbb{E}[\textit{number of } \mathsf{Y}\textit{-reduction steps from } N \textit{ to a value}] & \textit{if this is finite} \\ 0 & \textit{otherwise} \end{cases}$$

is a ranking function for M*, therefore every closed* Y*-past term is rankable.*

Proof. Let $C = \{N \in Rch(M) \mid N \text{ is } \mathsf{Y}\text{-PAST}\}$. Clearly C is almost complete, since if it were possible to reach a non-Y-PAST term from M with non-zero probability, M would also not be Y-PAST. For $N \in C$ not a value, each of the conditions required by Definition 3.2 for f to be a ranking function is quite easily verified. □

For most terms, any ranking function can trivially be made smaller by making the almost complete reachable set smaller, therefore there is no smallest ranking function. However, these smaller ranking functions that are generated are in a sense equivalent. Two ranking functions f, g where $\{N \mid f(N) = g(N)\}$ is an almost complete reachable set, can be considered essentially the same (this is an equivalence relation because the intersection of two almost complete reachable sets is also almost complete).

Theorem 3.6. *The ranking function* f *given in the previous theorem is the least ranking function on that term, up to equality on an almost-complete reachable set.*

Proof. Suppose g is another ranking function on the same term such that the set of terms N where $f(N) > g(N)$ is reachable with non-zero probability. There is then some such N such that $Rch(N) \cap C_g \cap C$ is an almost complete reachable set for N. The restrictions of f and g to $Rch(N)$ have the same properties assumed of f and g, so assume w.l.o.g. that $N = M$. Since at each step, almost surely

$$f(M_n) = \mathbb{E}\left[f(M_{n+1}) \mid \mathcal{F}_n\right] + \mathbf{1}_{M_n \to M_{n+1} \text{ is a } \mathsf{Y}\text{-reduction}}$$

exactly, and g is a ranking function, the difference $g - f$ is a supermartingale (with the same setup as in Theorem 3.2); therefore $\mathbb{E}[g(M_n)] \leq \mathbb{E}[f(M_n)] + g(M) - f(M)$, for all n. Now $\mathbb{E}[f(M_n)] = \sum_{k=n}^{\infty} \mathbb{P}[M_k = E[\mathsf{Y}N] \text{ for some } E, N]$, and this sum converges for $n = 0$, therefore the partial sums from n to ∞, $\mathbb{E}\left[f(M_n)\right]$, converge to 0 as $n \to \infty$, therefore as $g(M) - f(M) < 0$, eventually $\mathbb{E}[g(M_n)] < 0$, which is impossible. It follows that $g \geq f$ on some almost complete reachable set as required. □

I have referred already to the excessive complexity of defining ranking functions explicitly. To illustrate this, consider this example.

Example 3.2.1 (Geometric distribution)**.** Let

$$\Theta := \lambda f, n.\mathsf{if}(\mathsf{sample} - 0.5 < 0, n, f(n+1)).$$

The term $(\mathsf{Y}\,\Theta)\,\underline{0}$ generates a geometric distribution. Despite its simplicity, its *Rch* contains all the terms in Figure 3.1, for each $i \in \mathbb{N}, r \in I$.

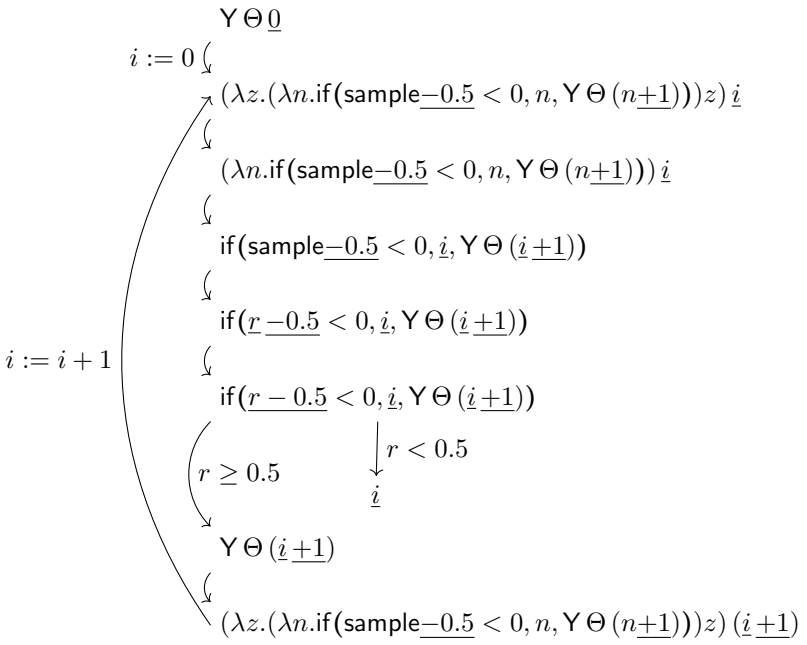


Figure 3.1: The reachable terms of $\mathsf{Y}\,\Theta\,\underline{0}$

Even in this simple case, defining a ranking function explicitly is awkward because of the number of cases, although in most cases, because the value need only be greater than or equal to that of the next term in sequence, it suffices to take the ranking function as having the same value as the next term, so that overall it takes only 3 distinct values. (I will explain why later.)

The definition of rankability is also inconvenient for syntactic sugar. When using the syntax $M \oplus_p N$, defined as $\mathsf{if}(\mathsf{sample} - p < 0, M, N)$, for example, it is easy to think of $M \oplus_p N$ as reducing to M or N, depending on the first value of $s \in \mathbb{S}$, with probability p resp. $(1-p)$. Technically though, it reduces first to $\mathsf{if}(\underline{r} - p < 0, M, N)$ for all $r \in I$, so those terms all need values of the ranking function too.

In both of these cases, there are only some values of the ranking function that are semantically important. Sparse ranking functions provide a way to use ranking functions that are only defined at these important terms, not all of the uninteresting ones in between.

Definition 3.5. Define a *sparse ranking function* on a closed term M to be a partial function $f : Rch(M) \rightharpoonup \mathbb{R}$ such that

- $f(N) \geq 0$ for all N where f is defined,
- f is defined at M,

- for any N in the domain of definition of f, evaluation of N will almost certainly eventually reach some O which is either a value or in the domain of definition of f, and $f(N) \geq \mathbb{E}[f(O) +$ the number of Y-reduction steps from N to $O]$ (where $f(O)$ is taken to be 0 if O is a value outside of the domain of f).

Theorem 3.5 makes even stronger assumptions than almost sure termination, so it isn't useful in proving a term AST, but it can be modified to start with a sparse ranking function and fill in the gaps rather than building a ranking function from the assumption of Y-PASTness. Providing a sparse ranking function is essentially part way between providing a ranking function and directly proving Y-PASTness. At one extreme, if a sparse ranking function is total, there is always only a single step between N and O, so the conditions for it to be a sparse ranking function in this case reduce to Definition 3.2 and the sparse ranking function is also a ranking function. At the other extreme, if a term M is Y-PAST, a partial function that is defined only at M (with $f(M) = \mathbb{E}$ [the number of Y-reduction steps from M to a value]) is a sparse ranking function.

Theorem 3.7 (Sparse function)**.** *Every sparse ranking function is a restriction of a ranking function.*

Proof. Take a closed term M and sparse ranking function f on M. Define $f_1 : Rch(M) \rightharpoonup \mathbb{R}$ by

$$f_1(N) := f(N) \text{ whenever } f(N) \text{ is defined,}$$
$$f_1(V) := 0 \text{ for values } V \text{ not in the domain of } f.$$

Define $(\text{next}(N,s), \_) = \text{red}^n(N,s)$ for the least $n \geq 0$ such that it's in the domain of f_1, and $g(N,s) := |\{m < n \mid \text{red}^m(N,s) \text{ is of the form } (E[\mathsf{Y}\,\lambda x.N'], s')\}|$. The function next is well-defined (i.e. n is finite) for almost all $N \in Rch(M)$ and $s \in \mathbb{S}$ by induction on the path from M to N, by the third condition on sparse ranking functions. Define $f_2(N) = \int_{\mathbb{S}} f_1(\text{next}(N,s)) + g(N,s)\,\mu_{\mathbb{S}}(\mathrm{d}s)$ and $f_3(N) = f_2(N)$ when $f_2(N)$ is finite, and an arbitrary value (say 0) otherwise. Take $C = \{N \in Rch(M) \mid f_2(N) < \infty\}$. Suppose for a contradition that C is not almost complete, then there is some time n such that $\mathbb{P}[M_n \notin C] > 0$, then there must be some $N \in \text{dom}(f)$ and $m < n$ such that $\mathbb{P}[M_n \notin C, \forall k \in (m,n).M_k \notin \text{dom}(f) \mid M_m = N] > 0$, therefore $f(N) \geq \infty$ by the third condition on a sparse ranking function, which is a contradiction therefore C is an almost complete reachable set. The (total) function f_3 agrees with f on f's domain, and it satisfies the conditions for a ranking function on the set C, therefore it is a ranking function of which f is a restriction, as desired. □

As a corollary, any term which admits a sparse ranking function terminates almost surely.

This theorem is the reason why the introduction of the almost complete reachable set in the definition of a ranking function was necessary in the first place.

Any Y-PAST term has a sparse ranking function trivially (defined only on that term itself), but there are some Y-PAST terms from which non-Y-PAST terms are reachable. For example, $\mathsf{if}(-\mathsf{sample} < 0, \underline{0}, \Omega)$ where $\Omega = (\mathsf{Y}\lambda x.x)\,\underline{0}$. It can reduce to Ω which is non-terminating, but it almost certainly terminates in 3 steps. If the ranking function condition were required to apply to every single reachable term, then this term would not be rankable because that would require Ω to be rankable too.

This simple case could be excluded by making the definition of a sparse ranking function stricter, so that for any N in the domain of definition of f, evaluation of N will certainly eventually reach some O which is either a value or in the domain of f, rather than merely almost certainly. This would still not be entirely satisfactory however. Consider this term:

$$\begin{aligned} M &= \mathsf{if}(-\mathsf{sample} < 0, \underline{0}, WX) \\ W &= \mathsf{Y}\lambda f n.\mathsf{if}(n < 0, \underline{0}, f(n-1)) \\ X &= \left\lfloor \frac{1}{\mathsf{sample}} \right\rfloor \\ W \to W' &= \lambda z.(\lambda n.\mathsf{if}(n < 0, \underline{0}, W(n-1))\, z \end{aligned}$$

(with $\left\lfloor \frac{1}{\underline{0}} \right\rfloor$ evaluating to some arbitrary finite value). M has a sparse ranking function with bounded steps given by $f(M) = 0, f(W'\underline{n}) = n+1$. M reduces after at most 7 steps to either $W'\underline{n}$ or $\underline{0}$, and each of those terms is Y-PAST, but the intermediate term WX is not Y-PAST because the probability of reaching $W'\underline{n}$ from WX decreases too slowly as $n \to \infty$. In order for even this more restricted form of sparse ranking function to extend to a ranking function in this case, it is therefore necessary to use the almost complete reachable set to exclude WX. There is the additional minor advantage that this complication makes rankability equivalent to Y-PAST, which is a nicer criterion than would be the case without the almost complete set.

Syntactic Sugar. Let $M \oplus_p N = \mathsf{if}(\mathsf{sample} - p < 0, M, N)$, for $p \in (0,1]$, then there are the pseudo-reduction relations

$$E[M \oplus_p N] \to^3 E[M] \qquad E[M \oplus_p N] \to^3 E[N]$$

$$\mathrm{red}^3(E[M \oplus_p N], s) = \begin{cases} (E[M], \pi_t(s)) & \text{if } \pi_h(s) < p \\ (E[N], \pi_t(s)) & \text{if } \pi_h(s) \geq p. \end{cases}$$

A sparse ranking function could be defined with respect to this shortcut reduction simply by replacing $\to$ and red in the definition of a sparse ranking function by a version that goes straight from $N \oplus_p O$ to N or O. Such a pseudo-sparse ranking function would then be a partial function from a subset of $Rch(M)$, so it could also be considered as a partial function from all of $Rch(M)$, and it would in fact also be an actual sparse ranking function. It is therefore possible to prove rankability directly using the shortcutted reductions.

A similar procedure would work for other forms of syntactic sugar. If a closed term N eventually reduces to one of a set of other terms $\{N_i \mid i \in I\}$ with certain probabilities, a sparse ranking function defined w.r.t. a reduction sequence that skips straight from N to N_i is also a valid sparse ranking function for the original reduction function, and therefore its existence implies almost sure termination. There is a caveat, however, that Y-reduction steps skipped over in the shortcut still need to be counted for the expected number of Y-reduction steps.

With this abbreviation, the geometric distribution example from earlier can be written as $(\mathsf{Y}\lambda fn.\ n \oplus_{0.5} f(n+1))\,\underline{0}$. It is then easy to see that the following is a sparse ranking function:

$$\begin{aligned}(\mathsf{Y}\lambda fn.\ n \oplus_{0.5} f(n+1))\,N &\mapsto 2\\ \underline{i} \oplus_{0.5} (\mathsf{Y}\lambda fn.\ n \oplus_{0.5} f(n+1))\,(\underline{i}+1) &\mapsto 1\\ \underline{i} &\mapsto 0.\end{aligned}$$

In fact, even the partial function $\mathsf{Y}\Theta N \mapsto 2$ alone is a sparse ranking function for this term.

3.3 Antitone Supermartingales and Ranking Functions

The use of sparse ranking functions partially deals with one weakness of the basic ranking function theorem. Another weakness that has already been discussed is the fact that non-Y-PAST terms such as the 1D unbiased random walk $\left(\mathsf{Y}\,\lambda fn.\,\mathsf{if}(n = 0, \underline{0}, f\,(n - \underline{1}) \oplus_{1/2} f\,(n + \underline{1}))\right)\underline{10}$ are not rankable. In order to correct this deficiency, we must allow the supermartingale to decrease more slowly in a controlled manner, similarly to what [40, Theorem 4.1] does for an imperative language. Antitone ranking functions, another generalisation of the basic notion, are an implementation of this, although as will be seen in Section 3.3.1, the approach taken here differs a little from that in [40].

Definition 3.6. The definition of *antitone ranking function* f for $M \in \Lambda^0$ is the same as that of ranking function (Definition 3.2) except that, for some antitone function $\varepsilon : \mathbb{R}_{\geq 0} \to \mathbb{R}_{>0}$ ("antitone" meaning: $r < r'$ implies $\varepsilon(r) \geq \varepsilon(r')$), the function f satisfies the condition

$$f(E[R']) \leq f(E[R]) - \varepsilon(f(E[R]))$$

where $R \to R'$ is the Y-redex rule, instead of the original rule with a constant decrease of 1. Any closed term for which an antitone ranking function exists is called *antitone rankable*.

Roughly speaking, the function ε specifies how fast M must terminate. The faster ε tends to 0, the slower M may terminate. Note that antitone ranking functions are actually a generalisation of ranking functions, even though the way we reference them may suggest that they are a type of ranking functions.

In particular, an antitone ranking function with $\varepsilon(x) = 1$ constantly is just a ranking function.

To match the notion of antitone ranking functions, we need an altered version of the notion of ranking supermartingales.

Definition 3.7. Given a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, and a supermartingale $(Y_n)_{n\geq 0}$ and a stopping time T adapted to filtration $(\mathcal{F}_n)_{n\geq 0}$, we say that $(Y_n)_{n\geq 0}$ is an *antitone strict supermartingale w.r.t.* T if for all $n \geq 0$, we have $Y_n \geq 0$, and there exists an antitone function $\varepsilon : \mathbb{R}_{\geq 0} \to \mathbb{R}_{>0}$ satisfying $\mathbb{E}[Y_{n+1} \mid \mathcal{F}_n] \leq Y_n - \varepsilon(Y_n) \cdot \mathbf{1}_{\{T>n\}}$.

Theorem 3.8. *Let $(Y_n)_{n\geq 0}$ be an antitone strict supermartingale w.r.t. stopping time T. Then $T < \infty$ a.s.*

Proof. First, as (Y_n) is a supermartingale, $\mathbb{E}[Y_n] \leq \mathbb{E}[Y_0]$. Therefore

$$
\begin{aligned}
& \mathbb{E}[Y_n \mid T > n] \\
= \quad & \{ \text{ rearranging terms } \} \\
& \frac{\mathbb{P}[T > n]\,\mathbb{E}[Y_n \mid T > n] + \mathbb{P}[T \leq n]\,\mathbb{E}[Y_n \mid T \leq n] - \mathbb{P}[T \leq n]\,\mathbb{E}[Y_n \mid T \leq n]}{\mathbb{P}[T > n]} \\
= \quad & \{ \text{ definition of conditional expectation } \} \\
& \frac{\mathbb{E}[Y_n] - \mathbb{P}[T \leq n]\,\mathbb{E}[Y_n \mid T \leq n]}{\mathbb{P}[T > n]} \\
\leq \quad & \{ \ Y_n \geq 0 \text{ always } \} \\
& \frac{\mathbb{E}[Y_n]}{\mathbb{P}[T > n]} \\
\leq \quad & \{ \ (Y_n)_n \text{ is a supermartingale } \} \\
& \frac{\mathbb{E}[Y_0]}{\mathbb{P}[T > n]}
\end{aligned}
$$

Claim: For all $0 < x \leq 1$, $\mathbb{P}[T > B_x] \leq x$, where $B_x = \left\lceil \dfrac{\mathbb{E}[Y_0] + 1}{x\ \varepsilon(\mathbb{E}[Y_0]\ x^{-1})} \right\rceil$ and $\varepsilon : \mathbb{R}_{\geq 0} \to \mathbb{R}_{>0}$ is the antitone function.

As the convex hull of ε (the greatest convex function less than or equal to it) satisfies all the conditions assumed of ε, in addition to being convex, assume wlog that ε is convex.

Assume for a contradiction that $\mathbb{P}[T > B_x] > x$. Then, take $n \leq B_x$. We have

$$
\begin{aligned}
& \mathbb{E}[Y_n - Y_{n+1}] \\
= \quad & \{ \ \mathcal{F}_n \subseteq \mathcal{F}_{n+1}, \text{ def. \& linearity of cond. expectation } \} \\
& \mathbb{E}[Y_n - \mathbb{E}[Y_{n+1} \mid \mathcal{F}_n]] \\
\geq \quad & \{ \text{ antitone strict assumption } \} \\
& \mathbb{E}\left[\varepsilon(Y_n) \cdot \mathbf{1}_{\{T>n\}}\right] \\
= \quad & \{ \text{ definition of expectation conditioning on an event } \}
\end{aligned}
$$

$$
\begin{aligned}
& \mathbb{P}[T > n]\ \mathbb{E}\left[\varepsilon(Y_n) \mid T > n\right] \\
\geq \quad & \{\ \text{Jensen's inequality}\ \} \\
& \mathbb{P}[T > n]\ \varepsilon(\mathbb{E}\left[Y_n \mid T > n\right]) \\
\geq \quad & \{\ \text{proved earlier}\ \} \\
& \mathbb{P}[T > n]\ \varepsilon\Big(\frac{\mathbb{E}\left[Y_0\right]}{\mathbb{P}[T > n]}\Big) \\
> \quad & \{\ \text{assumption, } \mathbb{P}[T > n] \geq \mathbb{P}[T > B_x] > x\ \} \\
& x\ \varepsilon\big(\mathbb{E}\left[Y_0\right]\ x^{-1}\big).
\end{aligned}
$$

Therefore, by a telescoping sum

$$
\begin{aligned}
\mathbb{E}\left[Y_{B_x}\right] = \mathbb{E}\left[Y_{B_x} - Y_0 + Y_0\right] &\leq \mathbb{E}\left[Y_0\right] - B_x\ x\ \varepsilon(\mathbb{E}\left[Y_0\right]\ x^{-1}) \\
&\leq -1 < 0
\end{aligned}
$$

which is a contradiction, therefore the claim must be true, therefore $\mathbb{P}[T > n] \to 0$ as $n \to \infty$, therefore $T < \infty$ a.s. □

It is essential in this theorem that ε be defined for all $\mathbb{R}_{\geq 0}$, not just the values that $(Y_n)_{n\geq 0}$ actually takes (or at least if $(Y_n)_{n\geq 0}$ is uniformly bounded, ε must be positive at the supremum as well as the realized values). The following example illustrates what goes wrong if this restriction is ignored.

Example 3.3.1. Take the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ where Ω is the closed interval of reals $[-1, 1]$, $\mathcal{F}$ the Borel σ-algebra, and $\mathbb{P}$ the uniform probability measure. Let $\omega \in [-1, 1]$. Define random variables T and $(Y_k)_{k\geq 0}$:

$$
T(\omega) := \begin{cases} \min\left\{n \in \mathbb{N} \mid \omega > 2^{-n}\right\} & \text{if } \omega \in (0, 1] \\ \infty & \text{otherwise} \end{cases}
$$

$$
Y_k(\omega) := \begin{cases} 4 - 2^{-k} & \text{if } \omega \in [-1, 2^{-k}] \text{ (equivalently } k < T(\omega)) \\ 0 & \text{otherwise} \end{cases}
$$

Plainly, T is a stopping time, and $(Y_n)_{n\geq 0}$ is a supermartingale, adapted to the filtration $(\mathcal{F}_n)_{n\geq 0}$ where $\mathcal{F}_n = \sigma(\{[x, y] \mid y \geq x \geq 2^{-n}\})$ (i.e. $\mathcal{F}$, but with all of the values $< 2^{-n}$ indistinguishable) for all n. In this case, $(Y_n)_{n\geq 0}$ either tends to 4 as $n \to \infty$, or drops to 0 at some point, with an exponentially decreasing probability. With respect to the antitone function $\varepsilon(x) = \frac{4-x}{4}$ and stopping time T, we have that $(Y_n)_{n\geq 0}$ is an antitone strict supermartingale, except that $\varepsilon(4) = 0$. Because ε fails to meet the conditions of Definition 3.7, even though $\varepsilon(Y_n)$ never actually reaches 0, Theorem 3.8 still does not apply, and indeed its conclusion is false, as $T = \infty$ with probability $\frac{1}{2}$.

Theorem 3.9 (Antitone ranking function soundness)**.** *If a closed PPCF term M is antitone rankable, then $T_M^{\mathsf{Y}} < \infty$ a.s. (equivalently, M is AST).*

Proof. As in Theorem 3.4, take the probability space $(\mathbb{S}, \Sigma_{\mathbb{S}}, \mu_{\mathbb{S}})$, and define the same r.v. $T_n, X_n, Y_n, T_M^{\mathsf{Y}}$. Thanks to Theorem 3.2, $(Y_n)_{n\geq 0}$ is a supermartingale. By the same logic as in Corollary 3.1 that implies the ranking function conditions make Y_n a ranking supermartingale w.r.t. T_M^{Y}, the antitone ranking function condition on M implies it is an antitone strict supermartingale. Thus, by Theorem 3.8, $T_M^{\mathsf{Y}} < \infty$ a.s. □

It can also be seen that for any particular ε, a specific bound on the rate of termination is implied by the proof.

Corollary 3.2. *If a closed PPCF term M is antitone rankable with antitone function ε and ranking function f, and ε is convex, then for all $0 < x \leq 1$, $\mathbb{P}[T_M^{\mathsf{Y}} > B_x] \leq x$, where $B_x = \left\lceil \dfrac{f(M)+1}{x\,\varepsilon(f(M)\,x^{-1})} \right\rceil$.*

As before, constructing antitone ranking fuctions completely is not necessary, and there is a corresponding notion of an antitone sparse ranking function.

Definition 3.8. An *antitone sparse ranking function* on a closed term M is a partial function $f : Rch(M) \rightharpoonup \mathbb{R}$ such that for some antitone function $\varepsilon : \mathbb{R}_{\geq 0} \to \mathbb{R}_{>0}$:

- $f(N) \geq 0$ for all N where f is defined,
- f is defined at M,
- for any N in the domain of definition of f, evaluation of N will almost certainly eventually reach some O which is either a value or in the domain of definition of f, and $f(N) \geq \mathbb{E}[f(O)+\varepsilon(f(O))\times$the number of Y-reduction steps from N to $O]$ (where $f(O)$ is taken to be 0 if O is a value outside of $\mathrm{dom}(f)$).

Theorem 3.10. *Every antitone sparse ranking function is a restriction of an antitone ranking function.*

Proof. The proof mostly proceeds in the same way as Theorem 3.7. Take a closed term M and an antitone sparse ranking function f on M, with a corresponding antitone function ε. Assume wlog that ε is convex (as if it isn't, we can just take its convex hull instead). Define $f_1 : Rch(M) \rightharpoonup \mathbb{R}$ by

$$f_1(N) = f(N) \text{ whenever } f(N) \text{ is defined,}$$
$$f_1(V) = 0 \text{ for values } V \text{ not in the domain of } f.$$

Define $(\mathrm{next}(N,s), \_) = \mathrm{red}^n(N,s)$ for the least $n \geq 0$ such that it's in the domain of f_1, and $g(N,s) := |\{m < n \mid \mathrm{red}^m(N,s) \text{ is of the form } (E[\mathsf{Y}\,\lambda x.N'], s')\}|$. The function next is well-defined (i.e. n is finite) for almost all $N \in Rch(M)$ and $s \in \mathbb{S}$ by induction on the path from M to N, by the third condition on antitone sparse ranking functions. Define $f_2(N) := \int_{\mathbb{S}} f_1(\mathrm{next}(N,s)) + \varepsilon(f_1(\mathrm{next}(N,s)))g(N,s)\,\mathrm{d}\mu_{\mathbb{S}}(s)$ and $f_3(N) = f_2(N)$ when $f_2(N)$ is finite, and 0 otherwise. Take $C = \{N \in Rch(M) \mid f_2(N) < \infty\}$. C is almost complete by

the same argument as in Theorem 3.7. For any term N where f is defined, $f_3(N) = f(N)$, and the value that f_3 would have at N if f were not defined at N is $\leq f(N)$, by the third condition on antitone sparse ranking functions. In order to show that f_3 is an antitone sparse ranking function, it therefore suffices to show that the value that f_2 would have had at each term if f were not defined at that term is at least the expectation of f_2 after one reduction step (plus $\varepsilon(f_2(N))$ if the reduction step is a Y-reduction). For any term N which is not of the form $E[R]$ for some Y-redex R, this is trivial. If R is a Y-redex, then $\varepsilon(f_2(N)) \leq \int_{\mathbb{S}} \varepsilon(f_1(\mathrm{next}(N, s)))\,\mathrm{d}\mu_{\mathbb{S}}(s)$ by the convexity of ε, because n is bounded. Therefore, the (total) function f_2, which agrees with f on f's domain, is an antitone ranking function on M (with the same function ε if it's convex). □

As a corollary, any term which admits an antitone sparse ranking function terminates almost surely.

In constructing an antitone sparse ranking function, there is not quite as much freedom in choosing which terms to give ranking function values as when constructing an ordinary (non-antitone) sparse ranking function, because if there is some reduction sequence $N \to \cdots \to O$ in $Rch(M)$ and the function is specified at N and O but none of the intermediate terms, the amount the function must decrease from N to O depends on the value of ε at $f(O)$ only, even if it would be smaller at the intermediate terms that are passed over. Adding extra intermediate specified terms may in some cases therefore make a smaller decrease possible. Consider, for example, the case of a term M_0 which reduces after 100 Y-reduction steps to M_1, which in turn reduces after 10 Y-reduction steps to M_2 (deterministically in both cases), and assume ε has already been chosen, with $\varepsilon(x) = 10^{-1-x}$ so that $\varepsilon(0) = 0.1$ and $\varepsilon(1) = 0.01$, and the value of the antitone sparse ranking function f at M_2 is 0. If f is specified at M_0 but not M_1 (and not at any other intermediate steps either), it must have $f(M_0) \geq \varepsilon(f(M_2)) \times 110 = 0.1 \times 110 = 11$, but if f is given a value at M_1 too, it is possible that $f(M_1) = 0 + \varepsilon(0) \times 10 = 1$ and $f(M_0) = 1 + \varepsilon(1) \times 100 = 2$. A case similar to this is shown in Figure 3.2. In some cases of course, just picking a smaller ε or larger starting value would suffice, but not always. As another example, if an antitone sparse ranking function defined only at the initial term exists, the term is Y-PAST, which as we shall see shortly, not all antitone rankable terms are. In this case, picking a smaller ε is therefore not sufficient, and only specifying more values will do. Also, even if a sparse antitone ranking function is specified for all $N \in Rch(M)$, the decrease required is still slightly higher than for an antitone ranking function, because it depends on the rank after the reduction rather than before, but this can be compensated for just by decreasing $\varepsilon(x)$ to $\varepsilon(\max(0, x - \varepsilon(x)))$.

3.3.1 Progress Conditions

The progress condition I have been using, that Y_n decreases by at least $\varepsilon(Y_n)$ in expectation, is different from that used in [40]. It is more similar to the notion

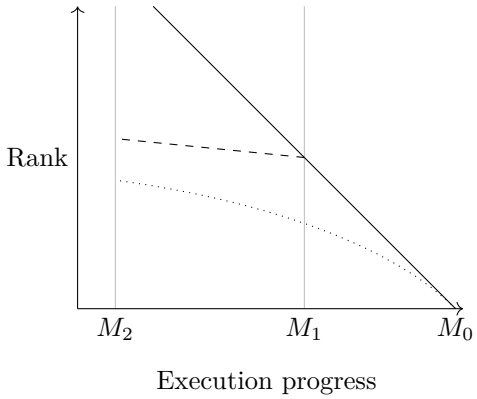


Figure 3.2: Sketch (not to scale) of some antitone sparse ranking functions with the same ε and different numbers of specified values. The solid line shows the increase required with only one specified value, the dashed line with two, and the dotted line shows what it could be if every value is specified (which is nearly the same as if the antitone ranking function is not sparse).

of linear ranking supermartingales from [13, §3.2], but without the restriction of linearity. The progress condition from [40, Theorem 4.1(ii)] can be stated (in the language I have been using, which also differs from theirs in some other irrelevant aspects) as a variant of the notion of an antitone strict supermartingale, which I will call a *quantile strict supermartingale* to differentiate between the two.

Definition 3.9. Given a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, and a supermartingale $(Y_n)_{n\geq 0}$ and a stopping time T adapted to filtration $(\mathcal{F}_n)_{n\geq 0}$, we say that $(Y_n)_{n\geq 0}$ is a *quantile strict supermartingale* w.r.t. T if there exist antitone functions $p : \mathbb{R}_{\geq 0} \to (0, 1]$ and $d : \mathbb{R}_{\geq 0} \to \mathbb{R}_{>0}$ such that for all $n \geq 0$, we have $Y_n \geq 0$ and $\mathbb{P}[Y_{n+1} \leq Y_n - d(Y_n) \mid \mathcal{F}_n] \geq p(Y_n) \cdot \mathbf{1}_{\{T>n\}}$.

The key difference from Definition 3.7 is that in an antitone strict supermartingale, given one value Y_n, the next value (conditional on Y_n) has the restriction that $\mathbb{E}[Y_{n+1}] \leq Y_n - \varepsilon(Y_n)$ (if $n < T$), whereas for a quantile strict supermartingale, the corresponding restriction is that $\mathbb{E}[Y_{n+1}] \leq Y_n$ and $\mathbb{P}[Y_{n+1} \leq Y_n - d(Y_n)] \geq p(Y_n)$, so the amount of variation is bounded below but the expectation is not required to decrease. The advantage of antitone strict supermartingales, and the reason I have departed from the choice made in [40], is that they are better behaved when taking subsequences. Expectations, which the antitone strict progress condition is based on, combine in a simple linear way which quantiles don't, so that the restriction on Y_{n+2} given Y_n is relatively simple. This is necessary in two places for constructing ranking functions for PPCF in a way that it isn't in [40]'s context of an imperative language. Firstly, the steps with the randomness (sample reduction steps) are separated from the steps which count for the progress condition (the Y reduction steps), and secondly, in the construction based on antitone sparse ranking functions, it is necessary to

combine several steps into one. In the context of an imperative language, neither of these are necessary because the orderly structure of the repetition with while loops provides a natural subset of execution steps on which the ranking function can be defined directly.

The difference between these two progress conditions is not actually very significant though, because of the near equivalence between them.

Theorem 3.11. *If $(Y_n)_{n\geq 0}$ is an antitone strict supermartingale, it is also a quantile strict supermartingale, and if it is a quantile strict supermartingale, $(\log(Y_n+1))_{n\geq 0}$ is an antitone strict supermartingale (with the same stopping time and filtration in both cases).*

Antitone ⇒ Quantile: Take an antitone strict supermartingale $(Y_n)_{n\geq 0}$, with filtration $(\mathcal{F}_n)_{n\geq 0}$, stopping time T and antitone function ε. Define $p(x) = \frac{\varepsilon(x)}{2x}$ and $d(x) = \frac{\varepsilon(x)}{2}$. I will show that $(Y_n)_{n\geq 0}$ is a quantile strict supermartingale with these values of p and d. The condition that $Y_n > 0$ follows by definition since $(Y_n)_{n\geq 0}$ is an antitone strict supermartingale.

For elements of the probability space where $T \leq n$, $\mathbb{P}[Y_{n+1} \leq Y_n - d(Y_n) \mid \mathcal{F}_n] \geq p(Y_n) \cdot \mathbf{1}_{\{T>n\}}$ trivially since $\mathbf{1}_{\{T>n\}}$ is 0 there, and a probability is always non-negative. For the cases where $T > n$ (i.e. since random variables are implicitly functions, all of the random variables below refer to the restrictions of those functions to the subset of the probability space where $T > n$),

$$\begin{aligned}
&\mathbb{P}[Y_{n+1} \leq Y_n - d(Y_n) \mid \mathcal{F}_n] \\
=&\mathbb{E}\left[\mathbf{1}_{Y_{n+1}\leq Y_n - d(Y_n)} \mid \mathcal{F}_n\right] \\
=&1 - \mathbb{E}\left[\mathbf{1}_{Y_{n+1}> Y_n - d(Y_n)} \mid \mathcal{F}_n\right] \\
\geq&1 - \mathbb{E}\left[\frac{Y_{n+1}}{Y_n - d(Y_n)} \mid \mathcal{F}_n\right] \\
=&1 - \frac{\mathbb{E}\left[Y_{n+1} \mid \mathcal{F}_n\right]}{Y_n - d(Y_n)} \\
\geq&1 - \frac{Y_n - \varepsilon(Y_n)\cdot \mathbf{1}_{T>n}}{Y_n - d(Y_n)} \\
=&1 - \frac{Y_n - \varepsilon(Y_n)}{Y_n - d(Y_n)} \\
=&1 - \frac{Y_n - \varepsilon(Y_n)}{Y_n - \frac{1}{2}\varepsilon(Y_n)} \\
=&\frac{\varepsilon(Y_n)}{2(Y_n - \frac{1}{2}\varepsilon(Y_n))} \\
>&\frac{\varepsilon(Y_n)}{2Y_n} \\
=&p(Y_n) \\
=&p(Y_n)\cdot \mathbf{1}_{\{T>n\}}
\end{aligned}$$

therefore $(Y_n)_{n\geq 0}$ is quantile strict.

Quantile ⇒ Antitone: Take the same setup as before, except $(Y_n)_{n\geq 0}$ quantile strict and p and d arbitrary functions as given in the definition of quantile strict supermartingales. Assume wlog that $d(0) < \frac{1}{2}$, since if $(Y_n)_{n\geq 0}$ is quantile strict with a larger value of d, it is also quantile strict with a smaller value. Since $Y_n \geq 0$, $\log(Y_n + 1) \geq 0$. The fact that $\log(Y_n + 1)$ is a supermartingale follows from Jensen's inequality. Define $\varepsilon(x) = E(p(e^x - 1), d(e^x - 1), e^x - 1)$, where $E(P, D, x) = P(\log(x + 1) - \frac{D}{x+1} - \log(x + 1 - D))$. Evaluating a few of the derivatives of E shows that ε is positive and antitone as required.

$$
\begin{aligned}
&E(P, 0, x) = 0\\
&\frac{\mathrm{d}}{\mathrm{d}D}E(P, D, x) = P(-\frac{1}{x+1} + \frac{1}{x+1-D}) = \frac{PD}{(x+1)(x+1-D)}\\
&\frac{\mathrm{d}}{\mathrm{d}D}E(P, D, x)|_{D=0} = 0\\
&\frac{\mathrm{d}}{\mathrm{d}D}E(P, D, x) > 0 \text{ for } P, D > 0, x \geq 0\\
&\frac{\mathrm{d}}{\mathrm{d}P}E(P, D, x) = \log(x + 1) - \frac{D}{x+1} - \log(x + 1 - D) \geq 0 \text{ for } D > 0, x \geq 0\\
&\frac{\mathrm{d}}{\mathrm{d}x}E(P, D, x) = P\left(\frac{1}{x+1} + \frac{D}{(x+1)^2} - \frac{1}{x+1-D}\right)\\
&\qquad\qquad = PD\left(\frac{1}{(x+1)^2} - \frac{1}{(x+1)(x+1-D)}\right) < 0 \text{ for } P, D > 0, x \geq 0
\end{aligned}
$$

Now the antitone strict condition follows reasonably directly, using the approximation $\log(x' + 1) \leq \log(x + 1) + \frac{x'-x}{x} - [x' < x - d(x)]E(1, d(x), x)$.

$$
\begin{aligned}
&\mathbb{E}\left[\log(Y_{n+1} + 1) \mid \mathcal{F}_n\right]\\
\leq&\mathbb{E}\left[\log(Y_n + 1) + \frac{Y_{n+1} - Y_n}{Y_n} - \mathbf{1}_{Y_{n+1}\leq Y_n - d(Y_n)}E(1, d(Y_n), Y_n) \mid \mathcal{F}_n\right]\\
=&\log(Y_n + 1) + \frac{\mathbb{E}\left[Y_{n+1} \mid \mathcal{F}_n\right] - Y_n}{Y_n} - \mathbb{P}[Y_{n+1} \leq Y_n - d(Y_n) \mid \mathcal{F}_n]E(1, d(Y_n), Y_n)\\
\leq&\log(Y_n + 1) - p(Y_n)\mathbf{1}_{T>n}E(1, d(Y_n), Y_n)\\
=&\log(Y_n + 1) - \varepsilon(\log(Y_n + 1)) \cdot \mathbf{1}_{T>n}
\end{aligned}
$$

□

This theorem implies in particular that any stopping time that can be proven almost surely finite using an antitone strict supermartingale can also be proven a.s. finite with a quantile strict supermartingale and vice-versa.

3.3.2 Examples

Now that almost the complete result has been proven (there is one more extension to go, described in Chapter 4, but its proof will be much more complicated),

here are some examples of how the ranking function method can be applied to some simple terms, some of which are particularly relevant to comparing it to previous methods for proving AST.

Example 3.3.2 (Random walk biased towards zero)**.**

$$M_1 = \Theta_1\,\underline{10}$$
$$\Theta_1 = \mathsf{Y}\,\lambda fn.\,\mathsf{if}(n = 0, 0, f\,(n-1) \oplus_{2/3} f\,(n+1))$$

This can be seen as a prototypical example of a rankable function, since the whole structure of the program is just changing some numerical value, decreasing it on average, repeatedly until it reaches 0. It therefore has a reasonably simple sparse ranking function

$$\Theta_1\,\underline{x} \mapsto 3x+1$$
$$\Theta_1\,(\underline{x}+1) \mapsto 3(x+1)+1$$
$$\Theta_1\,(\underline{x}-1) \mapsto 3(x-1)+1$$

This could equivalently be considered an antitone sparse ranking function with the constant antitone function $\varepsilon_1(x) = 1$. It is a little unsatisfactory because of the three cases that are logically basically the same thing. It would be nice to just have $\Theta_1\,\underline{x} \mapsto 3x+1$, but $\Theta_1\,(\underline{x}+1)$ actually reduces to $(\lambda z.(\lambda n.\mathsf{if}(n = 0, 0, \Theta_1\,(n-1) \oplus_{2/3} \Theta_1\,(n+1)))z)\,(\underline{x}+1)$, not $\Theta_1\,\underline{x+1}$ (and similarly for $\Theta_1\,(\underline{x}-1)$). The sparse ranking function could also just be defined as $(\lambda z.(\lambda n.\mathsf{if}(n = 0, 0, \Theta_1\,(n-1) \oplus_{2/3} \Theta_1\,(n+1)))z)\,\underline{x} \mapsto 3x+1$, but that is also a little more complicated than it feels ought to be necessary to express why M_1 is AST. In the rest of the examples in this section I will be ignoring this issue and pretending that the reduction order is the version that makes the ranking functions simpler. You can either treat the actual ranking function as implicit, or see Theorem 4.4 for the rigorous justification that this is actually fine.

Example 3.1.2, the unbiased random walk, revisited

$$M_2 = \Theta_2\,\underline{10}$$
$$\Theta_2 = \mathsf{Y}\,\lambda fn.\,\mathsf{if}(n = 0, 0, f\,(n-1) \oplus_{1/2} f\,(n+1))$$

This differs from M_1 only in that it has $\oplus_{1/2}$ instead of $\oplus_{2/3}$. It is more tricky because unlike M_1 it is not Y-PAST, so an antitone ranking function is necessary. Define the function $g_2 : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ by $g_2(x) = \ln(x+1)+1$. We can define an antitone sparse ranking function $f_2 : Rch(M_2) \rightharpoonup \mathbb{R}_{\geq 0}$ by $f_2(\Theta_2\,\underline{n}) = g_2(n)$ for $n \in \mathbb{N}$ and $\underline{0} \mapsto 0$.

For $n \geq 1$, $\Theta_2\,\underline{n}$ reduces in several steps to either $\Theta_2\,(\underline{n}-1)$ or $\Theta_2\,(\underline{n}+1)$,

each with probability 1/2, with one Y-reduction. Now

$$\begin{aligned}
&g_2(n) - \frac{g_2(n-1) + g_2(n+1)}{2} \\
&= \ln\Big(\frac{n+1}{\sqrt{n(n+2)}}\Big) \\
&= \frac{1}{2}\ln\left(\frac{(n+1)^2}{n(n+2)}\right) \\
&= \frac{1}{2}\ln\left(1 + \frac{1}{n(n+2)}\right) \\
&> \frac{1}{n(n+2)+1} \\
&> \frac{1}{2}\frac{1}{(n+1)^2} + \frac{1}{2}\frac{1}{(n+3)^2}
\end{aligned}$$

Moreover $\Theta_2\,\underline{0}$ reduces to $\underline{0}$ with one Y-reduction with a reduction in g_2 of 1. Therefore by setting $\varepsilon_2(x) = \frac{1}{(e^{x-1}+1)^2}$, the condition on how much g_2 must decrease is met. The function f_2 is also defined at $M_2 = \Theta_2\,\underline{10}$, and is non-negative, therefore it is an antitone sparse ranking function, and M_2 is AST.

In this example, it would clearly be simpler to give a sort of ranking function with something like the quantile strict condition instead of the antitone strict condition actually used in Theorem 3.9, but the same sort of trick of taking the logarithm worked here as in proving the equivalence of antitone strict supermartingales and quantile strict supermartingales in general (Theorem 3.11). It is perhaps a somewhat excessive amount of algebra for such a simple example, so some solution to the issues with the quantile strict condition mentioned in Section 3.3.1 would be useful if it could be found.

Sampling from continuous distributions is an essential feature of statistical probabilistic programming languages. (See e.g. Church [25], Stan [9], Anglican [50], Gen [17], Pyro [7], Edward [51] and Turing [23].) Methods of proving AST of probabilistic computation have been developed for probabilistic programs with discrete distributions (see e.g. [18, 31, 44, 33, 30, 39]). To our knowledge, the problem of proving AST of probabilistic functional programs with continuous distribution is new.

Example 3.3.3 (Continuous random walk)**.** In PPCF we can construct a function whose argument changes by a random amount at each recursive call: $\Theta\,\underline{10}$ where

$$\Theta := \mathsf{Y}\,\lambda f x.\ \mathsf{if}(x \leq 0, \underline{0}, f(x - \mathsf{sample}))$$

We can construct a sparse ranking function f for $\Theta\,\underline{10}$ as follows:

$$\begin{aligned}
\Theta\,\underline{l} &\mapsto l + 2 \\
\mathsf{if}(\underline{l} \leq 0, 0, \Theta\,(\underline{l} - \mathsf{sample})) &\mapsto l + 1 \\
\underline{0} &\mapsto 0.
\end{aligned}$$

For a more complex (not Y-PAST) example, consider the following continuous random walk: $\Xi\,\underline{10}$ where

$$\Xi := \mathsf{Y}\,\lambda f x.\ \mathsf{if}(x \le 0, \underline{0}, f(x - \mathsf{sample} + 1/2))$$

Let $g(x) := 2 + \ln(x+1)$, and let ε be the function specified by

$$\varepsilon(g(x-1/2)) = g(x) - \int_{x-1/2}^{x+1/2} g(y)\,\mathrm{d}y.$$

The limit of this as $x \to \infty$ and $g(x+1/2) \to \infty$ is 0, and $\frac{\mathrm{d}}{\mathrm{d}x}\varepsilon(g(x+1/2)) = \frac{1}{x+1} + \ln(1 - \frac{1}{x+1/2}) < 0$, and g is monotonic increasing; therefore ε is antitone and bounded below by 0. We define an antitone sparse ranking function by:

$$\Xi\,\underline{l} \mapsto g(l)$$
$$\underline{0} \mapsto 0$$

The value of g after one Y-reduction step is at least $g(l-1/2)$, therefore the expectation of ε after one Y-reduction step is at most $\varepsilon(g(l-1/2)) = g(x) - \int_{x-1/2}^{x+1/2} g(y)\,\mathrm{d}y$. Thus

- in case $l > 0$, g decreases by the required amount
- in case $l \le 0$, $g(\Xi\, l) \ge 2 + \ln(1/2) > \ln(\frac{3\sqrt{3}}{2}) - 1 = \varepsilon(g(0-1/2))$ as well.

Hence this is a valid antitone sparse ranking function and the term is AST.

Example 3.3.4 (Fair-in-the-limit random walk)**.** [40, §5.3]

$$\left(\mathsf{Y}\ \lambda f x.\ \mathsf{if}(x \le 0, \underline{0}, f(x-1) \oplus_{\frac{x}{2x+1}} f(x+1))\right) \underline{10}$$

To construct an antitone ranking function, we solve the recurrence relation:

$$z_0 = 0$$
$$n \ge 1, \quad z_n > \tfrac{n}{2n+1}\, z_{n-1} + \tfrac{n+1}{2n+1}\, z_{n+1}.$$

For $n > 2$, $z_n = \ln(n-1)$ works. The expected decrease is $\frac{1}{2}(\ln(1 + \frac{1}{(n-1)^2-1}) - \frac{1}{2n+1}\ln(1+\frac{2}{n-2}))$, and using the fact that $\frac{x}{x+1} \le \ln(1+x) \le x$ for $x > 0$, this is at least $\frac{1}{2}(\frac{1}{(n-1)^2} - \frac{1}{2n+1}\frac{2}{n-2}) = \frac{n^2}{2(n-1)^2(2n+1)(n-2)}$, which (again for $n > 2$) is positive and antitone. For $n = 0, 1, 2$, we take ε to be $9/40$ (the same as its value at 3), then set $z_2 = 2\ln 2 - \ln 3 - 1$, $z_1 = 3\ln 2 - 2\ln 3 - 3$, $z_0 = 4\ln 2 - 3\ln 3 - 6$. Some of those values are negative, but it's still bounded below, so by adding a constant offset it can be corrected, and the term is AST.

Example 3.3.5 (Escaping spline)**.** [40, §5.4]

$$\underbrace{\left(\mathsf{Y}\,\lambda f x.\, 0 \oplus_{\frac{1}{x+1}} f(x+1)\right)}_{\Xi}\ \underline{10}$$

In this case, the fact that the ranking function must decrease at each Y-step (in an antitone sparse ranking function) by the expected value of ε not at the current term, but at the next term where the ranking function is defined, is a little harder to deal with, because the variable x can change all the way to 0 in one step, therefore simply adding a small offset doesn't suffice to compensate for this fact.

Consider the candidate ranking function $\Xi\,\underline{n} \mapsto n{+}1$. For each n, $\Xi\,\underline{n}$ reduces to either $\underline{0}$ (with probability $\frac{1}{n+1}$) or $\Xi\,\underline{n+1}$, therefore the expected value of the ranking function is $\frac{n(n+2)}{n+1} = n + 1 - \frac{1}{(n+1)^2}$, and the required decrease is $\frac{\varepsilon(0)+n\varepsilon(n+2)}{n+1}$, therefore eventually, the expected decrease isn't enough, whatever the value of $\varepsilon(0)$ is.

If instead, we take the ranking function to be defined after the Y-reduction but before the sample-reduction as well, this can be resolved. Letting $\Theta[n] = \underline{0} \oplus_{\frac{1}{n}} \Xi\,n$:

$$\Theta[\underline{n}] \mapsto n$$
$$\Xi\,\underline{10} \mapsto 12.$$

For each n, $\Theta[n]$ reduces to either 0 or $\Theta[n+1]$, with a Y-reduction only in the latter case, and the condition that this is an antitone sparse ranking function is that $n \geq \frac{(n-1)(n+1)}{n} + \frac{n-1}{n}\varepsilon(n+1)$ and $12 \geq 11 + \varepsilon(11)$. These are satisfied by setting $\varepsilon(x) = \min(1, 1/x)$, therefore this term is AST.

Many of the recent advances in the development of AST verification methods [14, 11, 22, 40, 28, 13, 29] are concerned with loop-based programs. We can view such loops as tail-recursive programs that are, in particular, *affine recursive*, i.e., in each evaluation (or run) of the body of the recursion, recursive calls are made from at most one call site [18, §4.1]. (Note that whether a program is affine recursive cannot be checked by just counting textual occurrences of variables, since some parts of a term may end up being discarded without evaluation, for example in a if construct, and similarly, the recursive variable may be deleted or duplicated by a β or Y reduction.) Termination analysis of *non-affine recursive* probabilistic programs does not seem to have received much attention. Methods such as those presented in [18] are explicitly restricted to affine programs, and are unsound otherwise. By contrast, many probabilistic programming languages allow for richer recursive structures [50, 25, 37]. All of the examples shown so far in this section have had basically the same simple affine recursive structure, equivalent to a while loop with a single real variable, but the method of ranking functions accommodates the more variable structures of non-affine recursive programs as well.

Example 3.3.6 (Non-affine recursion)**.**

$$\Xi = \mathsf{Y}\lambda f x.x \oplus_{2/3} f(f(x+1))$$

A suitable sparse ranking function for $\Xi\,\underline{1}$ would be $\Xi^n\,\underline{i} \mapsto 3\,n$ (for $n \geq 0$), because $\Xi^n\,\underline{i}$ reduces to either $\Xi^{n+1}\,\underline{i+1}$ or $\Xi^{n-1}\,\underline{i+1}$, with probabilities 1/3

and 2/3. The value of i does not actually matter for the progress of this recursion. It is basically another variant of the biased random walk, except that the relevant variable is the number of copies of Ξ, instead of a real number in the term.

Example 3.3.7 (More complex non-affine recursion)**.**

$$(\mathsf{Y}\,\lambda f x.\,(\lambda e.X[e])\,\mathsf{sample})\,\underline{0} \quad \text{where}$$
$$X[e] := \mathsf{if}(e \le p - x^{-2}, x, f^2(x+1) \oplus_e f(x+1))$$

This example is a little more complex because of the use of a random sample, e, as a first class value, which cannot be modelled via discrete distributions.

We can use the ranking function method (coupled with the solution of linear recurrence relations) to show that provided $p \ge \frac{5-\sqrt{21}}{2}$, the program $\Xi\,\underline{3}$ is AST. Consider the edge case, that $p = \frac{5-\sqrt{21}}{2}$ exactly.

The term $\Xi^n\,\underline{x}$ (for $n > 0$, $x^{-2} < p$) reduces to either $\Xi^{n-1}\,\underline{x+1}$, $\Xi^n\,\underline{x+1}$ or $\Xi^{n+1}\,\underline{x+1}$, with probabilities $p - x^{-2}$, $(1-p+x^{-2})\frac{p-x^{-2}}{2}$ and $(1-p+x^{-2})(1-\frac{p-x^{-2}}{2})$ respectively. Let $a(\Xi^n\,\underline{x}) = n + 2\,x^{-1}$. This is a supermartingale (because the decrease in $2\,x^{-1}$ as x increases is enough to offset the average increase in n), but it does not satisfy the antitone-strict progress condition. It does however have a bounded-below variance, so the usual method of using $\ln a$ instead of just a works. Calculating the exact amount that $\ln a$ decreases is not necessary, because it can be bounded as follows: a changes by at least $1/2$ with probability at least $1/11$ (assuming $x \ge 3$), therefore $\ln(a)$ decreases in expectation by at least $\frac{1}{11}(\ln(a+\frac{1}{2}) - \ln(a) - \frac{1}{2a})$, using the fact that a linear approximation to ln, applied to a, at least doesn't increase, then adding the deviation of ln from its linear approximation $\ln(a) + \frac{x-a}{a}$ (and using the fact that the linear approximation is everywhere an overestimate), we obtain a sufficiently strong bound on the decrease of $\ln(a)$ that it's an antitone sparse ranking function. (It can obviously be extended to $\Xi\,\underline{1}$ too, which reduces to $\Xi^n\,\underline{3}$ or a value after a bounded number of steps, but that complicates the analysis a little.)

Example 3.3.8 (Higher-order recursion)**.** Previous versions of the ranking function method such as [40], which use an imperative language, are naturally restricted to first-order programs, but this is not an inherent limitation of the method as applied to PPCF. Consider the higher-order function $\Xi : (\mathsf{R} \to \mathsf{R} \to \mathsf{R}) \to \mathsf{R} \to \mathsf{R} \to \mathsf{R}$ recursively defined[4] by

$$\Xi := \mathsf{Y}\lambda\varphi\, f^{\mathsf{R}\to\mathsf{R}\to\mathsf{R}}\, s^{\mathsf{R}}\, n^{\mathsf{R}}.$$
$$\mathsf{if}(n \le 0, s, f\,n\,(\varphi\,f\,s\,(n-1)) \oplus_p f\,n\,(\varphi\,f\,s\,(n+1)))$$

Take any value $F : \mathsf{R} \to \mathsf{R} \to \mathsf{R}$ such that $F\,\underline{n}\,\underline{m}$ terminates with no Y reductions for all m and n. Then $\Xi\ F\ \underline{x}\ \underline{y}$ has the antitone sparse ranking function

$$F\,\underline{n_0}\,(F\,\underline{n_1}\,(\dots(\Xi\ F\,\underline{x}\,\underline{n})\dots) \mapsto g_2(n),$$

[4]Inspired by examples in [10, 45].

where g_2 is as in Example 3.1.2, using the same antitone function too.

The inequalities required for this to be an antitone sparse ranking function are satisfied with just the same reasoning as in Example 3.1.2, therefore this term too is antitone rankable.

Again, this is not actually the correct reduction order, in that F should be applied to its argument n_i and reduced to a value before its second argument is expanded, but this would complicate the presentation, and this version can be justified by Theorem 4.4 instead (the details of this are given in Example 4.5.2).

Note that for this example, it is essential that the ranking function is sparse, because that allows all of the details of the reduction of F to be skipped over with just the assumption that it terminates with no Y reductions. It would be at best very awkward to define a fully explicit antitone ranking function while F is unknown.

3.3.3 Completeness

Unlike for ranking functions, which only exist for terms which terminate quickly enough (i.e. are Y-PAST), antitone ranking functions can be constructed for arbitrarily slowly terminating terms by varying the antitone function ε. Indeed, Theorem 3.9 is complete, in the sense that

Theorem 3.12 (Antitone Ranking Completeness)**.** *Every (closed) AST term is antitone rankable.*

Proof. Take any closed AST term M. The antitone function ε that fits this term will be provided later, but for now let's just take it as given and assume it is antitone, convex, continuous, and small enough (in a sense that will only become clear later), and use it to define the ranking function. Define the random variable

$$T_n^{\mathsf{Y}}(s) = |\{k \geq n \mid M_k(s) \text{ is of the form } E[\mathsf{Y}\lambda x.N]\}|$$

where $M_k(s)$, as usual, is $\pi_0(\mathrm{red}^k(M, s))$. Note that, for any fixed $N \in Rch(M)$, the probability distribution of T_n^{Y}, conditional on $M_n = N$, is independent of n and all of the intermediate terms M_k for $0 \leq k \leq n$, since this is simply the Y stopping time of N, T_N^{Y} (although T_n^{Y} where $M_n = N$ does not have the same values on individual traces s as T_N^{Y}, only the same distribution). If N terminates deterministically after n Y-reduction steps, it must have a rank of at least y_n, where $(y_n)_{n\in\mathbb{N}}$ satisfies the recurrence relation

$$y_{n+1} - \varepsilon(y_{n+1}) = y_n, \quad y_0 = 0.$$

(The fact that this has a solution follows from the fact that ε is continuous, bounded below by 0, and antitone.) The full ranking function for all the reachable terms can then be defined as

$$f(N) = \mathbb{E}[y_{T_N^{\mathsf{Y}}}]$$

with the caveat that this expectation may be infinite, in which case $f(N)$ takes some arbitrary value, say 0, instead. This is okay so long as the set of terms

where $\mathbb{E}[y_{T_N^{\mathsf{Y}}}]$ is actually finite is an almost complete reachable set (this is the almost complete reachable set C that will be used for the antitone ranking function).

In particular, it is required that $\mathbb{E}[y_{T_M^{\mathsf{Y}}}] = \mathbb{E}[y_{T_0^{\mathsf{Y}}}]$ is finite. This is the condition for ε to be small enough, so let's now construct a suitable ε. Take an increasing sequence of natural numbers $(t_n)_{n\in\mathbb{N}}$ such that $t_0 = 0$ and $\mathbb{P}[T_M^{\mathsf{Y}} > t_n] \leq 2^{-n}$ (this is possible because the almost-sure termination of M implies $\lim_{n\to\infty} \mathbb{P}[T_M^{\mathsf{Y}} \geq n] = 0$), then define ε as the function whose graph is the convex hull of the points $\left(n, \frac{1}{t_{n+1}-t_n}\right)$, or more precisely,

$$\varepsilon(x) = \sup\left\{ z \in \mathbb{R} \;\middle|\; \exists m \in \mathbb{R}. \forall n \in \mathbb{N}.\, m(n-x) + z \leq \frac{1}{t_{n+1}-t_n} \right\}.$$

The inner formula $\forall n \in \mathbb{N}.\, m(n-x) + z \leq \frac{1}{t_{n+1}-t_n}$ is never satisfied for any positive m because $\frac{1}{t_{n+1}-t_n}$ is bounded above by 1, therefore the formula is antitone in x (if it is satisfied for some x, it will also be satisfied for the same z and m with smaller x), therefore the set $\{z \in \mathbb{R} \mid \dots\}$ and its supremum $\varepsilon(x)$ are also antitone in x. It is also positive (since $z = \inf_{n<2x} \frac{1}{2(t_{n+1}-t_n)} > 0$ is in the set with $m = -\frac{z}{x}$) and convex, since if $(x_1, \varepsilon(x_1))$ is strictly above the line passing through two points on the graph either side of it, $(x_0, \varepsilon(x_0)), (x_2, \varepsilon(x_2))$, then since there are straight lines passing arbitrarily close to $(x_1, \varepsilon(x_1))$ that pass below (or touch) each of the points $\left(n, \frac{1}{t_{n+1}-t_n}\right)$, one of them must pass above either $(x_0, \varepsilon(x_0))$ or $(x_2, \varepsilon(x_2))$, which is a contradiction. The fact that it is continuous follows from its convexity (with the special case of 0 at the edge of the domain being easy to check). It also has the property that $\varepsilon(n) \leq \frac{1}{t_{n+1}-t_n}$ for $n \in \mathbb{N}$, from which it will be possible to prove that $\mathbb{E}[y_{T_M^{\mathsf{Y}}}]$ is finite.

Define $g_k = \frac{k-t_n}{t_{n+1}-t_n}$ for $t_n \leq k \leq t_{n+1}$. This sequence satisfies the same recurrence relation as $(y_n)_{n\in\mathbb{N}}$ does, but with an inequality, $g_{n+1} - \varepsilon(g_{n+1}) \geq g_n$, because ε is antitone and $\varepsilon(n) \leq \frac{1}{t_{n+1}-t_n}$. Since the recurrence relation gives a value of y_{n+1} that is monotonic increasing as a function of y_n, it follows that $y_n \leq g_n$. From these results, we get

$$
\begin{aligned}
& \mathbb{E}[y_{T^{\mathsf{Y}}_M}] \\
\leq\ & \mathbb{E}[g_{T^{\mathsf{Y}}_M}] \\
=\ & \sum_{n=0}^{\infty} g_n \mathbb{P}[T^{\mathsf{Y}}_M = n] \\
=\ & \sum_{k=0}^{\infty} (g_{k+1} - g_k) \mathbb{P}[T^{\mathsf{Y}}_M > k] \\
=\ & \sum_{n=0}^{\infty} \sum_{k=t_n}^{t_{n+1}-1} (g_{k+1} - g_k) \mathbb{P}[T^{\mathsf{Y}}_M > k] \\
\leq\ & \sum_{n=0}^{\infty} \sum_{k=t_n}^{t_{n+1}-1} \frac{1}{t_{n+1} - t_n} \mathbb{P}[T^{\mathsf{Y}}_M > t_n] \\
\leq\ & \sum_{n=0}^{\infty} \frac{t_{n+1} - t_n}{t_{n+1} - t_n} 2^{-n} \\
=\ & 2 < \infty
\end{aligned}
$$

as required.

From this, it follows reasonably easily that $\mathbb{E}[y_{T^{\mathsf{Y}}_N}]$ is finite for almost all reachable N, not just M itself.

$$
\begin{aligned}
& \mathbb{E}[y_{T^{\mathsf{Y}}_M}] \\
=\ & \mathbb{E}[y_{T^{\mathsf{Y}}_0}] \\
\geq\ & \mathbb{E}[y_{T^{\mathsf{Y}}_n}] \\
=\ & \mathbb{E}[\mathbb{E}[y_{T^{\mathsf{Y}}_n} \mid \mathcal{F}_n]] \\
=\ & \mathbb{E}[\mathbb{E}[y_{T^{\mathsf{Y}}_{M_n}}]] \\
\geq\ & x \mathbb{P}[\mathbb{E}[y_{T^{\mathsf{Y}}_{M_n}}] \geq x]
\end{aligned}
$$

therefore $\mathbb{P}[\mathbb{E}[y_{T^{\mathsf{Y}}_{M_n}}] \geq x] \to 0$ as $x \to \infty$, therefore $\mathbb{E}[y_{T^{\mathsf{Y}}_{M_n}}]$ is almost surely finite, exactly what is needed to ensure that C, the set of terms where $\mathbb{E}[y_{T^{\mathsf{Y}}_N}]$ is actually finite, is an almost complete reachable set.

This establishes that the purported antitone ranking function f is well-defined, with a suitable ε and C, so all that remains is to check that it actually satisfies the antitone ranking function condition. Take some term in C of the form $E[R]$ where R is a redex.

In the simplest case, that $R \to R'$ is not a sample or Y reduction, it is required that $f(E[R]) \geq f(E[R'])$. $T^{\mathsf{Y}}_{E[R]} = T^{\mathsf{Y}}_{E[R']}$, therefore $\mathbb{E}[y_{T^{\mathsf{Y}}_{E[R]}}] = \mathbb{E}[y_{T^{\mathsf{Y}}_{E[R']}}]$, therefore since the former is finite (by assumption, $E[R] \in C$), so is the latter, and $f(E[R']) = \mathbb{E}[y_{T^{\mathsf{Y}}_{E[R']}}] = \mathbb{E}[y_{T^{\mathsf{Y}}_{E[R]}}] = f(E[R])$.

In the case that $R = \mathsf{sample}$, $T^{\mathsf{Y}}_{E[R]}(s) = T^{\mathsf{Y}}_{E[\underline{s(0)}]}(\pi_t(s))$ therefore, since $s(0)$ and $\pi_t(s)$ are independent and $\pi_t(s)$ has the same distribution as s,

$$\begin{aligned}
&\mathbb{E}_s[T^{\mathsf{Y}}_{E[R]}(s)] \\
=&\mathbb{E}_s[T^{\mathsf{Y}}_{E[\underline{s(0)}]}(\pi_t(s))] \\
=&\int_I \mathbb{E}_{\pi_t(s)}[T^{\mathsf{Y}}_{E[\underline{x}]}\pi_t(s)]\,\mathrm{d}x \\
=&\int_I \mathbb{E}[T^{\mathsf{Y}}_{E[\underline{x}]}]\,\mathrm{d}x
\end{aligned}$$

Now, since this integral has a finite value, the set of xs for which $\mathbb{E}[T^{\mathsf{Y}}_{E[\underline{x}]}] = \infty$ (and equivalently $\mathbb{E}[T^{\mathsf{Y}}_{E[\underline{x}]}] \neq f(E[\underline{x}])$) must be a null set, therefore the integral is equal to $\int_I f(E[\underline{x}])\,\mathrm{d}x$, as required.

In the case that $R \to R'$ with a Y reduction, it is required that $f(E[R]) - \varepsilon(f(E[R])) \geq f(E[R'])$. Note that $T^{\mathsf{Y}}_{E[R']} = T^{\mathsf{Y}}_{E[R]} - 1$. From Jensen's inequality, with ε convex, it follows that

$$\begin{aligned}
&f(E[R]) - \varepsilon(f(E[R])) \\
=&\mathbb{E}[y_{T^{\mathsf{Y}}_{E[R]}}] - \varepsilon(\mathbb{E}[y_{T^{\mathsf{Y}}_{E[R]}}]) \\
\geq&\mathbb{E}[y_{T^{\mathsf{Y}}_{E[R]}}] - \mathbb{E}[\varepsilon(y_{T^{\mathsf{Y}}_{E[R]}})] \\
=&\mathbb{E}[y_{T^{\mathsf{Y}}_{E[R]}} - \varepsilon(y_{T^{\mathsf{Y}}_{E[R]}})] \\
=&\mathbb{E}[y_{T^{\mathsf{Y}}_{E[R]}-1}] \\
=&\mathbb{E}[y_{T^{\mathsf{Y}}_{E[R']}}] \\
=&f(E[R'])
\end{aligned}$$

where as in the other cases, we have used the intermediate result that $\mathbb{E}[y_{T^{\mathsf{Y}}_{E[R']}}]$ is less than or equal to $f(E[R])$ and therefore finite to prove that $\mathbb{E}[y_{T^{\mathsf{Y}}_{E[R']}}] = f(E[R'])$. □

Chapter 4

Confluent Trace Semantics

When proving almost sure termination using ranking functions, it is necessary to consider all of the terms a given term may reduce to, and organise them into a manageable number of cases to assign ranking function values. Sometimes however, the reduction that the programmer has in mind may not be strictly the call-by-value order defined in Section 2.4; or the number of cases necessary for defining a sparse ranking function is excessive because the reduction does not proceed in a neat and orderly manner. This was seen in most of the examples in Section 3.3.2. Take Example 3.3.2,

$$M_1 = \Theta_1\,\underline{10}$$
$$\Theta_1 = \mathsf{Y}\,\lambda fn.\,\mathsf{if}(n = 0, 0, f\,(n-1) \oplus_{2/3} f\,(n+1))$$

It would be convenient to just take $\Theta_1\,\underline{x} \mapsto 3x+1$ as the sparse ranking function, but that is not technically correct, because $\Theta_1\,\underline{x}$ (for $x > 0$) reduces to either $\Theta_1\,(\underline{x}+1)$ or $\Theta_1\,(\underline{x}-1)$, but then it is the Θ_1 subterm that is reduced next, not $\underline{x} \pm 1$, so $\Theta_1\,\underline{x \pm 1}$ is skipped entirely. Intuitively, the precise reduction order isn't actually important here, and in general the programmer may have in mind a reduction order that is different from the actual call-by-value order that PPCF is defined with, so it would be nice to be able to define the (antitone) (sparse) ranking function with respect to that reduction strategy instead. In this example, the difference in reduction strategies is pretty trivial, but for other terms (e.g. Example 4.5.1), not only will the number of cases the reachable terms may be organised into be much less when using an alternative reduction strategy constructed to match the term in question, but the runtime with the alternative strategy is so much lower that the term is Y-PAST (and therefore rankable) with that alternative strategy but not the standard one. Of course, this means that alternative reduction strategies are useless for analyzing the runtime, but for termination, despite PPCF's probabilistic nature, there is a limited confluence result that will relate termination under the usual definition with termination under alternative reduction strategies (Theorem 4.3).

In order to be able to use alternative reduction strategies to prove properties of the program defined using the usual reduction strategy (as defined in Section 2.4), in this section I will first prove the relation between these different strategies, providing a confluent version of the semantics that encompasses both the standard reduction order and any valid alternative reduction orders. In order for confluence to work in a trace-based semantics, it is necessary to have some way of relating the samples that will be used in runs of the program using different reduction strategies. First, positions are defined as a way of addressing sample statements within a program independently of the reduction order. Next, a version of the reduction relation is presented that is nondeterministic, so that it allows a choice of what order to perform reductions in. The notion of positions is extended to potential positions, for samples which may appear later in the reduction sequence but not necessarily in the initial term. In order to allow potential positions in different reduction sequences to be considered equivalent, a relation $\sim^*$ is defined, and finally, all of these are used to construct a confluent version of the trace semantics, $\Rightarrow$, that is still nondeterministic in reduction order, but does specify the outcome of random choices. The desired properties of this semantics (confluence and equivalence to the standard semantics) are proved (Theorems 4.1 and 4.2), then it is used to give extended versions of ranking functions and the related AST results, Theorem 4.4.

4.1 The Failure of Confluence

Non-probabilistic lambda calculi are generally confluent, i.e. if a term A reduces to both B_1 and B_2 (possibly with multiple steps), there is some C to which both B_1 and B_2 reduce, so the reduction order mostly doesn't matter. There is a relatively trivial way in which confluence fails to hold in the probabilistic case, in that sample can reduce to either $\underline{0.1}$ or $\underline{0.2}$, but neither of these can reduce to anything else. This can be alleviated just by requiring that the same samples be used in each reduction sequence (although, as will be seen later, it can be quite complicated to define what counts as the same samples), or by considering distributions instead of trace-based semantics (a confluent distribution-based semantics is described in [21], but since it is the trace-based semantics that the ranking function method follows from, I will not be considering this possibility further here). In addition to this, there is a complication from the fact that β-reduction and Y-reduction can duplicate samples, so the outputs of the copies of the sample may be identical or independent, depending on whether the sample is taken before or after the β or Y reduction, as shown in the example.

Example 4.1.1. Consider $M = (\lambda x.x - x)\ \mathsf{sample}$. If we ignore the rules of call-by-value reduction, this can be reduced either by first reducing the sample, as $(\lambda x.x - x)\ \mathsf{sample} \to (\lambda x.x - x)\ \underline{r} \to \underline{r} - \underline{r} \to \underline{0}$, or by reducing the β redex first, like $(\lambda x.x - x)\ \mathsf{sample} \to \mathsf{sample} - \mathsf{sample} \to \underline{r} - \mathsf{sample} \to \underline{r} - \underline{t} \to \underline{r - t}$. No matter which samples in each reduction sequence are identified, these two sequences lead to different distributions of results, so confluence fails in this

case. $N = (\lambda f.f\underline{0} - f\underline{0})\ (\lambda\_.\mathsf{sample})$ is a similar case, except that the CBV result involves a single sample for M and two separate samples for N.

This issue can be avoided by restricting our attention to only a subset of reduction strategies. Clearly not all reductions fail to be confluent (at a minimum, the reductions of non-random terms that do not contain any samples are). It will turn out that in order to remain within the equivalence class of reduction strategies that are confluent with the canonical CBV semantics, it is sufficient to only permit β reduction where the argument is a value, and only permit sample reduction that is not inside a λ inside the argument of a β or Y redex. This deals with both cases in Example 4.1.1. The β reduction in M is forbidden until after the sample reduction because sample is not a value, and the sample reduction in N is forbidden until after the β reduction because it is inside a λ inside the argument of a β redex.

Even with such a restriction, a trace semantics in the style of the one already defined would not be entirely confluent, as, for example, if we reduce $\mathsf{sample} - \mathsf{sample}$ using the trace $(1, 0, \dots)$, it could result in either 1 or -1 depending on the order of evaluation of the samples, as that determines which sample from the pre-selected sequence is used for each one. To fix this, rather than pre-selecting samples according to the order they'll be drawn in, we will select them according to the position in the term where they'll be used instead.

4.2 Labelling Samples

A *position* is a finite sequence of steps into a term, defined inductively as

$$\begin{aligned}\alpha ::= \cdot \mid \lambda;\alpha \mid @_1;\alpha \mid @_2;\alpha \mid \underline{f}_i;\alpha \\ \mid \mathsf{Y};\alpha \mid \mathsf{if}_1;\alpha \mid \mathsf{if}_2;\alpha \mid \mathsf{if}_3;\alpha.\end{aligned}$$

The *subterm of M at α*, denoted $M \mid \alpha$, is defined as

$$\begin{aligned} M \mid \cdot &= M \\ \lambda x.M \mid \lambda;\alpha &= M \mid \alpha \\ M_1\, M_2 \mid @_i;\alpha &= M_i \mid \alpha \quad \text{for } i = 1, 2 \\ \underline{f}(M_1, \dots, M_n) \mid \underline{f}_i;\alpha &= M_i \mid \alpha \quad \text{for } i \le n \\ \mathsf{Y}M \mid \mathsf{Y};\alpha &= M \mid \alpha \\ \mathsf{if}(M_1 < 0, M_2, M_3) \mid \mathsf{if}_i;\alpha &= M_i \mid \alpha \quad \text{for } i = 1, 2, 3 \end{aligned}$$

so that every subterm is located at a unique position, but not every position corresponds to a subterm (e.g. $x\,y \mid \lambda$ is undefined). A position such that $M \mid \alpha$ does exist is said to *occur* in M, and the set of positions that occur in M is denoted $\mathrm{pos}(M)$. *Substitution* of N at position α in M, written $M[N/\alpha]$, is defined similarly. For example, let $M = \lambda x\, y.y\,\underline{5}$ and $\alpha = \lambda;\lambda;@_2$ then $M[\mathsf{sample}/\alpha] = \lambda x\, y.y\,\mathsf{sample}$. Substitution at a position is not capture-avoiding (in contrast to substitution for a variable, $M[N/x]$), so that a modified version

of a subterm may be substituted back in, and have the free variables in the subterm become bound again in the modified version of the term. Relatedly, the definition of a subterm at a position is not invariant under α equivalence, so that $M \mid \alpha$ may contain free variables that are not free in M. This is required in Definition 4.1.

Two subterms N_1 and N_2 of a term M, corresponding to positions α_1 and α_2, can overlap in a few different ways. If α_1 is a prefix of α_2 (written as $\alpha_1 \leq \alpha_2$), then N_2 is also a subterm of N_1. If neither $\alpha_1 \leq \alpha_2$ nor $\alpha_1 \geq \alpha_2$, the positions are said to be *disjoint*. The notion of disjointness is mostly relevant in that if α_1 and α_2 are disjoint, performing a substitution at α_1 will leave the subterm at α_2 unaffected.

Thus we can define a more general reduction relation[1] $\to$.

Definition 4.1. The binary relation $\to$ is defined by the following rules, each is conditional on a redex occurring at position α in the term M:

$$\begin{aligned}
\text{if } M \mid \alpha = (\lambda x.N)V,\ & M \to M[N[V/x]/\alpha] \\
\text{if } M \mid \alpha = \underline{f}(\underline{r}_1, \ldots, \underline{r}_n),\ & M \to M[\underline{f(r_1, \ldots, r_n)}/\alpha] \\
\text{if } M \mid \alpha = \mathsf{Y}\lambda x.N,\ & M \to M[\lambda z.N[(\mathsf{Y}\lambda x.N)/x]z/\alpha] \\
& \text{where } z \text{ is not bound at position } \alpha \text{ in } M \\
\text{if } M \mid \alpha = \mathsf{if}(\underline{r} < 0, N_1, N_2),\ & M \to M[N_1/\alpha] \text{ where } r < 0 \\
\text{if } M \mid \alpha = \mathsf{if}(\underline{r} < 0, N_1, N_2),\ & M \to M[N_2/\alpha] \text{ where } r \geq 0 \\
\text{if } M \mid \alpha = \mathsf{sample} \text{ and } \lambda \text{ does not occur after } @_2 \text{ or } \mathsf{Y} \text{ in } \alpha,\ & \\
& M \to M[\underline{r}/\alpha] \text{ where } r \in [0, 1].
\end{aligned}$$

In each of these cases, $M \mid \alpha$ is the *redex*, and the reduction *takes place at* α. Each subterm can be a redex in at most one way, but there can be multiple redexes at different positions.

Example 4.2.1. Consider the term $M = \mathsf{sample}{+}(\lambda x.\mathsf{sample}{\times}x)((\lambda y.\mathsf{sample})\underline{0})$. It contains several redexes. At position $+_1$ there is a sample, so M can reduce at this position to $\underline{0} + (\lambda x.\mathsf{sample} \times x)((\lambda y.\mathsf{sample})\underline{0})$, or $\underline{0.8} + (\lambda x.\mathsf{sample} \times x)((\lambda y.\mathsf{sample})\underline{0})$, or $\underline{r} + (\lambda x.\mathsf{sample} \times x)((\lambda y.\mathsf{sample})\underline{0})$ for any other $0 \leq r \leq 1$. It can also sample reduce at $+_2; @_1; \lambda; \times_1$ to $\mathsf{sample} + (\lambda x.\underline{r} \times x)((\lambda y.\mathsf{sample})\underline{0})$, or β reduce at $+_2; @_2$ to $\mathsf{sample} + (\lambda x.\mathsf{sample} \times x)\mathsf{sample}$, but it can't β reduce at $+_2$, because although $M|{+_2}$ is an application, the argument of that application is not a value so it's not a valid β redex, and it can't sample reduce at $+_2; @_2; @_1; \lambda$, because λ occurs after $@_2$ in that position, which is forbidden for sample reductions.

The argument of a β redex and the body of a Y redex may be duplicated by those reductions. It is therefore these cases that need to be handled carefully to avoid duplicating samples at the wrong time. In both cases, the potentially

[1] In this section $\to$ always means the relation defined in Definition 4.1, rather than the original version from Section 2.4.

duplicated part must already be a value, which excludes terms like sample or $\mathsf{sample} + \underline{1}$, which should be evaluated before being duplicated. In the other direction, if a sample occurs inside of a λ, it may need to be duplicated before being evaluated, which is why a sample reduction isn't allowed inside a λ inside a Y or the right side of an application. These restrictions are in some cases unnecessarily strict, for example, in $(\lambda x.x)((\lambda y.\mathsf{sample})\underline{0})$, it would be fine to evaluate the sample first, but they are at least sufficient to ensure confluence in terms of the distribution of results. Getting individual traces to behave correctly will take more work though.

You might consider labelling the pre-chosen samples by the positions in the term by using $I^{\{\alpha \mid (M|\alpha)=\mathsf{sample}\}}$ as the trace space, so then given a trace s, if $N \mid \alpha = \mathsf{sample}$ and α is a valid position for sample reduction, N reduces to $N[\underline{s(\alpha)}/\alpha]$, or the value of s at a position that in some sense is the equivalent in the original term M to the actual position α in N might be used. This would not be sufficient to solve the issue of different samples being used in corresponding locations in different reduction sequences though, because in some cases, a sample will be duplicated before being reduced. For example, in $(\lambda x.x\,\underline{0} \mathbin{\underline{+}} x\,\underline{0})(\lambda y.\mathsf{sample})$, both of the sample redexes that eventually occur originate at $@_2;\lambda$. It is therefore necessary to consider possible positions that may occur in other terms reachable from the original term. Even this is itself inadequate because some of the positions in different reachable terms need to be considered the same, and the number of reachable terms is in general uncountable, which leads to measure-theoretic issues.

Reduction relation on skeletons Recall from Section 2.4.2 that a skeleton is like a term but with all the numbers $\underline{r}$ replaced by a placeholder X. Technically there are still uncountably many skeletons, because every measurable function $\mathbb{R}^n \to \mathbb{R}$ is available in PPCF, but from any given term, only countably many skeletons are reachable. To avoid the measure-theoretic issues, we are thus led to consider the reduction relation on skeletons (and positions in a skeleton), which can be extended from the definitions on terms in the obvious way, with $\mathsf{if}(\mathsf{X} < 0, A, B)$ reducing nondeterministically to both A and B, sample reducing to X, and X considered as (the skeletal equivalent of) a value, so that $(\lambda x.A)\,\mathsf{X}$ reduces to $A[\mathsf{X}/x]$. For example, we have

$$\begin{aligned} & (\lambda x.\mathsf{if}(x < 0, x, \mathsf{X}))\,\mathsf{sample} \\ \to\ & (\lambda x.\mathsf{if}(x < 0, x, \mathsf{X}))\,\mathsf{X} \\ \to\ & \mathsf{if}(\mathsf{X} < 0, \mathsf{X}, \mathsf{X}) \\ \to\ & \mathsf{X}. \end{aligned}$$

Given a closed term M, let $L_0(M)$ be the set of pairs, the first element of which is a $\to$-reduction sequence of skeletons starting at $Sk(M)$, and the second of which is a position in the final skeleton of the reduction sequence. As with the traces from $I^{\mathbb{N}}$ used to pre-select samples to use in the standard trace semantics in Section 2.4.2, modified traces, which are elements of $I^{L_0(M)}$ (with one more

caveat introduced after Def. 4.4), will be used to pre-select a sample from I for each element of $L_0(M)$, which will then be used if a sample reduction is ever performed at that position after those reductions.

A (skeletal) reduction sequence here is not just a sequence of terms (or skeletons), but a sequence $(M_i)_{0\leq i\leq n}$ of terms (or skeletons) along with a sequence of positions $(\alpha_i)_{0\leq i<n}$ where $M_i \to M_{i+1}$ at α_i. The positions of the reductions will almost always just be left implicit, and the reduction sequence referred to by the terms (or skeletons) alone, but it is important that two reduction sequences with the same terms but different reductions are distinct. For example, $(\lambda x.x)((\lambda x.x)\,\mathsf{X})$ could reduce to $(\lambda x.x)\,\mathsf{X}$ with the redex at either $\cdot$ or $@_2$, and these give different reduction sequences. Also, the reduction sequences of skeletons will often be discussed as though they were just skeletons, identifying them with their final skeletons. With this abuse of notation, a reduction sequence N (actually $N_1 \to^* N_n = N$) may be said to reduce to a reduction sequence O, where the reduction sequence implicitly associated with the final skeleton O is $N_1 \to^* N_n \to O$.

Example 4.2.2 (Labelling samples by terms)**.** To illustrate why it is necessary to use skeletal reduction sequences to label samples rather than just reduction sequences, consider the terms

$$\begin{aligned} A[M] &= \mathsf{if}(\mathsf{if}(M > 0, I, I)\,(\lambda y.\mathsf{sample})\,\underline{0} - \underline{0.5} > 0, \underline{0}, \Omega) \\ B &= \mathsf{if}(\mathsf{sample} - \underline{0.5} > 0, \underline{0}, \Omega) \\ I &= \lambda x.x \quad \Omega = (\mathsf{Y}I)\,\underline{0} \end{aligned}$$

The term $A[\mathsf{sample}]$ reduces after a few steps to B, which then randomly does or does not terminate depending on whether the next sample used is greater than 0.5. The first sample, at position $\mathsf{if}_1;\underline{-}_1;@_1;@_1;\mathsf{if}_1$, does not actually affect the outcome, but if terms rather than skeletons were used to label samples, the value r of that sample would be included in the label of the second sample, since that is $(A[\mathsf{sample}] \to A[\underline{r}] \to^* B, \mathsf{if}_1;\underline{-}_1)$. The set of modified traces where $A[\mathsf{sample}]$ terminates would then be

$$\bigcup_{r\in[0,1]} \big\{ s \mid \begin{array}{l} s(A[\mathsf{sample}], \mathsf{if}_1;\underline{-}_1;@_1;@_1;\mathsf{if}_1) = r, \\ \quad s(A[\mathsf{sample}] \to A[\underline{r}] \to^* B, \mathsf{if}_1;\underline{-}_1) > 0.5 \big\}. \end{array}$$

This is a rather unwieldy expression, but the crucial part is that r occurs twice in the conditions on s: once as the value a sample must take, and once in the location of a sample. This means the set is unmeasurable[2], so the termination probability would not even be well-defined. Labelling samples by skeletons instead, this problem does not occur because there are only countably many reachable skeletons, and at each step in a reduction sequence, only finitely many could have occurred yet. Although skeletal reduction sequences omit the information on what the results of sampling were, they still contain all the necessary information on how many, and which, reductions took place.

[2]See [1]. The set has a similar structure to the non-measurable set $\phi^{-1}[1] \subset J^I$ from that article.

For this particular term, $Sk(A[\underline{r}])$ does not depend on the value of r, therefore the set where it terminates becomes simply the following, which is measurable.

$$\left\{s \mid s(Sk(A[\mathsf{sample}]) \to Sk(A[\underline{0}]) \to^* Sk(B), \mathsf{if}_1; \underline{-}_1) > 0.5\right\}$$

It is also not possible to simplify the definition by using reachable skeletons rather than the whole reduction sequence, because if the same skeleton is reached twice, different samples may be needed.

Example 4.2.3. Consider the term $M = \mathsf{Y}(\lambda f x.\, \mathsf{if}(\mathsf{sample} - \underline{0.5} < 0, f\, x, x))\, \underline{0}$, which reduces after a few steps to $N = \mathsf{if}(\mathsf{sample} - \underline{0.5} < 0, M, \underline{0})$. If we label samples by just skeletons and positions, and the pre-selected sample for $(Sk(N), \mathsf{if}_1; \underline{-}_1)$ is less than 0.5, N reduces back to M, then N again, then the same sample is used the next time, therefore it's an infinite loop, whereas if samples are labelled by reduction sequences, the samples for $M \to^* N$ are independent from the samples for $M \to^* N \to M \to^* N$, and so on.

This is still not quite sufficient to attain confluence because sometimes the same samples must be used at corresponding positions in different reduction sequences.

Example 4.2.4. The term $M = \mathsf{sample} + \mathsf{sample}$ has the reachable skeletons $N_1 = \mathsf{X} + \mathsf{sample}, N_2 = \mathsf{sample} + \mathsf{X}, O = \mathsf{X} + \mathsf{X}$ and X, with reductions $M \to N_1 \to O \to \mathsf{X}$ and $M \to N_2 \to O \to \mathsf{X}$. In the reduction $M \to N_1$, the sample labelled $(M, \underline{+}_1)$ is used, and in the reduction $N_2 \to O$, the sample labelled $(M \to N_2, \underline{+}_1)$ is used. Each of these samples becomes the value of the first numeral in O in their respective reduction sequences, therefore in order for confluence to be attained, they must be the same. Which elements of $L_0(M)$ must match can be described by the relation $\sim^*$:

Definition 4.2. Two elements of $L_0(M)$ are related by $\sim_\downarrow$ iff one of the following cases applies:

(i) If N reduces to O with the redex at position α, and β is a position in N disjoint from α, then $(N, \beta) \sim_\downarrow (O, \beta)$.

(ii) If N β-reduces to O at position α, β is a position in $N \mid \alpha; @_1; \lambda$ and $N \mid \alpha; @_1; \lambda; \beta$ is not the variable involved in the reduction, $(N, \alpha; @_1; \lambda; \beta) \sim_\downarrow (O, \alpha; \beta)$

(iii) If N if-reduces to O at position α, with the first resp. second branch being taken, and $\alpha; \mathsf{if}_i; \beta$ occurs in N (where $i = 2$ resp. 3), $(N, \alpha; \mathsf{if}_i; \beta) \sim_\downarrow (O, \alpha; \beta)$

$\sim_\uparrow$ is the reverse of $\sim_\downarrow$, and $\sim_p$ ("p" for parent-child) is the union of $\sim_\downarrow$ and $\sim_\uparrow$.

Definition 4.3. Two elements of $L_0(M)$ are related by $\sim_c$ ("c" for cousin) iff they are both of the form $(M \to^* N \to^* O \to^* O', \delta)$, with the same reduction sub-sequences $M \to^* N$ and $O \to^* O'$, the same position δ, and the reduction sub-sequences $N \to^* O_1, O_2$ given by one of the following cases (in either order, so $\sim_c$ is symmetric).

(a) N contains redexes at disjoint positions α_1 and α_2, O_1 is N reduced first at α_1 then α_2 and O_2 is N reduced first at α_2 then at α_1.

(b) $N \mid \alpha = \mathsf{if}(\underline{r} < 0, N_1, N_2)$, where $r < 0$ (or, respectively, $r \geq 0$), (N_2 resp. N_1) $\mid$ β is a redex, and O_1 is N reduced at α and O_2 is N reduced first at $\alpha; (\mathsf{if}_3$ resp. $\mathsf{if}_2); \beta$ then at α

(c) $N \mid \alpha = \mathsf{if}(\underline{r} < 0, N_1, N_2)$, where $r < 0$ (or, respectively, $r \geq 0$), (N_1 resp. N_2) $\mid$ β is a redex, and O_1 is N reduced first at α then at $\alpha; \beta$ and O_2 is N reduced first at $\alpha; (\mathsf{if}_2$ resp. $\mathsf{if}_3); \beta$ then at α

(d) $N \mid \alpha = (\lambda x.A)B$, there is a redex in A at position β, O_1 is N reduced first at α then at $\alpha; \beta$, and O_2 is N reduced first at $\alpha; @_1; \lambda; \beta$ then at α

(e) $N \mid \alpha = (\lambda x.A)B$, $B \mid \beta$ is a redex, $(\gamma_i)_i$ is a list of all the positions in A where $A \mid \gamma = x$, ordered from left to right, O_1 is N reduced first at $\alpha; @_2; \beta$ then at α, and O_2 is N reduced first at α then at $\alpha; \gamma_i; \beta$ for each i in order.

(f) $N \mid \alpha = \mathsf{Y}(\lambda x.A)$, A reduced at β is A', $(\gamma_i)_i$ is a list of all the positions where $A' \mid \gamma = x$, ordered from left to right, O_1 is N reduced first at $\alpha; \mathsf{Y}; \lambda; \beta$ then at α, and O_2 is N reduced first at α then at $\alpha; \lambda; @_1; \gamma_i; Y; \lambda; \beta$ for each i in order where γ_i is left of β then at $\alpha; \lambda; @_1; \beta$ then at $\alpha; \lambda; @_1; \gamma_i; Y; \lambda; \beta$ for the remaining values of i.

The rules defining $\sim_c$ can be more intuitively understood as diagrams, as shown in Figure 4.1. In the diagrams, a circle represents a term (or skeleton), and the leaves on the trees are subterms at the positions indicated by the labels in the other nodes. The labels on the arrows between trees are the positions of the reductions, and dashed arrows represent some number of reductions in a row. If a reduction sequence includes one branch of one of these cases at some point in it, then the positions in the reduction sequence obtained by substituting in the other branch are all related by $\sim_c$ to the corresponding positions in the original reduction sequence.

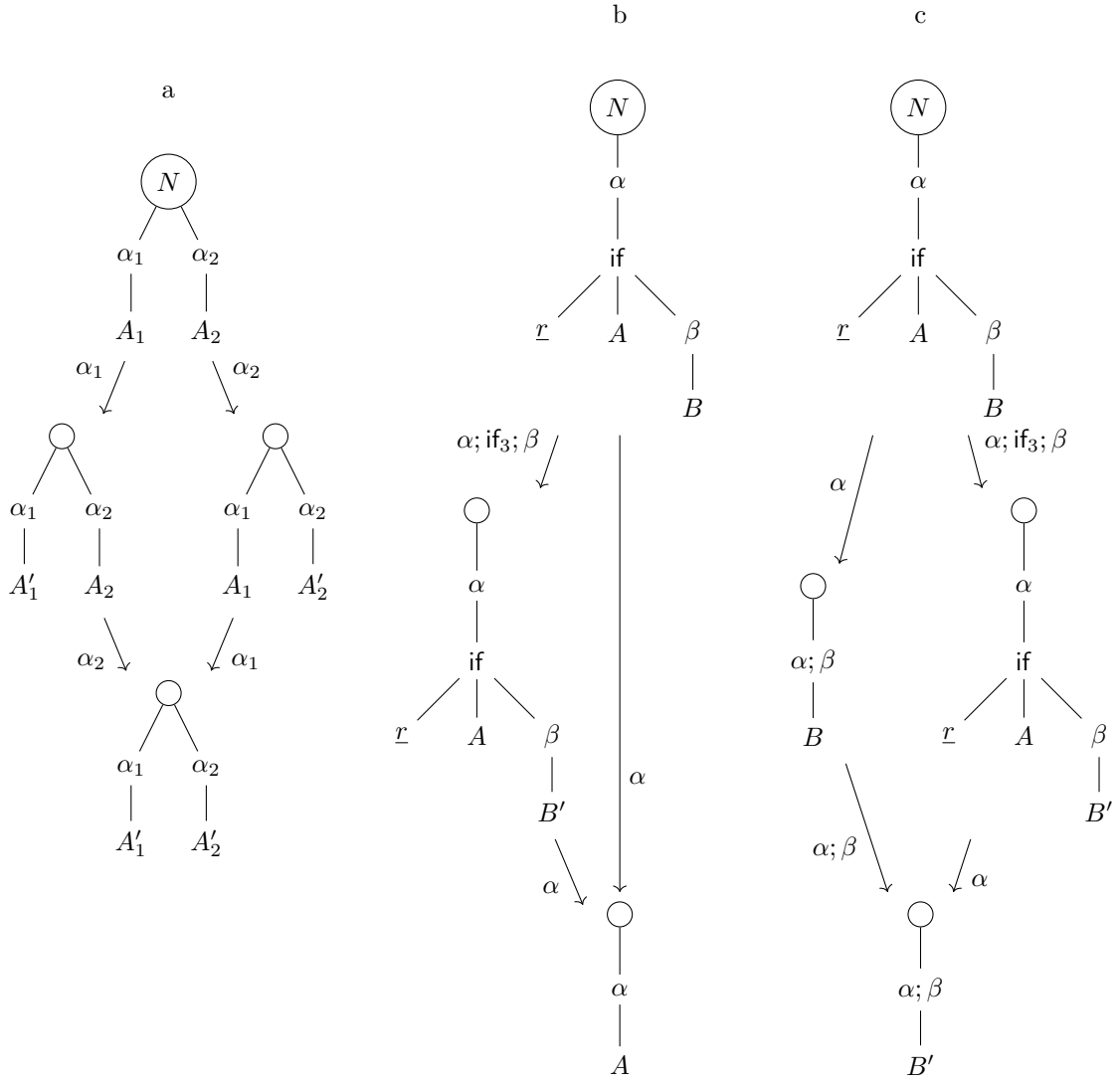
Definition 4.4. The relation $\sim$ on $L_0(M)$ is the union of $\sim_p$ and $\sim_c$.

Example 4.2.5. In Example 4.2.4, $(M, \underline{+}_1) \sim_p (M \to N_2, \underline{+}_1)$ by case i of $\sim_p$ (because the reduction $M \to N_2$ occurs at $\underline{+}_2$, which is disjoint from $\underline{+}_1$), and similarly, $(M, \underline{+}_2) \sim_p (M \to N_1, \underline{+}_2)$.

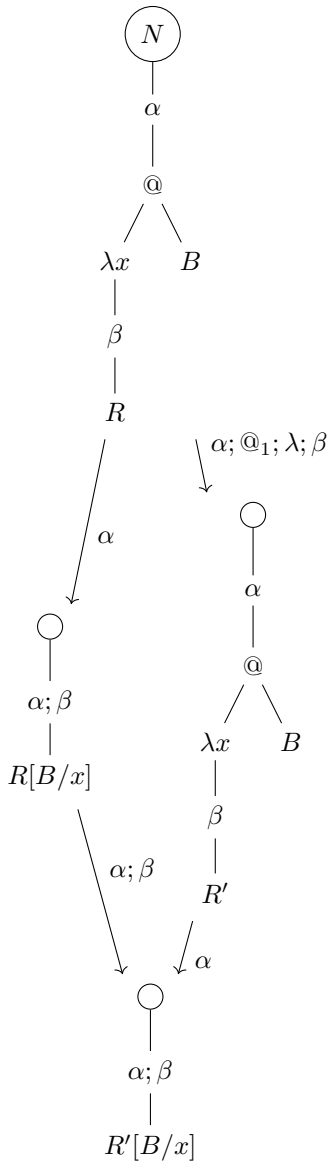
If we extend it to have three $\mathsf{samples}$, $\sim_c$ becomes necessary as well: Let $M_{sss} = \mathsf{sample} + \mathsf{sample} + \mathsf{sample}$ (taking the three-way addition to be a single primitive function), $M_{Xss} = \mathsf{X} + \mathsf{sample} + \mathsf{sample}$, and so on. There are then reduction sequences $M_{sss} \to M_{Xss} \to M_{XXs} \to M_{XXX} \to \mathsf{X}$ and $M_{sss} \to$

$M_{sXs} \to M_{XXs} \to M_{XXX} \to \mathsf{X}$. For the first two reductions, these reduction sequences take the same samples by $\sim_p$, case i, as in Example 4.2.4 . The next reduction uses the samples labelled by $(M_{sss} \to M_{Xss} \to M_{XXs}, \pm_3)$ and $(M_{sss} \to M_{sXs} \to M_{XXs}, \pm_3)$, which are related by $\sim_c$, case a, therefore when these reduction sequences reach M_{XXX}, they still contain all the same numbers, as desired.

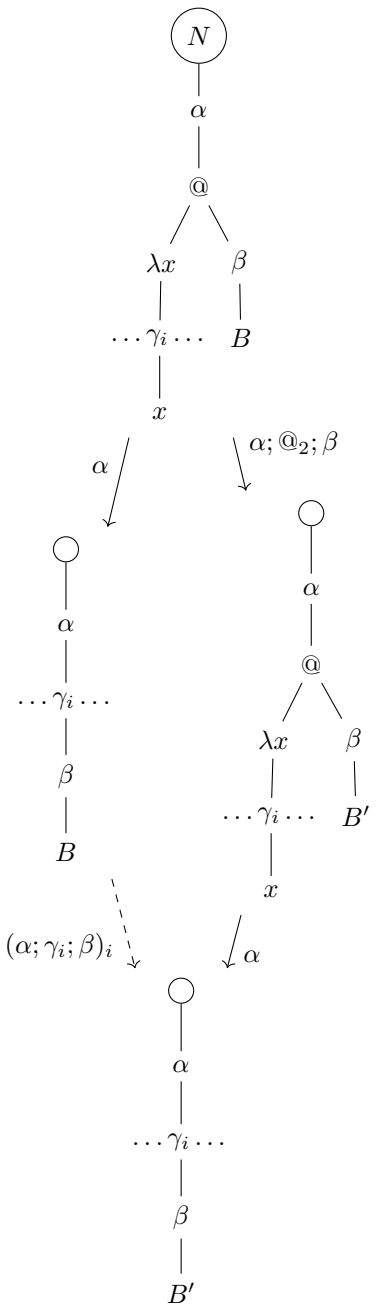
The reflexive transitive closure $\sim^*$ of this relation is used to define the set of *potential positions* $L(M) = L_0(M)/\sim^*$, and each equivalence class can be thought of as containing the various instance of the same position as it may occur across multiple reachable skeletons. If $(N, \alpha) \sim^* (O, \beta)$, then $N \mid \alpha$ and $O \mid \beta$ both have the same shape (i.e. they're either both the placeholder X, both variables, both applications, both samples etc.), therefore it's well-defined to talk of the set of *potential positions where there is a* sample, $L_s(M)$. The new sample space is then defined as $I^{L_s(M)}$, with the Borel σ-algebra and product measure. Since $I^{L_s(M)}$ is a countable product, the measure space is well-defined [2, Cor. 2.7.3].


a
b
c
N
α
if
r
A
β
B
α; if3; β
α; β
A′1
A′2
A1
A2
B′

d


N
α
@
λx
B
β
R
α
α; β
R[B/x]
α; β
α; @1; λ; β
α
@
λx
B
β
R′
α
α; β
R′[B/x]

e


N
α
@
λx
β
· · · γi · · ·
B
x
α
α; @2; β
α
α
· · · γi · · ·
β
B
α
@
λx
β
· · · γi · · ·
B′
x
(α; γi; β)i
α
α
· · · γi · · ·
β
B′

f

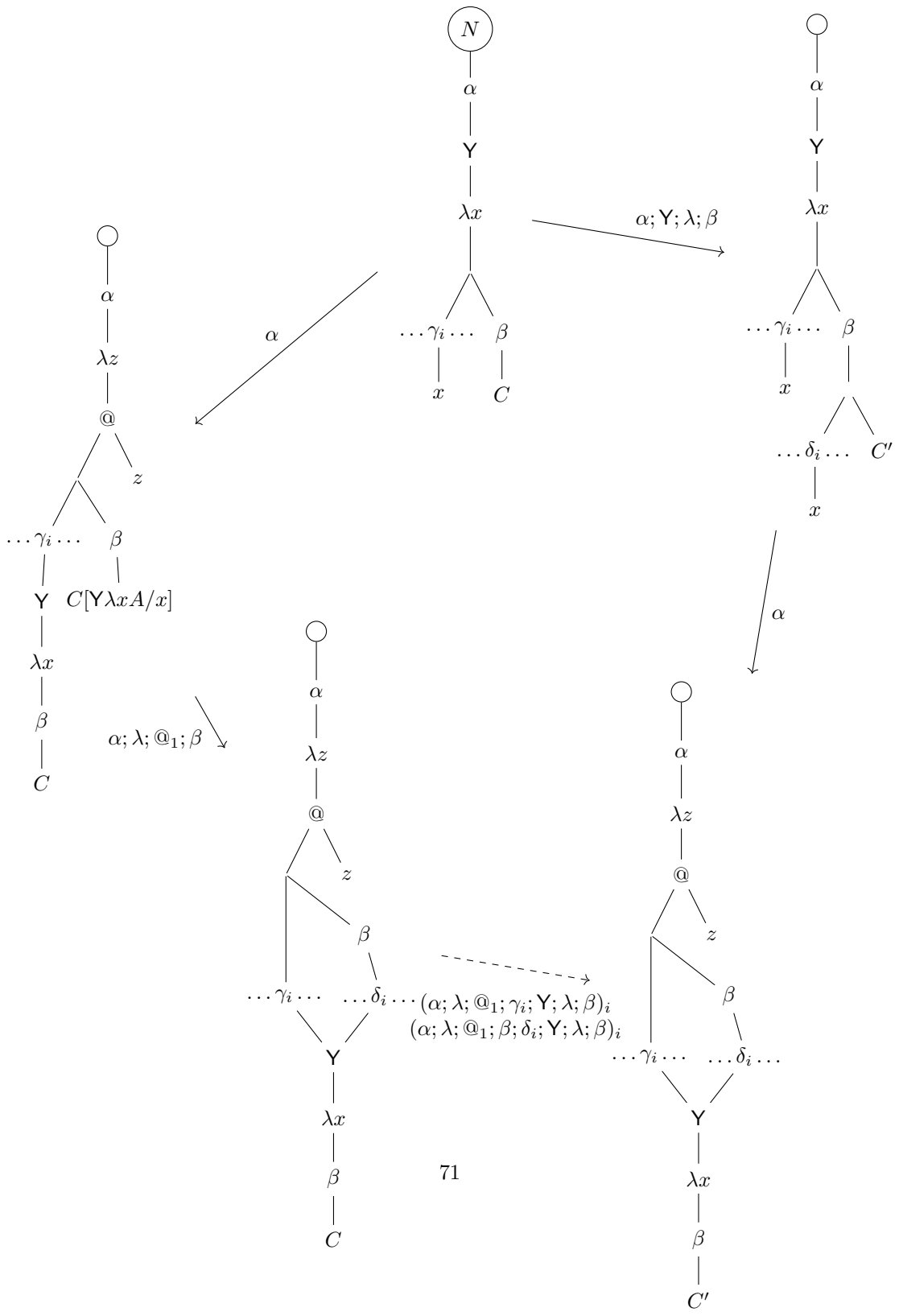


Figure 4.1: Illustration of the $\sim_c$ rules.

Although the six cases in the definition of $\sim_c$ are defined separately, there are some commonalities worth noting. Each case (except a, which is symmetrical) has a right branch and a left branch (the branches of case b are unfortunately displayed the wrong way around in the diagram).The right branch has reduction steps at two positions, β then α, where $\beta > \alpha$ (i.e. β is inside α), and in the left branch, the order of these reductions are swapped, so the reductions are at α then $\beta_1, \ldots, \beta_n$. The position α of one of the reductions is unchanged, and the other reduction may have multiple (in cases e and f) or 0 (in cases b and e) images in the left branch. The images (β_i) are all still inside (greater than or equal to) α, and although they are not generally equal to β, the subskeletons at these positions have a similar shape to the subskeleton (initially) at position β, so that the reductions at these positions are the same type of reduction (e.g. both β-reduction or both if-reduction). Case a is also similar, except that the relevant positions are all disjoint rather than contained in one another, and either branch could be considered the right branch.

If $(X, \alpha) \sim_c (Y, \alpha)$ then the final skeletons in X and Y are equal, and $(X, \beta) \sim_c (Y, \beta)$ for any β that occurs in this skeleton, therefore $\sim_c$ can be considered to apply to reduction sequences even without positions, so $X \sim_c Y$ iff $(X, \cdot) \sim_c (Y, \cdot)$.

4.3 Properties of $\sim^*$ and the Confluent Semantics

Before defining the new version of the reduction relation that corresponds to red in Section 2.4.2, some lemmas are necessary for it to be well-defined.

Lemma 4.1. *If $(A, \alpha) \sim_c (B, \alpha) \sim_\downarrow (C, \beta)$, then $(A, \alpha) \sim_\downarrow (B', \beta) \sim_c (C, \beta)$ for some B', and the reduction $A \to B'$ occurs at the same position as $B \to C$. The similar statement starting from $C \sim_\uparrow B \sim_c A$ is also true.*

Proof. Since $A \sim_c B$, their final skeletons are equal, therefore the reduction $B \to C$ can be appended to A to yield another skeletal reduction sequence B', which has the same final skeleton as C. Since $\sim_\downarrow$ does not depend on the earlier parts of the reduction sequence, $(A, \alpha) \sim_\downarrow (B', \beta)$ for the same reason that $(B, \alpha) \sim_\downarrow (C, \beta)$. The definition of $\sim_c$ splits the reduction sequences A and B into $M \to^* N \to^* O \to^* O'$. Since B' is just A plus $B \to C$ at the end, and C is just B with $B \to C$ at the end, we can extend the reduction sequence $O \to^* O'$ by appending $B \to C$, therefore $(B', \beta) \sim_c (C, \beta)$, so we have the required result. The reverse statement with $\sim_\uparrow$ is equivalent, because $\sim_\uparrow$ is the reverse of $\sim_\downarrow$ and $\sim_c$ is its own reverse. □

The parts of the confluence proof that are not directly related to probabilities and samples are similar in structure to the proof of the Church-Rosser theorem by Tait and Martin-Löf using the notion of parallel reduction [4, 49]. The notion I will use that corresponds to parallel reduction in this context is that of a *parallel reduction sequence*, a (skeletal) reduction sequence in which for

any pair of reductions at positions which aren't disjoint, the reduction at the innermost (greater) position occurs first.

Lemma 4.2. *If $X \sim_\uparrow Y \sim_\downarrow^* Z$, where the reductions $Y \to^* Z$ are in parallel order, then $X \sim_\downarrow^* W \sim_c^* V \sim_\uparrow^* Z$ for some W and V, where the reductions $X \to^* W$ are in parallel order.*

Proof. The sequence $X \sim_\uparrow Y \sim_\downarrow^* Z$ will be rearranged step by step until it is in the desired form, at each point freely using Lemma 4.1 to ensure there are no $\sim_c$s followed immediately by $\sim_\downarrow$s, or $\sim_\uparrow$s by $\sim_c$s. For any individual $\sim_\uparrow$ then $\sim_\downarrow$ pair, let $(A, \alpha) \sim_\uparrow (B, \beta) \sim_\downarrow (C, \gamma)$. The skeleton B reduces to both A and C. If these are the same reduction, then $(A, \alpha) = (C, \gamma)$ and this subsequence may be removed. If they are different, then the way they overlap corresponds to one of the cases in the definition of $\sim_c$, either case a if the reduction positions are disjoint, or one of the cases b–f if one of the reduction positions is inside the other. In any case, there is a reduction sequence from each of A and C to some common skeleton D (O_1 and O_2 in the definition of $\sim_c$). For the same reason that $(B, \beta) \sim_\downarrow (C, \gamma)$, there is some δ such that $(A, \alpha) \sim_\downarrow^* (D, \delta)$, and similarly for the same reason that $(B, \beta) \sim_\downarrow (A, \alpha)$, for the same δ, $(C, \gamma) \sim_\downarrow^* (D, \delta)$. There are a lot of cases to check for this statement, but all of them are rather simple, and similar to each other. There is therefore an alternative $\sim^*$ sequence from (A, α) to (C, γ), of the form $(A, \alpha) \sim_\downarrow^* (D_1, \delta) \sim_c (D_2, \delta) \sim_\uparrow^* (C, \gamma)$, where D_1 and D_2 are the alternative reduction sequences leading to the same skeleton D. There are only multiple $\sim_\downarrow$ steps in the result if the position of the reduction $B \to C$ is inside the position of the reduction $B \to A$, and similarly, there are only multiple $\sim_\uparrow$ steps in the result if the position of the reduction $B \to A$ is inside the position of the reduction $B \to C$. If there are multiple $\sim_\downarrow$ steps, they are not necessarily parallel, but in the only case where they aren't (case f of $\sim_c$), the subsequence of $\sim_\downarrow$ steps with the reductions at $\alpha; \lambda; @_1; \beta$ then $\alpha; \lambda; @_1; \gamma_i; \mathsf{Y}; \lambda; \beta$ for each i where $\gamma_i \geq \beta$ (in the notation of the definition of $\sim_c$) can be replaced by $\sim_\downarrow$s with the reductions at $\alpha; \lambda; @_1; \gamma_i'; \mathsf{Y}; \lambda; \beta$ then $\alpha; \lambda; @_1; \beta$, then another $\sim_c$, where (γ_i') is the list of positions in A (*before* the reduction at β to A', unlike (γ_i)) where $A|\gamma_i' = x$ and $\gamma_i' > \beta$.

The effect of this rearrangement is (ignoring the $\sim_c$s) to swap the order of the $\sim_\uparrow$ and the $\sim_\downarrow$, except that if the positions of one of these reductions is inside the other, that inner $\sim_p$ may be duplicated. If the positions are disjoint, both of them are unchanged (corresponding to case a of $\sim_c$), and if one is inside the other, then the outer position is unchanged and the other resultant positions, although they aren't equal to the original inner position, are still inside the outer position. Taking the sub-sequence of one $\sim_\uparrow$ followed by a parallel sequence of $\sim_\downarrow$s, this rearrangement can be repeatedly applied to move the $\sim_\uparrow$ further along the sequence. If at some point a $\sim_\uparrow$ and $\sim_\downarrow$ match (by relating the same (skeleton, position) pairs in opposite directions), they may be removed and the process may stop early. The $\sim_\uparrow$ may be duplicated if it passes a $\sim_\downarrow$ whose position is outside of the $\sim_\uparrow$'s position, but by the assumption that the sequence of $\sim_\downarrow$s is initially parallel, no position inside of this occurs later in the sequence, therefore all of the resultant $\sim_\uparrow$s pass the remaining $\sim_\downarrow$s without

changing or duplicating them. The $\sim_\uparrow$s may be further duplicated, but the number of steps left in this process is bounded by the product, for all of the remaining $\sim_\downarrow$s, of 2+ the number of occurrences of the variable relevant to the reduction, because for each $\sim_\downarrow$, that is more than the number of duplicates it could produce for any $\sim_\uparrow$ that passes it. It is also possible (earlier) for some of the $\sim_\downarrow$s to be duplicated, but the only case where this process would not terminate, that both the $\sim_\downarrow$s and the $\sim_\uparrow$s continue being duplicated forever so that the number of switches left to do never decreases, is prevented by the parallelness condition, as all of the duplications of $\sim_\downarrow$s occur before all of the duplications of $\sim_\uparrow$s.

It still remains to be shown that the sequence of $\sim_\downarrow$s left at the end of this can be made parallel. The changes that may have occurred since the previous version of this sequence (which was known to be parallel already), are that some of the $\sim_\downarrow$s may have been duplicated by passing the $\sim_\uparrow$, and if that $\sim_\uparrow$ matches one of the $\sim_\downarrow$s, that $\sim_\downarrow$ is removed. The position of the $\sim_\uparrow$'s reduction is the same for all of these duplications, because it is not changed until later in the sequence, when the $\sim_\downarrow$s with positions outside of that may occur. The resultant positions after a duplication are in the same position relative to each other, and therefore the order of the remaining $\sim_\downarrow$s is automatically parallel, except that (letting the position of the $\sim_\uparrow$'s reduction be α) if there are $\sim_\downarrow$s with reductions at positions $\alpha;@_1;\lambda;\beta$ and $\alpha;@_2;\gamma$, or $\alpha;\mathsf{Y};\lambda;\beta$ and $\alpha;\mathsf{Y};\lambda;\gamma$, some of the resultant $\sim_\downarrow$s' positions may be inside each other where they were originally not. Similarly to the case where a Y reduction and a reduction at a position inside it are swapped though, the order can be fixed by swapping some of the $\sim_\downarrow$ steps with each other. In the first case, the reduction at position $\alpha;@_1;\lambda;\beta$ ends up at $\alpha;\beta$, and the reduction at position $\alpha;@_2;\gamma$ ends up at the positions $\alpha;\delta_i;\gamma$, where (δ_i) is the positions of the variable relevant to the $\sim_\uparrow$'s reduction inside its lambda. The positions of the variable in the lambda may be different before the reduction at β, but the positions of the redexes corresponding to the original redex at $\alpha;@_2;\gamma$ are changed similarly. Rather than reducing at $\alpha;\beta$ then at $\alpha;\delta_i;\gamma$, it is therefore possible to reach the same point by reducing at $\alpha;\delta'_i;\gamma$ then at $\alpha;\beta$, where (δ'_i) is the positions of the relevant variable within the lambda before the reduction at $\alpha;@_1;\lambda;\beta$ (and again, a $\sim_c$ of some sort must also be introduced). The other case, that the $\sim_\uparrow$'s reduction is a Y-reduction, is similar. Each reduction which is duplicated has one image at $\alpha;\lambda;@_1$;its original relative position, and one for each occurrence of the variable. Those corresponding to variable occurrences may need to be moved to before those at $\lambda;@_1$. □

Lemma 4.3. *If $(N_1,\alpha_1)\sim^*(N_2,\alpha_2)$, there are some descendants N'_1,N'_2 of N_1 resp. N_2 (so that $N_1\to^* N'_1$ and $N_2\to^* N'_2$), and a position α' in both of them such that $(N_1,\alpha_1)\sim^*_\downarrow(N'_1,\alpha')\sim^*_c(N'_2,\alpha')\sim^*_\uparrow(N_2,\alpha_2)$.*

Proof. As in the previous lemma, at each stage of this proof it will be assumed that the $\sim_c$ steps and the $\sim_p$ steps in the sequence from (N_1,α_1) to (N_2,α_2) are rearranged such that there is never a $\sim_c$ immediately before a $\sim_\downarrow$, and there is

never a $\sim_c$ immediately after a $\sim_\uparrow$, by Lemma 4.1. With these rearrangements assumed, it is sufficient to prove that the $\sim_p$ steps can be rearranged so that all of the $\sim_\downarrow$ steps come before all of the $\sim_\uparrow$ steps (possibly introducing some $\sim_c$ steps in the process).

The rearrangement to put all of the $\sim_\downarrow$s before all of the $\sim_\uparrow$s proceeds by induction on the number of $\sim_\downarrow$s that occur after the first occurrence of $\sim_\uparrow$. If it's 0, all of the $\sim_p$s are already in the correct order, and we're done. Otherwise, take the subsequence from the first $\sim_\uparrow$ to the first $\sim_\downarrow$ after that. This must be rearranged to some $\sim_\downarrow$s followed by some $\sim_\uparrow$s, then once that's done, the number of $\sim_\downarrow$s after the first $\sim_\uparrow$ in the overall sequence will have decreased by 1.

Let this subsequence be $A \sim_\uparrow^n B \sim_\downarrow C$. Suppose for induction that $A \sim_\uparrow^k (D, \delta) \sim_\downarrow^* (E, \epsilon) \sim_c^* (E', \epsilon) \sim_\uparrow^* C$ for some D, δ, E, ϵ, and $0 \leq k \leq n$, where the reduction sequence $D \to^* E$ is parallel. This induction is in reverse with k decreasing from n to 0, so we need to prove that the inductive hypothesis is true for $k-1$, which follows directly from Lemma 4.2 and Lemma 4.1. For the base case where $k = n$, simply set $(D, \delta) = B$ and $(E, \epsilon) = C$. Any single reduction step is trivially in parallel order. If $k = 0$, then A is related to B by a sequence of $\sim_\downarrow$s then $\sim_\uparrow$s, as desired.

After all of these rearrangements are complete, all of the $\sim_\downarrow$s occur before all of the $\sim_\uparrow$s, and all of the $\sim_c$s must be in between them. All of the rearrangements leave the end points of the sequence unchanged, therefore this is an alternative sequence of $\sim$ steps between (N_1, α_1) and (N_2, α_2) of the form $(N_1, \alpha_1) \sim_\downarrow^* (N_1', \alpha') \sim_c^* (N_2', \alpha') \sim_\uparrow^* (N_2, \alpha_2)$. □

The structure of $\sim_c^*$ can be seen more clearly by choosing a canonical example from each equivalence class. For this, the member of the class that's in call-by-value order is used.

Definition 4.5. A reduction sequence is in call-by-value order if for every pair of reductions in the sequence at positions α and β, where the reduction at α happens first but the redex at β already exists when the reduction at α occurs (so cases such as $\alpha = \beta; @_2$ are excluded), either $\alpha \leq \beta$ or α and β are disjoint, with α to the left of β (i.e. the first elements of the sequences α, β where they differ are $@_1$ and $@_2$, $\underline{f}_i$ and $\underline{f}_j$ for $i < j$, or if_i and if_j for $i < j$, in that order).

Note that even if the reductions that occur in the sequence are in call-by-value order, it may not be the actual call-by-value reduction sequence starting from that point because some redexes may remain un-reduced even when other reductions that should happen afterwards occur. It is only required that the reductions that do occur occur in the correct order. This is analogous to standard reduction sequences, except that it is call-by-value rather than call-by-name.

Lemma 4.4. *For any reduction sequence X, there is a unique reduction sequence X_{cbv} that is related to X by $\sim_c^*$ and is in call-by-value order, and if $X \sim_c^* Y$, then $X_{cbv} = Y_{cbv}$.*

Proof. Define $(X_n)_{n\geq 0}$ recursively so that $X_0 = X$ and if X_n is not in CBV order, take the last pair of reductions in X_n which aren't in CBV order. They are adjacent in the sequence, as the CBV order is a total order. Let the positions of these reductions be α and β respecively. If they are disjoint, they can be considered as one of the branches of $\sim_c$ case a. Otherwise, $\beta < \alpha$, and the subskeleton at position β at the appropriate point in the reduction sequence is a redex with non-trivial subterms, therefore it's of one of the forms $\mathsf{if}(\mathsf{X} > 0, A, B)$, $(\lambda x.A)B$ or $\mathsf{Y}\lambda x.A$, therefore the reductions α and β form the O_2 branch of one of the cases b, c, d, e or f of $\sim_c$. In either case, define X_{n+1} as equal to X_n except taking the other (left) branch, so that the order of the reductions at α and β are switched (although the equivalent(s) of the reduction at α may not actually be at that position).

This sequence of swaps rearranges the reductions in X rather like insertion-sort (but in some cases duplicating or deleting the new element), and eventually reaches some X_n that is in CBV order. This is defined to be X_{cbv}.

If $Y \sim_c X \sim_c^* X_{cbv}$, and the reductions involved in $Y \sim_c X$ (in the sense of being the $N \to^* O_1$ or $N \to^* O_2$ in this instance of $\sim_c$) are not the last pair of reductions in Y that aren't in CBV order, it will be shown that the order of the $\sim_c$s can be rearranged so that the sequence $Y \sim_c^* X_{cbv}$ is the canonical sequence given above, therefore $Y_{cbv} = X_{cbv}$. To construct this rearrangement, consider the sequence of $\sim_c$ steps from Y to X_{cbv}. The first step takes some subsequence of the reduction sequence Y and swaps the order of those reductions. If $X = X_{cbv}$, then $X = Y_1 = Y_{cbv}$ already. Otherwise, consider the cases according to whether the image of this subsequence in X occurs earlier than the next pair of reductions to be swapped (i.e. the last pair of reductions in X that occur not in CBV order). If not, then either the step $Y \sim_c X$ changes this subsequence into CBV order, or out of it. If it is into, then it was the last out-of-order pair in Y therefore the whole sequence $Y \sim_c^* X_{cbv}$ is already in the canonical order and $Y_{cbv} = X_{cbv}$. If it switches the pair to the wrong order, then the result of that switch is the last out-of-order pair in X therefore $Y = X_1$, and the sequence $X_1 \sim_c^* X_{cbv}$ is just a suffix of the sequence $X \sim_c^* X_{cbv}$ therefore $(X_1)_{cbv} = X_{cbv}$ and again $Y_{cbv} = X_{cbv}$.

The remaining cases are when the subsequence of reduction steps involved in $Y \sim_c X$ occur before the last out-of-order pair in X, but possibly overlapping. If there is no overlap, then the $\sim_c$ steps do not interfere with each other at all, and can simply be performed in the other order. This gives a different sequence of $\sim_c$ steps $Y \sim_c Y_1 \sim_c X_1 \sim_c^* X_{cbv}$. As $Y_{cbv} = (Y_1)_{cbv}$, we can proceed by induction in this case and use the fact that $(Y_1)_{cbv} = (X_1)_{cbv} = X_{cbv}$ to acheive the result.

In the other case, there is some overlap between the regions involved in $Y \sim_c X$ and $X \sim_c X_1$, but the last reduction in $X \sim_c X_1$ is later than the last reduction in $Y \sim_c X$. There are only 2 reductions in X involved in $X \sim_c X_1$, therefore the region involved in $Y \sim_c X$ ends on the first of the reductions involved in $X \sim_c X_1$. Now consider the specific sequence of positions of the

reductions in X involved in either of these steps. Let the last of these positions be α, the first γ, and the second-last, third-last and so on $\beta_1, \beta_2, \ldots$ respectively, and let the positions of the redexes in Y involved in $Y \sim_c X$ be β then (γ_i) in order. Then $X \sim_c X_1$ swaps β_1 and α, and α comes before β_1 in CBV order (i.e. $\alpha < \beta_1$ or α is left of β_1), and the step $Y \sim_c X$ either proceeds forwards towards CBV order, swapping β below $\gamma_1 = \gamma$ to form γ then (β_i), or proceeds backwards, taking Y even farther from CBV order, swapping the (γ_i) to above β, forming γ then $\beta_1 = \beta$. In each of the cases below, it is required to show that the canonical sequence $(X_i)_i$ eventually reaches some term which is the same as one in $(Y_j)_j$, and from that point on, they match, therefore they reach the same end result: $X_{cbv} = Y_{cbv}$.

- $Y \sim_c X$ in the backwards direction (so that Y is closer to CBV order than X is): By swapping the roles of X and Y, and β and γ, this is equivalent to the $Y \sim_c X$ forwards case. The assumption that α is before β_1 in CBV order (so that β_1, α is the pair to be swapped in $X \sim_c X_1$) maps to the assumption that α is before γ_1 (so that the result doesn't just follow trivially by Y_1 being equal to X) and vice-versa. Then one of the cases below establishes that $X_i = Y_j$ for some i and j, therefore after swapping back, $Y_i = X_j$.

- $Y \sim_c X$ forwards and α comes after γ in CBV order: In this case, the γ and α reduction steps are already in the correct order in Y, therefore β and γ are already the last out-of-order pair of reductions in Y, $Y_1 = X$, and $Y \sim_c X \sim_c^* X_{cbv}$ is already the canonical sequence from Y therefore $Y_{cbv} = X_{cbv}$.

- $Y \sim_c X$ forwards and α is disjoint from γ and β: In this case, the sequence $(X_i)_i$ starts by swapping α and β_1 by case a, then because γ is disjoint from α and comes after is in CBV order, γ is right of α, therefore either all the remaining β_is, are $\geq \gamma$ (if β and γ aren't disjoint) or there are no remaining β_is (if β and γ are disjoint, in which case they swap by case a and there is only β_1). The sequence $(X_i)_i$ therefore proceeds to swap all the remaining β_is then γ, in order, below α, resulting in X_i for some i having as its relevant sub-reduction-sequence $\alpha, \gamma, \beta_k, \ldots, \beta_1$. The other sequence $(Y_j)_j$ starts by swapping γ then β below α, then swapping γ and β. Because α is disjoint from γ, the reduction there does not affect the sub-skeleton at position γ, therefore this last $\sim_c$ step proceeds identically to how it did in $Y \sim_c X$, resulting in $\gamma, \beta_k, \ldots, \beta_i$. Overall, this results in Y_3 having the relevant sub-reduction-sequence $\alpha, \gamma, \beta_k, \ldots, \beta_1$, therefore $Y_3 = X_i$ for the aforementioned value of i.

- $Y \sim_c X$ forwards, $\alpha \leq \gamma$ and α (and therefore also γ) is disjoint from β: In this case, $(X_i)_i$ proceeds by swapping $\beta_1 = \beta$ below α by case a, then swapping γ below α by one of the other cases, and $(Y_j)_j$ proceeds by swapping γ and α, then swapping β below α then all of the images of γ (which are $\geq \alpha$ and therefore left of β). In both cases, the subterm at α

when α and γ are swapped is unaffected by the reduction at β, therefore it produces the same result in both cases, therefore the overall sequence in both cases is α then the images of γ then β.

- Both γ and β are $> \alpha$ but they're disjoint from each other, and either the skeleton at position α is of the form $\mathsf{if}(\mathsf{X} > 0, A, B)$ or it is of the form AB and either both γ and β are $> \alpha; @_1$ or they are both $> \alpha; @_2$: In this case, $(X_i)_i$ proceeds by swapping β then γ below α, in each case possibly forming 0 or multiple images. The only way the number of images of each of these positions can be different is if the subskeleton at α is an if, and there are 0 of one of them and 1 of the other, in which case they're trivially in the correct order already. Otherwise, all of the images of β and γ are disjoint from one another, therefore none of the positions, or their relative order, change when they are swapped, therefore the next part of the sequence $(X_i)_i$ is just insertion-sort running on the images of β and γ with $\sim_c$ case a swaps until they're in the correct order. The sequence $(Y_j)_j$ starts by swapping γ then β below α, then as before, sorting the images into the correct order by case a swaps. The images originally formed of γ and β are the same in both cases, therefore the final order is the same, so $X_i = Y_j$ for some i and j.
- $Y \sim_c X$ forwards, $\alpha; @_1 < \gamma$, and $\alpha; @_2 < \beta$: Let x be the variable involved in the β-reduction at α, so that the sub-skeleton at α is $(\lambda x.A)B$ for some A, B, where $\gamma = \alpha; @_1; \lambda; \gamma'$ and $A \to A'$ at γ', and $\beta = \alpha; @_2; \beta'$, and $B \to B'$ at β'. In this case, $(X_i)_i$ starts by swapping β below α by case e, producing one image of β for each x in A', then swapping γ below α by case d, producing a single image $\alpha; \gamma'$. For each of the instances of x in A left of γ', the reduction at $\alpha; \gamma'$ is then swapped below the corresponding image of β. The sequence $(Y_j)_j$ starts by swapping γ below α then β below α, but in this case the images of β produced may be different. In this case the swap happens earlier in the reduction sequence than γ, so that the body of the lambda is still A rather than A', and the reduction at γ may rearrange or change the number of instances of x. Next, each of the images of β to the right of $\alpha; \gamma'$ are swapped with it by case a. At this point, the next few reductions before $\alpha; \gamma'$ will in general be images of β at positions inside $\alpha; \gamma'$. The subskeleton at position $\alpha; \gamma'$ before these reductions is $(A|\gamma')[B/x]$, then by the images of β this reduces to $(A|\gamma')[B'/x]$, then it reduces at its root position ($\alpha; \gamma'$ in the overall skeleton) to $(A'|\gamma')[B'/x]$. For each position of an x in $(A|\gamma')$, there are 0 or more corresponding positions of x in $(A'|\gamma')$, depending on what type of reduction $A \to A'$ is. Because B and B' cannot contain any instances of the variable involved in the reduction $A \to A'$ (if it's a β-reduction or Y-reduction), each instance of B in $(A|\gamma')[B/x]$ has the same set of images in $(A'|\gamma')[B/x]$ as the corresponding x in $(A|\gamma')$. When these images of β that overlap with the image of γ are swapped below it, each of them has its own images, one at each position of B in $(A'|\gamma')[B/x]$ corresponding to its original copy of B in $(A|\gamma')[B/x]$. After all of them are swapped below $\alpha; \gamma'$ (and rearranged

among themselved by case a), there is therefore one reduction at each copy of B in $(A'|\gamma')[B/x]$, i.e. one at position β' relative to each x in $(A'|\gamma')$. They are therefore the same as the images of β in X_1 that overlap with $\alpha;\gamma'$, because those are also one reduction at position β' relative to each x in $(A'|\gamma)$. Both $(X_i)_i$ and $(Y_j)_j$ therefore eventually reach a point where the relevant sub-reduction-sequence is α, the images of β to the left of $\alpha;\gamma'$, $\alpha;\gamma'$ itself, the aforementioned images of β that overlap with $\alpha;\gamma'$, then all the images of β to the right of $\alpha;\gamma'$.

- $Y \sim_c X$ forwards, γ and β are disjoint and both $> \alpha$, and the reduction at α is a Y-reduction: This is similar to the previous case, but more complicated, so it will need some definitions to explain properly what's going on. Let the term at α before any of the reductions be $\mathsf{Y}\lambda x.A$, let $\gamma = \alpha;\mathsf{Y};\lambda;\gamma'$, let $\beta = \alpha;\mathsf{Y};\lambda;\beta'$, let $A|\beta' = B \to B'$ with the reduction at $\cdot$, let $A|\gamma' = C \to C'$ with the reduction at $\cdot$, let the positions in A where x occurs that are left of γ', between γ' and β' (but still disjoint from both), and right of β' respectively be (δ_i^l), (δ_i^m) and (δ_i^r), let the positions in B, C, B' and C' respectively where x occurs be (δ_i^B), (δ_i^C), $(\delta_i^{B'})$ and $(\delta_i^{C'})$ (so that all the positions in A where x occurs, in left-to-right order, are $(\delta_i^l)_i, (\delta_i^C)_i, (\delta_i^m)_i, (\delta_i^B)_i, (\delta_i^r)_i$). The sequence $(X_i)_i$ starts by swapping β then γ below α. With the shorthand that $\gamma'(\epsilon) = \alpha;\lambda;@_1;\epsilon;\mathsf{Y};\lambda;\gamma'$, $\gamma'_0 = \alpha;\lambda;@_1;\gamma'$, and similarly for β', the relevant portion of X_2 is then α, $(\gamma'(\delta_i^l))_i$, γ'_0, $(\gamma'(\delta_i^{C'}))_i$, $(\gamma'(\delta_i^m))_i$, $(\gamma'(\delta_i^B))_i$, $(\gamma'(\delta_i^r))_i$, $(\beta'(\delta_i^l))_i$, $(\beta'(\delta_i^{C'}))_i$, $(\beta'(\delta_i^m))_i$, β'_0, $(\beta'(\delta_i^{B'}))_i$, $(\beta'(\delta_i^r))_i$. Next in $(X_k)_k$, for each i, $\gamma'(\delta_i^r)$ are swapped past all the images of β until it is immediately before $\beta'(\delta_i^r)$ by case a, then for each i, $\gamma'(\delta_i^B)$ is swapped past all the images of β until β'_0 by case a, then swapped with β'_0 by one of the other cases depending on what type of reduction $B \to B'$ is and the relative positions. Even if it's case d or f, all of the images of $\gamma'(\delta_i^B)$ are disjoint from each other (and from the positions of the other reductions after β'_0) because the redex, C, doesn't contain any instances of the variable involved in the reduction $B \to B'$. Expanding the definitions, this is swapping $\alpha;\lambda;@_1;\delta_i^B;\mathsf{Y};\lambda;\gamma'$ with $\alpha;\lambda;@_1;\beta'$ at a point in the reduction sequence where the subskeleton at $\alpha;\lambda;@_1;\beta'$ is B with a mixture of $\mathsf{Y}\lambda x.A$ and $\mathsf{Y}\lambda x.A[C'/\gamma']$ substituted for its occurrences of x, and one of these occurrences of x is at δ_i^B. For each δ_j^B, there are some corresponding $\delta_k^{B'}$, where the images of the x at δ_j^B end up after the reduction $B \to B'$. These cover all the $\delta_k^{B'}$, and uniquely, and for each δ_i^B, the images of $\zeta;\delta_i^B;\theta$ after swapping it with $\zeta;\beta'$ are precisely $(\zeta;\delta_k^{B'};\theta)$ for those same values of k, therefore after swapping all of the $(\gamma'(\delta_i^B))_i$ down past β'_0, their images are $(\gamma'(\delta_i^{B'}))_i$. After swapping with β'_0, each of these images than swaps by case a to take its place among the $(\beta'(\delta_i^{B'}))_i$. Next up in $(X_i)_i$, the $(\gamma'(\delta_i^m))_i$s and the $(\gamma'(\delta_i^{C'}))_i$s swap down past some images of β they're disjoint from to take their place before their matching image of β, then γ'_0 swaps past the $(\beta'(\delta_i^l))_i$, stopping immediately before $\beta'(\delta_1^{C'})$, then the $(\gamma'(\delta_i^l))_i$ and the $(\beta'(\delta_i^l))_i$ mix. The end re-

sult of the rearrangement of (this subsequence of) the reduction sequence is therefore α, $(\gamma'(\delta_i^l), \beta'(\delta_i^l))_i$, γ'_0, $(\gamma'(\delta_i^{C'}), \beta'(\delta_i^{C'}))_i$, $(\gamma'(\delta_i^m), \beta'(\delta_i^m))_i$, β'_0, $(\gamma'(\delta_i^{B'}), \beta'(\delta_i^{B'}))_i$, $(\gamma'(\delta_i^r), \beta'(\delta_i^r))_i$.

The other sequence, $(Y_j)_j$, proceeds similarly, but with the images of γ starting out after the images of β. It swaps all the images of β that are right of γ_0 to the appropriate places by case a, then swaps all the $(\beta'(\delta_i^C)_i$ first to then past γ'_0, resulting in $(\beta'(\delta_i^{C'}))_i$ for the same reason as with swapping those images of γ that overlapped with β'_0 past it in the $(X_i)_i$ case above, then finally swapping the $(\beta'(\delta_i^l))_i$s and the $(\gamma'(\delta_i^l))_i$s into the correct order. As required, this produces the same result as in $(X_i)_i$.

- $Y \to X$ forwards, and $\alpha < \gamma < \beta$: Let γ' be such that $\gamma = \alpha; \mathsf{if}_i; \gamma'$, $\alpha; @_1; \lambda; \gamma'$, $\alpha; @_2; \gamma'$ or $\alpha; \mathsf{Y}; \lambda; \gamma'$, so that γ' is the freely varying later part of γ as in the definition of $\sim_c$, whichever case applies to swapping γ and α. Similarly, let β' be the freely varying part of β relative to γ. Let the images of γ after swapping it with α (where this swap occurs later in the reduction sequence than β) then be $(\alpha; \delta_i; \gamma')_i$, and the images of β after swapping it with γ be $(\gamma; \epsilon_i; \beta')_i$. Depending on the skeleton at α, and γ's position within it, $(\delta_i)_i$ may be empty (case b), a singleton containing $\cdot$ (cases c and d), all the positions of the relevant variable in the lambda in the skeleton at α (case e), or $\lambda; @_1$ and $\lambda; @_1; \zeta; \mathsf{Y}; \lambda$ for each position ζ of the relevant variable (case f), but in all cases (except a, which is excluded because none of α, γ and β are disjoint) the set of images has this general structure. Furthermore, the positions $(\delta_i)_i$ are determined only by the general position of γ within α (the part excluded from γ'), and the positions of the relevant variable within the skeleton at α. They do not depend on α or γ, so that if both initial positions were moved somewhere else the relative positions of the images of γ would be unaffected, and similarly if some position within γ were swapped with α, the relative positions of its images would be the same (except to the extent that, in case f, the positions of the relevant variable are taken after the inner reduction takes place, so that some of those may still change). Because there are so many of them and they don't actually affect the multiset of positions where reductions take place, we will be ignoring swaps by case a in the proof of this case, and just assuming that all the final positions end up in the correct order.

  The final set of reduction positions that we will be proving both $(X_i)_i$ and $(Y_j)_j$ reach is $\alpha, (\alpha; \delta_i; \gamma', (\alpha; \delta_i; \gamma'; \epsilon_j; \beta')_j)_i$, where as usual the sequences are expanded out to the full list. For the sequence $(X_i)_i$, first the images of β in X_0 are $(\gamma, \epsilon_i, \beta')$. For each of these in turn, it is swapped with α. By the fact mentioned above that sub-positions are mapped to the same set of images except in case f, the images of $\gamma, \epsilon_j, \beta'$ after this swap are $(\alpha; \delta_i^j; \gamma', \epsilon_j, \beta')_i$, where $(\delta_i^j)_i$ is the same as $(\delta_i)_i$ except that in the case where this is not the first image of β swapped below α and the reduction at α is a Y-reduction (and therefore swaps with it proceed by case f), the

positions of the relevant variable are taken after the reductions at γ and $\gamma;\epsilon_k;\beta'$ for $k \leq j$ (this implies that the first of these, $\delta_i^n = \delta_i$). All the images $\alpha;\delta_i^j;\gamma';\epsilon_j;\beta'$ where $\delta_i^j > \lambda;@_1;\gamma';\epsilon_{j+1};\beta'$ are then swapped with $\alpha;\lambda;@_1;\gamma';\epsilon_{j+1};\beta'$ (which is $\alpha;\delta_i^{j+1};\gamma';\epsilon_{j+1};\beta'$ for some i). As in the case where $\alpha;\mathsf{Y} < \gamma,\beta$ and γ and β are disjoint, the effects of these swaps may duplicate or delete the reductions in such a way that all the remaining images of $\gamma;\epsilon_j;\beta'$ are $(\alpha;\delta_i^{j+1};\gamma';\epsilon_j;\beta'$. This is repeated with δ_i^{j+2} and so on until these images are all in the correct order, at which point they are $(\alpha;\delta_i;\gamma';\epsilon_j;\beta')_i$. After all of these are done, γ is swapped with α, yielding $(\alpha;\delta_i^{-1};\gamma')_i$. As with the images of β, the subset of these which are inside $\alpha;\lambda;@_1;\gamma';\epsilon_j;\beta'$ for each j in turn are swapped with it, until they are all in the correct order and the images left of γ are $(\alpha;\delta_i;\gamma')_i$. At this point, X_i has the desired value.

The sequence $(Y_j)_j$ proceeds similarly. First γ swaps with α, yielding α and $(\alpha;\delta_i;\gamma')_i$, then β swaps with α, yielding one image of β for each δ_i, except that, in case f, those images at positions inside $\alpha;\lambda;@_1;\gamma'$ may differ because they are earlier in the reduction sequence than any of the images of γ. Each of the images of β in turn is moved to the corresponding image of γ, except that those overlapping with $\alpha;\lambda;@_1;\gamma'$ may be duplicated or deleted on the way. The images of β after just this process (which does not actually correspond to any term in $(Y_j)_j$, but it's sufficiently independent that the order doesn't matter that much) are $(\alpha;\delta_i;\gamma';\beta'')$ (where $\gamma;\beta'' = \beta$). The images of these for each i, after swapping with $\alpha;\delta_i;\gamma'$, is $(\alpha;\delta_i;\gamma';\epsilon_j;\beta')_j$, because the reductions produced by a swap are unaffected by its overall location, and the skeleton at $\alpha;\delta_i;\gamma'$ at the relevant point in the sequence is equal to the skeleton at γ initially, therefore this is equivalent to immediately swapping β and γ as in $Y \sim_c X$ (except that in the case where the reduction at α is a Y-reduction, the skeleton at $\alpha;\lambda;@_1;\gamma'$ is not actually equal to the skeleton initially at γ because something was substituted in for the variable bound at $\alpha;\mathsf{Y}$, but this doesn't affect the variable involved in the reduction at γ, so this doesn't actually matter). Therefore also in this case, eventually the set of reductions in the relevant portion of Y_j is $\alpha,(\alpha;\delta_i;\gamma',(\alpha;\delta_i;\gamma';\epsilon_j;\beta')_j)_i$, therefore it is equal to some X_i.

In summary, if the swap in $Y \sim_c X$ doesn't overlap with the swap $X \sim_c X_0$, the sequence of swaps can be rearranged until it does, and in any other case, there is some Y_j that's equal to some X_i, therefore these sequences reach the same end point and if $Y \sim_c X$, then $Y_{cbv} = X_{cbv}$, and by chaining these together, if $Y \sim_c^* X$, $Y_{cbv} = X_{cbv}$.

If Y is in CBV order and $Y \sim_c^* X$, then $Y_{cbv} = X_{cbv}$ but also $Y_{cbv} = Y$ because the sequence $(Y_i)_i$ terminates immediately therefore X_{cbv} is the unique reduction sequence in call-by-value order that's related to X by $\sim_c^*$. □

Lemma 4.5. *The relation $\sim$ is defined on $L_0(M)$ with reference to a particular starting term M, so different versions, $\sim_M$ and $\sim_N$, can be defined starting at different terms. If $M \to N$, then $\sim_N^*$ is equal to the restriction of $\sim_M^*$ to $L_0(N)$.*

Proof. $\sim_N^*$ is trivially a subset of $\sim_M^*$ because $\sim_N$ is a subset of $\sim_M$.

In the other direction, suppose $(N_1, \alpha_1) \sim_M^* (N_2, \alpha_2)$ where both N_1 and N_2 are descendants of N. By Lemma 4.3, take $(N_1, \alpha_1) \sim_{p,M}^* (N_1', \alpha') \sim_{c,M}^* (N_2', \alpha') \sim_{p,M}^* (N_2, \alpha_2)$. The $\sim_p^*$ steps remain within $Rch(N)$, and $\sim_p$ does not depend on the history of the reduction sequences, therefore $(N_1, \alpha_1) \sim_{p,N}^* (N_1', \alpha') \sim_{c,M}^* (N_2', \alpha') \sim_{p,N}^* (N_2, \alpha_2)$.

Let X be the call-by-value reduction sequence related to both N_1' and N_2' by $\sim_{c,M}$ given by Lemma 4.4. As $M \to N$ is the first reduction in the sequence N_1', it is the last to be affected by the $\sim_{c,M}$ sequence $N_1' \sim_{c,M}^* X$ given by Lemma 4.4 therefore it can be split into $N_1' \sim_{c,M}^* Y_1 \sim_{c,M}^* X$ where Y_1 is in CBV order except possibly for its first reduction, which is still $M \to N$. Let the position of the reduction $M \to N$ be β. The rearrangement of the reduction sequence $Y_1 \sim_{c,M} X$ consists of moving the reduction at β down past the other reductions in Y_1, possibly duplicating or deleting it in the process, but not affecting the positions or the correct order for any of the other reductions. The reductions derived from $M \to N$ can be identified as follows:

A position in some (reduction sequence of) term(s) is *derived from* another if it is related by the reflexive transitive closure of $\rightsquigarrow$, where $(V, \gamma) \rightsquigarrow (W, \delta)$ just if $V \to W$ and one of the following cases holds:

- $(V, \gamma) \sim_p (W, \delta)$
- $\gamma = \delta$, $V \to W$ at ϵ, and $\epsilon > \gamma$
- $V \mid \epsilon = (\lambda x.U)Z, U|\zeta = x, \gamma = \epsilon; @_2; \theta$ and $\delta = \epsilon; \zeta; \theta$
- $V \mid \epsilon = \mathsf{Y}(\lambda x.U), \gamma = \epsilon; \mathsf{Y}; \lambda; \zeta$ and $\delta = \epsilon; \lambda; @_1; \zeta$
- $V \mid \epsilon = \mathsf{Y}(\lambda x.U), U|\zeta = x, \gamma = \epsilon; \theta$ and $\delta = \epsilon; \lambda; @_1; \zeta; \theta$

Crucially, the reductions in X can be partitioned into 2 sets: those at positions derived from (M, β), and those at positions equal to the reductions in Y_1 (in the same order as they occur in Y_1). Using the same construction for N_2' shows that the positions of the reductions in Y_2 are also the positions of the reductions in X other than those derived from (M, β), therefore $Y_1 = Y_2$ therefore $N_1' \sim_{c,M} Y_1 \sim_{c,M} N_2'$ and every reduction sequence in this sequence starts with $M \to N$ therefore $N_1' \sim_{c,N} N_2'$.

Combining this with the $\sim_{p,N}^*$s at the beginning and end then yields the desired result that $(N_1, \alpha_1) \sim_N^* (N_2, \alpha_2)$, therefore the restriction of $\sim_M^*$ to $\sim_N$'s domain ($Rch(N)$) is a subset of $\sim_N^*$ therefore the two versions of $\sim$ match on this domain, as desired. □

At each reduction step $M \to N$, the sample space must be restricted from $I^{L_s(M)}$ to $I^{L_s(N)}$. The injection $L_0(N) \to L_0(M)$ is trivial to define by appending $Sk(M) \to Sk(N)$ to each path, and using Lemma 4.5, this induces a

corresponding injection on the quotient, $L(N) \to L(M)$. The corresponding map $L_s(N) \to L_s(M)$ is then denoted $i(M \to N)$.

Definition 4.6 ($\Rightarrow$ reduction)**.** Unlike in the purely call-by-value case, the version of the reduction relation that takes into account samples is still a general relation rather than a function, so it is denoted "$\Rightarrow$" instead of "red", and it is a relation on $\biguplus_{M \in \Lambda_0} I^{L_s(M)}$. We write an element of $\biguplus_{M \in \Lambda_0} I^{L_s(M)}$ as (M', s) where the *term* $M' \in \Lambda^0$ and $s \in I^{L_s(M')}$.

$$
\begin{aligned}
(M, s) \Rightarrow (N, s \circ i(M \to N)) \text{ if } & M \to N \text{ at } \alpha \text{ and either} \\
& \text{the redex is not } \mathsf{sample}, \text{ or} \\
& M \mid \alpha = \mathsf{sample} \text{ and } N = M[\underline{s(Sk(M), \alpha)}/\alpha]
\end{aligned}
$$

This reduction relation now has all of the properties required of it. In particular, it can be considered an extension of the standard trace semantics (as will be seen later in Theorem 4.2), and also:

Theorem 4.1. *The relation $\Rightarrow$ is confluent.*

Proof. Suppose that $(M, s) \Rightarrow^* (M_1, s_1)$ and also $(M, s) \Rightarrow^* (M_2, s_2)$, then it is required to prove that there is some (M', s') such that both $(M_1, s_1) \Rightarrow^* (M', s')$ and $(M_2, s_2) \Rightarrow^* (M', s')$. First consider the special case where $(M, s) \Rightarrow (M_1, s_1)$ and $(M, s) \Rightarrow (M_2, s_2)$, with only a single step in each case. Let the positions of the redexes in $M \to M_1$ and $M \to M_2$ be α_1 and α_2 respectively.

First consider the case that α_1 and α_2 are disjoint. Let $M_1 = M[X_1/\alpha_1]$ and similarly for X_2, then let $M' = M[X_1/\alpha_1][X_2/\alpha_2]$. As the positions are disjoint, the substitutions commute and both M_1 and M_2 reduce (with $\to$) to M'. Let $s' = s \circ i(M \to M_1) \circ i(M_1 \to M')$. The injection $i(M \to M_1) \circ i(M_1 \to M')$ consists of prepending $M \to M_1 \to M'$ to each reduction sequence, but by case a of $\sim_c$, this is equivalent to prepending $M \to M_2 \to M'$, which is $i(M \to M_2) \circ i(M_2 \to M')$. In the case that the reduction $M \to M_2$ isn't a sample-reduction, this is enough to establish that $(M_1, s_1) \Rightarrow (M', s')$ (and similarly if the redex of $M \to M_1$ isn't sample, $(M_2, s_2) \Rightarrow (M', s')$). If it is sample though, in order for it to be the case that $(M_1, s_1) \Rightarrow (M', s')$, it is additionally necessary that X_2, the result of the reduction at α_2, be $s_1(M_1, \alpha_2)$, which follows from the fact that $(M, \alpha_2) \sim (M_1, \alpha_2)$ by case 1 of $\sim_p$. The case that the reduction $M \to M_1$ is a sample-reduction is similar.

The case that $\alpha_1 = \alpha_2$ is trivial, because there is at most one possible $\Rightarrow$ reduction at any given position, therefore $(M_1, s_1) = (M_2, s_2)$ already.

The remaining case is that $\alpha_1 < \alpha_2$ or $\alpha_1 > \alpha_2$. Assume without loss of generality that $\alpha_1 < \alpha_2$. For each possible case of what type of redex $M|\alpha_1$ is, and α_2's position within it, there is a corresponding case of $\sim_c$, and similarly to the case where α_1 and α_2 are disjoint, the term $O_1 = O_2$ from the definition of $\sim_c$ is a suitable value of M'. The term $M \mid \alpha_1$ can't be sample, because it has strict subterms, but the case that $M \mid \alpha_2 = \mathsf{sample}$ is still somewhat more complicated. α_2 can't be within $\alpha_1; @_2$ or $\alpha_1; \mathsf{Y}; \lambda$ (cases e and f of $\sim_c$)

because if $\alpha_2 > \alpha_1; @_2$, $M \mid \alpha_1; @_2$ must be a value, therefore $\alpha_2 > \alpha_1; @_2; \lambda$ and those are not valid positions to reduce a sample. The case that α_2 is within the branch of an if statement that is deleted by the reduction at α_1 (case b) doesn't actually present a problem because there is no corresponding reduction to $M \to M_2$ in the other branch, which leaves cases c and d, that α_2 is in the $\mathsf{if}(\cdot,\cdot,\cdot)$ branch that isn't deleted, and that $\alpha_2 < \alpha_1; @_1; \lambda$. The values that the samples take in these cases match because $(M, \alpha) \sim_p (M_1, \alpha; \beta)$ by cases 3 and 2 of $\sim_p$ respectively.

In the case that $(M, s) \Rightarrow^* (M_1, s_1)$ or $(M, s) \Rightarrow^* (M_2, s_2)$ by multiple steps, it is possible to repeatedly replace a pair of the form $A \Leftarrow B \Rightarrow C$ by $A \Rightarrow^* D \Leftarrow^* C$ by the construction above, but it is not immediate that this process terminates, because each $\Rightarrow$ or $\Leftarrow$ may be replaced by multiple, so the sequence of $\Rightarrow$s and $\Leftarrow$s from (M_1, s_1) to (M_2, s_2) may get longer at some steps. However, termination can be proved by noting that the structure of this process is identical to the process of swapping $\sim_\uparrow$ and $\sim_\downarrow$ steps in Lemma 4.2. To be more precise, in this case there is a sequence of $\Leftarrow$ and $\Rightarrow$ steps, each of which has an associated reduction position and initial and final terms, and if a $\Leftarrow$ immediately precedes a $\Rightarrow$, they may be swapped to produce some number of $\Rightarrow$s, followed by some number of $\Leftarrow$s. In the case of Lemma 4.2, there is a sequence of $\sim_\uparrow$ and $\sim_\downarrow$ steps, each of which has an associated reduction position and initial and final skeletons, and if a $\sim_\uparrow$ immediately precedes a $\sim_\downarrow$, they may be swapped to produce some number of $\sim_\downarrow$s followed by some number of $\sim_\uparrow$s. In both cases, the number and reduction positions of the resultant steps are determined by the case of $\sim_c$ that matches the way the initial reduction positions overlap, and the initial skeleton (or the skeleton of the initial term). The same argument that the process in Lemma 4.2 terminated is therefore applicable here too. At every stage, the sequence of reductions that results from one of the initial $\Rightarrow$s is a parallel sequence, and swapping a parallel sequence of $\Rightarrow$s with a $\Leftarrow$ always terminates, therefore each $\Rightarrow$ in turn can be moved past all of the $\Leftarrow$s, and the process as a whole will terminate in a state where all of the $\Rightarrow$s precede all of the $\Leftarrow$s, i.e. a pair of reduction sequences $(M_1, s_1) \Rightarrow^* (M', s') \Leftarrow^* (M_2, s_2)$. □

4.4 Reduction Strategies

The reduction relation $\Rightarrow$ is nondeterministic, so it admits multiple possible reduction strategies.

Definition 4.7. A *reduction strategy* starting from a closed term M is a measurable partial function f from $Rch(M)$ to positions, such that for any reachable term N where f is defined, $f(N)$ is a position of a redex in N, and if $f(N)$ is not defined, N is a value. Using a reduction strategy f, a subset of $\Rightarrow$ that isn't nondeterministic, $\Rightarrow_f$, can be defined by $(N, s) \Rightarrow_f (N', s')$ just if $(N, s) \Rightarrow (N', s')$ and N reduces to N' with the redex at $f(N)$.

The usual call-by-value semantics can be implemented as one of these reduction strategies, given by (with V a value and T a term that isn't a value and M a general term)

$$\begin{aligned}
\mathrm{cbv}(TM) &= @_1; \mathrm{cbv}(T) \\
\mathrm{cbv}(VT) &= @_2; \mathrm{cbv}(T) \\
\mathrm{cbv}(\underline{f}(V_1, \ldots, V_{k-1}, T, M_{k+1}, \ldots, M_n)) &= \underline{f}_k; \mathrm{cbv}(T) \\
\mathrm{cbv}(\mathsf{Y}T) &= \mathsf{Y}; \mathrm{cbv}(T) \\
\mathrm{cbv}(\mathsf{if}(T < 0, M_1, M_2)) &= \mathsf{if}_1; \mathrm{cbv}(T) \\
\mathrm{cbv}(V) &\text{ is undefined} \\
\mathrm{cbv}(T) &= \cdot \text{ otherwise}
\end{aligned}$$

(this last case covers redexes at the root position).

A closed term M terminates with a given reduction strategy f and samples s if there is some natural number n such that $(M, s) \Rightarrow_f^n (N, s')$ where f gives no reduction at N. The term *terminates almost surely w.r.t. f* if it terminates with f for almost all s.

With these definitions, it will now possible to relate the confluent trace semantics back to the standard trace semantics (after a few more lemmas).

Lemma 4.6. *If $(M, \delta) \sim^* (A, \gamma)$ and $M \to^* A$, then $(M, \delta) \sim_\downarrow^* (A, \gamma)$.*

Proof. This is a simple induction on $\sim^*$. In the base case, $(A, \gamma) = (M, \delta)$ therefore $(M, \delta) \sim_\downarrow^* (A, \gamma)$ trivially. Otherwise, suppose that $(M, \delta) \sim_\downarrow^* (B, \epsilon) \sim (A, \gamma)$. Either $(B, \epsilon) \sim_p (A, \gamma)$ or $(B, \epsilon) \sim_c (A, \gamma)$. In the first case, either $B \to A$, in which case the result follows directly, or $A \to B$, in which case the fact that each (reduction sequence of) skeleton(s) has only one parent implies that a $(M, \delta) \sim_\downarrow^* (A, \gamma)$ directly as a subsequence of the path to (B, ϵ).

In the $\sim_c$ case, consider the definition of $\sim_c$. Either $O_1' = A$ and $O_2' = B$ or vice-versa. As $M \to^* N \to^* O_1 \to^* B$ and $(M, \delta) \sim_\downarrow^* (B, \epsilon)$, and $\sim_\downarrow$ only relates positions in a term and its parent, there are some positions ζ, θ such that $(M, \delta) \sim_\downarrow^* (N, \zeta) \sim_\downarrow^* (O_1, \theta) \sim_\downarrow^* (B, \epsilon)$. It follows that $(O_2, \theta) \sim_\downarrow^* (A, \gamma)$ by following the same path, therefore it suffices to provide the only missing portion of the path from M to A, i.e. to prove that $(N, \zeta) \sim_\downarrow^* (O_2, \theta)$ given $(N, \zeta) \sim_\downarrow^* (O_1, \theta)$ (or vice-versa).

If ζ is disjoint from all the positions of reduction from N to O_1 and O_2 (and consequently $\zeta = \theta$), this follows from case 1 of $\sim_\downarrow$. Otherwise, this can be proved by taking cases from the definition of $\sim_c$. This is rather long, but all of the cases are similar. The general idea is that the reductions from N to O_1 correspond to the reductions from N to O_2, so that if a position is related by $\sim_\downarrow$ across that reduction, it is related in the other branch for the same reason. Case d, where $B = O_1'$ rather than O_2', is given here in more detail as an illustrative example:

Let I be N reduced at α, and J be N reduced at $\alpha; @_1; \lambda; \beta$, so that $N \to I \to O_1$ and $N \to J \to O_2$. All of the reduction positions are $\geq \alpha$,

and ζ is not disjoint from all of them, therefore ζ is not disjoint from α. Let ι be the position such that $(N,\zeta) \sim_\downarrow (I,\iota) \sim_\downarrow (O_1,\theta)$. The fact that $(N,\zeta) \sim_\downarrow (I,\iota)$ implies that $\zeta > \alpha;@_1;\lambda$. Let $\zeta = \alpha;@_1;\lambda;\kappa$ and $\theta = \alpha;\kappa'$. If κ is disjoint from β, then $(N,\zeta) \sim_\downarrow (J,\zeta) \sim_\downarrow (O_2,\alpha;\kappa') = (O_2;\theta)$ by cases 1 and 2 of $\sim_\downarrow$. In the other case, that κ isn't disjoint from β, $\kappa > \beta$ because none of the positions $\leq \alpha;@_1;\lambda;\beta$ in N are related to any position in I by $\sim_\downarrow$. As $(I,\alpha;\kappa) \sim_\downarrow (O_1,\alpha;\kappa')$ (with the redex at $\alpha;\beta$), for exactly the same reason $(N,\alpha;@_1;\lambda;\kappa) \sim_\downarrow (J,\alpha;@_1;\lambda;\kappa')$ (with the redex at $\alpha;@_1;\lambda;\beta$). Because $(N,\alpha;@_1;\lambda;\kappa) \sim_\downarrow (J,\alpha;@_1;\lambda;\kappa')$, $N|\alpha;@_1;\lambda;\kappa = J|\alpha;@_1;\lambda;\kappa'$, and $N|\alpha;@_1;\lambda;\kappa \neq$ the variable of $N|\alpha;@_1$, therefore $J|\alpha;@_1;\lambda;\kappa'$ is also not the variable therefore $(J,\alpha;@_1;\lambda;\kappa') \sim_\downarrow (O_2,\alpha;\kappa') = (O_2,\theta)$ by case 2 of $\sim_\downarrow$. Combining these results, $(N,\zeta) \sim_\downarrow^* (O_2;\theta)$ as desired. □

Lemma 4.7. *If $M \to N$, with the redex at position α, then no position in any term reachable from N is related by $\sim^*$ to (M,α).*

Proof. Suppose on the contrary that $(M,\alpha) \sim^* (A,\beta)$, where $N \to^* A$, then by Lemma 4.6, $(M,\alpha) \sim_\downarrow^* (A,\beta)$. $M \neq A$ therefore $(M,\alpha) \sim_\downarrow$ some position in N, but in all the cases of the definition of $\sim_\downarrow$, no position in the child term is related to the position of the redex. □

Essentially what Lemma 4.7 demonstrates is that the samples taken during any reduction sequence are independent of each other. This is made more precise in the following lemmas.

Lemma 4.8. *For any skeletons $M \to N$, with the redex at position α, and measurable set of samples $S \subset I^{L_s(N)}$,*

$$\mu(S) = \mu(\{s \in I^{L_s(M)} \mid s \circ i(M \to N) \in S\})$$

and furthermore, if $M|\alpha = \mathsf{sample}$, for any $S \subset I \times I^{L_s(N)}$,

$$\mu(S) = \mu(\{s \in I^{L_s(M)} \mid (s(M,\alpha), s \circ i(M \to N)) \in S\}).$$

Proof. If the result holds for all sets S of the form $\{s \in I^{L_s(N)} \mid \forall j : s(j) \in x_i\}$, where $(x_j)_{j\in J}$ is a family of measurable subsets of I indexed by some finite set $J \subset L_s(N)$ of potential positions, it also holds for all other S by taking limits and disjoint unions.

Take such a $(x_j)_{j\in J}$, and define $K = i(M \to N)[J]$. Because $i(M \to N)$ is injective, it defines a bijection between J and K. We can then calculate the measure $\mu(\{s \in I^{L_s(M)} \mid s \circ i(M \to N) \in S\}) = \mu(\{s \in I^{L_s(M)} \mid \forall k : s(k) \in x_{i(M\to N)^{-1}(k)}\}) = \prod_{k\in K} \mu_I(x_{i(M\to N)^{-1}(k)}) = \prod_{j\in J} \mu_I(x_j) = \mu(S)$.

In the case that $M|\alpha = \mathsf{sample}$, we can similarly consider only those sets S of the form $x_{\mathsf{sample}} \times \{s \in I^{L_s(N)} \mid \forall j : s(j) \in x_i\}$. The position α in M is not related to any position in $L_s(N)$, therefore $(M,\alpha) \notin K$. Again, we can calculate the measure $\mu(\{s \in I^{L_s(M)} \mid (s(M,\alpha), s \circ i(M \to N)) \in S\}) = \mu(\{s \in I^{L_s(M)} \mid s(M,\alpha) \in x_{\mathsf{sample}}, \forall k : s(k) \in x_{i(M\to N)^{-1}(k)}\}) = \mu_I(x_{\mathsf{sample}}) \prod_{k\in K} \mu_I(x_{i(M\to N)^{-1}(k)}) = \mu_I(x_{\mathsf{sample}}) \prod_{j\in J} \mu_I(x_j) = \mu(S)$. □

Lemma 4.9. *For any initial term M, reduction strategy f on M, natural number n, skeleton N with k holes, measurable set $T \subset \mathbb{R}^k$ and measurable set S of samples in $I^{L_s(N)}$,*

$$\begin{aligned}&\mu(\{s \in I^{L_s(M)} \mid \exists r \in T, s' \in S : (M,s) \Rightarrow_f^n (N[r], s')\})\\ &= \mu(\{s \in I^{L_s(M)} \mid \exists r \in T, s' \in I^{L_s(N)} : (M,s) \Rightarrow_f^n (N[r], s')\})\ \mu(S)\end{aligned}$$

Proof. Suppose, to begin with, that $n = 0$. Either $\exists r \in T : N[r] = M$, in which case both sides of the equation are $\mu(S)$, or there is no such r, in which case both sides of the equation are 0.

For $n > 0$, suppose for induction that the lemma is true for $n-1$, for all N, T and S. If $\exists r \in T, s' \in S : (M,s) \Rightarrow_f^n (N[r], s')$, the r and s' are necessarily unique, therefore we may assume that T is the product of k measurable subsets of $\mathbb{R}$, $(T_j)_{0 \le j < k}$, as all of the other cases follow by taking unions and limits of these rectangles. Let

$$\begin{aligned}R_s &= \{Sk(O) \mid O \in Rch(M), r \in \mathbb{R}^k, O|f(O[r]) = \mathsf{sample}, O \to N[r]\}\\ R_d &= \{Sk(O) \mid O \in Rch(M), r \in \mathbb{R}^k, O|f(O[r]) \neq \mathsf{sample}, O \to N[r]\}.\end{aligned}$$

Together, R_s and R_d contain all of the skeletons of terms that can reduce to N, and they're disjoint. For each of these skeletons, there is a map g_O from the reals in O to the reals in N, i.e. $g_O(r) = r'$ if $O[r] \to N[r']$ for $O \in R_d$, and $g_O(s, r) = r'$ if $O[r] \to N[r']$ with s as the value the sample takes in the reduction. All of these functions are measurable, as they are just rearrangements of vector components, or in the case that $O|f(O[r]) = \underline{x}(\mathsf{X}, \ldots, \mathsf{X})$, a combination of the measurable function x with a rearrangement of vector components. In the case that $O \in R_s$, the function g_O simply inserts the sample into the vector of reals at a certain index, call it h_O (although for some reduction strategies, there may be multiple possible reductions for the same O, so h_O also takes the location of the reduction as an argument).

$$\begin{aligned}&\mu(\{s \in I^{L_s(M)} \mid \exists r \in T, s' \in S : (M,s) \Rightarrow_f^n (N[r], s')\})\\ =&\mu(\{s \in I^{L_s(M)} \mid \exists O \in R_s, r \in \mathbb{R}^{k-1}, s' \in I^{L_s(O)}, r' \in T, s'' \in S : (M,s) \Rightarrow_f^{n-1} (O[r], s') \Rightarrow_f (N[r'], s'')\})\\ +&\mu(\{s \in I^{L_s(M)} \mid \exists O \in R_d, r \in \bigcup_l \mathbb{R}^l, s' \in I^{L_s(O)}, r' \in T, s'' \in S : (M,s) \Rightarrow_f^{n-1} (O[r], s') \Rightarrow_f (N[r'], s'')\})\end{aligned}$$

Let's consider each of these terms separately. First,

$$\begin{aligned}
&\mu(\{s \in I^{L_s(M)} \mid \exists O \in R_s, r \in \mathbb{R}^{k-1}, s' \in I^{L_s(O)}, r' \in T, s'' \in S : (M,s) \Rightarrow_f^{n-1} (O[r], s') \Rightarrow_f (N[r'], s'')\}) \\
=&\mu(\{s \in I^{L_s(M)} \mid \exists O \in R_s, \alpha \in \text{pos}(O), r \in \mathbb{R}^{k-1}, s' \in I^{L_s(O)}, r' \in T, s'' \in S : \\
&(M,s) \Rightarrow_f^{n-1} (O[r], s') \Rightarrow_f (N[r'], s''), f(O[r]) = \alpha\}) \\
=&\sum_{O \in R_s} \sum_{\alpha \in \text{pos}(O)} \mu(\{s \in I^{L_s(M)} \mid \exists r \in \mathbb{R}^{k-1}, s' \in I^{L_s(O)} : \\
&f(O[r]) = \alpha, g_O(r, s'(O,\alpha)) \in T, s' \circ i(O \to_\alpha N) \in S, (M,s) \Rightarrow_f^{n-1} (O[r], s')\}) \\
=&\sum_{O \in R_s} \sum_{\alpha \in \text{pos}(O)} \mu(\{s \in I^{L_s(M)} \mid \exists r \in \prod_{j \neq h(O,\alpha)} T_j, s' \in I^{L_s(O)} : \\
&f(O[r]) = \alpha, s'(O,\alpha) \in T_{h(O,\alpha)}, s' \circ i(O \to_\alpha N) \in S, (M,s) \Rightarrow_f^{n-1} (O[r], s')\}) \\
=&\sum_{O \in R_s} \sum_{\alpha \in \text{pos}(O)} \mu(\{s \in I^{L_s(M)} \mid \exists r \in \prod_{j \neq h(O,\alpha)} T_j, s' \in I^{L_s(O)} : f(O[r]) = \alpha, (M,s) \Rightarrow_f^{n-1} (O[r], s')\}) \\
&\mu(\{s' \in I^{L_s(O)} \mid s'(O,\alpha) \in T_{h(O,\alpha)}, s' \circ i(O \to_\alpha N) \in S\}) \\
=&\sum_{O \in R_s} \sum_{\alpha \in \text{pos}(O)} \mu(\{s \in I^{L_s(M)} \mid \exists r \in \prod_{j \neq h(O,\alpha)} T_j, s' \in I^{L_s(O)} : \\
&f(O[r]) = \alpha, (M,s) \Rightarrow_f^{n-1} (O[r], s')\}) \mu(T_{h(O,\alpha)}) \mu(S) \\
=&\sum_{O \in R_s} \sum_{\alpha \in \text{pos}(O)} \mu(\{s \in I^{L_s(M)} \mid \exists r \in \prod_{j \neq h(O,\alpha)} T_j, s' \in I^{L_s(O)} : \\
&f(O[r]) = \alpha, s'(O,\alpha) \in T_{h(O,\alpha)}, (M,s) \Rightarrow_f^{n-1} (O[r], s')\}) \mu(S) \\
=&\sum_{O \in R_s} \mu(\{s \in I^{L_s(M)} \mid \exists r \in \mathbb{R}^{k-1}, s' \in I^{L_s(O)} : g_O(r, s'(O, f(O[r]))) \in T, (M,s) \Rightarrow_f^{n-1} (O[r], s')\}) \mu(S) \\
=&\mu(\{s \in I^{L_s(M)} \mid \exists O \in R_s, r \in \mathbb{R}^{k-1}, s' \in I^{L_s(O)}, r' \in T, s'' \in I^{L_s(N)} : \\
&(M,s) \Rightarrow_f^{n-1} (O[r], s') \Rightarrow_f (N[r'], s'')\}) \mu(S)
\end{aligned}$$

The other case is similar.

$$\begin{aligned}
&\mu(\{s \in I^{L_s(M)} \mid \exists O \in R_d, r \in \bigcup_l \mathbb{R}^l, s' \in I^{L_s(O)}, r' \in T, s'' \in S : (M,s) \Rightarrow_f^{n-1} (O[r], s') \Rightarrow_f (N[r'], s'')\}) \\
=&\mu(\{s \in I^{L_s(M)} \mid \exists O \in R_d, \alpha \in \mathrm{pos}(O), r \in \bigcup_l \mathbb{R}^l, s' \in I^{L_s(O)}, r' \in T, s'' \in S : \\
&(M,s) \Rightarrow_f^{n-1} (O[r], s') \Rightarrow_f (N[r'], s'')\}, f(O[r]) = \alpha) \\
=&\sum_{O \in R_d} \sum_{\alpha \in \mathrm{pos}(O)} \mu(\{s \in I^{L_s(M)} \mid \exists r \in \bigcup_l \mathbb{R}^l, s' \in I^{L_s(O)} : \\
&f(O[r]) = \alpha, g_O(r) \in T, s' \circ i(O \to_\alpha N) \in S, (M,s) \Rightarrow_f^{n-1} (O[r], s')\}) \\
=&\sum_{O \in R_d} \sum_{\alpha \in \mathrm{pos}(O)} \mu(\{s \in I^{L_s(M)} \mid \exists r \in \bigcup_l \mathbb{R}^l, s' \in I^{L_s(O)} : \\
&f(O[r]) = \alpha, g_O(r) \in T, s' \circ i(O \to_\alpha N) \in S, (M,s) \Rightarrow_f^{n-1} (O[r], s')\}) \\
=&\sum_{O \in R_d} \sum_{\alpha \in \mathrm{pos}(O)} \mu(\{s \in I^{L_s(M)} \mid \exists r \in \bigcup_l \mathbb{R}^l, s' \in I^{L_s(O)} : f(O[r]) = \alpha, g_O(r) \in T, (M,s) \Rightarrow_f^{n-1} (O[r], s')\}) \\
&\mu(\{s' \in I^{L_s(O)} \mid s' \circ i(O \to_\alpha N) \in S\}) \\
=&\sum_{O \in R_d} \sum_{\alpha \in \mathrm{pos}(O)} \mu(\{s \in I^{L_s(M)} \mid \exists r \in \bigcup_l \mathbb{R}^l, s' \in I^{L_s(O)} : f(O[r]) = \alpha, g_O(r) \in T, (M,s) \Rightarrow_f^{n-1} (O[r], s')\})\mu(S) \\
=&\sum_{O \in R_d} \mu(\{s \in I^{L_s(M)} \mid \exists r \in \bigcup_l \mathbb{R}^l, s' \in I^{L_s(O)} : g_O(r) \in T, (M,s) \Rightarrow_f^{n-1} (O[r], s')\})\mu(S) \\
=&\mu(\{s \in I^{L_s(M)} \mid \exists O \in R_d, r \in \bigcup_l \mathbb{R}^l, s' \in I^{L_s(O)}, r' \in T, s'' \in I^{L_s(N)} : (M,s) \Rightarrow_f^{n-1} (O[r], s') \Rightarrow_f (N[r'], s'')\}) \\
&\mu(S)
\end{aligned}$$

Combining the terms together again, we can conclude

$$\begin{aligned}
&\mu(\{s \in I^{L_s(M)} \mid \exists r \in T, s' \in S : (M,s) \Rightarrow_f^n (N[r], s')\}) \\
=&\mu(\{s \in I^{L_s(M)} \mid \exists O \in R_s, r \in \mathbb{R}^{k-1}, s' \in I^{L_s(O)}, r' \in T, s'' \in I^{L_s(N)} : \\
&\quad (M,s) \Rightarrow_f^{n-1} (O[r], s') \Rightarrow_f (N[r'], s'')\})\mu(S) \\
+&\mu(\{s \in I^{L_s(M)} \mid \exists O \in R_d, r \in \bigcup_l \mathbb{R}^l, s' \in I^{L_s(O)}, r' \in T, s'' \in I^{L_s(N)} : \\
&\quad (M,s) \Rightarrow_f^{n-1} (O[r], s') \Rightarrow_f (N[r'], s'')\})\mu(S) \\
=&\mu(\{s \in I^{L_s(M)} \mid \exists r \in T, s' \in I^{L_s(N)} : (M,s) \Rightarrow_f^n (N[r], s')\})\mu(S)
\end{aligned}$$

□

Theorem 4.2. *A closed term M is AST with respect to* cbv *iff it is AST in the original sense of Section 2.4.2.*

Proof. For both red and $\Rightarrow_{\mathrm{cbv}}$, the probability measure on the sample space can be used to define a measure on the reduction sequences of each finite length. In

the case that the reduction sequences terminate after a finite number of steps by reaching a value, the reduction sequence defined using $\Rightarrow_{\mathrm{cbv}}$ should be taken to continue by repeating the last term indefinitely, as red does, even though $\Rightarrow_{\mathrm{cbv}}$ specifies no next step in this case. The distributions are equal for each length, by induction, as follows.

The base case, length 0, is trivial, because in both cases the distribution has probability 1 on the sequence containing only M.

For the inductive case, suppose that the distributions of reduction sequences of length n are equal.

For any term N which is not a value, there is a unique environment E and redex R such that $N = E[R]$. The position of the hole in E is equal to $\mathrm{cbv}(N)$, so that $N|\,\mathrm{cbv}(N) = R$, as the cases in the definition of cbv match the cases in the definition of environments. If $R \neq \mathsf{sample}$, let R' be the result of reducing R (which is equal in both versions of the semantics), then $E[R'] = N[R'/\,\mathrm{cbv}(N)]$ therefore $\mathrm{red}(N, s) = (E[R'], s)$ and $(N, s) \Rightarrow_{\mathrm{cbv}} (E[R'], s)$. The distribution of next terms, conditional on the previous term, in the case that that previous term reduces deterministically, is therefore equal for red and $\Rightarrow_{\mathrm{cbv}}$.

In the other case that $R = \mathsf{sample}$, the first (term) part of $\mathrm{red}(N, s)$ is $E[\underline{\pi_h(s)}]$. If k is the number of samples already taken in this reduction sequence, this is equivalent to $E[\underline{s_0(k)}]$, where s_0 is the initial sample. For any sufficiently small neighbourhood of N (i.e. containing only reduction sequences with the same skeletons), the probability of reaching this neighbourhood is independent of $s(k)$, because it depends only on the samples taken so far, and the samples are independent. The distribution of $s(k)$ conditional on reaching N is therefore the uniform distribution on I. Similarly for the other semantics, the distribution of remaining samples after n steps is independent of the term by Lemma 4.9, therefore the distribution of $s(Sk(N), \mathrm{cbv}(N))$ conditional on N is the uniform distribution on I. In either case, the distribution of the next term conditional on the previous term, in the case that it reduces randomly, is equal to the image under $r \mapsto E[\underline{r}]$ of the uniform distribution on I.

In any case, the distribution of reduction sequences after $n + 1$ steps, conditional on the reduction sequence after n steps, as defined by $\Rightarrow_{\mathrm{cbv}}$ and by red, is equal, and by the inductive hypothesis the distributions after n steps are equal, therefore by integrating over the reduction sequence up to step n, the distributions on reduction sequences up to step $n + 1$ are equal.

The distributions on reduction sequences of any finite length as defined by $\Rightarrow_{\mathrm{cbv}}$ and red are therefore equal, therefore so is the probability of having reached a value after n steps, therefore so is its limit, the probability of termination, therefore the probability of termination is 1 iff the probability of termination with respect to cbv is 1. □

It is also true that the distributions of terms produced by the standard semantics and by the reduction strategy cbv are the same, for much the same reason, although that is not relevant to the main ranking-function related results.

Theorem 4.3. *If M terminates with some reduction strategy f and trace s, it terminates with* cbv *and s.*

Proof. Suppose M terminates with the reduction strategy f and samples s. Let $(M, s) = (M_0, s_0) \Rightarrow_f \cdots \Rightarrow_f (M_n, s_n)$, where M_n is a value. By Lemma 4.4, the reduction sequence $(Sk(M_i))_i$ is related by $\sim_c^*$ to some skeletal reduction sequence X which is in call-by-value order. For each skeletal reduction sequence X_j in the sequence $(Sk(M_i))_i \sim_c \cdots \sim_c X$, it is possible to define a reduction sequence of terms such that $(M, s) = (N_{j,0}, s_{j,0}) \Rightarrow \cdots \Rightarrow (N_{j,n_j}, s_{j,n_j})$ and $Sk(N_{j,k}) = X_{j,k}$, by at each step applying the reduction at the same place as in X_j, essentially just filling in the reals in the skeletal reduction sequence to make it a reduction sequence of terms (it will be shown shortly that the correct branch is taken in each if-reduction). Assume for induction on j that $(N_{j,n_j}, s_{j,n_j}) = (M_n, s_n)$. To prove that this holds true for $j + 1$ as well, it suffices to show that the corresponding term and samples immediately after the reductions involved in this $\sim_c$ step match, as from that point on, the $\Rightarrow$-sequences will match, as they start from the same point and have all the same reduction positions. Letting the term and remaining samples immediately before the reductions involved in this $\sim_c$ step be (N, t) (which is the same in both N_j and N_{j+1} for the same reason again, they start at the same point and the sequences of reduction positions up to this point are the same), and the terms and samples after these reductions be (O_1, t_1) and (O_2, t_2), so that these match the N, O_1 and O_2 in the definition of $\sim_c$, $t_1 = t_2$ because both of them are, by Lemma 4.5, equal to the restriction of t to the subset of $Rch(N)$ that starts with the reduction sequences to O_1 and O_2 respectively, and the potential positions in those subsets of $Rch(N)$ are identified with each other by $\sim$ because of the same case of $\sim_c$ that means N_j and N_{j+1} are related. If none of the relevant reductions are sample-reductions, $O_1 = O_2$ trivially. If there are samples taken (which must be the arbitrary reduction at β), it is also required that the same samples are taken in both branches. Similarly to Theorem 4.1, the positions of these sample-reductions in each branch are related by $\sim_p^*$, therefore they correspond to the same sample in t, therefore $O_1 = O_2$. The fact that the terms are equal also implies that the same branch of an if statements will be taken.

There is therefore a reduction sequence $(M, s) = (N_0, s'_0) \Rightarrow \cdots \Rightarrow (N_m, s'_m)$ which ends in a value ($N_m = M_n$) and is in CBV order. It doesn't immediately follow that this is the $\Rightarrow_{\text{cbv}}$ reduction sequence starting from (M, s) because the fact that $(N_i)_i$ is in CBV order only means that the reductions that do take place in this reduction sequence are in the correct order, not that the reduction sequence finishes once it reaches a value, or that there are no other reductions that would come earlier in CBV order that don't occur at all in $(N_i)_i$. Suppose that at some point j in the reduction sequence, the reduction that occurs is not the next reduction in CBV order, but something else, and that the reduction that should be next is at position $\alpha = \text{cbv}(N_i)$, and the position of the actual reduction $N_j \to N_{j+1}$ is β. Suppose further, for a contradiction, that N_j is not a value. Either $\alpha = \cdot$, the root position, or the reduction at the root position

is dependent in some way on the reduction at α (e.g. $\alpha = \mathsf{if}_1$, or $\alpha = \pm; @_1$). In any case, all the reductions in the rest of the reduction sequence are after α in CBV order, therefore none of them are $\leq \alpha$, and there continues to be an unreduced redex at α, therefore the reduction at the root position never occurs, therefore N_m is not a value, which is a contradiction, therefore at the point (if any) where $(N_i)_i$ departs from the $\Rightarrow_{\text{cbv}}$ reduction sequence starting at (M, s), it has already reached a value, therefore $(M, s) \Rightarrow_{\text{cbv}} \ldots$ does reach a value eventually, i.e. M terminates with the reduction strategy cbv and samples s. □

It is not quite true, however, that all reduction strategies produce the same values (even ignoring cases where some reduction strategies terminating where others don't), because it is possible for a reduction strategy to overshoot the value that the standard semantics produces by, for example, performing reductions inside of a lambda. Even for the same value of the samples, the final values may be different. A more restricted result is true, however, that if a term terminates with some reduction strategy f and samples s, the value that $\Rightarrow_{\text{cbv}}$ terminates with can reduce via $\Rightarrow$ to the value that $\Rightarrow_f$ terminates with (as a corollary to this proof).

Corollary 4.1 (Reduction strategy independence)**.** *If M is AST with respect to any reduction strategy, it is AST.*

Proof. Suppose M is AST w.r.t. f. Let the set of samples with which it terminates with this reduction strategy be X. By Theorem 4.3, M also terminates with cbv and every element of X, and X has measure 1, by assumption, therefore M is AST with respect to cbv therefore by Theorem 4.2 it is AST. □

All of the theorems on the termination of rankable terms therefore extend to other reduction strategies too. The proofs of Theorems 3.4, 3.7, 3.9 and 3.10 are all sufficiently generic with respect to what the reduction relation actually is that they can be directly applied to other reduction strategies. They only require that the number of reductions that can occur without any of them being a Y-reduction is bounded for any starting term (which is true, because Theorem 3.3 applies equally to any reduction strategy). For a reduction strategy r and term M, just substitute $N \mid r(N)$ being a Y-redex for N being of the form $E[\mathsf{Y}\lambda x.O]$, and $r(N)$ being undefined for N being a value.

The domain of definition of the ranking functions also needs to be changed from the reachable terms $Rch(M)$ to the *reachable terms with respect to the reduction strategy* r,

$$Rch_r(M) := \{N \mid \exists n, (N_i)_{0 \leq i \leq n} : \\ N_0 = M, N_n = N, N_i \to N_{i+1} \text{ at } r(N_i)\}.$$

and almost complete reachable sets with respect to a reduction strategy are defined similarly.

More explicitly, the modified forms of the theorems are:

Definition 4.8. An *antitone ranking function on M w.r.t. a reduction strategy r* is a measurable function $f : Rch_r(M) \to \mathbb{R}$ such that $f(N) \geq 0$ for all N, and there exists an antitone function $\varepsilon : \mathbb{R}_{\geq 0} \to \mathbb{R}_{>0}$ and almost complete reachable set w.r.t. r, C, such that

- $f(N) \geq \varepsilon(f(N)) + f(N')$ if $N \mid r(N)$ is a Y-redex and N reduces to N' at $r(N)$
- $f(N) \geq \int_I f(N[\underline{x}/r(N)])\,\mathrm{d}x$ if $N \mid r(N) = \mathsf{sample}$
- $f(N) \geq f(N')$ if $N \mid r(N)$ is any other redex, where $N \to N'$ at $r(N)$

for all N in C. Any closed term for which an antitone ranking function with respect to a reduction strategy r exists is called *antitone rankable w.r.t. r*.

Definition 4.9. An *antitone sparse ranking function on M w.r.t. a reduction strategy r* is a partial function $f : Rch_r(M) \rightharpoonup \mathbb{R}$ such that for some antitone function $\varepsilon : \mathbb{R}_{\geq 0} \to \mathbb{R}_{>0}$,

- $f(N) \geq 0$ for all N where f is defined,
- f is defined at M,
- for any N in the domain of definition of f, evaluation of N at the positions specified by r will almost certainly eventually reach some O such that either $r(O)$ isn't defined or $f(O)$ is, and $f(N) \geq \mathbb{E}[f(O) + \varepsilon(f(O)) \times$ the number of Y-reduction steps from N to $O]$ (where $f(O)$ is taken to be 0 if O is a value outside of the domain of f).

If the antitone function is just the constant function 1 instead, then it is called a *(sparse) ranking function on M w.r.t. r* instead, and the term is called *rankable w.r.t. r*.

Theorem 4.4 (Confluent ranking)**.** *If a closed PPCF term M has a ranking function, sparse ranking function, antitone ranking function or antitone sparse ranking function w.r.t. a reduction strategy r, then M is AST w.r.t. r, and AST.*

Proof. The same arguments as in Theorems 3.4, 3.7, 3.9 and 3.10 imply that M is AST with respect to r. Applying Corollary 4.1, this implies that M is AST. □

4.5 Applying Alternative Reduction Strategies

Theorem 4.4 is not actually any more powerful than the original antitone ranking result, Theorem 3.9, since every AST term is already antitone rankable (Theorem 3.12). However, there are some terms which terminate more simply and directly via some alternative reduction strategy. When constructing an (antitone) sparse ranking function with respect to an alternative reduction strategy, it is intuitively simplest to define the sparse ranking function and the

reduction strategy together. For each key reachable term where the sparse ranking function is defined, simply decide which sub-term should be reduced next, and how far before the next key reachable term. The only restrictions to bear in mind on evaluating sub-terms are that a sample may not be evaluated inside a λ that's inside a Y or on the right of an application, and a β redex may not be reduced until its argument is a value. This will be demonstrated with some examples.

Example 4.5.1 (Sum of powers)**.** The following term picks a random natural number n, then computes $\sum_{k=0}^{n} k^n$.

$$\begin{aligned} M &= (\lambda n.P[n])\lfloor \mathsf{sample}^{-1/2} \rfloor \\ P[n] &= (\lambda p.\Xi[p]\, n)(\lambda x.\Theta[x]\, n)) \\ \Xi[p] &= \mathsf{Y}\lambda f n.\mathsf{if}\big(n = 0, \underline{0}, p\, n + f(n-1)\big) \\ \Theta[x] &= \mathsf{Y}\lambda f n.\mathsf{if}\big(n = 0, \underline{1}, x \times f(n-1)\big). \end{aligned}$$

With the CBV reduction strategy, the recursion in $\Theta[x]$ (that defines the function $k \mapsto k^x$) is executed separately for every term in the sum in Ξ. It is much simpler to use the following reduction strategy r instead:

$$\begin{aligned} & M \\ \to^* \; & P[\underline{n}] \\ \to^* \; & (\lambda p.\ \Xi[p]\ \underline{n})\ (\lambda x.\underbrace{x \times \cdots \times x \times}_{n} \underline{1}) \\ \to \; & \Xi[\lambda x.\underbrace{x \times \cdots \times x \times}_{n} \underline{1}]\ \underline{n} \\ \to^* \; & \underline{n^n} + \underline{(n-1)^n} + \cdots + \underline{0} \\ \to^* \; & \underline{\sum_{k=0}^{n} k^n} \end{aligned}$$

(this is a little underspecified, but the details are not important). A suitable sparse ranking function for it is

$$\begin{aligned} M &\mapsto \frac{\pi^2}{3} \\ P[\underline{n}] &\mapsto 2n+2 \\ (\lambda p.\ \Xi[p]\ \underline{n})\ (\lambda x.\underbrace{x \times \cdots \times x \times}_{n-k}(\Theta[x]\ \underline{k})) &\mapsto n+k+2 \\ \Xi[\lambda x.\underbrace{x \times \cdots \times x \times}_{n} \underline{1}]\ \underline{n} &\mapsto n+1 \\ \underline{n^n} + \cdots + \underline{(n-k+1)^n} + \Xi[\lambda x.x \times \ldots \underline{1}]\ \underline{k} &\mapsto k+1 \\ \underline{\sum_{k=0}^{n} k^n} &\mapsto 0. \end{aligned}$$

The first step is justified because the probability of reaching $P[\underline{n}]$ for any specific $n \geq 0$ is $(n+1)^{-2} - (n+2)^{-2} = \frac{2n+3}{(n+1)^2(n+2)^2}$, therefore the expected next value of the ranking function is $\sum_{n=0}^{\infty} \frac{(2n+2)(2n+3)}{(n+1)^2(n+2)^2} = \frac{\pi^2}{3}$. The other steps are deterministic, and involve 1 or 0 Y-reductions each. Given that all of the reduction steps after the first are deterministic, and the number of Y reduction steps is not that hard to count, defining the sparse ranking function's values only at M and $P[\underline{n}]$ would also have been reasonable, although in a similar term which had more random samples throughout, that would not have been so simple.

Providing a ranking function of some sort for this term in a similar level of detail would have been rather more complicated using only the standard reduction strategy. The recursion in Θ would have to be evaluated separately for every time it was used, so there would have been more (and more complex) terms to consider in the reduction sequence. Also, the number of Y-reductions from $P[\underline{n}]$ would have been n^2+2n+1 instead of $2n+2$, therefore the expected number of Y reductions starting from M would be infinite, i.e. M is not Y-PAST, therefore it is not even rankable. It would be possible to define an antitone sparse ranking function for it, but it is much more difficult to construct:

$$
\begin{aligned}
M &\mapsto 1.47 \\
P[\underline{n}] &\mapsto 1 + \ln((n+1)n+1) \\
\Xi[\lambda x.\Theta[x]\,\underline{n}]\,\underline{n} &\mapsto 1 + \ln((n+1)n+1) \\
\left.\begin{array}{r} \underline{n^n} + \cdots + \underline{k_1^n} + \\ \Xi[\lambda x.\Theta[x]\underline{n}](\underline{k_1}-1) \end{array}\right\} &\mapsto 1 + \ln((n+1)(k_1-1)+1) \\
\left.\begin{array}{r} \underline{n^n} + \cdots + \underline{(k_1+1)^n} + \\ \Theta[\underline{k_1}]\,\underline{n} + \\ \Xi[\lambda x.\Theta[x]\,\underline{n}](\underline{k_1}-1) \end{array}\right\} &\mapsto 1 + \ln((n+1)k_1+1) \\
\left.\begin{array}{r} \underline{n^n} + \cdots + \underline{(k_1+1)^n} + \\ \underbrace{\underline{k_1} \times \cdots \times \underline{k_1}}_{n-k_2+1} \times \\ \Theta[\underline{k_1}](\underline{k_2}-1) + \\ \Xi[\lambda x.\Theta[x]\,\underline{n}](\underline{k_1}-1) \end{array}\right\} &\mapsto 1 + \ln((n+1)k_1+k_2+1) \\
\underline{\textstyle\sum_{k=0}^{n} k^n} &\mapsto 0.
\end{aligned}
$$

It wouldn't even be possible in this case to give a sparser version of the antitone ranking function, defined only at M and $P[\underline{n}]$, because the amount that an antitone sparse ranking function must decrease at each Y-reduction step depends on the value of the ranking function at the next term where it is defined, therefore it is necessary to have these intermediate steps in order for the ranking function to be able to change sufficiently slowly.

Example 4.5.2. Even some of the examples given earlier implicitly use slightly non-standard reduction strategies to make the definitions of their ranking functions slightly simpler. For example, the antitone sparse ranking function in

Example 3.1.2 was defined at $\Theta_2\,\underline{n}$, where

$$\Theta_2 = \mathsf{Y}\,\lambda f n.\,\mathsf{if}\big(n = 0, 0, f\,(n-1) \oplus_{1/2} f\,(n+1)\big),$$

but $\Theta_2\,\underline{10}$ actually reduces to $\Theta_2\,(\underline{10}-1)$ or $\Theta_2\,(\underline{10}+1)$, and (taking the second of these), it then reduces to

$$(\lambda z.(\lambda n.\,\mathsf{if}\big(n = 0, 0, \Theta_2\,(n-1) \oplus_{1/2} \Theta_2\,(n+1)\big))z)\,(\underline{10}+1),$$

then

$$(\lambda z.(\lambda n.\,\mathsf{if}\big(n = 0, 0, \Theta_2\,(n-1) \oplus_{1/2} \Theta_2\,(n+1)\big))z)\,\underline{11},$$

bypassing $\Theta_2\,\underline{11}$ entirely. The antitone sparse ranking function in the form given can be justified anyway, by considering it to be using a reduction strategy which is the same as cbv except that in $\Theta_2\,(\underline{f}\,\underline{n})$, it evaluates the redex at $@_2$ first. Similar solutions can be given for the other examples in that section.

Example 3.3.8 also makes use of an alternative reduction strategy, but in a different way. In this case, it is to allow the antitone sparse ranking function to skip over all of those terms where the unknown function F is part way through evaluation.

Recall that

$$\begin{aligned}\Xi := \mathsf{Y}\lambda\varphi\, f^{\mathsf{R}\to\mathsf{R}\to\mathsf{R}}\, s^{\mathsf{R}}\, n^{\mathsf{R}}.\\ \quad \mathsf{if}\big(n \le 0, s, f\,n\,(\varphi\, f\, s\,(n-1)) \oplus_p f\,n\,(\varphi\, f\, s\,(n+1))\big)\end{aligned}$$

and we want to show that $\Xi\ F\ \underline{x}\ \underline{y}$ is rankable for any $F : \mathsf{R} \to \mathsf{R} \to \mathsf{R}$ such that F is a value and $F\,\underline{n}\,\underline{m}$ terminates with no Y reductions for all m and n. A suitable reduction strategy is

$$\underbrace{FA\ (FB\ (\ldots(}_{n}\ M)\ldots)) \mapsto @_2^n; \mathrm{cbv}(M), \text{ where } n \text{ is as large as possible.}$$

With this reduction strategy, the antitone sparse ranking function originally given for Example 3.3.8 is now correct.

Example 4.5.3. In Example 3.1.2, I mentioned that it was possible to rearrange the unbiased random walk in such a way that it used fewer Y-reduction steps and therefore became rankable, but put off actually providing a ranking function for the term to prove it. Now, with the ability to use sparse ranking functions with respect to alternative reduction strategies, that term is a bit more manageable. A term can be Y-PAST with respect to an alternative reduction strategy without being actually Y-PAST, as seen in Example 4.5.1, if the reduction strategy avoids duplicating some of the Y-reductions, so this will not be a rigorous proof that this term is rankable, but it should be reasonably obvious that the reduction strategy is not making the term terminate in fewer Y-reductions in this case, so it is actually Y-PAST.

The term in question is $W\,S\,\underline{10}$, where

$$\begin{aligned}W =&(Y\lambda wf.f(w(\lambda c.f(fc))))\\ S =&(\lambda cn.\mathsf{if}\big(n < 1, 0, c(n+\underline{1}) \oplus_{1/2} c(n-\underline{1})\big))\end{aligned}$$

We will also need the notation

$$\begin{aligned}S^{n+1}(C) =& \lambda n.\mathsf{if}(n < 1, 0, C(n + \underline{1}) \oplus_{1/2} C(n - \underline{1}))\\ S^0(C) =& C\end{aligned}$$

then the reduction strategy is

$$\begin{aligned}(W\ \lambda c.S^x(c))\ \underline{y} &\to^* S^x(W\ \lambda c.S^{2x}(c))\ \underline{y}\\ S^{z+1}(W\ \lambda c.S^x(c))\ \underline{y} &\to^* S^z(W\ \lambda c.S^x(c))\ \underline{y \pm 1},\ \text{where } y \geq 1\\ S^{z+1}(W\ S_x)\ \underline{0} &\to^* 0\end{aligned}$$

and the sparse ranking function is

$$f((W\ \lambda c.S^x(c))\ \underline{y}) = 2.1\frac{y}{\sqrt{x}} + 5.3$$

Take one of these terms where f is defined, with value f_0. Evaluating this term, it will pass through a sequence of terms of the form $S^{z+1}(W\ \lambda c.S^{2x}(c))\ \underline{y'}$, where the y's form a random walk of length x before reaching $(W\ \lambda c.S^{2x}(c))\ \underline{y'}$, or if y' reaches 0 at any point, it will terminate instead. In any case, there is one Y reduction in this sequence. Let y' be a random variable which is the value after these x steps (or if it terminates before then, let y' take the value if would have had without the termination), and let t be the event that this term terminates before reaching the next term where f is defined. The next value of f will be $f_1 = (1 - \mathbf{1}_t)(2.1\frac{y'}{\sqrt{2x}} + 5.3)$. Since x is a constant, and the expectation of y' conditional on t is 0 (since in that case it is effectively the result of a random walk starting at 0), the expectation of f_1 is

$$\begin{aligned}&\mathbb{E}\left[2.1\frac{y'}{\sqrt{2x}} + 5.3(1 - \mathbf{1}_t)\right]\\ =&2.1\frac{y}{\sqrt{2x}} + 5.3 - 5.3\mathbb{P}[t]\\ =&f_0 - \frac{2.1}{1 - \sqrt{2}}\frac{y}{\sqrt{x}} - 5.3\mathbb{P}[t]\end{aligned}$$

If $y' \leq 0$, clearly it terminates, therefore $\mathbb{P}[t] \geq \mathbb{P}[y' < 0]$. Because of the starting value, $|y - 10| \leq x - 1$, and x is always a power of 2, therefore if $\frac{y}{\sqrt{x}} \leq 0.2$, $x \geq 16$, so it follows from the Berry-Esseen Theorem[5] that if $\frac{y}{\sqrt{x}} \leq 0.2$, $\mathbb{P}[t] \geq 0.189$, therefore in this case, $5.3\mathbb{P}[t] > 1$ and $\mathbb{E}[f_1] \leq f_0 - 1$. In the other case, where $\frac{y}{\sqrt{x}} > 0.2$, the other term $\frac{2.1}{1-\sqrt{2}}\frac{y}{\sqrt{x}}$ is greater than 1, so again, $\mathbb{E}[f_1] \leq f_0 - 1$. The sparse ranking function condition is therefore satisfied, and this is a valid sparse ranking function w.r.t. the given reduction strategy.

Chapter 5

Further Work

Although the method of ranking functions provides a way of proving AST and PAST, if it is to be useful in practice, there should be some way of applying it at least semi-automatically. The ultimate goal is to allow interpreters and compilers for probabilistic languages to apply the various theorems that rely on AST (e.g. [46, 25, 36]) to actual programs, confident in the knowledge that the programs are actually AST. While it is possible for the programmer to check this, that was already possible without the formal proofs presented here, and is far from ideal, so some way of allowing compilers to construct or at least check ranking functions would be useful. Ranking functions for higher-order programs are inherently rather complicated, being functions from the potentially very large and varied space $Rch(M)$. This was why the sparse ranking function and alternative reduction strategy extensions were necessary, but it also would make automation of constructing ranking functions rather complicated. More so than ranking functions for imperative programs, or other methods such as the counting-based recursion analysis of [6]. Nevertheless, it seems plausible that it would be possible in at least some cases.

While the method of ranking functions I have presented is complete, there are still other directions it could be extended in to make it easier to apply. For one thing, it is not very compositional. Perhaps it would be possible to extend it such that it could be applied to sections of a program independently, rather than analyzing the whole program at once. If the progress condition for antitone ranking functions could be replaced with the quantile strict condition (Definition 3.9) without ruining sparseness, that would make using the method a little simpler. And of course, there are many completely different methods of proving AST that are suitable for different circumstances, and I'm sure there are many more yet to be discovered.

There have been many theorems that prove the equivalence of two different semanticses for PPCF and related languages[48, 8, 16, 36, 20], but these can't all be linked together to prove that all of them are equivalent at once, because they

use different versions of the language. It would seem worthwhile to present all of them in a unified way, with an operational trace semantics, a denotational trace semantics, an operational distribution semantics, a denotational distribution semantics, and perhaps more variants besides, all exactly equivalent to each other rather than just roughly equivalent as seems to be the case for currently proven results for any set of semanticses so broad.

One further research direction I think could be particularly interesting is re-defining how the samples are arranged in the confluent trace semantics in a more algebraic way, and connecting this to a similar version of denotational trace semantics. In the version of confluent trace semantics I described in Chapter 4, the labels for samples are based on the position where they occur, and which term in the reachable set they occur in. In a term $M = E[(\lambda x.A)\ B]$, which reduces to $M' = E[A[B/x]]$, all of the positions in the copies of B in M' are entirely unrelated in this scheme to the corresponding positions in M. In the alternative version of the confluent trace semantics I am imagining, if the redex is at position α, $A \mid \beta = x$, and B contains some position γ, then the position $\alpha;\beta;\gamma$ in M' would be considered equivalent to $f(\alpha,\beta,\gamma)$ for some operator f on some sort of algebraic structure generated by the positions in the initial term. With a suitable set of rules for equivalences within this structure, it may be possible to reconstruct the equivalences between positions in different reachable terms that were necessary to ensure confluence, while also bypassing a lot of the messier arguments involved in getting the original version of the confluent trace semantics to work.

The same labelling scheme would be particularly suitable also for a denotational trace semantics, since the compositional nature of this algebra of potential positions mirrors the compositional way a denotational semantics works. Such a semantics would be particularly easy to prove equivalent to that version of the confluent trace semantics, and would also make it easier to prove program equivalences which, as mentioned in Section 2.5, can be quite complicated in the more basic versions of denotational trace semantics.

Even without going so far as to entirely redefine how potential positions work, the proofs related to confluence could be considerably simplified if the Y-reduction rule were simplified. As noted in Section 2.4, the Y rule in PPCF is more complicated than the obvious $\mathsf{Y}M \to M(\mathsf{Y}M)$ rule in two ways, where either one alone would suffice. A large proportion of the reasoning in Lemma 4.4, the key lemma in the entire proof, is dedicated just to case f in Definition 4.3 (the definition of $\sim_c$), because case f deals with the Y-reduction rule which substitutes a term into itself, and is far more complicated than the other five cases. If instead, the reduction rule $\mathsf{Y}M \to \lambda z.M(\mathsf{Y}M)z$ had been used, the proofs could have been much simpler. Having already proven the more complicated version, I was not able to make this simplification due to time constraints.

More generally, the confluent trace semantics I have defined, and the proofs that it works right, are very specific to this particular version of PPCF, which is a little unsatisfactory. While I don't have any specific suggestions for how this might be done, it would be nice to have a more general version of the proof that does not require an enormous amount of effort to adapt to slightly different languages. Perhaps using the algebraic version of potential positions would be a step in this direction, I'm not sure.

Bibliography

[1] Robert J Aumann. "Borel structures for function spaces". In: *Illinois Journal of Mathematics* 5.4 (1961), pp. 614–630. URL: https://projecteuclid.org/journals/illinois-journal-of-mathematics/volume-5/issue-4/Borel-structures-for-function-spaces/10.1215/ijm/1255631584.pdf.

[2] Robert B. Ash and Catherine Doléans-Dade. *Probability and measure Theory*. Harcourt Academic Press, 2000.

[3] Giorgio Bacci, Robert Furber, Dexter Kozen, Radu Mardare, Prakash Panangaden, and Dana Scott. "Boolean-valued semantics for the stochastic λ-calculus". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. 2018, pp. 669–678. URL: https://strathprints.strath.ac.uk/74747/1/Bacci_etal_LICS_2018_Boolean_valued_semantics_for_the_stochastic_lamda_calculus.pdf.

[4] Henk Barendregt. *The Lambda Calculus, Its Syntax and Semantics*. 2nd. North-Holland, 1984.

[5] Paul van Beek. "An application of Fourier methods to the problem of sharpening the Berry-Esseen inequality". In: *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete* 23 (1972), pp. 187–196. URL: https://doi.org/10.1007/BF00536558.

[6] Raven Beutner and Luke Ong. "On Probabilistic Termination of Functional Programs with Continuous Distributions". In: *Proceedings of the 42nd ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2021, Canada, June 20-25, 2021*. ACM, 2021. URL: https://arxiv.org/pdf/2104.04990.pdf.

[7] Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. "Pyro: Deep universal probabilistic programming". In: *The Journal of Machine Learning Research* 20.1 (2019), pp. 973–978. URL: https://www.jmlr.org/papers/volume20/18-403/18-403.pdf.

[8] Johannes Borgström, Ugo Dal Lago, Andrew D. Gordon, and Marcin Szymczak. "A lambda-calculus foundation for universal probabilistic programming". In: *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming, ICFP 2016, Nara, Japan, September 18-22, 2016*. 2016, pp. 33–46. URL: https://arxiv.org/pdf/1512.08990.pdf.

[9] Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus A Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. "Stan: A probabilistic programming language". In: *Journal of statistical software* 76 (2017). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9788645/.

[10] Toby Cathcart Burn, C.-H. Luke Ong, and Steven J. Ramsay. "Higher-order constrained horn clauses for verification". In: *Proc. ACM Program. Lang.* 2.POPL (2018), 11:1–11:28. DOI: 10.1145/3158099. URL: https://doi.org/10.1145/3158099.

[11] Aleksandar Chakarov and Sriram Sankaranarayanan. "Probabilistic Program Analysis with Martingales". In: *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*. 2013, pp. 511–526. URL: https://link.springer.com/chapter/10.1007/978-3-642-39799-8_34.

[12] Krishnendu Chatterjee, Hongfei Fu, Petr Novotný, and Rouzbeh Hasheminezhad. "Algorithmic Analysis of Qualitative and Quantitative Termination Problems for Affine Probabilistic Programs". In: *ACM Trans. Program. Lang. Syst.* 40.2 (2018), 7:1–7:45. URL: https://dl.acm.org/doi/abs/10.1145/3174800.

[13] Krishnendu Chatterjee, Petr Novotný, and Dorde Zikelic. "Stochastic invariants for probabilistic termination". In: *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*. 2017, pp. 145–160. URL: https://dl.acm.org/doi/abs/10.1145/3009837.3009873.

[14] Jianhui Chen and Fei He. "Proving almost sure termination by omega-regular decomposition". In: *Proceedings of the 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2020, London, UK, June 15-20, 2020*. 2020, pp. 869–882. URL: https://dl.acm.org/doi/abs/10.1145/3385412.3386002.

[15] Roy L Crole. *Categories for types*. Cambridge University Press, 1993.

[16] Ryan Culpepper and Andrew Cobb. "Contextual Equivalence for Probabilistic Programs with Continuous Random Variables and Scoring". In: *Programming Languages and Systems - 26th European Symposium on Programming, ESOP 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*. Ed. by Hongseok Yang. Vol. 10201. Lecture Notes in Computer Science. Springer, 2017, pp. 368–392. DOI: 10.1007/978-3-

662-54434-1_14. URL: https://doi.org/10.1007/978-3-662-54434-1_14.

[17] Marco F Cusumano-Towner, Feras A Saad, Alexander K Lew, and Vikash K Mansinghka. "Gen: a general-purpose probabilistic programming system with programmable inference". In: *Proceedings of the 40th acm sigplan conference on programming language design and implementation*. 2019, pp. 221–236. URL: https://dl.acm.org/doi/abs/10.1145/3314221.3314642.

[18] Ugo Dal Lago and Charles Grellois. "Probabilistic Termination by Monadic Affine Sized Typing". In: *ACM Trans. Program. Lang. Syst.* 41.2 (2019), 10:1–10:65. URL: https://dl.acm.org/doi/abs/10.1145/3293605.

[19] Rick Durrett. *Probability Theory and Examples*. 5th. Cambridge Series in Statistical and Probabilistic Mathematics. CUP, 2019. URL: https://services.math.duke.edu/~rtd/PTE/PTE5_011119.pdf.

[20] Thomas Ehrhard, Michele Pagani, and Christine Tasson. "Measurable Cones and Stable, Measurable Functions: A Model for Probabilistic Higher-Order Programming". In: *Proc. ACM Program. Lang.* 2.POPL (Dec. 2017). DOI: 10.1145/3158147. URL: https://doi.org/10.1145/3158147.

[21] Claudia Faggian and Simona Ronchi Della Rocca. "Lambda calculus and probabilistic computation". In: *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE. 2019, pp. 1–13. URL: https://arxiv.org/pdf/1901.02853.

[22] Luis María Ferrer Fioriti and Holger Hermanns. "Probabilistic termination: Soundness, completeness, and compositionality". In: *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. 2015, pp. 489–501. URL: https://dl.acm.org/doi/pdf/10.1145/2676726.2677001.

[23] Hong Ge, Kai Xu, and Zoubin Ghahramani. "Turing: A Language for Flexible Probabilistic Inference". In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. Ed. by Amos Storkey and Fernando Perez-Cruz. Vol. 84. Proceedings of Machine Learning Research. PMLR, Sept. 2018, pp. 1682–1690. URL: https://proceedings.mlr.press/v84/ge18b.html.

[24] Michele Giry. "A categorical approach to probability theory". In: *Categorical Aspects of Topology and Analysis: Proceedings of an International Conference Held at Carleton University, Ottawa, August 11–15, 1981*. Springer. 2006, pp. 68–85.

[25] Noah D. Goodman, Vikash K. Mansinghka, Daniel M. Roy, Keith Bonawitz, and Joshua B. Tenenbaum. "Church: a language for generative models". In: *UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence, Helsinki, Finland, July 9-12, 2008*. 2008, pp. 220–229. arXiv: 1206.3255.

[26] Jean Goubault-Larrecq and Daniele Varacca. “Continuous random variables”. In: *2011 IEEE 26th Annual Symposium on Logic in Computer Science*. IEEE. 2011, pp. 97–106. URL: http://www.lsv.fr/Publis/PAPERS/PDF/GLV-lics2011.pdf.

[27] Philippe de Groote. “Strong Normalization in a Non-Deterministic Typed Lambda-Calculus”. In: *Logical Foundations of Computer Science, Third International Symposium, LFCS'94, St. Petersburg, Russia, July 11-14, 1994, Proceedings*. Ed. by Anil Nerode and Yuri V. Matiyasevich. Vol. 813. Lecture Notes in Computer Science. Springer, 1994, pp. 142–152. DOI: 10.1007/3-540-58140-5_15. URL: https://doi.org/10.1007/3-540-58140-5%5C_15.

[28] Mingzhang Huang, Hongfei Fu, and Krishnendu Chatterjee. “New Approaches for Almost-Sure Termination of Probabilistic Programs”. In: *Programming Languages and Systems - 16th Asian Symposium, APLAS 2018, Wellington, New Zealand, December 2-6, 2018, Proceedings*. 2018, pp. 181–201. arXiv: 1806.06683. URL: https://link.springer.com/chapter/10.1007/978-3-030-02768-1_11.

[29] Mingzhang Huang, Hongfei Fu, Krishnendu Chatterjee, and Amir Kafshdar Goharshady. “Modular verification for almost sure termination of probabilistic programs”. In: *Proc. ACM Program. Lang.* 3.OOPSLA (2019), 129:1–129:29. URL: https://dl.acm.org/doi/abs/10.1145/3360555.

[30] Benjamin Lucien Kaminski and Joost-Pieter Katoen. “On the Hardness of Almost-Sure Termination”. In: *Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part I*. 2015, pp. 307–318. arXiv: 1506.01930. URL: https://link.springer.com/chapter/10.1007/978-3-662-48057-1_24.

[31] Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Federico Olmedo. “Weakest precondition reasoning for expected runtimes of randomized algorithms”. In: *Journal of the ACM (JACM)* 65.5 (2018), pp. 1–68. URL: https://dl.acm.org/doi/abs/10.1145/3208102.

[32] Andrew Kenyon-Roberts and C-H Luke Ong. “Supermartingales, ranking functions and probabilistic lambda calculus”. In: *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE. 2021, pp. 1–13. arXiv: 2102.11164. URL: https://ieeexplore.ieee.org/abstract/document/9470550.

[33] Naoki Kobayashi, Ugo Dal Lago, and Charles Grellois. “On the Termination Problem for Probabilistic Higher-Order Recursive Programs”. In: *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*. 2019, pp. 1–14. URL: https://lmcs.episciences.org/6817.

[34] Dexter Kozen. “Semantics of Probabilistic Programs”. In: *J. Comput. Syst. Sci.* 22.3 (1981), pp. 328–350. URL: https://www.sciencedirect.com/science/article/pii/0022000081900362/pdf.

[35] Wonyeol Lee, Hangyeol Yu, and Hongseok Yang. "Reparameterization Gradient for Non-differentiable Models". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*. 2018, pp. 5558–5568. URL: https://proceedings.neurips.cc/paper_files/paper/2018/hash/b096577e264d1ebd6b41041f392eec23-Abstract.html.

[36] Carol Mak, C.-H. Luke Ong, Hugo Paquet, and Dominik Wagner. "Densities of almost surely terminating probabilistic programs are differentiable almost everywhere". In: *ESOP 2021*. https://arxiv.org/abs/2004.03924. 2021. arXiv: 2004.03924.

[37] Vikash K. Mansinghka, Daniel Selsam, and Yura N. Perov. "Venture: a higher-order probabilistic programming platform with programmable inference". In: *CoRR* abs/1404.0099 (2014). arXiv: 1404.0099. URL: http://arxiv.org/abs/1404.0099.

[38] Dominic Marchand. "Classical monte carlo and the metropolis algorithm: Revisiting the 2d ising model". In: *University of British Columbia, Department of Physics and Astronomy* (2005). URL: https://phas.ubc.ca/~berciu/TEACHING/PHYS503/PROJECTS/05_dominic.pdf.

[39] Annabelle McIver and Carroll Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems*. Monographs in Computer Science. Springer, 2005. ISBN: 978-0-387-40115-7.

[40] Annabelle McIver, Carroll Morgan, Benjamin Lucien Kaminski, and Joost-Pieter Katoen. "A new proof rule for almost-sure termination". In: *Proceedings of the ACM on Programming Languages* 2.POPL (2017), pp. 1–28. URL: https://dl.acm.org/doi/pdf/10.1145/3158121.

[41] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. "Equation of State Calculations by Fast Computing Machines". In: *The Journal of Chemical Physics* 21.6 (Dec. 2004), pp. 1087–1092. ISSN: 0021-9606. DOI: 10.1063/1.1699114. eprint: https://pubs.aip.org/aip/jcp/article-pdf/21/6/1087/8115285/1087_1_online.pdf. URL: https://doi.org/10.1063/1.1699114.

[42] Eugenio Moggi. *An abstract view of programming languages*. University of Edinburgh, Department of Computer Science, Laboratory for ..., 1989. URL: http://www.ics.uci.edu/~jajones/INF102-S18/readings/09_Moggi.pdf.

[43] Akihiko Nishimura, David B Dunson, and Jianfeng Lu. "Discontinuous Hamiltonian Monte Carlo for discrete parameters and discontinuous likelihoods". In: *Biometrika* 107.2 (2020), pp. 365–380. URL: https://academic.oup.com/biomet/article/107/2/365/5799014.

[44] Federico Olmedo, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. "Reasoning about recursive probabilistic programs". In: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*. 2016, pp. 672–681. URL: https://dl.acm.org/doi/abs/10.1145/2933575.2935317.

[45] C.-H. Luke Ong and Dominik Wagner. "HoCHC: A Refutationally Complete and Semantically Invariant System of Higher-order Logic Modulo Theories". In: *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*. IEEE, 2019, pp. 1–14. DOI: 10.1109/LICS.2019.8785784. URL: https://doi.org/10.1109/LICS.2019.8785784.

[46] Thomas Rainforth. "Automating inference, learning, and design using probabilistic programming". PhD thesis. University of Oxford, 2017. URL: https://ora.ox.ac.uk/objects/uuid:e276f3b4-ff1d-44bf-9d67-013f68ce81f0.

[47] Christian P Robert and George Casella. *Monte Carlo statistical methods*. Springer, 2004. DOI: 10.1007/978-1-4757-4145-2. URL: https://archive.org/details/springer_10.1007-978-1-4757-4145-2/page/n497/mode/2up.

[48] Sam Staton, Hongseok Yang, Frank Wood, Chris Heunen, and Ohad Kammar. "Semantics for Probabilistic Programming: Higher-Order Functions, Continuous Distributions, and Soft Constraints". In: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 525–534. ISBN: 9781450343916. DOI: 10.1145/2933575.2935313. URL: https://doi.org/10.1145/2933575.2935313.

[49] Masako Takahashi. "Parallel reductions in lambda-calculus". In: *Information and Computation* 118 (1995), pp. 120–127. URL: https://doi.org/10.1006/inco.1995.1057.

[50] David Tolpin, Jan-Willem van de Meent, and Frank D. Wood. "Probabilistic Programming in Anglican". In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part III*. 2015, pp. 308–311. URL: https://link.springer.com/chapter/10.1007/978-3-319-23461-8_36.

[51] Dustin Tran, Alp Kucukelbir, Adji B Dieng, Maja Rudolph, Dawen Liang, and David M Blei. "Edward: A library for probabilistic modeling, inference, and criticism". In: *arXiv preprint arXiv:1610.09787* (2016). URL: https://arxiv.org/abs/1610.09787.

[52] Matthijs Vákár, Ohad Kammar, and Sam Staton. "A domain theory for statistical probabilistic programming". In: *Proceedings of the ACM on Programming Languages* 3.POPL (2019), pp. 1–29. URL: https://dl.acm.org/doi/pdf/10.1145/3290349.

[53] Eric W. Weisstein. *Pólya's Random Walk Constants.* From MathWorld–A Wolfram Web Resource. URL: https://mathworld.wolfram.com/PolyasRandomWalkConstants.html (visited on 09/09/2024).

[54] David Williams. *Probability with Martingales*. Cambridge Mathematical Textbooks. Cambridge University Press, 1999.

[55] David Wingate, Andreas Stuhlmueller, and Noah Goodman. "Lightweight Implementations of Probabilistic Programming Languages Via Transformational Compilation". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Apr. 2011, pp. 770–778. URL: https://proceedings.mlr.press/v15/wingate11a.html.

[56] Yuan Zhou, Bradley J. Gram-Hansen, Tobias Kohn, Tom Rainforth, Hongseok Yang, and Frank Wood. "LF-PPL: A Low-Level First Order Probabilistic Programming Language for Non-Differentiable Models". In: *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*. 2019, pp. 148–157. URL: https://proceedings.mlr.press/v89/zhou19b.