Tropical Mathematics and the Lambda-Calculus II: Tropical Geometry of Probabilistic Programming Languages

DAVIDE BARBAROSSA, University of Bath, United Kingdom
PAOLO PISTONE, Université Claude Bernard Lyon 1, France

In the last few years there has been a growing interest towards methods for statistical inference and learning based on computational geometry and, notably, tropical geometry, that is, the study of algebraic varieties over the min-plus semiring. At the same time, recent work has demonstrated the possibility of interpreting higher-order probabilistic programming languages in the framework of tropical mathematics, by exploiting algebraic and categorical tools coming from the semantics of linear logic. In this work we combine these two worlds, showing that tools and ideas from tropical geometry can be used to perform statistical inference over higher-order probabilistic programs. Notably, we first show that each such program can be associated with a *degree* and a *n-dimensional polyhedron* that encode its *most likely* runs. Then, we use these tools in order to design an intersection type system that estimates most likely runs in a compositional and efficient way.

CCS Concepts: • **Theory of computation** → **Lambda calculus**; *Probabilistic computation*; Categorical semantics; • **Mathematics of computing** → Probabilistic inference problems; Linear programming.

Additional Key Words and Phrases: Probabilistic Lambda-Calculus, Tropical geometry, Relational Semantics, Linear Optimisation

ACM Reference Format:
Davide Barbarossa and Paolo Pistone. 2026. Tropical Mathematics and the Lambda-Calculus II: Tropical Geometry of Probabilistic Programming Languages. *Proc. ACM Program. Lang.* 10, POPL, Article 33 (January 2026), 30 pages. https://doi.org/10.1145/3776675

1 Introduction

From Probabilistic Models to Probabilistic Programming Languages. Probabilistic models play a fundamental role in many areas of computer science, such as, just to name a few, machine learning, bioinformatics, speech recognition, robotics and computer vision. For many common problems (like, for example, identifying the regions of DNA that code for some specific protein or tracking the location of a vehicle from the data produced by possibly faulty sensors) finding an exact solution requires to enumerate an impossibly large list of possibilities; by contrast, a probabilistic model may allow one to focus only on those (usually, much less) possibilities which are *more likely* to occur, under normal circumstances. In this respect, models like Bayesian Networks (BN) or Hidden Markov Models (HMM) provide an extremely well-studied and modular approach making the representation of (our current knowledge of) the system under study independent from the inference algorithms that can be applied in order to answer specific questions about it.

While probabilistic models provide a description of a system under conditions of uncertain knowledge, probabilistic programming languages (PPL) provide ways to *specify such models via programs*: the execution of the program produces the model, in the sense that the probabilistic reductions of the program describe the trajectories of the model. The inference tasks associated with

Authors' Contact Information: Davide Barbarossa, University of Bath, Bath, United Kingdom, db2437@bath.ac.uk; Paolo Pistone, Université Claude Bernard Lyon 1, Lyon, France, paolo.pistone@ens-lyon.fr.

ACM 2475-1421/2026/1-ART33
https://doi.org/10.1145/3776675

a model are thus naturally related to the study of the probabilistic execution of the corresponding program (e.g., what is the probability that the program will return True? Or that it terminates?).

The goal of PPL like e.g. Church or Anglican (to name two languages that rely on the LISP architecture), is to streamline the activity of probabilistic modeling, by exploiting features of programming languages like compositionality and higher-order functions. This becomes particularly relevant when considering possibly *infinitary* models that take temporality into account, like e.g. template-based Bayesian Networks, which can be conveniently described in higher-order functional languages, cf. [23]. The study of PPLs has recently seen a flourishing of research directions, going from more foundational/category-theoretic approaches [15, 30, 31, 48], to others more oriented towards inference algorithms and their efficiency like [23].

The Tropical Geometry of Probabilistic Models . The application of methods from computational algebraic geometry in areas like machine learning and statistical inference is well investigated. Among such methods a growing literature has explored the application of ideas from tropical geometry to the study of deep neural networks and graphical probabilistic models [12, 38, 45, 46, 53].

Tropical geometry is the study of polynomials and algebraic varieties defined over the min-plus (or the max-plus) semiring: a tropical polynomial is obtained from a standard polynomial by replacing + with min and × with +. Several computationally difficult problems expressible in the language of algebraic geometry admit a tropical counterpart which is purely combinatorial and, in some cases, tractable in an effective way. For example, while finding the roots of a polynomial is a paradigmatic undecidable problem, tropical roots can be computed in linear time and used to approximate the actual roots of the polynomial [43, 44].

Concerning probabilistic models, it has been observed that several inference algorithms based on convex optimization, like the *Viterbi algorithm*, have a “tropical flavor” [49]. Usually, graphical probabilistic models express the probability of an event as a polynomial p_E, which intuitively adds up the probabilities p_i of the (so many) mutually independent situations i that might produce E. A typical problem, for instance when computing Bayesian posteriors, is to know, given the knowledge that the event E occurred, which situations i are the *most likely* to have produced E. While comparing *all* the situations i is certainly not feasible, works like [45, 46] have shown that the study of the *Newton polytope* of the tropical polynomial associated to p_E provides an efficient method to select the potential solutions i.

The Tropical Geometry of PPLs. While ideas from tropical mathematics have been applied successfully to probabilistic models like HMM or BN, in this paper we are concerned with the following, broader, question: would it be possible to exploit the computational toolkit of tropical geometry as an inference engine for the programs of some higher-order programming language, and, as a consequence, for the large class of probabilistic models that this language may represent?

Our approach relies on a recent line of work [6] that has demonstrated the possibility of interpreting higher-order probabilistic languages within the setting of tropical mathematics. This interpretation relies on the *weighted relational semantics* (WRS) [35], a well-studied class of models of PCF and related languages that is parametric on the choice of a continuous semiring Q. The WRS arises from the literature on linear logic and has been at the heart of numerous investigations and results about programming languages with non-determinism, probabilities or even quantum primitives [13, 20, 21, 33, 34, 47].

When Q is the min-plus semiring (on $\mathbb{R}_{\geq 0} \cup \{\infty\}$ or $\mathbb{N} \cup \{\infty\}$), one obtains a semantics of probabilistic PCF (pPCF) that has been shown to capture the *most likely* behavior of a program [6, 35]. For example, of the many ways in which a program M of type Bool may reduce to True, only those which have the *highest* probability to occur are represented in the semantics. Therefore, the tropical WRS intuitively “solves” the inference task of selecting the most likely execution paths of a pPCF- program, and, along with them, the most likely trajectories in the corresponding probabilistic model. However, as this semantics is neither finitary nor computable in general, the obvious question is: to which extent could such "solutions" be produced in an effective way?

This paper provides an answer to this question, obtained in two steps: first, we exploit the tropical setting to show that the problem of describing a *most likely* trajectory for a pPCF program can be reduced to a search problem within some *finite* (albeit possibly very large) space. This fact is non-trivial, since, unlike standard BN, the probabilistic models corresponding to pPCF programs may have an *infinite* trajectory space. While this fact is established via a non-constructive argument, by combining ideas from tropical geometry and programming language theory, we design a recursive procedure that, given a higher-order program of ground type, explores the trajectories of the corresponding infinitary model in a compositional way. This procedure is proved to converge to a solution of the following two standard inference tasks (cf. Theorem 7.12):
(I1) select the most likely trajectories of the model,
(I2) for a fixed trajectory θ, identify the values of the probabilistic parameters making θ most likely.
The convergence of this procedure is itself non-recursive, as the underlying problem is undecidable, but the partial solutions provided at each stage of the computation are correct for a restricted set of trajectories approximating the behaviour of the program. Furthermore, we show that in several interesting cases (e.g. when considering an *affine* higher-order program) our method does indeed produce a correct answer, and it does so *efficiently*: while the number of possible trajectories is exponential in the size of the program, the most likely ones are found in polynomial time.

The Tropical Degree. As we said, our first result shows that the trajectory space for a pPCF program can be reduced to a finite one. This is obtained by exploiting a representation of such programs via *tropical polynomials*. Indeed, graphical probabilistic models like BN and HMM are often presented algebraically via families of polynomials in a given set of parameters. The WRS extends this presentation to pPCF programs, except that, due to their higher-order nature, programs correspond to *power series*, not just polynomials, in the parameters. Intuitively, if a finite sum of monomials is enough to add up finitely many independent trajectories that may lead to the same result, an infinite sum is required when the number of trajectories is infinite. When considering the interpretation of pPCF over the tropical semiring, these power series are turned into *tropical analytic functions* (*taf*, for short), that is, continuous functions that can be written as $f(x) = \inf_{i\in I} \phi_i(x)$, i.e. an inf of possibly infinitely many linear maps $\phi_i(x)$. While tropical polynomials and their geometric properties are very well-studied, the literature on taf is still scarce [6, 43].

Our central result here, Corollary 5.5 from Section 5, is that any program M of ground type, say Bool, is represented by a taf that is in fact a tropical polynomial (in other words, that f can be written as $f(x) = \min_{i\in J\subset I} \phi_i$ for some *finite* subset $J \subset I$). This follows from a general result (cf. Proposition 5.4) about tafs with discrete coefficients. The meaning of this result is that, among the many trajectories that may lead to the same event, only a *finite* portion has a chance of occurring "most likely". Intuitively, if we think of M as describing a probabilistic model that iterates a given procedure until it produces a given result o (a typical *Las Vegas* algorithm), then the probability that o was obtained after *no less* than n iterations will reach its maximum after a finite number D of steps: the greater the number of iterations in a reduction of M, the lower the probability that this reduction may actually have occurred. This number D is what we call *tropical degree of* M, and coincides with the (finite) degree of the tropical polynomial that captures the behavior of M.

Statistical Inference via Intersection Types and the Newton Polytope. Even once we have reduced the trajectories of our program to a finite set, this set may still be *too large* to be enumerated in practice. The second step is thus to show that the trajectory space can be further reduced to one of a more *tractable* size, in order to have a hope to access it in a feasible computational way. Our first result here is to associate a program M with a n-dimensional polyhedron $NP_{\min}(M)$, a variant of the standard Newton polytope called the *minimal Newton polytope*, that encodes the most likely runs of M in a more optimized way (cf. Theorem 6.6). Then, we design a type system $\mathbf{P}_{\text{trop}}$ that captures the most likely executions of the program compositionally in M (cf. Theorem 7.7), and we show that this system can be used to design a method that reconstructs the polytope $NP_{\min}(M)$ in a recursive way. As anticipated before, while the termination condition for this method is non-recursive, we discuss the situations in which it can be used to provide correct solutions efficiently. The system $\mathbf{P}_{\text{trop}}$ uses *non-idempotent intersection types* [4, 11, 18, 21], a well-studied technique to capture the quantitative behavior of higher-order programs. In $\mathbf{P}_{\text{trop}}$ a *single* type derivation may explore a *plurality* of possible executions of the program, at the same time *selecting* a small enough set of most likely trajectories; to do that it makes use of an algorithm to *compose* graphical models inspired by the Viterbi algorithm for HMM and relying on the computation of the minimal Newton polytopes of the underlying polynomials (cf. Theorem 6.9).

Outline of the Paper. In Section 2 we introduce the language $\mathrm{PCF}\langle\vec{X}\rangle$, a parameterized version of probabilistic PCF inspired from [21, 35] that will serve as our base language, and we illustrate how its programs describe potentially infinite discrete probabilistic models. Section 3 contains an informal overview on the problem of finding most likely explanations for $\mathrm{PCF}\langle\vec{X}\rangle$ programs and of our main ideas to overcome them. In Section 4 we introduce a parameterized version of the weighted relational semantics, yieding a model of $\mathrm{PCF}\langle\vec{X}\rangle$ in terms of formal power series. Section 5 contains our first result, that is, that first-order type programs of $\mathrm{PCF}\langle\vec{X}\rangle$ have a finite tropical degree. Section 6 contains our results on the minimal Newton polytope and our variant of the Viterbi algorithm to compute the tropical product of polynomials. In Section 7 we introduce the intersection type system $\mathbf{P}_{\text{trop}}$ and the method, relying on it, to enumerate the most likely runs. Finally, in Section 8 we discuss related work and future directions.

2 Parametric PCF: Specifying Models via Higher-Order Programs

In this section we introduce $\mathrm{PCF}\langle\vec{X}\rangle$, a variant of probabilistic PCF [21, 35], that will serve as our base language in the rest of the paper. We then illustrate in which sense the programs of $\mathrm{PCF}\langle\vec{X}\rangle$ describe a class of discrete (infinitary) probabilistic models that we will explore in the next sections.

2.1 The language $\mathrm{PCF}\langle\vec{X}\rangle$

The language $\mathrm{PCF}\langle\vec{X}\rangle$ differs from standard probabilistic PCF [21, 35] in that real probabilities are replaced by a finite number of *parameters* $X_1, \dots, X_n$. For instance, a probabilistic term $M \oplus_p N$, corresponding to a choice yielding M with probability p and N with probability $1-p$, is replaced in $\mathrm{PCF}\langle\vec{X}\rangle$ by a parametric term $M \oplus_X N$, intuitively corresponding to a choice between M and N depending on some *unknown* parameter X. A similar language with probabilistic parameters was used already in [21] to establish the full abstraction of the semantics of *probabilistic coherent spaces*.

The reason for considering, in this work, a language where explicit probabilities are replaced by parameters is twofold. Firstly, in probabilistic models like BN or HMM it is standard to take the underlying basic probabilities as parameters of the model, for instance when considering problems like that of computing the *maximum likelihood* of an event. In the next section we will make more precise the role of parameters in the inference tasks we consider in this work.

The second reason is that we will explore, *in parallel*, an interpretation of $\mathrm{PCF}\langle\vec{X}\rangle$ that associates parameters with actual probabilities $q \in [0,1]$ as well as a second interpretation that associates the same parameters with *negative log-probabilities* $z = -\ln q \in \mathbb{R}^{\infty}_{\geq 0}$. Indeed, the methods based on

$$\frac{}{\Gamma \vdash 0 : \mathbf{N}} \qquad \frac{\Gamma \vdash M : \mathbf{N}}{\Gamma \vdash \mathsf{succ}\, M, \mathsf{pred}\, M : \mathbf{N}} \qquad \frac{}{\Gamma \vdash 0, 1 : \mathrm{Bool}} \qquad \frac{\Gamma \vdash M : \mathrm{Bool}}{\Gamma \vdash M : \mathbf{N}}$$

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

$$\frac{\Gamma \vdash M : \mathbf{N} \qquad \Gamma \vdash N, P : A}{\Gamma \vdash \mathsf{ifz}(M, N, P) : A} \qquad \frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : A}{\Gamma \vdash M \oplus_X N : A} \qquad \frac{\Gamma \vdash M : A \to A}{\Gamma \vdash \mathrm{Y}M : A}$$

(a) Typing rules.

$$\begin{array}{cccc}
\mathsf{ifz}(0, M, N) \xrightarrow{[]} M & (\lambda x.M)N \xrightarrow{[]} M[N/x] & \mathsf{pred}\ 0 \xrightarrow{[]} 0 & M \oplus_{X_i} N \xrightarrow{X_i} M \\
\mathsf{ifz}(\mathsf{succ}\ \mathsf{n}, M, N) \xrightarrow{[]} N & \mathrm{Y}M \xrightarrow{[]} M(\mathrm{Y}M) & \mathsf{pred}\ (\mathsf{succ}\ M) \xrightarrow{[]} M & M \oplus_{X_i} N \xrightarrow{\overline{X_i}} N \\
\mathsf{ifz}(M, P, Q) \xrightarrow{\mu} \mathsf{ifz}(N, P, Q) & MP \xrightarrow{\mu} NP & \mathsf{pred}\ M \xrightarrow{\mu} \mathsf{pred}\ N & \mathsf{succ}\ M \xrightarrow{\mu} \mathsf{succ}\ N
\end{array}$$

(b) Parametric reduction rules. In the last line, we suppose $M \xrightarrow{\mu} N$.

Fig. 1. Rules of PCF$\langle \vec{X} \rangle$.

tropical geometry that we develop exploit the latter, more combinatorial, viewpoint as a means to *gain knowledge* about the former.

Definition 2.1. Let $X_1, \ldots, X_n$ be n distinct formal variables. The terms of PCF$\langle \vec{X} \rangle$ are defined by the following grammar (and quotiented by usual α-equivalence):

$$M ::= \ 0 \mid \mathsf{succ}\ M \mid \mathsf{pred}\ M \mid \mathsf{ifz}(M, M, M) \mid x \mid \lambda x.M \mid MM \mid \mathrm{Y}M \mid M \oplus_{X_i} M$$

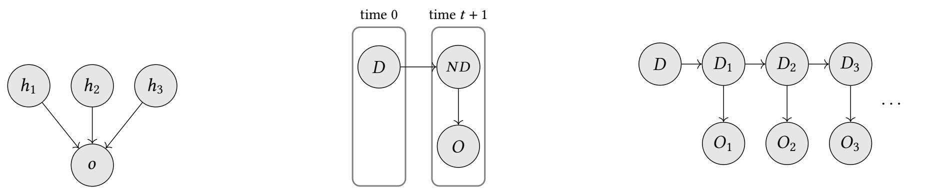
We let $\mathsf{n} := \mathsf{succ}^n(0)$. The types of PCF$\langle \vec{X} \rangle$ are defined by $A ::= \mathrm{Bool} \mid \mathbf{N} \mid A \to A$. The typing rules are presented in Fig. 1a (where contexts are finitely many variable declarations. Also, observe that we overload 0 and 1 as being both Booleans and integers).

For any set Σ, let $!\Sigma$ indicate the set of finite multisets over Σ. We indicate a multiset $\mu \in !\Sigma$ as a formal monomial $\prod_{a \in \Sigma} a^{\mu(a)}$. The operational semantics is given by a reduction relation $M \overset{\mu}{\twoheadrightarrow} N$, where $\mu \in !\{X_1, \overline{X_1}, \ldots, X_n, \overline{X_n}\}$, defined by the relexive and transitive closure of the rules in Fig. 1b, which include standard PCF weak head CbN reductions, as well as parametric reductions for the choice operator. For the reflexive closure, we set $M \overset{[]}{\twoheadrightarrow} M$; for the transitive closure, we set $M \overset{\mu \cdot \nu}{\twoheadrightarrow} P$ whenever $M \overset{\mu}{\twoheadrightarrow} N$ and $N \overset{\nu}{\twoheadrightarrow} P$.

Example 2.2. For $M_1 = (1 \oplus_X 0) \oplus_X ((1 \oplus_X 0) \oplus_X (0 \oplus_X 1))$ there are three reductions $M \overset{\mu}{\twoheadrightarrow} 0$, that give $\mu_1 = X\overline{X}, \mu_2 = \mu_3 = X\overline{X}^2$ and three reductions $M \overset{\nu}{\twoheadrightarrow} 1$, with $\nu_1 = X^2, \nu_2 = X^2\overline{X}, \nu_3 = \overline{X}^3$.

Remark 2.1 (relation with probabilistic PCF). *By reading the parameters $X_1, \ldots, X_n$ as reals $q_1, \ldots, q_n \in [0, 1]$ the typing and reduction rules of* PCF$\langle \vec{X} \rangle$ *are just rules for a standard PCF with biased choice operators $M \oplus_{q_i} N$ (where instead of adding to a multiset, we take the product in $[0, 1]$). In this way, standard properties like e.g. subject reduction are easily deduced from those of pPCF.*

Remark 2.2. *We could have chosen to label reductions with finite* words *over $X_i, \overline{X_i}$ instead of multisets, so that each label μ in $M \overset{\mu}{\twoheadrightarrow} N$ univocally determines one reduction of M. We chose multisets because this is more natural in view of the formal manipulations discussed in the next sections. We will quickly go back at the possibility of using words instead at the end of Section 7.*



(a) Bayesian Network for Example 2.2.

(b) Dynamical Bayesian Network for Example 2.3

(c) Unrolled Bayesian Network

Fig. 2. Examples of Bayesian Networks.

2.2 $\mathrm{PCF}\langle\vec{X}\rangle$ and Graphical Probabilistic Models

Consider again the term M_1 from Example 2.2. What kind of probabilistic model does M_1 specify? A possible illustration is provided in Fig. 2a: a simple Hidden Markov Model model with: one *observable* variable $o \in \{0,1\}$, corresponding to the result of the execution of M_1; three *hidden* independent random variables $h_1, h_2, h_3 \in \{0,1\}$ (read: left, right), each corresponding to the choice made at the i-th step of a reduction of M_1; a conditional probability $\mathbb{P}(o = 1 \mid h_1, h_2, h_3)$ (and similarly for $o = 0$), modeling the probability of M_1 reducing to 1 following the reduction specified by the h's. Notice that we choose three variables h's, while sometimes M_1 only performs two choices, for example in $M \overset{X^2}{\twoheadrightarrow} 1$. To solve this mismatch, we can declare that the previous reduction of M_1 corresponds to both $(h_1,h_2,h_3) = (0,0,0)$ and $(h_1,h_2,h_3) = (0,0,1)$ in the HMM. In this way, we have, parametrically in $X, \overline{X}$, that $\mathbb{P}(o = 1) = (X^3 + X^2\overline{X}) + X^2\overline{X} + \overline{X}^3$. For each probability assignment $X := p, \overline{X} := 1-p$ of the parameters, the above probability now becomes the correct

$$\mathbb{P}(o = 1) = p^2 + p^2(1-p) + (1-p)^3. \tag{1}$$

In fact, it is well-known that graphical models like HMM or BN can be encoded as terms of some PCF-like language [23, 26], and the overall goal of this section is to suggest that PCF programs can be thought as specifying complex HMM of some kind. However, due to their higher-order nature, as well as the possibility of defining functions recursively via the fixpoint Y, a general program of $\mathrm{PCF}\langle\vec{X}\rangle$ need *not* describe a finite probabilistic models like those illustrated so far.

Example 2.3. Let $M_2 = \mathrm{Y}B(ND) : \mathrm{Bool}$, where

$$B := \lambda f x.\mathrm{ifz}(x, \mathrm{ifz}(O0, f(N0), 1), \mathrm{ifz}(O1, f(N1), 1)) : (\mathrm{Bool} \to \mathrm{Bool}) \to \mathrm{Bool} \to \mathrm{Bool},$$

$D = 0 \oplus_{X_0} 1 : \mathrm{Bool}$ represents an initial Distribution of Booleans, $N = \lambda x.\mathrm{ifz}(x, 0 \oplus_{X_1} 1, 0 \oplus_{X_2} 1) : \mathrm{Bool} \to \mathrm{Bool}$ a probabilistic protocol to turn a distribution into a New one, and $O = \lambda x.\mathrm{ifz}(x, 0 \oplus_{X_3} 1, 0 \oplus_{X_4} 1) : \mathrm{Bool} \to \mathrm{Bool}$ another probabilistic protocol to Observe a Boolean value. The behaviour of M_2 can be recast in pseudocode as follows:

```
$d = \mathrm{sample}(D)$ ; while(true) { $d = \mathrm{sample}(Nd)$ ; $o = \mathrm{sample}(Od)$ ; if($o == 1$) {return 1} }.
```

We see that M_2 describes thus a *dynamic* Bayesian Network (cf. [32], ch. 6) as the one illustrated in Figg. 2b and 2c: a potentially infinite DAG constructed following an iterative pattern. Notice that the number of hidden and observed variables is potentially infinite: each iteration produces a new hidden variable D_i (corresponding to the value produced by applying N i times to D) and a new observation O_i. By contrast, the number of parameters of the model is finite, as it consists of the parameters $X_0 - X_4$ in the terms D, N, O.

In the example above, in order to reconstruct the model described by the program, an essential ingredient is to be able to identify a certain *pattern* that is repeated over and over, so as to provide a compact graphical representation of how the hidden and observed variables relate to each other. Obviously, extracting such patterns may be very hard, or even undecidable, to do for an arbitrary PCF$\langle\vec{X}\rangle$ program. Another solution would be to consider a somehow *flat* model with one observed variable o, corresponding to the final result of the execution (if any), conditioned on *infinitely many* hidden variables $h_1, h_2, h_3, \dots$, corresponding to the unbounded number of choices made during a terminating reduction (which may be arbitrarily long).

However, in a such a model the dependency of o on any variable h_i may be rather difficult to track explicitly, since *infinitely many* and *arbitrarily long* different reductions may lead to the same result. As we'll see, this dependency is in general not expressed by a polynomial as in standard HMM or BN, but by a *power series* in the parameters. At the same time, the flat model provides no insight on how such complex dependencies might be decomposed following the structure of the program. In other words, it is not *compositional.*

Reconstructing the probabilistic model underlying a program of PCF$\langle\vec{X}\rangle$ is indeed tantamount to reconstructing the semantics of the program. Still, as we'll see, linear logic and programming language theory provide us with precisely the good methods to, first, design a probabilistic semantics capturing the relevant power series and, second, design a syntactic method (i.e. a type system) to fully approximate the semantics of the program. This is the approach we describe through Sections 4-7. But before delving into that, let us look more closely at the inference tasks that we consider.

3 Can We Do Statistical Inference over PCF Programs?

The overall goal of this work is to demonstrate the possibility of inferring the most likely trajectories in the models specified by higher-order probabilistic programs. In this section we provide an informal overview on the difficulties lying ahead of this goal, and a first intuitive illustration of our two main ideas to overcome them: the notion of *tropical degree* (cf. Section 5), and an adaptation of the *Viterbi algorithm* from HMM to higher-order programs (cf. Sections 6 and 7).

3.1 Most Likely Explanations and the Tropical Degree

The typical inference task for a Bayesian Network is to compute the *marginal probabilities* associated with an assignment $\vec{\sigma}$ to the observed variables. This corresponds, intuitively, to summing up the probabilities of all trajectories producing the outcomes $\vec{\sigma}$, that is, of all possible assignments $\vec{\theta}$ to the hidden variables, as in (1). In a *finite* BN the marginal probabilities can be expressed as polynomials in the parameters and can be computed via algorithms like e.g. the *sum-product* algorithm [52].

In this work we are not interested in the problem of computing marginal probabilities. Indeed, when considering probabilistic models, like HMM, with a marked distinction between hidden and observed variables, a natural question is to predict the *most likely explanation* for a given outcome: supposing that we observed that our program returned True, what are the trajectories (i.e. the values of the hidden variables) that have the most chances of having produced this result?

More precisely, we are interested in the following two inference problems:

(I1) given both the observation $\vec{\sigma}$ and the values $\vec{q}$ assigned to the parameters, compute the *maximum a posteriori probabilities*

$$\max\left\{\mathbb{P}(\vec{o}=\vec{\sigma},\vec{h}=\vec{\theta},\vec{X}=\vec{q})\;\middle|\;\vec{\theta}\text{ assignment to the hidden variables}\right\} \tag{2}$$

or, equivalently, the *minimum* a posteriori *negative log*-probabilities:

$$\min\left\{-\ln\mathbb{P}(\vec{o}=\vec{\sigma},\vec{h}=\vec{\theta},\vec{X}=-\ln\vec{q})\;\middle|\;\vec{\theta}\text{ assignment to the hidden variables}\right\} \tag{3}$$

and produce one such assignment $\vec{\theta}$ to the hidden variables that provides a *most likely* explanation for the observation $\vec{\sigma}$;

(I2) given both the observation $\vec{\sigma}$ and the hidden data $\vec{\theta}$, identify the values of the parameters $\vec{X}$ that make the assignment $\vec{\theta}$ the most likely explanation of $\vec{\sigma}$, i.e. compute the set

$$\left\{\vec{q} \in [0,1]^n \;\middle|\; \vec{\theta} = \mathrm{argmax}_{\vec{\rho}}\{\mathbb{P}(\vec{o} = \vec{\sigma}, \vec{h} = \vec{\rho}, \vec{X} = \vec{q})\}\right\} \tag{4}$$

Coming back again to our running example M_1, let us show how to compute solutions to both problems. For (I1), considering the observation $\sigma = 1$, and fixing values e.g. $X = \overline{X} = \frac{1}{2}$ for the parameters, we are led from (1) and (3) to compute

$$\min\{-\ln X^2, -\ln X^2\overline{X}, -\ln \overline{X}^3\} = \min\{2z, 2z + \overline{z}, 3\overline{z}\} = 2z = 2\ln 2, \tag{5}$$

where $z = -\ln X, \overline{z} = -\ln \overline{X}$, showing that the leftmost reduction is the most-likely to produce the outcome 1 under the parameter assignment $X, \overline{X} \mapsto \frac{1}{2}$. For (I2), for instance, we may wish to know for which values of $X, \overline{X}$ the rightmost trajectory becomes the most likely to produce the result 1. Using (5), we are led to find $z, \overline{z}$ such that $\min\{2z, 2z + \overline{z}, 3\overline{z}\} = 3\overline{z}$, yielding the condition $\overline{z} \leq \frac{2}{3}z$, that is, $X \leq \overline{X}^{\frac{2}{3}}$ (e.g. $X = \frac{1}{4}, \overline{X} = \frac{3}{4}$).

At this point the connection with tropical geometry strikes the eye: the expression $\min\{2z, 2z + \overline{z}, 3\overline{z}\}$, obtained by replacing, in (1), the outer sum by a min, is an example of a *tropical polynomial*, i.e. a polynomial with min in place of + and + in place of ×. In fact, solving problems like (I1) essentially amounts to computing marginal probabilities in a tropical setting (i.e. in which sums are replaced by mins and multiplications by sums).

For *finite* HMMs, well-known algorithms like the *Viterbi algorithm* can be used to compute, in an efficient way, solutions to problems (I1) and (I2). This algorithm can indeed be seen as a "tropical" variant of the sum-product algorithm [49]. Still, as we discussed above, we are here considering models with infinitely many hidden variables, and thus, with infinitely many trajectories. How could one solve such an infinitary optimization problem? Here is where our fundamental idea comes into play: while a PCF$\langle\vec{X}\rangle$ program may well produce infinitely many, arbitrary long, different trajectories, one might well guess that, since the probability assigned with a trajectory is obtained by multiplying the same finite number of parameters at each iteration, such probabilities should start to *decrease* after a sufficiently long number of reduction steps.

For example, consider the experiment of repeatedly tossing a coin with bias X until a head is produced. This is represented in PCF$\langle\vec{X}\rangle$ by the program below

$$M_3 = Y(\lambda x.x \oplus_X 1) : \mathrm{Bool}$$

The probability of getting the first head at iteration $n+1$ is thus $X\overline{X}^n$ and the total probability is expressed by the *power series* $\mathbb{P}(M \twoheadrightarrow 1) = \sum_{n=1}^{\infty} X\overline{X}^n$, that sums over infinitely many trajectories. At the same time, it is clear that, across all these trajectories, the most likely explanation for a head is that we obtained it at the *first* iteration, since $q > q(1-q)^n$ for all possible choice q for X. Indeed, all this can be restated as the observation that, for $z = -\ln X, \overline{z} = -\ln \overline{X} \in [0, +\infty]$, the inf of the sequence below is reached by its first element:

$$\inf_n \left\{-\ln(X\overline{X}^n)\right\} = \inf_n \left\{z + n\overline{z}\right\} = z.$$

Consider now the term M_2 from Example 2.3. We will see (cf. Example 7.10) that, in a reduction $M_2 \overset{\mu}{\twoheadrightarrow} 1$ with n calls to Y, the monomial μ has degree $2n+3$, corresponding to $2n+3$ independent probabilistic choices, and the probability of getting 1 starts to decrease after the second iteration. This implies that a reduction $M_2 \overset{\mu}{\twoheadrightarrow} 1$ of *maximum* probability can always be found among those with $\deg \mu \leq 5$. In more algebraic terms, in the power series describing the probability $\mathbb{P}(M_2 \twoheadrightarrow 1)$, all monomials of degree greater than 5 are in fact *dominated* by some monomial of degree ≤ 5. Moving to the negative lns, the inf over *all* trajectories must reduce to a *finite* min, in fact a tropical polynomial, containing only the monomials of degree ≤ 5:

$$\inf_n \left\{\text{monomials of degree } n\right\} = \min \left\{ \text{ monomials of degree } n \leq 5\right\}.$$

This value 5 is what we call the *tropical degree* of M_2, noted $\mathfrak{d}(M_2)$: it is the smallest number n for which we can find a *finite* set S of trajectories such that all trajectories in S correspond to reductions with at most n choices, and any trajectory of the program is dominated by some trajectory in S. Our first result is that *any* program M : Bool of PCF$\langle\vec{X}\rangle$ has a finite tropical degree $\mathfrak{d}(M)$ (Corollary 5.5), that is, that the most likely explanations for the results produced by M can always be found within a *finite* trajectory space.

3.2 Most Likely Explanations *Efficiently*, via the Newton Polytope

Even once the space of trajectories for an arbitrary program M : Bool has been reduced to a finite one, this space may still be *too large* to explore in practice. For instance, consider the following higher-order pPCF program

$$M_4 = (\lambda x.x \oplus_{p_1} x)(\lambda x.x \oplus_{p_2} x) \dots (\lambda x.x \oplus_{p_n} x)1,$$

where $p_1, \dots, p_n \in [0, 1]$ are fixed probabilities. Observe that M_4 always terminates on the normal form 1, and there are exactly 2^n reductions $M_4 \twoheadrightarrow 1$, each of probability $q_{\theta_1,1} \cdots q_{\theta_n,n}$, where for $\theta \in \{0, 1\}^n$ we set $q_{0,i} := p_i$ and $q_{1,i} := 1 - p_i$.

Suppose now we want to find the probability of a most likely reduction path of M_4 (to 1). Writing $z_{0,i}$ for $-\ln p_i$ and $z_{1,i}$ for $-\ln(1 - p_i)$, the maximum probability above is the minimum of the corresponding negative log-probabilities:

$$\min_{\theta\in\{0,1\}^n} \left\{z_{\theta_1,1} + \cdots + z_{\theta_n,n}\right\}.$$

Remark also that computing the arg min, instead of the min above, gives the most likely trajectories. In either cases, a naïve approach to this computation would inspect *all* possible trajectories. However, this leads to computing and comparing 2^n different sums of positive reals, which is hardly feasible in practice. By contrast, a more efficient strategy for computing the same minimum is to compare (negative-log-)probabilities piece after piece, that is, to compute:

$$\min \left\{z_{0,1}, z_{1,1}\right\} + \cdots + \min \left\{z_{0,n}, z_{1,n}\right\}.$$

In this case we are computing n mins and summing n reals. Moreover, if we keep track, each time we compute a min, of the value $\theta_i \in \{0, 1\}$ producing the minimum, at the end of the computation we even obtain the most likely trajectories $\theta \in \{0, 1\}^n$.

This simple example illustrates the idea behind the already mentioned Viterbi algorithm. Both this algorithm and the sum-product algorithm for Bayesian networks can be seen as instances of a general "distributive law" algorithm [2]. Very roughly, the algorithm exploits the remark that in occurrences of the distributive law of (semi)rings like e.g. $(x+y)\cdot(z+w) = xz+xw+yz+yw$ there are, often, *less* operations to perform to evaluate the left-hand term, compared to the right-hand. So, whenever one is asked to evaluate a possibly too large sum of monomials, it is wise to try to use distributivity *from right to left* as much as possible, so as to express this sum as a product of simpler polynomials. In the case above, we reduced the problem of computing a min across 2^n (tropical) monomials $\mu_\theta := z_{\theta_1,1} + \cdots + z_{\theta_n,n}$ to that of computing the sum (indeed, the tropical product) of n polynomials $m_i := \min\{z_{0,i}, z_{1,i}\}$.

Let us write now the term M_4 in PCF$\langle\vec{X}\rangle$, by replacing probabilities with formal parameters, $M_4 = (\lambda x.x \oplus_{X_1} x)(\lambda x.x \oplus_{X_2} x)\dots(\lambda x.x \oplus_{X_n} x)1$. Then the distributive law argument as above suggests moving from considering *all* 2^n trajectories and look instead at the tropical product:

$$\min\{X_1, \overline{X_1}\} + \cdots + \min\{X_n, \overline{X_n}\} \tag{6}$$

But this time, since the $X_i, \overline{X_i}$ are not reals but just formal variables, it is not clear how to obtain a tropical polynomial from it other than by applying distributivity in *wrong sense*, that is, from left to right, thus getting back to an exponentially large min.

As we discuss through Sections 6 and 7, this is the point where tropical geometry comes to rescue us. In Section 6 we illustrate how we manipulate the *Newton polytope*, a n-dimension polyhedron that is an invariant of tropical polynomials, in order to extract a *not too large* polynomial from a sum like (6) and, more generally, to compute the tropical product of polynomials in an efficient way. Then, in Section 7, we will exploit this method to design an intersection type system $\mathbf{P}_{\mathrm{trop}}$ that enumerates the most likely trajectories of PCF$\langle\vec{X}\rangle$ programs: intuitively, type derivations *select* the most likely trajectories by applying our tropical version of the Viterbi algorithm recursively on the structure of the program. This will enable us to explore, in polynomial time, a space of trajectories of size exponential in the size of the program, providing a significant speed-up to the search for most-likely reductions.

4 Parametric Weighted Relational Semantics

In this section we introduce a semantics for PCF$\langle\vec{X}\rangle$-programs given in terms of *formal power series* whose variables include $\vec{X}$. This semantics is a parameterized version of the weighted relational semantics from [35]. While the presentation of this semantics requires us to combine algebraic and categorical language, the key points to look at for the following are equations (7), (8) and (9), which show how the formal power series in the semantics are related to the probabilities of the execution paths of PCF$\langle\vec{X}\rangle$-programs, as well as the fundamental observation, in Subsection 4.4, that *distinct* formal power series may induce *the same* (tropical) analytic function.

4.1 Formal Power Series

In the following, by semiring we mean commutative and with units 0 and 1. A semiring is *continuous* if it is ordered (compatible with + and $\cdot$) and (among other properties) it admits infinite sums. We will consider the following *continuous* semirings (cf. [35]): $\{0, 1\}$ with Boolean addition and multiplication, $\mathbb{N}^\infty$ with standard addition and multiplication, $\mathbb{R}^\infty_{\geq 0}$ with standard addition and multiplication, and $\mathbb{T}$, the *tropical semiring* (also noted $\mathbb{L}$, cf. [6]), corresponding to $\mathbb{R}^\infty_{\geq 0}$ with reversed order, with min as + and addition as $\cdot$.

We indicate multisets $\mu \in !\Sigma = \Sigma \to \mathbb{N}$ as formal monomials, which allows us to retain standard notations for polynomials/power series. For instance, the multiset $\mu \in !\{0, 1, 2\}$ with $\mu(0) = 2, \mu(1) = 1, \mu(2) = 0$ will be denoted as $0^2 1$ (or $0^2 1 2^0$). Often, for clarity, we introduce a set x_Σ of $\sharp\Sigma$ fresh formal variables x_a, one for each $a \in \Sigma$, and we denote $\mu \in !\Sigma$ by the formal monomial $\prod_{a\in\Sigma} x_a^{\mu(a)}$, also denoted x^μ. For instance, $0^2 1$ becomes the standard $x_0^2 x_1$ (or $x_0^2 x_1 x_2^0$).

Let Σ be a set and Q a semiring. We call $Q\{\!\{\Sigma\}\!\}$ the set of functions $!\Sigma \to Q$, and its elements are called *formal power series (*fps*, for short) over Q with (commuting) variables the elements of* Σ. Given $s \in Q\{\!\{\Sigma\}\!\}$, the image $s_\mu \in Q$ of $\mu \in !\Sigma$ is called the *coefficient of* s *at* μ and $\mathrm{supp}(s) := !\Sigma - s^{-1}0$ is called the *support* $\mathrm{supp}(s)$ *of* s. A fps s is *all-one* when all coefficients s_μ are either 0 or 1. When Σ is finite and the support is finite, s is a *formal polynomial.* We let $Q\{\Sigma\} \subseteq Q\{\!\{\Sigma\}\!\}$ indicate the set of formal polynomials. As usual, we visualize a fps $s \in Q\{\!\{\Sigma\}\!\}$ as the formal sum $s = \sum_{\mu\in!\Sigma} s_\mu x^\mu$, e.g. $s = s_{0^0 1^0} + s_{0^2 1} x_0^2 x_1 + s_{1 0^2} x_0 x_1^2 \in Q\{\!\{\{0, 1\}\}\!\}$. If $\Sigma = \Sigma_1 + \cdots + \Sigma_n$, then $Q\{\!\{\Sigma\}\!\}$ is canonically isomorphic to the set of functions $!\Sigma_1 \times \cdots \times !\Sigma_n \to Q$, which we call $Q\{\!\{\Sigma_1, \cdots, \Sigma_n\}\!\}$, whose elements can be visualized as formal power series with multiple sets $x_{\Sigma_1}, \ldots, x_{\Sigma_n}$ of variables.

The notations introduced above are implicitly compatible with the fact that $Q\{\!\{\Sigma\}\!\}$ is a commutative monoid with pointwise addition, with 0 being the fps $\sum_\mu 0x^\mu$. In fact, $Q\{\!\{\Sigma\}\!\}$ is a semiring with multiplication given by the usual Cauchy's formula: $(ss')_\mu := \sum_{\rho+\eta=\mu} s_\rho s'_\eta$ (this is a sum in Q and exists because it is finite, since μ is), i.e. $ss' = \sum_{\rho,\mu} s_\rho s'_\eta \, x^{\rho+\eta}$. The 1 for this multiplication is the polynomial 1 with our notation, i.e. $1x^{[\,]}$. Polynomials form a sub-semiring for this structure. If Q is continuous, $Q\{\!\{\Sigma\}\!\}$ is also continuous with respect to the pointwise partial order (the bottom element is 0 and supremas are pointwise). The *evaluation map at* $q \in Q^\Sigma$ is the continuous semiring homomorphism $Q\{\!\{\Sigma\}\!\} \to Q$ sending $\sum_\mu s_\mu x^\mu$ to $\sum_\mu s_\mu q^\mu$, where $q^\mu := \prod_{a\in\Sigma} q_a^{\mu(a)} \in Q$.

Any continuous semiring homomorphism $Q \to Q'$ lifts to a continuous semiring homomorphism $Q\{\!\{\Sigma\}\!\} \to Q'\{\!\{\Sigma\}\!\}$ by acting on the coefficients. Remark that sum, products, evaluation map and lifts of homomorphisms above, are all compatible with the bijection $Q\{\!\{\Sigma\}\!\} \simeq Q\{\!\{\Sigma_1, \cdots, \Sigma_n\}\!\}$ and so they are compatible with the multiple variables notation; for example, the evaluation map at $(q_1, \ldots, q_n) \in Q^{\Sigma_1} \times \cdots \times Q^{\Sigma_n}$ would now go from $Q\{\!\{\Sigma_1, \cdots, \Sigma_n\}\!\}$ to Q. Also, remark that for $Q = Q'\{\!\{Z\}\!\}$, a fps $s \in Q\{\!\{X\}\!\} = (Q'\{\!\{Z\}\!\})\{\!\{X\}\!\}$ is the same data as a fps $s \in Q'\{\!\{Z, X\}\!\}$.

We have the following folklore result (proven in the extended version [7]), where for any continuous semiring Q, $q \in Q$ and $n \in \mathbb{N}^\infty$, we write $nq := \sum_{i=1}^n q$ in Q.

Proposition 4.1. *$\mathbb{N}^\infty\{\!\{\Sigma\}\!\}$ is the free continuous commutative semiring on a finite set Σ. For any continuous commutative semiring Q and $q \in Q^\Sigma$, the unique map realizing the universal property is $\mathsf{ev}_q : \mathbb{N}^\infty\{\!\{\Sigma\}\!\} \to Q$, defined by $\mathsf{ev}_q(s) := \sum_\mu s_\mu q^\mu$.*

For a given continuous semiring Q, the category $Q\mathbf{Rel}$ [35] has sets as objects and matrices $Q^{X\times Y}$ as arrows $X \to Y$. The category $Q\mathbf{Rel}_!$ is the coKleisli category of $Q\mathbf{Rel}$ wrt the multiset comonad !, so its arrows $X \to Y$ are matrices in $Q^{!X\times Y}$. $Q\mathbf{Rel}_!$ is cartesian closed, with product $X + Y$, terminal object $\mathbf{1} = \{\star\}$ and exponential $!X \times Y$. Observe that sets in $Q\mathbf{Rel}_!$ play the role of indices, and a matrix $t \in Q^{!X\times Y}$ is the same data as a Y-indexed family of formal power series with commuting variables in X, namely $t = (\sum_{\mu\in !X} t_{\mu,y} x^\mu)_{y\in Y} \in Q\{\!\{X\}\!\}^Y$. This identification is compatible with units and compositions (and linearity): from now on, for us $Q\mathbf{Rel}_!$ *has sets as object and $Q\{\!\{X\}\!\}^Y$ as homsets $X \to Y$.*

Lastly, notice that for any continuous semiring homomorphism $\theta : Q \to Q'$, the induced one $Q\{\!\{\Sigma\}\!\} \to Q'\{\!\{\Sigma\}\!\}$ yields a (cartesian closed) identity-on-objects functor $F_\theta : Q\mathbf{Rel}_! \to Q'\mathbf{Rel}_!$.

4.2 Interpreting (Probabilistic) PCF-Programs as Formal Power Series

Before introducing the interpretation of PCF$\langle\vec{X}\rangle$-typing derivations, let us recall the interpretation $[\![-]\!]^Q$ of standard PCF in the category $Q\mathbf{Rel}_!$, for any continuous semiring Q [35]. Actually, [35] introduces a language PCF^Q with *weighted terms $q \cdot M$*, for q is an element of Q, and a generic choice operator M **or** N, and it is shown that, for any Q, PCF^Q can always be interpreted inside $Q\mathbf{Rel}_!$. The basic types Bool, $\mathbf{N}$ are interpreted by the sets $\{0, 1\}$ and $\mathbb{N}$, respectively, and arrow types $A \to B$ are interpreted as $![\![A]\!] \times [\![B]\!]$. A derivation of $x_1 : A_1, \ldots, x_n : A_n \vdash M : B$ is interpreted as a fps in $Q\{\!\{[\![A_1]\!], \ldots, [\![A_n]\!]\}\!\}^{[\![B]\!]}$, i.e. a $[\![B]\!]$-family of fps with variables in $[\![A_1]\!], \ldots, [\![A_n]\!]$. For instance, a closed program M : Bool is interpreted as a fps in $Q\{\!\{\emptyset\}\!\}^{\{0,1\}} \simeq Q^{\{0,1\}}$, in other words, by two elements[1] $[\![M]\!]_0, [\![M]\!]_1 \in Q$. Weighted and choice terms are interpreted via $[\![q \cdot M]\!] = q \cdot [\![M]\!]$ and $[\![M \textbf{ or } N]\!] = [\![M]\!] + [\![N]\!]$.

One obtains in this way an interpretation of usual probabilistic PCF [21] (*pPCF* for short) in $\mathbb{R}^\infty_{\geq 0}\mathbf{Rel}_!$, translating it into PCF$^{\mathbb{R}^\infty_{\geq 0}}$ via $M \oplus_p N := p \cdot M \textbf{ or } (1-p) \cdot N$. In fact, this interpretation

[1] As common, we simply write $[\![\Gamma \vdash M : A]\!]$ or even just $[\![M]\!]$, but we really mean $[\![\pi]\!]$ for π a given typing derivation for M.

precisely captures the probabilistic execution of closed terms [35]: the interpretation $[\![M]\!]^{\mathbb{R}^\infty_{\geq 0}} \in (\mathbb{R}^\infty_{\geq 0})^{\{0,1\}}$ of a program M : Bool consists in two real numbers $[\![M]\!]_0^{\mathbb{R}^\infty_{\geq 0}}$, $[\![M]\!]_1^{\mathbb{R}^\infty_{\geq 0}}$, describing the probability that M reduces to $\mathsf{i} = 0, 1$ respectively:

$$[\![M]\!]_{\mathsf{i}}^{\mathbb{R}^\infty_{\geq 0}} = \mathbb{P}(M \twoheadrightarrow \mathsf{i}) = \sum_{M \overset{p}{\twoheadrightarrow} \mathsf{i}} p \quad \textit{(where } M \overset{p}{\twoheadrightarrow} \mathsf{i} \textit{ indicates a reduction of probability } p\textit{).} \tag{7}$$

One also obtains an interpretation of pPCF in $\mathbb{T}\mathbf{Rel}_!$ by taking *negative log-probabilities* $-\ln p \in \mathbb{T}$ in place of p, that is, $M \oplus_p N := (-\ln p) \cdot M$ **or** $(-\ln(1-p)) \cdot N$. Since **or** is now interpreted by the min operation, this interpretation describes the negative log-probability of a *most likely* reduction:

$$[\![M]\!]_{\mathsf{i}}^{\mathbb{T}} = \inf\left\{ -\ln p \mid M \overset{p}{\twoheadrightarrow} \mathsf{i}\right\} = -\ln \sup\left\{p \mid M \overset{p}{\twoheadrightarrow} \mathsf{i}\right\}. \tag{8}$$

Example 4.2. For the closed pPCF term $M = 1 \oplus_p (1 \oplus_p 1)$, we have $[\![M]\!]_1^{\mathbb{R}^\infty_{\geq 0}} = p + p(1-p) + (1-p)^2 = 1$, i.e. the sum of the probabilities of all trajectories leading to 1, and $[\![M]\!]_1^{\mathbb{T}} = \min\{z, z+w, 2w\} = \min\{z, 2w\}$, where $z = -\ln p$, $w = -\ln(1-p)$, yielding e.g. $-\ln 2$ when $p = 1 - p = \frac{1}{2}$.

4.3 Interpreting PCF$\langle\vec{X}\rangle$-Programs as Formal Power Series

We now show how to interpret PCF$\langle\vec{X}\rangle$ inside *any* category $Q\mathbf{Rel}_!$. In fact, we interpret it in a "free way", factorizing any such interpretation. Let $\mathbb{X}$ be the set $\{X_1, \overline{X_1}, \dots, X_n, \overline{X_n}\}$. We can translate PCF$\langle\vec{X}\rangle$ to PCF$^{\mathbb{N}^\infty\{\!\{\mathbb{X}\}\!\}}$ via $M \oplus_{X_i} N := X_i \cdot M$ **or** $\overline{X_i} \cdot N$, and $[\![\_]\!]^{\mathbb{N}^\infty\{\!\{\mathbb{X}\}\!\}}$ gives then an interpretation of PCF$\langle\vec{X}\rangle$ inside $(\mathbb{N}^\infty\{\!\{\mathbb{X}\}\!\})\mathbf{Rel}_!$. We call it the *parametric interpretation* and note it $[\![\_]\!]^{X_1,\dots,X_n}$. That is, $[\![\Gamma \vdash M : A]\!]^{X_1,\dots,X_n} \in ((\mathbb{N}^\infty\{\!\{X_1, \overline{X_1}, \dots, X_n, \overline{X_n}\}\!\})\{\!\{[\![\Gamma]\!]\}\!\})^{[\![A]\!]} = (\mathbb{N}^\infty\{\!\{X_1, \overline{X_1}, \dots, X_n, \overline{X_n}, [\![\Gamma]\!]\}\!\})^{[\![A]\!]}$. For e.g. $n = 1$, it is a $[\![A]\!]$-family of fps $\sum_\mu (\sum_{i,j} s_{ij\mu} X^i \overline{X}^j) x^\mu = \sum_{i,j,\mu} s_{ij\mu} X^i \overline{X}^j x^\mu$ $(i, j \in \mathbb{N}, \mu \in ![\![\Gamma]\!])$.

Example 4.3. The parametric interpretation of the term $M = 1 \oplus_X (1 \oplus_X 1)$: Bool (the parametrization of the one in Example 4.2) consists in two fps $[\![M]\!]_0^{X,\overline{X}}, [\![M]\!]_1^{X,\overline{X}} \in \mathbb{N}^\infty\{\!\{X, \overline{X}\}\!\}$, namely $[\![M]\!]_0^{X,\overline{X}} = 0$ and $[\![M]\!]_1^{X,\overline{X}} = X + X\overline{X} + \overline{X}^2$, which represent the (weights of the) possible reductions.

Directly from [35, Theorem V.6], we get that for a closed term M of type e.g. Bool, and $\mathsf{i} \in \{0, 1\}$,

$$[\![M]\!]_{\mathsf{i}}^{X_1,\dots,X_n} = \sum\nolimits_{\vec{i},\vec{j} \in \mathbb{N}^n} \sharp(\vec{i},\vec{j}) X_1^{i_1} \overline{X}_1^{j_1} \dots X_n^{i_n} \overline{X}_n^{j_n} \tag{9}$$

where $\sharp(\vec{i},\vec{j})$ is the number of reductions to i of weight $X_1^{i_1} \overline{X}_1^{j_1} \dots X_n^{i_n} \overline{X}_n^{j_n}$.

Example 4.4. Remember $M_3 = \mathrm{Y}(\lambda x.1 \oplus_X x)$: Bool from the previous section. Its parametric interpretation yields two fps $[\![M_1]\!]_0^{X,\overline{X}}, [\![M_1]\!]_1^{X,\overline{X}}$ where $[\![M_2]\!]_0^{X,\overline{X}} = 0$, as M_2 cannot reduce to 0, and $[\![M_2]\!]_1^{X,\overline{X}} = \sum_n X\overline{X}^n$ describes the weights $\mu \simeq n$ of the *infinitely many* trajectories $M_2 \overset{\mu}{\twoheadrightarrow} 1$.

The parametric interpretation is indeed a parametrisation of the semantics in [35]: by Proposition 4.1, any choice $q \in Q^{\mathbb{X}}$ of actual values of parameters in a Q, canonically induces an interpretation of PCF$\langle\vec{X}\rangle$ inside $Q\mathbf{Rel}_!$ via the functor $F_{\mathsf{ev}_q} : (\mathbb{N}^\infty\{\!\{\mathbb{X}\}\!\})\mathbf{Rel}_! \to Q\mathbf{Rel}_!$. One easily checks that, if $\begin{bmatrix} X := p_X \\ \overline{X} := p_{\overline{X}} \end{bmatrix} \in (\mathbb{R}^\infty_{\geq 0})^{\mathbb{X}}$ then the produced interpretation of a term M of PCF$\langle\vec{X}\rangle$ coincides with the one of the PCF$^{\mathbb{R}^\infty_{\geq 0}}$-term $M[X := p_X, \overline{X} := p_{\overline{X}}]$. Similarly, if $\tau \in \mathbb{T}^{\mathbb{X}}$ associates $X_i, \overline{X_i}$ with negative log-probabilities $-\ln p_i, -\ln(1-p_i)$, the produced interpretation of PCF$\langle\vec{X}\rangle$ terms coincides with the one of the corresponding PCF$^{\mathbb{T}}$-terms.

Example 4.5. For M from Example 4.3, choosing the values $p, 1-p \in \mathbb{R}^\infty_{\geq 0}$ for $X, \overline{X}$ turns the fps $[\![M]\!]_1^{X,\overline{X}} = X + X\overline{X} + \overline{X}^2$ into the real number $[\![M]\!]_1^{\mathbb{R}^\infty_{\geq 0}} = p + p(1-p) + (1-p)^2$. Evaluating $X, \overline{X}$ as $-\ln p, -\ln(1-p) \in \mathbb{T}$ turns it into $[\![M]\!]_1^{\mathbb{T}} = \min\{-\ln p, -2\ln(1-p)\}$ (cf. Example 4.2)..

Example 4.6. Consider M_3 from Example 4.4; choosing $X, \overline{X}$ as $p, 1-p \in \mathbb{R}^{\infty}_{\geq 0}$ turns the fps $[\![M_2]\!]_1^{X,\overline{X}} = \sum_n X\overline{X}^n$ into $[\![M_2]\!]_1^{\mathbb{R}^{\infty}_{\geq 0}} = \sum_n p(1-p)^n = \frac{p}{1-p}$ (if $p \neq 0$). Evaluating them as $-\ln p, -\ln(1-p) \in \mathbb{T}$ turns it into $[\![M]\!]_1^{\mathbb{T}} = \inf_n\{-\ln p - n\ln(1-p)\} = -\ln p$.

4.4 Tropical Analytic Functions

By evaluating at points, formal power series define analytic functions via the map $(\_)^! : Q\{\!\{\Sigma\}\!\} \to [Q^{\Sigma} \to Q]$, where $s^!(q)$ evaluates s at q, i.e. $s^!(q) = \sum_{\mu\in\,!X} s_\mu q^\mu$. The central example we consider is the case of the analytic functions for $Q = \mathbb{T}$:

Definition 4.7. Let Σ have n elements. The functions $\mathbb{T}^n \to \mathbb{T}$ of shape $s^!$, for a fps $s \in \mathbb{T}\{\!\{\Sigma\}\!\}$, are called *tropical analytic functions* (*taf* for short, aka tropical power series) [6, 43]. Concretely,

$$s^!(x_1, \ldots, x_n) = \inf_{\mu\in\,!\Sigma} \{s_\mu + \mu \cdot x\}$$

with $\mu \cdot x := \sum_{i=1}^n \mu(i)x_i$. When s is a formal polynomial, the inf above is a min and $s^!$ is then called a *tropical polynomial function.* These are precisely the piecewise linear functions at the heart of tropical geometry, as we discuss in Section 6.

The map $(\_)^!$ from fps to the corresponding analytic function is *not*, in general, injective. This means that different formal power series may well induce *the same* analytic function. Notably, injectivity fails for $Q = \mathbb{T}$, as the following example shows.

Example 4.8. Let $Q := \mathbb{T}$, $\Sigma = \{*\}$. For a fixed $p \in \mathbb{T}$, let $t := \sum_n px^n \in \mathbb{T}\{\!\{x\}\!\}$ and $s := p \in \mathbb{T}\{\!\{x\}\!\}$. Then $t \neq s$ but $t^! = s^!$. In fact $t^!(q) = p + \inf_n nq = p = s^!(q)$ for all $q \in \mathbb{T}$.

As it will be seen since the next section, it is precisely this mismatch between tropical power series and the corresponding analytic functions that enables a combinatorial and efficient exploration of the most likely behaviour of probabilistic programs.

Remark 4.1. *The considerations above could be rephrased by considering a category $Q\mathbf{An}$ whose objects are sets and the homset from Σ to Y is $Q\mathbf{An}(\Sigma, Y)$, the set of functions $s^! : Q^{\Sigma} \to Q^Y$ defined by $s^!(q)_y = \sum_{\mu\in\,!X} s_{\mu,y} q^\mu$, for some Y-indexed family $s \in Q\{\!\{\Sigma\}\!\}^Y$ of fps.*

Via the map $(\_)^!$, any program $\Gamma \vdash M : A$ yields then a function $[\![M]\!]^! : Q^{[\![\Gamma]\!]} \to Q^{[\![A]\!]}$, *and one may ask what is the status of such interpretation. In the extended version [7] it is shown that $(\_)^!$ turns the exponential of $\mathbb{T}\mathbf{Rel}_!$ into a* weak *exponential in $\mathbb{T}\mathbf{An}$ (cf. [40]), so that the interpretation $[\![-]\!]^!$ produces a* non-extensional *model of* PCF$\langle\vec{X}\rangle$, *that is, one that satisfies the β-rule but not the η-rule.*

5 The Tropical Degree

Suppose M is a probabilistic algorithm that iterates a given protocol until a certain condition is satisfied, and suppose that the computation of M ends after n iterations producing the value V. As we observed in Section 3, we can expect that the probability of producing V after *no less* than n steps does not increase when n is large enough. In this section we show that, in PCF$\langle\vec{X}\rangle$, this intuition is correct and reflects a general phenomenon captured by the tropical semantics.

To state our general result, we first show how to associate, canonically, a *discrete* power series (i.e. with coefficients in $\mathbb{N}^\infty$) with a taf called the *tropicalization* of the power series.

The inclusion $\iota \in Q\{\!\{\Sigma\}\!\}^{\Sigma}$ that sends any element $X \in \Sigma$ onto itself induces the homomorphism $\mathrm{ev}_\iota : \mathbb{N}^\infty\{\!\{\Sigma\}\!\} \to Q\{\!\{\Sigma\}\!\}$. If Q is an idempotent semiring, one can check that ev_ι turns all 0 coefficients into $0 \in Q$ and all coefficients $n \neq 0$ onto $1 \in Q$. That is, it gives the characteristic series of the support: $\mathrm{ev}_\iota(s) = \sum_{\mu\in\mathrm{supp}(s)} x^\mu$. Composed with $(-)^!$, this yields a map $\mathrm{ev}_\iota^! : \mathbb{N}^\infty\{\!\{\Sigma\}\!\}^Y \to Q\mathbf{An}(\Sigma, Y)$. For $Q := \mathbb{T}$, which is idempotent since $+ = \inf$, the above lines yield the following:

Definition 5.1 (tropicalization). Let Σ a finite set. The homomorphism $\mathrm{ev}_\iota : \mathbb{N}^\infty\{\!\{\Sigma\}\!\} \to \mathbb{T}\{\!\{\Sigma\}\!\}$ is called t. Remark that, in terms of real numbers, $\mathrm{t}(s) = \sum_{\mu\in\mathrm{supp}(s)} 0x^\mu$ (the other coefficients, i.e. the $\mathrm{t}(s)_\mu$ for $\mu \notin \mathrm{supp}(s)$, being $+\infty$). We call $\mathrm{t}^! := \mathrm{ev}_\iota^! : \mathbb{N}^\infty\{\!\{\Sigma\}\!\}^Y \to [\mathbb{T}^\Sigma \to \mathbb{T}^Y]$ the *tropicalization map*. For $s \in \mathbb{N}^\infty\{\!\{\Sigma\}\!\}^Y$, the tropicalization $\mathrm{t}^!s : \mathbb{T}^\Sigma \to \mathbb{T}^Y$ of s is the taf concretely given by:

$$\mathrm{t}^!s(x)_y = \inf_{\mu\in\mathrm{supp}(s_y)} \mu\cdot x = \mathrm{t}^!s_y(x).$$

Example 5.2. Let $s \in \mathbb{N}^\infty\{\!\{X,\overline{X}\}\!\}$. Then $\mathrm{t}^!s : \mathbb{T}^2 \to \mathbb{T}$ is $\mathrm{t}^!s(X := x_1, \overline{X} := x_2) = \inf_{\mu\in\mathrm{supp}(s)} \mu(X)\cdot x_1 + \mu(\overline{X})\cdot x_2$. For instance, $\mathrm{t}^!s(X := 0, \overline{X} := +\infty) = \inf_{\mu\in\mathrm{supp}(s)}\{\mu(\overline{X})\cdot\infty\}$. It is immediate to see that this value is 0 if there is μ such that $s_\mu \neq 0$ and $\mu(\overline{X}) = 0$, while it is $+\infty$ otherwise. For instance, for $M = 1 \oplus_X 0$, we have $\mathrm{t}^![\![M]\!]_1 = X \in \mathbb{N}^\infty\{\!\{X,\overline{X}\}\!\}$, and $\mathrm{t}^![\![M]\!]_1(X := 0, \overline{X} := +\infty) = 0$. Similarly, $\mathrm{t}^![\![M]\!]_0(X := 0, \overline{X} := +\infty) = +\infty$. The first corresponds to the presence of the reduction $M \overset{X}{\twoheadrightarrow} 1$, the second to the absence of reduction $M \overset{\overline{X}}{\twoheadrightarrow} 0$. In fact, the choice $X := 0, \overline{X} := +\infty$ corresponds to choosing the left side of a probabilistic choice with probability 1 (so 0 for the right side), and $\mathrm{t}^![\![M]\!]_\mathrm{i}$ returns thus negative log-probabilities when computed on negative log-probabilities.

The situation of the previous example is not a coincidence. In fact, via tropicalisation, a program M : Bool is turned into two taf $\mathrm{t}^![\![M]\!]_\mathrm{i} : \mathbb{T}^\mathbb{X} \to \mathbb{T}$ which compute the negative log-probability of most likely reductions of M, as the following proposition shows. This is therefore a "parametrization" (allowing all choices of probabilities) of [6, Corollary 10] or [35, Theorem V.6].

Proposition 5.3. *Given M : Bool, let $p \in [0,1]^\mathbb{X}$ any assignment of probabilities p to the parameters. Then (remark that $M[X := p_X]$ is a PCF$^{[0,1]}$-term)*

$$(\mathrm{t}^![\![M]\!]_\mathrm{i}^\mathbb{X})(X := -\ln p_X, \overline{X} := -\ln p_{\overline{X}}) = -\ln\ \sup\{q \mid M[X := p_X] \overset{q}{\twoheadrightarrow} \mathrm{i}\}.$$

The negative log-probabilities above are computed as an inf across *all* trajectories leading to i. Our goal is now to show that, *independently of the parameters*, such an inf is always found within a *finite* set of trajectories (and is, therefore, a min). The key result to get there is the following:

Proposition 5.4. *Let $k \in \mathbb{N}$ and $\{s_n \mid n \in \mathbb{N}^k\} \subseteq \mathbb{N}^\infty$. Then there exists a* finite *set $S \subseteq \mathbb{N}^k$ such that $s_n < +\infty$ for all $n \in S$ and, for all $x \in \mathbb{T}^n$, $\inf_{n\in\mathbb{N}^k}\{nx + s_n\} = \min_{n\in S}\{nx + s_n\}$.*

Proof. Fix the well-founded order $m \prec n$ on $\mathbb{N}^k$ by saying that $m_i \leq n_i$ for all $1 \leq i \leq k$ *and* there is at least one $1 \leq j \leq k$ such that $m_j < n_j$. We claim that we can let $S := \{n \in \mathbb{N}^k \mid s_n < +\infty$ *and for all $m \prec n$ one has $s_m >\ s_n$*$\}$. Indeed, if S is infinite then one can easily see, using König's Lemma, that there is a chain $m_0 \prec m_1 \prec \cdots \subseteq S$. But by definition of S this gives a chain $s_{m_0} > s_{m_1} > \cdots \subseteq \mathbb{N}$, which is absurd. We have thus shown that S is finite. For the claimed equation, Wlog $S \neq \emptyset$ (otherwise on the one hand min $:= +\infty$, and on the other hand one can easily see, by induction on $\prec$, that inf $= +\infty$). Now fix $x \in \mathbb{T}^n$. We show by induction on n wrt $\prec$, that $\forall n \in \mathbb{N}^k$, $\exists m \in S$ such that $s_m + mx \leq s_n + nx$. Notice that this proves the desired equation.

Case $n = 0$: Wlog $n \notin S$. By definition of S, $s_n = +\infty$ and so any $m \in S \neq \emptyset$ works.

Case $n{>}0$: Wlog $n \notin S$. By definition of S, we have two cases: either $s_n = +\infty$, which is done as above. Or there is $n' \prec n$ with $s_{n'} \leq s_n$. But then $s_{n'} + n'x \leq s_n + n'x \leq s_n + (n - n')x + n'x = s_n + nx$. If $n' \in S$, take $m := n'$. If $n' \notin S$, take the $m \in S$ with $s_m + mx \leq s_{n'} + n'x$ given by the IH on n'. □

Remark 5.1 (not all taf are polynomials). *An essential ingredient in the (proof of the) result above is that of considering fps with coefficients in a discrete set (like $\mathbb{N}^\infty$). In general, a fps with coefficients in $\mathbb{T}$ needs not be equivalent to a polynomial: consider the fps $s = \sum_{n\in\mathbb{N}} \frac{1}{2^n}x^n \in \mathbb{T}\{\!\{x\}\!\}$; the corresponding tropical analytic function $s^! : \mathbb{T} \to \mathbb{T}$ is not a polynomial function, since $s^!(0) = \inf_n\{n\cdot 0 + 1/2^n\} = 0$ is an* inf *that cannot be reduced to a* min.

Corollary 5.5. *For all terms M : $\mathrm{Bool}^n \to \mathbf{N}$ (i.e.* $\mathrm{Bool} \to \cdots \to \mathrm{Bool} \to \mathbf{N}$*) of* $\mathrm{PCF}\langle\vec{X}\rangle$ *and* $i \in \mathbb{N}$ *there is an all-one polynomial* $s \in \mathbb{T}\{\mathbb{X} \cup \{0,1\}^n\}$ *with* $\mathrm{t}^!\llbracket M \rrbracket_{\mathrm{i}} = s^!$ *and* $\mathrm{supp}(s) \subseteq \mathrm{supp}(\llbracket M \rrbracket_{\mathrm{i}})$.

Proof. We have $\llbracket M \rrbracket_{\mathrm{i}} \in \mathbb{N}^\infty\{\!\{\Sigma\}\!\}$, for $\Sigma := \{X_1, \overline{X}_1, \dots, X_k, \overline{X}_k\} \cup \{0,1\}^n$. Then we can identify $!\Sigma \simeq \mathbb{N}^{2k+2^n}$. Let $s_\mu := 0$ if $\mu \in \mathrm{supp}(\llbracket M \rrbracket_{\mathrm{i}})$ and $:= +\infty$ otherwise. Then Proposition 5.4 gives a finite $S \subseteq !\Sigma$ such that $\mathrm{t}^!\llbracket M \rrbracket_{\mathrm{i}}(x) = \inf_{\mu \in \mathrm{supp}(\llbracket M \rrbracket_{\mathrm{i}})}\{\mu \cdot x\} = \inf_{\mu \in \mathbb{N}^k}\{\mu \cdot x + s_\mu\} = \min_{\mu \in S}\{\mu \cdot x\} = s^!(x)$, for the polynomial $s := \sum_{\mu \in S} 1_{\mathbb{T}} x_\Sigma^\mu \in \mathbb{T}\{\Sigma\}$. Moreover, remark that $\mathrm{supp}(s) = S$. Then, for $\mu \in \mathrm{supp}(s)$, from Proposition 5.4, we have $s_\mu < +\infty$, so $\mu \in \mathrm{supp}(\llbracket M \rrbracket_{\mathrm{i}})$ by definition of s_μ. □

Intuitively, the polynomial s takes into account only a finite number of the trajectories of M. Yet, the result above shows that the sup negative-log-probability across all trajectories is always found within the finite set S selected by s (and is, therefore, a max). This leads to the following:

Definition 5.6. Let $M : \mathrm{Bool}^n \to \mathbf{N}$ in $\mathrm{PCF}\langle\vec{X}\rangle$. For $i \in \mathbb{N}$, the *tropical degree* $\mathfrak{d}_i(M)$ *of* M *at* i is the minimum degree of an all-one polynomial $s \in \mathbb{T}\{\mathbb{X}\}$ with $\mathrm{t}^!\llbracket M \rrbracket_{\mathrm{i}} = s^!$ and $\mathrm{supp}(s) \subseteq \mathrm{supp}(\llbracket M \rrbracket_{\mathrm{i}})$.

The tropical degree expresses the fact that the sup of the probabilities of the execution paths of M to i can be obtained by only looking at a finite number of execution paths whose degree is at most $\mathfrak{d}_{\mathrm{i}}(M)$. For example, let us expand the definition above for a closed term M : Bool with parameters $X, \overline{X}$ and let, say, $\mathrm{i} = 0$. Remember that $\llbracket M \rrbracket_0^X = \sum_{i,j \in \mathbb{N}} \sharp(i,j) X^i \overline{X}^j$, with $\sharp(i,j)$ the number of reductions to 0 of weight $X^i \overline{X}^j$. Remembering Proposition 5.3 too, $\mathfrak{d}_0(M)$ is the *smallest* $d \in \mathbb{N}$ such that there is a *finite* $S \subseteq \mathbb{N}^2$ with

1. $\max_{(i,j) \in S} i + j = d$,
2. for all $(i,j) \in S$ there is a reduction $M \twoheadrightarrow 0$ of weight $X^i \overline{X}^j$,
3. for all $p \in [0,1]$, the sup of all probabilities P across *any* reduction $M[X := p] \overset{P}{\twoheadrightarrow} 0$ equals the max across the reductions chosen in S : $\sup\{P \mid M[X := p] \overset{P}{\twoheadrightarrow} 0\} = \max_{(i,j) \in S} p^i (1-p)^j$.

For example, one can easily see that the term M_3 from Section 3 satisfies $\mathfrak{d}(M_3) = 1$.

Example 5.7. Let M : Bool with parameters $X, \overline{X}$. Suppose that $M \twoheadrightarrow \mathrm{i}$, for a fixed $i \in \{0,1\}$. Let us show that $\mathfrak{d}_{\mathrm{i}}(\mathrm{i} \oplus_X (M \oplus_X \Omega)) = 1$, where $\Omega := \mathrm{Y}I$: Bool is non-terminating. This means showing that 1 is the smallest $d \in \mathbb{N}$ for which there is a finite $S \subseteq \mathbb{N}^2$ satisfying 1), 2), 3) from above. Observe that d cannot be 0, since there is no reduction $\mathrm{i} \oplus_X (M \oplus_X \Omega) \overset{X^0 \overline{X}^0}{\twoheadrightarrow} \mathrm{i}$. Let us show that $d = 1$ satisfies all the required properties: the point is to guess the right finite set of reductions of maximal degree 1. Take $S = \{\left[\begin{smallmatrix} X = 1 \\ \overline{X} = 0 \end{smallmatrix}\right]\} \subseteq \mathbb{N}^2 \simeq !\{X, \overline{X}\}$, i.e. select the reduction $\mathrm{i} \oplus_X (M \oplus_X \Omega) \overset{X}{\twoheadrightarrow} \mathrm{i}$. This reduction exists, so 1) and 2) are satisfied. For 3), we need to show that the selected set maximises the sup of all the possible probabilities of all the possible reductions. Since Ω does not terminate, the possible reductions to i are exactly those of weight either X or $X^{i+1}\overline{X}^{j+1}$, for some $i, j \in \mathbb{N}$ such that $M \overset{X^i \overline{X}^j}{\twoheadrightarrow} \mathrm{i}$. It is clear then that, for any choice of $p \in [0,1]$, the reduction corresponding to $\mathrm{i} \oplus_X (M \oplus_X \Omega) \overset{X}{\twoheadrightarrow} \mathrm{i}$ is the one with maximum weight.

Example 5.8. This example shows how the tropical degree is sensible to the number of parameters. Let M : Bool, now with parameters $X_1, \overline{X_1}, X_2, \overline{X_2}$. Suppose that $M \twoheadrightarrow \mathrm{i}$, for a fixed $i \in \{0,1\}$. Let us show that $\mathfrak{d}_{\mathrm{i}}(\mathrm{i} \oplus_{X_1} (M \oplus_{X_2} \Omega)) \geq 2$. It is easy, arguing as before, to see that the tropical degree cannot be 0. But now, contrarily to the above, we can also exclude it to be 1. In this case the reductions to i are either that of weight X_1, or those of weight $X_1^{i_1}\overline{X}_1^{j_1+1} X_2^{i_2+1}\overline{X}_2^{j_2}$, for $M \overset{X_1^{i_1}\overline{X}_1^{j_1} X_2^{i_2}\overline{X}_2^{j_2}}{\twoheadrightarrow} \mathrm{i}$. Now, while the only possible choice of a set S of reductions of maximal degree 1, is the one of weight X_1, this choice does not maximises all the possible probabilities, i.e. it does not satisfy 3). For example, assigning X_1, X_2 with, respectively, probabilities $(p_1, p_2) \in [0,1]^2$ such that $p_1 < \frac{1}{2}$ and $p_2 > 2p_1$ (for example, $p_1 = \frac{1}{4}$ and $p_2 = \frac{2}{3}$), one can see that a reduction $\mathsf{i} \oplus_{X_1} (M \oplus_{X_2} \Omega) \overset{\overline{X_1}}{\twoheadrightarrow} M \oplus_{X_2} \Omega \overset{X_2}{\twoheadrightarrow} M \overset{P'}{\twoheadrightarrow} \mathsf{i}$, with P' large enough, might yield a probability $(1-p_1)p_2P' > p_1$, thus violating 3).

Beyond these relatively simple cases, the general problem of finding the tropical degree of a $\mathrm{PCF}\langle\vec{X}\rangle$ program is not decidable (and indeed the proof of Corollary 5.5 is non-constructive).

Theorem 5.9. *Both the problems of input a number $d \in \mathbb{N}$ and a term M : Bool of $\mathrm{PCF}\langle X_1, \dots, X_k\rangle$ with $k \geq 2$, and of respective output "yes" if $\mathfrak{d}_{\mathsf{i}}(M) < d$ and "no" otherwise, and "yes" if $\mathfrak{d}_{\mathsf{i}}(M) = d$ and "no" otherwise, are* not *RE, and Π^0_1-hard*

Proof. We reduce the complementary of the halting problem (which is not RE and is Π^0_1-complete) to Problem 1 and to Problem 2. Given in input a closed M : Bool of $\mathrm{PCF}\langle\mathbb{X}\rangle$, take $X_1 \neq X_2$ (does not matter whether they belong to $\mathbb{X}$ or not) and let $\widetilde{M} := \mathsf{i} \oplus_{X_1} (M \oplus_{X_2} \Omega)$. If M is normalisable to i then, by arguing similarly to Example 5.8, we see that $\mathfrak{d}_{\mathsf{i}}(\widetilde{M}) \geq 2$. If M is not normalisable to i then, since Ω also is not normalisable to i, we have $[\![\widetilde{M}]\!]_{\mathsf{i}} = X_1$ and so $\mathfrak{d}_{\mathsf{i}}(\widetilde{M}) = 1$. Summing up, $\mathfrak{d}_{\mathsf{i}}(\widetilde{M}) = 1$ iff M is not normalisable to i iff $\mathfrak{d}_{\mathsf{i}}(\widetilde{M}) < 2$. Hence an oracle semideciding either problem 1 or 2 allows to semidecide the non-normalisability of M. □

The previous examples show that computing (or even estimating) the tropical degree by hand is a subtle task and, in general, exact values or even upper-estimations are not mechanisable. This poses obvious limitations to the what can be achieved in general, algorithmically. However, in the next Sections we will show that it is still possible to design an algorithm that progressively computes estimations of the tropical degree eventually *stabilizing* onto the correct value $\mathfrak{d}_{\mathsf{i}}(M)$.

6 Convex Geometry and Newton Polytopes

As already mentioned, in this and the next section, by combining the toolbox of tropical geometry with the one of programming language theory, we define an efficient procedure to solve the inference problems (I1) and (I2) for a term M : Bool/N, that is, to compute the maximum a posteriori (log)probabilities of producing a given value, say 1, and to produce a most likely explanation for it.

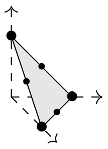
While the finiteness of the tropical degree ensures that we may restrict ourselves to explore only a finite set of reductions S, the size of S may still be exponentially large (cf. Section 3.2).

In this section we show that one can compute some polynomially bounded subset $S' \subset S$ that still contains enough trajectories to track the most likely ones. This set S' is obtained by associating a program (in fact, its associated fps) with a polytope, called the *minimal Newton polytope*, which is a variant of the well-known Newton polytope of a polynomial. The same method will then be used to design an algorithm, similar to the Viterbi algorithm, to compute the tropical product of polynomials in an efficient way. This algorithm will be the key ingredient, in the next section, to design a compositional procedure to track the most likely trajectories of a $\mathrm{PCF}\langle\vec{X}\rangle$-program.

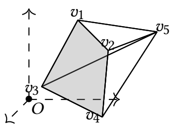
6.1 The Newton Polytope

In this subsection we recall the standard definition of the Newton polytope of a polynomial. Let us fix some all-one polynomial $s = \sum_{\mu} \mu \in \overline{\mathbb{R}}\{\Sigma\}$ in n variables, where $\overline{\mathbb{R}} := \mathbb{R} \cup \{+\infty\}$. This is the standard tropical semiring in which tropical algebraic geometry is usually carried on. It is well-known that the piece-wise linear function $f_s : \mathbb{R}^n \to \mathbb{R}$ defined by s by $f_s(x) = \min_{\mu \in \mathrm{supp}(s)}\{\mu \cdot x\}$ can be characterized via two, dual, geometric objects:

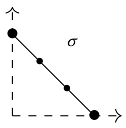
- the *tropical variety* $\gamma(f_s)$, i.e. the set of all *tropical roots* of s, i.e. the $x \in \mathbb{R}^n$ such that the minimum $f_s(x)$ is reached by *at least two* monomials (i.e., f_s is not differentiable at x);



(a) Geometric proof of the "old freshman dream" $(X_1+X_2+X_3)^2 = X_1^2 + X_2^2 + X_3^2$. The triangle is the convex hull of the points corresponding to the monomials in $(X_1 + X_2 + X_3)^2$. The triangle is spanned by the three vertices corresponding to X_1^2, X_2^2, X_3^2.



(b) Illustration of $NP_{\min}(s) \subset NP(s)$, for s given in Example 6.7: $NP(s)$ is the convex hull of $v_1, \ldots, v_5$. $NP_{\min}(s)$, coloured in grey, is the convex hull of the vertices $v_1, \ldots, v_4$. Notice that $v_5 \notin NP_{\min}(s)$, as it is not minimal (e.g. $v_3 \prec v_5$).



(c) For the term M_4 of Example 6.10, both $NP_{\min}(M_4)$ and $NP(M_4)$ are given by the segment σ. Its vertices $(0, 3), (3, 0)$ contain, by Proposition 6.5, the weights of at least one most likely reduction of M_4 (no matter how probabilities are assigned to the parameter X).

Fig. 3. Illustrations of the Newton polytope.

- the *Newton polytope* $NP(s)$, i.e. the convex hull in $\mathbb{R}^n_{\geq 0}$ of the points $\mu \in \mathbb{N}^n$ in $\mathrm{supp}(s)$.

$\gamma(f_s)$ and $NP(s)$ describe two polyhedra in $\mathbb{R}^n$ with dual graphs (see [36]), and any tropical root $a \in \gamma(f_s)$ of s uniquely identifies a *facet* F_x of $NP(s)$: a individuates $k \geq 2$ monomials $\mu_1, \ldots, \mu_k \in \mathbb{N}^n$ such that $\mu_1 \cdot a = \cdots = \mu_k \cdot a =: b \in \mathbb{R}$, so that a is, by construction, normal to the hyperplane H_a of $\mathbb{R}^n$ of equation (in z) $a \cdot z = b$ and H_a is the supporting hyperplane of a unique facet F_a of $NP(s)$, namely the one containing the points $\mu_1, \ldots, \mu_k$.

A crucial remark now is that, even if we defined $NP(s)$ as the convex hull of the possibly very large set of points $\mathrm{supp}(s)$, it is uniquely determined by the set $\mathrm{Vert}(NP(s)) \subseteq \mathrm{supp}(s) \subseteq NP(s)$ of its *vertices* which is, in general, much smaller and in average efficiently computable:

Theorem 6.1. *Let $s \in \overline{\mathbb{R}}\{x_1, \ldots, x_n\}$. Then then* $\#\mathrm{Vert}(NP(s)) = O(d^{2n-1})$ *([45, 46]), where d is the degree of s, and* $\mathrm{Vert}(NP(s))$ *can be computed with a randomised algorithm in expected time* $O(|s|^{\lfloor \frac{n}{2} \rfloor})$ *([8, p. 256 (line 4 from the bottom)]), where* $|s| := \#\mathrm{supp}(s)$.

A consequence of all this discussion is that, for a polynomial s of degree d, we can always find a polynomial s' formed by a *subset* (namely, $\mathrm{Vert}(NP(s))$) of the monomials of s of size polynomial in d such that $NP(s) = NP(s')$ and, crucially, the functions f_s and $f_{s'}$ coincide. Indeed:

Lemma 6.2. *Let $s = \sum_\mu \mu \in \overline{\mathbb{R}}\{\Sigma\}$. Then* $\min_{\mu \in \mathrm{supp}(s)}\{\mu \cdot x\} = \min_{\mu \in \mathrm{Vert}(NP(s))}\{\mu \cdot x\}$.

Proof. This is an immediate consequence of a well-known fact in linear optimisation (cfr. [7]): for a polytope P, the inf on P of a linear function $\mu \cdot x$ is found on the vertices of P. □

Example 6.3. Consider the polynomial $s = \sum_{i+j+k=2} X_1^i X_2^j X_3^k$. Then $NP(s)$, illustrated in gray in Fig. 3a, is the convex hull of all the points $(i, j, k) \in \mathbb{N}^3$ such that $i+j+k = 2$. $NP(s)$ is generated by its vertices, which are the three bold points $(2, 0, 0), (0, 2, 0), (0, 0, 2)$ in the figure. We deduce that f_s coincides with $f_{s'}$, where $s' = X_1^2 + X_2^2 + X_3^2$. What we have just described is in fact a geometric proof of the "old freshman dream" $(x_1 + x_2 + x_3)^2 = x_1^2 + x_2^2 + x_3^2$ for tropical polynomial functions.

6.2 The Minimal Newton Polytope

We now address the following question, indeed a finitary variant of the problem discussed in Section 5: given some very large, although finite, *polynomial* $s \in \mathbb{T}\{\mathbb{X}\}$, can we find a sufficiently *smaller*, and somehow *minimal*, all-one polynomial s' such that $\mathrm{t}^! s = (s')^!$?

We have seen that the Newton polytope $NP(s)$ and its small number of vertices (Theorem 6.1) precisely serves this purpose... but for the fact that $NP(s)$ characterizes the function $\mathbb{R}^n \longrightarrow \mathbb{R}$ defined by tropical polynomials over the tropical semiring $\mathbb{R} \cup \{+\infty\}$, while we are interested in the function $\mathrm{t}^! s : \mathbb{T}^n \longrightarrow \mathbb{T}$ defined by tropical polynomials s over the tropical semiring $\mathbb{T}$ (with carrier set $[0, +\infty]$). To overcome this mismatch, we introduce the following:

Definition 6.4. The *minimal Newton Polytope* $NP_{\min}(s)$ of $s = \sum_\mu s_\mu \mu \in \mathbb{T}\{\Sigma\}$ is the convex hull in $\mathbb{R}^n_{\geq 0}$ of the following set, which is easily seen to be its set of vertices and non-empty:

$$\mathrm{Vert}(NP_{\min}(s)) := \{\mu \in \mathbb{N}^n \mid \mu \textit{ minimal element of } (\mathrm{Vert}(NP(s)), \preceq)\} \subseteq \mathrm{supp}(s),$$

where $\preceq$ is the pointwise (well-founded) order. We also set $s_{\min} := \sum_{\mu \in \mathrm{Vert}(NP_{\min}(s))} \mu \in \mathbb{T}\{\Sigma\}$.

The polytope $NP_{\min}(s)$ precisely captures the behavior of the function $\mathrm{t}^! s(x) : \mathbb{T}^n \longrightarrow \mathbb{T}$: the latter coincides with the min computed over the monomials in $NP_{\min}(s)$:

Proposition 6.5. *Let $s \in \mathbb{T}\{\Sigma\}$. Then $\min_{\mu \in \mathrm{Vert}(NP(s))}\{\mu \cdot x\} = \min_{\mu \in \mathrm{Vert}(NP_{\min}(s))}\{\mu \cdot x\}$. In particular, since the latter is $\mathrm{t}^! s_{\min}(x)$, it follows from Lemma 6.2 that $\mathrm{t}^! s(x) = \mathrm{t}^! s_{\min}(x)$.*

Proof. Fix $x \in \mathbb{R}^n$. The $(\leq)$ is trivial. For $(\geq)$, we show, by induction on $\mu \in (\mathrm{Vert}(NP(s)), \preceq)$, that for all $\mu \in \mathrm{Vert}(NP(s))$, there is $\rho \in \mathrm{Vert}(NP_{\min}(s))$ such that $\rho \cdot x \leq \mu \cdot x$. If μ is minimal, then by definition $\mu \in \mathrm{Vert}(NP_{\min}(s))$, so we are done. If μ is not, there is $\nu \in \mathrm{Vert}(NP(s))$ with $\nu \prec \mu$. By IH there is $\rho \in \mathrm{Vert}(NP_{\min}(s))$ such that $\rho \cdot x \leq \nu \cdot x \leq \mu \cdot x$, and we are done. □

The following shows that the set of vertices of $\mathrm{NP}_{\min}(s)$ can be computed fast wrt s. Remembering that $|s|$ is the cardinality $\#(\mathrm{supp}(s))$ of the support of s, we have:

Theorem 6.6. *Let $s \in \mathbb{T}\{\Sigma\}$. The set $\mathrm{Vert}(NP_{\min}(s))$ (i.e. $s_{\min}$) can be computed with a randomised algorithm in expected time $O(n|s|^{\max\{2,n\}})$.*

Proof. First, compute $\mathrm{Vert}(NP(s))$. For each $\mu \in \mathrm{Vert}(NP(s))$, the minimality check for μ can be done in time $n(\#\mathrm{Vert}(NP(s)) - 1)$ and, since we have $\#\mathrm{Vert}(NP(s))$ of them, we have an additional time $\sim n\,\#\mathrm{Vert}(NP(s))^2$. By Theorem 6.1, we can compute $\mathrm{Vert}(NP(s))$ in expected time $O(|s|^{\lfloor \frac{n}{2} \rfloor})$. Also $\#\mathrm{Vert}(NP(s)) = O(|s|)$, whence the total time $O(|s|^{\lfloor \frac{n}{2} \rfloor}) + nO(|s|^2) = O(n|s|^{\max\{2,n\}})$. □

Example 6.7. Let $s = X_1^2X_2^3X_3^2 + X_1^3X_2^2X_3^2 + X_1X_2X_3^3 + X_1^3X_3^3 + X_1^5X_2^3X_3^4 + X_1^4X_2^2X_3^3 \in \mathbb{T}\{\!\{X_1, X_2, X_3\}\!\}$. Fig. 3b illustrates $NP(s)$, the convex hull of the points $v_1 = (2,3,2), v_2 = (3,2,2), v_3 = (1,1,3), v_4 = (3,0,3), v_5 = (5,3,4), v_6 = (4,2,3)$, of which only $v_1, \dots, v_5$ are vertices, as v_6 is convex combination of v_4, v_5. v_5 is the only non minimal vertex, since e.g. $v_1 \prec v_5$, so $\mathrm{Vert}(NP_{\min}(s)) = \{v_1, v_2, v_3, v_4\}$.

Remark 6.1. *While the algorithm for $NP_{\min}(s)$ in Theorem 6.6 rests on a brute force minimality check on $NP(s)$ (still polynomial in d), a potential speed up may arise from geometric considerations. In general, given a polytope P in $\mathbb{R}^n$, any of its facets f lies, by definition, on its supporting hyperplane H_f, and moreover P is all contained inside one of the two closed halfspaces H_f^+, H_f^- in which H_f divides $\mathbb{R}^n$. Call H_f^+ the one containing P. Let us call f* negatively oriented *if the normal unit vector to H_f towards H_f^- has all strictly negative coordinates (in the canonical base). Intuitively, f "sees the origin". Now, it can be easily proven (see [7]) that a vertex v belonging to some negatively oriented facet f of P is always minimal. The example in Fig. 3b illustrates this fact, since in this case $NP_{\min}(s)$ coincides with the unique negatively oriented facet of $NP(s)$. This suggests that, to compute the minimal vertices, one could start by first selecting the negatively oriented facets, and restrict the brute force minimality check to the remaining ones. However, the eventual speed up depends on the concrete representation of a polytope in use in the algorithm, so we leave such investigations for future work.*

It is worth to rephrase and summarise what we got so far: given a PCF$\langle X_1, \dots, X_n \rangle$ term $M : \mathbf{N}$ and some fixed outcome i, the parametric interpretation gives rise to a fps $[\![M]\!]_{\mathsf{i}}^{\mathbb{X}} \in \mathbb{N}^{\infty}\{\!\{\mathbb{X}\}\!\}$. If the latter is a polynomial, then we have (cfr. Definition 6.4) a minimal polynomial $(\mathsf{t}[\![M]\!]_{\mathsf{i}}^{\mathbb{X}})_{\min}$ or, equivalently, its Newton Polytope, call it $NP^{i}_{\min}(M)$. By Proposition 6.5, this polytope *approximates* the tropical degree $\mathfrak{d}_{\mathsf{i}}(M)$, in the sense that, for all probability assignment to the parameters of the program, the vertices of $NP^{i}_{\min}(M)$ (i.e. the minimal monomials in $[\![M]\!]_{\mathsf{i}}^{\mathbb{X}}$) contain (the monomial of) *at least one* most likely reduction. This reads as $\mathsf{t}^{!}[\![M]\!]_{\mathsf{i}}^{\mathbb{X}} = (\mathsf{t}^{!}[\![M]\!]_{\mathsf{i}}^{\mathbb{X}})_{\min}$, whence $\mathfrak{d}_{\mathsf{i}}(M) \leq \deg((\mathsf{t}[\![M]\!]_{\mathsf{i}}^{\mathbb{X}})_{\min})$. Figure 3c illustrates this discussion for the term M_4 from Example 6.10 (where the situation is more trivial: $\mathfrak{d}_1(M_4) = \deg([\![M_4]\!]_1^{\mathbb{X}}) = 3$, because all reductions of M_4 have degree 3).

But we can actually do the same even if the fps $[\![M]\!]_{\mathsf{i}}^{\mathbb{X}}$ is *not* a polynomial (i.e. its support is infinite): by arguing similarly to Proposition 5.4, one sees that the minimal monomials in its support are still in *finite* number, therefore one still has a *polynomial* $(\mathsf{t}[\![M]\!]_{\mathsf{i}}^{\mathbb{X}})_{\min}$ or, equivalently, its Newton Polytope, call it $NP^{i}_{\min}(M)$. Moreover, one can follow the same proof of Proposition 6.5 in order to show that $NP^{i}_{\min}(M)$ still satisfies the exact same approximation property as in the finite case.

In conclusion, we can *always* associate $M : \mathbf{N}$ with a minimal Polytope $NP^{i}_{\min}(M)$, i.e. a minimal all-one polynomial $(\mathsf{t}[\![M]\!]_{\mathsf{i}}^{\mathbb{X}})_{\min} \in \mathbb{T}\{\mathbb{X}\}$ with $\mathsf{t}^{!}[\![M]\!]_{\mathsf{i}}^{\mathbb{X}} = (\mathsf{t}^{!}[\![M]\!]_{\mathsf{i}}^{\mathbb{X}})_{\min}$ and $\mathfrak{d}_{\mathsf{i}}(M) \leq \deg((\mathsf{t}[\![M]\!]_{\mathsf{i}}^{\mathbb{X}})_{\min})$.

Notice that this is not in contradiction with Theorem 5.9: the above upper-bound always holds, but in general we can only hope to compute *all* the minimal monomials for finite interpretations.

6.3 The Viterbi-Newton Algorithm

Recall that, from the discussion around (6) in Section 3, tracking the most likely runs of an application MN requires to be able to compute (tropical) products of polynomials efficiently. We will now use the results from the previous subsection to define an algorithm **VN** to compute, given k polynomials $s_1, \dots, s_k$, a minimal polynomial s capturing the tropical product of the s_i.

First observe that the number of monomials in $s_1 \dots s_k$ grows exponentially in k. For instance, letting all s_i be the same polynomial $X_1 + \cdots + X_n$, we have that $s_1 \dots s_k = (X_1 + \cdots + X_n)^k$ contains $\binom{n+k-1}{k-1} \in O((n+k-1)^{k-1})$ distinct monomials. However, we have seen (cf. Example 6.3) that in the tropical setting we have $(X_1 + \cdots + X_n)^k = X_1^k + \cdots + X_n^k$. Indeed, as we show below, a sufficiently small set of monomials is enough to capture the polynomial function $\mathsf{t}^{!}(\prod_{i=1}^{k} s_i)$.

The main idea behind the algorithm described below is to compute the product as an operation performed directly over the minimal Newton polytopes $NP_{\min}(s_1), \dots, NP_{\min}(s_n)$, and producing the minimal polytope $NP_{\min}(s_1 \dots s_n)$ (this is reminiscent of the *polytope algebra* of [42]). The fundamental remark is that the product of polynomes translates into the *Minkowski sum* of the corresponding polytopes, defined as $A + B = \{v + w \mid v \in A, w \in B\}$ for two sets in $\mathbb{R}^n$. The set $\mathrm{Vert}(A + B)$ can be computed in time $O(nm)$, where $n = |\mathrm{Vert}(A)|, m = |\mathrm{Vert}(B)|$, cf. [16]. Using the well-known fact that $NP(s_1 s_2) = NP(s_1) + NP(s_2)$, we can prove:

Lemma 6.8. *Let $s_1, s_2 \in \mathbb{T}\{\!\{\Sigma\}\!\}$. $\mathrm{Vert}(NP_{\min}(s_1 s_2)) = (\mathrm{Vert}(NP_{\min}(s_1)) + \mathrm{Vert}(NP_{\min}(s_2)))_{\min}$.*

We now show the existence an algorithm (which we call **VN** – for “Viterbi+Newton”) to compute the minimal polynomial $(\prod_{i=1}^{k} s_i)_{\min}$ efficiently from $(s_1)_{\min}, \dots, (s_k)_{\min}$.

Theorem 6.9. *Let $k \geq 2$, $s_1, \dots, s_k \in \mathbb{T}\{\!\{\Sigma\}\!\}$ be minimal (i.e. such that $s_i = (s_i)_{\min}$), let $d := \max_i \deg(s_i)$ and $n := \#\Sigma$. There is an algorithm $\mathbf{VN}(s_1, \dots, s_k)$ computing $(\prod_{i=1}^{k} s_i)_{\min}$ in (expected) time $O(nd^{k(2n-1)\max\{2,n\}})$ (when $n \leq k$), and (deterministic) time $O((1 + k)d^{k(2n-1)})$ (when $n > k$). Moreover, $(\prod_{i=1}^{k} s_i)_{\min}$ has $O((kd)^{2n-1})$ monomials.*

Proof. We could directly compute the product $\Pi_{i=1}^{k} s_i$ (in time $O(d^{k(2n-1)})$, since Theorem 6.1 gives $|s_i| = O(d^{2n-1})$) and then extract its minimal Newton polytope, which gives, via Theorem

6.6, the first bound. When $n > k$, a speed-up is obtained by using Lemma 6.8: since $s_i = (s_i)_{\min}$, $NP(s_i) = NP_{\min}(s_i)$ and the vertices of $NP(s_i)$ are precisely the terms of s_i, so we can compute in time $O(d^{k(2n-1)})$ the Minkowski sum $NP(\prod_i s_i) = \sum_i NP(s_i)$ (which, again by Theorem 6.1, has $O((kd)^{2n-1})$ terms) and then do a quadratic minimization. This gives the (deterministic) time $O(d^{k(2n-1)} + (kd)^{2(2n-1)})$, leading to the claimed bound. □

Remark that $\mathrm{t}^!(\prod_{i=1}^k s_i)_{\min}(x) = \sum_{i=1}^k (s_i)_{\min}(x)$. By computing products via **VN**, once we fix the number of variables, the size of (tropical) products like (6) grows polynomially in both d and k.

Example 6.10. Consider $M_5 = (\lambda x.x \oplus_X x)(\lambda x.x \oplus_X x)\dots(\lambda x.x \oplus_X x)1$ similar to M_4 from Section 3.2. Each of the 2^n trajectories $M \overset{\mu}{\twoheadrightarrow} 1$ corresponds to a monomial $X^i\overline{X}^{n-i}$ and the sum of all such monomials produces the polynomial (with the same support as) $(X + \overline{X})^n$. Observe that, since all reductions to i have degree n, the tropical degree $\mathfrak{d}_{\mathrm{i}}(M_4) = n$, and we can select *all* the 2^n reductions to witness this fact. But this tells nothing wrt the most likely reductions. By contrast, by the old freshman dream (Example 6.3), the Newton polytope of $(X + \overline{X})^n$ only selects the two monomials $X^n, \overline{X}^n$. One can see that, for all assignment of X to probability $p \in [0, 1]$, the probability of the most likely reductions is always found within the two selected ones (i.e., $\max_{i=0}^n p^i(1-p)^{n-i} = \max\{p^n, (1-p)^n\}$, as one can easily check). In other words, the Newton Polytope (in fact, its minimal version) of (the parametric interpretation of) M_4, drastically reduces the search space for most likely reductions.

7 Tropical Intersection Type System

We now put all the work of the previous section in use for the analysis of probabilistic programs of PCF$\langle\vec{X}\rangle$: we introduce an intersection type system $\mathbf{P}_{\mathrm{trop}}$ that associates terms of PCF$\langle\vec{X}\rangle$ with *minimal* all-one polynomials describing their most-likely reductions. After proving soundness and completeness of $\mathbf{P}_{\mathrm{trop}}$ wrt the parametric WRS semantics, we describe an algorithm that converges onto the Newton polynomial, thus producing an answer to the inference tasks (I1) and (I2).

7.1 The Type System $\mathbf{P}_{\mathrm{trop}}$

Intersection type system have been largely used to capture the termination properties of higher-order programs. *Non-idempotent* (n. i.) intersection type systems, inspired from linear logic, have been shown to capture *quantitative* properties like e.g. the number of reduction steps [1, 11, 18]. In a probabilistic setting, [22] have introduced a n. i. intersection type system **P** for probabilistic PCF which precisely captures the probability that a program M : Bool reduces to, say, 1 in the following sense: for each reduction $M \xrightarrow{p} 1$ one can construct a derivation of the form $\vdash^p_{\mathbf{P}} M : 1$ so that

$$\mathbb{P}(M \twoheadrightarrow 1) = \sum \left\{ w(\pi) \;\middle|\; \pi \text{ is a derivation of } \vdash^p_{\mathbf{P}} M : 1 \text{ of weight } w(\pi) = p \right\}. \tag{10}$$

By replacing the positive real weights $p \in [0, 1]$ in the system **P** with the formal monomials of PCF$\langle\vec{X}\rangle$ one obtains, in a straightforward way, a type system that produces all the monomials μ occurring in a reduction $M \overset{\mu}{\twoheadrightarrow} 1$. In other words, the type system explores *all* possible reductions of M and produces the associated monomial. This provides a way to fully reconstruct the parametric interpretation $[\![M]\!]^{X_1,\dots,X_n} \in \mathbb{N}^\infty\{\!\{\mathbb{X}\}\!\}$ of a term.

Our goal, instead, is to design a type system that explores *multiple* reductions at once, excluding those whose probability is dominated, so as to restrict to a finite set of most likely reductions. The goal is thus to capture a finite polynomial corresponding to the tropicalization $\mathrm{t}^![\![M]\!]^{X_1,\dots,X_n}$ (in accordance with Theorem 5.5). A natural idea is to consider multiple **P**-derivations in parallel. Typically, while in the case of a choice $M \oplus_p N$ a derivation in **P** chooses whether to look at M or

N (that is, it chooses between the two reducts of $M \oplus_p N$), in our system the derivation branches so as to consider (and compare) both possible choices.

However, the feasibility of such a system is far from obvious: through reduction, even a term of small size may give rise to an exponentially large number of trajectories, as shown in the example below. Keeping track of all such trajectories through parallel branches in our type derivations can quickly become intractable (even for a computer-assisted formalization).

As already explained, this is where we exploit the minimal Newton polytope: while the rules of **P** produce the probability by progressively multiplying the monomials obtained at each previous step, considering multiple **P**-derivations at once requires computing formal polynomials by repeatedly multiplying other formal polynomials produced at previous steps. By using the results developed in Section 6, we are able to keep the size of such polynomials under control.

Definition 7.1 ($\mathbf{P}_{\text{trop}}$). The types of $\mathbf{P}_{\text{trop}}$ are defined by the grammar $a := n \in \mathbb{N} \mid [a, \dots a] \multimap a$. A *context* γ is a function from variables to multisets of $\mathbf{P}_{\text{trop}}$-types, all empty except finitely many. We write it as a finite sequence of variable declarations $x : \gamma(x)$ such that $\gamma(y) = []$ for all non-declared y. Given contexts γ, δ, we indicate as $\gamma + \delta$ the context obtained by summing their image variable-wise. A *pre-judgement* is an expression of the form

$$M : \left\langle \gamma_j \vdash^{s_j} a_j \right\rangle_{j \in J}$$

and stands for a finite family of judgements $\gamma_j \vdash^{s_j} M : a_j$, where s_j indicates a formal polynomial in $\mathbb{T}\{\mathbb{X}\}$. A pre-judgement is a *judgement* when the pairs $(\gamma_j, a_j)_{j \in J}$ are *pairwise distinct* and the polynomials s_j are *minimal.* Given a pre-judgement as above, we can always produce a judgement $M : \text{merge} \left\langle \gamma_j \vdash^{s_j} a_j \right\rangle_{j \in J}$ by merging equal typings (e.g. turning $\langle \gamma \vdash^{s} a \mid \gamma \vdash^{s'} a \rangle$ into $\langle \gamma \vdash^{s+s'} a \rangle$) and minimizing each obtained polynomial s via $\mathbf{VN}(s)$. The rules of $\mathbf{P}_{\text{trop}}$ are illustrated in Fig. 4.

Crucially, we design the rules so as to precisely keep track of the reductions selected by the *minimal* Newton Polytope of a compound term, by combining those of its constituent.

Except for the rule $(\emptyset)$, that introduces an empty family of judgements, each rule of $\mathbf{P}_{\text{trop}}$ results from a corresponding rule of **P** by extending it to families of judgements. The rules (n), (id), (S), (P), (λ) are self-explanatory: they correspond to rules that create no new parametric reduction. The rules (ifz), $(\oplus)$, (@) and (Y) deserve some discussion. The rule $(\oplus)$ collects a family of typings of M with polynomials s_i, and a family of typings of N with polynomials s'_j, to produce a family of typings of $M \oplus_X N$, with polynomials $s_i \cdot X$ and $s'_j \cdot \overline{X}$, that is successively merged. Observe that this precisely corresponds to keeping track of the reductions selected by the minimal Newton Polytope of $M \oplus_X N$, by combining those of M and N. The rule (ifz) works in a similar way, but uses $\mathbf{VN}(-)$ also *before* merging, since it needs to compute the possibly non-trivial tropical products $s_0 \cdot t_j$ and $s_{i+1} \cdot t'_j$. The application rule (@) collects, on the one hand, a family of typing $m_i \multimap b_i$ of M with polynomials s_i, where $m_i = [m_{i1}, \dots, m_{ip_i}]$; on the other hand, for each typing $m_i \multimap b_i$, and each type m_{ij} inside m_i, it collects a typing $N : m_{ij}$ with polynomials s'_{ij}. The conclusion of the rule computes minimal polynomials for the types b_i by calling $\mathbf{VN}(s_i, s'_{i1}, \dots, s'_{ip_i})$ to minimise the tropical multiplication $s_i \cdot \prod_j s'_{ij}$. The rule (Y) works in a very similar way.

Example 7.2. In Fig. 5 we illustrate a family π_n of derivations for the term M_3 from Section 2. M_3 admits arbitrary long reductions, the first one being the most likely. π_0 computes the weight of the most likely derivation $M_3 \overset{X}{\twoheadrightarrow} 1$; π_{n+1} compares the weights from all π_i, for $i \leq n$ with the weight of the $n + 1$th reduction, but ends up selecting in each case only the weight from π_0, since $(\sum_n X\overline{X}^n)_{\min} = X$. Hence, all π_n correctly compute the minimal polynomial, providing a correct estimation of the tropical degree $\mathfrak{d}_1(M_3) = 1$ of M_3.

$$\frac{}{M:\emptyset}\ \emptyset \qquad \frac{}{x : \left\langle x:[a_i] \vdash^1 a_i \right\rangle_{i\in I}}\ \mathrm{id} \qquad \frac{}{\mathsf{n} : \left\langle \vdash^1 n \right\rangle_{\{\star\}}}\ \mathrm{n} \qquad \frac{M : \left\langle \gamma_i \vdash^{s_i} n_i \right\rangle_{i\in I}}{\mathsf{succ}\, M : \left\langle \gamma_i \vdash^{s_i} n_i + 1 \right\rangle_{i\in I}}\ \mathrm{S} \qquad \frac{M : \left\langle \gamma_i \vdash^{s_i} n_i \right\rangle_{i\in I}}{\mathsf{pred}\, M : \left\langle \gamma_i \vdash^{s_i} n_i \dot{-} 1 \right\rangle_{i\in I}}\ \mathrm{P}$$

$$\frac{M : \left\langle \gamma_0 \vdash^{s_0} 0 \,\middle|\, \gamma_{i+1} \vdash^{s_{i+1}} i+1 \right\rangle_{i\in I\subset\mathbb{N}} \qquad N : \left\langle \delta_j \vdash^{t_j} a_j \right\rangle_{j\in J_0} \qquad P : \left\langle \delta'_j \vdash^{t'_j} a'_j \right\rangle_{j\in J_1}}{\mathsf{ifz}(M,N,P) : \mathrm{merge}\left\langle \gamma_0 + \delta_j \vdash^{\mathbf{VN}(s_0,t_j)} a_j \,\middle|\, \gamma_{i+1} + \delta'_j \vdash^{\mathbf{VN}(s_{i+1},t'_j)} a'_j \right\rangle_{i\in I, j\in J_0+J_1}}\ \mathrm{ifz}$$

$$\frac{M : \left\langle \gamma_i \vdash^{s_i} a_i \right\rangle_{i\in I} \qquad N : \left\langle \gamma_j \vdash^{s'_j} a_j \right\rangle_{j\in J}}{M \oplus_X N : \mathrm{merge}\left\langle \gamma_i \vdash^{s_i\cdot X} a_i \,\middle|\, \gamma_j \vdash^{s'_j\cdot\overline{X}} a_j \right\rangle_{i\in I, j\in J}}\ \oplus \qquad \frac{M : \left\langle \gamma_i, x : m_i \vdash^{s_i} b_i \right\rangle_{i\in I}}{\lambda x.M : \left\langle \gamma_i \vdash^{s_i} m_i \multimap b_i \right\rangle_{i\in I}}\ \lambda$$

$$\frac{M : \left\langle \gamma_i \vdash^{s_i} m_i \multimap b_i \right\rangle_{i\in I} \qquad N : \left\langle \left\langle \delta_{ij} \vdash^{s'_{ij}} m_{ij} \right\rangle_{j\in J_i} \right\rangle_{i\in I}}{MN : \mathrm{merge}\left\langle \gamma_i + \sum_j \delta_{ij} \vdash^{\mathbf{VN}(s_i, s'_{i1},\dots,s'_{ip_i})} b_i \right\rangle_{i\in I}}\ @ \qquad \frac{M : \left\langle \gamma_i \vdash^{s_i} m_i \multimap b_i \right\rangle_{i\in I} \qquad \mathrm{Y}M : \left\langle \left\langle \delta_{ij} \vdash^{s'_{ij}} m_{ij} \right\rangle_{j\in J_i} \right\rangle_{i\in I}}{\mathrm{Y}M : \mathrm{merge}\left\langle \gamma_i + \sum_j \delta_{ij} \vdash^{\mathbf{VN}(s_i, s'_{i1},\dots,s'_{ip_i})} b_i \right\rangle_{i\in I}}\ \mathrm{Y}$$

Fig. 4. Typing Rules of $\mathbf{P}_{\mathrm{trop}}$. In rules @ and Y, $m_i = [m_{i1}, \dots, m_{ip_i}]$ and $p_i = \mathrm{Card}(J_i)$.

Example 7.3. In Fig. 6 we illustrate a derivation for the term M_4 from Section 3.2, choosing $n = 2$ and $X_1 = X_2$. It computes the reduced polynomial $X^3 + \overline{X}^3$, thus correctly estimating $\mathfrak{d}_1(M_4) = 3$.

The number of families explored in parallel in a derivation is a parameter controlled by the user. For example, in a term $M \oplus_X N$ we can decide whether to explore both branches or only one, and this choice affects the size of the derivation $|\pi|$, that is, the number of rules. Instead, the size of the polynomials obtained through the derivation is not controlled by the user. Thanks to the estimation from Theorem 6.9, though, their size remains polynomial in $|\pi|$. Indeed, the following can be proved by induction on π:

Proposition 7.4. *Let π be a derivation of $M : \langle \Gamma_i \vdash^{s_i} a_i \rangle_{i\in I}$. Then $\max_i\{\deg s_i\} = O(|\pi|)$. As a consequence, $|s_i| \in O(|\pi|^{2n-1})$ for all $i \in I$.*

Given a derivation π of $M : \langle \vdash^s \mathsf{i} \rangle$, by replacing, in each rule (ifz), (@) and (Y), the minimized products $(\Pi_i s_i)_{\min}$ obtained by $\mathbf{VN}(s_1, \dots, s_n)$ with the full products $\Pi_i s_i$, we obtain in the end a *larger* polynomial, that we call $\mathrm{traj}^{\mathsf{i}}(\pi)$. Using Lemma 6.8 one can easily check that:

Proposition 7.5. *For any derivation π of $M : \langle \vdash^s \mathsf{i} \rangle$, $s = (\mathrm{traj}^{\mathsf{i}}(\pi))_{\min}$.*

Intuitively, the polynomial $\mathrm{traj}^{\mathsf{i}}(\pi)$ tracks *all* reductions of M that π explored and out of which it selected the most likely ones. This claim will be justified by the soundness theorem below. For instance, consider Example 7.3, if we replace, in the derivation of Fig. 6, products computed by $\mathbf{VN}$ by standard products, we obtain $\mathrm{traj}^1(\pi) = \sum_{i+j=3} X^i \overline{X}^j$, while $X^3 + \overline{X}^3 = (\mathrm{traj}^1(\pi))_{\min}$.

7.2 Soundness and Completeness of $\mathbf{P}_{\mathrm{trop}}$ for the Parametric WRS

Intuitively, a $\mathbf{P}_{\mathrm{trop}}$-derivation is an optimized way to collect multiple $\mathbf{P}$-derivations, which, in turn, encode the reductions of the underlying term. More precisely, for any choice of probabilities $p \in [0,1]^{\mathbb{X}}$, for any derivation of $M : \langle \Gamma_i \vdash^{s_i} a_i \rangle_{i\in I}$, for each $i \in I$ and for each monomial μ in s_i, there is a derivation of $\Gamma_i \vdash^{p^{\mu}} M[X := p] : a_i$ in $\mathbf{P}$, where $p^{\mu} = \Pi_{V\in\mathbb{X}} p_V^{\mu(V)}$ (cfr. Section 4.1) is the probability of the reduction of $M[X := p]$ (to some normal form) corresponding to μ. This suggests then that soundness and completeness can be *lifted* from $\mathbf{P}$ [22, Lemma 20 and Equation 11] to $\mathbf{P}_{\mathrm{trop}}$.

The fundamental ingredient is the notion of a $\mathbf{P}_{\mathrm{trop}}$-derivation *refining* a $\mathrm{PCF}\langle\vec{X}\rangle$-derivation. First, given a simple type A, a $\mathbf{P}_{\mathrm{trop}}$-type a, a simple context Γ and a $\mathbf{P}_{\mathrm{trop}}$-context γ, we say that (γ, a) refines (Γ, A) whenever $a \in [\![A]\!]$ (so e.g. 0, 1 refine Bool and $m \multimap b$ refines $A \to B$ whenever

$$\pi_0 : \quad \dfrac{\dfrac{\dfrac{x : \emptyset \qquad 1 : \left\langle \vdash^{1} 1 \right\rangle}{x \oplus_X 1 : \left\langle \vdash^{\overline{X}} 1 \right\rangle}}{\lambda x.x \oplus_X 1 : \left\langle \vdash^{\overline{X}} \emptyset \multimap 1 \right\rangle}}{\mathrm{Y}(\lambda x.x \oplus_X 1) : \left\langle \vdash^{\overline{X}} 1 \right\rangle} \qquad\qquad \pi_{n+1} : \quad \dfrac{\dfrac{\dfrac{1 : \left\langle \vdash^{1} 1 \right\rangle \qquad x : \left\langle x : [1] \vdash^{1} 1 \right\rangle}{x \oplus_X 1 : \left\langle \vdash^{\overline{X}} 1 \middle| x : [1] \vdash^{X} 1 \right\rangle}}{\lambda x.x \oplus_X T : \left\langle \vdash^{\overline{X}} \emptyset \multimap 1 \middle| \vdash^{X} [1] \multimap 1 \right\rangle} \qquad \begin{array}{c} \emptyset \big| \pi_n \\ \mathrm{Y}(\lambda x.x \oplus_X 1) : \left\langle \emptyset \middle| \vdash^{\overline{X}} 1 \right\rangle \end{array}}{\mathrm{Y}(\lambda x.x \oplus_X 1) : \left\langle \vdash^{\overline{X}} 1 \right\rangle}$$

Fig. 5. Derivations from Example 7.2.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x : \left\langle x : [[a] \multimap a] \vdash^{1} [a] \multimap a \right\rangle} \quad \dfrac{}{x : \left\langle x : [[a] \multimap a] \vdash^{1} [a] \multimap a \right\rangle}}{x \oplus_X x : \left\langle x : [[a] \multimap a] \vdash^{X+\overline{X}} [a] \multimap a \right\rangle}}{\lambda x.x \oplus_X x : \left\langle \vdash^{X+\overline{X}} [[a] \multimap a] \multimap [a] \multimap a \right\rangle} \quad \dfrac{\dfrac{\dfrac{}{x : \left\langle x : [[1] \multimap 1] \vdash^{1} [1] \multimap 1 \right\rangle} \quad \dfrac{}{x : \left\langle x : [[1] \multimap 1] \vdash^{1} [1] \multimap 1 \right\rangle}}{x \oplus_X x : \left\langle x : [[1] \multimap 1] \vdash^{X+\overline{X}} [1] \multimap 1 \right\rangle}}{\lambda x.x \oplus_X x : \left\langle \vdash^{X+\overline{X}} [[1] \multimap 1] \multimap [1] \multimap 1 \right\rangle}}{(\lambda x.x \oplus_X x)\lambda x.x \oplus_X x : \left\langle \vdash^{X^2+\overline{X}^2} [1] \multimap 1 \right\rangle} \quad \dfrac{\dfrac{\dfrac{}{x : \left\langle x : [1] \vdash^{1} 1 \right\rangle} \quad \dfrac{}{x : \left\langle x : [1] \vdash^{1} 1 \right\rangle}}{x \oplus_X x : \left\langle x : [1] \vdash^{X+\overline{X}} 1 \right\rangle}}{\lambda x.x \oplus_X x : \left\langle \vdash^{X+\overline{X}} [1] \multimap 1 \right\rangle}}{(\lambda x.x \oplus_X x)(\lambda x.x \oplus_X x)\lambda x.x \oplus_X x : \left\langle \vdash^{X^3+\overline{X}^3} [1] \multimap 1 \right\rangle} \quad \dfrac{}{1 : \left\langle \vdash^{1} 1 \right\rangle}}{(\lambda x.x \oplus_X x)(\lambda x.x \oplus_X x)(\lambda x.x \oplus_X x)1 : \left\langle \vdash^{X^3+\overline{X}^3} 1 \right\rangle}$$

Fig. 6. Derivation from Example 7.3, where $a = [1] \multimap 1$.

$m \in ![\![A]\!]$ and $b \in [\![B]\!]$) and $\gamma(x) \in ![\![\Gamma(x)]\!]$ for all x declared in Γ, and $\gamma(x) = []$ otherwise. Now, given a PCF$\langle\vec{X}\rangle$-derivation Π of $\Gamma \vdash M : A$ and a $\mathbf{P}_{\text{trop}}$-derivation π of $M : \left\langle \gamma_k \vdash^{s_k} a_k \right\rangle_k$ with (γ, a) refining (Γ, A), π refines Π when, intuitively, its rules match the corresponding rules in Π: for instance, if Π ends with the abstraction rule of conclusion $\Gamma \vdash M : A \to B$, π ends with the rule (λ) of conclusion $M : \langle \gamma_i \vdash M : m_i \multimap b_i \rangle_{i \in I}$, where $(\gamma_i, m_i \multimap b_i)$ refines $(\Gamma, A \to B)$. Two exceptions are the cast rule (which is skipped) and the Y-rule. For the latter, the intuition is that a $\mathbf{P}_{\text{trop}}$-derivation π for YM actually determines one possible *finite unfolding* of YM. More precisely, π must end with a *cluster* of h consecutive Y-rules of which the last one has premise of type $[] \multimap b_j$. By replacing each such (Y) with (@) one obtains then a $\mathbf{P}_{\text{trop}}$-derivation $\mathbf{unfold}(\pi)$ in which YM has been replaced by the unfolded term $M^{(h)}y$ (y is a fresh variable). Correspondingly, Π can also be transformed into a PCF$\langle\vec{X}\rangle$-derivation $\mathbf{unfold}^h(\Pi)$ of $M^{(h)}y$ by replacing the fixpoint rule with a cluster of h application rules; we say that π refines Π when $\mathbf{unfold}(\pi)$ refines $\mathbf{unfold}^h(\Pi)$. For the definition to make sense, a transfinite induction is required, by treating the Y-rule as a ω-rule.

By adapting the argument for $\mathbf{P}$ [22, Lemma 20], we obtain (by induction on a PCF$\langle\vec{X}\rangle$-derivation):

Theorem 7.6 (Soundness of $\mathbf{P}_{\text{trop}}$ wrt WRS). *Let* Π *a* PCF$\langle\vec{X}\rangle$*-derivation of* $\Gamma \vdash M : A$ *and* (γ, a) *refining* (Γ, A). *For all* $\mathbf{P}_{\text{trop}}$*-derivation* π *of* $M : \langle \gamma \vdash^s a \rangle$ *that refines* Π, $\mathrm{supp}(s) \subseteq \mathrm{supp}([\![M]\!]^{\mathbb{X}}_{\gamma,a})$.

It follows that, in particular, the minimal polynomials produced by typing derivations for a closed ground-type term M produce an over-approximation of the tropicalisation of M:

Corollary 7.7. *Let* Π *derive in* PCF$\langle\vec{X}\rangle$$\vdash M : \mathbf{N}$, $n \in \mathbb{N}$. *For all* π *derivation in* $\mathbf{P}_{\text{trop}}$ *of* $M : \langle \vdash^s n \rangle$ *that refines* Π, *we have* $\mathrm{t}^![\![M]\!]_{\mathsf{n}}(q) \leq s^!(q)$ *for all* $q \in \mathbb{T}^{2n}$ *(n being the number of parameters).*

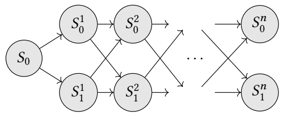
Proof. It follows from taking $\Gamma = \emptyset$, $A = \mathbf{N}$ in the soundness, which gives: for all $n \in \mathbb{N}$ and $q \in \mathbb{T}^{2n}$, $\mathrm{t}^![\![M]\!]^{\mathbb{X}}_{\mathsf{n}}(q) \leq \inf\{\mu \cdot q \mid \mu \in \mathrm{supp}(s)$ *for some* π *of* $M : \langle \vdash^s n \rangle\}$. □

In fact, we can say more: $\mathbf{P}_{\text{trop}}$ captures at least the minimal part of the parametric WRS. Reasoning in a similar way as for Theorem 7.6, one can show:

Theorem 7.8 (Completeness of $\mathbf{P}_{\text{trop}}$ wrt *minimal* WRS). *Let* Π *be a* PCF$\langle\vec{X}\rangle$*-derivation of* $\Gamma \vdash M : A$ *and* (γ, a) *refining* (Γ, A). *For all* $\mu \in \mathrm{supp}([\![M]\!]^{\mathbb{X}}_{\gamma,a})_{\min}$ *there is a* $\mathbf{P}_{\text{trop}}$*-derivation* π *of* $M : \langle \gamma \vdash^s a \rangle$ *that refines* Π *and with* $\mu \in \mathrm{supp}(s)$.

Notice that, together, Theorems 7.6 and 7.8 give the equality

$$\mathrm{t}^![\![M]\!]_{\mathsf{n}}(q) = \inf\{\mu \cdot q \mid \mu \in \mathrm{supp}(s) \textit{ for some } \pi \textit{ of } M : \langle \vdash^s n \rangle\}, \tag{11}$$



(a) Bayesian Network.

$$\dfrac{\overset{\pi_j}{P_j : \langle \vdash^{s_0^j} 0 \mid \vdash^{s_1^j} 1\rangle} \qquad \overset{\pi_{S,j+1,0}}{S_0^{j+1} : \langle \vdash^{u_0^{j+1}} 0 \mid \vdash^{u_1^{j+1}} 1\rangle} \qquad \overset{\pi_{S,j+1,1}}{S_1^{j+1} : \langle \vdash^{v_0^{j+1}} 0 \mid \vdash^{v_1^{j+1}} 1\rangle}}{P_{j+1} : \left\langle \vdash^{\mathbf{VN}(s_0^j \cdot u_0^{j+1} + s_1^j \cdot v_0^{j+1})} 0 \,\middle|\, \vdash^{\mathbf{VN}(s_0^j \cdot u_1^{j+1} + s_1^j \cdot v_1^{j+1})} 1 \right\rangle}\ \mathsf{ifz}$$

(b) Typing derivation π_{j+1} for P_{j+1}.

Fig. 7. Bayesian Network and typing derivation for Example 7.9.

showing that the taf interpreting M can be in principle *approximated* via larger and larger derivations, which we construct below. Actually, coherently with Corollary 5.5, we show in Theorem 7.11 that there is a *single* derivation reaching the inf, which is thus a (non computable, in general) min.

7.3 Constructing a Solution to Tasks (I1) and (I2)

We now exploit the type system $\mathbf{P}_{\text{trop}}$ to design an algorithm that, by reconstructing the Newton polynomial of a term, converges onto a correct solution to the inference tasks (I1) and (I2). Given a PCF$\langle\vec{X}\rangle$-term M : Bool with parameters $\vec{X}$ and a chosen output value $i \in \{0, 1\}$, the goal is to produce in output a finite set $\texttt{selected\_trajs}^i(M)$ of tuples (μ, w_μ, S), where $w_\mu \in \{0, 1\}^*$ is a sequence describing a candidate most likely trajectory $M \overset{\mu}{\twoheadrightarrow} \mathsf{i}$ and $S \subseteq [0, +\infty]^n$ is the set of $(-\log)$-values of the parameters $\vec{X}$ that make μ minimum across *all* $\mu \in \texttt{selected\_trajs}^i(M)$. The algorithm is divided in two phases: the construction of a suitable $\mathbf{P}_{\text{trop}}$-derivation of $M : \langle \vdash^s \mathsf{i}\rangle$, and the extraction of the set $\texttt{selected\_trajs}^i(M)$ from π, described in the following two subsections.

7.3.1 Constructing a Stable Derivation. The main idea for the construction of a derivation exploring the reductions of M was already suggested in Example 7.2: we progressively make π grow so as to explore *more and more* reductions, until the produced polynomial s *stabilizes*: anyhow the derivation may still grow, the polynomial s does not change.

As a first simple example, suppose M is a closed first-order term built using only $0, 1, \oplus_X$ and ifz$(-,-,-)$, so that its typing in PCF$\langle\vec{X}\rangle$ only contains the type Bool. It is not difficult then to construct, by induction on M, a derivation $\pi : \langle \vdash^{s_0} 0 \mid \vdash^{s_1} 1\rangle$ that explores *all* trajectories of M (i.e. for which $\text{supp}(\text{traj}^{\text{i}}(\pi)) = \text{supp}(\llbracket M\rrbracket_i)$). Notice that the derivation has size linear in the size of M, since the index sets I for any judgment can have at most two elements (the only two refinements $0, 1$ of the type Bool). In this simplified setting, as illustrated in the example below, the construction of s can then be seen as a parametric variant of the Viterbi algorithm.

Example 7.9. Consider closed terms $S_0, S_i^1, \dots, S_i^n$: Bool, $i = 0, 1$, $n > 0$ and let $P = P_n$: Bool, where $P_0 = S_0$ and $P_{j+1} = \mathsf{ifz}(P_j, S_0^{j+1}, S_1^{j+1})$. The term P : Bool encodes the graphical model in Fig. 7a. This model has 2^n trajectories leading to either S_0^n, S_1^n. There is a derivation π_n of $P : \langle \vdash^{s_0} 0 \mid \vdash^{s_1} 1\rangle$ (of size linear in P) capturing all such trajectories, made of a chain of ifz-rules (the derivations π_{j+1} for the terms P_j are illustrated in Fig. 7b, supposing given derivations for S_0 and the S_i^j). The polynomials $s_0 := s_0^n$ and $s_1 := s_1^n$ are constructed by choosing, at each step, the best way to expand either s_0^j or s_1^j with the monomials u_l^j, v_l^j coming from either S_0^j or S_1^j. If the u_l^j, v_l^j were scalars, this would indeed correspond to computing a Viterbi sequence for the model in Fig. 7a.

While in the case above it was possible to generate a single derivation encompassing all trajectories of the term, this is not possible in general, because the term may have *infinitely* many trajectories, which is typically the case if the term contains a fixpoint. Another problem is that, if the term has some *higher-order* applications PQ, the number of possible intersection types $m \multimap b$ refining the type $A \longrightarrow B$ of P is also infinite, since $m \in\, !\llbracket A\rrbracket$ may be arbitrarily long.

This is why we construct our derivation incrementally. For two fixed parameters $n, p \in \mathbb{N}$, we can always construct a derivation $\pi^{\mathsf{i}}_{n,p}(M)$ of $M : \langle \vdash^{s} \mathsf{i} \rangle$ that collects *all* **P**-derivations of $\vdash M : \mathsf{i}$ in which the rule (Y) is used at most n times and, for all intersection types $m \multimap b$, m has at most p elements and does not contain atoms $\mathsf{i} > p$. This corresponds to looking at reductions of M in which fixpoints are *unfolded at most n times* and, and in each β-reduction $(\lambda x.P)Q$, the term Q may be *duplicated at most p times*. In our algorithm, we initially set $n = p = 1$ and then progressively increment either n or p until reaching stability.

Remark 7.1 (complexity of $\pi^{\mathsf{i}}_{n,p}(M)$). *In general, $\pi^{\mathsf{i}}_{n,p}(M)$ may not be constructible efficiently (i.e. in polynomial time wrt $|M|$): the rules* (@) *and* (Y) *have arity at most $p+1$, while the corresponding operations on terms are at most binary, yielding a derivation much* larger *than the corresponding* $\mathrm{PCF}\langle\vec{X}\rangle$*-typing, as well as a* longer *computation time due to $p+1$-ary products (as the time for computing multiplications - optimized or not - grows exponentially in the number of factors, cf. Theorem 6.9); moreover, also the index sets I may grow very large, as they vary over all refinings $[a_1, \dots, a_k] \multimap b$, with $k \le p$, of the corresponding types $A \to B$. However, when $p = 1$, $\pi^{\mathsf{i}}_{n,1}(M)$, corresponding to looking to* affine *reductions (i.e. in which terms may be deleted but not duplicated) with at most n* Y*-unfolding, only requires* binary *multiplications* $\mathbf{VN}(s_1 s_2)$ *(computed in time $O(d^{4n+2})$ by Theorem 6.9) and has size $O(n|M|3^t)$, where t is the maximum size of a simple type in M. This bound is obtained by unfolding the subterms* $\mathrm{Y}P$ *as $P^i y$ (so that the sum of all such i does not exceed n), yielding a term of size $O(n|M|)$, observing that the arity of* (@), (Y) *is at most* 2*, and noticing that the index sets I cannot exceed the number of affine refinings of the corresponding type (which are obtained by replacing each atom* Bool, N *by any of* [], [0], [1]*), which is bounded by* 3^t.

Example 7.10. Let us illustrate our algorithm for the term M_2 from Example 2.3. Let π_0 be a derivation of $ND : \langle \vdash^{u_0} 0 \mid \vdash^{u_1} 1 \rangle$, capturing the four reductions of ND to either 0 or 1, where, letting $X^0 = X$ and $X^1 = \overline{X}$, $u_i = X_0 X_1^i + \overline{X_0} X_2^i$. One can similarly construct a derivation π_1 of $B : \langle \vdash^{v_i} [] \multimap [i] \multimap 1 \rangle_{b \in \{0,1\}}$, capturing the two reductions $(\mathrm{Y}B)\mathsf{i} \twoheadrightarrow B(\mathrm{Y}B)\mathsf{i} \twoheadrightarrow 1$ that unfold Y only once, where $v_i = \overline{X_{3+i}}$. We construct $\pi^{1}_{1,1}(M)$ by combining π_0 and π_1 via the Y-rule yielding $M_2 : \langle \vdash^{s} 1 \rangle$, where $s = u_0 v_0 + u_1 v_1$ captures all reductions of M_2 with one Y-unfolding. To construct $\pi^{1}_{n,2}(M)$, we must construct a derivation π_2 of $B : \langle \vdash^{t_{ij}} [[j] \multimap 1] \multimap [i] \multimap 1 \rangle_{i,j \in \{0,1\}}$, where $t_{ij} = X_{3+i} X^{j}_{1+i}$, which tracks all pairs of reductions $(\mathrm{Y}B)\mathsf{i} \twoheadrightarrow B(\mathrm{Y}B)\mathsf{i} \twoheadrightarrow (\mathrm{Y}B)(N\mathsf{i})$, of weight X_{3+i}, and $N\mathsf{i} \twoheadrightarrow \mathsf{j}$, of weight X^{j}_{1+i}. The derivation $\pi^{1}_{2,1}(M)$ of of $M_2 : \langle \vdash^{s} 1 \rangle$, is illustrated in Fig. 8, where now $s = u_0 v_0 + u_1 v_1 + u_0 t_{00} v_0 + u_0 t_{01} v_1 + u_1 t_{10} v_0 + u_1 t_{11} v_1$ captures all reductions of M_2 with at most 2 Y-unfoldings. With $n = 3$, we can iterate the process adding a new Y-rule. Intuitively, this should lead to add to s all monomials $w_{abc} := u_a t_{ab} t_{bc} v_c$, for $a, b, c \in \{0, 1\}$, corresponding to 3 iterations. However, each w_{abc} is dominated by one monomial in s: we have that either $a = b, b = c$ or $a = c$; if $a = b$ holds, then w_{abc} is dominated by $u_a t_{ac} v_c$, and similarly for the other cases. Hence $\pi^{1}_{3,1}(M)$ yields *the same* polynomial as $\pi^{1}_{2,1}(M)$. Moreover, further incrementing either n or p does not make the polynomial change. In fact, we can easily see that the minimal polynomial s produced by $\pi^{1}_{2,1}(M)$ coincides $NP^{1}_{\min}(M_2)$: an arbitrary trajectory of M_2 yields a monomial of the form $w_{a_1 \dots a_{n+1}} = u_{a_1} t_{a_1 a_2} \dots t_{a_n a_{n+1}} v_{a_{n+1}}$, and if $n \ge 2$, then the monomial $w_{a_1 \dots a_{n+1}}$ is dominated by some monomial in s. As anticipated in Section 3, it follows then that $\mathfrak{d}^1(M_2) = \deg(s) = 5$.

The result below shows that the algorithm above stabilizes onto the minimal Newton polynomial.

Theorem 7.11. *For any* $\mathrm{PCF}\langle\vec{X}\rangle$*-term M* : Bool *and* $\mathsf{i} \in \{0, 1\}$*, there exists $n, p \in \mathbb{N}$ such that the derivation $\pi^{\mathrm{stab}} := \pi^{\mathsf{i}}_{n,p}(M)$ is stable and proves $M : \langle \vdash^{s} \mathsf{i} \rangle$, where $s = (\mathsf{t}[\![M]\!]^{\mathbb{X}}_{\mathsf{i}})_{\min} = NP^{\mathsf{i}}_{\min}(M)$.*

$$\dfrac{\dfrac{\overset{\pi_1+\pi_2}{B : \left\langle \vdash^{v_i} [] \multimap [i] \multimap 1 \,\middle|\, \vdash^{t_{ij}} [[j] \multimap 1] \multimap [i] \multimap 1 \right\rangle_{i,j\in\{0,1\}}} \qquad \dfrac{\overset{\pi_1}{B : \left\langle \vdash^{v_j} [] \multimap [j] \multimap 1 \right\rangle_{j\in\{0,1\}}}}{\mathrm{Y}B : \left\langle \emptyset \,\middle|\, \vdash^{v_j} [j] \multimap 1 \right\rangle_{j\in\{0,1\}}}\,\mathrm{Y}}{\mathrm{Y}B : \left\langle \vdash^{v_i+t_{i0}v_0+t_{i1}v_1} [\mathtt{i}] \multimap 1 \right\rangle_{i\in\{0,1\}}}\,\mathrm{Y} \qquad \overset{\pi_0}{ND : \left\langle \vdash^{u_0} 0 \,\middle|\, \vdash^{u_1} 1 \right\rangle}}{M_2 : \left\langle \vdash^{u_0v_0+u_1v_1+u_0t_{00}v_0+u_0t_{01}v_1+u_1t_{10}v_0+u_1t_{11}v_1} 1 \right\rangle}\,@$$

Fig. 8. Derivation $\pi^{\mathrm{stab}} = \pi^1_{2,1}(M_2)$, where $u_i = X_0X_1^i + \overline{X_0}X_2^i$, $v_i = \overline{X}_{3+i}$ and $t_{ij} = X_{3+i}\overline{X}^j_{1+i}$.

Proof. By Corollary 5.5 $\mathrm{supp}(\llbracket M\rrbracket^{\mathbb{X}}_{\mathtt{i}})_{\min}$ is finite and by Theorem 7.8 all its points are reached by some $\mathbf{P}_{\mathrm{trop}}$-derivations. We can then set n and p to be larger than the corresponding numbers of Y-unfoldings and multiset sizes in all such (finitely many) derivations. □

Stabilization at n, p is a Π^0_1 (i.e. non-recursive) property: it means that *for any* n', p' larger than n, p, the produced polynomial does not change. Therefore we might not be able to tell *when* the algorithm did actually stabilize. Recall that, by Theorem 5.9, we cannot hope to compute $NP^{\mathtt{i}}_{\min}(M)$ in all situations (even though we always compute $(\mathrm{traj}^i(\pi))_{\min}$, by Proposition 7.5). In practice, we can let the proof-search algorithm terminate after the polynomial has remained stable for some fixed number of iterations (for instance, one stable iteration was enough to reach $NP^{\mathtt{i}}_{\min}(M)$ in Examples 7.2 and 7.10). Let $\tilde{\pi}^{\mathrm{stab}}$ be the derivation obtained after a finite number of stable iterations.

Remark 7.2 (reaching efficiency in the affine case). *As we observed, when the most-likely reductions of M are affine, we can reach π^{stab} via the derivations $\pi^{\mathtt{i}}_{n,1}(M)$, which have size $O(n|M|3^t)$. This is indeed the situation underlying our examples 7.2, 7.9 and 7.10. By contrast, any affine reduction of M can make at most $|M|$ probabilistic choices (since any choice strictly decreases the size of the term), so the trajectory space explored by each such derivation has in general size $O(2^{|M|})$.*

7.3.2 *Extracting a (Partial) Solution from $\tilde{\pi}^{\mathrm{stab}}$.*

We now show how the set $\mathtt{selected\_trajs}^i(M)$ is extracted from a (candidate) stable derivation.

By a straightforward adaptation of the algorithm **VN** (cf. Remark 2.2) and of the $\mathbf{P}_{\mathrm{trop}}$-rules we can perform a *traceback* of s, i.e. keep track, at each step of the construction of $\tilde{\pi}^{\mathrm{stab}}$, of a word $w_\mu \in \{0,1\}^*$, describing the sequence of probabilistic choices made during a reduction $M \overset{\mu}{\twoheadrightarrow} \mathtt{i}$; the word w_μ then traces back one *most likely explanation*. Moreover, for any monomial μ in s, we can compute the *normal cone* of (the convex polytope generated by) s at vertex μ, defined as $\mathcal{N}(\mu; s) = \{x \mid x \cdot \mu = \min_{\nu\in s} x \cdot \nu\}$, see [28, p. 193]. Computing $\mathcal{N}(\mu; s)$ corresponds to solving the linear system of inequalities $\{(\mu - \nu)x \leq 0\}_{\nu\in s}$, which can be done via linear programming (cf. [51]). Observe that $\mathcal{N}(\mu; s)$ precisely captures the set of ($-\ln$-)values that maximize μ across all vertices in s, that is, that make the pair (μ, w_μ) a most likely explanation across those in s.

We can then let $\mathtt{selected\_trajs}^i(M) = \{(\mu, w_\mu, \mathcal{N}(\mu; s)) \mid \mu \text{ in } s\}$. The following holds:

Theorem 7.12. (1) *the pairs (μ, w_μ) in $\mathtt{selected\_trajs}^i(M)$ identify the most likely trajectories in* $\mathrm{traj}^i(\tilde{\pi}^{\mathrm{stab}})$ *(resp. the most likely trajectories $M \twoheadrightarrow \mathtt{i}$ in case $\tilde{\pi}^{\mathrm{stab}} = \pi^{\mathrm{stab}}$).*

(2) *the sets $\mathcal{N}(\mu; s)$ in $\mathtt{selected\_trajs}^i(M)$ identify all ($-\ln$)-values of the parameters $\vec{X}$ minimizing μ across all* $\mathrm{traj}^i(\tilde{\pi}^{\mathrm{stab}})$ *(resp. across all trajectories $M \twoheadrightarrow \mathtt{i}$ in case $\tilde{\pi}^{\mathrm{stab}} = \pi^{\mathrm{stab}}$).*

Whenever $\tilde{\pi}^{\mathrm{stab}} = \pi^{\mathrm{stab}}$, i.e. the proof search reached the Newton polynomial, the result above states that $\mathtt{selected\_trajs}^i(M)$ provides a correct answer to both (I1) and (I2). Otherwise, it provides an answer to (I1) and (I2) only *relatively* to the trajectories explored by $\tilde{\pi}^{\mathrm{stab}} = \pi^{\mathtt{i}}_{n,p}(M)$. This means that any selected trajectory could still be dominated by some intuitively *larger* one, i.e. one requiring either more than n Y-unfoldings or β-reductions performing more than p duplications.

8 Conclusion

Related Work. A growing literature has explored foundational approaches to graphical probabilistic models and higher-order languages for them, both from a categorical [15, 30, 31, 48, 48] and from a more type-theoretic perspective [23]. Methods for statistical inference based on tropical polynomials and the Newton polytope, in the line of Section 4, have been explored for several types of graphical probabilistic models, including HMM and Boltzmann machines [14, 38, 45, 46, 49]. Tropical geometry has also been applied to the study of deep neural networks with ReLU activation functions [12, 38, 53], as well as to piecewise linear regression [39]. The interpretation of probabilistic PCF in the weighted relational model of linear logic is well-studied. The fully abstract model of probabilistic coherent spaces [21] relies on it. Tropical variants of this semantics are studied first in [35], and more recently in [6]. Beyond the one from [21], many other kinds of intersection type systems to capture probabilistic properties have been proposed, e.g. [3, 10, 29].

Future Work. In this paper we demonstrated the potential of combining methods from programming language theory (especially, weighted relational semantics) and tropical geometry, by applying them to study most likely behaviours of probabilistic higher-order programs. We are convinced that this interaction leads to many other interesting applications in programming language theory.

For instance, in this paper we only manipulated the tropicalization by means of the *trivial* valuation $\mathrm{val}_0 : \mathbb{N}^\infty \to \mathbb{T}$ that sends 0 to ∞ and all coefficients $n > 0$ to 0. This was enough to study the most probable outcomes, but considering other valuations may lead to capture different properties of probabilistic programs. For example, $\mathrm{val}_c : \mathbb{R}_{\geq 0} \to \overline{\mathbb{R}}$ defined by $\mathrm{val}_c(x) = -c \ln x$ yields an exciting connection with *differential privacy* [19] (already studied from the perspective of programming languages [5, 17, 24]), that we are currently exploring: for a function $f : \mathrm{db} \to \mathrm{Dist}(X)$ (corresponding to some probabilistic protocol), it is not difficult to see that f is ϵ-differentially private precisely when the function $\mathrm{val}_c \circ f : \mathrm{db} \to [X \to \overline{\mathbb{R}}]$ is ϵ-Lipschitz continuous, taking the Euclidean distance on $\overline{\mathbb{R}}$. As tropical polynomials are always Lipschitz-continuous (they are piecewise linear functions), this suggests that a program with *finite tropical degree* (w.r.t. the valuation val_c, not val_0 as in this work) could be proved differentially private.

Another important future work is to *implement* our type system, in order to mechanise, in interaction with the user, some inference tasks over programs as explained in Section 7. Related to that, we notice that the crucial notion of minimal monomial in a (Newton) polytope, that we develop in Section 6, is reminiscent of that of Gröbner basis in computational algebra. Similarly, we wonder whether the algebra of generating functions – essentially, the formal power series given by our parametric interpretation – reflects computational properties of programs.

Beyond that, there are other natural areas of applications. For instance, [6] illustrated a notion of differentiation for tropical power series, relying on the theory of cartesian differential categories [9, 37], that aligns with existing notions in the literature on tropical differential equations [27]. Finally, the growing interest towards higher-order frameworks for automatic differentiation [41, 50] suggests to look at the tropical methods currently employed for ReLU neural networks [25, 38].

Acknowledgments

The authors would like to greatly thank the anonymous reviewers for their careful reading and for several questions that helped us significantly improve the first version of the paper.

Davide Barbarossa has been funded by the EPSRC grant number EP/W035847/1. Paolo Pistone has been funded by the ANR grant number ANR-23-CPJ1-0054-01. For the purpose of Open Access the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

References

[1] Beniamino Accattoli, Stéphane Graham-Lengrand, and Delia Kesner. 2018. Tight typings and split bounds. *Proc. ACM Program. Lang.* 2, ICFP, Article 94 (July 2018), 30 pages. https://doi.org/10.1145/3236789

[2] S.M. Aji and R.J. McEliece. 2000. The generalized distributive law. *IEEE Transactions on Information Theory* 46, 2 (2000), 325–343. https://doi.org/10.1109/18.825794

[3] Melissa Antonelli, Ugo Dal Lago, and Paolo Pistone. 2022. Curry and Howard Meet Borel. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science* (Haifa, Israel) *(LICS '22)*. Association for Computing Machinery, New York, NY, USA, Article 45, 13 pages. https://doi.org/10.1145/3531130.3533361

[4] Victor Arrial, Giulio Guerrieri, and Delia Kesner. 2023. Quantitative Inhabitation for Different Lambda Calculi in a Unifying Framework. *Proc. ACM Program. Lang.* 7, POPL, Article 51 (Jan. 2023), 31 pages. https://doi.org/10.1145/3571244

[5] Arthur Azevedo de Amorim, Marco Gaboardi, Justin Hsu, Shin-ya Katsumata, and Ikram Cherigui. 2017. A Semantic Account of Metric Preservation. In *Proceedings POPL 2017* (Paris, France). Association for Computing Machinery, New York, NY, USA, 545–556. https://doi.org/10.1145/3009837.3009890

[6] Davide Barbarossa and Paolo Pistone. 2024. Tropical Mathematics and the Lambda-Calculus I: Metric and Differential Analysis of Effectful Programs. In *32nd EACSL Annual Conference on Computer Science Logic (CSL 2024) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 288)*, Aniello Murano and Alexandra Silva (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 14:1–14:23. https://doi.org/10.4230/LIPIcs.CSL.2024.14

[7] Davide Barbarossa and Paolo Pistone. 2025. Tropical Mathematics and the Lambda-Calculus II: Tropical Geometry of Probabilistic Programming Languages (Extended Version). (2025). https://arxiv.org/abs/2501.15637.

[8] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. 2008. *Computational Geometry: Algorithms and Applications* (3rd ed.). Springer-Verlag TELOS, Santa Clara, CA, USA. https://doi.org/10.1007/978-3-540-77974-2

[9] Richard F. Blute, Robin Cockett, and R.A.G. Seely. 2009. Cartesian Differential Categories. *Theory and Applications of Categories* 22, 23 (2009), 622–672.

[10] Flavien Breuvart and Ugo Dal Lago. 2018. On Intersection Types and Probabilistic Lambda Calculi. In *Proceedings PPDP 2018* (Frankfurt am Main, Germany) *(PPDP '18)*. Association for Computing Machinery, New York, NY, USA, Article 8, 13 pages. https://doi.org/10.1145/3236950.3236968

[11] Antonio Bucciarelli, Delia Kesner, and Daniel Ventura. 2017. Non-idempotent intersection types for the Lambda-Calculus. *Logic Journal of the IGPL* 25, 4 (2017), 431–464. https://doi.org/10.1093/jigpal/jzx018

[12] Vasileios Charisopoulos and Petros Maragos. 2017. Morphological Perceptrons: Geometry and Training Algorithms. In *Mathematical Morphology and Its Applications to Signal and Image Processing*, Jesús Angulo, Santiago Velasco-Forero, and Fernand Meyer (Eds.). Springer International Publishing, Cham, 3–15. https://doi.org/10.1007/978-3-319-57240-6_1

[13] Pierre Clairambault and Simon Forest. 2024. An Analysis of Symmetry in Quantitative Semantics. In *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2024, Tallinn, Estonia, July 8-11, 2024*, Pawel Sobocinski, Ugo Dal Lago, and Javier Esparza (Eds.). Association for Computing Machinery, New York, NY, USA, 26:1–26:13. https://doi.org/10.1145/3661814.3662092

[14] Maria Angelica Cueto, Jason Morton, and Bernd Sturmfels. 2010. Geometry of the restricted Boltzmann machine. *Algebraic Methods in Statistics and Probability* 516, 93 (2010), 135–153. https://doi.org/10.1090/conm/516/10172

[15] Fredrik Dahlqvist, Alexandra Silva, Vincent Danos, and Ilias Garnier. 2018. Borel Kernels and their Approximation, Categorically. *Electronic Notes in Theoretical Computer Science* 341 (2018), 91–119. https://doi.org/10.1016/j.entcs.2018.11.006 Proceedings of the Thirty-Fourth Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXIV).

[16] Sandip Das, Subhadeep Ranjan Dev, and Swami Sarvottamananda. 2021. A Worst-Case Optimal Algorithm to Compute the Minkowski Sum of Convex Polytopes. In *Algorithms and Discrete Applied Mathematics*, Apurva Mudgal and C. R. Subramanian (Eds.). Springer International Publishing, Cham, 179–195. https://doi.org/10.1007/978-3-030-67899-9_14

[17] Arthur Azevedo de Amorim, Marco Gaboardi, Justin Hsu, and Shin-ya Katsumata. 2019. Probabilistic Relational Reasoning via Metrics. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 1–19. https://doi.org/10.1109/LICS.2019.8785715

[18] Daniel de Carvalho. 2018. Execution time of λ-terms via denotational semantics and intersection types. *Mathematical Structures in Computer Science* 28, 7 (2018), 1169–1203. https://doi.org/10.1017/S0960129516000396

[19] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3–4 (Aug. 2014), 211–407. https://doi.org/10.1561/0400000042

[20] Thomas Ehrhard, Michele Pagani, and Christine Tasson. 2017. Measurable cones and stable, measurable functions: a model for probabilistic higher-order programming. *Proc. ACM Program. Lang.* 2, POPL, Article 59 (Dec. 2017), 28 pages. https://doi.org/10.1145/3158147

[21] Thomas Ehrhard, Michele Pagani, and Christine Tasson. 2018. Full Abstraction for Probabilistic PCF. *J. ACM* 65, 4, Article 23 (2018), 44 pages. https://doi.org/10.1145/3164540

[22] Thomas Ehrhard, Christine Tasson, and Michele Pagani. 2014. Probabilistic coherence spaces are fully abstract for probabilistic PCF. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (San Diego, California, USA) *(POPL '14)*. Association for Computing Machinery, New York, NY, USA, 309–320. https://doi.org/10.1145/2535838.2535865

[23] Claudia Faggian, Daniele Pautasso, and Gabriele Vanoni. 2024. Higher Order Bayesian Networks, Exactly. *Proc. ACM Program. Lang.* 8, POPL, Article 84 (Jan. 2024), 33 pages. https://doi.org/10.1145/3632926

[24] Marco Gaboardi, Andreas Haeberlen, Justin Hsu, Arjun Narayan, and Benjamin C. Pierce. 2013. Linear Dependent Types for Differential Privacy. *SIGPLAN Not.* 48, 1 (jan 2013), 357–370. https://doi.org/10.1145/2480359.2429113

[25] Jeffrey Giansiracusa and Stefano Mereta. 2024. A general framework for tropical differential equations. *manuscripta mathematica* 173, 3 (2024), 1273–1304. https://doi.org/10.1007/s00229-023-01492-5

[26] Andrew D. Gordon, Thomas A. Henzinger, Aditya V. Nori, and Sriram K. Rajamani. 2014. Probabilistic programming. In *Future of Software Engineering Proceedings* (Hyderabad, India) *(FOSE 2014)*. Association for Computing Machinery, New York, NY, USA, 167–181. https://doi.org/10.1145/2593882.2593900

[27] Dima Grigoriev. 2017. Tropical differential equations. *Advances in Applied Mathematics* 82 (2017), 120–128. https://doi.org/10.1016/j.aam.2016.08.002

[28] M. Ziegler Günter. 1995. *Lectures on Polytopes.* Graduate Texts in Mathematics, Vol. 152. Springer-Verlag, New York, NY, USA. https://doi.org/10.1007/978-1-4613-8431-1

[29] Willem Heijltjes and Georgina Majury. 2025. Simple Types for Probabilistic Termination. In *33rd EACSL Annual Conference on Computer Science Logic (CSL 2025) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 326)*, Jörg Endrullis and Sylvain Schmitz (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 31:1–31:21. https://doi.org/10.4230/LIPIcs.CSL.2025.31

[30] Chris Heunen, Ohad Kammar, Sam Staton, and Hongseok Yang. 2017. A convenient category for higher-order probability theory. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. Asociation for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1109/LICS.2017.8005137

[31] Bart Jacobs and Fabio Zanasi. 2020. The Logical Essentials of Bayesian Reasoning. In *Foundations of Probabilistic Programming*, Gilles Barthe, Joost-Pieter Katoen, and Alexandra Silva (Eds.). Cambridge University Press, Cambridge, UK, 295–332. https://doi.org/10.1017/9781108770750.010

[32] Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning.* The MIT Press, Cambridge, Massachusetts.

[33] James Laird. 2016. Weighted Relational Models for Mobility. In *1st International Conference on Formal Structures for Computation and Deduction, FSCD 2016, June 22-26, 2016, Porto, Portugal (LIPIcs, Vol. 52)*, Delia Kesner and Brigitte Pientka (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 24:1–24:15. https://doi.org/10.4230/LIPICS.FSCD.2016.24

[34] James Laird. 2020. Weighted models for higher-order computation. *Inf. Comput.* 275 (2020), 104645. https://doi.org/10.1016/J.IC.2020.104645

[35] Jim Laird, Giulio Manzonetto, Guy McCusker, and Michele Pagani. 2013. Weighted Relational Models of Typed Lambda-Calculi. In *2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science.* Asociation for Computing Machinery, New York, NY, USA, 301–310. https://doi.org/10.1109/LICS.2013.36

[36] Diane Maclagan and Bernd Sturmfels. 2015. *Introduction to tropical geometry.* Graduate Studies in Mathematics, Vol. 161. American Mathematical Society, Providence, Rhode Island.

[37] Giulio Manzonetto. 2012. What is a categorical model of the differential and the resource λ-calculi? *Mathematical Structures in Computer Science* 22, 3 (2012), 451–520. https://doi.org/DOI:10.1017/S0960129511000594

[38] Petros Maragos, Vasileios Charisopoulos, and Emmanouil Theodosis. 2021. Tropical Geometry and Machine Learning. *Proc. IEEE* 109, 5 (2021), 728–755. https://doi.org/10.1109/JPROC.2021.3065238

[39] Petros Maragos and Emmanouil Theodosis. 2020. Multivariate Tropical Regression and Piecewise-Linear Surface Fitting. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE Computer Society, 3822–3826. https://doi.org/10.1109/ICASSP40776.2020.9054058

[40] Simone Martini. 1992. Categorical models for non-extensional λ-calculi and combinatory logic. *Math. Struct. Comput. Sci.* 2, 3 (1992), 327–357. https://doi.org/10.1017/S096012950000150X

[41] Damiano Mazza and Michele Pagani. 2021. Automatic differentiation in PCF. *Proc. ACM Program. Lang.* 5, POPL (2021), 1–27. https://doi.org/10.1145/3434309

[42] Peter McMullen. 1989. The polytope algebra. *Advances in Mathematics* 78, 1 (1989), 76–130. https://doi.org/10.1016/0001-8708(89)90029-7

[43] Gian Maria Negri Porzio, Vanni Noferini, and Leonardo Robol. 2021. Tropical Laurent series, their tropical roots, and localization results for the eigenvalues of nonlinear matrix functions. (2021). https://arxiv.org/abs/2107.07982.

[44] Vanni Noferini, Meisam Sharify, and Françoise Tisseur. 2015. Tropical Roots as Approximations to Eigenvalues of Matrix Polynomials. *SIAM J. Matrix Anal. Appl.* 36, 1 (jan 2015), 138–157. https://doi.org/10.1137/14096637X

[45] Lior Pachter and Bernd Sturmfels. 2004. Parametric inference for biological sequence analysis. *Proc Natl Acad Sci U S A* 101, 46 (Nov 2004), 16138–16143. https://doi.org/10.1073/pnas.0406011101

[46] Lior Pachter and Bernd Sturmfels. 2004. Tropical geometry of statistical models. *Proceedings of the National Academy of Sciences* 101, 46 (2023/01/16 2004), 16132–16137. https://doi.org/10.1073/pnas.0406010101

[47] Michele Pagani, Peter Selinger, and Benoît Valiron. 2014. Applying quantitative semantics to higher-order quantum computing. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (San Diego, California, USA) *(POPL '14)*. Association for Computing Machinery, New York, NY, USA, 647–658. https://doi.org/10.1145/2535838.2535879

[48] Adam Scibior, Ohad Kammar, Matthijs Vákár, Sam Staton, Hongseok Yang, Yufei Cai, Klaus Ostermann, Sean K. Moss, Chris Heunen, and Zoubin Ghahramani. 2017. Denotational validation of higher-order Bayesian inference. *Proc. ACM Program. Lang.* 2, POPL, Article 60 (Dec. 2017), 29 pages. https://doi.org/10.1145/3158148

[49] Emmanouil Theodosis and Petros Maragos. 2018. Analysis of the Viterbi Algorithm Using Tropical Algebra and Geometry. In *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. 1–5. https://doi.org/10.1109/SPAWC.2018.8445777

[50] Matthijs Vákár and Tom Smeding. 2022. CHAD: Combinatory Homomorphic Automatic Differentiation. *ACM Trans. Program. Lang. Syst.* 44, 3 (2022), 20:1–20:49. https://doi.org/10.1145/3527634

[51] Wayne L. Winston. 1988. *Operations research: Applications and algorithms*. Vol. 18. Duxbury press, Boston. https://doi.org/10.1002/net.3230180310

[52] Henk Wymeersch. 2007. *Factor graphs and the sum–product algorithm*. Cambridge University Press, Cambridge, UK, 35–76. https://doi.org/10.1017/CBO9780511619199.006

[53] Liwen Zhang, Gregory Naitzat, and Lek-Heng Lim. 2018. Tropical Geometry of Deep Neural Networks. In *Proceedings ICML 2018 (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 5819–5827. http://proceedings.mlr.press/v80/zhang18i.html

Received 2025-07-10; accepted 2025-11-06