Checking trustworthiness of probabilistic computations in a typed natural deduction system

F. A. D'ASARO, *LUCI Lab, Department of Philosophy, University of Milan, 20122, Milan, Italy.*
Department of Human Sciences, University of Verona, 37129, Verona, Italy.
E-mail: fabioaurelio.dasaro@univr.it

F. A. GENCO, *LUCI Lab, Department of Philosophy, University of Milan, 20122, Milan, Italy.*
Center for Logic, Language and Cognition, University of Turin, 10124, Turin, Italy.

G. PRIMIERO, *LUCI Lab, Department of Philosophy, University of Milan, 20122, Milan, Italy.*

Abstract

In this paper we present the probabilistic typed natural deduction calculus TPTND, designed to reason about and derive trustworthiness properties of probabilistic computational processes, like those underlying current AI applications. Derivability in TPTND is interpreted as the process of extracting n samples of possibly complex outputs with a certain frequency from a given categorical distribution. We formalize trust for such outputs as a form of hypothesis testing on the distance between such frequency and the intended probability. The main advantage of the calculus is to render such notion of trustworthiness checkable. We present a computational semantics for the terms over which we reason and then the semantics of TPTND, where logical operators as well as a Trust operator are defined through introduction and elimination rules. We illustrate structural and metatheoretical properties, with particular focus on the ability to establish under which term evaluations and logical rules applications the notion of trustworthiness can be preserved.

1 Introduction

The extensive use of AI techniques in decision-making systems is significantly increasing the need for verification of the safety and reliability properties of probabilistic computations. The possibility of formal verification of such properties would grant the ability to consider such systems trustworthy. Nowadays, several approaches to verification of AI systems are emerging, see [41, 43] for overviews. Notably, approaches are focusing on model-checking techniques for safety, liveness and fairness properties, see e.g. [2, 37], program analysis and synthesis, see e.g. [22] or proof-checkers see e.g. [15, 26] for their increasing use also at the industrial level. In the technical literature, though, little is available on the formalization of a notion of trustworthiness itself in this context. Current logical approaches to computational trust in general are overviewed in Section 2.

In this paper, we introduce a derivation system dubbed *Trustworthy Probabilistic Typed Natural Deduction* (TPTND for short). The aim of this system is to formalize the task of inferential reasoning *about* probabilistic computational processes, in particular about their trustworthiness. We consider

https://doi.org/10.1093/logcom/exaf003

samples of such processes with corresponding frequencies and reason about their distance from corresponding theoretical probabilities. We start by defining an operational semantics to consider a probability space in which such computations can be thought to be evaluated. This operational semantics defines events whose results have a certain probability of occurring, and generalizes to samples of such events with frequencies of observed outputs. We then transform such terms under operations for update, conjunction, disjunction and dependency of outputs. A full probabilistic λ-calculus for formalizing experiments consisting of several executions of a program and for evaluating trust is introduced in [24]. This λ-calculus does not feature, however, any other primitive operation on probabilistic events.

In the language of TPTND, reasoning about such probabilistic computations is given in the form of logical rules for deriving theoretical probabilities, expected probabilities and frequencies. Accordingly, judgements of our language are sequents of the form $\Gamma \vdash \phi$ where Γ is a *context*, that is, a set of *assumptions* on random variables, and ϕ is a typed formula in one of the following forms:

- $x : \alpha_a$, declaring that the random variable x has been assigned value α with theoretical probability a;
- $t_n : \alpha_{\tilde{a}}$, declaring that a is the expected probability of obtaining the value α in a sample of n trials on the process t;
- $t_n : \alpha_f$, declaring that f is the frequency with which the value α has been obtained during a series of n trials on the process t.

Both an element of a context and a derived expression may express a statement about a random variable. The derivability relation $\vdash$ expresses dependency between probabilities. For example, the judgement

$$x : \alpha \vdash y : \beta_{0.1}$$

means that the random variable y has probability 0.1 of producing output β, provided x has output α. Under a context interpreted as representing a probability distribution, a derived formula expresses a sample extracted from such distribution. The type of the derived formula is then decorated with an expected probability and the term with a sample size. For example, the judgement

$$\Gamma \vdash toss_{100} : Heads_{\widetilde{0.35}}$$

may be interpreted as the statement saying that, under distribution Γ, after tossing a coin 100 times, one expects output *Heads* 35% of the times. Γ specifies the value assigned to the output of each random variable of interest in the context of an experiment consisting of coin tossing trials[1] (the possible outputs are thus *Heads* and *Tails*). Finally, we can also express any concrete experiment consisting of a series of trials by displaying the frequency of a particular output during the experiment. For instance,

$$\Gamma \vdash toss_{100} : Heads_{30/100},$$

which says that over 100 tosses of our coin, the output *Heads* was obtained 30 times.

The calculus defines also rules for Bayesian update and trust evaluation, where judgements occurring in the rule may include both probabilities and frequencies. In our example above, the theoretical probability of the coin assumed to be fair may be updated after a trial of n experiments, empirically run by throwing the coin. And under the assumption that the coin is fair (or biased),

[1] We talk of an *experiment* simply in order to refer to a series of trials on a particular process. A *trial* on a process, in turn, is just an execution of the process.

one may want to judge the trustworthiness of the process at hand, e.g. we may ask whether the *toss* process follows a fair distribution over the outcomes *Heads* and *Tails*. For example, assuming that Γ expresses the distribution corresponding to a fair coin c as the possible output declarations of one random variable $\Gamma = \{x_c : Heads_{0.5}, x_c : Tails_{0.5}\}$, the process *toss* should be considered trustworthy if, by increasing the number n of trials on it, the probability of output *Heads* gets arbitrarily closer to 0.5 in the limit, and in particular when the displayed frequency of *Heads* falls within a given confidence interval determined by an acceptable error rate. Note that the judgement above can also be formulated when the distribution denoted by Γ is unknown, i.e. when one does not know whether the tossed coin is fair or biased, and the frequency value of the output *Heads* is just empirically observed. Generalizing from this toy example, any probabilistic computation can be checked for trustworthiness in this sense. Therefore, our system can be employed to verify how much we should trust, for instance, a classifier to produce the correct output.

Meta-theoretical properties are formulated considering the computational transformations of empirical tests and their frequencies according to logical operations. In particular, we show which syntactic transformations of terms as they are denoted by inferences of the system

1. do not permit to infer unintended outputs for the processes; and
2. allow to infer whether the observed computation is trustworthy, and under which inferential steps such trustworthiness can be preserved.

We consider this system a useful step towards the formal design and possible automation of reasoning about probabilistic computations like those in use in Machine Learning techniques, as well as for the development of proof-checking protocols on such computational systems. In the following, we show the internal workings of TPTND by the use of simple examples, like dice rolling and coin tossing. However, we also hint at concrete applications using a more realistic example inspired by gender-bias verification.

The rest of this paper is structured as follows. In Section 2 we overview the literature on proof-theoretic formal verification of probabilistic systems and computational trust. In Section 3.1 we introduce a computational semantics defining how atomic processes of interest behave from an operational perspective and their closure under logical operations. In Section 3 we introduce our system TPTND, first through its syntax (Section 3.2), then through its inferential rules to reason: first about theoretical probabilities (Section 3.5); then about processes or computations executed under given probability distributions (Section 3.6). We further show how to model Bayesian Updating of random variables in TPTND (Section 3.7) and finally we introduce the inferential engine to reason about trust on the computational processes of interest (Section 3.8) and to derive structural properties on the relevant probability distributions (Section 3.9). In Section 4 we prove the main meta-theoretical results about TPTND, aiming at showing that given a semantics of well-behaving terms in a probabilistic space, the derivability relation of our system guarantees that outputs obtained with sufficient level of trustworthiness are preserved, and accordingly side-effects are avoided. We conclude in Section 5 indicating further steps of this research.

2 Related work

The present work offers an example of a formal verification method for trustworthy probabilistic computations as implemented for example by machine-learning systems. This area of research, focusing on the modeling of probabilistic systems and reasoning over satisfiability of properties of relevance is yet in its infancy, see [43]. Moreover, it mainly refers to the development of computational, temporal and hybrid logics as formal methods to prove trustworthy properties of

interest, see e.g. [5, 29, 40]; or with linear temporal logic properties defined over Markov decision processes, e.g. with reinforcement learning methods [23] or with imprecise probabilities [39].

In this context, the logic introduced in this paper offers a novel combination of methods and approaches. First of all, it refers to the design of inferential systems of the family of natural deduction systems, sequent calculi and typed λ-calculi, extended with probabilities. These systems and the associated formal proof-verification methods have so far been only little explored in their ability to check trustworthiness of probabilistic computational systems. Secondly, we consider trustworthiness as a property that can itself be expressed and formalized explicitly in the calculus, and therefore openly checked for satisfiability or derivability. Hence, our language TPTND combines these two areas by presenting a probabilistic derivation system with types in which trust is explicitly formulated as a functional operator on typed terms and in which meta-theoretical results are interpreted accordingly. In particular, we present safety intended as the ability to prove whether the system under consideration produces nothing beyond what it is intended and designed to do, within certain margins of certainty due to its probabilistic nature. Accordingly, two main areas of research are surveyed in the following.

The extension of inferential systems based on λ-calculi or type systems with probabilities has been recently explored in the literature. The interest is mainly motivated by the need to develop semantics for probabilistic processes and programs, and to perform inferences on them. There are examples of stochastic untyped λ-calculi, which focus on denotational or operational semantics with random variables, see respectively [7, 9] and the uniform formulation for both offered in [6]. In this first sense, the meaning of a probabilistic program consists in enriching the language of programs with the ability to model sampling from distributions, thereby rendering program evaluation a probabilistic process. In this way, the work in [9] introduces a measure space on terms and defines step-indexed approximations. A sampling-based semantics of a term is then defined as a function from a trace of random samples to a value. Differently, our work while also using expressions denoting values from samples, associates the probabilistic approximation with the value in terms of types, while indexing terms with the sample size. In this sense, obviously, the expressive power of our typed language is closer to [17], which is designed specifically to model Bayesian learning; nonetheless, their semantic explanation of expressions in the language is significantly different, as expressions in our system denote samplings that generate outputs with a certain frequency, assuming a certain theoretical probability distribution. Hence, the aim of TPTND and its inferential structure are in fact closer to works introducing some form of probability in calculi with types or natural deduction systems, in particular see [10–12, 19, 25]. [19] introduces a λ-calculus augmented with special 'probabilistic choice' constructs, i.e. terms of the form $M = \{p_1 M_1, \ldots, p_n M_n\}$, meaning that term M has probability $p_1, \ldots, p_n$ of reducing to one of the terms $M_1, \ldots, M_n$ respectively. Unlike TPTND, [19] deals with judgements that do not have a context and uses a subtyping relation for the term reduction. [11] introduces 'probabilistic sequents' of the form $\Gamma \vdash^n \Delta$, which are interpreted as stating that the probability of $\Gamma \vdash \Delta$ is greater than, or equal to, a function of the natural number n, that is, $1 - n\varepsilon$ for a fixed threshold ε. A similar formalism is introduced in [12], which, closer to the natural deduction in [10], formalizes probabilistic reasoning through sequents of the form $\Gamma \vdash_a^b \Delta$, which are interpreted as empirical statements of the form 'the probability of $\Gamma \vdash \Delta$ lies in the interval $[a, b]$'. Differently from this work, TPTND does not express explicitly probabilistic intervals, but only sharp probability values, while at the same time expressing such probability on output types, rather than on the derivability relation. Finally, [25] introduces the logic $\mathrm{P}\Lambda_{\rightarrow}$ where it is possible to mix 'basic' and 'probabilistic' formulas, the former being similar to standard Boolean formulas, and the latter being formed starting from the concept of a 'probabilistic operator' $P_{\geq s} M : \sigma$ stating that the probability of $M : \sigma$ is equal to or greater than s. The semantics for $\mathrm{P}\Lambda_{\rightarrow}$ is Kripke-like and its axiomatization is infinitary.

Most significantly for type-theoretical models of probabilistic reasoning, [1] introduces a quantitative logic with fuzzy predicates and conditioning of states. The computation rules of the system can be used for calculating conditional probabilities in two well-known examples of Bayesian reasoning in (graphical) models. In the same family, [42] offers a Probabilistic Dependent Type System (PDTS) via a functional language based on a subsystem of intuitionistic type theory including dependent sums and products, expanded to include stochastic functions. It provides a sampling-based semantics for the language based on non-deterministic β-reduction. A probabilistic logic from PDTS introduced as a direct result of the Curry–Howard isomorphism is derived, and shown to provide a universal representation for finite discrete distributions.

TPTND offers an integrated way to deal with probability intervals (similarly to [10]) and probabilistic choices (similarly to [19]). However, unlike these languages, TPTND was mainly designed to reason about and derive trustworthiness properties of computational processes. In fact, differently from all these other systems, TPTND has an explicit syntax both to deal with the number of experiments, and to prove trust in a process whenever the empirically verified probability is close enough to the theoretical one.

The formulation of the syntax of TPTND for the evaluation of trustworthiness of probabilistic computational processes recalls an entirely different literature, namely that of logical systems in which trust is explicitly formulated as an operator in the language. There are several logical frameworks providing an interpretation of one of the many aspects of computational trust, from manipulation to verification to assessment [3, 4, 18, 20, 21, 38]. Modal logics, and in particular knowledge and belief logics, have been extensively investigated for trust, see e.g. [27, 30, 31]. As mentioned above, much less investigated are proof systems in which trust is formulated explicitly as an operator or a functional expression. An exception is the family of logics `(un)SecureND` whose most complete formulation, including also a relational semantics, is offered in [32]. The proof-theoretic fragment of the logic has been applied to a number of areas, ranging from software management [8, 33, 34] to modelling a trust and reputation protocol for VANET [36]; and the evaluation of the trustworthiness of online sources [13, 14]. This proof-theoretical language, in its deterministic version also includes a Coq implementation (https://github.com/gprimiero/SecureNDC) for proof-checking. The system TPTND presented in this paper can be seen as the probabilistic extension of the fragment of `(un)SecureND` including only the closure of trust under negation, which corresponds to distrust in the latter system. The present version of TPTND significantly extends and improves on the version previously formulated in [16].

3 TPTND

We now introduce the system TPTND, which comprises a logical language that enables us to talk about probabilistic computational processes and related trustworthiness properties, and a proof system providing a set of rules to formally reason about these processes and establish whether they enjoy the trustworthiness properties expressible in the language. Before introducing the syntax of the system, let us present a procedural semantics that determines the kind of computational objects and phenomena that the language of TPTND is meant to describe.

3.1 Computational semantics

The computational semantics of TPTND is a procedural semantics, that is, a semantics that determines the objects of discourse by way of a definition of their behaviour in procedural terms. As

already mentioned, the semantics formally defines the objects and phenomena that the language of TPTND is supposed to describe.

In order to introduce the semantics, let us begin with a simple example that clarifies which kind of computational and probabilistic phenomena we are interested in and how we formalize them semantically.

EXAMPLE 3.1.
Suppose that we are interested in the probabilistic behaviour of two coins u and v. Suppose, moreover, that u is a fair coin while v returns heads (value that we denote by the type η) with probability $\frac{2}{3}$ and tails (value that we denote by the type τ) with probability $\frac{1}{3}$.

In our semantics, the coin u will be defined by the two following rules:

$$\overline{\mathcal{L} \mapsto_{\frac{1}{2}} \mathcal{L}, u : \eta} \qquad \overline{\mathcal{L} \mapsto_{\frac{1}{2}} \mathcal{L}, u : \tau}$$

indicating that, in any environment $\mathcal{L}$, we can always observe with probability $\frac{1}{2}$ that the coin u returns a value η (for heads) and with probability $\frac{1}{2}$ that u returns a value τ (for tails). Technically, what we called an environment is simply a list of typed terms, either like $u : \eta$ and $u : \tau$—thus indicating that we observed that a certain term u returned a certain value η or τ—or of the form $t_n : \alpha_a$—thus indicating that the frequency with which an output of type α has been produced by a process t during a series of n executions is a. Similarly, the coin v is defined by the following two rules:

$$\overline{\mathcal{L} \mapsto_{\frac{2}{3}} \mathcal{L}, v : \eta} \qquad \overline{\mathcal{L} \mapsto_{\frac{1}{3}} \mathcal{L}, v : \tau}$$

These four rules define all we know—and care—about the two coins. By using them, we can represent experiments on the two coins. For instance, without any additional assumption, we can produce the following sequence of events:

$$\mapsto_{\frac{1}{2}} u : \eta \mapsto_{\frac{1}{2}} u : \eta, u : \eta \mapsto_{\frac{1}{2}} u : \eta, u : \eta, u : \tau \mapsto_{\frac{1}{3}} u : \eta, u : \eta, u : \tau, v : \tau$$

representing the case in which we throw three times u and one time v and we obtain twice heads from u, one time tails from u and one time tails from v. Notice that the reduction arrows $\mapsto$ are labelled by the probability of the observation that we add to the list of terms preceding the arrow in order to obtain the list following the arrow.

Now that we have represented the process of collecting several observations on the two coins, we can gather them in a statement about what happened to one coin:

$$u : \eta, u : \eta, u : \tau, v : \tau \mapsto_1 u_3 : \eta_{\frac{2}{3}}, v : \tau$$

by the sampling$^{\mapsto}$ rule that we will define below. After the sampling, our environment contains one term indicating that u returned heads twice over three throws and one indicating that v returned tails once. The reduction obtained by using this rule has probability 1 since it does not represent a probabilistic event but simply the action of putting together several observations.

Suppose now that we keep throwing the dice v and that we obtain heads two times. We can collect our observations on the value tails obtained from v along with our observations on the value heads obtained from u—in both cases, we represent this by an application of the sampling$^{\mapsto}$ rule. What we obtain is the following list of terms: $u_3 : \eta_{\frac{2}{3}}, v_3 : \tau_{\frac{1}{3}}$. At this point, we can talk about the behaviour of both coins together by further putting together our knowledge into a term $\langle u_3 : \eta_{\frac{2}{3}}, v_3 : \tau_{\frac{1}{3}} \rangle$

$$\frac{}{\mathcal{L} \quad \mapsto_{a_i} \quad \mathcal{L}, t : \alpha^i}\ \text{event}^{\mapsto t}$$

where each a_i is a number in the unit interval $[0,1]$, t is an atomic term, $\alpha^1, \ldots, \alpha^m$ are the atomic types associated to t, $1 \leqslant i \leqslant m$, $(\sum_{i=1}^{m} a_i) = 1$

$$\frac{}{\mathcal{L}, t : \alpha^1, \ldots, t : \alpha^n \quad \mapsto_1 \quad \mathcal{L}, t_n : \alpha_f}\ \text{sampling}^{\mapsto}$$

where $f = \frac{|\{i \mid \alpha_i = \alpha\}|}{n}$

$$\frac{}{\mathcal{L}, t_n : \alpha_f, t_m : \alpha_g \quad \mapsto_1 \quad \mathcal{L}, t_{n+m} : \alpha_{f\cdot(n/(n+m))+g\cdot(m/(n+m))}}\ \text{update}^{\mapsto}$$

FIGURE 1. Evaluation rules for atomic terms.

describing the probabilistic behaviour of the pair of coins with respect to the result of throwing them that consists in obtaining once heads and once tails (represented by the conjunction type $\eta \times \tau$):

$$u_3 : \eta_{\frac{2}{3}}, v_3 : \tau_{\frac{1}{3}} \ \mapsto_1 \ \langle u_3 : \eta_{\frac{2}{3}}, v_3 : \tau_{\frac{1}{3}} \rangle : (\eta \times \tau)_{\frac{2}{3}\cdot\frac{1}{3}}$$

As we will see later, other logical operations can be employed to study the behaviour of processes—for instance, considering how frequently a process yields one of two values or considering how frequently a process yields a value if another, possibly related process yields a value with a particular frequency.

We now formally define the set of rules of the operational semantics. Technically, the presented calculus is meant as a proof-system for the deduction of judgements expressing the probability of obtaining certain outputs (types) from possibly opaque, probabilistic programs (terms). Therefore, all there is to specify about the nature and behaviour of the elements of the domain of discourse is their computational behaviour. These elements are, in particular, distinct computational processes yielding outputs of mutually exclusive types. Hence, the event of a process t yielding an output α—formally, $t : \alpha$—is always supposed to be stochastically independent from the event of another process u yielding another output β *and* from the event of another process u yielding the same output α. Hence, providing a semantics in terms of generic events and their probabilities would not be very interesting and is certainly not required to fully understand the workings of the proof-system at hand. The calculus is, indeed, *not* supposed to be a means to compute the probability of generic events or processes. We focus then on the reduction rules determining the computational behaviour of our processes, which are presented in Figures 1 and 2, and discuss their meaning.

While the syntax of terms will be explicitly specified in Definition 3.2 along with the rest of the syntax of the calculus TPTND, for the present section, as per usual practice, we just define a well-formed typed term as any element of a list $\mathcal{L}'$ occurring inside an expression of the form $\mathcal{L} \mapsto \mathcal{L}'$ that can be obtained by applying one or more of the rules in Figures 1 and 2. In other words, an evaluation rule acts on a list of typed terms (which, in the case of event$^{\mapsto t}$, might be empty) and produces a non-empty list of typed terms. Typed terms, therefore, can be of two forms:

- $t : \alpha$, denoting that the process t produced an output of type α; and
- $t_n : \alpha_f$, denoting that f is the frequency with which we obtained outputs of type α during an experiment consisting of n executions of t—in other words, this denotes that the process t produced $f \cdot n$ times an output of type α over a total of n executions.

$$\frac{}{\mathcal{L}, t_n : \alpha_f, t_n : \beta_g \quad \mapsto_1 \quad \mathcal{L}, t_n : (\alpha + \beta)_{f+g}}\ \mathrm{I}^{\mapsto}{+}$$

$$\frac{t_n : \beta_g}{\mathcal{L}, t_n : (\alpha + \beta)_f \quad \mapsto_1 \quad \mathcal{L}, t_n : \alpha_{f-g}}\ \mathrm{E}^{\mapsto}{+}_L$$

$$\frac{t_n : \alpha_g}{\mathcal{L}, t_n : (\alpha + \beta)_f \quad \mapsto_1 \quad \mathcal{L}, t_n : \beta_{f-g}}\ \mathrm{E}^{\mapsto}{+}_R$$

$$\frac{}{\mathcal{L}, t_n : \alpha_f, u_n : \beta_g \quad \mapsto_1 \quad \mathcal{L}, \langle t, u \rangle_n : (\alpha \times \beta)_{f \cdot g}}\ \mathrm{I}^{\mapsto}{\times}$$

$$\frac{snd(t)_n : \beta_g}{\mathcal{L}, t_n : (\alpha \times \beta)_f \quad \mapsto_1 \quad \mathcal{L}, fst(t)_n : \alpha_{f/g}}\ \mathrm{E}^{\mapsto}{\times}_L$$

$$\frac{fst(t)_n : \alpha_g}{\mathcal{L}, t_n : (\alpha \times \beta)_f \quad \mapsto_1 \quad \mathcal{L}, snd(t)_n : \beta_{f/g}}\ \mathrm{E}^{\mapsto}{\times}_R$$

$$\frac{x_u : \alpha_a}{\mathcal{L}, t_n : \beta_f \quad \mapsto_1 \quad \mathcal{L}, [x_u]t_n : (\alpha \to \beta)_{[a]f}}\ \mathrm{I}^{\mapsto}{\to}$$

$$\frac{x_u : \alpha_a}{\mathcal{L}, [x_u]t_n : (\alpha \to \beta)_{[a]f}, u_n : \alpha_g \quad \mapsto_1 \quad \mathcal{L}, t_n.[u_n : \alpha] : \beta_{g \cdot f}}\ \mathrm{E}^{\mapsto}{\to}$$

where α and β are metavariables for generic (possibly non-atomic) types, we assume, for $\mathrm{E}^{\mapsto}{+}_L, \mathrm{E}^{\mapsto}{+}_R, \mathrm{E}^{\mapsto}{\times}_L, \mathrm{E}^{\mapsto}{\times}_R$, that $\alpha \neq \beta$, and x_u is a designated variable that we associate to the term u

FIGURE 2. Logical term evaluation rules.

For each *atomic* process t that we wish to study, we should associate a list $\alpha^1, \ldots, \alpha^m$ of atomic types to it and we should define an event$^{\mapsto t}$ rule. Such a rule will define the probabilistic computational behaviour of t. Evaluating a list of typed terms will correspond, then, to collecting the results of experiments (see Figure 1) and of logical operations (see Figure 2) executed on (possibly several copies of) the probabilistic computational processes that we have introduced by the event$^{\mapsto t}$ rules.

We discuss now the specific meaning of each evaluation rule. We begin with the rules for evaluating atomic terms and then we consider the rules implementing logical operations on typed terms indicating the frequency of outputs of a certain type during experiments on probabilistic terms.

As already mentioned, the event$^{\mapsto t}$ rule defines the behaviour of the atomic term t. Technically, the rule enables us to add to our possibly empty list $\mathcal{L}$ a typed term indicating the outcome of one execution of t. Since t denotes a probabilistic process, the rule will add to the list a typed term of the form $t : \alpha^i$, which specifies that t produced an output of type $\alpha^i \in \{\alpha^1, \ldots, \alpha^m\}$. The choice of a specific type α^i is non-deterministic, and the probability of selecting the type α^i is a_i, for each

$i \in \{1, \ldots, m\}$. The conditions on this evaluation rule guarantee that the sum of the probabilities of all possible types of outputs of a term t is always 1.

The sampling rule enables us to collect the results of n executions of an atomic term t into one typed term $t_n : \alpha_f$ indicating the frequency of the outputs of a type α. In particular, the term $t_n : \alpha_f$ indicates the number of times an output of type α has been produced over the total number of outputs produced during the n executions of t, i.e. $f \cdot n$ will be the number of times t produced an output of type α during the experiment consisting of n executions of t.

The update rule combines, by a Bayesian update function, two typed terms $t_n : \alpha_f$ and $t_m : \alpha_g$ indicating the frequency with which an output of type α has been produced by t during, respectively, an experiment consisting of n executions of t and an experiment consisting of m executions of t, and produces a typed term $t_{n+m} : \alpha_{f\cdot(n/(n+m))+g\cdot(m/(n+m))}$ indicating the frequency with which an output of type α has been produced by t during an experiment consisting of $n + m$ executions of t.

We explain now the meaning of the logical operation rules, presented in Figure 2.

The rule $\mathrm{I}^{\mapsto}+$ enables us to combine two typed terms specifying the frequency of the outputs of type α during an experiment consisting of n executions of the process t and of the outputs of type β during an experiment consisting of n executions of t, in order to indicate the frequency of outputs of type either α or β (namely of outputs of type $\alpha + \beta$) during an experiment consisting of n executions of t.

The rules $\mathrm{E}^{\mapsto}+$ can be applied to a typed term indicating the frequency of outputs of type $\alpha + \beta$, with respect to an experiment consisting of n executions of t, in order to extract the information about the frequencies of the outputs of type α and of the outputs of type β. If we wish to extract the information about the frequencies of the outputs of type α we use $\mathrm{E}^{\mapsto}+_L$ and we need the information on the outputs of type β indicated in the premise of the rule. In order to extract the information about the frequencies of the outputs of type β we use $\mathrm{E}^{\mapsto}+_R$ and we need the information on the outputs of type α indicated in the premise of the rule.

The rule $\mathrm{I}^{\mapsto}\times$ enables us to combine two typed terms specifying the frequency of the outputs of type α during an experiment consisting of n executions of the process t and of the outputs of type β during an experiment consisting of n executions of the process u, in order to indicate the frequency of outputs of type $\alpha \times \beta$ (namely of pairs of outputs in which the first element is of type α and the second of type β) during an experiment consisting of n executions of the pair of processes $\langle t, u\rangle$.

The rules $\mathrm{E}^{\mapsto}\times$ can be applied to a typed term indicating the frequency of outputs of type $\alpha \times \beta$ (and thus of pairs of outputs in which the first element is of type α and the second of type β), with respect to an experiment consisting of n executions of t, in order to extract the information about the frequencies of the outputs of type α and of the outputs of type β occurring as first and second element, respectively, of the considered output pairs. If we wish to extract the information about the frequencies of the first elements of the pairs (those of type α) we use $\mathrm{E}^{\mapsto}\times_L$ and we need the information on the second elements of the considered pairs of outputs, expressed by the premise of the rule. In order to extract the information about the frequencies of the second elements of the pairs (those of type β) we use $\mathrm{E}^{\mapsto}\times_R$ and we need the information on the first elements of the considered pairs of outputs, expressed by the premise of the rule.

The rule $\mathrm{I}^{\mapsto}\ \to$ enables us to make explicit a dependency between the frequency of the outputs of type β of a process t and the probability of obtaining an output of type α from a process u. The rule yields a term $[x]t$ indicating the dependency of the process t on the value variable x indicating the theoretical probability of obtaining α. The type of the term $[x]t$ is $\alpha \to \beta$ in order to indicate the existence of a dependency also at the level of types.

The rule $\mathrm{E}^{\mapsto}\ \rightarrow$ enables us to combine a typed term $[x_u]t_n : (\alpha \rightarrow \beta)_{[a]f}$ with a typed term $u_n : \alpha_g$. Since the first term $[x_u]t_n : (\alpha \rightarrow \beta)_{[a]f}$ indicates that the frequency f of the outputs of type β produced by a process t depends on the probability a of obtaining an output of type α from a process u and since the second term $u_n : \alpha_g$ indicates that an experiment of n executions of the process u yielded outputs of type α with a frequency of g, the resulting term $t_n.[u_n : \alpha] : \beta_{g \cdot f}$, obtained by applying the first term to the second, indicates that the frequency of outputs of type β obtained during an experiment consisting of n executions of t is $f \cdot g$ since the frequency of this type of output depends on the, now known, frequency g of the outputs of type α produced by u.

3.2 Syntax

The syntax of TPTND is generated by the following grammar in Backus–Naur form:

DEFINITION 3.2 (Syntax).

$$\begin{aligned}
&\mathtt{X} := \mathtt{x} \mid \mathtt{x}_{\mathtt{T}} \mid \langle \mathtt{X}, \mathtt{X} \rangle \mid \mathit{fst}(\mathtt{X}) \mid \mathit{snd}(\mathtt{X}) \mid [\mathtt{X}]\mathtt{X} \mid \mathtt{X}.\mathtt{X} \mid \mathtt{X}.\mathtt{T} \\
&\mathtt{T} := \mathtt{t} \mid \langle \mathtt{T}, \mathtt{T} \rangle \mid \mathit{fst}(\mathtt{T}) \mid \mathit{snd}(\mathtt{T}) \mid [\mathtt{X}]\mathtt{T} \mid \mathtt{T}.\mathtt{T} \mid \mathit{Trust}(\mathtt{T}) \mid \mathit{UTrust}(\mathtt{T}) \\
&\mathtt{O} := \alpha \mid \mathtt{O}_r \mid \mathtt{O}_r^{\perp} \mid (\mathtt{O} \times \mathtt{O})_r \mid (\mathtt{O} + \mathtt{O})_r \mid (\mathtt{O} \rightarrow \mathtt{O})_{[r']r} \\
&\mathtt{C} := \{\} \mid \mathtt{C}, \mathtt{X} : \mathtt{C} \mid \mathtt{C}, \mathtt{X} : \mathtt{C}_r \mid \mathtt{C}, \mathtt{X} : \mathtt{C}_{[r,r']} \\
&\mathtt{S} := \mathtt{C} :: \mathit{distribution} \mid \mathtt{O} :: \mathit{output} \mid r :: \mathbb{R}
\end{aligned}$$

We denote *random variables* by $\mathtt{X}$. The metavariable $\mathtt{x}$ denotes a particular random variable, e.g. x, x', y, z. Indexed random variables $x_{\mathtt{T}}$ are tied to specific terms, meaning that they describe uncertainty about the output of the denoted process t. We may combine independent random variables into pairs of the form $\langle \mathtt{X}, \mathtt{X} \rangle$, which intuitively corresponds to their joint distribution. Pairs have associated projection functions $\mathit{fst}(\mathtt{X})$ and $\mathit{snd}(\mathtt{X})$ to extract the first and second variable. We use the constructions $[\mathtt{X}]\mathtt{X}'$ and $\mathtt{X}.\mathtt{X}'$ (with $\mathtt{X}'$ a metavariable for a possibly distinct variable) to express the dependency relation between random variables, i.e. the probability assigned to the random variable $\mathtt{X}$, given the probability assigned to $\mathtt{X}'$ and the corresponding evaluation. The construction $\mathtt{X}.\mathtt{T}$ expresses the update of the probability assigned to random variable $\mathtt{X}$ when using the information on the frequency of the output of a process $\mathtt{T}$.

Terms $\mathtt{T}$ are executed computational processes or computational experiments. The metavariable $\mathtt{t}$ denotes any constant term, e.g. $t, t', u, w, \ldots$. Terms with different names are regarded as *distinct* processes. When *one and the same* term for a process occurs in distinct branches of a derivation (possibly with different numerical subscripts), these must be intended as distinct runs or executions of the same process. Sometimes we use superscripts $t^1, \ldots, t^n$ to distinguish the first to the n^{th} distinct executions of a process t. We may compose processes into pairs $\langle \mathtt{T}, \mathtt{T} \rangle$ associated with projection functions $\mathit{fst}(\mathtt{T})$ and $\mathit{snd}(\mathtt{T})$. The construct $[\mathtt{X}]\,\mathtt{T}$ expresses the expected probability assigned to the output of process $\mathtt{T}$ given the theoretical probability assigned to $\mathtt{X}$; while the construct $\mathtt{T}.\mathtt{T}$ expresses the update of the expected probability assigned to the output of a process $\mathtt{T}$ with the expected probability of the output of another process of the form $\mathtt{T}$, where the latter is taken to substitute the dependency of $\mathtt{T}$ from some random variable $\mathtt{X}$. We include among terms the function-like expressions for trust: $\mathit{Trust}(\mathtt{T})$ expresses that the frequency of process $\mathtt{T}$ is considered trustworthy with respect to the theoretical probability of its output, and the trust operator is defined by an appropriate pair of introduction and elimination rules below. The function $\mathit{UTrust}(\mathtt{T})$ expresses negated trustworthiness for the frequency shown by the output of process $\mathtt{T}$ with respect to its intended theoretical probability.

Types $\mathtt{O}$ are output values of random variables and computational processes, with a probability $r \in [0, 1]$ attached. Similarly as in the case of terms and random variables, α and β are used as metavariables for generic types (thus possibly non atomic). Output $\alpha \times \beta$ expresses the independent occurrence of outputs α and β. Output $\alpha + \beta$ expresses the disjunction of outputs α and β. The construction $(\alpha \to \beta)_{[r']r}$ is interpreted as the probability r of output β for a given random variable or process, provided the probability of output α is r'. We use $\alpha^{\perp}$ to denote any output different than α (possibly including no output). We will use subscripts a, b, c, ... to denote instance values of probability r when referring to the output of a random variable $\mathtt{X}$. Hence, in a formula of the form $x : \alpha_a$, the subscript a denotes the probability associated with the event of random variable x taking value α. The formula $t_n : \alpha_{\tilde{a}}$ expresses the expected probability a that process t will output a value of type α. In the expression $t_n : \alpha_f$, the subscript denotes the empirical probability or frequency associated with output α extracted from n executions of the process denoted by t. By convention, we use $t : \alpha$ without subscript either in the term or in the type to denote a single execution of the process t with deterministic output α (i.e. $r = 1$).

Contexts $\mathtt{C}$ are sets of conditions, or assumptions on the theoretical probabilities of random variables required for the assertions made on the right of $\vdash$ to hold. Contexts are composed by lists of expressions of the form $x_t : \alpha_a$, with or without probability a attached. They presuppose type declarations $\alpha :: \mathit{output}$ for the occurring outputs, saying that α is a valid output. We assume that, even when the actual distribution of probabilities on the random variables of interest of a given system is unknown, a certain assumption on their values can be made, and in particular the assumption that the distribution is unknown can be stated.

Type declarations $\mathtt{S}$ include statements of the form: $\mathtt{C} :: \mathit{distribution}$, expressing that a given (possibly non empty) list of declarations of the form $x : \alpha_a$ is a probability distribution; $\alpha :: \mathit{output}$ expressing that α is a valid output value (e.g. a Boolean, or a Natural, or a List, etc.); $r :: \mathbb{R}$ expressing that r is a real number. These declarations are generally considered as implicit presuppositions of contexts.

3.3 Judgements

We now summarize the forms of judgements admissible in TPTND. A judgement of the form

$$x : \alpha \vdash y : \beta_b$$

says that the probability of random variable y to have value β is b, provided random variable x has value α. From the probabilistic point of view, this may be intuitively interpreted as $P(Y = \beta \mid X = \alpha) = b$. In the particular case of an empty context, we may simply interpret

$$\vdash y : \beta_b$$

as $P(Y = \beta) = b$.

A judgement of the form

$$x : \alpha_a \vdash y : \beta_b$$

says that the probability of random variable y to take value β is b if x takes value α with probability a. In other words, the probability $P(Y = \beta)$ is a function, say g, of $P(X = \alpha)$, and $g(a) = b$.

A judgement of the form

$$\vdash t : \alpha$$

says that an event or process t has taken place with output value α.

A judgement of the form

$$x_t : \alpha_a \vdash t : \beta$$

serves the purpose of reasoning about data while making assumptions on its probability distribution; it may be intuitively thought as saying that a process t produces output β assuming a variable x has probability a of producing output α. This form of judgement is then generalized and formulated as expected probabilities under some distribution:

$$x_t : \alpha_a \vdash t_n : \beta_{\tilde{b}}$$

says that a process t produces output β with expected probability $\tilde{b}$ over n executions assuming a variable x has probability a of producing output α.

Next, we can formulate this judgement under more assumptions on probability values for conditions to be satisfied on the left:

$$x_t : \alpha_a, \ldots, z_t : \nu_n \vdash t_n : \beta_{\tilde{b}}$$

says that provided variables $x, \ldots, z$ with probabilities $a, \ldots, n$ to produce outputs $\alpha, \ldots \nu$ respectively obtain, the associated process t produces output β with expected probability $\tilde{b}$.

The relative frequency of β in n executions of t is denoted by

$$x_t : \alpha_a, \ldots, z_t : \nu_n \vdash t_n : \beta_f$$

Note that as these are judgements based on assumptions, the evaluation of the frequency with which β is shown (or its expected probability to occur) requires the evaluation of the conditions: given a judgement of the form $x_t : \alpha_a \vdash t : \beta$, the assertion that t produces β requires the judgement $\vdash x : \alpha$ to hold; the same is true for multiple assumptions. In the case of a judgement $x_t : \alpha_a \vdash t_n : \beta_{\tilde{b}}$, the verification of the condition can only be formulated over the same amount of executions as t, in which case the judgement $\vdash u_n : \alpha_{\tilde{a}}$ is required.

Finally, a judgement of the form

$$x_t : \alpha_{[0,1]}, \ldots, z_t : \nu_{[0,1]} \vdash t_n : \beta_f$$

says that n executions of process t display output β with a frequency f under an unknown distribution of values to random variables. Note that while no inference is possible from a judgement that is indexed by a range $[0, 1]$ within the regular deductive rules of our system, we admit judgements of this form as a premise in the rule to infer trust or distrust in $t_n : \beta_f$ (see Section 3.8), as well as in the premise to apply contraction, where all the values in the interval may be applied to an output value and a function is executed to extract one of them (see Section 3.9).

3.4 Distribution construction rules

Contexts as a list of assumptions on probability distributions are inductively constructed, see Figure 3. They are generated by judgements of the form $\vdash x : \alpha_a$, expressing assignments of probabilities to output variables. An empty set {} represents the base case (rule 'base') to define distributions. Random variables assigning theoretical probability values to outputs can be added to a distribution as long as they respect the standard additivity requirement on probabilities ('extend'). Here α is intended as a metavariable for both atomic and compound output types; the cases for the rule 'extend' include joint distribution and extension by dependent variables, as justified below in the construction rules for random variables. The variant for deterministic outputs 'extend_det' has $x : \alpha \notin \Gamma$ in the second premise and it does not require the additivity condition. Indeed, the meaning

$$\frac{}{\{\} :: \mathit{distribution}}\ \text{base}$$

$$\frac{\Gamma :: \mathit{distribution} \qquad x : \alpha_a \notin \Gamma, \forall a \in [0,1] \qquad (\sum_{x:\rho_p \in \Gamma} p) + a \leqslant 1}{\Gamma, x : \alpha_a :: \mathit{distribution}}\ \text{extend}$$

$$\frac{\Gamma :: \mathit{distribution} \qquad x : \alpha \notin \Gamma}{\Gamma, x : \alpha :: \mathit{distribution}}\ \text{extend\_det}$$

$$\frac{\alpha :: \mathit{output} \qquad \ldots \qquad \omega :: \mathit{output}}{\{x : \alpha_{[0,1]}, \ldots, x : \omega_{[0,1]}\} :: \mathit{distribution}}\ \text{unknown}$$

FIGURE 3. Distribution construction.

of $x : \alpha$ is that the random variable x has been assigned the value α, and this is not in contrast with an assumption of the form $x : \alpha_a$ that establishes a theoretical probability for that variable. To denote an unknown distribution ('unknown') we use the (transfinite) construction assigning to all values of interest the interval of all possible probability values, provided the additivity condition of 'extend' is preserved. Note that, while the fragment of the calculus already introduced may not apply such unknown distribution to infer values on the outputs, as we will see in Section 3.8, the Trust fragment (see Figure 9) admits the use of an unknown distribution in the second premise of the introduction rules (IT/IUT) and in the conclusion of the corresponding eliminations (ET/EUT). When an unknown distribution is used, the frequency values assigned to the outputs on the right-hand side of our judgement must be understood as observed values under an unknown distribution.

EXAMPLE 3.3.

$$\Gamma = \{x_d : 1_{1/6}, \ldots, x_d : 6_{1/6}\}$$

is the theoretical probability distribution associated with the outputs of a fair die d.

EXAMPLE 3.4.

$$\Gamma = \{x_d : \mathit{Heads}_{1/2}, x_d : \mathit{Tails}_{1/2}, y_{d'} : \mathit{Heads}_{1/3}, y_{d'} : \mathit{Tails}_{2/3}\}$$

is the joint theoretical distribution of two independent random variables x_d and $y_{d'}$ associated with a fair coin d and biased coin d' respectively.

3.5 Reasoning about random variables

The inferential engine of TPTND for random variables is expressed by the rules in Figure 4.

The axiom-like rule 'identity$_1$' expresses the probabilistic fact that $P(X = \alpha \mid X = \alpha) = 1$. The rule 'identity$_2$' generalizes the former by expressing that the probability of a random variable x to have output α is a, provided variable x has theoretical probability a to output α.

The rule '$\perp$' encodes the probabilistic complement rule: if the probability of output α for the random variable x is a, the probability of x not producing output α is $1 - a$.

$$\frac{}{\Gamma, x : \alpha \vdash x : \alpha_1}\ \text{identity}_1$$

$$\frac{}{\Gamma, x : \alpha_a \vdash x : \alpha_a}\ \text{identity}_2$$

$$\frac{\Gamma \vdash x : \alpha_a}{\Gamma \vdash x : \alpha^{\perp}_{1-a}}\ \perp$$

$$\frac{\Gamma \vdash x : \alpha_a \qquad \Delta \vdash y : \beta_b \qquad \Gamma \perp\!\!\!\perp \Delta}{\Gamma, \Delta \vdash \langle x, y \rangle : (\alpha \times \beta)_{a \cdot b}}\ \text{I}\times$$

$$\frac{\Gamma \vdash x : (\alpha \times \beta)_c \qquad \Gamma \vdash \mathit{fst}(x) : \alpha_a}{\Gamma \vdash \mathit{snd}(x) : \beta_{c/a}}\ \text{E}\times_L$$

$$\frac{\Gamma \vdash x_n : (\alpha \times \beta)_c \qquad \Gamma \vdash \mathit{snd}(x) : \beta_b}{\Gamma \vdash \mathit{fst}(x) : \alpha_{c/b}}\ \text{E}\times_R$$

$$\frac{\Gamma \vdash x : \alpha_a \qquad \Gamma \vdash x : \beta_b}{\Gamma \vdash x : (\alpha + \beta)_{a+b}}\ \text{I}+$$

$$\frac{\Gamma \vdash x : (\alpha + \beta)_a \qquad \Gamma \vdash x : \alpha_{a'}}{\Gamma \vdash x : \beta_{a-a'}}\ \text{E}+_R$$

$$\frac{\Gamma \vdash x : (\alpha + \beta)_b \qquad \Gamma \vdash x : \beta_{b'}}{\Gamma \vdash x : \alpha_{b-b'}}\ \text{E}+_L$$

$$\frac{\Gamma, x : \alpha \vdash y : \beta_b}{\Gamma \vdash [x]y : (\alpha \to \beta)_b}\ \text{I}\to$$

$$\frac{\Gamma \vdash [x]y : (\alpha \to \beta)_b \qquad \Gamma \vdash x : \alpha_a}{\Gamma \vdash y.(x : \alpha) : \beta_{a \cdot b}}\ \text{E}\to$$

where we assume, for $\text{E}+_L, \text{E}+_R, \text{E}\times_L, \text{E}\times_R$, that $\alpha \neq \beta$

FIGURE 4. Rules for random variables.

The joint distribution of two independent random variables can be constructed with conjunction $\times$. We use the notation '$\Gamma \perp\!\!\!\perp \Delta$' with the intended meaning that contexts Γ and Δ are independent, meaning that for any $x : \alpha_a \in \Gamma$, $y : \beta_b \in \Delta$ distinct, is never the case that $\Delta \vdash x : \alpha_a$ and $\Gamma \vdash y : \beta_b$. Note that distinct variables can be made dependent through the use of $\to$ and that the same variable can occur in independent contexts. The rule generalizes the independence condition to entire distributions and since it assumes their independence, the joint distribution multiplies their probabilities. The elimination uses *fst* and *snd* to derive the probability of the other term by dividing probabilities.

Disjunction $+$ merges mutually exclusive outputs for a given variable by adding their probabilities. Elimination of disjunction amounts to subtracting them instead.

The connective $\rightarrow$ is used as a syntactical rewriting of dependent variables. The implication introduction constructs a variable of type $\alpha \rightarrow \beta$ where the probability b of y being β is explicitly made dependent on x being α. In the elimination, we assign a specific probability a to x being α and construct the application term $x.(y : \alpha)$, which is assigned output β with probability $a \cdot b$.

EXAMPLE 3.5 (Dependent variables).
We want to model a distribution with two random variables x and y corresponding to two coins, where x is fair. Variable y always returns *Heads* when x returns *Tails*, and vice-versa:

$$\frac{x : \mathit{Heads} \vdash y : \mathit{Tails}_1}{\vdash [x]y : \mathit{Heads} \rightarrow \mathit{Tails}_1}\ \mathrm{I}{\rightarrow}$$

$$\frac{x : \mathit{Tails} \vdash y : \mathit{Heads}_1}{\vdash [x]y : \mathit{Tails} \rightarrow \mathit{Heads}_1}\ \mathrm{I}{\rightarrow}$$

We show that y is also fair by eliminating implication:

$$\frac{\vdash [x]y : \mathit{Heads} \rightarrow \mathit{Tails}_1 \qquad \Gamma \vdash x : \mathit{Heads}_{0.5}}{\Gamma \vdash y.(x : \mathit{Heads}) : \mathit{Tails}_{1 \cdot 0.5}}\ \mathrm{E}{\rightarrow}$$

3.6 Reasoning about events, samples and expected probabilities

The inferential engine for single executions of computational processes performed under a given probability distribution is illustrated in Figure 5.

The rule 'experiment' is the declaration of the deterministic output α (without subscript) of one experiment or execution of process t (without subscript) under a random variable that assigns theoretical probability a to output α (and hence derives α with that probability). Experiments can then be combined for dependent and independent results, in the same way done above for random variables. We remark that the added condition $x_t : \alpha_a, x_t : \beta_b \in \Gamma$ to the $\mathrm{I}{+}_L$ and $\mathrm{I}{+}_r$ rules is needed to guarantee that, when $\alpha + \beta$ is inferred from α (respectively from β), β (respectively α) is indeed a possible output of t. Notice moreover that, given any term t, we associate to it a designated variable x_t, as indicated in Figure 5.

Inferences on processes that assume some probability value for a random variable are possible under the construction of the $\rightarrow$ connective: the introduction states that if process t outputs value β (with $r = 1$) provided the condition that x is assigned value α is satisfied with probability a, then one constructs term $[x]t$, which assigns the probability $a \cdot 1$ to the output $(\alpha \rightarrow \beta)$; the elimination says that on condition that a process u with value α obtains $(r = 1)$, process t using information u outputs value β $(r = 1)$, thereby eliminating the probabilistic condition.

Rules from Figure 5 allow to perform complex experiments. We now want to generalize to multiple executions of an experiment in Figure 6. This means to evaluate two things: first, the expected value of the probability of an output given n executions of an experiment; second, the frequency of a given output in n executions of an experiment. As expected probabilities are computed according to their theoretical counterparts, we also define logical operations on them.

The axiom-like rule 'expectation' says that given the theoretical probability a that the random variable associated with process t will output value α, the expected probability of having α over n executions of t is $\tilde{a}$. Provided the maximally expected probability of having α over n executions is exactly a, we use $\tilde{a}$ as a notational device to distinguish theoretical and expected probabilities.

$$\frac{}{\Gamma, x_t : \alpha_a \vdash t : \alpha}\ \text{experiment}$$

$$\frac{\Gamma \vdash t : \alpha \qquad \Delta \vdash u : \beta \qquad \Gamma \perp\!\!\!\perp \Delta}{\Gamma, \Delta \vdash \langle t, u \rangle : (\alpha \times \beta)}\ \text{I}\times$$

$$\frac{\Gamma \vdash t : (\alpha \times \beta)}{\Gamma \vdash \mathit{fst}(t) : \alpha}\ \text{E}\times_L \qquad \frac{\Gamma \vdash t : (\alpha \times \beta)}{\Gamma \vdash \mathit{snd}(t) : \beta}\ \text{E}\times_R$$

$$\frac{\Gamma \vdash t : \alpha}{\Gamma \vdash t : (\alpha + \beta)}\ \text{I}+_L \qquad \frac{\Gamma \vdash t : \beta}{\Gamma \vdash t : (\alpha + \beta)}\ \text{I}+_R$$

$$\text{with } x_t : \alpha_a, x_t : \beta_b \in \Gamma$$

$$\frac{\Gamma \vdash t : (\alpha + \beta) \qquad \Gamma \vdash t : (\alpha^{\perp})}{\Gamma \vdash t : \beta}\ \text{E}+_R$$

$$\frac{\Gamma \vdash t : (\alpha + \beta) \qquad \Gamma \vdash t : (\beta^{\perp})}{\Gamma \vdash t : \alpha}\ \text{E}+_L$$

$$\frac{\Gamma, x : \alpha_a \vdash t : \beta}{\Gamma \vdash [x]t : (\alpha \to \beta)_a}\ \text{I}\to \qquad \frac{\Gamma \vdash [x]t : (\alpha \to \beta)_a \qquad \Delta \vdash u : \alpha}{\Gamma, \Delta \vdash t.[u : \alpha] : \beta}\ \text{E}\to$$

where we assume, for $\text{E}+_L, \text{E}+_R$, that $\alpha \neq \beta$ and x_t is a designated variable that we associate to the term t.

FIGURE 5. Single-experiment rules.

In the rule 'sampling' we denote a frequency value f of output α from n actual executions of process t, each with its own deterministic output, where f is the number of cases in which α has occurred as output over the total number of cases n. As a general case, the distribution under which t is executed can be taken to be unknown. An experiment can be repeated several times for a given process t, and the rule 'update' tells how to calculate relative frequencies on the various number of executions, each provided by an instance of 'sampling'. Note that in the examples below to aid readability we will sometimes formulate the rule 'sampling' without appropriate premises (i.e. the corresponding several instances of the rule 'experiment'), but will always assume those are properly executed. Moreover, 'update' is obviously a shorthand for an extended 'sampling', but it is notationally useful when its premises occur at distinct stages of a derivation tree.

EXAMPLE 3.6.
Consider a die of which we know nothing, and in particular it is unknown whether it is fair or not. We denote this state of affairs with the distribution $\Gamma = \{x : 1_{[0,1]}, \ldots, x : 6_{[0,1]}\}$, as produced by the type-definition 'unknown'. We execute two sets of four experiments to evaluate the frequency of output 1.

$$\frac{\Gamma \vdash d^1 : 1 \qquad \Gamma \vdash d^2 : 1 \qquad \Gamma \vdash d^3 : 5 \qquad \Gamma \vdash d^4 : 6}{\Gamma \vdash d_4 : 1_{1/2}} \text{ sampling}$$

$$\frac{\Gamma \vdash d^1 : 3 \qquad \Gamma \vdash d^2 : 1 \qquad \Gamma \vdash d^3 : 5 \qquad \Gamma \vdash d^4 : 6}{\Gamma \vdash d_4 : 1_{1/4}} \text{ sampling}$$

$$\frac{\dfrac{\mathcal{D}_1 \qquad \mathcal{D}_2}{\{x : 1_{[0,1]}, \ldots, x : 6_{[0,1]}\} \vdash d_8 : 1_{3/8}} \text{ update}}{x_t : \alpha_a \vdash t_n : \alpha_{\tilde{a}}} \text{ expectation}$$

$$\frac{\Gamma \vdash t^1 : \alpha^1 \qquad \ldots \qquad \Gamma \vdash t^n : \alpha^n}{\Gamma \vdash t_n : \alpha_f} \text{ sampling}$$

$$\text{where } f = \tfrac{|\{i \mid \alpha^i = \alpha\}|}{n}$$

$$\frac{\Gamma \vdash t_n : \alpha_f \qquad \Gamma \vdash t_m : \alpha_{f'}}{\Gamma \vdash t_{n+m} : \alpha_{f \cdot (n/(n+m)) + f' \cdot (m/(n+m))}} \text{ update}$$

$$\frac{\Gamma \vdash t_n : \alpha_{\tilde{a}} \qquad \Gamma \vdash t_n : \beta_{\tilde{b}}}{\Gamma \vdash t_n : (\alpha + \beta)_{\tilde{a}+\tilde{b}}} \text{I+}$$

$$\frac{\Gamma \vdash t_n : (\alpha + \beta)_{\tilde{c}} \qquad \Gamma \vdash t_n : \alpha_{\tilde{a}}}{\Gamma \vdash t_n : \beta_{\tilde{c}-\tilde{a}}} \text{E+}_L$$

$$\frac{\Gamma \vdash t_n : (\alpha + \beta)_{\tilde{c}} \qquad \Gamma \vdash t_n : \beta_{\tilde{b}}}{\Gamma \vdash t_n : \beta_{\tilde{c}-\tilde{b}}} \text{E+}_R$$

$$\frac{\Gamma \vdash t_n : \alpha_{\tilde{a}} \qquad \Delta \vdash u_n : \beta_{\tilde{b}} \qquad \Gamma \perp\!\!\!\perp \Delta}{\Gamma, \Delta \vdash \langle t, u \rangle_n : (\alpha \times \beta)_{\widetilde{a \cdot b}}} \text{I}\times$$

$$\frac{\Gamma \vdash t_n : (\alpha \times \beta)_{\tilde{c}} \qquad \Gamma \vdash \mathit{fst}(t)_n : \alpha_{\tilde{a}}}{\Gamma \vdash \mathit{snd}(t)_n : \beta_{\widetilde{c/a}}} \text{E}\times_L$$

$$\frac{\Gamma \vdash t_n : (\alpha \times \beta)_{\tilde{c}} \qquad \Gamma \vdash \mathit{snd}(t)_n : \beta_{\tilde{b}}}{\Gamma \vdash \mathit{fst}(t)_n : \alpha_{\widetilde{c/b}}} \text{E}\times_R$$

$$\frac{\Gamma, x_t : \alpha_a \vdash t_n : \beta_{\tilde{b}}}{\Gamma \vdash [x]t_n : (\alpha \to \beta)_{[a]\tilde{b}}} \text{I}\to$$

$$\frac{\Gamma \vdash [x]t_n : (\alpha \to \beta)_{[a]\tilde{b}} \qquad \Delta \vdash u_n : \alpha_{\tilde{a}}}{\Gamma, \Delta \vdash t_n.[u_n : \alpha] : \beta_{\tilde{a} \cdot \tilde{b}}} \text{E}\to$$

where we assume, for $\text{E+}_L, \text{E+}_R$, that $\alpha \neq \beta$ and x_t is a designated variable that we associate to the term t.

FIGURE 6. Expected probabilities and sampling rules.

$$\dfrac{\Gamma \vdash d^1 : 1 \qquad \Gamma \vdash d^2 : 1 \qquad \Gamma \vdash d^3 : 5 \qquad \Gamma \vdash d^4 : 6}{\Gamma \vdash d_4 : 1_{1/2}}\ \text{sampling}$$

$$\dfrac{\Gamma \vdash d^1 : 3 \qquad \Gamma \vdash d^2 : 1 \qquad \Gamma \vdash d^3 : 5 \qquad \Gamma \vdash d^4 : 6}{\Gamma \vdash d_4 : 1_{1/4}}\ \text{sampling}$$

$$\dfrac{\mathcal{D}_1 \qquad \mathcal{D}_2}{\{x : 1_{[0,1]}, \ldots, x : 6_{[0,1]}\} \vdash d_8 : 1_{3/8}}\ \text{update}$$

Rule I+ introduces disjunction: intuitively, if under a distribution Γ a sample of process t produces output α with an expected probability $\widetilde{a}$, and output β with expected probability $\widetilde{b}$, then the expected probability of output α or output β by a run of process t is $\widetilde{a} + \widetilde{b}$. Note that this rule can also be seen as an abbreviation of an instance of *sampling* from n premises each of which is the result of introducing $+$ on a single experiment whose output is either α or β:

EXAMPLE 3.7.
The rule I+ followed by a sampling for a process with different outputs (including outputs possibly different from α or β):

$$\dfrac{\dfrac{\Gamma, x_t : \alpha_a, x_t : \beta_b \vdash t^1 : \alpha \qquad \Gamma, x_t : \alpha_a, x_t : \beta_b \vdash t^1 : \beta}{\Gamma, x_t : \alpha_a, x_t : \beta_b \vdash t^1 : (\alpha + \beta)}\ \text{I+} \qquad \cdots \qquad \dfrac{\Gamma, x_t : \alpha_a, x_t : \beta_b \vdash t^n : \alpha \qquad \Gamma, x_t : \alpha_a, x_t : \beta_b \vdash t^n : \beta}{\Gamma, x_t : \alpha_a, x_t : \beta_b \vdash t^n : (\alpha + \beta)}\ \text{I+}}{\Gamma, x_t : \alpha_a, x_t : \beta_b \vdash t_n : (\alpha + \beta)_f}\ \text{sampling}$$

Conversely by E+_R (respectively E+_L): if under a distribution Γ a process t produces output α or output β with expected probability $\widetilde{c}$, and the former (respectively, the latter) output has probability $\widetilde{a}$ (respectively, $\widetilde{b}$), then it produces the latter (respectively, the former) output with probability $\widetilde{c} - \widetilde{a}$ (respectively $\widetilde{c} - \widetilde{b}$). Note that also this rule can be seen as an abbreviation for an instance of 'sampling' from n premises each of which is the result of introducing $+$ on a single experiment whose output is either α or β, followed by an instance of 'sampling' of either output:

EXAMPLE 3.8.
Consider an application of the rule E+ used to derive the expected probability of one among different possible outputs of a given process:

$$\dfrac{\begin{array}{c}\vdots\\ \Gamma, x_t : \alpha_a, x_t : \beta_b \vdash t_n : (\alpha + \beta)_{\widetilde{a+b}}\end{array} \qquad \Gamma, x_t : \alpha_a, x_t : \beta_b \vdash t_n : \beta_{\widetilde{b}}}{\Gamma, x_t : \alpha_a, x_t : \beta_b \vdash t_n : \alpha_{\widetilde{(a+b)-b}}}\ \text{E+}$$

followed by a sampling:

$$\dfrac{\Gamma, x_t : \alpha_a, x_t : \beta_b \vdash t^1 : \alpha^1 \qquad \ldots \qquad \Gamma, x_t : \alpha_a, x_t : \beta_b \vdash t^n : \alpha^n}{\Gamma, x_t : \alpha_a, x_t : \beta_b \vdash t_n : \alpha_{f'}}\ \text{sampling}$$

EXAMPLE 3.9.
Consider a fair die d whose distribution of outputs is encoded in Γ:

$$\dfrac{\dfrac{\mathcal{D}_1}{\Gamma \vdash d_6 : 1_{\widetilde{1/6}}}\ \text{expectation} \qquad \dfrac{\mathcal{D}_2}{\Gamma \vdash d_6 : 3_{\widetilde{1/6}}}\ \text{expectation}}{\Gamma \vdash d_6 : (1+3)_{\widetilde{1/3}}}\ \text{I+}$$

now followed by a sampling whose observed outputs are the series 1, 3, 5, 1, 2, 6 in a sample of 6 throws. In this sample the frequency of 1 or 3 outputted by d is given in our syntax as follows

$$\frac{\Gamma \vdash d^1 : (1 + ... + 6)^1 \qquad \ldots \qquad \Gamma \vdash d^6 : (1 + ... + 6)^6}{\Gamma \vdash d_6 : 1_{3/6}} \text{ sampling}$$

Note that in both examples above, we have two conclusions with the same output, one that expresses the expected probability, one that expresses actual frequency over a number of executed trials. In the following it will be our aim to define operators to measure their distance.

The typing rule I$\times$ says that if two distinct processes (and thus independent: recall that we denote by different terms distinct processes, hence the distinct distributions Γ, Δ) t and u produce expected probabilities $\widetilde{a}$ and $\widetilde{b}$ for outputs α and β and samples of size n under distributions Γ and Δ respectively; the expected probability of jointly getting output α and β from the pair $\langle t, u\rangle$ under the joint distributions Γ, Δ, is given by $\widetilde{a \cdot b}$. By E$\times$, given a process that provides two distinct output types with associated probability $\tilde{c}$, and knowing the expected probability $\tilde{a}$ of the first composing process in the given sample, we infer the probability of the second process to produce the second output as $\widetilde{c/a}$; the dual rule has *snd* in the second premise and *fst* in the conclusion.

EXAMPLE 3.10.
Let d and g be two dice associated with distributions Δ and Γ respectively. We use distinct names for the distribution of distinct processes, although they might turn out to be identical, e.g. in the case of two fair dice. Assuming independence on Δ, Γ, output expectation on a sample of 18 throws of dice d and g is as follows:

$$\frac{\dfrac{}{\Delta \vdash d_{18} : 1_{\widetilde{2/3}}}\ \text{expectation} \qquad \dfrac{}{\Gamma \vdash g_{18} : 2_{\widetilde{1/6}}}\ \text{expectation}}{\Delta, \Gamma \vdash \langle d, g\rangle_{18} : (1 \times 2)_{\widetilde{1/9}}}\ \text{I}\times$$

An observation on a sampling of 18 throws of dice d might shows the series:

$$4, 1, 2, 1, 1, 1, 6, 1, 3, 1, 1, 1, 3, 1, 1, 1, 1, 5$$

and of 18 throws of dice g the series:

$$1, 3, 5, 2, 3, 1, 3, 4, 5, 2, 6, 4, 1, 3, 1, 3, 4, 2$$

The joint sampling, constructed by pairing the nth result of the first dice with the nth result of the second dice, is the following:

$$(4, 1), (1, 3), (2, 5), \underline{(1, 2)}, (1, 3), (1, 1), (6, 3), (1, 4), (3, 5),$$

$$\underline{(1, 2)}, (1, 6), (1, 4), (3, 1), (1, 3), (1, 1), (1, 3), (1, 4), (5, 2)$$

The experiment was a success. Indeed, among the 18 pairs, exactly two are of the form $(1, 2)$. Hence, as predicted, the frequency of $(1, 2)$ in this list is $2/18 = 1/9$. This outcome of the experiment on the pair of dice $\langle d, g\rangle$ can de formalized as follows:

$$\frac{\Delta, \Gamma \vdash \langle d, g\rangle^1 : (4 \times 1)^1 \ldots \Delta, \Gamma \vdash \langle d, g\rangle^{18} : (5 \times 2)^{18}}{\Delta, \Gamma \vdash \langle d, g\rangle_{18} : (1 \times 2)_{2/18}}$$

EXAMPLE 3.11.
Given two fair dice d, g whose distributions are encoded in respectively Δ, Γ. Again assuming independence, we could have

$$\Delta \times \Gamma = \{\langle x_d, x_g\rangle : 1 \times 1_{1/36}, \langle x_d, x_g\rangle : 1 \times 2_{1/36}, \ldots, \langle x_d, x_g\rangle : 6 \times 6_{1/36}\}$$

While $\langle d, g\rangle_{24} : (n \times m)_{\tilde{n}\cdot\tilde{m}}$ expresses the expected probability $\tilde{n} \cdot \tilde{m}$ of any pair $(n \times m)$ over 24 launches, for any specific such sequence of launches like the following: $(5,5)$, $(2,1)$, $(3,6)$, $(3,1)$, $(1,4)$, $(5,3)$, $(5,4)$, $(3,6)$, $(6,3)$, $(5,2)$, $(2,3)$, $(3,3)$, $(2,3)$, $(4,1)$, $(4,5)$, $(1,2)$, $(6,1)$, $(6,6)$, $(3,6)$, $(6,6)$, $(6,6)$, $(6,5)$, $(4,4)$, $(2,6)$, the frequency of any specific output results from an application of the sampling rule, such as:

$$\frac{\Delta, \Gamma \vdash \langle d, g\rangle^1 : (5 \times 5)^1 \qquad \ldots \qquad \Delta, \Gamma \vdash \langle d, g\rangle^{24} : (2 \times 6)^{24}}{\Delta \times \Gamma \vdash \langle d, g\rangle_{24} : (1 \times 2)_{1/24}}\ \text{sampling}$$

The rule I$\rightarrow$ says that, if assuming a variable has value α with probability a we can execute a process t, which, executed n times, shows output β with expected probability $\tilde{b}$, then we can construct the term $[x]t_n$, which taken in input a process of value α with probability a, will return a process of output value β with expected probability $\tilde{b}$ depending on a. Note that in $[a]\tilde{b}$, the sub-expression $[a]$ is only a notation to keep track of the theoretical probability assigned to the output in the antecedent. The corresponding elimination E$\rightarrow$ allows to verify the expected probability of β: it considers a term $[x]t_n$ of type $(\alpha \rightarrow \beta)$, which has expected probability $\tilde{b}$ depending on the probability a of value α, and providing in place of x a process u with expected probability $\tilde{a}$ formulated under a fresh random variable $y : \alpha_a$, it returns the process t that will display output β with an expected probability $\widetilde{a} \cdot \widetilde{b}$. The frequency of β in a sample of n experiments should therefore approximate this value.

EXAMPLE 3.12.
Consider a coin c, which we assume to be fair, i.e. whose assumed probability distribution on its outputs has the form

$$\Gamma := \{x_c : H_{0.5}, x_c : T_{0.5}\}$$

We might devise a system, which if the fair coin $c1$ returns *Head*, it will make another coin $c2$ return *Tail* with a frequency of 45% of the times over 1000 throws:

$$\frac{\Gamma, x_{c1} : H_{0.5} \vdash c2_{1000} : T_{\widetilde{0.45}}}{\Gamma \vdash [x]c2_{1000} : (H \rightarrow T)_{[0.5]\widetilde{0.45}}}\ \text{I}\rightarrow$$

In order to verify that the system is actually working in this way, we shall throw $c1$ a 1000 times. If we obtain the information that it returns heads a number of times which is close enough to the 50% of the total number of throws, we can use this information to obtain information on $c2$. Indeed, on the basis of the output of $c1$, $c2$ will return tails 45% of the times in which $c1$ returns heads. Therefore, we know that $c2$ will return tails the 45% of the 50% of the 1000 throws:

$$\frac{\vdash [x_{c1}]c2 : (H \rightarrow T)_{[0.5]\tilde{0.45}} \qquad x_{c1} : H_{0.5} \vdash c1_{1000} : H_{\widetilde{0.5}}}{x_{c1} : H_{0.5} \vdash c2_{1000}.[c1_{1000} : H] : T_{\widetilde{0.225}}}\ \text{E}\rightarrow$$

This final judgment says that we expect tails from the second coin and heads in the first coin with a frequency of 22.5% over 1000 times.

$$\frac{x : \alpha_{a_1} \vdash y : \beta_{b_1}, \ldots, x : \alpha_{a_m} \vdash y : \beta_{b_m} \qquad \sum_{i=1}^{m} b_i = 1}{\{\vdash [x]y : (\alpha \to \beta)_{[a_i]b_i} \mid 1 \leqslant i \leqslant m\}}\ \text{I-P}$$

$$\frac{\{\vdash [x]y : (\alpha \to \beta)_{[a_i]b_i} \mid 1 \leqslant i \leqslant m, \textstyle\sum_i b_i = 1\} \qquad \Gamma, x : \alpha_{a_i} \vdash t_n : \alpha_f}{\Gamma, x : \alpha_{a_i} \vdash (y.(t_n : \alpha_f)) : \beta_b}\ \text{E-P}$$

where

$$b = \frac{\overbrace{\binom{n}{n \cdot f} a_i^{f \cdot n} (1 - a_i)^{n - f \cdot n}}^{likelihood} \overbrace{b_i}^{prior}}{\underbrace{\binom{n}{n \cdot f} \sum_{j=1}^{m} a_j^{f \cdot n} (1 - a_j)^{n - f \cdot n} b_j}_{marginal}} = \frac{a_i^{f \cdot n} (1 - a_i)^{n - f \cdot n} b_i}{\sum_{j=1}^{m} a_j^{f \cdot n} (1 - a_j)^{n - f \cdot n} b_j}$$

FIGURE 7. Rules to update prior probabilities.

3.7 Bayesian updating

Random variables and experiments can be linked by the formulation of a form of $\to$ connective by which random variables are updated using the information generated by trials. The rules in Figure 7 express such process of updating prior probabilities in the light of new information provided by experiments. It corresponds to the use of the $\to$ connective introduced from a judgement of the form $x : \alpha_a^1, \ldots, x : \alpha_{a^n}^n \vdash y : \beta_b$, and eliminated with an additional premise of the form $\vdash t_n : \alpha_f$. The formula for b implements the Bayesian Updating in the specific case of a finite prior and a binomial likelihood function, which is the case we can express within our calculus. In particular, the prior probability associated with the point i under consideration is b_i, and its likelihood is $\binom{n}{n \cdot f} a_i^{f \cdot n} (1 - a_i)^{n - f \cdot n}$. The denominator marginalizes as usual over all points in the prior. Note that the binomial coefficient gets simplified out as it appears both in the numerator and denominator of the expression for b.

In the introduction, assuming that the probability b of variable y being assigned output β resulted from a sampling of n experiment within a model in which probabilities associated to random variables $x, \ldots, x^n$ to have values $\alpha, \ldots, \alpha^n$ is respectively $a, \ldots, a^n$, we introduce a list of implications which take all the exclusive hypotheses assigning a given probability a_i to variables x_i having value α_i and assigning a given variable y probability b to have value β; in the elimination: given such an implication with a list of exclusive hypotheses, and the result of a sampling on an experiment outputting α with frequency f, we update the prior probability b_i of β to b.

EXAMPLE 3.13.

In this example, we show how we can model the process of updating a discrete prior in a bayesian fashion. Let us assume we have five coins of three types. Coins of type A are fair, whereas coins of types B and C are biased, with a probability of 0.8 and 0.9 of landing heads respectively. We have 2 coins of type A, 1 coin of type B and 2 coins of type C. We blindly draw one of these five coins and toss it three times. Three coin tosses give that the coin lands heads 2 times. Our prior knowledge can

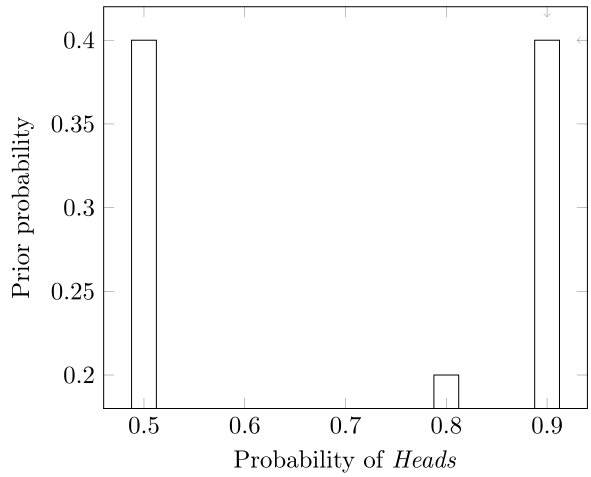


FIGURE 8. A graphical representation of the discrete prior in Example 3.13. The x axis represents the probability of the coin landing *Heads*; the y axis represents the corresponding prior probability.

be modelled with the three following sequents:

$$x : Heads_{0.5} \vdash y : Draw_{2/5}$$

$$x : Heads_{0.8} \vdash y : Draw_{1/5}$$

$$x : Heads_{0.9} \vdash y : Draw_{2/5}$$

that we denote by $\mathcal{P}$. They encode the discrete prior in Figure 8. We focus on the first of these three sequents, corresponding to the hypothesis that the coin we have drawn is of type A. We now consider data $\mathcal{D} = \{Heads, Heads, Tails\}$, which we represent with the sequent

$$x : Heads_{0.5} \vdash c_3 : Heads_{2/3}$$

and calculate the posterior probability of hypothesis $x : Heads_{0.5} \vdash y : Draw_{2/5}$ as follows:

$$\frac{\mathcal{P} \qquad x : Heads_{0.5} \vdash c_3 : Heads_{2/3}}{x : Heads_{0.5} \vdash y.(c_3 : Heads_{2/3}) : Draw_a}$$

where

$$a = \frac{0.5^2 \cdot (1 - 0.5) \cdot (2/5)}{0.5^2 \cdot (1 - 0.5) \cdot (2/5) + 0.8^2 \cdot (1 - 0.8) \cdot (1/5) + 0.9^2 \cdot (1 - 0.9) \cdot (2/5)} \approx 0.46$$

Therefore, the hypothesis that $x : Heads_{0.5}$ has increased its probability from $2/5$ to 0.46.

Note that once the prior has been updated in the conclusion of the previous tree, it is still possible to introduce again the implication

$$\vdash [x](y.c_3 : Heads_{2/3}) : (Heads \to Draw)_{[0.5]a}$$

and update it again with some new data.

$$\frac{\Gamma \vdash x : \alpha_a \qquad \Delta \vdash u_n : \alpha_f \qquad \mid a - f \mid \leqslant \epsilon(n)}{\Gamma, \Delta \vdash \mathit{Trust}(u_n : \alpha_f)}\ \mathrm{IT}$$

$$\frac{\Gamma \vdash \mathit{Trust}(u_n : \alpha_f)}{\Gamma, x_u : \alpha_{[a-\epsilon(n), a+\epsilon(n)]} \vdash u_n : \alpha_f}\ \mathrm{ET}$$

$$\frac{\Gamma \vdash x : \alpha_a \qquad \Delta \vdash u_n : \alpha_f \qquad \mid a - f \mid > \epsilon(n)}{\Gamma, \Delta \vdash \mathit{UTrust}(u_n : \alpha_f)}\ \mathrm{IUT}$$

$$\frac{\Gamma \vdash \mathit{UTrust}(u_n : \alpha_f)}{\Gamma, x_u : \alpha_{[0,1]-[a-\epsilon(n), a+\epsilon(n)]} \vdash u_n : \alpha_f}\ \mathrm{EUT}$$

FIGURE 9. Trust fragment.

3.8 Trust fragment

TPTND is designed to verify trustworthiness of probabilistic computations through the rules in Figure 9.

The intuition we want to implement in the calculus is that the control of a certain distance between a distribution of reference and the observed probability of an event is formally checkable. The choice of a confidence interval with respect to the expected probability of an event is a design choice that can be otherwise devised, see e.g. [35] for a version using KL as a distance in order to evaluate for biases. Moreover, while the rule mimics in a certain sense standard statistical inference properties, the calculus allows to combine such analysis with an automated deduction approach, which also allows to define some useful structural properties on the derivation mimicking the probabilistic process, thereby allowing formal properties to be proven in the next section. Formally, we interpret the above as follows. An experiment $\Delta \vdash u_n : \alpha_f$ where Δ is possibly unknown—that is, Δ is of the form $\{x : \alpha_{[0,1]}, \ldots, x : \omega_{[0,1]}\}$ as obtained by the 'unknown' rule in Figure 3—should be evaluated according to some metric against its intended or expected model, in turn expressed as a transparent distribution Γ. If the difference between the frequency f of output α in n experiments u and its theoretical probability a in the relevant intended and transparently formulated distribution Γ remains below a critical threshold, parametric with respect to the number of samples, then the process u outputting α with frequency f can be considered trustworthy (rule IT). The parametric threshold $\epsilon(n)$ is domain-specific and it may depend on the application. Then one can start reasoning on processes considered trustworthy: if $u_n : \alpha_f$ is trustworthy under distribution Γ, then there must exist an interval of probability values $[a - \epsilon(n), a + \epsilon(n)]$ for output $\alpha \in \Gamma$ within which $u_n : \alpha_f$ can be correctly sampled from Γ (rule ET). Note that Contraction from Section 3.9 can be applied to select the most appropriate value within such range.

Dual rules for negative trust can be devised. Rule IUT for the introduction of negative trust (or Untrust) expresses the failure on the condition for trust: if the value set for the difference between the frequency f of the output α in the experiment and the theoretical probability a in the relevant intended and transparently formulated distribution Γ is surpassed, the process can be labeled untrustworthy. The rule EUT serves the purpose of eliminating the negative trust function, i.e. it allows to start reasoning on processes considered untrustworthy: if $u_n : \alpha_f$ is considered untrustworthy under distribution Γ, then there exists a family of probability values in the interval $[0, 1]$ excluding the interval $[a - \epsilon(n), a + \epsilon(n)]$ for output $\alpha \in \Gamma$ within which $u_n : \alpha_f$ can be correctly sampled from

Γ. Hence, the rule identifies the range of theoretical values that make f a fair frequency, and again Contraction from Section 3.9 can be applied to select the most appropriate value within such range.

Note, in particular, that the first and second premise of each introduction rule allow for distinct distributions. While it is required that the first context expresses a distribution in which it is included the theoretical probability for a program of output α, this means that it is possible to evaluate for trustworthiness against it a program u of the same type even if the latter is executed under a distinct, and possibly opaque or even unknown distribution Δ. The easy case is obviously where $\Gamma \equiv \Delta$, but otherwise Δ might be inaccessible (i.e. formally unknown) while Γ represents the intended or assumed model of u.

Trust evaluation, as mentioned, is parametric in the number of executions, and this renders it a potentially dynamic process. For instance, one may take $\epsilon(n)$ to correspond to the 95% confidence interval under the Normal approximation to the Binomial Distribution. Then it is possible to derive the trustworthiness of some untrustworthy process (and vice versa) with some (very small) probability. The probability of this happening can be made arbitrarily small by extending the sample associated with the process.

EXAMPLE 3.14.
Let u be a die, which we have been provided with and we know nothing about, its distribution being denoted as Δ; and let Γ be the distribution defined in Example 3.3, representing a fair die. In this example, we focus on output 5. Let's assume that experimenting under the unknown context Δ gives us the following result:

$$\frac{\cdots \qquad \cdots \qquad \cdots}{\Delta \vdash u_{10} : 5_{1/2}}\ \text{sampling}$$

Our aim is to decide whether the system u can be assumed to represent a sample under a fair distribution over outputs $\{1, 2, \ldots, 6\}$, i.e. whether u can be therefore considered safe to use during a betting game. To this aim, we use the Introduction of Negative Trust, where we use the (exact) Binomial 95% confidence interval to set our threshold:

$$\frac{\Gamma \vdash x : 5_{1/6} \qquad \Delta \vdash u_{10} : 5_{1/2} \qquad 1/6 \notin [0.19, 0.81]}{\Gamma, \Delta \vdash \mathit{UTrust}(u_{10} : 5_{1/2})}\ \text{IUT}$$

that is, we do not trust process u to be extracted from a distribution that sufficiently approximates Γ.

Obviously, things may change as new evidence is collected. Let suppose we are provided with some more experimental evidence about process u, we then can use rule 'update':

$$\frac{\dfrac{\Delta \vdash u_{10} : 5_{1/2} \qquad \Delta \vdash u_{20} : 5_{3/20}}{\Delta \vdash u_{30} : 5_{4/15}}\ \text{upd} \qquad \Gamma \vdash x : 5_{1/6} \qquad 1/6 \in [0.12, 0.46]}{\Gamma, \Delta \vdash \mathit{Trust}(u_{30} : 5_{4/15})}\ \text{IT}$$

i.e. based on a total of 30 runs of process u, now we trust it to be extracted from the transparent and fair distribution Γ.

EXAMPLE 3.15.
Suppose we are given a commercial, closed-source software to automatically shortlist CVs according to a set of criteria. To this aim, we consider the output of the classification algorithm to fall into one of the following four categories: (1) male, shortlisted, (2) male, not shortlisted, (3) female, shortlisted,

$$\frac{\Gamma \vdash z : \beta_b \qquad \Delta \vdash y : \gamma_g \qquad \Gamma \perp\!\!\!\perp \Delta}{\Gamma, \Delta \vdash z : \beta_b}\ \text{Weakening}$$

$$\frac{\Gamma, x_t : \alpha_{a_1}, \ldots, x_t : \alpha_{a_n} \vdash t_n : \beta_f}{\Gamma, x_t : \alpha_{fun(a_1,\ldots,a_n)} \vdash t_n : \beta_f}\ \text{Contraction}$$

FIGURE 10. Structural rules.

(4) female, not shortlisted. To verify that the underlying algorithm does not have an inherent gender bias, we test whether we trust the frequency of any output of interest over these four classes to reflect the actual gender distribution in the current Italian population, as encoded in a distribution Γ. A proof that this is indeed the case would produce a certificate in the form of a natural deduction proof, that would go as follows:

$$\frac{\dfrac{\{\} :: \mathit{distribution}}{\{x : \mathit{female}_{0.52}\} :: \mathit{distribution}}\ \text{extend} \qquad \{\} \vdash c_{10} : \mathit{female}_{0.3} \qquad 0.52 \in [0.0667, 0.6525]}{\{x : \mathit{female}_{0.52}\} \vdash \mathit{Trust}(c_{10} : \mathit{female}_{0.3})}\ \text{IT}$$

where we assumed a 95% confidence interval. Note that a shortlist selection that would provide strictly less than $2/10$ or more than $9/10$ of female candidates would be considered untrustworthy according to this model. Note that if we sampled 30 CVs instead, shortlisting 7 female CVs, we would derive untrustworthiness:

$$\frac{\dfrac{\{\} :: \mathit{distribution}}{\{x : \mathit{female}_{0.52}\} :: \mathit{distribution}}\ \text{extend} \qquad \{\} \vdash c_{30} : \mathit{female}_{7/30} \qquad 0.52 \notin [0.1, 0.42]}{\{x : \mathit{female}_{0.52}\} \vdash \mathit{UTrust}(c_{30} : \mathit{female}_{7/10})}\ \text{IUT}$$

We would in fact consider untrustworthy any number of shortlisted CVs by female candidates strictly smaller than 10 or bigger than 21. Assume now a different population of interest:

$$\frac{\dfrac{\{\} :: \mathit{distribution}}{\{x : \mathit{female}_{0.7}\} :: \mathit{distribution}}\ \text{extend} \qquad \{\} \vdash c_{10} : \mathit{female}_{3/10} \qquad 0.7 \notin [0.0667, 0.6525]}{\{x : \mathit{female}_{0.7}\} \vdash \mathit{UTrust}(c_{10} : \mathit{female}_{3/10})}\ \text{IUT}$$

3.9 Structural rules

Structural rules in a proof-theoretic setting are used to determine valid operations on contexts. Under our interpretation, such rules are needed to establish coherent manipulation of assumptions on probability distributions under which either other random variables are assigned a value, or processes (programs, experiments) are sampled. In particular, our structural rules are essential to two aims:

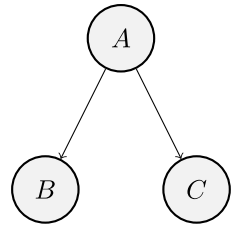
1. define the admissible extensions of value assignments to random variables through the 'extend' type definition;
2. define the extraction of plausible values of relevant random variables from experiments performed under multiple hypotheses and, more generally, under unknown distributions.

We provide constrained versions of the standard rules holding for our system in Figure 10.

The Weakening rule expresses the following principle: given the probability of a random variable z to have value β is b, provided the value assigned to some variable x is α; if the latter assignment

is independent of a distribution Δ, then the starting conditional probability can be extended by additional assumptions Δ and the probability b is left unaltered. In other words: a given probability distribution can be extended at will without inferring new probabilities, as long as no new dependencies are introduced. To show this case, and illustrate how Weakening by a confounding variable modifies the inferred output, we illustrate a number of examples.

EXAMPLE 3.16 (Confounding).
Consider the following graph:



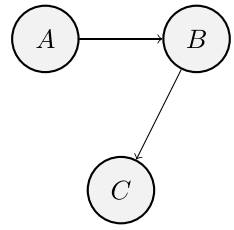
This graphical model means that $P(A,B,C) = P(B \mid A)P(C \mid A)P(A)$, i.e. A is a *confounding* variable as B and C depend on A but do not directly influence each other. In this example, A, B and C are probabilistic routines with outputs 0 and 1. A is a 'master' process in that it can decide to force 'slave' processes B and C to output specific values. A has a 50% chance of generating output 1 and 50% chance of producing output 0; if A outputs 0, it forces B and C to output 0; if A outputs 1, B and C both have 50% probability of producing 1 and 50% probability of producing 0. In this case, A acts as a *confounder*, in the sense that one may observe a dependency between B and C that, however, is not due to an explicit communication between the two processes.

In TPTND this false dependency is discarded, as the dependency from A is clearly stated, while B, C are shown to be coordinated processes by logical conjunction under the same assumption. The entire graph is expressed analytically by the following trees (where the shared context trivially satisfies our definition of $\perp\!\!\!\perp$ since we can derive $\{x_A : 0, x_A : 1\} :: \textit{distribution}$):

$$\dfrac{\dfrac{x_A : 0 \vdash y_B : 0_{\tilde{1}} \qquad x_A : 0 \vdash z_C : 0_{\tilde{1}}}{x_A : 0 \vdash \langle y_B, z_C \rangle : (0 \times 0)_{\tilde{1}}}\,\mathrm{I}\times}{\vdash [x_A]\langle y_B, z_C \rangle : (0 \to (0 \times 0))_{\tilde{1}}}\,\mathrm{I}\to$$

$$\dfrac{\dfrac{x_A : 1 \vdash y_B : 1_{\tilde{0.5}} \qquad x_A : 1 \vdash z_C : 1_{\tilde{0.5}}}{x_A : 1 \vdash \langle y_B, z_C \rangle : (1 \times 1)_{\tilde{0.25}}}\,\mathrm{I}\times}{\vdash [x_A]\langle y_B, z_C \rangle : (1 \to (1 \times 1))_{\tilde{0.25}}}\,\mathrm{I}\to$$

EXAMPLE 3.17 (Chain).
Consider the following different graph:



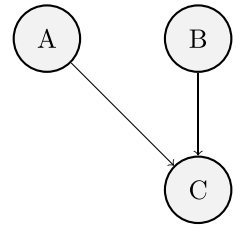
This graphical model means that $P(A,B,C) = P(C \mid B)P(B \mid A)P(A)$, i.e. C is a variable depending on B, whose effect in turn depends on A. In this example, A, B and C are probabilistic routines with outputs 0 and 1. A is a 'master' process in that it can decide to force 'slave' process B and this in turn process C to output specific values. A has a 50% chance of generating output 1 and 50% chance of producing output 0; if A outputs 0, it forces B to output 0; under the 50% probability that B outputs 0, C has 50% probability of producing 1 and 50% probability of producing 0.

$$\dfrac{\dfrac{y_B.[x_A:0]:0_a \vdash z_C:1_{\widetilde{0.5}}}{\vdash [y_B.[x_A:0]]z_C:(0\to 1)_{[a]\widetilde{0.5}}}\,\mathrm{I}\!\to \qquad \dfrac{\dfrac{x_A:0\vdash y_B:0_{\tilde{1}}}{\vdash [x_A]y_B:(0\to 0)_{\tilde{1}}}\,\mathrm{I}\!\to \qquad x_A:0\vdash x_A:0_{\tilde{a}}}{x_A:0\vdash y_B.[x_A:0]:0_{\tilde{a}\cdot\tilde{1}}}\,\mathrm{E}\!\to}{x_A:0\vdash [y_B.[x_A:0]]z_C:(0\to 1)_{[a]\widetilde{0.5}}}\,\text{Weakening}$$

Note that an evaluation of the probability of the variable z_C having value 1 depends ultimately on the value that will be assigned to variable a in the E$\to$ by the premise $x_A : 0 \vdash x_A : 0_{\tilde{a}}$. Note moreover how the condition of C is expressed by a probability $P(B \mid A)$ and this corresponds in the tree above to the fact that the context of z_C is of the form $y_B.[x_A : 0] : 0_a$ with probability a.

The previous example is illustrative of a variable y_B, which is not independent of x_A, hence the latter could not be used to weaken the context of z_C. On the other hand, the following example introduces the situation of a variable y_C dependent on x_A, and for which one could also state the dependency from a distinct variable x_B:

EXAMPLE 3.18 (Collider).
Consider now the following example:



In this example, the output of variable C is a collider of A and B. Again, A, B and C are probabilistic routines with outputs 0 and 1. A is a 'master' process in that it can decide to force 'slave' process C: A has a 50% chance of generating output 1 and 50% chance of producing output 0:

$$\dfrac{x_A:0_{0.5},x_A:1_{0.5}\vdash x_A:0_{0.5} \qquad x_A:0_{0.5},x_A:1_{0.5}\vdash x_A:1_{0.5}}{x_A:0_{0.5},x_A:1_{0.5}\vdash x_A:(0+1)_{0.5+0.5}}\,\mathrm{I}+$$

if A outputs 0, it forces C to output 0, while if A outputs 1, C will have only 50% chance to output 1:

$$\dfrac{\dfrac{\dfrac{x_A:0_{0.5}\vdash y_C:0_{\tilde{1}}}{\vdash [x_A]y_C:0\to 0_{[0.5]\tilde{1}}}\,\mathrm{I}\!\to \qquad x_A:0_{0.5}\vdash x_A:0_{0.5}}{x_A:0_{0.5}\vdash [x_A]y_C:0_{\widetilde{0.5}}}\,\mathrm{E}\!\to \qquad \dfrac{\dfrac{x_A:0_{0.5}\vdash y_C:1_{\tilde{0}}}{\vdash [x_A]y_C:(1\to 1)_{[0.5]\tilde{0}}}\,\mathrm{I}\!\to \qquad x_A:0_{0.5}\vdash x_A:0_{0.5}}{x_A:0_{0.5}\vdash [x_A:0_{0.5}]y_C:1_{\tilde{0}}}\,\mathrm{E}\!\to}{x_A:0_{0.5}\vdash [x_A:0_{0.5}]y_C:(0+1)_{\widetilde{0.5}}}\,\mathrm{I}+$$

$$\dfrac{\dfrac{\dfrac{x_A:1_{0.5}\vdash y_C:1_{\widetilde{0.5}}}{\vdash [x_A]y_C:(1\to 1)_{[0.5]\widetilde{0.5}}}\,\mathrm{I}\!\to \qquad x_A:1_{0.5}\vdash x_A:1_{0.5}}{x_A:1_{0.5}\vdash y_C.[x_A:1_{0.5}]:1_{\widehat{0.25}}}\,\mathrm{E}\!\to \qquad \dfrac{\dfrac{x_A:1_{0.5}\vdash y_C:0_{\widetilde{0.5}}}{\vdash [x_A]y_C:(1\to 0)_{[0.5]\widetilde{0.5}}}\,\mathrm{I}\!\to \qquad x_A:1_{0.5}\vdash x_A:1_{0.5}}{x_A:1_{0.5}\vdash y_C.[x_A:1_{0.5}]:0_{\widehat{0.25}}}\,\mathrm{E}\!\to}{x_A:1_{0.5}\vdash y_C.[x_A:1]:(0+1)_{\widetilde{0.5}}}\,\mathrm{I}+$$

Note that here the total probability of C outputting $(0+1)$ is 1, but this is expressed under different conditions, namely of process A outputting either 0 or 1. While we cannot derive this, by inspection of the two derivation trees, we can select the probability of $\vdash y_C.[x_A : 0] : 0_{0.5}$ in the leftmost branch of the first tree and the probability of $\vdash y_C.[x_A : 1] : 0_{0.25}$ and compute the overall probability of C having output 0 when A has value $(0 + 1)$ is $(0.5 + 0.25) = 0.75$. Similarly, the probabilities of $\vdash y_C.[x_A : 0] : 1_0$ and $\vdash y_C.[x_A : 1] : 1_{0.25}$ gives us the overall probability of output 1 from C as $(0 + 0.25) = 0.25$. Note, however, that this computation is not derivable.

Now consider that B is also a 'master' process in that it can decide to force 'slave' process C: B has a 50% chance of generating output 1 and 50% chance of producing output 0:

$$\frac{x_B : 0_{0.5} \vdash x_B : 0_{0.5} \qquad x_B : 1_{0.5} \vdash x_B : 1_{0.5}}{x_B : 0_{0.5}, x_B : 1_{0.5} \vdash x_B : (0+1)_{0.5+0.5}}\ \mathrm{I}+$$

if B outputs 1, it forces C to output 0, while if B outputs 0, C will have only 50% chance to output 1:

$$\dfrac{\dfrac{\dfrac{x_B : 1_{0.5} \vdash y_C : 0_{\widetilde{1}}}{\vdash [x_B : 1_{0.5}]y_C : (1 \to 0)_{[0.5]\widetilde{1}}}\ \mathrm{I}\to \qquad x_B : 1_{0.5} \vdash x_B : 1_{0.5}}{x_B : 1_{0.5} \vdash y_C.[x_B : 1_{0.5}] : 0_{\widetilde{0.5}}}\ \mathrm{E}\to \qquad \dfrac{\dfrac{x_B : 1_{0.5} \vdash y_C : 1_{\widetilde{0}}}{\vdash [x_B : 1_{0.5}]y_C : (1 \to 1)_{[0.5]\widetilde{0}}}\ \mathrm{I}\to \qquad x_B : 1_{0.5} \vdash x_B : 1_{0.5}}{x_B : 1_{0.5} \vdash y_C.[x_B : 1_{0.5}] : 1_{\widetilde{0}}}\ \mathrm{E}\to}{x_B : 1_{0.5} \vdash y_C.[x_B : 1_{0.5}] : (0+1)_{\widetilde{0.5}}}\ \mathrm{I}+$$

$$\dfrac{\dfrac{\dfrac{x_B : 0_{0.5} \vdash y_C : 1_{\widetilde{0.5}}}{\vdash [x_B : 0_{0.5}]y_C : (0 \to 1)_{[0.5]\widetilde{0.5}}}\ \mathrm{I}\to \qquad x_B : 0_{0.5} \vdash x_B : 0_{0.5}}{x_B : 0_{0.5} \vdash y_C.[x_B : 0_{0.5}] : 1_{\widetilde{0.25}}}\ \mathrm{E}\to \qquad \dfrac{\dfrac{x_B : 0_{0.5} \vdash y_C : 0_{\widetilde{0.5}}}{\vdash [x_B : 0_{0.5}]y_C : (0 \to 0)_{[0.5]\widetilde{0.5}}}\ \mathrm{I}\to \qquad x_B : 0_{0.5} \vdash x_B : 0_{0.5}}{x_B : 0_{0.5} \vdash y_C.[x_B : 0_{0.5}] : 0_{\widetilde{0.25}}}\ \mathrm{E}\to}{x_B : 0_{0.5} \vdash y_C.[x_B : 0_{0.5}] : (0+1)_{\widetilde{0.5}}}\ \mathrm{I}+$$

Note that here the total probability of C outputting $(0 + 1)$ is again 1, but this is expressed under different conditions, namely of process B outputting either 0 or 1. By inspection of the two derivation trees, we can select the probability of $\vdash y_C.[x_B : 1] : 0_{0.5}$ in the leftmost branch of the first tree and the probability of $\vdash y_C.[x_B : 0] : 0_{0.25}$ and compute the overall probability of output 0 as $(0.5 + 0.25) = 0.75$. Similarly, the probabilities of $\vdash y_C.[x_B : 1] : 1_0$ and $\vdash y_C.[x_B : 0] : 1_{0.25}$ gives us the overall probability of output 1 from C dependent from B is $(0 + 0.25) = 0.25$. Note, however, that this computation is not derivable.

In a similar vein, to compute the overall probability that $y_C : 0$ given $x_A : 1$ and $x_B : 1$, we can inspect the rightmost branch of the second of the previous two trees and the leftmost branch of the first of the two trees above: $(0.25 + 0.5) - (0.25 \cdot 0.5) = 0.625$. Or the overall probability that $y_C : 0$ given $x_A : 0$ and $x_B : 0$, inspecting the rightmost branch of the second of the previous two trees and the leftmost branch of the first of the two trees above: $(0.5 + 0.25) - (0.5 \cdot 0.25) = 0.625$. Again, this computation is available by inspection of the derivation trees, but not directly derivable.

On the other hand, if we formulate y_C under the condition $x_A : (0 + 1)$ (respectively $x_B : (0 + 1)$), we can express the direct dependency of C having output $(0 + 1)$ deriving it from the two disjuncts:

$$\frac{x_A : (0+1)_1 \vdash y_C : 0_{0.75} \qquad x_A : (0+1)_1 \vdash y_C : 1_{0.25}}{x_A : (0+1)_1 \vdash y_C : (0+1)_1}\ \mathrm{I}+$$

$$\frac{x_B : (0+1)_1 \vdash y_C : 0_{0.75} \qquad x_B : (0+1)_1 \vdash y_C : 1_{0.25}}{x_B : (0+1)_1 \vdash y_C : (0+1)_1}\ \mathrm{I}+$$

$$\frac{x_A : (0+1)_1, x_B : (0+1)_1 \vdash y_C : 0_{0.75} \qquad x_A : (0+1)_1, x_B : (0+1)_1 \vdash y_C : 1_{0.25}}{x_A : (0+1)_1, x_B : (0+1)_1 \vdash y_C : (0+1)_1}\ \mathrm{I}+$$

The Contraction rule expresses a form of plausibility test on hypotheses. Consider a probability distribution Γ that includes more than one distinct hypothesis on the theoretical probability a of some

process with output value α, on which the evaluation of an experiment t with output β with frequency f depends. The general case will be that Γ is unknown, so that for any output variable $\alpha \in \Gamma$, the entire set of possible probability values $[0, 1]$ is assigned. A contraction on such distribution is a function $fun[0, 1]$, which extracts one value $x_t : \alpha_a$. The function *fun* can be chosen at will; a sensible example is the Maximum Likelihood function $ML(a, a')$, which returns the value making the frequency f the most plausible as follows:

$$\frac{\Gamma, x_t : \alpha_a, x_t : \alpha_{a'} \vdash t_n : \alpha_f}{\Gamma, x_t : \alpha_{ML(a,a')} \vdash t_n : \alpha_f}\ \text{Contraction}$$

where $ML(a, a') = \text{argmax}_{x \in \{a,a'\}}(x^{\frac{f}{n}}(1-x)^{1-\frac{f}{n}})$.

EXAMPLE 3.19 (Contraction).
Let Δ be the unknown distribution associated to a die d with possible outputs $\{1, \ldots, 6\}$, i.e.:

$$\frac{1 :: \mathit{output}, \ldots, 6 :: \mathit{output}}{\Delta :: \mathit{distribution}}\ \text{unknown}$$

Technically, Δ could be re-written as the distribution that associates to each output $\{1, \ldots, 6\}$ the value $[0, 1]$. Consider now 10 throws of the die with sequence of outputs 1, 6, 6, 2, 6, 3, 4, 6, 1, 1. Then,

$$\frac{}{\Delta \vdash d_{10} : 6_{4/10}}\ \text{sampling}$$

Applying contractions multiple times can be abbreviated as follows:

$$\frac{\Delta \vdash d_{10} : 6_{4/10}}{\Delta \setminus \{x : 6_a \mid a \notin [0, 0.4) \cup (0.4, 1]\} \vdash d_{10} : 6_{4/10}}\ \text{Contraction}$$

where the context in the conclusion denotes the set obtained by the unknown distribution in the premise, removing from it the declaration that assigns any probability different than 0.4 to x having output 6, i.e. $\Delta \setminus \{x : 6_a \mid a \in [0, 0.4) \cup (0.4, 1]\} = \{x : 1_{[0,1]}, x : 2_{[0,1]}, x : 3_{[0,1]}, x : 4_{[0,1]}, x : 5_{[0,1]}, x : 6_{0.4}\}$.

EXAMPLE 3.20.
Consider the distribution Γ representing a fair coin, i.e. with 50% probability of landing Heads when tossed. Assume Δ expresses a coin that we know nothing about, i.e. Δ is unknown, which when tossed 10 times lands Heads only once. Under the confidence range expressed by $[0.0025, 0.4450]$, the coin is untrustworthy.

$$\frac{\Gamma, x : H_{0.5} :: \mathit{distribution} \qquad \Delta \vdash c_{10} : H_{1/10} \qquad 0.5 \notin [0.0025, 0.4450]}{\Gamma, x : H_{0.5}, \Delta \vdash \mathit{UTrust}(c_{10} : H_{1/10})}\ \text{IUT}$$

We can now decide to assume the coin c to be biased with a probability of only 0.3 of landing Heads when tossed: this can be done by applying an appropriate instance of contraction. Under this new assumption, the result of 10 tosses of the coin landing Heads only once is trustworthy.

$$\frac{\dfrac{x_c : H_{[0,1]}, \ldots, x_c : H_{[0,1]} \vdash c_{10} : H_{1/10}}{x_c : H_{0.3} \vdash c_{10} : H_{1/10}}\ \text{contraction} \qquad 0.3 \in [0.0025, 0.4450]}{x : H_{0.3} \vdash \mathit{Trust}(c_{10} : H_{1/10})}\ \text{IT}$$

Finally, the Cut rule is admissible as a detour of I→ and E→ rules when both the major and the minor premises result from sampling:

$$\dfrac{\dfrac{\Delta, x:\alpha_a \vdash t_n : \beta_{\tilde{b}}}{\Delta \vdash [x]t_n : (\alpha \to \beta)_{[a]\tilde{b}}}\,\mathrm{I}\to \qquad \Gamma \vdash u_n : \alpha_{\tilde{a}}}{\Gamma, \Delta \vdash t_n.[u:\alpha] : \beta_{\widetilde{a \cdot b}}}\,\mathrm{E}\to$$

4 Metatheory

Our meta-theoretical analysis aims at showing properties of computational terms with respect to expected behavior, as well as considering inference trees of TPTND to establish how logical rules behave with respect to inferring trustworthiness. Hence, a judgement $\Gamma \vdash t_n : \alpha_f$ is considered in the light of the computational transformations formalized by the relation $\mapsto_p$ for an expression of the form $t_n : \alpha_f$.

Our first step is to reduce expected probabilities when they depend on deterministic outputs. Deterministic substitution has the following meaning: consider a process t that has output β with expected probability $\tilde{b}$, under the assumption of a variable having value of type α with probability a; whenever output α is deterministically obtained by a process u, process t has output β with probability $\tilde{b}$ under the condition that u is executed and α is obtained.

THEOREM 4.1 (Deterministic Substitution).
If $x:\alpha_a \vdash t : \beta_{\tilde{b}}$ and $x_u : \alpha_a \vdash u:\alpha$, then there exists a term $s = t.\mathcal{C}_1[u]. \ldots .\mathcal{C}_n[u]$ where $1 \leq n$ and each $\mathcal{C}_i[u]$ is obtained by zero or more logical eliminations on u, and $\vdash s : \beta_{\tilde{b}}$.

PROOF. By structural induction on the antecedent of the first premise. To show the reduction to output $\beta_{\tilde{b}}$, for each case of $x:\alpha_a$ in the induction we first build one or more dependent terms of the form $[x]t : (\alpha' \to \beta)_{[a]\tilde{b}}$, then consider the deterministic output $u:\alpha$ in the second derivation and use it to infer the application $t.\mathcal{C}_1[u]. \ldots .\mathcal{C}_n[u]$:

- For α atomic, the case is immediate;
- For $\alpha \equiv \gamma \times \delta$, there are terms $\langle v, z\rangle$ obtained from independent distributions such that the following tree is a reduction to the base case:

$$\dfrac{\dfrac{\dfrac{\dfrac{x_v:\gamma, y_z:\delta \vdash t : \beta_{\tilde{b}}}{y_z:\delta \vdash [x]t : (\gamma \to \beta)_{[1]\tilde{b}}}\,\mathrm{I}\to \quad \dfrac{x_v:\gamma, y_z:\delta \vdash \langle v, z\rangle : (\gamma \times \delta)}{x_v:\gamma, y_z:\delta \vdash fst(\langle v, z\rangle) : \gamma}\,\mathrm{E}\times}{x_v:\gamma, y_z:\delta \vdash t.[fst(\langle v, z\rangle) : \gamma] : \beta_{\tilde{b}}}\,\mathrm{E}\to \quad \dfrac{x_v:\gamma, y_z:\delta \vdash \langle v, z\rangle : (\gamma \times \delta)}{x_v:\gamma, y_z:\delta \vdash snd(\langle v, z\rangle) : \delta}\,\mathrm{E}\times}{x_v:\gamma, y_z:\delta \vdash t.[fst(\langle v, z\rangle) : \gamma].[snd(\langle v, z\rangle) : \delta] : \beta_{1\cdot\tilde{b}}}\,\mathrm{E}\to$$

- For $\alpha \equiv \gamma + \delta$ is a metavariable for the distribution of possible exclusive outputs of a given term u. Then either $u : \gamma$ or $u : \delta$ (and respectively the other term has probability $= 0$); then use an additional assumption of type $\mathsf{O}^{\perp}$ to select either one through $E+$, and the base case applies.
- For $\alpha \equiv \gamma \to \gamma$ (as an instance of the rule *update* with unary executions and deterministic outputs), there are terms $u \equiv [y]v$ and v such that the following tree is a reduction to the base case:

$$\dfrac{\dfrac{y_v : \gamma \vdash t : \beta_{\tilde{b}}}{\vdash [y]t : (\gamma \to \beta)_{\tilde{b}}}\,\mathrm{I}\to \dfrac{\dfrac{y_v : \gamma \vdash v : \gamma}{\vdash [y]v : \gamma \to \gamma}\,\mathrm{I}\to y_v : \gamma \vdash v : \gamma}{y_v : \gamma \vdash v.[v : \gamma] : \gamma}\,\mathrm{E}\to}{y_v : \gamma \vdash t.[v.[v : \gamma] : \gamma] : \beta_{1\cdot\tilde{b}}}\,\mathrm{E}\to$$

□

We consider the computational semantics of samplings to prove properties about frequencies and expected probabilities for such terms. Recall that a computational term t is said to probabilistically reduce to a term t' if t can be transformed into t' according to one of the rewriting rules from Figures 1,2. First, we show that for atomic terms obtained by applications of reduction rules in Figure 1, the value of the frequency of any given output approximates, in the limit, its expected probability.

THEOREM 4.2.
For any atomic term t such that $\Gamma, x_t : \alpha_a \vdash t_n : \alpha_{\tilde{a}}$, and sequence of reductions

$$\mathcal{L}_1, t_{n_0} : \alpha_{f_0} \mapsto \cdots \mapsto \mathcal{L}_m, t_{n_m} : \alpha_{f_m} \mapsto \ldots$$

with $n_0 < \cdots < n_m < \ldots$ that only contains event, sampling and update rule applications, let us define ν as the function such that $\nu_t(m) = f_m$. We have then that

$$\lim_{m \to \infty} |a - \nu_t(m)| = 0$$

with probability 1.

PROOF. Let us define a random variable X with possible values 0 and 1 and expected value $E(X) = a$. A sequence of random variables $X_1, \ldots, X_n$ then encodes the count of how many times an output of type α is obtained during an experiment consisting of n executions of the term t. The expected value $E(X) = a$ indeed exactly corresponds to the probability that an application of the event$^{\mapsto t}$ rule produces a typed term of the form $t : \alpha$ indicating that t yielded an output of type α. This is due to the fact that, since $x_t : \alpha_a$ is in our context and our context is meant to formalize the behaviour of the process t, the event$^{\mapsto t}$ rule produces a typed term $t : \alpha$ with probability a.

Notice that a sequence $X_1, \ldots, X_n$ such that $\frac{X_1, \ldots, X_n}{n} = E(X) = a$ encodes a count of the number of outcomes of type α during an experiment consisting of n executions of t such that exactly a of the total outcomes was of type α.

In case all random variables $X_1, \ldots, X_n$ have the same expected value $E(X)$, the strong law of large numbers states that

$$\lim_{n \to \infty} \left(\frac{X_1 + \ldots + X_n}{n} \right) = E(X)$$

with probability 1. This, hence, holds for any sequence $X_1, \ldots, X_n$ of occurrences of the random variable X defined above.

Since

- each sequence $X_1, \ldots, X_n$ such that $\frac{X_1, \ldots, X_n}{n} = E(X) = a$ corresponds to an experiment consisting of n executions of t such that exactly a of the total outcomes was of type α,
- the closer $\frac{X_1, \ldots, X_n}{n}$ gets to $E(X) = a$, the closer the frequency of outcomes of type α produced by the event$^{\mapsto t}$ rules applications gets to the expected probability a,
- the value f_m of typed atomic terms of the form $t_m : \alpha_{f_m}$ produced by term evaluation rules directly and exclusively depends on the frequency with which a term $t : \alpha$ is obtained by the event$^{\mapsto t}$ rule;

the strong law of large numbers implies that, with probability 1, for m approaching infinity, any reduction

$$\mathcal{L}_1, t_{n_0} : \alpha_{f_0} \mapsto \cdots \mapsto \mathcal{L}_m, t_{n_m} : \alpha_{f_m}$$

with $n_0 < \cdots < n_m$ is such that f_m is closer and closer to a. Therefore, by definition of ν, we have

$$\lim_{m\to\infty} |a - \nu_t(m)| = 0$$

with probability 1. □

Before closing this result under logical operations, let us introduce some standard notation.

DEFINITION 4.3.
By $\mapsto^*$, as per standard practice, we denote the reflexive and transitive closure of the $\mapsto$ relation.

We can now close Theorem 4.2 under those logical operations that are related to the the reduction rules in Figure 2.

THEOREM 4.4.
For any term t such that $\Gamma \vdash t_{n_0} : \alpha_{f_0}$ and $\Gamma \vdash t_{n_0} : \alpha_{\tilde{a}}$ and sequence of reductions

$$\mapsto^* \mathcal{L}_1, t_{n_0} : \alpha_{f_0} \mapsto^* \cdots \mapsto^* \mathcal{L}_m, t_{n_m} : \alpha_{f_m} \mapsto^* \ldots$$

starting from the empty list and with $n_0 < \cdots < n_m < \ldots$, let us define ν as the function such that $\nu_t(m) = f_m$ (where, if f_m contains sub-expressions $[a]$ related to $\to$ types, we ignore them). We have then that

$$\lim_{m\to\infty} |a - \nu_t(m)| = 0$$

with probability 1.

PROOF. Since the subscript of t increases at the displayed steps of the reduction sequence

$$\mathcal{L}_1, t_{n_0} : \alpha_{f_0} \mapsto^* \cdots \mapsto^* \mathcal{L}_m, t_{n_m} : \alpha_{f_m} \mapsto^* \ldots$$

we have that the atomic terms occurring in $t_{n_0}, \ldots, t_{n_m}, \ldots$ occur with increasing subscripts themselves when they occur as typed terms during our reduction. Notice that the increase in the subscript of t cannot be exclusively due to $\mathrm{I}^{\mapsto}\times$ rule applications since the type of t is always the same at the displayed reduction steps (even though it might change at the non-displayed steps, during which, potentially, logical operation rules of elimination and introduction are applied). Since the atomic terms occurring in $t_{n_0}, \ldots, t_{n_m}, \ldots$ have increasing subscripts themselves when they occur as typed terms in our reduction, Theorem 4.2 guarantees that the frequencies expressed by these atomic terms when typed tends, for the subscripts of the terms increasing to infinity, to have difference 0 with respect to the expected probability of the types of these atomic terms. By induction on the number of term evaluation rule applications, we show that if the frequency expressed by the atomic terms occurring in $t_{n_0}, \ldots, t_{n_m}, \ldots$ when they occur as typed terms tends to have difference 0 with respect to the expected probability of their types, then the frequencies $f_0, \ldots, f_m \ldots$ expressed by the typed terms $t_{n_0} : \alpha_{f_0}, \ldots, t_{n_m} : \alpha_{f_m} \ldots$ tends to have difference 0 with respect to the expected probability a of t yielding α as output.

Let us reason by induction on the number of term evaluation rules applied.

If only one evaluation rule has been applied, it must be an event$^{\mapsto t}$ rule and α is atomic. In this case, our hypothesis on the atomic terms occurring in $t_{n_0}, \ldots, t_{n_m}, \ldots$ directly gives us what we need to prove.

Suppose then that for any typed term occurrence obtained by less than r term evaluation rule applications corresponding to logical operations the statement holds, we prove that it holds also for

any typed term occurrence obtained by r term evaluation rule applications corresponding to logical operations.

We reason on the r th term evaluation rule applied to obtain the considered term.

- If the term has been obtained by $\mathrm{I}^{\mapsto}+$, then $t_m : (\beta^0 + \beta^1)_f$ and, by inductive hypothesis, the statement holds for both occurrences $t_m : \beta^0_{g_0}$ and $t_m : \beta^1_{g_1}$, from which $t_m : (\beta^0+\beta^1)_f$ has been produced by $\mathrm{I}^{\mapsto}+$, with respect to the expected probabilities b_0 and b_1 such that $\Gamma \vdash t_m : \beta^0_{\tilde{b_0}}$ and $\Gamma \vdash t_m : \beta^1_{\tilde{b_1}}$. Since $\Gamma \vdash t_m : (\beta^0 + \beta^1)_{b_0 \tilde{+} b_1}$ and $t_m : (\beta^0 + \beta^1)_f$ for $f = g_0 + g_1$, we have that the statement holds also for $t_m : (\beta^0 + \beta^1)_f$.
- If the term has been obtained by $\mathrm{I}^{\mapsto}\times$, then there must be two subterms t^0, t^1 of $t_m : (\beta^0 \times \beta^1)_f$ such that $t^0_p : \beta^0_{g_0}, t^1_p : \beta^1_{g_1}$ and $t_m : \alpha_f = \langle t_0, t_1\rangle_p : (\beta^0 \times \beta^1)_{g_0 \cdot g_1}$. By inductive hypothesis, the statement holds for both occurrences $t^0_p : \beta^0_{g_0}$ and $t^1_p : \beta^1_{g_1}$ with respect to the expected probabilities b_0 and b_1 such that $\Gamma \vdash t^0 : \beta^0_{\tilde{b_0}}$ and $\Gamma \vdash t^1 : \beta^1_{\tilde{b_1}}$. Since $\Gamma \vdash \langle t^0, t^1\rangle_p : (\beta^0 \times \beta^1)_{b_0 \tilde{\cdot} b_1}$ and $t_m : \alpha_f = \langle t^0, t^1\rangle_p : (\beta^0 \times \beta^1)_{g_0 \cdot g_1}$, we have that the statement holds also for $t_m : \alpha_f$.
- If the term has been obtained by $\mathrm{I}^{\mapsto}\ \to$, then the term is $t_m : (\beta^0 \to \beta^1)_f$ and there must be a subterm t^1 of it such that $t^1_m : \beta^1_{g_1}$ and $t_m : \alpha_f = [x_u]t^1 : (\beta^0 \to \beta^1)_{[b_0]g_1}$. By inductive hypothesis, the statement holds for the occurrence $t^1_m : \beta^1_{g_1}$ with respect to the expected probability b_1 such that $\Gamma \vdash t^1_m : \beta^1_{\tilde{b_1}}$ Since $\Gamma \vdash t^1_m : (\beta^0 \to \beta^1)_{[b_0]\tilde{b_1}}$ and $t_m : \alpha_f = [x_u]t^1 : (\beta^0 \to \beta^1)_{[b_0]g_1}$, we have that the statement holds also for $t_m : \alpha_f$. Notice that the sub-expression $[b_0]$ is ignored for the purpose of computing the difference between $[b_0]g_1$ and $[b_0]b_1$.
- If the term has been obtained by $\mathrm{E}^{\mapsto}+_L$, then the term is $t_m : \alpha_{f_1 - f_2}$. By inductive hypothesis, the statement holds for the term $t_m : (\alpha + \beta)_{f_1}$, from which t_m has been obtained by $\mathrm{E}^{\mapsto}+_L$, with respect to the expected probability b_1 such that $\Gamma \vdash t_m : (\alpha + \beta)_{\tilde{b_1}}$ and the statement holds for the term $t_m : \beta_{f_2}$ (the premise of the $\mathrm{E}^{\mapsto}+_L$ application used to obtain the term, which can be obtained with two less term evaluation rule applications than $t_m : \alpha_{f_1 - f_2}$) with respect to the expected probability b_2 such that $\Gamma \vdash t_m : \beta_{\tilde{b_2}}$. Since $\Gamma \vdash t : \alpha_{b_1 \tilde{-} b_2}$, we have that the statement holds also for $t_m : \alpha_{f_1 - f_2}$.
- The case of $\mathrm{E}^{\mapsto}+_R$ is analogous.
- If the term has been obtained by $\mathrm{E}^{\mapsto}\times_L$, then the term is $fst(t)_m : \alpha_{f_1/f_2}$. By inductive hypothesis, the statement holds for the term $t_m : (\alpha \times \beta)_{f_1}$, from which $fst(t)_m$ is obtained by $\mathrm{E}^{\mapsto}\times_L$, with respect to the expected probability b_1 such that $\Gamma \vdash \langle fst(t), u\rangle_m : (\alpha \times \beta)_{\tilde{b_1}}$ and the statement holds for the term $snd(u)_m : \beta_{f_2}$ (the premise of the $\mathrm{E}^{\mapsto}\times_L$ application used to obtain the term, which can be obtained with two less term evaluation rule applications than $fst(t)_m : \alpha_{f_1/f_2}$) with respect to the expected probability b_2 such that $\Gamma \vdash u_m : \beta_{\tilde{b_2}}$. Since $\Gamma \vdash fst(t)_m : \alpha_{b_1 \tilde{/} b_2}$, we have that the statement holds also for $fst(t)_m : \alpha_{f_1/f_2}$.
- The case of $\mathrm{E}^{\mapsto}\times_R$ is analogous.
- If the term has been obtained by $\mathrm{E}^{\mapsto}\ \to$, then the term is $t_m.[u_m : \beta] : \alpha_{f_2 \cdot f_1}$. By inductive hypothesis, the statement holds for the terms from which $t_m.[u_m : \beta]$ has been obtained by $\mathrm{E}^{\mapsto}\ \to$, that is, $u_m : \beta_{f_2}$ with respect to the expected probability b_2 such that $\Gamma \vdash x_u : \beta_{\tilde{b_2}}$ and the term $[x_u]t_{m_1} : (\beta \to \alpha)_{[b_2]f_1}$ with respect to $\Gamma \vdash [x_u]t_m : (\beta \to \alpha)_{[b_2]\tilde{b_1}}$. Since $\Gamma \vdash t_m.[x_u : \beta] : \alpha_{b_2 \tilde{\cdot} b_1}$, we have that the statement holds also for $t_m.[u_m : \beta] : \alpha_{f_2 \cdot f_1}$. □

Now we show after which reductions the distance of the frequency from the theoretical probability of the term does not increase. Indeed, for some rather obvious arithmetical reasons, some rule applications might increase the distance between the frequency and the theoretical probability. Consider, for instance, the rule

$$\frac{}{\mathcal{L}, t_n : \alpha_f, t_n : \beta_g \quad \mapsto_1 \quad \mathcal{L}, t_n : (\alpha + \beta)_{f+g}}\ \mathrm{I}^{\mapsto}+$$

Even if the frequency f is very close to the theoretical probability that t returns α and g is very close to the theoretical probability that t returns the output β, there is no guarantee that the sum $f + g$ is at least as close to the sum of the theoretical probabilities that t returns an output α and that t returns an output β. In the following theorem, we list the rules that induce reductions that do not increase the distance between frequency and theoretical probability and we show that this is indeed the case.

THEOREM 4.5 (Output Preservation).
Consider any number ε and term evaluation reduction $\mathcal{L}, t^{1\prime}, \ldots, t^{r\prime} \mapsto \mathcal{L}, u'$ resulting from an application of $\mathrm{I}^{\mapsto} \to$, $\mathrm{I}^{\mapsto} \times$ or $\mathrm{E}^{\mapsto} \to$. If, for any $i \in \{1, \ldots, r, r+1\}$, the following hold:

- t^{r+1}, if any, is the premise of the rule applied,
- $t^{i\prime} = t^i_{n^i} : \beta^i_{f^i}$,
- the judgements $\Gamma \vdash t^i_{n^i} : \alpha^i_{\underset{\sim}{a^i}}$ and $\Gamma \vdash t^i_{n^i} : \alpha^i_{f^i}$ are derivable, and
- $|f^i - a^i| < \varepsilon$;

then the following hold as well:

- $u' = u_m : \beta_g$,
- $\Gamma \vdash u_m : \beta_{\tilde{b}}$ and
- $|g - b| < \varepsilon$.

PROOF. We reason by case on the term evaluation rule applied to obtain $u' = u_m : \beta_g$ and show that the statement holds.

- Suppose that $u' = u_m : \beta_g = \langle t^1, t^2 \rangle_m : (\alpha^1 \times \alpha^2_1)_{f^1 \cdot f^2}$ has been obtained by $\mathrm{I}^{\mapsto} \times$. Then, by the hypotheses on the terms to which we have applied the rule, we have $|f^1 - a^1| < \varepsilon$ and $|f^2 - a^2| < \varepsilon$, which imply that $|(f^1 \cdot f^2) - (a^1 \cdot a^2)| \leq \max(|f^1 - a^1|, |f^2 - a^2|) < \varepsilon$. Since $f^1 \cdot f^2 = g$ and $a^1 \cdot a^2 = b$, we have that $|g - b| < \varepsilon$, as desired.
- Suppose that $u' = u_m : \beta_g = [y_s]t^1_{n^1} : (\gamma \to \alpha^1)_{[c]f^1}$, for some term s, type γ and number c, has been obtained by $\mathrm{I}^{\mapsto} \to$. Then, by the hypotheses on the term to which we have applied the rule, we have $|f^1 - a^1| < \varepsilon$. Since $f^1 = g$ and $a^1 = b$, we have that $|g - b| < \varepsilon$, as desired. Notice that the sub-expression $[c]$ is ignored for the purpose of computing the difference between $[c]f^1$ and $[c]a^1$.
- Suppose that $u' = u_m : \beta_g = s_{n^1}.[t^2_{n^2} : \alpha^2] : \gamma_{f^2 \cdot c}$ has been obtained by $\mathrm{E}^{\mapsto} \to$. Then, we have that $t^1_{n^1} : \alpha^1_{f^1} = [y_{t^2}]s_{n^1} : (\alpha^2 \to \gamma)_{[a^2]c}$ where γ is a type and c is a frequency. Hence, by the hypotheses on the terms to which we have applied the rule, we have that $|f^2 - a^2| < \varepsilon$ and that $|c - a^1| < \varepsilon$. The latter inequality derives indeed from the fact that $f^1 = [a^2]c$ and that, as usual, we ignore the sub-expressions between square brackets in the compute of the difference between frequencies and theoretical probabilities. By putting everything together, we obtain that $|(f^2 \cdot c) - (a^2 \cdot a^1)| \leq \max(|f^2 - a^2|, |c - a^1|) < \varepsilon$. Now, by the local hypotheses on the form of $u_m : \beta_g$, we have that $g = f^2 \cdot c$. On the other hand, since b is the theoretical probability

associated to the frequency g in $u_m : \beta_g$ and since $u_m : \beta_g$ is obtained by applying $t^1_{n^1} : \alpha^1_{f^1}$ to $t^2_{n^2} : \alpha^2_{f^2}$, which have, respectively, associated theoretical probability a^1 and a^2, then $b = a^2 \cdot a^1$. But then, by $g = f^2 \cdot c$ and $b = a^2 \cdot a^1$, we can conclude that $|g - b| = |(f^2 \cdot c) - (a^2 \cdot a^1)|$, which, concatenated with the inequality above yields $|g - b| < \varepsilon$, as desired. □

THEOREM 4.6 (Subject reduction).
If $\mathcal{L} \mapsto \mathcal{L}'$ by an application of a term evaluation rule r with premises belonging to Π and, for any $t_n : \alpha_f \in \mathcal{L} \cup \Pi$, it holds that $\Gamma \vdash t_n : \alpha_f$; then, for any $u_m : \beta_g \in \mathcal{L}'$, it holds that $\Gamma \vdash u_m : \beta_g$

PROOF. Now, any term evaluation rule application that does not concern any introduction or elimination of occurrences of $+$, $\times$ or $\rightarrow$ can be easily simulated by one application of the sampling rule in Figure 6. Let us then consider the case in which a term evaluation rule corresponding to a logical operation is applied. Let us denote by $R^{\mapsto}$ the rule applied. Notice first that any term evaluation rule corresponding to a logical operation is completely analogous to one rule for the introduction or elimination of $+$, $\times$ or $\rightarrow$ in Figure 5. Hence, let us denote by $R^{\vdash}$ the rule in Figure 5 corresponding to $R^{\mathrm{I}}\rightarrow$.

Clearly, it is impossible to directly simulate the term evaluation step obtained by $R^{\mapsto}$ by an application of $R^{\vdash}$ since this rule, being it a single experiment deduction rule, cannot be applied to judgements of the form $\Gamma \vdash t_n : \alpha_f$ where f is a frequency. Nevertheless, notice that any derivation δ with conclusion $\Gamma \vdash t_n : \alpha_f$ must consist of upper parts only containing applications of single experiment deduction rules in Figure 5 and then a lower part only containing sampling and update rules (Figure 6). Indeed, the rules for sampling and update are necessary for deriving a conclusion of the form $\Gamma \vdash t_n : \alpha_f$, but it is impossible to apply single experiment rules to the conclusions of applications of the rules for sampling and update. Let us denote by $\delta_1, \ldots, \delta_n$ all largest sub-derivations of δ that do not contain any sampling or update rule applications and notice that the conclusion of δ will contain the same types and terms as the conclusions of $\delta_1, \ldots, \delta_n$, since sampling and update rules never change any type or the structure of any term.

Now, in order to derive $\Gamma \vdash u_m : \beta_g$, for any $u_m : \beta_g \in \mathcal{L}'$, it is enough to apply $R^{\vdash}$ to the conclusions of some of the sub-derivations $\delta_1, \ldots, \delta_n$ of the derivation δ of $\Gamma \vdash t_n : \alpha_f$ to obtain a derivation of $\Gamma \vdash u_m : \beta_b$. Afterwards, we can apply the sampling rule to suitably chosen premises in order to derive $\Gamma \vdash u_m : \beta_g$ with the desired frequency g. □

Theorem 4.5 implies that term reduction rule applications that correspond to suitable derivation steps of TPTND, preserve trustworthiness. The following corollary makes this explicit.

COROLLARY 4.7.
For any term evaluation reduction $\mathcal{L}, t^{1\prime}, \ldots, t^{r\prime} \mapsto \mathcal{L}, u'$ resulting from an application of $\mathrm{I}^{\mapsto} \rightarrow$, $\mathrm{I}^{\mapsto} \times$ or $\mathrm{E}^{\mapsto} \rightarrow$; if, for any $i \in \{1, \ldots, r, r+1\}$ where t^{r+1} is the eventual premise of the rule applied, $t^{i\prime} = t^i_{n^i} : \beta^i_{f^i}$, if $\Gamma \vdash \mathit{Trust}(t^i_{n^i} : \alpha^i_{f^i})$ is derivable, then $\Gamma \vdash \mathit{Trust}(u_m : \beta_g)$ where $u' = u_m : \beta_g$.

PROOF. Suppose that $\mathcal{L}, t^{1\prime}, \ldots, t^{r\prime} \mapsto \mathcal{L}, u'$, where all terms involved comply with the hypotheses of our statement, and that $\Gamma \vdash \mathit{Trust}(t^i_{n^i} : \alpha^i_{f^i})$ is derivable. Then, by the definition of the IT rule in Figure 9, we know that there is a number $\epsilon(n^i)$ such that

- $\Gamma \vdash x^i : \alpha^i_{a^i}$,
- $\Gamma \vdash t^i_{n^i} : \alpha^i_{f^i}$ and
- $|a^i - f^i| \leq \epsilon(n^i)$.

But then, by Theorem 4.5, we have that, for $u' = u_m : \beta_g$, both $\Gamma \vdash u_m : \beta_{\tilde{b}}$ and $|g - b| < \epsilon(m)$ hold. Moreover, by Theorem 4.6, we have that $\Gamma \vdash u_m : \beta_g$. Hence, by the definition of the IT rule in Figure 9, we can conclude that $\Gamma \vdash \mathit{Trust}(u_m : \beta_g)$ where $u' = u_m : \beta_g$. □

Finally: if a long-enough sequence of reductions occur, then a process will eventually be identified as trustworthy; else, the process remains untrustworthy. We make this explicit by the following final result:

THEOREM 4.8 (Progress).
If $\Gamma, x : \beta_a \vdash t_n : \beta_f$ and $\varepsilon = \epsilon(n) > 0$, then either $t_{n+m} : \beta_{f'}$ such that $t_n : \beta_f \to^* t_{n+m} : \beta_{f'}$ for $m \geq 0$ exists, and $\Gamma \vdash \mathit{Trust}(t_{n+m} : \beta_{f'})$, or $t_n : \beta_f$ is untrustworthy.

PROOF. If $\Gamma \vdash t_n : \beta_{\tilde{b}}$ and $|a - b| \leq \varepsilon = \epsilon(n) > 0$ (where $\epsilon(n)$ is the parametric threshold introduced in Section 3.8), then Theorems 4.4 and 4.6, along with the definition of the IT rule in Figure 9 – since $|a - b| \leq \varepsilon$ – guarantee that there exists $m \geq 0$ such that $t_n : \beta_f \to^* t_{n+m} : \beta_{f'}$ and $\Gamma \vdash \mathit{Trust}(t_{n+m} : \beta_{f'})$.

If, otherwise, $\Gamma \vdash t_n : \beta_{\tilde{b}}$ and $|a - b| > \varepsilon = \epsilon(n) > 0$, then Theorems 4.4 and 4.6, along with the definition of the IUT rule in Figure 9 – since $|a - b| > \varepsilon$ – guarantee that there exists $m \geq 0$ such that $t_n : \beta_f \to^* t_{n+m} : \beta_{f'}$ and $\Gamma \vdash \mathit{UTrust}(t_{n+m} : \beta_{f'})$, which means that the term u is untrustworthy. □

5 Conclusion and future work

We introduced TPTND, a typed natural deduction system whose judgements express the derivability of probabilistic values for random variables, terms for processes decorated by a sample size and types for output values. The intended interpretation of such judgements is to assert the validity of the probabilistic output expected for a given process under a probability distribution.

The main use of such a calculus is the evaluation of trustworthy probabilistic computations: trustworthiness is here intended as a property induced by an acceptable distance between output frequency and its theoretical probability, parametric with respect to the sample size under observation. Such notion of trustworthiness can be used to evaluate opaque distributions against intended or desirable models, e.g. in the context of Machine Learning systems. Conditions for trustworthiness are made explicit in TPTND by a safety result, defining the required structure of a TPTND derivation by rules that determine trust in the probability of a given output.

Several extensions of this work are planned, or have already been made. First, the language of TPTND can be extended with properties of the probability distributions in order to introduce bias, and the corresponding metric on trustworthiness can be adapted accordingly as in [35]. Moreover, imprecise probabilities can be added to model uncertainty further in the model under observation. Also, a subtyping relation can be formulated to identify classification partitions. Second, the computational semantics for the typed terms of TPTND proposed in [24] can represent the basis for the development of a Coq verification protocol for probabilistic trustworthy computations, extending the existing protocol for trust presented in [34] with one of the available Coq libraries for probabilistic reasoning, e.g. https://github.com/jtassarotti/polaris. A variation of TPTND can be devised for modelling processes with finite resources for experiments: in this format, safety can rely on a normal form for terms, which is reached after a fixed number of possible experiments. Finally, a relational semantics for this system has been presented in [28].

Acknowledgments

All authors thankfully acknowledge the support of the Italian Ministry of University and Research through the projects PRIN2017 n. 20173YP4N3 and PRIN2020 n. 2020SSKZ7R (BRIO–Bias,

Risk and Opacity in AI). Fabio Aurelio D'Asaro acknowledges the funding and support of PON 'Ricerca e Innovazione' 2014–2020 (PON R&I FSE-REACT EU), Azione IV.6 'Contratti di ricerca su tematiche Green' in the context of the project titled 'Il miglioramento dell'algorithmic fairness in ambito assicurativo: Un'analisi concettuale a partire da uno studio di caso'. Francesco Genco and Giuseppe Primiero further acknowledge the support of the Italian Ministry of University and Research through the Project 'Departments of Excellence 2023–2027' awarded to the Department of Philosophy 'Piero Martinetti'.

References

[1] R. Adams and B. Jacobs. A type theory for probabilistic and bayesian reasoning. In *21st International Conference on Types for Proofs and Programs, TYPES 2015, May 18-21, 2015, Tallinn, Estonia, volume 69 of LIPIcs*, T. Uustalu. ed., pp. 1–34. Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2015. https://doi.org/10.4230/LIPIcs.TYPES.2015.1.

[2] A. Albarghouthi, L. D'Antoni, S. Drews and A. V. Nori. Fairsquare: probabilistic verification of program fairness. *Proceedings of the ACM on Programming Languages*, **1**, 1–30, 2017. https://doi.org/10.1145/3133904.

[3] A. Aldini. Design and verification of trusted collective adaptive systems. *ACM Transactions on Modeling and Computer Simulation*, **28**, 1–27, 2018. https://doi.org/10.1145/3155337.

[4] A. Aldini and M. Tagliaferri. Logics to reason formally about trust computation and manipulation. In *Emerging Technologies for Authorization and Authentication—Second International Workshop, ETAA 2019, Luxembourg City, Luxembourg, September 27, 2019, Proceedings, volume 11967 of Lecture Notes in Computer Science*, A. Saracino and P. Mori, eds, pp. 1–15. Springer, Cham, Switzerland, 2019. https://doi.org/10.1007/978-3-030-39749-4_1.

[5] R. Alur, T. A. Henzinger and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, **22**, 181–201, 1996. https://doi.org/10.1109/32.489079.

[6] P. H. Azevedo de Amorim, D. Kozen, R. Mardare, P. Panangaden and M. Roberts. Universal semantics for the stochastic λ-calculus. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29–July 2, 2021*, L. Libkin, ed, pp. 1–12. IEEE, New York, US, 2021. https://doi.org/10.1109/LICS52264.2021.9470747.

[7] G. Bacci, R. Furber, D. Kozen, R. Mardare, P. Panangaden and D. S. Scott. Boolean-valued semantics for the stochastic λ-calculus. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09–12, 2018*, A. Dawar and E. Grädel eds, pp. 669–678. ACM, New York, US, 2018. https://doi.org/10.1145/3209108.3209175.

[8] J. Boender, G. Primiero and F. Raimondi. Minimizing transitive trust threats in software management systems. In *13th Annual Conference on Privacy, Security and Trust, PST 2015, Izmir, Turkey, July 21–23, 2015*, A. A. Ghorbani, V. Torra, H. Hisil, A. Miri, A. Koltuksuz, J. Zhang, M. Sensoy, J. García-Alfaro and I. Zincir., eds, pp. 191–198. IEEE Computer Society, New York, US, 2015. https://doi.org/10.1109/PST.2015.7232973.

[9] J. Borgström, U. D. Lago, A. D. Gordon and M. Szymczak. A lambda-calculus foundation for universal probabilistic programming. In *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming, ICFP 2016, Nara, Japan, September 18–22, 2016*, J. Garrigue, G. Keller and E. Sumii, eds, vol. **2016**, pp. 33–46. ACM, New York, US. https://doi.org/10.1145/2951913.2951942.

[10] M. Boričić. Inference rules for probability logic. *Publications de l'Institut Mathématique*, **100**, 77–86, 2016.

[11] M. Boričić. Suppes-style sequent calculus for probability logic. *Journal of Logic and Computation*, **27**, 1157–1168, 2017.

[12] M. Boričić. Sequent calculus for classical logic probabilized. *Archive for Mathematical Logic*, **58**, 119–136, 2019.

[13] D. Ceolin and G. Primiero. A granular approach to source trustworthiness for negative trust assessment. In *Trust Management XIII—13th IFIP WG 11.11 International Conference, IFIPTM 2019, Copenhagen, Denmark, July 17–19, 2019, Proceedings, volume 563 of IFIP Advances in Information and Communication Technology*, W. Meng, P. Cofta, C. D. Jensen and T. Grandison, eds, pp. 108–121. Springer, Cham, Switzerland, 2019. https://doi.org/10.1007/978-3-030-33716-2_9.

[14] D. Ceolin, F. Doneda and G. Primiero. Computable trustworthiness ranking of medical experts in Italy during the sars-cov-19 pandemic. In *GoodIT '21: Conference on Information Technology for Social Good, Roma, Italy, September 9–11, 2021*, O. Gaggi, P. Manzoni and C. E. Palazzi, eds, pp. 271–276. ACM, New York, US, 2021. https://doi.org/10.1145/3462203.3475907.

[15] B. Cook. Formal reasoning about the security of amazon web services. In *Computer Aided Verification*, H. Chockler and G. Weissenbacher, eds, pp. 38–47. Springer International Publishing, Cham, 2018.

[16] F. A. D'Asaro and G. Primiero. Probabilistic typed natural deduction for trustworthy computations. In *Proceedings of the 22nd International Workshop on Trust in Agent Societies (TRUST 2021) Co-located with the 20th International Conferences on Autonomous Agents and Multiagent Systems (AAMAS 2021), London, UK, May 3–7, 2021, volume 3022 of CEUR Workshop Proceedings*, D. Wang, R. Falcone and J. Zhang, eds, CEUR-WS.org, Aachen, Germany, 2021. http://ceur-ws.org/Vol-3022/paper3.pdf.

[17] F. Dahlqvist and D. Kozen. Semantics of higher-order probabilistic programs with conditioning. *Proceedings of the ACM on Programming Languages*, **4**, 1–29, 2020. https://doi.org/10.1145/3371125.

[18] R. Demolombe. Reasoning about trust: A formal logical framework. In *Trust Management, Second International Conference, iTrust 2004, Oxford, UK, March 29–April 1, 2004, Proceedings, volume 2995 of Lecture Notes in Computer Science*, C. D. Jensen, S. Poslad and T. Dimitrakos, eds, pp. 291–303. Springer, Cham, Switzerland, 2004. https://doi.org/10.1007/978-3-540-24747-0_22.

[19] A. Di Pierro. A type theory for probabilistic–calculus. In *From Lambda Calculus to Cybersecurity Through Program Analysis: Essays Dedicated to Chris Hankin on the Occasion of His Retirement*, A. Di Pierro, P. Malacaria, R. Nagarajan, eds, pp. 86–102. Springer, Cham, 2020.

[20] N. Drawel, J. Bentahar and E. M. Shakshuki. Reasoning about trust and time in a system of agents. In *The 8th International Conference on Ambient Systems, Networks and Technologies (ANT 2017) / The 7th International Conference on Sustainable Energy Information Technology (SEIT 109), 16–19 May 2017, Madeira, Portugal, volume 109 of Procedia Computer Science*, E. M. Shakshuki, ed, pp. 632–639. Elsevier, Amsterdam, Netherlands, 2017. https://doi.org/10.1016/j.procs.2017.05.369.

[21] N. Drawel, Q. Hongyang, J. Bentahar and E. M. Shakshuki. Specification and automatic verification of trust-based multi-agent systems. *Future Generation Computer Systems*, **107**, 1047–1060, 2020. https://doi.org/10.1016/j.future.2018.01.040.

[22] T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, S. A. Seshia and S. A. Seshia.. *VerifAI: A Toolkit for the Formal Design and Analysis of Artificial Intelligence-Based Systems*. Springer International Publishing, Cham, 2019.

[23] Q. Gao, D. Hajinezhad, Y. Zhang, Y. Kantaros and M. M. Zavlanos. Reduced variance deep reinforcement learning with temporal logic specifications. In *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS '19*, X. Liu, P. Tabuada, eds, pp 237–248. Association for Computing Machinery, New York, NY, USA, 2019. https://doi.org/10.1145/3302509.3311053.

[24] F. A. Genco and G. Primiero. A typed lambda-calculus for establishing trust in probabilistic programs. *CoRR* abs/2302.00958, **2302**, 1–35, 2023. https://doi.org/10.48550/arXiv.2302.00958.

[25] S. Ghilezan, J. Ivetić, S. Kašterović, Z. Ognjanović and N. Savić. Probabilistic reasoning about simply typed lambda terms. In *International Symposium on Logical Foundations of Computer Science*, S. Artemov, A. Nerode, eds, pp. 170–189. Springer, Cham, 2018.

[26] J. Harrison. Formal verification at intel. In *18th Annual IEEE Symposium of Logic in Computer Science, 2003*, S. Abramsky, ed, pp. 45–54. IEEE, New York, US, 2003. https://doi.org/10.1109/LICS.2003.1210044.

[27] A. Herzig, E. Lorini, J. F. Hübner and L. Vercouter. A logic of trust and reputation. *Logic Journal of IGPL*, **18**, 214–244, 2010. https://doi.org/10.1093/jigpal/jzp077.

[28] E. Kubyshkina and G. Primiero. A possible worlds semantics for trustworthy non-deterministic computations. *International Journal of Approximate Reasoning*, **172**, 109212, 2024. https://doi.org/10.1016/j.ijar.2024.109212 https://www.sciencedirect.com/science/article/pii/S0888613X24000999.

[29] M. Z. Kwiatkowska, G. Norman and D. Parker. Prism: probabilistic symbolic model checker. In *Proceedings of the 12th International Conference on Computer Performance Evaluation, Modelling Techniques and Tools, TOOLS '02*, T. Field, P. G. Harrison, J. Bradley, U. Harder, eds, pp. 200–204. Springer, Berlin, Heidelberg, 2002.

[30] C.-J. Liau. Belief, information acquisition, and trust in multi-agent systems—a modal logic formulation. *Artificial Intelligence*, **149**, 31–60, 2003. https://doi.org/10.1016/S0004-3702(03)00063-8.

[31] F. Liu and E. Lorini. Reasoning about belief, evidence and trust in a multi-agent setting. In *PRIMA 2017: Principles and Practice of Multi-Agent Systems—20th International Conference, Nice, France, October 30–November 3, 2017, Proceedings, volume 10621 of Lecture Notes in Computer Science*, B. An, A. L. C. Bazzan, J. Leite, S. Villata and L. W. N. van der Torre, eds, pp. 71–89. Springer, Cham, Switzerland, 2017. https://doi.org/10.1007/978-3-319-69131-2_5.

[32] G. Primiero. A logic of negative trust. *Journal of Applied Non-Classical Logics*, **30**, 193–222, 2020. https://doi.org/10.1080/11663081.2020.1789404.

[33] G. Primiero and J. Boender. Managing software uninstall with negative trust. In *Trust Management XI—11th IFIP WG 11.11 International Conference, IFIPTM 2017, Gothenburg, Sweden, June 12–16, 2017, Proceedings, volume 505 of IFIP Advances in Information and Communication Technology*, J.-P. Steghöfer and B. Esfandiari, eds, pp. 79–93. Springer, Cham, Switzerland, 2017.

[34] G. Primiero and J. Boender. Negative trust for conflict resolution in software management. *Web Intelligence*, **16**, 251–271, 2018. https://doi.org/10.3233/WEB-180393.

[35] G. Primiero and F. A. D'Asaro. Proof-checking bias in labeling methods. In *Proceedings of 1st Workshop on Bias, Ethical AI, Explainability and the Role of Logic and Logic Programming (BEWARE 2022) co-located with the 21th International Conference of the Italian Association for Artificial Intelligence (AI∗IA 2022), Udine, Italy, December 2, 2022, volume 3319 of CEUR Workshop Proceedings*, G. Boella, F. A. D'Asaro, A. Dyoub and G. Primiero, eds, pp. 9–19. CEUR-WS.org, Aachen, Germany, 2022. https://ceur-ws.org/Vol-3319/paper1.pdf.

[36] G. Primiero, F. Raimondi, T. Chen and R. Nagarajan. A proof-theoretic trust and reputation model for VANET. In *2017 IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops 2017, Paris, France, April 26–28, 2017*, IEEE Computer Society, ed, pp. 146–152. IEEE, New York, US, 2017. https://doi.org/10.1109/EuroSPW.2017.64.

[37] S. A. Seshia, A. Desai, T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, S. Shivakumar, M. Vazquez-Chanlatte and X. Yue. Formal specification for deep neural networks. In *Automated Technology for Verification and Analysis*, S. K. Lahiri and C. Wang, eds, pp. 20–34, Cham, Springer International Publishing, 2018.

[38] M. P. Singh. Trust as dependence: a logical approach. In *10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), Taipei, Taiwan, May 2–6, 2011, Volumes 1–3*, L. Sonenberg, P. Stone, K. Tumer and P. Yolum, eds, pp. 863–870. IFAAMAS, Liverpool, UK, 2011. http://portal.acm.org/citation.cfm?id=2031741&CFID=54178199&CFTOKEN=61392764.

[39] A. Termine, A. Antonucci, G. Primiero and A. Facchini. Logic and model checking by imprecise probabilistic interpreted systems. In *Multi-Agent Systems—18th European Conference, EUMAS 2021, Virtual Event, June 28–29, 2021, Revised Selected Papers, volume 12802 of Lecture Notes in Computer Science*, A. Rosenfeld and N. Talmon, eds, pp. 211–227. Springer, Cham, Switzerland, 2021. https://doi.org/10.1007/978-3-030-82254-5_13.

[40] A. Termine, G. Primiero and F. A. D'Asaro. Modelling accuracy and trustworthiness of explaining agents. In *Logic, Rationality, and Interaction—8th International Workshop, LORI 2021, Xi'ian, China, October 16–18, 2021, Proceedings, volume 13039 of Lecture Notes in Computer Science*, S Ghosh and T Icard, eds, pp 232–245. Springer, Cham, Switzerland, 2021. https://doi.org/10.1007/978-3-030-88708-7_19.

[41] C. Urban and A. Minè. A review of formal methods applied to machine learning, CoRR abs/2104.02466, **2104** 1–39, 2021. https://arxiv.org/pdf/2104.02466.

[42] J. H. Warrell. A probabilistic dependent type system based on non-deterministic beta reduction. *CoRR* abs/1602.06420, vol. 1602, pp. 1–15, 2016. https://arxiv.org/pdf/1602.06420

[43] J. M. Wing. Trustworthy ai. *Communications of the ACM*, **64**, 64–71, 2021. https://doi.org/10.1145/3448248.

Received 14 May 2024