Induction for Positive Almost Sure Termination
Isabelle Gnaedig

To cite this version:

HAL Id: inria-00182435
https://inria.hal.science/inria-00182435
Submitted on 10 Oct 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Induction for Positive Almost Sure Termination

Isabelle GNAEDIG
LORIA-INRIA,
615, rue du Jardin Botanique, BP 101,
F-54602 Villers-les-Nancy Cedex,
Phone: + 33 3 54 95 84 21
Isabelle.Gnaedig@loria.fr

Abstract
In this paper, we propose an inductive approach to prove positive almost sure termination of probabilistic rewriting under the innermost strategy. We extend to the probabilistic case a technique we proposed for termination of usual rewriting under strategies. The induction principle consists in assuming that terms smaller than the starting terms for an induction ordering are positively almost surely terminating. The proof is developed in generating proof trees, modeling rewriting trees, in alternatively applying abstraction steps, expressing the application of the induction hypothesis, and narrowing steps, simulating the possible rewriting steps after abstraction. This technique can be fully automatized, in particular for rewrite systems on constants, very useful to modelize probabilistic protocols.

Categories and Subject Descriptors F.3.1 [LOGICS AND MEANINGS OF PROGRAMS]: Specifying and Verifying Reasoning about Programs—Logics of programs, Mechanical verification, Specification techniques; F.4.2 [MATHEMATICAL LOGIC AND FORMAL LANGUAGES]: Grammars and Other Rewriting Systems; F.4.3 [MATHEMATICAL LOGIC AND FORMAL LANGUAGES]: Formal Languages—Algebraic language theory; G.3 [PROBABILITY AND STATISTICS]: I.1.3 [SYMBOLIC AND ALGEBRAIC MANIPULATION]: Languages and Systems—Evaluation strategies; I.2.3 [ARTIFICIAL INTELLIGENCE]: Deduction and Theorem Proving—Deduction, Inference engines, Mathematical induction; D.3.1 [PROGRAMMING LANGUAGES]: Formal Definitions and Theory; D.2.4 [SOFTWARE ENGINEERING]: Software/Program Verification—Correctness proofs, Formal methods, Validation

General Terms Algorithms, Languages, Verification

Keywords Abstraction, Constraint, Narrowing, Probability, Termination

1. Introducing the Problem
Probabilistic rewriting has recently been introduced to modelize systems, where probabilistic and non-deterministic phenomena are combined [7]. A lot of models of systems, formalisms or techniques have already been enriched with probabilities, but most of them are restricted to finite state systems. Let us cite automata based models [11, 42], Petri Nets [2, 38], process algebra [24], model checking techniques [29]. Note also the existence of the PRISM [30], and the APMC [25] tools.

Rewriting allows for expressing complex relations on infinite sets of states in a finite way, provided they are countable. Rewriting also provides a nice programming paradigm: the rule-based formalism allows to easily write programs in a declarative way and the underlying algebraic semantics enables automatable or semi-automatable correctness proofs on the programs. Rule-based programming languages and environments are now currently used for any kind of application. Let us cite ASF+SDF [8], Maude [9], CafeOBJ [17], ELAN [3], Stratego [43], or Tom [32]. A probabilistic dimension has recently been introduced in ELAN [6], Tom [18] and Maude [28] to increase the power of the languages to applications like probabilistic protocols.

In the context of probabilistic rewriting, the problem of termination naturally arises and in [4], the notions of simple almost sure termination and positive almost sure (PAS in short) termination have been proposed, as well as a method based on interpretations on the reals to ensure the second property. The first termination notion expresses that the probability for a given rewriting derivation to terminate is 1; the second, stronger and more useful from a practical point of view, expresses that the mean length of the derivations from a term is finite.

Then, in [5], rewriting strategies have been considered, and sufficient criterions, still based on interpretations on the reals, have been given for PAS termination under strategies. Here, we try to go one step further. In the previously cited paper, the considered strategies defined themselves with probabilities, expressing the ratio of the selection of a rule w.r.t to another. We tackle here the PAS termination problem for position strategies, defined by the position of the redexes in the terms to be rewritten, using an inductive approach we proposed for proving termination of non-probabilistic rewriting under the innermost [14], the outermost [15] and local strategies [13]. In this paper, we adapt our inductive technique to the probabilistic case, investigate how it then works, and give a class of systems for which it is of interest.

[Copyright notice will appear here once ‘preprint’ option is removed.]
We focus here on the innermost strategy, consisting in always rewriting at the lowest possible positions. This strategy is widely used in programming. It is often used as a built-in mechanism in the evaluation of rule-based or functional languages. In addition, for non-overlapping or locally confluent overlay systems [21] or systems satisfying critical peak conditions [22], innermost termination is equivalent to standard termination (i.e. termination for standard rewriting, which consists in rewriting without any strategy). Note that as proved in [26], leftmost innermost termination and innermost termination of rewriting are equivalent.

A formalism has recently been proposed to extend the Constraint Handling Rule process with probabilistic capabilities applied to the rewrite rules themselves [16, 35, 36, 34, 37]. This is, to our knowledge, except the previously cited works on ELAN, Tom and Maude, the only attempt to formalize probabilistic transitions using rule based languages. Notice that these papers do not focus on techniques for proving termination of such systems.

There are other works about termination with probabilities, but in the context of concurrent programs [30, 39]. They deal with almost sure termination, whereas we deal with positive almost sure termination.

The basic idea of our approach is the following. We introduce the notion of innermost PAS (IPAS in short) termination for a term, and suppose, for every term $t$ of a ground term algebra, that the terms smaller than $t$ for an induction ordering are PAS terminating. We then try to deduce that $t$ is also IPAS terminating. The principle of our inductive method lies on a double mechanism allowing to generate proof trees, which represent, by a lifting mechanism, the rewriting trees of the ground terms: abstraction and narrowing.

The paper is structured as follows. In Section 2, the background is presented. Section 3 is devoted to definitions of probabilistic rewriting. In Section 4, the material for our inductive technique in the probabilistic case is defined. Section 5 gives the algorithm generating proof trees and the IPAS termination result for finite proof trees. Finally, Section 6 presents a generalization to a given class of infinite proof trees.

2. The Background

We assume that the reader is familiar with the basic definitions and notations of term rewriting given for instance in [1, 12, 41]. $T(F, \mathcal{X})$ is the set of terms built from a given finite set $F$ of function symbols $f$ having arity $n \in \mathbb{N}$, and a set $\mathcal{X}$ of variables denoted $x, y, \ldots$. $T(\mathcal{F}, \mathcal{X})$ is the set of ground terms (without variables). The symbols of arity 0 are also terms called constants. Positions in a term are represented as sequences of integers. The empty sequence is the top position. Let $p$ and $p'$ be two positions. The position $p$ is said to be (a strict) prefix of $p'$ (and $p'$ suffix of $p$) if $p' = p \lambda$, where $\lambda$ is a (non-empty) sequence of integers. For a position $p$ of a term $t$, we note $t[p]$, the subterm of $t$ at position $p$, and $t[p]$, the term obtained in replacing by $s$ the subterm at position $p$ in $t$.

A substitution is an assignment from $\mathcal{X}$ to $T(F, \mathcal{X})$, written $\sigma = (x = t, \ldots, y = u)$. It uniquely extends to an endomorphism of $T(F, \mathcal{X})$. The result of applying $\sigma$ to a term $t \in T(F, \mathcal{X})$ is written $\sigma(t)$ or $\sigma t$. The domain of $\sigma$, denoted $\text{Dom}(\sigma)$, is the finite subset of $\mathcal{X}$ such that $\sigma x \neq x$. An instantiation or ground substitution is an assignment from $\mathcal{X}$ to $T(F)$. $\text{Id}$ denotes the identity substitution. The composition of substitutions $\sigma_1$ followed by $\sigma_2$ is denoted $\sigma_2 \sigma_1$.

A set $\mathcal{R}$ of rewrite rules or rewrite system (RS in short) on $T(\mathcal{F}, \mathcal{X})$ is a set of pairs of terms of $T(\mathcal{F}, \mathcal{X})$, denoted $l \rightarrow r$, such that $\text{Var}(r) \subseteq \text{Var}(l)$. Given a rewrite system $\mathcal{R}$, a function symbol in $\mathcal{F}$ is called a constructor if it does not occur in $\mathcal{R}$ at the top position of a left-hand side of rule, and is called a defined function symbol otherwise. The set of constructors of $\mathcal{F}$ for $\mathcal{R}$ is denoted $\mathcal{C}_R$, and the set of defined function symbols $\mathcal{D}_R$ ($\mathcal{R}$ is omitted when there is no ambiguity). In this paper, we only consider finite sets of function symbols and of rewrite rules.

The rewriting relation induced by $\mathcal{R}$ is denoted by $\rightarrow^\mathcal{R}$ ($\rightarrow$ if there is no ambiguity on $\mathcal{R}$), and defined by $s \rightarrow t$ if there is a substitution $\sigma$ and a position $p$ in $s$ such that $s[p] = \sigma t$ for some rule $l \rightarrow r$ of $\mathcal{R}$, and $t = s[\sigma]$. This is written $s \rightarrow_{\mathcal{R}, l, \sigma, r, t}$ where $p$, $l \rightarrow r$, $\sigma$ or $\mathcal{R}$ may be omitted; $s_0$ is called a redex. The innermost rewriting relation consists in always rewriting at the lowest possible positions.

Let $\mathcal{R}$ be a rewrite system on $T(\mathcal{F}, \mathcal{X})$. A term $t$ is narrowed into $t'$, at the non-variable position $p$, using the rewrite rule $l \rightarrow r$ of $\mathcal{R}$ and the substitution $\sigma$, when $\sigma$ is a most general unifier of $l_0(l)$ and $l$, and $t' \equiv \sigma(l_0[l_0[\sigma])]$. This is denoted $t \rightarrow_{\mathcal{R}, l, \sigma, r, t'}$ where $p$, $l \rightarrow r$, $\sigma$ or $\mathcal{R}$ may be omitted. It is always assumed that there is no variable in common between the rule and the term, i.e. that $\text{Var}(l_0) \cap \text{Var}(t) = \emptyset$.

An ordering $\succ$ on $T(\mathcal{F}, \mathcal{X})$ is said to be noetherian if there is no infinitely decreasing chain for this ordering. It is monotone iff for any pair of terms $t, t'$ of $T(\mathcal{F}, \mathcal{X})$, for any context $f(\ldots, t, \ldots, t')$. It has the subterm property iff for any $t \in T(\mathcal{F}, \mathcal{X})$, $f(\ldots, t, \ldots, t')$.

For $\mathcal{F}$ and $\mathcal{X}$ finite, if $\succ$ is monotone and has the subterm property, then it is noetherian [27]. If, in addition, $\succ$ is stable under substitution (for any substitution $\sigma$, any pair of terms $t, t' \in T(\mathcal{F}, \mathcal{X})$, $t \succ t'$ implies $\sigma t \succ \sigma t'$), then it is called a simplification ordering. A RS $\mathcal{R}$ (innermost) terminates if and only if every (innermost) derivation of the rewriting relation induced by $\mathcal{R}$ is finite. For any term $t$ of $T(\mathcal{F})$, $t$ (innermost) terminates if and only if every (innermost) rewriting derivation starting from $t$ is finite.

3. Probabilistic Rewriting

A $\sigma$-algebra on a set $\Omega$ is a set of subsets of $\Omega$ which contains the empty-set, and is stable by countable union and complementation. In particular, the set of subsets of a natural $\sigma$-algebra for any countable set. A measurable space $(\Omega, \mathcal{A})$ is a set with a $\sigma$-algebra on it. A probability is a function $P$ from a $\sigma$-algebra to $[0, 1]$, which is countably additive, and such that $P(\Omega) = 1$. A triplet $(\Omega, \sigma, P)$ is called a probability space. For more details, see [23].

A stochastic sequence on a set $A$ is a family $(X_i)_{i \in \mathbb{N}}$, of random variables defined on some fixed probability space $(\Omega, \sigma, P)$ with values on $A$.

Definition 1 (PARS). [4] Given some countable set $S$, we note $\text{Dist}(S)$ for the set of probability distributions on $S$; $\mu \in \text{Dist}(S)$ is a function $S \rightarrow [0, 1]$ that satisfies $\sum_{i \in S} \mu(i) = 1$. A probabilistic abstract reduction system (PARS) is a pair $A = (A, \rightarrow)$ consisting of a countable set $A$ and a relation $\rightarrow \subseteq A \times \text{Dist}(A)$. A state $a \in A$ with no $\mu$ such that $a \rightarrow \mu$ is said terminal.
A PARS is said deterministic if, for all a, there is at most one µ with a → µ. We denote Dist(A) for the set of distributions µ with a → µ for some a.

A history is a finite sequence a₀a₁...aₙ of elements of the state space A. It is non-terminal if a₀ is as well. A history expresses the evolution of a PARS.

Definition 2 (Deterministic Policy). [4] A (deterministic) policy φ, that can also be called a (deterministic) strategy, is a function that maps non-terminal histories to distributions in such a way that φ(a₀a₁...aₙ) = µ is always one (of the many possible) distribution µ with a₀ → µ. A history is said to be realizable, if for all i < n, if µᵢ denotes φ(a₀a₁...aᵢ), one has µᵢ(aᵢ₊₁) > 0.

The above definition assumes that strategies must be deterministic.

A derivation of A is then a stochastic sequence where the non-deterministic choices are given by some policy φ, and the probabilistic choices are governed by the corresponding distributions.

Definition 3 (Derivations). [4] A derivation π of A over policy φ is a stochastic sequence π = (πᵢ)ᵢ∈N on the set A ∪ {⊥} (where ⊥ is a new element: ⊥ ∉ A) such that for all n,

- $P(\pi_{n+1} = a | \pi_n = ⊥) = 1$,  
- $P(\pi_{n+1} = a | \pi_n = s) = 1$ if s ∈ A is terminal,  
- $P(\pi_{n+1} = ⊥ | \pi_n = s) = 0$ if s ∈ A is non-terminal,  
- and for all t ∈ A: $P(\pi_{n+1} = t | \pi_n = a₀, ..., a_{n-1}) = a₀$ = $\mu(t)$ whenever a₀a₁...aₙ is a realizable non-terminal history and $\mu = \phi(a₀a₁...aₙ)$.

If a derivation is such that $\pi_n = ⊥$ for some n, then $\pi_{n'} = ⊥$ almost surely for all $n' ≥ n$. Such a derivation is said to be terminating. If k is the biggest integer for which $\pi_k$ ≠ ⊥, then $\pi_k$ is called a normal form (of $\pi_0$). A non-terminating derivation is such that $\pi_n ∈ A (\pi_n ≠ ⊥)$ for all n.

The following definition is generalized from [5] to a class of policies Φ.

Definition 4 (PAS Termination). A PARS A = (A, →) will be said positively almost surely (PAS) terminating (under a class of strategies Φ) if for all policies $\phi ∈ Φ$, for all states $a ∈ A$, the mean number of reduction steps (the mean length of the derivations) to reach a normal form starting from a under policy $\phi$, denoted by $T(a, φ)$, is finite.

Definition 5 (Probabilistic Rewrite System). [4] Given a set of terms $T(F, X)$, a probabilistic rewrite rule is an element $l → M$ of $T(F, X) × Dist(T(F, X))$, such that for every $r ∈ T(F, X)$, if $M(r) > 0$, then $Var(r) ⊆ Var(l)$. A probabilistic rewrite system is a finite set $R$ of probabilistic rewrite rules.

A probabilistic abstract reduction system ($T(F, X), →_R$) over the set of terms $T(F, X)$ is associated to a probabilistic rewrite system where →ₐ is defined as follows.

Definition 6 (Reduction Relation). [4] The following PARS ($T(F, X), →_R$) over terms is associated to a probabilistic rewrite system R as follows: $l → M$ if there is a rule $l → M = (r₁ : p₁, ..., rₖ : pₖ) ∈ R$, some position p in t, some substitution $σ$, such that $t|_p = σ(l)$, and, for all $t'$, $\mu(t') = \sum_{r₁, ..., rₖ | E(t') | \{r₁|_p = σ(l)\}} M(r₁)$. For example, with the probabilistic rewrite rule $\{ f(x, y) → g(a) : 1/2 | y : 1/2 \}$ whose right hand side denotes the distribution with value 1/2 on $g(a)$ and value 1/2 on y, $f(b, c)$ rewrites to $g(a)$ with probability 1/2, to c with probability 1/2 and $f(b, g(a))$ rewrites to $g(a)$ with probability 1.

Innermost probabilistic rewriting consists in always applying the above definition at the lowest possible positions in the terms to be rewritten.

A probabilistic rewrite system R is innermost positively almost surely (IPAS) terminating if the associated PARS is PAS terminating under the class $\Phi_{\text{ran}}$ of policies $\phi$ corresponding to innermost rewriting derivations.

A term t on which no rule of R applies is said to be in normal form for R. If such a term t is on a(n) (innermost) rewriting derivation of a term a, then t is called (innermost) normal form of a, and is noted $aₙ$. To every rewriting derivation $a₀, a₁, ..., aₙ$ corresponds a derivation $π₀, π₁, ..., πₙ, πₙ₊₁, ...$ where $π_i = ⊥$ for $i > n$.

4. Inductively Proving Positive Almost Sure Termination

For proving that a probabilistic rewrite system on $T(F)$ is IPAS terminating, we introduce a local notion of IPAS termination on terms, and prove this property for every term of $T(F)$.

Definition 7. Let R be a probabilistic rewrite system on $T(F, X)$. A term t of $T(F)$ is said to be IPAS terminating if for every $φ ∈ \Phi_{\text{ran}}$, the mean number $T(φ, t)$ of rewriting steps to reach a normal form from t with R under the strategy $φ$ is finite.

For proving that a term t of $T(F)$ is IPAS terminating, we proceed by induction on $T(F)$ with a noetherian ordering $\succ$, assuming the property for every $t'$ such that $t \succ t'$. To warrant non-emptiness of $T(F)$, and a basis for the induction, we assume that $F$ contains at least one constructor constant. The main intuition is to observe probabilistic rewriting derivations starting from a ground term $t ∈ T(F)$ which is any instance of a pattern $g(x₁, ..., xₘ) ∈ T(F, X)$, for some defined function symbol $g ∈ D$, and variables $x₁, ..., xₘ$. Proving the property of IPAS termination on ground terms amounts to proving that every ground instance of the patterns $g(x₁, ..., xₘ)$ is IPAS terminating.

Rewriting derivations are simulated, using a lifting mechanism, by a proof tree developed from $g(x₁, ..., xₘ)$ on $T(F, X)$, for every $g ∈ D$, by alternatively using two main concepts, namely narrowing and abstraction. More precisely, narrowing schematizes the rewriting possibilities of terms. Abstraction simulates the reduction of subterms in the derivations until these subterms become normal forms. It expresses the application of the induction hypothesis on these subterms: if they are IPAS terminating, with a mean number of reduction steps, they rewrite into a normal form.

The schematization of ground rewriting derivations is achieved through constraints. The nodes of the developed proof trees are composed of a current term of $T(F, X)$, and a set of ground substitutions represented by a constraint progressively built along the successive abstraction and narrowing steps. Each node in an abstract tree schematizes a set of ground terms: all ground instances of the current term, that are solutions of the constraint.

The constraint is in fact composed of two kinds of formulas: ordering constraints, set to warrant the validity of the inductive steps, and abstraction constraints combined.
to narrowing substitutions, which effectively define the relevant sets of ground terms.

For a term \( t \) of \( T(F, X) \) occurring in a proof tree issued from a reference term \( t_{ref} = g(x_1, \ldots, x_m) \),

- first, the ground instances of some subterms \( t_j \) of \( t \) (characterized by the constraint associated to \( t \)) are supposed to be IPAS terminating, by the induction hypothesis, if \( \theta_{ref} \succ \theta t_j \) for the induction ordering \( \succ \) and for every \( \theta \) solution of the constraint associated to \( t \). They are replaced in \( t \) by abstraction variables \( X_j \) representing respectively any of their normal forms, implicitly corresponding to one of the normal forms they have when rewriting under any policy \( \phi \in \Phi_{\text{inn}} \). Reasoning by induction allows us to only suppose the existence of the normal forms without explicitly computing them. If the ground instances of the resulting term are IPAS terminating (either if the induction hypothesis can be applied to them, or if they can be proved IPAS terminating by other means, we will present later), then the ground instances of the initial term are IPAS terminating. Otherwise,

- the resulting term \( u = t[X_j{\mid i_1, \ldots, i_p}] \) (where \( i_1, \ldots, i_p \) are the abstraction positions in \( t \)) is narrowed in all possible ways into distributions \( \mu \), according to the possible instances of the \( X_j \). This corresponds to rewritings ground instances of \( u \) characterized by the constraint associated to \( u \) according to all non-deterministic choices and all probabilistic choices. Thus, all policies \( \phi \in \Phi_{\text{inn}} \) are explicitly expressed by the narrowing mechanism.

Then IPAS termination of the ground instances of \( t \) is reduced to IPAS termination of the ground instances of the terms \( v \) of the distributions \( \mu \). Now, if \( \theta_{ref} \succ \theta \) for every ground substitution \( \theta \) that is a solution of the constraint associated to \( v \), by the induction hypothesis, \( \theta \) is supposed to be IPAS terminating. Otherwise, the process is iterated on \( v \), until we get a term \( t' \) such that either \( \theta_{ref} \succ \theta t' \), or \( \theta t' \) can be proved IPAS terminating.

This technique is inspired from the one we proposed for proving innermost termination of non-probabilistic rewrite systems.

We now introduce some concepts to formalize and automate it.

### 4.1 Ordering Constraints

The induction ordering is constrained along the proof by inequalities between terms that must be comparable, each time the induction hypothesis is used in the abstraction mechanism.

This ordering is not defined a priori, but just has to verify inequalities of the form \( t \succ u_1, \ldots, u_m \), accumulated along the proof, and which are called ordering constraints. Thus, for establishing the inductive termination proof, it is sufficient to decide whether ordering constraints are satisfiable.

**Definition 8 (Ordering Constraint).** An ordering constraint is a pair of terms of \( T(F, X) \) noted \( (t \succ t') \). It is said to be satisfiable if there is an ordering \( \succ \), such that for every instantation \( \theta \) whose domain contains \( \text{Var}(t) \cup \text{Var}(t') \), we have \( \theta t \succ \theta t' \). We say that \( \succ \) satisfies \( (t \succ t') \).

A conjunction \( C \) of ordering constraints is satisfiable if there is an ordering satisfying all conjuncts. The empty conjunction, always satisfied, is denoted by \( \top \).

Satisfiability of a constraint conjunction \( C \) of this form is undecidable. But a sufficient condition for an ordering \( \succ \) on \( T(F, X) \) to satisfy \( C \) is that \( t \succ t' \) for every constraint \( t \succ t' \) of \( C \), and \( \succ \) is stable under substitution.

Simplification orderings fulfill such a condition. So, in practice, it is sufficient to find a simplification ordering \( \succ \) such that \( t \succ t' \) for every constraint \( t \succ t' \) of \( C \).

The ordering \( \succ \), defined on \( T(F, X) \), can then be seen as an extension of the induction ordering \( \succ \) on \( T(F) \). For convenience sake, \( \succ \) will also be written \( \succ \).

Solving ordering constraints in finding simplification orderings is a well-known problem. The simplest way and an automatable way to proceed is to test simple existing orderings like the subterm ordering, the Recursive Path Ordering, or the Lexicographic Path Ordering. This is often sufficient for the constraints considered here: thanks to the power of induction, they are often simpler than for termination methods directly using ordering for orienting rewrite rules.

If these simple orderings are not powerful enough, automatic solvers like Cam\textsuperscript{1} can provide adequate polynomial orderings.

### 4.2 Abstraction

To abstract a term \( t \) at positions \( i_1, \ldots, i_p \), where the \( t_j \) are supposed to have a normal form \( t_j\upharpoonright \), we replace the \( t_j \) by abstraction variables \( X_j \) representing respectively any of their possible normal forms for any policy \( \phi \in \Phi_{\text{inn}} \). Let us define these special variables more formally.

**Definition 9.** Let \( N \) be a set of variables disjoint from \( X \). Symbols of \( N \) are called abstraction variables. Substitutions and instantiations are extended to \( T(F, X \cup N) \) in the following way: for any substitution \( \sigma \) (resp. instantiation \( \theta \)) such that \( \text{Dom}(\sigma) \) (resp. \( \text{Dom}(\theta) \)) contains a variable \( X \in N \), \( x \in N \) (resp. \( \theta X \) ) is in innermost normal form.

**Definition 10 (Term Abstraction).** The term \( t[t_j{\mid c(i_1, \ldots, i_p)}] \) is said to be abstracted into the term \( u \) (calculated abstraction of \( t \) ) at positions \( \{i_1, \ldots, i_p\} \), where \( X_j, j \in \{i_1, \ldots, i_p\} \) are fresh distinct abstraction variables.

Termination on \( T(F) \) is in fact proved by reasoning on terms with abstraction variables, i.e., on terms of \( T(F, X \cup N) \). Ordering constraints are extended to pairs of terms of \( T(F, X \cup N) \). When subterms \( t_j \) are abstracted by \( X_j \), we state constraints on abstraction variables, called abstraction constraints to express that their instances can only be normal forms of the corresponding instances of \( t_j \). Initially, they are of the form \( t_j = X \) where \( t \in T(F, X \cup N) \), and \( X \in N \), but we will see later how they are combined with the substitutions used for the narrowing process.

### 4.3 Narrowing

After abstracting the current term \( t \) into \( t[X_j{\mid c(i_1, \ldots, i_p)}] \), we test whether the possible ground instances of \( t[X_j{\mid c(i_1, \ldots, i_p)}] \) are reducible, according to the possible values of the instances of the \( X_j \). This is achieved by innermost narrowing \( t[X_j{\mid c(i_1, \ldots, i_p)}] \).

To schematize innermost rewriting on ground terms, we need to refine the usual notion of narrowing. In fact, with the usual innermost narrowing relation, if a position \( p \) in a term \( t \) is a narrowing position, no suffix position of \( p \) can be a narrowing position as well. However, if we consider ground

\footnote{Available at http://cime.lri.fr/}
instances of $t$, we can have rewriting positions $p$ for some instances, and $p'$ for other instances, such that $p'$ is a suffix of $p$. So, when using the narrowing relation to schematize innermost rewriting of ground instances of $t$, the narrowing positions $p$ to consider depend on a set of ground instances of $t$, which is defined by excluding the ground instances of $t$ that would be narrowable at some suffix position of $p$. For instance, with the RS $R = \{g(a) \to a, f(g(x)) \to b\}$, the innermost narrowing positions of the term $f(g(X))$ are 1 with the narrowing substitution $\sigma = (X = a)$, and $\epsilon$ with any $\sigma$ such that $\sigma X \neq a$.

Let $\sigma$ be a substitution on $T(F, X \cup N)$. In the following, we identify $\sigma$ with the equality formula $\bigwedge_{i \in [1..m]} \sigma_t(x_i = t_i)$, with $x_i \in X \cup N$, $t_i \in T(F, X \cup N)$. Similarly, we call negation $\overline{\sigma}$ of the substitution $\sigma$ the formula $\bigvee_{i \in [1..n]} \overline{x_i}$.

**Definition 11.** A substitution $\sigma$ is said to be a constraint $\bigwedge_{i \in [1..n]} \overline{x_i}$, iff for every ground instantiation $\theta$, $\bigwedge_{i \in [1..n]} \overline{x_i} \neq \theta x_i$. A constrained substitution $\sigma$ is a formula $\sigma_0 \cap \bigwedge_{i \in [1..n]} \overline{x_i}$, where $\sigma_0$ is a substitution, and $\bigwedge_{i \in [1..n]} \overline{x_i}$ the constraint to be satisfied by $\sigma_0$.

**Definition 12** (Innermost Probabilistic Narrowing). A term $t \in T(F, X \cup N)$ innermost narrows into a distribution $\mu$ at the non-variable position $p$, using the rule $l \rightarrow M = (r_1 : p_1, \ldots, r_k : p_k) \in R$ with the constrained substitution $\sigma = \sigma_0 \cap \bigwedge_{i \in [1..m]} \overline{x_i}$, which is written $t \rightarrow^*_{l \rightarrow M, \sigma} \mu = (v_1 : p_1^1, \ldots, v_k : p_k)$ iff

- $\sigma_0(l) = \sigma_0(p)$
- for all $v_j, j \in [1..q]$, $v_j = \sigma_0(r_{i_j})$ for some $i \in [1..k]$
- $\mu(v_j) = p_j^i \sum_{v_i \in [1..q]} \overline{x_j} = \sigma_0(r_{i_j}) M(r_i)$

where $\sigma_0$ is the most general unifier of $t$ and $l$ at position $p$, and $\sigma_0, h \in [1..m]$ are all the most general unifiers of $\sigma$ and a left-hand side of rule of $R$, at suffix positions of $p$.

**Definition 13** ((S-narrowing)). A term $t \in T(F, X \cup N)$ $S$-narrows into a term $t' \in T(F, X \cup N)$ at the non-variable position $p$ of $t$, using the rule $l \rightarrow r \in R$ with the constrained substitution $\sigma = \sigma_0 \cap \bigwedge_{i \in [1..m]} \overline{x_i}$, which is written $t \rightarrow^*_{S, \sigma} t'$ iff

$\sigma_0(l) = \sigma_0(r) \land t' = \sigma_0(r_{i_j})$.

where $\sigma_0$ is the most general unifier of $l$, and $l$ and $\sigma, j \in [1..k]$ are all the most general unifiers of $\sigma$ for $l$, and a left-hand side $l$ of a rule of $R$, for all position $p$ which are S-better positions than $p$.

Notice that we are interested in the narrowing substitution applied to the current term $t$, but not in its definition on the variables of the left-hand side of the rule. So, the narrowing substitutions we consider are restricted to the variables of the narrowed term $t$.

### 4. Cumulating Constraints

Abstraction constraints have to be combined with the narrowing substitutions to characterize the ground terms schematized by the current term $t$ in the proof tree. Indeed, a narrowing branch on the current term $t$ with narrowing substitution $\sigma$ represents a rewriting branch for any ground instance of $\sigma$.

In addition, $\sigma$ has to satisfy the constraints on variables of $u$, already set in $A$. So, $\sigma$, considered as the narrowing constraint attached to the narrowing branch, is added to $A$. This leads to the introduction of abstraction constraint formulas.

**Definition 14** (Abstraction Constraint Formula). An abstraction constraint formula (ACF in short) is a formula $\bigwedge_{i \in [1..n]} \overline{x_i} \land \bigwedge_{i \in [1..m]} \overline{x_i}$, where $t_i, t_i', u_i \in T(F, X \cup N)$. $x_i \in X \cup N$.

**Definition 15** (Satisfiability of an ACF). An abstraction constraint formula $\bigwedge_{i \in [1..n]} \overline{x_i} \land \bigwedge_{i \in [1..m]} \overline{x_i}$ is satisfiable iff there exists at least one instantiation $\theta$ such that $\bigwedge_{i \in [1..n]} \overline{x_i} \land \bigwedge_{i \in [1..m]} \overline{x_i} \neq \theta v_i$.

The instantiation $\theta$ is then said to satisfy the ACF $A$ and is called solution of $A$.

For a better readability on examples, we can propagate $\sigma$ into $A$ (by applying $\sigma$ to $A$), thus getting instantiated abstraction constraints of the form $t_i \rightarrow t_i'$ from initial abstraction constraints of the form $t_i \rightarrow t_i'$.

An ACF $A$ is attached to each term $t$ in the proof tree. The ground substitutions solutions of $A$ defining the instances of the current term $u$, for which we are observing IPAS termination, when $A$ has no solution, the current node of the proof tree represents no ground term. Such nodes are then irrelevant for the proof. Detecting and suppressing them during a narrowing step allows us to control the narrowing mechanism, well known to easily diverge. So, we have the choice between generating only the relevant nodes of the proof tree, by testing the satisfiability of $A$ at each step, or stopping the proof on a branch on an irrelevant node, by testing the unsatisfiability of $A$.

Checking the satisfiability of $A$ is in general undecidable, but it is often easy in practice to exhibit an instantiation satisfying it. Automatable sufficient conditions are also under study. The unsatisfiability of $A$ is also undecidable in general, but simple automatable sufficient conditions can be used [20], as to test whether $A$ contains equalities $t_i \rightarrow u_i$, where $u_i$ is reducible. In the following, we present the procedure exactly simulating the rewriting trees, i.e. dealing with the satisfiability of $A$.

### 5. The Algorithm

We are now ready to describe the inference rules defining our mechanism. They transform a set $T$ of 3-tuples $(U, A, C)$ where $U = \{t\}$ or $\emptyset$, $t$ is the current term whose ground instances have to be proved IPAS terminating. $A$ is an abstraction constraint formula, $C$ is a conjunction of ordering constraints.

- The first rule abstracts the current term $t$ at given positions $i_1, \ldots, i_p$ into $t[X_j]_{j \in \{i_1, \ldots, i_p\}}$. The constraint $\bigwedge_{i \in \{i_1, \ldots, i_p\}} t_{i_j} > t_j$ is set in $C$. We do not need to associate any probability to the resulting term. The abstraction constraint $\bigwedge_{i \in \{i_1, \ldots, i_p\}} t_{i_j} = X_j$ is added to the ACF $A$. We call this rule Abstract.

The abstraction positions are chosen so that the abstraction mechanism captures the greatest possible number of rewriting steps; then we abstract all of the greatest possible subterms of $t = f(t_1, \ldots, t_m)$. More concretely, we try to abstract $t_1, \ldots, t_m$, and, for each $t_j = g(t_j, \ldots, t_n)$ that cannot be abstracted, we try to abstract $t_1', \ldots, t_n'$, and so on. In the worst case, we are driven to abstract leaves of the term, which are either variables, or constants.
is a constant, then to a term \( u \) as follows. When the induction hypothesis cannot be applied of every ground instance of \( u \) Narrow to a state, the current term has an associated probability if inductive reasoning, we then associate the alternative IPAS terminating. In the previous first and third steps of above. Otherwise, we can use the notion of usable rule, as be a predicate that is true iff every ground instance of \( u \) the inductive reasoning, we then associate the alternative narrowings, \( \mu \). Indeed, by Definition 9, every \( X \in T(F,N) \), and does not have any \( \sigma = (\sigma, C) \) with the respective associated narrowing substitutions \( \sigma_1 = 1d, \sigma_1 = 1d, \sigma_2 = (X = a), \sigma_2 = (X = a). \) stops if its leaves are states of the form \((\emptyset,A,C)\). We write \( \text{SUCCESS}(g,\succ) \) if the application of \( \phi \) to the condition \( t > a \). It is true in particular when \( u \in T(F,N) \) and is not narrowing. as above. Otherwise, we can use the notion of usable rule, as in \[20]. The rules are given in Table 1. They use a reference term \( t_{ref} = g(x_1, \ldots, x_m) \), where \( x_1, \ldots, x_m \in \mathcal{X} \) and \( g \in \mathcal{D} \) (if \( g \) is a constant, then \( t_{ref} = g \)). Note that, when a rule applies to a state, the current term has an associated probability if it has been generated by Narrow, and does not have any if it has been generated by Abstract. Hence the notation \{t; p\} in Table 1.

We generate the proof trees of \( \mathcal{R} \) by applying, for each defined symbol \( g \in \mathcal{D} \), the inference rules using the reference term \( t_{ref} = g(x_1, \ldots, x_m) \) on the initial set of 3-tuples \( \{ (t_{ref} = g(x_1, \ldots, x_m)), \top, \top \} \), with a specific strategy \( S \), repeating the following steps: first, apply Abstract, and then try Stop. Then try all possible applications of Narrow. Then, try Stop again.

Let us clarify that if \( A \) is satisfiable, the transformed forms of \( A \) by Abstract and Stop are also satisfiable. Moreover, the first application of Abstract generates \( A = (\land_i x_i \downarrow = X_i) \), always satisfied by the constructor constant supposed to exist in \( \mathcal{F} \). Thus, with strategy \( S \), it is useless to prove the satisfiability of \( A \) in the Abstract and Stop rules.

The process may not terminate if there is an infinite number of applications of Abstract and Narrow on the same branch of a proof tree. Nothing can be said in that case about termination. The process stops if no inference rule applies anymore. Then, when all branches of the proof trees end with an application of Stop, IPAS termination is established.

Given a proof tree, to every policy \( \phi \in \Phi_{\text{trn}} \) is associated a deterministic subtree of the proof tree, called \( \phi \)-deterministic subtree of the proof tree, expressing only probabilistic choices. In practice, it is obtained by only considering, at every branching node, the branches corresponding to a same probabilistic narrowing step, for a given position and a given rule.

A finite proof tree or one of its subtrees is said to be successful if its leaves are states of the form \((\emptyset,A,C)\). We write \( \text{SUCCESS}(g,\succ) \) if the application of \( \phi \) to the condition \( t > a \) gives a successful proof tree, whose sets \( C \) of ordering constraints are satisfied by the same ordering \( \succ \).

**Proposition 1.** Let \( \mathcal{R} \) be a probabilistic rewrite system on \( T(F,N) \) having at least one constructor constant. If there is a noetherian ordering \( \succ \) such that for each symbol \( g \in \mathcal{D} \), we have \( \text{SUCCESS}(g,\succ) \), then every term of \( T(F) \) is IPAS terminating.
In the proof of Proposition 1 [19], the information given by probabilities is not used. This means that for RS’s whose proof trees are finite, our method works as in the non-probabilistic case. This corroborates -and gives a formal proof of- the fact that if we remove the probabilities in a given RS, and replace the probabilistic choice by an non-deterministic choice, innermost termination of the resulting RS implies IPAS termination of the initial probabilistic system. So the probabilistic extension of our inductive approach is of real interest for systems whose IPAS termination is due to a probabilistic argument on infinite rewriting chains. We investigate this case in the next section.

Example 1. The following RS

\[ f(0,1,x) \rightarrow f(x,x,x) : 1 \]
\[ g(x,y) \rightarrow x : 1/10 \mid y : 9/10 \]

whose non-probabilistic transformation:

\[ f(0,1,x) \rightarrow f(x,x,x) \]
\[ g(x,y) \rightarrow x \]
\[ g(x,y) \rightarrow y \]

is well known to be innermost terminating, illustrates the above purpose.

Let us develop nevertheless the IPAS termination proof on the probabilistic RS to show how our technique works.

The defined symbols of \( F \) are here \( f \) and \( g \). Applying the rules on \( f(x_1,x_2,x_3) \), we get:

\[ t_{ref} = f(x_1,x_2,x_3) \]
\[ A = \top, C = \top \]
\[ \text{Abstract} \]
\[ f(X_1,X_2,X_3) \]
\[ A = (x_1 \downarrow = X_1 \land x_2 \downarrow = X_2 \land x_3 \downarrow = X_3) \]
\[ C = (f(x_1,x_2,x_3) > x_1,x_2,x_3) \]
\[ \sigma = (x_1 = 0, x_2 = 1) \]
\[ \text{Narrow} \]
\[ f(X_1,X_3,X_3) : 1 \]
\[ A = (x_1 \downarrow = 0 \land x_2 \downarrow = 1 \land x_3 \downarrow = X_3) \]
\[ C = (f(x_1,x_2,x_3) > x_1,x_2,x_3) \]
\[ \text{Stop} \]
\[ A = (x_1 \downarrow = 0 \land x_2 \downarrow = 1 \land x_3 \downarrow = X_3) \]
\[ C = (f(x_1,x_2,x_3) > x_1,x_2,x_3) \]

**Abstract** applies since \( f(x_1,x_2,x_3) > x_1,x_2,x_3 \) is satisfiable by any simplification ordering.

**Narrow** applies because \( A \land \sigma = (x_1 \downarrow = 0 \land x_2 \downarrow = 1 \land x_3 \downarrow = X_3) \), where \( \sigma = (X_1 = 0 \land X_2 = 1) \), is satisfiable by any ground instantiation \( \theta \) such that \( \theta x_1 = 0, \theta x_2 = 1 \) and \( \theta x_3 = \theta X_3 = 0 \).

Then **Stop** applies because \( f(X_1,X_3,X_3) \) is a non-narrowable term whose all variables are abstraction variables, and hence we have IPAST \((f(X_1,X_3,X_3))\).

Considering now \( g(x_1,x_2) \), we get the proof tree in Table 2.

\[ g(x_1,x_2) \rightarrow x_1,x_2 \] is satisfiable by any simplification ordering.

**Narrow** applies because \( A \land \sigma = (x_1 \downarrow = X_1 \land x_2 \downarrow = X_2) \), where \( \sigma = 1d \), is satisfiable by any ground instantiation \( \theta \) such that \( \theta x_1 = \theta X_1 = 0 \) and \( \theta x_2 = \theta X_2 = 0 \).

Then **Stop** applies on both branches because \( X_1 \) and \( X_2 \) are abstraction variables, hence we trivially have IPAST(X_1) and IPAST(X_2).

6. One Step further: Considering Infinite Proof Trees

Consider the following RS \( R \), which is IPAS terminating, but not terminating.

**Example 2.** \{ \( a \rightarrow a : 1/2 \mid b : 1/2 \) \}.

Here, innermost termination is equivalent to termination since we only have constants.

The only defined symbol of \( R \) is \( a \). So the previous algorithm generates the unique following proof tree:

\[ t_{ref} = a \]
\[ A = \top, C = \top \]
\[ \text{Narrow} \]
\[ \sigma = 1d \]
\[ a : 1/2 \]
\[ A = \top, C = \top \]
\[ \text{Stop} \]
\[ a : 1/2 \]
\[ A = \top, C = \top \]
\[ \text{Stop} \]
\[ a : 1/2 \]
\[ A = \top, C = \top \]
\[ \text{Stop} \]
\[ a : 1/2 \]
\[ A = \top, C = \top \]
\[ \text{Stop} \]
\[ a : 1/2 \]
\[ A = \top, C = \top \]
\[ \text{Stop} \]
\[ a : 1/2 \]
\[ A = \top, C = \top \]
\[ \text{Stop} \]
\[ a : 1/2 \]
\[ A = \top, C = \top \]
\[ \text{Stop} \]
\[ a : 1/2 \]
\[ A = \top, C = \top \]
\[ \text{Stop} \]
\[ a : 1/2 \]
\[ A = \top, C = \top \]
\[ \text{Stop} \]
\[ a : 1/2 \]
\[ A = \top, C = \top \]
\[ \text{Stop} \]
\[ a : 1/2 \]
\[ A = \top, C = \top \]
\[ \text{Stop} \]
\[ a : 1/2 \]
\[ A = \top, C = \top \]
\[ \text{Stop} \]
\[ a : 1/2 \]
\[ A = \top, C = \top \]
\[ \text{Stop} \]
\[ a : 1/2 \]
\[ A = \top, C = \top \]
\[ \text{Stop} \]
\[ a : 1/2 \]
\[ A = \top, C = \top \]
\[ \text{Stop} \]
\[ a : 1/2 \]
\[ A = \top, C = \top \]
\[ \text{Stop} \]
\[ a : 1/2 \]
\[ A = \top, C = \top \]

The first branch of this proof tree is infinite. Thanks to the lifting mechanism (obvious here since terms are constants), it represents the infinite rewriting branch of the derivation tree starting from \( a \). All other possible branches are finite. If we now consider the narrowing steps with the probabilities defined by the rule used, we observe that the infinite branch has the probability \( 1/2 \times 1/2 \times 1/2 \times \ldots = 0 \). But for every possible ground term represented by \( t_{ref} \) (here, the only constant \( a \)), there is at least one finite branch. Then, by definition of IPAS termination, \( a \) is IPAS terminating.

Let us now generalize and formalize this reasoning.

Definition 16. A proof tree, whose root state is noted \( s_0 \), is said infinitely successful if for every \( \phi \in \Phi_{fan} \), the \( \phi \)-deterministic subtree of the proof tree either is successful or fulfills the following conditions:

- there is one branch starting from \( s_0 \) with two states \( s_m \) and \( s_n \) such that \( s_n = s_m \),
- the states \( s_i = \{ (t_i : p_i) \}, (A_i, C_i) \) on this branch between \( s_m \) and \( s_n \) are such that \( A_i = A_m \) and \( C_i = C_m \),
- every state on this branch from \( s_0 \) until \( s_{m-1} \) has only brother states that are roots of successful subtrees.

Note that this definition subsumes the previous definition of successful proof tree given in Section 5. Note also that it implies that the sequence \( s_m, \ldots, s_n \) defines a cycle. Indeed, strategy \( S \) applies the inferences rules in the same way on
two equal states. Moreover, the cycle is unique because of the third condition of the definition.

We write $I-$SUCCESS$(g, \succ)$ if the application of $S$ on $\{t_{\text{ref}} = g(x_1, \ldots, x_m)\}$, $\mathcal{T}, \mathcal{L}$ gives an infinitely successful proof tree, whose sets $C$ of ordering constraints are satisfied by the same ordering $\succ$.

**Theorem 1.** Let $\mathcal{R}$ be a probabilistic rewrite system on $\mathcal{T}(F, X)$ having at least one constructor constant. If there is a noetherian ordering $\succ$ such that

- for each symbol $g \in \mathcal{D}$, we have $I-$SUCCESS$(g, \succ)$,
- for the cycle $\langle s_i = \{t_i : p_i \}, A, C, \mathcal{M} \rangle, i \in [m, n]$ with $s_n = s_m$, if it exists, of every $\phi$-deterministic proof subtree of the proof trees, there is $i$ such that $p_i < 1$,

then every term of $\mathcal{T}(F)$ is IPAS terminating.

Consider now the branch from $s_0$ to $s_n$ in Definition 16. We observe that if $A$ and $C$ do not change between $s_m$ and $s_n$, then the **Abstract** rule has not been applied between the two states. Only the **Narrow** rule has been applied and with narrowing substitutions equal to $I\delta$ (up to a variable renaming) on the given branch.

The class of RS's $\mathcal{R}$ generating proof trees whose possible cycles in the $\phi$-deterministic proof subtrees of the proof trees are such that:

- the first term of the cycle is of the form $f(x_1, \ldots, x_m)$ where the $x_i$ are either variables or constructor constants, and $f$ can be a constant,
- the successive rewrite rules of $\mathcal{R}$ used in the $k$ **Narrow** steps of the cycle are of the form
  $$f_j(x_1^j, \ldots, x_m^j) \rightarrow M_j = |i_j, t_{i_j} : p_{i_j}, j \in [1..k]|$$
  where $x_1^j, \ldots, x_m^j$ are also either variables or constructor constants, and the $f_j$ can be constants,
- $f_1(x_1^1, \ldots, x_m^1) = f(x_1, \ldots, x_m)$
- for $j \in [1..k - 1]$, the term $t_{i_j}$, for some $i_j$, generated by the rule $f_j(x_1^j, \ldots, x_m^j) \rightarrow M_j = |i_j, t_{i_j} : p_{i_j}$ on the branch of the cycle is equal to $f_{j+1}(x_1^{j+1}, \ldots, x_m^{j+1})$ (if $k = 1$, this condition is void),
- the term $t_{i_k}$, for some $i_k$, generated by the rule $f_k(x_1^k, \ldots, x_m^k) \rightarrow M_k = |i_k, t_{i_k} : p_{i_k}$ on the branch of the cycle is equal to $f(x_1, \ldots, x_m)$.

fulfills the above conditions on **Abstract** and **Narrow**. For a proof of this fact, see [19].

An important subclass of this class is the class $\mathcal{A}$ of RS's on constants, like the RS of the previous example, whose (I)PAS termination can now be proved.

Thanks to Theorem 1, the proof tree just has to be developed as follows. The branch having a cycle is stopped as soon as the cycle is detected, i.e. when a same state arises twice on the branch.

![Proof tree of the symbol $g$ in Example 1](image)

Another important subclass of this class is the class $\mathcal{B}$ of RS's of the form

$$\{f_j(x_1^{j_1}, \ldots, x_m^{j_1}) \rightarrow M_j = |i_j, t_{i_j} : p_{i_j}, j \in [1..k]| \}

where

- $x_1^{j_1}, \ldots, x_m^{j_1}$ are either variables or constructor constants, and the $f_j$ can be constants,
- for each $j \in [1..k]$, at most one $t_{i_j}$ is a left-hand side of rule $f_l(x_1^l, \ldots, x_m^l)$ for some $l \in [1..k]$, and the other possible $t_{i_j}$ of $M_j$ are not narrowable.
For this class, it can even be proved that all proof trees are infinitely successful (with any simplification ordering) \[19\]. If in addition, for every rule, at least one of the \(t_i\) of \(M_j\) is not narrowable, the second condition of Theorem 1 is fulfilled, hence the following result.

**Corollary 1.** Let \(R \in \mathcal{B}\). If every rule of \(R\) has at least a non-narrowable term in the distribution of its right-hand side, then \(R\) is \text{IPAS} terminating.

The previous example can also be proved (I)PAS terminating directly using Corollary 1.

This is not the case for the following RS, in the class \(A\) but not in \(B\), that requires to develop the proof trees.

**Example 3.** The RS \(\{a \rightarrow a : 1/4|c : 3/4, b \rightarrow a : 1/4|b : 3/4\}\) is (I)PAS terminating. The proof trees are:

In the second proof tree, \textbf{Stop} applies on \(a\) because a.

Note that on such an example, where the inductive principle is crucial, the real interpretation technique of \[4, 5\], is very hard to apply. Because this technique involves arguments that are local to one rule, and are not modular w.r.t. rules, this is also the case for examples where the cycle is generated by more than one rule like \(\{a \rightarrow c : 1, c \rightarrow a : 1/2|b : 1/2\}\), and that we easily handle.

For proofs of the results and additional examples like \(\{f(0,1,x) \rightarrow f(0,1,x) : 1/2|f(x,x,x) : 1/2|g(x,y) \rightarrow x : 1/10|y : 9/10\}\), examples with negations of substitutions, or cycles of length greater than 1, see \[19\].

7. **Conclusion**

In this paper, we have studied the termination problem of probabilistic rewrite systems. We have adapted the inductive technique, which we had proposed for termination of rewriting under strategies, to the probabilistic case. It consists in generating proof trees modeling rewriting trees on ground terms, by alternatively applying abstracting and narrowing steps. As a non-probabilistic RS can be seen as a probabilistic RS whose right-hand sides only have distributions with a unique term of probability 1, the theorem given here subsumes the results given in \[14, 20\] for termination under the innermost strategy.

Note the main differences of this generalization with the technique we proposed for the non-probabilistic case:

- the lifting mechanism enabling to simulate rewriting trees by proof trees: we have extended the usual narrowing relation to a probabilistic narrowing relation, in adequation with the considered probabilistic rewriting mechanism. To establish correctness of Proposition 1 and Theorem 1, we have generalized the classical lifting lemma of Middendorp and Hamoen \[31\] to the probabilistic case \[19\].
- the infinite proof trees, always generated when termination is due to probabilistic arguments, and not considered in the non-probabilistic case.

We have also given a class of RS’s for which this generalization is of interest. An interesting subclass of this class is composed by the RS’s on constants, like the first three examples of Section 6. Indeed, constants can modelize states of automata used for expressing protocols, and it often happens that probabilistic protocols regularly fall in the same state when they evolve. It can then be crucial to prove that such cycling situations have a null probability of occurring. Our technique allows it to happen.

In a more general way, our application area can seem limited, because of the restricted form of the rules in cycles we tackle at the moment, but most randomized algorithms \[33\] or telecommunication protocols (e.g. CSMA-CA protocol \[10\]) based on probabilistic arguments rely on very simple arguments involving very simple probabilistic rewrite rules. The reasoning for these rules, however, is often difficult to do \[33\]. This paper provides a way to do inductive reasoning for probabilistic systems. As far as we know, there have not been many investigations on this subject.

Moreover, the completeness results of \[4, 5\], based on real interpretations, are nice from a theoretical point of view, but not constructive, and no algorithmic help exists yet, to exhibit ad-hoc interpretations.

To the contrary, our method is operational. Detecting a cycle as specified in Definition 16 is automatable. As said before, for our approach, there are sufficient conditions for testing the unsatisfiability of \(A\) and \(C\) is often easy to satisfy with usual orderings. So our method can be automatized, and in a very simple way in the interesting case of RS’s on constants, where \(A = C = T\).

Finally, note the important fact that, if \(R\) is deterministic, innermost derivations are equivalent to standard derivations. So, our proof technique also establishes PAS termination of \(R\) for the standard strategy.

We now plan to generalize our theorem using infinite proof trees on a larger class of systems, and to investigate other techniques to ensure PAS termination.

8. **Acknowledgments**

We would like to thank Olivier Bournez for fruitful discussions on termination of probabilistic rewriting and on this paper, and for giving us motivating examples.

**References**


Preprint version - Definitive ACM version available at https://dl.acm.org/citation.cfm?id=1273943 11 2018/10/10