Rheinisch-Westfälische Technische Hochschule Aachen

Master's Thesis

Using Dependency Pairs for Proving Almost-Sure Termination of Probabilistic Term Rewriting

by

Jan-Christoph Kassing

Lehr- und Forschungsgebiet Informatik 2
Programming Languages and Verification

First Examiner:
Prof. Dr. Jürgen Giesl

Second Examiner:
Prof. Dr. Joost-Pieter Katoen

September 26, 2022

Abstract

In this thesis, we develop the first automatic termination analysis tool for probabilistic term rewriting systems based on dependency pairs. A term rewriting system is a formal computational model that is successfully used as a backend language of tools for analyzing other programming languages (e.g., Java). The dependency pair approach is one of the most powerful techniques to automatically analyze the termination of term rewriting systems. Research has already been carried out in this area for decades, resulting in a framework for automatic termination and runtime analysis.

We adapt this dependency pair framework to the probabilistic setting. Here, we have not only the possibility of deterministic or non-deterministic rewrite steps but also the possibility to perform a certain rewrite step only with a certain probability. For probabilistic programs, there exist several notions of "termination". In this thesis, we focus on innermost almost-sure termination, which means that the probabilistic term rewriting system terminates with probability one, and we can only rewrite at innermost positions. Whether we can further adapt the results of this thesis to handle an arbitrary evaluation strategy or a different notion of "termination" (e.g., positive almost-sure termination) remains an open question that we plan to address in the future.

In the end, we result in a new probabilistic dependency pair framework that automatically proves innermost almost-sure termination of probabilistic term rewriting systems. For this, we fundamentally changed the definition of dependency pairs and chains. We adapted three of the most important processors to the probabilistic setting: the reduction pair processor, the dependency graph processor, and the usable rules processor. Moreover, we also added two new processors designed explicitly for probabilistic term rewriting systems. We are currently implementing this new probabilistic framework in the Automated Program Verification Environment (AProVE).

Erklärung der Urheberschaft

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ort, Datum Unterschrift

Acknowledgements

I want to express my most profound appreciation to my supervisors Prof. Jürgen Giesl and Dr. Florian Frohn, for their patience, guidance, and all the ideas they contributed to this work. Special thanks to Prof. Jürgen Giesl for proofreading my initial ideas and proofs. Moreover, I want to thank everybody in the research group computer science 2 "Programming Languages and Verification", for their initial ideas on the subject. This made it much easier to get started. Thanks should also go to my study colleagues, particularly Paul Wirkus, Marvin Müller and Johannes Raufeisen, for their proofreading and suggestions, who were initially entirely foreign to the subject. I am also grateful for the support of my parents, brother, and sister. Specifically, my brother always inspired me to become a researcher myself. Lastly, I want to thank my partner for her moral support and belief in me.

Contents

1 Introduction

For many decades, term rewriting has shown to be a very powerful tool for several aspects of computer science. It has been used for equational reasoning, computer algebra systems, the study of the notion of computability, to implement functional programming languages, automatic verification, and so on [3]. A term rewriting system (TRS) is a model for computation. It consists of a finite number of rewrite rules $\ell \to r$ that describe how the system rewrites a term, which is a purely syntactical object. In verification, TRSs are used as a backend language of tools for analyzing other programming languages, e.g., Java [30]. Such a tool first transforms a given program $\mathcal{P}$ into a TRS $\mathcal{R}$ and then proceeds by analyzing $\mathcal{R}$. Due to its simple syntactic structure, it is easier to work with the TRS than the program itself. In this thesis, we are mainly interested in the automatic verification that a given TRS is terminating. The problem is undecidable in general [5]. Still, there are many approaches to automatically prove termination for a large number of TRSs, including TRSs that occur in real-world applications, e.g., using lexicographic path orders [18], using recursive path orders [10], using polynomial interpretations [24], etc. One of the most powerful approaches in practice is the dependency pair framework [1, 14]. Here, the main idea is to use a divide-and-conquer framework and to partition a big problem that is hard to solve into many simpler subproblems. To perform such a partitioning step, we use so-called DP processors. For the dependency pair framework, we refer to [1] for the first definition of dependency pairs and chains. The precise processors we use in this thesis can be found in [1, 14, 13]. In the probabilistic setting, we will use a new type of dependency tuples instead of dependency pairs. Dependency tuples were already used for the complexity analysis of TRSs [29]. In the future, this new type of dependency tuples may also be used to enhance the complexity analysis in the non-probabilistic setting.

Another very important research area is probabilistic programming and the verification of such a program. The connection between probability theory and computer science has been studied for a long time and has led to many interesting results. It turns out that many probabilistic algorithms are more efficient than their deterministic counterpart (e.g., [31]). Another area of computer science improved by probability theory is IT security, where we now have probabilistic encryption algorithms [15]. Probabilistic variants of computational models like Turing machines, automata, and λ-calculus have already been studied for a long time [21]. In the probabilistic setting, we have many notions of "termination". A qualitative translation of the non-probabilistic version would be almost-sure termination. A program is almost-surely terminating (AST) iff the probability for convergence tends to 1. A strictly stronger notion of termination is positive almost-sure termination (PAST). Here, we additionally require that the expected runtime is finite. It was shown that the decision problem for AST and PAST is even harder than the decision problem for termination. While the former problem is recursively enumerable, the decision problem whether a program is AST (and PAST) is already Π^0_2 (and Σ^0_2) complete [19]. There are many different approaches to prove AST based on weakest pre-expectation calculus [20],

abstract interpretations [28], type systems [4], and martingales [7, 8, 27]. However, only a few methods are suitable for programs with a complicated recursive structure or programs that mainly target algebraic structures like graphs, lists, etc. For recursive probabilistic programs on numbers, sized types have been proposed [9]. There is also an automated type-based analysis of probabilistic functional programs with data types from Wang et al. [32]. Most recently, Leutgeb et al. [25] created a fully-automated expected amortized cost analysis for probabilistic data structures that is based on this type-based analysis. They mainly target amortized costs of randomized variants of self-adjusting data structures like randomized splay trees, randomized splay heaps, and randomized meldable heaps. Furthermore, little research exists regarding a probabilistic variant of TRSs. In this thesis, we try to enhance both of these research areas further. The definition of a probabilistic TRS (PTRS) used in this thesis is based on [6, 2, 11]. Here, probabilistic rewriting is captured as a Markov decision process. A single rule may have multiple possible outcomes with a certain probability, while the rule selection remains non-deterministic. Hence, both non-determinism and probabilistic aspects are present. We will see that their combination leads to interesting results that differ from the non-probabilistic setting, where we only have determinism and non-determinism. While [11] adapts general concepts of abstract rewrite systems like confluence to the probabilistic setting, the notion of a PTRS was introduced in [6] and Avincini et al. [2] then enhanced their results and showed how the interpretation method could be adapted to prove PAST of a given PTRS. We will use their work and definitions regarding PTRSs, but instead of PAST, we want to prove AST automatically. The reason for that is that the dependency pair framework relies on the compositionality of the property that we want to analyze to modularize the termination proof. It is well known that while AST is compositional, PAST is not. We will also adapt the interpretation method of [2] and enhance it to prove AST for a PTRS.

This thesis develops the first technique to prove AST for PTRSs using dependency pairs automatically. For this, we adapt the dependency pair framework to the probabilistic setting. We will restrict our attention to innermost evaluation, which means that we only allow rewrite steps performed at an innermost position of a term. For the termination analysis of non-probabilistic TRSs, specific techniques exist for both innermost and arbitrary evaluation. We adapt the DP framework for innermost termination to prove innermost almost sure-termination of a given PTRS automatically. This adaption includes an entirely new definition of probabilistic dependency pairs and probabilistic chains. Furthermore, we adapt the most important processors to our new probabilistic framework, including the reduction pair processor, the dependency graph processor, and the usable rules processor. In addition to those three processors from the non-probabilistic framework, we introduce two new processors specifically designed for our new definition of probabilistic dependency pairs. We prove their soundness and completeness for all five processors, except the usable rules processor. The usable rules processor is only sound and not complete in the way we present it, but there exists a complete variant that requires further definitions that we omit for readability. In future work, we expect that our new DP framework can not only be used to prove AST, but it may also be possible to advance the framework further so that it can disprove AST for a given PTRS as well. Since the DP framework is a very powerful tool in the non-probabilistic setting, the goal is to achieve the same for the probabilistic setting. We are currently implementing this framework in the Automated Program Verification Environment (AProVE) and will further evaluate its applicability in the future.

Regarding the structure of this thesis, we start with the basic notation and definitions regarding term rewriting in Chapter 2. This also includes a first approach to prove termination using a direct application of polynomial interpretations. Then we introduce

the dependency pair framework for TRSs in Chapter 3. Here, we first define the notion of a dependency pair and a chain and then prove the chain criterion which explains how we start our framework. At the end of this chapter, we introduce the three most important processors we will adapt to the probabilistic setting and give an example of how this framework works. In Chapter 4, we present PTRSs and introduce all required notions and notations. After some new results regarding evaluation strategies, we present a novel way to infer AST using polynomial interpretations directly. Finally, in Chapter 5 we define the probabilistic dependency pair framework, including an entirely new definition of dependency pairs and chains, prove the chain criterion for this new type of chain, create the five processors that we already mentioned above, prove their soundness and completeness, and again give an example how this framework works. After that, we compare the non-probabilistic dependency pair framework with our new probabilistic variant in Chapter 6. We conclude in Chapter 7. In every chapter, we introduce a new type of rewrite system and prove certain witness theorems that show what kind of rewrite sequences we have to analyze to prove termination or AST. Furthermore, in every chapter, we show how we can automatically prove termination or AST using an application of polynomial interpretations. The proof structure for those reoccurring theorems is the same in every chapter. However, the details get more and more complicated in later chapters.

2 Term Rewriting

This chapter introduces the basic notions and notations of *term rewriting*. In Section 2.1, we define the syntax and semantics of the objects we want to work with. In term rewriting, one works with terms as the main object. A *term rewriting system* is then a set of rewrite rules describing how we can rewrite one term into another. Several different evaluation strategies exist for applying a rewrite rule to a term. After introducing and comparing some of the most important ones, we restrict our attention to innermost evaluation for the rest of this thesis. See e.g., [3] for a complete introduction to term rewriting in general. In Section 2.2, we present a first automatic approach to prove innermost termination of a given term rewriting system using so-called *polynomial interpretations*.

2.1 Syntax, Semantic and Evaluation Strategies

We start with the most basic definition of an *abstract rewrite system*. This is simply a set together with a binary relation.

Definition 2.1.1 (Abstract Rewrite System)**.** Let A be a non-empty set. A *rewrite relation* is a binary relation $\to \subseteq A \times A$. We often write "$a_1 \to a_2$" instead of "$(a_1, a_2) \in \to$" and say that $\to$ *acts* on A. The pair $(A, \to)$ is an *abstract rewrite system* (ARS). A (potentially infinite) sequence of elements $a_0, a_1, a_2, \ldots$ such that $a_i \to a_{i+1}$ for all i is called a rewrite sequence and will be denoted as

$$a_0 \to a_1 \to a_2 \to \ldots$$

We will only work with abstract rewrite systems acting on terms for this and the next chapter. Once we introduce probabilistic term rewriting and the probabilistic DP framework in Chapter 4 and Chapter 5, we also introduce other kinds of abstract rewrite systems that act on distributions or certain sets. Next, we define a *signature* from which we can build a term.

Definition 2.1.2 (Signature)**.** A *signature* $\Sigma = \biguplus_{n \in \mathbb{N}} \Sigma_n$ is a union of pairwise disjoint finite sets. An element $f \in \Sigma_n$ is called a function symbol of *arity* n. The elements of Σ_0 are called *constants*. We will always assume that Σ is finite and that $\Sigma_0 \neq \varnothing$.

Example 2.1.3 (Signature)**.** An example for a signature that we use throughout this whole chapter is $\Sigma_{plus} = \biguplus_{n \in \mathbb{N}} \Sigma_n$ with $\Sigma_0 := \{\mathcal{O}\}$, $\Sigma_1 := \{\mathsf{s}\}$, $\Sigma_2 := \{\mathsf{plus}\}$ and $\Sigma_n := \varnothing$ for all $n \geq 3$.

We can now define how to construct a term from a given signature.

Definition 2.1.4 (Term)**.** Let Σ be a signature and $\mathcal{V}$ be a non-empty, countable set of *variables* with $\mathcal{V} \cap \Sigma = \varnothing$. The set of *terms* $\mathcal{T}(\Sigma, \mathcal{V})$ over a signature Σ and a set of variables $\mathcal{V}$ for this signature is the smallest set with

- $\mathcal{V} \subseteq \mathcal{T}(\Sigma, \mathcal{V})$,
- If $f \in \Sigma_n$ and $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{V})$ then $f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{V})$.

For any term t, let $\mathcal{V}(t)$ be the set of variables occurring in t, which is recursively defined by

- If $t = x \in \mathcal{V}$ is a variable, then $\mathcal{V}(t) = \{x\}$,
- If $t = f(t_1, \ldots, t_n)$ with $f \in \Sigma_n$ and $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{V})$, then $\mathcal{V}(t) = \bigcup_{1 \leq i \leq n} \mathcal{V}(t_i)$.

A term t is called a *ground term* if we have $\mathcal{V}(t) = \varnothing$. We often write $\mathcal{T}(\Sigma)$ instead of $\mathcal{T}(\Sigma, \varnothing)$ for the set of ground terms.

Example 2.1.5 (Term)**.** Let Σ_{plus} be the signature from Example 2.1.3 and let $\{x, y\} \subseteq \mathcal{V}$. Then we have $t_1 \coloneqq \mathcal{O} \in \mathcal{T}(\Sigma_{plus}, \mathcal{V})$, $t_2 \coloneqq \mathsf{s}(x) \in \mathcal{T}(\Sigma_{plus}, \mathcal{V})$, and $t_3 \coloneqq \mathsf{plus}(\mathsf{s}(y), \mathsf{s}(\mathsf{s}(\mathcal{O}))) \in \mathcal{T}(\Sigma_{plus}, \mathcal{V})$. In this case, only t_1 is a ground term, as the other two contain variables.

Definition 2.1.6 (Subterm)**.** Let $t, q \in \mathcal{T}(\Sigma, \mathcal{V})$ be two terms. We say that q is a subterm of t (denoted by $q \trianglelefteq t$) iff $q = t$ or $t = f(t_1, \ldots, t_n)$ and $q \trianglelefteq t_i$ for some $1 \leq i \leq n$. We call q a *proper* subterm (denoted by $q \triangleleft t$) iff $q \trianglelefteq t$ and $q \neq t$.

Example 2.1.7 (Subterm)**.** Consider the terms from Example 2.1.5. Here, we have $t_1 \triangleleft t_3$ and no other subterm relation.

One of the essential definitions in term rewriting is a *substitution*. This is a function that instantiates the occurring variables with a fresh term that may also contain variables again.

Definition 2.1.8 (Substitution)**.** A *substitution* is a function $\sigma : \mathcal{V} \to \mathcal{T}(\Sigma, \mathcal{V})$ with $\sigma(x) = x$ for all but finitely many $x \in \mathcal{V}$. Therefore, σ can be identified with the finite set $\{x/\sigma(x) \mid x \in \mathcal{V}, \sigma(x) \neq x\}$. We often write $x\sigma$ instead of $\sigma(x)$. Substitutions can also be applied to terms: If $t = f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{V})$ then $t\sigma \coloneqq f(t_1\sigma, \ldots, t_n\sigma)$. The set of all substitutions for a given signature Σ and a given set of variables $\mathcal{V}$ is denoted by $\mathrm{Sub}(\Sigma, \mathcal{V})$. A substitution σ that maps every variable to a ground term will be called a ground substitution. Note that a ground substitution is technically not a substitution since we do not have $\sigma(x) = x$ for all but finitely many $x \in \mathcal{V}$ if the variable set is infinite. However, we will always only be interested in a finite amount of variable instantiations for such a substitution so that we can disregard every other instantiation and view it as a valid substitution.

A term $s \in \mathcal{T}(\Sigma, \mathcal{V})$ *matches* a term $t \in \mathcal{T}(\Sigma, \mathcal{V})$ if there exists a substitution σ such that $s\sigma = t$. Two terms $s, t \in \mathcal{T}(\Sigma, \mathcal{V})$ are *unifiable* if there exists a substitution σ such that $s\sigma = t\sigma$. Such a substitution σ is called a *unifier*. A *most general unifier* (mgu) is a unifier σ such that for every other unifier σ' there exists a substitution δ with $\sigma' = \delta\sigma$. It is a well-known fact that for two unifiable terms, there always exists a mgu, and one can compute a mgu in polynomial time.

Example 2.1.9 (Substitution)**.** Consider the terms from Example 2.1.5 and let $\sigma = \{x/\mathsf{s}(\mathcal{O}), y/\mathsf{plus}(x, x)\} \in \mathrm{Sub}(\Sigma_{plus}, \mathcal{V})$. Then $t_1\sigma = \mathcal{O}$, $t_2\sigma = \mathsf{s}(\mathsf{s}(\mathcal{O}))$ and $t_3\sigma = \mathsf{plus}(\mathsf{s}(\mathsf{plus}(x, x)), \mathsf{s}(\mathsf{s}(\mathcal{O})))$.

Two terms that are unifiable would be $\mathsf{plus}(\mathsf{s}(x), y)$ and $\mathsf{plus}(\mathsf{s}(\mathsf{s}(\mathcal{O})), \mathsf{s}(x))$. Here, the most general unifier is $\{x/\mathsf{s}(\mathcal{O}), y/\mathsf{s}(\mathsf{s}(\mathcal{O}))\}$.

Instead of only variables, we are also interested in changing a whole subterm into a new one. To address this subterm, we define the set of all *positions* inside a term.

Definition 2.1.10 (Position)**.** For a term $t \in \mathcal{T}(\Sigma, \mathcal{V})$, the set of *positions* $\mathrm{Occ}(t)$ is defined to be the smallest subset of $\mathbb{N}^*$ satisfying

- $\varepsilon \in \mathrm{Occ}(t)$,
- If $t = f(t_1, \ldots, t_n)$ then for all $1 \leq i \leq n$ and all $\pi \in \mathrm{Occ}(t_i)$ we have $i.\pi \in \mathrm{Occ}(t)$.

The position ε is called the *root position* of the term t and the symbol at this position is called the *root symbol*, denoted by $\mathrm{root}(t)$.

The prefix order

$$\pi \leq \tau :\Leftrightarrow \text{ there exists } \chi \in \mathbb{N}^* \text{ such that } \pi.\chi = \tau$$

is a partial order on $\mathbb{N}^*$. We will always refer to this prefix order when talking about the order of positions. We say that two positions $\pi, \tau \in \mathbb{N}^*$ are *orthogonal* (denoted by $\pi \perp \tau$) iff they are not comparable in this prefix order, i.e., $\pi \not\leq \tau$ and $\tau \not\leq \pi$.

If $\pi \in \mathrm{Occ}(t)$ then $t|_\pi$ denotes the subterm at position π, i.e.,

- $t|_\varepsilon = t$,
- If $t = f(t_1, \ldots, t_n)$ and $\pi = i.\pi'$ then $t|_\pi = t_i|_{\pi'}$.

Furthermore, if $r \in \mathcal{T}(\Sigma, \mathcal{V})$ and $\pi \in \mathrm{Occ}(t)$ then $t[r]_\pi$ denotes the term that results from replacing the subterm $t|_\pi$ with the new term r, i.e., we have:

- $t[r]_\varepsilon = r$,
- If $t = f(t_1, \ldots, t_n)$ and $\pi = i.\pi'$ then $t[r]_\pi = t_i[r]_{\pi'}$.

Example 2.1.11 (Position)**.** Again consider the terms from Example 2.1.5. We have

$$\mathrm{Occ}(t_3) = \{\varepsilon, 1, 2, 1.1, 2.1, 2.1.1\}.$$

Furthermore, $t_3|_2 = \mathsf{s}(\mathsf{s}(\mathcal{O}))$, and $t_3[y]_2 = \mathsf{plus}(\mathsf{s}(y), y)$, and $t_3[\mathsf{s}(\mathcal{O})]_{1.1} = \mathsf{plus}(\mathsf{s}(\mathsf{s}(\mathcal{O})), \mathsf{s}(\mathsf{s}(\mathcal{O})))$.

The following lemma states some important and useful properties for working with positions and subterms

Lemma 2.1.12. *Let $s, t, r \in \mathcal{T}(\Sigma, \mathcal{V})$ be terms and let $\pi, \tau \in \mathbb{N}^*$.*

1. $\mathrm{Occ}(t)$ *is closed under prefixes*
2. *If $\pi.\tau \in \mathrm{Occ}(t)$, then $t|_{\pi.\tau} = (t|_\pi)|_\tau$*
3. *If $\pi \in \mathrm{Occ}(s)$ and $\tau \in \mathrm{Occ}(t)$, then $(s[t]_\pi)|_{\pi.\tau} = t|_\tau$ and $(s[t]_\pi)[r]_{\pi.\tau} = s[t[r]_\tau]_\pi$*
4. *If $\pi.\tau \in \mathrm{Occ}(s)$, then $(s[t]_{\pi.\tau})|_\pi = (s|_\pi)[t]_\tau$ and $(s[t]_{\pi.\tau})[r]_\pi = s[r]_\tau$*
5. *If $\pi, \tau \in \mathrm{Occ}(s)$ and $\pi \perp \tau$, then $(s[t]_\pi)|_\tau = s|_\tau$ and $(s[t]_\pi)[r]_\tau = (s[r]_\tau)[t]_\pi$*

If we represent terms as a tree, these properties are pretty intuitive. But one can also prove everything using the formal definition of positions. Instead of changing a term at a specific position, we can also mark a specific position with a fresh symbol $\Box$, called a *hole*, where we can insert a term. Such a term $\mathcal{C} \in \mathcal{T}(\Sigma \uplus \{\Box\}, \mathcal{V})$ with a hole is called a Context.

Definition 2.1.13 (Context)**.** A term $\mathcal{C} \in \mathcal{T}(\Sigma \uplus \{\Box\}, \mathcal{V})$ is called a *context* if there is exactly one occurrence of $\Box$ inside of $\mathcal{C}$. If $\mathcal{C}|_\pi = \Box$, then $\mathcal{C}[t]$ is a shorthand for $\mathcal{C}[t]_\pi$

Example 2.1.14 (Context)**.** An example of a context over the signature Σ_{plus} would be $\mathcal{C} \coloneqq \mathsf{plus}(\Box, \mathsf{s}(\mathcal{O}))$. Here, we have $\mathcal{C}[\mathsf{s}(\mathcal{O})] = \mathsf{plus}(\mathsf{s}(\mathcal{O}), \mathsf{s}(\mathcal{O}))$.

Finally, we come to the notion of a *term rewriting system*.

Definition 2.1.15 (Term Rewriting System)**.** For $\ell, r \in \mathcal{T}(\Sigma, \mathcal{V})$, $\ell \to r$ is called a *rewrite rule* if $\mathcal{V}(r) \subseteq \mathcal{V}(\ell)$ and $\ell \notin \mathcal{V}$. A *term rewriting system* (TRS) $\mathcal{R}$ is a finite set of rewrite rules.

The TRS $\mathcal{R}$ induces a *rewrite relation* $\to_\mathcal{R} \subseteq \mathcal{T}(\Sigma, \mathcal{V}) \times \mathcal{T}(\Sigma, \mathcal{V})$ via

$$s \to_\mathcal{R} t \qquad :\Leftrightarrow \qquad s|_\pi = \ell\sigma \text{ and } t = s[r\sigma]_\pi,$$

for a position $\pi \in Occ(s)$, a rule $\ell \to r \in \mathcal{R}$, and a substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$. A subterm $s|_\pi$ that is matched by the left-hand side of a rule is called a *redex* (for "reducible expression").

Example 2.1.16 (Term Rewrite System)**.** Consider the TRS $\mathcal{R}_{plus}$ consisting of the following two rewrite rules over the signature Σ_{plus} and a set of variables with $\{x, y\} \subseteq \mathcal{V}$:

$$\mathsf{plus}(\mathcal{O}, y) \to y \tag{2.1}$$
$$\mathsf{plus}(\mathsf{s}(x), y) \to \mathsf{s}(\mathsf{plus}(x, y)) \tag{2.2}$$

This TRS simulates the addition of two natural numbers. Here, numbers are represented as terms using the zero function $\mathcal{O}$ and the successor function s. For example, the number 2 is represented by $\mathsf{s}(\mathsf{s}(\mathcal{O}))$. If we want to calculate "$2 + 1$" with this TRS, we would start with the term $\mathsf{plus}(\mathsf{s}(\mathsf{s}(\mathcal{O})), \mathsf{s}(\mathcal{O}))$. The left-hand side of Rule (2.2) matches the term $\mathsf{plus}(\mathsf{s}(\mathsf{s}(\mathcal{O})), \mathsf{s}(\mathcal{O}))$ using the substitution $\sigma = \{x/\mathsf{s}(\mathcal{O}), y/\mathsf{s}(\mathcal{O})\}$. Hence, we can rewrite on the root position ε so that

$$\mathsf{plus}(\mathsf{s}(\mathsf{s}(\mathcal{O})), \mathsf{s}(\mathcal{O})) \to_{\mathcal{R}_{plus}} \mathsf{s}(\mathsf{plus}(\mathsf{s}(\mathcal{O}), \mathsf{s}(\mathcal{O}))).$$

Then, the left-hand side of Rule (2.2) matches the subterm at position 1 using the substitution $\sigma = \{x/\mathcal{O}, y/\mathsf{s}(\mathcal{O})\}$. Hence, we can rewrite:

$$\mathsf{s}(\mathsf{plus}(\mathsf{s}(\mathcal{O}), \mathsf{s}(\mathcal{O}))) \to_{\mathcal{R}_{plus}} \mathsf{s}(\mathsf{s}(\mathsf{plus}(\mathcal{O}, \mathsf{s}(\mathcal{O})))).$$

Finally, the left-hand side of (2.1) matches the term just obtained at position 1.1. Thus, we have

$$\mathsf{s}(\mathsf{s}(\mathsf{plus}(\mathcal{O}, \mathsf{s}(\mathcal{O})))) \to_{\mathcal{R}_{plus}} \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathcal{O}))).$$

In the end, we result with the term $\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathcal{O})))$ that cannot be evaluated any further and represents the number 3, which is exactly what we wanted to compute.

We cannot apply any further rules to the term $\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathcal{O})))$ in our previous example. Such a term is called a *normal form*. This notion is not restricted to TRS but can be used for arbitrary ARS.

Definition 2.1.17 (Normal Form)**.** Let $(A, \to)$ be a ARS and let $a \in A$. We call a a *normal form* (w.r.t. $\to$) iff there is no $a' \in A$ with $a \to a'$. An element $b \in A$ is a normal form of a iff b is a normal form and $a \to^* b$. Here, $\to^*$ denotes the transitive and reflexive closure of $\to$. An element a is normalizing iff there exists a normal form b of a. An element a is strongly normalizing iff every rewrite sequence $a \to \ldots$ that starts with a will reach a normal form and is therefore finite. By $\mathtt{NF}_\to \subseteq A$ we will denote the set of all normal forms w.r.t. $\to$.

For a TRS $\mathcal{R}$, we will also say that a term is a normal form w.r.t. $\mathcal{R}$ (instead of $\to_\mathcal{R}$), and write $\mathtt{NF}_\mathcal{R}$ for $\mathtt{NF}_{\to_\mathcal{R}}$. Lastly, we come to the property we are mainly interested in, namely *termination*.

Definition 2.1.18 (Termination)**.** Let $\mathcal{R}$ be a TRS. We call it *terminating* if there is no infinite rewrite sequence

$$t_0 \to_\mathcal{R} t_1 \to_\mathcal{R} t_2 \to_\mathcal{R} \ldots$$

In other words, a TRS is terminating iff we always reach a normal form within a finite number of steps, which means that every term is strongly normalizing. Currently, the definition of term rewriting allows to apply rewrite steps at every possible position. An important restriction to this kind of *full* rewriting is so called *innermost* rewriting. Here, one can only rewrite at an innermost possible position.

Definition 2.1.19 (Innermost Rewriting and Termination)**.** Let $\mathcal{R}$ be a TRS and let $s, t \in \mathcal{T}(\Sigma, \mathcal{V})$. We say that a rewrite step $s \to_\mathcal{R} t$ is an *innermost* rewrite step (denoted $s \xrightarrow{\mathsf{i}}_\mathcal{R} t$) if all proper subterms of the redex $\ell\sigma$ are in normal form w.r.t. $\mathcal{R}$.

We call $\mathcal{R}$ *innermost terminating* if there is no infinite innermost rewrite sequence

$$t_0 \xrightarrow{\mathsf{i}}_\mathcal{R} t_1 \xrightarrow{\mathsf{i}}_\mathcal{R} t_2 \xrightarrow{\mathsf{i}}_\mathcal{R} \ldots$$

Note that a term is in normal form w.r.t. $\xrightarrow{\mathsf{i}}_\mathcal{R}$ iff it is in normal form w.r.t. $\to_\mathcal{R}$. Hence, our notation $\mathtt{NF}_\mathcal{R}$ is still valid for innermost rewriting. One can further restrict the rewrite relation, e.g., by only allowing to rewrite the leftmost innermost redex. This is called *leftmost innermost rewriting* and we can again define *leftmost innermost termination*. The next two theorems state the relations between these three types of evaluation strategies.

Theorem 2.1.20 (Innermost Termination vs. Leftmost Innermost Termination, [22])**.** *A TRS $\mathcal{R}$ is innermost terminating iff $\mathcal{R}$ is leftmost innermost terminating.*

Proof. Every infinite leftmost innermost rewrite sequence is also an infinite innermost rewrite sequence. For the proof of the nontrivial direction, we refer to [22]. ■

Theorem 2.1.21 (Innermost Termination vs. Full Termination)**.** *If a TRS $\mathcal{R}$ is terminating, it is also innermost terminating. On the other side, there exists TRSs that are innermost terminating but not terminating.*

Proof. Every infinite innermost rewrite sequence is also an infinite rewrite sequence. A counterexample of the other direction can be seen in Example 2.1.22. ■

Example 2.1.22 (Innermost Termination vs. Full Termination, [16])**.** Consider the TRS $\mathcal{R}$ consisting of the following two rewrite rules over the signature $\Sigma = \Sigma_0 \uplus \Sigma_1$ with $\Sigma_0 = \{\mathsf{a}, \mathsf{b}\}$ and $\Sigma_1 = \{\mathsf{f}\}$:

$$\mathsf{f}(\mathsf{a}) \to \mathsf{f}(\mathsf{a}) \tag{2.3}$$

$$\mathsf{a} \to \mathsf{b} \tag{2.4}$$

Here, $\mathcal{R}$ is innermost terminating as we have to rewrite the proper subterm a before we can use rule (2.3). In contrast to that, $\mathcal{R}$ is not terminating as we have the infinite rewrite sequence $\mathsf{f}(\mathsf{a}) \to_{\mathcal{R}} \mathsf{f}(\mathsf{a}) \to_{\mathcal{R}} \ldots$ that will never reach a normal form.

As the previous example shows, there is no equivalence between innermost and full termination. Still, there are specific syntactical properties of the TRS that guarantee that these two notions are equivalent [16]. For the rest of this thesis, we will restrict our attention to innermost evaluation.

Next, we see that we can restrict ourselves to specific rewrite sequences for innermost termination analysis. First of all, we can restrict the structure of the first term t_0 in a sequence to be the left-hand side of a rewrite rule with possibly instantiated variables such that every proper subterm is in normal form w.r.t. $\mathcal{R}$.

Theorem 2.1.23 (Witness Theorem for TRS)**.** *Let $\mathcal{R}$ be a TRS. If $\mathcal{R}$ is not innermost terminating then there exists an infinite innermost rewrite sequence $t_0 \xrightarrow{\mathsf{i}}_{\mathcal{R}} t_1 \xrightarrow{\mathsf{i}}_{\mathcal{R}} t_2 \xrightarrow{\mathsf{i}}_{\mathcal{R}} \ldots$ such that $t_0 = \ell\sigma$ for some rule $\ell \to r \in \mathcal{R}$ and some substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, such that every proper subterm of $\ell\sigma$ is in normal form w.r.t. $\mathcal{R}$.*

Proof. We will prove the contraposition, namely that if every innermost rewrite sequence $t_0 \xrightarrow{\mathsf{i}}_{\mathcal{R}} t_1 \xrightarrow{\mathsf{i}}_{\mathcal{R}} t_2 \xrightarrow{\mathsf{i}}_{\mathcal{R}} \ldots$ is finite where $t_0 = \ell\sigma$ for some rule $\ell \to r \in \mathcal{R}$ and some substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ and every proper subterm of t_0 is in normal form w.r.t. $\mathcal{R}$, then $\mathcal{R}$ is innermost terminating. We prove this by structural induction over the start term t_0.

If t_0 is in normal form w.r.t. $\mathcal{R}$, then obviously, every innermost rewrite sequence that starts with this term is finite. If $t_0 = \ell\sigma$ for some rule $\ell \to r \in \mathcal{R}$ and some substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ such that every proper subterm of $\ell\sigma$ is in normal form w.r.t. $\mathcal{R}$, then we know by our assumption that every innermost rewrite sequence that starts with this term is finite.

Now we regard the induction step, and assume that $t_0 = f(q_1, \ldots, q_k)$. Due to the innermost evaluation strategy, we can only rewrite at the root position if every proper subterm is in normal form. Hence, we first prove that every rewrite sequence $t_0 \xrightarrow{\neg\varepsilon\mathsf{i}}_{\mathcal{R}} t_1 \xrightarrow{\neg\varepsilon\mathsf{i}}_{\mathcal{R}} t_2 \xrightarrow{\neg\varepsilon\mathsf{i}}_{\mathcal{R}} \ldots$ is finite, where $\xrightarrow{\neg\varepsilon\mathsf{i}}_{\mathcal{R}}$ denotes the restriction of $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ that prohibits rewrite steps at the root position. By the induction hypothesis, we know that every innermost rewrite sequence that starts with q_i for some $1 \leq i \leq k$ is finite. If there exists an infinite rewrite sequence $t_0 \xrightarrow{\neg\varepsilon\mathsf{i}}_{\mathcal{R}} t_1 \xrightarrow{\neg\varepsilon\mathsf{i}}_{\mathcal{R}} t_2 \xrightarrow{\neg\varepsilon\mathsf{i}}_{\mathcal{R}} \ldots$, then we have to rewrite infinitely often below a position $1 \leq j \leq k$. But this would mean that we can find an infinite innermost rewrite sequence starting with q_j, which is not possible. Therefore, every rewrite sequence $t_0 \xrightarrow{\neg\varepsilon\mathsf{i}}_{\mathcal{R}} t_1 \xrightarrow{\neg\varepsilon\mathsf{i}}_{\mathcal{R}} t_2 \xrightarrow{\neg\varepsilon\mathsf{i}}_{\mathcal{R}} \ldots$ must be finite.

Next, we show that also every innermost rewrite sequence $t_0 \xrightarrow{\mathsf{i}}_{\mathcal{R}} t_1 \xrightarrow{\mathsf{i}}_{\mathcal{R}} t_2 \xrightarrow{\mathsf{i}}_{\mathcal{R}} \ldots$ is finite. By our previous paragraph, we know that there is some $N \in \mathbb{N}$ such that $t_0 \xrightarrow{\mathsf{i}}_{\mathcal{R}} \ldots \xrightarrow{\mathsf{i}}_{\mathcal{R}} t_N$ and t_N is in normal form w.r.t. $\xrightarrow{\neg\varepsilon\mathsf{i}}_{\mathcal{R}}$. Now, t_N is either also in normal form w.r.t. $\mathcal{R}$, which means that the whole innermost rewrite sequence is finite. Or we have $t_N = \ell\sigma$ for some rule $\ell \to r \in \mathcal{R}$ and some substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ such that every proper subterm of $\ell\sigma$ is in normal form w.r.t. $\mathcal{R}$. By our assumption, we then know that an innermost rewrite sequence that starts with t_N is finite, and this means that our whole rewrite sequence

$$t_0 \xrightarrow{\mathsf{i}}_{\mathcal{R}} \ldots \xrightarrow{\mathsf{i}}_{\mathcal{R}} t_N \xrightarrow{\mathsf{i}}_{\mathcal{R}} \ldots$$

must be finite. ■

The second theorem states that it suffices to only regard sequences of ground terms. This does not generally hold, as shown in Example 2.1.24, but we can extend the signature Σ with a fresh constant $\bot$ and a fresh unary function symbol h so that innermost termination of $\mathcal{R}$ over the signature Σ is equivalent to *ground innermost termination* of $\mathcal{R}$ over the signature $\Sigma \uplus \{\bot, \mathsf{h}\}$.

Example 2.1.24 (Innermost Ground Termination, [12])**.** Consider the TRS $\mathcal{R}$ consisting of the following two rewrite rules over the signature $\Sigma = \Sigma_0 \uplus \Sigma_1$ with $\Sigma_0 = \{\mathsf{a}\}$ and $\Sigma_1 = \{\mathsf{f}\}$:

$$\mathsf{f}(\mathsf{f}(x)) \to \mathsf{f}(\mathsf{f}(x)) \tag{2.5}$$
$$\mathsf{f}(\mathsf{a}) \to \mathsf{a} \tag{2.6}$$

Here, $\mathcal{R}$ is ground innermost terminating as the only possible ground terms have the form $\mathsf{f}^n(\mathsf{a})$ and thus, we have to rewrite the proper subterm $\mathsf{f}(\mathsf{a})$ before we can actually use rule (2.5). In comparison to that, $\mathcal{R}$ is not innermost terminating as we have the infinite innermost rewrite sequence $\mathsf{f}(\mathsf{f}(x)) \xrightarrow{\mathsf{i}}_{\mathcal{R}} \mathsf{f}(\mathsf{f}(x)) \xrightarrow{\mathsf{i}}_{\mathcal{R}} \dots$ that will never reach a normal form.

Theorem 2.1.25 (Ground Terms Suffice for TRSs, [12])**.** *Let $\mathcal{R}$ be a TRS over the signature Σ. Let $\bot$ be a fresh constant, and h be a fresh unary function symbol. Then $\mathcal{R}$ is innermost terminating over the signature Σ iff $\mathcal{R}$ is ground innermost terminating over the signature $\Sigma \uplus \{\bot, \mathsf{h}\}$. We say that a TRS is ground innermost terminating iff every innermost rewrite sequence of ground terms is finite.*

Proof. The theorem follows directly from the following equivalence. Let $t, t' \in \mathcal{T}(\Sigma, \mathcal{V})$ where the term t has the variables $x_1, \dots, x_m$. Then we have $t \xrightarrow{\mathsf{i}}_{\mathcal{R}} t'$ iff $t\sigma \xrightarrow{\mathsf{i}}_{\mathcal{R}} t'\sigma$, where $\sigma(x_i) = \mathsf{h}^i(\bot)$ for all $1 \leq i \leq m$ (i.e., both $t\sigma$ and $t'\sigma$ are ground terms). The two new fresh function symbols are needed to create arbitrarily many different ground terms in order to handle non-linear rewrite rules. By induction, we can then prove that a term t with variables $x_1, \dots, x_m$ starts an infinite innermost rewrite sequence iff the ground term $t\sigma$ does. ■

For the rest of this thesis, we will always assume that we have two such fresh symbols that do not occur in the rules of the TRS.

2.2 Direct Application of Polynomial Interpretations

We end this chapter with a first automatic approach to prove termination of a given TRS. Up to now, we are only working with syntactical objects (terms) that have no real *meaning*. We use Σ-*algebras* to give them a semantic.

Definition 2.2.1 (Σ-Interpretation, Σ-Algebra)**.** Let Σ be a signature, and $\mathcal{V}$ be a set of variables. A Σ-*interpretation* is a triple $\mathfrak{A} = (A, \alpha, \beta)$. Here, A is a non-empty set, called the carrier set of $\mathfrak{A}$. Moreover, we have $\alpha = (\alpha_f)_{f \in \Sigma}$ with $\alpha_f : A^n \to A$ for $f \in \Sigma_n$. The function α_f is the meaning of the function symbol f under the interpretation $\mathfrak{A}$. Lastly, $\beta : \mathcal{V} \to A$ is the variable assignment. For every Σ-interpretation $\mathfrak{A}$ we get a function $\mathfrak{A} : \mathcal{T}(\Sigma, \mathcal{V}) \to A$ by:

$$\begin{aligned} \mathfrak{A}(x) &= \beta(x) \text{ for all } x \in \mathcal{V} \\ \mathfrak{A}(f(t_1, \dots, t_n)) &= \alpha_f(\mathfrak{A}(t_1), \dots, \mathfrak{A}(t_n)) \text{ for all } f \in \Sigma_n, t_1, \dots, t_n \in \mathcal{T}(\Sigma, \mathcal{V}) \end{aligned}$$

We call $\mathfrak{A}(t)$ the interpretation of the term t under $\mathfrak{A}$. A Σ-interpretation without variable assignment (A, α) is called a Σ-*algebra*.

Termination only regards the syntactical rewrite relation of a TRS. Thus, we do not need to talk about the semantics of a term. However, using specific Σ-algebras is sometimes helpful to prove termination. One important type of such algebras is a *polynomial interpretation*. Polynomial interpretations are a classical technique to prove termination of TRSs [23, 24].

Definition 2.2.2 (Polynomial Interpretation)**.** A *polynomial interpretation* Pol is a Σ-algebra with carrier set $\mathbb{R}_{\geq 0}$ which maps every function symbol $f \in \Sigma$ to a polynomial $f_{Pol} \in \mathbb{R}_{\geq 0}[\mathcal{V}]$. It is *monotonic* if $x > y$ implies $f_{Pol}(\ldots, x, \ldots) > f_{Pol}(\ldots, y, \ldots)$ for all $f \in \Sigma$. It is *multilinear* if every f_{Pol} is of the following form with $c_V \in \mathbb{R}_{\geq 0}$ for all $V \subseteq \{x_1, \ldots, x_n\}$:

$$f_{Pol}(x_1, \ldots, x_n) = \sum_{V \subseteq \{x_1, \ldots, x_n\}} c_V \cdot \prod_{x \in V} x$$

Again, for $t \in \mathcal{T}(\Sigma, \mathcal{V})$, we write $Pol(t)$ for the interpretation of t under the Σ-algebra Pol. For an inequality containing variables, we say that it holds iff it is true for all instantiations of the variables. So $Pol(s) > Pol(t)$ holds for two terms $s, t \in \mathcal{T}(\Sigma, \mathcal{V})$, if this inequality is true for all instantiations of the occurring variables by numbers from $\mathbb{R}_{\geq 0}$.

To prove termination automatically, one has to find a mapping to monotonic polynomials with natural numbers as coefficients such that for every rewrite rule, the left-hand side is strictly greater than the right-hand side. This can then be automatized by searching for the coefficients using an SMT solver. Note that this approach can also be used to prove termination w.r.t. an arbitrary evaluation strategy.

Theorem 2.2.3 (Proving Innermost Termination Using Polynomial Interpretations)**.** *Let $\mathcal{R}$ be a TRS, and let $Pol : \mathcal{T}(\Sigma, \mathcal{V}) \to \mathbb{N}[\mathcal{V}]$ be a natural polynomial interpretation which is monotonic. Suppose that the following condition is satisfied:*

(1) *For every $\ell \to r \in \mathcal{R}$ we have $Pol(\ell) > Pol(r)$.*

Then $\mathcal{R}$ is innermost terminating.

Proof. By Theorem 2.1.25, we only have to regard ground terms. We start off by showing that the condition (1) of the theorem extends to rewrite steps instead of just rules:

(a) If $s, t \in \mathcal{T}(\Sigma)$ with $s \xrightarrow{\mathsf{i}}_{\mathcal{R}} t$, then we have $Pol(s) > Pol(t)$.

So let $s, t \in \mathcal{T}(\Sigma)$ with $s \xrightarrow{\mathsf{i}}_{\mathcal{R}} t$. Then there exist a rule $\ell \to r \in \mathcal{R}$ with $Pol(\ell) > Pol(r)$ (by (1)), a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and a position π such that $s|_\pi = \ell\sigma$, $t = s[r\sigma]_\pi$, and every proper subterm of $\ell\sigma$ is in normal form w.r.t. $\mathcal{R}$.

We perform induction on π. So in the induction base, let $\pi = \varepsilon$. Hence, we have $s = \ell\sigma$ and $t = r\sigma$. By assumption we have $Pol(\ell) > Pol(r)$. As these inequations hold for all instantiations of the occurring variables, we have

$$Pol(s) = Pol(\ell\sigma) > Pol(r\sigma) = Pol(t).$$

In the induction step, we have $\pi = i.\pi'$, $s = f(s_1, \ldots, s_i, \ldots, s_n)$ and $s_i \xrightarrow{\mathsf{i}}_{\mathcal{R}} t_i$. Then by the induction hypothesis we have $Pol(s_i) > Pol(t_i)$. For $t = f(s_1, \ldots, t_i, \ldots, s_n)$ we obtain

$$\begin{aligned}
Pol(s) &= Pol(f(s_1, \ldots, s_i, \ldots, s_n)) \\
&= f_{Pol}(Pol(s_1), \ldots, Pol(s_i), \ldots, Pol(s_n)) \\
&> f_{Pol}(Pol(s_1), \ldots, Pol(t_i), \ldots, Pol(s_n)) \\
&\qquad \text{(by monotonicity of } f_{Pol} \text{ and } Pol(s_i) > Pol(t_i)) \\
&= Pol(f(s_1, \ldots, t_i, \ldots, s_n)) \\
&= Pol(t).
\end{aligned}$$

Now, assume that $\mathcal{R}$ is not innermost terminating. Then there exists an infinite innermost rewrite sequence

$$t_0 \xrightarrow{\mathsf{i}}_{\mathcal{R}} t_1 \xrightarrow{\mathsf{i}}_{\mathcal{R}} t_2 \xrightarrow{\mathsf{i}}_{\mathcal{R}} \dots$$

But this would mean that

$$Pol(t_0) > Pol(t_1) > Pol(t_2) > \dots$$

is an infinite descending sequence of natural numbers since all the occurring terms are ground terms, which is a contradiction. Hence, $\mathcal{R}$ must be innermost terminating. ■

Example 2.2.4. Consider the TRS $\mathcal{R}_{plus}$ from Example 2.1.16. A polynomial interpretation that satisfies the condition of Theorem 2.2.3 for every rule of $\mathcal{R}_{plus}$ maps $\mathsf{s}(x)$ to $x+1$, $\mathsf{plus}(x,y)$ to $2x+y+1$, and $\mathcal{O}$ to 0. Now, we have

$$\begin{array}{lrcl} & Pol(\mathsf{plus}(\mathcal{O},y)) & > & Pol(y) \\ \Leftrightarrow & 2\cdot 0+y+1 & > & y \\ \Leftrightarrow & y+1 & > & y \end{array}$$

and

$$\begin{array}{lrcl} & Pol(\mathsf{plus}(\mathsf{s}(x),y)) & > & Pol(\mathsf{s}(\mathsf{plus}(x,y))) \\ \Leftrightarrow & 2\cdot(x+1)+y+1 & > & (2x+y+1)+1 \\ \Leftrightarrow & 2x+y+3 & > & 2x+y+2 \end{array}$$

We also give another example that shows how to deal with data structures in TRSs and how to prove innermost termination for a TRS that contains data structures.

Example 2.2.5 (Termination with Data Structures)**.** Consider the signature $\Sigma = \{\mathcal{O}, \mathsf{s}, \mathsf{len}, \mathsf{cons}, \mathsf{nil}\}$, a variable set with $\{x,y\} \subseteq \mathcal{V}$, and the following TRS $\mathcal{R}_{len}$ that computes the length of a given list:

$$\mathsf{len}(\mathsf{nil}) \to \mathcal{O} \tag{2.7}$$

$$\mathsf{len}(\mathsf{cons}(x,y)) \to \mathsf{s}(\mathsf{len}(y)) \tag{2.8}$$

Here, lists are represented by the two constructor symbols $\mathsf{nil} \in \Sigma_0$ for the empty list and $\mathsf{cons} \in \Sigma_2$, which prepends an element in front of another list. We can prove termination with the polynomial interpretation that maps $\mathsf{len}(x)$ to x, $\mathsf{s}(x)$ to x, $\mathsf{cons}(x,y)$ to $x+y+1$, and both nil and $\mathcal{O}$ to 0.

3 DP Framework for TRS

We are interested in the automatic termination analysis of TRS and have already introduced a way to do this using polynomial interpretations in the previous chapter. Generally, one can try to automatically find a well-founded order $\succ$ such that the rewrite relation $\to_{\mathcal{R}}$ implies this order $\succ$ (i.e., $s \to_{\mathcal{R}} t$ implies $s \succ t$). A relation is well-founded iff there is no infinite descending sequence $t_0 \succ t_1 \succ t_2 \succ \ldots$, which is exactly our desired property. If we can find such a well-founded order automatically, then we have automatically proven that our TRS is terminating [26]. However, there are relatively simple examples where we cannot prove termination automatically using a direct application of polynomial interpretations. This also includes a lot of TRSs that occur in real-world applications. Another way to prove termination automatically is to use dependency pairs. The basic idea of dependency pairs is that a recursive function terminates whenever its arguments "decrease" in every iteration. Currently, this is one of the best approaches to prove termination in practice.

Example 3.0.1 (TRS which Requires Dependency Pairs)**.** Consider the signature $\Sigma_{div} = \{\mathcal{O}, \mathsf{s}, \mathsf{minus}, \mathsf{div}\}$, a variable set with $\{x, y\} \subseteq \mathcal{V}$, and the TRS $\mathcal{R}_{div}$ consisting of the following rewrite rules that computes the integer division of two natural numbers:

$$\mathsf{minus}(x, \mathcal{O}) \to x \tag{3.1}$$
$$\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{minus}(x, y) \tag{3.2}$$
$$\mathsf{div}(\mathcal{O}, \mathsf{s}(y)) \to \mathcal{O} \tag{3.3}$$
$$\mathsf{div}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{s}\,(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y))) \tag{3.4}$$

Here, we cannot prove its termination using a direct application of polynomial interpretations due to rule (3.4). It is generally hard to use polynomial interpretations for termination proofs if the right-hand side of a rule is syntactically larger than the left-hand side. Let us assume for a contradiction that there exists a monotonic polynomial interpretation Pol such that $Pol(\ell) > Pol(r)$ for all rules $\ell \to r \in \mathcal{R}_{div}$. Then, we also have

$$Pol\Big(\mathsf{div}(\mathsf{s}(x), \mathsf{s}(y))\Big) > Pol\Big(\mathsf{s}\,(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y)))\Big)$$

Since Pol is monotonic, we must have $Pol(\mathsf{s}(x)) \geq x$ for all instantiations of x, and hence

$$Pol\Big(\mathsf{div}(\mathsf{s}(x), \mathsf{s}(y))\Big) > Pol\Big(\mathsf{s}\,(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y)))\Big) \geq Pol\Big(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y))\Big)$$

Now, this inequality needs to hold for all instantiations of the variables, so also if we instantiate y with $\mathsf{s}(x)$. In this case, we get

$$Pol\Big(\mathsf{div}(\mathsf{s}(x), \mathsf{s}(\mathsf{s}(x)))\Big) > Pol\Big(\mathsf{div}(\mathsf{minus}(x, \mathsf{s}(x)), \mathsf{s}(\mathsf{s}(x)))\Big)$$

We have $Pol(\mathsf{minus}(x, \mathsf{s}(x))) \geq Pol(\mathsf{s}(x))$ and thus by the monotonicity of Pol we get

$$Pol\Big(\mathsf{div}(\mathsf{minus}(x, \mathsf{s}(x)), \mathsf{s}(\mathsf{s}(x)))\Big) \geq Pol\Big(\mathsf{div}(\mathsf{s}(x), \mathsf{s}(\mathsf{s}(x)))\Big)$$

But this means that

$$Pol\Big(\mathsf{div}(\mathsf{s}(x),\mathsf{s}(\mathsf{s}(x)))\Big) > Pol\Big(\mathsf{div}(\mathsf{s}(x),\mathsf{s}(\mathsf{s}(x)))\Big)$$

which is a contradiction.

In the rest of this chapter, we introduce the dependency pair framework, how it works, and its main components. We start by precisely defining the notion of *dependency pairs*, *DP problems* and *chains* in Section 3.1. Then, in Section 3.2 we prove the main connection between a TRS $\mathcal{R}$ and its DP problem $(\mathcal{DP}(\mathcal{R}),\mathcal{R})$, which is the so called *chain criterion*. We explain the core idea behind the framework that is based on divide-and-conquer in Section 3.3. It recursively creates smaller subproblems using so-called *DP processors* until we solve all of them. Ultimately, we introduce the most important processors for this framework and show a final example of how the processors and the framework work in total.

3.1 Dependency Pairs and Chains

In addition to the rewrite rules of a TRS, we introduce a second type of rewrite rule, namely *dependency pairs*. First, we divide the signature into *defined symbols* and *constructor symbols*, and then extend it with a new set of *tuple symbols*. A TRS's defined symbols represent functions that can be evaluated, the constructor symbols represent the used data structures, and the tuple symbols represent a particular function call of some defined symbol. The tuple symbols are used to compare the arguments of two different function calls.

Definition 3.1.1 (Defined Symbols)**.** Let $\mathcal{R}$ be a TRS over the signature Σ. We decompose $\Sigma = \Sigma_C \uplus \Sigma_D$ such that $f \in \Sigma_D$ iff $f(\ell_1,\ldots,\ell_n) \to r \in \mathcal{R}$ for some terms $\ell_1,\ldots,\ell_n,r \in \mathcal{T}(\Sigma,\mathcal{V})$. The symbols $f \in \Sigma_C$ are called *constructors* and the symbols $f \in \Sigma_D$ are called *defined symbols*. For every defined symbol, we introduce a fresh *tuple symbol* $f^\#$ of the same arity. The set of all tuple symbols is defined as $\Sigma^\#$. For any term $r = f(r_1,\ldots,r_n)$ with defined root symbol f, let $r^\#$ denote the term $f^\#(r_1,\ldots,r_n)$. For every term $r = f^\#(t_1,\ldots,t_k)$ with tuple root symbol $f^\#$, let $r^\flat$ denote the term $f(t_1,\ldots,t_k)$.

Example 3.1.2 (Defined Symbols)**.** For the signature Σ_{div} and the TRS $\mathcal{R}_{div}$ from Example 3.0.1, we get $\Sigma_D = \{\mathsf{minus},\mathsf{div}\}$, $\Sigma_C = \{\mathsf{s},\mathcal{O}\}$ and thus

$$\Sigma \uplus \Sigma^\# = \{\mathsf{minus}^\#,\mathsf{div}^\#,\mathsf{minus},\mathsf{div},\mathsf{s},\mathcal{O}\}$$

To prove termination of a TRS, we have to compare the arguments of the left-hand side with the arguments of every defined symbol on the right-hand side for every rewrite rule in $\mathcal{R}$. To do so, we use dependency pairs.

Definition 3.1.3 (Dependency Pair, [14])**.** Let $\mathcal{R}$ be a TRS. If $\ell \to r \in \mathcal{R}$ and $t \trianglelefteq r$ is a subterm with defined root symbol, then $\ell^\# \to t^\#$ is a dependency pair of $\mathcal{R}$. The set of all dependency pairs is denoted by $\mathcal{DP}(\mathcal{R})$.

Example 3.1.4 (Dependency Pairs for $\mathcal{R}_{div}$)**.** For the TRS $\mathcal{R}_{div}$ from Example 3.0.1, we get the following three dependency pairs

$$\mathsf{minus}^\#(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{minus}^\#(x,y) \tag{3.5}$$

$$\mathsf{div}^\#(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{minus}^\#(x,y) \tag{3.6}$$

$$\mathsf{div}^{\#}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{div}^{\#}(\mathsf{minus}(x,y),\mathsf{s}(y)) \tag{3.7}$$

The first dependency pair results from rule (3.2). The other two results from the two different defined symbols in the right-hand side of (3.4).

Generally speaking, a dependency pair is a rewrite rule of the form $\ell^{\#} \to r^{\#}$ with $\ell, r \in \mathcal{T}(\Sigma, \mathcal{V})$. Hence, a set of dependency pairs $\mathcal{D}$ also induces a rewrite relation $\to_{\mathcal{D}} \subseteq \mathcal{T}\left(\Sigma \uplus \Sigma^{\#}, \mathcal{V}\right) \times \mathcal{T}\left(\Sigma \uplus \Sigma^{\#}, \mathcal{V}\right)$, just like a TRS via

$$s \to_{\mathcal{D}} t \qquad :\Leftrightarrow \qquad s|_{\pi} = \ell^{\#}\sigma \text{ and } t = s[r^{\#}\sigma]_{\pi},$$

for a position $\pi \in Occ(s)$, a dependency pair $\ell^{\#} \to r^{\#} \in \mathcal{D}$, and a substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$.

Now, we have two different types of objects that we can use for our termination analysis: dependency pairs and rewrite rules. We will work with both of them simultaneously. To be precise, we have a pair $(\mathcal{D}, \mathcal{R})$ consisting of a set of dependency pairs $\mathcal{D}$ and a TRS $\mathcal{R}$. Such a pair will be called a *DP problem*. We also have to define the notion of rewrite sequences we are interested in when dealing with DP problems. For a TRS $\mathcal{R}$ on its own, we looked at $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$-rewrite sequences. Now, we want to work with $(\mathcal{D}, \mathcal{R})$-*chains*. As we are dealing with innermost evaluation, we will work with innermost $(\mathcal{D}, \mathcal{R})$-chains as well. These chains work with a specific type of terms, namely *dependency terms*.

Definition 3.1.5 (Dependency Term)**.** Let $\Sigma = \Sigma_D \uplus \Sigma_C$ be a signature divided into defined and constructor symbols. By $\mathcal{T}^{\#}(\Sigma)$, we denote the set of all ground terms $t \in \mathcal{T}(\Sigma \uplus \Sigma^{\#})$ of the form

$$t = f^{\#}(t_1, \ldots, t_n)$$

such that $t_i \in \mathcal{T}(\Sigma)$ for all $1 \leq i \leq n$ and $f \in \Sigma_D$. A term $t \in \mathcal{T}^{\#}(\Sigma)$ is called a *dependency term*.

Note that if we rewrite a dependency term with a dependency pair or a rewrite rule, we again result in a dependency term.

Definition 3.1.6 (Innermost Chain, [14])**.** Let $(\mathcal{D}, \mathcal{R})$ be a DP problem. A (possibly infinite) sequence $t_0, t_1, t_2, \ldots$ of dependency terms (i.e., $t_i \in \mathcal{T}^{\#}(\Sigma)$ for all i) is an innermost $(\mathcal{D}, \mathcal{R})$-chain iff we have $t_i \xrightarrow{\mathsf{i}}_{\mathcal{D},\mathcal{R}} \circ \xrightarrow{\mathsf{i}}_{\mathcal{R}}^{*} t_{i+1}$ for all i. Here, $\xrightarrow{\mathsf{i}}_{\mathcal{D},\mathcal{R}}$ denotes the restriction of $\to_{\mathcal{D}}$ to rewrite steps such that every proper subterm of the used redex is in normal form w.r.t. $\mathcal{R}$. For a relation E over some set A, E^{*} denotes the transitive, reflexive closure and the composition of two relations E_1, E_2 over some set A is defined by

$$E_1 \circ E_2 = \{(x, z) \in A \times A \mid \exists y \in A : (x, y) \in E_1 \wedge (y, z) \in E_2\}$$

We will also write $\xrightarrow{\mathsf{i}}_{(\mathcal{D},\mathcal{R})}$ instead of $\xrightarrow{\mathsf{i}}_{\mathcal{D},\mathcal{R}} \circ \xrightarrow{\mathsf{i}}_{\mathcal{R}}^{*}$. A DP problem $(\mathcal{D}, \mathcal{R})$ is innermost terminating iff there is no infinite innermost $(\mathcal{D}, \mathcal{R})$-chain.

A chain can be seen as a sequence of function calls, where we use the TRS to evaluate their arguments. In this context, a dependency term is a function call that indicates the used function by its tuple symbol and stores the corresponding arguments. A dependency pair is a rewrite step from one function call to another, and the TRS is used to evaluate their arguments. As you can see, the dependency pairs are the dominating part of our definition of a chain. We have to use at least one dependency pair and then an arbitrary number of TRS rules to get to the next term in our sequence.

Example 3.1.7 (Innermost Chain)**.** Consider $\mathcal{R}_{div}$ and $\mathcal{DP}(\mathcal{R}_{div})$ from Example 3.0.1 and Example 3.1.4. An example for an innermost $(\mathcal{DP}(\mathcal{R}_{div}), \mathcal{R}_{div})$-chain would be

$$\mathsf{div}^{\#}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathcal{O})))), \mathsf{s}(\mathsf{s}(\mathcal{O}))) \tag{3.8}$$
$$\xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} \mathsf{div}^{\#}(\mathsf{s}(\mathsf{s}(\mathcal{O})), \mathsf{s}(\mathcal{O})) \tag{3.9}$$
$$\xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} \mathsf{minus}^{\#}(\mathsf{s}(\mathcal{O}), \mathsf{s}(\mathsf{s}(\mathcal{O}))) \tag{3.10}$$
$$\xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} \mathsf{minus}^{\#}(\mathcal{O}, \mathcal{O}) \tag{3.11}$$

For the first step, we use the dependency pair (3.7) to get

$$\mathsf{div}^{\#}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathcal{O})))), \mathsf{s}(\mathsf{s}(\mathcal{O}))) \xrightarrow{\mathrm{i}}_{\mathcal{D},\mathcal{R}} \mathsf{div}^{\#}(\mathsf{minus}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathcal{O})))), \mathsf{s}(\mathcal{O})), \mathsf{s}(\mathsf{s}(\mathcal{O})))$$

Then, we can use the TRS rules (3.1) and (3.2) to completely evaluate the subterm $\mathsf{minus}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathcal{O})))), \mathsf{s}(\mathcal{O}))$ as we have

$$\mathsf{minus}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathcal{O})))), \mathsf{s}(\mathcal{O})) \xrightarrow{\mathrm{i}}{}^{*}_{\mathcal{R}} \mathsf{s}(\mathsf{s}(\mathcal{O}))$$

and finally result with $\mathsf{div}^{\#}(\mathsf{s}(\mathsf{s}(\mathcal{O})), \mathsf{s}(\mathsf{s}(\mathcal{O})))$. For the second step, we used the dependency pair (3.6) and then no rewrite rule. Finally, we used the dependency pair (3.5) for our last step, and again no rewrite rules. The final term $\mathsf{minus}^{\#}(\mathcal{O}, \mathcal{O})$ is in normal form w.r.t. $\xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})}$, and thus we cannot extend the chain any further.

Note, we can only apply a $\xrightarrow{\mathrm{i}}_{\mathcal{D},\mathcal{R}}$ step at the root position by definition of a dependency term. Additionally, we are only working with ground terms and only allow rewrite steps if every proper subterm is in normal form w.r.t. $\mathcal{R}$. Hence, in an innermost $(\mathcal{D}, \mathcal{R})$-chain $t_0 \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} t_1 \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} t_2 \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} \dots$ for all i we have $t_i = \ell_i^{\#}\sigma_i$ for some dependency pair $\ell_i^{\#} \to r_i^{\#} \in \mathcal{D}$ that we use in the i-th rewrite step, some ground substitution $\sigma_i \in \mathrm{Sub}(\Sigma, \mathcal{V})$ and such that every proper subterm of $\ell_i^{\#}\sigma_i$ is in normal form w.r.t. $\mathcal{R}$. Chains can also be represented as a sequence of dependency pairs $s_0 \to t_0, s_1 \to t_1, s_2 \to t_2, \dots$ such that there is a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ with $t_i\sigma \xrightarrow{\mathrm{i}}{}^{*}_{\mathcal{R}} s_{i+1}\sigma$ for all i and every proper subterm of $s_i\sigma$ is in normal form w.r.t. $\mathcal{R}$. Here, we have to assume that the dependency pairs have renamed variables so that for $i \neq j$, the two dependency pairs $s_i \to t_i$ and $s_j \to t_j$ do not share a variable. Note that most of the literature will define $(\mathcal{D}, \mathcal{R})$-chains in this fashion and not in the more general way we do in this thesis. However, this more strict definition cannot be generalized to the probabilistic setting, as we will see later on. Instead, we will define a completely new type of dependency pair and a new type of chain based on trees in Chapter 5 that will still share this general chain structure.

3.2 Chain Criterion

In this section, we want to establish a connection between the TRS $\mathcal{R}$ and the DP problem $(\mathcal{DP}(\mathcal{R}), \mathcal{R})$. This connection is given by the following *chain criterion*. The core idea behind the chain criterion is that there exists an infinite innermost rewrite sequence for a TRS $\mathcal{R}$ iff there is an infinite innermost sequence of function calls, i.e., an infinite innermost $(\mathcal{DP}(\mathcal{R}), \mathcal{R})$-chain.

Theorem 3.2.1 (Chain Criterion, [1])**.**
A TRS $\mathcal{R}$ is innermost terminating iff the DP problem $(\mathcal{DP}(\mathcal{R}), \mathcal{R})$ is innermost terminating.

Proof.

complete: Assume that $(\mathcal{DP}(\mathcal{R}), \mathcal{R})$ is not innermost terminating. Then there exists an infinite innermost $(\mathcal{DP}(\mathcal{R}), \mathcal{R})$-chain

$$t_0^\#(\overline{v}_0) \xrightarrow{\text{i}}_{(\mathcal{DP}(\mathcal{R}),\mathcal{R})} t_1^\#(\overline{v}_1) \xrightarrow{\text{i}}_{(\mathcal{DP}(\mathcal{R}),\mathcal{R})} t_2^\#(\overline{v}_2) \xrightarrow{\text{i}}_{(\mathcal{DP}(\mathcal{R}),\mathcal{R})} \dots$$

with $\overline{v}_0$ being normal forms w.r.t. $\mathcal{R}$ and not containing any tuple symbols. Remember that this means $t_i^\#(\overline{v}_i) \xrightarrow{\text{i}}_{\mathcal{D},\mathcal{R}} \circ \xrightarrow{\text{i}}_{\mathcal{R}}^* t_{i+1}^\#(\overline{v}_{i+1})$ for all $i \in \mathbb{N}$. To be precise, we have

$$t_0^\#(\overline{v}_0) \xrightarrow{\text{i}}_{\mathcal{D},\mathcal{R}} t_1^\#(\overline{w}_1) \xrightarrow{\text{i}}_{\mathcal{R}}^* t_1^\#(\overline{v}_1) \xrightarrow{\text{i}}_{\mathcal{D},\mathcal{R}} t_2^\#(\overline{w}_2) \xrightarrow{\text{i}}_{\mathcal{R}}^* t_2^\#(\overline{v}_2) \xrightarrow{\text{i}}_{\mathcal{D},\mathcal{R}} \dots$$

Note that a dependency pair $f^\#(\overline{v}) \to g^\#(\overline{w})$ corresponds to a rewrite rule $f(\overline{v}) \to \mathcal{C}[g(\overline{w})] \in \mathcal{R}$ for some Context $\mathcal{C}$. Therefore, our infinite innermost $(\mathcal{DP}(\mathcal{R}), \mathcal{R})$-chain corresponds to the following infinite innermost rewrite sequence

$$t_0(\overline{v}_0) \xrightarrow{\text{i}}_{\mathcal{R}} \mathcal{C}_1[t_1(\overline{w}_1)] \xrightarrow{\text{i}}_{\mathcal{R}}^* \mathcal{C}_1[t_1(\overline{v}_1)] \xrightarrow{\text{i}}_{\mathcal{R}} \mathcal{C}_1[\mathcal{C}_2[t_2(\overline{w}_2)]] \xrightarrow{\text{i}}_{\mathcal{R}}^* \mathcal{C}_1[\mathcal{C}_2[t_2(\overline{v}_2)]] \xrightarrow{\text{i}}_{\mathcal{R}} \dots$$

Therefore, $\mathcal{R}$ is not innermost terminating as well.

sound: Assume that $\mathcal{R}$ is not innermost terminating. Then there exists an infinite innermost rewrite sequence

$$t_0 \xrightarrow{\text{i}}_{\mathcal{R}} t_1 \xrightarrow{\text{i}}_{\mathcal{R}} t_2 \xrightarrow{\text{i}}_{\mathcal{R}} \dots$$

By Theorem 2.1.25, we can assume that all occurring terms are ground terms. We will now inductively define an infinite innermost $(\mathcal{DP}(\mathcal{R}), \mathcal{R})$-chain

$$t_0' \xrightarrow{\text{i}}_{(\mathcal{DP}(\mathcal{R}),\mathcal{R})} t_1' \xrightarrow{\text{i}}_{(\mathcal{DP}(\mathcal{R}),\mathcal{R})} t_2' \xrightarrow{\text{i}}_{(\mathcal{DP}(\mathcal{R}),\mathcal{R})} \dots$$

Due to a minimality argument, we can find a subterm $f_0(\overline{v}_0)$ of t_0 such that

- $f_0(\overline{v}_0)$ starts an infinite innermost rewrite sequence
- $\overline{v}_0$ are strongly normalizing terms

We can rewrite the terms $\overline{v}_0$ using $\xrightarrow{\text{i}}_{\mathcal{R}}$ steps until we reach a normal form for all of them as we would do in our infinite innermost rewrite sequence. Since all of them are strongly normalizing, this can be done in a finite number of $\xrightarrow{\text{i}}_{\mathcal{R}}$ steps.

After this, we result with a term $f_0(\overline{w}_0)$, where all of the $\overline{w}_0$ are in normal form w.r.t. $\mathcal{R}$ and thus we can rewrite with the whole term $f_0(\overline{w}_0)$ being the redex. Hence, we have $f_0(\overline{w}_0) = f_0(\overline{u}_0)\sigma_0 \xrightarrow{\text{i}}_{\mathcal{R}} r_0\sigma_0$ using a rule $f_0(\overline{u}_0) \to r_0 \in \mathcal{R}$ and ground substitution $\sigma_0 \in \text{Sub}(\Sigma, \mathcal{V})$. Since, $f_0(\overline{v}_0)$ starts an infinite rewrite sequence, the same holds for $f_0(\overline{w}_0)$. Additionally, we know that all $\overline{w}_0 = \overline{u}_0\sigma_0$ are in normal form w.r.t. $\mathcal{R}$. This, together with a minimality argument, again implies that we can find a subterm $f_1(\overline{v}_1)$ of r_0 such that

- $f_1(\overline{v}_1)\sigma_0$ starts an infinite innermost rewrite sequence
- $\overline{v}_1\sigma_0$ are strongly normalizing terms

Our $(\mathcal{DP}(\mathcal{R}), \mathcal{R})$-chain starts with the term $t_0' := f_0^\#(\overline{w}_0)$ and first uses the dependency pair $f_0^\#(\overline{u}_0) \to f_1^\#(\overline{v}_1) \in \mathcal{DP}(\mathcal{R})$.

Now, we can apply the same procedure to determine the whole chain. Assume that we have already defined the chain up to the i-th term. From our previous induction step, we get a term $f_{i+1}(\overline{v}_{i+1})\sigma_i$ that starts an infinite innermost rewrite sequence

and all of the $\overline{v}_{i+1}\sigma_i$ are strongly normalizing terms. Again, we can rewrite the terms $\overline{v}_{i+1}\sigma_i$ using $\xrightarrow{\mathrm{i}}_{\mathcal{R}}$ steps until we reach a normal form for all of them as we would do in our infinite innermost rewrite sequence. Since all of them are strongly normalizing, this can be done in a finite number of $\xrightarrow{\mathrm{i}}_{\mathcal{R}}$ steps.

After this, we result with a term $f_{i+1}(\overline{w}_{i+1})$, where all of the $\overline{w}_{i+1}$ are in normal form w.r.t. $\mathcal{R}$ and we can rewrite with the whole term $f_{i+1}(\overline{w}_{i+1})$ being the redex. Thus, we have $f_{i+1}(\overline{w}_{i+1}) = f_{i+1}(\overline{u}_{i+1})\sigma_{i+1} \xrightarrow{\mathrm{i}}_{\mathcal{R}} r_{i+1}\sigma_{i+1}$ using a rule $f_{i+1}(\overline{u}_{i+1}) \to r_{i+1} \in \mathcal{R}$ and ground substitution $\sigma_{i+1} \in \mathrm{Sub}(\Sigma, \mathcal{V})$. Since, $f_{i+1}(\overline{v}_{i+1})$ starts an infinite rewrite sequence, the same holds for $f_{i+1}(\overline{w}_{i+1})$. Additionally, we know that all $\overline{w}_{i+1} = \overline{u}_{i+1}\sigma_{i+1}$ are in normal form w.r.t. $\mathcal{R}$. This, together with a minimality argument, again implies that we can find a subterm $f_{i+2}(\overline{v}_{i+2})$ of r_{i+1} such that

- $f_{i+2}(\overline{v}_{i+2})\sigma_{i+1}$ starts an infinite innermost rewrite sequence
- $\overline{v}_{i+2}\sigma_{i+1}$ are strongly normalizing terms

Then, we set $t'_{i+1} \coloneqq f^{\#}_{i+1}(\overline{w}_{i+1})$.

We now constructed a sequence $t'_0, t'_1, t'_2, \ldots$ and by construction we have $t'_i = f^{\#}_i(\overline{w}_i) \xrightarrow{\mathrm{i}}_{\mathcal{D},\mathcal{R}} f^{\#}_{i+1}(\overline{v}_{i+1}\sigma_i) \xrightarrow{\mathrm{i}}{}^{*}_{\mathcal{R}} f^{\#}_{i+1}(\overline{w}_{i+1}) = t'_{i+1}$ for all $i \in \mathbb{N}$ so that

$$t'_0 \xrightarrow{\mathrm{i}}_{(\mathcal{DP}(\mathcal{R}),\mathcal{R})} t'_1 \xrightarrow{\mathrm{i}}_{(\mathcal{DP}(\mathcal{R}),\mathcal{R})} t'_2 \xrightarrow{\mathrm{i}}_{(\mathcal{DP}(\mathcal{R}),\mathcal{R})} \cdots$$

This is an infinite innermost $(\mathcal{DP}(\mathcal{R}), \mathcal{R})$-chain, which shows that $(\mathcal{DP}(\mathcal{R}), \mathcal{R})$ is also not innermost terminating. ■

3.3 DP Framework and Processors

Now that we understand the relation between a TRS $\mathcal{R}$ and the corresponding DP problem $(\mathcal{DP}(\mathcal{R}), \mathcal{R})$, how can we use this for automatizing termination proofs? The key idea is to use a *divide-and-conquer* framework that works with these DP problems. We want to recursively split a DP problem $(\mathcal{D}, \mathcal{R})$ into a set of simpler subproblems $\{(\mathcal{D}_1, \mathcal{R}_1), \ldots, (\mathcal{D}_k, \mathcal{R}_k)\}$ and then solve all of the subproblems individually. For such a framework, we really need to work with DP problems and not just TRSs or sets of dependency pairs since:

- Working solely with a TRS $\mathcal{R}$ is not powerful enough, and it is relatively hard to split a TRS into multiple subproblems so that we can deduce the termination of the main TRS from the termination of the subproblems.
- Working solely with a set of dependency pairs $\mathcal{D}$ is not expressive enough as we cannot represent every possible rewrite sequence with dependency pairs on their own.

In order to split a DP problem, we use *DP processors*.

Definition 3.3.1 (DP Processor, [14])**.** Let $\mathcal{D}, \mathcal{D}_1, \ldots, \mathcal{D}_n$ be sets of dependency pairs and let $\mathcal{R}, \mathcal{R}_1, \ldots, \mathcal{R}_n$ be TRSs. A *(innermost) DP processor* Proc is of the form

$$\mathrm{Proc}(\mathcal{D}, \mathcal{R}) = \{(\mathcal{D}_1, \mathcal{R}_1), \ldots, (\mathcal{D}_n, \mathcal{R}_n)\}$$

A DP processor Proc is *sound* iff

$$\forall\, 1 \leq i \leq n : (\mathcal{D}_i, \mathcal{R}_i) \text{ is innermost terminating } \implies (\mathcal{D}, \mathcal{R}) \text{ is innermost terminating}$$

A DP processor Proc is *complete* iff

$$(\mathcal{D},\mathcal{R}) \text{ is innermost terminating } \implies \forall\, 1 \leq i \leq n : (\mathcal{D}_i,\mathcal{R}_i) \text{ is innermost terminating}$$

To summarize it, given a TRS $\mathcal{R}$, we start by dividing the signature into defined and constructor symbols. Then we generate the corresponding DP problem $(\mathcal{DP}(\mathcal{R}),\mathcal{R})$ and try to prove that this DP problem is innermost terminating using our divide-and-conquer framework and the DP processors.

In the following, we will introduce three of the most important processors for the dependency pair framework. Starting with the *dependency graph processor*, where we create a kind of control flow graph of the DP problem to split the set of dependency pairs into subsets such that every two dependency pairs can rewrite to each other. Then, we introduce a processor that decreases the number of rewrite rules in our TRS. Here, we remove the rules from the TRS that cannot be used to evaluate the right-hand side of any dependency pair. This is called the *usable rules processor*. In the end, we have the *reduction pair processor* that reduces the set of dependency pairs by ordering them using a polynomial interpretation, similar to the previously introduced direct application of polynomial interpretations. Note that there are a lot more existing processors that we do not mention in this thesis.

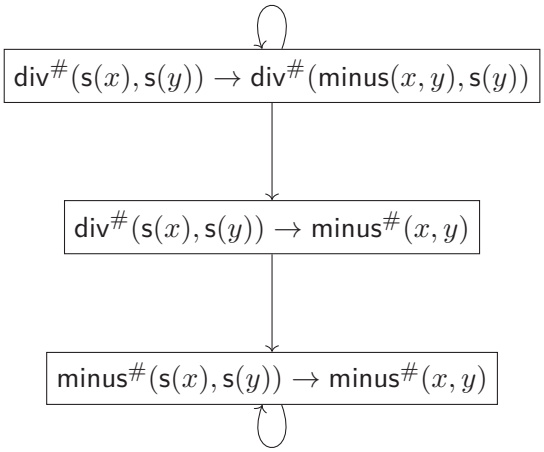
Dependency Graph Processor

The first processor that we want to introduce is the *dependency graph processor*. The *dependency graph* is a kind of control flow graph that shows which right-hand sides of dependency pairs can be rewritten to the left-hand side of another one. Or in other words, it shows which function calls can occur after each other.

Definition 3.3.2 (Dependency Graph, [14])**.** Let $(\mathcal{D},\mathcal{R})$ be a DP problem. The $(\mathcal{D},\mathcal{R})$-*dependency graph* is defined as the graph with node set $\mathcal{D}$ and there is an edge from $s \to t$ to $v \to w$ iff there exists ground substitutions $\sigma_1,\sigma_2 \in \mathrm{Sub}(\Sigma,\mathcal{V})$ such that $t\sigma_1 \xrightarrow{\mathsf{i}}{}^*_{\mathcal{R}} v\sigma_2$, and every proper subterm of $s\sigma_1$ and $v\sigma_2$ is in normal form w.r.t. $\mathcal{R}$.

Every innermost $(\mathcal{D},\mathcal{R})$-chain corresponds to a path through the $(\mathcal{D},\mathcal{R})$-dependency graph. On the other side, not every path through the dependency graph corresponds to a chain since we may use completely different substitutions on different edges in the path, and in a chain, the used substitution is carried over to the next rewrite step.

Example 3.3.3 ($(\mathcal{DP}(\mathcal{R}_{div}),\mathcal{R}_{div})$-Dependency Graph)**.** Let $\mathcal{R}_{div}$ be the PTRS from Example 3.0.1 and $\mathcal{DP}(\mathcal{R}_{div})$ be its dependency pairs from Example 3.1.4. The $(\mathcal{DP}(\mathcal{R}_{div}),\mathcal{R}_{div})$-dependency graph has the following form:



For the two edges regarding the two bottom nodes, we only have to use appropriate substitutions to get from the right-hand side of the one dependency pair to the left-hand side of the other. For the two edges at the top, we have to evaluate the subterm $\mathsf{minus}(x, y)$ using the rewrite rules from $\mathcal{R}_{div}$ so that we can get to the left-hand side of the two dependency pairs (with suitable substitutions).

The key idea for the processor is that we only need to look at the strongly connected components (SCCs) of the dependency graph in order to find an infinite innermost $(\mathcal{DP}(\mathcal{R}), \mathcal{R})$-chain. An infinite path in the dependency graph must eventually be stuck inside a single SCC due to the fact that we have only a finite number of dependency pairs, hence a finite dependency graph.

Theorem 3.3.4 (Dependency Graph Processor, [14])**.** *Let*

$$\mathrm{Proc}_{\mathtt{DG}}(\mathcal{D}, \mathcal{R}) = \{(\mathcal{D}_1, \mathcal{R}), \ldots, (\mathcal{D}_n, \mathcal{R})\}$$

where $\mathcal{D}_1, \ldots, \mathcal{D}_n$ are the SCCs of the $(\mathcal{D}, \mathcal{R})$-dependency graph. Then $\mathrm{Proc}_{\mathtt{DG}}$ *is sound and complete.*

Proof.

complete: Every innermost $(\mathcal{D}_i, \mathcal{R})$-chain is also an innermost $(\mathcal{D}, \mathcal{R})$-chain. Hence, if one of the $(\mathcal{D}_i, \mathcal{R})$ is not innermost terminating, then $(\mathcal{D}, \mathcal{R})$ is also not innermost terminating.

sound: Assume that $(\mathcal{D}, \mathcal{R})$ is not innermost terminating. Then there exists an infinite innermost $(\mathcal{D}, \mathcal{R})$-chain

$$t_0 \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} t_1 \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} t_2 \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} \ldots$$

such that for all $i \in \mathbb{N}$ we have $t_i = \ell_i^{\#}\sigma_i$ for some dependency pair $\ell_i^{\#} \to r_i^{\#} \in \mathcal{D}$ that we use in the i-th rewrite step, some ground substitution $\sigma_i \in \mathrm{Sub}(\Sigma, \mathcal{V})$ and such that every proper subterm of t_i is in normal form w.r.t. $\mathcal{R}$.

This chain corresponds to a unique path in the $(\mathcal{D}, \mathcal{R})$-dependency graph. To see this, let $i \in \mathbb{N}$. We use the dependency pair $\ell_i^{\#} \to r_i^{\#} \in \mathcal{D}$ in the i-th and the dependency pair $\ell_{i+1}^{\#} \to r_{i+1}^{\#} \in \mathcal{D}$ in the $(i+1)$-th rewrite step. We have to show that there exists an edge between those two dependency pairs. By definition of our $(\mathcal{D}, \mathcal{R})$-chain we have $t_i \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} t_{i+1}$ (i.e., $t_i \xrightarrow{\mathrm{i}}_{\mathcal{D},\mathcal{R}} \circ \xrightarrow{\mathrm{i}}_{\mathcal{R}}^{*} t_{i+1}$) with $t_i = \ell_i^{\#}\sigma_i$ and $t_{i+1} = \ell_{i+1}^{\#}\sigma_{i+1}$. To be more precise, we have

$$t_i = \ell_i^{\#}\sigma_i \xrightarrow{\mathrm{i}}_{\mathcal{D},\mathcal{R}} r_i^{\#}\sigma_i \xrightarrow{\mathrm{i}}_{\mathcal{R}}^{*} t_{i+1} = \ell_{i+1}^{\#}\sigma_{i+1}$$

Hence, we have $r_i^{\#}\sigma_i \xrightarrow{\mathrm{i}}_{\mathcal{R}}^{*} \ell_{i+1}^{\#}\sigma_{i+1}$ and thus there is an edge between $\ell_i^{\#} \to r_i^{\#}$ and $\ell_{i+1}^{\#} \to r_{i+1}^{\#}$ in the dependency graph.

Since the graph is finite, the path will eventually reach an SCC $\mathcal{D}_i$ and remain inside of it. In other words, we can find an $N \in \mathbb{N}$ such that the corresponding path for the chain

$$t_N \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} t_{N+1} \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} t_{N+2} \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} \ldots$$

is always inside of $\mathcal{D}_i$. This is then an infinite innermost $(\mathcal{D}_i, \mathcal{R})$-chain, and thus $(\mathcal{D}_i, \mathcal{R})$ is not innermost terminating. ■

In order to automatize the dependency graph processor, we need to automatically create the dependency graph. However this is not possible, as deciding whether there exists substitutions σ_1 and σ_2 such that $t\sigma_1 \xrightarrow{i}{}^*_{\mathcal{R}} v\sigma_2$ is undecidable in general. Therefore, we use an abstracted version of the dependency graph. This *abstracted dependency graph* is then a supergraph of our dependency graph, which means that we may have even more paths that do not correspond to some chain. However, this does not interfere with the soundness nor the completeness of the processor but only with its effectiveness. Next, we introduce our abstraction to automatically compute whether there is an edge between the two dependency pairs in the abstracted dependency graph or not.

Note that $t\sigma_1$ may only reduce to $v\sigma_2$ for some substitutions σ_1, σ_2, if either t has a defined root symbol or if both t and v have the same constructor symbol at the root. All of the subterms with defined root symbol may be evaluated with our TRS into new terms. Hence, we have to replace them with fresh variables. If $t\sigma_1$ reduces to $v\sigma_2$, then t' and v' must be unifiable, where t' is the result of this replacement of defined symbols into fresh variables from t. However, we only need to do this for subterms that are not equal to subterms of s since every proper subterm of s must be in normal form w.r.t. $\mathcal{R}$ by our innermost evaluation strategy.

Definition 3.3.5 (Connectable Terms, [1])**.** Let $s \in \mathcal{T}\left(\Sigma \uplus \Sigma^{\#}, \mathcal{V}\right)$ be some term. The function Cap_s is inductively defined as

$$\begin{array}{lll} \mathrm{Cap}_s(x) & = x & \text{, for variables } x, \\ \mathrm{Cap}_s(f(t_1,\ldots,t_n)) & = y & \text{, if } f \in \Sigma_D \text{ and } f(t_1,\ldots,t_n) \not\trianglelefteq s, \\ \mathrm{Cap}_s(f(t_1,\ldots,t_n)) & = f(\mathrm{Cap}_s(t_1),\ldots,\mathrm{Cap}_s(t_n)) & \text{, otherwise.} \end{array}$$

where y is the next variable in an infinite list of fresh variables.

Two terms t and v are *connectable* w.r.t. s if $\mathrm{Cap}_s(t)$ and v are unifiable by some mgu σ such that every proper subterm of $s\sigma$ and $v\sigma$ is in normal form w.r.t. $\mathcal{R}$.

Note that $\mathrm{Cap}_s(x)$ is not an actual *function* since we need to keep track of the fresh variables that we have already used. However, we can change the definition to a real function by adding additional arguments. We omit this here for readability. We can now use this abstraction to create our abstracted dependency graph.

Definition 3.3.6 (Abstracted Dependency Graph, [1])**.** For a DP problem $(\mathcal{D}, \mathcal{R})$ the nodes of the *abstracted* $(\mathcal{D}, \mathcal{R})$*-dependency graph* are the dependency pairs of $\mathcal{D}$ and there is an edge from $s \to t$ to $v \to w$ iff t and v are connectable w.r.t. s.

Theorem 3.3.7 (Computable Dependency Graph Processor, [1])**.** *Let*

$$\mathrm{Proc}_{\mathtt{CDG}}(\mathcal{D}, \mathcal{R}) = \{(\mathcal{D}_1, \mathcal{R}), \ldots, (\mathcal{D}_n, \mathcal{R})\}$$

where $\mathcal{D}_1, \ldots, \mathcal{D}_n$ are the SCCs of the abstracted $(\mathcal{D}, \mathcal{R})$-dependency graph. Then $\mathrm{Proc}_{\mathtt{CDG}}$ *is sound and complete.*

Proof. Let us first show that the abstracted dependency graph is a supergraph of the normal one. So assume that we have an edge between the two dependency pairs $s \to t$ and $v \to w$ in the $(\mathcal{D}, \mathcal{R})$-dependency graph. We will show by structural induction on t that if there exists a substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ such that every proper subterm of $s\sigma$ is in normal form w.r.t. $\mathcal{R}$ and $t\sigma \xrightarrow{i}{}^*_{\mathcal{R}} u$ for some term u, then there exists a substitution τ (whose domain only includes variables that are introduced in the construction of $\mathrm{Cap}_s(t)$) with $\mathrm{Cap}_s(t)\sigma\tau = u$. In particular, if there are substitutions σ_1, σ_2 such that every proper subterm of $s\sigma_1$ and $v\sigma_2$ is in normal form w.r.t. $\mathcal{R}$ and $t\sigma_1 \xrightarrow{i}{}^*_{\mathcal{R}} v\sigma_2$, then there exists a

substitution τ with $\mathrm{Cap}_s(t)\sigma_1\tau = v\sigma_2$. Hence, $\mathrm{Cap}_s(t)$ and v unify and the mgu μ is such that every proper subterm of $s\mu$ and $v\mu$ is in normal form w.r.t. $\mathcal{R}$.

If t equals a subterm of s, then t is in normal form w.r.t. $\mathcal{R}$ and hence $t\sigma = u$. Moreover, we have $\mathrm{Cap}_s(t) = t$ and hence $\mathrm{Cap}_s(t)\sigma = t\sigma = u$, so our desired substitution τ is the empty substitution.

If t is a variable or if $t = f(t_1, \ldots, t_n)$ for a defined symbol $f \in \Sigma_D$ such that t is not a subterm of s, then $\mathrm{Cap}_s(t)$ is a fresh variable. Let τ be the substitution that replaces $\mathrm{Cap}_s(t)$ by u. Then we get $\mathrm{Cap}_s(t)\sigma\tau = \mathrm{Cap}_s(t)\tau = u$.

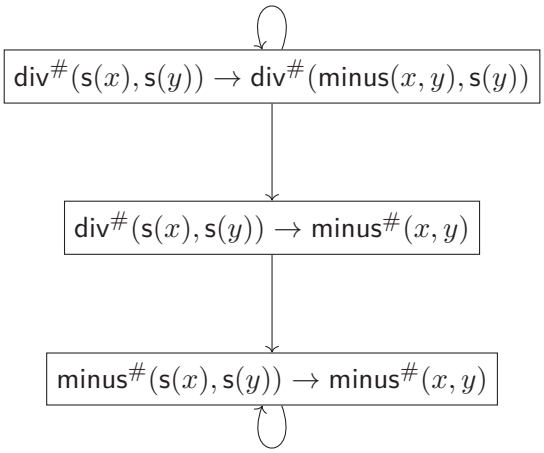
If $t = c(t_1, \ldots, t_n)$ for a constructor symbol $c \in \Sigma_C$ or for a tuple symbol $c \in \Sigma^\#$, then $\mathrm{Cap}_s(t) = c(\mathrm{Cap}_s(t_1), \ldots, \mathrm{Cap}_s(t_n))$. In this case, u has to be of the form $c(u_1, \ldots, u_n)$ and $t_j\sigma \xrightarrow{\mathsf{i}}_{\mathcal{R}}^* u_j$ holds for all $1 \leq j \leq n$. By our induction hypothesis we get substitutions τ_j such that $\mathrm{Cap}_s(t_j)\sigma\tau_j = u_j$. Since the variables in $\mathrm{Cap}_s(t_j)$ are disjoint from the variables in $\mathrm{Cap}_s(t_i)$ for all $i \neq j$, we can set $\tau \coloneqq \tau_1 \circ \ldots \circ \tau_n$ and get $\mathrm{Cap}_s(t_j)\sigma\tau = u_j$, and hence $\mathrm{Cap}_s(t)\sigma\tau = u$.

Now to the soundness and completeness of the computable dependency graph processor.

complete: Every innermost $(\mathcal{D}_i, \mathcal{R})$-chain is also an innermost $(\mathcal{D}, \mathcal{R})$-chain. Hence, if one of the $(\mathcal{D}_i, \mathcal{R})$ is not innermost terminating, then $(\mathcal{D}, \mathcal{R})$ is also not innermost terminating.

sound: Assume that $(\mathcal{D}, \mathcal{R})$ is not innermost terminating. Then there is some infinite innermost $(\mathcal{D}, \mathcal{R})$-chain. By Theorem 3.3.4, we then know that there is also an infinite innermost $(\mathcal{D}', \mathcal{R})$-chain, where $\mathcal{D}'$ is an SCC of the $(\mathcal{D}, \mathcal{R})$-dependency graph. Since the abstracted dependency graph is a super graph of the normal one, we can find an SCC $\mathcal{D}_i$ in the abstracted dependency graph with $\mathcal{D}' \subseteq \mathcal{D}_i$. Thus, we also have an infinite innermost $(\mathcal{D}_i, \mathcal{R})$-chain, which means that $(\mathcal{D}_i, \mathcal{R})$ is not innermost terminating as well. ■

Example 3.3.8 (Computable Dependency Graph Processor)**.** Let $\mathcal{R}_{div}$ be the PTRS from Example 3.0.1 and $\mathcal{DP}(\mathcal{R}_{div})$ be its dependency pairs from Example 3.1.4. The abstracted $(\mathcal{DP}(\mathcal{R}_{div}), \mathcal{R}_{div})$-dependency graph is the same as the $(\mathcal{DP}(\mathcal{R}_{div}), \mathcal{R}_{div})$-dependency graph and has the form:



The SCCs of this graph are $\{\mathsf{minus}^\#(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{minus}^\#(x, y)\} = \{(3.5)\}$ and $\{\mathsf{div}^\#(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{div}^\#(\mathsf{minus}(x, y), \mathsf{s}(y))\} = \{(3.7)\}$. Therefore, we have

$$\mathrm{Proc}_{\mathtt{CDG}}(\mathcal{DP}(\mathcal{R}_{div}), \mathcal{R}_{div}) = \{\big(\{(3.5)\}, \mathcal{R}_{div}\big), \big(\{(3.7)\}, \mathcal{R}_{div}\big)\}$$

Usable Rules Processor

The next processor that we want to introduce is the *usable rules processor*. This processor decreases the number of rewrite rules in our DP problem. The way we present the processor in this thesis is not complete. However, we are able to create a complete usable rules processor with some modifications to our DP problems that we describe later on. Remember the structure of a chain. We start with the left-hand side of a dependency pair $s_0 \to t_0$ together with a substitution σ such that $s_0\sigma$ is in normal form w.r.t. $\mathcal{R}$. Hence, the subterms of $t_0\sigma$ that are not in normal form w.r.t. $\mathcal{R}$ cannot be completely inside of the substitution. At least the root defined symbol must be part of t_0. Then rewriting with the TRS follows the same logic. Therefore, the right-hand sides of all dependency pairs and rewrite rules indicate which rewrite rules can be used in our chain. Those rewrite rules are called *usable rules*.

Example 3.3.9 (Usable Rules)**.** Let $\mathcal{R}_{div}$ be the PTRS from Example 3.0.1 and $\mathcal{DP}(\mathcal{R}_{div})$ be its dependency pairs from Example 3.1.4. After applying the dependency graph processor, we result with the DP problems $\Big(\{(3.5)\}, \mathcal{R}_{div}\Big)$ and $\Big(\{(3.7)\}, \mathcal{R}_{div}\Big)$. For the DP problem $\Big(\{(3.7)\}, \mathcal{R}_{div}\Big)$ we have the following rewrite rules and dependency pair left:

$$\begin{array}{ll} (3.1) & minus(x,0) \to x \\ (3.2) & minus(s(x),s(y)) \to minus(x,y) \\ (3.3) & div(0,s(y)) \to 0 \\ (3.4) & div(s(x),s(y)) \to s(div(minus(x,y),s(y))) \end{array} \qquad (3.7)\ div^{\#}(s(x),s(y)) \to div^{\#}(minus(x,y),s(y))$$

The right-hand side of the only dependency pair contains a subterm with minus at the root. Hence, we can use the rules (3.1) and (3.2) to evaluate this subterm. However, the defined symbol div does not occur in the right-hand side of the dependency pair nor in the right-hand side of the two minus-rules. Hence, the rules (3.3) and (3.4) cannot be used to evaluate the right-hand side of our dependency pair in an innermost chain.

The set $\mathcal{UR}(\mathcal{D},\mathcal{R})$ of usable rules is a computable over-approximation of all rewrite rules that can be used to rewrite the right-hand side of a dependency pair. To create this approximation, we recursively search for the defined root symbol of a rewrite rule in the right-hand side of all dependency pairs and rewrite rules, where we have already seen that they are usable. In this approximation, we exclude every rewrite rule, where the left-hand side has a redex as a proper subterm. Those rewrite rules can never be applied in an innermost rewrite step.

Definition 3.3.10 (Usable Rules, [1])**.** Let $(\mathcal{D},\mathcal{R})$ be a DP problem. For every $f \in \Sigma \uplus \Sigma^{\#}$ let $\mathrm{Rules}(\mathcal{R},f) \coloneqq \{\ell \to r \in \mathcal{R} \mid \mathrm{root}(\ell) = f, \ell \text{ has no redex as proper subterm}\}$. For any term t, the set of all *usable rules* $\mathcal{UR}(\mathcal{R},t)$ is recursively defined as

$$\begin{array}{lcl} \mathcal{UR}(\mathcal{R},x) & = & \varnothing \\ \mathcal{UR}(\mathcal{R},f(t_1,\ldots,t_n)) & = & \mathrm{Rules}(\mathcal{R},f) \cup \bigcup_{i=1}^{n} \mathcal{UR}(\mathcal{R}',t_j) \cup \\ & & \bigcup_{\ell\to r\in\mathrm{Rules}(\mathcal{R},f)} \mathcal{UR}(\mathcal{R}',r) \end{array}$$

where $\mathcal{R}' \coloneqq \mathcal{R} \setminus \mathrm{Rules}(\mathcal{R},f)$. The set of all *usable rules* for the DP problem $(\mathcal{D},\mathcal{R})$ is defined by

$$\mathcal{UR}(\mathcal{D},\mathcal{R}) \coloneqq \bigcup_{s\to t\in\mathcal{D}} \mathcal{UR}(\mathcal{R},t)$$

Theorem 3.3.11 (Usable Rules Processor, [1])**.** *Let*

$$\mathrm{Proc}_{\mathtt{UR}}(\mathcal{D},\mathcal{R}) = \{(\mathcal{D},\mathcal{UR}(\mathcal{D},\mathcal{R}))\}$$

Then $\mathrm{Proc}_{\mathtt{UR}}$ *is sound but not complete.*

Proof.

sound: Assume that $(\mathcal{D},\mathcal{R})$ is not innermost terminating. Then there exists an infinite innermost $(\mathcal{D},\mathcal{R})$-chain

$$t_0 \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} t_1 \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} t_2 \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} \dots$$

Fix some $i \in \mathbb{N}$. We have $t_i \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} t_{i+1}$ ($t_i \xrightarrow{\mathrm{i}}_{\mathcal{D},\mathcal{R}} \circ \xrightarrow{\mathrm{i}}{}^*_{\mathcal{R}} t_{i+1}$) and thus

$$t_i \xrightarrow{\mathrm{i}}_{\mathcal{D},\mathcal{R}} v_0 \xrightarrow{\mathrm{i}}_{\mathcal{R}} v_1 \xrightarrow{\mathrm{i}}_{\mathcal{R}} \dots \xrightarrow{\mathrm{i}}_{\mathcal{R}} v_k = t_{i+1}$$

for some $k \in \mathbb{N}$ and some terms $v_0, \dots, v_{k-1} \in \mathcal{T}^{\#}(\Sigma)$. We will prove by induction that all of the used rewrite rules are usable. In the end, we know that

$$t_i \xrightarrow{\mathrm{i}}_{\mathcal{D},\mathcal{R}} v_0 \xrightarrow{\mathrm{i}}_{\mathcal{UR}(\mathcal{D},\mathcal{R})} v_1 \xrightarrow{\mathrm{i}}_{\mathcal{UR}(\mathcal{D},\mathcal{R})} v_k = t_{i+1}$$

Hence, we also have infinite innermost $(\mathcal{D},\mathcal{UR}(\mathcal{D},\mathcal{R}))$-chain

$$t_0 \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{UR}(\mathcal{D},\mathcal{R}))} t_1 \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{UR}(\mathcal{D},\mathcal{R}))} t_2 \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{UR}(\mathcal{D},\mathcal{R}))} \dots$$

and thus $(\mathcal{D},\mathcal{UR}(\mathcal{D},\mathcal{R}))$ is also not innermost terminating.

Let us assume that in the first rewrite step $t_i \xrightarrow{\mathrm{i}}_{\mathcal{D},\mathcal{R}} v_0$ we use the dependency pair $s \to t \in \mathcal{D}$ and the ground substitution $\sigma \in \mathrm{Sub}(\Sigma,\mathcal{V})$ so that $t_i = s\sigma$, $v_0 = t\sigma$, and every proper subterm of $s\sigma$ is in normal form w.r.t. $\mathcal{R}$. Additionally, for the j-th rewrite step $v_j \xrightarrow{\mathrm{i}}_{\mathcal{R}} v_{j+1}$ with the TRS we use the rewrite rule $\ell_j \to r_j \in \mathcal{R}$, the ground substitution $\sigma_j \in \mathrm{Sub}(\Sigma,\mathcal{V})$, and the position π_j so that $v_j|_{\pi_j} = \ell_j\sigma_j$, $v_{j+1} = v_j[r_j\sigma_j]$, and every proper subterm of $\ell_j\sigma_j$ is in normal form w.r.t. $\mathcal{R}$.

For the induction base, we consider the rewrite step $v_0 \xrightarrow{\mathrm{i}}_{\mathcal{R}} v_1$. Here, we have $v_0 = t\sigma$ and $v_0|_{\pi_0} = \ell_0\sigma_0$. As we are dealing with innermost rewriting, the root defined symbol of ℓ_0 must be introduced by the term t and cannot be completely inside the substitution σ. Hence, we get $\ell_0 \to r_0 \in \mathcal{UR}(\mathcal{R},t) \subseteq \mathcal{UR}(\mathcal{D},\mathcal{R})$.

In the induction step, we consider the rewrite step $v_j \xrightarrow{\mathrm{i}}_{\mathcal{R}} v_{j+1}$. Here, we have $v_j = v_{j-1}[r_{j-1}\sigma_{j-1}]_{\pi_{j-1}}$ and $v_j|_{\pi_j} = \ell_j\sigma_j$. Again, the defined root symbol of ℓ_j must be introduced by a right-hand side of an earlier used TRS rule or by the right-hand side of the used dependency pair and cannot be completely inside of every used substitution. Since every earlier used rewrite rule is usable by our induction hypothesis, we also get $\ell \to r \in \mathcal{UR}(\mathcal{D},\mathcal{R})$. ■

Example 3.3.12 (Counterexample for Completeness)**.** Consider the signature $\Sigma = \{\mathsf{f},\mathsf{a}\}$, a variable set with $\{x\} \subseteq \mathcal{V}$, a TRS $\mathcal{R}$ consisting of the following rule:

$$\mathsf{a} \to \mathsf{a} \tag{3.12}$$

and a set of dependency pairs $\mathcal{D}$ consisting of the following dependency pair:

$$\mathsf{f}^{\#}(\mathsf{a},x) \to \mathsf{f}^{\#}(x,x) \tag{3.13}$$

Then, the DP problem $(\mathcal{D},\mathcal{R})$ is innermost terminating, since the only dependency pair (3.13) can not be used. The reason is that the left-hand side, namely $\mathsf{f}^{\#}(\mathsf{a},x)$, has the proper subterm a, which is not in normal form w.r.t. $\mathcal{R}$. However, the rule (3.12) is not usable, since the right-hand side of (3.13) does not contain the subterm a. We get $\mathrm{Proc}_{\mathtt{UR}}(\mathcal{D},\mathcal{R}) = \{(\mathcal{D},\mathcal{UR}(\mathcal{D},\mathcal{R}))\} = \{(\mathcal{D},\varnothing)\}$. Now, the DP problem $(\mathcal{D},\varnothing)$ is not innermost terminating anymore since there exists the following infinite $(\mathcal{D},\varnothing)$-chain

$$\mathsf{f}(\mathsf{a},\mathsf{a}) \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} \mathsf{f}(\mathsf{a},\mathsf{a}) \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} \dots$$

The reason here is that the proper subterm a is now in normal form w.r.t. the empty TRS.

The reason that the usable rules processor is not sound is due to the fact that the TRS $\mathcal{R}$ has two different roles in the DP problem $(\mathcal{D}, \mathcal{R})$. On the one side, we use the TRS for rewrite steps $\xrightarrow{\text{i}}_{\mathcal{R}}$. On the other side, we use the TRS for the innermost evaluation property in the definition of a chain. If we use an additional TRS $\mathcal{Q}$ to store the rules which we have to consider for innermost evaluation, then we can use the usable rules processor only for the TRS $\mathcal{R}$ but keep $\mathcal{Q}$ unchanged. This then results in a complete usable rules processor. In this thesis, we omit this third component for readability.

Example 3.3.13. Let $\mathcal{R}_{div}$ be the PTRS from Example 3.0.1 and $\mathcal{DP}(\mathcal{R}_{div})$ be its dependency pairs from Example 3.1.4. After applying the dependency graph processor, we result with the DP problems $\Big(\{(3.5)\}, \mathcal{R}_{div}\Big)$ and $\Big(\{(3.7)\}, \mathcal{R}_{div}\Big)$. For the DP problem $\Big(\{(3.7)\}, \mathcal{R}_{div}\Big)$ we have the following rewrite rules and dependency pair left:

$$\begin{array}{ll} (3.1) & minus(x,0) \to x \\ (3.2) & minus(s(x),s(y)) \to minus(x,y) \\ (3.3) & div(0,s(y)) \to 0 \\ (3.4) & div(s(x),s(y)) \to s(div(minus(x,y),s(y))) \end{array} \qquad (3.7)\ div^{\#}(s(x),s(y)) \to div^{\#}(minus(x,y),s(y))$$

And we get $\mathcal{UR}(\{(3.7)\}, \mathcal{R}_{div}) = \{(3.1), (3.2)\}$ as described in Example 3.3.9, For the DP problem $\Big(\{(3.5)\}, \mathcal{R}_{div}\Big)$ we have the following rewrite rules and dependency pair left:

$$\begin{array}{ll} (3.1) & minus(x,0) \to x \\ (3.2) & minus(s(x),s(y)) \to minus(x,y) \\ (3.3) & div(0,s(y)) \to 0 \\ (3.4) & div(s(x),s(y)) \to s(div(minus(x,y),s(y))) \end{array} \qquad (3.5)\ minus^{\#}(s(x),s(y)) \to minus^{\#}(x,y)$$

In this case, there is no defined symbol in the right-hand side of our only dependency pair and thus we have $\mathcal{UR}(\{(3.5)\}, \mathcal{R}_{div}) \coloneqq \varnothing$.

All in all, we get

$$\begin{aligned} \text{Proc}_{\text{UR}}\Big(\{(3.5)\}, \mathcal{R}_{div}\Big) &= \{\Big(\{(3.5)\}, \varnothing\Big)\} \\ \text{Proc}_{\text{UR}}\Big(\{(3.7)\}, \mathcal{R}_{div}\Big) &= \{\Big(\{(3.7)\}, \{(3.1), (3.2)\}\Big)\} \end{aligned}$$

Reduction Pair Processor

The last processor we want to introduce is the *reduction pair processor*. In the previous chapter, we introduced a technique to automatically prove innermost termination of a given TRS $\mathcal{R}$ by searching for a natural and monotonic polynomial interpretation Pol that strictly orders every rewrite rule $\ell \to r \in \mathcal{R}$ (i.e., $Pol(\ell) > Pol(r)$). We now want to apply the same logic to DP problems $(\mathcal{D}, \mathcal{R})$. It turns out that finding a polynomial interpretation for DP problems has several advantages compared to the direct application on a TRS. Remember that in the definition of a chain, we need to apply one step with $\xrightarrow{\text{i}}_{\mathcal{D},\mathcal{R}}$, and then we can use an arbitrary but finite amount of steps with $\xrightarrow{\text{i}}_{\mathcal{R}}$. The steps with the dependency pairs are the important ones, while the steps with the TRS are more of an auxiliary tool. Every infinite rewrite sequence must use an infinite amount of dependency pairs. Hence, it suffices to find a polynomial interpretation Pol such that all of the dependency pairs are strictly decreasing (i.e., for every $\ell^{\#} \to r^{\#} \in \mathcal{D}$ we have $Pol(\ell^{\#}) > Pol(r^{\#})$) and all of the TRS rules are non-increasing (i.e., for every $\ell \to r \in \mathcal{R}$ we have $Pol(\ell) \geq Pol(r)$). So the first advantage is that we only need a strict decrease in the comparison of the left-hand side to certain subterms with defined root symbols in the right-hand side. The second and most important advantage is that we can use

weakly monotonic interpretations instead of monotonic ones. This means that we do not require that $x > y$ implies $f_{Pol}(\ldots, x, \ldots) > f_{Pol}(\ldots, y, \ldots)$ for all $f \in \Sigma \uplus \Sigma^{\#}$ but only that $x \geq y$ implies $f_{Pol}(\ldots, x, \ldots) \geq f_{Pol}(\ldots, y, \ldots)$ for all $f \in \Sigma \uplus \Sigma^{\#}$. This means that we can use polynomials that completely disregard one of its arguments, which was not possible before. The third advantage is that the symbol $f \in \Sigma_D$ and the corresponding tuple symbol $f^{\#}$ are two different objects. The polynomial interpretation can therefore handle them differently, as you can see in Example 3.3.15. The final advantage is that we can make this approach modular using our dependency pair framework. Instead of requiring a strict decrease for all dependency pairs, we can also try to order only a part of the dependency pairs strictly, while the rest is only non-increasing. We can then remove the dependency pairs that are strictly decreasing and continue working with a smaller DP problem. Then, we can try to use another processor (like the dependency graph processor) before searching for a polynomial interpretation for the reduction pair processor again.

Theorem 3.3.14 (Reduction Pair Processor, [14])**.** *Let $(\mathcal{D}, \mathcal{R})$ be a DP problem, and let $Pol : \mathcal{T}\left(\Sigma \uplus \Sigma^{\#}, \mathcal{V}\right) \to \mathbb{N}[\mathcal{V}]$ be a natural polynomial interpretation which is weakly monotonic (i.e., $x \geq y$ implies $f_{Pol}(\ldots, x, \ldots) \geq f_{Pol}(\ldots, y, \ldots)$ for all $f \in \Sigma \uplus \Sigma^{\#}$). Suppose that we have $\mathcal{D} = \mathcal{D}_{\succcurlyeq} \uplus \mathcal{D}_{\succ}$ such that the following conditions hold.*

(1) *For every $\ell \to r \in \mathcal{R}$ we have $Pol(\ell) \geq Pol(r)$.*

(2) *For every $\ell^{\#} \to r^{\#} \in \mathcal{D}_{\succ}$, we have $Pol(\ell^{\#}) > Pol(r^{\#})$.*

(3) *For every $\ell^{\#} \to r^{\#} \in \mathcal{D}_{\succcurlyeq}$, we have $Pol(\ell^{\#}) \geq Pol(r^{\#})$.*

Then

$$\mathrm{Proc}_{\mathtt{RP}}(\mathcal{D}, \mathcal{R}) = \{(\mathcal{D}_{\succcurlyeq}, \mathcal{R})\}$$

is sound and complete.

Proof. We start off by showing that the conditions (1),(2) and (3) of the theorem extend to rewrite steps instead of just rules:

(a) If $s, t \in \mathcal{T}^{\#}(\Sigma)$ with $s \xrightarrow{\mathrm{i}}_{\mathcal{R}} t$ using the rule $\ell \to r \in \mathcal{R}$ with $Pol(\ell) \geq Pol(r)$, then we have $Pol(s) \geq Pol(t)$.

(b) If $s, t \in \mathcal{T}^{\#}(\Sigma)$ with $s \xrightarrow{\mathrm{i}}_{\mathcal{D},\mathcal{R}} t$ using the dependency pair $\ell^{\#} \to r^{\#} \in \mathcal{D}$ with $Pol(\ell^{\#}) \geq Pol(r^{\#})$, then we have $Pol(s) \geq Pol(t)$.

(c) If $s, t \in \mathcal{T}^{\#}(\Sigma)$ with $s \xrightarrow{\mathrm{i}}_{\mathcal{D},\mathcal{R}} t$ using the dependency pair $\ell^{\#} \to r^{\#} \in \mathcal{D}$ with $Pol(\ell^{\#}) > Pol(r^{\#})$, then we have $Pol(s) > Pol(t)$.

(a): In this case, there exist a rule $\ell \to r \in \mathcal{R}$ with $Pol(\ell) \geq Pol(r)$, a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and a position π such that $s|_{\pi} = \ell\sigma$, $t = s[r\sigma]_{\pi}$, and every proper subterm of $\ell\sigma$ is in normal form w.r.t. $\mathcal{R}$.

We perform induction on π. So in the induction base, let $\pi = \varepsilon$. Hence, we have $s = \ell\sigma$ and $t = r\sigma$. By assumption we have $Pol(\ell) \geq Pol(r)$. As these inequations hold for all instantiations of the occurring variables, we have

$$Pol(s) = Pol(\ell\sigma) \geq Pol(r\sigma) = Pol(t).$$

In the induction step, we have $\pi = i.\pi'$, $s = f(s_1, \ldots, s_i, \ldots, s_n)$ and $s_i \to_{\mathcal{R}} t_i$. Then by the induction hypothesis we have $Pol(s_i) \geq Pol(t_i)$. For $t = f(s_1, \ldots, t_i, \ldots, s_n)$ we obtain

$$\begin{array}{rcl} Pol(s) & = & Pol(f(s_1, \ldots, s_i, \ldots, s_n)) \\ & = & f_{Pol}(Pol(s_1), \ldots, Pol(s_i), \ldots, Pol(s_n)) \\ & \geq & f_{Pol}(Pol(s_1), \ldots, Pol(t_i), \ldots, Pol(s_n)) \\ & & \qquad \text{(by weak monotonicity of } f_{Pol} \text{ and } Pol(s_i) \geq Pol(t_i)\text{)} \\ & = & Pol(f(s_1, \ldots, t_i, \ldots, s_n)) \\ & = & Pol(t). \end{array}$$

(b), (c): In the last two cases, there exist a dependency pair $\ell^\# \to r^\# \in \mathcal{R}$ with either $Pol(\ell^\#) \geq Pol(r^\#)$ or $Pol(\ell^\#) > Pol(r^\#)$, a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, such that $s = \ell^\#\sigma$, $t = r^\#\sigma$ and every proper subterm of $\ell^\#\sigma$ is in normal form w.r.t. $\mathcal{R}$. As these inequations hold for all instantiations of the occurring variables, we have

$$Pol(s) = Pol(\ell^\#\sigma) \geq Pol(r^\#\sigma) = Pol(t).$$

or

$$Pol(s) = Pol(\ell^\#\sigma) > Pol(r^\#\sigma) = Pol(t).$$

Now, we can prove the soundness and completeness of this processor.

complete: Every innermost $(\mathcal{D}_{\succcurlyeq}, \mathcal{R})$-chain is also an innermost $(\mathcal{D}, \mathcal{R})$-chain. Hence, if $(\mathcal{D}_{\succcurlyeq}, \mathcal{R})$ is not innermost terminating, then $(\mathcal{D}, \mathcal{R})$ is also not innermost terminating.

sound: Assume that $(\mathcal{D}, \mathcal{R})$ is not innermost terminating. Then there exists an infinite innermost $(\mathcal{D}, \mathcal{R})$-chain

$$t_0 \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} t_1 \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} t_2 \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} \ldots$$

such that for all $i \in \mathbb{N}$ we have $t_i = \ell_i^\#\sigma_i$ for some dependency pair $\ell_i^\# \to r_i^\# \in \mathcal{D}$ that we use in the i-th rewrite step, some ground substitution $\sigma_i \in \mathrm{Sub}(\Sigma, \mathcal{V})$ and such that every proper subterm of $\ell_i^\#\sigma_i$ is in normal form w.r.t. $\mathcal{R}$. We will now show that if we use an infinite number of $\to_{\mathcal{D}_\succ}$ steps in this chain, then it would be terminating, which is a contradiction to our assumption.

So assume that we use an infinite number of $\to_{\mathcal{D}_\succ}$ steps in this chain. Then

$$Pol(t_0) \geq Pol(t_1) \geq Pol(t_2) \geq \ldots$$

would contain an infinite number of strict relations $>$ and hence is an infinite descending sequence of natural numbers (since all of the occurring terms are ground terms), which is a contradiction.

Hence, we see only a finite number of $\to_{\mathcal{D}_\succ}$ steps, and after N steps for some $N \in \mathbb{N}$, we will only see dependency pairs from $\mathcal{D}_{\succcurlyeq}$ be used. This means that

$$t_N \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} t_{N+1} \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} t_{N+2} \xrightarrow{\mathrm{i}}_{(\mathcal{D},\mathcal{R})} \ldots$$

does not use any dependency pairs from $\mathcal{D}_\succ$ and can also be seen as an infinite innermost $(\mathcal{D}_{\succcurlyeq}, \mathcal{R})$-chain

$$t_N \xrightarrow{\mathrm{i}}_{(\mathcal{D}_{\succcurlyeq},\mathcal{R})} t_{N+1} \xrightarrow{\mathrm{i}}_{(\mathcal{D}_{\succcurlyeq},\mathcal{R})} t_{N+2} \xrightarrow{\mathrm{i}}_{(\mathcal{D}_{\succcurlyeq},\mathcal{R})} \ldots$$

Thus $(\mathcal{D}_{\succcurlyeq}, \mathcal{R})$ is not innermost terminating as well. ■

Example 3.3.15 (Reduction Pair Processor)**.** Let $\mathcal{R}_{div}$ be the PTRS from Example 3.0.1 and $\mathcal{DP}(\mathcal{R}_{div})$ be its dependency pairs from Example 3.1.4. After applying the usable rules processor, we result with the DP problems $\Big(\{(3.5)\}, \varnothing\Big)$ and $\Big(\{(3.7)\}, \{(3.1), (3.2)\}\Big)$. For the DP problem $\Big(\{(3.5)\}, \varnothing\Big)$ we can use the polynomial interpretation that maps $\mathsf{s}(x)$ to $x + 1$ and $\mathsf{m}^{\#}(x, y)$ to x. Note that we can now use polynomial interpretations that completely disregard some of their arguments, which was not possible before. With this polynomial interpretation, we have a strict decrease for the only dependency pair so that we result with

$$\mathrm{Proc}_{\mathtt{UR}}\Big(\{(3.5)\}, \varnothing\Big) \;= \{\Big(\varnothing, \varnothing\Big)\}$$

For the DP problem $\Big(\{(3.7)\}, \{(3.1), (3.2)\}\Big)$ we can use the polynomial interpretation that maps $\mathcal{O}$ to 0, $\mathsf{s}(x)$ to $x+1$ and both $\mathsf{minus}(x, y)$ and $\mathsf{div}^{\#}(x, y)$ to x. With this polynomial interpretation, we again have a strict decrease for the only dependency pair, so that we result with

$$\mathrm{Proc}_{\mathtt{UR}}\Big(\{(3.7)\}, \{(3.1), (3.2)\}\Big) \;= \{\Big(\varnothing, \{(3.1), (3.2)\}\Big)\}$$

Example 3.3.16. Again, consider $\mathcal{R}_{div}$ and $\mathcal{DP}(\mathcal{R}_{div})$ from Example 3.0.1 and Example 3.1.4. We are also able to prove termination directly using only the reduction pair processor. The constraints of the reduction pair processor are satisfied by the polynomial interpretation which maps $\mathcal{O}$ to 1, $\mathsf{s}(x)$ to $2x+1$, $\mathsf{minus}^{\#}(x, y)$ and $\mathsf{div}^{\#}(x, y)$ to $x+1$, and all other non-constant function symbols to the projection on their first argument. Now, for the rewrite rules, we have the following inequalities, where we write d for div and m for minus to ease readability:

$$\begin{array}{c} Pol(\mathsf{m}(x, \mathcal{O})) \geq Pol(x) \\ x \geq x \end{array} \qquad \begin{array}{c} Pol(\mathsf{m}(\mathsf{s}(x), \mathsf{s}(y))) \geq Pol(\mathsf{m}(x, y)) \\ 2x + 1 \geq x \end{array}$$

$$\begin{array}{c} Pol(\mathsf{d}(\mathcal{O}, \mathsf{s}(y))) \geq Pol(\mathcal{O}) \\ 1 \geq 1 \end{array} \qquad \begin{array}{c} Pol(\mathsf{d}(\mathsf{s}(x), \mathsf{s}(y))) \geq Pol(\mathsf{s}(\mathsf{d}(\mathsf{m}(x, y), \mathsf{s}(y)))) \\ 2x + 1 \geq 2x + 1 \end{array}$$

and for the dependency pairs, we have

$$\begin{array}{rcl} Pol(\mathsf{m}^{\#}(\mathsf{s}(x), \mathsf{s}(y))) & > & Pol(\mathsf{m}^{\#}(x, y)) \\ (2x + 1) + 1 & > & x + 1 \\ 2x + 2 & > & x + 1 \end{array}$$

$$\begin{array}{rcl} Pol(\mathsf{d}^{\#}(\mathsf{s}(x), \mathsf{s}(y))) & > & Pol(\mathsf{m}^{\#}(x, y)) \\ (2x + 1) + 1 & > & x + 1 \\ 2x + 2 & > & x + 1 \end{array}$$

$$\begin{array}{rcl} Pol(\mathsf{d}^{\#}(\mathsf{s}(x), \mathsf{s}(y))) & > & Pol(\mathsf{d}^{\#}(\mathsf{m}(x, y), \mathsf{s}(y))) \\ (2x + 1) + 1 & > & x + 1 \\ 2x + 2 & > & x + 1 \end{array}$$

The reduction pair processor does not only have advantages compared to the direct application of polynomial interpretations but is even strictly stronger, as stated by the following theorem.

Theorem 3.3.17 (RPP Subsumes Direct Application)**.** *Let $\mathcal{R}$ be a TRS. Suppose that there is a natural monotonic polynomial interpretation $Pol : \mathcal{T}(\Sigma, \mathcal{V}) \to \mathbb{N}[\mathcal{V}]$ such that for every rule $\ell \to r \in \mathcal{R}$ we have $Pol(\ell) > Pol(r)$. Then there is also a weakly-monotonic*

polynomial interpretation $Pol' : \mathcal{T}\left(\Sigma \uplus \Sigma^{\#}, \mathcal{V}\right) \to \mathbb{N}[\mathcal{V}]$ *that satisfies the conditions of the reduction pair processor (Theorem* 3.3.14*) for the DP problem* $(\mathcal{DP}(\mathcal{R}), \mathcal{R})$ *such that* $Pol(\ell^{\#}) > Pol(r^{\#})$ *for every dependency pair* $\ell^{\#} \to r^{\#} \in \mathcal{DP}(\mathcal{R})$.

Proof. The weakly-monotonic polynomial interpretation $Pol' : \mathcal{T}\left(\Sigma \uplus \Sigma^{\#}, \mathcal{V}\right) \to \mathbb{N}[\mathcal{V}]$ that we are searching is defined by $Pol'(f^{\#}) := Pol'(f) := Pol(f)$ for all $f \in \Sigma_D$ and $Pol'(f) := Pol(f)$ for all $f \in \Sigma_C$. Then Pol' is monotonic and hence also weakly-monotonic. For all rewrite rules $\ell \to r \in \mathcal{R}$, we have

$$Pol'(\ell) = Pol(\ell) > Pol(r) = Pol'(r)$$

and for all dependency pairs $s \to t \in \mathcal{DP}(\mathcal{R})$ we have

$$Pol'(s) = Pol(s^{\flat}) \overset{(*)}{>} Pol(t^{\flat}) = Pol'(t)$$

Here, the step $(*)$ is true since for every dependency pair $s \to t \in \mathcal{DP}(\mathcal{R})$ there is a rewrite rule $\ell \to r \in \mathcal{R}$ and a position $\pi \in \mathrm{Occ}(r)$ such that $\ell^{\#} = s$ and $r|_{\pi}^{\#} = t$. Thus we have $s^{\flat} = \ell$ and $t^{\flat} = r|_{\pi}$. Since $r|_{\pi}$ is a subterm of r we get

$$Pol(s^{\flat}) = Pol(\ell) > Pol(r) \geq Pol(r|_{\pi}) = Pol(t^{\flat})$$

where the second step holds by monotonicity of Pol. ■

Example 3.3.18. Consider $\mathcal{R}_{plus}$ from Example 2.1.16. Here, we get the only dependency pair:

$$\mathsf{plus}^{\#}(\mathsf{s}(x), y) \to \mathsf{plus}^{\#}(x, y) \tag{3.14}$$

In Example 2.2.4 we have seen that the polynomial interpretation, which maps $\mathsf{s}(x)$ to $x+1$, $\mathsf{plus}(x, y)$ to $2x+y+1$, and $\mathcal{O}$ to 0, strictly orders every rewrite rule. The constraints of the reduction pair processor for the DP problem $(\mathcal{DP}(\mathcal{R}_{plus}), \mathcal{R}_{plus})$ are satisfied by the polynomial interpretation which maps $\mathsf{s}(x)$ to $x+1$, $\mathsf{plus}^{\#}(x, y)$ and $\mathsf{plus}(x, y)$ to $2x+y+1$, and $\mathcal{O}$ to 0. Now, the rewrite rules are all strictly ordered, and for the dependency pair, we get:

$$\begin{array}{rcl} Pol(\mathsf{plus}^{\#}(\mathsf{s}(x), y)) & > & Pol(\mathsf{plus}^{\#}(x, y)) \\ 2(x+1)+y+1 & > & 2x+y+1 \\ 2x+y+3 & > & 2x+y+1 \end{array}$$

Thus, we can also use the dependency pair framework to prove innermost termination for this TRS.

Example 3.3.19 (Termination with Data Structures Cont.)**.** Consider $\mathcal{R}_{len}$ from Example 2.2.5. Here, we get the only dependency pair:

$$\mathsf{len}^{\#}(\mathsf{cons}(x, y)) \to \mathsf{len}^{\#}(y) \tag{3.15}$$

We can apply the reduction pair processor to the DP problem $(\mathcal{DP}(\mathcal{R}_{len}), \mathcal{R}_{len})$ with the polynomial interpretation that maps $\mathsf{len}(x)$ to x, $\mathsf{s}(x)$ to x, $\mathsf{cons}(x, y)$ to $y+1$, and both nil and $\mathcal{O}$ to 0. Now, the only dependency pair is strictly ordered. Hence, $(\mathcal{DP}(\mathcal{R}_{len}), \mathcal{R}_{len})$ is innermost terminating and thus $\mathcal{R}_{len}$ as well.

Final Automatic Innermost Termination Proof

Finally, we present the final, completely automatic, innermost termination proof for our TRS $\mathcal{R}_{div}$ in the following figure.

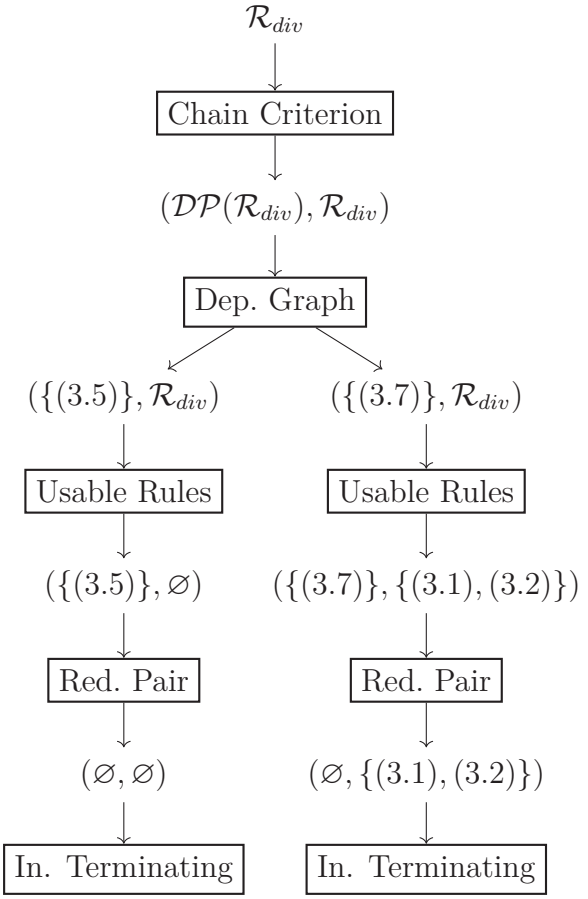


Figure 3.1: Automatic Innermost Termination Proof for $\mathcal{R}_{div}$

4 Probabilistic Term Rewriting

In the previous two chapters, we recapitulated the most important facts about automatic termination analysis of term rewriting systems, including one of the most powerful approaches, namely the dependency pair framework. Research in the field of term rewriting has been going on for decades. Another very prominent research area is probabilistic programming. In a probabilistic program, we have not only the possibility of deterministic or non-deterministic computation steps but also the possibility to perform a specific computation step only with a certain probability. We again want to prove termination automatically. The notion of termination can be interpreted in multiple ways for probabilistic programs. In this thesis, we focus our attention on almost-sure termination. In this chapter, we introduce the notion of a probabilistic term rewriting system (PTRS). We start with the basic definition and notations in Section 4.1. Instead of terms, we are now rewriting multi-distributions. In Section 4.2, we then view a rewrite sequence of multi-distributions as a tree and create a new characterization of almost-sure termination for a PTRS. Later, this new characterization is needed for some of the proofs, and we will even define the probabilistic version of a chain using trees in the next chapter. At the end of this chapter, we invent an automatic approach to prove almost-sure termination for a given probabilistic term rewriting system using polynomial interpretations in Section 4.3. Starting with Section 4.2, all of the presented results are new and our own contribution, if not mentioned otherwise.

4.1 Syntax, Semantics, and Evaluation Strategies

In this section, we recapitulate the notion of *probabilistic term rewriting systems*, which were introduced in [2, 11, 6]. While a term rewriting system is used to rewrite terms, a *probabilistic* TRS rewrites multi-distributions.

Definition 4.1.1 (Multi-Distribution)**.** A *multi-distribution* on a non-empty set A is a countable multiset μ of pairs $(p : a)$, where $0 < p \leq 1$ is a probability and $a \in A$ is an element, such that $\sum_{(p:a)\in\mu} p = 1$. The set of all multi-distributions on A is denoted by $\mathrm{Dist}(A)$. The *support* of a multi-distribution μ is defined as the multiset $\mathrm{Supp}(\mu) := \{a \mid (p : a) \in \mu \text{ for some } p\}$. The set of finitely supported multi-distributions is $\mathrm{FDist}(A) := \{\mu \mid \mu \in \mathrm{Dist}(A), \mathrm{Supp}(\mu) \text{ is finite}\}$. If $\mathrm{Supp}(\mu)$ does not contain any multiple occurrences of any $a \in A$ (i.e., if it is a set instead of a multiset), then μ is a distribution (and not just a multi-distribution). Every multi-distribution μ induces a mapping from A to $[0, 1]$, which is also denoted by μ: $\mu(a) = \sum_{(p:a)\in\mu} p$.

Again, we start with the simplest definition of a *probabilistic abstract rewrite system*. The only difference compared to an ARS is that we now have multi-distributions on the right-hand side of the rewrite relation.

Definition 4.1.2 (Probabilistic Abstract Rewrite System)**.** Let A be a non-empty set. A *probabilistic rewrite relation* is a subset $\to \subseteq A \times \mathrm{Dist}(A)$. We often write "$a \to \mu$" instead of "$(a, \mu) \in \to$". The pair $\mathcal{A} \coloneqq (A, \to)$ is a *probabilistic abstract rewrite system* (PARS).

Now, the definition of a probabilistic TRS is nearly the same as for a TRS as well. In this case, we have finitely supported multi-distributions on the right-hand side of the rewrite rules.

Definition 4.1.3 (Probabilistic Term Rewrite System)**.** Let Σ be a signature and $\mathcal{V}$ a set of variables. A *probabilistic term rewrite rule* is a pair $\ell \to \mu \in \mathcal{T}(\Sigma, \mathcal{V}) \times \mathrm{FDist}(\mathcal{T}(\Sigma, \mathcal{V}))$ such that $\mathcal{V}(r) \subseteq \mathcal{V}(\ell)$ for any $r \in \mathit{Supp}(\mu)$ and $\ell \notin \mathcal{V}$. A *probabilistic term rewriting system* (PTRS) is a finite set $\mathcal{R}$ of probabilistic term rewrite rules.

Similar to a TRS, the PTRS $\mathcal{R}$ induces a *probabilistic rewrite relation* $\to_{\mathcal{R}} \subseteq \mathcal{T}(\Sigma, \mathcal{V}) \times \mathrm{FDist}(\mathcal{T}(\Sigma, \mathcal{V}))$ via

$$s \to_{\mathcal{R}} \{p_1 : t_1, \dots, p_k : t_k\} \qquad :\Leftrightarrow \qquad s|_\pi = \ell\sigma \text{ and } t_j = s[r_j\sigma]_\pi \text{ for all } 1 \le j \le k,$$

for a position $\pi \in \mathrm{Occ}(s)$, a rule $\ell \to \{p_1 : r_1, \dots, p_k : r_k\} \in \mathcal{R}$, and a substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$. A subterm $s|_\pi$ that is matched by the left-hand side of a rule is once again called a *redex*. We call $s \to_{\mathcal{R}} \mu$ an innermost rewrite step (denoted $s \xrightarrow{\mathsf{i}}_{\mathcal{R}} \mu$) iff every proper subterm of the used redex $\ell\sigma$ is in normal form w.r.t. $\mathcal{R}$.

Analogous to the non-probabilistic setting, a term is in normal form w.r.t. $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ iff it is in normal form w.r.t. $\to_{\mathcal{R}}$. Hence, it suffices to speak about normal forms w.r.t. $\mathcal{R}$ for both cases again.

Example 4.1.4 (PTRS)**.** Consider the signature $\Sigma \coloneqq \{\mathcal{O}, \mathsf{s}, \mathsf{rd}\}$ and a set of variables with $x \in \mathcal{V}$. A PTRS $\mathcal{R}_{rw}$ is given by the following two probabilistic rewrite rules:

$$\mathsf{rd}(\mathsf{s}(x)) \to \{\tfrac{1}{2} : \mathsf{rd}(x),\ \tfrac{1}{2} : \mathsf{rd}(\mathsf{s}(\mathsf{s}(x)))\} \tag{4.1}$$
$$\mathsf{rd}(\mathcal{O}) \to \{1 : \mathcal{O}\}. \tag{4.2}$$

This PTRS corresponds to a fair random walk on the natural numbers that stops at 0. In the first rule, we have a chance of $\frac{1}{2}$ to decrease the number of s occurrences by one and a chance of $\frac{1}{2}$ to increase it by one.

A PARS can either be interpreted as a rewrite relation between elements with a certain probability or as a rewrite relation between *multi-distributions* of elements to track *all* possible rewrite sequences at once up to non-determinism. When considering the probability of termination for a PTRS, we have to consider every possible rewrite sequence with its probability on its own, and thus we use the second option. We can describe rewriting with PARSs in two different ways. On the one hand, a pair $(a, \mu) \in A \times \mathrm{Dist}(A)$ describes a single rewrite step, which we denote as $a \to \mu$. On the other hand, we can describe multiple rewrite steps on a single multi-distribution as a pair $(\mu_1, \mu_2) \in \mathrm{Dist}(A) \times \mathrm{Dist}(A)$, denoted $\mu_1 \rightrightarrows \mu_2$. For any $0 < p \le 1$ and any $\mu \in \mathrm{Dist}(A)$, let $p \cdot \mu \coloneqq \{(p \cdot q : a) \mid (q : a) \in \mu\}$.

Definition 4.1.5 (Lifting)**.** Let $(A, \to)$ be a PARS. The *lifting* $\rightrightarrows \subseteq \mathrm{Dist}(A) \times \mathrm{Dist}(A)$ of $\to \subseteq A \times \mathrm{Dist}(A)$ is the smallest relation that satisfies the following:

- If $a \in A$ is in normal form w.r.t. $\to$, then $\{1 : a\} \rightrightarrows \{1 : a\}$,
- If $a \to \mu$ then $\{1 : a\} \rightrightarrows \mu$,
- If for all $1 \le i \le k$ we have $\mu_i \rightrightarrows \nu_i$, $0 < p_i \le 1$ and $\sum_{1 \le i \le k} p_i = 1$, then $\bigcup_{1 \le i \le k} p_i \cdot \mu_i \rightrightarrows \bigcup_{1 \le i \le k} p_i \cdot \nu_i$.

A finite $\rightrightarrows$-rewrite sequence μ is a finite family $\mu = (\mu_k)_{k \in N}$ for some $N \in \mathbb{N}$ of multi-distributions such that $\mu_k \rightrightarrows \mu_{k+1}$ for all $1 \leq k \leq N$. An infinite $\rightrightarrows$-rewrite sequence μ is an infinite countable family $\mu = (\mu_k)_{k \in \mathbb{N}}$ of multi-distributions such that $\mu_k \rightrightarrows \mu_{k+1}$ for all $k \in \mathbb{N}$.

In order to ease readability, we will write for the rest of this thesis $n \in \mathbb{N}$ for the term $\mathsf{s}^n(\mathcal{O}) \coloneqq \underbrace{\mathsf{s}(\ldots(\mathsf{s}(}_{n\text{-times.}}\mathcal{O}\,)\ldots)$.

Example 4.1.6. Consider the PTRS $\mathcal{R}_{rw}$ from Example 4.1.4 and let $\rightrightarrows_{\mathcal{R}_{rw}}$ be the lifting of $\to_{\mathcal{R}_{rw}}$. Then we obtain the following rewrite sequence.

$$\begin{aligned} & \{1 : \mathsf{rd}(1)\} \\ \rightrightarrows_{\mathcal{R}_{rw}} \; & \{\tfrac{1}{2} : \mathsf{rd}(2), \tfrac{1}{2} : \mathsf{rd}(0)\} \\ \rightrightarrows_{\mathcal{R}_{rw}} \; & \{\tfrac{1}{4} : \mathsf{rd}(3), \tfrac{1}{4} : \mathsf{rd}(1), \tfrac{1}{2} : 0\} \\ \rightrightarrows_{\mathcal{R}_{rw}} \; & \{\tfrac{1}{8} : \mathsf{rd}(4), \tfrac{1}{8} : \mathsf{rd}(2), \tfrac{1}{8} : \mathsf{rd}(2), \tfrac{1}{8} : \mathsf{rd}(0), \tfrac{1}{2} : 0\} \end{aligned}$$

It is important to distinguish

$$\{\tfrac{1}{8} : \mathsf{rd}(4), \tfrac{1}{8} : \mathsf{rd}(2), \tfrac{1}{8} : \mathsf{rd}(2), \tfrac{1}{8} : \mathsf{rd}(0), \tfrac{1}{2} : 0\}$$

from

$$\{\tfrac{1}{8} : \mathsf{rd}(4), \tfrac{1}{4} : \mathsf{rd}(2), \tfrac{1}{8} : \mathsf{rd}(0), \tfrac{1}{2} : 0\}$$

because in the former multi-distribution, the two occurrences of $(\frac{1}{8} : \mathsf{rd}(2))$ could be rewritten differently if the PTRS had two different rules that apply to $\mathsf{rd}(2)$.

In the non-probabilistic case, we have a precise understanding of what it means for a program (or TRS) to terminate. In the probabilistic setting, however, it is not so clear how we should understand the notion of termination. The simplest idea would be to translate the definition of termination directly into the probabilistic setting.

Definition 4.1.7 (Sure Termination)**.** Let $\mathcal{R}$ be a PTRS. We call it *surely terminating* iff there is no infinite $\overset{\mathsf{i}}{\rightrightarrows}_{\mathcal{R}}$-rewrite sequence $(\mu_k)_{k \in \mathbb{N}}$ such that for all $i \in \mathbb{N}$, the support of the multi-distribution μ_i contains at least one term that is not in normal form w.r.t. $\mathcal{R}$.

However, this kind of termination seems too strict for probabilistic programs. For example, the PTRS $\mathcal{R}_{rw}$ from Example 4.1.4 is not surely terminating as the rewrite sequence indicated in Example 4.1.6 can be infinitely prolonged, and it is easy to see that for all $i \in \mathbb{N}$ the multi-distribution μ_i contains at least one term that is not in normal form w.r.t. $\mathcal{R}_{rw}$. In fact, sure termination of a PTRS does not depend on the probabilities but just on the possible rewrite steps. Hence, we can transform a PTRS into a non-probabilistic version and analyze the termination of the resulting TRS using our existing dependency pair framework for TRSs.

Definition 4.1.8 (Non-Probabilistic Transformation for PTRSs)**.** Let $\mathcal{R}$ be a PTRS. The *non-probabilistic transformation* $\mathrm{np}(\mathcal{R})$ is the TRS defined by

$$\mathrm{np}(\mathcal{R}) \coloneqq \{\ell \to r_j \mid \ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{R}, 1 \leq j \leq k\}$$

Here, we transform every probabilistic rewrite rule $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\}$ of a PTRS $\mathcal{R}$ into k individual non-probabilistic rewrite rules $\ell \to r_j$. It is easy to see that a PTRS $\mathcal{R}$ is surely terminating iff $\mathrm{np}(\mathcal{R})$ is terminating. Hence, we can simply apply our non-probabilistic dependency pair framework on $\mathrm{np}(\mathcal{R})$ to prove sure termination for a given PTRS $\mathcal{R}$.

Back to the PTRS $\mathcal{R}_{rw}$, one can see that the probability of termination increases over time. In fact, one can even prove that in the limit, the probability of termination is 1. This notion of termination is called *almost-sure termination*. In order to express this, we have to determine the probability for normal forms in a multi-distribution.

Definition 4.1.9 ($\mathtt{NF}_{\mathcal{R}}$ and $|\mu|_{\mathcal{R}}$)**.** Let $\mathcal{R}$ be a PTRS. Remember that $\mathtt{NF}_{\mathcal{R}} \subseteq \mathcal{T}(\Sigma, \mathcal{V})$ denotes the set of all normal forms w.r.t. $\mathcal{R}$. If $\mu \in \mathrm{FDist}(\mathcal{T}(\Sigma, \mathcal{V}))$, then let $|\mu|_{\mathcal{R}} \in [0,1]$ be defined as $|\mu|_{\mathcal{R}} = \sum_{(p:t)\in\mu, t\in\mathtt{NF}_{\mathcal{R}}} p$.

Example 4.1.10. Consider the multi-distribution

$$\mu = \{\tfrac{1}{8} : \mathsf{rd}(4), \tfrac{1}{8} : \mathsf{rd}(2), \tfrac{1}{8} : \mathsf{rd}(2), \tfrac{1}{8} : 0, \tfrac{1}{2} : 0\}$$

which results from the multi-distribution in Example 4.1.6 by replacing $\mathsf{rd}(0)$ with 0. Then $|\mu|_{\mathcal{R}_{rw}} = \frac{1}{8} + \frac{1}{2} = \frac{5}{8}$.

The following definition defines the probability for convergence (i.e., for "termination") of a $\rightrightarrows_{\mathcal{R}}$-rewrite sequence.

Definition 4.1.11 (Convergence Probability)**.** Let $\mathcal{R}$ be a PTRS and $(\mu_k)_{k\in\mathbb{N}}$ be an infinite $\rightrightarrows_{\mathcal{R}}$-rewrite sequence. Note that for every normal form t we have $\mu_k(t) \leq \mu_{k+1}(t) \leq 1$ for every $k \in \mathbb{N}$, i.e., $\lim_{k\to\infty} \mu_k(t)$ exists. Furthermore, $|\mu_k|_{\mathcal{R}} \leq |\mu_{k+1}|_{\mathcal{R}} \leq 1$ for every $k \in \mathbb{N}$, so the same argument proves the existence of $\lim_{k\to\infty} |\mu_k|_{\mathcal{R}}$. We say that $(\mu_k)_{k\in\mathbb{N}}$ converges with probability $q \in [0,1]$ if $\lim_{k\to\infty} |\mu_k|_{\mathcal{R}} = q$. We will also write $|\mu_\infty|_{\mathcal{R}}$ for $\lim_{k\to\infty} |\mu_k|_{\mathcal{R}}$.

Definition 4.1.12 (Almost-Sure Termination)**.** Let $\mathcal{R}$ be a PTRS. We call it *almost-surely terminating* (AST) iff every $\rightrightarrows_{\mathcal{R}}$-rewrite sequence converges with probability one, i.e., for every infinite $\rightrightarrows_{\mathcal{R}}$-rewrite sequence $(\mu_k)_{k\in\mathbb{N}}$ we have $|\mu_\infty|_{\mathcal{R}} = \lim_{k\to\infty} |\mu_k|_{\mathcal{R}} = 1$.

This definition is equivalent to the definitions of AST in the other papers on probabilistic term rewriting. E.g. Avanzini et. al. [2] use sub-distributions instead of multi-distributions and remove pairs once the corresponding term is in normal form w.r.t. $\mathcal{R}$. They then say that a PTRS is AST iff for every infinite $\rightrightarrows_{\mathcal{R}}$-rewrite sequence $(\mu_k)_{k\in\mathbb{N}}$ we have $\lim_{k\to\infty} ||\mu_k|| = 0$. Here, for a sub-distribution μ the value of $||\mu||$ is simply the sum over all occurring probabilities (i.e., $||\mu|| = \sum_{(p:t)\in\mu} p$). As they are removing normal forms from their distributions, this essentially means that the probability for terms not in normal form tends to 0. Using the notations from this thesis, this can be described by $\lim_{k\to\infty} 1 - |\mu_k|_{\mathcal{R}} = 0$ and this is equivalent to our notion of almost-sure termination, which requires $\lim_{k\to\infty} |\mu_k|_{\mathcal{R}} = 1$. The connection to Bournez et. al. [6] is also described in [2].

There are also other types of termination (e.g., positive almost-sure termination) that are important as well. In this thesis, we will restrict our attention to AST. It remains to be shown whether the results of this thesis can be applied to other kinds of termination as well. Remember that we restricted ourselves to innermost evaluations for this thesis. We can reformulate all of our new definitions with respect to innermost evaluation as well.

Definition 4.1.13 (Innermost Termination, Innermost AST)**.** Let $\mathcal{R}$ be a PTRS and let $\overset{\mathsf{i}}{\rightrightarrows}_{\mathcal{R}}$ be the lifting of $\overset{\mathsf{i}}{\to}_{\mathcal{R}}$. We call it *innermost surely terminating* iff there is no infinite $\overset{\mathsf{i}}{\rightrightarrows}_{\mathcal{R}}$-rewrite sequence $(\mu_k)_{k\in\mathbb{N}}$ such that for all $i \in \mathbb{N}$, the support of the multi-distribution μ_i contains at least one term that is not in normal form w.r.t. $\mathcal{R}$. We call it *innermost almost-surely terminating* (innermost AST) iff every infinite $\overset{\mathsf{i}}{\rightrightarrows}_{\mathcal{R}}$-rewrite sequence converges with probability one, i.e., for every infinite $\overset{\mathsf{i}}{\rightrightarrows}_{\mathcal{R}}$-rewrite sequence $(\mu_k)_{k\in\mathbb{N}}$ we have $|\mu_\infty|_{\mathcal{R}} = \lim_{k\to\infty} |\mu_k|_{\mathcal{R}} = 1$.

At the end of this section, we will once again have a short look at the connection between the different evaluation strategies. We can define *leftmost innermost evaluation* analogous to the non-probabilistic setting. It turns out that we have no equivalence between leftmost innermost evaluation and innermost evaluation w.r.t. AST. Only the trivial direction holds.

Theorem 4.1.14 (Leftmost Innermost AST vs. Innermost AST)**.** *If a PTRS $\mathcal{R}$ is innermost AST, then it is also leftmost innermost AST. On the other hand, there exists a PTRS that is leftmost innermost AST but not innermost AST.*

Proof. Every leftmost innermost rewrite sequence is also an innermost rewrite sequence. For a counterexample to the other direction, see Example 4.1.15. ■

Example 4.1.15 (Leftmost Innermost AST vs. Innermost AST)**.** Consider the PTRS $\mathcal{R}$ consisting of the following rewrite rules over the signature $\Sigma = \Sigma_0 \uplus \Sigma_2$ with $\Sigma_0 = \{\mathsf{a}, \mathsf{b}, \mathsf{c}_1, \mathsf{c}_2, \mathsf{d}_1, \mathsf{d}_2\}$ and $\Sigma_2 = \{\mathsf{f}\}$.

$$\mathsf{f}(\mathsf{c}_1, \mathsf{d}_1) \to \{1 : \mathsf{f}(\mathsf{a}, \mathsf{b})\} \tag{4.3}$$
$$\mathsf{f}(\mathsf{c}_2, \mathsf{d}_2) \to \{1 : \mathsf{f}(\mathsf{a}, \mathsf{b})\} \tag{4.4}$$
$$\mathsf{b} \to \{\tfrac{1}{2} : \mathsf{d}_1, \tfrac{1}{2} : \mathsf{d}_2\} \tag{4.5}$$
$$\mathsf{a} \to \{1 : \mathsf{c}_1\} \tag{4.6}$$
$$\mathsf{a} \to \{1 : \mathsf{c}_2\} \tag{4.7}$$
$$\tag{4.8}$$

Here, $\mathcal{R}$ is not innermost AST as we have the following innermost rewrite sequence that converges with probability 0:

$$\begin{aligned} & \{1 : \mathsf{f}(\mathsf{a}, \mathsf{b})\} \\ \xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \{\tfrac{1}{2} : \mathsf{f}(\mathsf{a}, \mathsf{d}_1), \tfrac{1}{2} : \mathsf{f}(\mathsf{a}, \mathsf{d}_2)\} \\ \xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \{\tfrac{1}{2} : \mathsf{f}(\mathsf{c}_1, \mathsf{d}_1), \tfrac{1}{2} : \mathsf{f}(\mathsf{c}_2, \mathsf{d}_2)\} \\ \xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \{\tfrac{1}{2} : \mathsf{f}(\mathsf{a}, \mathsf{b}), \tfrac{1}{2} : \mathsf{f}(\mathsf{a}, \mathsf{b})\} \\ \xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \dots \end{aligned}$$

In contrast, $\mathcal{R}$ is leftmost innermost AST because we have to rewrite a before we can rewrite b. Thus, we can either rewrite a to c_1 or c_2.

$$\begin{aligned} & \{1 : \mathsf{f}(\mathsf{a}, \mathsf{b})\} \\ \xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \{1 : \mathsf{f}(\mathsf{c}_1, \mathsf{b})\} \\ \xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \{\tfrac{1}{2} : \mathsf{f}(\mathsf{c}_1, \mathsf{d}_1), \tfrac{1}{2} : \mathsf{f}(\mathsf{c}_1, \mathsf{d}_2)\} \\ \xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \{\tfrac{1}{2} : \mathsf{f}(\mathsf{a}, \mathsf{b}), \tfrac{1}{2} : \mathsf{f}(\mathsf{c}_1, \mathsf{d}_2)\} \end{aligned}$$

Now we have a normal form with a certain probability after a fixed finite number of steps, and one can prove that every such rewrite sequence will converge with probability 1.

Whether the syntactic properties we mentioned in Chapter 2, which guarantee that innermost termination and full termination are equal, hold for probabilistic term rewriting and almost-sure termination remains an open question that we plan to address in the future.

4.2 Computation Trees

Next, we introduce a new way to describe a $\overset{\mathsf{i}}{\rightrightarrows}_{\mathcal{R}}$-rewrite sequence using trees.

Example 4.2.1 (Rewrite Sequence Tree)**.** Again, consider the PTRS $\mathcal{R}_{rw}$ from Example 4.1.4 and the finite $\overset{\mathsf{i}}{\rightrightarrows}_{\mathcal{R}_{rw}}$-rewrite sequence $\mu_0 \overset{\mathsf{i}}{\rightrightarrows}_{\mathcal{R}_{rw}} \mu_1 \overset{\mathsf{i}}{\rightrightarrows}_{\mathcal{R}_{rw}} \mu_2 \overset{\mathsf{i}}{\rightrightarrows}_{\mathcal{R}_{rw}} \mu_3$ from Example 4.1.6. We can represent this sequence as the following directed, labeled tree.

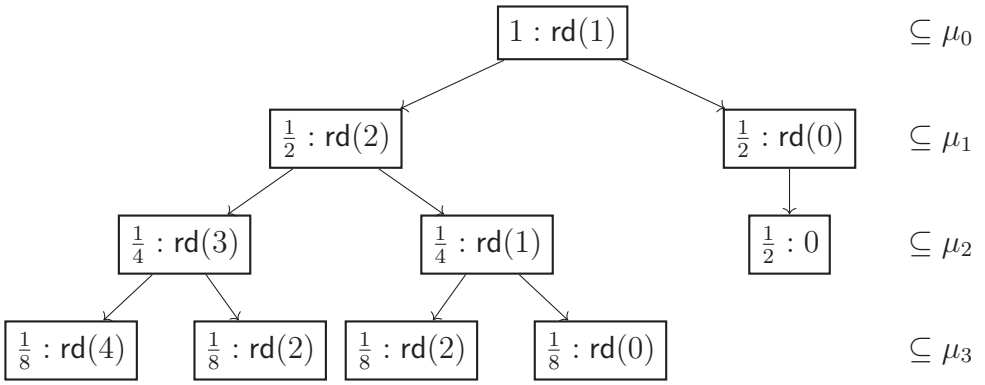


Figure 4.1: Tree Representation of the rewrite sequence from Example 4.1.6

The edges of the tree represent the used rewrite steps with $\overset{\mathsf{i}}{\to}_{\mathcal{R}}$, and the label of the nodes represent the pairs $(p : t)$ in our multi-distributions. There is one major difference between the $\overset{\mathsf{i}}{\rightrightarrows}_{\mathcal{R}}$-rewrite sequence and its tree representation. In the rewrite sequence, we would keep track of all normal forms, while in the tree, we turn normal forms into leaves and omit them in the next layer. Therefore, the nodes of the i-th layer of the tree represent a subset of μ_i that is only missing normal forms. For our example, the third layer of the tree is only a subset of μ_3 that is missing the pair $(\frac{1}{2} : 0)$, which is a leaf in the previous layer, since the term 0 is in normal form w.r.t. $\mathcal{R}_{rw}$.

In the rest of this section, we first introduce the general structure we want to work with, namely *rewrite sequence trees*, and their most important properties. Then we restrict the set of all rewrite sequence trees to those that represent $\overset{\mathsf{i}}{\rightrightarrows}_{\mathcal{R}}$-rewrite sequences for a given $\mathcal{R}$ and call them $\mathcal{R}$*-computation trees*. In the end, we use the tree representation of a rewrite sequence to prove our witness theorem for PTRS. For the rest of this section, let $\mathcal{R}$ be an arbitrary PTRS.

Definition 4.2.2 (Rewrite Sequence Tree)**.** A *rewrite sequence tree* (RST) is a directed, labeled tree $\mathfrak{T} = (V^{\mathfrak{T}}, E^{\mathfrak{T}}, L^{\mathfrak{T}})$ with

- $V^{\mathfrak{T}}$ is a possibly infinite, non-empty set of vertices.
- $E^{\mathfrak{T}} \subseteq V^{\mathfrak{T}} \times V^{\mathfrak{T}}$ is the set of directed edges.
- $L^{\mathfrak{T}} : V^{\mathfrak{T}} \to (0, 1] \times \mathcal{T}(\Sigma, \mathcal{V})$ labels every vertex by a probability and a term.

such that the following properties are satisfied:

(1.) $G^{\mathfrak{T}} = (V^{\mathfrak{T}}, E^{\mathfrak{T}})$ is a finitely branching, directed tree. Let $\mathrm{Leaf}^{\mathfrak{T}}$ be the set of all leaves in $G^{\mathfrak{T}}$ and let $\mathfrak{r}^{\mathfrak{T}}$ be the root node of $G^{\mathfrak{T}}$.

(2.) For all $x \in V^{\mathfrak{T}}$ with $L^{\mathfrak{T}}(x) = (p_x : t_x)$ and $xE^{\mathfrak{T}} \neq \varnothing$ we have

$$\sum_{y \in xE^{\mathfrak{T}}, L^{\mathfrak{T}}(y) = (p_y : t_y)} p_y = p_x$$

Here, by $xE^{\mathfrak{T}} \coloneqq \{y \mid (x,y) \in E^{\mathfrak{T}}\}$ we denote the set of all direct successors of x.

(3.) $L^{\mathfrak{T}}(\mathfrak{r}) = (1 : t)$ for some $t \in \mathcal{T}(\Sigma, \mathcal{V})$.

For a node $x \in V^{\mathfrak{T}}$ with $L^{\mathfrak{T}}(x) = (p : A)$, we will also write $p_x^{\mathfrak{T}}$ for p and $A_x^{\mathfrak{T}}$ for A. If the RST is clear from the context, we may omit it in the notations for readability (e.g., we write V for $V^{\mathfrak{T}}$). For a node $x \in V$, we write $\mathrm{d}(x)$ for the depth of x, that is, the number of edges in the path from the root to x.

Next, we introduce two important constructions regarding RSTs. First, we create induced sub RSTs. These are the RSTs that result when only looking at a certain subset of nodes.

Definition 4.2.3 (Induced Sub Rewrite Sequence Trees)**.** Let $\mathfrak{T} = (V, E, L)$ be an RST. Let $W \subseteq V$ be non-empty, weakly connected and for all $x \in W$ we have $xE \cap W = \varnothing$ or $xE \cap W = xE$. The property of being non-empty and weakly connected ensures that the resulting graph $G^{\mathfrak{T}[W]} = (W, E \cap (W \times W))$ is a tree again. The last property regarding the direct successors of a node ensures that the sum of probabilities for the direct successors of a node x is equal to the probability for the node x itself. We define the sub RST $\mathfrak{T}[W]$ by

$$\mathfrak{T}[W] \coloneqq (W, E \cap (W \times W), L^W)$$

Let $w \in W$ be the root of $G^{\mathfrak{T}[W]}$. To ensure that the root of our induced sub RST has the probability 1 again, we use the labeling $(L^W)(x) = (\frac{p_x^{\mathfrak{T}}}{p_w^{\mathfrak{T}}} : t_x^{\mathfrak{T}})$ for all nodes $x \in W$. If the original root of $\mathfrak{T}$ is still existent (i.e., $\mathfrak{r}^{\mathfrak{T}} \in W$), then we call the induced sub RST *grounded*.

Example 4.2.4 (Induced Sub Rewrite Sequence Tree)**.** The following RST $\mathfrak{T}_2$ is an induced sub RST of $\mathfrak{T}_1$ from Example 4.2.1.

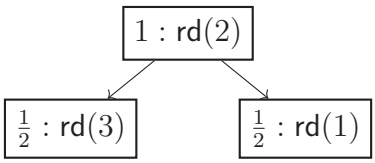


Figure 4.2: $\mathfrak{T}_2$ - Induced Sub RST of $\mathfrak{T}_1$

$\mathfrak{T}_2$ is not grounded since it does not contain the root from our original tree $\mathfrak{T}_1$. Instead, we start in $\mathfrak{T}_2$ with the left successor of the root in $\mathfrak{T}_1$.

Lemma 4.2.5. *Let $\mathfrak{T} = (V, E, L)$ be an RST and let $W \subseteq V$ be satisfying the conditions of Definition 4.2.3. Then $\mathfrak{T}[W] = (V', E', L')$ is an RST again.*

Proof. We have to show that $\mathfrak{T}[W] = (V', E', L') = (W, E \cap (W \times W), L^W)$ satisfies the conditions of Definition 4.2.2. Since W is non-empty and weakly connected, we know that the graph $G^{\mathfrak{T}[W]} = (W, E \cap (W \times W))$ is a finitely branching, directed tree again. Let $w \in W$ be the root of $G^{\mathfrak{T}[W]}$. For all $x \in W$ with $xE' \neq \varnothing$ we have $xE' = xE$ and thus

$$\sum_{y \in xE'} p_y^{\mathfrak{T}[W]} = \sum_{y \in xE'} \frac{p_y^{\mathfrak{T}}}{p_w^{\mathfrak{T}}} = \sum_{y \in xE} \frac{p_y^{\mathfrak{T}}}{p_w^{\mathfrak{T}}} = \frac{1}{p_w^{\mathfrak{T}}} \cdot \sum_{y \in xE} p_y^{\mathfrak{T}} = \frac{1}{p_w^{\mathfrak{T}}} \cdot p_x^{\mathfrak{T}} = \frac{p_x^{\mathfrak{T}}}{p_w^{\mathfrak{T}}} = p_x^{\mathfrak{T}[W]}$$

Finally, for the root $w \in W$, we have $p_w^{\mathfrak{T}[W]} = \frac{p_w^{\mathfrak{T}}}{p_w^{\mathfrak{T}}} = 1$ so that all of the conditions of an RST are satisfied. ∎

The second construction is the *extension* of an RST. Here, we exchange leaves of an existing TRS for new RSTs.

Definition 4.2.6 (Extension of Rewrite Sequence Trees)**.** Let $\mathfrak{T} = (V, E, L)$, $\mathfrak{T}' = (V', E', L')$ be RSTs with $V \cap V' = \varnothing$, and let $x \in \mathrm{Leaf}^{\mathfrak{T}}$ such that $t_x^{\mathfrak{T}} = t_{\mathfrak{r}^{\mathfrak{T}'}}^{\mathfrak{T}'}$ (i.e., the term of node x in $\mathfrak{T}$ is equal to the term of the root of $\mathfrak{T}'$). Then, we define the *extension* of $\mathfrak{T}$ w.r.t. the leaf x and the RST $\mathfrak{T}'$ (denoted as $\mathrm{Ext}(\mathfrak{T}, x, \mathfrak{T}')$) by

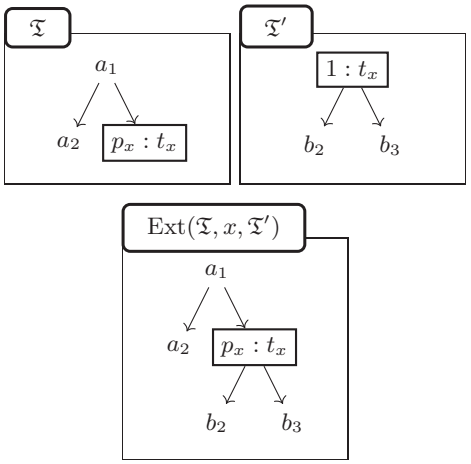
$$\mathrm{Ext}(\mathfrak{T}, x, \mathfrak{T}') \coloneqq (V_{\mathrm{Ext}}, E_{\mathrm{Ext}}, L_{\mathrm{Ext}})$$

with

$$\begin{aligned} V_{\mathrm{Ext}} &\coloneqq V \cup (V' \setminus \{\mathfrak{r}^{\mathfrak{T}'}\}) \\ E_{\mathrm{Ext}} &\coloneqq E \cup (E' \setminus \{(\mathfrak{r}^{\mathfrak{T}'}, y) \mid y \in \mathfrak{r}^{\mathfrak{T}'} E'\}) \\ &\quad \cup \{(x, y) \mid y \in \mathfrak{r}^{\mathfrak{T}'} E'\} \end{aligned}$$

And for all $z \in V_{\mathrm{Ext}}$ we define

$$L_{\mathrm{Ext}}(z) = \begin{cases} (p_z^{\mathfrak{T}}, t_z^{\mathfrak{T}}) & , z \in V, \\ (p_x^{\mathfrak{T}} \cdot p_z^{\mathfrak{T}'}, t_z^{\mathfrak{T}'}) & , z \in V'. \end{cases}$$



Example 4.2.7. Consider the RST $\mathfrak{T}_1$ from Example 4.2.1 and the RST $\mathfrak{T}_2$ from Example 4.2.4. We can use $\mathfrak{T}_2$ to extend $\mathfrak{T}_1$ in a leaf x that is labeled by $(\frac{1}{8} : \mathsf{rd}(2))$. Then $\mathrm{Ext}(\mathfrak{T}, x, \mathfrak{T}')$ has the form:

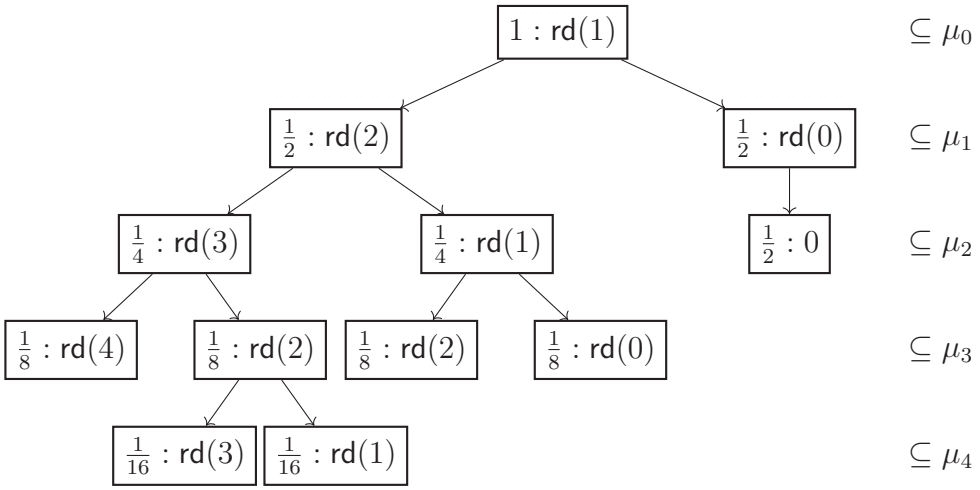


Figure 4.3: $\mathrm{Ext}(\mathfrak{T}_1, x, \mathfrak{T}_2)$

$\mathrm{Ext}(\mathfrak{T}_1, x, \mathfrak{T}_2)$ can be seen as a tree representation for a partial extension of our rewrite sequence, as we have set $(\frac{1}{16} : \mathsf{rd}(3)) \in \mu_4$ and $(\frac{1}{16} : \mathsf{rd}(3)) \in \mu_4$, but we have not yet decided how to rewrite the other pairs in μ_3.

Instead of extending a single leaf, we can also extend multiple leaves simultaneously.

Definition 4.2.8 (Family Extension of Rewrite Sequence Trees)**.** Let $\mathfrak{T} = (V, E, L)$ be an RST, $H \subseteq \mathrm{Leaf}^{\mathfrak{T}}$, and let $(\mathfrak{T}_x)_{x \in H}$ be a family of RSTs such that $\mathfrak{T}_x = (V_x, E_x, L_x)$, $t_x^{\mathfrak{T}} = t_{\mathfrak{r}^{\mathfrak{T}_x}}^{\mathfrak{T}_x}$ for all $x \in H$ and all occurring RSTs have disjoint node sets. Additionally, let $\mathrm{Ext}(\mathfrak{T}, x, \mathfrak{T}_x) = (V'_x, E'_x, L'_x)$. Then we define the *family extension* of $\mathfrak{T}$ w.r.t. the family $(\mathfrak{T}_x)_{x \in H}$ (denoted as $\mathrm{FamExt}(\mathfrak{T}, (\mathfrak{T}_x)_{x \in H})$) by

$$\begin{aligned} \mathrm{FamExt}(\mathfrak{T}, (\mathfrak{T}_x)_{x \in H}) &\coloneqq \textstyle\bigcup_{x \in H} \mathrm{Ext}(\mathfrak{T}, x, \mathfrak{T}_x) \\ &\coloneqq (\textstyle\bigcup_{x \in H} V'_x, \bigcup_{x \in H} E'_x, \bigcup_{x \in H} L'_x) \end{aligned}$$

Here, the labeling $\bigcup_{x\in H} L'_x$ is defined such that $\left(\bigcup_{x\in H} L'_x\right)|_{V'_x} = L'_x$ for all $x \in H$.

Lemma 4.2.9. *Let $\mathfrak{T} = (V, E, L)$, $\mathfrak{T}' = (V', E', L')$ be RSTs with $V \cap V' = \varnothing$, and let $x \in \mathrm{Leaf}^{\mathfrak{T}}$ such that $t^{\mathfrak{T}}_x = t^{\mathfrak{T}'}_{\mathfrak{r}^{\mathfrak{T}'}}$. Then $\mathrm{Ext}(\mathfrak{T}, x, \mathfrak{T}')$ is an RST. Furthermore, if we have $H \subseteq \mathrm{Leaf}^{\mathfrak{T}}$, and $(\mathfrak{T}_x)_{x\in H}$ is a family of RSTs such that $\mathfrak{T}_x = (V^x, E^x, L^x)$, $t^{\mathfrak{T}}_x = t^{\mathfrak{T}_x}_{\mathfrak{r}^{\mathfrak{T}_x}}$ for all $x \in H$ and with pairwise disjoint node sets. Then $\mathrm{FamExt}(\mathfrak{T}, (\mathfrak{T}_x)_{x\in H})$ is an RST as well.*

Proof. We show that $\mathrm{Ext}(\mathfrak{T}, x, \mathfrak{T}') = (V_{\mathrm{Ext}}, F_{\mathrm{Ext}}, L_{\mathrm{Ext}})$ satisfies the conditions of Definition 4.2.2. We exchange the leaf x with the tree $\mathfrak{T}'$. Since both $\mathfrak{T}$ and $\mathfrak{T}'$ are finitely branching, directed trees, we know that $\mathrm{Ext}(\mathfrak{T}, x, \mathfrak{T}')$ must be a finitely branching, directed tree as well. For the second property let $y \in V_{\mathrm{Ext}}$ with $yE_{\mathrm{Ext}} \neq \varnothing$. If we have $y = x$, then $xE_{\mathrm{Ext}} = \mathfrak{r}^{\mathfrak{T}'}E' \subseteq V'$ so that

$$\sum_{z\in xE_{\mathrm{Ext}}} p_z^{\mathrm{Ext}(\mathfrak{T},x,\mathfrak{T}')} = \sum_{z\in \mathfrak{r}^{\mathfrak{T}'}E'} p_x^{\mathfrak{T}} \cdot p_z^{\mathfrak{T}'} = p_x^{\mathfrak{T}} \cdot \sum_{z\in \mathfrak{r}^{\mathfrak{T}'}E'} p_z^{\mathfrak{T}'} = p_x^{\mathfrak{T}} \cdot p_{\mathfrak{r}^{\mathfrak{T}'}}^{\mathfrak{T}'} = p_x^{\mathfrak{T}} \cdot 1 = p_x^{\mathfrak{T}} = p_x^{\mathrm{Ext}(\mathfrak{T},x,\mathfrak{T}')}$$

If we have $y \in V$ with $y \neq x$, then also $yE_{\mathrm{Ext}} \subseteq V$ so that

$$\sum_{z\in yE_{\mathrm{Ext}}} p_z^{\mathrm{Ext}(\mathfrak{T},x,\mathfrak{T}')} = \sum_{z\in yE} p_z^{\mathfrak{T}} = p_y^{\mathfrak{T}} = p_y^{\mathrm{Ext}(\mathfrak{T},x,\mathfrak{T}')}$$

If we have $y \in V'$, then also $yE_{\mathrm{Ext}} \subseteq V'$ so that

$$\sum_{z\in yE_{\mathrm{Ext}}} p_z^{\mathrm{Ext}(\mathfrak{T},x,\mathfrak{T}')} = \sum_{z\in yE'} p_x^{\mathfrak{T}} \cdot p_z^{\mathfrak{T}'} = p_x^{\mathfrak{T}} \cdot \sum_{z\in yE_{\mathrm{Ext}}} p_z^{\mathfrak{T}'} = p_x^{\mathfrak{T}} \cdot p_y^{\mathfrak{T}'} = p_y^{\mathrm{Ext}(\mathfrak{T},x,\mathfrak{T}')}$$

Finally, we have $\mathfrak{r}^{\mathrm{Ext}(\mathfrak{T},x,\mathfrak{T}')} = \mathfrak{r}^{\mathfrak{T}} \in V$ and thus $p^{\mathrm{Ext}(\mathfrak{T},x,\mathfrak{T}')}_{\mathfrak{r}^{\mathrm{Ext}(\mathfrak{T},x,\mathfrak{T}')}} = p^{\mathfrak{T}}_{\mathfrak{r}^{\mathfrak{T}}} = 1$.

For a family extension, the arguments are completely analogous. We just have to distinguish the origin of the nodes for our case distinction. ■

For an RST, we are mostly interested in the probabilities stored in the leaves of the tree. This corresponds to the convergence probability of a rewrite sequence if we assume that all of the leaves represent normal forms, which is the case if the rewrite sequence is completely evaluated.

Definition 4.2.10 (Convergence Notations)**.** Let $\mathfrak{T}$ be an RST and $H \subseteq \mathrm{Leaf}$. Let $x_0, x_1, x_2, \ldots$ be an arbitrary enumeration of H. Then we define $|\mathfrak{T}|_H \in [0, 1]$ by:

$$|\mathfrak{T}|_H := \begin{cases} \sum_{v\in H} p_v & \text{, if } |H| \text{ finite,} \\ \lim_{i\to\infty} \sum_{v\in(x_k)_{k\leq i}} p_v & \text{, otherwise.} \end{cases}$$

If we have $|\mathfrak{T}|_H = c$ we say that $\mathfrak{T}$ *converges* w.r.t. H with probability c. If $H = \mathrm{Leaf}$ we simply say that $\mathfrak{T}$ *converges* with probability c.

Note that the value of $|\mathfrak{T}|_{\mathrm{Leaf}}$ does not depend on the enumeration but just on the RST $\mathfrak{T}$ due to the following lemma.

Lemma 4.2.11. *Let $\mathfrak{T}$ be an RST, $H \subseteq \mathrm{Leaf}$ be infinite, and let $x_0, x_1, x_2, \ldots$ be an arbitrary enumeration of H. Then $\sum_{v\in(x_k)_{k\leq i}} p_v$ is strictly increasing for $i \to \infty$ and bounded from above by 1. Hence, the sum $\sum_{v\in(x_k)_{k\leq i}} p_v$ is absolutely convergent for $i \to \infty$, i.e., $\lim_{i\to\infty} \sum_{v\in(x_k)_{k\leq i}} p_v$ exists and is therefore not dependent on the enumeration.*

Proof. It suffices, to proof this for $H = \mathrm{Leaf}$. Let $x_0, x_1, x_2, \ldots$ be an arbitrary enumeration of Leaf. Since $0 < p_x \leq 1$ for all $x \in V$ we know that the sum is strictly increasing. To see that it is bounded by 1 from above, note that for every finite RST $\mathfrak{T}' = (V', E', L')$ we have $|\mathfrak{T}'|_{\mathrm{Leaf}} = 1$.

We prove this by induction over the height of $\mathfrak{T}'$. In the induction base, we have an RST that only consists of the root $\mathfrak{r}$. Here, we have $|\mathfrak{T}'|_{\mathrm{Leaf}} = \sum_{x \in \mathrm{Leaf}} p_x = p_{\mathfrak{r}} = 1$ by definition of an RST. For the induction step, let $\mathfrak{T}'$ be an RST of height $h+1$. For $k \in \mathbb{N}$ let $\mathrm{D}(k) \coloneqq \{x \in V' \mid \mathrm{d}(x) = k\}$ denote the set of all nodes at depth k. Note that

$$\sum_{x \in \mathrm{D}(h+1) \cap \mathrm{Leaf}} p_x + \sum_{x \in \mathrm{D}(h) \cap \mathrm{Leaf}} p_x = \sum_{x \in \mathrm{D}(h)} p_x$$

since

$$\begin{array}{rl} & \sum_{x \in \mathrm{D}(h)} p_x \\ = & \sum_{x \in \mathrm{D}(h) \cap \mathrm{Leaf}} p_x + \sum_{x \in \mathrm{D}(h) \wedge xE \neq \varnothing} p_x \\ = & \sum_{x \in \mathrm{D}(h) \cap \mathrm{Leaf}} p_x + \sum_{x \in \mathrm{D}(h) \wedge xE \neq \varnothing} \sum_{y \in xE} p_y \\ = & \sum_{x \in \mathrm{D}(h) \cap \mathrm{Leaf}} p_x + \sum_{x \in \mathrm{D}(h+1)} p_x \\ = & \sum_{x \in \mathrm{D}(h) \cap \mathrm{Leaf}} p_x + \sum_{x \in \mathrm{D}(h+1) \cap \mathrm{Leaf}} p_x \end{array}$$

The reason for the last step is that $\mathfrak{T}'$ has height $h+1$, i.e., $\mathrm{D}(h+1) \subseteq \mathrm{Leaf}$. Thus, we can simply remove the nodes on the $(h+1)$-th depth of the tree and result in an RST $\mathfrak{T}'' \coloneqq \mathfrak{T}'[V' \setminus \mathrm{D}(h+1)]$ of height h with $|\mathfrak{T}'|_{\mathrm{Leaf}} = |\mathfrak{T}''|_{\mathrm{Leaf}}$. By our induction hypothesis, we get $|\mathfrak{T}''|_{\mathrm{Leaf}} = 1$, and this ends the proof.

Now assuming that there is an RST $\mathfrak{T}'$ with $|\mathfrak{T}'|_{\mathrm{Leaf}} > 1$, then there is an $N \in \mathbb{N}$ such that $\sum_{v \in (x_k)_{k \leq N}} p_v > 1$. This is a finite sum of finitely many leaves. Let $d_{\max} \coloneqq \max\{\mathrm{d}(x_1), \ldots, \mathrm{d}(x_N)\}$. Then $\bigcup_{0 \leq h \leq d_{\max}} \mathrm{D}(h)$ satisfies the conditions from Definition 4.2.3 and $\mathfrak{T}'' \coloneqq \mathfrak{T}'[\bigcup_{0 \leq h \leq d_{\max}} \mathrm{D}(h)]$ is a finite induced sub RST with $|\mathfrak{T}''|_{\mathrm{Leaf}} > 1$, which is a contradiction. ■

While for every finite RST $\mathfrak{T}$ we have $|\mathfrak{T}|_{\mathrm{Leaf}} = 1$, this does not hold for infinite RSTs in general. If you compare RSTs with $\overset{\mathrm{i}}{\rightrightarrows}_{\mathcal{R}}$-rewrite sequences, then every surely terminating rewrite sequence can be represented by a finite RST. Every infinite rewrite sequence that is AST can be represented by an infinite RST $\mathfrak{T}$ with $|\mathfrak{T}|_{\mathrm{Leaf}} = 1$. And every infinite rewrite sequence that is not AST can be represented by an infinite RST $\mathfrak{T}$ with $|\mathfrak{T}|_{\mathrm{Leaf}} < 1$.

Next, we investigate how the convergence ratio of an RST is impacted by our constructions.

Lemma 4.2.12. *Let $\mathfrak{T} = (V, E, L)$ and $\mathfrak{T}' = (V', E', L')$ be RSTs with $V \cap V' = \varnothing$, and let $x \in \mathrm{Leaf}^{\mathfrak{T}}$ such that $t^{\mathfrak{T}}_x = t^{\mathfrak{T}'}_{\mathfrak{r}^{\mathfrak{T}'}}$. Then we have*

$$|\operatorname{Ext}(\mathfrak{T}, x, \mathfrak{T}')|_{\mathrm{Leaf}} = |\mathfrak{T}|_{\mathrm{Leaf}} - p^{\mathfrak{T}}_x + p^{\mathfrak{T}}_x \cdot |\mathfrak{T}'|_{\mathrm{Leaf}}.$$

Furthermore, if we have $H \subseteq \mathrm{Leaf}^{\mathfrak{T}}$, and $(\mathfrak{T}_x)_{x \in H}$ is a family of RSTs such that $\mathfrak{T}_x = (V_x, E_x, L_x)$, $t^{\mathfrak{T}}_x = t^{\mathfrak{T}_x}_{\mathfrak{r}^{\mathfrak{T}_x}}$ for all $x \in H$ and with pairwise disjoint node sets. Then, we have

$$|\operatorname{FamExt}(\mathfrak{T}, (\mathfrak{T}_x)_{x \in H})|_{\mathrm{Leaf}} = |\mathfrak{T}|_{\mathrm{Leaf}} - |\mathfrak{T}|_H + \sum_{x \in H} p^{\mathfrak{T}}_x \cdot |\mathfrak{T}_x|_{\mathrm{Leaf}}.$$

Proof. Let $\mathfrak{T} = (V, E, L)$ and $\mathfrak{T}' = (V', E', L')$. Note that we replace a leaf, namely x, with the tree $\mathfrak{T}'$. If we remove x from the set of all leaves, then we decrease the value of $|\mathfrak{T}|_{\mathrm{Leaf}}$ by $p^{\mathfrak{T}}_x$. By definition of the new labeling $L^{\mathrm{Ext}(\mathfrak{T},x,\mathfrak{T}')}$ we have $p^{\mathrm{Ext}(\mathfrak{T},x,\mathfrak{T}')}_z = p^{\mathfrak{T}}_x \cdot p^{\mathfrak{T}'}_z$ for

all $z \in V'$. Therefore, the probability of leaves in the newly added part is

$$\begin{aligned} & \textstyle\sum_{z \in V' \wedge z \in \mathrm{Leaf}^{\mathrm{Ext}(\mathfrak{T},x,\mathfrak{T}')}} p_z^{\mathrm{Ext}(\mathfrak{T},x,\mathfrak{T}')} \\ = \; & \textstyle\sum_{z \in \mathrm{Leaf}^{\mathfrak{T}'}} p_z^{\mathrm{Ext}(\mathfrak{T},x,\mathfrak{T}')} \\ = \; & \textstyle\sum_{z \in \mathrm{Leaf}^{\mathfrak{T}'}} p_x^{\mathfrak{T}} \cdot p_z^{\mathfrak{T}'} \\ = \; & p_x^{\mathfrak{T}} \cdot \textstyle\sum_{z \in \mathrm{Leaf}^{\mathfrak{T}'}} p_z^{\mathfrak{T}'} \\ = \; & p_x^{\mathfrak{T}} \cdot |\mathfrak{T}'|_{\mathrm{Leaf}} \end{aligned}$$

All in all, we get $|\operatorname{Ext}(\mathfrak{T}, x, \mathfrak{T}')|_{\mathrm{Leaf}} = |\mathfrak{T}|_{\mathrm{Leaf}} - p_x^{\mathfrak{T}} + p_x^{\mathfrak{T}} \cdot |\mathfrak{T}'|_{\mathrm{Leaf}}$.

Now we consider the second part of the lemma. Let us first assume that $|H| = k \in \mathbb{N}$. Then we can apply the previous statement k times using induction to get our desired result. If $|H|$ is infinite, then we use an arbitrary enumeration $(x_n)_{n \in \mathbb{N}}$ of H. First, note that $\lim_{i \to \infty} \sum_{x \in (x_n)_{n \leq i}} p_x^{\mathfrak{T}} \cdot |\mathfrak{T}_x|_{\mathrm{Leaf}}$ exists, as the sequence $\sum_{x \in (x_n)_{n \leq i}} p_x^{\mathfrak{T}} \cdot |\mathfrak{T}_x|_{\mathrm{Leaf}}$ is strictly increasing for $i \to \infty$ and bounded from above by 1 because

$$\sum_{x \in (x_n)_{n \leq i}} p_x^{\mathfrak{T}} \cdot |\mathfrak{T}_x|_{\mathrm{Leaf}} \leq \sum_{x \in (x_n)_{n \leq i}} p_x^{\mathfrak{T}} \cdot 1 = \sum_{x \in (x_n)_{n \leq i}} p_x^{\mathfrak{T}} \leq |\mathfrak{T}|_{\mathrm{Leaf}} \leq 1$$

Next, let $\mathtt{FE} \coloneqq \operatorname{FamExt}(\mathfrak{T}, (\mathfrak{T}_x)_{x \in H})$ be the whole family extension and let $\mathtt{FE}_i \coloneqq \operatorname{FamExt}(\mathfrak{T}, (\mathfrak{T}_x)_{x \in (x_n)_{n \leq i}})$ be the family extension after performing the first i extension steps. We want to show that

$$|\mathtt{FE}|_{\mathrm{Leaf}} = \lim_{i \to \infty} |\mathtt{FE}_i|_{\mathrm{Leaf}} \tag{4.9}$$

To see this, note that

$$\mathrm{Leaf}^{\mathtt{FE}} = \lim_{i \to \infty} \mathrm{Leaf}^{\mathtt{FE}_i} \tag{4.10}$$

and for all $i \in \mathbb{N}$ all of the nodes that are contained in $\mathrm{Leaf}^{\mathtt{FE}_i}$ have the same probability in $\mathtt{FE}$ as in $\mathtt{FE}_i$, i.e.,

$$p_z^{\mathtt{FE}} = p_z^{\mathtt{FE}_i} \tag{4.11}$$

for all $z \in \mathrm{Leaf}^{\mathtt{FE}_i}$ and all $i \in \mathbb{N}$. Thus we have

$$\begin{aligned} & \lim_{i \to \infty} |\mathtt{FE}_i|_{\mathrm{Leaf}} \\ = \; & \lim_{i \to \infty} \textstyle\sum_{x \in \mathrm{Leaf}^{\mathtt{FE}_i}} p_x^{\mathtt{FE}_i} \\ = \; & \lim_{i \to \infty} \textstyle\sum_{x \in \mathrm{Leaf}^{\mathtt{FE}_i}} p_x^{\mathtt{FE}} \qquad \text{(by 4.11)} \\ = \; & \textstyle\sum_{x \in \mathrm{Leaf}^{\mathtt{FE}}} p_x^{\mathtt{FE}} \qquad \text{(by 4.10)} \\ = \; & |\mathtt{FE}|_{\mathrm{Leaf}} \end{aligned}$$

Finally, we get

$$\begin{aligned} & |\mathtt{FE}|_{\mathrm{Leaf}} \\ = \; & \lim_{i \to \infty} |\mathtt{FE}_i|_{\mathrm{Leaf}} \qquad \text{(by 4.9)} \\ = \; & \lim_{i \to \infty} \left(|\mathfrak{T}|_{\mathrm{Leaf}} - \textstyle\sum_{x \in (x_n)_{n \leq i}} p_x + \sum_{x \in (x_n)_{n \leq i}} p_x \cdot |\mathfrak{T}_x|_{\mathrm{Leaf}} \right) \\ = \; & |\mathfrak{T}|_{\mathrm{Leaf}} - \lim_{i \to \infty} \textstyle\sum_{x \in (x_n)_{n \leq i}} p_x + \lim_{i \to \infty} \sum_{x \in (x_n)_{n \leq i}} p_x \cdot |\mathfrak{T}_x|_{\mathrm{Leaf}} \\ = \; & |\mathfrak{T}|_{\mathrm{Leaf}} - |\mathfrak{T}|_H + \textstyle\sum_{x \in H} p_x \cdot |\mathfrak{T}_x|_{\mathrm{Leaf}} \end{aligned}$$

■

Corollary 4.2.13 (Little Extension Lemma)**.** *Let $\mathfrak{T} = (V, E, L)$ and $\mathfrak{T}' = (V', E', L')$ be RSTs with $V \cap V' = \varnothing$, and let $x \in \mathrm{Leaf}^{\mathfrak{T}}$ such that $t_x^{\mathfrak{T}} = t_{\mathfrak{r}_{\mathfrak{T}'}}^{\mathfrak{T}'}$. Then we have $|\operatorname{Ext}(\mathfrak{T}, x, \mathfrak{T}')|_{\mathrm{Leaf}} \leq |\mathfrak{T}|_{\mathrm{Leaf}}$. Moreover, $|\mathfrak{T}|_{\mathrm{Leaf}} = |\operatorname{Ext}(\mathfrak{T}, x, \mathfrak{T}')|_{\mathrm{Leaf}}$ iff $|\mathfrak{T}'|_{\mathrm{Leaf}} = 1$.*

Proof. We have

$$|\operatorname{Ext}(\mathfrak{T}, x, \mathfrak{T}')|_{\text{Leaf}} = |\mathfrak{T}|_{\text{Leaf}} - p_x^{\mathfrak{T}} + p_x^{\mathfrak{T}} \cdot |\mathfrak{T}'|_{\text{Leaf}} \leq |\mathfrak{T}|_{\text{Leaf}} - p_x^{\mathfrak{T}} + p_x^{\mathfrak{T}} \cdot 1 = |\mathfrak{T}|_{\text{Leaf}}$$

with equality iff $|\mathfrak{T}'|_{\text{Leaf}} = 1$. ■

Corollary 4.2.14 (Full Extension Lemma)**.** *Let $\mathfrak{T}$ be an RST, let $H \subseteq \text{Leaf}^{\mathfrak{T}}$, and $(\mathfrak{T}_x)_{x \in H}$ be a family of RSTs such that $\mathfrak{T}_x = (V_x, E_x, L_x)$, $t_x^{\mathfrak{T}} = t_{\text{r}^{\mathfrak{T}_x}}^{\mathfrak{T}_x}$ for all $x \in H$ and with pairwise disjoint node sets. Then we have $|\operatorname{FamExt}(\mathfrak{T}, (\mathfrak{T}_x)_{x \in \text{Leaf}})|_{\text{Leaf}} \leq |\mathfrak{T}|_{\text{Leaf}}$. Moreover, $|\mathfrak{T}|_{\text{Leaf}} = |\operatorname{FamExt}(\mathfrak{T}, (\mathfrak{T}_x)_{x \in \text{Leaf}})|_{\text{Leaf}}$ iff $|\mathfrak{T}_x|_{\text{Leaf}} = 1$ for all $x \in H$.*

Proof. We have

$$\begin{aligned}
& |\operatorname{FamExt}(\mathfrak{T}, (\mathfrak{T}_x)_{x \in H})|_{\text{Leaf}} \\
= \; & |\mathfrak{T}|_{\text{Leaf}} - |\mathfrak{T}|_H + \textstyle\sum_{x \in H} p_x^{\mathfrak{T}} \cdot |\mathfrak{T}_x|_{\text{Leaf}} \\
\leq \; & |\mathfrak{T}|_{\text{Leaf}} - |\mathfrak{T}|_H + \textstyle\sum_{x \in H} p_x^{\mathfrak{T}} \cdot 1 \\
= \; & |\mathfrak{T}|_{\text{Leaf}} - |\mathfrak{T}|_H + \textstyle\sum_{x \in H} p_x^{\mathfrak{T}} \\
= \; & |\mathfrak{T}|_{\text{Leaf}} - |\mathfrak{T}|_H + |\mathfrak{T}|_H \\
= \; & |\mathfrak{T}|_{\text{Leaf}}
\end{aligned}$$

with equality iff $|\mathfrak{T}_x|_{\text{Leaf}} = 1$ for all $x \in H$. ■

Lemma 4.2.15. *Let $\mathfrak{T}$ be an RST. Then we have $|\mathfrak{T}|_{\text{Leaf}} = 1$ iff for all induced sub RSTs $\mathfrak{T}'$ we have $|\mathfrak{T}'|_{\text{Leaf}} = 1$.*

Proof. Let $\mathfrak{T} = (V, E, L)$ be an RST. If every induced sub RST $\mathfrak{T}'$ converges with probability 1, then we know that also the induced sub RST $\mathfrak{T}[V] = \mathfrak{T}$ converges with probability 1. For the other direction, assume that $|\mathfrak{T}|_{\text{Leaf}} = 1$ and let $\mathfrak{T}'$ be an arbitrary induced sub RST of $\mathfrak{T}$. Then there exists a set W with $\mathfrak{T}' = \mathfrak{T}[W]$ satisfying the conditions of Definition 4.2.3. Let $w \in W$ be the root of $\mathfrak{T}[W]$ and let T_w be the induced sub RST that starts with w and contains every (not necessarily direct) successor of w. To be precise, let $wE^* \coloneqq \{y \in V \mid (z, y) \in E^*\}$ be the set of all (not necessarily direct) successor of w, including w itself. Here, by E^*, we denote the transitive and reflexive closure of E. Then we have $T_w \coloneqq \mathfrak{T}[wE^*]$. wE^* satisfies the conditions of Definition 4.2.3 since it is non-empty, weakly connected, and it contains every direct successor for each contained node. Note that we must have $|T_w|_{\text{Leaf}} = 1$. The reason is that if we had $|T_w|_{\text{Leaf}} < 1$, then

$$\begin{aligned}
& |\mathfrak{T}|_{\text{Leaf}} \\
= \; & \textstyle\sum_{x \in \text{Leaf}^{\mathfrak{T}}} p_x^{\mathfrak{T}} \\
= \; & \textstyle\sum_{x \in \text{Leaf}^{\mathfrak{T}} \setminus wE^*} p_x^{\mathfrak{T}} + \sum_{x \in \text{Leaf} \cap wE^*} p_x^{\mathfrak{T}} \\
= \; & \textstyle\sum_{x \in \text{Leaf}^{\mathfrak{T}} \setminus wE^*} p_x^{\mathfrak{T}} + \sum_{x \in \text{Leaf}^{T_w}} p_x^{\mathfrak{T}} \\
= \; & \textstyle\sum_{x \in \text{Leaf}^{\mathfrak{T}} \setminus wE^*} p_x^{\mathfrak{T}} + \sum_{x \in \text{Leaf}^{T_w}} p_w^{\mathfrak{T}} \cdot \frac{p_x^{\mathfrak{T}}}{p_w^{\mathfrak{T}}} \\
= \; & \textstyle\sum_{x \in \text{Leaf}^{\mathfrak{T}} \setminus wE^*} p_x^{\mathfrak{T}} + \sum_{x \in \text{Leaf}^{T_w}} p_w^{\mathfrak{T}} \cdot p_x^{T_w} \\
= \; & \textstyle\sum_{x \in \text{Leaf}^{\mathfrak{T}} \setminus wE^*} p_x^{\mathfrak{T}} + p_w^{\mathfrak{T}} \cdot \sum_{x \in \text{Leaf}^{T_w}} p_x^{T_w} \\
= \; & \textstyle\sum_{x \in \text{Leaf}^{\mathfrak{T}} \setminus wE^*} p_x^{\mathfrak{T}} + p_w^{\mathfrak{T}} \cdot |T_w|_{\text{Leaf}} \\
< \; & \textstyle\sum_{x \in \text{Leaf}^{\mathfrak{T}} \setminus wE^*} p_x^{\mathfrak{T}} + p_w^{\mathfrak{T}} \cdot 1 \\
= \; & \textstyle\sum_{x \in \text{Leaf}^{\mathfrak{T}} \setminus wE^*} p_x^{\mathfrak{T}} + p_w^{\mathfrak{T}} \\
= \; & \textstyle\sum_{x \in \text{Leaf}^{\mathfrak{T}} \setminus wE^+} p_x^{\mathfrak{T}} \\
= \; & \textstyle\sum_{x \in \text{Leaf}^{\mathfrak{T}[V \setminus wE^+]}} p_x^{\mathfrak{T}[V \setminus wE^+]} \\
= \; & |\mathfrak{T}[V \setminus wE^+]|_{\text{Leaf}} \\
\leq \; & 1
\end{aligned}$$

which is a contradiction to $|\mathfrak{T}|_{\text{Leaf}} = 1$. Here, $\mathfrak{T}[V \setminus wE^+]$ is the grounded induced sub RST of $\mathfrak{T}$, where we remove everything below w. As always, E^+ denotes the transitive

closure of E, and hence wE^+ is the set of all (not necessarily direct) successors of w. Again, $V \setminus wE^+$ is non-empty, weakly connected, and we remove either all successors of a node or none, so it satisfies the conditions of Definition 4.2.3.

Since T_w is a family extension of $\mathfrak{T}'$, we get $1 = |T_w|_{\text{Leaf}} \leq |\mathfrak{T}'|_{\text{Leaf}} \leq 1$ by the full extension lemma (Corollary 4.2.14) so that $|\mathfrak{T}'|_{\text{Leaf}} = 1$. ■

Now we analyze the connection between $\xrightarrow{\text{i}}_{\mathcal{R}}$-rewrite sequences and RSTs. Note that there are RSTs, that do not represent any $\xrightarrow{\text{i}}_{\mathcal{R}}$-rewrite sequence. The reason for that is that the definition of an RST does not depend on any PTRS $\mathcal{R}$. In fact, we have no restriction regarding the occurring terms in the labeling. Therefore, we introduce a refined version of an RST, namely an innermost $\mathcal{R}$-*Computation Tree*.

Definition 4.2.16 (Innermost $\mathcal{R}$-Computation Tree)**.** Let $\mathfrak{T} = (V, E, L)$ be a RTS. We call $\mathfrak{T}$ an innermost $\mathcal{R}$-*Computation Tree* iff the following condition hold:

(a) For every $x \in V$ with $xE = \{y_1, \ldots, y_k\} \neq \varnothing$ we have $t_x \xrightarrow{\text{i}}_{\mathcal{R}} \{\frac{p_{y_1}}{p_x} : t_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : t_{y_k}\}$

All of our constructions do not change the terms in the label of the nodes, and also, the ratio of p_y to p_x remains the same for every node x and successor y. Hence, the constructions also work for innermost $\mathcal{R}$-computation trees and not just RSTs.

Corollary 4.2.17. *Induces sub RSTs, extensions, and family extensions of innermost $\mathcal{R}$-computation trees are innermost $\mathcal{R}$-computation trees again.*

Proof. It is straightforward to see that all of the properties of Definition 4.2.16 hold since they hold for all of the initial innermost $\mathcal{R}$-computation trees. ■

We can now redefine innermost almost-sure termination PTRSs in terms of innermost computation trees.

Definition 4.2.18 (Innermost AST (2))**.** We call $\mathcal{R}$ *innermost almost-surely terminating* (innermost AST) iff every innermost $\mathcal{R}$-computation tree converges with probability one, i.e., for every innermost $\mathcal{R}$-computation tree $\mathfrak{T}$ we have $|\mathfrak{T}|_{\text{Leaf}} = 1$.

Next, we show that both of our definitions for innermost AST are equivalent. Before we do this, we prove that if a PTRS is not innermost AST, then there is also an infinite $\xrightarrow{\text{i}}_{\mathcal{R}}$-rewrite sequence $(\mu_k)_{k\in\mathbb{N}}$ *with a single start term* (i.e., $\mu_0 = \{1 : t\}$ for some $t \in \mathcal{T}(\Sigma, \mathcal{V})$) that converges with probability < 1. This is the reason why we only defined $\mathcal{R}$-computation *trees* and not $\mathcal{R}$-computation *forests*.

Lemma 4.2.19. *Suppose that there exists an infinite $\xrightarrow{\text{i}}_{\mathcal{R}}$-rewrite sequence $(\mu_k)_{k\in\mathbb{N}}$ that converges with probability < 1. Then there is also an infinite $\xrightarrow{\text{i}}_{\mathcal{R}}$-rewrite sequence $(\mu'_k)_{k\in\mathbb{N}}$ with a single start term (i.e., $\mu'_0 = \{1 : t\}$ for some $t \in \mathcal{T}(\Sigma, \mathcal{V})$) that converges with probability < 1.*

Proof. Let $(\mu_k)_{k\in\mathbb{N}}$ be an $\xrightarrow{\text{i}}_{\mathcal{R}}$-rewrite sequence that converges with probability < 1. Suppose that we have $\mu_0 = \{p_1 : t_1, \ldots, p_k : t_k\}$. Let $\{1 : t_i\}_{k\in\mathbb{N}}$ denote the infinite $\xrightarrow{\text{i}}_{\mathcal{R}}$-rewrite sequence that uses the same rules as μ does on the term t_i for every $1 \leq i \leq k$. Assume for a contradiction that for every $1 \leq i \leq k$ the $\xrightarrow{\text{i}}_{\mathcal{R}}$-rewrite sequence $\{1 : t_i\}_{k\in\mathbb{N}}$

converges with probability 1. Then we would have

$$\begin{aligned}|\mu_\infty|_{\mathcal{R}} &= \lim_{k\to\infty} \sum_{(p:t)\in\mu_k \wedge t\in \mathtt{NF}_{\mathcal{R}}} p \\ &= \lim_{k\to\infty} \sum_{1\le i\le k} p_i \cdot \sum_{(p:t)\in\{1:t_i\}_k \wedge t\in \mathtt{NF}_{\mathcal{R}}} p \\ &= \sum_{1\le i\le k} p_i \cdot \lim_{k\to\infty} \sum_{(p:t)\in\{1:t_i\}_k \wedge t\in \mathtt{NF}_{\mathcal{R}}} p \\ &= \sum_{1\le i\le k} p_i \cdot |\{1:t_i\}_\infty|_{\mathcal{R}} \\ &= \sum_{1\le i\le k} p_i \cdot 1 \\ &= \sum_{1\le i\le k} p_i \\ &= 1\end{aligned}$$

which is a contradiction to our assumption that we have $|\mu_\infty|_{\mathcal{R}} < 1$. Therefore, we have at least one $1 \le i \le k$ such that $\{1 : t_i\}_{k\in\mathbb{N}}$ converges with probability < 1. ■

Lemma 4.2.20. *There exists an infinite $\xrightarrow{\mathrm{i}}_{\mathcal{R}}$-rewrite sequence $(\mu_k)_{k\in\mathbb{N}}$ with $|(\mu_k)_{k\in\mathbb{N}}|_{\mathcal{R}} < 1$ iff there exists an innermost $\mathcal{R}$-computation tree $\mathfrak{T}$ with $|\mathfrak{T}|_{\mathrm{Leaf}} < 1$.*

Proof.

"⇒" Assume that there exists a $\xrightarrow{\mathrm{i}}_{\mathcal{R}}$-rewrite sequence $(\mu_k)_{k\in\mathbb{N}}$ that converges with probability < 1. By Lemma 4.2.19 we may assume that we have $\mu_0 = \{1 : t_\mathfrak{r}\}$ for some term $t_\mathfrak{r} \in \mathcal{T}(\Sigma, \mathcal{V})$.

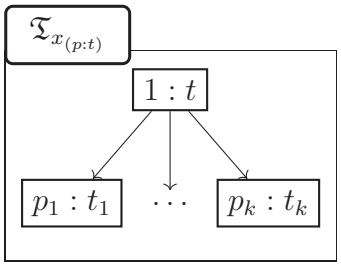
We will now create the corresponding tree representation for this rewrite sequence. This is the construction we applied to the rewrite sequence in Example 4.1.6 to get the tree representation in Example 4.2.1. To do so, we recursively define an innermost $\mathcal{R}$-computation tree $\mathfrak{T}_i = (V_i, E_i, L_i)$ for every $i \in \mathbb{N}$. During this construction, we will have for every $(p : t) \in \mu_i$ a (unique) corresponding leaf $x_{(p:t)}$ in $\mathfrak{T}_i$ labeled $L_i(x_{(p:t)}) = (p : t)$. Additionally, $\mathfrak{T}_{i+1}$ will be an extension of $\mathfrak{T}_i$ for every $i \in \mathbb{N}$.

We start with a single root $V_0 = \{\mathfrak{r}\}$, an empty set of edges $E_0 = \varnothing$ and the labeling $L(\mathfrak{r}) \coloneqq (1 : t_\mathfrak{r})$. Here, the properties are clearly satisfied.

Suppose we have already defined the innermost $\mathcal{R}$-computation tree $\mathfrak{T}_i$ for some $i \in \mathbb{N}$. We partition μ_i into the multisets $\mu_i = Z_{\mathtt{NF}} \uplus Z_{rew}$. Here, $Z_{\mathtt{NF}}$ contains all pairs $(p : t)$, were the term t is in normal form w.r.t. $\mathcal{R}$, and Z_{rew} contains all pairs $(p : t)$, were the term t is not in normal form w.r.t. $\mathcal{R}$.

For each $(p : t) \in Z_{\mathtt{NF}}$, we have a (unique) leaf $x_{(p:t)}$ in $\mathfrak{T}_i$ labeled $L_i(x_{(p:t)}) = (p : t)$, by our induction hypothesis. Since t is in normal form w.r.t. $\mathcal{R}$, we get $(p : t) \in \mu_{i+1}$. For our leaf in $\mathfrak{T}_i$, we do nothing as normal forms remain leaves in an innermost $\mathcal{R}$-computation tree.

For each $(p : t) \in Z_{rew}$, we again have a (unique) leaf $x_{(p:t)}$ in $\mathfrak{T}_i$ labeled $L_i(x_{(p:t)}) = (p : t)$, by our induction hypothesis. Since t is not in normal form w.r.t. $\mathcal{R}$, we can find a $\{p_1 : t_1, \dots, p_k : t_k\} \in FDist(\mathcal{T}(\Sigma, \mathcal{V}))$ such that $\{p \cdot p_1 : t_1, \dots, p \cdot p_k : t_k\} \subseteq \mu_{k+1}$ and $t \xrightarrow{\mathrm{i}}_{\mathcal{R}} \{p_1 : t_1, \dots, p_k : t_k\}$. Let $\mathfrak{T}_{x_{(p:t)}}$ be the innermost $\mathcal{R}$-computation tree depicted on the right.



We set $\mathfrak{T}_{i+1} \coloneqq \mathrm{FamExt}(\mathfrak{T}_i, (\mathfrak{T}_{x_{(p:t)}})_{(p:t)\in Z_{rew}})$. Here, we assume w.l.o.g. that every occurring tree has a pairwise disjoint node-set. It remains to show that our induction hypothesis is again satisfied for $i + 1$. Let $(p : t) \in \mu_{i+1}$. If t is in normal form w.r.t. $\mathcal{R}$ and the pair was already contained in μ_i, then by our induction hypothesis, there is a (unique) leaf $x_{(p:t)}$ in $\mathfrak{T}_i$ labeled $L_i(x_{(p:t)}) = (p : t)$. This leaf is also a leaf in $\mathfrak{T}_{i+1} = \mathrm{FamExt}(\mathfrak{T}_i, (\mathfrak{T}_{x_{(p:t)}})_{(p:t)\in Z_{rew}})$ so that our property

holds in this case. Otherwise, there exists a (unique) pair $(p' : t') \in \mu_i$ such that $t' \xrightarrow{\mathrm{i}}_{\mathcal{R}} \{p_1 : t_1, \ldots, p_k : t_k\}$, $\{p' \cdot p_1 : t_1, \ldots, p' \cdot p_k : t_k\} \subseteq \mu_{k+1}$, $p = p' \cdot p_j$, and $t = t_j$ for some $1 \leq j \leq k$. This means that there exists a (unique) leaf $x_{(p',t')}$ in $\mathfrak{T}_i$ labeled $L_i(x_{(p',t')}) = (p' : t')$ by our induction hypothesis. Now let y_j be the j-th successor of $x_{(p',t')}$ in $\mathfrak{T}_{x_{(p',t')}}$. We have

$$p_{y_j}^{\mathfrak{T}_{i+1}} = p_{y_j}^{\mathrm{FamExt}(\mathfrak{T}_i, (\mathfrak{T}_{x_{(p,t)}})_{(p,t)\in Z_{rew}})} = p_{y_j}^{\mathfrak{T}_{x_{(p',t')}}} = p_{x_{(p',t')}}^{\mathfrak{T}_i} \cdot p_{y_j}^{\mathfrak{T}_{x_{(p',t')}}} = p' \cdot p_j = p$$

and

$$t_{y_j}^{\mathfrak{T}_{i+1}} = t_{y_j}^{\mathrm{FamExt}(\mathfrak{T}_i, (\mathfrak{T}_{x_{(p,t)}})_{(p,t)\in Z_{rew}})} = t_{y_j}^{\mathfrak{T}_{x_{(p',t')}}} = t_j = t$$

and hence our property is again satisfied.

In the end, we set $\mathfrak{T} := \lim_{i\to\infty} \mathfrak{T}_i$. This is an innermost $\mathcal{R}$-computation tree again. First of all, $\mathfrak{T}$ is a directed tree again. Every other property of an innermost $\mathcal{R}$-computation tree is a local property regarding the nodes of the tree. Since every $\mathfrak{T}_i$ is an innermost $\mathcal{R}$-computation tree, and since after the i-th iteration of our creation, we do not change the nodes with a depth of less than i anymore, we know that $\mathfrak{T}$ must be an innermost $\mathcal{R}$-computation tree as well. Furthermore, note that we have $|\mathfrak{T}|_{\mathrm{Leaf}} = |\mu_\infty|_{\mathcal{R}}$, since

$$|\mathfrak{T}|_{\mathrm{Leaf}} = \lim_{i\to\infty} \sum_{x\in \mathrm{Leaf}^{\mathfrak{T}_i}, t_x \in \mathtt{NF}_{\mathcal{R}}} p_x = \lim_{i\to\infty} \sum_{(p:t)\in\mu_i, t\in\mathtt{NF}_{\mathcal{R}}} p = \lim_{i\to\infty} |\mu_i|_{\mathcal{R}} = |\mu_\infty|_{\mathcal{R}}$$

Hence, we have $|\mathfrak{T}|_{\mathrm{Leaf}} = |\mu_\infty|_{\mathcal{R}} < 1$.

"$\Leftarrow$" Assume that there exists an innermost $\mathcal{R}$-rewrite sequence $\mathfrak{T} = (V, E, L)$ that converges with probability < 1. W.l.o.G. we may assume that the following condition holds:

$$\text{for every leaf } x \in \mathrm{Leaf}, \text{ the corresponding term } t_x \text{ is in normal form w.r.t. } \mathcal{R}. \tag{4.12}$$

If there exists a leaf $x \in \mathrm{Leaf}$ such that the corresponding term t_x is not in normal form w.r.t. $\mathcal{R}$, then we can use arbitrary rewrite steps to completely evaluate it, and this can not increase the convergence probability of our tree $\mathfrak{T}$. Let $H \subseteq \mathrm{Leaf}$ be the set of leaves, where the corresponding term t_x is not in normal form w.r.t. $\mathcal{R}$. For each x, let $\mathfrak{T}_x$ be an arbitrary innermost $\mathcal{R}$-computation tree that satisfies (4.12). The family extension $\mathrm{FamExt}(\mathfrak{T}, (\mathfrak{T}_x)_{x\in H})$ now also satisfies (4.12) and due to the full extension lemma, we have $|\,\mathrm{FamExt}(\mathfrak{T}, (\mathfrak{T}_x)_{x\in H})|_{\mathrm{Leaf}} \leq |\mathfrak{T}|_{\mathrm{Leaf}} < 1$.

We will now inductively construct an infinite $\xrightarrow{\mathrm{i}}_{\mathcal{R}}$-rewrite sequence $(\mu_k)_{k\in\mathbb{N}}$ such that $|\mu_\infty|_{\mathcal{R}} = |\mathfrak{T}|_{\mathrm{Leaf}} < 1$. During this construction, we will have for every leaf $x_{(p:t)}$ in $\mathfrak{T}[D(i)]$ that is labeled $L(x_{(p:t)}) = (p : t)$ a (unique) pair $(p : t) \in \mu_i$. Here, for every $k \in \mathbb{N}$ we define $D(k) \coloneqq \{x \in W \mid \mathrm{d}(x) \leq k\}$ to be the set of all nodes in W that have a depth less or equal to k.

In the induction base, we have $D(0) = \{\mathfrak{r}\}$ and $L^{\mathfrak{T}}(\mathfrak{r}) = (1 : t_{\mathfrak{r}})$ for some $t_{\mathfrak{r}} \in \mathcal{T}(\Sigma, \mathcal{V})$. Hence, we start our rewrite sequence with $\mu_0 = \{1 : t_{\mathfrak{r}}\}$ and our property is satisfied.

For the inductive step, assume that we have already defined the finite rewrite sequence $(\mu_k)_{k\leq i}$ such that for every leaf $x_{(p:t)}$ in $\mathfrak{T}[D(i)]$ that is labeled $L(x_{(p:t)}) = (p : t)$ there is a (unique) pair $(p : t) \in \mu_i$. We now define the $(i+1)$-th step for the rewrite sequence, i.e., we define μ_{i+1}. If $x \in \mathrm{Leaf}^{\mathfrak{T}[D(i)]}$ is also a leaf in $\mathfrak{T}$, then t_x is in normal form w.r.t. $\mathcal{R}$ and we keep the pair $(p_x, t_x) \in \mu_{i+1}$ in the next multi-distribution of our rewrite sequence. If $x \in \mathrm{Leaf}^{\mathfrak{T}[D(i)]}$ is not a leaf in $\mathfrak{T}$, then t_x is not in normal

form w.r.t. $\mathcal{R}$, as $\mathfrak{T}$ satisfies (4.12). Since the edges of the tree, together with the labeling, represent valid $\xrightarrow{\mathrm{i}}_{\mathcal{R}}$ steps, we have

$$t_x \xrightarrow{\mathrm{i}}_{\mathcal{R}} \{\tfrac{p_{y_1}}{p_x} : t_{y_1}, \ldots, \tfrac{p_{y_k}}{p_x} : t_{y_k}\}$$

with $xE = \{y_1, \ldots, y_k\}$. We now set $(p_{y_1} : t_{y_1}) \in \mu_{i+1}$ for all $1 \leq j \leq k$.

It remains to show that our induction hypothesis is still satisfied for μ_{i+1}. Let $x_{(p:t)} \in \mathrm{Leaf}^{\mathfrak{T}[D(i+1)]}$ be a leaf in $\mathfrak{T}[D(i+1)]$ that is labeled $L(x_{(p:t)}) = (p : t)$. If $x_{(p:t)}$ was already a leaf inside the previous tree (i.e., $x_{(p:t)} \in \mathrm{Leaf}^{\mathfrak{T}[D(i)]}$), then we can find a (unique) pair $(p : t) \in \mu_i$ by our induction hypothesis. Since $\mathfrak{T}$ satisfies (4.12) we know that t must be in normal form w.r.t. $\mathcal{R}$ and hence we also get $(p : t) \in \mu_{i+1}$. Otherwise, the node x was not contained in $D(i)$. Let z be the predecessor of x in $\mathfrak{T}$. Since $\mathfrak{T}$ is an innermost $\mathcal{R}$-computation tree, we have

$$t_z \xrightarrow{\mathrm{i}}_{\mathcal{R}} \{\tfrac{p_{y_1}}{p_z} : t_{y_1}, \ldots, \tfrac{p_{y_k}}{p_z} : t_{y_k}\}$$

with $zE = \{y_1, \ldots, y_k\}$ and $x = y_j$ for some $1 \leq j \leq k$. By our induction hypothesis, we find a (unique) pair $(p_z, t_z) \in \mu_i$ and by construction, we have $(p_{y_1} : t_{y_1}) \in \mu_{i+1}$ for all $1 \leq j \leq k$. Hence, our induction hypothesis is again satisfied.

In the end, we have $(\mu_k)_{k\in\mathbb{N}}$ as an infinite $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequence, such that

$$|\mu_\infty|_{\mathcal{R}} = \lim_{i\to\infty} |\mu_i|_{\mathcal{R}} = \lim_{i\to\infty} \sum_{(p:t)\in\mu_i, t\in\mathtt{NF}_{\mathcal{R}}} p = \lim_{i\to\infty} \sum_{x\in\mathrm{Leaf}^{\mathfrak{T}[D(i)]}, t_x\in\mathtt{NF}_{\mathcal{R}}} p_x = |\mathfrak{T}|_{\mathrm{Leaf}}$$

and thus $|\mu_\infty|_{\mathcal{R}} = |\mathfrak{T}|_{\mathrm{Leaf}} < 1$. ■

Finally, we prove our witness theorem for PTRS. This will be done in terms of innermost $\mathcal{R}$-computation trees.

Theorem 4.2.21 (Witness Theorem for PTRS)**.** *If $\mathcal{R}$ is not innermost AST then there exists an innermost $\mathcal{R}$-computation tree $\mathfrak{T}$ that converges with probability < 1 and starts with $(1 : t)$ such that $t = \ell\sigma$ for some rule $\ell \to r \in \mathcal{R}$ and some substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and every proper subterm of $\ell\sigma$ is in normal form w.r.t. $\mathcal{R}$.*

We split the proof of this theorem into several smaller parts. In the non-probabilistic setting, we are able to prove such a witness theorem using a minimality argument. This is not possible anymore for PTRS, as such a minimality argument does not talk about the occurring probabilities so that the minimal term that starts a $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequence that is not AST does not have to be the left-hand side of a rewrite rule.

In the non-probabilistic setting, we know that for a given term $t \in \mathcal{T}(\Sigma, \mathcal{V})$ and a terminating TRS, there can only be a finite number of different rewrite sequences that start with t. This is due to the fact that our TRS only has finitely many rules and that every term only has finitely many positions where we can apply a rewrite rule. A similar statement holds for PTRSs and almost-sure termination. Here, for a given term $t \in \mathcal{T}(\Sigma, \mathcal{V})$, and every $0 < \varepsilon \leq 1$, we can find a natural N_t such that for all $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequences $(\mu_k)_{k\in\mathbb{N}}$ that start with $(1 : t)$, we have $|\mu_{N_t}|_{\mathcal{R}} > 1 - \varepsilon$. This means that the natural number N_t is the same for all $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequences that start with $\{1 : t\}$.

Lemma 4.2.22. *Let $t \in \mathcal{T}(\Sigma, \mathcal{V})$. Assume that every $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequence $(\mu_k)_{k\in\mathbb{N}}$ that starts with $\mu_0 = \{1 : t\}$ converges with probability 1. Then for every $0 < \varepsilon \leq 1$ there is a natural number $N \in \mathbb{N}$ such that $|\mu_N|_{\mathcal{R}} > 1 - \varepsilon$ holds for all such $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequences.*

Proof. Let $t \in \mathcal{T}(\Sigma, \mathcal{V})$. Assume that every $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequence $(\mu_k)_{k\in\mathbb{N}}$ that starts with $\mu_0 = \{1 : t\}$ converges with probability 1. Let $\varepsilon > 0$. We assume for a contradiction that for every natural number $N \in \mathbb{N}$ we can find a $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequence $(\mu_k^{(N)})_{k\in\mathbb{N}}$ that starts with $\mu_0^{(N)} = \{1 : t\}$ such that $|\mu_N^{(N)}|_{\mathcal{R}} \leq 1 - \varepsilon$. Then, we can construct a $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequence $(\mu'_k)_{k\in\mathbb{N}}$ that starts with $\mu'_0 = \{1 : t\}$ such that $|\mu'_N|_{\mathcal{R}} \leq 1 - \varepsilon$ for all $N \in \mathbb{N}$, and hence we have $|\mu'_\infty| \leq 1 - \varepsilon$, which is a contradiction to our assumption that it converges with probability 1.

We inductively construct an infinite tree T that is finitely branching such that every path through this tree represents a valid $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequence that starts with $(1 : t)$. Additionally, we will construct it in a way such that every node inside this tree represents a multi-distribution ν that satisfies $|\nu|_{\mathcal{R}} \leq 1 - \varepsilon$. Let $\mu_0 = \{1 : t\}$ be the root of T. Here, we must have $|\{1 : t\}|_{\mathcal{R}} \leq 1 - \varepsilon$ by our assumption that $|\mu_0^{(0)}|_{\mathcal{R}} \leq 1 - \varepsilon$ and $\mu_0^{(0)} = \{1 : t\}$. In the induction step, assume that we have already constructed the tree up to the i-th depth. For each node ν in the i-th depth, let Z_ν be the set of all possible distributions ν' such that $\nu \overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}} \nu'$. Note that Z_ν is finite, as there are only finitely many different rules that we can apply with finite support, ν itself has only finite support, and there are only finitely many positions for each term in the support of ν that we can use for the rewrite step. We add every distribution $\nu' \in Z_\nu$ with $|\nu'|_{\mathcal{R}} \leq 1 - \varepsilon$ to the $i + 1$-th depth of our tree and add an edge from ν to ν'.

Note that for all $N \in \mathbb{N}$ the N-th depth of the tree is non-empty due to our assumption that for every natural number $N \in \mathbb{N}$ there exists a $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequence that starts with $\mu_0^{(N)} = \{1 : t\}_{k\in\mathbb{N}}$ such that $|\mu_N^{(N)}|_{\mathcal{R}} \leq 1 - \varepsilon$. This implies $|\mu_n^{(N)}|_{\mathcal{R}} \leq 1 - \varepsilon$ for every $n < N$, which means our tree has a path of length N for every $N \in \mathbb{N}$.

Now we have an infinite tree that is finitely branching, which means that the tree has an infinite path by König's Lemma. This path represents our desired $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequence that does not converge with probability 1, which is a contradiction. ■

In terms of computation trees, this means that for a given term $t \in \mathcal{T}(\Sigma, \mathcal{V})$, and every $0 < \varepsilon \leq 1$, we can find a depth $N_t \in \mathbb{N}$ such that for all innermost $\mathcal{R}$-computation trees $\mathfrak{T}$ that start with $(1 : t)$, we have $\sum_{x\in\mathrm{Leaf},d(x)\leq N_t} > 1 - \varepsilon$.

Lemma 4.2.23. *Let $t \in \mathcal{T}(\Sigma, \mathcal{V})$. Assume that every innermost $\mathcal{R}$-computation tree $\mathfrak{T}$ that starts with $\mu_0 = (1 : t)$ converges with probability 1. Then for every $0 < \varepsilon \leq 1$ there is a natural number $N_t \in \mathbb{N}$ such that $\sum_{x\in\mathrm{Leaf}^{\mathfrak{T}},d^{\mathfrak{T}}(x)\leq N_t} > 1 - \varepsilon$ holds for all such innermost $\mathcal{R}$-computation trees $\mathfrak{T}$.*

Proof. Assume for a contradiction that there exists an innermost $\mathcal{R}$-computation tree $\mathfrak{T}_k$ that starts with $\mu_0 = (1 : t)$ and such that $\sum_{x\in\mathrm{Leaf}^{\mathfrak{T}},d^{\mathfrak{T}}(x)\leq k} \leq 1 - \varepsilon$ for every $k \in \mathbb{N}$. For every $k \in \mathbb{N}$, we can create a $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequence $(\mu_i^{(k)})_{i\in\mathbb{N}}$ that starts with $(1 : t)$ such that $|\mu_k^{(k)}|_{\mathcal{R}} \leq 1 - \varepsilon$. Here, we use the construction from the proof of Lemma 4.2.20. But this would mean that there exists a $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequence $(\mu_i^{(k)})_{i\in\mathbb{N}}$ that starts with $(1 : t)$ such that $|\mu_k^{(k)}|_{\mathcal{R}} \leq 1 - \varepsilon$ for every $k \in \mathbb{N}$ and this is a contradiction to Lemma 4.2.22. ■

We can now use this fact for our proof of the witness theorem.

Proof of Theorem 4.2.21. We will prove the contraposition. This proof follows the same structure as the proof for Theorem 2.1.23. In this case, we talk about computation trees instead of rewrite sequences, and the inductive step is a lot harder compared to

Theorem 2.1.23 due to the fact that we are not talking about the existence or absence of an infinite path but about the value of the convergence probability in the limit.

Assume that every innermost $\mathcal{R}$-computation tree $\mathfrak{T}$ that starts with $(1 : t)$ such that $t = \ell\sigma$ for some rule $\ell \to r \in \mathcal{R}$, some substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and where every proper subterm of $\ell\sigma$ is in normal form w.r.t. $\mathcal{R}$, converges with probability 1. We now prove that then also every innermost $\mathcal{R}$-computation tree $\mathfrak{T}$ that starts with $(1 : t)$ for some arbitrary $t \in \mathcal{T}(\Sigma, \mathcal{V})$ converges with probability 1, and thus $\mathcal{R}$ is innermost AST. This will be proven by structural induction over the term t.

If t is in normal form, then obviously every innermost $\mathcal{R}$-computation tree $\mathfrak{T}$ that starts with $(1 : t)$ converges with probability 1. If $t = \ell\sigma$ for some rule $\ell \to r \in \mathcal{R}$ and some substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ such that every proper subterm of $\ell\sigma$ is in normal form w.r.t. $\mathcal{R}$, then we know by our assumption that every innermost $\mathcal{R}$-computation tree $\mathfrak{T}$ that starts with $(1 : t)$ converges with probability 1.

Now we regard the induction step, and assume that $t = f(q_1, \ldots, q_n)$. Here, our induction hypothesis is that every innermost $\mathcal{R}$-computation tree $\mathfrak{T}$ that starts with $(1 : q_i)$ for some $1 \leq i \leq n$ converges with probability 1. Due to the innermost evaluation strategy, we can only rewrite at the root position if every proper subterm is in normal form w.r.t. $\mathcal{R}$. Hence, we will first prove that every innermost $\mathcal{R}$-computation tree $\mathfrak{T}$ that starts with $(1 : t)$ converges with probability 1 assuming that we do not perform rewrite steps at the root position and then allow arbitrary rewrite steps in a second step. To be precise, we first show that every $\xrightarrow{\neg\varepsilon\mathsf{i}}_{\mathcal{R}}$-computation tree $\mathfrak{T}$ that starts with $(1 : t)$ converges with probability 1. Here, an $\xrightarrow{\neg\varepsilon\mathsf{i}}_{\mathcal{R}}$-computation tree is an innermost $\mathcal{R}$-computation tree, where every node, together with its successors and the labeling, represents a rewrite step with $\xrightarrow{\neg\varepsilon\mathsf{i}}_{\mathcal{R}}$. Remember that $\xrightarrow{\neg\varepsilon\mathsf{i}}_{\mathcal{R}}$ denotes the restriction of $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ that prohibits rewrite steps at the root position.

Note that if we rewrite a term q_i to $\{p_1 : q_{i,1}, \ldots, p_k : q_{i,k}\}$, then we get a distribution $\{p_1 : f(q_1, \ldots, q_{i,1}, \ldots, q_n), \ldots, p_k : f(q_1, \ldots, q_{i,k}, \ldots, q_n)\}$. Now, the terms q_j with $j \neq i$ occur multiple times in this distribution, and we may use different rules to rewrite them. Hence, the order in which we rewrite the different q_i matters and cannot be arbitrarily chosen (as seen in Example 4.1.15). Again, the main difference to the proof of Theorem 2.1.23 is that we are working with almost-sure termination instead of sure termination. We now have to estimate the probability of termination instead of proving the non-existence of an infinite path. All in all, we have the following steps for the proof of the induction step:

1) We show that every $\xrightarrow{\neg\varepsilon\mathsf{i}}_{\mathcal{R}}$-computation tree $\mathfrak{T}$ that starts with $(1 : t)$ converges with probability 1. In order to do this, we
   - 1.1) Cut $\mathfrak{T}$ at a specific point to get $\mathfrak{T}_{\mathcal{C}}$
   - 1.2) Partition the leaves $\mathrm{Leaf}^{\mathfrak{T}_{\mathcal{C}}} = Z_{\mathrm{Leaf}} \uplus Z_1 \uplus \ldots \uplus Z_n$
   - 1.3) Prove that the sum of probabilities in Z_i for each $1 \leq i \leq k$ is small enough. In order to do this, we
     - 1.3.1) Transform $\mathfrak{T}_{\mathcal{C}}$ into $\mathfrak{T}_{\mathcal{C}}^{(i)}$
     - 1.3.2) Introduce the general idea of $\mathrm{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)}, i)$
     - 1.3.3) Create $\mathrm{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)}, i)$
2) Then, we additionally allow rewrite steps at the root position and show that every $\mathcal{R}$-computation tree $\mathfrak{T}$ that starts with $(1 : t)$ converges with probability 1.

1) Every $\xrightarrow{\neg\varepsilon\mathsf{i}}_{\mathcal{R}}$-computation tree $\mathfrak{T}$ converges with probability 1.

Assume for a contradiction that there exists a $\xrightarrow{\neg\varepsilon\mathsf{i}}_{\mathcal{R}}$-computation tree $\mathfrak{T} = (V, E, L)$ that starts with $(1 : f(q_1, \ldots, q_n))$ and converges with probability < 1. We now prove that for every $0 < \delta < 1$ we can find an $H \in \mathbb{N}$ such that $\sum_{x \in \mathrm{Leaf}^{\mathfrak{T}}, d(x) \leq H} > 1 - \delta$, which means that $|\mathfrak{T}|_{\mathrm{Leaf}} = \lim_{k \to \infty} \sum_{x \in \mathrm{Leaf}^{\mathfrak{T}}, d(x) \leq k} = 1$, which is our desired contradiction.

Let $0 < \delta < 1$. We define an additional function $\mathtt{pif} : V \setminus \mathrm{Leaf} \to \{1, \ldots, n\}$, called the *position indicator function*, that maps every inner node $x \in V \setminus \mathrm{Leaf}$ to $i \in \{1, \ldots, n\}$ iff the labeling together with the edge relation represents a rewrite step below position i. The position $\mathtt{pif}(x)$ is called the *position indicator* for the inner node x.

We will illustrate the ideas throughout this proof using the following $\mathcal{R}_{rw}$-computation tree $\mathfrak{F}$ for the PTRS $\mathcal{R}_{rw}$ from Example 4.1.4, that does not use rewrite steps at the root position. Here, f is some arbitrary symbol of arity 2.

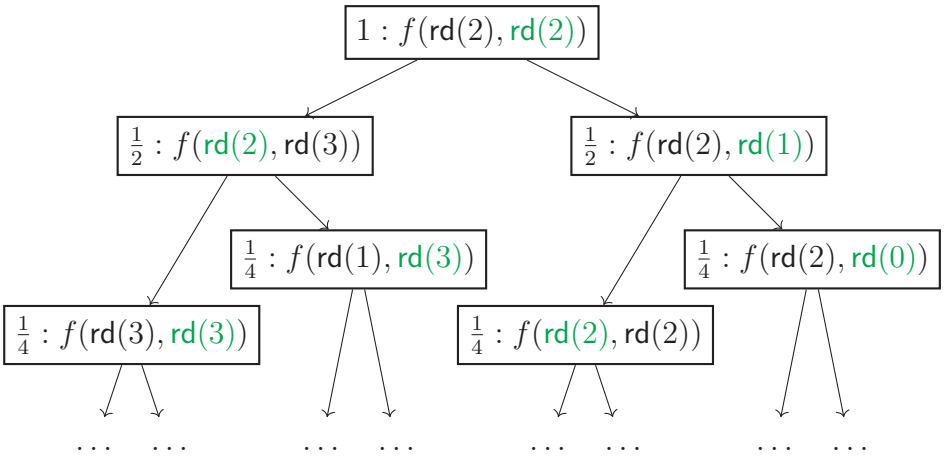


Figure 4.4: Example tree $\mathfrak{F}$. The used redex for every rewrite step is highlighted in green.

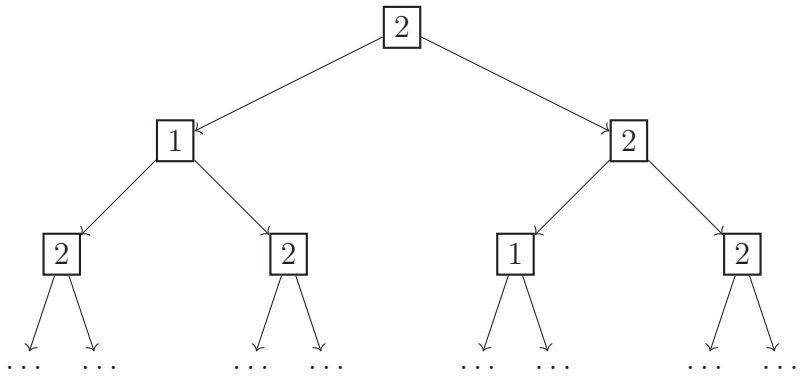


Figure 4.5: Example tree $\mathfrak{F}$ again, but nodes are replaced by their corresponding position indicator.

In Figure 4.4 one can see the actual computation tree and in Figure 4.5, we have the same tree, but we write the corresponding position indicator instead of the labeling for every node.

1.1) Cut the computation tree at a specific point to get $\mathfrak{T}_{\mathcal{C}}$

The first idea is to cut the edges of $\mathfrak{T}$ at specific points so that we result in a finite, grounded induced sub RST $\mathfrak{T}_{\mathcal{C}}$ of some height H. By our induction hypothesis, we know that every

rewrite sequence that starts with $\{1 : q_i\}$ for some $1 \leq i \leq n$ converges with probability 1. Thus, by Lemma 4.2.23 we can find an $N_i \in \mathbb{N}$ such that $\sum_{x \in \mathrm{Leaf}^{\mathfrak{T}}, d^{\mathfrak{T}}(x) \leq N_i} > 1 - \alpha$ with $\alpha \coloneqq \frac{\delta}{n}$ for all innermost $\mathcal{R}$-computation trees $\mathfrak{T}$ that start with $(1 : q_i)$, for all $1 \leq i \leq n$. We will cut the tree $\mathfrak{T}$ after the $(N_1 + 1)$-th node with position indicator 1 on every path. Next, we proceed with this cut tree and repeat the same construction for every $2 \leq i \leq n$. This results in a finite, grounded, induced sub RST $\mathfrak{T}_{\mathcal{C}} = (V_{\mathcal{C}}, E_{\mathcal{C}}, L|_{V_{\mathcal{C}}})$ of some height H, since every number from $\{1, \ldots, n\}$ can only occur finitely often as the position indicator in a path.

In Figure 4.6 one can see the idea of this cut. The tree $\mathfrak{F}_{\mathcal{C}}$ results from $\mathfrak{F}$ from Figures 4.4 and 4.5 after the cut and if we assume for simplicity that $N_1 = 1$ and $N_2 = 1$. Note that the values N_1 and N_2 are not the correct values for this tree, but they are just used to visualize the construction.

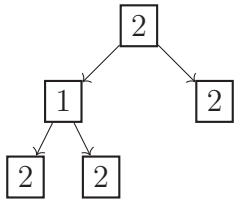


Figure 4.6: $\mathfrak{F}_{\mathcal{C}}$

1.2) Partition the leaves $\mathrm{Leaf}^{\mathfrak{T}_{\mathcal{C}}} = Z_{\mathrm{Leaf}} \uplus Z_1 \uplus \ldots \uplus Z_n$

For all $1 \leq i \leq n$ we define $Z_i \coloneqq \{x \in \mathrm{Leaf}^{\mathfrak{T}_{\mathcal{C}}} \mid \mathtt{pif}(x) = i\}$ to be the set of all leaves that have a position indicator of i. Furthermore, let $Z_{\mathrm{Leaf}} \coloneqq \{x \in \mathrm{Leaf}^{\mathfrak{T}_{\mathcal{C}}} \mid x \in \mathrm{Leaf}^{\mathfrak{T}}\}$ be the set of all leaves that are also leaves in $\mathfrak{T}$. Note that a leaf x in our cut tree $\mathfrak{T}_{\mathcal{C}}$ must either have a position indicator $\mathtt{pif}(x) = i$ for some $1 \leq i \leq n$ and hence $x \in Z_i$, or x has no position indicator, which means that x was already a leaf in $\mathfrak{T}$ and hence $x \in Z_{\mathrm{Leaf}}$. Therefore, we have

$$\mathrm{Leaf}^{\mathfrak{T}_{\mathcal{C}}} = Z_{\mathrm{Leaf}} \uplus Z_1 \uplus \ldots \uplus Z_n \tag{4.13}$$

Since every node $x \in Z_{\mathrm{Leaf}}$ has a depth that is less or equal to H and is a leaf in $\mathfrak{T}$, we have $Z_{\mathrm{Leaf}} \subseteq \{x \in \mathrm{Leaf}^{\mathfrak{T}} \mid \mathrm{d}(x) \leq H\}$, where the subset relation may be a proper one, as we may cut before reaching a leaf of depth at most H. Therefore, we have

$$\sum_{x \in \mathrm{Leaf}^{\mathfrak{T}} \wedge \mathrm{d}(x) \leq H} p_x \geq \sum_{x \in Z_{\mathrm{Leaf}}} p_x \tag{4.14}$$

Furthermore, we have $|\mathfrak{T}_{\mathcal{C}}|_{\mathrm{Leaf}} = \sum_{x \in \mathrm{Leaf}^{\mathfrak{T}_{\mathcal{C}}}} p_x = 1$, since the tree $\mathfrak{T}_{\mathcal{C}}$ is finite. All in all, we get

$$\begin{aligned}
& \textstyle\sum_{x \in \mathrm{Leaf}^{\mathfrak{T}}, d(x) \leq H} p_x \\
\geq\ & \textstyle\sum_{x \in Z_{\mathrm{Leaf}}} p_x \\
& \qquad \text{(by (4.14))} \\
=\ & 1 - \textstyle\sum_{x \in \mathrm{Leaf}^{\mathfrak{T}_{\mathcal{C}}} \setminus Z_{\mathrm{Leaf}}} p_x \\
& \qquad \text{(as } \textstyle\sum_{x \in \mathrm{Leaf}^{\mathfrak{T}_{\mathcal{C}}}} p_x = 1 \text{ and } Z_{\mathrm{Leaf}} \subseteq \mathrm{Leaf}^{\mathfrak{T}_{\mathcal{C}}}\text{)} \\
=\ & 1 - \textstyle\sum_{x \in \bigcup_{1 \leq i \leq n} Z_i} p_x \\
& \qquad \text{(as } \mathrm{Leaf}^{\mathfrak{T}_{\mathcal{C}}} \setminus Z_{\mathrm{Leaf}} = \textstyle\bigcup_{1 \leq i \leq n} Z_i\text{)} \\
=\ & 1 - \textstyle\sum_{1 \leq i \leq n} \sum_{x \in Z_i} p_x \\
& \qquad \text{(as the } Z_i \text{ are all pairwise disjoint)}
\end{aligned}$$

If we are able to show that

$$\sum_{x \in Z_i} p_x < \alpha \tag{4.15}$$

holds for each $1 \leq i \leq n$, then we would get

$$\begin{aligned} & 1 - \sum_{1 \leq i \leq n} \sum_{x \in Z_i} p_x \\ > \; & 1 - \sum_{1 \leq i \leq n} \alpha \\ & \qquad \text{(by (4.15))} \\ = \; & 1 - n \cdot \alpha \\ = \; & 1 - n \cdot \tfrac{\delta}{n} \\ = \; & 1 - \delta, \end{aligned}$$

And thus $\sum_{x \in \mathrm{Leaf}^{\mathfrak{T}}, d(x) \leq H} p_x > 1 - \delta$, which is what we wanted to show. Hence, it remains to prove $\sum_{x \in Z_i} p_x < \alpha$.

1.3) Prove that the sum of probability for each Z_i is small enough

Fix some $1 \leq i \leq n$. We now want to show that $\sum_{x \in Z_i} p_x < \alpha$ (4.15) holds. Remember that Z_i only contains nodes $x \in V_{\mathcal{C}}$ with $\mathtt{pif}(x) = i$ and where the path from the root to x contains $(N_i + 1)$-th times the position indicator i.

1.3.1) Transform $\mathfrak{T}_{\mathcal{C}}$ into $\mathfrak{T}_{\mathcal{C}}^{(i)}$

We start by transforming $\mathfrak{T}_{\mathcal{C}}$ into $\mathfrak{T}_{\mathcal{C}}^{(i)}$. Let $\mathfrak{T}_{\mathcal{C}}^{(i)} = (V_{\mathcal{C}}, E_{\mathcal{C}}, L_{\mathcal{C}}^{(i)})$ be the RST with labeling $L_{\mathcal{C}}^{(i)}(x) = (p_x^{\mathfrak{T}}, t_x^{\mathfrak{T}}|_i)$ for all $x \in V_{\mathcal{C}}$. This means that we have the same tree structure and the same probabilities but only use the subterm at position i for every node.

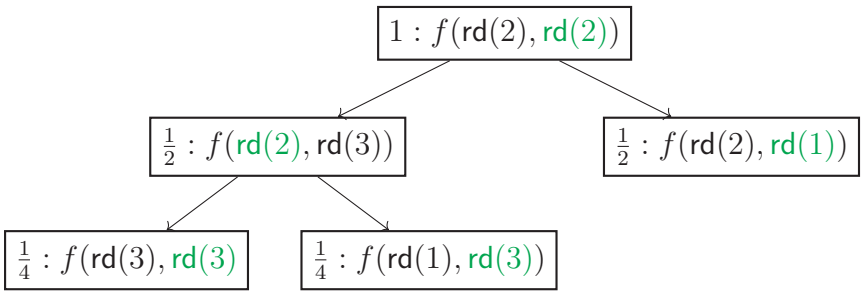


Figure 4.7: $\mathfrak{F}_{\mathcal{C}}$ with the real labeling again.

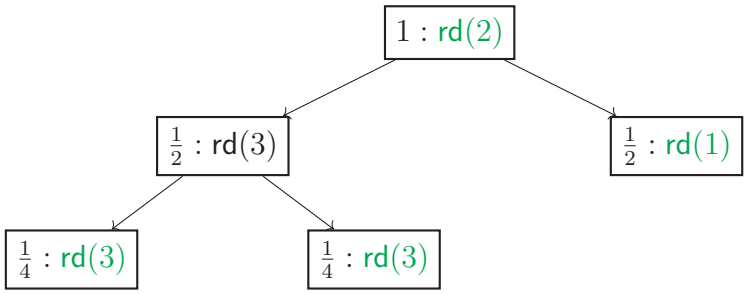


Figure 4.8: $\mathfrak{F}_{\mathcal{C}}^{(2)}$ with adjusted labeling.

As you can see in Figure 4.8, the tree $\mathfrak{T}_{\mathcal{C}}^{(i)}$ does not have to be an innermost $\mathcal{R}$-computation tree anymore. At every node with position indicator $j \neq i$, the subterm at position i stays the same. This means that we have certain nodes that split the same term into multiple ones with a certain probability, and afterward, we can apply different rules to the different copies.

If we assume for simplicity that every node in $\mathfrak{T}_{\mathcal{C}}^{(i)}$ has the position indicator i, which means that we only rewrite at a position below i, then $\mathfrak{T}_{\mathcal{C}}^{(i)}$ is a $\mathcal{R}$-computation tree that

starts with $(1 : q_i)$. Hence, we have

$$\sum_{x \in \mathrm{Leaf}^{\mathfrak{T}}, d^{\mathfrak{T}}(x) \leq N_i} p_x > 1 - \alpha \tag{4.16}$$

by definition of N_i. Furthermore, we have $\mathrm{Leaf}^{\mathfrak{T}_{\mathcal{C}}^{(i)}} = \mathrm{Leaf}^{\mathfrak{T}_{\mathcal{C}}} = Z_{\mathrm{Leaf}} \uplus Z_i$, so that

$$\sum_{x \in Z_i} p_x = 1 - \sum_{x \in Z_{\mathrm{Leaf}}} p_x = 1 - \sum_{x \in \mathrm{Leaf}^{\mathfrak{T}}, d^{\mathfrak{T}}(x) \leq N_i} p_x < 1 - (1 - \alpha) = \alpha$$

However, if we have multiple different position indicators as in Figure 4.6, then $\mathfrak{T}_{\mathcal{C}}^{(i)}$ is not a $\mathcal{R}$-computation tree that starts with $(1 : q_i)$, but we can transform it into multiple ones with a certain probability.

1.3.2) Introduce the general idea of Split$(\mathfrak{T}_{\mathcal{C}}^{(i)}, i)$

In order to show (4.15) in the general case, we use a double counting argument. Instead of directly counting the probabilities for every node $x \in Z_i$, we create a set $\mathrm{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)}, i)$ containing pairs (p_T, T) of a probability $p_T \in (0, 1]$ and an innermost $\mathcal{R}$-computation tree T that start with $(1 : q_i)$. Additionally, T will only contain nodes that have the position indicator i. Then we count the probability for all those trees multiplied by their probability for the nodes in Z_i.

In order to create the set $\mathrm{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)}, i)$, we will iteratively remove the nodes with a position indicator $j \neq i$ from our tree and then split the tree into multiple ones.

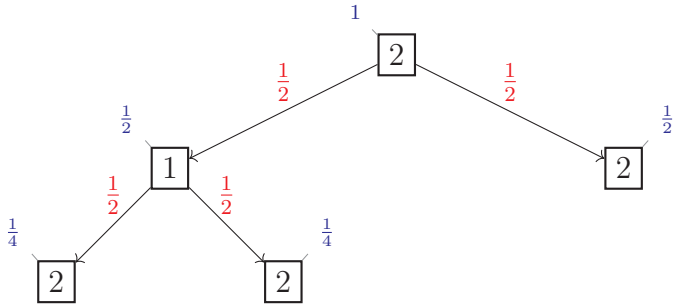
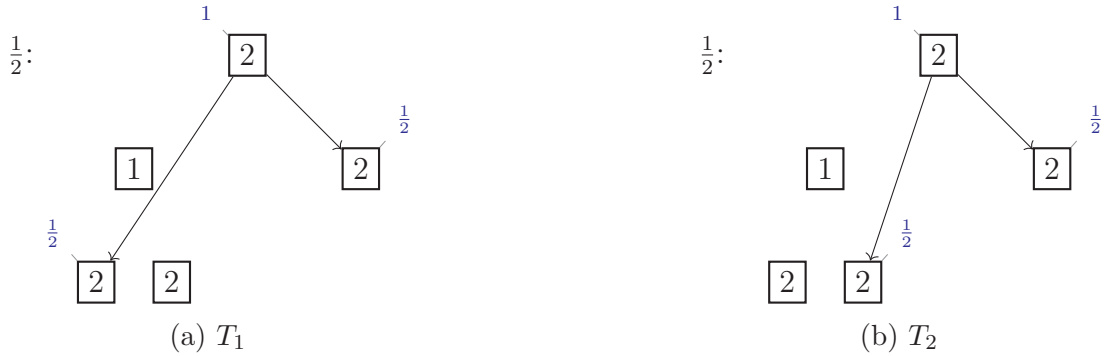


Figure 4.9: The tree $\mathfrak{F}_{\mathcal{C}}^{(2)}$ with the probabilities of the nodes (blue) and the probabilities for the edges (red), implicitly defined by the used rewrite rules.



(a) T_1 (b) T_2

Figure 4.10: The set $\mathrm{Split}(\mathfrak{F}_{\mathcal{C}}^{(2)}, 2) = \{(\frac{1}{2}, T_1), (\frac{1}{2}, T_2)\}$.

In Figure 4.9 you can see the example tree after the cut from before. This time, we added the precise probabilities to the nodes and edges. In order to compute $\sum_{x \in L_2} p_x$ for this

tree, we want to extract the possible innermost $\mathcal{R}$-computation trees that only contain nodes with a position indicator 2 and their corresponding probabilities. In Figure 4.10 one can then see the trees and their probabilities from $\text{Split}(\mathfrak{F}_{\mathcal{C}}, 2)$. For the node with position indicator 1, we have a rewrite step that takes place at a position orthogonal to 2. Hence, this rewrite step is not of any importance for the subterm below position 2, and we can either directly move to the left or right successor. If we do such a split of the tree, we need to extract the probability for this rewrite step out of the tree.

Now to the formal construction of $\text{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)}, i)$. We want to have the following properties:

(1.) $\sum_{(p_T,T)\in\text{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)},i)} p_T = 1$. This means that the sum of the probabilities for all possible innermost $\mathcal{R}$-computation trees in $\text{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)}, i)$ is one.

(2.) For every $(p_T, T) \in \text{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)}, i)$, we have $\sum_{x\in Z_i} \overline{p_{(T,x)}} < \alpha$, where

$$\overline{p_{(T,x)}} = \begin{cases} p_x^T & \text{if } x \in V^T, \\ 0 & \text{otherwise.} \end{cases}$$

This means that for each tree in $\text{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)}, i)$ the probability for all nodes inside of Z_i is strictly smaller than α.

(3.) For all $x \in V_{\mathcal{C}}$ we have $p_x^{\mathfrak{T}_{\mathcal{C}}} = \sum_{(p_T,T)\in\text{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)},i)} p_T \cdot \overline{p_{(T,x)}}$ This means that the probability for node x in our cut tree $\mathfrak{T}_{\mathcal{C}}$ is equal to the sum over all trees T that contain x, where we multiply the probability of the tree T by the probability of node x in T.

If all of these properties are satisfied, then we have

$$\begin{aligned}
& \textstyle\sum_{x\in Z_i} p_x^{\mathfrak{T}_{\mathcal{C}}} \\
= \; & \textstyle\sum_{x\in Z_i} \sum_{(p_T,T)\in\text{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)},i)} p_T \cdot \overline{p_{(T,x)}} \\
& \qquad \text{(by 3.)} \\
= \; & \textstyle\sum_{(p_T,T)\in\text{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)},i)} \sum_{x\in Z_i} p_T \cdot \overline{p_{(T,x)}} \\
= \; & \textstyle\sum_{(p_T,T)\in\text{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)},i)} p_T \cdot \sum_{x\in Z_i} \overline{p_{(T,x)}} \\
< \; & \textstyle\sum_{(p_T,T)\in\text{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)},i)} p_T \cdot \alpha \\
& \qquad \text{(by 2.)} \\
= \; & \textstyle\alpha \cdot \sum_{(p_T,T)\in\text{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)},i)} p_T \\
& \qquad \text{(by 1.)} \\
= \; & \alpha \cdot 1 \\
= \; & \alpha
\end{aligned}$$

so that (4.15) is satisfied and this would end the proof. It remains to construct $\text{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)}, i)$.

1.3.3) Create $\text{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)}, i)$

We will now recursively remove all nodes with a position indicator $j \neq i$ and construct a set M that satisfies the following properties.

(a) $\sum_{(p_T,T)\in M} p_T = 1$.

(b) T is an RST for all $(p_T, T) \in M$.

(c) For all $x \in V_{\mathcal{C}}$ with $\texttt{pif}(x) = i$ we have $p_x^{\mathfrak{T}_{\mathcal{C}}} = \sum_{(p_T,T)\in M} p_T \cdot \overline{p_{(T,x)}}$.

We start with $M \coloneqq \{(1, \mathfrak{T}_{\mathcal{C}}^{(i)})\}$. Here, we clearly have all of the three properties satisfied. Now, assuming that there is still a tree $\mathfrak{t} \in M$ that contains a node $v \in V^{\mathfrak{t}}$ with a position indicator $j \neq i$ that is not a leaf in $\mathfrak{t}$. We will now split $\mathfrak{t}$ into multiple trees that do not contain v anymore but move directly to one of its successors.

First, assume that v is not the root of $\mathfrak{t}$. Let $vE^{\mathfrak{t}} = \{w_1, \dots, w_m\}$ be the direct successors of v in $\mathfrak{t}$ and let z be the predecessor of v in $\mathfrak{t}$. Instead of one tree $\mathfrak{t}$ with the edges $(z, v), (v, w_1), \dots, (v, w_m)$, we split the tree into m different trees $\mathfrak{t}_1, \dots, \mathfrak{t}_m$ such that for every $1 \leq h \leq m$ the tree $\mathfrak{t}_h$ contains a direct edge from z to w_h. In addition to that, the unreachable nodes get removed, and we also have to adjust the probabilities of all (not necessarily direct) successors of w_h (including w_h itself) in $\mathfrak{t}_h$.

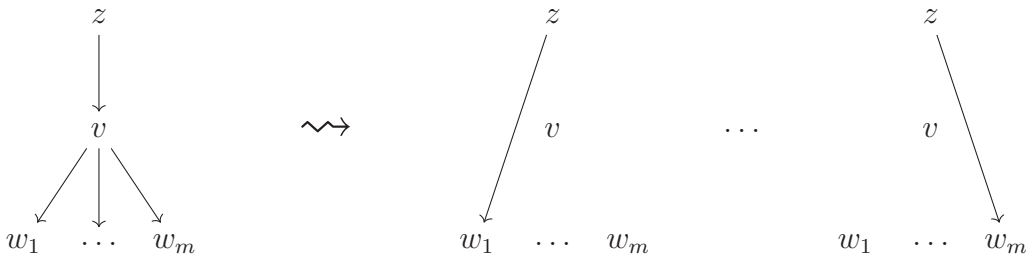


Figure 4.11: Splitting node v to create m-different trees, where we directly move to one of its successors

To be precise, we set $\mathfrak{t}_h \coloneqq (V^{\mathfrak{t}_h}, E^{\mathfrak{t}_h}, L^{\mathfrak{t}_h})$, with

$$V^{\mathfrak{t}_h} \coloneqq (V^{\mathfrak{t}} \setminus v(E^{\mathfrak{t}})^*) \cup w_h(E^{\mathfrak{t}})^*$$
$$E^{\mathfrak{t}_h} \coloneqq (E \setminus (v(E^{\mathfrak{t}})^* \times v(E^{\mathfrak{t}})^*)) \cup \{(z, w_h)\} \cup (E \cap (w_h(E^{\mathfrak{t}})^* \times w_h(E^{\mathfrak{t}})^*))$$

Furthermore, let $p_h \coloneqq \frac{p^{\mathfrak{t}}_{w_h}}{p^{\mathfrak{t}}_v}$. Then, the labeling is defined by

$$L^{\mathfrak{t}_h}(x) = \begin{cases} (\frac{1}{p_h} \cdot p^{\mathfrak{t}}_x, t^{\mathfrak{t}}_x) & x \in w_h(E^{\mathfrak{t}})^*, \\ (p^{\mathfrak{t}}_x, t^{\mathfrak{t}}_x) & \text{otherwise.} \end{cases}$$

Note, that

$$\sum_{1 \leq h \leq m} p_h = \sum_{1 \leq h \leq m} \frac{p^{\mathfrak{t}}_{w_h}}{p^{\mathfrak{t}}_v} = \frac{1}{p^{\mathfrak{t}}_v} \cdot \sum_{1 \leq h \leq m} p^{\mathfrak{t}}_{w_h} = \frac{1}{p^{\mathfrak{t}}_v} \cdot p^{\mathfrak{t}}_v = 1 \tag{4.17}$$

since $\mathfrak{t}$ is an RST by induction hypothesis.

If v is the root of $\mathfrak{t}$, then we do the same construction, but we have no predecessor z of v and directly start with the node w_h as the new root. Hence, we have to use the edge relation

$$E^{\mathfrak{t}_h} \coloneqq (E \cap (w_h(E^{\mathfrak{t}})^* \times w_h(E^{\mathfrak{t}})^*))$$

and the rest stays the same.

In the end, we set

$$M' \coloneqq M \setminus \{(p_{\mathfrak{t}}, \mathfrak{t})\} \cup \{(p_{\mathfrak{t}} \cdot p_1, \mathfrak{t}_1), \dots, (p_{\mathfrak{t}} \cdot p_m, \mathfrak{t}_m)\}$$

This construction is exactly what we are doing in Figure 4.9 and Figure 4.10 for the only node with position indicator 1. It remains to prove that our induction hypothesis is still satisfied for M'.

(a) We have

$$\begin{aligned}
& \textstyle\sum_{(p_T,T)\in M'} p_T \\
= \; & \textstyle\sum_{(p_T,T)\in M\setminus\{(p_\mathfrak{t},\mathfrak{t})\}} p_T + \sum_{(p_T,T)\in\{(p_\mathfrak{t}\cdot p_1,\mathfrak{t}_1),\dots,(p_\mathfrak{t}\cdot p_m,\mathfrak{t}_m)\}} p_T \\
= \; & \textstyle\sum_{(p_T,T)\in M\setminus\{(p_\mathfrak{t},\mathfrak{t})\}} p_T + \sum_{1\le j\le m} p_\mathfrak{t}\cdot p_j \\
= \; & \textstyle\sum_{(p_T,T)\in M\setminus\{(p_\mathfrak{t},\mathfrak{t})\}} p_T + p_\mathfrak{t}\cdot\sum_{1\le j\le m} p_j \\
= \; & \textstyle\sum_{(p_T,T)\in M\setminus\{(p_\mathfrak{t},\mathfrak{t})\}} p_T + p_\mathfrak{t}\cdot 1 \qquad \text{(by (4.17))} \\
= \; & \textstyle\sum_{(p_T,T)\in M\setminus\{(p_\mathfrak{t},\mathfrak{t})\}} p_T + p_\mathfrak{t} \\
= \; & \textstyle\sum_{(p_T,T)\in M} p_T \\
\overset{IH}{=} \; & 1
\end{aligned}$$

(b) Let $1 \le h \le m$. We have to prove that $\mathfrak{t}_h$ is an RST. We constructed $\mathfrak{t}_h$ by skipping the node v in $\mathfrak{t}$ and directly moving from z to w_h (or starting with w_h if v was the root node). Hence, $(V^{\mathfrak{t}_h}, E^{\mathfrak{t}_h})$ is still a finitely branching, directed tree. Let $x \in V^{\mathfrak{t}_h}$ with $xE^{\mathfrak{t}_h} \neq \varnothing$. If $x \in w_h(E^\mathfrak{t})^*$, then $xE^{\mathfrak{t}_h} = xE^\mathfrak{t}$ and thus

$$\sum_{y\in xE^{\mathfrak{t}_h}} p_y^{\mathfrak{t}_h} = \sum_{y\in xE^{\mathfrak{t}_h}} \frac{1}{p_h}\cdot p_y^\mathfrak{t} = \frac{1}{p_h}\cdot\sum_{y\in xE^{\mathfrak{t}_h}} p_y^\mathfrak{t} = \frac{1}{p_h}\cdot\sum_{y\in xE^\mathfrak{t}} p_y^\mathfrak{t} = \frac{1}{p_h}\cdot p_x^\mathfrak{t} = p_x^{\mathfrak{t}_h}$$

If v was not the root and $x = z$, then $zE^{\mathfrak{t}_h} = (zE^\mathfrak{t} \setminus \{v\}) \cup \{w_h\}$ and thus

$$\begin{aligned}
\sum_{y\in zE^{\mathfrak{t}_h}} p_y^{\mathfrak{t}_h} &= \sum_{y\in(zE^\mathfrak{t}\setminus\{v\})\cup\{w_h\}} p_y^{\mathfrak{t}_h} = \sum_{y\in(zE^\mathfrak{t}\setminus\{v\})} p_y^{\mathfrak{t}_h} + p_{w_h}^{\mathfrak{t}_h} = \sum_{y\in(zE^\mathfrak{t}\setminus\{v\})} p_y^\mathfrak{t} + \frac{1}{p_h}\cdot p_{w_h}^\mathfrak{t} \\
&= \sum_{y\in(zE^\mathfrak{t}\setminus\{v\})} p_y^\mathfrak{t} + \tfrac{p_v^\mathfrak{t}}{p_{w_h}^\mathfrak{t}}\cdot p_{w_h}^\mathfrak{t} = \sum_{y\in(zE^\mathfrak{t}\setminus\{v\})} p_y^\mathfrak{t} + p_v^\mathfrak{t} = \sum_{y\in zE^\mathfrak{t}} p_y^\mathfrak{t} = p_z^\mathfrak{t}
\end{aligned}$$

Otherwise, we have $x \in V^\mathfrak{t} \setminus (v(E^\mathfrak{t})^* \cup \{z\})$. This means $p_y^{\mathfrak{t}_h} = p_y^\mathfrak{t}$ for all $y \in xE^{\mathfrak{t}_h}$ and $xE^{\mathfrak{t}_h} = xE^\mathfrak{t}$, and thus

$$\sum_{y\in xE^{\mathfrak{t}_h}} p_y^{\mathfrak{t}_h} = \sum_{y\in xE^{\mathfrak{t}_h}} p_y^\mathfrak{t} = \sum_{y\in xE^\mathfrak{t}} p_y^\mathfrak{t} = p_x^\mathfrak{t} = p_x^{\mathfrak{t}_h}$$

For the last property, note that if v is not the root in $\mathfrak{t}$, then the root and its labeling did not change, so that we have $p_{\mathfrak{r}^{\mathfrak{t}_h}}^{\mathfrak{t}_h} = p_{\mathfrak{r}^\mathfrak{t}}^\mathfrak{t} = 1$. If v was the root, then w_h is the new root with

$$p_{w_h}^{\mathfrak{t}_h} = \frac{1}{p_h}\cdot p_{w_h}^\mathfrak{t} = \tfrac{p_v^\mathfrak{t}}{p_{w_h}^\mathfrak{t}}\cdot p_{w_h}^\mathfrak{t} = \tfrac{1}{p_{w_h}^\mathfrak{t}}\cdot p_{w_h}^\mathfrak{t} = 1$$

(c) Let $x \in V$ with $\mathtt{pif}(x) = i$. If we have $x \notin V^\mathfrak{t}$, then also $x \notin V^{\mathfrak{t}_h}$ for all $1 \le h \le m$ and thus

$$\begin{aligned}
& \textstyle\sum_{(p_T,T)\in M'} p_T\cdot\overline{p_{(T,x)}} \\
= \; & \textstyle\sum_{(p_T,T)\in M\setminus\{(p_\mathfrak{t},\mathfrak{t})\}\cup\{(p_\mathfrak{t}\cdot p_1,\mathfrak{t}_1),\dots,(p_\mathfrak{t}\cdot p_m,\mathfrak{t}_m)\}} p_T\cdot\overline{p_{(T,x)}} \\
= \; & \textstyle\sum_{(p_T,T)\in M\setminus\{(p_\mathfrak{t},\mathfrak{t})\}} p_T\cdot\overline{p_{(T,x)}} + \sum_{(p_T,T)\in\{(p_\mathfrak{t}\cdot p_1,\mathfrak{t}_1),\dots,(p_\mathfrak{t}\cdot p_m,\mathfrak{t}_m)\}} p_T\cdot\overline{p_{(T,x)}} \\
= \; & \textstyle\sum_{(p_T,T)\in M\setminus\{(p_\mathfrak{t},\mathfrak{t})\}} p_T\cdot\overline{p_{(T,x)}} + \sum_{(p_T,T)\in\{(p_\mathfrak{t}\cdot p_1,\mathfrak{t}_1),\dots,(p_\mathfrak{t}\cdot p_m,\mathfrak{t}_m)\}} p_T\cdot 0 \\
= \; & \textstyle\sum_{(p_T,T)\in M\setminus\{(p_\mathfrak{t},\mathfrak{t})\}} p_T\cdot\overline{p_{(T,x)}} \\
= \; & \textstyle\sum_{(p_T,T)\in M\setminus\{(p_\mathfrak{t},\mathfrak{t})\}} p_T\cdot\overline{p_{(T,x)}} + p_\mathfrak{t}\cdot 0 \\
= \; & \textstyle\sum_{(p_T,T)\in M\setminus\{(p_\mathfrak{t},\mathfrak{t})\}} p_T\cdot\overline{p_{(T,x)}} + p_\mathfrak{t}\cdot\overline{p_{(\mathfrak{t},x)}} \\
= \; & \textstyle\sum_{(p_T,T)\in M} p_T\cdot\overline{p_{(T,x)}} \\
\overset{IH}{=} \; & p_x^{\mathfrak{T}_\mathcal{C}}
\end{aligned}$$

If we have $x \in V^{\mathfrak{t}}$ and $x \notin v(E^{\mathfrak{t}})^*$, then $x \in V^{\mathfrak{t}_h}$ for all $1 \leq h \leq m$ and $p_x^{\mathfrak{t}} = p_x^{\mathfrak{t}_h}$, and hence

$$\begin{array}{ll} & \sum_{(p_T,T)\in M'} p_T \cdot \overline{p_{(T,x)}} \\ = & \sum_{(p_T,T)\in M\setminus\{(p_{\mathfrak{t}},\mathfrak{t})\}\cup\{(p_{\mathfrak{t}}\cdot p_1,\mathfrak{t}_1),\ldots,(p_{\mathfrak{t}}\cdot p_m,\mathfrak{t}_m)\}} p_T \cdot \overline{p_{(T,x)}} \\ = & \sum_{(p_T,T)\in M\setminus\{(p_T,T)\}} p_T \cdot \overline{p_{(T,x)}} + \sum_{(p_T,T)\in\{(p_{\mathfrak{t}}\cdot p_1,\mathfrak{t}_1),\ldots,(p_{\mathfrak{t}}\cdot p_m,\mathfrak{t}_m)\}} p_T \cdot \overline{p_{(T,x)}} \\ = & \sum_{(p_T,T)\in M\setminus\{(p_T,T)\}} p_T \cdot \overline{p_{(T,x)}} + \sum_{1\leq h\leq m} p_{\mathfrak{t}} \cdot p_h \cdot p_x^{\mathfrak{t}_h} \\ = & \sum_{(p_T,T)\in M\setminus\{(p_T,T)\}} p_T \cdot \overline{p_{(T,x)}} + \sum_{1\leq h\leq m} p_{\mathfrak{t}} \cdot p_h \cdot p_x^{\mathfrak{t}} \\ = & \sum_{(p_T,T)\in M\setminus\{(p_T,T)\}} p_T \cdot \overline{p_{(T,x)}} + p_{\mathfrak{t}} \cdot p_x^{\mathfrak{t}} \cdot \sum_{1\leq h\leq m} p_h \\ = & \sum_{(p_T,T)\in M\setminus\{(p_T,T)\}} p_T \cdot \overline{p_{(T,x)}} + p_{\mathfrak{t}} \cdot p_x^{\mathfrak{t}} \cdot 1 \\ = & \sum_{(p_T,T)\in M} p_T \cdot \overline{p_{(T,x)}} \\ \overset{IH}{=} & p_x^{\mathfrak{T}_{\mathcal{C}}} \end{array}$$

Otherwise we have $x \in V^{\mathfrak{t}}$ and $x \in v(E^{\mathfrak{t}})^*$. This means that we have $x \in V^{\mathfrak{t}_h}$ and $x \in w_h(E^{\mathfrak{t}})^*$ for some $1 \leq h \leq m$ and $x \notin V^{\mathfrak{t}_g}$ for all $g \neq h$. Furthermore, we have $p_x^{\mathfrak{t}_h} = \frac{1}{p_h} \cdot p_x^{\mathfrak{t}}$, and hence

$$\begin{array}{ll} & \sum_{(p_T,T)\in M'} p_T \cdot \overline{p_{(T,x)}} \\ = & \sum_{(p_T,T)\in M\setminus\{(p_{\mathfrak{t}},\mathfrak{t})\}\cup\{(p_{\mathfrak{t}}\cdot p_1,\mathfrak{t}_1),\ldots,(p_{\mathfrak{t}}\cdot p_m,\mathfrak{t}_m)\}} p_T \cdot \overline{p_{(T,x)}} \\ = & \sum_{(p_T,T)\in M\setminus\{(p_T,T)\}} p_T \cdot \overline{p_{(T,x)}} + \sum_{(p_T,T)\in\{(p_{\mathfrak{t}}\cdot p_1,\mathfrak{t}_1),\ldots,(p_{\mathfrak{t}}\cdot p_m,\mathfrak{t}_m)\}} p_T \cdot \overline{p_{(T,x)}} \\ = & \sum_{(p_T,T)\in M\setminus\{(p_T,T)\}} p_T \cdot \overline{p_{(T,x)}} + p_{\mathfrak{t}} \cdot p_h \cdot p_x^{\mathfrak{t}_h} \\ = & \sum_{(p_T,T)\in M\setminus\{(p_T,T)\}} p_T \cdot \overline{p_{(T,x)}} + p_{\mathfrak{t}} \cdot p_h \cdot \frac{1}{p_h} \cdot p_x^{\mathfrak{t}} \\ = & \sum_{(p_T,T)\in M\setminus\{(p_T,T)\}} p_T \cdot \overline{p_{(T,x)}} + p_{\mathfrak{t}} \cdot p_x^{\mathfrak{t}} \\ = & \sum_{(p_T,T)\in M} p_T \cdot \overline{p_{(T,x)}} \\ \overset{IH}{=} & p_x^{\mathfrak{T}_{\mathcal{C}}} \end{array}$$

In the end, we result with a set M that satisfies (a), (b) and (c), and for every $(p_T, T) \in M$, and all inner nodes $x \in V^T \setminus \mathrm{Leaf}^T$, we have $\mathtt{pif}(x) = i$. Then M is our desired set and we define $\mathrm{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)}, i) \coloneqq M$. It remains to prove that $\mathrm{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)}, i)$ satisfies the conditions (1.), (2.) and (3.). Properties (1.) and (3.) are satisfied due to our induction hypothesis in the end.

To see that (2.) holds, let $(p_T, T) \in \mathrm{Split}(\mathfrak{T}_{\mathcal{C}}^{(i)}, i)$. By our induction hypothesis from the last step, we know that T is an RST. We only have to show that for every $x \in V^T$ with $xE^T = \{y_1, \ldots, y_k\} \neq \varnothing$ we have $t_x^T \xrightarrow{\mathsf{i}}_{\mathcal{R}} \{\frac{p_{y_1}^T}{p_x^T} : t_{y_1}^T, \ldots, \frac{p_{y_k}^T}{p_x^T} : t_{y_k}^T\}$. So let $x \in V^T$ with $xE^T = \{y_1, \ldots, y_k\} \neq \varnothing$. In our construction, we only removed nodes with a position indicator $j \neq i$. Let $xE^{\mathfrak{T}_{\mathcal{C}}^{(i)}} = \{z_1, \ldots, z_k\}$. Since $\mathtt{pif}(x) = i$, we know that

$$t_x^{\mathfrak{T}_{\mathcal{C}}^{(i)}} \xrightarrow{\mathsf{i}}_{\mathcal{R}} \{\frac{p_{z_1}^{\mathfrak{T}_{\mathcal{C}}^{(i)}}}{p_x^{\mathfrak{T}_{\mathcal{C}}^{(i)}}} : t_{z_1}^{\mathfrak{T}_{\mathcal{C}}^{(i)}}, \ldots, \frac{p_{z_k}^{\mathfrak{T}_{\mathcal{C}}^{(i)}}}{p_x^{\mathfrak{T}_{\mathcal{C}}^{(i)}}} : t_{z_k}^{\mathfrak{T}_{\mathcal{C}}^{(i)}}\}$$

If we can show that $\frac{p_{z_h}^{\mathfrak{T}_{\mathcal{C}}^{(i)}}}{p_x^{\mathfrak{T}_{\mathcal{C}}^{(i)}}} = \frac{p_{y_h}^T}{p_x^T}$ and $t_{z_h}^{\mathfrak{T}_{\mathcal{C}}^{(i)}} = t_{y_h}^T$ holds for every $1 \leq h \leq k$, then we are done.

Let $z_h = a_1, \ldots, a_d = y_h$ be the path from z_h to y_h in $\mathfrak{T}_{\mathcal{C}}^{(i)}$. Every node $a_2, \ldots, a_{d-1}$ got removed in our construction so all of them must have a position indicator of $j \neq i$. But this means that we have never used a rewrite step below position i, so that $t_{z_h}^{\mathfrak{T}_{\mathcal{C}}^{(i)}} = t_{y_h}^T$. Furthermore, we extracted all the probabilities for nodes with position indicator $j \neq i$ out of the tree, so that $\frac{p_{z_h}^{\mathfrak{T}_{\mathcal{C}}^{(i)}}}{p_x^{\mathfrak{T}_{\mathcal{C}}^{(i)}}} = \frac{p_{y_h}^T}{p_x^T}$ and this ends the proof for this part.

2) We additionally allow rewrite steps at the root position

Now, we additionally allow rewrite steps at the root position. Again, assume for a contradiction that there exists an innermost $\mathcal{R}$-computation tree $\mathfrak{T} = (V, E, L)$ that starts with $(1 : f(q_1, \ldots, q_n))$ and converges with probability < 1. We again prove that for every $0 < \delta < 1$ we can find an $H \in \mathbb{N}$ such that $\sum_{x \in \mathrm{Leaf}^{\mathfrak{T}}, d(x) \leq H} > 1 - \delta$, which means that $|\mathfrak{T}|_{\mathrm{Leaf}} = \lim_{k \to \infty} \sum_{x \in \mathrm{Leaf}^{\mathfrak{T}}, d(x) \leq k} = 1$, which is our desired contradiction.

Let $0 < \delta < 1$. We start by doing the same construction as above to create a grounded induced sub $\mathcal{R}$-computation tree $\mathfrak{T}_{\mathcal{C}}$, where the labeling does not represent any rewrite steps at the root position such that $|\mathfrak{T}_{\mathcal{C}}|_{Z_{\mathrm{Leaf}}} > 1 - \alpha$ with $\alpha \coloneqq 1 - \sqrt[2]{1 - \delta}$. For each leaf $x \in Z_{\mathrm{Leaf}}$, every proper subterm of t_x is in normal form w.r.t. $\mathcal{R}$. Let us look at the induced sub computation tree T_x of $\mathfrak{T}$ that starts at x (i.e., $T_x = \mathfrak{T}[xE^*]$). Since every proper subterm of t_x is in normal form w.r.t. $\mathcal{R}$, we know by the prerequisite of Theorem 4.2.21 that $|T_x|_{\mathrm{Leaf}} = 1$. Let $\mathrm{d}_x(y)$ be the depth of the node y in the tree T_x. Moreover, let $D_x(k) \coloneqq \{y \in V^{T_x} \mid \mathrm{d}_x(y) \leq k\}$ be the set of nodes in T_x that have a depth of at most k. Since $|T_x|_{\mathrm{Leaf}} = 1$ and $| \circ |_{\mathrm{Leaf}}$ is monotone w.r.t. the depth of the tree T_x, we can find an $N_x \in \mathbb{N}$ such that $|T_x[D_x(N_x)]|_{\mathrm{Leaf}} > 1 - \alpha$.

Note that Z_{Leaf} is finite and thus $N_{\max} = \max\{N_x \mid x \in Z_{\mathrm{Leaf}}\}$ exists. Now, we finally have

$$\begin{aligned}
& \textstyle\sum_{x \in \mathrm{Leaf}^{\mathfrak{T}}, d^{\mathfrak{T}}(x) \leq H + N_{\max}} p_x^{\mathfrak{T}} \\
\geq\ & \textstyle\sum_{x \in Z_{\mathrm{Leaf}}} \sum_{y \in \mathrm{Leaf}^{T_x} \wedge \mathrm{d}_x(y) \leq N_{\max}} p_y^{\mathfrak{T}} \\
=\ & \textstyle\sum_{x \in Z_{\mathrm{Leaf}}} \sum_{y \in \mathrm{Leaf}^{T_x} \wedge \mathrm{d}_x(y) \leq N_{\max}} p_x^{\mathfrak{T}} \cdot p_y^{T_x} \\
=\ & \textstyle\sum_{x \in Z_{\mathrm{Leaf}}} p_x^{\mathfrak{T}} \cdot \sum_{y \in \mathrm{Leaf}^{T_x} \wedge \mathrm{d}_x(y) \leq N_{\max}} p_y^{T_x} \\
\geq\ & \textstyle\sum_{x \in Z_{\mathrm{Leaf}}} p_x^{\mathfrak{T}} \cdot \sum_{y \in \mathrm{Leaf}^{T_x} \wedge \mathrm{d}_x(y) \leq N_x} p_y^{T_x} \\
>\ & \textstyle\sum_{x \in Z_{\mathrm{Leaf}}} p_x^{\mathfrak{T}} \cdot (1 - \alpha) \\
=\ & (1 - \alpha) \cdot \textstyle\sum_{x \in Z_{\mathrm{Leaf}}} p_x^{\mathfrak{T}} \\
>\ & (1 - \alpha) \cdot (1 - \alpha) \\
=\ & (1 - \alpha)^2 \\
=\ & 1 - \delta
\end{aligned}$$

The first inequality holds since every leaf in T_x with a depth of at most $N_{\max}$ (in T_x) for some $x \in Z_{\mathrm{Leaf}}$ must also be a leaf in $\mathfrak{T}$ with a depth of at most $H + N_{\max}$, since x is at a depth of at most H. The last inequality holds since

$$(1 - \alpha)^2 = 1 - \delta \Leftrightarrow 1 - \alpha = \sqrt[2]{1 - \delta} \Leftrightarrow \alpha = 1 - \sqrt[2]{1 - \delta}.$$ ■

Furthermore, we have the same statement regarding ground innermost AST as we had in the non-probabilistic setting with ground innermost termination.

Theorem 4.2.24 (Ground Terms Suffice for PTRSs)**.** *Let $\mathcal{R}$ be a PTRS over the signature Σ. Let $\bot$ be a fresh constant, and h be a fresh unary function symbol. Then $\mathcal{R}$ is innermost AST over the signature Σ iff $\mathcal{R}$ is ground innermost AST over the signature $\Sigma \uplus \{\bot, \mathsf{h}\}$.*

Proof. Remember that we have the following equivalence for non-probabilistic TRSs. Let $t, t' \in \mathcal{T}(\Sigma, \mathcal{V})$ where the term t has the variables $x_1, \ldots, x_m$. Then we have $t \xrightarrow{\mathsf{i}}_{\mathcal{R}} t'$ iff we have $t\sigma \xrightarrow{\mathsf{i}}_{\mathcal{R}} t'\sigma$, where $\sigma(x_i) = \mathsf{h}^i(\bot)$ for all $1 \leq i \leq m$ (i.e., both $t\sigma$ and $t'\sigma$ are ground terms). This equivalence can be extended to PTRSs as follows: Let $\mu, \mu' \in \mathrm{FDist}(\mathcal{T}(\Sigma, \mathcal{V}))$ where the terms in the support of μ contain the variables $x_1, \ldots, x_m$. Then we have $\mu \overset{\mathsf{i}}{\Rightarrow}_{\mathcal{R}} \mu'$ iff we have $\mu\sigma \overset{\mathsf{i}}{\Rightarrow}_{\mathcal{R}} \mu'\sigma$. Here, by $\mu\sigma$ we denote the multi-distribution $\mu\sigma \coloneqq \{p_1 : t_1\sigma, \ldots, p_k : t_k\sigma\}$. By induction, we can then prove that a finitely supported multi-distribution μ with variables $x_1, \ldots, x_m$ starts an infinite innermost rewrite sequence that is not innermost AST iff the multi-distribution $\mu\sigma$, which only contains ground terms, does. ■

4.3 Direct Application of Polynomial Interpretations

We end this chapter by introducing an automatic approach to analyzing innermost AST for a given PTRS. This approach adapts the approach of Theorem 2.2.3 to the probabilistic setting, i.e., it uses a direct application of polynomial interpretations to prove innermost AST. To do so, we have to restrict our polynomial interpretations to be *multilinear* so that they are concave. The property of being concave is needed to lift the inequations of just rewrite rules to arbitrary rewrite steps. The idea for this comes from [2], where they use polynomial interpretations to prove PAST for a given PTRS. Again, even though we restrict our attention to innermost AST, this theorem also works for AST in general.

Theorem 4.3.1 (Proving Innermost AST with Polynomial Interpretations)**.** *Let $\mathcal{R}$ be a PTRS and let $Pol : \mathcal{T}(\Sigma, \mathcal{V}) \to \mathbb{N}[\mathcal{V}]$ be a natural, monotonic, and multilinear polynomial interpretation. Suppose that the following conditions hold for every rule $\ell \to \{p_1 : r_1, \dots, p_k : r_k\} \in \mathcal{R}$:*

(1) *There exists an $1 \leq j \leq k$ with $Pol(\ell) > Pol(r_j)$,*

(2) *$Pol(\ell) \geq \sum_{1 \leq j \leq k} p_j \cdot Pol(r_j)$.*

Then $\mathcal{R}$ is innermost AST.

Proof. By Theorem 4.2.24, it is enough to prove the desired result for ground terms. Note that if $t \in \mathcal{T}(\Sigma)$ then $Pol(t) \in \mathbb{N}$. The main idea for this proof comes from [27]. For this proof, we will work with $\overset{\text{i}}{\rightrightarrows}_{\mathcal{R}}$-rewrite sequences again instead of innermost $\mathcal{R}$-computation trees. The core steps of this proof are the following:

1. We extend the conditions to rewrite steps instead of just rules
2. We create a rewrite sequence $\mu^{\leq N}$ for any $N \in \mathbb{N}$
3. We prove that $|\mu_\infty^{\leq N}|_{\mathcal{R}} \geq p^N$ for any $N \in \mathbb{N}$
4. We prove that $|\mu_\infty^{\leq N}|_{\mathcal{R}} = 1$ for any $N \in \mathbb{N}$
5. Finally, we prove that $|\mu_\infty|_{\mathcal{R}} = 1$

Here, $p > 0$ is the minimal probability that occurs in the rules of $\mathcal{R}$. As $\mathcal{R}$ has only finitely many rules and all occurring multi-distributions have finite support, this minimum is well defined.

1. We extend the conditions to rewrite steps instead of just rules

We first show that the conditions (1) and (2) of the theorem also hold for rewrite steps instead of just rules. Let $s \in \mathcal{T}(\Sigma)$ and $\{p_1 : t_1, \dots, p_k : t_k\} \in FDist(\mathcal{T}(\Sigma))$ with $s \xrightarrow{\text{i}}_{\mathcal{R}} \{p_1 : r_1, \dots, p_k : r_k\}$. We want to show that:

(a) There exists a $1 \leq j \leq k$ with $Pol(s) > Pol(t_j)$,

(b) We have $Pol(s) \geq \sum_{1 \leq j \leq k} p_j \cdot Pol(t_j)$.

By definition of the rewrite relation, there exist a rule $\ell \to \{p_1 : r_1, \dots, p_k : r_k\} \in \mathcal{R}$, a substitution $\sigma \in \text{Sub}(\Sigma, \mathcal{V})$, and a position π of s such that $s|_\pi = \ell\sigma$, every proper subterm of $\ell\sigma$ is in normal form w.r.t. $\mathcal{R}$, and $t_j = s[r_j\sigma]_\pi$ for all $1 \leq j \leq k$.

(a) We perform induction on π. So in the induction base, let $\pi = \varepsilon$. Hence, we have $s = \ell\sigma$ and $t_j = r_j\sigma$ for all $1 \leq j \leq k$. By requirement (1), there is a $1 \leq j \leq k$ with $Pol(\ell) > Pol(r_j)$. As these inequalities hold for all instantiations of the occurring variables, for $t_j = r_j\sigma$, we have

$$Pol(s) = Pol(\ell\sigma) > Pol(r_j\sigma) = Pol(t_j).$$

In the induction step, we have $\pi = i.\pi'$, $s = f(s_1, \ldots, s_i, \ldots, s_n)$, $s_i \to_{\mathcal{R}} \{p_1 : r_{i,1}, \ldots, p_k : r_{i,k}\}$, and $t_j = f(s_1, \ldots, t_{i,j}, \ldots, s_n)$ with $t_{i,j} = s[r_j\sigma]_{\pi'}$ for all $1 \leq j \leq k$. Then by the induction hypothesis there is a $1 \leq j \leq k$ with $Pol(s_i) > Pol(t_{i,j})$. For $t_j = f(s_1, \ldots, t_{i,j}, \ldots, s_n)$ we obtain

$$\begin{aligned} Pol(s) &= Pol(f(s_1, \ldots, s_i, \ldots, s_n)) \\ &= f_{Pol}(Pol(s_1), \ldots, Pol(s_i), \ldots, Pol(s_n)) \\ &> f_{Pol}(Pol(s_1), \ldots, Pol(t_{i,j}), \ldots, Pol(s_n)) \\ &\qquad \text{(by monotonicity of } f_{Pol} \text{ and } Pol(s_i) > Pol(t_{i,j})) \\ &= Pol(f(s_1, \ldots, t_{i,j}, \ldots, s_n)) \\ &= Pol(t_j). \end{aligned}$$

(b) We again perform induction on π. So in the induction base $\pi = \varepsilon$ we again have $s = \ell\sigma$ and $t_j = r_j\sigma$ for all $1 \leq j \leq k$. Then by (2), we know that $Pol(\ell) \geq \sum_{1 \leq j \leq k} p_j \cdot Pol(r_j)$, This inequality holds for all instantiations of the occurring variables. Thus, we obtain

$$Pol(s) = Pol(\ell\sigma) \geq \sum_{1 \leq j \leq k} p_j \cdot Pol(r_j\sigma) = \sum_{1 \leq j \leq k} p_j \cdot Pol(t_j).$$

In the induction step, we have $\pi = i.\pi'$, $s = f(s_1, \ldots, s_i, \ldots, s_n)$, $s_i \to_{\mathcal{R}} \{p_1 : r_{i,1}, \ldots, p_k : r_{i,k}\}$, and $t_j = f(s_1, \ldots, t_{i,j}, \ldots, s_n)$ with $t_{i,j} = s[r_j\sigma]_{\pi'}$ for all $1 \leq j \leq k$. Then by the induction hypothesis we have $Pol(s_i) \geq \sum_{1 \leq j \leq k} p_j \cdot Pol(t_{i,j})$. And thus

$$\begin{aligned} Pol(s) &= Pol(f(s_1, \ldots, s_i, \ldots, s_n)) \\ &= f_{Pol}(Pol(s_1), \ldots, Pol(s_i), \ldots, Pol(s_n)) \\ &\geq f_{Pol}(Pol(s_1), \ldots, \textstyle\sum_{1 \leq j \leq k} p_j \cdot Pol(t_{i,j}), \ldots, Pol(s_n)) \\ &\qquad \text{(by monotonicity of } f_{Pol} \text{ and } Pol(s_i) \geq \textstyle\sum_{1 \leq j \leq k} p_j \cdot Pol(t_{i,j})) \\ &\geq \textstyle\sum_{1 \leq j \leq k} p_i \cdot f_{Pol}(Pol(s_1), \ldots, Pol(t_{i,j}), \ldots, Pol(s_n)), \\ &\qquad \text{(as } f_{Pol} \text{ is componentwise concave due to multilinearity)} \\ &= \textstyle\sum_{1 \leq j \leq k} p_j \cdot Pol(f(s_1, \ldots, t_{i,j}, \ldots, s_n)) \\ &= \textstyle\sum_{1 \leq j \leq k} p_j \cdot Pol(t_j) \end{aligned}$$

where the step regarding the concavity is due to Jensen's inequality, as f_{Pol} is a componentwise concave function due to multilinearity.

2. We create a rewrite sequence $\mu^{\leq N}$ for any $N \in \mathbb{N}$

Due to our restriction to ground terms, we can define the *value* $V(t)$ of any $t \in \mathcal{T}(\Sigma)$.

$$V : \mathcal{T}(\Sigma) \to \mathbb{N}, t \mapsto \begin{cases} Pol(t) + 1, & \text{if } t \in \mathcal{T}(\Sigma) \setminus \mathtt{NF}_{\mathcal{R}} \\ 0, & \text{if } t \in \mathtt{NF}_{\mathcal{R}} \end{cases}$$

and the value of any $\mu \in \mathrm{FDist}(\mathcal{T}(\Sigma))$

$$V : \mathrm{FDist}(\mathcal{T}(\Sigma)) \to \mathbb{R}_{\geq 0}, \mu \mapsto \sum_{(p:t) \in \mu} p \cdot V(t)$$

For any $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequence $(\mu_k)_{k\in\mathbb{N}}$ where $Supp(\mu_0) \subseteq \mathcal{T}(\Sigma)$ and any $N \in \mathbb{N}$, we now consider a modified sequence $(\mu_k^{\leq N})_{k\in\mathbb{N}}$, where we inductively replace each μ_i by $\mu_i^{\leq N}$ starting from μ_0. To obtain $\mu_i^{\leq N}$, we replace $(p : t) \in \mu_i$ by $(p : \top)$ if $V(t) > N$. Otherwise, we keep $(p : t)$. Here, $\top$ is a newly introduced symbol with $V(\top) = N + 1$. If at least one term is replaced in a μ_i, we replace $(\mu_k)_{k>i}$ by a new sequence before proceeding; starting from $\mu_i^{\leq N}$, we use the same rewrite steps as the original $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequence, except for the new normal forms $\top$, which stay the same. Of course, the result is not a valid $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequence anymore.

Our goal is to prove that every $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequence $(\mu_k)_{k\in\mathbb{N}}$ where $\mathrm{Supp}(\mu_0)$ only consists of ground terms converges with probability one, i.e., that $|\mu_\infty|_{\mathcal{R}} = 1$. The proof now proceeds similarly to [27]. First of all, we prove that $|\mu_\infty^{\leq N}|_{\mathcal{R}} = 1$ for any $N \in \mathbb{N}$, i.e., we prove that this new kind of sequence converges with probability one. Note that we now also consider $\top \in \mathtt{NF}_{\mathcal{R}}$. However, this does not yet prove that $|\mu_\infty|_{\mathcal{R}} = 1$, which we will show in a second step.

3. We prove that $|\mu_\infty^{\leq N}|_{\mathcal{R}} \geq p^N$ for any $N \in \mathbb{N}$

Due to (a), for every ground term s that is not in normal form, there is a rewrite step $s \xrightarrow{\mathrm{i}}_{\mathcal{R}} \delta$ such that $\delta(t) \geq p$ and $Pol(s) > Pol(t)$ (and thus, also $V(s) > V(t)$) for some $t \in \mathcal{T}(\Sigma)$. So for any $N \in \mathbb{N}$ and any μ_0, we have $|\mu_N^{\leq N}|_{\mathcal{R}} \geq p^N$. The reason is that for every term in $Supp(\mu_i^{\leq N})$ that is not in normal form, there is always a term in $Supp(\mu_{i+1}^{\leq N})$ which decreases the value of V by at least 1. So for all terms in $Supp(\mu_0^{\leq N})$ that are not in normal form, we reach a normal form in at most N steps with at least probability p^N.

4. We prove that $|\mu_\infty^{\leq N}|_{\mathcal{R}} = 1$ for any $N \in \mathbb{N}$

Note that $|\mu_k^{\leq N}|_{\mathcal{R}}$ is bounded and weakly monotonically increasing, so that $|\mu_\infty^{\leq N}|_{\mathcal{R}}$ must exist and we have $|\mu_\infty^{\leq N}|_{\mathcal{R}} \geq |\mu_N^{\leq N}|_{\mathcal{R}}$. Remember, that we write $|\mu_\infty^{\leq N}|_{\mathcal{R}}$ for $\lim_{k\to\infty} |\mu_k^{\leq N}|_{\mathcal{R}}$ Hence, for any $N \in \mathbb{N}$, we have

$$p_N^\star := \inf_{(\mu_k)_{k\in\mathbb{N}} \text{ is a rewrite sequence}} (|\mu_\infty^{\leq N}|_{\mathcal{R}}) \geq p^N > 0 \tag{4.18}$$

We now prove by contradiction that this is enough to ensure $p_N^\star = 1$. So assume that $p_N^\star < 1$. Then we define $\varepsilon := \frac{p_N^\star \cdot (1-p_N^\star)}{2} > 0$. By definition of the infimum, $p_N^\star + \varepsilon$ is not a lower bound of $|\mu_\infty^{\leq N}|_{\mathcal{R}}$ for all $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequences. Hence, there must exist a $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$-rewrite sequence $(\mu_k)_{k\in\mathbb{N}}$ such that

$$p_N^\star \leq |\mu_\infty^{\leq N}|_{\mathcal{R}} < p_N^\star + \varepsilon. \tag{4.19}$$

By the monotonicity of $|\cdot|$ w.r.t. $\overset{\mathrm{i}}{\Rightarrow}_{\mathcal{R}}$ steps, there must exist a natural number $m^\star \in \mathbb{N}$ such that

$$|\mu_{m^*}^{\leq N}|_{\mathcal{R}} > \tfrac{p_N^\star}{2}. \tag{4.20}$$

Then we have

$$|\mu_\infty^{\leq N}|_{\mathcal{R}} = |\mu_{m^*}^{\leq N}|_{\mathcal{R}} + \sum_{(p:t)\in\mu_{m^*}^{\leq N} \wedge t\notin \mathtt{NF}_{\mathcal{R}}} p \cdot |\{1 : t\}_\infty^{\leq N}|_{\mathcal{R}}, \tag{4.21}$$

where $\{1 : t\}_k^{\leq N}$ denotes the sequence starting with $\{1 : t\}$, where the same rules are applied as for t in $(\mu_k^{\leq N})_{k\geq m^*}$.

Furthermore, we have

$$\sum_{(p:t)\in\mu_{m^*}^{\leq N} \wedge t\notin \mathtt{NF}_{\mathcal{R}}} p = 1 - |\mu_{m^*}^{\leq N}|_{\mathcal{R}} \tag{4.22}$$

So we obtain

$$\begin{aligned}
& p^\star_N + \varepsilon \\
> {} & |\mu^{\leq N}_\infty|_\mathcal{R} \qquad \text{(by (4.19))} \\
= {} & |\mu^{\leq N}_{m^*}|_\mathcal{R} + \sum_{(p:t)\in\mu^{\leq N}_{m^*}\wedge t\notin\mathtt{NF}_\mathcal{R}} p \cdot \underbrace{\lim_{k\to\infty} |\{1:t\}^{\leq N}_k|_\mathcal{R}}_{\geq p^\star_N} \qquad \text{(by (4.21) and (4.18))} \\
\geq {} & |\mu^{\leq N}_{m^*}|_\mathcal{R} + \sum_{(p:t)\in\mu^{\leq N}_{m^*}\wedge t\notin\mathtt{NF}_\mathcal{R}} p \cdot p^\star_N \\
= {} & |\mu^{\leq N}_{m^*}|_\mathcal{R} + p^\star_N \cdot \sum_{(p:t)\in\mu^{\leq N}_{m^*}\wedge t\notin\mathtt{NF}_\mathcal{R}} p \\
= {} & |\mu^{\leq N}_{m^*}|_\mathcal{R} + p^\star_N \cdot (1 - |\mu^{\leq N}_{m^*}|_\mathcal{R}) \qquad \text{(by (4.22))} \\
= {} & p^\star_N + |\mu^{\leq N}_{m^*}|_\mathcal{R} - p^\star_N \cdot |\mu^{\leq N}_{m^*}|_\mathcal{R} \\
= {} & p^\star_N + |\mu^{\leq N}_{m^*}|_\mathcal{R} \cdot (1 - p^\star_N) \\
> {} & p^\star_N + (1 - p^\star_N) \cdot \tfrac{p^\star_N}{2} \qquad \text{(by (4.20))} \\
= {} & p^\star_N + \varepsilon, \quad \lightning
\end{aligned}$$

a contradiction. So $p^\star_N = 1$. In particular, this means that for every $N \in \mathbb{N}$ and every $\xrightarrow{\mathsf{i}}_\mathcal{R}$-rewrite sequence $(\mu_k)_{k\in\mathbb{N}}$ that only consists out of ground terms, we have

$$|\mu^{\leq N}_\infty|_\mathcal{R} = \lim_{k\to\infty} |\mu^{\leq N}_k|_\mathcal{R} = 1. \tag{4.23}$$

5. We prove that $|\mu_\infty|_\mathcal{R} = 1$

Finally, we have to prove that $|\mu_\infty|_\mathcal{R} = \lim_{k\to\infty} |\mu_k|_\mathcal{R} = 1$ holds as well, i.e., that every $\xrightarrow{\mathsf{i}}_\mathcal{R}$-rewrite sequence that only consists out of ground terms converges with probability one. By (b), we have that $\mu \xrightarrow{\mathsf{i}}_\mathcal{R} \mu'$ implies $Pol(\mu) \geq Pol(\mu')$ and therefore, $V(\mu) \geq V(\mu')$. While our modified sequence $(\mu^{\leq N}_k)_{k\in\mathbb{N}}$ is not a rewrite sequence anymore, we still have $V(\mu^{\leq N}_i) \geq V(\mu^{\leq N}_{i+1})$ because either this is a valid $\xrightarrow{\mathsf{i}}_\mathcal{R}$-step or a term with a larger value is replaced by a smaller or equal one, i.e., $\top$. Additionally, $V(\mu^{\leq N}_i)$ is bounded from below by 0, so that $\lim_{k\to\infty}(V(\mu^{\leq N}_k))$ exists and we have

$$V(\mu^{\leq N}_0) \geq \lim_{k\to\infty} (V(\mu^{\leq N}_k)) \tag{4.24}$$

Now we fix $N \in \mathbb{N}$ and a $\xrightarrow{\mathsf{i}}_\mathcal{R}$-rewrite sequence $(\mu_k)_{k\in\mathbb{N}}$, and obtain the corresponding transformed sequence $(\mu^{\leq N}_k)_{k\in\mathbb{N}}$. Note that by (4.23) we have $|\mu^{\leq N}_\infty|_\mathcal{R} = 1 = q_N + (1 - q_N)$, where $q_N = \lim_{k\to\infty} \sum_{(p:t)\in\mu^{\leq N}_k \wedge t\in\mathtt{NF}_\mathcal{R}\wedge t\neq\top} p$. So what is $\lim_{k\to\infty}(V(\mu^{\leq N}_k))$? The probabilities of zero entries (i.e., terms $t \in \mathtt{NF}_\mathcal{R} \setminus \{\top\}$ where $V(t) = 0$) add up to q_N, while the entries $\top$ with value $V(t) \geq N+1$ add up to probability $1 - q_N$. So $\lim_{k\to\infty}(V(\mu^{\leq N}_k))$ has at least the value $q_N \cdot 0 + (1 - q_N) \cdot (N + 1) = (1 - q_N) \cdot (N + 1)$. Thus,

$$V(\mu_0) \geq V(\mu^{\leq N}_0) \geq \lim_{k\to\infty} (V(\mu^{\leq N}_k)) \geq (1 - q_N) \cdot (N + 1),$$

which implies $q_N \geq 1 - \frac{V(\mu_0)}{N+1}$.

Note that q_N is weakly monotonically increasing and bounded from above by 1 for $N \to \infty$. Hence, $q := \lim_{N\to\infty} q_N$ exists and $1 \geq q \geq \lim_{N\to\infty}(1 - \frac{V(\mu_0)}{N+1}) = 1$, i.e., $q = 1$. Hence, we obtain $|\mu_\infty|_\mathcal{R} = \lim_{k\to\infty} |\mu_k|_\mathcal{R} = \lim_{N\to\infty} q_N = q = 1$. So the PTRS $\mathcal{R}$ is innermost AST. ■

Example 4.3.2. Consider the PTRS $\mathcal{R}_{rw}$ from Example 4.1.4. A polynomial interpretation that satisfies (1) and (2) for every rule of $\mathcal{R}_{rw}$ maps $\mathsf{s}(x)$ to $x+1$, $\mathsf{rd}(x)$ to x, and $\mathcal{O}$ to 0. Hence, $\mathcal{R}_{rw}$ is innermost AST.

Example 4.3.3 (Innermost AST with Data Structures)**.** Consider the signature $\Sigma = \{\mathcal{O}, \mathsf{s}, \mathsf{len}, \mathsf{cons}, \mathsf{nil}\}$, a variable set with $\{x, y\} \subseteq \mathcal{V}$, and the following PTRS $\mathcal{R}_{len}$ that is probabilistic variant from the TRS in Example 2.2.5:

$$\mathsf{len}(\mathsf{nil}) \to \{\tfrac{1}{2} : \mathsf{len}(\mathsf{nil}), \tfrac{1}{2} : \mathcal{O}\} \tag{4.25}$$

$$\mathsf{len}(\mathsf{cons}(x, y)) \to \{\tfrac{1}{2} : \mathsf{len}(\mathsf{cons}(x, y)), \tfrac{1}{2} : \mathsf{s}(\mathsf{len}(y))\} \tag{4.26}$$

Here, we again have a chance of $\frac{1}{2}$ to do the actual rewrite step and a chance of $\frac{1}{2}$ to do nothing.

We can prove innermost AST using the same polynomial interpretation as in Example 2.2.5 that maps $\mathsf{len}(x)$ to x, $\mathsf{s}(x)$ to x, $\mathsf{cons}(x, y)$ to $x + y + 1$, and both nil and $\mathcal{O}$ to 0.

5 DP Framework for PTRS

This chapter introduces the first automatic approach to prove innermost almost-sure termination for a given PTRS using dependency pairs. We start by changing our running example regarding the integer division into a probabilistic version, where a direct application of polynomial interpretations to prove AST as in Theorem 4.3.1 is once again not possible. After that, we show why a straightforward adaption of the dependency pairs from the non-probabilistic setting is not possible. The rest of this chapter is then structured like Chapter 3. We first define our new notions of probabilistic dependency pairs in the probabilistic setting in Section 5.1. Instead of dependency pairs, we will work with dependency tuples. Similar to the $\mathcal{R}$-computation tree of a PTRS $\mathcal{R}$, we then define how a $(\mathcal{P},\mathcal{S})$-computation tree for our probabilistic DP problem $(\mathcal{P},\mathcal{S})$ looks like and directly define our probabilistic notion of a chain using those trees in Section 5.2. We can then prove the chain criterion for the probabilistic chains in Section 5.3. In the end, in Section 5.4, we talk about the general structure of the probabilistic dependency pair framework and adapt the previously introduced processors to the probabilistic setting. In addition to the three processors from the non-probabilistic setting, we will also introduce two new processors, namely the *usable pairs processor* and the *not probabilistic processor*. The *not probabilistic processor* transforms a probabilistic DP problem into a non-probabilistic DP problem if the probabilities are of a trivial form. Then, we can use our existing framework for the non-probabilistic setting after this transformation. The *usable pairs processor* can be seen as an extension to the dependency graph processor that is needed due to the fact that we are working with dependency tuples instead of dependency pairs. Hence, this processor was not needed in the non-probabilistic setting.

Example 5.0.1 (PTRS which Requires Dependency Pairs)**.** Consider the signature $\Sigma = \{\mathcal{O}, \mathsf{s}, \mathsf{minus}, \mathsf{div}\}$, a variable set with $\{x, y\} \subseteq \mathcal{V}$, and the following PTRS $\mathcal{R}_{div}$:

$$\mathsf{minus}(x,\mathcal{O}) \to \{\tfrac{1}{2} : \mathsf{minus}(x,\mathcal{O}), \tfrac{1}{2} : x\} \tag{5.1}$$
$$\mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)) \to \{\tfrac{1}{2} : \mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)), \tfrac{1}{2} : \mathsf{minus}(x,y)\} \tag{5.2}$$
$$\mathsf{div}(\mathcal{O},\mathsf{s}(y)) \to \{\tfrac{1}{2} : \mathsf{div}(\mathcal{O},\mathsf{s}(y)), \tfrac{1}{2} : \mathcal{O}\} \tag{5.3}$$
$$\mathsf{div}(\mathsf{s}(x),\mathsf{s}(y)) \to \{\tfrac{1}{2} : \mathsf{div}(\mathsf{s}(x),\mathsf{s}(y)), \tfrac{1}{2} : \mathsf{s}(\mathsf{div}(\mathsf{minus}(x,y),\mathsf{s}(y)))\} \tag{5.4}$$

We now have a chance of $\frac{1}{2}$ to do the actual rewrite step and a chance of $\frac{1}{2}$ to do nothing (i.e., the terms stay the same). We will see that this PTRS is AST using our new probabilistic DP framework. The reason for this is that the underlying TRS is terminating, and with every step, we have a certain probability ($\frac{1}{2}$) to make a step "towards termination" and in the other case, we do nothing. Similar to the non-probabilistic setting, a direct application of polynomial interpretations as in Theorem 4.3.1 is not possible. In order to satisfy the constraints of Theorem 4.3.1, we would need to find a natural, monotonic, and multilinear polynomial interpretation *Pol*, such that for every rule $\ell \to r \in \mathcal{R}_{div}$ there is at least one term in the support of the right-hand side that has a strictly smaller value than the

left-hand side. But this is not possible for (5.4):

$$\mathsf{div}(\mathsf{s}(x),\mathsf{s}(y)) \to \{\tfrac{1}{2} : \mathsf{div}(\mathsf{s}(x),\mathsf{s}(y)), \tfrac{1}{2} : \mathsf{s}(\mathsf{div}(\mathsf{minus}(x,y),\mathsf{s}(y)))\}$$

The first term in the support of the right-hand side, namely $\mathsf{div}(\mathsf{s}(x),\mathsf{s}(y))$, is equal to the left-hand side, so it has the same polynomial value. Additionally, the second term in the support of the right-hand side is $\mathsf{s}(\mathsf{div}(\mathsf{minus}(x,y),\mathsf{s}(y)))$ and we have seen in Example 3.0.1, that there is no natural and monotonic polynomial interpretation Pol such that

$$Pol(\mathsf{div}(\mathsf{s}(x),\mathsf{s}(y))) > Pol(\mathsf{s}(\mathsf{div}(\mathsf{minus}(x,y),\mathsf{s}(y))))$$

Next, we investigate the different possibilities to define the dependency pairs in the probabilistic setting and what challenges occur. For a term $t \in \mathcal{T}(\Sigma,\mathcal{V})$, let $\mathrm{Sub}_D(t) \coloneqq \{r \mid r \text{ is a (not necessarily proper) subterm of } t \text{ with defined root symbol}\}$ be the set of all its subterms with defined root symbol. In the non-probabilistic case, for a rule $\ell \to r$ with $\mathrm{Sub}_D(r) = \{r_1,\dots,r_m\}$, one obtains the m dependency pairs $\ell^\# \to r_i^\#$ with $1 \le i \le m$. Therefore, two natural ideas to define dependency pairs would be as follows.

Definition 5.0.2 (Options for Dependency Pairs)**.** Let $\mathcal{R}$ be a PTRS. For any rule $\ell \to \{p_1 : r_1,\dots,p_k : r_k\} \in \mathcal{R}$, we define the set of its *dependency pairs* as either A:

$$\{\, \ell^\# \to \{p_1 : r_1,\dots,p_i : t_i^\#,\dots,p_k : r_k\} \mid t_i \in \mathrm{Sub}_D(r_i) \text{ with } 1 \le i \le k \,\}.$$

Or B:

$$\{\, \ell^\# \to \{p_1 : t_1^\#,\dots,p_k : t_k^\#\} \mid t_i \in \mathrm{Sub}_D(r_i) \text{ for all } 1 \le i \le k \,\}.$$

If $\mathrm{Sub}_D(r_i) = \varnothing$, then we insert a constructor $\bot$ into $\mathrm{Sub}_D(r_i)$ that does not occur in the left-hand side of some rewrite rule in $\mathcal{R}$.

In both these definitions, we replace a term with only a single dependency term in the right-hand side. The following example shows that this is undesirable. Consider the following two PTRSs $\mathcal{R}_1$ and $\mathcal{R}_2$ over the signature $\{\mathsf{f},\mathsf{g},\bot\}$, where f and $\bot$ are constructors and there is the following rule to reduce g:

$$\mathcal{R}_1 : \mathsf{g} \to \{\tfrac{1}{2} : \mathsf{f}(\mathsf{g},\mathsf{g}), \tfrac{1}{2} : \bot\}$$
$$\mathcal{R}_2 : \mathsf{g} \to \{\tfrac{1}{2} : \mathsf{f}(\mathsf{g},\mathsf{g},\mathsf{g}), \tfrac{1}{2} : \bot\}$$

$\mathcal{R}_1$ is AST since it corresponds to a fair random walk stopping at 0, where the number of gs in a term denotes the current position. $\mathcal{R}_2$ is not AST as it corresponds to a random walk stopping at 0, where there is an equal chance of reducing the value by 1 or increasing the value by 2. For both $\mathcal{R}_1$ and $\mathcal{R}_2$, both of the definitions of dependency pairs from above would yield the only dependency pair:

$$\mathsf{g}^\# \to \{\tfrac{1}{2} : \mathsf{g}^\#, \tfrac{1}{2} : \bot\}.$$

This dependency pair is clearly AST since it corresponds to a program that flips a coin until head comes up once and then terminates. So these definitions would not yield a sound approach for proving AST, as they result in the same dependency pairs for a PTRS that is AST and a PTRS that is not AST.

One might get the impression that the problem is due to "sibling" terms (since the gs in our example are at orthogonal positions). However, this is not the case. Consider

$$\mathcal{R}_3 : \mathsf{g}(x) \to \{\tfrac{1}{2} : \mathsf{g}(\mathsf{g}(\mathsf{g}(x))), \tfrac{1}{2} : \bot\}$$

which would result in the following three different dependency pairs.

$$\mathsf{g}^{\#}(x) \to \{\tfrac{1}{2} : \mathsf{g}^{\#}(\mathsf{g}(\mathsf{g}(x))), \tfrac{1}{2} : \bot\}$$
$$\mathsf{g}^{\#}(x) \to \{\tfrac{1}{2} : \mathsf{g}^{\#}(\mathsf{g}(x)), \tfrac{1}{2} : \bot\}$$
$$\mathsf{g}^{\#}(x) \to \{\tfrac{1}{2} : \mathsf{g}^{\#}(x), \tfrac{1}{2} : \bot\}$$

If one considers an innermost evaluation and counts the numbers of gs in a term, $\mathcal{R}_3$ corresponds to a random walk where the value is increased by 2 with probability $\frac{1}{2}$ and the value is decreased by 1 with probability $\frac{1}{2}$. However, every infinite innermost chain with the dependency pairs above would converge with probability 1 since in every step with a dependency pair, one immediately reaches $\bot$ with probability $\frac{1}{2}$. So nested terms need to be treated in a similar way to sibling terms.

In fact, variant A in Definition 5.0.2 is not only unsound, but it would also be unsuitable for termination analysis because satisfying the constraints of the reduction pair processor would be similar to the task of satisfying the constraints for proving AST directly with polynomial orderings. For instance, the rule (5.4) would yield two dependency pairs, where one of them is $\mathsf{div}^{\#}(\mathsf{s}(x), \mathsf{s}(y)) \to \{\frac{1}{2} : \mathsf{div}^{\#}(\mathsf{s}(x), \mathsf{s}(y)), \frac{1}{2} : \mathsf{s}(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y)))\}$. Orienting this dependency pair with the reduction pair processor is essentially the same as orienting the original rule with a polynomial ordering due to the fact that we need to compare the term $\mathsf{div}^{\#}(\mathsf{s}(x), \mathsf{s}(y))$ with a tuple symbol at the root, with the original right-hand side $\mathsf{s}(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y)))$ of the rule.

It is also tempting to only regard single terms of a distribution in isolation, i.e., as in Variant A, which only allows one term on the right-hand side to contain tuple symbols. Then one might be further tempted to define dependency pairs as in Variant A, but replacing all terms not containing a tuple symbol by $\bot$. But that would even be unsound for PTRSs, where all terms on the right-hand side only contain a single occurrence of a defined symbol. This can be seen with the PTRS $f(\mathsf{s}(x)) \to \{\frac{3}{4} : f(\mathsf{s}(\mathsf{s}(x))), \frac{1}{4} : f(x)\}$ which is not AST. But if one only created the dependency pairs $f^{\#}(\mathsf{s}(x)) \to \{\frac{3}{4} : f^{\#}(\mathsf{s}(\mathsf{s}(x))), \frac{1}{4} : \bot\}$ and $f^{\#}(\mathsf{s}(x)) \to \{\frac{3}{4} : \bot, \frac{1}{4} : f^{\#}(x)\}$, and there is no useful definition of chains which does not converge with probability one in this example.

All of these examples show that when considering dependency pairs for a PTRS, we do not only have to look at the different occurring defined symbols in the distribution on the right-hand side on their own, but we need to regard them all together and also take the number of their occurrences into account. Therefore, instead of dependency pairs, we have to work with dependency tuples. Here, we couple all of the defined symbols in the right-hand side of a rule $\ell \to r$ with $\mathrm{Sub}_D(r) = \{r_1, \ldots, r_m\}$ together and create a single dependency tuple $\ell^{\#} \to \mathsf{Com}_m(r_1^{\#}, \ldots, r_m^{\#})$. This means that we deal with a single dependency tuple instead of possibly m different dependency pairs. Here, Com_m is a fresh constructor symbol of arity m. Dependency tuples are already used in the analysis of TRSs, e.g., for automatic complexity analysis [29].

Definition 5.0.3 (Options for Dependency Tuples)**.** Let $\mathcal{R}$ be a PTRS. For any rule $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{R}$, we define the set of its *dependency tuples* as C:

$$\{\, \ell^{\#} \to \{p_1 : \mathsf{Com}_{m_1}(t_{1,1}^{\#}, \ldots, t_{1,m_1}^{\#}), \ldots, p_k : \mathsf{Com}_{m_k}(t_{k,1}^{\#}, \ldots, t_{k,m_k}^{\#})\} \mid \mathrm{Sub}_D(r_i) = \{t_{i,1}, \ldots, t_{i,m_i}\} \text{ with } 1 \le i \le k \,\}.$$

The straightforward idea for the definition of chains would be that they result from an alternation of one step with a dependency tuple and then an arbitrary amount of steps with $\mathcal{R}$, where the latter would be expressed by the relation $\xrightarrow{\mathsf{i}}{}^{*}_{\mathcal{R}}$. However, then the

problem would be that in such chains, the distributions introduced by rewrite steps with dependency tuples are independent of the distributions introduced by the rewrite steps with $\mathcal{R}$.

Example 5.0.4 (Problem with Dependency Tuples)**.** To illustrate the problem, consider a signature with $\Sigma_D = \{\mathsf{f}, \mathsf{a}, \mathsf{b}, \mathsf{c}\}$ and $\Sigma_C = \{\mathcal{O}\}$, and a PTRS $\mathcal{R}_{tup}$ with the rules $\mathsf{f}(\mathcal{O}) \to \{1 : \mathsf{f}(\mathsf{a})\}$ and $\mathsf{a} \to \{\frac{1}{2} : \mathsf{b}, \frac{1}{2} : \mathsf{c}\}$, and let there also be further rules for all defined symbols. Let $\mathcal{DT}(\mathcal{R}_{tup})$ be the set of its dependency tuples defined as in Definition 5.0.3. The dependency tuple for the f-rule is

$$\mathsf{f}^{\#}(\mathcal{O}) \to \{1 : \mathsf{Com}_2(\mathsf{f}^{\#}(\mathsf{a}), \mathsf{a}^{\#})\}$$

and the dependency tuple for the a-rule is

$$\mathsf{a}^{\#} \to \{\tfrac{1}{2} : \mathsf{Com}_1(\mathsf{b}^{\#}), \tfrac{1}{2} : \mathsf{Com}_1(\mathsf{c}^{\#})\}$$

Consider the rewrite sequence

$$\{1 : \mathsf{f}(\mathcal{O})\} \overset{\mathsf{i}}{\Rightarrow}_{\mathcal{R}_{tup}} \{1 : \mathsf{f}(\mathsf{a})\} \overset{\mathsf{i}}{\Rightarrow}_{\mathcal{R}_{tup}} \{\tfrac{1}{2} : \mathsf{f}(\mathsf{b}), \tfrac{1}{2} : \mathsf{f}(\mathsf{c})\}$$

When constructing the corresponding chain, we get

$$\begin{array}{ll}
 & \{\ 1 : \mathsf{f}^{\#}(\mathcal{O})\} \\
\overset{\mathsf{i}}{\Rightarrow}_{\mathcal{DT}(\mathcal{R}_{tup})} & \{\ 1 : \mathsf{Com}_2(\mathsf{f}^{\#}(\mathsf{a}), \mathsf{a}^{\#})\} \\
\overset{\mathsf{i}}{\Rightarrow}_{\mathcal{DT}(\mathcal{R}_{tup})} & \{\ \tfrac{1}{2} : \mathsf{Com}_2(\mathsf{f}^{\#}(\mathsf{a}), \mathsf{Com}_1(\mathsf{b}^{\#})), \tfrac{1}{2} : \mathsf{Com}_2(\mathsf{f}^{\#}(\mathsf{a}), \mathsf{Com}_1(\mathsf{c}^{\#}))\} \\
\overset{\mathsf{i}}{\Rightarrow}_{\mathcal{R}_{tup}} & \{\ \tfrac{1}{4} : \mathsf{Com}_2(\mathsf{f}^{\#}(\mathsf{b}), \mathsf{Com}_1(\mathsf{b}^{\#})), \tfrac{1}{4} : \mathsf{Com}_2(\mathsf{f}^{\#}(\mathsf{c}), \mathsf{Com}_1(\mathsf{b}^{\#})), \\
 & \quad \tfrac{1}{4} : \mathsf{Com}_2(\mathsf{f}^{\#}(\mathsf{b}), \mathsf{Com}_1(\mathsf{c}^{\#})), \tfrac{1}{4} : \mathsf{Com}_2(\mathsf{f}^{\#}(\mathsf{c}), \mathsf{Com}_1(\mathsf{c}^{\#}))\}
\end{array}$$

This is undesired since the red terms in the last distribution do not correspond to terms in the original rewrite sequence. This would also be problematic for the reduction pair processor later on since these undesired terms are not guaranteed to decrease w.r.t. the polynomial ordering. So we have to ensure that the other "copies" of the redex are reduced in the same way when rewriting with a dependency tuple.

5.1 Dependency Pairs

In order to create a sound and complete method for an automatic AST analysis, we have to adjust the definition of a dependency tuple. We create a new *coupled positional dependency tuple*, where we couple a *positional dependency tuple* together with the original rewrite rule that was used to create it. A positional dependency tuple is a dependency tuple where we additionally store the position in the original right-hand side of each subterm with defined root symbol. This means that we are not only dealing with terms but work with pairs (t, π) consisting of a term t and a position π. The additionally stored original rewrite rule is used to solve the problem indicated in Example 5.0.4 so that we can rewrite every "copy" of the redex in the same way. This then results in a sound chain criterion since we can mirror every rewrite sequence that was possible with the PTRS $\mathcal{R}$ with our coupled dependency tuples. The positions are used to create a sound chain criterion that is also complete. Currently, the processors do not use the positions in any way, so they are really only needed for the completeness of the chain criterion. The right-hand side of a positional dependency tuple no longer consists purely of terms. Hence, we decided to use sets instead of Com_m-terms to combine them. In the following, we precisely define the coupled positional dependency tuples and explain how it solves the problem described in Example 5.0.4.

Definition 5.1.1 (Defined Symbols)**.** Let $\mathcal{R}$ be a PTRS over the signature Σ. Defined symbols, constructor symbols, tuple symbols and dependency terms are defined as in Definition 3.1.1 and Definition 3.1.5. Moreover, for every term $t \in \mathcal{T}(\Sigma, \mathcal{V})$, we define the set $\mathrm{Sub}^{\#}_{DPos}(t) = \{(r^{\#}, \pi) \mid r \text{ is a (not necessarily proper) subterm of } t \text{ at position } \pi \text{ with defined root symbol}\}$ that is the set of all subterms with defined root symbol, where we replaced the root symbol with the corresponding tuple symbol, together with their corresponding position inside of t.

Example 5.1.2 (Defined Symbols)**.** Consider the PTRS $\mathcal{R}_{div}$ from Example 5.0.1. Here, we have $\Sigma_C = \{\mathcal{O}, \mathsf{s}\}$, $\Sigma_D = \{\mathsf{div}, \mathsf{minus}\}$, and

$$\mathrm{Sub}^{\#}_{DPos}(\mathsf{s}(\mathsf{div}(\mathsf{minus}(x,y),\mathsf{s}(y)))) = \{\Big(\mathsf{div}^{\#}(\mathsf{minus}(x,y),\mathsf{s}(y)),1\Big),\Big(\mathsf{minus}^{\#}(x,y),1.1\Big)\}.$$

In this chapter, we use PARSs that act on sets instead of terms. This means that we are dealing with sets twice. On the one hand, we use multisets for our distributions. Here, we have to use a rewrite rule on every term in the support of the distribution that is not already in normal form. On the other hand, we use ordinary sets inside of our distributions that we are actually rewriting. To distinguish between those two types of sets, we will call the first one a *distribution multiset* and the second one a *term representation set*. For distribution multisets, we use typical brackets $\{\circ\}$, and for term representation sets, we use $\{\!\{\circ\}\!\}$. Whenever we are using sets (or multisets) that neither correspond to a distribution nor a term (e.g., a set like $\mathrm{Sub}^{\#}_{DPos}(t)$), then we also use the standard notation $\{\circ\}$.

We will work with sets from $\mathrm{Pot}(\mathcal{T}^{\#}(\Sigma) \times \mathbb{N}^*)$. Every standard definition regarding terms (e.g., substitutions) will be lifted to sets $A \in \mathrm{Pot}(\mathcal{T}^{\#}(\Sigma) \times \mathbb{N}^*)$ in the obvious way, where we apply the definition to all terms inside of this set, if not mentioned otherwise. For example, if we have $A \in \mathrm{Pot}(\mathcal{T}^{\#}(\Sigma) \times \mathbb{N}^*)$, then for a substitution σ we define $A\sigma \coloneqq \sigma(A) \coloneqq \{(\sigma(t), \pi) \mid (t, \pi) \in A\}$.

Now, let us come to the *dp transformation* that generates the right-hand sides of our positional dependency tuples. This simply is equal to $\mathrm{Sub}^{\#}_{DPos}(t)$ due to the fact that we are working with sets.

Definition 5.1.3 (Transformation dp)**.** Let $t \in \mathcal{T}(\Sigma, \mathcal{V})$. We define $dp(t) \coloneqq \mathrm{Sub}^{\#}_{DPos}(t)$. Similarly, for any $\mu = \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathrm{FDist}(\mathcal{T}(\Sigma, \mathcal{V}))$, we define

$$dp(\mu) \coloneqq \{p_1 : dp(r_1), \ldots, p_k : dp(r_k)\}$$

For a rule $\ell \to \mu = \{p_1 : r_1, \ldots, p_k : r_k\}$ we set

$$dp(\ell \to \mu) = \ell^{\#} \to dp(\mu)$$

Example 5.1.4 (Transformation dp)**.** Consider the term $t \coloneqq \mathsf{s}(\mathsf{div}(\mathsf{minus}(x,y),\mathsf{s}(y)))$. Then we have $dp(t) = \{\!\{\Big(\mathsf{div}^{\#}(\mathsf{minus}(x,y),\mathsf{s}(y)),1\Big),\Big(\mathsf{minus}^{\#}(x,y),1.1\Big)\}\!\}$. And for the rule $\mathsf{minus}(x,\mathcal{O}) \to \mu = \{\frac{1}{2} : \mathsf{minus}(x,\mathcal{O}), \frac{1}{2} : x\}$ we have

$$dp(\mathsf{minus}(x,\mathcal{O}) \to \mu) = \mathsf{minus}^{\#}(x,\mathcal{O}) \to \{\frac{1}{2} : \{\!\{\Big(\mathsf{minus}^{\#}(x,\mathcal{O}),\varepsilon\Big)\}\!\}, \frac{1}{2} : \varnothing\}$$

Note that the set $dp(t)$ will be used within our computation and has a corresponding semantic, while the set $\mathrm{Sub}^{\#}_{DPos}(t)$ is only a definition. This is the reason why we use the term representation set notation for dp and the typical set notation for $\mathrm{Sub}^{\#}_{DPos}(t)$ even though they are syntactically the same.

In the non-probabilistic version of a chain, we worked with dependency terms. In the probabilistic setting, we want to work with *well-positioned positional dependency tuple sets*. These are sets that can be seen as a set representation of a term t using only the subterms with defined root symbol and their position inside of t.

Definition 5.1.5 (Positional Dependency Tuple Sets and Well-Positioned Sets)**.** Let $\mathsf{PDTS} \subseteq \mathrm{Pot}(\mathcal{T}^{\#}(\Sigma) \times \mathbb{N}^*)$ be the set of all finite sets that have the form

$$t = \{\!\!\{\left(t_1^{\#}, \pi_1\right), \ldots, \left(t_n^{\#}, \pi_n\right)\}\!\!\}$$

such that $t_i^{\#} \in \mathcal{T}^{\#}(\Sigma)$ and $\pi_i \in \mathbb{N}^*$ for all $i \in [1, n]$.

We call $A \in \mathsf{PDTS}$ *well-positioned* iff there exists a ground term $t \in \mathcal{T}(\Sigma)$ with $A \subseteq dp(t)$. The set of all well-positioned positional dependency tuple sets is

$$\mathsf{PDTS}_{\mathsf{wp}} := \{A \in \mathsf{PDTS} \mid A \text{ is well-positioned}\}$$

Example 5.1.6 (Positional Dependency Tuple Sets)**.** Let $\Sigma = \{\mathsf{f}, \mathsf{g}\}$ and assume that both f and g are defined symbols. First, consider the two sets

$$A = \{\!\!\{\left(\mathsf{f}(\mathsf{g},\mathsf{g}), \varepsilon\right), \left(\mathsf{g}^{\#}, 1\right), \left(\mathsf{g}^{\#}, 2\right)\}\!\!\}, \quad A' = \{\!\!\{\left(\mathsf{f}^{\#}(\mathsf{g}^{\#},\mathsf{g}^{\#}), \varepsilon\right), \left(\mathsf{g}^{\#}, 1\right), \left(\mathsf{g}^{\#}, 2\right)\}\!\!\}$$

Here, we have $A \notin \mathsf{PDTS}$ as $f(\mathsf{g},\mathsf{g}) \notin \mathcal{T}^{\#}(\Sigma)$ since it does not contain a tuple symbol at the root and $A' \notin \mathsf{PDTS}$ as $\mathsf{f}^{\#}(\mathsf{g}^{\#},\mathsf{g}^{\#}) \notin \mathcal{T}^{\#}(\Sigma)$ since it contains a tuple symbol at a non-root position. In contrast, we have

$$A'' = \{\!\!\{\left(\mathsf{f}^{\#}(\mathsf{g},\mathsf{g}), \varepsilon\right), \left(\mathsf{g}^{\#}, 1\right), \left(\mathsf{g}^{\#}, 2\right)\}\!\!\} \in \mathsf{PDTS}$$

Example 5.1.7 (Well-Positioned Sets)**.** Again, consider the signature $\Sigma = \{\mathsf{f}, \mathsf{g}\}$ from Example 5.1.6. The set

$$A = \{\!\!\{\left(\mathsf{f}^{\#}(\mathsf{g},\mathsf{g}), 1\right), \left(\mathsf{g}^{\#}, 1.1\right), \left(\mathsf{g}^{\#}, 1.2\right)\}\!\!\} \in \mathsf{PDTS}$$

is well-positioned, since $A \subseteq dp(\mathsf{f}(\mathsf{f}(\mathsf{g},\mathsf{g}),\mathsf{g}))$. On the other hand, we have

$$A' = \{\!\!\{\left(\mathsf{g}^{\#}, 1\right), \left(\mathsf{f}^{\#}(\mathsf{g},\mathsf{g}), 1\right)\}\!\!\} \in \mathsf{PDTS}$$

not well-positioned as we have two different pairs in A', namely $\left(\mathsf{g}^{\#}, 1\right)$ and $\left(\mathsf{f}^{\#}(\mathsf{g},\mathsf{g}), 1\right)$, with the same position and it is not possible to have two different subterms at the same position in a ground term $t \in \mathcal{T}(\Sigma)$. Also the set

$$A'' = \{\!\!\{\left(\mathsf{f}^{\#}(\mathsf{g},\mathsf{g}), \varepsilon\right), \left(\mathsf{g}^{\#}, 1\right), \left(\mathsf{f}^{\#}(\mathsf{g},\mathsf{g}), 2\right)\}\!\!\} \in \mathsf{PDTS}$$

is not well-positioned as we have $\left(\mathsf{f}^{\#}(\mathsf{g},\mathsf{g}), \varepsilon\right), \left(\mathsf{f}^{\#}(\mathsf{g},\mathsf{g}), 2\right) \in A$ and $\varepsilon.2 = 2$ but $\mathsf{f}^{\#}(\mathsf{g},\mathsf{g})|_2 \neq \mathsf{f}(\mathsf{g},\mathsf{g})$. And this is not possible for well-positioned sets by Lemma 5.1.8.

In Example 5.0.4 we saw that we have to rewrite the "copies" of a term in the same way as the term itself. The positions in our well-positioned sets indicate what other terms contain such a copy and where we can find them.

Lemma 5.1.8 (Properties of Well-Positioned Sets)**.** *Let* $A \in \mathsf{PDTS}_{\mathsf{wp}}$.

- *If we have* $(t_1, \pi_1), (t_2, \pi_2) \in A$ *such that there is a position* $\chi \in \mathbb{N}^+$ *with* $\pi_1.\chi = \pi_2$*, then we have* $t_1|_\chi = t_2^\flat$
- *For all* $B \subseteq A$ *we have* $B \in \mathsf{PDTS}_{\mathsf{wp}}$

Proof. Let $t \in \mathcal{T}(\Sigma)$ such that $A \subseteq dp(t)$. Then we have

$$t_1|_\chi \overset{\chi \neq \varepsilon}{=} t_1^\flat|_\chi \overset{t_1^\flat = t|_{\pi_1}}{=} (t|_{\pi_1})|_\chi = t|_{\pi_1.\chi} \overset{\pi_1.\chi = \pi_2}{=} t|_{\pi_2} \overset{t|_{\pi_2} = t_2^\flat}{=} t_2^\flat$$

Additionally, for every $B \subseteq A$ we have $B \subseteq A \subseteq dp(t)$ and thus $B \subseteq dp(t)$, so B is well-positioned as well. ■

Positional dependency tuples are still not expressive enough to create our desired dependency pair framework. We are now able to locate the copies of a term, but we are yet unable to determine how we should rewrite the copies since our positional dependency tuples only contain information about the defined symbols and everything below a defined symbol, but not about the whole original rewrite rule. The idea for building dependency pairs for the probabilistic setting is to define a new rewrite relation that works with rules operating on *pairs*. Instead of having a rule $\ell \to \{p_1 : r_1, \dots, p_k : r_k\}$ from $\mathcal{R}$ and its corresponding positional dependency tuple $\ell^\# \to \{p_1 : dp(r_1), \dots, p_k : dp(r_k)\}$ separately, we couple them together and write them as $(\ell^\#, \ell) \to \{p_1 : (dp(r_1), r_1), \dots, p_k : (dp(r_k), r_k)\}$. This type of rewriting system is called a *probabilistic pair term rewriting system (PPTRS)*, and the rules are called *coupled positional dependency tuples*. For simplicity, we will also call them (probabilistic) dependency tuples again. Instead of working with a single PTRS for proving almost-sure termination, our dependency pair framework works on pairs $(\mathcal{P}, \mathcal{S})$, where $\mathcal{P}$ is a PPTRS, and $\mathcal{S}$ is a PTRS, like in the non-probabilistic setting. We again call this pair $(\mathcal{P}, \mathcal{S})$ a (probabilistic) DP problem. Now, we can give the definition of the dependency tuples for a given PTRS.

Definition 5.1.9 (Coupled Positional Dependency Tuple)**.** Let $\mathcal{R}$ be a PTRS. Then for every $\ell \to \mu = \{p_1 : r_1, \dots, p_k : r_k\} \in \mathcal{R}$ we define the *coupled positional dependency tuple* (or simply dependency tuple) as

$$\mathcal{DT}(\ell \to \mu) \coloneqq (\ell^\#, \ell) \to \{p_1 : (dp(r_1), r_1), \dots, p_k : (dp(r_k), r_k)\}$$

The set of all coupled positional dependency tuples $\mathcal{DT}(\mathcal{R})$ is then defined as

$$\mathcal{DT}(\mathcal{R}) \coloneqq \{\mathcal{DT}(\ell \to \mu) \mid \ell \to \mu \in \mathcal{R}\}.$$

Example 5.1.10 (Coupled Positional Dependency Tuple)**.** Consider $\mathcal{R}_{div}$ from Example 5.0.1. Here, we get $\mathcal{DT}(\mathcal{R}_{div}) = \{\mathcal{DT}((5.1)), \mathcal{DT}((5.2)), \mathcal{DT}((5.3)), \mathcal{DT}((5.4))\}$ with

$$\begin{aligned}
\mathcal{DT}((5.1)) = {} & \Big(\mathsf{minus}^\#(x, \mathcal{O}), \mathsf{minus}(x, \mathcal{O})\Big) \\
& \to \{ \tfrac{1}{2} : \Big(\{\!\!\{\Big(\mathsf{minus}^\#(x, \mathcal{O}), \varepsilon\Big)\}\!\!\}, \mathsf{minus}(x, \mathcal{O})\Big), \\
& \qquad \tfrac{1}{2} : \Big(\varnothing, x\Big)\} \\
\mathcal{DT}((5.2)) = {} & \Big(\mathsf{minus}^\#(\mathsf{s}(x), \mathsf{s}(y)), \mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y))\Big) \\
& \to \{ \tfrac{1}{2} : \Big(\{\!\!\{\Big(\mathsf{minus}^\#(\mathsf{s}(x), \mathsf{s}(y)), \varepsilon\Big)\}\!\!\}, \mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y))\Big), \\
& \qquad \tfrac{1}{2} : \Big(\{\!\!\{\Big(\mathsf{minus}^\#(x, y), \varepsilon\Big)\}\!\!\}, \mathsf{minus}(x, y)\Big)\} \\
\mathcal{DT}((5.3)) = {} & \Big(\mathsf{div}^\#(\mathcal{O}, \mathsf{s}(y)), \mathsf{div}(\mathcal{O}, \mathsf{s}(y))\Big) \\
& \to \{ \tfrac{1}{2} : \Big(\{\!\!\{\Big(\mathsf{div}^\#(\mathcal{O}, \mathsf{s}(y)), \varepsilon\Big)\}\!\!\}, \mathsf{div}(\mathcal{O}, \mathsf{s}(y))\Big), \\
& \qquad \tfrac{1}{2} : \Big(\varnothing, \mathcal{O}\Big)\} \\
\mathcal{DT}((5.4)) = {} & \Big(\mathsf{div}^\#(\mathsf{s}(x), \mathsf{s}(y)), \mathsf{div}(\mathsf{s}(x), \mathsf{s}(y))\Big) \\
& \to \{ \tfrac{1}{2} : \Big(\{\!\!\{\Big(\mathsf{div}^\#(\mathsf{s}(x), \mathsf{s}(y)), \varepsilon\Big)\}\!\!\}, \mathsf{div}(\mathsf{s}(x), \mathsf{s}(y))\Big), \\
& \qquad \tfrac{1}{2} : \Big(\{\!\!\{\Big(\mathsf{div}^\#(\mathsf{minus}(x, y), \mathsf{s}(y)), 1\Big), \Big(\mathsf{minus}^\#(x, y), 1.1\Big)\}\!\!\}, \\
& \qquad\qquad\qquad \mathsf{s}(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y)))\Big)\}
\end{aligned}$$

Next, we will precisely define the notion of a PPTRS and how a PPTRS $\mathcal{P}$ rewrites sets in $\mathsf{PDTS}_{\mathsf{wp}}$ w.r.t. $\mathcal{S}$. Additionally, we also have to define how the PTRS $\mathcal{S}$ rewrites sets in

$\mathsf{PDTS}_{\mathsf{wp}}$, as this will be different from the ordinary rewrite relation of a PTRS for terms. Beforehand, we look at our example from before and show what we are trying to archive with this new PPTRS.

Example 5.1.11 (Resolving the Problem with Dependency Tuples)**.** Consider the PTRS $\mathcal{R}_{tup}$ from Example 5.0.4 and the rewrite sequence

$$\{1 : \mathsf{f}(\mathsf{s})\} \overset{\mathsf{i}}{\Rightarrow}_{\mathcal{R}_{tup}} \{1 : \mathsf{f}(\mathsf{a})\} \overset{\mathsf{i}}{\Rightarrow}_{\mathcal{R}_{tup}} \{\tfrac{1}{2} : \mathsf{f}(\mathsf{b}), \tfrac{1}{2} : \mathsf{f}(\mathsf{c})\}$$

again. The coupled positional dependency tuple for the f-rule is

$$(\mathsf{f}^{\#}(\mathsf{s}), \mathsf{f}(\mathsf{s})) \to \{1 : \Big(\{\!\{\big(\mathsf{f}^{\#}(\mathsf{a}), \varepsilon\big), \big(\mathsf{a}^{\#}, 1\big)\}\!\}, \mathsf{f}(\mathsf{a})\Big)\}$$

and the coupled positional dependency tuple for the a-rule is

$$(\mathsf{a}^{\#}, \mathsf{a}) \to \{\tfrac{1}{2} : \Big(\{\!\{\big(\mathsf{b}^{\#}, \varepsilon\big)\}\!\}, \mathsf{b}\Big), \tfrac{1}{2} : \Big(\{\!\{\big(\mathsf{c}^{\#}, \varepsilon\big)\}\!\}, \mathsf{c})\Big)\}$$

With these two dependency tuples, we get the following rewrite sequence:

$$\begin{array}{ll} & \{\quad 1 : \{\!\{\big(\mathsf{f}^{\#}(\mathsf{s}), \varepsilon\big)\}\!\}\} \\ \overset{\mathsf{i}}{\Rightarrow}_{\mathcal{DT}(\mathcal{R}_{tup}),\mathcal{R}_{tup}} & \{\quad 1 : \{\!\{\big(\mathsf{f}^{\#}(\mathsf{a}), \varepsilon\big), \big(\mathsf{a}^{\#}, 1\big)\}\!\}\} \\ \overset{\mathsf{i}}{\Rightarrow}_{\mathcal{DT}(\mathcal{R}_{tup}),\mathcal{R}_{tup}} & \{\quad \tfrac{1}{2} : \{\!\{\big(\mathsf{f}^{\#}(\mathsf{b}), \varepsilon\big), \big(\mathsf{b}^{\#}, 1\big)\}\!\}, \tfrac{1}{2} : \{\!\{\big(\mathsf{f}^{\#}(\mathsf{c}), \varepsilon\big), \big(\mathsf{c}^{\#}, 1\big)\}\!\}\} \end{array}$$

In the second step, we are able to simultaneously rewrite the $\mathsf{a}^{\#}$ together with its copy inside of $\mathsf{f}^{\#}(\mathsf{a})$. The additional positions are used to detect the "copies" of a term so that we can rewrite them in the same way as the term itself. When rewriting the term $\mathsf{a}^{\#}$ in the second rewrite step, the corresponding position 1 indicates that the pair $\big(\mathsf{f}^{\#}(\mathsf{a}), \varepsilon\big)$ contains a copy of a, since $\varepsilon < 1$. The two positions also indicate, where this copy is, namely at the position 1 of $\mathsf{f}^{\#}(\mathsf{a})$, since $\varepsilon.1 = 1$. We can then use the original rewrite rule that is contained in our coupled positional dependency tuple on the term $\mathsf{f}^{\#}(\mathsf{a})$ in order to rewrite the a accordingly. All in all, using this new rewrite relation, we have no undesired terms in our final distribution and can completely mirror the original rewrite sequence.

Definition 5.1.12 (Pair Rewriting, PPTRS)**.** A set of rules $\mathcal{P}$ of the form

$$(\ell^{\#}, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\}$$

is called a *probabilistic pair term rewriting system (PPTRS)* iff the *projection* to the second component $\mathcal{P}_2 = \{\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \mid (\ell^{\#}, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in \mathcal{P}\}$ is a PTRS, and we have $C_j \subseteq dp(r_j)$ for all $1 \leq j \leq k$. We denote the projection to the PTRS rules as $\mathrm{proj}_2(\mathcal{P}) \coloneqq \mathcal{P}_2$. Finally, let $\mathcal{S}$ be a PTRS.

The rewrite relation of a PPTRS operates on sets $A \in \mathsf{PDTS}_{\mathsf{wp}}$. A well-positioned set A rewrites with the PPTRS $\mathcal{P}$ to $\{p_1 : B_1, \ldots, p_k : B_k\}$ w.r.t. $\mathcal{S}$ (denoted $\overset{\mathsf{i}}{\to\!\!\triangleright}_{\mathcal{P},\mathcal{S}}$) iff there is a dependency tuple $(\ell^{\#}, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in \mathcal{P}$, a pair $(t, \pi) \in A$, called the *main rewrite pair*, and a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ such that $t = \ell^{\#}\sigma$, and every proper subterm of $\ell^{\#}\sigma$ is in normal form w.r.t. $\mathcal{S}$. Then, the following conditions hold for all $1 \leq j \leq k$:

- Let

$$\begin{aligned} M_{rew} &\coloneqq \{\!\{(a, \tau) \in A \mid \tau < \pi\}\!\} \\ M_{\perp} &\coloneqq \{\!\{(a, \tau) \in A \mid \tau \perp \pi\}\!\} \\ M_{<} &\coloneqq \{\!\{(a, \tau) \in A \mid \pi < \tau\}\!\} \end{aligned}$$

  Note that we have

$$A = \{\!\{(t, \pi)\}\!\} \uplus M_{rew} \uplus M_{\perp} \uplus M_{<},$$

- The main rewrite pair $\{\!\{(t,\pi)\}\!\}$ gets replaced by $M_j^+ := \{\!\{(r'\sigma, \pi.\pi') \mid (r',\pi') \in C_j\}\!\}$.
- For all $(a,\tau) \in M_{rew}$ there exists a $\chi_a \in \mathbb{N}^+$ such that $\tau.\chi_a = \pi$. Since A is well-positioned, we get $a|_{\chi_a} = t^\flat = \ell\sigma$ by Lemma 5.1.8. Then we can rewrite a to $a[r_j\sigma]_{\chi_a}$ using the substitution σ, the position χ_a and the rewrite rule $\ell \to r_j$, since all proper subterms of $\ell\sigma$ are in normal form w.r.t. $\mathcal{S}$.
- Finally, we have two possibilities: If we have $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{S}$, then

$$B_j \coloneqq M_j^+ \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \tau) \mid (a,\tau) \in M_{rew}\}\!\} \uplus M_\bot.$$

  otherwise, if the rule is not in $\mathcal{S}$, then

$$B_j \coloneqq M_j^+ \uplus M_\bot.$$

The last property essentially means that if the second component of a dependency tuple is not contained in the PTRS $\mathcal{S}$, then we are not allowed to use it. So the second components that are not contained in $\mathcal{S}$ are completely useless, and we could remove them. We keep them here for readability so that we do not have to deal with a third type of rewrite rule, where the second component is deleted.

The rewrite relation of a PPTRS is based on the rewrite relation of a PTRS on terms, as we are rewriting well-positioned sets that represent terms. Let us shortly explain the different sets that occur in this definition. Assume that we perform a rewrite step using the dependency tuple $(\ell^\#, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\}$, main rewrite pair (t,π), and ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$.

- M_j^+ is the set that results from applying the positional dependency tuple $\ell^\# \to \{p_1 : C_1, \ldots, p_k : C_k\}$ to the main rewrite pair $(t,\pi) \in A$. Here, we append the new position inside of C_j to the existing position π.
- M_{rew} is the set of all pairs (a,τ) that needs to be rewritten as well, indicated by the additional position. The set $\{\!\{(a[r_j\sigma]_{\chi_a}, \tau) \mid (a,\tau) \in M_{rew}\}\!\}$ then contains the correctly rewritten terms, where we applied the rule $\ell \to r_j$ at the correct position and using the same substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$. We omit this set if the rule $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\}$ is not in $\mathcal{S}$. This means that only our additional PTRS $\mathcal{S}$ indicates whether we are allowed to apply ordinary rewrite rules or not.
- $M_\bot$ contains all (a,τ) that have a position orthogonal to π and therefore stay the same.
- $M_<$ contains all (a,τ) that have a position below π. This set will be removed so that our resulting set is well-positioned again. Note that $a^\flat$ must be in normal form w.r.t. $\mathcal{S}$ as we are dealing with innermost evaluation, so we are only removing normal forms w.r.t. $\mathcal{S}$ in this way.

Example 5.1.13 (Rewriting with $\xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}}$)**.** Consider the following PTRS $\mathcal{S}$ over the signature $\Sigma = \Sigma_D = \{\mathsf{f}, \mathsf{g}, \mathsf{a}, \mathsf{b}, \mathsf{c}\}$:

$$\mathcal{S}: \ \mathsf{f}(\mathsf{a}) \to \{1 : \mathsf{g}(\mathsf{b})\}$$

The dependency tuple for this rule has the form:

$$\mathcal{P}: \ (\mathsf{f}^\#(\mathsf{a}), \mathsf{f}(\mathsf{a})) \to \{1 : \Big(\{\!\{\big(\mathsf{g}^\#(\mathsf{b}), \varepsilon\big), \big(\mathsf{b}^\#, 1\big)\}\!\}, \mathsf{g}(\mathsf{b})\Big)\}$$

Consider the well-positioned set

$$A \coloneqq \{\!\{\left(\mathsf{c}^{\#}(\mathsf{f}(\mathsf{a}),\mathsf{f}(\mathsf{a})),\varepsilon\right),\left(\mathsf{f}^{\#}(\mathsf{a}),2\right),\left(\mathsf{f}^{\#}(\mathsf{a}),1\right),\left(\mathsf{a}^{\#},1.1\right)\}\!\}$$

A is well-positioned, since $A \subseteq dp\big(\mathsf{c}(\mathsf{f}(\mathsf{a}),\mathsf{f}(\mathsf{a}))\big)$. Let us take a closer look at the following rewrite step:

$$A \xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}} \{1 : A'\} = \{1 : \{\!\{\left(\mathsf{c}^{\#}(\mathsf{g}(\mathsf{b}),\mathsf{f}(\mathsf{a})),\varepsilon\right),\left(\mathsf{f}^{\#}(\mathsf{a}),2\right),\left(\mathsf{g}^{\#}(b),1\right),\left(b^{\#},1.1\right)\}\!\}\}$$

The resulting set is well-positioned again, since $A' \subseteq dp(\mathsf{c}(\mathsf{g}(\mathsf{b}),\mathsf{f}(\mathsf{a})))$. Here, we use the main rewrite pair $\left(\mathsf{f}^{\#}(\mathsf{a}),1\right)$, the identity substitution id and the only dependency tuple. Note that every proper subterm of $\mathsf{f}^{\#}(\mathsf{a})$ is in normal form w.r.t. $\mathcal{S}$. Following the notation of Definition 5.1.12, we get

$$\{\!\{\left(\mathsf{c}^{\#}(\mathsf{f}(\mathsf{a}),\mathsf{f}(\mathsf{a})),\varepsilon\right),\left(\mathsf{f}^{\#}(\mathsf{a}),2\right),\left(\mathsf{f}^{\#}(\mathsf{a}),1\right),\left(\mathsf{a}^{\#},1.1\right)\}\!\} = \{\!\{(t,\pi)\}\!\} \uplus M_{rew} \uplus M_{\bot} \uplus M_{<}$$

with

$$\begin{aligned} \{\!\{(t,\pi)\}\!\} &= \{\!\{\left(\mathsf{f}^{\#}(\mathsf{a}),1\right)\}\!\} & M_{<} &= \{\!\{\left(\mathsf{a}^{\#},1.1\right)\}\!\} \\ M_{rew} &= \{\!\{\left(\mathsf{c}^{\#}(\mathsf{f}(\mathsf{a}),\mathsf{f}(\mathsf{a})),\varepsilon\right)\}\!\} & M_{\bot} &= \{\!\{\left(\mathsf{f}^{\#}(\mathsf{a}),2\right)\}\!\} \end{aligned}$$

Hence, we keep the pair $\left(\mathsf{f}^{\#}(\mathsf{a}),2\right)$ and remove $\left(\mathsf{a}^{\#},1.1\right)$. Furthermore, we have

$$\begin{aligned} & M_1^{+} & & \{\!\{(a[r_1\sigma]_{\chi_a},\tau) \mid (a,\tau) \in M_{rew}\}\!\} \\ &= \{\!\{\left(\mathsf{g}^{\#}(b),1\right),\left(b^{\#},1.1\right)\}\!\} & &= \{\!\{\left(\mathsf{c}^{\#}(\mathsf{g}(\mathsf{b}),\mathsf{f}(\mathsf{a})),\varepsilon\right)\}\!\} \end{aligned}$$

In the pair $\left(\mathsf{c}^{\#}(\mathsf{f}(\mathsf{a}),\mathsf{f}(\mathsf{a})),\varepsilon\right)$, we have to rewrite the copy of $\mathsf{f}(\mathsf{a})$ at position 1 because the additional position indicates this. Finally, the set M_1^{+} contains all new pairs introduced by the right-hand side of the used dependency tuple. Thus we result with

$$\begin{aligned} A' &= M_j^{+} \uplus \{\!\{(a[r_j\sigma]_{\chi_a.\rho},\tau) \mid (a,\tau) \in M_{rew}\}\!\} \uplus M_{\bot} \\ &= \{\!\{\left(\mathsf{c}^{\#}(\mathsf{g}(\mathsf{b}),\mathsf{f}(\mathsf{a})),\varepsilon\right),\left(\mathsf{f}^{\#}(\mathsf{a}),2\right),\left(\mathsf{g}^{\#}(b),1\right),\left(b^{\#},1.1\right)\}\!\} \end{aligned}$$

If we look at the terms that those sets are representing, then we have

$$\mathsf{c}(\mathsf{f}(\mathsf{a}),\mathsf{f}(\mathsf{a})) \xrightarrow{\mathsf{i}}_{\mathcal{S}} \{1 : \mathsf{c}(\mathsf{g}(\mathsf{b}),\mathsf{f}(\mathsf{a}))\}$$

Example 5.1.14 (Resolving the Problem with Dependency Tuples (Details))**.** Consider the rewrite sequence

$$\begin{aligned} & \{ \quad 1 : \{\!\{\left(\mathsf{f}^{\#}(\mathsf{s}),\varepsilon\right)\}\!\}\} \\ \xrightarrow{\mathsf{i}}_{\mathcal{DT}(\mathcal{R}_{tup}),\mathcal{R}_{tup}} & \{ \quad 1 : \{\!\{\left(\mathsf{f}^{\#}(\mathsf{a}),\varepsilon\right),\left(\mathsf{a}^{\#},1\right)\}\!\}\} \\ \xrightarrow{\mathsf{i}}_{\mathcal{DT}(\mathcal{R}_{tup}),\mathcal{R}_{tup}} & \{ \quad \tfrac{1}{2} : \{\!\{\left(\mathsf{f}^{\#}(\mathsf{b}),\varepsilon\right),\left(\mathsf{b}^{\#},1\right)\}\!\},\tfrac{1}{2} : \{\!\{\left(\mathsf{f}^{\#}(\mathsf{c}),\varepsilon\right),\left(\mathsf{c}^{\#},1\right)\}\!\}\} \end{aligned}$$

from Example 5.1.11. We will take a closer look at the second rewrite step. Here, we use the main rewrite pair $(\mathsf{a}^{\#},1)$, the identity substitution id and the dependency tuple

$$(\mathsf{a}^{\#},\mathsf{a}) \to \{\tfrac{1}{2} : \Big(\{\!\{\left(\mathsf{b}^{\#},\varepsilon\right)\}\!\},\mathsf{b}\Big),\tfrac{1}{2} : \Big(\{\!\{\left(\mathsf{c}^{\#},\varepsilon\right)\}\!\},\mathsf{c}\Big)\}$$

Following the notation of Definition 5.1.12, we get

$$\{\!\{\left(\mathsf{f}^{\#}(\mathsf{a}),\varepsilon\right),\left(\mathsf{a}^{\#},1\right)\}\!\} = \{\!\{(t,\pi)\}\!\} \uplus M_{rew} \uplus M_{\bot} \uplus M_{<}$$

with

$$\{\!\{(t,\pi)\}\!\} = \{\!\{(\mathsf{a}^\#,1)\}\!\} \qquad M_< = \varnothing$$
$$M_{rew} = \{\!\{(\mathsf{f}^\#(\mathsf{a}),\varepsilon)\}\!\} \qquad M_\perp = \varnothing$$

For $j = 1$ we get

$$\{\!\{\left(\mathsf{f}^\#(\mathsf{b}),\varepsilon\right),\left(\mathsf{b}^\#,1\right)\}\!\} = B_1 = M_1^+ \uplus \{\!\{(a[r_1\sigma]_{\chi_a},\tau) \mid (a,\tau) \in M_{rew}\}\!\} \uplus M_\perp$$

since the rule is still contained in our PTRS. We have

$$\begin{aligned} & M_1^+ \\ &= \{\!\{(r'\sigma,\pi.\pi') \mid (r',\pi') \in C_1\}\!\} \\ &= \{\!\{(r'id,1.\pi') \mid (r',\pi') \in \{\!\{(\mathsf{b}^\#,\varepsilon)\}\!\}\}\!\} \\ &= \{\!\{(\mathsf{b}^\# id,1.\varepsilon)\}\!\} \\ &= \{\!\{(\mathsf{b}^\#,1)\}\!\} \end{aligned} \qquad \begin{aligned} & \{\!\{(a[r_1\sigma]_{\chi_a},\tau) \mid (a,\tau) \in M_{rew}\}\!\} \\ &= \{\!\{(a[r_1\sigma]_{\chi_a},\tau) \mid (a,\tau) \in \{\!\{(\mathsf{f}^\#(\mathsf{a}),\varepsilon)\}\!\}\}\!\} \\ &= \{\!\{(\mathsf{f}^\#(\mathsf{a})[\mathsf{b}id]_1,\varepsilon)\}\!\} \quad (\text{since } \varepsilon.1 = 1) \\ &= \{\!\{(\mathsf{f}^\#(\mathsf{b}),\varepsilon)\}\!\} \end{aligned}$$

and $M_\perp = \varnothing$. Hence, we result with $\{\!\{\left(\mathsf{f}^\#(\mathsf{b}),\varepsilon\right),\left(\mathsf{b}^\#,1\right)\}\!\}$ in the first part of our final distribution. Completely analogous, we result with $\{\!\{\left(\mathsf{f}^\#(\mathsf{c}),\varepsilon\right),\left(\mathsf{c}^\#,1\right)\}\!\}$ in the second part of our final distribution.

Next, we show that rewriting a well-positioned set always results in well-positioned sets again. The proof of this also shows how a rewrite step with the PPTRS corresponds to a rewrite step with the PTRS on terms.

Lemma 5.1.15 (Rewriting Well-Positioned Sets with $\xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}}$)**.** *Let $A \in \mathsf{PDTS}_{\mathsf{wp}}$ and let $A \xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}} \mu$. Then every $B \in \mathsf{PDTS}$ with $\mu(B) > 0$ is well-positioned.*

Proof. Let $\mu = \{p_1 : B_1, \ldots, p_k : B_k\}$. We have $A \xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}} \mu$ and this means that there is a dependency tuple $(\ell^\#,\ell) \to \{p_1 : (C_1,r_1), \ldots, p_k : (C_k,r_k)\} \in \mathcal{P}$, a main rewrite pair $(t,\pi) \in A$, and a ground substitution $\sigma \in \mathrm{Sub}(\Sigma,\mathcal{V})$ such that the conditions from Definition 5.1.12 hold. Let $d \in \mathcal{T}(\Sigma)$ such that $A \subseteq dp(d)$. We will show for all $1 \le j \le k$ that we have $B_j \subseteq dp(d[r_j\sigma]_\pi)$ and hence, B_j is well-positioned. For this, we assume that $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{S}$. If this does not hold, then we result in a set B'_j such that $B'_j \subseteq B_j$ and the property of being well-positioned is closed under taking subsets by Lemma 5.1.8.

Let $1 \le j \le k$ and let $(t_1,\tau) \in B_j$. We have to show that $t_1^\flat = (d[r_j\sigma]_\pi)|_\tau$. We distinguish between the origin of the pair (t_1,τ) in our rewrite step:

- If we have $(t_1,\tau) \in M_j^+ = \{\!\{(r'\sigma,\pi.\pi') \mid (r',\pi') \in C_j\}\!\}$, then there is a position $\chi \in \mathbb{N}^*$ with $\pi.\chi = \tau$ and a term r' with $(r',\chi) \in C_j \subseteq dp(r_j)$ such that $(t_1,\tau) = (r'\sigma,\pi.\chi)$. We get

$$(d[r_j\sigma]_\pi)|_\tau \overset{\pi.\chi=\tau}{=} r_j\sigma|_\chi \overset{\chi\in\mathrm{Occ}(r_j)}{=} r_j|_\chi\sigma \overset{r_j|_\chi=r'^\flat}{=} (r')^\flat\sigma = (r'\sigma)^\flat = t_1^\flat$$

- If we have $(t_1,\tau) \in M_j^+ = \{\!\{(a[r_j\sigma]_{\chi_a},\tau) \mid (a,\tau) \in M_{rew}\}\!\}$, then there exist $(a,\tau) \in A$ and a non-empty position $\chi \in \mathbb{N}^+$ such that $\tau.\chi = \pi$ and $t_1 = a[r_j\sigma]_\chi$. We get

$$(d[r_j\sigma]_\pi)|_\tau \overset{\tau.\chi=\pi}{=} (d|_\tau)[r_j\sigma]_\chi \overset{d|_\tau=a^\flat}{=} (a^\flat)[r_j\sigma]_\chi \overset{\chi\neq\varepsilon}{=} (a[r_j\sigma]_\chi)^\flat \overset{a[r_j\sigma]_\chi=t_1}{=} t_1^\flat$$

- If we have $(t_1,\tau) \in M_\perp$, then $(t_1,\tau) \in A \subseteq dp(d)$ and $\pi \perp \tau$. We get

$$(d[r_j\sigma]_\pi)|_\tau \overset{\pi\perp\tau}{=} d|_\tau \overset{d|_\tau=t_1^\flat}{=} t_1^\flat$$

All in all, B_j is well-positioned for all $1 \leq j \leq k$. ■

We show with the following example why storing the additional position is necessary.

Example 5.1.16 (Why do we need to store the positions?)**.** Let $\mathcal{R}$ be a PTRS over the signature $\Sigma = \Sigma_D = \{\mathsf{f}, \mathsf{g}, \mathsf{h}, \mathsf{a}, \mathsf{b}, \mathsf{c}\}$ that contains the three rules

$$\begin{aligned} \mathsf{a} &\to \{1 : \mathsf{f}(\mathsf{h}(\mathsf{g}), \mathsf{g})\} \\ \mathsf{g} &\to \{1 : \mathsf{b}\} \\ \mathsf{g} &\to \{1 : \mathsf{c}\} \end{aligned}$$

Using only coupled dependency tuples (with Com-symbols instead of term representation sets $\{\!\{\circ\}\!\}$), we would have $\mathcal{DT}(\mathcal{R}) = \{\mathcal{DT}(1), \mathcal{DT}(2), \mathcal{DT}(3)\}$ with

$$\begin{aligned} \mathcal{DT}(1) = &\quad (\mathsf{a}^\#, \mathsf{a}) \to (\mathsf{Com}_4(\mathsf{f}^\#(\mathsf{h}(\mathsf{g}), \mathsf{g}), \mathsf{h}^\#(\mathsf{g}), \mathsf{g}^\#, \mathsf{g}^\#), \mathsf{f}(\mathsf{h}(\mathsf{g}), \mathsf{g})) \\ \mathcal{DT}(2) = &\quad (\mathsf{g}^\#, \mathsf{g}) \to (\mathsf{Com}_1(\mathsf{b}^\#), \mathsf{b}) \\ \mathcal{DT}(3) = &\quad (\mathsf{g}^\#, \mathsf{g}) \to (\mathsf{Com}_1(\mathsf{c}^\#), \mathsf{c}) \end{aligned}$$

We are not tracking the positions of the two different $\mathsf{g}^\#$'s inside of our Com-Term. If we would allow rewriting arbitrary "copies" of the flattened used redex in a step with our PPTRS, then we would be able to do the following rewrite sequence (with $\mathcal{P} = \mathcal{DT}(\mathcal{R})$ and $\mathcal{S} = \mathcal{R}$):

$$\begin{aligned} \{1 : \mathsf{Com}(\mathsf{a}^\#)\} &\overset{\mathsf{i}}{\rightrightarrows}_{\mathcal{P},\mathcal{S}} \{\, 1 : \mathsf{Com}(\mathsf{f}^\#(\mathsf{h}(\mathsf{g}), \mathsf{g}), \mathsf{h}^\#(\mathsf{g}), \mathsf{g}^\#, \mathsf{g}^\#)\} \\ &\overset{\mathsf{i}}{\rightrightarrows}_{\mathcal{P},\mathcal{S}} \{\, 1 : \mathsf{Com}(\mathsf{f}^\#(\mathsf{h}(\mathsf{g}), \mathsf{b}), \mathsf{h}^\#(\mathsf{g}), \mathsf{Com}(\mathsf{b}^\#), \mathsf{g}^\#)\} \\ &\overset{\mathsf{i}}{\rightrightarrows}_{\mathcal{P},\mathcal{S}} \{\, 1 : \mathsf{Com}(\mathsf{f}^\#(\mathsf{h}(\mathsf{c}), \mathsf{b}), \mathsf{h}^\#(\mathsf{b}), \mathsf{Com}(\mathsf{b}^\#), \mathsf{Com}(\mathsf{c}^\#))\} \end{aligned}$$

And the last term does not correspond to any term in a $\overset{\mathsf{i}}{\rightrightarrows}_{\mathcal{R}}$-rewrite sequence since we have both $\mathsf{h}(\mathsf{c})$ and $\mathsf{h}^\#(\mathsf{b})$ as subterms. Hence, we can not use our PTRS to mirror this rewrite sequence, and this leads to the fact that our chain criterion is not complete in this case.

Next, we define how a PTRS $\mathcal{S}$ rewrites sets in $\mathsf{PDTS}_{\mathsf{wp}}$. Essentially, these are the same sets and restrictions used in the definition of $\overset{\mathsf{i}}{\to\!\triangleright}_{\mathcal{P},\mathcal{S}}$, but there are some slight differences due to the fact that we can not rewrite at the root position of the main rewrite pair anymore (since it has a tuple symbol at the root and our PTRS does not contain tuple symbols).

Definition 5.1.17. Let $\mathcal{S}$ be a PTRS. A set $A \in \mathsf{PDTS}_{\mathsf{wp}}$ rewrites with the PTRS $\mathcal{S}$ to $\{p_1 : B_1, \ldots, p_k : B_k\}$ (denoted $\overset{\mathsf{i}}{\to\!\triangleright}_{\mathcal{S}}$) iff there is a rule $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{S}$, a pair $(t, \pi) \in A$, again called the *main rewrite pair*, a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and a position ρ such that $t|_\rho = \ell\sigma$ and all proper subterms of $\ell\sigma$ are in normal form w.r.t. $\mathcal{S}$. Additionally, there is no pair $(t', \pi') \in A$ such that π' is strictly between π and $\pi.\rho$. Then, the following conditions hold for all $1 \leq j \leq k$:

- Let
$$\begin{aligned} M_{rew} &:= \{\!\{(a, \tau) \in A \mid \tau < \pi\}\!\} \\ M_\perp &:= \{\!\{(a, \tau) \in A \mid \tau \perp \pi.\rho\}\!\} \\ M_< &:= \{\!\{(a, \tau) \in A \mid \pi.\rho \leq \tau\}\!\} \end{aligned}$$
Again, we have
$$A = \{\!\{(t, \pi)\}\!\} \uplus M_{rew} \uplus M_\perp \uplus M_<,$$
due to the fact that there is no pair (t', π') such that π' is strictly between π and $\pi.\rho$.

- The main rewrite pair $\{\!\{(t,\pi)\}\!\}$ gets replaced by $M_j^+ := \{\!\{(t[r_j\sigma]_\rho,\pi)\}\!\}$
- For all $(a,\tau) \in M_{rew}$ there exists a $\chi_a \in \mathbb{N}^+$ such that $\tau.\chi_a = \pi$. Since A is well-positioned, we get $a|_{\chi_a} = t^\flat$ by Lemma 5.1.8. Then we can rewrite a to $a[r_j\sigma]_{\chi_a.\rho}$ using the substitution σ, the position $\chi_a.\rho$, and the rule $\ell \to r_j$, since all proper subterms of $\ell\sigma$ are in normal form w.r.t. $\mathcal{S}$.
- Finally, we have

$$B_j = M_j^+ \uplus \{\!\{(a[r_j\sigma]_{\chi_a.\rho},\tau) \mid (a,\tau) \in M_{rew}\}\!\} \uplus M_\perp.$$

The restriction that there is no pair (t',π') such that π' is strictly between π and $\pi.\rho$ means that the main rewrite pair is at the lowest possible position so that we apply the rewrite rule to every copy with a position above. Such a condition was not needed when defining the rewrite relation of a PPTRS in Definition 5.1.12, as we can only rewrite at the root position of the main rewrite pair (t,π) using a dependency tuple. Finally, our case distinction for the sets $M_\perp$ and $M_<$ do not dependent on the position π of the main rewrite pair but on the position $\pi.\rho$, where our actual rewrite step takes place. Again, when rewriting with a PPTRS, then we can only rewrite at the root position of our main rewrite pair. Hence, we solely use π in the case distinction of Definition 5.1.12. Furthermore, in the definition of $M_<$ we use $\pi.\rho \leq \tau$ (instead of $\pi.\rho < \tau$) this time. This means that our rewrite relation $\xrightarrow{\mathsf{i}}_{\mathcal{S}}$ may remove the pair $(t|_\rho^\#,\pi.\rho)$ if it exists in the set A.

Example 5.1.18 (Rewriting with $\xrightarrow{\mathsf{i}}_{\mathcal{S}}$)**.** Consider the PTRS $\mathcal{S}$, the PPTRS $\mathcal{P}$ and the set A from Example 5.1.13. Let us take a closer look at the following rewrite step with $\xrightarrow{\mathsf{i}}_{\mathcal{S}}$:

$$A \xrightarrow{\mathsf{i}}_{\mathcal{S}} \{1 : A'\} = \{1 : \{\!\{\left(\mathsf{c}^\#(\mathsf{g}(\mathsf{b}),\mathsf{f}(\mathsf{a})),\varepsilon\right),\left(\mathsf{f}^\#(\mathsf{a}),2\right)\}\!\}\}$$

Here, we use the main rewrite pair $\left(\mathsf{c}^\#(\mathsf{f}(\mathsf{a}),\mathsf{f}(\mathsf{a})),\varepsilon\right)$, the identity substitution id and the only rewrite rule. Following the notation of Definition 5.1.17, we get

$$\{\!\{\left(\mathsf{c}^\#(\mathsf{f}(\mathsf{a}),\mathsf{f}(\mathsf{a})),\varepsilon\right),\left(\mathsf{f}^\#(\mathsf{a}),2\right),\left(\mathsf{f}^\#(\mathsf{a}),1\right),\left(\mathsf{a}^\#,1.1\right)\}\!\} = \{\!\{(t,\pi)\}\!\} \uplus M_{rew} \uplus M_\perp \uplus M_<$$

with

$$\{\!\{(t,\pi)\}\!\} = \{\!\{\left(\mathsf{c}^\#(\mathsf{f}(\mathsf{a}),\mathsf{f}(\mathsf{a})),\varepsilon\right)\}\!\} \qquad M_< = \{\!\{\left(\mathsf{f}^\#(\mathsf{a}),1\right),\left(\mathsf{a}^\#,1.1\right)\}\!\}$$
$$M_{rew} = \varnothing \qquad M_\perp = \{\!\{\left(\mathsf{f}^\#(\mathsf{a}),2\right)\}\!\}$$

Note that the position for the definition of the sets $M_<$ and $M_\perp$ is not the position ε from the main rewrite pair, but the position 1, where we actually rewrite inside of the main rewrite pair. Additionally, note that the pair with position 1 is also contained in $M_<$ and gets removed as well. Hence, we remove the pairs $\left(\mathsf{f}^\#(\mathsf{a}),1\right)$ and $\left(\mathsf{a}^\#,1.1\right)$. Furthermore, we have

$$M_1^+ = \{\!\{\left(\mathsf{c}^\#(\mathsf{g}(\mathsf{b}),\mathsf{f}(\mathsf{a})),\varepsilon\right)\}\!\}$$

and thus result with

$$A' = M_j^+ \uplus \{\!\{(a[r_j\sigma]_{\chi_a.\rho},\tau) \mid (a,\tau) \in M_{rew}\}\!\} \uplus M_\perp = \{\!\{\left(\mathsf{c}^\#(\mathsf{g}(\mathsf{b}),\mathsf{f}(\mathsf{a})),\varepsilon\right),\left(\mathsf{f}^\#(\mathsf{a}),2\right)\}\!\}$$

Again, we prove that a rewrite step with $\xrightarrow{\mathsf{i}}_{\mathcal{S}}$ starting with a well-positioned set results in sets that are also well-positioned.

Lemma 5.1.19 (Rewriting Well-Positioned Sets with $\xrightarrow{\mathsf{i}}_{\mathcal{S}}$)**.** *Let A be well-positioned and let $A \xrightarrow{\mathsf{i}}_{\mathcal{S}} \mu$. Then every $B \in \mathsf{PDTS}$ with $\mu(B) > 0$ is well-positioned.*

Proof. This proof is similar to the one of Lemma 5.1.15. Let $\mu = \{p_1 : B_1, \dots, p_k : B_k\}$. We have $A \xrightarrow{\mathsf{i}}_{\mathcal{S}} \mu$ and this means that there is a rule $\ell \to \{p_1 : r_1, \dots, p_k : r_k\} \in \mathcal{S}$, a main rewrite pair $(t, \pi) \in A$, a position $\rho \in \mathbb{N}^+$, and a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ such that the conditions from Definition 5.1.17 hold. Let $d \in \mathcal{T}(\Sigma)$ such that $A \subseteq dp(d)$. We will show that for all $1 \leq j \leq k$ we have $B_j \subseteq dp(d[r_j\sigma]_{\pi.\rho})$ and hence, B_j is well-positioned.

Let $1 \leq j \leq k$ and let $(t_1, \tau) \in B_j$. We have to show that $t_1^\flat = (d[r_j\sigma]_{\pi.\rho})|_\tau$. We distinguish between the origin of the pair (t_1, τ) in our rewrite step:

- If we have $(t_1, \tau) \in M_j^+ = \{\!\{(t[r_j\sigma]_\rho, \pi)\}\!\}$, then $(t_1, \tau) = (t[r_j\sigma]_\rho, \pi)$. We get

$$(d[r_j\sigma]_{\pi.\rho})|_\tau \overset{\tau=\pi}{=} (d[r_j\sigma]_{\pi.\rho})|_\pi = (d|_\pi)[r_j\sigma]_\rho \overset{d|_\pi = t^\flat}{=} (t^\flat)[r_j\sigma]_\rho \overset{\rho \neq \varepsilon}{=} (t[r_j\sigma]_\rho)^\flat \overset{t[r_j\sigma]_\rho = t_1}{=} t_1^\flat$$

- If we have $(t_1, \tau) \in M_j^+ = \{\!\{(a[r_j\sigma]_{\chi\alpha}, \tau) \mid (a, \tau) \in M_{rew}\}\!\}$, then there exist $(a, \tau) \in A$ and a position $\chi\mathbb{N}^+$ such that $\tau.\chi = \pi$ and $t_1 = a[r_j\sigma]_{\chi.\rho}$. We get

$$(d[r_j\sigma]_{\pi.\rho})|_\tau \overset{\tau.\chi=\pi}{=} (d|_\tau)[r_j\sigma]_{\chi.\rho} \overset{d|_\tau = a^\flat}{=} (a^\flat)[r_j\sigma]_{\chi.\rho} \overset{\chi.\rho \neq \varepsilon}{=} (a[r_j\sigma]_{\chi.\rho})^\flat \overset{a[r_j\sigma]_{\chi.\rho} = t_1}{=} t_1^\flat$$

- If we have $(t_1, \tau) \in M_\perp$, then $(t_1, \tau) \in A \subseteq dp(d)$ and $\pi \perp \tau$. We get

$$(d[r_j\sigma]_{\pi.\rho})|_\tau \overset{\pi \perp \tau}{=} d|_\tau \overset{d|_\tau = t_1^\flat}{=} t_1^\flat$$

All in all, B_j is well-positioned for all $1 \leq j \leq k$. ■

In Section 4.2, we have seen how we can represent a $\overset{\mathsf{i}}{\rightrightarrows}_{\mathcal{R}}$-rewrite sequence as a tree. Instead of defining the probabilistic version of an innermost $(\mathcal{P}, \mathcal{S})$-chain using some kind of lifting of $\xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}}$ and $\xrightarrow{\mathsf{i}}_{\mathcal{S}}$, we go one step further and define an innermost $(\mathcal{P}, \mathcal{S})$-chain directly via its tree representation in the next section.

5.2 Computation Trees and Chains

For the rest of this section, let $(\mathcal{P}, \mathcal{S})$ be an arbitrary DP problem. In this chapter, we want to introduce *chain trees*. A chain tree is defined like a rewrite sequence tree, but we have an additional subset $P \subseteq V$ of the inner nodes to indicate whether we used a rewrite step with $\xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}}$ or $\xrightarrow{\mathsf{i}}_{\mathcal{S}}$. After the first definitions and lemmas regarding this new structure, we define $(\mathcal{P}, \mathcal{S})$-*computation tree*, which is the notion of a chain in the probabilistic setting.

Example 5.2.1 (Chain Tree)**.** Consider the following PPTRS $\mathcal{P}$, over the signature Σ with $\Sigma_D = \{\mathsf{f}, \mathsf{a}, \mathsf{b}, \mathsf{c}\}$ and $\Sigma_C = \{\mathcal{O}\}$, containing the rules

$$(\mathsf{a}^\#, \mathsf{a}) \to \{\tfrac{1}{2} : (\{\!\{(\mathsf{f}^\#(\mathsf{b}), \varepsilon), (\mathsf{b}^\#, 1)\}\!\}, \mathsf{f}(\mathsf{b})), \tfrac{1}{2} : (\{\!\{(\mathsf{f}^\#(\mathsf{c}), \varepsilon), (\mathsf{c}^\#, 1)\}\!\}, \mathsf{f}(\mathsf{c}))\} \tag{5.5}$$

$$(\mathsf{f}^\#(\mathcal{O}), \mathsf{f}(\mathcal{O})) \to \{1 : (\{\!\{(\mathsf{a}^\#, \varepsilon)\}\!\}, \mathsf{a})\} \tag{5.6}$$

$$(\mathsf{b}^\#, \mathsf{b}) \to \{1 : (\{\!\{(\mathsf{a}^\#, \varepsilon)\}\!\}, \mathsf{a})\} \tag{5.7}$$

and the PTRS $\mathcal{S}$ containing the rules

$$\mathsf{c} \to \{1 : \mathcal{O}\} \tag{5.8}$$

$$\mathsf{b} \to \{1 : \mathsf{a}\} \tag{5.9}$$

A $(\mathcal{P}, \mathcal{S})$-computation tree is depicted in the following figure. Here, we omit the positions for readability.

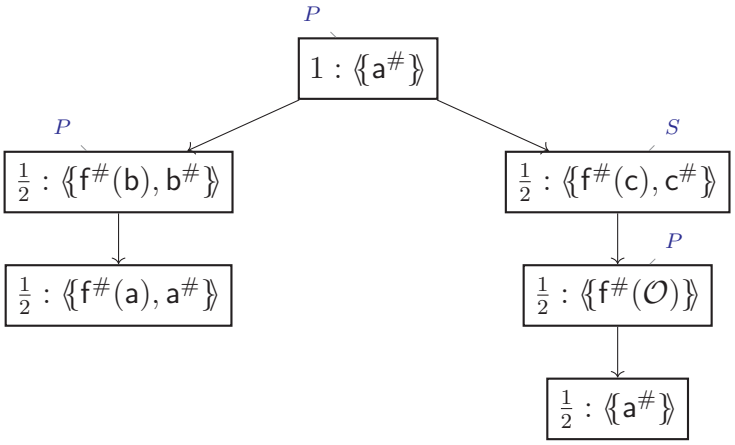


Figure 5.1: Example $(\mathcal{P}, \mathcal{S})$-computation tree $\mathfrak{T}_1$

The edges of the tree represent the used rewrite steps, and the nodes represent pairs $(p : t)$. By the additional labels P and S, we indicate whether we used a rewrite step with $\xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}}$ or $\xrightarrow{\mathsf{i}}_{\mathcal{S}}$.

The two major differences between a $(\mathcal{P}, \mathcal{S})$-computation tree and a $\mathcal{R}$-computation tree for some PTRS $\mathcal{R}$ is the structure induced by the new labels P and S. Remember that a chain in the non-probabilistic setting used the rewrite relation $\xrightarrow{\mathsf{i}}_{\mathcal{D},\mathcal{R}} \circ \xrightarrow{\mathsf{i}}_{\mathcal{R}}^{*}$. This is now represented by the paths in our $(\mathcal{P}, \mathcal{S})$-computation tree. More precisely, we restrict ourselves to trees, such that on every path that starts on some node x, we will have a node labeled P after a finite amount of steps. This means that on every path that starts at the root, we have one step with the PPTRS and then an arbitrary but finite amount of steps of the PTRS on every path, similar to the definition of a non-probabilistic chain. However, we do not require that the nodes from P are all at the same layer, as one can see in Figure 5.1.

We start with the basic definitions and notations regarding the structures that we want to work with, namely *chain trees*.

Definition 5.2.2 (Chain Tree)**.** A *chain tree* is a directed, labeled tree $\mathfrak{T} = (V^{\mathfrak{T}}, E^{\mathfrak{T}}, L^{\mathfrak{T}}, P^{\mathfrak{T}})$ with

- $V^{\mathfrak{T}}$ is a possibly infinite, non-empty set of vertices.
- $E^{\mathfrak{T}} \subseteq V^{\mathfrak{T}} \times V^{\mathfrak{T}}$ is the set of directed edges.
- $L^{\mathfrak{T}} : V^{\mathfrak{T}} \to (0, 1] \times \mathsf{PDTS}_{\mathsf{wp}}$ labels every vertex by a probability and a well-positioned positional dependency tuple set.
- $P^{\mathfrak{T}} \subseteq V^{\mathfrak{T}}$ is a subset of the nodes that indicates whether we used the PPTRS for the rewrite step or the PTRS.

such that the following properties are satisfied:

1. $G^{\mathfrak{T}} = (V^{\mathfrak{T}}, E^{\mathfrak{T}})$ is a finitely branching, directed tree. Let $\mathrm{Leaf}^{\mathfrak{T}}$ be the set of all leaves in $G^{\mathfrak{T}}$ and let $\mathfrak{r}^{\mathfrak{T}}$ be the root node of $G^{\mathfrak{T}}$.

2. For all $x \in V^{\mathfrak{T}}$ with $L^{\mathfrak{T}}(x) = (p_x : A_x)$ and $xE^{\mathfrak{T}} \neq \varnothing$ we have

$$\sum_{y \in xE^{\mathfrak{T}}, L^{\mathfrak{T}}(y) = (p_y : A_y)} p_y = p_x$$

   Here, by $xE^{\mathfrak{T}} \coloneqq \{y \mid (x, y) \in E^{\mathfrak{T}}\}$ we denote the set of direct successors of x.

3. $P^{\mathfrak{T}} \subseteq V^{\mathfrak{T}} \setminus \mathrm{Leaf}^{\mathfrak{T}}$ only contains inner nodes.

4. $L^{\mathfrak{T}}(\varepsilon) = (1, A)$ for some $A \in \mathsf{PDTS}_{\mathsf{wp}}$.

We will write $S^{\mathfrak{T}} \coloneqq (V^{\mathfrak{T}} \setminus \mathrm{Leaf}^{\mathfrak{T}}) \setminus P^{\mathfrak{T}}$ for the set of all inner nodes that are not in $P^{\mathfrak{T}}$. Note that $V^{\mathfrak{T}} = P^{\mathfrak{T}} \uplus S^{\mathfrak{T}} \uplus \mathrm{Leaf}^{\mathfrak{T}}$. For a node $x \in V^{\mathfrak{T}}$ with $L^{\mathfrak{T}}(x) = (p : A)$, we will also write $p_x^{\mathfrak{T}}$ for p and $A_x^{\mathfrak{T}}$ for A. If the chain tree is clear from the context, we may omit it in the notations for readability (e.g., we write V for $V^{\mathfrak{T}}$). For a node $x \in V$, we write $\mathrm{d}(x)$ for the depth of x, that is, the number of edges in the path from the root to x.

All of the constructions and lemmas about RSTs can be adapted to chain trees with the same results. For the constructions, we only have to define how the set P changes. The proofs are most of the time the same as for RSTs since the set P is not of any importance yet. We will start using the new set P when talking about $(\mathcal{P}, \mathcal{S})$-computation trees.

Definition 5.2.3 (Induced Sub Chain Tree)**.** Let $\mathfrak{T} = (V, E, L, P)$ be a chain tree. Let $W \subseteq V$ be non-empty, weakly connected and for all $x \in W$ we have $xE \cap W = \varnothing$ or $xE \cap W = xE$. The property of being non-empty and weakly connected ensures that the resulting graph $G^{\mathfrak{T}[W]} = (W, E \cap (W \times W))$ is a tree again. The last property regarding the successors of a node ensures that the sum of probabilities for the successors of a node x is equal to the probability for the node x itself. Then, we define the sub chain tree $\mathfrak{T}[W]$ by

$$\mathfrak{T}[W] \coloneqq (W, (E \cap (W \times W)), L^W, (P \cap (W \setminus W_{\mathrm{Leaf}})))$$

Here, by W_{Leaf}, we denote the leaves of the tree $G^{\mathfrak{T}[W]}$ so that the new set $P \cap (W \setminus W_{\mathrm{Leaf}})$ only contains inner nodes. Let $w \in W$ be the root of $G^{\mathfrak{T}[W]}$. To ensure that the root of our induced sub chain tree has the probability 1 again, we use the labeling $(L^W)(x) = (\frac{p_x^{\mathfrak{T}}}{p_w^{\mathfrak{T}}} : A_x^{\mathfrak{T}})$ for all nodes $x \in W$. If we have $\mathfrak{r}^{\mathfrak{T}} \in W$, then we call the induced sub chain tree *grounded*.

Example 5.2.4 (Induced Sub Chain Tree)**.** The following chain tree $\mathfrak{T}_2$ is a grounded, induced sub chain tree of $\mathfrak{T}_1$ from Example 5.2.1.

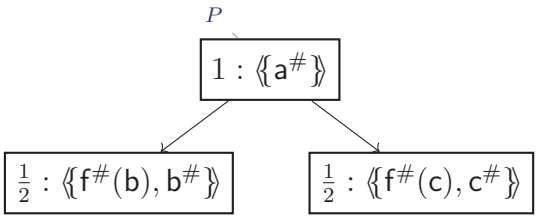


Figure 5.2: $\mathfrak{T}_2$

Since this is a grounded, induced sub chain tree, we do not have to adjust the probabilities. However, we have to adjust the set P so that it does not contain any leaf.

Lemma 5.2.5. *Let $\mathfrak{T} = (V, E, L, P)$ be a chain tree and let $W \subseteq V$ be satisfying the conditions of Definition 5.2.3. Then $\mathfrak{T}[W] = (V', E', L', P')$ is a chain tree again.*

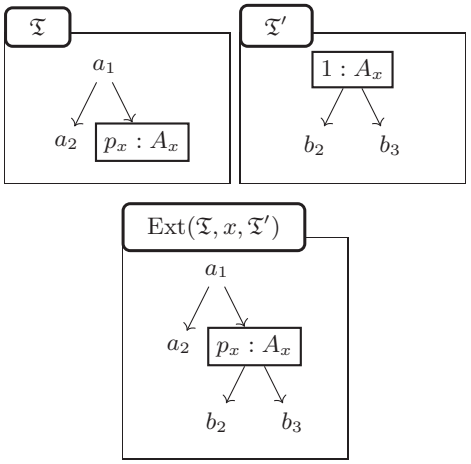
Proof. Same proof as for Lemma 4.2.5. We only have to additionally prove that $P' \subseteq V' \setminus \mathrm{Leaf}^{\mathfrak{T}[W]}$ holds. But this is ensured due to the fact that we explicitly remove the leaves from P' in our construction. ■

Definition 5.2.6 (Extension of Chain Trees)**.** Let $\mathfrak{T} = (V, E, L, P)$, $\mathfrak{T}' = (V', E', L', P')$ be chain trees with $V \cap V' = \varnothing$, and let $x \in \mathrm{Leaf}^{\mathfrak{T}}$ such that $A^{\mathfrak{T}}_x = A^{\mathfrak{T}'}_{\mathfrak{r}^{\mathfrak{T}'}}$ (i.e., the set of node x in $\mathfrak{T}$ is equal to the set of the root of $\mathfrak{T}'$). Then, we define the *extension* of $\mathfrak{T}$ w.r.t. the leaf x and the chain tree $\mathfrak{T}'$ (denoted as $\mathrm{Ext}(\mathfrak{T}, x, \mathfrak{T}')$) by

$$\mathrm{Ext}(\mathfrak{T}, x, \mathfrak{T}') \coloneqq (V_{\mathrm{Ext}}, E_{\mathrm{Ext}}, L_{\mathrm{Ext}}, P_{\mathrm{Ext}})$$

with

$$\begin{aligned} V_{\mathrm{Ext}} &\coloneqq V \cup (V' \setminus \{\mathfrak{r}^{\mathfrak{T}'}\}) \\ E_{\mathrm{Ext}} &\coloneqq E \cup (E' \setminus \{(\mathfrak{r}^{\mathfrak{T}'}, y) \mid y \in \mathfrak{r}^{\mathfrak{T}'} E'\}) \\ &\quad \cup \{(x, y) \mid y \in \mathfrak{r}^{\mathfrak{T}'} E'\} \\ P_{\mathrm{Ext}} &\coloneqq \begin{cases} P \cup P' \cup \{x\} & \mathfrak{r}^{\mathfrak{T}'} \in P' \\ P \cup P' & \mathfrak{r}^{\mathfrak{T}'} \notin P' \end{cases} \end{aligned}$$



And for all $z \in V_{\mathrm{Ext}}$ we define

$$L_{\mathrm{Ext}}(z) = \begin{cases} (p^{\mathfrak{T}}_z, A^{\mathfrak{T}}_z) & , z \in V, \\ (p^{\mathfrak{T}}_x \cdot p^{\mathfrak{T}'}_z, A^{\mathfrak{T}'}_z) & , z \in V'. \end{cases}$$

Example 5.2.7. Consider the chain tree $\mathfrak{T}_1$ from Example 5.2.1 and the chain tree $\mathfrak{T}_2$ from Example 5.2.4. We can use $\mathfrak{T}_2$ to extend $\mathfrak{T}_1$ in the leaf x that is labeled by $(\frac{1}{2} : \{\!\{a^{\#}\}\!\})$. Then $\mathrm{Ext}(\mathfrak{T}, x, \mathfrak{T}')$ has the form:

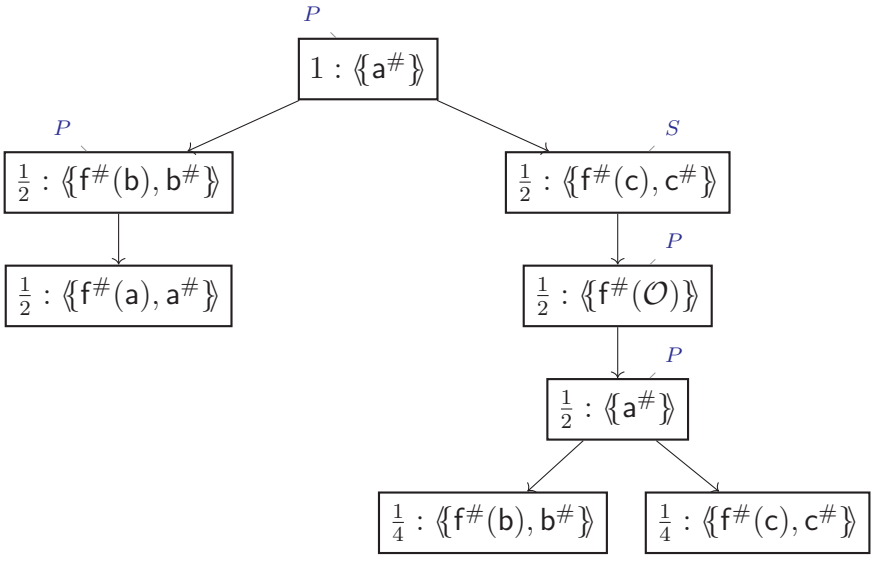


Figure 5.3: $\mathrm{Ext}(\mathfrak{T}_1, x, \mathfrak{T}_2)$

Again, we are also able to perform multiple (even infinite) extension steps simultaneously.

Definition 5.2.8 (Family Extension of Chain Trees)**.** Let $\mathfrak{T} = (V, E, L, P)$ be a chain tree, $H \subseteq \mathrm{Leaf}^{\mathfrak{T}}$, and let $(\mathfrak{T}_x)_{x \in H}$ be a family of chain trees such that $\mathfrak{T}_x = (V_x, E_x, L_x, P_x)$, $A^{\mathfrak{T}}_x = A^{\mathfrak{T}_x}_{\mathfrak{r}^{\mathfrak{T}_x}}$ for all $x \in H$ and all occurring chain trees have disjoint node sets. Additionally, let $\mathrm{Ext}(\mathfrak{T}, x, \mathfrak{T}_x) = (V'_x, E'_x, L'_x, P'_x)$. Then we define the *family extension* of $\mathfrak{T}$ w.r.t. the

family $(\mathfrak{T}_x)_{x\in H}$ (denoted as $\mathrm{FamExt}(\mathfrak{T},(\mathfrak{T}_x)_{x\in H})$) by

$$\begin{aligned}\mathrm{FamExt}(\mathfrak{T},(\mathfrak{T}_x)_{x\in H}) &\coloneqq \bigcup_{x\in H}\mathrm{Ext}(\mathfrak{T},x,\mathfrak{T}_x)\\ &\coloneqq (\bigcup_{x\in H}V'_x,\bigcup_{x\in H}E'_x,\bigcup_{x\in H}L'_x,\bigcup_{x\in H}P'_x)\end{aligned}$$

Here, the labeling $\bigcup_{x\in H}L'_x$ is defined such that $\left(\bigcup_{x\in H}L'_x\right)|_{V'_x}=L'_x$ for all $x\in H$.

Lemma 5.2.9. *Let $\mathfrak{T}=(V,E,L,P)$, $\mathfrak{T}'=(V',E',L',P')$ be chain trees with $V\cap V'=\varnothing$, and let $x\in\mathrm{Leaf}^{\mathfrak{T}}$ such that $A^{\mathfrak{T}}_x=A^{\mathfrak{T}'}_{\mathfrak{r}^{\mathfrak{T}'}}$. Then $\mathrm{Ext}(\mathfrak{T},x,\mathfrak{T}')$ is a chain tree. Furthermore, if we have $H\subseteq\mathrm{Leaf}^{\mathfrak{T}}$, and $(\mathfrak{T}_x)_{x\in H}$ is a family of chain trees such that $\mathfrak{T}_x=(V_x,E_x,L_x,P_x)$, $A^{\mathfrak{T}}_x=A^{\mathfrak{T}_x}_{\mathfrak{r}^{\mathfrak{T}'}}$ for all $x\in H$ and with pairwise disjoint node sets. Then $\mathrm{FamExt}(\mathfrak{T},(\mathfrak{T}_x)_{x\in H})$ is a chain tree as well.*

Proof. Same proof as for Lemma 4.2.9. We only have to additionally prove that $P^{\mathrm{Ext}(\mathfrak{T},x,\mathfrak{T}')}$ only contains inner nodes. But this holds due to the fact that $P^{\mathfrak{T}}$ only contains inner nodes from $\mathfrak{T}$ and $P^{\mathfrak{T}'}$ only contains inner nodes for $\mathfrak{T}'$. Furthermore, we only add x into P_{Ext} if the root of $\mathfrak{T}'$ is in P'. But this means that x can not be a leaf anymore. For extension families, we have the same reasoning again. ■

Definition 5.2.10 (Convergence Notations)**.** Let $\mathfrak{T}$ be a chain tree and let $H\subseteq\mathrm{Leaf}$. Let $x_0,x_1,x_2,\ldots$ be an arbitrary enumeration of H. Then we define:

$$|\mathfrak{T}|_H\coloneqq\begin{cases}\sum_{v\in H}p_v & ,\text{ if }|H|\text{ finite,}\\ \lim_{i\to\infty}\sum_{v\in(x_k)_{k\leq i}}p_v & ,\text{ otherwise.}\end{cases}$$

If we have $|\mathfrak{T}|_H=c$ we say that $\mathfrak{T}$ *converges* w.r.t. H with probability c. If $H=\mathrm{Leaf}$, we simply say that $\mathfrak{T}$ converges with probability c.

Again, the value of $|\mathfrak{T}|_H$ does not depend on the enumeration but just on the chain tree $\mathfrak{T}$ due to the following lemma.

Lemma 5.2.11. *Let $\mathfrak{T}$ be a chain tree, $H\subseteq\mathrm{Leaf}$ be infinite, and let $x_0,x_1,x_2,\ldots$ be an arbitrary enumeration of H. Then $\sum_{v\in(x_k)_{k\leq i}}p_v$ is strictly increasing for $i\to\infty$ and bounded from above by 1. Hence, the sum $\sum_{v\in(x_k)_{k\leq i}}p_v$ is absolutely convergent for $i\to\infty$, i.e., $\lim_{i\to\infty}\sum_{v\in(x_k)_{k\leq i}}p_v$ exists.*

Proof. Same proof as for Lemma 4.2.11. ■

While for every finite chain tree $\mathfrak{T}$ we have $|\mathfrak{T}|_{\mathrm{Leaf}}=1$, this does not hold for infinite chain trees in general. Again, analogous to rewrite sequence trees, we can have infinite chain trees that converge with a probability <1 or even 0 if there is no leaf at all.

Next, we investigate how the absolute Leaf value is impacted by our constructions. Again due to the fact that the new set P is not of any importance for this, we get the same results as for RSTs in the previous chapter.

Lemma 5.2.12. *Let $\mathfrak{T}=(V,E,L,P),\mathfrak{T}'=(V',E',L',P')$ be chain trees with $V\cap V'=\varnothing$, and let $x\in\mathrm{Leaf}^{\mathfrak{T}}$ such that $A^{\mathfrak{T}}_x=A^{\mathfrak{T}'}_{\mathfrak{r}^{\mathfrak{T}'}}$. Then we have*

$$|\,\mathrm{Ext}(\mathfrak{T},x,\mathfrak{T}')|_{\mathrm{Leaf}}=|\mathfrak{T}|_{\mathrm{Leaf}}-p^{\mathfrak{T}}_x+p^{\mathfrak{T}}_x\cdot|\mathfrak{T}'|_{\mathrm{Leaf}}.$$

Furthermore, if we have $H\subseteq\mathrm{Leaf}^{\mathfrak{T}}$, and $(\mathfrak{T}_x)_{x\in H}$ is a family of RSTs such that $\mathfrak{T}_x=(V_x,E_x,L_x,P_x)$, $A^{\mathfrak{T}}_x=A^{\mathfrak{T}_x}_{\mathfrak{r}^{\mathfrak{T}_x}}$ for all $x\in H$ and with pairwise disjoint node sets. Then, we have

$$|\,\mathrm{FamExt}(\mathfrak{T},(\mathfrak{T}_x)_{x\in H})|_{\mathrm{Leaf}}=|\mathfrak{T}|_{\mathrm{Leaf}}-|\mathfrak{T}|_H+\sum_{x\in H}p^{\mathfrak{T}}_x\cdot|\mathfrak{T}_x|_{\mathrm{Leaf}}.$$

Proof. Same proof as for Lemma 4.2.12. ■

Corollary 5.2.13 (Little Extension Lemma)**.** *Let* $\mathfrak{T} = (V, E, L, P), \mathfrak{T}' = (V', E', L', P')$ *be chain trees with* $V \cap V' = \varnothing$*, and let* $x \in \mathrm{Leaf}^{\mathfrak{T}}$ *such that* $A^{\mathfrak{T}}_x = A^{\mathfrak{T}'}_{\mathfrak{r}_{\mathfrak{T}'}}$*. Then we have* $|\operatorname{Ext}(\mathfrak{T}, x, \mathfrak{T}')|_{\mathrm{Leaf}} \leq |\mathfrak{T}|_{\mathrm{Leaf}}$*. Moreover,* $|\mathfrak{T}|_{\mathrm{Leaf}} = |\operatorname{Ext}(\mathfrak{T}, x, \mathfrak{T}')|_{\mathrm{Leaf}}$ *iff* $|\mathfrak{T}'|_{\mathrm{Leaf}} = 1$*.*

Proof. Same proof as for Corollary 4.2.13. ■

Corollary 5.2.14 (Full Extension Lemma)**.** *Let* $\mathfrak{T}$ *be a chain tree, let* $H \subseteq \mathrm{Leaf}^{\mathfrak{T}}$ *and let* $(\mathfrak{T}_x)_{x \in H}$ *be a family of RSTs such that* $\mathfrak{T}_x = (V_x, E_x, L_x, P_x)$*,* $A^{\mathfrak{T}}_x = A^{\mathfrak{T}_x}_{\mathfrak{r}_{\mathfrak{T}_x}}$ *for all* $x \in H$ *and with pairwise disjoint node sets. Then we have* $|\operatorname{FamExt}(\mathfrak{T}, (\mathfrak{T}_x)_{x \in \mathrm{Hole}})|_{\mathrm{Leaf}} \leq |\mathfrak{T}|_{\mathrm{Leaf}}$*. Moreover,* $|\mathfrak{T}|_{\mathrm{Leaf}} = |\operatorname{FamExt}(\mathfrak{T}, (\mathfrak{T}_x)_{x \in \mathrm{Hole}})|_{\mathrm{Leaf}}$ *iff* $|\mathfrak{T}_x|_{\mathrm{Leaf}} = 1$ *for all* $x \in H$*.*

Proof. Same proof as for Corollary 4.2.14. ■

Lemma 5.2.15. *Let* $\mathfrak{T}$ *be a chain tree. Then we have* $|\mathfrak{T}|_{\mathrm{Leaf}} = 1$ *iff for all induced sub chain trees* $\mathfrak{T}'$ *we have* $|\mathfrak{T}'|_{\mathrm{Leaf}} = 1$*.*

Proof. Same proof as for Lemma 4.2.15. ■

Finally, we define $(\mathcal{P}, \mathcal{S})$-computation trees, which is the probabilistic version of a chain.

Definition 5.2.16 ($(\mathcal{P}, \mathcal{S})$-computation tree)**.** Let $\mathfrak{T} = (V, E, L, P)$ be a chain tree. We say that $\mathfrak{T}$ is a $(\mathcal{P}, \mathcal{S})$-*computation tree* iff additionally the following conditions hold:

(a) For every $x \in P$ with $xE = \{y_1, \ldots, y_k\}$ we have $A_x \xrightarrow{\mathrm{i}}_{\mathcal{P},\mathcal{S}} \{\frac{p_{y_1}}{p_x} : A_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : A_{y_k}\}$

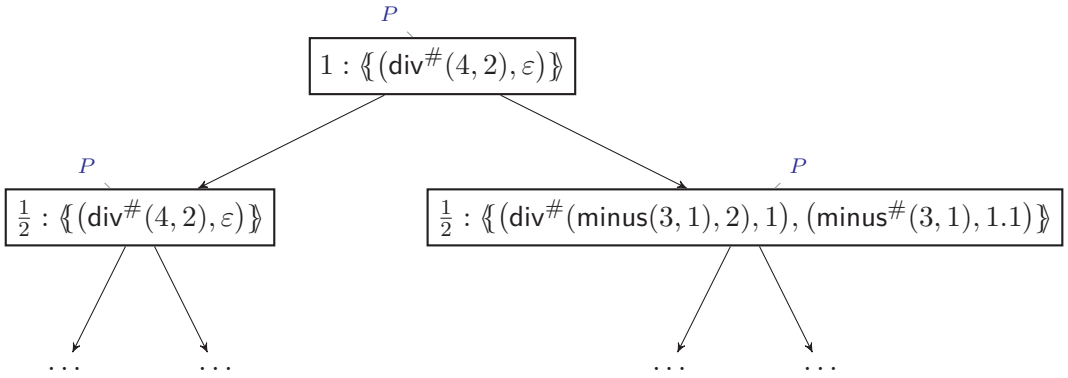
(b) For every $x \in S$ with $xE = \{y_1, \ldots, y_k\}$ we have $A_x \xrightarrow{\mathrm{i}}_{\mathcal{S}} \{\frac{p_{y_1}}{p_x} : A_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : A_{y_k}\}$

(c) Every infinite path contains an infinite amount of P nodes. Since we are dealing with finitely-branching trees, this is equivalent to saying that every path that starts at some node x has a node in P after a finite amount of steps. Or that there exists no infinite path of purely S nodes.

Note that our $(\mathcal{P}, \mathcal{S})$-computation tree does not have to start with a node from P, while in the non-probabilistic chain, we are required to use a dependency pair at the start. However, whether we start with a dependency pair or not is not of any importance. The important property is that we cannot use infinitely many rewrite rules without using any dependency pair.

Most of the conditions of a $(\mathcal{P}, \mathcal{S})$-computation tree only depend on the local neighborhood of the nodes. To be precise, in order to verify that (a), (b) and the properties of a chain tree hold, it suffices to look at each node together with its direct successors. The only global property that we have is (c), where we need to look at the infinite paths in the tree.

Example 5.2.17 ($(\mathcal{DT}(\mathcal{R}_{div}), \mathcal{R}_{div})$-Computation Tree)**.** Again, we consider the PTRS $\mathcal{R}_{div}$ from Example 5.0.1 and the corresponding set of dependency tuples from Example 5.1.10. Here, we have the following possible $(\mathcal{DT}(\mathcal{R}_{div}), \mathcal{R}_{div})$-computation tree that does not use any $\xrightarrow{\mathrm{i}}_{\mathcal{S}}$ steps.



Definition 5.2.18 (Innermost Termination and Innermost AST for $(\mathcal{P},\mathcal{S})$)**.** We say that a DP problem $(\mathcal{P},\mathcal{S})$ is innermost surely terminating iff there is no infinite $(\mathcal{P},\mathcal{S})$-computation tree. A DP problem $(\mathcal{P},\mathcal{S})$ is innermost almost-surely terminating (innermost AST) iff for all $(\mathcal{P},\mathcal{S})$-computation trees $\mathfrak{T}$ we have $|\mathfrak{T}|_{\text{Leaf}} = 1$.

Note that we defined $(\mathcal{P},\mathcal{S})$-computation trees using the relations $\overset{\text{i}}{\rightarrowtriangle}_{\mathcal{P},\mathcal{S}}$ and $\overset{\text{i}}{\rightarrowtriangle}_{\mathcal{S}}$ and thus we implicitly defined everything w.r.t. innermost evaluation.

Again, we can use the constructions regarding chain trees for $(\mathcal{P},\mathcal{S})$-computation trees as well.

Corollary 5.2.19. *Induced sub chain trees, extensions and family extensions of $(\mathcal{P},\mathcal{S})$-computation trees are $(\mathcal{P},\mathcal{S})$-computation trees again.*

Proof. It is straightforward to see that all of the properties of Definition 5.2.16 hold since they hold for all of the initial $(\mathcal{P},\mathcal{S})$-computation trees. ■

Note that in a family extension, we extend the original $(\mathcal{P},\mathcal{S})$-computation tree at parallel positions. The infinite extension of a $(\mathcal{P},\mathcal{S})$-computation tree, where we place $(\mathcal{P},\mathcal{S})$-computation trees after each other does not have to be a $(\mathcal{P},\mathcal{S})$-computation tree anymore. To see this, consider $(\mathcal{P},\mathcal{S})$-computation trees that only consist of nodes from S. Then all of the $(\mathcal{P},\mathcal{S})$-computation trees must be finite in order to satisfy the global property, but an infinite extension of those $(\mathcal{P},\mathcal{S})$-computation trees after each other would contain an infinite path purely out of S nodes, which is not allowed.

Next, we prove two lemmas, which will be the most important tools for our proofs of the processors later on. First, we prove the *P-Partition Lemma*. It states that if we can partition $\mathcal{P} = \mathcal{P}_1 \uplus \mathcal{P}_2$ such that $(\mathcal{P}_1,\mathcal{S})$ is AST, then for $(\mathcal{P},\mathcal{S})$ it suffices to look at $(\mathcal{P},\mathcal{S})$-computation trees such that every infinite path has an infinite amount of P nodes that correspond to steps with $\mathcal{P}_2$. This does not only work for a partition $\mathcal{P} = \mathcal{P}_1 \uplus \mathcal{P}_2$, but we can even more generally speak about the set P of a chain tree itself. If we have a chain tree $\mathfrak{T}$ that converges with probability < 1 and suppose that we can partition the set $P^{\mathfrak{T}} = P_1 \uplus P_2$ of the chain tree such that for every induced sub chain tree $\mathfrak{T}'$ that only uses nodes from P_1 converges with probability 1, then there is a grounded induced sub chain tree that converges with probability < 1 such that on every infinite path, there is an infinite amount of P_2 nodes.

Lemma 5.2.20 (P-Partition Lemma)**.** *Let $\mathfrak{T} = (V,E,L,P)$ be a chain tree that converges with probability < 1. Assume that we can partition $P = P_1 \uplus P_2$ such that for every induced sub chain tree that only contains P nodes from P_1 converges with probability 1. Then there is a grounded induced sub chain tree $\mathfrak{T}'$ that converges with probability < 1 such that every infinite path has an infinite number of P_2 nodes.*

Proof. Let $\mathfrak{T} = (V, E, L, P)$ be a chain tree with $|\mathfrak{T}|_{\text{Leaf}} = c < 1$ for some $c \in \mathbb{R}$. Since we have $0 \leq c < 1$, there is an $\varepsilon > 0$ such that $c + \varepsilon < 1$. Remember that the formula for the geometrical sum is:

$$\sum_{n=1}^{\infty} \left(\frac{1}{d}\right)^n = \frac{1}{d-1}, \text{ for all } \frac{1}{|d|} < 1$$

We set $d \coloneqq \frac{1}{\varepsilon} + 2$ and get

$$\frac{1}{\varepsilon} + 1 < \frac{1}{\varepsilon} + 2 \Leftrightarrow \frac{1}{\varepsilon} + 1 < d \Leftrightarrow \frac{1}{\varepsilon} < d - 1 \Leftrightarrow \frac{1}{d-1} < \varepsilon$$

We will now create a grounded induced sub chain tree $\mathfrak{T}' = (V', E', L', P')$ such that every infinite path will have an infinite number of P_2 nodes and such that

$$|\mathfrak{T}'|_{\text{Leaf}} \leq |\mathfrak{T}|_{\text{Leaf}} + \sum_{n=1}^{\infty} \left(\frac{1}{d}\right)^n \tag{5.10}$$

Then, we finally have

$$|\mathfrak{T}'|_{\text{Leaf}} \leq |\mathfrak{T}|_{\text{Leaf}} + \sum_{n=1}^{\infty} \left(\frac{1}{d}\right)^n = |\mathfrak{T}|_{\text{Leaf}} + \frac{1}{d-1} = c + \frac{1}{d-1} < c + \varepsilon < 1$$

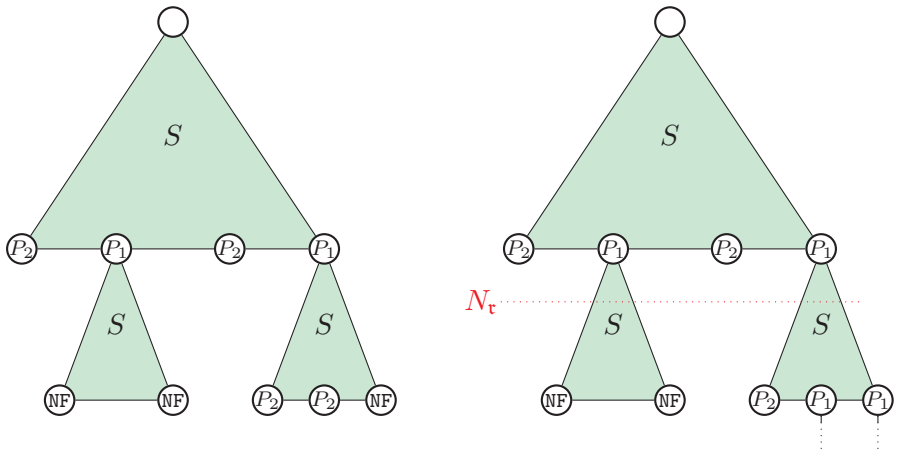
The idea of this construction is that we cut infinite sub-paths of pure P_1 and S nodes as soon as the probability for normal forms on that path is high enough. In this way, one then obtains paths where after finitely many P_1 nodes, there is always a P_2 node.

The construction goes as follows. For any node $x \in V$ let $\mathcal{L}_2(x)$ be the number of P_2 nodes in the path from the root to x. Furthermore, for any set $W \subseteq V$ and $k \in \mathbb{N}$ let $\mathfrak{L}_2(W, k) \coloneqq \{x \in W \mid \mathcal{L}_2(x) \leq k \vee (x \in P \wedge \mathcal{L}_2(x) \leq k + 1)\}$ be the set of all nodes in W that have at most k nodes from P_2 in the path from the root to its predecessor. So if $x \in W$ is not in P_2, then we have at most k nodes from P_2 in the path from the root to itself and if $x \in W$ is in P_2, then we have at most $k + 1$ nodes from P_2 in the path from the root to itself. We will inductively create a set $V_k \subseteq V$ such that $V_k \subseteq \mathfrak{L}_2(V, k)$ and then define the grounded induced sub chain tree as $\mathfrak{T}' \coloneqq \mathfrak{T}[\bigcup_{k \in \mathbb{N}} V_k]$. In other words, we are going inductively over the $\mathcal{L}_2$-levels of the tree $\mathfrak{T}$ and cut infinite sub-paths of pure P_1 and S nodes as soon as the probability for normal forms on that path is high enough.

In the base case, we consider the induced sub chain tree $T_{\mathfrak{t}} \coloneqq \mathfrak{T}[\mathfrak{L}_2(V, 0)]$. This tree only contains P nodes from P_1. While the node set $\mathfrak{L}_2(V, 0)$ itself may contain nodes in P_2, this can only occur at the leaves of $T_{\mathfrak{t}}$, and by definition of an induced sub chain tree, we remove every leaf from P in the creation of $T_{\mathfrak{t}}$. Using our assumption we get $|T_{\mathfrak{t}}|_{\text{Leaf}} = 1$. In Figure 5.4 one can see the different possibilities for $T_{\mathfrak{t}}$. Either $T_{\mathfrak{t}}$ is finite, or $T_{\mathfrak{t}}$ is infinite. In the first two cases, we can add all the nodes to V_0 since there is no infinite path of pure P_1 and S nodes. Hence, we define $V_0 \coloneqq \mathfrak{L}_2(V, 0)$. In the last case, we have to cut the tree at a specific depth once the probability of leaves is high enough. Let $\mathrm{d}_{\mathfrak{t}}(y)$ be the depth of the node y in the tree $T_{\mathfrak{t}}$. Moreover, let $D_{\mathfrak{t}}(k) \coloneqq \{x \in \mathfrak{L}_2(V, 0) \mid \mathrm{d}_{\mathfrak{t}}(y) \leq k\}$ be the set of nodes in $T_{\mathfrak{t}}$ that have a depth of at most k. Since $|T_{\mathfrak{t}}|_{\text{Leaf}} = 1$ and $|\circ|_{\text{Leaf}}$ is monotone w.r.t. the depth of the tree $T_{\mathfrak{t}}$, we can find an $N_{\mathfrak{t}} \in \mathbb{N}$ such that

$$\sum_{x \in \text{Leaf}^{T_{\mathfrak{t}}}, d_{\mathfrak{t}}(x) \leq N_{\mathfrak{t}}} p_x^{T_{\mathfrak{t}}} \geq 1 - \frac{1}{d}$$

We include all nodes from $D_{\mathfrak{t}}(N_{\mathfrak{t}})$ in V_0 and delete every other node of $T_{\mathfrak{t}}$. In other words, we cut the tree after depth $N_{\mathfrak{t}}$. This cut can be seen in Figure 5.4, indicated by the red line. Therefore, we set $V_0 \coloneqq D_{\mathfrak{t}}(N_{\mathfrak{t}})$ in this case.



(a) $T_\mathfrak{r}$ finite (b) $T_\mathfrak{r}$ infinite

Figure 5.4: Possibilities for $T_\mathfrak{r}$

For the induction step, assume that we have already defined a subset $V_i \subseteq \mathfrak{L}_2(V, i)$. Let $H_i \coloneqq \{x \in V_i \mid x \in P \wedge \mathcal{L}_2(x) = i + 1\}$ be the set of leaves in $\mathfrak{T}[V_i]$ that are in P_2. Additionally, let $x \in H_i$. For each x, we consider the induced sub chain tree that starts at x until we see the next node from P_2, including the node itself. Everything below such a node will be cut. To be precise, we set $V_x \coloneqq \mathfrak{L}_2(xE^*, i + 1)$ and look at the tree $T_x = (V_x, E_x, L_x) \coloneqq \mathfrak{T}[V_x]$.

First, we show that $|T_x|_{\text{Leaf}} = 1$. For every successor y of x, the induced sub chain tree $T_y \coloneqq T_x[y(E_x)^*]$ of T_x that starts at y does not contain any nodes from P_2. Hence, we have $|T_y|_{\text{Leaf}} = 1$, by our assumption. Then, T_x is the family extension of the finite chain tree $\mathfrak{T}[\{x\} \cup xE]$ and the family $(T_y)_{y \in xE}$. Since $\mathfrak{T}[\{x\} \cup xE]$ is finite we have $|\mathfrak{T}[\{x\} \cup xE]|_{\text{Leaf}} = 1$ and since we have $|T_y|_{\text{Leaf}} = 1$ for all $y \in xE$, we get $|T_x|_{\text{Leaf}} = 1$ by the full extension lemma (Corollary 5.2.14).

Now, for the construction, we have the same cases as before. Either T_x is finite or T_x is infinite. In the first case, we can add all the nodes again. For this, we set $Z_x \coloneqq V_x$. In the second case, we once again cut the tree at a specific depth once the probability for leaves is high enough. Let $\mathrm{d}_x(z)$ be the depth of the node z in the tree T_x. Moreover, let $D_x(k) \coloneqq \{x \in V_x \mid \mathrm{d}_x(z) \leq k\}$ be the set of nodes in T_x that have a depth of at most k. Since $|T_x|_{\text{Leaf}} = 1$ and $|\circ|_{\text{Leaf}}$ is monotone w.r.t. the depth of the tree T_x, we can find an $N_x \in \mathbb{N}$ such that

$$\sum_{y \in \text{Leaf}^{T_x}, d_x(y) \leq N_x} p_y^{T_x} \geq 1 - \left(\frac{1}{d}\right)^{i+1} \cdot \frac{1}{|H_i|}$$

We will include all nodes from $D_x(N_x)$ in V_{i+1} and delete every other node of T_x. In other words, we cut the tree after depth N_x. Therefore, we set $Z_x \coloneqq D_x(N_x)$ in this case.

We do this for each $x \in H$ and in the end, we set $V_{i+1} \coloneqq V_i \cup \bigcup_{x \in H} Z_x$.

It is straightforward to see that $\bigcup_{k \in \mathbb{N}} V_k$ satisfies the conditions of Definition 5.2.3, as we only cut after certain nodes in our construction. Hence, $\bigcup_{k \in \mathbb{N}} V_k$ is non-empty, weakly connected, and we either have no or all successors of a node inside of it. Furthermore, $\mathfrak{T}' \coloneqq \mathfrak{T}[\bigcup_{k \in \mathbb{N}} V_k]$ is a grounded induced sub chain tree and does not contain an infinite path of pure P_1 and S nodes as we cut every such path after a finite depth. The last thing we need to prove is that $|\mathfrak{T}'|_{\text{Leaf}} \leq |\mathfrak{T}(\mu)|_{\text{Leaf}} + \sum_{n=1}^{\infty} \left(\frac{1}{d}\right)^n$ holds. During the i-th construction iteration, we may increase the value of $|\mathfrak{T}(\mu)|_{\text{Leaf}}$ by the sum of all probabilities

corresponding to the new leaves resulting from the cuts. As we cut at most $|H_i|$ trees in the i-th iteration and for each such tree, we added at most a total probability of $\left(\frac{1}{d}\right)^{i+1} \cdot \frac{1}{|H_i|}$, the value of $|\mathfrak{T}(\mu)|_{\text{Leaf}}$ might increase by

$$|H_i| \cdot \left(\frac{1}{d}\right)^{i+1} \cdot \frac{1}{|H_i|} = \left(\frac{1}{d}\right)^{i+1}$$

in the i-th iteration, and hence in total, we then get

$$|\mathfrak{T}'|_{\text{Leaf}} \leq |\mathfrak{T}(\mu)|_{\text{Leaf}} + \sum_{n=1}^{\infty} \left(\frac{1}{d}\right)^{n}$$ ■

Next, we introduce *split-nodes* for our computation trees. These are nodes that split into multiple ones without changing the set in the labeling of the node.

Definition 5.2.21 ($(\mathcal{P},\mathcal{S})$-computation tree with splits)**.** We exchange the property (b) of Definition 5.2.16 by the following:

(b-split) For every $x \in S$ with $xE = \{y_1, \ldots, y_k\}$ we have $A_x \overset{\text{i}}{\rightarrow\!\triangleright}_{\mathcal{S}} \{\frac{p_{y_1}}{p_x} : A_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : A_{y_k}\}$ or $A_{y_j} = A_x$ for all $1 \leq j \leq k$.

A chain-tree $\mathfrak{T} = (V, E, L, P)$ that satisfies the conditions from Definition 5.2.16 but with (b-split) instead of (b), is called a $(\mathcal{P},\mathcal{S})$-*computation tree with splits*. A node $x \in S$ with $xE = \{y_1, \ldots, y_k\}$, $A_{y_j} = A_x$ for all $1 \leq j \leq k$, and $A_x \overset{\text{i}}{\not\rightarrow\!\triangleright}_{\mathcal{S}} \{\frac{p_{y_1}}{p_x} : A_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : A_{y_k}\}$ is called a *split-node*.

Allowing split-nodes does not change the property of being AST as described in the following lemma

Lemma 5.2.22 (Splitting Lemma)**.** *If there exists a $(\mathcal{P},\mathcal{S})$-computation tree $\mathfrak{T}$ with splits such that $|\mathfrak{T}|_{\text{Leaf}} < 1$, then there is also a $(\mathcal{P},\mathcal{S})$-computation tree $\mathfrak{T}$ without splits such that $|\mathfrak{T}|_{\text{Leaf}} < 1$.*

Proof. The proof of this theorem combines the ideas of the proofs from Lemma 4.2.22 and Theorem 4.2.21. Assume that there exists a $(\mathcal{P},\mathcal{S})$-computation tree $\mathfrak{T} = (V, E, L, P)$ with splits such that $|\mathfrak{T}|_{\text{Leaf}} = c < 1$. From $\mathfrak{T}$, we will now create an infinite, finitely branching, labeled tree $\mathfrak{F}$ such that the label of every node X inside of this tree represents a $(\mathcal{P},\mathcal{S})$-computation tree $\mathfrak{T}_X = (V_X, E_X, L_X, P_X)$ without splits, such that $V_X \subseteq V$, and the sum of all probabilities for leaves in $\mathfrak{T}_X$ that are also leaves in $\mathfrak{T}$ is less than c (i.e., $\sum_{x \in \text{Leaf}^{\mathfrak{T}_X} \wedge x \in \text{Leaf}^{\mathfrak{T}}} p_x^{\mathfrak{T}_X} \leq c$). Since this tree $\mathfrak{F}$ is infinite and finitely branching, there must exist an infinite path by König's lemma. This infinite path will correspond to a $(\mathcal{P},\mathcal{S})$-computation tree $\mathfrak{T}_{\text{lim}}$ without splits such that $\text{Leaf}^{\mathfrak{T}_{\text{lim}}} \subseteq \text{Leaf}^{\mathfrak{T}}$ and thus we have

$$|\mathfrak{T}_{\text{lim}}|_{\text{Leaf}} = \sum_{x \in \text{Leaf}^{\mathfrak{T}_{\text{lim}}}} p_x^{\mathfrak{T}_{\text{lim}}} = \sum_{x \in \text{Leaf}^{\mathfrak{T}_{\text{lim}}} \wedge x \in \text{Leaf}^{\mathfrak{T}}} p_x^{\mathfrak{T}_{\text{lim}}} \leq c < 1$$

so that $\mathfrak{T}_{\text{lim}}$ converges with probability < 1, and this ends the proof.

Now to the precise construction of the tree $\mathfrak{F}$. The root of $\mathfrak{F}$ is labeled with the induced sub chain tree $\mathfrak{T}[\{\mathfrak{r}^{\mathfrak{T}}\}]$ of $\mathfrak{T}$ that only consists of the root. Here, $\mathfrak{T}[\{\mathfrak{r}^{\mathfrak{T}}\}]$ is a finite $(\mathcal{P},\mathcal{S})$-computation tree and $\sum_{x \in \text{Leaf}^{\mathfrak{T}[\{\mathfrak{r}^{\mathfrak{T}}\}]} \wedge x \in \text{Leaf}^{\mathfrak{T}}} p_x^{\mathfrak{T}[\{\mathfrak{r}^{\mathfrak{T}}\}]} = 0 \leq c$, since the root of $\mathfrak{T}$ can not be a leaf in $\mathfrak{T}$, otherwise, we would have $|\mathfrak{T}|_{\text{Leaf}} = 1$.

Let $x_1 x_2 \ldots$ be an enumeration of V such that there exists no $i < j$ with $\mathrm{d}(x_i) > \mathrm{d}(x_j)$. This means that our enumeration starts with the root, then enumerates all nodes of depth 1, then all nodes of depth 2, and so on.

In the i-th iteration of our construction of the tree $\mathfrak{F}$, we consider node x_i and all of the nodes in depth i of the tree $\mathfrak{F}$. Let $V_i \coloneqq \{x_1, \dots, x_i\} \cup \bigcup_{1 \leq j \leq i} x_j E$ and let $\mathfrak{T}_i \coloneqq \mathfrak{T}[V_i]$ be the induced sub $(\mathcal{P}, \mathcal{S})$-computation tree consisting of the nodes $x_1, \dots, x_i$ together with their successors. The set V_i satisfies the conditions of Definition 5.2.3 because it is weakly connected by definition of our enumeration and we either have all successors existent or none. We now use the same construction as in Theorem 4.2.21 to create a set $\mathrm{Split}(\mathfrak{T}_i)$ that contains all possible $(\mathcal{P}, \mathcal{S})$-computation trees that are contained in our $(\mathcal{P}, \mathcal{S})$-computation tree $\mathfrak{T}_i$ with splits. Instead of the nodes with a position indicator of $j \neq i$ that we used in Theorem 4.2.21 to split a tree into multiple ones, this time we split at the *split-nodes*. The rest of the construction is completely the same, and the details can be found in the proof of Theorem 4.2.21. After this construction, we get a set $\mathrm{Split}(\mathfrak{T}_i)$ that satisfies the following:

(1.) $\sum_{(p_T, T) \in \mathrm{Split}(\mathfrak{T}_i)} p_T = 1$. This means that the sum of the probabilities for all possible $(\mathcal{P}, \mathcal{S})$-computation trees is one.

(2.) For all $x \in V_i$ we have $p_x^{\mathfrak{T}_i} = \sum_{(p_T, T) \in \mathrm{Split}(\mathfrak{T}_i)} p_T \cdot \overline{p_{(T,x)}}$, where

$$\overline{p_{(T,x)}} = \begin{cases} p_x^T & \text{if } x \in V^T, \\ 0 & \text{otherwise.} \end{cases}$$

This means that the probability for node x in our tree $\mathfrak{T}_i$ is equal to the sum over all trees T that contain x, where we multiply the probability of the tree T by the probability of node x in T.

(3.) For all $(p_T, T) \in \mathrm{Split}(\mathfrak{T}_i)$ the tree T is a $(\mathcal{P}, \mathcal{S})$-computation tree without splits.

For the tree $\mathfrak{F}$, we have a node in the $(i+1)$-th depth for every tree T inside of $\mathrm{Split}(\mathfrak{T}_i)$ such that $\sum_{x \in \mathrm{Leaf}^T \wedge x \in \mathrm{Leaf}^{\mathfrak{T}}} p_x^T \leq c$ and label it accordingly. We draw an edge from a node X in depth i to the node Y in depth $i+1$ if the corresponding trees are either the same, if the tree for node Y is the result of the tree for node X if we add the successors of x_i or if the tree for node Y is the result of the tree for node X if we split at node x_i, i.e., if we remove x_i and directly move to one of its successors. A node X can only have a finite amount of direct successors in the tree $\mathfrak{F}$ since the node x_i can only have a finite amount of direct successors in $\mathfrak{T}$ so that the split at node x_i can only result in a finite amount of possible trees. If x_i is not a split-node, then there is a unique successor for each tree. Hence, $\mathfrak{F}$ is finitely branching.

We now have to prove that there exists a node in every depth of the tree $\mathfrak{F}$. Let $i \in \mathbb{N}$. We have to show that there exists a pair $(p_T, T) \in \mathrm{Split}(\mathfrak{T}_i)$ such that $\sum_{x \in \mathrm{Leaf}^T \wedge x \in \mathrm{Leaf}^{\mathfrak{T}}} p_x^T \leq c$. Assume for a contradiction that $\sum_{x \in \mathrm{Leaf}^T \wedge x \in \mathrm{Leaf}^{\mathfrak{T}}} p_x^T > c$ holds for all $(p_T, T) \in \mathrm{Split}(\mathfrak{T}_i)$. Then we would have

$$\begin{aligned}
& \sum_{x \in \mathrm{Leaf}^{\mathfrak{T}_i} \wedge x \in \mathrm{Leaf}^{\mathfrak{T}}} p_x^{\mathfrak{T}_i} \\
= \; & \sum_{x \in \mathrm{Leaf}^{\mathfrak{T}_i} \wedge x \in \mathrm{Leaf}^{\mathfrak{T}}} \sum_{(p_T, T) \in \mathrm{Split}(\mathfrak{T}_i)} p_T \cdot \overline{p_{(T,x)}} && \text{(by 2.)} \\
= \; & \sum_{(p_T, T) \in \mathrm{Split}(\mathfrak{T}_i)} \sum_{x \in \mathrm{Leaf}^{\mathfrak{T}_i} \wedge x \in \mathrm{Leaf}^{\mathfrak{T}}} p_T \cdot \overline{p_{(T,x)}} \\
= \; & \sum_{(p_T, T) \in \mathrm{Split}(\mathfrak{T}_i)} p_T \cdot \sum_{x \in \mathrm{Leaf}^{\mathfrak{T}_i} \wedge x \in \mathrm{Leaf}^{\mathfrak{T}}} \overline{p_{(T,x)}} \\
= \; & \sum_{(p_T, T) \in \mathrm{Split}(\mathfrak{T}_i)} p_T \cdot \sum_{x \in \mathrm{Leaf}^T \wedge x \in \mathrm{Leaf}^{\mathfrak{T}}} p_x^T \\
> \; & \sum_{(p_T, T) \in \mathrm{Split}(\mathfrak{T}_i)} p_T \cdot c && \text{(as } \textstyle\sum_{x \in \mathrm{Leaf}^T \wedge x \in \mathrm{Leaf}^{\mathfrak{T}}} p_x^T > c\text{)} \\
= \; & c \cdot \sum_{(p_T, T) \in \mathrm{Split}(\mathfrak{T}_i)} p_T \\
= \; & c \cdot 1 && \text{(by 1.)} \\
= \; & c
\end{aligned}$$

But this is a contradiction since this means

$$c < \sum_{x \in \mathrm{Leaf}^{\mathfrak{T}_i} \wedge x \in \mathrm{Leaf}^{\mathfrak{T}}} p_x^{\mathfrak{T}_i} \leq \sum_{x \in \mathrm{Leaf}^{\mathfrak{T}}} p_x^{\mathfrak{T}} = c$$

Hence, there exists a node in every depth of the tree $\mathfrak{F}$. Note that if a tree T is contained in depth i of the tree $\mathfrak{F}$, then for every predecessor T' of T in $\mathfrak{F}$ we also have $\sum_{x \in \mathrm{Leaf}^{T'} \wedge x \in \mathrm{Leaf}^{\mathfrak{T}}} p_x^{T'} \leq c$ and thus the whole path is contained in $\mathfrak{F}$. This means that $\mathfrak{F}$ is an infinite and finitely branching tree, so there must exist an infinite path by König's lemma.

Finally, we show that this infinite path corresponds to a $(\mathcal{P}, \mathcal{S})$-computation tree $\mathfrak{T}_{\lim}$ without splits such that $\mathrm{Leaf}^{\mathfrak{T}_{\lim}} \subseteq \mathrm{Leaf}^{\mathfrak{T}}$. Let $\mathfrak{T}_1, \mathfrak{T}_2, \ldots$ be the finite $(\mathcal{P}, \mathcal{S})$-computation trees without splits of the nodes in the infinite path in $\mathfrak{F}$. We define the tree $\mathfrak{T}_{\lim}$ by $\mathfrak{T}_{\lim} \coloneqq \lim_{i \to \infty} \mathfrak{T}_i$. Note that there is some natural number $N_j \in \mathbb{N}$ for every $j \in \mathbb{N}$, such that everything below the j-th depth is the same for all trees $\mathfrak{T}_a$ with $a > N_j$. Hence, the limit exists and all local properties for a $(\mathcal{P}, \mathcal{S})$-computation tree are satisfied for $\mathfrak{T}_{\lim}$. For the only global property regarding the infinite paths in $\mathfrak{T}_{\lim}$, note that every infinite path in $\mathfrak{T}_{\lim}$ corresponds to an infinite path in $\mathfrak{T}$ that is only missing split-nodes. Since split-nodes are contained in $\mathcal{S}$, we can be sure that every infinite path in $\mathfrak{T}_{\lim}$ must contain an infinite amount of P nodes.

Lastly, we have to show that $\mathrm{Leaf}^{\mathfrak{T}_{\lim}} \subseteq \mathrm{Leaf}^{\mathfrak{T}}$. Let $x \in \mathrm{Leaf}^{\mathfrak{T}_{\lim}}$. We have $\mathrm{d}(x) = j$ for some $j \in \mathbb{N}$ and by our previous observation, there must be a $N_j \in \mathbb{N}$ such that $x \in \mathrm{Leaf}^{\mathfrak{T}_a}$ for all $a > N_j$. But this is only possible if x is already a leaf in $\mathfrak{T}$ because otherwise, we would add one of its successors into the tree afterward, or we would skip the node x if it is a split-node. Thus we have $\mathrm{Leaf}^{\mathfrak{T}_{\lim}} \subseteq \mathrm{Leaf}^{\mathfrak{T}}$ and this ends the proof. ■

At the end of this section, we once again prove our theorem regarding the structure of a witness that $(\mathcal{P}, \mathcal{S})$ is not innermost AST. We will see that we can once again start with a fully instantiated left-hand side of some dependency tuple in the PPTRS such that every proper subterm is in normal form. We have only defined computation trees and no computation *forests*. Hence, we have implicitly defined that our probabilistic chains start with a single set.

Theorem 5.2.23 (Witness Theorem of probabilistic DP problems)**.** *If $(\mathcal{P}, \mathcal{S})$ is not innermost AST then there exists a $(\mathcal{P}, \mathcal{S})$-computation tree $\mathfrak{T} = (V, E, L, P)$ that converges with probability < 1 and starts with $(1 : \{\!\{(t^{\#}, \varepsilon)\}\!\})$ such that $t^{\#} = \ell^{\#}\sigma$ for some dependency tuple $(\ell^{\#}, \ell) \to \ldots \in \mathcal{P}$, some ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and every proper subterm of $\ell^{\#}\sigma$ is in normal form w.r.t. $\mathcal{S}$.*

In order to prove this, we first introduce two new definitions regarding the syntactical structure of our sets.

Definition 5.2.24 (Orthogonal Cuts and Hierarchical Sets)**.** Let $A \in \mathsf{PDTS}_{\mathsf{wp}}$. We say that (B, C) is an *orthogonal cut* of A if we have $B \neq \varnothing$, $C \neq \varnothing$, $A = B \uplus C$ and for all $(t_1, \pi_1) \in B$, and all $(t_2, \pi_2) \in C$ we have $\pi_1 \perp \pi_2$.

If there does not exist an orthogonal cut of A then there is a pair $(t, \pi) \in A$, called the *top pair*, such that $\pi \leq \pi'$ for all $(t', \pi') \in A$. Here, we say that A is *hierarchical*.

The core idea of an orthogonal cut is that in a rewrite step, we change only pairs of one of the parts in the cut. The other part will stay the same.

Example 5.2.25 (Rewriting Orthogonal Cuts)**.** Remember the DP problem from Example 5.1.13. For the set

$$A \coloneqq \{\!\{\left(\mathsf{f}^{\#}(\mathsf{a}), 2\right), \left(\mathsf{f}^{\#}(\mathsf{a}), 1\right), \left(\mathsf{a}^{\#}, 1.1\right)\}\!\}$$

we have an orthogonal cut of A given by the two sets

$$A_1 \coloneqq \{\!\{\left(\mathsf{f}^{\#}(\mathsf{a}), 2\right)\}\!\}, A_2 \coloneqq \{\!\{\left(\mathsf{f}^{\#}(\mathsf{a}), 1\right), \left(\mathsf{a}^{\#}, 1.1\right)\}\!\}$$

If we now look at the following rewrite step:

$$A \xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}} \{1 : B\} = \{1 : \{\!\{\left(\mathsf{f}^{\#}(\mathsf{a}), 2\right), \left(\mathsf{g}^{\#}(b), 1\right), \left(b^{\#}, 1.1\right)\}\!\}\}$$

Then we have

$$B_1 \coloneqq \{\!\{\left(\mathsf{f}^{\#}(\mathsf{a}), 2\right)\}\!\}, B_2 \coloneqq \{\!\{\left(\mathsf{g}^{\#}(b), 1\right), \left(b^{\#}, 1.1\right)\}\!\}$$

as an orthogonal cut of B. As one can see, one part of the orthogonal cut stays the same. If we remove the pair $\left(\mathsf{f}^{\#}(\mathsf{a}), 2\right)$ from A, then the resulting set

$$A' \coloneqq \{\!\{\left(\mathsf{f}^{\#}(\mathsf{a}), 1\right), \left(\mathsf{a}^{\#}, 1.1\right)\}\!\}$$

is hierarchical. Here we have the top pair $\left(\mathsf{f}^{\#}(\mathsf{a}), 1\right)$.

Lemma 5.2.26 (Orthogonal Cuts and Rewrite Steps)**.** *Let $A \in \mathsf{PDTS}_{\mathsf{wp}}$ and let (A_1, A_2) be an orthogonal cut of A. If we have*

$$A \xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}} \{p_1 : B_1, \ldots, p_k : B_k\} \text{ or } A \xrightarrow{\mathsf{i}}_{\mathcal{S}} \{p_1 : B_1, \ldots, p_k : B_k\}$$

using a main rewrite pair $(t, \pi) \in A_1$, then $A_2 \subseteq B_j$ for every $1 \leq j \leq k$.

Proof. Assume that we have $A \xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}} \{p_1 : B_1, \ldots, p_k : B_k\}$ and use the main rewrite pair $(t, \pi) \in A_1$. Let $1 \leq j \leq k$. Following the notation from Definition 5.1.12, we have

$$A = \{\!\{(t, \pi)\}\!\} \uplus M_{rew} \uplus M_{\perp} \uplus M_{<}$$

and

$$B_j = M_j^{+} \uplus M_{\perp} \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a, \chi) \in M_{rew}\}\!\}$$

Since for every $(a, \alpha) \in A_2$ we have $\alpha \perp \pi$, by definition of an orthogonal cut, we get $A_2 \subseteq M_{\perp} \subseteq B_j$. The case for $A \xrightarrow{\mathsf{i}}_{\mathcal{S}} \{p_1 : B_1, \ldots, p_k : B_k\}$ is completely analogous. ■

Now, we are able to prove the main lemma for the witness theorem.

Lemma 5.2.27. *If $(\mathcal{P}, \mathcal{S})$ is not innermost AST then there exists an innermost $(\mathcal{P}, \mathcal{S})$-computation tree $\mathfrak{T}$ that converges with probability < 1 and starts with $(1 : A)$ for some $A \in \mathsf{PDTS}_{\mathsf{wp}}$ such that $|A| = 1$.*

Proof. We will prove the contraposition. This proof follows the same structure as the proof for Theorem 2.1.23 and Theorem 4.2.21. But as we are rewriting on sets, we can not argue in the induction step about rewriting at the root position. Hence, we use our new definitions regarding orthogonal cuts and hierarchical sets to create our desired case distinction in this way. The case that there exists an orthogonal cut was in Theorem 2.1.23 and Theorem 4.2.21, the case where we do not rewrite at the root position. The case for a hierarchical set was then the second step, where we allowed rewriting at the root position again.

Assume that every $(\mathcal{P},\mathcal{S})$-computation tree $\mathfrak{T}$ that starts with $(1 : A)$ for some term $A \in \mathsf{PDTS}_{\mathsf{wp}}$ such that $|A| = 1$ converges with probability 1. We prove by induction over $|A|$, that then all $(\mathcal{P},\mathcal{S})$-computation trees $\mathfrak{T}$ that start with $(1 : A)$ for some $A \in \mathsf{PDTS}_{\mathsf{wp}}$ converge with probability 1.

If we have $|A| = 0$, then the set is in normal form and the case $|A| = 1$ is precisely our assumption. Hence, all $(\mathcal{P},\mathcal{S})$-computation trees $\mathfrak{T}$ that start with $(1 : A)$ such that $|A| \in \{0, 1\}$ converge with probability 1.

In the induction step, we assume that every $(\mathcal{P},\mathcal{S})$-computation tree $\mathfrak{T} = (V, E, L, P)$ that start with $(1 : A')$ such that $|A'| < |A|$ converges with probability 1. We now distinguish two different cases:

- If there exists an orthogonal cut (A_1, A_2) of A, then assume for a contradiction that there exists a $(\mathcal{P},\mathcal{S})$-computation tree $\mathfrak{T} = (V, E, L, P)$ that starts with $(1 : A)$ and converges with probability < 1. We will now do the following steps:

  1) Partition the set P into the sets P_1 and P_2 according to the position of the used main rewrite pair.
  2) Create a chain tree $\mathfrak{T}^{(1)}$ that starts with $(1 : A_1)$ and $|\mathfrak{T}^{(1)}|_{\text{Leaf}} < 1$.
  3) Use the P-Partition Lemma to get a $(\mathcal{P},\mathcal{S})$-computation tree $\mathfrak{T}'$ with $|\mathfrak{T}'|_{\text{Leaf}} < 1$.
  4) Create a $(\mathcal{P},\mathcal{S})$-computation tree $\mathfrak{T}'^{(2)}$ that starts with $(1 : A_2)$ and $|\mathfrak{T}^{(2)}|_{\text{Leaf}} < 1$.

  In the end, we created a $(\mathcal{P},\mathcal{S})$-computation tree $\mathfrak{T}'^{(2)}$ that starts with A_2 such that $|\mathfrak{T}'^{(2)}|_{\text{Leaf}} < 1$. This is a contradiction since we have $|A_2| < |A|$ (by definition of an orthogonal cut) and by induction hypothesis, we must have $|\mathfrak{T}'^{(2)}|_{\text{Leaf}} = 1$.

  1) Partition the set P

  Let

  $$X_1 \coloneqq \{\pi \mid (t,\pi) \in A_1 \wedge \neg\exists (t',\pi') \in A_1 : \pi' < \pi\}$$

  be the top positions in A_1 and similarly

  $$X_2 \coloneqq \{\pi \mid (t,\pi) \in A_2 \wedge \neg\exists (t',\pi') \in A_2 : \pi' < \pi\}$$

  be the top positions in A_2. Note that rewriting with $\xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}}$ or $\xrightarrow{\mathsf{i}}_{\mathcal{S}}$ can only extend existent positions. Hence, for every $x \in V$ the set A_x can only contain pairs that have a position below some position in X_1 or X_2. To be precise, we have

  $$A_x = A_x^{(1)} \uplus A_x^{(2)} \coloneqq \{(t,\pi) \in A_x \mid \exists \tau \in X_1 : \tau \leq \pi\} \uplus \{(t,\pi) \in A_x \mid \exists \tau \in X_2 : \tau \leq \pi\}$$

  If both of these are non-empty, then they form an orthogonal cut of A_x, e.g., we have $A_{\mathfrak{r}} = A = A_x^{(1)} \uplus A_x^{(2)} = A_1 \uplus A_2$ for the root and (A_1, A_2) is an orthogonal cut of A.

  We can partition P into the sets

  $$P_1 := \{x \in P \mid x \text{ together with the labeling and its successors represents a } \xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}} \text{ step with a main rewrite pair position below some } \pi \in X_1\}$$
  $$P_2 := \{x \in P \mid x \text{ together with the labeling and its successors represents a } \xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}} \text{ step with a main rewrite pair position below some } \pi \in X_2\}$$

2) Create a chain tree $\mathfrak{T}^{(1)}$

We start by constructing a chain tree $\mathfrak{T}^{(1)} = (V, E, L^{(1)}, P_1)$ with the labeling $L^{(1)}(z) = (p_z^{\mathfrak{T}}, A_z^{(1)})$ for all $z \in V$. This means that $\mathfrak{T}$ and $\mathfrak{T}^{(1)}$ have the same tree structure and the same probabilities, but we only use the sets of the first part of our orthogonal cut in the labeling. Hence, we have $|\mathfrak{T}^{(1)}|_{\text{Leaf}} = |\mathfrak{T}|_{\text{Leaf}} < 1$.

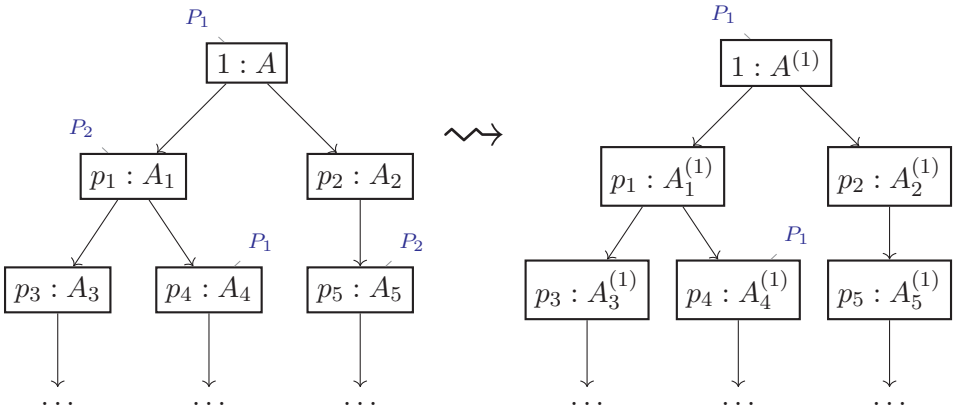


Figure 5.5: Construction of $\mathfrak{T}^{(1)}$

Rewrite steps with a main rewrite pair position below X_2 do not change the set of the first part of our orthogonal cut as described by Lemma 5.2.26. Here, we can use split-nodes to perform no rewrite step but mirror the tree structure. In order to do this, we have to remove every node from P_2 and move it into S. This means that every local property of a $(\mathcal{P}, \mathcal{S})$-computation tree is satisfied for $\mathfrak{T}^{(1)}$. However, if there exists an infinite path in $\mathfrak{T}$ that only sees nodes from P_2 and $\mathcal{S}$, then this path would not contain an infinite amount of P_1 nodes in the tree $\mathfrak{T}^{(1)}$. Hence, $\mathfrak{T}^{(1)}$ might not be a $(\mathcal{P}, \mathcal{S})$-computation tree anymore as the global property might not be satisfied. However, every induced sub chain tree such that every infinite path has an infinite amount of P_1 nodes is a $(\mathcal{P}, \mathcal{S})$-computation tree with splits. This also includes every finite induced sub chain tree as there does not exist an infinite path in such trees.

3) Use the P-Partition Lemma

In order to use the P-Partition Lemma for the tree $\mathfrak{T}$, we have to show that every induced sub chain tree $\mathfrak{T}'$ of $\mathfrak{T}$ that only contains P nodes from P_1 converges with probability 1. Let $\mathfrak{T}' = (V', E', L', P')$ be an induced sub chain tree of $\mathfrak{T}$ that does not contain nodes from P_2. There exists a set W satisfying the conditions of Definition 5.2.3 such that $\mathfrak{T}' = \mathfrak{T}[W]$. Let $w \in W$ be the root of $\mathfrak{T}'$. Since $\mathfrak{T}$ and $\mathfrak{T}^{(1)}$ have the same tree structure, we have $\mathfrak{T}^{(1)}[W]$ as an induced sub chain tree of $\mathfrak{T}^{(1)}$ with $\mathfrak{T}^{(1)}[W] = \mathfrak{T}'$. Moreover, $\mathfrak{T}^{(1)}[W]$ is a $(\mathcal{P}, \mathcal{S})$-computation tree, since the set W does not contain any inner nodes from P_2. Note that $\mathfrak{T}^{(1)}[W]$ starts with $(1 : A_w^{(1)})$ and that every $(\mathcal{P}, \mathcal{S})$-computation tree that starts with $(1 : A_1)$ converges with probability 1 by our induction hypothesis. We now show that $\mathfrak{T}^{(1)}[W]$ is the induced sub chain tree of a $(\mathcal{P}, \mathcal{S})$-computation tree that starts with $(1 : A_1)$ and thus, we know that $\mathfrak{T}^{(1)}[W]$ must be converging with probability 1 as well.

Let $\mathfrak{r} = y_1 \dots y_m = w$ be the path from the root to w in $\mathfrak{T}$. The tree we are looking for is $T \coloneqq \mathfrak{T}^{(1)}[\bigcup_{1 \le j \le m} y_j E \cup W]$. This is a grounded, induced sub chain tree. Every infinite path in T must visit the node w and corresponds to an infinite path in $\mathfrak{T}^{(1)}[W]$ so that it has an infinite amount of P_1 nodes. Furthermore, $\bigcup_{1 \le j \le m} y_j E \cup W$

satisfies the conditions of Definition 5.2.3, as the set is weakly connected and we always have the whole successor set contained or no successor at all. Therefore, T is a $(\mathcal{P}, \mathcal{S})$-computation tree that starts with A_1 and must converge with probability 1 by our induction hypothesis.

Now, we have shown that the conditions for the P-Partition Lemma (Lemma 5.2.20) are satisfied. We can now apply the P-Partition Lemma to get a grounded, induced chain tree $\mathfrak{T}'$ of $\mathfrak{T}$ with $|\mathfrak{T}'|_{\text{Leaf}} < 1$ such that on every infinite path, we will have an infinite number of P_2 nodes.

4) Create a $(\mathcal{P}, \mathcal{S})$-computation tree $\mathfrak{T}'^{(2)}$

We can now create a $(\mathcal{P}, \mathcal{S})$-computation tree $\mathfrak{T}'^{(2)} = (V', E', L'^{(2)}, P_2 \cap V')$ with splits and the labeling $L'^{(2)}(z) = (p_z^{\mathfrak{T}'}, A_z^{(2)})$ that starts with $(1 : A_2)$ and has the same underlying tree structure and probabilities for the nodes as $\mathfrak{T}'$. Since the tree structure and the probabilities are the same, we then get $|\mathfrak{T}'^{(2)}|_{\text{Leaf}} = |\mathfrak{T}'|_{\text{Leaf}}$. Since $|\mathfrak{T}'|_{\text{Leaf}} < 1$ we thus have $|\mathfrak{T}'^{(2)}|_{\text{Leaf}} < 1$, which is our desired contradiction.

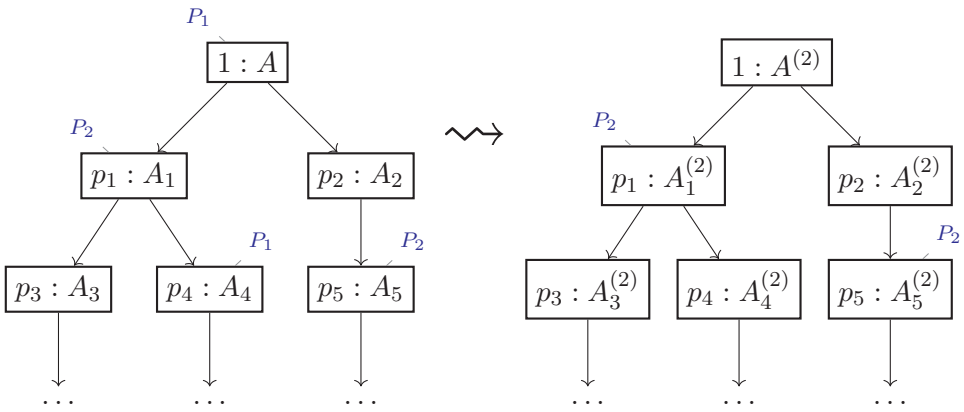


Figure 5.6: Construction of $\mathfrak{T}'^{(2)}$

The core idea of this construction is that we remove every node from P_2 and use split-nodes to do nothing, analogous to the previous construction, where we did the same for P_1. Again, all local properties for a $(\mathcal{P}, \mathcal{S})$-computation tree with splits are satisfied, and since we know that every path has an infinite amount of P_2 nodes in $\mathfrak{T}'$, we also know that the global property for $\mathfrak{T}'^{(2)}$ is satisfied.

- If we have A hierarchical, then there is a top pair (t, π) inside of A. Assume for a contradiction that there exists a $(\mathcal{P}, \mathcal{S})$-computation tree $\mathfrak{T} = (V, E, L, P)$ that starts with $(1 : A)$ and converges with probability < 1. By $\mathfrak{T}_{\mathcal{C}} = (V', E', L', P')$ we denote the $(\mathcal{P}, \mathcal{S})$-computation tree that results when cutting everything after a node that represents a rewrite step with a main rewrite pair position π. This tree can be transformed into a $(\mathcal{P}, \mathcal{S})$-computation tree $\mathfrak{T}'_{\mathcal{C}} = (V', E', L'', P')$ with the labeling $L''(x) = (p_x, \{(a, \alpha) \in A_x \mid \alpha \neq \pi\})$ that uses the same probabilities but removes the top pair (t, π) from every occurring set. Since we always rewrite with a main rewrite pair position strictly below π in $\mathfrak{T}'_{\mathcal{C}}$ (as we cut before such a rewrite step occurs), we know that $\mathfrak{T}'_{\mathcal{C}}$ is a $(\mathcal{P}, \mathcal{S})$-computation tree. Furthermore, $\mathfrak{T}'_{\mathcal{C}}$ starts with $A \setminus \{(t, \pi)\}$ and since $|A \setminus \{(t, \pi)\}| < |A|$ we get $|\mathfrak{T}'_{\mathcal{C}}|_{\text{Leaf}} = 1$ by our induction hypothesis. Now $\mathfrak{T}'_{\mathcal{C}}$ and $\mathfrak{T}_{\mathcal{C}}$ have the same tree structure and the same probabilities so that $|\mathfrak{T}_{\mathcal{C}}|_{\text{Leaf}} = |\mathfrak{T}'_{\mathcal{C}}|_{\text{Leaf}} = 1$.

  If we rewrite with the main rewrite position π, then every other pair gets removed, as every other position is below π. Hence, the leaves inside of $\mathfrak{T}'$ that are not in normal

form yet, can be extended with $(\mathcal{P},\mathcal{S})$-computation trees $\mathfrak{T}''$ that start with $\{(t',\pi)\}$ for some term $t' \in \mathcal{T}^{\#}(\Sigma)$. Again, we get $|\mathfrak{T}''|_{\text{Leaf}} = 1$ for all these computation trees by our assumption. Now, $\mathfrak{T}$ is the family extension of trees that converge with probability 1 and by the full extension lemma (Corollary 5.2.14) we get $|\mathfrak{T}|_{\text{Leaf}} = 1$, which is our desired contradiction. ■

Proof of Theorem 5.2.23. Assume that $(\mathcal{P},\mathcal{S})$ is not innermost AST. By Lemma 5.2.27 there exists a $(\mathcal{P},\mathcal{S})$-computation tree $\mathfrak{T}$ that start with $(1 : A)$ for some $A \in \mathsf{PDTS}_{\mathsf{wp}}$ with $|A| = 1$. This means that we start with the singleton set $\{\!\{(t^{\#},\pi)\}\!\}$, and hence the precise position π is irrelevant so that we can assume that we have $\pi = \varepsilon$. If the root is inside of P, then we can only perform a rewrite step if $t^{\#} = \ell^{\#}\sigma$ for some rule $(\ell^{\#},\ell) \to \ldots \in \mathcal{P}$, some ground substitution $\sigma \in \text{Sub}(\Sigma,\mathcal{V})$ and such that every proper subterm of $\ell^{\#}\sigma$ is in normal form w.r.t. $\mathcal{S}$.

If the root is not inside of $\mathcal{S}$, then we show that there exists an induced sub $(\mathcal{P},\mathcal{S})$-computation tree that starts with $(1 : A')$ for some $A' \in \mathsf{PDTS}_{\mathsf{wp}}$ with $|A| = 1$, converges with probability < 1, and such that the root is contained in P. This tree is then of our desired form by the previous paragraph again.

The grounded, induced sub $(\mathcal{P},\mathcal{S})$-computation tree $\mathfrak{T}'$ that results from cutting every edge after the first node in P must be finite. Hence, we have a finite set of leaves for $\mathfrak{T}'$, given by $\text{Leaf}^{\mathfrak{T}'} = \{z_1,\ldots,z_k\}$, and every leaf $z \in \text{Leaf}^{\mathfrak{T}'}$ is either in P or a leaf in $\mathfrak{T}$ as well. Note that rewriting with $\xrightarrow{\text{i}}_{\mathcal{S}}$ can not increase the size of the set, so that $|A_z| = |A| = 1$ for all $z \in \text{Leaf}^{\mathfrak{T}'}$ holds. Let $T_z := \mathfrak{T}[zE^*]$ be the induced sub $(\mathcal{P},\mathcal{S})$-computation tree that starts at z. If we assume for a contradiction that T_z converges with probability 1 for every $z \in \text{Leaf}^{\mathfrak{T}'}$, then also $\mathfrak{T}$ converges with probability 1 as it is the family extension of trees that converge with probability 1 and by the full extension lemma (Corollary 5.2.14) we get $|\mathfrak{T}|_{\text{Leaf}} = 1$, which is a contradiction to our assumption. Hence, there must exists a $z \in \text{Leaf}^{\mathfrak{T}'}$ such that T_z converges with probability < 1. This is then a $(\mathcal{P},\mathcal{S})$-computation tree of our desired from as we have $|A_z| = 1$ and the root is contained in P. ■

5.3 Chain Criterion

Before we talk about the DP framework and the processors, we have to prove the chain criterion for the probabilistic dependency tuples.

Theorem 5.3.1 (Probabilistic Chain Criterion)**.** *A PTRS $\mathcal{R}$ is innermost AST iff $(\mathcal{DT}(\mathcal{R}),\mathcal{R})$ is innermost AST.*

This proof is more involved than the one for the non-probabilistic dependency pairs due to the more complex definition of a probabilistic chain. We will split this proof into two smaller parts. We start by defining some important sets that we consistently use throughout both parts of the proof.

Definition 5.3.2. Let $\mathcal{R}$ be a PTRS. For a set $A \in \mathsf{PDTS}_{\mathsf{wp}}$ we define $\text{Sub}^{\text{Main}}_{\text{set}}(A,\mathcal{R}) := \{(r,\pi) \mid (r,\pi) \in A,\ r^{\flat}$ is not in normal form w.r.t. $\mathcal{R}$ and every proper subterm of $r^{\flat}$ is in normal form w.r.t. $\mathcal{R}\}$ and $\text{Sub}^{\text{Poss}}_{\text{set}}(A,\mathcal{R}) := \{(r,\pi) \mid (r,\pi) \in A,\ r^{\flat}$ is not in normal form w.r.t. $\mathcal{R}\}$. For a term t we define $\text{Sub}^{\text{Main}}_{\text{term}}(t,\mathcal{R}) := \{(r^{\#},\pi) \mid r$ is a subterm of t at position π with defined root symbol that is not in normal form w.r.t. $\mathcal{R}$ and every proper subterm of r is in normal form w.r.t. $\mathcal{R}\}$ and $\text{Sub}^{\text{Poss}}_{\text{term}}(t,\mathcal{R}) := \{(r^{\#},\pi) \mid r$ is a subterm of t at position π with defined root symbol that is not in normal form w.r.t. $\mathcal{R}\}$.

$\mathrm{Sub}^{\mathrm{Main}}_{\mathrm{set}}(A,\mathcal{R})$ and $\mathrm{Sub}^{\mathrm{Main}}_{\mathrm{term}}(t,\mathcal{R})$ contain precisely the pairs/subterms that may be used for an innermost rewrite step. To be precise, $\mathrm{Sub}^{\mathrm{Main}}_{\mathrm{set}}(A,\mathcal{R})$ contains exactly the pairs that may be used as main rewrite pair for $\xrightarrow{\mathsf{i}}_{\mathcal{DT}(\mathcal{R}),\mathcal{R}}$ steps, and $\mathrm{Sub}^{\mathrm{Main}}_{\mathrm{term}}(t,\mathcal{R})$ contains exactly the pairs where the corresponding term is a possible redex for the PTRS $\mathcal{R}$. On the other hand, the sets $\mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(A,\mathcal{R})$ and $\mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{term}}(t,\mathcal{R})$ contain all of the pairs/subterms that may be used now or in future rewrite steps as main rewrite pair/redex, due to the fact that the term has a defined root symbol and is not in normal form w.r.t. $\mathcal{R}$.

Example 5.3.3 (Important Sets)**.** Consider the PTRS $\mathcal{R}_{div}$ from Example 5.0.1. For the term $t \coloneqq \mathsf{s}(\mathsf{div}(\mathsf{minus}(x,y),\mathsf{s}(y)))$, we have

$$\begin{aligned}\mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{term}}(t,\mathcal{R}) &= \{\left(\mathsf{div}^{\#}(\mathsf{minus}(x,y),\mathsf{s}(y)),1\right),\left(\mathsf{minus}^{\#}(x,y),1.1\right)\}\\ \mathrm{Sub}^{\mathrm{Main}}_{\mathrm{term}}(t,\mathcal{R}) &= \{\left(\mathsf{minus}^{\#}(x,y),1.1\right)\}\end{aligned}$$

For the set $A \coloneqq \{\left(\mathsf{div}^{\#}(\mathsf{minus}(x,y),\mathsf{s}(y)),1\right),\left(\mathsf{minus}^{\#}(x,y),1.1\right),\left(\mathsf{div}^{\#}(\mathcal{O},\mathcal{O}),2\right)\}$, we have

$$\begin{aligned}\mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(A,\mathcal{R}) &= \{\left(\mathsf{div}^{\#}(\mathsf{minus}(x,y),\mathsf{s}(y)),1\right),\left(\mathsf{minus}^{\#}(x,y),1.1\right)\}\\ \mathrm{Sub}^{\mathrm{Main}}_{\mathrm{set}}(A,\mathcal{R}) &= \{\left(\mathsf{minus}^{\#}(x,y),1.1\right)\}\end{aligned}$$

Here, the term in the pair $\left(\mathsf{div}^{\#}(\mathcal{O},\mathcal{O}),2\right)$ is in normal form w.r.t. $\mathcal{R}$ and thus not contained in $\mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(A,\mathcal{R})$ and not contained in $\mathrm{Sub}^{\mathrm{Main}}_{\mathrm{set}}(A,\mathcal{R})$.

We can now prove the first part of our chain criterion.

Lemma 5.3.4 (Proving Innermost AST with Dependency Tuples (Part 1))**.** *Let $\mathcal{R}$ be a PTRS and $t \in \mathcal{T}(\Sigma)$ with a defined root symbol. Then the following is equivalent:*

- *There exists an innermost $\mathcal{R}$-computation tree $\mathfrak{T} = (V,E,L)$ that converges with probability < 1 and starts with $(1:t)$ such that $t = \ell\sigma$ for some rule $\ell \to r \in \mathcal{R}$, some ground substitution $\sigma \in \mathrm{Sub}(\Sigma,\mathcal{V})$, and every proper subterm of $\ell\sigma$ is in normal form w.r.t. $\mathcal{R}$.*

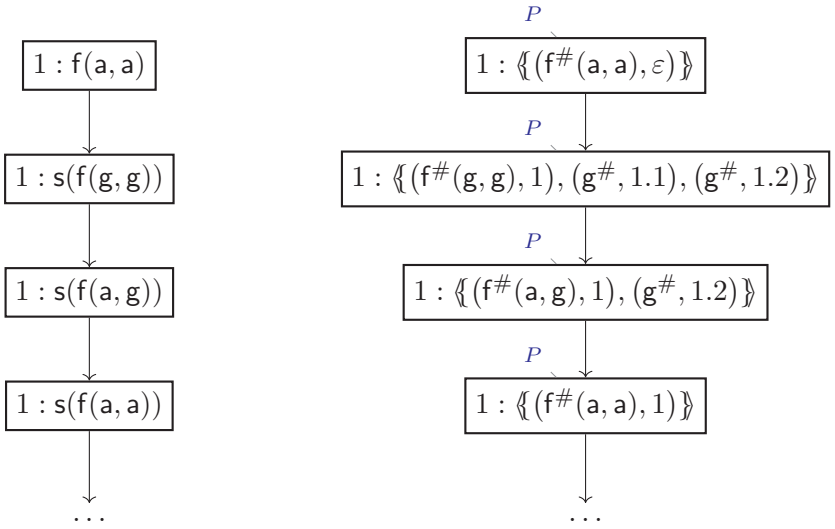
- *There exists a $(\mathcal{DT}(\mathcal{R}),\mathcal{R})$-computation tree $\mathfrak{T} = (V,E,L,P)$ that converges with probability < 1 and starts with $(1 : \{\!\{(t^{\#},\varepsilon)\}\!\})$ such that $t^{\#} = \ell^{\#}\sigma$ for some dependency tuple $(\ell^{\#},\ell) \to \ldots \in \mathcal{DT}(\mathcal{R})$, some ground substitution $\sigma \in \mathrm{Sub}(\Sigma,\mathcal{V})$, and every proper subterm of $\ell^{\#}\sigma$ is in normal form w.r.t. $\mathcal{R}$. Furthermore, we have $P = V \setminus \mathrm{Leaf}$ (i.e., we do not use any $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ steps).*

Example 5.3.5 (Illustration of Part 1)**.** Consider the following PTRS $\mathcal{R}$ over the signature Σ with $\Sigma_D = \{\mathsf{f},\mathsf{g}\}$ and $\Sigma_C = \{\mathsf{a},\mathsf{s}\}$ with the rules:

$$\begin{array}{lrcl}(1) & \mathsf{f}(\mathsf{a},\mathsf{a}) & \to & \{1:\mathsf{s}(\mathsf{f}(\mathsf{g},\mathsf{g}))\}\\ (2) & \mathsf{g} & \to & \{1:\mathsf{a}\}\end{array}$$

This is not AST, as it can be seen as a non-probabilistic TRS that does not terminate. Now, consider the PPTRS $\mathcal{DT}(\mathcal{R})$. Here, we have the dependency tuples:

$$\begin{array}{lrcl}\mathcal{DT}((1)) & (\mathsf{f}^{\#}(\mathsf{a},\mathsf{a}),\mathsf{f}(\mathsf{a},\mathsf{a})) & \to & \{1:(\{\!\{\left(\mathsf{f}^{\#}(\mathsf{g},\mathsf{g}),1\right),\left(\mathsf{g}^{\#},1.1\right),\left(\mathsf{g}^{\#},1.2\right)\}\!\},\mathsf{s}(\mathsf{f}(\mathsf{g},\mathsf{g})))\}\\ \mathcal{DT}((2)) & (\mathsf{g}^{\#},\mathsf{g}) & \to & \{1:(\varnothing,\mathsf{a})\}\end{array}$$



(a) $\mathcal{R}$-computation tree $\mathfrak{T}$ (b) $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$-computation tree $\mathfrak{T}'$

Figure 5.7: Transformation of the following proof part (1).

Consider the infinite innermost $\mathcal{R}$-computation tree $\mathfrak{T}$ in Figure 5.7a. This tree is an infinite path without leaves, so it converges with probability 0. We can construct the $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$-computation tree $\mathfrak{T}'$ depicted in Figure 5.7b that converges with probability 0 as well and only uses $\xrightarrow{\mathsf{i}}_{\mathcal{DT}(\mathcal{R}),\mathcal{R}}$ steps. If we start with $\mathfrak{T}'$ for the other direction of the proof, we will get $\mathfrak{T}$.

Proof of Lemma 5.3.4.

"⇒" Let $\mathfrak{T} = (V, E, L)$ be an innermost $\mathcal{R}$-computation tree that converges with probability < 1 and starts with t such that $t = \ell\sigma$ for some rule $\ell \to r \in \mathcal{R}$, some ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and every proper subterm of $\ell\sigma$ is in normal form w.r.t. $\mathcal{R}$. We will construct a $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$-computation tree $\mathfrak{T}' = (V, E, L', V \setminus \mathrm{Leaf}^{\mathfrak{T}})$ with the same underlying tree structure and an adjusted labeling such that $p_x^{\mathfrak{T}} = p_x^{\mathfrak{T}'}$ for all $x \in V$. Since the tree structure and the probabilities are the same, we then get $|\mathfrak{T}|_{\mathrm{Leaf}} = |\mathfrak{T}'|_{\mathrm{Leaf}}$. To be precise, the set of leaves in $\mathfrak{T}$ is equal to the set of leaves in $\mathfrak{T}'$, and they have the same probability. Since $|\mathfrak{T}|_{\mathrm{Leaf}} < 1$ we thus have $|\mathfrak{T}'|_{\mathrm{Leaf}} < 1$. Additionally, the computation tree $\mathfrak{T}'$ will start with $(1 : \{\!\{(t^{\#}, \varepsilon)\}\!\})$ and therefore, there exists a $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$-computation tree $\mathfrak{T}'$ that converges with probability < 1 with our desired properties.

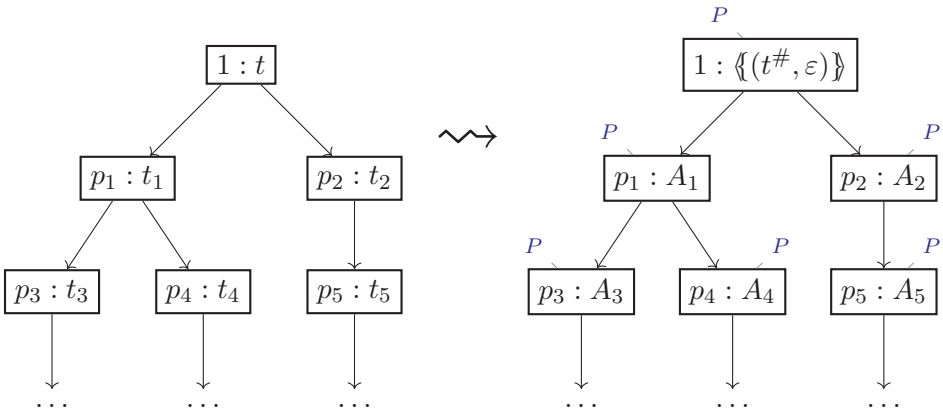


Figure 5.8: Construction for the proof of the direction "⇒"

We construct the new labeling L' for the $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$-computation tree inductively such that for all inner nodes $x \in V \setminus \text{Leaf}$ with children nodes $xE = \{y_1, \ldots, y_k\}$ we have $A_x \xrightarrow{\text{i}}_{\mathcal{DT}(\mathcal{R}),\mathcal{R}} \{\frac{p_{y_1}}{p_x} : A_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : A_{y_k}\}$. Then every property of Definition 5.2.16 is satisfied so that $\mathfrak{T}'$ is a $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$-computation tree. Let $X \subseteq V$ be the set of nodes where we have already defined the labeling $L'(x)$. During our construction, we ensure that the following property holds:

$$\text{For every node } x \in X \text{ we have } \text{Sub}^{\text{Poss}}_{\text{term}}(t_x, \mathcal{R}) \subseteq \text{Sub}^{\text{Poss}}_{\text{set}}(A_x, \mathcal{R}) \text{ and hence } \text{Sub}^{\text{Main}}_{\text{term}}(t_x, \mathcal{R}) \subseteq \text{Sub}^{\text{Main}}_{\text{set}}(A_x, \mathcal{R}). \tag{5.11}$$

This means that the corresponding term t_x for the node x in $\mathfrak{T}$ has at most the same possible redexes as we have possible main rewrite pairs for the corresponding set A_x in $\mathfrak{T}'$.

We start by setting $A_{\mathfrak{r}} \coloneqq \{\!\{(t^{\#}, \varepsilon)\}\!\}$. Here, we have $\text{Sub}^{\text{Poss}}_{\text{term}}(t_{\mathfrak{r}}, \mathcal{R}) = \text{Sub}^{\text{Poss}}_{\text{term}}(t, \mathcal{R}) = \text{Sub}^{\text{Poss}}_{\text{set}}(\{\!\{(t^{\#}, \varepsilon)\}\!\}, \mathcal{R}) = \text{Sub}^{\text{Poss}}_{\text{set}}(A_{\mathfrak{r}}, \mathcal{R})$, since every proper subterm of t is in normal form w.r.t. $\mathcal{R}$.

As long as there is still an inner node $x \in X$ such that its successors are not contained in X, we do the following. Let $xE = \{y_1, \ldots, y_k\}$ be the set of its successors. We need to define the corresponding sets $A_{y_1}, \ldots, A_{y_k}$ for the nodes $y_1, \ldots, y_k$.

Since x is not a leaf, we have $t_x \xrightarrow{\text{i}}_{\mathcal{R}} \{\frac{p_{y_1}}{p_x} : t_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : t_{y_k}\}$. This means that there is a rule $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{R}$, a position π, and a ground substitution $\sigma \in \text{Sub}(\Sigma, \mathcal{V})$ such that $t_x|_\pi = \ell\sigma$ and every proper subterm of $\ell\sigma$ is in normal form w.r.t. $\mathcal{R}$. Furthermore, we have $t_{y_j} = t_x[r_j\sigma]_\pi$ for all $1 \leq j \leq k$. So the labeling of the successors $y_1, \ldots, y_k$ in $\mathfrak{T}$ is $L(y_j) = (p_x \cdot p_j : t_x[r_j\sigma]_\pi)$ for all $1 \leq j \leq k$.

Let $\mu = \{p_1 : r_1, \ldots, p_k : r_k\}$. The corresponding dependency tuple for the rule $\ell \to \mu$ is $\mathcal{DT}(\ell \to \mu) = (\ell^{\#}, \ell) \to \{p_1 : (dp(r_1), r_1), \ldots, p_k : (dp(r_k), r_k)\}$. Furthermore, we have $(\ell^{\#}\sigma, \pi) \in \text{Sub}^{\text{Main}}_{\text{term}}(t_x, \mathcal{R}) \subseteq_{(IH)} \text{Sub}^{\text{Main}}_{\text{set}}(A_x, \mathcal{R}) \subseteq A_x$. Hence, we can rewrite A_x with $(\ell^{\#}, \ell) \to \{p_1 : (dp(r_1), r_1), \ldots, p_k : (dp(r_k), r_k)\} \in \mathcal{DT}(\mathcal{R})$, the ground substitution $\sigma \in \text{Sub}(\Sigma, \mathcal{V})$, and the main rewrite pair $(\ell^{\#}\sigma, \pi)$, since every proper subterm of $\ell^{\#}\sigma$ is in normal form w.r.t. $\mathcal{R}$. We get $A_x \xrightarrow{\text{i}}_{\mathcal{DT}(\mathcal{R}),\mathcal{R}} \{p_1 : B^x_1, \ldots, p_k : B^x_k\}$ with

$$A_x = \{\!\{(\ell^{\#}\sigma, \pi)\}\!\} \uplus M_{rew} \uplus M_{\bot} \uplus M_{<}$$

and

$$B^x_j = M^+_j \uplus M_{\bot} \uplus \{\!\{(a[r_j\sigma]_{\chi a}, \chi) \mid (a, \chi) \in M_{rew}\}\!\}$$

for all $1 \leq j \leq k$ as in Definition 5.1.12.

For our new labeling, we set $A_{y_j} \coloneqq B^x_j$. It remains to show that our induction hypothesis (5.11) is still satisfied for this new labeling, i.e., that we have $\text{Sub}^{\text{Poss}}_{\text{term}}(t_{y_j}, \mathcal{R}) \subseteq \text{Sub}^{\text{Poss}}_{\text{set}}(A_{y_j}, \mathcal{R}) \Leftrightarrow \text{Sub}^{\text{Poss}}_{\text{term}}(t_x[r_j\sigma]_\pi, \mathcal{R}) \subseteq \text{Sub}^{\text{Poss}}_{\text{set}}(B^x_j, \mathcal{R})$ for all $1 \leq j \leq k$.

Let $1 \leq j \leq k$ and $((t_x[r_j\sigma]_\pi|_\tau)^{\#}, \tau) \in \text{Sub}^{\text{Poss}}_{\text{term}}(t_x[r_j\sigma]_\pi, \mathcal{R})$. We have the following possibilities:

- If $\pi \leq \tau$, then there is a $\chi \in \mathbb{N}^*$ such that $\pi.\chi = \tau$ and we have $(t_x[r_j\sigma]_\pi|_\tau)^{\#} = (r_j\sigma|_\chi)^{\#}$. Due to the innermost strategy, we must have $\chi \in \text{Occ}(r_j)$ and $r_j|_\chi \notin \mathcal{V}$. This means that at least the defined root symbol of $r_j\sigma|_\chi$ must be inside of r_j and thus we have $((r_j|_\chi)^{\#}, \chi) \in dp(r_j)$. Thus, $((t_x[r_j\sigma]_\pi|_\tau)^{\#}, \tau) = ((r_j\sigma|_\chi)^{\#}, \pi.\chi) \in M^+_j \subseteq B^x_j$ and hence $((t_x[r_j\sigma]_\pi|_\tau)^{\#}, \tau) \in \text{Sub}^{\text{Poss}}_{\text{set}}(B^x_j, \mathcal{R})$.

- If $\tau < \pi$, then there is a $\chi \in \mathbb{N}^+$ such that $\tau.\chi = \pi$ and we have $t_x[r_j\sigma]_\pi|_\tau = t_x|_\tau[r_j\sigma]_\chi$. Furthermore, we have $((t_x|_\tau)^\#, \tau) \in \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{term}}(t_x, \mathcal{R}) \subseteq_{(IH)} \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(A_x, \mathcal{R})$, as $t_x|_\tau$ is not in normal form w.r.t. $\mathcal{R}$ (the used redex is inside of this term). Again since $\tau < \pi$, we have $((t_x|_\tau)^\#, \tau) \in M_{rew}$ and thus $((t_x|_\tau[r_j\sigma]_\chi)^\#, \tau) \in B^x_j$, which means $((t_x|_\tau[r_j\sigma]_\chi)^\#, \tau) \in \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(B^x_j, \mathcal{R})$.
- If $\tau \perp \pi$, then $t_x[r_j\sigma]_\pi|_\tau = t_x|_\tau$. Hence, we have $((t_x|_\tau)^\#, \tau) \in \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{term}}(t_x, \mathcal{R}) \subseteq_{(IH)} \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(A_x, \mathcal{R})$, as $t_x|_\tau = t_x[r_j\sigma]_\pi|_\tau$ is not in normal form w.r.t. $\mathcal{R}$ (because we have $((t_x[r_j\sigma]_\pi|_\tau)^\#, \tau) \in \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{term}}(t_x[r_j\sigma]_\pi, \mathcal{R})$). Since $\tau \perp \pi$, we have $((t_x|_\tau)^\#, \tau) \in M_\perp$ and thus $((t_x|_\tau)^\#, \tau) \in B^x_j$, which means $((t_x|_\tau)^\#, \tau) \in \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(B^x_j, \mathcal{R})$.

"$\Leftarrow$" Let $\mathfrak{T} = (V, E, L, P)$ be a $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$-computation tree that converges with probability < 1 and starts with $(1 : \{\!\{(t^\#, \varepsilon)\}\!\})$ such that $t^\# = \ell^\#\sigma$ for some dependency tuple $(\ell^\#, \ell) \to \ldots \in \mathcal{DT}(\mathcal{R})$, some ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and every proper subterm of $\ell^\#\sigma$ is in normal form w.r.t. $\mathcal{R}$. Furthermore, we have $P = V \setminus \mathrm{Leaf}$ (i.e., we do not use any $\xrightarrow{\mathrm{i}}_{\mathcal{R}}$ steps). We will construct an $\mathcal{R}$-computation tree $\mathfrak{T}' = (V, E, L')$ with the same underlying tree structure and an adjusted labeling such that $p^{\mathfrak{T}}_x = p^{\mathfrak{T}'}_x$ for all $x \in V$. Since the tree structure and the probabilities are the same, we then get $|\mathfrak{T}|_{\mathrm{Leaf}} = |\mathfrak{T}'|_{\mathrm{Leaf}}$. To be precise, the set of leaves in $\mathfrak{T}$ is equal to the set of leaves in $\mathfrak{T}'$, and they have the same probability. Since $|\mathfrak{T}|_{\mathrm{Leaf}} < 1$ we thus have $|\mathfrak{T}'|_{\mathrm{Leaf}} < 1$. Additionally, the computation tree $\mathfrak{T}'$ will start with $(1 : t)$ and thus there exists a $\mathcal{R}$-computation tree $\mathfrak{T} = (V, E, L)$ that converges with probability < 1 with our desired properties.

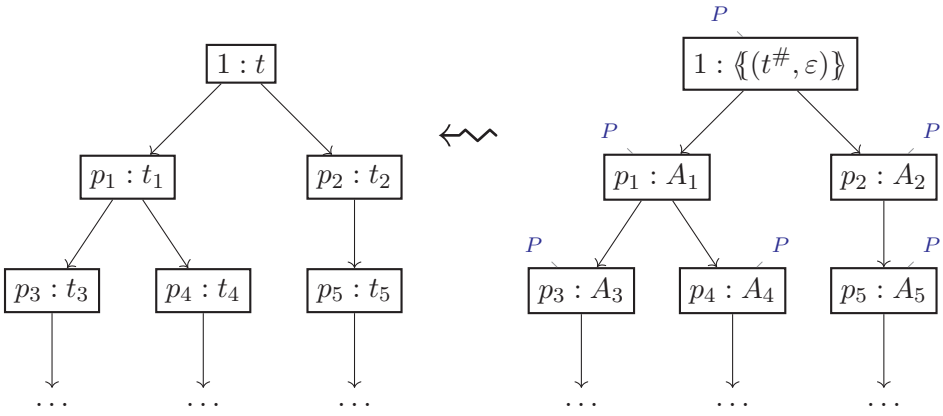


Figure 5.9: Construction for the proof of the direction "$\Leftarrow$"

We construct the new labeling L' for the $\mathcal{R}$-computation tree inductively such that for all inner nodes $x \in V \setminus \mathrm{Leaf}$ we have $t_x \xrightarrow{\mathrm{i}}_{\mathcal{R}} \{\frac{p_{y_1}}{p_x} : t_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : t_{y_k}\}$. Then every property of Definition 4.2.16 is satisfied so that $\mathfrak{T}'$ is a $\mathcal{R}$-computation tree. Let $X \subseteq V$ be the set of nodes where we have already defined the labeling $L'(x)$. During our construction, we ensure that the following property holds:

$$\text{For every node } x \in X \text{ we have } \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(A_x, \mathcal{R}) \subseteq \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{term}}(t_x, \mathcal{R}) \text{ and hence } \mathrm{Sub}^{\mathrm{Main}}_{\mathrm{set}}(A_x, \mathcal{R}) \subseteq \mathrm{Sub}^{\mathrm{Main}}_{\mathrm{term}}(t_x, \mathcal{R}). \tag{5.12}$$

This means that the corresponding set A_x for the node x in $\mathfrak{T}$ has at most the same possible main rewrite pairs as we have possible redexes for the corresponding term t_x in $\mathfrak{T}'$.

We start by setting $t_\mathfrak{r} \coloneqq t$. Here, we have $\mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(A_\mathfrak{r}, \mathcal{R}) = \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(\{\!\{(t^\#, \varepsilon)\}\!\}, \mathcal{R}) = \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{term}}(t, \mathcal{R}) = \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{term}}(t_\mathfrak{r}, \mathcal{R})$, since every proper subterm of $t^\#$ is in normal form w.r.t. $\mathcal{R}$.

As long as there is still an inner node $x \in X$ such that its successors are not contained in X, we do the following. Let $xE = \{y_1, \ldots, y_k\}$ be the set of its successors. We need to define the corresponding terms $t_{y_1}, \ldots, t_{y_k}$ for the nodes $y_1, \ldots, y_k$.

Since x is not a leaf and $\mathfrak{T}$ is a $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$-computation tree with $P = V \setminus \mathrm{Leaf}$, we have $x \in P$ and thus $A_x \overset{\mathsf{i}}{\rightarrowtriangle}_{\mathcal{DT}(\mathcal{R}),\mathcal{R}} \{\frac{p_{y_1}}{p_x} : A_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : A_{y_k}\}$. This means that there is a dependency tuple $(\ell^\#, \ell) \to \{p_1 : (dp(r_1), r_1), \ldots, p_k : (dp(r_k), r_k)\} \in \mathcal{DT}(\mathcal{R})$, a ground substitution $\sigma \in \mathrm{Sub}\,(\Sigma, \mathcal{V})$, and a main rewrite pair $(q, \pi) \in A_x$ such that $q = \ell^\#\sigma$ and all proper subterms of $\ell^\#\sigma$ are in normal form w.r.t. $\mathcal{R}$. Furthermore, we have $A_{y_k} = B^x_j$ for all $1 \leq j \leq k$ with

$$A_x = \{\!\{(q, \pi)\}\!\} \uplus M_{rew} \uplus M_\perp \uplus M_<$$

and

$$B^x_j = M^+_j \uplus M_\perp \uplus \{\!\{(a[r_j\sigma]_{\chi a}, \chi) \mid (a, \chi) \in M_{rew}\}\!\}$$

as in Definition 5.1.12. This means that the labeling of the successors $y_1, \ldots, y_k$ in $\mathfrak{T}$ is $L(y_j) = (p_x \cdot p_j : B^x_j)$ for all $1 \leq j \leq k$.

We have $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{R}$ and $(\ell^\#\sigma, \pi) \in \mathrm{Sub}^{\mathrm{Main}}_{\mathrm{term}}(A_x, \mathcal{R}) \subseteq_{(IH)} \mathrm{Sub}^{\mathrm{Main}}_{\mathrm{set}}(t_x, \mathcal{R})$. Hence, we can rewrite t_x with the rule $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\}$, the ground substitution $\sigma \in \mathrm{Sub}\,(\Sigma, \mathcal{V})$, and the position π, since $t_x|_\pi = \ell\sigma$ and every proper subterm of $\ell\sigma$ is in normal form w.r.t. $\mathcal{R}$. We get $t_x \overset{\mathsf{i}}{\to}_\mathcal{R} \{p_1 : t_x[r_1\sigma]_\pi, \ldots, p_k : t_x[r_k\sigma]_\pi\}$. For our new labeling, we set $t_{y_j} \coloneqq t_x[r_j\sigma]_\pi$. It remains to show that our induction hypothesis (5.12) is still satisfied for this new labeling, i.e., that we have $\mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(A_{y_j}, \mathcal{R}) \subseteq \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{term}}(t_{y_j}, \mathcal{R}) \Leftrightarrow \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(B^x_j, \mathcal{R}) \subseteq \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{term}}(t_x[r_j\sigma]_\pi, \mathcal{R})$ for all $1 \leq j \leq k$.

Let $1 \leq j \leq k$ and $(b, \tau) \in \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(B^x_j, \mathcal{R})$. We have the following possibilities:

- If $(b, \tau) \in M^+_j$, then we find a position $\chi \in \mathbb{N}^*$ and a term r' such that $(b, \tau) = ((r'\sigma)^\#, \pi.\chi)$ and $(r'^\#, \chi) \in dp(r_j)$. Hence, we have $t_x[r_j\sigma]_\pi|_\tau = t_x[r_j\sigma]_\pi|_{\pi.\chi} = r_j\sigma|_\chi = r'\sigma$ and thus $(b, \tau) = ((r'\sigma)^\#, \pi.\chi) \in \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{term}}(t_x[r_j\sigma]_\pi, \mathcal{R})$.
- If we have $(b, \tau) \in \{\!\{(a[r_j\sigma]_{\chi a}, \tau) \mid (a, \tau) \in M_{rew}\}\!\}$, then we have a pair $(a, \tau) \in A_x$ and a position $\chi \in \mathbb{N}^+$ with $\tau.\chi = \pi$. By definition, the same rule that we applied to t is applied to a at position χ, so that we result in $b = a[r_j\sigma]_\chi$. Note that $(a, \tau) \in \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(A_x, \mathcal{R}) \subseteq_{(IH)} \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{term}}(t_x, \mathcal{R})$, since a is not in normal form w.r.t. $\mathcal{R}$ (the used main rewrite pair has a position below τ). Hence, we have $(t_x[r_j\sigma]_\pi|_\tau)^\# = (t_x|_\tau[r_j\sigma]_\chi)^\# = (t_x|_\tau)^\#[r_j\sigma]_\chi = a[r_j\sigma]_\chi = b$ and thus $(b, \tau) \in \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{term}}(t_x[r_j\sigma]_\pi, \mathcal{R})$.
- If we have $(b, \tau) \in M_\perp$, then we have $(b, \tau) \in A_x$ and $(b, \tau) \in \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(A_x, \mathcal{R}) \subseteq_{(IH)} \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{term}}(t_x, \mathcal{R})$, as $b^\flat$ is not in normal form w.r.t. $\mathcal{R}$ (because we have $(b, \tau) \in \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(B^x_j, \mathcal{R})$). Since $(b, \tau) \in M_\perp$, we have $\tau \perp \pi$ and $t_x[r_j\sigma]_\pi|_\tau = t_x|_\tau$. Hence, $(b, \tau) \in \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{term}}(t_x[r_j\sigma]_\pi, \mathcal{R})$.

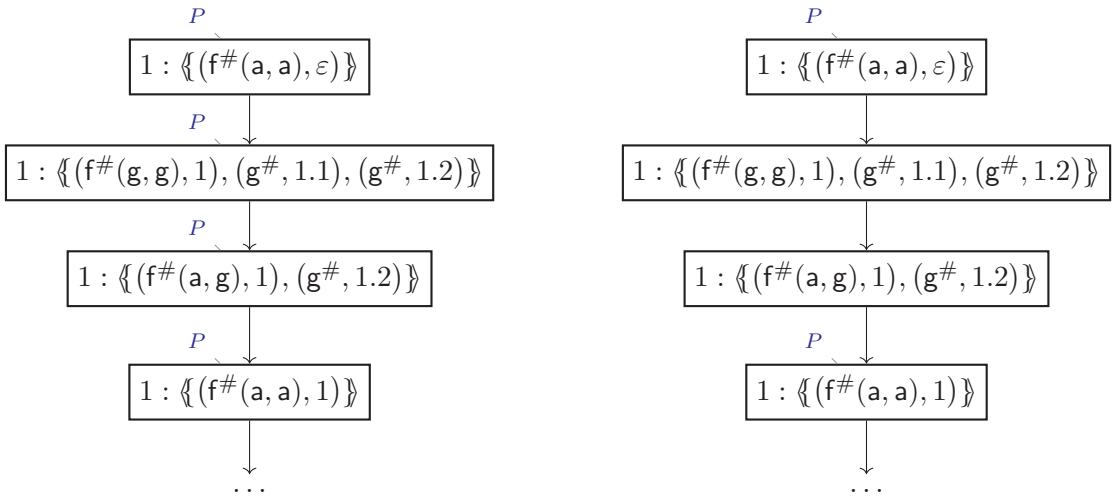
■

In our second part of the chain criterion, we now allow steps with $\overset{\mathsf{i}}{\rightarrowtriangle}_\mathcal{R}$.

Lemma 5.3.6 (Proving Innermost AST with Dependency Tuples (2))**.** *Let $\mathcal{R}$ be a PTRS and $t \in \mathcal{T}(\Sigma)$ with a defined root symbol. Then the following is equivalent:*

- *There exists a* $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$*-computation tree* $\mathfrak{T} = (V, E, L, P)$ *that converges with probability* < 1 *and starts with* $(1 : \{\!\{(t^\#, \varepsilon)\}\!\})$ *such that* $t^\# = \ell^\#\sigma$ *for some dependency tuple* $(\ell^\#, \ell) \to \ldots \in \mathcal{DT}(\mathcal{R})$*, some ground substitution* $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$*, and every proper subterm of* $\ell^\#\sigma$ *is in normal form w.r.t.* $\mathcal{R}$*. Furthermore, we have* $P = V \setminus \mathrm{Leaf}$ *(i.e., we do not use any* $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ *steps).*
- *There exists a* $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$*-computation tree* $\mathfrak{T} = (V, E, L, P)$ *that converges with probability* < 1 *and starts with* $(1 : \{\!\{(t^\#, \varepsilon)\}\!\})$ *such that* $t^\# = \ell^\#\sigma$ *for some dependency tuple* $(\ell^\#, \ell) \to \ldots \in \mathcal{DT}(\mathcal{R})$*, some ground substitution* $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$*, and every proper subterm of* $\ell^\#\sigma$ *is in normal form w.r.t.* $\mathcal{R}$*. Here, we also allow steps with* $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$*.*

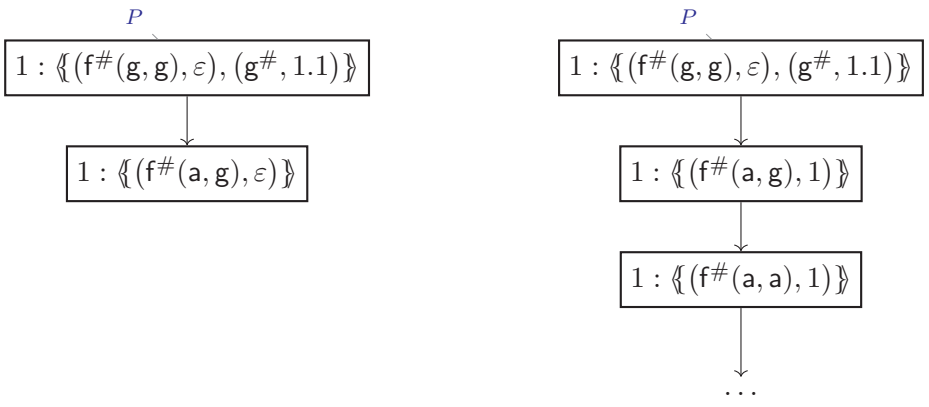
Example 5.3.7 (Illustration of Part 2)**.** Again, consider the PTRS $\mathcal{R}$ and the PPTRS $\mathcal{DT}(\mathcal{R})$ from Example 5.3.5. The idea for the following construction is that everything that is possible with $\mathcal{R}$ is also possible with the second component of $\mathcal{DT}(\mathcal{R})$.



(a) $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$-comp. tree $\mathfrak{T}$ without $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ (b) $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$-comp. tree $\mathfrak{T}'$ with $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$

Figure 5.10: Transformation of the following proof part (2).

In Figure 5.10 one can see a $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$-computation tree with and one without $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ steps. In this case, there is no difference between both computation trees. However, in Figure 5.11, we can see that this is not always the case. Here, the $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$-computation tree that does not use $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ steps is not able to mirror the rewrite steps with $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$.



(a) $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$-comp. tree $\mathfrak{T}$ without $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ (b) $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$-comp. tree $\mathfrak{T}'$ with $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$

Figure 5.11: Failing case for the transformation.

The set $\{\!\{\left(\mathsf{f}^{\#}(\mathsf{a},\mathsf{g}),\varepsilon\right)\}\!\}$ is in normal form w.r.t. $\xrightarrow{\mathsf{i}}_{\mathcal{DT}(\mathcal{R}),\mathcal{R}}$ but we can still use $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ steps to rewrite the g of the term in the pair $\left(\mathsf{f}^{\#}(\mathsf{a},\mathsf{g}),\varepsilon\right)$. The reason for this is that while the PTRS rule does the same as the projection to the second component of $\mathcal{DT}(\mathcal{R})$, we still need the pair $\left(\mathsf{g}^{\#},1.2\right)$ to be existent in the set in order to use the PPTRS rule. The idea of the following proof is that if we start with $(1 : \{\!\{(t,\pi)\}\!\})$ such that $t = \ell^{\#}\sigma$ for some PPTRS rule with left-hand side $(\ell^{\#},\ell)$ and some ground substitution $\sigma \in \mathrm{Sub}\,(\Sigma,\mathcal{V})$, and we only use PPTRS rules to rewrite the terms, then we can be sure that the required main rewrite pair always exists to mirror the $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ step with the second component of $\mathcal{DT}(\mathcal{R})$.

Proof of Lemma 5.3.6.

"$\Rightarrow$" Every $(\mathcal{DT}(\mathcal{R}),\mathcal{R})$-computation tree that does not use any $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ steps can also be seen as a $(\mathcal{DT}(\mathcal{R}),\mathcal{R})$-computation tree in general.

"$\Leftarrow$" Let $\mathfrak{T} = (V,E,L,P)$ be a $(\mathcal{DT}(\mathcal{R}),\mathcal{R})$-computation tree that converges with probability < 1 and starts with $(1 : \{\!\{(t^{\#},\varepsilon)\}\!\})$ such that $t^{\#} = \ell^{\#}\sigma$ for some dependency tuple $(\ell^{\#},\ell) \to \ldots \in \mathcal{DT}(\mathcal{R})$, some ground substitution $\sigma \in \mathrm{Sub}\,(\Sigma,\mathcal{V})$, and every proper subterm of $\ell^{\#}\sigma$ is in normal form w.r.t. $\mathcal{R}$. We will now create a $(\mathcal{DT}(\mathcal{R}),\mathcal{R})$-computation tree $\mathfrak{T}' = (V,E,L',P')$ that also starts with $(1 : \{\!\{(t^{\#},\varepsilon)\}\!\})$, with the same underlying tree structure, and an adjusted labeling such that $p_x^{\mathfrak{T}} = p_x^{\mathfrak{T}'}$ for all $x \in V$. Since the tree structure and the probabilities are the same, we then get $|\mathfrak{T}|_{\mathrm{Leaf}} = |\mathfrak{T}'|_{\mathrm{Leaf}}$. To be precise, the set of leaves in $\mathfrak{T}$ is equal to the set of leaves in $\mathfrak{T}'$, and they have the same probability. Since $|\mathfrak{T}|_{\mathrm{Leaf}} < 1$ we thus have $|\mathfrak{T}'|_{\mathrm{Leaf}} < 1$. Furthermore, we will have $P' = V \setminus \mathrm{Leaf}^{\mathfrak{T}'}$ so that $\mathfrak{T}'$ does not use any $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ steps. Therefore, there exists a $(\mathcal{DT}(\mathcal{R}),\mathcal{R})$-computation tree with our desired properties.

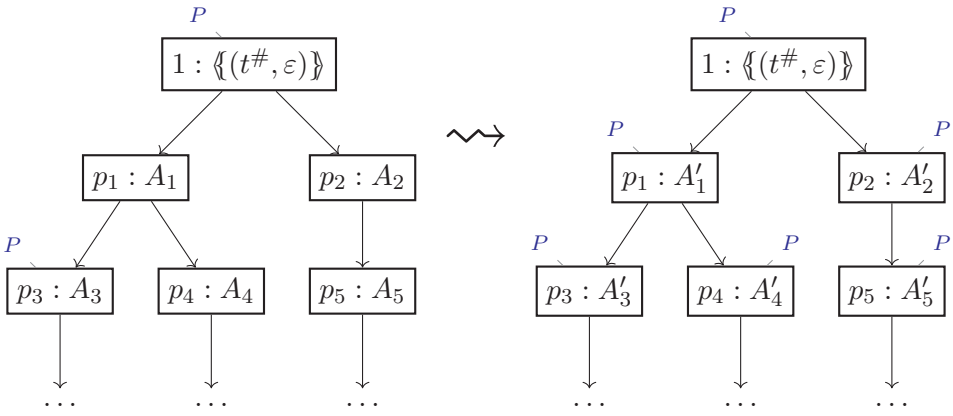


Figure 5.12: Construction for the proof of the direction "$\Leftarrow$"

The core idea of this construction is that everything that is possible with $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ can also be done with the second component of $\mathcal{DT}(\mathcal{R})$, as described in Example 5.3.7.

We construct the new labeling L' for the $(\mathcal{DT}(\mathcal{R}),\mathcal{R})$-computation tree $\mathfrak{T}'$ inductively such that for all inner nodes $x \in V \setminus \mathrm{Leaf}$ we have $A'_x \xrightarrow{\mathsf{i}}_{\mathcal{DT}(\mathcal{R}),\mathcal{R}} \{\frac{p_{y_1}}{p_x} : A'_{y_1},\ldots,\frac{p_{y_k}}{p_x} : A'_{y_k}\}$. Then every property of Definition 5.2.16 is satisfied so that $\mathfrak{T}'$ is a $(\mathcal{DT}(\mathcal{R}),\mathcal{R})$-computation tree. Let $X \subseteq V$ be the set of nodes where we have already defined the labeling $L'(x)$. During our construction, we ensure that the following property holds:

$$\text{For every node } x \in X \text{ we have } \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(\mathsf{PosEx}(A_x),\mathcal{R}) \subseteq A'_x \text{ and hence } \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(\mathsf{PosEx}(A_x),\mathcal{R}) \subseteq \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{term}}(A'_x,\mathcal{R}) \text{ and } \mathrm{Sub}^{\mathrm{Main}}_{\mathrm{set}}(\mathsf{PosEx}(A_x),\mathcal{R}) \subseteq \mathrm{Sub}^{\mathrm{Main}}_{\mathrm{term}}(A'_x,\mathcal{R}). \tag{5.13}$$

Here, the position extension of a $A \in \mathsf{PDTS}_{\mathsf{wp}}$, denoted by $\mathsf{PosEx}(A)$ is defined as

$$\mathsf{PosEx}(A) := A \cup \{(t|_\tau^\#, \pi.\tau) \mid (t^\#, \pi) \in A \ \wedge \ \tau \in \mathbb{N}^+ \ \wedge \ \mathrm{root}(t|_\tau) \in \Sigma_D\}$$

The position extension $\mathsf{PosEx}(A)$ contains all pairs that are missing in A, but the corresponding term has a defined root symbol and is a proper subterm of some other pair in A. These pairs might be needed to mirror a rewrite step with $\mathcal{R}$ using the second component of $\mathcal{DT}(\mathcal{R})$. This means that the corresponding set A'_x for the node x in $\mathfrak{T}'$ contains every possible main rewrite pair that we might need to use in order to mirror the rewrite steps applied to the set A_x in $\mathfrak{T}$.

For example, if we consider our signature $\Sigma = \{\mathsf{div}, \mathsf{minus}, \mathsf{s}, \mathcal{O}\}$ with $\Sigma_D = \{\mathsf{div}, \mathsf{minus}\}$ and $\Sigma_C = \{\mathsf{s}, \mathcal{O}\}$, and the set $A \coloneqq \{\!\{(\mathsf{div}^\#(\mathsf{minus}(x, y), \mathsf{s}(y)), 1)\}\!\}$, then we have

$$\mathsf{PosEx}(A) = \{\!\{(\mathsf{div}^\#(\mathsf{minus}(x, y), \mathsf{s}(y)), 1), (\mathsf{minus}^\#(x, y), 1.1)\}\!\}$$

If we would rewrite the inner minus in A with $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$, then we are able to mirror this rewrite step in $\mathsf{PosEx}(A)$ with $\xrightarrow{\mathsf{i}}_{\mathcal{DT}(\mathcal{R}),\mathcal{R}}$ using the main rewrite pair $(\mathsf{minus}^\#(x, y), 1)$.

We start by setting $A'_{\mathfrak{r}} \coloneqq A_{\mathfrak{r}} = \{\!\{(t^\#, \varepsilon)\}\!\}$. Here, we have $\mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(\mathsf{PosEx}(A_{\mathfrak{r}}), \mathcal{R}) = \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(\mathsf{PosEx}(\{\!\{(t^\#, \varepsilon)\}\!\}), \mathcal{R}) = \{\!\{(t^\#, \varepsilon)\}\!\} = A'_{\mathfrak{r}}$, since every proper subterm of t is in normal form w.r.t. $\mathcal{R}$.

As long as there is still an inner node $x \in X$ such that its successors are not contained in X, we do the following. Let $xE = \{y_1, \ldots, y_k\}$ be the set of its successors. We need to define the corresponding sets $A'_{y_1}, \ldots, A'_{y_k}$ for the nodes $y_1, \ldots, y_k$.

Since x is not a leaf and $\mathfrak{T}$ is a $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$-computation tree, we have $A_x \xrightarrow{\mathsf{i}}_{\mathcal{DT}(\mathcal{R}),\mathcal{R}} \{\frac{p_{y_1}}{p_x} : A_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : A_{y_k}\}$ (or $A_x \xrightarrow{\mathsf{i}}_{\mathcal{R}} \{\frac{p_{y_1}}{p_x} : A_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : A_{y_k}\}$).

If we have $A_x \xrightarrow{\mathsf{i}}_{\mathcal{DT}(\mathcal{R}),\mathcal{R}} \{\frac{p_{y_1}}{p_x} : A_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : A_{y_k}\}$, then there is a dependency tuple $(\ell^\#, \ell) \to \{p_1 : (dp(r_1), r_1), \ldots, p_k : (dp(r_k), r_k)\} \in \mathcal{DT}(\mathcal{R})$, a main rewrite pair $(q, \pi) \in A_x$, and a ground substitution $\sigma \in \mathrm{Sub}\,(\Sigma, \mathcal{V})$ such that $q = \ell^\#\sigma$, and all proper subterms of $\ell^\#\sigma$ are in normal form w.r.t. $\mathcal{R}$. Furthermore, we have

$$A_x = \{\!\{(q, \pi)\}\!\} \uplus M_{rew} \uplus M_\perp \uplus M_<$$

and

$$A_{y_j} = M_j^+ \uplus M_\perp \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a, \chi) \in M_{rew}\}\!\}$$

for all $1 \leq j \leq k$ as in Definition 5.1.12.

Since we have $(q, \pi) \in A_x$, we get $(q, \pi) \in \mathsf{PosEx}(A_x)$ and since $q^\flat$ is not in normal form w.r.t. $\mathcal{R}$ but every proper subterm of $q^\flat$ is in normal form w.r.t. $\mathcal{R}$, we also get $(q, \pi) \in \mathrm{Sub}^{\mathrm{Main}}_{\mathrm{set}}(\mathsf{PosEx}(A_x), \mathcal{R}) \subseteq_{(IH)} A'_x$. Hence, we can also apply the dependency tuple $(\ell^\#, \ell) \to \{p_1 : (dp(r_1), r_1), \ldots, p_k : (dp(r_k), r_k)\} \in \mathcal{DT}(\mathcal{R})$ to A'_x using the main rewrite pair (q, π) and the ground substitution σ since $q = \ell^\#\sigma$ and every proper subterm of $\ell^\#\sigma$ is in normal form w.r.t. $\mathcal{R}$, so that we have $A'_x \xrightarrow{\mathsf{i}}_{\mathcal{DT}(\mathcal{R}),\mathcal{R}} \{p_1 : B_1, \ldots, p_k : B_k\}$ with

$$A'_x = \{\!\{(q, \pi)\}\!\} \uplus M'_{rew} \uplus M'_\perp \uplus M'_<$$

and

$$B_j^x = M_j'^+ \uplus M'_\perp \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a, \chi) \in M'_{rew}\}\!\}$$

for all $1 \leq j \leq k$ as in Definition 5.1.12. Then we set $A'_{y_j} \coloneqq B_j^x$ for all $1 \leq j \leq k$. It remains to prove that our claim (5.13) still holds, i.e., we have $\mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(\mathsf{PosEx}(A_{y_j}), \mathcal{R}) \subseteq A'_{y_j}$ for all $1 \leq j \leq k$.

Let $1 \leq j \leq k$. We have to show that for every $(b^\#, \zeta_1) \in A_{y_j}$ and every position $\zeta_2 \in \mathbb{N}^*$ such that $b|_{\zeta_2}$ has defined root symbol and is not in normal form w.r.t. $\mathcal{R}$, we have $(b|^\#_{\zeta_2}, \zeta_1.\zeta_2) \in A'_{y_j}$. We distinguish according to the origin of $(b^\#, \zeta_1) \in A_{y_j}$.

- If we have $(b^\#, \zeta_1) \in M_j^+$, then we find a position $\chi \in \mathbb{N}^*$ and a term r' such that $(b^\#, \tau) = (r'^\#\sigma, \pi.\chi)$ and $(r'^\#, \chi) \in dp(r_j)$. Due to the innermost strategy, we know that $b|_{\zeta_2}$ cannot be completely inside the substitution σ, but at least the defined root symbol must be introduced by r_j. Therefore, we have $b|_{\zeta_2} = r'\sigma|_{\zeta_2} = r'|_{\zeta_2}\sigma = r_j|_\chi|_{\zeta_2}\sigma = r_j|_{\chi.\zeta_2}\sigma$ and we must have $(r_j|^\#_{\chi.\zeta_2}, \chi.\zeta_2) \in dp(r_j)$. Since we use the same dependency tuple, the same substitution and the same main rewrite pair, we also have $(b|^\#_{\zeta_2}, \zeta_1.\zeta_2) = (r_j|^\#_{\chi.\zeta_2}\sigma, \pi.\chi.\zeta_2) \in M_j'^+ \subseteq A'_{y_j}$.
- If we have $(b^\#, \zeta_1) \in \{\!\{(a[r_j\sigma]_{\chi_a.\rho}, \tau) \mid (a, \tau) \in M_{rew}\}\!\}$, then there is a pair $(a^\#, \zeta_1) \in A_x$, and a position $\chi_1 \in \mathbb{N}^+$ such that $\zeta_1.\chi_1 = \pi$ and $b^\# = a^\#[r_j\sigma]_{\chi_1}$. Furthermore, we have $(a^\#, \zeta_1) \in \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(\mathsf{PosEx}(A_x), \mathcal{R}) \subseteq_{(IH)} A'_x$.
  - If we have $\pi \leq \zeta_1.\zeta_2$, then we can find a position $\chi_2 \in \mathbb{N}^*$ such that $\pi.\chi_2 = \zeta_1.\zeta_2$. Since $\zeta_1.\chi_1 = \pi$ and $\pi.\chi_2 = \zeta_1.\zeta_2$ we get $\chi_1.\chi_2 = \zeta_2$ and thus $b|_{\zeta_2} = a[r_j\sigma]_{\chi_1}|_{\zeta_2} = r_j\sigma|_{\chi_2}$ with defined root and not in normal form w.r.t. $\mathcal{R}$. Due to the innermost strategy, we know that $b|_{\zeta_2}$ cannot be completely inside the substitution σ, but at least the defined root symbol must be introduced by r_j. Thus, we have $b|_{\zeta_2} = r_j\sigma|_{\chi_2} = r_j|_{\chi_2}\sigma$ and $(r_j|^\#_{\chi_2}, \chi_2) \in dp(r_j)$. Since we use the same dependency tuple, the same substitution and the same main rewrite pair, we also have $(b|^\#_{\zeta_2}, \zeta_1.\zeta_2) = (r_j|^\#_{\chi_2}\sigma, \pi.\chi_2) \in M_j'^+ \subseteq A'_{y_j}$.
  - If we have $\zeta_1.\zeta_2 < \pi$, then we can find a position $\chi_2 \in \mathbb{N}^+$ such that $\zeta_1.\zeta_2.\chi_2 = \pi$. Then $b|_{\zeta_2} = a[r_j\sigma]_{\chi_1}|_{\zeta_2} = a|_{\zeta_2}[r_j\sigma]_{\chi_2}$. We have $(a|^\#_{\zeta_2}, \zeta_1.\zeta_2) \in \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(\mathsf{PosEx}(A_x), \mathcal{R}) \subseteq_{(IH)} A'_x$. Thus, we also have $(b|^\#_{\zeta_2}, \zeta_1.\zeta_2) = ((a|_{\zeta_2}[r_j\sigma]_{\chi_2})^\#, \zeta_1.\zeta_2) = (a|^\#_{\zeta_2}[r_j\sigma]_{\chi_2}, \zeta_1.\zeta_2) \in M'_{rew} \subseteq A'_{y_j}$.
  - If we have $\pi \perp \zeta_1.\zeta_2$, then we have $b|_{\zeta_2} = a[r_j\sigma]_{\chi_1}|_{\zeta_2} = a|_{\zeta_2}$ and $\mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(\mathsf{PosEx}(A_x), \mathcal{R}) \subseteq_{(IH)} A'_x$. Since $\pi \perp \zeta_1.\zeta_2$, we get $(b|^\#_{\zeta_2}, \zeta_1.\zeta_2) = (a|^\#_{\zeta_2}, \zeta_1.\zeta_2) \in M'_\perp \subseteq A'_{y_j}$.
- If we have $(b^\#, \zeta_1) \in M_\perp$, then we have $\zeta_1 \perp \pi$ and $(b^\#, \zeta_1) \in A_x$. Then, $(b|^\#_{\zeta_2}, \zeta_1.\zeta_2) \in \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(\mathsf{PosEx}(A_x), \mathcal{R}) \subseteq_{(IH)} A'_x$. Since we have $\zeta_1 \perp \pi$ we also have $\zeta_1.\zeta_2 \perp \pi$ and thus $(b|^\#_{\zeta_2}, \zeta_1.\zeta_2) \in M'_\perp \subseteq A'_{y_j}$.

Thus, we have $\mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(\mathsf{PosEx}(A_{y_j}), \mathcal{R}) \subseteq A'_{y_j}$.

If we have $A_x \xrightarrow{\mathsf{i}}_{\mathcal{R}} \{\frac{p_{y_1}}{p_x} : A_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : A_{y_k}\}$, then there is a rule $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{R}$, a main rewrite pair $(q, \pi) \in A_x$, a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and a non-empty position $\tau \in \mathbb{N}^+$ with $q|_\tau = \ell\sigma$, such that all proper subterms of $\ell\sigma$ are in normal form w.r.t. $\mathcal{R}$. Furthermore, we have

$$A_x = \{\!\{(q, \pi)\}\!\} \uplus M_{rew} \uplus M_\perp \uplus M_<$$

and

$$A_{y_j} = M_j^+ \uplus M_\perp \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a, \chi) \in M_{rew}\}\!\}$$

for all $1 \leq j \leq k$ as in Definition 5.1.17.

Since we have $\mathrm{root}(q|_\tau) \in \Sigma_D$ and $(q, \pi) \in A_x$, we get $(q|^\#_\tau, \pi.\tau) \in \mathsf{PosEx}(A_x)$ and since $q|_\tau$ is not in normal form w.r.t. $\mathcal{R}$ but every proper subterm of $q|_\tau$ is in normal form w.r.t. $\mathcal{R}$, we also get $(q|^\#_\tau, \pi.\tau) \in \mathrm{Sub}^{\mathrm{Main}}_{\mathrm{set}}(\mathsf{PosEx}(A_x), \mathcal{R}) \subseteq_{(IH)} \mathrm{Sub}^{\mathrm{Main}}_{\mathrm{set}}(A'_x, \mathcal{R})$.

Using the same substitution σ, we get $q|_\tau^\# = \ell^\#\sigma$ and all proper subterms of $\ell^\#\sigma$ are in normal form w.r.t. $\mathcal{R}$. Hence, we can apply the dependency tuple $(\ell^\#, \ell) \to \{p_1 : (dp(r_1), r_1), \ldots, p_k : (dp(r_k), r_k)\} \in \mathcal{DT}(\mathcal{R})$ to A'_x, so that we have $A'_x \xrightarrow{\mathsf{i}}_{\mathcal{DT}(\mathcal{R}),\mathcal{R}} \{p_1 : B_1, \ldots, p_k : B_k\}$ with

$$A'_x = \{\!\{(q, \pi)\}\!\} \uplus M'_{rew} \uplus M'_\perp \uplus M'_<$$

and

$$B^x_j = M'^+_j \uplus M'_\perp \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a, \chi) \in M'_{rew}\}\!\}$$

for all $1 \leq j \leq k$ as in Definition 5.1.12. Then we set $A'_{y_j} \coloneqq B^x_j$ for all $1 \leq j \leq k$. It remains to prove that our claim (5.13) still holds, i.e., that we have $\mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(\mathsf{PosEx}(A_{y_j}), \mathcal{R}) \subseteq A'_{y_j}$ for all $1 \leq j \leq k$.

Let $1 \leq j \leq k$. We have to show that for every $(b^\#, \zeta_1) \in A_{y_j}$ and every position ζ_2 such that $b|_{\zeta_2}$ has defined root symbol and is not in normal form w.r.t. $\mathcal{R}$, we have $(b|^\#_{\zeta_2}, \zeta_1.\zeta_2) \in A'_{y_j}$. We distinguish according to the origin of $(b^\#, \zeta_1) \in A_{y_j}$.

- If we have $(b^\#, \zeta_1) \in M^+_j \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a, \chi) \in M_{rew}\}\!\} = \{(q[r_j\sigma]_\tau, \pi)\} \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a, \chi) \in M_{rew}\}\!\}$, then there is a pair $(a^\#, \zeta_1) \in A_x$, and position $\chi_1 \in \mathbb{N}^*$ such that we have $\zeta_1.\chi_1 = \pi$ and $b^\# = a^\#[r_j\sigma]_{\chi_1.\tau}$. Furthermore, we have $(a^\#, \zeta_1) \in \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(\mathsf{PosEx}(A_x), \mathcal{R}) \subseteq_{(IH)} A'_x$.
  - If we have $\pi.\tau \leq \zeta_1.\zeta_2$, then we can find a position $\chi_2 \in \mathbb{N}^*$ such that $\pi.\tau.\chi_2 = \zeta_1.\zeta_2$. Since $\zeta_1.\chi_1 = \pi$ and $\pi.\tau.\chi_2 = \zeta_1.\zeta_2$ we get $\chi_1.\tau.\chi_2 = \zeta_2$ and thus $b|_{\zeta_2} = a[r_j\sigma]_{\chi_1.\tau}|_{\zeta_2} = r_j\sigma|_{\chi_2}$ with defined root and not in normal form w.r.t. $\mathcal{R}$. Due to the innermost strategy we know that $b|_{\zeta_2}$ cannot be completely inside the substitution σ but at least the defined root symbol must be introduced by r_j, so that $r_j\sigma|_{\chi_2} = r_j|_{\chi_2}\sigma$. Therefore, we have $(r_j|^\#_{\chi_2}, \chi_2) \in dp(r_j)$ and thus $(b|^\#_{\zeta_2}, \zeta_1.\zeta_2) = (r_j|^\#_{\chi_2}\sigma, \pi.\chi_2) \in M'^+_j \subseteq A'_{y_j}$.
  - If we have $\zeta_1.\zeta_2 < \pi.\tau$, then we can find a position $\chi_2 \in \mathbb{N}^+$ such that $\zeta_1.\zeta_2.\chi_2 = \pi.\tau$. Then $b|_{\zeta_2} = a[r_j\sigma]_{\chi_1.\tau}|_{\zeta_2} = a|_{\zeta_2}[r_j\sigma]_{\chi_2}$. We have $(a|^\#_{\zeta_2}, \zeta_1.\zeta_2) \in \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(\mathsf{PosEx}(A_x), \mathcal{R}) \subseteq_{(IH)} A'_x$. Thus, we also have $(b|^\#_{\zeta_2}, \zeta_1.\zeta_2) = ((a|_{\zeta_2}[r_j\sigma]_{\chi_2})^\#, \zeta_1.\zeta_2) = (a|^\#_{\zeta_2}[r_j\sigma]_{\chi_2}, \zeta_1.\zeta_2) \in M'_{rew} \subseteq A'_{y_j}$.
  - If we have $\pi.\tau \perp \zeta_1.\zeta_2$, then we have $b|_{\zeta_2} = a[r_j\sigma]_{\chi_1}|_{\zeta_2} = a|_{\zeta_2}$. Furthermore, we have $(a|^\#_{\zeta_2}, \zeta_1.\zeta_2) \in \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(\mathsf{PosEx}(A_x), \mathcal{R}) \subseteq_{(IH)} A'_x$. Since $\pi.\tau \perp \zeta_1.\zeta_2$, we also get $(b|^\#_{\zeta_2}, \zeta_1.\zeta_2) = (a|^\#_{\zeta_2}, \zeta_1.\zeta_2) \in M'_\perp \subseteq A'_{y_j}$.
- If we have $(b^\#, \zeta_1) \in M_\perp$, then we have $(b^\#, \zeta_1) \in A_x$ and thus $(b^\#, \zeta_1) \in \mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(\mathsf{PosEx}(A_x), \mathcal{R}) \subseteq_{(IH)} A'_x$. Since we have $\zeta_1 \perp \pi$ we also have $\zeta_1.\zeta_2 \perp \pi$ and thus $(b|^\#_{\zeta_2}, \zeta_1.\zeta_2) \in M'_\perp \subseteq A'_{y_j}$.

Thus, we have $\mathrm{Sub}^{\mathrm{Poss}}_{\mathrm{set}}(\mathsf{PosEx}(A_{y_j}), \mathcal{R}) \subseteq A'_{y_j}$. ■

Proof of Theorem 5.3.1 (Chain Criterion). Let $\mathcal{R}$ be a PTRS.

"⇒" Assume that $\mathcal{R}$ is not innermost AST. Then by Theorem 4.2.21 and Theorem 4.2.24, there exists an innermost $\mathcal{R}$-computation tree $\mathfrak{T}$ that converges with probability < 1 and starts with $(1 : t)$ such that $t = \ell\sigma$ for some rule $\ell \to r \in \mathcal{R}$, some ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and every proper subterm of $\ell\sigma$ is in normal form w.r.t. $\mathcal{R}$. By Lemma 5.3.4 we get the existence of a $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$-computation tree $\mathfrak{T} = (V, E, L, P)$ that converges with probability < 1 and starts with $(1 : \{\!\{(t^\#, \varepsilon)\}\!\})$ such that $t = \ell^\#\sigma$ for some dependency tuple $(\ell^\#, \ell) \to \ldots \in \mathcal{DT}(\mathcal{R})$, some ground

substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and every proper subterm of $\ell^{\#}\sigma$ is in normal form w.r.t. $\mathcal{R}$. Hence, $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$ is not innermost AST.

"$\Leftarrow$" Assume that $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$ is not innermost AST. Then by Theorem 5.2.23 there exists a $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$-computation tree $\mathfrak{T}$ that converges with probability < 1 and starts with $(1 : \{\!\{(t^{\#}, \varepsilon)\}\!\})$ such that $t^{\#} = \ell^{\#}\sigma$ for some dependency tuple $(\ell^{\#}, \ell) \to \ldots \in \mathcal{DT}(\mathcal{R})$, some ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and every proper subterm of $\ell^{\#}\sigma$ is in normal form w.r.t. $\mathcal{R}$. By Lemma 5.3.6 we get the existence of a $(\mathcal{DT}(\mathcal{R}), \mathcal{R})$-computation tree $\mathfrak{T}'$ that converges with probability < 1 and starts with $(1 : \{\!\{(t^{\#}, \varepsilon)\}\!\})$ such that $t^{\#} = \ell^{\#}\sigma$ for some dependency tuple $(\ell^{\#}, \ell) \to \ldots \in \mathcal{DT}(\mathcal{R})$, some ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, every proper subterm of $\ell^{\#}\sigma$ is in normal form w.r.t. $\mathcal{R}$, and such that we do not use any $\xrightarrow{\mathrm{i}}_{\mathcal{R}}$ steps. By Lemma 5.3.4 we get the existence of an innermost $\mathcal{R}$-computation tree that converges with probability < 1 and starts with $(1 : t)$ such that $t = \ell\sigma$ for some rule $\ell \to r \in \mathcal{R}$, some ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and every proper subterm of t is in normal form w.r.t. $\mathcal{R}$. Hence, $\mathcal{R}$ is not innermost AST. ■

5.4 DP Framework and Processors

Finally, we come to the whole probabilistic dependency pair framework. The core ideas from the non-probabilistic framework remain the same. We want to split a DP problem $(\mathcal{P}, \mathcal{S})$, consisting of a PPTRS $\mathcal{P}$ and a PTRS $\mathcal{S}$, into many simpler subproblems $(\mathcal{P}_1, \mathcal{S}_1), \ldots, (\mathcal{P}_k, \mathcal{S}_k)$ and then solve all of the subproblems individually. In order to split a DP problem, we use processors again. The definition of a probabilistic processor now works with innermost AST instead of innermost termination.

Definition 5.4.1. Let $\mathcal{P}, \mathcal{P}_1, \ldots, \mathcal{P}_n$ be PPTRSs and let $\mathcal{S}, \mathcal{S}_1, \ldots, \mathcal{S}_n$ be PTRSs. A *(innermost AST) DP processor* Proc is of the form

$$\mathrm{Proc}(\mathcal{P}, \mathcal{S}) = \{(\mathcal{P}_1, \mathcal{S}_1), \ldots, (\mathcal{P}_n, \mathcal{S}_n)\}$$

A (innermost AST) DP processor Proc is called *sound* iff

$$\forall\, 1 \leq i \leq n : (\mathcal{P}_i, \mathcal{S}_i) \text{ is innermost AST} \implies (\mathcal{P}, \mathcal{S}) \text{ is innermost AST}$$

A (probabilistic innermost AST) DP processor Proc is called *complete* iff

$$(\mathcal{P}, \mathcal{S}) \text{ is innermost AST} \implies \forall\, 1 \leq i \leq n : (\mathcal{P}_i, \mathcal{S}_i) \text{ is innermost AST}$$

In the following, we will once again talk about the three different processors that we introduced in Chapter 3 and see how we can adapt them to the probabilistic setting. These were the dependency graph processor, the usable rules processor, and the reduction pair processor. In addition to that, we also introduce two new processors, namely the *usable pairs processor* and the *not probabilistic processor*, specifically designed for probabilistic DP problems.

Not Probabilistic Processor

We start with a new processor, called the *not probabilistic processor*. Let $(\mathcal{P}, \mathcal{S})$ be an arbitrary (probabilistic) DP problem. The idea of the not probabilistic processor is that if every dependency tuple in $\mathcal{P}$ has the form $(\ell^{\#}, \ell) \to \{1 : (A, r)\}$ and every probabilistic rewrite rule in $\mathcal{S}$ has the form $\ell' \to \{1 : r'\}$ then this DP problem $(\mathcal{P}, \mathcal{S})$ can also be seen

as a non-probabilistic DP problem since there is no probabilistic choice involved in any rewrite step. In this scenario, innermost AST and innermost sure termination coincide. Therefore, we can transform the problem into a non-probabilistic DP problem $(\mathcal{D}, \mathcal{R})$ and use our existing non-probabilistic DP framework to prove innermost termination automatically. The advantages of using our non-probabilistic framework are that we (currently) have more processors, and more importantly, the processors are specialized for the non-probabilistic setting. We compare the two frameworks in Chapter 6 and see the precise advantages of our non-probabilistic framework in this scenario.

We have already defined the transformation of a PTRS $\mathcal{S}$ to a TRS using the non-probabilistic transformation $\mathrm{np}(\mathcal{S})$. For the not probabilistic processor, we also need to define how to transform a PPTRS $\mathcal{P}$ into a set $\mathrm{np}(\mathcal{P})$ of dependency pairs.

Definition 5.4.2 (Non-Probabilistic Transformation for PPTRS)**.** Let $\mathcal{P}$ be a PPTRS. The *non-probabilistic transformation* $\mathrm{np}(\mathcal{P})$ is the set of dependency pairs defined by

$$\mathrm{np}(\mathcal{P}) \coloneqq \{\ell^\# \to t^\# \mid (\ell^\#, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in \mathcal{P}, 1 \leq j \leq k, (t^\#, \pi) \in C_j\}$$

Example 5.4.3 (Non-Probabilistic Transformation for PPTRS (1))**.** Consider the signature $\Sigma_{div} = \{\mathcal{O}, \mathsf{s}, \mathsf{minus}, \mathsf{div}\}$, a variable set with $\{x, y\} \subseteq \mathcal{V}$, and the following PTRS $\mathcal{R}_{NPdiv}$:

$$\mathsf{minus}(x, \mathcal{O}) \to \{1 : x\} \tag{5.14}$$
$$\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) \to \{1 : \mathsf{minus}(x, y)\} \tag{5.15}$$
$$\mathsf{div}(\mathcal{O}, \mathsf{s}(y)) \to \{1 : \mathcal{O}\} \tag{5.16}$$
$$\mathsf{div}(\mathsf{s}(x), \mathsf{s}(y)) \to \{1 : \mathsf{s}(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y)))\} \tag{5.17}$$

This PTRS is a probabilistic version of our TRS from Example 3.0.1. The set $\mathcal{DT}(\mathcal{R}_{NPdiv})$ contains the following four dependency tuples:

$$\begin{aligned}
\mathcal{DT}((5.14)) = {} & \Big(\mathsf{minus}^\#(x, \mathcal{O}), \mathsf{minus}(x, \mathcal{O})\Big) \\
& \to \{\ \ 1 : \ \Big(\varnothing, x\Big)\} \\
\mathcal{DT}((5.15)) = {} & \Big(\mathsf{minus}^\#(\mathsf{s}(x), \mathsf{s}(y)), \mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y))\Big) \\
& \to \{\ \ 1 : \ \Big(\{\!\!\{\Big(\mathsf{minus}^\#(x, y), \varepsilon\Big)\}\!\!\}, \mathsf{minus}(x, y)\Big)\} \\
\mathcal{DT}((5.16)) = {} & \Big(\mathsf{div}^\#(\mathcal{O}, \mathsf{s}(y)), \mathsf{div}(\mathcal{O}, \mathsf{s}(y))\Big) \\
& \to \{\ \ 1 : \ \Big(\varnothing, \mathcal{O}\Big)\} \\
\mathcal{DT}((5.17)) = {} & \Big(\mathsf{div}^\#(\mathsf{s}(x), \mathsf{s}(y)), \mathsf{div}(\mathsf{s}(x), \mathsf{s}(y))\Big) \\
& \to \{\ \ 1 : \ \Big(\{\!\!\{\Big(\mathsf{div}^\#(\mathsf{minus}(x, y), \mathsf{s}(y)), 1\Big), \Big(\mathsf{minus}^\#(x, y), 1.1\Big)\}\!\!\}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad \mathsf{s}(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y)))\Big)\}
\end{aligned}$$

The non-probabilistic transformation $\mathrm{np}(\mathcal{DT}(\mathcal{R}_{NPdiv}))$ contains the following three dependency pairs:

$$\mathsf{minus}^\#(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{minus}^\#(x, y) \tag{5.18}$$
$$\mathsf{div}^\#(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{minus}^\#(x, y) \tag{5.19}$$
$$\mathsf{div}^\#(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{div}^\#(\mathsf{minus}(x, y), \mathsf{s}(y)) \tag{5.20}$$

These are the same dependency pairs as we have in Example 3.1.4, where we directly use the non-probabilistic TRS $\mathcal{R}_{div}$ and then generate its non-probabilistic dependency pairs.

Theorem 5.4.4 (Not Probabilistic Processor)**.** *Let $(\mathcal{P},\mathcal{S})$ be a probabilistic DP problem such that every dependency tuple in $\mathcal{P}$ has the form $(\ell^\#,\ell)\to\{1:(A,r)\}$ and every probabilistic rewrite rule in $\mathcal{S}$ has the form $\ell'\to\{1:r'\}$. Then*

$$\mathrm{Proc}_{\mathtt{NP}}(\mathcal{P},\mathcal{S})=\{(np(\mathcal{P}),np(\mathcal{S}))\}$$

is sound and complete in the sense that the probabilistic DP problem $(\mathcal{P},\mathcal{S})$ is innermost AST iff the non-probabilistic DP problem $(np(\mathcal{P}),np(\mathcal{S}))$ is innermost terminating.

Proof. Let $(\mathcal{P},\mathcal{S})$ be a probabilistic DP problem such that every dependency tuple in $\mathcal{P}$ has the form $(\ell^\#,\ell)\to\{1:(A,r)\}$ and every probabilistic rewrite rule in $\mathcal{S}$ has the form $\ell'\to\{1:r'\}$. Note that every $(\mathcal{P},\mathcal{S})$-computation tree is a single (not necessarily finite) path. For a chain tree $\mathfrak{T}$ that is only a single path, we have only two different possibilities for $|\mathfrak{T}|_{\mathrm{Leaf}}$. If the path is finite, then $|\mathfrak{T}|_{\mathrm{Leaf}}=1$, since we have a single leaf in this tree with probability 1. Or we have an infinite path, which means that there is no leaf at all and hence $|\mathfrak{T}|_{\mathrm{Leaf}}=0$.

complete: Assume that $(\mathrm{np}(\mathcal{P}),\mathrm{np}(\mathcal{S}))$ is not innermost terminating. Then there exists an infinite innermost $(\mathrm{np}(\mathcal{P}),\mathrm{np}(\mathcal{S}))$-chain

$$t_0\xrightarrow{\mathrm{i}}_{(\mathrm{np}(\mathcal{P}),\mathrm{np}(\mathcal{S}))}t_1\xrightarrow{\mathrm{i}}_{(\mathrm{np}(\mathcal{P}),\mathrm{np}(\mathcal{S}))}t_2\xrightarrow{\mathrm{i}}_{(\mathrm{np}(\mathcal{P}),\mathrm{np}(\mathcal{S}))}\cdots$$

such that for all $i\in\mathbb{N}$ we have $t_i=\ell_i^\#\sigma_i$ for some dependency pair $\ell_i^\#\to r_i^\#\in\mathrm{np}(\mathcal{P})$ that we use in the i-th rewrite step, some ground substitution $\sigma_i\in\mathrm{Sub}\,(\Sigma,\mathcal{V})$ and every proper subterm of $\ell_i^\#\sigma_i$ is in normal form w.r.t. $\mathcal{S}$. Note that a term is in normal form w.r.t. $\mathcal{S}$ iff it is in normal form w.r.t. $\mathrm{np}(\mathcal{S})$ since they have the same left-hand sides. From this infinite innermost $(\mathrm{np}(\mathcal{P}),\mathrm{np}(\mathcal{S}))$-chain, we will now construct an infinite $(\mathcal{P},\mathcal{S})$-computation tree $\mathfrak{T}=(V,E,L,P)$. By our previous analysis, we then know that this infinite innermost $(\mathcal{P},\mathcal{S})$-computation tree must be an infinite path, and thus $|\mathfrak{T}|_{\mathrm{Leaf}}=0$, which means that $(\mathcal{P},\mathcal{S})$ is not innermost AST.

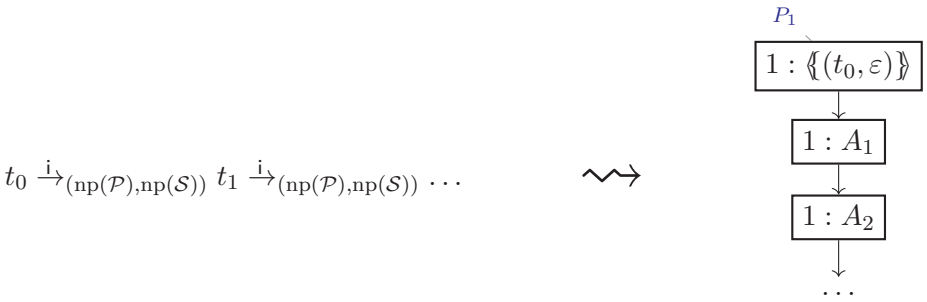


Figure 5.13: Construction for this proof.

We start our construction with $L(\mathfrak{r})=(1:\{\!\{(t_0^\#,\varepsilon)\}\!\})$. In the non-probabilistic rewrite sequence, we have $t_0\xrightarrow{\mathrm{i}}_{(\mathrm{np}(\mathcal{P}),\mathrm{np}(\mathcal{S}))}t_1$, so there exists a natural number $m_0\in\mathbb{N}$ such that

$$t_0=\ell_0^\#\sigma_0\xrightarrow{\mathrm{i}}_{\mathrm{np}(\mathcal{P})}r_0^\#\sigma_0=v_1\xrightarrow{\mathrm{i}}_{\mathrm{np}(\mathcal{S})}v_2\xrightarrow{\mathrm{i}}_{\mathrm{np}(\mathcal{S})}\ldots\xrightarrow{\mathrm{i}}_{\mathrm{np}(\mathcal{S})}v_{m_0}=t_1=\ell_1^\#\sigma_1$$

We now prove that we can mirror this rewrite sequence in $\mathfrak{T}$ such that the pair (v_i,π_i) is contained in the set A_i for all $1\le i\le m$ and some position π_i. Let

$(\ell_0^\#, \ell_0) \to \{1 : (C_0, r_0')\} \in \mathcal{P}$ be the probabilistic dependency tuple that was used to create the dependency pair $\ell_0^\# \to r_0^\#$ in our non probabilistic transformation $\mathrm{np}(\mathcal{P})$. This means that we have $(r_0^\#, \pi_0) \in C_0$ for some position π_0. Since we have $t_0 = \ell_0^\#\sigma_0$ such that every proper subterm of $\ell_0^\#\sigma_0$ is in normal form w.r.t. $\mathcal{S}$, we can also rewrite $\{\!\{(t_0, \varepsilon)\}\!\}$ with the dependency tuple $(\ell_0^\#, \ell_0) \to \{1 : (C_0, r_0')\} \in \mathcal{P}$, the main rewrite pair (t_0, ε) and the ground substitution $\sigma_0 \in \mathrm{Sub}(\Sigma, \mathcal{V})$. We result with $A_1 = C_0\sigma_0$ and thus we have $(v_1, \pi_0) = (r_0^\#\sigma_0, \pi_0) \in C_0\sigma_0 = A_1$ for some position π_0.

In the inductive step, we assume that we have $(v_i, \pi_i) \in A_i$ for some position π_i and some $1 \leq i < m$. In our non-probabilistic rewrite sequence we have $v_i \xrightarrow{\mathrm{i}}_{\mathcal{S}} v_{i+1}$ using a rule $\ell_i' \to r_i' \in \mathrm{np}(\mathcal{S})$, a ground substitution $\delta_i \in \mathrm{Sub}(\Sigma, \mathcal{V})$ and a position $\tau \in \mathbb{N}^+$ such that $v_i|_\tau = \ell_i'\delta_i$, every proper subterm of $\ell_i'\delta_i$ is in normal form w.r.t. $\mathcal{S}$, and $v_{i+1} = v_i[r_i'\delta_i]_\tau$. We can mirror this rewrite step with the rule $\ell_i' \to \{1 : r_i'\} \in \mathcal{S}$ and the set A_i. However, by definition of $\xrightarrow{\mathrm{i}}_{\mathcal{S}}$, we have to apply the rewrite rule at the lowest possible position in the set A_i so that we rewrite every other pair above with the same rule. Let $A_i^{\downarrow\pi_i} \coloneqq \{(a, \alpha) \in A_i \mid \pi_i \leq \alpha < \pi_i.\tau\}$ be the set of all pairs in A_i that have a position below π_i and strictly above $\pi_i.\tau$. Furthermore, let $(t_{\max}, \pi_{\max})$ be the pair in $A_i^{\downarrow\pi_i}$ with maximal position (i.e., there is no $(t', \pi') \in A_i^{\downarrow\pi_i}$ with $\pi_{\max} < \pi'$) and let ρ be the position such that $\pi_{\max}.\rho = \pi_i.\tau$. We can rewrite the set A_i with the rule $\ell_i' \to \{1 : r_i'\} \in \mathcal{S}$, the ground substitution $\delta_i \in \mathrm{Sub}(\Sigma, \mathcal{V})$, the main rewrite pair $(t_{\max}, \pi_{\max})$ and the position ρ such that we have

$$A_i = \{\!\{(t_{\max}, \pi_{\max})\}\!\} \uplus M_{rew} \uplus M_\perp \uplus M_<$$

and

$$A_{i+1} = M^+ \uplus M_\perp \uplus \{\!\{(a[r_i'\sigma]_{\chi_a.\rho}, \chi) \mid (a, \chi) \in M_{rew}\}\!\}$$

as in Definition 5.1.17. We have $(v_i, \pi_i) \in \{\!\{(t_{\max}, \pi_{\max})\}\!\} \uplus M_{rew}$, so that we apply the same rule with the same substitution at the same position. Let χ be the position such that $\pi_i.\chi = \pi_{\max}$. Then we have $\pi_i.\chi.\rho = \pi_i.\tau$ and we get $(v_{i+1}, \pi_i) = (v_i[r_i'\delta_i]_\tau, \pi_i) = (v_i[r_i'\delta_i]_{\chi.\rho}, \pi_i) \in M^+ \uplus \{\!\{(a[r_i'\sigma]_{\chi_a}, \chi) \mid (a, \chi) \in M_{rew}\}\!\} \subseteq A_{i+1}$.

At the end of this induction, we result with A_{m_0}. Next, we can then mirror the step $t_1 \xrightarrow{\mathrm{i}}_{(\mathrm{np}(\mathcal{P}),\mathrm{np}(\mathcal{S}))} t_2$ from our non-probabilistic rewrite sequence with the same construction and so on. This results in an infinite $(\mathcal{P}, \mathcal{S})$-computation tree. To see that the result is a $(\mathcal{P}, \mathcal{S})$-computation tree, note that all of the local properties are satisfied since every edge represents a rewrite step with $\xrightarrow{\mathrm{i}}_{\mathcal{P},\mathcal{S}}$ or $\xrightarrow{\mathrm{i}}_{\mathcal{S}}$. The global property is also satisfied since, in an infinite innermost $(\mathrm{np}(\mathcal{P}), \mathrm{np}(\mathcal{S}))$-chain, we use an infinite amount of dependency pairs so that our resulting chain tree has an infinite amount of nodes in P.

sound: Assume that $(\mathcal{P}, \mathcal{S})$ is not innermost AST. By Theorem 5.2.23 there exists a $(\mathcal{P}, \mathcal{S})$-computation tree $\mathfrak{T} = (V, E, L, P)$ that converges with a probability < 1 and starts with $(1 : \{\!\{(t^\#, \varepsilon)\}\!\})$ such that $t^\# = \ell^\#\sigma$ for some ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, a dependency tuple $(\ell^\#, \ell) \to \{1 : (C_1, r_1)\} \in \mathcal{P}$, and every proper subterm of $\ell^\#\sigma$ is in normal form w.r.t. $\mathcal{S}$. Our previous analysis shows that this tree must be an infinite path. From $\mathfrak{T}$, we will now construct an infinite innermost $(\mathrm{np}(\mathcal{P}), \mathrm{np}(\mathcal{S}))$-chain, which shows that $(\mathrm{np}(\mathcal{P}), \mathrm{np}(\mathcal{S}))$ is not innermost terminating.

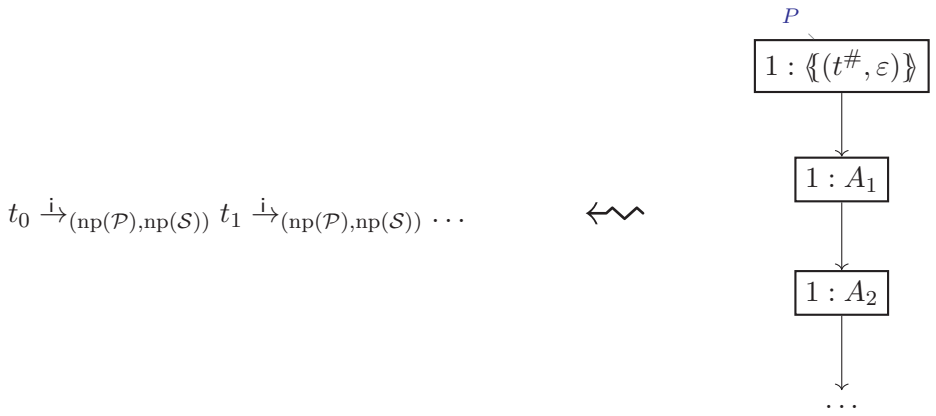


Figure 5.14: Construction for this proof.

We start our infinite chain with the term $t_0 \coloneqq t^\#$. We have $\{\!\{(t^\#, \varepsilon)\}\!\} \xrightarrow{\mathrm{i}}_{\mathcal{P},\mathcal{S}} \{1 : A_1\}$, so that there is a dependency tuple $(\ell^\#, \ell) \to \{p_1 : (C, r)\} \in \mathcal{P}$, the main rewrite pair $(t^\#, \varepsilon)$, and a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ such that $t^\# = \ell^\#\sigma$ and all proper subterms of $\ell^\#\sigma$ are in normal form w.r.t. $\mathcal{S}$. Then $A_1 = M_1^+ = C\sigma$ as in Definition 5.1.12.

Let π_i denote the position of the used main rewrite pair in the i-th rewrite step. Analogous to the minimality criterion we used in the proof of the non-probabilistic chain criterion (Theorem 3.2.1), there must be a pair $(t_1, \alpha_1) \in A_1$ such that

- we will never rewrite with a main rewrite pair position above α_1 (i.e., $\neg\exists i \in \mathbb{N} : \pi_i < \alpha_1$),
- we will rewrite with the main rewrite pair position α_1 in a $\xrightarrow{\mathrm{i}}_{\mathcal{P},\mathcal{S}}$ step after a finite amount of steps,
- and we have infinitely many rewrite steps with $\xrightarrow{\mathrm{i}}_{\mathcal{P},\mathcal{S}}$ and a main rewrite pair position below α_1 (i.e., $\alpha_1 \geq \pi_i$ for infinitely many $i \in \mathbb{N}$).

We have $(t_1, \alpha_1) \in A_1 = C\sigma$, so that there is a pair $(r', \alpha_1) \in C$ with $(t_1, \alpha_1) = (r'\sigma, \alpha_1)$.

We can rewrite the term t_0 with the dependency pair $\ell^\# \to r' \in \mathrm{np}(\mathcal{P})$, using the ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ since $t_0 = t^\# = \ell^\#\sigma$ and all proper subterms of $\ell^\#\sigma$ are in normal form w.r.t. $\mathcal{S}$. Hence, we result with $t_0 \xrightarrow{\mathrm{i}}_{\mathrm{np}(\mathcal{P})} r'\sigma$.

Now for every following rewrite step $i \in \mathbb{N}$, we currently have a term t_i such that $(t_i, \alpha_i) \in A_i$ and

- we will never rewrite with a main rewrite pair position above α_i (i.e., $\neg\exists j \in \mathbb{N} : \pi_j < \alpha_i$),
- we will rewrite with the main rewrite pair position α_i in a $\xrightarrow{\mathrm{i}}_{\mathcal{P},\mathcal{S}}$ step after a finite amount of steps,
- and we have infinitely many rewrite steps with $\xrightarrow{\mathrm{i}}_{\mathcal{P},\mathcal{S}}$ and a main rewrite pair position below α_i (i.e., $\alpha_i \geq \pi_j$ for infinitely many $j \in \mathbb{N}$).

We have the following two cases.

1) If we have $A_i \xrightarrow{\mathrm{i}}_{\mathcal{P},\mathcal{S}} \{1 : A_{i+1}\}$ using a dependency tuple $(\ell^\#, \ell) \to \{1 : (C, r)\} \in \mathcal{P}$, a main rewrite pair $(q, \pi) \in A_x$, and a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ with $q = \ell^\#\sigma$, such that all proper subterms of $\ell^\#\sigma$ are in normal form w.r.t. $\mathcal{S}$. Then

$$A_i = \{\!\{(q, \pi)\}\!\} \uplus M_{rew} \uplus M_\perp \uplus M_<$$

and

$$A_{i+1} = M_1^+ \uplus M_\perp \uplus \{\!\{(a[r\sigma]_{\chi_a}, \chi) \mid (a,\chi) \in M_{rew}\}\!\}$$

as in Definition 5.1.12.

We know that π cannot be above α_i. If we have $\pi \perp \alpha_i$, then $(t_i, \alpha_i) \in M_\perp$ so that $(t_i, \alpha_i) \in A_{i+1}$. In this case, we do nothing in our rewrite sequence and set $t_{i+1} \coloneqq t_i$. The pair $(t_i, \alpha_i) \in A_{i+1}$ still satisfies our minimality conditions.

If we have $(t_i, \alpha_i) \in \{\!\{(q,\pi)\}\!\}$ (i.e., the pair (t_i, α_i) is the main rewrite pair), then we replace $\{\!\{(t_i, \alpha_i)\}\!\}$ by $C\sigma$. Again by a minimality criterion, there must be a pair $(t_{i+1}, \alpha_{i+1}) \in C\sigma \subseteq A_{i+1}$ such that

- we will never rewrite with a main rewrite pair position above α_{i+1} (i.e., $\neg\exists j \in \mathbb{N} : \pi_j < \alpha_i$),
- we will rewrite with the main rewrite pair position α_{i+1} in a $\xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}}$ step after a finite amount of steps,
- and we have infinitely many rewrite steps with $\xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}}$ and a main rewrite pair position below α_{i+1} (i.e., $\alpha_{i+1} \geq \pi_j$ for infinitely many $j \in \mathbb{N}$).

We have $(t_{i+1}, \alpha_{i+1}) \in C\sigma$, so that there is a pair $(r', \alpha_{i+1}) \in C$ with $(t_{i+1}, \alpha_{i+1}) = (r'\sigma, \alpha_{i+1})$. We can rewrite the term t_i with the dependency pair $\ell^\# \to r' \in \mathrm{np}(\mathcal{P})$ and using the ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ since $t_i = q = \ell^\#\sigma$ and all proper subterms of $\ell^\#\sigma$ are in normal form w.r.t. $\mathcal{S}$. Hence, we result with $t_i \xrightarrow{\mathsf{i}}_{\mathrm{np}(\mathcal{P})} r'\sigma = t_{i+1}$.

If we have $(t_i, \alpha_i) \in M_{rew}$, then we rewrite the term t_i with $\mathcal{S}$. There exists a position χ such that $\alpha_i.\chi = \pi$ and we have $(t_i[r'\sigma]_\chi, \alpha_i) \in \{\!\{(a[r\sigma]_{\chi_a}, \chi) \mid (a,\chi) \in M_{rew}\}\!\} \subseteq A_{i+1}$. We set $t_{i+1} \coloneqq t_i[r'\sigma]_\chi$ and get $t_i \xrightarrow{\mathsf{i}}_{\mathcal{S}} t_{i+1}$. Again, this pair (t_{i+1}, π_i) satisfies our minimality criterions.

2) If we have $A_i \xrightarrow{\mathsf{i}}_{\mathcal{S}} \{1 : A_{i+1}\}$, then there is a rule $\ell \to \{1 : r\} \in \mathcal{S}$, a main rewrite pair $(q,\pi) \in A_i$, a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and a position τ with $q|_\tau = \ell\sigma$, such that all proper subterms of $\ell\sigma$ are in normal form w.r.t. $\mathcal{S}$. Then

$$A_i = \{\!\{(q,\pi)\}\!\} \uplus M_{rew} \uplus M_\perp \uplus M_<$$

and

$$A_{i+1} = M_1^+ \uplus M_\perp \uplus \{\!\{(a[r\sigma]_{\chi_a}, \chi) \mid (a,\chi) \in M_{rew}\}\!\}$$

as in Definition 5.1.17.

We know that π cannot be above α_i. If we have $\pi \perp \alpha_i$, then $(t_i, \alpha_i) \in M_\perp$ so that $(t_i, \alpha_i) \in A_{i+1}$. In this case, we do nothing in our rewrite sequence and set $t_{i+1} \coloneqq t_i$. The pair $(t_i, \alpha_i) \in A_{i+1}$ still satisfies our minimality conditions.

Otherwise we have $(t_i, \alpha_i) \in \{\!\{(q,\pi)\}\!\} \uplus M_{rew}$. There exists a position $\chi \in \mathbb{N}^*$ such that $\alpha_i.\chi = \pi$ and we have $(t_i[r\sigma]_\chi, \alpha_i) \in \{\!\{(a[r\sigma]_{\chi_a}, \chi) \mid (a,\chi) \in M_{rew}\}\!\} \subseteq A_{i+1}$. We set $t_{i+1} \coloneqq t_i[r\sigma]_\chi$ and get $t_i \xrightarrow{\mathsf{i}}_{\mathcal{S}} t_{i+1}$. Again, this pair (t_{i+1}, π_i) satisfies our minimality criterions.

This construction creates a sequence $t_0, t_1, \ldots$ of terms such that

$$t_0 \xrightarrow{\mathsf{i}}_{\mathrm{np}(\mathcal{P})} t_1 \ (\xrightarrow{\mathsf{i}}_{\mathrm{np}(\mathcal{P})} \vee \xrightarrow{\mathsf{i}}_{\mathrm{np}(\mathcal{S})} \vee =)\ t_2 \ (\xrightarrow{\mathsf{i}}_{\mathrm{np}(\mathcal{P})} \vee \xrightarrow{\mathsf{i}}_{\mathrm{np}(\mathcal{S})} \vee =)\ \ldots$$

Additionally, by our minimality criterion, we know that we use an infinite amount of $\xrightarrow{\mathsf{i}}_{\mathrm{np}(\mathcal{P})}$ steps so that this is an infinite $(\mathrm{np}(\mathcal{P}), \mathrm{np}(\mathcal{S}))$-chain. Therefore, $(\mathrm{np}(\mathcal{P}), \mathrm{np}(\mathcal{S}))$ is not innermost terminating. ■

Example 5.4.5 (Not Probabilistic Processor)**.** Consider the PTRS $\mathcal{R}_{NPdiv}$ from Example 5.4.3. We have

$$\mathrm{Proc}_{\mathtt{NP}}(\mathcal{DT}(\mathcal{R}_{NPdiv}), \mathcal{R}_{NPdiv}) = \{(\mathcal{DP}(\mathcal{R}_{div}), \mathcal{R}_{div})\}$$

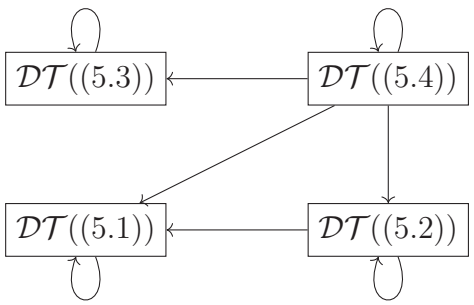
where $\mathcal{R}_{div}$ is the TRS from Example 3.0.1. Hence, we can use our non-probabilistic framework to analyze this DP problem instead of the probabilistic framework. In Chapter 6, we will see the advantages that this brings.

Dependency Graph Processor

Next, we adapt the dependency graph processor to the probabilistic setting. The idea of the dependency graph is to indicate which dependency tuples can rewrite to each other or, in other words, which function calls can occur after each other. The possibility of rewriting to each other is not restricted by the probabilities in any way. In fact, for the dependency graph, it suffices to work with the non-probabilistic structure of the PTRS $\mathcal{S}$.

Definition 5.4.6 (Dependency Graph)**.** Let $(\mathcal{P}, \mathcal{S})$ be a DP problem. The $(\mathcal{P}, \mathcal{S})$-*dependency graph* is defined as the graph with node set $\mathcal{P}$. There is an arc from $(\ell_1^\#, \ell_1) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\}$ to $(\ell_2^\#, \ell_2) \to \ldots$ if there are ground substitutions $\sigma_1, \sigma_2 \in \mathrm{Sub}(\Sigma, \mathcal{V})$ and a pair $(t^\#, \pi) \in C_i$ for some $1 \leq i \leq k$ such that $t^\#\sigma_1 \xrightarrow{\mathrm{i}}{}^*_{\mathrm{np}(\mathcal{S})} \ell_2^\#\sigma_2$ and all proper subterms of $\ell_1^\#\sigma_1$ and $\ell_2^\#\sigma_2$ are in normal form w.r.t. $\mathcal{S}$.

Example 5.4.7 (Dependency Graph)**.** Let $\mathcal{R}_{div}$ be the PTRS from Example 5.0.1 and $\mathcal{DT}(\mathcal{R}_{div})$ be the set of its dependency tuples from Example 5.1.10. The $(\mathcal{DT}(\mathcal{R}_{div}), \mathcal{R}_{div})$-dependency graph has the following form:



In the non-probabilistic setting, every chain corresponds to a path inside the dependency graph. In the probabilistic setting, there is no direct correspondence between paths in the $(\mathcal{P}, \mathcal{S})$-computation tree and paths in the dependency graph anymore due to the fact that we are working with dependency tuples and keeping track of multiple dependency terms simultaneously. Instead, one can think of it as having multiple tokens placed on the dependency graph that represent the current dependency terms in our set. If we rewrite with $\xrightarrow{\mathrm{i}}{\!\!\triangleright}_{\mathcal{P},\mathcal{S}}$, then we can move the token of the used main rewrite pair along the edges of the dependency graph and may additionally split the token into multiple ones if we again have multiple tuple symbols on the right-hand side of our used dependency tuple. So the idea of the dependency graph, that it shows which dependency tuples can rewrite to each other, still remains. Therefore, the idea of only looking at the SCCs of this graph remains as well.

For the proof of the probabilistic dependency graph processor, we need a new definition regarding the ordering of the strongly connected components and the singleton sets of nodes that do not belong to any SCC.

Definition 5.4.8 ($>_{\mathfrak{G}}$)**.** Let $(\mathcal{P}, \mathcal{S})$ be a DP problem and $\mathfrak{G} = (\mathcal{P}, E)$ be its dependency graph. Additionally, let $\mathcal{Q}_1, \ldots, \mathcal{Q}_n$ be the SCCs of $\mathfrak{G}$ and let $W \coloneqq \{\mathcal{Q}_1, \ldots, \mathcal{Q}_n\} \cup \{\{v\} \subseteq \mathcal{P} \mid v \text{ is not in an SCC of } \mathfrak{G}\}$ be the set of all SCCs and all singleton sets of nodes that do not belong to any SCC. Let $X_1, X_2 \in W$. We say that X_2 is a direct successor of X_1 (denoted $X_1 >_{\mathfrak{G}} X_2$) if there exist nodes $v \in X_1$ and $w \in X_2$ such that $(v, w) \in E$.

Example 5.4.9. For the dependency graph from Example 5.4.7, all of the singleton sets are SCCs. Therefore, the relation $>_{\mathfrak{G}}$ for this dependency graph is indicated by its edge relation.

For a finite graph $\mathfrak{G}$, the relation $>_{\mathfrak{G}}$ is well-founded, and we can therefore use it for a well-founded induction proof.

Theorem 5.4.10 (Dependency Graph Processor)**.** *Let*

$$\mathrm{Proc}_{\mathtt{DG}}(\mathcal{P}, \mathcal{S}) = \{(\mathcal{Q}_1, \mathcal{S}), \ldots, (\mathcal{Q}_n, \mathcal{S})\}$$

where $\mathcal{Q}_1, \ldots, \mathcal{Q}_n$ are the SCCs of the $(\mathcal{P}, \mathcal{S})$-dependency graph. Then $\mathrm{Proc}_{\mathtt{DG}}$ *is sound and complete.*

Proof.

complete: Every $(\mathcal{Q}_i, \mathcal{S})$-computation tree is also a $(\mathcal{P}, \mathcal{S})$-computation tree. Hence, if one of the $(\mathcal{Q}_i, \mathcal{S})$ is not innermost AST, then $(\mathcal{P}, \mathcal{S})$ is also not innermost AST.

sound: Suppose that every $(\mathcal{Q}_i, \mathcal{S})$-computation tree converges with probability 1 for every $1 \leq i \leq n$. We prove that then also every $(\mathcal{P}, \mathcal{S})$-computation tree converges with probability 1. Let $W \coloneqq \{\mathcal{Q}_1, \ldots, \mathcal{Q}_n\} \cup \{\{v\} \subseteq \mathcal{P} \mid v \text{ is not in an SCC of } \mathfrak{G}\}$ be the set of all SCCs and all singleton sets of nodes that do not belong to any SCC. The core steps of this proof are the following:

1. We show that every DP problem $(X, \mathcal{S})$ with $X \in W$ is innermost AST.
2. We show that composing SCCs maintains the innermost AST property.
3. We show that for every $X \in W$, the DP problem $(\bigcup_{X >_{\mathfrak{G}}^* Y} Y, \mathcal{S})$ is innermost AST by well-founded induction over $>_{\mathfrak{G}}$.
4. We conclude that $(\mathcal{P}, \mathcal{S})$ must be innermost AST.

1. Every DP problem $(X, \mathcal{S})$ with $X \in W$ is innermost AST.

We start by proving the following:

$$\text{Every DP problem } (X, \mathcal{S}) \text{ with } X \in W \text{ is innermost AST.} \tag{5.21}$$

To prove (5.21), note that if X is an SCC, then it follows from our assumption that $(X, \mathcal{S})$ is innermost AST. If X is a singleton set of a node that does not belong to any SCC, then assume for a contradiction that $(X, \mathcal{S})$ is not innermost AST. By Theorem 5.2.23 there exists a $(X, \mathcal{S})$-computation tree $\mathfrak{T} = (V, E, L, P)$ that converges with a probability < 1 and starts with $(1 : \{\!\{(t^{\#}, \varepsilon)\}\!\})$ such that $t^{\#} = \ell^{\#}\sigma$ for some ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, the only dependency tuple $(\ell^{\#}, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in X$, and every proper subterm of $\ell^{\#}\sigma$ is in normal form w.r.t. $\mathcal{S}$. The first rewrite step at the root of $\mathfrak{T}$ must therefore be

$$\{\!\{(t^{\#}, \varepsilon)\}\!\} \xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}} \{p_1 : C_1\sigma, \ldots, p_k : C_k\sigma\}$$

since we can only use $(t^\#, \varepsilon)$ as the main rewrite pair and use the only dependency tuple.

Assume for a contradiction that there exists a node $x \in P$ in $\mathfrak{T}$ that is not the root (i.e., $x \neq \varepsilon$). W.l.o.G. let x be reachable from the root without seeing any other node from P. This means that A_x contains a pair (d, τ) such that $d = \ell^\# \sigma'$ for some ground substitution $\sigma' \in \mathrm{Sub}(\Sigma, \mathcal{V})$, the only dependency tuple $(\ell^\#, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in X$, and every proper subterm of $\ell^\# \sigma'$ is in normal form w.r.t. $\mathcal{S}$. We now show that we must have a self loop in the dependency graph for the dependency tuple $(\ell^\#, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in X$, which is a contradiction to our assumption that X is a singleton of a dependency tuple that is not on any SCC.

Let $\mathfrak{r} = z_1 \ldots z_m = x$ be the path from the root to x. Remember, for the first rewrite step at the root we have $\{\!\{(t^\#, \varepsilon)\}\!\} \xrightarrow{\mathrm{i}}_{\mathcal{P},\mathcal{S}} \{p_1 : C_1\sigma, \ldots, p_k : C_k\sigma\}$, so that $A_{z_1} = C_j\sigma$ for some $1 \leq j \leq k$. After that, we only use $\xrightarrow{\mathrm{i}}_{\mathcal{S}}$ steps in the path since all of the nodes $z_2, \ldots, z_{m-1}$ are contained in S. Note that rewriting with $\xrightarrow{\mathrm{i}}_{\mathcal{S}}$ cannot add new pairs to a set but can only rewrite the terms of existing pairs or remove pairs. Therefore, we must have a pair $(d', \tau) \in C_j\sigma$ with the same position as (d, τ). This means that there is some pair $(r', \tau) \in C_j$ with $(r'\sigma, \tau) = (d', \tau)$. We will now recursively define a rewrite sequence $r'\sigma = d' \xrightarrow{\mathrm{i}}{}^*_{\mathrm{np}(\mathcal{S})} d = \ell^\# \sigma'$ which means that there must be a self-loop for the only dependency tuple in X.

We construct this rewrite sequence inductively over the length of the path $z_1 \ldots z_m$. The idea of this construction is that the path itself is our desired $\xrightarrow{\mathrm{i}}{}^*_{\mathrm{np}(\mathcal{S})}$-sequence, but we may have edges where the term does not change, e.g., if we rewrite with a main rewrite pair that has a position that is orthogonal to τ. Those rewrite steps can simply be ignored since the term does not change. In this construction, we ensure that after the i-th construction step, we currently have a rewrite sequence $d' \xrightarrow{\mathrm{i}}{}^*_{\mathrm{np}(\mathcal{S})} d_i$ and the last term in this sequence is contained in A_{z_i} (i.e., we have $(d_i, \tau) \in A_{z_i}$). We start with $d_1 \coloneqq d'$. Here, we have $(d_1, \tau) = (d', \tau) \in A_{z_1}$.

For the inductive step, assume we have already defined the rewrite sequence $d' \xrightarrow{\mathrm{i}}{}^*_{\mathrm{np}(\mathcal{S})} d_i$. The node z_i, together with the labeling and successors, represent a step with $\xrightarrow{\mathrm{i}}_{\mathcal{S}}$. This means that we have $A_{z_i} \xrightarrow{\mathrm{i}}_{\mathcal{S}} \{p'_1 : B_1, \ldots, p'_{k'} : B_{k'}\}$ with $A_{z_{i+1}} = B_h$ for some $1 \leq h \leq k'$, using the main rewrite pair $(q, \alpha) \in A_{z_i}$, a ground substitution $\sigma^{(i)} \in \mathrm{Sub}(\Sigma, \mathcal{V})$, the rule $\ell^{(i)} \to \{p_1^{(i)} : r_1^{(i)}, \ldots, p_{k'}^{(i)} : r_{k'}^{(i)}\} \in \mathcal{S}$, some position $\rho \in \mathbb{N}^+$ such that $a|_\rho = \ell^{(i)}\sigma^{(i)}$, and every proper subterm of $\ell^{(i)}\sigma^{(i)}$ is in normal form w.r.t. $\mathcal{S}$. Furthermore, we have

$$A_{z_i} = \{\!\{(q, \alpha)\}\!\} \uplus M_{rew} \uplus M_\perp \uplus M_<$$

and

$$A_{z_{i+1}} = M_h^+ \uplus \{\!\{(a[r_j\sigma]_{\chi a.\rho}, \chi) \mid (a, \chi) \in M_{rew}\}\!\} \uplus M_\perp$$

as in Definition 5.1.17. The case $(d_i, \tau) \in M_<$ is not possible, as we cannot remove the pair in the path to x. Hence, we have the following two possibilities:

- If $\alpha \perp \tau$, then $(d_i, \tau) \in M_\perp$ and hence $(d_i, \tau) \in A_{z_{i+1}}$. Here, we set $d_{i+1} = d_i$ and get $d' \xrightarrow{\mathrm{i}}{}^*_{\mathrm{np}(\mathcal{S})} d_i = d_{i+1}$ and $(d_{i+1}, \tau) \in A_{z_{i+1}}$.
- If $\tau \leq \alpha$, then there is a position $\zeta \in \mathbb{N}^*$ with $\tau.\zeta = \alpha$ and $(d_i, \tau) \in \{\!\{(q, \alpha)\}\!\} \cup M_{rew}$. In this case, we apply the same rule that we applied to the main rewrite pair to d_i at position $\zeta.\rho$. This means that we have $(d_i[r_h^{(i)}\sigma^{(i)}]_{\zeta.\rho}, \tau) \in A_{y_{i+1}}$. We can apply the non-probabilistic rewrite rule

$\ell^{(i)} \to r_h^{(i)} \in \mathrm{np}(\mathcal{S})$ to d_i at position $\zeta.\rho$ using the substitution $\sigma^{(i)}$ to get $d_i[r_h^{(i)}\sigma^{(i)}]_{\zeta.\rho}$. Hence, we set $d_{i+1} \coloneqq d_i[r_h^{(i)}\sigma^{(i)}]_{\zeta.\rho}$ and get $d' \xrightarrow{\mathrm{i}}{}^*_{\mathrm{np}(\mathcal{S})} d_i \xrightarrow{\mathrm{i}}_{\mathrm{np}(\mathcal{S})} d_{i+1}$ and $(d_{i+1}, \tau) \in A_{z_{i+1}}$.

Now, we have proven that the $(\mathcal{P}, \mathcal{S})$-computation tree $\mathfrak{T}$ does not contain a node $x \in P$ that is not the root. By definition of a $(\mathcal{P}, \mathcal{S})$-computation, every infinite path must contain an infinite number of nodes in P. Thus, every path in $\mathfrak{T}$ must be finite, which means that $\mathfrak{T}$ is finite itself. Since every finite computation tree converges with probability 1, this is a contradiction to our assumption that the tree converges with probability < 1.

2. Composing SCCs maintains the innermost AST property.

Next, we show that composing SCCs maintains the innermost AST property. More precisely, we show the following:

$$\begin{aligned}&\text{Let } \overline{X} \subseteq W \text{ and } \overline{Y} \subseteq W \text{ such that there are no } X_1, X_2 \in \overline{X} \text{ and } Y \in \overline{Y}\\ &\text{with } X_1 >^*_{\mathfrak{G}} Y >^*_{\mathfrak{G}} X_2 \text{ and } Y \notin \overline{X} \text{ and such that there are no } Y_1, Y_2 \in \overline{Y}\\ &\text{and } X \in \overline{X} \text{ with } Y_1 >^*_{\mathfrak{G}} X >^*_{\mathfrak{G}} Y_2 \text{ and } X \notin \overline{Y}. \text{ If both } (\textstyle\bigcup_{X\in\overline{X}} X, \mathcal{S}) \text{ and}\\ &(\textstyle\bigcup_{Y\in\overline{Y}} Y, \mathcal{S}) \text{ are innermost AST, then } (\textstyle\bigcup_{X\in\overline{X}} X \cup \bigcup_{Y\in\overline{Y}} Y, \mathcal{S}) \text{ is innermost}\\ &\text{AST as well.}\end{aligned} \tag{5.22}$$

To show (5.22), we assume that both $(\bigcup_{X\in\overline{X}} X, \mathcal{S})$ and $(\bigcup_{Y\in\overline{Y}} Y, \mathcal{S})$ are innermost AST. Let $Z \coloneqq \bigcup_{X\in\overline{X}} X \cup \bigcup_{Y\in\overline{Y}} Y$. The property in (5.22) for $\overline{X}$ and $\overline{Y}$ says that a path between two nodes from $\bigcup_{X\in\overline{X}} X$ that only sees nodes from Z must also be a path that only sees nodes from $\bigcup_{X\in\overline{X}} X$, so that $\bigcup_{Y\in\overline{Y}} Y$ can not be used to "create" new paths between two nodes from $\overline{X}$, and vice versa. Assume for a contradiction that $(Z, \mathcal{S})$ is not innermost AST. By Theorem 5.2.23 there exists a $(Z, \mathcal{S})$-computation tree $\mathfrak{T} = (V, E, L, P)$ that converges with a probability < 1 and starts with $\{\!\{(t^\#, \varepsilon)\}\!\}$ such that $t^\# = \ell^\#\sigma$ for some ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, a dependency tuple $(\ell^\#, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in Z$, and every proper subterm of $\ell^\#\sigma$ is in normal form w.r.t. $\mathcal{S}$.

W.l.o.G., we may assume that the dependency tuple that is used for the rewrite step at the root is inside of $\bigcup_{X\in\overline{X}} X$. Otherwise, simply swap $\bigcup_{X\in\overline{X}} X$ with $\bigcup_{Y\in\overline{Y}} Y$.

We can partition the set P of our $(Z, \mathcal{S})$-computation tree $\mathfrak{T}$ into the sets

- $P_1 \coloneqq \{x \in P \mid x \text{ together with the labeling and its successors represents a step with a dependency tuple from } \bigcup_{X\in\overline{X}} X\}$
- $P_2 \coloneqq \{x \in P \mid x \notin P_1\}$

Note that in the case of $x \in P_2$, we know that x together with its successors and the labeling represents a step with a dependency tuple from $\bigcup_{Y\in\overline{Y}} Y$ that is not in $\bigcup_{X\in\overline{X}} X$. We know that every $(\bigcup_{Y\in\overline{Y}} Y, \mathcal{S})$-computation tree converges with probability 1, since $(\bigcup_{Y\in\overline{Y}} Y, \mathcal{S})$ is innermost AST. Thus, also every $(\bigcup_{Y\in\overline{Y}} Y \setminus \bigcup_{X\in\overline{X}} X, \mathcal{S})$-computation tree converges with probability 1. Furthermore, we have $|\mathfrak{T}|_{\mathrm{Leaf}} < 1$ by our assumption. We can apply the P-Partition Lemma (Lemma 5.2.20) and find a grounded induced sub $(Z, \mathcal{S})$-computation tree $\mathfrak{T}' = (V', E', L', P \cap V')$ with $|\mathfrak{T}'|_{\mathrm{Leaf}} < 1$ such that every infinite path has an infinite number of P_1 edges. Since $\mathfrak{T}'$ is a grounded induced sub computation tree of $\mathfrak{T}$ it must also start with $(1 : \{\!\{(t^\#, \varepsilon)\}\!\})$.

We now construct a $(\bigcup_{X\in\overline{X}} X, \mathcal{S})$-computation tree $\mathfrak{T}'' = (V', E', L'', P_1 \cap V')$ with the same underlying tree structure and an adjusted labeling such that $p_x^{\mathfrak{T}'} = p_x^{\mathfrak{T}''}$ for all $x \in V'$. Since the tree structure and the probabilities are the same, we then

get $|\mathfrak{T}'|_{\text{Leaf}} = |\mathfrak{T}''|_{\text{Leaf}}$. To be precise, the set of leaves in $\mathfrak{T}'$ is equal to the set of leaves in $\mathfrak{T}''$, and every leaf has the same probability. Since $|\mathfrak{T}'|_{\text{Leaf}} < 1$ we thus have $|\mathfrak{T}''|_{\text{Leaf}} < 1$, which is a contradiction to our assumption that $(\bigcup_{X \in \overline{X}} X, \mathcal{S})$ is innermost AST.

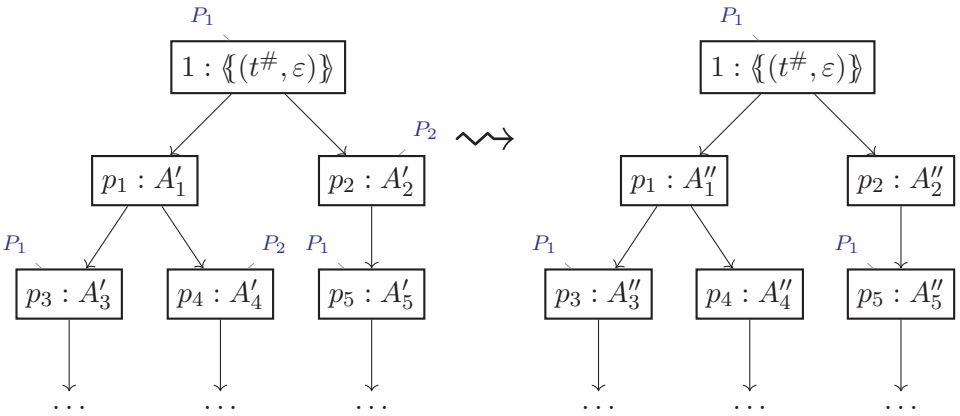
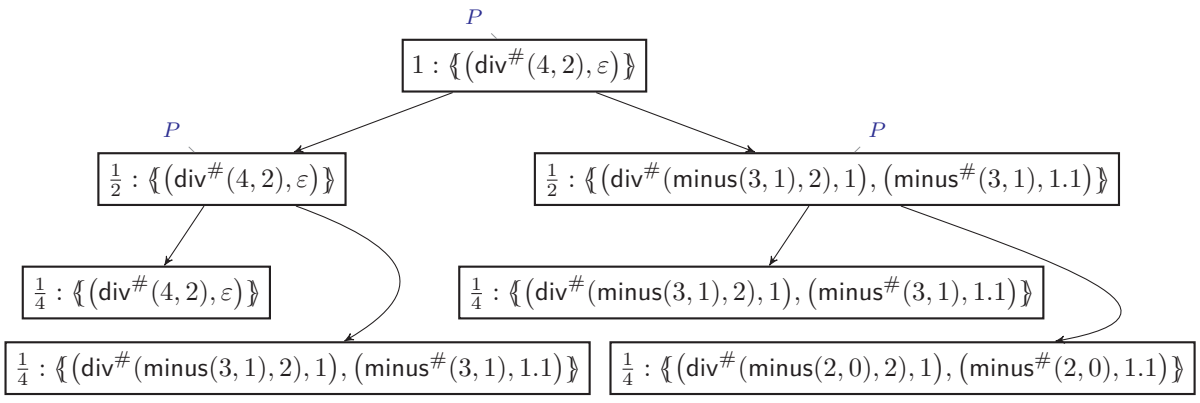


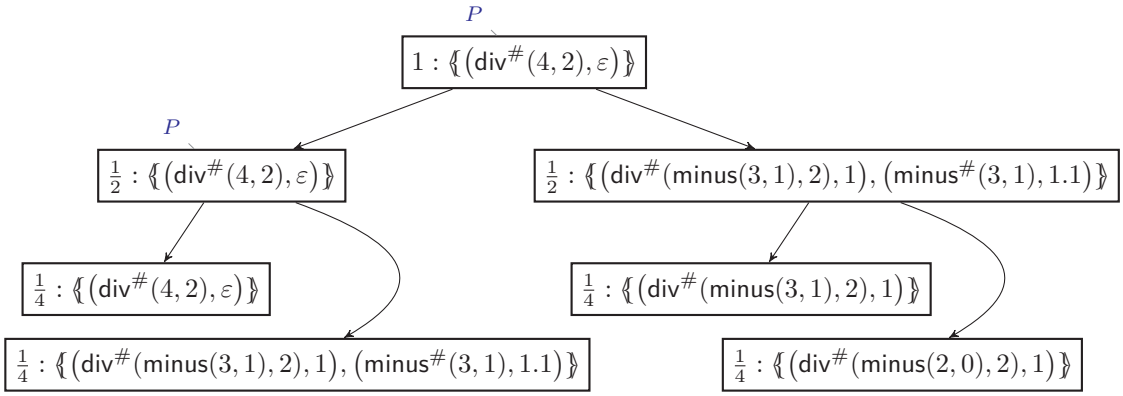
Figure 5.15: Construction for this proof. Every node $x \in P_2$ is removed from P.

The core idea of this construction is that pairs introduced by rewrite steps at a node $x \in P_2$ are not of importance for our computation. The reason for that is that the rewrite step is done using a dependency tuple from $\bigcup_{Y \in \overline{Y}} Y$ that is not contained in $\bigcup_{X \in \overline{X}} X$. By our assumption (5.22) we know that such a dependency tuple has no path in the dependency graph to a dependency tuple in $\bigcup_{X \in \overline{X}} X$. Hence, by definition of the dependency graph, we are never able to use these pairs as a main rewrite pair for a rewrite step with a dependency tuple from $\bigcup_{X \in \overline{X}} X$. We can therefore use the PTRS $\mathcal{S}$ to mirror the rewrite step of the second component of the used dependency tuple from $\bigcup_{Y \in \overline{Y}} Y$, and we completely ignore the first component. If the second component is not applicable (e.g., if the projection to the second component of the used dependency tuple is not contained in $\mathcal{S}$), then we use split-nodes to do nothing but mirror the tree structure. This means that we are removing every node $x \in P_2$ from P, but since every infinite path contains an infinite amount of P_1 nodes, we can be sure that the resulting tree still has an infinite amount of P nodes on every infinite path and thus the resulting tree is a $(\bigcup_{X \in \overline{X}} X, \mathcal{S})$-computation tree.

For example, if we use our DP problem $(\mathcal{DT}(\mathcal{R}_{div}), \mathcal{R}_{div})$ with the dependency graph shown in Example 5.4.7 and look at the following $(\mathcal{DT}(\mathcal{R}_{div}), \mathcal{R}_{div})$-computation tree $\mathfrak{T}$:



We can partition the SCCs of the dependency graph into the sets $\overline{X} = \{(5.3), (5.4)\}$ (the dependency tuples for $\mathsf{div}^\#$) and $\overline{X} = \{(5.1), (5.2)\}$ (the dependency tuples for $\mathsf{minus}^\#$). Then $\overline{X}$ and $\overline{Y}$ satisfies our condition (5.22). Our construction would now instead of the dependency tuples from $\overline{Y}$ use the PTRS rules (5.1) and (5.2) to evaluate the inner minus subterm instead. This results in the following $(\mathcal{DT}(\mathcal{R}_{div}), \mathcal{R}_{div})$-computation tree, that does not use any dependency tuples from $\overline{Y}$:



For the right node, we use the PTRS rule (5.2) instead of the dependency tuple $\mathcal{DT}((5.2))$. In this case, we remove the pair $\left(\mathsf{minus}^\#(3,1), 1\right)$. The corresponding term has a $\mathsf{minus}^\#$ at the root, which means that it can not be used as a main rewrite pair for a step with a dependency tuple from $\overline{X}$. This is indicated by the dependency graph.

We now construct the new labeling L'' for the $(\bigcup_{X \in \overline{X}} X, \mathcal{S})$-computation tree $\mathfrak{T}''$ recursively. Let $Q \subseteq V$ be the set of nodes, where we have already defined the labeling $L''(x)$. During our construction, we ensure that the following property holds:

$$\text{For every node } x \in Q \text{ we have } A'_x \setminus \mathrm{Junk}(A'_x, \overline{X}) \subseteq A''_x. \tag{5.23}$$

Here, $\mathrm{Junk}(A'_x, \overline{X})$ denotes the set of all pairs in A'_x that can never be used as a main rewrite pair for a dependency tuple from $\overline{X}$, indicated by the dependency graph. To be precise, we define $(q, \pi) \in \mathrm{Junk}(A'_x, \overline{X}) :\Leftrightarrow (q, \pi) \in A'_x$ and there is no $Z \in W$ with $Z >^*_{\mathfrak{G}} X$ for some $X \in \overline{X}$ such that there is a dependency tuple $(\ell^\#, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in Z$, and a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ with $q \xrightarrow{\mathsf{i}}{}^*_{\mathrm{np}(\mathcal{S})} \ell^\#\sigma$, and every proper subterm of $\ell^\#\sigma$ is in normal form w.r.t. $\mathcal{S}$.

We start by setting $A''_{\mathfrak{t}} \coloneqq A'_{\mathfrak{t}} = \{\!\{(t^\#, \varepsilon)\}\!\}$. Here, our induction property (5.23) is clearly satisfied as we have

$$A'_{\mathfrak{t}} \setminus \mathrm{Junk}(A'_{\mathfrak{t}}, \overline{X}) \subseteq A'_{\mathfrak{t}} = A''_{\mathfrak{t}}$$

As long as there is still an inner node $x \in Q$ such that their successors are not contained in Q, we do the following. Let $xE = \{y_1, \ldots, y_k\}$ be the set of its successors. We need to define the corresponding terms for the nodes $y_1, \ldots, y_k$.

Since x is not a leaf and $\mathfrak{T}'$ is a $(Z, \mathcal{S})$-computation tree, we have

$$A'_x \xrightarrow{\mathsf{i}}{}_{Z,\mathcal{S}} \{\tfrac{p_{y_1}}{p_x} : A'_{y_1}, \ldots, \tfrac{p_{y_k}}{p_x} : A'_{y_k}\} \text{ or } A_x \xrightarrow{\mathsf{i}}{}_{\mathcal{S}} \{\tfrac{p_{y_1}}{p_x} : A'_{y_1}, \ldots, \tfrac{p_{y_k}}{p_x} : A'_{y_k}\}$$

In Figure 5.16, one can see the different cases that we have to handle in this construction.

	Case	Subcase	Construction
1)	$\xrightarrow{\mathsf{i}}_{Z,\mathcal{S}}$ and $x \in P_1$		Mirror the rewrite step with the same dependency tuple, same main rewrite pair and same substitution
2)	$\xrightarrow{\mathsf{i}}_{Z,\mathcal{S}}$ and $x \in P_2$	Coupled rewrite rule not in $\mathcal{S}$	Use a split-node to mirror the tree structure, ignore the rewrite step
3.1)	$\xrightarrow{\mathsf{i}}_{Z,\mathcal{S}}$ and $x \in P_2$	Coupled rewrite rule in $\mathcal{S}$ and $\neg\exists(t',\pi') \in A''_x : \pi' \leq \pi$	Use a split-node to mirror the tree structure, ignore the rewrite step
3.2)	$\xrightarrow{\mathsf{i}}_{Z,\mathcal{S}}$ and $x \in P_2$	Coupled rewrite rule in $\mathcal{S}$ and $\exists(t',\pi') \in A''_x : \pi' \leq \pi$	Mirror rewrite step with the rule in $\mathcal{S}$ and same substitution at the lowest possible position
4.1)	$\xrightarrow{\mathsf{i}}_{\mathcal{S}}$	$\neg\exists(t',\pi') \in A''_x : \pi' \leq \pi$	Use a split-node to mirror the tree structure, ignore the rewrite step
4.2)	$\xrightarrow{\mathsf{i}}_{\mathcal{S}}$	$\exists(t',\pi') \in A''_x : \pi' \leq \pi$	Mirror rewrite step with the same rewrite rule and same substitution at the lowest possible position

Figure 5.16: Case Distinction for the inductive step

1) If we have $A'_x \xrightarrow{\mathsf{i}}_{Z,\mathcal{S}} \{\frac{p_{y_1}}{p_x} : A'_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : A'_{y_k}\}$ using a dependency tuple $(\ell^\#, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in \bigcup_{X \in \overline{X}} X$ (i.e., $x \in P_1$), a main rewrite pair $(q, \pi) \in A'_x$, and a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ such that $q = \ell^\#\sigma$ and all proper subterms of $\ell^\#\sigma$ are in normal form w.r.t. $\mathcal{S}$. Then

$$A'_x = \{\!\{(q,\pi)\}\!\} \uplus M'_{rew} \uplus M'_\perp \uplus M'_<$$

and

$$A'_{y_j} = M_j^{+'} \uplus M'_\perp \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a,\chi) \in M'_{rew}\}\!\}$$

for all $1 \leq j \leq k$ as in Definition 5.1.12.

Here, we have $(\ell^\#, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in \bigcup_{X \in \overline{X}} X$ (i.e., we use a dependency pair that is inside of $\bigcup_{X \in \overline{X}} X$), and thus the pair (q, π) cannot be inside of $\mathrm{Junk}(A'_x, \overline{X})$. Hence, we must have $(q,\pi) \in A'_x \setminus \mathrm{Junk}(A'_x, \overline{X}) \subseteq_{(IH)} A''_x$. Thus, we can rewrite the set A''_x using the dependency tuple $(\ell^\#, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in \bigcup_{X \in \overline{X}} X$, the main rewrite pair $(q, \pi) \in A''_x$, and the substitution σ as we have $q = \ell^\#\sigma$ and all proper subterms of $\ell^\#\sigma$ are in normal form w.r.t. $\mathcal{S}$.

This means that we have $A''_x \xrightarrow{\mathsf{i}}_{Z,\mathcal{S}} \{\frac{p_{y_1}}{p_x} : B^x_1, \ldots, \frac{p_{y_k}}{p_x} : B^x_k\}$ with

$$A''_x = \{\!\{(q,\pi)\}\!\} \uplus M''_{rew} \uplus M''_\perp \uplus M''_<$$

and

$$B^x_j = M_j^{+''} \uplus M''_\perp \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a,\chi) \in M''_{rew}\}\!\}$$

for all $1 \leq j \leq k$ as in Definition 5.1.12. We define the set in the labeling of the successors $y_1, \ldots, y_k$ in $\mathfrak{T}''$ to be $A''_{y_j} = B^x_j$ for all $1 \leq j \leq k$. It remains to show

that our induction hypothesis is still satisfied for this new labeling, i.e., we have $A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X}) \subseteq A''_{y_j}$ for all $1 \leq j \leq k$.

Let $1 \leq j \leq k$ and $(b, \rho) \in A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X})$. We have the following possibilities:

- If $(b, \rho) \in M_j^{+'}$, then also $(b, \rho) \in M_j^{+''}$ (since $M_j^{+'} = M_j^{+''}$ as we are using the same dependency pair with the same substitution and the same main rewrite pair) and thus $(b, \rho) \in A''_{y_j}$.
- If $(b, \rho) \in \{\!\{(a[r_j\sigma]_{\chi_a}, \alpha) \mid (a, \alpha) \in M'_{rew}\}\!\}$, then we have a pair $(a, \rho) \in A'_x$ and a position $\chi \in \mathbb{N}^+$ with $\rho.\chi = \pi$. By definition, the same rule that we applied to q is applied to a at position χ, so that we result with $b = a[r_j\sigma]_\chi$.
  If we have $(a, \rho) \in \mathrm{Junk}(A'_x, \overline{X})$, then we also have $(b, \rho) = (a[r_j\sigma]_\chi, \rho) \in \mathrm{Junk}(A'_{y_j}, \overline{X})$, since $a \xrightarrow{\mathrm{i}}_{\mathrm{np}(\mathcal{S})} a[r_j\sigma]_\chi = b$, and this is a contradiction to our assumption that $(b, \rho) \in A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X})$. Thus, we have $(a, \rho) \in A'_x \setminus \mathrm{Junk}(A'_x, \overline{X}) \subseteq_{(IH)} A''_x$, and $(a, \rho) \in M''_{rew}$, since $\rho < \pi$. Now the same rewrite step takes place at the same position as above, so that we get $(b, \rho) = (a[r_j\sigma]_\chi, \rho) \in A''_{y_j}$.
- If we have $(b, \rho) \in M'_\perp$, then $(b, \rho) \in A'_x$. If we have $(b, \rho) \in \mathrm{Junk}(A'_x, \overline{X})$, then we would also have $(b, \rho) \in \mathrm{Junk}(A'_{y_j}, \overline{X})$, and this is a contradiction to our assumption that $(b, \rho) \in A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X})$. Thus, we have $(b, \rho) \in A'_x \setminus \mathrm{Junk}(A'_x, \overline{X}) \subseteq_{(IH)} A''_x$. Furthermore, $\rho \perp \pi$ and thus also $(b, \rho) \in M''_\perp$, and hence $(b, \rho) \in A''_{y_j}$.

2) If we have $A'_x \xrightarrow{\mathrm{i}}_{Z,\mathcal{S}} \{\frac{p_{y_1}}{p_x} : A'_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : A'_{y_k}\}$ using a dependency tuple $(\ell^\#, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in \bigcup_{Y \in \overline{Y}} Y \setminus \bigcup_{X \in \overline{X}} X$ (i.e., $x \in P_2$), a main rewrite pair $(q, \pi) \in A'_x$, and a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ such that $q = \ell^\#\sigma$, all proper subterms of $\ell^\#\sigma$ are in normal form w.r.t. $\mathcal{S}$, and $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \notin S$. Then

$$A'_x = \{\!\{(q, \pi)\}\!\} \uplus M'_{rew} \uplus M'_\perp \uplus M'_<$$

and

$$A'_{y_j} = M_j^{+'} \uplus M'_\perp$$

for all $1 \leq j \leq k$ as in Definition 5.1.12. In this case, we can simply use a split-node to do nothing, as this rewrite step is completely irrelevant to our construction. This means that we set $A''_{y_j} \coloneqq A''_x$ for all $1 \leq j \leq k$. It remains to show that our induction hypothesis is still satisfied for this new labeling, i.e., we have $A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X}) \subseteq A''_{y_j}$ for all $1 \leq j \leq k$.

Let $1 \leq j \leq k$ and $(b, \rho) \in A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X})$. We have the following possibilities:

- If $(b, \rho) \in M_j^{+'}$, then there is $(r', \chi) \in C_j \subseteq dp(r_j)$ such that $(b, \rho) = (r'\sigma, \pi.\chi)$. Since our used dependency pair is inside of Y for some $Y \in \overline{Y}$ but not contained in $\bigcup_{X \in \overline{X}} X$, we know by (5.22) that $Y \not\geq^*_{\mathfrak{G}} X$ for all $X \in \overline{X}$. Intuitively, we started with a dependency tuple from $\overline{X}$, reaching Y, so we cannot reach any $X \in \overline{X}$ anymore. By definition of the dependency graph, this means that there is no $Z \in W$ with $Z >^*_{\mathfrak{G}} X$ for some $X \in \overline{X}$ such that there is a dependency tuple $(\ell'^\#, \ell') \to \ldots \in Z$, and a ground substitution $\sigma' \in \mathrm{Sub}(\Sigma, \mathcal{V})$ with $b \xrightarrow{\mathrm{i}}^*_{\mathrm{np}(\mathcal{S})} \ell'^\#\sigma'$, and every proper subterm of $\ell'^\#\sigma'$ is in normal form w.r.t. $\mathcal{S}$. Hence, we have $(b, \rho) \in \mathrm{Junk}(A'_{y_j}, \overline{X})$ and this is a contradiction to our assumption that $(b, \rho) \in A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X})$. Thus, this case is not possible.

– If we have $(b,\rho) \in M'_\perp$, then $(b,\rho) \in A_x$. If we have $(b,\rho) \in \mathrm{Junk}(A'_x, \overline{X})$, then we also have $(b,\rho) \in \mathrm{Junk}(A'_{y_j}, \overline{X})$, and this is a contradiction to our assumption that $(b,\rho) \in A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X})$. Thus, we have $(b,\rho) \in A'_x \setminus \mathrm{Junk}(A'_x, \overline{X}) \subseteq_{(IH)} A''_x$, then we get $(b,\rho) \in A''_x = A''_{y_j}$ by our construction.

3) If we have $A'_x \xrightarrow{\mathrm{i}}_{Z,\mathcal{S}} \{\frac{p_{y_1}}{p_x} : A'_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : A'_{y_k}\}$ using a dependency tuple $(\ell^\#, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in \bigcup_{Y \in \overline{Y}} Y \setminus \bigcup_{X \in \overline{X}} X$ (i.e., $x \in P_2$), a main rewrite pair $(q,\pi) \in A'_x$, and a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ such that $q = \ell^\#\sigma$, all proper subterms of $\ell^\#\sigma$ are in normal form w.r.t. $\mathcal{S}$, and $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in S$. Then

$$A'_x = \{\!\{(q,\pi)\}\!\} \uplus M'_{rew} \uplus M'_\perp \uplus M'_<$$

and

$$A'_{y_j} = M_j^{+'} \uplus M'_\perp \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a,\chi) \in M'_{rew}\}\!\}$$

for all $1 \le j \le k$ as in Definition 5.1.12.

We need to differentiate between two possible cases again:

– **3.1)** If there is no pair $(t',\pi') \in A''_x$ with a position above π (i.e., $\pi' \le \pi$), then we can not perform any rewrite step with our new labeling. Hence, we use a split-node to mirror the tree structure without changing the sets in the labeling. This means that we set $A''_{y_j} \coloneqq A''_x$ for all $1 \le j \le k$. It remains to show that our induction hypothesis is still satisfied for this new labeling, i.e., we have $A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X}) \subseteq A''_{y_j}$ for all $1 \le j \le k$.

  Let $1 \le j \le k$ and $(b,\rho) \in A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X})$. We have the following possibilities:

  * If $(b,\rho) \in M_j^{+'}$, then there is $(r',\chi) \in C_j \subseteq dp(r_j)$ such that $(b,\rho) = (r'\sigma, \pi.\chi)$. Since our used dependency pair is inside of Y for some $Y \in \overline{Y}$ but not contained in $\bigcup_{X \in \overline{X}} X$, we know by (5.22) that $Y \not\geq^*_{\mathfrak{G}} X$ for all $X \in \overline{X}$. Intuitively, we started with a dependency tuple from $\overline{X}$, reaching Y, so we cannot reach any $X \in \overline{X}$ anymore. By definition of the dependency graph, this means that there is no $Z \in W$ with $Z >^*_{\mathfrak{G}} X$ for some $X \in \overline{X}$ such that there is a dependency tuple $(\ell'^\#, \ell') \to \ldots \in Z$, and a ground substitution $\sigma' \in \mathrm{Sub}(\Sigma, \mathcal{V})$ with $b \xrightarrow{\mathrm{i}}^*_{\mathrm{np}(\mathcal{S})} \ell'^\#\sigma'$, and every proper subterm of $\ell'^\#\sigma'$ is in normal form w.r.t. $\mathcal{S}$. Hence, we have $(b,\rho) \in \mathrm{Junk}(A'_{y_j}, \overline{X})$ and this is a contradiction to our assumption that $(b,\rho) \in A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X})$. Thus, this case is not possible.
  * If $(b,\rho) \in \{\!\{(a[r_j\sigma]_{\chi_a}, \rho) \mid (a,\rho) \in M'_{rew}\}\!\}$, then we have a pair $(a,\rho) \in A_x$ and a position χ with $\rho.\chi = \pi$. By definition, the same rule that we applied to t is applied to a at position χ, so that we result with $b = a[r_j\sigma]_\chi$. The pair (a,ρ) can not be inside of A''_x (as there is no pair with a position above π in A''_x), hence we must have $(a,\rho) \in \mathrm{Junk}(A'_x, \overline{X})$ by our induction hypothesis. For the resulting pair $(b,\rho) = (a[r_j\sigma]_\chi, \rho)$, we have $a \xrightarrow{\mathrm{i}}_{\mathrm{np}(\mathcal{S})} a[r_j\sigma]_\chi = b$ and hence, we also have $(b,\rho) \in \mathrm{Junk}(A'_{y_j}, \overline{X})$, and this is a contradiction to our assumption that $(b,\rho) \in A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X})$. Thus, this case is not possible again.
  * If we have $(b,\rho) \in M'_\perp$, then $(b,\rho) \in A_x$. If we have $(b,\rho) \in \mathrm{Junk}(A'_x, \overline{X})$, then we also have $(b,\rho) \in \mathrm{Junk}(A'_{y_j}, \overline{X})$, and this is a contradiction to our assumption that $(b,\rho) \in A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X})$. Thus, we have

$(b,\rho) \in A'_x \setminus \mathrm{Junk}(A'_x, \overline{X}) \subseteq_{(IH)} A''_x$, then we get $(b,\rho) \in A''_x = A''_{y_j}$ by our construction.

– **3.2)** If there exists a pair $(t',\pi') \in A''_x$ with a position above π (i.e., $\pi' \leq \pi$), then we can mirror the rewrite step at the lowest possible position that is above π. Let $D \coloneqq \{(t',\pi') \in A''_x \mid \pi' \leq \pi\}$ be the set of all pairs in A''_x with a position above π. Furthermore, let $(t_{\max}, \pi_{\max})$ the pair in D with maximal position (i.e., there is no pair $(t'',\pi'') \in D$ with $\pi_{\max} < \pi''$). We can now mirror the rewrite step performed in our original computation tree with the main rewrite pair $(t_{\max}, \pi_{\max})$ in our new computation tree. To be precise, let τ be the position such that $\pi_{\max}.\tau = \pi$. We can then rewrite A''_x with the rule $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{S}$, the main rewrite pair $(t_{\max}, \pi_{\max}) \in A''_x$, the ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and the position τ, since $t_{\max}|_\tau = t = \ell^\# \sigma$ and all proper subterms of $\ell^\#\sigma$ are in normal form w.r.t. $\mathcal{S}$. This means that we have $A''_x \xrightarrow{\mathrm{i}}_{\mathcal{S}} \{\frac{p_{y_1}}{p_x} : B^x_1, \ldots, \frac{p_{y_k}}{p_x} : B^x_k\}$ with

$$A''_x = \{\!\{(t_{\max}, \pi_{\max})\}\!\} \uplus M''_{rew} \uplus M''_\perp \uplus M''_<$$

and

$$B^x_j = M_j^{+''} \uplus M''_\perp \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a,\chi) \in M''_{rew}\}\!\}$$

for all $1 \leq j \leq k$ as in Definition 5.1.17. We define the set in the labeling of the successors $y_1, \ldots, y_k$ in $\mathfrak{T}''$ to be $A''_{y_j} = B^x_j$ for all $1 \leq j \leq k$. It remains to show that our induction hypothesis is still satisfied for this new labeling, i.e., we have $A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X}) \subseteq A''_{y_j}$ for all $1 \leq j \leq k$.

Let $1 \leq j \leq k$ and $(b,\rho) \in A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X})$. We have the following possibilities:

* If $(b,\rho) \in M_j^{+'}$, then there is $(r',\chi) \in C_j \subseteq dp(r_j)$ such that $(b,\rho) = (r'\sigma, \pi.\chi)$. Since our used dependency pair is inside of Y for some $Y \in \overline{Y}$ but not contained in $\bigcup_{X \in \overline{X}} X$, we know by (5.22) that $Y \not\geq^*_{\mathfrak{G}} X$ for all $X \in \overline{X}$. Intuitively, we started with a dependency tuple from $\overline{X}$, reaching Y, so we cannot reach any $X \in \overline{X}$ anymore. By definition of the dependency graph, this means that there is no $Z \in W$ with $Z >^*_{\mathfrak{G}} X$ for some $X \in \overline{X}$ such that there is a dependency tuple $(\ell'^\#, \ell') \to \ldots \in Z$, and a ground substitution $\sigma' \in \mathrm{Sub}(\Sigma, \mathcal{V})$ with $b \xrightarrow{\mathrm{i}}^*_{\mathrm{np}(\mathcal{S})} \ell'^\#\sigma'$, and every proper subterm of $\ell'^\#\sigma'$ is in normal form w.r.t. $\mathcal{S}$. Hence, we have $(b,\rho) \in \mathrm{Junk}(A'_{y_j}, \overline{X})$ and this is a contradiction to our assumption that $(b,\rho) \in A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X})$. Thus, this case is not possible.
* If $(b,\rho) \in \{\!\{(a[r_j\sigma]_{\chi_a}, \rho) \mid (a,\rho) \in M'_{rew}\}\!\}$, then we have a pair $(a,\rho) \in A_x$ and a position χ with $\rho.\chi = \pi$. By definition, the same rule that we applied to t is applied to a at position χ, so that we result with $b = a[r_j\sigma]_\chi$. If we have $(a,\rho) \in \mathrm{Junk}(A'_x, \overline{X})$, then we also have $(b,\rho) = (a[r_j\sigma]_\chi, \rho) \in \mathrm{Junk}(A'_{y_j}, \overline{X})$, as this means $a \xrightarrow{\mathrm{i}}_{\mathrm{np}(\mathcal{S})} a[r_j\sigma]_\chi = b$, and this is a contradiction to our assumption that $(b,\rho) \in A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X})$. Thus, we have $(a,\rho) \in A'_x \setminus \mathrm{Junk}(A'_x, \overline{X}) \subseteq_{(IH)} A''_x$, then we also have $(a,\rho) \in D$ and $\rho \leq \pi_{\max}$. This means that $(a,\rho) \in M''_{rew}$ and there exists some position χ' so that $\rho.\chi' = \pi_{\max}$ and hence $(a[r_j\sigma]_{\chi'.\tau}, \rho) \in A''_{y_j}$. But since $\rho.\chi'.\tau = \pi_{\max}.\tau = \pi$, we have $\chi'.\tau = \chi$, get $a[r_j\sigma]_{\chi'.\tau} = a[r_j\sigma]_\chi = b$, and thus $(b,\rho) \in A''_{y_j}$.
* If we have $(b,\rho) \in M'_\perp$, then $(b,\rho) \in A_x$ and $\rho \perp \pi_{\max}.\tau$. If we have $(b,\rho) \in \mathrm{Junk}(A'_x, \overline{X})$, then we also have $(b,\rho) \in \mathrm{Junk}(A'_{y_j}, \overline{X})$, and this is

a contradiction to our assumption that $(b,\rho) \in A'_{y_j} \setminus \text{Junk}(A'_{y_j}, \overline{X})$. Thus, we have $(b,\rho) \in A'_x \setminus \text{Junk}(A'_x, \overline{X}) \subseteq_{(IH)} A''_x$. Furthermore, $\rho \bot \pi_{\max}.\tau$ and thus also $\rho \bot \pi$, since $\pi_{\max}.\chi = \pi$. We then also get $(b,\rho) \in M''_\bot \subseteq A''_{y_j}$.

4) If we have $A'_x \xrightarrow{\text{i}}_{\mathcal{S}} \{\frac{p_{y_1}}{p_x} : A'_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : A'_{y_k}\}$, then there is a rule $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{S}$, a main rewrite pair $(q,\pi) \in A'_x$, a ground substitution $\sigma \in \text{Sub}(\Sigma, \mathcal{V})$, and a position $\tau \in \mathbb{N}^+$ with $q|_\tau = \ell\sigma$, such that all proper subterms of $\ell\sigma$ are in normal form w.r.t. $\mathcal{S}$. Then

$$A'_x = \{\!\{(q,\pi)\}\!\} \uplus M'_{rew} \uplus M'_\bot \uplus M'_<$$

and

$$A'_{y_j} = M_j^{+'} \uplus M'_\bot \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a,\chi) \in M'_{rew}\}\!\}$$

for all $1 \leq j \leq k$ as in Definition 5.1.17.

We need to differentiate between two possible cases again:

- **4.1) If** we have no pair $(t',\pi') \in A''_x$ with a position above π (i.e., $\pi' \leq \pi$), then we can not perform any rewrite step with our new labeling. Hence, we use a split-node to mirror the tree structure without changing the sets in the labeling. This means that we set $A''_{y_j} \coloneqq A''_x$ for all $1 \leq j \leq k$. It remains to show that our induction hypothesis is still satisfied for this new labeling, i.e., we have $A'_{y_j} \setminus \text{Junk}(A'_{y_j}, \overline{X}) \subseteq A''_{y_j}$ for all $1 \leq j \leq k$.
  Let $1 \leq j \leq k$ and $(b,\rho) \in A'_{y_j} \setminus \text{Junk}(A'_{y_j}, \overline{X})$. We have the following possibilities:
  - If $(b,\rho) \in M_j^{+'} \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \rho) \mid (a,\rho) \in M'_{rew}\}\!\}$, then we have a pair $(a,\rho) \in A_x$ and a position χ with $\rho.\chi = \pi$. By definition, the same rule that we applied to t at position τ is applied to a at position $\chi.\tau$, so that we result with $b = a[r_j\sigma]_{\chi.\tau}$. The pair (a,ρ) can not be inside of A''_x (as there is no pair with a position above π in A''_x), hence we have $(a,\rho) \in \text{Junk}(A'_x, \overline{X})$ by our induction hypothesis. This means that there is no $Z \in W$ with $Z >^*_{\mathfrak{G}} X$ for some $X \in \overline{X}$ such that there is a dependency tuple $(\ell'^\#, \ell') \to \ldots \in Z$, a ground substitution $\sigma' \in \text{Sub}(\Sigma, \mathcal{V})$, and $a \xrightarrow{\text{i}}{}^*_{\text{np}(\mathcal{S})} \ell'^\#\sigma'$, and every proper subterm of $\ell'^\#\sigma'$ is in normal form w.r.t. $\mathcal{S}$. For the resulting pair $(b,\rho) = (a[r_j\sigma]_{\chi.\tau}, \rho)$, we have $a \xrightarrow{\text{i}}_{\text{np}(\mathcal{S})} a[r_j\sigma]_{\chi.\tau} = b$ and hence, we also have $(b,\rho) \in \text{Junk}(A'_{y_j}, \overline{X})$, and this is a contradiction to our assumption that $(b,\rho) \in A'_{y_j} \setminus \text{Junk}(A'_{y_j}, \overline{X})$. Thus, this case is not possible.
  - If we have $(b,\rho) \in M'_\bot$, then $(b,\rho) \in A_x$. If we have $(b,\rho) \in \text{Junk}(A'_x, \overline{X})$, then we also have $(b,\rho) \in \text{Junk}(A'_{y_j}, \overline{X})$, and this is a contradiction to our assumption that $(b,\rho) \in A'_{y_j} \setminus \text{Junk}(A'_{y_j}, \overline{X})$. Thus, we have $(b,\rho) \in A'_x \setminus \text{Junk}(A'_x, \overline{X}) \subseteq_{(IH)} A''_x$, then we get $(b,\rho) \in A''_x = A''_{y_j}$ by our construction.
- **4.2) If** there exists a pair $(t',\pi') \in A''_x$ with a position above π (i.e., $\pi' \leq \pi$), then we can mirror the rewrite step at the lowest possible position that is above π. Let $D \coloneqq \{(t',\pi') \in A''_x \mid \pi' \leq \pi\}$ be the set of all pairs in A''_x with a position above π. Furthermore, let $(t_{\max}, \pi_{\max})$ the pair in D with maximal position (i.e., there is no pair $(t'',\pi'') \in D$ with $\pi_{\max} < \pi''$). We can now mirror the rewrite step performed in our original computation tree with the main rewrite pair $(t_{\max}, \pi_{\max})$. To be precise, let χ be the position such that $\pi_{\max}.\chi = \pi$. We can then rewrite A''_x with the rule $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{S}$, the main rewrite

pair $(t_{\max}, \pi_{\max}) \in A''_x$, the substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and the position $\chi.\tau$ since $t_{\max}|_{\chi.\tau} = t|_\tau = \ell\sigma$ and all proper subterms of $\ell\sigma$ are in normal form w.r.t. $\mathcal{S}$. This means that we have $A''_x \xrightarrow{\mathrm{i}}_{\mathcal{S}} \{\frac{p_{y_1}}{p_x} : B^x_1, \ldots, \frac{p_{y_k}}{p_x} : B^x_k\}$ with

$$A''_x = \{\!\{(t_{\max}, \pi_{\max})\}\!\} \uplus M''_{rew} \uplus M''_\perp \uplus M''_<$$

and

$$B^x_j = M^{+''}_j \uplus M''_\perp \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a, \chi) \in M''_{rew}\}\!\}$$

for all $1 \leq j \leq k$ as in Definition 5.1.17. We define the set in the labeling of the successors $y_1, \ldots, y_k$ in $\mathfrak{T}''$ to be $A''_{y_j} = B^x_j$ for all $1 \leq j \leq k$. It remains to show that our induction hypothesis is still satisfied for this new labeling, i.e., we have $A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X}) \subseteq A''_{y_j}$ for all $1 \leq j \leq k$.

Let $1 \leq j \leq k$ and $(b, \rho) \in A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X})$. We have the following possibilities:

* If $(b, \rho) \in M^{+'}_j \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \rho) \mid (a, \rho) \in M'_{rew}\}\!\}$, then we have a pair $(a, \rho) \in A_x$ and a position χ' with $\rho.\chi' = \pi$. By definition, the same rule that we applied to t at position τ is applied to a at position $\chi'.\tau$, so that we result with $b = a[r_j\sigma]_{\chi'.\tau}$.
  If we have $(a, \rho) \in \mathrm{Junk}(A'_x, \overline{X})$, then we also have $(b, \rho) = (a[r_j\sigma]_{\chi'.\tau}, \rho) \in \mathrm{Junk}(A'_{y_j}, \overline{X})$, as this means $a \xrightarrow{\mathrm{i}}_{\mathrm{np}(\mathcal{S})} a[r_j\sigma]_{\chi'.\tau} = b$, and this is a contradiction to our assumption that $(b, \rho) \in A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X})$. Thus, we have $(a, \rho) \in A'_x \setminus \mathrm{Junk}(A'_x, \overline{X}) \subseteq_{(IH)} A''_x$, then we also have $(a, \rho) \in D$ and we have $\chi \leq \pi_{\max}$. This means that $(a, \rho) \in M''_{rew}$ and there exists some position χ'' so that $\rho.\chi'' = \pi_{\max}$ and hence $(a[r_j\sigma]_{\chi''.\chi.\tau}, \rho) \in A''_{y_j}$. But since $\rho.\chi''.\chi = \pi_{\max}.\chi = \pi$, we have $\chi''.\chi = \chi'$, get $a[r_j\sigma]_{\chi''.\chi.\tau} = a[r_j\sigma]_{\chi'.\tau} = b$, and thus $(b, \rho) \in A''_{y_j}$.
* If we have $(b, \rho) \in M'_\perp$, then $(b, \rho) \in A_x$. If we have $(b, \rho) \in \mathrm{Junk}(A'_x, \overline{X})$, then we also have $(b, \rho) \in \mathrm{Junk}(A'_{y_j}, \overline{X})$, and this is a contradiction to our assumption that $(b, \rho) \in A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j}, \overline{X})$. Thus, we have $(b, \rho) \in A'_x \setminus \mathrm{Junk}(A'_x, \overline{X}) \subseteq_{(IH)} A''_x$. Furthermore, $\rho\perp\pi_{\max}.\chi.\tau$ and thus also $\rho\perp\pi.\tau$, since $\pi_{\max}.\chi = \pi$. We then also get $(b, \rho) \in M''_\perp \subseteq A''_{y_j}$.

This was the last case and ended the construction and this proof part. We have now shown that (5.22) holds.

3. For every $X \in W$, the DP problem $(\bigcup_{X >^*_{\mathfrak{G}} Y} Y, \mathcal{S})$ is innermost AST.

Using (5.21) and (5.22), by well-founded induction on $>_{\mathfrak{G}}$ we now prove that

$$\text{for every } X \in W, \text{ the DP problem } (\textstyle\bigcup_{X >^*_{\mathfrak{G}} Y} Y, \mathcal{S}) \text{ is innermost AST.} \qquad (5.24)$$

For the base case, we consider an $X \in W$ that is minimal w.r.t. $>_{\mathfrak{G}}$. Hence, we have $\bigcup_{X >^*_{\mathfrak{G}} Y} Y = X$. By (5.21), $(X, \mathcal{S})$ must be innermost AST.

For the induction step, we consider an $X \in W$ and assume that every $(\bigcup_{Y >^*_{\mathfrak{G}} Z} Z, \mathcal{S})$ is innermost AST for every $Y \in W$ with $X >^+_{\mathfrak{G}} Y$. Let $\mathtt{Succ}(X) \coloneqq \{Y \in W \mid X >_{\mathfrak{G}} Y\} = \{Y_1, \ldots Y_m\}$ be the set of all direct successors of X. The induction hypothesis states that $(\bigcup_{Y_u >^*_{\mathfrak{G}} Z} Z, \mathcal{S})$ is innermost AST for all $1 \leq u \leq m$. We first prove by induction that for all $1 \leq u \leq m$ every $(\bigcup_{1 \leq i \leq u} \bigcup_{Y_i >^*_{\mathfrak{G}} Z} Z, \mathcal{S})$ is innermost AST.

In the inner induction base, we have $u = 1$ and hence $(\bigcup_{1 \leq i \leq u} \bigcup_{Y_i >^*_{\mathfrak{G}} Z} Z, \mathcal{S}) = (\bigcup_{Y_1 >^*_{\mathfrak{G}} Z} Z, \mathcal{S})$. By our outer induction hypothesis we know that $(\bigcup_{Y_1 >^*_{\mathfrak{G}} Z} Z, \mathcal{S})$ is innermost AST.

In the inner induction step, assume that the claim holds for some $1 \leq u < m$. Then $(\bigcup_{Y_{u+1} >^*_{\mathfrak{G}} Z} Z, \mathcal{S})$ is innermost AST by our outer induction hypothesis and $(\bigcup_{1 \leq i \leq u} \bigcup_{Y_i >^*_{\mathfrak{G}} Z} Z, \mathcal{S})$ is innermost AST by our inner induction hypothesis. By (5.22), we know that then $(\bigcup_{1 \leq i \leq u+1} \bigcup_{Y_i >^*_{\mathfrak{G}} Z} Z, \mathcal{S})$ is innermost AST as well. The conditions for (5.22) are clearly satisfied as we are using the reflexive, transitive closure in both $\bigcup_{1 \leq i \leq u} \bigcup_{Y_i >^*_{\mathfrak{G}} Z} Z$ and $\bigcup_{Y_{u+1} >^*_{\mathfrak{G}} Z} Z$.

Now, we have shown that $(\bigcup_{1 \leq i \leq m} \bigcup_{Y_i >^*_{\mathfrak{G}} Z} Z, \mathcal{S})$ is innermost AST. We know that $(X, \mathcal{S})$ is innermost AST by our assumption and that $(\bigcup_{1 \leq i \leq m} \bigcup_{Y_i >^*_{\mathfrak{G}} Z} Z, \mathcal{S})$ is innermost AST. Hence, by (5.22) we have $(\bigcup_{X >^*_{\mathfrak{G}} Y} Y, \mathcal{S})$ innermost AST. Again, the conditions of (5.22) are satisfied, since X is strictly greater than everything inside of $\bigcup_{X >^*_{\mathfrak{G}} Y} Y$ w.r.t. $>_{\mathfrak{G}}$.

4. $(\mathcal{P}, \mathcal{S})$ is innermost AST.

Now, in (5.24) we have shown that $(\bigcup_{X >^*_{\mathfrak{G}} Y} Y, \mathcal{S})$ for every $X \in W$ is innermost AST. Let $X_1, \ldots, X_m \in W$ be the maximal elements w.r.t. $>_{\mathfrak{G}}$. Finally, we prove by induction that every $(\bigcup_{1 \leq i \leq u} \bigcup_{X_i >^*_{\mathfrak{G}} Y} Y, \mathcal{S})$-chain converges with probability 1 for all $1 \leq u \leq m$ by (5.22), analogous to the previous induction. Again, the conditions of (5.22) are satisfied as we are dealing with the reflexive, transitive closure of $>_{\mathfrak{G}}$. In the end, we know that $(\bigcup_{1 \leq i \leq m} \bigcup_{X_i >^*_{\mathfrak{G}} Y} Y, \mathcal{S}) = (\mathcal{P}, \mathcal{S})$ is innermost AST and this ends the proof. ■

As for the non-probabilistic dependency graph, the construction of the probabilistic dependency graph is not computable for the same reason, and hence we need to work with an abstracted version again. Here, we can use the same abstraction function as before since the definition of the probabilistic dependency graph relies on the non-probabilistic TRS $\mathrm{np}(\mathcal{S})$, just like the non-probabilistic dependency graph.

Definition 5.4.11 (Abstracted Dependency Graph)**.** Let $(\mathcal{P}, \mathcal{S})$ be a DP problem. The *abstracted $(\mathcal{P}, \mathcal{S})$-dependency graph* is defined as the graph with node set $\mathcal{P}$. There is an arc from $(\ell^\#, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\}$ to $(\ell'^\#, \ell') \to \ldots$ if there is a pair $(t, \pi) \in C_i$ for some $1 \leq i \leq k$ such that t and $\ell'^\#$ are connectable w.r.t. $\ell^\#$.

Here, we can actually use the same proof as in Theorem 3.3.7 to show that the abstracted dependency graph is a supergraph of the normal dependency graph. Once again, this directly implies that the following computable processor is also sound and complete.

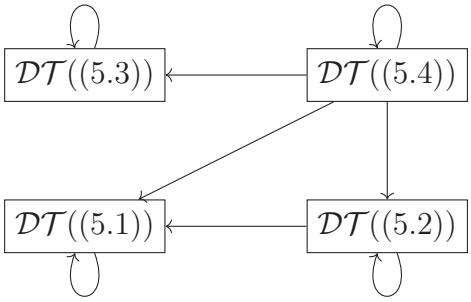
Theorem 5.4.12 (Computable Dependency Graph Processor)**.** *Let*

$$\mathrm{Proc}_{\mathtt{CDG}}(\mathcal{P}, \mathcal{S}) = \{(\mathcal{Q}_1, \mathcal{S}), \ldots, (\mathcal{Q}_n, \mathcal{S})\}$$

where $\mathcal{Q}_1, \ldots, \mathcal{Q}_n$ are the SCCs of the abstracted $(\mathcal{P}, \mathcal{S})$-dependency graph. Then $\mathrm{Proc}_{\mathtt{CDG}}$ *is sound and complete.*

Proof. Same proof as for Theorem 3.3.7. We only have argue in terms of dependency tuples $(\ell^\#, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\}$ instead of dependency pairs $\ell^\# \to r^\#$ in order to show that the abstracted dependency graph is a supergraph of the normal dependency graph. ■

Example 5.4.13 (Computable Dependency Graph Processor)**.** Let $\mathcal{R}_{div}$ be the PTRS from Example 5.0.1 and $\mathcal{DT}(\mathcal{R}_{div})$ be the set of its dependency tuples from Example 5.1.10. The abstracted $(\mathcal{DT}(\mathcal{R}_{div}), \mathcal{R}_{div})$-dependency graph has the same form as the $(\mathcal{DT}(\mathcal{R}_{div}), \mathcal{R}_{div})$-dependency graph:



The SCCs of this Graph are the four singleton sets $\{\mathcal{DT}((5.1))\}$, $\{\mathcal{DT}((5.2))\}$, $\{\mathcal{DT}((5.3))\}$ and $\{\mathcal{DT}((5.4))\}$. Hence, we get

$$\begin{aligned} & \mathrm{Proc}_{\mathtt{CDG}}((\mathcal{DT}(\mathcal{R}_{div}), \mathcal{R}_{div})) \\ = \; & \{(\{\mathcal{DT}((5.1))\}, \mathcal{R}_{div}), (\{\mathcal{DT}((5.2))\}, \mathcal{R}_{div}), (\{\mathcal{DT}((5.3))\}, \mathcal{R}_{div}), (\{\mathcal{DT}((5.4))\}, \mathcal{R}_{div})\} \end{aligned}$$

and remain with four smaller DP problems that only contain one dependency tuple.

Usable Pairs Processor

After applying the dependency graph processor, we may have dependency terms in the right-hand side of a dependency tuple that cannot be rewritten with any remaining dependency tuple. Hence, we create a new processor similar to the dependency graph processor, but it only works on the level of the sets inside the right-hand side of a dependency tuple. Similar to the usable rules processor, we remove every pair that cannot be used as a main rewrite pair in a $\xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}}$ step (no matter how we rewrite the term in the pair with $\mathcal{S}$). This processor was not needed in the non-probabilistic setting because there, we work with dependency pairs and not dependency tuples. Hence, the corresponding dependency pair for a not usable pair would already be removed by the dependency graph processor since it is not in any SCC of the dependency graph.

Example 5.4.14 (Usable Pairs)**.** Let $\mathcal{R}_{div}$ be the PTRS from Example 5.0.1 and $\mathcal{DT}(\mathcal{R}_{div})$ be the set of its dependency tuples from Example 5.1.10. After using the dependency graph, we get the following 4 DP problems.

$$(\{\mathcal{DT}((5.1))\}, \mathcal{R}_{div}),\ (\{\mathcal{DT}((5.2))\}, \mathcal{R}_{div}),\ (\{\mathcal{DT}((5.3))\}, \mathcal{R}_{div}),\ (\{\mathcal{DT}((5.4))\}, \mathcal{R}_{div})$$

Let us take a closer look at $(\{\mathcal{DT}((5.4))\}, \mathcal{R}_{div})$. Here, we have the only dependency tuple

$$\begin{aligned} \mathcal{DT}((5.4)) = \; & \Big(\mathsf{div}^{\#}(\mathsf{s}(x), \mathsf{s}(y)), \mathsf{div}(\mathsf{s}(x), \mathsf{s}(y))\Big) \\ \to \{ \; & \tfrac{1}{2} : \Big(\{\!\!\{\Big(\mathsf{div}^{\#}(\mathsf{s}(x), \mathsf{s}(y)), \varepsilon\Big)\}\!\!\}, \mathsf{div}(\mathsf{s}(x), \mathsf{s}(y))\Big), \\ & \tfrac{1}{2} : \Big(\{\!\!\{\Big(\mathsf{div}^{\#}(\mathsf{minus}(x, y), \mathsf{s}(y)), 1\Big), \Big(\mathsf{minus}^{\#}(x, y), 1.1\Big)\}\!\!\}, \\ & \qquad\qquad \mathsf{s}(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y)))\Big)\} \end{aligned}$$

The red pair has the tuple symbol $\mathsf{minus}^{\#}$ at the root of its term. However, the only dependency tuple can only rewrite terms with a $\mathsf{div}^{\#}$ root symbol. Hence, this pair can never be used as a main rewrite pair in a $\xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}}$ step, no matter how we rewrite this pair with $\mathcal{S}$.

Definition 5.4.15 (Usable Pairs)**.** Let $(\mathcal{P}, \mathcal{S})$ be a DP problem. Additionally, let $(\ell_1^{\#}, \ell_1) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in \mathcal{P}$ and $1 \leq i \leq k$. We call a pair $(t^{\#}, \pi) \in C_i$ *usable* w.r.t. $(\mathcal{P}, \mathcal{S})$ iff there is a dependency tuple $(\ell_2^{\#}, \ell_2) \to \ldots \in \mathcal{P}$, and ground

substitutions $\sigma_1, \sigma_2 \in \mathrm{Sub}(\Sigma, \mathcal{V})$ such that $t^\# \sigma_1 \xrightarrow{\mathrm{i}}{}^*_{\mathrm{np}(\mathcal{S})} \ell_2^\# \sigma_2$ and all proper subterms of $\ell_1^\# \sigma_1$ and $\ell_2^\# \sigma_2$ are in normal form w.r.t. $\mathcal{S}$.

For a set A, let $\mathcal{UP}(A, \mathcal{P}, \mathcal{S}) \coloneqq \{(t^\#, \pi) \in A \mid (t^\#, \pi) \text{ usable w.r.t. } (\mathcal{P}, \mathcal{S})\}$ be the set of all pairs in A that are usable. The transformation that removes all non-usable pairs in the right-hand side of all dependency tuples is denoted by:

$$\mathcal{T}_{\mathbf{UP}}(\mathcal{P}, \mathcal{S}) = \{(\ell^\#, \ell) \to \{p_1 : (\mathcal{UP}(C_1, \mathcal{P}, \mathcal{S}), r_1), \ldots, p_k : (\mathcal{UP}(C_k, \mathcal{P}, \mathcal{S}), r_k)\} \mid (\ell^\#, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in \mathcal{P}\}$$

Theorem 5.4.16 (Usable Pairs Processor)**.** *Let*

$$\mathrm{Proc}_{\mathtt{UP}}(\mathcal{P}, \mathcal{S}) = \{(\mathcal{T}_{\mathbf{UP}}(\mathcal{P}, \mathcal{S}), \mathcal{S})\}.$$

Then $\mathrm{Proc}_{\mathtt{UP}}$ *is sound and complete.*

Proof.

complete: Assume that $(\mathcal{T}_{\mathbf{UP}}(\mathcal{P}, \mathcal{S}), \mathcal{S})$ is not innermost AST. By Theorem 5.2.23, there exists a $(\mathcal{T}_{\mathbf{UP}}(\mathcal{P}, \mathcal{S}), \mathcal{S})$-computation tree $\mathfrak{T} = (V, E, L, P)$ that converges with probability < 1 and starts with $(1 : \{\!\{(t^\#, \varepsilon)\}\!\})$ such that $t^\# = \ell^\# \sigma$ for some dependency tuple $(\ell^\#, \ell) \to \{p_1 : (\mathcal{UP}(C_1, \mathcal{P}, \mathcal{S}), r_1), \ldots, p_k : (\mathcal{UP}(C_k, \mathcal{P}, \mathcal{S}), r_k)\} \in \mathcal{T}_{\mathbf{UP}}(\mathcal{P}, \mathcal{S})$ and some ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ such that every proper subterm of $\ell^\# \sigma$ is in normal form w.r.t. $\mathcal{S}$. We will now create a $(\mathcal{P}, \mathcal{S})$-computation tree $\mathfrak{T}' = (V, E, L', P)$ that also starts with $(1 : \{\!\{(t^\#, \varepsilon)\}\!\})$ with the same underlying tree structure, the same set P, and an adjusted labeling such that $p_x^{\mathfrak{T}} = p_x^{\mathfrak{T}'}$ for all $x \in V$. Since the tree structure and the probabilities are the same, we then get $|\mathfrak{T}|_{\mathrm{Leaf}} = |\mathfrak{T}'|_{\mathrm{Leaf}}$. To be precise, the set of leaves in $\mathfrak{T}$ is the same as the set of leaves in $\mathfrak{T}'$, and they have the same probabilities. Since $|\mathfrak{T}|_{\mathrm{Leaf}} < 1$ we thus have $|\mathfrak{T}'|_{\mathrm{Leaf}} < 1$. Therefore, there exists a $(\mathcal{P}, \mathcal{S})$-computation tree that converges with a probability < 1 and thus $(\mathcal{P}, \mathcal{S})$ is not innermost AST.

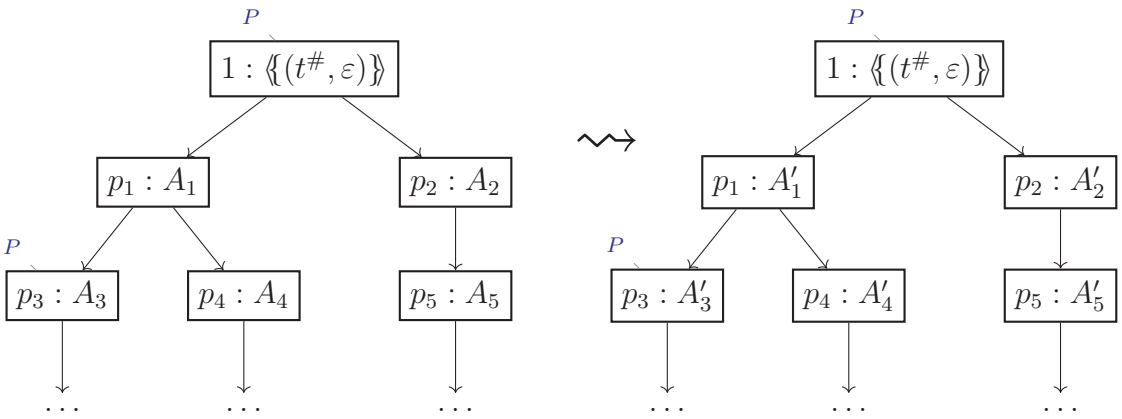


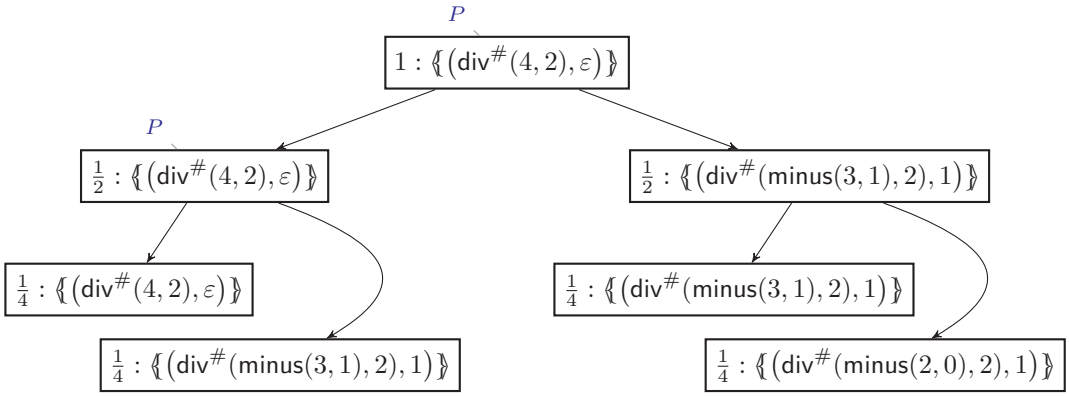
Figure 5.17: Construction in this proof direction.

The core idea of this construction is that adding pairs into the sets of the right-hand side of a dependency tuple does not impact the rewrite steps. Hence, every rewrite step without the unusable pairs is possible if we add them again.

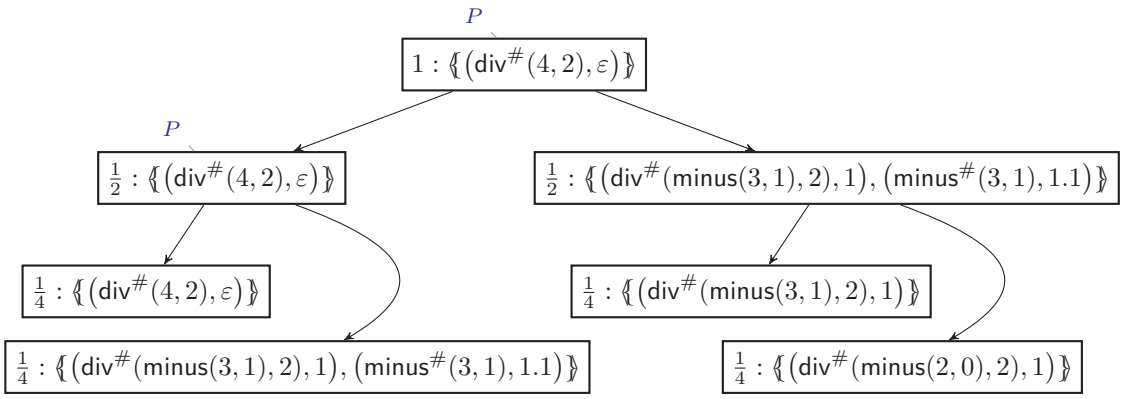
For example, if we use our DP problem $(\{\mathcal{DT}((5.4))'\}, \mathcal{R}_{div})$ with the dependency tuple

$$\begin{aligned}\mathcal{DT}((5.4))' = \quad & \Big(\mathsf{div}^{\#}(\mathsf{s}(x),\mathsf{s}(y)), \mathsf{div}(\mathsf{s}(x),\mathsf{s}(y))\Big) \\ & \to \{ \quad \tfrac{1}{2}: \quad \Big(\{\!\{\Big(\mathsf{div}^{\#}(\mathsf{s}(x),\mathsf{s}(y)),\varepsilon\Big)\}\!\}, \mathsf{div}(\mathsf{s}(x),\mathsf{s}(y))\Big), \\ & \qquad\quad \tfrac{1}{2}: \quad \Big(\{\!\{\Big(\mathsf{div}^{\#}(\mathsf{minus}(x,y),\mathsf{s}(y)),1\Big)\}\!\}, \mathsf{s}(\mathsf{div}(\mathsf{minus}(x,y),\mathsf{s}(y)))\Big)\}\end{aligned}$$

and look at the following $(\{\mathcal{DT}((5.4))'\}, \mathcal{R}_{div})$-computation tree $\mathfrak{T}$:



Then we can also add the missing pair $\Big(\mathsf{minus}^{\#}(x,y),1\Big)$ back into our dependency tuple and thus into the computation tree to get:



Now, this is a $(\{\mathcal{DT}((5.4))\}, \mathcal{R}_{div})$-computation tree and as one can see the additional pair does not impact the computation at all.

We now construct the new labeling L' for the $(\mathcal{P},\mathcal{S})$-computation tree $\mathfrak{T}'$ recursively. Let $X \subseteq V$ be the set of nodes, where we have already defined the labeling $L'(x)$. During our construction, we ensure that the following property holds:

$$\text{For every node } x \in X \text{ we have } A_v \subseteq A'_v. \tag{5.25}$$

We start by setting $A'_{\mathfrak{r}} := A_{\mathfrak{r}} = \{\!\{(t^{\#},\varepsilon)\}\!\}$. Here, our induction property (5.25) is clearly satisfied as we have $A'_{\mathfrak{r}} = A_{\mathfrak{r}}$.

As long as there is still an inner node $x \in X$ such that their successors are not contained in X, we do the following. Let $xE = \{y_1,\ldots,y_k\}$ be the set of its successors. We need to define the corresponding sets for the nodes $y_1,\ldots,y_k$.

Since x is not a leaf, we have $A_x \xrightarrow{\mathsf{i}}_{\mathcal{T}_{\mathbf{UP}}(\mathcal{P},\mathcal{S}),\mathcal{S}} \{\frac{p_{y_1}}{p_x} : A_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : A_{y_k}\}$ or $A_x \xrightarrow{\mathsf{i}}_{\mathcal{S}} \{\frac{p_{y_1}}{p_x} : A_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : A_{y_k}\}$.

If we have $A_x \xrightarrow{\mathsf{i}}_{\mathcal{T}_{\mathbf{UP}}(\mathcal{P},\mathcal{S}),\mathcal{S}} \{\frac{p_{y_1}}{p_x} : A_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : A_{y_k}\}$ using a dependency tuple $(\ell^\#, \ell) \to \{p_1 : (\mathcal{UP}(C_1, \mathcal{P}, \mathcal{S}), r_1), \ldots, p_k : (\mathcal{UP}(C_k, \mathcal{P}, \mathcal{S}), r_k)\} \in \mathcal{T}_{\mathbf{UP}}(\mathcal{P}, \mathcal{S})$, a main rewrite pair $(q, \pi) \in A_x$, and a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ such that $q = \ell^\#\sigma$ and all proper subterms of $\ell^\#\sigma$ are in normal form w.r.t. $\mathcal{S}$. Then

$$A_x = \{\!\{(q, \pi)\}\!\} \uplus M_{rew} \uplus M_\perp \uplus M_<$$

and

$$A_{y_j} = M_j^+ \uplus M_\perp \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a, \chi) \in M_{rew}\}\!\}$$

for all $1 \leq j \leq k$ as in Definition 5.1.12.

By our induction hypothesis, we get $(q, \pi) \in A_x \subseteq_{(IH)} A'_x$. Thus, we can rewrite the set A'_x using the dependency tuple $(\ell^\#, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in \mathcal{T}_{\mathbf{UP}}(\mathcal{P}, \mathcal{S})$, the main rewrite pair $(q, \pi) \in A'_x$, and the substitution σ as we have $q = \ell^\#\sigma$ and all proper subterms of $\ell^\#\sigma$ are in normal form w.r.t. $\mathcal{S}$.

This means that we have $A'_x \xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}} \{\frac{p_{y_1}}{p_x} : B_1^x, \ldots, \frac{p_{y_k}}{p_x} : B_k^x\}$ with

$$A'_x = \{\!\{(q, \pi)\}\!\} \uplus M'_{rew} \uplus M'_\perp \uplus M'_<$$

and

$$B_j^x = M_j^{+'} \uplus M'_\perp \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a, \chi) \in M'_{rew}\}\!\}$$

for all $1 \leq j \leq k$ as in Definition 5.1.12. We define the set in the labeling of the successors $y_1, \ldots, y_k$ in $\mathfrak{T}'$ to be $A'_{y_j} = B_j^x$ for all $1 \leq j \leq k$. It remains to show that our induction hypothesis is still satisfied for this new labeling, i.e., we have $A_{y_j} \subseteq A'_{y_j}$ for all $1 \leq j \leq k$.

Let $1 \leq j \leq k$ and $(b, \rho) \in A_{y_j}$. We have the following possibilities:

- If $(b, \rho) \in M_j^+$, then also $(b, \rho) \in M_j^{+'}$ (since $M_j^+ \subseteq M_j^{+'}$ as we are using the same dependency tuple, where we have $\mathcal{UP}(C_j, \mathcal{P}, \mathcal{S}) \subseteq C_j$ for the set inside the right-hand side, the same substitution and the same main rewrite pair) and thus $(b, \rho) \in A'_{y_j}$.
- If $(b, \rho) \in \{\!\{(a[r_j\sigma]_{\chi_a}, \rho) \mid (a, \rho) \in M_{rew}\}\!\}$, then we have a pair $(a, \rho) \in A_x$ and a position χ with $\rho.\chi = \pi$. By definition, the same rule that we applied to t is applied to a at position χ, so that we result with $b = a[r_j\sigma]_\chi$.

  We have $(a, \rho) \in A_x \subseteq_{(IH)} A'_x$, and hence $(a, \rho) \in M'_{rew}$, since $\rho < \pi$. Now the same rewrite step takes place at the same position as above, so that we get $(b, \rho) = (a[r_j\sigma]_\chi, \rho) \in A'_{y_j}$.
- If we have $(b, \rho) \in M_\perp$, then $(b, \rho) \in A_x \subseteq_{(IH)} A'_x$. Furthermore, $\rho \perp \pi$ and thus also $(b, \rho) \in M'_\perp$, and hence $(b, \rho) \in A'_{y_j}$.

If we have $A_x \xrightarrow{\mathsf{i}}_{\mathcal{S}} \{\frac{p_{y_1}}{p_x} : A_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : A_{y_k}\}$, then there is a rule $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{S}$, a main rewrite pair $(q, \pi) \in A_x$, a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and a position $\tau \in \mathbb{N}^+$ with $q|_\tau = \ell\sigma$, such that all proper subterms of $\ell\sigma$ are in normal form w.r.t. $\mathcal{S}$. Then

$$A_x = \{\!\{(t, \pi)\}\!\} \uplus M_{rew} \uplus M_\perp \uplus M_<$$

and

$$A_{y_j} = M_j^+ \uplus M_\perp \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a, \chi) \in M_{rew}\}\!\}$$

for all $1 \leq j \leq k$ as in Definition 5.1.17.

By our induction hypothesis, we get $(q, \pi) \in A_x \subseteq_{(IH)} A'_x$. Thus, we can rewrite the set A'_x using the rule $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{S}$, the main rewrite pair $(q, \pi) \in A'_x$, the substitution σ, and the position τ as we have $q|_\tau = \ell\sigma$ and all proper subterms of $\ell\sigma$ are in normal form w.r.t. $\mathcal{S}$.

This means that we have $A'_x \xrightarrow{\mathsf{i}}_{\mathcal{S}} \{\frac{p_{y_1}}{p_x} : B^x_1, \ldots, \frac{p_{y_k}}{p_x} : B^x_k\}$ with

$$A'_x = \{\!\{(q, \pi)\}\!\} \uplus M'_{rew} \uplus M'_\perp \uplus M'_<$$

and

$$B^x_j = M_j^{+'} \uplus M'_\perp \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a, \chi) \in M'_{rew}\}\!\}$$

for all $1 \leq j \leq k$ as in Definition 5.1.17. We define the set in the labeling of the successors $y_1, \ldots, y_k$ in $\mathfrak{T}'$ to be $A'_{y_j} = B^x_j$ for all $1 \leq j \leq k$. It remains to show that our induction hypothesis is still satisfied for this new labeling, i.e., we have $A_{y_j} \subseteq A'_{y_j}$ for all $1 \leq j \leq k$.

Let $1 \leq j \leq k$ and $(b, \rho) \in A_{y_j}$. We have the following possibilities:

- If $(b, \rho) \in M_j^+ \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \rho) \mid (a, \rho) \in M_{rew}\}\!\}$, then we have a pair $(a, \rho) \in A_x$ and a position $\chi \in \mathbb{N}^*$ with $\rho.\chi = \pi$. By definition, the same rule that we applied to t at position τ is applied to a at position $\chi.\tau$, so that we result with $b = a[r_j\sigma]_{\chi.\tau}$.

  We have $(a, \rho) \in A_x \subseteq_{(IH)} A'_x$, and hence $(a, \rho) \in \{\!\{(t, \pi)\}\!\} \uplus M'_{rew}$, since $\rho \leq \pi$. Now the same rewrite step takes place at the same position as above, so that we get $(b, \rho) = (a[r_j\sigma]_{\chi.\tau}, \rho) \in A'_{y_j}$.

- If we have $(b, \rho) \in M_\perp$, then $(b, \rho) \in A_x \subseteq_{(IH)} A'_x$. Furthermore, $\rho \perp \pi.\tau$ and thus also $(b, \rho) \in M'_\perp$, and hence $(b, \rho) \in A'_{y_j}$.

sound: Assume that $(\mathcal{P}, \mathcal{S})$ is not innermost AST. By Theorem 5.2.23, there exists a $\mathfrak{T} = (V, E, L, P)$ be a $(\mathcal{P}, \mathcal{S})$-computation tree that converges with probability < 1 and starts with $\{\!\{(t^\#, \varepsilon)\}\!\}$ such that $t^\# = \ell^\#\sigma$ for some rule $(\ell^\#, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in \mathcal{P}$ and some ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and every proper subterm of $\ell^\#\sigma$ is in normal form w.r.t. $\mathcal{S}$. We will now create a $(\mathcal{T}_{\mathbf{UP}}(\mathcal{P}, \mathcal{S}), \mathcal{S})$-computation tree $\mathfrak{T}' = (V, E, L', P)$ that also starts with $\{\!\{(t^\#, \varepsilon)\}\!\}$ with the same underlying tree structure, the same set P, and an adjusted labeling such that $p^{\mathfrak{T}}_x = p^{\mathfrak{T}'}_x$ for all $x \in V$. Since the tree structure and the probabilities are the same, we then get $|\mathfrak{T}|_{\mathrm{Leaf}} = |\mathfrak{T}'|_{\mathrm{Leaf}}$. To be precise, the set of leaves in $\mathfrak{T}$ is the same as the set of leaves in $\mathfrak{T}'$, and they have the same probabilities. Since $|\mathfrak{T}|_{\mathrm{Leaf}} < 1$ we thus have $|\mathfrak{T}'|_{\mathrm{Leaf}} < 1$. Therefore, there exists a $(\mathcal{T}_{\mathbf{UP}}(\mathcal{P}, \mathcal{S}), \mathcal{S})$-computation tree that converges with a probability < 1 and thus $(\mathcal{T}_{\mathbf{UP}}(\mathcal{P}, \mathcal{S}), \mathcal{S})$ is not innermost AST.

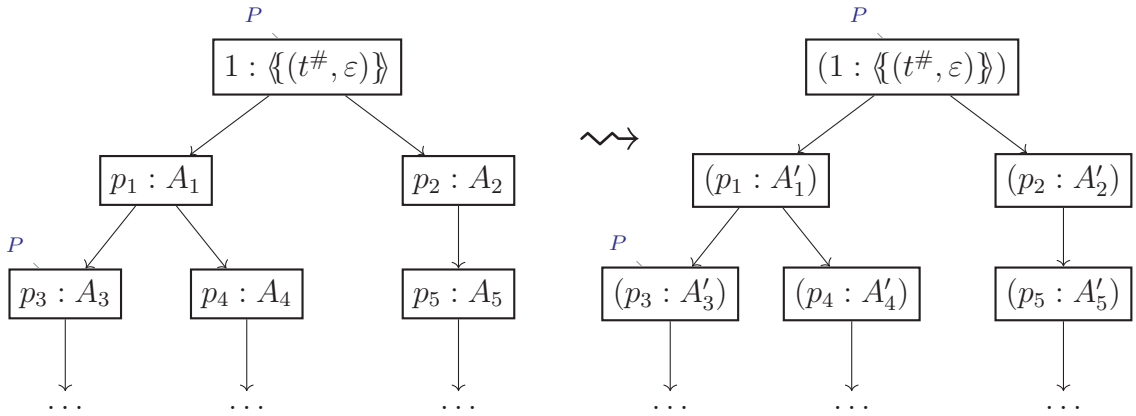


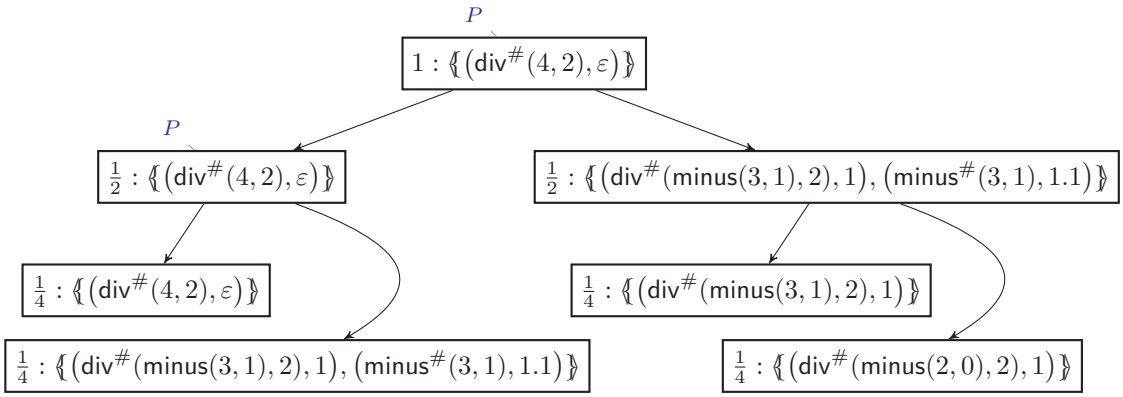
Figure 5.18: Construction in this proof direction.

The core idea of this construction is that pairs that cannot be rewritten to the left-hand side of a dependency tuple using $\xrightarrow{\mathsf{i}}_{\mathcal{S}}$ steps can never be used as a main rewrite pair for a $\xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}}$ step. This means that we can simply remove those pairs from the rules and every set in our computation tree and still result in a valid computation tree. If we use an unusable pair as a main rewrite pair for a $\xrightarrow{\mathsf{i}}_{\mathcal{S}}$ step, then we can change the main rewrite pair to mirror this step for all usable pairs, or if this step only impacts unusable pairs, then we can use a split-node to do nothing but mirror the tree structure. The general construction is similar to the construction used in the proof of the dependency graph processor (Theorem 5.4.10).

For example, if we use our DP problem $(\{\mathcal{DT}((5.4))\}, \mathcal{R}_{div})$ with the dependency tuple

$$\begin{aligned}\mathcal{DT}((5.4)) = \quad & \Big(\mathsf{div}^\#(\mathsf{s}(x),\mathsf{s}(y)), \mathsf{div}(\mathsf{s}(x),\mathsf{s}(y))\Big) \\ & \to \{ \quad \tfrac{1}{2} : \quad \Big(\{\!\{\Big(\mathsf{div}^\#(\mathsf{s}(x),\mathsf{s}(y)),\varepsilon\Big)\}\!\}, \mathsf{div}(\mathsf{s}(x),\mathsf{s}(y))\Big), \\ & \qquad\quad \tfrac{1}{2} : \quad \Big(\{\!\{\Big(\mathsf{div}^\#(\mathsf{minus}(x,y),\mathsf{s}(y)),1\Big), \Big(\mathsf{minus}^\#(x,y),1.1\Big)\}\!\}, \\ & \qquad\qquad\qquad\qquad \mathsf{s}(\mathsf{div}(\mathsf{minus}(x,y),\mathsf{s}(y)))\Big)\}\end{aligned}$$

and look at the following $(\{\mathcal{DT}((5.4))\}, \mathcal{R}_{div})$-computation tree $\mathfrak{T}$:

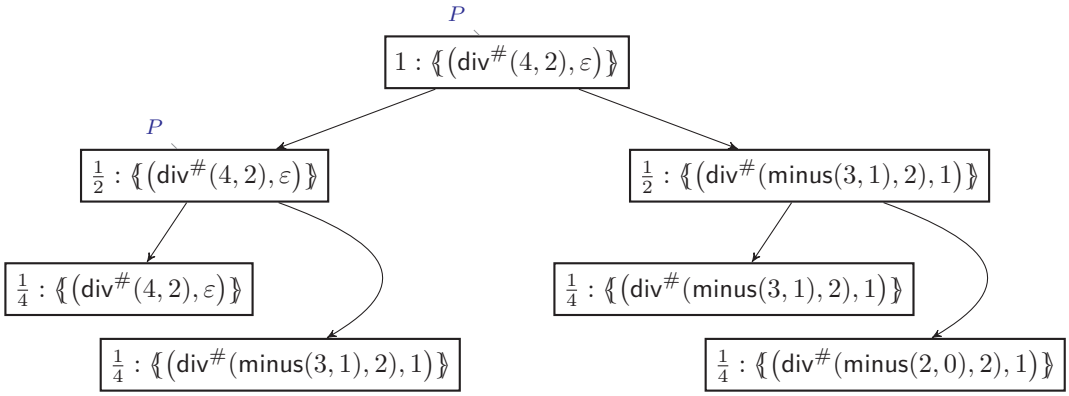


Then we can also remove the unusable pair $\Big(\mathsf{minus}^\#(x,y),1\Big)$, so that we are working

with the dependency tuple $\mathcal{DT}((5.4))'$:

$$\begin{aligned}\mathcal{DT}((5.4))' = \ & \Big(\mathsf{div}^{\#}(\mathsf{s}(x),\mathsf{s}(y)),\mathsf{div}(\mathsf{s}(x),\mathsf{s}(y))\Big)\\ & \to \{\ \tfrac{1}{2}:\ \Big(\{\!\{\Big(\mathsf{div}^{\#}(\mathsf{s}(x),\mathsf{s}(y)),\varepsilon\Big)\}\!\},\mathsf{div}(\mathsf{s}(x),\mathsf{s}(y))\Big),\\ & \qquad\ \tfrac{1}{2}:\ \Big(\{\!\{\Big(\mathsf{div}^{\#}(\mathsf{minus}(x,y),\mathsf{s}(y)),1\Big)\}\!\},\mathsf{s}(\mathsf{div}(\mathsf{minus}(x,y),\mathsf{s}(y)))\Big)\}\end{aligned}$$

Now, we can remove every pair corresponding to this removed pair $\Big(\mathsf{minus}^{\#}(x,y),1\Big)$ from our tree to get



And this is a $(\{\mathcal{DT}((5.4))'\},\mathcal{R}_{div})$-computation tree, and as one can see, removing the pair does not impact the computation at all since we can not rewrite the pair anyways with a dependency tuple.

We now construct the new labeling L' for the $(\mathcal{T}_{\mathbf{UP}}(\mathcal{P},\mathcal{S}),\mathcal{S})$-computation tree $\mathfrak{T}'$ recursively. Let $X \subseteq V$ be the set of nodes, where we have already defined the labeling $L'(x)$. During our construction, we ensure that the following property holds for every node $x \in X$:

$$\text{For every node } x \in X \text{ we have } A_v \setminus \mathrm{Junk}(A_v) \subseteq A'_v. \tag{5.26}$$

Here, $\mathrm{Junk}(A_v)$ denotes the set of all pairs in A_v that can never be used as a main rewrite pair for a dependency tuple from $\mathcal{P}$, as in the soundness proof of Theorem 5.4.10. To be precise, we define $(q,\pi) \in \mathrm{Junk}(A_v) :\Leftrightarrow (q,\pi) \in A_v$ and there is no dependency tuple $(\ell^{\#},\ell) \to \{p_1 : (C_1,r_1),\dots,p_k : (C_k,r_k)\} \in \mathcal{P}$ and ground substitution $\sigma \in \mathrm{Sub}\,(\Sigma,\mathcal{V})$, such that $q \xrightarrow{\mathsf{i}}{}^{*}_{\mathrm{np}(\mathcal{S})} \ell^{\#}\sigma$, and every proper subterm of $\ell^{\#}\sigma$ is in normal form.

We start by setting $A'_{\mathfrak{r}} \coloneqq A_{\mathfrak{r}} = \{\!\{(t,\varepsilon)\}\!\}$. Here, our induction property (5.26) is clearly satisfied as we have $A'_{\mathfrak{r}} = A_{\mathfrak{r}}$.

As long as there is still an inner node $x \in X$ such that their successors are not contained in X, we do the following. Let $xE = \{y_1,\dots,y_k\}$ be the set of its successors. We need to define the corresponding sets for the nodes $y_1,\dots,y_k$.

Since x is not a leaf and $\mathfrak{T}$ is a $(\mathcal{P},\mathcal{S})$-computation tree, we have

$$A_x \xrightarrow{\mathsf{i}}{}_{\mathcal{P},\mathcal{S}} \{\tfrac{p_{y_1}}{p_x} : A_{y_1},\dots,\tfrac{p_{y_k}}{p_x} : A_{y_k}\} \text{ or } A_x \xrightarrow{\mathsf{i}}{}_{\mathcal{S}} \{\tfrac{p_{y_1}}{p_x} : A_{y_1},\dots,\tfrac{p_{y_k}}{p_x} : A_{y_k}\}$$

If we have $A_x \xrightarrow{\mathsf{i}}{}_{\mathcal{P},\mathcal{S}} \{\tfrac{p_{y_1}}{p_x} : A_{y_1},\dots,\tfrac{p_{y_k}}{p_x} : A_{y_k}\}$ using a dependency tuple $(\ell^{\#},\ell) \to \{p_1 : (C_1,r_1),\dots,p_k : (C_k,r_k)\} \in \mathcal{P}$, a main rewrite pair $(q,\pi) \in A_x$,

and a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ such that $q = \ell^{\#}\sigma$ and all proper subterms of $\ell^{\#}\sigma$ are in normal form w.r.t. $\mathcal{S}$. Then

$$A_x = \{\!\{(q, \pi)\}\!\} \uplus M_{rew} \uplus M_{\perp} \uplus M_{<}$$

and

$$A_{y_j} = M_j^{+} \uplus M_{\perp} \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a, \chi) \in M_{rew}\}\!\}$$

for all $1 \leq j \leq k$ as in Definition 5.1.12.

The pair (q, π) cannot be inside of $\mathrm{Junk}(A_x)$. Hence, we must have $(q, \pi) \in A_x \setminus \mathrm{Junk}(A_x) \subseteq_{(IH)} A'_x$. Thus, we can rewrite the set A'_x using the dependency tuple $(\ell^{\#}, \ell) \to \{p_1 : (\mathcal{UP}(C_1, \mathcal{P}, \mathcal{S}), r_1), \ldots, p_k : (\mathcal{UP}(C_k, \mathcal{P}, \mathcal{S}), r_k)\} \in \mathcal{T}_{\mathbf{UP}}(\mathcal{P}, \mathcal{S})$, the main rewrite pair $(q, \pi) \in A'_x$, and the substitution σ as we have $q = \ell^{\#}\sigma$ and all proper subterms of $\ell^{\#}\sigma$ are in normal form w.r.t. $\mathcal{S}$.

This means that we have $A'_x \xrightarrow{\mathrm{i}}_{\mathcal{T}_{\mathbf{UP}}(\mathcal{P},\mathcal{S}),\mathcal{S}} \{\frac{p_{y_1}}{p_x} : B_1^x, \ldots, \frac{p_{y_k}}{p_x} : B_k^x\}$ with

$$A'_x = \{\!\{(q, \pi)\}\!\} \uplus M'_{rew} \uplus M'_{\perp} \uplus M'_{<}$$

and

$$B_j^x = M_j^{+'} \uplus M'_{\perp} \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a, \chi) \in M'_{rew}\}\!\}$$

for all $1 \leq j \leq k$ as in Definition 5.1.12. We define the set in the labeling of the successors $y_1, \ldots, y_k$ in $\mathfrak{T}'$ to be $A'_{y_j} \coloneqq B_j^x$ for all $1 \leq j \leq k$. It remains to show that our induction hypothesis is still satisfied for this new labeling, i.e., we have $A_{y_j} \setminus \mathrm{Junk}(A_{y_j}) \subseteq A'_{y_j}$ for all $1 \leq j \leq k$.

Let $1 \leq j \leq k$ and $(b, \rho) \in A_{y_j} \setminus \mathrm{Junk}(A_{y_j})$. We have the following possibilities:

- If $(b, \rho) \in M_j^{+}$, then there is $(r', \chi) \in C_j \subseteq dp(r_j)$ such that $(b, \rho) = (r'\sigma, \pi.\chi)$. If $(r', \chi) \in \mathcal{UP}(C_j, \mathcal{P}, \mathcal{S})$, then we also have $(b, \rho) = (r'\sigma, \pi.\chi) \in M_j^{+'} \subseteq A'_{y_j}$ since we use the same substitution and the same main rewrite pair. If $(r', \chi) \notin \mathcal{UP}(C_j, \mathcal{P}, \mathcal{S})$, then there is no dependency tuple $(\ell'^{\#}, \ell') \to \ldots \in \mathcal{P}$, and ground substitution $\sigma' \in \mathrm{Sub}(\Sigma, \mathcal{V})$ such that $b \xrightarrow{\mathrm{i}}{}^{*}_{\mathrm{np}(\mathcal{S})} \ell'^{\#}\sigma'$, and every proper subterm of $\ell'^{\#}\sigma'$ is in normal form w.r.t. $\mathcal{S}$, by definition of the usable pairs. Hence, we have $(b, \rho) = (r'\sigma, \pi.\chi) \in \mathrm{Junk}(A'_{y_j})$ and this is a contradiction to our assumption that $(b, \rho) \in A'_{y_j} \setminus \mathrm{Junk}(A'_{y_j})$, so this case is not possible.
- If $(b, \rho) \in \{\!\{(a[r_j\sigma]_{\chi_a}, \rho) \mid (a, \rho) \in M_{rew}\}\!\}$, then we have a pair $(a, \rho) \in A'_x$ and a position $\chi \in \mathbb{N}^{+}$ with $\rho.\chi = \pi$. By definition, the same rule that we applied to t is applied to a at position χ, so that we result with $b = a[r_j\sigma]_{\chi}$.

  If we have $(a, \rho) \in \mathrm{Junk}(A'_x)$, then we also have $(b, \rho) = (a[r_j\sigma]_{\chi}, \rho) \in \mathrm{Junk}(A_{y_j})$, as this means $a \xrightarrow{\mathrm{i}}_{\mathrm{np}(\mathcal{S})} a[r_j\sigma]_{\chi} = b$, and this is a contradiction to our assumption that $(b, \rho) \in A_{y_j} \setminus \mathrm{Junk}(A_{y_j})$. Thus, we have $(a, \rho) \in A_x \setminus \mathrm{Junk}(A_x) \subseteq_{(IH)} A'_x$, and $(a, \rho) \in M'_{rew}$, since $\rho < \pi$. Now the same rewrite step takes place at the same position as above, so that we get $(b, \rho) = (a[r_j\sigma]_{\chi}, \rho) \in A'_{y_j}$.
- If we have $(b, \rho) \in M_{\perp}$, then $(b, \rho) \in A_x$. If we have $(b, \rho) \in \mathrm{Junk}(A_x)$, then we also have $(b, \rho) \in \mathrm{Junk}(A_{y_j})$, and this is a contradiction to our assumption that $(b, \rho) \in A_{y_j} \setminus \mathrm{Junk}(A_{y_j})$. Thus, we have $(b, \rho) \in A_x \setminus \mathrm{Junk}(A_x) \subseteq_{(IH)} A'_x$. Furthermore, $\rho \perp \pi$ and thus also $(b, \rho) \in M'_{\perp}$, and hence $(b, \rho) \in A'_{y_j}$.

If we have $A_x \xrightarrow{\mathrm{i}}_{\mathcal{S}} \{\frac{p_{y_1}}{p_x} : A_{y_1}, \ldots, \frac{p_{y_k}}{p_x} : A_{y_k}\}$, then there is a rule $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{S}$, a main rewrite pair $(q, \pi) \in A_x$, a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and

a position $\tau \in \mathbb{N}^+$ with $q|_\tau = \ell\sigma$, such that all proper subterms of $\ell\sigma$ are in normal form w.r.t. $\mathcal{S}$. Then

$$A_x = \{\!\{(q, \pi)\}\!\} \uplus M_{rew} \uplus M_\perp \uplus M_<$$

and

$$A_{y_j} = M_j^+ \uplus M_\perp \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a, \chi) \in M_{rew}\}\!\}$$

for all $1 \leq j \leq k$ as in Definition 5.1.17.

We need to differentiate between two possible cases:

- If we have no pair $(t', \pi') \in A'_x$ with a position above π (i.e., $\pi' \leq \pi$), then we can not perform any rewrite step with our new labeling. Hence, we use a split-node to mirror the tree structure. This means that we set $A'_{y_j} \coloneqq A'_x$ for all $1 \leq j \leq k$. It remains to show that our induction hypothesis is still satisfied for this new labeling, i.e., we have $A_{y_j} \setminus \text{Junk}(A_{y_j}) \subseteq A'_{y_j}$ for all $1 \leq j \leq k$.
  Let $1 \leq j \leq k$ and $(b, \rho) \in A_{y_j} \setminus \text{Junk}(A_{y_j})$. We have the following possibilities:
  - If $(b, \rho) \in M_j^+ \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \rho) \mid (a, \rho) \in M_{rew}\}\!\}$, then we have a pair $(a, \rho) \in A_x$ and a position $\chi \in \mathbb{N}^*$ with $\rho.\chi = \pi$. By definition, the same rule that we applied to q at position τ is applied to a at position $\chi.\tau$, so that we result with $b = a[r_j\sigma]_{\chi.\tau}$. The pair (a, ρ) can not be inside of A'_x (as there is no pair with a position above π in A'_x), hence we have $(a, \rho) \in \text{Junk}(A_x)$ by our induction hypothesis. For the resulting pair $(b, \rho) = (a[r_j\sigma]_{\chi.\tau}, \rho)$, we have $a \xrightarrow{\text{i}}_{\text{np}(\mathcal{S})} a[r_j\sigma]_{\chi.\tau} = b$ and hence, we also have $(b, \rho) \in \text{Junk}(A_{y_j})$, and this is a contradiction to our assumption that $(b, \rho) \in A_{y_j} \setminus \text{Junk}(A_{y_j})$. Thus, this case is not possible.
  - If we have $(b, \rho) \in M_\perp$, then $(b, \rho) \in A_x$. If we have $(b, \rho) \in \text{Junk}(A_x)$, then we also have $(b, \rho) \in \text{Junk}(A_{y_j})$, and this is a contradiction to our assumption that $(b, \rho) \in A_{y_j} \setminus \text{Junk}(A_{y_j})$. Thus, we have $(b, \rho) \in A_x \setminus \text{Junk}(A_x) \subseteq_{(IH)} A'_x$ and $(b, \rho) \in A'_x = A'_{y_j}$ by our construction.
- If there exists a pair $(t', \pi') \in A'_x$ with a position above π (i.e., $\pi' \leq \pi$), then we can mirror the rewrite step at the lowest possible position that is above π. Let $D \coloneqq \{(t', \pi') \in A'_x \mid \pi' \leq \pi\}$ be the set of all pairs in A'_x with a position above π. Furthermore, let $(t_{\max}, \pi_{\max})$ the pair in D with maximal position (i.e., there is no pair $(t'', \pi'') \in D$ with $\pi_{\max} < \pi''$). We can now mirror the rewrite step performed in our original computation tree with the main rewrite pair $(t_{\max}, \pi_{\max})$. To be precise, let $\chi \in \mathbb{N}^*$ be the position such that $\pi_{\max}.\chi = \pi$. We can then rewrite A'_x with the rule $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{S}$, the main rewrite pair $(t_{\max}, \pi_{\max}) \in A'_x$, the substitution $\sigma \in \text{Sub}(\Sigma, \mathcal{V})$, and the position $\chi.\tau$ since $t_{\max}|_{\chi.\tau} = t|_\tau = \ell\sigma$ and all proper subterms of $\ell\sigma$ are in normal form w.r.t. $\mathcal{S}$. This means that we have $A'_x \xrightarrow{\text{i}}_{\triangleright\mathcal{S}} \{\frac{p_{y_1}}{p_x} : B_1^x, \ldots, \frac{p_{y_k}}{p_x} : B_k^x\}$ with

  $$A'_x = \{\!\{(t_{\max}, \pi_{\max})\}\!\} \uplus M'_{rew} \uplus M'_\perp \uplus M'_<$$

  and

  $$B_j^x = M_j^{+'} \uplus M'_\perp \uplus \{\!\{(a[r_j\sigma]_{\chi_a}, \chi) \mid (a, \chi) \in M'_{rew}\}\!\}$$

  for all $1 \leq j \leq k$ as in Definition 5.1.17. We define the set in the labeling of the successors $y_1, \ldots, y_k$ in $\mathfrak{T}'$ to be $A'_{y_j} = B_j$ for all $1 \leq j \leq k$. It remains to show that our induction hypothesis is still satisfied for this new labeling, i.e., we have $A_{y_j} \setminus \text{Junk}(A_{y_j}) \subseteq A'_{y_j}$ for all $1 \leq j \leq k$.
  Let $1 \leq j \leq k$ and $(b, \rho) \in A_{y_j} \setminus \text{Junk}(A_{y_j})$. We have the following possibilities:

* If $(b,\rho) \in M_j^+ \uplus \{\!\{(a[r_j\sigma]_{\chi_a},\rho) \mid (a,\rho) \in M_{rew}\}\!\}$, then we have a pair $(a,\rho) \in A_x$ and a position χ' with $\rho.\chi' = \pi$. By definition, the same rule that we applied to q at position τ is applied to a at position $\chi'.\tau$, so that we result with $b = a[r_j\sigma]_{\chi'.\tau}$.
  If we have $(a,\rho) \in \mathrm{Junk}(A_x)$, then we also have $(b,\rho) = (a[r_j\sigma]_{\chi'.\tau},\rho) \in \mathrm{Junk}(A_{y_j})$, as this means $a \xrightarrow{\mathrm{i}}_{\mathrm{np}(\mathcal{S})} a[r_j\sigma]_{\chi'.\tau} = b$, and this is a contradiction to our assumption that $(b,\rho) \in A_{y_j} \setminus \mathrm{Junk}(A_{y_j})$. Thus, we have $(a,\rho) \in A_x \setminus \mathrm{Junk}(A_x) \subseteq_{(IH)} A'_x$, and we also have $(a,\rho) \in D$, and $\chi \leq \pi_{\max}$. This means that $(a,\rho) \in M'_{rew}$ and there exists some position χ'' so that $\rho.\chi'' = \pi_{\max}$ and hence $(a[r_j\sigma]_{\chi''.\chi.\tau},\rho) \in A'_{y_j}$. But since $\rho.\chi''.\chi = \pi_{\max}.\chi = \pi$, we have $\chi''.\chi = \chi'$, get $a[r_j\sigma]_{\chi''.\chi.\tau} = a[r_j\sigma]_{\chi'.\tau} = b$, and thus $(b,\rho) \in A'_{y_j}$.
* If we have $(b,\rho) \in M_\perp$, then $(b,\rho) \in A_x$. If we have $(b,\rho) \in \mathrm{Junk}(A_x)$, then we also have $(b,\rho) \in \mathrm{Junk}(A_{y_j})$, and this is a contradiction to our assumption that $(b,\rho) \in A_{y_j} \setminus \mathrm{Junk}(A_{y_j})$. Thus, we have $(b,\rho) \in A_x \setminus \mathrm{Junk}(A_x) \subseteq_{(IH)} A'_x$. Furthermore, $\rho\perp\pi_{\max}.\chi.\tau$ and thus also $\rho\perp\pi.\tau$, since $\pi_{\max}.\chi = \pi$. We then also get $(b,\rho) \in M'_\perp$ and thus $(b,\rho) \in A'_{y_j}$. ■

Once again, this definition is not computable, and we have to use our abstraction.

Definition 5.4.17 (Abstracted Usable Pairs)**.** Let $(\mathcal{P},\mathcal{S})$ be a DP problem. Additionally, let $(\ell^\#,\ell) \to \{p_1 : (C_1,r_1),\ldots,p_k : (C_k,r_k)\} \in \mathcal{P}$ and $1 \leq i \leq k$. We call a pair $(t,\pi) \in C_i$ *abstract usable* w.r.t. $(\mathcal{P},\mathcal{S})$ iff there is another rule $(\ell'^\#,\ell') \to \ldots \in \mathcal{P}$, and ground substitutions $\sigma_1,\sigma_2 \in \mathrm{Sub}(\Sigma,\mathcal{V})$ such that t and $\ell'^\#$ are connectable w.r.t. $\ell^\#$.

For a set A, let $\mathcal{AUP}(A,\mathcal{P},\mathcal{S}) \coloneqq \{(t^\#,\pi) \in A \mid (t,\pi) \text{ abstract usable w.r.t. } (\mathcal{P},\mathcal{S})\}$ be the set of all pairs in A that are abstract usable. The transformation that removes all abstract non-usable pairs in the right-hand side of dependency tuples is denoted by:

$$\mathcal{T}_{\mathbf{AUP}}(\mathcal{P},\mathcal{S}) = \{(\ell^\#,\ell) \to \{p_1 : (\mathcal{AUP}(C_1,\mathcal{P},\mathcal{S}),r_1),\ldots,p_k : (\mathcal{AUP}(C_k,\mathcal{P},\mathcal{S}),r_k)\} \mid (\ell^\#,\ell) \to \{p_1 : (C_1,r_1),\ldots,p_k : (C_k,r_k)\} \in \mathcal{P}\}$$

Theorem 5.4.18 (Computable Usable Pairs Processor)**.** *Let*

$$\mathrm{Proc}_{\mathtt{CUP}}(\mathcal{P},\mathcal{S}) = \{(\mathcal{T}_{\mathbf{AUP}}(\mathcal{P},\mathcal{S}),\mathcal{S})\}.$$

Then $\mathrm{Proc}_{\mathtt{CUP}}$ *is sound and complete.*

Proof. We again use the same proof as in Theorem 3.3.7 to show that $\mathcal{UP}(C,\mathcal{P},\mathcal{S}) \subseteq \mathcal{AUP}(C,\mathcal{P},\mathcal{S})$ holds for all sets C. We only have argue in terms of dependency tuples $(\ell^\#,\ell) \to \{p_1 : (C_1,r_1),\ldots,p_k : (C_k,r_k)\}$ instead of dependency pairs $\ell^\# \to r^\#$. This means that our computable usable pairs processor only removes pairs that would also be removed by the usable pairs processors of Theorem 5.4.22. Again, this approximation does not interfere with the soundness nor the completeness but just with the effectiveness of this processor. The constructions in both directions of the proof of Theorem 5.4.16 still work if we remove fewer pairs. ■

Example 5.4.19 (Computable Usable Pairs Processor)**.** Let $\mathcal{R}$ be the PTRS from Example 5.0.1 and $\mathcal{DT}(\mathcal{R})$ be the set of its dependency tuples from Example 5.1.10. After the execution of the computable dependency graph processor in Example 5.4.13 we get the following four DP problems

$$(\{\mathcal{DT}((5.1))\},\mathcal{R}),(\{\mathcal{DT}((5.2))\},\mathcal{R}),(\{\mathcal{DT}((5.3))\},\mathcal{R}),(\{\mathcal{DT}((5.4))\},\mathcal{R})$$

Let us look at the last one in more detail. Here, we have the dependency tuple

$$\begin{aligned}\mathcal{DT}((5.4)) \;=\; & \Big(\mathsf{div}^{\#}(\mathsf{s}(x),\mathsf{s}(y)),\mathsf{div}(\mathsf{s}(x),\mathsf{s}(y))\Big)\\ & \to \{\;\tfrac{1}{2}:\;\Big(\{\!\{\Big(\mathsf{div}^{\#}(\mathsf{s}(x),\mathsf{s}(y)),\varepsilon\Big)\}\!\},\mathsf{div}(\mathsf{s}(x),\mathsf{s}(y))\Big),\\ & \qquad\tfrac{1}{2}:\;\Big(\{\!\{\Big(\mathsf{div}^{\#}(\mathsf{minus}(x,y),\mathsf{s}(y)),1\Big)\}\!\},\Big(\mathsf{minus}^{\#}(x,y),1.1\Big)\}\!\},\\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathsf{s}(\mathsf{div}(\mathsf{minus}(x,y),\mathsf{s}(y)))\Big)\}\end{aligned}$$

As described in Example 5.4.14, the red pair inside of the right-hand side is not abstract usable. Hence, we can remove it and result with the following dependency tuple:

$$\begin{aligned}\mathcal{DT}((5.4))' \;=\; & \Big(\mathsf{div}^{\#}(\mathsf{s}(x),\mathsf{s}(y)),\mathsf{div}(\mathsf{s}(x),\mathsf{s}(y))\Big)\\ & \to \{\;\tfrac{1}{2}:\;\Big(\{\!\{\Big(\mathsf{div}^{\#}(\mathsf{s}(x),\mathsf{s}(y)),\varepsilon\Big)\}\!\},\mathsf{div}(\mathsf{s}(x),\mathsf{s}(y))\Big),\\ & \qquad\tfrac{1}{2}:\;\Big(\{\!\{\Big(\mathsf{div}^{\#}(\mathsf{minus}(x,y),\mathsf{s}(y)),1\Big)\}\!\},\mathsf{s}(\mathsf{div}(\mathsf{minus}(x,y),\mathsf{s}(y)))\Big)\}\end{aligned}$$

Hence, we have

$$\mathrm{Proc}_{\mathsf{CUP}}(\{\mathcal{DT}((5.4))\},\mathcal{R}) = \{(\{\mathcal{DT}((5.4))'\},\mathcal{R})\}$$

and remain with a smaller DP problem that contains fewer pairs inside of the sets in the right-hand side of the dependency tuples. The other three DP problems stay the same, as every pair is usable.

Usable Rules Processor

The next processor we want to adapt is the usable rules processor. The idea and the actual definition do not change a lot in the probabilistic setting since it does not rely on the probabilities but only on the non-probabilistic structure analogous to the dependency graph. We only have to adjust the recursive definition to regard every term in the support of the distribution of the right-hand side of a rule.

Definition 5.4.20 (Usable Rules)**.** Let $(\mathcal{P},\mathcal{S})$ be a DP problem. For every $f \in \Sigma \uplus \Sigma^{\#}$ let $\mathrm{Rules}(\mathcal{S},f) \coloneqq \{\ell \to \mu \in \mathcal{S} \mid \mathrm{root}(\ell) = f, \ell \text{ has no redex as proper subterm}\}$. For any term t, the set of all *usable rules* $\mathcal{UR}(\mathcal{S},t)$ is inductively defined as

$$\begin{aligned}\mathcal{UR}(\mathcal{S},x) &= \varnothing\\ \mathcal{UR}(\mathcal{S},f(t_1,\dots,t_n)) &= \mathrm{Rules}(\mathcal{S},f) \cup \textstyle\bigcup_{i=1}^{n} \mathcal{UR}(\mathcal{S}',t_j) \cup\\ &\quad \textstyle\bigcup_{\ell\to\mu\in\mathrm{Rules}(\mathcal{S},f)} \bigcup_{r\in\mathrm{Supp}(\mu)} \mathcal{UR}(\mathcal{S}',r)\end{aligned}$$

where $\mathcal{S}' \coloneqq \mathcal{S} \setminus \mathrm{Rules}(\mathcal{S},f)$. The set of all *usable rules* for the DP problem $(\mathcal{P},\mathcal{S})$ is defined by

$$\mathcal{UR}(\mathcal{P},\mathcal{S}) \coloneqq \bigcup_{\ell^{\#}\to\mu\in\mathrm{proj}_1(\mathcal{P})} \bigcup_{B\in\mathrm{Supp}(\mu)} \bigcup_{(t,\pi)\in B} \mathcal{UR}(\mathcal{S},t)$$

Here, by $\mathrm{proj}_1(\mathcal{P})$ we denote the projection to the first component, i.e., we have

$$\mathrm{proj}_1(\mathcal{P}) \coloneqq \{\ell^{\#} \to \{p_1 : C_1,\dots,p_k : C_k\} \mid (\ell^{\#},\ell) \to \{p_1 : (C_1,r_1),\dots,p_k : (C_k,r_k)\} \in \mathcal{P}\}$$

Note that we ignore the projection to the second component of dependency tuples. We only need to find every rewrite rule that can be used to evaluate the sets in the right-hand side of a dependency tuple. The additional stored rewrite rule can only be applied if it is also in $\mathcal{S}$ so that we already check it in our recursive definition regarding the usable rules in $\mathcal{S}$.

Example 5.4.21 (Usable Rules)**.** Let $\mathcal{R}_{div}$ be the PTRS from Example 5.0.1 and $\mathcal{DT}(\mathcal{R}_{div})$ be the set of its dependency tuples from Example 5.1.10. After the execution of the computable dependency graph processor in Example 5.4.13 and the usable pairs processor in Example 5.4.19 we get the following four DP problems

$$(\{\mathcal{DT}((5.1))\}, \mathcal{R}_{div}), (\{\mathcal{DT}((5.2))\}, \mathcal{R}_{div}), (\{\mathcal{DT}((5.3))\}, \mathcal{R}_{div}), (\{\mathcal{DT}((5.4))'\}, \mathcal{R}_{div})$$

Let us look at the last one in more detail. Here, we have the dependency tuple

$$\begin{aligned}\mathcal{DT}((5.4))' \;=\; & \Big(\mathsf{div}^{\#}(\mathsf{s}(x),\mathsf{s}(y)),\mathsf{div}(\mathsf{s}(x),\mathsf{s}(y))\Big)\\ & \to \{\;\tfrac{1}{2}:\;\Big(\{\!\!\{\Big(\mathsf{div}^{\#}(\mathsf{s}(x),\mathsf{s}(y)),\varepsilon\Big)\}\!\!\},\mathsf{div}(\mathsf{s}(x),\mathsf{s}(y))\Big),\\ & \qquad\tfrac{1}{2}:\;\Big(\{\!\!\{\Big(\mathsf{div}^{\#}(\mathsf{minus}(x,y),\mathsf{s}(y)),1\Big)\}\!\!\},\mathsf{s}(\mathsf{div}(\mathsf{minus}(x,y),\mathsf{s}(y)))\Big)\}\end{aligned}$$

and still all of the four PTRS rules

$$\mathsf{minus}(x,\mathcal{O}) \to \{\tfrac{1}{2}:\mathsf{minus}(x,\mathcal{O}),\tfrac{1}{2}:x\} \tag{5.27}$$
$$\mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)) \to \{\tfrac{1}{2}:\mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)),\tfrac{1}{2}:\mathsf{minus}(x,y)\} \tag{5.28}$$
$$\mathsf{div}(\mathcal{O},\mathsf{s}(y)) \to \{\tfrac{1}{2}:\mathsf{div}(\mathcal{O},\mathsf{s}(y)),\tfrac{1}{2}:\mathcal{O}\} \tag{5.29}$$
$$\mathsf{div}(\mathsf{s}(x),\mathsf{s}(y)) \to \{\tfrac{1}{2}:\mathsf{div}(\mathsf{s}(x),\mathsf{s}(y)),\tfrac{1}{2}:\mathsf{s}(\mathsf{div}(\mathsf{minus}(x,y),\mathsf{s}(y)))\} \tag{5.30}$$

The rules (5.27) and (5.28) are usable as the root symbol of the left-hand side of those rules is minus and occurs in the right-hand side of $\mathcal{DT}((5.4))'$. However, the rules (5.29) and (5.30) are not usable, since div does not occur in the sets of the right-hand side of the dependency tuple nor in the right-hand side of the rules (5.27) and (5.28). Thus we have

$$\mathcal{UR}(\{\mathcal{DT}((5.4))\}, \mathcal{R}_{div}) = \{(5.27),(5.28)\}$$

Similar to the non-probabilistic usable rules processor, we only have soundness but no completeness. The reason is the same as before: the PTRS $\mathcal{S}$ in our DP problem $(\mathcal{P},\mathcal{S})$ has two functionalities. We can use it for a rewrite step, and we use it to determine the normal forms for our innermost evaluation strategy. If we add an additional TRS $\mathcal{Q}$ that is solely responsible for indicating what kind of normal forms we are interested in, then the usable rules processor would also be complete.

Theorem 5.4.22 (Usable Rules Processor)**.** *Let*

$$\mathrm{Proc}_{\mathtt{UR}}(\mathcal{P},\mathcal{S}) = \{(\mathcal{P},\mathcal{UR}(\mathcal{P},\mathcal{S}))\}$$

Then $\mathrm{Proc}_{\mathtt{UR}}$ *is sound but not complete.*

Proof. This proof is similar to the proof of the non-probabilistic usable rules processor (Theorem 3.3.11). We only have to adjust it to the new setting.

sound: Assume that $(\mathcal{P},\mathcal{S})$ is not AST. By Theorem 5.2.23, there exists a $(\mathcal{P},\mathcal{S})$-computation tree $\mathfrak{T} = (V,E,L,P)$ that converges with probability < 1 and starts with $\{\!\!\{(t^{\#},\varepsilon)\}\!\!\}$ such that $t^{\#} = \ell^{\#}\sigma$ for some dependency tuple $(\ell^{\#},\ell) \to \{p_1 : (C_1,r_1),\ldots,p_k : (C_k,r_k)\} \in \mathcal{P}$ and some ground substitution $\sigma \in \mathrm{Sub}\,(\Sigma,\mathcal{V})$, and every proper subterm of $\ell^{\#}\sigma$ is in normal form w.r.t. $\mathcal{S}$.

Rules $\ell \to \delta \in \mathcal{S}$ that are not usable (i.e., $\ell \to \delta \notin \mathcal{UR}(\mathcal{P},\mathcal{S})$) will never be used in such a $(\mathcal{P},\mathcal{S})$-computation tree that starts with the instantiated left-hand side of a dependency tuple such that every proper subterm is in normal form. Hence, we can

also view $\mathfrak{T}$ as a $(\mathcal{P},\mathcal{UR}(\mathcal{P},\mathcal{S}))$-computation tree that converges with probability < 1 and thus $(\mathcal{P},\mathcal{UR}(\mathcal{P},\mathcal{S}))$ is not AST.

It remains to prove that there is no node x in $\mathfrak{T}$ that, together with the labeling and its successors, represents a rewrite step with a rule from $\mathcal{S} \setminus \mathcal{UR}(\mathcal{P},\mathcal{S})$. Assume for a contradiction that there is some node $x \in V$ in this computation tree that (together with the successors and the labeling) represents a rewrite step with a rule from $\mathcal{S} \setminus \mathcal{UR}(\mathcal{P},\mathcal{S})$. Let $\mathfrak{r} = y_1 \dots y_m = x$ be the path from the root to x. W.l.o.G. we can assume that all of the used rewrite rules on this path are usable, i.e., there is no $1 \leq j < m$ such that the node y_j, together with the labeling and its successors, represents a rewrite step with a rule from $\mathcal{S} \setminus \mathcal{UR}(\mathcal{P},\mathcal{S})$.

Let (a,α) be the main rewrite pair, and ρ be the position for the rewrite step at node x. Rewriting with $\xrightarrow{\mathsf{i}}_{\mathcal{S}}$ can not introduce new pairs but only rewrite or remove existing ones. Hence, there must be a pair (a',α) that was introduced earlier in the path by some dependency tuple from $\mathcal{P}$ and has the same position as (a,α). Furthermore, we do not rewrite the pair (a',α) at the root position nor remove it completely until we reach x in the path. This means that there is some $1 \leq h < m$ such that $A_{y_h} \xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}} \{p_1 : B_1, \dots, p_k : B_k\}$ using a dependency tuple $(\ell^\#,\ell) \to \{p_1 : (C_1, r_1), \dots, p_k : (C_k, r_k)\}$, a ground substitution $\sigma \in \mathrm{Sub}\,(\Sigma,\mathcal{V})$ and a main rewrite pair $(q,\tau) \in A_{y_h}$ such that and

$$A_{y_h} = \{\!\{(q,\tau)\}\!\} \uplus M_{rew} \uplus M_\perp \uplus M_<$$

and

$$B_j = M_j^+ \uplus M_\perp \uplus \{\!\{(a[r_j\sigma]_{\chi a},\chi) \mid (a,\chi) \in M_{rew}\}\!\}$$

for all $1 \leq j \leq k$ as in Definition 5.1.12. Furthermore, we have $A_{y_{h+1}} = B_j$ for some $1 \leq j \leq k$ and $(a',\alpha) \in M_j^+$, so that there exists a pair $(r',\chi) \in C_j$ with $(a',\alpha) = (r'\sigma,\tau.\chi)$. Furthermore, in the rest of the path to x, we only use steps with a main rewrite pair position that is orthogonal to α or below α. Hence, this pair does not get removed anymore but only gets rewritten with $\xrightarrow{\mathsf{i}}_{\mathcal{S}}$ or remains the same (so for $\xrightarrow{\mathsf{i}}_{\mathcal{P},\mathcal{S}}$ steps the pair is either inside of $M_\perp$ or M_{rew} and for $\xrightarrow{\mathsf{i}}_{\mathcal{S}}$ steps the pair is either inside of $M_\perp$, M_{rew} or the main rewrite pair itself). This means that $a \xrightarrow{\mathsf{i}}^*_{\mathrm{np}(\mathcal{S})} a'$. The symbol $a|_\rho$ can not be inside of every substitution in this rewrite sequence (due to the innermost property). Hence, at least the root symbol of the used rewrite rule $\ell' \to \{p'_1 : r'_1, \dots, p'_k : r'_k\}$ for node x must be introduced by some left-hand side of a rewrite rule or the dependency tuple, but this is a contradiction, as this would imply that the rule that is used at node x would be usable since every rule in the path is usable by assumption. ■

Example 5.4.23 (Counterexample for completeness)**.** We can use the same counterexample as in Example 3.3.12. We only need to adjust it to our new type of dependency tuples. Consider the signature $\Sigma = \{\mathsf{f},\mathsf{a}\}$, a variable set with $\{x\} \subseteq \mathcal{V}$ and the PTRS $\mathcal{S}$ consisting of the following rule:

$$\mathsf{a} \to \{1 : \mathsf{a}\} \tag{5.31}$$

and a set of dependency tuples $\mathcal{P}$ consisting of the following dependency tuple:

$$(\mathsf{f}^\#(\mathsf{a},x),\mathsf{f}(\mathsf{a},x)) \to \{1 : (\{\!\{(\mathsf{f}(x,x),\varepsilon)\}\!\},\mathsf{f}(x,x))\} \tag{5.32}$$

Then, the DP problem $(\mathcal{P}, \mathcal{S})$ is innermost AST, since the only dependency tuple (5.32) can not be used. The reason is that the first component in the left-hand side, namely $\mathsf{f}^{\#}(\mathsf{a}, x)$, contains a proper subterm that is not in normal form w.r.t. $\mathcal{S}$, namely a. However, the rule (5.31) is not usable since the right-hand side of the only dependency tuple does not contain a anywhere. We get $\mathrm{Proc}_{\mathtt{UR}}(\mathcal{P}, \mathcal{S}) = \{(\mathcal{P}, \mathcal{UR}(\mathcal{P}, \mathcal{S}))\} = \{(\mathcal{P}, \varnothing)\}$. Now, the DP problem $(\mathcal{P}, \varnothing)$ is not innermost AST anymore since we have the $(\mathcal{P}, \varnothing)$-computation tree $\mathfrak{T}$, depicted on the right, that converges with probability 0. The reason here is that the proper subterm a is now in normal form w.r.t. the empty PTRS.

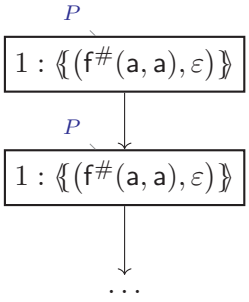


Figure 5.19: $\mathfrak{T}$

Example 5.4.24 (Usable Rules Processor)**.** Let $\mathcal{R}_{div}$ be the PTRS from Example 5.0.1 and $\mathcal{DT}(\mathcal{R}_{div})$ be the set of its dependency tuples from Example 5.1.10. After the execution of the computable dependency graph processor in Example 5.4.13 and the computable usable pairs processor in Example 5.4.19, we get the following four DP problems

$$(\{\mathcal{DT}((5.1))\}, \mathcal{R}), (\{\mathcal{DT}((5.2))\}, \mathcal{R}), (\{\mathcal{DT}((5.3))\}, \mathcal{R}), (\{\mathcal{DT}((5.4))'\}, \mathcal{R})$$

The usable rules for all four DP problems are

$$\mathcal{UR}(\{\mathcal{DT}((5.1))\}, \mathcal{R}_{div}) = \varnothing$$

$$\mathcal{UR}(\{\mathcal{DT}((5.2))\}, \mathcal{R}_{div}) = \varnothing$$

$$\mathcal{UR}(\{\mathcal{DT}((5.3))\}, \mathcal{R}_{div}) = \varnothing$$

$$\mathcal{UR}(\{\mathcal{DT}((5.4))'\}, \mathcal{R}_{div}) = \{(5.1), (5.2)\}$$

and thus, we get

$$\mathrm{Proc}_{\mathtt{UR}}((\mathcal{DT}((5.1)), \mathcal{R}_{div})) = \{(\{\mathcal{DT}((5.1))\}, \varnothing)\}$$

$$\mathrm{Proc}_{\mathtt{UR}}((\mathcal{DT}((5.2)), \mathcal{R}_{div})) = \{(\{\mathcal{DT}((5.2))\}, \varnothing)\}$$

$$\mathrm{Proc}_{\mathtt{UR}}((\mathcal{DT}((5.3)), \mathcal{R}_{div})) = \{(\{\mathcal{DT}((5.3))\}, \varnothing)\}$$

$$\mathrm{Proc}_{\mathtt{UR}}((\mathcal{DT}((5.4))', \mathcal{R}_{div})) = \{(\{\mathcal{DT}((5.4))'\}, \{(5.1), (5.2)\})\}$$

and remain with four smaller DP problems that all contain fewer rewrite rules.

Reduction Pair Processor

The last processor we want to adapt is the reduction pair processor. Since we are using a rewrite relation that operates on sets, we have to lift polynomial interpretations from terms to our positional dependency tuple sets.

Definition 5.4.25 (Polynomial Interpretations for PDTS)**.** Let Pol be a polynomial interpretation. We lift Pol from terms $\mathcal{T}(\Sigma, \mathcal{V})$ to PDTS by defining

$$Pol(A) = \sum_{(t,\pi)\in A} Pol(t)$$

for all $A \in \mathsf{PDTS}$. So in particular, $Pol(\varnothing) = \sum_{(t,\pi)\in\varnothing} Pol(t) = 0$. Abusing the notation, we write Pol for both the Σ-algebra and its lifting to sets in PDTS.

This lifting to sets already shows the first downside compared to the reduction pair processor in the non-probabilistic setting. We have to take the sum of the values of every defined symbol on the right-hand side of a rule into account, as we are working with dependency tuples and not just pairs anymore. We have seen that both the defined symbols and their number of occurrences on the right-hand side matter. Hence, we cannot regard them separately anymore. The second disadvantage is that we have to further restrict the polynomial interpretations. We have to use *multilinear* polynomial interpretation as they are concave, and we need this concavity to "exchange" the expected value and the value of the polynomial interpretation f_{Pol} for every symbol $f \in \Sigma$. The idea for this comes from [2], where they used this idea to prove PAST automatically for a given PTRS.

We first prove the main technical lemma regarding the reduction pair processor and then prove the real processor itself afterward.

Lemma 5.4.26 (Proving Innermost AST on CTs with Reduction Pair Processor)**.** *Let $(\mathcal{P}, \mathcal{S})$ be a DP problem, and let $Pol : \mathcal{T}\left(\Sigma \uplus \Sigma^{\#}, \mathcal{V}\right) \to \mathbb{N}[\mathcal{V}]$ be a natural, multilinear polynomial interpretation which is weakly monotonic (i.e., $x \geq y$ implies $f_{Pol}(\ldots, x, \ldots) \geq f_{Pol}(\ldots, y, \ldots)$ for all $f \in \Sigma \cup \Sigma^{\#}$). Suppose that we have $\mathcal{P} = \mathcal{P}_{\succcurlyeq} \uplus \mathcal{P}_{\succ}$ and the following conditions hold.*

(1) *For every $(\ell^{\#}, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in \mathcal{P}_{\succ}$, there exists a $1 \leq j \leq k$ with*

$$Pol(\ell^{\#}) > Pol(C_j)$$

If $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{S}$, then we additionally require that

$$Pol(\ell) \geq Pol(r_j)$$

(2) *For every $(\ell^{\#}, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in \mathcal{P}$, we have*

$$Pol(\ell^{\#}) \geq \sum_{1 \leq j \leq k} p_j \cdot Pol(C_j)$$

(3) *For every $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{S}$ we have*

$$Pol(\ell) \geq \sum_{1 \leq j \leq k} p_j \cdot Pol(r_j)$$

Let $\mathfrak{T} = (V, E, L, P)$ be a $(\mathcal{P}, \mathcal{S})$-computation tree. We can partition $P = P_{\succcurlyeq} \uplus P_{\succ}$ according to $\mathcal{P} = \mathcal{P}_{\succcurlyeq} \uplus \mathcal{P}_{\succ}$. If $\mathfrak{T}$ satisfies

(+) *Every infinite path has an infinite number of $P_{\succ}$ nodes.*

then $|\mathfrak{T}|_{\text{Leaf}} = 1$.

Proof. Remember that the definition of our chains works with ground instances. This proof proceeds similar to the proof of Theorem 4.3.1 and thus is based on [27]. The difference to Theorem 4.3.1 is that this time we are talking about computation trees and we have two different rewrite relations that we need to deal with. All in all, the core steps of this proof are again the following:

1. We extend the conditions to rewrite steps instead of just rules (and thus, to edges of a computation tree)

2. We create a computation tree $\mathfrak{T}^{\leq N}$ for any $N \in \mathbb{N}$
3. We prove that $|\mathfrak{T}^{\leq N}|_{\text{Leaf}} \geq p^N$ for any $N \in \mathbb{N}$
4. We prove that $|\mathfrak{T}^{\leq N}|_{\text{Leaf}} = 1$ for any $N \in \mathbb{N}$
5. Finally, we prove that $|\mathfrak{T}|_{\text{Leaf}} = 1$

Part (1.) and (3.) are much more involved in this proof compared to Theorem 4.3.1 due to the more complex rewrite relation that we are dealing with. The other parts are nearly the same as before. We only have to adjust everything to computation trees instead of rewrite sequences. Here, $p > 0$ is the minimal probability that occurs in the rules of $\mathcal{P}$ and $\mathcal{S}$. As $\mathcal{P}$ and $\mathcal{S}$ both have only finitely many rules and all occurring multi-distributions have finite support, this minimum is well defined.

1. We extend the conditions to rewrite steps instead of just rules

We start off by showing that the conditions (1), (2), and (3) of the lemma extend to rewrite steps instead of just rules:

(a) If $s \in \mathcal{T}(\Sigma)$ and $\delta = \{p_1 : t_1, \ldots, p_k : t_k\} \in FDist(\mathcal{T}(\Sigma))$ with $s \xrightarrow{\text{i}}_{\mathcal{S}} \delta$ using the rule $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{S}$ with $Pol(\ell) \geq Pol(r_j)$ for some $1 \leq j \leq k$, then we have $Pol(s) \geq Pol(t_j)$.

(b) If $A \in \mathsf{PDTS}_{\mathsf{wp}}$ and $\delta = \{p_1 : B_1, \ldots, p_k : B_k\} \in FDist(\mathsf{PDTS}_{\mathsf{wp}})$ with $A \xrightarrow{\text{i}}_{\mathcal{P},\mathcal{S}} \delta$, then there exists a $1 \leq j \leq k$ with $Pol(A) > Pol(B_j)$.

(c) If $s \in \mathcal{T}(\Sigma)$ and $\delta = \{p_1 : t_1, \ldots, p_k : t_k\} \in FDist(\mathcal{T}(\Sigma))$ with $s \xrightarrow{\text{i}}_{\mathcal{S}} \delta$, then we have $Pol(s) \geq \sum_{1 \leq i \leq k} p_i \cdot Pol(t_i)$.

(d) If $A \in \mathsf{PDTS}_{\mathsf{wp}}$ and $\delta = \{p_1 : B_1, \ldots, p_k : B_k\} \in FDist(\mathsf{PDTS}_{\mathsf{wp}})$ with $A \xrightarrow{\text{i}}_{\mathcal{P},\mathcal{S}} \delta$, then $Pol(A) \geq \sum_{1 \leq i \leq k} p_i \cdot Pol(B_i)$.

(e) If $A \in \mathsf{PDTS}_{\mathsf{wp}}$ and $\delta = \{p_1 : B_1, \ldots, p_k : B_k\} \in FDist(\mathsf{PDTS}_{\mathsf{wp}})$ with $A \xrightarrow{\text{i}}_{\mathcal{S}} \delta$, then $Pol(A) \geq \sum_{1 \leq j \leq k} p_j \cdot Pol(B_j)$.

(a) In this case, there exist a rule $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{S}$ with $Pol(\ell) \geq Pol(r_j)$ for some $1 \leq j \leq k$, a ground substitution $\sigma \in \text{Sub}(\Sigma, \mathcal{V})$, and a position π of s such that $s|_\pi = \ell\sigma$, all proper subterms of $\ell\sigma$ are in normal form w.r.t. $\mathcal{S}$, and $t_h = s[r_h\sigma]_\pi$ for all $1 \leq h \leq k$.

We perform induction on π. So in the induction base, let $\pi = \varepsilon$. Hence, we have $s = \ell\sigma$ and $\delta = \{p_1 : r_1\sigma, \ldots, p_k : r_k\sigma\}$. By assumption we have $Pol(\ell) \geq Pol(r_j)$ for some $1 \leq j \leq k$. As these inequations hold for all instantiations of the occurring variables, for $t_j = r_j\sigma$ we have

$$Pol(s) = Pol(\ell\sigma) \geq Pol(r_j\sigma) = Pol(t_j).$$

In the induction step, we have $\pi = i.\pi'$, $s = f(s_1, \ldots, s_i, \ldots, s_n)$, $s_i \xrightarrow{\text{i}}_{\mathcal{S}} \{p_1 : t_{i,1}, \ldots, p_k : t_{i,k}\}$, and $t_h = f(s_1, \ldots, t_{i,h}, \ldots, s_n)$ for all $1 \leq h \leq k$. Then by the

induction hypothesis we have $Pol(s_i) \geq Pol(t_{i,j})$. For $t_j = f(s_1, \ldots, t_{i,j}, \ldots, s_n)$ we obtain

$$\begin{array}{rcl} Pol(s) & = & Pol(f(s_1, \ldots, s_i, \ldots, s_n)) \\ & = & f_{Pol}(Pol(s_1), \ldots, Pol(s_i), \ldots, Pol(s_n)) \\ & \geq & f_{Pol}(Pol(s_1), \ldots, Pol(t_{i,j}), \ldots, Pol(s_n)) \\ & & \qquad \text{(by weak monotonicity of } f_{Pol} \text{ and } Pol(s_i) \geq Pol(t_{i,j})) \\ & = & Pol(f(s_1, \ldots, t_{i,j}, \ldots, s_n)) \\ & = & Pol(t_j). \end{array}$$

(b) In this case, there exist a dependency tuple $(\ell^\#, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in \mathcal{P}$, a main rewrite pair $(t, \pi) \in A$, and a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$ such that $t = \ell^\#\sigma$ and all proper subterms of $\ell^\#\sigma$ are in normal form w.r.t. $\mathcal{S}$. Let us first assume that $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{S}$. Then we have $B_j = M_j^+ \cup \{\!\{(a[r_j\sigma]_{\chi_a}, \tau) \mid (a, \tau) \in M_{rew}\}\!\} \cup M_\perp$ as in Definition 5.1.12. By Requirement (1), there exists a $1 \leq j \leq k$ with $Pol(\ell^\#) > Pol(C_j)$ and $Pol(\ell) \geq Pol(r_j)$. As these inequations hold for all instantiations of the occurring variables, we have

$$\begin{array}{rcl} Pol(A) & = & \sum_{(q,\tau)\in A} Pol(q) \\ & = & \sum_{(q,\tau)\in\{(t,\pi)\}} Pol(q) + \sum_{(q,\tau)\in M_{rew}} Pol(q) + \sum_{(q,\tau)\in M_\perp} Pol(q) \\ & & \quad + \sum_{(q,\tau)\in M_<} Pol(q) \\ & \geq & \sum_{(q,\tau)\in\{(t,\pi)\}} Pol(q) + \sum_{(q,\tau)\in M_{rew}} Pol(q) + \sum_{(q,\tau)\in M_\perp} Pol(q) \\ & & \qquad \text{(removing } M_<\text{)} \\ & = & Pol(t) + \sum_{(q,\tau)\in M_{rew}} Pol(q) + \sum_{(q,\tau)\in M_\perp} Pol(q) \\ & = & Pol(\ell^\#\sigma) + \sum_{(q,\tau)\in M_{rew}} Pol(q) + \sum_{(q,\tau)\in M_\perp} Pol(q) \\ & & \qquad \text{(as } t = \ell^\#\sigma\text{)} \\ & > & Pol(C_j\sigma) + \sum_{(q,\tau)\in M_{rew}} Pol(q) + \sum_{(q,\tau)\in M_\perp} Pol(q) \\ & & \qquad \text{(as } Pol(\ell^\#) > Pol(C_j)\text{, and hence } Pol(\ell^\#\sigma) > Pol(C_j\sigma)\text{)} \\ & = & \sum_{q\in C_j\sigma} Pol(q) + \sum_{(q,\tau)\in M_{rew}} Pol(q) + \sum_{(q,\tau)\in M_\perp} Pol(q) \\ & = & \sum_{(q,\tau)\in M_j^+} Pol(q) + \sum_{(q,\tau)\in M_{rew}} Pol(q) + \sum_{(q,\tau)\in M_\perp} Pol(q) \\ & & \qquad \text{(by definition of } M_j^+\text{)} \\ & \geq & \sum_{(q,\tau)\in M_j^+} Pol(q) + \sum_{(q,\tau)\in M_{rew}} Pol(q[r_j\sigma]_\chi) + \sum_{(q,\tau)\in M_\perp} Pol(q) \\ & & \qquad \text{(by } Pol(\ell) \geq Pol(r_j) \text{ and (a))} \\ & = & \sum_{(q,\tau)\in B_j} Pol(q) \\ & = & Pol(B_j) \end{array}$$

If we have $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \notin \mathcal{S}$, then we have $B_j = M_j^+ \cup M_\perp$ and hence it is a subset of $B_j' := M_j^+ \cup \{\!\{(a[r_j\sigma]_{\chi_a}, \tau) \mid (a, \tau) \in M_{rew}\}\!\} \cup M_\perp$ (B_j in the previous case). Therefore, we get $Pol(A) > Pol(B_j') \geq Pol(B_j)$.

(c) In this case, there exist a rule $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{S}$ with $Pol(\ell) \geq \sum_{1\leq j\leq k} p_j \cdot Pol(r_j)$, a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and a position π of s such that $s|_\pi = \ell\sigma$, all proper subterms of $\ell\sigma$ are in normal form w.r.t. $\mathcal{S}$, and $t_h = s[r_h\sigma]_\pi$ for all $1 \leq h \leq k$.

We perform induction on π. So in the induction base $\pi = \varepsilon$ we have $s = \ell\sigma$ and $\delta = \{p_1 : r_1\sigma, \ldots, p_k : r_k\sigma\}$. As $Pol(\ell) \geq \sum_{1\leq j\leq k} p_j \cdot Pol(r_j)$ holds for all instantiations of the occurring variables, for $t_j = r_j\sigma$ we obtain

$$Pol(s) \;=\; Pol(\ell\sigma) \;\geq\; \sum_{1\leq j\leq k} p_j \cdot Pol(r_j\sigma) \;=\; \sum_{1\leq j\leq k} p_j \cdot Pol(t_j).$$

In the induction step, we have $\pi = i.\pi'$, $s = f(s_1, \ldots, s_i, \ldots, s_n)$, $s_i \xrightarrow{\mathsf{i}}_{\mathcal{S}} \{p_1 : t_{i,1}, \ldots, p_k : t_{i,k}\}$, and $t_h = f(s_1, \ldots, t_{i,h}, \ldots, s_n)$ for all $1 \leq h \leq k$. Then by the induction hypothesis we have $Pol(s_i) \geq \sum_{1 \leq j \leq k} p_j \cdot Pol(t_{i,j})$. Thus, we have

$$\begin{array}{rcl}
Pol(s) & = & Pol(f(s_1, \ldots, s_i, \ldots, s_n)) \\
& = & f_{Pol}(Pol(s_1), \ldots, Pol(s_i), \ldots, Pol(s_n)) \\
& \geq & f_{Pol}(Pol(s_1), \ldots, \sum_{1 \leq j \leq k} p_j \cdot Pol(t_{i,j}), \ldots, Pol(s_n)) \\
& & \quad \text{(by weak monotonicity of } f_{Pol} \text{ and } Pol(s_i) \geq \sum_{1 \leq j \leq k} p_j \cdot Pol(t_{i,j})) \\
& \geq & \sum_{1 \leq j \leq k} p_i \cdot f_{Pol}(Pol(s_1), \ldots, Pol(t_{i,j}), \ldots, Pol(s_n)), \\
& & \quad \text{(as } f_{Pol} \text{ is componentwise concave due to multilinearity)} \\
& = & \sum_{1 \leq j \leq k} p_j \cdot Pol(f(s_1, \ldots, t_{i,j}, \ldots, s_n))
\end{array}$$

(d) In this case, there exist a dependency tuple $(\ell^\#, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in \mathcal{P}$, a main rewrite pair $(t, \pi) \in A$, and a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, such that $t = \ell^\#\sigma$ and all proper subterms of $\ell^\#\sigma$ are in normal form w.r.t. $\mathcal{S}$. Let us first assume that $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{S}$. Then we have $B_j = M_j^+ \cup \{\!\{(a[r_j\sigma]_{\chi_a}, \tau) \mid (a, \tau) \in M_{rew}\}\!\} \cup M_\perp$ as in Definition 5.1.12. By Requirement (2), we have $Pol(\ell^\#) \geq \sum_{1 \leq j \leq k} p_j \cdot Pol(C_j)$ and by (3) we have $Pol(\ell) \geq \sum_{1 \leq j \leq k} p_j \cdot Pol(r_j)$. As these inequations hold for all instantiations of the occurring variables, we have

$$\begin{array}{rcl}
Pol(A) & = & \sum_{(q,\tau) \in A} Pol(q) \\
& = & \sum_{(q,\tau) \in \{(t,\pi)\}} Pol(q) + \sum_{(q,\tau) \in M_{rew}} Pol(q) + \sum_{(q,\tau) \in M_\perp} Pol(q) \\
& & + \sum_{(q,\tau) \in M_<} Pol(q) \\
& \geq & \sum_{(q,\tau) \in \{(t,\pi)\}} Pol(q) + \sum_{(q,\tau) \in M_{rew}} Pol(q) + \sum_{(q,\tau) \in M_\perp} Pol(q) \\
& & \quad \text{(removing } M_< \text{)} \\
& = & Pol(t) + \sum_{(q,\tau) \in M_{rew}} Pol(q) + \sum_{(q,\tau) \in M_\perp} Pol(q) \\
& = & Pol(\ell^\#\sigma) + \sum_{(q,\tau) \in M_{rew}} Pol(q) + \sum_{(q,\tau) \in M_\perp} Pol(q) \\
& & \quad \text{(as } t = \ell^\#\sigma \text{)} \\
& \geq & \sum_{1 \leq j \leq k} p_j \cdot Pol(C_j\sigma) + \sum_{(q,\tau) \in M_{rew}} Pol(q) + \sum_{(q,\tau) \in M_\perp} Pol(q) \\
& & \quad \text{(by } Pol(\ell^\#) \geq \sum_{1 \leq j \leq k} p_j \cdot Pol(C_j), \text{)} \\
& & \qquad \text{and hence } Pol(\ell^\#\sigma) \geq \sum_{1 \leq j \leq k} p_j \cdot Pol(C_j\sigma)) \\
& = & \sum_{1 \leq j \leq k} p_j \cdot \left(\sum_{q \in C_j\sigma} Pol(q)\right) + \sum_{(q,\tau) \in M_{rew}} Pol(q) + \sum_{(q,\tau) \in M_\perp} Pol(q) \\
& = & \sum_{1 \leq j \leq k} p_j \cdot \left(\sum_{(q,\tau) \in M_j^+} Pol(q)\right) + \sum_{(q,\tau) \in M_{rew}} Pol(q) + \sum_{(q,\tau) \in M_\perp} Pol(q) \\
& & \quad \text{(by definition of } M_j^+ \text{)} \\
& \geq & \sum_{1 \leq j \leq k} p_j \cdot \left(\sum_{(q,\tau) \in M_j^+} Pol(q)\right) + \sum_{(q,\tau) \in M_{rew}} \left(\sum_{1 \leq j \leq k} p_j \cdot Pol(q[r_j\sigma]_\chi)\right) \\
& & + \sum_{(q,\tau) \in M_\perp} Pol(q) \\
& & \quad \text{(by } Pol(\ell) \geq \sum_{1 \leq j \leq k} p_j \cdot Pol(r_j) \text{ and (c))} \\
& = & \sum_{1 \leq j \leq k} p_j \cdot \left(\sum_{(q,\tau) \in M_j^+} Pol(q)\right) + \sum_{1 \leq j \leq k} \left(\sum_{(q,\tau) \in M_{rew}} p_j \cdot Pol(q[r_j\sigma]_\chi)\right) \\
& & + \sum_{(q,\tau) \in M_\perp} Pol(q) \\
& = & \sum_{1 \leq j \leq k} p_j \cdot \left(\sum_{(q,\tau) \in M_j^+} Pol(q)\right) + \sum_{1 \leq j \leq k} p_j \cdot \left(\sum_{(q,\tau) \in M_{rew}} Pol(q[r_j\sigma]_\chi)\right) \\
& & + \sum_{(q,\tau) \in M_\perp} Pol(q) \\
& = & \sum_{1 \leq j \leq k} p_j \cdot \left(\sum_{(q,\tau) \in M_j^+} Pol(q) + \sum_{(q,\tau) \in M_{rew}} Pol(q[r_j\sigma]_\chi)\right) \\
& & + \sum_{(q,\tau) \in M_\perp} Pol(q)
\end{array}$$

$$
\begin{aligned}
&= \sum_{1\le j\le k} p_j \cdot \Big(\sum_{(q,\tau)\in M_j^+} Pol(q) + \sum_{(q,\tau)\in M_{rew}} Pol(q[r_j\sigma]_\chi)\Big) \\
&\qquad + \sum_{1\le j\le k} p_j \cdot \Big(\sum_{(q,\tau)\in M_\perp} Pol(q)\Big) \\
&\qquad\qquad \text{(as } \textstyle\sum_{1\le j\le k} p_j = 1\text{)} \\
&= \sum_{1\le j\le k} p_j \cdot \Big(\sum_{(q,\tau)\in M_j^+} Pol(q) + \sum_{(q,\tau)\in M_{rew}} Pol(q[r_j\sigma]_\chi) + \sum_{(q,\tau)\in M_\perp} Pol(q)\Big) \\
&= \sum_{1\le j\le k} p_j \cdot \Big(\sum_{(q,\tau)\in B_j} Pol(q)\Big) \\
&= \sum_{1\le j\le k} p_j \cdot Pol(B_j)
\end{aligned}
$$

If we have $\ell \to \{p_1 : r_1, \dots, p_k : r_k\} \notin \mathcal{S}$, then we have $B_j = M_j^+ \cup M_\perp$ and hence it is a subset of $B_j' := M_j^+ \cup \{\!\{(a[r_j\sigma]_{\chi_a}, \tau) \mid (a,\tau) \in M_{rew}\}\!\} \cup M_\perp$ (B_j in the previous case). Therefore, we get $Pol(A) \ge \sum_{1\le i\le k} p_i \cdot Pol(B_i') \ge \sum_{1\le i\le k} p_i \cdot Pol(B_i)$.

(e) In this case, there exist a rule $\ell \to \{p_1 : r_1, \dots, p_k : r_k\} \in \mathcal{S}$, a ground substitution $\sigma \in \mathrm{Sub}(\Sigma, \mathcal{V})$, and a position $\tau \in \mathbb{N}^+$ such that $t|_\tau = \ell\sigma$ and all proper subterms of $\ell\sigma$ are in normal form w.r.t. $\mathcal{S}$. Then $B_j = M_j^+ \cup \{\!\{(a[r_j\sigma]_{\chi_a.\tau}, \chi) \mid (a,\chi) \in M_{rew}\}\!\} \cup M_\perp$ as in Definition 5.1.17. Now, it follows that

$$
\begin{aligned}
Pol(A) &= \sum_{(q,\tau)\in A} Pol(q) \\
&= \sum_{(q,\tau)\in\{(t,\pi)\}} Pol(q) + \sum_{(q,\tau)\in M_{rew}} Pol(q) + \sum_{(q,\tau)\in M_\perp} Pol(q) \\
&\qquad + \sum_{(q,\tau)\in M_<} Pol(q) \\
&\ge \sum_{(q,\tau)\in\{(t,\pi)\}} Pol(q) + \sum_{(q,\tau)\in M_{rew}} Pol(q) + \sum_{(q,\tau)\in M_\perp} Pol(q) \\
&\qquad\qquad \text{(removing } M_<\text{)} \\
&= \sum_{(q,\tau)\in\{(t,\pi)\}\cup M_{rew}} Pol(q) + \sum_{(q,\tau)\in M_\perp} Pol(q) \\
&\ge \sum_{(q,\tau)\in\{(t,\pi)\}\cup M_{rew}} \Big(\sum_{1\le j\le k} p_j \cdot Pol(q[r_j\sigma]_{\chi.\tau})\Big) + \sum_{(q,\tau)\in M_\perp} Pol(q) \\
&\qquad\qquad \text{(by } Pol(\ell) \ge \textstyle\sum_{1\le j\le k} p_j \cdot Pol(r_j) \text{ and (c))} \\
&= \sum_{1\le j\le k} \Big(\sum_{(q,\tau)\in\{(t,\pi)\}\cup M_{rew}} p_j \cdot Pol(q[r_j\sigma]_{\chi.\tau})\Big) + \sum_{(q,\tau)\in M_\perp} Pol(q) \\
&= \sum_{1\le j\le k} p_j \cdot \Big(\sum_{(q,\tau)\in\{(t,\pi)\}\cup M_{rew}} Pol(q[r_j\sigma]_{\chi.\tau})\Big) + \sum_{(q,\tau)\in M_\perp} Pol(q) \\
&= \sum_{1\le j\le k} p_j \cdot \Big(\sum_{(q,\tau)\in\{(t,\pi)\}\cup M_{rew}} Pol(q[r_j\sigma]_{\chi.\tau})\Big) + \sum_{1\le j\le k} p_j \cdot \Big(\sum_{(q,\tau)\in M_\perp} Pol(q)\Big) \\
&\qquad\qquad \text{(as } \textstyle\sum_{1\le j\le k} p_j = 1\text{)} \\
&= \sum_{1\le j\le k} p_j \cdot \Big(\sum_{(q,\tau)\in\{(t,\pi)\}\cup M_{rew}} Pol(q[r_j\sigma]_{\chi.\tau}) + \sum_{(q,\tau)\in M_\perp} Pol(q)\Big) \\
&= \sum_{1\le j\le k} p_j \cdot \Big(\sum_{(q,\tau)\in B_j} Pol(q)\Big) \\
&= \sum_{1\le j\le k} p_j \cdot Pol(B_j)
\end{aligned}
$$

2. We create a computation tree $\mathfrak{T}^{\le N}$ for any $N \in \mathbb{N}$

Let $\mathfrak{T} = (V, E, L, P)$ be a $(\mathcal{P}, \mathcal{S})$-computation tree that satisfies (+). Due to our restriction to ground terms, we can define the *value* of a node $x \in V$ in our computation tree.

$$
\mathcal{V}al : V \to \mathbb{N}, x \mapsto \begin{cases} 0, & \text{if } x \in \mathrm{Leaf} \\ Pol(A_x) + 1, & \text{otherwise} \end{cases}
$$

For any $N \in \mathbb{N}$, we create a modified tree $\mathfrak{T}^{\le N}$, where we cut everything below a node x of the tree with $\mathcal{V}al(x) > N$. Let $C := \mathrm{Leaf}^{\mathfrak{T}^{\le N}} \setminus \mathrm{Leaf}^{\mathfrak{T}}$ be the set of all new leaves in $\mathfrak{T}^{\le N}$ due to the cut. So for all $x \in C$ we have $\mathcal{V}al(x) > N$.

Our goal is to prove that we have $|\mathfrak{T}|_{\mathrm{Leaf}} = 1$. First of all, we prove that $|\mathfrak{T}^{\le N}|_{\mathrm{Leaf}} = 1$ for any $N \in \mathbb{N}$. However, this does not yet prove that $|\mathfrak{T}|_{\mathrm{Leaf}} = 1$, which we will show in a second step.

3. We prove that $|\mathfrak{T}^{\leq N}|_{\text{Leaf}} \geq p^N$ for any $N \in \mathbb{N}$

This part of the proof drastically changes from the proof of Theorem 4.3.1, due to the fact that in a rewrite step $A \xrightarrow{\text{i}}_{\mathcal{S}} \{p_1 : B_1, \ldots, p_k : B_k\}$ (or a rewrite step $A \xrightarrow{\text{i}}_{\mathcal{P},\mathcal{S}} \{p_1 : B_1, \ldots, p_k : B_k\}$ with a dependency tuple from $\mathcal{P}_{\succcurlyeq}$), we cannot guarantee that there exists an $1 \leq j \leq k$ with $Pol(A) > Pol(B_j)$. Here, it is also possible to have $Pol(A) = Pol(B_j)$ for all $1 \leq j \leq k$. Hence, there does not have to be a single witness path of length N in $\mathfrak{T}^{\leq N}$ that shows termination with a chance of at least p^N. Instead, we have to use multiple witness paths of a finite length to ensure that we have $|\mathfrak{T}^{\leq N}|_{\text{Leaf}} \geq p^N$. We now first explain how to find such a set of witness paths in a finite induced sub $(\mathcal{P}, \mathcal{S})$-computation tree that starts with a node from $\mathcal{P}_{\succ}$ and then we prove by induction that we have $|\mathfrak{T}^{\leq N}|_{\text{Leaf}} \geq p^N$ in a second step.

3.1. We find witnesses in induced sub chain trees of $\mathfrak{T}^{\leq N}$

In this part, we prove a first observation regarding the existence of certain witnesses that shows a guaranteed value decrease. For every $x \in P_{\succ}$, let T_x be the induced sub chain tree that starts at x and where we cut every edge after the second node from $P_{\succ}$. Since $\mathfrak{T}$ satisfies $(+)$ and is finitely branching, we know that T_x must be finite. We want to prove that for such a tree T_x, we have a set of leaves (a set of witnesses) that show a certain value decrease compared to the root value $\mathcal{V}al(x)$. To be precise, we want to prove that there exists a set $W^x \subseteq \text{Leaf}^{T_x}$ of leaves in T_x with the following two properties:

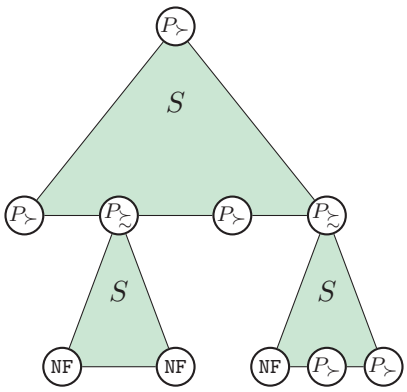


Figure 5.20: T_x

(W-1) For all $w \in W^x$ we have $\overline{\mathcal{V}al}(w) < \overline{\mathcal{V}al}(x)$

(W-2) $\sum_{w \in W^x} p_w^{T_x} \cdot p^{\overline{\mathcal{V}al}(w)} \geq p^{\overline{\mathcal{V}al}(x)}$

Here, we use an adjusted value function $\overline{\mathcal{V}al}$, so that for every node $x \in V^{\mathfrak{T}^{\leq N}}$ we have

$$\overline{\mathcal{V}al}(x) = \begin{cases} 0 & \text{, if } x \text{ is a leaf in } \mathfrak{T}^{\leq N}, \\ \mathcal{V}al(A_x) & \text{, otherwise.} \end{cases}$$

This is the same value function as before, except for the nodes in C. For all $x \in C$ we have $\mathcal{V}al(x) > N$ and $\overline{\mathcal{V}al}(x) = 0$. The first property (W-1) says that all of our witnesses have a strictly smaller value than the root. Furthermore, we have to be careful that the probabilities for our witnesses are not too low. The second property (W-2) says that the sum of all probabilities for the witnesses is still big enough. The additional $p^{\overline{\mathcal{V}al}(w)}$ is used to allow smaller probabilities for our witnesses if the value decrease is high enough.

In order to show the existence of such a set W^x, we prove by induction on the height H of T_x that there exists a set of nodes W_i^x such that for all $1 \leq i \leq H$ we have

(W-1!) For all $w \in W_i^x$ we have $\overline{\mathcal{V}al}(w) < \overline{\mathcal{V}al}(x)$

(W-2!) $\sum_{w \in W_i^x} p_w^{T_x} \cdot p^{\overline{\mathcal{V}al}(w)} \geq p^{\overline{\mathcal{V}al}(x)}$

(W-3!) Every $w \in W_i^x$ is a leaf before the i-th depth or at depth i of the tree T_x

Then in the end if $i = H$ is the height of the tree T_x, we get a set $W^x := W^x_H$ such that every node in W^x_H is a leaf in T_x (i.e., $W^x_H \subseteq \text{Leaf}^{T_x}$) and both (W-1) and (W-2) are satisfied.

In the induction base, we consider depth $i = 0$ and look at the rewrite step at the root. The first edge represents a rewrite step with $\mathcal{P}_\succ$. Let $xE = \{y_1, \dots, y_k\}$ be the set of the successors of x. We have $A_x \xrightarrow{\text{i}}_{\mathcal{P},\mathcal{S}} \{p_1 : A_{y_1}, \dots, p_k : A_{y_k}\}$ using a rule from $\mathcal{P}_\succ$.

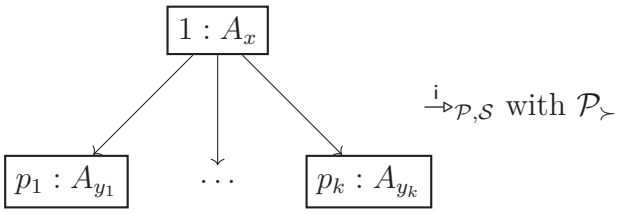


Figure 5.21: Nodes in the induction base

Due to (b) there is a $1 \leq j \leq k$ with $Pol(A_x) > Pol(A_{y_j})$. This means $Pol(A_x) - 1 \geq Pol(A_{y_j})$, and hence $\mathcal{V}al(A_x) - 1 \geq \mathcal{V}al(A_{y_j})$, which also implies that $\overline{\mathcal{V}al}(x) - 1 \geq \overline{\mathcal{V}al}(y_j)$. Since $0 < p \leq 1$, we therefore have $p^{\overline{\mathcal{V}al}(x)-1} \leq p^{\overline{\mathcal{V}al}(y_j)}$. Thus we can set $W^x_0 = \{y_j\}$ and have (W-1!) satisfied, since $\overline{\mathcal{V}al}(x) - 1 \geq \overline{\mathcal{V}al}(y_j)$ so $\overline{\mathcal{V}al}(y_j) < \overline{\mathcal{V}al}(x)$. Property (W-3!) is clearly satisfied and (W-2!) holds as well since we have

$$\sum_{w \in W^x_0} p^{T_x}_w \cdot p^{\overline{\mathcal{V}al}(w)} = p_j \cdot p^{\overline{\mathcal{V}al}(y_j)} \overset{p_j \geq p}{\geq} p \cdot p^{\overline{\mathcal{V}al}(y_j)} \geq p \cdot p^{\overline{\mathcal{V}al}(x)-1} = p^{\overline{\mathcal{V}al}(x)}$$

In the induction step, we consider depth $i > 0$. Due to the induction hypothesis, there is a set W^x_{i-1} that satisfies (W-1!), (W-2!), and (W-3!). For every node $w \in W^x_{i-1}$ that is not a leaf, let $wE = \{y^w_1, \dots, y^w_k\}$ be the set of its successors. We rewrite A_w either with $\xrightarrow{\text{i}}_{\mathcal{P},\mathcal{S}}$ and a rule from $\mathcal{P}_\succsim$ or with $\xrightarrow{\text{i}}_{\mathcal{S}}$ and a rule from $\mathcal{S}$ which rewrites A_w to a multi-distribution $\{p^w_1 : A_{y^w_1}, \dots, p^w_k : A_{y^w_k}\}$.

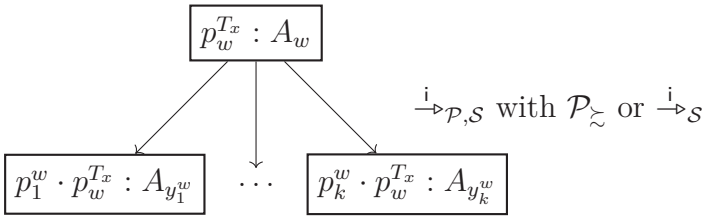


Figure 5.22: Induction Step

Due to (d) and (e), we have $Pol(A_w) \geq \sum_{1 \leq j \leq k} p_j \cdot Pol(A_{y^w_j})$. Hence, we either have $Pol(A_w) = Pol(A_{y^w_j})$ for all $1 \leq j \leq k$ or there exists at least one $1 \leq j \leq k$ with $Pol(A_w) > Pol(A_{y^w_j})$. We partition the set W^x_{i-1} into the disjoint subsets $W^{x(1)}_{i-1}$, $W^{x(2)}_{i-1}$, and $W^{x(3)}_{i-1}$, where

- $w \in W^{x(1)}_{i-1} :\Leftrightarrow w \in W^x_{i-1}$ and w a leaf.
- $w \in W^{x(2)}_{i-1} :\Leftrightarrow w \in W^x_{i-1}$ and $Pol(A_w) = Pol(A_{y^w_j})$ for all $1 \leq j \leq k$.
- $w \in W^{x(3)}_{i-1} :\Leftrightarrow w \in W^x_{i-1}$ and there exists a $1 \leq j \leq k$ with $Pol(A_w) > Pol(A_{y^w_j})$. We denote this node y^w_j by w^+.

In the first case, we already have $\overline{\mathcal{V}al}(w) = 0$. In the second case, we get $\overline{\mathcal{V}al}(w) = \overline{\mathcal{V}al}(y_j^w)$ for all $1 \leq j \leq k$. And in the third case, we have $Pol(w) > Pol(w^+)$. So for all of these nodes, we can be sure that the value does not increase. We can now define W_i^x as:

$$\begin{aligned} W_i^x &= W_{i-1}^{x(1)} \\ &\cup \bigcup_{w \in W_{i-1}^{x(2)}} \{y_1^w, \ldots, y_k^w\} \\ &\cup \{w^+ \mid w \in W_{i-1}^{x(3)}\} \end{aligned}$$

Intuitively, this means that every leaf remains inside of the set of witnesses ($W_{i-1}^{x(1)}$) and for every inner node w we have two cases. If there exists a successor w^+ with a strictly smaller value, then we exchange the node w with this successor w^+ in our set of witnesses ($\{w^+ \mid w \in W_{i-1}^{x(3)}\}$). Otherwise, all of the successors $y_1^w, \ldots, y_k^w$ of the node w have the same value that is also equal to the value of the node itself, so that we have to replace w with all of the successors in our set of witnesses as there is no single node with a guaranteed value decrease ($\bigcup_{w \in W_{i-1}^{x(2)}} \{y_1^w, \ldots, y_k^w\}$).

It remains to show that our induction hypothesis is still satisfied for W_i^x. In order to see that (W-1!) is satisfied, note that we have $\overline{\mathcal{V}al}(w) < \overline{\mathcal{V}al}(x)$, for all $w \in W_{i-1}^x$ by our induction hypothesis. For all $w' \in W_{i-1}^{x(1)}$, we have $\overline{\mathcal{V}al}(w) = 0$ and thus $\overline{\mathcal{V}al}(w') = 0 < \overline{\mathcal{V}al}(x)$. For all $w' \in W_{i-1}^{x(2)}$, we have $\overline{\mathcal{V}al}(w') = \overline{\mathcal{V}al}(w)$ for some $w \in W_{i-1}^x$ and thus $\overline{\mathcal{V}al}(w') = \overline{\mathcal{V}al}(w) < \overline{\mathcal{V}al}(x)$. Lastly, for all $w' \in W_{i-1}^{x(3)}$, we have $\overline{\mathcal{V}al}(w') < \overline{\mathcal{V}al}(w)$ for some $w \in W_{i-1}^x$ and thus $\overline{\mathcal{V}al}(w') < \overline{\mathcal{V}al}(w) < \overline{\mathcal{V}al}(x)$.

(W-3!) holds as well, since every node $w \in W_{i-1}$ that is not a leaf, is at depth $i-1$ of the tree T_x by our induction hypothesis. We exchange each such node with one or all of its successors. The leaves in W_{i-1} are at a depth of at most i by induction hypothesis and remain in W_i. Hence, all of the nodes of W_i are at a depth of at most i, and the nodes that are no leaves are at a depth of precisely i.

Finally, we regard (W-2!). For $\bigcup_{w \in W_{i-1}^{x(2)}} \{y_1^w, \ldots, y_k^w\}$ we have:

$$\begin{aligned} & \sum_{w' \in \bigcup_{w \in W_{i-1}^{x(2)}} \{y_1^w, \ldots, y_k^w\}} p_{w'}^{T_x} \cdot p^{\overline{\mathcal{V}al}(w')} \\ = \; & \sum_{w \in W_{i-1}^{x(2)}} \sum_{w' \in \{y_1^w, \ldots, y_k^w\}} p_{w'}^{T_x} \cdot p^{\overline{\mathcal{V}al}(w')} \\ = \; & \sum_{w \in W_{i-1}^{x(2)}} \sum_{1 \leq j \leq k} p_{y_j^w}^{T_x} \cdot p^{\overline{\mathcal{V}al}(y_j^w)} \\ = \; & \sum_{w \in W_{i-1}^{x(2)}} \sum_{1 \leq j \leq k} p_{y_j^w}^{T_x} \cdot p^{\overline{\mathcal{V}al}(w)} && (\text{as } \overline{\mathcal{V}al}(y_j^w) = \overline{\mathcal{V}al}(w) \text{ for all } 1 \leq j \leq k) \\ = \; & \sum_{w \in W_{i-1}^{x(2)}} \sum_{1 \leq j \leq k} p_j^w \cdot p_w^{T_x} \cdot p^{\overline{\mathcal{V}al}(w)} && (\text{as } p_{y_j^w}^{T_x} = p_j^w \cdot p_w^{T_x} \text{ for all } 1 \leq j \leq k) \\ = \; & \sum_{w \in W_{i-1}^{x(2)}} p_w^{T_x} \cdot p^{\overline{\mathcal{V}al}(w)} \cdot \sum_{1 \leq j \leq k} p_j^w \\ = \; & \sum_{w \in W_{i-1}^{x(2)}} p_w^{T_x} \cdot p^{\overline{\mathcal{V}al}(w)} \cdot 1 && (\text{as } \textstyle\sum_{1 \leq j \leq k} p_j^w = 1) \\ = \; & \sum_{w \in W_{i-1}^{x(2)}} p_w^{T_x} \cdot p^{\overline{\mathcal{V}al}(w)} \end{aligned}$$

For $\{w^+ \mid w \in W_{i-1}^{x(3)}\}$ we have:

$$\begin{aligned} & \sum_{w' \in \{w^+ \mid w \in W_{i-1}^{x(3)}\}} p_{w'}^{T_x} \cdot p^{\overline{\mathcal{V}al}(w')} \\ = \; & \sum_{w \in W_{i-1}^{x(3)}} p_{w^+}^{T_x} \cdot p^{\overline{\mathcal{V}al}(w^+)} \\ \geq \; & \sum_{w \in W_{i-1}^{x(3)}} p \cdot p_w^{T_x} \cdot p^{\overline{\mathcal{V}al}(w^+)} && (\text{as } p_{w^+}^{T_x} \geq p \cdot p_w^{T_x}) \\ \geq \; & \sum_{w \in W_{i-1}^{x(3)}} p \cdot p_w^{T_x} \cdot p^{\overline{\mathcal{V}al}(w)-1} && (\text{as } \overline{\mathcal{V}al}(w^+) \leq \overline{\mathcal{V}al}(w) + 1) \\ = \; & \sum_{w \in W_{i-1}^{x(3)}} p_w^{T_x} \cdot p^{\overline{\mathcal{V}al}(w)} \end{aligned}$$

All in all, we have

$$\begin{array}{lll} & \sum_{w' \in W_i^x} p_{w'}^{T_x} \cdot p^{\overline{\mathcal{V}al}(w')} & \\ = & \sum_{w' \in W_{i-1}^{x(1)}} p_{w'}^{T_x} \cdot p^{\overline{\mathcal{V}al}(w')} & \\ & + \sum_{w' \in \bigcup_{w \in W_{i-1}^{x(2)}} \{y_1^w, \ldots, y_k^w\}} p_{w'}^{T_x} \cdot p^{\overline{\mathcal{V}al}(w')} & \\ & + \sum_{w' \in \{w^+ \mid w \in W_{i-1}^{x(3)}\}} p_{w'}^{T_x} \cdot p^{\overline{\mathcal{V}al}(w')} & \\ \geq & \sum_{w \in W_{i-1}^{x(1)}} p_w^{T_x} \cdot p^{\overline{\mathcal{V}al}(w)} & \\ & + \sum_{w \in W_{i-1}^{x(2)}} p_w^{T_x} \cdot p^{\overline{\mathcal{V}al}(w)} & \\ & + \sum_{w \in W_{i-1}^{x(3)}} p_w^{T_x} \cdot p^{\overline{\mathcal{V}al}(w)} & \\ = & \sum_{w \in W_{i-1}^{x}} p_w^{T_x} \cdot p^{\overline{\mathcal{V}al}(w)} & \\ \overset{IH}{\geq} & p^{\overline{\mathcal{V}al}(x)} & \end{array}$$

3.2. We prove that $|\mathfrak{T}^{\leq N}|_{\text{Leaf}} \geq p^N$ for any $N \in \mathbb{N}$ by induction

We now want to prove that $|\mathfrak{T}^{\leq N}|_{\text{Leaf}} \geq p^N$ holds for any $N \in \mathbb{N}$. Let Z_k denote the set of nodes in $\mathfrak{T}^{\leq N}$ from $P_\succ$ that have precisely $k-1$ nodes from $P_\succ$ above them (so they are themselves the k-th node from $P_\succ$). Let Leaf_k denote the set of all leaves in $\mathfrak{T}^{\leq N}$ that are reachable through a path that uses less than k nodes from $P_\succ$. We show by induction that for all $k \in [1, N+1]$, we have

$$\sum_{x \in Z_k \cup \text{Leaf}_k \wedge 0 \leq \overline{\mathcal{V}al}(x) \leq N+1-k} p_x^{\mathfrak{T}^{\leq N}} \cdot p^{\overline{\mathcal{V}al}(x)} \geq p^N$$

Then, for $k = N+1$ we finally have:

$$\begin{array}{lll} p^N & \leq & \sum_{x \in Z_{N+1} \cup \text{Leaf}_{N+1} \wedge 0 \leq \overline{\mathcal{V}al}(x) \leq N+1-(N+1)} p_x^{\mathfrak{T}^{\leq N}} \cdot p^{\overline{\mathcal{V}al}(x)} \\ & = & \sum_{x \in Z_{N+1} \cup \text{Leaf}_{N+1} \wedge 0 \leq \overline{\mathcal{V}al}(x) \leq 0} p_x^{\mathfrak{T}^{\leq N}} \cdot p^{\overline{\mathcal{V}al}(x)} \\ & = & \sum_{x \in Z_{N+1} \cup \text{Leaf}_{N+1} \wedge \overline{\mathcal{V}al}(x) = 0} p_x^{\mathfrak{T}^{\leq N}} \cdot p^{\overline{\mathcal{V}al}(x)} \\ & = & \sum_{x \in Z_{N+1} \cup \text{Leaf}_{N+1} \wedge \overline{\mathcal{V}al}(x) = 0} p_x^{\mathfrak{T}^{\leq N}} \cdot 1 \\ & = & \sum_{x \in Z_{N+1} \cup \text{Leaf}_{N+1} \wedge \overline{\mathcal{V}al}(x) = 0} p_x^{\mathfrak{T}^{\leq N}} \\ & = & \sum_{x \in \text{Leaf}_{N+1}} p_x^{\mathfrak{T}^{\leq N}} \\ & \leq & \sum_{x \in \text{Leaf}^{\mathfrak{T}^{\leq N}}} p_x^{\mathfrak{T}^{\leq N}} \\ & = & |\mathfrak{T}^{\leq N}|_{\text{Leaf}} \end{array}$$

In the induction base, we have $k = 1$, and thus

$$\begin{array}{lll} & \sum_{x \in Z_1 \cup \text{Leaf}_1 \wedge 0 \leq \overline{\mathcal{V}al}(x) \leq N+1-1} p_x^{\mathfrak{T}^{\leq N}} \cdot p^{\overline{\mathcal{V}al}(x)} & \\ = & \sum_{x \in Z_1 \cup \text{Leaf}_1 \wedge 0 \leq \overline{\mathcal{V}al}(x) \leq N} p_x^{\mathfrak{T}^{\leq N}} \cdot p^{\overline{\mathcal{V}al}(x)} & \\ \geq & \sum_{x \in Z_1 \cup \text{Leaf}_1 \wedge 0 \leq \overline{\mathcal{V}al}(x) \leq N} p_x^{\mathfrak{T}^{\leq N}} \cdot p^N & (\text{since } \overline{\mathcal{V}al}(x) \leq N) \\ = & p^N \cdot \sum_{x \in Z_1 \cup \text{Leaf}_1 \wedge 0 \leq \overline{\mathcal{V}al}(x) \leq N} p_x^{\mathfrak{T}^{\leq N}} & \\ = & p^N \cdot \sum_{x \in Z_1 \cup \text{Leaf}_1} p_x^{\mathfrak{T}^{\leq N}} & \\ = & p^N \cdot 1 & (\text{since } \sum_{x \in Z_1 \cup \text{Leaf}_1} p_x^{\mathfrak{T}^{\leq N}} = 1) \\ = & p^N & \end{array}$$

Here, we have $\sum_{x \in Z_1 \cup \text{Leaf}_1} p_x^{\mathfrak{T}^{\leq N}} = 1$, since $Z_1 \cup \text{Leaf}_1$ are the leaves of the finite, grounded, induced sub chain tree, were we cut everything below the first node of $\mathcal{P}_\succ$ (i.e., we cute directly after the nodes in Z_1).

In the induction step, we assume that the statement is true for some $k \in [1, N]$. Then we have

$$\begin{aligned}
p^N &\overset{IH}{\leq} \textstyle\sum_{x \in Z_k \cup \mathrm{Leaf}_k \wedge 0 \leq \overline{\mathcal{V}al}(x) \leq N+1-k} p_x^{\mathfrak{T}^{\leq N}} \cdot p^{\overline{\mathcal{V}al}(x)} \\
&= \textstyle\sum_{x \in \mathrm{Leaf}_k} p_x \cdot p^{\overline{\mathcal{V}al}(x)} \; + \; \sum_{x \in Z_k \wedge 1 \leq \overline{\mathcal{V}al}(x) \leq N+1-k} p_x^{\mathfrak{T}^{\leq N}} \cdot p^{\overline{\mathcal{V}al}(x)} \\
&\leq \textstyle\sum_{x \in \mathrm{Leaf}_k} p_x \cdot p^{\overline{\mathcal{V}al}(x)} \; + \; \sum_{x \in Z_k \wedge 1 \leq \overline{\mathcal{V}al}(x) \leq N+1-k} p_x^{\mathfrak{T}^{\leq N}} \cdot \textstyle\sum_{w \in W^x} p_w^{T_x} \cdot p^{\overline{\mathcal{V}al}(w)} \\
&\qquad \text{(existence of the set } W^x \text{ by the previous part and (W-2))} \\
&= \textstyle\sum_{x \in \mathrm{Leaf}_k} p_x \cdot p^{\overline{\mathcal{V}al}(x)} \; + \; \sum_{x \in Z_k \wedge 1 \leq \overline{\mathcal{V}al}(x) \leq N+1-k} \textstyle\sum_{w \in W^x} p_x^{\mathfrak{T}^{\leq N}} \cdot p_w^{T_x} \cdot p^{\overline{\mathcal{V}al}(w)} \\
&= \textstyle\sum_{x \in \mathrm{Leaf}_k} p_x \cdot p^{\overline{\mathcal{V}al}(x)} \; + \; \sum_{x \in Z_k \wedge 1 \leq \overline{\mathcal{V}al}(x) \leq N+1-k} \textstyle\sum_{w \in W^x} p_w^{\mathfrak{T}^{\leq N}} \cdot p^{\overline{\mathcal{V}al}(w)} \\
&\qquad \text{(as } p_x^{\mathfrak{T}^{\leq N}} \cdot p_w^{T_x} = p_w^{\mathfrak{T}^{\leq N}}\text{)}
\end{aligned}$$

Now, every node inside of W^x is either contained in Leaf_{k+1} or contained in Z_{k+1}. The reason for that is that W^x only contains leaves from T_x and a leaf in T_x is either also a leaf in $\mathfrak{T}^{\leq N}$ so that they are contained in Leaf_{k+1}, or contained in $\mathcal{P}_\succ$ and thus in Z_{k+1}, since x is contained in Z_k and there is no other inner node from $\mathcal{P}_\succ$ in T_x. Furthermore, we know that $\overline{\mathcal{V}al}(w) < \overline{\mathcal{V}al}(x)$ for all $w \in W^x$ by (W-1). Thus we get

$$\begin{aligned}
&\textstyle\sum_{x \in \mathrm{Leaf}_k} p_x \cdot p^{\overline{\mathcal{V}al}(x)} \; + \; \sum_{x \in Z_k \wedge 1 \leq \overline{\mathcal{V}al}(x) \leq N+1-k} \textstyle\sum_{w \in W^x} p_w^{\mathfrak{T}^{\leq N}} \cdot p^{\overline{\mathcal{V}al}(w)} \\
\leq \; &\textstyle\sum_{w \in Z_{k+1} \cup \mathrm{Leaf}_{k+1} \wedge 0 \leq \overline{\mathcal{V}al}(w) \leq N+1-(k+1)} p_w^{\mathfrak{T}^{\leq N}} \cdot p^{\overline{\mathcal{V}al}(w)}
\end{aligned}$$

Now, we have shown that $|\mathfrak{T}^{\leq N}|_{\mathrm{Leaf}} \geq p^N$.

4. We prove that $|\mathfrak{T}^{\leq N}|_{\mathrm{Leaf}} = 1$ for any $N \in \mathbb{N}$

This part is completely analogous to the fourth part of the proof of Theorem 4.3.1. We only need to speak in terms of computation trees instead of rewrite sequences. We have proven that $|\mathfrak{T}^{\leq N}|_{\mathrm{Leaf}} \geq p^N$ holds for all $(\mathcal{P}, \mathcal{S})$-computation trees that satisfy (+). Hence, for any $N \in \mathbb{N}$, we have

$$p_N^\star := \inf_{\mathfrak{T} \text{ is a } (\mathcal{P},\mathcal{S})\text{-computation tree satisfying (+)}} (|\mathfrak{T}^{\leq N}|_{\mathrm{Leaf}}) \geq p^N > 0 \tag{5.33}$$

We now prove by contradiction that this is enough to ensure $p_N^\star = 1$. So assume that $p_N^\star < 1$. Then we define $\varepsilon := \frac{p_N^\star \cdot (1 - p_N^\star)}{2} > 0$. By definition of the infimum, $p_N^\star + \varepsilon$ is not a lower bound of $|\mathfrak{T}^{\leq N}|_{\mathrm{Leaf}}$ for all $(\mathcal{P}, \mathcal{S})$-computation trees that satisfy (+). Hence, there must exist a $(\mathcal{P}, \mathcal{S})$-computation tree $\mathfrak{T}$ satisfying (+) such that

$$p_N^\star \leq |\mathfrak{T}^{\leq N}|_{\mathrm{Leaf}} < p_N^\star + \varepsilon. \tag{5.34}$$

For readability, we set $Z := \mathrm{Leaf}^{\mathfrak{T}^{\leq N}}$ to be the set of leaves of the tree $\mathfrak{T}^{\leq N}$ and $\overline{Z} := V^{\mathfrak{T}^{\leq N}} \setminus \mathrm{Leaf}^{\mathfrak{T}^{\leq N}}$ to be the set of inner nodes of the tree $\mathfrak{T}^{\leq N}$. By the monotonicity of $|\cdot|_{\mathrm{Leaf}}$ w.r.t. the depth of the tree $\mathfrak{T}^{\leq N}$, there must exist a natural number $m^\star \in \mathbb{N}$ such that

$$\sum_{x \in Z \wedge \mathrm{d}(x) \leq m^*} p_x > \tfrac{p_N^\star}{2}. \tag{5.35}$$

For every $x \in V$ with $\mathrm{d}(x) = m^\star$ and $x \notin Z$, we define the induced sub $(\mathcal{P}, \mathcal{S})$-computation tree starting at node x by $\mathfrak{T}^{\leq N}(x) := \mathfrak{T}^{\leq N}[x(E^{\mathfrak{T}^{\leq N}})^*]$. Then we have

$$|\mathfrak{T}^{\leq N}|_{\mathrm{Leaf}} = \sum_{x \in Z \wedge \mathrm{d}(x) \leq m^*} p_x + \sum_{x \in \overline{Z} \wedge \mathrm{d}(x) = m^*} p_x \cdot |\mathfrak{T}^{\leq N}(x)|_{\mathrm{Leaf}}, \tag{5.36}$$

Furthermore, we have

$$\sum_{x \in \overline{Z} \wedge \mathrm{d}(x) = m^*} p_x = 1 - \sum_{x \in Z \wedge \mathrm{d}(x) \leq m^*} p_x \tag{5.37}$$

since $\sum_{x\in\overline{Z}\wedge \mathrm{d}(x)=m^*} p_x + \sum_{x\in Z\wedge \mathrm{d}(x)\leq m^*} p_x = 1$, as these are the leaves of the finite, grounded induced sub chain tree, where we cut every edge after the nodes of depth m^*. We obtain

$$
\begin{aligned}
& p_N^\star + \varepsilon \\
> {} & |\mathfrak{T}^{\leq N}|_{\mathrm{Leaf}} \qquad \text{(by (5.34))} \\
= {} & \sum_{x\in Z\wedge \mathrm{d}(x)\leq m^*} p_x + \sum_{x\in\overline{Z}\wedge \mathrm{d}(x)=m^*} p_x \cdot \underbrace{|\mathfrak{T}^{\leq N}(x)|_{\mathrm{Leaf}}}_{\geq p_N^\star} \qquad \text{(by (5.36) and (5.33))} \\
\geq {} & \sum_{x\in Z\wedge \mathrm{d}(x)\leq m^*} p_x + \sum_{x\in\overline{Z}\wedge \mathrm{d}(x)=m^*} p_x \cdot p_N^\star \\
= {} & \sum_{x\in Z\wedge \mathrm{d}(x)\leq m^*} p_x + p_N^\star \cdot \sum_{x\in\overline{Z}\wedge \mathrm{d}(x)=m^*} p_x \\
= {} & \sum_{x\in Z\wedge \mathrm{d}(x)\leq m^*} p_x + p_N^\star \cdot (1 - \sum_{x\in Z\wedge \mathrm{d}(x)\leq m^*} p_x) \qquad \text{(by (5.37))} \\
= {} & p_N^\star + \sum_{x\in Z\wedge \mathrm{d}(x)\leq m^*} p_x - p_N^\star \cdot \sum_{x\in Z\wedge \mathrm{d}(x)\leq m^*} p_x \\
= {} & p_N^\star + \sum_{x\in Z\wedge \mathrm{d}(x)\leq m^*} p_x \cdot (1 - p_N^\star) \\
> {} & p_N^\star + (1 - p_N^\star) \cdot \tfrac{p_N^\star}{2} \qquad \text{(by (5.35))} \\
= {} & p_N^\star + \varepsilon, \quad \lightning
\end{aligned}
$$

a contradiction. So $p_N^\star = 1$ (see [17, Thm. 2.2]). In particular, this means that for every $N \in \mathbb{N}$ and every $(\mathcal{P}, \mathcal{S})$-computation tree $\mathfrak{T}$ satisfying (+), we have

$$|\mathfrak{T}^{\leq N}|_{\mathrm{Leaf}} = 1. \tag{5.38}$$

5. Finally, we prove that $|\mathfrak{T}|_{\mathrm{Leaf}} = 1$

We adjust the value function for this new tree $\mathfrak{T}^{\leq N}$ once again and define:

$$\widehat{\mathcal{V}al} : V^{\mathfrak{T}^{\leq N}} \to \mathbb{N}, x \mapsto \begin{cases} N+1, & \text{if } x \in C \\ 0, & \text{if } x \in \mathrm{Leaf}^{\mathfrak{T}} \\ Pol(A_x) + 1, & \text{otherwise} \end{cases}$$

Now for a node $x \in V^{\mathfrak{T}^{\leq N}}$ we have $0 \leq \widehat{\mathcal{V}al}(x) \leq N+1$. Furthermore, we have $\mathcal{V}al(\mathfrak{r}^{\mathfrak{T}}) \geq \widehat{\mathcal{V}al}(\mathfrak{r}^{\mathfrak{T}^{\leq N}})$ for the root of $\mathfrak{T}$. Let $|\mathfrak{T}^{\leq N}|_{\widehat{\mathcal{V}al}} \coloneqq \sum_{x\in\mathrm{Leaf}^{\mathfrak{T}^{\leq N}}} p_x \cdot \widehat{\mathcal{V}al}(x)$. Note that this sum is monotonic increasing and bounded from above by $N+1$, since

$$\sum_{x\in\mathrm{Leaf}^{\mathfrak{T}^{\leq N}}} p_x \cdot \widehat{\mathcal{V}al}(x) \leq \sum_{x\in\mathrm{Leaf}^{\mathfrak{T}^{\leq N}}} p_x \cdot (N+1) = (N+1) \cdot \sum_{x\in\mathrm{Leaf}^{\mathfrak{T}^{\leq N}}} p_x \leq (N+1) \cdot 1 = N+1$$

and thus absolutely convergent. We have $\widehat{\mathcal{V}al}(\mathfrak{r}^{\mathfrak{T}^{\leq N}}) \geq |\mathfrak{T}^{\leq N}|_{\widehat{\mathcal{V}al}}$ because every edge is either a $\xrightarrow{\mathrm{i}}_{\mathcal{P},\mathcal{S}}$ or $\xrightarrow{\mathrm{i}}_{\mathcal{S}}$ step and due to (d) and (e), we know that the (expected) value is non-increasing.

Now we fix $N \in \mathbb{N}$ and a computation tree $\mathfrak{T}$ satisfying (+), and obtain the corresponding transformed tree $\mathfrak{T}^{\leq N}$. Note that by (5.38) we have $|\mathfrak{T}^{\leq N}|_{\mathrm{Leaf}} = 1 = q_N + (1 - q_N)$, where $q_N \coloneqq \sum_{x\in\mathrm{Leaf}^{\mathfrak{T}^{\leq N}} \wedge \widehat{\mathcal{V}al}(x)=0} p_x$. So what is $|\mathfrak{T}^{\leq N}|_{\widehat{\mathcal{V}al}}$? The probabilities of zero-valued nodes (i.e., leaves in the original tree) add up to q_N, while the probabilities of new leaves due to a cut add up to probability $1 - q_N$. So $|\mathfrak{T}^{\leq N}|_{\widehat{\mathcal{V}al}}$ has at least the value $q_N \cdot 0 + (1 - q_N) \cdot (N+1) = (1 - q_N) \cdot (N+1)$. Thus,

$$\mathcal{V}al(\mathfrak{r}^{\mathfrak{T}}) \geq \widehat{\mathcal{V}al}(\mathfrak{r}^{\mathfrak{T}^{\leq N}}) \geq |\mathfrak{T}^{\leq N}|_{\widehat{\mathcal{V}al}} \geq (1 - q_N) \cdot (N+1),$$

which implies $q_N \geq 1 - \frac{\mathcal{V}al(\mathfrak{r}^{\mathfrak{T}})}{N+1}$. Note that q_N is weakly monotonically increasing and bounded by 1 for $N \to \infty$. Hence, $q := \lim_{N\to\infty} q_N$ exists and $1 \geq q \geq \lim_{N\to\infty}(1 - \frac{\mathcal{V}al(\mathfrak{r}^{\mathfrak{T}})}{N+1}) = 1$, i.e., $q = 1$. Hence, we obtain $|\mathfrak{T}|_{\text{Leaf}} = \lim_{N\to\infty} q_N = q = 1$. Since we regard the limit of $N \to \infty$, this holds for any computation tree satisfying (+). ■

We have finally proven the main technical part of the reduction pair processor in the probabilistic setting. Next, we want to prove the soundness and completeness of the actual processor, which is fairly easy using our technical lemma from before in addition to the P-Partition lemma.

Theorem 5.4.27 (Reduction Pair Processor)**.** *Let $(\mathcal{P}, \mathcal{S})$ be a DP problem, and let $Pol : \mathcal{T}\left(\Sigma \uplus \Sigma^{\#}, \mathcal{V}\right) \to \mathbb{N}[\mathcal{V}]$ be a natural, multilinear polynomial interpretation which is weakly monotonic (i.e., $x \geq y$ implies $f_{Pol}(\ldots, x, \ldots) \geq f_{Pol}(\ldots, y, \ldots)$ for all $f \in \Sigma \cup \Sigma^{\#}$). Suppose that we have $\mathcal{P} = \mathcal{P}_{\succcurlyeq} \uplus \mathcal{P}_{\succ}$ and the following conditions hold:*

(1) *For every $(\ell^{\#}, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in \mathcal{P}_{\succ}$, there exists a $1 \leq j \leq k$ with*

$$Pol(s) > Pol(C_j)$$

If $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{S}$, then we additionally require that

$$Pol(\ell) \geq Pol(r_j)$$

(2) *For every $(\ell^{\#}, \ell) \to \{p_1 : (C_1, r_1), \ldots, p_k : (C_k, r_k)\} \in \mathcal{P}$, we have*

$$Pol(s) \geq \sum_{1\leq j\leq k} p_j \cdot Pol(C_j)$$

(3) *For every $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{S}$ we have*

$$Pol(\ell) \geq \sum_{1\leq j\leq k} p_j \cdot Pol(r_j)$$

Then

$$\text{Proc}_{\texttt{RP}}(\mathcal{P}, \mathcal{S}) = \{(\mathcal{P}_{\succcurlyeq}, \mathcal{S})\}$$

is sound and complete.

Proof.

complete: Every $(\mathcal{P}_{\succcurlyeq}, \mathcal{S})$-computation tree is also a $(\mathcal{P}, \mathcal{S})$-computation tree. Hence, if $(\mathcal{P}_{\succcurlyeq}, \mathcal{S})$ is not innermost AST, then $(\mathcal{P}, \mathcal{S})$ is also not innermost AST.

sound: Suppose that every $(\mathcal{P}_{\succcurlyeq}, \mathcal{S})$-computation tree converges with probability 1. Assume for a contradiction that there exists a $(\mathcal{P}, \mathcal{S})$-computation tree $\mathfrak{T} = (V, E, L, P)$ that converges with probability < 1. We can partition $P = P_{\succcurlyeq} \uplus P_{\succ}$, according to $\mathcal{P}_{\succcurlyeq} \uplus \mathcal{P}_{\succ}$. Then we apply the P-Partition Lemma (Lemma 5.2.20), which means that there must be a $(\mathcal{P}, \mathcal{S})$-computation tree $\mathfrak{T}'$ with $|\mathfrak{T}'|_{\text{Leaf}} < 1$ such that every infinite path will have an infinite number of edges corresponding to $\mathcal{P}_{\succ}$ steps. But this is a contradiction to Lemma 5.4.26. ■

Example 5.4.28. Consider $\mathcal{R}_{div}$ and $\mathcal{DT}(\mathcal{R}_{div})$ from Example 5.0.1 and Example 5.1.10. After applying the usable rules processor in Example 5.4.24, we have the following four DP problems left:

$$(\{\mathcal{DT}((5.1))\}, \varnothing)$$

$$(\{\mathcal{DT}((5.2))\}, \varnothing)$$

$$(\{\mathcal{DT}((5.3))\}, \varnothing)$$

$$(\{\mathcal{DT}((5.4))'\}, \{(5.1), (5.2)\})$$

The constraints of the reduction pair processor for every DP problem are satisfied by the polynomial interpretation which maps $\mathcal{O}$ to 0, $\mathsf{s}(x)$ to $2x+1$, $\mathsf{minus}^{\#}(x,y)$ and $\mathsf{div}^{\#}(x,y)$ to $x+1$, and all other non-constant function symbols to the projection on their first argument. Hence, we result with

$$\mathrm{Proc}_{\mathtt{RP}}((\mathcal{DT}((5.1)), \mathcal{R}_{div})) = \{(\varnothing, \varnothing)\}$$

$$\mathrm{Proc}_{\mathtt{RP}}((\mathcal{DT}((5.2)), \mathcal{R}_{div})) = \{(\varnothing, \varnothing)\}$$

$$\mathrm{Proc}_{\mathtt{RP}}((\mathcal{DT}((5.3)), \mathcal{R}_{div})) = \{(\varnothing, \varnothing)\}$$

$$\mathrm{Proc}_{\mathtt{RP}}((\mathcal{DT}((5.4))', \mathcal{R}_{div})) = \{(\varnothing, \{(5.1), (5.2)\})\}$$

Example 5.4.29. Consider $\mathcal{R}_{div}$ and $\mathcal{DT}(\mathcal{R}_{div})$ from Example 5.0.1 and Example 5.1.10. We are also able to directly use the reduction pair processor to prove innermost AST for the DP problem $(\mathcal{DT}(\mathcal{R}_{div}), \mathcal{R}_{div})$. The constraints of the reduction pair processor, such that every dependency tuple get removed, are satisfied by the polynomial interpretation which maps $\mathcal{O}$ to 0, $\mathsf{s}(x)$ to $3x+2$, $\mathsf{minus}^{\#}(x,y)$ and $\mathsf{div}^{\#}(x,y)$ to $x+1$, and all other non-constant function symbols to the projection on their first argument.

While for the non-probabilistic reduction pair processor, we have seen that it subsumes the direct application of polynomial interpretations (Theorem 3.3.17), it does not seem likely that a similar result holds for the probabilistic reduction pair processor. The reason for this is that we are working with dependency tuples instead of dependency pairs in the probabilistic DP framework. This means that we have to take the sum of the polynomial values of all occurring subterms with a defined root symbol. This sum may contain the same subterm multiple times. Now, we can not use the same polynomial ordering as for the direct application anymore.

Example 5.4.30 (Innermost AST with Data Structures)**.** Consider $\mathcal{R}_{len}$ from Example 4.3.3. Here, we get the following dependency tuples: Here, we get $\mathcal{DT}(\mathcal{R}_{len}) = \{\mathcal{DT}((2.7)), \mathcal{DT}((2.8))\}$ with

$$\begin{aligned}
\mathcal{DT}((2.7)) = \;& \Big(\mathsf{len}^{\#}(\mathsf{nil}), \mathsf{len}(\mathsf{nil})\Big) \\
& \to \{\;\tfrac{1}{2}: \Big(\{\!\!\{\Big(\mathsf{len}^{\#}(\mathsf{nil}), \varepsilon\Big)\}\!\!\}, \mathsf{len}(\mathsf{nil})\Big), \tfrac{1}{2}: \Big(\varnothing, \mathcal{O}\Big)\} \\
\mathcal{DT}((2.8)) = \;& \Big(\mathsf{len}^{\#}(\mathsf{cons}(x,y)), \mathsf{len}(\mathsf{cons}(x,y))\Big) \\
& \to \{\;\tfrac{1}{2}: \Big(\{\!\!\{\Big(\mathsf{len}^{\#}(\mathsf{cons}(x,y)), \varepsilon\Big)\}\!\!\}, \mathsf{len}(\mathsf{cons}(x,y))\Big), \\
& \qquad \tfrac{1}{2}: \Big(\{\!\!\{\Big(\mathsf{len}^{\#}(y), 1\Big)\}\!\!\}, \mathsf{s}(\mathsf{len}(y))\Big)\}
\end{aligned}$$

We can apply the reduction pair processor with the polynomial interpretation that maps $\mathsf{len}(x)$ to x, $\mathsf{s}(x)$ to x, $\mathsf{cons}(x,y)$ to $y+1$, and both nil and $\mathcal{O}$ to 0.

Final Automatic Innermost AST Proof

Finally, we present the final automatic innermost AST proof for our PTRS $\mathcal{R}_{div}$ in the following figure.

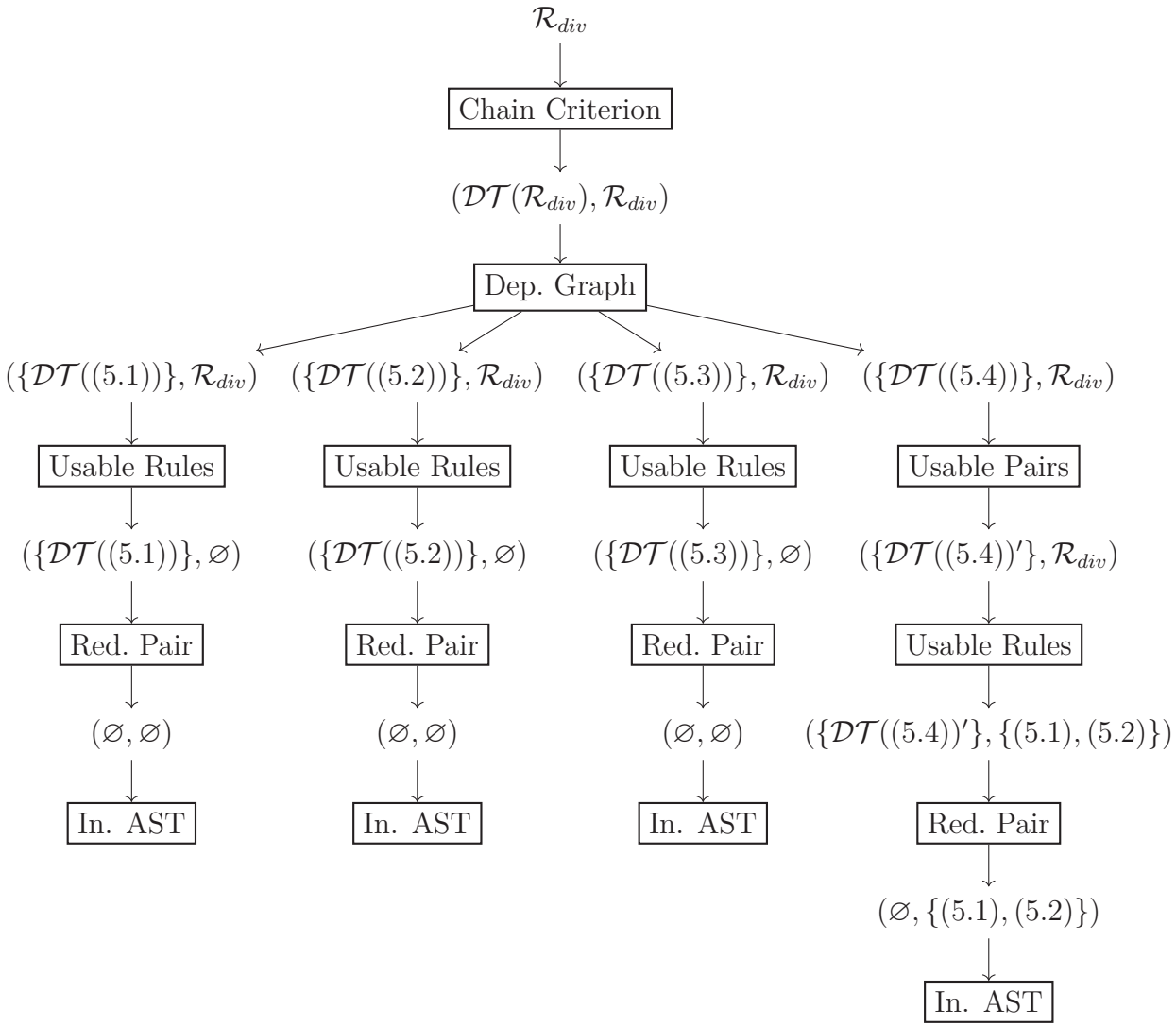


Figure 5.23: Automatic Innermost Ast Proof for $\mathcal{R}_{div}$

6 Comparison of the Frameworks

In this chapter, we will illustrate the differences between the two frameworks in more detail. To do so, we use the TRS for the integer division $\mathcal{R}_{div}$ from Example 3.0.1. First, let us specify the TRS here again and also present a PTRS variant of it. Consider the signature $\Sigma_{div} = \{\mathcal{O}, \mathsf{s}, \mathsf{minus}, \mathsf{div}\}$, a variable set with $\{x, y\} \subseteq \mathcal{V}$, and the following TRS $\mathcal{R}_{div}$ that computes the integer division of two natural numbers:

$$\mathsf{minus}(x, \mathcal{O}) \to x \tag{6.1}$$
$$\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{minus}(x, y) \tag{6.2}$$
$$\mathsf{div}(\mathcal{O}, \mathsf{s}(y)) \to \mathcal{O} \tag{6.3}$$
$$\mathsf{div}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{s}\,(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y))) \tag{6.4}$$

The corresponding PTRS variant $\mathcal{S}_{div}$ has the form:

$$\mathsf{minus}(x, \mathcal{O}) \to \{1 : x\} \tag{6.5}$$
$$\mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) \to \{1 : \mathsf{minus}(x, y)\} \tag{6.6}$$
$$\mathsf{div}(\mathcal{O}, \mathsf{s}(y)) \to \{1 : \mathcal{O}\} \tag{6.7}$$
$$\mathsf{div}(\mathsf{s}(x), \mathsf{s}(y)) \to \{1 : \mathsf{s}\,(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y)))\} \tag{6.8}$$

Dependency Pairs

We get the following three dependency pairs in the non-probabilistic framework:

$$\mathsf{minus}^{\#}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{minus}^{\#}(x, y) \tag{6.9}$$
$$\mathsf{div}^{\#}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{minus}^{\#}(x, y) \tag{6.10}$$
$$\mathsf{div}^{\#}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{div}^{\#}(\mathsf{minus}(x, y), \mathsf{s}(y)) \tag{6.11}$$

and the following four dependency tuples in the probabilistic framework:

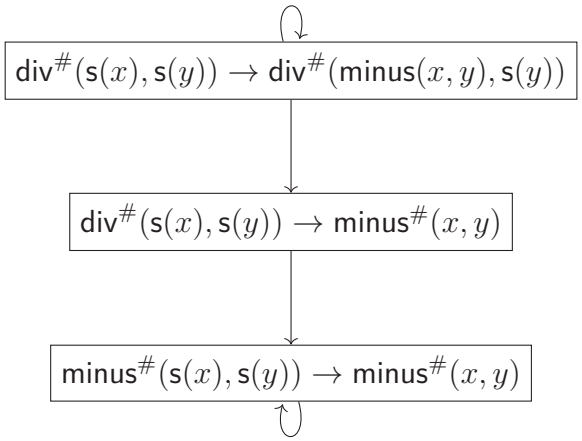
$$\begin{aligned}
\mathcal{DT}((6.5)) = \;& (\mathsf{minus}^{\#}(x, \mathcal{O}), \mathsf{minus}(x, \mathcal{O})) \\
& \to \{\;\; 1 : \;\; \big(\varnothing, x\big)\} \\
\mathcal{DT}((6.6)) = \;& (\mathsf{minus}^{\#}(\mathsf{s}(x), \mathsf{s}(y)), \mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y))) \\
& \to \{\;\; 1 : \;\; \big(\{\!\!\{(\mathsf{minus}^{\#}(x, y), \varepsilon)\}\!\!\}, \mathsf{minus}(x, y))\} \\
\mathcal{DT}((6.7)) = \;& (\mathsf{div}^{\#}(\mathcal{O}, \mathsf{s}(y)), \mathsf{div}(\mathcal{O}, \mathsf{s}(y))) \\
& \to \{\;\; 1 : \;\; \big(\varnothing, \mathcal{O}\big)\} \\
\mathcal{DT}((6.8)) = \;& (\mathsf{div}^{\#}(\mathsf{s}(x), \mathsf{s}(y)), \mathsf{div}(\mathsf{s}(x), \mathsf{s}(y))) \\
& \to \{\;\; 1 : \;\; \big(\{\!\!\{(\mathsf{div}^{\#}(\mathsf{minus}(x, y), \mathsf{s}(y)), 1), (\mathsf{minus}^{\#}(x, y), 1.1)\}\!\!\}, \\
& \qquad\qquad\qquad\qquad \mathsf{s}(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y)))\big)\}
\end{aligned}$$

The main difference here is that we are working with dependency tuples instead of dependency pairs in the probabilistic framework. We create an own dependency tuple for

each of the four rules and include all of the defined symbols on the right-hand side of the rule. However, the dependency pairs $\mathcal{DT}((6.5))$ and $\mathcal{DT}((6.7))$ only have an empty set in the distribution on the right-hand side. Hence, those two rules will be automatically removed by our dependency graph processor. More importantly, we have stored both of the defined symbols from the right-hand side of (6.8) in a single dependency tuple in the probabilistic framework, while we generate two separate dependency pairs for them in the non-probabilistic framework. We now go over each processor and explain the differences in both frameworks.

Dependency Graph Processor and Usable Pairs

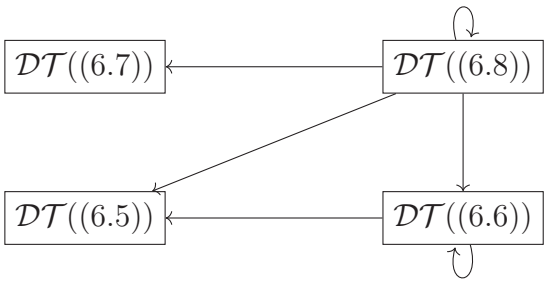
For the non-probabilistic framework, we get the following $(\mathcal{DP}(\mathcal{R}_{div}), \mathcal{R}_{div})$-dependency graph:



Here, we have two SCCs and get

$$\mathrm{Proc}_{\mathtt{DG}}(\mathcal{DP}(\mathcal{R}_{div}), \mathcal{R}_{div}) = \{\Big(\{(6.9)\}, \mathcal{R}_{div}\Big), \Big(\{(6.11)\}, \mathcal{R}_{div}\Big)\}$$

For the probabilistic framework, we get the following $(\mathcal{DT}(\mathcal{S}_{div}), \mathcal{S}_{div})$-dependency graph:



Here, we have two SCCs as well and get

$$\mathrm{Proc}_{\mathtt{DG}}(\mathcal{DT}(\mathcal{S}_{div}), \mathcal{S}_{div}) = \{\Big(\{\mathcal{DT}((6.6))\}, \mathcal{S}_{div}\Big), \Big(\{\mathcal{DT}((6.8))\}, \mathcal{S}_{div}\Big)\}$$

We can see that the dependency pairs with an empty set in the distribution on the right-hand side get removed. The difference for both frameworks is that in the non-probabilistic framework, we got rid of the dependency pair $\mathsf{div}^{\#}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{minus}^{\#}(x, y)$ as it is not contained in any SCC. In the probabilistic framework, we coupled the two dependency pairs together in the dependency tuple for the fourth rule, which means that we are still

keeping track of the dependency term $\mathsf{minus}^{\#}(x,y)$ in the set of the right-hand side of $\mathcal{DT}((6.8))$. Here comes our new processor, the usable pairs processor, into the play. Using the usable pairs processor, we can remove the pair $\Big(\mathsf{minus}^{\#}(x,y),1.1\Big)$ from the right-hand side of $\mathcal{DT}((6.8))$ in the DP problem $(\{\mathcal{DT}((6.8))\},\mathcal{R}_{div})$ so that we result with

$$\mathrm{Proc}_{\mathtt{UP}}(\{\mathcal{DT}((6.8))\},\mathcal{R}_{div}) = \{\Big(\{\mathcal{DT}((6.8))'\},\mathcal{R}_{div}\Big)\}$$

with

$$\begin{aligned}\mathcal{DT}((6.8))' = \quad & \Big(\mathsf{div}^{\#}(\mathsf{s}(x),\mathsf{s}(y)),\mathsf{div}(\mathsf{s}(x),\mathsf{s}(y))\Big)\\ & \to \{\quad 1: \quad \Big(\{\!\{\Big(\mathsf{div}^{\#}(\mathsf{minus}(x,y),\mathsf{s}(y)),1\Big)\}\!\},\mathsf{s}(\mathsf{div}(\mathsf{minus}(x,y),\mathsf{s}(y)))\Big)\}\end{aligned}$$

All in all, we get the same result for both frameworks w.r.t. the dependency graph. This is not a big surprise since the definition of the probabilistic dependency graph, and the usable pairs processor only looked at the non-probabilistic structure of the PTRS. Furthermore, we use the same abstraction for both frameworks for the computable dependency graph processor, so there is also no difference in that regard.

Usable Rules Processor

The usable pairs processor is completely the same for both frameworks, as we are again using the non-probabilistic structure of the PTRS and the dependency tuples in the probabilistic framework. In the non-probabilistic framework, we get

$$\begin{aligned}\mathrm{Proc}_{\mathtt{UR}}\Big(\{(6.9)\},\mathcal{R}_{div}\Big) &= \{\Big(\{(6.9)\},\varnothing\Big)\}\\ \mathrm{Proc}_{\mathtt{UR}}\Big(\{(6.11)\},\mathcal{R}_{div}\Big) &= \{\Big(\{(6.11)\},\{(6.1),(6.2)\}\Big)\}\end{aligned}$$

and in the probabilistic framework, we get

$$\begin{aligned}\mathrm{Proc}_{\mathtt{UR}}\Big(\{\mathcal{DT}((6.6))\},\mathcal{S}_{div}\Big) &= \{\Big(\{\mathcal{DT}((6.6))\},\varnothing\Big)\}\\ \mathrm{Proc}_{\mathtt{UR}}\Big(\{\mathcal{DT}((6.8))'\},\mathcal{S}_{div}\Big) &= \{\Big(\{\mathcal{DT}((6.8))'\},\{(6.5),(6.6)\}\Big)\}\end{aligned}$$

Reduction Pair Processor

The reduction pair processor is the only processor where we have a fundamental difference between both frameworks. Besides the obvious disadvantage that we are only allowing multilinear polynomial interpretations in the probabilistic setting, there is also a difference based on the fact that we are working with dependency tuples instead of dependency pairs. Let us once again look at the DP problems $(\mathcal{DP}(\mathcal{R}_{div}),\mathcal{R}_{div})$ and $(\mathcal{DT}(\mathcal{S}_{div}),\mathcal{S}_{div})$. If we want to remove all of the dependency pairs using the reduction pair processor, then in the non-probabilistic framework, we need to find a polynomial interpretation such that

$$Pol(\mathsf{minus}^{\#}(\mathsf{s}(x),\mathsf{s}(y))) > Pol(\mathsf{minus}^{\#}(x,y)) \tag{6.12}$$

$$Pol(\mathsf{div}^{\#}(\mathsf{s}(x),\mathsf{s}(y))) > Pol(\mathsf{minus}^{\#}(x,y)) \tag{6.13}$$

$$Pol(\mathsf{div}^{\#}(\mathsf{s}(x),\mathsf{s}(y))) > Pol(\mathsf{div}^{\#}(\mathsf{minus}(x,y),\mathsf{s}(y))) \tag{6.14}$$

In the probabilistic framework, we would need to find a polynomial interpretation such that

$$Pol(\mathsf{minus}^{\#}(x, \mathcal{O})) > Pol(\varnothing) = 0 \tag{6.15}$$

$$Pol(\mathsf{minus}^{\#}(\mathsf{s}(x), \mathsf{s}(y))) > Pol(\mathsf{minus}^{\#}(x, y)) \tag{6.16}$$

$$Pol(\mathsf{div}^{\#}(\mathcal{O}, \mathsf{s}(y))) > Pol(\varnothing) = 0 \tag{6.17}$$

$$\begin{aligned} Pol(\mathsf{div}^{\#}(\mathsf{s}(x), \mathsf{s}(y))) &> Pol(\{\!\{\left(\mathsf{div}^{\#}(\mathsf{minus}(x, y), \mathsf{s}(y)), 1\right), \left(\mathsf{minus}^{\#}(x, y), 1.1\right)\}\!\}) \\ &= Pol(\mathsf{div}^{\#}(\mathsf{minus}(x, y), \mathsf{s}(y))) + Pol(\mathsf{minus}^{\#}(x, y)) \end{aligned} \tag{6.18}$$

This is strictly harder to satisfy than the three inequations (6.12), (6.13) and (6.14), due to the fact that we have the sum of $Pol(\mathsf{div}^{\#}(\mathsf{minus}(x, y), \mathsf{s}(y)))$ and $Pol(\mathsf{minus}^{\#}(x, y))$ in the inequations for the probabilistic RPP, while we have to independent inequations for the non-probabilistic RPP. The reason for this difference is that in order to prove AST, we have to take the number of occurrences of a defined symbol into account, while for a termination proof, we only have to show the existence of an infinite path.

7 Conclusion

In this thesis, we developed new results in the research areas of probabilistic rewriting and automatic termination analysis of probabilistic programs. For probabilistic rewriting, we used the definition of a probabilistic term rewriting system of Avanzini et al. [2]. We adapted different rewrite strategies like innermost rewriting and leftmost innermost rewriting to the probabilistic setting. Here, we have shown that in contrast to the non-probabilistic setting, we have no equality between leftmost innermost almost-sure termination and innermost almost-sure termination. We also created a new way to view rewrite sequences as a tree and characterized almost-sure termination with this new tree representation.

Regarding the automatic termination analysis for probabilistic term rewriting systems, we first developed a technique to prove AST automatically by finding a polynomial interpretation that satisfies certain inequalities. In addition to this direct application of polynomial interpretations, we were able to adapt the dependency pair framework to the probabilistic setting. This adaption includes a new definition of dependency tuples and chains. For this new probabilistic dependency pair framework, we were able to adapt three of the most important processors (the reduction pair processor, the dependency graph processor, and the usable rules processor) to the probabilistic setting. It seems very likely that we will also be able to adapt further processors to the probabilistic setting in the future. Furthermore, we introduced two new processors specifically designed for the probabilistic setting (the usable pairs processor and the not probabilistic processor). We were able to prove the soundness and completeness of all processors, except for the usable rules processors. To create a complete usable rules processor, we need a third component in our DP problem, which we omitted for readability in this thesis. With all those processors, we can now split a huge DP problem into simpler ones until we prove innermost AST for all of them. Since the DP framework is very powerful in the non-probabilistic setting, we expect that also the probabilistic DP framework will be very effective. We are currently implementing our new probabilistic DP framework into the Automated Program Verification Environment (AProVE) to test its applicability in practice and compare it to other existing tools that analyze PTRSs, which currently only is the tool of Avanzini et al. [2].

Whether there exist syntactic properties which guarantee that innermost almost-sure termination and almost-sure termination are equal remains an open question that we plan to address in the future. If there exist such properties similar to the ones in the non-probabilistic setting, then we can also prove almost-sure termination w.r.t. an arbitrary evaluation strategy using our newly developed techniques for innermost almost-sure termination. An important open problem for future research is adapting our probabilistic DP framework for innermost AST to a probabilistic DP framework for AST with an arbitrary evaluation strategy so that we do not rely on the syntactic properties mentioned before. The critical problem here is the definition of our probabilistic dependency tuples.

To allow arbitrary evaluation strategies, we have to adjust the definition of the dependency tuples and the PPTRS to create a sound chain criterion in this case. If we manage to do this, the processors must also be adapted to this new setting. Note that the usable rules processor only works w.r.t. innermost evaluation, but all of the other processors also work w.r.t. an arbitrary evaluation strategy in the non-probabilistic setting.

Currently, we cannot prove that a given PTRS is not innermost AST automatically, as there exists no probabilistic processor to do so. If we can adapt a processor from the non-probabilistic setting that shows non-termination, then we can also use our DP framework and all of its processors to disprove AST since every processor we have so far is also complete (except for the usable rules processor that can be made complete with an additional third component in our DP problem). And if we are able to disprove AST automatically, then it would also be interesting to adapt our DP framework to prove that a given PTRS converges only with a probability below some fixed bound and to compute this bound automatically. Another important question is whether we can adapt our DP framework to prove positive almost-sure termination. For such a task, we would need to adapt our DP framework more drastically since PAST is not compositional, while AST is, and one of the core lemmas (the P-Partition Lemma 5.2.20) for our probabilistic DP framework heavily relies on the compositionality of AST. Since we want to analyze arbitrary probabilistic programs, it is also an important task to adapt the transformation of other programming languages like Java into term rewriting systems to the probabilistic setting so that we also can analyze more complex probabilistic programming languages. Finally, another interesting task is to create a framework that automatically analyzes the expected runtimes of probabilistic term rewriting systems. Here, we have an existing framework in the non-probabilistic setting that analyzes the complexity of a given TRS and already works with dependency tuples. It is also tempting to use our new definitions of coupled positional dependency tuples to revisit this complexity analysis in the non-probabilistic setting.

Bibliography

[1] Thomas Arts and Jürgen Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.

[2] Martin Avanzini, Ugo Dal Lago, and Akihisa Yamada. On probabilistic term rewriting. *Science of Computer Programming*, 185, 2020.

[3] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. 1999.

[4] Raven Beutner and Luke Ong. On probabilistic termination of functional programs with continuous distributions. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, pages 1312–1326, 2021.

[5] Marc Bezem, JW Klop, and Roel de Vrijer. *Term rewriting systems*. 2003.

[6] Olivier Bournez and Florent Garnier. Proving positive almost-sure termination. In *Term Rewriting and Applications*, pages 323–337, 2005.

[7] Aleksandar Chakarov and Sriram Sankaranarayanan. Probabilistic program analysis with martingales. In *Computer Aided Verification*, pages 511–526, 2013.

[8] Krishnendu Chatterjee, Hongfei Fu, and Amir Kafshdar Goharshady. Termination analysis of probabilistic programs through positivstellensatz's. In *Computer Aided Verification*, pages 3–22, 2016.

[9] Ugo Dal Lago and Charles Grellois. Probabilistic termination by monadic affine sized typing. In *Programming Languages and Systems*, pages 393–419, 2017.

[10] Nachum Dershowitz. Orderings for term-rewriting systems. *Theoretical computer science*, 17:279–301, 1982.

[11] Claudia Faggian. Probabilistic rewriting and asymptotic behaviour: on termination and unique normal forms. *Logical Methods in Computer Science*, 18, 2022.

[12] Jürgen Giesl and Aart Middeldorp. Innermost termination of context-sensitive rewriting. In *Developments in Language Theory*, pages 231–244, 2003.

[13] Jürgen Giesl, René Thiemann, and Peter Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 301–331, 2005.

[14] Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, and Stephan Falke. Mechanizing and improving dependency pairs. *Journal of Automated Reasoning*, 37:155–203, 2006.

[15] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[16] Bernhard Gramlich. Abstract relations between restricted termination and confluence properties of rewrite systems. *Fundamenta Informaticae*, 24:2–23, 1995.

[17] Sergiu Hart, Micha Sharir, and Amir Pnueli. Termination of probabilistic concurrent program. *ACM Transactions on Programming Languages and Systems*, 5:356–380, 1983.

[18] S. Kamin and J.-J. Lévy. Two generalizations of the recursive path ordering. 1980.

[19] Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. On the hardness of analyzing probabilistic programs. *Acta Informatica*, 56:255–285, 2019.

[20] Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Federico Olmedo. Weakest precondition reasoning for expected runtimes of randomized algorithms. *Journal of the ACM*, 65:1–68, 2018.

[21] Dexter Kozen. Semantics of probabilistic programs. *Journal of Computer and System Sciences*, 22:328–350, 1981.

[22] M.R.K. Krishna Rao. Some characteristics of strong innermost normalization. *Theoretical Computer Science*, 239:141–164, 2000.

[23] Dallas S. Lankford. Canonical algebraic simplification in computational logic. *Technical Report ATP-25*, 1975.

[24] Dallas S. Lankford. On proving term rewriting systems are Noetherian. *Memo MTP-3*, 1979.

[25] Lorenz Leutgeb, Georg Moser, and Florian Zuleger. Automated expected amortised cost analysis of probabilistic data structures. In *Computer Aided Verification*, pages 70–91, 2022.

[26] Z. Manna and S. Ness. On the termination of Markov algorithms. In *Proceedings of the 3rd Hawaii International Conference on System Science*, pages 789–792, 1970.

[27] Annabelle McIver, Carroll Morgan, Benjamin Lucien Kaminski, and Joost-Pieter Katoen. A new proof rule for almost-sure termination. In *Proceedings of the ACM on Programming Languages*, volume 2, pages 1–28, 2017.

[28] David Monniaux. An abstract analysis of the probabilistic termination of programs. In *International Static Analysis Symposium*, pages 111–126, 2001.

[29] Lars Noschinski, Fabian Emmes, and Jürgen Giesl. Analyzing innermost runtime complexity of term rewriting by dependency pairs. *Journal of Automated Reasoning*, 51:27–56, 2013.

[30] Carsten Otto, Marc Brockschmidt, Christian von Essen, and Jürgen Giesl. Automated Termination Analysis of Java Bytecode by Term Rewriting. In *Proceedings of the 21st International Conference on Rewriting Techniques and Applications*, volume 6, pages 259–276, 2010.

[31] Michael O Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12:128–138, 1980.

[32] Di Wang, David M. Kahn, and Jan Hoffmann. Raising expectations: Automating expected cost analysis with types. In *Proceedings of the ACM on Programming Languages*, volume 4, 2020.