Narrowing Trees for Syntactically Deterministic Conditional Term Rewriting Systems

Naoki Nishida
Graduate School of Informatics, Nagoya University, Nagoya, Japan
nishida@i.nagoya-u.ac.jp
https://orcid.org/0000-0001-8697-4970

Yuya Maeda
Graduate School of Informatics, Nagoya University, Nagoya, Japan
yuya@trs.css.i.nagoya-u.ac.jp

Abstract

A narrowing tree for a constructor term rewriting system and a pair of terms is a finite representation for the space of all possible innermost-narrowing derivations that start with the pair and end with non-narrowable terms. Narrowing trees have grammar representations that can be considered regular tree grammars. Innermost narrowing is a counterpart of constructor-based rewriting, and thus, narrowing trees can be used in analyzing constructor-based rewriting to normal forms. In this paper, using grammar representations, we extend narrowing trees to syntactically deterministic conditional term rewriting systems that are constructor systems. We show that narrowing trees are useful to prove two properties of a normal conditional term rewriting system: one is infeasibility of conditional critical pairs and the other is quasi-reducibility.

2012 ACM Subject Classification Theory of computation → Rewrite systems

Keywords and phrases conditional term rewriting, innermost narrowing, regular tree grammar

Digital Object Identifier 10.4230/LIPIcs.FSCD.2018.26

Funding This work was partially supported by JSPS KAKENHI Grant Number JP17H01722.

Acknowledgements We gratefully acknowledge the anonymous reviewers for their useful comments and suggestions to improve the paper.

1 Introduction

Conditional term rewriting [32, Chapter 7] is known to be more complicated than unconditional term rewriting in the sense of analyzing properties, e.g., *operational termination* [21] (*quasi-decreasingness* [32]), *confluence* [37], and *reachability* [6]. A popular approach to the analysis of conditional term rewriting systems (CTRSs, for short) is to transform a CTRS into an unconditional term rewriting system (a TRS, for short) that is in general an overapproximation of the CTRS w.r.t. reduction (cf. [32]). This approach enables us to use existing techniques for the analysis of TRSs. For example, a CTRS is operationally terminating if the *unraveled* TRS [22, 32] is terminating [5]. To prove termination of the unraveled TRS, we can use many techniques for proving termination of TRSs (cf. [32]). On the other hand, it is not so easy to analyze reachability which is relevant to, e.g., *infeasibility* of conditions – non-existence of substitutions satisfying conditions – of conditional rewrite rules, conditional critical pairs, etc.

3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018).
Editor: Hélène Kirchner; Article No. 26; pp. 26:1–26:20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Let us consider to prove confluence of the following *normal* 1-CTRS [31] defining *even* and *odd* predicates over the non-negative integers represented by $\mathsf{0}$ and s:

$$\mathcal{R}_1 = \left\{ \begin{array}{lll} \mathsf{e}(\mathsf{0}) \to \mathsf{true}, & \mathsf{e}(\mathsf{s}(x)) \to \mathsf{true} \Leftarrow \mathsf{o}(x) \twoheadrightarrow \mathsf{true}, & \mathsf{e}(\mathsf{s}(x)) \to \mathsf{false} \Leftarrow \mathsf{e}(x) \twoheadrightarrow \mathsf{true}, \\ \mathsf{o}(\mathsf{0}) \to \mathsf{false}, & \mathsf{o}(\mathsf{s}(x)) \to \mathsf{true} \Leftarrow \mathsf{e}(x) \twoheadrightarrow \mathsf{true}, & \mathsf{o}(\mathsf{s}(x)) \to \mathsf{false} \Leftarrow \mathsf{o}(x) \twoheadrightarrow \mathsf{true} \end{array} \right\}$$

Unfortunately, neither a transformational approach in [10, 9] nor a direct approach to reachability analysis to prove infeasibility of conditional critical pairs succeeds in proving confluence of $\mathcal{R}_1$. For example, $\mathcal{R}_1$ has the following four critical pairs:

$$\begin{array}{ll} \langle \mathsf{true}, \mathsf{false} \rangle \Leftarrow \mathsf{o}(x) \twoheadrightarrow \mathsf{true} \ \&\ \mathsf{e}(x) \twoheadrightarrow \mathsf{true} & \langle \mathsf{false}, \mathsf{true} \rangle \Leftarrow \mathsf{o}(x) \twoheadrightarrow \mathsf{true} \ \&\ \mathsf{e}(x) \twoheadrightarrow \mathsf{true} \\ \langle \mathsf{true}, \mathsf{false} \rangle \Leftarrow \mathsf{e}(x) \twoheadrightarrow \mathsf{true} \ \&\ \mathsf{o}(x) \twoheadrightarrow \mathsf{true} & \langle \mathsf{false}, \mathsf{true} \rangle \Leftarrow \mathsf{e}(x) \twoheadrightarrow \mathsf{true} \ \&\ \mathsf{o}(x) \twoheadrightarrow \mathsf{true} \end{array}$$

An operationally terminating CTRS is confluent if all critical pairs of the CTRS are infeasible (cf. [1, 4]). To prove infeasibility of the critical pairs above, it suffices to show non-existence of terms t such that $\mathsf{o}(t) \to^*_{\mathcal{R}_1} \mathsf{true}$ and $\mathsf{e}(t) \to^*_{\mathcal{R}_1} \mathsf{true}$. Thanks to the meaning of *even* and *odd* predicates, it would be easy for human to notice that such a term t does not exist. However, it is not so easy to mechanize a way to show non-existence of t. In fact, confluence provers for CTRSs, ConCon [35], CO3 [25], and CoScart [8], based on e.g., transformations of CTRSs into TRSs and/or reachability analysis for infeasibility of conditional critical pairs failed to prove confluence of $\mathcal{R}_1$ (see Confluence Competition 2016 [1] and 2017,[2] `489.trs` or `522.trs`). Note that a *semantic approach* in [19, 18] can prove confluence of $\mathcal{R}_1$ using AGES [11], a tool for generating logical models of order-sorted first-order theories (cf. [20]).

The (non-)existence of a term t with $\mathsf{o}(t) \to^*_{\mathcal{R}_1} \mathsf{true}$ and $\mathsf{e}(t) \to^*_{\mathcal{R}_1} \mathsf{true}$ can be reduced to the (non-)existence of substitutions θ such that $\mathsf{o}(x) \rightsquigarrow^*_{\theta,\mathcal{R}_1} \mathsf{true}$ and $\mathsf{e}(x) \rightsquigarrow^*_{\theta,\mathcal{R}_1} \mathsf{true}$, where $\rightsquigarrow$ denotes the *narrowing* step [14]. In addition, the non-existence of such substitutions can be reduced to the emptiness of the set of such substitutions, i.e., the emptiness of $\{\theta \mid \mathsf{o}(x) \rightsquigarrow^*_{\theta,\mathcal{R}_1} \mathsf{true},\ \mathsf{e}(x) \rightsquigarrow^*_{\theta,\mathcal{R}_1} \mathsf{true}\}$. From this viewpoint, the enumeration of substitutions obtained by narrowing from a pair of terms would be useful in analyzing rewriting sequences that starts with instances of the pair.

A *narrowing tree* [29] for a sufficiently complete constructor TRS $\mathcal{R}$ with the root pair $s \twoheadrightarrow t$ where t is a constructor term is a finite representation that defines the set of substitutions θ such that the pair $s \twoheadrightarrow t$ narrows to a special constant $\top$ by *innermost* narrowing $\overset{i}{\rightsquigarrow}_{\mathcal{R}}$ with a substitution θ (i.e., $(s \twoheadrightarrow t) \overset{i}{\rightsquigarrow}^*_{\theta,\mathcal{R}} \top$ and thus $\theta s \overset{c}{\to}^*_{\mathcal{R}} \theta t$). Note that $\twoheadrightarrow$ is considered a binary symbol, $(x \twoheadrightarrow x) \to \top$ is assumed to be implicitly included in $\mathcal{R}$, and $\overset{c}{\to}_{\mathcal{R}}$ denotes the *constructor-based rewriting* step which applies rewrite rules to *basic* terms. Note that a basic terms is of the form $f(u_1, \ldots, u_n)$ with a defined symbol f and constructor terms $u_1, \ldots, u_n$. A narrowing tree can be the enumeration of substitutions obtained by innermost narrowing of $\mathcal{R}$ to $\top$. The idea of narrowing trees has been extended to finite representations of SLD trees for logic programs [30].

In this paper, we extend narrowing trees to *syntactically deterministic conditional* term rewriting systems (a SDCTRS, for short) that are constructor systems. The class of SDCTRSs is reasonable to model functional programs. We do not directly extend narrowing trees to conditional systems, but we convert an SDCTRS to an equivalent unconditional constructor system that may have extra variables. Narrowing trees for the converted constructor system can be used for the original SDCTRS, i.e., they represent all substitutions derived by innermost narrowing of the original SDCTRS.

[1] `http://cops.uibk.ac.at/results/?y=2016&c=CTRS`
[2] `http://cops.uibk.ac.at/results/?y=2017-full-run&c=CTRS`

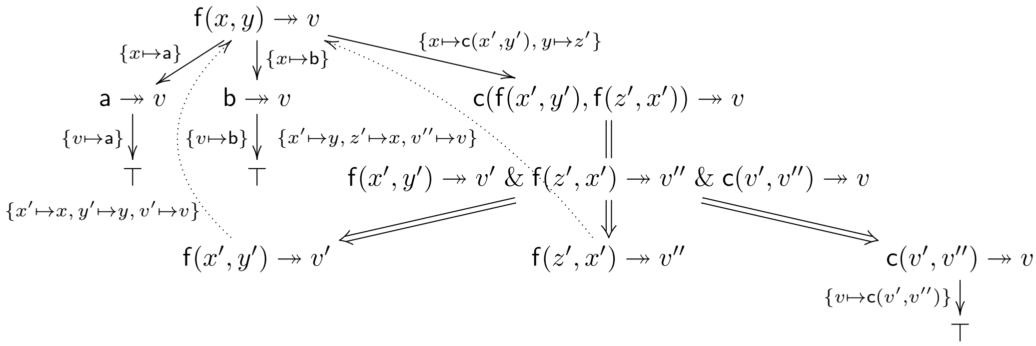

Figure 1 A narrowing tree for $\mathsf{f}(x, y) \twoheadrightarrow v$ w.r.t. $\mathcal{R}_2$.

Consider the sufficiently complete constructor TRS $\mathcal{R}_2 = \{\ \mathsf{f}(\mathsf{a}, z) \to \mathsf{a},\ \mathsf{f}(\mathsf{b}, z) \to \mathsf{b},\ \mathsf{f}(\mathsf{c}(x, y), z) \to \mathsf{c}(\mathsf{f}(x, y), \mathsf{f}(z, x))\ \}$. A narrowing tree for the pair $\mathsf{f}(x, y) \twoheadrightarrow v$ is illustrated in Figure 1. Labeled solid arrows "$\xrightarrow{\theta}$" represent innermost-narrowing steps with relevant substitutions θ,[3] double-line arcs "$=$" decompose nests of defined symbols (*flattening*), double arrows "$\Longrightarrow$" divide equations to single ones (*splitting*), labeled dotted arrows "$\overset{\delta}{\cdots\!>}$" visualize the existence of a variant node connected via a *renaming*[4] δ (*recursion*). The narrowing tree in Figure 1 can be written by the following *grammar representation* [29] that can be considered a *regular tree grammar* [3]:

$$\begin{array}{l} \Gamma_{\mathsf{f}(x,y)\twoheadrightarrow v} \to \{v \mapsto \mathsf{a}\} \bullet \{x \mapsto \mathsf{a}\} \mid \{v \mapsto \mathsf{b}\} \bullet \{x \mapsto \mathsf{b}\} \\ \qquad \mid \left(\begin{array}{c} \Gamma_{\mathsf{f}(x,y)\twoheadrightarrow v} \bullet \{x' \mapsto x, ,\ y' \mapsto y,\ v' \mapsto v\} \\ \& \\ \Gamma_{\mathsf{f}(x,y)\twoheadrightarrow v} \bullet \{x' \mapsto y,\ z' \mapsto x,\ v'' \mapsto v\} \\ \& \\ \{v \mapsto \mathsf{c}(v', v'')\} \end{array} \right) \bullet \{x \mapsto \mathsf{c}(x', y'),\ y \mapsto z'\} \end{array} \tag{1}$$

The binary symbols $\bullet$ and & are interpreted by standard composition and *parallel composition* [13, 33], respectively. Parallel composition $\Uparrow$ of two idempotent substitutions returns a most general unifier of the substitutions if the substitutions are unifiable. For example, $\{y' \mapsto \mathsf{a},\ y \mapsto \mathsf{a}\} \Uparrow \{y' \mapsto y\}$ returns $\{y' \mapsto \mathsf{a},\ y \mapsto \mathsf{a}\}$ and $\{y' \mapsto \mathsf{a},\ y \mapsto \mathsf{b}\} \Uparrow \{y' \mapsto y\}$ fails. Due to parallel composition (i.e., occurrence of &), it is not so easy to not only analyze but also simplify grammar representations of narrowing trees. In the remaining of this paper, we do not deal with narrowing trees but their grammar representations.

Throughout this paper, we aim at proving infeasibility of the condition $\mathsf{o}(x) \twoheadrightarrow \mathsf{true}\ \&\ \mathsf{e}(x) \twoheadrightarrow \mathsf{true}$ for $\mathcal{R}_1$[5] w.r.t. constructor-based rewriting. To this end, we first show that every

[3] One may think that y of $\mathsf{f}(x, y) \twoheadrightarrow v$ in Figure 1 does not have to be instantiated by z' because y is received by a variable that can be seen as patternless. However, the tree is used two or more times via dotted arrows, and the reuse always starts with $\mathsf{f}(x, y) \twoheadrightarrow v$ that is connected by means of a renaming attached with dotted arrows. To avoid any conflict of using y, we always introduce only fresh variables at narrowing steps, i.e., $\mathsf{f}(x, y) \overset{i}{\rightsquigarrow}_{\mathcal{R}_2} \mathsf{c}(\mathsf{f}(x', y), \mathsf{f}(z', x'))$ is not allowed (see Definition 3 in Section 3).

[4] To be precise, δ (e.g., $\{x' \mapsto y,\ z' \mapsto x,\ v'' \mapsto v\}$ in Figure 1) is not a renaming, while we can write an exact renaming. However, we write such a substitution, so-called a *prenaming* [17], obtained by restricting a renaming to variables that we are interested in because the renaming is used to rename a particular term.

[5] We use $\mathcal{R}_1$, which is an SDCTRS but also a normal 1-CTRS, as a leading example of this paper because $\mathcal{R}_1$ is reasonable to illustrate how we can use the grammar representation of a narrowing tree to prove confluence of a CTRS.

constructor SDCTRS can be converted to an equivalent unconditional constructor system which may have extra variables (Section 3). Secondly, we revisit *compositionality* of innermost narrowing, relaxing some assumptions in [29] (Section 4). Thirdly, we introduce grammar representations of sets of idempotent substitutions as regular tree grammars (Section 5) and a construction of narrowing trees for given unconditional constructor systems (Section 6). Fourthly, we show some methods to simplify grammar representations of narrowing trees (Section 7). Finally, we show that grammar representations of narrowing trees are useful to prove infeasibility of conditional critical pairs of $\mathcal{R}_1$ and *quasi-reducibility* [16] of $\mathcal{R}_1$ with usual sorts for natural numbers and boolean values (Section 8). Quasi-reducibility is that every ground basic term is defined (i.e., reducible). For (operationally) terminating (C)TRSs, quasi-reducibility is equivalent to *sufficient completeness* (cf. [15, 2]). The results in this paper would straightforwardly be extended to *many sorted* systems. Differences to related work are described in Section 9, and proofs of theorems are shown in Appendix B.

The contribution of this paper is to show a method that can prove (1) confluence of $\mathcal{R}_1$, for which all existing confluence provers other than AGES fail to prove confluence, and (2) quasi-reducibility of $\mathcal{R}_1$.

2 Preliminaries

In this section, we recall basic notions and notations of term rewriting [1, 32] and regular tree grammars [3].

Throughout the paper, we use $\mathcal{V}$ as a countably infinite set of *variables*. Let $\mathcal{F}$ be a *signature*, a finite set of *function symbols* f each of which has its own fixed arity. We often write $f/n \in \mathcal{F}$ instead of "an n-ary symbol $f \in \mathcal{F}$", and so on. The set of *terms* over $\mathcal{F}$ and V ($\subseteq \mathcal{V}$) is denoted by $\mathcal{T}(\mathcal{F}, V)$, and $\mathcal{T}(\mathcal{F}, \emptyset)$, the set of *ground* terms, is abbreviated to $\mathcal{T}(\mathcal{F})$. The set of variables appearing in any of terms $t_1, \ldots, t_n$ is denoted by $\mathcal{V}ar(t_1, \ldots, t_n)$. For a term t and a position p of t, the *subterm* of t at p is denoted by $t|_p$. Given terms s, t and a position p of s, we denote by $s[t]_p$ the term obtained from s by replacing the subterm $s|_p$ at p by t.

A *substitution* σ is a mapping from variables to terms such that the number of variables x with $\sigma(x) \neq x$ is finite, and is naturally extended over terms. The *domain* and *range* of σ are denoted by $\mathcal{D}om(\sigma)$ and $\mathcal{R}an(\sigma)$, respectively. The set of variables in $\mathcal{R}an(\sigma)$ is denoted by $\mathcal{VR}an(\sigma)$. We may denote σ by $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ if $\mathcal{D}om(\sigma) = \{x_1, \ldots, x_n\}$ and $\sigma(x_i) = t_i$ for all $1 \leq i \leq n$. The *identity* substitution is denoted by *id*. The set of substitutions that range over a signature $\mathcal{F}$ and a set V of variables is denoted by $\mathit{Subst}(\mathcal{F}, V)$: $\mathit{Subst}(\mathcal{F}, V) = \{\sigma \mid \sigma \text{ is a substitution, } \mathcal{R}an(\sigma) \subseteq \mathcal{T}(\mathcal{F}, V)\}$. The application of a substitution σ to a term t is abbreviated to σt, and σt is called an *instance* of t. Given a set V of variables, $\sigma|_V$ denotes the *restricted* substitution of σ w.r.t. V: $\sigma|_V = \{x \mapsto \sigma x \mid x \in \mathcal{D}om(\sigma) \cap V\}$. A substitution σ is called a *renaming* if σ is a bijection on $\mathcal{V}$. The *composition* $\theta \cdot \sigma$ (simply $\theta\sigma$) of substitutions σ and θ is defined as $(\theta \cdot \sigma)(x) = \theta(\sigma(x))$. A substitution σ is called *idempotent* if $\sigma\sigma = \sigma$ (i.e., $\mathcal{D}om(\sigma) \cap \mathcal{VR}an(\sigma) = \emptyset$). A substitution σ is called *more general than* a substitution θ, written by $\sigma \leq \theta$, if there exists a substitution δ such that $\delta\sigma = \theta$. A finite set E of term equations $s \approx t$ is called *unifiable* if there exists a *unifier* of E such that $\sigma s = \sigma t$ for all term equations $s \approx t$ in E. A *most general unifier* (mgu, for short) of E is denoted by $mgu(E)$ if E is unifiable. Terms s and t are called *unifiable* if $\{s \approx t\}$ is unifiable. The application of a substitution θ to E is defined as $\theta(E) = \{\theta s \approx \theta t \mid s \approx t \in E\}$.

An *oriented conditional rewrite rule* over a signature $\mathcal{F}$ is a triple (ℓ, r, c), denoted by $\ell \to r \Leftarrow c$, such that the *left-hand side* ℓ is a non-variable term in $\mathcal{T}(\mathcal{F}, \mathcal{V})$, the *right-hand* *side* r is a term in $\mathcal{T}(\mathcal{F},\mathcal{V})$, and the *conditional part* c is a sequence $s_1 \twoheadrightarrow t_1, \ldots, s_k \twoheadrightarrow t_k$ of term pairs ($k \geq 0$) where $s_1, t_1, \ldots, s_k, t_k \in \mathcal{T}(\mathcal{F},\mathcal{V})$. In particular, a conditional rewrite rule is called *unconditional* if the conditional part is the empty sequence (i.e., $k = 0$), and we may abbreviate it to $\ell \to r$. Variables in $\mathcal{V}ar(r,c) \setminus \mathcal{V}ar(\ell)$ are called *extra variables* of the rule. An *oriented conditional term rewriting system* (a CTRS, for short) over $\mathcal{F}$ is a set of oriented conditional rewrite rules over $\mathcal{F}$. A CTRS is called an (unconditional) *term rewriting system* (TRS) if every rule $\ell \to r \Leftarrow c$ in the CTRS is unconditional and satisfies $\mathcal{V}ar(\ell) \supseteq \mathcal{V}ar(r)$. The *reduction relation* $\to_\mathcal{R}$ of a CTRS $\mathcal{R}$ is defined as $\to_\mathcal{R} = \bigcup_{n\geq 0} \to_{(n),\mathcal{R}}$, where $\to_{(0),\mathcal{R}} = \emptyset$, and $\to_{(i+1),\mathcal{R}} = \{(s[\sigma\ell]_p, s[\sigma r]_p) \mid s \in \mathcal{T}(\mathcal{F},\mathcal{V}),\ \ell \to r \Leftarrow s_1 \twoheadrightarrow t_1, \ldots, s_k \twoheadrightarrow t_k \in \mathcal{R},\ \sigma s_1 \to^*_{(i),\mathcal{R}} \sigma t_1,\ \ldots,\ \sigma s_k \to^*_{(i),\mathcal{R}} \sigma t_k\}$ for $i \geq 0$. To specify the position where the rule is applied, we may write $\to_{p,\mathcal{R}}$ instead of $\to_\mathcal{R}$. The *underlying unconditional system* $\{\ell \to r \mid \ell \to r \Leftarrow c \in \mathcal{R}\}$ of $\mathcal{R}$ is denoted by $\mathcal{R}_u$. A term t is called a *normal form* (of $\mathcal{R}$) if t is irreducible w.r.t. $\mathcal{R}$. A substitution σ is called *normalized* (w.r.t. $\mathcal{R}$) if σx is a normal form of $\mathcal{R}$ for each variable $x \in \mathcal{D}om(\sigma)$. A CTRS $\mathcal{R}$ is called *Type 1* (1-CTRS, for short) if every rule $\ell \to r \Leftarrow c \in \mathcal{R}$ satisfies that $\mathcal{V}ar(r,c) \subseteq \mathcal{V}ar(\ell)$; *Type 3* (3-CTRS, for short) if every rule $\ell \to r \Leftarrow c \in \mathcal{R}$ satisfies that $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(\ell,c)$; *normal* if for every rule $\ell \to r \Leftarrow s_1 \twoheadrightarrow t_1, \ldots, s_k \twoheadrightarrow t_k \in \mathcal{R}$, all $t_1, \ldots, t_k$ are ground normal forms of $\mathcal{R}_u$; *deterministic* (a DCTRS, for short) if, for every rule $\ell \to r \Leftarrow s_1 \twoheadrightarrow t_1, \ldots, s_k \twoheadrightarrow t_k \in \mathcal{R}$, $\mathcal{V}ar(s_i) \subseteq \mathcal{V}ar(\ell, t_1, \ldots, t_{i-1})$ for all $1 \leq i \leq k$.

The sets of *defined symbols* and *constructors* of a CTRS $\mathcal{R}$ over a signature $\mathcal{F}$ are denoted by $\mathcal{D}_\mathcal{R}$ and $\mathcal{C}_\mathcal{R}$, respectively: $\mathcal{D}_\mathcal{R} = \{f \mid f(u_1,\ldots,u_n) \to r \Leftarrow c \in \mathcal{R}\}$ and $\mathcal{C}_\mathcal{R} = \mathcal{F} \setminus \mathcal{D}_\mathcal{R}$. Terms in $\mathcal{T}(\mathcal{C}_\mathcal{R},\mathcal{V})$ are called *constructor terms of* $\mathcal{R}$. A substitution in $\mathcal{S}ubst(\mathcal{C}_\mathcal{R},\mathcal{V})$ is called a *constructor substitution of* $\mathcal{R}$. A term of the form $f(t_1,\ldots,t_n)$ with $f/n \in \mathcal{D}_\mathcal{R}$ and $t_1,\ldots,t_n \in \mathcal{T}(\mathcal{C}_\mathcal{R},\mathcal{V})$ is called *basic*. A CTRS $\mathcal{R}$ is called a *constructor system* if for every rule $\ell \to r \Leftarrow c$ in $\mathcal{R}$, ℓ is basic. A CTRS $\mathcal{R}$ is called a *pure-constructor system* (a pc-CTRS, for short) if for every rule $\ell \to r \Leftarrow s_1 \twoheadrightarrow t_1, \ldots, s_k \twoheadrightarrow t_k \in \mathcal{R}$, all of $\ell, s_1, \ldots, s_k$ are basic and all of $r, t_1, \ldots, t_k$ are constructor terms [24]. A 3-DCTRS $\mathcal{R}$ is called *syntactically deterministic* (an SDCTRS, for short) if for every rule $\ell \to r \Leftarrow s_1 \twoheadrightarrow t_1, \ldots, s_k \twoheadrightarrow t_k \in \mathcal{R}$, every t_i is a constructor term or a ground normal form of $\mathcal{R}_u$.

A CTRS $\mathcal{R}$ is called *operationally terminating* if there are no infinite well-formed trees in a certain logical inference system [21] – operational termination means that the evaluation of conditions must either successfully terminate or fail in finite time. Two terms s and t are said to be *joinable*, written as $s \downarrow_\mathcal{R} t$, if there exists a term u such that $s \to^*_\mathcal{R} u \leftarrow^*_\mathcal{R} t$. A CTRS $\mathcal{R}$ is called *confluent* if $t_1 \downarrow_\mathcal{R} t_2$ for any terms t_1, t_2 with $t_1 \leftarrow^*_\mathcal{R} \cdot \to^*_\mathcal{R} t_2$.

A *regular tree grammar* is a quadruple $\mathcal{G} = (S, \mathcal{N}, \mathcal{F}, \mathcal{P})$ such that $\mathcal{F}$ is a signature, $\mathcal{N}$ is a finite set of *non-terminals* (constants not in $\mathcal{F}$), $S \in \mathcal{N}$, and $\mathcal{P}$ is a finite set of *production rules* of the form $A \to \beta$ with $A \in \mathcal{N}$ and $\beta \in \mathcal{T}(\mathcal{F} \cup \mathcal{N})$. Note that $A \to \beta_1 \mid \ldots \mid \beta_n$ stands for $A \to \beta_1, \ldots, A \to \beta_n$. Given a non-terminal $S' \in \mathcal{N}$, the set $\{t \in \mathcal{T}(\mathcal{F}) \mid S' \to^*_\mathcal{P} t\}$ is the *language generated by* $\mathcal{G}$ *from* S', denoted by $L(\mathcal{G}, S')$. The *initial* non-terminal S does not play an important role in this paper. A *regular tree language* is a language generated by a regular tree grammar from one of its non-terminals. The class of regular tree languages is equivalent to the class of *recognizable tree languages* which are recognized by *tree automata*. This means that the *intersection (non-)emptiness problem* for regular tree languages is decidable.

▶ **Example 1.** The regular tree grammar $\mathcal{G}_1 = (X, \{X, X'\}, \{0/0, \mathsf{s}/1\}, \{\, X \to 0 \mid \mathsf{s}(X'),\ X' \to \mathsf{s}(X) \,\})$ generates the sets of even and odd numbers over 0 and s from X and X', respectively: $L(\mathcal{G}_1, X) = \{\mathsf{s}^{2n}(0) \mid n \geq 0\}$ $(= L(\mathcal{G}_1))$ and $L(\mathcal{G}_1, X') = \{\mathsf{s}^{2n+1}(0) \mid n \geq 0\}$.

3 From Constructor SDCTRSs to Unconditional Constructor Systems

In this section, we show that every constructor SDCTRS can be converted to an equivalent unconditional constructor system w.r.t. *constructor-based rewriting* and *innermost narrowing* for *goal clauses*. Since SDCTRSs possibly have extra variables, we relax the requirement "$\mathcal{V}ar(\ell) \supseteq \mathcal{V}ar(r)$" for TRSs, i.e., we allow unconditional rules to have extra variables.

We denote a pair of terms s, t by $s \twoheadrightarrow t$ (not an equation $s \approx t$) because we analyze conditions of rewrite rules and distinguish the left- and right-hand sides of the pair $s \twoheadrightarrow t$. We deal with pairs of terms as terms by considering $\twoheadrightarrow$ a binary function symbol. For this reason, we apply many notions for terms to pairs of terms without notice. For readability, when we deal with $s \twoheadrightarrow t$ as a term, we often put it in parentheses: $(s \twoheadrightarrow t)$. As in [23], we assume that any CTRS in this paper implicitly includes the rule $(x \twoheadrightarrow x) \to \top$ where $\top$ is a special constant. The rule $(x \twoheadrightarrow x) \to \top$ is used to test equivalence between two terms t_1, t_2 via $t_1 \twoheadrightarrow t_2$. A pair $s \twoheadrightarrow t$ of terms s, t is called a *goal* of a constructor SDCTRS $\mathcal{R}$ if the left-hand side s is either a constructor term or a basic term and the right-hand side t is a constructor term.

To deal with a conjunction of pairs $e_1, \ldots, e_k$ of terms (e_i is either $s_i \twoheadrightarrow t_i$ or $\top$) as a term, we write $e_1 \mathbin{\&} \cdots \mathbin{\&} e_k$ by using an associative binary symbol &. We call such a term an *equational term*. Unlike [29], to avoid & to be a defined symbol, we do not use any rule for &, e.g., $(\top \mathbin{\&} x) \to x$. Instead of derivations ending with $\top$, we consider derivations that end with terms in $\mathcal{T}(\{\top, \&\})$. We assume that none of &, $\twoheadrightarrow$, or $\top$ is included in the range of any substitution below. In the following, we denote conditional parts of rules by equational terms, e.g., $\ell \to r \Leftarrow s_1 \twoheadrightarrow t_1 \mathbin{\&} \cdots \mathbin{\&} s_k \twoheadrightarrow t_k$. Note that the empty sequence of a conditional part is denoted by $\top$. An equational term is called a *goal clause* of a constructor SDCTRS $\mathcal{R}$ if it is a conjunction of goals for $\mathcal{R}$. Note that for a goal clause T, any instance θT with θ a constructor substitution is a goal clause.

▶ **Example 2.** The equational term $\mathsf{e}(x) \twoheadrightarrow \mathsf{true} \mathbin{\&} \mathsf{o}(x) \twoheadrightarrow \mathsf{true}$ is a goal clause of $\mathcal{R}_1$.

3.1 Constructor-based Rewriting and Innermost Narrowing

Following [28], we define *constructor-based conditional rewriting* on goal clauses as follows: for a goal clause $S = U \mathbin{\&} s \twoheadrightarrow t \mathbin{\&} S'$ with $U \in \mathcal{T}(\{\top, \&\})$, we write $S \xrightarrow{c}_{\mathcal{R}} T$ if there exist a non-variable position p of $(s \twoheadrightarrow t)$, a rule $\ell \to r \Leftarrow C$ in $\mathcal{R}$, and a constructor substitution σ such that $(s \twoheadrightarrow t)|_p$ is basic, $(s \twoheadrightarrow t)|_p = \sigma\ell$, and $T = U \mathbin{\&} \sigma C \mathbin{\&} (s \twoheadrightarrow t)[\sigma r]_p \mathbin{\&} S'$. The constructor-based rewriting under the leftmost strategy is denoted by $\xrightarrow{lc}_{\mathcal{R}}$. It is clear that for a goal clause S and a normal form T of $\mathcal{R}$, $S \xrightarrow{c}{}^{*}_{\mathcal{R}} T$ if and only if $S \xrightarrow{lc}{}^{*}_{\mathcal{R}} T$.

The *narrowing* relation [34, 14] mainly extends rewriting by replacing matching with unification. This paper follows the formalization in [28], while we use the rule $(x \twoheadrightarrow x) \to \top$ instead of the corresponding inference rule.

▶ **Definition 3** (innermost narrowing)**.** Let $\mathcal{R}$ be a CTRS. A goal clause $S = U \mathbin{\&} s \twoheadrightarrow t \mathbin{\&} S'$ with $U \in \mathcal{T}(\{\top, \&\})$ is said to *conditionally narrow* into an equational term T at an innermost position, written as $S \overset{i}{\rightsquigarrow}_{\mathcal{R}} T$, if there exist a non-variable position p of $(s \twoheadrightarrow t)$, a variant $\ell \to r \Leftarrow C$ of a rule in $\mathcal{R}$, and a constructor substitution σ such that $\mathcal{V}ar(\ell, r, C) \cap \mathcal{V}ar(S) = \emptyset$, $(s \twoheadrightarrow t)|_p$ is basic, $(s \twoheadrightarrow t)|_p$ and ℓ are unifiable, $\sigma = mgu(\{(s \twoheadrightarrow t)|_p \approx \ell\})$, and $T = U \mathbin{\&} \sigma C \mathbin{\&} \sigma((s \twoheadrightarrow t)[r]_p) \mathbin{\&} \sigma S'$. Note that all extra variables of $\ell \to r \Leftarrow C$ remain in T as *fresh* variables which do not appear in S. We assume that $\mathcal{V}ar(S) \cap \mathcal{VR}an(\sigma|_{\mathcal{V}ar((s \twoheadrightarrow t)|_p)}) = \emptyset$ (i.e., $\sigma|_{\mathcal{V}ar((s \twoheadrightarrow t)|_p)}$ is idempotent) and $\mathcal{V}ar((s \twoheadrightarrow t)|_p) \subseteq \mathcal{D}om(\sigma)$. We write $S \overset{li}{\rightsquigarrow}_{\mathcal{R}} T$ if p is the leftmost among innermost narrowable positions in $(s \twoheadrightarrow t)$. We write $S \overset{i}{\rightsquigarrow}_{\sigma|_{\mathcal{V}ar(S)}, \mathcal{R}} T$ to make the substitution explicit.

An example of innermost narrowing and constructor-based rewriting can be seen in Appendix A. Let $\overset{x}{\leadsto}_{\mathcal{R}}$ be either $\overset{i}{\leadsto}_{\mathcal{R}}$ or $\overset{li}{\leadsto}_{\mathcal{R}}$. An *innermost narrowing derivation* $T_0 \overset{x}{\leadsto}^*_{\sigma,\mathcal{R}} T_n$ (and $T_0 \overset{x}{\leadsto}^n_{\sigma,\mathcal{R}} T_n$) denotes a sequence of narrowing steps $T_0 \overset{x}{\leadsto}_{\sigma_1,\mathcal{R}} \cdots \overset{x}{\leadsto}_{\sigma_n,\mathcal{R}} T_n$ with $\sigma = (\sigma_n \cdots \sigma_1)|_{\mathcal{V}ar(T_0)}$ an idempotent substitution. When we consider two (or more) narrowing derivations $S_1 \overset{x}{\leadsto}^*_{\sigma_1,\mathcal{R}} T_1$ and $S_2 \overset{x}{\leadsto}^*_{\sigma_2,\mathcal{R}} T_2$, we assume that $\mathcal{VR}an(\sigma_1) \cap \mathcal{VR}an(\sigma_2) = \emptyset$.

As in [29], for the sake of simplicity, we first consider the leftmost innermost narrowing. After showing basic properties of compositionality, we drop this restriction (see Theorem 14).

Constructor-based rewriting and innermost narrowing of constructor SDCTRSs have the following relationships (cf. [28]).

▶ **Theorem 4.** *Let $\mathcal{R}$ be a constructor SDCTRS, T a goal clause, and $U \in \mathcal{T}(\{\top, \&\})$.*

1. *If $T \overset{li}{\leadsto}^*_{\sigma,\mathcal{R}} U$, then $\sigma T \overset{lc}{\to}^*_{\mathcal{R}} U$ (i.e., $\sigma s \overset{lc}{\to}^*_{\mathcal{R}} \sigma t$ for all goals $s \twoheadrightarrow t$ in T).*
2. *For a constructor substitution θ, if $\theta T \overset{lc}{\to}^*_{\mathcal{R}} U$, then there exists an idempotent constructor substitution σ such that $T \overset{li}{\leadsto}^*_{\sigma,\mathcal{R}} U$ and $\sigma \leq \theta$.*

3.2 Converting to Unconditional Constructor Systems

We say that a constructor SDCTRS $\mathcal{R}$ over a signature $\mathcal{F}$ is *equivalent* to a constructor SDCTRS $\mathcal{R}'$ over $\mathcal{F}$ *w.r.t.* $\overset{c}{\to}$ *and* $\overset{i}{\leadsto}$ if $\mathcal{D}_{\mathcal{R}} = \mathcal{D}_{\mathcal{R}'}$ and both of the following hold:

- for any goal clause T, $T \overset{lc}{\to}^*_{\mathcal{R}} U$ for some term $U \in \mathcal{T}(\{\top, \&\})$ if and only if $T \overset{lc}{\to}^*_{\mathcal{R}'} U'$ for some term $U' \in \mathcal{T}(\{\top, \&\})$, and
- for any goal clause T, $T \overset{li}{\leadsto}^*_{\theta,\mathcal{R}} U$ for some term $U \in \mathcal{T}(\{\top, \&\})$ if and only if $T \overset{li}{\leadsto}^*_{\theta,\mathcal{R}'} U'$ for some term $U' \in \mathcal{T}(\{\top, \&\})$.

Note that $\mathcal{C}_{\mathcal{R}} = \mathcal{C}_{\mathcal{R}'}$. In this section, we first convert a constructor SDCTRS to a pc-CTRS that is equivalent to the SDCTRS w.r.t. $\overset{c}{\to}$ and $\overset{i}{\leadsto}$, and then convert the pc-CTRS to a constructor TRS that is equivalent to the pc-CTRS w.r.t. $\overset{c}{\to}$ and $\overset{i}{\leadsto}$.

To convert a constructor SDCTRS $\mathcal{R}$, we adopt a stepwise transformation for each rule $\ell \to r \Leftarrow C \in \mathcal{R}$ as follows (cf. [26, Definition 23]).

▶ **Definition 5.** We transform each rule $\ell \to r \Leftarrow C$ of a constructor SDCTRS $\mathcal{R}$ as follows:

1. We replace r and C by a fresh variable y and $C, r \twoheadrightarrow y$, respectively, if $r \notin \mathcal{T}(\mathcal{C}_{\mathcal{R}}, \mathcal{V})$.
2. For each condition $s \twoheadrightarrow t$ in the resulting conditional part, if t contains a defined symbol, then we replace $s \twoheadrightarrow t$ by $(s \twoheadrightarrow x)$ & $(t \twoheadrightarrow x)$, where x is a fresh variable.[6]
3. We remove all nests of defined symbols in the resulting conditional part by replacing a condition $s[f(u_1, \ldots, u_n)]_p \twoheadrightarrow t$ with $(f(u_1, \ldots, u_n) \twoheadrightarrow x)$ & $(s[x]_p \twoheadrightarrow t)$, where f is a defined symbol, $p > \varepsilon$, and x is a fresh variable that does not appear in the intermediate rule. This operation is so-called a *flattening* [29] shown in Section 4.
4. If the resulting rule has a condition $s \twoheadrightarrow t$ with s, t constructor terms, then (1) we drop the rule from $\mathcal{R}$ whenever s and t are not unifiable, and (2) otherwise, we drop the condition $s \twoheadrightarrow t$ by applying an *mgu* of s, t to the rule [27, p. 292] (see also [26, Theorem 26]).

We denote the resulting CTRS by $Pc(\mathcal{R})$.

[6] If C contains a condition $s \twoheadrightarrow t$ such that t contains a defined symbol, then rule $\ell \to r \Leftarrow C$ is never used in constructor-based rewriting of goal clauses to terms in $\mathcal{T}(\{\top, \&\})$ because t is not a constructor term and any instance of $s \twoheadrightarrow t$ is never reduced to any term in $\mathcal{T}(\{\top, \&\})$. However, we do not drop the rule from $\mathcal{R}$ because defined symbols are preserved during the conversion and the rule can be used for the standard rewriting $\to_{\mathcal{R}}$.

▶ **Theorem 6.** *Let $\mathcal{R}$ be a constructor SDCTRS over a signature $\mathcal{F}$. Then, $Pc(\mathcal{R})$ is a pc-CTRS over $\mathcal{F}$ and is equivalent to $\mathcal{R}$ w.r.t. $\xrightarrow{c}$ and $\overset{i}{\leadsto}$.*

Let $\mathcal{R}$ be a pc-CTRS over $\mathcal{F}$. We denote the TRS $\{(\ell \twoheadrightarrow y) \to C \mathrel{\&} (r \twoheadrightarrow y) \mid \ell \to r \Leftarrow C \in \mathcal{R},\ y \in \mathcal{V} \setminus \mathcal{V}ar(\ell, r, C)\}$ by $\mathit{Trs}(\mathcal{R})$. Since the conditional part is a goal clause, the generated right-hand side $C \mathrel{\&} (r \twoheadrightarrow y)$ is a goal clause. Thus, for a goal clause T, if $T \xrightarrow{c}_{\mathcal{R}} T'$ or $T \overset{i}{\leadsto}_{\mathcal{R}} T'$, then T' is a goal clause. It is clear that $\mathit{Trs}(\mathcal{R})$ is a constructor TRS, $\mathcal{D}_{\mathit{Trs}(\mathcal{R})} = \{\twoheadrightarrow\}$, and $\mathcal{C}_{\mathit{Trs}(\mathcal{R})} = \mathcal{F} \cup \{\top, \&\}$.

▶ **Theorem 7.** *Let $\mathcal{R}$ be a pc-CTRS over a signature $\mathcal{F}$. Then, $\mathit{Trs}(\mathcal{R})$ is a constructor TRS over $\mathcal{F} \cup \{\twoheadrightarrow, \top, \&\}$ and is equivalent to $\mathcal{R}$ w.r.t. $\xrightarrow{c}$ and $\overset{i}{\leadsto}$.*

▶ **Example 8.** For $\mathcal{R}_1$ in Section 1, we obtain the following TRS by applying $\mathit{Trs}(\cdot)$ to $\mathcal{R}_1$:

$$\mathit{Trs}(\mathcal{R}_1) = \left\{ \begin{array}{lll} (\mathsf{e}(\mathsf{0}) \twoheadrightarrow y) \to (\mathsf{true} \twoheadrightarrow y), & (\mathsf{e}(\mathsf{s}(x)) \twoheadrightarrow y) \to (\mathsf{o}(x) \twoheadrightarrow \mathsf{true}) \mathrel{\&} (\mathsf{true} \twoheadrightarrow y), \\ & (\mathsf{e}(\mathsf{s}(x)) \twoheadrightarrow y) \to (\mathsf{e}(x) \twoheadrightarrow \mathsf{true}) \mathrel{\&} (\mathsf{false} \twoheadrightarrow y), \\ (\mathsf{o}(\mathsf{0}) \twoheadrightarrow y) \to (\mathsf{false} \twoheadrightarrow y), & (\mathsf{o}(\mathsf{s}(x)) \twoheadrightarrow y) \to (\mathsf{e}(x) \twoheadrightarrow \mathsf{true}) \mathrel{\&} (\mathsf{true} \twoheadrightarrow y), \\ & (\mathsf{o}(\mathsf{s}(x)) \twoheadrightarrow y) \to (\mathsf{o}(x) \twoheadrightarrow \mathsf{true}) \mathrel{\&} (\mathsf{false} \twoheadrightarrow y) \end{array} \right\}$$

For example, the following narrowing derivation holds for both $\mathcal{R}_1$ and $\mathit{Trs}(\mathcal{R}_1)$: $(\mathsf{e}(x) \twoheadrightarrow \mathsf{true}) \overset{li}{\leadsto}_{\{x \mapsto \mathsf{s}(x_1)\}} (\mathsf{o}(x_1) \twoheadrightarrow \mathsf{true}) \mathrel{\&} (\mathsf{true} \twoheadrightarrow \mathsf{true}) \overset{li}{\leadsto}_{\{x_1 \mapsto \mathsf{0}\}} (\mathsf{false} \twoheadrightarrow \mathsf{true}) \mathrel{\&} (\mathsf{true} \twoheadrightarrow \mathsf{true})$.

As a consequence of Theorems 6 and 7, we obtain the following corollary.

▶ **Corollary 9.** *Let $\mathcal{R}$ be a constructor SDCTRS over a signature $\mathcal{F}$. Then, $\mathit{Trs}(Pc(\mathcal{R}))$ is a constructor TRS over $\mathcal{F} \cup \{\twoheadrightarrow, \top, \&\}$ and is equivalent to $\mathcal{R}$ w.r.t. $\xrightarrow{c}$ and $\overset{i}{\leadsto}$.*

4 Compositionality of Innermost Narrowing

Compositionality of innermost narrowing under *parallel composition* of idempotent substitutions is a key to ensure equivalence between substitutions obtained at innermost-narrowing steps and those defined by grammar representations of narrowing trees. In this section, we recall parallel composition, and revisit compositionality of innermost narrowing for TRSs. Since the counterpart of $\overset{i}{\leadsto}_{\mathcal{R}}$ is constructor-based rewriting $\xrightarrow{c}_{\mathcal{R}}$, sufficient completeness is required in [29] to have $\xrightarrow{c}_{\mathcal{R}} = \xrightarrow{i}_{\mathcal{R}}$ on ground terms. However, sufficient completeness is not necessary for compositionality and we do not force this property to TRSs.

We first recall *parallel composition* $\Uparrow$ of idempotent substitutions [13, 33], which is one of the most important key operations to enable us to construct *finite* narrowing trees. Given a substitution $\theta = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, we denote the set of term equations $\{x_1 \approx t_1, \ldots, x_n \approx t_n\}$ by $\widehat{\theta}$.

▶ **Definition 10** (parallel composition $\Uparrow$ [33])**.** Let θ_1 and θ_2 be idempotent substitutions. Then, we define $\Uparrow$ as follows: $\theta_1 \Uparrow \theta_2 = \mathit{mgu}(\widehat{\theta_1} \cup \widehat{\theta_2})$ if $\widehat{\theta_1} \mathrel{\&} \widehat{\theta_2}$ is unifiable, and otherwise, $\theta_1 \Uparrow \theta_2 = \mathit{fail}$. Note that we define $\theta_1 \Uparrow \theta_2 = \mathit{fail}$ if θ_1 or θ_2 is not idempotent. Parallel composition is extended to sets Θ_1, Θ_2 of idempotent substitutions in the natural way: $\Theta_1 \Uparrow \Theta_2 = \{\theta_1 \Uparrow \theta_2 \mid \theta_1 \in \Theta_1,\ \theta_2 \in \Theta_2,\ \theta_1 \Uparrow \theta_2 \neq \mathit{fail}\}$.

We often have two or more substitutions that can be results of $\theta_1 \Uparrow \theta_2$ ($\neq \mathit{fail}$), while they are unique up to variable renaming. To simplify the semantics of grammar representations for substitutions, we adopt an idempotent substitution σ with $\mathcal{D}om(\theta_1) \cup \mathcal{D}om(\theta_2) \subseteq \mathcal{D}om(\sigma)$ as a result of $\theta_1 \Uparrow \theta_2$ ($\neq \mathit{fail}$). Idempotent substitutions we can adopt as results of $\theta_1 \Uparrow \theta_2$ under the convention are unique up to variable renaming, but not exactly unique in general.

▶ **Example 11.** The parallel composition $\{x \mapsto \mathsf{s}(z),\, y \mapsto z\} \Uparrow \{x \mapsto w\}$ may return $\{x \mapsto \mathsf{s}(z),\, y \mapsto z,\, w \mapsto \mathsf{s}(z)\}$, but we do not allow $\{x \mapsto \mathsf{s}(y),\, z \mapsto y,\, w \mapsto \mathsf{s}(y)\}$ as a result. On the other hand, $\{x \mapsto \mathsf{s}(z),\, y \mapsto z\} \Uparrow \{x \mapsto y\}$ fails.

Let $\overset{x}{\leadsto}_{\mathcal{R}}$ be either $\overset{i}{\leadsto}_{\mathcal{R}}$ or $\overset{li}{\leadsto}_{\mathcal{R}}$. For a constructor SDCTRS $\mathcal{R}$ and a goal clause T, we define the *success set* of T (w.r.t. $\overset{x}{\leadsto}_{\mathcal{R}}$), which is the set of *successful* substitutions derived by $\overset{x}{\leadsto}_{\mathcal{R}}$, as follows: $\mathit{Suc}(\overset{x}{\leadsto}_{\mathcal{R}}, T) = \{\theta \mid \exists U \in \mathcal{T}(\{\top, \&\}).\ T \overset{x}{\leadsto}{}^{*}_{\theta,\mathcal{R}} U\}$. Note that $\mathcal{D}om(\theta) \subseteq \mathcal{V}ar(T)$ for any substitution $\theta \in \mathit{Suc}(\overset{x}{\leadsto}_{\mathcal{R}}, T)$. We extend the restriction of substitutions to sets of substitutions: $\Theta|_V = \{\theta|_V \mid \theta \in \Theta\}$.

▶ **Theorem 12** (compositionality [29]). *For a constructor TRS $\mathcal{R}$ and goal clauses T_1, T_2, $\mathit{Suc}(\overset{li}{\leadsto}_{\mathcal{R}}, T_1 \,\&\, T_2) = \Big(\mathit{Suc}(\overset{li}{\leadsto}_{\mathcal{R}}, T_1) \Uparrow \mathit{Suc}(\overset{li}{\leadsto}_{\mathcal{R}}, T_2)\Big)|_{\mathcal{V}ar(T_1,T_2)}$ up to variable renaming.*

Note that & is just a binary symbol to construct conjunctions of goals, and $\Uparrow$ is a binary operator for parallel composition. In Theorem 12, we restrict $\mathit{Suc}(\overset{li}{\leadsto}_{\mathcal{R}}, T_1) \Uparrow \mathit{Suc}(\overset{li}{\leadsto}_{\mathcal{R}}, T_2)$ to $\mathcal{V}ar(T_1, T_2)$ because parallel composition may make the domain of a resulting substitution in $\mathit{Suc}(\overset{li}{\leadsto}_{\mathcal{R}}, T_1) \Uparrow \mathit{Suc}(\overset{li}{\leadsto}_{\mathcal{R}}, T_2)$ include a variable that does not appear in $T_1 \,\&\, T_2$. Theorem 12 enables us to, given $T_1 \,\&\, T_2$, compute $\mathit{Suc}(\overset{li}{\leadsto}_{\mathcal{R}}, T_1)$ and $\mathit{Suc}(\overset{li}{\leadsto}_{\mathcal{R}}, T_2)$ separately, so-called *splitting*, instead of computing $\mathit{Suc}(\overset{li}{\leadsto}_{\mathcal{R}}, T_1 \,\&\, T_2)$, and then apply parallel composition to them under the variable restriction to $\mathcal{V}ar(T_1, T_2)$.

Let T be an equational term, p a position of T such that the root symbol of $T|_p$ is none of $\twoheadrightarrow$, $\top$, and &. A *flattening* of T w.r.t. p is given by $T[x]_p \,\&\, (T|_p \twoheadrightarrow x)$ where x is a fresh variable [29]. Note that $T|_p$ may be a variable, but, to avoid any redundant replacement, we allow $T|_p$ to be a variable only if the replacement is in the process of linearizing a basic term in T. Thanks to Theorem 12, we can use flattening in computing the success set of T.

▶ **Theorem 13** (flattening [29]). *Let $\mathcal{R}$ be a constructor TRS, T a goal clause, and T' a flattening of T w.r.t. a position p of T. Then, $\mathit{Suc}(\overset{li}{\leadsto}_{\mathcal{R}}, T) = (\mathit{Suc}(\overset{li}{\leadsto}_{\mathcal{R}}, T'))|_{\mathcal{V}ar(T)}$ up to variable renaming.*

As in Theorem 12, we restrict $\mathit{Suc}(\overset{li}{\leadsto}_{\mathcal{R}}, T')$ to $\mathcal{V}ar(T)$ in Theorem 13 because a variable in T' but not in T may appear in the domain of a substitution in $\mathit{Suc}(\overset{li}{\leadsto}_{\mathcal{R}}, T')$, but not in T.

Thanks to Theorems 12 and 13, we can show that innermost-narrowing steps to a ground normal form in $\mathcal{T}(\{\top, \&\})$ can be replaced by leftmost ones.

▶ **Theorem 14** ([29]). *Let $\mathcal{R}$ be a constructor TRS and T a goal clause. Then, $\mathit{Suc}(\overset{i}{\leadsto}_{\mathcal{R}}, T) = \mathit{Suc}(\overset{li}{\leadsto}_{\mathcal{R}}, T)$ up to variable renaming.*

Thanks to Theorem 14, both Theorems 12 and 13 hold for $\overset{i}{\leadsto}_{\mathcal{R}}$. To make the proof of Theorem 13 simpler, we adopt $T[x]_p \,\&\, T|_p \twoheadrightarrow x$ as a result of flattening. However, thanks to Theorem 14, we may adopt $T|_p \twoheadrightarrow x \,\&\, T[x]_p$ as a result of flattening.

As mentioned above, in [29], $\mathcal{R}$ is restricted to a sufficiently complete constructor TRS without extra variables. However, sufficient completeness is not used for proving Theorems 12, 13, and 14, and the existence of extra variables does not affect the proofs of Theorems 12, 13, and 14. For this reason, Theorems 12, 13, and 14 hold for constructor TRSs with extra variables.

5 Grammar Representation for Sets of Idempotent Substitutions

In this section, we formalize grammar representations that define sets of idempotent substitutions. Since substitutions derived by narrowing steps are assumed to be idempotent, we only deal with idempotent substitutions which introduce only *fresh* variables not appearing in any previous term. The formalization here is based on *success set equations* in [29].

In the following, a renaming δ is used to rename a particular term t and we assume that $\delta|_{\mathcal{V}ar(t)}$ is injective on $\mathcal{V}ar(t)$. For this reason, as described in Footnote 4, we write $\delta|_{\mathcal{V}ar(t)}$ instead of δ, and call $\delta|_{\mathcal{V}ar(t)}$ a *renaming for t* (simply, a renaming).

We first introduce terms to represent idempotent substitutions computed using $\cdot$ and $\Uparrow$. We prepare the signature Σ consisting of the following symbols:

- idempotent substitutions which are considered constants, (basic elements)
- a constant $\varnothing$, (the empty set/non-existence)
- an associative binary symbol $\bullet$, (standard composition)
- an associative binary symbol &, and (parallel composition)
- a binary symbol REC. (recursion with renaming)

We use infix notation for $\bullet$ and &, and may omit parentheses with the precedence such that $\bullet$ has a higher priority than &.

We deal with terms over Σ and some constants which are used as non-terminals of grammar representations, where we allow such constants to only appear in the first argument of REC. Note that a term without any constant may appear in the first argument of REC. Given a finite set $\mathcal{N}$ of constants, we denote the set of such terms by $\mathcal{T}(\Sigma \cup \mathcal{N})$. We assume that each constant in $\mathcal{N}$ has a term t (possibly a goal clause) as subscript such as Γ_t. For an expression $\textsc{rec}(\Gamma_t, \delta)$, the role of Γ_t is recursion to generate terms in $\mathcal{T}(\Sigma)$. To reuse substitutions generated by recursion, we connect them with other substitutions via some renaming δ. For this reason, we restrict the second argument of REC to renamings and we require each term $\textsc{rec}(\Gamma_t, \delta)$ to satisfy $\mathcal{VR}an(\delta) = \mathcal{V}ar(t)$.

▶ **Example 15.** The following are instances of terms in $\mathcal{T}(\Sigma)$: $\{y \mapsto 0\} \bullet \{x \mapsto \mathsf{s}(y)\}$, $(\{x' \mapsto \mathsf{s}(y)\} \bullet \{x \mapsto x'\})$ & $\{x \mapsto \mathsf{s}(\mathsf{s}(z))\}$, $(\varnothing$ & $\{y \mapsto z\}) \bullet \{x \mapsto \mathsf{s}(y)\}$, and $\textsc{rec}(\{x \mapsto 0,\, y \mapsto \mathsf{s}(y')\}, \{x' \mapsto x,\, y' \mapsto y\}) \bullet \{y \mapsto \mathsf{s}(x')\}$.

As described in Section 3, in computing $\sigma_1 \Uparrow \sigma_2$ from two narrowing derivations $S_1 \overset{i}{\rightsquigarrow}{}^*_{\sigma_1,\mathcal{R}} T_1$ and $S_2 \overset{i}{\rightsquigarrow}{}^*_{\sigma_2,\mathcal{R}} T_2$, we assume that $\mathcal{VR}an(\sigma_1) \cap \mathcal{VR}an(\sigma_2) = \emptyset$. To satisfy this assumption explicitly in the semantics for $\mathcal{T}(\Sigma)$, we introduce an operation $\mathit{fresh}_\delta(\cdot)$ of substitutions to make a substitution introduce only variables that do not appear in $\mathcal{D}om(\delta) \cup \mathcal{VR}an(\delta)$: for substitutions σ, δ, we define $\mathit{fresh}_\delta(\sigma)$ by $(\xi \cdot \sigma)|_{\mathcal{D}om(\sigma)}$ where ξ is a renaming such that $\mathcal{D}om(\xi) \supseteq \mathcal{VR}an(\sigma)$ and $\mathcal{VR}an(\xi|_{\mathcal{VR}an(\sigma)}) \cap (\mathcal{D}om(\delta) \cup \mathcal{VR}an(\delta) \cup \mathcal{D}om(\sigma)) = \emptyset$. The subscript δ of $\mathit{fresh}_\delta(\cdot)$ is used to specify freshness of variables. We say that a variable x is *fresh* w.r.t. a set X of variables if $x \notin X$.

The semantics of terms in $\mathcal{T}(\Sigma)$ to define substitutions is inductively defined as follows:

- $[\![\, \theta \,]\!] = \theta$ if θ is a substitution,
- $[\![\, e_1 \bullet e_2 \,]\!] = [\![\, e_1 \,]\!] \cdot [\![\, e_2 \,]\!]$ if $[\![\, e_2 \,]\!] \neq \mathit{fail}$ and $[\![\, e_1 \,]\!] \neq \mathit{fail}$,
- $[\![\, e_1 \mathbin{\&} e_2 \,]\!] = (\theta_1 \Uparrow \theta_2)|_{\mathcal{D}om(\theta_1) \cup \mathcal{D}om(\theta_2)}$ if $[\![\, e_1 \,]\!] \neq \mathit{fail}$ and $[\![\, e_2 \,]\!] \neq \mathit{fail}$, where $\theta_1 = [\![\, e_1 \,]\!]$ and $\theta_2 = \mathit{fresh}_{\theta_1}([\![\, e_2 \,]\!])$,
- $[\![\, \textsc{rec}(e, \delta) \,]\!] = (\mathit{fresh}_\delta([\![\, e \,]\!]) \cdot \delta)|_{\mathcal{D}om(\delta)}$ if $[\![\, e \,]\!] \neq \mathit{fail}$ and $\mathcal{VR}an(\delta) = \mathcal{D}om([\![\, e \,]\!])$,
- otherwise, $[\![\, e \,]\!] = \mathit{fail}$ (e.g., $[\![\, \varnothing \,]\!] = \mathit{fail}$).

Notice that a constant Γ_t is not included in $\mathcal{T}(\Sigma)$, and thus, $[\![\, \Gamma_t \,]\!]$ is not defined above. Since $\Uparrow$ may fail, we allow to have *fail*, e.g., $[\![\, \{y \mapsto \mathsf{s}(z)\} \bullet \{x \mapsto y\} \mathbin{\&} \{x \mapsto 0\} \,]\!] = \mathit{fail}$. The number of variables appearing in a regular tree grammar defined below is finite. However, we would like to use regular tree grammars to define infinitely many substitutions such that the maximum number of variables we need cannot be fixed. To solve this problem, in the definition of $[\![\, \textsc{rec}(e, \delta) \,]\!]$, we introduced the operation $\mathit{fresh}_\delta(\cdot)$ that make all variables introduced by $[\![e]\!]$ *fresh w.r.t.* $\mathcal{D}om(\delta) \cup \mathcal{VR}an(\delta)$. In [29], this operation is implicitly considered, but in this paper, we explicitly introduced REC to the syntax in order to interpret terms in $\mathcal{T}(\Sigma)$ precisely. To assume $\mathcal{VR}an([\![e_1]\!]) \cap \mathcal{VR}an([\![e_2]\!]) = \emptyset$ for $[\![e_1 \ \& \ e_2]\!]$, we also introduced $\mathit{fresh}_{\theta_1}(\cdot)$ in the case of $[\![e_1 \ \& \ e_2]\!]$.

▶ **Example 16.** The expressions in Example 15 are interpreted as follows: $[\![\{y \mapsto 0\} \bullet \{x \mapsto \mathsf{s}(y)\}]\!] = \{x \mapsto \mathsf{s}(0),\, y \mapsto 0\}$, $[\![(\{x' \mapsto \mathsf{s}(y)\} \bullet \{x \mapsto x'\}) \ \& \ \{x \mapsto \mathsf{s}(\mathsf{s}(z))\}]\!] = \{x \mapsto \mathsf{s}(\mathsf{s}(z)),\, x' \mapsto \mathsf{s}(\mathsf{s}(z))\}$, $[\![(\varnothing \ \& \ \{y \mapsto z\}) \bullet \{x \mapsto \mathsf{s}(y)\}]\!] = \mathit{fail}$, and $[\![\mathrm{REC}(\{x \mapsto 0,\, y \mapsto \mathsf{s}(y')\}, \{x' \mapsto x,\, y' \mapsto y\}) \bullet \{y \mapsto \mathsf{s}(x')\}]\!] = \{x' \mapsto 0,\, y' \mapsto \mathsf{s}(y''),\, y \mapsto \mathsf{s}(0)\}$.

To define sets of idempotent substitutions, we adopt regular tree grammars. In the following, we drop the third component from grammars constructed below because the third one is fixed to Σ and a finite number of substitutions that are clear from production rules. A *substitution-set grammar* (SSG) for a term t_0 is a regular tree grammar $\mathcal{G} = (\Gamma_{t_0}, \mathcal{N}, \mathcal{P})$ such that $\mathcal{N}$ is a finite set of non-terminals Γ_t, $\Gamma_{t_0} \in \mathcal{N}$, and $\mathcal{P}$ is a finite set of production rules of the form $\Gamma_t \to \beta$ with $\beta \in \mathcal{T}(\Sigma \cup \mathcal{N})$. Note that $L(\mathcal{G}, \Gamma_t) = \{e \in \mathcal{T}(\Sigma) \mid \Gamma_t \to^*_{\mathcal{G}} e\}$ for each $\Gamma_t \in \mathcal{N}$. The set of substitutions defined by $\mathcal{G}$ from $\Gamma_t \in \mathcal{N}$ is defined as $[\![L(\mathcal{G}, \Gamma_t)]\!] = \{[\![e]\!] \mid e \in L(\mathcal{G}, \Gamma_t),\ [\![e]\!] \neq \mathit{fail}\}$.

▶ **Example 17.** The SSG $\mathcal{G}_3 = (\Gamma_x, \{\Gamma_x, \Gamma_y\}, \{\, \Gamma_x \to \{x \mapsto 0\} \mid \mathrm{REC}(\Gamma_y, \{x' \mapsto y\}) \bullet \{x \mapsto \mathsf{s}(x')\},\ \Gamma_y \to \mathrm{REC}(\Gamma_x, \{x' \mapsto x\}) \bullet \{y \mapsto \mathsf{s}(x')\} \,\})$ generates a set of expressions to define substitutions replacing x by even numbers over $0/0$ and $\mathsf{s}/1$. We have that $L(\mathcal{G}_3) = L(\mathcal{G}_3, \Gamma_x) = \{\{x \mapsto 0\},\ \mathrm{REC}((\mathrm{REC}(\{x \mapsto 0\}, \{x' \mapsto x\}) \bullet \{y \mapsto \mathsf{s}(x')\}), \{x' \mapsto y\}) \bullet \{x \mapsto \mathsf{s}(x')\},\ \ldots\}$, and $[\![L(\mathcal{G}_3, \Gamma_x)]\!] = \{\{x \mapsto \mathsf{s}^{2n}(0)\} \mid n \geq 0\}$.

6 Construction of Grammar Representations of Narrowing Trees

In this section, given a pc-CTRS and a goal clause, we show a construction of an SSG for the success set of the goal clause w.r.t. innermost narrowing of the CTRS. Since every constructor SDCTRS can be converted to an equivalent pc-CTRS w.r.t. $\xrightarrow{c}$ and $\overset{i}{\leadsto}$, we only consider pc-CTRSs. We employ the idea of narrowing trees, but we directly construct SSGs. In the following, we let $\mathcal{R}$ be a pc-CTRS over a signature $\mathcal{F}$ unless noted otherwise.

For a goal clause $T = (s_1 \twoheadrightarrow t_1) \ \& \cdots \& \ (s_n \twoheadrightarrow t_n)$, we denote the set of ground constructor terms appearing as right-hand sides of goals in T by $\mathit{Crhs}(T)$: $\mathit{Crhs}(T) = \{t_1, \ldots, t_n\} \cap \mathcal{T}(\mathcal{C}_\mathcal{R})$. We abuse Crhs for $\mathcal{R}$ and a goal clause T: $\mathit{Crhs}(\mathcal{R}, T) = \mathit{Crhs}(T) \cup \bigcup_{\ell \to r \Leftarrow C \in \mathcal{R}} \mathit{Crhs}(C)$. For example, $\mathit{Crhs}(\mathcal{R}_1, \mathsf{e}(x) \twoheadrightarrow \mathsf{true} \ \& \ \mathsf{o}(x) \twoheadrightarrow \mathsf{true}) = \{\mathsf{true}\}$. It is clear that $\mathit{Crhs}(\mathcal{R}, T)$ is finite.

Let T be a goal clause that does not contain $\top$. We prepare the set of constants $\mathcal{N}_{\mathcal{R},T} = \{\Gamma_T\} \cup \{\Gamma_{f(x_1,\ldots,x_n) \twoheadrightarrow u} \mid f/n \in \mathcal{D}_\mathcal{R},\ x_1, \ldots, x_n \in \mathcal{V},\ f(x_1, \ldots, x_n) \text{ is linear},\ u \in \mathit{Crhs}(\mathcal{R}, T) \cup (\mathcal{V} \setminus \{x_1, \ldots, x_n\})\}$. Note that $\mathcal{N}_{\mathcal{R},T}$ is finite up to variable renaming w.r.t. subscripts, and thus, we consider $\mathcal{N}_{\mathcal{R},T}$ a set of representatives: $\mathcal{V}ar(T') \cap \mathcal{V}ar(T'') = \emptyset$ and T' is not a variant of T'' for any different non-terminals $\Gamma_{T'}, \Gamma_{T''} \in \mathcal{N}_{\mathcal{R},T}$. We construct an SSG from $\mathcal{R}$ and T as follows: $\mathit{SSG}(\mathcal{R}, T) = (\Gamma_T, \mathcal{N}_{\mathcal{R},T}, \{\Gamma_{T'} \to \Phi_0(T') \mid \Gamma_{T'} \in \mathcal{N}_{\mathcal{R},T}\})$, where $\Phi_b(\cdot)$ with $b \in \{0, 1\}$ is inductively defined as follows:

Splitting $\Phi_b(T_1 \ \& \cdots \& \ T_n) = \Phi_1(T_1) \ \& \cdots \& \ \Phi_1(T_n)$,

Narrowing $\Phi_0(f(x_1, \ldots, x_n) \twoheadrightarrow u) = \Phi_1(T_1) \bullet \sigma_1 \mid \cdots \mid \Phi_1(T_m) \bullet \sigma_m$ if $f(x_1, \ldots, x_n)$ is basic and linear, and $x_1, \ldots, x_n \in \mathcal{V}$, where $\{(T', \sigma) \mid (f(x_1, \ldots, x_n) \twoheadrightarrow u) \overset{i}{\leadsto}_{\sigma,\mathcal{R}} T',\ \mathcal{VR}an(\sigma_i) \cap (\bigcup_{\Gamma_{T'} \in \mathcal{N}_{\mathcal{R},T}} \mathcal{V}ar(T')) = \emptyset\} = \{(T_1, \sigma_1), \ldots, (T_m, \sigma_m)\}$,

Narrowing $\Phi_b(t \twoheadrightarrow u) = \mathit{mgu}(\{t \approx u\})$ if $t, u \in \mathcal{T}(\mathcal{C}_\mathcal{R}, \mathcal{V})$ and t, u are unifiable,

Failure $\Phi_b(t \twoheadrightarrow u) = \varnothing$ if $t, u \in \mathcal{T}(\mathcal{C}_\mathcal{R}, \mathcal{V})$ and t, u are not unifiable,

Recursion $\Phi_1(f(x_1,\ldots,x_n) \twoheadrightarrow u) = \textsc{rec}(\Gamma_{f(x'_1,\ldots,x'_n)\twoheadrightarrow u'}, \{x_1 \mapsto x'_1,\ \ldots,\ x_n \mapsto x'_n\} \cup \delta)$ if $f(x_1,\ldots,x_n)$ is basic and linear, $\Gamma_{f(x'_1,\ldots,x'_n)\twoheadrightarrow u'} \in \mathcal{N}_{\mathcal{R},T}$, $x_1,\ldots,x_n \in \mathcal{V} \setminus \{x'_1,\ldots,x'_n\}$, $u' \in \mathcal{T}(\mathcal{C}_\mathcal{R}) \cup (\mathcal{V} \setminus (\{x_1,\ldots,x_n\} \cup \mathcal{V}ar(u)))$, and either $u = u' \in \mathcal{T}(\mathcal{C}_\mathcal{R})$ or $u, u' \in \mathcal{V}$, where if $u \in \mathcal{T}(\mathcal{C}_\mathcal{R})$, then $\delta = id$, and otherwise, $\delta = \{u \mapsto u'\}$), and

Flattening $\Phi_b(f(u_1,\ldots,u_n) \twoheadrightarrow u) = \Phi_1(f(y_1,\ldots,y_n) \twoheadrightarrow y)\ \&\ (u_1 \twoheadrightarrow y_1)\ \&\ \cdots\ \&\ (u_n \twoheadrightarrow y_n)\ \&\ (u \twoheadrightarrow y)$ if $f(u_1,\ldots,u_n) \twoheadrightarrow u$ is not a variant of $f(x'_1,\ldots,x'_n) \twoheadrightarrow u'$ with $\mathcal{V}ar(u_1,\ldots,u_n,u) \cap (\{x'_1,\ldots,x'_n\} \cup \mathcal{V}ar(u')) = \emptyset$ for any $\Gamma_{f(x'_1,\ldots,x'_n)\twoheadrightarrow u'} \in \mathcal{N}_{\mathcal{R},T}$, where $y_1,\ldots,y_n$, are fresh distinct variables w.r.t. $\mathcal{V}ar(u_1,\ldots,u_n,u) \cup \bigcup_{\Gamma_{T'} \in \mathcal{N}_{\mathcal{R},T}} \mathcal{V}ar(T')$. Note that we do not have to added the goal $u \twoheadrightarrow y$ to the result if $u \in \mathit{Crhs}(\mathcal{R},T)$.

Note that we may omit $\Gamma_{T'}$ and its production rules if $\Gamma_{T'}$ is not relevant to Γ_T. The subscript b of $\Phi_b(\cdot)$ is used to specify whether the call of $\Phi_b(\cdot)$ is initial or not. Without the subscript, for $\Gamma_{f(x_1,\ldots,x_n)\twoheadrightarrow u}$, we only construct $\Gamma_{f(x_1,\ldots,x_n)\twoheadrightarrow u} \to \textsc{rec}(id, \Gamma_{f(x_1,\ldots,x_n)\twoheadrightarrow u})$ which is meaningless. The definition of $\Phi_b(\cdot)$ follows the definition of a single step of narrowing, splitting under parallel composition, and flattening in the natural way. For example, the semantics of $\textsc{rec}$ takes renamings for $\mathcal{V}ar(\ell, r, C) \cap \mathcal{V}ar(S) = \emptyset$ in the definition of innermost-narrowing into account and enables us to reuse substitutions generated by the first argument of $\textsc{rec}$.

▶ **Example 18.** For $\mathcal{R}_1$ and the goal clause $\mathsf{e}(x) \twoheadrightarrow \mathsf{true}\ \&\ \mathsf{o}(x) \twoheadrightarrow \mathsf{true}$, we prepare constants $\Gamma_{\mathsf{e}(x)\twoheadrightarrow\mathsf{true}\&\mathsf{o}(x)\twoheadrightarrow\mathsf{true}}$, $\Gamma_{\mathsf{e}(x')\twoheadrightarrow\mathsf{true}}$, and $\Gamma_{\mathsf{o}(x'')\twoheadrightarrow\mathsf{true}}$ because we have that $\mathit{Crhs}(\mathcal{R}_1, \mathsf{e}(x) \twoheadrightarrow \mathsf{true}\ \&\ \mathsf{o}(x) \twoheadrightarrow \mathsf{true}) = \{\mathsf{true}\}$. For the goal $\mathsf{e}(x') \twoheadrightarrow \mathsf{true}$, we have the following conversion:

$$\begin{aligned}\Phi_0(\mathsf{e}(x') \twoheadrightarrow \mathsf{true}) = id \bullet \{x' \mapsto \mathsf{0}\} \mid \big(\textsc{rec}(\Gamma_{\mathsf{o}(x'')\twoheadrightarrow\mathsf{true}}, \{x_1 \mapsto x''\})\ \&\ id\big) \bullet \{x' \mapsto \mathsf{s}(x_1)\}\\ \mid \big(\textsc{rec}(\Gamma_{\mathsf{e}(x')\twoheadrightarrow\mathsf{true}}, \{x_2 \mapsto x'\})\ \&\ \varnothing\big) \bullet \{x' \mapsto \mathsf{s}(x_2)\}\end{aligned}$$

From the conversion above, the SSG $\mathcal{G}_2$ with the following production rules is constructed:

$$\begin{aligned}\Gamma_{\mathsf{e}(x)\twoheadrightarrow\mathsf{true}\&\mathsf{o}(x)\twoheadrightarrow\mathsf{true}} &\to \textsc{rec}(\Gamma_{\mathsf{e}(x')\twoheadrightarrow\mathsf{true}}, \{x \mapsto x'\})\ \&\ \textsc{rec}(\Gamma_{\mathsf{o}(x'')\twoheadrightarrow\mathsf{true}}, \{x \mapsto x''\})\\ \Gamma_{\mathsf{e}(x')\twoheadrightarrow\mathsf{true}} &\to id \bullet \{x' \mapsto \mathsf{0}\} \mid \big(\textsc{rec}(\Gamma_{\mathsf{o}(x'')\twoheadrightarrow\mathsf{true}}, \{x_1 \mapsto x''\})\ \&\ id\big) \bullet \{x' \mapsto \mathsf{s}(x_1)\}\\ &\quad \mid \big(\textsc{rec}(\Gamma_{\mathsf{e}(x')\twoheadrightarrow\mathsf{true}}, \{x_2 \mapsto x'\})\ \&\ \varnothing\big) \bullet \{x' \mapsto \mathsf{s}(x_2)\}\\ \Gamma_{\mathsf{o}(x'')\twoheadrightarrow\mathsf{true}} &\to \varnothing \bullet \{x'' \mapsto \mathsf{0}\} \mid \big(\textsc{rec}(\Gamma_{\mathsf{e}(x')\twoheadrightarrow\mathsf{true}}, \{x_3 \mapsto x'\})\ \&\ id\big) \bullet \{x'' \mapsto \mathsf{s}(x_3)\}\\ &\quad \mid \big(\textsc{rec}(\Gamma_{\mathsf{o}(x'')\twoheadrightarrow\mathsf{true}}, \{x_4 \mapsto x''\})\ \&\ \varnothing\big) \bullet \{x'' \mapsto \mathsf{s}(x_4)\}\end{aligned}$$

▶ **Theorem 19.** *Let T be a goal clause without $\top$. Then, $[\![L(\mathit{SSG}(\mathcal{R},T), \Gamma_T)]\!] = \mathit{Suc}(\overset{i}{\rightsquigarrow}_\mathcal{R}, T)$ up to variable renaming.*

Note that Theorem 19 corresponds to [29, Theorem 20]. For a constructor SDCTRS $\mathcal{R}$ and a goal clause T, Theorem 6 enables us to use $\mathit{SSG}(\mathit{Pc}(\mathcal{R}), T)$ for $\mathcal{R}$.

7 Simplification of Grammar Representations

In this section, we show some methods to simplify production rules of SSGs. Given an SSG $\mathcal{G} = (\Gamma_T, \mathcal{N}, \mathcal{P})$, we extend the semantics of terms in $\mathcal{T}(\Sigma)$ to sets of terms in $\mathcal{T}(\Sigma \cup \mathcal{N})$ as follows: $\{\![e]\!\}_\mathcal{G} = [\![L((\Gamma_T, \mathcal{N} \cup \{\Gamma_e\}, \mathcal{P} \cup \{\Gamma_e \to e\}), \Gamma_e)]\!]$ for $e \in \mathcal{T}(\Sigma \cup \mathcal{N})$, where $\Gamma_e \notin \mathcal{N}$. We say that terms $e_1, e_2 \in \mathcal{T}(\Sigma \cup \mathcal{N})$ are *semantically equivalent w.r.t. $\mathcal{G}$* if $\{\![e_1]\!\}_\mathcal{G} = \{\![e_2]\!\}_\mathcal{G}$ up to variable renaming.

We first compute subexpressions consisting of substitutions, $\bullet$, $\&$, and $\varnothing$. The following equivalences trivially hold:

▶ **Theorem 20.** *Let $\mathcal{G} = (\Gamma_T, \mathcal{N}, \mathcal{P})$, θ_1, θ_2 idempotent substitutions, and $e \in \mathcal{T}(\Sigma \cup \mathcal{N})$. Then, all of the following hold: $\{\![\theta_1 \bullet \theta_2]\!\}_\mathcal{G} = \{\![\theta_1 \cdot \theta_2]\!\}_\mathcal{G}$, $\{\![e \bullet \varnothing]\!\}_\mathcal{G} = \{\![\varnothing \bullet e]\!\}_\mathcal{G} = \{\![\varnothing\ \&\ e]\!\}_\mathcal{G} = \{\![e\ \&\ \varnothing]\!\}_\mathcal{G} = \{\![\varnothing]\!\}_\mathcal{G}$ ($= \emptyset$), and $\{\![id\ \&\ e]\!\}_\mathcal{G} = \{\![e\ \&\ id]\!\}_\mathcal{G} = \{\![e]\!\}_\mathcal{G}$.*

Following Theorem 20, we simplify subexpressions to the smallest one among semantically equivalent terms w.r.t. $\mathcal{G}$ (e.g., replace $e \bullet \varnothing$ by $\varnothing$) as much as possible.

▶ **Example 21.** The production rules of $\mathcal{G}_2$ in Example 18 are simplified as follows:

$$\begin{aligned}\Gamma_{\mathsf{e}(x)\twoheadrightarrow\mathsf{true}\&\mathsf{o}(x)\twoheadrightarrow\mathsf{true}} &\to \textsc{rec}(\Gamma_{\mathsf{e}(x')\twoheadrightarrow\mathsf{true}}, \{x \mapsto x'\}) \;\&\; \textsc{rec}(\Gamma_{\mathsf{o}(x'')\twoheadrightarrow\mathsf{true}}, \{x \mapsto x''\})\\ \Gamma_{\mathsf{e}(x')\twoheadrightarrow\mathsf{true}} &\to \{x' \mapsto \mathsf{0}\} \mid \textsc{rec}(\Gamma_{\mathsf{o}(x'')\twoheadrightarrow\mathsf{true}}, \{x_1 \mapsto x''\}) \bullet \{x' \mapsto \mathsf{s}(x_1)\}\\ \Gamma_{\mathsf{o}(x'')\twoheadrightarrow\mathsf{true}} &\to \textsc{rec}(\Gamma_{\mathsf{e}(x')\twoheadrightarrow\mathsf{true}}, \{x_3 \mapsto x'\}) \bullet \{x'' \mapsto \mathsf{s}(x_3)\}\end{aligned}$$

The occurrence of & in SSGs makes it difficult to simplify and analyze grammar representations of narrowing trees. Since the second and third production rules in Example 21 no longer contain &, we focus on $\textsc{rec}(\Gamma_{\mathsf{e}(x')\twoheadrightarrow\mathsf{true}}, \{x \mapsto x'\}) \;\&\; \textsc{rec}(\Gamma_{\mathsf{o}(x'')\twoheadrightarrow\mathsf{true}}, \{x \mapsto x''\})$ which is the right-hand side of the first rule. Let us consider the sets L_1, L_2 of terms substituted for x by means of substitutions in $\{\![\, \textsc{rec}(\Gamma_{\mathsf{e}(x')\twoheadrightarrow\mathsf{true}}, \{x \mapsto x'\}) \,]\!\}_{\mathcal{G}_2}$ and $\{\![\, \textsc{rec}(\Gamma_{\mathsf{o}(x'')\twoheadrightarrow\mathsf{true}}, \{x \mapsto x''\}) \,]\!\}_{\mathcal{G}_2}$, respectively: $L_1 = \{\sigma x \mid \sigma \in \{\![\, \textsc{rec}(\Gamma_{\mathsf{e}(x')\twoheadrightarrow\mathsf{true}}, \{x \mapsto x'\}) \,]\!\}_{\mathcal{G}_2}\}$, and $L_2 = \{\sigma x \mid \sigma \in \{\![\, \textsc{rec}(\Gamma_{\mathsf{o}(x'')\twoheadrightarrow\mathsf{true}}, \{x \mapsto x''\}) \,]\!\}_{\mathcal{G}_2}\}$. If $L_1 \cap L_2 = \emptyset$, then $\{\![\, \textsc{rec}(\Gamma_{\mathsf{e}(x')\twoheadrightarrow\mathsf{true}}, \{x \mapsto x'\}) \;\&\; \textsc{rec}(\Gamma_{\mathsf{o}(x'')\twoheadrightarrow\mathsf{true}}, \{x \mapsto x''\}) \,]\!\}_{\mathcal{G}_2} = \emptyset$ and we obtain $\Gamma_{\mathsf{e}(x)\twoheadrightarrow\mathsf{true}\&\mathsf{o}(x)\twoheadrightarrow\mathsf{true}} \to \varnothing$ which is our goal of simplification in this section. To generate L_1 and L_2, we transform the second and third production rules in Example 21 into a regular tree grammar generating the sets L_1 and L_2. In the rest of this section, we assume that the signature $\mathcal{F}$ contains a constant.

Let $\mathcal{G}$ be an SSG $(\Gamma_{T_0}, \mathcal{N}, \mathcal{P})$ and T a goal clause such that $\Gamma_T \in \mathcal{N}$. We denote by $\mathcal{P}|_{\Gamma_T}$ the set of production rules that are reachable from Γ_T. We assume that any rule in $\mathcal{P}|_{\Gamma_T}$ is of the form $\Gamma_{T'} \to \theta_1 \mid \cdots \mid \theta_m \mid \textsc{rec}(\Gamma_{T_1}, \delta_1) \bullet \sigma_1 \mid \cdots \mid \textsc{rec}(\Gamma_{T_n}, \delta_n) \bullet \sigma_n$, where $\theta_1, \ldots, \theta_m, \sigma_1, \ldots, \sigma_n$ are idempotent substitutions. Note that $\Gamma_{T'} \to \textsc{rec}(\Gamma_{T''}, \delta)$ is considered $\Gamma_{T'} \to \textsc{rec}(\Gamma_{T''}, \delta) \bullet id$. Note also that the following construction is applicable under this assumption. The regular tree grammar obtained from $\mathcal{G}$ and a variable x in T, written as $RTG(\mathcal{G}, T, x)$, is $(\Gamma^x_T, \mathcal{N}', \mathcal{P}' \cup \mathcal{P}'' \cup \{A \to g(\overbrace{A, \ldots, A}^{n}) \mid g/n \in \mathcal{C}_\mathcal{R}\})$ such that

- $\mathcal{N}' = \{\, \Gamma^{x'}_{T'} \mid \Gamma_{T'} \in \mathcal{N},\ x' \in \mathcal{V}ar(T') \,\} \cup \{A\}$, and
- $\mathcal{P}' = \{\, \Gamma^{x'}_{T'} \to \xi_{\mathcal{V}ar(\theta_i x')}(\theta_i x') \mid x' \in \mathcal{V}ar(T'),\ \Gamma_{T'} \to \theta_i \in \mathcal{P} \,\}$, and
- $\mathcal{P}'' = \{\, \Gamma^{x'}_{T'} \to \Big(\{y \mapsto \Gamma^{\delta_j y}_{T_j} \mid y \in \mathcal{D}om(\delta_j)\} \cup \xi_{\mathcal{V}ar(\sigma_j x') \setminus \mathcal{D}om(\delta_j)}\Big)(\sigma_j x') \mid x' \in \mathcal{V}ar(T'),$ $\Gamma_{T'} \to \textsc{rec}(\Gamma_{T_j}, \delta_j) \bullet \sigma_j \in \mathcal{P} \,\}$,

where $\xi_X = \{y \mapsto A \mid y \in X\}$, which corresponds to ξ in the definition of $fresh_\delta(\cdot)$. Note that the non-terminal A generates $\mathcal{T}(\mathcal{C}_\mathcal{R})$ and corresponds to a fresh variable.

▶ **Theorem 22.** *Let $\mathcal{G}$ be an SSG $(\Gamma_{T_0}, \mathcal{N}, \mathcal{P})$, $\Gamma_{T_1}, \Gamma_{T_2} \in \mathcal{N}$, $x \in \mathcal{V}$, $x_1 \in \mathcal{V}ar(T_1)$, $x_2 \in \mathcal{V}ar(T_2)$, $RTG(\mathcal{G}, T_1, x_1)$, $RTG(\mathcal{G}, T_2, x_2)$ be constructed, and δ_1, δ_2 be renamings such that $\mathcal{VR}an(\delta_i) = \mathcal{V}ar(T_i)$ and $\delta_i x = x_i$ for $i = 1, 2$. If $L(RTG(\mathcal{G}, T_1, x_1)) \cap L(RTG(\mathcal{G}, T_2, x_2)) = \emptyset$, then $\{\![\, \textsc{rec}(\Gamma_{T_1}, \delta_1) \;\&\; \textsc{rec}(\Gamma_{T_2}, \delta_2) \,]\!\}_\mathcal{G} = \{\![\, \varnothing \,]\!\}_\mathcal{G}$.*

▶ **Example 23.** From the production rules in Example 21, we obtain the following regular tree grammars:

- $\mathcal{G}_2' = RTG(\mathcal{G}_2, \mathsf{e}(x') \twoheadrightarrow \mathsf{true}, x') = (\Gamma^{x'}_{\mathsf{e}(x')\twoheadrightarrow\mathsf{true}}, \{\Gamma^{x'}_{\mathsf{e}(x')\twoheadrightarrow\mathsf{true}}, \Gamma^{x''}_{\mathsf{o}(x'')\twoheadrightarrow\mathsf{true}}\}, \mathcal{P}')$, and
- $\mathcal{G}_2'' = RTG(\mathcal{G}_2, \mathsf{o}(x'') \twoheadrightarrow \mathsf{true}, x'') = (\Gamma^{x''}_{\mathsf{o}(x'')\twoheadrightarrow\mathsf{true}}, \{\Gamma^{x'}_{\mathsf{e}(x')\twoheadrightarrow\mathsf{true}}, \Gamma^{x''}_{\mathsf{o}(x'')\twoheadrightarrow\mathsf{true}}\}, \mathcal{P}')$,

where

- $\mathcal{P}' = \{\ \Gamma^{x'}_{\mathsf{e}(x')\twoheadrightarrow\mathsf{true}} \to \mathsf{0} \mid \mathsf{s}(\Gamma^{x''}_{\mathsf{o}(x'')\twoheadrightarrow\mathsf{true}}),\ \ \Gamma^{x''}_{\mathsf{o}(x'')\twoheadrightarrow\mathsf{true}} \to \mathsf{s}(\Gamma^{x'}_{\mathsf{e}(x')\twoheadrightarrow\mathsf{true}})\ \}$.

We can decide the intersection emptiness problem of $L(\mathcal{G}_2')$ and $L(\mathcal{G}_2'')$, and the answer is true: $L(\mathcal{G}_2') \cap L(\mathcal{G}_2'') = \emptyset$. Thanks to Theorem 22, we can replace the expression $\textsc{rec}(\Gamma_{\mathsf{e}(x')\twoheadrightarrow\mathsf{true}}, \{x \mapsto x'\}) \;\&\; \textsc{rec}(\Gamma_{\mathsf{o}(x'')\twoheadrightarrow\mathsf{true}}, \{x \mapsto x''\})$ by $\varnothing$, and thus we can transform the first production rule in Example 21 into $\Gamma_{\mathsf{e}(x)\twoheadrightarrow\mathsf{true}\&\mathsf{o}(x)\twoheadrightarrow\mathsf{true}} \to \varnothing$.

In summary, the simplification proposed in this section is to replace subexpressions by semantically equivalent smaller ones as much as possible by following Theorems 20 and 22. This simplification always halts because the number of Σ-symbols in equations is strictly decreasing at every simplification step. In addition, it is clear that results of the simplification are unique.

If a constructor SDCTRS $\mathcal{R}$ has a nest of defined symbols or a goal clause T contains, e.g., $(f(\vec{x}) \twoheadrightarrow x') \mathrel{\&} (g(\vec{y}) \twoheadrightarrow y')$, to simplify $\mathit{SSG}(\mathit{Trs}(\mathit{Pc}(\mathcal{R})), T)$ as much as possible, we apply the simplification based on Theorem 22 at least once, i.e., we try to solve the intersection emptiness problem of regular tree grammars at least once, which is EXPTIME-complete. The number of occurrence of $\bullet$ and & is at most $O(n)$, where n is the size of $\mathcal{R}$ and T. Therefore, the cost of the overall simplification is EXPTIME-complete.

8 Applications

In this section, we show that grammar representations of narrowing trees are useful to prove (1) infeasibility of conditional critical pairs of $\mathcal{R}_1$ and (2) quasi-reducibility of $\mathcal{R}_1$ with usual sorts for the non-negative integers and the boolean values.

Two conditional rewrite rules $\ell_1 \to r_1 \Leftarrow C_1$ and $\ell_2 \to r_2 \Leftarrow C_2$ that are renamed to have no shared variable are said to be *overlapping* if there exists a non-variable position p of ℓ_1 such that $\ell_1|_p$ and ℓ_2 are unifiable, and $p \neq \varepsilon$ if one of the rules is a renamed variant of the other. In this case, given $\sigma = \mathit{mgu}(\{\ell_1|_p \approx \ell_2\})$, the triple $(\sigma(\ell_1[r_2]_p), \sigma r_1, \sigma C_1 \mathrel{\&} \sigma C_2)$, denoted by $\langle \sigma(\ell_1[r_2]_p), \sigma r_1 \rangle \Leftarrow \sigma C_1 \mathrel{\&} \sigma C_2$, is called a *conditional critical pair* of $\mathcal{R}$. A conditional critical pair $\langle s, t \rangle \Leftarrow s_1 \twoheadrightarrow t_1 \mathrel{\&} \cdots \mathrel{\&} s_k \twoheadrightarrow t_k$ is called *infeasible* if there exists no substitution θ such that $\theta s_i \to^*_{\mathcal{R}} \theta t_i$ for all $1 \le i \le k$, and called *joinable* if $\theta s \downarrow_{\mathcal{R}} \theta t$ for any substitution θ such that $\theta s_i \to^*_{\mathcal{R}} \theta t_i$ for all $1 \le i \le k$. Note that infeasible conditional critical pairs are joinable and unconditional critical pairs are feasible. Therefore, from [4, Theorem 3.8] and [21, Theorem 3], an operationally terminating CTRS $\mathcal{R}$ is confluent if all critical pairs of $\mathcal{R}$ are infeasible.

▶ **Example 24.** Consider $\mathcal{R}_1$ in Section 1. It follows from Example 23 that $\mathit{Suc}(\overset{i}{\leadsto}_{\mathcal{R}_1}, \mathsf{o}(x) \twoheadrightarrow \mathsf{true} \mathrel{\&} \mathsf{e}(x) \twoheadrightarrow \mathsf{true}) = \emptyset$. This means that there exists no constructor term t such that $\mathsf{o}(t) \overset{c}{\to}^*_{\mathcal{R}_1} \mathsf{true}$ and $\mathsf{e}(t) \overset{c}{\to}^*_{\mathcal{R}_1} \mathsf{true}$. Assume that there exists a term t such that $\mathsf{o}(t) \to^*_{\mathcal{R}_1} \mathsf{true}$ and $\mathsf{e}(t) \to^*_{\mathcal{R}_1} \mathsf{true}$. Since $\overset{c}{\to}_{\mathcal{R}_1} = \overset{c}{\to}_{\mathit{Trs}(\mathcal{R}_1)}$ and $\mathit{Trs}(\mathcal{R}_1)$ is non-erasing, t should be a ground constructor term. Since $\mathit{Trs}(\mathcal{R}_1)$ is a constructor system, we have that $\mathsf{o}(t) \overset{c}{\to}^*_{\mathcal{R}_1} \mathsf{true}$ and $\mathsf{e}(t) \overset{c}{\to}^*_{\mathcal{R}_1} \mathsf{true}$, and hence $\mathsf{o}(x) \overset{i}{\leadsto}^*_{\theta_1, \mathcal{R}_1} \mathsf{true}$ and $\mathsf{e}(x) \overset{i}{\leadsto}^*_{\theta_2, \mathcal{R}_1} \mathsf{true}$ for some constructor substitutions θ_1, θ_2. It follows from Theorem 12 that $\theta_1 \Uparrow \theta_2 \in \mathit{Suc}(\overset{i}{\leadsto}_{\mathcal{R}_1}, \mathsf{o}(x) \twoheadrightarrow \mathsf{true} \mathrel{\&} \mathsf{e}(x) \twoheadrightarrow \mathsf{true})$. This contradicts the fact that $\mathit{Suc}(\overset{i}{\leadsto}_{\mathcal{R}_1}, \mathsf{o}(x) \twoheadrightarrow \mathsf{true} \mathrel{\&} \mathsf{e}(x) \twoheadrightarrow \mathsf{true}) = \emptyset$. Therefore, all critical pairs of $\mathcal{R}_1$ are infeasible, and hence $\mathcal{R}_1$ is confluent.

A CTRS $\mathcal{R}$ is called *quasi-reducible* [16] if any ground basic term is not a normal form. $\mathcal{R}$ is called *sufficiently complete* if for every ground term t, there exists a ground constructor term u such that $t \to^*_{\mathcal{R}} u$ [12]. Note that if an operationally terminating CTRS is quasi-reducible, then the CTRS is sufficiently complete.

▶ **Example 25.** CTRS $\mathcal{R}_1$ in Section 1 is not quasi-reducible since $\mathsf{e}(\mathsf{true})$ is not defined. Thus, let us consider the sorts with $\mathsf{0} : \mathit{nat}$, $\mathsf{s} : \mathit{nat} \to \mathit{nat}$, $\mathsf{true}, \mathsf{false} : \mathit{bool}$, and $\mathsf{e}, \mathsf{o} : \mathit{nat} \to \mathit{bool}$. For quasi-reducibility of $\mathcal{R}_1$ with the sorts, it suffices to show that $\mathsf{e}(\mathsf{s}^n(\mathsf{0}))$ and $\mathsf{o}(\mathsf{s}^n(\mathsf{0}))$ with $n \ge 0$ are reducible. It follows from the unconditional rules $\mathsf{e}(\mathsf{0}) \to \mathsf{true}$ and $\mathsf{o}(\mathsf{0}) \to \mathsf{false}$ that $\mathsf{e}(\mathsf{0})$ and $\mathsf{o}(\mathsf{0})$ are reducible. From the production rules in Example 23, we can show that $L(\mathcal{G}'_2) \cup L(\mathcal{G}''_2) = \mathcal{T}(\{\mathsf{0}, \mathsf{s}\})$, and hence $\mathsf{e}(\mathsf{s}^n(\mathsf{0}))$ and $\mathsf{o}(\mathsf{s}^n(\mathsf{0}))$ with $n > 0$ are reducible. Therefore, $\mathcal{R}_1$ with the sorts is quasi-reducible, and hence sufficiently complete.

9 Related Work

One of the closest related work to be compared with our results must be reachability analysis for CTRSs. A well-investigated approach is *tree automata techniques* (cf. [7, 6]): given a (C)TRS and two terms s, t, we construct a tree automaton that over-approximately recognizes all descendants of any ground instance of s, and solves the intersection emptiness problem between the automaton and another one for ground instances of t. To prove infeasibility of $\mathsf{o}(x) \twoheadrightarrow \mathsf{true}$ & $\mathsf{e}(x) \twoheadrightarrow \mathsf{true}$ w.r.t. $\xrightarrow{c}_{\mathcal{R}_1}$ via reachability, we convert it to the reachability problem from ground instances of $\mathsf{c}(\mathsf{o}(x), \mathsf{e}(x))$ to $\mathsf{c}(\mathsf{true}, \mathsf{true})$. Tree automata techniques need overapproximation for non-linear terms, and thus, the reachability problem is solved as the reachability from $\mathsf{c}(\mathsf{o}(x'), \mathsf{e}(x''))$ to $\mathsf{c}(\mathsf{true}, \mathsf{true})$. Due to this linearization, the non-existence of a ground instance of x cannot be proved. The method in [36] for proving infeasibility of conditional critical pairs analyzes reachability using the underlying TRSs – $\{\ \mathsf{e}(0) \to \mathsf{true},\ \mathsf{e}(\mathsf{s}(x)) \to \mathsf{true},\ \mathsf{e}(\mathsf{s}(x)) \to \mathsf{false},\ \mathsf{o}(0) \to \mathsf{false},\ \mathsf{o}(\mathsf{s}(x)) \to \mathsf{true},\ \mathsf{o}(\mathsf{s}(x)) \to \mathsf{false}\ \}$ for $\mathcal{R}_1$ – , and thus, the non-existence of a ground t with $\mathsf{o}(t) \to^*_{\mathcal{R}_1} \mathsf{true}$ and $\mathsf{e}(t) \to^*_{\mathcal{R}_1} \mathsf{true}$ cannot be proved. On the other hand, the approach in this paper is to construct a regular tree grammar that can be seen as a tree automaton, and that recognizes ground terms given at an argument of a defined symbol we are interested in.

Another important related work is a *semantic approach* to infeasibility analysis for conditional rewrite rules and conditional critical pairs of CTRSs [19, 18], which uses AGES [11] based on the methods in [20]. The semantic approach reduces infeasibility of conditions $s_1 \twoheadrightarrow t_1, \ldots, s_k \twoheadrightarrow t_k$ to the existence of a logical model for the theory $\overline{\mathcal{R}} \cup \{\neg(\exists \overrightarrow{X}.\ s_1 \to^* t_1 \wedge \cdots \wedge s_k \to^* t_k)\}$, where $X = \mathcal{V}ar(s_1, t_1, \ldots, s_k, t_k)$ and $\overline{\mathcal{R}}$ is a first-order theory obtained by $\mathcal{R}$. The power of proving infeasibility relies on that of generating a model for the theory. For example, infeasibility of $x < y \twoheadrightarrow \mathsf{true}, y < x \twoheadrightarrow \mathsf{true}$ w.r.t. $\mathcal{R}_4 = \{\ 0 < \mathsf{s}(y) \to \mathsf{true},\ x < 0 \to \mathsf{false},\ \mathsf{s}(x) < \mathsf{s}(y) \to x < y\ \}$ can be reduced to the existence of a model for $\overline{\mathcal{R}_4} \cup \{\neg(\forall x, y.\ x < y \to^* \mathsf{true} \wedge y < x \to^* \mathsf{true})\}$, but AGES did not find any model for the theory via its web interface with default parameters. The power of our method for proving infeasibility relies on the success of simplifying SSGs to $\Gamma_T \to \varnothing$. For this reason, it is not so easy to compare these two approaches from theoretical point of view to prove infeasibility of conditions. On the other hand, our result can be used to prove quasi-reducibility of $\mathcal{R}_1$ with usual sorts for the non-negative integers and the boolean values.

10 Conclusion

In this paper, we extended grammar representations of narrowing trees to constructor SDCTRSs, and showed that grammar representations are useful to prove confluence and quasi-reducibility of a normal CTRS. We will implement the construction and simplification of grammar representations for narrowing trees, and will introduce them into CO3 [25] to use them to prove confluence of constructor SDCTRSs. In addition, we will make an empirical comparison of the tree automata approach, the semantic approach, and ours to infeasibility analysis of constructor SDCTRSs after implementing our method. Narrowing trees define constructor substitutions obtained by innermost narrowing. For this reason, the usefulness is limited to constructor-based rewriting only. A further direction of this research will be to extend narrowing trees to other kinds of narrowing, e.g., *basic* narrowing [14].

References

1 Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

2 Adel Bouhoula and Florent Jacquemard. Sufficient completeness verification for conditional and constrained TRS. *Journal of Applied Logic*, 10(1):127–143, 2012.

3 Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Löding, Sophie Tison, and Marc Tommasi. Tree Automata Techniques and Applications, 2007. Release October, 12th 2007.

4 Nachum Dershowitz, Mitsuhiro Okada, and G. Sivakumar. Canonical conditional rewrite systems. In *Proceedings of the 9th International Conference on Automated Deduction*, volume 310 of *Lecture Notes in Computer Science*, pages 538–549. Springer, 1988.

5 Francisco Durán, Salvador Lucas, José Meseguer, Claude Marché, and Xavier Urbain. Proving termination of membership equational programs. In Nevin Heintze and Peter Sestoft, editors, *Proceedings of the 2004 ACM SIGPLAN Workshop on Partial Evaluation and Semantics-based Program Manipulation*, pages 147–158. ACM, 2004.

6 Guillaume Feuillade and Thomas Genet. Reachability in conditional term rewriting systems. *Electronic Notes in Theoretical Computer Science*, 86(1):133–146, 2003.

7 Thomas Genet. Decidable approximations of sets of descendants and sets of normal forms. In Tobias Nipkow, editor, *Proceedings of the 9th International Conference on Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, pages 151–165. Springer, 1998.

8 Karl Gmeiner. CoScart: Confluence prover in Scala. In Ashish Tiwari and Takahito Aoto, editors, *Proceedings of the 4th International Workshop on Confluence*, page 45, 2015.

9 Karl Gmeiner and Naoki Nishida. Notes on structure-preserving transformations of conditional term rewrite systems. In Manfred Schmidt-Schauß, Masahiko Sakai, David Sabel, and Yuki Chiba, editors, *Proceedings of the first International Workshop on Rewriting Techniques for Program Transformations and Evaluation*, volume 40 of *OpenAccess Series in Informatics*, pages 3–14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014.

10 Karl Gmeiner, Naoki Nishida, and Bernhard Gramlich. Proving confluence of conditional term rewriting systems via unravelings. In Nao Hirokawa and Vincent van Oostrom, editors, *Proceedings of the 2nd International Workshop on Confluence*, pages 35–39, 2013.

11 Raúl Gutiérrez, Salvador Lucas, and Patricio Reinoso. A tool for the automatic generation of logical models of order-sorted first-order theories. In Alicia Villanueva, editor, *Proceedings of the XVI Jornadas sobre Programación y Lenguages*, pages 215–230, 2016. Tool available at `http://zenon.dsic.upv.es/ages/`.

12 John V. Guttag. *The Specification and Application to Programming of Abstract Data Types*. PhD thesis, University of Tronto, Toronto, Canada, 1975.

13 Manuel V. Hermenegildo and Francesca Rossi. On the correctness and efficiency of independent and-parallelism in logic programs. In Ewing L. Lusk and Ross A. Overbeek, editors, *Proceedings of the North American Conference on Logic Programming*, pages 369–389. MIT Press, 1989.

14 Jean-Marie Hullot. Canonical forms and unification. In Wolfgang Bibel and Robert A. Kowalski, editors, *Proceedings of the 5th Conference on Automated Deduction*, volume 87 of *Lecture Notes in Computer Science*, pages 318–334. Springer, 1980.

15 Deepak Kapur, Paliath Narendran, and Hantao Zhang. On sufficient-completeness and related properties of term rewriting systems. *Acta Informatica*, 24(4):395–415, 1987.

16 Emmanuel Kounalis. Completeness in data type specifications. In Bob F. Caviness, editor, *Proceedings of the European Conference on Computer Algebra*, volume 204 of *Lecture Notes in Computer Science*, pages 348–362. Springer, 1985.

17 Marija Kulaš. A practical view on renaming. In Sibylle Schwarz and Janis Voigtländer, editors, *Proceedings of the 29th and 30th Workshops on (Constraint) Logic Programming and 24th International Workshop on Functional and (Constraint) Logic Programming, and 24th International Workshop on Functional and (Constraint) Logic Programming*, volume 234 of *Electronic Proceedings in Theoretical Computer Science*, pages 27–41, 2017.

18 Salvador Lucas. A semantic approach to the analysis of rewriting-based systems. In Fabio Fioravanti and John P. Gallagher, editors, *Pre-Proceedings of the 27th International Symposium on Logic-Based Program Synthesis and Transformation*, 2017. *CoRR*, abs/1709.05095.

19 Salvador Lucas and Raúl Gutiérrez. A semantic criterion for proving infeasibility in conditional rewriting. In Beniamino Accattoli and Bertram Felgenhauer, editors, *Proceedings of the 6th International Workshop on Confluence*, pages 15–20, 2017.

20 Salvador Lucas and Raúl Gutiérrez. Automatic synthesis of logical models for order-sorted first-order theories. *Journal of Automated Reasoning*, 60(4):465–01, 2018.

21 Salvador Lucas, Claude Marché, and José Meseguer. Operational termination of conditional term rewriting systems. *Information Processing Letters*, 95(4):446–453, 2005.

22 Massimo Marchiori. Unravelings and ultra-properties. In Michael Hanus and Mario Rodríguez-Artalejo, editors, *Proceedings of the 5th International Conference on Algebraic and Logic Programming*, volume 1139 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 1996.

23 Aart Middeldorp and Erik Hamoen. Completeness results for basic narrowing. *Applicable Algebra in Engineering, Communication and Computing*, 5:213–253, 1994.

24 Masanori Nagashima, Masahiko Sakai, and Toshiki Sakabe. Determinization of conditional term rewriting systems. *Theoretical Computer Science*, 464:72–89, 2012.

25 Naoki Nishida, Takayuki Kuroda, Makishi Yanagisawa, and Karl Gmeiner. CO3: a COnverter for proving COnfluence of COnditional TRSs. In Ashish Tiwari and Takahito Aoto, editors, *Proceedings of the 4th International Workshop on Confluence*, page 42, 2015.

26 Naoki Nishida, Adrián Palacios, and Germán Vidal. Reversible computation in term rewriting. *Journal of Logical and Algebraic Methods in Programming*, 94:128–149, 2018.

27 Naoki Nishida and Germán Vidal. Program inversion for tail recursive functions. In Manfred Schmidt-Schauß, editor, *Proceedings of the 22nd International Conference on Rewriting Techniques and Applications*, volume 10 of *Leibniz International Proceedings in Informatics*, pages 283–298. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2011.

28 Naoki Nishida and Germán Vidal. Computing more specific versions of conditional rewriting systems. In Elvira Albert, editor, *Revised Selected Papers of the 22nd International Symposium on Logic-Based Program Synthesis and Transformation*, volume 7844 of *Lecture Notes in Computer Science*, pages 137–154. Springer, 2013.

29 Naoki Nishida and Germán Vidal. A finite representation of the narrowing space. In Gopal Gupta and Ricardo Peña, editors, *Revised Selected Papers of the 23rd International Symposium on Logic-Based Program Synthesis and Transformation*, volume 8901 of *Lecture Notes in Computer Science*, pages 54–71. Springer, 2014.

30 Naoki Nishida and Germán Vidal. A framework for computing finite SLD trees. *Journal of Logic and Algebraic Methods in Programming*, 84(2):197–217, 2015.

31 Naoki Nishida, Makishi Yanagisawa, and Karl Gmeiner. On proving confluence of conditional term rewriting systems via the computationally equivalent transformation. In Takahito Aoto and Delia Kesner, editors, *Proceedings of the 3rd International Workshop on Confluence*, pages 24–28, 2014.

32 Enno Ohlebusch. *Advanced Topics in Term Rewriting*. Springer, 2002.

33 Catuscia Palamidessi. Algebraic properties of idempotent substitutions. In Mike Paterson, editor, *Proceedings of the 17th International Colloquium on Automata, Languages and*

Programming, volume 443 of *Lecture Notes in Computer Science*, pages 386–399. Springer, 1990.

34 James R. Slagle. Automated theorem-proving for theories with simplifiers, commutativity and associativity. *Journal of the ACM*, 21(4):622–642, 1974.

35 Thomas Sternagel and Aart Middeldorp. Conditional confluence (system description). In Gilles Dowek, editor, *Proceedings of the Joint International Conference on Rewriting and Typed Lambda Calculi*, volume 8560 of *Lecture Notes in Computer Science*, pages 456–465. Springer, 2014.

36 Thomas Sternagel and Aart Middeldorp. Infeasible conditional critical pairs. In Ashish Tiwari and Takahito Aoto, editors, *Proceedings of the 4th International Workshop on Confluence*, pages 13–17, 2015.

37 Taro Suzuki, Aart Middeldorp, and Tetsuo Ida. Level-confluence of conditional rewrite systems with extra variables in right-hand sides. In Jieh Hsiang, editor, *Proceedings of the 6th International Conference on Rewriting Techniques and Applications*, volume 914 of *Lecture Notes in Computer Science*, pages 179–193. Springer, 1995.

A Example of Innermost Narrowing and Constructor-based Rewriting

▶ **Example 26.** Consider $\mathcal{R}_1$ in Section 1 again. We have infinitely many leftmost innermost narrowing derivations starting from $\mathsf{e}(x) \twoheadrightarrow \mathsf{true}$:

- $(\mathsf{e}(x) \twoheadrightarrow \mathsf{true}) \overset{li}{\rightsquigarrow}_{\{x\mapsto\mathsf{0}\},\mathcal{R}_1} (\mathsf{true} \twoheadrightarrow \mathsf{true}) \overset{li}{\rightsquigarrow}_{id,\mathcal{R}_1} \top$,
- $(\mathsf{e}(x) \twoheadrightarrow \mathsf{true}) \overset{li}{\rightsquigarrow}_{\{x\mapsto\mathsf{s}(x_1)\},\mathcal{R}_1} (\mathsf{o}(x_1) \twoheadrightarrow \mathsf{true}) \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true}) \overset{li}{\rightsquigarrow}_{\{x_1\mapsto\mathsf{0}\},\mathcal{R}_1} (\mathsf{false} \twoheadrightarrow \mathsf{true}) \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true})$,
- $(\mathsf{e}(x) \twoheadrightarrow \mathsf{true}) \overset{li}{\rightsquigarrow}_{\{x\mapsto\mathsf{s}(x_1)\},\mathcal{R}_1} (\mathsf{o}(x_1) \twoheadrightarrow \mathsf{true}) \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true}) \overset{li}{\rightsquigarrow}_{\{x_1\mapsto\mathsf{s}(x_2)\},\mathcal{R}_1} (\mathsf{e}(x_2) \twoheadrightarrow \mathsf{true}) \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true}) \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true}) \overset{li}{\rightsquigarrow}_{\{x_2\mapsto\mathsf{0}\},\mathcal{R}_1} (\mathsf{true} \twoheadrightarrow \mathsf{true}) \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true}) \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true}) \overset{li}{\rightsquigarrow}_{id,\mathcal{R}_1} \top \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true}) \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true}) \overset{li}{\rightsquigarrow}_{id,\mathcal{R}_1} \top \,\&\, \top \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true}) \overset{li}{\rightsquigarrow}_{id,\mathcal{R}_1} \top \,\&\, \top \,\&\, \top$,
- ...

The following leftmost constructor-based rewriting derivations correspond to the above narrowing derivations, respectively:

- $(\mathsf{e}(\mathsf{0}) \twoheadrightarrow \mathsf{true}) \overset{lc}{\to}_{\mathcal{R}_1} (\mathsf{true} \twoheadrightarrow \mathsf{true}) \overset{lc}{\to}_{\mathcal{R}_1} \top$,
- $(\mathsf{e}(\mathsf{s}(\mathsf{0})) \twoheadrightarrow \mathsf{true}) \overset{lc}{\to}_{\mathcal{R}_1} (\mathsf{o}(\mathsf{0}) \twoheadrightarrow \mathsf{true}) \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true}) \overset{lc}{\to}_{\mathcal{R}_1} (\mathsf{false} \twoheadrightarrow \mathsf{true}) \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true})$,
- $(\mathsf{e}(\mathsf{s}(\mathsf{s}((\mathsf{0})))) \twoheadrightarrow \mathsf{true}) \overset{lc}{\to}_{\mathcal{R}_1} (\mathsf{o}(\mathsf{s}(\mathsf{0})) \twoheadrightarrow \mathsf{true}) \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true}) \overset{lc}{\to}_{\mathcal{R}_1} (\mathsf{e}(\mathsf{0}) \twoheadrightarrow \mathsf{true}) \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true}) \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true}) \overset{lc}{\to}_{\mathcal{R}_1} (\mathsf{true} \twoheadrightarrow \mathsf{true}) \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true}) \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true}) \overset{lc}{\to}_{\mathcal{R}_1} \top \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true}) \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true}) \overset{lc}{\to}_{\mathcal{R}_1} \top \,\&\, \top \,\&\, (\mathsf{true} \twoheadrightarrow \mathsf{true}) \overset{lc}{\to}_{\mathcal{R}_1} \top \,\&\, \top \,\&\, \top$,
- ...

B Proofs of Theorems

In this appendix, we show proofs of Theorems 4, 6, 7, 19, and 22.

▶ **Theorem 4.** *Let $\mathcal{R}$ be a constructor SDCTRS, T a goal clause, and $U \in \mathcal{T}(\{\top, \&\})$.*

1. *If $T \overset{li}{\rightsquigarrow}^*_{\sigma,\mathcal{R}} U$, then $\sigma T \overset{lc}{\to}^*_{\mathcal{R}} U$ (i.e., $\sigma s \overset{lc}{\to}^*_{\mathcal{R}} \sigma t$ for all goals $s \twoheadrightarrow t$ in T).*
2. *For a constructor substitution θ, if $\theta T \overset{lc}{\to}^*_{\mathcal{R}} U$, then there exists an idempotent constructor substitution σ such that $T \overset{li}{\rightsquigarrow}^*_{\sigma,\mathcal{R}} U$ and $\sigma \leq \theta$.*

Proof. The first claim can be straightforwardly proved by induction on the length of $T \overset{li}{\leadsto}{}^*_{\sigma,\mathcal{R}} U$. In [28], the second claim is proved for a constructor SDCTRS $\mathcal{R}$ such that for each rule $\ell \to r \Leftarrow s_1 \twoheadrightarrow t_1 \mathbin{\&} \cdots \mathbin{\&} s_k \twoheadrightarrow t_k$, all $t_1, \ldots, t_k$ are constructor terms. Any rule $\ell \to r \Leftarrow s_1 \twoheadrightarrow t_1 \mathbin{\&} \cdots \mathbin{\&} s_k \twoheadrightarrow t_k$ is not used in $\overset{c}{\to}_{\mathcal{R}}$ if there exists some i such that t_i contains a defined symbol. In addition, in the proof, $\mathcal{R}$ does not have to be deterministic or a 3-CTRS. For this reason, the proof in [28, Lemma 17] can be a proof of this theorem. ◀

We show some lemmas to prove Theorem 6.

▶ **Lemma 27.** *Let $\mathcal{R}$ be a constructor SDCTRS over a signature $\mathcal{F}$ such that $\mathcal{R} = \mathcal{R}_0 \uplus \{\ell \to r \Leftarrow C\}$ where r is not a constructor term of $\mathcal{R}$. Let $\mathcal{R}' = \mathcal{R}_0 \cup \{\ell \to x \Leftarrow C \mathbin{\&} r \twoheadrightarrow x\}$, where x is a fresh variable. Then, $\mathcal{R}'$ is a constructor SDCTRS over $\mathcal{F}$ and is equivalent to $\mathcal{R}$ w.r.t. $\overset{c}{\to}$ and $\overset{i}{\leadsto}$.*

Proof. By definition, it is clear that $\mathcal{D}_{\mathcal{R}} = \mathcal{D}_{\mathcal{R}'}$. The remaining properties for equivalence w.r.t. $\overset{c}{\to}$ and $\overset{i}{\leadsto}$ can straightforwardly proved by induction on the numbers of rewriting and narrowing steps, respectively. ◀

▶ **Lemma 28.** *Let $\mathcal{R}$ be a constructor SDCTRS over a signature $\mathcal{F}$ such that $\mathcal{R} = \mathcal{R}_0 \cup \{\ell \to r \Leftarrow C_1 \mathbin{\&} s_i \twoheadrightarrow t_i \mathbin{\&} C_2\}$ where r is a constructor term, $p \neq \varepsilon$, and t_i is a ground normal form of $\mathcal{R}_u$ but not a constructor term. Let $\mathcal{R}' = \mathcal{R}_0 \cup \{\ell \to r \Leftarrow C_1 \mathbin{\&} s_i \twoheadrightarrow x \mathbin{\&} t_i \twoheadrightarrow x \mathbin{\&} C_2\}$, where x is a fresh variable. Then, $\mathcal{R}'$ is a constructor SDCTRS over $\mathcal{F}$ and is equivalent to $\mathcal{R}$ w.r.t. $\overset{c}{\to}$ and $\overset{i}{\leadsto}$.*

Proof. By definition, it is clear that $\mathcal{D}_{\mathcal{R}} = \mathcal{D}_{\mathcal{R}'}$. Since t_i contains a defined symbol, t_i is a ground normal form of $\mathcal{R}_u$ and there is no rule $\ell' \to r' \Leftarrow C'$ in $\mathcal{R}$ such that ℓ' matches a subterm of t_i. To use $\ell \to r \Leftarrow C_1 \mathbin{\&} s_i \twoheadrightarrow t_i \mathbin{\&} C_2$ in a constructor-based rewriting to a term in $\mathcal{T}(\{\top, \&\})$, an instance of $s_i \twoheadrightarrow t_i$ should be reduced to a term in $\mathcal{T}(\{\top, \&\})$. However, such a reduction is impossible for constructor-based rewriting because $t_i \twoheadrightarrow t_i$ cannot be rewritten or narrowed to $\top$. For this reason, $\ell \to r \Leftarrow C_1 \mathbin{\&} s_i \twoheadrightarrow t_i \mathbin{\&} C_2$ is never used in constructor-based rewriting or innermost narrowing to a term in $\mathcal{T}(\{\top, \&\})$. For the same reason, $\ell \to r \Leftarrow C_1 \mathbin{\&} s_i \twoheadrightarrow x \mathbin{\&} t_i \twoheadrightarrow x \mathbin{\&} C_2 \in \mathcal{R}'$ is never used in constructor-based rewriting of $\mathcal{R}'$ to terms in $\mathcal{T}(\{\top, \&\})$, either. Therefore, it is clear that for a goal clause T and a term $U \in \mathcal{T}(\{\top, \&\})$, (a) $T \overset{lc}{\to}{}^*_{\mathcal{R}} U$ if and only if $T \overset{lc}{\to}{}^*_{\mathcal{R}'} U$, and (b) $T \overset{li}{\leadsto}{}^*_{\theta,\mathcal{R}} U$ if and only if $T \overset{li}{\leadsto}{}^*_{\theta,\mathcal{R}'} U$. ◀

▶ **Lemma 29.** *Let $\mathcal{R}$ be a constructor SDCTRS over a signature $\mathcal{F}$ such that $\mathcal{R} = \mathcal{R}_0 \cup \{\ell \to r \Leftarrow C_1 \mathbin{\&} s_i[s']_p \twoheadrightarrow t_i \mathbin{\&} C_2\}$ where r is a constructor term, $p \neq \varepsilon$, and s' is rooted by a defined symbol. Let $\mathcal{R}' = \mathcal{R}_0 \cup \{\ell \to r \Leftarrow C_1 \mathbin{\&} s' \twoheadrightarrow x \mathbin{\&} s_i[x]_p \twoheadrightarrow t_i \mathbin{\&} C_2\}$, where x is a fresh variable. Then, $\mathcal{R}'$ is a constructor SDCTRS over $\mathcal{F}$ and is equivalent to $\mathcal{R}$ w.r.t. $\overset{c}{\to}$ and $\overset{i}{\leadsto}$.*

Proof. By definition, it is clear that $\mathcal{D}_{\mathcal{R}} = \mathcal{D}_{\mathcal{R}'}$. The remaining properties for equivalence w.r.t. $\overset{c}{\to}$ and $\overset{i}{\leadsto}$ can straightforwardly proved by induction on the numbers of rewriting and narrowing steps, respectively. ◀

▶ **Theorem 6.** *Let $\mathcal{R}$ be a constructor SDCTRS over a signature $\mathcal{F}$. Then, $Pc(\mathcal{R})$ is a pc-CTRS over $\mathcal{F}$ and is equivalent to $\mathcal{R}$ w.r.t. $\overset{c}{\to}$ and $\overset{i}{\leadsto}$.*

Proof. By definition, it is clear that $Pc(\mathcal{R})$ is a pc-CTRS over the signature $\mathcal{F}$ and $\mathcal{D}_{\mathcal{R}} = \mathcal{D}_{Pc(\mathcal{R})}$. The remaining properties can be proved by Lemmas 27, 28, and 29, and [26, Theorem 26]. ◀

▶ **Theorem 7.** *Let $\mathcal{R}$ be a pc-CTRS over a signature $\mathcal{F}$. Then, $Trs(\mathcal{R})$ is a constructor TRS over $\mathcal{F} \cup \{\twoheadrightarrow, \top, \&\}$ and is equivalent to $\mathcal{R}$ w.r.t. $\xrightarrow{c}$ and $\overset{i}{\rightsquigarrow}$.*

Proof. By definition, it is clear that (1) $Trs(\mathcal{R})$ is a constructor TRS over $\mathcal{F} \cup \{\twoheadrightarrow, \top, \&\}$, (2) $\mathcal{D}_\mathcal{R} = \mathcal{D}_{Trs(\mathcal{R})}$, (3) $(s \twoheadrightarrow t) \xrightarrow{c}_\mathcal{R} T$ if and only if $(s \twoheadrightarrow t) \xrightarrow{c}_{Trs(\mathcal{R})} T$ and (4) $(s \twoheadrightarrow t) \overset{i}{\rightsquigarrow}_{\theta,\mathcal{R}} T$ if and only if $(s \twoheadrightarrow t) \overset{i}{\rightsquigarrow}_{\theta,Trs(\mathcal{R})} T$. Using (3), we can prove that for a goal clause T and a term $U \in \mathcal{T}(\{\top, \&\})$, $T \xrightarrow{lc}{}^*_\mathcal{R} U$ if and only if $T \xrightarrow{lc}{}^*_{\mathcal{R}'} U$, by induction on the number of rewriting steps. Using (4), we can prove that for a goal clause T and a term $U \in \mathcal{T}(\{\top, \&\})$, $T \overset{li}{\rightsquigarrow}{}^*_{\theta,\mathcal{R}} U$ if and only if $T \overset{li}{\rightsquigarrow}{}^*_{\theta,\mathcal{R}'} U$, by induction on the number of narrowing steps. ◀

▶ **Theorem 19.** *Let T be a goal clause without $\top$. Then, $[\![L(SSG(\mathcal{R}, T), \Gamma_T)]\!] = Suc(\overset{i}{\rightsquigarrow}_\mathcal{R}, T)$ up to variable renaming.*

Proof. Thanks to Theorems 12, 13, and 14, a constructor substitution θ for $T \overset{i}{\rightsquigarrow}{}^*_{\theta,\mathcal{R}} U \in \mathcal{T}(\{\top, \&\})$ can be obtained by (1) splitting, (2) flattening, and (3) narrowing applied to goals of the form $f(x_1, \ldots, x_n) \twoheadrightarrow u$ such that $u \in (\mathcal{V} \setminus \{x_1, \ldots, x_n\}) \cup \mathcal{T}(\mathcal{C}_\mathcal{R})$ and $x_1, \ldots, x_n$ are distinct variables. The application of these operations is exactly the same as the application of production rules in $SSG(\mathcal{R}, T)$. Therefore, this theorem holds. ◀

The following lemma is used to prove Theorem 22.

▶ **Lemma 30.** *Let $\mathcal{G}$ be an SSG $(\Gamma_{T_0}, \mathcal{N}, \mathcal{P})$, $\Gamma_T \in \mathcal{N}$, $x \in \mathcal{V}ar(T)$, and $RTG(\mathcal{G}, T, x)$ be constructed. Then, $\{\xi\theta x \mid \theta \in [\![L(\mathcal{G}, \Gamma_T)]\!],\ \xi \in Subst(\mathcal{C}_\mathcal{R}),\ \mathcal{D}om(\eta) = \mathcal{V}ar(\theta x)\} \subseteq L(RTG(\mathcal{G}, T, x))$.*

Proof. Suppose that e is generated by n steps of applying production rules obtained from $\mathcal{P}|_{\Gamma_T}$. Then, it is easy to prove this lemma by induction on n. ◀

The converse inclusion in Lemma 30 does not hold in general even if $[\![L(\mathcal{G}, \Gamma_T)]\!]$ is a set of ground substitutions.

▶ **Example 31.** Consider an SSG $\mathcal{G}_4 = (\Gamma_{x\twoheadrightarrow\mathsf{a}}, \{\Gamma_{x\twoheadrightarrow\mathsf{a}}, \Gamma_{z\twoheadrightarrow\mathsf{b}}\}, \{\ \Gamma_{x\twoheadrightarrow\mathsf{a}} \to \textsc{rec}(\Gamma_{z\twoheadrightarrow\mathsf{b}}, \{y \mapsto z\}) \bullet \{x \mapsto \mathsf{c}(y, y)\},\ \Gamma_{z\twoheadrightarrow\mathsf{b}} \to \{z \mapsto \mathsf{a}\} \mid \{z \mapsto \mathsf{b}\}\ \})$. For goal $x \twoheadrightarrow \mathsf{a}$ and variable x, we have the regular tree grammar $RTG(\mathcal{G}_4, x \twoheadrightarrow \mathsf{a}, x) = (\Gamma^x_{x\twoheadrightarrow\mathsf{a}}, \{\Gamma^x_{x\twoheadrightarrow\mathsf{a}}, \Gamma^z_{z\twoheadrightarrow\mathsf{b}}\}, \{\ \Gamma^x_{x\twoheadrightarrow\mathsf{a}} \to \mathsf{c}(\Gamma^z_{z\twoheadrightarrow\mathsf{b}}, \Gamma^z_{z\twoheadrightarrow\mathsf{b}}),\ \Gamma^z_{z\twoheadrightarrow\mathsf{b}} \to \mathsf{a} \mid \mathsf{b}\ \})$. The term $\mathsf{c}(\mathsf{a}, \mathsf{b})$ is included in $L(RTG(\mathcal{G}_4, x \twoheadrightarrow \mathsf{a}, x), \Gamma^x_{x\twoheadrightarrow\mathsf{a}})$, but there is no substitution θ such that $\theta x = \mathsf{c}(\mathsf{a}, \mathsf{b})$ and $\sigma \leq \theta$ for some σ in $[\![L(\mathcal{G}_4, \Gamma_{x\twoheadrightarrow\mathsf{a}})]\!] = \{\ \{x \mapsto \mathsf{c}(\mathsf{a}, \mathsf{a})\},\ \{x \mapsto \mathsf{c}(\mathsf{b}, \mathsf{b})\}\ \}$.

▶ **Theorem 22.** *Let $\mathcal{G}$ be an SSG $(\Gamma_{T_0}, \mathcal{N}, \mathcal{P})$, $\Gamma_{T_1}, \Gamma_{T_2} \in \mathcal{N}$, $x \in \mathcal{V}$, $x_1 \in \mathcal{V}ar(T_1)$, $x_2 \in \mathcal{V}ar(T_2)$, $RTG(\mathcal{G}, T_1, x_1), RTG(\mathcal{G}, T_2, x_2)$ be constructed, and δ_1, δ_2 be renamings such that $\mathcal{VR}an(\delta_i) = \mathcal{V}ar(T_i)$ and $\delta_i x = x_i$ for $i = 1, 2$. If $L(RTG(\mathcal{G}, T_1, x_1)) \cap L(RTG(\mathcal{G}, T_2, x_2)) = \emptyset$, then $\{\![\ \textsc{rec}(\Gamma_{T_1}, \delta_1)\ \&\ \textsc{rec}(\Gamma_{T_2}, \delta_2)\ ]\!\}_\mathcal{G} = \{\![\ \varnothing\ ]\!\}_\mathcal{G}$.*

Proof. We proceed by contradiction. Assume that $L(RTG(\mathcal{G}, T_1, x_1)) \cap L(RTG(\mathcal{G}, T_2, x_2)) = \emptyset$ and $\{\![\ \textsc{rec}(\Gamma_{T_1}, \delta_1)\ \&\ \textsc{rec}(\Gamma_{T_2}, \delta_2)\ ]\!\}_\mathcal{G} \neq \{\![\ \varnothing\ ]\!\}_\mathcal{G}$. Then, there exists a constructor substitution $\theta \in \{\![\ \textsc{rec}(\Gamma_{T_1}, \delta_1)\ \&\ \textsc{rec}(\Gamma_{T_2}, \delta_2)\ ]\!\}_\mathcal{G}$, and hence there exist constructor substitutions θ_1, θ_2 such that $\theta_1 \in \{\![\ \Gamma_{T_1}\ ]\!\}_\mathcal{G}$, $\theta_2 \in \{\![\ \Gamma_{T_2}\ ]\!\}_\mathcal{G}$, and $\theta = (\theta_1 \cdot \delta_1) \Uparrow (\theta_2 \cdot \delta_2)$. Thus, it follows from Lemma 30 that $\xi\theta x \in L(RTG(\mathcal{G}, T_1, x_1)) \cap L(RTG(\mathcal{G}, T_2, x_2))$ for some $\xi \in Subst(\mathcal{C}_\mathcal{R})$. This contradicts the assumption. ◀