On Transforming Narrowing Trees into Regular Tree Grammars Generating Ranges of Substitutions*

Naoki Nishida
Graduate School of Informatics
Nagoya University
Nagoya, Japan
nishida@i.nagoya-u.ac.jp

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

The grammar representation of a narrowing tree for a syntactically deterministic conditional term rewriting system and a pair of terms is a regular tree grammar that generates expressions for substitutions obtained by all possible innermost-narrowing derivations that start with the pair and end with particular non-narrowable terms. In this paper, under a certain syntactic condition, we show a transformation of the grammar representation of a narrowing tree into another regular tree grammar that overapproximately generates the ranges of ground substitutions generated by the grammar representation. In our previous work, such a transformation is restricted to the ranges w.r.t. a given single variable, and thus, the usefulness is limited. We extend the previous transformation by representing the range of a ground substitution as a tuple of terms, which is obtained by the coding for finite trees. We show a precise definition of the transformation and prove that the language of the transformed regular tree grammar is an overapproximation of the ranges of ground substitutions generated by the grammar representation. We leave an experiment to evaluate the usefulness of the transformation as future work.

1 Introduction

Conditional term rewriting [25, Chapter 7] is known to be more complicated than unconditional term rewriting in the sense of analyzing properties, e.g., *operational termination* [17], *confluence* [29], and *reachability* [5]. A popular approach to the analysis of conditional rewriting is to transform a conditional term rewriting system (a CTRS, for short) into an unconditional term rewriting system (a TRS, for short) that is in general an overapproximation of the CTRS in terms of reduction. This approach enables us to use existing techniques for the analysis of TRSs. For example, a CTRS is operationally terminating if the *unraveled* TRS [18, 25] is terminating [4]. To prove termination of the unraveled TRS, we can use many techniques for proving termination of TRSs (cf. [25]). On the other hand, it is not so easy to analyze *reachability* which is relevant to, e.g., *(in)feasibility* of conditions.

Let us consider to prove confluence of the following *syntactically deterministic 3*-CTRS [25, Example 7.1.5] defining the *gcd* operator over the natural numbers represented by $\mathsf{0}$ and s:

$$\mathcal{R}_1 = \left\{ \begin{array}{rlr} x < \mathsf{0} \to \mathsf{false}, & \mathsf{0} - \mathsf{s}(y) \to \mathsf{0}, & \\ \mathsf{0} < \mathsf{s}(y) \to \mathsf{true}, & x - \mathsf{0} \to x, & \\ \mathsf{s}(x) < \mathsf{s}(y) \to x < y, & \mathsf{s}(x) - \mathsf{s}(y) \to x - y, & \\ \mathsf{gcd}(x,x) \to x, & & \\ \mathsf{gcd}(\mathsf{s}(x),\mathsf{0}) \to \mathsf{s}(x), & \mathsf{gcd}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{gcd}(x-y,\mathsf{s}(y)) \Leftarrow y < x \twoheadrightarrow \mathsf{true}, & \\ \mathsf{gcd}(\mathsf{0},\mathsf{s}(y)) \to \mathsf{s}(y), & \mathsf{gcd}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{gcd}(\mathsf{s}(x),y-x) \Leftarrow x < y \twoheadrightarrow \mathsf{true} & \end{array} \right\}$$

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

J. Niehren & D. Sabel (eds.): Rewriting Techniques
for Program Transformations and Evaluation (WPTE 2018)
EPTCS 289, 2019, pp. 68–87, doi:10.4204/EPTCS.289.5

A transformational approach in [11, 10] does not succeed in proving confluence of $\mathcal{R}_1$. On the other hand, a direct approach to reachability analysis to prove *infeasibility* of the *conditional critical pairs* (i.e., non-existence of substitutions satisfying conditions), which is implemented in some confluence provers, does not prove confluence of $\mathcal{R}_1$ well, either. Let us consider the critical pairs of $\mathcal{R}_1$:

$$\begin{array}{lllll}
\langle & \mathsf{s}(x), & \mathsf{gcd}(x-x,\mathsf{s}(x)) & \rangle \Leftarrow x < x \twoheadrightarrow \mathsf{true}, \\
\langle & \mathsf{gcd}(x-x,\mathsf{s}(x)), & \mathsf{s}(x) & \rangle \Leftarrow x < x \twoheadrightarrow \mathsf{true}, \\
\langle & \mathsf{s}(x), & \mathsf{gcd}(\mathsf{s}(x),x-x) & \rangle \Leftarrow x < x \twoheadrightarrow \mathsf{true}, \\
\langle & \mathsf{gcd}(\mathsf{s}(x),x-x), & \mathsf{s}(x) & \rangle \Leftarrow x < x \twoheadrightarrow \mathsf{true}, \\
\langle & \mathsf{gcd}(x-y,\mathsf{s}(y)), & \mathsf{gcd}(\mathsf{s}(x),y-x) & \rangle \Leftarrow x < y \twoheadrightarrow \mathsf{true},\ y < x \twoheadrightarrow \mathsf{true}, \\
\langle & \mathsf{gcd}(\mathsf{s}(x),y-x), & \mathsf{gcd}(x-y,\mathsf{s}(y)) & \rangle \Leftarrow x < y \twoheadrightarrow \mathsf{true},\ y < x \twoheadrightarrow \mathsf{true}
\end{array}$$

Note that the above critical pairs are symmetric because they are caused by overlaps at the root position only. An operationally terminating CTRS is confluent if all critical pairs of the CTRS are infeasible (cf. [1, 3]). Operational termination of $\mathcal{R}_1$ can be proved by, e.g., AProVE [8]. To prove infeasibility of the critical pairs above, it suffices to show both (i) non-existence of terms t such that $t < t \to^*_{\mathcal{R}_1} \mathsf{true}$, and (ii) non-existence of terms t_1,t_2 such that $t_1 < t_2 \to^*_{\mathcal{R}_1} \mathsf{true}$ and $t_2 < t_1 \to^*_{\mathcal{R}_1} \mathsf{true}$. Thanks to the meaning of $<$, it would be easy for a human to notice that such terms t,t_1,t_2 do not exist. However, it is not so easy to mechanize a way to show non-existence of t,t_1,t_2. In fact, confluence provers for CTRSs, ConCon [28], CO3 [20], and CoScart [9], based on e.g., transformations of CTRSs into TRSs or reachability analysis for infeasibility of conditional critical pairs, failed to prove confluence of $\mathcal{R}_1$ (see Confluence Competition 2016, 2017, and 2018,[1] `327.trs`). In addition, a *semantic approach* in [16, 15] cannot prove confluence of $\mathcal{R}_1$ using AGES [12], a tool for generating logical models of order-sorted first-order theories—non-existence of t_1,t_2 above cannot be proved via its web interface with default parameters. Timbuk 3.2 [7], which is based on tree automata techniques [6], cannot prove infeasibility of $x < y \twoheadrightarrow \mathsf{true},\ y < x \twoheadrightarrow \mathsf{true}$ w.r.t. the rules for $<$ under the default use.

The non-existence of a term t with $t < t \to^*_{\mathcal{R}_1} \mathsf{true}$ can be reduced to the non-existence of substitutions θ such that $x < x \rightsquigarrow^*_{\theta,\mathcal{R}_1} \mathsf{true}$, where $\rightsquigarrow$ denotes the *narrowing* step [14]—for example, $x < y \rightsquigarrow_{\{x\mapsto 0,\ y\mapsto \mathsf{s}(y')\},\mathcal{R}_1} \mathsf{true}$. In addition, the non-existence of such substitutions can be reduced to the emptiness of the set of the substitutions, i.e., the emptiness of $\{\theta \mid x < x \rightsquigarrow^*_{\theta,\mathcal{R}_1} \mathsf{true}\}$. From this viewpoint, for a pair of terms, the enumeration of substitutions obtained by narrowing would be useful in analyzing rewriting that starts with instances of the pair. To analyze sets of substitutions derived by *innermost* narrowing, *narrowing trees* [23] are useful. For example, infeasibility of conditional critical pairs of some normal 1-CTRS can be proved by using the *grammar representation* of a narrowing tree [21]. Simplification of the grammar representation implies the non-existence of substitutions satisfying the conditional part of a critical pair. However, there are some examples (shown later) for which the simplification method in [21] does not succeed in converting grammar representations to those explicitly representing the empty set.

In this paper, under a certain syntactic condition, we show a transformation of the grammar representation of a narrowing tree into a *regular tree grammar* [2] (an RTG, for short) that overapproximately generates the ranges of ground substitutions generated by the grammar representation. The aim of the transformation is to simplify grammar representations as much as possible together with the existing one in [21].

Let $\mathcal{R}$ be a *syntactically deterministic* 3-CTRS (a 3-SDCTRS, for short) that is a constructor system, s a *basic* term, and t a constructor term, where basic terms are of the form $f(u_1,\ldots,u_n)$ with a defined

[1] `http://cops.uibk.ac.at/results/?y=2018&c=CTRS`

symbol f and constructor terms $u_1,\dots,u_n$. A *narrowing tree* [23, 21] of $\mathcal{R}$ with the root pair $s \twoheadrightarrow t$ is a finite representation that defines the set of substitutions θ such that the pair $s \twoheadrightarrow t$ narrows to a particular ground term $u_\top$ consisting of a special binary symbol & and a special constant $\top$ by *innermost* narrowing $\overset{i}{\leadsto}_{\mathcal{R}}$ with a substitution θ (i.e., $(s \twoheadrightarrow t) \overset{i}{\leadsto}{}^*_{\theta,\mathcal{R}} u_\top$ and thus $\theta s \xrightarrow{c}{}^*_{\mathcal{R}} \theta t$). Note that $\twoheadrightarrow$ is considered a binary symbol, $(x \twoheadrightarrow x) \to \top$ is assumed to be implicitly included in $\mathcal{R}$, and $\xrightarrow{c}_{\mathcal{R}}$ denotes the *constructor-based rewriting* step which applies rewrite rules to basic terms. Such a narrowing tree can be the enumeration of substitutions obtained by innermost narrowing of $\mathcal{R}$ to ground terms consisting of & and $\top$. The idea of narrowing trees has been extended to finite representations of SLD trees for logic programs [24].

Using narrowing trees, it is easy to see that there is no substitution θ such that $x < x \overset{i}{\leadsto}{}^*_{\theta,\mathcal{R}_1}$ true, and hence the above four critical pairs with $x < x \twoheadrightarrow \mathsf{true}$ are infeasible. Let us now consider to prove infeasibility of $x < y \twoheadrightarrow \mathsf{true}$, $y < x \twoheadrightarrow \mathsf{true}$. A narrowing tree for $x < y \twoheadrightarrow \mathsf{true}\ \&\ y < x \twoheadrightarrow \mathsf{true}$ can be represented by the following *grammar representation* [23, 21] that can be considered an RTG (see Section 4):

$$\begin{aligned} \Gamma_{x<y\twoheadrightarrow\mathsf{true}\&y<x\twoheadrightarrow\mathsf{true}} &\to \Gamma_{x<y\twoheadrightarrow\mathsf{true}} \,\&\, \Gamma_{y<x\twoheadrightarrow\mathsf{true}} \\ \Gamma_{x<y\twoheadrightarrow\mathsf{true}} &\to \{x \mapsto \mathsf{0},\ y \mapsto \mathsf{s}(y_2)\} \\ &\quad \mid \textsc{rec}(\Gamma_{x<y\twoheadrightarrow\mathsf{true}}, \{x_3 \mapsto x,\ y_3 \mapsto y\}) \bullet \{x \mapsto \mathsf{s}(x_3),\ y \mapsto \mathsf{s}(y_3)\} \\ \Gamma_{y<x\twoheadrightarrow\mathsf{true}} &\to \textsc{rec}(\Gamma_{x<y\twoheadrightarrow\mathsf{true}}, \{x \mapsto y,\ y \mapsto x\}) \end{aligned} \tag{1}$$

We denote by $\mathcal{G}_1$ the RTG with the initial non-terminal $\Gamma_{x<y\twoheadrightarrow\mathsf{true}\&y<x\twoheadrightarrow\mathsf{true}}$, the other non-terminals $\Gamma_{x<y\twoheadrightarrow\mathsf{true}}, \Gamma_{y<x\twoheadrightarrow\mathsf{true}}$, and the above production rules. We also denote by $\mathcal{P}_1$ the set of the above production rules, i.e., (1). Substitutions are considered constants, and the RTG generates terms over &, $\varnothing$, $\bullet$, REC, and substitutions. The binary symbols $\bullet$ and & are interpreted by standard composition and *parallel composition* [13, 26], respectively. Parallel composition $\Uparrow$ of two substitutions returns a most general unifier of the substitutions if the substitutions are unifiable (see Definition 4.2). 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. The symbol REC is used for recursion, which is interpreted as standard composition of a renaming and a substitution recursively generated. To simplify the discussion in the remainder of this section, following the meaning of the operators, we simplify the rules of $\Gamma_{x<y\twoheadrightarrow\mathsf{true}}$ and $\Gamma_{y<x\twoheadrightarrow\mathsf{true}}$ as follows:

$$\begin{aligned} \Gamma_{x<y\twoheadrightarrow\mathsf{true}} &\to \{x \mapsto \mathsf{0},\ y \mapsto \mathsf{s}(y_2)\} \mid \Gamma_{x<y\twoheadrightarrow\mathsf{true}} \bullet \{x \mapsto \mathsf{s}(x),\ y \mapsto \mathsf{s}(y)\} \\ \Gamma_{y<x\twoheadrightarrow\mathsf{true}} &\to \Gamma_{x<y\twoheadrightarrow\mathsf{true}} \bullet \{x \mapsto y,\ y \mapsto x\} \end{aligned} \tag{2}$$

In our previous work [21], to show the emptiness of the set of substitutions generated from e.g., $\Gamma_{x<y\twoheadrightarrow\mathsf{true}} \,\&\, \Gamma_{y<x\twoheadrightarrow\mathsf{true}}$, we transform the grammar representation to an RTG that overapproximately generates the ranges of ground substitutions w.r.t. a single variable. For example, for x, the production rules of (2) is transformed into the following ones:

$$\begin{array}{ll} \Gamma^x_{x<y\twoheadrightarrow\mathsf{true}} \to \mathsf{0} \mid \mathsf{s}(\Gamma^x_{x<y\twoheadrightarrow\mathsf{true}}) & \Gamma^x_{y<x\twoheadrightarrow\mathsf{true}} \to \Gamma^y_{x<y\twoheadrightarrow\mathsf{true}} \\ \Gamma^y_{x<y\twoheadrightarrow\mathsf{true}} \to \mathsf{s}(A) \mid \mathsf{s}(\Gamma^y_{x<y\twoheadrightarrow\mathsf{true}}) & A \to \mathsf{0} \mid \mathsf{s}(A) \mid \mathsf{true} \mid \mathsf{false} \end{array}$$

Note that non-terminal A generates arbitrary ground constructor terms. Since we focus on x only, non-terminals $\Gamma^x_{x<y\twoheadrightarrow\mathsf{true}}$ and $\Gamma^x_{y<x\twoheadrightarrow\mathsf{true}}$ generate $\{\mathsf{s}^n(a) \mid n \geq 0,\ a \in \{\mathsf{0},\mathsf{true},\mathsf{false}\}\}$ and $\{\mathsf{s}^n(a) \mid n > 0,\ a \in \{\mathsf{0},\mathsf{true},\mathsf{false}\}\}$, respectively, and we cannot prove that there is no substitution generated from $\Gamma_{x<y\twoheadrightarrow\mathsf{true}} \,\&\, \Gamma_{y<x\twoheadrightarrow\mathsf{true}}$.

In this paper, we aim at showing that there is no substitution generated by (2) from the initial non-terminal $\Gamma_{x<y\twoheadrightarrow\mathsf{true}\&y<x\twoheadrightarrow\mathsf{true}}$, i.e., showing that $L(\mathcal{G}_1, \Gamma_{x<y\twoheadrightarrow\mathsf{true}}) \cap L(\mathcal{G}_1, \Gamma_{y<x\twoheadrightarrow\mathsf{true}}) = \emptyset$. To this end,

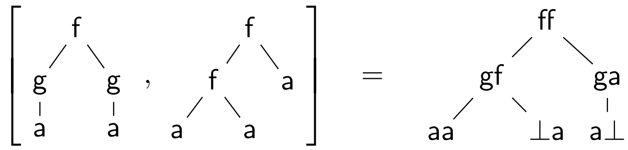

Figure 1: the coding of $\mathsf{f}(\mathsf{g}(\mathsf{a}),\mathsf{g}(\mathsf{a}))$ and $\mathsf{f}(\mathsf{f}(\mathsf{a},\mathsf{a}),\mathsf{a})$.

under a certain syntactic condition, we show a transformation of the grammar representation of a narrowing tree into an RTG that overapproximately generates the ranges of ground substitutions generated by the grammar representation (Section 5). More precisely, using the idea of *coding* for tuples of ground terms [2, Section 3.2.1] (see Figure 1), we extend a transformation in [21] w.r.t. a single variable to two variables. It is straightforward to further extend the transformation to three or more variables. We do not explain how to, given a constructor 3-SDCTRS, construct (the grammar representation of) a narrowing tree, and concentrate on how to transform a grammar representation into an RTG that generates the ranges of ground substitutions generated by the grammar representation.

Outline of Our Approach Using the rules of (2), we briefly illustrate the outline of the transformation. Roughly speaking, we apply the coding for tuples of terms to the range of substitutions, e.g., $\mathsf{0}$ and $\mathsf{s}(y_2)$ for $\{x \mapsto \mathsf{0},\ y \mapsto \mathsf{s}(y_2)\}$. The rules for $\Gamma_{x<y\twoheadrightarrow\mathsf{true}}$ are transformed into

$$\Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}} \to \mathsf{0s}(\bot A) \qquad \Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}} \to \mathsf{ss}(\Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}}).$$

where the non-terminal $\bot A$ generates ground terms obtained by applying the coding to $\bot$ and ground constructor terms. The coding of $\mathsf{s}(x)$ and $\mathsf{s}(y)$ is $\mathsf{ss}(xy)$. Variables x,y are instantiated by substitutions generated from $\Gamma_{x<y\twoheadrightarrow\mathsf{true}}$, and hence we replaced xy by $\Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}}$. The rule for $\Gamma_{y<x\twoheadrightarrow\mathsf{true}}$ is transformed into

$$\Gamma^{(x,y)}_{y<x\twoheadrightarrow\mathsf{true}} \to \Gamma^{(y,x)}_{x<y\twoheadrightarrow\mathsf{true}}.$$

Since x,y are swapped by $\{x \mapsto y,\ y \mapsto x\}$, we generate a new non-terminal $\Gamma^{(y,x)}_{x<y\twoheadrightarrow\mathsf{true}}$ and its rules as well as the above rules:

$$\Gamma^{(y,x)}_{x<y\twoheadrightarrow\mathsf{true}} \to \mathsf{s0}(A\bot) \qquad \Gamma^{(y,x)}_{x<y\twoheadrightarrow\mathsf{true}} \to \mathsf{ss}(\Gamma^{(y,x)}_{x<y\twoheadrightarrow\mathsf{true}}).$$

where the non-terminal $A\bot$ generates ground terms obtained by applying the coding to ground constructor terms and $\bot$. Every ground term generated from $\Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}}$ contains $\mathsf{0s}$, and every ground term generated from $\Gamma^{(x,y)}_{y<x\twoheadrightarrow\mathsf{true}}$ contains $\mathsf{s0}$. Neither $\mathsf{0s}$ nor $\mathsf{s0}$ is shared by the languages of $\Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}}$ and $\Gamma^{(x,y)}_{y<x\twoheadrightarrow\mathsf{true}}$, and hence there is no substitution which corresponds to an expression generated from $\Gamma_{x<y\twoheadrightarrow\mathsf{true}\&y<x\twoheadrightarrow\mathsf{true}}$. For this reason, we can transform $\Gamma_{x<y\twoheadrightarrow\mathsf{true}\&y<x\twoheadrightarrow\mathsf{true}}$ of (1) into

$$\Gamma_{x<y\twoheadrightarrow\mathsf{true}\&y<x\twoheadrightarrow\mathsf{true}} \to \varnothing$$

which means that there exist no constructor substitution θ satisfying the condition $x < y \twoheadrightarrow \mathsf{true}\ \&\ y < x \twoheadrightarrow \mathsf{true}$ under the constructor-based rewriting.

One may think that tuples of terms are enough for our goal. However, substitutions are generated by standard compositions, and tuples makes us introduce composition of tuples. For example, the range of $\sigma = \{x \mapsto \mathsf{f}(x',\mathsf{g}(\mathsf{a})),\ y \mapsto \mathsf{f}(y',\mathsf{a})\}$ is represented as a tuple $\mathsf{tup}_2(\mathsf{f}(x',\mathsf{g}(\mathsf{a})),\mathsf{f}(y',\mathsf{a}))$, where tup_2 is a binary symbol for tuples of two terms. To apply $\theta = \{x' \mapsto \mathsf{g}(\mathsf{a}),\ y' \mapsto \mathsf{f}(\mathsf{a},\mathsf{a})\}$ to the tuple, we reconstruct a tuple from $\mathsf{tup}_2(\mathsf{f}(x',\mathsf{g}(\mathsf{a})),\mathsf{f}(y',\mathsf{a}))$ and θ. On the other hand, the coding of terms makes us avoid the reconstruction and use standard composition of substitutions to compute the range of composed substitution. For example, σ and θ can be represented by $\{xy \mapsto \mathsf{ff}(x'y',\mathsf{ga}(\mathsf{a}\bot))\}$ and $\{x'y' \mapsto \mathsf{gf}(\mathsf{aa},\bot\mathsf{a})\}$, respectively, where both xy and $x'y'$ are considered single variables.

Using the rules for $\Gamma_{x<y\twoheadrightarrow\mathsf{true}}$ of (2), we further show that the weakness of the above approach of using tuples. Let us try to transform the rules of $\Gamma_{x<y\twoheadrightarrow\mathsf{true}}$ into an RTG that generates $\{\mathsf{tup}_2(\mathsf{s}^m(0),\mathsf{s}^n(a)) \mid 0 \leq m < n,\ a \in \{0,\mathsf{true},\mathsf{false}\}\}$. The first rule $\Gamma_{x<y\twoheadrightarrow\mathsf{true}} \to \{x \mapsto 0,\ y \mapsto \mathsf{s}(y_2)\}$ is transformed into $\Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}} \to \mathsf{tup}_2(0,\mathsf{s}(A))$ with the rules of A above. The second rule $\Gamma_{x<y\twoheadrightarrow\mathsf{true}} \to \Gamma_{x<y\twoheadrightarrow\mathsf{true}} \bullet \{x \mapsto \mathsf{s}(x),\ y \mapsto \mathsf{s}(y)\}$ is transformed into $\Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}} \to \mathsf{tup}_2(\mathsf{s}(\Gamma^{x}_{x<y\twoheadrightarrow\mathsf{true}}),\mathsf{s}(\Gamma^{y}_{x<y\twoheadrightarrow\mathsf{true}}))$ with the rules of $\Gamma^{x}_{x<y\twoheadrightarrow\mathsf{true}}$ and $\Gamma^{y}_{x<y\twoheadrightarrow\mathsf{true}}$ above. These rules generates not only terms in $\{\mathsf{tup}_2(\mathsf{s}^m(0),\mathsf{s}^n(a)) \mid 0 \leq m < n,\ a \in \{0,\mathsf{true},\mathsf{false}\}\}$ but also other terms, e.g., $\mathsf{tup}_2(\mathsf{s}(0),\mathsf{s}(0))$. The term $\mathsf{tup}_2(\mathsf{s}(0),\mathsf{s}(0))$ should not be generated because the term can be a common element generated by $\Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}}$ and $\Gamma^{(x,y)}_{y<x\twoheadrightarrow\mathsf{true}}$ and we cannot prove $\Gamma_{x<y\twoheadrightarrow\mathsf{true}}\ \&\ \Gamma_{y<x\twoheadrightarrow\mathsf{true}}$ does not generate any substitution.

2 Preliminaries

In this section, we recall basic notions and notations of term rewriting [1, 25] and regular tree grammars [2]. Familiarity with basic notions on term rewriting [1, 25] is assumed.

2.1 Terms and Substitutions

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, denoted by $\mathit{arity}(f)$. 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 $\mathcal{V}ar(t_1,\ldots,t_n)$. We denote the set of positions of a term t by $\mathcal{P}os(t)$. For a term t and a position p of t, the *subterm* of t at p is denoted by $t|_p$. The function symbol at the *root* position ε of a term t is denoted by $\mathit{root}(t)$. 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 $\mathcal{VR}an(\sigma)$: $\mathcal{VR}an(\sigma) = \bigcup_{x\in\mathcal{D}om(\sigma)} \mathcal{V}ar(\sigma x)$. 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) of E is denoted by $\mathit{mgu}(E)$ 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, denoted by θE, is defined as $\theta E = \{\theta s \approx \theta t \mid s \approx t \in E\}$.

2.2 Conditional Rewriting

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 $s_1 \twoheadrightarrow t_1,\dots,s_k \twoheadrightarrow t_k$ of term pairs ($k \geq 0$) where $s_1,t_1,\dots,s_k,t_k \in \mathcal{T}(\mathcal{F},\mathcal{V})$. 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* (a TRS, for short) 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 $\to_{(i+1),\mathcal{R}} = \{(s[\sigma\ell]_p, s[\sigma r]_p) \mid s \in \mathcal{T}(\mathcal{F},\mathcal{V}),\ \ell \to r \Leftarrow s_1 \twoheadrightarrow t_1,\dots,s_k \twoheadrightarrow t_k \in \mathcal{R},\ \sigma s_1 \to^*_{(i),\mathcal{R}} \sigma t_1,\ \dots,\ \sigma s_k \to^*_{(i),\mathcal{R}} \sigma t_k\}$ 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 3* (3-CTRS, for short) if every rule $\ell \to r \Leftarrow c \in \mathcal{R}$ satisfies that $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(\ell,c)$. $\mathcal{V}ar(s_i) \subseteq \mathcal{V}ar(\ell,t_1,\dots,t_{i-1})$ 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: $\mathcal{D}_\mathcal{R} = \{root(\ell) \mid \ell \to r \Leftarrow c \in \mathcal{R}\}$ 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 $Subst(\mathcal{C}_\mathcal{R},\mathcal{V})$ is called a *constructor substitution of* $\mathcal{R}$. A term of the form $f(t_1,\dots,t_n)$ with $f/n \in \mathcal{D}_\mathcal{R}$ and $t_1,\dots,t_n \in \mathcal{T}(\mathcal{C}_\mathcal{R},\mathcal{V})$ 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 3-DCTRS $\mathcal{R}$ is called *syntactically deterministic* (an SDCTRS, for short) if for every rule $\ell \to r \Leftarrow s_1 \twoheadrightarrow t_1,\dots,s_k \twoheadrightarrow t_k \in \mathcal{R}$, 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 [17]—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 such that $t_1 \leftarrow^*_\mathcal{R} \cdot \to^*_\mathcal{R} t_2$.

2.3 Innermost Conditional Narrowing

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 $s \twoheadrightarrow t$. In addition, 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 bracket it such as $(s \twoheadrightarrow t)$. As in [19], any CTRS in this paper is assumed to implicitly include the rule $(x \twoheadrightarrow x) \to \top$ where $\top$ is a special constant. The rule $(x \twoheadrightarrow x) \to \top$ is used to test structural equivalence between two terms t_1,t_2 by means of $t_1 \twoheadrightarrow t_2$.

To deal with a conjunction of pairs $e_1,\dots,e_k$ of terms (e_i is either $s_i \twoheadrightarrow t_i$ or $\top$) as a term, we write $e_1 \,\&\, \cdots \,\&\, e_k$ by using an associative binary symbol &. We call such a term an *equational term*. Unlike [23], to avoid & to be a defined symbol, we do not use any rule for &, e.g., $(\top \,\&\, x) \to x$. 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, for a constructor 3-SDCTRS $\mathcal{R}$, a pair $s \twoheadrightarrow t$ of terms is called a *goal* of $\mathcal{R}$ 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. An equational term is called a *goal clause* of $\mathcal{R}$ 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 2.1 The equational term $x < y \twoheadrightarrow \mathsf{true} \mathrel{\&} y < x \twoheadrightarrow \mathsf{true}$ is a goal clause of $\mathcal{R}_1$.

The *narrowing* relation [27, 14] mainly extends rewriting by replacing matching with unification. This paper follows the formalization in [22], while we use the rule $(x \twoheadrightarrow x) \to \top$ instead of the corresponding inference rule. Let $\mathcal{R}$ be a CTRS. A goal clause $S = U \mathrel{\&} s \twoheadrightarrow t \mathrel{\&} S'$ 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 $T = U \mathrel{\&} \sigma C \mathrel{\&} \sigma((s \twoheadrightarrow t)[r]_p) \mathrel{\&} \sigma S'$. 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 $S \overset{i}{\rightsquigarrow}_{\sigma|_{\mathcal{V}ar(S)},\mathcal{R}} T$ to make the substitution explicit. An *innermost narrowing derivation* $T_0 \overset{i}{\rightsquigarrow}{}^*_{\sigma,\mathcal{R}} T_n$ (and $T_0 \overset{i}{\rightsquigarrow}{}^n_{\sigma,\mathcal{R}} T_n$) denotes a sequence of narrowing steps $T_0 \overset{i}{\rightsquigarrow}_{\sigma_1,\mathcal{R}} \cdots \overset{i}{\rightsquigarrow}_{\sigma_n,\mathcal{R}} T_n$ with $\sigma = (\sigma_n \cdots \sigma_1)|_{\mathcal{V}ar(T_0)}$ an idempotent substitution. When we consider two (or more) narrowing derivations $S_1 \overset{i}{\rightsquigarrow}{}^*_{\sigma_1,\mathcal{R}} T_1$ and $S_2 \overset{i}{\rightsquigarrow}{}^*_{\sigma_2,\mathcal{R}} T_2$, we assume that $\mathcal{VR}an(\sigma_1) \cap \mathcal{VR}an(\sigma_2) = \emptyset$.

Innermost narrowing is a counterpart of *constructor-based rewriting* (cf. [22]). Following [22], we define *constructor-based conditional rewriting* on goal clauses as follows: for a goal clause $S = U \mathrel{\&} s \twoheadrightarrow t \mathrel{\&} S'$ 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 $T = U \mathrel{\&} \sigma C \mathrel{\&} (s \twoheadrightarrow t)[\sigma r]_p \mathrel{\&} S'$.

Theorem 2.2 ([21]) *Let $\mathcal{R}$ be a constructor SDCTRS, T a goal clause, and $U \in \mathcal{T}(\{\top,\&\})$.*

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

Example 2.3 Consider $\mathcal{R}_1$ in Section 1 again. The following is an instance of innermost conditional narrowing of $\mathcal{R}_1$:

$$
\begin{array}{l}
(\mathsf{gcd}(\mathsf{s}^4(0),y) \twoheadrightarrow z) \mathrel{\&} (\mathsf{s}(0) < z \twoheadrightarrow \mathsf{true}) \\
\quad \overset{i}{\rightsquigarrow}_{\{y\mapsto \mathsf{s}(y_1)\},\mathcal{R}_1} (y_1 < \mathsf{s}^3(0) \twoheadrightarrow \mathsf{true}) \mathrel{\&} (\mathsf{gcd}(\mathsf{s}^3(0)-y_1,\mathsf{s}(y_1)) \twoheadrightarrow z) \mathrel{\&} (\mathsf{s}(0) < z \twoheadrightarrow \mathsf{true}) \\
\quad \overset{i}{\rightsquigarrow}{}^2_{\{y_1\mapsto \mathsf{s}(0)\},\mathcal{R}_1} (\mathsf{true} \twoheadrightarrow \mathsf{true}) \mathrel{\&} (\mathsf{gcd}(\mathsf{s}^3(0)-\mathsf{s}(0),\mathsf{s}^2(0)) \twoheadrightarrow z) \mathrel{\&} (\mathsf{s}(0) < z \twoheadrightarrow \mathsf{true}) \\
\quad \overset{i}{\rightsquigarrow}_{id,\mathcal{R}_1} \top \mathrel{\&} (\mathsf{gcd}(\mathsf{s}^3(0)-\mathsf{s}(0),\mathsf{s}^2(0)) \twoheadrightarrow z) \mathrel{\&} (\mathsf{s}(0) < z \twoheadrightarrow \mathsf{true}) \\
\quad \overset{i}{\rightsquigarrow}{}^2_{id,\mathcal{R}_1} \top \mathrel{\&} (\mathsf{gcd}(\mathsf{s}^2(0),\mathsf{s}^2(0)) \twoheadrightarrow z) \mathrel{\&} (\mathsf{s}(0) < z \twoheadrightarrow \mathsf{true}) \\
\quad \overset{i}{\rightsquigarrow}_{id,\mathcal{R}_1} \top \mathrel{\&} (\mathsf{s}^2(0) \twoheadrightarrow z) \mathrel{\&} (\mathsf{s}(0) < z \twoheadrightarrow \mathsf{true}) \\
\quad \overset{i}{\rightsquigarrow}_{\{z\mapsto \mathsf{s}^2(0)\},\mathcal{R}_1} \top \mathrel{\&} \top \mathrel{\&} (\mathsf{s}(0) < \mathsf{s}^2(0) \twoheadrightarrow \mathsf{true}) \\
\quad \overset{i}{\rightsquigarrow}{}^2_{id,\mathcal{R}_1} \top \mathrel{\&} \top \mathrel{\&} (\mathsf{true} \twoheadrightarrow \mathsf{true}) \\
\quad \overset{i}{\rightsquigarrow}_{id,\mathcal{R}_1} \top \mathrel{\&} \top \mathrel{\&} \top
\end{array}
$$

The following constructor-based rewriting derivation corresponds to the above narrowing derivation:

$$\begin{array}{l}
(\mathsf{gcd}(\mathsf{s}^4(0),\mathsf{s}^2(0)) \twoheadrightarrow \mathsf{s}^2(0))\ \&\ (\mathsf{s}(0) < \mathsf{s}^2(0) \twoheadrightarrow \mathsf{true}) \\
\quad \xrightarrow{c}_{\mathcal{R}_1} (\mathsf{s}(0) < \mathsf{s}^3(0) \twoheadrightarrow \mathsf{true})\ \&\ (\mathsf{gcd}(\mathsf{s}^3(0) - \mathsf{s}(0),\mathsf{s}^2(0)) \twoheadrightarrow \mathsf{s}^2(0))\ \&\ (\mathsf{s}(0) < \mathsf{s}^2(0) \twoheadrightarrow \mathsf{true}) \\
\quad \xrightarrow{c}_{\mathcal{R}_1} (\mathsf{true} \twoheadrightarrow \mathsf{true})\ \&\ (\mathsf{gcd}(\mathsf{s}^3(0) - \mathsf{s}(0),\mathsf{s}^2(0)) \twoheadrightarrow \mathsf{s}^2(0))\ \&\ (\mathsf{s}(0) < \mathsf{s}^2(0) \twoheadrightarrow \mathsf{true}) \\
\quad \xrightarrow{c}_{\mathcal{R}_1} \top\ \&\ (\mathsf{gcd}(\mathsf{s}^3(0) - \mathsf{s}(0),\mathsf{s}^2(0)) \twoheadrightarrow \mathsf{s}^2(0))\ \&\ (\mathsf{s}(0) < \mathsf{s}^2(0) \twoheadrightarrow \mathsf{true}) \\
\quad \xrightarrow{c}{}^2_{\mathcal{R}_1} \top\ \&\ (\mathsf{gcd}(\mathsf{s}^2(0),\mathsf{s}^2(0)) \twoheadrightarrow \mathsf{s}^2(0))\ \&\ (\mathsf{s}(0) < \mathsf{s}^2(0) \twoheadrightarrow \mathsf{true}) \\
\quad \xrightarrow{c}_{\mathcal{R}_1} \top\ \&\ (\mathsf{s}^2(0) \twoheadrightarrow \mathsf{s}^2(0))\ \&\ (\mathsf{s}(0) < \mathsf{s}^2(0) \twoheadrightarrow \mathsf{true}) \\
\quad \xrightarrow{c}_{\mathcal{R}_1} \top\ \&\ \top\ \&\ (\mathsf{s}(0) < \mathsf{s}^2(0) \twoheadrightarrow \mathsf{true}) \\
\quad \xrightarrow{c}{}^2_{\mathcal{R}_1} \top\ \&\ \top\ \&\ (\mathsf{true} \twoheadrightarrow \mathsf{true}) \\
\quad \xrightarrow{c}_{\mathcal{R}_1} \top\ \&\ \top\ \&\ \top
\end{array}$$

2.4 Regular Tree Grammars

A *regular tree grammar* (an RTG, for short) 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 $\beta \in \mathcal{T}(\mathcal{F} \cup \mathcal{N})$. 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 is not so relevant in this paper. A *regular tree language* is a language generated by an RTG 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 2.4 The RTG $\mathcal{G}_2 = (X, \{X, X'\}, \{0/0, \mathsf{s}/1\}, \{X \to 0,\ X \to \mathsf{s}(X'),\ X' \to \mathsf{s}(X)\})$ generates the sets of even and odd numbers over 0 and s from X and X', respectively: $L(\mathcal{G}_2, X) = L(\mathcal{G}_2) = \{\mathsf{s}^{2n}(0) \mid n \geq 0\}$ and $L(\mathcal{G}_2, X') = \{\mathsf{s}^{2n+1}(0) \mid n \geq 0\}$.

3 Coding of Tuples of Ground Terms

In this section, we introduce the notion of *coding* of tuples of ground terms [2, Section 3.2.1]. To simplify discussions, we consider pairs of terms.

Let $\mathcal{F}$ be a signature. We prepare the signature $\mathcal{F}' = (\mathcal{F} \cup \{\bot\})^2$, where $\bot$ is a new constant. For symbols $\mathsf{f}_1, \mathsf{f}_2 \in \mathcal{F}$, we denote the function symbol $(\mathsf{f}_1, \mathsf{f}_2) \in \mathcal{F}'$ by $\mathsf{f}_1\mathsf{f}_2$, and the arity of $\mathsf{f}_1\mathsf{f}_2$ is $\max(\mathit{arity}(\mathsf{f}_1), \mathit{arity}(\mathsf{f}_2))$. The coding of pairs of ground terms, $[\cdot,\ \cdot]$, is recursively defined as follows:

- $[\,\mathsf{f}(s_1,\ldots,s_m),\ \mathsf{g}(t_1,\ldots,t_n)\,] = \mathsf{fg}([\,s_1,\ t_1\,],\ldots,[\,s_m,\ t_m\,],[\,\bot,\ t_{m+1}\,],\ldots,[\,\bot,\ t_n\,])$ if $m \leq n$,
- $[\,\mathsf{f}(s_1,\ldots,s_m),\ \mathsf{g}(t_1,\ldots,t_n)\,] = \mathsf{fg}([\,s_1,\ t_1\,],\ldots,[\,s_n,\ t_n\,],[\,s_{n+1},\ \bot\,],\ldots,[\,s_m,\ \bot\,])$ if $m > n$,
- $[\,\mathsf{f}(s_1,\ldots,s_m),\ \bot\,] = \mathsf{f}\bot([\,s_1,\ \bot\,],\ldots,[\,s_m,\ \bot\,])$, and
- $[\,\bot,\ \mathsf{g}(t_1,\ldots,t_n)\,] = \bot\mathsf{g}([\,\bot,\ t_1\,],\ldots,[\,\bot,\ t_n\,])$.

Note that $\mathcal{P}os([\,t_1,\ t_2\,]) = \mathcal{P}os(t_1) \cup \mathcal{P}os(t_2)$. Note also that for $i = 1,2$ and for $p \in \mathcal{P}os([\,t_1,\ t_2\,])$, if $p \notin \mathcal{P}os(t_i)$, then $\bot$ is complemented for t_i. As described in [2, Section 3.2.1], the basic idea of coding is to stack function symbols as illustrated in Figure 1.

Example 3.1 As in Figure 1, $[\,\mathsf{f}(\mathsf{g}(\mathsf{a}),\mathsf{g}(\mathsf{a})),\ \mathsf{f}(\mathsf{f}(\mathsf{a},\mathsf{a}),\mathsf{a})\,] = \mathsf{ff}(\mathsf{gf}(\mathsf{aa},\bot\mathsf{a}),\mathsf{ga}(\mathsf{a}\bot))$.

4 Grammar Representations for Sets of Idempotent Substitutions

In this section, we briefly introduce grammar representations that define sets of idempotent substitutions. We follow the formalization in [21], which is based on *success set equations* in [23]. Since substitutions derived by narrowing steps are assumed to be idempotent, we deal with only idempotent substitutions which introduce only *fresh* variables not appearing in any previous term.

In the following, a renaming ξ is used to (partially) rename a particular term t w.r.t. a set X of variables with $X \subseteq \mathcal{V}ar(t) \cap \mathcal{D}om(\xi)$ by assuming that $\xi|_X$ is injective on X (i.e., for all variables $x, y \in X$, if $x \neq y$ then $\xi x \neq \xi y$) and $\mathcal{VR}an(\xi|_X) \cap (\mathcal{V}ar(t) \setminus X) = \emptyset$. For this reason, we write $\xi|_X$ instead of ξ, and call $\xi|_X$ a *renaming for t* (simply, a renaming).

We first introduce terms to represent idempotent substitutions computed using composition operators $\cdot$ and $\Uparrow$. We prepare the signature Σ consisting of the following symbols [21]:

- a finite number of 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 brackets with the precedence such that $\bullet$ has a higher priority than &.

We deal with terms over Σ and some constants used for 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 ($\Sigma \cap \mathcal{N} = \emptyset$), 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 to generate substitutions (more precisely, terms in $\mathcal{T}(\Sigma)$) from Γ_t, e.g., recursively, and the role of δ is to connect such substitutions with other substitutions if necessary, where the application of δ to some term results in t. For this reason, we restrict the second argument of REC to renamings, and for each term $\textsc{rec}(\Gamma_t, \delta)$, we require δ to be an idempotent renaming (i.e., $\mathcal{D}om(\delta) \cap \mathcal{VR}an(\delta) = \emptyset$ and δ is injective on $\mathcal{D}om(\delta)$) such that $\mathcal{VR}an(\delta) \subseteq \mathcal{V}ar(t)$, and $\mathcal{D}om(\delta) \cap (\mathcal{V}ar(t) \setminus \mathcal{VR}an(\delta)) = \emptyset$.

Example 4.1 ([21]) The following are terms in $\mathcal{T}(\Sigma)$:

- $\{y \mapsto \mathsf{0}\} \bullet \{x \mapsto \mathsf{s}(y)\}$,
- $(\{x' \mapsto \mathsf{s}(y)\} \bullet \{x \mapsto x'\})$ & $\{x \mapsto \mathsf{s}(\mathsf{s}(z))\}$,
- $(\varnothing$ & $\{y \mapsto z\}) \bullet \{x \mapsto \mathsf{s}(y)\}$, and
- $\textsc{rec}(\{x \mapsto \mathsf{0},\ y \mapsto \mathsf{s}(y')\}, \{x' \mapsto x,\ y' \mapsto y\}) \bullet \{y \mapsto \mathsf{s}(x')\}$.

Note that substitutions $\{y \mapsto \mathsf{0}\}$, $\{x \mapsto \mathsf{s}(y)\}$, $\{x' \mapsto \mathsf{s}(y)\}$, $\{x \mapsto x'\}$, $\{x \mapsto \mathsf{s}(\mathsf{s}(z))\}$, $\{y \mapsto z\}$, $\{x \mapsto \mathsf{0},\ y \mapsto \mathsf{s}(y')\}$, $\{x' \mapsto x,\ y' \mapsto y\}$, $\{y \mapsto \mathsf{s}(x')\}$ are considered constants.

Next, we recall *parallel composition* $\Uparrow$ of idempotent substitutions [13, 26], 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 4.2 (parallel composition $\Uparrow$ [26]) *Let θ_1 and θ_2 be idempotent substitutions. Then, we define $\Uparrow$ as follows: $\theta_1 \Uparrow \theta_2 = mgu(\widehat{\theta_1} \cup \widehat{\theta_2})$ if $\widehat{\theta_1}$ & $\widehat{\theta_2}$ is unifiable, and otherwise, $\theta_1 \Uparrow \theta_2 = fail$. Note that we define $\theta_1 \Uparrow \theta_2 = fail$ 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: $\Theta_1 \Uparrow \Theta_2 = \{\theta_1 \Uparrow \theta_2 \mid \theta_1 \in \Theta_1,\ \theta_2 \in \Theta_2,\ \theta_1 \Uparrow \theta_2 \neq fail\}$.*

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

Example 4.3 ([21]) 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 because y appears in the range. On the other hand, $\{x \mapsto \mathsf{s}(z),\ y \mapsto z\} \Uparrow \{x \mapsto y\} = fail$.

A key of construction of narrowing trees (and their grammar representations) is *compositionality* of innermost narrowing (cf. [21]): S_1 & $S_2 \overset{i}{\leadsto}^*_{\sigma,\mathcal{R}} T$ if and only if $S_1 \overset{i}{\leadsto}^*_{\sigma_1,\mathcal{R}} T_1$, $S_2 \overset{i}{\leadsto}^*_{\sigma_2,\mathcal{R}} T_2$, $T = T_1$ & T_2, and $\sigma = \sigma_1 \Uparrow \sigma_2$. To compute a substitution derived by innermost narrowing from a goal clause S_1 & S_2, we compute substitutions σ_1 and σ_2 derived by innermost narrowing from S_1 and S_2, respectively, and then compute $\sigma_1 \Uparrow \sigma_2$. When we compute $\sigma_1 \Uparrow \sigma_2$ from two narrowing derivations $S_1 \overset{i}{\leadsto}^*_{\sigma_1,\mathcal{R}} T_1$ and $S_2 \overset{i}{\leadsto}^*_{\sigma_2,\mathcal{R}} T_2$, 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 $fresh_\delta(\cdot)$ 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 $fresh_\delta(\sigma)$ by $(\xi \cdot \sigma)|_{\mathcal{D}om(\sigma)}$ where ξ is a renaming such that $\mathcal{D}om(\xi) = \mathcal{VR}an(\sigma)$ and $\mathcal{VR}an(\xi) \cap (\mathcal{D}om(\delta) \cup \mathcal{VR}an(\delta) \cup \mathcal{D}om(\sigma)) = \emptyset$.[2] The subscript δ of $fresh_\delta(\cdot)$ is used to specify freshness of variables—we say that a variable x is *fresh* w.r.t. a set X of variables if $x \notin X$.

A term e in $\mathcal{T}(\Sigma)$ defines a substitution. The semantics of terms in $\mathcal{T}(\Sigma)$ is inductively defined as follows [21]:

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

Notice that Γ_t, a non-terminal used in an RTG, is not included in $\mathcal{T}(\Sigma)$, and thus, $[\![\Gamma_t]\!]$ is not defined. Since $\Uparrow$ may fail, we allow to have *fail*, e.g., $[\![\{y \mapsto \mathsf{s}(x)\} \bullet \{x \mapsto y\} \,\&\, \{z \mapsto 0\}]\!] = fail$. The number of variables appearing in an RTG defined below is finite. However, we would like to use RTGs 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 $[\![\mathrm{REC}(e, \delta)]\!]$, we introduced the operation $fresh_\delta(\cdot)$ that makes all variables introduced by $[\![e]\!]$ *fresh w.r.t.* $\mathcal{D}om(\delta) \cup \mathcal{VR}an(\delta)$. In [23], this operation is implicitly considered, but in [21], REC is explicitly introduced to the syntax in order to convert 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 $fresh_{\theta_1}(\cdot)$ in the case of $[\![e_1 \,\&\, e_2]\!]$.

[2] For $\mathcal{VR}an(\xi)$, we choose variables not appearing in any substitutions in Σ.

The semantics of terms in $\mathcal{T}(\Sigma)$ is naturally extended to subsets of $\mathcal{T}(\Sigma)$ as follows: for a set $L \subseteq \mathcal{T}(\Sigma)$, $[\![L]\!] = \{[\![e]\!] \mid e \in L,\ [\![e]\!] \neq \mathit{fail}\}$.

Example 4.4 ([21]) The expressions in Example 4.1 are interpreted as follows:

- $[\![\, \{y \mapsto \mathsf{0}\} \bullet \{x \mapsto \mathsf{s}(y)\} \,]\!] = \{y \mapsto \mathsf{0}\} \cdot \{x \mapsto \mathsf{s}(y)\} = \{x \mapsto \mathsf{s}(\mathsf{0}),\ y \mapsto \mathsf{0}\}$,
- $[\![\, (\{x' \mapsto \mathsf{s}(y)\} \bullet \{x \mapsto x'\}) \mathbin{\&} \{x \mapsto \mathsf{s}(\mathsf{s}(z))\} \,]\!]$
  $= \Big(\{x \mapsto \mathsf{s}(y),\ x' \mapsto \mathsf{s}(y)\} \Uparrow \mathit{fresh}_{\{x \mapsto \mathsf{s}(y),\ x' \mapsto \mathsf{s}(y)\}}(\{x \mapsto \mathsf{s}(\mathsf{s}(z'))\})\Big)|_{\{x,x'\}}$
  $= (\{x \mapsto \mathsf{s}(y),\ x' \mapsto \mathsf{s}(y)\} \Uparrow \{x \mapsto \mathsf{s}(\mathsf{s}(z'))\})|_{\{x,x'\}}$
  $= (\{x \mapsto \mathsf{s}(\mathsf{s}(z')),\ x' \mapsto \mathsf{s}(\mathsf{s}(z'))\})|_{\{x,x'\}}$ [3]
  $= \{x \mapsto \mathsf{s}(\mathsf{s}(z')),\ x' \mapsto \mathsf{s}(\mathsf{s}(z'))\}$,
- $[\![\, (\varnothing \mathbin{\&} \{y \mapsto z\}) \bullet \{x \mapsto \mathsf{s}(y)\} \,]\!] = \mathit{fail}$ (since $[\![\varnothing]\!] = \mathit{fail}$ and then $[\![\, \varnothing \mathbin{\&} \{y \mapsto z\} \,]\!] = \mathit{fail}$), and
- $[\![\, \mathrm{REC}(\{x \mapsto \mathsf{0},\ y \mapsto \mathsf{s}(y')\}, \{x' \mapsto x,\ y' \mapsto y\}) \bullet \{y \mapsto \mathsf{s}(x')\} \,]\!]$
  $= \Big(\mathit{fresh}_{\{x' \mapsto x,\ y' \mapsto y\}}(\{x \mapsto \mathsf{0},\ y \mapsto \mathsf{s}(y')\}) \cdot \{x' \mapsto x,\ y' \mapsto y\}\Big)|_{\{x',y'\}} \cdot \{y \mapsto \mathsf{s}(x')\}$
  $= (\{x \mapsto \mathsf{0},\ y \mapsto \mathsf{s}(y'')\} \cdot \{x' \mapsto x,\ y' \mapsto y\})|_{\{x',y'\}} \cdot \{y \mapsto \mathsf{s}(x')\}$
  $= \{x' \mapsto \mathsf{0},\ y' \mapsto \mathsf{s}(y''),\ y \mapsto \mathsf{s}(\mathsf{0})\}$.

To define sets of idempotent substitutions, we adopt RTGs. In the following, we drop the third component from grammars constructed below because the third one is fixed to Σ with a finite number of substitutions that are clear from production rules. A *substitution-set grammar* (SSG) for a term t_0 is an RTG $\mathcal{G} = (\Gamma_{t_0}, \mathcal{N}, \mathcal{P})$ 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 $\Gamma_t \in \mathcal{N}$, and the numbers of variables appearing in $L(\mathcal{G}, \Gamma_t)$ is finite. The set of substitutions generated by $\mathcal{G}$ from $\Gamma_t \in \mathcal{N}$ is $[\![L(\mathcal{G}, \Gamma_t)]\!]$, i.e., $[\![L(\mathcal{G}, \Gamma_t)]\!] = \{[\![e]\!] \mid e \in L(\mathcal{G}, \Gamma_t),\ [\![e]\!] \neq \mathit{fail}\}$. Note that the number of variables in $\bigcup_{\theta \in [\![L(\mathcal{G}, \Gamma_t)]\!]} \mathcal{VR}an(\theta)$ may be infinite because of the interpretation for REC.

Example 4.5 The RTG $\mathcal{G}_1$ in Section 1 is an SSG for a term $\Gamma_{x<y \twoheadrightarrow \mathsf{true} \& y<x \twoheadrightarrow \mathsf{true}}$. We have that

$$L(\mathcal{G}_1, \Gamma_{x<y \twoheadrightarrow \mathsf{true}}) = \left\{ \begin{array}{l} \{x \mapsto \mathsf{0},\ y \mapsto \mathsf{s}(y_2)\}, \\ \mathrm{REC}(\{x \mapsto \mathsf{0},\ y \mapsto \mathsf{s}(y_2)\}, \{x_3 \mapsto x,\ y_3 \mapsto y\}) \bullet \{x \mapsto \mathsf{s}(x_3),\ y \mapsto \mathsf{s}(y_3)\}, \\ \mathrm{REC}\left(\begin{array}{c} \mathrm{REC}(\{x \mapsto \mathsf{0},\ y \mapsto \mathsf{s}(y_2)\}, \{x_3 \mapsto x,\ y_3 \mapsto y\}) \\ \bullet \\ \{x \mapsto \mathsf{s}(x_3),\ y \mapsto \mathsf{s}(y_3)\} \end{array}, \{x_3 \mapsto x,\ y_3 \mapsto y\} \right) \bullet \{x \mapsto \mathsf{s}(x_3),\ y \mapsto \mathsf{s}(y_3)\}, \\ \ldots \end{array} \right\}$$

and $[\![L(\mathcal{G}_1, \Gamma_{x<y \twoheadrightarrow \mathsf{true}})]\!] = \{\{x \mapsto \mathsf{s}^m(\mathsf{0}),\ y \mapsto \mathsf{s}^n(a)\} \mid 0 \leq m < n,\ a \in \{\mathsf{0}, \mathsf{true}, \mathsf{false}\}\}$.

5 Transforming SSGs into RTGs Generating Ranges of Substitutions

In this section, given a goal clause T and two variables x_1, x_2 appearing in T, we show a transformation of an SSG $\mathcal{G} = (\Gamma_{T_0}, \mathcal{N}, \mathcal{P})$ into an RTG $\mathcal{G}'$ such that $L(\mathcal{G}', \Gamma_T^{(x_1,x_2)}) \supseteq \{[\, \xi\theta x_1,\ \xi\theta x_2 \,] \mid \theta \in L(\mathcal{G}, \Gamma_T),\ \xi \in \mathit{Subst}(\mathcal{C}),\ \mathcal{V}ar(\theta x_1, \theta x_2) \subseteq \mathcal{D}om(\xi)\}$, where $\mathcal{C}$ is a set of constructors we deal with. Note that T does not

[3] Note that $\{x \mapsto \mathsf{s}(y),\ x' \mapsto \mathsf{s}(y)\} \Uparrow \{x \mapsto \mathsf{s}(\mathsf{s}(z'))\} = \{x \mapsto \mathsf{s}(\mathsf{s}(z')),\ x' \mapsto \mathsf{s}(\mathsf{s}(z')),\ y \mapsto \mathsf{s}(z')\}$.

have to be T_0. The transformation is an extension of the transformation in [21, Section 7] and applicable to SSGs satisfying a certain syntactic condition shown later. In the following, we aim at showing that $L(\mathcal{G}_1, \Gamma_{x<y\twoheadrightarrow\mathsf{true}}) \cap L(\mathcal{G}_1, \Gamma_{y<x\twoheadrightarrow\mathsf{true}}) = \emptyset$. We use $\mathcal{C}$ as a set of constructors unless noted otherwise.

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 following form:

$$\Gamma_{T'} \to \theta_1 \mid \cdots \mid \theta_m \mid \mathrm{REC}(\Gamma_{T_1}, \delta_1) \bullet \theta_{m+1} \mid \cdots \mid \mathrm{REC}(\Gamma_{T_n}, \delta_n) \bullet \theta_{m+n}$$

where $\mathcal{VR}an(\delta_j) = \mathcal{V}ar(T_j)$[4] for all $1 \leq j \leq n$, and $\theta_1, \ldots, \theta_{m+n}$ are idempotent substitutions such that $\mathcal{D}om(\theta_j) = \mathcal{V}ar(T')$ for all $1 \leq j \leq m+n$. Note that $\Gamma_{T'} \to \mathrm{REC}(\Gamma_{T''}, \delta)$ is considered $\Gamma_{T'} \to \mathrm{REC}(\Gamma_{T''}, \delta) \bullet id$. In addition, for each $\Gamma_{T'} \to \mathrm{REC}(\Gamma_{T_i}, \delta_i) \bullet \theta_{m+i}$ with $1 \leq i \leq n$, we assume that for all variables x, y in T' and for each position $p \in \mathcal{P}os(\delta\theta_{m+i}x) \cap \mathcal{P}os(\delta\theta_{m+i}y)$, all of the following hold:

- if $(\delta\theta_{m+i}x)|_p \in \mathcal{V}ar(T_i)$, then $(\delta\theta_{m+i}y)|_p \in \mathcal{V}ar(T_i) \cup \mathcal{T}(\mathcal{C}, \mathcal{V} \setminus \mathcal{V}ar(T_i))$, and
- if $(\delta\theta_{m+i}y)|_p \in \mathcal{V}ar(T_i)$, then $(\delta\theta_{m+i}x)|_p \in \mathcal{V}ar(T_i) \cup \mathcal{T}(\mathcal{C}, \mathcal{V} \setminus \mathcal{V}ar(T_i))$.

This assumption implies that for such x, y, and p, the terms $(\delta\theta_{m+i}x)|_p$ and $(\delta\theta_{m+i}y)|_p$ satisfy one of the following:

(a) both are rooted by function symbols,

(b) both are variables in $\mathcal{V}ar(T_i)$,

(c) one is a variable in $\mathcal{V}ar(T_i)$ and the other is a term in $\mathcal{T}(\mathcal{C}, \mathcal{V} \setminus \mathcal{V}ar(T_i))$, or

(d) both are terms in $\mathcal{T}(\mathcal{C}, \mathcal{V} \setminus \mathcal{V}ar(T_i))$.

For example, both $\mathcal{P}_1|_{\Gamma_{x<y\twoheadrightarrow\mathsf{true}}}$ and $\mathcal{P}_1|_{\Gamma_{y<\twoheadrightarrow\mathsf{true}}}$ satisfy the above assumption.

Our idea of extending the previous transformation is the use of coding; Roughly speaking, for $\Gamma_{T'} \to \mathrm{REC}(\Gamma_{T_i}, \delta_i) \bullet \theta_{m+i}$ with $1 \leq i \leq n$ and for all variables x, y in T', we apply *coding* to $\delta\theta_{m+i}x$ and $\delta\theta_{m+i}y$. A variable in $\mathcal{V}ar(T_i)$, which is instantiated by substitutions generated from Γ_{T_i}, may prevent us from constructing a finite number of production rules (see Example 5.3 below). For this reason, we expect any variable[5] in $\mathcal{V}ar(\delta\theta_{m+i}x, \delta\theta_{m+i}y) \cap \mathcal{V}ar(T_i)$ to be coded with

- $\perp$ (the case where the precondition "$p \in \mathcal{P}os(\delta\theta_{m+i}x) \cap \mathcal{P}os(\delta\theta_{m+1}y)$" does not hold),
- another variable in $\mathcal{V}ar(\delta\theta_{m+i}x, \delta\theta_{m+i}y) \cap \mathcal{V}ar(T_i)$ (the case where (b) above holds), or
- a constructor term without any variable in $\mathcal{V}ar(T_i)$ (the case where (c) above holds).

Definition 5.1 *We denote the set of constructor terms appearing in substitutions in $\mathcal{P}$ by Patterns$(\mathcal{P})$, where such constructor terms are instantiated with a non-terminal A introduced during the transformation below: Patterns$(\mathcal{P}) = \{\{x \mapsto A \mid x \in \mathcal{V}ar(t)\}(t) \mid \theta$ appears in $\mathcal{P}$, $s \in \mathcal{VR}an(\theta)$, $t \trianglelefteq s\}$.*[6] *We denote the set of variables appearing in $\mathcal{N}$ by Vars$(\mathcal{N})$: Vars$(\mathcal{N}) = \bigcup_{\Gamma_{T'} \in \mathcal{N}} \mathcal{V}ar(T')$. The RTG obtained from $\mathcal{G}$ and variables x_1, x_2 in T, denoted by $\mathcal{R}an(\mathcal{G}, T, x_1, x_2)$, is $(\Gamma_T^{(x_1,x_2)}, \mathcal{N}' \cup \mathcal{N}_A, \mathcal{P}'_1 \cup \mathcal{P}'_2 \cup \mathcal{P}_{AA} \cup \mathcal{P}_{A\perp} \cup \mathcal{P}_{\perp A})$ such that*

[4] In defining SSGs, we only required that $\mathcal{VR}an(\delta_j) \subseteq \mathcal{V}ar(T_j)$, but to make the transformation below precise, we require that $\mathcal{VR}an(\delta_j) = \mathcal{V}ar(T_j)$. This requirement is not restrictive because SSGs for narrowing trees satisfy this requirement because δ_j connects T_j with a renamed variant which has no shared variable with T_j.

[5] This is not the case where either (a) or (d) holds.

[6] The current definition of *Patterns*$(\mathcal{P})$ is not well optimized and *Patterns*$(\mathcal{P})$ may include some terms that are not necessary for the transformation. However, for readability, we adopt this simpler definition.

- $\mathcal{N}' = \{\ \Gamma_{T'}^{(x,y)},\ \Gamma_{T'}^{(x,t)},\ \Gamma_{T'}^{(t,y)} \mid x,y \in \mathit{Vars}(\mathcal{N}),\ \Gamma_{T'} \in \mathcal{N},\ t \in \mathit{Patterns}(\mathcal{P}) \cup \{\bot\}\ \}$,
- $\mathcal{N}_A = \{\ AA,\ A\bot,\ \bot A\ \}$,
- $\mathcal{P}'_1 = \{\ \Gamma_{T'}^{(t_1,t_2)} \to u \mid \Gamma_{T'} \to \theta \in \mathcal{P},\ \Gamma_{T'}^{(t_1,t_2)} \in \mathcal{N}',\ \xi_A = \{x \mapsto A \mid x \in \mathcal{V}ar(\theta t_1, \theta t_2)\},\ u \in \langle \xi_A \theta t_1,\ \xi_A \theta t_2 \rangle_\top\ \}$,
- $\mathcal{P}'_2 = \{\ \Gamma_{T'}^{(t_1,t_2)} \to u \mid \Gamma_{T'} \to \mathrm{REC}(\Gamma_{T''}, \delta) \bullet \theta \in \mathcal{P},\ \Gamma_{T'}^{(t_1,t_2)} \in \mathcal{N}', \xi_A = \{x \mapsto A \mid x \in \mathcal{V}ar(\delta\theta t_1, \delta\theta t_2) \setminus \mathcal{V}ar(T'')\},\ u \in \langle \xi_A \delta\theta t_1,\ \xi_A \delta\theta t_2 \rangle_{T''}\ \}$,
- $\mathcal{P}_{AA} = \{\ AA \to u \mid \mathsf{f}/m, \mathsf{g}/n \in \mathcal{C},\ u \in \langle \mathsf{f}(A,\ldots,A),\ \mathsf{g}(A,\ldots,A) \rangle_\top\ \}$,
- $\mathcal{P}_{A\bot} = \{\ A\bot \to u \mid \mathsf{f}/m \in \mathcal{C},\ u \in \langle \mathsf{f}(A,\ldots,A),\ \bot \rangle_\top\ \}$, *and*
- $\mathcal{P}_{\bot A} = \{\ \bot A \to u \mid \mathsf{g}/n \in \mathcal{C},\ u \in \langle \bot,\ \mathsf{g}(A,\ldots,A) \rangle_\top\ \}$,

where $\langle \cdot,\ \cdot \rangle_{T'}$*, which takes a goal clause* T' *and two terms in* $\mathcal{T}(\mathcal{F} \cup \{A\}, \mathcal{V}ar(T'))$ *as input and returns a set of terms in* $\mathcal{T}(\mathcal{F} \cup \mathcal{N}' \cup \mathcal{N}_A)$*, is recursively defined as follows:*

- $\langle x,\ y \rangle_{T'} = \{\ \Gamma_{T'}^{(x,y)}\ \}$, *where* $x,y \in \mathcal{V}$,
- $\langle x,\ t \rangle_{T'} = \{\ \Gamma_{T'}^{(x,t)}\ \}$, *where* $x \in \mathcal{V}$ *and* $t \in \mathit{Patterns}(\mathcal{P})$,
- $\langle x,\ \bot \rangle_{T'} = \{\ \Gamma_{T'}^{(x,\bot)}\ \}$, *where* $x \in \mathcal{V}$,
- $\langle t,\ y \rangle_{T'} = \{\ \Gamma_{T'}^{(A,y)}\ \}$, *where* $y \in \mathcal{V}$ *and* $t \in \mathit{Patterns}(\mathcal{P})$,
- $\langle \bot,\ y \rangle_{T'} = \{\ \Gamma_{T'}^{(\bot,y)}\ \}$, *where* $y \in \mathcal{V}$,
- $\langle A,\ A \rangle_{T'} = \{\ AA\ \}$,
- $\langle A,\ \bot \rangle_{T'} = \{\ A\bot\ \}$,
- $\langle \bot,\ A \rangle_{T'} = \{\ \bot A\ \}$,
- $\langle \bot,\ \mathsf{g}(t_1,\ldots,t_n) \rangle_{T'} = \{\ \bot\mathsf{g}(u_1,\ldots,u_n) \mid 1 \le i \le n,\ u_i \in \langle \bot,\ t_i \rangle_{T'}\ \}$,
- $\langle \mathsf{f}(s_1,\ldots,s_m),\ \bot \rangle_{T'} = \{\ \mathsf{f}\bot(u_1,\ldots,u_m) \mid 1 \le i \le m,\ u_i \in \langle s_i,\ \bot \rangle_{T'}\ \}$,
- $\langle A,\ \mathsf{g}(t_1,\ldots,t_n) \rangle_{T'} = \{\ \mathsf{fg}(u_1,\ldots,u_m,u_{m+1},\ldots,u_n) \mid \mathsf{f}/m \in \mathcal{C},\ m < n,\ 1 \le i \le m,\ u_i \in \langle A,\ t_i \rangle_{T'},\ 1 \le j \le n-m,\ u_{m+j} \in \langle \bot,\ t_{m+j} \rangle_{T'}\ \} \cup \{\ \mathsf{fg}(u_1,\ldots,u_n,u_{n+1},\ldots,u_m) \mid \mathsf{f}/m \in \mathcal{C},\ m \ge n,\ 1 \le i \le n,\ u_i \in \langle A,\ t_i \rangle_{T'},\ 1 \le j \le m-n,\ u_{n+j} \in \langle \bot,\ t_{n+j} \rangle_{T'}\ \}$,
- $\langle \mathsf{f}(s_1,\ldots,s_m),\ A \rangle_{T'} = \{\ \mathsf{fg}(u_1,\ldots,u_m,u_{m+1},\ldots,u_n) \mid \mathsf{g}/n \in \mathcal{C},\ m < n,\ 1 \le i \le m,\ u_i \in \langle s_i,\ A \rangle_{T'},\ 1 \le j \le n-m,\ u_{m+j} \in \langle \bot,\ A \rangle_{T'}\ \} \cup \{\ \mathsf{fg}(u_1,\ldots,u_n,u_{n+1},\ldots,u_m) \mid \mathsf{g}/n \in \mathcal{C},\ m \ge n,\ 1 \le i \le n,\ u_i \in \langle s_i,\ A \rangle_{T'},\ 1 \le j \le m-n,\ u_{n+j} \in \langle s_{n+j},\ \bot \rangle_{T'}\ \}$,
- $\langle \mathsf{f}(s_1,\ldots,s_m),\ \mathsf{g}(t_1,\ldots,t_n) \rangle_{T'} = \{\ \mathsf{fg}(u_1,\ldots,u_m,u_{m+1},\ldots,u_n) \mid 1 \le i \le m,\ u_i \in \langle s_i,\ t_i \rangle_{T'},\ 1 \le j \le n-m,\ u_{m+j} \in \langle \bot,\ t_{m+j} \rangle_{T'}\ \}$ *if* $m < n$, *and*
- $\langle \mathsf{f}(s_1,\ldots,s_m),\ \mathsf{g}(t_1,\ldots,t_n) \rangle_{T'} = \{\ \mathsf{fg}(u_1,\ldots,u_n,u_{n+1},\ldots,u_m) \mid 1 \le i \le n,\ u_i \in \langle s_i,\ t_i \rangle_{T'},\ 1 \le j \le m-n,\ u_{n+j} \in \langle s_{n+j},\ \bot \rangle_{T'}\ \}$ *if* $m \ge n$.

Note that the non-terminal AA generates $\{[t_1,\ t_2] \mid t_1,t_2 \in \mathcal{T}(\mathcal{C})\}$, the non-terminal $A\bot$ generates $\{[t_1,\ \bot] \mid t_1 \in \mathcal{T}(\mathcal{C})\}$, and the non-terminal $\bot A$ generates $\{[\bot,\ t_2] \mid t_2 \in \mathcal{T}(\mathcal{C})\}$. Note also that we generate only production rules that are reachable from $\Gamma_T^{(x_1,x_2)}$, and drop from $\mathcal{N}' \cup \mathcal{N}_A$ non-terminals not appearing in the generated production rules.

Example 5.2 Consider $\mathcal{G}_1 = (\Gamma_{x<y\twoheadrightarrow\mathsf{true}\&y<x\twoheadrightarrow\mathsf{true}}, \{\Gamma_{x<y\twoheadrightarrow\mathsf{true}\&y<x\twoheadrightarrow\mathsf{true}}, \Gamma_{x<y\twoheadrightarrow\mathsf{true}}, \Gamma_{y<x\twoheadrightarrow\mathsf{true}}\}, \mathcal{P}_1)$ in Section 1. We have that

- $\mathit{Patterns}(\mathcal{P}_1) = \{\mathsf{0}, \mathsf{s}(A), A\}$, and
- $\mathit{Vars}(\{\Gamma_{x<y\twoheadrightarrow\mathsf{true}\&y<x\twoheadrightarrow\mathsf{true}}, \Gamma_{x<y\twoheadrightarrow\mathsf{true}}, \Gamma_{y<x\twoheadrightarrow\mathsf{true}}\}) = \{x,y\}$.

Let us focus on $\Gamma_{x<y\twoheadrightarrow\mathsf{true}}$ and x,y. Since neither $\Gamma_{x<y\twoheadrightarrow\mathsf{true}\&y<x\twoheadrightarrow\mathsf{true}}$ nor $\Gamma_{y<x\twoheadrightarrow\mathsf{true}}$ is reachable from $\Gamma_{x<y\twoheadrightarrow\mathsf{true}}$ by $\mathcal{P}_1$, when we construct the RTG $\mathcal{R}an(\mathcal{G}_1, \Gamma_{x<y\twoheadrightarrow\mathsf{true}}, x, y)$, we do not take into account $\Gamma_{x<y\twoheadrightarrow\mathsf{true}\&y<x\twoheadrightarrow\mathsf{true}}, \Gamma_{y<x\twoheadrightarrow\mathsf{true}}$, and their rules. The RTG $\mathcal{R}an(\mathcal{G}_1, \Gamma_{x<y\twoheadrightarrow\mathsf{true}}, x, y) = (\Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}}, \mathcal{N}' \cup \mathcal{N}_A, \mathcal{P}'_1 \cup \mathcal{P}'_2 \cup \mathcal{P}_{AA} \cup \mathcal{P}_{A\bot} \cup \mathcal{P}_{\bot A})$ is constructed as follows:

- $\mathcal{N}' = \{\Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}},\ \Gamma^{(y,x)}_{x<y\twoheadrightarrow\mathsf{true}},\ \Gamma^{(x,y)}_{y<x\twoheadrightarrow\mathsf{true}}\}$,
- $\mathcal{N}_A = \{AA,\ A\bot,\ \bot A\}$,
- $\mathcal{P}'_1 = \{\Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}} \to \mathsf{0s}(\bot A)\}$, because $\Gamma_{x<y\twoheadrightarrow\mathsf{true}} \to \{x \mapsto \mathsf{0},\ y \mapsto \mathsf{s}(y_2)\} \in \mathcal{P}_1$ and $\langle \mathsf{0},\ \mathsf{s}(A)\rangle_\top = \{\mathsf{0s}(\bot A)\}$,
- $\mathcal{P}'_2 = \{\Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}} \to \mathsf{ss}(\Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}})\}$, because $\Gamma_{x<y\twoheadrightarrow\mathsf{true}} \to \mathrm{REC}(\Gamma_{x<y\twoheadrightarrow\mathsf{true}}, \{x_3 \mapsto x,\ y_3 \mapsto y\}) \bullet \{x \mapsto \mathsf{s}(x_3),\ y \mapsto \mathsf{s}(y_3)\} \in \mathcal{P}_1$ and $\langle \mathsf{s}(x),\ \mathsf{s}(y)\rangle_{x<y\twoheadrightarrow\mathsf{true}} = \{\mathsf{ss}(\Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}})\}$,
- $\mathcal{P}_{AA} = \{\, AA \to u \mid u \in \{\mathsf{00},\ \mathsf{0s}(\bot A),\ \mathsf{0true},\ \mathsf{0false},\ \mathsf{s0}(A\bot),\ \mathsf{ss}(AA),\ \mathsf{strue}(A\bot),\ \mathsf{sfalse}(A\bot),\ \mathsf{true0},\ \mathsf{trues}(\bot A),\ \mathsf{truetrue},\ \mathsf{truefalse},\ \mathsf{false0},\ \mathsf{falses}(\bot A),\ \mathsf{falsetrue},\ \mathsf{falsefalse}\}\,\}$,
- $\mathcal{P}_{A\bot} = \{\, A\bot \to u \mid u \in \{\mathsf{0}\bot,\ \mathsf{s}\bot(A\bot),\ \mathsf{true}\bot,\ \mathsf{false}\bot\}\,\}$, and
- $\mathcal{P}_{\bot A} = \{\, \bot A \to u \mid u \in \{\bot\mathsf{0},\ \bot\mathsf{s}(\bot A),\ \bot\mathsf{true},\ \bot\mathsf{false}\}\,\}$.

For $\Gamma_{y<x\twoheadrightarrow\mathsf{true}}$ and x,y, we add $\Gamma^{(x,y)}_{y<x\twoheadrightarrow\mathsf{true}} \to \Gamma^{(y,x)}_{x<y\twoheadrightarrow\mathsf{true}}$ to the above production rules. Rules that are not reachable from $\Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}}$ or $\Gamma^{(x,y)}_{y<x\twoheadrightarrow\mathsf{true}}$ can be dropped from $\mathcal{R}an(\mathcal{G}_1, \Gamma_{x<y\twoheadrightarrow\mathsf{true}}, x, y)$, obtaining an RTG, denoted by $\mathcal{G}_4$, with the following production rules:

$$
\begin{array}{ll}
\Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}} \to \mathsf{0s}(\bot A) \mid \mathsf{ss}(\Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}}) & A\bot \to \mathsf{0}\bot \mid \mathsf{s}\bot(A\bot) \mid \mathsf{true}\bot \mid \mathsf{false}\bot \\
\Gamma^{(y,x)}_{x<y\twoheadrightarrow\mathsf{true}} \to \mathsf{s0}(A\bot) \mid \mathsf{ss}(\Gamma^{(y,x)}_{x<y\twoheadrightarrow\mathsf{true}}) & \bot A \to \bot\mathsf{0} \mid \bot\mathsf{s}(\bot A) \mid \bot\mathsf{true} \mid \bot\mathsf{false} \\
\Gamma^{(x,y)}_{y<x\twoheadrightarrow\mathsf{true}} \to \Gamma^{(y,x)}_{x<y\twoheadrightarrow\mathsf{true}} &
\end{array}
$$

It is easy to see that

- $L(\mathcal{G}_4, \Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}}) \subseteq \mathcal{T}(\{\mathsf{0s}, \mathsf{ss}, \bot\mathsf{0}, \bot\mathsf{s}, \bot\mathsf{true}, \bot\mathsf{false}\})$,
- $L(\mathcal{G}_4, \Gamma^{(x,y)}_{y<x\twoheadrightarrow\mathsf{true}}) \subseteq \mathcal{T}(\{\mathsf{s0}, \mathsf{ss}, \mathsf{0}\bot, \mathsf{s}\bot, \mathsf{true}\bot, \mathsf{false}\bot\})$,

and hence, there is no shared constant between the two sets. This means that

$$L(\mathcal{G}_4, \Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}}) \cap L(\mathcal{G}_4, \Gamma^{(x,y)}_{y<x\twoheadrightarrow\mathsf{true}}) = \emptyset$$

and hence

$$[\![L(\mathcal{G}_1, \Gamma_{x<y\twoheadrightarrow\mathsf{true}})]\!] \cap [\![L(\mathcal{G}_1, \Gamma_{y<x\twoheadrightarrow\mathsf{true}})]\!] = \emptyset.$$

Note that the emptiness problem of RTGs is decidable, and hence we can decide the emptiness problem of $L(\mathcal{G}_4, \Gamma^{(x,y)}_{x<y\twoheadrightarrow\mathsf{true}}) \cap L(\mathcal{G}_4, \Gamma^{(x,y)}_{y<x\twoheadrightarrow\mathsf{true}})$.

The following example illustrates both why not all SSGs can be transformed and why we adopt the assumption.

Example 5.3 Let $\mathcal{G}_5$ be the following SSG which does not satisfy the assumption:

$$(\Gamma_{x\twoheadrightarrow y}, \{\Gamma_{x\twoheadrightarrow y}\}, \{\ \Gamma_{x\twoheadrightarrow y} \to \{x \mapsto 0,\ y \mapsto 0\} \mid \mathrm{REC}(\Gamma_{x\twoheadrightarrow y}, \{x' \mapsto x,\ y' \mapsto y\}) \bullet \{x \mapsto \mathsf{s}(x'),\ y \mapsto \mathsf{s}(\mathsf{s}(y'))\}\ \}).$$

The domains of substitutions generated by $\mathcal{G}_5$ w.r.t. x,y is $\{(\mathsf{s}^n(0), \mathsf{s}^{2n}(0)) \mid n \geq 0\}$ which is not recognizable. This implies that there is no RTG generating this set, while every substitution appearing in $\mathcal{G}_5$ preserves linearity.

Let us now apply $\mathcal{R}an$ to $\mathcal{G}_5$, while $\mathcal{G}_5$ does not satisfy the assumption. To generate rules from $\Gamma_{x\twoheadrightarrow y} \to \mathrm{REC}(\Gamma_{x\twoheadrightarrow y}, \{x' \mapsto x,\ y' \mapsto y\}) \bullet \{x \mapsto \mathsf{s}(x'),\ y \mapsto \mathsf{s}(\mathsf{s}(y'))\}$, we need to compute $\langle \mathsf{s}(x),\ \mathsf{s}(\mathsf{s}(y))\rangle_{\Gamma_{x\twoheadrightarrow y}}$, resulting in $\mathsf{ss}(\langle x,\ \mathsf{s}(y)\rangle_{\Gamma_{x\twoheadrightarrow y}})$. The first argument x of $\langle x,\ \mathsf{s}(y)\rangle_{\Gamma_{x\twoheadrightarrow y}}$ cannot be instantiated any more without $\Gamma_{x\twoheadrightarrow y}$. Then, let us define $\langle x,\ \mathsf{s}(y)\rangle_{\Gamma_{x\twoheadrightarrow y}} = \Gamma^{(x,\mathsf{s}(y))}_{x\twoheadrightarrow y}$. Then, the non-terminal $\Gamma^{(x,\mathsf{s}(y))}_{x\twoheadrightarrow y}$ is not generated in computing the set of non-terminals ($\mathcal{N}' \cup \mathcal{N}_A$ in Definition 5.1). Let us now add $\Gamma^{(x,\mathsf{s}(y))}_{x\twoheadrightarrow y}$ into the set of non-terminals, and generate rules for $\Gamma^{(x,\mathsf{s}(y))}_{x\twoheadrightarrow y}$ from $\Gamma_{x\twoheadrightarrow y} \to \mathrm{REC}(\Gamma_{x\twoheadrightarrow y}, \{x' \mapsto x,\ y' \mapsto y\}) \bullet \{x \mapsto \mathsf{s}(x'),\ y \mapsto \mathsf{s}(\mathsf{s}(y'))\}$. Then, we need non-terminal $\Gamma^{(x,\mathsf{s}(\mathsf{s}(\mathsf{s}(y))))}_{x\twoheadrightarrow y}$. In summary, we need infinitely many non-terminals and their production rules. The assumption enables us to avoid such a case.

Finally, we show correctness of the transformation in Definition 5.1, i.e., that $L(\mathcal{R}an(\mathcal{G}, T, x_1, x_2))$ is an overapproximation of the ranges of ground substitutions obtained from $[\![L(\mathcal{G}, \Gamma_T)]\!]$ w.r.t. x_1, x_2. We first show some auxiliary lemmas, and then show the main theorem.

Lemma 5.4 *Let T be a goal clause, $t_1, t_2 \in \mathcal{T}(\mathcal{C}, \mathcal{V})$, $\theta \in Subst(\mathcal{C})$, $\xi \in Subst(\mathcal{C})$ such that $\mathcal{D}om(\theta) \cap \mathcal{D}om(\xi) = \emptyset$ and $\mathcal{D}om(\theta) \cup \mathcal{D}om(\xi) = \mathcal{V}ar(t_1, t_2)$. Note that $\theta \cup \xi = \theta\xi = \xi\theta$. Let $\xi_A = \{x \mapsto A \mid x \in \mathcal{D}om(\xi)\}$ and $u \in \langle \xi_A t_1,\ \xi_A t_2\rangle_T$. Suppose that for all positions $p \in \mathcal{P}os(t_1) \cap \mathcal{P}os(t_2)$, both of the following hold:*

- *if $t_1|_p \in \mathcal{D}om(\theta)$, then $t_2|_p \in \mathcal{D}om(\theta) \cup \mathcal{T}(\mathcal{C}, \mathcal{D}om(\xi))$, and*
- *if $t_2|_p \in \mathcal{D}om(\theta)$, then $t_1|_p \in \mathcal{D}om(\theta) \cup \mathcal{T}(\mathcal{C}, \mathcal{D}om(\xi))$.*

Then, all of the following hold:

(a) $\mathcal{P}os([\theta\xi t_1,\ \theta\xi t_2]) \supseteq \mathcal{P}os([t_1,\ t_2]) \supseteq \mathcal{P}os(u)$ (i.e., $\mathcal{P}os(t_1) \cup \mathcal{P}os(t_2) \supseteq \mathcal{P}os(u)$),

(b) for any position $p \in \mathcal{P}os(t_1) \cap \mathcal{P}os(t_2)$, all of the following hold:

- *if $t_1|_p = x \in \mathcal{D}om(\theta)$ and $t_2|_p = y \in \mathcal{D}om(\theta)$, then $([t_1,\ t_2])|_p = xy$ (i.e., $([\theta\xi t_1,\ \theta\xi t_2])|_p = [\theta x,\ \theta y])$) and $u|_p = \Gamma^{(x,y)}_T$*
- *if $t_1|_p = x \in \mathcal{D}om(\theta)$ and $t_2|_p = y \in \mathcal{D}om(\xi)$, then $([t_1,\ t_2])|_p = xy$ (i.e., $([\theta\xi t_1,\ \theta\xi t_2])|_p = [\theta x,\ \xi y]$) and $u|_p = \Gamma^{(x,A)}_T$*
- *if $t_1|_p = x \in \mathcal{D}om(\xi)$ and $t_2|_p = y \in \mathcal{D}om(\theta)$, then $([t_1,\ t_2])|_p = xy$ (i.e, $([\theta\xi t_1,\ \theta\xi t_2])|_p = [\xi x,\ \theta y]$) and $u|_p = \Gamma^{(A,y)}_T$*
- *if $t_1|_p = x \in \mathcal{D}om(\xi)$ and $t_2|_p = y \in \mathcal{D}om(\xi)$, then $([t_1,\ t_2])|_p = xy$ (i.e., $([\theta\xi t_1,\ \theta\xi t_2])|_p = [\xi x,\ \xi y]$) and $u|_p = AA$*
- *if $t_1|_p = x \in \mathcal{D}om(\theta)$ and $root(t_2|_p) = \mathsf{g} \in \mathcal{C}$, then $root(([t_1,\ t_2])|_p) = x\mathsf{g}$ (i.e., $([\theta\xi t_1,\ \theta\xi t_2])|_p = [\theta x,\ \xi(t_2|_p)]$) and $u|_p = \Gamma^{(x,\xi_A(t_2|_p))}_T$*

- *if* $t_1|_p = x \in \mathcal{D}om(\xi)$ *and* $root(t_2|_p) = \mathsf{g} \in \mathcal{C}$*, then* $root(([t_1,\ t_2])|_p) = x\mathsf{g}$ *(i.e.,* $([\theta\xi t_1,\ \theta\xi t_2])|_p = [\xi x,\ \xi(t_2|_p)]$*),* $u|_p \in \langle A,\ \xi_A(t_2|_p)\rangle_\perp$*, and there exists a term* $t_1' \in \mathcal{T}(\mathcal{C},\mathcal{V})$ *and a term* $u' \in \langle A,\ \xi_A(t_2|_p)\rangle_\perp$ *such that* $t_1' \leq \xi x$*,* $u' = [\xi_A' t_1',\ t_2|_p]$*,*
  - *for all* $q \in \mathcal{P}os(t_1') \cap \mathcal{P}os(t_2|_p)$*,* $\xi_A'(t_1'|_q) = A$ *if and only if* $t_2|_{pq} = A$*,*[7] *and*
  - *for all* $q \in \mathcal{P}os(t_1') \setminus \mathcal{P}os(t_2|_p)$*,* $\xi_A'(t_1'|_q) = A$*,*[8]

  where $\xi_A' = \{x \mapsto A \mid x \in \mathcal{D}om(t_1')\}$*,*
- *if* $root(t_1|_p) = \mathsf{f} \in \mathcal{C}$ *and* $t_2|_p = y \in \mathcal{D}om(\theta)$*, then* $root(([t_1,\ t_2])|_p) = \mathsf{f}y$ *(i.e.,* $([\theta\xi t_1,\ \theta\xi t_2])|_p = [\xi(t_1|_p),\ \theta y]$*) and* $u|_p = \Gamma_T^{(\xi_A(t_1|_p),y)}$
- *if* $root(t_1|_p) = \mathsf{f} \in \mathcal{C}$ *and* $t_2|_p = y \in \mathcal{D}om(\xi)$*, then* $root(([t_1,\ t_2])|_p) = \mathsf{f}y$ *(i.e.,* $([\theta\xi t_1,\ \theta\xi t_2])|_p = [\xi(t_1|_p),\ \xi y]$*),* $u|_p \in \langle \xi_A(t_1|_p),\ A\rangle_\top$*, and there exists a term* $t_2' \in \mathcal{T}(\mathcal{C},\mathcal{V})$ *and a term* $u' \in \langle \xi_A(t_1|_p),\ A\rangle_\top$ *such that* $t_2' \leq \xi y$*,* $u' = [t_1|_p,\ \xi_A' t_2']$*,*
  - *for all* $q \in \mathcal{P}os(t_2') \cap \mathcal{P}os(t_1|_p)$*,* $\xi_A'(t_2'|_q) = A$ *if and only if* $t_1|_{pq} = A$*, and*
  - *for all* $q \in \mathcal{P}os(t_2') \setminus \mathcal{P}os(t_1|_p)$*,* $\xi_A'(t_2'|_q) = A$*,*

  where $\xi_A' = \{x \mapsto A \mid x \in \mathcal{D}om(t_2')\}$*,*
- *if* $root(t_1|_p) = \mathsf{f} \in \mathcal{C}$ *and* $root(t_2|_p) = \mathsf{g} \in \mathcal{C}$*, then* $root(([t_1,\ t_2])|_p) = root(([\theta\xi t_1,\ \theta\xi t_2])|_p) = root(u|_p) = \mathsf{fg}$ *(i.e.,* $([\theta\xi t_1,\ \theta\xi t_2])|_p = [\theta\xi(t_1|_p),\ \theta\xi(t_2|_p)]$*),*

(c) for any position $p \in \mathcal{P}os(t_1) \setminus \mathcal{P}os(t_2)$*, both of the following hold:*

- *if* $t_1|_p = x \in \mathcal{D}om(\theta)$*, then* $([t_1,\ t_2])|_p = x\perp$ *(i.e,* $([\theta\xi t_1,\ \theta\xi t_2])|_p = ([\theta x,\ \perp])|_p$*) and* $u|_p = \Gamma_T^{(x,\perp)}$*,*
- *if* $t_1|_p = x \in \mathcal{D}om(\xi)$*, then* $([t_1,\ t_2])|_p = x\perp$ *(i.e.,* $([\theta\xi t_1,\ \theta\xi t_2])|_p = ([\xi x,\ \perp])|_p$*) and* $u|_p = A\perp$*, and*
- *if* $root(t_1|_p) = \mathsf{f} \in \mathcal{C}$*, then* $root(([t_1,\ t_2])|_p) = root(([\theta\xi t_1,\ \theta\xi t_2])|_p) = root(u|_p) = \mathsf{f}\perp$ *(i.e.,* $([\theta\xi t_1,\ \theta\xi t_2])|_p = ([\theta\xi(t_1|_p),\ \perp])|_p$*),*

and

(d) for any position $p \in \mathcal{P}os(t_2) \setminus \mathcal{P}os(t_1)$*, both of the following hold:*

- *if* $t_2|_p = y \in \mathcal{D}om(\theta)$*, then* $([t_1,\ t_2])|_p = \perp y$ *(i.e.,* $([\theta\xi t_1,\ \theta\xi t_2])|_p = ([\perp,\ \theta y])|_p$*) and* $u|_p = \Gamma_T^{(\perp,y)}$*,*
- *if* $t_2|_p = y \in \mathcal{D}om(\xi)$*, then* $([t_1,\ t_2])|_p = \perp y$ *(i.e.,* $([\theta\xi t_1,\ \theta\xi t_2])|_p = ([\perp,\ \xi y])|_p$*) and* $u|_p = \perp A$*, and*
- *if* $root(t_2|_p) = \mathsf{g} \in \mathcal{C}$*, then* $root(([t_1,\ t_2])|_p) = root(([\theta\xi t_1,\ \theta\xi t_2])|_p) = root(u|_p) = \perp\mathsf{g}$ *(i.e.,* $([\theta\xi t_1,\ \theta\xi t_2])|_p = ([\perp,\ \theta\xi(t_2|_p)])|_p$*).*

Proof. By definition, the claim (a) is trivial. The claims (b)–(d) can be proved by induction on the length of p. For the claims (b), (c), and (d), we make a case distinction depending on what $t_1|_p$ and $t_2|_p$ are, what $t_1|_p$ is, and what $t_2|_p$ is, respectively. □

Lemma 5.5 *Let* T *be a goal clause,* $t_1, t_2 \in \mathcal{T}(\mathcal{C},\mathcal{V})$*,* $\theta \in Subst(\mathcal{C})$*,* $\xi \in Subst(\mathcal{C})$ *such that* $\mathcal{D}om(\theta) \cap \mathcal{D}om(\xi) = \emptyset$ *and* $\mathcal{D}om(\theta) \cup \mathcal{D}om(\xi) = \mathcal{V}ar(t_1,t_2)$*. Note that* $\theta \cup \xi = \theta\xi = \xi\theta$*. Let* $\xi_A = \{x \mapsto A \mid x \in \mathcal{D}om(\xi)\}$*. Suppose that for all positions* $p \in \mathcal{P}os(t_1) \cap \mathcal{P}os(t_2)$*, both of the following hold:*

- *if* $t_1|_p \in \mathcal{V} \cap \mathcal{D}om(\theta)$*, then* $t_2|_p \in \mathcal{D}om(\theta) \cup \mathcal{T}(\mathcal{C},\mathcal{D}om(\xi))$*, and*

[7] This implies that if $q \in \mathcal{P}os(t_1') \cap \mathcal{P}os(t_2|_p)$, then $pq \in \mathcal{P}os(u')$ and $u|_{pq} = AA$.
[8] This implies that if $q \in \mathcal{P}os(t_1') \setminus \mathcal{P}os(t_2|_p)$, then $pq \in \mathcal{P}os(u')$ and $u|_{pq} = A\perp$.

- *if* $t_2|_p \in \mathcal{V} \cap \mathcal{D}om(\theta)$, *then* $t_1|_p \in \mathcal{D}om(\theta) \cup \mathcal{T}(\mathcal{C}, \mathcal{D}om(\xi))$.

Then, there exists a term $u \in \langle \xi_A t_1,\ \xi_A t_2 \rangle_T$, *a context* $C[] \in \mathcal{T}((\mathcal{C} \cup \{\perp\})^2 \cup \{\Box\})$, *terms* $S_1, \ldots, S_n$, *and non-terminals* $\Gamma_1, \ldots, \Gamma_n$ *such that* $[\,\theta\xi t_1,\ \theta\xi t_2\,] = C[S_1, \ldots, S_n]$, $u = C[\Gamma_1, \ldots, \Gamma_n]$, *and for all* $1 \leq i \leq n$, *all of the following hold:*

- $S_i = xy$ *if and only if* $\Gamma_i = \Gamma_T^{(x,y)}$,
- $S_i = [\,\theta x,\ \xi(t_2|_p)\,]$ *if and only if* $\Gamma_i = \Gamma_T^{(x, \xi_A(t_2|_p))}$ *for some* $p \in \mathcal{P}os(t_2)$,
- $S_i = [\,\theta x,\ \perp\,]$ *if and only if* $\Gamma_i = \Gamma_T^{(x,\perp)}$,
- $S_i = [\,\xi(t_1|_p),\ \theta y\,]$ *if and only if* $\Gamma_i = \Gamma_T^{(\xi_A(t_1|_p), y)}$ *for some* $p \in \mathcal{P}os(t_1)$,
- $S_i = [\,\perp,\ \theta y\,]$ *if and only if* $\Gamma_i = \Gamma_T^{(\perp, y)}$,
- $S_i = [\,\xi(t_1|_p),\ \xi(t_2|_p)\,]$ *for some* $p \in \mathcal{P}os(\xi t_1) \cap \mathcal{P}os(\xi t_2)$ *if and only if* $\Gamma_i = AA$,
- $S_i = [\,\xi(t_1|_p),\ \perp\,]$ *for some* $p \in \mathcal{P}os(\xi t_1) \setminus \mathcal{P}os(\xi t_2)$ *if and only if* $\Gamma_i = A\perp$, *and*
- $S_i = [\,\perp,\ \xi(t_2|_p)\,]$ *for some* $p \in \mathcal{P}os(\xi t_2) \setminus \mathcal{P}os(\xi t_1)$ *if and only if* $\Gamma_i = \perp A$.

Proof. Using Lemma 5.4, this lemma can be proved by structural induction on t_1, t_2. □

Lemma 5.6 *Let* $\mathcal{G}$ *be an SSG* $(\Gamma_{T_0}, \mathcal{N}, \mathcal{P})$, $\Gamma_T \in \mathcal{N}$, $x_1, x_2 \in \mathcal{V}ar(T)$, $\mathcal{R}an(\mathcal{G}, T, x_1, x_2)$ *be constructed, and* $\mathcal{G}' = \mathcal{R}an(\mathcal{G}, T, x_1, x_2)$. *Let* $t_1, t_2 \in \mathcal{T}(\mathcal{C}, \mathcal{V})$, $\xi \in Subst(\mathcal{C})$ *with* $\mathcal{D}om(\xi) \supseteq \mathcal{V}ar(t_1, t_2)$, *and* $\xi_A = \{x \mapsto A \mid x \in \mathcal{V}ar(t_1, t_2)\}$. *Then, all of the following hold:*

- *there exists a term* $u \in \langle \xi_A t_1,\ \xi_A t_2 \rangle_\top$ *such that* $u \to^*_{\mathcal{G}'} [\,\xi t_1,\ \xi t_2\,]$,
- *there exists a term* $u \in \langle \xi_A t_1,\ \perp \rangle_\top$ *such that* $u \to^*_{\mathcal{G}'} [\,\xi t_1,\ \perp\,]$, *and*
- *there exists a term* $u \in \langle \perp,\ \xi_A t_2 \rangle_\top$ *such that* $u \to^*_{\mathcal{G}'} [\,\perp,\ \xi t_2\,]$.

Proof. Using the definition of $\mathcal{P}_{AA}$, $\mathcal{P}_{A\perp}$, and $\mathcal{P}_{\perp A}$, and Lemma 5.4, this lemma can be proved by structural induction on t_1, t_2. □

Theorem 5.7 *Let* $\mathcal{G}$ *be an SSG* $(\Gamma_{T_0}, \mathcal{N}, \mathcal{P})$, $\Gamma_T \in \mathcal{N}$, $x_1, x_2 \in \mathcal{V}ar(T)$, *and* $\mathcal{R}an(\mathcal{G}, T, x_1, x_2)$ *be constructed (i.e.,* $\mathcal{P}|_T$ *satisfies the assumption). Then,*

$$L(\mathcal{R}an(\mathcal{G}, T, x_1, x_2)) \supseteq \{[\,\xi\theta x_1,\ \xi\theta x_2\,] \mid \theta \in [\![L(\mathcal{G}, \Gamma_T)]\!],\ \xi \in Subst(\mathcal{C}),\ \mathcal{V}ar(\theta x_1, \theta x_2) = \mathcal{D}om(\xi)\}.$$

Proof. Let $\mathcal{G}' = \mathcal{R}an(\mathcal{G}, T, x_1, x_2)$. It suffices to show that for all $\Gamma_{T'} \in \mathcal{N}$, $t_1, t_2 \in \mathcal{V}ar(T') \cup Patterns(\mathcal{P}) \cup \{\perp\}$ with $\{t_1, t_2\} \cap \mathcal{V} \neq \emptyset$, and $e \in L(\mathcal{G}, \Gamma_{T'})$ with $\theta = [\![e]\!]$, we have $[\,\xi\theta t_1,\ \xi\theta t_2\,] \in L(\mathcal{G}', \Gamma_{T'}^{(t_1,t_2)})$ for all substitutions $\xi \in Subst(\mathcal{C})$ with $\mathcal{D}om(\xi) = \mathcal{V}ar(\xi\theta t_1, \xi\theta t_2)$. We prove this claim by induction on the length of derivations from $\Gamma_{T'}$ to e. We make a case distinction depending on which rule is applied at the first step.

- The case where $\Gamma_{T'} \to \theta$ is applied. By construction, we have the following production rule $\Gamma_{T'}^{(t_1,t_2)} \to u \in \mathcal{G}'$ for each $u \in \langle \xi_A\theta t_1,\ \xi_A\theta t_2 \rangle_\top$, where $\xi_A = \{x \mapsto A \mid x \in \mathcal{V}ar(\xi_A\theta t_1, \xi_A\theta t_2)\}$. Then, the claim follows from Lemma 5.6.

- The remaining case where $\Gamma_{T'} \to \textsc{rec}(\Gamma_{T''}, \delta) \bullet \sigma$ is applied. Suppose that $\Gamma_{T''} \to^*_{\mathcal{G}'} e'$ and $\theta = [\![\textsc{rec}(e', \delta) \bullet \sigma]\!]$. Let $\theta' = [\![e']\!]$. Then, $\theta = (\theta'\delta)|_{\mathcal{D}om(\delta)}\sigma$. By construction, we have the following production rule $\Gamma_{T'}^{(t_1,t_2)} \to u \in \mathcal{G}'$ for each $u \in \langle \xi_A\delta\sigma t_1,\ \xi_A\delta\sigma t_2\rangle_{T'}$. where $\xi_A = \{x \mapsto A \mid x \in \mathcal{V}ar(\delta\sigma t_1, \delta\sigma t_2) \setminus \mathcal{V}ar(T'')\}$. By the assumption, $\mathcal{V}ar(T'') = \mathcal{VR}an(\delta)$, and thus, $\mathcal{D}om(\xi_A) \cap \mathcal{VR}an(\delta) = \emptyset$. Since ξ_A is a ground substitution, we have that $\xi_A\delta\sigma t_i = \delta\xi_A|_{\mathcal{VR}an(\sigma)\setminus\mathcal{D}om(\delta)}\sigma t_i$. Since ξ is a ground substitution and $\mathcal{D}om(\xi) \supseteq \mathcal{VR}an(\theta) = \mathcal{VR}an((\theta'\delta)|_{\mathcal{D}om(\delta)}\sigma)$, we have that
  - $\xi\theta t_i = \xi(\theta'\delta)|_{\mathcal{D}om(\delta)}\sigma t_i = \xi\theta'\delta\xi|_{\mathcal{D}om(\mathcal{VR}an(\sigma)\setminus\mathcal{D}om(\delta)}\sigma t_i$, and
  - $\xi_A\delta\sigma t_i = \xi_A\delta\xi_A|_{\mathcal{D}om(\mathcal{VR}an(\sigma)\setminus\mathcal{D}om(\delta)}\sigma t_i$,

  and hence

  $$\langle \xi_A\delta\sigma t_1,\ \xi_A\delta\sigma t_2\rangle_{T''} = \langle \xi_A\delta\xi_A|_{\mathcal{D}om(\mathcal{VR}an(\sigma)\setminus\mathcal{D}om(\delta)}\sigma t_1,\ \xi_A\delta\xi_A|_{\mathcal{D}om(\mathcal{VR}an(\sigma)\setminus\mathcal{D}om(\delta)}\sigma t_2\rangle_{T''}.$$

  It follows from Lemma 5.5 that there exists a term $u \in \langle \xi_A\delta\sigma t_1,\ \xi_A\delta\sigma t_2\rangle_{T''}$, a context $C[] \in \mathcal{T}((\mathcal{C} \cup \{\perp\})^2 \cup \{\square\})$, terms $S_1, \ldots, S_n$, and non-terminals $\Gamma_1, \ldots, \Gamma_n$ such that $[\,\xi\theta t_1,\ \xi\theta t_2\,] = C[S_1, \ldots, S_n]$, $u = C[\Gamma_1, \ldots, \Gamma_n]$, and for all $1 \le i \le n$, all of the following hold:
  - $S_i = xy$ if and only if $\Gamma_i = \Gamma_{T''}^{(x,y)}$,
  - $S_i = [\,\xi\theta x,\ \xi\theta(t_2|_p)\,]$ if and only if $\Gamma_i = \Gamma_{T''}^{(x,\xi_A(t_2|_p))}$ for some $p \in \mathcal{P}os(t_2)$,
  - $S_i = [\,\xi\theta x,\ \perp\,]$ if and only if $\Gamma_i = \Gamma_{T''}^{(x,\perp)}$,
  - $S_i = [\,\xi\theta(t_1|_p),\ \xi\theta y\,]$ if and only if $\Gamma_i = \Gamma_{T''}^{(\xi_A(t_1|_p),y)}$ for some $p \in \mathcal{P}os(t_1)$,
  - $S_i = [\,\perp,\ \xi\theta y\,]$ if and only if $\Gamma_i = \Gamma_{T''}^{(\perp,y)}$,
  - $S_i = [\,\xi\theta(t_1|_p),\ \xi\theta(t_2|_p)\,]$ for some $p \in \mathcal{P}os(\xi\theta t_1) \cap \mathcal{P}os(\xi\theta t_2)$ if and only if $\Gamma_i = AA$,
  - $S_i = [\,\xi\theta(t_1|_p),\ \perp\,]$ for some $p \in \mathcal{P}os(\xi\theta t_1) \setminus \mathcal{P}os(\xi\theta t_2)$ if and only if $\Gamma_i = A\perp$, and
  - $S_i = [\,\perp,\ \xi\theta(t_2|_p)\,]$ for some $p \in \mathcal{P}os(\xi\theta t_2) \setminus \mathcal{P}os(\xi\theta t_1)$ if and only if $\Gamma_i = \perp A$.

  In the case where Γ_i is $\Gamma_{T''}^{(x,y)}$, $\Gamma_{T''}^{(x,\xi_A(t_2|_p))}$, $\Gamma_{T''}^{(x,\perp)}$, $\Gamma_{T''}^{(\xi_A(t_1|_p),y)}$, or $\Gamma_{T''}^{(\perp,y)}$, it follows from the induction hypothesis that $\Gamma_i \to^*_{\mathcal{G}'} S_i$. In the remaining case where Γ_i is AA, $A\perp$, or $\perp A$, it follows from Lemma 5.6 that $\Gamma_i \to^*_{\mathcal{G}'} S_i$. Therefore, we have that $\Gamma_{T'}^{(t_1,t_2)} \to_{\mathcal{G}'} u = C[\Gamma_1, \ldots, \Gamma_n] \to^*_{\mathcal{G}'} C[S_1, \ldots, S_n] = [\,\xi\theta t_1,\ \xi\theta t_2\,]$, and hence, $[\,\xi\theta t_1,\ \xi\theta t_2\,] \in L(\mathcal{G}', \Gamma_{T'}^{(t_1,t_2)})$. $\square$

The converse inclusion (i.e., $L(\mathcal{R}an(\mathcal{G}, T, x_1, x_2)) \subseteq \{[\,\xi\theta x_1,\ \xi\theta x_2\,] \mid \theta \in [\![L(\mathcal{G}, \Gamma_T)]\!],\ \ldots\}$) does not hold in general (cf. [21, Example 31]).

6 Conclusion

In this paper, under a certain syntactic condition, we showed a transformation of the grammar representation of a narrowing tree into an RTG that overapproximately generates the ranges of ground substitutions generated by the grammar representation. We showed a precise definition of the transformation and proved that the language of the transformed RTG is an overapproximation of the ranges of ground substitutions generated by the grammar representation. We will make an experiment to evaluate the usefulness of the transformation in e.g., proving confluence of CTRSs.

The syntactic assumption in Section 5 is a sufficient condition to, given an SSG, obtain an RTG that generates the ranges of ground substitutions generated by the SSG. It is not known yet whether the assumption is a necessary condition or not. We will try to clarify this point.

As stated in Section 5, the converse inclusion of Theorem 5.7, $L(\mathcal{R}an(\mathcal{G},T,x_1,x_2)) \subseteq \{[\,\xi\theta x_1,\ \xi\theta x_2\,] \mid \theta \in [\![L(\mathcal{G},\Gamma_T)]\!],\ \ldots\}$, does not hold in general. However, the converse inclusion must hold for an SSG such that all substitutions in the SSG preserve linearity, i.e., for any substitution σ in the SSG, σx is linear for all $x \in \mathcal{D}om(\sigma)$, and $\mathcal{V}ar(\sigma x) \cap \mathcal{V}ar(\sigma y) = \emptyset$ for all $x,y \in \mathcal{D}om(\sigma)$ such that $x \neq y$. We will prove this conjecture and try to find other sufficient conditions for the converse inclusion.

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

References

[1] Franz Baader & Tobias Nipkow (1998): *Term Rewriting and All That*. Cambridge University Press, doi:10.1017/CBO9781139172752.

[2] Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Löding, Sophie Tison & Marc Tommasi (2007): *Tree Automata Techniques and Applications*. Available on: `http://www.grappa.univ-lille3.fr/tata`. Release October, 12th 2007.

[3] Nachum Dershowitz, Mitsuhiro Okada & G. Sivakumar (1988): *Canonical Conditional Rewrite Systems*. In: *Proceedings of the 9th International Conference on Automated Deduction*, *Lecture Notes in Computer Science* 310, Springer, pp. 538–549, doi:10.1007/BFb0012855.

[4] Francisco Durán, Salvador Lucas, José Meseguer, Claude Marché & Xavier Urbain (2004): *Proving termination of membership equational programs*. In: *Proceedings of the 2004 ACM SIGPLAN Workshop on Partial Evaluation and Semantics-based Program Manipulation*, ACM, pp. 147–158, doi:10.1145/1014007.1014022.

[5] Guillaume Feuillade & Thomas Genet (2003): *Reachability in Conditional Term Rewriting Systems*. *Electronic Notes in Theoretical Computer Science* 86(1), pp. 133–146, doi:10.1016/S1571-0661(04)80658-3.

[6] Thomas Genet & Vlad Rusu (2010): *Equational approximations for tree automata completion*. *Journal of Symbolic Computation* 45(5), pp. 574–597, doi:10.1016/j.jsc.2010.01.009.

[7] Thomas Genet & Valérie Viet Triem Tong (2001): *Reachability Analysis of Term Rewriting Systems with Timbuk*. In Robert Nieuwenhuis & Andrei Voronkov, editors: *Proceedings of the 8th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, *Lecture Notes in Computer Science* 2250, Springer, pp. 695–706, doi:10.1007/3-540-45653-8_48.

[8] Jürgen Giesl, Peter Schneider-Kamp & René Thiemann (2006): *AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework*. In: *Proceedings of the 3rd International Joint Conference on Automated Reasoning*, *Lecture Notes in Computer Science* 4130, Springer, pp. 281–286, doi:10.1007/11814771_24.

[9] Karl Gmeiner (2015): *CoScart: Confluence Prover in Scala*. In: *Proceedings of the 4th International Workshop on Confluence*, p. 45.

[10] Karl Gmeiner & Naoki Nishida (2014): *Notes on Structure-Preserving Transformations of Conditional Term Rewrite Systems*. In: *Proceedings of the first International Workshop on Rewriting Techniques for Program Transformations and Evaluation*, *OpenAccess Series in Informatics* 40, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, pp. 3–14, doi:10.4230/OASIcs.WPTE.2014.3.

[11] Karl Gmeiner, Naoki Nishida & Bernhard Gramlich (2013): *Proving Confluence of Conditional Term Rewriting Systems via Unravelings*. In: *Proceedings of the 2nd International Workshop on Confluence*, pp. 35–39.

[12] Raúl Gutiérrez, Salvador Lucas & Patricio Reinoso (2016): *A tool for the automatic generation of logical models of order-sorted first-order theories*. In: *Proceedings of the XVI Jornadas sobre Programación y Lenguages*, pp. 215–230. Tool available at `http://zenon.dsic.upv.es/ages/`.

[13] Manuel V. Hermenegildo & Francesca Rossi (1989): *On the Correctness and Efficiency of Independent And-Parallelism in Logic Programs*. In: *Proceedings of the North American Conference on Logic Programming*, MIT Press, pp. 369–389.

[14] Jean-Marie Hullot (1980): *Canonical Forms and Unification*. In: *Proceedings of the 5th Conference on Automated Deduction*, *Lecture Notes in Computer Science* 87, Springer, pp. 318–334, doi:10.1007/3-540-10009-1_25.

[15] Salvador Lucas (2018): *A Semantic Approach to the Analysis of Rewriting-Based Systems*. In: *Revised Selected Papers of the 27th International Symposium on Logic-Based Program Synthesis and Transformation*, *Lecture Notes in Computer Science* 10855, Springer, pp. 180–97, doi:10.1007/978-3-319-94460-9_11.

[16] Salvador Lucas & Raúl Gutiérrez (2017): *A Semantic Criterion for Proving Infeasibility in Conditional Rewriting*. In: *Proceedings of the 6th International Workshop on Confluence*, pp. 15–20.

[17] Salvador Lucas, Claude Marché & José Meseguer (2005): *Operational termination of conditional term rewriting systems*. *Information Processing Letters* 95(4), pp. 446–453, doi:10.1016/j.ipl.2005.05.002.

[18] Massimo Marchiori (1996): *Unravelings and Ultra-properties*. In: *Proceedings of the 5th International Conference on Algebraic and Logic Programming*, *Lecture Notes in Computer Science* 1139, Springer, pp. 107–121, doi:10.1007/3-540-61735-3_7.

[19] Aart Middeldorp & Erik Hamoen (1994): *Completeness Results for Basic Narrowing*. *Applicable Algebra in Engineering, Communication and Computing* 5, pp. 213–253, doi:10.1007/BF01190830.

[20] Naoki Nishida, Takayuki Kuroda, Makishi Yanagisawa & Karl Gmeiner (2015): *CO3: a COnverter for proving COnfluence of COnditional TRSs*. In: *Proceedings of the 4th International Workshop on Confluence*, p. 42.

[21] Naoki Nishida & Yuya Maeda (2018): *Narrowing Trees for Syntactically Deterministic Conditional Term Rewriting Systems*. In: *Proceedings of the 3rd International Conference on Formal Structures for Computation and Deduction*, *Leibniz International Proceedings in Informatics* 108, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, pp. 26:1–26:20, doi:10.4230/LIPIcs.FSCD.2018.26.

[22] Naoki Nishida & Germán Vidal (2013): *Computing More Specific Versions of Conditional Rewriting Systems*. In: *Revised Selected Papers of the 22nd International Symposium on Logic-Based Program Synthesis and Transformation*, *Lecture Notes in Computer Science* 7844, Springer, pp. 137–154, doi:10.1007/978-3-642-38197-3_10.

[23] Naoki Nishida & Germán Vidal (2014): *A Finite Representation of the Narrowing Space*. In: *Revised Selected Papers of the 23rd International Symposium on Logic-Based Program Synthesis and Transformation*, *Lecture Notes in Computer Science* 8901, Springer, pp. 54–71, doi:10.1007/978-3-319-14125-1_4.

[24] Naoki Nishida & Germán Vidal (2015): *A framework for computing finite SLD trees*. *Journal of Logic and Algebraic Methods in Programming* 84(2), pp. 197–217, doi:10.1016/j.jlamp.2014.11.006.

[25] Enno Ohlebusch (2002): *Advanced Topics in Term Rewriting*. Springer, doi:10.1007/978-1-4757-3661-8.

[26] Catuscia Palamidessi (1990): *Algebraic Properties of Idempotent Substitutions*. In: *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, *Lecture Notes in Computer Science* 443, Springer, pp. 386–399, doi:10.1007/BFb0032046.

[27] James R. Slagle (1974): *Automated Theorem-Proving for Theories with Simplifiers, Commutativity and Associativity*. *Journal of the ACM* 21(4), pp. 622–642, doi:10.1145/321850.321859.

[28] Thomas Sternagel & Aart Middeldorp (2014): *Conditional Confluence (System Description)*. In: *Proceedings of the Joint International Conference on Rewriting and Typed Lambda Calculi*, *Lecture Notes in Computer Science* 8560, Springer, pp. 456–465, doi:10.1007/978-3-319-08918-8_31.

[29] Taro Suzuki, Aart Middeldorp & Tetsuo Ida (1995): *Level-Confluence of Conditional Rewrite Systems with Extra Variables in Right-Hand Sides*. In: *Proceedings of the 6th International Conference on Rewriting Techniques and Applications*, *Lecture Notes in Computer Science* 914, Springer, pp. 179–193, doi:10.1007/3-540-59200-8_56.