A Rule-Based Framework for Solving Regular Context Sequence Constraints *

Temur Kutsia[1]** and Mircea Marin[2]***

[1] Research Institute for Symbolic Computation
Johannes Kepler University, A-4040 Linz, Austria
tkutsia@risc.uni-linz.ac.at
[2] Graduate School of Systems and Information Engineering
University of Tsukuba, Tsukuba 305-8573, Japan
mmarin@cs.tsukuba.ac.jp

Abstract. We propose a framework for solving equational and membership constraints for terms built over individual, sequence, function, and context variables and flexible arity symbols. Each membership constraint couples a variable with a regular expression on terms or contexts. There can be several membership constraints with the same constrained variable, and expressions may contain variables themselves. A membership constraint is satisfied if an instance of the constrained variable belongs to the language generated by the corresponding instance of the regular expression. We identify sufficient syntactic restrictions that allow us to use matching techniques for solving such constraints, describe a complete algorithm, and discuss applications.

1 Introduction

We propose a framework for solving constraint systems that consist of equations and membership constraints. Terms are built from flexible arity symbols and individual, sequence, function, and constant variables. The equational part of constraint systems is a unification problem. Membership constraints restrict context and sequence variables to belong certain regular languages. We call such systems regular context sequence constraints, Rcs constraints in short, to underline their most important ingredients.

The equational part of Rcs constraints subsumes context unification that is a very hard problem: It may have infinitely many incomparable solutions,

* **Note:** This is a full and revised version of the paper submitted to Rta'2006 in Seattle. In particular, it corrects Example 3 on page 5, the definition of well-moded membership constraint on page 6, and makes the formulation of the rules SVREC,CxREC,CVREC, IORET and IOREC more accurate; see pages 8, 10, and 21.

** Supported by the Austrian Science Foundation (FWF) under the Project SFB F1302 and F1322.

*** Supported by the JSPS Grant-in-Aid no. 17700025 for Scientific Research sponsored by the Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT).

and decidability is still open [30]. In contrast, matching with context (and sequence) variables is decidable and finitary. Therefore, we identify conditions on constraints under which unification can be replaced with iterative matching. Such replacement problems have been studied by number of authors in different contexts; see, e.g., [15, 24, 2, 1, 34]. Like them, we approach this problem using the idea of modes, indicating how the arguments of equality should be used, and define a class of well-moded context sequence unification problems. These are the problems that can be reordered into well-moded logic program queries.

We define notions of regular terms and regular contexts. Intuitively, regular terms are terms over the alphabet extended with the standard regular operators. Regular contexts are contexts (i.e. terms with a single hole constant) over the same extended alphabet. Regular terms contain no holes. By regular expression we mean a regular term or a regular context. Note that we use this term in a somewhat more liberal way than the standard definition of regular (tree) expressions [13], because in our regular expressions variables may occur. However, when we define languages generated by regular expressions, these variables behave like (function or individual) constants. A language generated by regular terms contains sequences of terms (hedges), and a language generated by regular contexts contains contexts.

Membership constraints constrain context variables with regular contexts, and sequence variables with regular terms. Constraints are very flexible: Constrained variables may occur in regular expressions, may be constrained with several regular expressions, need not occur in equations, regular expressions may contain unconstrained variables, etc. For the decidability purposes, membership constraints have to fulfill certain natural well-modedness condition. We define regular well-moded context sequence constraints, RWCS constraints in short, as RCS constraints with well-moded equational and membership parts.

Solving an RWCS constraint means to find a substitution that solves the equations and satisfies the membership constraints: The instance of each constrained variable should belong to the language generated by the instance of the regular expression that constrains the variable. We propose a rule-based algorithm that provides a sound, terminating, and complete solving method for RWCS constraints. Flexibility and expressiveness of the framework suggests quite broad range of possible applications. At the end of the paper we briefly discuss some of them, related to constraint solving, programming, and querying.

The paper is organized as follows: Sect. 2 defines the terminology. Sect. 3 introduces regular context sequence constraints. Sect. 4 introduces the solving framework. Sect. 5 discuses an extension of the framework with extra regular operators and restrictions with respect to variable occurrences. Sect. 6 discusses related work and possible applications.

2 Preliminaries

We consider the alphabet consisting of the following mutually disjoint sets of individual variables $\mathcal{V}_{\mathrm{Ind}}$, sequence variables $\mathcal{V}_{\mathrm{Seq}}$, function variables $\mathcal{V}_{\mathrm{Fun}}$, context

variables $\mathcal{V}_{\mathrm{Con}}$, and function symbols $\mathcal{F}$. The sets $\mathcal{V}_{\mathrm{Ind}}$, $\mathcal{V}_{\mathrm{Seq}}$, $\mathcal{V}_{\mathrm{Fun}}$, and $\mathcal{V}_{\mathrm{Con}}$ are countable. The set $\mathcal{F}$ is finite or countably infinite. All the symbols in $\mathcal{F}$ except a distinguished constant $\circ$ (called a *hole*) have flexible arity. We will use x, y, z for individual variables, $\overline{x}, \overline{y}, \overline{z}$ for sequence variables, F, G, H for function variables, $\overline{C}, \overline{D}, \overline{E}$ for context variables, and a, b, c, f, g, h for function symbols.

Terms are constructed using the following grammar:

$$t ::= x \mid \overline{x} \mid \circ \mid f(t_1, \ldots, t_n) \mid F(t_1, \ldots, t_n) \mid \overline{C}(t).$$

In $\overline{C}(t)$ the term t can not be a sequence variable. We will write a for the term $a()$ where $a \in \mathcal{F}$. Terms will be denoted with s, t, r. The *head* of a term is its root symbol. A *ground* term is a term without variables. A *context* is a term with a single occurrence of the hole constant $\circ$. A context C may be applied to a term s that is not a sequence variable, written $C[s]$, and the result is the term consisting of C with $\circ$ replaced by s. We will use C and D for contexts.

A *substitution* is a mapping from individual variables to those terms which are not sequence variables and contain no holes, from sequence variables to finite, possibly empty sequences of terms without holes, from function variables to function variables and symbols, and from context variables to contexts, such that all but finitely many individual and function variables are mapped to themselves, all but finitely many sequence variables are mapped to themselves considered as singleton sequences, and all but finitely many context variables are mapped to themselves applied to the hole. For example, the mapping $\{x \mapsto f(a, \overline{y}), \overline{x} \mapsto \ulcorner\urcorner, \overline{y} \mapsto \ulcorner a, \overline{C}(f(b)), x\urcorner, F \mapsto g, \overline{C} \mapsto g(\circ)\}$ is a substitution.[3] We will use $\sigma, \vartheta, \varphi$, and ε for substitutions, where ε denotes the empty substitution.

For a substitution σ, the domain is the set of variables $dom(\sigma) = \{v \mid v \in \mathcal{V}_{\mathrm{Ind}} \cup \mathcal{V}_{\mathrm{Seq}} \cup \mathcal{V}_{\mathrm{Fun}}, v\sigma \neq v\} \cup \{\overline{C} \mid \overline{C}\sigma \neq \overline{C}(\circ)\}$ and the range is the set $ran(\sigma) = \bigcup_{v \in dom(\sigma)} \{v\sigma\}$.

Substitutions are extended to terms: $v\sigma = \sigma(v)$ for $v \in \mathcal{V}_{\mathrm{Ind}} \cup \mathcal{V}_{\mathrm{Seq}}$, $\overline{C}(t)\sigma = \sigma(\overline{C})[t\sigma]$, $F(t_1, \ldots, t_n)\sigma = \sigma(F)(t_1\sigma, \ldots, t_n\sigma)$, $f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma)$.

A substitution σ is *more general* than ϑ, denoted $\sigma \leq \vartheta$, if there exists a φ such that $\sigma\varphi = \vartheta$. A substitution σ is *more general* than ϑ *on a set of variables* $\mathcal{V}$, denoted $\sigma \leq^{\mathcal{V}} \vartheta$, if there exists a φ such that $v\sigma\varphi = v\vartheta$ for all $v \in \mathcal{V}$. A *term equation* is a pair of terms $\langle s, t \rangle$, written $s \approx t$, where s and t contain no holes and are not sequence variables. The fact that the equation $s \approx t$ has to be solved is written as $s \approx^? t$. A set of variables of a syntactic object O is denoted by $vars(O)$. A *context sequence unification problem,* CSU problem in short, is a finite multiset of equations. A CSU problem is called *well-moded* if it can be ordered as $s_1 \approx^? t_1, \ldots, s_n \approx^? t_n$ where t_1 is ground and $vars(t_j) \subseteq \bigcup_{i=1}^{j-1} vars(s_i)$ for $1 < j \leq n$.

Substitutions are extended to equations in the usual way. A *solution* of a CSU problem $\{s_1 \approx^? t_1, \ldots, s_n \approx^? t_n\}$ is a substitution σ such that $s_i\sigma = t_i\sigma$ for all $1 \leq i \leq n$. A *complete set of solutions* of a CSU problem Γ is a set of substitutions S such that (i) each element of S is a solution of Γ, and (ii) for

[3] To improve readability we write sequences between the symbols $\ulcorner$ and $\urcorner$.

each solution ϑ of Γ there exist a substitution $\sigma \in S$ such that $\sigma \leq \vartheta$. The set S is a *minimal complete set of solutions* of Γ if it is a complete set and two distinct elements of S are incomparable with respect to $\leq$.

An equation $s \approx^? t$ is called a *matching equation*, written $s \ll t$, if t is ground. We will call s the *query* and t the *data*. A CSM *problem* is a CSU problem where each equation is a matching equation. A solution of CSM problem is called a *matcher*. We will show that matching techniques are, in fact, sufficient to solve well-moded unification problems.

Just as a remark, to underline expressiveness of having individual, sequence, function, and context variables in the same language, consider the context unification problem $\{\overline{C}_1(a) \approx^? \overline{C}_2(b)\}$. In the standard case solvability of this problem depends on the signature (or one has to consider n-ary context variables). For instance, one of the solutions is $\{\overline{C}_1 \mapsto f(\circ, b), \overline{C}_2 \mapsto f(a, \circ)\}$, where f does not occur in the problem itself. In our language the problem has a most general unifier $\{\overline{C}_1 \mapsto F(\overline{x}, \overline{D}_1(\circ), \overline{y}, \overline{D}_2(b), \overline{z}), \overline{C}_2 \mapsto F(\overline{x}, \overline{D}_1(a), \overline{y}, \overline{D}_2(\circ), \overline{z})\}$. However, it is not a well-moded problem and is outside the scope of the framework we consider in this paper. Below we consider only well-moded CSU problems.

Example 1. We want to select from the (tree form of) data term $f(g(a,a,a), g(h(b), h(b)), h(c,c))$ the contexts under which there are at least two leaves (not complex subtrees), and all those leaves are the same. We write this problem as a matching equation $\overline{C}(F(G(),\ G(), G()^*)) \ll f(g(a,a,a), g(h(b), h(b)), h(c,c))$ that can be solved by the following two substitutions: $\{\overline{C} \mapsto f(\circ, g(h(b), h(b)), h(c,c)), F \mapsto g, G \mapsto a\}$ and $\{\overline{C} \mapsto f(g(a,a,a), g(h(b), h(b)), \circ), F \mapsto h, G \mapsto c\}$. Complex subtrees are not selected: $G \in \mathcal{V}_{\mathrm{Fun}}$ can not be mapped to $h(b)$.

3 Regular Context Sequence Constraints

We would like to extend terms by allowing regular expressions in their arguments. We write $\ulcorner\urcorner$ for the empty sequence, "," for sequence concatenation, "." for context concatenation, "|" for sequence choice, "+" for context choice, "*" for sequence repetition, and "*" for context repetition. A *regular expression* $\mathtt{E}$ is a common name for a *regular expression for hole-free terms* (that we abbreviate as *regular term*) $\mathtt{T}$ and a *regular expression for contexts* (abbreviated as *regular context*) $\mathtt{C}$, defined by the following grammars:

$$\mathtt{T} ::= x \mid \overline{x} \mid f(\mathtt{T}) \mid F(\mathtt{T}) \mid \overline{C}(\mathtt{T}) \mid \ulcorner\urcorner \mid \ulcorner \mathtt{T}_1, \mathtt{T}_2 \urcorner \mid \mathtt{T}_1 | \mathtt{T}_2 \mid \mathtt{T}^* \mid \mathtt{C}(\mathtt{T}).$$
$$\mathtt{C} ::= \circ \mid f(\mathtt{T}_1, \mathtt{C}, \mathtt{T}_2) \mid F(\mathtt{T}_1, \mathtt{C}, \mathtt{T}_2) \mid \overline{C}(\mathtt{C}) \mid \mathtt{C}_1.\mathtt{C}_2 \mid \mathtt{C}_1 + \mathtt{C}_2 \mid \mathtt{C}^\star.$$

The regular term $\mathtt{T}$ in $\overline{C}(\mathtt{T})$ and $\mathtt{C}(\mathtt{T})$ should satisfy the condition $\mathit{head}(\mathtt{T}) \in \mathcal{V}_{\mathrm{Ind}} \cup \mathcal{V}_{\mathrm{Fun}} \cup \mathcal{V}_{\mathrm{Con}} \cup \mathcal{F}$. We assume that $\mathtt{C}(\mathtt{T})$ is $\mathtt{C}[\mathtt{T}]$ if $\mathtt{C}$ is a context, and do not distinguish between $\mathtt{E}_1, \ulcorner\urcorner, \mathtt{E}_2$ and $\mathtt{E}_1, \mathtt{E}_2$.

Example 2. $\overline{C}(F(f(\overline{x})^*, g(\circ).h(\circ))) + f(\ulcorner g(x), x \urcorner^*, g(\circ))^\star$ is a regular context. $(\overline{C}(F(f(\overline{x})^*, g(\circ).h(\circ))) + f(g(x), x)^*, g(\circ))^\star)\,(f(\overline{x}, a^*))$ is a regular term.

Substitutions are extended to regular expressions in the usual way. We call a regular term *r-term* if it is either $\mathtt{T}_1 \mid \mathtt{T}_2$ or $\mathtt{T}^*$. A regular context is an *r-context* if it is either $\mathtt{C}_1.\mathtt{C}_2$, $\mathtt{C}_1 + \mathtt{C}_2$, or $\mathtt{C}^\star$. A regular term is a term (sometimes we also say *pure term*) if it does not contain any regular operator. The same holds for contexts.

The regular language $\mathcal{L}(\mathtt{T})$ generated by a regular term $\mathtt{T}$ is a set of sequences of hole-free pure terms (or hedges, in terms of [3]) defined as follows:

$$\begin{aligned}
\mathcal{L}(v) &= \{v\}, \text{ if } v \in \mathcal{V}_{\mathrm{Ind}} \cup \mathcal{V}_{\mathrm{Seq}}.\\
\mathcal{L}(\mathtt{h}(\mathtt{T})) &= \{\mathtt{h}(t_1,\ldots,t_n) \mid \ulcorner t_1,\ldots,t_n \urcorner \in \mathcal{L}(\mathtt{T})\},\ \mathtt{h} \in \mathcal{V}_{\mathrm{Fun}} \cup \mathcal{F}.\\
\mathcal{L}(\overline{C}(\mathtt{T})) &= \{\overline{C}(t) \mid t \in \mathcal{L}(\mathtt{T})\}.\\
\mathcal{L}(\ulcorner\urcorner) &= \{\ulcorner\urcorner\}.\\
\mathcal{L}(\ulcorner \mathtt{T}_1, \mathtt{T}_2 \urcorner) &= \{\ulcorner t_1,\ldots,t_k,t_{k+1},\ldots,t_n \urcorner \mid\\
&\qquad \ulcorner t_1,\ldots,t_k \urcorner \in \mathcal{L}(\mathtt{T}_1), \ulcorner t_{k+1},\ldots,t_n \urcorner \in \mathcal{L}(\mathtt{T}_2)\}.\\
\mathcal{L}(\mathtt{T}_1 \mid \mathtt{T}_2) &= \mathcal{L}(\mathtt{T}_1) \cup \mathcal{L}(\mathtt{T}_2).\\
\mathcal{L}(\mathtt{T}^*) &= \bigcup_{n\geq 0} \mathcal{L}(\mathtt{T})^n.\\
\mathcal{L}(\mathtt{C}(\mathtt{T})) &= \{C[t] \mid C \in \mathcal{L}(\mathtt{C}), t \in \mathcal{L}(\mathtt{T})\}.
\end{aligned}$$

Here $\mathcal{L}(\mathtt{T})^0 = \{\ulcorner\urcorner\}$ and $\mathcal{L}(\mathtt{T})^{n+1} = \{\ulcorner t_1,\ldots,t_k,t_{k+1},\ldots,t_m \urcorner \mid \ulcorner t_1,\ldots,t_k \urcorner \in \mathcal{L}(\mathtt{T})^n, \ulcorner t_{k+1},\ldots,t_m \urcorner \in \mathcal{L}(\mathtt{T})\}$. The regular language $\mathcal{L}(\mathtt{C})$ generated by a regular context $\mathtt{C}$ is a set of pure contexts defined as follows:

$$\begin{aligned}
\mathcal{L}(\circ) &= \{\circ\}.\\
\mathcal{L}(\mathtt{h}(\mathtt{T}_1, \mathtt{C}, \mathtt{T}_2)) &= \{\mathtt{h}(t_1,\ldots,t_k,C,t_{k+1},\ldots,t_n) \mid C \in \mathcal{L}(\mathtt{C}),\\
&\qquad \ulcorner t_1,\ldots,t_k \urcorner \in \mathcal{L}(\mathtt{T}_1), \ulcorner t_{k+1},\ldots,t_n \urcorner \in \mathcal{L}(\mathtt{T}_2)\}, \mathtt{h} \in \mathcal{V}_{\mathrm{Fun}} \cup \mathcal{F}.\\
\mathcal{L}(\overline{C}(\mathtt{C})) &= \{\overline{C}(C) \mid C \in \mathcal{L}(\mathtt{C})\}.\\
\mathcal{L}(\mathtt{C}_1.\mathtt{C}_2) &= \{C_1[C_2] \mid C_1 \in \mathcal{L}(\mathtt{C}_1), C_2 \in \mathcal{L}(\mathtt{C}_2)\}.\\
\mathcal{L}(\mathtt{C}_1 + \mathtt{C}_2) &= \mathcal{L}(\mathtt{C}_1) \cup \mathcal{L}(\mathtt{C}_2).\\
\mathcal{L}(\mathtt{C}^\star) &= \bigcup_{n\geq 0} \mathcal{L}(\mathtt{C})^n,
\end{aligned}$$

where $\mathcal{L}(\mathtt{C})^0 = \{\circ\}$ and $\mathcal{L}(\mathtt{C})^{n+1} = \{C_1[C_2] \mid C_1 \in \mathcal{L}(\mathtt{C})^n, C_2 \in \mathcal{L}(\mathtt{C})\}$.

Example 3. Let $\mathtt{T}$ be $f(\overline{C}(\circ), x^*, b)^\star$. Then

$$\begin{aligned}
\mathcal{L}(\mathtt{T}) = \{&\circ, f(\overline{C}(\circ), b), f(\overline{C}(\circ), x, b), \ldots, f(\overline{C}(\circ), x, \ldots, x, b), \ldots,\\
&f(\overline{C}(f(\overline{C}(\circ), b)), b), f(\overline{C}(f(\overline{C}(\circ), b)), x, b), \ldots\}.
\end{aligned}$$

Variables behave like function symbols when languages are generated from regular expressions.

Note that a regular context $\mathtt{C}$ can not be applied to another regular expression unless $\mathtt{C}$ is a pure context. The following example demonstrates this:

Example 4. The expressions $f(\circ)^\star(a)$ and $f(a)^*$ generate different languages:

$$\mathcal{L}(f(\circ)^\star(a)) = \{a, f(a), f(f(a)), \ldots\}$$
$$\mathcal{L}(f(a)^*) = \{\ulcorner\urcorner, f(a), \ulcorner f(a), f(a)\urcorner, \ldots\}.$$

We write $\mathcal{L}(\mathtt{T})_1$ for $\mathcal{L}(\mathtt{T}) \setminus \{\ulcorner\urcorner\}$, $\mathcal{L}(\mathtt{C})_1$ for $\mathcal{L}(\mathtt{C}) \setminus \{\circ\}$, $\mathcal{L}(\mathtt{T})_0 = \mathcal{L}(\mathtt{T})$ and $\mathcal{L}(\mathtt{C})_0 = \mathcal{L}(\mathtt{C})$. *Membership atoms* are atoms of the form Ts $\mathtt{in}$ $\mathtt{T}$ or Cv $\mathtt{in}$ $\mathtt{C}$, where Ts is a finite, possibly empty, sequence of terms, and Cv is either a context or a context variable. *Membership constraints* are finite multisets of pairs $\{(ma_1, \mathtt{f}_1), \ldots, (ma_n, \mathtt{f}_n)\}$ where ma's are a membership atoms and $\mathtt{f}$'s are flags that are boolean expressions (with the possible values 0 or 1). Sometimes, slightly abusing the notation, we also refer to pairs $(ma, \mathtt{f})$ as membership constraints. The intuition behind $(Ts$ $\mathtt{in}$ $\mathtt{T}, \mathtt{f})$ is that $Ts \in \mathcal{L}(\mathtt{T})_\mathtt{f}$.[4] Similarly, the intuition behind $(Cv$ $\mathtt{in}$ $\mathtt{C}, \mathtt{g})$ is that $Cv \in \mathcal{L}(\mathtt{C})_\mathtt{g}$. It will be needed later to guarantee the termination of the algorithm. A membership constraint is called *well-moded* if it can be ordered as $\{(\overline{v}_1$ $\mathtt{in}$ $\mathtt{E}_1, \mathtt{f}_1), \ldots, (\overline{v}_n$ $\mathtt{in}$ $\mathtt{E}_n, \mathtt{f}_n)\}$ where $\overline{v}$'s are sequence or context variables (called *constrained* variables), and $\mathtt{E}$'s are regular expressions such that $\overline{v}_i \notin vars(\mathtt{E}_j)$ for $1 \leq i \leq j \leq n$.

A *regular well-moded context sequence constraint* (RWCS constraint in short) is a finite multiset

$$\{s_1 \approx^? t_1, \ldots, s_n \approx^? t_n,\ (\overline{x}_1 \text{ in } \mathtt{T}_1, \mathtt{f}_1), \ldots, (\overline{x}_m \text{ in } \mathtt{T}_m, \mathtt{f}_m),$$
$$(\overline{C}_1 \text{ in } \mathtt{C}_1, \mathtt{g}_1), \ldots, (\overline{C}_k \text{ in } \mathtt{C}_k, \mathtt{g}_k)\},$$

where $\{s_1 \approx^? t_1, \ldots, s_n \approx^? t_n\}$ is a well-moded CSU problem, $\{(\overline{x}_1$ $\mathtt{in}$ $\mathtt{T}_1, \mathtt{f}_1)$, $\ldots, (\overline{x}_m$ $\mathtt{in}$ $\mathtt{T}_m, \mathtt{f}_m), (\overline{C}_1$ $\mathtt{in}$ $\mathtt{C}_1, \mathtt{g}_1), \ldots, (\overline{C}_k$ $\mathtt{in}$ $\mathtt{C}_k, \mathtt{g}_k)\}$ is a well-moded membership constraint, and the flag for each constrained variable that does not occur in the equations is 0. We use Γ and Δ to denote RWCS constraints. A substitution σ is called a *solution* for such a constraint if $s_i\sigma = t_i\sigma$, $\mathtt{f}_j\sigma \in \{0, 1\}$, $\mathtt{g}_l\sigma \in \{0, 1\}$, $\overline{x}_j\sigma \in \mathcal{L}(\mathtt{T}_j\sigma)_{\mathtt{f}_j\sigma}$, and $\overline{C}_l\sigma \in \mathcal{L}(\mathtt{C}_l\sigma)_{\mathtt{g}_l\sigma}$ for all $1 \leq i \leq n$, $1 \leq j \leq m$, and $1 \leq l \leq k$. We denote the solution set of Γ with $sol(\Gamma)$ and say that Γ is *satisfiable* if $sol(\Gamma) \neq \emptyset$.

We say that a variable is *equational* in an RWCS constraint Γ if it occurs in the equational part of Γ. It is *potentially equational* in Γ if it occurs in a membership constraint of Γ that constrains an equational or a potentially equational variable. Otherwise a variable occurring in Γ is *nonequational* in Γ. We denote the sets of equational, potentially equational, and nonequational variables of Γ respectively with $eqv(\Gamma)$, $peqv(\Gamma)$, and $neqv(\Gamma)$.

In the rest of the paper we will use the symbol $\ll$ for $\approx^?$ to underline that matching techniques will be applied to solve equations.

4 The Framework

Our goal is to develop an algorithm to solve RWCS constraints. It turns out that constrained variable occurrences have a crucial influence on the type and

[4] Note that $(Ts$ $\mathtt{in}$ $\mathtt{T}^*, 1)$ does not have the same meaning as $(Ts$ $\mathtt{in}$ $\ulcorner\mathtt{T}, \mathtt{T}^*\urcorner, 0)$: Just take a^* as $\mathtt{T}$.

decidability of RWCS constraints. We will consider two properties that characterize the constraints: DCV (all constrained variables are distinct) and ECV (each constrained variable occurs in equations, i.e. is equational).

First, we describe a general rule-based solving framework and then show how the algorithm behaves on constraints for which DCV, ECV, or their combinations hold. The most general case is when a RWCS constraint is not restricted with the properties DCV and ECV. However, decidability of such constraints depends on decidability of context unification: If C_1 and C_2 are two contexts that contain context variables and s is a term that does not contain a context variable $\overline{C}$, then to decide whether the constraint $\{s \ll t, (\overline{C} \;\mathsf{in}\; C_1, 0), (\overline{C} \;\mathsf{in}\; C_2, 0)\}$ is solvable, we need to decide whether a context equation $C_1 \approx^? C_2$ has a solution. Decidability of context unification is an open problem [30]. Hence decidability of the following problem is open:

> Given two regular contexts $\mathtt{C}_1$ and $\mathtt{C}_2$, find a substitution σ such that the intersection of two regular tree languages $\mathcal{L}(\mathtt{C}_1\sigma) \cap \mathcal{L}(\mathtt{C}_2\sigma)$ is not empty.

In order to avoid context unification problems, we require RWCS constraints to satisfy a weaker form of ECV: Every variable that is constrained with more that one membership constraint should be an equational variable.

As we will see, weak ECV is indeed enough to have decidable finitary RWCS constraint solving. In the rest of the paper we assume that RWCS constraints satisfy weak ECV and do not mention this explicitly.

Now we present the inference system for our framework. It consists of five groups of rules presented below. Rules operate on systems. A *system* is either the symbol $\bot$ (failure) or a pair $\Gamma;\sigma$, where Γ is a RWCS constraint and σ is a substitution. The indices n and m are non-negative unless otherwise stated.

The first group of rules does not affect membership constraints. It is denoted by $\mathfrak{R}_{\text{eq}}$ and consists of the following 11 rules:

T: **Trivial**

$$\{t \ll t\} \cup \Gamma;\ \sigma \Longrightarrow \Gamma;\ \sigma.$$

IVE: **Individual Variable Elimination**

$$\{x \ll t\} \cup \Gamma;\ \sigma \Longrightarrow \Gamma\vartheta;\ \sigma\vartheta, \qquad \text{where } \vartheta = \{x \mapsto t\}.$$

FVE: **Function Variable Elimination**

$$\{F(s_1, \ldots, s_n) \ll f(t_1, \ldots, t_m)\} \cup \Gamma;\ \sigma \\ \Longrightarrow \{f(s_1\vartheta, \ldots, s_n\vartheta) \ll f(t_1, \ldots, t_m)\} \cup \Gamma\vartheta;\ \sigma\vartheta, \quad \text{where } \vartheta = \{F \mapsto f\}.$$

PD: **Partial Decomposition**

$$\{f(s_1, \ldots, s_n) \ll f(t_1, \ldots, t_m)\} \cup \Gamma;\ \sigma \\ \Longrightarrow \{s_1 \ll t_1, \ldots, s_{k-1} \ll t_{k-1}, f(s_k, \ldots, s_n) \ll f(t_k, \ldots, t_m)\} \cup \Gamma;\ \sigma,$$

if $f(s_1, \ldots, s_n) \neq f(t_1, \ldots, t_m)$, $s_k \in \mathcal{V}_{\text{Seq}}$ for some $1 < k \leq \min(n, m)+1$, and $s_i \notin \mathcal{V}_{\text{Seq}}$ for all $1 \leq i < k$.

TD: **Total Decomposition**

$$\{f(s_1, \ldots, s_n) \ll f(t_1, \ldots, t_n)\} \cup \Gamma;\ \sigma \Longrightarrow \{s_1 \ll t_1, \ldots, s_n \ll t_n\} \cup \Gamma;\ \sigma,$$

if $f(s_1, \ldots, s_n) \neq f(t_1, \ldots, t_n)$ and $s_i \notin \mathcal{V}_{\text{Seq}}$ for all $1 \leq i \leq n$.

SVD: **Sequence Variable Deletion**

$$\{f(\overline{x}, s_1, \ldots, s_n) \ll t\} \cup \Gamma;\ \sigma \Longrightarrow \{f(s_1\vartheta, \ldots, s_n\vartheta) \ll t\} \cup \Gamma\vartheta;\ \sigma\vartheta,$$

where $\vartheta = \{\overline{x} \mapsto \ulcorner\urcorner\}$ and $\overline{x}$ is not constrained in Γ.

W: **Widening**

$$\{f(\overline{x}, s_1, \ldots, s_n) \ll f(t, t_1, \ldots, t_m)\} \cup \Gamma;\ \sigma \Longrightarrow \{f(\overline{x}, s_1\vartheta, \ldots, s_n\vartheta) \ll f(t_1, \ldots, t_m)\} \cup \Gamma\vartheta;\ \sigma\vartheta,$$

where $\vartheta = \{\overline{x} \mapsto \ulcorner t, \overline{x}\urcorner\}$ and $\overline{x}$ is not constrained in Γ.

CVD: **Context Variable Deletion**

$$\{\overline{C}(s) \ll t\} \cup \Gamma;\ \sigma \Longrightarrow \{s\vartheta \ll t\} \cup \Gamma\vartheta;\ \sigma\vartheta,$$

where $\vartheta = \{\overline{C} \mapsto \circ\}$ and $\overline{C}$ is not constrained in Γ.

D: **Deepening**

$$\{\overline{C}(s) \ll f(t_1, \ldots, t_m)\} \cup \Gamma;\ \sigma \Longrightarrow \{\overline{C}(s\vartheta) \ll t_j\} \cup \Gamma\vartheta;\ \sigma\vartheta,$$

where $m > 0$, $\vartheta = \{\overline{C} \mapsto f(t_1, \ldots, t_{j-1}, \overline{C}(\circ), t_{j+1}, \ldots, t_m)\}$ for some $1 \leq j \leq m$, and $\overline{C}$ is not constrained in Γ.

SC: **Symbol Clash**

$$\{f(s_1, \ldots, s_n) \ll g(t_1, \ldots, t_m)\} \cup \Gamma;\ \sigma \Longrightarrow \bot, \qquad \text{if } f \notin \mathcal{V}_{\mathrm{Con}} \cup \mathcal{V}_{\mathrm{Fun}} \text{ and } f \neq g.$$

AD: **Arity Disagreement**

$$\{f(s_1, \ldots, s_n) \ll f(t_1, \ldots, t_m)\} \cup \Gamma;\ \sigma \Longrightarrow \bot,$$

if $m \neq n$ and $s_i \notin \mathcal{V}_{\mathrm{Seq}}$ for all $1 \leq i \leq n$, or $m = 0$ and $s_i \notin \mathcal{V}_{\mathrm{Seq}}$ for some $1 < i \leq n$.

The second group, the $\mathfrak{R}_{\mathrm{rt}}$, consists of 9 rules that transform equations coupled with constraints for regular terms, where the first argument of the left hand side of the selected equation is a sequence variable constrained by a single membership constraint. The symbol `NonEmptySeq` in the rules satisfies the equalities $\texttt{NonEmptySeq}() = 0$ and $\texttt{NonEmptySeq}(r_1, \ldots, r_n) = 1$ if $r_i \notin \mathcal{V}_{\mathrm{Seq}}$ for some $1 \leq i \leq n$, and $\oplus$ is the exclusive or: $0 \oplus 0 = 1 \oplus 1 = 0$ and $1 \oplus 0 = 0 \oplus 1 = 1$.

ESRET: **Empty Sequence as a Regular Term**

$$\{f(\overline{x}, s_1, \ldots, s_n) \ll t,\ (\overline{x} \text{ in } \ulcorner\urcorner, \mathtt{f})\} \cup \Gamma;\ \sigma \Longrightarrow \begin{cases} \{f(\overline{x}, s_1, \ldots, s_n)\vartheta \ll t\} \cup \Gamma\vartheta;\ \sigma\vartheta, \text{ with } \vartheta = \{\overline{x} \mapsto \ulcorner\urcorner\} & \text{if } \mathtt{f} = 0, \\ \bot & \text{if } \mathtt{f} = 1, \end{cases}$$

where $\overline{x}$ is not constrained in Γ.

TRET: **Term as a Regular Term**

$$\{f(\overline{x}, s_1, \ldots, s_n) \ll t,\ (\overline{x} \text{ in } s, \mathtt{f})\} \cup \Gamma;\ \sigma \Longrightarrow \{f(\overline{x}, s_1, \ldots, s_n)\vartheta \ll t\} \cup \Gamma\vartheta;\ \sigma\vartheta,$$

where $\vartheta = \{\overline{x} \mapsto s\}$, $s \notin \mathcal{V}_{\mathrm{Seq}}$, and $\overline{x}$ is not constrained in Γ.

SVRET: **Sequence Variable as a Regular Term**

$$\{f(\overline{x}, s_1, \ldots, s_n) \ll t,\ (\overline{x} \text{ in } \overline{y}, \mathtt{f})\} \cup \Gamma;\ \sigma \Longrightarrow \{f(\overline{x}, s_1, \ldots, s_n)\vartheta \ll t\} \cup \Gamma'\vartheta;\ \sigma\vartheta,$$

where if $\mathtt{f} = 0$ then $\vartheta = \{\overline{x} \mapsto \overline{y}\}$ and $\Gamma' = \Gamma$. If $\mathtt{f} = 1$ and $\overline{y}$ is not constrained in Γ then $\vartheta = \{\overline{x} \mapsto \ulcorner z, \overline{z}\urcorner, \overline{y} \mapsto \ulcorner z, \overline{z}\urcorner\}$ where z and $\overline{z}$ are fresh, and $\Gamma' = \Gamma$. If $\mathtt{f} = 1$ and $\overline{y}$ is constrained in Γ then $\vartheta = \{\overline{x} \mapsto \overline{y}\}$ and Γ' is obtained from Γ by replacing each $(\overline{y}, \mathtt{T}, \mathtt{g})$ with $(\overline{y}, \mathtt{T}, 1)$. The variable $\overline{x}$ is not constrained in Γ.

SVA1: **Sequence Variable Abstraction 1**

$$\{f(\overline{x}, s_1, \ldots, s_n) \ll t,\ (\overline{x} \text{ in } \mathtt{T}, \mathtt{f})\} \cup \Gamma;\ \sigma \\ \Longrightarrow \{f(\overline{x}, s_1, \ldots, s_n) \ll t,\ (\overline{x} \text{ in } \mathtt{T}', \mathtt{f}),\ (\overline{y} \text{ in } \mathtt{T}'', 0)\} \cup \Gamma;\ \sigma,$$

where $\mathtt{T}''$ occurs in $\mathtt{T}$ as a proper r-subterm whose no proper superterm is an r-term and no proper supercontext is an r-context, and $\mathtt{T}'$ is obtained from $\mathtt{T}$ by replacing the occurrence of $\mathtt{T}''$ with a fresh sequence variable $\overline{y}$. $\overline{x}$ is not constrained in Γ.

CVA1: **Context Variable Abstraction 1**

$$\{f(\overline{x}, s_1, \ldots, s_n) \ll t,\ (\overline{x} \text{ in } \mathtt{T}, \mathtt{f})\} \cup \Gamma;\ \sigma \\ \Longrightarrow \{f(\overline{x}, s_1, \ldots, s_n) \ll t,\ (\overline{x} \text{ in } \mathtt{T}', \mathtt{f}),\ (\overline{C} \text{ in } \mathtt{C}, 0)\} \cup \Gamma;\ \sigma,$$

where $\mathtt{C}$ occurs in $\mathtt{T}$ as an r-subcontext whose no proper supercontext is an r-context and no proper superterm is an r-term, and $\mathtt{T}'$ is obtained from $\mathtt{T}$ by replacing the occurrence of $\mathtt{C}$ with a fresh context variable $\overline{C}$. $\overline{x}$ is not constrained in Γ.

ChRET: **Choice as a Regular Term**

$$\{f(\overline{x}, s_1, \ldots, s_n) \ll t,\ (\overline{x} \text{ in } \mathtt{T}_1|\mathtt{T}_2, \mathtt{f})\} \cup \Gamma;\ \sigma \\ \Longrightarrow \{f(\overline{x}, s_1, \ldots, s_n) \ll t,\ (\overline{x} \text{ in } \mathtt{T}_i, \mathtt{f})\} \cup \Gamma;\ \sigma,\ \text{for } i = 1, 2.$$

$\overline{x}$ is not constrained in Γ.

CRET: **Concatenation as a Regular Term**

$$\{f(\overline{x}, s_1, \ldots, s_n) \ll t,\ (\overline{x} \text{ in } \ulcorner \mathtt{T}_1, \mathtt{T}_2 \urcorner, \mathtt{f})\} \cup \Gamma;\ \sigma \\ \Longrightarrow \{f(\overline{x}, s_1, \ldots, s_n)\vartheta \ll t,\ (\overline{y}_1 \text{ in } \mathtt{T}_1, \mathtt{f}_1), (\overline{y}_2 \text{ in } \mathtt{T}_2, \mathtt{f}_2)\} \cup \Gamma\vartheta;\ \sigma\vartheta,$$

where $\overline{y}_1$ and $\overline{y}_2$ are fresh variables, $\vartheta = \{\overline{x} \mapsto \ulcorner \overline{y}_1, \overline{y}_2 \urcorner\}$, and $\mathtt{f}_1$ and $\mathtt{f}_2$ are computed as follows: If $\mathtt{f} = 0$ then $\mathtt{f}_1 = \mathtt{f}_2 = 0$ else $\mathtt{f}_1 = 0$ and $\mathtt{f}_2 = \mathtt{NonEmptySeq}(\overline{y}_1) \oplus 1$. The variable $\overline{x}$ is not constrained in Γ.

RRET1: **Repetition as a Regular Term 1**

$$\{f(\overline{x}, s_1, \ldots, s_n) \ll t,\ (\overline{x} \text{ in } \mathtt{T}^*, 0)\} \cup \Gamma;\ \sigma \Longrightarrow \{f(\overline{x}, s_1, \ldots, s_n)\vartheta \ll t\} \cup \Gamma\vartheta;\ \sigma\vartheta,$$

where $\vartheta = \{\overline{x} \mapsto \ulcorner \urcorner\}$ and $\overline{x}$ is not constrained in Γ.

RRET2: **Repetition as a Regular Term 2**

$$\{f(\overline{x}, s_1, \ldots, s_n) \ll t,\ (\overline{x} \text{ in } \mathtt{T}^*, \mathtt{f})\} \cup \Gamma;\ \sigma \\ \Longrightarrow \{f(\overline{x}, s_1, \ldots, s_n)\vartheta \ll t,\ (\overline{y} \text{ in } \mathtt{T}, 1),\ (\overline{x} \text{ in } \mathtt{T}^*, 0)\} \cup \Gamma\vartheta;\ \sigma\vartheta,$$

where $\overline{y}$ is a fresh variable, $\overline{x}$ is not constrained in Γ, and $\vartheta = \{\overline{x} \mapsto \ulcorner \overline{y}, \overline{x} \urcorner\}$.

The third group of rules operates on equations and constraints on regular contexts. The top context variable of the left hand side of the selected equation is constrained by a single membership constraint. This group is denoted by $\mathfrak{R}_{\mathrm{rc}}$. The symbol `NonEmptyCtx` in the rules satisfies the equalities $\mathtt{NonEmptyCtx}(\circ) = 0$ and $\mathtt{NonEmptyCtx}(C) = 1$ if the context C contains at least one symbol different from context variables and the hole constant, and $\oplus$, as above, is the exclusive or. There are the following 10 rules in this group:

HREC: **Hole as a Regular Context**

$$\{\overline{C}(s) \ll t,\ (\overline{C} \text{ in } \circ, \mathtt{g})\} \cup \Gamma;\ \sigma \\ \Longrightarrow \begin{cases} \{\overline{C}(s)\vartheta \ll t\} \cup \Gamma\vartheta;\ \sigma\vartheta, \text{ with } \vartheta = \{\overline{C} \mapsto \circ\} & \text{if } \mathtt{g} = 0, \\ \bot & \text{if } \mathtt{g} = 1. \end{cases}$$

where $\overline{C}$ is not constrained in Γ.

CxREC1: **Context as a Regular Context 1**

$$\{\overline{C}(s) \ll t,\ (\overline{C}\ \mathsf{in}\ C, \mathtt{g})\} \cup \Gamma;\ \sigma \Longrightarrow \{\overline{C}(s)\vartheta \ll t\} \cup \Gamma\vartheta;\ \sigma\vartheta,$$

where $\vartheta = \{\overline{C} \mapsto C\}$, $C \neq \circ$, and $head(C) \notin \mathcal{V}_{\mathrm{Con}}$. The variable $\overline{C}$ is not constrained in Γ.

CxREC2: **Context as a Regular Context 2**

$$\{\overline{C}(s) \ll t,\ (\overline{C}\ \mathsf{in}\ \overline{D}(C), \mathtt{g})\} \cup \Gamma;\ \sigma \Longrightarrow \{\overline{C}(s) \ll t, (\overline{C}\ \mathsf{in}\ \overline{D}(\circ).C, \mathtt{g})\} \cup \Gamma;\ \sigma,$$

where $C \neq \circ$. The variable $\overline{C}$ is not constrained in Γ.

CVREC: **Context Variable in a Regular Context**

$$\{\overline{C}(s) \ll t,\ (\overline{C}\ \mathsf{in}\ \overline{D}(\circ), \mathtt{g})\} \cup \Gamma;\ \sigma \Longrightarrow \{\overline{C}(s)\vartheta \ll t\} \cup \Gamma'\vartheta;\ \sigma\vartheta,$$

where if $\mathtt{g} = 0$ then $\vartheta = \{\overline{C} \mapsto \overline{D}(\circ)\}$ and $\Gamma' = \Gamma$. If $\mathtt{g} = 1$ and $\overline{D}$ is not constrained in Γ then $\vartheta = \{\overline{C} \mapsto F(\overline{x}, \overline{E}(\circ), \overline{y}), \overline{D} \mapsto F(\overline{x}, \overline{E}(\circ), \overline{y})\}$ where $F, \overline{x}, \overline{E}$, and $\overline{y}$ are fresh, and $\Gamma' = \Gamma$. If $\mathtt{g} = 1$ and $\overline{D}$ is constrained in Γ then $\vartheta = \{\overline{C} \mapsto \overline{D}(\circ)\}$ and Γ' is obtained from Γ by replacing each $(\overline{D}, \mathtt{C}, \mathtt{f})$ with $(\overline{D}, \mathtt{C}, 1)$. The variable $\overline{C}$ is not constrained in Γ.

SVA2: **Sequence Variable Abstraction 2**

$$\{\overline{C}(s) \ll t,\ (\overline{C}\ \mathsf{in}\ \mathtt{C}, \mathtt{g})\} \cup \Gamma;\ \sigma \Longrightarrow \{\overline{C}(s) \ll t,\ (\overline{C}\ \mathsf{in}\ \mathtt{C}', \mathtt{g}),\ (\overline{x}\ \mathsf{in}\ \mathtt{T}, 0)\} \cup \Gamma;\ \sigma,$$

where $\mathtt{T}$ occurs in $\mathtt{C}$ as an r-subterm whose no proper superterm is an r-term and no proper supercontext is an r-cintext, and $\mathtt{C}'$ is obtained from $\mathtt{C}$ by replacing the occurrence of $\mathtt{T}$ with a fresh sequence variable $\overline{x}$. The variable $\overline{C}$ is not constrained in Γ.

CVA2: **Context Variable Abstraction 2**

$$\{\overline{C}(s) \ll t,\ (\overline{C}\ \mathsf{in}\ \mathtt{C}, \mathtt{g})\} \cup \Gamma;\ \sigma \Longrightarrow \{\overline{C}(s) \ll t,\ (\overline{C}\ \mathsf{in}\ \mathtt{C}', \mathtt{g}),\ (\overline{D}\ \mathsf{in}\ \mathtt{C}'', 0)\} \cup \Gamma;\ \sigma,$$

where $\mathtt{C}''$ occurs in $\mathtt{C}$ as an r-subcontext whose no proper supercontext is an r-context and no proper superterm is an r-term, and $\mathtt{C}'$ is obtained from $\mathtt{C}$ by replacing the occurrence of $\mathtt{C}''$ with a fresh context variable $\overline{D}$. The variable $\overline{C}$ is not constrained in Γ.

ChREC: **Choice as a Regular Context**

$$\{\overline{C}(s) \ll t,\ (\overline{C}\ \mathsf{in}\ \mathtt{C}_1 + \mathtt{C}_2, \mathtt{g})\} \cup \Gamma;\ \sigma \Longrightarrow \{\overline{C}(s) \ll t,\ (\overline{C}\ \mathsf{in}\ \mathtt{C}_i, \mathtt{g})\} \cup \Gamma;\ \sigma,$$

for $i = 1, 2$. The variable $\overline{C}$ is not constrained in Γ.

CREC: **Concatenation as a Regular Context**

$$\{\overline{C}(s) \ll t,\ (\overline{C}\ \mathsf{in}\ \mathtt{C}_1.\mathtt{C}_2, \mathtt{g})\} \cup \Gamma;\ \sigma$$
$$\Longrightarrow \{\overline{C}(s)\vartheta \ll t,\ (\overline{D}_1\ \mathsf{in}\ \mathtt{C}_1, \mathtt{g}_1), (\overline{D}_2\ \mathsf{in}\ \mathtt{C}_2, \mathtt{g}_2)\} \cup \Gamma\vartheta;\ \sigma\vartheta,$$

where $\overline{D}_1$ and $\overline{D}_2$ are fresh variables, $\vartheta = \{\overline{C} \mapsto \overline{D}_1(\overline{D}_2(\circ))\}$, and $\mathtt{g}_1$ and $\mathtt{g}_2$ are computed as follows: If $\mathtt{g} = 0$ then $\mathtt{g}_1 = \mathtt{g}_2 = 0$ else $\mathtt{g}_1 = 0$ and $\mathtt{g}_2 = \mathtt{NonEmptyCtx}(\overline{D}_1) \oplus 1$. The variable $\overline{C}$ is not constrained in Γ.

RREC1: **Repetition as a Regular Context 1**

$$\{\overline{C}(s) \ll t,\ (\overline{C}\ \mathsf{in}\ \mathtt{C}^\star, 0)\} \cup \Gamma;\ \sigma \Longrightarrow \{\overline{C}(s)\vartheta \ll t\} \cup \Gamma\vartheta;\ \sigma\vartheta,$$

where $\vartheta = \{\overline{C} \mapsto \circ\}$. The variable $\overline{C}$ is not constrained in Γ.

RREC2: **Repetition as a Regular Context 2**

$\{\overline{C}(s) \ll t,\ (\overline{C}\ \mathsf{in}\ \mathtt{C}^\star, \mathtt{g})\} \cup \Gamma;\ \sigma$
$\quad \Longrightarrow \{\overline{C}(s)\vartheta \ll t,\ (\overline{D}\ \mathsf{in}\ \mathtt{C}, 1),\ (\overline{C}\ \mathsf{in}\ \mathtt{C}^\star, 0)\} \cup \Gamma\vartheta;\ \sigma\vartheta,$

where $\overline{D}$ is a fresh variable, $\overline{C}$ is not constrained in Γ, and $\vartheta = \{\overline{C} \mapsto \overline{D}(\overline{C}(\circ))\}$.

In the fourth group we have only two rules. They deal with the cases when a variable is constrained by more than one membership constraint. This group is denoted by $\mathfrak{R}_{\mathrm{int}}$, underlining the fact that they are related with the intersection of regular languages:

IRET: **Intersection of Regular Terms**

$\{f(\overline{x}, s_1, \ldots, s_n) \ll t,\ (\overline{x}\ \mathsf{in}\ \mathtt{T}_1, \mathtt{f}_1), (\overline{x}\ \mathsf{in}\ \mathtt{T}_2, \mathtt{f}_2)\} \cup \Gamma;\ \sigma$
$\quad \Longrightarrow \{f(\overline{x}, s_1, \ldots, s_n) \ll t,\ (\overline{x}\ \mathsf{in}\ \mathtt{T}_1, \mathtt{f}_1), f(\overline{y}) \ll f(\overline{x}), (\overline{y}\ \mathsf{in}\ \mathtt{T}_2, \mathtt{f}_2)\} \cup \Gamma;\ \sigma,$

where $\overline{y}$ is a fresh variable.

IREC: **Intersection of Regular Contexts**

$\{\overline{C}(s) \ll t,\ (\overline{C}\ \mathsf{in}\ \mathtt{C}_1, \mathtt{g}_1), (\overline{C}\ \mathsf{in}\ \mathtt{C}_2, \mathtt{g}_2)\} \cup \Gamma;\ \sigma$
$\quad \Longrightarrow \{\overline{C}(s) \ll t,\ (\overline{C}\ \mathsf{in}\ \mathtt{C}_1, \mathtt{g}_1), \overline{D}(a) \ll \overline{C}(a), (\overline{D}\ \mathsf{in}\ \mathtt{C}_2, \mathtt{g}_2)\} \cup \Gamma;\ \sigma,$

where $\overline{D}$ is a fresh variable and a is a fresh constant.

The fifth group, $\mathfrak{R}_{\mathrm{sat}}$, consists of a single rule:

SAT: **Satisfiability**

$\Gamma;\ \sigma \Longrightarrow \emptyset;\ \sigma,$

where Γ is a well-moded membership constraint, with all flags 0.

We denote by $\mathfrak{R}$ the set $\mathfrak{R}_{\mathrm{eq}} \cup \mathfrak{R}_{\mathrm{rt}} \cup \mathfrak{R}_{\mathrm{rc}} \cup \mathfrak{R}_{\mathrm{int}} \cup \mathfrak{R}_{\mathrm{sat}}$.

Lemma 1. *Every rule in $\mathfrak{R}$ preserves well-modedness.*

Proof. Let $\Gamma = \{s_1 \ll t_1, \ldots, s_n \ll t_n, (\overline{v}_1\ \mathsf{in}\ \mathtt{E}_1, \mathtt{f}_1), \ldots, (\overline{v}_m\ \mathsf{in}\ \mathtt{E}_m, \mathtt{f}_m)\}$ be a RWCS constraint such that t_1 is ground, $vars(t_j) \subseteq \bigcup_{i=1}^{j-1} vars(s_i)$ for $1 < j \leq n$ and $\overline{v}_i \notin vars(\mathtt{E}_j)$ for $1 \leq i \leq j \leq m$. If there are several equations the ground right hand side we assume that a transformation rule selects always the first one in the given ordering which is the leftmost one.

It is easy to observe that the rules from $\mathfrak{R}_{\mathrm{eq}}$ preserve well-modedness: They either eliminate variables, or keep the set of variables and their occurrences in terms unchanged.

The rules from $\mathfrak{R}_{\mathrm{rt}}$ and $\mathfrak{R}_{\mathrm{rc}}$ either eliminate a variable with a term or a term sequence that does not contain this variable, or replace a regular expression with a fresh variable, or does not change the variable set and the right hand sides of equations at all. It implies that well-modedness is not violated. Just note that in the case of RRET and RREC rules the constraints for fresh variables $\overline{y}$ and $\overline{D}$ will come immediately after the constraints for old variables $\overline{x}$ and $\overline{C}$ in the ordering that verifies well-modedness.

In the intersection rules the new equalities $f(\overline{y}) \ll f(\overline{x})$ and $\overline{D}(a) \ll \overline{C}(a)$ should be immediately after the old ones, $f(\overline{x}, s_1, \ldots, s_n) \ll t$ and $\overline{C}(s) \ll t$ respectively, in the ordering for well-modedness. □

It is obvious that each rule in $\mathfrak{R}$ preserves the weak Ecv property. We assume that from the beginning the flag for each constrained equational variable is set either to 0 or 1. It guarantees that at each step the flag of a selected membership constraint is 0 or 1. We call the substitutions computed at transformation steps (the ϑ's in the rules in $\mathfrak{R}$) the *local* substitutions. We may write $\Gamma_1;\sigma_1 \Longrightarrow_{\mathsf{R},\vartheta} \Gamma_2;\sigma_2$ to indicate that the system $\Gamma_1;\sigma_1$ was transformed into $\Gamma_2;\sigma_2$ by applying the rule $\mathsf{R} \in \mathfrak{R}$ with the local substitution ϑ. A *derivation* is a sequence $\Gamma_1;\sigma_1 \Longrightarrow_{\mathsf{R}_1,\vartheta_1} \Gamma_2;\sigma_2 \Longrightarrow_{\mathsf{R}_2,\vartheta_2} \cdots$ of system transformations. Some of the subscripts will be omitted if they are not relevant. We will sometimes use the abbreviation $\Gamma_1;\sigma_1 \Longrightarrow_{\vartheta}^{+} \Gamma_n;\sigma_n$ for the derivation $\Gamma_1;\sigma_1 \Longrightarrow_{\vartheta_1} \Gamma_2;\sigma_2 \Longrightarrow_{\vartheta_2} \cdots \Longrightarrow_{\vartheta_{n-1}} \Gamma_n;\sigma_n$, where $\vartheta = \vartheta_1 \cdots \vartheta_{n-1}$.

Definition 1. *A constraint solving algorithm $\mathfrak{C}$ is any program that takes a system $\Gamma;\varepsilon$ as input, where Γ is a* Rwcs *constraint, and uses the rules in $\mathfrak{R}$ to generate a complete tree of derivations in the following way:*

1. *The root of the tree is labeled with $\Gamma;\varepsilon$.*
2. *Each branch of the tree is a derivation. The nodes in the tree are systems.*
3. *Each rule selects an equation with the ground right hand side.*[5]
4. *If several rules, or different instances of the same rule are applicable to a node, they are applied concurrently. No rules are applicable to the leaves.*

The leaves of a tree are labeled either with systems of the form $\emptyset;\sigma$ or with $\bot$. The branches that end with $\emptyset;\sigma$ are *successful branches,* and those that end with $\bot$ are *failed branches.* A substitution σ is called an *answer of Γ computed by $\mathfrak{C}$*, or just a *computed answer* of Γ if $\emptyset;\sigma$ is a successful branch of the solving tree for Γ. We denote by $\mathcal{CA}_{\mathfrak{C}}(\Gamma)$ the set of answers of Γ computed by $\mathfrak{C}$. Mostly we will be interested in the set $\mathcal{CA}_{\mathfrak{C}}(\Gamma)|_{vars(\Gamma)}$: the restriction of $\mathcal{CA}_{\mathfrak{C}}(\Gamma)$ to $vars(\Gamma)$.

The following lemma immediately follows from the construction of $\mathfrak{C}$:

Lemma 2. *Let Γ be a* Rwcs *problem. Then for every $\sigma \in \mathcal{CA}_{\mathfrak{C}}(\Gamma)$*

1. *$v\sigma$ is ground for every $v \in eqv(\Gamma)$.*
2. *$v \notin dom(\sigma)$ for every $v \in neqv(\Gamma)$.*

If a potentially equational variable occurs in the domain of a computed answer then its image is also ground. For instance, the constraint $\Gamma = \{f(\overline{x},\overline{y}) \ll f(g(a),a), (\overline{x} \text{ in } g(\overline{z})^*, 0), (\overline{z} \text{ in } \overline{y}, 0)\}$, where $\overline{z} \in peqv(\Gamma)$, has a solution $\{\overline{x} \mapsto \ulcorner\urcorner, \overline{y} \mapsto \ulcorner g(a),a)\urcorner\}$ that leaves $\overline{z}$ unchanged, and has another solution $\{\overline{x} \mapsto g(a), \overline{y} \mapsto a, \overline{z} \mapsto a\}$ which maps $\overline{z}$ to the ground term a.

The flags prevent looping for the cases when regular expressions accept the empty sequence or hole under star. (See also [17].)

[5] Well-modedness guarantees that such an equation exists and that, in fact, the algorithm can use only matching.

Example 5. Consider a derivation of $\{f(\overline{x}) \ll f(b,a), (\overline{x} \ \mathsf{in}\ (a^*|b^*)^*, 0)\}$:

$$\begin{aligned}
&\{f(\overline{x}) \ll f(b,a), (\overline{x} \ \mathsf{in}\ (a^*|b^*)^*, 0)\}; \varepsilon \\
\Longrightarrow\ &\{f(\overline{y},\overline{x}) \ll f(b,a), (\overline{y} \ \mathsf{in}\ (a^*|b^*), 1), (\overline{x} \ \mathsf{in}\ (a^*|b^*)^*, 0)\}; \{\overline{x} \mapsto \ulcorner \overline{y}, \overline{x} \urcorner\} \\
\Longrightarrow\ &\{f(\overline{y},\overline{x}) \ll f(b,a), (\overline{y} \ \mathsf{in}\ a^*, 1), (\overline{x} \ \mathsf{in}\ (a^*|b^*)^*, 0)\}; \{\overline{x} \mapsto \ulcorner \overline{y}, \overline{x} \urcorner\}.
\end{aligned}$$

If we did not have flags, in particular, the flag 1 for $\overline{y}$, here we could replace $\overline{y}$ with $\ulcorner\urcorner$ by the RRET1 rule, obtain the initial problem and, hence, a loop. But the flag 1 prevents this, and the derivation continues with the RRET2 rule:

$$\begin{aligned}
\Longrightarrow\ &\{f(\overline{z},\overline{y},\overline{x}) \ll f(b,a), (\overline{z} \ \mathsf{in}\ a, 1), (\overline{y} \ \mathsf{in}\ a^*, 0), (\overline{x} \ \mathsf{in}\ (a^*|b^*)^*, 0)\}; \\
&\qquad \{\overline{x} \mapsto \ulcorner \overline{z}, \overline{y}, \overline{x} \urcorner, \overline{y} \mapsto \ulcorner \overline{z}, \overline{y} \urcorner\} \\
\Longrightarrow\ &\{f(a,\overline{y},\overline{x}) \ll f(b,a), (\overline{y} \ \mathsf{in}\ a^*, 0), (\overline{x} \ \mathsf{in}\ (a^*|b^*)^*, 0)\}; \\
&\qquad \{\overline{x} \mapsto \ulcorner a, \overline{y}, \overline{x} \urcorner, \overline{y} \mapsto \ulcorner a, \overline{y} \urcorner, \overline{z} \mapsto a\} \\
\Longrightarrow\ &\{a \ll b, f(\overline{y},\overline{x}) \ll f(a), (\overline{y} \ \mathsf{in}\ a^*, 0), (\overline{x} \ \mathsf{in}\ (a^*|b^*)^*, 0)\}; \\
&\qquad \{\overline{x} \mapsto \ulcorner a, \overline{y}, \overline{x} \urcorner, \overline{y} \mapsto \ulcorner a, \overline{y} \urcorner, \overline{z} \mapsto a\} \\
\Longrightarrow\ &\bot.
\end{aligned}$$

4.1 Examples

Table 1 shows a successful derivation for the system $\{\overline{C}(x) \ll f(a),\ \overline{D}(\overline{E}(x)) \ll \overline{C}(g(x)),\ (\overline{D} \ \mathsf{in}\ \overline{C}(\circ).g(\circ), 0)\};\ \varepsilon$. Table 2 shows a successful derivation for $\{\overline{C}(F(\overline{y})) \ll f(b, g(f(a))), G(\overline{x}) \ll F(\overline{y}),\ (\overline{x} \ \mathsf{in}\ f(\circ)^\star(a), 0),\ (\overline{x} \ \mathsf{in}\ f(\overline{z})^*, 0),\ (\overline{u} \ \mathsf{in}\ f(\overline{w}^*), 0)\};\ \varepsilon$ where some simple steps are contracted. Selected equations and membership constraints are framed.

4.2 Soundness of $\mathfrak{C}$

Lemma 3. *If $\Gamma_1; \sigma_1 \Longrightarrow_\vartheta \Gamma_2; \sigma_2$ by a rule in $\mathfrak{R} \setminus \mathfrak{R}_{\mathrm{sat}}$ and φ is a solution of Γ_2 then $\vartheta\varphi$ is a solution of Γ_1.*

Proof. For IREC We have

$$\begin{aligned}
\Gamma_1 =&\{\overline{C}(s) \ll t, (\overline{C} \ \mathsf{in}\ \mathtt{C}_1, \mathtt{g}_1), (\overline{C} \ \mathsf{in}\ \mathtt{C}_2, \mathtt{g}_2)\} \cup \Delta, \quad \vartheta = \varepsilon, \\
\Gamma_2 =&\{\overline{C}(s) \ll t, (\overline{C} \ \mathsf{in}\ \mathtt{C}_1, \mathtt{g}_1), \overline{D}(a) \ll \overline{C}(a), (\overline{D} \ \mathsf{in}\ \mathtt{C}_2, \mathtt{g}_2)\} \cup \Delta,
\end{aligned}$$

where $\overline{D}$ is a fresh variable and a is a fresh constant. Assume φ is a solution of Γ_2. Then $\overline{C}(s)\varphi = t, \overline{C}\varphi \in \mathcal{L}(\mathtt{C}_1\varphi)_{\mathtt{g}_1\varphi}, \overline{D}(a)\varphi = \overline{C}(a)\varphi$, and $\overline{D}\varphi = \mathcal{L}(\mathtt{C}_2\varphi)_{\mathtt{g}_2\varphi}$. Since a is a fresh constant, $\overline{D}(a)\varphi = \overline{C}(a)\varphi$ implies that $\overline{D}\varphi = \overline{C}\varphi$ and, hence, $\overline{C}\varphi = \mathcal{L}(\mathtt{C}_2\varphi)_{\mathtt{g}_2\varphi}$. Therefore, $\varphi = \vartheta\varphi$ is a solution of Γ_1.

For IRET the lemma can be proved in a similar way. For the variable abstraction rules the it immediately follows from the definition of solution and the definitions of languages generated by regular terms and regular contexts. For the other rules in $\mathfrak{R} \setminus \mathfrak{R}_{\mathrm{sat}}$ the lemma follows from Lemma 1 in [21]. □

$$
\begin{aligned}
&\{\boxed{\overline{C}(x) \ll f(a)},\ \overline{D}(\overline{E}(x)) \ll \overline{C}(g(x)),\ (\overline{D}\ \mathsf{in}\ \overline{C}(\circ).g(\circ),0)\};\ \varepsilon\\
\Longrightarrow\ &\{\boxed{x \ll a},\ \overline{D}(\overline{E}(x)) \ll f(g(x)),\ (\overline{D}\ \mathsf{in}\ f(\circ).g(\circ),0)\};\ \{\overline{C} \mapsto f(\circ)\}\\
\Longrightarrow\ &\{\boxed{\overline{D}(\overline{E}(a)) \ll f(g(a))},\ \boxed{(\overline{D}\ \mathsf{in}\ f(\circ).g(\circ),0)}\};\ \{\overline{C} \mapsto f(\circ), x \mapsto a\}\\
\Longrightarrow\ &\{\boxed{\overline{D}_1(\overline{D}_2(\overline{E}(a))) \ll f(g(a))},\ \boxed{(\overline{D}_1\ \mathsf{in}\ f(\circ),0)},\ (\overline{D}_2\ \mathsf{in}\ g(\circ),0)\};\\
&\qquad\{\overline{C} \mapsto f(\circ), x \mapsto a, \overline{D} \mapsto \overline{D}_1(\overline{D}_2(\circ))\}\\
\Longrightarrow\ &\{\boxed{f(\overline{D}_2(\overline{E}(a))) \ll f(g(a))},\ (\overline{D}_2\ \mathsf{in}\ g(\circ),0)\};\\
&\qquad\{\overline{C} \mapsto f(\circ), x \mapsto a, \overline{D} \mapsto f(\overline{D}_2(\circ)), \overline{D}_1 \mapsto f(\circ)\}\\
\Longrightarrow\ &\{\boxed{\overline{D}_2(\overline{E}(a)) \ll g(a)},\ \boxed{(\overline{D}_2\ \mathsf{in}\ g(\circ),0)}\};\\
&\qquad\{\overline{C} \mapsto f(\circ), x \mapsto a, \overline{D} \mapsto f(\overline{D}_2(\circ)), \overline{D}_1 \mapsto f(\circ)\}\\
\Longrightarrow\ &\{\boxed{g(\overline{E}(a)) \ll g(a)}\};\\
&\qquad\{\overline{C} \mapsto f(\circ), x \mapsto a, \overline{D} \mapsto f(g(\circ)), \overline{D}_1 \mapsto f(\circ), \overline{D}_2 \mapsto g(\circ)\}\\
\Longrightarrow\ &\{\boxed{\overline{E}(a) \ll a}\};\\
&\qquad\{\overline{C} \mapsto f(\circ), x \mapsto a, \overline{D} \mapsto f(g(\circ)), \overline{D}_1 \mapsto f(\circ), \overline{D}_2 \mapsto g(\circ)\}\\
\Longrightarrow\ &\{\boxed{a \ll a}\};\\
&\qquad\{\overline{C} \mapsto f(\circ), x \mapsto a, \overline{D} \mapsto f(g(\circ)), \overline{D}_1 \mapsto f(\circ), \overline{D}_2 \mapsto g(\circ), \overline{E} \mapsto \circ\}\\
\Longrightarrow\ &\emptyset;\ \{\overline{C} \mapsto f(\circ), x \mapsto a, \overline{D} \mapsto f(g(\circ)), \overline{D}_1 \mapsto f(\circ), \overline{D}_2 \mapsto g(\circ), \overline{E} \mapsto \circ\}
\end{aligned}
$$

Table 1. A successful derivation of the system $\{\overline{C}(x) \ll f(a),\ \overline{D}(\overline{E}(x)) \ll \overline{C}(g(x)),\ (\overline{D}\ \mathsf{in}\ \overline{C}(\circ).g(\circ),0)\};\ \varepsilon$.

Corollary 1. *If $\Gamma_1;\sigma_1 \Longrightarrow^+_\vartheta \Gamma_2;\sigma_2$ in $\mathfrak{R} \setminus \mathfrak{R}_{\mathrm{sat}}$ and φ is a solution for Γ_2, then $\vartheta\varphi$ is a solution for Γ_1.*

Proof. By induction on the length of the derivation. □

Theorem 1 (Soundness of $\mathfrak{C}$). *Let Γ be a* RWCS *constraint and $\sigma \in \mathcal{CA}_{\mathfrak{C}}(\Gamma)$. Then there exists a solution $\vartheta \in sol(\Gamma)$ such that $\sigma \leq \vartheta$.*

Proof. Let $\Gamma;\varepsilon \Longrightarrow^+_\sigma \emptyset;\sigma$ be a derivation that corresponds to $\sigma \in \mathcal{CA}_{\mathfrak{C}}(\Gamma)$. First assume that $\mathfrak{R}_{\mathrm{sat}}$ is not used in this derivation. Since ε is a solution of $\emptyset$, by Corollary 1, $\varepsilon\sigma = \sigma$ is a solution of Γ. If $\mathfrak{R}_{\mathrm{sat}}$ is used, the derivation has a form $\Gamma;\varepsilon \Longrightarrow^+_\sigma \Delta;\sigma \Longrightarrow_{\mathsf{SAT}} \emptyset;\sigma$. By Corollary 1, if φ is a solution of Δ then $\sigma\varphi$ is a solution of Γ. It remains to show that such a φ exists. Since Δ is well-moded and constrains variables maximum once, there exists $(\overline{v}\ \mathsf{in}\ \mathtt{E},0) \in \Delta$ such that $\mathtt{E}$ does not contain any constrained variables. Since $\mathcal{L}(\mathtt{E})_0$ is nonempty, take an arbitrary element $e \in \mathcal{L}(\mathtt{E})_0$ and define $\varphi_1 = \{\overline{v} \mapsto e\}$. The constraint $(\Delta \setminus \{\overline{v}\ \mathsf{in}\ \mathtt{E},0\})\varphi_1$ has all the properties of Δ, therefore we can construct φ_2 in the similar way, and

$$
\begin{aligned}
&\{\boxed{\overline{C}(F(\overline{y})) \ll f(b, g(f(a)))}, G(\overline{x}) \ll F(\overline{y}),\ (\overline{x}\ \mathsf{in}\ f(\circ)^{\star}(a), 0),\ (\overline{x}\ \mathsf{in}\ f(\overline{z})^{*}, 0),\\
&\qquad (\overline{u}\ \mathsf{in}\ f(\overline{z}^{*}), 0)\};\varepsilon\\
\Longrightarrow\ &\{\boxed{F(\overline{y}) \ll g(f(a))}, G(\overline{x}) \ll F(\overline{y}), (\overline{x}\ \mathsf{in}\ f(\circ)^{\star}(a), 0),\ (\overline{x}\ \mathsf{in}\ f(\overline{z})^{*}, 0),\\
&\qquad (\overline{u}\ \mathsf{in}\ f(\overline{z}^{*}), 0)\};\ \{\overline{C} \mapsto f(b, \circ)\}\\
\Longrightarrow\ \cdots \Longrightarrow\ &\{\boxed{G(\overline{x}) \ll g(f(a))}, (\overline{x}\ \mathsf{in}\ f(\circ)^{\star}(a), 0),\ (\overline{x}\ \mathsf{in}\ f(\overline{z})^{*}, 0),\\
&\qquad (\overline{u}\ \mathsf{in}\ f(\overline{z}^{*}), 0)\};\ \{\overline{C} \mapsto f(b, \circ), F \mapsto g, \overline{y} \mapsto f(a)\}\\
\Longrightarrow\ \cdots \Longrightarrow\ &\{\boxed{g(\overline{x}) \ll g(f(a))}, \boxed{(\overline{x}\ \mathsf{in}\ f(\circ)^{\star}(a), 0)},\ g(\overline{w}) \ll g(\overline{x}),\ (\overline{w}\ \mathsf{in}\ f(\overline{z})^{*}, 0),\\
&\qquad (\overline{u}\ \mathsf{in}\ f(\overline{z}^{*}), 0)\};\ \{\overline{C} \mapsto f(b, \circ), F \mapsto g, \overline{y} \mapsto f(a), G \mapsto g\}\\
\Longrightarrow\ &\{\boxed{g(\overline{x}) \ll g(f(a))}, \boxed{(\overline{x}\ \mathsf{in}\ \overline{D}(a), 0)},\ (\overline{D}\ \mathsf{in}\ f(\circ)^{\star}, 0),\ g(\overline{w}) \ll g(\overline{x}),\\
&\qquad (\overline{w}\ \mathsf{in}\ f(\overline{z})^{*}, 0), (\overline{u}\ \mathsf{in}\ f(\overline{z}^{*}), 0)\};\{\overline{C} \mapsto f(b, \circ), F \mapsto g, \overline{y} \mapsto f(a), G \mapsto g\}\\
\Longrightarrow\ \cdots \Longrightarrow\ &\{\boxed{\overline{D}(a) \ll f(a)}, \boxed{(\overline{D}\ \mathsf{in}\ f(\circ)^{\star}, 0)},\ g(\overline{w}) \ll g(\overline{D}(a)), (\overline{w}\ \mathsf{in}\ f(\overline{z})^{*}, 0),\\
&\qquad (\overline{u}\ \mathsf{in}\ f(\overline{z}^{*}), 0)\};\ \{\overline{C} \mapsto f(b, \circ), F \mapsto g, \overline{y} \mapsto f(a), G \mapsto g, \overline{x} \mapsto \overline{D}(a)\}\\
\Longrightarrow\ \cdots \Longrightarrow\ &\{\boxed{g(\overline{w}) \ll g(f(a))}, \boxed{(\overline{w}\ \mathsf{in}\ f(\overline{z})^{*}, 0)}, (\overline{u}\ \mathsf{in}\ f(\overline{z}^{*}), 0)\};\\
&\qquad \{\overline{C} \mapsto f(b, \circ), F \mapsto g, \overline{y} \mapsto f(a), G \mapsto g, \overline{x} \mapsto f(a), \overline{D}_1 \mapsto f(\circ), \overline{D} \mapsto f(\circ)\}\\
\Longrightarrow\ \cdots \Longrightarrow\ &\{\boxed{(\overline{u}\ \mathsf{in}\ f(a^{*}), 0)}\};\ \{\overline{C} \mapsto f(b, \circ), F \mapsto g, \overline{y} \mapsto f(a), G \mapsto g, \overline{x} \mapsto f(a),\\
&\qquad \overline{D}_1 \mapsto f(\circ), \overline{D} \mapsto f(\circ), \overline{w}_1 \mapsto f(a), \overline{w} \mapsto f(a), \overline{z} \mapsto a\}\\
\Longrightarrow\ &\emptyset;\ \{\overline{C} \mapsto f(b, \circ), F \mapsto g, \overline{y} \mapsto f(a), G \mapsto g, \overline{x} \mapsto f(a),\\
&\qquad \overline{D}_1 \mapsto f(\circ), \overline{D} \mapsto f(\circ), \overline{w}_1 \mapsto f(a), \overline{w} \mapsto f(a), \overline{z} \mapsto a\}.
\end{aligned}
$$

Table 2. A successful derivation for $\{\overline{C}(F(\overline{y})) \ll f(b, g(f(a))),\ G(\overline{x}) \ll F(\overline{y}),\ (\overline{x}\ \mathsf{in}\ f(\circ)^{\star}(a), 0),\ (\overline{x}\ \mathsf{in}\ f(\overline{z})^{*}, 0), (\overline{u}\ \mathsf{in}\ f(\overline{z}^{*}), 0)\};\varepsilon$.

so on. This process ends after n steps, where n is the number of elements in Δ. The substitution $\varphi = \varphi_1 \cdots \varphi_n$ is a solution of Δ. □

4.3 Termination of $\mathfrak{C}$

Proving termination requires to define a complexity measure for RWCS constraints. First we need to introduce auxiliary notions.

The *size* of a pure term t, denoted $tsize(t)$, is defined as the number of symbols in t if t is ground, and ∞ if t contains variables. It is assumed that $\infty > n$ for any nonnegative integer n.

The *regular size* of a regular expression $\mathtt{E}$, denoted $rsize(\mathtt{E})$, is $1 + r_1 + r_2 + r_3 + r_4$, where r_1 is the number of occurrences of regular operators[6] in $\mathtt{E}$, r_2 is the number of context variable applications of the form $\overline{C}(\mathtt{C})$ where $\mathtt{C} \neq \circ$, r_3 is the number of context applications of the form $\mathtt{C}(\mathtt{E}')$ where $\mathtt{C}$ is an r-context, and $r_4 = 1$ if $head(\mathtt{E}) \in \mathcal{V}_{\mathrm{Fun}} \cup \mathcal{F}$ and is 0 otherwise.

Example 6. Regular sizes of some regular expressions:

$$\begin{aligned} rsize(a \mid \ulcorner\urcorner) &= 2 \\ rsize(a \mid f(b)^*) &= 3 \\ rsize(\ulcorner a, b, c\urcorner \mid f(b)^*) &= 5 \\ rsize(\ulcorner a, b, c\urcorner \mid f(a, b)^*) &= 5 \\ rsize(\ulcorner a, b, c\urcorner \mid f((a, b)^*)) &= 7 \\ rsize(\ulcorner(\overline{C}(\overline{D}(\overline{E}(\circ))).f(\overline{x}, \circ))(a), f(\ulcorner a, b\urcorner^*)\urcorner) &= 9 \end{aligned}$$

The *regular size of a regular expression* $\mathtt{E}$ *with respect to a membership constraint* M, denoted $regsize(\mathtt{E}, M)$, is defined as the multiset union $regsize(\mathtt{E}, M) = \{rsize(\mathtt{E})\} \dot\cup R$, where R is itself a multiset union of all $regsize(\mathtt{E}_1, M)$'s such that $(\overline{v} \text{ in } \mathtt{E}_1, \mathtt{f}) \in M$ and $\overline{v} \in vars(\mathtt{E})$.

Example 7. Let $\mathtt{E}$ be a regular expression that is the application of a regular context on a regular term: $f(\overline{x}, \overline{x}^*, \circ)^\star.g(\overline{D}(\circ))\ (f(\overline{x}, g(\overline{C}(a), \overline{C}(f(\overline{x})), f(\overline{y})), \overline{y}))$. Let M be a membership constraint of the form $\{(\overline{x} \text{ in } \overline{y}, 0), (\overline{x} \text{ in } f(a^*), 1), (\overline{C} \text{ in } \overline{D}(\circ)^\star.f(a, \circ), 1), (\overline{C} \text{ in } f(\overline{y}^*, \circ)^\star, 1), (\overline{y} \text{ in } \ulcorner a, b^*\urcorner, 1)\}$. Then $rsize(\mathtt{E}) = 7$ and $regsize(\mathtt{E}, M) = \{7, 1, 3, 3, 3, 3, 3, 3\}$ obtained as follows:

$$\begin{aligned} regsize(\mathtt{E}, M) &= \{rsize(\mathtt{E})\} \dot\cup regsize(\overline{y}, M) \dot\cup regsize(f(a^*), M) \\ &\quad \dot\cup regsize(\overline{D}(\circ)^\star.f(a, \circ), M) \dot\cup regsize(f(\overline{y}^*, \circ)^\star, M) \\ &\quad \dot\cup regsize(\ulcorner a, b^*\urcorner, M) \\ &= \{7\} \dot\cup \{1, 3\} \dot\cup \{3\} \dot\cup \{3\} \dot\cup \{3, 3\} \dot\cup \{3\} \\ &= \{7, 1, 3, 3, 3, 3, 3, 3\}. \end{aligned}$$

A *position* is a sequence of positive integers. For a term t and a position p, $symb(t, p)$ denotes the *symbol of* t *at position* p: $symb(t, \ulcorner\urcorner) = t$ if $t \in \mathcal{V}_{\mathrm{Ind}} \cup \mathcal{V}_{\mathrm{Seq}}$, $symb(t, \ulcorner\urcorner) = head(t)$ if $t \notin \mathcal{V}_{\mathrm{Ind}} \cup \mathcal{V}_{\mathrm{Seq}}$, $symb(\mathsf{h}(t_1, \ldots, t_n), \ulcorner i, i_1, \ldots, i_m\urcorner) = symb(t_i, \ulcorner i_1, \ldots, i_m\urcorner)$ if $1 \leq i \leq n$ and $\mathsf{h} \in \mathcal{F} \cup \mathcal{V}_{\mathrm{Fun}} \cup \mathcal{V}_{\mathrm{Con}}$ ($n = 1$ for $\mathsf{h} \in \mathcal{V}_{\mathrm{Con}}$). In all other cases $symb(t, p)$ is undefined. Positions are ordered with the ordering $\succ$ that is a lexicographic extension of the standard ordering $>$ on positive integers. Moreover, we assume to have a constant ∞ such that $\infty \succ p$ for any position p.

[6] We do not count occurrences of the comma "," between $\mathtt{E}$'s in the expressions of the form $\mathsf{h}(\mathtt{E}_1, \ldots, \mathtt{E}_n)$, where $\mathsf{h} \in \mathcal{V}_{\mathrm{Fun}} \cup \mathcal{F}$, and assume that no $\mathtt{E}_i$ is a concatenation of sequences.

Let t be a pure term and M be a membership constraint. A position p in t is called a *nonzero-constrained position* in t with respect to M if $symb(t,p)$ is a variable constrained by M such that for all $(symb(t,p)\ \mathsf{in}\ \mathtt{E},\mathtt{e}) \in M$ the flag $\mathtt{e} \neq 0$. The minimal element (with respect to $\succ$) of the set of all nonzero-constrained positions in t with respect to M is denoted by $\min_{\succ}(t,M)$. If this set is empty then $\min_{\succ}(t,M) = \infty$.

Example 8. Let t be a term $f(\overline{x},\overline{u},\overline{w},g(\overline{D}(a),\overline{C}(f(\overline{x})),f(\overline{y})),\overline{y})$.

1. $\min_{\succ}(t,M) = 3$ where

$$M = \{(\overline{x}\ \mathsf{in}\ \overline{y},0),(\overline{x}\ \mathsf{in}\ f(a)^*,1),(\overline{u}\ \mathsf{in}\ f(a)^*,0),\\ (\overline{w}\ \mathsf{in}\ \overline{C}(a),\mathtt{NonEmptySeq}(\overline{u})\oplus 1),(\overline{C}\ \mathsf{in}\ \overline{D}(\circ)^\star.f(a,\circ),1),\\ (\overline{C}\ \mathsf{in}\ f(\overline{y}^*,\circ)^\star,1),(\overline{y}\ \mathsf{in}\ \ulcorner a,b^*\urcorner,1)\}.$$

2. $\min_{\succ}(t,M) = \ulcorner 4,2\urcorner$ where

$$M = \{(\overline{x}\ \mathsf{in}\ \overline{y},0),(\overline{x}\ \mathsf{in}\ f(a)^*,1),(\overline{u}\ \mathsf{in}\ f(a)^*,0),(\overline{w}\ \mathsf{in}\ \overline{C}(a),0),\\ (\overline{C}\ \mathsf{in}\ \overline{D}(\circ)^\star.f(a,\circ),1),(\overline{C}\ \mathsf{in}\ f(\overline{y}^*,\circ)^\star,1),(\overline{y}\ \mathsf{in}\ \ulcorner a,b^*\urcorner,1)\}.$$

3. $\min_{\succ}(t,M) = \ulcorner 4,3,1\urcorner$ where

$$M = \{(\overline{x}\ \mathsf{in}\ \overline{y},0),(\overline{x}\ \mathsf{in}\ f(a)^*,1),(\overline{u}\ \mathsf{in}\ f(a)^*,0),(\overline{w}\ \mathsf{in}\ \overline{C}(a),0),\\ (\overline{C}\ \mathsf{in}\ \overline{D}(\circ)^\star.f(a,\circ),0),(\overline{C}\ \mathsf{in}\ f(\overline{y}^*,\circ)^\star,1),(\overline{y}\ \mathsf{in}\ \ulcorner a,b^*\urcorner,1)\}.$$

The *constrained variable prefix* of a term t with respect to a membership constraint M, denoted $cvp(t,M)$, is the multiset of all $regsize(E_{\overline{v}},M)$'s such that $\overline{v}$ is a constrained variable occurring in a position $p \preceq \min_{\succ}(t,M)$, $E_{\overline{v}} = \overline{v}$ if $\overline{v}$ is a sequence variable, and $E_{\overline{v}} = \overline{v}(\circ)$ if $\overline{v}$ is a context variable.

Example 9. Let t be a term $f(\overline{x},\overline{u},\overline{w},g(\overline{D}(a),\overline{C}(f(\overline{x})),f(\overline{y})),\overline{y})$.

1. $cvp(t,M) = \{regsize(\overline{x},M),regsize(\overline{u},M),regsize(\overline{w},M)\} = \{\{1,1,3,3\},\{1,2\},\{1,2,3,3,3\}\}$ where

$$M = \{(\overline{x}\ \mathsf{in}\ \overline{y},0),(\overline{x}\ \mathsf{in}\ f(a^*),1),(\overline{u}\ \mathsf{in}\ f(a)^*,0),\\ (\overline{w}\ \mathsf{in}\ \overline{C}(a),\mathtt{NonEmptySeq}(\overline{u})\oplus 1),(\overline{C}\ \mathsf{in}\ \overline{D}(\circ)^\star.f(a,\circ),1),\\ (\overline{C}\ \mathsf{in}\ f(\overline{y}^*,\circ)^\star,1),(\overline{y}\ \mathsf{in}\ \ulcorner a,b^*\urcorner,1)\}.$$

2. $cvp(t,M) = \{regsize(\overline{x},M),regsize(\overline{u},M)\} = \{\{1,1,3,3\},\{1,1\}\}$ where

$$M = \{(\overline{x}\ \mathsf{in}\ \overline{y},0),(\overline{x}\ \mathsf{in}\ f(a^*),1),(\overline{u}\ \mathsf{in}\ f(a),1),(\overline{v}\ \mathsf{in}\ f(a)^*,0),\\ (\overline{w}\ \mathsf{in}\ \overline{C}(a),\mathtt{NonEmptySeq}(\overline{u})\oplus 1),(\overline{C}\ \mathsf{in}\ \overline{D}(\circ)^\star.f(a,\circ),1),\\ (\overline{C}\ \mathsf{in}\ f(\overline{y}^*,\circ)^\star,1),(\overline{y}\ \mathsf{in}\ \ulcorner a,b^*\urcorner,1)\}.$$

3. $cvp(t, M) = \{regsize(\overline{x}, M),\ regsize(\overline{u}, M),\ regsize(\overline{w}, M),\ regsize(\overline{C}(\circ), M)\}$ $= \{\{1, 1, 3, 3\},\ \{1, 2\},\ \{1, 2, 3, 3, 3\},\ \{1, 3, 3, 3\}\}$ where

$$M = \{(\overline{x}\ \mathsf{in}\ \overline{y}, 0), (\overline{x}\ \mathsf{in}\ f(a^*), 1), (\overline{u}\ \mathsf{in}\ f(a)^*, 0), (\overline{w}\ \mathsf{in}\ \overline{C}(a), 0),$$
$$(\overline{C}\ \mathsf{in}\ \overline{D}(\circ)^\star.f(a, \circ), 1), (\overline{C}\ \mathsf{in}\ f(\overline{y}^*, \circ)^\star, 1), (\overline{y}\ \mathsf{in}\ \ulcorner a, b^* \urcorner, 1)\}.$$

4. $cvp(t, M) = \{regsize(\overline{x}, M),\ regsize(\overline{u}, M),\ regsize(\overline{w}, M),\ regsize(\overline{C}(\circ), M),$ $regsize(\overline{x}, M),\ regsize(\overline{y}, M)\} = \{\{1, 1, 3, 3\},\ \{1, 2\},\ \{1, 2, 3, 3, 3\},\ \{1, 3, 3,\ 3\},$ $\{1, 1, 3, 2\},\ \{1, 3\}\}$ where

$$M = \{(\overline{x}\ \mathsf{in}\ \overline{y}, 0), (\overline{x}\ \mathsf{in}\ f(a^*, 1), (\overline{u}\ \mathsf{in}\ f(a)^*, 0), (\overline{w}\ \mathsf{in}\ \overline{C}(a), 0),$$
$$(\overline{C}\ \mathsf{in}\ \overline{D}(\circ)^\star.f(a, \circ), 0), (\overline{C}\ \mathsf{in}\ f(\overline{y}^*, \circ)^\star, 1), (\overline{y}\ \mathsf{in}\ \ulcorner a, b^* \urcorner, 1)\}.$$

With each RWCS constraint $\Gamma \neq \bot$ we associate a *complexity measure,* $cm(\Gamma)$, as a tuple $\langle n_1, n_2, n_3, n_4, n_5, n_6\rangle$, where n_1, n_5 and n_6 are nonnegative integers, n_2 is a multiset of positive integers and ∞, n_3 is a triple of nonnegative integers, and n_4 is a multiset of multisets of positive integers defined as follows:

$n_1 =$ the number of variables constrained by more than one membership constraint.

$n_2 = \dot\cup_{s \ll t \in \Gamma} \{tsize(t)\}$.

$n_3 = \langle m_1, m_2, m_3\rangle$ where

m_1 is the number of equations $s \ll t \in \Gamma$ such that the head of s is a variable,

m_2 is the number of equations $s \ll t \in \Gamma$ such that the head of s is a context variable,

m_3 is the number of equations $s \ll t \in \Gamma$ such that the first argument of s is a sequence variable.

$n_4 = \dot\cup_{s \ll t \in \Gamma} cvp(s, M)$, where M is the membership constraint part of Γ.

$n_5 =$ the number of distinct variables in Γ.

$n_6 =$ the number of constraints of the form $(\overline{C}\ \mathsf{in}\ \overline{D}(\mathtt{C}), \mathtt{g}),\ \mathtt{C} \neq \circ$ in Γ.

For $\bot$ we define $cm(\bot) = \langle 0, \{0\}\rangle$. The ordering $>$ compares measures lexicographically. Obviously, $>$ is well-founded.

Theorem 2 (Termination of $\mathfrak{C}$). $\mathfrak{C}$ *terminates on any input.*

Proof. Termination of $\mathfrak{C}$ follows from the fact that every rule R in $\mathfrak{R}$ strictly decreases the complexity measure: If $\Gamma_1; \sigma_1 \Longrightarrow_{\mathsf{R}} \Gamma_2; \sigma_2$ then $cm(\Gamma_1) > cm(\Gamma_2)$, and $cm(\Gamma) > cm(\bot)$ for any Γ. Table 4.3 shows which rule in $\mathfrak{R}$ decreases which component of the regular complexity measure. □

Rule	n_1	n_2	n_3	n_4	n_5
T	$=$	$>$			
IVE	$=$	$>$			
FVE	$=$	$\geq$	$=$	$=$	$>$
PD	$=$	$>$			
TD	$=$	$>$			
SVD	$=$	$\geq$	$\geq$	$=$	$>$
W	$=$	$>$			
CVD	$=$	$\geq$	$\geq$	$=$	$>$
D	$=$	$>$			
SC	$\geq$	$>$			
AD	$\geq$	$>$			
ESRET, if $\mathtt{f}=0$	$=$	$\geq$	$\geq$	$>$	
ESRET, if $\mathtt{f}=1$	$\geq$	$>$			
TRET	$=$	$\geq$	$>$		
SVRET, if $\mathtt{f}=0$	$=$	$=$	$=$	$>$	
SVRET, if $\mathtt{f}=1$					
$\overline{y}$ not constrained	$=$	$=$	$>$		
$\overline{y}$ constrained	$=$	$=$	$=$	$>$	
SVA1	$=$	$=$	$=$	$>$	
CVA1	$=$	$=$	$=$	$>$	

Rule	n_1	n_2	n_3	n_4	n_5	n_6
ChRET	$=$	$=$	$=$	$>$		
CRET	$=$	$=$	$=$	$>$		
RRET1	$=$	$\geq$	$\geq$	$>$		
RRET2	$=$	$=$	$=$	$>$		
HREC, if $\mathtt{g}=0$	$=$	$\geq$	$\geq$	$>$		
HREC, if $\mathtt{g}=1$	$\geq$	$>$				
CxREC1	$=$	$\geq$	$>$			
CxREC2	$=$	$=$	$=$	$=$	$=$	$>$
CVREC, if $\mathtt{g}=0$	$=$	$=$	$=$	$>$		
CVREC, if $\mathtt{g}=1$						
$\overline{D}$ not constrained	$=$	$=$	$>$			
$\overline{D}$ constrained	$=$	$=$	$=$	$>$		
SVA2	$=$	$=$	$=$	$>$		
CVA2	$=$	$=$	$=$	$>$		
ChREC	$=$	$=$	$=$	$>$		
CREC	$=$	$=$	$=$	$>$		
RREC1	$=$	$\geq$	$\geq$	$>$		
RREC2	$=$	$=$	$=$	$>$		
IRET, IREC	$>$					
SAT	$=$	$>$				

Table 3. Rules in $\mathfrak{R}$ on the regular complexity measure. The equality sign $=$ means the component remains unchanged, $>$ means it strictly decreases, and $\geq$ means it does not increase.

4.4 Completeness of $\mathfrak{C}$

Theorem 3 (Completeness of $\mathfrak{C}$). *Let Γ be a* RWCS *constraint, ϑ be a solution of Γ, $Q = eqv(\Gamma)$, and $V = vars(\Gamma)$. Then there exists a computed answer $\sigma \in \mathcal{CA}_{\mathfrak{C}}$ such that $\sigma|_Q = \vartheta|_Q$ and $\sigma|_V \leq \vartheta|_V$.*

Proof. We use well-founded induction on complexity measures. Assume that for any RWCS constraint Γ' if $cm(\Gamma) > cm(\Gamma')$ then for any solution ϑ' of Γ' there exists a derivation $\Gamma'; \varepsilon \Longrightarrow^+ \emptyset; \sigma'$ such that $\sigma'|_{Q'} = \vartheta'|_{Q'}$ and $\sigma'|_{V'} \leq \vartheta'|_{V'}$ where $Q' = eqv(\Gamma')$ and $V' = vars(\Gamma')$. We show how to build the desired derivation from $\Gamma; \varepsilon$ for a solution ϑ of Γ.

If Γ does not contain equations then the theorem is trivial, because $\sigma = \varepsilon$ and $eqv(\Gamma) = \emptyset$. Therefore, we assume that Γ contains equations. We pick an arbitrary equation $s \ll t$ from Γ such that t is ground, and represent Γ as $\{s \ll t\} \cup \Delta$. Depending on the form of $s \ll t$ we have three cases: s and t are the same terms, s is an individual variable, or s is a compound term different from t. The first two cases as well as all the subcases of the third one, except the ones considered below, can be handled in the same way as in the proofs of Theorem 3 and Theorem 8 in [21].

We consider here first those subcases of the third case where the first argument of s is a sequence variable constrained by more than one membership constraint, or where the head of s is a context variable constrained by more than one membership constraint. Let $s = f(\overline{x}, s_1, \ldots, s_n)$ and represent Δ as

$\Delta = \{(\overline{x}\ \mathsf{in}\ \mathtt{T}_1, \mathtt{f}_1), (\overline{x}\ \mathsf{in}\ \mathtt{T}_2, \mathtt{f}_2)\} \cup \Phi$. We have $\overline{x}\vartheta = \ulcorner t_1, \ldots, t_m \urcorner$, $m \geq 0$, for some ground t's. We transform $\Gamma; \varepsilon$ with the step $\Gamma; \varepsilon \Longrightarrow_{\mathsf{IRET}} \Psi; \varepsilon$ where $\Psi = \{f(\overline{x}, s_1, \ldots, s_n) \ll t,\ (\overline{x}\ \mathsf{in}\ \mathtt{T}_1, \mathtt{f}_1), f(\overline{y}) \ll f(\overline{x}), (\overline{y}\ \mathsf{in}\ \mathtt{T}_2, \mathtt{f}_2)\} \cup \Phi$. Since ϑ is a solution of Γ, the substitution $\vartheta\psi$ is a solution of Ψ, where $\psi = \{\overline{y} \mapsto \ulcorner t_1, \ldots, t_m \urcorner\}$. We assume without loss of generality that $\overline{y}$ occurs neither in the domain nor in the range of ϑ. By the induction hypothesis, there exists a derivation $\Psi; \varepsilon \Longrightarrow^+ \emptyset; \sigma$ with $\sigma|_{Q'} = \vartheta\psi|_{Q'}$ and $\sigma|_{V'} \leq \vartheta\psi|_{V'}$ where $Q' = \mathit{eqv}(\Psi)$ and $V' = \mathit{vars}(\Psi)$. Since $Q \subset Q'$ and $\overline{y}$ does not occur in ϑ, we have $\sigma|_Q = \vartheta\psi|_Q = \vartheta|_Q$. As for the case with restriction to V, since $\overline{y} \in \mathit{eqv}(\Psi)$, by Lemma 2, $\overline{y}\sigma|_{V'}$ is ground. Then the only possibility is $\overline{y}\sigma|_{V'} = \ulcorner t_1, \ldots, t_m \urcorner$, because $\overline{y}\vartheta\psi|_{V'} = \ulcorner t_1, \ldots, t_m \urcorner$. Moreover, $\overline{y}$ does not occur in the range of $\sigma|_V$. Hence, for all $v \in \mathit{dom}(\sigma|_V)$ we have $v\sigma|_V = v\sigma|_{V'}$. Similarly, since $\overline{y}$ does not occur in ϑ, for all $v \in \mathit{dom}(\vartheta\psi|_V)$ we have $v\vartheta\psi|_V = v\vartheta\psi|_{V'}$. It implies that $\sigma|_V \leq \vartheta\psi|_V$. But since $\vartheta\psi|_V = \vartheta|_V$ we finally get $\sigma|_V \leq \vartheta|_V$.

The case when the head of s is a context variable constrained by more than one regular constraint can be proved in a similar way.

Now assume the first argument of s is a sequence variable constrained only once, by a regular term whose head is not a regular operator, and that contains a proper r-subterm or a proper r-subcontext. Let $s = f(\overline{x}, s_1, \ldots, s_n)$ and represent Δ as $\Delta = \{(\overline{x}\ \mathsf{in}\ \mathtt{T}, \mathtt{f})\} \cup \Phi$. We have $\overline{x}\vartheta = r$, where r is ground. Assume $\mathtt{T}$ contains a proper r-subterm $\mathtt{T}''$ such that no proper superterm of $\mathtt{T}''$ is an r-term and no proper supercontext of $\mathtt{T}''$ is an r-context. We transform $\Gamma; \varepsilon$ with the step $\Gamma; \varepsilon \Longrightarrow_{\mathsf{SVA1}} \Psi; \varepsilon$ where $\Psi = \{f(\overline{x}, s_1, \ldots, s_n) \ll t,\ (\overline{x}\ \mathsf{in}\ \mathtt{T}', \mathtt{f}), (\overline{y}\ \mathsf{in}\ \mathtt{T}'', 0)\} \cup \Phi$, where $\mathtt{T}'$ is obtained from $\mathtt{T}$ by replacing the occurrence of $\mathtt{T}''$ with a fresh sequence variable $\overline{y}$. Since ϑ is a solution of Γ, the substitution $\vartheta\psi$ is a solution of Ψ, where $\psi = \{\overline{y} \mapsto \ulcorner t_1, \ldots, t_m \urcorner\}$ for some ground t's. We assume without loss of generality that $\overline{y}$ occurs neither in the domain nor in the range of ϑ. By the induction hypothesis, there exists a derivation $\Psi; \varepsilon \Longrightarrow^+ \emptyset; \sigma$ with $\sigma|_{Q'} = \vartheta\psi|_{Q'}$ and $\sigma|_{V'} \leq \vartheta\psi|_{V'}$ where $Q' = \mathit{eqv}(\Psi)$ and $V' = \mathit{vars}(\Psi)$. Since $Q = Q'$ and $\overline{y}$ does not occur in ϑ, we have $\sigma|_Q = \vartheta\psi|_Q = \vartheta|_Q$. To prove $\sigma|_V \leq \vartheta|_V$ we first observe that $\overline{y}\sigma|_{V'}$ is ground, although $\overline{y}$ is just a potentially equational variable of Ψ. (It happens because there is no proper r-superterm and r-supercontext of $\mathtt{T}''$ in $\mathtt{T}$ that could disappear (by taking an empty sequence or an empty hole as its instance, or choosing another branch by choice operator) during the derivation. Hence, in the derivation there is no step that turns $\overline{y}$ into a nonequational variable. Therefore, $\overline{y}$ eventually becomes an equational variable and, by Lemma 2, gets bound to a ground sequence of terms.) By the same reasoning as in the case with IRET above we conclude that $\sigma|_V \leq \vartheta|_V$. □

5 Extensions and Restrictions

We can add the intersection "∩" and complementation "¬" into the list of regular operators and redefine the notions of regular expressions and generated languages correspondingly. However, we have to handle membership constraints that contain such operators with some care in order not to have problems with

decidability. The idea is that no rule in the algorithm should make the variables constrained by regular expressions that contain $\cap$ and $\neg$ nonequational. To guarantee this, we require that the RWCS constraints satisfy the following condition (besides already mentioned weak ECV): All the variables constrained by a regular expression that contains $\cap$ or $\neg$ should be equational.

For $\cap$ we introduce the following two rules: IORET in $\mathfrak{R}_{\mathrm{rt}}$ and IOREC in $\mathfrak{R}_{\mathrm{rc}}$. It is not a surprise that they are pretty similar to IRET and IREC that also deal with intersection.

IORET: **Intersection Operator in Regular Terms**

$$\{f(\overline{x}, s_1, \ldots, s_n) \ll t,\ (\overline{x} \text{ in } \mathtt{T}_1 \cap \mathtt{T}_2, \mathtt{f})\} \cup \Gamma;\ \sigma$$
$$\Longrightarrow \{f(\overline{x}, s_1, \ldots, s_n) \ll t,\ (\overline{x} \text{ in } \mathtt{T}_1, \mathtt{f}), f(\overline{y}) \ll f(\overline{x}), (\overline{y} \text{ in } \mathtt{T}_2, \mathtt{f})\} \cup \Gamma;\ \sigma.$$

IOREC: **Intersection Operator in Regular Contexts**

$$\{\overline{C}(s) \ll t,\ (\overline{C} \text{ in } \mathtt{C}_1 \cap \mathtt{C}_2, \mathtt{g})\} \cup \Gamma;\ \sigma$$
$$\Longrightarrow \{\overline{C}(s) \ll t,\ (\overline{C} \text{ in } \mathtt{C}_1, \mathtt{g}), \overline{D}(a) \ll \overline{C}(a), (\overline{D} \text{ in } \mathtt{C}_2, \mathtt{g})\} \cup \Gamma;\ \sigma.$$

As for the complementation, we ignore the constraints of the form $(\overline{x} \text{ in } \neg\mathtt{T}, \mathtt{f})$ and $(\overline{C} \text{ in } \neg\mathtt{C}, \mathtt{g})$ until the algorithm $\mathfrak{C}$ computes an answer σ, and then check whether $\overline{x}\sigma \notin L(\mathtt{T}\sigma)_{\mathtt{f}\sigma}$ and $\overline{C}\sigma \notin L(\mathtt{C}\sigma)_{\mathtt{g}\sigma}$ hold. Since all $\overline{x}$ and $\overline{C}$ are equational, as well as all the variables in $\mathtt{T}$ and $\mathtt{C}$, their instances $\overline{x}\sigma$, $\overline{C}\sigma$, $\mathtt{T}\sigma$, and $\mathtt{C}\sigma$ are all ground, and the membership test can be done using automata.

On the other hand, by restricting variable occurrences we can obtain instances of the framework.

DCV+ECV. The most restricted instance of the framework deals with RWCS problems that satisfy DCV and ECV, i.e. all the constrained variables are distinct and equational.

Example 10. A RWCS constraint that satisfies DCV and ECV:

$$\{\overline{C}(f(\overline{x})) \ll g(f(a,b), h(f(a), f)),\ F(x,y) \ll \overline{C}(b),$$
$$(\overline{C} \text{ in } \overline{D}(h(\overline{y}^*, \circ, \overline{x}^*)), 0),\ (\overline{x} \text{ in } (f(\overline{z}) \mid a)^*, 0)\}.$$

To solve such problems the rules $\mathfrak{R}_{\mathrm{eq}}$, $\mathfrak{R}_{\mathrm{rt}}$, and $\mathfrak{R}_{\mathrm{rc}}$ are sufficient. If we restrict this case further, considering only those regular expressions that are built from pure terms and contexts using regular operators (and forbid constrained variables to occur in them), we obtain the case that was studied in [22]. There it was shown that $\mathfrak{R}_{\mathrm{eq}}$, $\mathfrak{R}_{\mathrm{rt}}$, and $\mathfrak{R}_{\mathrm{rc}}$, without variable abstraction rules, give sound, terminating, and complete solving method.

DCV. Constrained variables, all distinct, need not occur in equations (i.e. can be potentially equational or nonequational).

Example 11. A RWCS constraint that satisfies only DCV:

$$\{\overline{C}(f(\overline{x})) \ll g(f(a,b), h(f(a), f)),\ F(x,y) \ll \overline{C}(b),\ (\overline{C} \text{ in } \overline{D}(h(\overline{y}^*, \circ, \overline{x}^*)), 0),$$
$$(\overline{x} \text{ in } (f(\overline{z}) \mid a)^*, 0),\ (\overline{y} \text{ in } b^*, 0),\ (\overline{w} \text{ in } \ulcorner f(\circ)^\star(a), \overline{y}^* \urcorner, 0)\}.$$

The rules $\mathfrak{R}_{\mathrm{eq}}$, $\mathfrak{R}_{\mathrm{rt}}$, $\mathfrak{R}_{\mathrm{rc}}$, and $\mathfrak{R}_{\mathrm{sat}}$ provide sound, terminating, and complete solving method for this case.

ECV. Constrained variables need not all be distinct, but they must occur in equations (i.e. must be equational variables).

Example 12. A RWCS constraint that satisfies only ECV:

$$\{\overline{C}(f(\overline{x})) \ll g(f(a,b),h(f(a),f)),\ F(x,y) \ll \overline{C}(b),$$
$$(\overline{C}\ \mathsf{in}\ \overline{D}(h(\overline{y}^*,\circ,\overline{x}^*)),0),\ (\overline{x}\ \mathsf{in}\ (f(\overline{z})\mid a)^*,0),\ (\overline{x}\ \mathsf{in}\ f(\overline{z})^*\mid a,0)\}.$$

In this case for sound, terminating, and complete solving method we need the rules $\mathfrak{R}_{\mathrm{eq}}$, $\mathfrak{R}_{\mathrm{rt}}$, $\mathfrak{R}_{\mathrm{rc}}$, and $\mathfrak{R}_{\mathrm{int}}$.

6 Related Work and Concluding Remarks

The framework is flexible and expressive: It allows to traverse the data term, represented as a tree, in horizontal and in vertical directions (using four different kinds of variables). Variables can be constrained by regular expressions, and the regular expressions can occur in the query terms.[7] The same variable can be constrained in several ways. Moreover, the regular expressions the framework processes can be seen as an "extension" of the standard ones [13] with (unrestricted) variable occurrences. Well-modedness allows avoiding variable occurrence check and guarantees that matching techniques can be applied for solving.

Our framework subsumes various formalisms used in constraint solving, programming, and querying. Here we just mention the most closely related ones. Note that the types of problems we can solve are problematic for automata-based approaches because of variables in regular expressions.

Context matching [33] is a particular instance of RWCS constraint solving where only context and individual variables are allowed, no regular expressions are considered, and each equation to be solved is a matching equation.

Context sequence matching [22] can be obtained by the following restrictions: Regular terms and regular contexts should be built by regular operators from pure terms and pure contexts only; all the equations to be solved should be matching equations (and should not involve regular operators); each sequence and context variable can be constrained by maximum one membership constraint (DCV); all constrained variables should occur in matching equations (ECV); no constrained variable can occur in regular expressions.

[7] Query terms with regular expressions can be reduced to query terms without regular expressions and with membership constraints, by variable abstraction techniques. We did not spell it in this paper, but it is straightforward. For instance, $\{f(a^*,g(\circ)\mid \overline{C}(f(a\mid b,\circ))(f(b^*)))\ll t\}$ is transformed into $\{f(\overline{x},\overline{D}(f(\overline{y})))\ll t,(\overline{x}\ \mathsf{in}\ a^*,0),(\overline{D}\ \mathsf{in}\ g(\circ)\mid\overline{C}(f(a\mid b,\circ)),0),(\overline{y}\ \mathsf{in}\ b^*,0)\}$.

Example 13. Let $\Gamma = \{\overline{C}(b) \ll f(a, a, f(a, f(b))), (\overline{C} \text{ in } f(a^*, \circ)^\star, 0)\}$ be a RWCS constraint. The algorithm $\mathcal{C}$ solves it with $\{\overline{C} \mapsto f(a, a, f(a, f(\circ)))\}$. This constraint can not be expressed in the syntax of [22]. An attempt of writing Γ in the form $\Delta = \{\overline{C}(b) \ll f(a, a, f(a, f(b))), (\overline{C} \text{ in } f(\overline{x}, \circ)^\star, 0), (\overline{x} \text{ in } a^*, 0)\}$ does not give an equivalent transformation: Δ is not solvable, because $\overline{x}$ will get instantiated with $\ulcorner a, a \urcorner$ and will not match a.

The pattern matching mechanism of the programming language of MATHEMATICA [35] can be directly modeled in our framework. For this we do not even need context variables. The pattern constructs like `RepeatedNull`, `Repeated`, `Alternatives`, `Optional`, `Except` have straightforward counterparts in our regular expressions. The "shortest first match" semantics of MATHEMATICA for sequence variables [5] can be captured by first trying those rules from $\mathfrak{R}$ that assign $\ulcorner\urcorner$ to sequence variables, and stopping the algorithm $\mathfrak{C}$ when the first matcher is computed.

A rule-based system ρLOG [25], implements matching with individual, function, sequence and context variables that is subsumed by RWCS constraint solving. Various special versions of the framework found their way into the mathematical software system THEOREMA [6].

The equational formulae with membership constraints [14] have similarities and differences with RWCS constraints. The fragment that can be represented as an instance of our framework consists of existentially closed constraints in disjunctive normal form, where disjuncts contain no negated equality and are well-moded. To express this fragment we need to consider only individual and sequence variables and ground regular terms without star in membership constraints.

Comon [12] restricts equational and membership constraints in a different way than we do: He requires any occurrence of the same context variable to be always applied to the same term (that gives a decidable fragment), while we require well-modedness. Otherwise, his sort expressions correspond to our regular terms without variables and the star operator. Context expressions correspond to regular contexts without sequence and function variables. Sequence and function variables do not occur in the equations either.

Like [12], we can easily express possible schematization involving a regular expression of star height larger than 2 like, for instance, writing $(f(\circ)^\star.g(\circ))^\star(a)$ for $f^{n_1}(g(f^{n_2}(g(\cdots f^{n_2}(g(a))\cdots))))$. This can not be represented in the formalisms developed for unification of terms schemes [8, 32, 11, 18]. On the other hand, these formalisms can represent the set of all ground terms of the form $f(g^n(a), h^n(a))$ which can not be expressed in our framework.

We can encode one-step rewrite constraints [7] $s_1 \to t_1$ by $l_1 \to r_i, \ldots, s_n \to t_n$ by $l_n \to r_n$ (that says that the term s_i can be rewritten to a term t_i by the rule $l_i \to r_i$ in one step) as $\{\overline{C}_1(l_1) \ll s_1, t_1 \ll \overline{C}_1(r_1), \ldots, \overline{C}_n(l_n) \ll s_n, t_n \ll \overline{C}_n(r_n)\}$ provided that $vars(r_i) \subseteq vars(l_i)$ for each i, variables in rules and terms are disjoint, s_1 is ground and $vars(s_i) \subseteq \bigcup_{j=1}^{i-1} vars(t_j)$. These conditions make sure that the obtained constraint is well-moded. Niehren et al [29] generalize one-step rewriting constraints by specifying the position where the term is to

be rewritten and impose the ordering constraints on positions. Under the well-modedness restrictions we can express not only this generalization but any other that specify positions in terms of regular contexts, and also one-step rewrite constraints for rewrite systems that contain not only individual but also sequence and context variables. It implies that well-moded one-step rewrite constraints with such extended rewrite rules are decidable. Moreover, again under well-modedness restrictions equality up-to constraints [27] (that subsume one-step rewrite constraints) can be expressed.

Regular expression pattern matching [19] is used for tree manipulation, primarily for XML, in a statically typed setting. Regular expression patterns basically correspond to our regular terms without individual variables and contexts. The effect of regular contexts is achieved by recursion on pattern names (under certain restrictions that guarantee that the language remains regular). Regular expression patterns are restricted to be linear. We do not have such a restriction.

Niehren et al [28] use tree automata for multi-slot information extraction from semistructured data. The automata are restricted to be unambiguous that limits n-ary queries to finite unions of Cartesian closed queries (Cartesian products of monadic queries), but this restricted case is processed efficiently. RWCS constraint solving is closely related with some other solving methods proposed for querying and transforming semistructured data (XML, in particular), like simulation unification [4] of XCERPT, unification with sequence variables [20] of XCENTRIC [10], path expression matching of XPATH [9], matching of incomplete regular expressions [31], just to name a few.

We can also extend the framework to work on multitrees [23] that are unranked unordered trees. The property of being unordered can be expressed by the equality $f(\overline{x}, x, \overline{y}, y, \overline{z}) = f(\overline{x}, y, \overline{y}, x, \overline{z})$ and the corresponding rule can be directly incorporated into the inference system of our algorithm. In this way we, in fact, get a mixture of constraints over ordered and unordered unranked trees. However, a naive straightforward way of adding a rule for unordered terms would be very inefficient since it would consider all possible permutations of the arguments. There are known techniques for efficient commutative and associative-commutative matching (see, e.g. [16]) that can be adapted for this case.

We can use well-moded membership constraints to express regular hedge grammars (without multiple recursion). For instance, let G be the following regular hedge grammar (N, T, S, P) from [26]:

$$
\begin{aligned}
N &= \{\mathit{Doc}, \mathit{Para1}, \mathit{Para2}, \mathit{Pcdata}\} \\
T &= \{\mathbf{doc}, \mathbf{para}, \mathbf{pcdata}\} \\
S &= \{(\mathbf{doc}, \mathit{Doc})\} \\
P &= \{\mathit{Doc} \to (\mathbf{para}[\mathit{Para1}], \mathbf{para}[\mathit{Para2}]^*), \\
&\qquad \mathit{Para1} \to \epsilon, \mathit{Para2} \to \mathbf{pcdata}[\mathit{Pcdata}], \mathit{Pcdata} \to \epsilon\}
\end{aligned}
$$

It has a straightforward translation into the following well-moded membership constraint (overlined identifiers are sequence variables and bold face ones are

function symbols):

$$\{(\overline{Doc} \text{ in } \mathbf{doc}(\mathbf{para}(\overline{Para1}), \mathbf{para}(\overline{Para2})^*), 0), (\overline{Para1} \text{ in } \ulcorner\urcorner, 0),$$
$$(\overline{Para2} \text{ in } \mathbf{pcdata}(\overline{Pcdata}), 0), (\overline{Pcdata} \text{ in } \ulcorner\urcorner, 0)\}.$$

In fact, this constraint can be further simplified:

$$\{(\overline{Doc} \text{ in } \mathbf{doc}(\mathbf{para}(), \mathbf{para}(\mathbf{pcdata}())^*), 0)\}.$$

We can easily model single recursion in regular grammars with well-moded membership constraint. Mutual recursion can not be modeled because of well-modedness restriction, and multiple recursion would need multiple occurrences of the hole constant.

Nevertheless, in our opinion, having RWCS constraint solving as the matching mechanism for an XML querying and transformation language would make it very flexible and expressive. In [22] we demonstrated how an instance of RWCS constraint solving, context sequence matching, can be used in this purpose.

Other application areas of our framework are implementation of rewriting and rewriting strategies, and representing and matching schemas in program synthesis.

An experimental PROLOG implementation of our algorithm is available at: `http://www.risc.uni-linz.ac.at/people/tkutsia/software.html`.

References

1. K. R. Apt and S. Etalle. On the unification free PROLOG programs. In A. M. Borzyszkowski and S. Sokolowski, editors, *Proc. of* MFCS'93, volume 711 of LNCS, pages 1–19. Springer, 1993.
2. I. Attali and P. Franchi-Zannettacci. Unification-free execution of TYPOL programs by semantic attribute evaluation. In R. A. Kowalski and K. A. Bowen, editors, *Proc. of 5th* ICLP/SLP, pages 160–177. MIT Press, 1988.
3. A. Brüggemann-Klein, M. Murata, and D. Wood. Regular tree and regular hedge languages over unranked alphabets. Technical Report HKUST-TCSC-2001-05, Hong Kong University of Science and Technology, 2001.
4. F. Bry and S. Schaffert. Towards a declarative query and transformation language for XML and semistructured data: Simulation unification. In *Proc. of* ICLP, number 2401 in LNCS. Springer, 2002.
5. B. Buchberger. MATHEMATICA as a rewrite language. In T. Ida, A. Ohori, and M. Takeichi, editors, *Proc. of the 2nd Fuji Int. Workshop on Functional and Logic Programming*, pages 1–13, Shonan Village Center, Japan, 1996. World Scientific.
6. B. Buchberger, A. Crăciun, T. Jebelean, L. Kovács, T. Kutsia, K. Nakagawa, F. Piroi, N. Popov, J. Robu, M. Rosenkranz, and W. Windsteiger. THEOREMA: Towards computer-aided mathematical theory exploration. *J. Applied Logic*, 2006. To appear.
7. A.-C. Caron, J.-L. Coquidé, and M. Dauchet. Encompassment properties and automata with constraints. In C. Kirchner, editor, *Proc. of* RTA'93, volume 690 of LNCS, pages 328–342. Springer, 1993.

8. H. Chen and J. Hsiang. Logic programming with recurrence domains. In J. L. Albert, B. Monien, and M. Rodríguez-Artalejo, editors, *Proc. of* ICALP'91, volume 510 of LNCS, pages 20–34. Springer, 1991.
9. J. Clark and S. DeRose, editors. *XML Path Language (XPath) Version 1.0*. W3C, 1999. Available from: `http://www.w3.org/TR/xpath/`.
10. J. Coelho and M. Florido. XCENTRIC: A logic programming language for XML. Technical Report DCC-2005-X, DCC-FC and LIACC, University of Porto, 2005.
11. H. Comon. On unification of terms with integer exponents. *Mathematical Systems Theory*, 28(1):67–88, 1995.
12. H. Comon. Completion of rewrite systems with membership constraints. Part II: Constraint solving. *J. Symb. Comp.*, 25(4):421–453, 1998.
13. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available from: `http://www.grappa.univ-lille3.fr/tata`, 1997.
14. H. Comon and C. Delor. Equational formulae with membership constraints. *Inf. Comput.*, 112(2):167–216, 1994.
15. P. Deransart and J. Maluszynski. Relating logic programs and attribute grammars. *J. Log. Program.*, 2(2):119–155, 1985.
16. S. Eker. Fast matching in combinations of regular equational theories. *Electronic Notes in Theoretical Computer Science*, 4, 1996.
17. A. Frisch and L. Cardelli. Greedy regular expression matching. In *Proc. of* ICALP'04, pages 618–629, 2004.
18. M. Hermann and R. Galbavý. Unification of infinite sets of terms schematized by primal grammars. *Theor. Comput. Sci.*, 176(1–2):111–158, 1997.
19. H. Hosoya and B. Pierce. Regular expression pattern matching for XML. *J. Functional Programming*, 13(6):961–1004, 2003.
20. T. Kutsia. Unification with sequence variables and flexible arity symbols and its extension with pattern-terms. In J. Calmet, B. Benhamou, O. Caprotti, L. Henocque, and V. Sorge, editors, *Proc. of Joint* AISC'2002—CALCULEMUS'2002 *Conference*, volume 2385 of LNAI, pages 290–304. Springer, 2002.
21. T. Kutsia and M. Marin. Matching with regular constraints. Technical Report 05-05, RISC, Johannes Kepler University, Linz, 2005.
22. T. Kutsia and M. Marin. Matching with regular constraints. In G. Sutcliffe and A. Voronkov, editors, *Proc. of* LPAR'05, volume 3835 of LNAI, pages 215–229. Springer, 2005.
23. D. Lugiez. Multitree automata that count. *Theor. Comput. Sci.*, 333(1–2):225–263, 2005.
24. J. Maluszynski and H. Komorowski. Unification-free execution of logic programs. In *Proc. of* SLP, pages 78–86, 1985.
25. M. Marin. ρLog. `http://www.score.is.tsukuba.ac.jp/~mmarin/RhoLog/`, 2005.
26. M. Murata, D. Lee, and M. Mani. Taxonomy of XML schema languages using formal language theory. In *Extreme Markup Languages*, 2001.
27. J. Niehren, M. Pinkal, and P. Ruhrberg. On equality up-to constraints over finite trees, context unification, and one-step rewriting. In W. McCune, editor, *Proc. of* CADE-14, volume 1249 of LNCS, pages 34–48. Springer, 1997.
28. J. Niehren, L. Planque, J.-M. Talbot, and S. Tison. N-ary queries by tree automata. In *Proc. of* DBPL*'05*, 2005.
29. J. Niehren, S. Tison, and R. Treinen. On rewrite constraints and context unification. *Inf. Process. Lett.*, 74(1-2):35–40, 2000.
30. The RTA List of Open Problems. Problem #90. Available from the Web: `http://www.lsv.ens-cachan.fr/rtaloop/problems/90.html`.

31. S. Okui and T. Suzuki. Pattern matching incompletely RE-typed expressions via transformations. IPSJ *Transactions on Programming*, 47(SIG 0(PRO 29)), 2006. To appear.
32. G. Salzer. The unification of infinite sets of terms and its applications. In A. Voronkov, editor, *Logic Programming and Automated Reasoning, Proc. of* LPAR'92, volume 624 of LNCS, pages 409–420. Springer, 1992.
33. M. Schmidt-Schauß and J. Stuber. On the complexity of linear and stratified context matching problems. *Theory Comput. Systems*, 37:717–740, 2004.
34. F. van Raamsdonk. Translating logic programs into conditional rewriting systems. In L. Naish, editor, *Proc. of 14th* ICLP, pages 168–182. MIT Press, 1997.
35. S. Wolfram. *The* MATHEMATICA *Book*. Wolfram Media, 5th edition, 2003.