Solving Regular Constraints for Hedges and Contexts

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 an algorithm for constraint solving over hedges and contexts built over individual, sequence, function, and context variables and flexible arity symbols, where the admissible bindings of sequence variables and context variables can be constrained to languages represented by regular hedge or regular context expressions. We identify sufficient syntactic restrictions that enable to solve such constraints by matching techniques, and describe a solving algorithm that is sound and complete.

1 Introduction

This paper extends our previous work on regular constraint solving [21] by identifying a class of more general regular constraints. The constraints discussed in this paper consist of two parts: (1) the equational part, which is a finite multiset of equations between hedges, and (2) the membership part, which is a finite multiset of constraints for context variables and sequence variables that may be occur in the hedges of the equational part. Membership constraints constrain sequence variables with regular hedge expressions, and context variables with regular context expressions (that are a special kind of regular tree expressions).

Compared to [21], there are several novelties in the current work. First, we allow regular operators to occur in the arguments of regular expressions, whereas in our previous work they were allowed to appear on the top level only. This extension brings complete power of regular hedge and context expressions. For instance, now we can consider regular expressions like $f(x^*|(a,b^*))|g(x)^*$, where "$,$", "$|$", and "*" are regular hedge operators for concatenation, choice, and repetition, respectively.

⋆ 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).

Second, the equational parts of the constraints now are unification problems instead of matching ones. However, unification with sequence, and especially with context variables is a quite hard problem: They may have infinite minimal complete set of unifiers, and decidability of context unification is still an open problem. (For more details, see, e.g., [3, 15, 18–20] on sequence unification and [10, 22, 23, 32, 33, 36] on context unification.) Therefore, we consider only well-moded problems, for which unification can be replaced by iterated matching. Such replacement techniques have been studied by number of authors in different contexts; see, e.g., [12, 25, 2, 1, 35]. Having well-moded unification instead of pure matching looks quite an obvious extension, but it has deeper consequences. For instance, it allows us to consider membership constraints that constrain the same variable with different regular expressions (intersection of regular languages): Such constraints can be reduced to well-moded problems where each variable is constrained only once.

Third, contexts can be applied to hedges (instead of to single terms only). It means that in the context the hole can be replaced with a sequence of terms.

All these extensions increase the expressive power, and allow us to consider regular hedge and context constraints in the same framework. We call the class of constraints described so far regular well-moded constraints, or RWMC in short.

One should note that we use the term "regular expression" in a somewhat more liberal way than the standard definition of regular tree expressions [11] and regular hedge expressions [4], because in our regular expressions variables may occur. The constraint solving algorithm described in this paper finds substitutions for those variables as well.

Another remark should be made about regular context expressions: They in some sense combine features of regular hedge and tree expressions. Languages generated by regular context expressions are sets of contexts, that are hedges with exactly one hole.

Solving an RWMC 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 RWMC's. The flexibility and expressiveness of the approach suggest a broad range of possible applications. We briefly discuss some of them, related to constraints, 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 algorithm. Sect. 5 discusses related work and concludes the paper. To make paper self-contained, we put proofs in the appendix.

2 Preliminaries

We consider an 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. 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 and *hedges* are constructed using the following grammar where t stands for terms, he stands for hedge elements, and $\tilde{t}$ stands for hedges:

$$t ::= x \mid \circ \mid f(\tilde{t}) \mid F(\tilde{t}) \qquad he ::= \overline{x} \mid t \mid \overline{C}(\tilde{t}) \qquad \tilde{t} ::= he_1, \ldots, he_n$$

where $n \geq 0$ is the length of the hedge. For readability, sometimes we use parentheses around hedges, like $(x, \overline{x}, \overline{C}())$. We will write a for the term $a()$ where $a \in \mathcal{F}$. Terms are denoted with s, t, r. Hedges are denoted with $\tilde{s}, \tilde{t}, \tilde{r}$. A *context* is a hedge with a single occurrence of the hole constant $\circ$. A context C may be applied to a hedge $\tilde{t}$, written $C[\tilde{t}]$, and the result is the hedge consisting of C with the occurrence of $\circ$ replaced by the hedge $\tilde{t}$. We use C and D for contexts.

A *substitution* is a mapping from individual variables to hole-free terms, from sequence variables to hole-free hedges, from function variables to function variables and symbols, and from context variables to contexts, such that all but finitely many individual, sequence, and function variables are mapped to themselves, and all but finitely many context variables are mapped to themselves applied to the hole. For example, the mapping $\{x \mapsto f(a, \overline{y}), \overline{x} \mapsto (), \overline{y} \mapsto (a, \overline{C}(f(b)), x), F \mapsto g, \overline{C} \mapsto g(\circ)\}$ is a substitution. 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 $dom(\sigma) = \{v \in \mathcal{V}_{\text{Ind}} \cup \mathcal{V}_{\text{Seq}} \cup \mathcal{V}_{\text{Fun}} \mid v\sigma \neq v\} \cup \{\overline{C} \in \mathcal{V}_{\text{Con}} \mid \overline{C}\sigma \neq \overline{C}(\circ)\}$. Substitutions are extended to terms and hedges:

$$\begin{aligned} x\sigma &= \sigma(x). & \overline{x}\sigma &= \sigma(\overline{x}). \\ \circ\sigma &= \circ. & ()\sigma &= (). \\ f(\tilde{t})\sigma &= f(\tilde{t}\sigma). & \overline{C}(\tilde{t})\sigma &= \sigma(\overline{C})[\tilde{t}\sigma]. \\ F(\tilde{t})\sigma &= \sigma(F)(\tilde{t}\sigma). & (he_1, \ldots, he_n)\sigma &= (he_1\sigma, \ldots, he_n\sigma). \end{aligned}$$

For example, the instance of the hedge $(f(\overline{C}(a, \overline{x}, b), x), F(\overline{y}), \overline{x})$ under the substitution $\sigma = \{x \mapsto f(a, \overline{y}), \overline{x} \mapsto (), \overline{y} \mapsto (a, \overline{C}(f(b)), x), F \mapsto g, \overline{C} \mapsto \circ\}$ is the hedge $(f(a, b, f(a, \overline{y})), g(a, \overline{C}(f(b)), x))$.

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 *hedge equation* is a pair of hedges $\langle \tilde{s}, \tilde{t} \rangle$, written $\tilde{s} \approx \tilde{t}$. Substitutions are extended to equations in the usual way. The fact that the equation $\tilde{s} \approx \tilde{t}$ has to be solved is written as $\tilde{s} \approx^? \tilde{t}$. An *equational constraint* is a finite multiset of hedge equations. A substitution σ is called a *solution* of an equational constraint $\{\tilde{s}_1 \approx^? \tilde{t}_1, \ldots, \tilde{s}_n \approx^? \tilde{t}_n\}$ iff $\{\tilde{s}_1\sigma = \tilde{t}_1\sigma, \ldots, \tilde{s}_n\sigma = \tilde{t}_n\sigma\}$.

The set of variables of a syntactic object O is denoted by $vars(O)$. An equational constraint is *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 $1 < j \leq n$.

3 Regular Constraints

We introduce regular expressions for hole-free hedges and contexts. We write `()` for the empty hedge regular expression, "`,`" for hedge concatenation, "`.`" for context concatenation, "`|`" for hedge choice, "`+`" for context choice, "`*`" for hedge repetition, and "`⋆`" for context repetition. A *regular expression* `Re` is a common name for a *regular hedge expression* (for hole-free hedges) `Rh` and a *regular context expression* `Rc`. They are defined by the following grammars: (The metavariable `h` ranges over function symbols and function and context variables.)

$$\begin{aligned}
\texttt{Re} &::= \texttt{Rh} \mid \texttt{Rc} \\
\texttt{Rh} &::= x \mid \overline{x} \mid \texttt{h(Rh)} \mid \texttt{()} \mid \texttt{Rh,Rh} \mid \texttt{Rh|Rh} \mid \texttt{Rh}^{*} \mid \texttt{Rc(Rh)}. \\
\texttt{Rc} &::= \circ \mid \texttt{Rh,Rc} \mid \texttt{Rc,Rh} \mid \texttt{h(Rc)} \mid \texttt{Rc.Rc} \mid \texttt{Rc+Rc} \mid \texttt{Rc}^{\star}.
\end{aligned}$$

Example 1. $\overline{C}(F(f(\overline{x})^{*}\texttt{,}g(\circ)\texttt{.}h(\circ)))\texttt{+}f((g(x)\texttt{,}x)^{*}\texttt{,}g(\circ))^{\star}$ is a regular context expression and $(\overline{C}(F(f(\overline{x})^{*}\texttt{,}g(\circ)\texttt{.}h(\circ)))\texttt{+}f((g(x)\texttt{,}x)^{*}\texttt{,}g(\circ))^{\star})\,(f(\overline{x}\texttt{,}a^{*}))$ is a regular hedge expression.

Now we extend substitutions to regular expressions. For a regular expression `Re` and a substitution σ, the regular expression $\texttt{Re}\sigma$ is obtained from `Re` by performing the following two steps:

1. Let σ_r be the mapping $\{v \mapsto r(\sigma(v)) \mid v \in \mathit{dom}(\sigma)\}$ where $r(\sigma(v))$ is defined as follows:
   - $r(t) = t$ if $t \in \mathcal{V}_{\mathrm{Ind}} \cup \mathcal{V}_{\mathrm{Fun}} \cup \mathcal{V}_{\mathrm{Seq}} \cup \mathcal{V}_{\mathrm{Con}} \cup \mathcal{F} \cup \{\circ\}$,
   - $r((he_1, \ldots, he_n)) = (r(he_1)\texttt{,}\ldots\texttt{,}r(he_n))$,
   - $r(\texttt{h}(\tilde{t})) = r(\texttt{h})(r(\tilde{t}))$

   where $\texttt{h} \in \mathcal{F} \cup \mathcal{V}_{\mathrm{Fun}} \cup \mathcal{V}_{\mathrm{Con}}$ and `,` is considered to be right-associative.
2. Replace each subexpression $\texttt{Rc}(\texttt{Re}')$, where `Rc` contains no other regular operator except possibly "`,`", with the regular expression obtained by plugging $\texttt{Re}'$ into the hole of `Rc`. Also, replace each subexpression of the form `(),Rh` and `Rh,()` with `Rh`. Repeat the step 2 until it is no longer applicable.

Example 2. Let $\texttt{Re} = f(\overline{C}(a^{*}\texttt{,}g(\overline{D}(\overline{x}\texttt{,}\circ))))^{\star}$ and $\sigma = \{\overline{x} \mapsto (), \overline{C} \mapsto F(x, \circ), \overline{D} \mapsto G(a, b, c, \circ)\}$. Then $\texttt{Re}\sigma = f(F(x\texttt{,}a^{*}\texttt{,}g(G(a\texttt{,}b\texttt{,}c\texttt{,}\circ))))^{\star}$.

The regular language $[\![\texttt{Rh}]\!]$ generated by a regular hedge expression `Rh` is a set of hole-free hedges defined as follows:

$$\begin{aligned}
[\![v]\!] &= \{v\}, \text{ where } v \in \mathcal{V}_{\mathrm{Ind}} \cup \mathcal{V}_{\mathrm{Seq}}. \\
[\![\texttt{h(Rh)}]\!] &= \{\texttt{h}(\tilde{s}) \mid \tilde{s} \in [\![\texttt{Rh}]\!]\}. \\
[\![\texttt{()}]\!] &= \{()\}. \\
[\![\texttt{Rh}_1\texttt{,}\texttt{Rh}_2]\!] &= \{(\tilde{s}_1, \tilde{s}_2) \mid \tilde{s}_1 \in [\![\texttt{Rh}_1]\!],\ \tilde{s}_2 \in [\![\texttt{Rh}_2]\!]\}. \\
[\![\texttt{Rh}_1\texttt{|}\texttt{Rh}_2]\!] &= [\![\texttt{Rh}_1]\!] \cup [\![\texttt{Rh}_2]\!]. \\
[\![\texttt{Rh}^{*}]\!] &= \bigcup_{n \geq 0} [\![\texttt{Rh}]\!]^{n}. \\
[\![\texttt{Rc(Rh)}]\!] &= \{C[\tilde{s}] \mid C \in [\![\texttt{Rc}]\!],\ \tilde{s} \in [\![\texttt{Rh}]\!]\}.
\end{aligned}$$

Here $[\![\mathtt{Rh}]\!]^0 = \{()\}$ and $[\![\mathtt{Rh}]\!]^{n+1} = \{(\tilde{s}_1, \tilde{s}_2) \mid \tilde{s}_1 \in [\![\mathtt{Rh}]\!], \tilde{s}_2 \in [\![\mathtt{Rh}]\!]^n\}$ for $n \geq 0$.

The regular language $[\![\mathtt{Rc}]\!]$ generated by a regular context expression $\mathtt{Rc}$ is a set of contexts defined as follows:

$$\begin{aligned}
[\![\circ]\!] &= \{\circ\}. \\
[\![\mathtt{Rh,Rc}]\!] &= \{(\tilde{s}, C) \mid \tilde{s} \in [\![\mathtt{Rh}]\!],\ C \in [\![\mathtt{Rc}]\!]\}. \\
[\![\mathtt{Rc,Rh}]\!] &= \{(C, \tilde{s}) \mid C \in [\![\mathtt{Rc}]\!],\ \tilde{s} \in [\![\mathtt{Rh}]\!]\}. \\
[\![\mathtt{h(Rc)}]\!] &= \{\mathtt{h}(C) \mid C \in [\![\mathtt{Rc}]\!]\}. \\
[\![\mathtt{Rc}_1\mathtt{.Rc}_2]\!] &= \{C_1[C_2] \mid C_1 \in [\![\mathtt{Rc}_1]\!],\ C_2 \in [\![\mathtt{Rc}_2]\!]\}. \\
[\![\mathtt{Rc}_1\mathtt{+Rc}_2]\!] &= [\![\mathtt{Rc}_1]\!] \cup [\![\mathtt{Rc}_2]\!]. \\
[\![\mathtt{Rc}^\star]\!] &= \bigcup_{n \geq 0} [\![\mathtt{Rc}]\!]^n,
\end{aligned}$$

where $[\![\mathtt{Rc}]\!]^0 = \{\circ\}$ and $[\![\mathtt{Rc}]\!]^{n+1} = \{C_1[C_2] \mid C_1 \in [\![\mathtt{Rc}]\!], C_2 \in [\![\mathtt{Rc}]\!]^n\}$ for $n \geq 0$.

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

$$\begin{aligned}
[\![\mathtt{Rc}]\!] = \{&\circ, f(\overline{C}(\circ), b), f(\overline{C}(\circ), x, b), \ldots, f(\overline{C}(\circ), x, \ldots, x, b), \ldots, \\
&f(\overline{C}(f(\overline{C}(\circ), b)), b), f(\overline{C}(f(\overline{C}(\circ), b)), x, b), \ldots\}.
\end{aligned}$$

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

Example 4. The expressions $f(\circ)^\star(a)$ and $f(a)^*$ generate different languages: $[\![f(\circ)^\star(a)]\!] = \{a, f(a), f(f(a)), \ldots\}$ while $[\![f(a)^*]\!] = \{(), f(a), (f(a), f(a)), \ldots\}$.

Membership atoms are atoms of the form $(\tilde{s}\ \mathsf{in}\ \mathtt{Rh}, \mathtt{f})$ or $(Cv\ \mathsf{in}\ \mathtt{Rc}, \mathtt{f})$, where $\tilde{s}$ is a hedge, Cv is either a context or a context variable, and $\mathtt{f}$ is a flag. *Flag* is either 0, 1, or $\mathtt{trivial}(\tilde{t})$, where $\mathtt{trivial}$ satisfies the equalities:

- $\mathtt{trivial}(\circ) = \mathtt{trivial}(()) = 1$.
- $\mathtt{trivial}(\tilde{t}) = 0$ if the hedge $\tilde{t}$ contains at least one symbol from the set $\mathcal{V}_{\mathrm{Ind}} \cup \mathcal{V}_{\mathrm{Fun}} \cup \mathcal{F} \setminus \{\circ\}$.

Hence, if $\mathtt{trivial}(\tilde{t}) = 0$ then no instance of $\tilde{t}$ can be $\circ$ or $()$. *Membership constraints* are finite multisets of membership atoms. A membership constraint is *well-moded* if it can be ordered as $\{(\overline{v}_1\ \mathsf{in}\ \mathtt{Re}_1, \mathtt{f}_1), \ldots, (\overline{v}_n\ \mathsf{in}\ \mathtt{Re}_n, \mathtt{f}_n)\}$ where $\overline{v}$'s are sequence or context variables (called *constrained* variables) such that $\overline{v}_i \notin vars(\mathtt{Re}_j)$ for $1 \leq i \leq j \leq n$ and $\overline{v}_i \notin vars(\mathtt{f}_k)$ for $1 \leq k \leq i \leq n$.

A *regular well-moded constraint* (RWMC in short) is a finite multiset of equations and membership atoms of the form

$$\{\tilde{s}_1 \approx^? \tilde{t}_1, \ldots, \tilde{s}_n \approx^? \tilde{t}_n,\ (\overline{v}_1\ \mathsf{in}\ \mathtt{Re}_1, \mathtt{f}_1) \ldots, (\overline{v}_m\ \mathsf{in}\ \mathtt{Re}_m, \mathtt{f}_m)\},$$

where $\tilde{s}$'s and $\tilde{t}$'s are hole-free hedges, $\{\tilde{s}_1 \approx^? \tilde{t}_1, \ldots, \tilde{s}_n \approx^? \tilde{t}_n\}$ is a well-moded equational constraint, $\{(\overline{v}_1\ \mathsf{in}\ \mathtt{Re}_1, \mathtt{f}_1) \ldots, (\overline{v}_m\ \mathsf{in}\ \mathtt{Re}_m, \mathtt{f}_m)\}$ is a well-moded membership constraint, and for each $(\overline{v}\ \mathsf{in}\ \mathtt{Re}, \mathtt{f})$, if $\overline{v}$ does not occur in

s's then $\mathtt{f} = 0$.[3] We use Γ and Δ to denote RWMC constraints. A substitution σ is called a *solution* for such a constraint if

$$\tilde{s}_1\sigma = \tilde{t}_1\sigma, \ldots, \tilde{s}_n\sigma = \tilde{t}_n\sigma,\ \overline{v}_1\sigma \in [\![\mathtt{Re}_1\sigma]\!]_{\mathtt{f}_1\sigma}, \ldots, \overline{v}_m\sigma \in [\![\mathtt{Re}_m\sigma]\!]_{\mathtt{f}_m\sigma},$$
$$\mathtt{f}_1\sigma \in \{0,1\}, \ldots, \mathtt{f}_m\sigma \in \{0,1\},$$

where $[\![\mathtt{Rh}]\!]_1 = [\![\mathtt{Rh}]\!] \setminus \{()\}$, $[\![\mathtt{Rh}]\!]_0 = [\![\mathtt{Rh}]\!]$, $[\![\mathtt{Rc}]\!]_1 = [\![\mathtt{Rc}]\!] \setminus \{\circ\}$, and $[\![\mathtt{Rc}]\!]_0 = [\![\mathtt{Rc}]\!]$. 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 RWMC constraint Γ if it occurs in the equational part of Γ. We denote the set of equational variables of Γ with $eqv(\Gamma)$. 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 Constraint Solving

We start with RWMC solving where all constrained variables are distinct. Later we will lift this restriction.

The inference system consists of five groups of rules presented below. Rules operate on systems. A *system* is either the symbol $\bot$ (failure) or a pair $\Gamma;\sigma$. To save space, we will not spell explicitly rules that transform systems into $\bot$ (so called *failure* rules). The reader can assume that the systems that do not satisfy the conditions of the transformation rules below are transformed into $\bot$.

The first group of rules does not affect membership constraints. It is denoted by $\mathfrak{R}_{\mathrm{Eq}}$ and consists of the following 7 rules:

T: **Trivial**
$\{() \ll ()\} \cup \Gamma;\ \sigma \Longrightarrow \Gamma;\ \sigma.$

IVE: **Individual Variable Elimination**
$\{(x, \tilde{s}) \ll (t, \tilde{t})\} \cup \Gamma;\ \sigma \Longrightarrow \{\tilde{s} \ll \tilde{t}\} \cup \Gamma\vartheta;\ \sigma\vartheta, \qquad$ where $\vartheta = \{x \mapsto t\}$.

FVE: **Function Variable Elimination**
$\{(F(\tilde{s}_1), \tilde{s}_2) \ll (f(\tilde{t}_1), \tilde{t}_2)\} \cup \Gamma;\ \sigma \Longrightarrow \{\tilde{s}_1\vartheta \ll \tilde{t}_1,\ \tilde{s}_2\vartheta \ll \tilde{t}_2\} \cup \Gamma\vartheta;\ \sigma\vartheta,$
where $\vartheta = \{F \mapsto f\}$.

Dec: **Decomposition**
$\{(f(\tilde{s}_1), \tilde{s}_2) \ll (f(\tilde{t}_1), \tilde{t}_2)\} \cup \Gamma;\ \sigma \Longrightarrow \{\tilde{s}_1 \ll \tilde{t}_1,\ \tilde{s}_2 \ll \tilde{t}_2\} \cup \Gamma;\ \sigma.$

SVE: **Sequence Variable Elimination**
$\{(\overline{x}, \tilde{s}) \ll (\tilde{t}_1, \tilde{t}_2)\} \cup \Gamma;\ \sigma \Longrightarrow \{\tilde{s}\vartheta \ll \tilde{t}_2\} \cup \Gamma\vartheta;\ \sigma\vartheta$
where $\overline{x}$ is not constrained in Γ and $\vartheta = \{\overline{x} \mapsto \tilde{t}_1\}$.

[3] The last condition is imposed to guarantee that every RWMC that consists of only membership constraints where each variable is constrained once, is solvable.

F: **Flattening**

$\{(\overline{C}(\tilde{s}_1), \tilde{s}_2) \ll (\tilde{t}_1, \tilde{t}_2, \tilde{t}_3, \tilde{t}_4)\} \cup \Gamma;\ \sigma \Longrightarrow \{\tilde{s}_1\vartheta \ll \tilde{t}_2,\ \tilde{s}_2\vartheta \ll \tilde{t}_4\} \cup \Gamma\vartheta;\ \sigma\vartheta,$
where $\overline{C}$ is not constrained in Γ and $\vartheta = \{\overline{C} \mapsto (\tilde{t}_1, \circ, \tilde{t}_3)\}$.

D: **Deepening**

$\{(\overline{C}(\tilde{s}_1), \tilde{s}_2) \ll (\tilde{t}_1, f(\tilde{r}), \tilde{t}_2, \tilde{t}_3)\} \cup \Gamma;\ \sigma \Longrightarrow \{\overline{D}(\tilde{s}_1\vartheta) \ll \tilde{r},\ \tilde{s}_2\vartheta \ll \tilde{t}_3\} \cup \Gamma\vartheta; \sigma\vartheta,$
where $\overline{C}$ is not constrained in Γ, $\vartheta = \{\overline{C} \mapsto (\tilde{t}_1, f(\overline{D}(\circ)), \tilde{t}_2)\}$ and $\overline{D}$ is fresh.

Note that SVE covers also the case when $\overline{x}$ is replaced by the empty hedge (). For instance, applying SVE on $\{(\overline{x}, a) \ll a\}; \varepsilon$ can give $\{a \ll a\}; \{\overline{x} \mapsto ()\}$. The rules SVE, F, and D are nondeterministic rules.

The second group, the $\mathfrak{R}_{\mathrm{Hg}}$, consists of 10 rules that transform equations coupled with constraints for regular hedge expressions, where the first term of the left hand side of the selected equation is a sequence variable constrained by a membership constraint.

IVHg: **Individual Variable as Regular Hedge Expression**

$\{(\overline{x}, \tilde{s}) \ll \tilde{t},\ (\overline{x} \text{ in } x, \mathtt{f})\} \cup \Gamma;\ \sigma \Longrightarrow \{(x, \tilde{s}\vartheta) \ll \tilde{t}\} \cup \Gamma\vartheta;\ \sigma\vartheta$
where $\vartheta = \{\overline{x} \mapsto x\}$.

SVHg: **Sequence Variable as Regular Hedge Expression**

$\{(\overline{x}, \tilde{s}) \ll \tilde{t},\ (\overline{x} \text{ in } \overline{y}, \mathtt{f}_1)\} \cup \Gamma;\ \sigma \Longrightarrow \{(\overline{y}, \overline{z}, \tilde{s}\vartheta) \ll \tilde{t}, (\overline{z} \text{ in } (), \mathtt{f}_2)\} \cup \Gamma\vartheta;\ \sigma\vartheta$
where $\vartheta = \{\overline{x} \mapsto \overline{y}\}$, $\overline{z}$ is fresh, $\mathtt{f}_2 = 0$ if $\mathtt{f}_1 = 0$ and $\mathtt{f}_2 = \mathtt{trivial}(\overline{y})$ if $\mathtt{f}_1 = 1$.

EHHg: **Empty Hedge as a Regular Hedge Expression**

$\{(\overline{x}, \tilde{s}) \ll \tilde{t},\ (\overline{x} \text{ in } (), 0)\} \cup \Gamma;\ \sigma \Longrightarrow \{\tilde{s}\vartheta \ll \tilde{t}\} \cup \Gamma\vartheta;\ \sigma\vartheta,$
where $\vartheta = \{\overline{x} \mapsto ()\}$.

FHg: **Function Symbol or Variable in a Regular Hedge Expression**

$\{(\overline{x}, \tilde{s}) \ll \tilde{t},\ (\overline{x} \text{ in } \mathtt{g}(\mathtt{Rh}), \mathtt{f})\} \cup \Gamma;\ \sigma \Longrightarrow \{(\mathtt{g}(\overline{y}), \tilde{s}\vartheta) \ll \tilde{t}, (\overline{y} \text{ in } \mathtt{Rh}, 0)\} \cup \Gamma\vartheta;\ \sigma\vartheta,$
where $\mathtt{g} \in \mathcal{V}_{\mathrm{Fun}} \cup \mathcal{F}$, $\vartheta = \{\overline{x} \mapsto \mathtt{g}(\overline{y})\}$, $\overline{y}$ is fresh.

CVHg: **Context Variable in a Regular Hedge Expression**

$\{(\overline{x}, \tilde{s}) \ll \tilde{t},\ (\overline{x} \text{ in } \overline{C}(\mathtt{Rh}), \mathtt{f}_1)\} \cup \Gamma;\ \sigma \Longrightarrow \{(\overline{C}(\overline{y}), \tilde{s}\vartheta) \ll \tilde{t}, (\overline{y} \text{ in } \mathtt{Rh}, \mathtt{f}_2)\} \cup \Gamma\vartheta;\ \sigma\vartheta,$
where $\vartheta = \{\overline{x} \mapsto \overline{C}(\overline{y})\}$, $\overline{y}$ is fresh, $\mathtt{f}_2 = 0$ if $\mathtt{f}_1 = 0$ and $\mathtt{f}_2 = \mathtt{trivial}(\overline{C}(\circ))$ if $\mathtt{f}_1 = 1$.

ChHg: **Choice as a Regular Hedge Expression**

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

CHg: **Concatenation as a Regular Hedge Expression**

$\{(\overline{x}, \tilde{s}) \ll \tilde{t},\ (\overline{x} \text{ in } (\mathtt{Rh}_1, \mathtt{Rh}_2), \mathtt{f}_1)\} \cup \Gamma;\ \sigma$
$\Longrightarrow \{(\overline{y}_1, \overline{y}_2, \tilde{s}\vartheta) \ll \tilde{t},\ (\overline{y}_1 \text{ in } \mathtt{Rh}_1, 0),\ (\overline{y}_2 \text{ in } \mathtt{Rh}_2, \mathtt{f}_2)\} \cup \Gamma\vartheta;\ \sigma\vartheta,$
where $\overline{y}_1$ and $\overline{y}_2$ are fresh variables, $\vartheta = \{\overline{x} \mapsto (\overline{y}_1, \overline{y}_2)\}$, $\mathtt{f}_2 = 0$ if $\mathtt{f}_1 = 0$ and $\mathtt{f}_2 = \mathtt{trivial}(\overline{y}_1)$ if $\mathtt{f}_1 = 1$.

RHg1: **Repetition as a Regular Hedge Expression 1**

$\{(\overline{x}, \tilde{s}) \ll \tilde{t},\ (\overline{x} \text{ in } \mathtt{Rh}^*, 0)\} \cup \Gamma;\ \sigma \Longrightarrow \{\tilde{s}\vartheta \ll \tilde{t}\} \cup \Gamma\vartheta;\ \sigma\vartheta,$

where $\vartheta = \{\overline{x} \mapsto ()\}$.

RHg2: **Repetition as a Regular Hedge Expression 2**

$\{(\overline{x}, \tilde{s}) \ll \tilde{t},\ (\overline{x} \text{ in } \mathtt{Rh}^*, \mathtt{f})\} \cup \Gamma;\ \sigma$
$\Longrightarrow \{(\overline{y}, \overline{z}, \tilde{s}\vartheta) \ll \tilde{t},\ (\overline{y} \text{ in } \mathtt{Rh}, 1),\ (\overline{z} \text{ in } \mathtt{Rh}^*, 0)\} \cup \Gamma\vartheta;\ \sigma\vartheta,$

where $\overline{y}$ and $\overline{z}$ are fresh variables and $\vartheta = \{\overline{x} \mapsto (\overline{y}, \overline{z})\}$.

AHg: **Application in a Regular Hedge Expression**

$\{(\overline{x}, \tilde{s}) \ll \tilde{t},\ (\overline{x} \text{ in } \mathtt{Rc}(\mathtt{Rh}), \mathtt{f})\} \cup \Gamma;\ \sigma$
$\Longrightarrow \{(\overline{x}, \tilde{s}) \ll \tilde{t},\ (\overline{x} \text{ in } \overline{C}(\mathtt{Rh}), \mathtt{f}),\ (\overline{C} \text{ in } \mathtt{Rc}, 0)\} \cup \Gamma;\ \sigma,$

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

The third group of rules operates on equations and constraints on regular context expressions. This group is denoted by $\mathfrak{R}_{\mathrm{Ctx}}$. The context variable of the left hand side of the selected equation in the rules in $\mathfrak{R}_{\mathrm{Ctx}}$ is constrained by a single membership constraint. We have the following 9 rules in $\mathfrak{R}_{\mathrm{Ctx}}$:

HCtx: **Hole as a Regular Context Expression**

$\{(\overline{C}(\tilde{s}_1), \tilde{s}_2) \ll \tilde{t},\ (\overline{C} \text{ in } \circ, 0)\} \cup \Gamma;\ \sigma \Longrightarrow \{(\tilde{s}_1, \tilde{s}_2)\vartheta \ll \tilde{t}\} \cup \Gamma\vartheta;\ \sigma\vartheta,$

where $\vartheta = \{\overline{C} \mapsto \circ\}$.

HCCtx: **Hedge-Context Concatenation as a Regular Context Expression**

$\{(\overline{C}(\tilde{s}_1), \tilde{s}_2) \ll (\tilde{t}_1, \tilde{t}_2),\ (\overline{C} \text{ in } \mathtt{Rh},\mathtt{Rc},\ \mathtt{f}_1)\} \cup \Gamma;\ \sigma$
$\Longrightarrow \{\overline{x} \ll \tilde{t}_1,\ (\overline{x} \text{ in } \mathtt{Rh}, 0),\ (\overline{D}(\tilde{s}_1\vartheta), \tilde{s}_2\vartheta) \ll \tilde{t}_2,\ (\overline{D} \text{ in } \mathtt{Rc}, \mathtt{f}_2)\} \cup \Gamma\vartheta;\ \sigma\vartheta,$

where $\overline{x}$ and $\overline{D}$ are fresh variables, $\vartheta = \{\overline{C} \mapsto (\overline{x}, \overline{D}(\circ))\}$, $\mathtt{f}_2 = 0$ if $\mathtt{f}_1 = 0$ and $\mathtt{f}_2 = \mathtt{trivial}(\overline{x})$ if $\mathtt{f}_1 = 1$.

CHCtx: **Context-Hedge Concatenation as a Regular Context Expression**

$\{(\overline{C}(\tilde{s}_1), \tilde{s}_2) \ll (\tilde{t}_1, \tilde{t}_2),\ (\overline{C} \text{ in } \mathtt{Rc},\mathtt{Rh},\ \mathtt{f}_1)\} \cup \Gamma;\ \sigma$
$\Longrightarrow \{\overline{D}(\tilde{s}_1\vartheta) \ll \tilde{t}_1,\ (\overline{D} \text{ in } \mathtt{Rc}, 0),\ (\overline{x}, \tilde{s}_2) \ll \tilde{t}_2,\ (\overline{x} \text{ in } \mathtt{Rh}, \mathtt{f}_2)\} \cup \Gamma\vartheta;\ \sigma\vartheta,$

where $\overline{x}$ and $\overline{D}$ are fresh variables, $\vartheta = \{\overline{C} \mapsto (\overline{D}(\circ), \overline{x})\}$, $\mathtt{f}_2 = 0$ if $\mathtt{f}_1 = 0$ and $\mathtt{f}_2 = \mathtt{trivial}(\overline{D}(\circ))$ if $\mathtt{f}_1 = 1$.

FCtx: **Function Symbol or Variable in a Regular Context Expression**

$\{(\overline{C}(\tilde{s}_1), \tilde{s}_2) \ll \tilde{t},\ (\overline{C} \text{ in } \mathsf{h}(\mathtt{Rc}), \mathtt{f})\} \cup \Gamma;\ \sigma$
$\Longrightarrow \{(\mathsf{h}(\overline{D}(\tilde{s}_1\vartheta)), \tilde{s}\vartheta) \ll \tilde{t},\ (\overline{D} \text{ in } \mathtt{Rc}, 0)\} \cup \Gamma\vartheta;\ \sigma\vartheta,$

where $\mathsf{h} \in \mathcal{V}_{\mathrm{Fun}} \cup \mathcal{F}$, $\overline{D}$ is fresh, $\vartheta = \{\overline{C} \mapsto \mathsf{h}(\overline{D}(\circ))\}$, and $\overline{C}$ is not constrained in Γ.

CVCtx: **Context Variable in a Regular Context Expression**

$\{(\overline{C}(\tilde{s}_1), \tilde{s}_2) \ll \tilde{t},\ (\overline{C} \text{ in } \overline{D}(\mathtt{Rc}), \mathtt{f}_1)\} \cup \Gamma;\ \sigma$
$\Longrightarrow \{(\overline{D}(\overline{E}(\tilde{s}_1\vartheta)), \tilde{s}_2\vartheta) \ll \tilde{t},\ (\overline{E} \text{ in } \mathtt{Rc}, \mathtt{f}_2)\} \cup \Gamma\vartheta;\ \sigma\vartheta,$

where $\overline{E}$ is fresh, $\vartheta = \{\overline{C} \mapsto \overline{D}(\overline{E}(\circ))\}$, $\mathtt{f}_2 = 0$ if $\mathtt{f}_1 = 0$ and $\mathtt{f}_2 = \mathtt{trivial}(\overline{D}(\circ))$ if $\mathtt{f}_1 = 1$.

ChCtx: **Choice as a Regular Context Expression**

$\{(\overline{C}(\tilde{s}_1), \tilde{s}_2) \ll \tilde{t},\ (\overline{C} \text{ in } \mathtt{Rc}_1\mathtt{+Rc}_2, \mathtt{f})\} \cup \Gamma;\ \sigma$
$\quad \Longrightarrow \{(\overline{C}(\tilde{s}_1), \tilde{s}_2) \ll \tilde{t},\ (\overline{C} \text{ in } \mathtt{Rc}_i, \mathtt{f})\} \cup \Gamma;\ \sigma \quad \text{for } i = 1, 2.$

CCtx: **Concatenation as a Regular Context Expression**

$\{(\overline{C}(\tilde{s}_1), \tilde{s}_2) \ll \tilde{t},\ (\overline{C} \text{ in } \mathtt{Rc}_1\mathtt{.Rc}_2, \mathtt{f}_1)\} \cup \Gamma;\ \sigma$
$\quad \Longrightarrow \{(\overline{D}(\overline{E}(\tilde{s}_1\vartheta)), \tilde{s}_2\vartheta) \ll \tilde{t},\ (\overline{D} \text{ in } \mathtt{Rc}_1, 0),\ (\overline{E} \text{ in } \mathtt{Rc}_2, \mathtt{f}_2)\} \cup \Gamma\vartheta;\ \sigma\vartheta,$

where $\overline{D}$ and $\overline{E}$ are fresh variables, $\vartheta = \{\overline{C} \mapsto \overline{D}(\overline{E}(\circ))\}$, and $\mathtt{f}_2 = 0$ if $\mathtt{f}_1 = 0$ and $\mathtt{f}_2 = \mathtt{trivial}(\overline{D}(\circ))$ if $\mathtt{f}_1 = 1$.

RCtx1: **Repetition as a Regular Context Expression 1**

$\{(\overline{C}(\tilde{s}_1), \tilde{s}_2) \ll t,\ (\overline{C} \text{ in } \mathtt{Rc}^\star, 0)\} \cup \Gamma;\ \sigma \Longrightarrow \{(\tilde{s}_1, \tilde{s}_2)\vartheta \ll t\} \cup \Gamma\vartheta;\ \sigma\vartheta,$

where $\vartheta = \{\overline{C} \mapsto \circ\}$.

RCtx2: **Repetition as a Regular Context Expression 2**

$\{(\overline{C}(\tilde{s}_1), \tilde{s}_2) \ll \tilde{t},\ (\overline{C} \text{ in } \mathtt{Rc}^\star, \mathtt{f})\} \cup \Gamma;\ \sigma$
$\quad \Longrightarrow \{(\overline{D}(\overline{E}(\tilde{s}_1\vartheta), \tilde{s}_2\vartheta) \ll t, (\overline{D} \text{ in } \mathtt{Rc}, 1),\ (\overline{E} \text{ in } \mathtt{Rc}^\star, 0)\} \cup \Gamma\vartheta;\ \sigma\vartheta,$

where $\overline{D}$ and $\overline{E}$ are fresh variables, and $\vartheta = \{\overline{C} \mapsto \overline{D}(\overline{E}(\circ))\}$.

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

MC: **Membership Constraints**

$\Gamma;\ \sigma \Longrightarrow \emptyset;\ \sigma, \qquad$ if Γ consists only of regular constraints.

We denote by $\mathfrak{R}$ the set $\mathfrak{R}_{\mathrm{Eq}} \cup \mathfrak{R}_{\mathrm{Hg}} \cup \mathfrak{R}_{\mathrm{Ctx}} \cup \mathfrak{R}_{\mathrm{Sat}}$.

The following lemma is straightforward to verify by just inspecting the rules:

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

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 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 $\Gamma; \emptyset$ as input, where Γ is an* RWMC*, 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 $\Gamma; \emptyset$.*
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.*[4]
4. *If several rules, or different instances of the same rule are applicable to a node, they are applied concurrently.*

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

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 the leaf of a successful branch of the solving tree for Γ. We denote by $compans_{\mathfrak{C}}(\Gamma)$ the set of answers of Γ computed by $\mathfrak{C}$. Mostly we will be interested in the set $compans_{\mathfrak{C}}(\Gamma)|_{vars(\Gamma)}$: the restriction of $compans_{\mathfrak{C}}(\Gamma)$ to $vars(\Gamma)$.

Flags prevent looping in the cases when regular expressions accept the empty sequence or the hole under star. (In principle, one could avoid using flags by rewriting such problematic regular expressions, but it would cause exponential blowup of the regular expression size. See [14] for more discussion.)

Example 5. Consider a derivation of $\{\overline{x} \ll (b,a),\ (\overline{x} \ \mathsf{in}\ (a^* \,|\, b^*)^*, 0)\}$ (selected equations and membership pairs are framed):

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

If we did not have flags, in particular, the flag in $(\overline{y} \ \mathsf{in}\ a^*, 1)$ set to 1, at this step we could replace $\overline{y}$ with $()$, obtain the initial problem and, hence, a loop. But the flag prevents this, and the derivation continues with the RHg2 rule:

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

The main theorem states that $\mathfrak{C}$ is a sound, terminating, and complete solving method for RWMC constraints:

Theorem 1 (Main Theorem). *Let Γ be a* RWMC*. Then*

- *For each $\sigma \in compans_{\mathfrak{C}}(\Gamma)$ there exists $\vartheta \in sol(\Gamma)$ such that $\sigma \leq \vartheta$.*
- *$\mathfrak{C}$ terminates on Γ.*
- *For each $\vartheta \in sol(\Gamma)$ there exists $\sigma \in compans_{\mathfrak{C}}(\Gamma)$ such that $\sigma|_{eqv(\Gamma)} = \vartheta|_{eqv(\Gamma)}$ and $\sigma \leq^{vars(\Gamma)} \vartheta$.*

Proof. See the appendix. □

Now we lift the restriction that required all constrained variables in RWMC's to be distinct. However, in order to avoid context and sequence unification we have to require that the variables constrained by more than one regular expression are equational. Such constraints can be reduced to RWMC's by exhaustive application of the following two rules:

IHg: **Intersection of Regular Hedge Expressions**

$$\{(\overline{x}, \tilde{s}) \ll \tilde{t},\ (\overline{x} \text{ in } \mathtt{Rh}_1, \mathtt{f}_1)\,(\overline{x} \text{ in } \mathtt{Rh}_2, \mathtt{f}_2)\} \cup \Gamma;\ \sigma$$
$$\Longrightarrow \{(\overline{x}, \tilde{s}) \ll \tilde{t},\ (\overline{x} \text{ in } \mathtt{Rh}_1, \mathtt{f}_1), \overline{y} \ll \overline{x},\ (\overline{y} \text{ in } \mathtt{Rh}_2, \mathtt{f}_2)\} \cup \Gamma;\ \sigma,$$

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

ICtx: **Intersection of Regular Context Expressions**

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

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

It is easy to observe that these rules are sound and terminating.

5 Related Work and Concluding Remarks

The approach is flexible and expressive: It allows to traverse the data, represented as a hedge, in horizontal and in vertical directions (using four different kinds of variables). Variables can be constrained by regular expressions. The same variable can be constrained in several ways. Regular expressions may themselves contain variables and the algorithm computes their instantiations as well. Well-modedness allows to use iterated matching instead of full-scale unification.

Our work is related to various formalisms used in constraint solving, programming, and querying. Here we just mention some, most close ones.

Context matching [34] is a particular instance of RWMC 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 [21] can be obtained by the following restrictions: Regular expressions can have regular operators (except hedge concatenation) in the top level only; contexts should be a single expression (not a hedge); contexts may apply to terms, not to hedges; all the equations should be matching equations; each sequence and context variable can be constrained by maximum one membership constraint; all constrained variables should be equational; no constrained variable can occur in regular expressions.

Example 6. Let $\Gamma = \{\overline{C}(b) \ll f(a, a, f(a, f(b))), (\overline{C} \text{ in } f(a^*, \circ)^\star, 0)\}$ be a RWMC constraint. The algorithm $\mathfrak{C}$ solves it with $\{\overline{C} \mapsto f(a, a, f(a, f(\circ)))\}$. This constraint can not be expressed in the syntax of [21]. 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 (a, a) and will not match a.

We can directly model the pattern matching mechanism of the programming language of MATHEMATICA [37]. 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 [6] can be captured by first trying those rules from $\mathfrak{R}$ that assign () to sequence variables, and stopping the algorithm $\mathfrak{C}$ when the first matcher is computed.

A rule-based system ρLOG [26] implements matching with individual, function, sequence, and context variables subsumed by RWMC. Various special versions of RWMC solving found their way into the mathematical software system THEOREMA [7].

Comon [10] 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 hedges without variables and the star operator. Context expressions correspond to regular contexts expressions without sequence and function variables. Sequence and function variables do not occur in the equations either.

We can encode one-step rewrite constraints [8] $s_1 \to t_1$ 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 [30] 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 [28] (that subsume one-step rewrite constraints) can be expressed.

Regular expression pattern matching [17] is used for tree manipulation, primarily for XML, in a statically typed setting. Regular expression patterns basically correspond to our regular hedge expressions without individual variables and contexts. The effect of regular context expressions 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 [29] 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. For monadic queries an efficient and expressive information extraction approach,

monadic Datalog, was proposed by Gottlob and Koch [16]. RWMC constraint solving is closely related with some other solving methods proposed for querying and transforming semistructured data (XML, in particular), like simulation unification [5] of XCERPT, 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 [24] 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. [13]) that can be adapted for this case.

We can use well-moded membership constraints to express regular hedge grammars [27], without multiple recursion.

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.* 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. H. Boley. *A Tight, Practical Integration of Relations and Functions*, volume 1712 of LNAI. Springer, 1999.
4. 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.
5. F. Bry and S. Schaffert. Towards a declarative query and transformation language for XML and semistructured data: Simulation unification. In *Proc.* ICLP'02, number 2401 in LNCS. Springer, 2002.
6. 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.
7. 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.
8. A.-C. Caron, J.-L. Coquidé, and M. Dauchet. Encompassment properties and automata with constraints. In C. Kirchner, editor, *Proc.* RTA'93, volume 690 of LNCS, pages 328–342. Springer, 1993.
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. H. Comon. Completion of rewrite systems with membership constraints. Part II: Constraint solving. *J. Symb. Comp.*, 25(4):421–453, 1998.
11. 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.
12. P. Deransart and J. Maluszynski. Relating logic programs and attribute grammars. *J. Log. Program.*, 2(2):119–155, 1985.
13. S. Eker. Fast matching in combinations of regular equational theories. *Electronic Notes in Theoretical Computer Science*, 4, 1996.
14. A. Frisch and L. Cardelli. Greedy regular expression matching. In *Proc.* ICALP'04, pages 618–629, 2004.
15. M. Ginsberg. The MVL theorem proving system. SIGART *Bull.*, 2(3):57–60, 1991.
16. G. Gottlob and Ch. Koch. Monadic Datalog and the expressive power of languages for web information retrieval. *J.* ACM, 51(1):74–113, 2004.
17. H. Hosoya and B. Pierce. Regular expression pattern matching for XML. *J. Functional Programming*, 13(6):961–1004, 2003.
18. T. Kutsia. *Solving and Proving in Equational Theories with Sequence Variables and Flexible Arity Symbols.* PhD thesis, Johannes Kepler University, Linz, 2002.
19. 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.* AISC'2002—CALCULEMUS'2002 *Conference*, volume 2385 of LNAI, pages 290–304. Springer, 2002.
20. T. Kutsia. Solving equations involving sequence variables and sequence functions. In B. Buchberger and J. A. Campbell, editors, *Proc.* AISC *'04*, volume 3249 of LNAI, pages 157–170. Springer, 2004.
21. T. Kutsia and M. Marin. Matching with regular constraints. In G. Sutcliffe and A. Voronkov, editors, *Proc.* LPAR'05, volume 3835 of LNAI, pages 215–229. Springer, 2005.
22. J. Levy. Linear second-order unification. In H. Ganzinger, editor, *Proc.* RTA *'96*, volume 1103 of LNCS, pages 332–346. Springer, 1996.
23. J. Levy and M. Villaret. Linear second-order unification and context unification with tree-regular constraints. In L. Bachmair, editor, *Proc.* RTA *'2000*, volume 1833 of LNCS, pages 156–171. Springer, 2000.
24. D. Lugiez. Multitree automata that count. *Theor. Comput. Sci.*, 333(1–2):225–263, 2005.
25. J. Maluszynski and H. Komorowski. Unification-free execution of logic programs. In *Proc.* SLP, pages 78–86, 1985.
26. M. Marin and T. Kutsia. Foundations of the rule-based system ρLog. *J. Applied Non-Classical Logics*, 2005. To Appear.
27. M. Murata, D. Lee, and M. Mani. Taxonomy of XML schema languages using formal language theory. In *Extreme Markup Languages*, 2001.
28. 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.* CADE-14, volume 1249 of LNCS, pages 34–48. Springer, 1997.
29. J. Niehren, L. Planque, J.-M. Talbot, and S. Tison. N-ary queries by tree automata. In *Proc.* DBPL *'05*, 2005.
30. J. Niehren, S. Tison, and R. Treinen. On rewrite constraints and context unification. *Inf. Process. Lett.*, 74(1-2):35–40, 2000.
31. S. Okui and T. Suzuki. Pattern matching incompletely RE-typed expressions via transformations. IPSJ *Transactions on Programming*, 47, 2006. To appear.

32. M. Schmidt-Schauß. Unification of stratified second-order terms. Internal Report 12/94, Johann-Wolfgang-Goethe-Universität, Frankfurt, Germany, 1994.
33. M. Schmidt-Schauß and K. U. Schulz. Solvability of context equations with two context variables is decidable. *J. Symb. Comput.*, 33(1):77–122, 2002.
34. M. Schmidt-Schauß and J. Stuber. On the complexity of linear and stratified context matching problems. *Theory Comput. Systems*, 37:717–740, 2004.
35. F. van Raamsdonk. Translating logic programs into conditional rewriting systems. In L. Naish, editor, *Proc.* ICLP'97, pages 168–182. MIT Press, 1997.
36. M. Villaret. *On Some Variants of Second-Order Unification.* PhD thesis, Universitat Politècnica de Catalunya, Barcelona, 2004.
37. S. Wolfram. *The* MATHEMATICA *Book.* Wolfram Media, 5th edition, 2003.

A Proofs

A.1 Soundness of $\mathfrak{C}$

Lemma 2. *If* $\Gamma_1;\sigma_1 \Longrightarrow_\vartheta \Gamma_2;\sigma_2$ *in* $\mathfrak{R}\setminus\mathfrak{R}_{\mathrm{Sat}}$ *and* $\sigma \in sol(\Gamma_2)$ *then* $\vartheta\sigma \in sol(\Gamma_1)$.

Proof. We prove the lemma for SVHg and CVCtx rules. For the other rules it is either similar or trivial.

SVHg: We have

$$\begin{aligned}
\Gamma_1 &= \{(\overline{x},\tilde{s}) \ll \tilde{t},\ (\overline{x} \text{ in } \overline{y},\ \mathtt{f}_1)\} \cup \Delta,\\
\vartheta &= \{\overline{x} \mapsto \overline{y}\},\\
\Gamma_1\vartheta &= \{(\overline{y},\tilde{s}\vartheta) \ll \tilde{t},\ (\overline{y} \text{ in } \overline{y},\ \mathtt{f}_1)\} \cup \Delta\vartheta,\\
\Gamma_2 &= \{(\overline{y},\overline{z},\tilde{s}\vartheta) \ll \tilde{t},\ (\overline{z} \text{ in } (),\ \mathtt{f}_2)\} \cup \Delta\vartheta.
\end{aligned}$$

Let σ be a solution of Γ_2. Then $\overline{z}\sigma = ()$ and $(\overline{y},\overline{z},\tilde{s}\vartheta)\sigma = (\overline{y},\tilde{s}\vartheta)\sigma$, which implies that σ is a solution of $\Gamma_1\vartheta$ and $\sigma\vartheta$ is a solution of Γ_1.

CVCtx: We have

$$\begin{aligned}
\Gamma_1 &= \{(\overline{C}(\tilde{s}_1),\tilde{s}_2) \ll \tilde{t},\ (\overline{C} \text{ in } \overline{D}(\mathtt{Rc}),\ \mathtt{f}_1)\} \cup \Delta,\\
\vartheta &= \{\overline{C} \mapsto \overline{D}(\overline{E}(\circ))\},\\
\Gamma_1\vartheta &= \{(\overline{D}(\overline{E}(\tilde{s}_1\vartheta)),\tilde{s}_2\vartheta) \ll \tilde{t},\ (\overline{D}(\overline{E}(\circ)) \text{ in } \overline{D}(\mathtt{Rc}\vartheta),\ \mathtt{f}_1)\} \cup \Delta\vartheta,\\
\Gamma_2 &= \{(\overline{D}(\overline{E}(\tilde{s}_1\vartheta)),\tilde{s}_2\vartheta) \ll \tilde{t},\ (\overline{E} \text{ in } \mathtt{Rc}\vartheta,\ \mathtt{f}_2)\} \cup \Delta\vartheta,.
\end{aligned}$$

Let σ be a solution of Γ_2. Then $\overline{E}\sigma \in [\![\mathtt{Rc}\vartheta\sigma]\!]_{\mathtt{f}_2\sigma}$. If $\mathtt{f}_1 = 0$, then $\mathtt{f}_2\sigma = 0$, $\overline{D}(\overline{E}(\circ))\sigma \in [\![\overline{D}(\mathtt{Rc}\vartheta)\sigma]\!]$, and, hence, σ is a solution of $\Gamma_1\vartheta$. If $\mathtt{f}_1 = 1$, then either $\overline{D}\sigma \neq \circ$ and $\mathtt{f}_2\sigma = 0$, or $\overline{D}\sigma = \circ$ and $\mathtt{f}_2\sigma = 1$. In both cases $\overline{D}(\overline{E}(\circ))\sigma \neq \circ$ and, therefore, $\overline{D}(\overline{E}(\circ)\sigma \in [\![\overline{D}(\mathtt{Rc}\vartheta)\sigma]\!]_1$. Hence, also in this case σ is a solution of $\Gamma_1\vartheta$. In implies that $\vartheta\sigma$ is a solution of Γ_1. □

Corollary 1. *If* $\Gamma_1;\sigma_1 \Longrightarrow^+_\vartheta \Gamma_2;\sigma_2$ *in* $\mathfrak{R}\setminus\mathfrak{R}_{\mathrm{Sat}}$ *and* $\sigma \in sol(\Gamma_2)$, *then* $\vartheta\sigma \in sol(\Gamma_1)$.

Theorem 2 (Soundness of $\mathfrak{C}$). *Let* Γ *be a* RWMC *and* $\sigma \in compans_{\mathfrak{C}}(\Gamma)$. *Then there exists* $\vartheta \in sol(\Gamma)$ *such that* $\sigma \leq \vartheta$.

Proof. Let $\Gamma;\emptyset \Longrightarrow^{+}_{\sigma} \emptyset;\sigma$ be the derivation that computes σ. First assume that $\mathfrak{R}_{\mathrm{Sat}}$ is not used in this derivation. Since $\varepsilon \in sol(\emptyset)$, by Corollary 1 we get that $\sigma\varepsilon = \sigma \in sol(\Gamma)$. If $\mathfrak{R}_{\mathrm{Sat}}$ is used, the derivation has a form $\Gamma;\emptyset \Longrightarrow^{+}_{\sigma} \Delta;\sigma \Longrightarrow_{\mathsf{MC},\varepsilon} \emptyset;\sigma$. By Corollary 1, if $\varphi \in sol(\Delta)$ then $\sigma\varphi \in sol(\Gamma)$. It remains to show that such a φ exists. Since Δ is well-moded and constrains variables maximum once, there exists $(\overline{v}\ \mathsf{in}\ \mathtt{Re},\ 0) \in \Delta$ such that $\mathtt{Re}$ does not contain any constrained variables. Since $[\![\mathtt{Re}]\!]$ is nonempty, take arbitrary $e \in [\![\mathtt{Re}]\!]$ and define $\varphi_1 = \{\overline{v} \mapsto e\}$. The constraint $(\Delta \setminus \{\overline{v}\ \mathsf{in}\ \mathtt{Re},\ 0\})\varphi_1$ has all the properties of Δ, therefore we can proceed in the similar way. This process ends after n steps, where n is the number of elements in Δ, and $\varphi_1 \cdots \varphi_n \in sol(\Delta)$. □

A.2 Termination of $\mathfrak{C}$

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

The *size of a hedge* $\tilde{t}$, denoted $tsize(\tilde{t})$, is defined as the number of symbols in $\tilde{t}$ if $\tilde{t}$ is ground, and ∞ if $\tilde{t}$ contains variables. $tsize(()) = 0$. It is assumed that $\infty > n$ for any nonnegative integer n.

The *size of a regular expression* $\mathtt{Re}$, denoted $rsize(\mathtt{Re})$, is the number of symbols in $\mathtt{Re}$.

Let $\overline{v} \in \mathcal{V}_{\mathrm{Seq}} \cup \mathcal{V}_{\mathrm{Con}}$ and $(\overline{v}\ \mathsf{in}\ \mathtt{Re}, \mathtt{f}) \in \Gamma$ for an RWMC constraint Γ. We associate to $\overline{v}$ its *regular measure* with respect to Γ, which is denoted by $rm(\overline{v},\Gamma)$ and is defined as a multiset of pairs: $rm(\overline{v},\Gamma) = \{rsize(\mathtt{Re})\} \dot\cup \dot\bigcup_{\overline{u}\in vars(\mathtt{Re})} rm(\overline{u},\Gamma)$, where the operation $\dot\cup$ denotes multiset union. Since Γ is well-moded, rm is well-defined.

A *position* is a sequence of positive integers. For a hedge $\tilde{s}$ and a position p, $symb(\tilde{s},p)$ denotes the *symbol of* $\tilde{s}$ *at position* p*:*

- $symb(\tilde{s},()) = \tilde{s}$ if $\tilde{s} \in \mathcal{V}_{\mathrm{Ind}} \cup \mathcal{V}_{\mathrm{Seq}} \cup \{\circ\}$.
- $symb(\mathsf{h}(\tilde{t}),()) = \mathsf{h}$ where $\mathsf{h} \in \mathcal{F} \cup \mathcal{V}_{\mathrm{Fun}} \cup \mathcal{V}_{\mathrm{Con}}$.
- $symb(\mathsf{h}(\tilde{t}),(i,i_1,\ldots,i_n)) = symb(\tilde{t},(i,i_1,\ldots,i_n))$ where $\mathsf{h} \in \mathcal{F} \cup \mathcal{V}_{\mathrm{Fun}} \cup \mathcal{V}_{\mathrm{Con}}$.
- $symb((t_1,\ldots,t_m),(i,i_1,\ldots,i_n)) = symb(t_i,(i_1,\ldots,i_n))$, where $1 \le i \le m$.

In all other cases $symb(\tilde{s},p)$ is undefined. Positions are ordered with the ordering $\succ$ that is the 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.

Let $\tilde{s}$ be a hedge and Γ be an RWMC constraint. We say that a position p in $\tilde{s}$ is a *trivial position* with respect to Γ if $symb(\tilde{s},p) = \overline{v} \in \mathcal{V}_{\mathrm{Seq}} \cup \mathcal{V}_{\mathrm{Con}}$ and $(\overline{v}\ \mathsf{in}\ \mathtt{Re},\ 0) \in \Gamma$. Otherwise the position is a *nontrivial position*. The $\succ$-minimal nontrivial position in $\tilde{s}$ with respect to Γ, if exists, is denoted by $\min_{\succ}(\tilde{s},\Gamma)$. Otherwise $\min_{\succ}(\tilde{s},\Gamma) = \infty$.

The *constrained variable prefix* of a hedge $\tilde{s}$ with respect to a RWMC constraint Γ, denoted $cvp(\tilde{s},\Gamma)$, is the multiset union of all $rm(\overline{v},\Gamma)$'s such that $\overline{v}$ is a constrained variable occurring in a position p with $\min_{\succ}(\tilde{s},\Gamma) \succeq p$.[5]

[5] In fact, in this definition we need only the membership part of Γ.

Example 7. Let the membership part of Γ be

$$\{(\overline{x} \text{ in } (), 0),\ (\overline{C} \text{ in } (f(\overline{y} \mid \overline{x}, \circ \mid f(\circ))^\star, b), 0),\ (\overline{y} \text{ in } (c, d)^*, 1),\ (\overline{z} \text{ in } a^*, 0)\}.$$

1. Let $\tilde{s}$ be $(\overline{x}, \overline{C}(f(\overline{x}), \overline{y}), \overline{z})$. Then $\min_{\succ}(\tilde{s}, \Gamma) = (2, 1)$ and

$$cvp(\tilde{s}, \Gamma) = rm(\overline{x}, \Gamma) \dot\cup rm(\overline{C}, \Gamma) = \{0\} \dot\cup \{12, 4, 0\} = \{0, 0, 4, 12\}.$$

2. Let $\tilde{s}$ be $(\overline{x}, \overline{C}(\overline{x}, \overline{y}), \overline{z})$. Then $\min_{\succ}(\tilde{s}, \Gamma) = (2, 2)$ and

$$cvp(\tilde{s}, \Gamma) = rm(\overline{x}, \Gamma) \dot\cup rm(\overline{C}, \Gamma) \dot\cup rm(\overline{x}, \Gamma) \dot\cup rm(\overline{y}, \Gamma) = \{0, 12, 4, 0, 0, 4\}.$$

To each RWMC constraint Γ we associate a *complexity measure,* $cm(\Gamma)$, as a tuple $\langle n_1, n_2, n_3, n_4 \rangle$, where n_1 and n_4 are nonnegative integers, n_2 is a multiset of positive integers and ∞, and n_3 is a multiset of nonnegative integers:

$$\begin{aligned}
n_1 &= \text{the number of distinct unconstrained variables in } \Gamma.\\
n_2 &= \dot\bigcup_{\tilde{s} \ll \tilde{t} \in \Gamma} \{tsize(\tilde{t})\}.\\
n_3 &= \dot\bigcup_{\tilde{s} \ll \tilde{t} \in \Gamma} cvp(\tilde{s}, \Gamma).\\
n_4 &= \text{the number of symbols in } \Gamma.
\end{aligned}$$

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

Theorem 3 (Termination of $\mathfrak{C}$). *$\mathfrak{C}$ terminates on any input either with $\bot$ or with a system $\emptyset; \sigma$.*

Proof. Termination of $\mathfrak{C}$ follows from the fact that every rule R in $\mathfrak{R}$ strictly decreases the complexity measure: If $\Gamma_1; \sigma_1 \Longrightarrow \Gamma_2; \sigma_2$ then $cm(\Gamma_1) > cm(\Gamma_2)$, and $cm(\Gamma) > cm(\bot)$ for any Γ. Table A.2 shows which rule in $\mathfrak{R}$ decreases which component of the regular complexity measure.

Furthermore, any RWMC constraint must fit into one of the cases mentioned on the left hand sides of the rules in $\mathfrak{R}$, so that a rule can always be applied to a system with non-empty constraint. (Recall that we implicitly assume the presence of failure rules that have not been given here to save space.) Therefore, a system to which no rule applies is either $\bot$ or has a form $\emptyset; \sigma$. □

Rule	n_1	n_2	n_3
T, Dec, D	=	>	
IVE, FVE, SVE, F	>		

Rule	n_1	n_2	n_3	n_4
$\mathfrak{R}_{\mathrm{Hg}}$, $\mathfrak{R}_{\mathrm{Ctx}}$	=	=	>	
MC	=	=	=	>

Table 1. Rules in $\mathfrak{R}$ on the regular complexity measure. The equality sign = means the component remains unchanged, > means it strictly decreases.

A.3 Completeness of $\mathfrak{C}$

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

Lemma 3. *Let Γ be a* RWMC *problem. Then $v\sigma$ is ground for every $v \in eqv(\Gamma)$ and $\sigma \in compans_{\mathfrak{C}}(\Gamma)$.*

Theorem 4 (Completeness of $\mathfrak{C}$). *Let Γ be a* RWMC*, $\vartheta \in sol(\Gamma)$, $Q = eqv(\Gamma)$, and $V = vars(\Gamma)$. Then there exists $\sigma \in compans_{\mathfrak{C}}(\Gamma)$ such that $\sigma|_Q = \vartheta|_Q$ and $\sigma \leq^V \vartheta$.*

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

We assume that Γ contains equations, otherwise the theorem is trivial. We pick an arbitrary equation $\tilde{s} \ll \tilde{t}$ from Γ such that $\tilde{t}$ is ground, and represent Γ as $\{\tilde{s} \ll \tilde{t}\} \cup \Delta$. If $\tilde{s} = \tilde{t}$ the theorem is straightforward. Assume $\tilde{s} \neq \tilde{t}$. We consider below only nontrivial cases.

Let $\tilde{s}$ be $(\overline{C}(\tilde{s}_1), \tilde{s}_2)$ such that $\overline{C}$ is not constrained. Obviously, $\overline{C}\vartheta$ must be a ground context, say C. Then we transform $\Gamma; \emptyset$ with $\Gamma; \emptyset \Longrightarrow_{\mathsf{R}} \Gamma_1; \sigma_1$, where Γ_1, σ_1, and R are defined as follows: If $C = (\tilde{t}_1, \circ, \tilde{t}_3)$ and $\tilde{t} = (\tilde{t}_1, \tilde{t}_2, \tilde{t}_3, \tilde{t}_4)$ then $\sigma_1 = \{\overline{C} \mapsto (\tilde{t}_1, \circ, \tilde{t}_3)\}$, $\Gamma_1 = \{\tilde{s}_1\sigma_1 \ll \tilde{t}_2, \tilde{s}_2\sigma_1 \ll \tilde{t}_4\} \cup \Delta\sigma_1$, and $\mathsf{R} = \mathsf{F}$. If $C \neq (\tilde{t}_1, \circ, \tilde{t}_3)$ then C should be $(\tilde{t}_1, f(D), \tilde{t}_2)$, where D is a context and $\tilde{t} = (\tilde{t}_1, f(\tilde{r}), \tilde{t}_2, \tilde{t}_3)$. Then $\sigma_1 = \{\overline{C} \mapsto (\tilde{t}_1, f(\overline{D}(\circ)), \tilde{t}_2)\}$, $\Gamma_1 = \{\overline{D}(\tilde{s}_1)\sigma_1 \ll \tilde{r},\ \tilde{s}_2\sigma_1 \ll \tilde{t}\} \cup \Delta\sigma_1$, and $\mathsf{R} = \mathsf{D}$. In either case $cm(\Gamma) > cm(\Gamma_1)$, there exists φ such that $\sigma_1\varphi = \vartheta$, and $\varphi \in sol(\Gamma_1)$. By the induction hypothesis there exists a derivation $\Gamma_1; \emptyset \Longrightarrow^+ \emptyset; \psi$ with $\psi|_{Q_1} = \varphi|_{Q_1}$ and $\psi \leq^{V_1} \varphi$ where $Q_1 = eqv(\Gamma_1)$ and $V_1 = vars(\Gamma_1)$. Moreover, we have $(Q_1 \cup \{\overline{C}\}) \setminus \{\overline{D}\} = Q$, $(V_1 \cup \{\overline{C}\}) \setminus \{\overline{D}\} = V$, and $(\sigma_1\psi)|_Q = (\sigma_1\varphi)|_Q = \vartheta|_Q$ and $(\sigma_1\psi) \leq^V (\sigma_1\varphi) = \vartheta|_V$. Hence, we obtain the desired derivation $\Gamma; \emptyset \Longrightarrow_{\mathsf{R}} \Gamma_1; \sigma_1 \Longrightarrow^+ \emptyset; \sigma_1\psi$.

Now assume $\tilde{s}$ is $f(\overline{C}(\tilde{s}_1), \tilde{s}_2)$ with $\overline{C}$ constrained with exactly one membership constraint $\overline{C}$ in $\mathtt{Rc}$. Let Δ_0 be $\Delta \setminus \{\overline{C}\ \mathsf{in}\ \mathtt{Rc},\ \mathtt{f}\}$. Obviously, $\overline{C}\vartheta$ must be a ground context, say C. Depending on the form of $\mathtt{Rc}$ we have different cases. Here we consider only the case $\mathtt{Rc} = \mathtt{Rc}_1^\star$:

We transform $\Gamma; \emptyset$ with $\Gamma; \emptyset \Longrightarrow_{\mathsf{R}} \Gamma_1; \sigma_1$, where Γ_1, σ_1, and R are defined as follows: If $C = \circ$ then $\sigma_1 = \{\overline{C} \mapsto \circ\}$, $\Gamma_1 = \{f(\tilde{s}_1, \tilde{s}_2)\sigma_1 \ll t\} \cup \Delta_0\sigma_1$, and $\mathsf{R} = \mathsf{RCtx1}$. In this case $\mathtt{f} = 0$. If $C \neq \circ$ then $\sigma_1 = \{\overline{C} \mapsto \overline{D}(\overline{E}(\circ))\}$, $\Gamma_1 = \{(\overline{C}(\tilde{s}_1\sigma_1), \tilde{s}_2\sigma_1) \ll t,\ (\overline{D}\ \mathsf{in}\ \mathtt{Rc}_1,\ 1),\ (\overline{E}\ \mathsf{in}\ \mathtt{Rc}_1^\star,\ 0)\} \cup \Delta_0\sigma_1$, and $\mathsf{R} = \mathsf{RCtx2}$. In either case $cm(\Gamma) > cm(\Gamma_1)$ and there exists φ such that $\sigma_1\varphi = \vartheta$ and φ is a solution of Γ_1. By the induction hypothesis there exists a derivation $\Gamma_1; \emptyset \Longrightarrow^+ \emptyset; \psi$ with $\psi|_{Q_1} = \varphi|_{Q_1}$ and $\psi \leq^{V_1} \varphi$ where $Q_1 = eqv(\Gamma_1)$ and $V_1 = vars(\Gamma_1)$. Moreover, we have $(Q_1 \cup \{\overline{C}\}) \setminus \{\overline{D}, \overline{E}\} = Q$, $(V_1 \cup \{\overline{C}\}) \setminus \{\overline{D}, \overline{E}\} = V$, and $(\sigma_1\psi)|_Q = (\sigma_1\varphi)|_Q = \vartheta|_Q$ and $\sigma_1\psi \leq^{V_1} \sigma_1\varphi = \vartheta|_V$. Hence, we obtain the derivation $\Gamma; \emptyset \Longrightarrow_{\mathsf{R}} \Gamma_1; \sigma_1 \Longrightarrow^+ \emptyset; \sigma_1\psi$ with the desired properties.

Finally, assume $\tilde{s}$ is $(\overline{C}(\tilde{s}_1), \tilde{s}_2)$ with $\overline{C}$ constrained with the membership constraints $(\overline{C}$ in $\mathtt{Rc}_1,\ fl_1)$ and $(\overline{C}$ in $\mathtt{Rc}_2,\ \mathtt{f}_2)$. Let Δ_0 be $\Delta \setminus \{(\overline{C}$ in $\mathtt{Rc}_1,\ \mathtt{f}_1),$ $(\overline{C}$ in $\mathtt{Rc}_2,\ \mathtt{f}_2)\}$. We have $\overline{C}\vartheta = C$ for some ground C. We transform $\Gamma; \emptyset$ with the step $\Gamma; \emptyset \Longrightarrow_{\mathsf{ICtx}} \Gamma_1; \emptyset$ where $\Gamma_1 = \{(\overline{C}(\tilde{s}_1), \tilde{s}_2) \ll \tilde{t},\ (\overline{C}$ in $\mathtt{Rc}_1,\ \mathtt{f}_1),\ \overline{D}(a) \ll \overline{C}(a), (\overline{D}$ in $\mathtt{Rc}_2,\ \mathtt{f}_2)\} \cup \Delta_0$. Since $\vartheta \in sol(\Gamma)$, the substitution $\vartheta\sigma_1 \in sol(\Gamma_1)$, where $\sigma_1 = \{\overline{D} \mapsto C\}$. We assume without loss of generality that $\overline{D}$ does not occur in ϑ. By the induction hypothesis, there exists a derivation $\Gamma_1; \varepsilon \Longrightarrow^+ \emptyset; \psi$ with $\psi|_{Q_1} = \vartheta\sigma_1|_{Q_1}$ and $\psi \leq^{V_1} \vartheta\sigma_1$ where $Q_1 = eqv(\Gamma_1)$ and $V_1 = vars(\Gamma_1)$. Since $Q \subset Q_1$ and $\overline{D}$ does not occur in ϑ, we have $\psi|_Q = \vartheta\sigma_1|_Q = \vartheta|_Q$. As for the case with restriction to V, since $\overline{D} \in eqv(\Gamma_1)$, by Lemma 3, $\overline{D}\psi|_{V_1}$ is ground. Then the only possibility is $\overline{D}\psi|_{V_1} = C$, because $\overline{D}\vartheta\sigma_1|_{V_1} = C$. Moreover, $\overline{D}$ does not occur in the range of $\psi|_V$. Hence, for all $v \in dom(\psi|_V)$ we have $v\psi|_V = v\psi|_{V_1}$. Similarly, since $\overline{D}$ does not occur in ϑ, for all $v \in dom(\vartheta\sigma_1|_V)$ we have $v\vartheta\sigma_1|_V = v\vartheta\sigma_1|_{V_1}$. It implies that $\psi|_V \leq \vartheta\sigma_1|_V$. But since $\vartheta\sigma_1|_V = \vartheta|_V$ we finally get $\psi \leq^V \vartheta$. □