Matching in Flat Theories*

Temur Kutsia

Research Institute for Symbolic Computation
Johannes Kepler University, A-4040, Linz, Austria
kutsia@risc.uni-linz.ac.at

Abstract. Flat theory with sequence variables and flexible arity symbols has a decidable infinitary matching and unification. We briefly describe a minimal complete flat matching procedure and discuss its relations with the flat matching implemented in the MATHEMATICA system.

1 Preliminaries

Sequence variables can be instantiated with a finite, possible empty, sequence of terms. We use $\overline{x}, \overline{y}, \ldots$ to denote them. We build terms over individual and sequence variables, and fixed and flexible arity function symbols in the usual way, with the only restriction that sequence variables are not allowed to be direct arguments of fixed arity symbols. Similarly, in equalities over terms, sequence variables can not be the direct arguments of the equality symbol. For instance, for a binary g and flexible arity f, $f(\overline{x}, g(x, y)) \simeq f(\overline{y})$ is an equation, while $f(\overline{x}, g(x, \overline{y})) \simeq f(\overline{y})$ and $\overline{x} \simeq g(x, y)$ are not.

Let $\mathcal{F}_{\mathrm{Fix}}$ be a set of fixed arity function symbols and $\mathcal{F}_{\mathrm{Flex}}$ be a set of flexible arity function symbols. We write $\mathcal{T}(\mathcal{F}_{\mathrm{Fix}} \cup \mathcal{F}_{\mathrm{Flex}}, \mathcal{V}_{\mathrm{Ind}} \cup \mathcal{V}_{\mathrm{Seq}})$ for the set of terms over the signature $\mathcal{F}_{\mathrm{Fix}} \cup \mathcal{F}_{\mathrm{Flex}}$, a set $\mathcal{V}_{\mathrm{Ind}}$ of individual variables, and a set $\mathcal{V}_{\mathrm{Seq}}$ of sequence variables.

We assume that the reader is familiar with the standard notions of unification theory [2]. Here we mention only non-straightforward generalizations for sequence variables and flexible arity symbols.

Definition 1. *A* binding *is either a pair $x \leftarrow s$ where $x \in \mathcal{V}_{\mathrm{Ind}}$ and $s \in \mathcal{T}(\mathcal{F}_{\mathrm{Fix}} \cup \mathcal{F}_{\mathrm{Flex}}, \mathcal{V}_{\mathrm{Ind}} \cup \mathcal{V}_{\mathrm{Seq}}) \setminus (\{x\} \cup \mathcal{V}_{\mathrm{Seq}})$, or an expression $\overline{x} \leftarrow s_1, \ldots, s_n$ where $\overline{x} \in \mathcal{V}_{\mathrm{Seq}}$ and $s_1, \ldots, s_n$ is a (possibly empty) sequence of terms such that $s_1 \neq \overline{x}$ if $n = 1$.*

A substitution *is a finite set of bindings $\{x_1 \leftarrow s_1, \ldots, x_n \leftarrow s_n, \overline{x}_1 \leftarrow t^1_1, \ldots, t^1_{k_1}, \ldots, \overline{x}_m \leftarrow t^m_1, \ldots, t^m_{k_m}\}$ with $x_1, \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_n$ distinct variables.*

Greek letters denote substitutions, with ε for the empty substitution. The instance of a term $\overline{x}$ w.r.t. a substitution θ, $\overline{x}\theta$, is the sequence $s_1, \ldots, s_n$, if $\overline{x} \leftarrow s_1, \ldots, s_n \in \theta$, and $\overline{x}$ otherwise. The notion of instance for terms other than sequence variables is defined in the usual way.

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

Definition 2. *A substitution θ is* more general *than σ on a finite set of variables V modulo a theory E ($\theta \preceq_E^V \sigma$) iff there exists a substitution λ such that*

- *for all $\overline{x} \in V$, the binding $\overline{x} \leftarrow$ does not belong to λ; there exist terms $t_1, \ldots, t_n, s_1, \ldots, s_n$, $n \geq 0$, such that $\overline{x}\sigma = t_1, \ldots, t_n$, $\overline{x}\theta\lambda = s_1, \ldots, s_n$, and for each $1 \leq i \leq n$, either t_i and s_i are the same sequence variables, or $t_i \simeq_E s_i$;*
- *for all $x \in V$, $x\sigma \simeq_E x\theta\lambda$.*

Example 1. $\{\overline{x} \leftarrow \overline{y}\} \preceq_\emptyset^{\{\overline{x},\overline{y}\}} \{\overline{x} \leftarrow a, b, \quad \overline{y} \leftarrow a, b\}$, but not $\{\overline{x} \leftarrow \overline{y}\} \preceq_\emptyset^{\{\overline{x},\overline{y}\}} \{\overline{x} \leftarrow, \overline{y} \leftarrow\}$.

Flat theory, or shortly F-theory, is defined as $E = \{f(\overline{x}, f(\overline{y}), \overline{z}) \simeq f(\overline{x}, \overline{y}, \overline{z})\}$; $f \in \mathcal{F}_{\mathrm{Flex}}$ is called a flat flexible arity symbol. It should be noted that although (free or flat) unification with sequence variables and flexible arity symbols looks similar to A-unification, there are essential differences illustrated by the following example (even without sequence variables). Let $f(x, f(y, z)) \simeq^? f(f(a, b), c)$ be a unification problem, where x, y, z are individual variables, and a, b, c are constants. For associative f the minimal complete set of solutions is the singleton $\{\{x \leftarrow a, y \leftarrow b, z \leftarrow c\}\}$; for free flexible arity symbol f the problem has no solution; and for flat flexible arity f there are 23 substitutions in the minimal complete set of solutions: $\{\{x \leftarrow f(), y \leftarrow f(), z \leftarrow f(a, b, c)\}$, $\{x \leftarrow f(), y \leftarrow a, z \leftarrow f(a, b, c)\}$, $\{x \leftarrow f(), y \leftarrow f(a), z \leftarrow f(a, b, c)\}$, $\{x \leftarrow f(), y \leftarrow f(a, b), z \leftarrow c\}$, $\{x \leftarrow f(), y \leftarrow f(a, b), z \leftarrow f(c)\}$, $\{x \leftarrow f(), y \leftarrow f(a, b, c), z \leftarrow f()\}$, $\{x \leftarrow a, y \leftarrow f(), z \leftarrow f(b, c)\}$, $\{x \leftarrow f(a), y \leftarrow f(), z \leftarrow f(b, c)\}$, $\{x \leftarrow a, y \leftarrow b, z \leftarrow c\}$, $\{x \leftarrow a, y \leftarrow f(b), z \leftarrow c\}$, $\{x \leftarrow a, y \leftarrow b, z \leftarrow f(c)\}$, $\{x \leftarrow a, y \leftarrow f(b), z \leftarrow f(c)\}$, $\{x \leftarrow f(a), y \leftarrow b, z \leftarrow c\}$, $\{x \leftarrow f(a), y \leftarrow f(b), z \leftarrow c\}$, $\{x \leftarrow f(a), y \leftarrow b, z \leftarrow f(c)\}$, $\{x \leftarrow f(a), y \leftarrow f(b), z \leftarrow f(c)\}$, $\{x \leftarrow a, y \leftarrow f(b, c), z \leftarrow f()\}$, $\{x \leftarrow f(a), y \leftarrow f(b, c), z \leftarrow f()\}$, $\{x \leftarrow f(a, b), y \leftarrow f(), z \leftarrow c\}$, $\{x \leftarrow f(a, b), y \leftarrow f(), z \leftarrow f(c)\}$, $\{x \leftarrow f(a, b), y \leftarrow c, z \leftarrow f()\}$, $\{x \leftarrow f(a, b), y \leftarrow f(c), z \leftarrow f()\}$, $\{x \leftarrow f(a, b, c), y \leftarrow f(), z \leftarrow f()\}\}$.

Below we consider general F-unification and F-matching, i.e., besides flat flexible arity symbols we have also free fixed and free flexible arity symbols.

2 Unification and Matching

General flat unification is decidable. It can be proved using the Baader-Schulz method [1], reducing a flat unification solvability problem to a combination of solvability problems of word equations (with certain additional restrictions) and syntactic unification. For the details we refer to [4].

Flat unification is infinitary. Unification procedure is designed as a tree generation process, in the breadth-first manner. Each node of the tree is labeled either with a unification problem (kept in flattened form), with $\top$, or with $\perp$. The root node is labeled with the unification problem to be solved. $\top$ and $\perp$ are terminal nodes. Before expanding a non-terminal node, we first check whether the problem attached to the node is solvable. Children of the root are obtained by projection (elimination of all possible subsets of the set of sequence variables on the root note). Children of other non-terminal nodes are obtained by transformation rules. Due to a lack of space we do not describe those rules here, neither more details about the procedure. They can be found in [4]. Unifiers are constructed by composing substitutions on the edges of branches with the terminal node $\top$. The procedure enumerates minimal complete set of unifiers.

We give a bit more details about general F-matching, since it has some interesting properties and applications. Decidability and the existence of a minimal set of matchers for a general F-matching problem follows from the corresponding properties of general F-unification. Interestingly, it turns out that F-matching is infinitary, as the following simple example illustrates:

Example 2. The minimal complete set of solutions for the F-matching problem $f(\overline{x}) \ll^?_F f(a)$ with flat f is $\{\{\overline{x} \leftarrow a\}, \{\overline{x} \leftarrow f(a)\}, \{\overline{x} \leftarrow a, f()\}, \{\overline{x} \leftarrow f(a), f()\}, \{\overline{x} \leftarrow f(), a\}, \{\overline{x} \leftarrow f(), f(a)\}, \{\overline{x} \leftarrow f(), a, f()\}, \ldots\}$.

F-matching procedure is designed like the one for F-unification – as a tree generation process. It enumerates the minimal complete set of matchers[1]. Transformation rules for F-matching are given in Appendix. Each of them has one of the following forms: $\mathfrak{M} \rightsquigarrow \perp$ or $\mathfrak{M} \rightsquigarrow \langle\langle \mathfrak{M}_1, \sigma_1 \rangle, \ldots \langle \mathfrak{M}_n, \sigma_n \rangle\rangle$, where $\mathfrak{M}$ is a matching problem, each of the successors $\mathfrak{M}_i$ is either $\top$ or a new matching problem, and each σ_i is a substitution used to generate $\mathfrak{M}_i$ from $\mathfrak{M}$.

The rules involving $f()$ in the substitutions are the reason why F-matching is infinitary. However, they are indispensable for completeness, as the following examples illustrate:

Example 3. The unique solution $\{x \leftarrow f()\}$ of $f(x, a) \ll^?_F f(a)$ with flat f can not be computed unless the transformation substitution $\{x \leftarrow f()\}$ is allowed.

Example 4. The transformation rule $\{\overline{x} \leftarrow f(), \overline{x}\}$ is crucial for computing a unifier $\{\overline{x} \leftarrow f(), a\}$ of $f(\overline{x}, g(\overline{x})) \ll^?_F f(a, g(f(), a))$, where f is flat and g is free. The derivation is shown on Fig. 1. Composing substitutions top-down on the edges gives the solution.

3 Flat Matching in Mathematica

Mathematica system [6] implements matching modulo flatness. The algorithm itself, to our knowledge, is nowhere described, but is briefly explained on examples in [6][2]. However, it is not hard to observe that the algorithm is not complete. It does not match, for instance, $f(x, a)$ to $f(a)$, $f(x, g(x))$ to $f(a, g(a))$, or $f(\overline{x}, g(\overline{x}))$ to $f(a, g(f(a)))$, where f is flat and g is free.

[1] To ensure minimality, during the tree generation process an additional effort is needed each time when a new solution appears: to delete from the already computed (finite) set of solutions those substitutions which violate minimality condition.

[2] In fact, this is the case not only with this particular algorithm, but, in general, with the evaluation semantics of the programming language of Mathematica. See [3] for more discussion on this topic

$$f(\overline{x}, g(\overline{x})) \ll^?_F f(a, g(f(), a))$$

$$\Big\downarrow \{\overline{x} \leftarrow f(), \overline{x}\}$$

$$f(\overline{x}, g(f(), \overline{x})) \ll^?_F f(a, g(f(), a))$$

$$\Big\downarrow \{\overline{x} \leftarrow a\}$$

$$f(a, g(f(), a)) \ll^?_F f(a, g(f(), a))$$

$$\Big| \ \varepsilon$$

$$\top$$

Fig. 1. Computing a solution for $f(\overline{x}, g(\overline{x})) \ll^?_F f(a, g(f(), a))$.

The main difference between the F-matching procedure and the Mathematica flat matching is that the latter does not consider transformation rules involving $f()$. It makes Mathematica flat matching finitary.

Example 5. Let $f(\overline{x} \ll^?_F f()$ be a matching problem, where f is flat. The F-matching procedure enumerates the infinite minimal complete set of matchers $\{\{\overline{x} \leftarrow \}, \{\overline{x} \leftarrow f()\}, \{\overline{x} \leftarrow f(), f()\}, \ldots\}$. Mathematica flat matching algorithm returns a single solution $\{\overline{x} \leftarrow \}\}$.

Another difference is in the case where an individual variable x matches a single argument s_1 in a term with a flat head f. The F-matching procedure returns four substitutions as it is shown in the third case of Eliminate on Fig. 2, while the Mathematica matching algorithm chooses only the last two of those four. If in the same situation we have a sequence variable $\overline{x}$, the F-matching procedure tries 9 different ways to resolve the case (the fourth rule of Eliminate on Fig. 2), while Mathematica would choose only the second and sixth.

On the other hand, Mathematica can verify that each solution computed by the F-matching procedure is correct, e.g., it sees $f(x, g(x))\{x \leftarrow a\}$ and $f(a, g(a))$ as identical expressions, although, as it was already mentioned, the Mathematica matching algorithm can not compute the substitution $\{x \leftarrow a\}$ that matches $f(x, g(x))$ to $f(a, g(a))$.

To summarize the similarities and differences between the F-matching and flat matching in Mathematica, on Fig. 4 we give rules that can simulate the behavior of Mathematica flat matching. Note that in the tree generation process we do not need to check solvability at the nodes anymore. For a given matching problem, the output of the procedure involving the rules on Fig. 4 would be identical to the set of all possible matchers Mathematica matching algorithm computes (One can see all the Mathematica matchers using the function `ReplaceList`, for instance.).

However, when MATHEMATICA tries to match patterns in the left hand side of its assignments or rules to some expression, from all possible matchers it selects the first one it finds[3]. We can simulate also this behavior, imposing an order of choosing successors in the `EliminateM` step and in the projection phase, and stopping the development of the tree whenever the first solution appears.

4 Conclusion

We considered flat theories with sequence variables and flexible arity symbols. The interesting feature of such a theory is that it has decidable infinitary general matching and unification. We gave a sketch of a procedure that enumerates a minimal complete set of solutions and discussed a relation of the matching procedure with the pattern matching algorithm of MATHEMATICA.

5 Acknowledgements

I am grateful to Prof. Bruno Buchberger who supervised this work, and to Mircea Marin with whom I had useful discussions.

References

1. F. Baader and K. U. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. *Journal of Symbolic Computation*, 21(2):211–244, 1996.
2. F. Baader and W. Snyder. Unification theory. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 8, pages 445–532. Elsevier Science, 2001.
3. B. Buchberger. Mathematica as a rewrite language. In T. Ida, A. Ohori, and M. Takeichi, editors, *Proceedings of the Second Fuji International Workshop on Functional and Logic Programming*, pages 1–13, Shonan Village Center, Japan, 1–4 November 1996. World Scientific.
4. T. Kutsia. Solving and proving in equational theories with sequence variables and flexible arity symbols. Technical Report 02-09, RISC, Linz, Austria, 2002.
5. M. Marin. Introducing sequentica, 2002. http://www.score.is.tsukuba.ac.jp/~mmarin/Sequentica/.
6. S. Wolfram. *The Mathematica Book*. Cambridge University Press and Wolfram Research, Inc., fourth edition, 1999.

[3] The Sequentica package [5] allows the user to gain more control on the selection of matchers.

A Transformation Rules

$$
\begin{array}{lll}
\textsf{Success:} & t \ll_{\emptyset}^{?} t \rightsquigarrow \langle\langle \top,\ \varepsilon\rangle\rangle. & \\
 & x \ll_{F}^{?} t \rightsquigarrow \langle\langle \top,\ \{x \leftarrow t\}\rangle\rangle. & \\
\textsf{Failure:} & f() \ll_{F}^{?} f(t_1, \tilde{t}) \rightsquigarrow \bot. & \\
 & f(t_1, \tilde{t}) \ll_{F}^{?} f() \rightsquigarrow \bot, & \text{if } t_1 \notin \mathcal{V}_{\mathrm{Ind}} \cup \mathcal{V}_{\mathrm{Seq}}. \\
 & f(t_1, \tilde{t}) \ll_{F}^{?} f(s_1, \tilde{s}) \rightsquigarrow \bot, & \text{if } t_1 \ll_{F}^{?} s_1 \rightsquigarrow \bot. \\
 & c_1 \ll_{F}^{?} c_2 \rightsquigarrow \bot, & \text{if } c_1 \neq c_2. \\
 & h_1(\tilde{t}) \ll_{F}^{?} h_2(\tilde{s}) \rightsquigarrow \bot, & \text{if } h_1 \neq h_2. \\
 & h() \ll_{F}^{?} h(t_1, \tilde{t}) \rightsquigarrow \bot. & \\
 & h(t_1, \tilde{t}) \ll_{F}^{?} h() \rightsquigarrow \bot. & \\
 & h(t_1, \tilde{t}) \ll_{F}^{?} h(s_1, \tilde{s}) \rightsquigarrow \bot, & \text{if } t_1 \ll_{F}^{?} s_1 \rightsquigarrow \bot. \\
\textsf{Split:} & h_1(t_1, \tilde{t}) \ll_{F}^{?} h_1(s_1, \tilde{s}) \rightsquigarrow & \text{if } t_1 \notin \mathcal{V}_{\mathrm{Ind}} \cup \mathcal{V}_{\mathrm{Seq}} \text{ and} \\
 & \quad \langle\langle h_1(r_1, \tilde{t}\sigma_1) \ll_{F}^{?} h_1(q_1, \tilde{s}),\ \sigma_1\rangle, & t_1 \ll_{F}^{?} s_1 \rightsquigarrow \langle\langle r_1 \ll_{F}^{?} q_1,\ \sigma_1\rangle, \\
 & \quad \ldots, \langle h_1(r_k, \tilde{t}\sigma_k) \ll_{F}^{?} h_1(q_k, \tilde{s}),\ \sigma_k\rangle\rangle & \quad \ldots, \langle r_k \ll_{F}^{?} q_k,\ \sigma_k\rangle\rangle. \\
\textsf{Eliminate:} & f(x, \tilde{t}) \ll_{F}^{?} f() \rightsquigarrow \langle\langle f(\tilde{t}\sigma) \ll_{F}^{?} f(), \sigma\rangle\rangle, & \text{where } \sigma = \{x \leftarrow f()\}. \\
 & f(\overline{x}, \tilde{t}) \ll_{F}^{?} f() \rightsquigarrow \langle\langle f(\tilde{t}\sigma_1) \ll_{F}^{?} f(), \sigma_1\rangle\rangle, & \text{where } \sigma_1 = \{\overline{x} \leftarrow f()\}, \\
 & \quad \langle\langle f(\overline{x}, \tilde{t}\sigma_2) \ll_{F}^{?} f(), \sigma_2\rangle\rangle, & \quad \sigma_2 = \{\overline{x} \leftarrow f(), \overline{x}\}. \\
 & f(x, \tilde{t}) \ll_{F}^{?} f(s_1, \tilde{s}) \rightsquigarrow \langle\langle f(\tilde{t}\sigma_1) \ll_{F}^{?} f(s_1, \tilde{s}),\ \sigma_1\rangle, & \text{where } \sigma_1 = \{x \leftarrow f()\}, \\
 & \quad \langle f(\tilde{t}\sigma_2) \ll_{F}^{?} f(\tilde{s}),\ \sigma_2\rangle, & \quad \sigma_2 = \{x \leftarrow s_1\}, \\
 & \quad \langle f(\tilde{t}\sigma_3) \ll_{F}^{?} f(\tilde{s}),\ \sigma_3\rangle, & \quad \sigma_3 = \{x \leftarrow f(s_1)\}, \\
 & \quad \langle f(x, \tilde{t}\sigma_4) \ll_{F}^{?} f(\tilde{s}),\ \sigma_4\rangle, & \quad \sigma_4 = \{x \leftarrow f(s_1, x)\}. \\
 & f(\overline{x}, \tilde{t}) \ll_{F}^{?} f(s_1, \tilde{s}) \rightsquigarrow \langle\langle f(\tilde{t}\sigma_1) \ll_{F}^{?} f(s_1, \tilde{s}),\ \sigma_1\rangle, & \text{where } \sigma_1 = \{\overline{x} \leftarrow f()\}, \\
 & \quad \langle f(\tilde{t}\sigma_2) \ll_{F}^{?} f(\tilde{s}),\ \sigma_2\rangle, & \quad \sigma_2 = \{\overline{x} \leftarrow s_1\}, \\
 & \quad \langle f(\tilde{t}\sigma_3) \ll_{F}^{?} f(\tilde{s}),\ \sigma_3\rangle, & \quad \sigma_3 = \{\overline{x} \leftarrow f(s_1)\}, \\
 & \quad \langle f(\overline{x}, \tilde{t}\sigma_4) \ll_{F}^{?} f(\tilde{s}),\ \sigma_4\rangle, & \quad \sigma_4 = \{\overline{x} \leftarrow f(s_1, \overline{x})\}, \\
 & \quad \langle f(\overline{x}, \tilde{t}\sigma_5) \ll_{F}^{?} f(s_1, \tilde{s}),\ \sigma_5\rangle, & \quad \sigma_5 = \{\overline{x} \leftarrow f(), \overline{x})\}, \\
 & \quad \langle f(\overline{x}, \tilde{t}\sigma_6) \ll_{F}^{?} f(\tilde{s}),\ \sigma_6\rangle, & \quad \sigma_6 = \{\overline{x} \leftarrow s_1, \overline{x}\}, \\
 & \quad \langle f(\overline{x}, \tilde{t}\sigma_7) \ll_{F}^{?} f(\tilde{s}),\ \sigma_7\rangle, & \quad \sigma_7 = \{\overline{x} \leftarrow s_1, f(\overline{x})\}, \\
 & \quad \langle f(\overline{x}, \tilde{t}\sigma_8) \ll_{F}^{?} f(\tilde{s}),\ \sigma_8\rangle, & \quad \sigma_8 = \{\overline{x} \leftarrow f(s_1), \overline{x}\}, \\
 & \quad \langle f(\overline{x}, \tilde{t}\sigma_9) \ll_{F}^{?} f(\tilde{s}),\ \sigma_9\rangle\rangle, & \quad \sigma_9 = \{\overline{x} \leftarrow f(s_1), f(\overline{x})\}
\end{array}
$$

Fig. 2. F-matching. Transformation rules. $\tilde{t}$ and $\tilde{s}$ are sequences of terms. $h \in \mathcal{F}_{\mathrm{Fix}} \cup \mathcal{F}_{\mathrm{Flex}}$ is free. $h_1, h_2 \in \mathcal{F}_{\mathrm{Fix}} \cup \mathcal{F}_{\mathrm{Flex}}$ can be free or flat. $f \in \mathcal{F}_{\mathrm{Flex}}$ is flat.

$$\begin{array}{ll}
\textsf{Eliminate (cont.):} & f(t,\tilde{t}) \ll^?_F f(t,\tilde{s}) \rightsquigarrow \langle\langle f(\tilde{t}) \ll^?_F f(\tilde{s}),\ \varepsilon\rangle\rangle. \\
& h(t_1,\tilde{t}) \ll^?_F h(s_1,\tilde{s}) \rightsquigarrow \langle\langle g(\tilde{t}\sigma) \ll^?_F g(\tilde{s}),\ \sigma\rangle\rangle, \text{ if } t_1 \ll^?_F s_1 \rightsquigarrow \langle\langle \top,\ \sigma\rangle\rangle. \\
& h(\overline{x},\tilde{t}) \ll^?_F h(s_1,\tilde{s}) \rightsquigarrow \langle\langle g(\tilde{t}\sigma_1) \ll^?_F g(\tilde{s}),\ \sigma_1\rangle, \text{ where } \sigma_1 = \{\overline{x} \leftarrow s_1\}, \\
& \quad \langle g(\overline{x},\tilde{t}\sigma_2) \ll^?_F g(\tilde{s}),\ \sigma_2\rangle\rangle, \qquad\qquad \sigma_2 = \{\overline{x} \leftarrow s_1, \overline{x}\}.
\end{array}$$

Fig. 3. F-matching. Transformation rules (cont.). $\tilde{t}$ and $\tilde{s}$ are sequences of terms. $h \in \mathcal{F}_{\text{Fix}} \cup \mathcal{F}_{\text{Flex}}$ is free. $g \in \mathcal{F}_{\text{Flex}}$ is a new free symbol if $h \in \mathcal{F}_{\text{Fix}}$ in the same rule, otherwise $g = h$. $f \in \mathcal{F}_{\text{Flex}}$ is flat.

$$\begin{array}{lll}
\textsf{SuccessM:} & t \ll^?_\emptyset t \rightsquigarrow \langle\langle \top,\ \varepsilon\rangle\rangle. & \\
& x \ll^?_F t \rightsquigarrow \langle\langle \top,\ \{x \leftarrow t\}\rangle\rangle. & \\
\textsf{FailureM:} & f() \ll^?_F f(t_1,\tilde{t}) \rightsquigarrow \bot. & \\
& f(t_1,\tilde{t}) \ll^?_F f() \rightsquigarrow \bot. & \\
& f(t_1,\tilde{t}) \ll^?_F f(s_1,\tilde{s}) \rightsquigarrow \bot, & \text{if } t_1 \ll^?_F s_1 \rightsquigarrow \bot. \\
& c_1 \ll^?_F c_2 \rightsquigarrow \bot, & \text{if } c_1 \neq c_2. \\
& h_1(\tilde{t}) \ll^?_F h_2(\tilde{s}) \rightsquigarrow \bot, & \text{if } h_1 \neq h_2. \\
& h() \ll^?_F h(t_1,\tilde{t}) \rightsquigarrow \bot. & \\
& h(t_1,\tilde{t}) \ll^?_F h() \rightsquigarrow \bot. & \\
& h(t_1,\tilde{t}) \ll^?_F h(s_1,\tilde{s}) \rightsquigarrow \bot, & \text{if } t_1 \ll^?_F s_1 \rightsquigarrow \bot. \\
\textsf{SplitM:} & h_1(t_1,\tilde{t}) \ll^?_F h_1(s_1,\tilde{s}) \rightsquigarrow & \text{if } t_1 \notin \mathcal{V}_{\text{Ind}} \cup \mathcal{V}_{\text{Seq}} \text{ and} \\
& \quad \langle\langle h_1(r_1,\tilde{t}\sigma_1) \ll^?_F h_1(q_1,\tilde{s}),\ \sigma_1\rangle, & t_1 \ll^?_F s_1 \rightsquigarrow \langle\langle r_1 \ll^?_F q_1,\ \sigma_1\rangle, \\
& \quad \ldots, \langle h_1(r_k,\tilde{t}\sigma_k) \ll^?_F h_1(q_k,\tilde{s}),\ \sigma_k\rangle\rangle & \quad \ldots, \langle r_k \ll^?_F q_k,\ \sigma_k\rangle\rangle. \\
\textsf{EliminateM:} & f(t,\tilde{t}) \ll^?_F f(t,\tilde{s}) \rightsquigarrow & \\
& \quad \langle\langle f(\tilde{t}) \ll^?_F f(\tilde{s}),\ \varepsilon\rangle\rangle. & \\
& f(x,\tilde{t}) \ll^?_F f(s_1,\tilde{s}) \rightsquigarrow & \text{where} \\
& \quad \langle\langle f(\tilde{t}\sigma_1) \ll^?_F f(\tilde{s}),\ \sigma_1\rangle, & \quad \sigma_1 = \{x \leftarrow f(s_1)\}, \\
& \quad \langle f(x,\tilde{t}\sigma_2) \ll^?_F f(\tilde{s}),\ \sigma_2\rangle\rangle, & \quad \sigma_2 = \{x \leftarrow f(s_1,x)\}. \\
& f(\overline{x},\tilde{t}) \ll^?_F f(s_1,\tilde{s}) \rightsquigarrow & \text{where} \\
& \quad \langle\langle f(\tilde{t}\sigma_1) \ll^?_F f(\tilde{s}),\ \sigma_1\rangle, & \quad \sigma_1 = \{\overline{x} \leftarrow s_1\}, \\
& \quad \langle f(\overline{x},\tilde{t}\sigma_2) \ll^?_F f(\tilde{s}),\ \sigma_2\rangle\rangle, & \quad \sigma_2 = \{\overline{x} \leftarrow s_1, \overline{x}\}. \\
& h(t_1,\tilde{t}) \ll^?_F h(s_1,\tilde{s}) \rightsquigarrow & \text{if } t_1 \ll^?_F s_1 \rightsquigarrow \langle\langle \top,\ \sigma\rangle\rangle. \\
& \quad \langle\langle g(\tilde{t}\sigma) \ll^?_F g(\tilde{s}),\ \sigma\rangle\rangle, & \\
& h(\overline{x},\tilde{t}) \ll^?_F h(s_1,\tilde{s}) \rightsquigarrow & \text{where} \\
& \quad \langle\langle g(\tilde{t}\sigma_1) \ll^?_F g(\tilde{s}),\ \sigma_1\rangle, & \quad \sigma_1 = \{\overline{x} \leftarrow s_1\}, \\
& \quad \langle g(\overline{x},\tilde{t}\sigma_2) \ll^?_F g(\tilde{s}),\ \sigma_2\rangle\rangle, & \quad \sigma_2 = \{\overline{x} \leftarrow s_1, \overline{x}\}.
\end{array}$$

Fig. 4. Transformation rules to simulate MATHEMATICA flat matching. $\tilde{t}$ and $\tilde{s}$ are sequences of terms. $h_1, h_2 \in \mathcal{F}_{\text{Fix}} \cup \mathcal{F}_{\text{Flex}}$ can be free or flat. $h \in \mathcal{F}_{\text{Fix}} \cup \mathcal{F}_{\text{Flex}}$ is free. $g \in \mathcal{F}_{\text{Flex}}$ is a new free symbol if $h \in \mathcal{F}_{\text{Fix}}$ in the same rule, otherwise $g = h$. $f \in \mathcal{F}_{\text{Flex}}$ is flat.