Petri Net Algorithms in the Theory of Matrix Grammars

Dirk Hauschildt and Matthias Jantzen
FB Informatik, Universität Hamburg
Vogt-Kölln-Straße 30, 22527 Hamburg
{dirk,jantzen}@informatik.uni-hamburg.de

Abstract

This paper shows that the languages over a one-letter alphabet generated by a context-free matrix grammar are always regular. Moreover we give a decision procedure for the question of whether a context-free matrix language is finite. Hereby we strengthen a result of [Mk 92] and settle a number of open questions in [DP 89]. Both results are obtained by a reduction to Petri net problems.

1 Introduction

Petri nets and vector addition systems are different representations of the same construct. While their notation as nets emphasizes their role as a specification and analysis tool for distributed systems, their alternative definition as vector replacement systems illustrates their importance in the theory of formal languages. Here they are regarded as semi-Thue systems for commutative monoids over a finite alphabet in contrast to the usually considered free monoid. The alphabet is represented in the set of places, the transitions play the role of grammatical rules, and the actually derived sentential form is encoded in the marking.

Additional features of grammars, such as the distinction between intermediate sentential forms and actual elements of the generated language, do not occur in the definitions of the nets, but can be implemented by considering only those elements of the reachability set of which submarkings on certain places have a fixed, predetermined value.

Therefore, Petri net theory is applicable to certain problems in formal language theory in which

- the ordering of the letters in words of the generated language is irrelevant, and
- the ordering of the letters within sentential forms does not influence the further derivation (except for the ordering of the result).

The first condition is fulfilled if the alphabet contains only one letter or for problems like emptiness or finiteness of a language. The second requirement is a characterization of context-free grammars.

Petri net theory is also applicable to extensions of ordinary grammars such as matrix grammars and random context grammars because their additional requirements can be easily expressed in terms of the nets.

We will give two examples of this method by solving two problems about matrix grammars stated as open in [DP 89].

- Firstly we show that languages of a matrix grammar or a random context grammar over $\{a\}$ are always regular.
- Secondly we give a decision procedure for the question of whether the language of a matrix grammar or of a random context grammar is finite.

For this purpose we make use of the semilinearity algorithm presented in [Ha 90]. Given a specification of a portion of the reachability set of a net, this algorithm decides whether the specified set is semilinear. It produces a semilinear representation of the set in that case and two close estimates otherwise.

Section 2 gives the net theoretical definitions needed in the sequel. The semilinearity algorithm, on which our results are based, is introduced in Section 3. The next section developes some applications of the algorithm in the field of Petri nets. Our two main results mentioned above are presented in Section 5. Finally, Section 6 contains a more detailed list of open problems which have been solved with our methods.

2 Definitions

2.1 Petri Nets

A ***Petri net*** (more exactly: place/transition net) $N = (P, T, F, B)$ consists of the set P of places, the set T of transitions, and two weight functions

$$\begin{aligned} F &: T \to \mathbb{N}^P, \\ B &: T \to \mathbb{N}^P. \end{aligned}$$

P and T are assumed to be finite and disjoint.

Markings are functions $\underline{m} : P \mapsto \mathbb{N}$ (usually written as vectors in $\mathbb{N}^P$). Sometimes we will associate an initial marking $\underline{m}_0$ and/or a final marking $\underline{m}_f$ with a net. This is done by adding $\underline{m}_0$ and/or $\underline{m}_f$ to the above 4-tuple.

F is called the ***forward incidence function***. $F(t)$ describes how many tokens are moved from the places to the transition t when t fires. Conversely, the ***backward incidence function*** B determines the number of tokens returned by a transition. Hence, $\Delta := B - F$ describes the change of the marking produced by the firing of a transition. It is called the ***incidence function*** of the net. Similar to the vector notation for markings, F, B, and Δ are usually written as matrices.

A transition t is ***enabled*** in the marking $\underline{m}$ if and only if $F(t) \leq \underline{m}$, this is denoted by $\underline{m}\ (t\rangle)$. The firing of a transition t transforms $\underline{m}$ into $\underline{m}' = \underline{m} + \Delta(t)$ (in symbols $\underline{m}\ (t\rangle\ \underline{m}'$). This definition is recursively extended to sequences of transitions: A transition sequence $w = t_1 \cdots t_n \in T^*$ is enabled in $\underline{m}$ iff $\underline{m}\ (t_1\rangle$ and $t_2 \cdots t_n$ is enabled in $\underline{m} + \Delta(t_1)$. Then $\underline{m}\ (w\rangle\ \underline{m} + \Delta(w)$. Moreover, we assume $\underline{m}\ (\lambda\rangle\ \underline{m}$ for arbitrary markings $\underline{m}$ of $\mathbb{N}^P$.

2.2 (Semi-)Linear Sets

A constant $\underline{c}$ and a set of periods $X = \{\underline{x}_1, \ldots, \underline{x}_n\}$, each of which is an element of $\mathbb{Z}^A$, for some finite set A, defines the linear set

$$L := \mathcal{N}(\underline{c};\, X) := \left\{ \underline{c} + X \cdot \underline{r} \;\middle|\; \underline{r} \in \mathbb{N}^X \right\}.$$

The information encoded in X is twofold: Considered as a set, it defines the basic periods of L, and considered as a mapping, it describes which periods can be added to elements of L without leaving the set. The image $X \cdot \underline{r}$ of the homomorphic mapping $X : \mathbb{N}^X \mapsto \mathbb{Z}^A$ is the set $\mathcal{N}(\underline{0};\, X)$ of linear combinations of X. If $r_x \neq 0$ for all $\underline{x} \in X$ we call $X \cdot \underline{r}$ a proper linear combination of X.

A semilinear set SL is the union of finitely many linear sets. If the sets of periods of all linear components of SL coincide, i.e., if $SL = \bigcup \left\{ \mathcal{N}(\underline{c};\, X) \;\middle|\; \underline{c} \in C \right\}$ for a fixed set X, we characterize it more shortly by $\mathcal{N}(C;\, X)$.

The ***Parikh mapping*** $\psi : A^* \to \mathbb{N}^A$ relates every word $w \in A$ to the vector $\underline{x} \in \mathbb{N}^A$ for which x_a contains the number of occurrences of a in w.

2.3 The Reachability Relation

The reachability relation R_N of a Petri net N is defined by

$$R_N \;:=\; \{ (\underline{a}, \underline{b}) \in \mathbb{N}^P \times \mathbb{N}^P \;\mid \exists w \in T^* \text{ such that } \underline{a}\ (w\rangle\ \underline{b} \,\}\,.$$

It is often useful to extend this relation by some information about the paths connecting $\underline{a}$ and $\underline{b}$. Therefore we define the extended reachability relation

$$ER_N \;:=\; \left\{ (\underline{a}, \underline{b}, \underline{f}) \in \mathbb{N}^P \times \mathbb{N}^P \times \mathbb{N}^T \;\middle|\; \exists w \in T^* \text{ such that } \underline{a}\ (w\rangle\ \underline{b} \text{ and } \psi(w) = \underline{f} \right\}.$$

The index is omitted if the net is evident from the context.

We will also consider the set of paths leading from a certain fixed initial marking to a fixed final marking. Given a homomorphism $h : T^* \mapsto A^*$, where A is a finite alphabet, we define the *labelled terminal Petri net language* $L(N, h, \underline{m}_0, \underline{m}_f)$ by

$$L(N, h, \underline{m}_0, \underline{m}_f) \;:=\; \{ h(w) \in A^* \mid \underline{m}_0\ (w\rangle\ \underline{m}_f \}.$$

3 The Semilinearity Algorithm

In order to formulate queries about portions of the reachability set, a kind of specification language for reachability sets was introduced in [Ha 90]. It contains ***semilinearity problems*** of the form $\mathcal{P} = (N, L, \pi)$ where

- (i) N is a Petri net,
- (ii) $L \subseteq \mathbb{N}^P \times \mathbb{N}^P \times \mathbb{N}^T$ defines the portion of the reachability relation to be dealt with, and
- (iii) the homomorphism $\pi : \mathbb{N}^P \times \mathbb{N}^P \times \mathbb{N}^T \to V$ selects the type of information to be computed. Usually, π is just a projection.

The set of interest specified by $\mathcal{P}$ is $SOL_{\mathcal{P}} := \pi(ER_N \cap L)$.

The ordinary reachability set of N from a given initial marking $\underline{m}_0$ can be specified by setting L to $\{\underline{m}_0\} \times \mathbb{N}^P \times \mathbb{N}^T$ and $\pi(\underline{a}, \underline{b}, \underline{f}) := \underline{b}$.

Given such a triplet $\mathcal{P}$, the ***semilinearity algorithm*** presented in [Ha 90] computes two semilinear sets $LB_{\mathcal{P}}$ and $UB_{\mathcal{P}}$ with

$$LB_{\mathcal{P}} \subseteq SOL_{\mathcal{P}} \subseteq UB_{\mathcal{P}}$$

such that

$$\text{DIM}\,(UB'_{\mathcal{P}} \setminus LB'_{\mathcal{P}}) \geq \text{DIM}\,(UB_{\mathcal{P}} \setminus LB_{\mathcal{P}})$$

holds for every pair LB', UB' of semilinear sets with $LB' \subseteq SOL_{\mathcal{P}} \subseteq UB'$. This assertion implies $LB_{\mathcal{P}} = SOL_{\mathcal{P}} = UB_{\mathcal{P}}$ for the case that $SOL_{\mathcal{P}}$ itself is semilinear.

This goal is achieved by iteratively computing sets $\Gamma_{\mathcal{P}}$ of so-called MGTS's (marked graph transition sequences) each of which describes a portion of $SOL_{\mathcal{P}}$. These structures are adopted from similar constructs used in algorithms for the reachability problem: given a net N and two markings $\underline{m}_0$, $\underline{m}_f$, $\underline{m}_f$ is reachable from $\underline{m}_0$. An MGTS U is essentially the same as a regular constraint graph with a consistent labelling in [Ma 84] and still very similar to a GVASS satisfying θ in [Ko 82]. The name MGTS itself firstly occurred in (a predecessor of) [La 88]. In fact, every MGTS computed in the semilinearity algorithm is the result of an invocation of the reachability algorithm.

The exact structure of an MGTS is not important for our purposes and will not be discussed here. With every MGTS U we can associate a set SOL_U containing the elements of $SOL_{\mathcal{P}}$ described by U. Unfortunately, no method to characterize SOL_U in a more effective way than by giving U itself is known so far. The set $\Gamma_{\mathcal{P}}$ of MGTS's is a complete description of $SOL_{\mathcal{P}}$ by virtue of the fact that

$$\bigcup\{SOL_U \mid U \in \Gamma\} \;=\; SOL_{\mathcal{P}}.$$

The additional information about $SOL_{\mathcal{P}}$ contained in Γ comprises in two statements. Firstly, a fixed, linear upper estimate UB_U of SOL_U is attached to every $U \in \Gamma$. Theorem 3.1 below exhibits a method to compute also some lower bounds for SOL_U. By defining one of them as LB_U we determine two first approximations

$$LB_0 \;:=\; \bigcup\{LB_U \mid U \in \Gamma\} \quad\text{and}\quad UB_0 \;:=\; \bigcup\{UB_U \mid U \in \Gamma\} \tag{1}$$

of $SOL_{\mathcal{P}}$. The algorithm proceeds by recursively considering the linear components of $UB_0 \setminus LB_0$ as L. This way, it evaluates some closer information about the still undecided regions. The process halts when no more progress can be made this way.

The problem to be solved in every round i of this procedure is to determine the lower bounds $LB_U \subseteq SOL_U$ for every $U \in \Gamma_i$ in such a way that $\text{DIM}\,(UB_i \setminus LB_i) < \text{DIM}\,(L_i)$. One can show that UB_{i-1} and LB_{i-1} were already optimal (in the sense that the dimension of the undecided region is minimal) when such lower bounds cannot be found.

Theorem 5.3.7 in [Ha 90] is used to construct linear subsets of some SOL_U. Some notation introduced in that paper will not be needed for our purposes. To avoid unnecessary definitions we give only a simplified version of that theorem.

Theorem 3.1 *Let U be an MGTS with $UB_U = \mathcal{N}(\underline{c}\,; X)$. Given a finite subset C of UB_B and a proper linear combination $\underline{x}$ of X, one can find a constant $n \geq 0$ with*

$$\mathcal{N}\big(C + n \cdot \underline{x}\,;\, \{\underline{x}\}\big) \subseteq SOL_U.$$

Informally, the theorem says that one can shift any element of $UB_U = \mathcal{N}(\underline{c}\,; X)$ into SOL_U by sufficiently often adding an arbitrary proper linear combination $\underline{x}$ of X to it. The number n in the lemma can be obtained by determining the necessary number of shift operations for every $\underline{c} \in C$ and then building the maximum.

4 Applications in the Field of Petri Nets

This section extracts the information needed in connection with matrix grammars out of Theorem 3.1. We consider an MGTS U with $UB_U = \mathcal{N}(\underline{c}\,; X)$. If the set X of periods is empty, i.e., if $UB_U = \{\underline{c}\}$, the theorem (applied to $C := \{\underline{c}\}$ and $\underline{x} := \underline{0}$) reveals that $\underline{c} \in SOL_U$. With $|X| \geq 1$ the theorem shows that SOL_U contains infinitely many elements. This leads to our first observation.

Corollary 4.1 *For a given semilinearity problem $\mathcal{P}$ it is decidable whether $SOL_{\mathcal{P}}$ is finite.*

Proof: We just have to compute the set $\Gamma_{\mathcal{P}}$ of MGTS's. SOL_U is infinite if and only if there is a $U \in \Gamma$ with $UB_U =: \mathcal{N}(\underline{c}\,; X)$ and $|X| \geq 1$. $\square$

The other case we consider contains semilinearity problems $\mathcal{P} = (N, L, \pi)$ in which π maps L to the set $\mathbb{N}$ of natural numbers. We will show that in this case $UB_0 \setminus LB_0$ is finite, i.e., that the first round of the semilinearity algorithm decides $x \in SOL_{\mathcal{P}}$ for all but finitely many numbers x. By Equation (1), it suffices to consider the UB_U and LB_U, $U \in \Gamma_{\mathcal{P}}$, separately.

Lemma 4.2 *Let $\mathcal{P} = (N, L, \pi)$ be a semilinearity problem with $\pi : L \rightarrow \mathbb{N}$. For every MGTS $U \in \Gamma_P$ there exists a subset LB_U of SOL_U such that $UB_U \setminus LB_U$ is finite.*

Proof: Let $UB_U = \mathcal{N}(c\,; X)$ and m be a proper linear combination of X. By the choice of π, m is a natural number. If $m = 0$, UB_U is finite and the lemma holds trivially.

Assuming $m \geq 1$, we partition UB_U into residue classes

$$R_i = \{x \in \mathbb{N} \mid x \equiv i \pmod{m}\}$$

and select a point $r_i \in R_i$ for every R_i not disjoint to UB_U. By applying Theorem 3.1 to

$$C \;:=\; \{r_i \mid R_i \cap UB_U \neq \emptyset\} \quad \text{and} \quad \underline{x} \;:=\; m,$$

we obtain a subset $LB_U := \mathcal{N}\big(C + n \cdot m\,;\, \{m\}\big)$ of SOL_U which contains all elements of UB_U which are not smaller than a certain bound, namely $n \cdot m + \textsc{max}\{r_i \mid R_i \cap UB_U \neq \emptyset\}$. To see this we consider the equivalence classes R_i separately. Nothing is to prove if $R_i \cap UB_U$ is empty. Otherwise the subset

$$\mathcal{N}\big(r_i + n \cdot m\,;\, \{\underline{m}\}\big) \;=\; \{x \in \mathbb{N} \mid x \geq r_i + m \cdot n,\; x \equiv r_i \pmod{m}\}$$

of LB_U contains all sufficiently large elements of R_i. $\square$

Since the semilinearity algorithm selects all its lower bounds as suggested by the lemma, it nearly has finished its task after one round. The second round only has to consider the elements of the set difference $UB_0 \setminus LB_0$ one at a time to determine whether they belong to SOL_P. This leads to our second corollary.

Corollary 4.3 *Let $\mathcal{P} = (N, L, \pi)$ be a semilinearity problem with $\pi : L \to I\!N$. Then $SOL_{\mathcal{P}}$ is an effectively computable semilinear set.*

A similar approach was applied in [KLM 89] to coverability graphs instead of MGTS's. It was used to show that projections of reachability sets to one coordinate are always semilinear. In Section 5 we make use of the opportunity to restrict our attention to certain linear subsets of the reachability set. Therefore we cannot apply that result directly.

Corollary 4.3 can be easily used to show that the set $\{|w| \mid w \in L\}$ is semilinear for every Petri net language L. It is well known that this condition is equivalent for L being regular if the alphabet of L contains only one letter. Hence we obtain the following corollary.

Corollary 4.4 *Let $L = L(N, h, \underline{m}_0, \underline{m}_f)$ be a Petri net language with $h : T^* \mapsto \{a\}^*$. Then L is an effectively computable regular set.*

5 Applications in the Field of Matrix Grammars

It is well known that programmed grammars generate the same class of languages as matrix grammars with appearance checking. Arbitrary matrix grammars with appearance checking and λ-rules generate the recursively enumerable languages, while those without λ-rules generate only context-sensitive languages that are in *NP*. Before explaining the new results obtained we give the definition of context-free matrix grammars with and without appearance checking as it is used in [DP 89].

Definition 5.1 *Let $G = (V_N, V_T, S, R)$ be a context-free grammar with nonterminal alphabet V_N, terminal alphabet V_T, initial symbol $S \in V_N$, and a set $R \subseteq V_N \times (V_N \cup V_T)^*$ in which all productions carry a unique label from a set Σ. If r is a label of the production $A \to w$ we write $r : A \to w$.*

*A context-free matrix grammar Γ based on the context-free grammar G is specified by the triplet $\Gamma := (G, M, F)$ where $M \subseteq \Sigma^*$ is a finite set of sequences of production labels, each of which is called a **matrix** and the set $F \subseteq \Sigma$, containing the labels of those productions that can be passed over in case they are not applicable. This is called **appearance checking**.*

Derivation of sentential forms proceeds as follows: $u \underset{ac}{\Longrightarrow} v$ holds if and only if there exists a matrix $m = r_1 r_2 \cdots r_n$ in M and some strings $w_0, w_1, \ldots, w_n \in (V_N \cup V_T)^*$ such that $u = w_0$, $v = w_n$, and for each $i \in \{1, \ldots, n\}$ either of the cases (a) or (b) holds:

(a) $w_{i-1} = w'_{i-1}\, A_i\, w''_{i-1}$ and $w_i = w'_{i-1}\, v_i\, w''_{i-1}$,

(b) $w_i = w_{i-1}$, *A_i does not occur in w_i, and* $r_i \in F$.

If $F = \emptyset$ then we write $\Longrightarrow$ instead of $\underset{ac}{\Longrightarrow}$. Note that in this case all non-applicable productions lead to a blocking derivation. The language generated by the matrix grammar $\Gamma := (G, M, F)$ is defined as

$$L(\Gamma) \; := \; \{w \in V_T^* \mid A_0 \overset{*}{\underset{ac}{\Longrightarrow}} w\}.$$

Definition 5.2 *The family of all context-free matrix languages with possible appearance checking is denoted by $\mathscr{L}(M, \mathscr{CF}, ac)$ or by $\mathscr{L}(M, \mathscr{CF}{-}\lambda, ac)$ depending on whether the underlying context-free grammar is λ-free. The corresponding classes of languages that can be generated by grammars without appearance checking, i.e., with $F = \emptyset$ are denoted by $\mathscr{L}(M, \mathscr{CF})$ and $\mathscr{L}(M, \mathscr{CF}{-}\lambda)$.*

It is known that $\mathscr{L}(M, \mathscr{CF}{-}\lambda, ac)$ is an AFL strictly contained in the family of context-sensitive languages (see [Ro 69], [vL 75], and [DP 89]), while $\mathscr{L}(M, \mathscr{CF}, ac)$ equals the family of recursively enumerable sets. The inclusions

$$\mathscr{L}(M, \mathscr{CF}{-}\lambda) \subseteq \mathscr{L}(M, \mathscr{CF}{-}\lambda, ac) \quad \text{and} \quad \mathscr{L}(M, \mathscr{CF}{-}\lambda) \subseteq \mathscr{L}(M, \mathscr{CF})$$

are obvious from the definition. It was conjectured, but not proved in [DP 89] that these inclusions are proper. This assumption will be confirmed in this section by proving that each language over a one-letter alphabet within the former family is necessarily regular.

It was shown in [DP 89, pp. 267-270] that Petri nets (with a fixed initial marking) can be simulated by matrix grammars, i.e., given N and $\underline{m}_0$, one can compute a matrix grammar Γ such that the set of reachable markings equals $\psi(L(\Gamma))$.

We will demonstrate here that the reverse simulation can be accomplished as well. Let $\Gamma = (G, M, \emptyset)$ be a context-free matrix grammar with $G = (V_N, V_T, R, S)$ its underlying grammar, and $A = (Q, \Sigma, q_0, \{q_0\})$ be the canonical finite automaton accepting M^*, the set of legal applications of sets of rules of G.

The place set of the net $N = (P, T, F, B)$ shall contain one place for every state of A and one place for every (terminal or nonterminal) symbol of G. This is achieved by setting $P := V_n \cup V_T \cup Q$. Moreover, the transition set coincides with Σ. For every rule $r\colon A \to w$ let $q \xrightarrow{r} q'$ be the corresponding arc of A. Then we set

$$F(r) \; := \; \psi(Aq) \quad \text{and} \quad B(r) \; := \; \psi(wq').$$

If the net N is started from an initial marking $\underline{m}_0$ containing one token on S and one token on q_0, it simulates derivations of Γ step by step. Every reachable marking contains the Parikh image of the current sentential form in $V_N \cup V_T$ and the state of the automaton in Q.

The other two parameters of the semilinearity problem are used to select the actual elements of $L(\Gamma)$. We are interested in transition sequences leading from $\underline{m}_0 = \psi(S q_0)$ to any final marking in which q_0 contains one token and all elements of V_N and Q are empty. This assures that the 'derived' sentential form contains no more nonterminals and the automaton is in its (initial and) only final state. Hence we define

$$L \; := \; \big\{ (\psi(S q_0),\, \psi(v q_0),\, \psi(w)) \,\big|\, v \in N_T^*,\, w \in \Sigma^* \big\}.$$

The mapping π is used to project an element $(\underline{a}, \underline{b}, \underline{f})$ of L onto the portion of $\underline{b}$ describing the final marking on V_T. This way we obtain

$$SOL_{\mathcal{P}} = \{\psi(w) \mid w \in L(\Gamma)\}$$

as required.

With the help of this simulation we easily develop the assertions proposed in the introduction.

Theorem 5.3 *All languages over one-letter alphabets in $\mathscr{L}(M, \mathscr{CF})$ are regular.*

Proof: Let $\{a\}^* \supseteq L \in \mathscr{L}(M, \mathscr{CF})$ and $\mathcal{P} = (N, L, \pi)$ be the semilinearity problem derived from L. Then π maps L into $\mathbb{N}^{V_T} = \mathbb{N}$. Hence Corollary 4.3 is applicable and shows that $SOL_{\mathcal{P}} = \psi(L(\Gamma))$ is semilinear. Consequently, $L(\Gamma)$ itself is regular. $\square$

By using the same simulation, the finiteness problem for the families of context-free matrix grammars without appearence checking can be reduced to Corollary 4.1.

Theorem 5.4 *Finiteness for context-free matrix languages without appearance checking is decidable.*

Proof: Again let $L \in \mathscr{L}(M, \mathscr{CF})$ and $\mathcal{P} = (N, L, \pi)$ be the semilinearity problem derived from L. Then $SOL_{\mathcal{P}} = \psi(L(\Gamma))$ being finite is equivalent to $L(\Gamma)$ itself being finite. Hence the theorem follows from Corollary 4.1. $\square$

6 Answers to Open Questions

Problem 1.3.3 of [DP 89] summarizes the six open problems in Table 1.3.3, the list of decision questions for matrix and random context grammars. Four of them consider the question for which of the families

$$\mathscr{L}(RC, \mathscr{CF}\text{–}\lambda), \quad \mathscr{L}(M, \mathscr{CF}\text{–}\lambda), \quad \mathscr{L}(RC, \mathscr{CF}), \quad \text{and} \quad \mathscr{L}(M, \mathscr{CF})$$

the finiteness problem is decidable. Since random context grammars can be effectively transformed in matrix grammars, all four cases can be reduced to Theorem 5.4, and answered with “yes”.

The other two decidability questions given in the table, the word problems for

$$\mathscr{L}(RC, \mathscr{CF}) \quad \text{and} \quad \mathscr{L}(M, \mathscr{CF}),$$

can be solved with the methods presented in the book itself. If one wants to decide whether a string w is an element of a matrix language L one first computes the matrix language $L' := L \cap \{w\}$ (Lemma 1.3.5, closedness under intersection by regular sets) and then determines whether L' is empty (decidable by Theorem 1.3.4).

We now have a look at our other main result, Theorem 5.3, which tells us that matrix languages over a one-letter alphabet are regular. One of its implications is that the open problem 1.1.1 of [DP 89] questioning whether the context sensitive, but not context free languages

$$\{a^{2^n} \mid n \geq 1\}, \quad \{a^{n^2} \mid n \geq 1\}, \quad \text{and} \quad \{a^n \mid n \text{ is a prime number }\} \tag{2}$$

belonging to $\mathscr{L}(M,\mathscr{CF})$ can be answered in the negative[1]. Hence, these sets serve as the "concrete" languages in $\mathscr{L}(\mathscr{CS}) \setminus \mathscr{L}(M,\mathscr{CF})$ sought for in [DP 89, Problem 1.2.2].

Since context-free matrix grammars with appearance checking, but without λ-rules exist for the three languages of Equation (2), we can answer the last question in [DP 89, Problem 1.2.3] as well:

Corollary 6.1 *The inclusion $\mathscr{L}(M,\mathscr{CF}-\lambda) \subset \mathscr{L}(M,\mathscr{CF}-\lambda, ac)$ is strict.*

The same observation further shows that $\mathscr{L}(M,\mathscr{CF})$ is not a superset of $\mathscr{L}(M,\mathscr{CF}-\lambda, ac)$ and strictly contained in $\mathscr{RE}$ ([DP 89, Problem 1.2.1]). The last inclusion could as well be deduced from the decidability of the membership problem for $\mathscr{L}(M,\mathscr{CF})$ (see the remark above).

The inclusions $\mathscr{L}(RC,\mathscr{CF}) \subseteq \mathscr{L}(M,\mathscr{CF}) \subset \mathscr{RE}$ have some implications to closure properties as well:

Corollary 6.2 *The families $\mathscr{L}(RC,\mathscr{CF})$ and $\mathscr{L}(M,\mathscr{CF})$ are not closed with respect to intersection or to complementation.*

Proof: It is well known (see [GGH 67, Theorem 3.1]) that the closure of $\mathscr{CF}$ with respect to homomorphism and intersection is the class of recursive enumerable languages. Since both, $\mathscr{L}(RC,\mathscr{CF})$ and $\mathscr{L}(M,\mathscr{CF})$, contain $\mathscr{CF}$ and are closed with respect to homomorphism, but are strictly contained in $\mathscr{RE}$, they cannot be closed with respect to intersection.

Now the result about complementation, by De Morgan's law, follows from the fact that the two families are closed with respect to union. $\square$

Finally we consider the class $\mathscr{L}(\lambda\, USC)$ of unordered scattered context grammars having only a single terminal symbol. These grammars are almost identical to Petri nets with only one transition label. The class of languages generated by such grammars coincides with the family of Petri net languages. Hence we can apply Corollary 4.4 to state:

Corollary 6.3 *All languages over one-letter alphabets in $\mathscr{L}(\lambda\, USC)$ are regular.*

7 Conclusion

By applying decidability results on semilinearity problems about Petri net reachability relations proved in [Ha 90] we solved a number of long-standing open problems in the theory of regulated string rewriting.

Specifically we have proved that the following three classes of languages coincide over a one-letter alphabet:

[1]It was already shown in [La 88] by a reduction to the reachability problem that (two of) these sets cannot be Petri net languages.

- the family of languages defined by context-free matrix grammars without appearance checking
- the terminal Petri net languages
- the regular sets

Moreover, we proved the decidability of the finiteness problem for the family of languages defined by context-free matrix grammars without appearance checking.

A number of problems, stated as open in [DP 89], have been settled as corollaries of the above two theorems.

References

[DP 89] J.Dassow, G.Paun: *Regular rewriting in formal language theory*, EATCS Monographs on Theoretical Computer Science, Springer-Verlag (1989).

[GGH 67] S.Ginsburg, S.Greibach, and M.A.Harrison: *One-way stack automata*, Journal of the ACM 14 (1967), 389-418.

[Ha 90] D.Hauschildt: *Semilinearity of the reachability set is decidable for Petri nets*, Doctoral Thesis, Dept. of Computer Science, University of Hamburg (1990).
also appeared as: Dept. of Computer Science, University of Hamburg, Techn. Report No. FBI-HH-B-146/90.

[Ko 82] S.R.Kosaraju: *Decidability of reachability in vector addition systems*, Proc. 14th Ann. ACM STOC (1982), 267-281.

[KLM 89] H.Kleine-Büning, T.Lettmann, E.Mayr: *Projections of vector addition system reachability sets are semilinear*, TCS 64 (1989), 343-350.

[La 88] J.L.Lambert: *Consequences of the decidability of the reachability problem for Petri nets*, Advances in Petri Nets 1988, LNCS 340 (1988), 266-282.

[Ma 84] E.W.Mayr: *An algorithm for the general Petri net reachability problem*, SIAM Journal of Computation 13 (1984), 441-460.

[Mk 92] E.Mäkinen: *On the generative capacity of context-free matrix grammars over one-letter alphabet*, Fundamenta Informaticae 16 (1992), 93-97.

[GW 89] J.Gonczarowski, M.K.Warmuth: *Scattered versus context-sensitive rewriting*, Acta Informatica 27, (1989), 81-95.

[Ro 69] D.J Rosenkranz: *Programmed grammars and classes of formal languages*, Journal of the ACM 16 (1969), 107-131.

[vL 75] J.van Leeuwen: *Extremal properties of non-deterministic time complexity classes*, International Computing Symposium (E. Gelenbe and D.Poitier, eds.), North-Holland/American Elsevier (1975), 61-64.