Petri Net Controlled Finite Automata

Berndt Farwer
Department of Computer Science, Durham University, UK
berndt.farwer@durham.ac.uk

Matthias Jantzen, Manfred Kudlek, Heiko Rölke, Georg Zetzsche
Department Informatik, MIN-Fakultät, Universität Hamburg, DE
{jantzen,kudlek,roelke,zetzsch}@informatik.uni-hamburg.de

Abstract. We present a generalization of finite automata using Petri nets as control, called Concurrent Finite Automata for short. Several modes of acceptance, defined by final markings of the Petri net, are introduced, and their equivalence is shown. The class of languages obtained by λ-free concurrent finite automata contains both the class of regular sets and the class of Petri net languages defined by final marking, and is contained in the class of context-sensitive languages.

Keywords: Finite automata, Petri nets.

1. Introduction

In classical Turing machines the control is given by a finite automaton. It is an interesting idea to use as control a Petri net in order to introduce concurrency into automata theory. This leads to machines with the possibility of creating an arbitrary number of heads on the tape. The heads are represented by tokens of the Petri net pointing to positions on the tape (or vice versa). The heads can only be distinguished if they are associated to different places of the Petri net, or point to different tape positions. A computation step of such a concurrent machine can be performed only if all heads involved are on the same tape position and their corresponding tokens are located in the places forming the pre-condition of one of the Petri net’s transitions. This model and some results for concurrent Turing machines have been considered in [1, 2].

* Address for correspondence: Fachbereich Informatik, Universität Hamburg, Vogt-Kölln-Str. 30, D-22527 Hamburg, Germany
This model can be adopted in a straightforward manner to the simpler model of finite automata. Since finite automata allow the input word only to be read sequentially from left to right, the tape heads corresponding to the tokens put into the places of the post-condition of a transition will point to the tape position immediately to the right of the previous one, or – in the case of a \( \lambda \)-move – to the same position.

Section 2 gives the basic definitions of multisets, Petri nets, Petri net languages, concurrent finite automata, and acceptance modes for formal languages. In Section 3 we consider simple normal forms. In particular, we show that various application modes for transitions can be reduced to simple equivalent ones wrt. to acceptance. In particular we show that leftmost parallel application of transitions suffices. In Section 4 we explore some relationship to other well-known language families, such as Chomsky and Petri net languages, and present some illustrating examples.

2. Definitions

In this section we give the basic definitions of multisets, Petri nets, and concurrent finite automata.

Definition 2.1. (Multisets)

A multiset over a set \( A \) is a mapping \( \mu : A \rightarrow \mathbb{N} \). The set of all multisets over \( A \) is denoted by \( A^\mathbb{N} \). If \( A \) is finite then \( A^\mathbb{N} \) can be identified with \( \mathbb{N}^{|A|} \). Sometimes we use the alternative representation \( \mu = (\mu(a_1), \ldots, \mu(a_k)) \) with \( A = \{a_1, \ldots, a_k\} \), \( k = |A| \). If some of the \( \mu(a_j) = 0 \) also a notation \( \langle a_{j(1)}, \ldots, a_{j(m)} \rangle \) as ensemble with \( \{j(1), \ldots, j(m)\} \subseteq \{1, \ldots, k\} \) is used. The empty multiset is denoted by \( 0 \) and is defined by \( \forall a \in A : 0(a) = 0 \).

Definition 2.2. (Operations on multisets)

Let \( \mu \) and \( \nu \) be multisets. Their sum \( \mu \oplus \nu \) is defined by
\[
\forall a \in A : (\mu \oplus \nu)(a) = \mu(a) + \nu(a) .
\]

Clearly, \( \oplus \) is associative and commutative, with neutral element \( 0 \).

A partial order \( \sqsubseteq \) on multisets is given by
\[
\mu \sqsubseteq \nu \iff \forall a \in A : \mu(a) \leq \nu(a) .
\]

The difference \( \nu \ominus \mu \) of two multisets is defined by
\[
\forall a \in A : (\nu \ominus \mu)(a) = \max(\nu(a) - \mu(a), 0) .
\]

Furthermore, for \( k \in \mathbb{N} \) the multiset \( k\mu \) is defined by
\[
\forall a \in A : (k\mu)(a) = k\mu(a) .
\]

Definition 2.3. (P/T net)

A place/transition Petri net (P/T net) is a quadruple \( N = (P, T, g, \mu_0) \), where \( P \) is a finite set of places, \( T \) a finite set of transitions, \( g \) a mapping \( g : T \rightarrow \mathbb{N}^{|P|} \times \mathbb{N}^{|P|} \) assigning to each \( t \in T \) a pair \( g(t) = (\alpha, \beta) \), and \( \mu_0 \in \mathbb{N}^{|P|} \) the initial marking.

A transition \( t \in T \) with \( g(t) = (\alpha, \beta) \) is enabled at marking \( \mu \in \mathbb{N}^{|P|} \), if \( \alpha \sqsubseteq \mu \). In this case \( t \) can fire (be applied on) marking \( \mu \), yielding the new marking \( \mu' = (\mu \oplus \alpha) \oplus \beta \) where \( \mu, \mu' \in \mathbb{N}^{|P|} \).
Definition 2.4. (Petri net languages)

Petri net languages are defined by a P/T net $N = (P, T, g, \mu_0)$ and a transition labelling function $\sigma : T \to \Sigma$ or $\sigma : T \to \Sigma \cup \{\lambda\}$ with a finite alphabet $\Sigma$.

For a transition sequence $\tau \in T^*$ $\sigma$ is extended to $T^*$ and $\sigma(\tau) \in \Sigma^*$ denotes the canonical extension of $\sigma$. For a sequence $\tau = t_1 \cdots t_n$ there is a sequence of markings $\mu_0, \mu_1, \cdots, \mu_n$ corresponding to $\tau$ with $t_j$ leading from $\mu_j$ to $\mu_{j+1}$ for $0 \leq j < n$. To define a language several generating conditions are possible, e.g.

1. By deadlock (of the P/T net $N$).
2. By a final set of markings $\mathcal{F} \subseteq P^\oplus = \mathbb{N}[P]$.

This set should be recursive. In this case $\mu_n \in \mathcal{F}$ must hold for $\sigma(\tau)$ to be included in the language. Special cases are

(a) $\mathcal{F} = \mathbb{N}[P]$  
(b) $\mathcal{F}$ finite  
(c) $|\mathcal{F}| = 1$ singleton  
(d) $\mathcal{F} = \{\mu_f\}$ and $|\mu_f| = 1$.

We shall follow [4, 8] considering variants 2(c) and 2(d). In addition, it has been shown there that $|\mu_f| = |\mu_0| = 1$ suffices, too. Furthermore, only P/T nets with transitions $t$ with $g(t) = (\alpha, \beta)$ where $\alpha \neq 0$ will be considered.

Definition 2.5. (Petri net language classes)

The following classes of Petri net languages can be defined [4, 8]:

- $L_0^\lambda$ Petri net languages generated by Petri nets with final marking $\mu_f \neq \mu_0$, i.e. $\forall t \in T : \sigma(t) \neq \lambda$
- $\mathcal{CSS} = L_0^\bullet$ where $X^\bullet = X \cup \{L \cup \{\lambda\} \mid L \in X\}$ for any language class $X$.

Trivially, $L_0 \subset \mathcal{CSS} = L_0^\bullet \subseteq L_0^\lambda$.

Remark: If $\mu_0 \neq \mu_f$ is removed in the definition then $L_0 = L_0^\bullet = \mathcal{CSS}$.

Definition 2.6. (Concurrent finite automaton)

A Concurrent Finite Automaton (CFA) is a triple $C = (N, \Sigma, \sigma)$ where $N = (P, T, g, \mu_0)$ is a P/T net, $\Sigma$ is a finite tape alphabet, and $\sigma : T \to \Sigma \cup \{\lambda\}$ is a transition labelling function. Only P/T nets having transitions $t \in T$ with $g(t) = (\alpha, \beta)$ where $\alpha \neq 0$ are allowed. Transition $t$ with $\sigma(t) = \lambda$ are called $\lambda$-transitions, and such with $g(t) = (\alpha, 0)$ erasing transitions.

An expression $(\nu(0), x(1), \nu(1), \cdots, x(m), \nu(m), x(m+1), \nu(m+1))$ denotes a configuration of a concurrent finite automaton $C$, where $\nu(i) \in \mathbb{N}[P]$ for $0 \leq i \leq m+1$, $x(i) \in \Sigma$ for $1 \leq i \leq m$, and $x(m+1) = \# \not\in \Sigma$, the end marker of the word $w = x(1) \cdots x(m)$ on the tape. In most cases it will be abbreviated by $\nu(0)x(1)\nu(1) \cdots x(m)\nu(m)x(m+1)\nu(m+1)$. Thus $\nu : \mathbb{N} \to \mathbb{N}[P]$ is a function attaching to each tape position a multiset. $\bigoplus_{i=0}^{m+1} \nu(i)$ represents the marking of the Petri net $N$. The initial configuration is given by $\mu_0x(1)0 \cdots 0x(m)0\#0$. $\#$ is introduced to have a uniform notation also for other kinds of automata where end markers are necessary.
Each token in a place of the P/T net represents a head on the tape. The multiset \( \nu(i) \) represents the fact that there are heads of corresponding multiplicities on position \( i \) of the tape. In each step a transition of the P/T net \( N \) fires, taking away tokens from input places and corresponding heads from position \( i \) on the tape, and putting tokens on output places and corresponding heads on position \( i + 1 \), or \( i \) if \( \lambda \)-transitions are concerned. Thus heads can move only to the right, or stay on the same position.

In the simplest application of a transition the CFA works sequentially. Only tokens corresponding to one tape position \( i \) can be taken away, and new tokens put have to correspond to tape position \( i + 1 \), or \( i \) in case of \( \lambda \)-transitions. Then the successor configuration after applying a transition \( t \) with \( g(t) = (\alpha, \beta) \) and \( \sigma(t) = x_i \) (or \( \sigma(t) = \lambda \)) is given by

\[
\nu'(i) = \nu(i) \ominus \alpha, \quad \nu'(i+1) = \nu(i+1) \oplus \beta \quad \text{and} \quad \nu'(j) = \nu(j) \quad \text{for} \quad j \neq i, j \neq i + 1,
\]

or \( \nu'(i) = \nu(i) \ominus \alpha \oplus \beta \) and \( \nu'(j) = \nu(j) \) for \( j \neq i \), respectively.

Several acceptance modes, analogous to Petri nets, can be defined here too.

1. Acceptance by deadlock (of the P/T net \( N \)).
2. Acceptance by a final set of markings \( \mathcal{F} \subseteq P^{\oplus} = N^{[P]} \). Again, this set should be recursive.

Special cases are

(a) \( \mathcal{F} \) finite

(b) \( |\mathcal{F}| = 1 \) singleton

(c) \( \mathcal{F} = \{\mu_f\} \) with final configuration \( 0x_1 \cdots 0x_m0\#\mu_f \)

(d) \( \mathcal{F} = \{\mu_f\} \) with final configuration \( 0x_1 \cdots 0x_m0\#\mu_f \) and \( |\mu_f| = 1 \).

In any case, reaching a final marking, at least one head has to be on position after \( \# \), i.e. the entire input word has to be read (2(b)). Another possibility is that all heads represented by a final marking have to be on position after \( \# \) (2(c), 2(d)).

Note that in a place \( p \in P \) there can be tokens pointing to different tape positions. Note also that in the case \( \sigma : T \rightarrow \Sigma \) (\( \lambda \)-free CFA) the accepted languages can also contain \( \lambda \).

Classes of languages accepted by such CFA can be defined as follows. Following again [4, 8] we shall consider only acceptance modes 2(b), 2(c), and 2(d), and mainly only CFAs without erasing transitions. Because of the lemmas to follow the final marking \( \mu_f \) is not included in the definition of \( N \).

**Definition 2.7.** (Language classes)

\[
\begin{align*}
C_{01}^\lambda & \text{ languages accepted by CFAs in mode 2(b)} \\
C_{02}^\lambda & \text{ languages accepted by CFAs in mode 2(c)} \\
C_{03}^\lambda & \text{ languages accepted by CFAs in mode 2(d)} \\
C_{01} & \text{ languages accepted by \( \lambda \)-free CFAs in mode 2(b)} \\
C_{02} & \text{ languages accepted by \( \lambda \)-free CFAs in mode 2(c)} \\
C_{03} & \text{ languages accepted by \( \lambda \)-free CFAs in mode 2(d)}.
\end{align*}
\]

Trivially, \( C_{03} \subseteq C_{02} \subseteq C_{01} \) and \( C_{03}^\lambda \subseteq C_{02}^\lambda \subseteq C_{01}^\lambda \), as well as \( C_{0i}^\lambda \subseteq C_{0i}^\lambda \) for \( 1 \leq i \leq 3 \).
Language classes for which transitions \( t \) with \( g(t) = (\alpha, 0) \) are allowed, will be denoted by \( C^\alpha_{0i} \) or \( C^\lambda_{0i} \).

Similarly, \( C^\lambda_{03} \subseteq C^\lambda_{02} \subseteq C^\lambda_{01} \) and \( C^\alpha_{03} \subseteq C^\alpha_{02} \subseteq C^\alpha_{01} \), as well as \( C^\alpha_{0i} \subseteq C^\lambda_{0i} \) for \( 1 \leq i \leq 3 \).

The classes of regular, context-free, context-sensitive, and recursively enumerable languages are denoted by \( \text{REG}, \text{CF}, \text{CS}, \text{and RE}, \) respectively.

So far the application of transitions has been defined in a sequential manner. However, it is also possible to apply transitions in a parallel (concurrent) way.

**Definition 2.8. (Parallel transitions)**

Consider the set \( T = \{t_1, \ldots, t_{|T|}\} \) with \( g(t_j) = (\alpha_j, \beta_j) \in \mathbb{N}^{|P|} \times \mathbb{N}^{|P|} \) for \( 1 \leq j \leq |T| \). For a multiset \( \tau = \bigoplus_{j=1}^{|T|} \tau(t_j) \) of transitions define \( g(\tau) = (\alpha_\tau, \beta_\tau) \) by \( \alpha_\tau = \bigoplus_{j=1}^{|T|} \tau(t_j) \alpha_j \) and \( \beta_\tau = \bigoplus_{j=1}^{|T|} \tau(t_j) \beta_j \).

In a distributed parallel application \( \tau = \bigoplus_{i=1}^m \tau_i \) where \( \tau_i = \bigoplus_{j=1}^{|T|} \tau_i(t_j) t_j \). This means that if \( \nu(0)x(0) \cdots x(m)x(m+1)\nu(m+1) \) is a configuration, then the application of \( \tau \) is defined in the following way:

\[
\sigma(t_j) = x(i) \text{ for } t_j \text{ with } \tau_i(t_j) > 0, \alpha_{\tau_i} \subseteq \nu(i), \nu'(i) = \nu(i) \ominus \alpha_{\tau_i},
\]

and in the case of \( \lambda \)-transitions (at some tape position)

\[
\sigma(t_i) = \lambda \text{ for } t_i \text{ with } \tau_i(t_j) > 0, \nu'(i) = \nu(i) \ominus \alpha_{\tau_i} \oplus \beta_{\tau_i}.
\]

A parallel application at one tape position \( i \) only, is defined as follows:

\[
\tau = \bigoplus_{j=1}^{|T|} \tau(t_j) t_j \text{ where } \sigma(t_j) = x(i) \text{ for } t_j \text{ with } \tau(t_j) > 0.
\]

Note that both modes contain applications of single (sequential) transitions as special case.

In the last case leftmost application is possible, too. This means that the transitions are applied for the position with smallest \( i \) first, as long as possible.

Trivially, in this case parallel and sequential application are equivalent since a sequence of applications of single transitions can be also achieved by one parallel application, and vice versa.

### 3. Normal Forms

In this section we show that general forms of CFAs can be reduced to simple equivalent (w.r.t. language acceptance) normal forms. This will be done first for sequential application of transitions.

A simple form is 1 initial place and 1 final place only.

**Lemma 3.1.** CFAs having one initial place \( p_0 \) with initial marking \( (p_0) \) suffice.

**Proof:**

In the case of \( C^\lambda_{0i} \) add a new place \( p_0 \) as well as a new transition \( t_0 = ((p_0), \mu_0) \) with \( \sigma(t_0) = \lambda \).

In the case of \( C^\lambda_{0i} \) let \( M_0 = \{ \nu \in P^\oplus \mid \exists t \in T : g(t) = (\alpha, \beta), \alpha \subseteq \mu_0, \alpha \subseteq \mu_0, \nu = \mu_0 \ominus \alpha \ominus \beta \} \) as well as \( T_0 = \{ t \in T \mid g(t) = (\alpha, \beta), \mu_0 \ominus \alpha \ominus \beta \in M_0 \} \), the set of initial transitions. Add a new initial place \( p_0 \) and define new initial transitions by
Lemma 3.2. \( T'_0 = \{ [t,x] | t \in T_0, x \in \Sigma, g([t,x]) = (\langle p_0 \rangle, \mu_0 \oplus \alpha \oplus \beta), g(t) = (\alpha, \beta), \sigma([t,x]) = \sigma(t) = x \}. \)

Let \( T' = (T \setminus T_0) \cup T'_0 \). In both cases the initial marking is \( \langle p_0 \rangle \) and the accepted languages are the original ones.

The same construction holds for \( C_{0i}^{\lambda} \) and \( C_{0i}^{\lambda} \) as well. \( \Box \)

Lemma 3.2. For the cases \( C_{02}, C_{02}, C_{02}^{\lambda}, \) and \( C_{02}^{\lambda} \) one final place suffices, i.e. \( C_{02} \subseteq C_{03}^{\lambda}, C_{02} \subseteq C_{03}, C_{02}^{\lambda} \subseteq C_{03}^{\lambda}, \) and \( C_{02}^{\lambda} \subseteq C_{03}^{\lambda} \).

Proof:
Note that for \( C_{02}^{\lambda} \) the end marker \( \# \) has to be read. Define the set of final markings as \( M_{f} = \{ \nu \in P_{\oplus} | \exists t \in T : g(t) = (\alpha, \beta), \nu = \mu_{f} \oplus \alpha \oplus \beta \} \) and consider the set of final transitions \( T_{f} = \{ t \in T | g(t) = (\alpha, \beta), \sigma(t) = \# , \mu_{f} \oplus \alpha \oplus \beta = \nu \} \).

1. Add a final place \( p_{f} \) and define \( T'_{f} = \{ t' | g(t') = (\nu, \langle p_{f} \rangle), t \in T_{f}, \sigma(t') = \# \} \), the new set of final transitions. Let \( T' = (T \setminus T_{f}) \cup T'_{f} \). Then the same language is accepted with final marking \( \langle p_{f} \rangle \).

The construction also holds for \( C_{02}, C_{02}^{\lambda}, \) and \( C_{02}^{\lambda} \). \( \Box \)

Lemma 3.3. \( C_{01}^{\lambda} \subseteq C_{02}, C_{01} \subseteq C_{02}, C_{01}^{\lambda} \subseteq C_{02}^{\lambda}, \) and \( C_{01}^{\lambda} \subseteq C_{02}^{\lambda} \).

Proof:
For each place \( p \) add a new place \( \overline{p} \), and for each place \( p \) and symbol \( x \in \Sigma \) new transitions \( [p,x] \) with \( g([p,x]) = (\langle p \rangle, \langle \overline{p} \rangle) \) and \( \sigma([p,x]) = x \), as well as \( \langle \overline{p},x \rangle \) with \( g([\overline{p},x]) = (\langle \overline{p} \rangle, \langle \overline{p} \rangle) \) and \( \sigma([\overline{p},x]) = x \). These transitions transfer tokens for the final marking from \( p \) to \( \overline{p} \) and transport them to the right, remaining on \( \overline{p} \). Add also transitions \( [\overline{p}, \#] \) with \( g([\overline{p}, \#]) = (\langle \overline{p} \rangle, \langle \overline{p} \rangle) \) and \( \sigma([\overline{p}, \#]) = \# \).

Let \( \overline{p} \) be defined by \( \overline{p}(p) = 0 \) and \( \overline{p}(\overline{p}) = \mu(p) \).

1. Add a new place \( p_{s} \). For each transition \( t \) with \( g(t) = (\alpha, \beta) \) and \( \sigma(t) = \# \), i.e. those for markings related to the end marker, add a new transition \( t' \) with \( g(t') = (\alpha, \beta \oplus p_{s}) \) and \( \sigma(t') = \# \). These allow that at least 1 token will be put on \( p_{s} \).

Taking now the new final marking \( \overline{p}_{f} \oplus \langle p_{s} \rangle \) ensures that at least one of the original final transitions has fired. The other contributions to \( \overline{p}_{f} \) either come from other final transitions or from other positions on the tape.

Thus the new CFA accepts the same language with final marking \( \overline{p}_{f} \oplus \langle p_{s} \rangle \). Note that this holds for \( \lambda \)-free CFAs as well as for CFAs with \( (0,0) \)-transitions. \( \Box \)

From the previous lemmas we derive the following theorem, yielding only four implicitly defined language classes, denoted by \( C_{0}^{\lambda}, C_{0}, C_{0}^{\lambda}, \) and \( C'_{0} \), respectively.

Theorem 3.1. \( C_{01}^{\lambda} = C_{02}^{\lambda} = C_{03}^{\lambda} = C_{0}^{\lambda}, \) \( C_{01} = C_{02} = C_{03} = C_{0}, \)

as well as \( C'_{01} = C'_{02} = C'_{03} = C'_{0}^{\lambda}, \) \( C'_{01} = C'_{02} = C'_{03} = C'_{0}, \)

Furthermore, one initial and one final place suffice. \( \Box \)

Another result is

Theorem 3.2. \( C_{0}^{\lambda} = C_{0}, C_{0}^{\lambda} = C_{0}^{\lambda}, C_{0}^{\lambda} = C'_{0}, C_{0}^{\lambda} = C_{0}^{\lambda}. \)
Proof:
Let \( (N, \Sigma, \sigma) \) be a \( \lambda \)-free CFA with \( N = (P, T, g, \mu_0) \) and accepted language \( L \in C_0 \). Add to \( T \) a transition \( t' \) with \( g(t') = (\mu_0, \langle p_f \rangle) \), \( \sigma(t') = \# \) where \( p_f \in P \) is the final place. Then the new accepted language is \( L' = L \cup \{ \lambda \} \in C_0 \).

In the same way this follows for \( C_0^\lambda \), \( C'_0 \), and \( C'_0^\lambda \).

Finally, we show that for \( C_0 \), \( C_0^\lambda \), \( C'_0 \), and \( C'_0^\lambda \) the leftmost maximal parallel mode of application of transitions suffices.

Lemma 3.4. Languages from all languages classes \( C_0 \), \( C_0^\lambda \), \( C'_0 \), and \( C'_0^\lambda \) can also be accepted if transitions are applied in a leftmost maximal parallel manner.

Proof:
First note that tokens related to a position \( j \) on the tape don’t influence transitions applied on configurations related to positions \( i < j \). Since mode 2d suffices, all places except the final one \( p_f \) have to be emptied, resulting in the final configuration \( \langle p_f \rangle \). Therefore, transitions can be applied in a leftmost maximal parallel way, removing all tokens related to one position \( i \). Otherwise there would remain some tokens related to \( i \) without possibility to be removed later. If \( \lambda \)-transitions are concerned, these can be applied several times before all new tokens point to \( i + 1 \).

The following example illustrates how a CFA is working.

Example 3.1. \( L_1 = \{ a^n b^n \mid n \geq 0 \} \in C_0 \).

The CFA \( C_1 \) is given in Figure 1. Initial marking is \( \langle 0 \rangle \), final marking \( \langle 3 \rangle \). \( C_1 \) starts in 0 and counts the \( a \)'s in 1. When 0 encounters \( b \) one token is put into 2, and then tokens in 1 are reduced whenever a \( b \) is encountered, until the end marker \( \# \) is found.

4. Relations to other Language Classes

In this section we present some relationships to other language classes, such as regular sets, context-sensitive languages, and Petri net languages.
**Theorem 4.1.** All regular languages \( \text{REG} \) are accepted by \( \lambda \)-free CFAs with final-marking condition in mode 2(d), i.e. \( \text{REG} \subseteq \mathcal{C}_0^* = \mathcal{C}_0 \).

**Proof:**
Without loss of generality consider a DFA \( A = (\Sigma, Q, q_0, Q_F, \delta) \) accepting \( L(A) = L \), and define a CFA \( C = (N, \Sigma, \sigma) \) by \( N = (P, T, g, \mu_0) \) with \( P = \{ q \mid q \in Q \} \cup \{ p_f \}, T = T_A \cup T_\lambda \) where the transitions are given by \( T_A = \{ [q, x, r] \mid \delta(q, x) = r \} \cup \{ [q, \#, p_f] \mid q \in Q_F \}, g([q, x, r]) = ([q], ([r])), \) with \( \sigma([q, x, r]) = x, \sigma([q, \#, p_f]) = \# \), and \( T_\lambda = \{ t_f \}, g(t_f) = (0, (q_0, (p_f))) \) with \( \sigma(t_f) = \# \) if \( \lambda \in L \), \( T_\lambda = \emptyset \) if \( \lambda \notin L \).

Then \( C \) simulates \( A \), having in each configuration exactly one token in some \( q \in P \), corresponding to the configuration of \( A \). \( \square \)

From example 1 follows

**Theorem 4.2.** \( \text{REG} \subseteq \mathcal{C}_0 \).

**Theorem 4.3.** All Petri net languages are accepted by CFAs with final-marking condition and the final set corresponding to that of the Petri net, i.e. \( \mathcal{L}_0^\lambda \subseteq \mathcal{C}_0^\lambda, \mathcal{L}_0 \subseteq \mathcal{C}_0, \mathcal{CSS} \subseteq \mathcal{C}_0^* = \mathcal{C}_0 \).

**Proof:**
Let \( N = (P, T, g, \mu_0) \) be a P/T-net with transition labelling function \( \sigma : T \rightarrow \Sigma \cup \{ \lambda \} \). Consider this as the underlying P/T net of a CFA \( C \). Note that for Petri net languages the last symbol \( x \) of a word has to be generated by a \( \lambda \)-free transition labelled with \( x \), possibly followed by \( \lambda \)-transitions. Let the final marking be \( \mu_f \).

Construct a CFA \( C = (N', \Sigma, \sigma') \) where \( N' = (P', T', g', \mu'_0) \) with places \( P' = P \cup \{ q \} \), transitions \( T' = T \cup \{ [p, \alpha] \mid p \in P, \alpha \in \Sigma \} \cup \{ t_f \} \) where \( \mu' = (\mu'_0, \mu'_f), g'(t) = (\alpha \oplus \langle q \rangle, \beta \oplus \langle q \rangle) \) if \( g(t) = (\alpha, \beta), \sigma'(t) = \sigma(t) \), with \( g'([p, \alpha]) = ([p], \langle \alpha \rangle) \), \( \sigma'([p, \alpha]) = \alpha, \sigma'(t) = \sigma(t) \) for \( t \in T \), \( \mu'_0 = \mu_0, \mu'_f = \mu_f, g'(t_f) = (\mu_f \oplus \langle q \rangle, \mu'_f) \), and \( \sigma'(t_f) = \# \).

Then \( w \in \Sigma^* \) is accepted by \( C \) iff \( w \) is generated by \( N \). \( \square \)

**Theorem 4.4.** \( \mathcal{C}_0 \subseteq \mathcal{CS} \).

**Proof:**
Consider a CFA \( C = (N, \Sigma, \sigma) \) with Petri net \( N = (P, T, \mu_0) \). Define \( m = |\mu_0| \) as well as \( k = \max\{ \left| \frac{s(t)}{t} \right| \mid (\alpha, \beta) \in T' \} \).

If \( \nu(0) x_1 \nu(1) \cdots x_n \nu(n) \# \nu(n + 1) \) is any configuration in a marking sequence \( \mu_0, \mu_1, \cdots, \mu_s \) with \( \mu_i \rightarrow \mu_{i+1} \), i.e. \( \sum_{j=0}^{n} \nu(j) = \mu_i \) for some \( 0 \leq i \leq s \), then \( |\nu_j| \leq m \cdot k^j \) for \( 0 \leq j \leq n + 1 \). Actually, this bound can be achieved only if \( \nu_i = 0 \) for \( 0 \leq i < j \). The total number of tokens in any reachable configuration is bounded by \( m \cdot \sum_{j=1}^{n+1} k^j = \frac{m \cdot k^{n+2} - 1}{k-1} \leq m \cdot \sum_{j=1}^{n+1} k^j = m' \cdot k^n \). This needs at most linear space for binary encoding.

Since transitions using tokens in \( \nu_i \) with \( 0 \leq i < j \) are not influenced by transitions using tokens in \( \nu_j \), it is always possible to fire those if activated before firing transitions using tokens in \( \nu_j \). If no transition is activated there can remain some tokens yielding a multiset \( \mu'_i \). However, for the contribution to the final marking some transitions can remain activated. Starting on position 0 these remaining markings will be successively added, eventually summing up in the final marking \( \mu_f \).
The CFA $C$ can be simulated by a nondeterministic LBA $A$ that has the encoding of the P/T net on track 1 of its tape (this requires at most a constant amount of space), and the input $w\#$ of $C$ on track 2 of its tape, and uses another track (track 3) for working on the input. This involves in particular marking the two adjacent positions on which $C$ is currently working. Both, track 2 and 3 need at most a linear amount of space w.r.t. the length of the input. Track 4 and 5 contain the markings of the two adjacent positions. Since $|P|$ is finite again at most linear space is required.

A nondeterministically changes the two adjacent positions to the right by one position. The remaining marking on the left position is added to the multiset on track 6. After that the content of track 4 is erased and the content of track 5 written on track 4, while the content of track 5 is erased. All tracks 4, 5, 6 only require linear space.

After firing at least one transition with label $\#$, the multiset $\mu'$ on track 6 is compared with $\mu_f$. If $\mu' = \mu_f$ then $w$ is accepted. Otherwise more transitions with label $\#$ may be fired. If always $\mu' \neq \mu_f$ then $w$ is rejected. 

Since a transition $(\alpha, 0)$ doesn’t change the constant $k$ in Theorem 4.4 we also get

**Corollary 4.1.** $C'_0 \subseteq CS$. 

Two other examples exhibit strict inclusions of Petri net languages in language classes defined by CFAs.

**Example 4.1.** $L_2 = \{w \in \{a, b\}^*c^k \mid k = \text{bin}(w)\} \in C_0$ where $\text{bin}(w) \in \mathbb{N}$ is the integer represented in binary by $w$ with $a, b$ for 0, 1. The CFA $C_2$ is shown in Figure 2, with initial marking $\langle 0 \rangle$, final marking $\langle 3 \rangle$. $C_2$ interpretes $w$ as a binary encoding of $n$ ($a$ for 0, $b$ for 1), putting for each $b$ a token on place 2. These can move to the right, doubling at each step, until reaching a $c$. Then there are exactly $\text{bin}(w)$ tokens in 2. They are consumed by the $c$’s when moving right.

**Example 4.2.** $L_3 = \{w \in \{a, b\}^*c^k \mid 0 \leq k \leq \text{bin}(w)\} \in C_0$. The CFA $C_3$ is presented in Figure 3, with initial marking $\langle 0 \rangle$, final marking $\langle 3 \rangle$. $C_3$ works similar to $C_2$. 

![Figure 2](image1.png)

![Figure 3](image2.png)
Since $L_3 \not\in L_0^\uparrow$ and $L_2 \not\in L_0^\downarrow$, and by [3, 5], the following theorem is implied.

**Theorem 4.5.** $L_0^\uparrow \subset C_0$, $C_0 \not\subset L_0^\downarrow$, and $L_0^\downarrow \subset C_0$.

Finally, we present two other examples both demonstrating $C_0 \not\subseteq CF$.

**Example 4.3.** $L_4 = \{a^n b^{2^n} \mid n \geq 0\} \in C_0$. The CFA $C_4$ is shown in Figure 4, with initial marking $\langle 0 \rangle$, final marking $\langle 5 \rangle$. $C_4$ puts 2 tokens on place 2 for each $a$ read. These tokens double when moving to the right until reaching $b$ such that then $2^n$ tokens are in 2. After that they can move to the right and are consumed by the $b$'s.

**Example 4.4.** $L_9 = \{wcw \mid w \in \{a, b\}^*\} \in C_0'$. The CFA is given in Figure 5, with initial marking $\langle 0 \rangle$, final marking $\langle 7 \rangle$. $C_9$ first produces from the input $ucv \# 2^{|w|} + bin(u)$ tokens in place 2 which are shifted to 5, and after reading $c 2^{|v|} + bin(v)$ tokens in place 4 which are erased one by one using a transition $t$ with $g(t) = (\#, 0)$. Tokens remain in 4 or 5 iff $u \neq v$. 
From the considerations above the following language hierarchy follows where the relations among Petri net languages are from [3, 5]:

\[
L_0^{\lambda} \rightarrow CSS \rightarrow L_0 \rightarrow \lambda \rightarrow C_0 \rightarrow \lambda \rightarrow C_0^{\lambda} \rightarrow CS \rightarrow RE
\]

References


