Concurrent Finite Automata

Matthias Jantzen[*] Manfred Kudlek[*] Georg Zetzsche[*]

January 22, 2009

Abstract

We present a generalization of finite automata using Petri nets as control. Acceptance is defined by final markings of the Petri net. 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.

1 Introduction

In classical finite automata, the input is read by one head that moves across one symbol in every step. In order to investigate the impact of concurrency on automata accepting languages, our generalization of finite automata allows arbitrarily many heads that can move concurrently. These heads are distributed across the input and in particular, different parts of the input can be processed at the same time. A similar model, that applies the idea of multiple independent heads to Turing machines instead of finite automata is investigated in [FKR06].

The concurrency is achieved by using a Petri net that describes the movement of the heads. For every position on the input, there is a multiset of heads, which is interpreted as a marking of the Petri net. If a transition fires at some position of the input, the corresponding preset is removed from that position and the postset is added at the next position. In contrast to multi-head automata, there is no global state, since the ability to fire only depends on the heads at the respective position. Therefore, different parts of the word can be processed concurrently.

It turns out that this model is equivalent to an automaton which, for every symbol on the input, solves a system of algebraic equations and applies

[*]Department Informatik, Universität Hamburg, E-Mail: {jantzen,kudlek,3zetzsch}@informatik.uni-hamburg.de

some homomorphism to the solution to obtain the next configuration. These two definitions are presented in section 2. Section 3 and 4 give an overview of the results obtained so far concerning the languages accepted by CFA.

2 Definitions

Definition 1. *A set M with an operation $+ : M \times M \to M$ is a* monoid*, if the operation is associative and there is a neutral element $0 \in M$ for which $0+x = x+0 = x$ for every $x \in M$. M is called* commutative*, if $x+y = y+x$ for all $x, y \in M$. For $x, y \in M$, let $x \sqsubseteq y$ iff there is a $z \in M$ with $y = x+z$.*

For every set A, we have the set $A^{\oplus}$ of mappings $\mu : A \to \mathbb{N}$. The elements of $A^{\oplus}$ are called multisets *over A. With the operation $\oplus$, defined by $(\mu \oplus \nu)(a) = \mu(a) + \nu(a)$, $A^{\oplus}$ becomes a commutative monoid with the neutral element $\mathbf{0}$, $\mathbf{0}(a) := 0$ for every $a \in A$. In the case $\mu \sqsubseteq \nu$ we can define $(\nu \ominus \mu)(a) := \nu(a) - \mu(a)$ for all $a \in A$. If A is finite, let $|\mu| := \sum_{a \in A} \mu(a)$. These definitions are carried over to $\mathbb{N}^k$ by noting that $\mathbb{N}^k \cong \{a_1, \dots, a_k\}^{\oplus}$.*

A concurrent finite automaton is given by the following data.

Definition 2. *A* CFA *is a sextuple $C = (\Sigma, N, \sigma, \mu_0, \mathcal{F}, \#)$, where*

- *Σ is an alphabet and $\# \notin \Sigma$ is the* end marker *symbol,*
- *$N = (P, T, \partial_0, \partial_1)$ is a Petri net, where P (T) is the set of places (transitions) and $\partial_0, \partial_1 : T^{\oplus} \to P^{\oplus}$ are homomorphisms that specify the pre- and post-multisets. Furthermore, $\partial_0(t) \neq \mathbf{0}$ for every $t \in T$. In the case that $\partial_1(t) \neq \mathbf{0}$ for every $t \in T$, C is called* non-erasing*.*
- *$\sigma : T \to \Sigma \cup \{\lambda\}$ defines the corresponding symbol for every transition. A transition $t \in T$ is called* λ-transition*, if $\sigma(t) = \lambda$. C is called* λ-free*, if it does not contain λ-transitions.*
- *$\mu_0 \in P^{\oplus}$ is the* initial marking *and $\mathcal{F}$ is a finite set of* final markings*.*

Now we give two definitions of the accepted language that are equivalent, that is, the same language classes result from these definitions. The first one directly describes the firing of the transitions in the underlying net.

Definition 3. *Let $C = (\Sigma, N, \sigma, \mu_0, \mathcal{F}, \#)$ a CFA, where $N = (P, T, \partial_0, \partial_1)$. Then a* configuration *is a tuple $(\nu_0, a_1, \nu_1, \dots, a_n, \nu_n, \#, \nu_{n+1})$, where $\nu_0, \dots, \nu_{n+1} \in P^{\oplus}$ and $a_1, \dots, a_n \in \Sigma$. On the set of configurations of C, we define the binary relation $\xrightarrow[C]{}$. It describes the firing of one transition. Let*

$$(\nu_0, a_1, \nu_1, \dots, a_n, \nu_n, \#, \nu_{n+1}) \xrightarrow[C]{} (\nu'_0, a_1, \nu'_1, \dots, a_n, \nu'_n, \#, \nu'_{n+1})$$

Operator	$\mathcal{C}_0$	$\mathcal{C}'_0$	$\mathcal{C}^\lambda_0$
$\cap R$	+	+	+
$L_1 \cdot L_2$	+	?	+
$\cup$	+	+	+
$\cap$	+	+	+
h^{-1}	+	+	?
h	-	-	-
λ-free h	?	?	+

$$\begin{tikzcd}
 & & \mathbf{RE} & \\
 & \mathcal{C}^\lambda_0 = \mathcal{C}'^\lambda_0 \arrow[ur] & & \mathbf{CS} \arrow[ul] \\
\mathbf{CF} \arrow[ur] & & \mathcal{C}'_0 \arrow[ul, no head] \arrow[ur] & \\
 & \mathcal{L}^\lambda_0 \arrow[uu] & \mathcal{C}_0 \arrow[u, no head] & \\
 & \mathcal{CSS} = \mathcal{L}^\blacktriangle_0 \arrow[u] \arrow[ur] & & \\
\mathbf{REG} \arrow[uuu] \arrow[ur] & \mathcal{L}_0 \arrow[u] & &
\end{tikzcd}$$

Figure 1: Closure properties and relations of the language classes.

iff there are $t \in T, j \in \{1, \ldots, n+1\}$ and one of the following conditions holds:

- *$\sigma(t) = a_j$ (where $a_{n+1} = \#$), $\nu'_{j-1} = \nu_{j-1} \ominus \partial_0(t)$, $\nu'_j = \nu_j \oplus \partial_1(t)$, and $\nu'_i = \nu_i$ for every $i \neq j$.*
- *$\sigma(t) = \lambda$ and $\partial_0(t) \sqsubseteq \nu_j$, $\nu'_j = \nu_j \ominus \partial_0(t) \oplus \partial_1(t)$ and $\nu'_i = \nu_i$ for every $i \neq j$.*

So the firing of a non-λ-transition moves tokens across one symbol whereas λ-transitions work like ordinary Petri net transitions on the multiset at one position. Then for $w \in \Sigma^*$, $w = a_1 \cdots a_n$, $a_1, \ldots, a_n \in \Sigma$, it is $w \in L_1(C)$ iff $(\mu_0, a_1, \mathbf{0}, \ldots, a_n, \mathbf{0}, \#, \mathbf{0}) \xrightarrow[C]{*} (\mathbf{0}, a_1, \mathbf{0}, \ldots, a_n, \mathbf{0}, \#, \mu)$ for some $\mu \in \mathcal{F}$, where $\xrightarrow[C]{*}$ denotes the reflexive transitive closure of $\xrightarrow[C]{}$. Thus, a word is accepted if all the heads are on the rightmost position and a final marking has been reached.

One possible variant is to accept a word also if the final marking is reached in a distributed manner. So for $w \in \Sigma^*$, $w = a_1 \cdots a_n$, $a_1, \ldots, a_n \in \Sigma$, let $w \in L_2(C)$ iff $(\mu_0, a_1, \mathbf{0}, \ldots, a_n, \mathbf{0}, \#, \mathbf{0}) \xrightarrow[C]{*} (\nu_0, a_1, \nu_1, \ldots, a_n, \nu_n, \#, \nu_{n+1})$, where $\nu_0 \oplus \cdots \oplus \nu_{n+1} \in \mathcal{F}$ and $\nu_{n+1} \neq \mathbf{0}$.

Now we present another definition, where the automaton solves a system of algebraic equations and obtains the next configuration by applying a homomorphism to the solution.

Definition 4. *Using the notation $T_x := \{t \in T \mid \sigma(t) = x\}$ for $x \in \Sigma \cup \{\lambda\}$, we define for every CFA C the binary relation $\underset{C}{\Longrightarrow}$ on $\Sigma^* \times P^\oplus$ by*

$$(w, \mu) \underset{C}{\Longrightarrow} (wa, \mu') \text{ if } \exists v \in T_a^\oplus : \partial_0(v) = \mu \wedge \partial_1(v) = \mu',$$

for $w \in (\Sigma \cup \{\#\})^*,\ a \in \Sigma,\ \mu, \mu' \in P^\oplus$ and

$$(w, \mu) \underset{C}{\Longrightarrow} (w, \mu') \text{ if } \exists t \in T_\lambda : \partial_0(t) \sqsubseteq \mu \wedge \mu' = \mu \ominus \partial_0(t) \oplus \partial_1(t),$$

for $w \in (\Sigma \cup \{\#\})^*$ and $\mu, \mu' \in P^\oplus$. If $\overset{*}{\underset{C}{\Longrightarrow}}$ denotes the reflexive transitive closure of $\underset{C}{\Longrightarrow}$, then the accepted language of C is

$$L_3(C) := \{w \in \Sigma^* \mid \exists \mu \in \mathcal{F} : (\lambda, \mu_0) \overset{*}{\underset{C}{\Longrightarrow}} (w\#, \mu)\}.$$

Now it is not hard to see that $L_3(C) = L_1(C)$ for every CFA C. Furthermore, it can be shown that the following definitions of language classes accepted by different types of CFA do not depend on whether one chooses $L_1(C)$ or $L_2(C)$ as the language of C. By $\mathcal{C}_0$ we denote the class of languages accepted by non-erasing λ-free CFA. $\mathcal{C}_0'$ is the class of languages accepted by λ-free CFA, $\mathcal{C}_0^\lambda$ the class of languages accepted by non-erasing CFA, and $\mathcal{C}_0'^\lambda$ the class of languages accepted by arbitrary CFA. **REG**, **CF**, **CS**, **RE** denotes the class of regular, context-free, context-sensitive, recursively enumerable languages, respectively. Then let $\mathcal{L}_0^\lambda$, $(\mathcal{L}_0)$ be the class of Petri net languages generated by (λ-free) Petri nets with final marking, as defined in [Hack76]. For alphabets Σ, Γ, a homomorphism $h : \Sigma^* \to \Gamma^*$ is called *coding*, if $h(a) \in \Gamma$ for all $a \in \Sigma$. For a language class $\mathcal{L}$, let $\hat{\mathcal{H}}(\mathcal{L})$ $(\mathcal{H}^{cod}(\mathcal{L}))$ be the class of all languages $h(L)$ with $L \in \mathcal{L}$, where h is an arbitrary homomorphism (coding).

3 Relations To Other Language Classes

By a simple construction, one obtains $\mathcal{C}_0' \subseteq \mathcal{C}_0^\lambda$ and $\mathcal{C}_0^\lambda = \mathcal{C}_0'^\lambda$. Furthermore, it is easy to see that (λ-free) Petri nets can be simulated by (λ-free) CFA. These inclusion is even proper, so we have $\mathcal{L}_0 \subset \mathcal{C}_0$ and $\mathcal{L}_0^\lambda \subset \mathcal{C}_0^\lambda$. While it is still open whether all context-free languages are accepted by CFA, the application of codings allows a construction similar to one used for push-down-automata: For every context-free language L, there is a coding h and a non-erasing λ-free CFA C such that $L = h(L(C))$. Thus, **CF** $\subseteq \mathcal{H}^{cod}(\mathcal{C}_0)$. Applying arbitrary homomorphisms to languages accepted by CFA leads to the whole class of recursively enumerable languages: For every recursively enumerable language L, there is a (possibly erasing) homomorphism h and a non-erasing λ-free CFA C such that $L = h(L(C))$. Thereby, h and C are effectively constructible. In particular, it is $\mathbf{RE} = \hat{\mathcal{H}}(\mathcal{C}_0)$ and the emptiness problem is undecidable even for non-erasing λ-free CFA. In contrast, the word problem is decidable for every type of CFA and therefore we have the proper inclusion $\mathcal{C}_0 \subset \mathbf{RE}$. For every language of a λ-free CFA, there is an algorithm accepting it in linear space and quadratic time, so we have $\mathcal{C}'_0 \subseteq \mathrm{NTimeSpace}(n^2, n) \subset \mathbf{CS}$.

4 Closure Properties

An easy consequence from $\mathcal{C}_0 \subseteq \mathcal{C}'_0 \subseteq \mathcal{C}^\lambda_0 \subset \mathbf{RE}$ and $\mathbf{RE} = \hat{\mathcal{H}}(\mathcal{C}_0)$ is the fact that $\mathcal{C}_0, \mathcal{C}'_0, \mathcal{C}^\lambda_0$ are not closed under arbitrary homomorphisms. Nevertheless, at least the class $\mathcal{C}^\lambda_0$ is closed under non-erasing homomorphisms. This and $\mathbf{CF} \subseteq \mathcal{H}^{cod}(\mathcal{C}_0)$ imply $\mathbf{CF} \subseteq \mathcal{C}^\lambda_0$.

In order to prove that the CFA-languages are closed unter inverse homomorphisms, a lemma about finitely generated monoids was needed. For a commutative monoid M, a subset $S \subseteq M$ is called *quasi-invertible* iff for every $a, b \in M$, $a \in S$ and $a + b \in S$ imply $b \in S$. For example, kernels of homomorphisms are quasi-invertible. Now the lemma states that quasi-invertible submonoids of finitely generated monoids are finitely generated as well. With this lemma, one can prove that $\mathcal{C}_0$ and $\mathcal{C}'_0$ are closed unter inverse homomorphisms.

An overview of the closure properties and the known relations of the language classes is given in figure 1. Thereby, $\cap R$, $L_1 \cdot L_2$, $\cup$, $\cap$, h^{-1}, h, λ-free h stand for intersection with regular languages, concatenation, union, intersection, inverse homomorphism, arbitrary homomorphism and non-erasing homomorphism, respectively.

References

[FKR06] Berndt Farwer, Manfred Kudlek, and Heiko Rölke. *Concurrent turing machines.* Fundamenta Informaticae, 79(3-4):303-317, 2007.

[Hack76] M. Hack. *Petri Net Languages.* Cambridge, Mass.: MIT, Laboratory Computer Science, TR-159, 1976.