

RWTH Aachen University of Technology  
Chair of Computer Science VII  
Seminar on Automata Theory, WS 2009/2010

# The Universal Automaton

---

*Construction and NFA-minimisation*

by *David R. Piegdon*

January 15, 2010

Supervisor: Christof Löding

The author, student of computer science at the *RWTH Aachen University of Technology*, may be reached via email. To obtain this paper or other information, please refer to:

*David Rasmus Piegdon* <[david.rasmus.piegdon@rwth-aachen.de](mailto:david.rasmus.piegdon@rwth-aachen.de)>  
<http://david.piegdon.de/products.html>

This text was written on *Linux* with *vim*, typeset with  $\text{\LaTeX}$  and generated by an implementation of *RFC2795*. Please send fruit to the author. No bananas.

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Preliminaries</b>	<b>3</b>
2.1	Definitions . . . . .	3
2.2	Algorithms . . . . .	4
2.3	Properties . . . . .	5
<b>3</b>	<b>The Universal Automaton</b>	<b>5</b>
3.1	Construction of the Universal Automaton . . . . .	6
3.2	An Example . . . . .	8
<b>4</b>	<b>Minimal NFAs for regular languages</b>	<b>9</b>
4.1	Morphisms into the Universal Automaton . . . . .	9
4.2	NFA-minimisation by morphisms . . . . .	10
4.3	Example (continued) . . . . .	11
	<b>List of Figures</b>	<b>12</b>
	<b>References</b>	<b>12</b>

# 1 Introduction

This paper is a result of the seminar on automata theory at the chair of computer science VII at the RWTH Aachen University of Technology. As such, it is based on [LS08] and does not contain any new research.

The universal automaton of a regular language is canonically defined, thus there is a bijection between universal automata and regular languages. For any finite automaton, there exists a morphism into the universal automaton of its language. As this includes all minimal nondeterministic automata (NFA) for a given language, it is possible to find a minimal NFA for a given regular language by means of its morphism into its universal automaton.

This paper shortly introduces mathematical formalisms to work with automata, then briefly defines the universal automaton for a regular language, and then shows how to construct this automaton. It concludes in showing how to use this automaton to construct a minimal NFA for the same language.

As a part of this seminar, Michael Brysch wrote a paper on the mathematical foundations of and more background on the universal automaton (also based on [LS08]). The universal automaton was subject to a lot of research, mostly concerned about NFA-minimisation and star-height of regular languages. For a general overview on the universal automaton, research on it and further references, the reader is referred to [LS08] and [CC03].

## 2 Preliminaries

### 2.1 Definitions

Let  $S$  be a set. Then  $|S|$  is the diameter (or size) of  $S$ .  $2^S$  denotes the powerset of  $S$ ,  $2^S = \{s \mid s \subseteq S\}$ . This convention is convenient, as it holds that  $|2^S| = 2^{|S|}$ .

In the following, let  $\Sigma$  be an alphabet and  $w$  be a word  $w = a_0a_1\dots a_n$  of length  $n + 1$ ,  $\forall m \in \{0\dots n\} : a_m \in \Sigma$ .  $\epsilon$  denotes the empty word (of length 0). A language  $L$  is a set of words,  $L \subseteq \Sigma^*$ .

A *nondeterministic finite automaton* (NFA) is a 5-tuple  $\mathcal{A} = (Q, \Sigma, \Delta, Q_I, Q_F)$ , where  $Q$  is a set of states,  $\Delta$  is a transition-relation  $\Delta \subseteq Q \times \Sigma \times Q$ ,  $Q_I \subseteq Q$  the set of initial states and  $Q_F \subseteq Q$  the set of final states. A *transition* in the automaton by a label  $a \in \Sigma$  from a state  $q$  to a state  $q_t$  exists, iff  $(q, a, q_t) \in \Delta$  and one writes  $q \vdash_a q_t$ . The *possible successors*  $R = \{q_1, q_2, \dots\}$  of a set of states  $P$  and a label  $a$  can be written as  $R = \{r \mid p \in P \wedge (p, a, r) \in \Delta\} = \Delta(P, a)$ . For a singleton set  $P = \{p\}$  one may also write  $\Delta(p, a)$ . For a word  $w$  we recursively extend  $\Delta$  to be  $\Delta(P, w) = \Delta(\Delta(P, a_0a_1\dots a_{n-1}), a_n)$  the set of states that may be reached from  $P$  via  $w$ .

An NFA is *deterministic* (a DFA), iff  $Q_I$  is singleton and  $\forall q \in Q, \forall a \in \Sigma : \Delta(q, a)$  is singleton. One then writes  $q_i$  for the single state in  $Q_I$ ,  $\delta$  for the deterministic transition relation  $\Delta$  and  $(Q, \Sigma, \delta, q_i, Q_F)$  for the DFA. The successor  $q_t$  for a deterministic transition  $(q, a, q_t) \in \delta$  can then be written as  $\delta(q, a) = q_t$ . Likewise, for a word  $w$ , we extend  $\delta$  recursively to be the single state reachable from

$q$  by  $w$ :  $\delta(q, w) = \delta(\delta(q, a_0 a_1 \dots a_{n-1}), a_n)$ . For a set of states  $P$ , we also define  $\delta(P, a) = \{\delta(p, a) \mid p \in P\}$  and  $\delta(P, w) = \delta(\delta(P, a_0 a_1 \dots a_{n-1}), a_n)$ .

A *run* over a finite word  $w$  is a sequence of states  $q_0 q_1 \dots q_n q_{n+1}$  such that  $q_0 \vdash_{a_0} q_1 \vdash_{a_1} \dots \vdash_{a_n} q_{n+1}$  are valid transitions. A run is *accepting*, iff  $q_0 \in Q_I$  and  $q_{n+1} \in Q_F$ . Otherwise it is *rejecting*. A word  $w$  is accepted by an automaton, iff there exists an accepting run for  $w$ . The *language*  $L_{\mathcal{A}}$  of an automaton is the set of all words accepted by the automaton:  $L_{\mathcal{A}} = \{w \mid \mathcal{A} \text{ accepts } w\}$ .

The *past* of a state  $q$  is the set of words  $P(q) = \{w \mid \exists q_i \in Q_I : q \in \Delta(q_i, w)\}$  (i.e. there exists a run for  $w$  with its first state being an initial state and its last state being  $q$ ). Likewise, the *future* of a state  $q$  is the set of words  $F(q) = \{w \mid \Delta(q, w) \cap Q_F \neq \emptyset\}$  (i.e. there exists a run for  $w$  with its first state being  $q$  and its last state being accepting).

For a word  $v$  we define the *left* (resp. *right*) *quotient* of a language  $L$  as  $v^{-1}L := \{w \mid v \cdot w \in L\}$  (resp.  $Lv^{-1} := \{w \mid w \cdot v \in L\}$ ).  $v^{-1}L$  is also called a *residual language*. For two languages  $X, Y$ , let  $(X, Y)$  be a *subfactorisation* of a language  $L$ , if  $X \cdot Y \subseteq L$ .  $(X, Y)$  is called a *factorisation* if it is maximal for inclusion (i.e.  $\forall X', Y' : X \subseteq X' \wedge Y \subseteq Y' \rightarrow X = X' \wedge Y = Y'$ ). For a factorisation  $(X, Y)$ ,  $X$  is called the *left factor* and  $Y$  the *right factor*.  $\mathcal{F}_L$  is the *set of factorisations* of  $L$ .

For a language  $L$ , two words  $w, v$  are *Nerode equivalent* ( $w \sim_L v$ ) iff  $w^{-1}L = v^{-1}L$ . For two states  $q, r \in Q$  of an automaton  $\mathcal{A}$  we define Nerode equivalence as  $q \sim_{L_{\mathcal{A}}} r :\Leftrightarrow F(q) = F(r)$ .

For a word  $w = a_0 a_1 a_2 \dots a_n$ , the *reverse word* is  $\overleftarrow{w} = a_n \dots a_2 a_1 a_0$ . For a language  $L$ , the *reverse language* is  $\overleftarrow{L} = \{\overleftarrow{w} \mid w \in L\}$ . And for an automaton  $\mathcal{A} = (Q, \Sigma, \Delta, Q_I, Q_F)$ , the *reverse automaton* is  $\overleftarrow{\mathcal{A}} = (Q, \Sigma, \overleftarrow{\Delta}, Q_F, Q_I)$  with  $(q_s, a, q_r) \in \overleftarrow{\Delta}$  iff  $(q_r, a, q_s) \in \Delta$ . I.e. the initial and final states are swapped and all transitions are reversed.

For a language  $L$  we define the *complement language*  $\overline{L} = \{w \mid w \notin L\}$ .

## 2.2 Algorithms

*Trimming of automata.* An automaton is *trimmed* by pruning states that are not reachable from an initial state and merging states that can not reach a final state.

*Determinisation of NFAs.* A nondeterministic finite automaton can be transformed into a language-equivalent deterministic finite automaton. This can be done by *powerset-construction*: For an NFA  $A = (Q, \Sigma, \Delta, Q_I, Q_F)$ , a DFA  $DET(A) = (Q^d, \Sigma, \delta^d, q_I^d, Q_F^d)$  is constructed with:  $Q^d = 2^Q$ , i.e. each state of the determinised automaton represents a set of states in the NFA.  $\delta^d = \{(q, a, q_t) \mid \forall r_t \in q_t : \exists r \in q : (r, a, r_t) \in \Delta \wedge \forall r \in q, r_t \in Q : (r, a, r_t) \in \Delta \rightarrow r_t \in q_t\}$ ,  $q_I^d = Q_I$  and  $Q_F^d = \{q \mid q \in Q^d \wedge q \cap Q_F \neq \emptyset\}$ . Then, the automaton is trimmed. The implication is that there no longer exist two states with the same past. Any states  $q, r$  with  $P(q) = P(r)$  are merged into one state. The resulting DFA can be exponentially larger as the NFA, as it can have up to  $2^{|Q|}$  states. Determinisation is an EXPTIME-hard problem (see e.g. [vGP08]). It can inflate to exponential space and may need exponential time to be solved.

*Co-determinisation of NFAs.*  $CODET(\mathcal{A}) = \overleftarrow{\overleftarrow{\mathcal{A}}}$ . I.e. to codeterminize, we reverse an automaton, determinise it and reverse it again. Likewise, the implication here is that no two states with the same future exist. For all states  $q, r$  with  $F(q) = F(r)$  it holds that  $q = r$ . As a result, there only exists a single final state  $q_F$  in the new automaton with  $F(q_F) = \{\epsilon\}$  and  $P(q_F) = L_{\mathcal{A}}$ . For obvious reasons, co-determinisation is in the same complexity class as determinisation.

*Creating the minimal DFA of an NFA.*  $MIN(\mathcal{A}) = DET(CODET(\mathcal{A}))$ . Though the described method is not very efficient, it is good at demonstrating how the minimisation works: first states are merged if they have the same future; then states are merged if they have the same past. The result is a deterministic automaton with all states  $q, r$  that are Nerode-equivalent  $q \sim_L r$  being merged. As it is required to determinise the automaton, obtaining a minimal DFA for a given NFA may require exponential time and result in an exponentially larger DFA.

*Finding a minimal NFA for a regular language.* This is a hard problem. An algorithm is described in section 4.2.

## 2.3 Properties

It is well known that a language  $L$  is *regular* iff there exists an NFA  $\mathcal{A}$  with  $L_{\mathcal{A}} = L$  and that for all NFAs there exists a DFA and even a (up to state-isomorphism) unique *minimal (canonical) DFA* representing the same language and also that the *class of regular languages* is closed under complement, union and intersection.

The reader will also easily verify that  $\overleftarrow{L_{\mathcal{A}}} = L_{\overleftarrow{\mathcal{A}}}$  and that  $\overleftarrow{\overleftarrow{L}} = L$ . As it also holds that  $L_{\mathcal{A}} = L_{DET(\mathcal{A})}$ , it further holds that  $L_{\mathcal{A}} = L_{CODET(\mathcal{A})}$ .

## 3 The Universal Automaton

In the following, let  $L$  be a regular language over the alphabet  $\Sigma$ .

The universal automaton for  $L$  is, like the minimal DFA, unique up to state-isomorphism. Thus it is called the *canonical* automaton in several papers (e.g. in [CC03]). As the author feels that most people connect the term "canonical" with the minimal DFA, the term "universal" is used throughout this paper. The universal automaton  $\mathcal{U}_L$  for  $L$  is defined as:

$$\mathcal{U}_L = (\mathcal{F}_L, \Sigma, \Delta^U, Q_I^U, Q_F^U)$$

where  $\mathcal{F}_L = \{(X, Y) \mid (X, Y) \text{ is factorisation of } L\}$ , as described in 2.1; thus each state corresponds to a factorisation of  $L$ . Furthermore:

$$\Delta^U = \{((X, Y), a, (X', Y')) \in \mathcal{F}_L \times \Sigma \times \mathcal{F}_L \mid X \cdot a \cdot Y' \subseteq L\}$$

$$Q_I^U = \{(X, Y) \mid \epsilon \in X\} \quad ; \quad Q_F^U = \{(X, Y) \mid \epsilon \in Y\}$$

For more on the basic features and properties of the universal automaton, please refer to [CC03], [LS08] or the paper by Michael Brysch in this seminar.

### 3.1 Construction of the Universal Automaton

To construct  $\mathcal{U}_L$  for a language  $L$ , we need to find all factorisations of  $L$ .

**Lemma 1.** *For  $\mathcal{F}_L$ , there exists a bijection between the right- and the left factors.*

*Proof.* Assume the opposite: there does not exist a bijection, thus there either exists  $(X, Y) \in \mathcal{F}_L$  and  $(X, Y') \in \mathcal{F}_L$  with  $Y \neq Y'$ , or there exists  $(X, Y)$  and  $(X', Y)$  with  $X \neq X'$ . Assume the first. Then there exists a factorisation  $(X, Y \cup Y')$ , thus  $(X, Y)$  and  $(X, Y')$  are no factorisations  $\downarrow$ . Be equal means, the proof holds for  $(X, Y)$  vs.  $(X', Y)$ .  $\square$

Due to Lemma 1, it is enough to only find the right or left factors for a language to begin with.

**Lemma 2.** *a) Any intersection of right quotients is a left factor and any intersection of left quotients is a right factor. b) For every  $(X, Y) \in \mathcal{F}_L$ :*

$$X = \bigcap_{y \in Y} Ly^{-1} \quad ; \quad Y = \bigcap_{x \in X} x^{-1}L$$

*Proof.* For a): Let  $Y = \bigcap_{w \in W} w^{-1}L$  be an intersection of left quotients. Now assume  $Y$  is not a factor,  $Y$  then has to be a non-maximal subfactor, hence  $\exists y \notin Y : \forall w \in W : w \cdot y \in L$ . But if so, then  $\forall w \in W : y \in w^{-1}L$ , thus  $y \in Y$   $\downarrow$ .

For b): Let  $x \in X$  and  $y \in Y$ .  $(X, Y) \in \mathcal{F}_L \Rightarrow \forall x, y : x \cdot y \in x \cdot (x^{-1}L) \Rightarrow \forall x, y : y \in (x^{-1}L) \Rightarrow \forall y : y \in \bigcap_x x^{-1}L \Rightarrow Y \subseteq \bigcap_x x^{-1}L$ . Further:  $w \in \bigcap_x x^{-1}L \Rightarrow \forall x : w \in x^{-1}L$ . Due to the maximality of factors, it holds that  $w \in Y$ , thus  $Y \supseteq \bigcap_x x^{-1}L$ . The same follows for  $X$  via  $\overleftarrow{L}$ .  $\square$

As noted in 2.2, codetermination of an automaton results in an automaton where no states with the same future exist. A property of codetermined automata is, that for a state  $q$  and a label  $a \in \Sigma$ , there exists at most one predecessor-state  $p$  with  $q \in \Delta(p, a)$  (i.e. the incoming transitions are deterministic) and that exactly one final state exists.<sup>1</sup> This is a property that can be used to find the set of factorisations for a language.

Let  $\mathcal{A}_L = (Q, \Sigma, \delta, q_I, Q_F)$  be a DFA that accepts  $L$  and  $\mathcal{C}_L = \text{CODET}(\mathcal{A}_L) = (Q^C, \Sigma, \Delta^C, Q_I^C, \{q_F^C\})$ . As codetermination is based on determinisation,  $Q^C$  is a subset of  $2^Q$ . With this kept in mind, denote by  $\mathcal{I}_{\mathcal{A}}$  the closure of  $Q^C$  under intersection.

---

<sup>1</sup>This immediately follows from the definition of codetermination.

**Theorem 1.**  $\psi_{\mathcal{A}} : \mathcal{I}_{\mathcal{A}} \rightarrow \mathcal{F}_L$  is a bijection:

$$\psi_{\mathcal{A}}(P) = (X, Y) \quad \text{with} \quad Y = \bigcap_{p \in P} F(p)$$

*Proof.* Every intersection of left quotients is a right factor (Lemma 2). With  $p \in P$ ,  $F(p)$  is a left quotient, thus  $Y$  is a right factor. Thus,  $\psi_{\mathcal{A}}$  is well defined. To prove that it is a bijection, we prove that the inverse function is a one-to-one mapping between  $\mathcal{F}_L$  and  $\mathcal{I}_{\mathcal{A}}$ :

$$\chi_{\mathcal{A}} := \psi_{\mathcal{A}}^{-1} : \mathcal{F}_L \rightarrow 2^Q \quad \text{with} \quad \chi_{\mathcal{A}}((X, Y)) = \{ p \in Q \mid Y \subseteq F(p) \}$$

Let  $(X, Y) \in \mathcal{F}_L$ ,  $P = \{ p \in Q \mid Y \subseteq F(p) \}$  and  $R = \{ S \in Q^C \mid P \subseteq S \} \subseteq 2^Q$ .  $q \notin P \Rightarrow \exists y \in Y : y \notin F(q)$ . It follows that  $\forall S \in Q^C : y \in F(S) \rightarrow q \notin S \wedge S \in R$ , as  $y \in \bigcap_{p \in P} F(p)$ . Hence, a state  $p$  is in all states of  $R$  if and only if  $Y \subseteq F(p)$ , thus  $\bigcap_{S \in R} S = \{ p \in Q \mid Y \subseteq F(p) \} = \chi_{\mathcal{A}}((X, Y)) = P$ .  $P = \bigcap_{S \in R} S \in \mathcal{I}_{\mathcal{A}}$ .  $\square$

**Lemma 3.** Let  $S \in \mathcal{I}_{\mathcal{A}}$  and  $\psi_{\mathcal{A}}(S) = (X, Y)$ . It holds that

$$a) \quad S = \bigcup_{x \in X} \delta(q_I, x) \quad ; \quad b) \quad X = \bigcup_{s \in S} P(s)$$

*Proof.* a)  $\bigcup_{x \in X} \delta(q_I, x) = \{ p \mid \exists x \in X : x \in P(p) \} = \{ p \mid \forall y \in Y : y \in F(p) \} = \{ p \mid Y \subseteq F(p) \} = \chi_{\mathcal{A}}(X, Y) = S$ .

b) Consider a state  $s \in S$ . According to  $\chi_{\mathcal{A}}$ , it holds that  $Y \subseteq F(s)$ . Thus, due to factorisations being maximal and  $\mathcal{A}_L$  being deterministic, any word  $v$  in  $P(s)$  has to be in  $X$ , as  $v \cdot F(s)$  is a subset of  $L$ .  $\square$

**Theorem 2.** Construction of  $\mathcal{U}_L$ . Given a DFA  $\mathcal{A}_L = (Q, \Sigma, \delta, q_I, Q_F)$ , the automaton  $\mathcal{V}_L$  is isomorphic to  $\mathcal{U}_L$ , with

$$\begin{aligned} \mathcal{V}_L &= (\mathcal{I}_{\mathcal{A}}, \Sigma, \Delta^V, Q_I^V, Q_F^V) \quad , \quad \Delta^V = \{ (P, a, S) \mid \delta(P, a) \subseteq S \} \\ Q_I^V &= \{ P \in \mathcal{I}_{\mathcal{A}} \mid q_I \in P \} \quad \text{and} \quad Q_F^V = \{ P \in \mathcal{I}_{\mathcal{A}} \mid P \subseteq Q_F \} \end{aligned}$$

*Proof.* We already defined a bijection from  $\mathcal{I}_{\mathcal{A}}$  into  $\mathcal{F}_L$ . What remains to be shown is that the definitions of initial states, final states and transitions match those of the universal automaton:

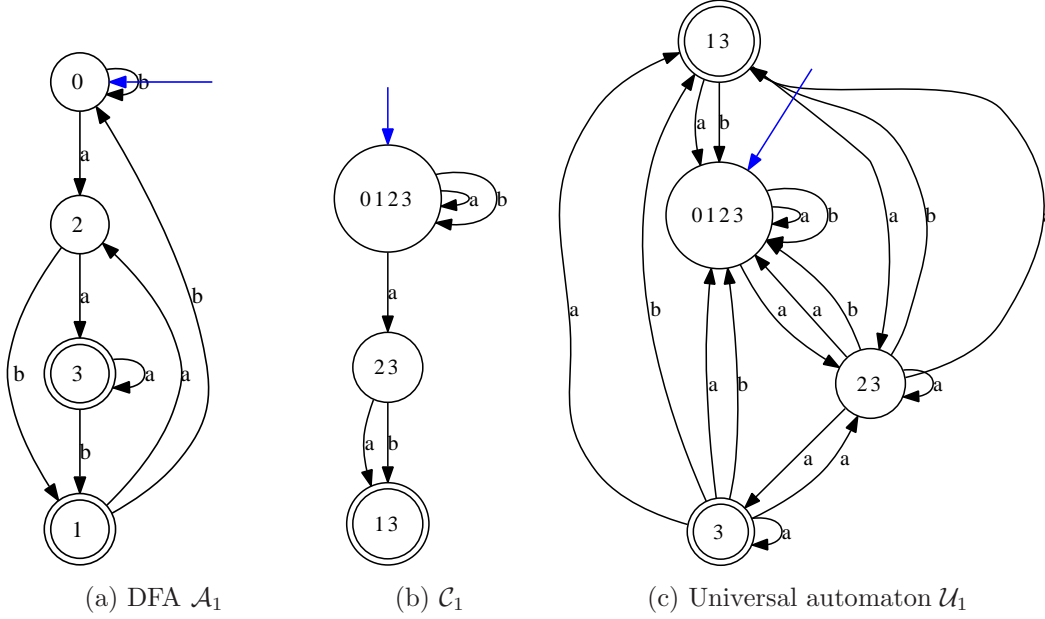
According to Lemma 3, for all  $q_I^V \in Q_I^V$  it holds that  $\psi_{\mathcal{A}}(q_I^V) = (X, Y)$  with  $X = \bigcup_{s \in q_I^V} P(s)$ . Thus for all states  $q \in \mathcal{I}_{\mathcal{A}} : q \in Q_I^V$  iff  $P(q_I) \subseteq P(q)$ ,  $\epsilon \in P(q)$ ,  $\epsilon \in X$ . Thus,  $\psi_{\mathcal{A}}(Q_I^V) = \{ (X, Y) \mid \epsilon \in X \} = Q_I^U$ .

Using the definition of  $\psi_{\mathcal{A}}$ , we obtain: for all  $q_F^V \in Q_F^V : \psi_{\mathcal{A}}(q_F^V) = (X, Y)$  with  $Y = \bigcap_{p \in q_F^V} F(p)$ . As  $q_F^V \subseteq Q_F$ , it holds that  $\psi_{\mathcal{A}}(Q_F^V) = \{ (X, Y) \mid \epsilon \in Y \} = Q_F^U$ .

Transitions: Given two states  $P, S \in Q^V$ , the corresponding factorisations  $(X_P, Y_P)$  and  $(X_S, Y_S)$  are well defined by  $Y_P = \bigcap_{p \in P} F(p)$  and  $Y_S = \bigcap_{s \in S} F(s)$ . It holds that

$$\begin{aligned} \delta(P, a) \subseteq S &\iff Y_S \subseteq \bigcap_{p \in \delta(P, a)} F(p) \\ \iff a \cdot Y_S &\subseteq \bigcap_{p \in P} F(p) = Y_P \iff X_P \cdot a \cdot Y_S \subseteq L \end{aligned}$$

$\square$

Figure 1: Automata for  $L_1$ 

**Theorem 3.** *Given a DFA for a language, the construction of the corresponding universal automaton may need exponential time and result in an automaton of exponential size.*

*Proof.* As the DFA needs to be codetermined by powerset-construction, the required time may be exponential and the resulting automaton may be exponential in size. (Closure under intersection does not increase this.) Furthermore, for each element of  $\Delta^V$ , one has to calculate  $\delta(P, a \in \Sigma)$ , where  $|P| \leq |Q|$ , thus  $O(|Q|)$ . Considering that the universal automaton may have up to  $2^{|Q|}$  states,  $\Delta^V$  may have up to  $2^{|Q|} \cdot |\Sigma| \cdot 2^{|Q|}$  Elements, resulting in  $O(2^{|Q|} \cdot |Q| \cdot |\Sigma|)$  calls to  $\delta$  and  $O(2^{|Q|} \cdot |\Sigma|)$  subset-tests for the calculation of  $\Delta^V$  alone. For the calculation of initial and final states, one has merely to do  $O(2^{|Q|})$  subset tests.  $\square$

### 3.2 An Example

Consider the language  $L_1 = \Sigma^* a \Sigma$  with  $\Sigma = \{a, b\}$ . The minimal DFA  $\mathcal{A}_1 = (Q^A, \Sigma, \delta^A, q_I^A, Q_F^A)$  for this language is given in Figure 1a, the codetermined  $\mathcal{C}_1 = COD(\mathcal{A}_1) = (Q^C, \Sigma, \Delta^C, Q_I^C, \{q_F^C\})$  in Figure 1b.

As can be seen from Figure 1b,  $\mathcal{C}_1$  has the states  $Q^C = \{\{0, 1, 2, 3\}, \{1, 3\}, \{2, 3\}\}$ . Closing under intersection gives

$$\mathcal{I}_1 = \{\{0, 1, 2, 3\}, \{1, 3\}, \{2, 3\}, \{3\}\}$$

Thus the resulting universal automaton  $\mathcal{U}_1 = (Q^U, \Sigma, \Delta^U, Q_I^U, Q_F^U)$  has 4 states, as can be seen in Figure 1c. With the earlier introduced formulas, one can easily calculate:

$$\begin{array}{lll}
Q_I^U = \{\{0, 1, 2, 3\}\} & ; & Q_F^U = \{\{1, 3\}, \{3\}\} \\
\delta^A(\{0, 1, 2, 3\}, a) = \{2, 3\} & & \delta^A(\{0, 1, 2, 3\}, b) = \{0, 1\} \\
\delta^A(\{1, 3\}, a) = \{2, 3\} & & \delta^A(\{1, 3\}, b) = \{0, 1\} \\
\delta^A(\{2, 3\}, a) = \{3\} & & \delta^A(\{2, 3\}, b) = \{1\} \\
\delta^A(\{3\}, a) = \{3\} & & \delta^A(\{3\}, b) = \{1\}
\end{array}$$

From these,  $\Delta^U$  can be calculated as seen in Figure 1c.

$$\begin{array}{l}
\Delta^U = \{ \\
(\{0, 1, 2, 3\}, a, \{0, 1, 2, 3\}) \quad (\{0, 1, 2, 3\}, a, \{2, 3\}), \quad (\{0, 1, 2, 3\}, b, \{0, 1, 2, 3\}), \\
(\{1, 3\}, a, \{0, 1, 2, 3\}), \quad (\{1, 3\}, a, \{2, 3\}), \quad (\{1, 3\}, b, \{0, 1, 2, 3\}) \\
(\{2, 3\}, a, \{0, 1, 2, 3\}), \quad (\{2, 3\}, a, \{1, 3\}), \quad (\{2, 3\}, a, \{2, 3\}), \\
(\{2, 3\}, a, \{3\}), \quad (\{2, 3\}, b, \{0, 1, 2, 3\}), \quad (\{2, 3\}, b, \{1, 3\}), \\
(\{3\}, a, \{0, 1, 2, 3\}), \quad (\{3\}, a, \{1, 3\}), \quad (\{3\}, a, \{2, 3\}), \\
(\{3\}, a, \{3\}), \quad (\{3\}, b, \{0, 1, 2, 3\}), \quad (\{3\}, b, \{1, 3\}), \\
\}
\end{array}$$

## 4 Minimal NFAs for regular languages

As there exists a mapping from any automaton accepting a language into the universal automaton of this language, as we will see, one can use the universal automaton to find a smallest NFA. The following section will introduce the notion of this mapping and show, how it helps in finding a smallest NFA.

Finding a minimal NFA for a given language is a PSPACE-complete problem [JR93]. A PSPACE algorithm to find a minimal NFA for a given language is: enumerate all NFAs by increasing size, test them for language equality to the given language. Notice that constructing the universal automaton is EXPTIME (see Theorem 3). As  $\text{PSPACE} \subseteq \text{EXPTIME}$ , using the universal automaton to find a minimal NFA may not be the most efficient way to find a minimal NFA. However, experience has shown that the construction of the universal automaton can be done rather efficiently most of the time. With the increased performance in the enumeration-process, this method can then compete with other minimisation methods.

### 4.1 Morphisms into the Universal Automaton

**Definition 1.** *Given two automata  $\mathcal{A}$ ,  $\mathcal{B}$  and a function  $\varphi : Q^A \rightarrow Q^B$  with  $\mathcal{A} = (Q^A, \Sigma, \Delta^A, Q_I^A, Q_F^A)$  and  $\mathcal{B} = (Q^B, \Sigma, \Delta^B, Q_I^B, Q_F^B)$ , then  $\varphi$  is a morphism (between automata), iff:*

$$\varphi(Q_I^A) \subseteq Q_I^B \quad \wedge \quad \varphi(Q_F^A) \subseteq Q_F^B \quad \wedge \quad \forall (q, a, q_t) \in \Delta^A : (\varphi(q), a, \varphi(q_t)) \in \Delta^B$$

**Theorem 4.** *For any automaton accepting a language  $L$  there exists a morphism into  $\mathcal{U}_L$ .<sup>2</sup>*

<sup>2</sup>This is a basic property of the universal automaton from which it derives its name. See [LS08], Theorem 2.16, where even even a stronger variant of this is shown.

**Definition 2.** Given a language  $L$ , universal automaton  $\mathcal{U}_L = (Q^U, \Sigma, \Delta^U, Q_I^U, Q_F^U)$  and any automaton  $\mathcal{A} = (Q, \Sigma, \Delta, Q_I, Q_F)$  with  $L = L_{\mathcal{A}}$ , the left canonical morphism  $\varphi : Q \rightarrow Q^U$  is:

$$\varphi(q) = (X_q, Y_q) \quad \text{with} \quad Y_q = \{v \in \Sigma^* \mid P(q) \cdot v \subseteq L\} = \bigcap_{w \in P(q)} w^{-1}L$$

If the universal automaton is known, this map can be computed in polynomial time. For an example of the morphism for  $L_1$  from section 3.2, see Figure 2a.

## 4.2 NFA-minimisation by morphisms

**Lemma 4.** For any minimal automaton the left canonical morphism  $\varphi$  is injective.

*Proof.* Assume the opposite:  $\varphi$  is not injective, thus there exist two different states  $q, r$  with  $\varphi(q) = \varphi(r)$ . From the definition of  $\varphi$  follows that  $F(q) = F(r)$ , thus  $\bigcap_{u \in P(q)} u^{-1}L = \bigcap_{v \in P(r)} v^{-1}L$ , hence  $q$  and  $r$  can be merged into a single state.  $\downarrow$  The considered automaton is not minimal.  $\square$

**Definition 3.** A subautomaton of an automaton  $\mathcal{A}$  is a copy of  $\mathcal{A}$  with a subset of its states and a subset of its transitions removed. Transitions from and to the removed states are removed as well.

**Theorem 5.** Any minimal automaton  $\mathcal{M}_L$  of a language  $L$  is a subautomaton of  $\mathcal{U}_L$

*Proof.* Obviously,  $\mathcal{M}_L$  is smaller or equal than  $\mathcal{U}_L$ . Thus the left canonical morphism  $\varphi$  is injective. Due to the nature of morphisms, given  $\varphi$  and  $\mathcal{U}_L$ , one can obtain  $\mathcal{M}_L$  by removing all states in  $\mathcal{U}_L$  that are not hit by  $\varphi$ . Removal of superfluous transitions may be required, too, to meet the exact  $\mathcal{M}_L$ .  $\square$

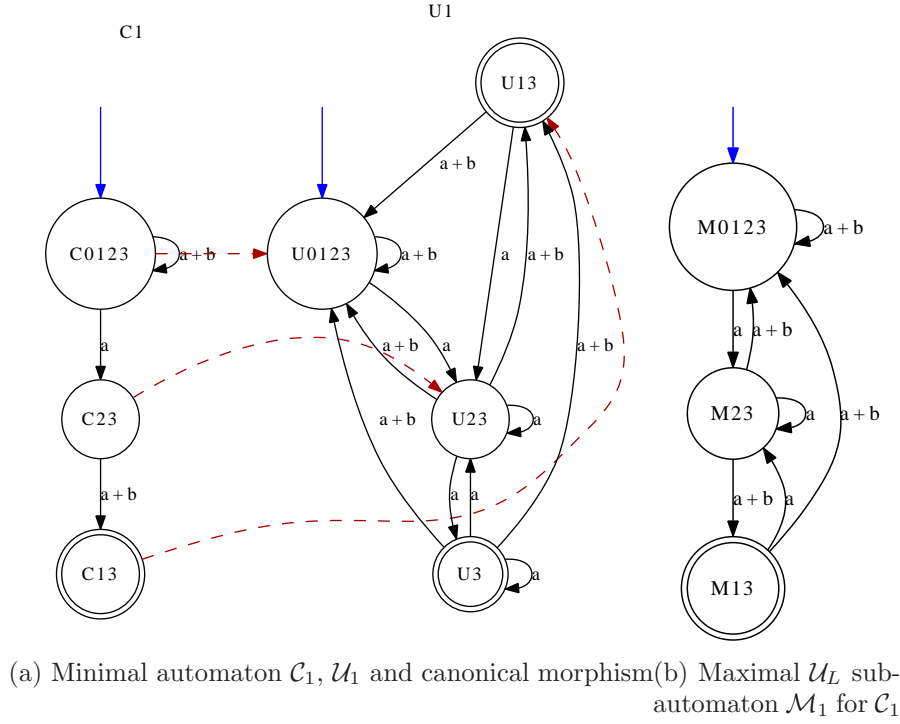
From Theorem 5 follows, that finding a minimal automaton for a language  $L$  is as simple as testing all subautomata of  $\mathcal{U}_L$  (in size-increasing order) for language equality to  $\mathcal{U}_L$ . As for a  $\mathcal{U}_L$  with  $n$  states, that would be up to  $2^n$  equivalence tests ; this is still a hard problem. However, there have been several advances in heuristics to exclude wrong subautomata in a fast way [KW70] [Pol05]. In the following, a condition is presented that was inspired by [MP95].

**Theorem 6.** Let  $\mathcal{A} = (Q, \Sigma, \delta, q_I, Q_F)$  be a DFA and  $\mathcal{U}_L$  be the correspondingly constructed universal automaton. A state set  $Q^M \subseteq \mathcal{I}_L$ <sup>3</sup> imposes a subautomaton  $\mathcal{M}_L = (Q^M, \Sigma, \Delta^M, Q_I^M, Q_F^M)$  on  $\mathcal{U}_L$  accepting the same language, if:

- a)  $\bigcup_{F \in Q^M, F \subseteq Q_F} F = Q_F$
- b)  $\forall a \in \Sigma, \forall q \in Q, \forall P \in Q^M :$   
 $[\delta(q, a) = p \wedge p \in P] \rightarrow [\exists S \in Q^M : q \in S \wedge \delta(S, a) \subseteq P]$

---

<sup>3</sup> $\mathcal{I}_L$ , the set of states in the universal automaton, is a subset of  $2^Q$ . It follows that  $Q^M \subseteq 2^Q$ .

Figure 2: Morphism for  $\mathcal{C}_1$  and according subautomaton

*Proof.* Let  $w = a_0a_1\dots a_n \in L_{\mathcal{A}}$  be a word. The run of  $w$  in  $\mathcal{A}$  is  $q_I q_1 \dots q_n q_F$ <sup>4</sup>.

According to a), there exists a final state  $F \in Q_F^M$  with  $q_F \in F$ . Starting from this state and moving backwards over the word, b) gives: if there exists a state  $P_i \in Q^M$  with  $q_i \in P_i$ , from  $\delta(q_{i-1}, a_{i-1}) = q_i$  follows that there also exists a state  $P_{i-1} \in Q^M$  with  $p_{i-1} \in P_{i-1}$  and  $(P_{i-1}, a_{i-1}, P_i) \in \Delta^M$ . Once that by induction we pass the first character  $a_0$  in the word, in  $\mathcal{A}$  the current state would be  $q_I$ . In  $\mathcal{M}_L$ , the corresponding state  $P_0$  is initial as well, as  $q_I \in P_0$ . Thus  $\mathcal{M}_L$  accepts  $w$ .  $\square$

### 4.3 Example (continued)

Recall the example from section 3.2:  $L_1 = \Sigma^* a \Sigma$  with  $\Sigma = \{a, b\}$ . The reader will have noticed that a smallest possible automaton for  $L_1$  is, by chance,  $\mathcal{C}_1$ . The left canonical morphism from  $\mathcal{C}_1$  is shown in Figure 2a, the maximal subautomaton  $\mathcal{M}_1$  of  $\mathcal{U}_1$  under this morphism is shown in 2b.

$\mathcal{M}_1$  is language equivalent to  $\mathcal{U}_1$  and  $\mathcal{C}_1$ , even though  $\mathcal{C}_1$  has less transitions. The reader will easily notice that the new transitions are superfluous in that they do not change the language; they are just a saturation of the subautomaton with all edges not changing the language. Verifying the conditions of Theorem 6 for  $\mathcal{M}_1$  is left as an exercise to the reader.

<sup>4</sup> $q_I \vdash_{a_0} q_1 \vdash_{a_1} \dots q_n \vdash_{a_n} q_F$  with  $q_I$  initial state and  $q_F$  final state.

## List of Figures

1	Automata for $L_1$ . . . . .	8
2	Morphism for $\mathcal{C}_1$ and according subautomaton . . . . .	11

## References

- [CC03] Jean-Marc Champarnaud and Fabien Coulon. Theoretical study and implementation of the canonical automaton. *Fundamenta Informaticae*, 55(1), 2003.
- [JR93] Tao Jiang and B. Ravikumar. Minimal NFA problems are hard. *SIAM Journal on Computing*, 22(6):1117–1141, 1993.
- [KW70] T. Kameda and P. Weiner. On the state minimization of nondeterministic finite automata. *Computers, IEEE Transactions on*, C-19(7):617–627, July 1970.
- [LS08] Sylvain Lombardy and Jacques Sakarovitch. The universal automaton. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata*, pages 457–504. Amsterdam University Press, 2008.
- [MP95] Oliver Matz and Andreas Potthoff. Computing small nondeterministic finite automata. In *Proc. Workshop on Tools and Algorithms for the Construction and Analysis of Systems, Dept. of CS, Univ. of Aarhus*, pages 74–88. SpringerVerlag, 1995. BRICS Note Series.
- [Pol05] Libor Polák. Minimalizations of NFA using the universal automaton. In Michael Domaratzki, Alexander Okhotin, Kai Salomaa, and Sheng Yu, editors, *Implementation and Application of Automata*, volume 3317 of *LNCS*, pages 325–326, 2005.
- [vGP08] Rob van Glabbeek and Bas Ploeger. Five determinisation algorithms. In Oscar H. Ibarra and Bala Ravikumar, editors, *Implementation and Applications of Automata*, volume 5148 of *LNCS*, pages 161–170. Springer, 2008.