Technische Universität München **Winter term 2009/10**
I7
Prof. J. Esparza / J. Křetínský / M. Luttenberger **19.10.2009**

Solution

Automata and Formal Languages – Homework 1

Due 29.10.2009.

Exercise 1.1

In the lecture, it was shown how to obtain from a finite automaton a regular expression representing the same language by iteratively eliminating states of the automaton. In this exercise we will see that eliminating states corresponds to eliminating variables from a linear system of equations over the algebraic structure given by the set of languages 2^{Σ^*} with set union, language concatenation and the Kleene star as operations.

(a) You might remember Arden's Lemma from the introductionary course on formal language theory. Arden's Lemma says:

Given two languages $A, B \subseteq \Sigma^*$ with $\varepsilon \notin A$ then there is a unique language $X \subseteq \Sigma^*$ satisfying $X = AX \cup B$ and this language is given by A^*B.

Prove Arden's Lemma.

(b) Solve the following system given in two variables X, Y over $\Sigma = \{a, b, c, d, e, f\}$.

$$\begin{aligned} X &= \{a\}X \cup \{b\}Y \cup \{c\} \\ Y &= \{d\}X \cup \{e\}Y \cup \{f\}. \end{aligned}$$

Hint: Consider X as a constant language and solve the equation for Y using Arden's Lemma.

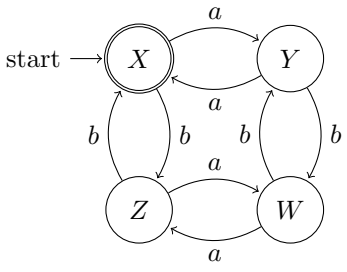
We can associate with any finite automaton $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$ a linear equation system as follows:

We take as variables the states of the automaton. For every state X we then have the equation

$$X = \left(\bigcup_{Y \in Q} \bigcup_{a \in \Sigma} \{a\}\delta(X, a, Y)Y \right) \cup \{\varepsilon\}\chi_F(X),$$

with $\delta(X, a, Y) = \{\varepsilon\}$ if $Y \in \delta(X, a)$, $\delta(X, a, Y) = \emptyset$ otherwise; and $\chi_F(X) = \{\varepsilon\}$ if $X \in F$, $\chi_F(X) = \emptyset$ otherwise.

(c) Consider the automaton depicted on the left and the equation system we obtain from it (shown on the right):



$$\begin{aligned} X &= \{a\}Y \cup \{b\}Z \cup \{\varepsilon\} \\ Y &= \{a\}X \cup \{b\}W \\ Z &= \{b\}X \cup \{a\}W \\ W &= \{b\}Y \cup \{a\}Z \end{aligned}$$

Calculate the solution of this linear system by iteratively eliminating variables. Start with Y, then eliminate Z, finally W.

Compare the solution you obtain to the regular expression obtained in the script for this automaton.

Solution:

(a) One easily checks that A^*B is a solution:

$$A^*B \stackrel{\text{by def.}}{=} (\bigcup_{k\geq 0} A^k)B = \bigcup_{k\geq 0} A^kB = B \cup \bigcup_{k\geq 1} A^kB = B \cup A\bigcup_{k\geq 0} A^kB = B \cup A(A^*B).$$

Assume there is another language L satisfying $L = AL \cup B$.

If $L = \emptyset$, then $B = \emptyset$ and subsequently $A^*B = \emptyset$ and $L = A^*B$ follow.

So, assume $L \neq \emptyset$. We may repeatedly substitute $AL \cup B$ for L in this equation, yielding:

$$\begin{array}{lcll} L & = & AL \cup B & \\ L & = & A(AL \cup B) \cup B & = B \cup AB \cup A^2L \\ L & = & A(A(AL \cup B) \cup B) \cup B & = B \cup AB \cup A^2B \cup A^3L \\ & \vdots & & \end{array}$$

Using induction we obtain thus for all $k \geq 0$:

$$L = A^{k+1}L \cup \bigcup_{l=0}^{k} A^lB. \qquad (*)$$

From this it follows that $A^kB \subseteq L$ for all $k \geq 0$, and so $A^*B \subseteq L$.

Choose now any word $w \in L$ and denote by $n := |w|$ its length. By $(*)$ we then have

$$w \in L \Leftrightarrow w \in A^{n+1}L \cup \bigcup_{l=0}^{n} A^lB.$$

As we require that $\varepsilon \notin A$ (and $L \neq \emptyset$ by assumption), any word of $A^{n+1}L$ has to have length at least $n+1$, so $w \in \bigcup_{l=0}^{n} A^lB$ and, thus, $w \in A^*B$. We conclude $L \subseteq A^*B$ and $L = A^*B$.

(b) The solution of the equation $Y = \{d\}X \cup \{e\}Y \cup \{f\} = \{e\}Y \cup (\{d\}X \cup \{f\})$ is given by $\{e\}^*(\{d\}X \cup \{f\})$ by Arden's Lemma – independent of the value of X. Substituting this into the first equation, we obtain

$$X = \{a\}X \cup \{b\}\{e\}^*(\{d\}X \cup \{f\}) \cup \{c\} = (\{a\} \cup \{b\}\{e\}^*\{d\})X \cup (\{b\}\{e\}^*\{f\} \cup \{c\}) = \mathcal{L}(a + be^*d)X \cup \mathcal{L}(be^*f \cup c).$$

yielding $X = \mathcal{L}((a + be^*d)^*(be^*f + c))$ and $Y = \mathcal{L}(e^*(d(a + be^*d)^*(be^*f + c) + f))$.

(c) In order to eleminate Y, we simply substitute the equation $Y = \{a\}X \cup \{b\}W$ into the remaining equations, yielding:

$$\begin{array}{lcl} X & = & \{aa\}X \cup \{ab\}W \cup \{b\}Z \cup \{\varepsilon\} \\ Z & = & \{b\}X \cup \{a\}W \\ W & = & \{a\}Z \cup \{ba\}X \cup \{bb\}W \end{array}$$

Similarly, we may eliminate Z:

$$\begin{array}{lcl} X & = & \{aa\}X \cup \{ab\}W \cup \{bb\}X \cup \{ba\}W \cup \{\varepsilon\} = \{aa, bb\}X \cup \{ab, ba\}W \cup \{\varepsilon\} \\ W & = & \{ab\}X \cup \{aa\}W \cup \{ba\}X \cup \{bb\}W = \{aa, bb\}W \cup \{ab, ba\}X \end{array}$$

The parametrized solution for W then is $\mathcal{L}((aa + bb)^*(ab + ba))X$. So, we obtain the single equation

$$X = \mathcal{L}((aa + bb))X \cup \mathcal{L}((ab + ba)(aa + bb)^*(ab + ba))X \cup \{\varepsilon\}$$

whose least solution is $X = \mathcal{L}(((aa + bb) + (ab + ba)(aa + bb)^*(ab + ba))^*)$, the same regular expression as given in the script. The Elemination of states in the procedure described in the lecture therefore can be seen as the elimination of the corresponding variables in the underlying linear equation system.

Exercise 1.2

Let Σ be an alphabet. We define the operator $||: \Sigma^* \times \Sigma^* \to 2^{\Sigma^*}$ as follows:

$$v||u := u||v, \quad u||\varepsilon := \{u\}, \quad au||bv := \{aw \mid w \in u||bv\} \cup \{bw \mid w \in au||v\} \text{ for } a, b \in \Sigma, u, v \in \Sigma^*.$$

Examples:

$$b||d = \{bd, db\}, \quad ab||d = \{abd, adb, dab\}, \quad ab||cd = \{cabd, acbd, abcd, cadb, acdb, cdab\}.$$

- Show that for two regular languages $L_1, L_2 \subseteq \Sigma^*$ their interleaving

$$L_1||L_2 := \bigcup_{u \in L_1, v \in L_2} u||v$$

is also regular.

Solution: Let $\mathcal{A}_i = (Q_i, \Sigma, \delta_i, q_0^{(i)}, F_i)$ be a DFA with $L_i = \mathcal{L}(\mathcal{A}_i)$ (for $i = 1, 2$). We make use of the product automata construction, i.e., we construct an automaton with states $Q_1 \times Q_2$. While in the constructions seen in class both automaton move when a symbol is read, we now choose nondeterministically one of the two automata which is to move accoringly to the symbol read, while the other one does not change its state, i.e.,

$$\delta((q, q'), a) := \{(\delta_1(q, a), q'), (q, \delta_2(q', a))\}.$$

It is left to the reader to show that this automaton indeed accepts exactly $L_1 || L_2$.

Exercise 1.3

Let Σ_1, Σ_2 be two alphabets. A map $h : \Sigma_1^* \to \Sigma_2^*$ is called a *homomorphism* if it respects the empty word and concatenation, i.e.,

$$h(\varepsilon) = \varepsilon \text{ and } h(w_1 w_2) = h(w_1)h(w_2) \text{ for all } w_1, w_2 \in \Sigma_1^*.$$

Assume that $h : \Sigma_1^* \to \Sigma_2^*$ is a homorphism. Note that h is completely determined by its values on Σ_1.

(a) Let $\mathcal{A}$ be a finite automaton over the alphabet Σ_1. Describe how to constuct a finite automaton accepting the language

$$h(\mathcal{L}(\mathcal{A})) := \{h(w) \mid w \in \mathcal{L}(\mathcal{A})\}.$$

(b) Let $\mathcal{A}'$ be a finite automaton over the alphabet Σ_2. Describe how to construct a finite automaton accepting the language

$$h^{-1}(\mathcal{L}(\mathcal{A}')) := \{w \in \Sigma_1^* \mid h(w) \in \mathcal{L}(\mathcal{A}')\}.$$

(c) Recall that the language $\{0^n 1^n \mid n \in \mathbb{N}\}$ is context free, but not regular. Use the preceding two results to show that $\{(01^k 2)^n 3^n \mid k, n \in \mathbb{N}\}$ is also not regular.

Solution:

(a) Let $\mathcal{A} = (Q, \Sigma_1, \delta, q_0, F)$ be a DFA. In the lecture, you have seen finite automata whose transitions are labeled by regular expressions, and not only by letters. We make use of this extension here. We construct from $\mathcal{A}$ a finite automoton $\mathcal{A}' = (Q, \Sigma_2, \delta', q_0, F)$ whose transitions are labeled by words over Σ_2, more precisely by the words $h(\Sigma_1) := \{h(a) \mid a \in \Sigma_1\}$. Note that this set is finite as Σ_1 is finite.

We then set for all $a \in \Sigma_1$

$$\delta'(q, h(a)) := \delta(q, a).$$

Otherwise δ' is defined to be the empty set.

This basically means that we apply h to the edge labels of the graph underlying $\mathcal{A}$, i.e., if $q \xrightarrow{a} q'$ in $\mathcal{A}$, then $q \xrightarrow{h(a)} q'$ in $\mathcal{A}'$.

We now show that $\mathcal{L}(\mathcal{A}') = h(\mathcal{L}(\mathcal{A})$.

- Consider some word $w = a_1 a_2 \dots a_n \in \mathcal{L}(\mathcal{A})$. Hence, there is an accepting run of $\mathcal{A}$ on w, i.e.,

  $$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \dots \xrightarrow{a_n} q_n \text{ with } q_n \in F.$$

  By definition of δ' we therefore have $q_i \xrightarrow{h(a_i)} q_{i+1}$ in $\mathcal{A}'$ for all transitions along this run, implying that $w' = h(w)$ is accepted by $\mathcal{A}'$. Hence, $h(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A}')$
- Assume thus that $w' \in \mathcal{L}(\mathcal{A}')$. Then there is some accepting run of $\mathcal{A}'$

  $$q_0 \xrightarrow{u_1} q_1 \xrightarrow{u_2} q_2 \dots \xrightarrow{u_l} q_n \text{ with } q_n \in F \text{ and } u_i \in h(\Sigma_1).$$

  By definition of δ' we find for every transition $q_i \xrightarrow{u_i} q_{i+1}$ of $\mathcal{A}'$ some $a_i \in \Sigma_1$ with $h(a_i) = u_i$ such that $q_i \xrightarrow{a_i} q_{i+1}$ in $\mathcal{A}$. By construction,

  $$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \dots \xrightarrow{a_l} q_n \text{ with } q_n \in F$$

  is a run of $\mathcal{A}$, in particular, it is an accepting run. So, $a_1 a_2 \dots a_l \in \mathcal{L}(\mathcal{A})$ and $h(a_1 a_2 \dots a_l) = w'$. Therefore, $\mathcal{L}(\mathcal{A}') \subseteq h(\mathcal{L}(\mathcal{A}))$.

(b) Now we are given a finite automaton $\mathcal{A}' = (Q, \Sigma_2, \delta', q_0, F)$ over the alphabet Σ_2', w.l.o.g. $\mathcal{A}'$ is deterministic, and we need to construct a finite automaton $\mathcal{A}$ accepting $h^{-1}(\mathcal{L}(\mathcal{A}'))$.

As $\mathcal{A}'$ is assumed to be deterministic, δ' can be thought of as a map from $Q \times \Sigma_2$ to Q and we may extend this map to $Q \times \Sigma_2^*$ in the natural way:

$$\delta'(q, \varepsilon) := q \text{ and } \delta'(q, a_1 a_2 \dots a_n) := \delta'(\dots \delta'(\delta'(q, a_1), a_2) \dots, a_n).$$

The idea now is that a transition of $\mathcal{A}$ labeled by $a \in \Sigma_1$ summarizes the behavior of $\mathcal{A}'$ when reading the word $h(a)$.

Hence set

$$\delta(q, a) := \delta'(q, h(a)) \text{ for all } a \in \Sigma_1.$$

We claim that $\mathcal{A} = (Q, \Sigma_1, \delta, q_0, F)$ then accepts exactly $h^{-1}(\mathcal{L}(\mathcal{A}'))$.

- $\mathcal{L}(\mathcal{A}) \subseteq h^{-1}(\mathcal{L}(\mathcal{A}'))$:

  Choose some $w = a_1 a_2 \dots a_n \in \mathcal{L}(\mathcal{A})$, i.e.,

  $$F \ni \delta(q_0, w) = \delta(\dots, \delta(\delta(q_0, a_1), a_2) \dots, a_n) \overset{\text{by Induction}}{=} \delta'(\dots, \delta'(\delta'(q_0, h(a_1)), h(a_2)) \dots, h(a_n)) = \delta'(q_0, h(w)).$$

  So, $h(w) \in \mathcal{L}(\mathcal{A}')$, i.e., $w \in h^{-1}(\mathcal{L}(\mathcal{A}'))$.

- $\mathcal{L}(\mathcal{A}) \supseteq h^{-1}(\mathcal{L}(\mathcal{A}'))$:

  Let $w = a_1 a_2 \dots a_n \in h^{-1}(\mathcal{L}(\mathcal{A}'))$, i.e., $h(w) \in \mathcal{L}(\mathcal{A}')$, i.e.,

  $$F \ni \delta'(q_0, h(w)) = \delta'(\dots, \delta'(\delta'(q_0, h(a_1)), h(a_2)) \dots, h(a_n)) \overset{\text{by Induction}}{=} \delta(\dots, \delta(\delta(q_0, a_1), a_2) \dots, a_n) = \delta(q_0, w).$$

  So, $w \in \mathcal{L}(\mathcal{A})$.

(c) Set $L := \{(01^k2)^n 3^n \mid k, n \geq 0\}$.

Let $h,\ \{0,1,2,3\}^* \to \{0,1\}^*$ be the homomorphism uniquely determined by

$$h(0) = 0, h(1) = \varepsilon, h(2) = \varepsilon, h(3) = 1.$$

Then $h(L) = \{0^n 1^n \mid n \geq 0\}$.

So, if L was regular, i.e., if there was some finite automaton $\mathcal{A}$ with $L = \mathcal{L}(\mathcal{A})$, then by the preceding results there would also be a finite automaton $\mathcal{A}'$ with $\mathcal{L}(\mathcal{A}') = \{0^n 1^n \mid n \geq 0\}$. Contradiction.

Exercise 1.4

For L_1, L_2 regular languages over an alphabet Σ, the *left quotient* of L_1 by L_2 is defined by

$$L_2 \backslash L_1 := \{v \in \Sigma^* \mid \exists u \in L_2 :\ uv \in L_1\}$$

(a) Use the fact that regular languages are closed under homomorphisms, inverse homomorphisms, concatenation and intersection to prove they are closed under quotienting.

(b) Given finite automata $\mathcal{A}_1, \mathcal{A}_2$, construct an automaton $\mathcal{A}$ such that

$$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_2) \backslash \mathcal{L}(\mathcal{A}_1)$$

(c) Is there any difference when taking the *right quotient* $L_1 / L_2 := \{u \in \Sigma^* \mid \exists v \in L_2 :\ uv \in L_1\}$?

Solution:

(a) Let L_1 and L_2 be regular languages over Σ. Let us denote a barred copy of the alphabet Σ by $\overline{\Sigma} = \{\overline{a} \mid a \in \Sigma\}$ (assuming that Σ and $\overline{\Sigma}$ are disjoint). We define a homomorphism $h : \Sigma \cup \overline{\Sigma} \to \Sigma$ as follows:

$$h(a) = a \quad \text{for every } a \in \Sigma$$
$$h(\overline{a}) = a \quad \text{for every } a \in \Sigma$$

Thus $h^{-1}(L_1)$ consists of words from L_1 with all possible combinations of letters being barred or not. (E.g. $h^{-1}(\{ab\}) = \{ab, a\overline{b}, \overline{a}b, \overline{a}\overline{b}\}$.)

We now intersect $h^{-1}(L_1)$ with a regular language $L_2.\overline{\Sigma}^*$ in order to get all words from L_1 with prefix from L_2 but with the remaining suffix being barred.

We can now apply homomorphism $\overline{h}$ defined by

$$\overline{h}(a) = \varepsilon \quad \text{for every } a \in \Sigma$$
$$\overline{h}(\overline{a}) = a \quad \text{for every } a \in \Sigma$$

in order to obtain the suffixes only, now being unbarred. Hence,

$$L_2 \backslash L_1 = \overline{h}(h^{-1}(L_1) \cap L_2.\overline{\Sigma}^*)$$

proves the regularity of the quotient.

(b) In order to accept a word $v \in L_2 \backslash L_1$, we need to guess a word $u \in L_2$ and check whether $uv \in L_1$. Therefore, we can build a parallel composition of automata accepting L_1 and L_2 using the product construction and replace all transitions by ε-transitions (we are guessing the prefix that actually is not there) and adding ε-transitions from all states corresponding to final states for L_2 to the respective state of the automaton for L_1.

Formally, let $\mathcal{A}_i = (Q_i, \Sigma, \delta_i, q_i, F_i)$ be such that $\mathcal{L}(\mathcal{A}_i) = L_i$ for $i \in \{1, 2\}$. We construct

$$\mathcal{A} = ((Q_1 \times Q_2) \cup Q_1, \Sigma, \delta, (q_1, q_2), F_1)$$

so that $\mathcal{L}(\mathcal{A}) = L_2 \backslash L_1$. We set the transition relation δ as follows:

$(p, r) \xrightarrow{\varepsilon} (p', r')$	for every $a \in \Sigma$ with $p \xrightarrow{a}_1 p'$ and $q \xrightarrow{a}_2 q'$	(guessing the prefix)
$(p, r) \xrightarrow{\varepsilon} p$	for every $r \in F_2$	(prefix is in L_2)
$p \xrightarrow{a} p'$	for every $p \xrightarrow{a}_1 p'$	(checking the suffix)

where $q \xrightarrow{a}_i q'$ denotes $\delta_i(q, a) \ni q'$.

(c) Similarly as in (a), we have

$$L_1 / L_2 = \overline{h}(h^{-1}(L_1) \cap \overline{\Sigma}^*.L_2)$$

The direct construction of an automaton recognizing the right quotient is not as straightforward as in the case with left quotient: we need to check the intersection of L_2 with the language recognized by the automaton $\mathcal{A}_1$ with any initial state. An easier approach is to make use of the *reverse* construction together with the construction above, since

$$L_1 / L_2 = (L_2^R \backslash L_1^R)^R$$

Exercise 1.5

Let L_1, L_2 be regular languages. Determine the inclusion relation between the following languages:

- L_1
- $(L_1 / L_2).L_2$
- $(L_1.L_2) / L_2$

Solution: None of the inclusions holds in general. Let

$$\begin{aligned} L_1 &= \{a, b\} \\ L_2 &= \{b, bb\} \end{aligned}$$

Then quotienting removes all words from L_1 not having a suffix in L_2 and appending L_2 may add new suffixes as follows:

$$\begin{aligned} L_1 / L_2 &= \{\varepsilon\} \\ (L_1 / L_2).L_2 &= \{b, bb\} \\ L_1.L_2 &= \{ab, abb, bb, bbb\} \\ (L_1.L_2) / L_2 &= \{a, ab, \varepsilon, b, bb\} \end{aligned}$$

which disproves all inclusions except for $(L_1 / L_2).L_2 \subseteq (L_1.L_2)/L_2$ and $L_1 \subseteq (L_1.L_2) / L_2$. To disprove the former, let $L_1 = \{a, b\}, L_2 = \{b, ab\}$, then $(L_1 / L_2).L_2 = \{b, ab\} \not\subseteq \{\varepsilon, a, b, aa, ba\} = (L_1.L_2)/L_2$. To disprove the latter, let $L_1 = \{a\}, L_2 = \emptyset$, then $(L_1.L_2) / L_2 = \emptyset / \emptyset = \emptyset \not\supseteq \{a\}$.

We can at least prove the last inclusion holds for $L_1 = \emptyset$ or $L_2 \neq \emptyset$. The former case is trivial, for the latter let $v \in L_2$. If $u \in L_1$ then $uv \in L_1 L_2$ and thus $u \in (L_1.L_2) / L_2$.