COMP3630 – Theory Of Computation:

Assignment 1

Australian National University

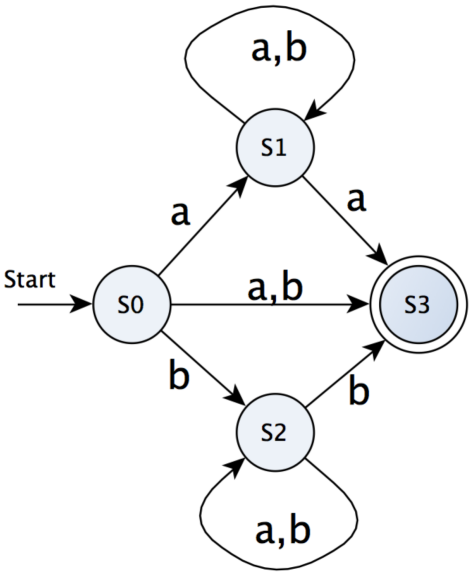
Lecturers: JinBo Huang and Dirk Pattison

Handed out: **08 March 2013**. Due: **22 March 2013**.

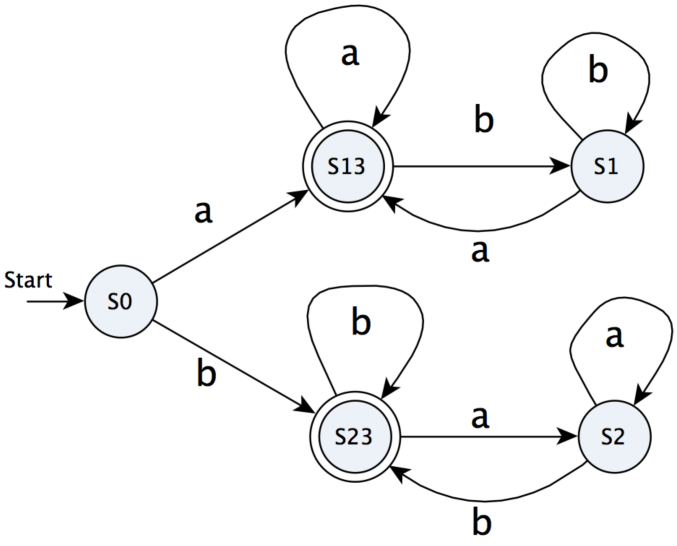
Uid:U5223173 Name: Jimmy Lin E-mail: linxin@gmail.com

Exercise 1 DFAs and NFAs

1. NFA



2. DFA



Exercise 2 Higman's Lemma

Proof:

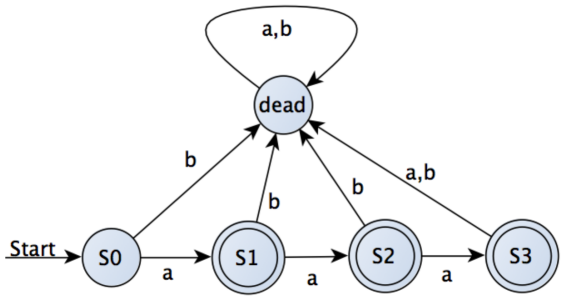
Based on the condition that L has alphabet with only letter (let's just specify it as letter **a** here, and no ε transition), we **only need to care about the string in language L with longest length**. This is because S(L) consists of all strings that come from the deletion of strings in L and **the S(L) is only dependent on the longest string in the language L** given the condition that only one letter is allowed.

Here, we assume what we care about – number n as **the longest length of strings in L.** And for any language with one letter in its alphabet Σ, there must exist a string whose length is largest among all strings in this language. It could be 1, 2 or even infinite. And the form of the language S(L) should be

$$S(L) = \{a, aa, aaa, \dots, a^n\}$$

where **n is fixed** in each case as the longest length of strings in L. Then let's bifurcate our discussion according to the finiteness of n.

If n is finite, it is simple to prove that S(L) is regular language. Because we can easily work out a corresponding DFA to describe the S(L). Let's take n = 3 as example, the DFA corresponds to S(L) is as follows.



What if n is infinite, e.g $L = \{w | w = a^p a^p, p \geq 0\}$? In this case, S(L) can be expressed as one regular expression. That is,

$$S(L) = L(a^*)$$

Therefore, S(L) is a regular language when n is infinite.

In summary, S(L) is always a regular language whatever n is finite or not, let alone the regularity of language L.

Exercise 3 Relationships Between Regular Languages

$$L_1 = L((ab + ac^*)^*) \qquad L_2 = L(a(b + c^*)^*)$$

1. $L_1 = L_2$

Proof by contradiction.

Assume that $L_1 = L_2$, which means

$$For\ \forall\ w \in L_1, w \in L_2\ \ and\ \ For\ \forall\ w \in L_2, w \in L_1$$

$$\Leftrightarrow\ \ \neg\ \exists\ w \in L_1, w \notin L_2\ \ and\ \ \neg\ \exists\ w \in L_2, w \notin L_1 \qquad \text{---(0)}$$

Obviously, $w = a(b + c)a(b + c) \in L_1$, $but\ \notin L_2$, which means

$$\exists\ w \in L_1, w \notin L_2$$

That contradicts to the previous statement (0). Therefore, we should negate the initial assumption, and get

$$L_1 \neq L_2.$$

2. $L_1 \cap L_2 = \varnothing$

Proof by contradiction.

Assume that $L_1 \cap L_2 = \emptyset$, which means

$$For\ \forall\ w \in L_1, w \notin L_2$$

$$\Leftrightarrow\ \ \neg\ \exists\ w \in L_1, w \in L_2 \qquad \text{--- (1)}$$

However, based on the given regular expression, we can easily find that

$$When\ w = ab + ac, w \in L_1\ and\ w \in L_2$$

that is,

$$\exists\ w \in L_1, w \in L_2$$

which contradicted to our previous statement (1). Therefore, negate what we assume in the very beginning.

$$L_1 \cap L_2 \neq \emptyset$$

3. $L_1 \subseteq L_2$

Proof by contradiction

Assume that $L_1 \subseteq L_2$,which means

$$For\ \forall w \in L_1, w \in L_2$$

$$\Leftrightarrow\ \neg\ \exists\, w \in L_1, w \notin L_2 \qquad \text{---(2)}$$

However, based on given regular expression, we have

$$w = a(b+c)a(b+c) \in L_1\ , but\ \notin L_2$$

that is,

$$\exists\, w \in L_1, w \notin L_2$$

which contradicts to our previous statement (2), therefore we negate the assumption at the very beginning.

$$L_1 \nsubseteq L_2$$

4. $L_2 \subseteq L_1$.

Proof by contradiction

Assume that $L_2 \subseteq L_1$,which means

$$For\ \forall w \in L_2, w \in L_1$$

$$\Leftrightarrow\ \neg\ \exists\, w \in L_2, w \notin L_1 \qquad \text{---(3)}$$

However, based on given regular expression, we have

$$w = a(b+c)(b+c) \in L_2\ , but\ \notin L_2$$

That is,

$$\exists\, w \in L_1, w \notin L_2$$

Which contradicts to our previous statement (3), therefore we negate the assumption at the very beginning.

$$L_2 \nsubseteq L_1$$

Exercise 4 Use of Pumping Lemma (to be modified)

1. The set of strings of 0's and 1's that are of the form ww, that is, some string repeated.

   Proof by contradiction.

   We assume $L = \{ww | w = (0+1)^*\}$ is regular language.

   Let's consider $w = 0^l 10^l 1$.

   According to Pumping Lemma,

   $$\exists l, w' = xyz, |xyz| \geq l$$

   Such that the following three conditions are satisfied

   $$(1)|xy| \leq l \quad (2)\, y \neq \varepsilon \quad (3) for\ \forall\, k \geq 0, xy^k z \in L$$

   As $|xy| \leq l$, y must be 0s. For this reason, the string $w'' = xyyz$ would have more 0s in its first part than its second part. And, generally

   $$xy^k z \notin L\ (k \neq 1)$$

   Which contradicts to our statement (3), so we negate the assumption

   $$L = \{ww | w = (0+1)^*\}\ is\ not\ regular\ language$$

2. The set of strings of 0's and 1's that are of the form ww^R, that is, some string followed by its reverse.

   Proof by contradiction.

   We assume $L = \{ww^R | w = (0+1)^*\}$ is regular language.

   Let's consider $w = 0^l 110^l$.

   According to Pumping Lemma,

   $$\exists l, w' = xyz, |xyz| \geq l$$

Such that the following three conditions are satisfied

$$(1)|xy| \leq l \quad (2)\, y \neq \varepsilon \quad (3) for\ \forall\, k \geq 0, xy^k z \in L$$

As $|xy| \leq l$, y must be 0s. For this reason, the string $w'' = xyyz$ would have more 0s in its first part than its second part. And, generally

$$xy^k z \notin L\ (k \neq 1)$$

Which contradicts to our statement (3), so we negate the assumption

$$L = \{ww^R | w = (0+1)^*\}\ is\ not\ regular\ language$$

3. The set of strings of 0's and 1's of the form $w\overline{w}$, where $\overline{w}$ is formed from w by replacing all 0's by 1's, and vice versa; e.g., 011 = 100, and 011100 is an example of a string in the language.

   Proof by contradiction.

   We assume $L = \{w\overline{w} | w = (0+1)^*\}$ is regular language.

   Let's consider $w = 0^l 11^l 0$.

   According to Pumping Lemma,

$$\exists l, w' = xyz, |xyz| \geq l$$

   Such that the following three conditions are satisfied

$$(1)|xy| \leq l \quad (2)\, y \neq \varepsilon \quad (3) for\ \forall\, k \geq 0, xy^k z \in L$$

   As $|xy| \leq l$, y must be 0s. For this reason, the string $w'' = xyyz$ would have more 0s in its first part than its second part. And, generally

$$xy^k z \notin L\ (k \neq 1)$$

   Which contradicts to our statement (3), so we negate the assumption

$$L = \{w\overline{w} | w = (0+1)^*\}\ is\ not\ regular\ language$$

4. The set of strings of the form w1^n, where w is a string of 0's and 1's of length n.

Proof by contradiction.

We assume $L = \{w1^n | w = (0+1)^n\}$ is regular language.

Let's consider $w = (0+1)^l 1^l$.

According to Pumping Lemma,

$$\exists n, w' = xyz, |xyz| \geq l$$

Such that the following three conditions are satisfied

$$(1)|xy| \leq l \quad (2)\, y \neq \varepsilon \quad (3) for\ \forall\, k \geq 0, xy^k z \in L$$

As $|xy| \leq l$, y must be a mixture of 0s and 1s. For this reason, the string $w'' = xyyz$ would have strings of large legth in its first part than the 1s in the second part. And, generally

$$xy^k z \notin L\ (k \neq 1)$$

Which contradicts to our statement (3), so we negate the assumption

$$L = \{w1^n | w = (0+1)^n\}\ is\ not\ regular\ language$$

Exercise 5 Closure Properties

Show that the regular languages are closed under the following operations.

1. $min(L) = \{w | w \in L, but\ no\ proper\ prefix\ of\ w \in L\}$

Automaton-based approach. Let's start with a DFA A for L.

To get the automaton A* that simulates min(L), we can **replace all transition arcs that start from accepting states (including loop arcs of accepting states) in A with arcs from the same accepting state but to dead state**. (This would keep A as a DFA, rather than NFA because of incomplete transitions). The new DFA is corresponding to min(L) since **the automaton would not accept any symbol for halting once it reached the accepting state**.

Hence, we can say that min(L) is expressible by the DFA A*, and for this reason, any regular language is closed under min operations.

2. $max(L) = \{w | w \in L, wx \notin L\ for\ any\ nonempty\ x\}$

Automaton-based approach. Let's start with a DFA A for L.

To get the automaton A* that simulates max(L), we can **change all accepting states that have at least one transition arc pointing to non-dead states to be non-accepting states**. The resulting DFA is corresponding to max(L) since **all remained accepting state only has transition to dead state, and it would be impossible for the new DFA to accept longer string**.

Hence, we can say that max(L) is expressible by the DFA A*, and for this reason, any regular language is closed under max operations.

3. $init(L) = \{w | wx \in L\ for\ some\ x\}$

Again, we prove the closure property of maximization operation informally by automaton-based approach. Let's start with a DFA A for L.

To get the automaton A* that simulates init(L), we can **make start state, accepting state and all immediate states (dead states excluded) in A to be accepting states.** The A* corresponds to init(L) in that **A* accepts all strings in the paths of moving towards accepting states for A**.

Hence, we can say that init (L) is expressible by the DFA A*, and for this reason, any regular language is closed under init operations.

Exercise 6 DFA Minimization

1. Draw the table of distinguishabilities for this automaton.

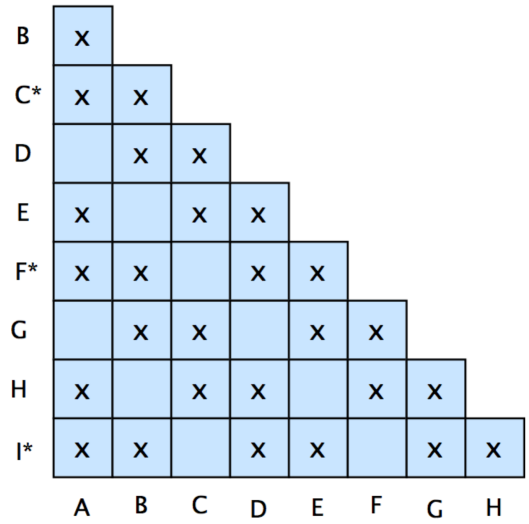


Table of distinguishabilities

2. Construct the minimum-state equivalent DFA.

Based on the work being done in question 1, we establish state blocks by

$$block\ S_0\ contains\ states\ A, D, G$$

$$block\ S_1\ contains\ states\ B, H, E$$

$$block\ S_2\ contains\ states\ C, F, I$$

The minimum-state equivalent DFA is as follows (presented graphically)