6.840 **Dan Ports**

Theory of Computation drkp@mit.edu

2004/09/22

Problem 1-1

10

Let DROP-OUT(A) be the language consisting of all the strings that can be created by removing one symbol from a string in A. We show that the class of regular languages is closed under this operation.

Let L be a regular language and M be the DFA that recognizes L. We show how to construct a NFA that recognizes DROP-OUT(L). To do so, we construct a new machine M' that contains two "copies" of M (we refer to them M_1 and M_2, though they are actually parts of a larger machine rather than separate machines). The start state of M' will be the start state of M_1, and the accept states will be the accept states of M_2. M_1 and M_2 will each contain internal edges: replicas of the edges in M. In addition, we add additional edges crossing from M_1 to M_2: suppose a is a state in M and a_1 and a_2 are the corresponding states in M_1 and M_2 respectively, and similarly b, b_1, and b_2. Then if there is a transition from a to b in M, we add an ϵ-transition from a_1 to b_2. Thus M_1 corresponds to the states before the removed symbol, M_2 corresponds to the states after the removed symbol, and jumping between them across the ϵ-transition corresponds to skipping the removed symbol. A picture follows.

(over)

More formally, let $M = (Q, \Sigma, \delta, q, F)$, and define $M' = (Q', \Sigma', \delta', q', F')$.

$$Q' = \{r_1, r_2 : r \in Q\}$$
$$\Sigma' = \Sigma$$
$$q' = q_1$$
$$F' = \{f_2 : f \in F\}$$

The transition function δ' contains three classes of transitions:

1. from a_1 to b_1 with symbol σ if $\delta(a, \sigma) = b$
2. from a_2 to b_2 with symbol σ if $\delta(a, \sigma) = b$
3. from a_1 to b_2 with the empty string if there exists a σ such that $\delta(a, \sigma) = b$.

6.840
Theory of Computation
2004/09/22

(10)

Dan Ports
drkp@mit.edu

Problem 1-2

a) Let N be a NFA with n states, with $L(N) \neq \emptyset$. Then there is some string w in $L(N)$. If w has length less than n, then we are done, so assume it has length $|w| \geq n$. Observe that the fact that w is accepted by N indicates that there is a path P of at least $|w|$ transitions from the start state of N to an accept state (it may have more than $|w|$ transitions because of ϵ-transitions). So it passes through $|w| + 1$ states, including the start and end states. But there are only n states in the machine, so by the pigeonhole principle, there must be a cycle in the graph. This cycle can be removed (repeatedly if necessary) to produce a path from the start state to the accept state in $n - 1$ transitions. This corresponds to a string of length at most $n - 1$ that is accepted and therefore in $L(N)$.

✓3

b) Consider the NFA N that consists of only one state, which is both the start state and an accept state, and no transitions. This NFA accepts the empty string, since the start state is an accept state, but does not accept any non-empty strings since there are no transitions from the state.

✓3

c) Let N be an NFA with n states, and define $f(n) = 2^n$. We show that if $\overline{L(N)} \neq \emptyset$ then $w \in \overline{L(N)}$ for some w with length less than $f(n)$.[1] Assume $\overline{L(N)} \neq \emptyset$, i.e. that there is a string x that is not accepted by N. We show that a string w can be produced with length less than 2^n.

Convert N to the equivalent DFA, M. M has at most 2^n states. x is not accepted by N, so it is not accepted by M either, so there is a path of transitions in M that leads to a reject state. As in part a, we remove cycles from this path of transitions in order to create a string of length less than 2^n (the number of states in M) that leads to the same reject state.

✓4

[1] This bound may not be tight, but it's an upper bound, and it looks like a "reasonable function" to me.

10/10

6.840

Theory of Computation

2004/09/22

Dan Ports

drkp@mit.edu

Problem 1-3

Consider the language $L = \left\{wxw \ : \ w, x \in \{0,1\}^+\right\}$. We show by contradiction that L is not regular. Assume the contrary. Then by the pumping lemma, there exists a pumping length p. Consider the string $a = 0^p100^p1$. Note that $a \in L$. By the pumping lemma, $a = xyz$, $\forall i \geq 0, xy^iz \in L$, $|y| > 0$. and $|xy| \leq p$. By the last condition, y is located in the initial p zeros. Therefore, a can be pumped up to achieve the string $a' = 0^{p+k}100^p1$ for any $k > 0$, which will be in L. Arbitrarily, take $k = 2$. So $a' = xwx$ for some non-empty x and w. Since a' ends with a 1, and the last character in a must be the last character in x, x must end with a 1. Therefore, x must be the string $0^{p+2}1$, since this is the first prefix of a' that ends with a 1. But then the remaining part of the string following this prefix is $0^{p+1}1$, and this must be equal to yx, which is a contradiction. So L is not regular.

6.840
Theory of Computation
2004/09/22

Dan Ports
drkp@mit.edu

10

Problem 1-4

Let $E = \{\mathtt{a}^i\mathtt{b}^j \ : \ i \neq j, 2i \neq j\}$. We show that E is a context-free language.
Note that E can be expressed as the union of three languages:

1. $\{\mathtt{a}^i\mathtt{b}^j \ : \ i < \frac{j}{2}\}$
2. $\{\mathtt{a}^i\mathtt{b}^j \ : \ \frac{j}{2} < i < j\}$
3. $\{\mathtt{a}^i\mathtt{b}^j \ : \ i > j\}$

The first and last languages are clearly context-free; the second is more subtle. Note that if $i = x$ and $j = y$ satisfies the inequality $\frac{j}{2} < i < j$, then $i = x+1$ and $j = y+2$ also satisfies the inequality, and likewise $i = x+2$ and $j = y+2$. These recurrences, combined with the base cases $i = 2, j = 3$ and $i = 3, j = 4$ completely characterize the set of integers that satisfies this inequality. The proof is by induction.

So E can be expressed as a context-free grammar:

- $A \leftarrow \epsilon \mid \mathtt{a}A$
- $B \leftarrow \epsilon \mid \mathtt{b}B$
- $C \leftarrow \epsilon \mid \mathtt{a}C\mathtt{b}$
- $D \leftarrow \epsilon \mid \mathtt{a}D\mathtt{bb}$
- $F \leftarrow \mathtt{aabbb} \mid \mathtt{aaabbbb} \mid \mathtt{a}F\mathtt{bb} \mid \mathtt{aa}F\mathtt{bb}$
- $E \leftarrow DB\mathtt{b} \mid F \mid \mathtt{a}AC$

F corresponds to the recurrence explained above; the three terms of E correspond to the three component languages enumerated above.

6.840
Theory of Computation
2004/09/22

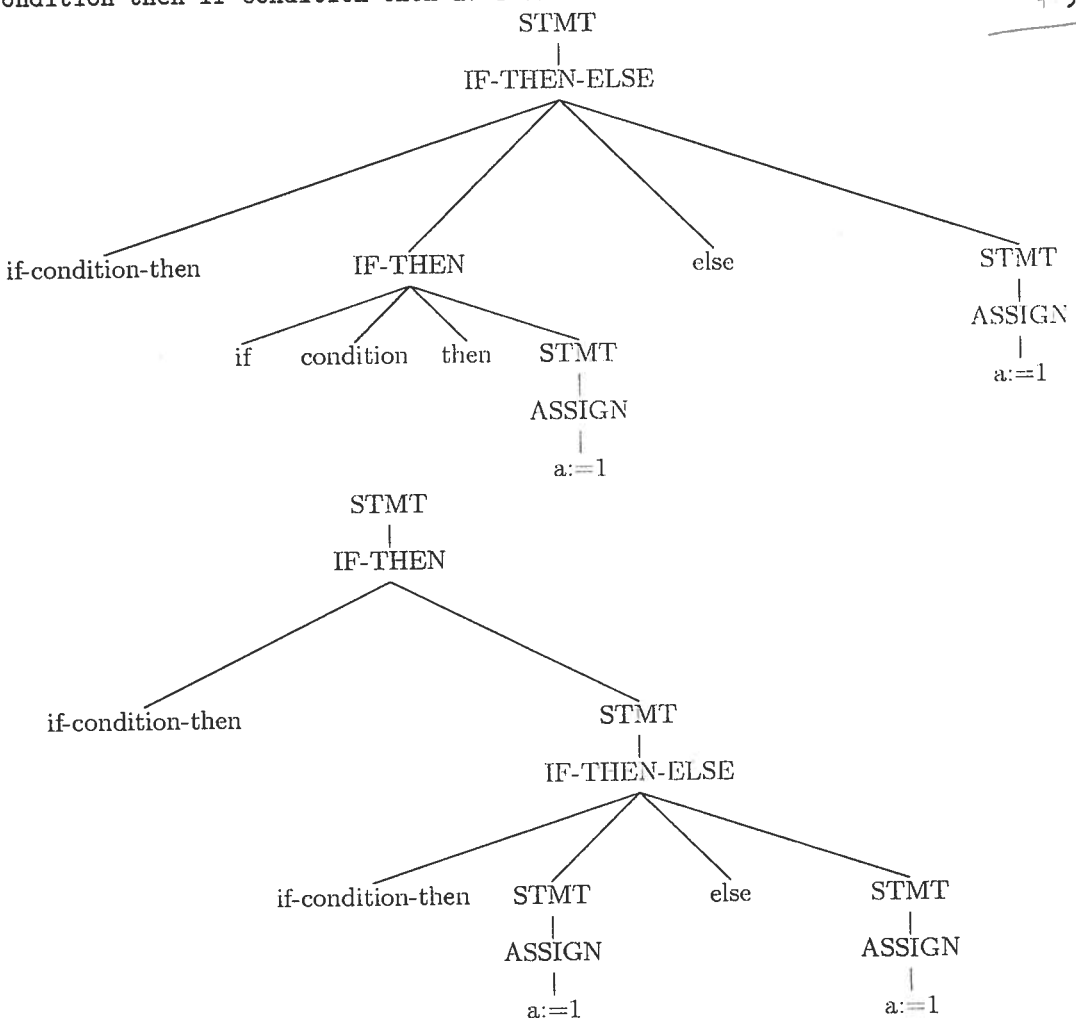
Dan Ports
drkp@mit.edu

10

Problem 1-5

a) G is ambiguous. The following string has two possible parse trees:
`if condition then if condition then a:=1 else a:=1`

+5



b) The following grammar is unambiguous and defines the same language:

- STMT → A | B
- A → ASSIGN | `if condition then` A `else` A
- B → `if condition then` STMT | `if condition then` A `else` B
- ASSIGN → `a:=1`

A represents subexpressions in which every `if` statement has an `else` clause, while B represents those where at least one `else` is missing.

can't generate... +3

if cond then ≠ A

if cond then if cond then a=1 else a=1

else a=1