Recitation 8 - Review of Homework 3 and Turing Machine Example

John Chilton

June 8, 2007



- ► Homework 3 Questions
- ▶ Turing Machine Example

Take notes, these slides won't be posted.

Problem 1. Provide CFGs for:

$$L_1 = \{a^m b^n c^p d^q | m + n = p + q\}$$

$$L_2 = \{x_1 \# x_2 \# \dots \# x_k \mid x_i \in \{a, b\}^* \text{ and for some } i \text{ and } j, x_i = x_j^R\}$$

Problem 1 Problem 3 w/o shorthand Problem 9a Problem 10

$$a^m c^p d^q$$

with m = p + q.

$$a^m c^p d^q$$

with
$$m = p + q$$
.

$$a^m c^p d^q = a^q a^p c^p d^q$$

$$a^m c^p d^q$$

with
$$m = p + q$$
.

$$a^m c^p d^q = a^q a^p c^p d^q$$

This can be seen as $a^q T d^q$ where T is variable that generates an equal number of as and cs.

$$a^m c^p d^q$$

with m = p + q.

$$a^m c^p d^q = a^q a^p c^p d^q$$

This can be seen as $a^q T d^q$ where T is variable that generates an equal number of as and cs.

$$S \rightarrow aSd \mid T$$

 $T \rightarrow aTc \mid \epsilon$

Problem 1
Problem 3 w/o shorthand
Problem 9a
Problem 10

- - ▶ a^q Td^q

- $ightharpoonup m > q : a^q a^{m-q} b^n c^p d^q.$
 - ▶ a^q Td^q
 - ▶ Generated all the *d*s, T needs equal number of *a*s plus *b*s as *c*s.

- ightharpoonup m > q: $a^q a^{m-q} b^n c^p d^q$.
 - ▶ a^q Td^q
 - ▶ Generated all the ds, T needs equal number of as plus bs as cs.
- $p q > m : a^m b^n c^p d^{q-m} d^m.$

- $ightharpoonup m > q : a^q a^{m-q} b^n c^p d^q.$
 - ▶ a^q Td^q
 - Generated all the ds, T needs equal number of as plus bs as cs.
- - ▶ a^m Ud^m

- - ▶ a^q Td^q
 - Generated all the ds, T needs equal number of as plus bs as cs.
- - ▶ a^m Ud^m
 - Generated all the as, U needs equal number of cs plus ds as bs.

$$S \rightarrow aSd \mid T \mid U$$
 $T \rightarrow aTc \mid V$
 $U \rightarrow bUd \mid V$
 $V \rightarrow bVc \mid \epsilon$

$$\{x_1 \# x_2 \# \dots \# x_k \mid x_i \in \{a, b\}^* \text{ and for some } i \text{ and } j, x_i = x_j^R\}$$

Big Idea: Assume i <= j. x_i doesn't depend on anything before it and x_j doesn't depend on anything after it. Each part is independent, so break it up:

$$\{x_1 \# x_2 \# \dots \# x_k \mid x_i \in \{a, b\}^* \text{ and for some } i \text{ and } j, x_i = x_j^R\}$$

Big Idea: Assume i <= j. x_i doesn't depend on anything before it and x_j doesn't depend on anything after it. Each part is independent, so break it up:

$$S \rightarrow ETB$$

$$\{x_1 \# x_2 \# \dots \# x_k \mid x_i \in \{a, b\}^* \text{ and for some } i \text{ and } j, x_i = x_j^R\}$$

Big Idea: Assume i <= j. x_i doesn't depend on anything before it and x_j doesn't depend on anything after it. Each part is independent, so break it up:

$$S \rightarrow ETB$$

E will generate everything before x_i , B will generate everything after x_j , and T will generate everything from x_i to x_j .

$$X \rightarrow 0X \mid 1X \mid \epsilon$$

We can use this rule to generate individual x_k that don't matter. Shouldn't use it to generate x_i or x_i though.

Problem 1 Problem 3 w/o shorthand Problem 9a Problem 10

$$S \rightarrow ETB$$

Problem 1 Problem 3 w/o shorthand Problem 9a Problem 10

$$S \rightarrow ETB$$

Consider E.

▶ x_i could be x_1 so $E \rightarrow \epsilon$ must be a rule.

$S \rightarrow ETB$

- \triangleright x_i could be x_1 so $E \rightarrow \epsilon$ must be a rule.
- ▶ If E isn't ϵ then it better be a sequence of x_k s separated by #s.

$S \rightarrow ETB$

- \triangleright x_i could be x_1 so $E \rightarrow \epsilon$ must be a rule.
- ▶ If E isn't ϵ then it better be a sequence of x_k s separated by #s.
 - \triangleright $E \rightarrow \#XE \mid \epsilon$
 - \triangleright $E \rightarrow X \# E \mid \epsilon$
 - ightharpoonup E
 ightharpoonup $\#X\#E \mid \epsilon$

$$S \rightarrow ETB$$

- \triangleright x_i could be x_1 so $E \rightarrow \epsilon$ must be a rule.
- ▶ If *E* isn't ϵ then it better be a sequence of x_k s separated by #s.
 - $ightharpoonup E
 ightarrow X\#E \mid \epsilon$

Where we are at...

$$S \rightarrow ETB$$
 $X \rightarrow 0X \mid 1X \mid \epsilon$
 $E \rightarrow X\#E \mid \epsilon$
 $B \rightarrow \#XB \mid \epsilon$
 $T \rightarrow generate everything from x_i to $x_j$$

Since x_j is the reverse of x_i , whenever we a symbol to the front of x_i we need to add the same symbol to the end of x_j . So T will include at least

$$T \rightarrow 0T0 \mid 1T1$$

Two cases for how to end x_i and x_j . i = j and $i \neq j$.

Since x_j is the reverse of x_i , whenever we a symbol to the front of x_i we need to add the same symbol to the end of x_j . So T will include at least

$$T \rightarrow 0T0 \mid 1T1$$

Two cases for how to end x_i and x_j . i = j and $i \neq j$.

▶ If i = j, then the base case is the middle symbol, which could be 0, 1, or ϵ if $|x_i|$ is even. Add rules $T \to 0 \mid 1 \mid \epsilon$.

Since x_j is the reverse of x_i , whenever we a symbol to the front of x_i we need to add the same symbol to the end of x_j . So T will include at least

$$T \rightarrow 0T0 \mid 1T1$$

Two cases for how to end x_i and x_j . i = j and $i \neq j$.

- ▶ If i = j, then the base case is the middle symbol, which could be 0, 1, or ϵ if $|x_i|$ is even. Add rules $T \to 0 \mid 1 \mid \epsilon$.
- ▶ If $i \neq j$. There are a couple ways to think about this.

Problem 1
Problem 3 w/o shorthand
Problem 9a
Problem 10

$$T \rightarrow 1T1 \mid 0T0 \mid 1 \mid 0 \mid \epsilon \mid \dots$$

$$T \rightarrow 1T1 \mid 0T0 \mid 1 \mid 0 \mid \epsilon \mid \dots$$

▶ If j = i + 1: x_i and x_j are separated by #, so add the rule $T \to \#$.

$$T \rightarrow 1T1 \mid 0T0 \mid 1 \mid 0 \mid \epsilon \mid \dots$$

- ▶ If j = i + 1: x_i and x_j are separated by #, so add the rule $T \to \#$.
- ▶ Else: # comes after x_i and before x_j and between there can be as many x_k s separated by # as there need to be

$$T \rightarrow 1T1 \mid 0T0 \mid 1 \mid 0 \mid \epsilon \mid \dots$$

- ▶ If j = i + 1: x_i and x_j are separated by #, so add the rule $T \to \#$.
- ▶ Else: # comes after x_i and before x_j and between there can be as many x_k s separated by # as there need to be
 - T → #U#

$$T \rightarrow 1T1 \mid 0T0 \mid 1 \mid 0 \mid \epsilon \mid \dots$$

- ▶ If j = i + 1: x_i and x_j are separated by #, so add the rule $T \to \#$.
- ▶ Else: # comes after x_i and before x_j and between there can be as many x_k s separated by # as there need to be
 - ► *T* → #*U*#
 - $U \to X \# U \mid X$

Easier way to end T when $x_i \neq x_j$. A pound separates x_i and x_j . If more than this separates them its a pound followed by a series of x_k s ending in another #. So add the rule

$$T \rightarrow \#E$$

So one answer could be:

$$S \rightarrow ETB$$

 $X \rightarrow 0X \mid 1X \mid \epsilon$
 $E \rightarrow X \# E \mid \epsilon$
 $B \rightarrow \# XB \mid \epsilon$
 $T \rightarrow 1T1 \mid 0T0 \mid 1 \mid 0 \mid \epsilon \mid \# E$

Another less structured approach:

$$\begin{split} S &\to T \mid J\#T \mid T\#J \mid J\#T\#J \\ J &\to 0J \mid 1J \mid \#J \mid \epsilon \\ T &\to 1T1 \mid 0T0 \mid 1 \mid 0 \mid \epsilon \mid \# \mid \#J\# \end{split}$$

Converting a CFG to a PDA, the last few steps. Consider

$$S \rightarrow aSd \mid T$$

 $T \rightarrow aTc \mid \epsilon$

Convert to PDA.

Problem 9 part a.

$$A = \{a^q \mid q \text{ is prime}\}$$

Problem 9 part a.

$$A = \{a^q \mid q \text{ is prime}\}$$

- ▶ aaa ∈ A
- aaaa ∉ A

$$|vxy| \le p \Rightarrow b + c + d \le p$$

$$|vxy| \le p \Rightarrow b + c + d \le p$$

$$|vy| > 0 \Rightarrow b + d > 0$$

$$|vxy| \le p \Rightarrow b+c+d \le p$$

$$|vy| > 0 \Rightarrow b + d > 0$$

$$|s| > p \Rightarrow a + c + e > 0$$

$$|vxy| \le p \Rightarrow b+c+d \le p$$

$$|vy| > 0 \Rightarrow b + d > 0$$

$$|s| > p \Rightarrow a + c + e > 0$$

▶
$$uv^m xy^m z \in A \Rightarrow a + c + e + m * (b + d)$$
 is prime for any $m \ge 0$

$$|vxy| \le p \Rightarrow b+c+d \le p$$

$$|vy| > 0 \Rightarrow b + d > 0$$

$$|s| > p \Rightarrow a + c + e > 0$$

▶
$$uv^m xy^m z \in A \Rightarrow a + c + e + m * (b + d)$$
 is prime for any $m \ge 0$

▶ Call
$$a' = a + c + e$$
, and note $a' > 0$.

$$|vxy| \le p \Rightarrow b + c + d \le p$$

$$|vy| > 0 \Rightarrow b + d > 0$$

$$|s| > p \Rightarrow a + c + e > 0$$

▶
$$uv^m xy^m z \in A \Rightarrow a + c + e + m * (b + d)$$
 is prime for any $m \ge 0$

▶ Call
$$a' = a + c + e$$
, and note $a' > 0$.

▶ Call
$$b' = b + d$$
, and note $b' > 0$.

$$|vxy| \le p \Rightarrow b + c + d \le p$$

$$|vy| > 0 \Rightarrow b + d > 0$$

$$|s| > p \Rightarrow a + c + e > 0$$

▶
$$uv^m xy^m z \in A \Rightarrow a + c + e + m * (b + d)$$
 is prime for any $m \ge 0$

▶ Call
$$a' = a + c + e$$
, and note $a' > 0$.

▶ Call
$$b' = b + d$$
, and note $b' > 0$.

$$uv^{a'+b'+1}xy^{a'+b'+1}z = 1^{a'+b'(a'+b'+1)} = 1^{a'(1+b')+b'(1+b')} = 1^{(a'+1)(b'+1)}$$
. Contradiction!



Problem 1 Problem 3 w/o shorthand Problem 9a Problem 10

Problem 10. Let G be a CFG in Chomsky normal form that contains b variables. Show that, if G generates some string with a derivation having at least 2^b steps, L(G) contains an infinite number of strings.

Problem 1 Problem 3 w/o shorthan Problem 9a Problem 10

Big Idea: We can show that a grammar in Chompsky normal form with b variables can generate an infinite number of strings if it has some derivation with a parse tree that has some branch with b+1 variables.

Big Idea: We can show that a grammar in Chompsky normal form with b variables can generate an infinite number of strings if it has some derivation with a parse tree that has some branch with b+1 variables.

- Step 1) Argue why this is the case
- ▶ Step 2) Show that if some parse tree corresponds to 2^b steps in a derivation, than it must have a branch with b+1 variables.

 $A = \{ w \mid w \text{ has an equal number of 0s and 1s} \}$

 $A = \{w \mid w \text{ has an equal number of 0s and 1s}\}$

Implementation level description:

- Mark off the first unmarked symbol, if all symbols have been marked then accept.
- ▶ If the first symbol was a 0, scan through the tape and mark off first 1
- ▶ If the first symbol was a 1, scan through the tape and also mark off first 0
- ▶ If a 0 or 1 is not found, reject, else return to beginning of the tape and repeat.

$q010011 \sqcup$

*xq*10011⊔

qxx0011

xqx0011

xxq0011

xxxq011

xxx0q11

xxxq0x1

xxqx0x1

xqxx0x1

qxxx0x1

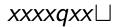
xqxx0x1

xxqx0x1

xxxq0x1

xxxxqx1

xxxxxq1



xxxqxxx

xxqxxxx

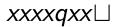
xqxxxxx

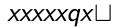
qxxxxxx

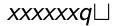
xqxxxxx

xxqxxxx

xxxqxxx







A slightly easier approach to implement...

q010011

 $\sqcup q10011 \sqcup$

$$q \sqcup x0011 \sqcup$$

 $\sqcup q \times 0011 \sqcup$

 $\sqcup xq0011 \sqcup$

 $\sqcup x \sqcup q011 \sqcup$

 $\sqcup x \sqcup 0q11 \sqcup$

$$\sqcup x \sqcup q0x1 \sqcup$$



$$\sqcup x \sqcup q0x1 \sqcup$$







