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:

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

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

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

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.

$$\begin{aligned} S &\rightarrow \; aSd \mid T \\ T &\rightarrow \; aTc \mid \epsilon \end{aligned}$$

Now lets apply the same trick to $a^m b^n c^p d^q$. Start by matching the outer as and ds. We don't know if there are more as or ds though.

Now lets apply the same trick to $a^m b^n c^p d^q$. Start by matching the outer as and ds. We don't know if there are more as or ds though.

- $m > q$: $a^q a^{m-q} b^n c^p d^q$.

Now lets apply the same trick to $a^m b^n c^p d^q$. Start by matching the outer as and ds. We don't know if there are more as or ds though.

- $m > q$: $a^q a^{m-q} b^n c^p d^q$.
  - $a^q T d^q$

Now lets apply the same trick to $a^m b^n c^p d^q$. Start by matching the outer *a*s and *d*s. We don't know if there are more *a*s or *d*s though.

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

Now lets apply the same trick to $a^m b^n c^p d^q$. Start by matching the outer *a*s and *d*s. We don't know if there are more *a*s or *d*s though.

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

Now lets apply the same trick to $a^m b^n c^p d^q$. Start by matching the outer as and ds. We don't know if there are more as or ds though.

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

Now lets apply the same trick to $a^m b^n c^p d^q$. Start by matching the outer *a*s and *d*s. We don't know if there are more *a*s or *d*s though.

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

$$\begin{aligned}
S &\rightarrow aSd \mid T \mid U \\
T &\rightarrow aTc \mid V \\
U &\rightarrow bUd \mid V \\
V &\rightarrow bVc \mid \epsilon
\end{aligned}$$

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

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:

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

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:

$$S \to ETB$$

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.

$$X \to 0X \mid 1X \mid \epsilon$$

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

$$S \rightarrow ETB$$

Consider E.

$$S \rightarrow ETB$$

Consider E.

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

$$S \to ETB$$

Consider E.

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

$$S \rightarrow ETB$$

Consider E.

- 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_ks separated by #s.
  - $E \rightarrow \#XE \mid \epsilon$
  - $E \rightarrow X\#E \mid \epsilon$
  - $E \rightarrow \#X\#E \mid \epsilon$

$$S \longrightarrow ETB$$

Consider E.

- x_i could be x_1 so $E \longrightarrow \epsilon$ must be a rule.
- If E isn't ϵ then it better be a sequence of x_ks separated by $\#$s.
  - $E \longrightarrow X\#E \mid \epsilon$

Where we are at...

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

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 $T \rightarrow 0 \mid 1 \mid \epsilon$.

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 $T \rightarrow 0 \mid 1 \mid \epsilon$.
- If $i \neq j$. There are a couple ways to think about this.

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

First attempt. Break $i \neq j$ into two cases.

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

First attempt. Break $i \neq j$ into two cases.

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

$$T \to 1T1 \mid 0T0 \mid 1 \mid 0 \mid \epsilon \mid \ldots$$

First attempt. Break $i \neq j$ into two cases.

- 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_ks separated by $\#$ as there need to be

$$T \to 1T1 \mid 0T0 \mid 1 \mid 0 \mid \epsilon \mid \ldots$$

First attempt. Break $i \neq j$ into two cases.

- 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_ks separated by $\#$ as there need to be
  - $T \to \#U\#$

$$T \to 1T1 \mid 0T0 \mid 1 \mid 0 \mid \epsilon \mid \ldots$$

First attempt. Break $i \neq j$ into two cases.

- 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_ks separated by $\#$ as there need to be
  - $T \to \#U\#$
  - $U \to X\#U \mid X$

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_ks 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:

$$S \rightarrow T \mid J\#T \mid T\#J \mid J\#T\#J$$
$$J \rightarrow 0J \mid 1J \mid \#J \mid \epsilon$$
$$T \rightarrow 1T1 \mid 0T0 \mid 1 \mid 0 \mid \epsilon \mid \# \mid \#J\#$$

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

$$\begin{aligned} S &\rightarrow aSd \mid T \\ T &\rightarrow aTc \mid \epsilon \end{aligned}$$

Convert to PDA.

Problem 9 part a.

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

- $aaa \in A$

Problem 9 part a.

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

- $aaa \in A$
- $aaaa \notin A$

Consider $w = 1^q$ where q is the smallest prime strictly larger the pumping length,p, of A. Now consider some decomposition $s = uvxyz$ as promised by the pumping lemma. So $s = 1^a 1^b 1^c 1^d 1^e$.

Consider $w = 1^q$ where q is the smallest prime strictly larger the pumping length,p, of A. Now consider some decomposition $s = uvxyz$ as promised by the pumping lemma. So $s = 1^a 1^b 1^c 1^d 1^e$.

- $|vxy| \leq p \Rightarrow b + c + d \leq p$

Consider $w = 1^q$ where q is the smallest prime strictly larger the pumping length,p, of A. Now consider some decomposition $s = uvxyz$ as promised by the pumping lemma. So $s = 1^a 1^b 1^c 1^d 1^e$.

- $|vxy| \leq p \Rightarrow b + c + d \leq p$
- $|vy| > 0 \Rightarrow b + d > 0$

Consider $w = 1^q$ where q is the smallest prime strictly larger the pumping length,p, of A. Now consider some decomposition $s = uvxyz$ as promised by the pumping lemma. So $s = 1^a 1^b 1^c 1^d 1^e$.

- $|vxy| \leq p \Rightarrow b + c + d \leq p$
- $|vy| > 0 \Rightarrow b + d > 0$
- $|s| > p \Rightarrow a + c + e > 0$

Consider $w = 1^q$ where q is the smallest prime strictly larger the pumping length,p, of A. Now consider some decomposition $s = uvxyz$ as promised by the pumping lemma. So $s = 1^a1^b1^c1^d1^e$.

- $|vxy| \leq p \Rightarrow b + c + d \leq p$
- $|vy| > 0 \Rightarrow b + d > 0$
- $|s| > p \Rightarrow a + c + e > 0$
- $uv^mxy^mz \in A \Rightarrow a + c + e + m * (b + d)$ is prime for any $m \geq 0$

Consider $w = 1^q$ where q is the smallest prime strictly larger the pumping length,p, of A. Now consider some decomposition $s = uvxyz$ as promised by the pumping lemma. So $s = 1^a 1^b 1^c 1^d 1^e$.

- $|vxy| \leq p \Rightarrow b + c + d \leq p$
- $|vy| > 0 \Rightarrow b + d > 0$
- $|s| > p \Rightarrow a + c + e > 0$
- $uv^m xy^m z \in A \Rightarrow a + c + e + m * (b + d)$ is prime for any $m \geq 0$
- Call $a' = a + c + e$, and note $a' > 0$.

Consider $w = 1^q$ where q is the smallest prime strictly larger the pumping length,p, of A. Now consider some decomposition $s = uvxyz$ as promised by the pumping lemma. So $s = 1^a 1^b 1^c 1^d 1^e$.

- $|vxy| \leq p \Rightarrow b + c + d \leq p$
- $|vy| > 0 \Rightarrow b + d > 0$
- $|s| > p \Rightarrow a + c + e > 0$
- $uv^m xy^m z \in A \Rightarrow a + c + e + m * (b + d)$ is prime for any $m \geq 0$
- Call $a' = a + c + e$, and note $a' > 0$.
- Call $b' = b + d$, and note $b' > 0$.

Consider $w = 1^q$ where q is the smallest prime strictly larger the pumping length,p, of A. Now consider some decomposition $s = uvxyz$ as promised by the pumping lemma. So $s = 1^a 1^b 1^c 1^d 1^e$.

- $|vxy| \leq p \Rightarrow b + c + d \leq p$
- $|vy| > 0 \Rightarrow b + d > 0$
- $|s| > p \Rightarrow a + c + e > 0$
- $uv^m xy^m z \in A \Rightarrow a + c + e + m * (b + d)$ is prime for any $m \geq 0$
- 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 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.

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.

$$q010011\sqcup$$

$$xq10011\sqcup$$

$$qxx0011\sqcup$$

$$xqx0011\sqcup$$

$$xxq0011\sqcup$$

$$xxxq011\sqcup$$

$$xxx0q11\sqcup$$

$$xxxq_0x1\sqcup$$

$$xxqx0x1\sqcup$$

$$xqxx0x1\sqcup$$

$$qxxx0x1\sqcup$$

$$xqxx0x1\sqcup$$

$$xxqx0x1\sqcup$$

$$xxxq0x1\sqcup$$

$$xxxxqx1\sqcup$$

$$xxxxxq1\sqcup$$

$$xxxxqxx\sqcup$$

$$xxxqxxx\sqcup$$

$$xxqxxxx\sqcup$$

$$xqxxxxx\sqcup$$

$$qxxxxxx\sqcup$$

$$xqxxxxx\sqcup$$

$$xxqxxxx\sqcup$$

$$xxxqxxx\sqcup$$

$$xxxxqxx\sqcup$$

$$xxxxxqx\sqcup$$

$$xxxxxxq\sqcup$$

A slightly easier approach to implement...

$$q010011\sqcup$$

$$\sqcup q 10011 \sqcup$$

$$q \sqcup x0011\sqcup$$

$$\sqcup q x 0011 \sqcup$$

$$\sqcup x q 0011 \sqcup$$

$$\sqcup x \sqcup q 011 \sqcup$$

$$\sqcup x \sqcup 0q11\sqcup$$

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

$$\sqcup x q \sqcup 0 x 1 \sqcup$$

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

$$\sqcup x \sqcup \sqcup q x 1 \sqcup$$

$$\sqcup x \sqcup \sqcup x q 1 \sqcup$$

$$\sqcup x \sqcup \sqcup q x x \sqcup$$

$$\sqcup x \sqcup q \sqcup xx \sqcup$$

$$\sqcup x \sqcup \sqcup q x x \sqcup$$

$$\sqcup x \sqcup \sqcup xqx\sqcup$$

$$\sqcup x \sqcup \sqcup xxq\sqcup$$