Recitation 10 - Homework 4 Solutions

John Chilton

June 8, 2007

Today

- Homework 4
- Practice Exam Problems

Problem 2 (Exercise 3.8b). Give an implementation-level description of a Turing machine which decidesthe following language.

$$\{w \mid w \text{ contains twice as many 0s and 1s}\}$$

We talked about an approach for doing something like this last week. Take this approach and adapt it.

$$A = \{w \mid w \text{ contains twice as many 0s as 1s}\}$$

Implementation level description:

$$A = \{w \mid w \text{ contains twice as many 0s as 1s}\}$$

Implementation level description:

- Mark off the first unmarked symbol, if all symbols have been marked then accept.

$$A = \{w \mid w \text{ contains twice as many 0s as 1s}\}$$

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, return back to the beginning and scan through and mark off the first 0.

$$A = \{w \mid w \text{ contains twice as many 0s as 1s}\}$$

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, return back to the beginning and scan through and mark off the first 0.
- If the first symbol was a 1, scan through the tape and mark off the first two 0s

$$A = \{w \mid w \text{ contains twice as many 0s as 1s}\}$$

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, return back to the beginning and scan through and mark off the first 0.
- If the first symbol was a 1, scan through the tape and mark off the first two 0s
- If you found a first symbol but the other one or two, *reject*, else return to beginning of the tape and repeat.

Problem 3 (Exercise 3.6). Theorem 3.21 states a language is Turing recognizable iff some enumerator enumerates it. Part of the proof was to construct an enumerator to enumerate the language recognized by some Turing machine M.

Problem 3 (Exercise 3.6). Theorem 3.21 states a language is Turing recognizable iff some enumerator enumerates it. Part of the proof was to construct an enumerator to enumerate the language recognized by some Turing machine M.

An enumerator is like a Turing machine, but instead of accepting or rejecting it prints out the strings of the language it enumerates. If E enumerates A it will only print out strings in A and given enough time it will print out any given string in A.

M *recognizes* A, $E_{:(}$ doesn't enumerate A, but $E_{:)}$ does. Why?

$E_{:(}$ = Ignore input.

1. Repeat for each string $s_i = s_1, s_2, s_3, \ldots$
2. Run M on s_i, if it accepts, print s_i

$E_{:)}$ = Ignore input.

1. Repeat for each string $i = 1, 2, 3,, \ldots$
2. Run M on $s_1, s_2, \ldots, s_i$ for i steps
3. Print each of the strings that are accepted, if any

Problem 4. Explain why the following is not a description of a legitimate Turing machine.

$M_{\text{bad}} =$ The input is a polynomial p over variables $x_1, \ldots, x_k$.

1. Try all possible settings of $x_1, \ldots, x_k$ to integer values.
2. Evaluate p on all of these settings.
3. If any of these settings evaluates to 0, *accept*; else, *reject*.

Problem 6. Consider a form of Turing machines where instead of having the head having the options to go left or right at each step, its options are to move to the right or stay put. Show that this variant has less power than Turing machines, and argue about what class of languages it does recognize. (Hint: Argue about the class of languages it recognizes first.)

This variant of turning machine can only recognize regular languages. Why?

This variant of turning machine can only recognize regular languages. Why? No memory. If you move to the left, can't move right, can't bring with you what was on the tape, all you know is the current input and the state you are in. This is just like a DFA. Technically you have one character of memory, but that can simulated in the DFA using the state.

Problem 7. (Problem 3.16 from text) Show that Turing-recognizable languages are closed under: concatenation (2), star (3), and intersection (1).

Start: Let A_1 and A_2 be two Turing-recognizable languages, and let M_1 and M_2 be two Turing-machines that recognize the respective languages.

Start: Let A_1 and A_2 be two Turing-recognizable languages, and let M_1 and M_2 be two Turing-machines that recognize the respective languages.

- Construct Turing machine M that recognizes $A_1 \cap A_2$ using M_1 and M_2.
- Construct Turing machine M that recognizes $A_1 \circ A_2$ using M_1 and M_2.
- Construct Turing machine M that recognizes A_1^* using M_1.
- Remember M_1 and M_2 recognize, not decide A_1 and A_2.

Start: Let A_1 and A_2 be two Turing-recognizable languages, and let M_1 and M_2 be two Turing-machines that recognize the respective languages.

- Construct Turing machine M that recognizes $A_1 \cap A_2$ using M_1 and M_2.
- Construct Turing machine M that recognizes $A_1 \circ A_2$ using M_1 and M_2.
- Construct Turing machine M that recognizes A_1^* using M_1.
- Remember M_1 and M_2 recognize, not decide A_1 and A_2.

Don't use diagram or implementation level description. Use pseudo code, examples on page 153, and 163 toward bottom.

Let A_1 and A_2 be two Turing-*decidable* languages, and let M_1 and M_2 be two Turing-machines that *decide* the respective languages. The following machine M decides $A_1 \cap A_2$, hence Turing-decidable languages are closed under intersection.

$M :=$ "On input w:

- Run M_1 on input w, if it rejects, *reject*
- Run M_2 on input w, if it rejects, *reject*
- Else, *accept*."

Let A_1 and A_2 be two Turing-*recognizable* languages, and let M_1 and M_2 be two Turing-machines that *recognize* the respective languages. The following machine M recognizes $A_1 \cap A_2$, hence Turing-recognizable languages are closed under intersection.

$M :=$ "On input w:

- Repeat the following for $i = 1, 2, 3, \ldots$
-     Run M_1 on input w for i steps
-     Run M_2 on input w for i steps
-     If M_1 and M_2 both accepted, *accept*, else continue"

Let A_1 and A_2 be two Turing-*decidable* languages, and let M_1 and M_2 be two Turing-machines that *decide* the respective languages. The following machine M decides $A_1 \circ A_2$, hence Turing-decidable languages are closed under concatenation.

$M :=$ "On input w:

- For $j = 0, 1, ..., |w|$
- Run M_1 on the first j symbols of w
- Run M_2 on the remaining symbols of w
- If both machines accept for some j, *accept*
- Else *reject*."

Let A_1 and A_2 be two Turing-*recognizable* languages, and let M_1 and M_2 be two Turing-machines that *recognize* the respective languages. The following machine M recognizes $A_1 \circ A_2$, hence Turing-decidable languages are closed under concatenation.

$M :=$ "On input w:

- For $i = 1, 2, 3, ...$
- For $j = 0, 1, ..., |w|$
- Run M_1 on the first j symbols of w for i steps
- Run M_2 on the remaining symbols of w for i steps
- If both machines accept for some j, *accept*

Let A_1 be a Turing-*decidable* language, and let M_1 be a Turing-machines that *decides* the language. The following machine M decides A_1^*, hence Turing-decidable languages are closed under star.

$M :=$ "On input w:

- For each of the $2^{|w|-1}$ ways to split w into non-empty substrings:
- Run M_1 on each of these substrings
- If the machine accepts for each substring, *accept*
- Else *reject*."

Let A_1 be a Turing-*recognizable* language, and let M_1 be a Turing-machines that *recognizes* the language. The following machine M recognizes A_1^*, hence Turing-recognizable languages are closed under star.

$M :=$ "On input w:

- For $i = 1, 2, 3, \ldots$
- For each of the $2^{|w|-1}$ ways to split w into non-empty substrings:
- Run M_1 on each of these substrings for i steps
- If the machine accepts for each substring, *accept*

Construct a CFG for:

$$\{w \mid w = a^m b^n \text{ for } n \leq m \leq 2n\}$$

Is this grammar ambiguous?

Construct a PDA for:

$$\{w \mid \text{the number of } a\text{s and } b\text{s equals the number of } c\text{s and } d\text{s}\}$$

Construct a PDA for:

$$\{w \mid \text{the number of } a\text{s and } b\text{s equals the number of } c\text{s and } d\text{s}\}$$

Rules:

- S - generates equal number of $as + bs$ as $cs + ds$.

Rules:

- S - generates equal number of as + bs as cs + ds.
- E - generates one more c+ds than a+bs.

Rules:

- S - generates equal number of as + bs as cs + ds.
- E - generates one more c+ds than a+bs.
- F - generates one more a+bs than c+ds.

Show that the following language is not context-free.

$$A = \{a^{n^2}\}$$

Show that the following language is not context-free.

$$A = \{a^{n^2}\}$$

Assume A is context free, and use the pumping lemma to derive contradiction.

Consider $s = a^{p^2}$, and assume some decomposition $s = uvxyz$ as defined by the pumping lemma exists.

Consider $s = a^{p^2}$, and assume some decomposition $s = uvxyz$ as defined by the pumping lemma exists.

- Consider uv^2xy^2z.

Consider $s = a^{p^2}$, and assume some decomposition $s = uvxyz$ as defined by the pumping lemma exists.

- Consider uv^2xy^2z.
- $|vy| > 0 \implies |uv^2xy^2z| > |s| = p^2$

Consider $s = a^{p^2}$, and assume some decomposition $s = uvxyz$ as defined by the pumping lemma exists.

- Consider uv^2xy^2z.
- $|vy| > 0 \implies |uv^2xy^2z| > |s| = p^2$
- $|vxy| \leq p \implies |uv^2xy^2z| \leq |s| + p = p^2 + p$

Consider $s = a^{p^2}$, and assume some decomposition $s = uvxyz$ as defined by the pumping lemma exists.

- Consider uv^2xy^2z.
- $|vy| > 0 \implies |uv^2xy^2z| > |s| = p^2$
- $|vxy| \leq p \implies |uv^2xy^2z| \leq |s| + p = p^2 + p$
- So $p^2 \leq |uv^2xy^2z| \leq p^2 + p < p^2 + 2p + 1 = (p+1)^2$.

Consider $s = a^{p^2}$, and assume some decomposition $s = uvxyz$ as defined by the pumping lemma exists.

- Consider uv^2xy^2z.
- $|vy| > 0 \implies |uv^2xy^2z| > |s| = p^2$
- $|vxy| \leq p \implies |uv^2xy^2z| \leq |s| + p = p^2 + p$
- So $p^2 \leq |uv^2xy^2z| \leq p^2 + p < p^2 + 2p + 1 = (p+1)^2$.
- Hence $|uv^2xy^2z|$ lies between two consecutive perfect squares and cannot be in A.