Problem Set 7

Fall 11

Due: Thursday, 1st December, 2011, 11:00 am before class begins
Please follow the homework format guidelines posted on the class web page:
`http://www.cs.uiuc.edu/class/fa11/cs373/`

1. [**Category**: Aliens, **Points**: 5]

   Prove that the problem of deciding whether there are aliens is decidable or undecidable. More precisely, is there a TM that will take as input "Are there aliens?" and accepts it if there are aliens, and rejects it if there aren't any. (The TM rejects all other strings.)

Solution:

Proof: This problem is decidable. Consider two TM T_Y and T_N. T_Y does nothing but accepts, and T_N does nothing but rejects. Then if there are aliens, T_Y will be a decider for this problem; otherwise, T_N will be a decider for this problem. In either case, there must be a decider for the problem (either T_Y or T_N). Hence this problem is decidable. ■

2. [**Category**: Undecidability, **Points**: 20]

   Let L be the set of all encoding of Turing machines and words, $\langle M, w\rangle$ such that M when run on w at some point moves right for three consecutive steps. Prove that L is undecidable.

Solution:

Proof: We will show this by reducing A_{TM} to L. Since A_{TM} is undecidable, it follows that L is undecidable.

Assume there is a decider M that decides L. The Turing machine M_u deciding A_{TM} will, on input $\langle M, w\rangle$, build a TM M_w such that $\langle M_w, w\rangle \in L$ iff M accepts w. It will feed $\langle M_w, w\rangle$ to M to figure out whether M_w moves right for three consecutive steps on w, from which it will deduce whether M accepts w.

Intuitively, we build the TM M_w by extending M with more state to remember how many consecutive right moves have been made recently (we only remember zero, one or two steps, the number of states is still finite). If the recent consecutive right moves is only zero or one, then M_w simulates the behavior of M. If the recent consecutive right moves are two, then M_w still simulate M if M moves left or does not move; otherwise M_w will make one left move and then two right moves. When M accepts, M_w simulates it by making three consecutive moves and then accepts.

Basically M_w simulates M without making three consecutive right moves, but it always make three consecutive right moves before accepting. Hence M_w makes three consecutive right moves on w iff M accepts w. Hence the Turing machine M_u above decides A_{TM}.

■

3. [**Category**: CFG design, **Points**: 20]

   Consider well-formed arithmetic expressions on numbers with four binary operators $\{+, -, *, /\}$ and one unary operator $\{-\}$ (negative sign). A number is any string over $\{0, 1, \ldots 9\}$ (starting with 0s is fine). To avoid ambiguity, consider expressions which are parenthesized every time an operation is used. Design a context-free grammar for arithmetic expressions. That is construct a grammar G such that $L(G)$ is the set of all valid arithmetic expressions.

   Here are three examples that should be in $L(G)$:

   $((((1335 + 21) * 3222) - 431)/565)$

   $(745 - (-((003 - (101 + 134545452)) + (345 - 4453))))$

   $(1/0)$

   Here are five examples that should NOT be in $L(G)$:

   $(1 + 2 - 3 * 4)$

   $(1 - -(2 + 3))$

   $(2 + 3($

   $1 + 2$

   $((1) + 2)$

   After your construction, show the following two strings are valid arithmetic expressions by explicitly showing every yield step of applying rules in G.

   (a) $(2 + (-(1 * 3)))$

   (b) $((4/5) + (5 * (6 + 7)))$

Solution:

$S \to (S + S) \mid (S - S) \mid (S * S) \mid (S/S) \mid (-S) \mid N$
$N \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \mid NN$

4. [**Category**: CFG Design, **Points**: 20]

   We want to show that a subset of HTML documents is a context-free language. For our purposes, we will consider a subset of HTML restricted to the tags: `html, body, ul, li`. In particular, the document must have the open tags and close tags matched properly, and satisfy the following conditions:

- The document must start with an open `<html>` tag and close with `</html>` and there should be no other `html` tag and all text must be contained within these tags.
- There is only one open `body` tag (and its matching close tag)
- All `ul` tags occur within the `body` block. There can be any number of `ul` blocks, and all `li` must blocks occur within an immediate `ul` block. A `ul` block need not have any `li` blocks within it.
- There can be text anywhere within the `<html>` block, between any tags.
- Text is any sequence of a–z, A–Z, and the space character.

Hence such documents start with the `html` tag followed by some text followed by a `body` block. The `body` block consists of nested `ul` blocks that have sequences of `li` blocks, and text in between the tags.

For example, the following is a well-formed document:

```
<html> Heading <body> Blah Blah <ul><li> first item </li><li>second</li>
<ul><li>This is nested at second level</li></ul></ul></body> </html>
```

Solution:

$S \to$ `<html>` E `<body>` B `</body>` E `</html>`
$B \to EB \mid$ `<ul>` U `</ul>` $B \mid \epsilon$
$U \to EU \mid$ `<ul>` U `</ul>` $U \mid$ `<li>` B `</li>` $U \mid \epsilon$
$E \to EE \mid a \mid \cdots \mid z \mid A \mid \cdots \mid Z \mid \sqcup \mid \epsilon$

5. CNF Conversion [**Category**: Proof., **Points**: 7+7+6] Consider the grammar G:

$$\begin{array}{lcl} S & \to & 0A0 \mid 1B1 \mid BB \\ A & \to & C \\ B & \to & S \mid A \\ C & \to & S \mid \epsilon \end{array}$$

(a) First, add a rule $S_0 \to S$ to G and eliminate ϵ-productions, obtaining G_1. Write down precisely the set of nullable variables, and the resulting grammar G_1.

(b) Eliminate any unit productions in G_1, obtaining G_2. Write down precisely the set of all transitive unit derivations, and the resulting grammar G_2.

(c) Put G_2 into Chomsky Normal Form G_3.

Solution:

(a) First of all, add a new start variable S_0 with $S_0 \to S$. The set of nullable variables are $\{S_0, S, A, B, C\}$. Adding productions that replace each appearance of nullable variables by ϵ obtains G_1:

$$\begin{array}{lcl} S_0 & \to & S \mid \epsilon \\ S & \to & 0A0 \mid 00 \mid 1B1 \mid 11 \mid BB \mid B \\ A & \to & C \\ B & \to & S \mid A \\ C & \to & S \end{array}$$

(b) The unit rules in G_1 are: $S_0 \to S, S \to B, A \to C, B \to S, B \to A, C \to S$. After elimination we get G_2:

$$\begin{array}{lcl} S_0 & \to & 0A0 \mid 00 \mid 1B1 \mid 11 \mid BB \mid \epsilon \\ S & \to & 0A0 \mid 00 \mid 1B1 \mid 11 \mid BB \\ A & \to & 0A0 \mid 00 \mid 1B1 \mid 11 \mid BB \\ B & \to & 0A0 \mid 00 \mid 1B1 \mid 11 \mid BB \\ C & \to & 0A0 \mid 00 \mid 1B1 \mid 11 \mid BB \end{array}$$

(c) We introduce two new rules $P \to 0, Q \to 1$ to eliminate mixing rules. Then we also introduce $X \to AP, Y \to BQ$ to eliminate long rules:

$$\begin{array}{lcl} S_0 & \to & PX \mid PP \mid QY \mid QQ \mid BB \mid \epsilon \\ S & \to & PX \mid PP \mid QY \mid QQ \mid BB \\ A & \to & PX \mid PP \mid QY \mid QQ \mid BB \\ B & \to & PX \mid PP \mid QY \mid QQ \mid BB \\ C & \to & PX \mid PP \mid QY \mid QQ \mid BB \\ X & \to & AP \\ Y & \to & BQ \\ P & \to & 0 \\ Q & \to & 1 \end{array}$$

6. CYK [**Category**: Comprehension, **Points**: 20]

Use CYK algorithm to determine whether or not the given string belongs to the grammar. Your answer should include either "yes" or "no" and a chart that you built using CYK.

You are required to use the CYK algorithm; do not just give a derivation or an argument as to why the word does not belong to the language.

Determine whether the string (i) *aabbbb* and (ii) *aabaab* belong to the language.

$$
\begin{aligned}
S &\longrightarrow AP \mid AB \\
E &\longrightarrow AP \mid EB \mid b \\
P &\longrightarrow EB \\
A &\longrightarrow a \\
B &\longrightarrow b
\end{aligned}
$$

Solution:

aabbbb - yes, *aabaab* - no.

S,E,P					
S,E	S,E,P				
∅	S,E,P	E,P			
∅	S,E	E,P	E,P		
∅	S	E,P	E,P	E,P	
A	A	B,E	B,E	B,E	B,E
a	a	b	b	b	b

∅					
∅	∅				
∅	∅	∅			
∅	∅	∅	∅		
∅	S	∅	∅	S	
A	A	B,E	A	A	B,E
a	a	b	a	a	b