Automata Qual Exam (Spring 2011)
Answer ALL questions (Closed Book Exam)

Question 1

We consider the concept of Queue Automata which makes use of a queue. (In contrast, a PDA uses a stack.)

Assume that there are four queue operations: `isEmptyQueue, front, enqueue`, and `dequeue` where `front` returns the front element without removing it, while `dequeue` removes the front element.

(a) (10 points) Give a formal definition of a Queue Automaton. Specifically, define the transition function in detail, explain how the new configuration (instantaneous description) is computed, and define when a string is accepted.

(b) (15 points) Explain how a Turing Machine can be simulated by a Queue Automaton.

Answer: We represent the tape contents by the queue. Consider the tape contents:

```
a b a a b* b b a ..........
                (blank symbols)
```

where $*$ indicates the current position of the tape head. As a first attempt, the corresponding queue contents are:

```
b b b a $ a b a a
          ^
          tape begins here
```

with the front of the queue pointing to the symbol currently read by the tape head, and the delimiter symbol \$ indicates the beginning the tape head. If the Turing machine changes the tape content from b to a followed by a move to the right, the Queue Automaton dequeues, followed by enqueue-ing of the symbol a. For a left move of the Turing Machine, there is a technical difficulty when 'cycling' through the queue to bring the last symbol of the queue to the front. After dequeue-ing the first symbol, we need to mark it special before enqueue-ing. However, after cycling through the contents for a number of steps, and by the time we see the specially marked symbol again, we would have passed our target for stopping. There is no simple way to 'peek' at the next symbol behind the

front of the queue. We can solve this difficulty by modifying the representation with the current symbol maintained in the state of the Queue Automaton. Together with the `front` operation (which does not dequeue the front element), we can achieve the 'peeking' effect needed to stop the cycling at the right position. Another aspect that needs to discuss is when the Turing Machine moves into the area holding only blank symbols. In that case, the Queue Automaton is currently maintaining the rightmost symbol of the tape in the state, while the `front` operation returns the $ symbol. To simulate a move into the blank area of the Turing tape, we just need to enqueue our current symbol (maintained by the state), then re-new the current symbol (again maintained by the state) with a blank symbol.

Question 2

Given two DFAs (deterministic finite automata) $M = (P, \Sigma, \delta, p_0, F)$ and $N = (Q, \Sigma, \delta', q_0, F')$. We consider constructing a new DFA that simulates simulaneously both DFAs in parallel. (Some books may call this the "cartesian" product of two DFA.)

(a) (5 points) Give a formal definition of the cartesian product of M and N.

Answer: $(P \times Q, \Sigma, \delta", (p_0, q_0), F")$ where $\delta"((p, q), a) = (\delta(p, a), \delta'(q, a))$.

(b) (10 points) Explain how one can decide if two given DFAs are equivalent (in that both recognize the same language) by inspecting the cartesian product of the two DFAs.

Answer: Choose $F" = \{(p, q) \mid p \in F,\ q \notin F'\} \cup \{(p, q) \mid p \notin F,\ q \in F'\}$. M and N are equivalent if the product automaton recognizes an empty set of strings, which can be determined by running a depth first search on the graph of the transitions, and see if any accepting state is reached.

Question 3 (20 points)

Consider the following grammar:

$\text{exp} \longrightarrow (\ \text{exp}\ )\ \mid\ \text{exp} + \text{exp}\ \mid\ \text{exp} * \text{exp}\ \mid\ \text{exp}\ \hat{}\ \text{exp}\ \mid\ 0\ \mid\ 1$

Suppose $+$ and $*$ are left associative; but $\hat{}$ is right associative. Also, suppose $\hat{}$ is of higher precedence than $*$, which again is of higher precedence than $+$. Re-write the grammar so that it reflects the intended meaning given, and is unambiguous.

Answer:
exp $\longrightarrow$ exp + term | term
term $\longrightarrow$ term * factor | factor
factor $\longrightarrow$ primary ^ factor | primary
primary $\longrightarrow$ 0 | 1 | (exp)

Question 4

Consider the pumping lemma for context-free languages. In the textbooks, the proof of the pumping lemma is explained with reference to parse trees for strings that are long enough.

(a) (20 points) Given a specific parse tree, explain how one can determine if the parse tree can be "pumped". That is, give a syntactic property that characterizes "pump-able" parse trees.

Answer:
A parse tree can be pumped when a tree node n and one of its descendent node n' are both labeled by the same nondeterminal. *One can also talk about whether the pumping is proper in that pumping up (down) will increase (decrease) the length the yield.*

(b) (20 points) Consider a specific configuration (instantaneous description) sequence of the computation of a PDA on an input string. Explain how one can determine if the computation can be "pumped". That is, give a syntactic property that characterizes "pump-able" configuration (instantaneous description) sequence. **Hint:** Translate the syntactic condition from part (a) about parse tress to a condition about configuration (instantaneous description) sequences of a PDA.

Answer:
A configuration sequence is pump-able if there exists four time instants $t_1 < t_2 < t_3 < t_4$ which configurations are C_1, C_2, C_3, C_4 respectively such that
(1) the PDA are at the same state and same top stack symbol in C_1 and C_2,
(2) the PDA are at the same state and same top stack symbol in C_3 and C_4,
(3) the stack has the same height at C_1 and C_4,
(4) the stack has the same height at C_2 and C_3,
(5) the stack height at C_2 is higher than that at C_1,
(6) between C_1 and C_2, stack symbols below the top stack symbol in C_1 were not accessed,
(7) between C_3 and C_4, stack symbols below the top stack symbol in C_4 were not accessed,
(8) between C_2 and C_3, stack symbols below the top stack symbol in C_2 were not accessed.