Advanced Topics in Automata
Exercise 7

Submission: June 17, 2003

Exercise

A *weak alternating automaton* is an alternating Büchi automaton that in addition has the following structure. There is a partial order on the states of the automaton such that the transition of a state can go only to states that are lower in the partial order. Furthermore, every equivalence class of the partial order is either contained in the set of accepting states or disjoint from the set of accepting states. A weak alternating automaton where every equivalence class of the partial order contains exactly 1 state is called 1-weak. These limitations make weak alternating automata very interesting. For one, complementation of weak alternating automata is linear. Checking the emptiness of an AWW with 1-letter input alphabet is linear (not quadratic).

It turns out that the alternating automaton that results from the translation of an LTL formula is 1-weak. The alternating Büchi automaton that we constructed in the last class is weak. More formally, we have the following.

Consider an alternating Büchi automaton $A = \langle \Sigma, Q, q_0, \rho, F \rangle$. We say that A is *weak* if there exists a partition $Q_1, \ldots Q_m$ of Q and there is a partial order $\leq$ on the sets such that q' may appear in transitions of q only if $[q'] \leq [q]$ (where $[q]$ denotes the set Q_i such that $q \in Q_i$). Furthermore, we demand that forall i either $Q_i \subseteq F$ or $Q_i \cap F = \emptyset$. We say that A is *1-weak* if every set Q_i contains exactly one state.

70% Prove that complementation of alternating weak automata is linear.

15% Show that the automaton that is the result of the translation from LTL to alternating automata is 1-weak.

15% Show that the automaton that is the result of the complementation construction of Kupferman and Vardi (taught last lecture) is weak.

0% **Food for thought.** Give a linear algorithm for checking the emptiness of AWW with 1-letter input alphabet.

Food for thought

A deterministic automaton, has exactly one run over every word. We determine whether the word is accepted or not by checking whether this run is fair or not. A *backward deterministic* automaton has exactly one fair run over every word. We determine acceptance by checking whether this run starts from an initial state or not.

Consider, a (presumably) NBW $N = \langle \Sigma, Q, Q_0, \delta, F \rangle$. We say that N is *backward deterministic* if for every word $w = w_0, w_1, \ldots \in \Sigma^\omega$ there exists exactly one sequence $r = t_0, t_1, \ldots \in Q^\omega$ such that:

- Forall i we have $t_{i+1} \in \delta(t_i, w_i)$.
- $inf(r) \cap F \neq \emptyset$.

We say that a run $r = t_0, \ldots$ of a backward deterministic automaton A accepts a word w iff $t_0 \in Q_0$.

Apparently, the NBW constructed for an LTL formula (not in the method that we studied in class) is backward deterministic.

Use the following facts to show that for every NBW we can construct a backward deterministic automaton accepting the same language.

- For every NBW we can construct a weak alternating automaton accepting the same language and another that accepts the complementary language.

  The connection between the two should be established as part of the homework.

- For an ABW A let $\overline{A}$ denote ACW complement of A. Let Q denote the set of states of A and $\overline{Q} = \{\overline{q} \mid q \in Q\}$ denote the set of states of $\overline{Q}$. Every word is either accepted by A_q or $\overline{A}_{\overline{q}}$ where A_q is the automaton A with q as initial state. It is never the case that some word is accepted by both A_q and $\overline{A}_{\overline{q}}$.

- We can represent a subset of the states of A as $\{0,1\}^Q$. A 0 stands for the state is not in the subset and a 1 stands for the state is in the subset. Having a nondeterministic automaton in state $Q' \subseteq Q$ means that the suffix should be accepted from all the states in Q'. Equivalently consider the set $\{-1,1\}^Q$ where -1 stands for $\overline{q}$ should accept the suffix and 1 stands for q should accept the suffix.