AGTA Tutorial sheet 6 (solutions)

Question 1

Consider a game graph $G = (V, E, v_0, F)$, where the (finite) set of vertices $V = V_1 \cup V_2$ is partitioned into set of vertices $V_1$ (belonging to player 1) and set of vertices $V_2$ (belonging to player 2), $E$ being the set of edges of $G$. We are also given a start vertex $v_0 \in V$, and a set $F \subset V$ of target (good) vertices. Denote $E(v)$ the set of all successor vertices for $v$, i.e $E(v) = \{v' \in V | (v, v') \in E\}$. We can assume, without loss of generality, that $\forall v, E(v) \neq \emptyset$, i.e. every vertex has at least one successor vertex. (This is because if there were any vertices without successors we could just add an edge from any such vertices to a new “absorbing dead-end” vertex with a self-loop to itself, so that if we every reach a vertex without any successor we will never thereafter reach any vertex in $F$.)

Let $\Pi$ denote the set of infinite paths in $G$. For $\pi \in \Pi$, $\pi = v_0v_1...$, let us denote the set of states that appear infinitely often in $\pi$ as $\text{inf}(\pi) = \{v \in V | \forall i \geq 0, \exists j \geq i, v_j = v\}$.

The play $\pi$ is a win for player 1 if $\text{inf}(\pi) \cap F \neq \emptyset$, and otherwise it is a win for player 2 (i.e. a loss for player 1).

Our goal is an efficient algorithm for computing which player has a winning strategy, and for computing a memoryless winning strategy for that player, given such a game, starting at the start vertex $v_0$.

For any set of nodes $S \subseteq V$, let $\text{Win}'_1(S)$ denote the set of nodes $v \in V$ such that starting from $v$ player 1 has a winning strategy to force the game to reach a vertex in $S$ in at least one or more steps (so, we do not necessarily have $S \subseteq \text{Win}'_1(S)$).

It is easy to adapt the algorithm given in class for the win-lose reachability game on a graph (see the slides for Lecture 12, page 8), to compute the set $\text{Win}'_1(S)$. Namely, given the set $S \subseteq V$, the algorithm for computing $\text{Win}'_1(S)$ is as follows:

1. Initialize: $\text{Win}'_1 := \{v \in V_1 | \exists (v, v') \in E \text{ such that } v' \in S\} \cup \{v \in V_2 | \forall (v, v') \in E : v' \in S\};$
   $\text{St}_1 := \emptyset;$

2. Repeat
   Foreach $v \notin \text{Win}'_1$:
   If $pl(v) = 1$ & $\exists (v, v') \in E : v' \in \text{Win}'_1$
   $\text{Win}'_1 := \text{Win}'_1 \cup \{v\}$; $\text{St}_1 := \text{St}_1 \cup \{v \mapsto v'\}$;
   If $pl(v) = 2$ & $\forall (v, v') \in E : v' \in \text{Win}'_1 \cup S$
   $\text{Win}'_1 := \text{Win}'_1 \cup \{v\}$;

Until The set $\text{Win}'_1$ does not change;
Return($\text{Win}'_1$)
When this algorithm halts, the final set $Win'_1$ that it returns is precisely the set $Win'_1(S)$. Moreover, the algorithm also computes the (partial) pure memoryless strategy $St_l$ for player 1 such that using this strategy, starting from every vertex $v \in Win'_1(S)$, the play will reach a vertex in $S$ in one or more steps, no matter what player 2 does.

Furthermore, we can also use the computed set $Win'_1(S)$ to easily derive a memoryless strategy $St_2$ for player 2 which makes sure that, starting from any vertex $v \notin Win'_1(S)$, no matter what player 1 does, the set $S$ will not be reached in one or more steps. Namely, the strategy $St_2$ for player 2 is constructed as follows: since we have assumed (without loss of generality) that every vertex has at least one outgoing edge, it follows that for every vertex $v \notin Win'_1(S)$ such that $v \in V_2$, there must exist an outgoing edge $(v, v') \in E$ to a vertex $v' \notin Win'_1 \cup S$ (otherwise, $v$ would have been added to $Win'_1$ during the algorithm). The strategy $St_2$ for player 2 simply chooses one such outgoing edge $v \mapsto v'$ for every such vertex $v \in V_2$ such that $v \notin Win'_1(S)$. (For all other vertices $v \in V_2$ where $v \in Win'_1(S)$, it doesn’t matter what edge player 2 chooses to play from those vertices in its memoryless strategy $St_2$.)

Suppose $F$ is the set of “target” vertices that player 1 would like to visit infinitely often in the game. Let us now consider the following sequence of subsets of $F$.

Let $F_0 := F$, and for all integers $i \geq 0$, let

$$F_{i+1} := F \cap Win'_1(F_i)$$

(We could also equivalently define $F_{i+1} := F_i \cap Win'_1(F_i)$.)

It can be shown, by induction on $i$, that $F_i$ denotes the set of target vertices $v \in F$ starting from which player 1 has a strategy to revisit target vertices in $F$ at least $i$ times. Note that we can compute the sets $F_{i+1}$ iteratively, by repeatedly applying the above algorithm to compute the sets $Win'_1(F_i)$, $i = 0, 1, 2, \ldots$ It is also easy to show (by induction on $i$) that:

$$F = F_0 \supseteq F_1 \supseteq F_2 \supseteq F_3 \ldots$$

In other words, $F_i \supseteq F_{i+1}$, for all $i \in \mathbb{N}$.

Thus, for each $i \in \mathbb{N}$, either $F_{i+1} \subset F_i$, in which case $|F_{i+1}| \leq |F_i| - 1$, or else $F_{i+1} = F_i$. Now, notice that $F$ is a finite set. Thus if $|F| = k$ then, clearly there is some smallest $i \leq k$ such that $F_i = F_{i+1} = F_{i+2} = \ldots$.

Let $F^*$ be equal to $F_i$ for the smallest $i \leq k$ such that $F_i = F_{i+1}$.

Note that player 1 has a memoryless strategy, $St_1$, computed by the algorithm that computes $Win'_1(F^*)$, such that starting at every $v \in F^*$, using $St_1$, the play reaches a vertex in $F^*$ in one or more steps, regardless what player 2 does. Hence in fact (by reusing that same memoryless strategy $St_1$), starting from every vertex $v \in F^*$ player 1 has a strategy to infinitely often visit a node in $F^*$, regardless what player 2 does.

Finally, $Win'_1(F^*)$ is the set of all vertices (including vertices not in $F$) starting from which player 1 has a winning strategy to force visiting vertices
in $F$ infinitely often, and the memoryless strategy to do so is given by $St_1$ computed when we compute $Win'_1(F^*)$.

The algorithm for solving a reachability game on a graph (i.e., computing a set $Win'_1(S)$), can be done in linear time $O(|E| + |V|)$ in the size of the game graph. Thus computing each $F_i$ at iteration $i$ of the algorithm requires $O(|E| + |V|)$ running time. Since there are at most $|F| = k$ iterations, the running time of the algorithm is $O(|F|(|E| + |V|))$. 