AGTA Tutorial sheet 6 (solutions)

1 Question 1

Consider a game graph $G = (V, E, v_0, F)$, where the (finite) set of vertices $V = V_1 \bigcup V_2$ 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 good (target) states. Denote $E(v)$ the set of all successor vertices for v, i.e $E(v) = \{v' \in V | (v, v') \in E\}$. We assume that $\forall v, E(v) \neq \varnothing$, i.e every state has a successor state (no deadlocks). Let Π denote the set of infinite paths in G. For $\pi \in \Pi$, $\pi = v_0 v_1 ...$, let us denote the set of states that appear infinitely often in π as

$$inf(\pi) = \{v \in V | \forall i \geq 0, \exists j \geq i, v_i = v\}$$

The play π is a win for player 1 if $inf(\pi) \cap F \neq \varnothing$, and otherwise it is a win for player 2 (i.e a loss for player 1). Describe an efficient algorithm to find which player wins, and to extract a memoryless winning strategy, given such a game.

1.1 An Algorithm

For any set of nodes $S \subseteq V$, let $Win'_1(S)$ 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 one or more steps (so, we do *not* necessarily have $S \subseteq Win'_1(S)$).

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 $Win'_1(S)$.

Namely, in that algorithm, in step 1., instead of initializing $Win_1 := Good$, we simply instead initialize Win_1 as follows

$$Win_1 := \{v \in V_1 \mid \exists (v, v') \in E \text{ such that } v' \in S\} \cup \{v \in V_2 \mid \forall (v, v') \in E \; v' \in S\}$$

(In the above V_1 denotes the vertices of the game graph belonging to player 1, and V_2 denotes the vertices belonging to player 2.)

The rest of that algorithm remains unchanged. This revised version of that algorithm will compute precisely the set $Win'_1(S)$. Note that algorithm also computes, a memoryless strategy for player 1, such that using that strategy, starting from every vertex $v \in Win'_1(S)$ the play will reach a vertex in S (no matter what player 2 does).

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

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

In other words, F_i denotes the set of target vertices $v \in F$ such that player 1 has a strategy to revisit target vertices in F at least i times, starting from v.

Note that we can computing the sets F_{i+1} by just repeatedly using reachability game algorithm to compute the sets $Win'_1(F_i)$.

It is easy to show (by induction on i) that:

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

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

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

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

But then, player 1 has a memoryless strategy, using which, starting at every $v \in F^*$, the play reaches a vertex in F^* in one or more steps. Hence in fact (by reusing those same memoryless strategies starting from each node in F^*), starting from every vertex $v \in F^*$ player 1 has a strategy to infinitely often visit a node in F^*.

Finally, $F^* \cup Win'_1(F^*)$ 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.

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 $O(|F|(|E| + |V|))$.