AAGT Homework #2

Jakub Grondziowski

I'm assuming that in this game there are no sinks (No way for the game to ever end, as there are always some edges exiting from every node). I'm also assuming that the word α is finite. I will start with describing the algorithm first.

Task c

First step of solving this game is to detect places where Eve can guarantee that the word α will occur, regardless of what moves Adam will make. To detect such places, we first need to find all vertices that are marked with the last letter of word α. Let's call this set F^0 (set F^n will denote the set of currently regarded nodes at n-th step of algorithm below.

Now for the algorithm: For the duration of the algorithm we keep track of the position p in word α that we are currently investigating. We of course start with $p = length(\alpha) - 1$. For each set of vertices F we look for predecessors of all $v \in F^n$. We do so by linearly checking all nodes and their exiting edges. If the checked node is a predecessor of some $v \in F^n$ and the letter that marks the node is equal to letter at position $p - 1$ in word α, we can add the node to the next set F^{n+1} if and only if:
- This node belongs to Eve,
- This node belongs to Adam and all exiting edges lead to vertices that are in F^n.

Note that in the worst case, when Adam has a node which exits to multiple other nodes, we only add it to the next set if there is no way for him to interrupt the process of building the word, since all descendant nodes guarantee the creation of the word will finish. For clarification, we do not add the node to F^{n+1} if:
- The letter that marks the regarded node does not match the letter on position $p - 1$ in α,
- The node belongs to Adam and at least one edge exits to a node that is not in set F.

After we've found all correct predecessors of every node in F^n, we repeat the algorithm for the next set F^{n+1} and $p = p - 1$, until one of this condition is true:
- F^n is empty,
- $p == 0$.

If the algorithm stops because the set of vertices was empty, it means that there are no nodes in game G from which Eve can guarantee to construct the word α. That means that Adam has a winning strategy from all positions in such game. If the algorithm stops because we reached the start of the word, the final set of vertices F^{n+1} is our answer, which we'll use in the next part of the algorithm.

After the search algorithm is done, we end up with a set of vertices that are winning for Eve: we've just proven that Eve can force the word occurrence from those nodes onward. Now the game is simply a reaching game for Eve, and a safety game for Adam, where the objective for Eve is to reach one of the vertices from previously calculated set. We can easily calculate the winning region of Eve by gradually calculating the attractor to the desired set. The vertices that do not end up in the attractor are then considered Adam's winning vertices.

This algorithm's cost is around $O(e * l)$, where e is the number of edges in graph and l is the length of the word α. For each letter in word α in the worst case scenario we must check every exiting edge of every node in our graph in order to calculate the predecessors of all nodes in F. The part where we calculate the winning region in reachability game can be ignored, since it is calculated in $O(e)$ time.

Task a

We know from the tutorials that if Eve has a winning strategy in a reachability game, it is also a positional strategy. Since we reduced the sub-word game to a simple reachability game, Eve indeed has a positional strategy whenever she has a winning one.

Task b

Same goes for Adam, as the safety game is a dual game to reachability game. We've also seen in the tutorials that if Adam has a winning strategy in a safety game, it is also a positional one.