Universität Leipzig
Institut für Informatik
Winter 2026/2027

Prof. Dr. Karin Quaas

Exercises

Games in Automata and Logic

Solutions for Exercise Sheet 3 (for 2.11.2026)

Exercise 3.1

Prove or disprove the truth of the following FO formulas.

- $\exists g \forall v\, \varphi(g,v) \;\rightarrow\; \forall v \exists g\, \varphi(g,v)$
- $\forall v \exists g\, \varphi(g,v) \;\rightarrow\; \exists g \forall v\, \varphi(g,v)$

SOLUTION:

- $\exists g \forall v \varphi(g,v) \;\rightarrow\; \forall v \exists g \varphi(g,v)$ is true. For the proof, we show the truth of the equivalent formula $\exists g \forall v \varphi(g,v) \;\rightarrow\; \forall v' \exists g' \varphi(g',v')$. Suppose $\exists g \forall v \varphi(g,v)$ is true. Let v' be arbitrary. Let g' be g. Then $\varphi(v',g')$ holds by assumption.
- $\forall v \exists g \varphi(g,v) \;\rightarrow\; \exists g \forall v \varphi(g,v)$ is not true. We give a counterexample. Suppose $\varphi(x,y)$ expresses that x is the successor of y (x, y range over the natural numbers). Then $\forall v \exists g \varphi(g,v)$ expresses that for all numbers v there exists a successor g (which is true). However, it is not true that there exists a number g that is successor of every number v.

Exercise 3.2

Let $A = (V, V_A, V_E, E)$ and let $T \subseteq V$. Consider the algorithm for computing the attractor $\mathrm{Attr}_E(T)$, presented in Lecture 3:

$$\mathrm{Attr}_E(T) := \bigcup_{i \geq 0} \mathrm{Attr}_E^i(T),$$

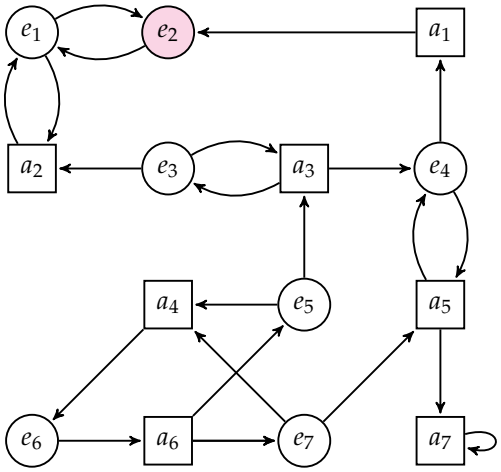
where

$$\begin{aligned} \mathrm{Attr}_E^0(T) &:= T \\ \mathrm{Attr}_E^{i+1}(T) &:= \mathrm{Attr}_E^i(T) \cup \mathrm{Force}_E(\mathrm{Attr}_E^i(T)) \end{aligned}$$

and

$$\begin{aligned} \mathrm{Force}_E(U) \;:=\; & \{v \in V_E \mid (v,u) \in E \text{ for some } u \in U\} \cup \\ & \{v \in V_A \mid (v,u) \in E \text{ implies } u \in U\} \end{aligned}$$

for every $U \subseteq V$. A naïve implementation of the attractor algorithm repeatedly scans all vertices until a fixed point is reached, yielding a quadratic running time. Show how to implement the algorithm in linear time. For explanation purposes, you may apply your linear time algorithm to the arena depicted on page 2, with $T = \{e_2\}$.



SOLUTION: The main difficulty is the computation of $\text{Force}_\text{E}(U)$: for every $u \in U$, we need to compute the set of prececessors (denoted by $Pred(u)$). And for the predecessors v of u that are in V_A we also need to compute the set of successors (denoted by $Succ(v)$), because we only add such a predecessor to $\text{Force}_\text{E}(U)$ if *all* its successors are already in U. So we actually want to avoid computing successors and predecessors more than once. The idea is to have a counter function c which stores for every vertex in V_A how many successor nodes are not yet in Attr. Initially, we set $c(v)$ to the number of successors of v. If a vertex v is a predecessor of a vertex in the current attractor set, then we decrement its counter value $c(v)$ (because a successor of v is in the attractor). If $c(v) = 0$, we can add v to the attractor set.

```
1:  A ← T
2:  initialize queue Q with all vertices in T
3:  for all v ∈ V_A do
4:      c(v) ← |Succ(v)|
5:  end for
6:  while Q is not empty do
7:      remove u from Q
8:      for all v ∈ Pred(u) do
9:          if v ∉ A then
10:             if v ∈ V_E then
11:                 add v to A
12:                 insert v into Q
13:             else
14:                 c(v) ← c(v) − 1
15:                 if c(v) = 0 then
16:                     add v to A
17:                     insert v into Q
18:                 end if
19:             end if
```

20: **end if**
21: **end for**
22: **end while**
23: **return** A

Initialization: $A = \{e_2\}$, $Q = \{e_2\}$, $c(a_1, a_2, a_3, a_4, a_5, a_6, a_7) = (1, 1, 2, 1, 2, 2, 1)$

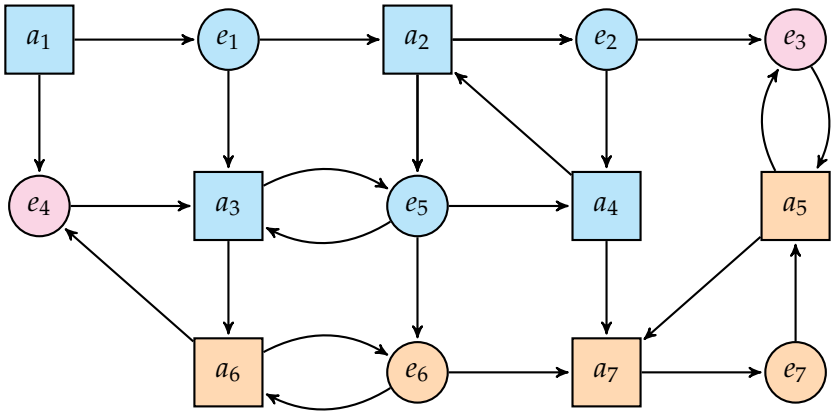
current v	$Pred(v)$	Q	c	A
		e_2	$(1,1,2,1,2,2,1)$	$\{e_2\}$
e_2	e_1, a_1	$e_1 a_1$	$(0,1,2,1,2,2,1)$	$\{e_2, e_1, a_1\}$
e_1	a_2, e_2	$a_1 a_2$	$(0,0,2,1,2,2,1)$	$\{e_2, e_1, a_1, a_2\}$
a_1	e_4	$a_2 e_4$	$(0,0,2,1,2,2,1)$	$\{e_2, e_1, a_1, a_2, e_4\}$
a_2	e_1, e_3	$e_4 e_3$	$(0,0,2,1,2,2,1)$	$\{e_2, e_1, a_1, a_2, e_4, e_3\}$
e_4	a_3, a_5	e_3	$(0,0,1,1,1,2,1)$	$\{e_2, e_1, a_1, a_2, e_4, e_3\}$
e_3	a_3	a_3	$(0,0,0,1,1,2,1)$	$\{e_2, e_1, a_1, a_2, e_4, e_3, a_3\}$
a_3	e_5	e_5	$(0,0,0,1,1,2,1)$	$\{e_2, e_1, a_1, a_2, e_4, e_3, a_3, e_5\}$
a_5	a_6		$(0,0,0,1,1,1,1)$	$\{e_2, e_1, a_1, a_2, e_4, e_3, a_3, e_5\}$

The queue only holds vertices that are definitely in the attractor and that may contribute to other attractors (their predecessors). If a current attractor vertex has an Eve-predecessor v, we can immediately add v to the attractor and to the queue (for checking its predecessors in one of the next iterations). If a current attractor vertex has an A-predecessor v, we actually need to check whether already all other sucessors of v are already in the attractor. This information is stored in c. The invariant for c is: $c(v)$ holds the current number of successors of v not yet in attractor. If it is 0, we can add v to the attractor and to the queue (for checking its predecessors in one of the next iterations). Otherwise we do not add it to neither the attractor nor the queue. If v actually belongs to the attractor, there will be a future iteration, where one of its other predecessors is the current vertex and finally $c(v)$ is equal to 0 (because evertyime a predecessor of v is considered, $c(v)$ is decremented).

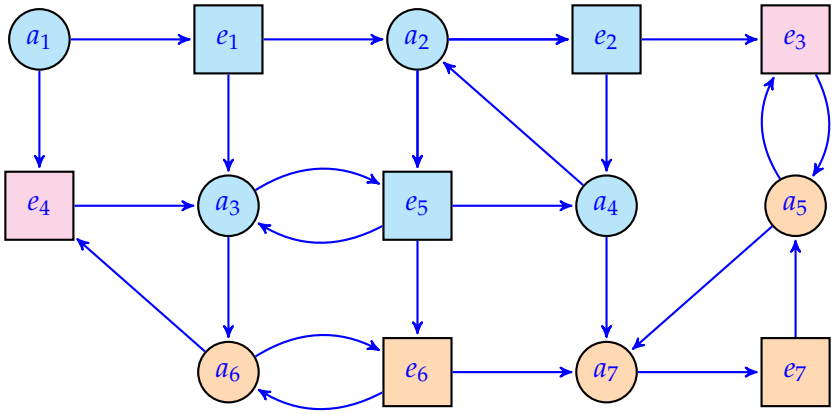
Note: we only consider each vertex at most one time, we only compute its predecessor sets at most one time, and we only compute the successor set of each Alice-vertex at most one time. This algorithm runs in linear time because it never asks the same question twice.

Exercise 3.3

Consider the following arena. Eve wins if she can force a play to staying in the set of blue vertices, that is, $\{a_1, e_1, a_2, e_2, a_3, e_5, a_4\}$. From which vertices does Eve have a winning strategy?



SOLUTION: This is a safety game with $T = \{a_1, e_1, a_2, e_2, a_3, e_5, a_4\}$. We exploit that safety games are dual to reachability games. Let $G = (A, \text{Safety}(A, T))$ with $A = (V, V_A, V_E, E)$ as depicted. Define a reachability game by $G' = (A', \text{Reach}(V \setminus T))$ with $A' = (V, V_E, V_A, E)$ with the ownership of the vertices flipped. Hence the arena A' looks as follows:



We compute the winning region $W_E(G')$ according to Lecture 3, for $T' = \{e_3, e_4, a_5, a_6, e_6, a_7, e_7\}$.

- $\text{Attr}_E^0(T') = T'$
- $\text{Attr}_E^1(T') = \text{Attr}_E^0(T') \cup \{a_1, a_3, a_4\}$
- $\text{Attr}_E^2(T') = \text{Attr}_E^1(T') \cup \{e_2, e_5\}$
- $\text{Attr}_E^3(T') = \text{Attr}_E^2(T') \cup \{a_2\}$
- $\text{Attr}_E^4(T') = \text{Attr}_E^3(T') \cup \{e_1\} = V$

Hence $W_E(G') = V$, $W_A(G') = \emptyset$, and winning strategy g' for Eve. Then we set $W_E(G) = W_A(G') = \emptyset$. Hence Eve cannot win this game, no matter where the play starts from.