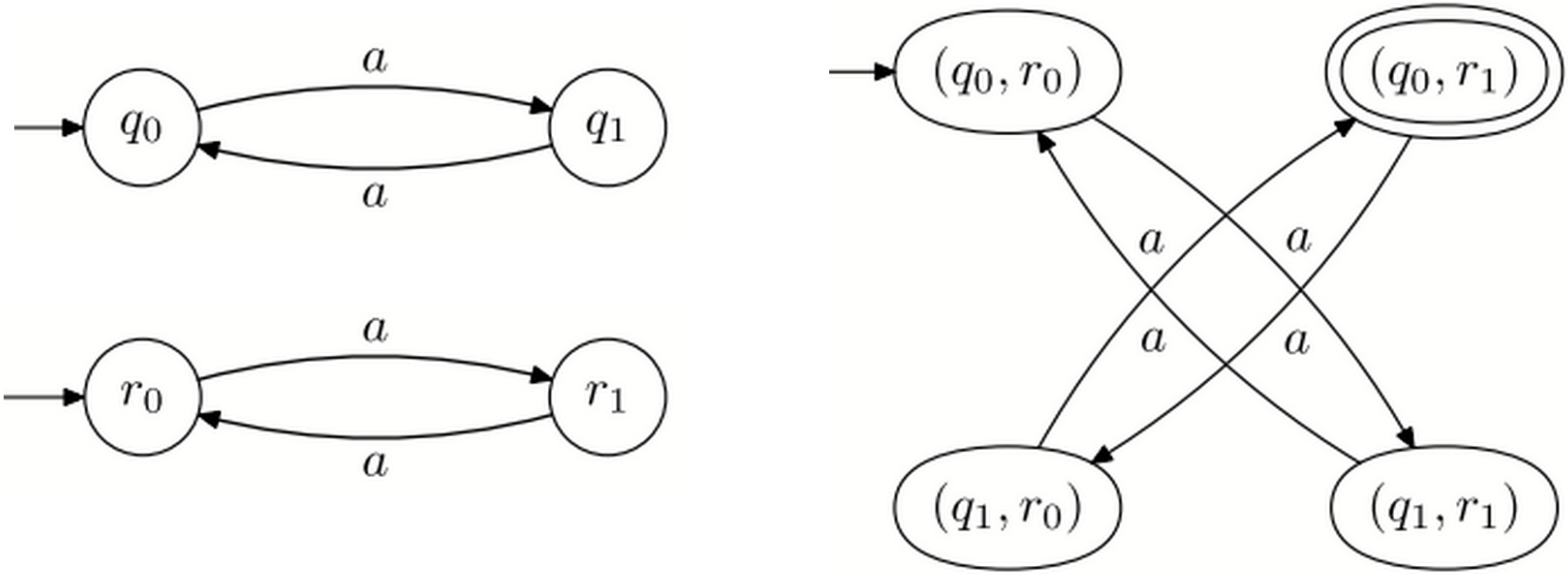
Implementing boolean operations

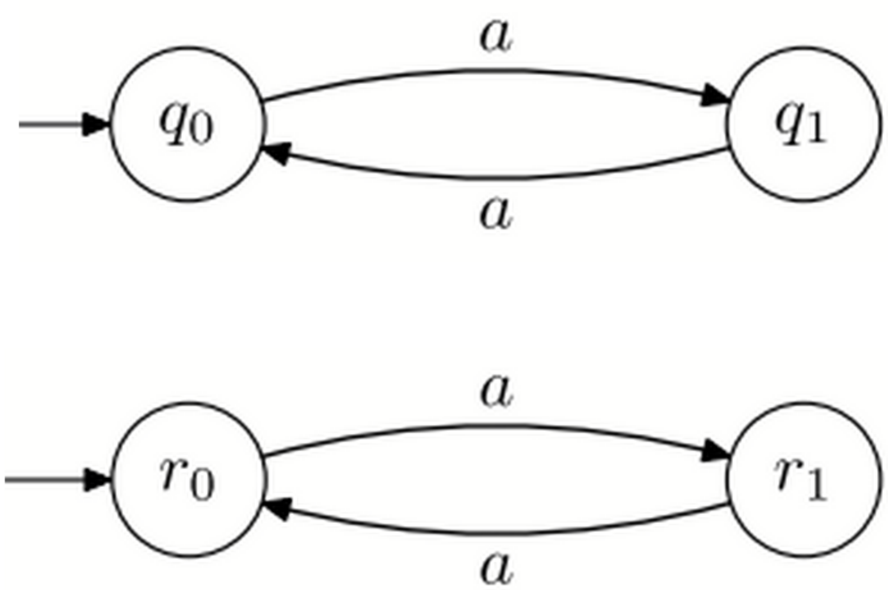
Intersection

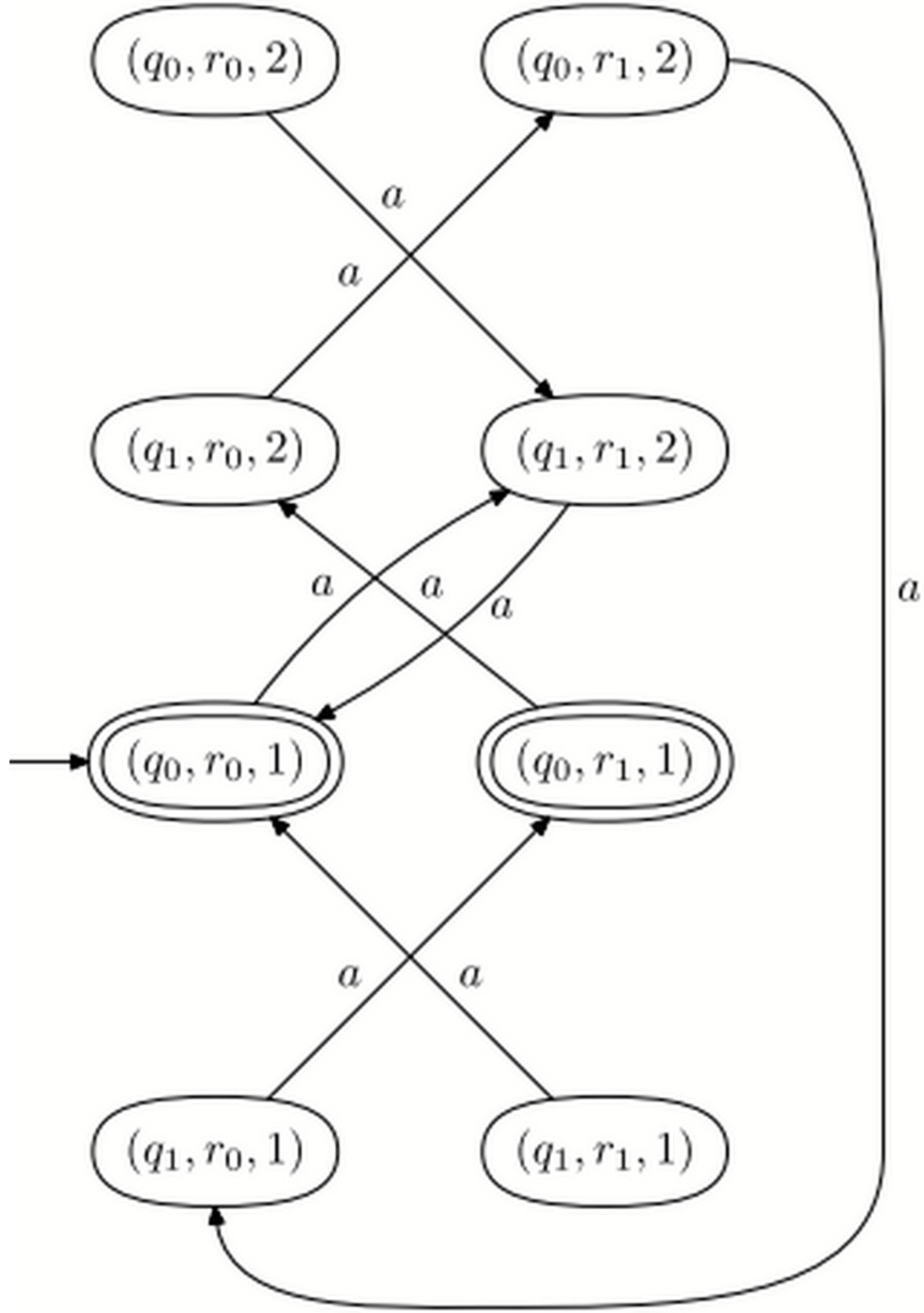
The algorithm for NFAs does not work ...



Apply the same idea as in the conversion NGA => NBA

- Take two copies of the pairing A1 X A2
- Redirect the arcs leaving F1 in the first copy to the second copy
- Redirect the arcs leaving F2 in the second copy to the first copy
- The final states are the F1-states of the first copy


a
q_0
q_1
a
a
r_0
r_1
a


$(q_0, r_0, 2)$
$(q_0, r_1, 2)$
$(q_1, r_0, 2)$
$(q_1, r_1, 2)$
$(q_0, r_0, 1)$
$(q_0, r_1, 1)$
$(q_1, r_0, 1)$
$(q_1, r_1, 1)$
a

IntersNBA(A_1, A_2)
Input: NBAs $A_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$, $A_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$
Output: NBA $A_1 \cap A_2 = (Q, \Sigma, \delta, q_0, F)$ with $\mathcal{L}_\omega(A_1 \cap^\omega A_2) = \mathcal{L}_\omega(A_1) \cap \mathcal{L}_\omega(A_2)$

$Q, \delta, F \leftarrow \emptyset$
$q_0 \leftarrow [q_{01}, q_{02}, 1]$
$W \leftarrow \{\ [q_{01}, q_{02}, 1]\ \}$
while $W \neq \emptyset$ **do**
  pick $[q_1.q_2, i]$ **from** W
  add $[q_1, q_2, i]$ **to** Q'
  if $q_1 \in F_1$ **and** $i = 1$ **then add** $[q_1, q_2, 1]$
    for all $a \in \Sigma$ **do**
      for all $q_1' \in \delta_1(q_1, a), q_2' \in \delta(q_2, a)$ **do**
        if $i = 1$ **and** $q_1' \notin F_1$ **then**
          add $([q_1, q_2, 1], a, [q_1', q_2', 1])$ **to** δ
          if $[q_1', q_2', 1] \notin Q'$ **then add** $[q_1', q_2', 1]$ **to** W
        if $i = 1$ **and** $q_1' \in F_1$ **then**
          add $([q_1, q_2, 1], a, [q_1', q_2', 2])$ **to** δ
          if $[q_1', q_2', 2] \notin Q'$ **then add** $[q_1', q_2', 2]$ **to** W
        if $i = 2$ **and** $q_2' \notin F_2$ **then**
          add $([q_1, q_2, 2], a, [q_1', q_2', 2])$ **to** δ
          if $[q_1', q_2', 2] \notin Q'$ **then add** $[q_1', q_2', 2]$ **to** W
        if $i = 2$ **and** $q_2' \in F_2$ **then**
          add $([q_1, q_2, 2], a, [q_1', q_2', 1])$ **to** δ
          if $[q_1', q_2', 1] \notin Q'$ **then add** $[q_1', q_2', 1]$ **to** W
  return $(Q, \Sigma, \delta, q_0, F)$

Special cases / Improvements

- All states of at least one of A1 and A2 are accepting

  In this case we do not need the second copy of
  A1 X A2. The algorithm for NFAs works.

- Intersection of NBAs A1, A2, ..., Ak:

  Do NOT apply the algorithm for two NBAs (k-1) times.
  Proceed instead as in NGA => NBA
  (k n1 ... nk states instead of exponential number in k)

Complement

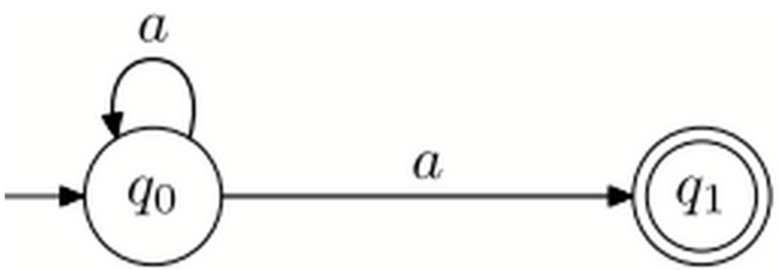
Main result proved by Büchi: NBAs are closed under complement.

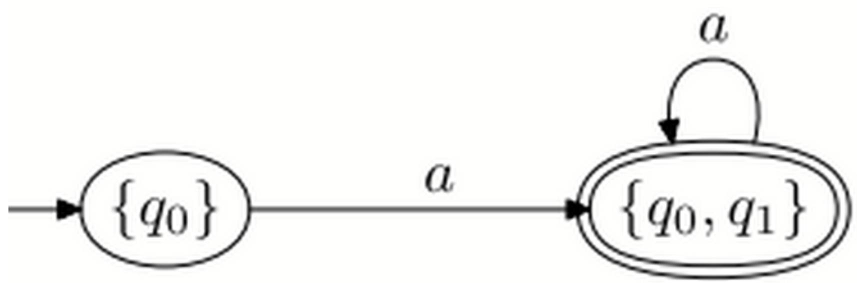
Many later improvements in recent years.

Construction radically different from the one for NFAs.

Problems

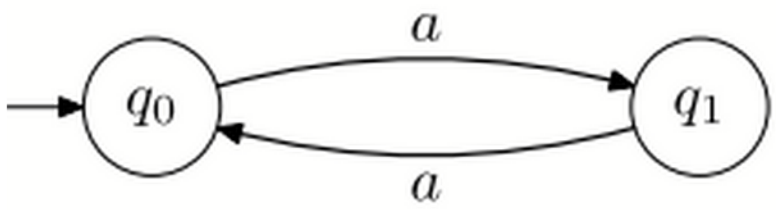
- The powerset construction does not work

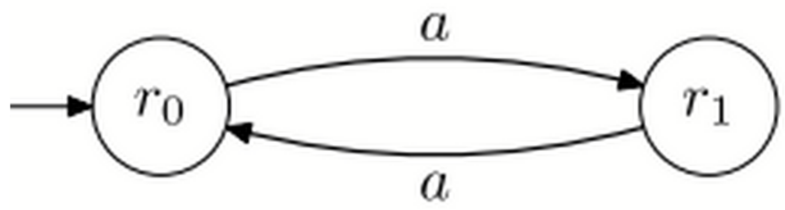




- No other determinization construction works

- Exchanging accepting and non-accepting states in DBAs also fails



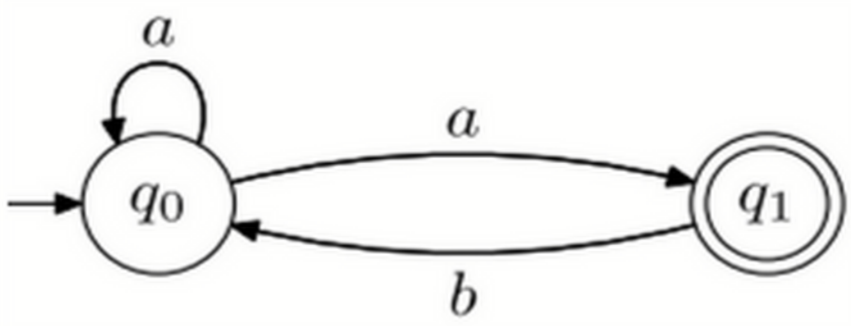


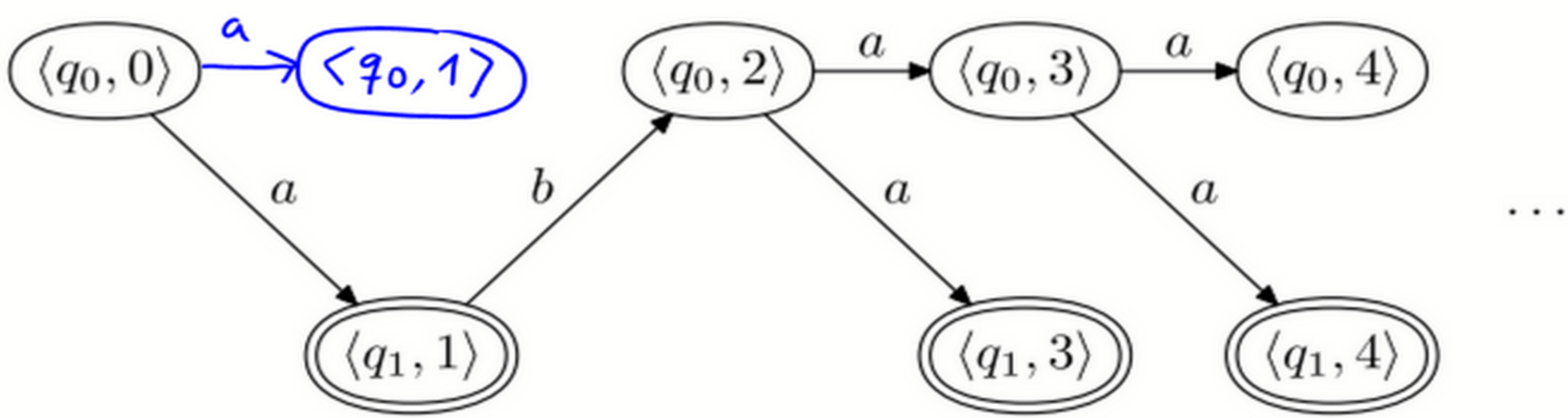
First informal ideas

Let NBA A with n states. We search for a complement automaton C(A).

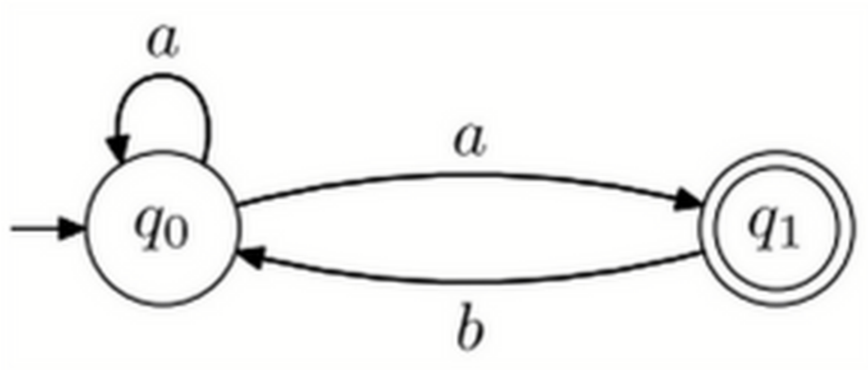
If A does not accept a word w, then no run of A on w is accepting. C(A) must "get this information" from at least one one of its runs on w , so that it can accept.

We first have a closer look at why the powerset construction does not work.

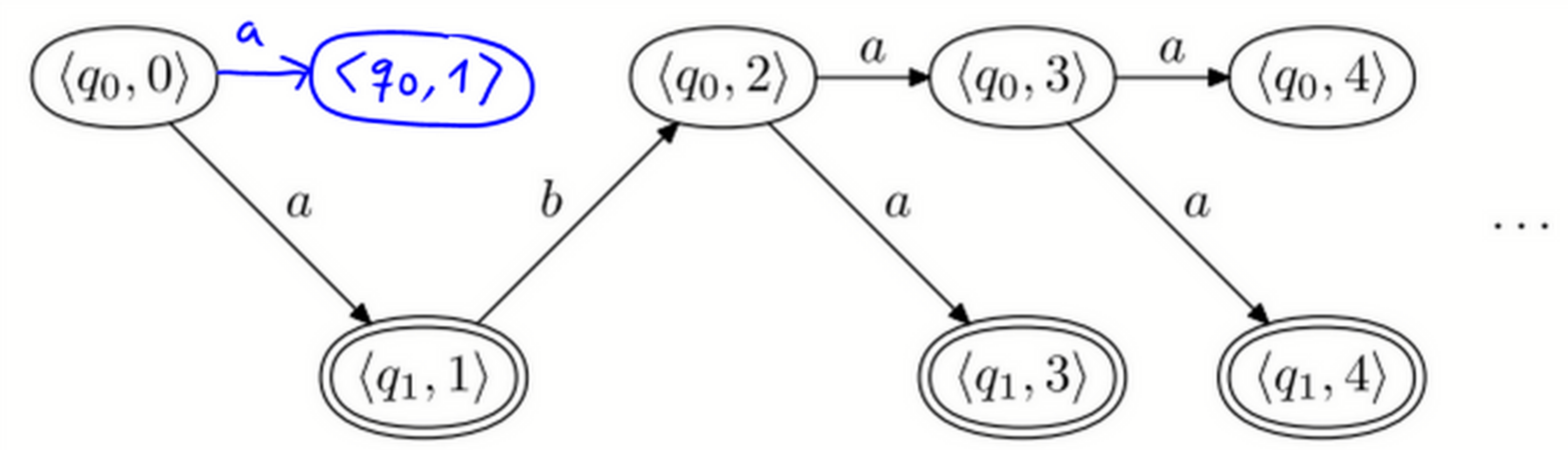

a
a
q_0
q_1
b


$\langle q_0, 0\rangle$
a
$\langle q_0, 1\rangle$
$\langle q_0, 2\rangle$
a
$\langle q_0, 3\rangle$
a
$\langle q_0, 4\rangle$
a
b
a
a
…
$\langle q_1, 1\rangle$
$\langle q_1, 3\rangle$
$\langle q_1, 4\rangle$

dag(abaa …)



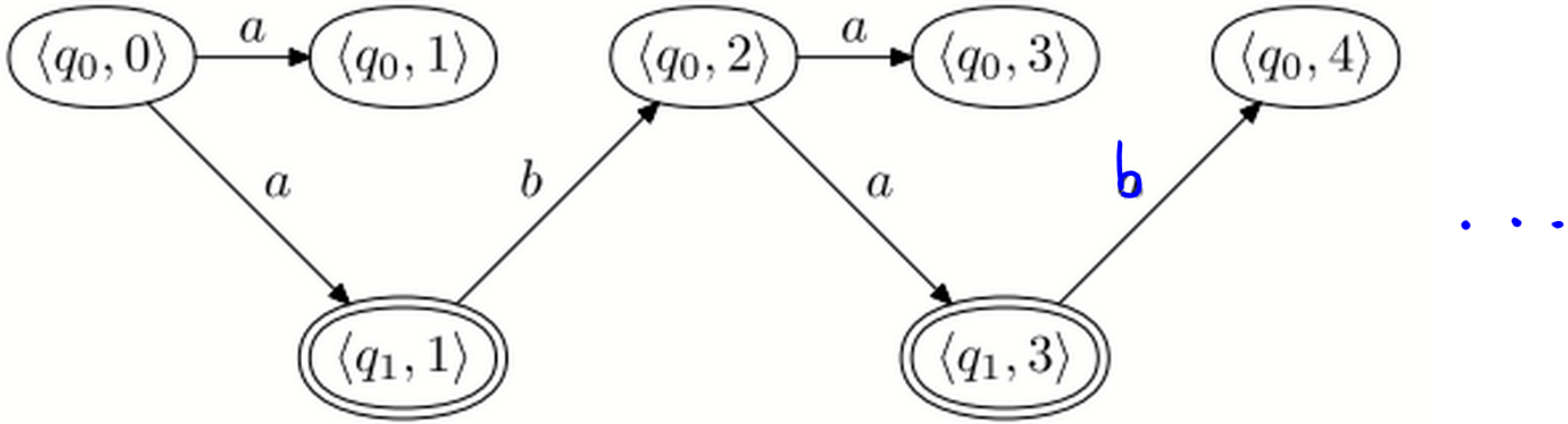
a
a
q_0
q_1
b



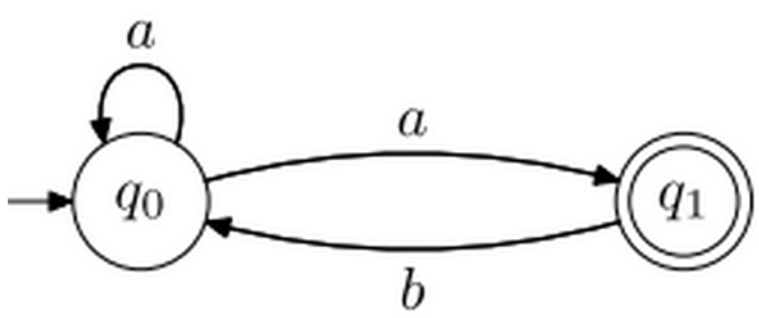
$\langle q_0, 0\rangle$
a
$\langle q_0, 1\rangle$
$\langle q_0, 2\rangle$
a
$\langle q_0, 3\rangle$
a
$\langle q_0, 4\rangle$
a
b
a
a
$\langle q_1, 1\rangle$
$\langle q_1, 3\rangle$
$\langle q_1, 4\rangle$
...

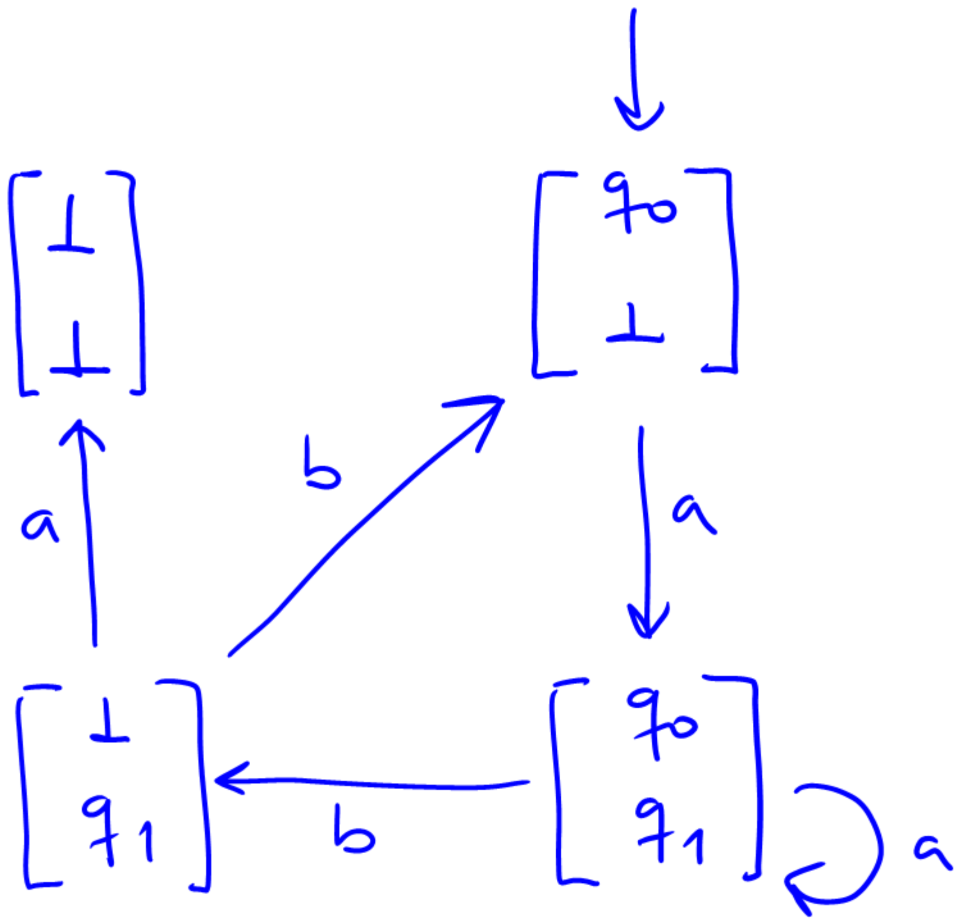
dag(abaa ...)

Slicing a dag



$$\begin{bmatrix} q_0 \\ \\ \\ \perp \end{bmatrix} \xrightarrow{a} \begin{bmatrix} q_0 \\ \\ \\ q_1 \end{bmatrix} \xrightarrow{b} \begin{bmatrix} q_0 \\ \\ \\ \perp \end{bmatrix} \xrightarrow{a} \begin{bmatrix} q_0 \\ \\ \\ q_1 \end{bmatrix} \xrightarrow{b} \begin{bmatrix} q_0 \\ \\ \\ \perp \end{bmatrix} \cdots$$


a
a
q_0
q_1
b


$\perp$
$\perp$
q_0
$\perp$
b
a
a
$\perp$
q_1
b
q_0
q_1
a

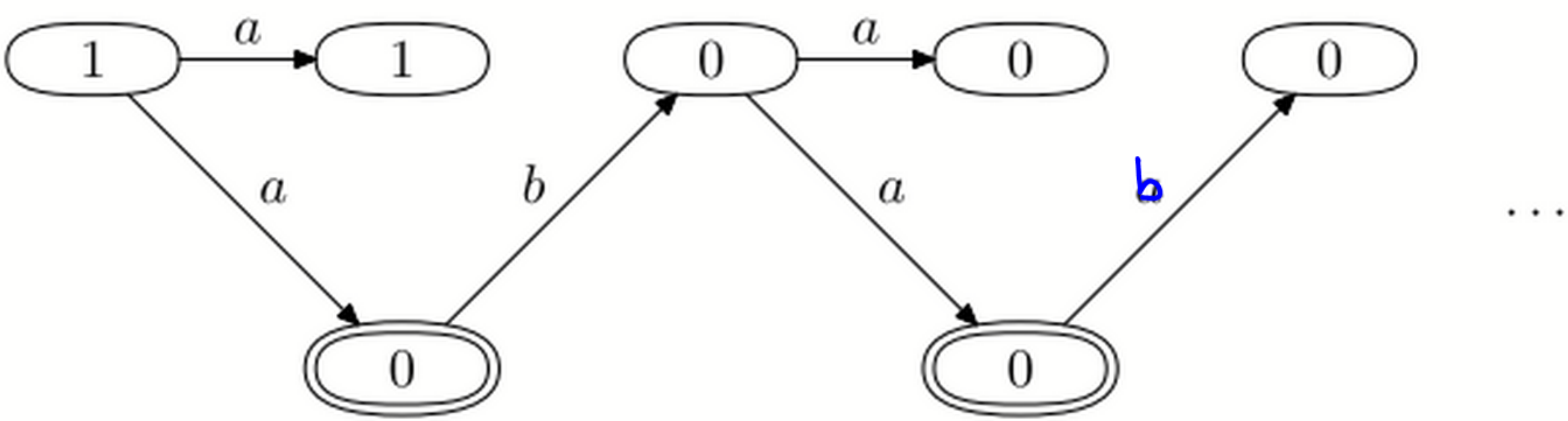
The powerset construction only retains information about which states are visited, but no information about the connections between them.

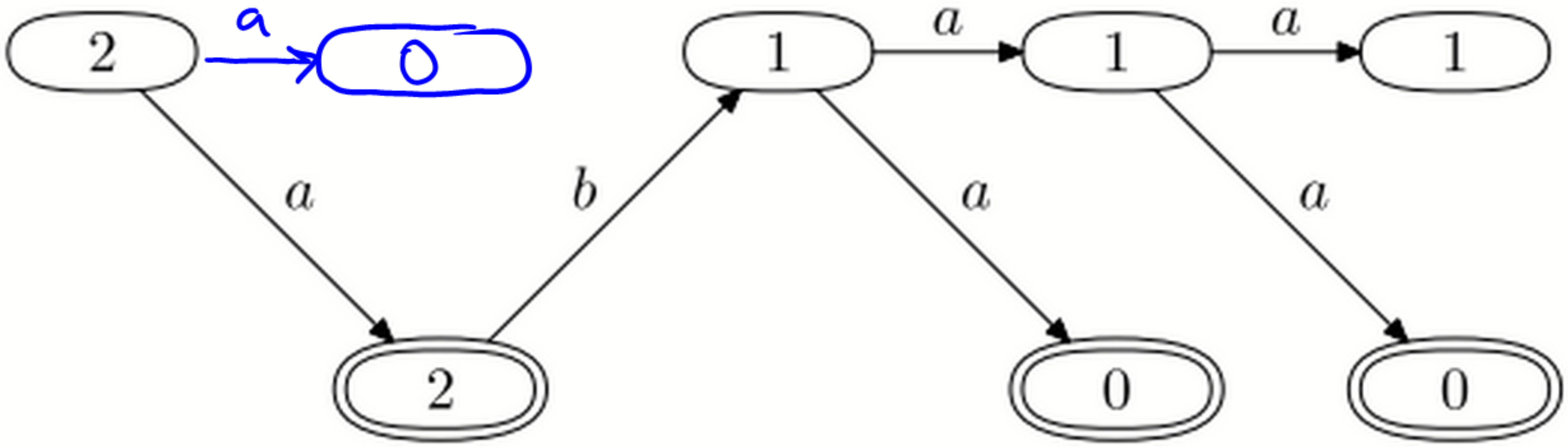
So we have to add information to the states …

Rankings

A ranking of dag(w) is a function that assigns to each node of dag(w) a rank: a number in the range [0 ... 2n] satisfying two conditions:

- ranks never increase along paths, and
- ranks of accepting states are even





The ranks along an infinite path of dag(w) stabilize at an stable rank.
Observe: if the stable rank of a path is odd, then the path cannot visit accepting states infinitely often.
So: if all paths have odd stable ranks, the word is not accepted.
If all paths have odd stable ranks, we say the ranking is

If dag(w) admits an odd ranking, then A does not accept w.

Imagine we can prove: if A does not accept w, then dag(w) admits an odd ranking. Then:

design C(A) so that it accepts w if and only if dag(w) admits an odd ranking.

A does not accept w => dag(w) admits an odd ranking

Assume A does not accept w. We construct an odd ranking for dag(w).

Procedure:

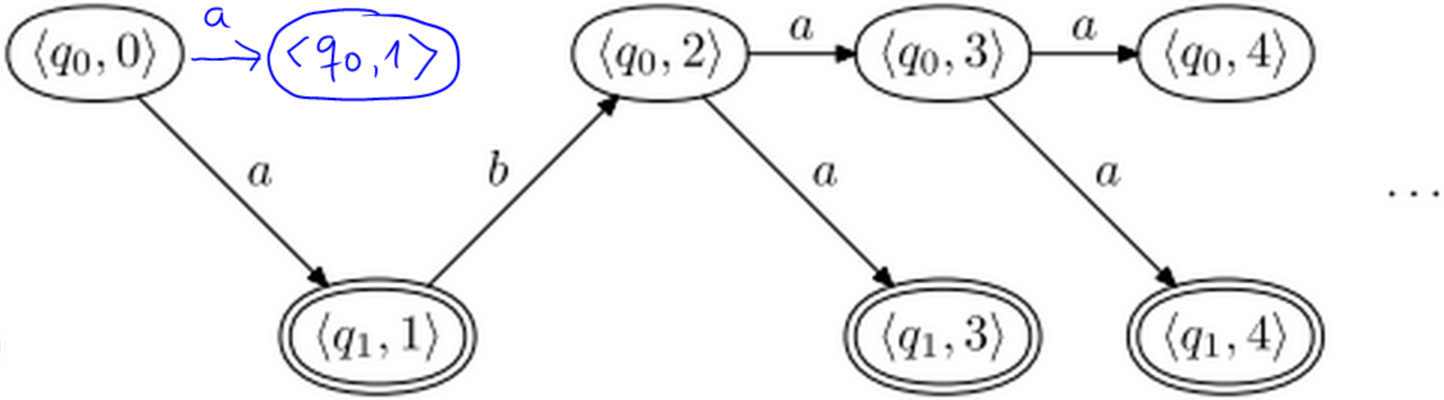
- we proceed in n rounds, each round with steps n.0 and n.1
- each step removes a set of nodes together with all its descendants
- the nodes removed at step i.j get rank 2i+j

Step i.0: remove all nodes having only finitely many successors.

Step i.1: remove nodes none of whose descendants (including themselves) is accepting

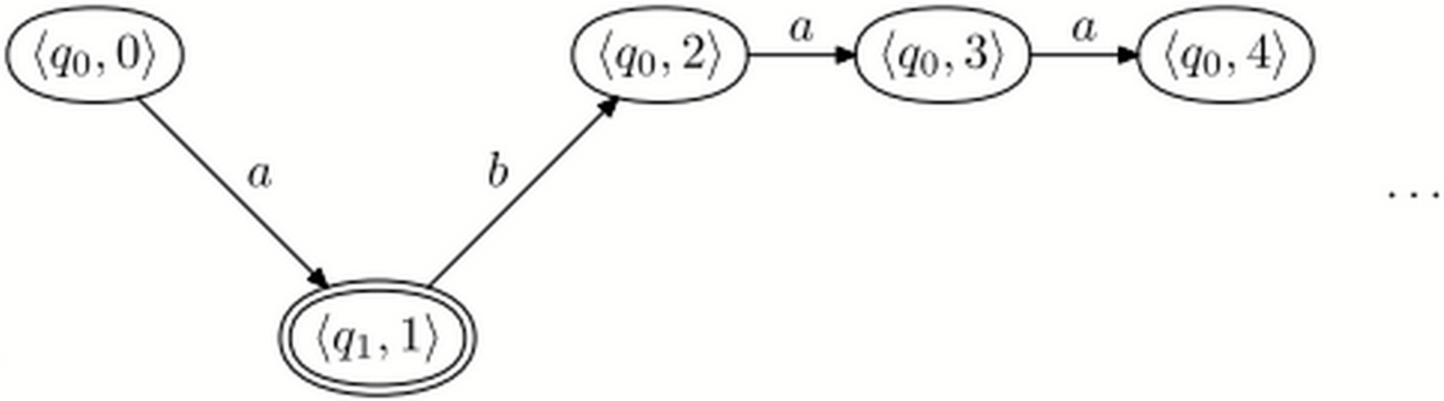
Observe: ranks along a path cannot increase
accepting states can only be removed at step i.0

Remains to prove: after n rounds there are no nodes left.



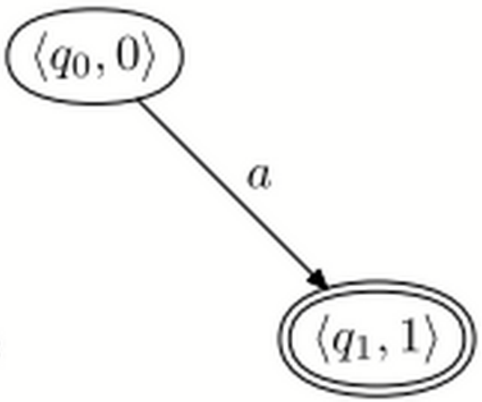
$\langle q_0, 0\rangle$
a
$\langle q_0, 1\rangle$
$\langle q_0, 2\rangle$
a
$\langle q_0, 3\rangle$
a
$\langle q_0, 4\rangle$
a
b
a
a
$\dots$
$\langle q_1, 1\rangle$
$\langle q_1, 3\rangle$
$\langle q_1, 4\rangle$

D_0



$\langle q_0, 0\rangle$
$\langle q_0, 2\rangle$
a
$\langle q_0, 3\rangle$
a
$\langle q_0, 4\rangle$
a
b
$\dots$
$\langle q_1, 1\rangle$

D_1



$\langle q_0, 0\rangle$
a
$\langle q_1, 1\rangle$

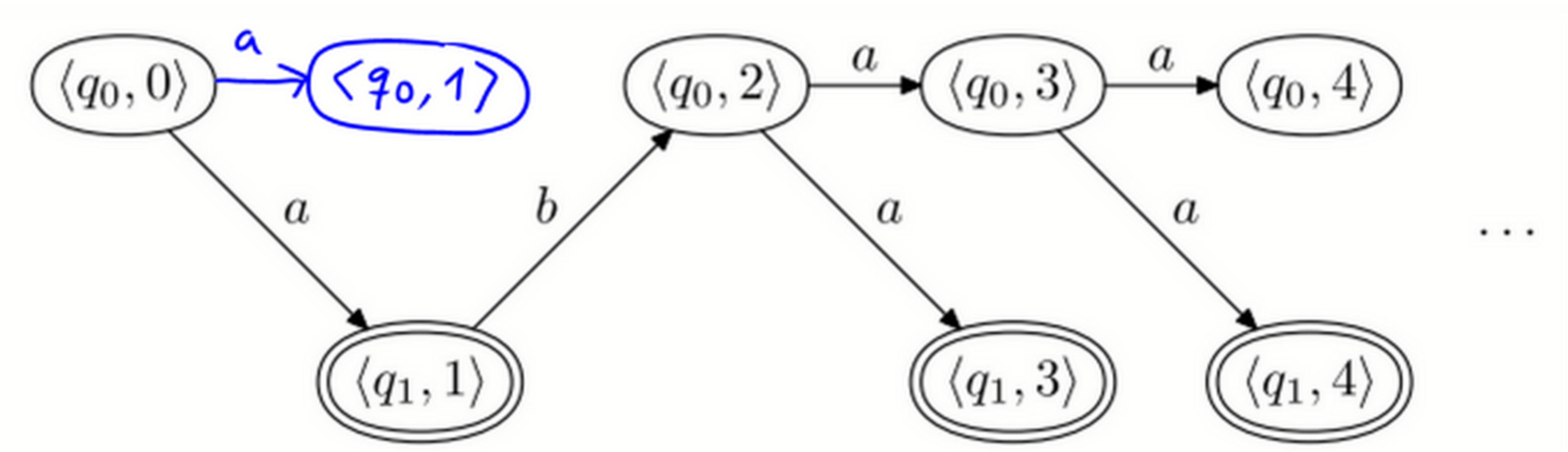
D_2

Observe:

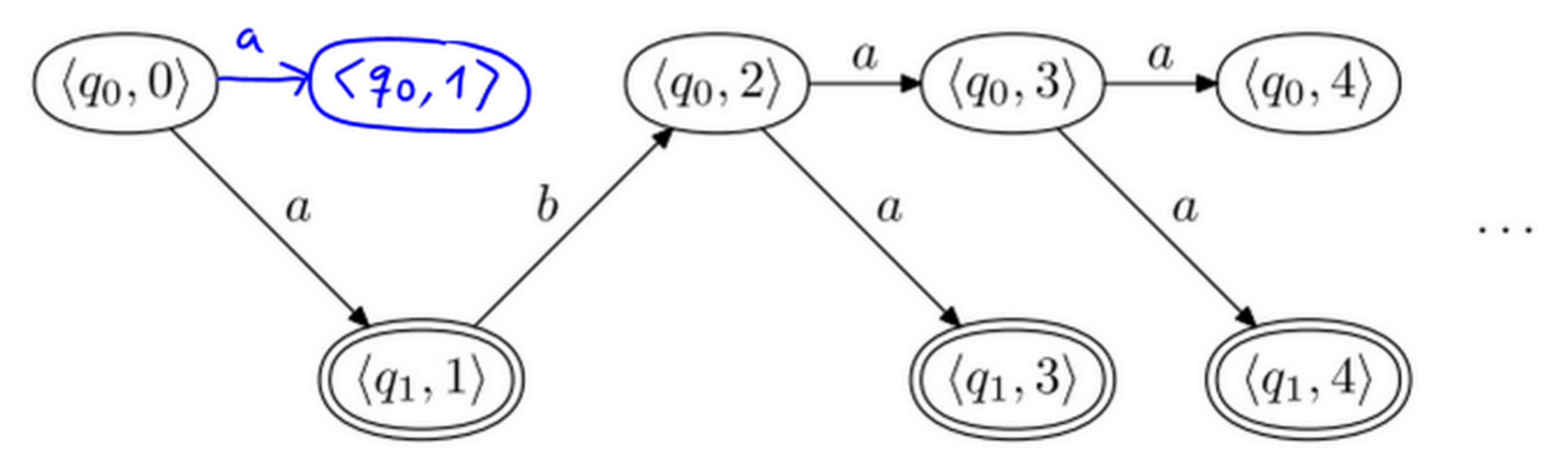
- ranks along a path cannot increase
- accepting states can only be removed at step i.0

Remains to prove: after n rounds there are no nodes left.

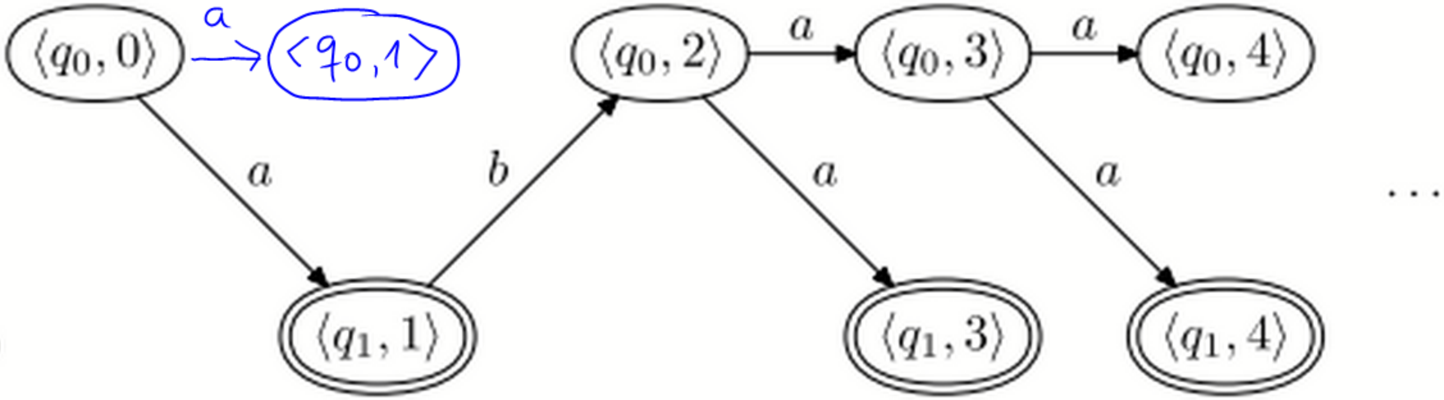
For this we first split the dag into "slices"



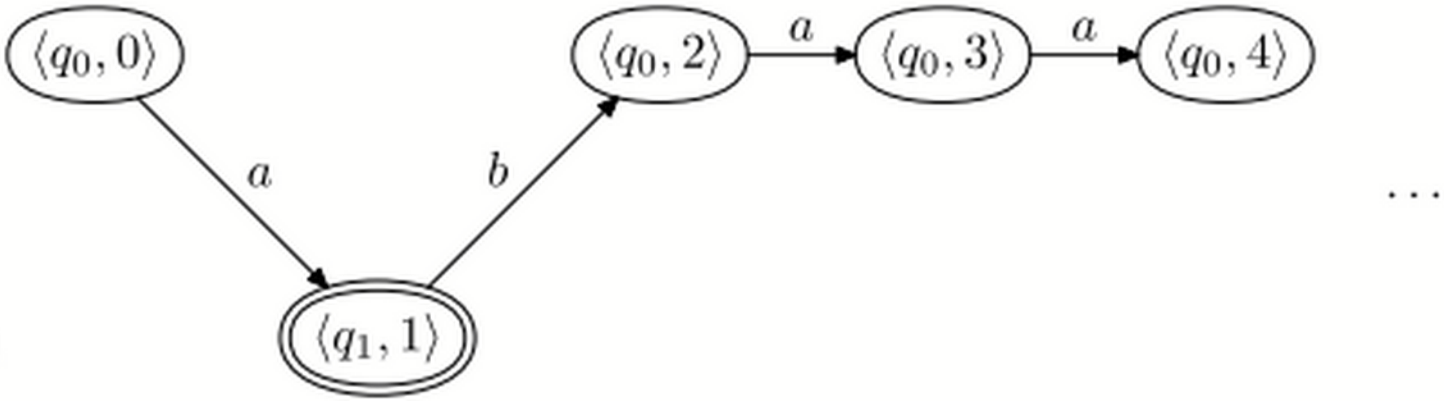
Each slice has a "width"



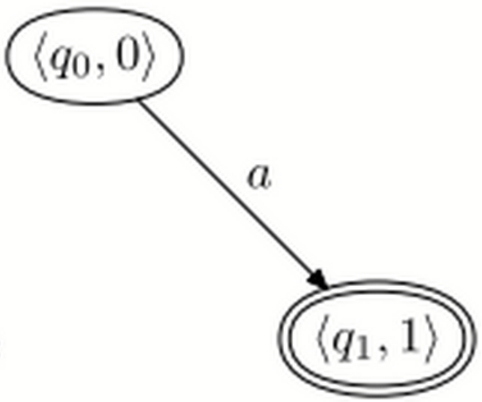
The "width" of the whole dag is defined as the largest width that appears infinitely often.


⟨q0,0⟩
a
⟨q0,1⟩
a
b
⟨q0,2⟩
a
⟨q0,3⟩
a
⟨q0,4⟩
a
a
...
⟨q1,1⟩
⟨q1,3⟩
⟨q1,4⟩

D_0


⟨q0,0⟩
⟨q0,2⟩
a
⟨q0,3⟩
a
⟨q0,4⟩
a
b
...
⟨q1,1⟩

D_1


⟨q0,0⟩
a
⟨q1,1⟩

D_2

Little lemma: Each round decreases the width of the dag by at least 1.

Since the initial width is at most n, it follows that there are at most n rounds.

So every node gets assigned a number between 0 and 2n.

Achieved so far:

1. We define a mapping *dag* which assigns to each word $w \in \Sigma^\omega$ a directed acyclic graph $dag(w)$. We also define an *odd ranking* of $dag(w)$ as a labelling of the nodes of $dag(w)$ by natural numbers satisfying certain properties.

2. We prove that w is rejected by A if and only if $dag(w)$ admits an odd ranking.

3. We construct an NBA $\overline{A}$ which accepts w if and only if $dag(w)$ admits an odd ranking.

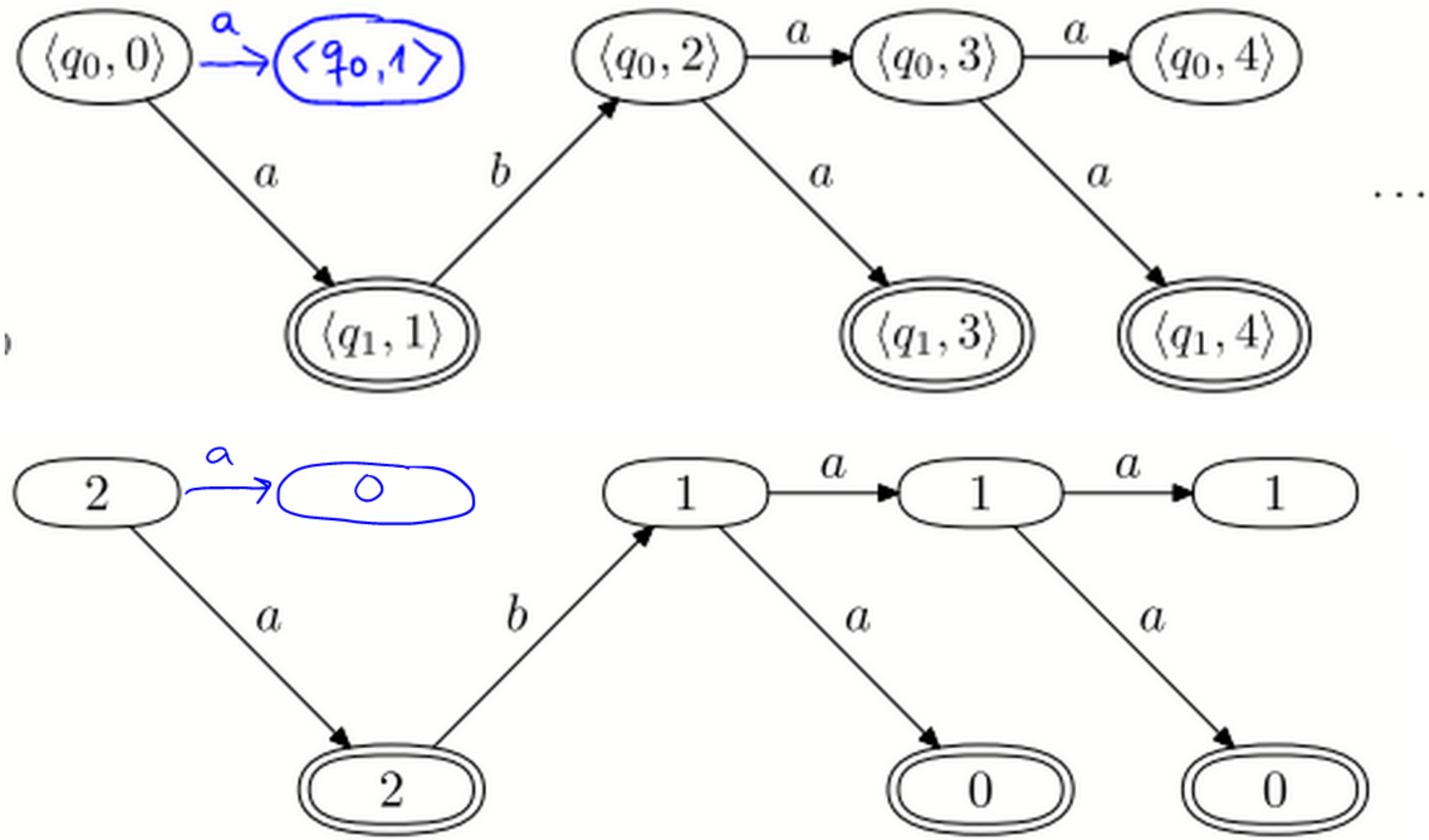
3. We construct an NBA $\overline{A}$ which accepts w if and only if $dag(w)$ admits an odd ranking.

First idea:

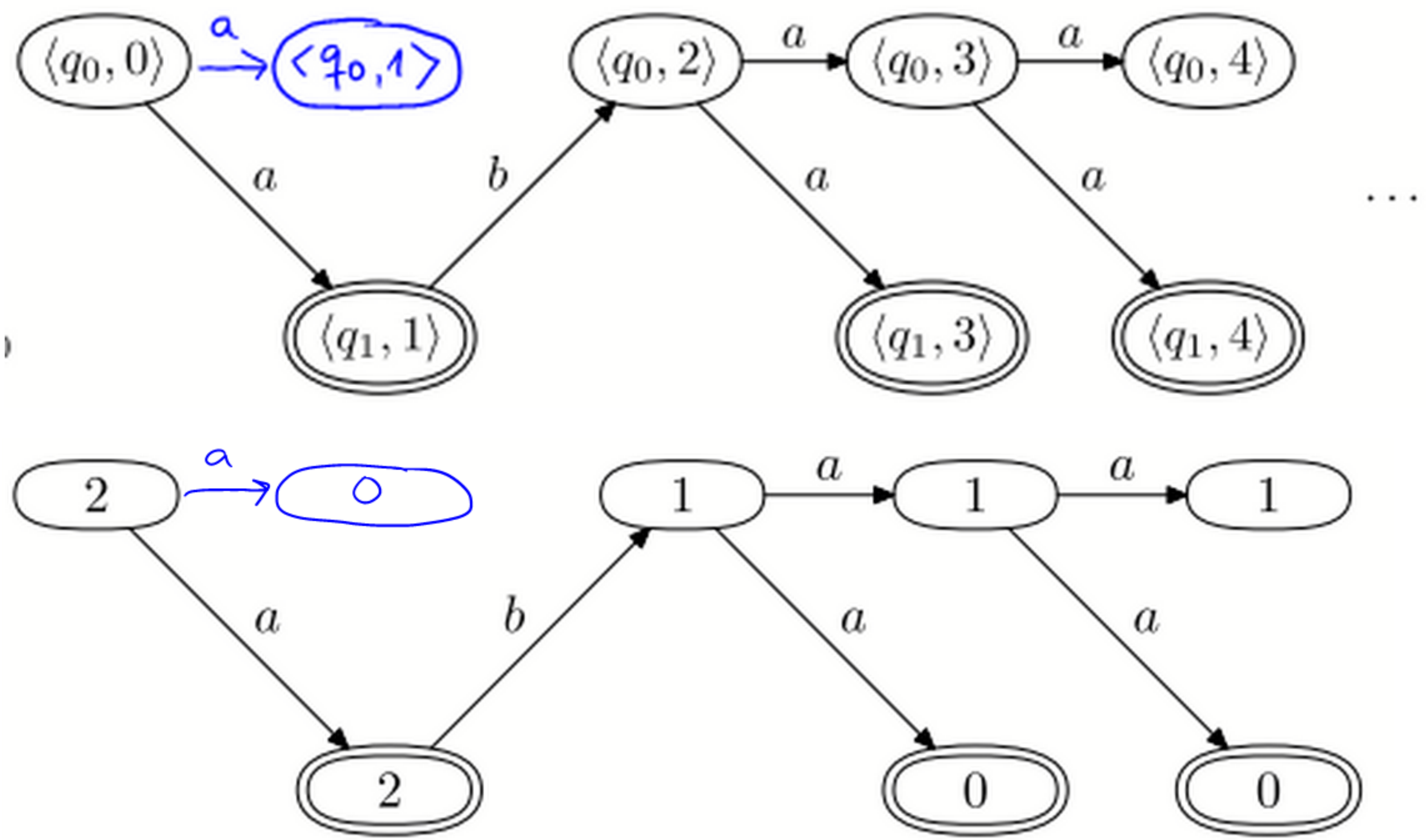
- Choose the states and transitions of C(A) so that the runs of C(A) on w correspond to the rankings of dag(w).

- Choose the accepting states so that the accepting runs correspond to the odd rankings.

Choosing the states and transitions



Choosing the states and transitions



$$\begin{bmatrix}2\\ \\ \perp\end{bmatrix} \xrightarrow{a} \begin{bmatrix}0\\ \\ 2\end{bmatrix} \xrightarrow{b} \begin{bmatrix}1\\ \\ \perp\end{bmatrix} \xrightarrow{a} \begin{bmatrix}1\\ \\ 0\end{bmatrix} \xrightarrow{a} \begin{bmatrix}1\\ \\ 0\end{bmatrix} \quad \ldots$$

States: level rankings, vectors over [0,…,2n] and

Transitions: there is a transition between two level rankings (states) if we can see them as two consecutive slices in a ranking.

- Textual notation (lecture notes): lr |--> lr'

Choosing the accepting states

Unfortunately, there is no way to characterize the odd ranks by means of a Büchi condition.

Solution: add more information to the states.

Recall: a ranking is odd if every infinite path contains infinitely many nodes of odd rank.

Checkpoint set: set of levels such that between any two consecutive levels every path visits an state of odd rank at least once.

A ranking is odd iff it has an infinite checkpoint set.

The additional information will allow C(A) to identify checkpoints

Add to each level ranking a new component: A set O of states.

A state q belongs to O if there is a path starting at the last checkpoint and ending at q that does not visit any states of odd rank.

Informally: a state of O "owes" is the endpoint of a path that "owes" a visit to nodes of odd rank.

State: level ranking + set of owing states

Transitions take care of suitably updating the owing set.

- The initial state is the pair $[lr_0, \{q_0\}]$, where $lr_0(q_0) = 2n$, and $lr_0(q) = \perp$ for every $q \neq q_0$. Observe that q_0 'owes" a visit to a node of odd rank.

When the set of owing states is nonempty, $\overline{A}$ updates it:

- If $O \neq \emptyset$, then $\langle lr', O' \rangle \in \overline{\delta}(\langle lr, O \rangle, a)$ iff $lr \overset{a}{\mapsto} lr'$ and $O' = \{q' \in \delta(O, a) \mid lr'(q') \text{ is even }\}$.

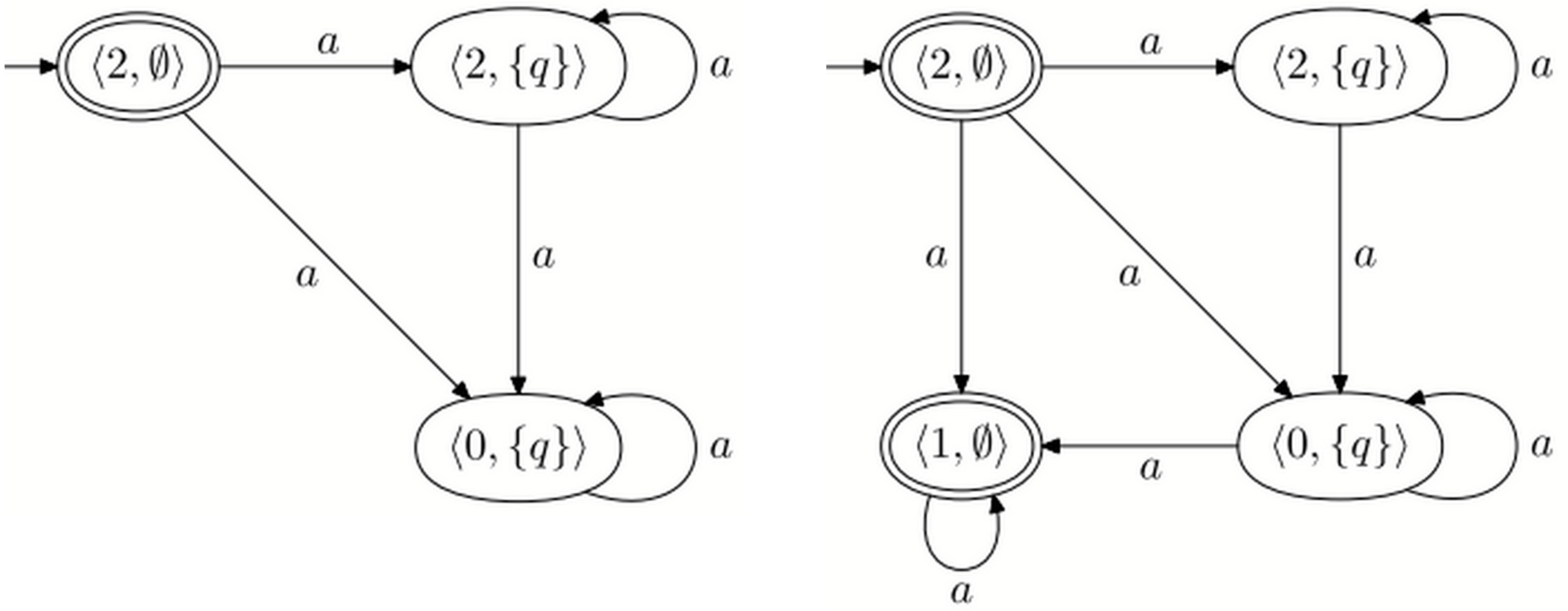
When the set of owing states is empty, $\overline{A}$ has reached a checkpoint, and it starts searching for the next one; all states of even rank are owing:

- If $O = \emptyset$, then $\langle lr', O' \rangle \in \overline{\delta}(\langle lr, O \rangle, a)$ iff $lr \overset{a}{\mapsto} lr'$ and $O' = \{q' \in Q \mid lr'(q') \text{ is even }\}$.

The accepting states are those at which a checkpoint is reached:

- a state $[lr, O]$ is accepting if $O = \emptyset$.

Example 12.3 We construct the complements $\overline{A}_1$ and $\overline{A}_2$ of the two one-state NBAs over the alphabet $\{a\}$: $A_1 = (\{q\}, \{a\}, \{q\}, \delta, \{q\})$ and $A_2 = (\{a\}, \{q\}, \{q\}, \delta, \emptyset)$, where $\delta(q, a) = \{q\}$. We have $\mathcal{L}_\omega(A_1) = a^\omega$ and $\mathcal{L}_\omega(A_2) = \emptyset$.



CompNBA(A)

Input: NBA $A = (Q, \Sigma, \delta, q_0, F)$

Output: NBA $\overline{A} = (\overline{Q}, \Sigma, \overline{\delta}, \overline{q}_0, \overline{F})$ with $\mathcal{L}_\omega(\overline{A}) = \overline{\mathcal{L}_\omega(A)}$

$\overline{Q}, \overline{\delta}, \overline{F} \leftarrow \emptyset$
$\overline{q}_0 \leftarrow [lr_0, \{q_0\}]$
$W \leftarrow \{\ [lr_0, \{q_0\}]\ \}$
while $W \neq \emptyset$ **do**
    pick $[lr, P]$ **from** W; **add** $[lr, P]$ **to** $\overline{Q}$
    if $P = \emptyset$ **then add** $[lr, P]$ **to** $\overline{F}$
    for all $a \in \Sigma, lr' \in \mathcal{R}$ such that $lr \overset{a}{\mapsto} lr'$ **do**
        if $P \neq \emptyset$ **then** $P' \leftarrow \{q \in \delta(P, a) \mid lr'(q) \text{ is even}\}$
        else $P' \leftarrow \{q \in Q \mid lr'(q) \text{ is even}\}$
        add $([lr, P], a, [lr', P'])$ **to** $\overline{\delta}$
        if $[lr', P'] \notin \overline{Q}$ **then add** $[lr', P']$ **to** W
return $(\overline{Q}, \Sigma, \overline{\delta}, \overline{q}_0, \overline{F})$

Size of C(A)

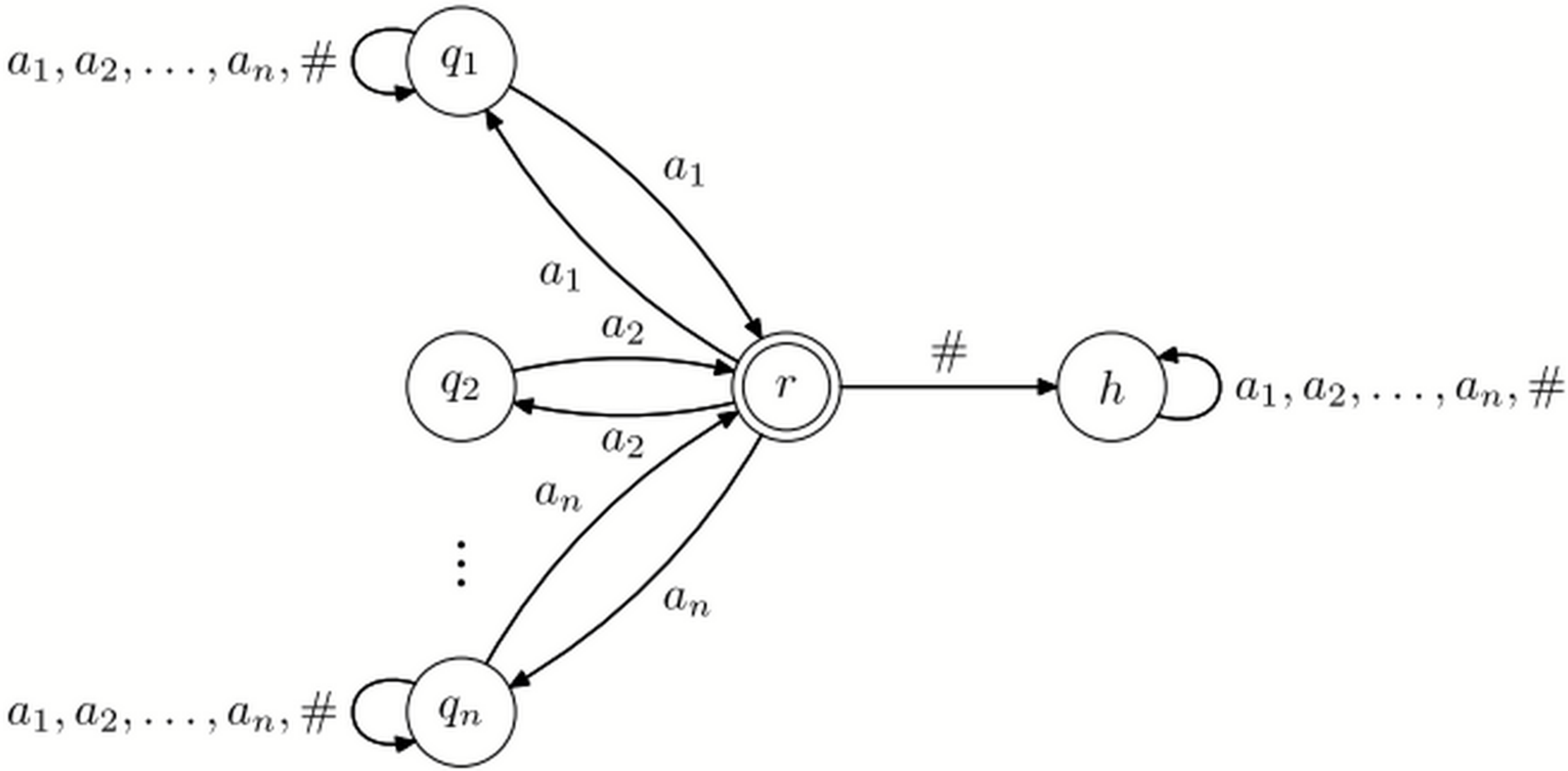
Assume A has n states.

Upper bound: number of level rankings is $(2n+2)^n$
number of owing sets is 2^n
So C(A) has at most $4^n (n+1)^n$ states
i.e., $2^{O(n \log n)}$ states.

Lower bound: $n!$ states, which is also $2^{O(n \log n)}$

Let $\Sigma_n = \{1, \ldots, n, \#\}$. We associate to a word $w \in \Sigma_n^\omega$ the following directed graph $G(w)$: the nodes of $G(w)$ are $1, \ldots, n$ and there is an edge from i to j if w contains infinitely many occurrences of the word ij. Define L_n as the language of infinite words $w \in A^\omega$ for which $G(w)$ has at least a cycle and define $\overline{L}_n$ as the complement of L_n.

We first show that for all $n \geq 1$, L_n is recognized by a Büchi automaton with $n + 2$ states. Let A_n be the automaton shown in Figure 12.10. We show that A_n accepts L_n.



Proposition 12.4 *For all $n \geq 1$, every NBA recognizing $\overline{L}_n$, has at least $n!$ states.*

Proof: We need some preliminaries. Given a permutation $\tau = \langle \tau(1), \ldots, \tau(n) \rangle$ of $\langle 1, \ldots, n \rangle$, we identify τ and the word $\tau(1) \ldots \tau(n)$. We make two observations:

(a) $(\tau\#)^\omega \in \overline{L}_n$ for every permutation τ.

(b) If a word w contains infinitely many occurrences of two different permutations τ and τ' of $1 \ldots n$, then $w \in L_n$.

Now, let A be a Büchi automaton recognizing $\overline{L}_n$, and let τ, τ' be two arbitrary permutations of $(1, \ldots, n)$. By (a), there exist runs ρ and ρ' of A accepting $(\tau\#)^\omega$ and $(\tau'\#)^\omega$, respectively. We prove that the intersection of $\inf(\rho)$ and $\inf(\rho')$ is empty. This implies that A contains at least as many final states as permutations of $(1, \ldots, n)$, which proves the Proposition.