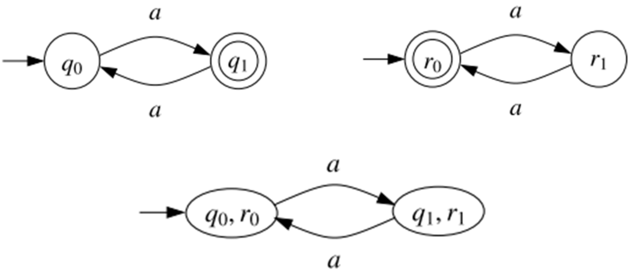
2. Implementing Boolean Operations for Büchi Automata

Intersection of NBAs

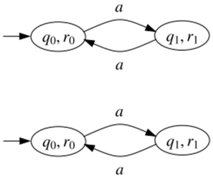
- The algorithm for NFAs does not work ...



Solution

Apply the same idea as in the conversion NGA ⇒ NBA

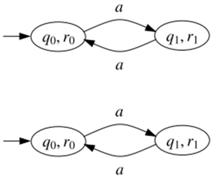
1. Take two copies of the pairing $[A_1, A_2]$.



Solution

Apply the same idea as in the conversion NGA ⇒ NBA

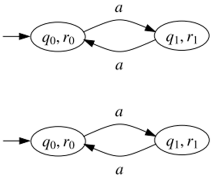
1. Take two copies of the pairing $[A_1, A_2]$.
2. Redirect transitions of the first copy leaving F_1 to the second copy.



Solution

Apply the same idea as in the conversion NGA ⇒ NBA

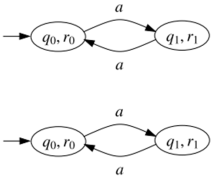
1. Take two copies of the pairing $[A_1, A_2]$.
2. Redirect transitions of the first copy leaving F_1 to the second copy.
3. Redirect transitions of the second copy leaving F_2 to the second copy.



Solution

Apply the same idea as in the conversion NGA ⇒ NBA

1. Take two copies of the pairing $[A_1, A_2]$.
2. Redirect transitions of the first copy leaving F_1 to the second copy.
3. Redirect transitions of the second copy leaving F_2 to the second copy.
4. Set F to the set F_1 in the first copy.



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 $A_1 \cap_\omega A_2 = (Q, \Sigma, \delta, q_0, F)$ with $L_\omega(A_1 \cap_\omega A_2) = L_\omega(A_1) \cap L_\omega(A_2)$

$Q, \delta, F \leftarrow \emptyset$
$q_0 \leftarrow [q_{01}, q_{02}, 1]$
$W \leftarrow \{\, [q_{01}, q_{02}, 1] \,\}$
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** $[q_1, q_2, 1]$ **to** F'
   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

- If all states of at least one of A_1 and A_2 are accepting, the algorithm for NFAs works.
- Intersection of NBAs $A_1, A_2, \ldots, A_k$
  - Do NOT apply the algorithm for two NBAs $(k-1)$ times.
  - Proceed instead as in the translation

    NGA $\Rightarrow$ NBA: take k copies of $[A_1, A_2, \ldots, A_k]$

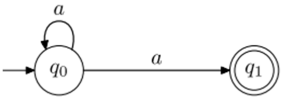
    ($kn_1 \ldots n_k$ states instead of $2^k n_1 \ldots n_k$)

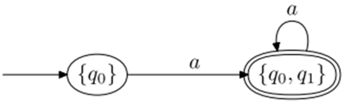
Complement

- Main result proved by Büchi: NBAs are closed under complement.
- Many later improvements in recent years.
- Construction radically different from the one for NFAs.

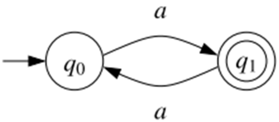
Problems

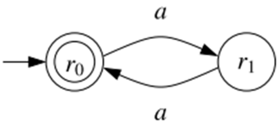
- The powerset construction does not work.





- Exchanging final and non-final states in DBAs also fails.



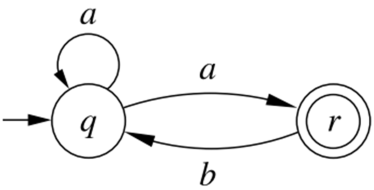


Solution

- Extend the idea used to determinize co-Büchi automata with a new component.
- Recall: a NBA accepts a word w iff some path of $dag(w)$ visits final states infinitely often.
- Goal: given NBA A, construct NBA $\bar{A}$ such that:

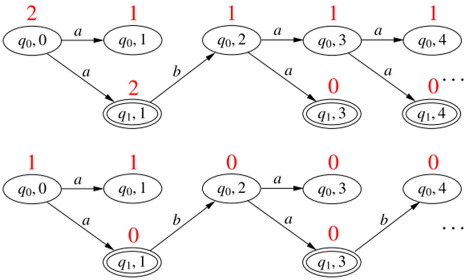
A rejects w
iff
no path of $dag(w)$ visits accepting states of A i.o.
iff
some run of $\bar{A}$ visits accepting states of $\bar{A}$ i.o.
iff
$\bar{A}$ accepts w

Running example



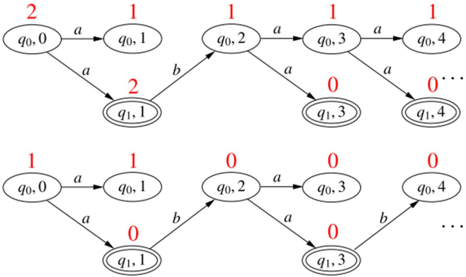
Rankings

- Mappings that associate to every node of $dag(w)$ a rank (a natural number) such that
  - ranks never increase along a path, and
  - ranks of accepting nodes are even.



Odd rankings

- A ranking is odd if every infinite path of $dag(w)$ visits nodes of odd rank i.o.



Prop.: no path of $dag(w)$ visits accepting states of A i.o.
iff
$dag(w)$ has an odd ranking

Proof: Ranks along infinite paths eventually reach a stable rank.

(←): The stable rank of every path is odd. Since accepting nodes have even rank, no path visits accepting nodes i.o.

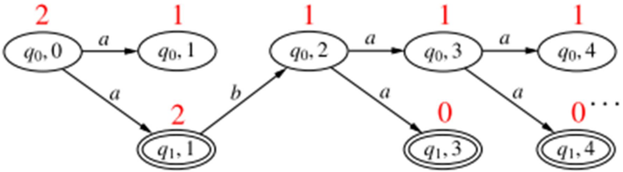
(→): We construct a ranking satisfying the conditions.
Give each accepting node $\langle q, l \rangle$ rank $2k$, where k is the maximal number of accepting nodes in a path starting at $\langle q, l \rangle$.
Give a non-accepting node $\langle q, l \rangle$ rank $2k + 1$, where 2k is the maximal even rank among its descendants.

- Goal:

> A rejects w
> iff
> $dag(w)$ has an odd ranking
> iff
> some run of $\bar{A}$ visits accepting states of $\bar{A}$ i.o.
> iff
> $\bar{A}$ accepts w

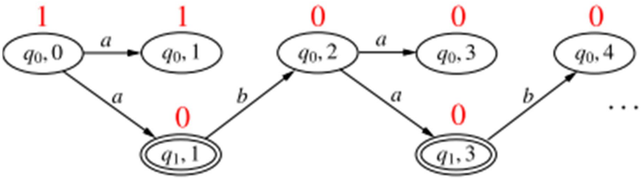
- Idea: design $\bar{A}$ so that
  - its runs on w are the rankings of $dag(w)$, and
  - its acceptings runs on w are the odd rankings of $dag(w)$.

Representing rankings



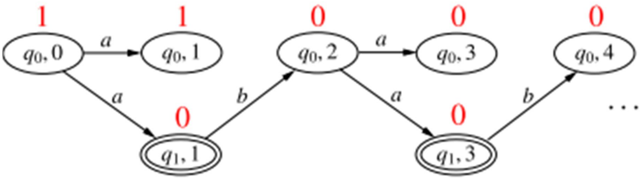
$$\begin{bmatrix}2\\ \bot\end{bmatrix} \xrightarrow{a} \begin{bmatrix}1\\ 2\end{bmatrix} \xrightarrow{b} \begin{bmatrix}1\\ \bot\end{bmatrix} \xrightarrow{a} \begin{bmatrix}1\\ 0\end{bmatrix} \xrightarrow{a} \begin{bmatrix}1\\ 0\end{bmatrix} \cdots$$

Representing rankings



$$\begin{bmatrix}1\\\bot\end{bmatrix} \stackrel{a}{\to} \begin{bmatrix}1\\0\end{bmatrix} \stackrel{b}{\to} \begin{bmatrix}0\\\bot\end{bmatrix} \stackrel{a}{\to} \begin{bmatrix}0\\0\end{bmatrix} \stackrel{b}{\to} \begin{bmatrix}0\\\bot\end{bmatrix} \cdots$$

Representing rankings



$$\begin{bmatrix}1\\ \bot\end{bmatrix} \xrightarrow{a} \begin{bmatrix}1\\ 0\end{bmatrix} \xrightarrow{b} \begin{bmatrix}0\\ \bot\end{bmatrix} \xrightarrow{a} \begin{bmatrix}0\\ 0\end{bmatrix} \xrightarrow{b} \begin{bmatrix}0\\ \bot\end{bmatrix} \dots$$

- We can determine if $\begin{bmatrix}n_1\\ n_2\end{bmatrix} \xrightarrow{l} \begin{bmatrix}n'_1\\ n'_2\end{bmatrix}$ may appear in a ranking by just looking at n_1, n_2, n'_1, n'_2 and l : ranks should not increase.

First draft for $\bar{A}$

- For a two-state A (more states analogous):
  - States: all $\begin{bmatrix} n_1 \\ n_2 \end{bmatrix}$ where accepting states get even rank
  - Initial states: all states of the form $\begin{bmatrix} n_1 \\ \bot \end{bmatrix}$
  - Transitions: all $\begin{bmatrix} n_1 \\ n_2 \end{bmatrix} \xrightarrow{a} \begin{bmatrix} n_1' \\ n_2' \end{bmatrix}$ s.t . ranks don´t increase
- The runs of the automaton on a word w correspond to all the rankings of $dag(w)$.
- Observe: $\bar{A}$ is a NBA even if A is a DBA, because there are many rankings for the same word.

Problems to solve

- How to choose the accepting states?
  - They should be chosen so that a run is accepted iff its corresponding ranking is odd.
- Potentially infinitely many states (because rankings can contain arbitrarily large numbers)