Safra's Büchi determinization algorithm

Aditya Oak

Seminar on Automata Theory
28 Jan 2016

Introduction

- Proposed by S. Safra in 1988
- For determinization of non-deterministic Büchi automaton
- Gives equivalent Rabin or Muller automaton
- Involves construction of multiple powersets in a tree structure called as Safra Trees

Applying powerset construction to Büchi automaton(1)

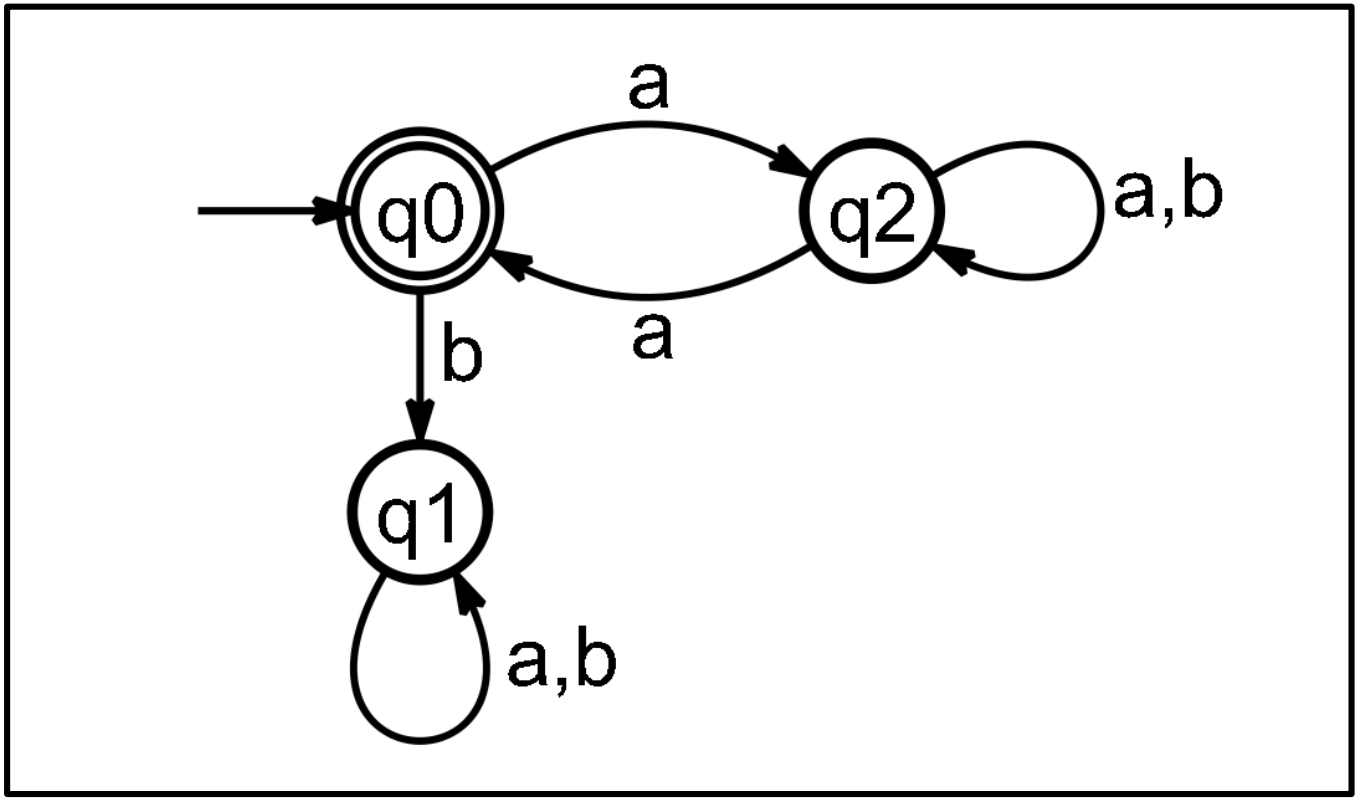


Fig. 1. Non-deterministic Büchi automaton

Accepts $a(aab)^{\omega}$

Applying powerset construction to Büchi automaton(2)

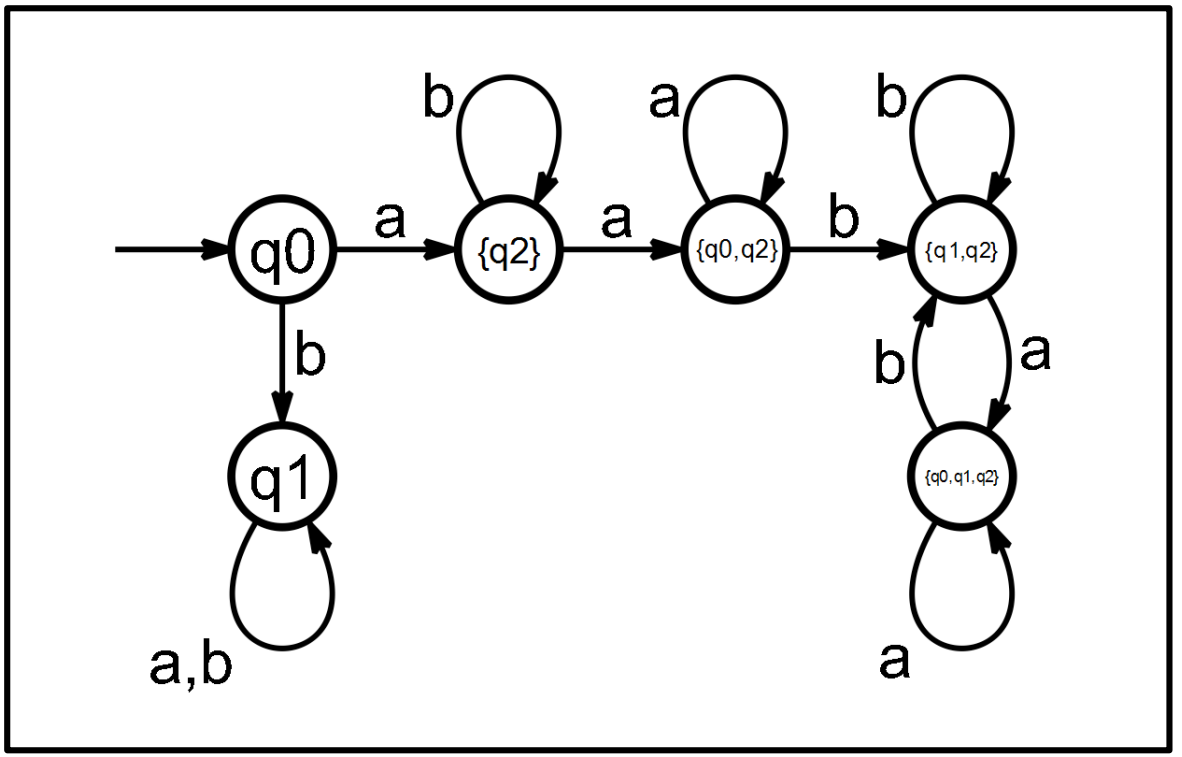


Fig. 2. Deterministic automaton after powerset construction

Acceptance condition for accepting **$a(aab)^{\omega}$** ?

No Rabin acceptance condition is applicable. No accepting condition uniquely accepts the language.

Hence classical powerset construction does not work in this scenario !!!

Intuition behind Safra's algorithm(1)

- **Trick 1 :** “Initialize new runs of macrostates starting from recurring states”[1, p. 47].

  This allows the construction of an accepting run of the original automaton. A new component is added to Safra tree whenever recurring state occurs in a macrostate.
  Using this, we track the paths which traverse through recurring states.

  (Known as 'Branch Accepting')

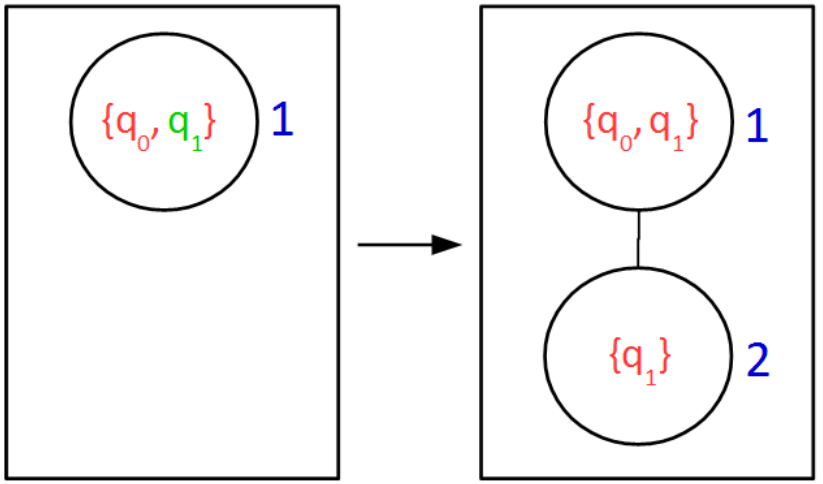


Fig. 3. Branch Accepting

Intuition behind Safra's algorithm(2)

- **Trick 2 : "**Keep track of joining runs of the non-deterministic Büchi automaton just once"[1, p. 47].

  $q_1 q_2 \ldots f q_i \ldots q_{j-1} q_j \ldots q_n q_{n+1}$ and

  $q'_1 q'_2 \ldots q'_{i-1} q'_i \ldots f' q'_j \ldots q'_n q_{n+1}$

  Both runs are joining in state q_{n+1}, after passing through a recurring state. Hence it is sufficient to continue with only one added component, second component can be removed as both the components hold the same information.

  (Known as 'Horizontal Merge')

Intuition behind Safra's algorithm(3)

- **Trick 3 : “**If all states in a macrostate have a recurring state as predecessor, delete the corresponding components”[1, p. 48].

  When a node's label is equal to the union of the labels of its child nodes, then they all track the same run and hence all child nodes can be removed.

  (Known as 'Vertical Merge')

Safra Tree(1)

- Consists of nodes
- Safra trees **form the states of resultant automaton**
- Each node in a Safra tree has
  - Name – Unique number in a single tree ({1, . . . ,2*n*})
    (*n* – number of states of original automaton)
  - Label / Macrostate – Set of states of original automaton
  - Mark (!) - A Boolean (i.e. either a node is marked or unmarked)

Safra Tree(2)

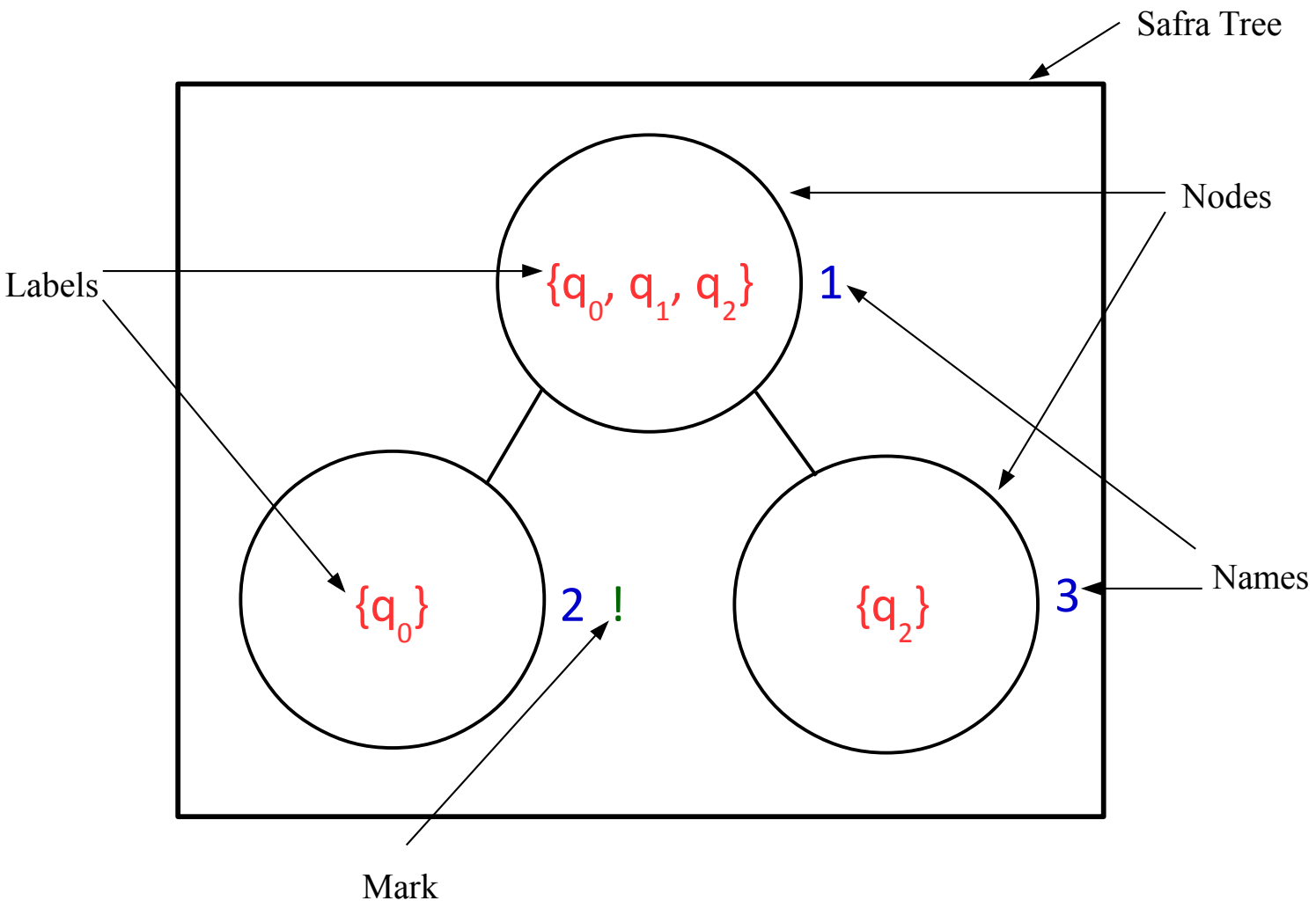


Fig. 4. A Safra Tree

Safra Tree(3)

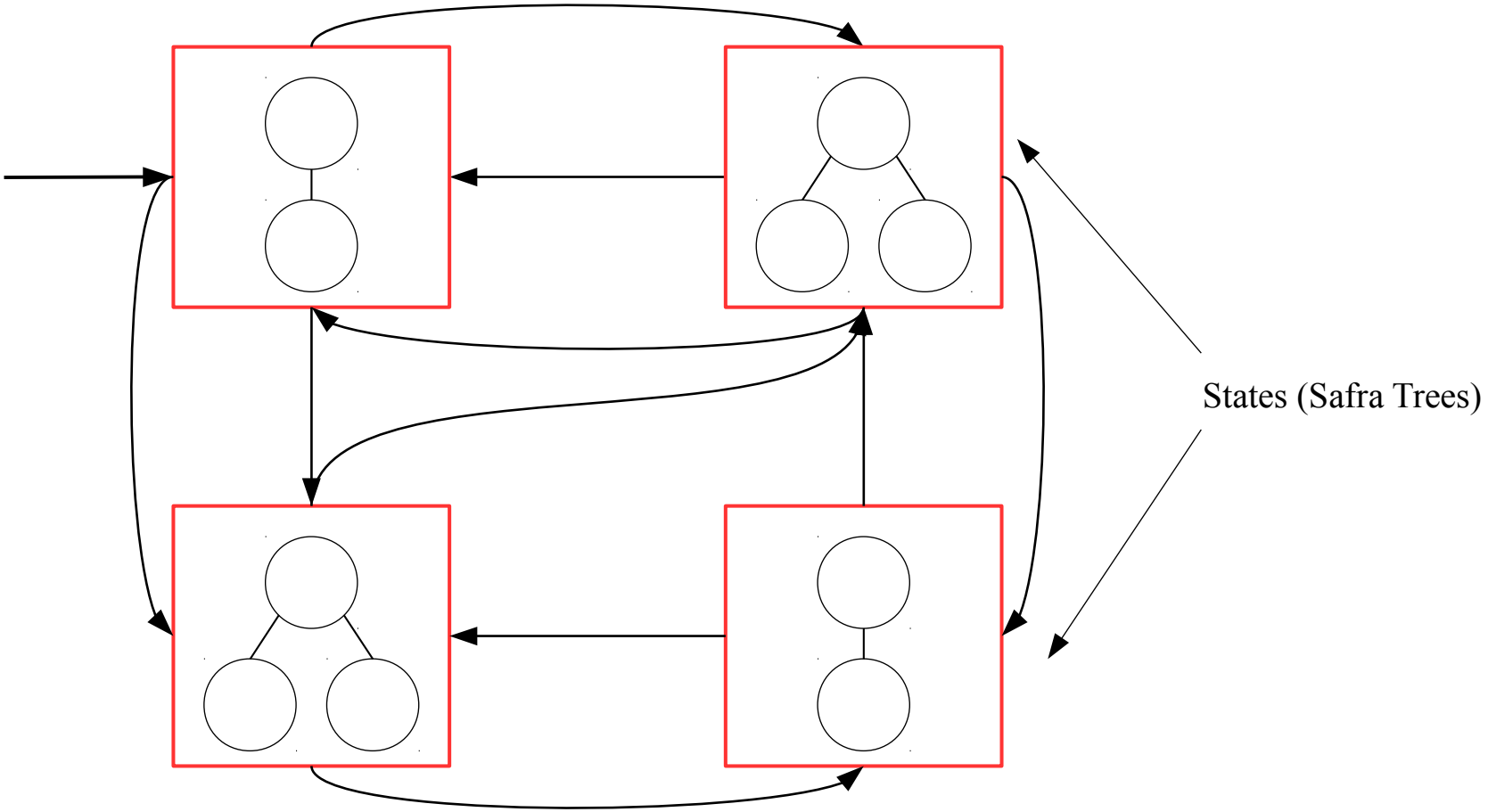
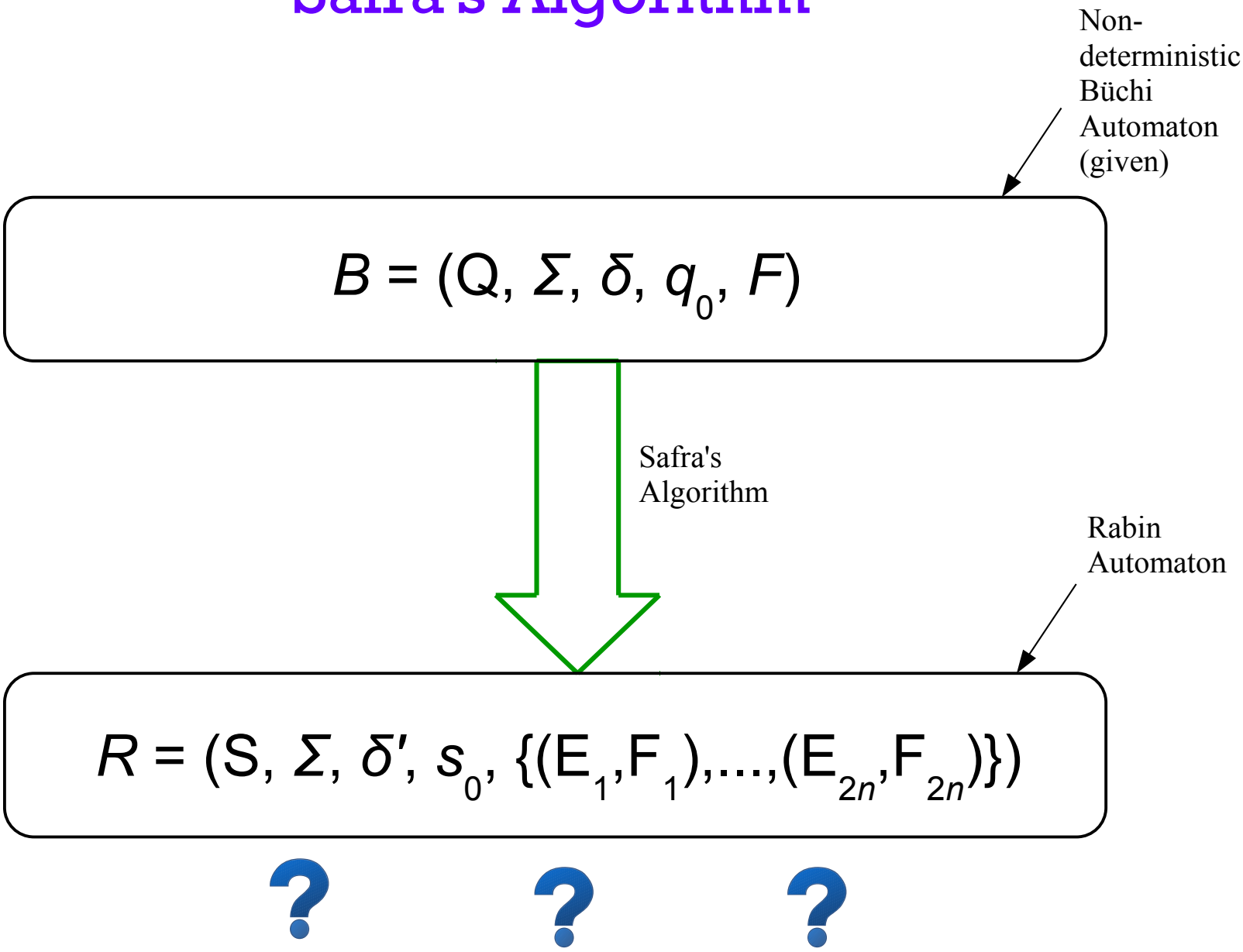


Fig. 5. Resultant deterministic Rabin or Muller automaton

Safra's Algorithm



Safra's Algorithm – computation of Rabin automaton parameters (1)

- Computation of initial state s_0 -
  - Initial state s_0 is a Safra tree with single node having name 1 and label q_0

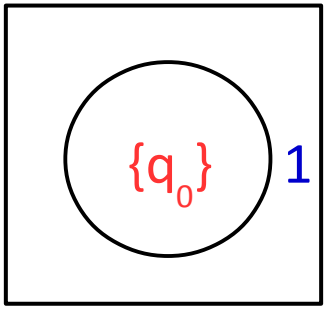


Fig. 6. State s_0

- Computation of S -
  - S is set of all reachable Safra trees from initial Safra tree s_0

Safra's Algorithm – computation of Rabin automaton parameters (2)

- Computation of δ' (transition function) -
  - Every single transition in Rabin automaton is obtained using following six steps
    1. ***Remove marks*** – All the marks (!) in a Safra tree are removed.
    2. ***Branch accepting*** – For every node in a Safra tree, if label of a node contains at least one accepting state, then new youngest child node of this node is added with unique name. Label of this child node is set of accepting states in parent node.

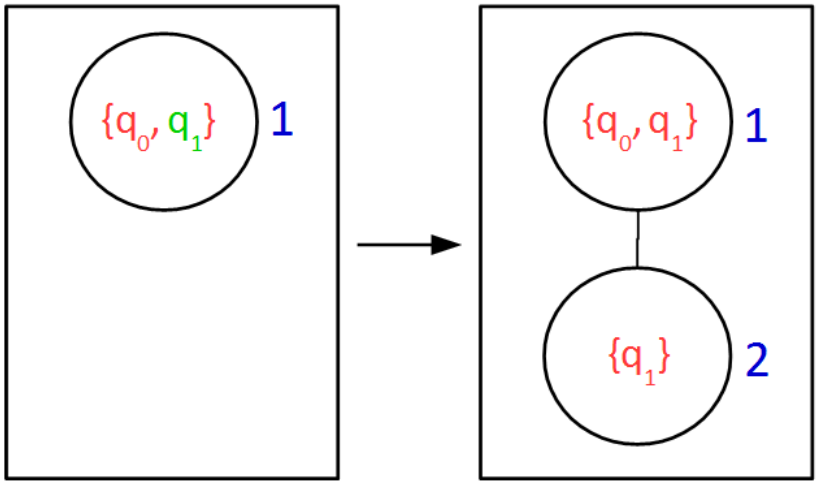


Fig. 7. Branch Accepting

Safra's Algorithm – computation of Rabin automaton parameters (3)

- Computation of δ' (transition function) (cont.) -

  3. ***Powerset construction*** – Apply powerset construction on every node of Safra tree.

  4. ***Horizontal merge*** – If a particular state in a node's label is also present in the label of the node's older brother then that state is removed from the node's label and also from all its children.

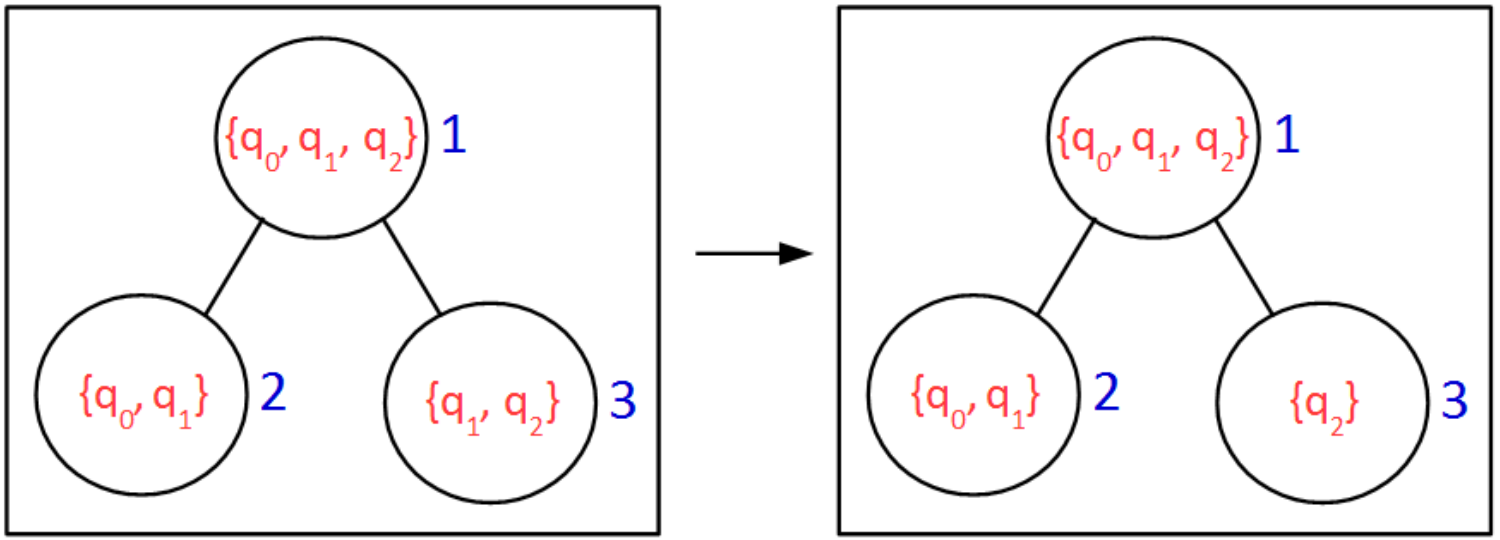


Fig. 8. Horizontal Merge

Safra's Algorithm – computation of Rabin automaton parameters (4)

- Computation of δ' (transition function) (cont.) -

  5. ***Remove empty nodes*** – Nodes having empty labels are removed.

  6. ***Vertical merge*** – For every node whose label is equal to the union of the labels of its child nodes, all its child nodes are removed and node is marked with '!'.

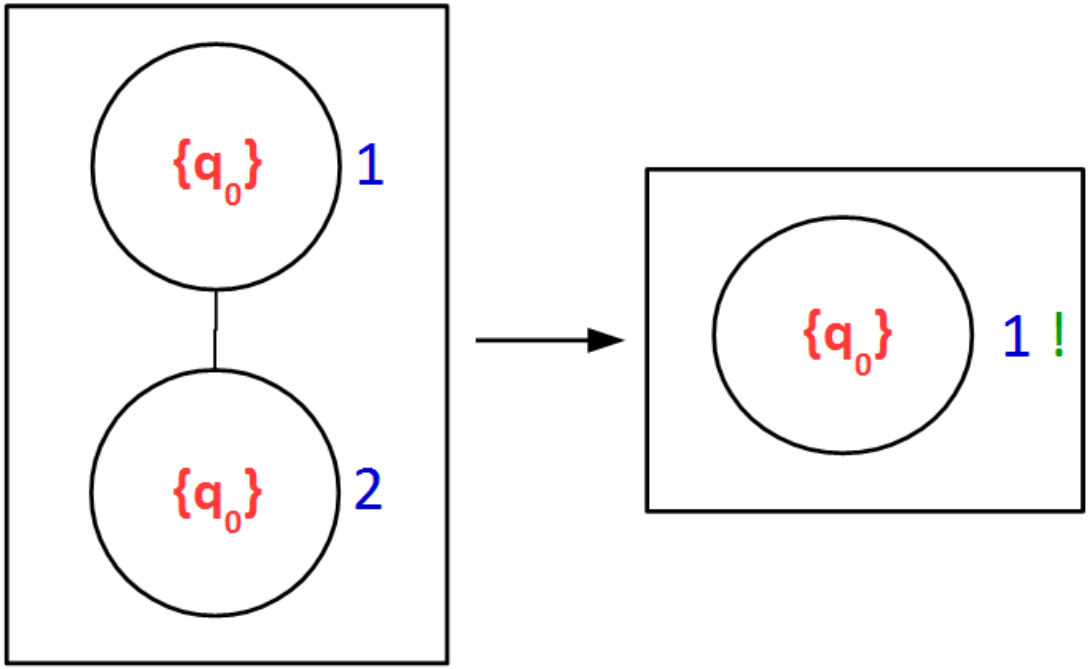


Fig. 9. Vertical Merge

Safra's Algorithm – computation of Rabin automaton parameters (5)

- Computation of acceptance condition -
  - For set of pairs $\{(E_1,F_1),\ldots,(E_{2n},F_{2n})\}$,

    E_i : Safra trees without node *i (i* is name of a node*)*

    F_i : Safra trees with node *i* marked '!'

Safra's Algorithm – Example

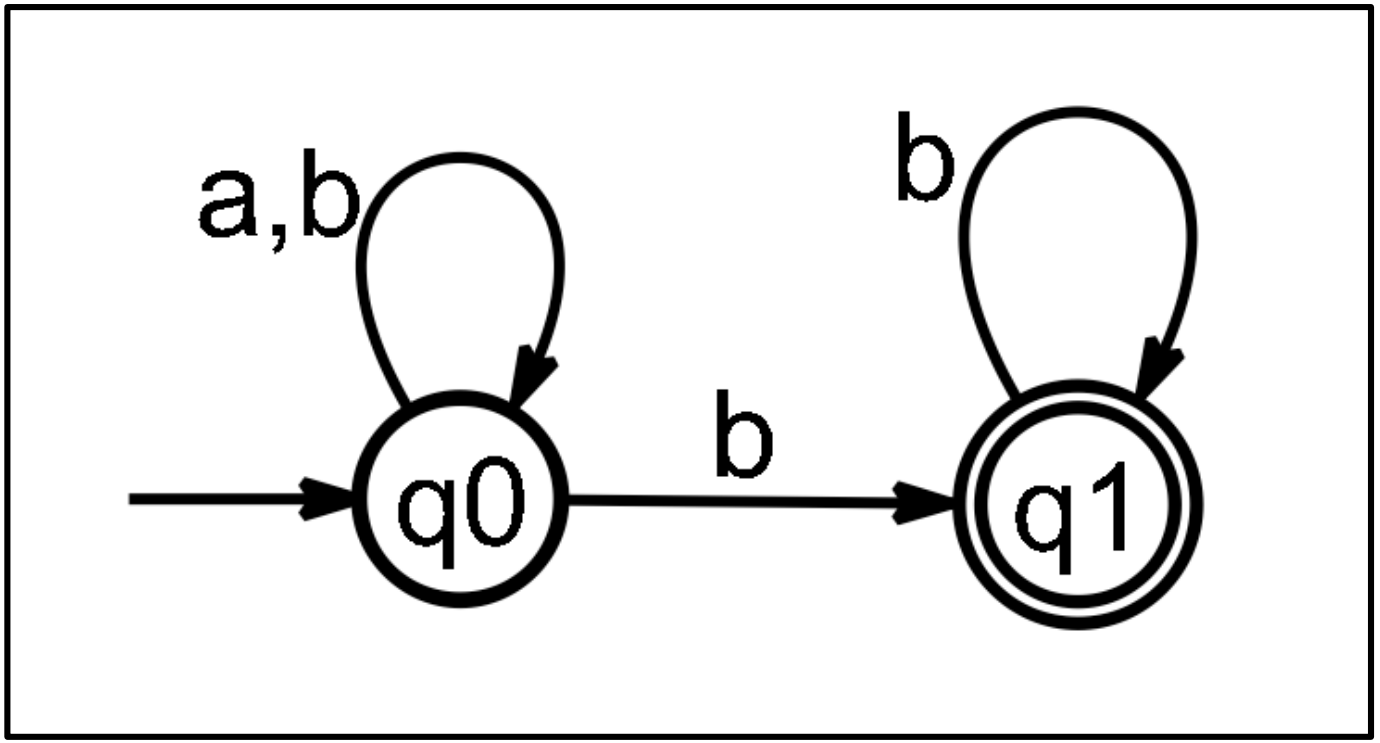


Fig. 10. Büchi automaton

Accepts finitely many **a**s and infinitely many **b**s i.e. $\Sigma^*(b)^\omega$

Safra's Algorithm – computing s_0

- Computation of initial state s_0 -
  - Initial state s_0 is a Safra tree with single node having name 1 and label q_0

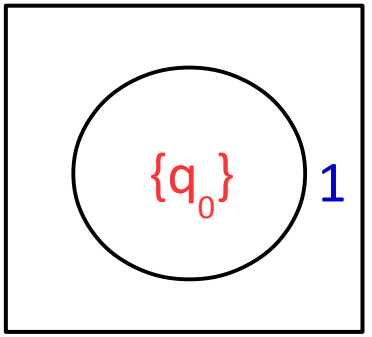
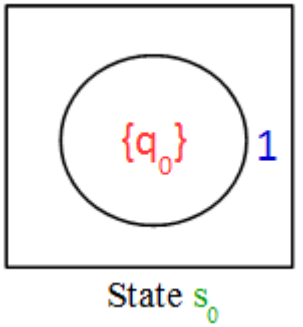


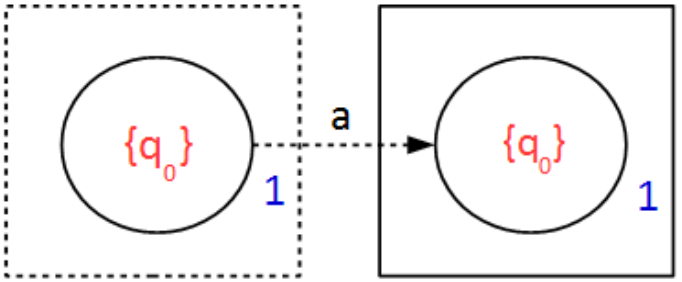
Fig. 11. State s_0

Safra's Algorithm – computing δ'

- Computation of $\delta'(s_0, a)$ -



  - Step 1) **Remove marks** – No change
  - Step 2) **Branch accepting** - No change
  - Step 3) **Powerset construction** -
    - In Büchi automaton $\delta(q_0, a) = q_0$



  - Step 4) **Horizontal merge** – No change
  - Step 5) **Remove empty nodes** – No change
  - Step 6) **Vertical merge** – No change

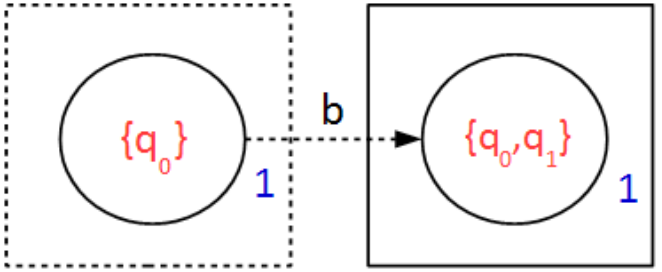
  Hence we have $\delta'(s_0, a) = s_0$

Rabin automaton parameters computed so far.

- s_0

Safra's Algorithm – computing δ'

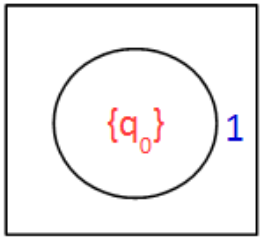
- Computation of $\delta'(s_0, b)$ -
  - Step 1) **Remove marks** – No change
  - Step 2) **Branch accepting** – No change
  - Step 3) **Powerset construction** -
    - In Büchi automaton $\delta(q_0, b) = \{q_0, q_1\}$



  - Step 4) **Horizontal merge** – No change
  - Step 5) **Remove empty nodes** – No change
  - Step 6) **Vertical merge** – No change

    New state obtained !!!

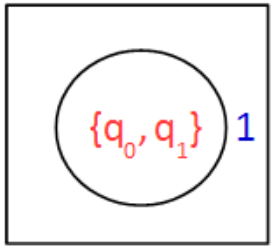
    Hence we have $\delta'(s_0, b) = s_1$



State s_0

Rabin automaton parameters computed so far.

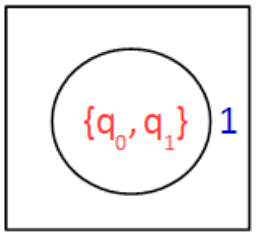
- s_0
- $\delta'(s_0, a) = s_o$



State s_1

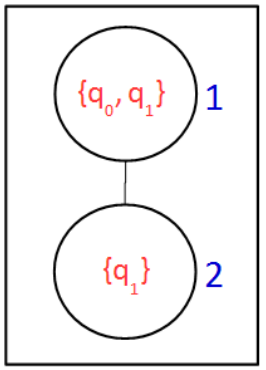
Safra's Algorithm – computing δ′

- Computation of $\delta'(s_1, a)$ -

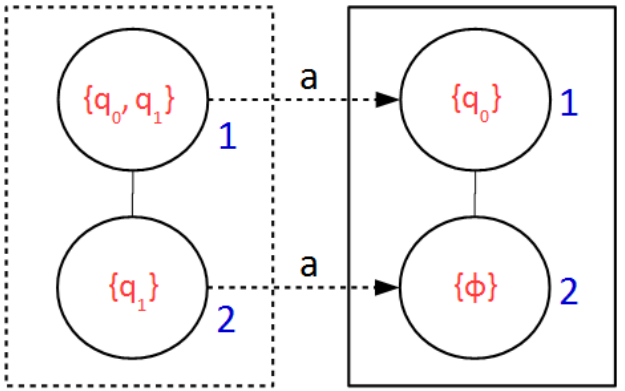


State s_1

  - Step 1) **Remove marks** – No change
  - Step 2) **Branch accepting** – Applicable, as q_1 is an accepting state in original automaton, hence new child with unique name has to be added.



- Step 3) **Powerset construction** -

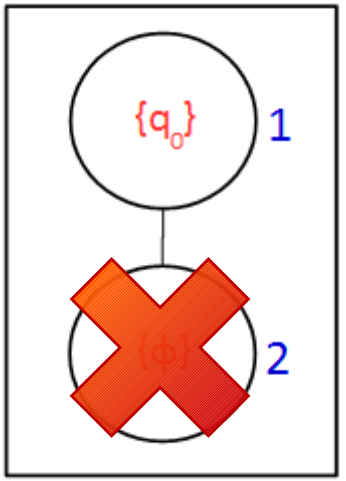


Rabin automaton parameters computed so far.

- s_0
- $\delta'(s_0, a) = s_o$
- $\delta'(s_0, b) = s_1$
- s_1

Safra's Algorithm – computing δ'

- Computation of $\delta'(s_1, a)$ (cont.) -
  - Step 4) **Horizontal merge** – No change
  - Step 5) **Remove empty nodes** – Applicable, node 2 is removed



  - Step 6) **Vertical merge –** No change

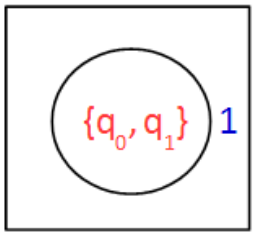
    Hence we have $\delta'(s_1, a) = s_0$

Rabin automaton parameters computed so far.

- s_0
- $\delta'(s_0, a) = s_0$
- $\delta'(s_0, b) = s_1$
- s_1

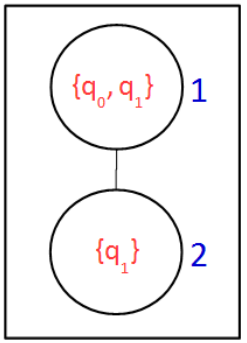
Safra's Algorithm – computing δ′

- Computation of $\delta'(s_1, b)$ -

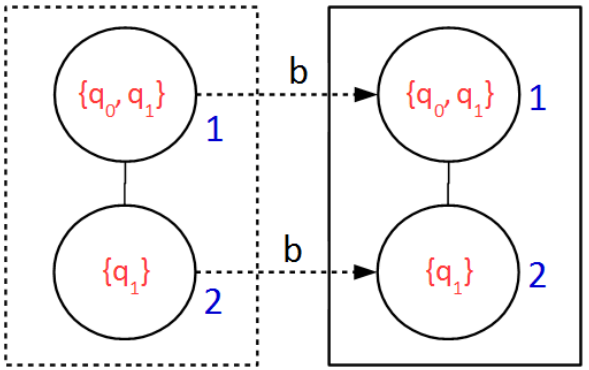


State s_1

  - Step 1) **Remove marks** – No change
  - Step 2) **Branch accepting** – Applicable, as q_1 is an accepting state in original automaton, hence new child with unique name has to be added.



  - Step 3) **Powerset construction** -



Rabin automaton parameters computed so far.

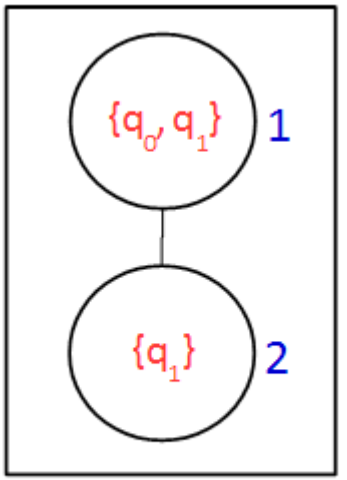
- s_0
- $\delta'(s_0, a) = s_o$
- $\delta'(s_0, b) = s_1$
- s_1
- $\delta'(s_1, a) = s_o$

Safra's Algorithm – computing δ'

- Computation of $\delta'(s_1, b)$ (cont.) -
  - Step 4) **Horizontal merge** – No change
  - Step 5) **Remove empty nodes** – No change
  - Step 6) **Vertical merge** – No change

    New state obtained !!!

    Hence we have $\delta'(s_1, b) = s_2$



State s_2

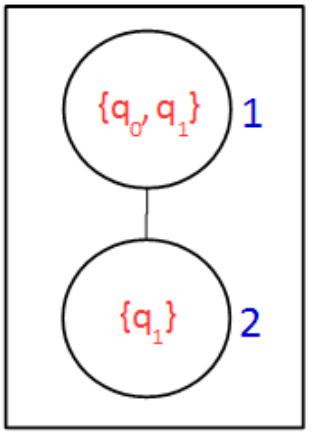
Rabin automaton parameters computed so far.

- s_0
- $\delta'(s_0, a) = s_o$
- $\delta'(s_0, b) = s_1$
- s_1
- $\delta'(s_1, a) = s_o$

Safra's Algorithm – computing δ'

- Computation of $\delta'(s_2, a)$ -
  - After applying all the six steps again we get

    $\delta'(s_2, a) = s_0$

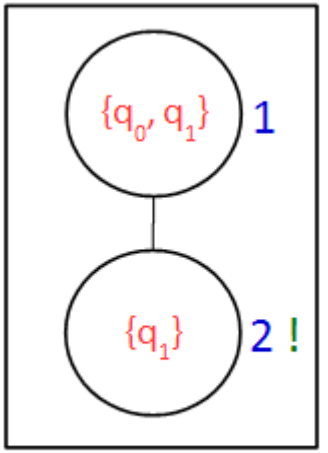


State s_2

- Computation of $\delta'(s_2, b)$ -
  - After applying all the six steps again we get

    $\delta'(s_2, b) = s_3$

Only difference between state $\mathbf{s_2}$ and state $\mathbf{s_3}$ is the marking of node 2. Therefore transitions from state $\mathbf{s_3}$ are same as that from state $\mathbf{s_2}$.



State s_3

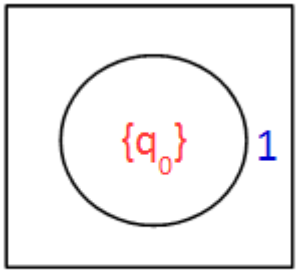
Rabin automaton parameters computed so far.

- s_0
- $\delta'(s_0, a) = s_o$
- $\delta'(s_0, b) = s_1$
- s_1
- $\delta'(s_1, a) = s_o$
- $\delta'(s_1, b) = s_2$
- s_2

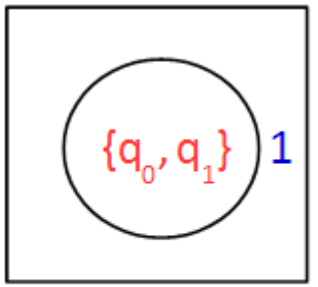
Safra's Algorithm – computing δ′

Therefore we get transitions and states of Rabin automaton as -

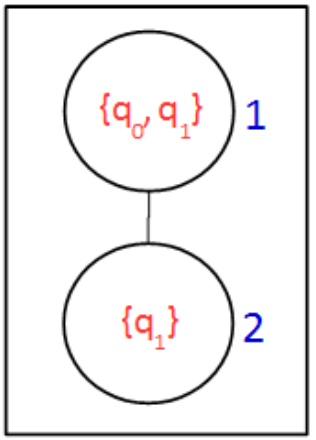
- $\delta'(s_0, a) = s_o$
- $\delta'(s_0, b) = s_1$
- $\delta'(s_1, a) = s_o$
- $\delta'(s_1, b) = s_2$
- $\delta'(s_2, a) = s_0$
- $\delta'(s_2, b) = s_3$
- $\delta'(s_3, a) = s_0$
- $\delta'(s_3, b) = s_3$



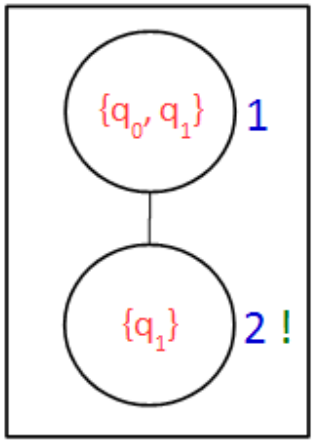
State s_0



State s_1



State s_2



State s_3

Safra's Algorithm – Acceptance condition

Set of accepting pairs for Rabin automaton -

(For set of pairs $\{(E_1,F_1),\ldots,(E_{2n},F_{2n})\}$,
E_i : Safra trees without node i
F_i : Safra trees with node i marked '!')

$(E_1, F_1) = (\phi, \phi)$ as node 1 is present in all the states and it is never marked.
$(E_2, F_2) = (\{s_0, s_1\}, \{s_3\})$ as node 2 is absent in states $\mathbf{s_0}$ and $\mathbf{s_1}$ and node is marked in state $\mathbf{s_3}$.

Safra's Algorithm – equivalent Rabin automaton

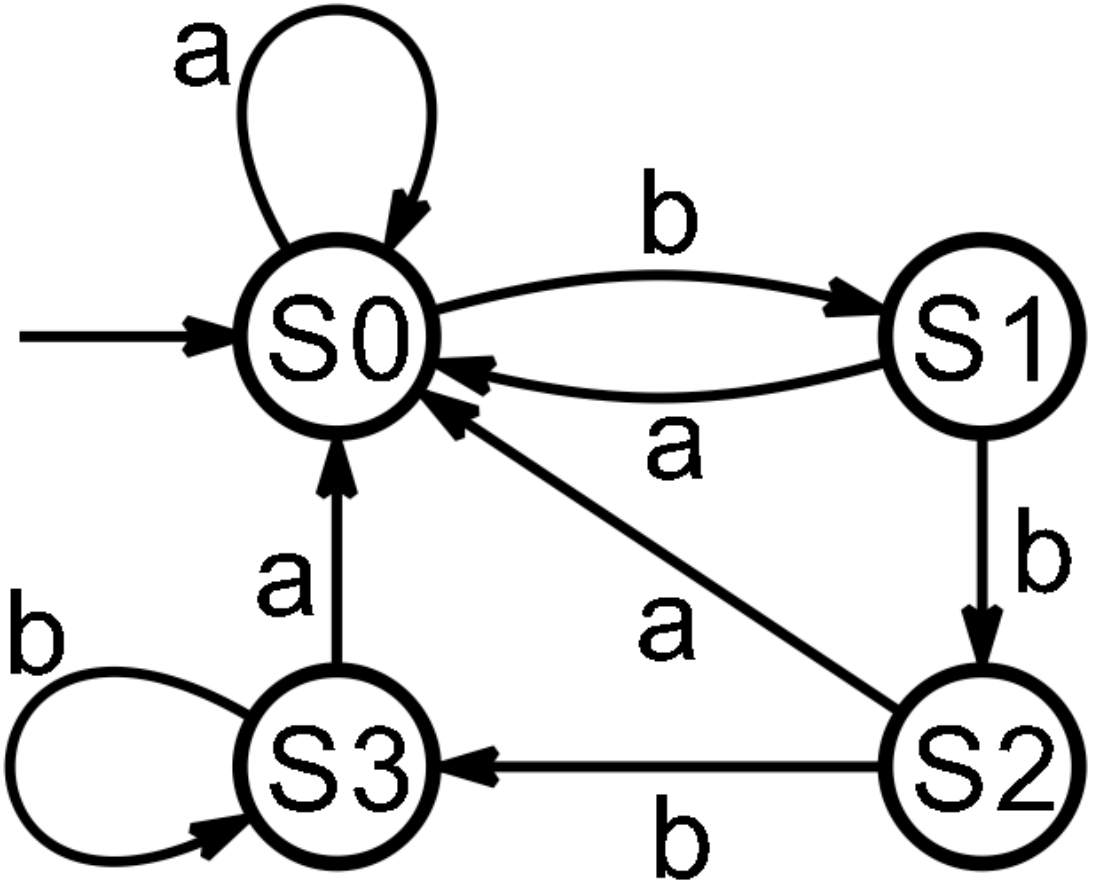


Fig. 12. Equivalent Rabin automaton

Accepts $\Sigma^*(b)^\omega$

Safra's Algorithm – Remarks

- State complexity - $2^{O(n \log n)}$ (improved over previous complexities of $2^{2^{O(n)}}$)
- Optimal for conversion into Rabin automaton
- Using this algorithm as an intermediate step, Büchi automaton can be complemented with same complexity which is also optimal.

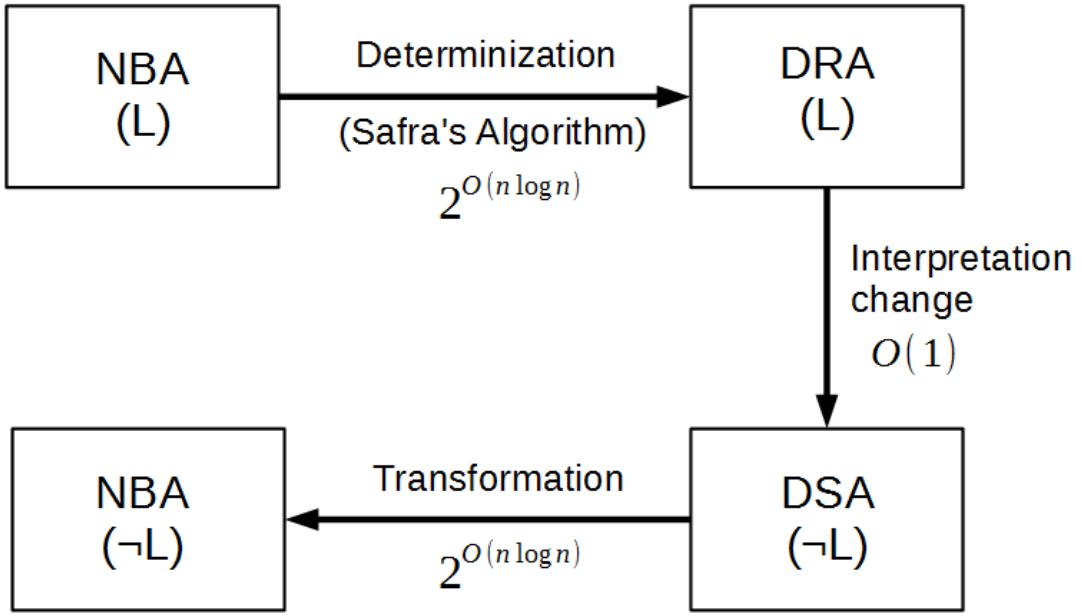


Fig. 13. Complementation steps
Source : [2][5]

References

1. Gradel E., Thomas W., Wilke T. : Automata, logics, and infinite games : A guide to current research (2002)

2. Sadegh G. : Complementing Büchi Automata. https://www.lrde.epita.fr/~sadegh/buchi-complementation-techrep.pdf (2009)

3. Bienvenu M. : Automata on infinite words and trees. http://www.informatik.uni-bremen.de/tdki/lehre/ws09/automata/automata-notes.pdf (2010)

4. Safra S. : On the Complexity of ω – Automata (1988)

5. Panigrahi D. : Complementing Büchi Automata : Safra's construction. http://www.powershow.com/view1/1eeb50-ZDc1Z/Complementing_B_powerpoint_ppt_presentation

6. K. Narayan Kumar : Safra's Determinization construction. http://www.cmi.ac.in/~kumar/words/lecture11.pdf