TU Graz

SCIENCE
PASSION
TECHNOLOGY

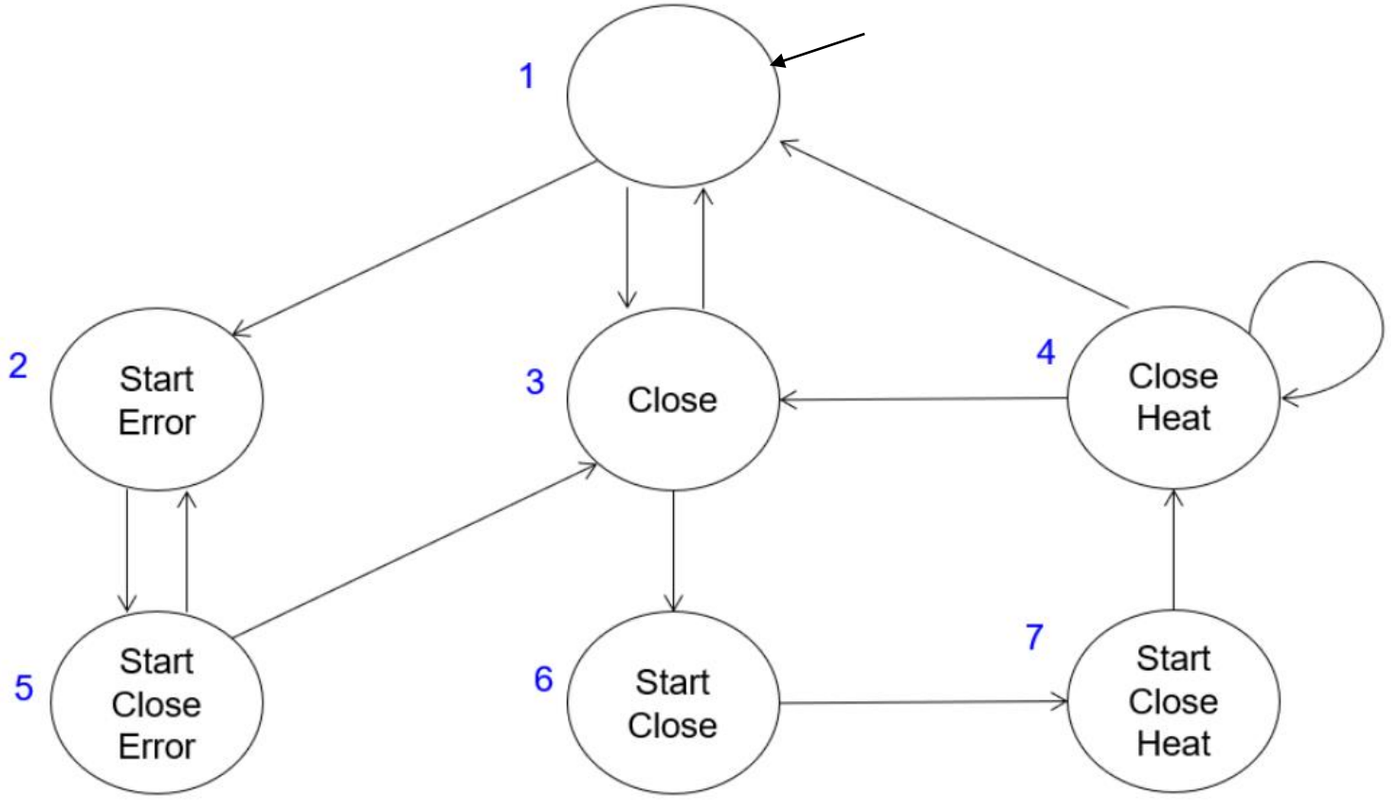
Model Checking for LTL

Bettina Könighofer
bettina.koenighofer@tugraz.at

Plan for Today

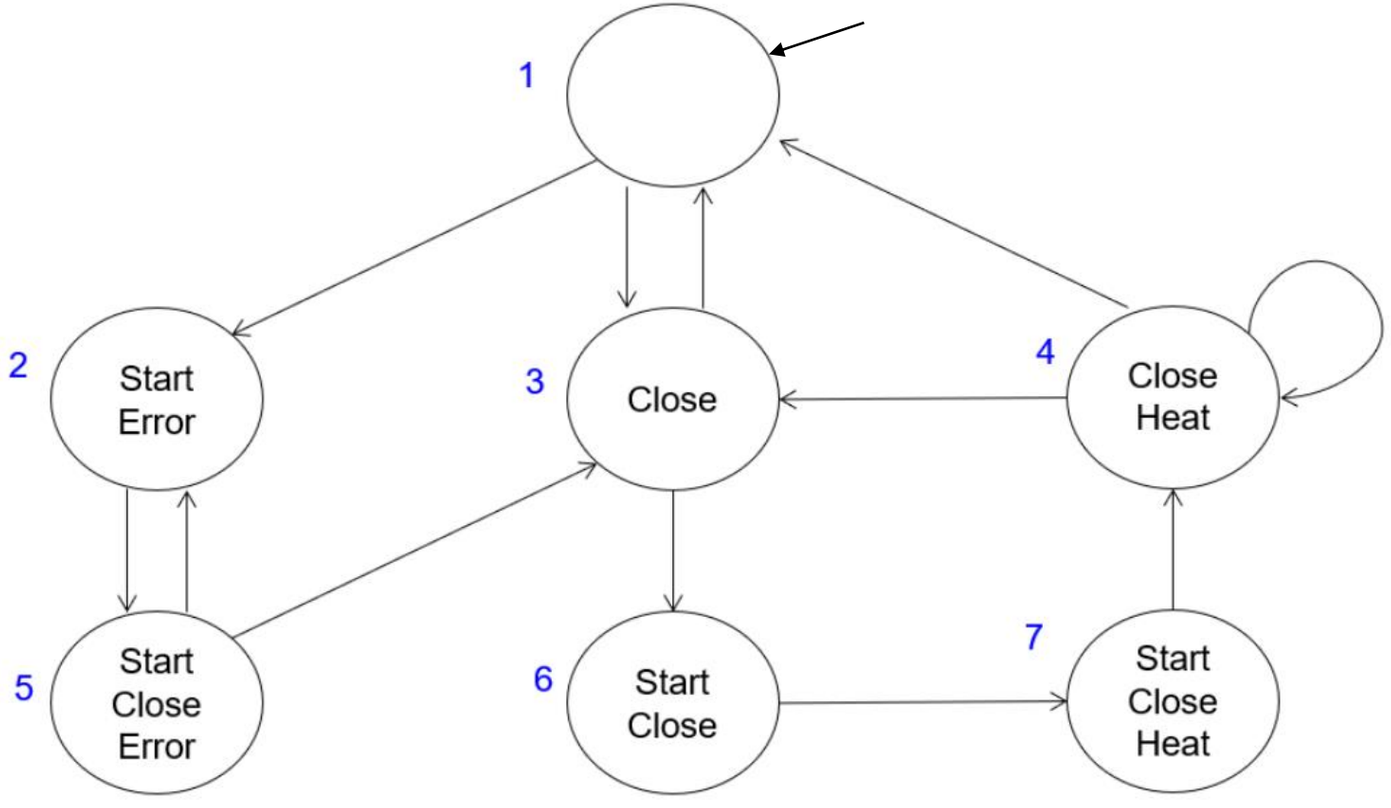
- Presentation of Homework
- LTL Model Checking
  - Recap / Overview
  - Basics of Automata Theory
  - LTL Model-Checking Algorithm

Homework Task 6a: $f_1 = \neg EG(start \rightarrow EX\ (error))$



1. Check $start$, $error$ $\implies$ $[[start]]_M = \{2, 5, 6, 7\}$, $[[error]]_M = \{2, 5\}$
2. Check $f_1^{(1)} \equiv \mathbf{EX}$ error $\implies$ $[[f_1^{(1)}]]_M = \{1, 2, 5\}$
3. Check $f_1^{(2)} \equiv \text{start} \rightarrow f_1^{(1)}$ $\implies$ $[[f_1^{(2)}]]_M = \{1, 2, 3, 4, 5\}$
4. Check $f_1^{(3)} \equiv \mathbf{EG}\ f_1^{(2)}$ $\implies$ $[[f_1^{(3)}]]_M = \{1, 2, 3, 4, 5\}$
5. Check $f_1 \equiv \neg f_1^{(3)}$ $\implies$ $[[f_1]]_M = \{6, 7\}$

Homework Task 6b: $f_2 = EF(\, E(start\ U\ close) \wedge EG\ close)$



$$[[close]]_M = \{3, 4, 5, 6, 7\}$$

1. Check $start$, $close$ $\implies$ $[[start]]_M = \{2, 5, 6, 7\}$
2. Check $f_2^{(1)} \equiv \mathbf{EG}$ close $\implies$ $[[f_2^{(1)}]]_M = \{3, 4, 6, 7\}$
3. Check $f_2^{(2)} \equiv \mathbf{E}$ (start $\mathbf{U}$ close) $\implies$ $[[f_2^{(2)}]]_M = \{2, 3, 4, 5, 6, 7\}$
4. Check $f_2^{(3)} \equiv f_2^{(2)} \wedge f_2^{(1)}$ $\implies$ $[[f_2^{(3)}]]_M = \{3, 4, 6, 7\}$
5. Check $f_2 \equiv \mathbf{EF}\ f_2^{(3)}$ $\implies$ $[[f_2]]_M = \{1, 2, 3, 4, 5, 6, 7\}$

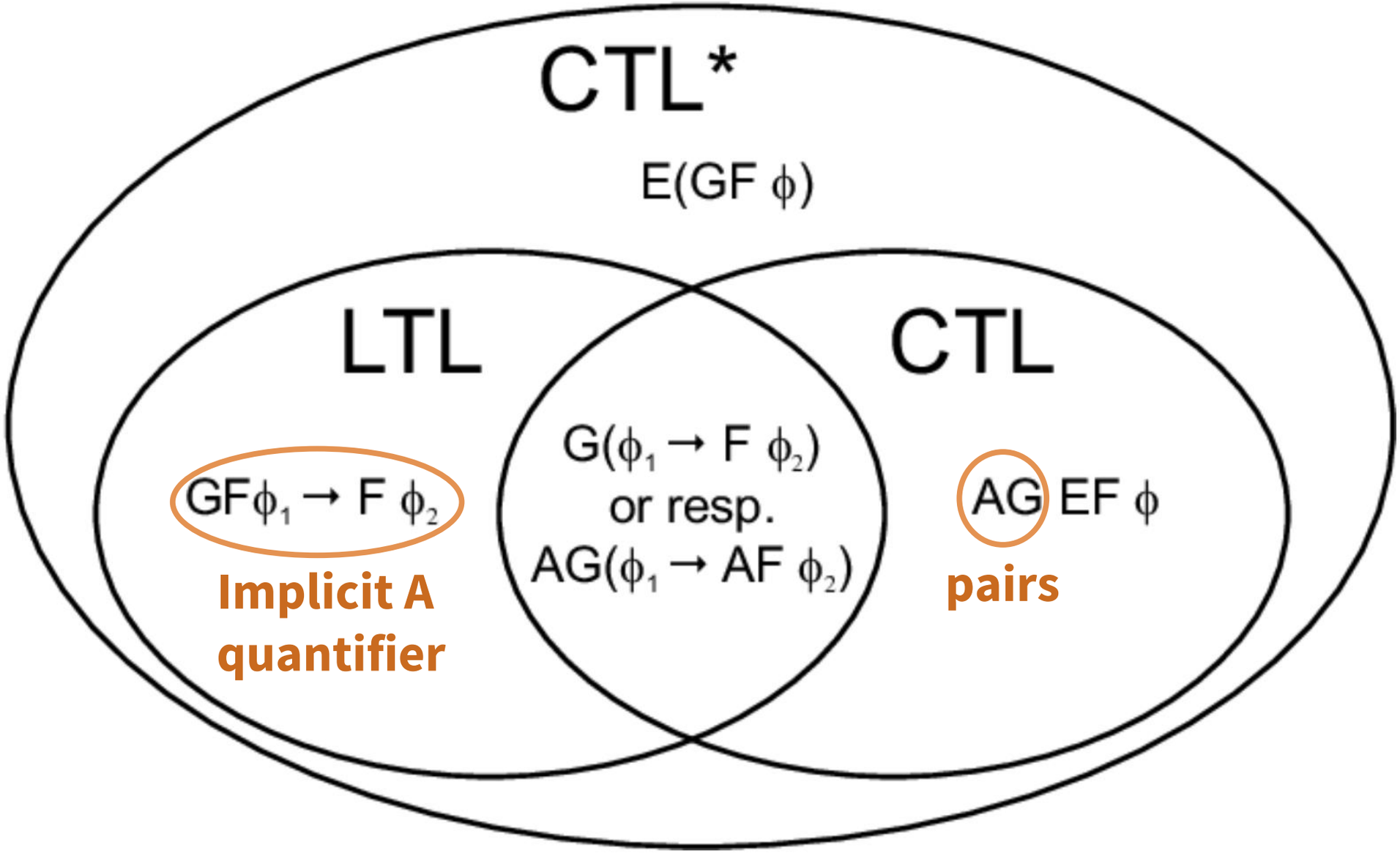
As $S_0 \subseteq [[f_2]]_M$, $M \models f_2$ holds.

Plan for Today

- Presentation of Homework

- LTL Model Checking
  - Recap / Overview
  - Basics of Automata Theory
  - LTL Model-Checking Algorithm

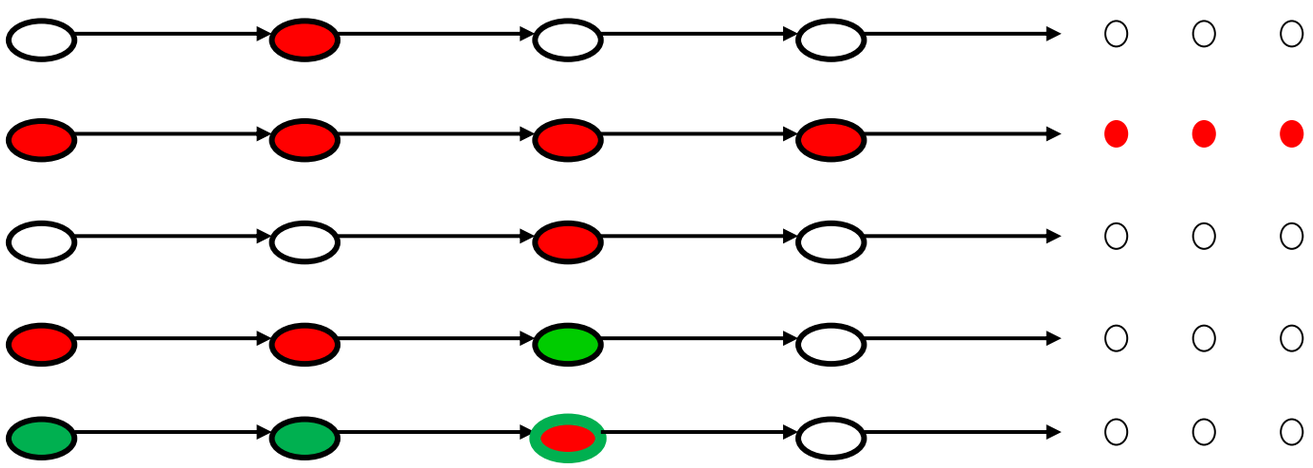
Recap - LTL/CTL/CTL*



Recap - Temporal Operators

- Temporal operators
  - Describe **properties** along a given **path**
- AP: a set of atomic propositions, $p, q \in AP$

- **Next:** Xp
- **Globally:** Gp
- **Eventually:** Fp
- **Until:** pUq
- **Release:** pRq



pRq … "p releases q": q has to hold until p holds.
However, p is not required to hold eventually.

Recap - LTL - Syntax

- **State formulas**
  - $\boldsymbol{A}g$ where g is a path formula
- **Path formulas**
  - $\boldsymbol{p} \in \boldsymbol{AP}$
  - $\neg g_1$, $g_1 \vee g_2$, $g_1 \wedge g_2$, $\boldsymbol{X} g_1$, $\boldsymbol{G} g_1$, $g_1 \boldsymbol{U} g_2$, $g_1 \boldsymbol{R} g_2$ where g_1 and g_2 are path formulas

Model Checking of LTL

- Given a Kripke structure M and a LTL formula φ:
  Does $M \models \varphi$?

Model Checking of LTL

- Given a Kripke structure M and a LTL formula φ:
  Does $M \models \varphi$?

- **Automata-based Algorithm**

1. Construct $\neg\varphi$
2. Construct a Büchi automaton $\mathcal{S}_{\neg\varphi}$
3. Translate M to an automaton $\mathcal{A}$.
4. Construct the automaton $\mathcal{B}$ with $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{S}_{\neg\varphi})$
5. If $\mathcal{L}(\mathcal{B}) = \emptyset \Rightarrow M \models \varphi$
6. If $\mathcal{L}(\mathcal{B}) \neq \emptyset \Rightarrow M \not\models \varphi$. A word $v \cdot w^{\omega} \in \mathcal{L}(\mathcal{B})$ is a **counterexample**
   → a trace in M that does not satisfy φ

Plan for Today

- LTL Model Checking
  - Recap / Overview
  - Basics of Automata Theory
    - Finite automata on finite words (Regular automata)
    - Finite automata on infinite words (Büchi automata)
      - Definitions: Automata, word, run, language
      - Deterministic vs non-deterministic automata
      - Intersection of automata
      - Checking emptiness of automata
    - Kripke structures to automata
  - LTL Model-Checking Algorithm

Finite Automata on Finite Words – Regular Automata

- **Definitions**
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- Complement
- Intersection of NFAs

Finite Automata on Finite Words – Regular Automata

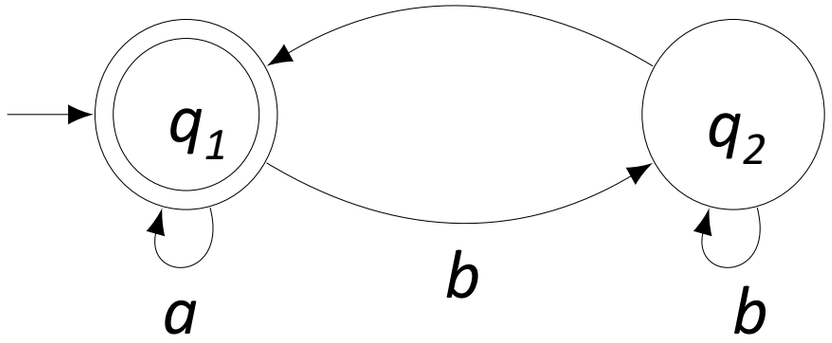
- **Definitions**
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- Complement
- Intersection of NFAs

- $\mathcal{A} = (\Sigma, Q, \Delta, Q^0, F)$
  - Σ is the finite **alphabet**
  - Q is the finite set of **states**
  - $\Delta \subseteq Q \times \Sigma \times Q$ is the **transition relation**
  - Q^0 is the set of **initial states**
  - F is the set of **accepting states**

Finite Automata on Finite Words – Regular Automata

- **Definitions**
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- Complement
- Intersection of NFAs

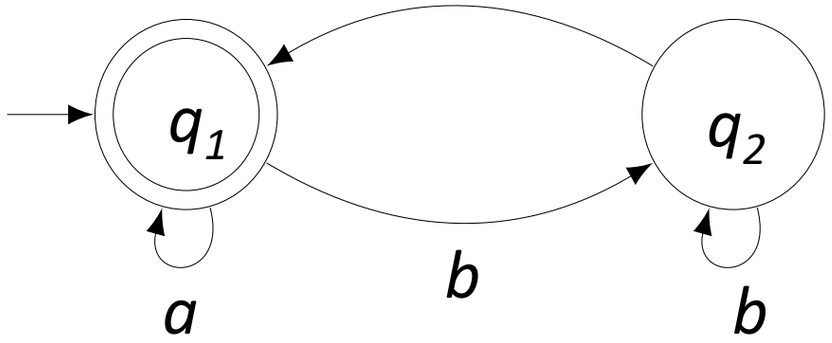
- $\mathcal{A} = (\Sigma, Q, \Delta, Q^0, F)$
  - Σ is the finite **alphabet**
  - Q is the finite set of **states**
  - $\Delta \subseteq Q \times \Sigma \times Q$ is the **transition relation**
  - Q^0 is the set of **initial states**
  - F is the set of **accepting states**



Finite Automata on Finite Words – Regular Automata

- **Definitions**
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- Complement
- Intersection of NFAs

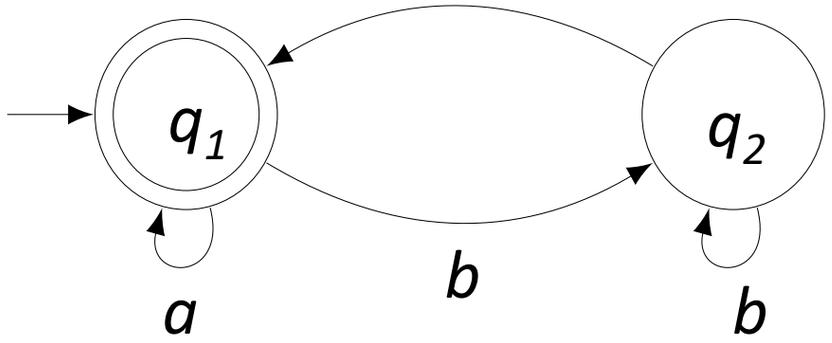
- $\mathcal{A} = (\Sigma, Q, \Delta, Q^0, F)$
  - $\Sigma = \{a, b\}$
  - $Q = \{q_1, q_2\}$
  - $\Delta = \{ (q_1, a, q_1), (q_1, b, q_2), (q_2, a, q_1), (q_2, b, q_2)\}$
  - $Q^0 = \{q_1\}$
  - $F = \{q_1\}$



- **Definitions**
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- Complement
- Intersection of NFAs

Words and Runs on Finite Automata

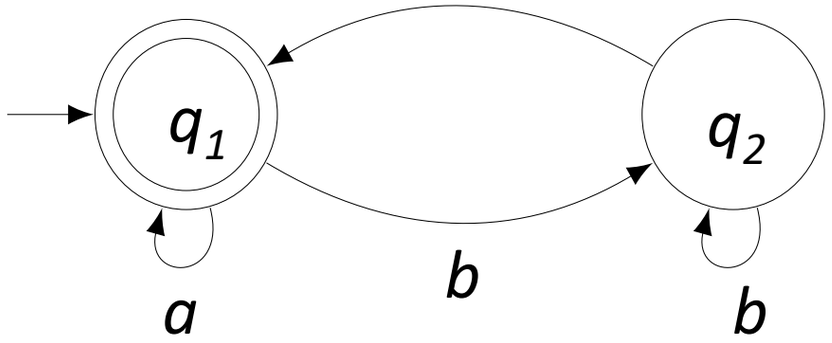
- A **word v** is a **sequence** in $\Sigma^\star$ of length $|v|$
- A **run ρ** is a **path** in the automaton $\mathcal{A}$.



Words and Runs on Finite Automata

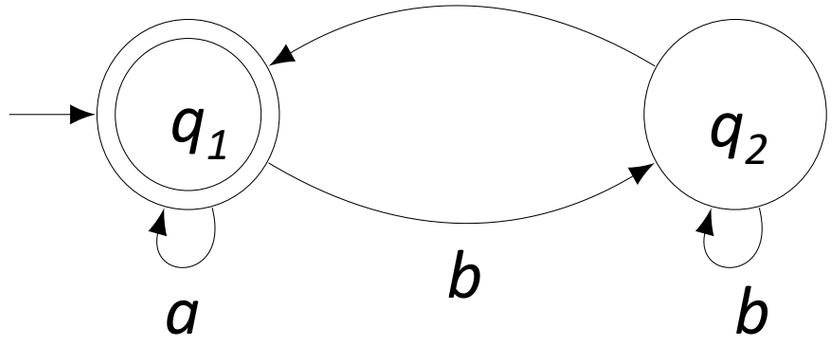
- A **word** v is a **sequence** in $\Sigma^\star$ of length $|v|$
- A **run** ρ is a **path** in the automaton $\mathcal{A}$.

- Given a **word** $v = a_1, a_2, \ldots, a_n$ and automaton $\mathcal{A}$
- → A **run** $\rho = q_0, q_1, \ldots q_n$ of $\mathcal{A}$ is a sequence of states s.t.
  - $q_0 \in Q^0$
  - $\forall i: 0 \leq i \leq n-1: (q_i, a_{i+1} q_{i+1}) \in \Delta$



Accepting Words / Runs / Language of Finite Automata

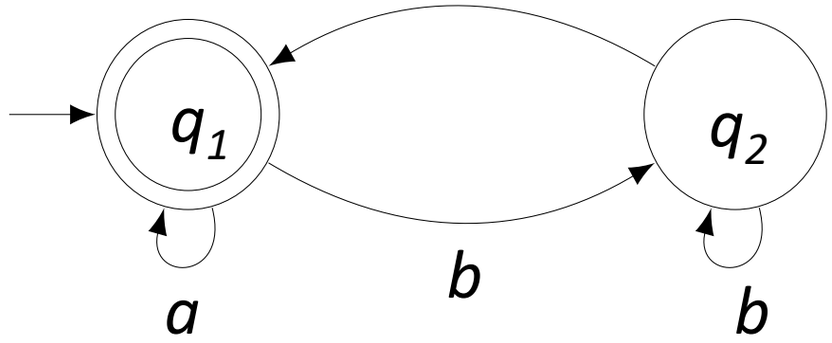
- **Definitions**
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- Complement
- Intersection of NFAs



Accepting Words / Runs / Language of Finite Automata

- **Definitions**
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- Complement
- Intersection of NFAs

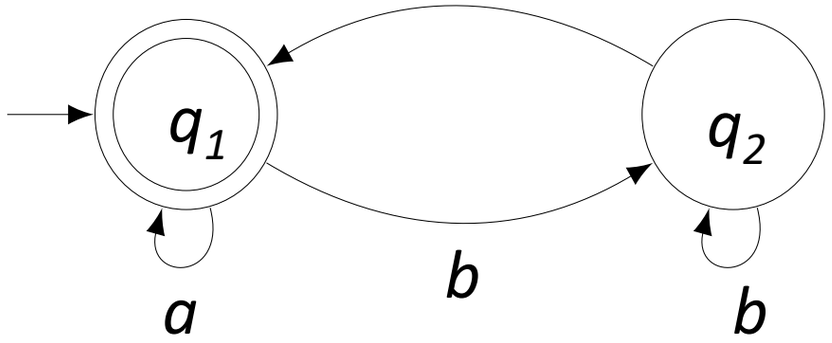
- A **run** $\rho = q_0, q_1, \ldots q_n$ is **accepting** $\Leftrightarrow q_n \in F$



Accepting Words / Runs / Language of Finite Automata

- A **run** $\rho = q_0, q_1, \ldots q_n$ is **accepting** $\Leftrightarrow q_n \in F$

- $\mathcal{A}$ **accepts** a **word** $v = a_1, a_2, \ldots, a_n \Leftrightarrow$
  if there is a corresponding **accepting run** ρ



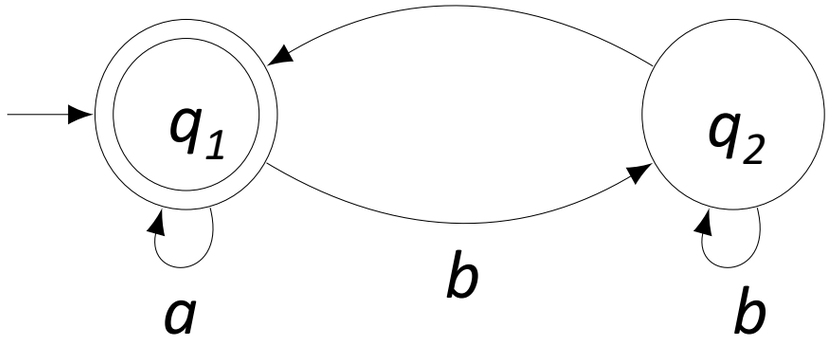
Accepting Words / Runs / Language of Finite Automata

- **Definitions**
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- Complement
- Intersection of NFAs

- A **run** $\rho = q_0, q_1, \ldots q_n$ is **accepting** $\Leftrightarrow q_n \in F$

- $\mathcal{A}$ **accepts** a **word** $v = a_1, a_2, \ldots, a_n \Leftrightarrow$
  if there is a corresponding **accepting run** ρ

- $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^*$ … **Language** of $\mathcal{A}$, is the set of words v that $\mathcal{A}$ accepts.
  - Languages accepted by finite automata are ***regular languages***.



Accepting Words / Runs / Language of Finite Automata

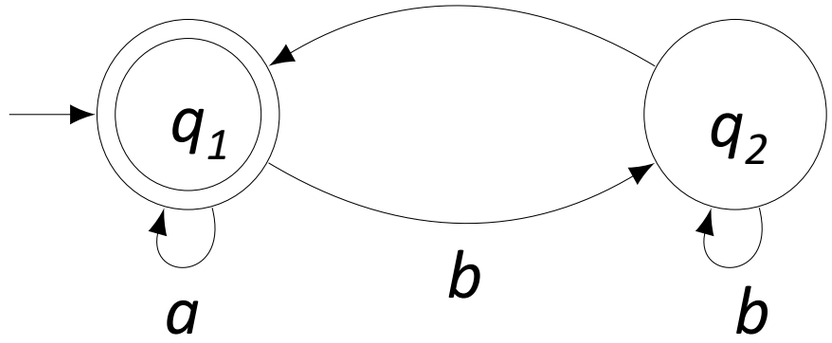
- **Definitions**
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- Complement
- Intersection of NFAs

- A **run** $\rho = q_0, q_1, \ldots q_n$ is **accepting** $\Leftrightarrow q_n \in F$

- $\mathcal{A}$ **accepts** a **word** $v = a_1, a_2, \ldots, a_n \Leftrightarrow$
  if there is a corresponding **accepting run** ρ

- $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^*$ … **Language** of $\mathcal{A}$, is the set of words v that $\mathcal{A}$ accepts.
  - Languages accepted by finite automata are ***regular languages***.

What is $\mathcal{L}(\mathcal{A})$?



Accepting Words / Runs / Language of Finite Automata

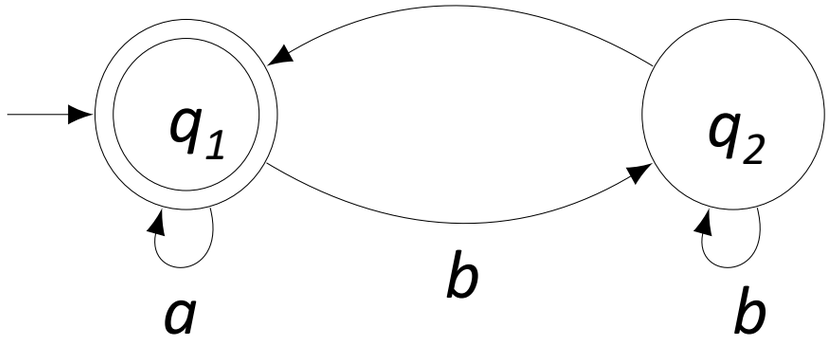
- **Definitions**
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- Complement
- Intersection of NFAs

- A **run** $\rho = q_0, q_1, \ldots q_n$ is **accepting** $\Leftrightarrow q_n \in F$

- $\mathcal{A}$ **accepts** a **word** $v = a_1, a_2, \ldots, a_n \Leftrightarrow$
  if there is a corresponding **accepting run** ρ

- $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^*$ … **Language** of $\mathcal{A}$, is the set of words v that $\mathcal{A}$ accepts.
  - Languages accepted by finite automata are ***regular languages***.

- What is $\mathcal{L}(\mathcal{A})$?
  - $\mathcal{L}(\mathcal{A})$ = {empty word} $\cup$ {all words that end with a}
    = $\{\varepsilon\} \cup \{a, b\}^* a$



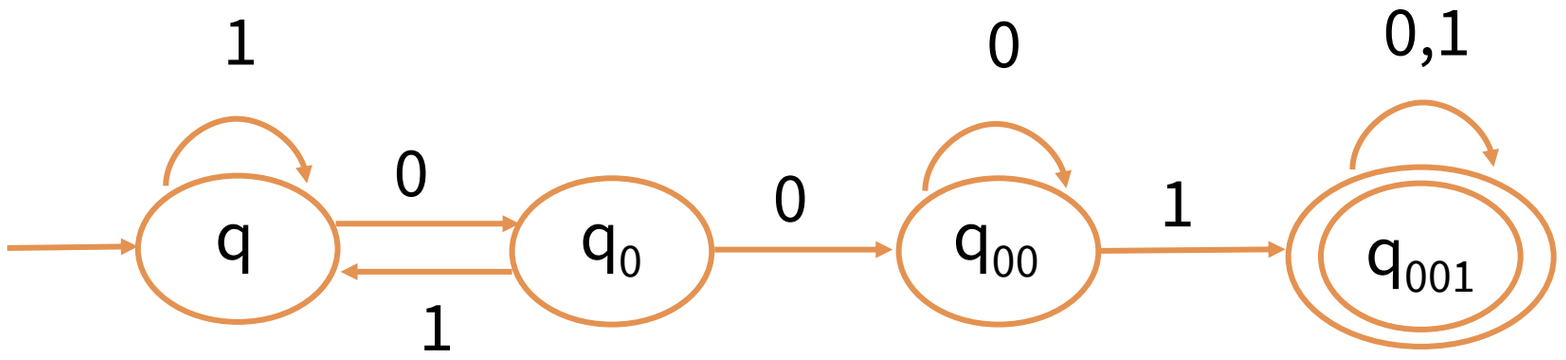
Task: Build a Regular Automata

- **Definitions**
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- Complement
- Intersection of NFAs

- Build an automaton that accepts all and only those strings that contain **001**.

Task: Build a Regular Automata

- Build an automaton that accepts all and only those strings that contain **001**.

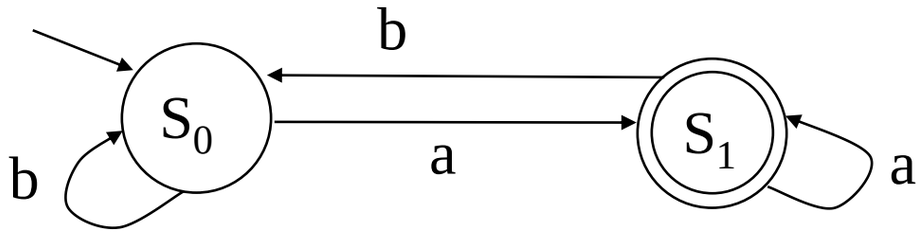


Deterministic and Non-Deterministic Automata

- Definitions
- **DFAs and NFAs**
- NFAs to DFAs (Subset Constr)
- Complement
- Intersection of NFAs

Deterministic and Non-Deterministic Automata

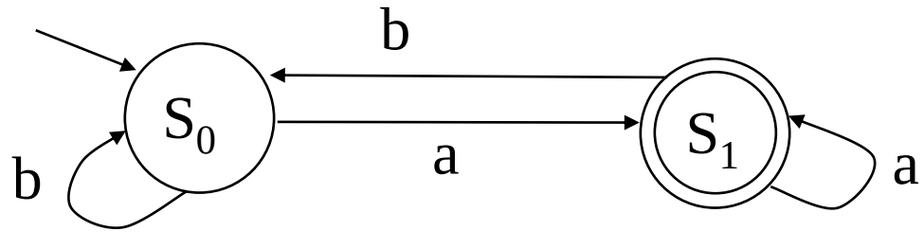
- **Deterministic Finite Automata (DFA)**:
- $\mathcal{A}$ is **deterministic** if Δ is a **function** (one output for each input).
  - $|Q^0| = 1$
  - $\forall q \in Q, \forall a \in \Sigma: |\Delta(q, a)| \leq 1$
- Det. automata have **exactly one** run for each word.



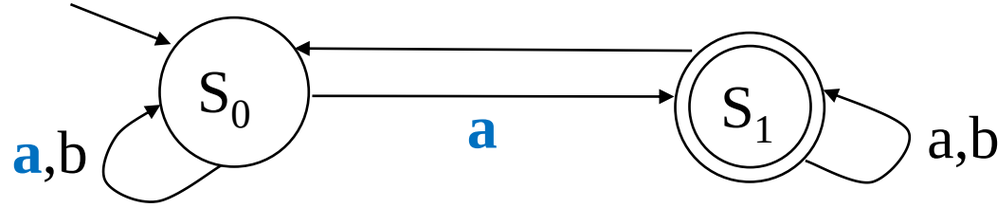
- Definitions
- **DFAs and NFAs**
- NFAs to DFAs (Subset Constr)
- Complement
- Intersection of NFAs

Deterministic and Non-Deterministic Automata

- **Deterministic Finite Automata (DFA)**:
- $\mathcal{A}$ is **deterministic** if Δ is a **function** (one output for each input).
  - $|Q^0| = 1$
  - $\forall q \in Q, \forall a \in \Sigma: |\Delta(q, a)| \leq 1$
- Det. automata have **exactly one** run for each word.



- **Non-deterministic Finite Automata (NFA)**:
  - Can have transitions $(q, a, q'), (q, a, q'') \in \Delta$ and $q'' \neq q'$
  - Can have **ε-transitions** (transitions without a letter)

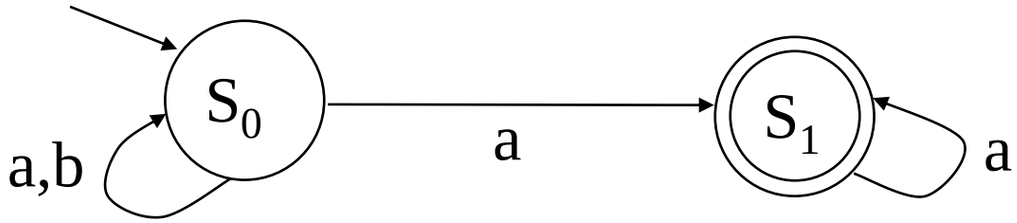


- Definitions
- **DFAs and NFAs**
- NFAs to DFAs (Subset Constr)
- Complement
- Intersection of NFAs

Language of an NFA

- An NFA accepts all words that have a run that ends in an accepting state

What is the language of this automaton?

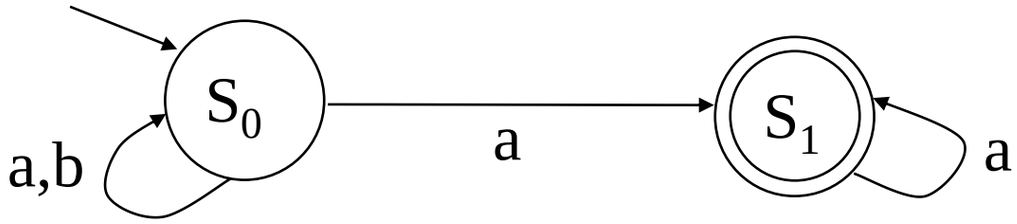


Language of an NFA

- An NFA accepts all words that have a run that ends in an accepting state

- What is the language of this automaton?

$$\mathcal{L}(\mathcal{A}) = \{\text{all words that end with a}\}$$

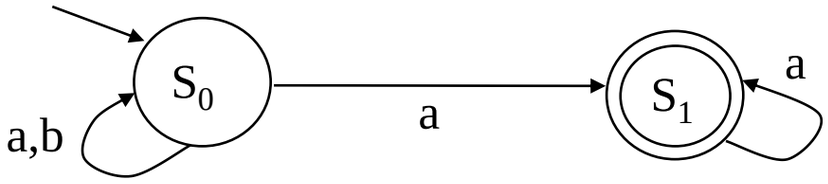


NFA on Finite Words to DFA

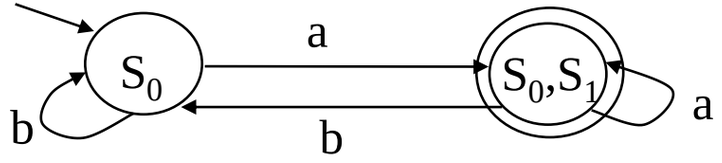
- Definitions
- DFAs and NFAs
- **NFAs to DFAs (Subset Constr)**
- Complement
- Intersection of NFAs

Any non-deterministic finite automata on ***finite words*** can be translated into an equivalent deterministic automaton.

NFA $\mathcal{A}$



Equivalent DFA $\mathcal{A}$'

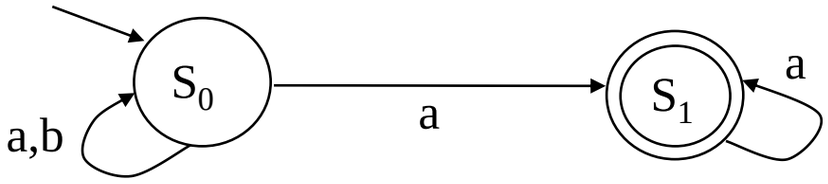


- Definitions
- DFAs and NFAs
- **NFAs to DFAs (Subset Constr)**
- Complement
- Intersection of NFAs

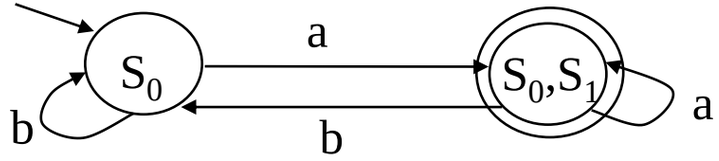
Algorithm: NFA to finite words to DFA

- **Subset-Construction (exponential blow-up)**

NFA $\mathcal{A}$



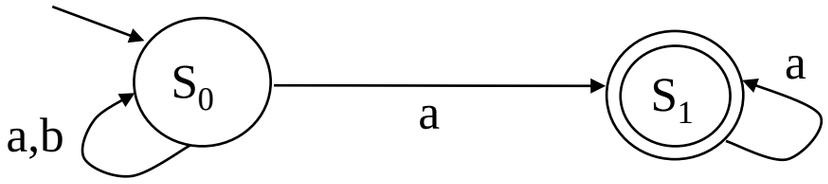
Equivalent DFA $\mathcal{A}$'



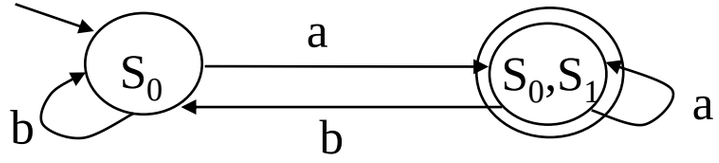
Algorithm: NFA to finite words to DFA

- **Subset-Construction (exponential blow-up)**
  - NFA: $\mathcal{A} = (\Sigma, Q, \Delta, Q^0, F)$
  - DFA: $\mathcal{A}' = (\Sigma, \mathbf{P(Q)}, \Delta', \{Q^0\}, F')$ with P(Q)…powerset of Q

NFA $\mathcal{A}$



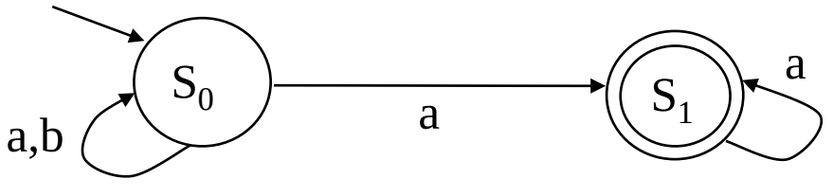
Equivalent DFA $\mathcal{A}'$



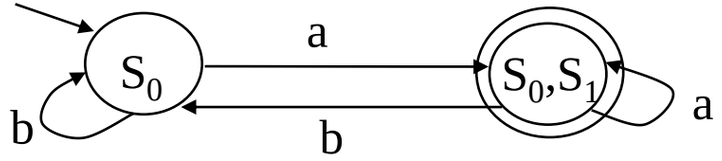
Algorithm: NFA to finite words to DFA

- **Subset-Construction (exponential blow-up)**
  - NFA: $\mathcal{A} = (\Sigma, Q, \Delta, Q^0, F)$
  - DFA: $\mathcal{A}' = (\Sigma, \mathbf{P(Q)}, \Delta', \{Q^0\}, F')$ with P(Q)…powerset of Q
    - Each state in $\mathcal{A}'$ corresponds to the **set of states** in $\mathcal{A}$ that is reached after reading an input sequence
    - Δ': $P(Q) \times \Sigma \rightarrow P(Q)$ where $(Q_1, a, Q_2) \in \Delta'$ if $Q_2 = \bigcup_{q \in Q_1} \{q' | (q, a, q') \in \Delta\}$

NFA $\mathcal{A}$



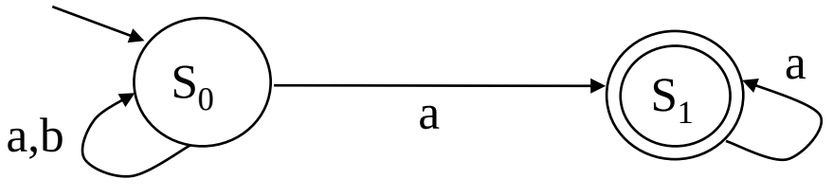
Equivalent DFA $\mathcal{A}'$



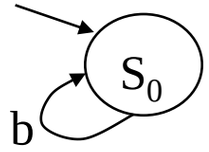
Algorithm: NFA to finite words to DFA

- **Subset-Construction (exponential blow-up)**
  - NFA: $\mathcal{A} = (\Sigma, Q, \Delta, Q^0, F)$
  - DFA: $\mathcal{A}' = (\Sigma, \mathbf{P(Q)}, \Delta', \{Q^0\}, F')$ with P(Q)…powerset of Q
    - Each state in $\mathcal{A}'$ corresponds to the **set of states** in $\mathcal{A}$ that is reached after reading an input sequence
    - Δ': $P(Q) \times \Sigma \rightarrow P(Q)$ where $(Q_1, a, Q_2) \in \Delta'$ if $Q_2 = \bigcup_{q \in Q_1} \{q' | (q, a, q') \in \Delta\}$
- Example:
  $(\{s_0\}, b, \{s_0\}) \in \Delta'$ since $(s_0, b, s_0) \in \Delta$

NFA $\mathcal{A}$



Equivalent DFA $\mathcal{A}'$

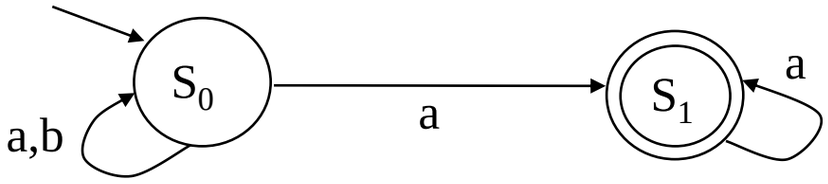


- Definitions
- DFAs and NFAs
- **NFAs to DFAs (Subset Constr)**
- Complement
- Intersection of NFAs

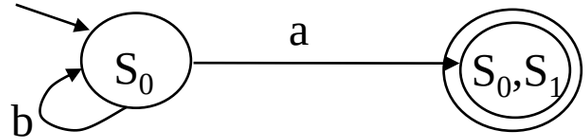
Algorithm: NFA to finite words to DFA

- **Subset-Construction (exponential blow-up)**
  - NFA: $\mathcal{A} = (\Sigma, Q, \Delta, Q^0, F)$
  - DFA: $\mathcal{A}' = (\Sigma, P(Q), \Delta', \{Q^0\}, F')$ with $P(Q)$…powerset of Q
    - Each state in $\mathcal{A}'$ corresponds to the **set of states** in $\mathcal{A}$ that is reached after reading an input sequence
    - Δ': $P(Q) \times \Sigma \rightarrow P(Q)$ where $(Q_1, a, Q_2) \in \Delta'$ if $Q_2 = \bigcup_{q \in Q_1} \{q' | (q, a, q') \in \Delta\}$
- Example:
  $(\{s_0\}, a, \{s_0, s_1\}) \in \Delta'$ since $(s_0, a, s_0) \in \Delta$ and $(s_0, a, s_1) \in \Delta$

NFA $\mathcal{A}$



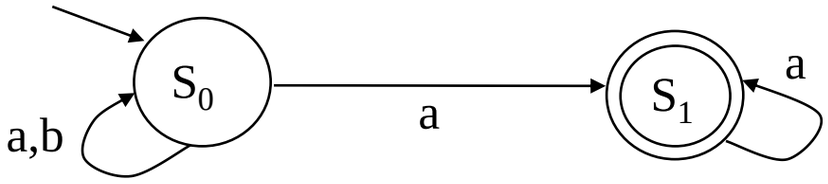
Equivalent DFA $\mathcal{A}'$



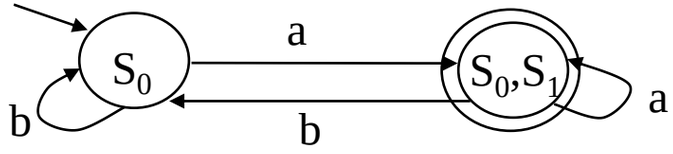
Algorithm: NFA to finite words to DFA

- **Subset-Construction (exponential blow-up)**
  - NFA: $\mathcal{A} = (\Sigma, Q, \Delta, Q^0, F)$
  - DFA: $\mathcal{A}' = (\Sigma, \mathbf{P(Q)}, \Delta', \{Q^0\}, F')$ with P(**Q**)…powerset of **Q**
    - Each state in $\mathcal{A}'$ corresponds to the **set of states** in $\mathcal{A}$ that is reached after reading an input sequence
    - Δ': $P(Q) \times \Sigma \to P(Q)$ where $(Q_1, a, Q_2) \in \Delta'$ if $Q_2 = \bigcup_{q \in Q_1} \{q' | (q, a, q') \in \Delta\}$
- Example:
  $(\{s_0\}, a, \{s_0, s_1\}) \in \Delta'$ since $(s_0, a, s_0) \in \Delta$ and $(s_0, a, s_1) \in \Delta$

NFA $\mathcal{A}$



Equivalent DFA $\mathcal{A}'$

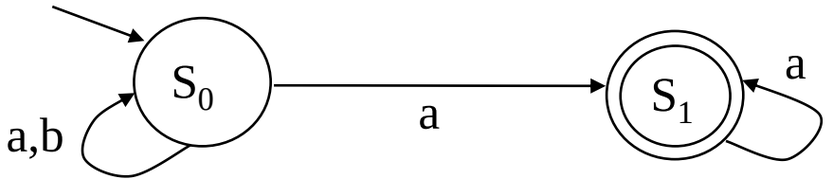


- Definitions
- DFAs and NFAs
- **NFAs to DFAs (Subset Constr)**
- Complement
- Intersection of NFAs

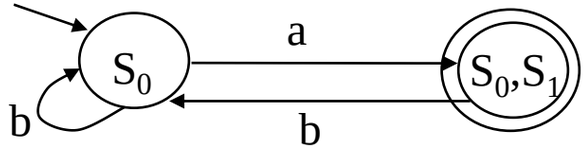
Algorithm: NFA to finite words to DFA

- **Subset-Construction (exponential blow-up)**
  - NFA: $\mathcal{A} = (\Sigma, Q, \Delta, Q^0, F)$
  - DFA: $\mathcal{A}' = (\Sigma, P(Q), \Delta', \{Q^0\}, F')$ with $P(Q)$…powerset of Q
    - Each state in $\mathcal{A}'$ corresponds to the **set of states** in $\mathcal{A}$ that is reached after reading an input sequence
    - Δ': $P(Q) \times \Sigma \to P(Q)$ where $(Q_1, a, Q_2) \in \Delta'$ if $Q_2 = \bigcup_{q \in Q_1} \{q' | (q, a, q') \in \Delta\}$
- Example:
  $(\{s_0, s_1\}, b, \{\boldsymbol{s_0}\}) \in \Delta'$ since $(s_0, b, \boldsymbol{s_0}) \in \Delta$

NFA $\mathcal{A}$



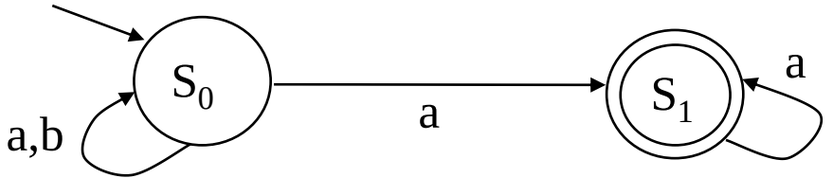
Equivalent DFA $\mathcal{A}'$



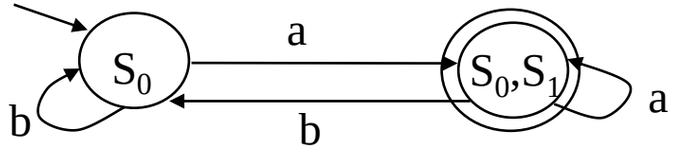
Algorithm: NFA to finite words to DFA

- **Subset-Construction (exponential blow-up)**
  - NFA: $\mathcal{A} = (\Sigma, Q, \Delta, Q^0, F)$
  - DFA: $\mathcal{A}' = (\Sigma, \mathbf{P(Q)}, \Delta', \{Q^0\}, F')$ with P(Q)…powerset of Q
    - Each state in $\mathcal{A}'$ corresponds to the **set of states** in $\mathcal{A}$ that is reached after reading an input sequence
    - Δ': $P(Q) \times \Sigma \rightarrow P(Q)$ where $(Q_1, a, Q_2) \in \Delta'$ if $Q_2 = \bigcup_{q \in Q_1} \{q' | (q, a, q') \in \Delta\}$
- Example:
  $(\{s_0, s_1\}, a, \{\mathbf{s_0}, \mathbf{s_1}\}) \in \Delta'$ since $(s_0, a, \mathbf{s_0}) \in \Delta$ and $(s_0, a, \mathbf{s_1}) \in \Delta$ and $(s_1, a, \mathbf{s_1}) \in \Delta$

NFA $\mathcal{A}$



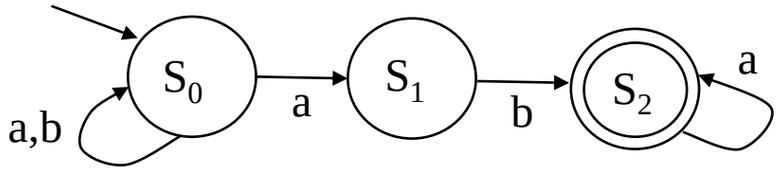
Equivalent DFA $\mathcal{A}'$



Example 1/2: NFA to finite words to DFA

- **Subset-Construction (exponential blow-up)**
  - NFA: $\mathcal{A} = (\Sigma, Q, \Delta, Q^0, F)$
  - DFA: $\mathcal{A}' = (\Sigma, \mathbf{P(Q)}, \Delta', \{Q^0\}, F')$ with P(Q)…powerset of Q
    - Δ': $P(Q) \times \Sigma \rightarrow P(Q)$ where $(Q_1, a, Q_2) \in \Delta'$ if $Q_2 = \bigcup_{q \in Q_1} \{q' | (q, a, q') \in \Delta\}$
    - $F' = \{Q' | Q' \cap F \neq \emptyset\}$

NFA $\mathcal{A}$

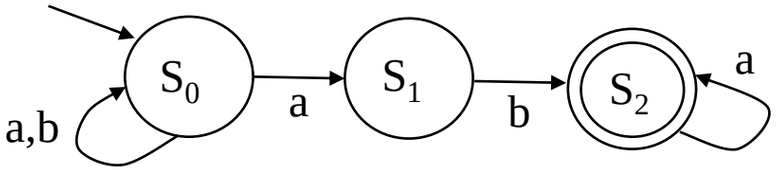


Equivalent DFA $\mathcal{A}'$?

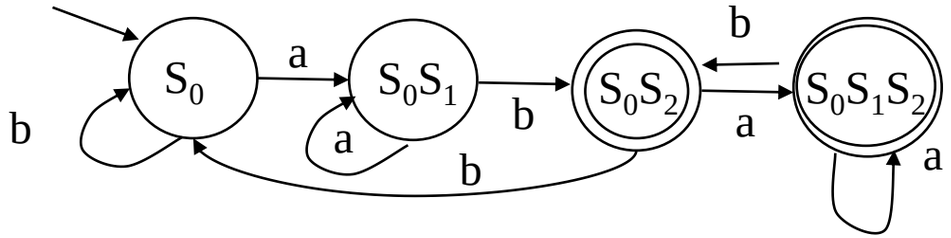
Example 1/2: NFA to finite words to DFA

- **Subset-Construction (exponential blow-up)**
  - NFA: $\mathcal{A} = (\Sigma, Q, \Delta, Q^0, F)$
  - DFA: $\mathcal{A}' = (\Sigma, P(Q), \Delta', \{Q^0\}, F')$ with $P(Q)$…powerset of Q
    - Δ': $P(Q) \times \Sigma \to P(Q)$ where $(Q_1, a, Q_2) \in \Delta'$ if $Q_2 = \bigcup_{q \in Q_1} \{q' | (q, a, q') \in \Delta\}$
    - $F' = \{Q' | Q' \cap F \neq \emptyset\}$

NFA $\mathcal{A}$



Equivalent DFA $\mathcal{A}'$?

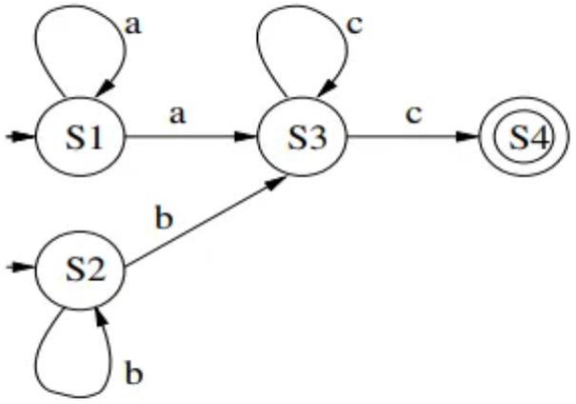


- Definitions
- DFAs and NFAs
- **NFAs to DFAs (Subset Constr)**
- Complement
- Intersection of NFAs

Example 2/2: NFA to finite words to DFA

- **Subset-Construction (exponential blow-up)**
  - NFA: $\mathcal{A} = (\Sigma, Q, \Delta, Q^0, F)$
  - DFA: $\mathcal{A}' = (\Sigma, \mathbf{P(Q)}, \Delta', \{Q^0\}, F')$ with P(Q)…powerset of Q
    - Δ': $P(Q) \times \Sigma \rightarrow P(Q)$ where $(Q_1, a, Q_2) \in \Delta'$ if $Q_2 = \bigcup_{q \in Q_1} \{q' | (q, a, q') \in \Delta\}$
    - $F' = \{Q' | Q' \cap F \neq \emptyset\}$

NFA $\mathcal{A}$



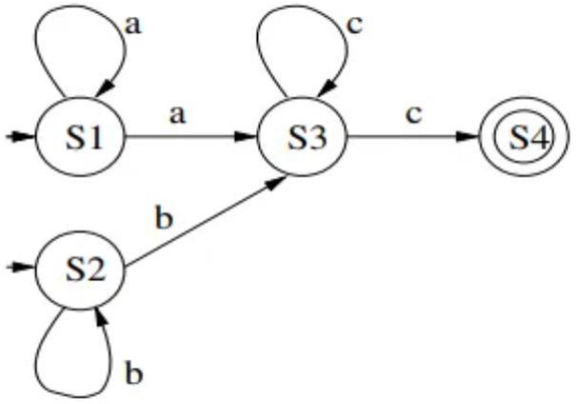
Equivalent DFA $\mathcal{A}'$?

- Definitions
- DFAs and NFAs
- **NFAs to DFAs (Subset Constr)**
- Complement
- Intersection of NFAs

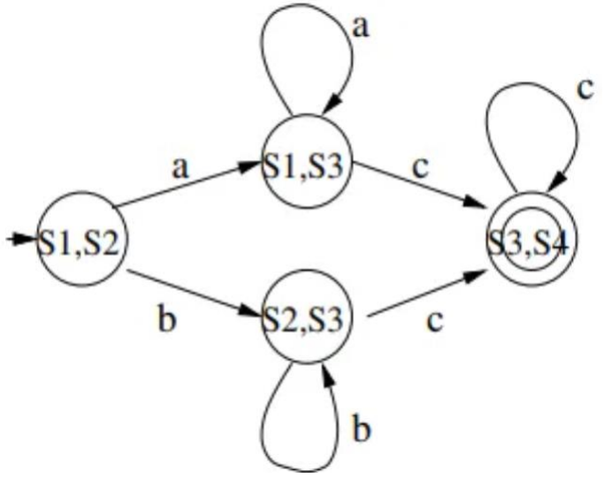
Example 2/2: NFA to finite words to DFA

- **Subset-Construction (exponential blow-up)**
  - NFA: $\mathcal{A} = (\Sigma, Q, \Delta, Q^0, F)$
  - DFA: $\mathcal{A}' = (\Sigma, \mathbf{P(Q)}, \Delta', \{Q^0\}, F')$ with P(Q)…powerset of Q
    - Δ': $P(Q) \times \Sigma \rightarrow P(Q)$ where $(Q_1, a, Q_2) \in \Delta'$ if $Q_2 = \bigcup_{q \in Q_1} \{q' | (q, a, q') \in \Delta\}$
    - $F' = \{Q' | Q' \cap F \neq \emptyset\}$

NFA $\mathcal{A}$



Equivalent DFA $\mathcal{A}'$?

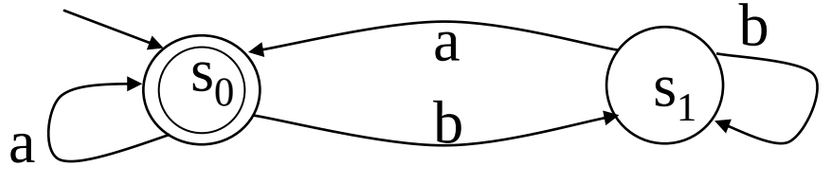


Complement of DFAs

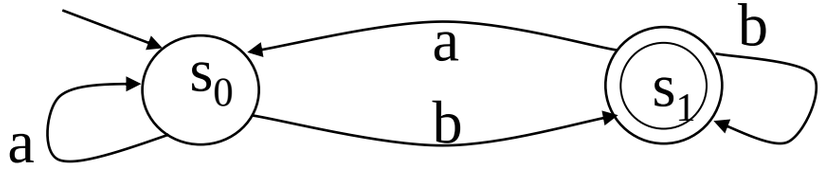
- Definitions
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- **Complement**
- Intersection of NFAs

- The **complement** automaton $\overline{\mathcal{A}}$ **accepts** exactly those words that are **rejected** by $\mathcal{A}$

$\mathcal{A}$



$\overline{\mathcal{A}}$

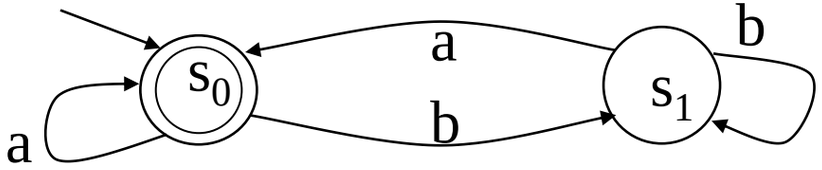


Complement of DFAs

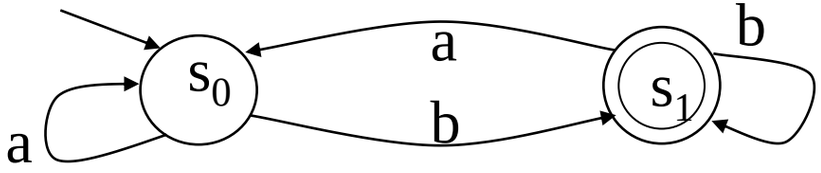
- Definitions
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- **Complement**
- Intersection of NFAs

- The **complement** automaton $\overline{\mathcal{A}}$ **accepts** exactly those words that are **rejected** by $\mathcal{A}$
- Algorithm: Construction of $\overline{\mathcal{A}}$
  - **Substitution** of **accepting** and **non-accepting** states

$\mathcal{A}$



$\overline{\mathcal{A}}$

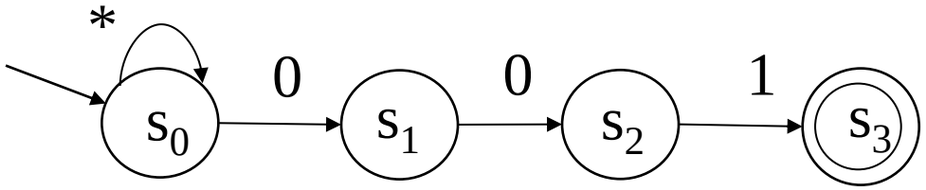


- Definitions
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- **Complement**
- Intersection of NFAs

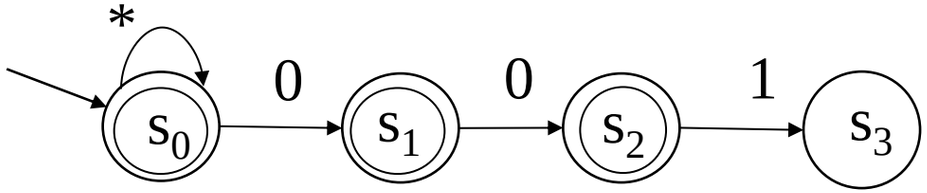
Example: Complement of NFAs

- NFA $\mathcal{A}$ accepts words that **end with 001**
- Is $\mathcal{A}$' the compelent $\overline{\mathcal{A}}$ of $\mathcal{A}$?

$\mathcal{A}$



$\mathcal{A}$'

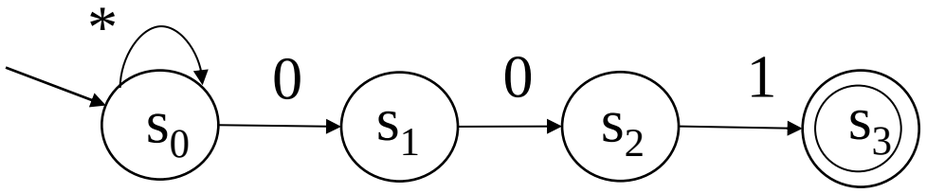


- Definitions
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- **Complement**
- Intersection of NFAs

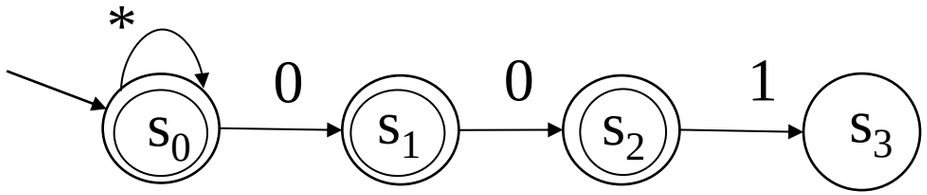
Example: Complement of NFAs

- NFA $\mathcal{A}$ accepts words that **end with 001**
- Is $\mathcal{A}$' the compelent $\overline{\mathcal{A}}$ of $\mathcal{A}$?

$\mathcal{A}$



$\mathcal{A}$'



NO! The language of this automaton is {0,1}*

Algorithm: Complement of NFAs

- Definitions
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- **Complement**
- Intersection of NFAs

- The complement automaton $\overline{\mathcal{A}}$ accepts exactly those words that are rejected by $\mathcal{A}$
- Algorithm to construct $\overline{\mathcal{A}}$?

Algorithm: Complement of NFAs

- Definitions
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- **Complement**
- Intersection of NFAs

- The complement automaton $\overline{\mathcal{A}}$ accepts exactly those words that are rejected by $\mathcal{A}$
- Algorithm to construct $\overline{\mathcal{A}}$?
  1. **Convert NFA to DFA** (make NFA deterministic)
  2. **Substitution** of **accepting** and **non-accepting** states

Intersection of NFAs

- Definitions
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- Complement
- **Intersection of NFAs**

Intersection of NFAs

- Definitions
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- Complement
- **Intersection of NFAs**

- The **intersection** of two languages L_1 and L_2 is $\boldsymbol{L_1 \cap L_2 = \{ w \mid w \in L_1 \; and \; w \in L_2 \}}$

- **Algorithm:** Compute **Product Automaton** $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ s.t. $\boldsymbol{L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)}$

Intersection of NFAs

- Definitions
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- Complement
- **Intersection of NFAs**

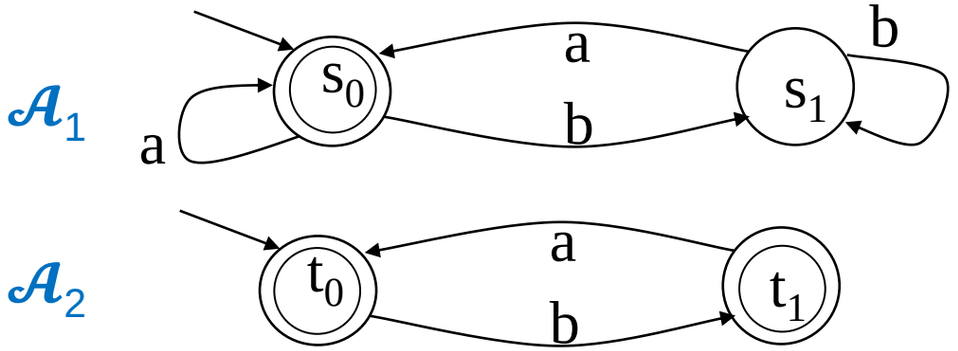
- The **intersection** of two languages L_1 and L_2 is $\boldsymbol{L_1 \cap L_2 = \{ w \mid w \in L_1 \; and \; w \in L_2 \}}$

- **Algorithm:** Compute **Product Automaton** $\boldsymbol{\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2}$ s.t. $\boldsymbol{L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)}$
  - $Q = Q_1 \times Q_2$
  - $\Delta\big((q_1, q_2), a\big) = (\Delta_1(q_1, a), \Delta_2(q_2, a))$
  - $Q^0 = Q_1^0 \times Q_2^0$
  - $(q_1, q_2) \in F \Leftrightarrow q_1 \in F_1$ and $q_2 \in F_2$

Intersection of NFAs

- Definitions
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- Complement
- **Intersection of NFAs**

- **Algorithm:** Compute **Product Automaton** $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ s.t. $\boldsymbol{L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)}$

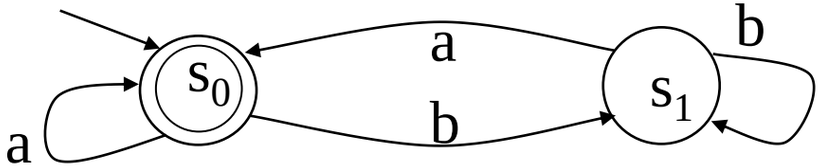


Intersection of NFAs

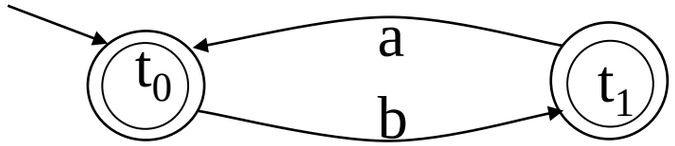
- Definitions
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- Complement
- **Intersection of NFAs**

- **Algorithm:** Compute **Product Automaton** $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ s.t. $\boldsymbol{L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)}$
  - $Q = Q_1 \times Q_2$ ➢ States: (s_0,t_0), (s_0,t_1), (s_1,t_0), (s_1,t_1)
  - $Q^0 = Q_1^0 \times Q_2^0$ ➢ Initial state: (s_0,t_0)

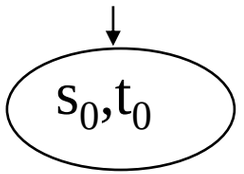
$\mathcal{A}_1$



$\mathcal{A}_2$



$\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$

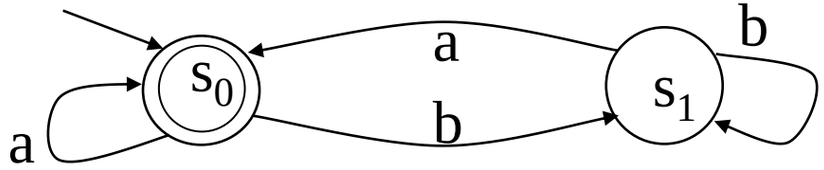


Intersection of NFAs

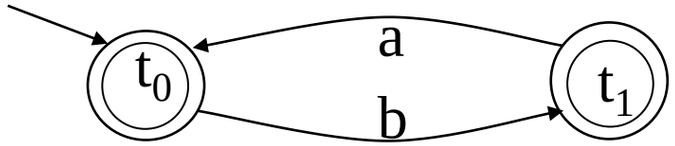
- Definitions
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- Complement
- **Intersection of NFAs**

- **Algorithm:** Compute **Product Automaton** $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ s.t. $\boldsymbol{L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)}$
  - $\Delta\big((q_1, q_2), a\big) = (\Delta_1(q_1, a), \Delta_2(q_2, a))$

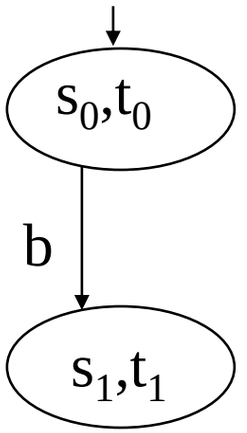
$\mathcal{A}_1$



$\mathcal{A}_2$



$\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$

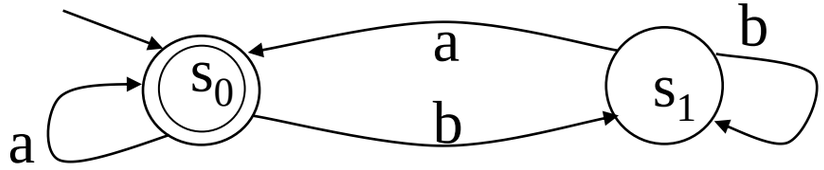


Intersection of NFAs

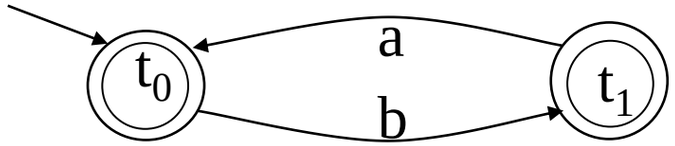
- Definitions
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- Complement
- **Intersection of NFAs**

- **Algorithm:** Compute **Product Automaton** $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ s.t. $\boldsymbol{L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)}$
  - $\Delta\big((q_1, q_2), a\big) = (\Delta_1(q_1, a), \Delta_2(q_2, a))$

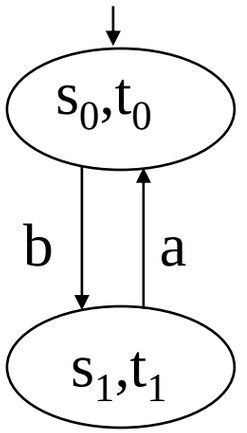
$\mathcal{A}_1$



$\mathcal{A}_2$



$\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$

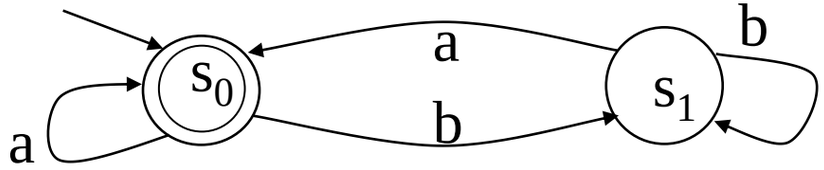


Intersection of NFAs

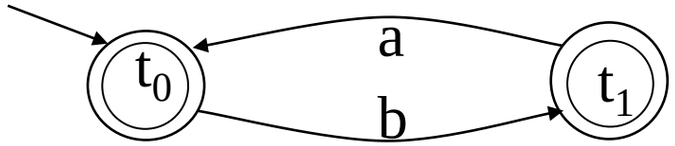
- Definitions
- DFAs and NFAs
- NFAs to DFAs (Subset Constr)
- Complement
- **Intersection of NFAs**

- **Algorithm:** Compute **Product Automaton** $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ s.t. $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$
  - $(q_1, q_2) \in F \Leftrightarrow q_1 \in F_1$ and $q_2 \in F_2$

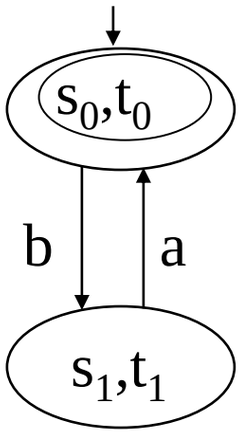
$\mathcal{A}_1$



$\mathcal{A}_2$



$\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$



Plan for Today

- LTL Model Checking
  - Recap / Overview
  - Basics of Automata Theory
    - Finite automata on finite words (Regular automata)
    - **Finite automata on infinite words (Büchi automata)**
      - Definitions: Automata, word, run, language
      - Deterministic vs non-deterministic automata
      - Intersection of automata
      - Checking emptiness of automata
    - Kripke structures to automata
  - LTL Model-Checking Algorithm

Automata on Infinite Words

- **Definitions**
- Det/Nondet. Büchi
- Complementation
- Intersection Büchi
- Emptiness Büchi

- We are interested in reactive systems
  - Designed to not hold during normal execution

- Runs are infinite sequences
  - Words are $v \in \Sigma^{\omega}$, where ω denotes **infinitely many** (i.e., $|v| = \infty$)

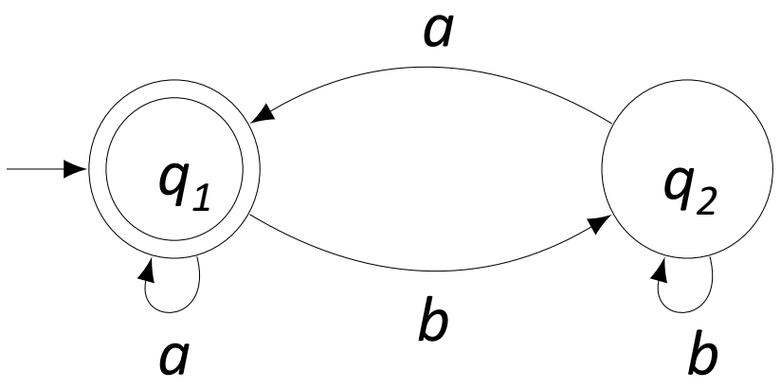
- Languages accepted by finite automata on infinite words are called ***ω-regular languages.***

- **Büchi Automata** → Simplest automata over **infinite words**

- **Definitions**
- Det/Nondet. Büchi
- Complementation
- Intersection Büchi
- Emptiness Büchi

Büchi Automata

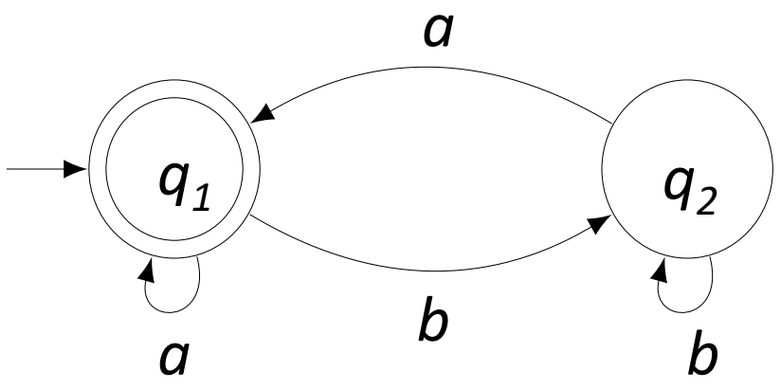
- $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$
  - Σ is the finite **alphabet**
  - Q is the finite set of **states**
  - $\Delta \subseteq Q \times \Sigma \times Q$ is the **transition relation**
  - Q^0 is the set of **initial states**
  - F is the set of **accepting states**



Accepting Words / Runs / Language of Büchi Automata

- **Definitions**
- Det/Nondet. Büchi
- Complementation
- Intersection Büchi
- Emptiness Büchi

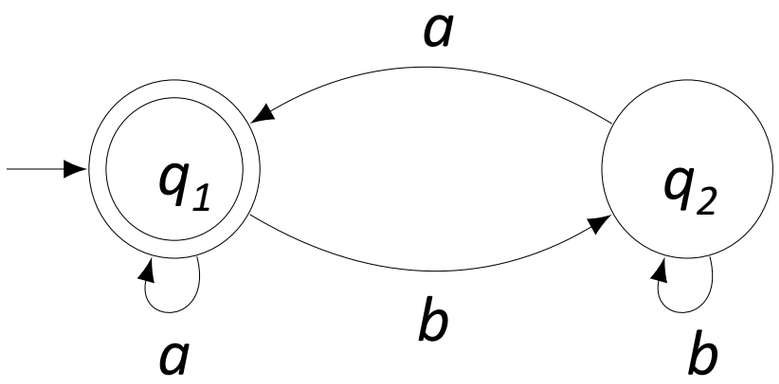
- $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$



Accepting Words / Runs / Language of Büchi Automata

- $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$

- An **infinite run ρ** is **accepting** ⇔ **ρ** visits an **accepting state infinitely** often.
  - $inf(\rho)$ … set of states in ρ that appear **infinitely often**
  - $inf(\rho) \cap F \neq \emptyset$

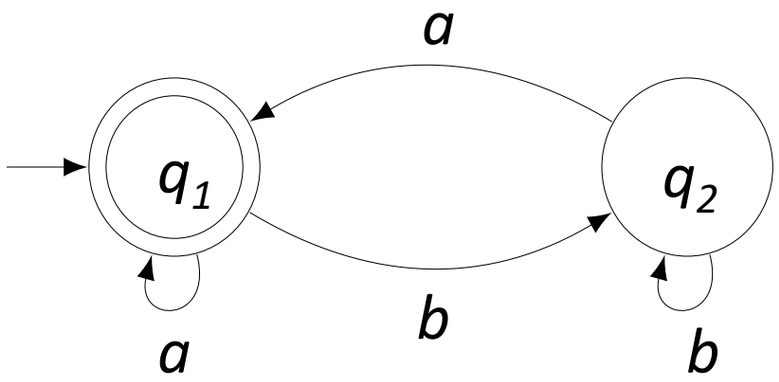


Accepting Words / Runs / Language of Büchi Automata

- $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$

- An **infinite run ρ** is **accepting** ⇔ **ρ** visits an **accepting state infinitely** often.
  - $inf(\rho)$ … set of states in ρ that appear **infinitely often**
  - $inf(\rho) \cap F \neq \emptyset$

- $\mathcal{L}(\mathcal{B}) \subseteq \Sigma^\omega$ is the set of **all infinite words** that $\mathcal{B}$ **accepts**



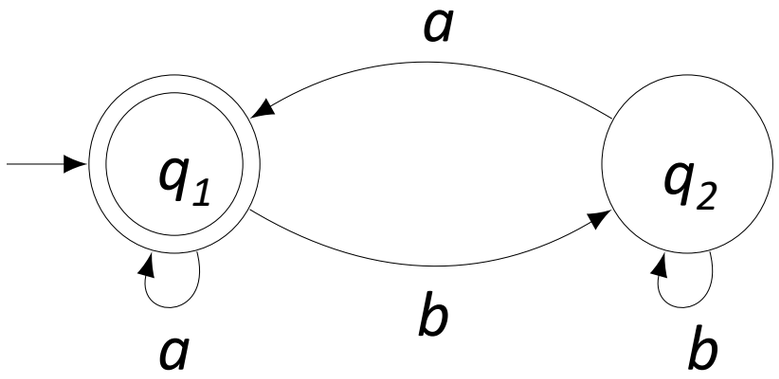
Accepting Words / Runs / Language of Büchi Automata

- **Definitions**
- Det/Nondet. Büchi
- Complementation
- Intersection Büchi
- Emptiness Büchi

- $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$

- An **infinite run ρ** is **accepting** ⇔ **ρ** visits an **accepting state infinitely** often.
  - $inf(\rho)$ … set of states in ρ that appear **infinitely often**
  - $inf(\rho) \cap F \neq \emptyset$

- $\mathcal{L}(\mathcal{B}) \subseteq \Sigma^\omega$ is the set of **all infinite words** that $\mathcal{B}$ **accepts**



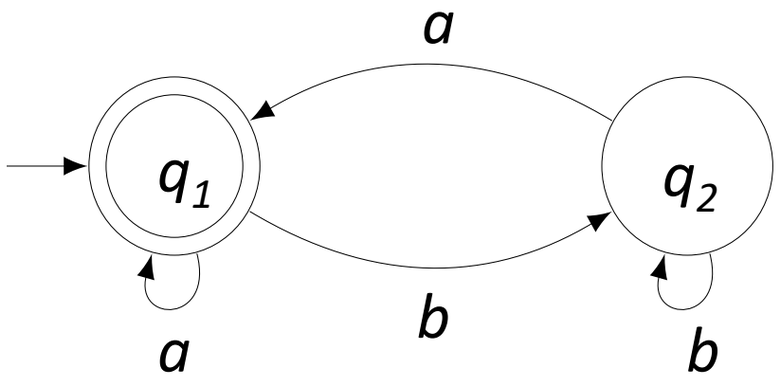
What is the language of this automaton? (In LTL)

Accepting Words / Runs / Language of Büchi Automata

- $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$

- An **infinite run ρ** is **accepting** ⇔ **ρ** visits an **accepting state infinitely** often.
  - $inf(\rho)$ … set of states in ρ that appear **infinitely often**
  - $inf(\rho) \cap F \neq \emptyset$

- $\mathcal{L}(\mathcal{B}) \subseteq \Sigma^\omega$ is the set of **all infinite words** that $\mathcal{B}$ **accepts**



What is the language of this automaton?

$\mathcal{L}(\mathcal{B})$ = {words with infinitely many a}
or
$\mathcal{L}(\mathcal{B}) = (\{a, b\}^\star a)^\omega$ or in LTL: $\boldsymbol{GF(a)}$

Deterministic /Non-Deterministic Büchi Automata

- Definitions
- **Det/Nondet. Büchi**
- Complementation
- Intersection Büchi
- Emptiness Büchi

Deterministic Büchi automata are **strictly less expressive** than nondeterministic Büchi automata.

- → not every nondeterministic Büchi automaton has an equivalent deterministic Büchi one.

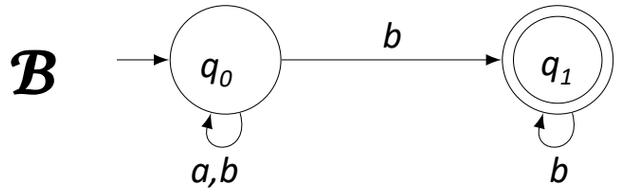
Deterministic /Non-Deterministic Büchi Automata

- Definitions
- **Det/Nondet. Büchi**
- Complementation
- Intersection Büchi
- Emptiness Büchi

Deterministic Büchi automata are **strictly less expressive** than nondeterministic Büchi automata.

- → not every nondeterministic Büchi automaton has an equivalent deterministic Büchi one.

- **Proof Idea:** (details see book)
  - **Non-det**. Büchi automaton $\mathcal{B}$
  - $\mathcal{L}(\mathcal{B})$ = {words with a **finitely** many a} in LTL: FGb



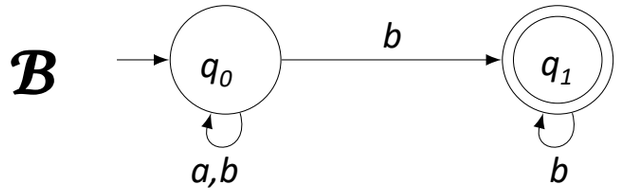
Deterministic /Non-Deterministic Büchi Automata

Deterministic Büchi automata are **strictly less expressive** than nondeterministic Büchi automata.

- → not every nondeterministic Büchi automaton has an equivalent deterministic Büchi one.

- **Proof Idea:** (details see book)

  - **Non-det.** Büchi automaton $\mathcal{B}$
  - $\mathcal{L}(\mathcal{B})$ = {words with a **finitely** many a} in LTL: FGb



  - The proof shows that there is no **deterministic** Büchi automaton for "**finitely many**"

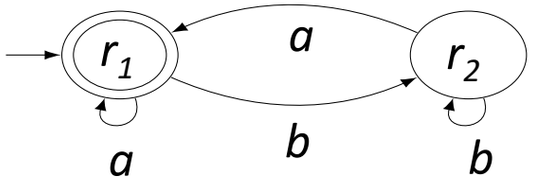
Deterministic /Non-Deterministic Büchi Automata

- Definitions
- Det/Nondet. Büchi
- **Complementation**
- Intersection Büchi
- Emptiness Büchi

Deterministic Büchi automata are **not closed under complementation.**

- Why?
- Hint: Automata below

$\mathcal{A}$:



Deterministic /Non-Deterministic Büchi Automata

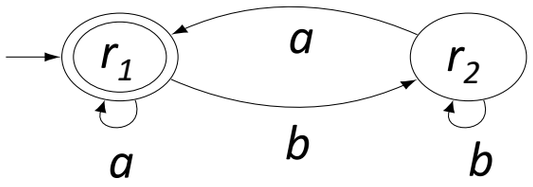
- Definitions
- Det/Nondet. Büchi
- **Complementation**
- Intersection Büchi
- Emptiness Büchi

Deterministic Büchi automata are **not closed under complementation.**

- **Proof:**
  - Deterministic Büchi automaton $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) = \{$words with **infinitely** many a$\}$.
  - $\overline{\mathcal{A}}$ **accepts** those words that are **rejected** by $\mathcal{A}$ → $\mathcal{L}(\overline{\mathcal{A}}) = \{$words with **finitely** many a$\}$.

  - → There there is no **deterministic** Büchi automaton for "**finitely many**" (see Theorem before). □

$\mathcal{A}$:



Deterministic /Non-Deterministic Büchi Automata

- Definitions
- Det/Nondet. Büchi
- **Complementation**
- Intersection Büchi
- Emptiness Büchi

Non-Deterministic Büchi automata are **closed under complementation.**

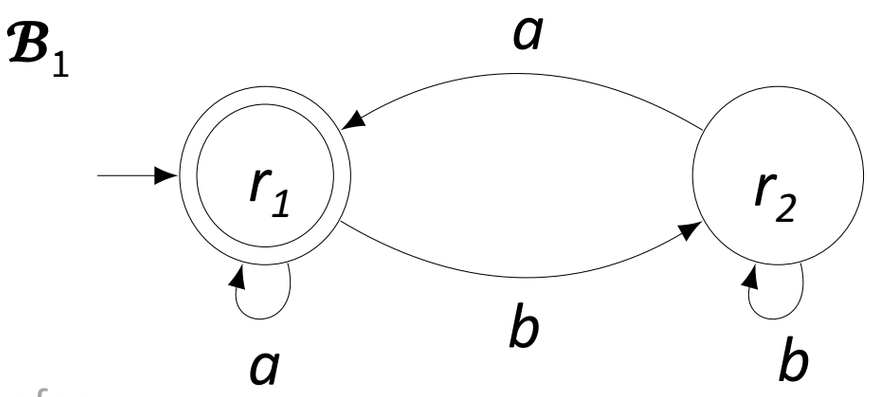
- The construction is **very complicated**.
  - We will not discuss it here. Algorithms try to avoid it.

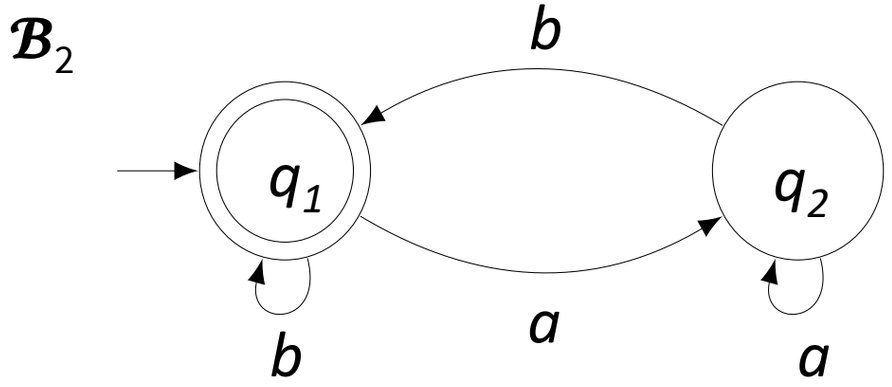
- **Büchi** showed an algorithm for complementation that is **double exponential** in the size n of the automaton.
- **Safra** proved that it can be done by $2^{O(n \log n)}$

Intersection of Büchi Automata

- Definitions
- Det/Nondet. Büchi
- Complementation
- **Intersection Büchi**
- Emptiness Büchi

- What is $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$?

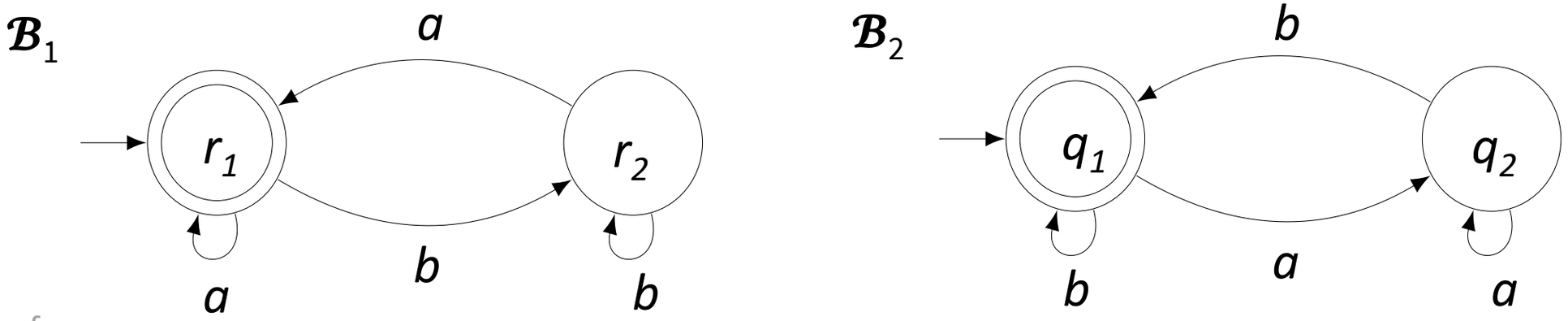




Intersection of Büchi Automata

- Definitions
- Det/Nondet. Büchi
- Complementation
- **Intersection Büchi**
- Emptiness Büchi

- $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ = {words with infinitely many a **and** infinitely many b}

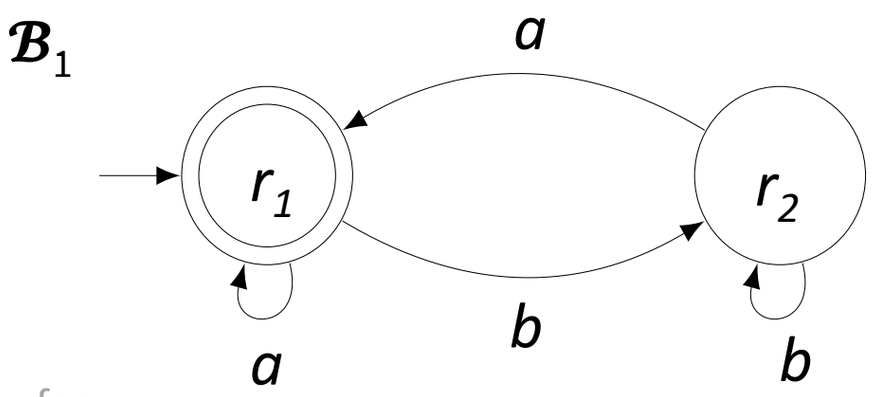


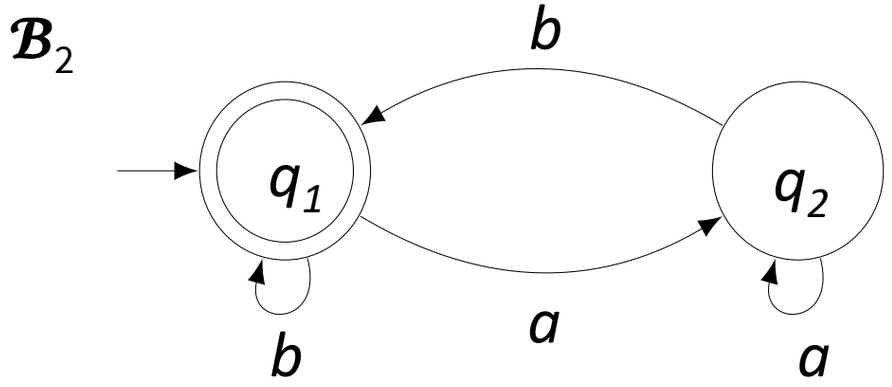
Intersection of Büchi Automata

- Definitions
- Det/Nondet. Büchi
- Complementation
- **Intersection Büchi**
- Emptiness Büchi

- $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ = {words with infinitely many a **and** infinitely many b}

- A standard intersection does not work –automaton has no accepting states!



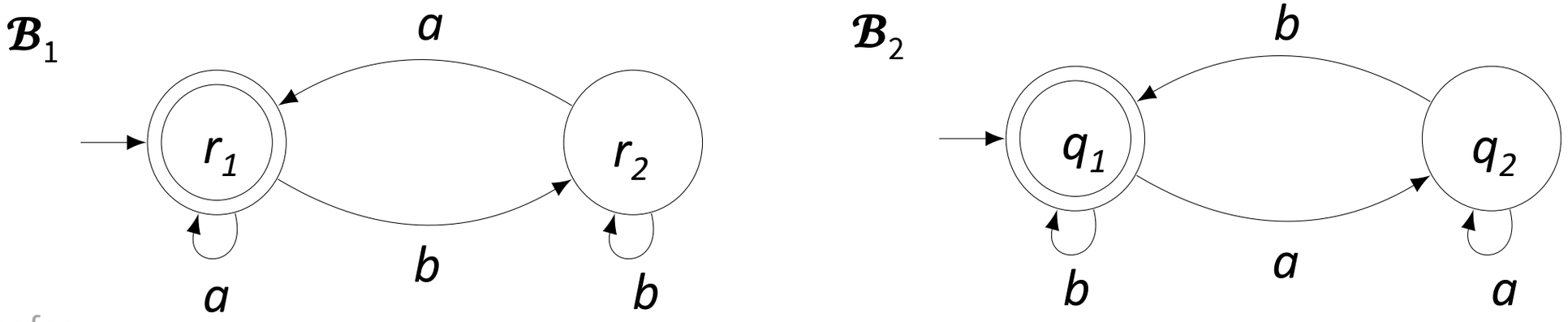


Intersection of Büchi Automata

- Definitions
- Det/Nondet. Büchi
- Complementation
- **Intersection Büchi**
- Emptiness Büchi

- $L(\mathcal{A}_1) \cap L(\mathcal{A}_2) =$ {words with infinitely many a **and** infinitely many b}

- A standard intersection does not work –automaton has no accepting states!
- **Solution: Introduce counter!**

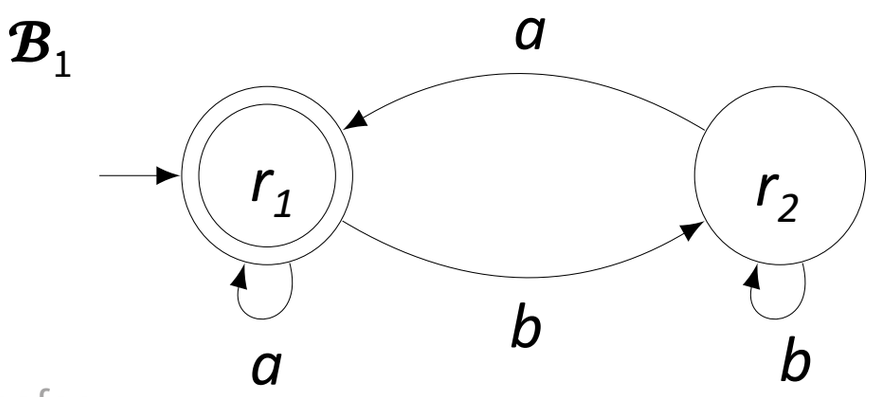


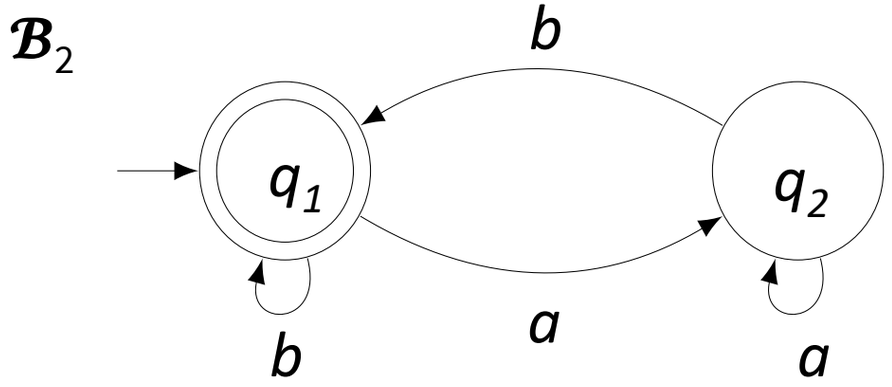
Intersection of Büchi Automata

- Definitions
- Det/Nondet. Büchi
- Complementation
- **Intersection Büchi**
- Emptiness Büchi

- $L(\mathcal{A}_1) \cap L(\mathcal{A}_2) =$ {words with infinitely many a **and** infinitely many b}

- A standard intersection does not work –automaton has no accepting states!
- **Solution: Introduce counter!**





- Definitions
- Det/Nondet. Büchi
- Complementation
- **Intersection Büchi**
- Emptiness Büchi

Algorithm: Intersection of Büchi Automata

- Given $\mathcal{B}_1 = (\mathbf{\Sigma}, \mathbf{Q}_1, \mathbf{\Delta}_1, \mathbf{Q}_1{}^0, \mathbf{F}_1)$ and $\mathcal{B}_2 = (\mathbf{\Sigma}, \mathbf{Q}_2, \mathbf{\Delta}_2, \mathbf{Q}_2{}^0, \mathbf{F}_2)$
- $\mathcal{B} = (\mathbf{\Sigma}, \mathbf{Q}, \mathbf{\Delta}, \mathbf{Q}^0, \mathbf{F})$ s.t. $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$ is defined as follows:

- Definitions
- Det/Nondet. Büchi
- Complementation
- **Intersection Büchi**
- Emptiness Büchi

Algorithm: Intersection of Büchi Automata

- Given $\mathcal{B}_1 = (\Sigma, Q_1, \Delta_1, Q_1^0, F_1)$ and $\mathcal{B}_2 = (\Sigma, Q_2, \Delta_2, Q_2^0, F_2)$
- $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$ s.t. $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$ is defined as follows:
  - $Q = Q_1 \times Q_2 \times \{0, 1, 2\}$
  - $Q^0 = Q_1^0 \times Q_2^0 \times \{0\}$
  - $F = Q_1 \times Q_2 \times \{2\}$

- Definitions
- Det/Nondet. Büchi
- Complementation
- **Intersection Büchi**
- Emptiness Büchi

Algorithm: Intersection of Büchi Automata

- Given $\mathcal{B}_1 = (\Sigma, Q_1, \Delta_1, Q_1^0, F_1)$ and $\mathcal{B}_2 = (\Sigma, Q_2, \Delta_2, Q_2^0, F_2)$
- $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$ s.t. $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$ is defined as follows:
  - $Q = Q_1 \times Q_2 \times \{0, 1, 2\}$
  - $Q^0 = Q_1^0 \times Q_2^0 \times \{0\}$
  - $F = Q_1 \times Q_2 \times \{2\}$
  - $((q_1,q_2,x), a, (q'_1,q'_2,x')) \in \Delta \Leftrightarrow$
    1. $(q_1,a,q'_1) \in \Delta_1$ and $(q_2,a,q'_2) \in \Delta_2$ and
    2. If $x=0$ and $q'_1 \in F_1$ then $x'=1$
       If $x=1$ and $q'_2 \in F_2$ then $x'=2$
       If $x=2$ then $x'=0$
       Else, $x'=x$

Algorithm: Intersection of Büchi Automata

- Definitions
- Det/Nondet. Büchi
- Complementation
- **Intersection Büchi**
- Emptiness Büchi

- Given $\mathcal{B}_1 = (\Sigma, Q_1, \Delta_1, Q_1^0, F_1)$ and $\mathcal{B}_2 = (\Sigma, Q_2, \Delta_2, Q_2^0, F_2)$
- $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$ s.t. $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$ is defined as follows:
  - $Q = Q_1 \times Q_2 \times \{0, 1, 2\}$
  - $Q^0 = Q_1^0 \times Q_2^0 \times \{0\}$
  - $F = Q_1 \times Q_2 \times \{2\}$
  - $((q_1,q_2,x), a, (q'_1,q'_2,x')) \in \Delta \Leftrightarrow$
    1. $(q_1,a,q'_1) \in \Delta_1$ and $(q_2,a,q'_2) \in \Delta_2$ and
    2. If $x=0$ and $q'_1 \in F_1$ then $x'=1$
       If $x=1$ and $q'_2 \in F_2$ then $x'=2$
       If $x=2$ then $x'=0$
       Else, $x'=x$

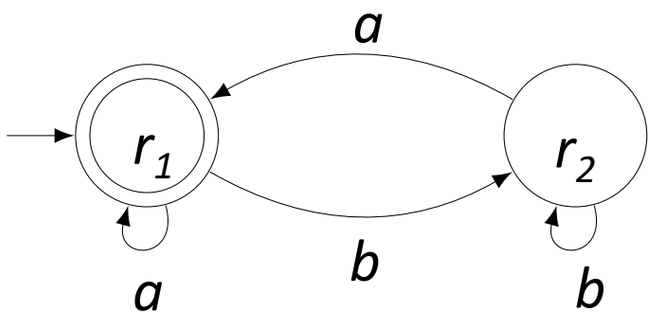
Intuition:
$x=0$ … waiting for $s \in F_1$
$x=1$ … waiting for $s \in F_2$

If some s with $x=2$ is visited inf often, then states from F_1 and states from F_2 have been visited inf often.

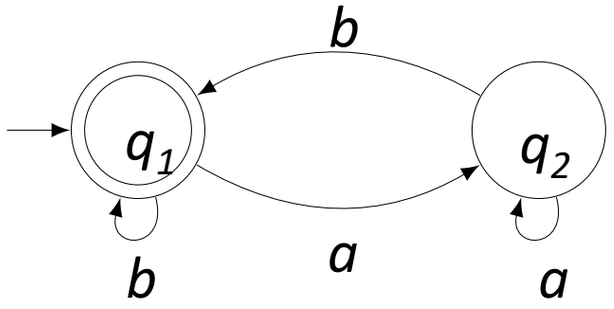
Example: Intersection of Büchi Automata

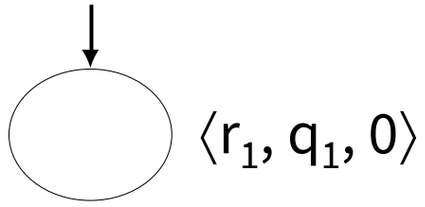
- Compute $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$ s.t. $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$
  - First copy: waits for $s \in F_1$
  - Second copy: waits for $s \in F_2$
  - Third copy: all states are accepting

$\mathcal{B}_1$



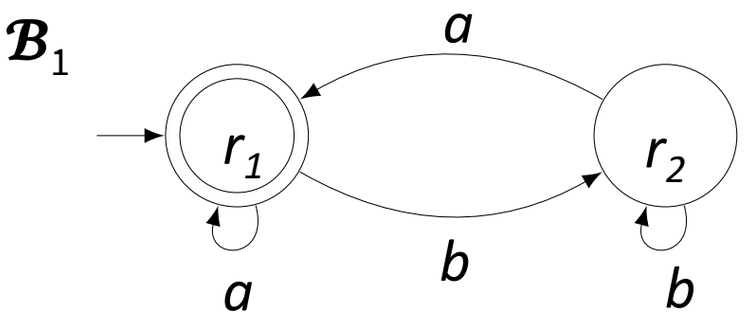
$\mathcal{B}_2$

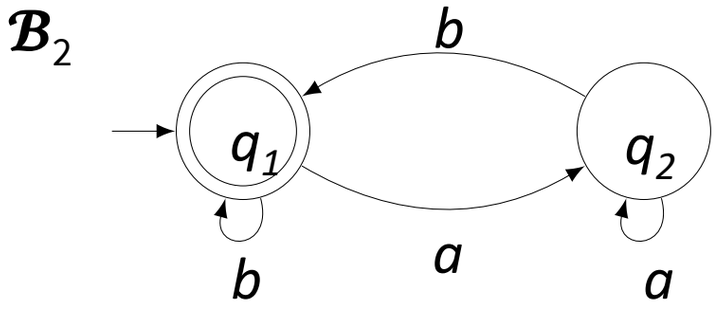


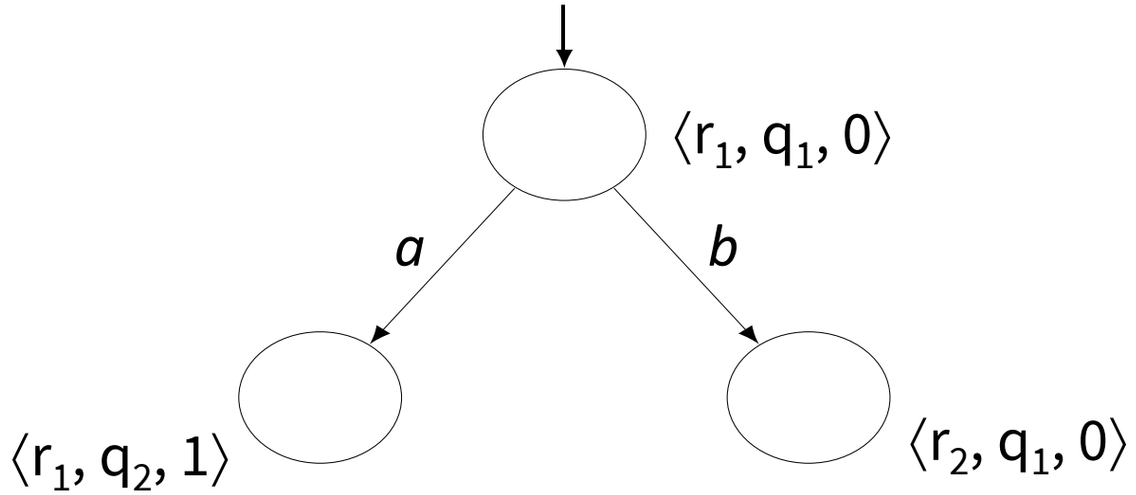


Example: Intersection of Büchi Automata

- Compute $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$ s.t. $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$
  - First copy: waits for $s \in F_1$
  - Second copy: waits for $s \in F_2$
  - Third copy: all states are accepting

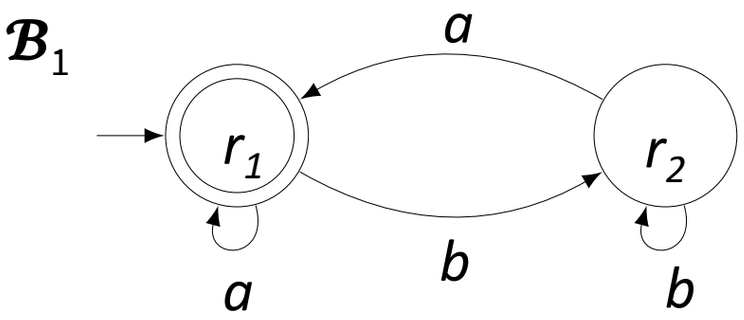


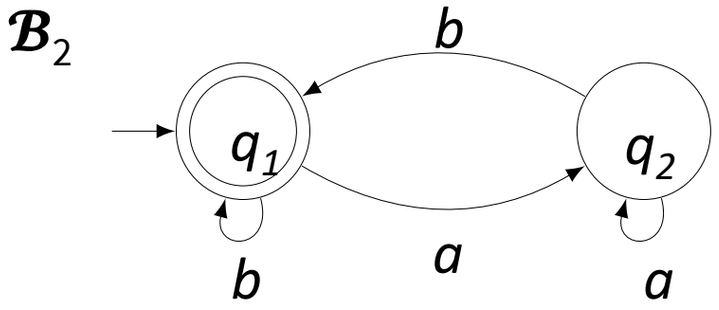


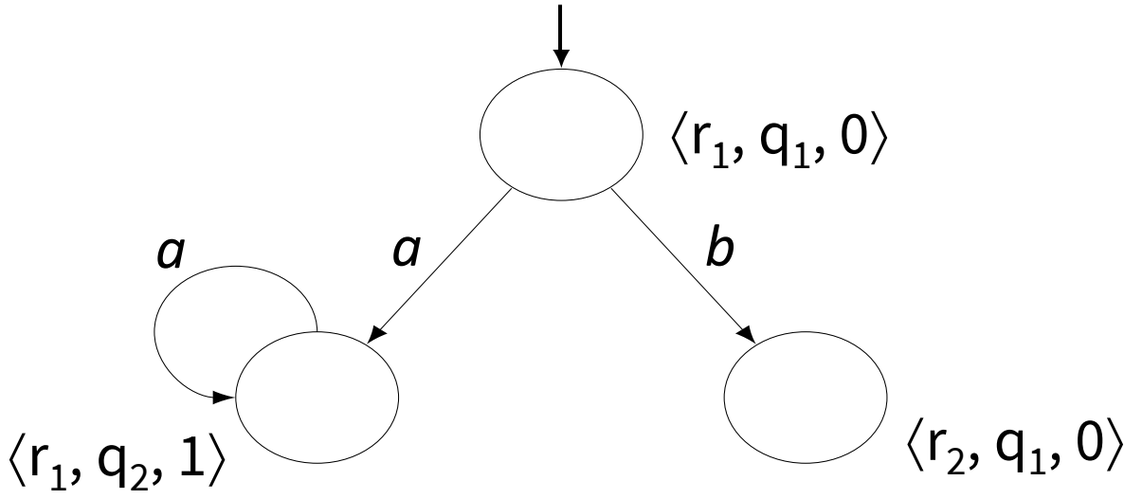


Example: Intersection of Büchi Automata

- Compute $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$ s.t. $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$
  - First copy: waits for $s \in F_1$
  - Second copy: waits for $s \in F_2$
  - Third copy: all states are accepting

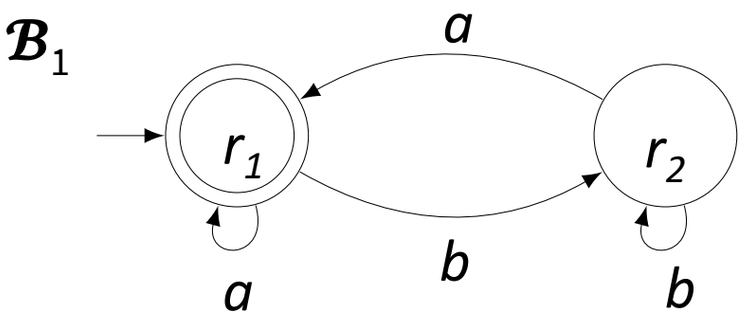


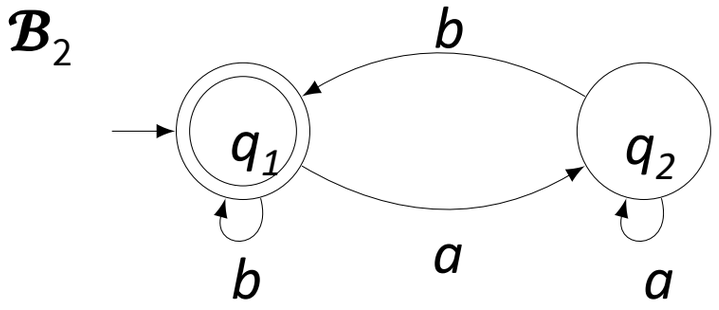


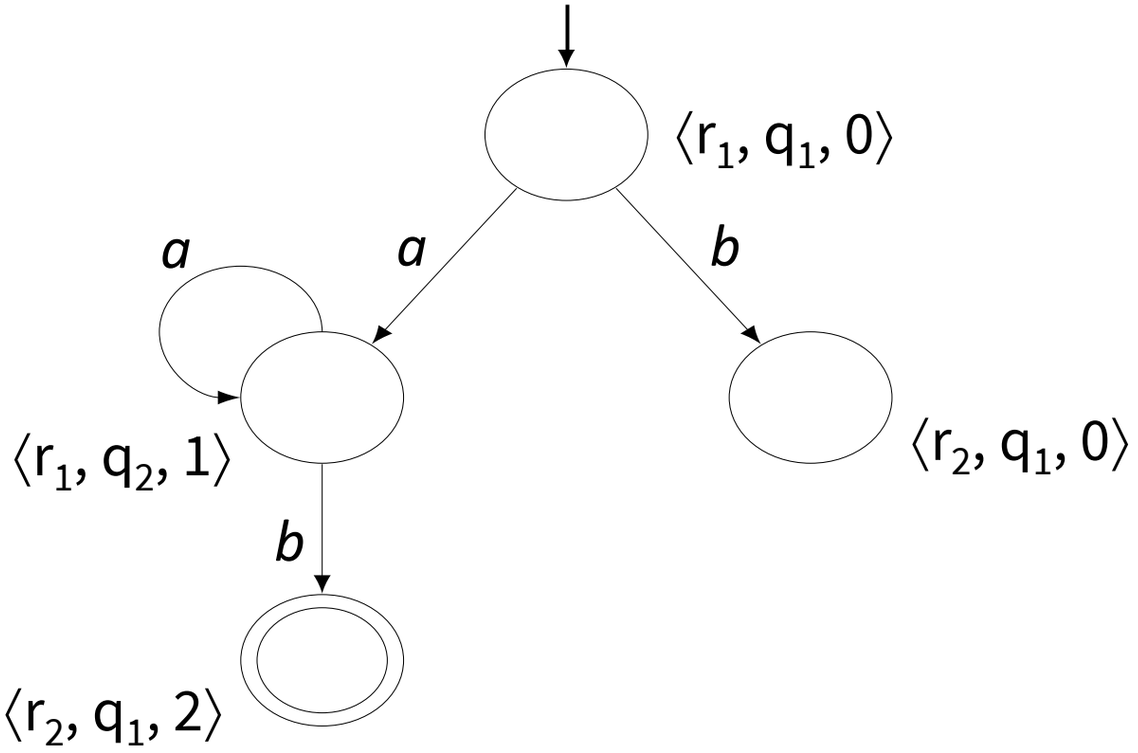


Example: Intersection of Büchi Automata

- Compute $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$ s.t. $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$
  - First copy: waits for $s \in F_1$
  - Second copy: waits for $s \in F_2$
  - Third copy: all states are accepting

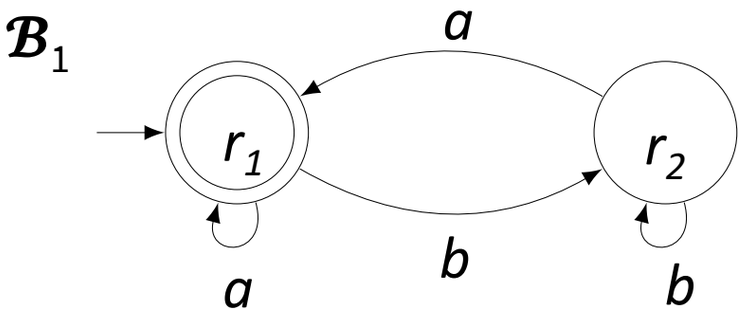


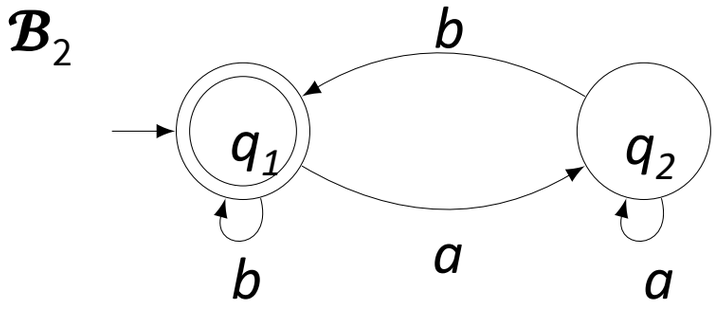


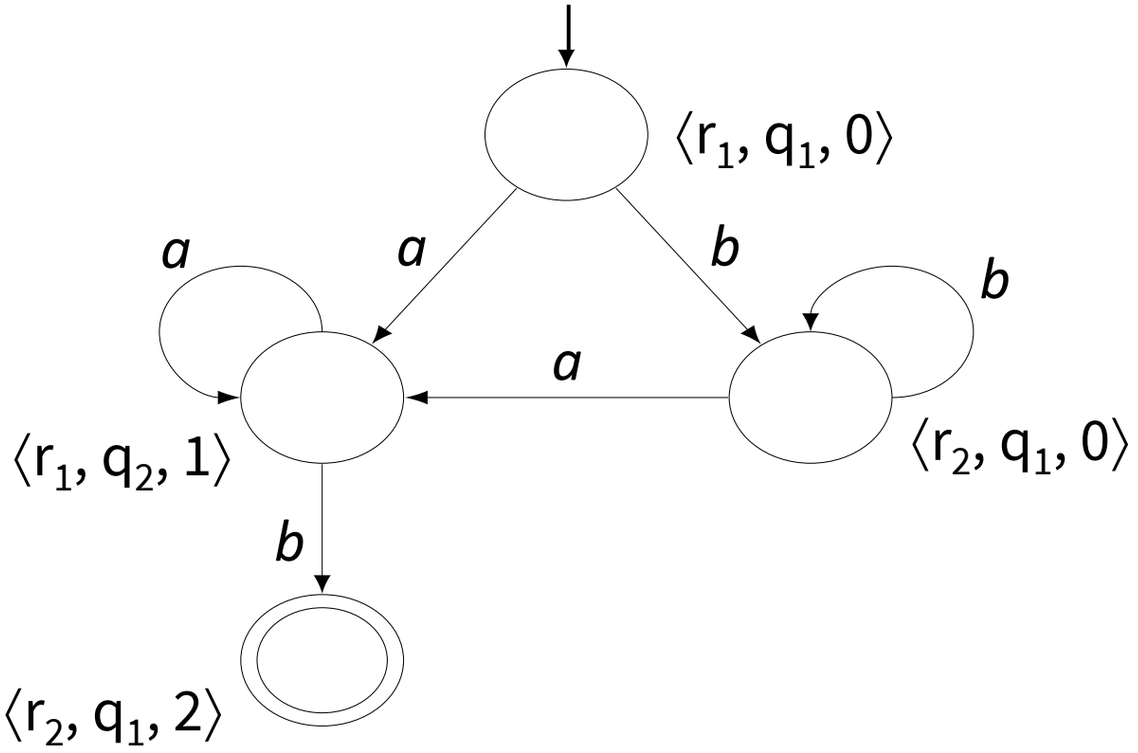


Example: Intersection of Büchi Automata

- Compute $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$ s.t. $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$
  - First copy: waits for $s \in F_1$
  - Second copy: waits for $s \in F_2$
  - Third copy: all states are accepting

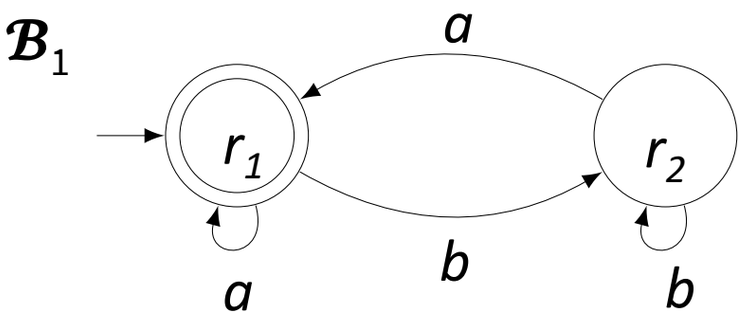


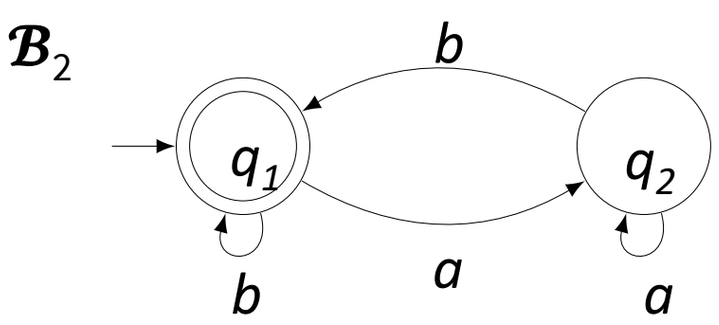


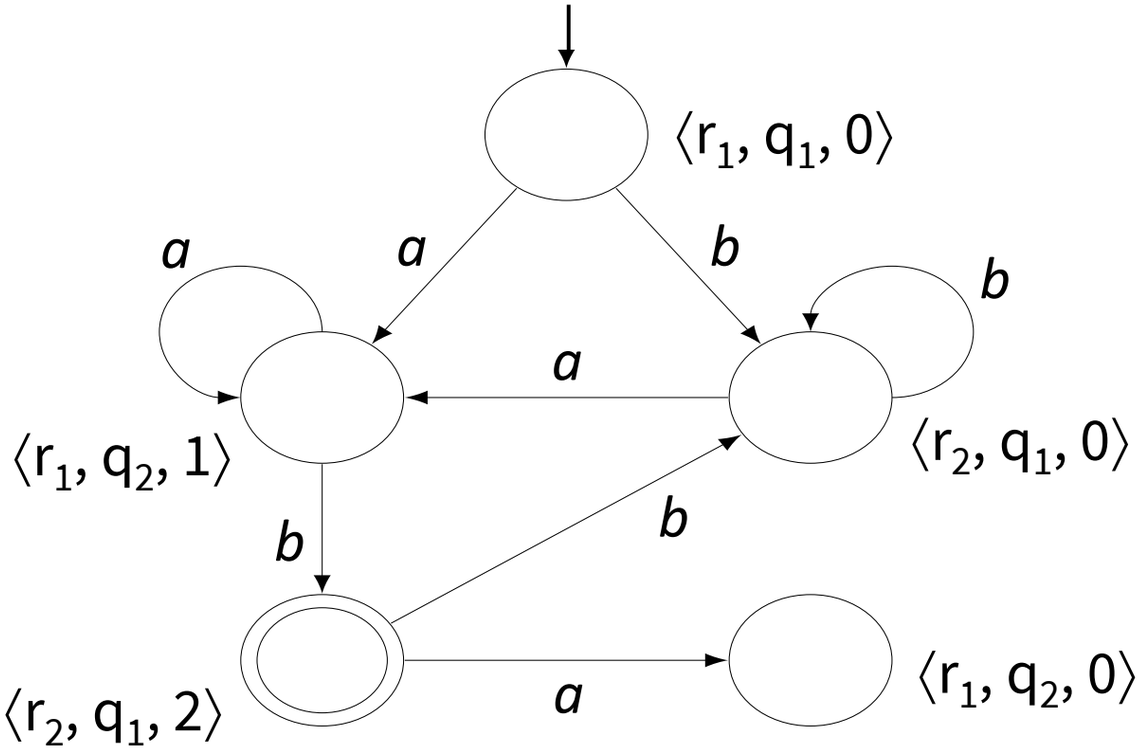


Example: Intersection of Büchi Automata

- Compute $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$ s.t. $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$
  - First copy: waits for $s \in F_1$
  - Second copy: waits for $s \in F_2$
  - Third copy: all states are accepting

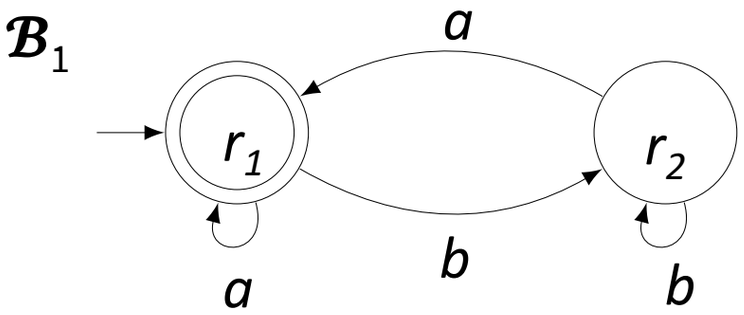


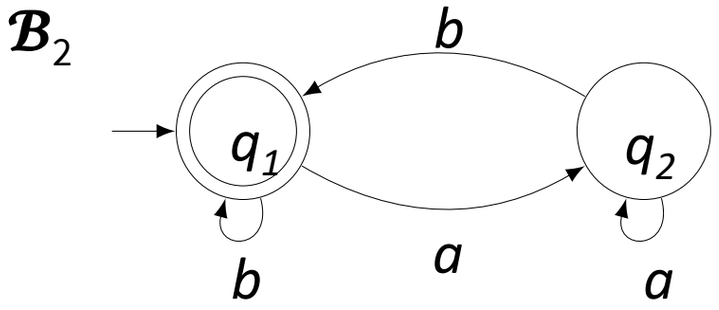


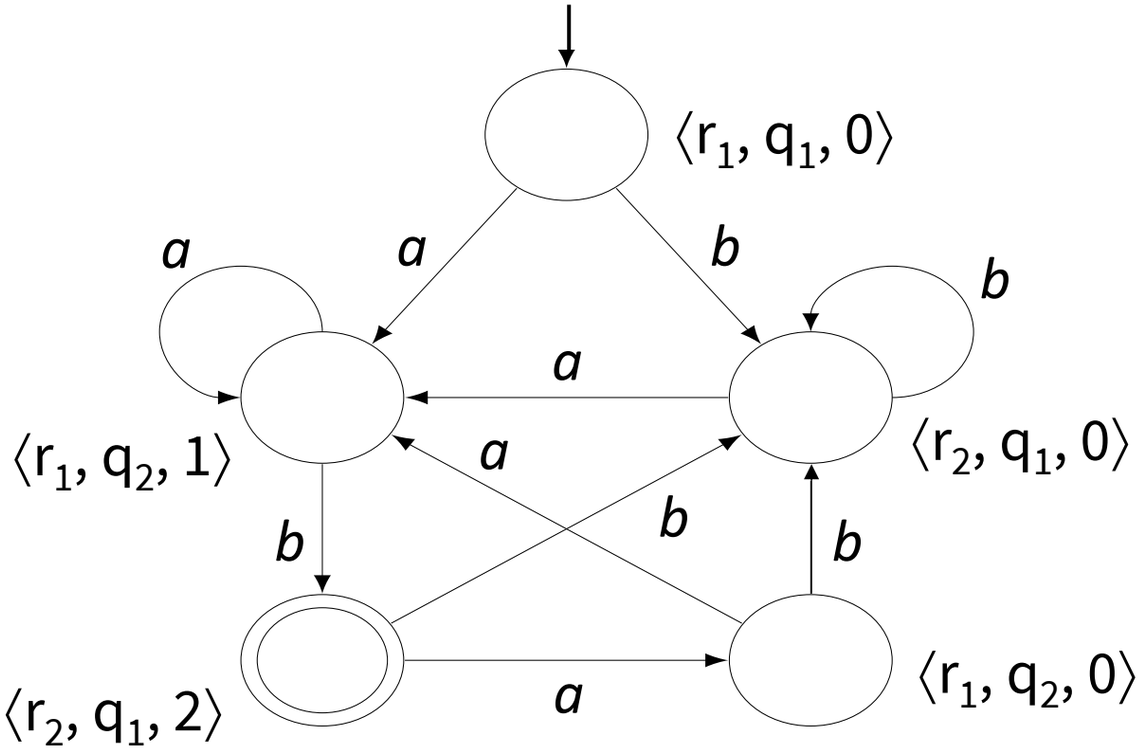


Example: Intersection of Büchi Automata

- Compute $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$ s.t. $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$
  - First copy: waits for $s \in F_1$
  - Second copy: waits for $s \in F_2$
  - Third copy: all states are accepting

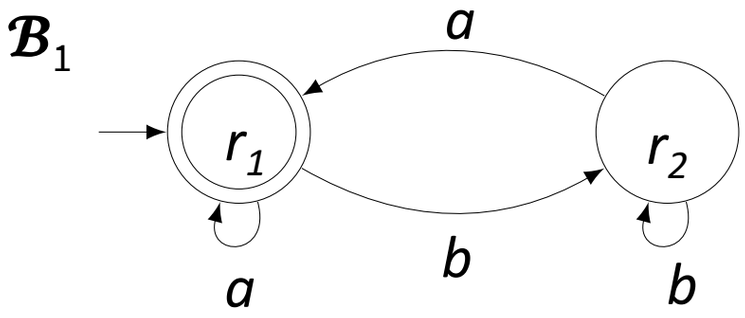


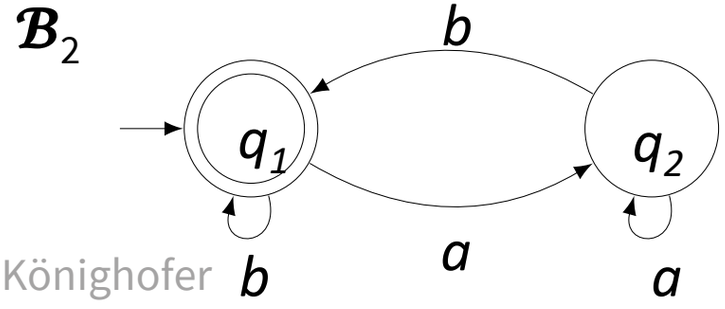


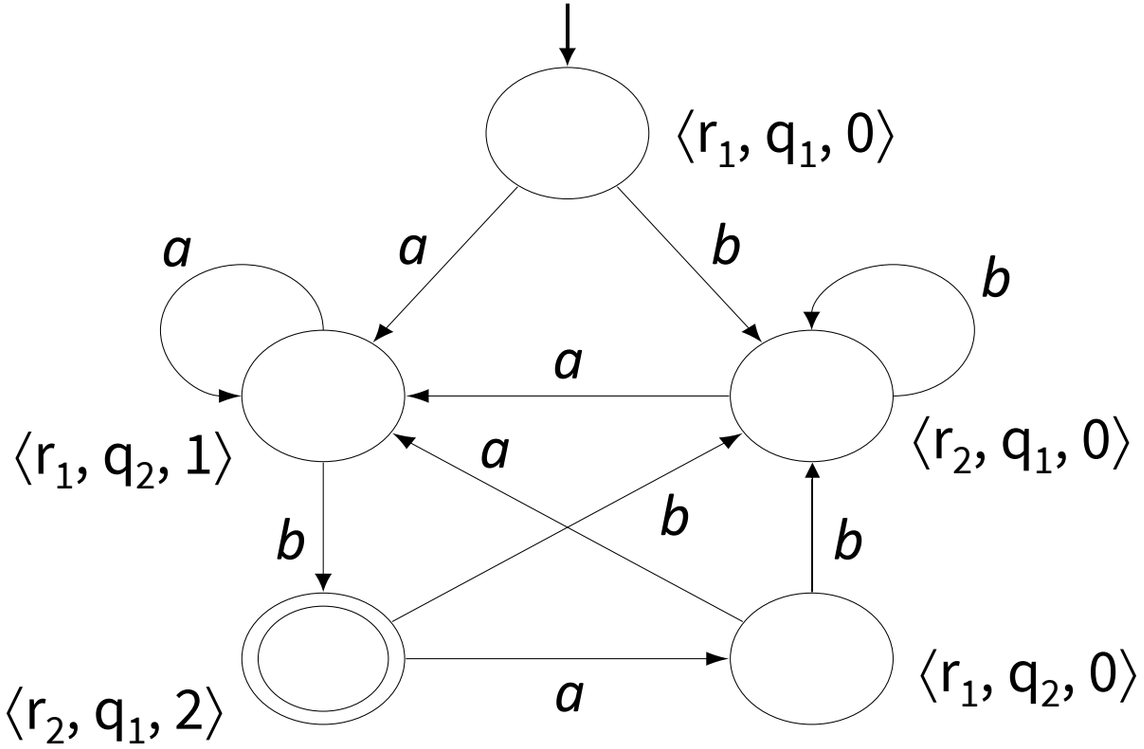


Example: Intersection of Büchi Automata

- Question
  - In every interval we first wait for an s in $\mathbf{F}_1$ and then wait for an s in $\mathbf{F}_2$.
  - We **ignore accepting states** that don't appear in this order.
  - Can we **miss accepting paths in $\mathcal{B}$** ?

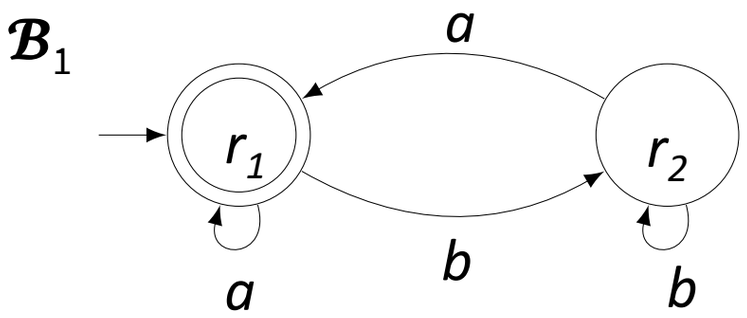


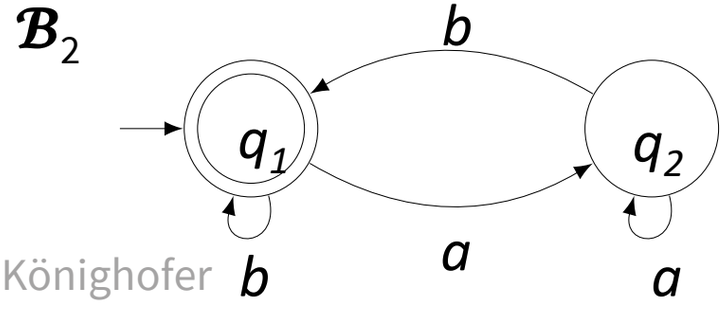


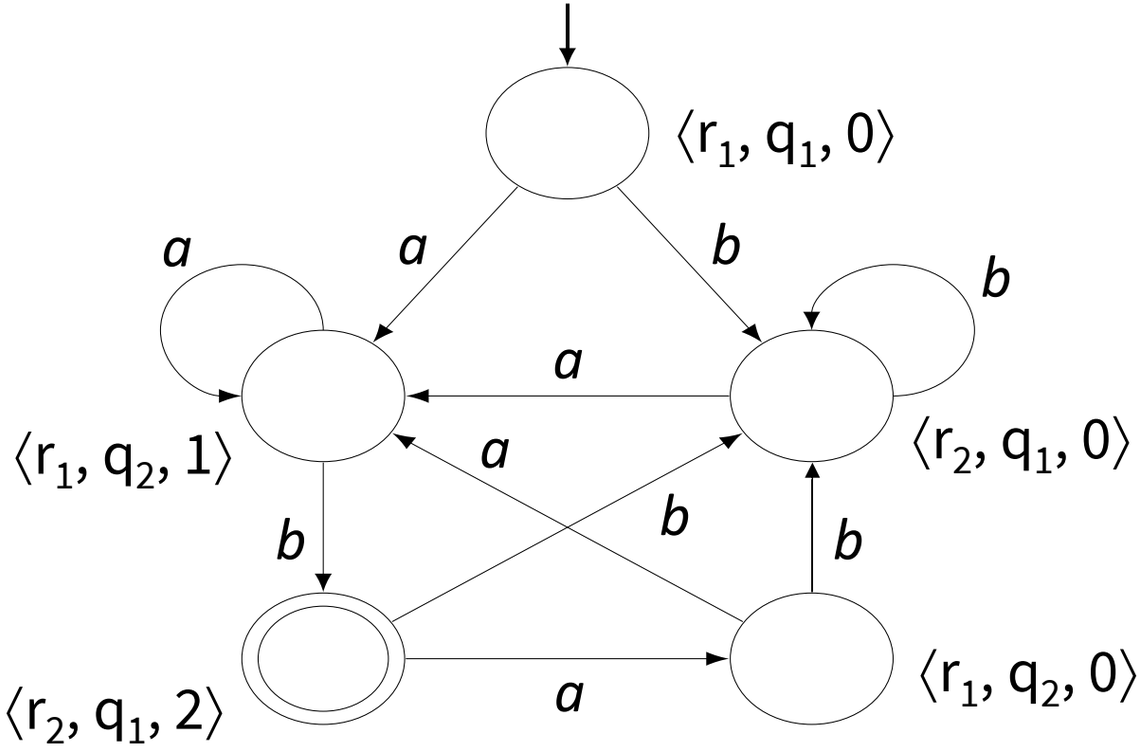


Example: Intersection of Büchi Automata

- Question
  - In every interval we first wait for an s in $\mathbf{F}_1$ and then wait for an s in $\mathbf{F}_2$.
  - We **ignore accepting states** that don't appear in this order.
  - Can we **miss accepting paths in $\mathcal{B}$** ?
    - **NO!** An accepting path visits infinitely many s from $\mathbf{F}_1$ and $\mathbf{F}_2$.
      → We can ignore finitely many accepting states in each interval







- Definitions
- Det/Nondet. Büchi
- Complementation
- **Intersection Büchi**
- Emptiness Büchi

Model Checking of LTL

- Given a Kripke structure M and a LTL formula φ:
  Does $M \models \varphi$?

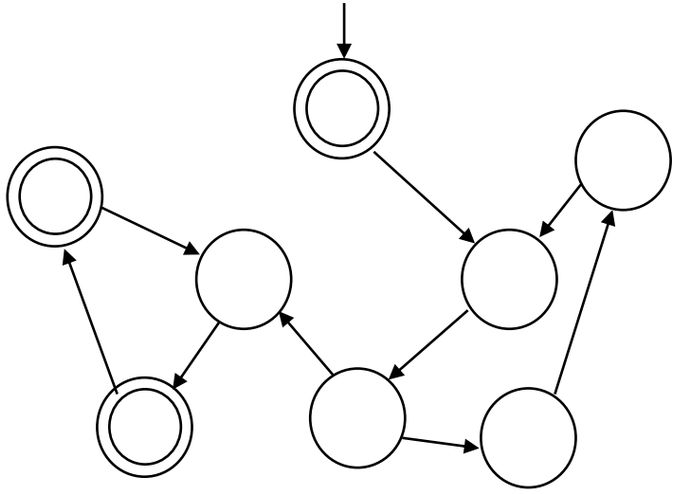
- **Automata-based Algorithm**

1. Construct $\neg\varphi$ ✓
2. Construct a Büchi automaton $\mathcal{S}_{\neg\varphi}$
3. Translate M to an automaton $\mathcal{A}$.
4. Construct the automaton $\mathcal{B}$ with $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{S}_{\neg\varphi})$ ✓
5. If $\mathcal{L}(\mathcal{B}) = \emptyset \Rightarrow M \models \varphi$ ⬅
6. If $\mathcal{L}(\mathcal{B}) \neq \emptyset \Rightarrow M \not\models \varphi$. A word $v \cdot w^{\omega} \in \mathcal{L}(\mathcal{B})$ is a **counterexample**
   → a trace in M that does not satisfy φ

Checking for Emptiness of $\mathcal{L}(\mathcal{B})$

- An infinite run ρ is **accepting** ⇔ it visits an **accepting state** an **infinitely often**.
  - $\inf(\rho) \cap F \neq \emptyset$

- $\mathcal{L}(\mathcal{B}) = \emptyset$ if there is **no reachable accepting state** on a **cycle**



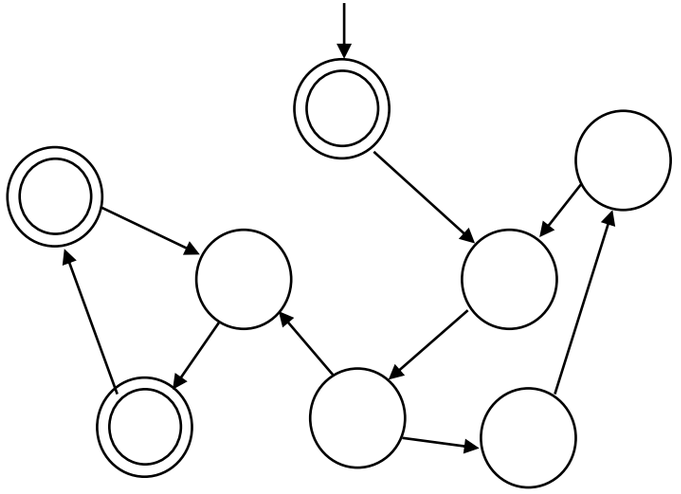
Checking for Emptiness of $\mathcal{L}(\mathcal{B})$

- Definitions
- Det/Nondet. Büchi
- Complementation
- Intersection Büchi
- **Emptiness Büchi**

- An infinite run ρ is **accepting** ⇔ it visits an **accepting state** an **infinitely often**.
  - $\inf(\rho) \cap F \neq \emptyset$

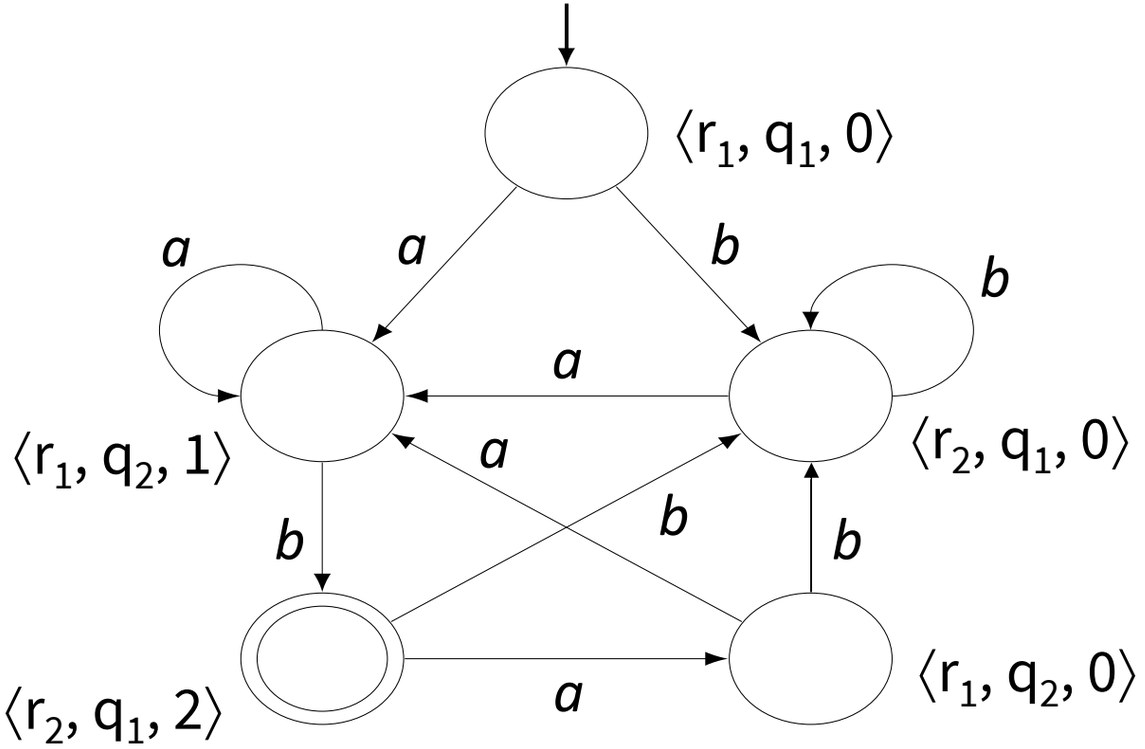
- $\mathcal{L}(\mathcal{B}) = \emptyset$ if there is **no reachable accepting state** on a **cycle**

- **Algorithm to check Emptiness for Büchi Automata:**
  - *$\mathcal{L}(\mathcal{B})$ is nonempty ⇔ The graph induced by $\mathcal{B}$ contains a path from an initial state to a state $t \in F$ and a path from t back to itself.*



Example: Checking for Emptiness of $\mathcal{L}(\mathcal{B})$

- Is the language $\mathcal{L}(\mathcal{B})$ empty?

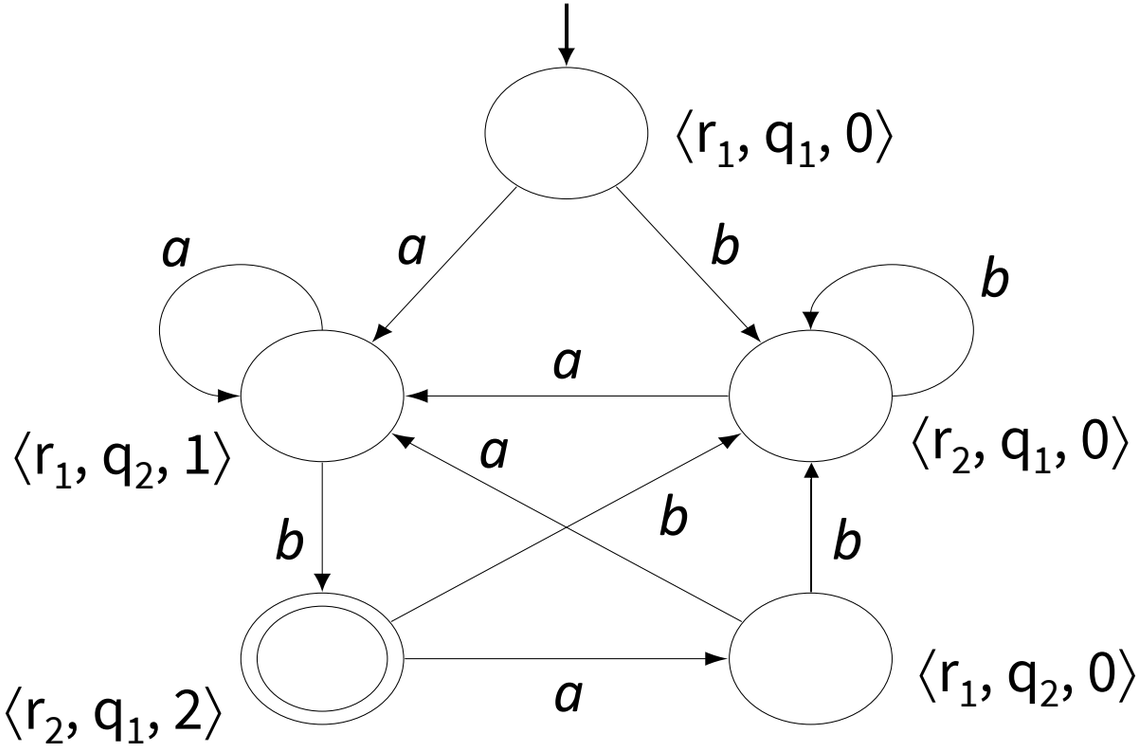


- Definitions
- Det/Nondet. Büchi
- Complementation
- Intersection Büchi
- **Emptiness Büchi**

Example: Checking for Emptiness of $\mathcal{L}(\mathcal{B})$

- Is the language $\mathcal{L}(\mathcal{B})$ empty?

- No: <r_2,q_1,2> is accepting and reachable from <r_1,q_1,0> and reachable from itself.



- Definitions
- Det/Nondet. Büchi
- Complementation
- **Intersection Büchi**
- Emptiness Büchi

Model Checking of LTL

- Given a Kripke structure M and a LTL formula φ:
  Does $M \models \varphi$?

- **Automata-based Algorithm**

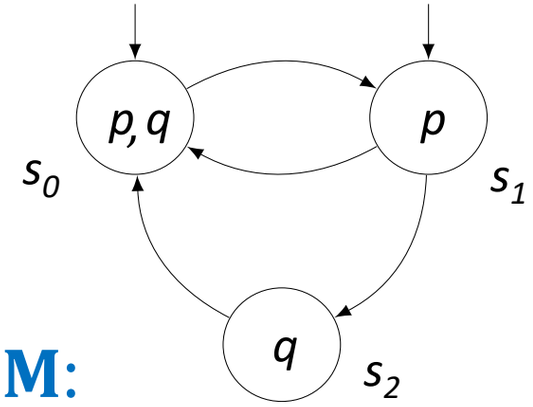
1. Construct $\neg\varphi$ ✓
2. Construct a Büchi automaton $\mathcal{S}_{\neg\varphi}$
3. Translate M to an automaton $\mathcal{A}$. ⬅
4. Construct the automaton $\mathcal{B}$ with $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{S}_{\neg\varphi})$ ✓
5. If $\mathcal{L}(\mathcal{B}) = \emptyset \Rightarrow M \models \varphi$ ✓
6. If $\mathcal{L}(\mathcal{B}) \neq \emptyset \Rightarrow M \not\models \varphi$. A word $v \cdot w^{\omega} \in \mathcal{L}(\mathcal{B})$ is a **counterexample** ✓
   → a trace in M that does not satisfy φ

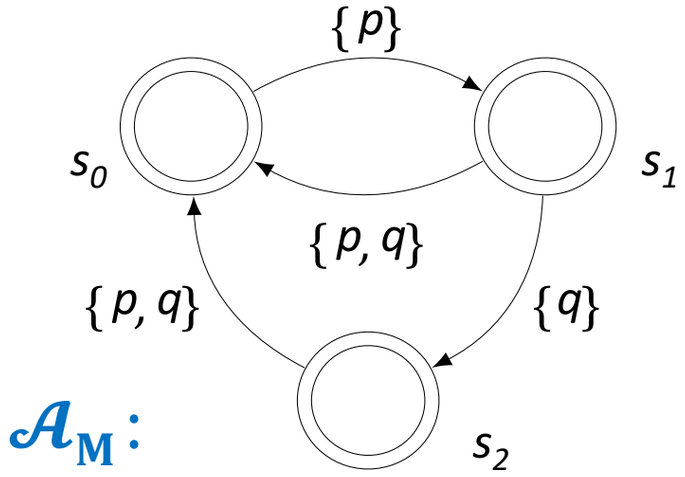
Plan for Today

- LTL Model Checking
  - Recap / Overview
  - Basics of Automata Theory
    - Finite automata on finite words (Regular automata)
    - Finite automata on infinite words (Büchi automata)
      - Definitions: Automata, word, run, language
      - Deterministic vs non-deterministic automata
      - Intersection of automata
      - Checking emptiness of automata
    - Kripke structures to automata
  - LTL Model-Checking Algorithm

Kripke Structure to Büchi Automaton

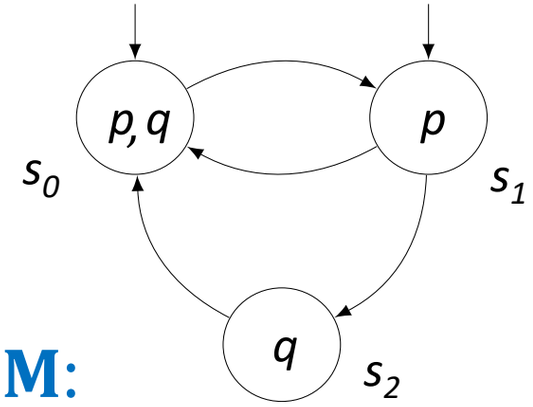
- Move labels to incoming transitions
  - Push labels backwards
- All states are accepting

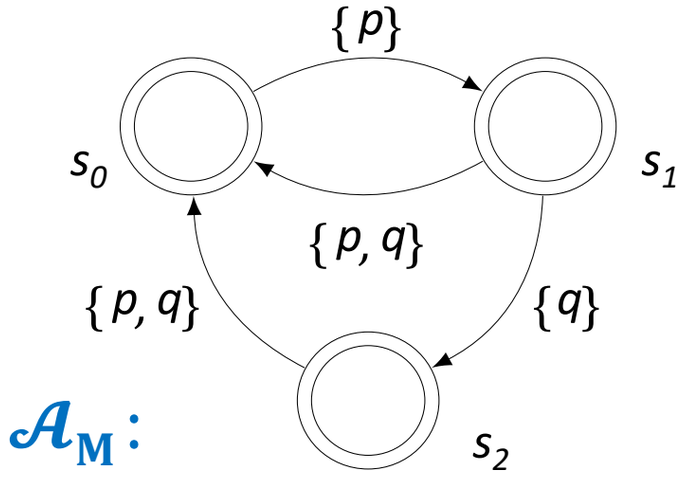




Kripke Structure to Büchi Automaton

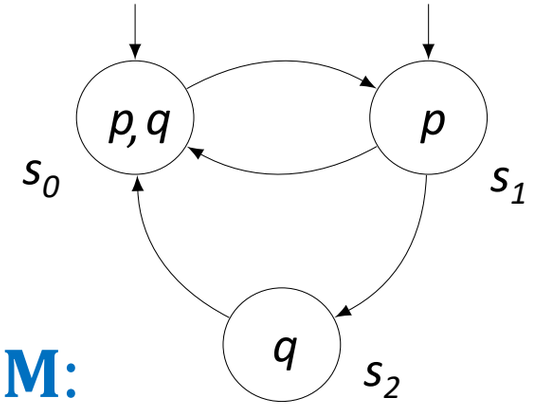
- Move labels to incoming transitions
  - Push labels backwards
- All states are accepting
- What about initial states?

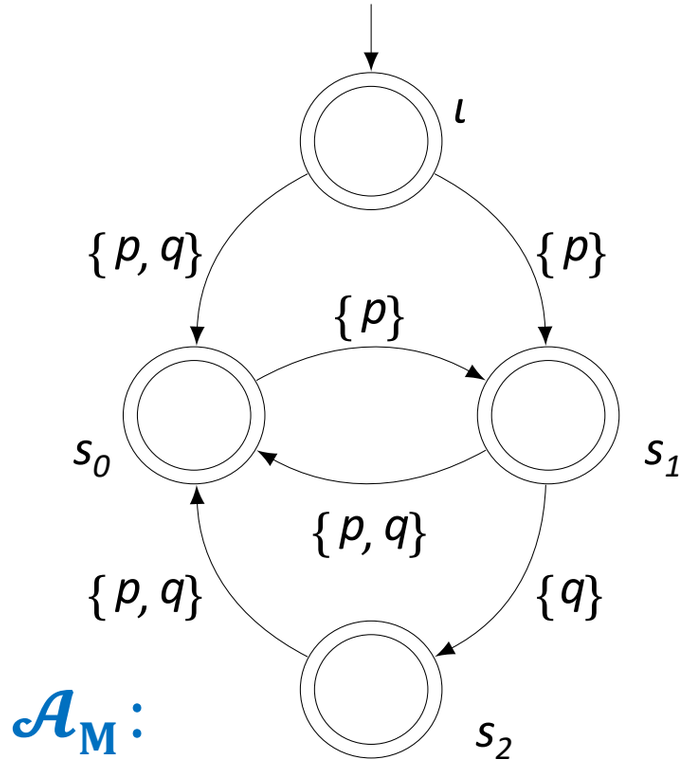




Kripke Structure to Büchi Automaton

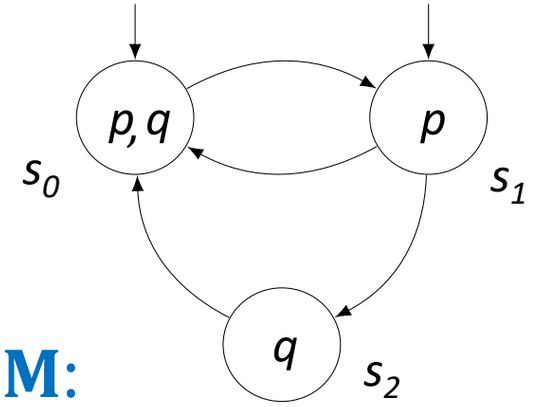
- Move labels to incoming transitions
  - Push labels backwards
- All states are accepting

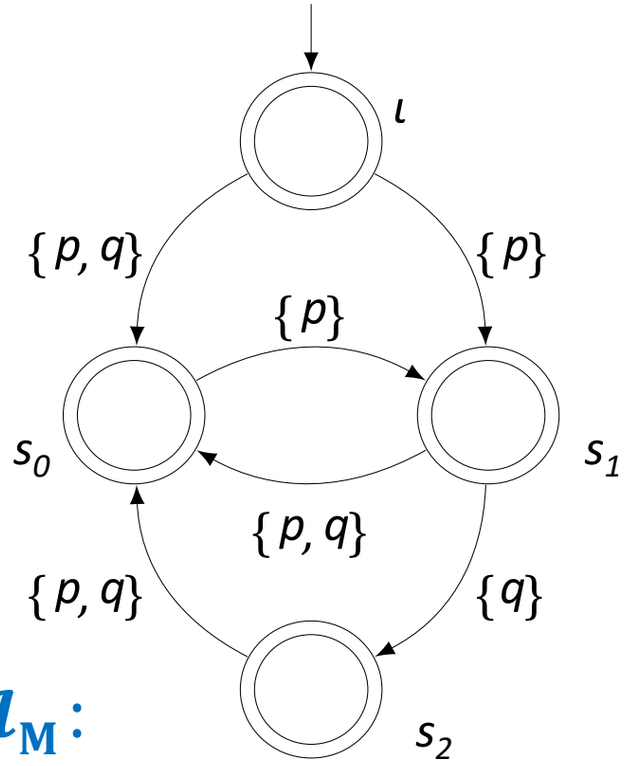




Kripke Structure to Büchi Automaton

- **Algorithm**:
- $\mathbf{M} = (\mathbf{S}, \mathbf{S}_0, \mathbf{R}, \mathbf{AP}, \mathbf{L}) \Rightarrow \mathcal{A}_{\mathbf{M}} = (\mathbf{\Sigma}, \mathbf{S} \cup \{\iota\}, \mathbf{\Delta}, \{\iota\}, \mathbf{S} \cup \{\iota\})$, where $\mathbf{\Sigma} = \mathbf{P(AP)}$.
  - $(s, a, s') \in \mathbf{\Delta} \Leftrightarrow (s, s') \in R$ and $a = L(s')$
  - $(\iota, a, s) \in \mathbf{\Delta} \Leftrightarrow s \in \mathbf{S}_0$ and $a = L(s)$





Model Checking of LTL

- Given a Kripke structure M and a LTL formula φ:
  Does $M \models \varphi$?

- **Automata-based Algorithm**

1. Construct $\neg\varphi$ ✓
2. Construct a Büchi automaton $\mathcal{S}_{\neg\varphi}$ ⬅ Next Week
3. Translate M to an automaton $\mathcal{A}$. ✓
4. Construct the automaton $\mathcal{B}$ with $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{S}_{\neg\varphi})$ ✓
5. If $\mathcal{L}(\mathcal{B}) = \emptyset \Rightarrow M \models \varphi$ ✓
6. If $\mathcal{L}(\mathcal{B}) \neq \emptyset \Rightarrow M \not\models \varphi$. A word $v \cdot w^{\omega} \in \mathcal{L}(\mathcal{B})$ is a **counterexample** ✓
   → a trace in M that does not satisfy φ

Plan for Today

- LTL Model Checking
  - Recap / Overview
  - Basics of Automata Theory
    - Finite automata on finite words (Regular automata)
    - Finite automata on infinite words (Büchi automata)
      - Definitions: Automata, word, run, language
      - Deterministic vs non-deterministic automata
      - Intersection of automata
      - Checking emptiness of automata
    - Kripke structures to automata
  - LTL Model-Checking Algorithm

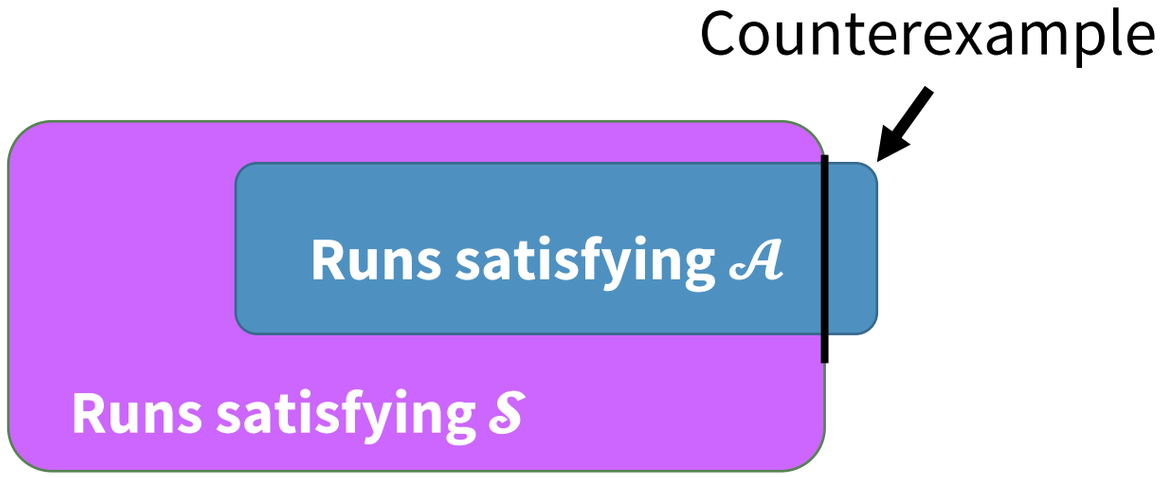
LTL MC when system $\mathcal{A}$ and spec $\mathcal{S}$ are given as Büchi Automata

- **$\mathcal{A}$ satisfies $\mathcal{S}$** if $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{S})$
  - Is any behaviour of $\mathcal{A}$ allowed by $\mathcal{S}$

LTL MC when system $\mathcal{A}$ and spec $\mathcal{S}$ are given as Büchi Automata

- **$\mathcal{A}$ satisfies $\mathcal{S}$** if $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{S})$
  - Is any behaviour of $\mathcal{A}$ allowed by $\mathcal{S}$

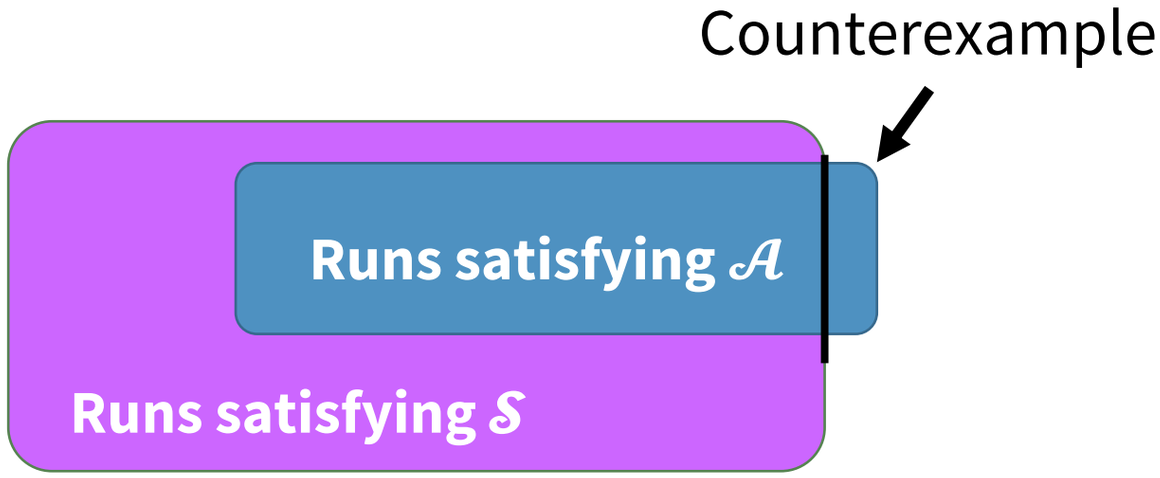
- $\mathcal{A}$ does **not** satisfy $\mathcal{S}$ if $\mathcal{L}(\mathcal{A}) \not\subseteq \mathcal{L}(\mathcal{S})$



LTL MC when system $\mathcal{A}$ and spec $\mathcal{S}$ are given as Büchi Automata

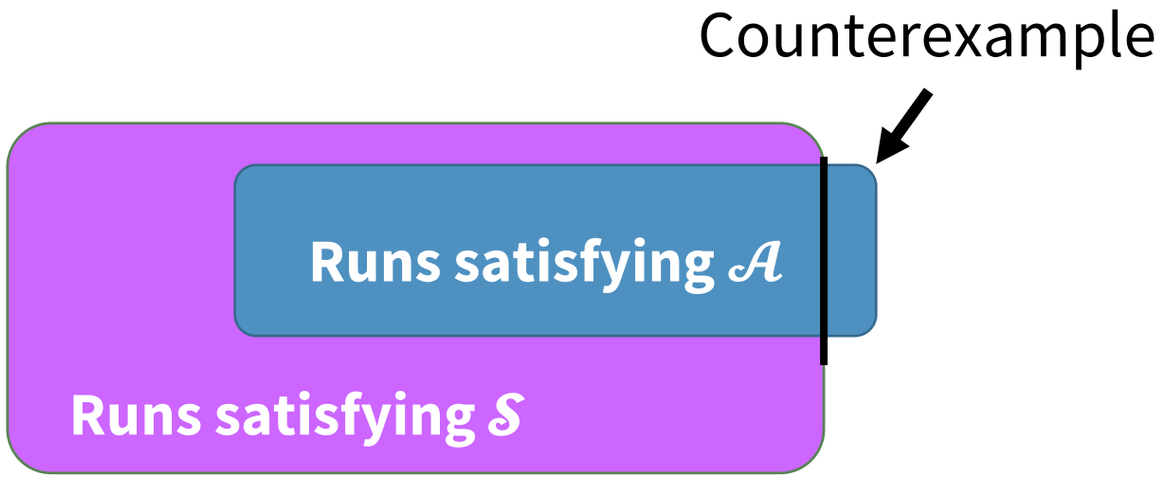
- $\mathcal{A}$ **satisfies** $\mathcal{S}$ if $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{S})$
  - Is any behaviour of $\mathcal{A}$ allowed by $\mathcal{S}$

- $\mathcal{A}$ does **not** satisfy $\mathcal{S}$ if $\mathcal{L}(\mathcal{A}) \not\subseteq \mathcal{L}(\mathcal{S})$
- LTL MC:
  - $\mathcal{L}(\mathcal{A}) \not\subseteq \mathcal{L}(\mathcal{S}) \equiv \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\overline{\mathcal{S}}) \neq \emptyset$



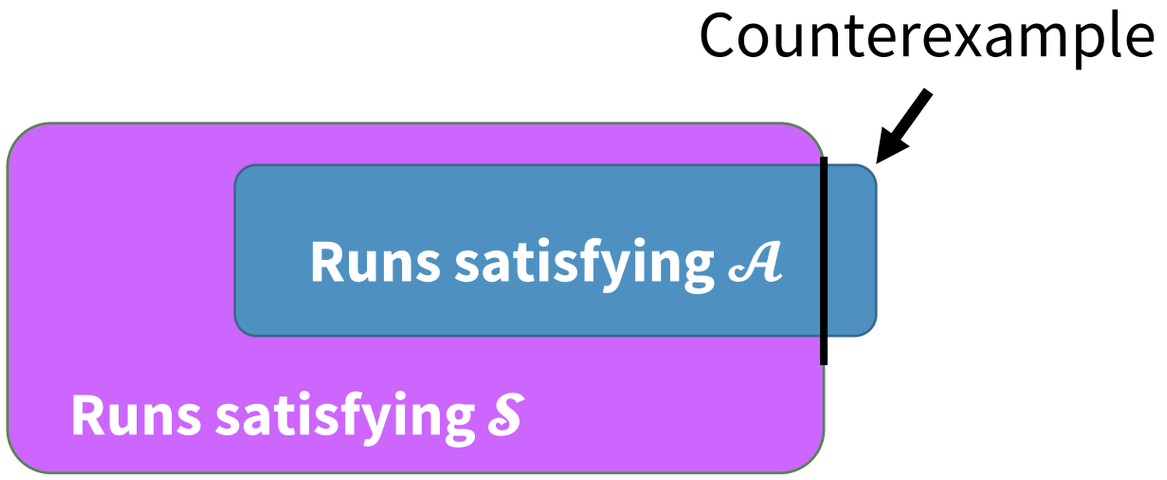
LTL MC when system $\mathcal{A}$ and spec $\mathcal{S}$ are given as Büchi Automata

- **Algorithm – Version 1**
  1. Compute complement of $\mathcal{S} \rightarrow \overline{\mathcal{S}}$
  2. Construct the automaton $\mathcal{B}$ with $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\overline{\mathcal{S}})$
  3. If $\mathcal{L}(\mathcal{B}) = \emptyset \Rightarrow \mathcal{A}$ satisfies $\mathcal{S}$
  4. Otherwise, a word $v \cdot w^{\omega} \in \mathcal{L}(\mathcal{B})$ is a counterexample



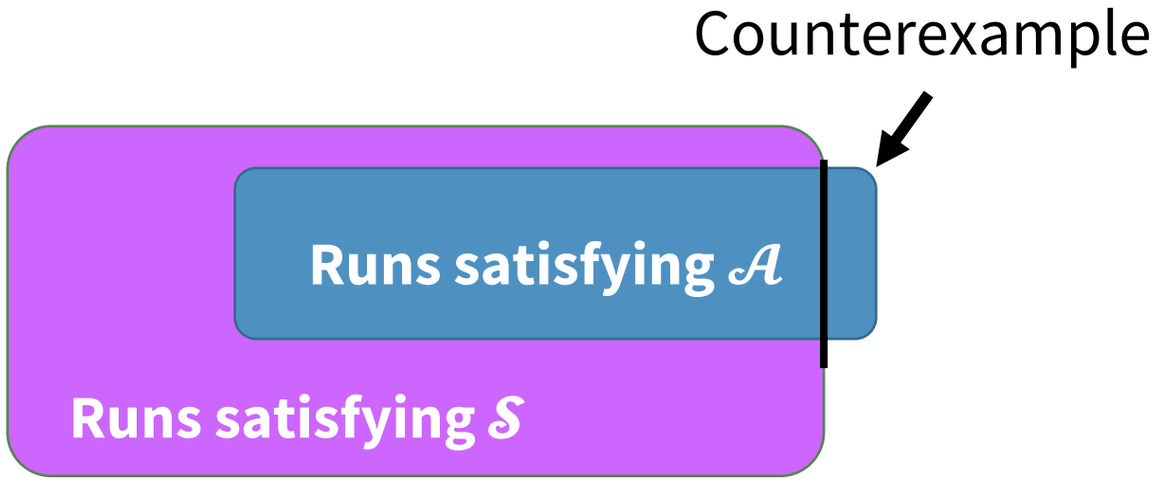
LTL MC when system $\mathcal{A}$ and spec $\mathcal{S}$ are given as Büchi Automata

- **Algorithm – Version 1**
  1. Compute complement of $\mathcal{S} \rightarrow \overline{\mathcal{S}}$ **very hard**
  2. Construct the automaton $\mathcal{B}$ with $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\overline{\mathcal{S}})$
  3. If $\mathcal{L}(\mathcal{B}) = \emptyset \Rightarrow \mathcal{A}$ satisfies $\mathcal{S}$
  4. Otherwise, a word $v \cdot w^{\omega} \in \mathcal{L}(\mathcal{B})$ is a counterexample



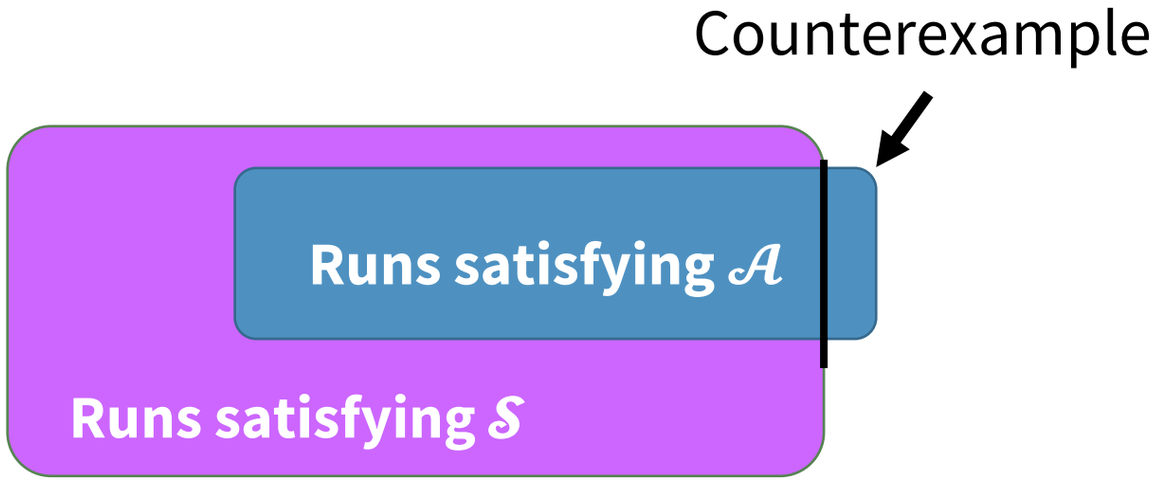
LTL MC when system $\mathcal{A}$ and spec $\mathcal{S}$ are given as Büchi Automata

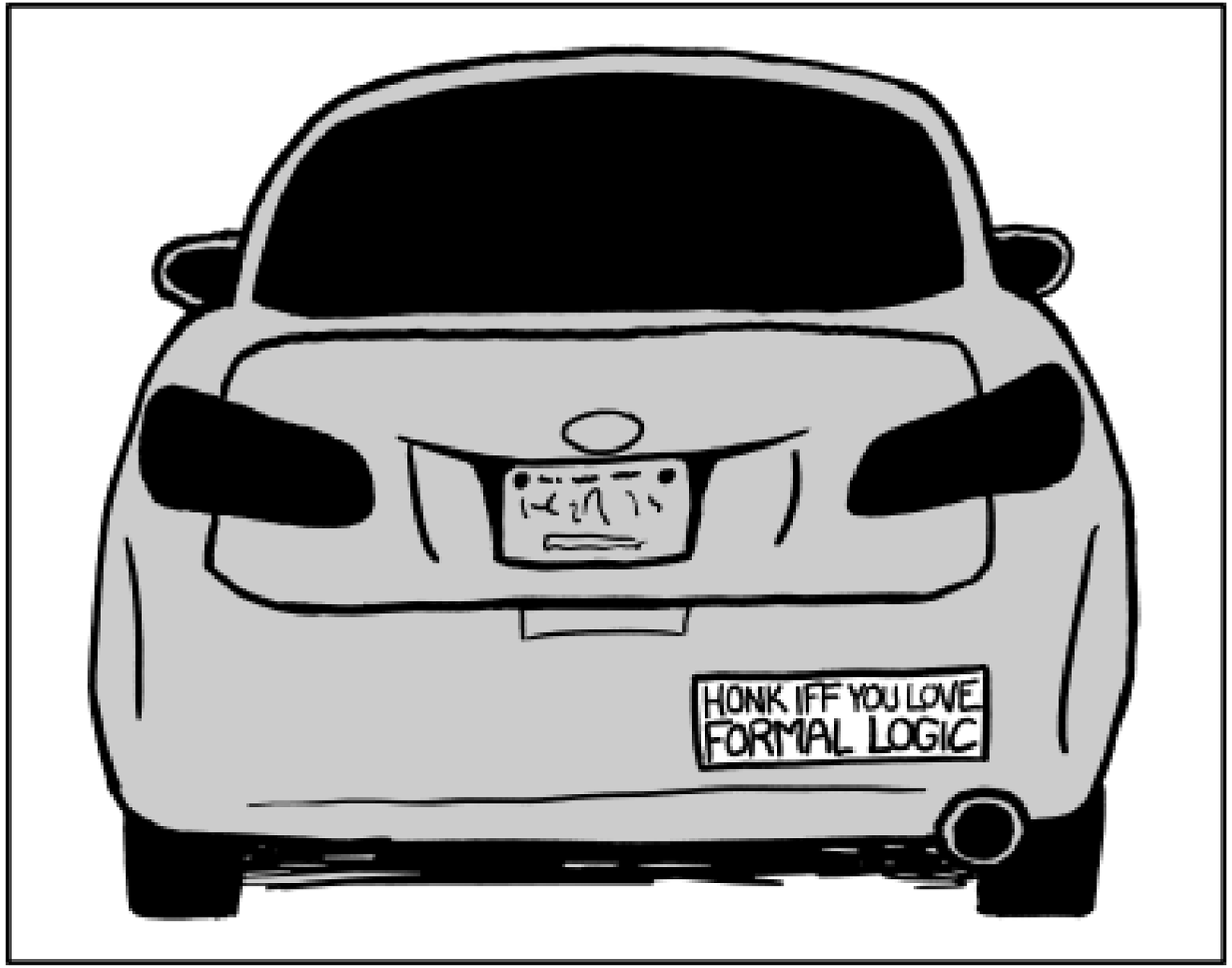
- **Algorithm – Version 2**
  1. Construct $\neg\varphi$
  2. Construct Büchi Automaton $\mathcal{S}_{\neg\varphi}$
  3. Construct the automaton $\mathcal{B}$ with $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\overline{\mathcal{S}})$
  4. If $\mathcal{L}(\mathcal{B}) = \emptyset \Rightarrow \mathcal{A}$ satisfies $\mathcal{S}$
  5. Otherwise, a word $v \cdot w^{\omega} \in \mathcal{L}(\mathcal{B})$ is a counterexample



LTL MC when system $\mathcal{A}$ and spec $\mathcal{S}$ are given as Büchi Automata

- **Algorithm – Version 2**
  1. Construct $\neg\varphi$
  2. Construct Büchi Automaton $\mathcal{S}_{\neg\varphi}$ ← Next Week
  3. Construct the automaton $\mathcal{B}$ with $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\overline{\mathcal{S}})$
  4. If $\mathcal{L}(\mathcal{B}) = \emptyset \Rightarrow \mathcal{A}$ satisfies $\mathcal{S}$
  5. Otherwise, a word $v \cdot w^{\omega} \in \mathcal{L}(\mathcal{B})$ is a counterexample




HONK IFF YOU LOVE
FORMAL LOGIC


ANY
QUESTIONS
?