Game-Theoretic Approach to Planning and Synthesis

Giuseppe De Giacomo Antonio Di Stasio Giuseppe Perelli Shufang Zhu

ERC Advanced Grant
WhiteMech:
White-box Self Programming Mechanisms
SAPIENZA
UNIVERSITÀ DI ROMA

PhD-AI Course
4-8 July, 2022

Finite and infinite trace languages

- An alphabet is a (finite) set of symbols (letters). E.g. $\Sigma = \{a, b\}$
- A finite trace over Σ is a finite sequence of letters. E.g. $w = ababbab$
- An infinite trace over Σ is an infinite sequence of letters. E.g. $w = ababbab\ldots$
- The sets of all finite and infinite traces are denoted Σ^* and Σ^ω, respectively.
- A finite language is a subset $L \subseteq \Sigma^*$. E.g. $L =$ "traces ending with an a"
- An infinite language is a subset $L \subseteq \Sigma^\omega$. E.g. $L =$ "traces containing a finite number of a"

- An alphabet is a (finite) set of symbols (letters). E.g. $\Sigma = \{a, b\}$
- A finite trace over Σ is a finite sequence of letters. E.g. $w = ababbab$
- An infinite trace over Σ is an infinite sequence of letters. E.g. $w = ababbab\ldots$
- The sets of all finite and infinite traces are denoted Σ^* and Σ^ω, respectively.
- A finite language is a subset $L \subseteq \Sigma^*$. E.g. $L =$ "traces ending with an a"
- An infinite language is a subset $L \subseteq \Sigma^\omega$. E.g. $L =$ "traces containing a finite number of a"

Language recognition problem

Determine whether a trace w belongs to a language L.

- An alphabet is a (finite) set of symbols (letters). E.g. $\Sigma = \{a, b\}$
- A finite trace over Σ is a finite sequence of letters. E.g. $w = ababbab$
- An infinite trace over Σ is an infinite sequence of letters. E.g. $w = ababbab\ldots$
- The sets of all finite and infinite traces are denoted Σ^* and Σ^ω, respectively.
- A finite language is a subset $L \subseteq \Sigma^*$. E.g. $L =$ "traces ending with an a"
- An infinite language is a subset $L \subseteq \Sigma^\omega$. E.g. $L =$ "traces containing a finite number of a"

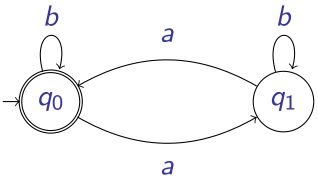
Language recognition problem

Determine whether a trace w belongs to a language L.

Automata are computational devices used to solve language recognition and related problems.

A Deterministic Finite-State Automaton (DFA) is a tuple $\mathcal{D} = \langle Q, \Sigma, s, \delta, F \rangle$ with:

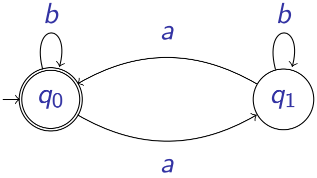
- Q finite set of states
- Σ finite alphabet
- $q_0 \in Q$ initial state
- $F \subseteq Q$ set of final states
- $\delta : Q \times \Sigma \rightarrow Q$ transition function



Recognizes the traces with an even number of a.

A Deterministic Finite-State Automaton (DFA) is a tuple $\mathcal{D} = \langle Q, \Sigma, s, \delta, F \rangle$ with:

- Q finite set of states
- Σ finite alphabet
- $q_0 \in Q$ initial state
- $F \subseteq Q$ set of final states
- $\delta : Q \times \Sigma \to Q$ transition function

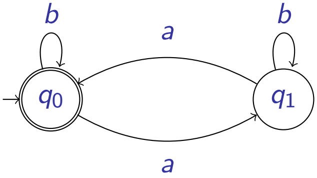


Recognizes the traces with an even number of a.

A trace $w \in \Sigma^*$ is read on $\mathcal{D}$ by starting from q_0 and following the transition function, generating a run $\rho \in Q^*$.
We say $w \in \mathcal{L}(\mathcal{D}) \subseteq \Sigma^*$ if the corresponding run ρ ends in a final state.

A Deterministic Finite-State Automaton (DFA) is a tuple $\mathcal{D} = \langle Q, \Sigma, s, \delta, F \rangle$ with:

- Q finite set of states
- Σ finite alphabet
- $q_0 \in Q$ initial state
- $F \subseteq Q$ set of final states
- $\delta : Q \times \Sigma \to Q$ transition function



Recognizes the traces with an even number of a.

A trace $w \in \Sigma^*$ is read on $\mathcal{D}$ by starting from q_0 and following the transition function, generating a run $\rho \in Q^*$.
We say $w \in \mathcal{L}(\mathcal{D}) \subseteq \Sigma^*$ if the corresponding run ρ ends in a final state.

Sample execution: $q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_1 \xrightarrow{a} q_0 \xrightarrow{b} q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_1$

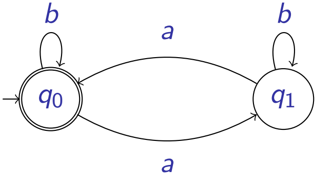
SAPIENZA UNIVERSITÀ DI ROMA

For a given $\mathcal{D} = \langle Q, \Sigma, q_0, \delta, F\rangle$, the dual automaton $\overline{\mathcal{D}} = \langle Q, \Sigma, q_0, \delta, Q \setminus F\rangle$ is such that

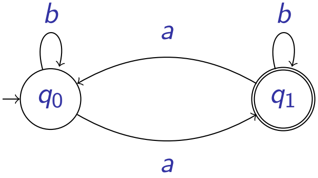
$$\mathcal{L}(\overline{\mathcal{D}}) = \Sigma^* \setminus \mathcal{L}(\mathcal{D})$$

For a given $\mathcal{D} = \langle Q, \Sigma, q_0, \delta, F \rangle$, the dual automaton $\overline{\mathcal{D}} = \langle Q, \Sigma, q_0, \delta, Q \setminus F \rangle$ is such that

$$\mathcal{L}(\overline{\mathcal{D}}) = \Sigma^* \setminus \mathcal{L}(\mathcal{D})$$



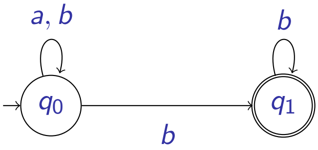
Recognizes traces with an even number of a.



Recognizes traces with an odd number of a.

A Nondeterministic Finite-State Automaton (NFA) is a tuple $\mathcal{N} = \langle Q, \Sigma, I, \delta, F \rangle$ with:

- Q finite set of states
- Σ finite alphabet
- $I \subseteq Q$ set of initial states
- $F \subseteq Q$ set of final states
- $\delta : Q \times \Sigma \to 2^Q$ nondeterministic transition function



Recognizes the traces that end with b.

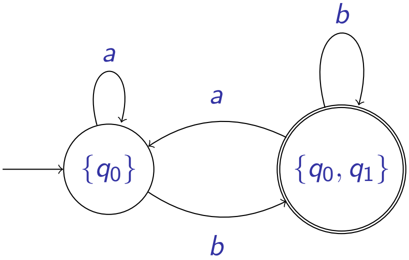
More than one run is possible on the same trace $w \in \Sigma^*$.
The automaton $\mathcal{N}$ accepts $w \in \mathcal{L}(\mathcal{N})$ if at least one run is accepting.

SAPIENZA UNIVERSITÀ DI ROMA

The subset construction

Let $\mathcal{N} = \langle Q, \Sigma, I, \delta, F\rangle$ be a nondeterministic automaton. Consider the deterministic automaton $\mathcal{D}_{\mathcal{N}} = \langle 2^Q, \Sigma, Q_0, \delta', \mathcal{F}\rangle$ with:

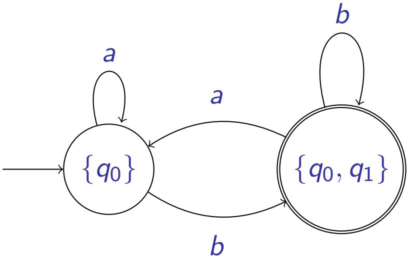
- $Q_0 = I$
- $\mathcal{F} = \{Q' \subseteq Q : Q' \cap F \neq \emptyset\}$
- $\delta'(Q', \sigma) = \bigcup_{q \in Q'} \delta(q, \sigma)$



The subset construction

Let $\mathcal{N} = \langle Q, \Sigma, I, \delta, F \rangle$ be a nondeterministic automaton. Consider the deterministic automaton $\mathcal{D}_{\mathcal{N}} = \langle 2^Q, \Sigma, Q_0, \delta', \mathcal{F} \rangle$ with:

- $Q_0 = I$
- $\mathcal{F} = \{Q' \subseteq Q : Q' \cap F \neq \emptyset\}$
- $\delta'(Q', \sigma) = \bigcup_{q \in Q'} \delta(q, \sigma)$



Intuition: $\mathcal{D}_{\mathcal{N}}$ runs all the possible executions of $\mathcal{N}$ in parallel.
If one of them accepts the trace in $\mathcal{N}$, then it does so in $\mathcal{D}_{\mathcal{N}}$.

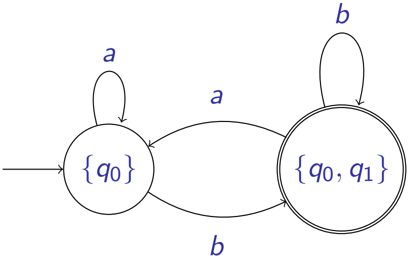
$\mathcal{L}(\mathcal{D}_{\mathcal{N}}) = \mathcal{L}(\mathcal{N})$

SAPIENZA UNIVERSITÀ DI ROMA

The subset construction

Let $\mathcal{N} = \langle Q, \Sigma, I, \delta, F\rangle$ be a nondeterministic automaton. Consider the deterministic automaton $\mathcal{D}_\mathcal{N} = \langle 2^Q, \Sigma, Q_0, \delta', \mathcal{F}\rangle$ with:

- $Q_0 = I$
- $\mathcal{F} = \{Q' \subseteq Q : Q' \cap F \neq \emptyset\}$
- $\delta'(Q', \sigma) = \bigcup_{q \in Q'} \delta(q, \sigma)$



Intuition: $\mathcal{D}_\mathcal{N}$ runs all the possible executions of $\mathcal{N}$ in parallel.
If one of them accepts the trace in $\mathcal{N}$, then it does so in $\mathcal{D}_\mathcal{N}$.

$$\mathcal{L}(\mathcal{D}_\mathcal{N}) = \mathcal{L}(\mathcal{N})$$

Observation: $|Q_{\mathcal{D}_\mathcal{N}}| = 2^{|Q_\mathcal{N}|}$.
Unfortunately, this exponential blow-up cannot be avoided.

A NFA $\mathcal{N}$ is complemented by:

1. NFA determinization $\mathcal{N} \Rightarrow \mathcal{D}_{\mathcal{N}}$
2. DFA complementation $\mathcal{D}_{\mathcal{N}} \Rightarrow \overline{\mathcal{D}_{\mathcal{N}}}$

A NFA $\mathcal{N}$ is complemented by:

1. NFA determinization $\mathcal{N} \Rightarrow \mathcal{D}_{\mathcal{N}}$
2. DFA complementation $\mathcal{D}_{\mathcal{N}} \Rightarrow \overline{\mathcal{D}_{\mathcal{N}}}$

Observation: the determinizing operation comes with an exponential blow-up in the size of the state-space of the automaton.

Union construction

Take two NFAs $\mathcal{N}_1 = \langle Q_1, \Sigma, I_1, \delta_1, F_1 \rangle$ and $\mathcal{N}_2 = \langle Q_2, \Sigma, I_2, \delta_2, F_2 \rangle$ defined over Σ.
The union automaton $\mathcal{N}_1 \cup \mathcal{N}_2 = \langle Q, \Sigma, I, \delta, F \rangle$ is defined as:

SAPIENZA
UNIVERSITÀ DI ROMA

Union construction

Take two NFAs $\mathcal{N}_1 = \langle Q_1, \Sigma, I_1, \delta_1, F_1 \rangle$ and $\mathcal{N}_2 = \langle Q_2, \Sigma, I_2, \delta_2, F_2 \rangle$ defined over Σ.
The union automaton $\mathcal{N}_1 \cup \mathcal{N}_2 = \langle Q, \Sigma, I, \delta, F \rangle$ is defined as:

$$Q = Q_1 \cup Q_2 \text{ [a]}$$

$$I = I_1 \cup I_2$$

$$F = F_1 \cup F_2$$

$$\delta(q, \sigma) = \begin{cases} \delta_1(q, \sigma), & q \in Q_1 \\ \delta_2(q, \sigma), & q \in Q_2 \end{cases}$$

[a] We assume that Q_1 and Q_2 are disjoint.

SAPIENZA UNIVERSITÀ DI ROMA

Union construction

Take two NFAs $\mathcal{N}_1 = \langle Q_1, \Sigma, I_1, \delta_1, F_1 \rangle$ and $\mathcal{N}_2 = \langle Q_2, \Sigma, I_2, \delta_2, F_2 \rangle$ defined over Σ.
The union automaton $\mathcal{N}_1 \cup \mathcal{N}_2 = \langle Q, \Sigma, I, \delta, F \rangle$ is defined as:

$$Q = Q_1 \cup Q_2 \text{ [a]}$$

$$I = I_1 \cup I_2$$

$$F = F_1 \cup F_2$$

$$\delta(q, \sigma) = \begin{cases} \delta_1(q, \sigma), & q \in Q_1 \\ \delta_2(q, \sigma), & q \in Q_2 \end{cases}$$

[a] We assume that Q_1 and Q_2 are disjoint.

Intuition: $\mathcal{N}_1 \cup \mathcal{N}_2$ chooses nondeterministically to execute either $\mathcal{N}_1$ or $\mathcal{N}_2$.

SAPIENZA UNIVERSITÀ DI ROMA

Union construction

Take two NFAs $\mathcal{N}_1 = \langle Q_1, \Sigma, I_1, \delta_1, F_1 \rangle$ and $\mathcal{N}_2 = \langle Q_2, \Sigma, I_2, \delta_2, F_2 \rangle$ defined over Σ.
The union automaton $\mathcal{N}_1 \cup \mathcal{N}_2 = \langle Q, \Sigma, I, \delta, F \rangle$ is defined as:

$$Q = Q_1 \cup Q_2 \text{ [a]}$$

$$I = I_1 \cup I_2$$

$$F = F_1 \cup F_2$$

$$\delta(q, \sigma) = \begin{cases} \delta_1(q, \sigma), & q \in Q_1 \\ \delta_2(q, \sigma), & q \in Q_2 \end{cases}$$

[a] We assume that Q_1 and Q_2 are disjoint.

Intuition: $\mathcal{N}_1 \cup \mathcal{N}_2$ chooses nondeterministically to execute either $\mathcal{N}_1$ or $\mathcal{N}_2$.
$\mathcal{L}(\mathcal{N}_1 \cup \mathcal{N}_2) = \mathcal{L}(\mathcal{N}_1) \cup \mathcal{L}(\mathcal{N}_2)$.

Synchronous product construction

Take two NFAs $\mathcal{N}_1 = \langle Q_1, \Sigma, I_1, \delta_1, F_1 \rangle$ and $\mathcal{N}_2 = \langle Q_2, \Sigma, I_2, \delta_2, F_2 \rangle$ defined over Σ.
The product automaton $\mathcal{N}_1 \times \mathcal{N}_2 = \langle Q, \Sigma, I, \delta, F \rangle$ is defined as:

SAPIENZA UNIVERSITÀ DI ROMA

Synchronous product construction

Take two NFAs $\mathcal{N}_1 = \langle Q_1, \Sigma, I_1, \delta_1, F_1 \rangle$ and $\mathcal{N}_2 = \langle Q_2, \Sigma, I_2, \delta_2, F_2 \rangle$ defined over Σ.
The product automaton $\mathcal{N}_1 \times \mathcal{N}_2 = \langle Q, \Sigma, I, \delta, F \rangle$ is defined as:

$$Q = Q_1 \times Q_2$$

$$I = I_1 \times I_2$$

$$F = F_1 \times F_2$$

$$\delta((q_1, q_2), \sigma) = (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma))$$

SAPIENZA
UNIVERSITÀ DI ROMA

Synchronous product construction

Take two NFAs $\mathcal{N}_1 = \langle Q_1, \Sigma, I_1, \delta_1, F_1 \rangle$ and $\mathcal{N}_2 = \langle Q_2, \Sigma, I_2, \delta_2, F_2 \rangle$ defined over Σ.
The product automaton $\mathcal{N}_1 \times \mathcal{N}_2 = \langle Q, \Sigma, I, \delta, F \rangle$ is defined as:

$$Q = Q_1 \times Q_2$$

$$I = I_1 \times I_2$$

$$F = F_1 \times F_2$$

$$\delta((q_1, q_2), \sigma) = (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma))$$

Intuition: $\mathcal{N}_1 \times \mathcal{N}_2$ executes $\mathcal{N}_1$ and $\mathcal{N}_2$ in parallel.

Sapienza Università di Roma

Synchronous product construction

Take two NFAs $\mathcal{N}_1 = \langle Q_1, \Sigma, I_1, \delta_1, F_1 \rangle$ and $\mathcal{N}_2 = \langle Q_2, \Sigma, I_2, \delta_2, F_2 \rangle$ defined over Σ.
The product automaton $\mathcal{N}_1 \times \mathcal{N}_2 = \langle Q, \Sigma, I, \delta, F \rangle$ is defined as:

$$Q = Q_1 \times Q_2$$

$$I = I_1 \times I_2$$

$$F = F_1 \times F_2$$

$$\delta((q_1, q_2), \sigma) = (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma))$$

Intuition: $\mathcal{N}_1 \times \mathcal{N}_2$ executes $\mathcal{N}_1$ and $\mathcal{N}_2$ in parallel.
$\mathcal{L}(\mathcal{N}_1 \times \mathcal{N}_2) = \mathcal{L}(\mathcal{N}_1) \cap \mathcal{L}(\mathcal{N}_2)$.

- $\mathcal{D}_1$ recognizes the traces with an even number of b's
- $\mathcal{D}_2$ recognizes the traces with at least an occurrence of a
- The product automaton $\mathcal{D}_1 \times \mathcal{D}_2$.

SAPIENZA UNIVERSITÀ DI ROMA

Regular expressions

$$\alpha := \varepsilon \mid a \mid \alpha \cdot \alpha \mid \alpha + \alpha \mid \alpha^*$$

Every regular expression α denotes a language $\mathcal{L}(\alpha)$.

- The traces ending with a b $\qquad (a+b)^* \cdot b$
- The traces with an a on every odd index $\qquad (a+b) \cdot (a \cdot (a+b))^*$
- The traces with an odd number of a $\qquad b^* \cdot (a \cdot b^*) \cdot ((a \cdot b^*) \cdot (a \cdot b^*))^*$

Regular expressions

$$\alpha := \varepsilon \mid a \mid \alpha \cdot \alpha \mid \alpha + \alpha \mid \alpha^*$$

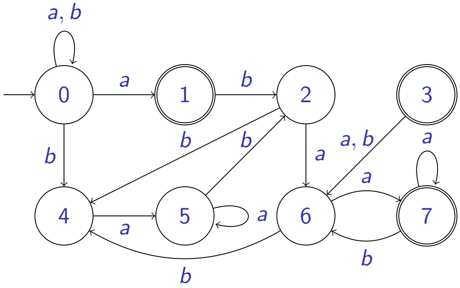
Every regular expression α denotes a language $\mathcal{L}(\alpha)$.

- The traces ending with a b — $(a + b)^* \cdot b$
- The traces with an a on every odd index — $(a + b) \cdot (a \cdot (a + b))^*$
- The traces with an odd number of a — $b^* \cdot (a \cdot b^*) \cdot ((a \cdot b^*) \cdot (a \cdot b^*))^*$

Theorem

1. *For every regular expression α, there exists a NFA $\mathcal{N}_\alpha$ such that $\mathcal{L}(\alpha) = \mathcal{L}(\mathcal{N}_\alpha)$.*

2. *For every NFA $\mathcal{N}$, there exists a regular expression $\alpha_\mathcal{N}$ such that $\mathcal{L}(\alpha_\mathcal{N}) = \mathcal{L}(\mathcal{N})$.*



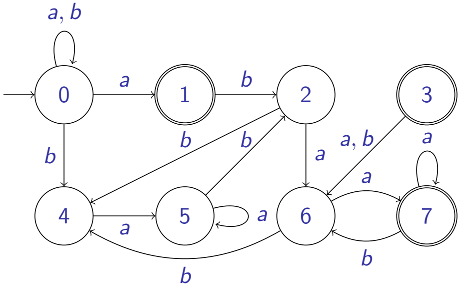
Question

Given a NFA $\mathcal{N}$, decide whether

$$\mathcal{L}(\mathcal{N}) \stackrel{?}{\neq} \emptyset$$

Does a trace w accepted by $\mathcal{N}$ exist?

The nonemptiness problem for NFA



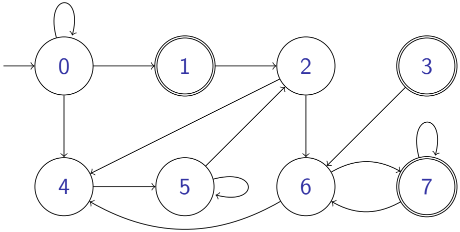
Question

Given a NFA $\mathcal{N}$, decide whether

$$\mathcal{L}(\mathcal{N}) \overset{?}{\neq} \emptyset$$

Does a trace w accepted by $\mathcal{N}$ exist?

Observation: a trace w is accepted by $\mathcal{N}$ iff there exists a run whose path starts in 0 and ends in a final state F.



Question

Given a NFA $\mathcal{N}$, decide whether

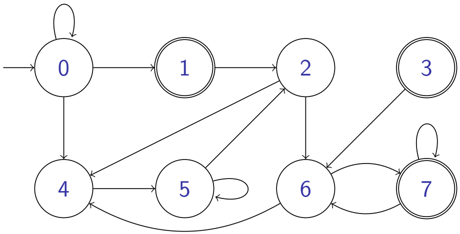
$$\mathcal{L}(\mathcal{N}) \overset{?}{\neq} \emptyset$$

Does a trace w accepted by $\mathcal{N}$ exist?

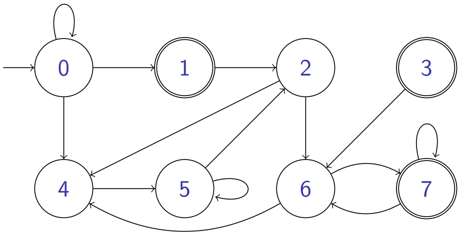
Observation: a trace w is accepted by $\mathcal{N}$ iff there exists a run whose path starts in 0 and ends in a final state F.

Solution: Nonemptiness of NFAs reduces to reachability over graphs.

Reachability with fix-point theory

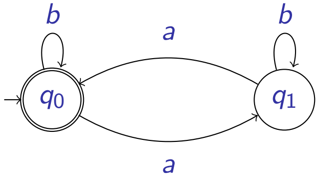


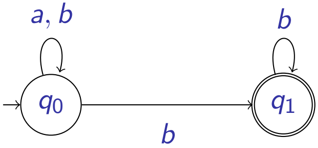
Reachability with fix-point theory



$$\mathrm{Reach}(F) = \mu\mathcal{Z}.(F \vee \langle \mathrm{next} \rangle \mathcal{Z})$$

Deterministic (DBA) and nondeterministic (NBA) Büchi automata are of the same type of DFA and NFA.

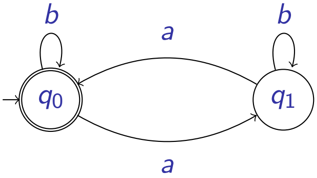


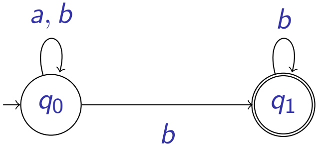


However, they read infinite traces $w \in \Sigma^\omega$.

As there is no last state in the corresponding runs ρ, the acceptance condition is to visit a final state in F infinitely many times.

Deterministic (DBA) and nondeterministic (NBA) Büchi automata are of the same type of DFA and NFA.



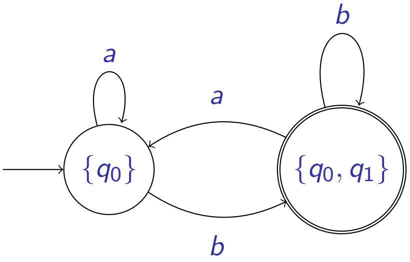


However, they read infinite traces $w \in \Sigma^\omega$.

As there is no last state in the corresponding runs ρ, the acceptance condition is to visit a final state in F infinitely many times.

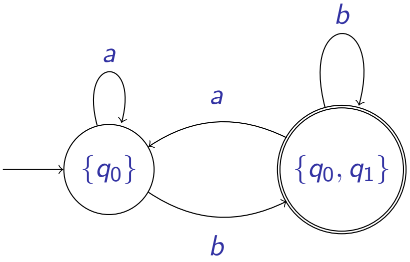
What are the languages recognized by the DBA and NBA depicted above?

Subset construction no longer works



What is the infinite trace language accepted by this subset construction automaton?

Subset construction no longer works



What is the infinite trace language accepted by this subset construction automaton?
The symbol b occurs infinitely many times (but also a might!) $(\Sigma^* \cdot b)^\omega$.

NBA cannot be determinized

SAPIENZA
UNIVERSITÀ DI ROMA

Theorem

The language $L = \{w \in \Sigma^\omega : w \text{ contains finitely many } a's\}$ can be recognized by a NBA but not by any DBA.

Corollary

NBAs are strictly more expressive than DBAs.

Theorem

For a given NBA $\mathcal{N}$, there exists a NBA $\overline{\mathcal{N}}$ such that $\mathcal{L}(\overline{\mathcal{N}}) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{N})$.

SAPIENZA
UNIVERSITÀ DI ROMA

Theorem

For a given NBA $\mathcal{N}$, there exists a NBA $\overline{\mathcal{N}}$ such that $\mathcal{L}(\overline{\mathcal{N}}) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{N})$.

However, the current techniques for the construction of $\overline{\mathcal{N}}$ are not trivial.

An entire research area in Formal Methods has been tackling this problem for many decades.

Luckily, we are not going to need this in our course.
Just note that, as for NFAs, there is an unavoidable exponential blow-up.

SAPIENZA
UNIVERSITÀ DI ROMA

Union construction

Take two NBAs $\mathcal{N}_1 = \langle Q_1, \Sigma, I_1, \delta_1, F_1 \rangle$ and $\mathcal{N}_2 = \langle Q_2, \Sigma, I_2, \delta_2, F_2 \rangle$ defined over Σ.
The union automaton $\mathcal{N}_1 \cup \mathcal{N}_2 = \langle Q, \Sigma, I, \delta, F \rangle$ is defined as:

SAPIENZA
UNIVERSITÀ DI ROMA

Union construction

Take two NBAs $\mathcal{N}_1 = \langle Q_1, \Sigma, I_1, \delta_1, F_1 \rangle$ and $\mathcal{N}_2 = \langle Q_2, \Sigma, I_2, \delta_2, F_2 \rangle$ defined over Σ. The union automaton $\mathcal{N}_1 \cup \mathcal{N}_2 = \langle Q, \Sigma, I, \delta, F \rangle$ is defined as:

$$Q = Q_1 \cup Q_2$$

$$I = I_1 \cup I_2$$

$$F = F_1 \cup F_2$$

$$\delta(q, \sigma) = \begin{cases} \delta_1(q, \sigma), & q \in Q_1 \\ \delta_2(q, \sigma), & q \in Q_2 \end{cases}$$

SAPIENZA UNIVERSITÀ DI ROMA

Union construction

Take two NBAs $\mathcal{N}_1 = \langle Q_1, \Sigma, I_1, \delta_1, F_1 \rangle$ and $\mathcal{N}_2 = \langle Q_2, \Sigma, I_2, \delta_2, F_2 \rangle$ defined over Σ. The union automaton $\mathcal{N}_1 \cup \mathcal{N}_2 = \langle Q, \Sigma, I, \delta, F \rangle$ is defined as:

$$Q = Q_1 \cup Q_2$$

$$I = I_1 \cup I_2$$

$$F = F_1 \cup F_2$$

$$\delta(q, \sigma) = \begin{cases} \delta_1(q, \sigma), & q \in Q_1 \\ \delta_2(q, \sigma), & q \in Q_2 \end{cases}$$

The union automaton guesses at the beginning whether to follow either $\mathcal{N}_1$ or $\mathcal{N}_2$.
Being on finite or infinite traces does not make any difference.

SAPIENZA UNIVERSITÀ DI ROMA

Union construction

Take two NBAs $\mathcal{N}_1 = \langle Q_1, \Sigma, I_1, \delta_1, F_1 \rangle$ and $\mathcal{N}_2 = \langle Q_2, \Sigma, I_2, \delta_2, F_2 \rangle$ defined over Σ.
The union automaton $\mathcal{N}_1 \cup \mathcal{N}_2 = \langle Q, \Sigma, I, \delta, F \rangle$ is defined as:

$$Q = Q_1 \cup Q_2$$

$$I = I_1 \cup I_2$$

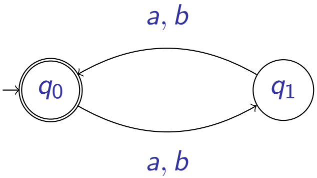
$$F = F_1 \cup F_2$$

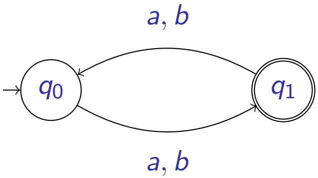
$$\delta(q, \sigma) = \begin{cases} \delta_1(q, \sigma), & q \in Q_1 \\ \delta_2(q, \sigma), & q \in Q_2 \end{cases}$$

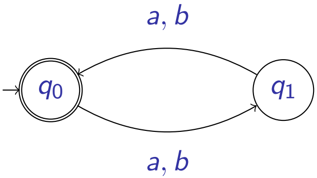
The union automaton guesses at the beginning whether to follow either $\mathcal{N}_1$ or $\mathcal{N}_2$.
Being on finite or infinite traces does not make any difference.
$\mathcal{L}(\mathcal{N}_1 \cup \mathcal{N}_2) = \mathcal{L}(\mathcal{N}_1) \cup \mathcal{L}(\mathcal{N}_2)$

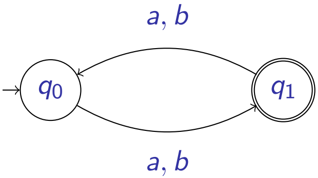
Closure properties

Intersection: synchronous product does not work









$$\mathcal{L}(\mathcal{D}_1 \times \mathcal{D}_2) = \emptyset!$$

We need a more clever way to deal with language intersection.

SAPIENZA
UNIVERSITÀ DI ROMA

Another product construction

Take two NBAs $\mathcal{N}_1 = \langle Q_1, \Sigma, I_1, \delta_1, F_1 \rangle$ and $\mathcal{N}_2 = \langle Q_2, \Sigma, I_2, \delta_2, F_2 \rangle$ defined over Σ.
The product automaton $\mathcal{N}_1 \otimes \mathcal{N}_2 = \langle Q, \Sigma, I, \delta, F \rangle$ is defined as:

SAPIENZA UNIVERSITÀ DI ROMA

Another product construction

Take two NBAs $\mathcal{N}_1 = \langle Q_1, \Sigma, I_1, \delta_1, F_1\rangle$ and $\mathcal{N}_2 = \langle Q_2, \Sigma, I_2, \delta_2, F_2\rangle$ defined over Σ.
The product automaton $\mathcal{N}_1 \otimes \mathcal{N}_2 = \langle Q, \Sigma, I, \delta, F\rangle$ is defined as:

$$Q = Q_1 \times Q_2 \times \{1, 2\}$$

$$I = I_1 \times I_2 \times \{1\}$$

$$F = F_1 \times Q_2 \times \{1\}$$

$$\delta((q_1, q_2, 1), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma), 1), & \text{if } q_1 \notin F_1 \\ (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma), 2), & \text{if } q_1 \in F_1 \end{cases}$$

$$\delta((q_1, q_2, 2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma), 2), & \text{if } q_2 \notin F_2 \\ (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma), 1), & \text{if } q_2 \in F_2 \end{cases}$$

Another product construction

Take two NBAs $\mathcal{N}_1 = \langle Q_1, \Sigma, I_1, \delta_1, F_1 \rangle$ and $\mathcal{N}_2 = \langle Q_2, \Sigma, I_2, \delta_2, F_2 \rangle$ defined over Σ.
The product automaton $\mathcal{N}_1 \otimes \mathcal{N}_2 = \langle Q, \Sigma, I, \delta, F \rangle$ is defined as:

$$Q = Q_1 \times Q_2 \times \{1, 2\}$$

$$I = I_1 \times I_2 \times \{1\}$$

$$F = F_1 \times Q_2 \times \{1\}$$

$$\delta((q_1, q_2, 1), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma), 1), & \text{if } q_1 \notin F_1 \\ (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma), 2), & \text{if } q_1 \in F_1 \end{cases}$$

$$\delta((q_1, q_2, 2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma), 2), & \text{if } q_2 \notin F_2 \\ (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma), 1), & \text{if } q_2 \in F_2 \end{cases}$$

$\mathcal{N}_1 \otimes \mathcal{N}_2$ switches the index every time a corresponding final state is found.

Another product construction

Take two NBAs $\mathcal{N}_1 = \langle Q_1, \Sigma, I_1, \delta_1, F_1 \rangle$ and $\mathcal{N}_2 = \langle Q_2, \Sigma, I_2, \delta_2, F_2 \rangle$ defined over Σ.
The product automaton $\mathcal{N}_1 \otimes \mathcal{N}_2 = \langle Q, \Sigma, I, \delta, F \rangle$ is defined as:

$$Q = Q_1 \times Q_2 \times \{1, 2\}$$

$$I = I_1 \times I_2 \times \{1\}$$

$$F = F_1 \times Q_2 \times \{1\}$$

$$\delta((q_1, q_2, 1), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma), 1), & \text{if } q_1 \notin F_1 \\ (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma), 2), & \text{if } q_1 \in F_1 \end{cases}$$

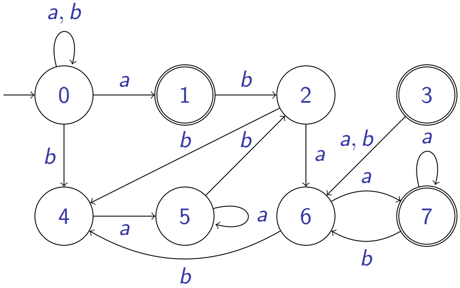
$$\delta((q_1, q_2, 2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma), 2), & \text{if } q_2 \notin F_2 \\ (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma), 1), & \text{if } q_2 \in F_2 \end{cases}$$

$\mathcal{N}_1 \otimes \mathcal{N}_2$ switches the index every time a corresponding final state is found.
$\mathcal{L}(\mathcal{N}_1 \otimes \mathcal{N}_2) = \mathcal{L}(\mathcal{N}_1) \cap \mathcal{L}(\mathcal{N}_2)$.

A language is called ω-regular if it the union of expressions of the form $\alpha \cdot \beta^\omega$ with α and β being regular languages.

Theorem

1. For every ω-regular language L, there exists a NBA $\mathcal{N}_L$ such that $\mathcal{L}(\mathcal{N}) = L$.
2. For every NBA $\mathcal{N}$, the language $\mathcal{L}(\mathcal{N})$ is ω-regular.

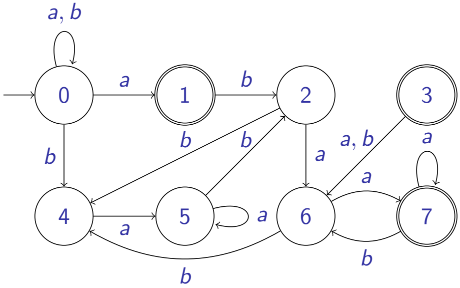


Question

Given a NBA $\mathcal{N}$, decide whether

$$\mathcal{L}(\mathcal{N}) \overset{?}{\neq} \emptyset$$

Does a trace w accepted by $\mathcal{N}$ exist?



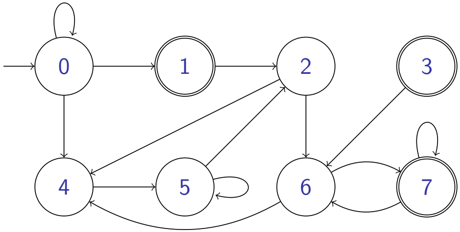
Question

Given a NBA $\mathcal{N}$, decide whether

$$\mathcal{L}(\mathcal{N}) \stackrel{?}{\neq} \emptyset$$

Does a trace w accepted by $\mathcal{N}$ exist?

Observation: a trace w is accepted by $\mathcal{N}$ iff there exists a run whose path starts in 0 and visits a final state in F infinitely many times.



Question

Given a NBA $\mathcal{N}$, decide whether

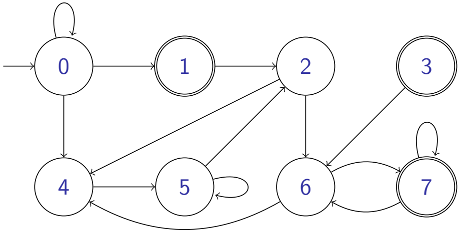
$$\mathcal{L}(\mathcal{N}) \overset{?}{\neq} \emptyset$$

Does a trace w accepted by $\mathcal{N}$ exist?

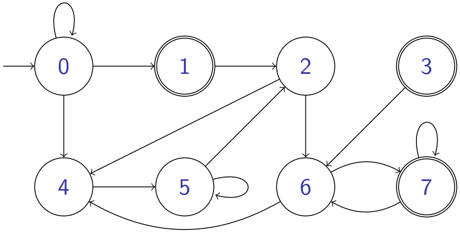
Observation: a trace w is accepted by $\mathcal{N}$ iff there exists a run whose path starts in 0 and visits a final state in F infinitely many times.

Solution: Nonemptiness of NBAs reduces to recurrent reachability over graphs.

Recurrent reachability with fix-point theory



Recurrent reachability with fix-point theory



$$\begin{aligned}\mathrm{Buchi}(F) &= \nu\mathcal{Y}.(\mathrm{Reach}(F \wedge \langle \mathrm{next}\rangle \mathcal{Y}))\\ &= \nu\mathcal{Y}.(\mu\mathcal{Z}.((\underbrace{F \wedge \langle \mathrm{next}\rangle \mathcal{Y}}_{\text{Nested fix-point}}) \vee \langle \mathrm{next}\rangle \mathcal{Z}))\end{aligned}$$

A Generalized Nondeterministic Büchi Automaton (GNBA) is a tuple $\mathcal{N} = \langle Q, \Sigma, I, \delta, \mathcal{F} \rangle$ where everything is as for a standard NBA except that

$$\mathcal{F} = (F_1, F_2, \ldots, F_n)$$

A run ρ in $\mathcal{N}$ is accepting iff it visits every F_i infinitely often.

Theorem

It holds that $\mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{N}_1 \otimes \ldots \otimes \mathcal{N}_n)$, where $\mathcal{N}_i = \langle Q, \Sigma, q_0, \delta, F_i \rangle$.

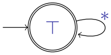
Theorem

For an LTL formula φ, we can construct a generalized nondeterministic Büchi automaton $\mathcal{N}_\varphi = \langle Q, \Sigma, I, \delta, \mathcal{F} \rangle$ such that $\mathcal{L}(\mathcal{N}_\varphi) = \mathcal{L}(\varphi)$.

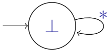
We will now look into the details on the construction of $\mathcal{N}_\varphi$.

Construction Intuition
Boolean cases

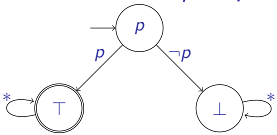
Automaton for $\varphi = \top$



Automaton for $\varphi = \bot$



Automaton for $\varphi = p$

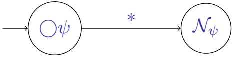


Automaton for $\varphi = \neg\psi$: $\overline{\mathcal{N}_\psi}$

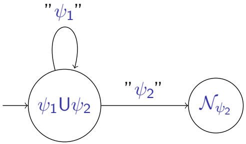
Automaton for $\varphi = \psi_1 \wedge \psi_2$: $\mathcal{N}_{\psi_1} \otimes \mathcal{N}_{\psi_2}$

Automaton for $\varphi = \psi_1 \vee \psi_2$: $\mathcal{N}_{\psi_1} \cup \mathcal{N}_{\psi_2}$

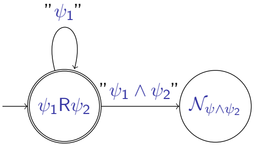
Automaton for $\varphi = \bigcirc\psi$



Automaton for $\varphi = \psi_1 \mathsf{U} \psi_2$



Automaton for $\varphi = \psi_1 \mathsf{R} \psi_2$



Definition (Fischer-Ladner Closure)

For a given LTL formula φ, the FS-closure of φ, denoted $\mathsf{cl}(\varphi)$ is the set of subformulas of φ and their negation (where $\neg\neg\psi = \psi$). It is (recursively) defined as follows:

SAPIENZA UNIVERSITÀ DI ROMA

Definition (Fischer-Ladner Closure)

For a given LTL formula φ, the FS-closure of φ, denoted $\mathsf{cl}(\varphi)$ is the set of subformulas of φ and their negation (where $\neg\neg\psi = \psi$). It is (recursively) defined as follows:

- $\varphi \in \mathsf{cl}(\varphi)$
- If $\psi \in \mathsf{cl}(\varphi)$ then $\neg\psi \in \mathsf{cl}(\varphi)$
- If $\psi_1 \wedge \psi_2 \in \mathsf{cl}(\varphi)$ then $\psi_1, \psi_2 \in \mathsf{cl}(\varphi)$
- If $\bigcirc\psi \in \mathsf{cl}(\varphi)$ then, $\psi \in \mathsf{cl}(\varphi)$
- If $\psi_1 \mathsf{U} \psi_2 \in \mathsf{cl}(\varphi)$ then $\psi_1, \psi_2 \in \mathsf{cl}(\varphi)$

SAPIENZA UNIVERSITÀ DI ROMA

Definition (Fischer-Ladner Closure)

For a given LTL formula φ, the FS-closure of φ, denoted $\mathsf{cl}(\varphi)$ is the set of subformulas of φ and their negation (where $\neg\neg\psi = \psi$). It is (recursively) defined as follows:

- $\varphi \in \mathsf{cl}(\varphi)$
- If $\psi \in \mathsf{cl}(\varphi)$ then $\neg\psi \in \mathsf{cl}(\varphi)$
- If $\psi_1 \wedge \psi_2 \in \mathsf{cl}(\varphi)$ then $\psi_1, \psi_2 \in \mathsf{cl}(\varphi)$
- If $\bigcirc\psi \in \mathsf{cl}(\varphi)$ then, $\psi \in \mathsf{cl}(\varphi)$
- If $\psi_1 \mathsf{U} \psi_2 \in \mathsf{cl}(\varphi)$ then $\psi_1, \psi_2 \in \mathsf{cl}(\varphi)$

For example, $\varphi = p \wedge ((\bigcirc p)\mathsf{U} q)$

$$\mathsf{cl}(\varphi) = \{p \wedge ((\bigcirc p)\mathsf{U} q), \neg(p \wedge ((\bigcirc p)\mathsf{U} q)), p, \neg p, (\bigcirc p)\mathsf{U} q, \neg((\bigcirc p)\mathsf{U} q), \bigcirc p, \neg\bigcirc p, q, \neg q\}$$

SAPIENZA UNIVERSITÀ DI ROMA

Atoms

A set $\alpha \subset \mathsf{cl}(\varphi)$ is called atom if it is maximally consistent, that is:

- For all $\psi \in \mathsf{cl}(\varphi)$ either $\psi \in \alpha$ or $\neg\psi \in \alpha$ (maximality)
- $\psi_1 \wedge \psi_2 \in \alpha$ iff $\psi_1, \psi_2 \in \alpha$ (consistency)

By $\mathrm{Atoms}(\varphi) = \{\alpha \subset \mathsf{cl}(\varphi) : \alpha \text{ is an atom } \}$

SAPIENZA
UNIVERSITÀ DI ROMA

Atoms

A set $\alpha \subset \mathsf{cl}(\varphi)$ is called atom if it is maximally consistent, that is:

- For all $\psi \in \mathsf{cl}(\varphi)$ either $\psi \in \alpha$ or $\neg\psi \in \alpha$ (maximality)
- $\psi_1 \wedge \psi_2 \in \alpha$ iff $\psi_1, \psi_2 \in \alpha$ (consistency)

By $\mathrm{Atoms}(\varphi) = \{\alpha \subset \mathsf{cl}(\varphi) : \alpha \text{ is an atom } \}$

The space set of $\mathcal{N}_\varphi$ is defined as $Q = \mathrm{Atoms}(\varphi)$.

Intuition: a state α in the automaton carries out the information on which subformulas of φ need to be satisfied when the computation starts from α itself.

SAPIENZA UNIVERSITÀ DI ROMA

Atoms

A set $\alpha \subset \mathsf{cl}(\varphi)$ is called atom if it is maximally consistent, that is:

- For all $\psi \in \mathsf{cl}(\varphi)$ either $\psi \in \alpha$ or $\neg\psi \in \alpha$ (maximality)
- $\psi_1 \wedge \psi_2 \in \alpha$ iff $\psi_1, \psi_2 \in \alpha$ (consistency)

By $\mathrm{Atoms}(\varphi) = \{\alpha \subset \mathsf{cl}(\varphi) : \alpha \text{ is an atom }\}$

The space set of $\mathcal{N}_\varphi$ is defined as $Q = \mathrm{Atoms}(\varphi)$.

Intuition: a state α in the automaton carries out the information on which subformulas of φ need to be satisfied when the computation starts from α itself.

Observation: the size of $\mathcal{N}_\varphi$ is exponential in the length of φ. Once again, this exponential blow-up is unavoidable.

Initial states, transition function, and final states of the automaton

Initial states

$I = \{\alpha \in Q : \varphi \in \alpha\}$ (every atom α containing φ is an initial state)

SAPIENZA UNIVERSITÀ DI ROMA

Initial states

$I = \{\alpha \in Q : \varphi \in \alpha\}$ (every atom α containing φ is an initial state)

Transition function

Take two atoms α and α' together with $\sigma \in \Sigma = 2^{\mathrm{Prop}}$.
We say that $\alpha' \in \delta(\alpha, \sigma)$ if

- $\sigma = \alpha \cap \mathrm{Prop}$ (Advance only if you read something consistent)
- $\bigcirc\psi \in \alpha$ iff $\psi \in \alpha'$ (Check ψ at the next stage)
- $\psi_1 \mathsf{U} \psi_2 \in \alpha$ iff either $\psi_2 \in \alpha$ or both $\psi_1 \in \alpha$ and $\psi_1 \mathsf{U} \psi_2 \in \alpha'$ (Keep checking U if needed)

SAPIENZA
UNIVERSITÀ DI ROMA

Initial states

$I = \{\alpha \in Q : \varphi \in \alpha\}$ (every atom α containing φ is an initial state)

Transition function

Take two atoms α and α' together with $\sigma \in \Sigma = 2^{\mathrm{Prop}}$.
We say that $\alpha' \in \delta(\alpha, \sigma)$ if

- $\sigma = \alpha \cap \mathrm{Prop}$ (Advance only if you read something consistent)
- $\bigcirc\psi \in \alpha$ iff $\psi \in \alpha'$ (Check ψ at the next stage)
- $\psi_1 \mathsf{U} \psi_2 \in \alpha$ iff either $\psi_2 \in \alpha$ or both $\psi_1 \in \alpha$ and $\psi_1 \mathsf{U} \psi_2 \in \alpha'$ (Keep checking U if needed)

Final states

$\mathcal{F} = (F_{\psi_1 \cup \psi_2})_{\psi_1 \cup \psi_2 \in \mathsf{cl}(\varphi)}$ with
$F_{\psi_1 \mathsf{U} \psi_2} = \{\alpha \in Q : \psi_2 \in \alpha \text{ or } \neg(\psi_1 \mathsf{U} \psi_2) \in \alpha\}$, for each $\psi_1 \mathsf{U} \psi_2 \in \mathsf{cl}(\varphi)$

Several constructions of $\mathcal{N}_\varphi$ are available in the literature, including online tools:

- https://spot.lrde.epita.fr/app/
- http://www.lsv.fr/ gastin/ltl2ba/index.php
- https://owl.model.in.tum.de/try/

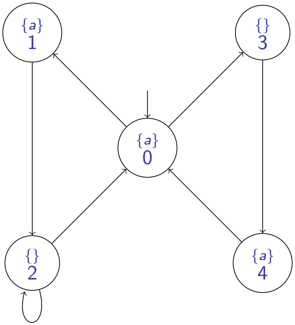
These constructions are always hard to handle manually, as they provide exponentially sized automata.
However, the general construction is not always necessary in practice.

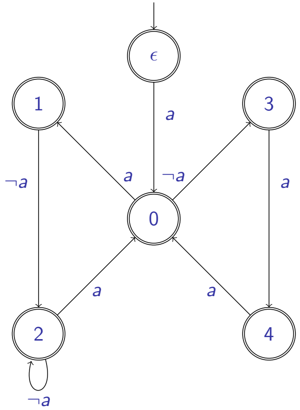
- $p \mathrm{U} q$
- $\Diamond p$
- $\Box p$
- $q \mathrm{U} (\bigcirc \bigcirc p)$
- $\Box (p \rightarrow \Diamond q)$
- $\Box \Diamond p$
- $\Diamond \Box p$
- $\Box \Diamond p \land \Box \Diamond q$

A labeled transition system $\mathcal{T} = \langle S, S_0, E(\subseteq S \times S), \lambda\rangle$ with $\lambda : S \to 2^{\mathrm{Prop}}$ is turned into a NBA $\mathcal{N}_{\mathcal{T}} = \langle \Sigma, Q, Q_0, \delta, F\rangle$ with:

- $\Sigma = 2^{\mathrm{Prop}}$
- $Q = S \cup \{\epsilon\}$
- $Q_0 = \{\epsilon\}$
- $\delta(\epsilon, \sigma) = \{s \in S_0 : \sigma = \lambda(s_0)\}$
  $\delta(s, \sigma) = \{s' \in S : (s, s') \in E \text{ and } \sigma = \lambda(s')\}$
- $F = Q$

The labeling of states is pushed backward to the incoming edges.
A root state is included to push the initial state labels backward.
Every state is accepting.





Theorem

For every labeled transition system $\mathcal{T}$, the automaton $\mathcal{N}_{\mathcal{T}}$ recognizes all and only those infinite traces that are generated by $\mathcal{T}$.

SAPIENZA
UNIVERSITÀ DI ROMA

Problem

For a given LTS $\mathcal{T}$ and an LTL formula φ, Model Checking is the problem of verifying that all the executions of $\mathcal{T}$ satisfy φ. Equivalently

$$\mathcal{T} \models \varphi \Longleftrightarrow \mathcal{L}(\mathcal{T}) \subseteq \mathcal{L}(\varphi)$$

SAPIENZA UNIVERSITÀ DI ROMA

Problem

For a given LTS $\mathcal{T}$ and an LTL formula φ, Model Checking is the problem of verifying that all the executions of $\mathcal{T}$ satisfy φ. Equivalently

$$\mathcal{T} \models \varphi \iff \mathcal{L}(\mathcal{T}) \subseteq \mathcal{L}(\varphi)$$

Automata-Theoretic Approach

- $\mathcal{T} \to \mathcal{N}_{\mathcal{T}}$ $\qquad \mathcal{L}(\mathcal{T}) = \mathcal{L}(\mathcal{N}_{\mathcal{T}})$
- $\varphi \to \mathcal{N}_{\varphi}$ $\qquad \mathcal{L}(\varphi) = \mathcal{L}(\mathcal{N}_{\varphi})$

SAPIENZA UNIVERSITÀ DI ROMA

Problem

For a given LTS $\mathcal{T}$ and an LTL formula φ, Model Checking is the problem of verifying that all the executions of $\mathcal{T}$ satisfy φ. Equivalently

$$\mathcal{T} \models \varphi \iff \mathcal{L}(\mathcal{T}) \subseteq \mathcal{L}(\varphi)$$

Automata-Theoretic Approach

- $\mathcal{T} \to \mathcal{N}_{\mathcal{T}}$ $\qquad$ $\mathcal{L}(\mathcal{T}) = \mathcal{L}(\mathcal{N}_{\mathcal{T}})$
- $\varphi \to \mathcal{N}_{\varphi}$ $\qquad$ $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{N}_{\varphi})$
- $\mathcal{L}(\mathcal{T}) \subseteq \mathcal{L}(\varphi) \iff \mathcal{L}(\mathcal{N}_{\mathcal{T}}) \subseteq \mathcal{L}(\mathcal{N}_{\varphi})$

SAPIENZA
UNIVERSITÀ DI ROMA

Problem

For a given LTS $\mathcal{T}$ and an LTL formula φ, Model Checking is the problem of verifying that all the executions of $\mathcal{T}$ satisfy φ. Equivalently

$$\mathcal{T} \models \varphi \iff \mathcal{L}(\mathcal{T}) \subseteq \mathcal{L}(\varphi)$$

Automata-Theoretic Approach

- $\mathcal{T} \to \mathcal{N}_{\mathcal{T}}$ $\qquad \mathcal{L}(\mathcal{T}) = \mathcal{L}(\mathcal{N}_{\mathcal{T}})$
- $\varphi \to \mathcal{N}_{\varphi}$ $\qquad \mathcal{L}(\varphi) = \mathcal{L}(\mathcal{N}_{\varphi})$
- $\mathcal{L}(\mathcal{T}) \subseteq \mathcal{L}(\varphi) \iff \mathcal{L}(\mathcal{N}_{\mathcal{T}}) \subseteq \mathcal{L}(\mathcal{N}_{\varphi}) \iff \mathcal{L}(\mathcal{N}_{\mathcal{T}}) \cap \mathcal{L}(\overline{\mathcal{N}_{\varphi}}) = \emptyset$

Ouch! We need to complement a NBA! Can we avoid it?

SAPIENZA
UNIVERSITÀ DI ROMA

Problem

For a given LTS $\mathcal{T}$ and an LTL formula φ, Model Checking is the problem of verifying that all the executions of $\mathcal{T}$ satisfy φ. Equivalently

$$\mathcal{T} \models \varphi \iff \mathcal{L}(\mathcal{T}) \subseteq \mathcal{L}(\varphi)$$

Automata-Theoretic Approach

- $\mathcal{T} \to \mathcal{N}_{\mathcal{T}}$ $\qquad \mathcal{L}(\mathcal{T}) = \mathcal{L}(\mathcal{N}_{\mathcal{T}})$
- $\varphi \to \mathcal{N}_{\varphi}$ $\qquad \mathcal{L}(\varphi) = \mathcal{L}(\mathcal{N}_{\varphi})$
- $\mathcal{L}(\mathcal{T}) \subseteq \mathcal{L}(\varphi) \iff \mathcal{L}(\mathcal{N}_{\mathcal{T}}) \subseteq \mathcal{L}(\mathcal{N}_{\varphi}) \iff \mathcal{L}(\mathcal{N}_{\mathcal{T}}) \cap \mathcal{L}(\overline{\mathcal{N}_{\varphi}}) = \emptyset$

Ouch! We need to complement a NBA! Can we avoid it? $\qquad \mathcal{L}(\overline{\mathcal{N}_{\varphi}}) = \mathcal{L}(\mathcal{N}_{\neg\varphi})$

- Consider a model $\mathcal{T}$ and an LTL property φ
- $\mathcal{T} \models \varphi$ if for all the paths π of $\mathcal{T}$, it holds that $\pi \models \varphi$, namely if $\pi \in \mathcal{L}(\varphi)$.
- Equivalently, $\mathcal{T}$ admits no path π such that $\pi \models \neg\varphi$ (no counterexample)
- More formally

$$\begin{aligned}\mathcal{T} \models \varphi &\Leftrightarrow \mathcal{L}(\mathcal{T}) \subseteq \mathcal{L}(\varphi) \\ &\Leftrightarrow \mathcal{L}(\mathcal{T}) \cap \overline{\mathcal{L}(\varphi)} = \emptyset \\ &\Leftrightarrow \mathcal{L}(\mathcal{T}) \cap \mathcal{L}(\neg\varphi) = \emptyset\end{aligned}$$

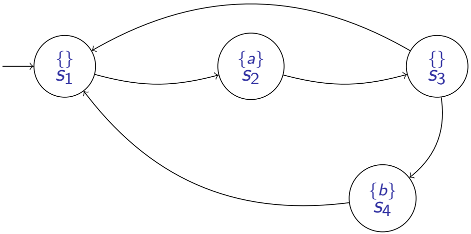
- Input:
  - a model $\mathcal{T}$ and
  - a formula φ
- Construction:
  - Construct the automaton $\mathcal{N}_{\mathcal{T}}$ from the LTS
  - Construct the automaton $\mathcal{N}_{\neg\varphi}$ from the LTL formula
  - Construct the product automaton $\mathcal{N}_{\mathcal{T},\neg\varphi} = \mathcal{N}_{\mathcal{T}} \otimes \mathcal{N}_{\neg\varphi}$
- Solve nonemptiness problem:

$$\mathcal{L}(\mathcal{N}_{\mathcal{T},\neg\varphi}) \stackrel{?}{\neq} \emptyset$$

Output:

- Yes if $\mathcal{L}(\mathcal{N}_{\mathcal{T},\neg\varphi}) = \emptyset$
- No if otherwise (and show a counterexample path $\pi \models \neg\varphi$)

Exercise



$$\bigcirc a \wedge (\Box(b \rightarrow \bigcirc a)) \wedge \Diamond a$$