Game-Theoretic Approach to Temporal Synthesis

Linear-Time Temporal Logic

Antonio Di Stasio **Giuseppe Perelli** [1] Shufang Zhu

2nd European Summer School on Artificial Intelligence
Athens (Greece) 15-19 July 2024

[1] The realization of this course was partially supported by MUR under the PRIN programme, grant B87G22000450001 (PINPOINT).

SAPIENZA UNIVERSITÀ DI ROMA

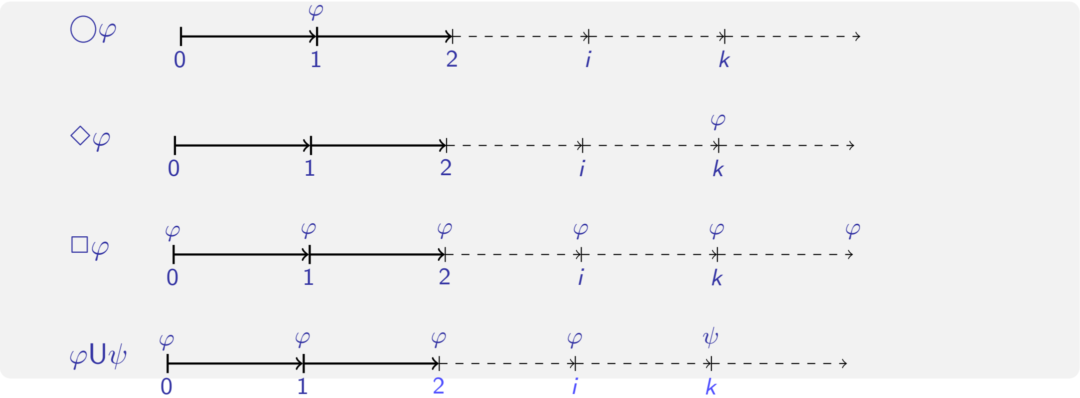
A standard language for talking about infinite state sequences.

Amir Pnueli - The Temporal Logic of Programs. - FOCS'77

$\top$	truth constant	$\bigcirc\phi$	in the next state...
p	primitive propositions	$\Diamond\phi$	will eventually be the case
$\neg\phi$	classical negation	$\Box\phi$	is always the case
$\phi \vee \psi$	classical disjunction	$\phi \mathsf{U} \psi$	ϕ until ψ
$\phi \wedge \psi$	classical conjunction	$\phi \mathsf{R} \psi$	ϕ release ψ

Minimal syntax

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi \mathsf{U} \varphi$$



LTL formulas are evaluated on infinite traces, that is, obtained from an infinite path.

The language defined by an LTL formula φ is $\mathcal{L}(\varphi) = \{w \in \Sigma^\omega : w \models \varphi\}$.

SAPIENZA UNIVERSITÀ DI ROMA

Eventually I will graduate	$\Diamond\mathtt{degree}$
The plane will never crash	$\Box\neg\mathtt{crash}$
I will eat pizza infinitely often	$\Box\Diamond\mathtt{eatPizza}$
...and they all lived happily ever after	$\Diamond\Box\mathtt{happy}$
We are not friends until you apologise	$(\neg\mathtt{friends})\mathcal{U}\mathtt{youApologise}$
Every time it is requested, a document will be printed	$\Box(\mathtt{print\_req} \rightarrow \Diamond\mathtt{print})$
The two processes are never active at the same time	$\Box\neg(\mathtt{proc}_1 \wedge \mathtt{proc}_2)$

Describe temporal modalities recursively

- $\varphi \mathsf{U} \psi \equiv \psi \vee (\varphi \wedge \bigcirc \varphi \mathsf{U} \psi)$ — $\varphi \mathsf{U} \psi$ is a "solution" of $\Psi = \psi \vee (\varphi \wedge \bigcirc \Psi)$
- $\Diamond \psi \equiv \psi \vee \bigcirc \Diamond \psi$ — $\Diamond \psi$ is a solution of $\Psi = \psi \vee \bigcirc \Psi$
- also $\Box \psi \equiv \neg \Diamond \neg \psi \equiv \psi \wedge \bigcirc \Box \psi$ — $\Box \psi$ is a solution of $\Psi = \psi \wedge \bigcirc \Psi$

SAPIENZA UNIVERSITÀ DI ROMA

Define the Release operator R in a way that the following holds:

$\varphi R \psi \equiv \neg(\neg\varphi U \neg\psi)$

it also holds that

$\varphi U \psi \equiv \neg(\neg\varphi R \neg\psi)$ (Release is dual of Until)

PNF

Positive Normal Form for LTL: for $a \in AP$

$$\varphi ::= \text{true} \mid \text{false} \mid a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \bigcirc\varphi \mid \varphi U \varphi \mid \varphi R \varphi$$

Theorem

Each LTL formula φ admits an equivalent in PNF sometimes denoted $\text{pnf}(\varphi)$

SAPIENZA UNIVERSITÀ DI ROMA

LTL_f

$$\varphi ::= A \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2 \mid \bullet\varphi \mid \Diamond\varphi \mid \Box\varphi \mid Last$$

A: **atomic** propositions

$\neg\varphi$, $\varphi_1 \wedge \varphi_2$: **boolean** connectives

$\bigcirc\varphi$: "**next step exists** and at **next** step (of the trace) φ holds"

$\varphi_1 \mathsf{U} \varphi_2$: "**eventually** φ_2 holds, and φ_1 holds **until** φ_2 does"

$\bullet\varphi \doteq \neg\bigcirc\neg\varphi$: "**if next step exists** then at **next** step φ holds" *(weak next)*

$\Diamond\varphi \doteq \top\mathsf{U}\varphi$: "$\varphi$ will **eventually** hold"

$\Box\varphi \doteq \neg\Diamond\neg\varphi$: "from current till last instant φ will **always** hold"

$Last \doteq \neg\bigcirc\top$: denotes **last** instant of trace.

LTL$_f$ semantics difference with LTL

- $\Diamond \texttt{degree}$
- $\Box \neg \texttt{crash}$
- $\Box \Diamond \texttt{eatPizza}$
- $\Diamond \Box \texttt{happy}$
- $(\neg \texttt{friends}) \mathcal{U} \texttt{youApologise}$

- 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"

SAPIENZA UNIVERSITÀ DI ROMA

- 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 problems

Recognition: Determine whether a trace w belongs to a language L.

Game-Theoretic Generation: Two players generate w. Player one wants $w \in L$.

SAPIENZA UNIVERSITÀ DI ROMA

- 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 problems

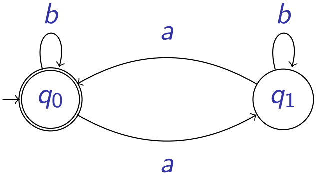
Recognition: Determine whether a trace w belongs to a language L.

Game-Theoretic Generation: Two players generate w. Player one wants $w \in L$.

Automata are computational devices used to solve language 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
- $\delta : Q \times \Sigma \to Q$ transition function
- $F \subseteq Q$ set of final states

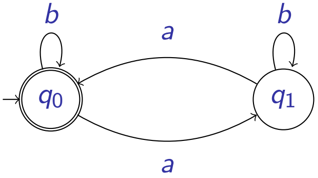


Recognizes the traces with an even number of a.

SAPIENZA UNIVERSITÀ DI ROMA

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
- $\delta : Q \times \Sigma \to Q$ transition function
- $F \subseteq Q$ set of final states



Recognizes the traces with an even number of a.

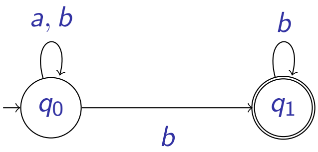
A trace $w \in \Sigma^*$ is read on $\mathcal{D}$ by starting from q_0 and following the transition function, generating a run $\rho \in Q^*$.
We say $w \in \mathcal{L}(\mathcal{D}) \subseteq \Sigma^*$ if the corresponding run ρ ends in a final state.

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

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.

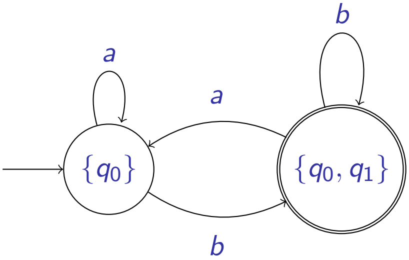
More than one run is possible on the same trace $w \in \Sigma^*$.
The automaton $\mathcal{N}$ accepts $w \in \mathcal{L}(\mathcal{N})$ if at least one run is accepting.

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:

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

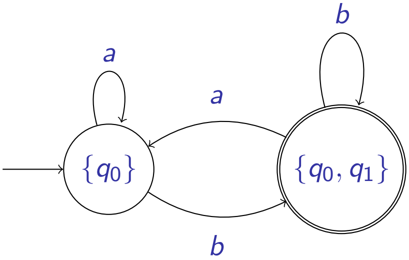


SAPIENZA UNIVERSITÀ DI ROMA

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:

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



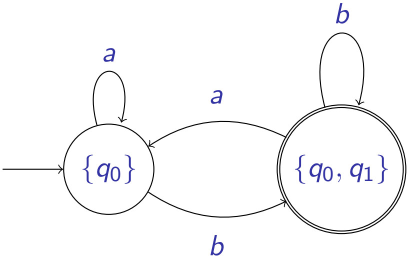
Intuition: $\mathcal{D}_{\mathcal{N}}$ runs all the possible executions of $\mathcal{N}$ in parallel.
If one of them accepts the trace in $\mathcal{N}$, then it does so in $\mathcal{D}_{\mathcal{N}}$.

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

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 $\mathcal{D}_{\mathcal{N}} \Rightarrow \overline{\mathcal{D}_{\mathcal{N}}}$

SAPIENZA UNIVERSITÀ DI ROMA

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.

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:

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:

$$Q = Q_1 \cup Q_2$$ [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:

$$Q = Q_1 \cup Q_2$$ [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.

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.
$\mathcal{L}(\mathcal{N}_1 \times \mathcal{N}_2) = \mathcal{L}(\mathcal{N}_1) \cap \mathcal{L}(\mathcal{N}_2)$.

SAPIENZA UNIVERSITÀ DI ROMA

- $\mathcal{D}_1$ 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 — $(a + b)^* \cdot b$
- The traces with an a on every odd index — $(a + b) \cdot (a \cdot (a + b))^*$
- The traces with an odd number of a — $b^* \cdot (a \cdot b^*) \cdot ((a \cdot b^*) \cdot (a \cdot b^*))^*$

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 $\mathcal{L}(\alpha)$.

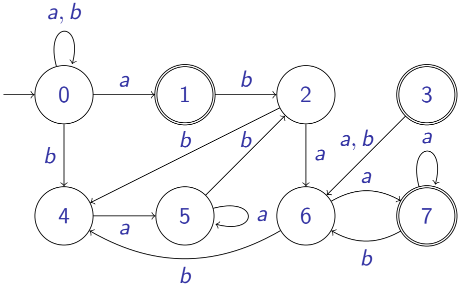
- The traces ending with a b — $(a + b)^* \cdot b$
- The traces with an a on every odd index — $(a + b) \cdot (a \cdot (a + b))^*$
- The traces with an odd number of a — $b^* \cdot (a \cdot b^*) \cdot ((a \cdot b^*) \cdot (a \cdot b^*))^*$

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 $\mathcal{L}(\alpha_\mathcal{N}) = \mathcal{L}(\mathcal{N})$.*

The nonemptiness problem for NFA

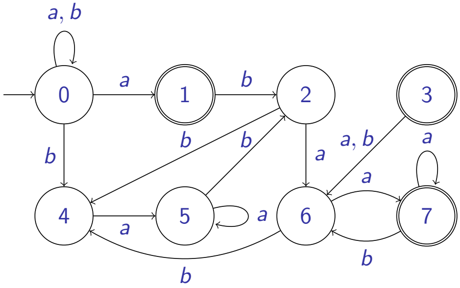


Question

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

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

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



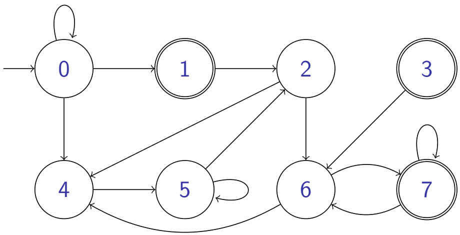
Question

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

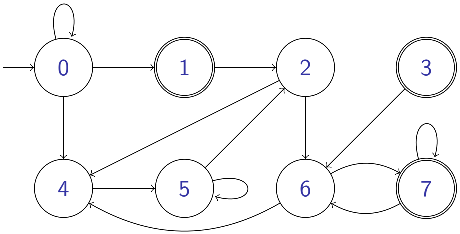
$$\mathcal{L}(\mathcal{N}) \overset{?}{\neq} \emptyset$$

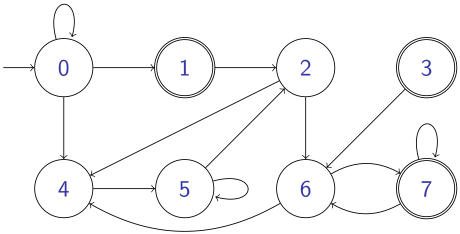
Does a trace w accepted by $\mathcal{N}$ exist?

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





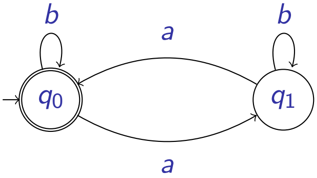
$$\text{Reach}(F) = \mu\mathcal{Z}.(F \vee \langle \text{next} \rangle \mathcal{Z})$$

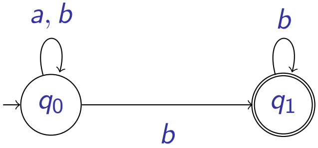
Deterministic (DBA) and nondeterministic (NBA) Büchi automata are of the same type of DFA and NFA.

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.



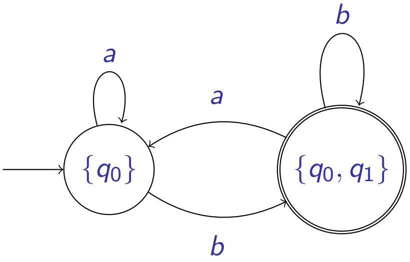


However, they read infinite traces $w \in \Sigma^\omega$.

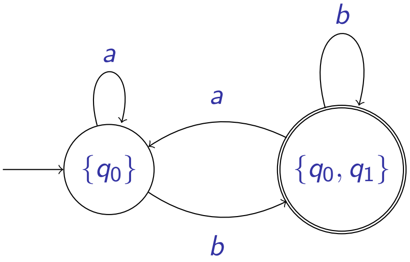
As there is no last state in the corresponding runs ρ, the acceptance condition is to visit a final state in F infinitely many times.

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?



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.

SAPIENZA
UNIVERSITÀ DI ROMA

Theorem

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

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:

$$Q = Q_1 \cup Q_2$$

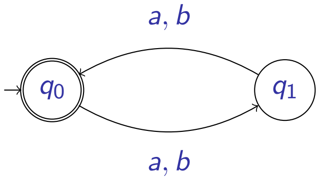
$$I = I_1 \cup I_2$$

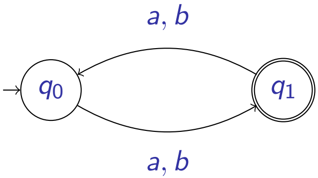
$$F = F_1 \cup F_2$$

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

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





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:

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

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

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

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

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

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:

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

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

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

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

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

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

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:

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

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

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

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

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

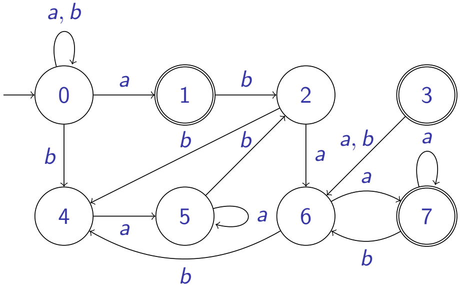
$\mathcal{N}_1 \otimes \mathcal{N}_2$ switches the index every time a corresponding final state is found.
$\mathcal{L}(\mathcal{N}_1 \otimes \mathcal{N}_2) = \mathcal{L}(\mathcal{N}_1) \cap \mathcal{L}(\mathcal{N}_2)$.

SAPIENZA UNIVERSITÀ DI ROMA

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.



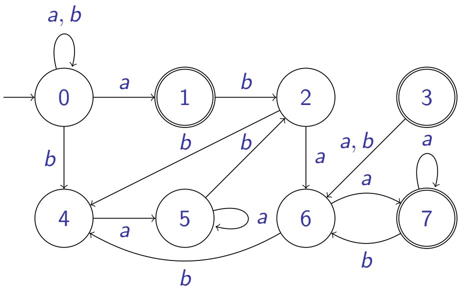
Question

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

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

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

SAPIENZA UNIVERSITÀ DI ROMA



Question

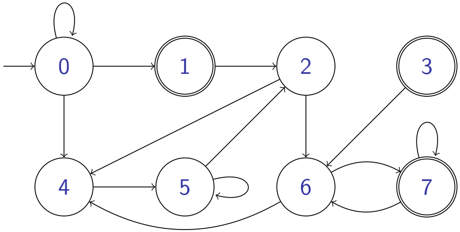
Given a NBA $\mathcal{N}$, decide whether

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

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.

SAPIENZA UNIVERSITÀ DI ROMA



Question

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

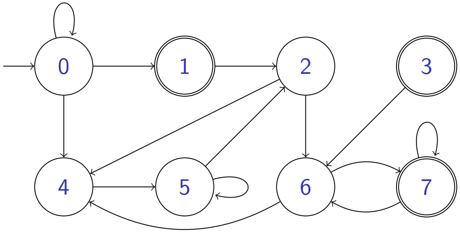
$$\mathcal{L}(\mathcal{N}) \overset{?}{\neq} \emptyset$$

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

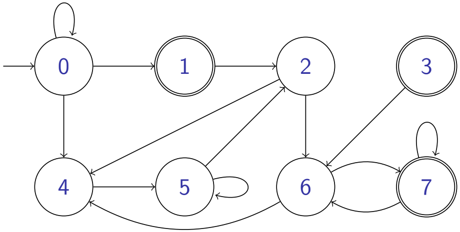
Observation: a trace w is accepted by $\mathcal{N}$ iff there exists a run whose path starts in 0 and visits a final state in F infinitely many times.

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

Recurrent reachability with fix-point theory



SAPIENZA UNIVERSITÀ DI ROMA



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

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

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

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

Theorem

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

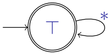
SAPIENZA UNIVERSITÀ DI ROMA

Theorem

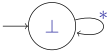
For an LTL formula φ, we can construct a generalized nondeterministic Büchi automaton $\mathcal{N}_\varphi = \langle Q, \Sigma, I, \delta, \mathcal{F} \rangle$ such that $\mathcal{L}(\mathcal{N}_\varphi) = \mathcal{L}(\varphi)$.

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

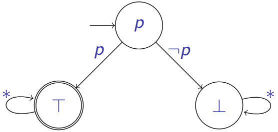
Automaton for $\varphi = \top$



Automaton for $\varphi = \bot$



Automaton for $\varphi = p$



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

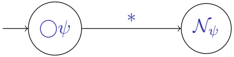
Automaton for $\varphi = \psi_1 \wedge \psi_2$: $\mathcal{N}_{\psi_1} \otimes \mathcal{N}_{\psi_2}$

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

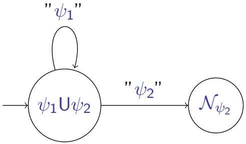
Construction Intuition

Until and Release operators

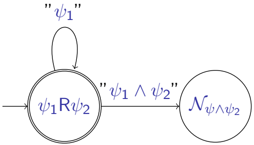
Automaton for $\varphi = \bigcirc\psi$



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



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



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:

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 $\psi_1, \psi_2 \in \mathsf{cl}(\varphi)$

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

SAPIENZA UNIVERSITÀ DI ROMA

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)

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.

SAPIENZA UNIVERSITÀ DI ROMA

A Deterministic Parity Automaton (DPA) is a tuple $\mathcal{D} = \langle Q, \Sigma, s, \delta, \alpha \rangle$ with:

- Q finite set of states
- Σ finite alphabet
- $q_0 \in Q$ initial state
- $\delta : Q \times \Sigma \to Q$ transition function
- $\alpha : Q \to \mathbb{N}$ coloring function

– $\inf(\rho) = \{\alpha(q) \in \mathbb{N} : q \text{ occurs infinitely often in } \rho\}$.
– ρ is accepting if the max value in $\inf(\rho)$ is even.

Observe: Büchi acceptance condition is a special case of parity with $\alpha^{-1}(2) = F$ and $\alpha^{-1}(1) = Q \setminus F$.

SAPIENZA UNIVERSITÀ DI ROMA

Theorem

For every Nondeterministic Büchi automaton $\mathcal{N}$, there exists a Deterministic Parity Automaton $\mathcal{D}$ of size exponential w.r.t. $\mathcal{N}$, such that

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

Theorem

For every LTL formula φ, there exists a Deterministic Parity Automaton $\mathcal{D}_\varphi$ of size double exponential w.r.t. $|\varphi|$, such that

$$\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{D}_\varphi).$$

Theorem

For every LTL_f/LDL_f formula φ, there exists a Deterministic Finite-State Automaton (DFA) $\mathcal{D}_\varphi$ of size double exponential w.r.t. $|\varphi|$, such that

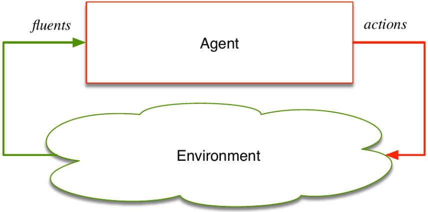
$$\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{D}_\varphi).$$

SAPIENZA UNIVERSITÀ DI ROMA

Synthesis

- Agent acts in a (nondeterministic) Environment
- Agent controls *actions*
- Environment controls *fluents*
- **Task** is given to agent
- **Task** talks both about fluents and actions
- Agent has to realize the **task** in spite of how the Environment reacts.

Agent in Environment



Agent process/behavior

Agent process/behavior (also called, "plan", "strategy", "policy", "protocol"):

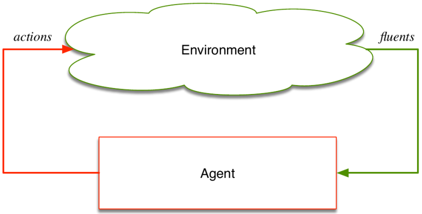
$$\sigma_a : (\mathit{fluents})^* \to \mathit{actions}$$

where

$(\mathit{fluents})^*$ denotes the **history** of what observed so far by the agent (a finite sequence of fluents configurations)

actions denotes the **next action** that the agent does

SAPIENZA UNIVERSITÀ DI ROMA



Environment process/behavior

Environment process/behavior:

$$\sigma_e : (actions)^* \to fluents$$

where

$(actions)^*$ denotes the **history** of what observed so far by the environment (a finite sequence of agent actions)

fluents denotes the **next effects** that the environment brings about.

SAPIENZA UNIVERSITÀ DI ROMA

Traces

Observe that both the agent process and the environment process:

$$\sigma_a : \quad (fluents)^* \quad \to \quad actions$$

$$\sigma_e : \quad (actions)^* \quad \to \quad fluents$$

cannot be executed in isolation.

But they **can be executed together**, generating a **trace** (sometime also call a "play"):

$$trace(\sigma_a, \sigma_e) = F_0 \cup A_0; F_1 \cup A_1; \cdots$$

SAPIENZA UNIVERSITÀ DI ROMA

- Agent controlling truth-value of set of variables $\mathcal{Y}$
- Environment controlling truth-value of set of variables $\mathcal{X}$
- An LTL (or LTL$_f$) formula φ over variables $\mathcal{X} \cup \mathcal{Y}$ specifying *correctness*

Find a strategy σ_a for the agent such that, for each strategy σ_e of the environment:

$$trace(\sigma_a, \sigma_e) \models \varphi$$

SAPIENZA
UNIVERSITÀ DI ROMA

Turn the specification into a corresponding Deterministic Automaton $\varphi \rightsquigarrow \mathcal{D}_\varphi$

Turn the Deterministic Automaton into a 2-player game $\mathcal{D}_\varphi \rightsquigarrow \mathcal{G}_{\mathcal{D}_\varphi}$

Solve the 2-player game and extract a strategy σ_0 for player 0;

Strategy σ_0 corresponds to the one σ_a for the agent in the original synthesis problem!