Department of Mathematics, Computer Science and Physics, University of Udine

The Safety Fragment of Temporal Logics on Infinite Sequences

Lesson 13

Luca Geatti
luca.geatti@uniud.it
Angelo Montanari
angelo.montanari@uniud.it

May 16th, 2024

Outline

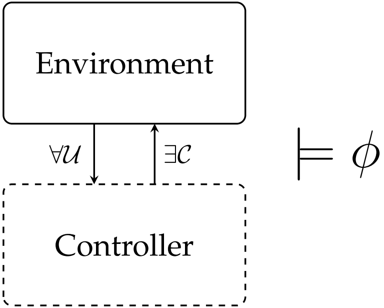
1. Background
   1. Regular and ω-regular languages
   2. The First- and Second-order Theory of One Successor
   3. Automata over finite and infinite words
   4. Linear Temporal Logic
2. The safety fragment of LTL and its theoretical features
   1. Definition of Safety and Cosafety
   2. Characterizations and Normal Forms
   3. Kupferman and Vardi's Classification

Outline

REACTIVE SYNTHESIS

from safety fragments of LTL

Reactive Synthesis
Definition and Classic Approach



1. What are realizability and reactive synthesis?
   - model-based design: all the effort on the quality of the specification
   - culmination of declarative programming
2. Complexity:
   - for S1S: non-elementary
   - for LTL: 2EXPTIME-complete.

Definition (Strategy)

Let $\Sigma = \mathcal{C} \cup \mathcal{U}$ be an alphabet partitioned into the set of controllable variables $\mathcal{C}$ and the set of uncontrollable ones $\mathcal{U}$, such that $\mathcal{C} \cap \mathcal{U} = \varnothing$. A *strategy for Controller* is a function

$$g : (2^{\mathcal{U}})^+ \rightarrow 2^{\mathcal{C}}$$

that, given the sequence $\mathsf{U} = \langle \mathsf{U}_0, \ldots, \mathsf{U}_n \rangle$ of choices made by *Environment* so far, determines the current choices $\mathsf{C}_n = g(\mathsf{U})$ of *Controller*.

Definition (Strategy)

Let $\Sigma = \mathcal{C} \cup \mathcal{U}$ be an alphabet partitioned into the set of controllable variables $\mathcal{C}$ and the set of uncontrollable ones $\mathcal{U}$, such that $\mathcal{C} \cap \mathcal{U} = \varnothing$. A *strategy for Controller* is a function

$$g : (2^{\mathcal{U}})^{+} \to 2^{\mathcal{C}}$$

that, given the sequence $\mathsf{U} = \langle \mathsf{U}_0, \ldots, \mathsf{U}_n \rangle$ of choices made by *Environment* so far, determines the current choices $\mathsf{C}_n = g(\mathsf{U})$ of *Controller*.

Definition (Realizability and Synthesis)

Let ϕ be a temporal formula over the alphabet $\Sigma = \mathcal{C} \cup \mathcal{U}$. We say that ϕ is *realizable* if and only if

- $\exists g : (2^{\mathcal{U}})^{+} \to 2^{\mathcal{C}}$
- $\forall \omega$-sequence $\mathsf{U} = \langle \mathsf{U}_0, \mathsf{U}_1, \ldots \rangle \in (2^{\mathcal{U}})^{\omega}$
- $\langle \mathsf{U}_0 \cup g(\langle \mathsf{U}_0 \rangle), \mathsf{U}_1 \cup g(\langle \mathsf{U}_0, \mathsf{U}_1 \rangle), \ldots \rangle \models \phi$

In this case, g is called *winning strategy*. If ϕ is realizable, the synthesis problem is the problem of computing such a strategy g.

Definition (Finitely representable strategies)

Let $g : (2^{\mathcal{U}})^+ \to 2^{\mathcal{C}}$ be a strategy. We say that g is *finitely representable* iff there exists a Mealy machine M_g *"equivalent"* to g.

Proposition (Small model property of LTL)

Let ϕ be an LTL *formula and $n = |\phi|$. If ϕ is realizable, then there exists a finitely representable winning strategy g such that its corresponding Mealy machine has at most $2^{2^{c \cdot n}}$ states, for some constant c.*

Reference:

Amir Pnueli and Roni Rosner (1989). "On the Synthesis of a Reactive Module". In: *Proceedings of POPL'89*. ACM Press, pp. 179–190. DOI: 10.1145/75277.75293

- Realizability is modeled as a *two-players game* over an *arena/automaton* $\mathcal{A}_\phi$ built from ϕ:
  - Controller player: his objective is to enforce the satisfaction of the specification, no matter the choices of the other player (winning strategy)
  - Environment player: his objective is to enforce the violation of the specification, no matter the choices of the other player
- Environment player moves first.
- The game is played on *deterministic* automata obtained from the initial specification.
  - there are simple algorithms for synthesis over deterministic arenas

$$\Rightarrow \text{ backward fixpoint computations}$$

  - LTL formula $\phi \rightsquigarrow$ DRA $\mathcal{A}_\phi$

We consider first the case of *finite words*.

Standard Approach:

$$\mathsf{LTL_f}\ \phi \longrightarrow \mathsf{NFA}\ \mathcal{A}_\phi \xrightarrow{\text{determinization}} \mathsf{DFA}\ \mathcal{A}'_\phi \longrightarrow \text{reachability game}$$

- The DFA $\mathcal{A}'_\phi$ is *equivalent* to ϕ:

$$\mathcal{L}(\mathcal{A}'_\phi) = \mathcal{L}(\phi)$$

- Controller can force to the game to reach a *final* state of $\mathcal{A}'_\phi$ iff there is a winning strategy for the formula ϕ:
  - playing over the DFA $\mathcal{A}'_\phi$ is equivalent to solve the reactive synthesis problem for ϕ.

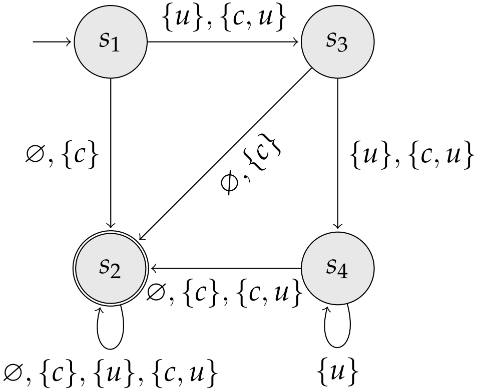
Definition (Strong Predecessor)

Let $\mathcal{A} = \langle Q, 2^{\mathcal{U}} \cup 2^{\mathcal{C}}, q_0, \delta, F\rangle$ be a DFA and let $S \subseteq Q$. We define the *strong precedessors* of S as follows:

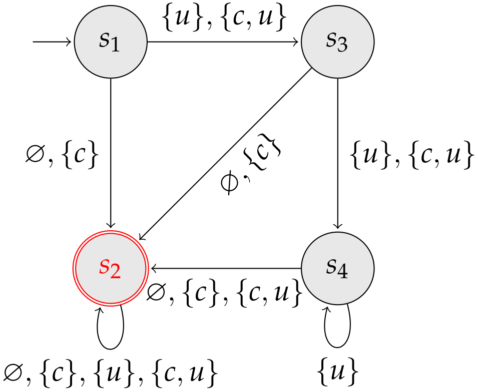
$$\mathtt{pre}(S) := \{s \in Q \mid \forall u \in 2^{\mathcal{U}} \,.\, \exists c \in 2^{\mathcal{C}} \,.\, s \xrightarrow{u,c} s', \text{ for some } s' \in S\}$$

$\mathtt{pre}(S)$ is the set of states of $\mathcal{A}$ from which Controller can force the game into a state of S in one step.

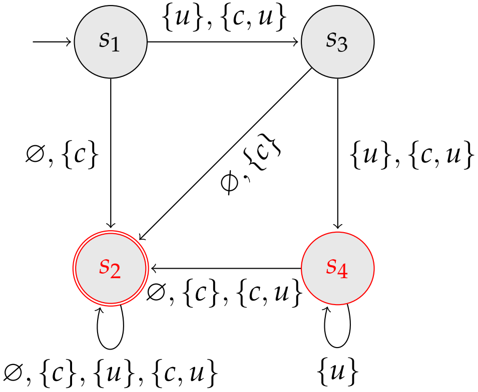
- The *winning region* is the set of states from which Controller can force the game to reach a final state.
  - $\Rightarrow$ reachability games
- Computation of the winning region (greatest fixed point):
  - $W_0 := F$
  - $W_{i+1} := W_i \cup \mathtt{pre}(W_i)$
- We stop when $W_i = W_{i+1}$ (fixed point).
- Controller wins iff $q_0 \in W_i$. The initial specification is *realizable*.
- Otherwise, Environment has a strategy for violating the specification.



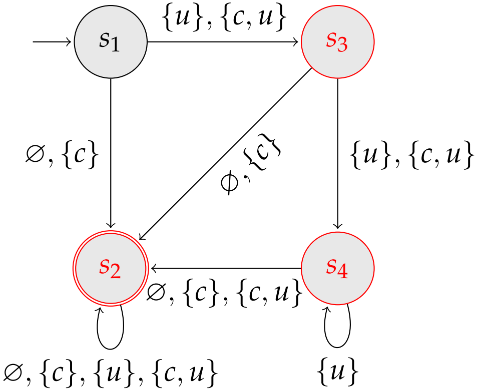
- DFA for the formula $\mathsf{F}(u \rightarrow \mathsf{XX}c)$, with $u \in \mathcal{U}$ and $c \in \mathcal{C}$.



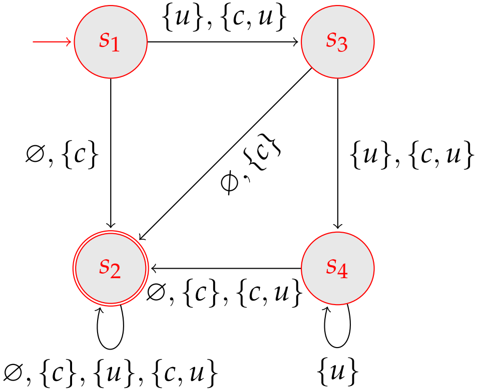
- DFA for the formula $\mathsf{F}(u \to \mathsf{XX}c)$, with $u \in \mathcal{U}$ and $c \in \mathcal{C}$.
- $W_0 := \{s_2\}$



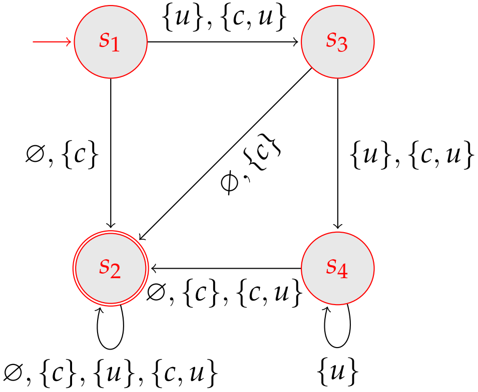
- DFA for the formula $\mathsf{F}(u \rightarrow \mathsf{XX}c)$, with $u \in \mathcal{U}$ and $c \in \mathcal{C}$.
- $W_0 := \{s_2\}$
- $W_1 := \{s_2, s_4\}$



- DFA for the formula $\mathsf{F}(u \rightarrow \mathsf{XX}c)$, with $u \in \mathcal{U}$ and $c \in \mathcal{C}$.
- $W_0 := \{s_2\}$
- $W_1 := \{s_2, s_4\}$
- $W_2 := \{s_2, s_4, s_3\}$



- DFA for the formula $\mathsf{F}(u \rightarrow \mathsf{XX}c)$, with $u \in \mathcal{U}$ and $c \in \mathcal{C}$.
- $W_0 := \{s_2\}$
- $W_1 := \{s_2, s_4\}$
- $W_2 := \{s_2, s_4, s_3\}$
- $W_3 := \{s_2, s_4, s_3, s_1\}$



- DFA for the formula $\mathsf{F}(u \rightarrow \mathsf{XX}c)$, with $u \in \mathcal{U}$ and $c \in \mathcal{C}$.
- $W_0 := \{s_2\}$
- $W_1 := \{s_2, s_4\}$
- $W_2 := \{s_2, s_4, s_3\}$
- $W_3 := \{s_2, s_4, s_3, s_1\}$
- $W_3 \cap I \neq \varnothing \Rightarrow$ the formula is realizable.

The case of Infinite Words

Standard approach:

$$\text{LTL } \phi \;\longrightarrow\; \text{NBA } \mathcal{A}(\phi) \;\xrightarrow{\text{determinization}}\; \text{DRA } \mathcal{A}(\phi) \;\longrightarrow\; \text{game solver}$$

The case for *infinite words* (like in the case for LTL) is much more difficult. Two reasons:

- Büchi games
- NBA cannot be determinized easily. Indeed, *Safra's construction* is:
  - very complicated
  - difficult to implement
  - not amenable to optimizations

The case of Infinite Words

Standard approach:

LTL ϕ

$\downarrow$

NBA $\mathcal{A}(\phi)$

$\downarrow$ · determinization

DRA $\mathcal{A}(\phi)$

$\downarrow$

game solver

Research mainly focused on two lines

1. finding good algorithms for the average case
   - Safraless approaches
     - Bounded synthesis
2. restricting the expressiveness of the specification language
   - GR(1)
   - SafetyLTL

$$\mathsf{SafetyLTL}\ \phi \xrightarrow{\text{bad prefixes}} \mathsf{DFA}\ \mathcal{A}_{bad} \longrightarrow \text{reachability game}$$

Game:

- Now, Controller moves first
- Goal of Controller: always avoid final states of $\mathcal{A}_{bad}$.
- Goal of Environment: reach a final state of $\mathcal{A}_{bad}$.

Reactive Synthesis of Safety Properties
SafetyLTL

SafetyLTL ϕ

$\downarrow$ · bad prefixes

DFA $\mathcal{A}_{bad}$

$\downarrow$

reachability game

Pros:

- infinite words $\rightsquigarrow$ finite word
- Safra's algorithm is *avoided*.
- We use standard subset construction for $\mathcal{A}_{bad}$:
  - easily implementable
  - easily optimizable

Cons:

- the size of $\mathcal{A}_{bad}$ is $2^{2^{\Theta(n)}}$.
- this is prohibitive when ϕ is large.

Tool: SSyft

SafetyLTL ϕ

$\downarrow$ bad prefixes

DFA $\mathcal{A}_{bad}$

$\downarrow$

reachability game

Reference:

Shufang Zhu et al. (2017). "A Symbolic Approach to Safety LTL Synthesis". In: *Proceedings of the 13th International Haifa Verification Conference*. Ed. by Ofer Strichman and Rachel Tzoref-Brill. Vol. 10629. Lecture Notes in Computer Science. Springer, pp. 147–162. DOI: 10.1007/978-3-319-70389-3_10

Link: https://github.com/Shufang-Zhu/Syft-safety

Reactive Synthesis of Safety Properties
SafetyLTL

$$
\begin{array}{c}
\text{SafetyLTL } \phi \\
\Big\downarrow \textcolor{orange}{\text{bad prefixes}} \\
\text{DFA } \mathcal{A}_{bad} \\
\Big\downarrow \\
\text{reachability game}
\end{array}
$$

Tool: SSyft

1. Let ϕ be a SafetyLTL formula.
2. Translate $\neg\phi$ into an equivalent formula ψ of S1S[FO] interpreted over finite words.
   - the models of ψ are exactly the *bad prefixes* of ϕ
3. Call the tool MONA for building the equivalent and *minimal* DFA.
4. Solve a reachability game.

Reactive Synthesis of Safety Properties

SafetyLTL

SafetyLTL ϕ

$\downarrow$ bad prefixes

DFA $\mathcal{A}_{bad}$

$\downarrow$

reachability game

- MONA is a very efficient tool for the construction of automata starting from formulas.
- MONA implements decision procedures for the Weak Second-order Theory of One or Two successors.
- Link : `https://www.brics.dk/mona/`

$$
\begin{array}{c}
\text{SafetyLTL } \phi \\
\Big\downarrow \cdot \text{ bad prefixes} \\
\text{DFA } \mathcal{A}_{bad} \\
\Big\downarrow \\
\text{reachability game}
\end{array}
$$

Theorem

SafetyLTL *realizability is* 2EXPTIME*-complete.*

Reference:

Alessandro Artale et al. (2023). "Complexity of Safety and coSafety Fragments of Linear Temporal Logic". In: *Proc. of the 36th AAAI Conf. on Artificial Intelligence*. AAAI Press

Logics	Problems		
	satisfiability	model checking	realizability
coSafetyLTL	PSPACE-c	???	2EXPTIME-c
F(pLTL)	PSPACE-c	???	EXPTIME-c
LTL[X, F]	NP-c	???	EXPTIME-c

Logics	Problems		
	satisfiability	model checking	realizability
SafetyLTL	PSPACE-c	???	2EXPTIME-c
G(pLTL)	PSPACE-c	???	EXPTIME-c
LTL[$\widetilde{\mathsf{X}}$, G]	PSPACE-c	???	EXPTIME-c

Logics	Problems		
	satisfiability	model checking	realizability
coSafetyLTL	PSPACE-c	???	2EXPTIME-c
F(pLTL)	PSPACE-c	???	EXPTIME-c
LTL[X, F]	NP-c	???	EXPTIME-c

Logics	Problems		
	satisfiability	model checking	realizability
SafetyLTL	PSPACE-c	???	2EXPTIME-c
G(pLTL)	PSPACE-c	???	EXPTIME-c
LTL[$\widetilde{X}$, G]	PSPACE-c	???	EXPTIME-c

G(pLTL) ϕ

$\downarrow$ · deterministic automaton · of singly exponential size

DFA $\mathcal{A}$

$\downarrow$

safety game

Algorithm:

1. Let $\mathsf{G}(\alpha)$ be a formula of G(pLTL).

Theorem

ϕ is realizable (with Environment moving first) iff $\neg\phi$ is unrealizable (with Controller moving first).

$\mathsf{G}(\alpha)$ is realizable iff $\mathsf{F}(\neg\alpha)$ is unrealizable (with Controller moving first).

$\mathsf{G}(\mathsf{pLTL})\ \phi$

$\downarrow$ · deterministic automaton · of singly exponential size

DFA $\mathcal{A}$

$\downarrow$

safety game

Algorithm:

1. Let $\mathsf{G}(\alpha)$ be a formula of $\mathsf{G}(\mathsf{pLTL})$.
   $\mathsf{G}(\alpha)$ is realizable iff $\mathsf{F}(\neg\alpha)$ is unrealizable
2. Build the DFA $\mathcal{A}_{\neg\alpha}$ for $\neg\alpha \in \mathsf{pLTL}$
   - this can be done in $2^{\mathcal{O}(n)}$
   - we will see later its construction
3. Solve a reachability game on $\mathcal{A}_{\neg\alpha}$:
   - if Controller (that moves first) wins:
     - $\mathsf{F}(\neg\alpha)$ is realizable
     - $\mathsf{G}(\alpha)$ is unrealizable
   - if Environment wins:
     - $\mathsf{F}(\neg\alpha)$ is unrealizable
     - $\mathsf{G}(\alpha)$ is realizable

Reactive Synthesis of Safety Properties

G(pLTL)

$$\begin{tikzcd} \mathsf{G(pLTL)}\ \phi \arrow[d, "\substack{\cdot\ \text{deterministic automaton}\\ \cdot\ \text{of singly exponential size}}"] \\ \mathsf{DFA}\ \mathcal{A} \arrow[d] \\ \text{safety game} \end{tikzcd}$$

- Advantages:
  - The size of $|\mathcal{A}|$ is $2^{\mathcal{O}(n)}$:
    - singly exponential
    - one exponential smaller than the set of bad prefixes of a SafetyLTL formula.
  - The translation from pLTL into DFA can be done in a purely symbolic fashion

Reference:

Alessandro Cimatti et al. (2021). "Extended bounded response LTL: a new safety fragment for efficient reactive synthesis". In: *Formal Methods in System Design*, 1–49 (published online on November 18, 2021, doi: 10.1007/s10703-021-00383–3)

Theorem

For any formula ϕ of pLTL *with $n = |\phi|$, there exists a* DFA *$\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\phi)$ and $|\mathcal{A}| \in 2^{\mathcal{O}(n)}$.*

Reference:

Giuseppe De Giacomo et al. (2021). "Pure-past linear temporal and dynamic logic on finite traces". In: *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pp. 4959–4965

Theorem

For any formula ϕ of pLTL *with $n = |\phi|$, there exists a* DFA *$\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\phi)$ and $|\mathcal{A}| \in 2^{\mathcal{O}(n)}$.*

Intuition:

Since past already happened, there is no need for nondeterminism.

There is this useful asymmetry:

- *The automaton reads from left to right;*
- *The pure past formula predicates from right to left.*

Theorem

For any formula ϕ of pLTL *with $n = |\phi|$, there exists a* DFA *$\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\phi)$ and $|\mathcal{A}| \in 2^{\mathcal{O}(n)}$.*

De Giacomo *et al*. prove the result passing from alternating automata.

Theorem

For any alternating finite automaton $\mathcal{A}$, there exists a DFA *for its reverse language of size singly exponential in $|\mathcal{A}|$.*

Reference:

Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer (1981). "Alternation". In: *J. ACM* 28.1, pp. 114–133. DOI: 10.1145/322234.322243. URL: https://doi.org/10.1145/322234.322243

Theorem

For any formula ϕ of pLTL *with $n = |\phi|$, there exists a* DFA *$\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\phi)$ and $|\,\mathcal{A}\,| \in 2^{\mathcal{O}(n)}$.*

Here we give a direct construction.

Definition (Closure of pLTL formulas)

The *closure* of a pLTL formula ϕ over the atomic propositions $\mathcal{AP}$, denoted as $\mathcal{C}(\phi)$, is the smallest set of formulas satisfying the following properties:

- $\mathsf{Y}\phi \in \mathcal{C}(\phi)$
- $\phi \in \mathcal{C}(\phi)$, and, for each subformula ϕ' of ϕ, $\phi' \in \mathcal{C}(\phi)$
- for each $p \in \mathcal{AP}$, $p \in \mathcal{C}(\phi)$ if and only if $\neg p \in \mathcal{C}(\phi)$
- if $\phi_1 \mathsf{S} \phi_2 \in \mathcal{C}(\phi)$, then $\mathsf{Y}(\phi_1 \mathsf{S} \phi_2) \in \mathcal{C}(\phi)$
    - if $\mathsf{O}\phi_1 \in \mathcal{C}(\phi)$, then $\mathsf{Y}(\mathsf{O}\phi_1) \in \mathcal{C}(\phi)$
- if $\phi_1 \mathsf{T} \phi_2 \in \mathcal{C}(\phi)$, then $\widetilde{\mathsf{Y}}(\phi_1 \mathsf{T} \phi_2) \in \mathcal{C}(\phi)$
    - if $\mathsf{H}\phi_1 \in \mathcal{C}(\phi)$, then $\widetilde{\mathsf{Y}}(\mathsf{H}\phi_1) \in \mathcal{C}(\phi)$

- We denote by $\mathcal{C}_{\mathsf{Y}}(\phi)$ the set of formulas of type $\mathsf{Y}\phi_1$ in $\mathcal{C}(\phi)$.
- We denote by $\mathcal{C}_{\widetilde{\mathsf{Y}}}(\phi)$ the set of formulas of type $\widetilde{\mathsf{Y}}\phi_1$ in $\mathcal{C}(\phi)$.

Definition (Stepped Normal Form)

Let ϕ be a pLTL formula over the atomic propositions $\mathcal{AP}$. Its *stepped normal form*, denoted by $\mathrm{snf}(\phi)$, is defined as follows:

$$\begin{aligned}
\mathrm{snf}(\ell) &= \ell && \text{where } \ell \in \{p, \neg p\}, \text{ for } p \in \mathcal{AP} \\
\mathrm{snf}(\otimes\, \phi_1) &= \otimes\, \phi_1 && \text{where } \otimes \in \{\mathsf{Y}, \widetilde{\mathsf{Y}}\} \\
\mathrm{snf}(\phi_1 \otimes \phi_2) &= \mathrm{snf}(\phi_1) \otimes \mathrm{snf}(\phi_2) && \text{where } \otimes \in \{\wedge, \vee\} \\
\mathrm{snf}(\phi_1\, \mathsf{S}\, \phi_2) &= \mathrm{snf}(\phi_2) \vee (\mathrm{snf}(\phi_1) \wedge \mathsf{Y}(\phi_1\, \mathsf{S}\, \phi_2)) \\
\mathrm{snf}(\phi_1\, \mathsf{T}\, \phi_2) &= \mathrm{snf}(\phi_2) \wedge (\mathrm{snf}(\phi_1) \vee \widetilde{\mathsf{Y}}(\phi_1\, \mathsf{T}\, \phi_2))
\end{aligned}$$

Example: $\mathrm{snf}(\mathsf{O}q) = q \vee \mathsf{YO}q$.

Given a set $S \subseteq \mathcal{C}_{\mathsf{Y}}(\phi) \cup \mathcal{C}_{\widetilde{\mathsf{Y}}}(\phi)$ and a $\sigma \in 2^{\mathcal{AP}}$, we write $S, \sigma \models \phi$ iff ϕ is true when:

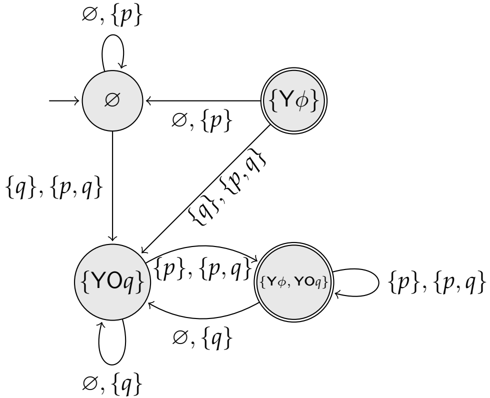
- S is used for interpreting the subformulas of type $\mathsf{Y}\alpha$ and $\widetilde{\mathsf{Y}}\alpha$
- σ is used for interpreting proposition letters in $\mathcal{AP}$

Example:

- $S = \{\mathsf{YO}q\}$
- $\sigma = \varnothing$
- $S, \sigma \models q \vee \mathsf{YO}q$

Given $\phi \in$ LTL we define the DFA $\mathcal{A}_\phi = \langle Q, \Sigma, q_0, \delta, F \rangle$ as follows:

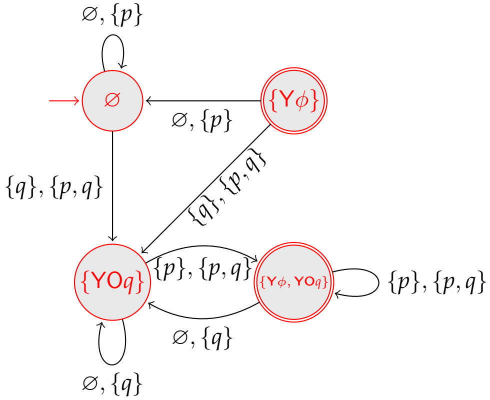
Example: $\phi := p \land \mathsf{YO}q$



Given $\phi \in$ LTL we define the DFA $\mathcal{A}_\phi = \langle Q, \Sigma, q_0, \delta, F \rangle$ as follows:

- $Q = 2^{\mathcal{C}_{\mathsf{Y}}(\phi) \cup \mathcal{C}_{\widetilde{\mathsf{Y}}}(\phi)}$
  - $Q = \{\varnothing, \{\mathsf{Y}\phi\}, \{\mathsf{YO}q\}, \{\mathsf{Y}\phi, \mathsf{YO}q\}\}$

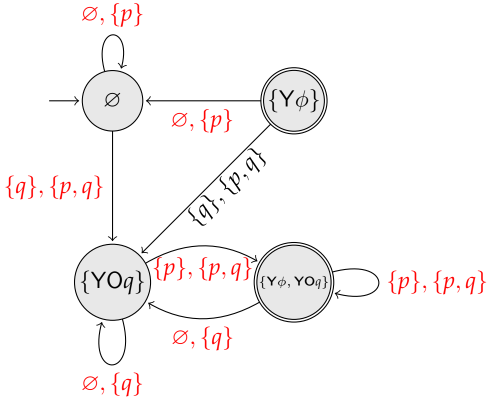
Example: $\phi := p \wedge \mathsf{YO}q$



Given $\phi \in$ LTL we define the DFA $\mathcal{A}_\phi = \langle Q, \Sigma, q_0, \delta, F \rangle$ as follows:

- $Q = 2^{\mathcal{C}_{\mathsf{Y}}(\phi) \cup \mathcal{C}_{\widetilde{\mathsf{Y}}}(\phi)}$
    - $Q = \{\varnothing, \{\mathsf{Y}\phi\}, \{\mathsf{YO}q\}, \{\mathsf{Y}\phi, \mathsf{YO}q\}\}$
- $\Sigma = 2^{\mathcal{AP}}$
    - $\Sigma = \{\varnothing, \{p\}, \{q\}, \{p, q\}\}$

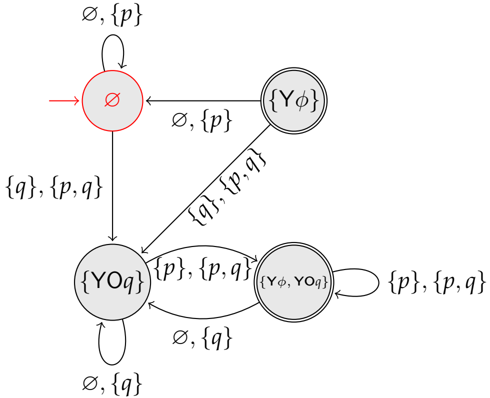
Example: $\phi := p \wedge \mathsf{YO}q$



Given $\phi \in$ LTL we define the DFA $\mathcal{A}_\phi = \langle Q, \Sigma, q_0, \delta, F\rangle$ as follows:

- $Q = 2^{\mathcal{C}_{\mathsf{Y}}(\phi) \cup \mathcal{C}_{\widetilde{\mathsf{Y}}}(\phi)}$
  - $Q = \{\varnothing, \{\mathsf{Y}\phi\}, \{\mathsf{YO}q\}, \{\mathsf{Y}\phi, \mathsf{YO}q\}\}$
- $\Sigma = 2^{\mathcal{AP}}$
  - $\Sigma = \{\varnothing, \{p\}, \{q\}, \{p, q\}\}$
- $q_0 = \mathcal{C}_{\widetilde{\mathsf{Y}}}(\phi)$
  - $q_0 = \varnothing$

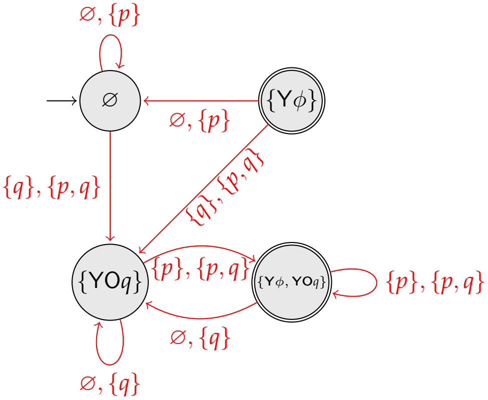
Example: $\phi := p \wedge \mathsf{YO}q$



Given $\phi \in$ LTL we define the DFA $\mathcal{A}_\phi = \langle Q, \Sigma, q_0, \delta, F \rangle$ as follows:

- $Q = 2^{\mathcal{C}_{\mathsf{Y}}(\phi) \cup \mathcal{C}_{\widetilde{\mathsf{Y}}}(\phi)}$
  - $Q = \{\varnothing, \{\mathsf{Y}\phi\}, \{\mathsf{YO}q\}, \{\mathsf{Y}\phi, \mathsf{YO}q\}\}$
- $\Sigma = 2^{\mathcal{AP}}$
  - $\Sigma = \{\varnothing, \{p\}, \{q\}, \{p, q\}\}$
- $q_0 = \mathcal{C}_{\widetilde{\mathsf{Y}}}(\phi)$
  - $q_0 = \varnothing$
- $\delta(q, \sigma) = \{\mathsf{Y}\psi, \widetilde{\mathsf{Y}}\psi \in \mathcal{C}_{\mathsf{Y}}(\phi) \cup \mathcal{C}_{\widetilde{\mathsf{Y}}}(\phi) \mid q, \sigma \models \mathrm{snf}(\psi)\}$
  - see figure

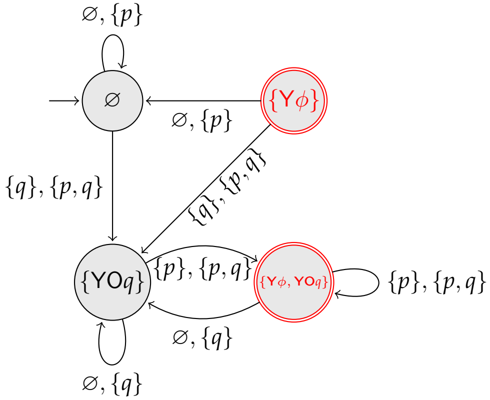
Example: $\phi \coloneqq p \wedge \mathsf{YO}q$



Given $\phi \in$ LTL we define the DFA $\mathcal{A}_\phi = \langle Q, \Sigma, q_0, \delta, F \rangle$ as follows:

- $Q = 2^{\mathcal{C}_{\mathsf{Y}}(\phi) \cup \mathcal{C}_{\widetilde{\mathsf{Y}}}(\phi)}$
  - $Q = \{\varnothing, \{\mathsf{Y}\phi\}, \{\mathsf{YO}q\}, \{\mathsf{Y}\phi, \mathsf{YO}q\}\}$
- $\Sigma = 2^{\mathcal{AP}}$
  - $\Sigma = \{\varnothing, \{p\}, \{q\}, \{p, q\}\}$
- $q_0 = \mathcal{C}_{\widetilde{\mathsf{Y}}}(\phi)$
  - $q_0 = \varnothing$
- $\delta(q, \sigma) = \{\mathsf{Y}\psi, \widetilde{\mathsf{Y}}\psi \in \mathcal{C}_{\mathsf{Y}}(\phi) \cup \mathcal{C}_{\widetilde{\mathsf{Y}}}(\phi) \mid q, \sigma \models \mathrm{snf}(\psi)\}$
  - see figure
- $F = \{S \subseteq \mathcal{C}_{\mathsf{Y}}(\phi) \cup \mathcal{C}_{\widetilde{\mathsf{Y}}}(\phi) \mid \mathsf{Y}\phi \in S\}$
  - $F = \{\{\mathsf{Y}\phi\}, \{\mathsf{Y}\phi, \mathsf{YO}q\}\}$

Example: $\phi := p \wedge \mathsf{YO}q$



Theorem

$\mathsf{G}(\mathsf{pLTL})$ *realizability is* EXPTIME*-complete.*

Theorem

$\mathsf{F}(\mathsf{pLTL})$ *realizability is* EXPTIME*-complete.*

Reference:

Alessandro Artale et al. (2023). "Complexity of Safety and coSafety Fragments of Linear Temporal Logic". In: *Proc. of the 36th AAAI Conf. on Artificial Intelligence*. AAAI Press

Logics	Problems		
	satisfiability	model checking	realizability
coSafetyLTL	PSPACE-c	???	2EXPTIME-c
F(pLTL)	PSPACE-c	???	EXPTIME-c
LTL[X, F]	NP-c	???	EXPTIME-c

Logics	Problems		
	satisfiability	model checking	realizability
SafetyLTL	PSPACE-c	???	2EXPTIME-c
G(pLTL)	PSPACE-c	???	EXPTIME-c
LTL[$\widetilde{\mathsf{X}}$, G]	PSPACE-c	???	EXPTIME-c

Theorem

$\mathsf{G}(\mathsf{pLTL})$ *realizability is* EXPTIME*-complete.*

- Pure past LTL plays a crucial role for safety fragments
- SafetyLTL realizability is 2EXPTIME-complete
- …but $\mathsf{G}(\mathsf{pLTL})$ and SafetyLTL are expressively equivalent

Theorem

G(pLTL) *realizability is* EXPTIME*-complete.*

- Pure past LTL plays a crucial role for safety fragments
- SafetyLTL realizability is 2EXPTIME-complete
- …but G(pLTL) and SafetyLTL are expressively equivalent

Two questions:

1. Succinctness

   Can SafetyLTL *be exponentially more succinct than* G(pLTL)*?*

2. Pastification algorithms

   Can a logic be efficiently translated into a pure-past one, by preserving equivalence?

REFERENCES

Bibliography I

Alessandro Artale et al. (2023). "Complexity of Safety and coSafety Fragments of Linear Temporal Logic". In: *Proc. of the 36th AAAI Conf. on Artificial Intelligence*. AAAI Press.

Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer (1981). "Alternation". In: *J. ACM* 28.1, pp. 114–133. DOI: 10.1145/322234.322243. URL: https://doi.org/10.1145/322234.322243.

Alessandro Cimatti et al. (2021). "Extended bounded response LTL: a new safety fragment for efficient reactive synthesis". In: *Formal Methods in System Design*, 1–49 (published online on November 18, 2021, doi: 10.1007/s10703-021-00383–3).

Giuseppe De Giacomo et al. (2021). "Pure-past linear temporal and dynamic logic on finite traces". In: *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pp. 4959–4965.

Amir Pnueli and Roni Rosner (1989). "On the Synthesis of a Reactive Module". In: *Proceedings of POPL'89*. ACM Press, pp. 179–190. DOI: 10.1145/75277.75293.

Shufang Zhu et al. (2017). "A Symbolic Approach to Safety LTL Synthesis". In: *Proceedings of the 13th International Haifa Verification Conference*. Ed. by Ofer Strichman and Rachel Tzoref-Brill. Vol. 10629. Lecture Notes in Computer Science. Springer, pp. 147–162. DOI: 10.1007/978-3-319-70389-3_10.