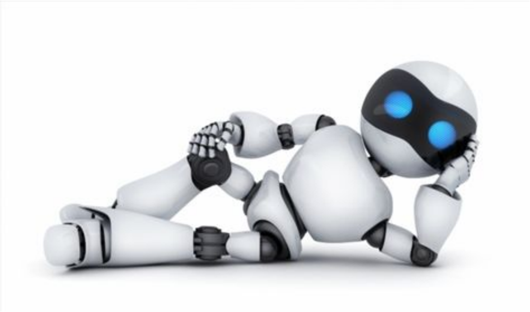
Game-Theoretic Approach to Temporal Synthesis

Notable Cases of LTL_f Synthesis under LTL Specifications

Antonio Di Stasio   Giuseppe Perelli   **Shufang Zhu**

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



- Two-player games, playing adversarially
  - Reachability, Safety, Büchi, Parity
- Temporal logic specifications
  - LTL, LTL_f, LDL_f

- Automata and temporal logics to automata
  - LTL_f and DFA, LTL and Büchi automata, parity automata
- Synthesis, automata-theoretic approaches to synthesis
  - LTL_f synthesis as reachability game on a DFA

- Markovian domain: planning domain, actions only depend on the current state
- Non-Markovian domain: actions also depend on history, safety properties
- Considering also liveness: Linear Temporal Logic (LTL)
- LTL_f synthesis under LTL environment specifications
  - Agent goal LTL_f φ, Environment specification LTL $\mathcal{E}$
  - Reduction to implication $\mathcal{E} \rightarrow \varphi$
  - Two-stage technique, parity game

Notable cases of LTL_f synthesis under environment specifications

- Diminish the difficulty of constructing the arena from the environment specification
- No reduction to parity game
- Possibly no reduction to implication

- LTL_f synthesis under Environment **safety** specifications

- LTL_f synthesis under Environment **safety** specifications
- Enrich environment specification, **safety** and

- LTL_f synthesis under Environment **safety** specifications
- Enrich environment specification, **safety** and
  - Simple Fairness [1]

[1]Zhu et al.: LTL_f Synthesis with Fairness and Stability Assumptions.

- LTL_f synthesis under Environment **safety** specifications
- Enrich environment specification, **safety** and
    - Simple Stability [1]

[1] Zhu et al.: LTL_f Synthesis with Fairness and Stability Assumptions.

- LTL_f synthesis under Environment **safety** specifications
- Enrich environment specification, **safety** and
  - Generalized Reactivity (1) [1]

[1] De Giacomo et al.: Finite-Trace and Generalized-Reactivity Specifications in Temporal Synthesis.

Outline

- LTL_f synthesis under Environment **safety** specifications
- Enrich environment specification, **safety** and
  - Simple Fairness
  - Simple Stability
  - Generalized Reactivity (1)
- LTL_f synthesis under Environment **safety** specifications, without reduction to implication

LTL_f synthesis under Environment **safety** specifications

Environment Safety Specifications

Environment stays in expected boundaries

Environment stays in expected boundaries

- Initial state of the environment
  - The block is on the table, and the robot's grasper is empty

Environment stays in expected boundaries

- Initial state of the environment
  - The block is on the table, and the robot's grasper is empty
- How the environment changes with respect to agent actions
  - Whenever the robot **picks up** the block, the block is in the robot's grasper

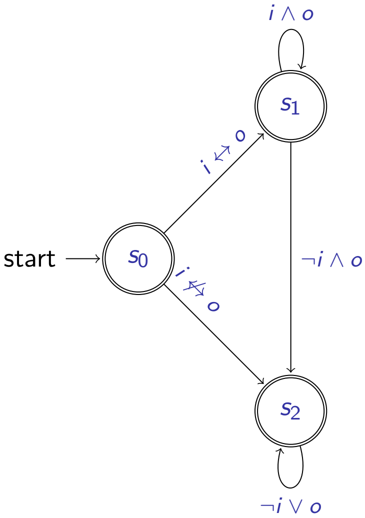
– PDDL, planning domain

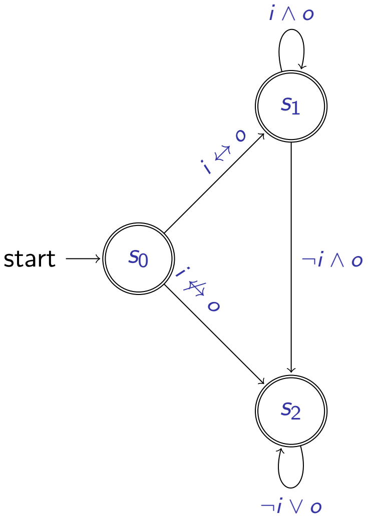
```
(:action pick-up
        :parameters (?block)
        :precondition (and (ontable ?block) (handempty))
        :effect
        (and (not (ontable ?block))
          (not (handempty))
          (holding ?block)))
```

- Logic specification language
  - Safety fragment of LTL, disallowing $\mathcal{U}$ operator
    - $G(\bigcirc block\_in\_grasper \wedge \bigcirc \neg block\_on\_table \leftrightarrow pick\_up)$

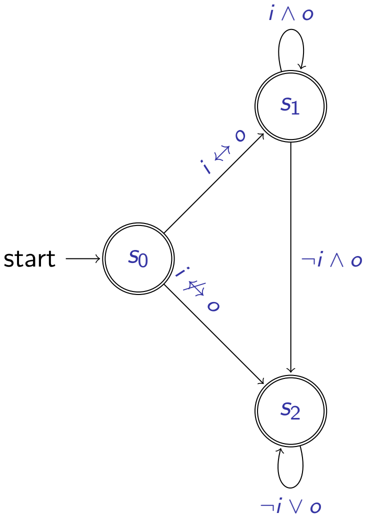
– Deterministic Safety Automata (DSA)

- Deterministic Safety Automata (DSA), tuple $\mathcal{D} = (2^{\mathcal{P}}, \mathcal{S}, s_0, \delta)$, where
  - $2^{\mathcal{P}}$ is the alphabet
  - $\mathcal{S}$ is a finite set of states
  - s_0 is the initial state
  - $\delta : \mathcal{S} \times 2^{\mathcal{P}} \mapsto \mathcal{S}$ is the transition function

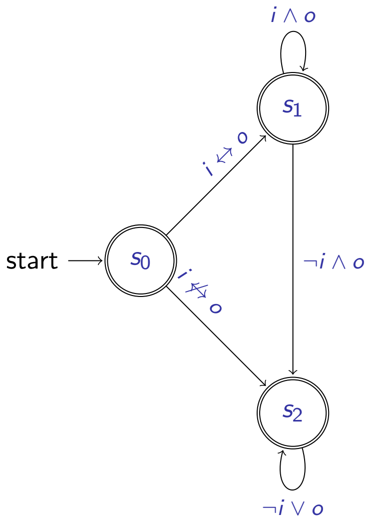

$i \wedge o$
s_1
$i \leftrightarrow o$
start
s_0
$\neg i \wedge o$
$i \not\leftrightarrow o$
s_2
$\neg i \vee o$



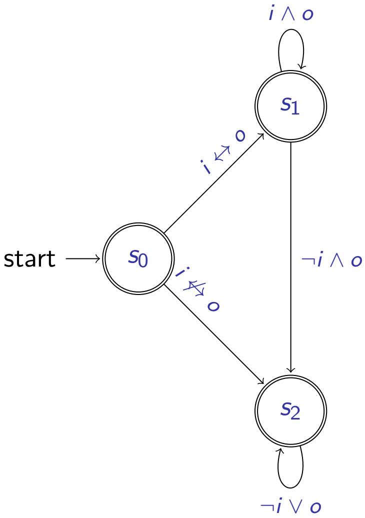
- $2^{\mathcal{P}} = 2^{\{i,o\}}$



- $2^{\mathcal{P}} = 2^{\{i,o\}}$
- $\mathcal{S} = \{s_0, s_1, s_2\}$



- $2^{\mathcal{P}} = 2^{\{i,o\}}$
- $\mathcal{S} = \{s_0, s_1, s_2\}$
- $\delta : \mathcal{S} \times 2^{\mathcal{P}} \mapsto \mathcal{S}$



$\pi \in \mathcal{L}(\mathcal{D})$ if the run r of $\mathcal{D}$ on π is infinite

Example: $\pi = (i \wedge o)^{\omega}$, $r = s_0(s_1)^{\omega}$

- Logic specification language
  - Safety fragment of LTL, disallowing $\mathcal{U}$ operator
    - Restriction on writing formulas
  - LTL_f formulas, no restrictions on syntax, alternative semantics interpretation

Safety specification:

- interpreted on infinite traces
- once violated, **exists** a finite prefix that breaks the specification

Safety specification:

- interpreted on infinite traces
- once violated, **exists** a finite prefix that breaks the specification

Safety specification (alternative view):

- interpreted on infinite traces
- once hold, **all** finite prefixes are good

Safety specification:

– specifications that hold for all finite prefixes of an infinite trace

Safety specification:

- specifications that hold for all finite prefixes of an infinite trace
- utilize LTL_f to specify expected property on finite traces

LTL_f to specify safety specifications

- Safety specifications: all finite prefixes of an infinite trace are "good"
- "good": specified in LTL_f
- Alternative notion of satisfaction that interprets an LTL_f formula over all finite prefixes of an infinite trace

Definition

An infinite trace π satisfies an LTL_f formula φ on *all prefixes*, denoted $\pi \models_\forall \varphi$, if every non-empty finite prefix of π satisfies φ. That is, $\pi^k = \pi_1, \ldots, \pi_k \models \varphi$, for every $1 \leq k \leq |\pi|$. [1]

[1] De Giacomo et al.: Finite-Trace and Generalized-Reactivity Specifications in Temporal Synthesis.

Theorem

Every first-order safety specification can be expressed as an LTL_f formula on all prefixes. [1]

[1] De Giacomo et al.: Finite-Trace and Generalized-Reactivity Specifications in Temporal Synthesis.

Reasoning on LTL_f specifying safety specifications as reasoning on automata

- **Given:** LTL_f formula φ specifying safety specifications
- **Obtain:** The corresponding DSA $\mathcal{D}$ such that

$$\mathcal{L}(\mathcal{D}) = \{\pi \mid \pi^k = \pi_1, \ldots, \pi_k \models \varphi, \text{ for every } 1 \leq k \leq |\pi|\}$$

Given: LTL_f formula φ specifying safety properties

(i) Construct the DFA $\mathcal{A} = (2^{\mathcal{P}}, \mathcal{S}, s_0, \delta, \mathcal{F})$ of φ

(ii) **Obtain** the DSA as $\mathcal{D} = (2^{\mathcal{P}}, \mathcal{S}', s_0, \delta')$

(a) remove all non-accepting states

(b) remove all the transitions leading to non-accepting states

– all prefixes should satisfy φ, the run goes through accepting states $\mathcal{F}$

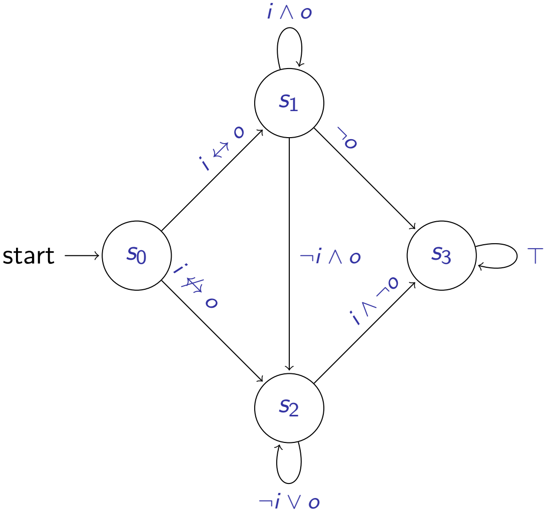
Given: LTL_f formula φ specifying safety properties

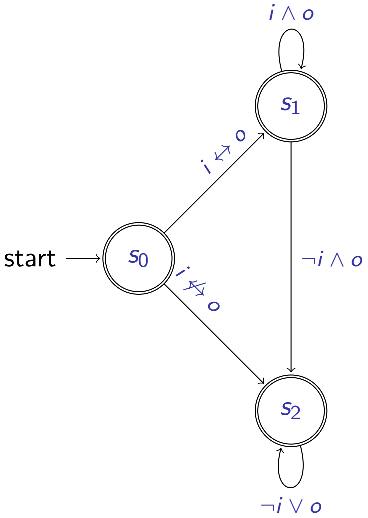
(i) Construct the DFA $\mathcal{A} = (2^{\mathcal{P}}, \mathcal{S}, s_0, \delta, \mathcal{F})$ of φ

(ii) **Obtain** the DSA as $\mathcal{D} = (2^{\mathcal{P}}, \mathcal{S}', s_0, \delta')$

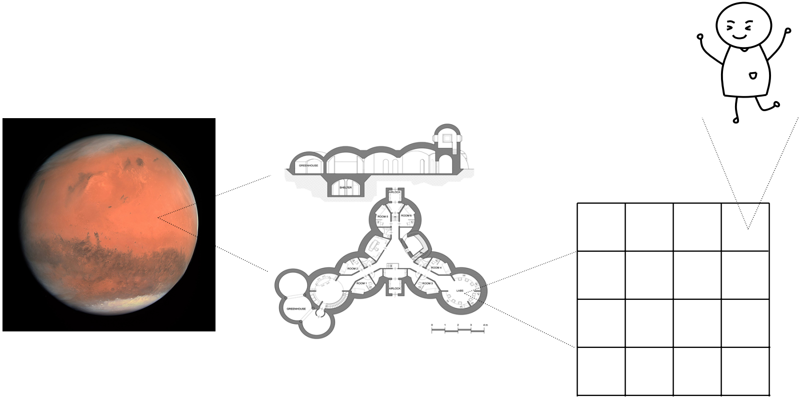
- $\mathcal{S}' = \mathcal{F}$
- $\delta'(s, \alpha) = \begin{cases} \delta(s, \alpha) & \text{if } s \in \mathcal{F} \\ \perp & \text{otherwise} \end{cases}$

DSA Construction of Safety Specification in LTL_f Formulas





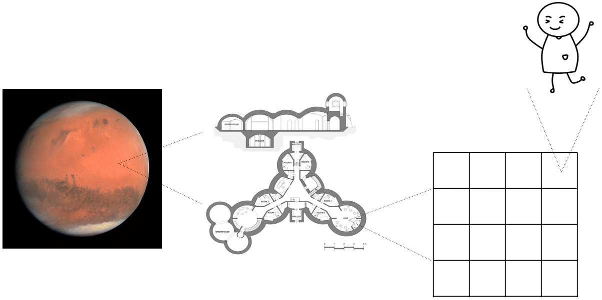
$i \wedge o$
s_1
$i \leftrightarrow o$
start
s_0
$\neg i \wedge o$
$i \not\leftrightarrow o$
s_2
$\neg i \vee o$



Mars image from ESA_OSIRIS, shelter image from "A Moon Base With Active Radiation Shielding" Caldera et al. 2018.

Environment: shelter

Agent: lab cleaning robot



The robot is sent to clean the dust in lab

- **Environment:** the status of the lab (including the robot) stays in safe boundaries
- **Agent:** clean the lab and leave

- **Environment:** the status of the lab (including the robot) stays in safe boundaries

- **Environment:** the status of the lab (including the robot) stays in safe boundaries *Dust*, *RobotOut*

- **Environment:** the status of the lab (including the robot) stays in safe boundaries *Dust*, *RobotOut*

- **Agent:** clean the lab and leave

- **Environment:** the status of the lab (including the robot) stays in safe boundaries
  Dust, *RobotOut*

- **Agent:** clean the lab and leave
  get_out, *clean_dust*

Essential fragments of the environment safety specification $\mathcal{E}_s$:

There is *Dust* in the lab, and the robot is inside

$$\neg RobotOut \wedge Dust$$

If robot *clean_dust*, then *Dust* is removed

$$\Box((clean\_dust) \rightarrow (\circ(\neg RobotOut \wedge \neg Dust)))$$

If the robot moves out, then the robot is out

$$\Box(get\_out \rightarrow \circ RobotOut)$$

Agent goal $\mathcal{G}$:

$$\Diamond(\mathit{RobotOut} \wedge \neg \mathit{Dust})$$

Given:

Environment safety specification $\mathcal{E}_s$

Agent goal φ

Obtain:

Agent strategy $\sigma_{ag} : (2^{\mathcal{X}})^+ \to 2^{\mathcal{Y}}$

$$\forall \sigma_{env} \rhd \mathcal{E}_s, trace(\sigma_{ag}, \sigma_{env})^k \models \varphi \text{ for some } k \in \mathbb{N}.$$

Agent task φ under Environment safety specification $\mathcal{E}_s$

as

Implication $\mathcal{E}_s \rightarrow \varphi$

Implication $\mathcal{E}_s \to \varphi$

$$\forall \sigma_{env} \textbf{ IF } trace(\sigma_{ag}, \sigma_{env}) \models_\forall \mathcal{E}_s \textbf{ THEN } trace(\sigma_{ag}, \sigma_{env})^k \models \varphi \text{ for some } k \in \mathbb{N}$$

Implication $\mathcal{E}_s \rightarrow \varphi$

$$\forall \sigma_{env} \textbf{ IF } trace(\sigma_{ag}, \sigma_{env}) \models_\forall \mathcal{E}_s \textbf{ THEN } trace(\sigma_{ag}, \sigma_{env})^k \models \varphi \text{ for some } k \in \mathbb{N}$$

$$\Rightarrow \forall \sigma_{env} \textbf{ NEG } trace(\sigma_{ag}, \sigma_{env}) \models_\forall \mathcal{E}_s \textbf{ OR } trace(\sigma_{ag}, \sigma_{env})^k \models \varphi \text{ for some } k \in \mathbb{N}$$

Implication $\mathcal{E}_s \to \varphi$

$\forall \sigma_{env}$ **IF** $trace(\sigma_{ag}, \sigma_{env}) \models_{\forall} \mathcal{E}_s$ **THEN** $trace(\sigma_{ag}, \sigma_{env})^k \models \varphi$ for some $k \in \mathbb{N}$

$\Rightarrow \forall \sigma_{env}$ **NEG** $trace(\sigma_{ag}, \sigma_{env}) \models_{\forall} \mathcal{E}_s$ **OR** $trace(\sigma_{ag}, \sigma_{env})^k \models \varphi$ for some $k \in \mathbb{N}$

$\Rightarrow \forall \sigma_{env}$ **NEG** $trace(\sigma_{ag}, \sigma_{env}) \models_{\forall} \mathcal{E}_s$ **OR** $trace(\sigma_{ag}, \sigma_{env})^k \models \varphi$ for some $k \in \mathbb{N}$

Implication $\mathcal{E}_s \rightarrow \varphi$

$\forall \sigma_{env}$ **IF** $trace(\sigma_{ag}, \sigma_{env}) \models_\forall \mathcal{E}_s$ **THEN** $trace(\sigma_{ag}, \sigma_{env})^k \models \varphi$ for some $k \in \mathbb{N}$

$\Rightarrow \forall \sigma_{env}$ **NEG** $trace(\sigma_{ag}, \sigma_{env}) \models_\forall \mathcal{E}_s$ **OR** $trace(\sigma_{ag}, \sigma_{env})^k \models \varphi$ for some $k \in \mathbb{N}$

$\Rightarrow \forall \sigma_{env}$ **NEG** $trace(\sigma_{ag}, \sigma_{env}) \models_\forall \mathcal{E}_s$ **OR** $trace(\sigma_{ag}, \sigma_{env})^k \models \varphi$ for some $k \in \mathbb{N}$

$\Rightarrow \forall \sigma_{env}$ $trace(\sigma_{ag}, \sigma_{env})^k \models \neg\mathcal{E}_s$ **OR** $trace(\sigma_{ag}, \sigma_{env})^k \models \varphi$ for some $k \in \mathbb{N}$

Implication $\mathcal{E}_s \rightarrow \varphi$

$\forall \sigma_{env}$ **IF** $trace(\sigma_{ag}, \sigma_{env}) \models_\forall \mathcal{E}_s$ **THEN** $trace(\sigma_{ag}, \sigma_{env})^k \models \varphi$ for some $k \in \mathbb{N}$

$\Rightarrow \forall \sigma_{env}$ **NEG** $trace(\sigma_{ag}, \sigma_{env}) \models_\forall \mathcal{E}_s$ **OR** $trace(\sigma_{ag}, \sigma_{env})^k \models \varphi$ for some $k \in \mathbb{N}$

$\Rightarrow \forall \sigma_{env}$ **NEG** $trace(\sigma_{ag}, \sigma_{env}) \models_\forall \mathcal{E}_s$ **OR** $trace(\sigma_{ag}, \sigma_{env})^k \models \varphi$ for some $k \in \mathbb{N}$

$\Rightarrow \forall \sigma_{env}$ $trace(\sigma_{ag}, \sigma_{env})^k \models \neg\mathcal{E}_s$ **OR** $trace(\sigma_{ag}, \sigma_{env})^k \models \varphi$ for some $k \in \mathbb{N}$

$\Rightarrow \forall \sigma_{env}$ $trace(\sigma_{ag}, \sigma_{env})^k \models \neg\mathcal{E}_s \vee \varphi$ for some $k \in \mathbb{N}$

Implication $\mathcal{E}_s \rightarrow \varphi$

$\forall \sigma_{env}$ **IF** $trace(\sigma_{ag}, \sigma_{env}) \models_{\forall} \mathcal{E}_s$ **THEN** $trace(\sigma_{ag}, \sigma_{env})^k \models \varphi$ for some $k \in \mathbb{N}$

$\Rightarrow \forall \sigma_{env}$ **NEG** $trace(\sigma_{ag}, \sigma_{env}) \models_{\forall} \mathcal{E}_s$ **OR** $trace(\sigma_{ag}, \sigma_{env})^k \models \varphi$ for some $k \in \mathbb{N}$

$\Rightarrow \forall \sigma_{env}$ **NEG** $trace(\sigma_{ag}, \sigma_{env}) \models_{\forall} \mathcal{E}_s$ **OR** $trace(\sigma_{ag}, \sigma_{env})^k \models \varphi$ for some $k \in \mathbb{N}$

$\Rightarrow \forall \sigma_{env}$ $trace(\sigma_{ag}, \sigma_{env})^k \models \neg\mathcal{E}_s$ **OR** $trace(\sigma_{ag}, \sigma_{env})^k \models \varphi$ for some $k \in \mathbb{N}$

$\Rightarrow \forall \sigma_{env}$ $trace(\sigma_{ag}, \sigma_{env})^k \models \neg\mathcal{E}_s \vee \varphi$ for some $k \in \mathbb{N}$

$\Rightarrow$ Synthesis of LTL_f formula $(\neg\mathcal{E}_s \vee \varphi)$

Algorithm for LTL_f synthesis under Environment safety specifications

1. Given task φ (LTL_f), env safety specification $\mathcal{E}_s$ (LTL_f)
2. Obtain LTL_f formula $\neg\mathcal{E}_s \vee \varphi$

Algorithm for LTL_f synthesis

1. Given LTL_f formula
2. Compute corresponding DFA (double-exponential)
3. Synthesize winning strategy for reachability game (linear)
4. Return strategy

LTL_f Synthesis Under Environment Safety and Fairness Specifications

Simple Fairness: certain environment behavior occurs infinitely often

Simple Fairness: certain environment behavior occurs infinitely often

- LTL-definable

Simple Fairness: certain environment behavior occurs infinitely often

- LTL-definable
- $\Box\Diamond\alpha$: Boolean formula α holds infinitely often, α over env vars $\mathcal{I}$

Mars Robot Example Continue

Environment Safety:

Environment Safety:

- Initial state of the lab

Environment Safety:

- Initial state of the lab
- How the *Dust* and *Robot_Out* change with respect to agent actions

Environment Safety:

- Initial state of the lab
- How the *Dust* and *Robot_Out* change with respect to agent actions

Simple Environment Fairness:

Environment Safety:

- Initial state of the lab
- How the *Dust* and *Robot_Out* change with respect to agent actions

Simple Environment Fairness:

- $\Box\Diamond\neg Dust$: *Dust* cleaned by the lab manager infinitely often

Given:

Environment safety specification $\mathcal{E}_s$, Environment fairness specification $\mathcal{E}_f$

Agent goal φ

Obtain:

Agent strategy $\sigma_{ag} : (2^{\mathcal{X}})^+ \to 2^{\mathcal{Y}}$

$$\forall \sigma_{env} \triangleright \mathcal{E}_s \wedge \mathcal{E}_f, trace(\sigma_{ag}, \sigma_{env})^k \models \varphi \text{ for some } k \in \mathbb{N}.$$

Agent task φ under Environment specification $\mathcal{E}_s \wedge \mathcal{E}_f$

as

Implication $\mathcal{E}_s \wedge \mathcal{E}_f \rightarrow \varphi$

Agent task φ under Environment specification $\mathcal{E}_s \wedge \mathcal{E}_f$

as

Implication $\mathcal{E}_s \wedge \mathcal{E}_f \rightarrow \varphi$

$\Rightarrow \mathcal{E}_f \rightarrow (\neg\mathcal{E}_s \vee \varphi)$

Agent task φ under Environment specification $\mathcal{E}_s \wedge \mathcal{E}_f$

as

Implication $\mathcal{E}_s \wedge \mathcal{E}_f \to \varphi$

$\Rightarrow \mathcal{E}_f \to (\neg\mathcal{E}_s \vee \varphi)$

$\Rightarrow \mathcal{E}_f \to \varphi'$

$$\textbf{IF } \underbrace{\textit{environment fairness}}_{\mathrm{LTL}\ \Box\Diamond(\alpha)} \textbf{ THEN } \underbrace{\textit{agent goal } \varphi'}_{\mathrm{LTL}_f}$$

IF $\underbrace{\textit{environment fairness}}_{\text{LTL}_{\Box\Diamond(\alpha)}}$ **THEN** $\underbrace{\textit{agent goal } \varphi'}_{\text{LTL}_f}$

Agent: $\mathit{trace}(\sigma_{ag}, \sigma_{env})$ satisfies either

IF $\underbrace{\textit{environment fairness}}_{\text{LTL } \Box\Diamond(\alpha)}$ **THEN** $\underbrace{\textit{agent goal } \varphi'}_{\text{LTL}_f}$

Agent: $\textit{trace}(\sigma_{ag}, \sigma_{env})$ satisfies either

– Fairness $\Box\Diamond(\alpha)$ fails

IF $\underbrace{\textit{environment fairness}}_{\mathrm{LTL}\ \Box\Diamond(\alpha)}$ **THEN** $\underbrace{\textit{agent goal}\ \varphi'}_{\mathrm{LTL}_f}$

Agent: $\mathit{trace}(\sigma_{ag}, \sigma_{env})$ satisfies either

- Fairness $\Box\Diamond(\alpha)$ fails
- φ' holds, reaching accepting states of $\mathcal{A}'_{\varphi}$

IF $\underbrace{\textit{environment fairness}}_{\mathrm{LTL}_{\Box\Diamond(\alpha)}}$ **THEN** $\underbrace{\textit{agent goal } \varphi'}_{\mathrm{LTL}_f}$

Environment: $\mathit{trace}(\sigma_{ag}, \sigma_{env})$ satisfies both

IF $\underbrace{\textit{environment fairness}}_{\text{LTL } \Box\Diamond(\alpha)}$ **THEN** $\underbrace{\textit{agent goal } \varphi'}_{\text{LTL}_f}$

Environment: $\mathit{trace}(\sigma_{ag}, \sigma_{env})$ satisfies both

- Fairness $\Box\Diamond(\alpha)$ holds

IF $\underbrace{\textit{environment fairness}}_{\text{LTL } \Box\Diamond(\alpha)}$ **THEN** $\underbrace{\textit{agent goal } \varphi'}_{\text{LTL}_f}$

Environment: $\textit{trace}(\sigma_{ag}, \sigma_{env})$ satisfies both

- Fairness $\Box\Diamond(\alpha)$ holds
- φ' fails, never visiting accepting states of $\mathcal{A}_\varphi$

Environment: $trace(\sigma_{ag}, \sigma_{env})$ satisfies both

– Fairness $\Box\Diamond(\alpha)$ holds: Büchi game condition

Environment: $\mathit{trace}(\sigma_{ag}, \sigma_{env})$ satisfies both

- Fairness $\Box\Diamond(\alpha)$ holds: Büchi game condition
- φ' fails, never visiting accepting states of $\mathcal{A}_{\varphi}$: Safety game condition

– Büchi game condition (recurrent reachability)

$$\text{Buchi}(T) = \nu\mathcal{Y}.(\mu\mathcal{Z}.\text{force}_e(\underline{\delta(s, I \cup O) \in (T \cap \mathcal{Y})} \vee \underline{\delta(s, I \cup O) \in \mathcal{Z}}))$$

$\text{force}_e : \forall O \exists I$, since agent moves first

– Büchi game condition (recurrent reachability)

$$\text{Buchi}(T) = \nu\mathcal{Y}.(\mu\mathcal{Z}.\text{force}_e(\underline{\delta(s, I \cup O) \in (T \cap \mathcal{Y})} \vee \underline{\delta(s, I \cup O) \in \mathcal{Z}}))$$

– Safety game condition

$$\text{Safe}(S) = \nu\mathcal{Y}.(\text{force}_e(\delta(s, I \cup O) \in S \cap \mathcal{Y}))$$

$\text{force}_e : \forall O \exists I$, since agent moves first

Environment

– Büchi condition and Safety condition

$$\mathsf{Buchi}(\alpha) = \nu\mathcal{Y}.(\mu\mathcal{Z}.\mathsf{force}_e(\underline{I \models \alpha \wedge \delta(s, I \cup O) \in \mathcal{Y}} \vee \underline{\delta(s, I \cup O) \in \mathcal{Z}}))$$
$$\mathsf{Safe}(S) = \nu\mathcal{Y}.(\mathsf{force}_e(\delta(s, I \cup O) \in S \cap \mathcal{Y}))$$

– Büchi-Safety condition $\mathsf{Buchi\text{-}Safe}(\alpha, S) =$

$$\nu\mathcal{Y}.(\mu\mathcal{Z}.\mathsf{force}_e(\underline{I \models \alpha \wedge \delta(s, I \cup O) \in (\mathcal{Y} \cap S)} \vee \underline{\delta(s, I \cup O) \in (\mathcal{Z} \cap S)}))$$

$\mathsf{force}_e : \forall O \exists I$, since agent moves first

Environment

- Büchi-Safety condition

$$\nu\mathcal{Y}.(\mu\mathcal{Z}.\text{force}_e(\underline{I \models \alpha \wedge \delta(s, I \cup O) \in (\mathcal{Y} \cap S)} \vee \underline{\delta(s, I \cup O) \in (\mathcal{Z} \cap S)}))$$

Agent

- **Negate** Büchi-Safety condition

$$\mu\mathcal{Y}.(\nu\mathcal{Z}.\text{force}_{ag}(\underline{I \models \neg\alpha \vee \delta(s, I \cup O) \in (\mathcal{Y} \cup \overline{S})} \wedge \underline{\delta(s, I \cup O) \in (\mathcal{Z} \cup \overline{S})}))$$

$\text{force}_e : \forall O \exists I$, $\text{force}_{ag} : \exists O \forall I$, $\overline{S} = \mathcal{F}$, accepting states of DFA $\mathcal{A}_{\varphi'}$

Realizability

- $s_0 \in \mathcal{Y}_\infty$

Abstract strategy

- Assume $\mathcal{Z} = \mathcal{Y}_\infty$
- for $s \in \mathcal{Y}_{i+1} \backslash \mathcal{Y}_i$, set O to be

$$\forall I(\underline{I \models \neg\alpha \vee \delta(s, I \cup O) \in (\mathcal{Y}_i \cup \mathcal{F})} \wedge \underline{\delta(s, I \cup O) \in (\mathcal{Y}_\infty \cup \mathcal{F})})$$

LTL_f Synthesis Under Environment Safety and Fairness Specifications

1. Given task φ (LTL_f), env safety spec. $\mathcal{E}_s$ (LTL_f), env fairness spec. $\mathcal{E}_f$ (LTL)
2. Obtain new task $\varphi' = \neg\mathcal{E}_s \vee \varphi$ (LTL_f)
3. Compute corresponding DFA of φ' (double-exponential)
4. Synthesize winning strategy for **Neg** Büchi-Safety game (quadratic)
5. Return strategy

LTL_f Synthesis Under Environment Safety and Stability Specifications

Simple Stability: certain environment behavior eventually occurs forever

Simple Stability: certain environment behavior eventually occurs forever

– LTL-definable

Simple Stability: certain environment behavior eventually occurs forever

- LTL-definable
- $\Diamond\Box\alpha$: Boolean formula α eventually holds forever, α over env vars $\mathcal{I}$

Given:

Environment safety specification $\mathcal{E}_s$, Environment stability specification $\mathcal{E}_{st}$

Agent goal φ

Obtain:

Agent strategy $\sigma_{ag} : (2^{\mathcal{X}})^+ \to 2^{\mathcal{Y}}$

$$\forall \sigma_{env} \; \rhd \mathcal{E}_s \wedge \mathcal{E}_{st}, trace(\sigma_{ag}, \sigma_{env})^k \models \varphi \text{ for some } k \in \mathbb{N}.$$

Agent task φ under Environment specification $\mathcal{E}_s \wedge \mathcal{E}_{st}$

as

Implication $\mathcal{E}_{st} \rightarrow (\neg\mathcal{E}_s \vee \varphi)$

Agent task φ under Environment specification $\mathcal{E}_s \wedge \mathcal{E}_{st}$

as

Implication $\mathcal{E}_{st} \rightarrow (\neg\mathcal{E}_s \vee \varphi)$

$\Rightarrow \mathcal{E}_{st} \rightarrow \varphi'$

IF $\underbrace{\textit{environment stability}}_{\text{LTL } \Diamond\Box(\alpha)}$ **THEN** $\underbrace{\textit{agent goal } \varphi'}_{\text{LTL}_f}$

Environment: $\textit{trace}(\sigma_{ag}, \sigma_{env})$ satisfies both

– Stability $\Diamond\Box(\alpha)$ holds: coBüchi game condition

IF $\underbrace{\textit{environment stability}}_{\text{LTL } \Diamond\Box(\alpha)}$ **THEN** $\underbrace{\textit{agent goal } \varphi'}_{\text{LTL}_f}$

Environment: $\mathit{trace}(\sigma_{ag}, \sigma_{env})$ satisfies both

- Stability $\Diamond\Box(\alpha)$ holds: coBüchi game condition
- φ' fails, never visiting accepting states of $\mathcal{A}'_{\varphi}$: Safety game condition

– coBüchi game condition (reach and stay)

$$\text{coBuchi}(T) = \mu\mathcal{Y}.(\nu\mathcal{Z}.\text{force}_e(\underline{\delta(s, I \cup O) \in (T \cup \mathcal{Y})} \wedge \underline{\delta(s, I \cup O) \in \mathcal{Z}}))$$

$\text{force}_e : \forall O \exists I$, since agent moves first

– coBüchi game condition (reach and stay)

$$\mathrm{coBuchi}(T) = \mu\mathcal{Y}.(\nu\mathcal{Z}.\mathrm{force}_e(\underline{\delta(s, I \cup O) \in (T \cup \mathcal{Y})} \wedge \underline{\delta(s, I \cup O) \in \mathcal{Z}}))$$

– Safety game condition

$$\mathrm{Safe}(S) = \nu\mathcal{Y}.(\mathrm{force}_e(\delta(s, I \cup O) \in S \cap \mathcal{Y}))$$

$\mathrm{force}_e : \forall O \exists I$, since agent moves first

Environment

- coBüchi condition and Safety condition

$$\mathsf{coBuchi}(\alpha) = \mu\mathcal{Y}.(\nu\mathcal{Z}.\mathsf{force}_e(\underline{I \models \alpha \vee \delta(s, I \cup O) \in \mathcal{Y}} \wedge \underline{\delta(s, I \cup O) \in \mathcal{Z}}))$$

$$\mathsf{Safe}(S) = \nu\mathcal{Y}.(\mathsf{force}_e(\delta(s, I \cup O) \in S \cap \mathcal{Y}))$$

- coBüchi-Safety condition $\mathsf{coBuchi\text{-}Safe}(\alpha, S) =$

$$\mu\mathcal{Y}.(\nu\mathcal{Z}.\mathsf{force}_e(\underline{I \models \alpha \vee \delta(s, I \cup O) \in (\mathcal{Y} \cap S)} \wedge \underline{\delta(s, I \cup O) \in (\mathcal{Z} \cap S)}))$$

$\mathsf{force}_e : \forall O \exists I$, since agent moves first

Environment

- coBüchi-Safety condition $\text{coBuchi-Safe}(\alpha, S) =$

$$\mu\mathcal{Y}.(\nu\mathcal{Z}.\text{force}_e(\underline{I \models \alpha \vee \delta(s, I \cup O) \in (\mathcal{Y} \cap S)} \wedge \underline{\delta(s, I \cup O) \in (\mathcal{Z} \cap S)}))$$

Agent

- **Negate** coBüchi-Safety condition

$$\nu\mathcal{Y}.(\mu\mathcal{Z}.\text{force}_{ag}(\underline{I \models \neg\alpha \wedge \delta(s, I \cup O) \in (\mathcal{Y} \cup \overline{S})} \vee \underline{\delta(s, I \cup O) \in (\mathcal{Z} \cup \overline{S})}))$$

$\text{force}_e : \forall O \exists I$, $\text{force}_{ag} : \exists O \forall I$, $\overline{S} = \mathcal{F}$ accepting states of $\mathcal{A}_{\varphi'}$

LTL_f Synthesis Under Environment Safety and Stability Specifications

1. Given task φ (LTL_f), env safety spec. $\mathcal{E}_s$ (LTL_f), env stability spec. $\mathcal{E}_{st}$ (LTL)
2. Obtain new task $\varphi' = \neg\mathcal{E}_s \vee \varphi$ (LTL_f)
3. Compute corresponding DFA of φ' (double-exponential)
4. Synthesize winning strategy for **Neg** coBüchi-Safety game (quadratic)
5. Return strategy

LTL_f Synthesis Under Environment Safety and GR(1) Specifications

Generalized Reactivity (1), GR(1)

- powerful notion of fairness specification

$$\mathcal{E}_{gr} = \bigwedge_{i=1}^{m} \Box\Diamond\mathcal{J}_i \rightarrow \bigwedge_{j=1}^{n} \Box\Diamond\mathcal{K}_j$$

$\mathcal{J}$ and $\mathcal{K}$ are propositional formulas on $\mathcal{I} \cup \mathcal{O}$

Recall: Simple fairness ($\Box\Diamond\alpha$) and stability ($\Diamond\Box\alpha$), α propositional formula over env vars $\mathcal{I}$

Environment Safety:

- Initial state

Environment Safety:

- Initial state
- Environment change wrt agent actions *Dust*, *Robot_Out*

Environment Safety:

- Initial state
- Environment change wrt agent actions *Dust*, *Robot_Out*, and *Robot_Wait*

Environment Safety:

- Initial state
- Environment change wrt agent actions *Dust*, *Robot_Out*, and *Robot_Wait*

Environment GR(1):

- $\Box\Diamond Robot\_Wait \to \Box\Diamond\neg Dust$

Given:

Environment safety specification $\mathcal{E}_s$, Environment GR(1) specification $\mathcal{E}_{gr}$

Agent goal φ

Obtain:

Agent strategy $\sigma_{ag} : (2^{\mathcal{X}})^+ \to 2^{\mathcal{Y}}$

$$\forall \sigma_{env} \ \rhd \mathcal{E}_s \wedge \mathcal{E}_{gr}, \mathit{trace}(\sigma_{ag}, \sigma_{env})^k \models \varphi \text{ for some } k \in \mathbb{N}.$$

Agent task φ under Environment specification $\mathcal{E}_s \wedge \mathcal{E}_{gr}$

as

Implication $\Rightarrow \mathcal{E}_{gr} \rightarrow \varphi'$

Agent task φ under Environment specification $\mathcal{E}_s \wedge \mathcal{E}_{gr}$

as

Implication $\Rightarrow \mathcal{E}_{gr} \rightarrow \varphi'$

Environment: GR(1)-Safety

Agent: Negate “GR(1)-Safety”

Agent task φ under Environment specification $\mathcal{E}_s \wedge \mathcal{E}_{gr}$

as

Implication $\Rightarrow \mathcal{E}_{gr} \rightarrow \varphi'$

Environment: GR(1)-Safety

Agent: Negate "GR(1)-Safety"

$$\mathcal{E}_{gr} = \bigwedge_{i=1}^{m} \Box\Diamond\mathcal{J}_i \rightarrow \bigwedge_{j=1}^{n} \Box\Diamond\mathcal{K}_j$$

How to solve Negate "GR(1)-Safety"? ☹

Agent task φ under Environment specification $\mathcal{E}_s \wedge \mathcal{E}_{gr}$

as

Implication $\Rightarrow \mathcal{E}_{gr} \rightarrow \varphi'$

Environment: GR(1)-Safety

Agent: Negate "GR(1)-Safety"

$$\mathcal{E}_{gr} = \bigwedge_{i=1}^{m} \Box\Diamond\mathcal{J}_i \rightarrow \bigwedge_{j=1}^{n} \Box\Diamond\mathcal{K}_j$$

GR(1) game, novel approaches ☺

GR(1) game $\mathcal{G} = \langle \mathcal{V}, \mathcal{P}_0, \mathcal{P}_1, \theta_0, \theta_1, \rho_0, \rho_1, \psi \rangle$

- $\mathcal{V} = \mathcal{P}_0 \cup \mathcal{P}_1$ state space
- θ_1 Boolean formula over $\mathcal{P}_1$ initial of Player-1, θ_0 Boolean formula over $\mathcal{V}$ initial state of Player-0
- ρ_1 Boolean formula over $\mathcal{V} \cup \mathcal{P}_1'$ transitions of Player-1, ρ_0 Boolean formula over $\mathcal{V} \cup \mathcal{P}_1' \cup \mathcal{P}_0'$ transitions of Player-0
- Winning condition GR(1) formula ψ

Player-1 wins by violating ψ, or Player-0 cannot continue;
Otherwise, Player-0 wins

$P_0 = (P_0^0, P_0^1, P_0^2) \qquad P_1 = (P_1^0, P_1^1, P_1^2)$

Start	$\left.\begin{array}{l} P_0^0 = t \\ P_0^1 = f \\ P_0^2 = t \end{array}\right\} \theta_1$	$P_0^0 = t$, $P_0^1 = f$, $P_0^2 = f$	...	...
	$\left.\begin{array}{l} P_1^0 = f \\ P_1^1 = t \\ P_1^2 = f \end{array}\right\} \theta_0$	$P_1^0 = f$, $P_1^1 = f$, $P_1^2 = t$	...	...
	Play-1	Play-1		
	Play-0	Play-0		

Play-1 : $\boxed{\begin{array}{l} P_0^0 = t,\ P_0^1 = f,\ P_0^2 = t,\ P_1^0 = f,\ P_1^1 = t,\ P_1^2 = f \\ (P_0^0)' = t,\ (P_0^1)' = f,\ (P_0^2)' = f \end{array}} \models \varphi_1$

Play-0 : $\boxed{\begin{array}{l} P_0^0 = t,\ P_0^1 = f,\ P_0^2 = t,\ P_1^0 = f,\ P_1^1 = t,\ P_1^2 = t \\ (P_0^0)' = t,\ (P_0^1)' = f,\ (P_0^2)' = f,\ (P_1^0)' = f,\ (P_1^1)' = f,\ (P_1^2)' = t \end{array}} \models \varphi_0$

- Advanced techniques to solve GR(1) game
- Efficient implementation reduces from cubic time to quadratic time $O(m \cdot n \cdot |N|^2)$

Utilize GR(1) game solver to deal with LTL_f synthesis under GR(1) specifications

$\mathcal{G} = \langle \mathcal{V}, \mathcal{P}_0, \mathcal{P}_1, \theta_0, \theta_1, \rho_0, \rho_1, \psi \rangle$ (Player-1 moves first)

- $\mathcal{P}_0$ of Player-0, $\mathcal{P}_1$ of Player-1
- ρ_0, ρ_1 encodes possible moves of both players
- Player-1 wins by violating ψ or Player-0 not being able to continue

Synthesis of implication $\Rightarrow \mathcal{E}_{gr} \rightarrow \varphi'$

- Agent wins by violating $\mathcal{E}_{gr}$, or visiting $\mathcal{F}$ of $\mathcal{D}_{\varphi'}$

$\mathcal{G} = \langle \mathcal{V}, \mathcal{P}_0, \mathcal{P}_1, \theta_0, \theta_1, \rho_0, \rho_1, \psi \rangle$ (Player-1 moves first)

- $\mathcal{P}_0$ of Player-0, $\mathcal{P}_1$ of Player-1
- ρ_0, ρ_1 encodes possible moves of both players
- **Player-1** wins by violating ψ or Player-0 not being able to continue

Synthesis of implication $\Rightarrow \mathcal{E}_{gr} \rightarrow \varphi'$

- **Agent** wins by violating $\mathcal{E}_{gr}$, or visiting $\mathcal{F}$ of $\mathcal{D}_{\varphi'}$

Player-1 wins by **violating** $\mathcal{E}_{gr}$, or **visiting** $\mathcal{F}$ **of** $\mathcal{D}_{\varphi'}$ (not able to continue)

Define GR(1) game $\mathcal{G} = \langle \mathcal{V}, \mathcal{P}_0, \mathcal{P}_1, \theta_0, \theta_1, \rho_0, \rho_1, \psi \rangle$

- $\mathcal{V} = \mathcal{I} \cup \mathcal{O} \cup \mathcal{S}$, $\mathcal{P}_1 = \mathcal{O}$, and $\mathcal{P}_0 = \mathcal{I} \cup \mathcal{S}$
- $\theta_1 = \top$, and θ_0 formula satisfied by $V \in 2^{\mathcal{V}}$ iff $V\mid_{\mathcal{S}} = s_0$
- $\rho_1 = \top$, and ρ_0 formula satisfied by $(V \cup V') \in 2^{\mathcal{V} \cup \mathcal{V}'}$ iff $\delta(V\mid_{\mathcal{S}}, V'\mid_{\mathcal{I}' \cup \mathcal{O}'}) = V'\mid_{\mathcal{S}'}$ **and** $V'\mid_{\mathcal{S}'} \notin \mathcal{F}$
- $\psi = \mathcal{E}_{gr}$

DFA of φ' is $\mathcal{A}_{\varphi'} = (\mathcal{I}, \mathcal{O}, \mathcal{S}, s_0, \delta, \mathcal{F})$

Existing GR(1) game solvers[2]

- **Realizability**
- **Abstract strategy** for Player-1

[2] E.g., **Slugs** (https://github.com/VerifiableRobotics/slugs)

LTL_f Synthesis Under Environment Safety and GR(1) Specifications

1. Given task φ (LTL_f), env safety spec. $\mathcal{E}_s$ (LTL_f), env GR(1) spec. $\mathcal{E}_{gr}$ (LTL)
2. Obtain new task $\varphi' = \neg\mathcal{E}_s \vee \varphi$ (LTL_f)
3. Compute corresponding DFA of φ' (double-exponential)
4. Construct GR(1) game (linear)
5. Synthesize winning strategy for Player-1 in GR(1) game (quadratic)
6. Return strategy

LTL_f Synthesis Under Environment Safety Specifications: No reduction to implication

Given:

Environment safety specification $\mathcal{E}_s$

Agent goal φ

Obtain:

Agent strategy $\sigma_{ag} : (2^{\mathcal{X}})^+ \to 2^{\mathcal{Y}}$

$$\forall \sigma_{env} \rhd \mathcal{E}_s, trace(\sigma_{ag}, \sigma_{env})^k \models \varphi \text{ for some } k \in \mathbb{N}.$$

Key step: How to abstract $\{\sigma_{env} \mid \sigma_{env} \rhd \mathcal{E}_s\}$?

Definition

Let $\mathcal{P} = \langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$ be a synthesis problem, the maximally permissive strategy of φ is $\text{MaxSet}(\varphi) = \{\sigma_{ag} \mid \sigma_{ag} \rhd \varphi\}$.[3]

[3] Zhu and De Giacomo: Synthesis of Maximally Permissive Strategies for LTL_f Specifications

- Deterministic: $\sigma : (2^{\mathcal{I} \cup \mathcal{O}})^* \times 2^{\mathcal{I}} \to 2^{\mathcal{O}}$, returns a single agent action
- Nondeterministic: $\Pi : (2^{\mathcal{I} \cup \mathcal{O}})^* \times 2^{\mathcal{I}} \to 2^{2^{\mathcal{O}}}$, returns a set of agent actions to choose from nondeterministically

Every nondeterministic strategy Π captures a set of deterministic strategies σ

Theorem

Safety specifications admit a nondeterministic strategy Π that captures the maximally permissive strategy.[4]

Let's construct Π_{env} that captures $\{\sigma_{env} \mid \sigma_{env} \ \triangleright \mathcal{E}_s\}$ ☺

[4] Bernet et al.: Permissive strategies: from parity games to safety games

- Construct the corresponding DSA $\mathcal{D} = (\mathcal{I}, \mathcal{O}, \mathcal{S}, s_0, \delta)$
- Solve a safety game on $\mathcal{D}$, considering the environment as Player-0
- Obtain the winning region Win of Player-0

Attention: Agent moves first here!

Π_{env} capturing $\{\sigma_{env} \mid \sigma_{env} \;\rhd \mathcal{E}_s\}$, a transducer $\mathcal{T} = (\mathcal{I}, \mathcal{O}, \mathrm{Win}, s_0, \varrho, \omega)$

- Win: Player-0 winning states
- $\omega : \mathrm{Win} \times 2^{\mathcal{O}} \to 2^{(2^{\mathcal{I}})}$ is the output function such that
  $$\omega(s, O) = \begin{cases} \{I \mid \delta(s, I \cup O) \in \mathrm{Win}\} & \text{if } s \in \mathrm{Win}, \\ \emptyset & \text{otherwise.} \end{cases}$$
- $\varrho : \mathrm{Win} \times 2^{\mathcal{O}} \to 2^{\mathrm{Win}}$ is the transition function such that
  $\varrho(s, O) = \{s' \mid s' = \delta(s, I \cup O) \textit{ and } I \in \omega(s, O)\}$

Attention: Agent moves first here!

Π_{env} capturing $\{\sigma_{env} \mid \sigma_{env} \ \triangleright \mathcal{E}_s\}$, a transducer $\mathcal{T} = (\mathcal{I}, \mathcal{O}, \mathsf{Win}, s_0, \varrho, \omega)$

- Win: Player-0 winning states
- $\omega : \mathsf{Win} \times 2^{\mathcal{O}} \to 2^{(2^{\mathcal{I}})}$ is the output function such that
  $$\omega(s, O) = \begin{cases} \{I \mid \delta(s, I \cup O) \in \mathsf{Win}\} & \text{if } s \in \mathsf{Win}, \\ \emptyset & \text{otherwise.} \end{cases}$$
- $\varrho : \mathsf{Win} \times 2^{\mathcal{O}} \to 2^{\mathsf{Win}}$ is the transition function such that
  $\varrho(s, O) = \{s' \mid s' = \delta(s, I \cup O) \textit{ and } I \in \omega(s, O)\}$

Minimize $\mathcal{T}$ to speed up the subsequent computation ☺

$\mathcal{T}$ representing $\sigma_{env} \triangleright \mathcal{E}_s$, and agent goal φ

Agent strategy $\sigma_{ag} : (2^{\mathcal{X}})^+ \to 2^{\mathcal{Y}}$

$$\forall \sigma_{env} \triangleright \mathcal{E}_s, trace(\sigma_{ag}, \sigma_{env})^k \models \varphi \text{ for some } k \in \mathbb{N}.$$

- Intersection of $\mathcal{T}$ and DFA $\mathcal{A}_\varphi$
- Reachability game

LTL_f Synthesis Under Environment Safety Specifications (Direct)

1. Given task φ (LTL_f), env safety spec. $\mathcal{E}_s$ (LTL_f)
2. Construct DSA $\mathcal{D}$ of $\mathcal{E}_s$ (double-exponential)
3. Construct the MaxSet $\mathcal{T}_{\mathcal{E}_s}$ (linear)
4. Compute corresponding DFA $\mathcal{A}_\varphi$ of φ (double-exponential)
5. Intersection of $\mathcal{T}$ and $\mathcal{A}_\varphi$ (poly)
6. Synthesize winning strategy for reachability game (linear)
7. Return strategy

– Notable cases of LTL_f synthesis under Environment specifications

  – Safety, Safety & Fairness, Safety & Stability, Safety & GR(1)

[5] Zhu and De Giacomo: Synthesis of Maximally Permissive Strategies for LTL_f Specifications.

- Notable cases of LTL_f synthesis under Environment specifications
  - **Safety**, Safety & Fairness, Safety & Stability, Safety & GR(1)

Directions to explore:

[5] Zhu and De Giacomo: Synthesis of Maximally Permissive Strategies for LTL_f Specifications.

– Notable cases of LTL_f synthesis under Environment specifications

  – Safety, Safety & Fairness, Safety & Stability, Safety & GR(1)

Directions to explore:

– MaxSet of other environment specification fragments?

[5] Zhu and De Giacomo: Synthesis of Maximally Permissive Strategies for LTL_f Specifications.

- Notable cases of LTL_f synthesis under Environment specifications
  - Safety, Safety & Fairness, Safety & Stability, Safety & GR(1)

Directions to explore:

- MaxSet of other environment specification fragments?
  - LTL_f specification, two nondeterministic strategies to capture MaxSet [5]

[5] Zhu and De Giacomo: Synthesis of Maximally Permissive Strategies for LTL_f Specifications.

- Notable cases of LTL_f synthesis under Environment specifications
  - Safety, Safety & Fairness, Safety & Stability, Safety & GR(1)

Directions to explore:

- MaxSet of other environment specification fragments?
  - LTL_f specification, two nondeterministic strategies to capture MaxSet [5]
  - What about Fairness? Stability? GR(1)?

[5] Zhu and De Giacomo: Synthesis of Maximally Permissive Strategies for LTL_f Specifications.