LTL_f Synthesis Under Environment Specifications

Game-Theoretic Approach to Temporal Synthesis

Antonio Di Stasio

University of Oxford (joining City, University of London in August)

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

European Summer School on Artificial Intelligence
Athens, 15 - 19 July, 2024

Reactive Synthesis

Basic Idea: "Mechanical translation of human-understandable task specifications to a program that is known to meet the specifications." [Vardi - The Siren Song of Temporal Synthesis 2018]

Given a specification φ over input (fluents) F, controlled by the environment, and outputs (actions) A, controlled by agent, expressed in:

LTL (Pnueli 1977) or LTL$_f$ (De Giacomo, Vardi 2013)

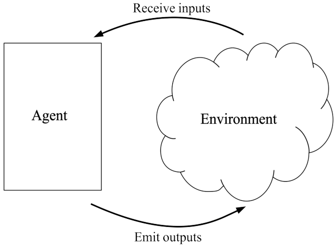
Syntax:

$$\varphi ::= a \mid \varphi \wedge \varphi \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi\mathcal{U}\varphi \mid \Diamond\varphi \mid \Box\varphi$$

Semantic:

A trace $trace$ is an infinite (LTL) or finite (LTL$_f$) sequence over F and A. We write $trace \models \varphi$ to mean that τ satisfies φ.

Reactive Synthesis



Agent and Environment Strategies, and Traces

For an agent strategy $\sigma_{ag} : F^* \to A$ and an environment strategy $\sigma_{env} : A^+ \to F$, the trace

$$trace(\sigma_{ag}, \sigma_{env}) = (A_1 \cup F_1), (A_2 \cup F_2) \ldots \in (2^{F \cup A})^\omega$$

denotes the unique trace induced by both σ_{ag} and σ_{env}.

Synthesis Problem (Church, 1962)

Given an LTL / LTLf task $Goal$ for the agent,

Find agent strategy σ_{ag} such that $\forall \sigma_{env}.\, trace(\sigma_{ag}, \sigma_{env}) \models Goal$

LTL Synthesis

Algorithm for LTL synthesis

Given LTL formula φ

1: Compute corresponding NBA (exponential)

2: Determinize NBA into DPA (exp in states, poly in priorities)

3: Synthesize winning strategy for Parity Game (poly in states, exp in priorities)

Complexity

LTL synthesis is 2EXPTIME-complete

Tools

- Spot[a]: a platform for LTL and ω-automata manipulation.
- Strix[b]: So far, the best tool for solving LTL synthesis.

[a] https://spot.ire.epita.fr/
[b] https://strix.model.in.tum.de/

LTL_f Synthesis

Algorithm for LTL_f synthesis

Given LTL_f formula φ

1: Compute corresponding NFA (exponential)

2: Determinize NFA to DFA (exponential)

3: Synthesize winning strategy for DFA game (linear)

Complexity

LTL_f synthesis is 2EXPTIME-complete

Tools

- [a]ltlf2dfaa: a tool for traslating ltlf into DFA.
- Syft, Lysa, Lydia,Cynthiab, etc.

[a]http://ltlf2dfa.diag.uniroma1.it/

Finite (Unbounded) Horizon in AI

Artificial Intelligence and in particular the Knowledge Representation and Planning community well aware of temporal logics since a long time.

- Temporally extended goals [BacchusKabanza96] - infinite/finite
- Temporal constraints on trajectories [GereviniHslumLongSaettiDimopoulos09 - PDDL3.0 2009] - finite
- Declarative control knowledge on trajectories [BaierMcIlraith06] - finite
- Procedural control knowledge on trajectories [BaierFrizMcIlraith07] - finite
- Temporal specification in planning domains [CalvaneseDeGiacomoVardi02] - infinite
- Planning via model checking - infinite

  Branching time (CTL) [CimattiGiunchigliaGiunchigliaTraverso97]
  Linear time (LTL) [DeGiacomoVardi99]

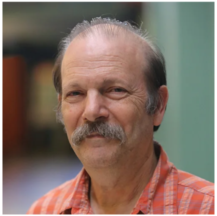
Foundations borrowed from temporal logics studied in CS, in particular:
Linear Temporal Logic (LTL) [Pnueli77].

However:

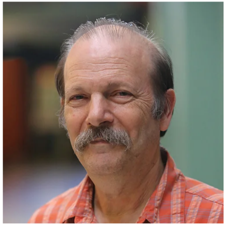
Often, LTL is interpreted on finite trajectories/traces.

We should consider for finite traces specifications

But that is easy



Precisely!



LTL_f in AI

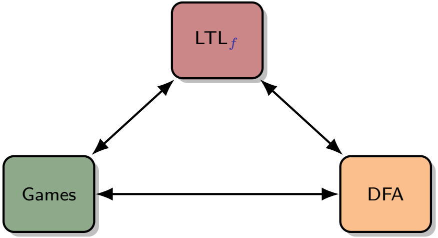
We are interested in building

AI Agents

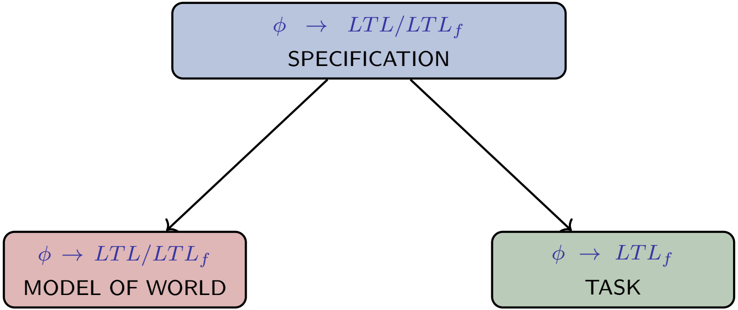
Linear temporal logics on finite traces are a fantastic tool for this enterprise, because it gives computational concreteness to the famous **Logics-Automata-Games** triangle from Formal Methods:

Agent Tasks terminate: Use LTL_f

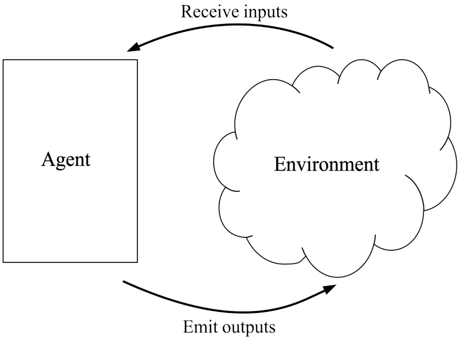
- Because it is the agent that is planning/reasoning.
- If the task would not terminate, the agent would be stuck into doing the same task forever.
- We want to focus on autonomous intelligent agents that (1) get a task, (2) reason/plan autonomously to solve it, (3) execute the plan, (4) get another task, and so on.



Synthesis with a Model of the World



Planning in nondeterministic domains



Domain

- Planning consider the agent acting in a **(nondeterministic) domain**
- The domain is a **model of how the world** (i.e. the environment) works
- That is, it is a **specification of the possible environment strategies**

Planning in nondeterministic domains

Nondeterministic domain

$\mathcal{D} = (2^{\mathcal{F}}, \mathcal{A}, s_0, \delta, \alpha)$ where:

- $\mathcal{F}$ fluents (atomic propositions)
- $\mathcal{A}$ actions (atomic symbols)
- $2^{\mathcal{F}}$ set of states
- s_0 initial state (initial assignment to fluents)
- $\alpha(s) \subseteq \mathcal{A}$ represents action preconditions
- $\delta(s, a, s')$ with $a \in \alpha(s)$ represents action effects.

Traces of D

Given a nondeterministic domain $D = (2^F, A, s_0, \alpha, \delta)$:

Traces

A D trace $s_0, a_1, s_1, \ldots, s_n$ induces a corresponding LTL-trace:

- If we pair action and the resulting state: $(dummy, s_0), (a_1, s_1), \ldots, (a_n, s_n)$, where dummy is a dummy starting action.
- if we pair state and the next action: $(s_0, a_1), (s_1, a_2), \ldots, (s_{n-1}, a_n), (s_n, dummy)$, where dummy is a dummy ending action.

The way we pair actions and states changes how we specify properties in LTL:

- If we pair action and the resulting state, we write: $\Box(\varphi_1 \rightarrow \bigcirc(a \rightarrow \varphi_2))$
- If we pair state and the next action, we write: $\Box((\varphi_1 \land a) \rightarrow \bigcirc\varphi_2)$

Planning (or Synthesis with a model of the world)

Domain as a specification of the environment

$$[[Dom]] = \{\sigma_{env} | \sigma_{env} \text{ compliant with } Dom\}$$

Planning in nondeterministic domains

Given an task $Goal$ for the agent, and a domain Dom modeling the environment

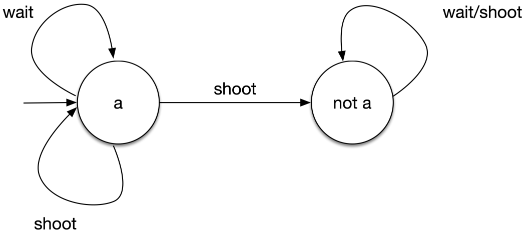
$$\text{Find agent behavior } \sigma_{ag} \text{ such that } \forall \sigma_{env} \in [[Dom]].trace(\sigma_{ag}\sigma_{env}) \models Goal$$

Which kinds of environment assumptions can the agent make?

For example let the assumption be formed by $Env_1 \wedge Env_2$ where:

Env_1 is the LTL formula expressing the dynamics of the environment (as a planning domain):

$\square(alive \rightarrow \bigcirc(wait \rightarrow alive))$
$\square(alive \rightarrow \bigcirc(shoot \rightarrow (alive \vee \neg alive))$
$\square(\neg alive \rightarrow \bigcirc(wait \rightarrow \neg alive))$
$\square(\neg alive \rightarrow \bigcirc(shoot \rightarrow \neg alive))$
$\square((wait \vee shoot) \wedge (wait \rightarrow \neg shoot) \wedge (shoot \rightarrow \neg wait))$



Env_2 is the LTL formula expressing some fairness over nondeterministic effects, e.g.,

$$\square\Diamond shoot \rightarrow \Diamond\neg a$$

Let $Goal$ be an LTL$_f$ formula which expresses an agent task, e.g.,

$$\Diamond\neg a$$

Which kinds of environment assumptions can the agent make?

For example let the assumption be formed by $Env_1 \wedge Env_2$ where:

Env_1 is the LTL formula expressing the dynamics of the environment (as a planning domain):

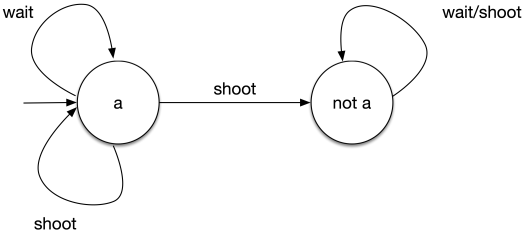
$\square(alive \rightarrow \bigcirc(wait \rightarrow alive))$

$\square(alive \rightarrow \bigcirc(shoot \rightarrow (alive \vee \neg alive))$

$\square(\neg alive \rightarrow \bigcirc(wait \rightarrow \neg alive))$

$\square(\neg alive \rightarrow \bigcirc(shoot \rightarrow \neg alive))$

$\square((wait \vee shoot) \wedge (wait \rightarrow \neg shoot) \wedge (shoot \rightarrow \neg wait))$



Env_2 is the LTL formula expressing some fairness over nondeterministic effects, e.g.,

$$\square\Diamond shoot \rightarrow \Diamond\neg a$$

Let $Goal$ be an LTL$_f$ formula which expresses an agent task, e.g.,

$$\Diamond\neg a$$

Which kinds of environment assumptions can the agent make?

For example let the assumption be formed by $Env_1 \wedge Env_2$ where:

Env_1 is the LTL formula expressing the dynamics of the environment (as a planning domain):

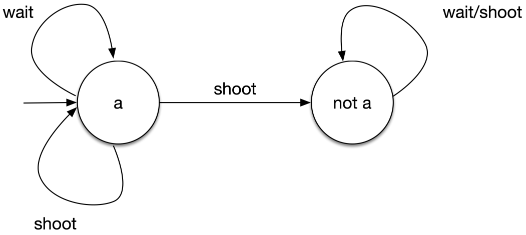
$\Box(alive \rightarrow \bigcirc(wait \rightarrow alive))$

$\Box(alive \rightarrow \bigcirc(shoot \rightarrow (alive \vee \neg alive))$

$\Box(\neg alive \rightarrow \bigcirc(wait \rightarrow \neg alive))$

$\Box(\neg alive \rightarrow \bigcirc(shoot \rightarrow \neg alive))$

$\Box((wait \vee shoot) \wedge (wait \rightarrow \neg shoot) \wedge (shoot \rightarrow \neg wait))$



Env_2 is the LTL formula expressing some fairness over nondeterministic effects, e.g.,

$$\Box\Diamond shoot \rightarrow \Diamond\neg a$$

Let $Goal$ be an LTL$_f$ formula which expresses an agent task, e.g.,

$$\Diamond\neg a$$

Safety Properties

Definition

A safety property is a property which specifies that some (bad) behavior will never occur.

Examples:

"always at most one process is in its critical section"

"money can only be withdrawn once a correct PIN has been provided"

Important property

Any infinite trace violating the property has a finite prefix that is "bad";

... two processes are in the critical section ...

.. in which money is withdrawn without issuing a PIN before..

Usually: $\Box\neg\ldots$

Safety Properties

Let $P \subseteq \Sigma^\omega$ be a property over Σ.

Definition

P is a safety property if there exists a language of finite words $L \subseteq \Sigma^*$ such that for every $w \in P$ all finite prefixes of w belong to L.

Safety Properties in LTL_f

Safety properties are properties on infinite traces, but if they can be broken at all, they can be broken with a finite prefix. This allows for capturing safety environment specification in LTL_f.

Planning Domains as Safety Properties

Planning Domains as Safety Properties

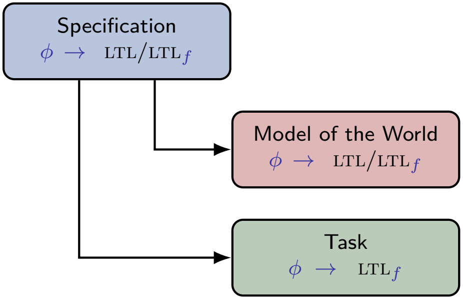
$$\Box(\varphi_1 \rightarrow \bigcirc(a \rightarrow \varphi_2))$$

Fully observable nondeterministic planning domains can be seen as safety properties in LTL/LTL_f: the environment forever reacts to actions as specified by the planning domain.

Synthesis with a Model of the World

Which environment specifications can the agent make?

- Nondeterministic planning domains; [DeGiacomoRubinIJCAI2018]
- Forms of fairness ($\Box\Diamond\phi$) and stability ($\Diamond\Box\phi$); [ZhuDeGiacomoPuVardiAAAI2020]
- Safety and Co-Safety (also called guarantee or reachability property) in tasks and environment specifications; [AminofDeGiacomoDiStasioFranconRubinZhuEUMAS23]
- GR(1) formulas; [DeGiacomoDiStasioTabajaraVardiZhuIJCAI2021]



Environments Specifications as LTL formulas

A natural generalization is to consider general environment specifications expressed as arbitrary LTL formulas. [DeGiacomoDiStasioVardiZhuKR2020]

Specifying possible environment specifications in LTL/LTL_f

Environment specifications in LTL/LTL_f

Let Env be an LTL/LTL_f formula over F and A.

$$[[Env]] = \{\sigma_{env} | \forall \sigma_{ag}.trace(\sigma_{ag}, \sigma_{env}) \models Env\}$$

i.e Env denotes all environment strategies that play according to the specification whatever is the agent strategy.

Synthesis under environment specifications in LTL/LTL_f

Given an LTL/LTL_f task $Task$ for the agent, and an LTL/LTL_f environment specification Env:

Find agent strategy σ_{ag} such that $\forall \sigma_{env} \in [[Env]].trace(\sigma_{ag}, \sigma_{env}) \models Goal$

Environment specifications in LTL/LTL$_f$

Consistent environment specifications

Is any LTL/LTL$_f$ formula a valid environment specification? No, Env needs to be "consistent"!:

$$[[Env]] \neq \emptyset \qquad \text{i.e. } \exists \sigma_e . \forall \sigma_{ag} . trace(\sigma_{ag}, \sigma_e) \models Env$$

Synthesis Under Environment Specifications

Environment Specifications

Let Env be an LTL/LTL$_f$ formula over $F \cup A$.

$$[[Env]] = \{\sigma_{env} | \sigma_{env} \text{ satisfies } Env \text{ whatever is the agent strategy}\}$$

Synthesis under environment specifications in LTL/LTL$_f$

Given an LTL/ LTL$_f$ task $Goal$ for the agent, and an LTL/LTL$_f$ environment specification Env:

$$\text{Find agent strategy } \sigma_{ag} \text{ such that } \forall \sigma_{env} \in [[Env]].trace(\sigma_{ag}, \sigma_{env}) \models Goal$$

Theorem [AminofDeGiacomoMuranoRubinICAPS2019]

To find agent strategy realizing $Goal$ under the environment specification Env, we can use standard LTL/LTL$_f$ synthesis for

$$Env \rightarrow Goal$$

Understanding why the reduction works is not immediate.

After all we are moving from a problem of the form:

$1-$ Find agent strategy σ_{ag} such that $\forall \sigma_{env} \in [[Env]].trace(\sigma_{ag}, \sigma_{env}) \models Goal$

to a problem of the form:

$2-$ Find agent strategy σ_{ag} such that $\forall \sigma_{env}.trace(\sigma_{ag}, \sigma_{env}) \models Env \rightarrow Goal$

Understanding why the reduction works is not immediate.

After all we are moving from a problem of the form:

$1-$ Find agent strategy σ_{ag} such that $\forall \sigma_{env} \in [[Env]].trace(\sigma_{ag}, \sigma_{env}) \models Goal$

to a problem of the form:

$2-$ Find agent strategy σ_{ag} such that $\forall \sigma_{env}.trace(\sigma_{ag}, \sigma_{env}) \models Env \rightarrow Goal$

In fact, one direction does hold on a strategy-by-strategy basis:

Theorem 1

Let Env be an LTL environment specification and $Goal$ an LTL goal. Then every agent strategy that realizes $Env \rightarrow Goal$ also realizes $Goal$ under environment specification Env.

Proof

- Let σ_{ag} be an agent strategy that realizes $Env \rightarrow Goal$, i.e., every trace induced by σ_{ag} satisfies $Env \rightarrow Goal$.
- To show that σ_{ag} realizes $Goal$ under the environment specification Env, let σ_{env} be an environment strategy realizing Env.
- We have that the trace $trace(\sigma_{ag}, \sigma_{env})$ induced by both strategies satisfies $Goal$.

However, **the converse does not hold**:

Theorem 2

It is not the case that, for every LTL environment specification Env and LTL goal $Goal$, every agent strategy that realizes $Goal$ under the environment specification Env also realizes $Env \rightarrow Goal$.

Proof

- Let $A = \{a\}$ and $F = \{f\}$, and let $Env = f \rightarrow a$ and $Goal = f \rightarrow \neg a$.
- First note that Env is a consistent LTL environment specification. Moreover, every environment strategy enforcing Env begins by playing $\neg f$.
- Every agent strategy realizes $Goal$ under the environment specification Env.
- However, not every agent strategy realizes $Env \rightarrow Goal$ (eg., the agent plays a in its first turn and the environment plays f).

Although the converse does not hold, the two problems are inter-reducible:

Theorem 3

Suppose Env is an LTL environment specification. The following are equivalent:

1. There is an agent strategy realizing $Env \to Goal$, i.e.,

$$\exists \sigma_{ag} \forall \sigma_{env} . trace(\sigma_{ag}, \sigma_{env}) \models Env \to Goal$$

2. There is an agent strategy realizing $Goal$ under environment specification Env, i.e.,

$$\exists \sigma_{ag} \forall \sigma_{env} \in [[Env]] . trace(\sigma_{ag}, \sigma_{env}) \models Goal$$

Proof: $1 \to 2$

Theorem 2 gives us $1 \to 2$.

Proof: $2 \to 1$

- Suppose 1 does not hold, i.e., the agent does not have a strategy to realize $Env \to Goal$.

- [Martin 1975] The environment has a strategy to realize $\neg(Env \to Goal)$, i.e., $\exists \sigma_{env} \forall \sigma_{ag}.trace(\sigma_{ag}, \sigma_{env}) \models Env \wedge \neg Goal$, i.e., σ_{env} realizes Env.

- Suppose that 2 holds and take σ_{ag} realizing $Goal$ under environment specification Env. Then by definition of realizability under environment specifications and using the fact that σ_{env} realizes Env, we have that $trace(\sigma_{ag}, \sigma_{env}) \models Goal$.

- On the other hand, we have already seen that $trace(\sigma_{ag}, \sigma_{env}) \models \neg Goal$, a contradiction.

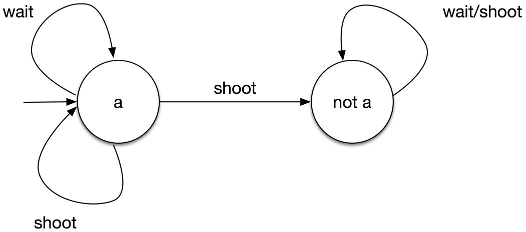
LTL$_f$ Synthesis Under LTL Environment Specifications

LTLf Synthesis Under LTL Environment Specifications

For example let the assumption be formed by $Env_1 \wedge Env_2$ where:

Env_1 is the LTL formula expressing the dynamics of the environment (as a planning domain):

$$\Box(alive \rightarrow \bigcirc(wait \rightarrow alive))$$
$$\Box(alive \rightarrow \bigcirc(shoot \rightarrow (alive \vee \neg alive)))$$
$$\Box(\neg alive \rightarrow \bigcirc(wait \rightarrow \neg alive))$$
$$\Box(\neg alive \rightarrow \bigcirc(shoot \rightarrow \neg alive))$$
$$\Box((wait \wedge shoot) \wedge (wait \rightarrow \neg shoot))$$



Env_2 is the LTL formula expressing some fairness over nondeterministic effects, e.g.,

$$\Box\Diamond shoot \rightarrow \Diamond\neg a$$

Let $Goal$ be an LTL_f formula which expresses an agent task, e.g.,

$$\Diamond\neg a$$

LTLf Synthesis Under LTL Environment Specifications

Problem

Solve the synthesis problem for

$$Env_1 \wedge Env_2 \rightarrow Goal$$

Naive Solution

Translate to LTL and then do standard LTL synthesis for $Env_1 \wedge Env_2 \rightarrow Goal$.

... but we can exploit the simplicity of dealing with LTL_f given:

LTLf Synthesis Under LTL Environment Specifications

Problem

Solve the synthesis problem for

$$Env_1 \wedge Env_2 \rightarrow Goal$$

Naive Solution

Translate to LTL and then do standard LTL synthesis for $Env_1 \wedge Env_2 \rightarrow Goal$.

... but we can exploit the simplicity of dealing with LTL_f given:

LTLf Synthesis Under LTL Environment Specifications

Problem

Solve the synthesis problem for

$$Env_1 \wedge Env_2 \rightarrow Goal$$

Naive Solution

Translate to LTL and then do standard LTL synthesis for $Env_1 \wedge Env_2 \rightarrow Goal$.

... but we can exploit the simplicity of dealing with LTL_f given:

LTLf Synthesis Under LTL Environment Specifications

Problem

Solve the synthesis problem for

$$Env_1 \wedge Env_2 \rightarrow Goal$$

Naive Solution

Translate to LTL and then do standard LTL synthesis for $Env_1 \wedge Env_2 \rightarrow Goal$.

... but we can exploit the simplicity of dealing with LTL_f given:

- Env_1: LTL

LTLf Synthesis Under LTL Environment Specifications

Problem

Solve the synthesis problem for

$$Env_1 \wedge Env_2 \rightarrow Goal$$

Naive Solution

Translate to LTL and then do standard LTL synthesis for $Env_1 \wedge Env_2 \rightarrow Goal$.

... but we can exploit the simplicity of dealing with LTL_f given:

- Env_1: ~~LTL~~ $\rightarrow$ LTL_f

LTLf Synthesis Under LTL Environment Specifications

Problem

Solve the synthesis problem for

$$Env_1 \wedge Env_2 \rightarrow Goal$$

Naive Solution

Translate to LTL and then do standard LTL synthesis for $Env_1 \wedge Env_2 \rightarrow Goal$.

... but we can exploit the simplicity of dealing with LTL_f given:

- Env_1: ~~LTL~~ $\rightarrow$ LTL_f
- Env_2: LTL

LTLf Synthesis Under LTL Environment Specifications

Problem

Solve the synthesis problem for

$$Env_1 \land Env_2 \rightarrow Goal$$

Naive Solution

Translate to LTL and then do standard LTL synthesis for $Env_1 \land Env_2 \rightarrow Goal$.

... but we can exploit the simplicity of dealing with LTL_f given:

- Env_1: ~~LTL~~ $\rightarrow$ LTL_f
- Env_2: LTL
- $Goal$: LTL_f

LTLf Synthesis Under LTL Environment Specifications

Separating LTL_f environment specificaitons

$$(Env_1 \land Env_2 \rightarrow Goal) \iff (Env_2 \rightarrow Env_1 \rightarrow Goal) \iff (Env_2 \rightarrow \neg Env_1 \lor Goal)$$

where $Goal' = \neg Env_1 \lor Goal$ is expressed in LTL_f and Env_2 in LTL.

Problem

Solve the synthesis problem for

$$Env_2 \rightarrow Goal'$$

How can we exploit that $Goal'$ is LTL_f?

Two-stage technique!

LTLf Synthesis Under LTL Environment Specifications

Separating LTL_f environment specificaitons

$$(Env_1 \wedge Env_2 \rightarrow Goal) \iff (Env_2 \rightarrow Env_1 \rightarrow Goal) \iff (Env_2 \rightarrow \neg Env_1 \vee Goal)$$

where $Goal' = \neg Env_1 \vee Goal$ is expressed in LTL_f and Env_2 in LTL.

Problem

Solve the synthesis problem for

$$Env_2 \rightarrow Goal'$$

How can we exploit that $Goal'$ is LTL_f?

Two-stage technique!

LTLf Synthesis Under LTL Environment Specifications

Separating LTL_f environment specificaitons

$$(Env_1 \wedge Env_2 \rightarrow Goal) \iff (Env_2 \rightarrow Env_1 \rightarrow Goal) \iff (Env_2 \rightarrow \neg Env_1 \vee Goal)$$

where $Goal' = \neg Env_1 \vee Goal$ is expressed in LTL_f and Env_2 in LTL.

Problem

Solve the synthesis problem for

$$Env_2 \rightarrow Goal'$$

How can we exploit that $Goal'$ is LTL_f?

Two-stage technique!

LTLf Synthesis Under LTL Environment Specifications

Separating LTL$_f$ environment specificaitons

$$(Env_1 \land Env_2 \rightarrow Goal) \iff (Env_2 \rightarrow Env_1 \rightarrow Goal) \iff (Env_2 \rightarrow \neg Env_1 \lor Goal)$$

where $Goal' = \neg Env_1 \lor Goal$ is expressed in LTL$_f$ and Env_2 in LTL.

Problem

Solve the synthesis problem for

$$Env_2 \rightarrow Goal'$$

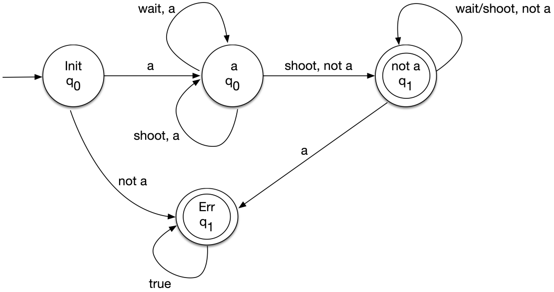
How can we exploit that $Goal'$ is LTL$_f$?

Two-stage technique!

Two-Stage Technique for Synthesis

1 Stage

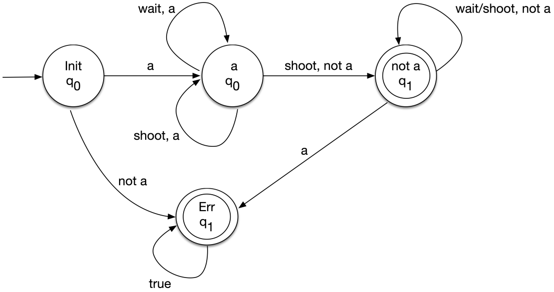
- Compute the corresponding DFA $\mathcal{A}$ of $\neg Env_1 \vee Goal$.
- Solve the reachability game for the agent over $\mathcal{A}$.
- Check whether the initial state is winning for the agent.
- If the initial state is not winning go to Stage 2, otherwise return the agent winning strategy.



Two-Stage Technique for Synthesis

1 Stage

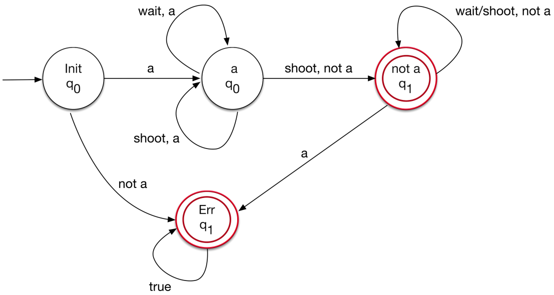
- Compute the corresponding DFA $\mathcal{A}$ of $\neg Env_1 \vee Goal$.
- Solve the reachability game for the agent over $\mathcal{A}$.
- Check whether the initial state is winning for the agent.
- If the initial state is not winning go to Stage 2, otherwise return the agent winning strategy.



Two-Stage Technique for Synthesis

1 Stage

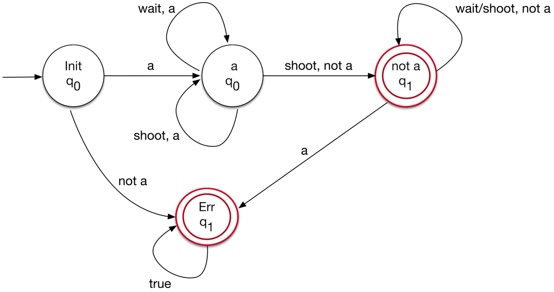
- Compute the corresponding DFA $\mathcal{A}$ of $\neg Env_1 \vee Goal$.
- Solve the reachability game for the agent over $\mathcal{A}$.
- Check whether the initial state is winning for the agent.
- If the initial state is not winning go to Stage 2, otherwise return the agent winning strategy.



Two-Stage Technique for Synthesis

1 Stage

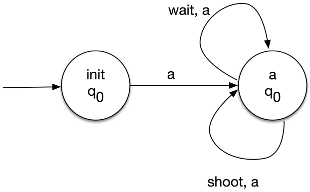
- Compute the corresponding DFA $\mathcal{A}$ of $\neg Env_1 \vee Goal$.
- Solve the reachability game for the agent over $\mathcal{A}$.
- Check whether the initial state is winning for the agent.
- If the initial state is not winning go to Stage 2, otherwise return the agent winning strategy.



Two-Stage Technique for Synthesis

2 Stage

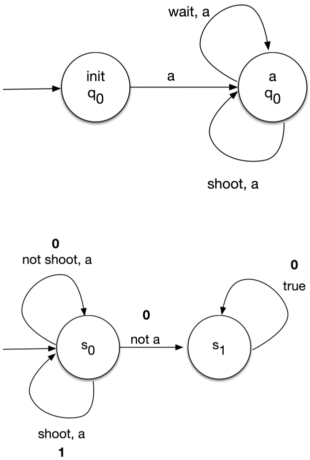
- Remove from $\mathcal{A}$ the agent winning set of Stage 1, say $\mathcal{A}'$.
- Compute the corresponding DPA $\mathcal{B}$ of Env_2.
- Do the cartesian product between $\mathcal{A}'$ and $\mathcal{B}$.
- Solve the parity game for the environment over $\mathcal{A}' \times \mathcal{B}$.
- Check if the initial state is winning for the agent; if not return "Unrealizable".
- Return the agent winning strategy by combing the agent winning strategies in Stage 1 and 2.



Two-Stage Technique for Synthesis

2 Stage

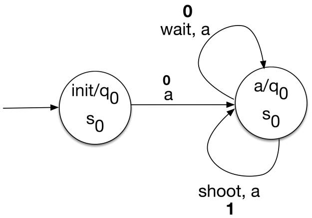
- Remove from $\mathcal{A}$ the agent winning set of Stage 1, say $\mathcal{A}'$.
- Compute the corresponding DPA $\mathcal{B}$ of Env_2.
- Do the cartesian product between $\mathcal{A}'$ and $\mathcal{B}$.
- Solve the parity game for the environment over $\mathcal{A}' \times \mathcal{B}$.
- Check if the initial state is winning for the agent; if not return "Unrealizable".
- Return the agent winning strategy by combing the agent winning strategies in Stage 1 and 2.



Two-Stage Technique for Synthesis

2 Stage

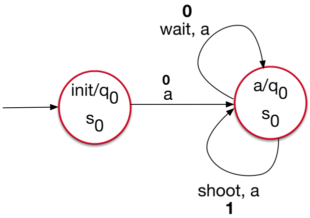
- Remove from $\mathcal{A}$ the agent winning set of Stage 1, say $\mathcal{A}'$.
- Compute the corresponding DPA $\mathcal{B}$ of Env_2.
- Do the cartesian product between $\mathcal{A}'$ and $\mathcal{B}$.
- Solve the parity game for the environment over $\mathcal{A}' \times \mathcal{B}$.
- Check if the initial state is winning for the agent; if not return "Unrealizable".
- Return the agent winning strategy by combing the agent winning strategies in Stage 1 and 2.



Two-Stage Technique for Synthesis

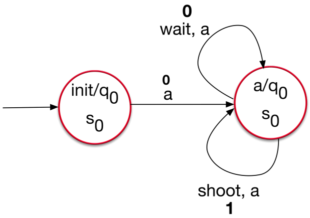
2 Stage

- Remove from $\mathcal{A}$ the agent winning set of Stage 1, say $\mathcal{A}'$.
- Compute the corresponding DPA $\mathcal{B}$ of Env_2.
- Do the cartesian product between $\mathcal{A}'$ and $\mathcal{B}$.
- Solve the parity game for the environment over $\mathcal{A}' \times \mathcal{B}$.
- Check if the initial state is winning for the agent; if not return "Unrealizable".
- Return the agent winning strategy by combing the agent winning strategies in Stage 1 and 2.



2 Stage

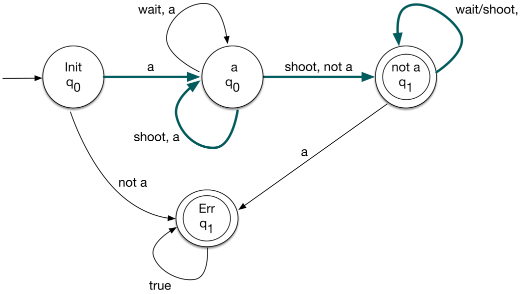
- Remove from $\mathcal{A}$ the agent winning set of Stage 1, say $\mathcal{A}'$.
- Compute the corresponding DPA $\mathcal{B}$ of Env_2.
- Do the cartesian product between $\mathcal{A}'$ and $\mathcal{B}$.
- Solve the parity game for the environment over $\mathcal{A}' \times \mathcal{B}$.
- Check if the initial state is winning for the agent; if not return "Unrealizable".
- Return the agent winning strategy by combing the agent winning strategies in Stage 1 and 2.



Two-Stage Technique for Synthesis

2 Stage

- Remove from $\mathcal{A}$ the agent winning set of Stage 1, say $\mathcal{A}'$.
- Compute the corresponding DPA $\mathcal{B}$ of Env_2.
- Do the cartesian product between $\mathcal{A}'$ and $\mathcal{B}$.
- Solve the parity game for the environment over $\mathcal{A}' \times \mathcal{B}$.
- Check if the initial state is winning for the agent; if not return "Unrealizable".
- Return the agent winning strategy by combing the agent winning strategies in Stage 1 and 2.



Experimental Analysis

We have

- implemented the two-stage technique in a new tool called **2SLS**, written in C++, that exploits CUDD package as library for the manipulation of Binary Decisions Diagrams (BDDs);
- compared **2SLS** to a direct reduction to LTL synthesis by employing the LTLf -to-LTL translator **SPOT** and **Strix** (Meyer, Sickert, and Luttenberger 2018) as the LTL synthesis solver;
- compared **2SLS** with FSyft and StSyft (Zhu et al. 2020) in special cases where environment specifications are LTL formulas of the form $\Box\Diamond a$ (fairness) and $\Diamond\Box a$ (stability), with a propositional.

Experiments on Fairness and Stability

- Given a counter game where the environment chooses whether to increment the counter or not and the agent can choose to grant the request or ignore it;
- The fairness environment specification is $\square\Diamond increment$; the stability environment specification is $\Diamond\square increment$;
- The goal is to get the counter having all bits set to 1.

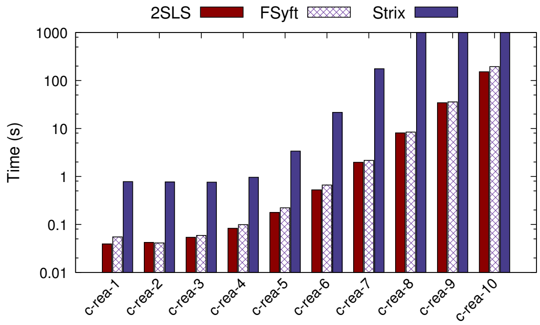


Figure: LTL_f synthesis under fairness environment specification.

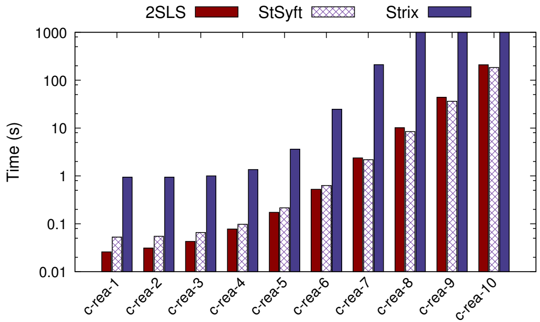


Figure: LTL_f synthesis under stability environment specificationmptions.

Experiments of General LTL Environment Specifications

- Given $Goal$ as a conjunction of increasing size of random LTL_f formulas of the form $\Box(p_j \rightarrow \Diamond q_j)$ with p_j and q_j propositions under the control of the environment and the agent, respectively;
- Env is a conjunction of formulas of the form $(\Box\Diamond p_i \vee \Diamond\Box q_i)$, where we start with one conjunct and introduce a new conjunct every 10 conjuncts in $Goal$.

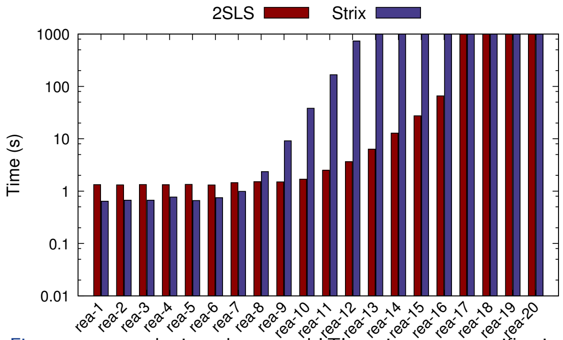


Figure: LTL_f synthesis under general LTL environment specifications.