Two-Stage Technique for LTLf Synthesis Under LTL Assumptions

Giuseppe De Giacomo[1], **Antonio Di Stasio**[1], Moshe Y. Vardi[2], Shufang Zhu[3]

[1]University of Rome "La Sapienza", [2]Rice University
[3]Shanghai Industrial Control Safety Innovation Technology Co.

ERC Advanced Grant
WhiteMech:
White-box Self Programming Mechanisms

SAPIENZA
UNIVERSITÀ DI ROMA

Reactive Synthesis

Given a specification φ over inputs $\mathcal{I}$ and outputs $\mathcal{O}$, expressed in:

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

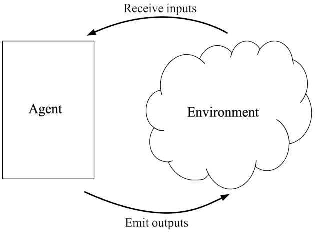
Syntax:

$$\varphi ::= a \mid \varphi \wedge \varphi \mid \neg\varphi \mid \circ\varphi \mid \varphi \, \mathrm{U} \, \varphi \mid \Diamond\varphi \mid \Box\varphi$$

Semantic:

A trace *trace* is an infinite (LTL) or finite (LTL$_f$) sequence over $\mathcal{I}$ and $\mathcal{O}$. We write $trace \models \varphi$ to mean that τ satisfies φ.

Reactive Synthesis



Agent and Environment Strategies, and Traces

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

$$trace(\sigma_{ag}, \sigma_{env}) = (i_1 \cup o_1), (i_2 \cup o_2) \ldots \in 2^{I \cup O}$$

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

Problem

Given an LTL/ LTL$_f$ task *Goal* for the agent

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

LTL and LTLf Synthesis

Algorithm for LTL synthesis

Given LTL formula φ

1: Compute corresponding NBA (Nondeterministic Buchi Aut.) (exponential)

2: Determinize NBA into DPA (Deterministic Parity Aut.) (exp in states, poly in priorities)

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

Algorithm for LTL_f synthesis

Given LTL_f formula φ

1: Compute corresponding NFA (Nondeterministic Finite Aut.) (exponential)

2: Determinize NFA to DFA (Deterministic Finite Aut.) (exponential)

3: Synthesize winning strategy for DFA game (linear)

Complexity

LTL and LTL_f synthesis are 2EXPTIME-complete

Synthesis Under Assumptions

Environment Assumptions

Let *Env* be an LTL/LTL$_f$ formula over $I \cup O$.

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

Synthesis with environment assumptions in LTL/LTL$_f$

Given an LTL/LTL$_f$ task *Goal* for the agent, and an LTL/LTL$_f$ environment assumption *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

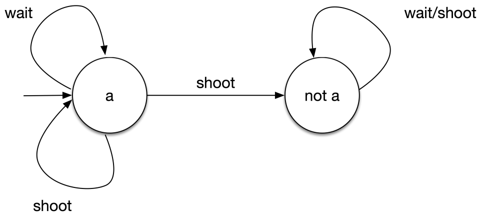
$$Env \rightarrow Goal$$

LTLf Synthesis Under LTL Assumptions

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):

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

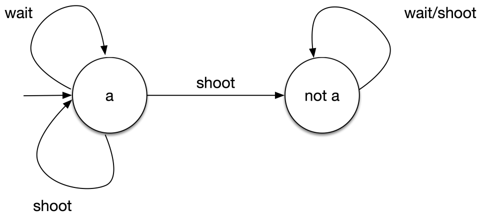


LTLf Synthesis Under LTL Assumptions

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):

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



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

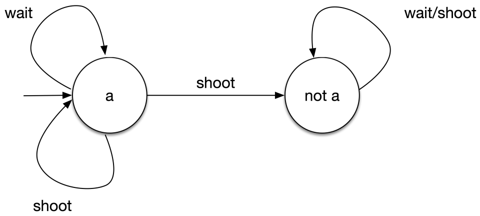
$$\Box\Diamond shoot \rightarrow \Diamond\neg a$$

LTLf Synthesis Under LTL Assumptions

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):

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



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 Assumptions

Problem

Solve the synthesis problem for

$$Env_1 \wedge Env_2 \rightarrow Goal$$

LTLf Synthesis Under LTL Assumptions

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 $Env_1 \wedge Env_2 \rightarrow Goal$.

LTLf Synthesis Under LTL Assumptions

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 Assumptions

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 Assumptions

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~~ $\rightarrow \mathrm{LTL}_f$

LTLf Synthesis Under LTL Assumptions

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~~ $\rightarrow \text{LTL}_f$
- Env_2: LTL

LTLf Synthesis Under LTL Assumptions

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
- $Goal$: LTL_f

LTLf Synthesis Under LTL Assumptions

Separating LTL_f assumptions

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

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

LTLf Synthesis Under LTL Assumptions

Separating LTL_f assumptions

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

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

Problem

Solve the synthesis problem for

$$Env_2 \rightarrow Goal'$$

LTLf Synthesis Under LTL Assumptions

Separating LTL$_f$ assumptions

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

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$_f$?

LTLf Synthesis Under LTL Assumptions

Separating LTL_f assumptions

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

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

Problem

Solve the synthesis problem for

$$Env_2 \rightarrow Goal'$$

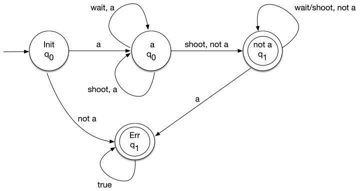
How can we exploit that $Goal'$ is LTL_f?

Two-stage technique!

Two-Stage Technique for Synthesis

1 ° Stage

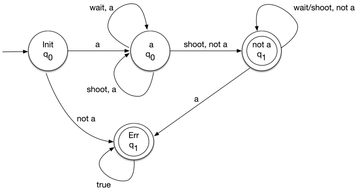
1. Compute the corresponding DFA $\mathcal{A}$ of $\neg Env_1 \vee Goal$.



Two-Stage Technique for Synthesis

1 ° Stage

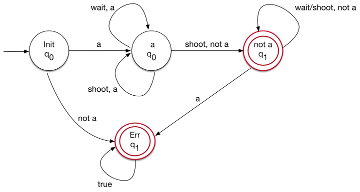
1. Compute the corresponding DFA $\mathcal{A}$ of $\neg Env_1 \vee Goal$.
2. Solve the reachability game for the agent over $\mathcal{A}$.



Two-Stage Technique for Synthesis

1 ° Stage

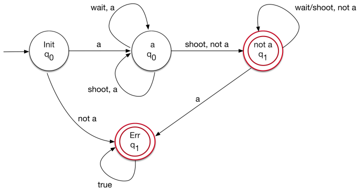
1. Compute the corresponding DFA $\mathcal{A}$ of $\neg Env_1 \vee Goal$.
2. Solve the reachability game for the agent over $\mathcal{A}$.
3. Check whether the initial state is winning for the agent.



Two-Stage Technique for Synthesis

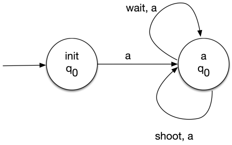
1 ° Stage

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



2 ° Stage

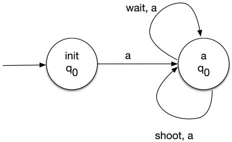
1. Remove from $\mathcal{A}$ the agent winning set of Stage 1, say $\mathcal{A}'$.

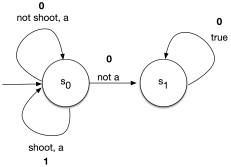


Two-Stage Technique for Synthesis

2° Stage

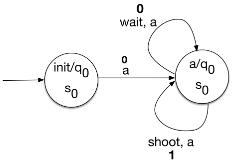
1. Remove from $\mathcal{A}$ the agent winning set of Stage 1, say $\mathcal{A}'$.
2. Compute the corresponding DPA $\mathcal{B}$ of Env_2.





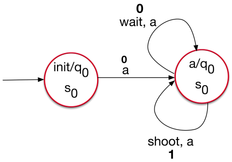
2 ° Stage

1. Remove from $\mathcal{A}$ the agent winning set of Stage 1, say $\mathcal{A}'$.
2. Compute the corresponding DPA $\mathcal{B}$ of Env_2.
3. Do the cartesian product between $\mathcal{A}'$ and $\mathcal{B}$.



2 ° Stage

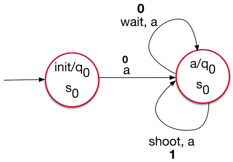
1. Remove from $\mathcal{A}$ the agent winning set of Stage 1, say $\mathcal{A}'$.
2. Compute the corresponding DPA $\mathcal{B}$ of Env_2.
3. Do the cartesian product between $\mathcal{A}'$ and $\mathcal{B}$.
4. Solve the parity game for the environment over $\mathcal{A}' \times \mathcal{B}$.



Two-Stage Technique for Synthesis

2 ° Stage

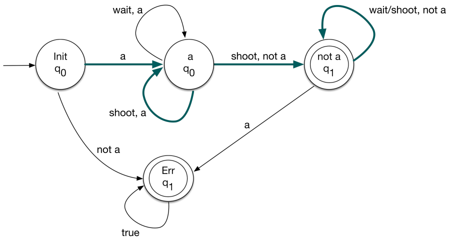
1. Remove from $\mathcal{A}$ the agent winning set of Stage 1, say $\mathcal{A}'$.
2. Compute the corresponding DPA $\mathcal{B}$ of Env_2.
3. Do the cartesian product between $\mathcal{A}'$ and $\mathcal{B}$.
4. Solve the parity game for the environment over $\mathcal{A}' \times \mathcal{B}$.
5. Check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



Two-Stage Technique for Synthesis

2 ° Stage

1. Remove from $\mathcal{A}$ the agent winning set of Stage 1, say $\mathcal{A}'$.
2. Compute the corresponding DPA $\mathcal{B}$ of Env_2.
3. Do the cartesian product between $\mathcal{A}'$ and $\mathcal{B}$.
4. Solve the parity game for the environment over $\mathcal{A}' \times \mathcal{B}$.
5. Check if the initial state is winning for the agent; if not return "Unrealizable".
6. 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 assumptions 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 assumption is $\Box\Diamond$*increment*; the stability assumption is $\Diamond\Box$*increment*;
- The goal is to get the counter having all bits set to 1.

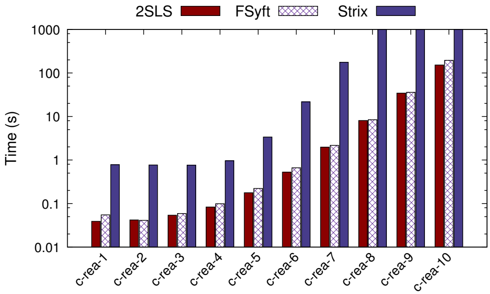


Figure 1: LTL_f synthesis under fairness assumptions.

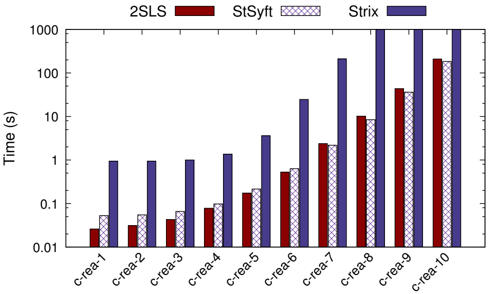


Figure 2: LTL_f synthesis under stability assumptions.

Experiments of General LTL Assumptions

- 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*.

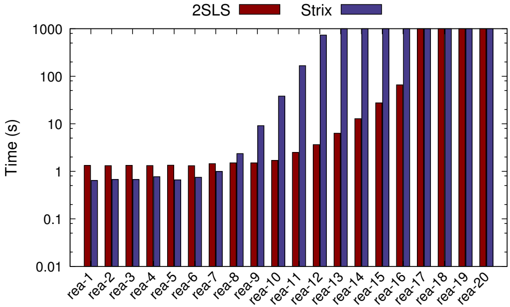


Figure 3: LTL$_f$ synthesis under general LTL assumptions.

Conclusion

We have:

- devised a two-stage technique for solving LTLf synthesis under LTL assumptions;
- implemented it in a new tool **2SLS**;
- showed the effectiveness by means of benchmarks.

Future Work

- Implement a tool to deal directly with planning domains.
- Consider assumptions expressed as GR(1) formulas.

Conclusion

We have:

- devised a two-stage technique for solving LTLf synthesis under LTL assumptions;
- implemented it in a new tool **2SLS**;
- showed the effectiveness by means of benchmarks.

Future Work

- Implement a tool to deal directly with planning domains.
- Consider assumptions expressed as GR(1) formulas.

to be continued....