LTL Realizability via Safety and Reachability Games

Alberto Camacho[1] Christian Muise[2] Jorge A. Baier[3] Sheila A. McIlraith[1]

[1] University of Toronto, Canada [2] IBM Research [3] Pontificia Universidad Católica de Chile

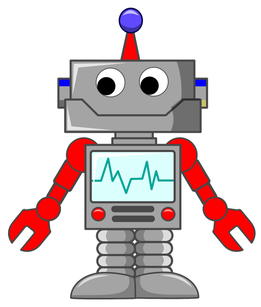
We address the problem of LTL realizability and synthesis. State of the art techniques rely on so-called *bounded synthesis* methods, which reduce the problem to a *safety* game. Realizability is determined by solving synthesis in a *dual* game. We provide a unified view of duality, and introduce novel *bounded realizability* methods via reductions to *reachability* games. Further, we introduce algorithms, based on AI automated planning, to solve these safety and reachability games. This is the the first complete approach to LTL realizability and synthesis via automated planning. Experiments illustrate that reductions to reachability games are an alternative to reductions to safety games, and show that planning can be a competitive approach to LTL realizability and synthesis.

www.cs.toronto.edu/~acamacho/synkit
contact: acamacho@cs.toronto.edu

LTL Realizability and Synthesis

Environment Player
Controls $\mathcal{X}$ variables

vs.



System Player
Controls $\mathcal{Y}$ variables

LTL Synthesis is usually interpreted as a 2-player game between the **Environment** and **System**.
In each *turn*:

- Environment selects $X_k \subseteq \mathcal{X}$
- Agent selects $Y_k \subseteq \mathcal{Y}$

A **play** is an *infinite* sequence of turns

$$w = (X_1 \cup Y_1)(X_2 \cup Y_2)\cdots$$

w is *winning* iff w satisfies a **given LTL specification**.

Realizability: Does there exist a winning strategy?
Synthesis: Compute a winning strategy

Linear Temporal Logic (LTL)

$$\varphi := p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \bullet\varphi \mid \varphi_1 \mathcal{U} \varphi_2$$

Atomic propositions $p \in AP$
Logic connectives: $\wedge, \vee, \neg$

Basic operators:	Other operators:
Next: $\bigcirc\varphi$	**Eventually:** $\varphi \equiv \mathbf{true}\mathcal{U}\varphi$
Weak Next: $\bullet\varphi$	**Always:** $\Box\varphi \equiv \neg\Diamond\neg\varphi$
Until: $\psi\mathcal{U}\chi$	**Release:** $\psi\chi \equiv \neg(\neg\psi\mathcal{U}\neg\chi)$

LTL and Infinite Word Automata

Satisfaction of an LTL formula can be checked with automata.
UCW: all runs must be accepting
NBW: some run must be accepting

Bounded Synthesis and Safety Games

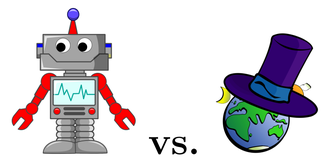
- *1959:* Circuit Synthesis introduced by Alonzo Church
- *1989:* LTL synthesis introduced by Pnueli and Rosner
- ...
- ... Big gap where no practical tools existed ...
- ...
- *2006:* Bounded synthesis introduced by Kupferman & Vardi
- Practical LTL synthesis tools Acacia, Lily, Unbeast,...
- *2016:* SYNTCOMP Annual LTL synthesis competition

vs.

Compute Winning strategy

Step 1: Transform LTL formula φ into an UCW $\mathcal{A}_\varphi$.
Step 2: Find winning strategy to UkCW games over $\mathcal{A}_\varphi$ for $k = 0, 1, \ldots$

- Specification is realizable iff some UkCW game is winning



vs.

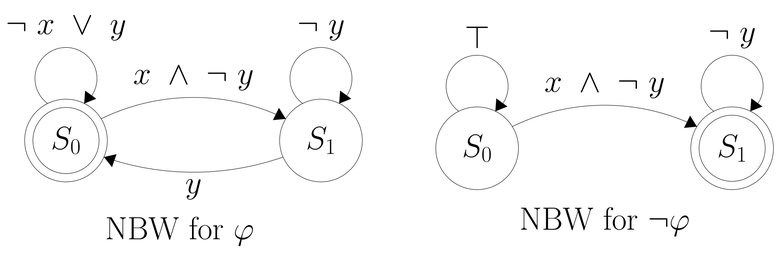
Compute Unrealizability Certificate

Step 1: Transform $\neg\varphi$ into an UCW $\mathcal{A}_{\neg\varphi}$.
Step 2: Find winning strategy to UkCW games over $\mathcal{A}_{\neg\varphi}$ for $k = 0, 1, \ldots$

- Specification unrealizable iff some UkCW game is winning

Bounded Realizability and Reachability Games

Example: $\varphi = \Box(x \rightarrow \Diamond y)$



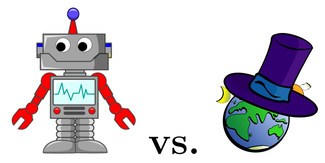
NkBW accepts a word if there exists a run that hits k or more accepting states.

vs.

Proof the specification is *not* realizable

Step 1: Transform LTL formula φ into an NBW $\mathcal{A}_\varphi$.
Step 2: Find winning strategy to NkBW games over $\mathcal{A}_\varphi$ for $k = 0, 1, \ldots$

- Spec. unrealizable iff some NkBW game not winning



vs.

Proof the specification is realizable

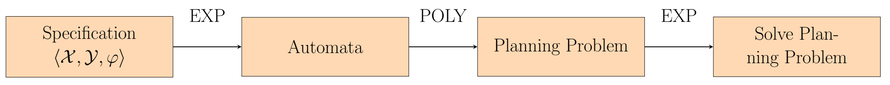
Step 1: Transform $\neg\varphi$ into an NBW $\mathcal{A}_{\neg\varphi}$.
Step 2: Find winning strategy to NkBW games over $\mathcal{A}_{\neg\varphi}$ for $k = 0, 1, \ldots$

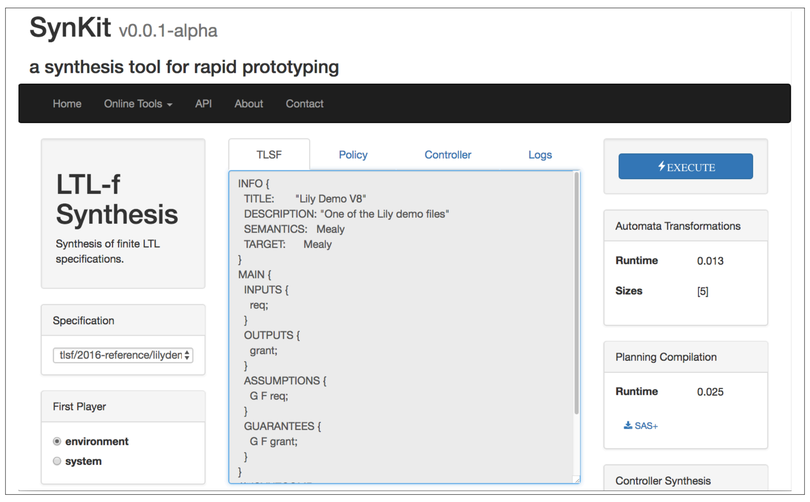
- Specification is realizable iff some NkBW game not winning

Experimental Results

	Real.		Unreal.							
Benchmark	NkBW	UkCW	NkBW	UkCW	SynKit	Acacia$^+$	Bowser	Bosy	Itlsynt	Party
Amba Decomposed (23)	15	14	–	–	15	14	18	22	22	22
Detector (6)	1	2	–	–	2	4	3	4	3	6
Detector Unreal (6)	–	–	4	5	5	0	4	6	3	6
Full Arbiter (6)	1	2	–	–	2	4	2	2	5	6
Full Arbiter Unreal (12)	–	–	9	9	9	0	8	12	12	12
Genbuf (5)	0	0	–	–	0	4	0	0	0	3
Generalized Buffer (5)	0	0	–	–	0	5	0	0	0	3
Lilydemo (24)	16	16	5	5	21	19	24	24	23	24
Load Balancer (5)	1	2	–	–	2	2	2	2	3	4
Load Balancer Unreal (12)	–	–	11	11	11	0	7	11	11	11
Loadcomp (4)	1	2	–	–	2	4	4	4	4	4
Loadfull (4)	1	2	–	–	2	3	4	4	4	4
LTL2DBA (27)	18	23	–	–	23	26	24	24	27	27
LTL2DPA (24)	17	23	–	–	23	23	24	24	23	24
Prio Arbitrer (6)	1	1	–	–	1	5	3	3	4	5
Prio Arbitrer Unreal (4)	–	–	1	1	1	0	3	3	3	3
RR Arbiter (6)	1	1	–	–	1	3	3	3	5	2
RR Arbiter Unreal (4)	–	–	1	2	2	0	3	2	3	3
Simple Arbiter (6)	1	2	–	–	2	4	3	4	4	5
Simple Arbiter Unreal (11)	–	–	2	2	2	0	6	8	9	8

Realizability and Synthesis via Automated AI Planning





Summary

- Introduced bounded realizability via NkBW reachability games
- Exploited planning to address LTL realizability and synthesis
- Algorithms and empirical evaluation
- Stepping stone towards synthesizing programs for IoT

References

[1] Orna Kupferman, Moshe Y. Vardi. **Safraless Decision Procedures.** FOCS 2005, pp 531-542.
[2] Sven Schewe, Bernd Finkbeiner. **Bounded Synthesis.** ATVA 2007, pp 474-488.