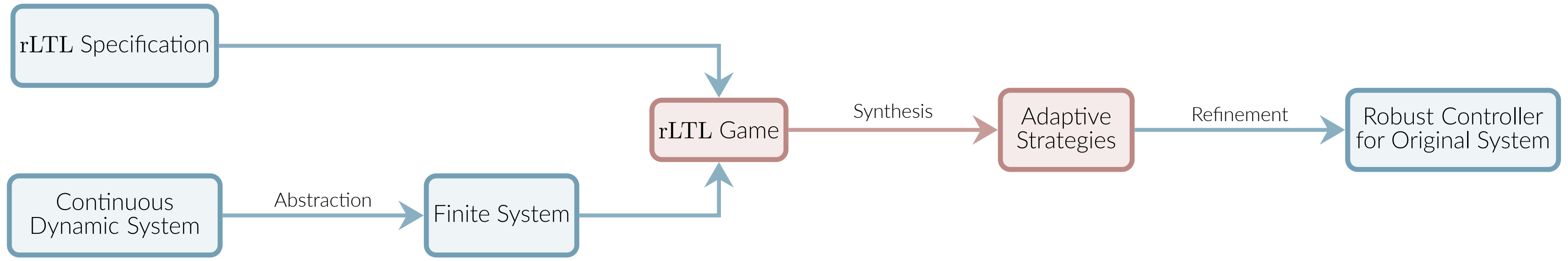
Adaptive Strategies for rLTL Games

Satya Prakash Nayak [1] Daniel Neider [2] Martin Zimmermann [3]

[1]Chennai Mathematical Institute [2]Max Planck Institute for Software Systems [3]University of Liverpool

Goal

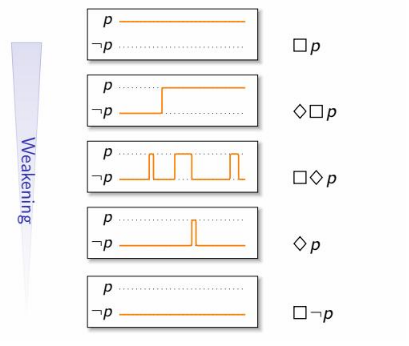
- We consider the problem of synthesizing most robust controllers using the Abstraction-Based Controller Design (ABCD) [1].
- To preserve robustness, we consider the specifications for the controllers to be expressed in Robust Linear Temporal Logic (rLTL) [2], which allows the reasoning about how robust the specification is.
- ABCD assumes the environment to always act antagonistically, which is often a non-realistic assumption. So is there a way of satisfying the specification "better" if the environment is not antagonistic?
- Suppose we want to satisfy some property p; and assuming the environment to be antagonistic, the best we can achieve is to satisfy p at finitely many positions of a word. We would like the controller to satisfy p at infinitely many positions (or even better) if possible when the environment is not antagonistic.



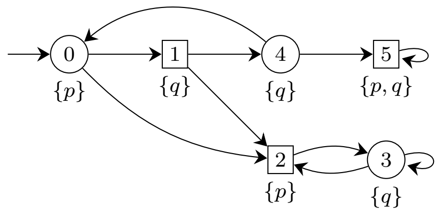
Robust Linear Temporal Logic

- The difference between "minor" and "major" violations of a formula cannot be distinguished in a 2-valued semantics.
- Consider the formula $\varphi = \Box p$, which demands that p holds at all positions of a word. Clearly, φ is violated even if p does not hold at only a single position, which is a very minor violation.
- To distinguish various degrees of violations, rLTL adopts a 5-valued semantics.
- For the formula $\Box p$, the robust version is written as $\boxdot p$, then, the five truth values distinguish the various degree of violations as shown in the figure on the right. Let $b_{\Box p}$ denotes the truth value for the top case in the figure, $b_{\Diamond\Box p}$ denotes truth value for the next case and so on.
- With this intuition, we can define a preference on truth values as follows:

$$b_{\Box p} > b_{\Diamond\Box p} > b_{\Box\Diamond p} > b_{\Diamond p} > b_{\Box\neg p}.$$



Robust LTL Games



□ : Player E (Environment)

○ : Player C (Controller)

rLTL Specification $\varphi = \boxdot p$

- The value of a play is the value of the rLTL formula φ on the word induced by labels of the play. For example, the value of the play $012323\ldots$ is the value of the formula $\boxdot p$ on the word $\{p\}\{q\}\{p\}\{q\}\ldots$
- Player C's objective is to maximize the value while Player E's wants is to minimize it.
- From play prefix 01, the controller strategy $\{0 \to 1; 4 \to 1; 3 \to 2\}$ enforces the play to visit 0 or 2 infinitely often, hence enforces the value $b_{\Box\Diamond p}$.
- As we have seen above, the classical synthesis algorithm is based on an overly pessimistic assumption on the environment, so we introduce two kinds of adaptive strategies.

References

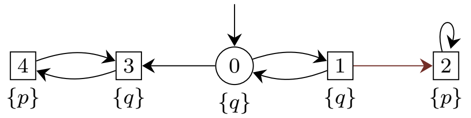
[1] Belta et al., 2017, *Formal methods for discrete-time dynamical systems*, Vol. 15, Springer.
[2] Tabuada and Neider, 2016, *Robust Linear Temporal Logic*, CSL.LIPIcs, Vol. 62, pp. 10:1–10:21.

Weakly Adaptive Strategy

- Weakly adaptive strategy is a strategy that adapts its moves to ensure the optimality even when the environment has made a bad move (by "bad", we mean the moves which are not optimal).
- Formally, a controller strategy σ is weakly adaptive if no strategy enforces a better value than σ from any play prefix.
- In the example on the bottom left, the best possible scenario for Player C assuming Player E plays his best moves is to enforce a play where p holds at infinitely many positions
- A classical strategy for Player C is $\{0 \to 1; 3 \to 2; 4 \to 1\}$ which enforces the play to visit the vertex 2 infinitely often.
- However, if Player E makes a bad move of $1 \to 4$, then Player C can force the play to eventually just stay at the vertex 5, and hence, p holds eventually always.
- Therefore, a weakly adaptive strategy for Player C is $\{0 \to 1; 3 \to 2; 4 \to 1\}$ which enforces a play where p holds eventually always if the token ever reaches the vertex 4; otherwise, enforces a play where p holds at infinitely many positions.

Strongly Adaptive Strategy

- Strongly adaptive strategy is a weakly adaptive strategy that also maximizes the opportunities of the environment making bad moves.
- For the example on the bottom left, another weakly adaptive strategy for Player C is $\{0 \to 2; 3 \to 2; 4 \to 1\}$. However, then the token can never reach the vertex 4 and hence, there cannot be a play where p holds eventually always.
- Hence, $\{0 \to 1; 3 \to 2; 4 \to 1\}$ is a better one and such a strategy is strongly adaptive.



- Now for the above game, if Player E plays his best moves, then the best possible play Player C can enforce is the one where p holds at infinitely many positions (e.g., a play with suffix $03434\ldots$).
- Unless Player E makes a bad move by moving along $1 \to 2$, any weakly adaptive strategy for Player C will eventually make him move the token to 3.
- But if Player C moves along $0 \to 1$, then there is a chance of Player E making a bad move of $1 \to 2$, and hence the token stays at the vertex 2, inducing a play where p holds eventually always.
- So, if σ_k is a strategy for Player C, which makes him move along $0 \to 1$ the first k times it reaches 0 and then moves to 3; then σ_{k+1} is always a better strategy than σ_k. Hence, no strongly adaptive strategy exists.

Theoretical Results

- Weakly adaptive strategy always exists for a game, whereas strongly adaptive strategy may not exist for some cases.
- It can be shown that both the strategies can be computed (if exists) in doubly-exponential time, and hence are not harder than the classical synthesis problems.

satyaprakash@cmi.ac.in neider@mpi-sws.org Martin.Zimmermann@liverpool.ac.uk