Sweap: Reactive Synthesis for Infinite-State Integer Problems

Shaun Azzopardi[1]
Luca Di Stefano[2]
Nir Piterman[3]

[1]Dedaub [2]TU Wien [3]Univ. Gothenburg/Chalmers

CAV, July 2026

∞-state Synthesis: Sweap's Setting

Novel features are highlighted

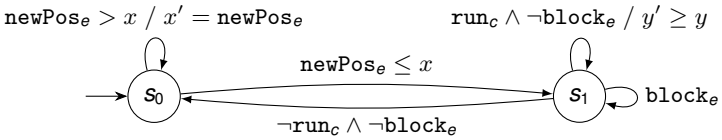
Arena: $A = \langle V, \mathbb{E}, \mathbb{C}, val_0, \delta \rangle$ Game: $\langle A, \varphi \rangle$

- V: state variables (integers)
- $\mathbb{E}$: inputs (Boolean or integers), $\mathbb{C}$: outputs (Boolean)
- val_0: initial valuation of state variables (optional)
- δ: transition relation (nondeterminism resolved by controller)
- $\varphi \in \mathrm{LTL}(\mathbb{E} \cup \mathbb{C} \cup \mathcal{P}r)$, $\mathcal{P}r =$ set of predicates over V
- E.g., $G\ ((x = 0) \Rightarrow F\ (x = 5))$

Realizability modulo arena

- Let environment $\mathcal{E}$ set the values of predicates.
- A trace is concretisable iff $\mathcal{E}$'s values for predicates agree with the valuation of V at every step.
- φ is realizable modulo A iff there is a strategy that preserves φ on every concretisable trace

Example: Infinite Race



$$V = \{x \overset{\text{init}}{=} 0, y \overset{\text{init}}{=} 0\}; \quad \mathbb{E} = \{\texttt{newPos}_e, \texttt{block}_e\}; \quad \mathbb{C} = \{\texttt{run}_c\}$$

Goal: $GF(s_1 \wedge \neg\texttt{block}) \implies GF(s_0 \wedge (y > x))$

Assumption

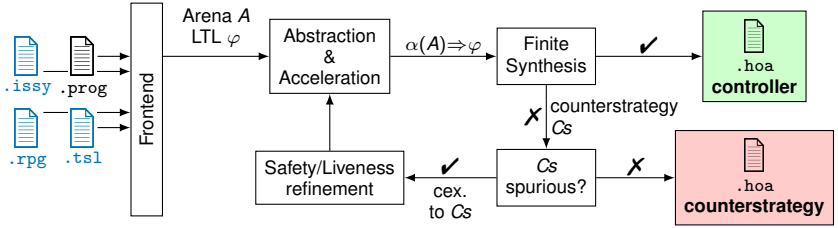
Environment cannot stay in s_0 forever, cannot block controller forever

Guarantee

Controller cannot stay in s_1 forever, must be ahead of the environment ∞ often

Sweap's CEGAR Approach

Novel features are highlighted

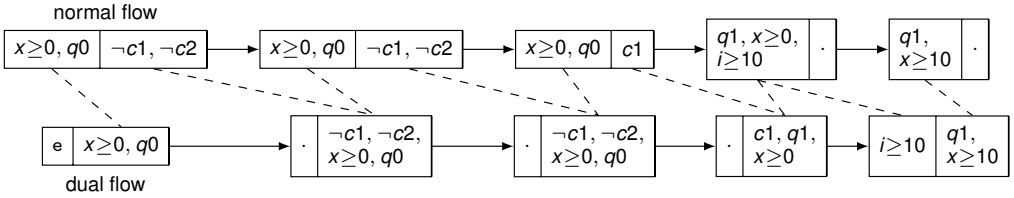


- Liveness refinement prunes unconcretisable infinite loops
- Acceleration: fairness assumptions on truth values of predicates
- Finite synthesis backends: Strix, SemML (new default)
- Main algorithmic contribution: dualization

Dualization

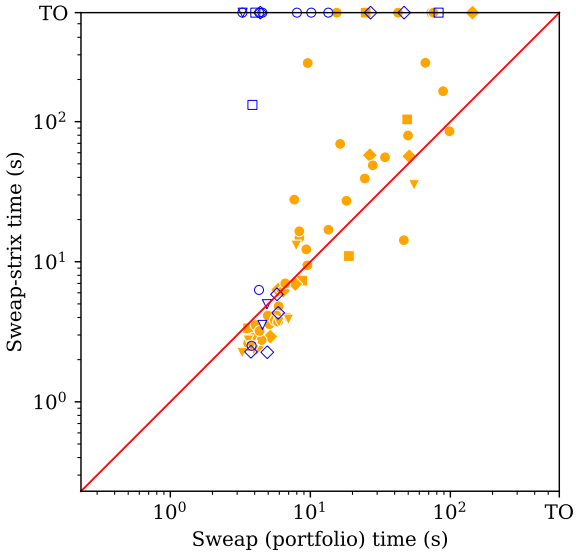
- Recall: we let $\mathcal{E}$ choose predicate values
- This approach does not terminate on some problems
- Specifically, those that do not admit finite counterstrategies

- Dualization flips control of the predicates to the controller
- However, this slightly breaks the order of play in the game
  $\Rightarrow$ We also turn φ into equirealisable φ^{dual}
  (add Xs, put additional constraints on $\mathcal{E}$)



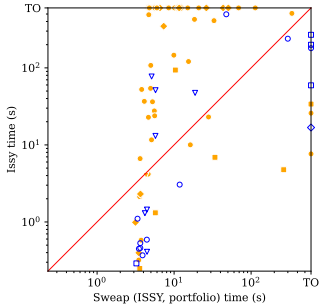
Evaluation of novel features

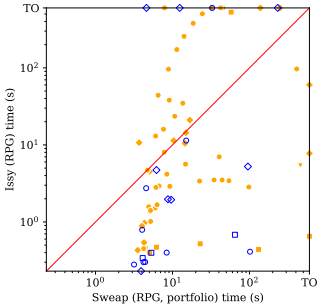
- `Sweap-strix`
  Uses Strix, no dualization
  (Roughly same as CAV'25 + bug fixes and optimisations)
- `Sweap (portfolio)`
  Uses SemML
  Best of normal + dual workflows
- 95 benchmarks (72 R, 23 U) in Sweap's native format
- 19 new verdicts (12 by dualization)
- Strix faster on smaller instances

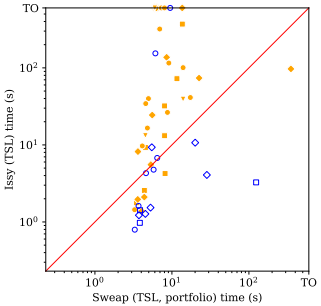


Evaluation vs Issy

Format	# Benchmarks	Tool	Realisable S	Realisable F	Realisable U	Unrealisable S	Unrealisable F	Unrealisable U
ISSY	115 (89 R, 26 U)	Issy	42	18	2	20	**14**	3
		Sweap	**70**	**53**	**40**	**21**	5	**4**
RPG	86 (66 R, 20 U)	Issy	52	**34**	2	16	**14**	0
		Sweap	**57**	25	**7**	**20**	6	**4**
TSL	65 (45 R, 20 U)	Issy	32	17	2	14	**10**	0
		Sweap	**38**	**21**	**8**	**17**	7	**3**







Conclusion

- CEGAR promising for ∞-state synthesis
- Dualization helps on unrealizable specifications
- Issy still (sometimes) faster, less memory hungry

Future work:

- Go beyond black-box finite synthesis
- Go beyond LIA
- Go beyond “monolithic” game arena

https://github.com/shaunazzopardi/sweap/

Funded by the European Union. The views and opinions expressed are those of the authors alone and do not necessarily reflect those of the European Union or the European Executive Agency for Health and Digital (HADEA). Neither the European Union nor the funding authority can be held liable for them. RobustifAI Project, ID 101212818