Strix

Explicit Reactive Synthesis Strikes Back!

Philipp J. Meyer Salomon Sickert Michael Luttenberger

CAV 2018
15th July

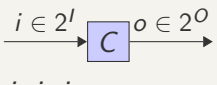
Technical University of Munich

Reactive Synthesis

Problem

Given: A specification by an LTL formula φ over $\Sigma = 2^{I \uplus O}$, where I are input and O are output signals.

Construct: A controller satisfying the specification, if one exists.



Goals of reactive synthesis

- *Fast* realizability check and *efficient* controller synthesis.
- Construction of *small* and *understandable* controllers.

Reactive Synthesis: current state of the art

Symbolic bounded synthesis

- ⊕ Produces small controllers.
- ⊕ Can deal efficiently with large alphabets.
- ⊖ Has problems with specifications requiring large controllers.

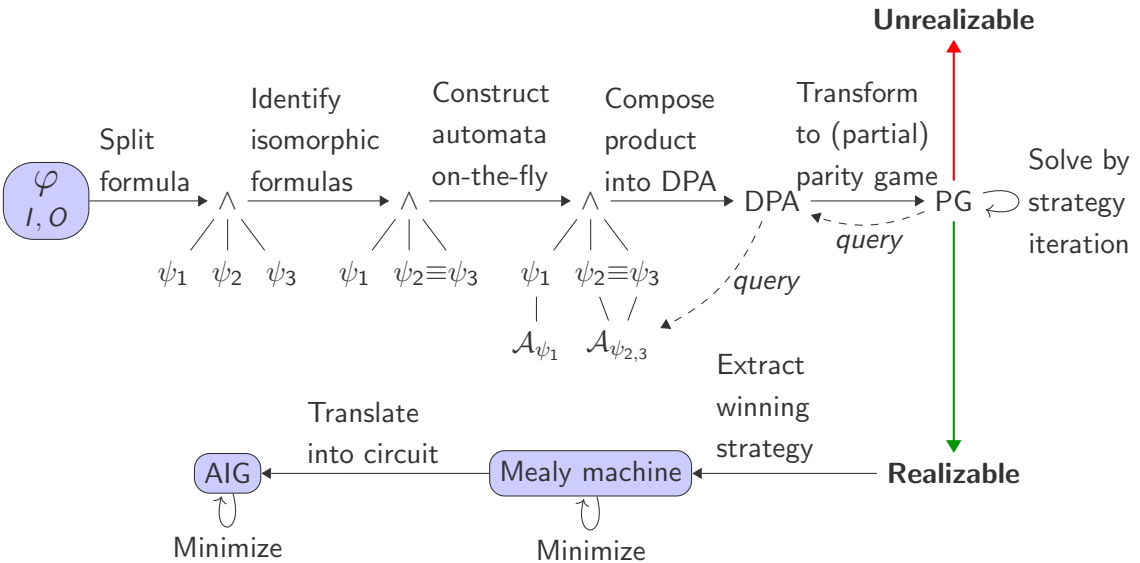
Explicit state synthesis

- ⊕ Can outperform bounded approaches.
- ⊖ May have memory issues.
- ⊖ Often produces unneccessarily large controllers.

Strix takes the **explicit state** approach, yet it

- has a *lighter memory* footprint,
- often produces *very small* controllers,
- *is efficient* for a large class of specifications.

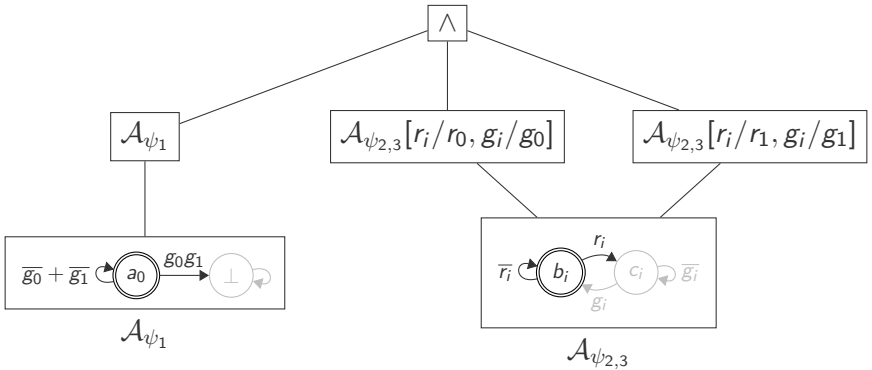
Overview of Strix



Example: a simple arbiter with two clients

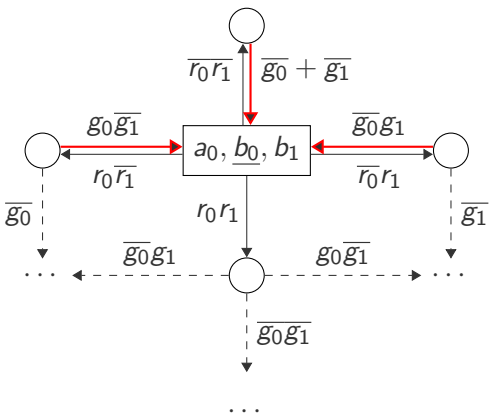
$I = \{r_0, r_1\}$ $\quad O = \{g_0, g_1\}$ $\quad \varphi = \underbrace{\mathbf{G}(\neg g_0 \vee \neg g_1)}_{\psi_1\text{: Safety}} \wedge \underbrace{\mathbf{G}(r_0 \rightarrow \mathbf{F} g_0)}_{\psi_2\text{: Büchi}} \wedge \underbrace{\mathbf{G}(r_1 \rightarrow \mathbf{F} g_1)}_{\psi_3\text{: Büchi}}$

$$\psi_2[r_0/r_i, g_0/g_i] \equiv \psi_3[r_1/r_i, g_1/g_i]$$



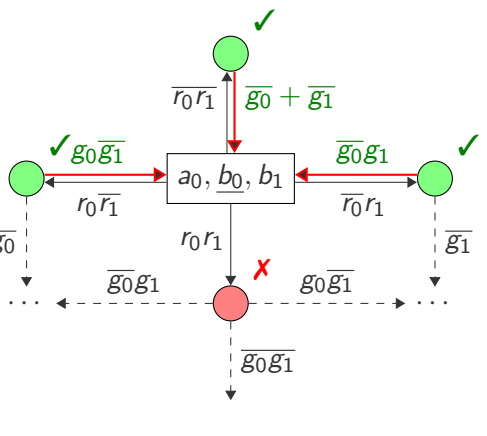
Parity game construction

- Construct the product DPA/parity game on-the-fly using adapted LAR.
- Solve it incrementally by strategy iteration with permissive strategies.



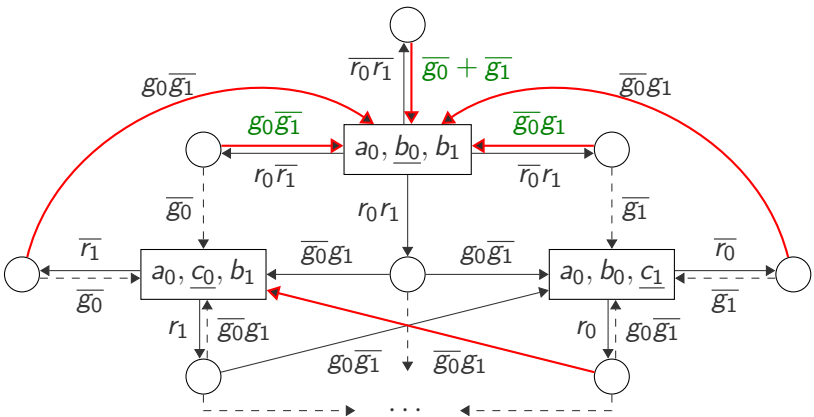
Parity game construction

- Construct the product DPA/parity game on-the-fly using adapted LAR.
- Solve it incrementally by strategy iteration with permissive strategies.



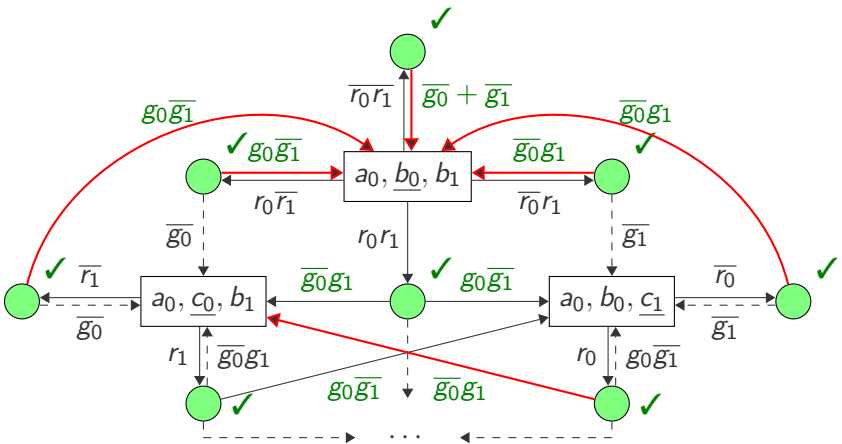
Parity game construction

- Construct the product DPA/parity game on-the-fly using adapted LAR.
- Solve it incrementally by strategy iteration with permissive strategies.



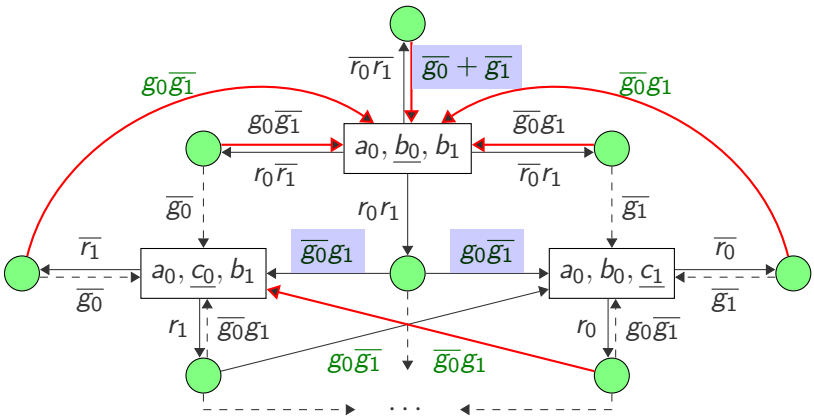
Parity game construction

- Construct the product DPA/parity game on-the-fly using adapted LAR.
- Solve it incrementally by strategy iteration with permissive strategies.



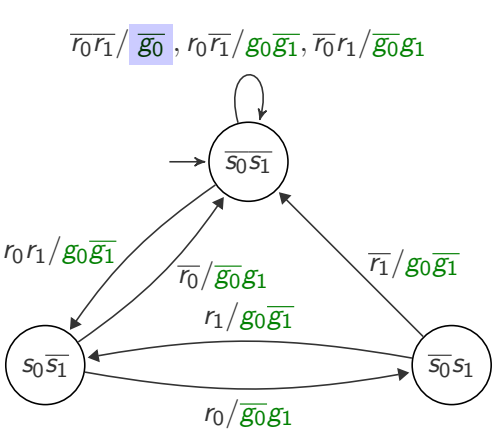
Parity game construction

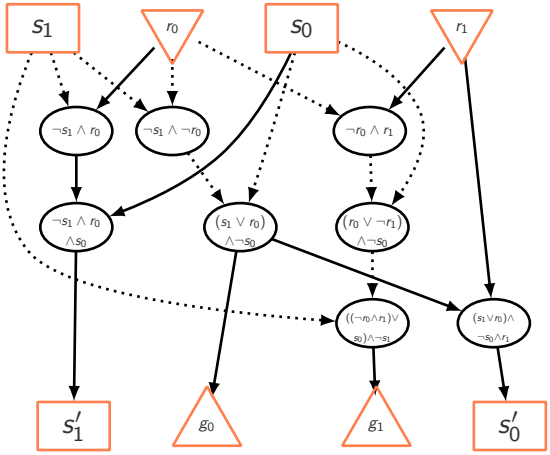
- Construct the product DPA/parity game on-the-fly using adapted LAR.
- Solve it incrementally by strategy iteration with permissive strategies.



Constructing the controller

- Mealy machine constructed from the winning strategy.
- And-Inverter Graph (AIG) generated from the Mealy machine using BDDs.





Minimization of the controller

- Minimize the Mealy machine using `MeMin`.
- Reduce the size of the AIG using `ABC`.
- Generate the AIG from:
  - unminimized Mealy machine.
  - minimized Mealy machine.
- Map states to latches by:
  - direct incremental mapping.
  - mapping the states of each product automaton to a separate set of latches.
- No combination is optimal for all specification.
- `Strix` tries all combinations and returns the smallest AIG.

Experimental evaluation

		Our system		SYNTCOMP2017		
		Strix (18.07)	ltlsynt (2.6)	Party	ltlsynt	BoSy
Solved	**Realizability**	**232**	208	224	195	181
	Synthesis	**227**	190	203	182	181
	Total Quality	**395**	189	308	180	298
	Average Quality	**1.74**	0.99	1.52	0.99	1.64
	prioritized_arbiter_7	65.22	91.36	372.95	TIME	TIME
	round_robin_arbiter_7	84.99	MEM	TIME	TIME	TIME
	ltl2dba_U1_8	1.29	5.97	872.104	TIME	TIME
	ltl2dba_U1_10	4.93	TIME	TIME	TIME	TIME
	amba_..._encode_12	77	3258	1040	3251	369
	full_arbiter_6	365	122744	7603	26678	TIME
	full_arbiter_7	577	608941	42492	MEM	TIME
	ltl2dba_E_6	11	3952	1955	3952	TIME
	ltl2dba_E_8	15	38454	TIME	TIME	TIME
	ltl2dba_Q_6	152	18276	TIME	15998	TIME
	ltl2dba_Q_8	292	398155	TIME	TIME	TIME

Experimental evaluation

`Strix` won all tracks of the LTL category for **SYNTCOMP 2018**:

- Realizability (sequential)
- Realizability (parallel)
- Synthesis (sequential)
- Synthesis (parallel)

Detailed results will be published at `http://www.syntcomp.org/`

Synthesis of complete AMBA specification

AMBA AHB: industrial bus protocol by ARM.

- A common case study for reactive synthesis.
- GR(1) or decompositional synthesis approaches existing.
- We use the *full unmodified* specification for $n = 2$ masters.

	`Strix` (18.07)	`ltlsynt` (2.6)
Time (seconds)	345	523
Size (Mealy Machine)	91	306
Size (AIG)	2829	93561

Conclusion

- `Strix` often outperforms bounded approaches.
  - Splitting into simple fragments and isomorphism detection.
  - Efficient and optimized automata and product arena construction.
  - Incremental solver that allows early termination and is easy to parallelize.
- `Strix` can produce even smaller controllers at the same time.
  - Permissive strategies allow further minimization.
  - Structure from splitting can be exploited by circuit generation.

Thank you!

Try `Strix` online: `https://strix.model.in.tum.de/`