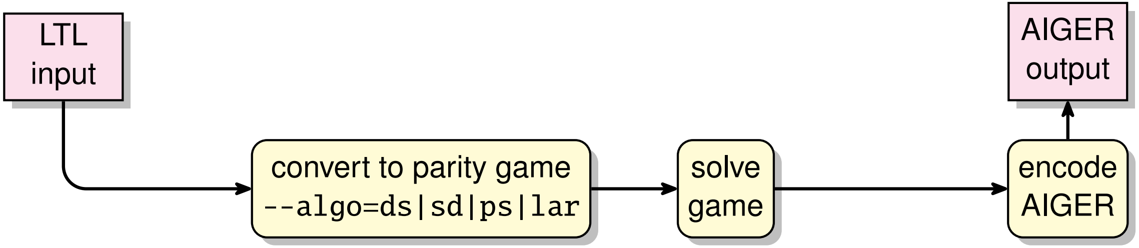
What's new in `ltlsynt`?

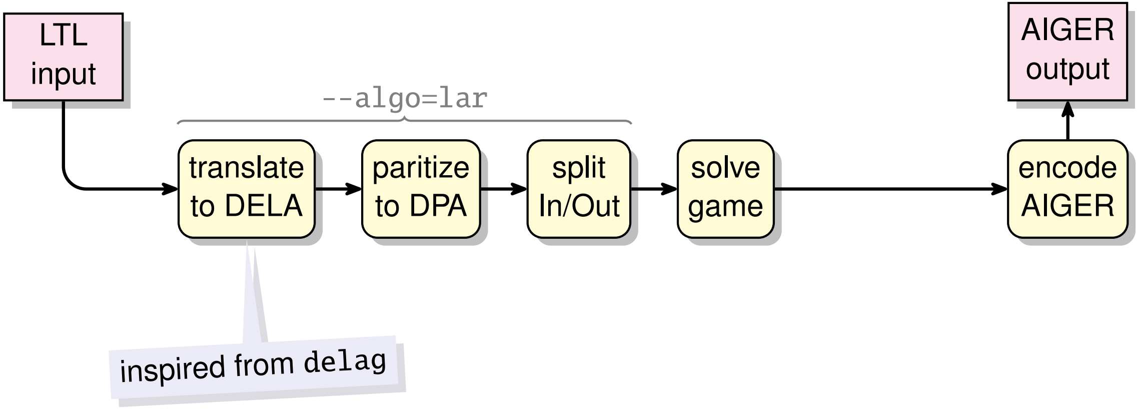
Florian Renkin, Philipp Schlehuber,
Alexandre Duret-Lutz, Adrien Pommellet

SYNT'21

Basic Workflow of `ltlsynt`

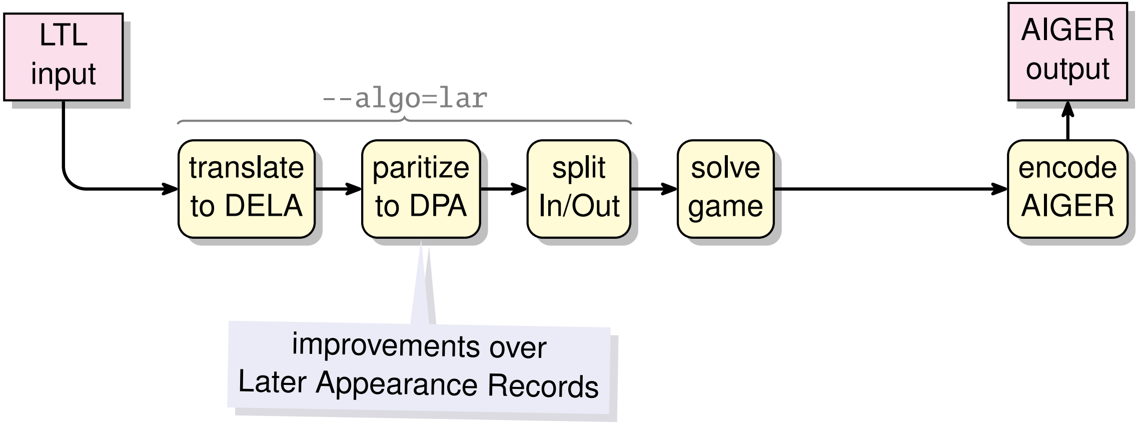


Basic Workflow of `ltlsynt`

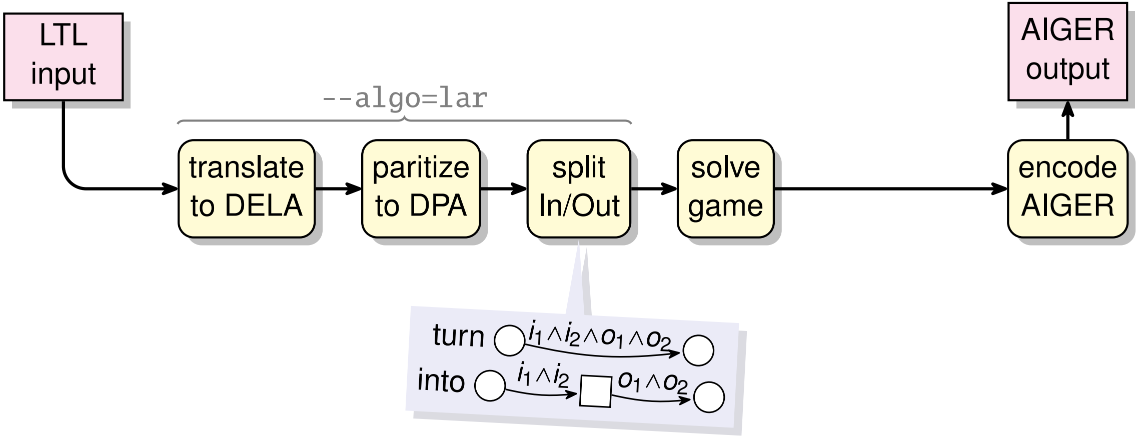


Müller and Sickert. LTL to deterministic Emerson-Lei automata. *GandALF'17*

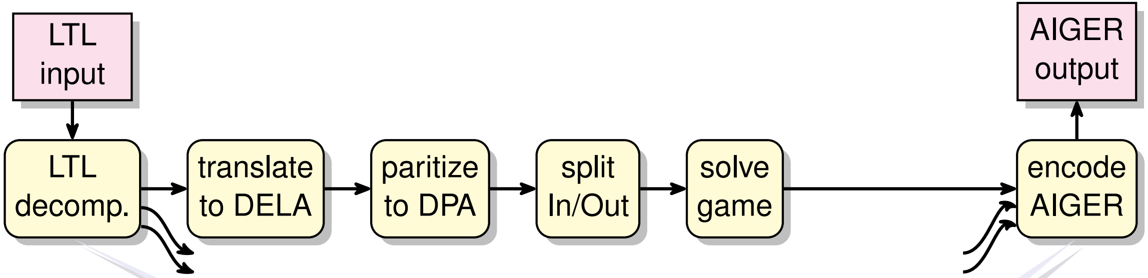
Basic Workflow of `ltlsynt`



Basic Workflow of `ltlsynt`



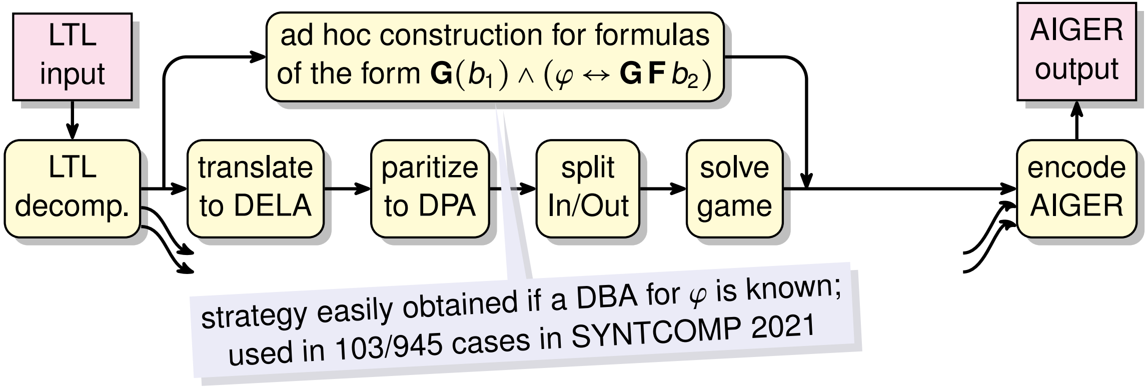
Basic Workflow of `ltlsynt`



separate specification as $\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \cdots$ where each φ_i has unique output variables; process each φ_i separately

merge strategies during encoding (possible sharing of gates above inputs)

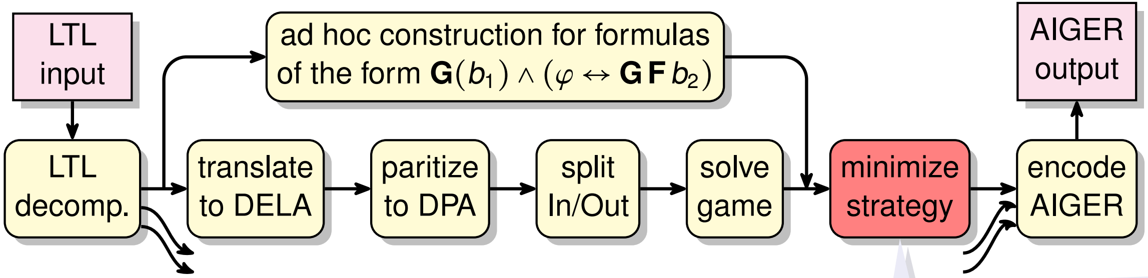
Basic Workflow of `ltlsynt`



LTL input
LTL decomp.
ad hoc construction for formulas of the form $\mathbf{G}(b_1) \wedge (\varphi \leftrightarrow \mathbf{G}\,\mathbf{F}\,b_2)$
translate to DELA
paritize to DPA
split In/Out
solve game
encode AIGER
AIGER output

strategy easily obtained if a DBA for φ is known;
used in 103/945 cases in SYNTCOMP 2021

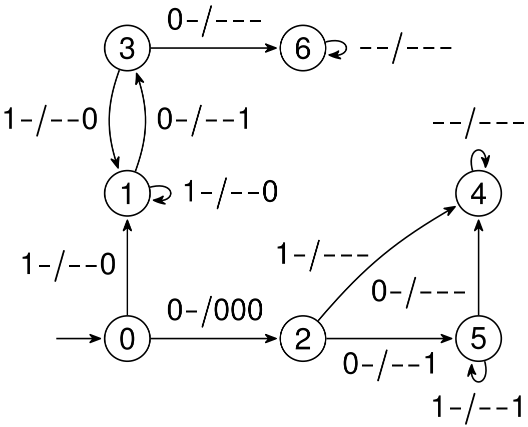
Strategy Minimization


LTL input
ad hoc construction for formulas of the form $\mathbf{G}(b_1) \wedge (\varphi \leftrightarrow \mathbf{G}\,\mathbf{F}\,b_2)$
AIGER output
LTL decomp.
translate to DELA
paritize to DPA
split In/Out
solve game
minimize strategy
encode AIGER

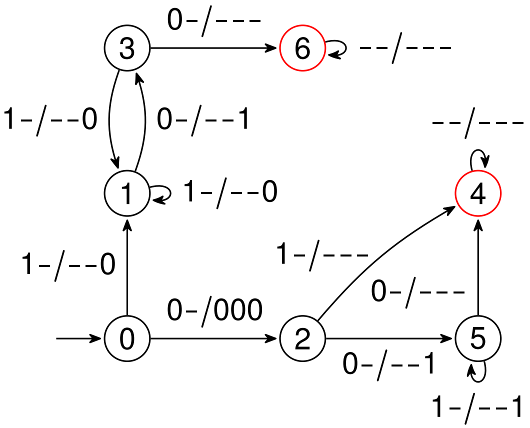
multiple methods:

1. bisimulation-based (fast, coarse)
2. bisimulation w/ output assignments
3. SAT-based (slow, precise)

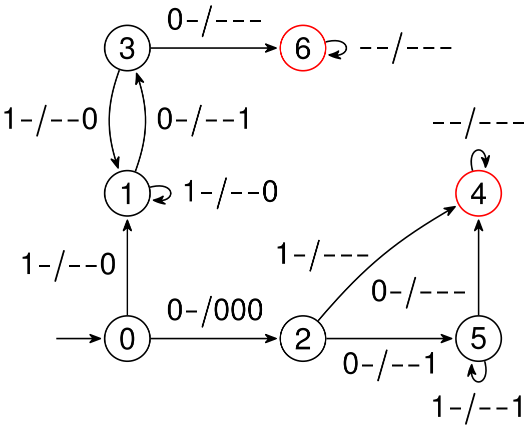
Minimizing Incompletely Specified Mealy Machines

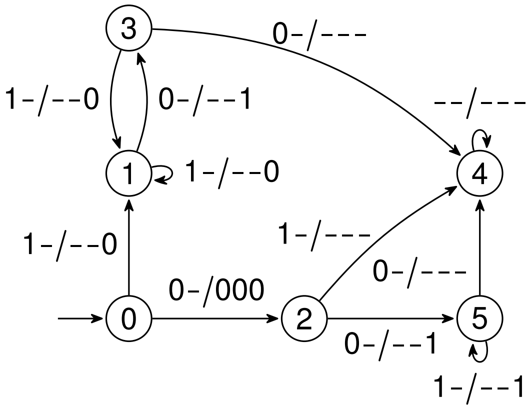


Minimizing ISMM with Bisimulation

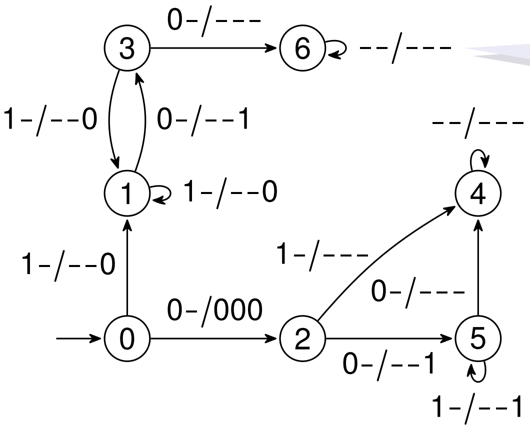


Minimizing ISMM with Bisimulation



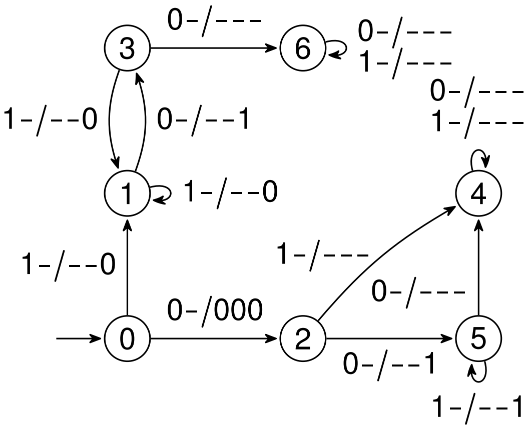


Minimizing ISMM with Bisimulation and Output Assignment

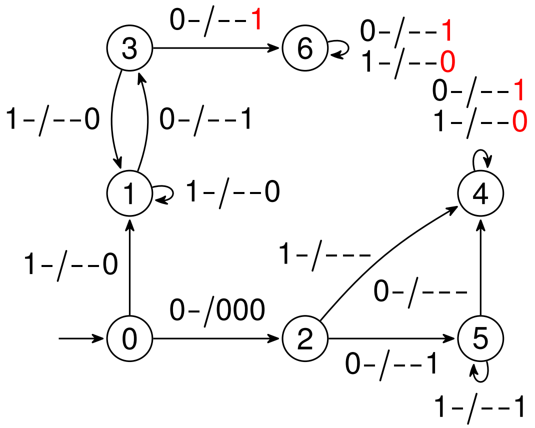


let's assign "don't care" outputs to improve bisimulation quotient

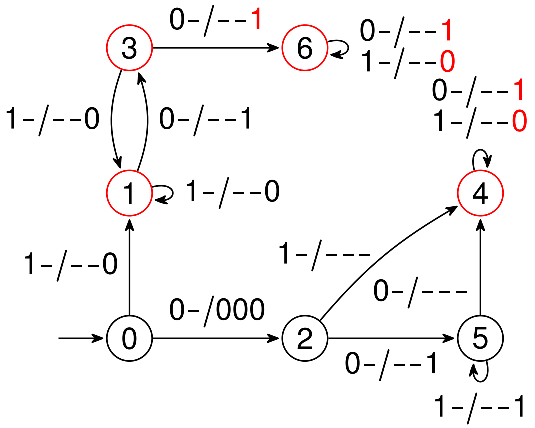
Minimizing ISMM with Bisimulation and Output Assignment

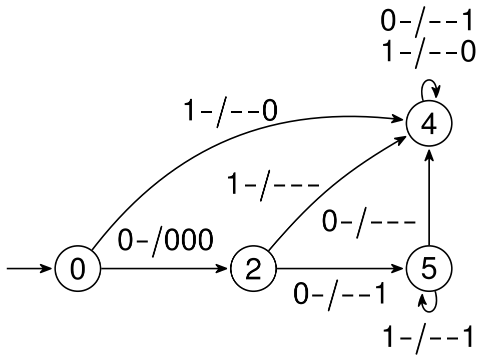


Minimizing ISMM with Bisimulation and Output Assignment

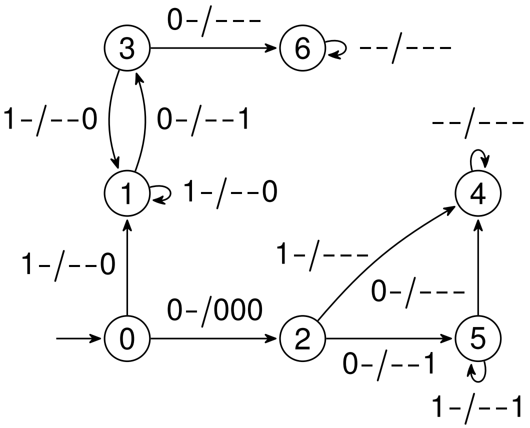


Minimizing ISMM with Bisimulation and Output Assignment



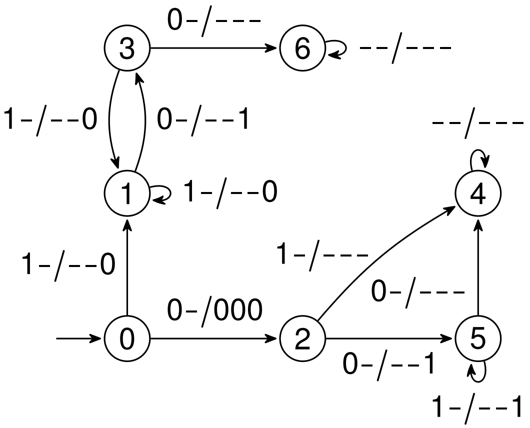


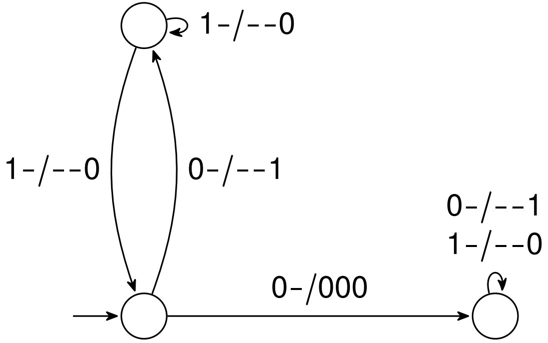
Minimizing ISMM with SAT (Similar to MeMin)



Abel and Reineke. MeMin: SAT-based exact minimization of incompletely specified Mealy machines. *ICCAD'15*

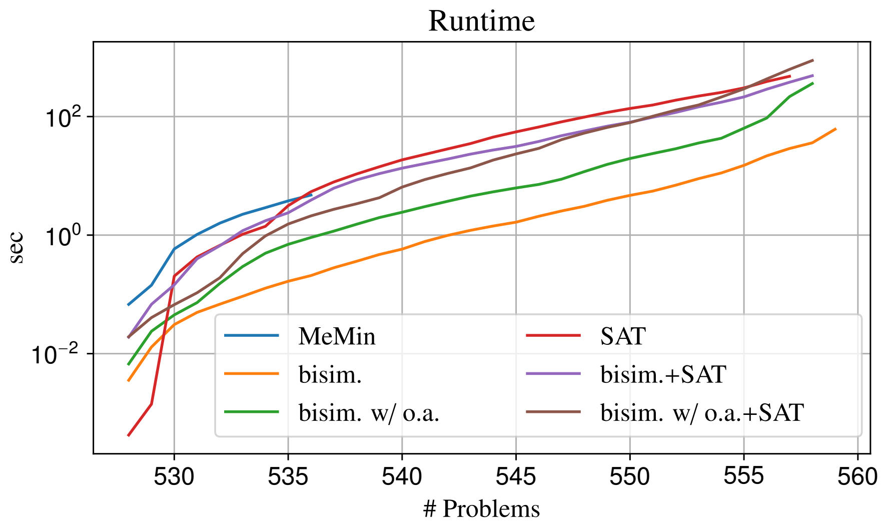
Minimizing ISMM with SAT (Similar to MeMin)





Abel and Reineke. MeMin: SAT-based exact minimization of incompletely specified Mealy machines. *ICCAD'15*

Benchmarks

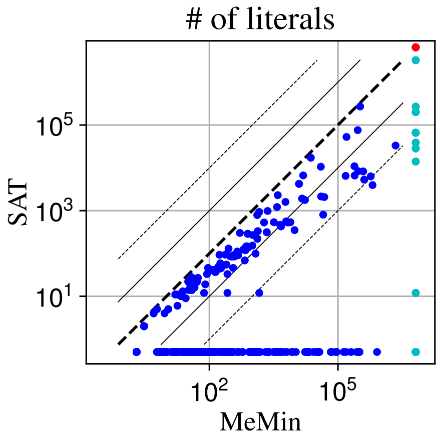

Runtime
sec
10^2
10^0
10^{-2}
530
535
540
545
550
555
560
# Problems
MeMin
bisim.
bisim. w/ o.a.
SAT
bisim.+SAT
bisim. w/ o.a.+SAT

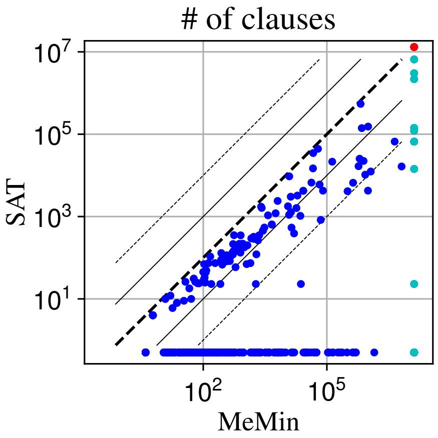
Benchmarks — cont'd

			size ratio of non-minimal cases	
method	**#solved**	**#minimal**	**mean**	**median**
no reduction	560	63%	9.05	1.44
bisimulation	560	76%	1.73	1.5
bisim. w/ output ass.	559	96%	1.39	1.39
SAT	557	100%		
bisimulation + SAT	558	100%		
bisim. w/ output ass. + SAT	559	97%	1.18	1.15
MeMin	536	100%		

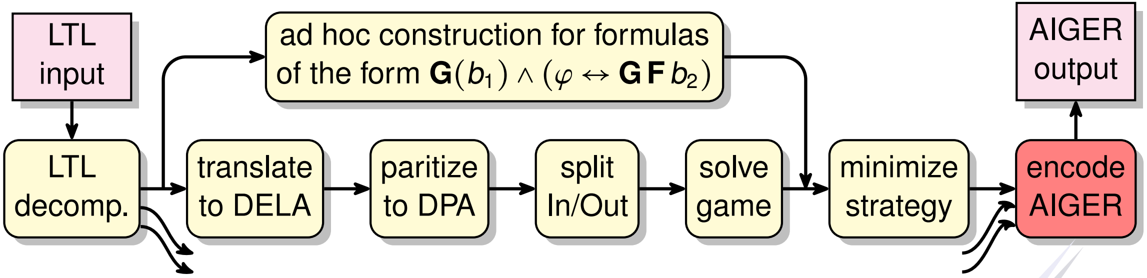
Differences to MeMin

- **Using BDDs instead of Cubes to label edges**
  More expressive edge labels e.g. 1–/ $\{10, 01\}$
- **Improved usage of a priori knowledge about the solution**





AIGER Encoding


LTL input
ad hoc construction for formulas of the form $\mathbf{G}(b_1) \land (\varphi \leftrightarrow \mathbf{G}\,\mathbf{F}\,b_2)$
AIGER output
LTL decomp.
translate to DELA
paritize to DPA
split In/Out
solve game
minimize strategy
encode AIGER

experimentation with multiple encodings

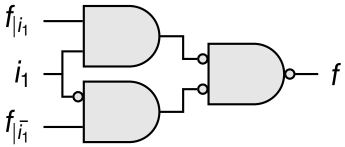
Testing Different BDD to Aiger Encodings

Conditions are represented by BDDs and translated into circuits

Testing Different BDD to Aiger Encodings

Conditions are represented by BDDs and translated into circuits

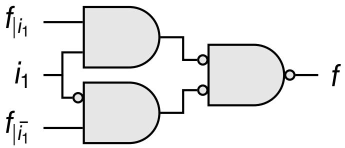
- If-then-else form (**ITE**):
  $f = (i_1 \wedge f_{|i_1}) \vee (\overline{i_1} \wedge f_{|\overline{i_1}})$



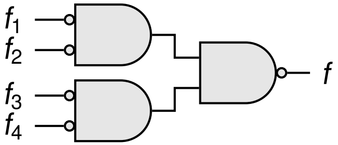
Testing Different BDD to Aiger Encodings

Conditions are represented by BDDs and translated into circuits

- If-then-else form (**ITE**):
  $f = (i_1 \wedge f_{|i_1}) \vee (\overline{i_1} \wedge f_{|\overline{i_1}})$



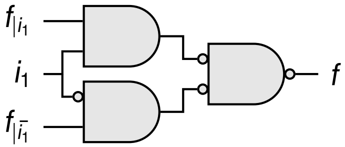
- Irredundant sum of products (**ISOP**):
  $f = f_1 \vee f_2 \vee f_3 \vee f_4$



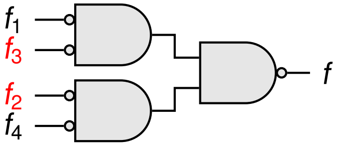
Testing Different BDD to Aiger Encodings

Conditions are represented by BDDs and translated into circuits

- If-then-else form (**ITE**):
  $f = (i_1 \wedge f_{|i_1}) \vee (\bar{i}_1 \wedge f_{|\bar{i}_1})$



- Irredundant sum of products (**ISOP**):
  $f = f_1 \vee f_2 \vee f_3 \vee f_4$

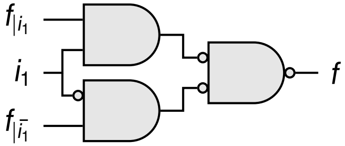


- If f_1 and f_3 appear frequently together in the strategy, reorder inputs to improve sharing (**OPTIM**)

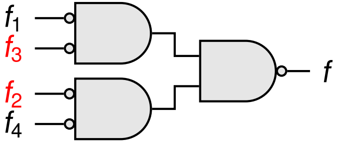
Testing Different BDD to Aiger Encodings

Conditions are represented by BDDs and translated into circuits

- If-then-else form (**ITE**):
  $f = (i_1 \wedge f_{|i_1}) \vee (\bar{i}_1 \wedge f_{|\bar{i}_1})$



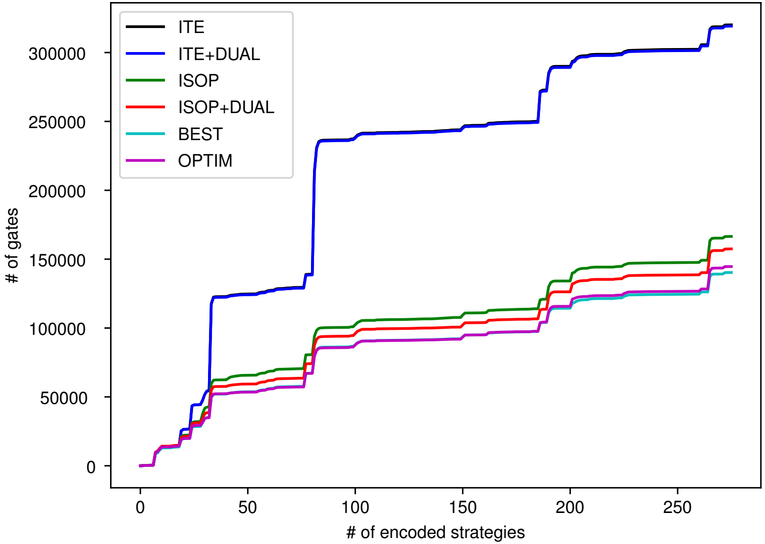
- Irredundant sum of products (**ISOP**):
  $f = f_1 \vee f_2 \vee f_3 \vee f_4$



- If f_1 and f_3 appear frequently together in the strategy, reorder inputs to improve sharing (**OPTIM**)

- Encode f and $\bar{f}$ and keep the smallest circuit (**DUAL**)

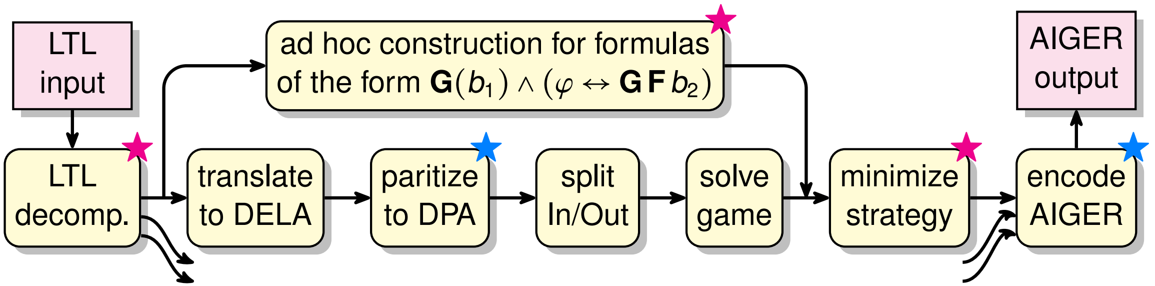
Benchmarking Aiger Encodings



BEST=
ITE+DUAL|ISOP+DUAL

total encoding time [s]	
ITE	1.0
ITE+DUAL	1.6
ISOP	2.7
ISOP+DUAL	5.1
BEST	6.2
OPTIM	4170

Conclusion


LTL input
ad hoc construction for formulas of the form $\mathbf{G}(b_1) \wedge (\varphi \leftrightarrow \mathbf{G}\,\mathbf{F}\,b_2)$ ★
AIGER output
LTL decomp. ★
translate to DELA
paritize to DPA ★
split In/Out
solve game
minimize strategy ★
encode AIGER ★

★: new in 2021
★: improved in 2021