Learning and Guessing Winning Policies in LTL Synthesis via Semantics

Jan Křetínský

Technical University of Munich ⇝ Masaryk University Brno

joint work with
Christian Backs, Alexander Manta, Tobias Meggendorfer, Maximilian Prokop, Sabine Rieder, and Askhan Zarhah

IARCS Verification Seminar
October 15, 2024

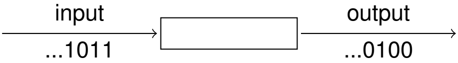
Alonzo Church, 1957

"Given a requirement which a circuit is to satisfy, we may suppose the requirement expressed in some suitable logistic system which is an extension of restricted recursive arithmetic. The synthesis problem is then to find recursion equivalences representing a circuit that satisfies the given requirement (or alternatively, to determine that there is no such circuit)."

Alonzo Church, 1957

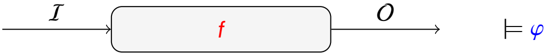
"Given a requirement which a circuit is to satisfy, we may suppose the requirement expressed in some suitable logistic system which is an extension of restricted recursive arithmetic. The synthesis problem is then to find recursion equivalences representing a circuit that satisfies the given requirement (or alternatively, to determine that there is no such circuit)."

Given a requirement on a bit stream transformation



fill the box by a machine with output, satisfying the requirement (or state that the requirement is not satisfiable).

Synthesis problem formally

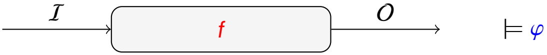


Given a specification φ and
atomic propositons partitioned into I(nput) and O(utput),
synthesize a (finite circuit) *f*:

$$\forall \text{ input stream } \mathcal{I} : \qquad \mathcal{I} \parallel \mathcal{O} \models \varphi$$

where

Synthesis problem formally

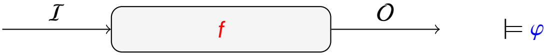


Given a specification φ and
atomic propositons partitioned into I(nput) and O(utput),
synthesize a (finite circuit) *f*:

$$\forall \text{ input stream } \mathcal{I} : \qquad \mathcal{I} \parallel \mathcal{O} \models \varphi$$

where

Synthesis problem formally



Given a specification φ and
atomic propositons partitioned into I(nput) and O(utput),
synthesize a (finite circuit) f:

$$\forall \text{ input stream } \mathcal{I} : \qquad \mathcal{I} \parallel \mathcal{O} \models \varphi$$

where

$\mathcal{I} = i_1 i_2 i_3 \cdots$ with $i_n \subseteq \mathcal{I}$

$\mathcal{O} = o_1 o_2 o_3 \cdots = f(i_1) f(i_1 i_2) f(i_1 i_2 i_3) \cdots$ with $o_n \subseteq \mathcal{O}$

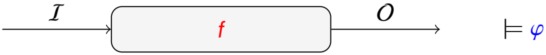
$\mathcal{I} \parallel \mathcal{O} = i_1 \cup o_1 \quad i_2 \cup o_2 \quad i_3 \cup o_3 \cdots$

$$\varphi ::= a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{F}\varphi \mid \mathbf{G}\varphi \mid \varphi \mathbf{U} \varphi$$

Example: $I = \{a\}, O = \{b\}, \varphi = \mathbf{G}(a \text{ xor } b)$

	ℓ_1	ℓ_2	$\cdots$
$\mathcal{I}$	$\{a\}$	$\emptyset$	
$\mathcal{O}$	$\emptyset$	$\{b\}$	
$\mathcal{I} \parallel \mathcal{O}$	$\{a\}$	$\{b\}$	

Synthesis problem formally



Given a specification φ and
atomic propositons partitioned into I(nput) and O(utput),
synthesize a (finite circuit) f:

$$\forall \text{ input stream } \mathcal{I} : \qquad \mathcal{I} \parallel \mathcal{O} \models \varphi$$

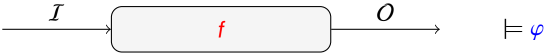
where
$\mathcal{I} = i_1 i_2 i_3 \cdots \quad$ with $i_n \subseteq \mathcal{I}$
$\mathcal{O} = o_1 o_2 o_3 \cdots = f(i_1) f(i_1 i_2) f(i_1 i_2 i_3) \cdots \quad$ with $o_n \subseteq \mathcal{O}$
$\mathcal{I} \parallel \mathcal{O} = i_1 \cup o_1 \quad i_2 \cup o_2 \quad i_3 \cup o_3 \cdots$

$$\varphi ::= a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{F}\varphi \mid \mathbf{G}\varphi \mid \varphi \mathbf{U} \varphi$$

Example: $I = \{r\}, O = \{g\}, \varphi = \mathbf{G}(r \Rightarrow \mathbf{F}g)$

	ℓ_1	ℓ_2	$\cdots$
$\mathcal{I}$	$\{r\}$	$\{r\}$	
$\mathcal{O}$	$\emptyset$	$\{g\}$	
$\mathcal{I} \parallel \mathcal{O}$	$\{r\}$	$\{r, g\}$	

Synthesis problem formally



Given a specification φ and
atomic propositons partitioned into I(nput) and O(utput),
synthesize a (finite circuit) f:

$$\forall \text{ input stream } \mathcal{I}: \qquad \mathcal{I} \parallel \mathcal{O} \models \varphi$$

where

$\mathcal{I} = i_1 i_2 i_3 \cdots \quad$ with $i_n \subseteq \mathcal{I}$

$\mathcal{O} = o_1 o_2 o_3 \cdots = f(i_1) f(i_1 i_2) f(i_1 i_2 i_3) \cdots \quad$ with $o_n \subseteq \mathcal{O}$

$\mathcal{I} \parallel \mathcal{O} = i_1 \cup o_1 \quad i_2 \cup o_2 \quad i_3 \cup o_3 \cdots$

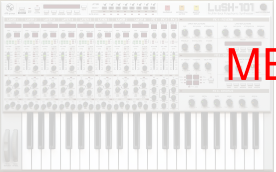
$$\varphi ::= a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{F}\varphi \mid \mathbf{G}\varphi \mid \varphi \mathbf{U} \varphi$$

Example:

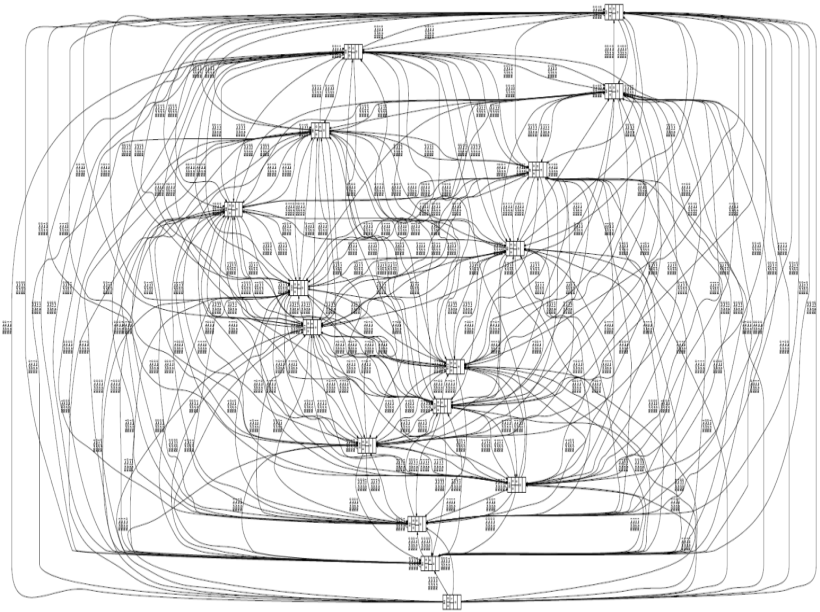
	ℓ_1	ℓ_2	$\cdots$
$\mathcal{I}$			
$\mathcal{O}$			
$\mathcal{I} \parallel \mathcal{O}$			

Church, Ramadge&Wonham, Pnueli&Rosner,...

Formal methods and machine learning



MEM-OUT



Formal methods and machine learning

Formal methods: precise synthesis

+ optimality of results
– scalability issues

+ precise outputs (guaranteed safety)
– precise inputs (known models)

+ specification-oriented
– ad hoc algorithms

Learning: insightful guesses

– weaker guarantees
+ scalable

– imprecise outputs
+ imprecise inputs

– simple objectives only
+ problem-independent

Formal methods and machine learning

Formal methods: precise synthesis

- \+ optimality of results
- – scalability issues

- \+ precise outputs (guaranteed safety)
- – precise inputs (known models)

- \+ specification-oriented
- – ad hoc algorithms

Learning: insightful guesses

- – weaker guarantees
- \+ scalable

- – imprecise outputs
- \+ imprecise inputs

- – simple objectives only
- \+ problem-independent

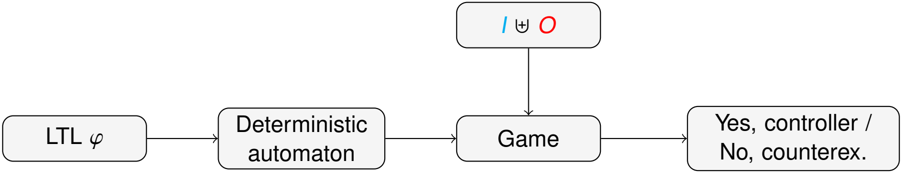
precise computation

focus on important stuff

Classic solution

LTL synthesis:

$\{r\} \uplus \{g\}$

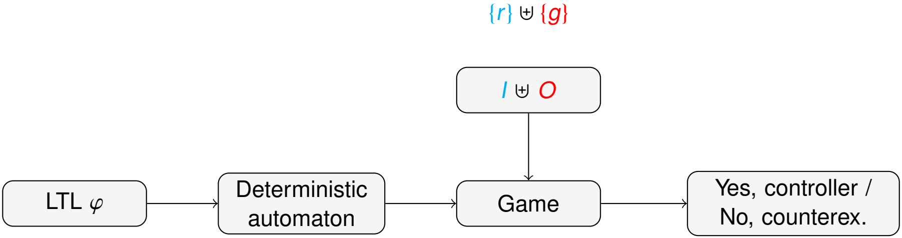


$\mathbf{G}(r \Rightarrow \mathbf{F}g)$

"every r is followed by g"

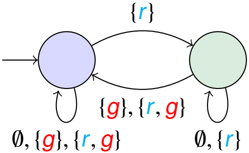
Classic solution

LTL synthesis:



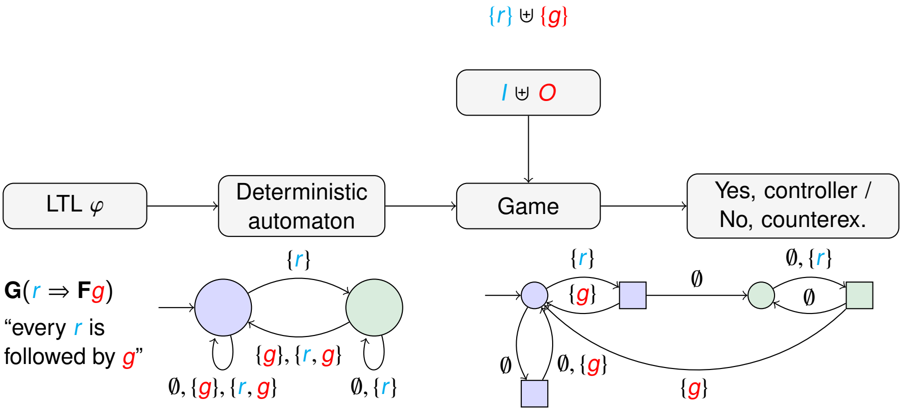
$\mathbf{G}(r \Rightarrow \mathbf{F}g)$

"every r is followed by g"



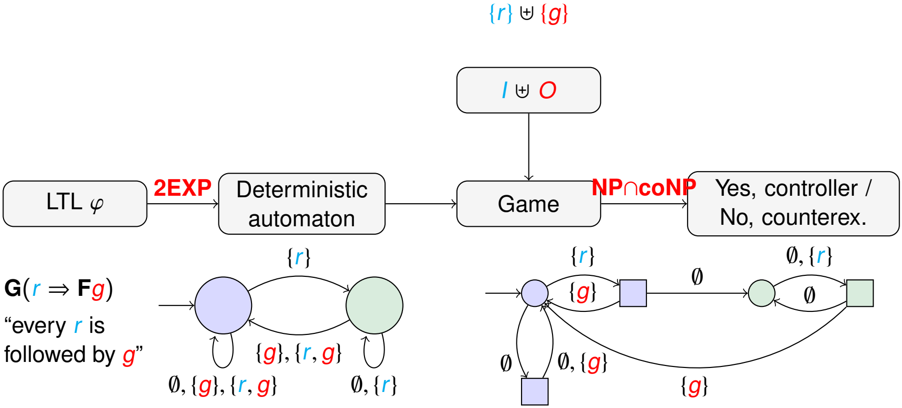
Classic solution

LTL synthesis:



Classic solution

LTL synthesis:



Fragments:

- R. Alur, S. La Torre: Deterministic generators and games for LTL fragments. ACM ToCL 2004
- N. Piterman, A. Pnueli, Y. Sa'ar: Synthesis of Reactive(1) Designs. VMCAI 2006

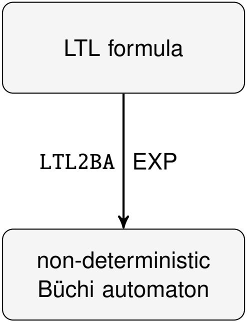
Safraless:

- O. Kupferman, N. Piterman, M. Y. Vardi: Safraless Compositional Synthesis. CAV 2006
- B. Jobstmann, R. Bloem: Optimizations for LTL Synthesis. FMCAD 2006

Bounded synthesis:

- S. Schewe, B. Finkbeiner: Bounded Synthesis. ATVA 2007
- R. Ehlers: Unbeast: Symbolic Bounded Synthesis. TACAS 2011
- A. Bohy, V. Bruyère, E. Filiot, N. Jin, J.-F. Raskin: Acacia+, a Tool for LTL Synthesis. CAV 2012
- P. Faymonville, B. Finkbeiner, L. Tentrup: BoSy: An Experimentation Framework for Bounded Synthesis. CAV 2017

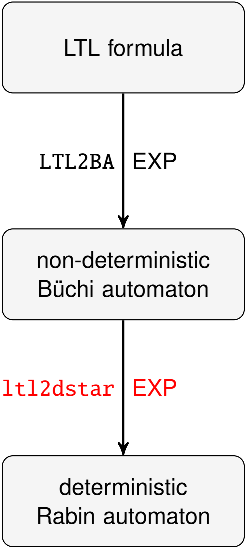
New theory needed: LTL→automata



$\bigwedge_{i \in \{1,\ldots,n\}} \mathbf{GF} a_i \Rightarrow \mathbf{GF} b_i$	NBA LTL2BA
$n = 1$	4
$n = 2$	14
$n = 3$	40

A. Gaiser

New theory needed: LTL→automata

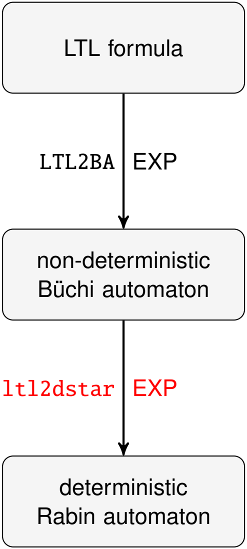


$\bigwedge_{i \in \{1,\dots,n\}} \mathbf{GF} a_i \Rightarrow \mathbf{GF} b_i$	NBA LTL2BA
$n = 1$	4
$n = 2$	14
$n = 3$	40

S. Safra

determinization [Safra'88,. . .]

New theory needed: LTL→automata



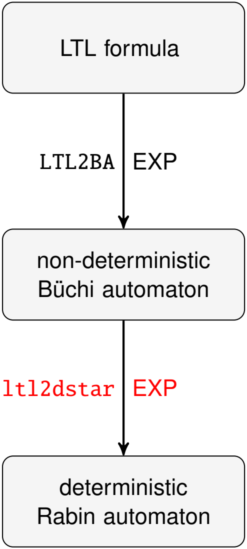
$\bigwedge_{i\in\{1,\dots,n\}} \mathbf{GF}a_i \Rightarrow \mathbf{GF}b_i$	NBA `LTL2BA`	DRA `ltl2dstar`
$n = 1$	4	4
$n = 2$	14	$> 10^4$
$n = 3$	40	$> 10^6$

S. Safra

determinization [Safra'88,. . .]

- inefficient

New theory needed: LTL→automata



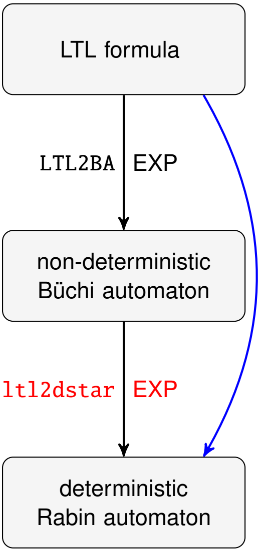
$\bigwedge_{i\in\{1,\dots,n\}} \mathbf{GF}a_i \Rightarrow \mathbf{GF}b_i$	NBA LTL2BA	DRA ltl2dstar
$n = 1$	4	4
$n = 2$	14	$> 10^4$
$n = 3$	40	$> 10^6$


O. Kupferman

determinization [Safra'88,. . .]

- inefficient
- "messy" [Kupferman'12]

New theory needed: LTL→automata



$\bigwedge_{i \in \{1,\ldots,n\}} \mathbf{GF} a_i \Rightarrow \mathbf{GF} b_i$	NBA LTL2BA	DRA ltl2dstar	DRA Rabinizer
$n = 1$	4	4	4
$n = 2$	14	$> 10^4$	18
$n = 3$	40	$> 10^6$	462

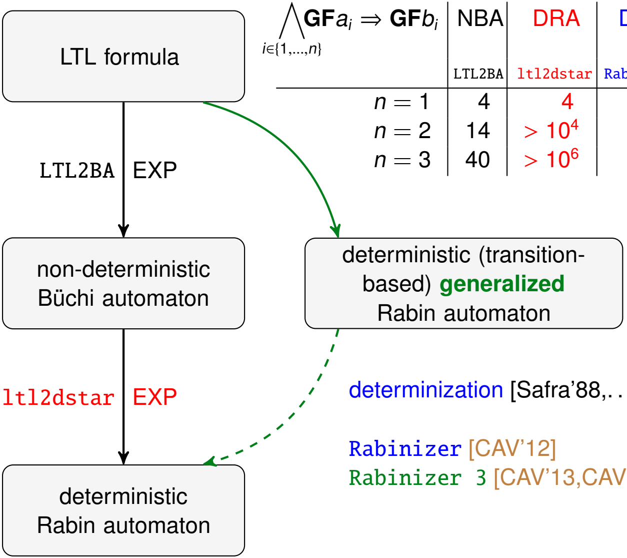
determinization [Safra'88, . . .]→large and messy

Rabinizer [CAV'12]

- smaller automata
- logical structure of states

New theory needed: LTL→automata

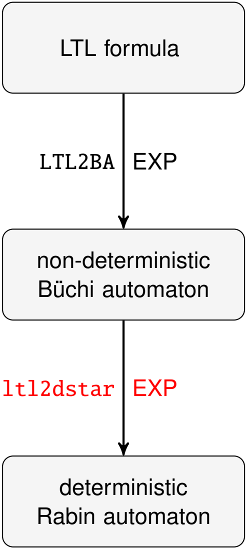
$\bigwedge_{i\in\{1,\dots,n\}} \mathbf{GF}a_i \Rightarrow \mathbf{GF}b_i$	NBA LTL2BA	DRA ltl2dstar	DRA Rabinizer	DTGRA Rabinizer 3
$n = 1$	4	4	4	1
$n = 2$	14	$> 10^4$	18	1
$n = 3$	40	$> 10^6$	462	1



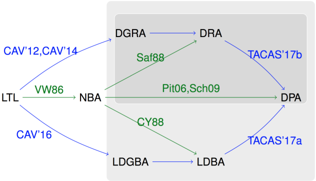
determinization [Safra'88,. . .]→large and messy

Rabinizer [CAV'12]
Rabinizer 3 [CAV'13,CAV'14]

New theory needed: LTL→automata



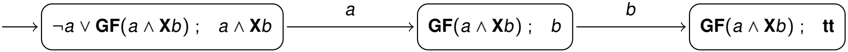
$\bigwedge_{i\in\{1,\dots,n\}} \mathbf{GF}a_i \Rightarrow \mathbf{GF}b_i$	NBA	DRA	DRA	DTGRA
	LTL2BA	ltl2dstar	Rabinizer	Rabinizer 3
$n = 1$	4	4	4	1
$n = 2$	14	$> 10^4$	18	1
$n = 3$	40	$> 10^6$	462	1

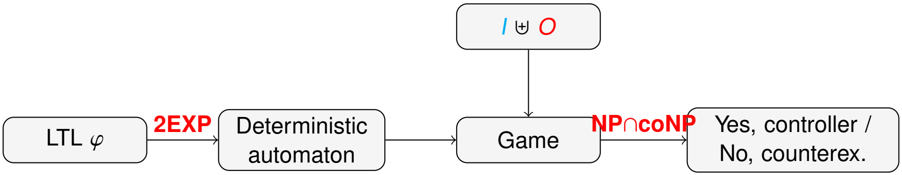


determinization [Safra'88,. . .]→large and messy

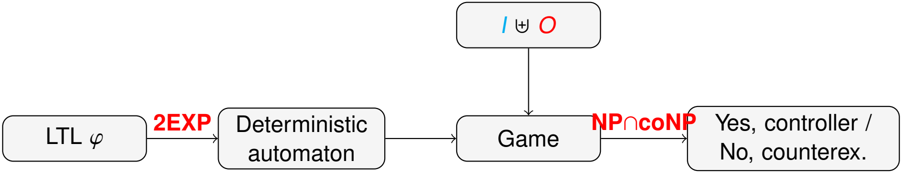
Rabinizer [CAV'12]
Rabinizer 3 [CAV'13,CAV'14]
... [CAV'16,TACAS'17a,TACAS'17b,CAV'18]
... [LICS'18, Journal of ACM'20]
→ **small and with logical structure**

Semantic labelling: Example

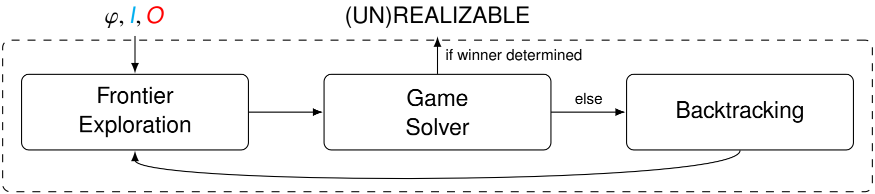



$I \uplus O$
LTL φ
2EXP
Deterministic automaton
Game
NP∩coNP
Yes, controller / No, counterex.

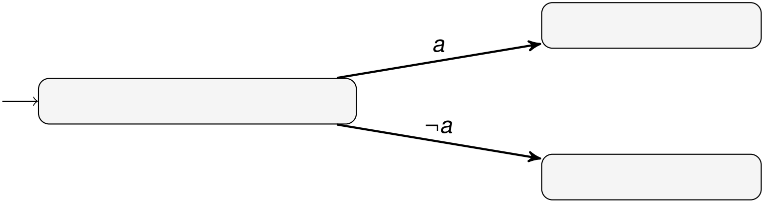
New solutions with new translations



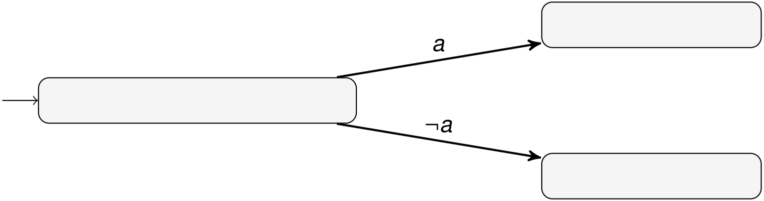
On-the-fly approaches:



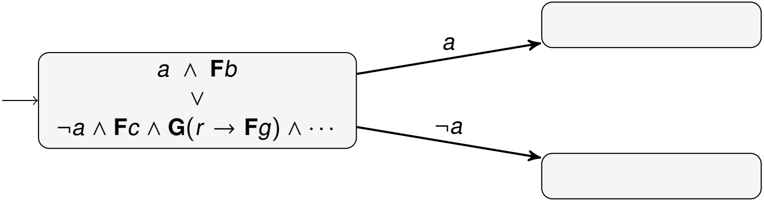
Which one to choose?



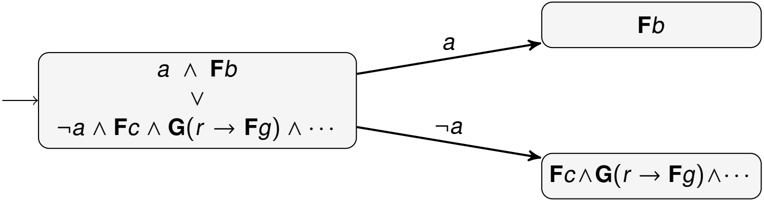
Which one to choose?



Which one to choose?

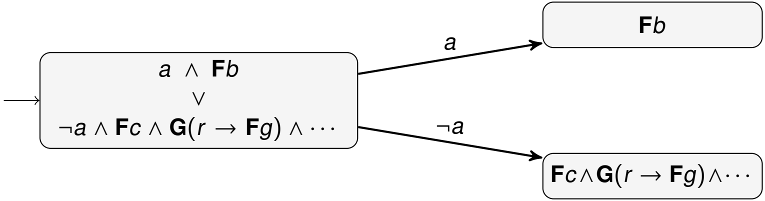


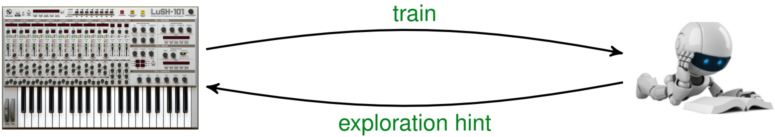
Which one to choose?



New solutions with new translations

Which one to choose?





Semantic labelling

- similar to NBA
- obscured by Safra
- recovered by direct approaches

$\Longrightarrow$ learning what to do

Naive heuristic: Trueness

Which formula is easier to satisfy?

Naive heuristic: Trueness

Which formula is easier to satisfy?

Naive approximation:

1. see the formula as Boolean, e.g. $a \wedge \mathbf{GF}a$ as $A \wedge B$
2. compute the probability of satisfaction by a random assignment

$$\frac{\#\text{ satisfying assignments}}{\#\text{ all assignments}}$$

(easy for BDDs)

Naive heuristic: Trueness

Which formula is easier to satisfy?

Naive approximation:

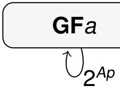
1. see the formula as Boolean, e.g. $a \wedge \mathbf{GF}a$ as $A \wedge B$
2. compute the probability of satisfaction by a random assignment

$$\frac{\text{\# satisfying assignments}}{\text{\# all assignments}}$$

   (easy for BDDs)

We ignore

- ownership of propositions
- modal structure
- subgoals



Naive heuristic: Trueness

Which formula is easier to satisfy?

Naive approximation:

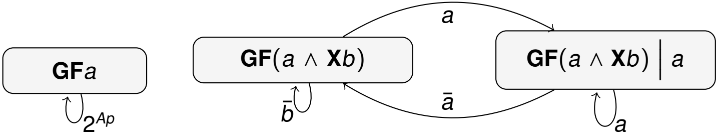
1. see the formula as Boolean, e.g. $a \wedge \mathbf{GF}a$ as $A \wedge B$
2. compute the probability of satisfaction by a random assignment

$$\frac{\text{\# satisfying assignments}}{\text{\# all assignments}}$$

(easy for BDDs)

We ignore

- ownership of propositions
- modal structure
- subgoals



Trueness in action

Immediately solved games by initialization[1]

	Random	Trueness
(Co-)Safety	32%	65%
Near (Co-)Safety	11%	67%
Parity	10%	56%

Strix [MSL18][2]

- winning LTL tracks in SyntComp 2018–2023
- by new translations + on-the-fly strategy iteration
- trueness-aided exploration + classic algorithm for parity games

[1] J. K., A. Manta, T. Meggendorfer: Semantic Labelling and Learning for Parity Game Solving in LTL Synthesis. ATVA 2019

[2] P. Meyer, S. Sickert, M. Luttenberger: Strix: Explicit Reactive Synthesis Strikes Back! CAV 2018

Learning the selector heuristic

Involve machine learning[3][4]

1. Consider further features
2. Learn on winning/losing transitions from previously solved games

[3] J. K., A. Manta, T. Meggendorfer: Semantic Labelling and Learning for Parity Game Solving in LTL Synthesis. ATVA 2019

[4] J. K., T. Meggendorfer, M. Prokop, S. Rieder: Guessing Winning Policies in LTL Synthesis by Semantic Learning. CAV 2023

Involve machine learning[3][4]

1. Consider further features
2. Learn on winning/losing transitions from previously solved games

Some aspects:

- Models: SVM, DT, NN, GNN,. . .

[3] J. K., A. Manta, T. Meggendorfer: Semantic Labelling and Learning for Parity Game Solving in LTL Synthesis. ATVA 2019

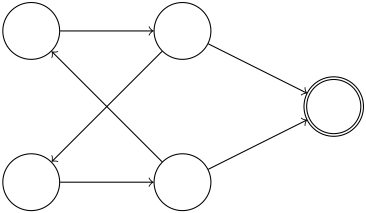
[4] J. K., T. Meggendorfer, M. Prokop, S. Rieder: Guessing Winning Policies in LTL Synthesis by Semantic Learning. CAV 2023

Involve machine learning[3][4]

1. Consider further features
2. Learn on winning/losing transitions from previously solved games

Some aspects:

- Models: SVM, DT, NN, GNN,...
- Ground truth: Beyond safety no maximally permissive strategies!



[3] J. K., A. Manta, T. Meggendorfer: Semantic Labelling and Learning for Parity Game Solving in LTL Synthesis. ATVA 2019

[4] J. K., T. Meggendorfer, M. Prokop, S. Rieder: Guessing Winning Policies in LTL Synthesis by Semantic Learning. CAV 2023

SVM in action

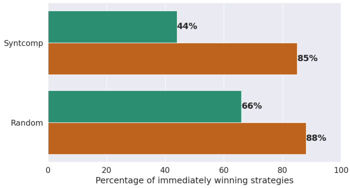
Games where the a-priori learnt strategy is already winning [ATVA'19]:

category	random	trueness	svm
small	20.00%	60.00%	**93.33%**
small safety	16.67%	**100.00%**	**100.00%**
small p-safety	16.67%	75.00%	**100.00%**
small co-safety	41.67%	**100.00%**	**100.00%**
small p-co-safety	33.33%	83.33%	**100.00%**
large	6.67%	53.33%	**93.33%**
large safety	0.00%	**100.00%**	**100.00%**
large co-safety	23.08%	**100.00%**	**100.00%**
lily	11.11%	44.44%	**55.56%**
ltl2dba	0.00%	0.00%	**62.50%**
ltl2dpa	0.00%	0.00%	**54.55%**
total	15.91%	68.18%	**89.39%**

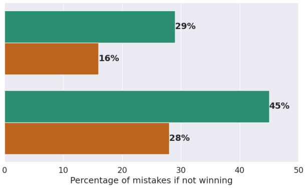
SVM in action

Trueness – How easy is it to satisfy the formula (in one step)?
Train SVM with SYNTCOMP data and manual features

Predicted strategy winning?



How many changes by SI, if not winning?



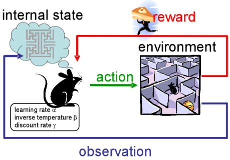
Summary: Circumventing Safra and PG-complexity

Translation LTL→automata:

- 2EXP (via Safra's determinization)
- new, direct approaches
  - practically more efficient
  - preserving semantic information

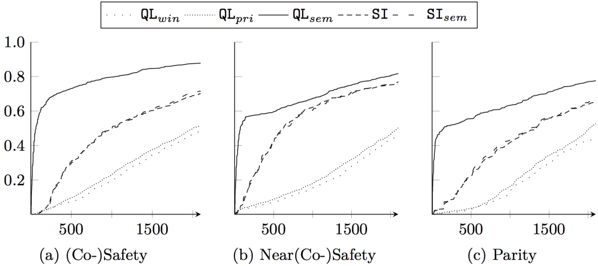
Solving parity games

- NP∩coNP
- heuristics
  - initialization
  - on the fly

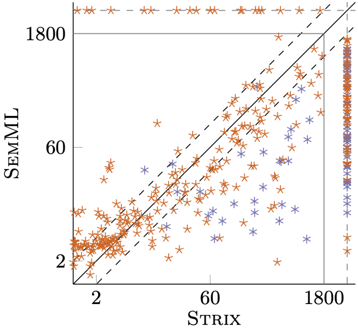


Solved games by number of evaluation steps

- QL_{win}: Q-learning with only win/loss as reward signal
- QL_{pri}: Q-learning with priority-based rewards
- QL_{sem}: Q-learning with semantic rewards



On the fly synthesis: The SemML tool



SYNTCOMP		SemML solved	SemML unsolved
Strix	solved	951	27
Strix	unsolved	49	84

Speed-ups:

SYNTCOMP	0	5	30	300
ratio	0.09	1.37	2.08	3.58
count	951	148	89	30

Synthetic	0	5	30	300
ratio	8.56	8.56	9.48	13.44
count	30	30	28	14

Conclusions

- semantic information $\Longrightarrow$ apply learning
- SemML winning SyntComp
- not easy to learn well (complex discrete structure)
- lots of future work on (i) learning and (ii) theory

Conclusions

- semantic information $\Longrightarrow$ apply learning
- SemML winning SyntComp
- not easy to learn well (complex discrete structure)
- lots of future work on (i) learning and (ii) theory

Thank you!

General need for a “soft” distance between formulae

Proximity and relationship between formulae beyond logical implication
$\Longrightarrow$ embedding of formulae into spaces where learning can be done well[5][6]

Further applications:

“Learning” model checking Determine (a probability of) satisfaction of a formula based on satisfaction of other (unrelated) formulae.

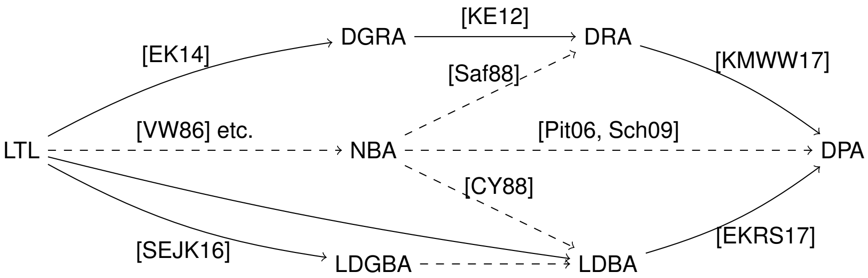
Translating, sanitizing and simplifying specifications Find the closest simple formula to the inadequate translation from English to logic.

Requirement mining Lifting the search problem from the discrete combinatorial space of syntactic structures of formulae to a continuous space in which distances preserve semantic similarity.

[5] L. Bortolussi, G. M. Gallo, J.K., L. Nenzi: Learning Model Checking and the Kernel Trick for Signal Temporal Logic on Stochastic Processes. TACAS 2022

[6] G. Saveri, L. Nenzi, L. Bortolussi, J.K.: stl2vec: Semantic and Interpretable Vector Representation of Temporal Logic. ECAI 2024

LTL to automata



$\bigwedge_{i\in\{1,\dots,n\}} \mathbf{GF}a_i \Rightarrow \mathbf{GF}b_i$	NBA LTL2BA	DRA ltl2dstar	DRA Rabinizer	DTGRA Rabinizer 3
$n = 1$	4	4	4	1
$n = 2$	14	$> 10^4$	18	1
$n = 3$	40	$> 10^6$	462	1

Costas Courcoubetis and Mihalis Yannakakis.
Verifying temporal properties of finite-state probabilistic programs.
In *FOCS*, pages 338–345, 1988.

Javier Esparza and Jan Křetínský.
From LTL to deterministic automata: A Safraless compositional approach.
In *CAV*, pages 192–208, 2014.

Javier Esparza, Jan Křetínský, Jean-Francois Raskin, and Salomon Sickert.
From LTL and limit-deterministic Büchi automata to deterministic parity automata.
In *TACAS*, pages 426–442, 2017.

Jan Křetínský and Javier Esparza.
Deterministic automata for the (F,G)-fragment of LTL.
In *CAV*, volume 7358 of *LNCS*, pages 7–22, 2012.

Jan Křetínský, Tobias Meggendorfer, Clara Waldmann, and Maximilian Weininger.
Index appearance record for transforming rabin automata into parity automata.
In *TACAS*, pages 443–460, 2017.

Nir Piterman.
From nondeterministic Büchi and Streett automata to deterministic parity automata.
In *LICS*, pages 255–264, 2006.

Shmuel Safra.
On the complexity of omega-automata.
In *FOCS*, pages 319–327, 1988.

Sven Schewe.
Tighter bounds for the determinisation of Büchi automata.
In *FoSSaCS*, volume 5504 of *LNCS*, pages 167–181, 2009.

Salomon Sickert, Javier Esparza, Stefan Jaax, and Jan Kretínský.
Limit-deterministic büchi automata for linear temporal logic.
In *CAV*, pages 312–332, 2016.

Moshe Y. Vardi and Pierre Wolper.
An automata-theoretic approach to automatic program verification (preliminary report).
In *LICS*, pages 332–344, 1986.

Games where the initial strategy is winning

category	games	random		trueness		svm	
small	15	3	20.00%	9	60.00%	14	**93.33%**
small safety	12	2	16.67%	12	**100.00%**	12	**100.00%**
small p-safety	12	2	16.67%	9	75.00%	12	**100.00%**
small co-safety	12	5	41.67%	12	**100.00%**	12	**100.00%**
small p-co-safety	12	4	33.33%	10	83.33%	12	**100.00%**
large	15	1	6.67%	8	53.33%	14	**93.33%**
large safety	13	0	0.00%	13	**100.00%**	13	**100.00%**
large co-safety	13	3	23.08%	13	**100.00%**	13	**100.00%**
lily	9	1	11.11%	4	44.44%	5	**55.56%**
ltl2dba	8	0	0.00%	0	0.00%	5	**62.50%**
ltl2dpa	11	0	0.00%	0	0.00%	6	**54.55%**
total	132	21	15.91%	90	68.18%	118	**89.39%**