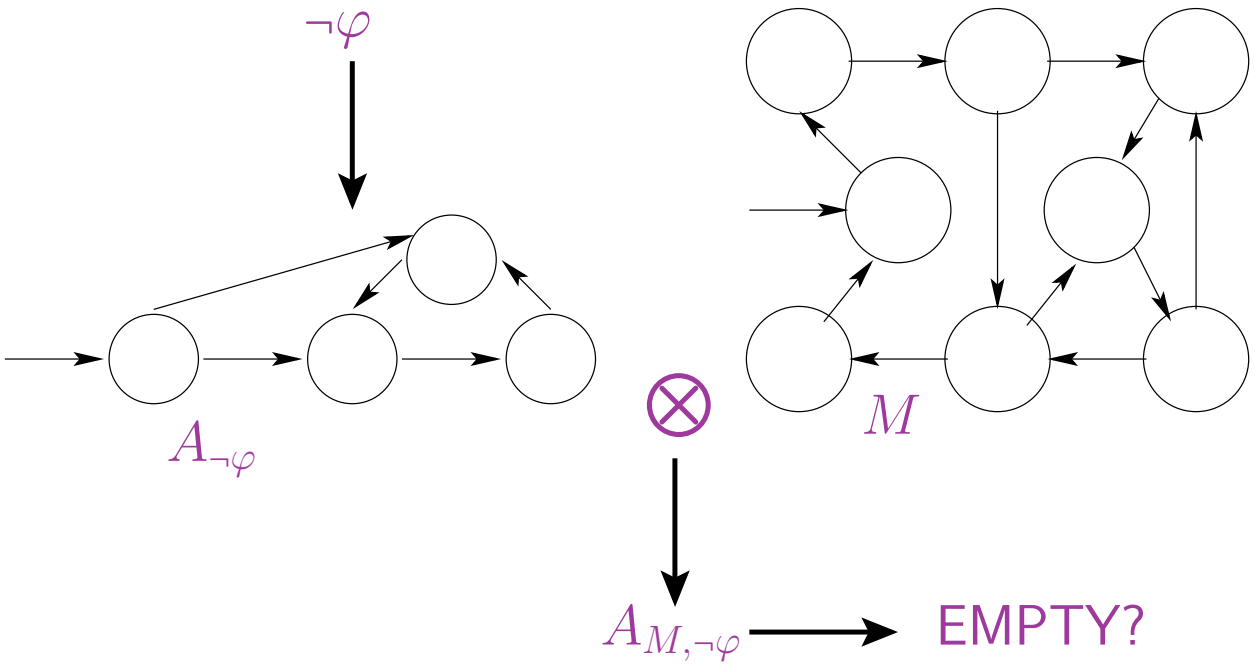
Developing an Open-Source, State-of-the-Art Symbolic Model-Checking Framework for the Model-Checking Research Community

Kristin Yvonne Rozier
Iowa State University

30th International Symposium on Model Checking Software
April 11, 2024

Automata-Theoretic Approach to Model Checking[1]



[1] Vardi, Wolper, LICS, 1986

Small Aircraft Transportation System (SAT)[2]

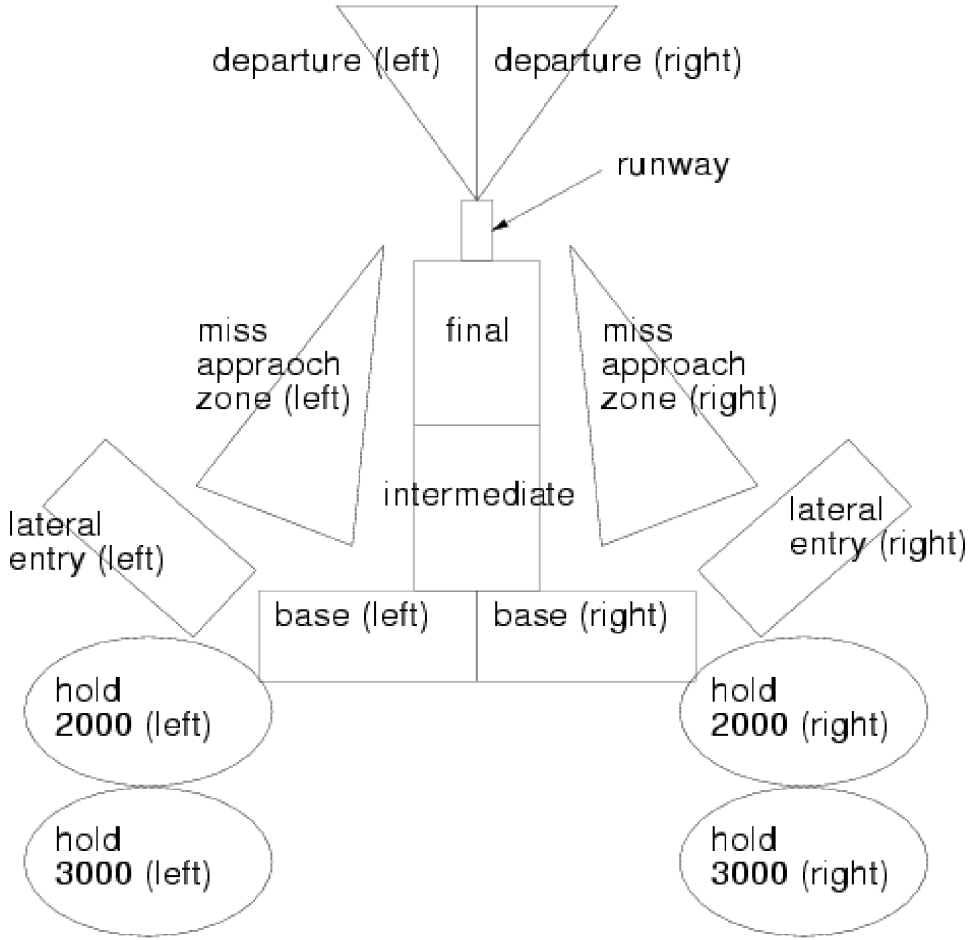


Figure 2. Operational Zones of the SCA.

[2] V. Carreño, C. Muñoz. "Safety Verification of the Small Aircraft Transportation System Concept of Operations."

Symbolic Model Checking[3]

$$\varphi = (p \mathcal{U} q)$$

```
module main() {
  VAR
p: boolean;
q: boolean;
EL_X__p_U_q : boolean;
DEFINE S__p_U_q := q | (p & EL_X__p_U_q);
TRANS  ( EL_X__p_U_q = (next(S__p_U_q) ))
FAIRNESS (!S__p_U_q | q)
SPEC !(S__p_U_q & EG TRUE) }
```

```
module main() {
p: boolean;
q: boolean;
FAIRNESS  TRUE; }
```

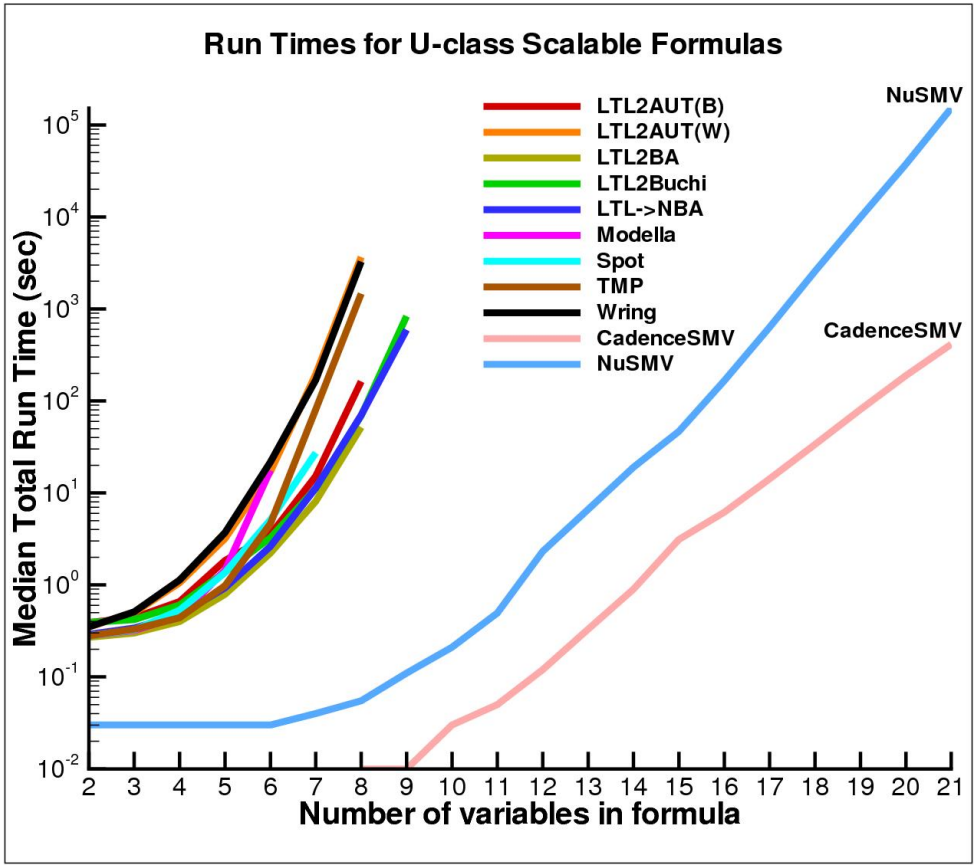
symbolic $A_{\neg\varphi}$ $\otimes$ universal M

$A_{M,\neg\varphi}$  nuXmv:EMPTY?

[3] Clarke, Grumberg, Hamaguchi, "Another look at LTL model checking." FMSD, 1997

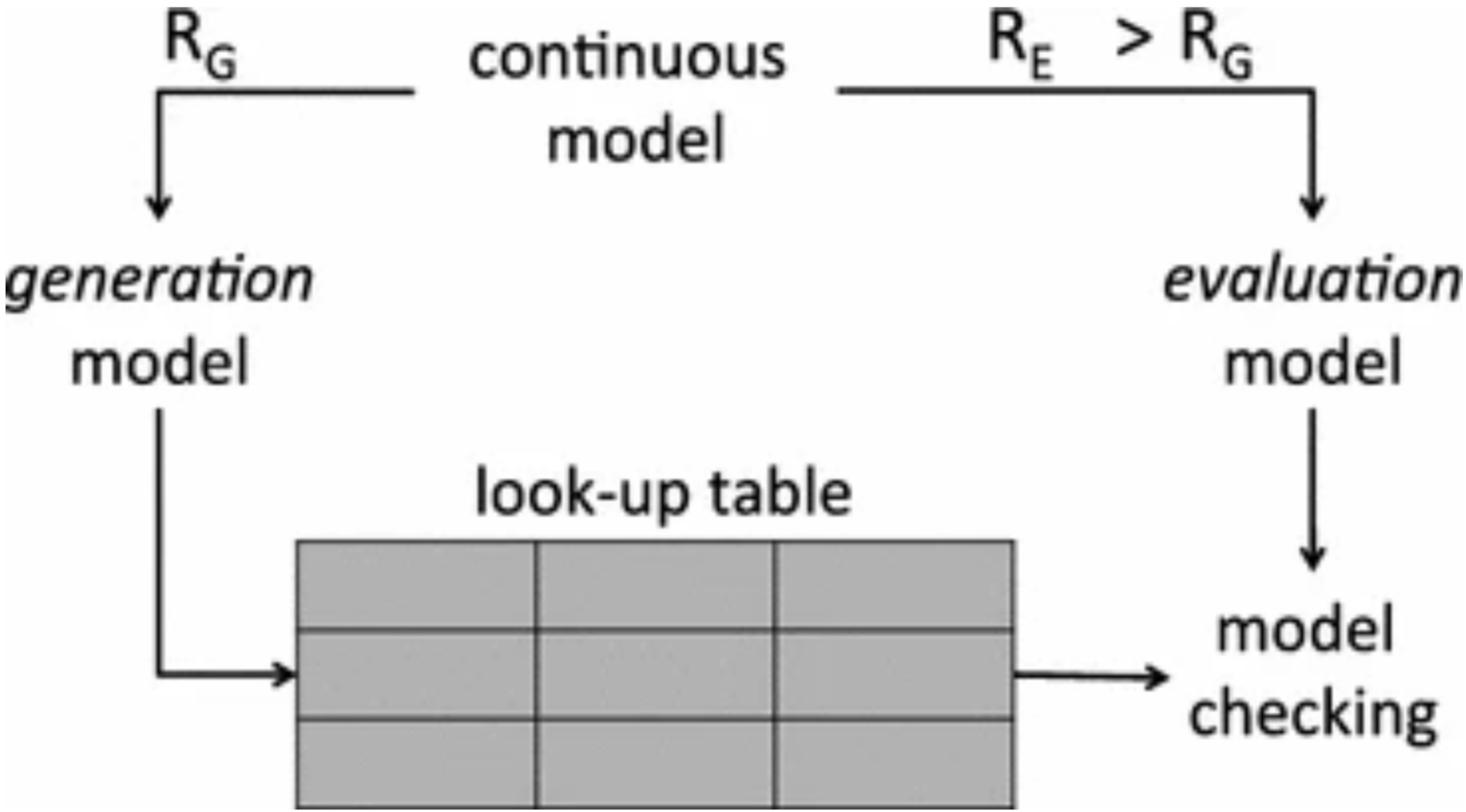
Symbolic Model Checkers in Satisfiability Checking

Symbolic Model Checkers push performance to industrial levels. [4]



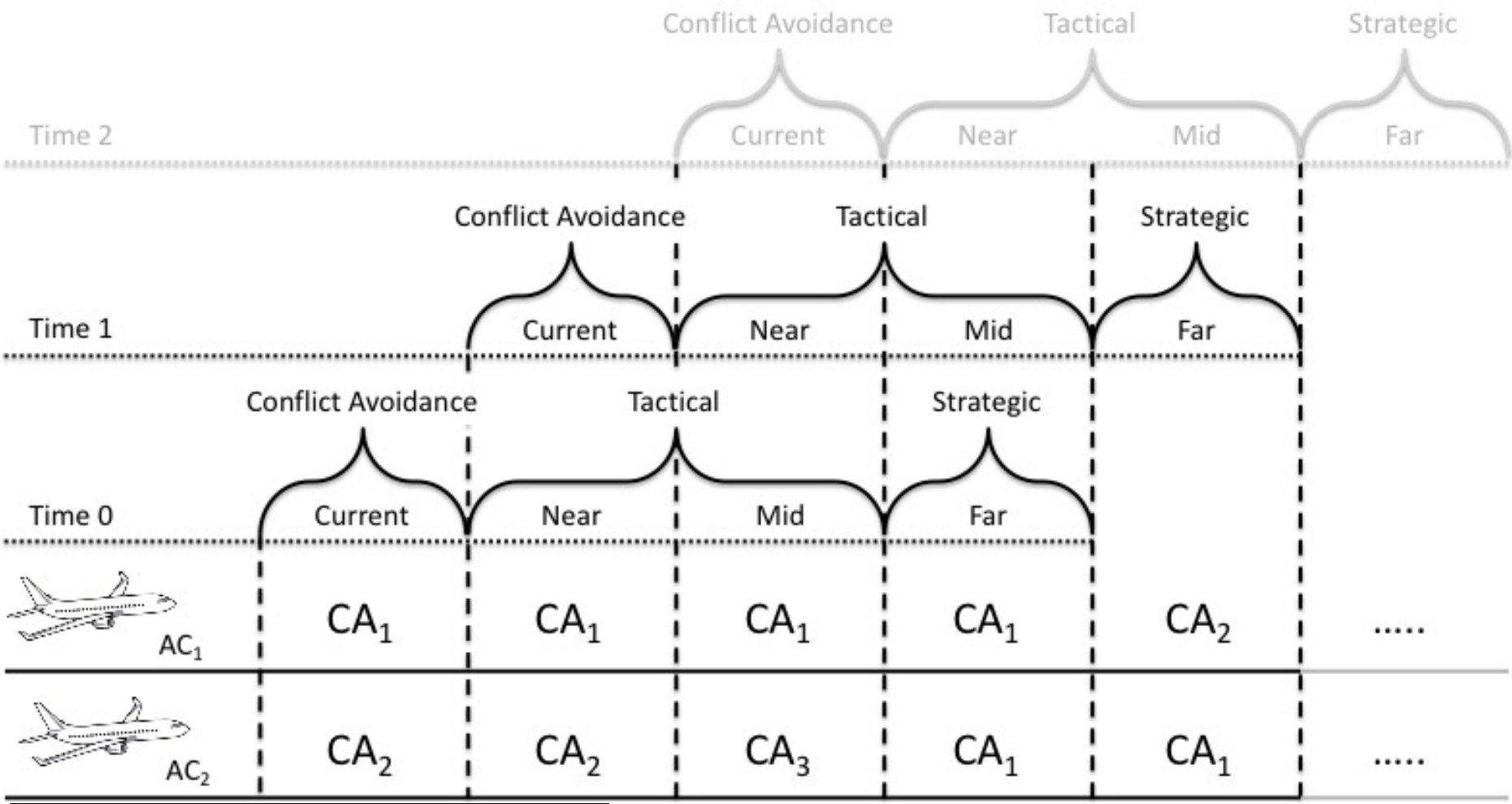
[4] Rozier and Vardi, "LTL Satisfiability Checking." SPIN'07.

Model Checking ACAS-X Controller[5]



[5] C. von Essen, D. Giannakopoulou. "Probabilistic verification and synthesis of the next generation airborne collision avoidance system." TACAS 2014

Formal Modeling: Time Windows[6]



[6] C. Mattarei, A. Cimatti, M. Gario, S. Tonetta, K. Y. Rozier. "Comparing Different Functional Allocations in Automated Air Traffic Control Design." In Formal Methods in Computer-Aided Design (FMCAD), IEEE/ACM, 2015.

Full-Scale Design Space Verification[7]

Summary of modeled dimensions and variants:
NASA Automated Airspace Concept

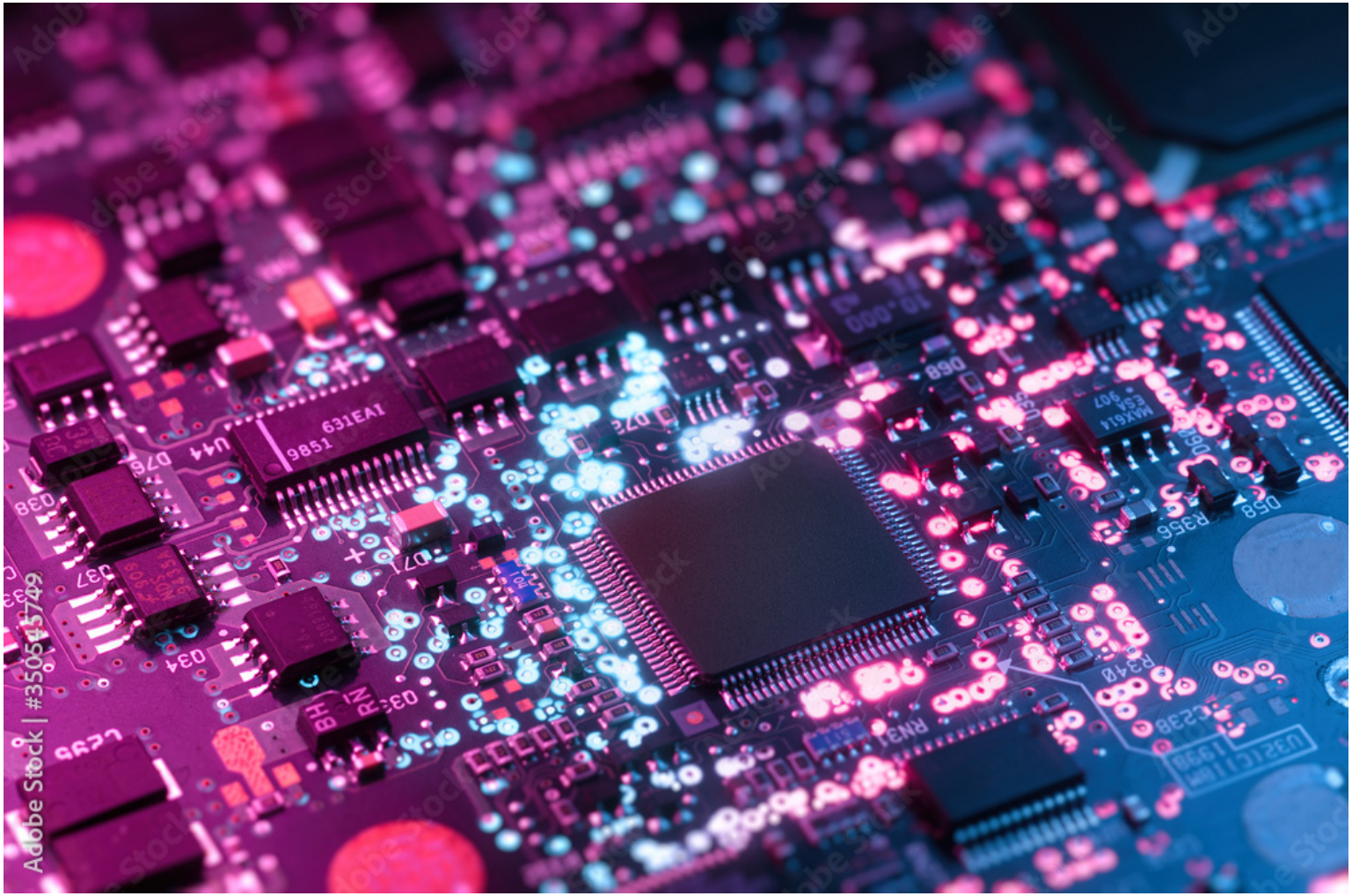
Name	Possible Values	Size
SSEP TS SA	ATC, SELF, SATC	3
SSEP SS SA	ATC, SELF, SATC	3
Instances	4, 3+1, 2+2, 1+3, 4	5
Info Sharing (GSEP-to-SSEP)	None, Current, Near, Mid, Far	5
Info Sharing (SSEP-to-ATC)	None, Current, Near, Mid, Far	5
Burdening Rules	Undef, *GSEP*, *SSEP*	3
ACDR Implementations	Simple, Asymmetric, Non-Receptive	3
Com Steps	1, 2, ...	2
TOTAL	-	**20,250**
Focus group	-	**1,620**

[7] Marco Gario, Alessandro Cimatti, Cristian Mattarei, Stefano Tonetta and Kristin Y. Rozier. "Model Checking at Scale: Automated Air Traffic Control Design Space Exploration." In *Computer Aided Verification (CAV)*, 2016.

Focusing on Linear Models of Time...[8]

[8] Vardi, "Branching vs. Linear Time: Final showdown." TACAS, 2001

Focusing on "Hardware" Model Checking



Evolution of Model-Checking Algorithms

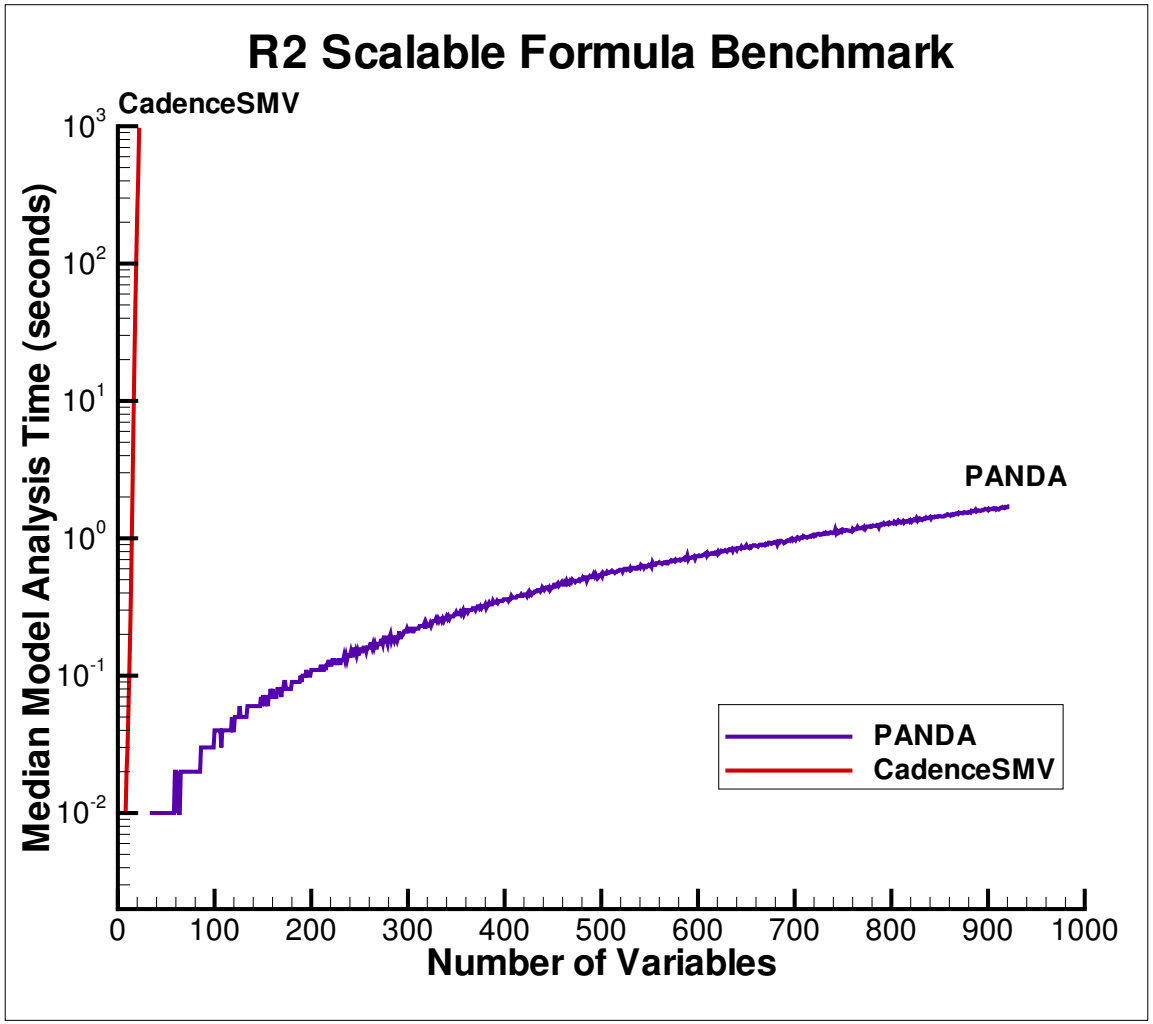
1. BDD-based
2. SAT-based / bounded model checking
3. IC3 / k-liveness

Evolution of Model-Checking Algorithms

1. BDD-based
2. SAT-based / bounded model checking
3. IC3 / k-liveness

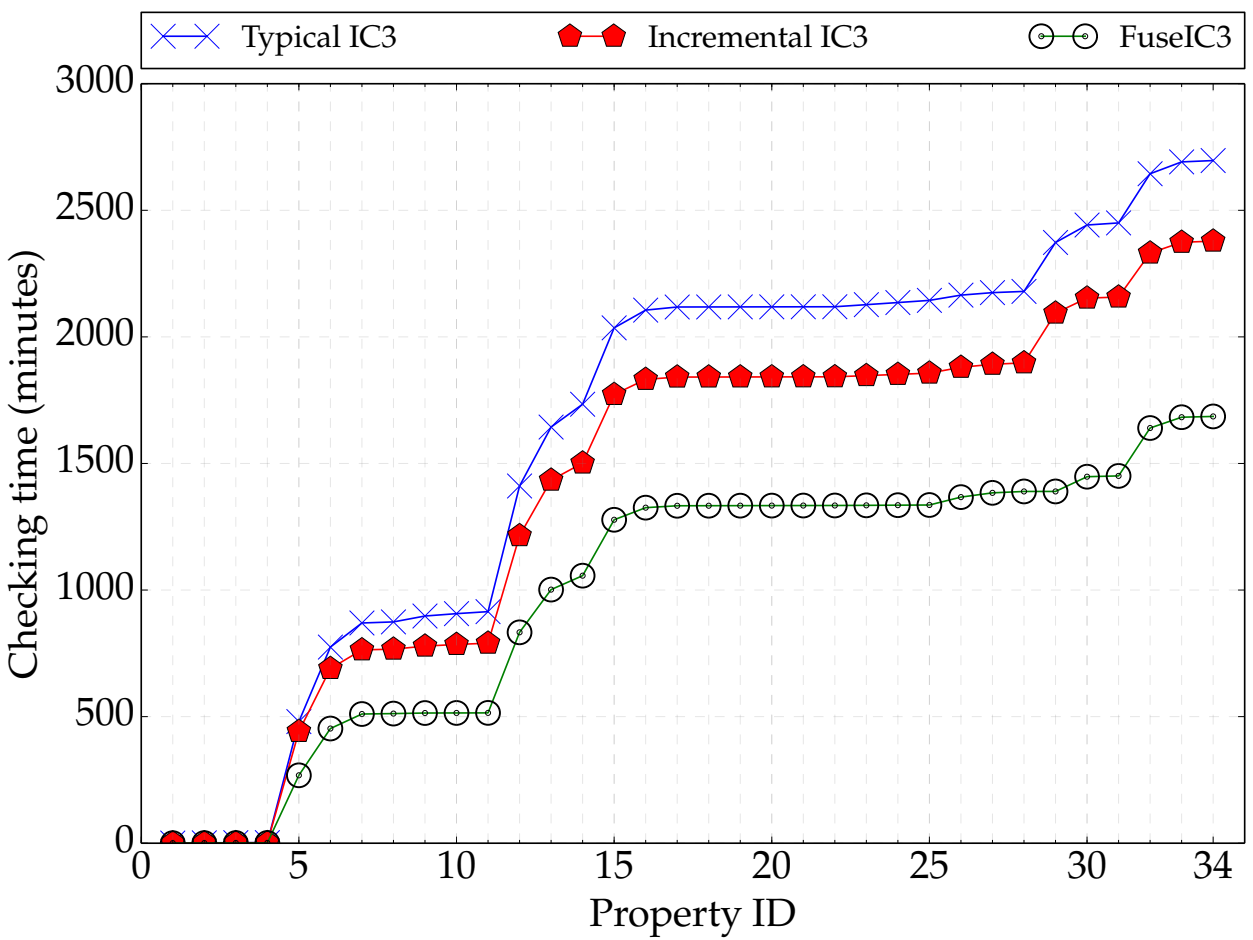
Symbolic model checking progressed from bit-level to word level

The Problem[9]



[9] K.Y.Rozier and M.Y.Vardi, "A Multi-Encoding Approach for LTL Symbolic Satisfiability Checking," FM 2011.

FuseIC3: An Algorithm for Checking Large Design Spaces[10]



Model checking **34 formulas** over **1,620 models** is **5.48x faster**

[10] Rohit Dureja and Kristin Yvonne Rozier. "FuseIC3: An Algorithm for Checking Large Design Spaces." In Formal Methods in Computer-Aided Design (FMCAD), IEEE/ACM, Vienna, Austria, October 2-6, 2017.

The Major Problems

- nuXmv **closed source**: can't check internal algorithm

The Major Problems

- nuXmv **closed source**: can't check internal algorithm
- IBM's tools **closed source**: same problem

The Major Problems

- nuXmv **closed source**: can't check internal algorithm
- IBM's tools **closed source**: same problem
- **Can't build** on either tool

The Major Problems

- nuXmv **closed source**: can't check internal algorithm
- IBM's tools **closed source**: same problem
- **Can't build** on either tool
- Have **models in SMV** language: can't MC with IBM's algorithm

The Major Problems

- nuXmv **closed source**: can't check internal algorithm
- IBM's tools **closed source**: same problem
- **Can't build** on either tool
- Have **models in SMV** language: can't MC with IBM's algorithm
- Have **new algorithm**; also can't check SMV models with it

The Major Problems

- nuXmv **closed source**: can't check internal algorithm
- IBM's tools **closed source**: same problem
- **Can't build** on either tool
- Have **models in SMV** language: can't MC with IBM's algorithm
- Have **new algorithm**; also can't check SMV models with it

Solution: program an **entire model checker** taking SMV as input and implementing all three algorithms from scratch to compare them

The Major Problems

- nuXmv **closed source**: can't check internal algorithm
- IBM's tools **closed source**: same problem
- **Can't build** on either tool
- Have **models in SMV** language: can't MC with IBM's algorithm
- Have **new algorithm**; also can't check SMV models with it

Solution: program an **entire model checker** taking SMV as input and implementing all three algorithms from scratch to compare them

- not publishable, LOTS of time, hard to get right, waste of effort

The Major Problems

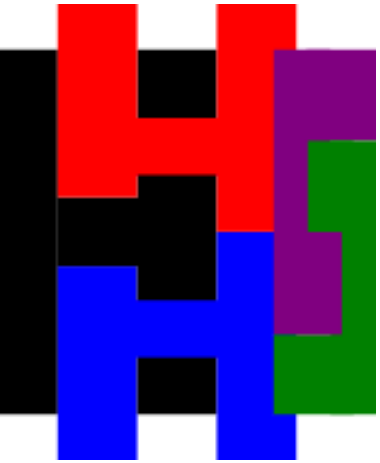
- nuXmv **closed source**: can't check internal algorithm
- IBM's tools **closed source**: same problem
- **Can't build** on either tool
- Have **models in SMV** language: can't MC with IBM's algorithm
- Have **new algorithm**; also can't check SMV models with it

Solution: program an **entire model checker** taking SMV as input and implementing all three algorithms from scratch to compare them

- not publishable, LOTS of time, hard to get right, waste of effort

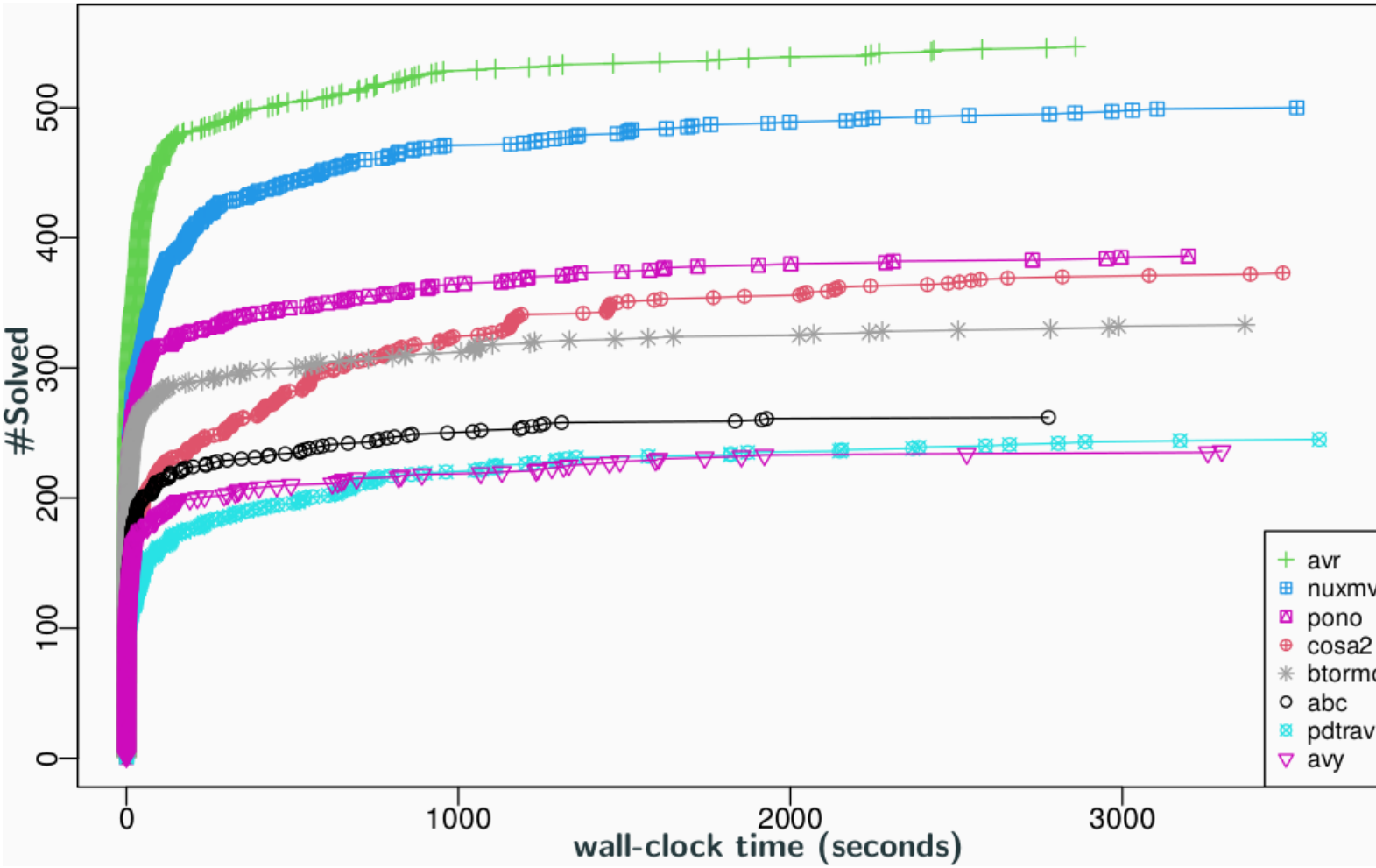
How do we do better?

HWMCC: Hardware Model Checking Competition (2020)



Word-level reasoning

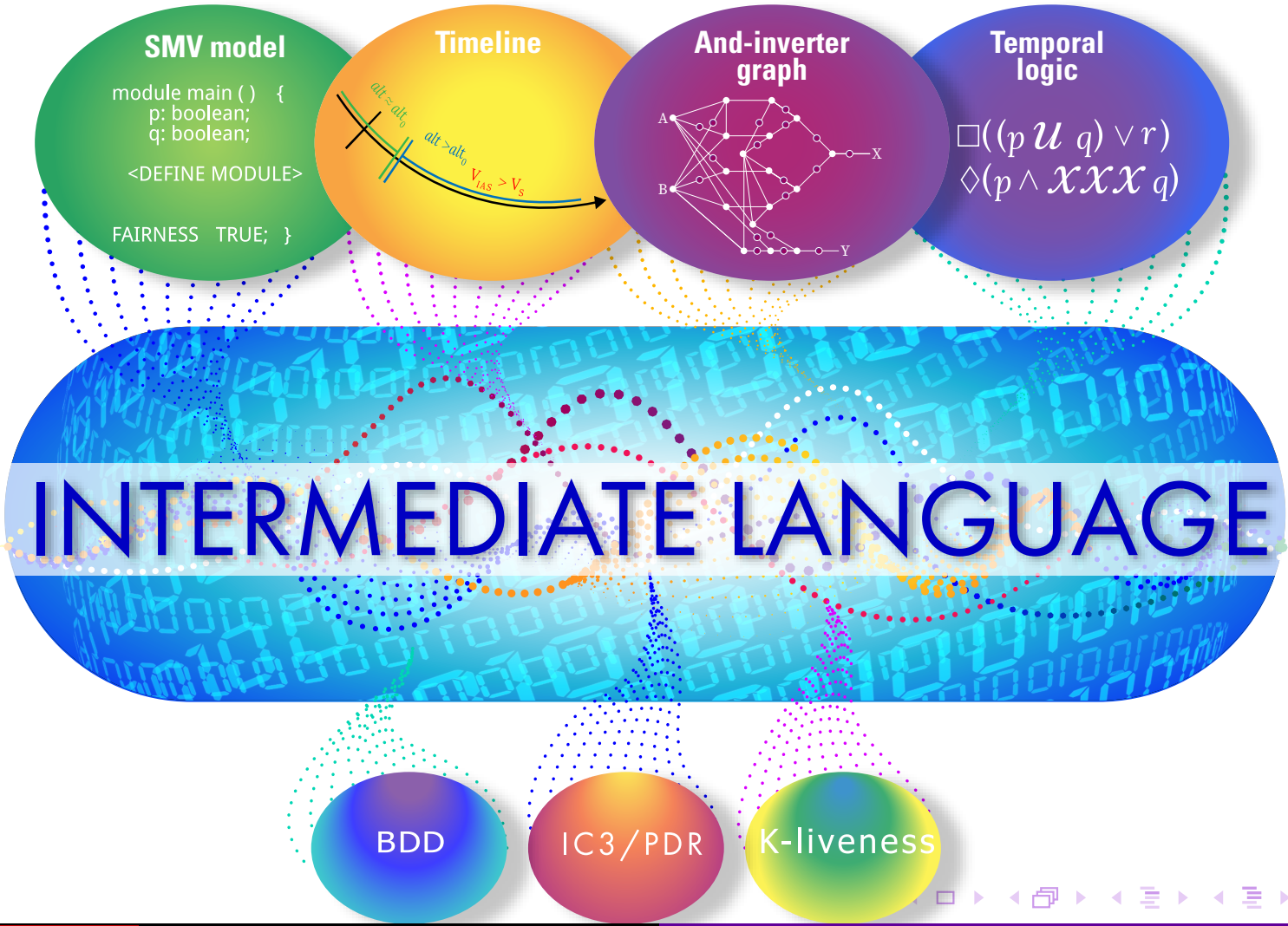
BTOR2



The Problem Continues...

- nuXmv, CadenceSMV, others are **closed source**
- ABC, HWMCC tools are **limited to low-level modeling languages**
- No **open-source, research-enabling connection** between:
  - Rich modeling languages with real-world benchmark models
  - State-of-the-art back-end MC algorithms

MoXI: Model eXchange Interlingua



Goals for Intermediate Language

- Allow adding a **modeling language** via **translation to/from MoXI**
- Allow adding an **MC algorithm** via **translation to/from MoXI**
- MoXI is efficient/accessible so as to **encourage usage in future MCs**
- MoXI: suitable for on-going **community standard**

Basis for MoXI as the Intermediate Language

- **"Best of" previous work**: SMT-LIB, VMT-LIB-Iowa, VMT-LIB-FBK, SAL/Sally, Cryptol, OCRA, synchronous reactive components, ...
- Relatively **simple**, but **complete**
- **Easy to parse**
- Allow **different encodings** (from/to higher/lower levels)

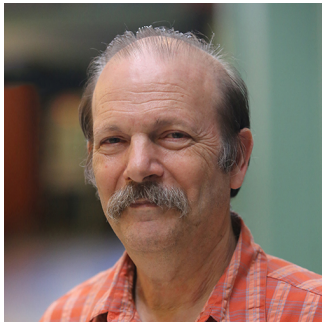
Core Design Team

Investigators:

K.Y. Rozier

Natarajan Shankar

Cesare Tinelli

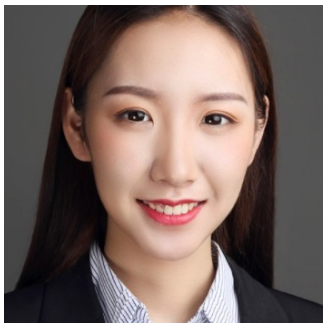


Moshe Vardi

Students:

Laura Gamboa Guzman

Chris Johannsen



Yi Lin

Technical Advisory Board (TAB)

Rajeev Alur (Univ of Pennsylvania)
Clark Barrett (Stanford)
Dirk Beyer (LMU Munich)
Armin Biere (Albert-Ludwigs Univ)
Nikolaj Bjorner (Microsoft Research)
Dimitra Giannakopoulou (Amazon)
Alberto Griggio (FBK)
Orna Grumberg (Technion)
Aarti Gupta (Princeton)
Arie Gurfinkel (Univ of Waterloo)
Ahmed Irfan (SRI)
John Matthews (Intel)
Ken McMillan (UT Austin)
Alan Mishchenko (Berkeley)
Karem Sakallah (Univ of Michigan)
Bernhard Steffen (TU Dortmund)
Aaron Tomb (Amazon)
Stefano Tonetta (FBK)

MoXI: Base Logic

- **Many-sorted first-order logic** w/ equality, quantifiers, let binders, algebraic datatypes (like SMT-LIB V2)
- Extention to (first-order) **temporal logic** w/ discrete, linear time, finite & infinite trace-based semantics
- Commands for defining and verifying **multi-component reactive systems**
- Checks for **reachability conditions**, **deadlocks** (indirectly: state & transition invariants)
- Includes **fairness conditions** on system inputs

Language Design Highlights: Transition Systems

Transition system $S = (\ I_S[\boldsymbol{i}, \boldsymbol{o}, \boldsymbol{s}],\quad T_S[\boldsymbol{i}, \boldsymbol{o}, \boldsymbol{s}, \boldsymbol{i'}, \boldsymbol{o'}, \boldsymbol{s'}]\ )$

where

· $\boldsymbol{i}$ and $\boldsymbol{i'}$ are two tuples of *input variables* with the same length and type;
· $\boldsymbol{o}$ and $\boldsymbol{o'}$ are two tuples of *output variables* with the same length and type;
· $\boldsymbol{s}$ and $\boldsymbol{s'}$ are two tuples of *local variables* with the same length and type;
· I_S is the system's *initial state condition*, expressed as a formula with no (free) variables from $\boldsymbol{i'}$, $\boldsymbol{o'}$, and $\boldsymbol{s'}$;
· T_S is the system's *transition condition*, expressed as a formula over the variables $\boldsymbol{i}$, $\boldsymbol{o}$, $\boldsymbol{s}$, $\boldsymbol{i'}$, $\boldsymbol{o'}$, and $\boldsymbol{s'}$.

Language Design Highlights: System Checking Command

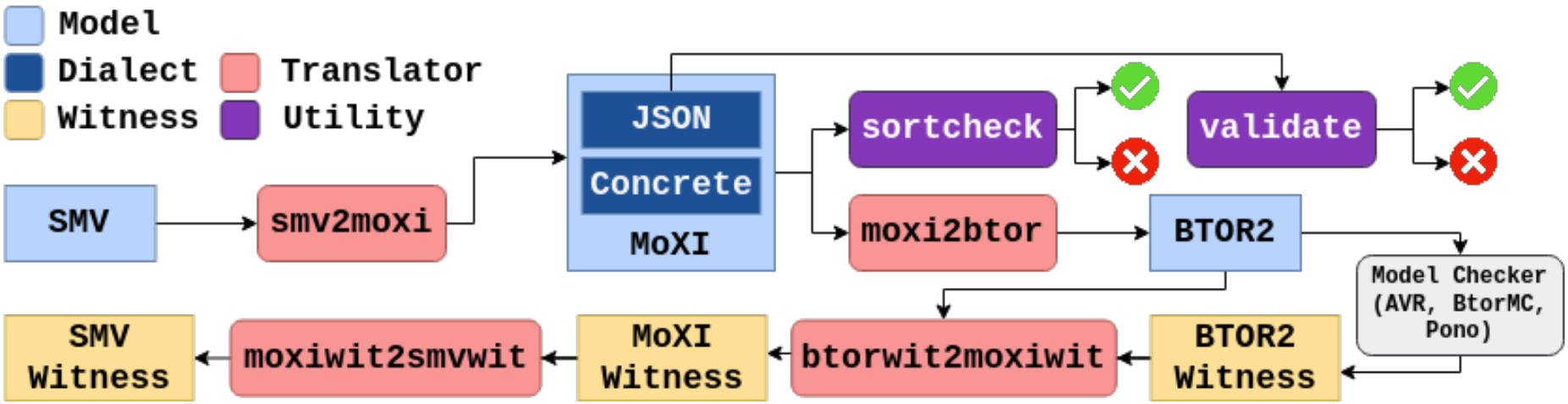
(check-system S
:input ((i_1 δ_1) $\cdots$ (i_m δ_m))
:output ((o_1 τ_1) $\cdots$ (o_n τ_n))
:local ((s_1 σ_1) $\cdots$ (s_p σ_p))
:assumption (a A)
:fairness (f F)
:reachable (r R)
:current (c C)
:query (q (g_1 $\cdots$ g_q))
:queries ((q_1 ($g_{1,1}$ $\cdots$ g_{1,n_1})) $\cdots$ (q_t ($g_{t,1}$ $\cdots$ g_{t,n_t})))
)

Language Design Highlights: Check System Response

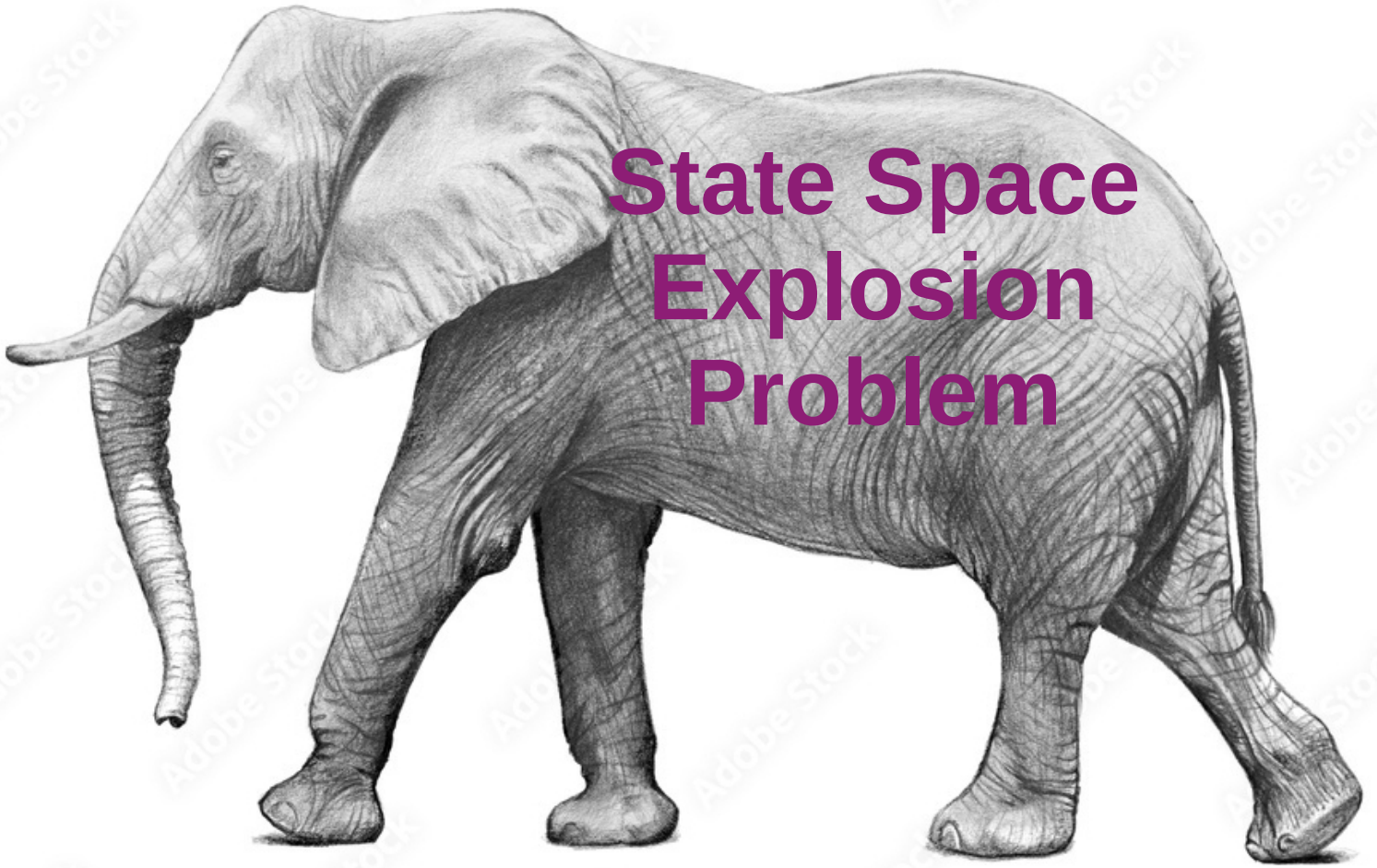
```
(check-system-response
 :query (q1 :result sat :model m :trace t)
 :query (q2 :result unsat :certificate c)
 :query (q3 :result unknown)                    ; for timeouts and other cases
 :trace (t :prefix p :lasso l)                  ; t = pl^omega
 :model (m M)                                   ; M is model in SMT-LIB format

; state/valuation
 :trail (p ( ((i i_0) (o o_0) (s s_0) (r r_0) (f f_0))
               ...
             ((i i_k) (o o_k) (s s_k) (r r_k) (f f_k))
           )
         )
 :trail (l ( ( ... ) ... ( ... ) ))
 :certificate (c :inv F :k n)
)
```

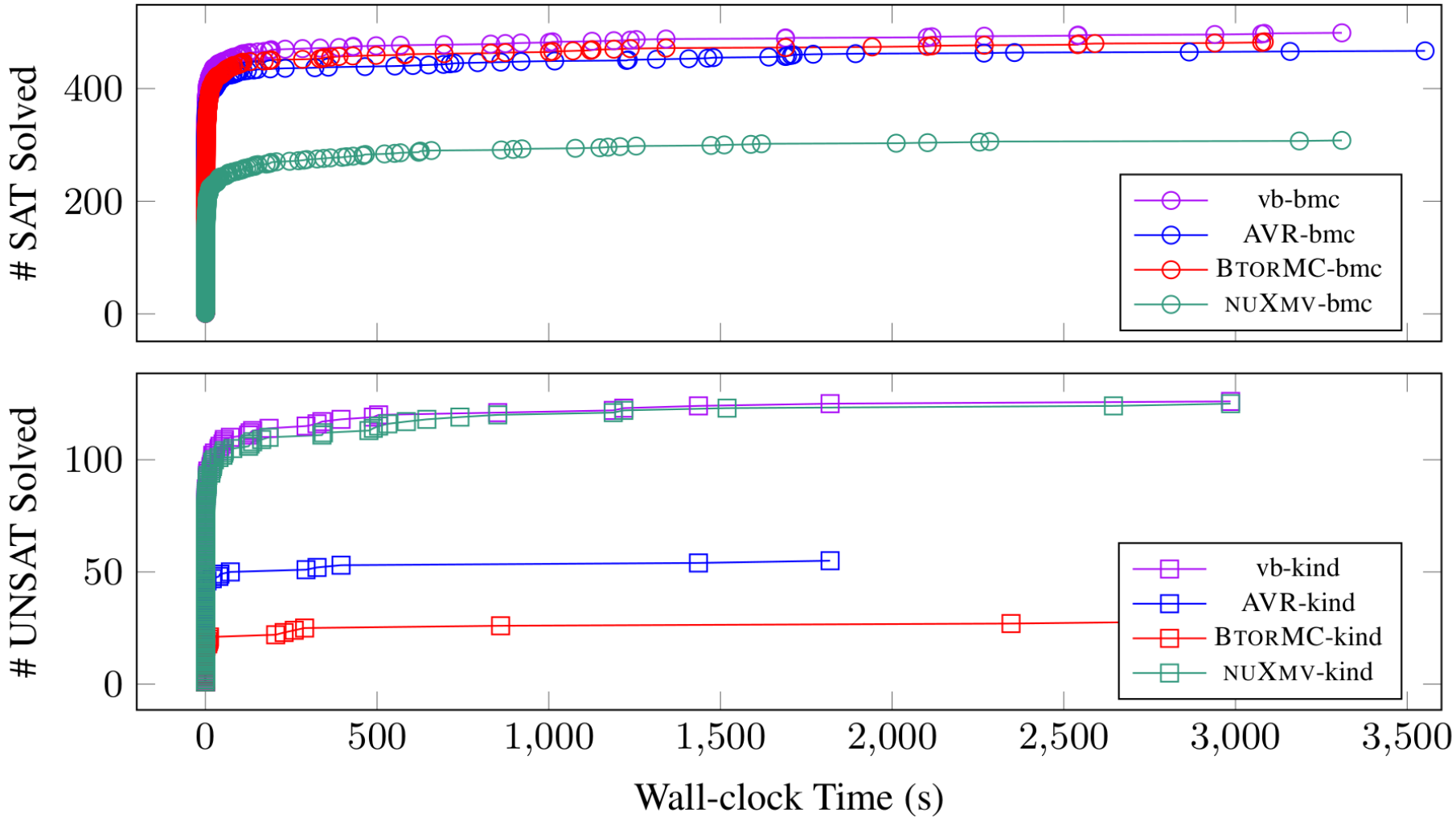
Building the First Translators[11]



[11] C. Johannsen, K. Nukala, R. Dureja, A. Irfan, N. Shankar, C. Tinelli, M. Y. Vardi, K. Y. Rozier. "Symbolic Model-Checking Intermediate-Language Tool Suite." CAV 2024.

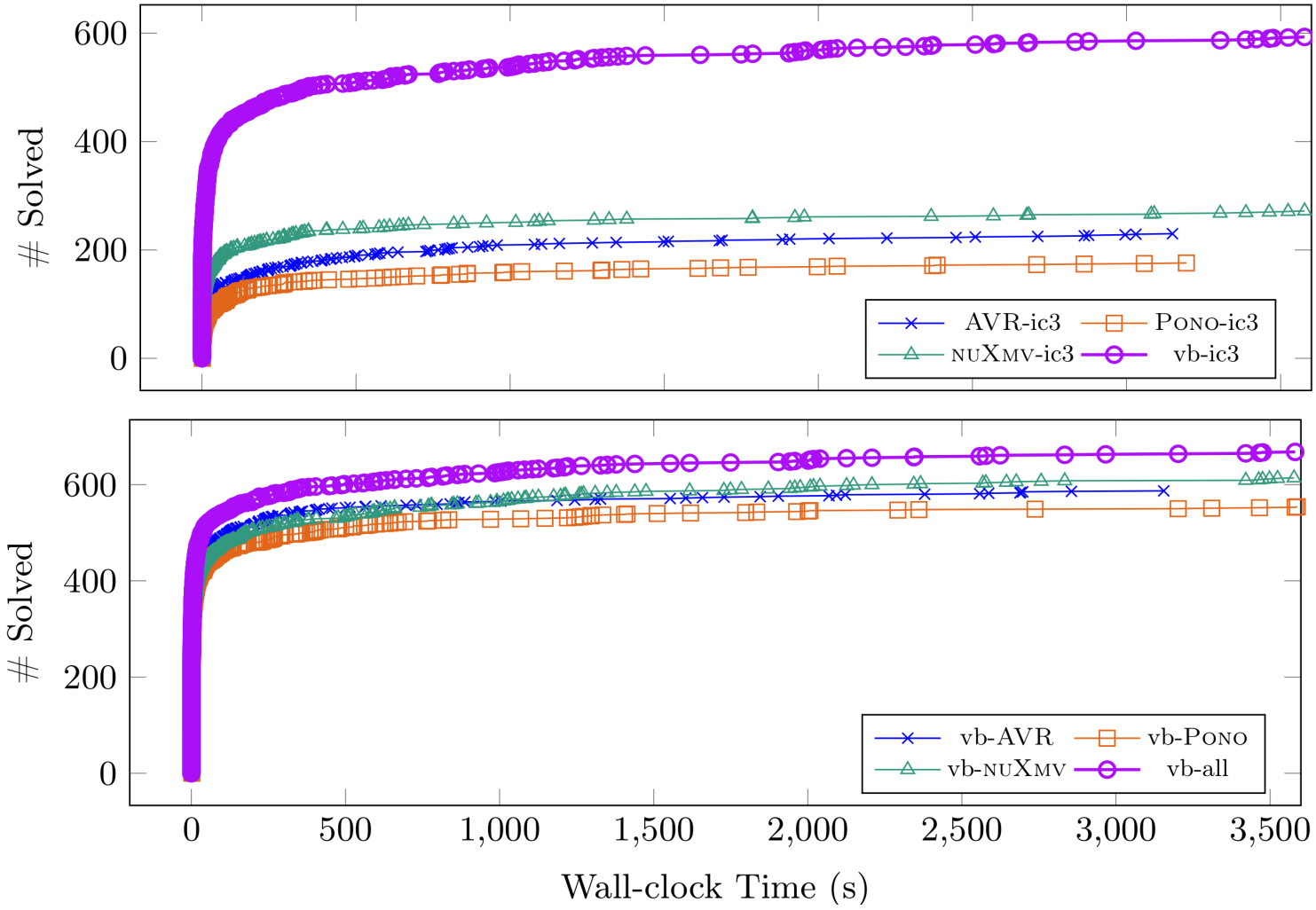

State Space
Explosion
Problem

Experiments: 960 Boolean/word/array SMV Models[12]



[12] C. Johannsen, K. Nukala, R. Dureja, A. Irfan, N. Shankar, C. Tinelli, M. Y. Vardi, K. Y. Rozier. "Symbolic Model-Checking Intermediate-Language Tool Suite." CAV 2024.

960 QF_BV & QF_ABV benchmarks (nuXmv release)[13]



[13] K. Y. Rozier, R. Dureja, A. Irfan, C. Johannsen, K. Nukala, N. Shankar, C. Tinelli, M. Y. Vardi. "MoXI: An Intermediate Language for Symbolic Model Checking." SPIN 2024.

Join the Conversation!

OSSyM

Technical Information | CAV 2024 | Workshop Agenda | Organizational Information

July 23rd 2024

The OSSyM workshop aims to introduce the progress to-date on this collaborative effort and involve the full international research community to reduce barriers to developing new model-checking algorithms and research platforms. The workshop will include tutorials and feedback from the international Model Checking Technical Advisory Board on a design for an extensible framework centering on an intermediate language that will unify popular front-end modeling languages with state-of-the-art back-end model-checking tools.

Open-Source, State-of-the-Art Symbolic Model-Checking Framework for the Model-Checking Research Community

Project Links

Home: https://modelchecker.temporallogic.org

GitHub Organization: https://modelchecker.github.io/

MoXI Language Definition:
https://github.com/ModelChecker/IL/blob/main/description.md

CAV 2024 Workshop: OSSyM:
https://laboratory.temporallogic.org/ossym/

FMCAD 2023 Workshop:
https://github.com/ModelChecker/FMCAD23-Tutorial

SPIN 2024 paper: MoXI semantics:
https://research.temporallogic.org/papers/SPIN2024.pdf

CAV 2024 tool paper: MoXI translators:
https://research.temporallogic.org/papers/CAV2024.pdf
artifact: https://zenodo.org/records/10946779

Summary

The time has come for model-checking community standards

- **Participate:** email list, language design feedback, community forum: **OSSyM@CAV 2024**
- Available Now: **SMV** $\leftrightarrow$ **MoXI** $\leftrightarrow$ **BTOR2**
- **Contribute** future translators:
  - Your Modeling Language $\leftrightarrow$ **MoXI**
  - **MoXI** $\leftrightarrow$ Your Back-end MC Algorithm
  - **MoXI** $\leftrightarrow$ Your Proof Assistant
- **Optimize! Extend! Benchmark! Compare! Research!**

modelchecker.temporallogic.org