

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

Kristin Yvonne Rozier

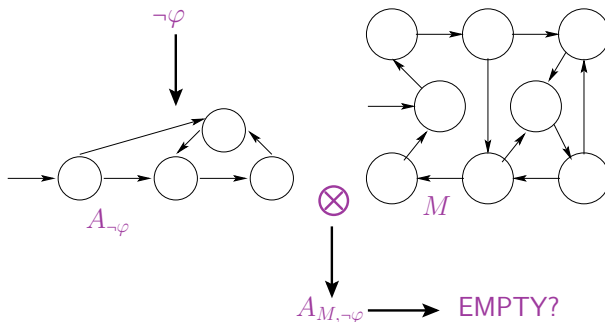
Iowa State University



25th Brazilian Symposium on Formal Methods

December 8, 2022

Automata-Theoretic Approach to Model Checking¹



¹Vardi, Wolper, LICS, 1986

Symbolic Model Checking²

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



```
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 $A_{\neg\varphi}$



universal M



$A_{M, \neg\varphi}$

→ nuXmv:EMPTY?

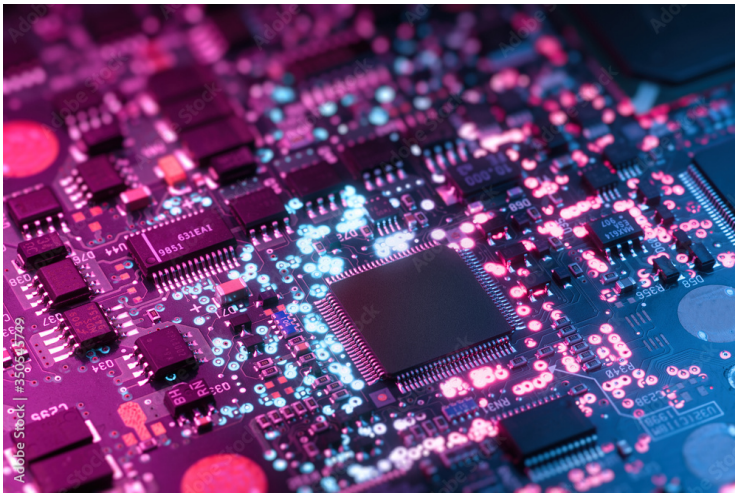
² Clarke, Grumberg, Hamaguchi, "Another look at LTL model checking." FMSD, 1997.

Focusing on Linear Models of Time...³



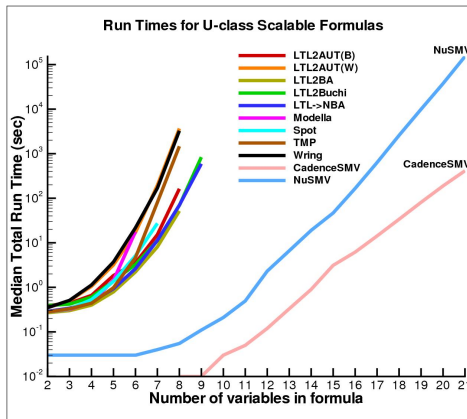
³Vardi, "Branching vs. Linear Time: Final showdown." TACAS, 2001

Focusing on “Hardware” Model Checking



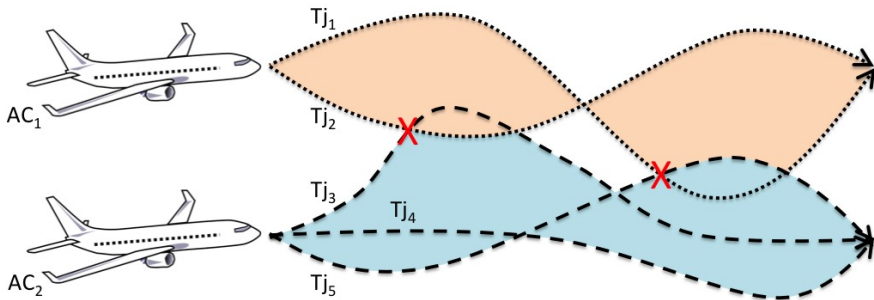
Symbolic Model Checkers in Satisfiability Checking

Symbolic Model Checkers push performance to industrial levels. ⁴



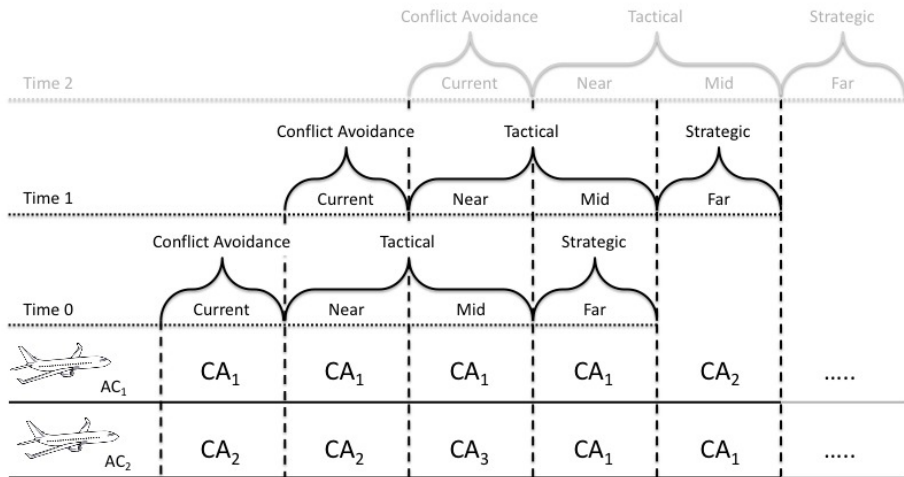
⁴Rozier and Vardi, “LTL Satisfiability Checking.” SPIN’07.

Formal Modeling: Conflict Areas⁵

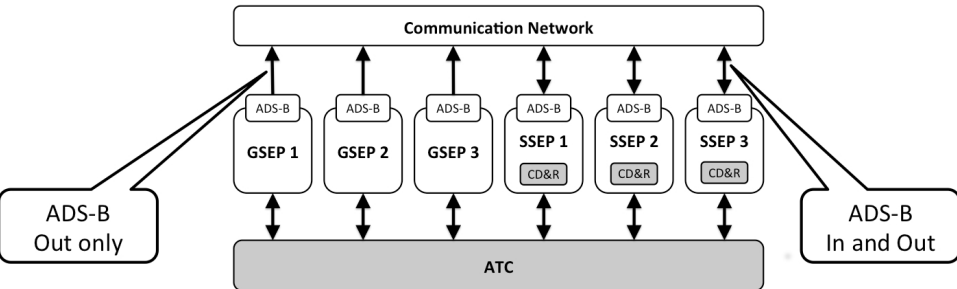


⁵ Cristian Mattarei, Alessandro Cimatti, Marco Gario, Stefano Tonetta and Kristin Y. Rozier. "Comparing Different Functional Allocations in Automated Air Traffic Control Design." In Formal Methods in Computer-Aided Design (FMCAD), IEEE/ACM, 2015.

Formal Modeling: Time Windows



Formal Modeling: System Components⁶



⁶ 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. [↩](#) [⌂](#) [⌕](#) [🔄](#)

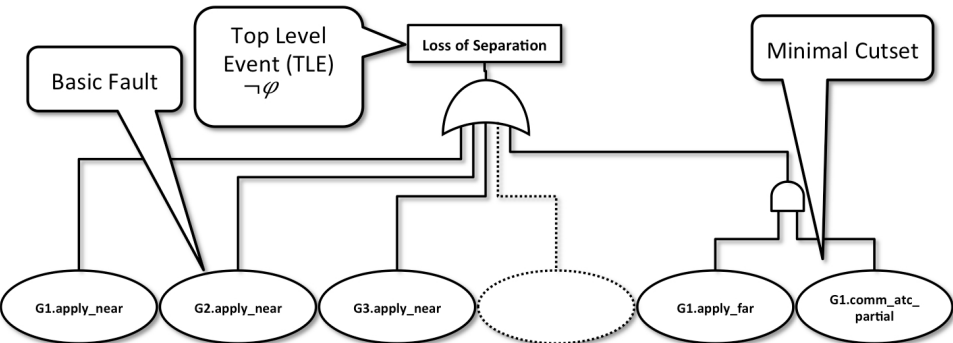
Full-Scale Design Space Verification⁷

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, <i>GSEP</i> , <i>SSEP</i>	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. ▶ ◀ ≡ ≡ ≡ 🔍 ↺

Fault Tree Analysis⁸



⁸ Cristian Mattarei, Alessandro Cimatti, Marco Gario, Stefano Tonetta and Kristin Y. Rozier. "Comparing Different Functional Allocations in Automated Air Traffic Control Design." In Formal Methods in Computer-Aided Design (FMCAD), IEEE/ACM, 2015.

Evolution of Model-Checking Algorithms

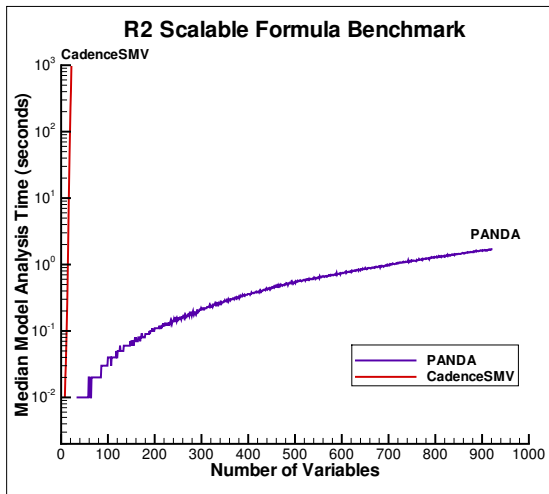
- 1 BDD-based
- 2 SAT-based / bounded model checking
- 3 IC3 / k-liveness

Evolution of Model-Checking Algorithms

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

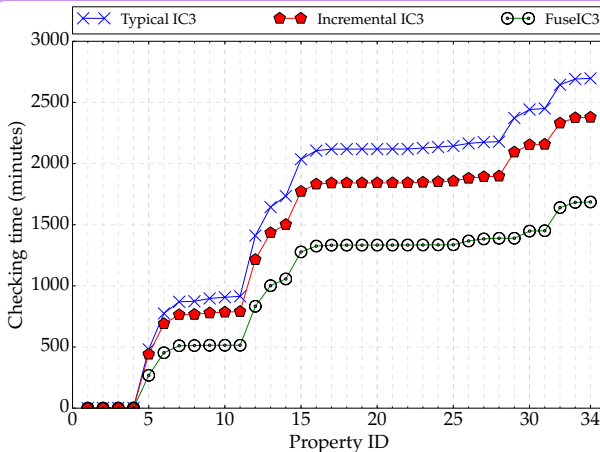
Symbolic model checking progressed from bit-level to word level

The Problem⁹



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

FuselC3: An Algorithm for Checking Large Design Spaces¹⁰



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

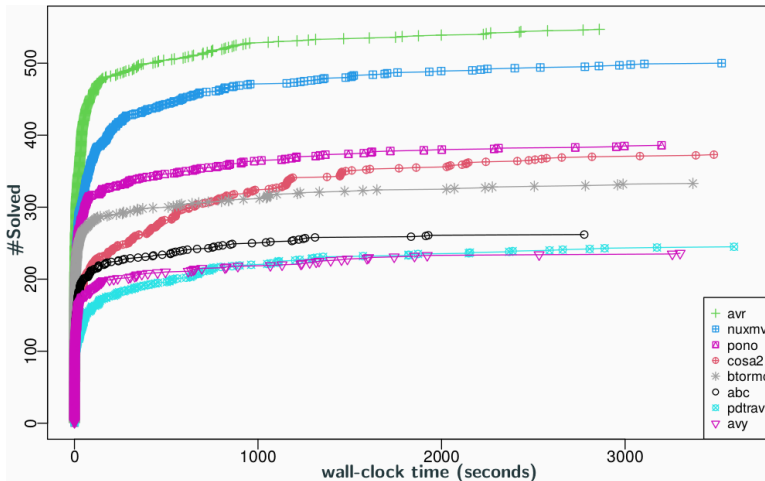
¹⁰ Rohit Dureja and Kristin Yvonne Rozier. "FuselC3: An Algorithm for Checking Large Design Spaces." In Formal Methods in Computer-Aided Design (FMCAD), IEEE/ACM, Vienna, Austria, October 2-6, 2017.

HWMCC: Hardware Model Checking Competition (2020)



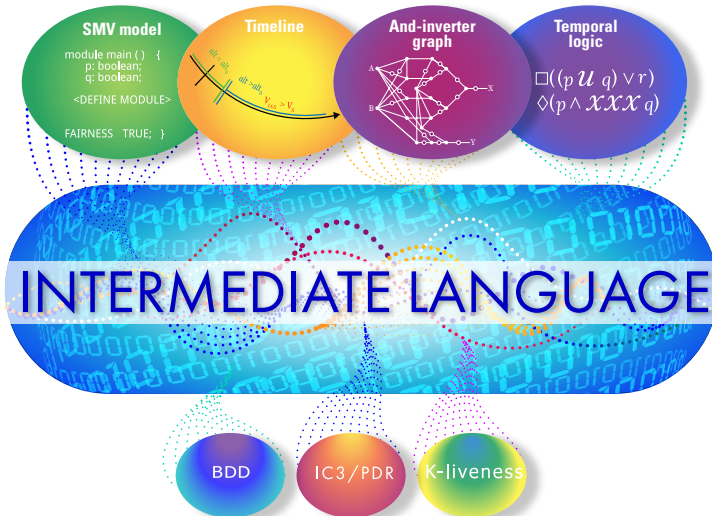
Word-level
reasoning

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



Goals for Intermediate Language

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

Basis for Intermediate Language

- “Best of” previous work: SMT-LIB, VMT-LIB-Iowa, VMT-LIB-FBK, 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

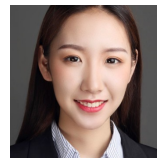
Students:



Laura Gamboa Guzman



Chris Johannsen



Yi Lin

Technical Advisory Board (TAB)

Rajeev Alur (Univ of Pennsylvania)

Clark Barrett (Stanford)

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)

Project Links

Home:

<https://modelchecker.temporallogic.org>

GitHub Organization:

<https://modelchecker.github.io/>

Intermediate Language Definition:

<https://github.com/ModelChecker/IL/blob/main/description.md>

Language Design Highlights: Transition Systems

Transition system $S = (I_S[i, o, s], T_S[i, o, s, i', o', s'])$

where

- i and i' are two tuples of *input variables* with the same length and type;
- o and o' are two tuples of *output variables* with the same length and type;
- s and 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 i' , o' , and s' ;
- T_S is the system's *transition condition*, expressed as a formula over the variables i , o , s , i' , o' , and s' .

Language Design Highlights: System Checking Command

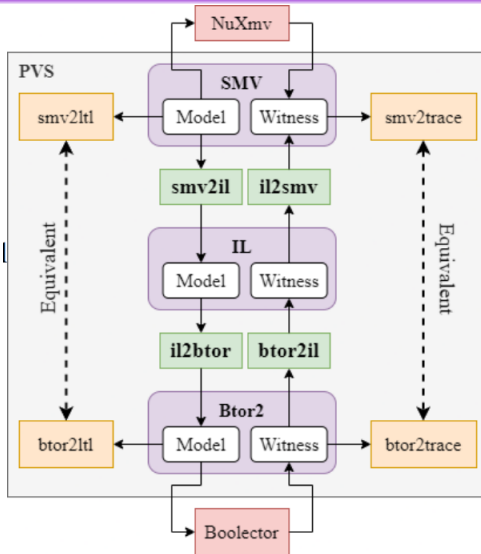
```
(check-system S
  :input ( (  $i_1 \delta_1$  )  $\cdots$  (  $i_m \delta_m$  ) )
  :output ( (  $o_1 \tau_1$  )  $\cdots$  (  $o_n \tau_n$  ) )
  :local ( (  $s_1 \sigma_1$  )  $\cdots$  (  $s_p \sigma_p$  ) )
  :assumption (  $a A$  )
  :fairness (  $f F$  )
  :reachable (  $r R$  )
  :current (  $c C$  )
  :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 = plomega
: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



Summary

The time has come for model-checking community standards

- **Participate:** name vote, email list, language design feedback, community forums
- Coming Soon: **PVS2C: SMV++** $\leftrightarrow$ **IL** $\leftrightarrow$ **BTOR2**
- **Contribute** future translators:
 - Your Modeling Language $\leftrightarrow$ Intermediate Language
 - Intermediate Language $\leftrightarrow$ Your Back-end MC Algorithm
- **Optimize! Expand! Compare! Research!**

modelchecker.temporallogic.org