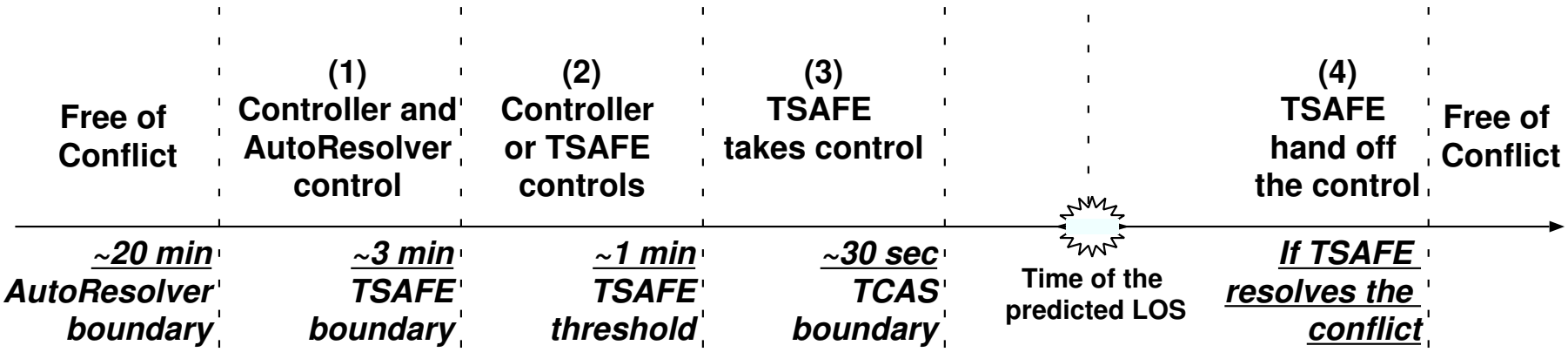
Challenges in Tool Integration: Making Formal Methods Universal

Kristin Yvonne Rozier
Iowa State University

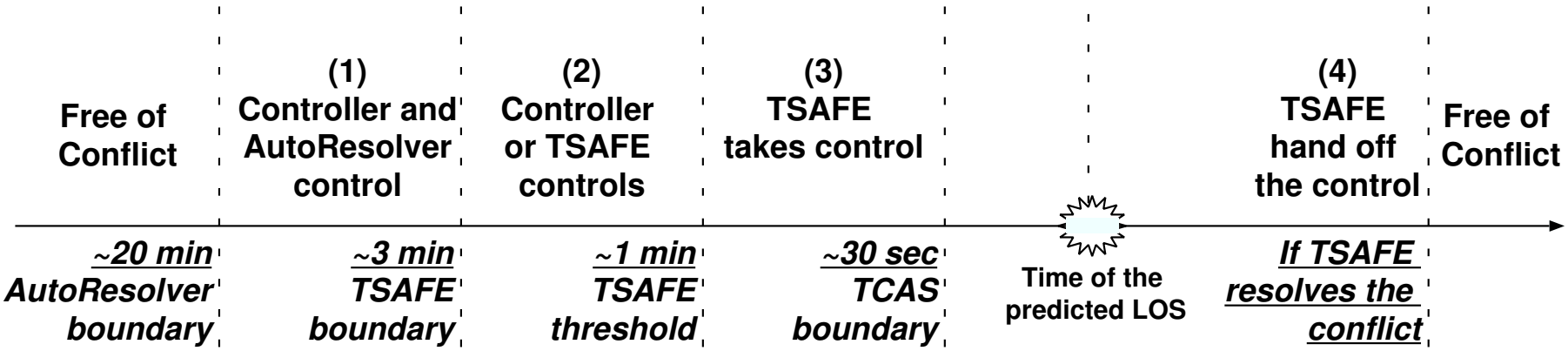
5th Workshop on Cooperative Software Verification (COOP)
April 7, 2024

AAC Operational Concept[1]



[1] H Erzberger, K Heere. "Algorithm and operational concept for resolving short-range conflicts." Proc. IMechE G J. Aerosp. Eng. 224 (2) (2010) 225–243.

AAC Operational Concept[2]



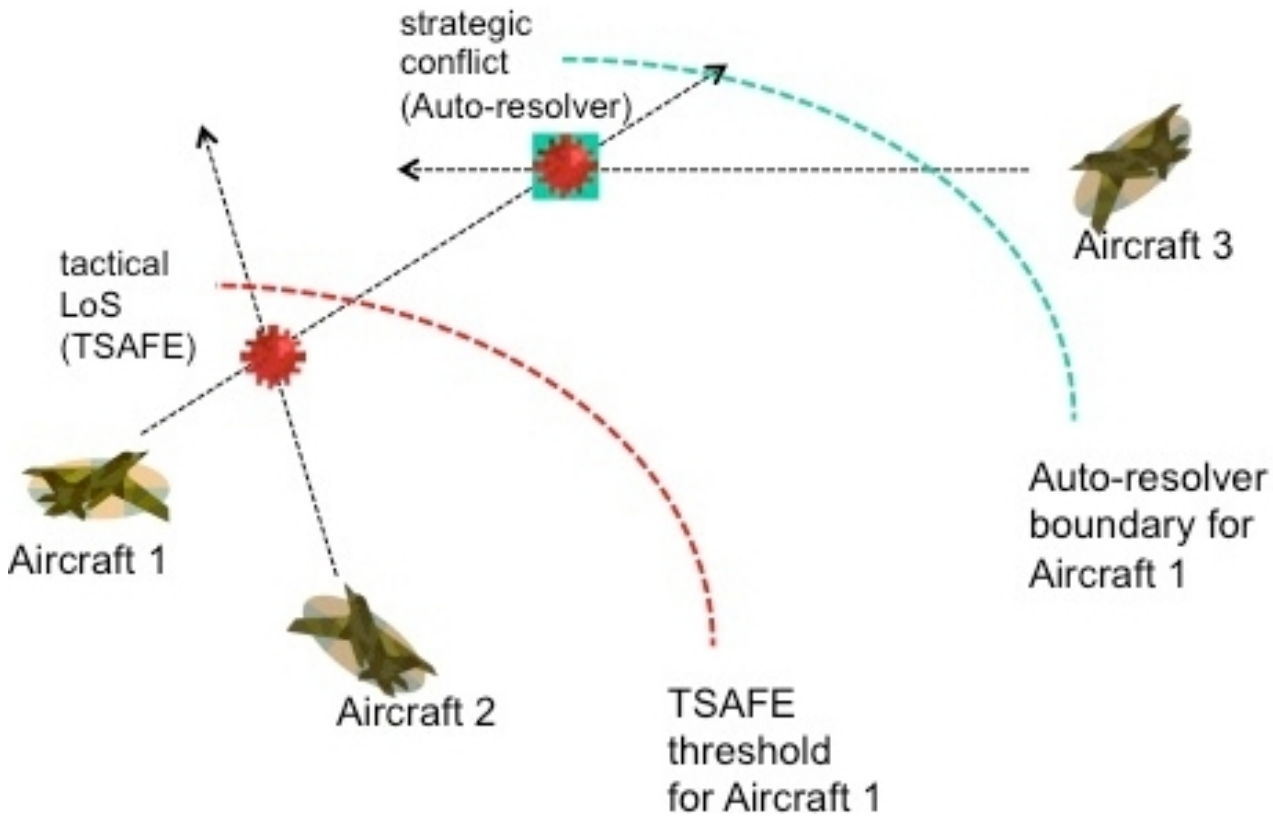
LTL Model Checking triggered system design changes[1]

[1] Y. Zhao and K.Y. Rozier. "Formal Specification and Verification of a Coordination Protocol for an Automated Air Traffic Control System." SCP Journal, vol-96, no-3, pg 337-353, 2014.

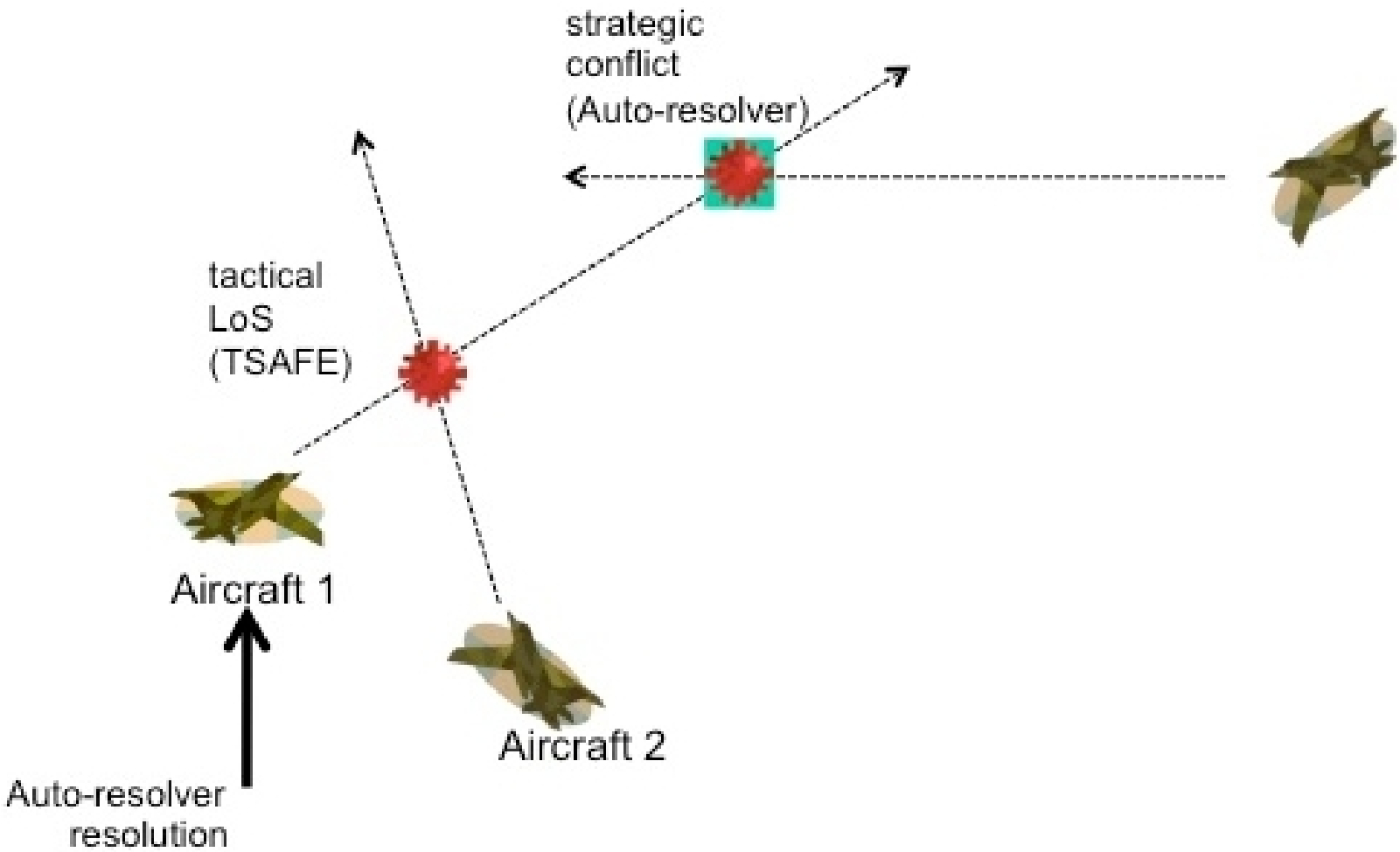
[2] H Erzberger, K Heere. "Algorithm and operational concept for resolving short-range conflicts." Proc. IMechE G J. Aerosp. Eng. 224 (2) (2010) 225–243.

Counterexample

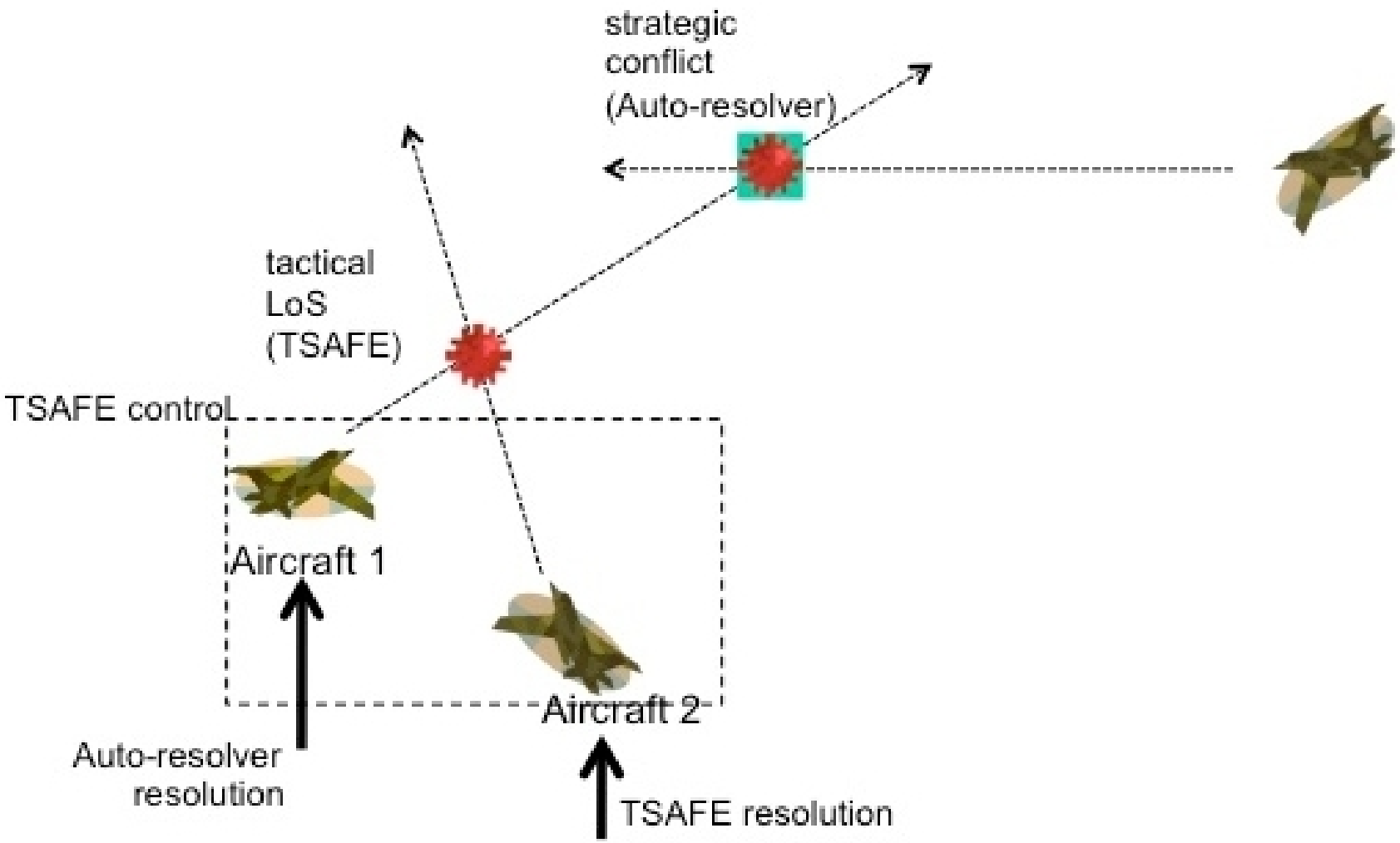
Specification: "If the controller hands off the control of an aircraft to TSAFE, this aircraft will not execute commands from the controller or Autoresolver."



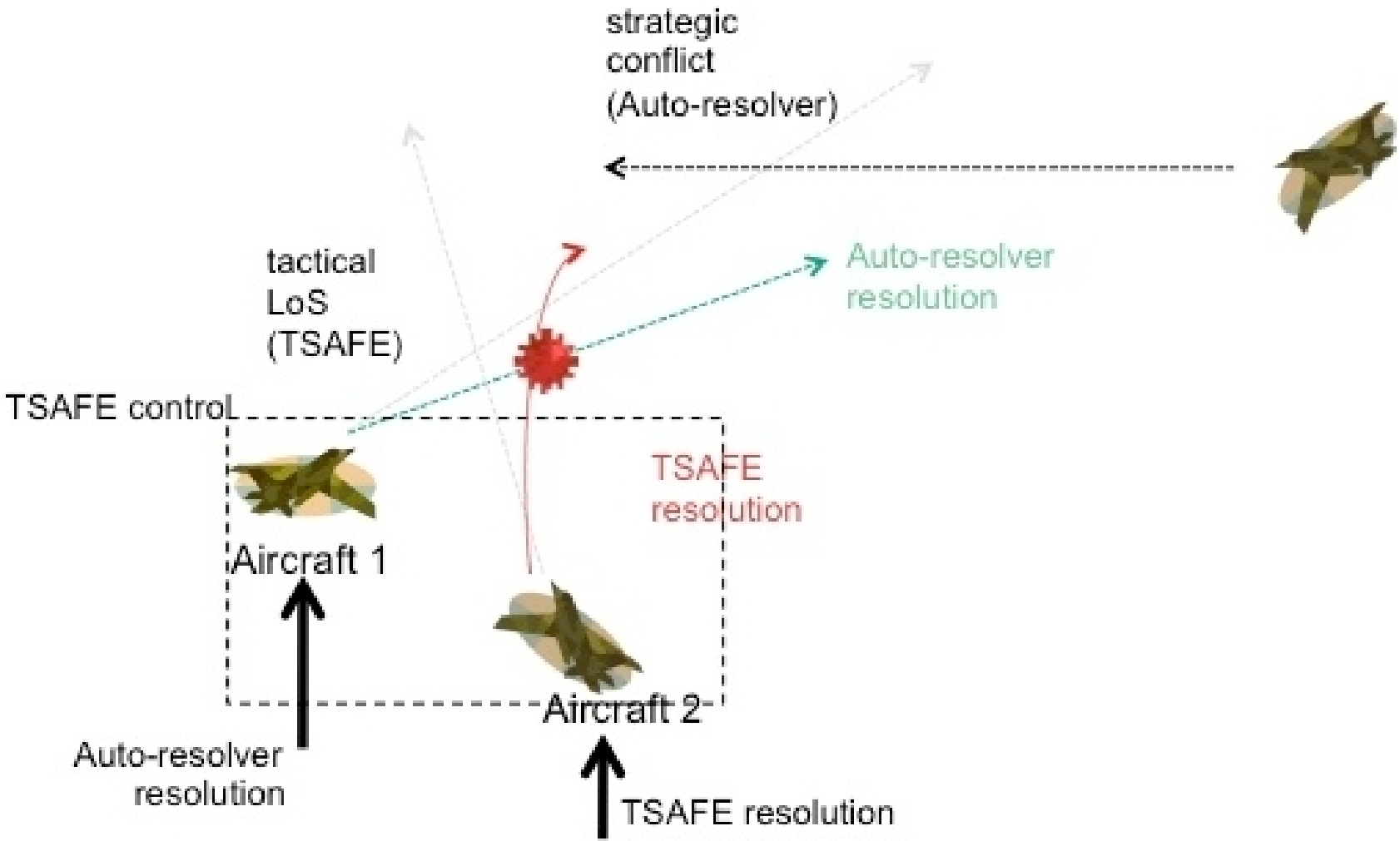
Counterexample



Counterexample

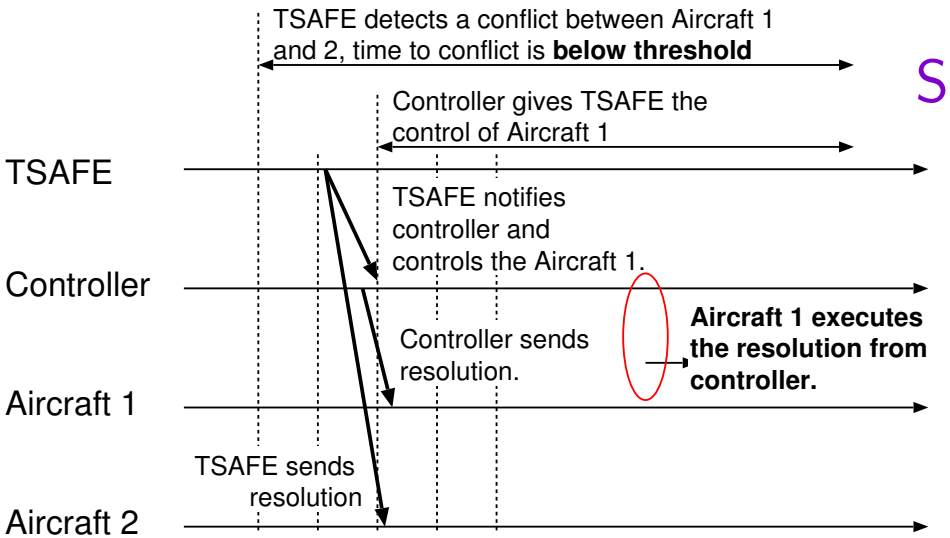


Counterexample



Counterexample: Fixed![3]

Specification: "If the controller hands off the control of an aircraft to TSAFE, this aircraft will not execute commands from the controller or Autoresolver."

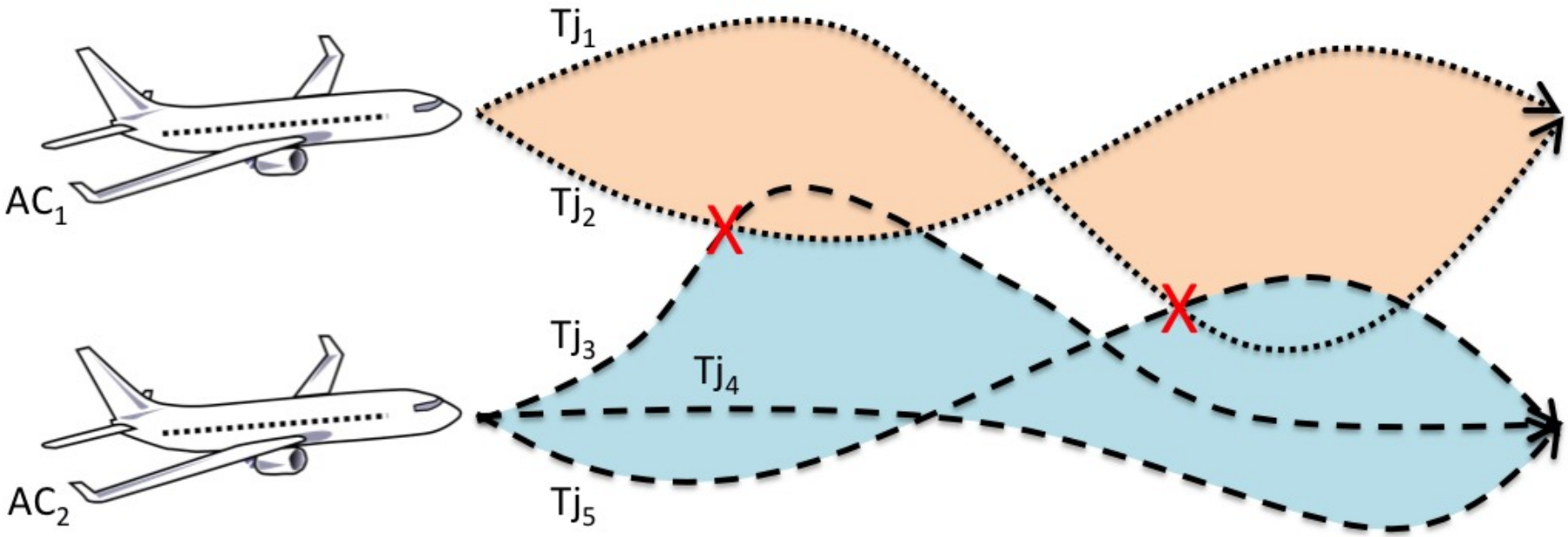


Solution:

- A1 receives notice of control transfer and "hold current route" resolution from TSAFE
- AR/controller's command will be superseded and ignored

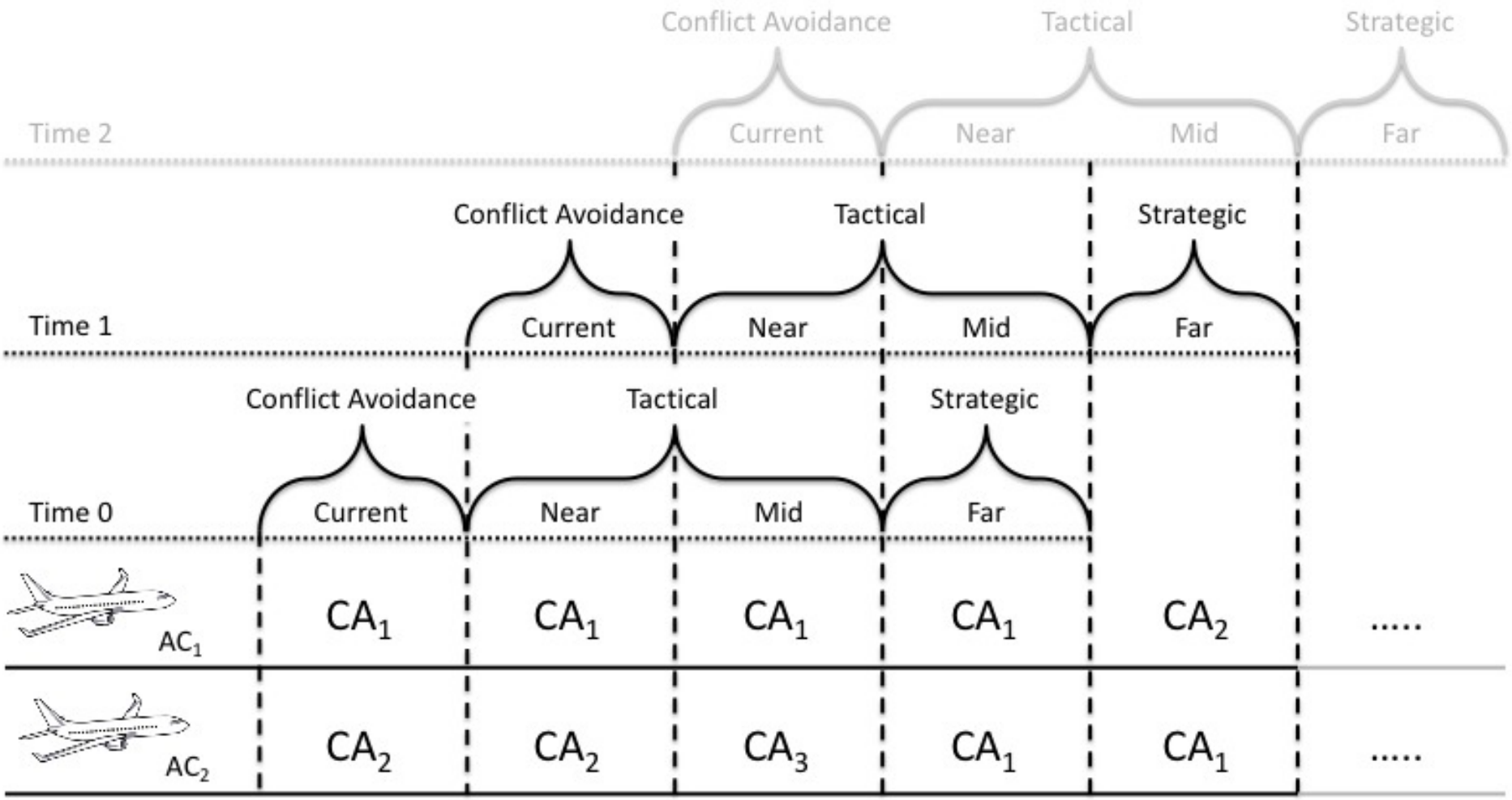
[3] Zhao, Yang, and Rozier, Kristin Yvonne. "Formal Specification and Verification of a Coordination Protocol for an Automated Air Traffic Control System." In AVoCS 2012.

Formal Modeling: Conflict Areas[4]

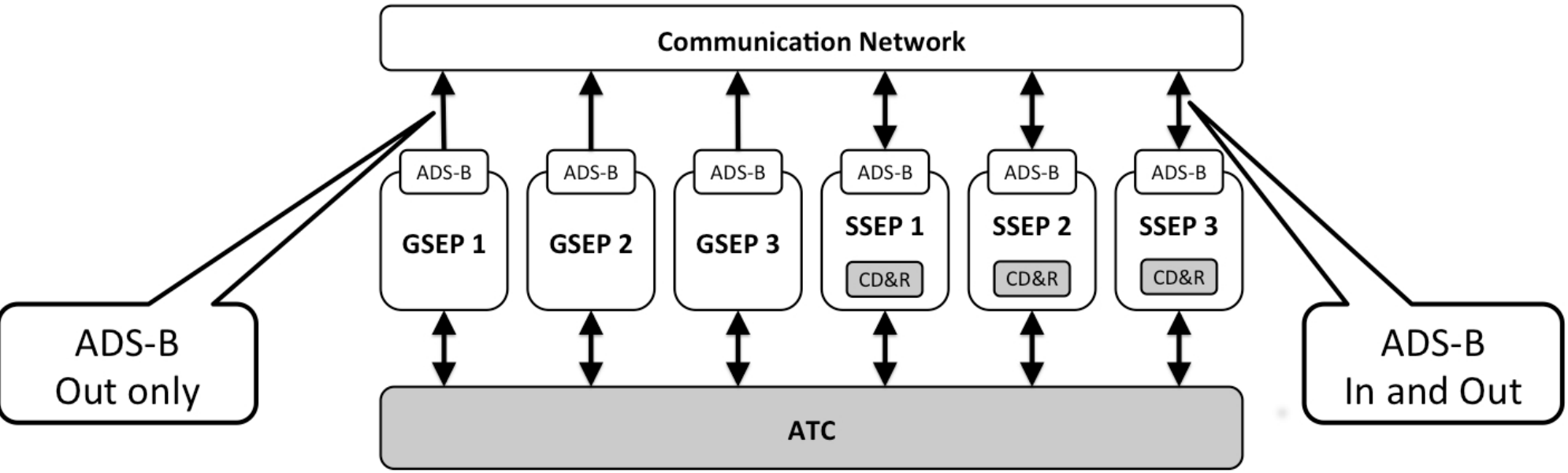


[4] Cristian Mattarei, Alessandro Cimatti, Marco Gario, Stefano Tonetta and Kristin Y. Rozier. "Comparing Different Functional Allocations in Automated Air Traffic Control Design." In Formal Methods in Computer-Aided Design (FMCAD), IEEE/ACM, 2015.

Formal Modeling: Time Windows



Formal Modeling: System Components (via OCRA Contracts)[5]



[5] 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.

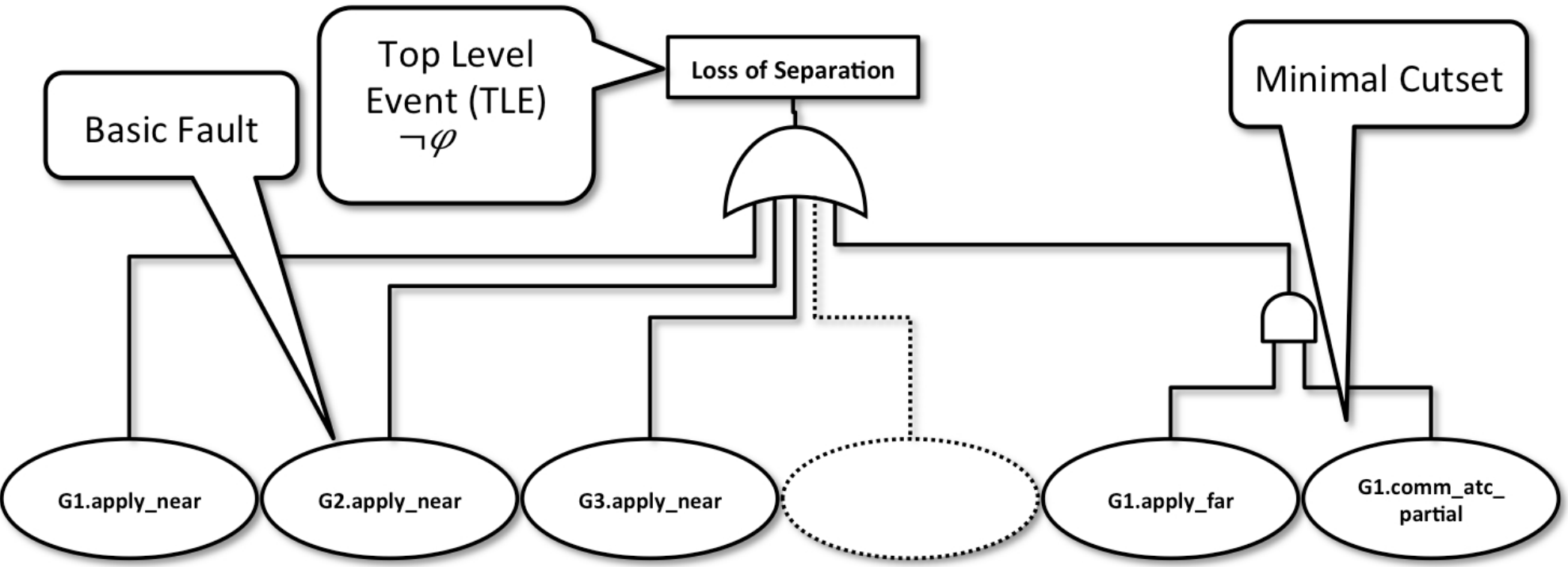
Full-Scale Design Space Verification[6]

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**

[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.

Fault Tree Analysis: xSAP[7]

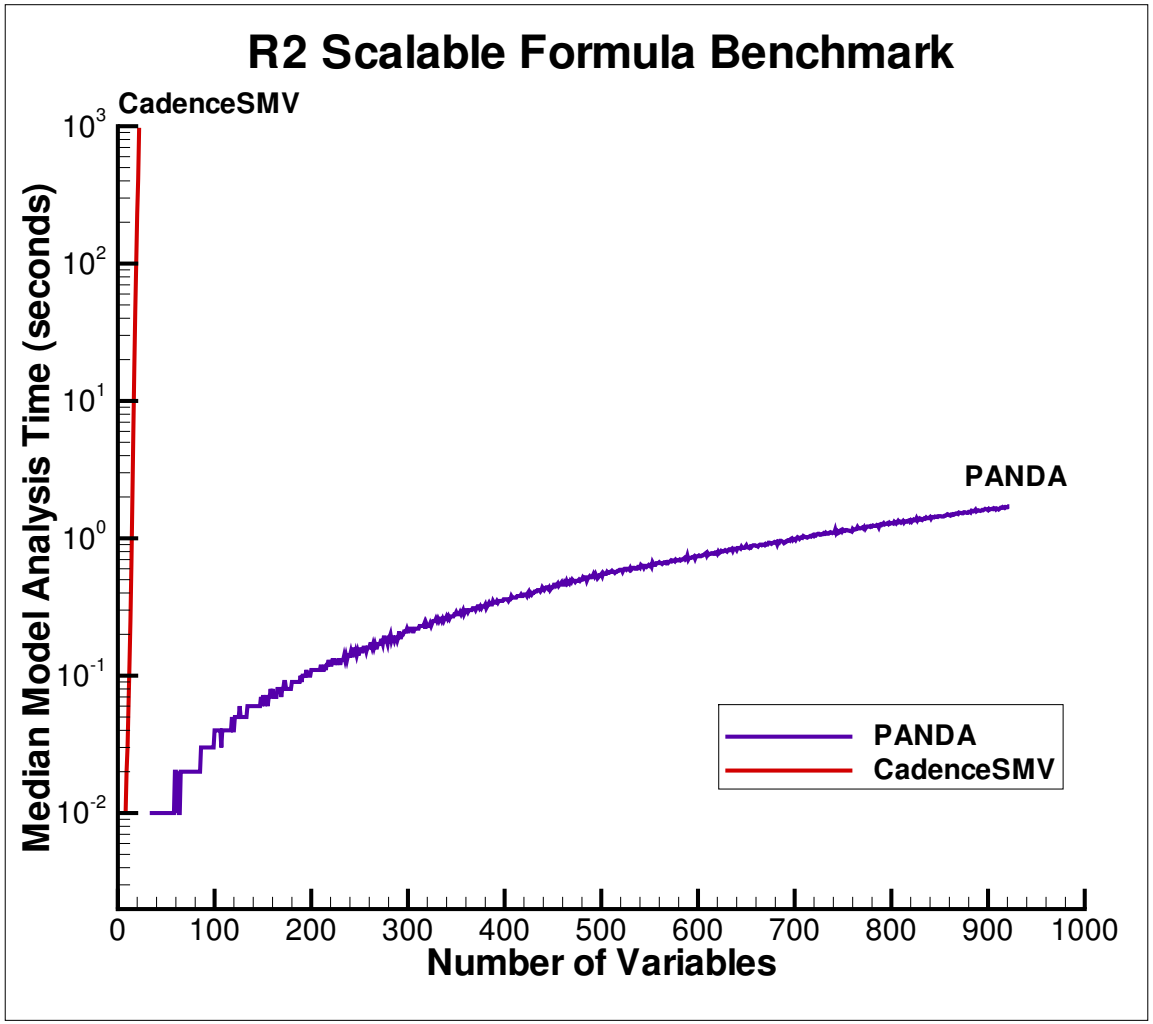


[7] Cristian Mattarei, Alessandro Cimatti, Marco Gario, Stefano Tonetta and Kristin Y. Rozier. "Comparing Different Functional Allocations in Automated Air Traffic Control Design." In Formal Methods in Computer-Aided Design (FMCAD), IEEE/ACM, 2015.

The Next Challenge

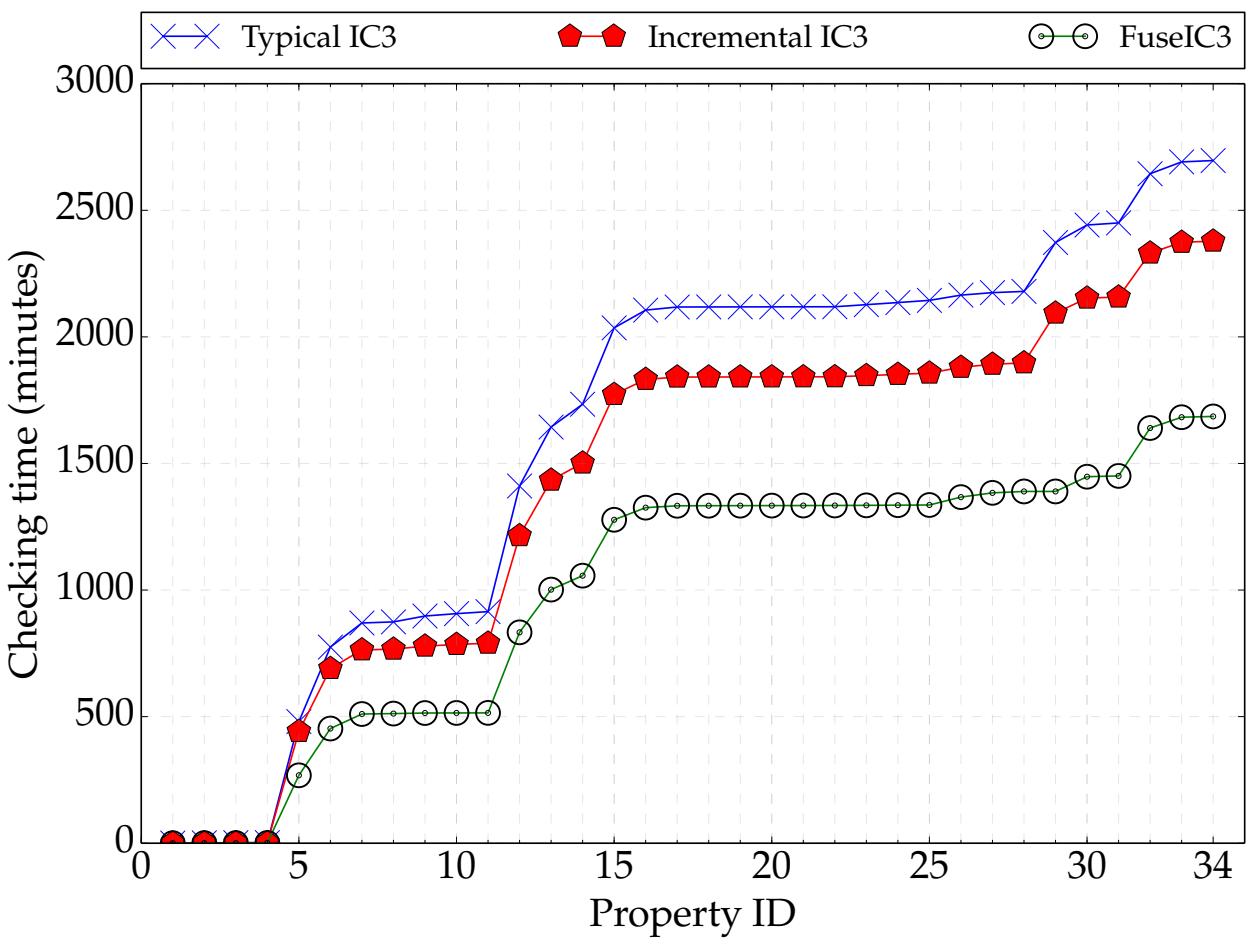
How do we do efficient design-space model checking?

Encoding Matters[8]



[8] 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[9]



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

[9] 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

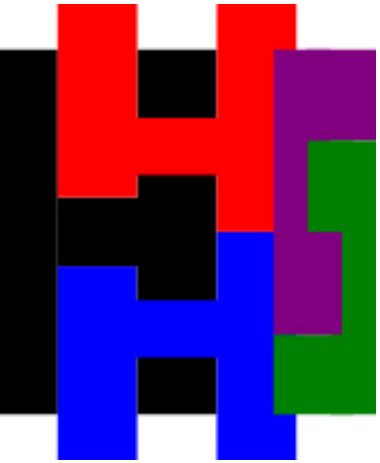
- nuXmv **closed source**: can't check internal algorithm
- IBM's tools **closed source**: same problem
- **Can't build** on either tool
- Have **models in SMV** language: can't MC with IBM's algorithm
- Have **new algorithm**; also can't check SMV models with it

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

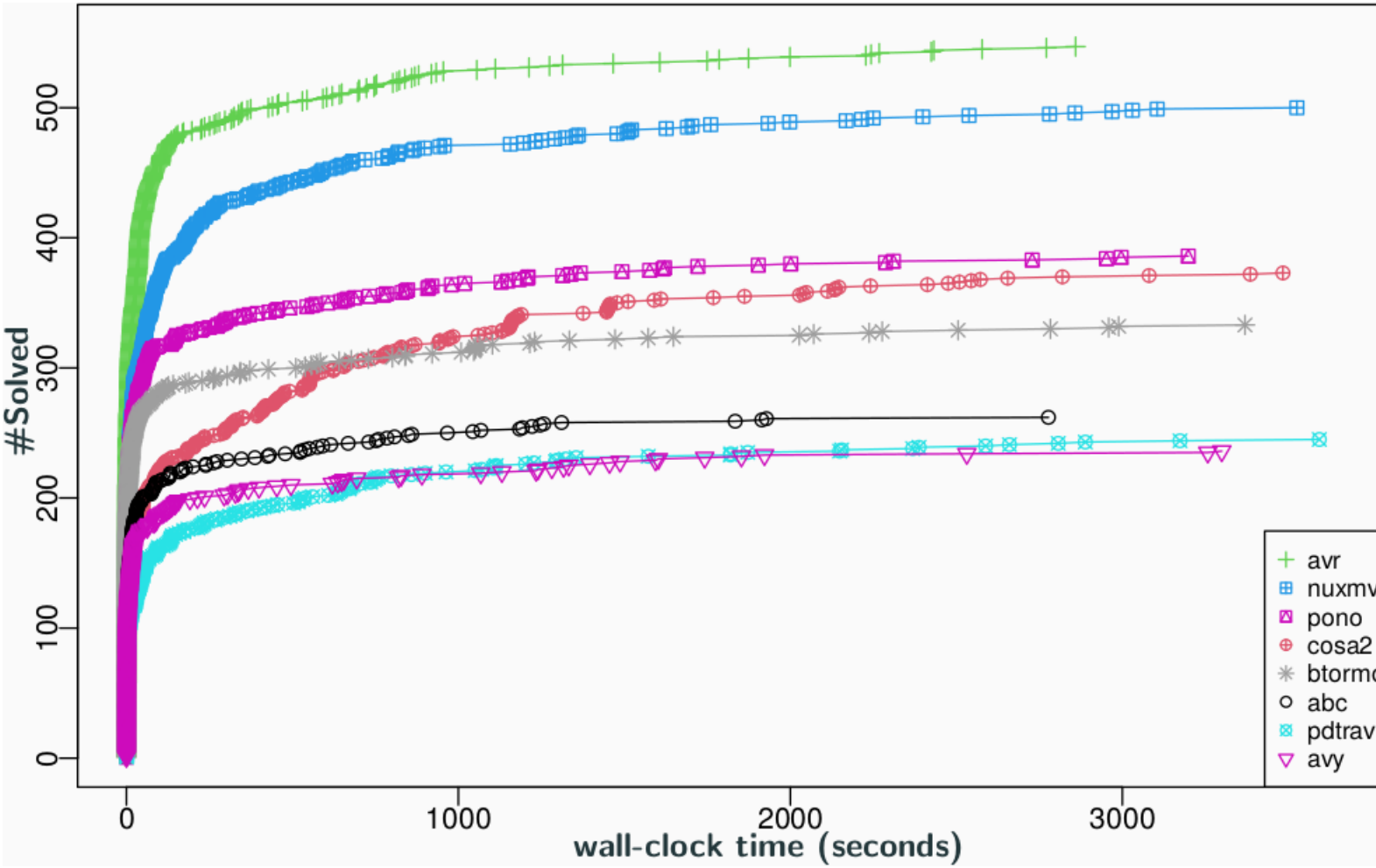
How do we do better?

HWMCC: Hardware Model Checking Competition (2020)



Word-level reasoning

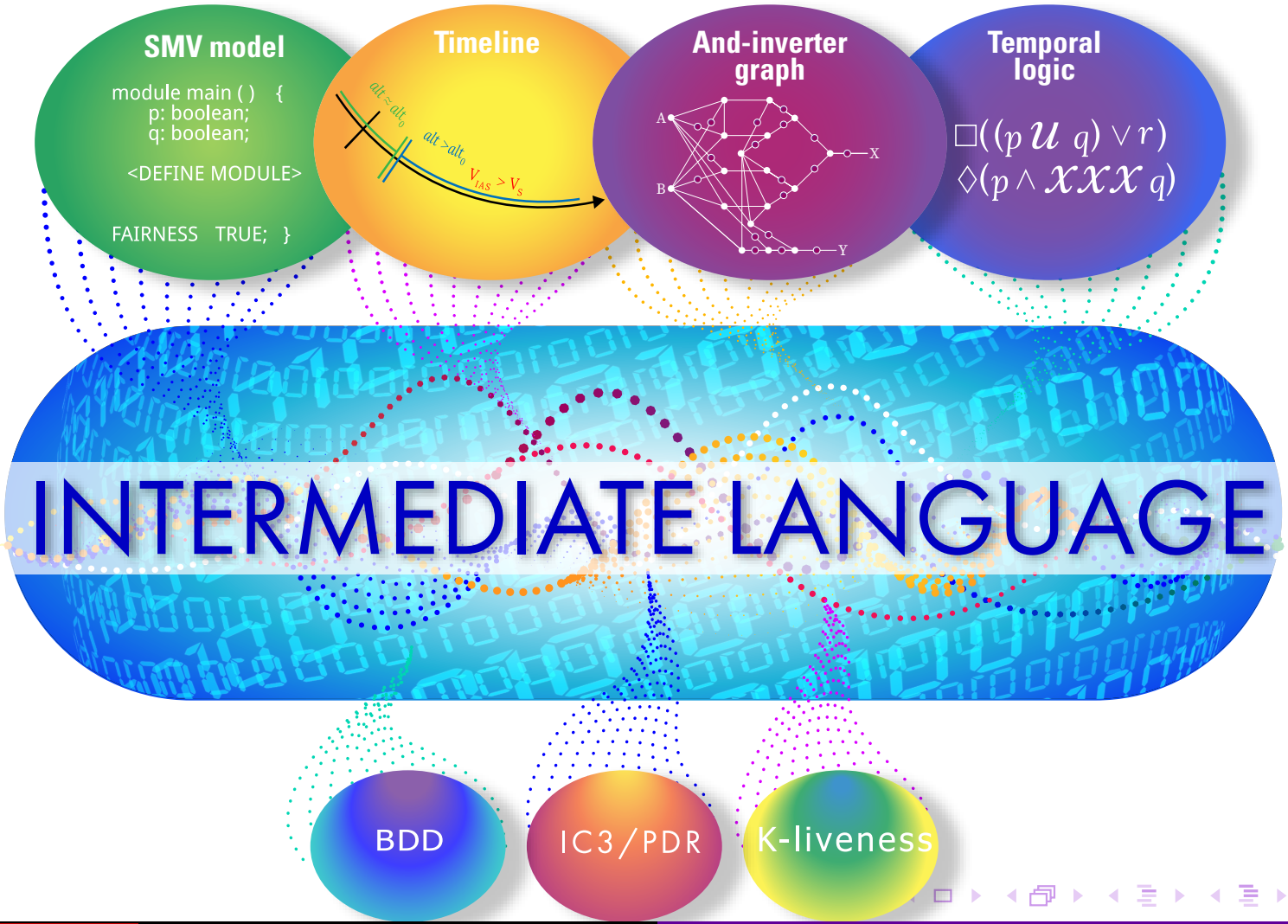
BTOR2



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**

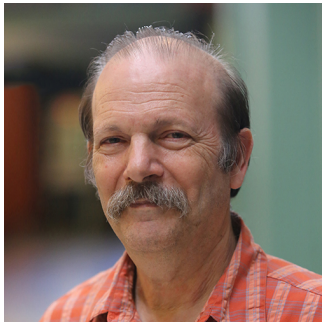
Core Design Team

Investigators:

K.Y. Rozier

Natarajan Shankar

Cesare Tinelli

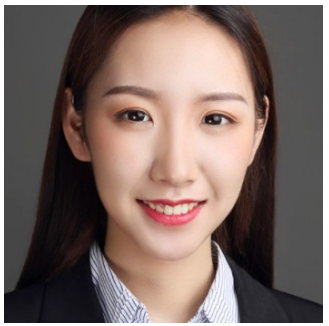


Moshe Vardi

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)

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

Summary: Model Checking Coordination Effort

The time has come for model-checking community standards

- **Participate: use MoXI**, **SPIN Keynote**, email list, language design feedback, community forums (OSSyM@CAV!)
- Available Now: **SMV** $\leftrightarrow$ **MoXI** $\leftrightarrow$ **BTOR2**
- **Contribute** future translators:
  - Your Modeling Language $\leftrightarrow$ MoXI
  - MoXI $\leftrightarrow$ Your Back-end MC Algorithm
- **Optimize! Expand! Compare! Research!**

modelchecker.temporallogic.org

SBMF Keynote: https://www.youtube.com/watch?v=XjkjVPOKVT8

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

The Next Challenge

What about Runtime Verification?

Shouldn't that be easier?

The Next Challenge

What about Runtime Verification?

Shouldn't that be easier?

Need an RV Benchmark Competition!

What is an RV Benchmark?

A Simple 3-Tuple:

1. Input Stream
2. TL Formula: **MLTL**
3. Output Stream

Quickly gets complicated . . . gold stars for these?

- Sampling? Sensor filters?
- Real-time performance?
- Integrating Hardware/Software?

What are we benchmarking?

- Correctness!
- speed/timing?
- other metrics?

Need an RV Benchmark Competition!

What is an RV Benchmark?

A Simple 3-Tuple:

1. Input Stream
2. TL Formula: **MLTL**
3. Output Stream

Quickly gets complicated . . . gold stars for these?

- Sampling? Sensor filters?
- Real-time performance?
- Integrating Hardware/Software?

What are we benchmarking?

- Correctness!
- speed/timing?
- other metrics?

Need an RV Benchmark Competition!

What is an RV Benchmark?

A Simple 3-Tuple:

1. Input Stream
2. TL Formula: **MLTL**
3. Output Stream

Quickly gets complicated . . . gold stars for these?

- Sampling? Sensor filters?
- Real-time performance?
- Integrating Hardware/Software?

What are we benchmarking?

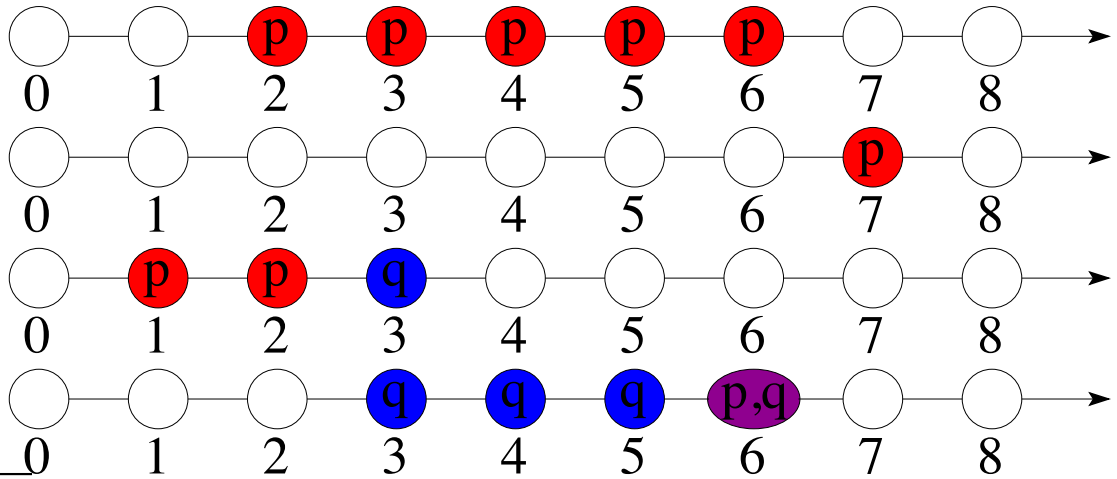
- Correctness!
- speed/timing?
- other metrics?

MLTL: A Good Specification Language[10]

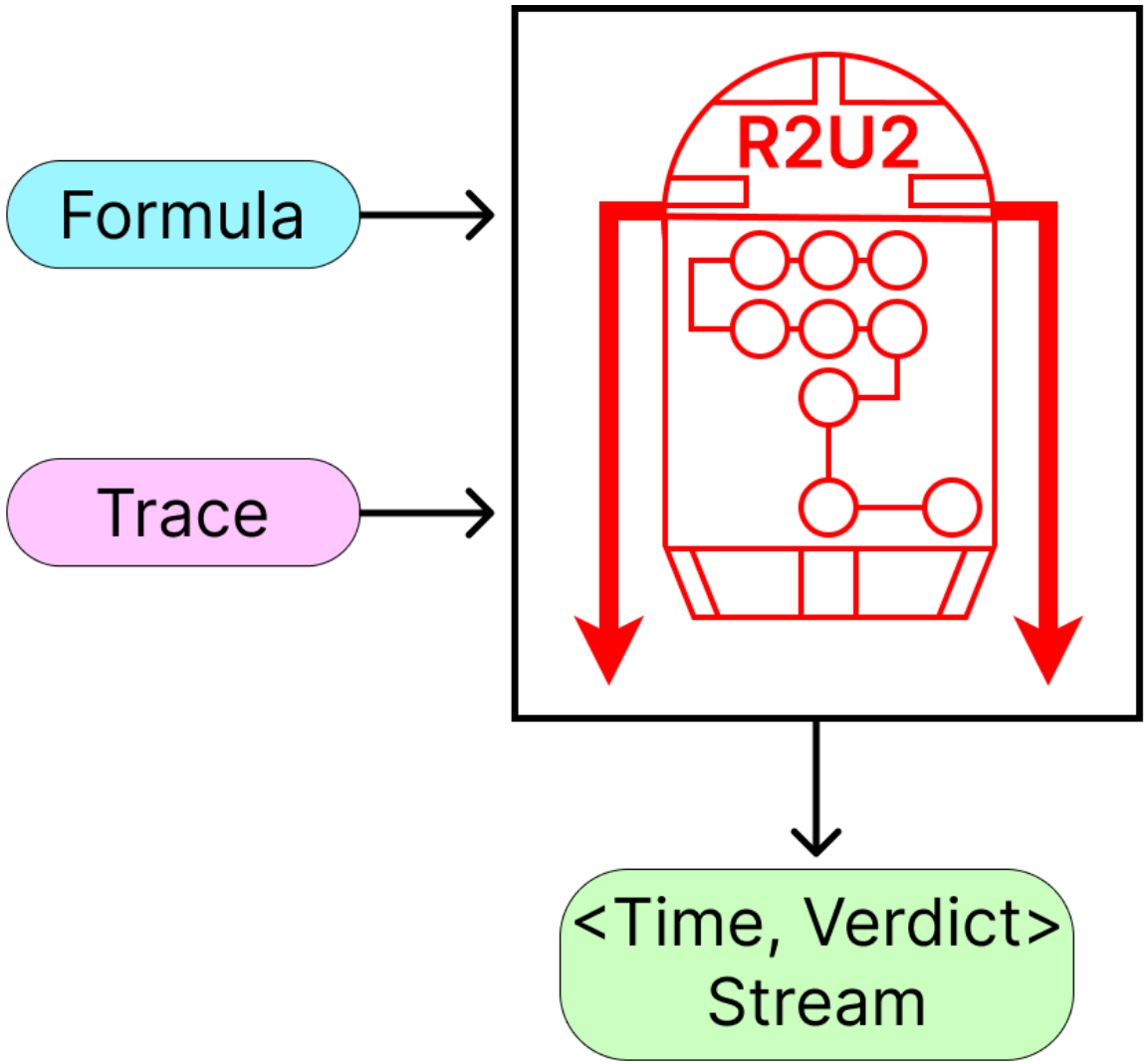
Mission-Time Temporal Logic (MLTL) reasons about *integer-bounded* timelines:

- finite set of atomic propositions {p q}
- Boolean connectives: $\neg$, $\wedge$, $\vee$, and $\rightarrow$
- temporal connectives *with time bounds*:

Symbol	Operator	Timeline
$\Box_{[2,6]} p$	$\text{ALWAYS}_{[2,6]}$	
$\Diamond_{[0,7]} p$	$\text{EVENTUALLY}_{[0,7]}$	
$p \mathcal{U}_{[1,5]} q$	$\text{UNTIL}_{[1,5]}$	
$p \mathcal{R}_{[3,8]} q$	$\text{RELEASE}_{[3,8]}$	



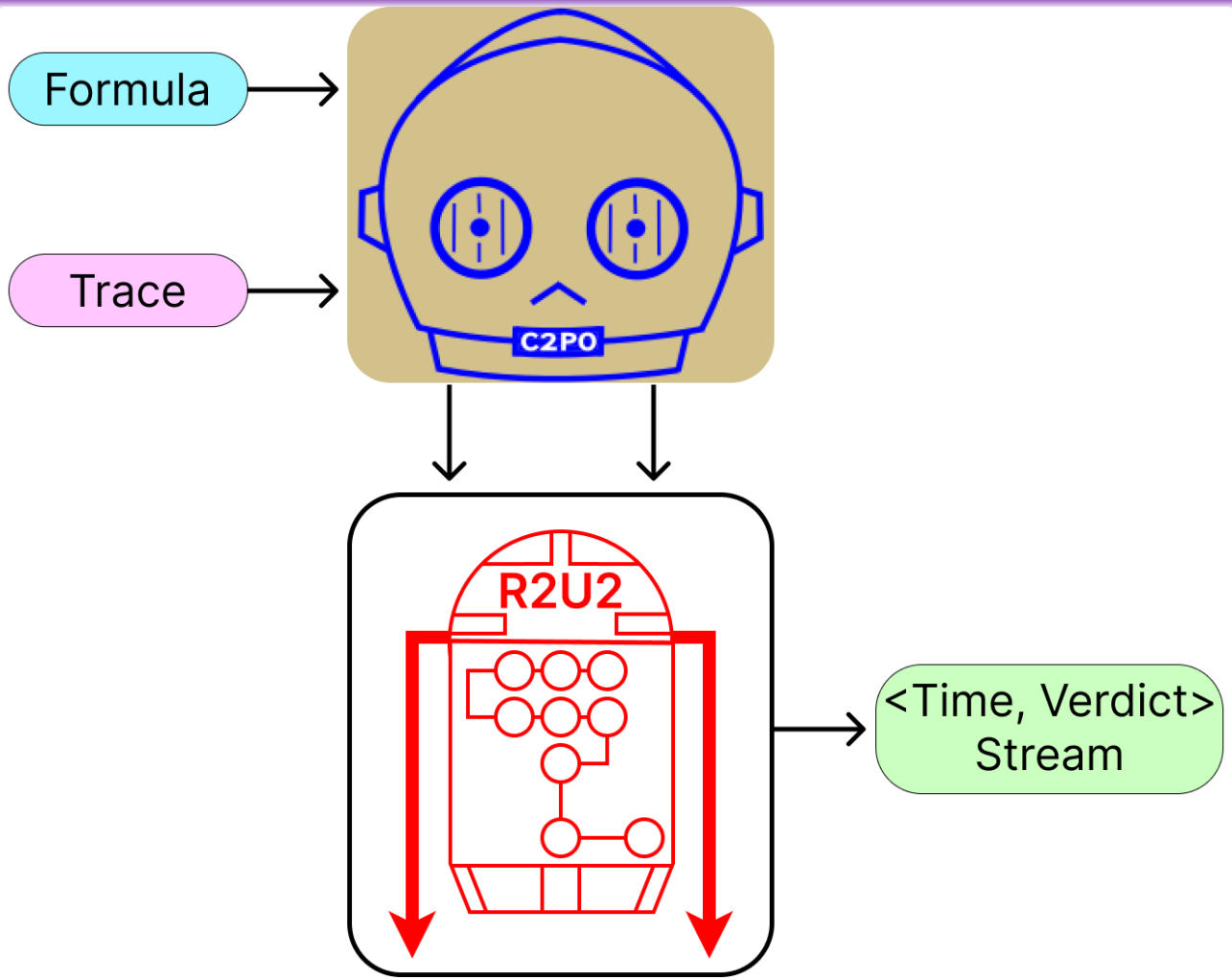
[10] T. Reinbacher, K.Y. Rozier, J. Schumann. "Temporal-Logic Based Runtime Observer Pairs for System Health Management of Real-Time Systems." TACAS 2014.



11 12

[11] K. Y. Rozier, J. Schumann. "R2U2: Tool Overview." RV-CUBES, 2017.

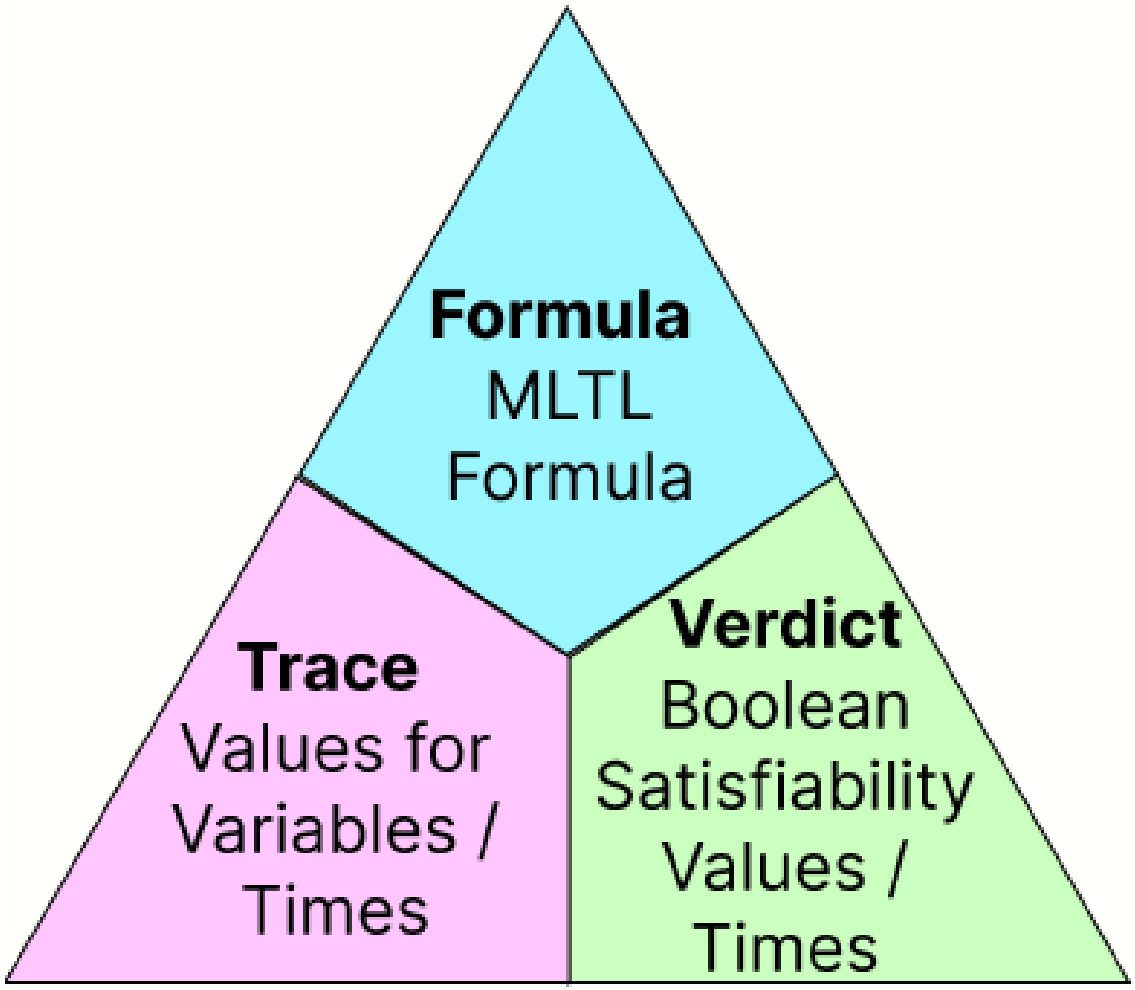
[12] T. Reinbacher, K. Y. Rozier, J. Schumann. "Temporal-Logic Based Runtime Observer Pairs for System Health Management of Real-Time Systems." TACAS, 2014.



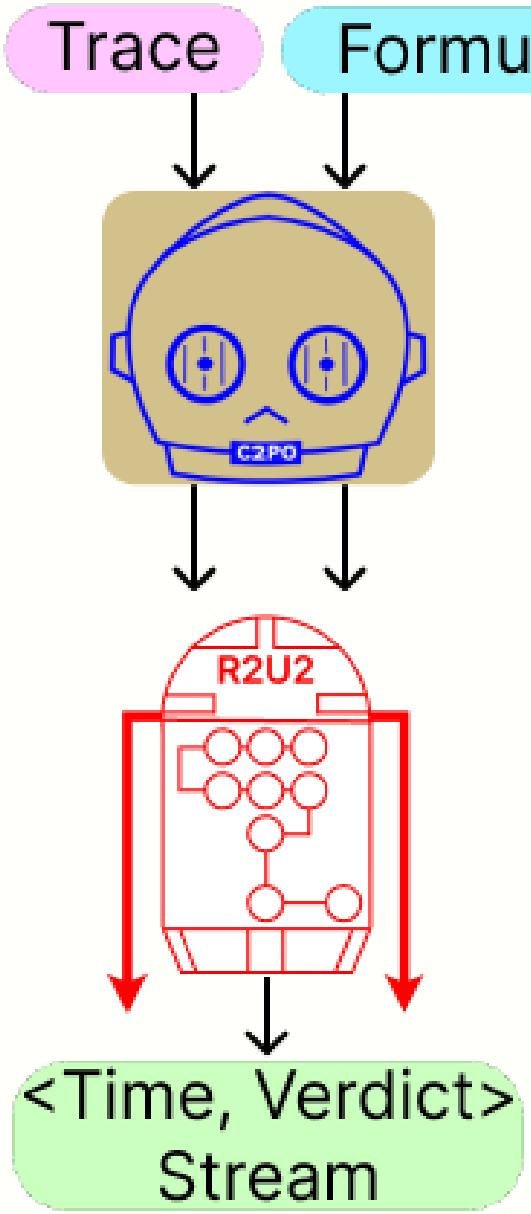
13 14

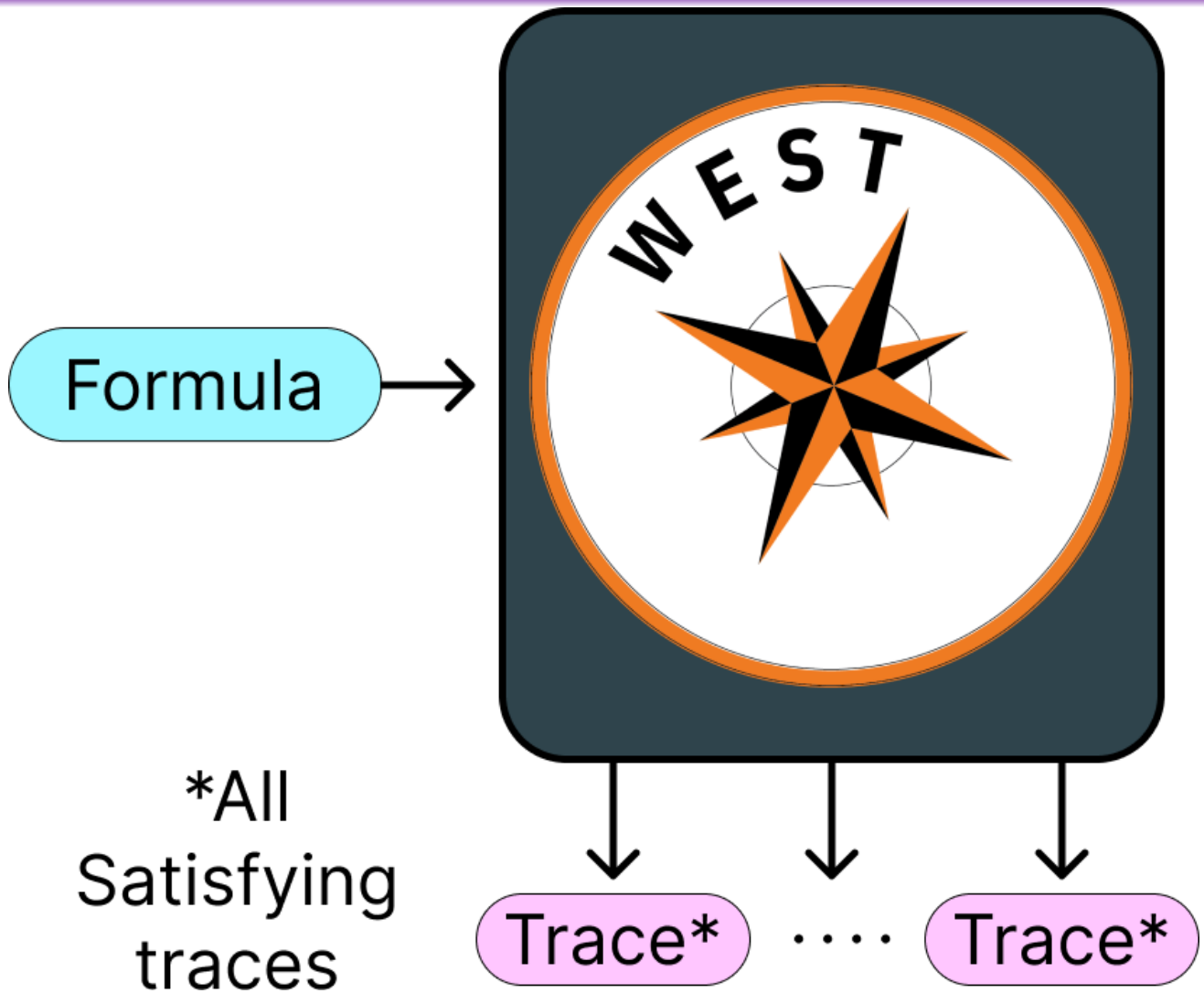
[13] C. Johannsen, P. H.Jones, B. Kempa, K. Y. Rozier, P. Zhang. "R2U2 Version 3.0: Re-imagining a Toolchain for Specification, Resource Estimation, and Optimized Observer Generation for Runtime Verification in Hardware and Software." CAV, 2023.

[14] C. Johannsen, B. Kempa, P. H. Jones, K. Y. Rozier, T. Wongpiromsarn. "Impossible Made Possible: Encoding Intractable Specifications via Implied Domain Constraints." FMICS, 2023.



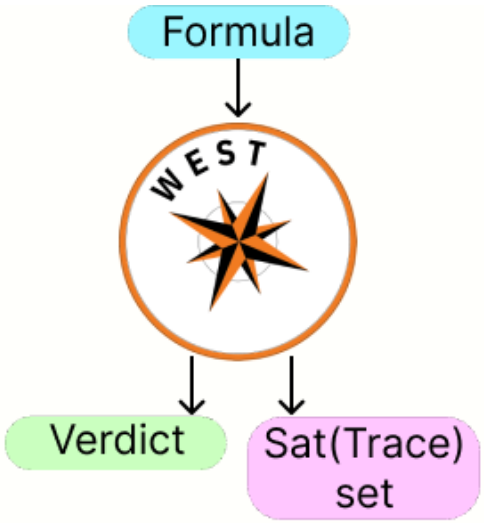
Given any 2 you can find the 3rd

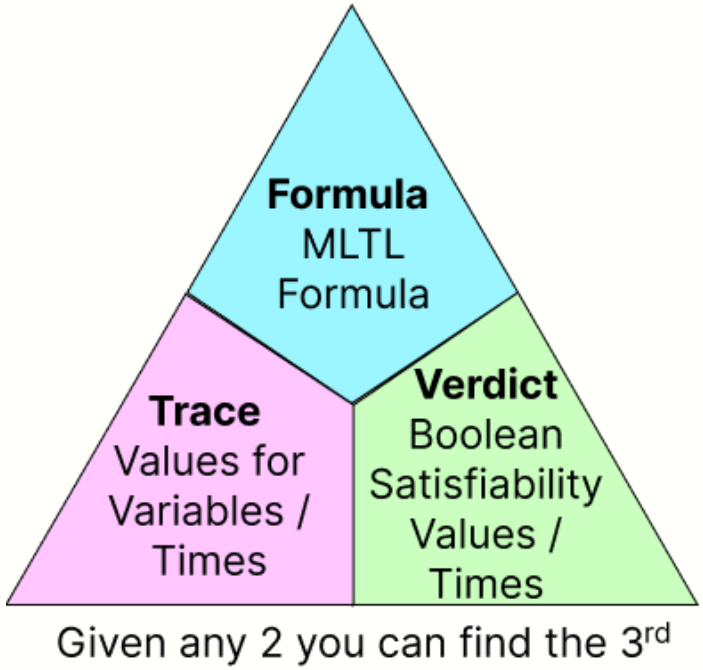


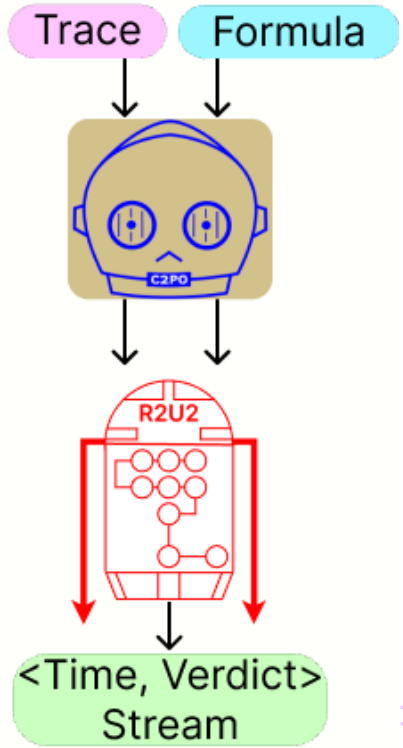


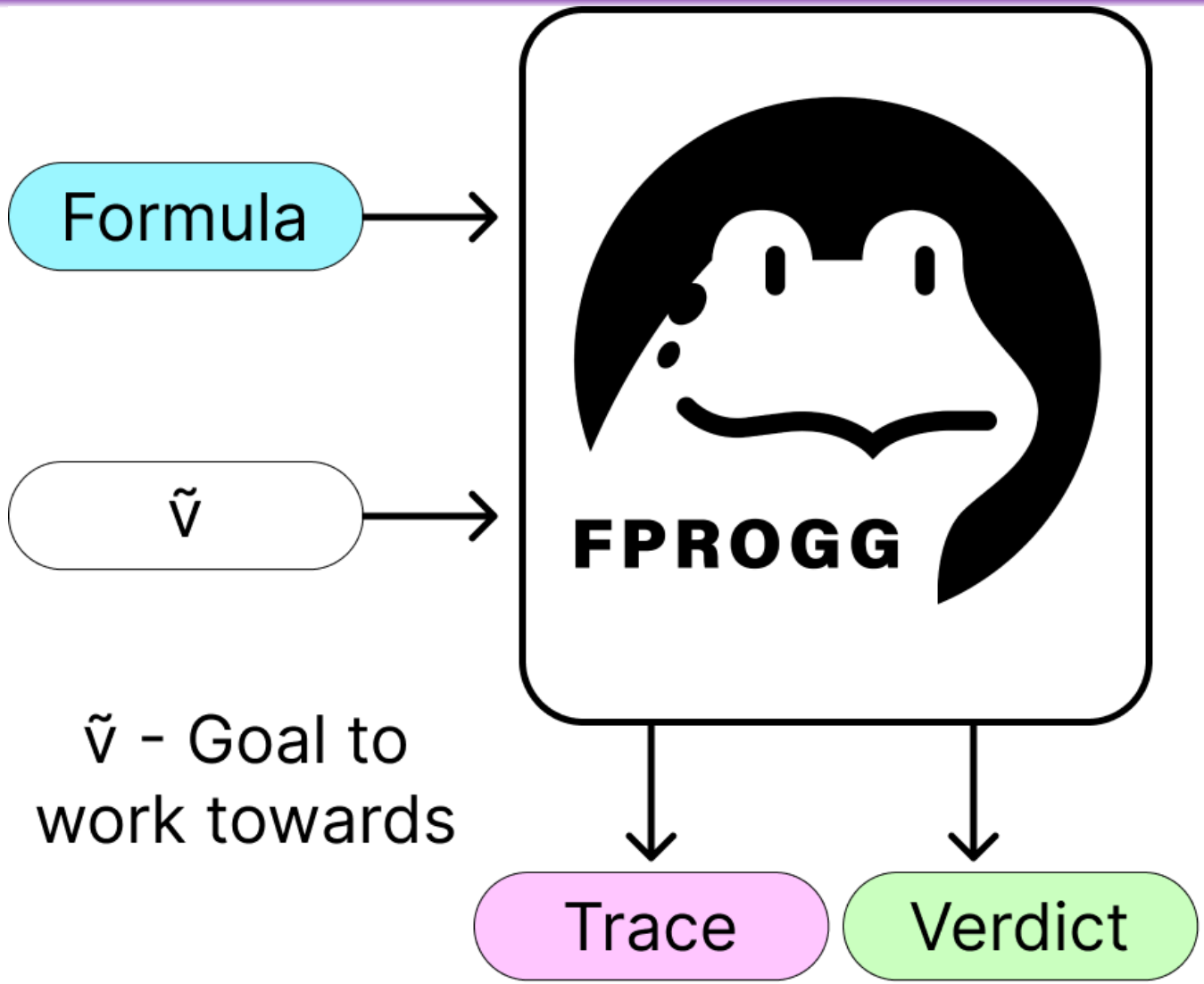
15

[15] J. Elwing, L. Gamboa-Guzman, J. Sorkin, C. Travesset, Z. Wang, K. Y. Rozier. "Mission-Time LTL (MLTL) Formula Validation via Regular Expressions." In *integrated Formal Methods* (iFM), 2023.


Formula
WEST
Verdict
Sat(Trace) set

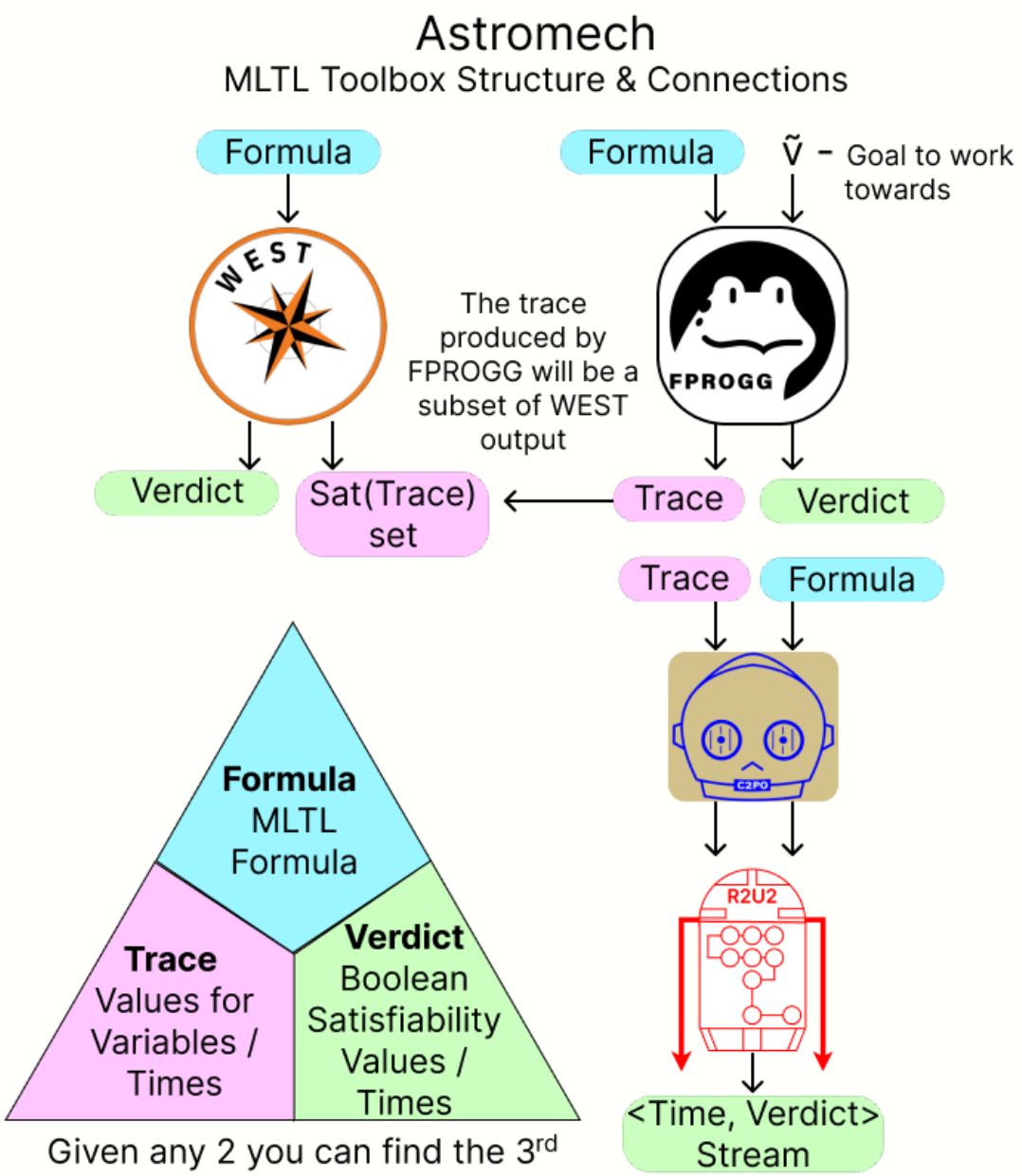

Formula
MLTL
Formula
Trace
Values for
Variables /
Times
Verdict
Boolean
Satisfiability
Values /
Times
Given any 2 you can find the 3rd

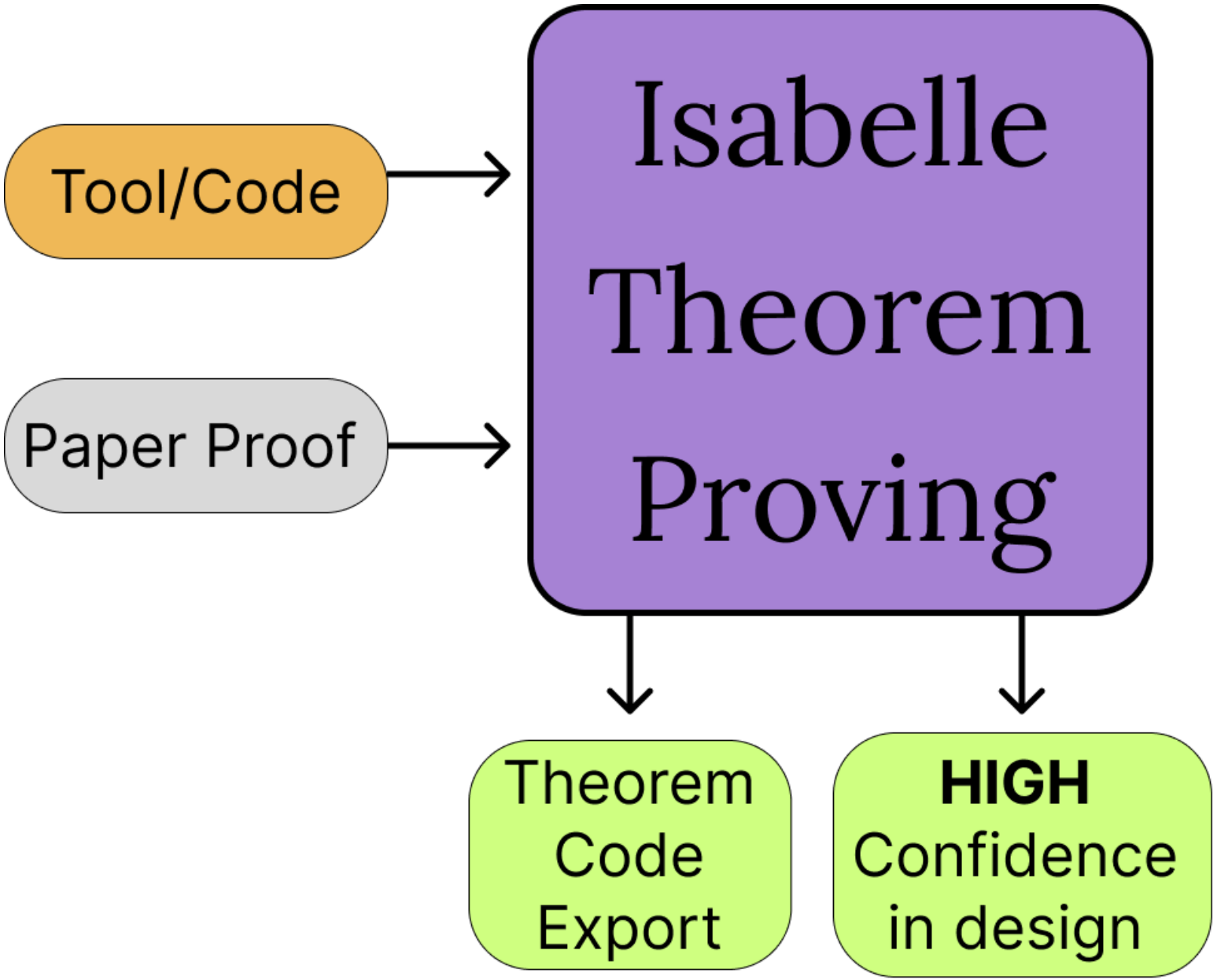

Trace
Formula
C2PO
R2U2
<Time, Verdict>
Stream



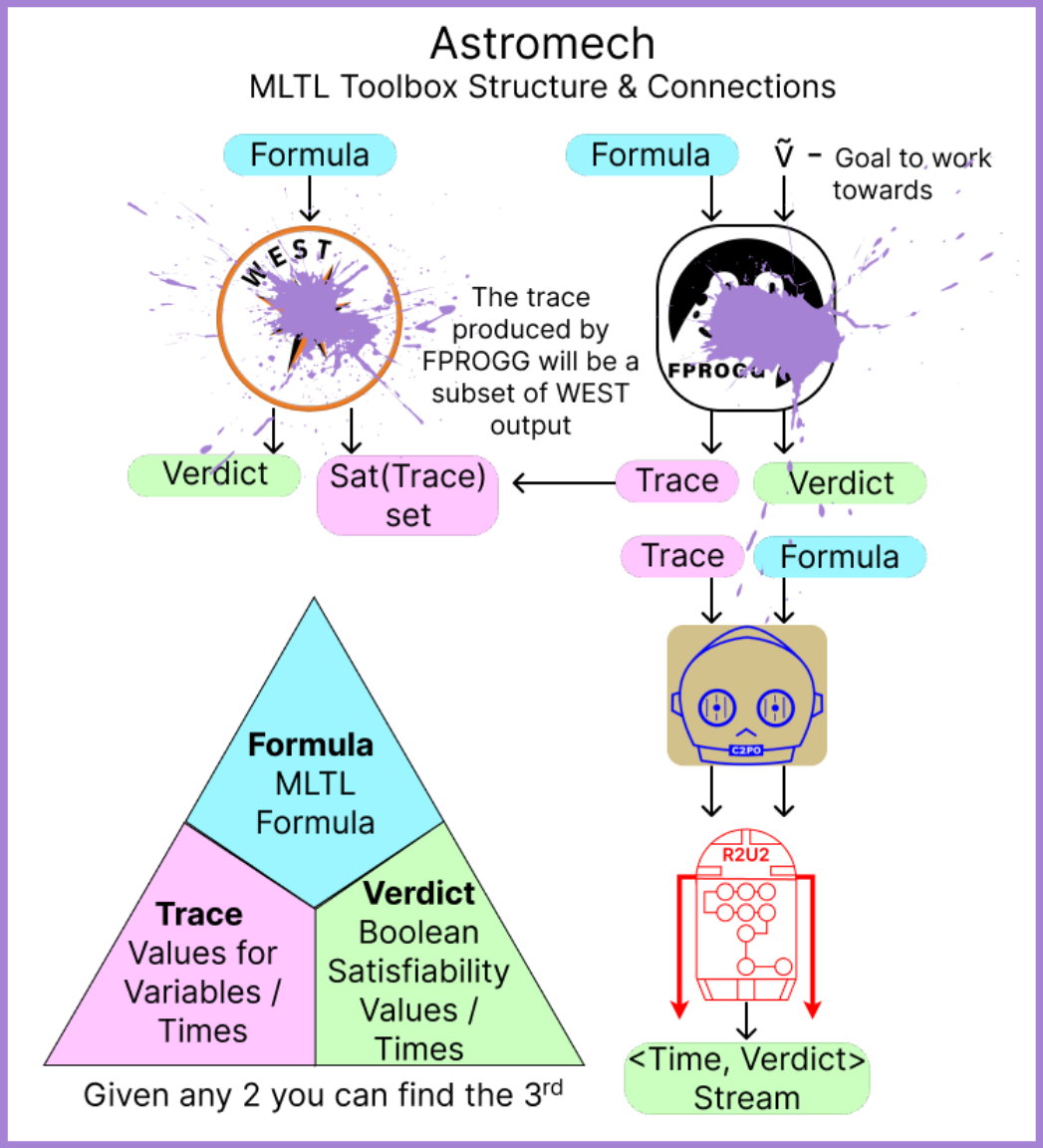
16

[16] A. Rosentrater, K. Y. Rozier, "FPROGG: A Formula Progression-Based MLTL Benchmark Generator." Under submission, 2024.


Astromech
MLTL Toolbox Structure & Connections
Formula
Formula
$\tilde{V}$ – Goal to work towards
WEST
FPROGG
The trace produced by FPROGG will be a subset of WEST output
Verdict
Sat(Trace) set
Trace
Verdict
Trace
Formula
C2PO
R2U2
<Time, Verdict> Stream
Formula MLTL Formula
Trace Values for Variables / Times
Verdict Boolean Satisfiability Values / Times
Given any 2 you can find the 3rd


Tool/Code
Paper Proof
Isabelle Theorem Proving
Theorem Code Export
HIGH Confidence in design

Isabelle Theorem Proving



Tool Integration Challenges & Opportunities

Center on input logics (e.g., MLTL)

- **FRET** integrates MLTL [17]
- **nuXmv** integrates MLTL[18]
- FRET → **Ogma**[19] → **Copilot**[20]
- **Pacti**[21] ↔ R2U2?

---

[17] D. Giannakopoulou, A. Mavridou, J. Rhein, T. Pressburger, J. Schumann, N. Shi. "Formal requirements elicitation with FRET." (REFSQ-2020).

[18] nuXmv 1.1.0 (2016-05-10) Release Notes: `https://es-static.fbk.eu/tools/nuxmv/downloads/NEWS.txt`.

[19] I. Perez, A. Mavridou, T. Pressburger, A. Goodloe, D. Giannakopoulou. "Automated translation of natural language requirements to runtime monitors." TACAS, 2022.

[20] I. Perez, F. Dedden, A. Goodloe. "Copilot 3." No. NF1676L-35996. 2020.

[21] I. Incer, A. Badithela, J. Graebener, P. Mallozzi, A. Pandey, S. Yu, A. Benveniste et al. "Pacti: Scaling assume-guarantee reasoning for system analysis and design." arXiv preprint arXiv:2303.17751 (2023).

Challenges

- Integration of different verification methods into **one model**

Challenges

- Integration of different verification methods into **one model**
  - FBK example: nuXmv model checker + OCRA contracts + xSAP fault tree generator

Challenges

- Integration of different verification methods into **one model**
  - FBK example: nuXmv model checker + OCRA contracts + xSAP fault tree generator
  - How to do this with other model checking input languages?

Challenges

- Integration of different verification methods into **one model**
  - FBK example: nuXmv model checker + OCRA contracts + xSAP fault tree generator
  - How to do this with other model checking input languages?
  - OCRA/xSAP **instrument SMV input language**
  - **Standard formats** for components like model-checking contracts, fault tree analysis?

Challenges

- Integration of different verification methods into **one model**
  - FBK example: nuXmv model checker + OCRA contracts + xSAP fault tree generator
  - How to do this with other model checking input languages?
  - OCRA/xSAP **instrument SMV input language**
  - **Standard formats** for components like model-checking contracts, fault tree analysis?
- How to **agree in the community** on standard **logics**, **intermediate languages**, **output/counterexample formats**?
  - RV competition fail
  - MoXI success (hopefully)

Challenges

- Integration of different verification methods into **one model**
  - FBK example: nuXmv model checker + OCRA contracts + xSAP fault tree generator
  - How to do this with other model checking input languages?
  - OCRA/xSAP **instrument SMV input language**
  - **Standard formats** for components like model-checking contracts, fault tree analysis?
- How to **agree in the community** on standard **logics**, **intermediate languages**, **output/counterexample formats**?
  - RV competition fail
  - MoXI success (hopefully)
- How to **catalog tool connections**: when using one tool, **automatically see** what other tools work with it? Giant GitHub page?

Opportunities

- **Specification logic as a common interface**
  - Via specification **validation tools** (e.g., WEST)
  - Via **test-case generators** (e.g., FPROGG)

[22] K. Y. Rozier, M. Y. Vardi. "LTL satisfiability checking." SPIN, 2007.

Opportunities

- **Specification logic as a common interface**
  - Via specification **validation tools** (e.g., WEST)
  - Via **test-case generators** (e.g., FPROGG)
- Overcoming **cultural barriers to integration**
  - 2007: LTL SAT as a front end to every model checker[22] . . . still outstanding

[22] K. Y. Rozier, M. Y. Vardi. "LTL satisfiability checking." SPIN, 2007.

Opportunities

- **Specification logic as a common interface**
  - Via specification **validation tools** (e.g., WEST)
  - Via **test-case generators** (e.g., FPROGG)
- Overcoming **cultural barriers to integration**
  - 2007: LTL SAT as a front end to every model checker[22] . . . still outstanding
- Propose **standard formats** where they aren't:
  - MC counterexamples
  - RV output (stream-based)
  - Offline RV intermediate language? (Online won't work. . .)
  - So many more standards possible!

[22] K. Y. Rozier, M. Y. Vardi. "LTL satisfiability checking." SPIN, 2007.

Opportunities

- **Specification logic as a common interface**
  - Via specification **validation tools** (e.g., WEST)
  - Via **test-case generators** (e.g., FPROGG)
- Overcoming **cultural barriers to integration**
  - 2007: LTL SAT as a front end to every model checker[22] . . . still outstanding
- Propose **standard formats** where they aren't:
  - MC counterexamples
  - RV output (stream-based)
  - Offline RV intermediate language? (Online won't work. . .)
  - So many more standards possible!
- build model checkers for **MoXI**!

[22] K. Y. Rozier, M. Y. Vardi. "LTL satisfiability checking." SPIN, 2007.

Tool Integration and Cooperation is Key to Making Formal Methods Universal!

- **International Community Standards:** specification logics, intermediate languages, output formats
- **Documentation & Interfaces**, e.g., for validation
- **Integration Index**: GitHub? How to do this?
- **Open-Source Standards**: methods to build on state of the art without replication

laboratory.temporallogic.org