Enabling FDIR design through diagnosability and recoverability analysis

Benjamin Bittner

bittner@fbk.eu

University of Trento / Fondazione Bruno Kessler

NPI Final Presentation - 7/12/2016

Introduction

Timed Failure Propagation Graphs

Diagnosability Analysis

Conclusion

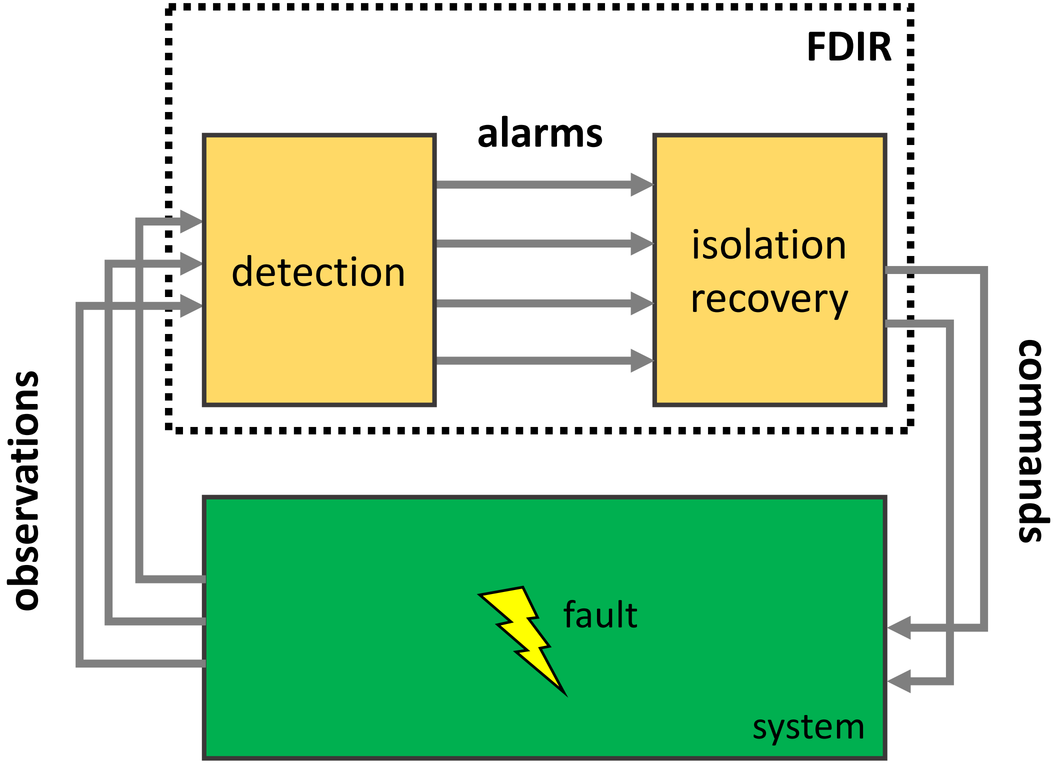
ESA Networking/Partnering Initiative (NPI)

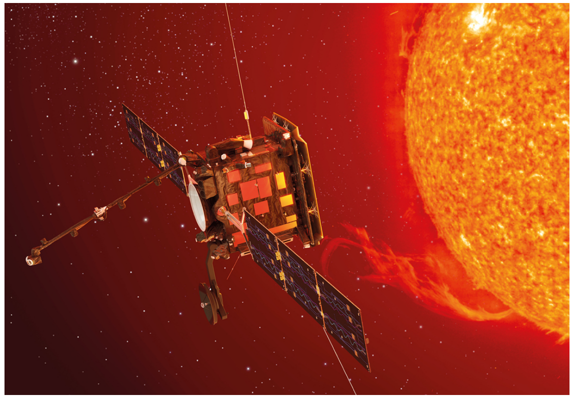
- PhD at University of Trento / Fondazione Bruno Kessler (Trento, Italy)
- supervisors: Alessandro Cimatti and Marco Bozzano

- co-financed by ESA through the NPI program
- builds on / inspired by other ESA projects (COMPASS, FAME)

aim: **automated tools to support formal FDIR design**

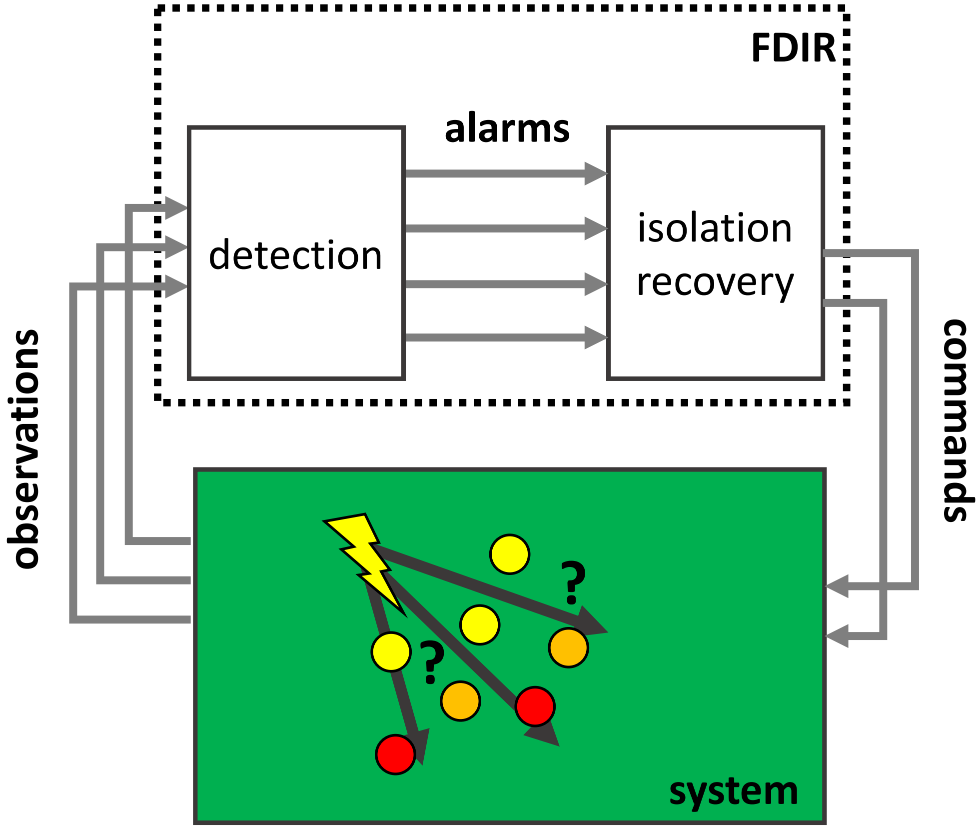
Fault Management via FDIR



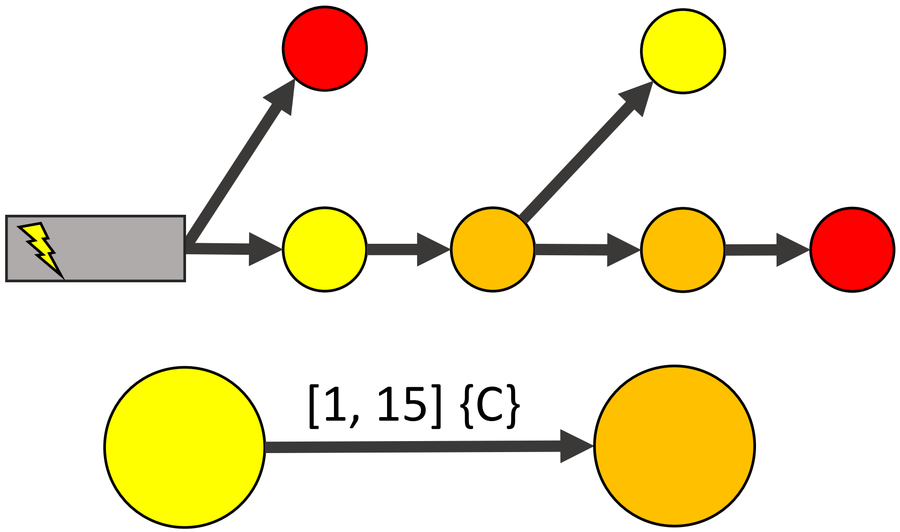


- faults vs. safety / availability
- need for fault management
- classical paradigm: FDIR

Effects of Faults? Propagation Speed?

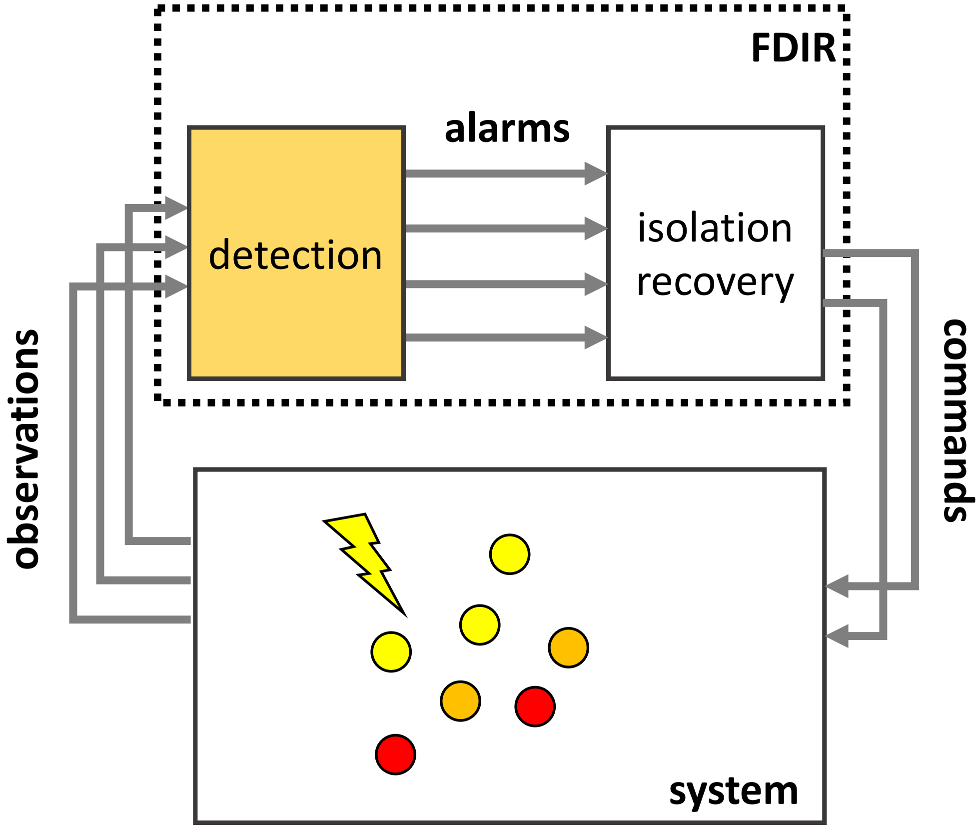


Timed Failure Propagation Graphs



- Validation w.r.t. system model?
- Automated generation?

Can effective detection be implemented?



Diagnosability Analysis

Fault diagnosable?

yes (plus optional reduction of observables)

no (plus debugging info)

Automating verification and observables optimization?

Introduction

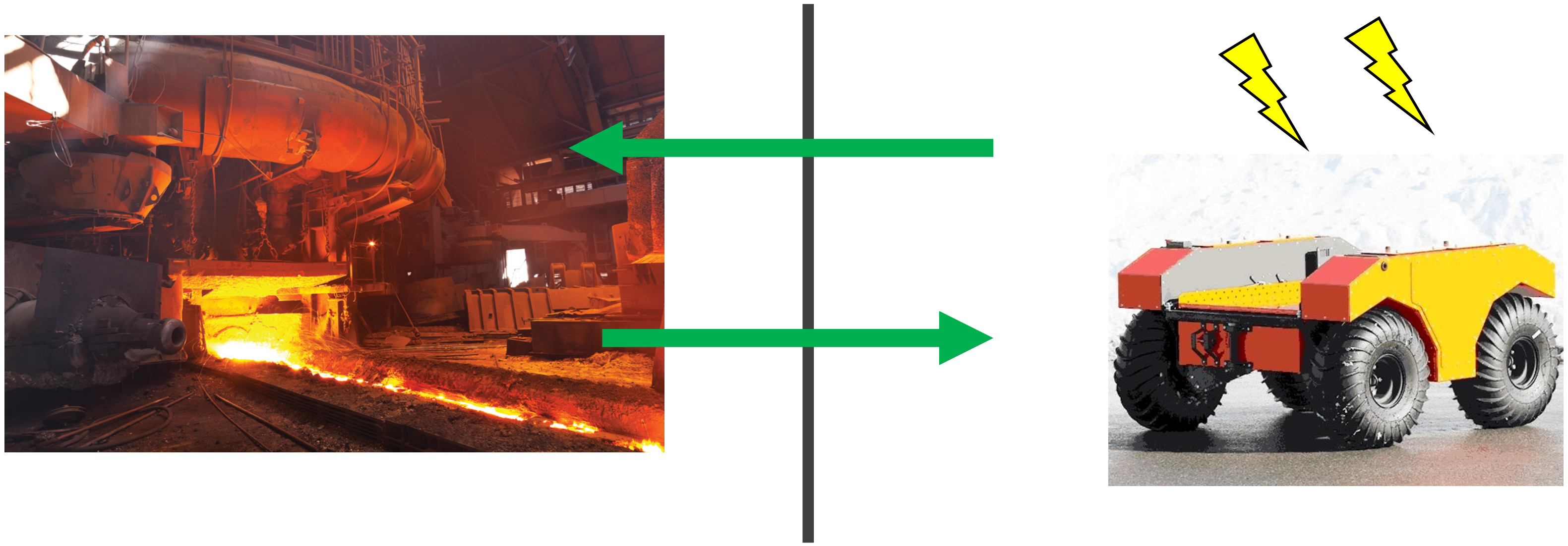
Timed Failure Propagation Graphs

- **TFPG formalism**
- Behavioral Validation
- Synthesis
- Implementation & Benchmarks
- Case studies

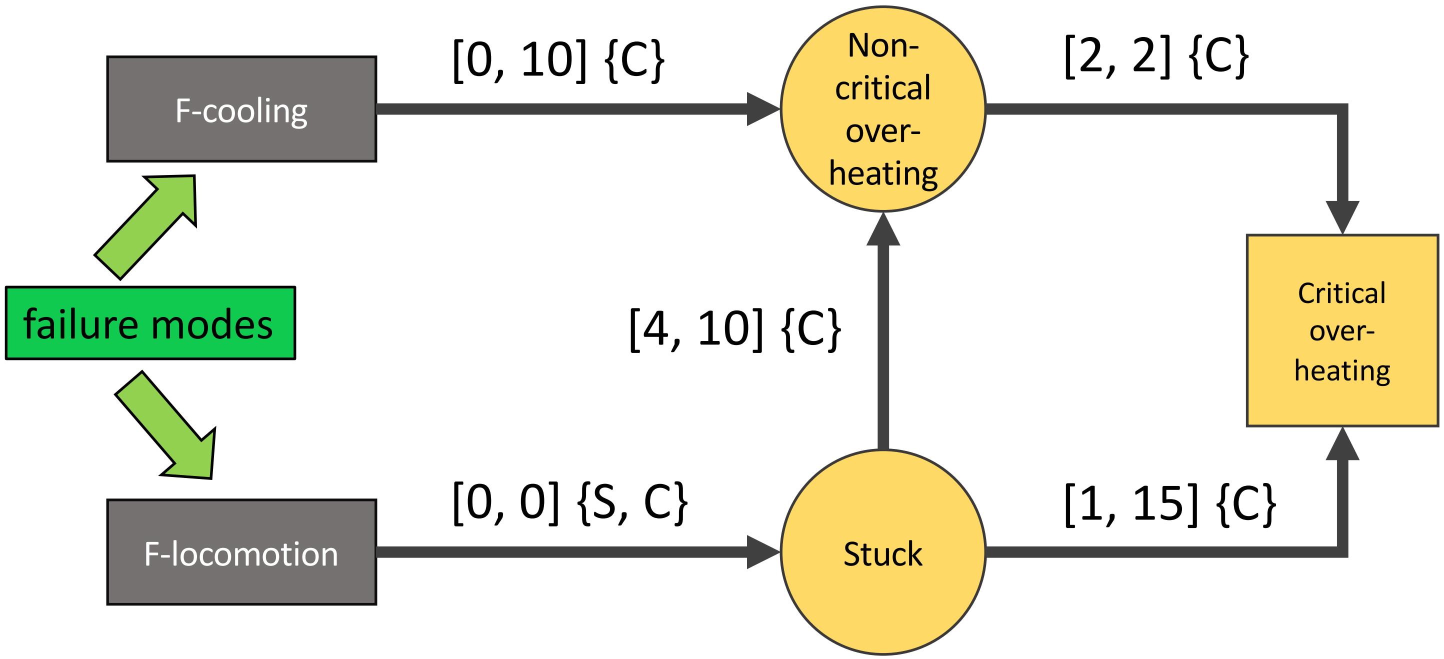
Diagnosability Analysis

Conclusion

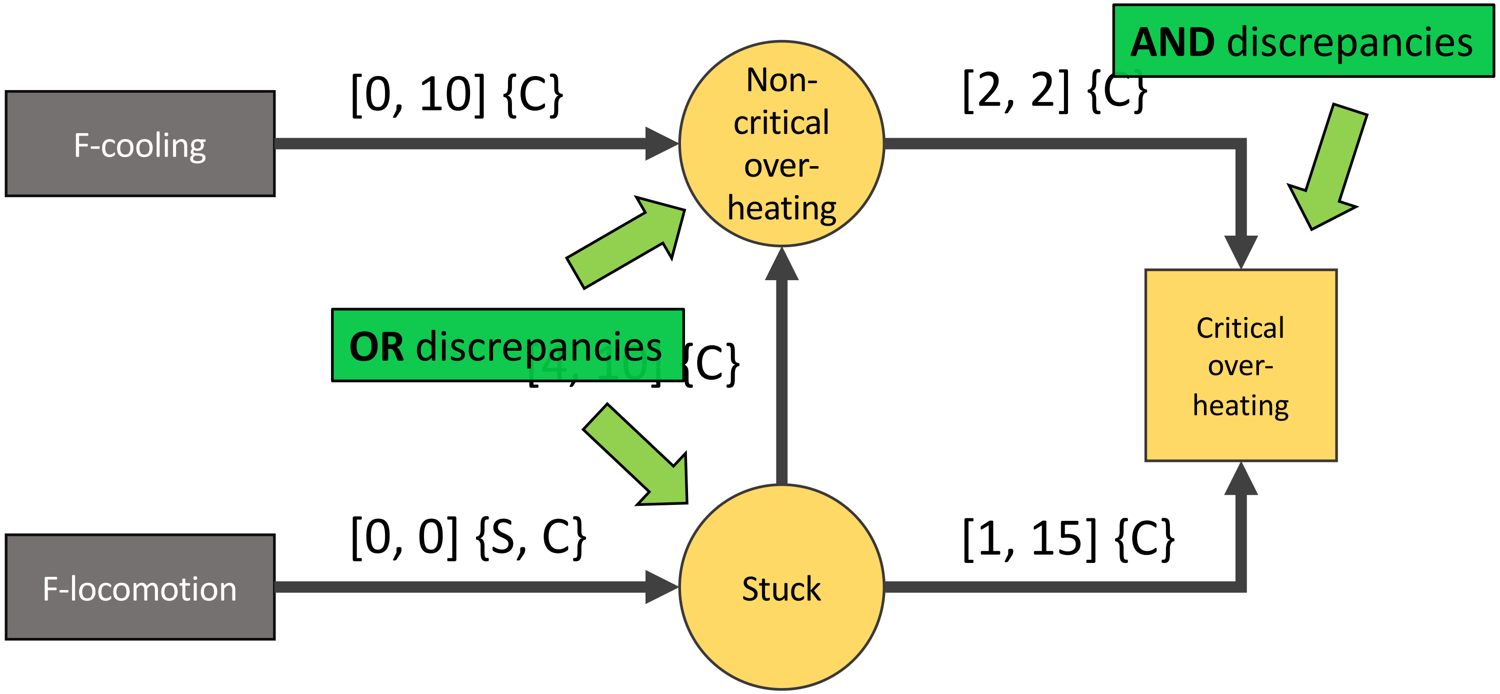
Running Example: Industrial Furnace Robot



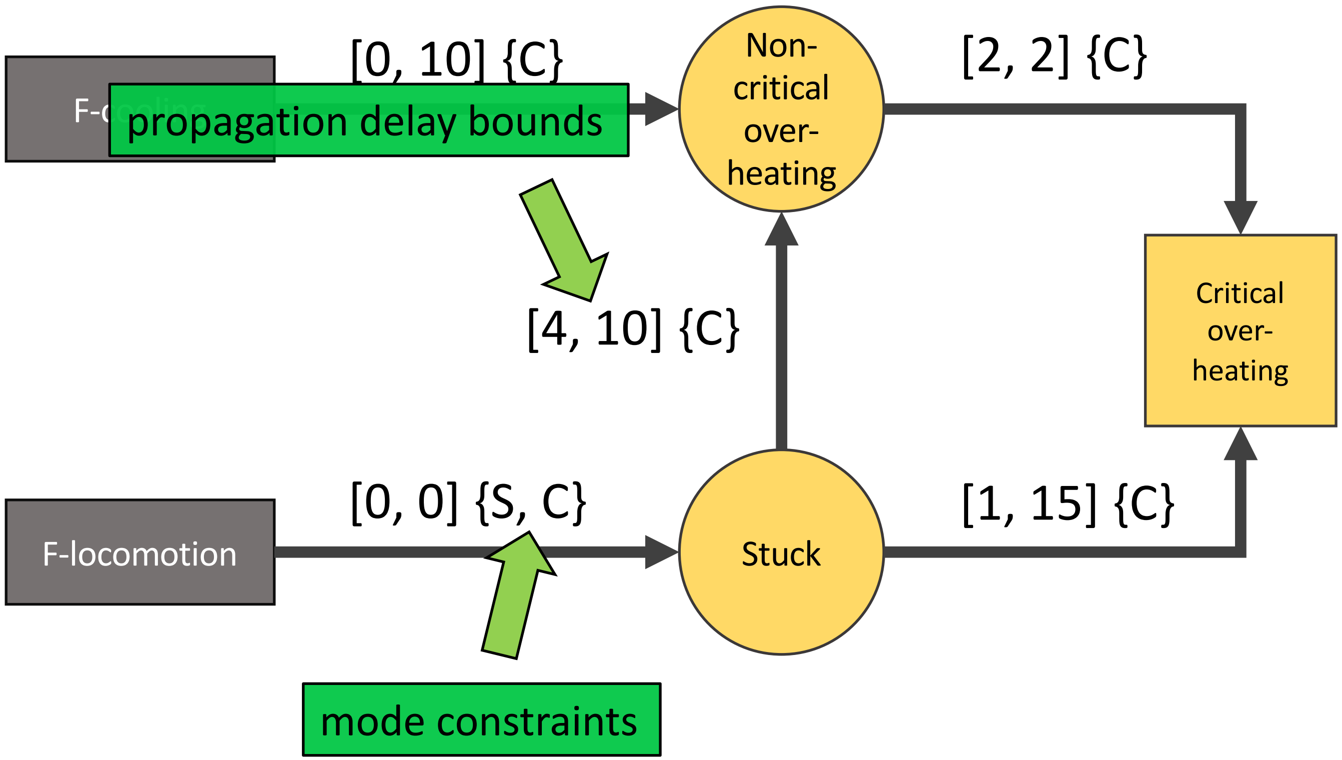
Timed Failure Propagation Graphs



Timed Failure Propagation Graphs



Timed Failure Propagation Graphs



Introduction

Timed Failure Propagation Graphs

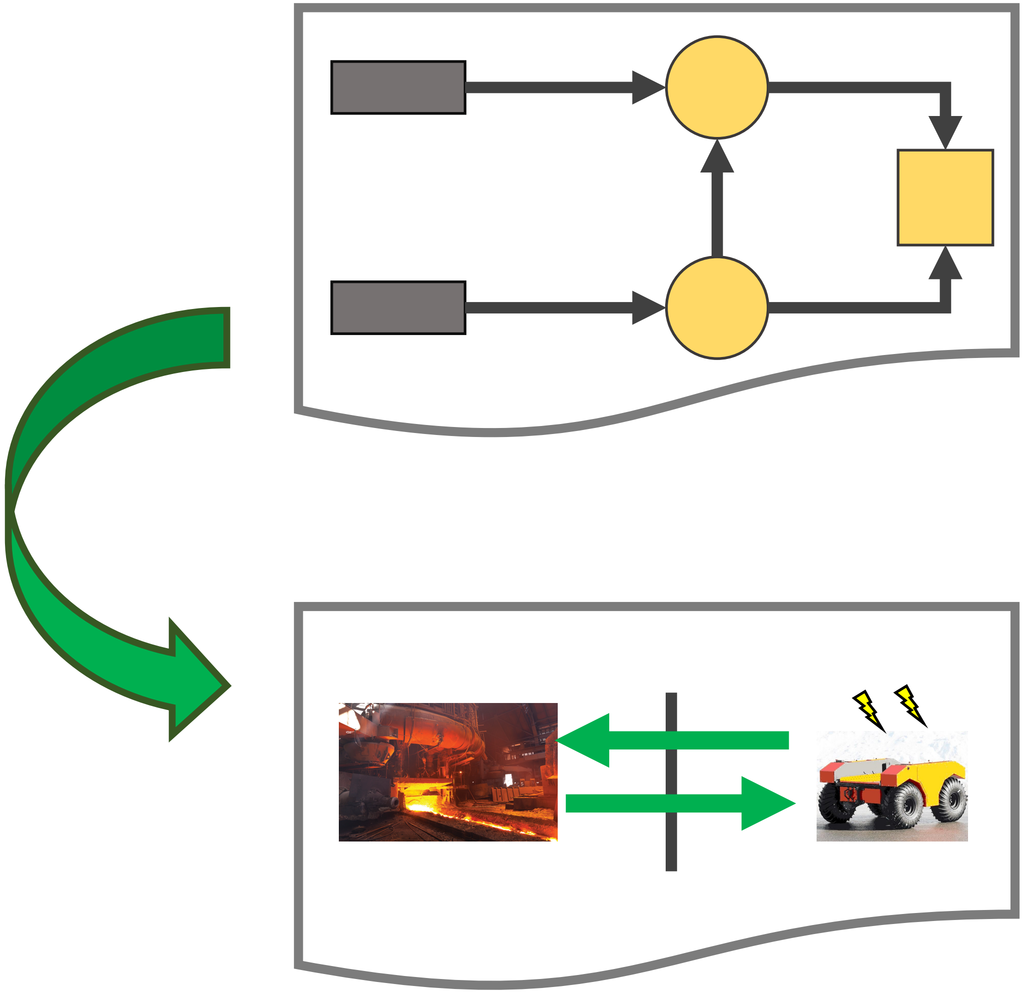
- TFPG formalism
- **Behavioral Validation**
- Synthesis
- Implementation & Benchmarks
- Case studies

Diagnosability Analysis

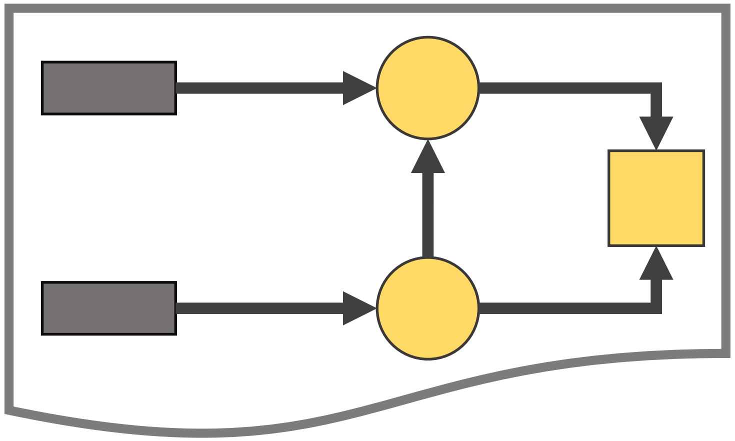
Conclusion

Problem 1:
Behavioral Validation

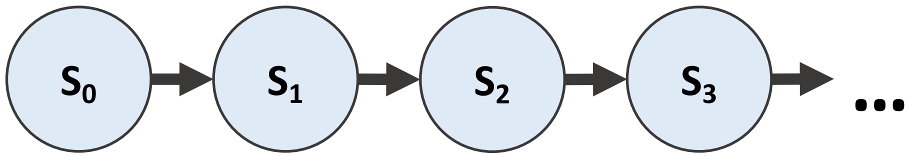
Is a given TFPG a good representation of the system behavior under faults?

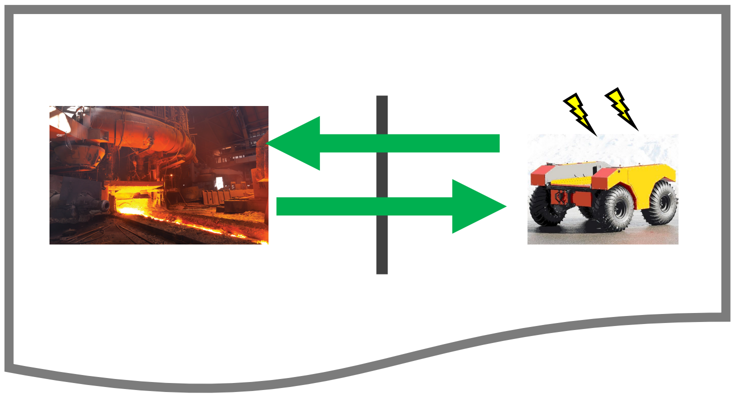


Formal Background



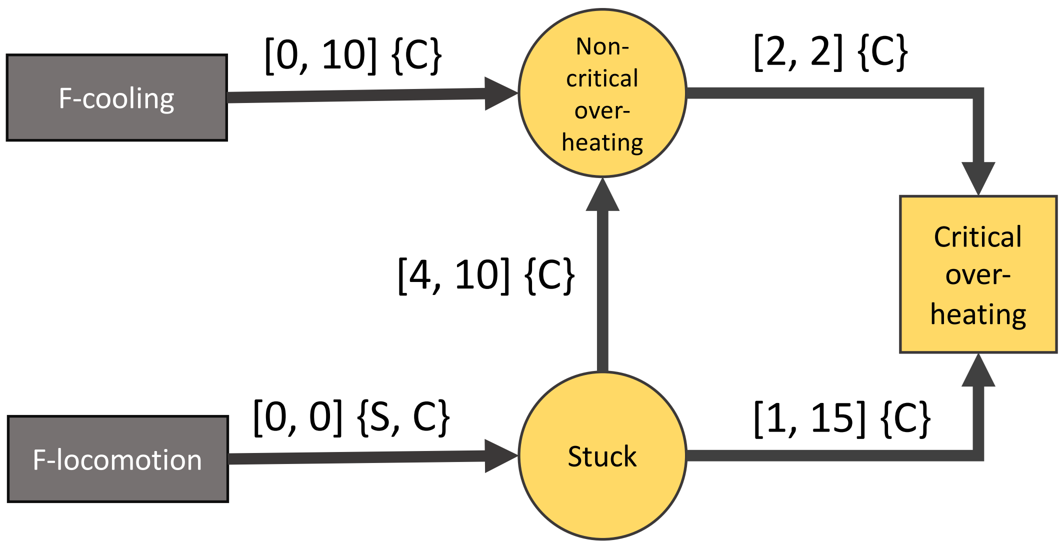
- infinite-state transition systems
- sequences of states with time-stamps



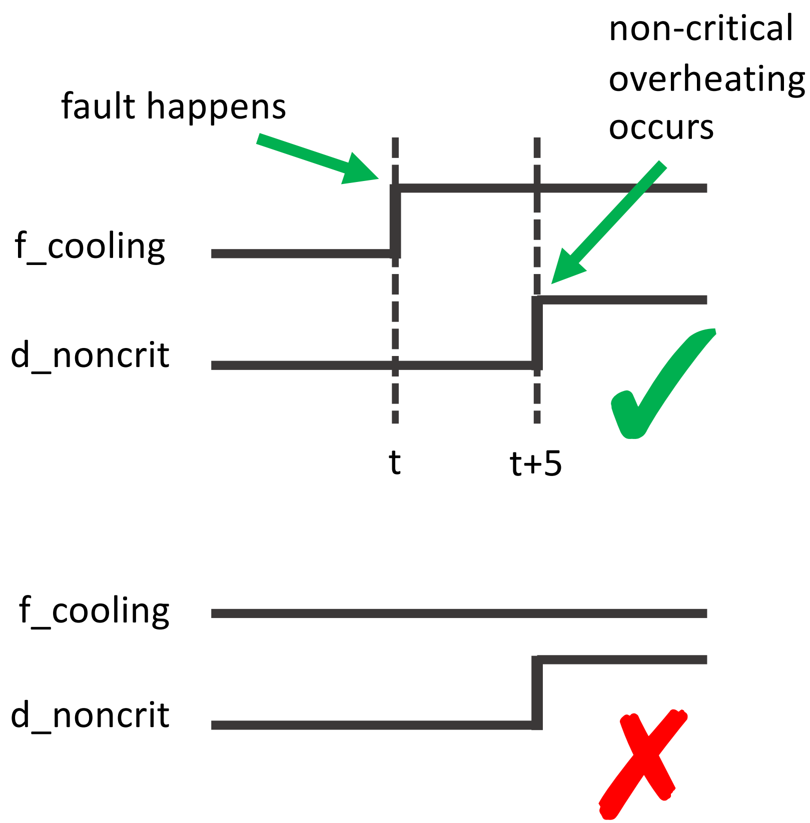


- Metric Temporal Logic (MTL)
- symbolic model-checking
  - exhaustive exploration of all behaviors

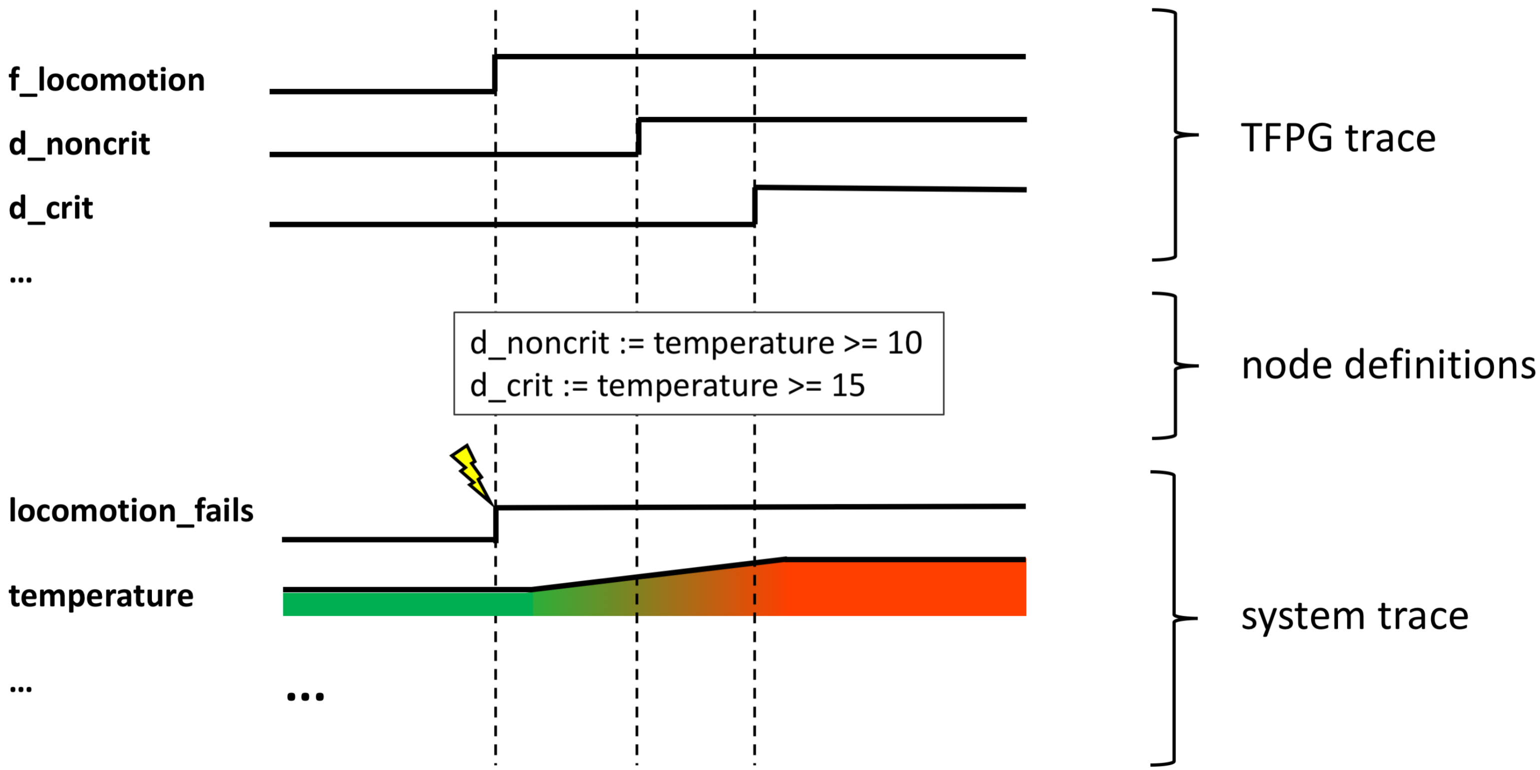
Trace-based TFPG semantics



TFPG constraints satisfied on TFPG traces?



TFPG traces vs. system traces



TFPG Behavioral Validation

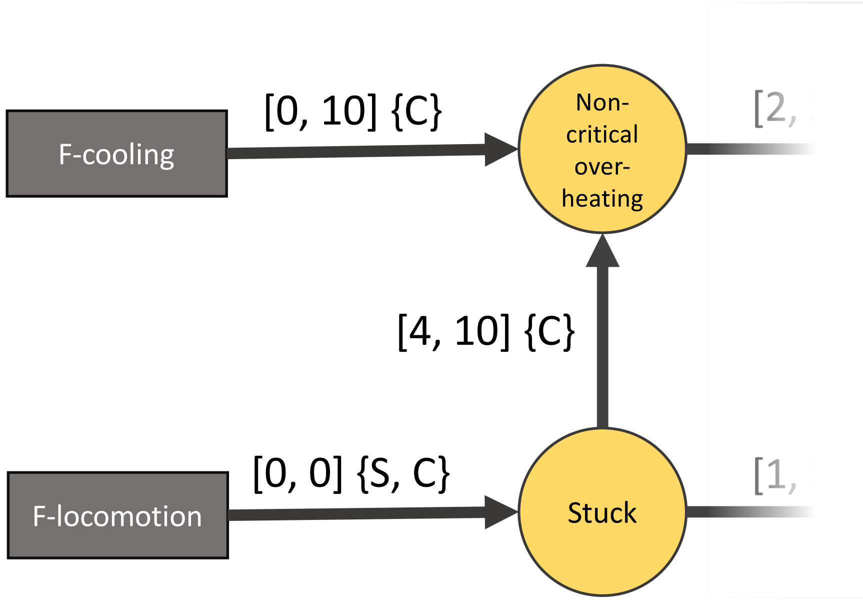
Completeness Property

The TFPG abstraction of every system trace satisfies the TFPG constraints.

Tightness Property

Edge constraints cannot be tightened without breaking TFPG completeness.

Properties are verified with **model-checking**.



Introduction

Timed Failure Propagation Graphs

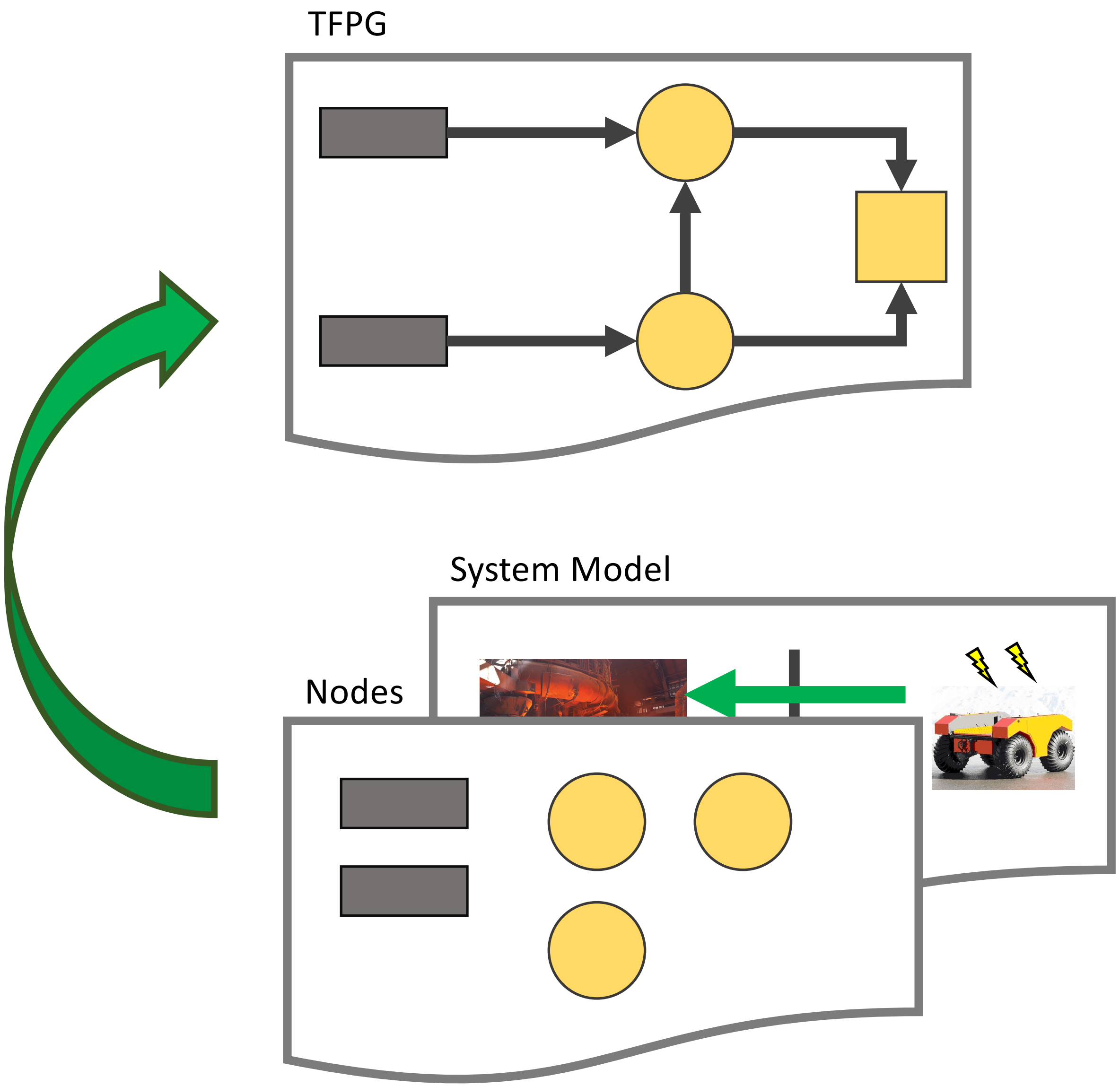
- TFPG formalism
- Behavioral Validation
- **Synthesis**
- Implementation & Benchmarks
- Case studies

Diagnosability Analysis

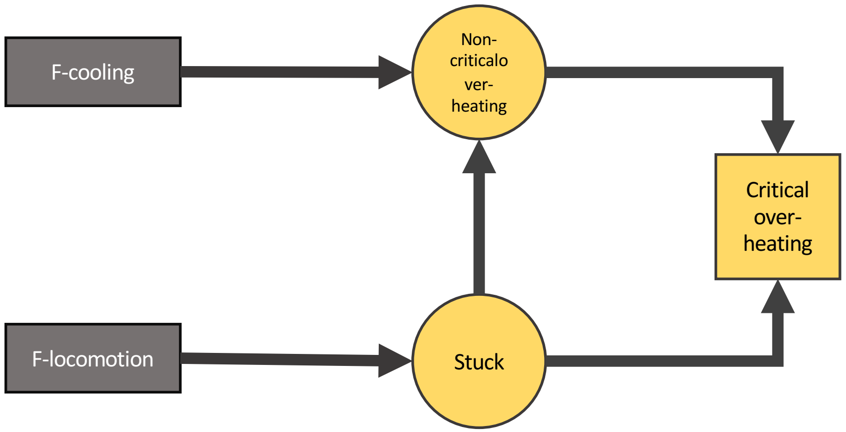
Conclusion

Problem 2:
Synthesis

How to generate TFPG automatically?

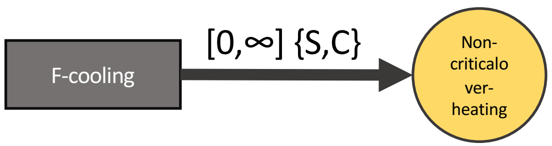


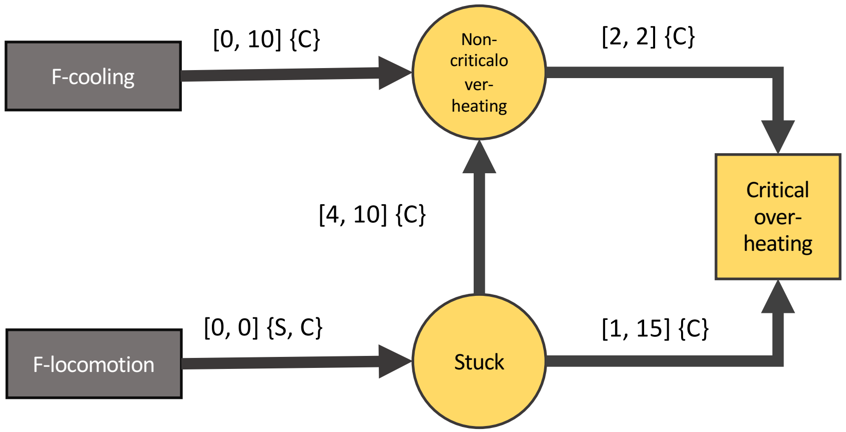
TFPG Synthesis



Part I: Compute Graph

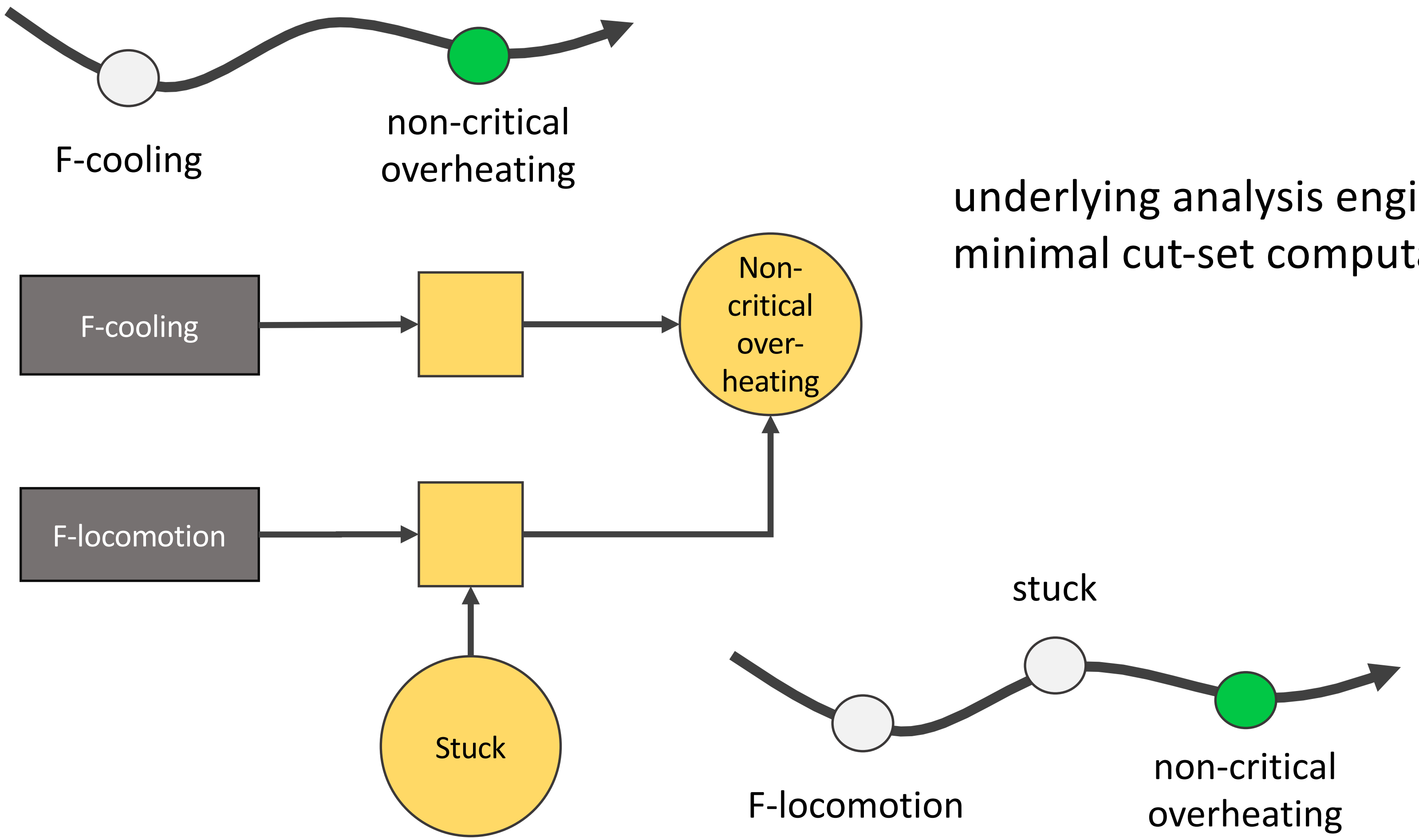
Edges are maximally permissive:





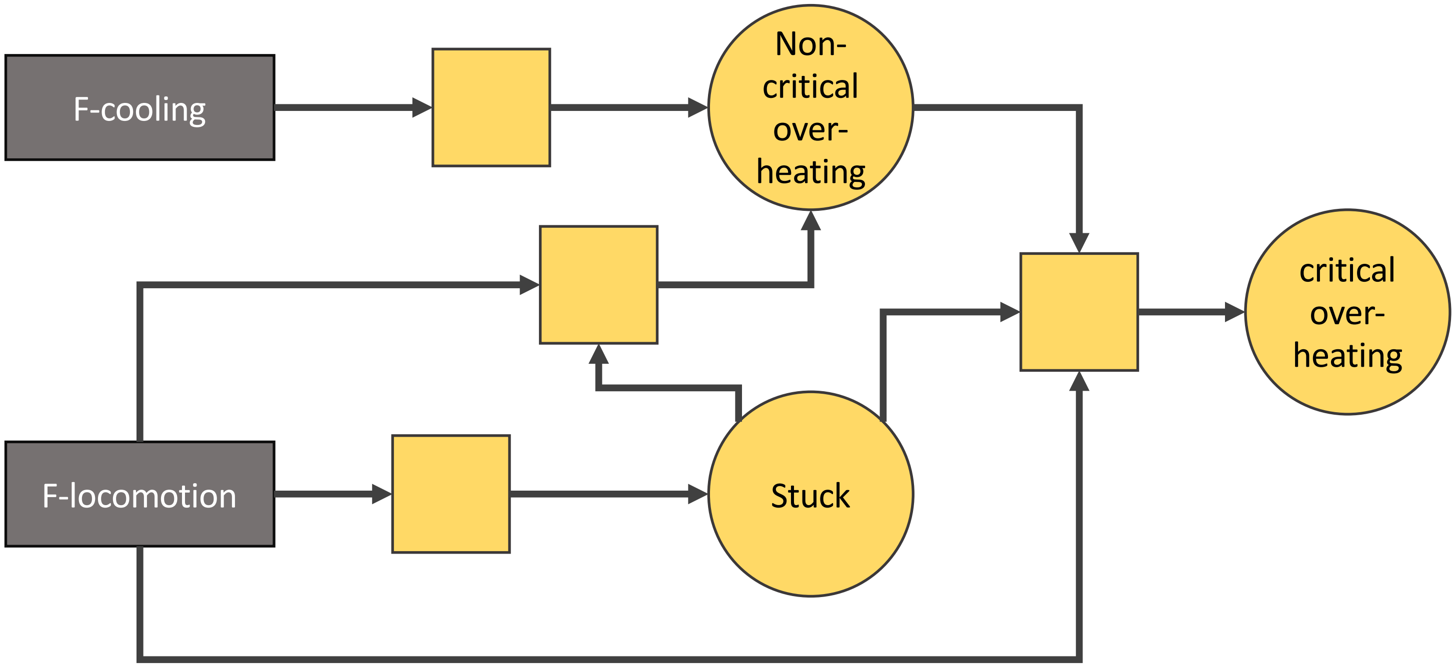
Part II: Compute tight edge constraints

Step 1: Compute Precedence Constraints



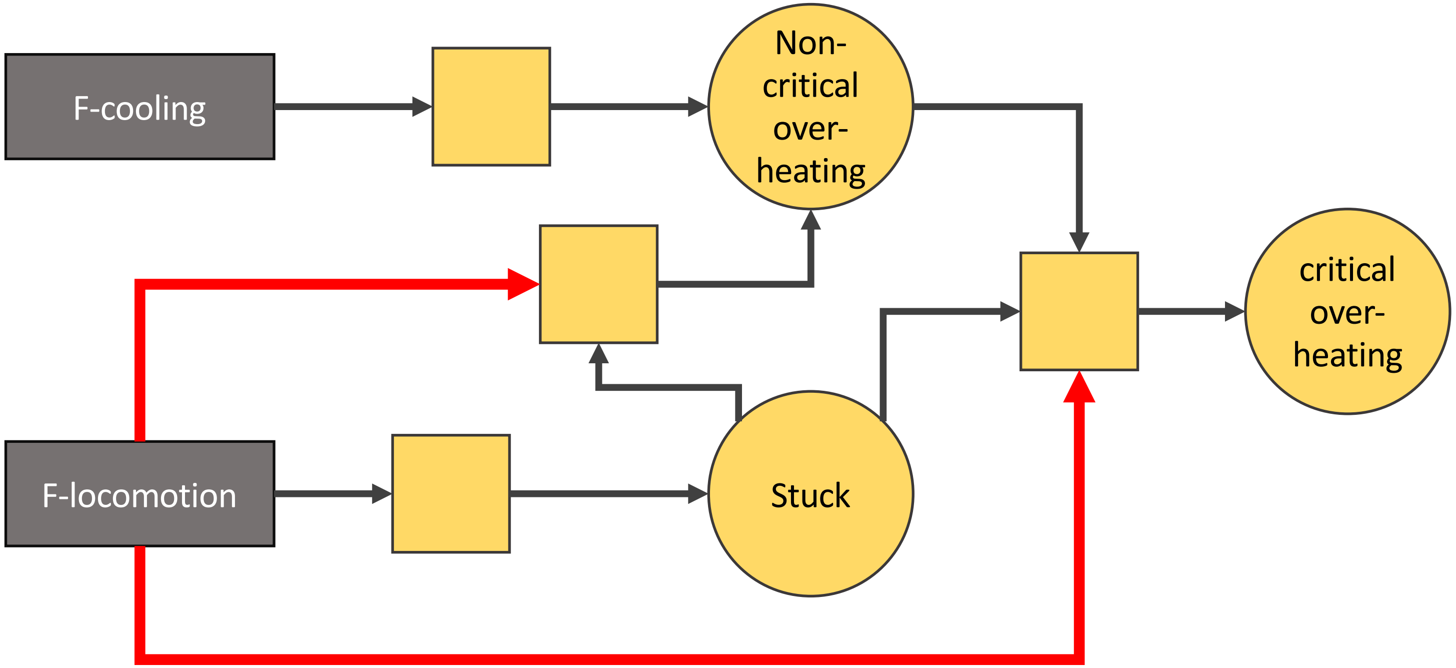
underlying analysis engine:
minimal cut-set computation

Step 2: Instantiate TFPG



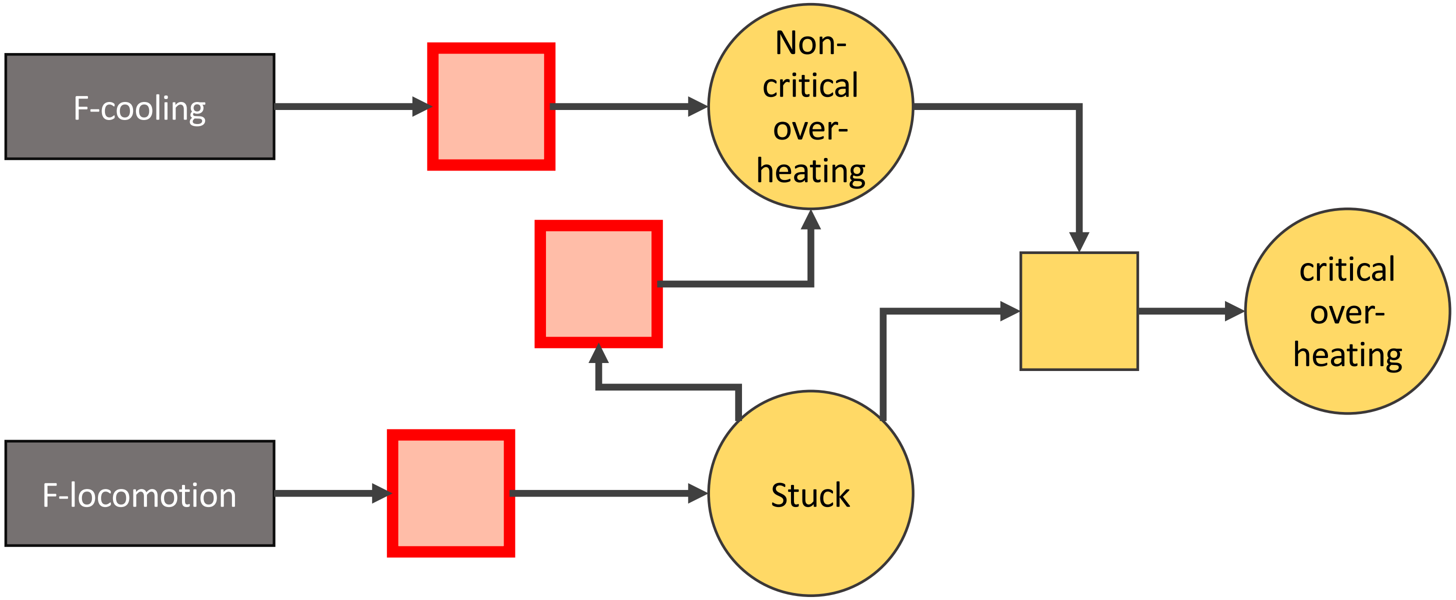
edges are labeled with tmin=0, tmax=∞, modes=ALL

Step 3: Simplification



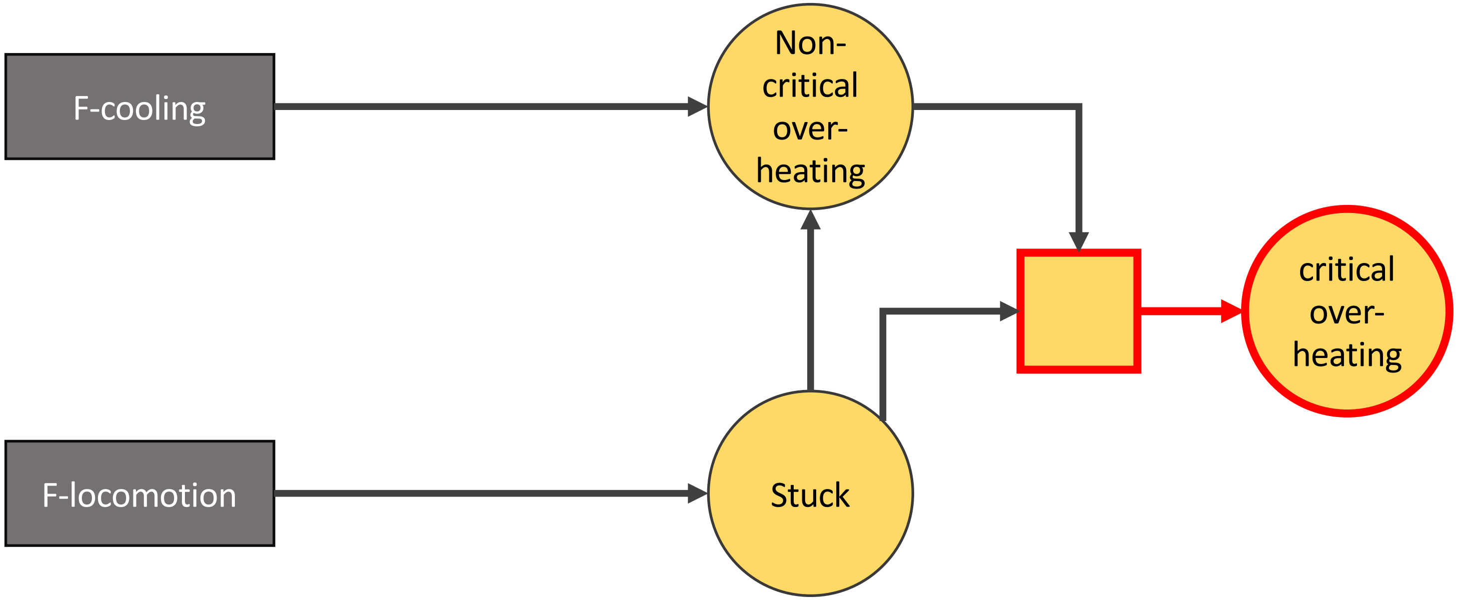
use Boolean reasoning to identify redundant edges

Step 3: Simplification



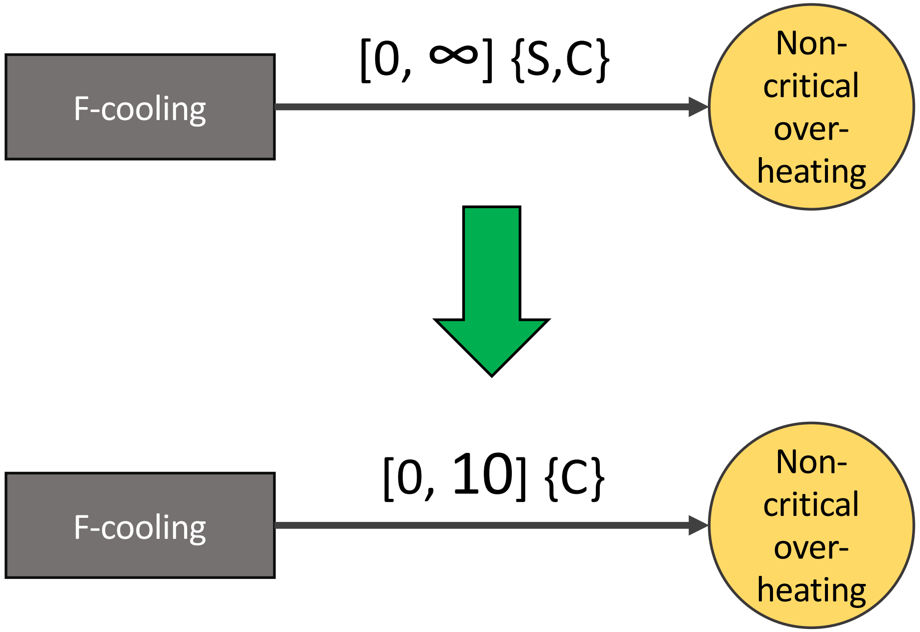
remove unnecessary auxiliary nodes

Step 3: Simplification



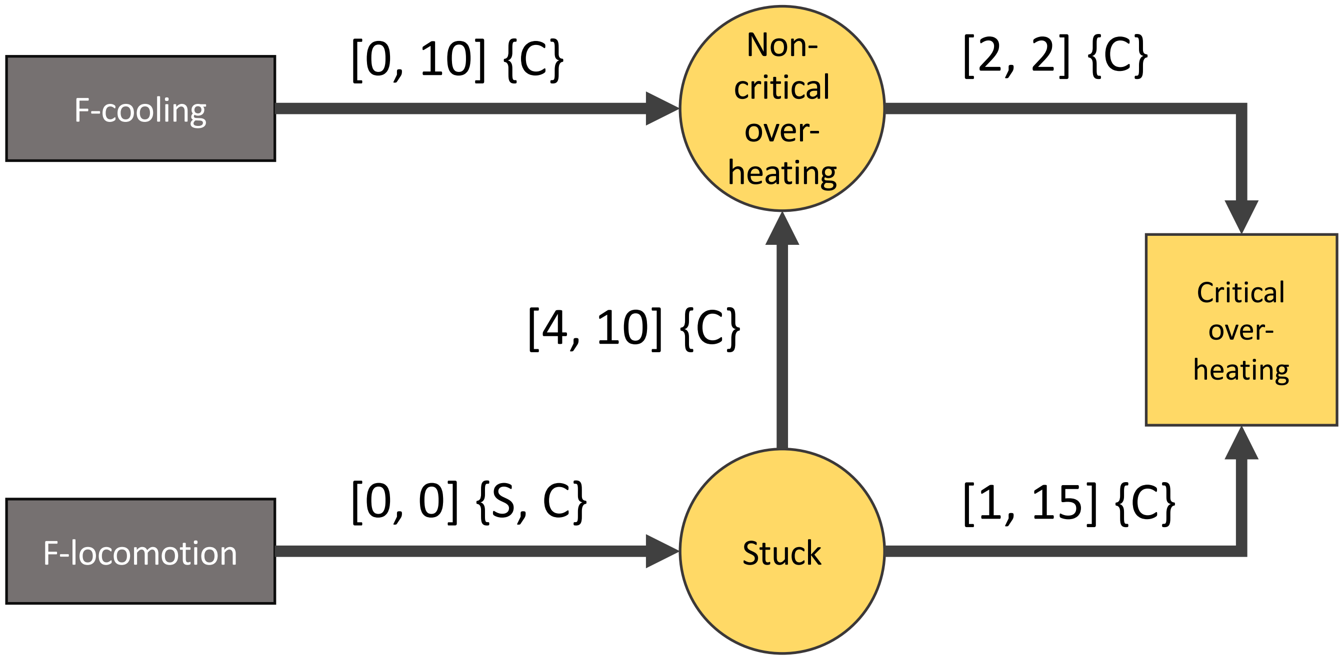
identify AND discrepancies

Automated Tightening



based on model-checking iterations

Resulting TFPG



- is complete and tight
- accurately encodes precedence constraints

Introduction

Timed Failure Propagation Graphs

- TFPG formalism
- Behavioral Validation
- Synthesis
- **Implementation & Benchmarks**
- Case studies

Diagnosability Analysis

Conclusion

Implementation and Benchmarks

implemented in **xSAP**

- back-end of COMPASS for model-based safety analysis
- linked to **nuXmv** / **NuSMV**, symbolic model-checker for infinite-state transition systems

Completeness Check → avg compl. time

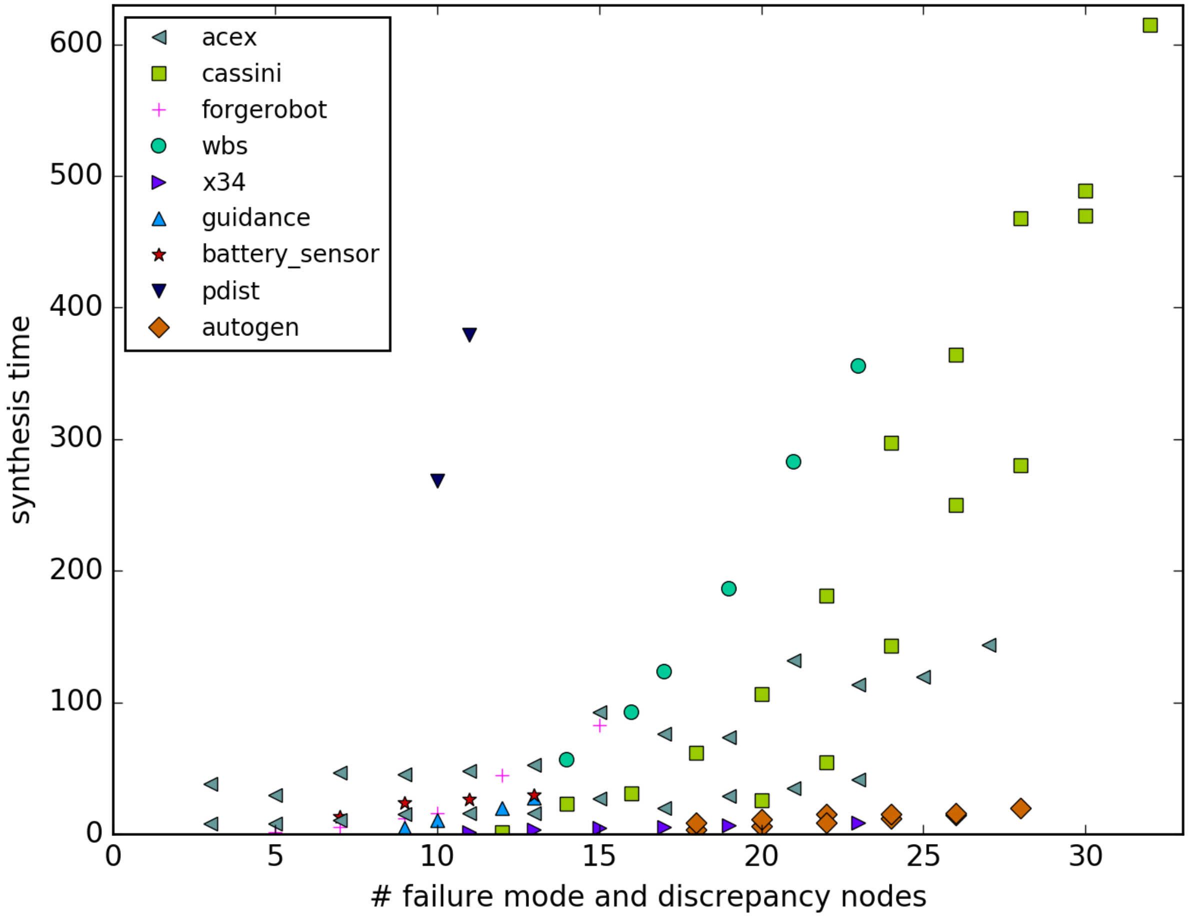
Edge Tightening → avg tighten. time

model	avg compl. time	avg tighten. time	TFPGs	avg. FM	avg. D
acex-10	171 (1.0)	731 (1.0)	15	2	15
acex-12	334 (1.0)	838 (0.9)	33	2	17
autogen	156 (1.0)	925 (1.0)	66	8	15
battery	23 (1.0)	71 (1.0)	23	4	6
cassini2	28 (1.0)	75 (1.0)	15	10	6
cassini4	322 (1.0)	1179 (0.9)	39	16	10
forge-B	103 (1.0)	160 (1.0)	3	2	3
forge-R1	2 (1.0)	10 (1.0)	3	2	3
forge-R2	24 (1.0)	224 (1.0)	3	4	8
forge-R3	145 (1.0)	↑ (0.0)	3	6	13
guidance	14 (1.0)	94 (1.0)	12	6	6
pdist	622 (1.0)	2776 (0.2)	6	7	7
wbs	67 (1.0)	n.a.	1	9	8
x34	21 (1.0)	n.a.	1	9	18

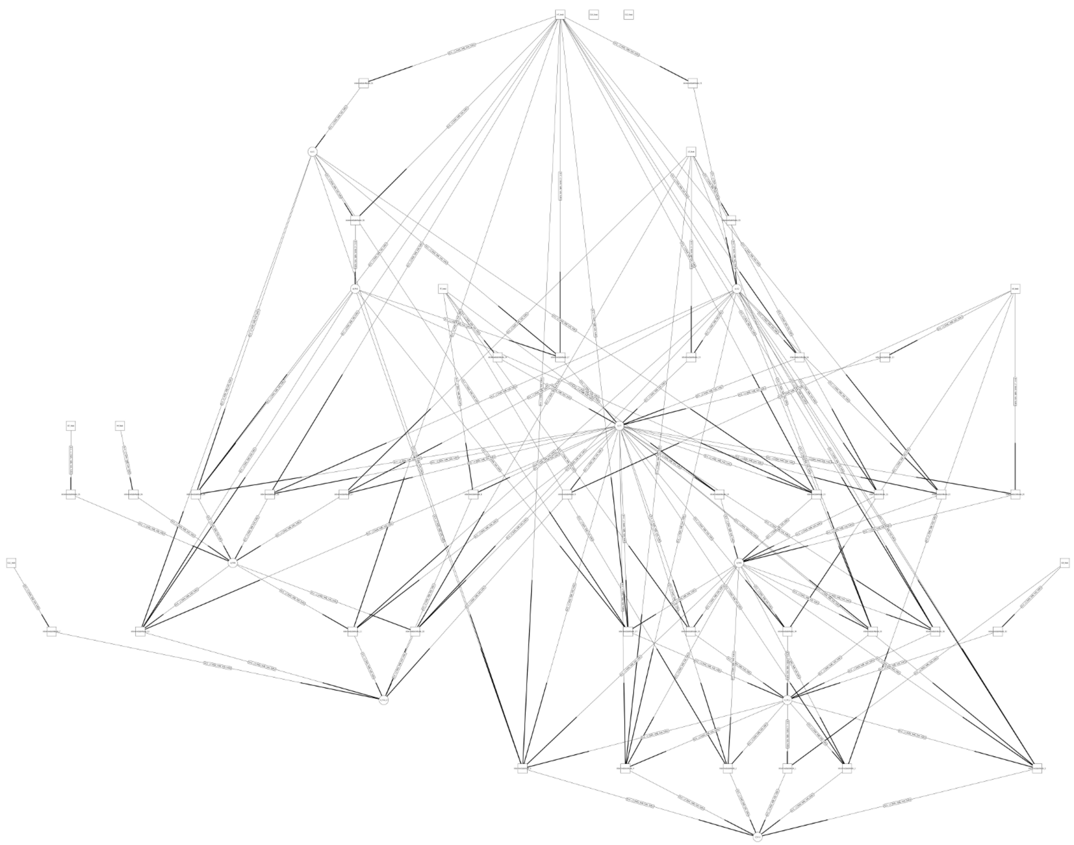
seconds (#solved/#total)

timeout: 1h / memory: 4GB

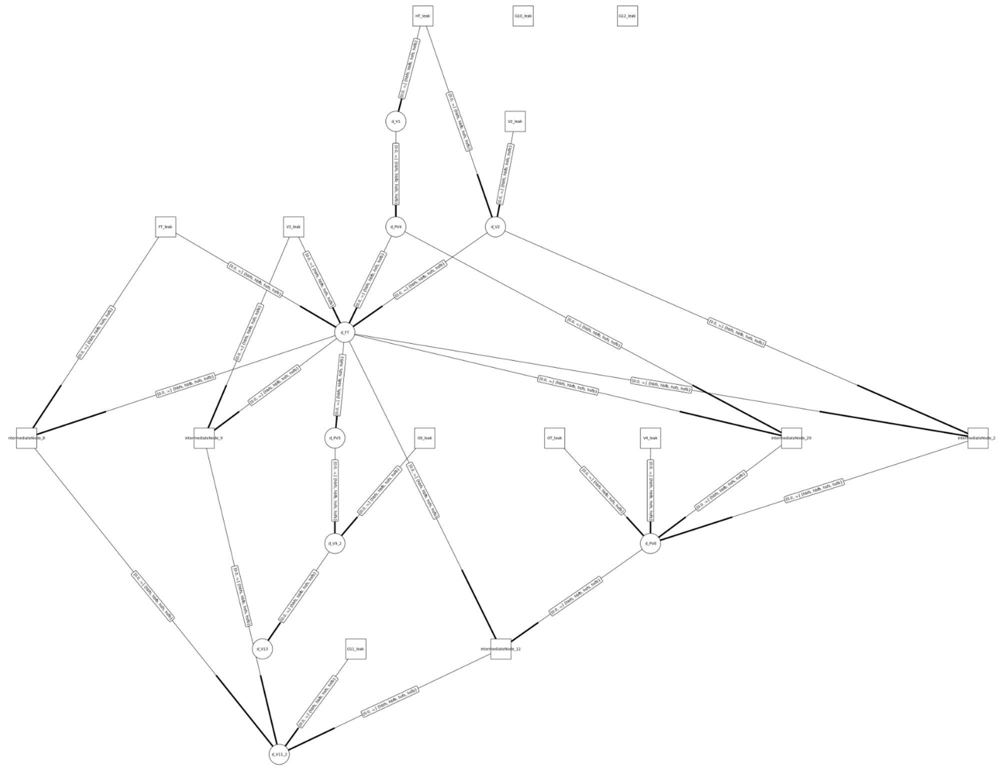
Synthesis and Simplification of Graph



Effect of Simplification



verbose



simplified

Introduction

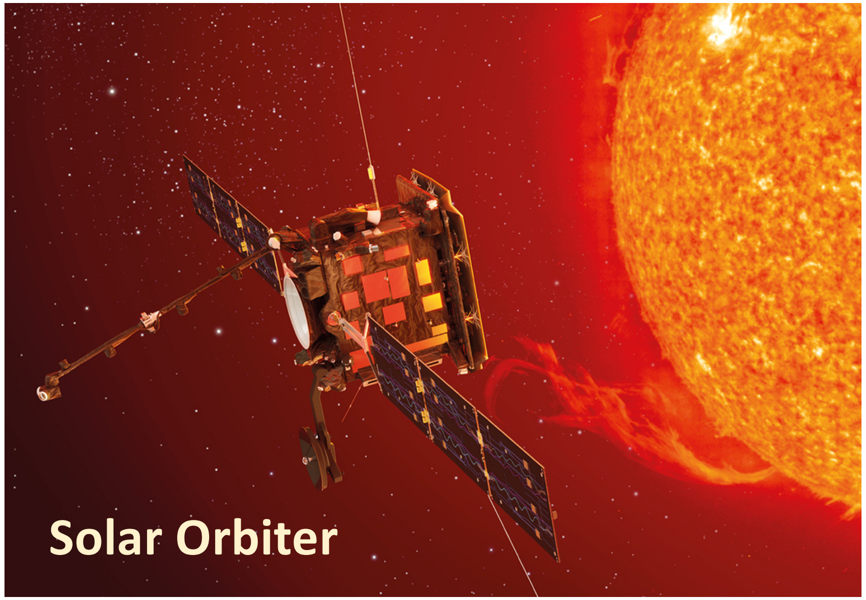
Timed Failure Propagation Graphs

- TFPG formalism
- Behavioral Validation
- Synthesis
- Implementation & Benchmarks
- **Case studies**

Diagnosability Analysis

Conclusion

Three Case Studies



© ESA

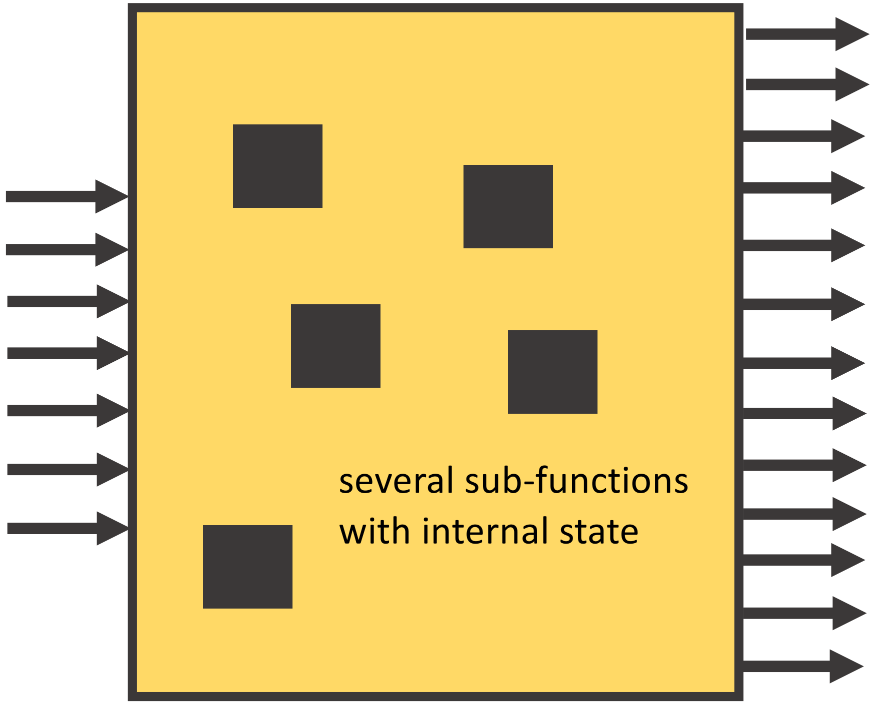
- Solar Orbiter (SOLO): sun-orbiting science mission under development
- three case studies performed during research stay at ESTEC
- focus on problems connected to attitude and orbit control
- submission of five issues to SOLO FDIR CDR panel (4 major)

Case Study 1: Software-based Propagation

7 input signals:
- raw sensor readings
- BIT values

13 faults from FMECA

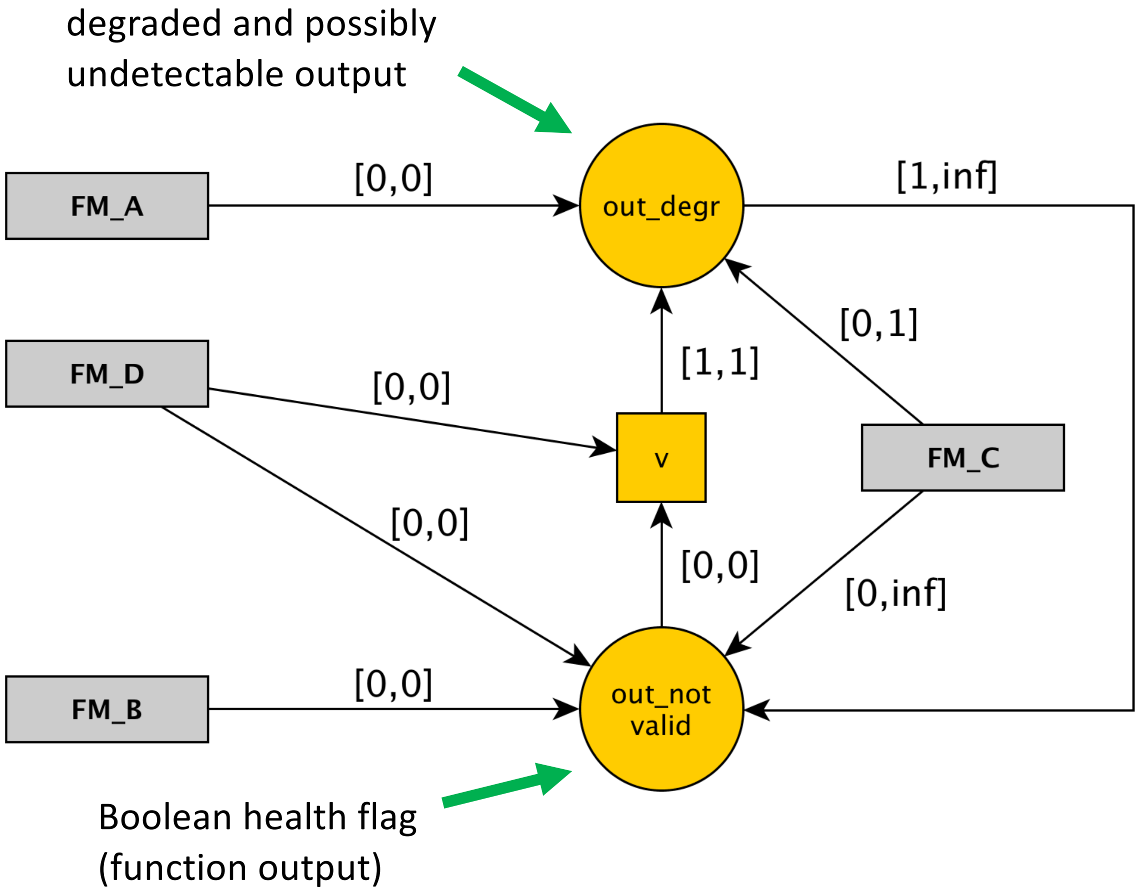
Abstract representation of values and functions.



13 output signals:
- converted readings
- computed values
- health flags

Gyroscope Channel Processing Function
(called cyclically)

Case Study 1: Software-based Propagation



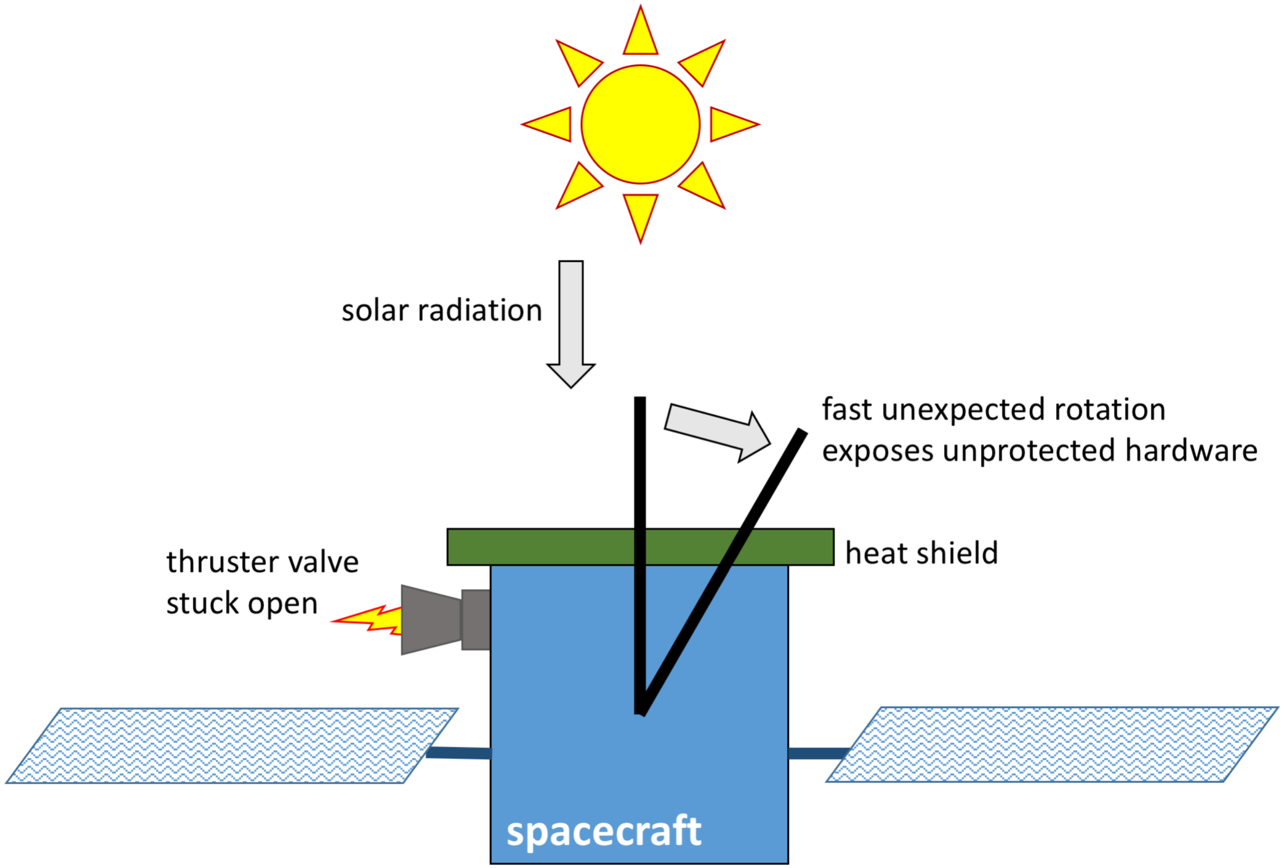
TFPG synthesis

- graph synthesis: 4sec
- tightening: 43min

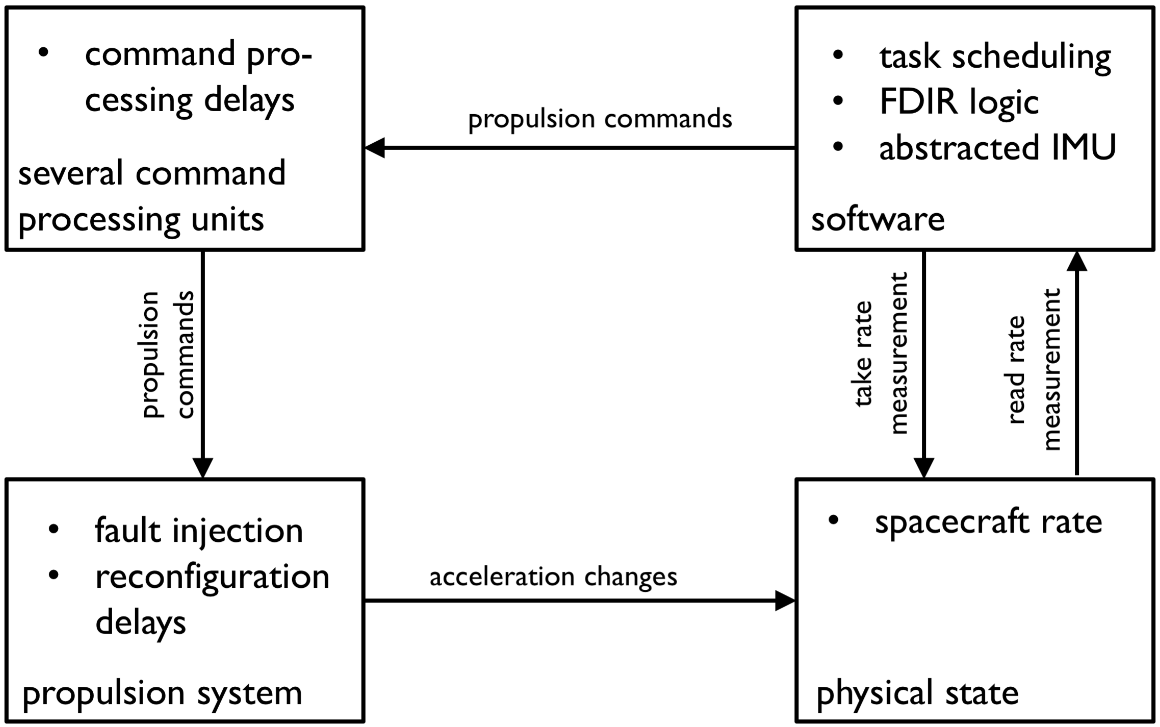
Findings

- adequacy of developed modeling framework
- graph simplification for improved readability

Case Study 2: System-level Propagation



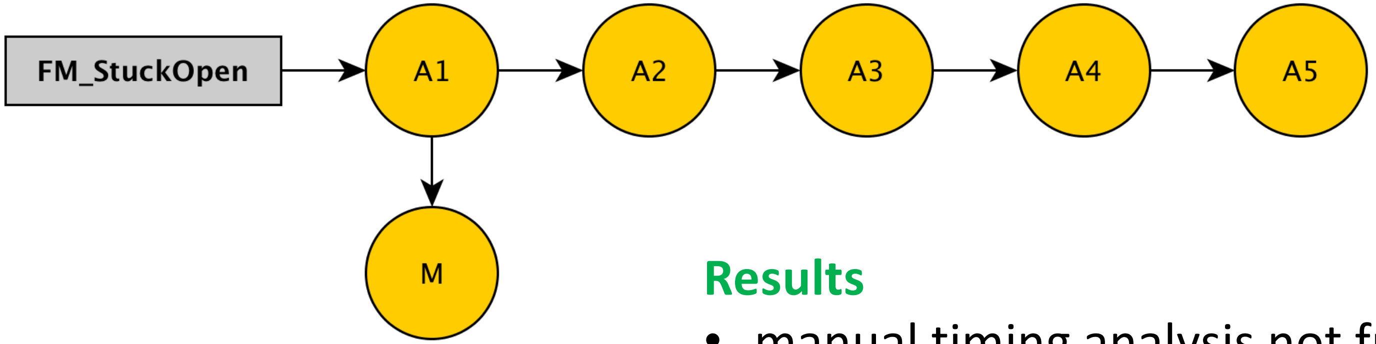
Case Study 2: System-level Propagation



Modeling

- physically realistic model
- accurate acceleration constants
- accurate delay modeling (milliseconds to several seconds)

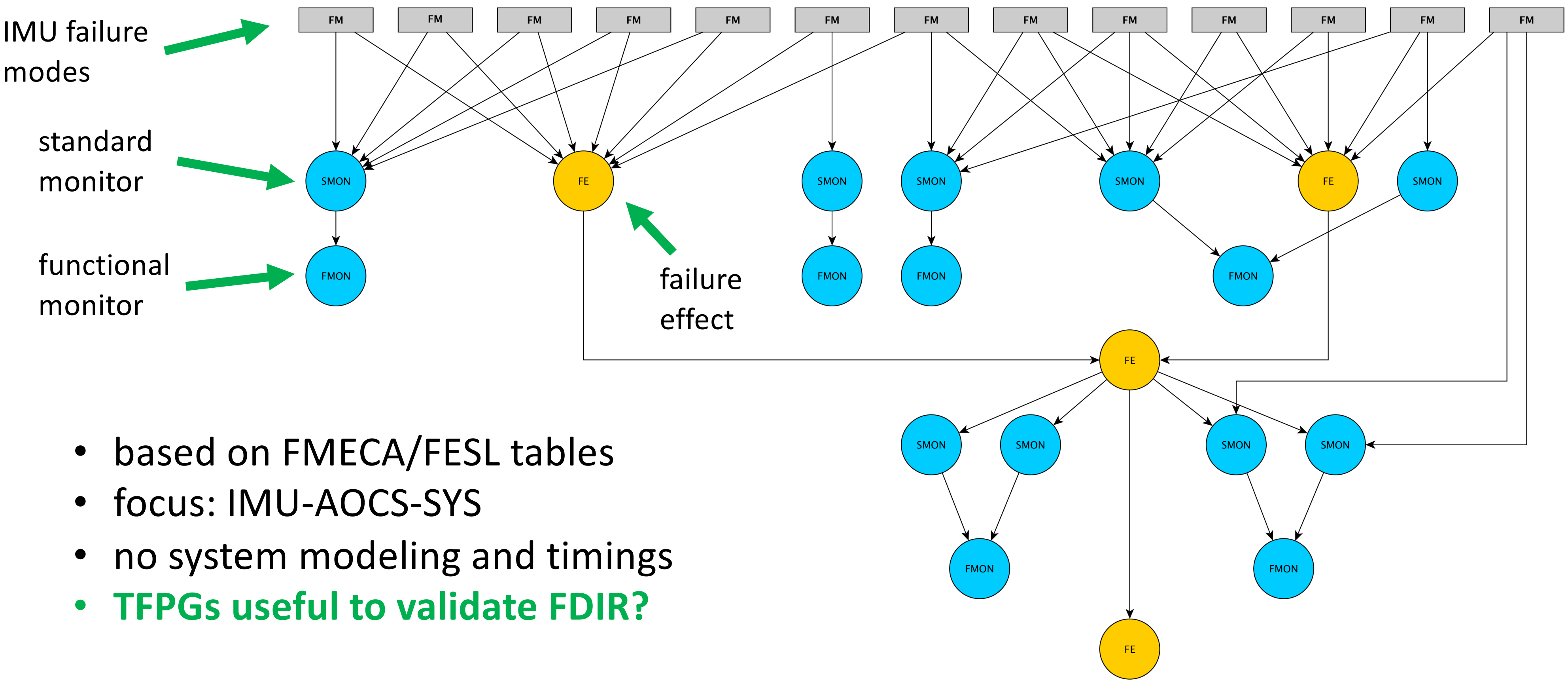
Case Study 2: System-level Propagation



Results

- manual timing analysis not fully corresponding with automated one
- first version of TFPG was not complete
- some delays (of isolation phases) were longer than estimated
- <u>completeness proved</u> on final TFPG in 30min

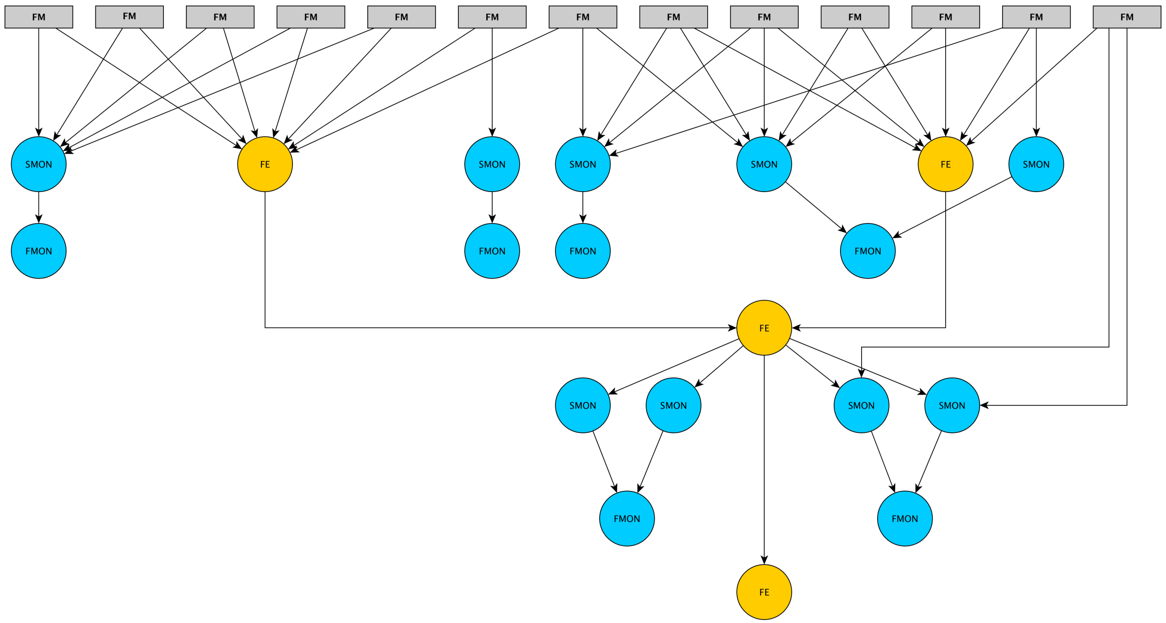
Case Study 3: Architectural Propagation



- based on FMECA/FESL tables
- focus: IMU-AOCS-SYS
- no system modeling and timings
- **TFPGs useful to validate FDIR?**

Case Study 3: Architectural Propagation

- TFPGs force engineers to be explicit about propagations and respective delays
- **corresponds to reasoning** on propagation and monitoring **during FDIR review**
- enables reasoning about:
  - time-critical propagations
  - monitor tuning



Introduction

Timed Failure Propagation Graphs

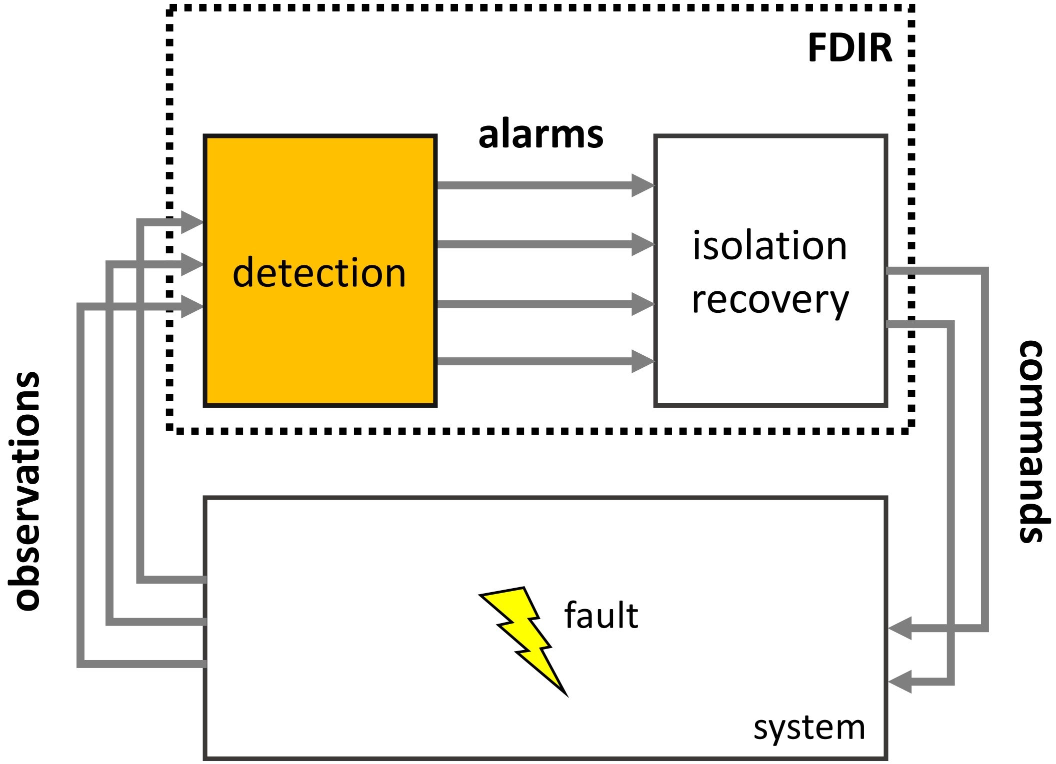
Diagnosability Analysis

- **Alarm Specification Language**
- Verification of Diagnosability
- Optimization of Observables
- Implementation & Benchmarks

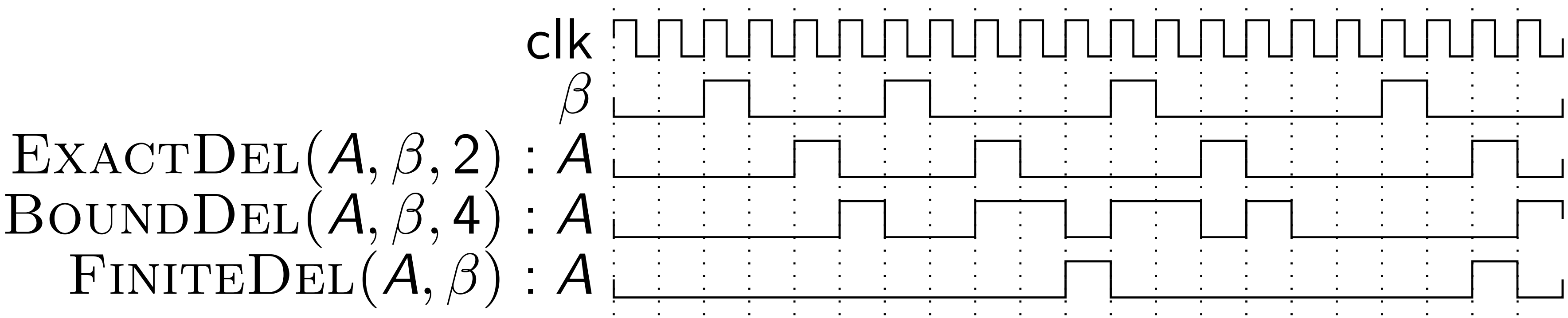
Conclusion

Alarm Specification Language (ASL)

- used to express **formal requirements on alarms** to be generated
- developed in ESA projects on FDIR design with model-based technology (AUTOGEF / FAME)



Alarm Specification Language

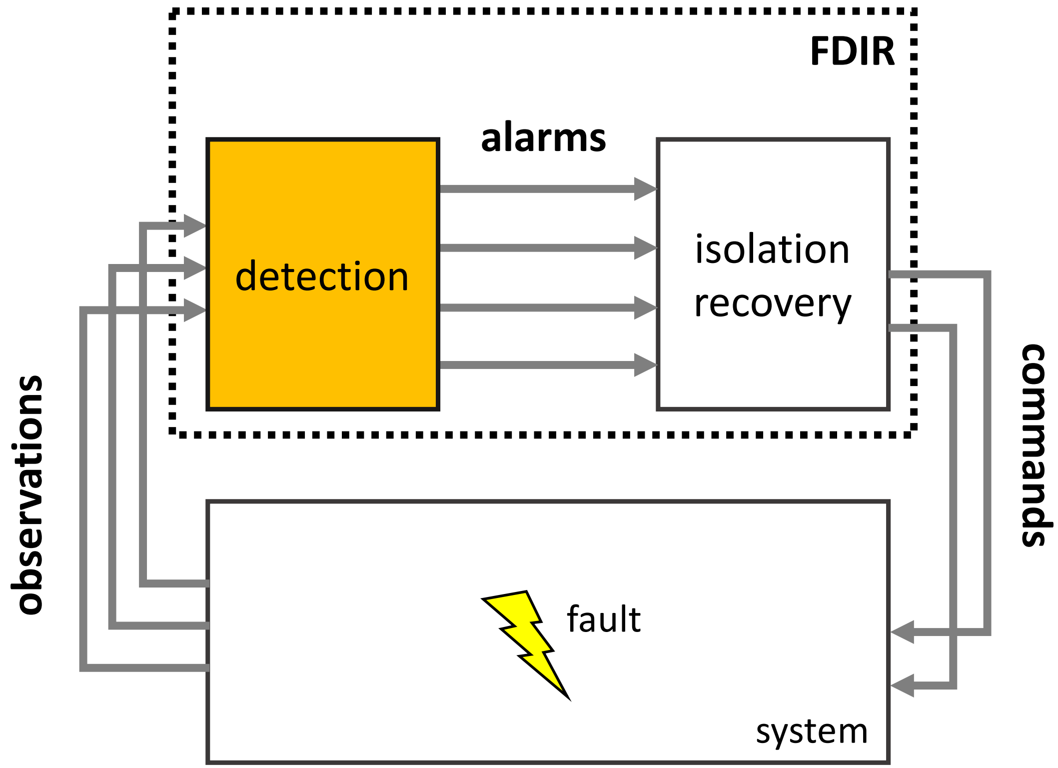


Alarm Specification Language (ASL)

Given an ASL specification:

1. **Diagnosability**: Can a corresponding detection module be implemented?

2. **Sensor Optimization**: Subsets of observables optimizing cost and guaranteeing diagnosability?



Key Framework Features

expressive specification language

- temporally extended diagnosis conditions
- various forms of delay bound requirements
- operational context

rich representation of system dynamics

- infinite-state transition system

automated algorithms for important design problems

- verification of diagnosability
- optimal selection (synthesis) of observables

Introduction

Timed Failure Propagation Graphs

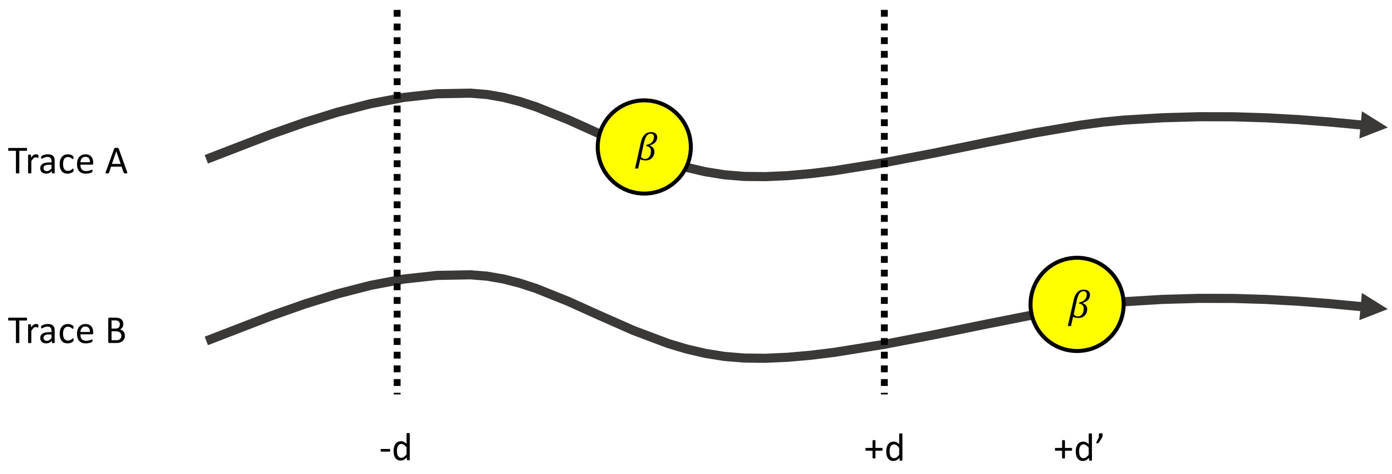
Diagnosability Analysis

- Alarm Specification Language
- **Verification of Diagnosability**
- Optimization of Observables
- Implementation & Benchmarks

Conclusion

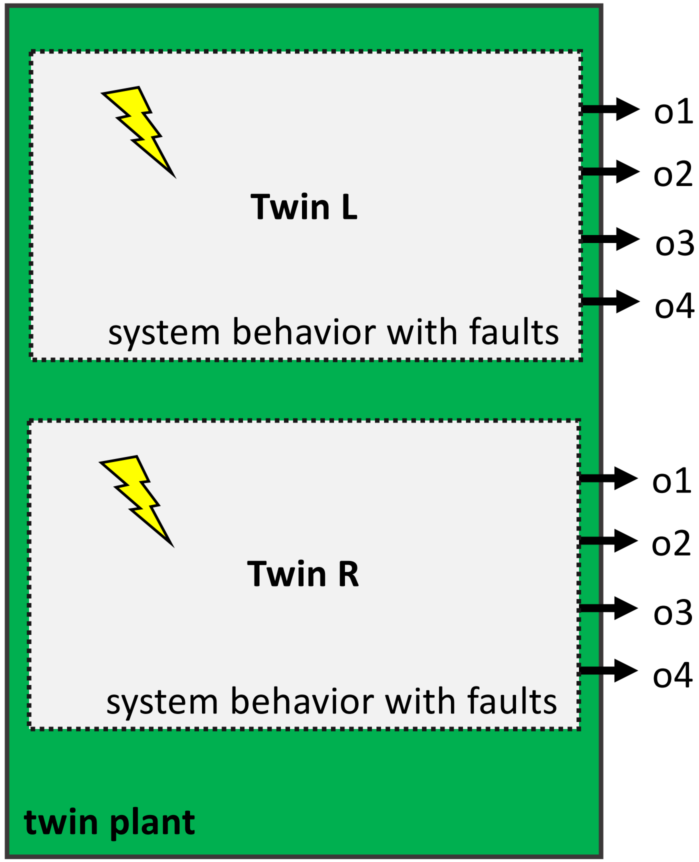
Critical Pairs

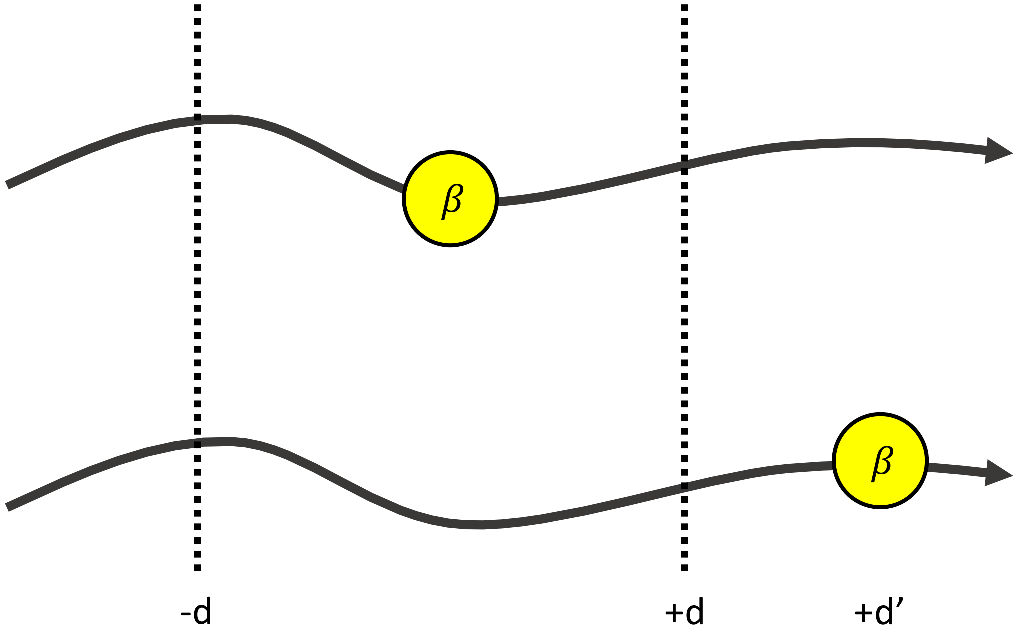
- counterexamples to diagnosability (bounded delay)



- same readings of observables on both traces
- alarm cannot be raised within required time limit; based on available information, beta might or might not have occurred.

Twin Plant





Use twin plant and model-checking to look for critical pairs.

Introduction

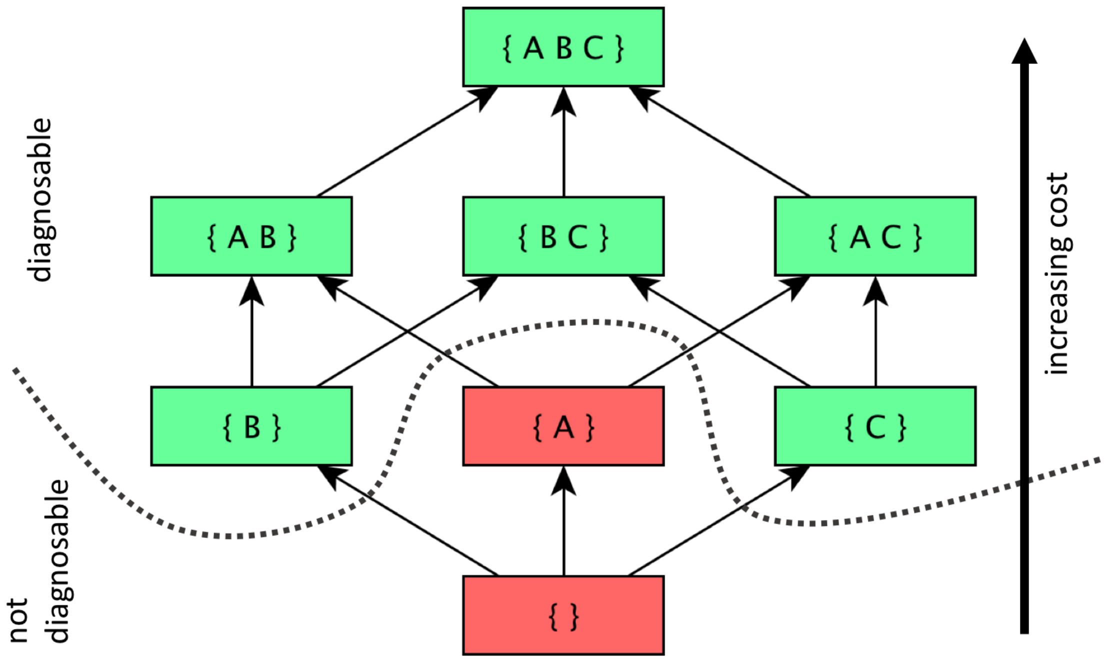
Timed Failure Propagation Graphs

Diagnosability Analysis

- Alarm Specification Language
- Verification of Diagnosability
- **Optimization of Observables**
- Implementation & Benchmarks

Conclusion

Synthesis of Observables



possible combinations of observables

What sets of observables guarantee diagnosability?

- usual synthesis approach: <u>**enumerative**</u>
- our proposal: <u>**symbolic**</u> approach
- optimization:
  - minimality
  - cost-optimality
- based on parameterized version of twin-plant

Introduction

Timed Failure Propagation Graphs

Diagnosability Analysis

- Alarm Specification Language
- Verification of Diagnosability
- Optimization of Observables
- **Implementation & Benchmarks**

Conclusion

Experiments

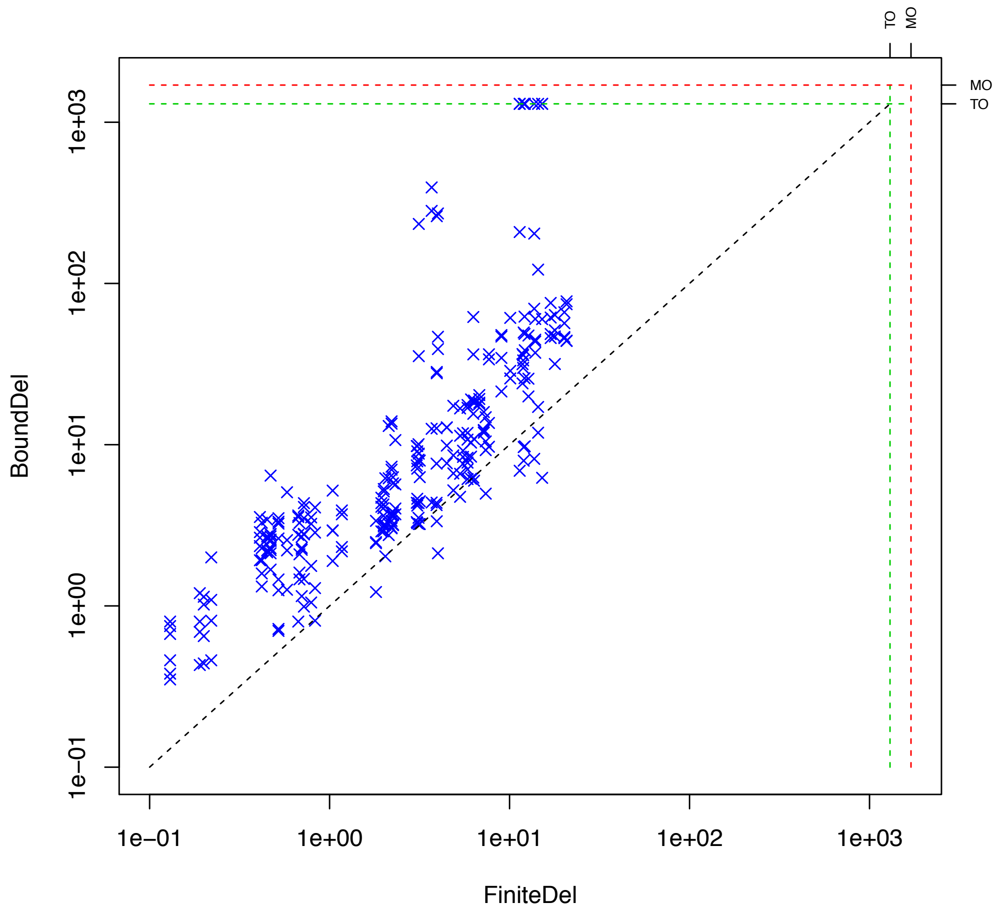
Implementation

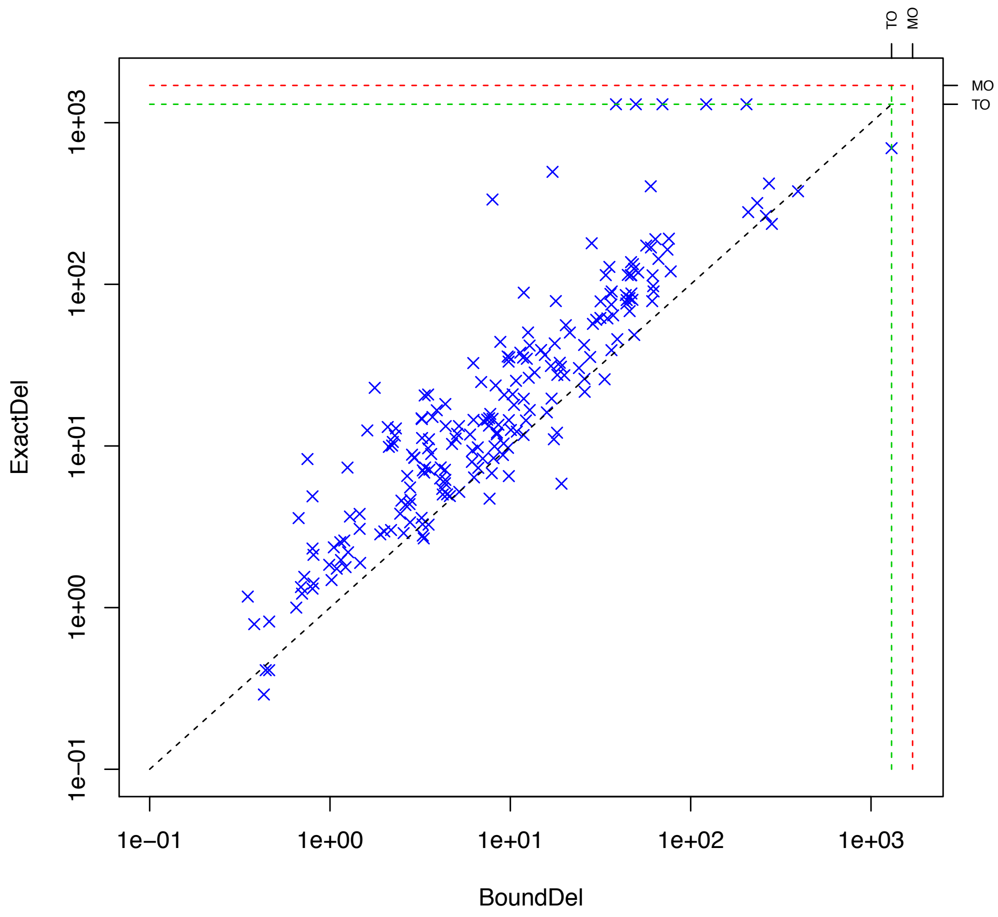
- implementation within xSAP, based on nuXmv
- standard LTL procedures for verification of diagnosability
- off-the-shelf parameter synthesis for synthesis of observables

Benchmark Models

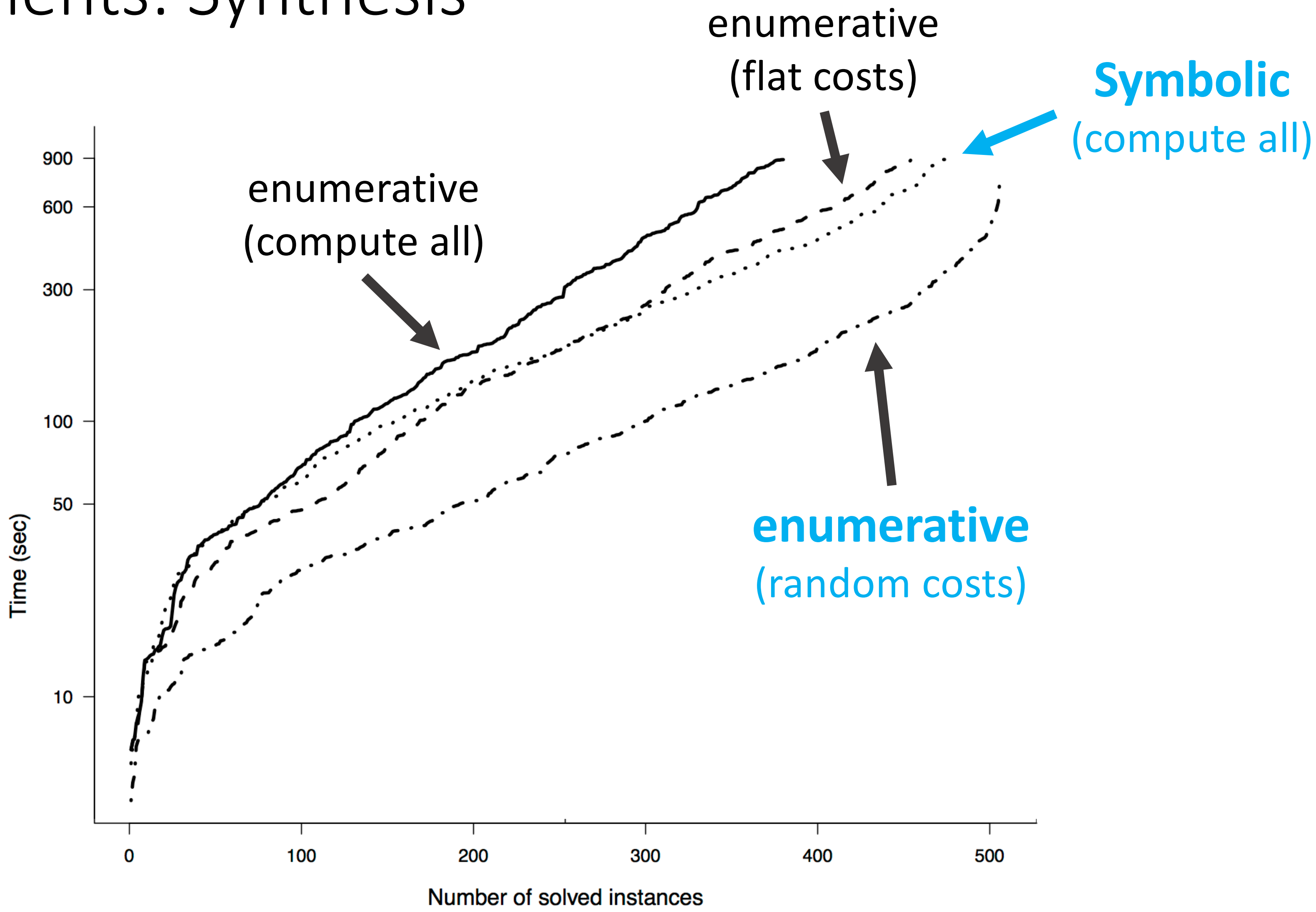
- similar to models used for TFPG experiments
- different timing model (one-step-one-tick vs. timestamps)

Experiments: Verification

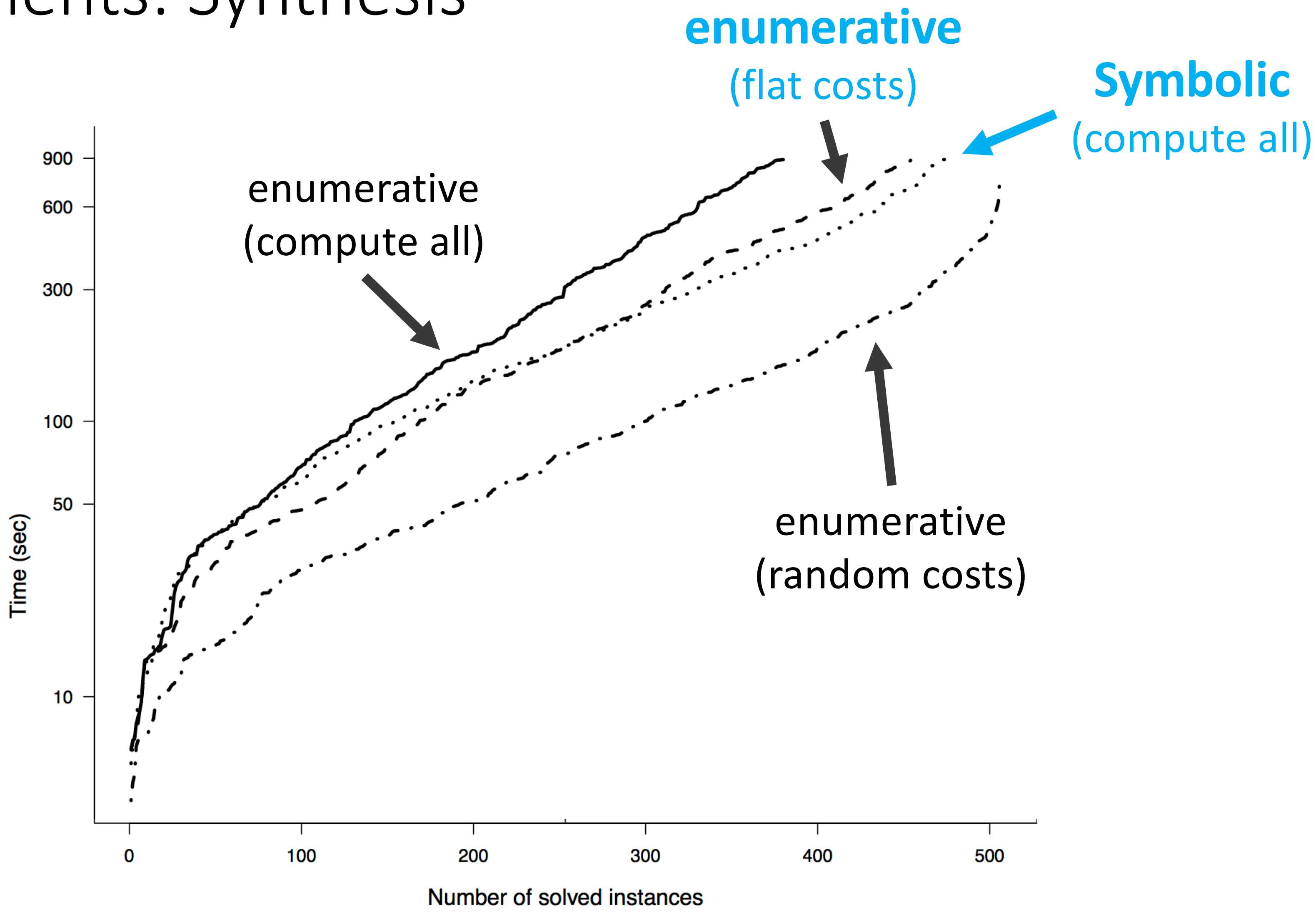




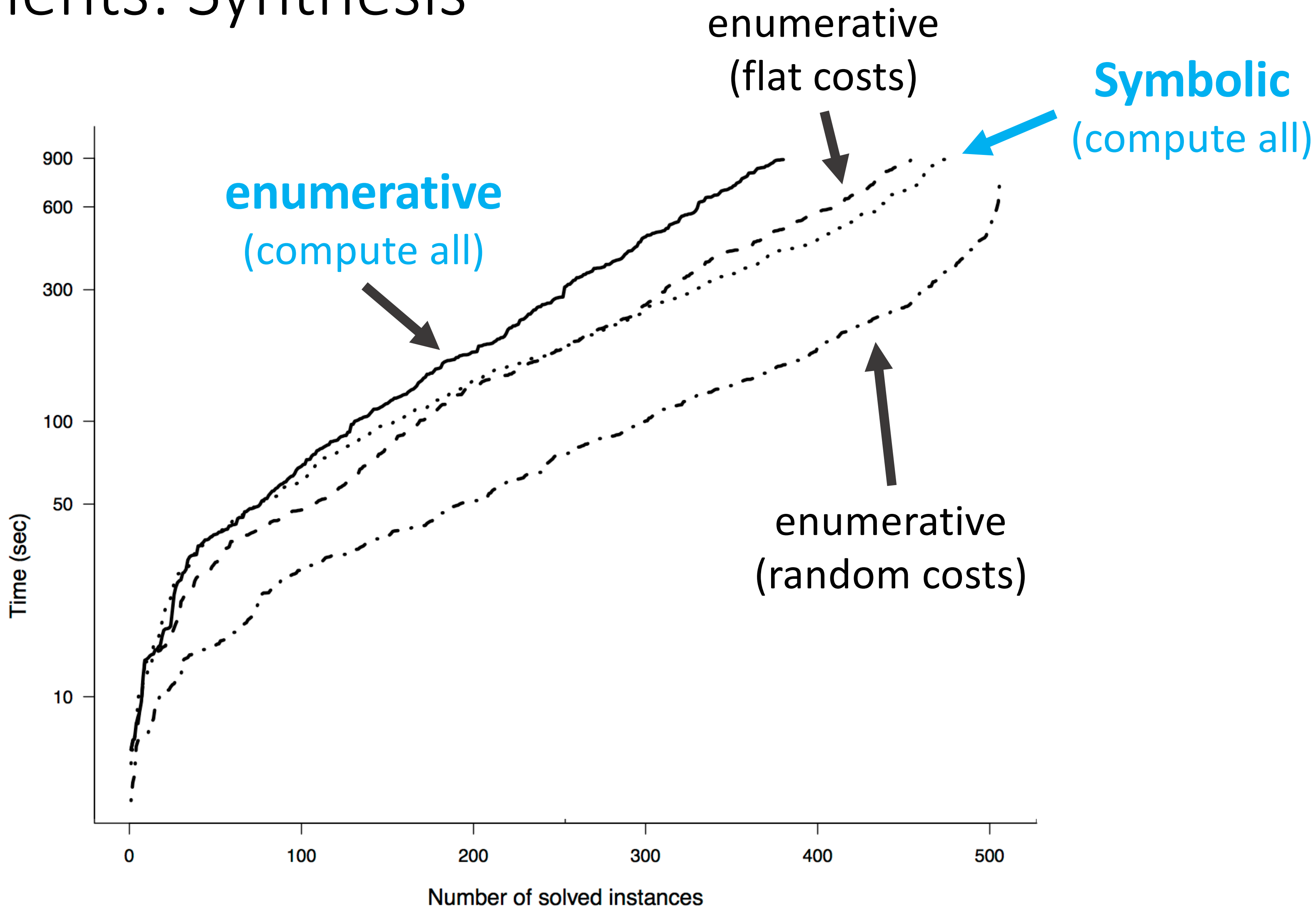
Experiments: Synthesis



Experiments: Synthesis



Experiments: Synthesis



Introduction

Timed Failure Propagation Graphs

Diagnosability Analysis

Conclusion

Contributions

Timed Failure Propagation Graphs [AAAI16, IJCAI16, TACAS16]

1. trace-based semantics for TFPGs
2. formal abstraction properties
3. validation w.r.t. system model
4. automated synthesis procedures
5. case studies on ESA satellite project

Diagnosability Analysis [AAAI12, FMCAD14, AIJ-TBS]

1. extension of alarm specification language with notion of context
2. twin-plant method to verify diagnosability
3. reduction of verification to model-checking
4. reduction of observables selection to parametric model-checking

Future Work

All Areas

- support for continuous time (hybrid automata)

Timed Failure Propagation Graphs

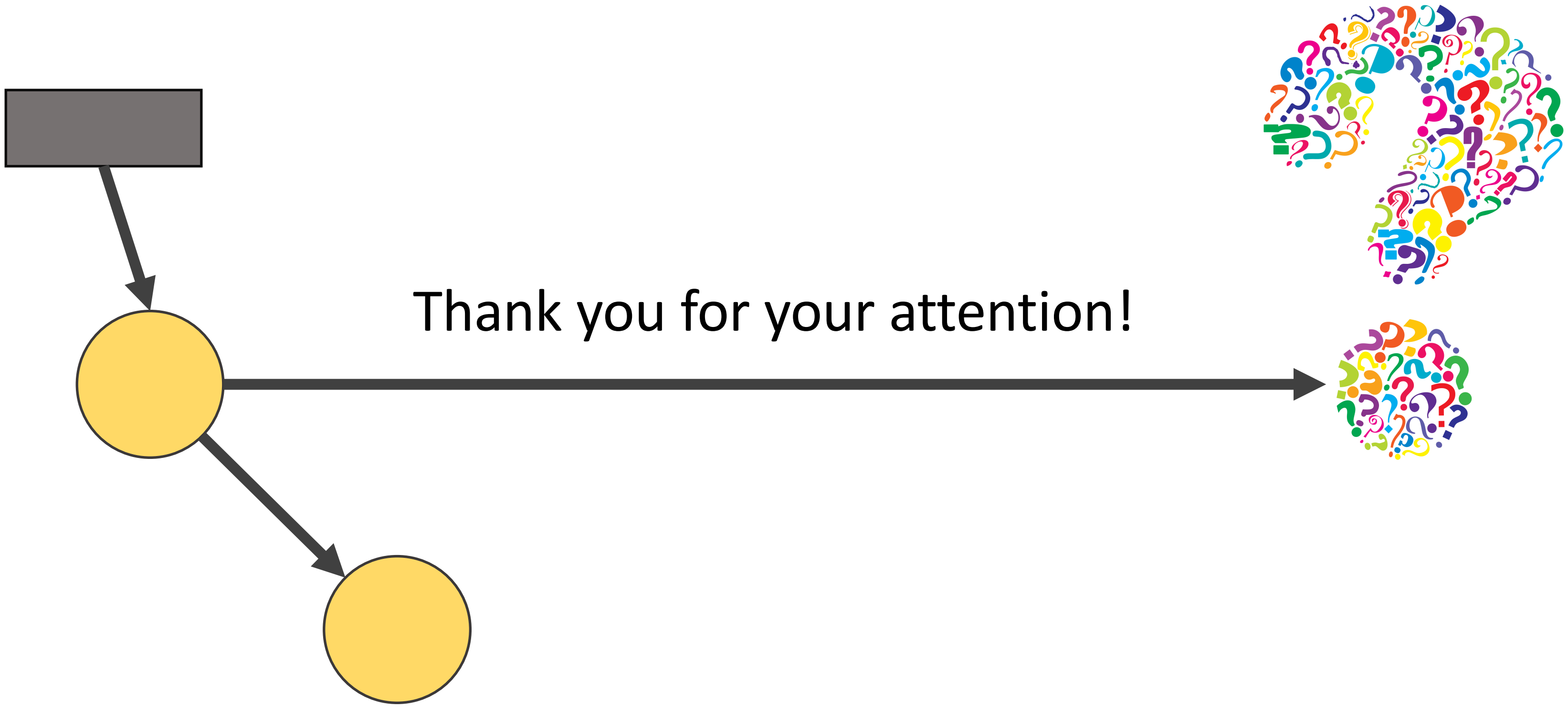
- performance improvements (compositional approach?)
- diagnosability-conscious synthesis

Diagnosability

- extend critical-pair approach to cover corner cases
- bounded-recall (history windows vs. full logs)

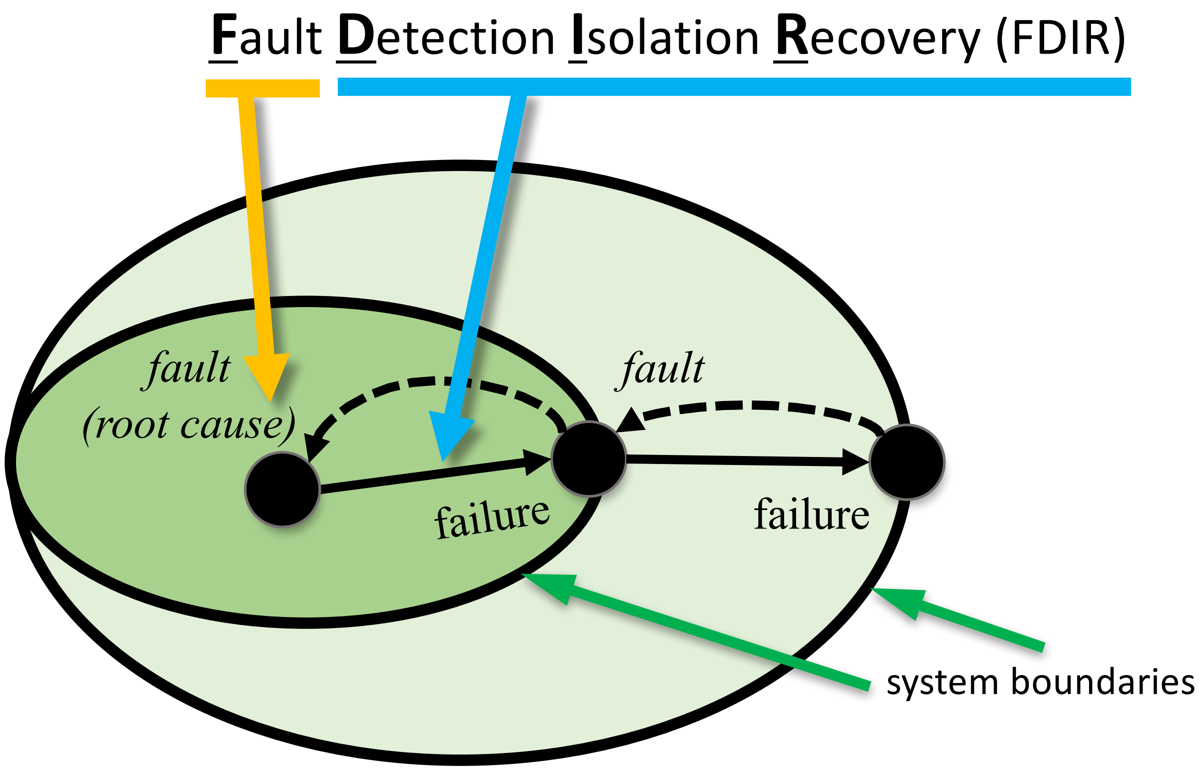
Recoverability

- formal specification language, feasibility analysis

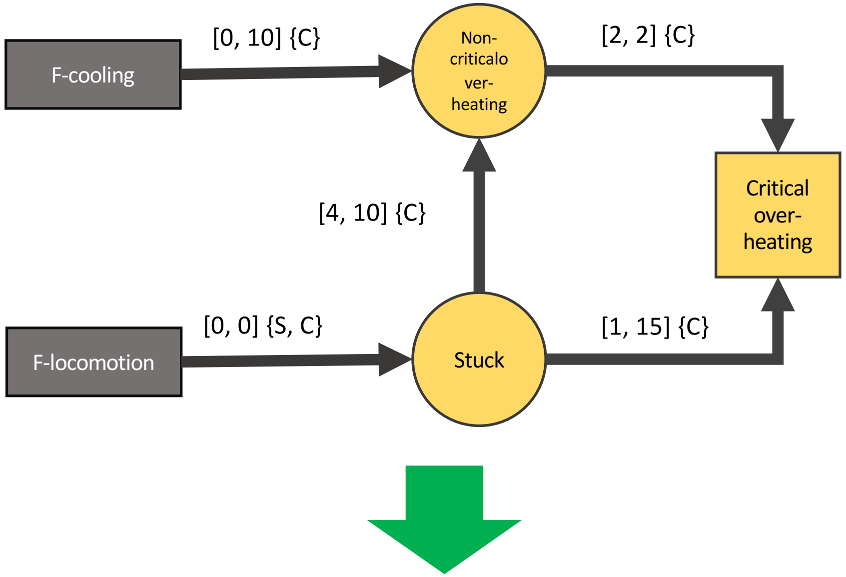

Thank you for your attention!

Appendix

Fault Management via FDIR



Diagnosability (and diagnoser synthesis) on TFPGs




```
MODULE main
VAR system_mode : {SafeZone, CriticalZone};
VAR forgerobot_failuremode_FailCooling :
failuremode (trans_type);VAR
forgerobot_failuremode_FailLocomotion :
failuremode (trans_type);
…
```

- synthesis of diagnoser via TFPG
- workflow
  - translate TFPG to transition system
  - analyze diagnosability (classical definition)
  - synthesize diagnoser
- evaluated by Thales Alenia Space on ExoMars TGO case study (FAME)

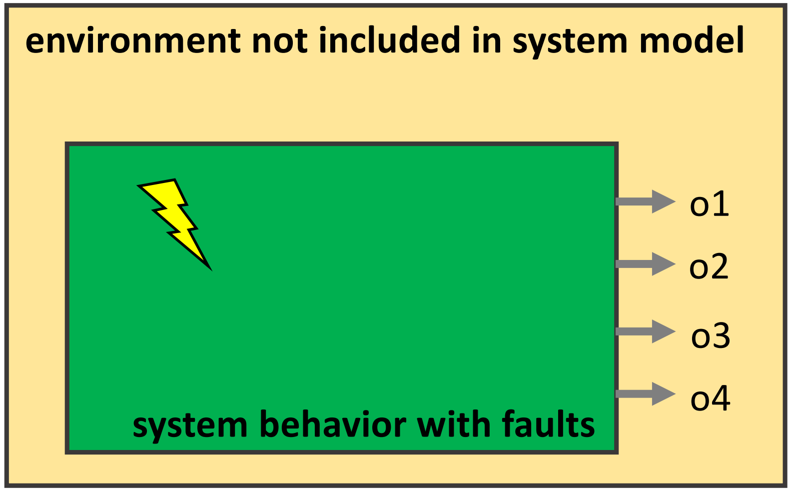
Operational Context

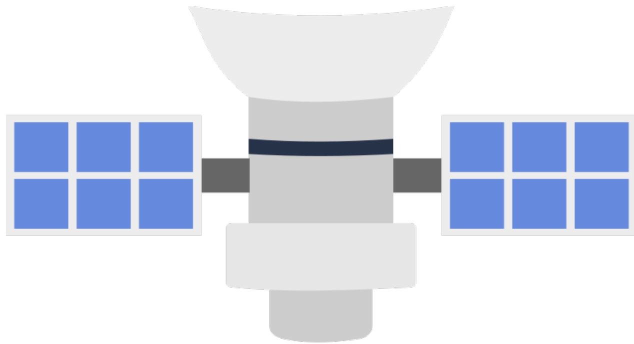
- diagnosability might depend on assumptions on the general environment (e.g. controller) not included in system model
- diagnosis context encoded in LTL

- $\psi \coloneqq G\ F\ (v.open \wedge v.in > 0)$
  Periodically, fluid is transferred into open valve.

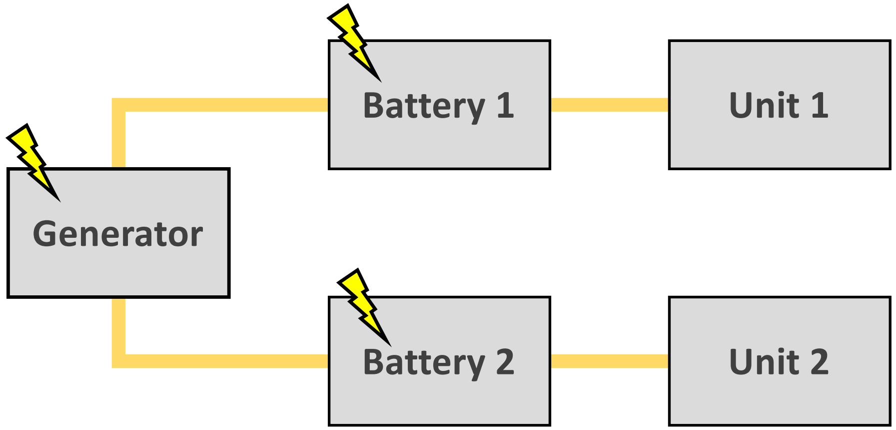
- $\psi \coloneqq G\ (\ sys.has\_power\ )$
  The system is always powered.

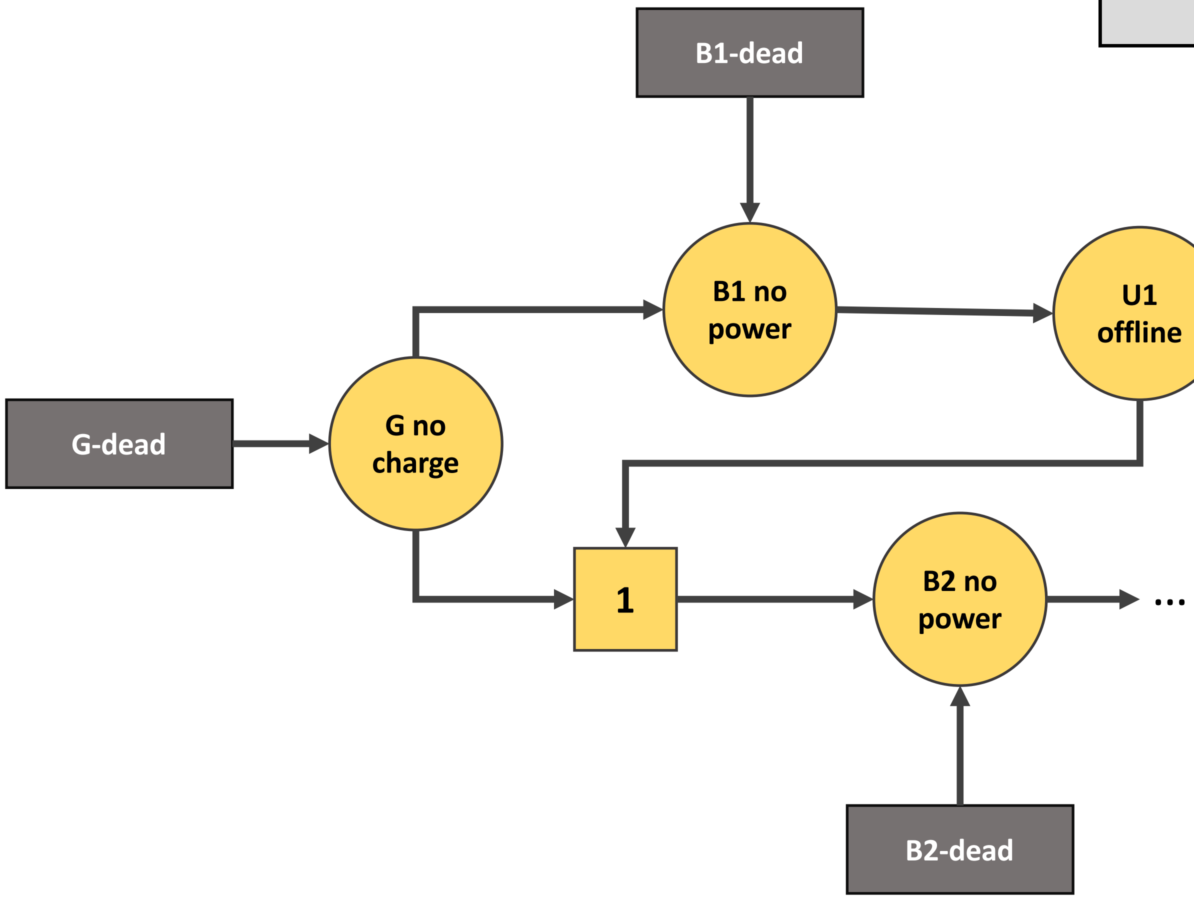
- $\psi \coloneqq F\ (\ engine.thrust = full\ )$
  The engines will eventually provide full thrust.

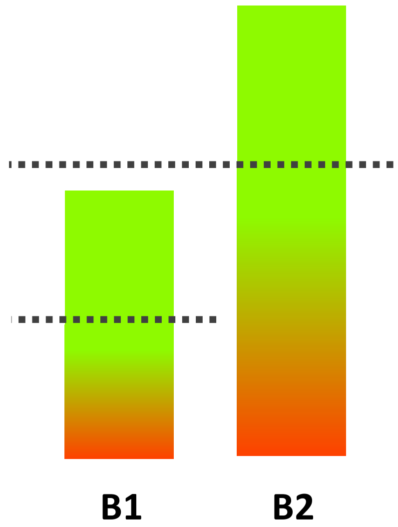




Power Supply System







Metric Temporal Logic

- syntax: $\phi ::= p|\neg\phi|\phi_1 \wedge \phi_2|\phi_1 \mathsf{U}^I \phi_2|\phi_1 \mathsf{S}^I \phi_2$
- intervals used in paper: $[a, \infty)$ and (a, ∞)
- point-wise semantics, interpreted on timed words:

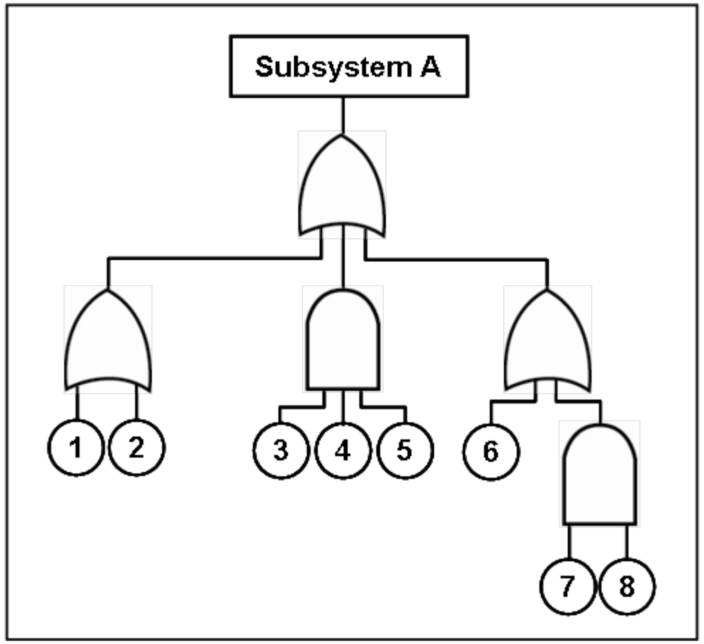
  $(s_0, \tau_0), (s_1, \tau_1), \ldots$
- since-operator: $\pi[k] \models \phi_1 \mathsf{S}^I \phi_2$ iff

  $\exists i \leq k \cdot \tau_k - \tau_i \in I$ and $\pi[i] \models \phi_2$ and $\forall i < j \leq k \cdot \pi[j] \models \phi_1$

Parametric Model-Checking

- parameters ***P***: lower/upper delay bounds
- parameterized system model: $\mathrm{M}_{\boldsymbol{P}}$
- model-checking problem: $\mathrm{M}_{\boldsymbol{P}} \vDash \bar{p} \rightarrow \Phi_{completeness}$
- reuse inductive invariant generated by model-checker
  - inductive invariant: over-approximation of reachable states
  - $I \Rightarrow INV$
  - $INV \wedge T \Rightarrow INV'$
  - use *INV* to check further candidates without calling model-checker
  - strengthen initial states and transition relation
    - $I \coloneqq I \wedge INV$
    - $T \coloneqq T \wedge INV$

Classical Failure Analyses



Item	**failure mode**	**Local effects**	**Subsystem effect**	**System Effect**
Brake Manifold	Internal Leakage	Decreased pressure	No Left Wheel Braking	Severely Reduced Aircraft deceleration

Monitoring?

Propagation delays?

AND/OR semantics?

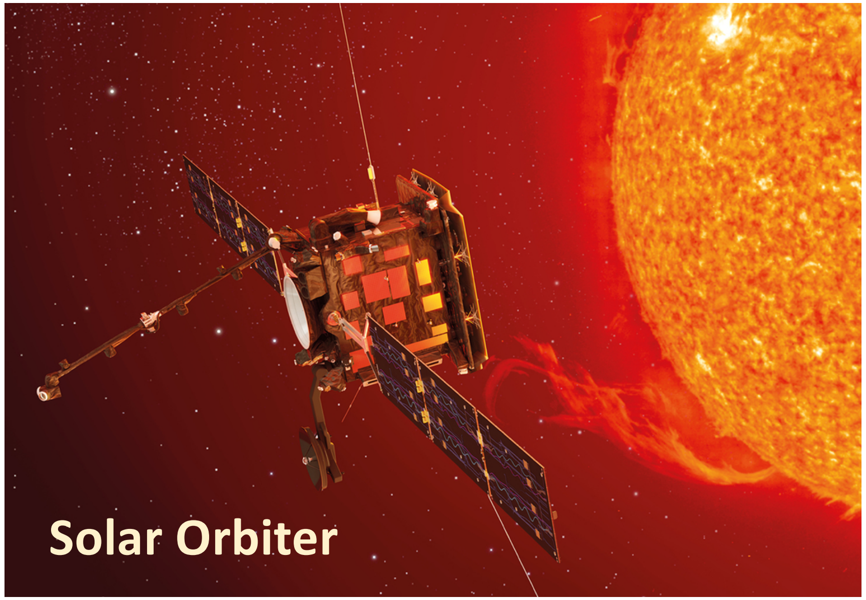
Granularity of modeling?

Mode constraints?

Using a single representation?

Contributions to FDIR Critical Design Review

- submission of five issues to SOLO FDIR CDR panel (4 major, 1 minor)
- the **need to be explicit** helped identifying ambiguities in documentation
- issues were disposed by
  - confirming our interpretations or providing detailed explanations
  - recognizing corner cases and confirming correct FDIR response
  - improving documentation



TFPGs: Related Work

- TFPG maturation with historical **maintenance data**
  - Strasser and Sheppard (2011)
  - estimate probability of missing/wrong edges, no time and mode information
  - cannot be applied at design time
- TFPG synthesis for local components from **data/control flow graph**
  - Dubey et al. (2013)
  - integration of component TFPGs based on component topology
  - non-functional interactions and dynamic evolution not captured
- TFPG synthesis for component behaviors modeled by **timed automata**
  - Priesterjahn et al. (2013)
  - no formal characterization of synthesis result, no validation algorithm
  - discrepancies bound to input/output ports of components
- TFPG standalone validation **without system model**
  - based e.g. on SMT-solving, Bozzano et al. (2015)

TFPGs in Academia and Industry

- TFPGs Definition
  - Misra et. al, DX Workshop 1992
  - Karsai, Abdelwahed, Biswas, AIAA-GNC 2003
- Use of TFPGs in industry (eg Boeing, NASA, ESA)
  - Hayden et. al, Diagnostic Technology Evaluation Report For On-Board Crew Launch Vehicle 2006
  - Ofsthun, Abdelwahed, Autotestcon 2007
  - Atlas et. al, IEEE Aerospace Conference 2001
- Applications of TFPGs
  - Misra et. al, SPIE IS Symposium 1994
  - Dubey, Karsai, Mahadevan, Dagstuhl Seminar 2010
  - Dubey, Karsai, Mahadevan, IEEE Aerospace Conference 2011
- Industrial projects
  - ESA COMPASS/FAME (with Thales Alenia Space)
  - ESA COMPASS/HASDEL (with Airbus Defence & Space)
  - internal case studies at OHB

Completeness Proof Obligations (OR nodes)

$$\psi_{\mathtt{OR}\cdot A}(d,\Gamma) := \mathsf{G}((\mathsf{O}\gamma_d) \rightarrow \mathsf{O}((\mathsf{O}\gamma_d) \wedge \bigvee_{e=(v,d)\in E} ((\mathsf{O}\gamma_v) \wedge \gamma_{\mu(e)}\mathsf{S}^{\geq tmin(e)}(\mathsf{O}\gamma_v) \wedge \gamma_{\mu(e)})))$$

whenever ***d*** activates — some ***e*** has been active for at least tmin

unexpected activations?

missed activations?

$$\psi_{\mathtt{OR}\cdot B}(d,\Gamma) := \mathsf{G}\neg(\bigvee_{e=(v,d)\in E} ((\mathsf{O}\gamma_v) \wedge \gamma_{\mu(e)} \wedge \neg(\mathsf{O}\gamma_d)\mathsf{S}^{>tmax(e)}((\mathsf{O}\gamma_v) \wedge \gamma_{\mu(e)} \wedge \neg(\mathsf{O}\gamma_d)))$$

never — some ***e*** is active for more than tmax — without ***d*** activating

TFPG is complete ***iff*** the proof obligations for all nodes hold on the system model.

Algorithm Graph Synthesis

Algorithm 1 TFPG-by-FTA

Inputs: system model S; set of failure modes F; set of discrepancies D; set of modes M; association map Γ.

Steps

instantiate each discrepancy $d \in D$ as an `OR` node;
instantiate each failure mode $f \in F$ as an `FM` node;
for all $d \in D$ **do**
    for all $mcs \in MCS(\gamma_d, \{\gamma_{d'} | d' \in F \cup D\}, S)$ **do**
        instantiate a fresh virtual `AND` node v
        create unconstrained edge (v, d)
        for all $\gamma_{v'} \in mcs$ **do**
            create unconstrained edge (v', v)
        end for
    end for
end for

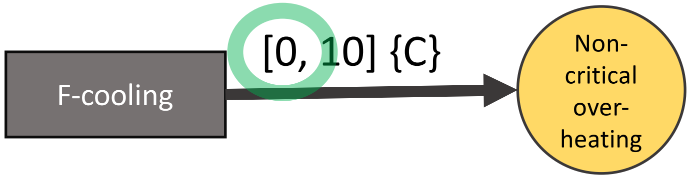
Simplification

$$\phi_{prec}(G) := \bigwedge_{d \in D} (\mathbf{d} \rightarrow \bigvee_{(v,d) \in E(G)} \bigwedge_{(v',v) \in E(G)} \mathbf{v}')$$

- express precedence constraints among user-defined nodes in Boolean formula
- use SAT solver to check if new propagation patterns are possible when removing edges (conjuncts)
- preserves completeness and graph correctness

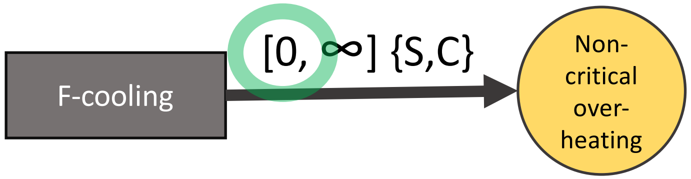
Tightness Checking

- Check if some edge parameter can be improved without breaking any completeness proof obligations.



$$\psi_{\mathtt{OR}\cdot A}(d,\Gamma) := \mathsf{G}((\mathsf{O}\gamma_d) \rightarrow \mathsf{O}((\mathsf{O}\gamma_d) \wedge \bigvee_{e=(v,d)\in E} ((\mathsf{O}\gamma_v) \wedge \gamma_{\mu(e)} \mathsf{S}^{\geq tmin(e)} (\mathsf{O}\gamma_v) \wedge \gamma_{\mu(e)})))$$

Automated Tightening

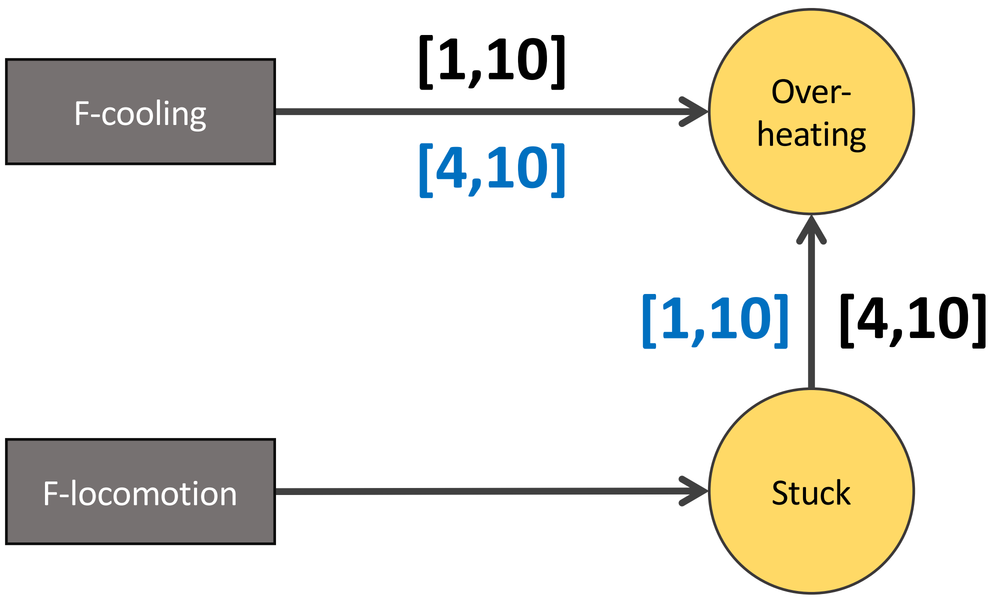


$$\psi_{\mathtt{OR}\cdot A}(d,\Gamma) := \mathsf{G}((\mathsf{O}\gamma_d) \rightarrow \mathsf{O}((\mathsf{O}\gamma_d) \wedge \bigvee_{e=(v,d)\in E} ((\mathsf{O}\gamma_v) \wedge \gamma_{\mu(e)} \mathsf{S}^{\geq tmin(e)} (\mathsf{O}\gamma_v) \wedge \gamma_{\mu(e)})))$$

Use parametric proof obligations to search for tight edge constraints.

Tightening – Multiple Solutions

- multiple tight solutions might exist
- connected with simultaneous propagations?
- to be investigated

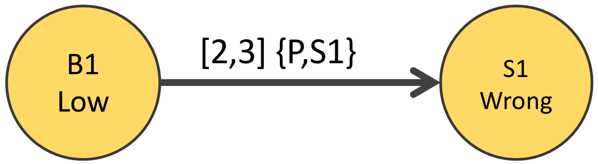


TFPG Tools: Implementation

- implemented in xSAP
  - back-end of COMPASS for model-based safety analysis
  - linked to nuXmv, symbolic model-checker for infinite-state transition systems
- behavioral validation
  - checking of MTL by reduction to reachability problems
- synthesis
  - precedence constraints computed via minimal cut-set procedures in xSAP
  - graph simplification via SAT-procedures of MathSAT
  - tightening via techniques from parametric model-checking

Reduction to Reachability

```
DEFINE EState := h_B1_LOW & (Mode_P | Mode_S1);
VAR ETime : real;
ASSIGN init(ETime) := case
       EState : 0;
       TRUE : -1;
       esac;
ASSIGN next(ETime) := case
       h_S1_WRONG : ETime;
       !next(EState) : -1;
       !EState & next(EState) : 0;
       TRUE : ETime + t_#delta;
       esac;
```



reduce MTL proof obligations to invariance properties:

- $h\_S1\_WRONG \rightarrow ETime \geq 2$
- $ETime \leq 3$

Experimental Evaluation

- finite-state system models
  - Acex/Autogen: derived from partially random graphs
  - PowerDist: power distribution management
  - Guidance: Space Shuttle engine contingency procedure
  - WBS: aircraft wheel-braking system (AIR6110)
  - X34: Livingstone model of experimental space-plane propulsion system
- infinite-state system models
  - ForgeRobot: model of robot working in industrial forge
  - Battery Sensor: running example
  - Cassini: spacecraft propulsion systems

Critical Pairs

- critical pair exists ➔ condition not diagnosable within time bound
- What if no critical pair exists?

Exact Delay	**Bounded Delay**	**Finite Delay**
Proves diagnosability.	Proves diagnosability if beta is a permanent condition (standard assumption). Else, guarantees diagnosability within 2d.	Proves diagnosability for finite-state models and context encoded as a safety property.

Critical Pairs as LTL formulae

$$G\ (\,twin_L.sys.has\_power\,) \land G(twin_R.sys.has\_power)$$
$$\land$$
$$F\ (ObsEq\ \land\ Y^{d}\ twin_L.fault\ \land\ H^{\leq 2d}\ \neg twin_R.fault)$$