Abstraction in Action: Bridging the Gap Between Formal Tools and Their Practical Use*

Alma L. Juarez-Dominguez, Nancy A. Day

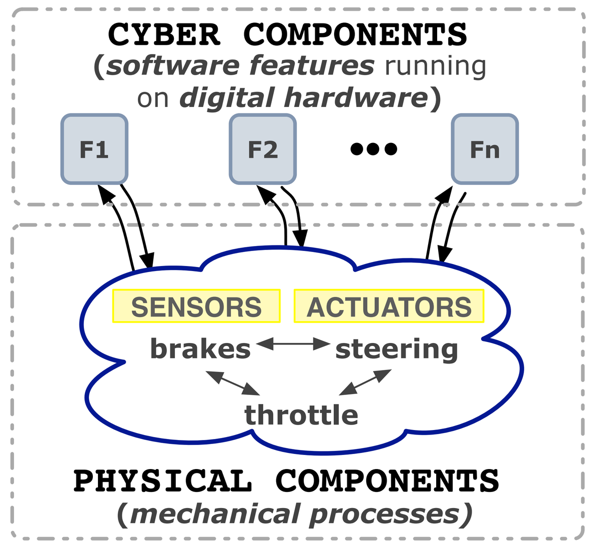
University of Waterloo

28th International System Safety Conference, Minneapolis, MN, USA (*Aug 30 - Sep 03, 2010*)

Motivation

- In a complex, electronic, software-intensive system, a ***feature*** is a bundle of system functionality as a user recognizes it.
  - ***Features*** are units that companies use to guide the design and marketing of the functionality offered to their customers.
  - ***Features*** often automate a manual process or activity.

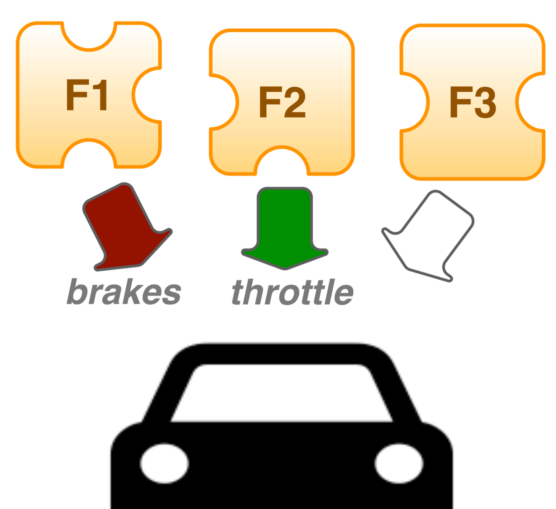
  Example: call waiting in telephony.



- In the automotive domain, an ***Advanced Safety Feature*** makes use of sensors, cameras, and even GPS devices to help the driver to be aware of dangers, and when possible, control the dynamics of the vehicle to avoid unsafe outcomes.

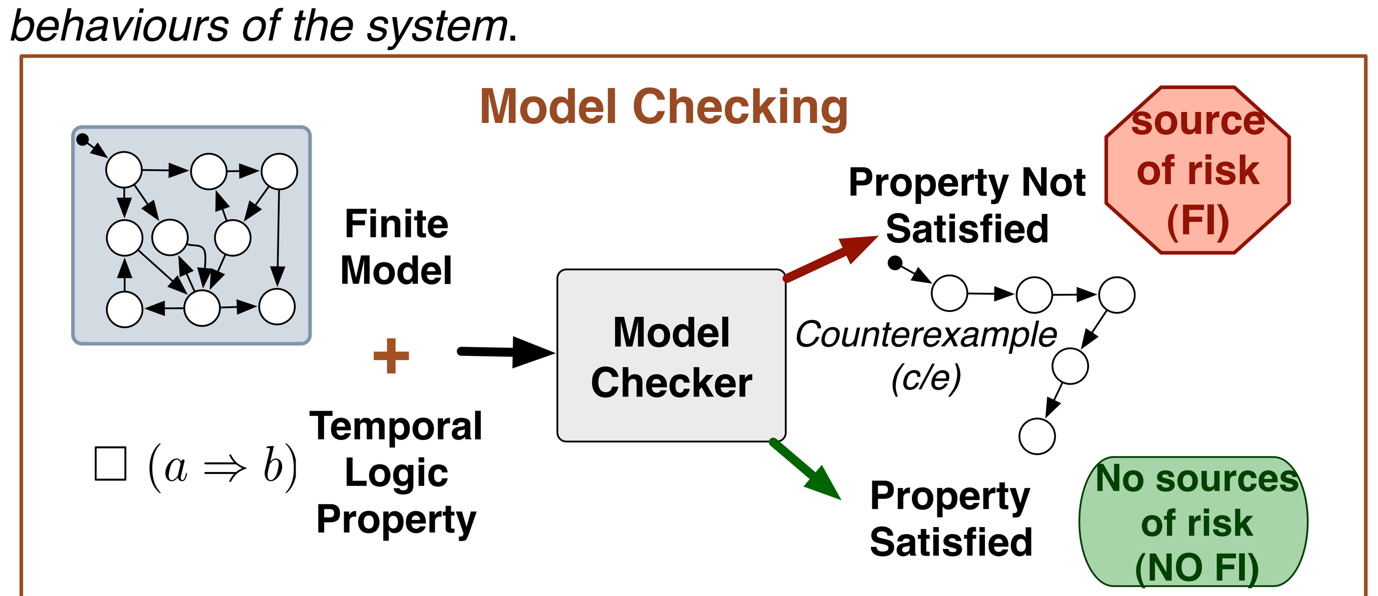
  Example: collision avoidance in a vehicle.

- In the automotive domain, a ***feature interaction (FI)*** arises from the activation of two or more features whose output requests to the actuators **create contradictory physical forces** on the mechanical processes, potentially at distinct times, that **cause unsafe outcomes** [1,2].
  - A ***feature interaction*** does not arise from the failure of individual components, but from the intended functionality of correctly implemented features.
  - ***Feature interactions*** are becoming more prevalent as systems increase in complexity, and can be a source of significant risk.



Example: when one feature requests to apply the ***brakes*** as another feature requests to apply the ***throttle***.

- To ***detect feature interactions*** in the automotive domain, we use **symbolic model checking** at design time, which allows us to *examine* ***exhaustively*** *all behaviours of the system.*



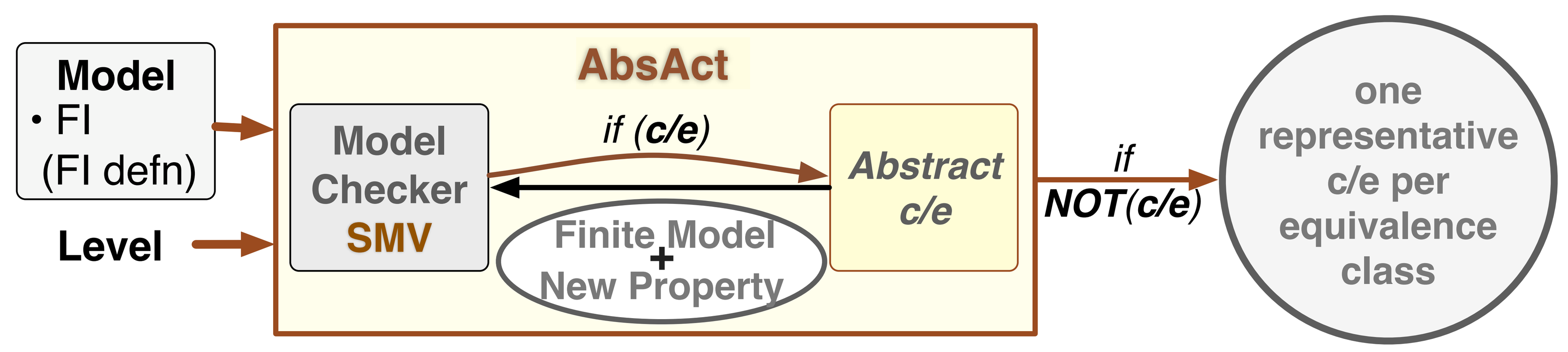
- It is **advantageous** to find ***all paths*** that **do not satisfy the property** before fixing the model because the correction may depend on several factors that can only be recognized by looking at ***all*** counterexamples.
  - The set of ***all*** counterexamples is often **too large** to **generate** and **comprehend**.

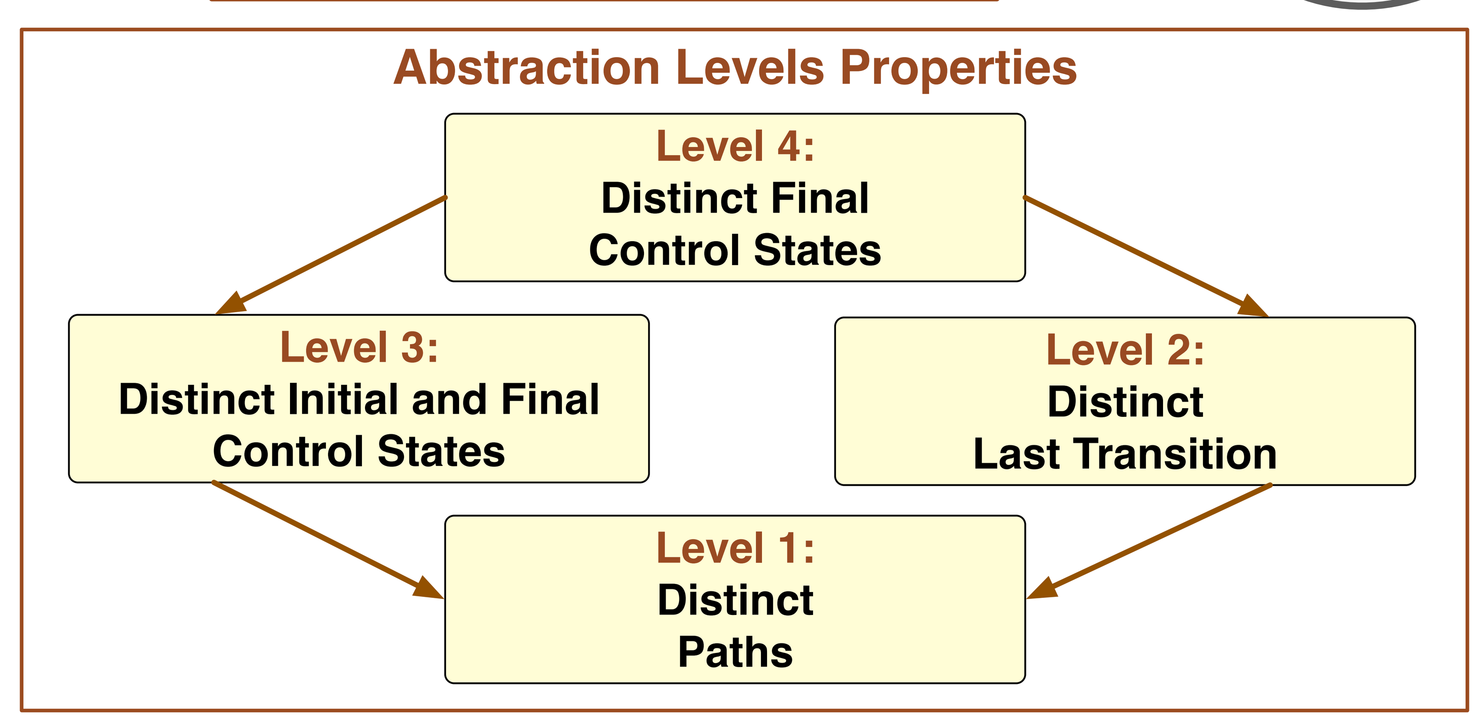
Methodology

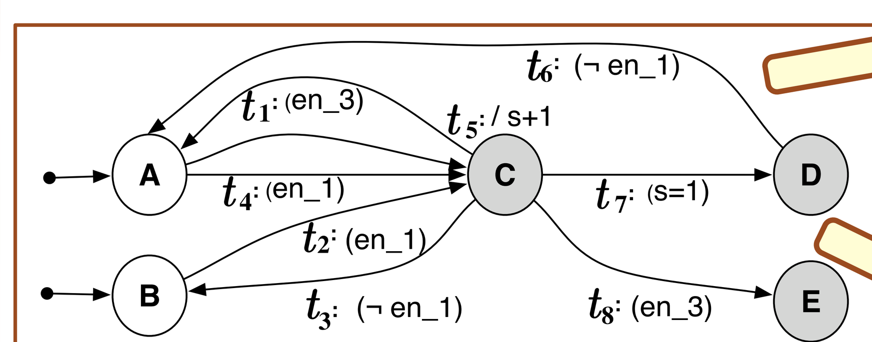
- **Goal:** Use of our tool to put ***"abstraction in action"*** by generating only a **reduced number of cases** that **represent all** possible **sources of risk** (i.e., **errors, inconsistencies, contradictions, feature interactions**) in a system.
  - We **abstract the counterexamples** based on the **modelling concepts** of **control states and transitions**.
  - Our **tool AbsAct** uses the ***model checker SMV*** [3] in an iterative process to generate all sources of risks (counterexamples).
  - For **every abstraction level**, we begin by checking the property:

    $$\text{Globally}(\neg \textbf{FI})$$

    For ***each iteration***, when a counterexample is generated, we **abstract** it by representing its **equivalence class** using a linear temporal logic (LTL) [4] formula according to the desired level of abstraction. Then, we **disjunct** this LTL expression with the original property **and** the LTL expressions representing previously generated counterexamples (equivalence classes).

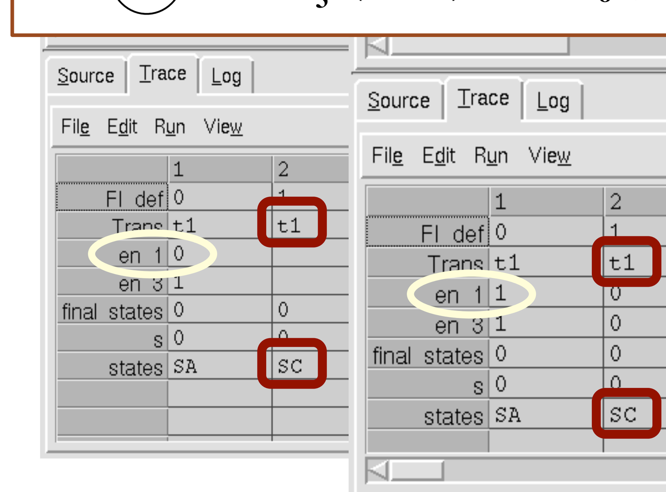






C,D and **E** are **control states** where an **FI** occurs

Total number of counterexamples: 183 (no abstraction)



Example: Path $\langle t_1 \rangle$ allows us to reach the critical state **C**, even when `en_1` is 1 or 0. These **variations** do not add to the understanding of the risk, but are generated by the model checker if no abstraction is used.

Level 1	Level 2	Level 3	Level 4
$[\langle t_1 \rangle]$ - 43	$[t_1]$ - 53	[**A**,**C**] - 58	[**C**] - 86
$[\langle t_4 \rangle]$ - 15	$[t_4]$ - 21		
$[\langle t_2 \rangle]$ - 28	$[t_2]$ - 12	[**B**,**C**] - 28	
$[\langle t_1, t_7 \rangle]$ - 21	$[t_7]$ - 67	[**A**,**D**] - 45	[**D**] - 67
$[\langle t_4, t_7 \rangle]$ - 24			
$[\langle t_2, t_7 \rangle]$ - 22		[**B**,**D**] - 22	
$[\langle t_1, t_8 \rangle]$ - 12	$[t_8]$ - 30	[**A**,**E**] - 18	[**E**] - 30
$[\langle t_4, t_8 \rangle]$ - 6			
$[\langle t_2, t_8 \rangle]$ - 12		[**B**,**E**] - 12	

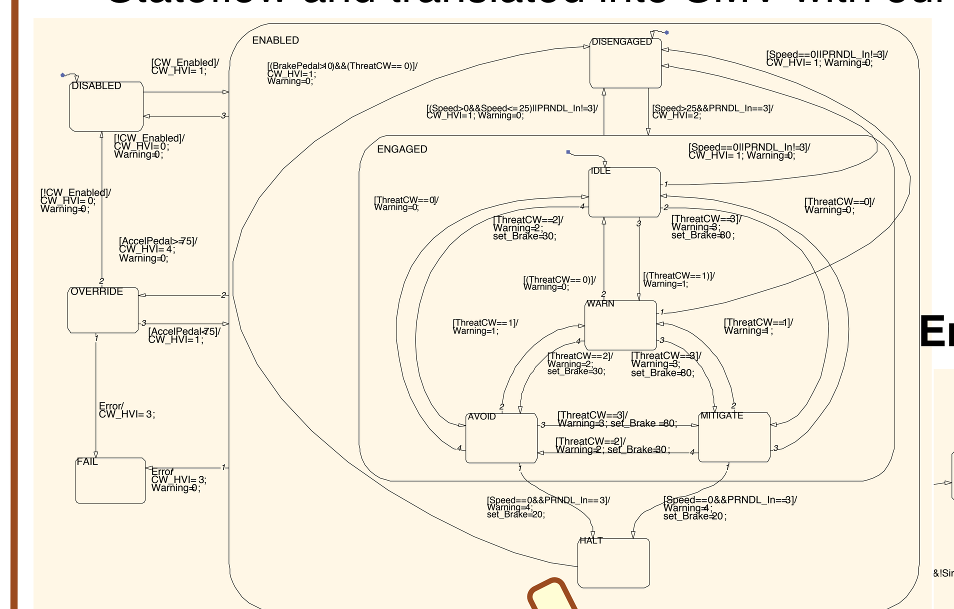
Equivalence classes by abstraction level and number of counterexamples abstracted per equivalence class

Contributions

- ***Development of the tool AbsAct***: ***Automatic generation*** of a **reduced number of cases** that **represent all** possible **sources of risk** in a system during model checking iterations.
- ***Definition of a scale of abstraction levels***: Representation of equivalence classes as **LTL properties**, which each **create equivalence classes of counterexamples**.

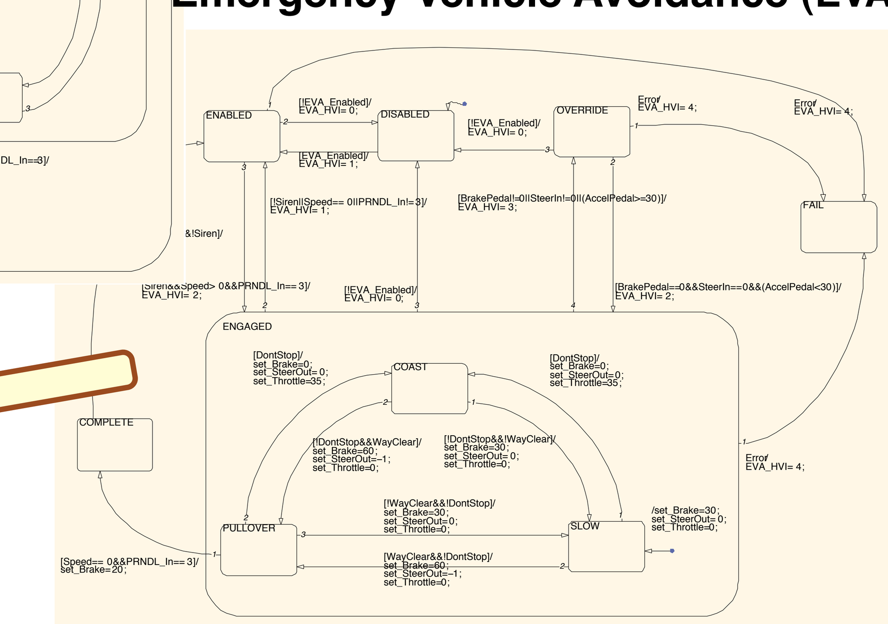
Case Study

- Detection of **sources of risks** in automotive features: **Collision Avoidance** (**CA**) and **Emergency Vehicle Avoidance** (**EVA**), designed in Matlab's Stateflow and translated into SMV with our tool **mdl2smv** [5].



Collision Avoidance (CA)

Emergency Vehicle Avoidance (EVA)



Condition for an **FI** to occur: (set_Brake > 60)

	CA			EVA		
	Iterations	BDD Nodes	Time	Iterations	BDD Nodes	Time
Level 4	1	5658	7.5s	1	4557	7.5s
Level 3	1	1043	7.6s	1	8477	7.6s
Level 2	3	10656	8.0s	2	8543	7.8s
Level 1	5	305495	14.2s	2	35874	8.3s

	Iterations	Equiv. Classes	BDD Nodes	Time
Level 4	3	3	846	7.4s
Level 3	6	6	2359	7.8s
Level 2	5	5	1653	7.6s
Level 1	7	9	18308	8.3s

Number of cycle iterations, the number of equivalence classes discovered per level, the maximum BDD size for all cycles, and the total time for all iterations

References

[1] A. L. Juarez-Dominguez, J. J. Joyce, and R. Debouk. *Feature interaction as a source of risk in complex software-intensive systems.* Proc. of the 25th Int. System Safety Conference, 2007

[2] M. Broy, I. H. Krüger, A. Pretschner, and C. Salzmann. *Engineering Automotive Software.* Proc. of the IEEE. Vol. 95, No. 2, February 2007

[3] K. L. McMillan. Symbolic model checking. Kluwer Academic, 1993.

[4] Z. Manna and A. Pnueli. The Temporal Logic of Reactive and Concurrent Systems: Specification. Springer-Verlag, First edition, 1992.

[5] A. L. Juarez-Dominguez, N. A. Day and J. J. Joyce. *Modelling Feature Interactions in the Automotive Domain.* Proc. of the Int. Workshop on Modelling in Software Engineering, 2008

*A. L. Juarez-Dominguez, and N. A. Day. *On-the-Fly Counterexample Abstraction for Model Checking Invariants.* Technical Report, Cheriton School of Computer Science, University of Waterloo. June, 2010. http://www.cs.uwaterloo.ca/~aljuarez/Docs/TR_June2010.pdf

More information is available at: **http://www.cs.uwaterloo.ca/~aljuarez/research.html**