A brief introduction to timed automata

part A: why?

Mariëlle Stoelinga

Formal Methods & Tools
Software systems correctness: It’s about time

**Airbags**
- Inflator
- Crash Sensor
- Nitrogen Gas

**Medical**
- WCET: very low

**Networks**
- Jitter, packet delay

**Trains**
- Gates must close on time

**KEY:** Are these systems correct?
- Meet timing requirements, eg deadlines?
- TA modeling & analysis!
Software systems correctness: **model-based design**

**Validation & verification**
- Model Checking:
  - Synchronous Data Flow
  - Timed Automata (TA)
  - Markov Chains

**Boehm's cost of SW errors**

- Requirements
- Design
- Coding
- Testing
- Deployment
Engineering correct systems: the model checking approach

- formalize system model (TA)
- model checker
- State space reduction
- debugging
- violated + counter example
- out of memory
- satisfied
- meets??

Requirements
- desired property
- • model checking is effective to show system correctness
- • BUT: MC is as good as the model you provide
Engineering correct systems: The model checking approach

System

- Model

meets??

Safety Requirements

Invariant:
at_crossing → gates closed

- System invariant:
  - at_crossing

Model checker

- Model
  - GoDown
  - Lowering
  - Open
  - Gate
  - Hg ≤ 10

- Violated + counter example
  - Out of memory

- Satisfied
Summary

This screen cast

- Timing behavior is crucial in systems correctness
- Model checking
  - Does a system model meets a correctness property?
  - Fully automatic technique
  - As good as models you provide
- Timed automata
  - Model checking real-time behavior

Next screen cast

- Details of TA model
- Details of property specification logic
- Small, realistic example: rail road controller
  - Model checking real-time behavior
Thank you for your attention
& See you next time!