

Core Course Preview

Introduction to Model Checking

<https://clegra.github.io/mc/mc.html>

Emilio Tuosto

<https://cs.gssi.it/emilio.tuosto/>

Clemens Grabmayer

<https://clegra.github.io>

Department of Computer Science



December 11, 2025

Model Checking

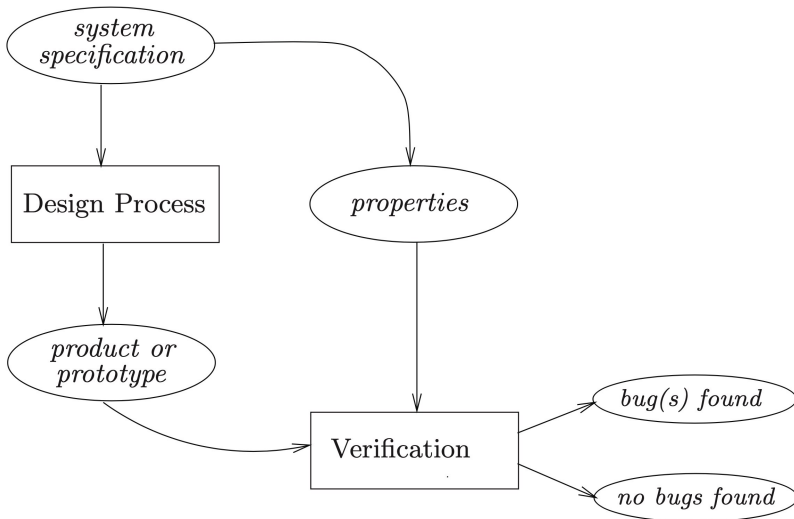
... is an effective automatable technique:

- ▶ *to expose potential software design errors;*
- ▶ *that, given a finite-state model of a system and a formal property, systematically checks whether this property holds for that model.*

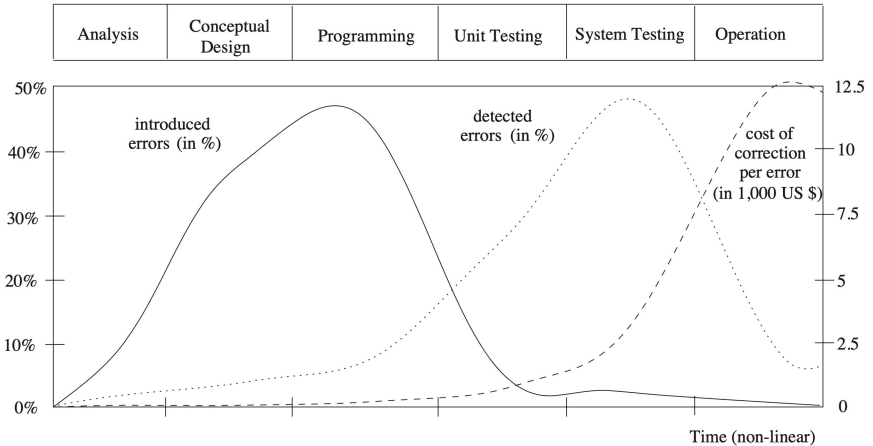
Strengths:

- ▶ widely applied in industry
for: embedded systems, software engineering, hardware design, explainable AI
- ▶ supports **partial verification** (of system parts)
- ▶ provides **diagnostic information** for debugging
- ▶ has sound **mathematical underpinning** (logic and process theory)

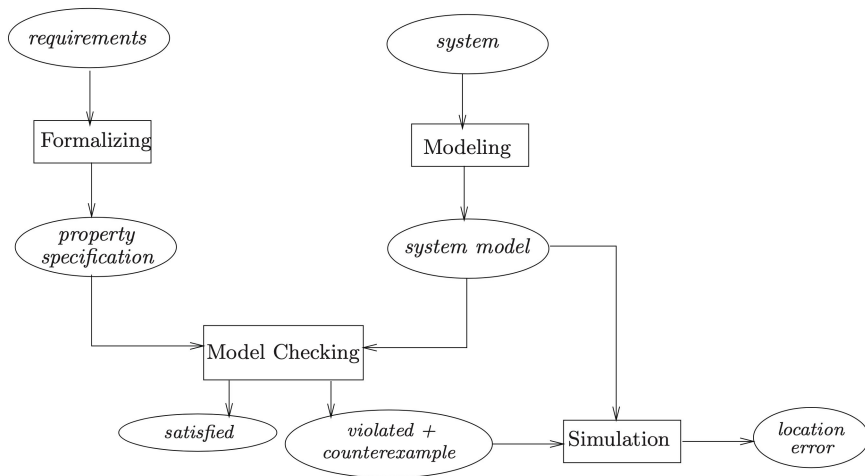
Hard-/Software Verification (traditionally)



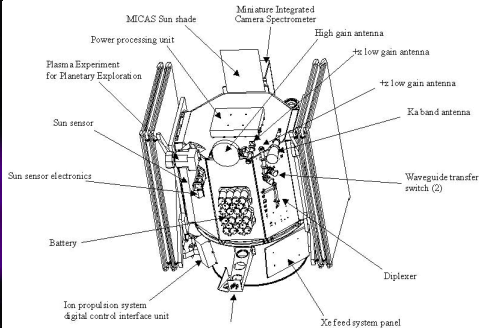
Error introduction, detection, and repair costs



Model checking



Deep Space 1



- ▶ Flyby of asteroid 9969 Braille (1999)
- ▶ Entered the coma of Comet Borrelly (2001)
- ▶ Model checking discovered **5 concurrency errors**

Example (program concurrency/non-determinism)

Programs [Inc](#), [Dec](#), and [Reset](#) cooperate, and use a shared variable x :

proc [Inc](#)

while true

do

if $x < 200$

then $x := x + 1$

fi

od

proc [Dec](#)

while true

do

if $x > 0$

then $x := x - 1$

fi

od

proc [Reset](#)

while true

do

if $x = 200$

then $x := 0$

fi

od

Example (program concurrency/non-determinism)

Programs [Inc](#), [Dec](#), and [Reset](#) cooperate, and use a shared variable x :

```
proc Inc
  while true
    do
      if  $x < 200$ 
        then  $x := x + 1$ 
      fi
    od
```

```
proc Dec
  while true
    do
      if  $x > 0$ 
        then  $x := x - 1$ 
      fi
    od
```

```
proc Reset
  while true
    do
      if  $x = 200$ 
        then  $x := 0$ 
      fi
    od
```

Question: Is $0 \leq x \leq 200$ always guaranteed?

Modeling (by labeled transition systems)

proc [Inc](#)

while true

do

if $x < 200$

then $x := x + 1$

fi

od

proc [Dec](#)

while true

do

if $x > 0$

then $x := x - 1$

fi

od

proc [Reset](#)

while true

do

if $x = 200$

then $x := 0$

fi

od

Modeling (by labeled transition systems)

proc [Inc](#)

while true

do

1: **if** $x < 200$

2: **then** $x := x + 1$

fi

od

proc [Dec](#)

while true

do

1: **if** $x > 0$

2: **then** $x := x - 1$

fi

od

proc [Reset](#)

while true

do

1: **if** $x = 200$

2: **then** $x := 0$

fi

od

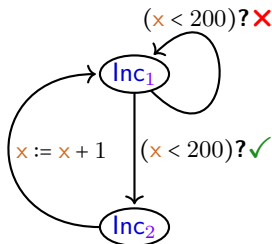
Modeling (by labeled transition systems)

proc Inc

while true

do

1: **if** $x < 200$
 2: **then** $x := x + 1$
fi
od

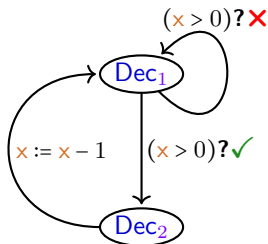


proc Dec

while true

do

1: **if** $x > 0$
 2: **then** $x := x - 1$
fi
od

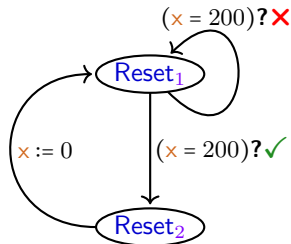


proc Reset

while true

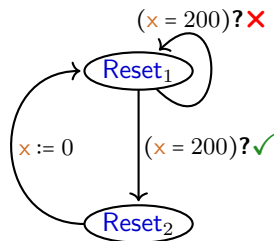
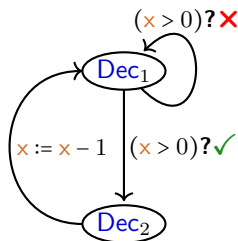
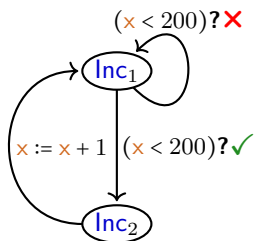
do

1: **if** $x = 200$
 2: **then** $x := 0$
fi
od



Labeled transition systems (LTSs)

Formalizing properties (in temporal logic)



$$Inc_1 \parallel Dec_1 \parallel Reset_1 \stackrel{?}{\models} \Box(0 \leq x \wedge x \leq 200) \quad (\text{Linear-TL formula})$$

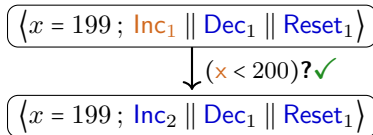
Counterexample (offending execution trace)

$$\langle x = 199 ; \text{Inc}_1 \parallel \text{Dec}_1 \parallel \text{Reset}_1 \rangle$$

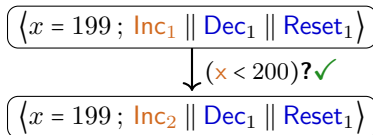
Counterexample (offending execution trace)

$$\langle x = 199 ; \text{Inc}_1 \parallel \text{Dec}_1 \parallel \text{Reset}_1 \rangle$$

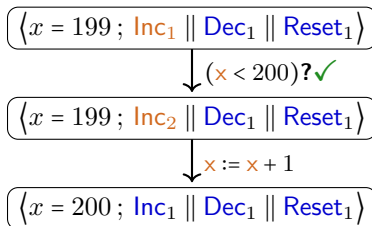
Counterexample (offending execution trace)



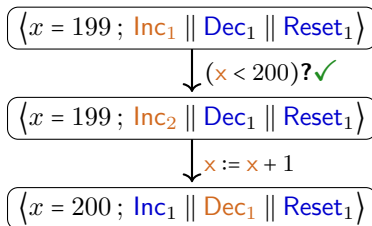
Counterexample (offending execution trace)



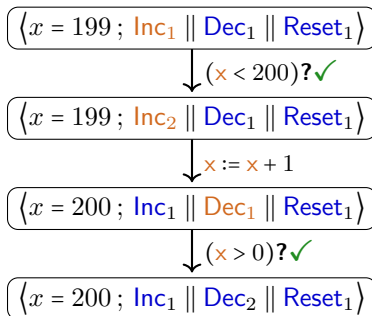
Counterexample (offending execution trace)



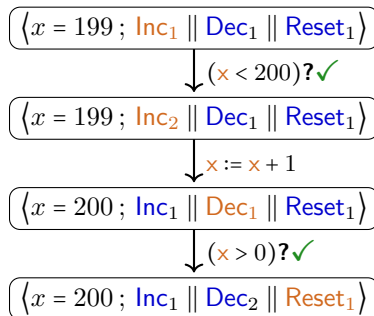
Counterexample (offending execution trace)



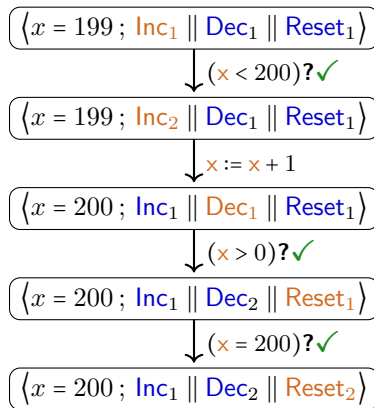
Counterexample (offending execution trace)



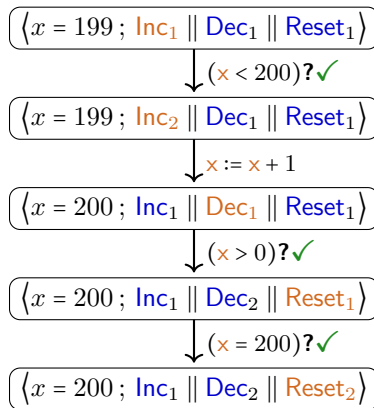
Counterexample (offending execution trace)



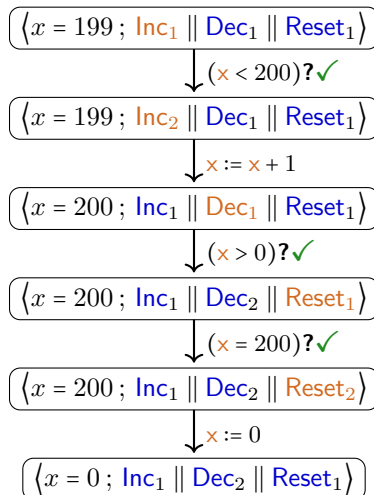
Counterexample (offending execution trace)



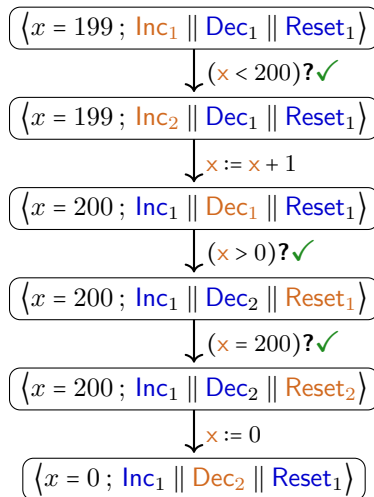
Counterexample (offending execution trace)



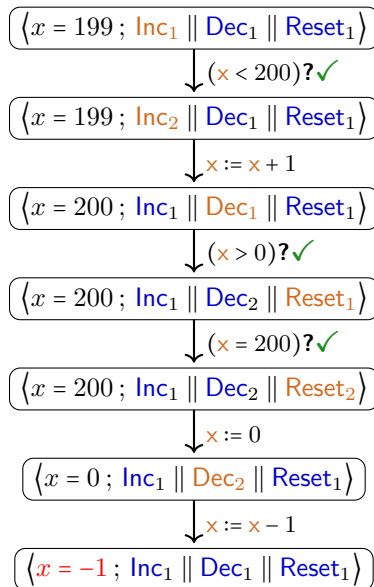
Counterexample (offending execution trace)



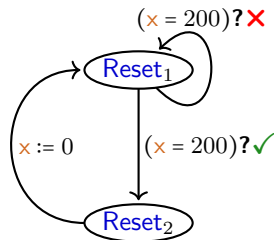
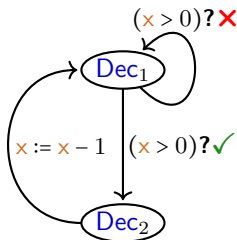
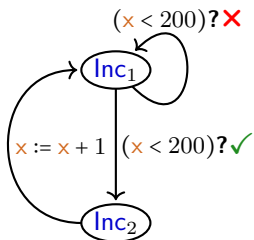
Counterexample (offending execution trace)



Counterexample (offending execution trace)

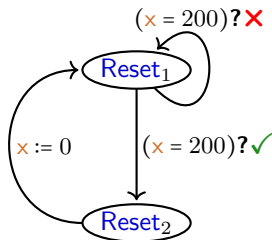
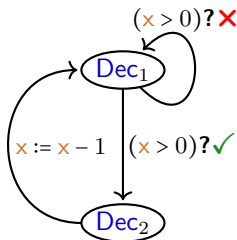
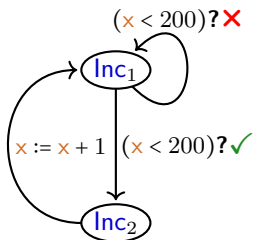


Formalizing properties (in temporal logic)



$\text{Inc}_1 \parallel \text{Dec}_1 \parallel \text{Reset}_1 \not\models \Box(0 \leq x \wedge x \leq 200)$ (Linear-TL formula)

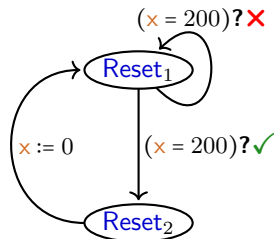
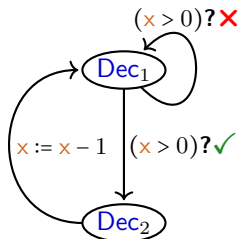
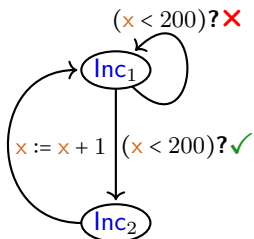
Formalizing properties (in temporal logic)



$\text{Inc}_1 \parallel \text{Dec}_1 \parallel \text{Reset}_1 \not\models \Box(0 \leq x \wedge x \leq 200)$ (Linear-TL formula)

$\text{Inc}_1 \parallel \text{Dec}_1 \parallel \text{Reset}_1 \models \Diamond(x < 0)$ (LTL formula)

Formalizing properties (in temporal logic)



$Inc_1 \parallel Dec_1 \parallel Reset_1 \not\models \Box(0 \leq x \wedge x \leq 200)$ (Linear-TL formula)

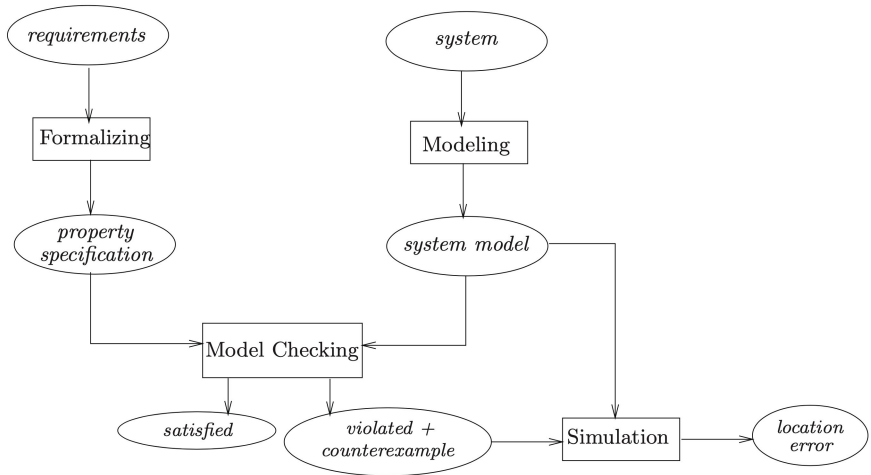
$Inc_1 \parallel Dec_1 \parallel Reset_1 \models \Diamond(x < 0)$ (LTL formula)

$Inc_1 \parallel Dec_1 \parallel Reset_1 \not\models \forall \Box(0 \leq x \wedge x \leq 200)$ (Computation-Tree-L formula)

$Inc_1 \parallel Dec_1 \parallel Reset_1 \models \exists \Box(0 \leq x \wedge x \leq 200)$ (CTL formula)

$Inc_1 \parallel Dec_1 \parallel Reset_1 \models \forall \Box \exists \Diamond(x < 0)$ (CTL formula)

Model checking

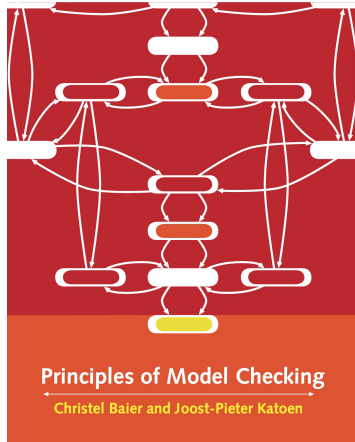


Any [such] verification is only as good as the model of the system.

Topics of the course

- ▶ modeling systems by **labeled transition systems (LTSs)**
- ▶ **fairness**
- ▶ **Linear Temporal Logic (LTL)**
 - ▶ model checking formulas
 - ▶ express properties by **Büchi automata**
 - ▶ model check LTSs and properties via **product automata**
- ▶ **Computation Tree Logic (CTL)**
- ▶ **partial** model checking
 - ▶ partially known systems (state properties/states/transitions)
- ▶ analysing system behavior with the **Maude model-checker**

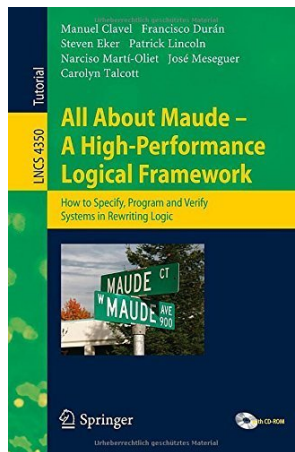
Book



- pdf available:

https://is.ifmo.ru/books/_principles_of_model_checking.pdf

Book Maude



- pdf available:

<https://maude.cs.illinois.edu/w/images/0/0d/Maude-book.pdf>

Organization

Lectures (Emilio 3 / Clemens 4)

- ▶ presentations on blackboard
- ▶ notes after the lecture (notes 2025/26 available)
- ▶ January 19 – January 28 (7 lectures)

Organization

Lectures (Emilio 3 / Clemens 4)

- ▶ presentations on blackboard
- ▶ notes after the lecture (notes 2025/26 available)
- ▶ January 19 – January 28 (7 lectures)

Webpage

- ▶ <https://clegra.github.io/mc/mc.html>

Organization

Lectures (Emilio 3 / Clemens 4)

- ▶ presentations on blackboard
- ▶ notes after the lecture (notes 2025/26 available)
- ▶ January 19 – January 28 (7 lectures)

Webpage

- ▶ <https://clegra.github.io/mc/mc.html>

Exam

- ▶ options:
 - ▶ small verification project (of an algorithm, e.g. in [MaudE3](#))
 - ▶ presentation about a paper
 - ▶ written exam?

Organization

Lectures (Emilio 3 / Clemens 4)

- ▶ presentations on blackboard
- ▶ notes after the lecture (notes 2025/26 available)
- ▶ January 19 – January 28 (7 lectures)

Webpage

- ▶ <https://clegra.github.io/mc/mc.html>

Exam

- ▶ options:
 - ▶ small verification project (of an algorithm, e.g. in [MaudE3](#))
 - ▶ presentation about a paper
 - ▶ written exam?

Thank you – we are looking forward to the course!