

Introduction to Model Checking

(Preview of Core Module)

Clemens Grabmayer

<https://clegra.github.io>

Emilio Tuosto

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

Department of Computer Science



December 2, 2024

Model Checking

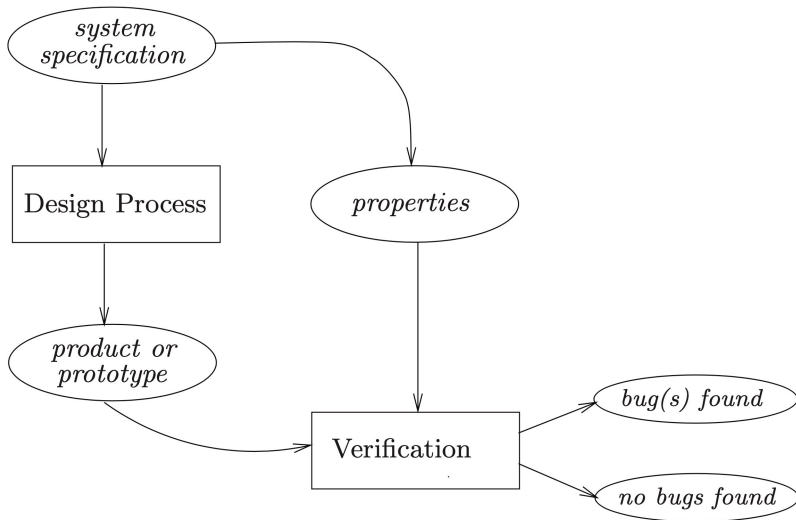
... is an effective automatable technique:

- ▶ *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.*

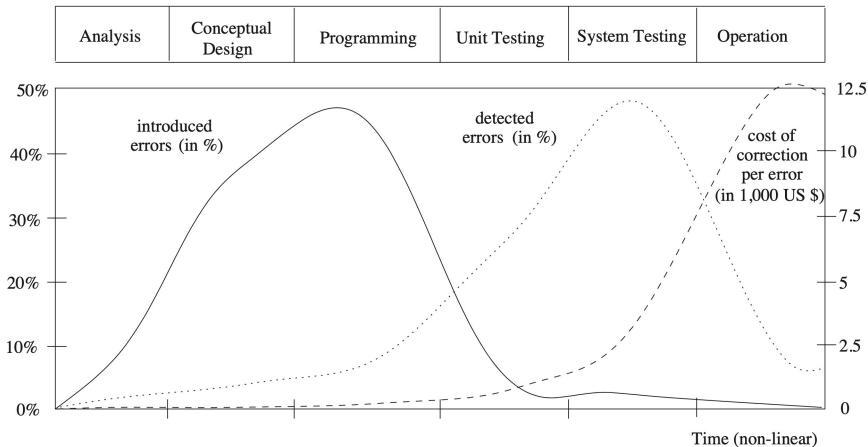
Strengths:

- ▶ 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)

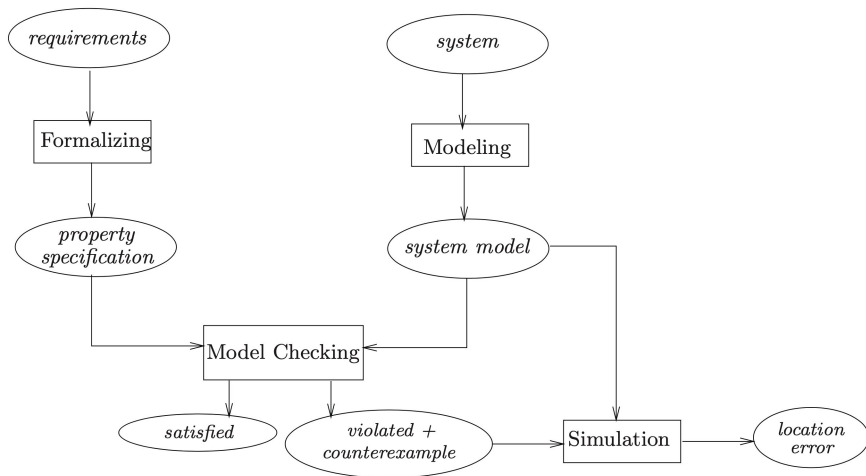
Hard-/Software Verification (traditionally)



Error introduction, detection, and repair costs



Model checking



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

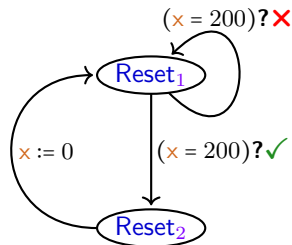
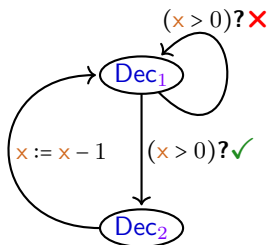
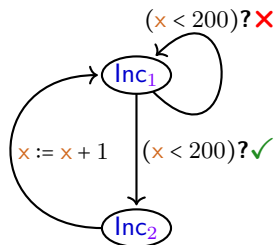
while true

do

```

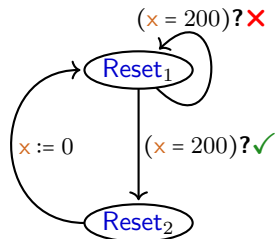
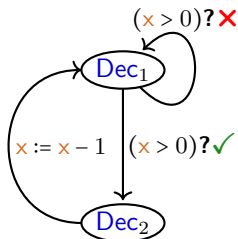
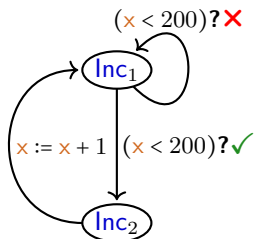
1:   if x = 200
2:   then x := 0
      fi
      od

```



Labeled transition systems (LTSs)

Formalizing properties (in temporal logic)



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

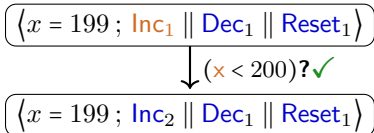
Counterexample (offending execution trace)

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

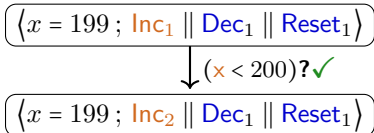
Counterexample (offending execution trace)

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

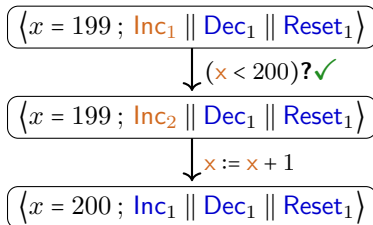
Counterexample (offending execution trace)



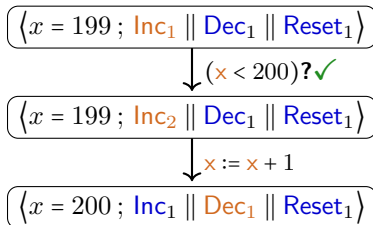
Counterexample (offending execution trace)



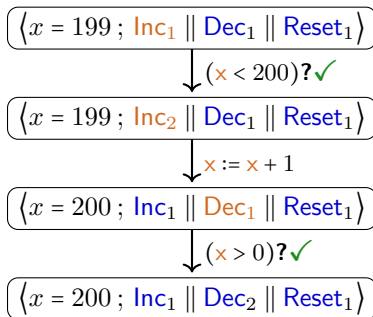
Counterexample (offending execution trace)



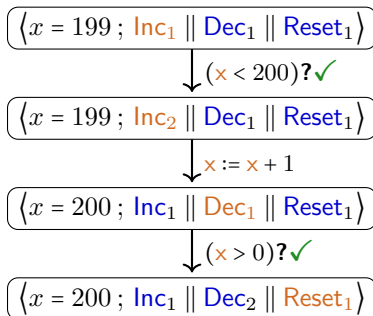
Counterexample (offending execution trace)



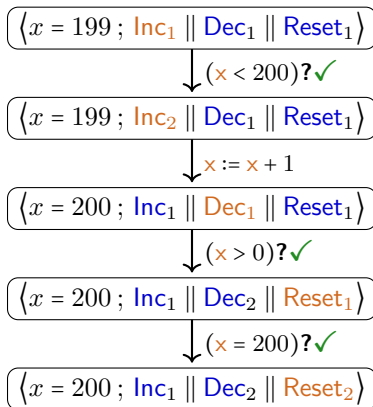
Counterexample (offending execution trace)



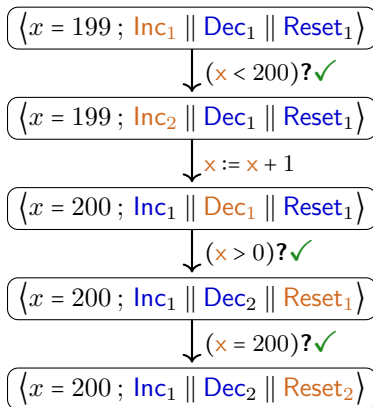
Counterexample (offending execution trace)



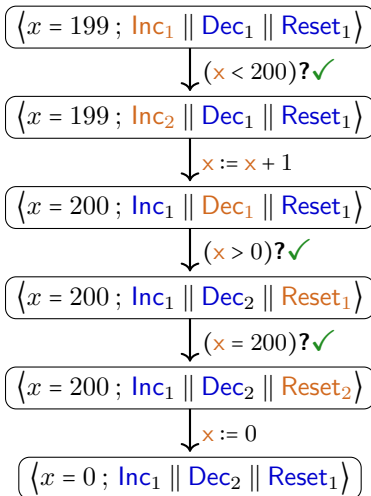
Counterexample (offending execution trace)



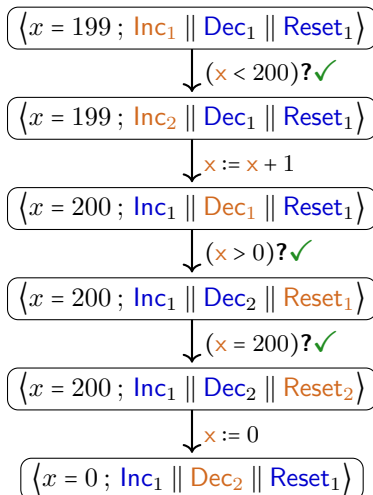
Counterexample (offending execution trace)



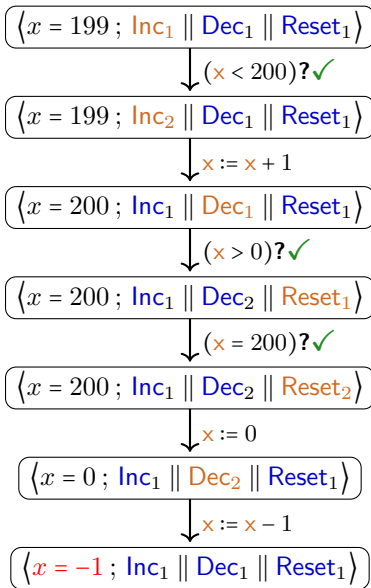
Counterexample (offending execution trace)



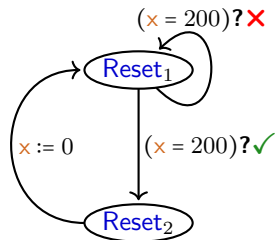
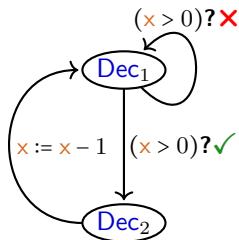
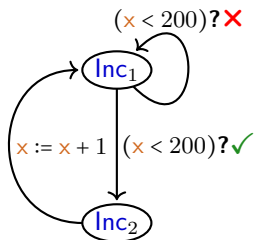
Counterexample (offending execution trace)



Counterexample (offending execution trace)

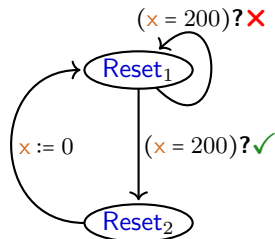
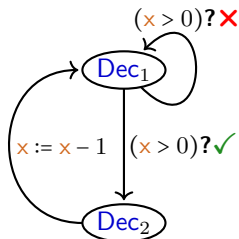
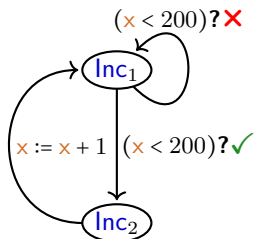


Formalizing properties (in temporal logic)



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

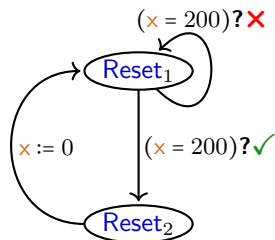
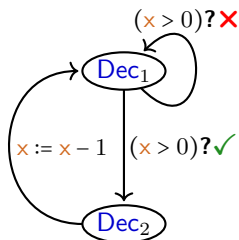
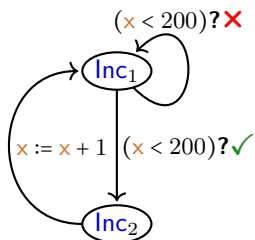
Formalizing properties (in temporal logic)



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

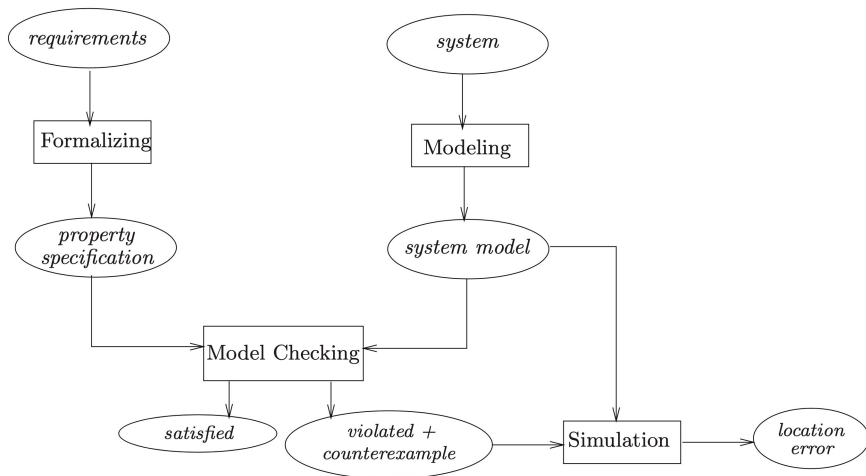
$Inc_1 \parallel Dec_1 \parallel Reset_1 \models \Diamond(x < 0)$ (LTL formula)

Formalizing properties (in temporal logic)



- Inc₁ || Dec₁ || Reset₁ $\neq \square(0 \leq x \wedge x \leq 200)$ (Linear-TL formula)
- Inc₁ || Dec₁ || Reset₁ $= \diamond(x < 0)$ (LTL formula)
- Inc₁ || Dec₁ || Reset₁ $\neq \forall \square(0 \leq x \wedge x \leq 200)$ (Computation-Tree-L formula)
- Inc₁ || Dec₁ || Reset₁ $= \exists \square(0 \leq x \wedge x \leq 200)$ (CTL formula)
- Inc₁ || Dec₁ || Reset₁ $= \forall \square \exists \diamond(x < 0)$ (CTL formula)

Model checking

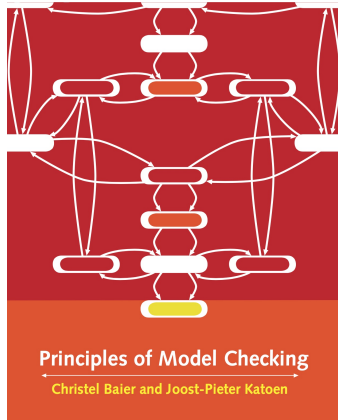


Any [such] verification is only as good as the model of the system.

Topics of the module

- ▶ 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 **mCRL2 model-checker toolbox**

Book



- ▶ pdf available:

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

Organization

Lectures (Clemens 5/Emilio 2)

- ▶ presentations on blackboard
- ▶ notes after the lecture (notes 2024/25 available)
- ▶ February (first/second week)

Organization

Lectures (Clemens 5/Emilio 2)

- ▶ presentations on blackboard
- ▶ notes after the lecture (notes 2024/25 available)
- ▶ February (first/second week)

Exam

- ▶ options:
 - ▶ small verification project (of an algorithm, e.g. in [mCRL2](#))
 - ▶ presentation about a paper
 - ▶ written exam?

Organization

Lectures (Clemens 5/Emilio 2)

- ▶ presentations on blackboard
- ▶ notes after the lecture (notes 2024/25 available)
- ▶ February (first/second week)

Exam

- ▶ options:
 - ▶ small verification project (of an algorithm, e.g. in [mCRL2](#))
 - ▶ presentation about a paper
 - ▶ written exam?

Thank you – we are looking forward to the course!