Model Checking and Games

Rüdiger Ehlers, Clausthal University of Technology

September 2019

Motivation & Introduction

Safety of systems

Driving question of this course

How can we ensure that a *reactive* system does what it is supposed to be doing?

Safety of systems

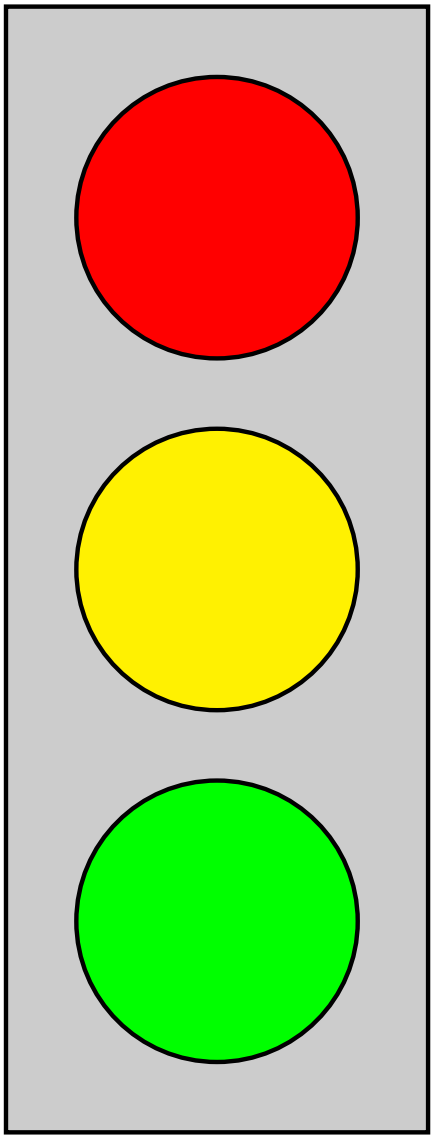
Driving question of this course

How can we ensure that a *reactive* system does what it is supposed to be doing?

Secondary driving question of this course

...And how can we automate what we do to achieve this to the greatest extent possible?

Example 1: Traffic light



Example property (1)

Never ever at an intersection, a car coming from the north and a car coming from the east both see a green light.

Example property (2)

Every car eventually sees a green light.

Example 2: Ariane 5 rocket, June 1996

Dowson, 1997

"On 4 June 1996, the Ariane 501 satellite launch failed catastrophically 40 seconds after initiation of the flight sequence, incurring a direct cost of approximately $ 370 million. The Inquiry Board Report (IBR) clearly identifies the proximate cause of the disaster as a software failure; but in other respects it is one of the more astonishing engineering documents of our time."

Example 3: Intel Pentium FDIV bug

Description

- Incorrect implementation of the `FDIV` instruction (missing rows in a look-up table for a standard division algorithm)
- As a result, the result of a division was off in certain cases

Cost

Loss of 500 million USD for Intel (plus reputation loss)

Ensuring the correctness of systems

Basic approaches

- **Build it right**
- **Built it, then verify that it's right.**

Building it right (covered in this course)

Idea: By using a chain of provably correct steps, obtain an implementation from a specification

Built it, then verify that it's right (also covered)

Idea: Starting from a model of the system to build, use a series of provably correct steps to show that the system has the correct properties.

Most important ingredient

Specifications!

Without good specifications, all activities for ensuring the correctness of systems are moot.

Central activities

- Verification: Are we building the system right?
- Validation: Are we building the right system?

Model checking

Basic idea

Verify the correct behavior of a system based on a *model* of its structure and/or behavior in an automated way.

Central questions

- How do we represent systems?
- How do we represent specifications?
- How do we test that they fit together?

$$\phi \models \psi$$

Model checking vs. testing

Advantages of testing

- It utilizes the system to test directly, no need to build a model
- Test cases are natural to write for programmers

Advantages of model checking

- Guarantees to find *all* bugs in the model

Model checking vs. deductive verification

Advantages of deductive verification

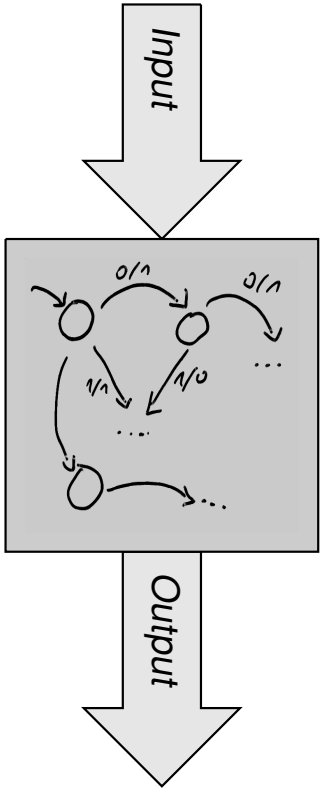
- Can prove correctness of highly complex systems

Advantages of model checking

- Automates the correctness checking process of the implementation.

Beyond model checking: Synthesis

Verification:



$+ \quad \mathbf{G}(r \rightarrow \mathbf{X}g) \quad \Rightarrow$

Satisfied

Not satisfied
(+ counter-example)

Beyond model checking: Synthesis

Synthesis:

$\mathbf{G}(r \rightarrow \mathbf{X}g)$

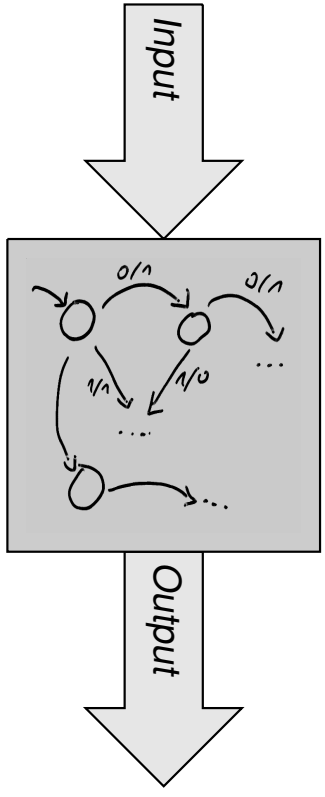
$+$

Input $= \{r, \ldots\}$
Output $= \{g, \ldots\}$

$\Longrightarrow$

Realizable $\longrightarrow$

Not realizable



Summary

In this course, we will deal with methods to assure the *safe* behavior of reactive systems, i.e., those that continuously interact with their environment.

We will focus on methods that lead to high *automation* levels while doing so.

We will look at both the theory and practice of this area.

Structure of this course

Basics

1. Modelling reactive systems
2. Modelling specifications
3. Verifying specifications against models

Tools

Spin

Symbolic verification

1. Binary decision diagrams / Algorithms for BDDs
2. Satisfiability solving & Bounded Model checking

Tools

(dd/CUDD)
Alloy

Reactive synthesis

1. Introduction to reactive synthesis

Tools

Slugs

Organisation

Block course

Dates / Times

- We will meet from 10:00-11:30 and 11:50 to 13:40 each day (Monday to Friday) for the next two weeks.
- Mixture of lectures & tutorials

Exercise sheets

- There will be one each day (8 in total).
- Please solve until the *next* lecture.
- Please return them by **e-mail**.
- The discussion of your solutions will happen on the day after you hand in your exercise sheet.

Resources

Link to all slides, lecture notes, and exercise sheets

`https://modelchecking.ruediger-ehlers.de`

(Linked to from the Stud.IP page)

Material overview

- Slides
- Whiteboard figures/text documents
- Exercise sheets

Note that the slides and the whiteboard part documents *interleave*.

References I

Mark Dowson. The ariane 5 software failure. *ACM SIGSOFT Software Engineering Notes*, 22(2):84–86, 1997.