1. System Verification: Introduction

José Proença

System Verification (CC4084) 2025/2026

CISTER – U.Porto, Porto, Portugal

https://fm-dcc.github.io/sv2526

What are Formal Methods?

What are formal methods?

Formal methods are techniques to model complex systems using rigorous mathematical models

Specification

Define part of the system using a modelling language

Verification

Prove properties.
Show correctness.
Find bugs.

Implementation

Generate correct code.

All formal models are wrong

All formal models are wrong

... but some of them are usefull!

Program verification vs. System verification

Program verification

- software (code)
- $+$ annotations (logic)
- $+$ some user interaction
- $=$ correctness proof

Program verification vs. System verification

Program verification

- software (code)
- $+$ annotations (logic)
- $+$ some user interaction
- $=$ correctness proof

SYSTEM verification

- system specification (model)
- $+$ system requirements (logic)
- $+$ some user interaction
- $+$ fixing parameters/scenarios
- $=$ correctness proof

In this course: we will focus on model-checking

Contents of the module

Syllabus

- Introduction to model-checking
- CCS: a simple language for concurrency
  - Syntax
  - Semantics
  - Equivalence
  - mCRL2: modelling
- Dynamic logic
  - Syntax
  - Semantics
  - Relation with equivalence
  - mCRL2: verification
- Timed Automata
  - Syntax
  - Semantics (composition, Zeno)
  - Equivalence
  - UPPAAL: modelling
- Temporal logics (LTL/CTL)
  - Syntax
  - Semantics
  - UPPAAL: verification
- Probabilistic and stochastic systems
  - Going probabilistic
  - UPPAAL: monte-carlo

Logistics

Useful information

Relevant class material and announcements will be posted on the website periodically

`https://fm-dcc.github.io/sv2526`

E-mail

- jose.proenca@fc.up.pt

Office hours (please send an email the day before if you wish to meet):

- *José Proença:* Thursday morning

Assessment

Assessment will consist of

- 70% – an individual test at the end (*época normal*);
- 30% – a group assignment with 2 parts involving the use of the mCRL2 and the Uppaal model checkers; and
- 100% – Final (optional) exam during the extra period (*época de recurso*).

What is model-checking?

What is model-checking?

Check Requirements of a Model

using **Formal Methods**

Example: coffee machine

$$\mathcal{M}, s \models \phi$$

does the model

$\mathcal{M}$

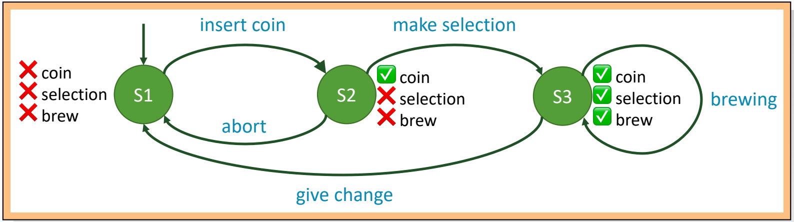
in state

s

satisfies the **requirement**

ϕ

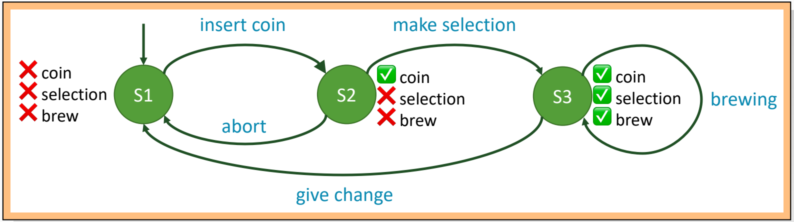
Example: coffee machine - the MODEL



Actions **States** **Propositions**

Just building the model is often a large contribution

Example: coffee machine - the REQUIREMENTS



$\mathcal{M}, S2 \models coin$

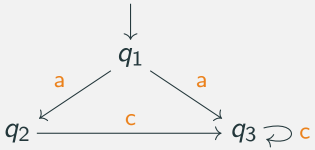
means coin holds in state *S2*

$\mathcal{M}, S1 \models [make\ selection]\ selection$

means selection holds in every state reachable with *"make selection"* from *S1*

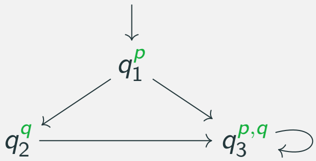
Actions vs. States

Focus on events



- desired/forbidden sequences of actions
- Process algebra to generate models
- $\mathcal{M}, q_2 \models [a]\mathit{false}$

Focus on states



- reachable/forbidden states
- Language/Diagram to generate models
- $\mathcal{M}, q_1 \models p$, $\mathcal{M}, q_1 \models F\ G\ p$

Note 1/2: Models and logics are tightly connected

$$\mathcal{M}, q_2 \models [a]\mathit{false}$$

- Models that safisty exactly the same requirements:
  equivalence (e.g. bisimulation, trace equivalence)
- Models that satisfy a subset of requirements:
  inclusion (e.g. simulation, trace inclusion)
- A model should only capture the necessary to show its requirements.

Note 2/2: Chose your Model and Logic wisely

$$\mathcal{M}, q_2 \models [a]\mathit{false}$$

- **Real-time:** how long it takes between actions
- **Differential dynamic:** state evolves using differential equations
- **Beliefs:** who knows what
- **Deontic:** obrigatory and permitted actions
- **Fuzzy:** other values instead of truth values
- **Probabilistic:** the odds of something occuring
- *Many tools:* mCRL2, UPPAAL, Spin, NuSMV (NuXMV), TLA+, Maude, Storm, CPN (petri nets)