NSWI101: System Behaviour Models and Verification

1. Modelling basics

Jan Kofroň

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

Department of
Distributed and
Dependable
Systems

Syllabus I.

- Mathematical structures for behaviour modelling:
  - Labelled transition systems, Kripke structures, Timed automata, Markov chains
- Specification of system properties
  - Temporal logics: LTL, CTL, TCTL, PCTL
- Basic verification tasks
  - Equivalence checking and model checking

Syllabus II.

- Decidability and complexity
  - … of equivalence checking and model checking with respect to model type
- Software tools for model checking
- Hard issues in formal verification
  - Infinite-state systems
  - State explosion problem
  - Strategies to fight it

GRADING

- Final grades will be determined by the quality of homework and the result of the final exam in the following ratio:
  - 55% Assignments (several home works)
  - 45% Final exam (max. 100 points)
    - >= 80 $\rightarrow$ 1
    - >= 71 $\rightarrow$ 2
    - >= 62 $\rightarrow$ 3
- Final exam—written test

Information

All information available at the course web page:

`https://d3s.mff.cuni.cz/teaching/nswi101/`

Part I: Introduction

NEED FOR FORMAL METHODS: STRIKING STORIES

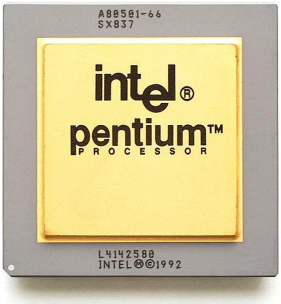
Ariane 5, 1996

- False angle of attack caused by incorrect altitude data following software exception
- The rocket self-destructed in 37 seconds after launch
- Software exception—overflow in conversion of 64-bit floating-point number to 16-bit signed integer value caused operand error
- The error occurred because of unexpectedly high value of sensed horizontal velocity
- The value was much higher than expected because early part of Ariane 5 trajectory differed from that of Ariane 4—higher velocity values
- Direct cost €500M, indirect cost €2,000M

NEED FOR FORMAL METHODS: STRIKING STORIES

Intel: Pentium FDIV bug, 1994

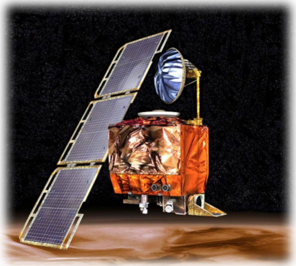
- "Imprecision" of FDIV operation firstly not admitted
- CPUs called off after publishing proof
- Cost $500M



Need for formal methods: Striking stories

NASA Mars Climate Orbiter

- September 30, 1999
- Peer review preliminary findings indicate that one team used English units while the other used metric units for key spacecraft operations



NASA Mars Polar Lander

- December, 1999
- The leading theory is that surface contact detector located on landing struts mistakenly interpreted the force of landing struts deployment as contact with the surface, causing landing rockets to shut down prematurely and probe to impact at a too-high velocity

Nissan, 2015

- Sensor failure caused not detecting human in the seat
- Airbag malfunction (failing in car crash) appeared
- In 2015, Nissan recalled 3.5 millions of cars to fix this
- Airbag problems reported by other car manufacturers in 2016: General Motors—GMC, Chevrolet, Buick, Cadillac

Boeing 737 MAX, 2018

- New device introduced into Boeing 737 MAX (MCAS) to compensate too steep take-off
- MCAS relies on single sensor
- MCAS can reset itself after a pilot intervention
- Info on MCAS was *not* put into manuals!
- Two air crafts with passengers crashed (2018 and 2019)
- The air crafts were grounded

ACHIEVING SYSTEM RELIABILITY

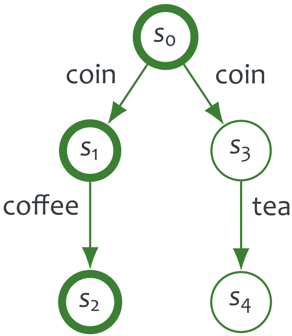
- Experimental methods
  - Testing—applied to the system itself
  - Simulation—experimenting with a model of a system

ACHIEVING SYSTEM RELIABILITY

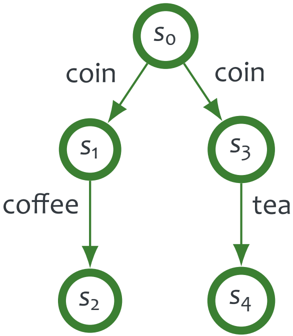
- Experimental methods
  - Testing—applied to the system itself
  - Simulation—experimenting with a model of a system
- Formal methods
  - Deductive verification—theorem proving
  - **Equivalence checking**—comparing two specifications (models)
  - **Model checking**—checking a particular property of a model (even code)

Program execution vs. Verification

Program execution



Program verification

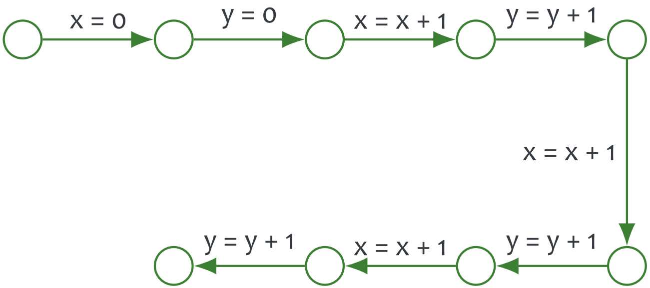


Part II: Labelled Transition Systems

Labelled Transition System—LTS

```
x:=0;
y:=0;

for i:=1 to 3 {
  x:=x+1;
  y:=y+1;
}
```



LTS DEFINITION

Labelled Transition System is a triple $(S, Act \rightarrow)$:

- S is set of states (domain)
- Act is set of labels (actions)
- $\rightarrow$ is transition relation: $\rightarrow \subseteq S \times Act \times S$

LTS vs. Finite automaton

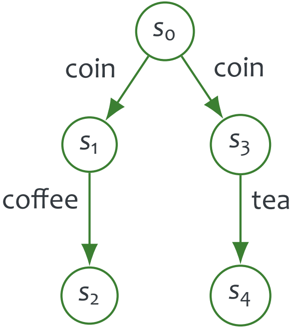
LTS:

- $(S, Act, \rightarrow)$
- *Trace*—sequence of labels following one path in LTS
- LTS corresponds to set of traces reachable in the LTS

Finite Automaton:

- $(S, Act, \rightarrow, I, A)$
- Additionally sets of initial and accepting states
- Notion of *word* and *language* accepted by automaton

LTS—Example



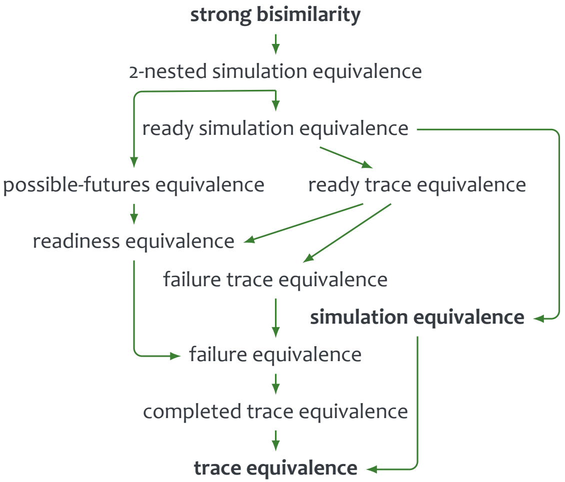
$$traces(s) = \{\sigma \in Act^* | s \stackrel{\sigma}{\Longrightarrow}\}$$
$$traces(s_0) = \{\epsilon, \text{coin}, \text{coin.coffee}, \text{coin.tea}\}$$

Note that due to absence of initial and accepting states, trace can terminate at any state.

Comparing LTSs

Many relations to compare LTSs between each other:

- Trace preorder/equivalence
- Simulation preorder/equivalence
- Bisimilarity
- Readiness equivalence
- Failure equivalence
- …



TRACE PREORDER AND EQUIVALENCE

States s and t are in **trace preorder relation** ($s \leq_t t$) iff

$$traces(s) \subseteq traces(t)$$

States s and t are **trace equivalent** ($s =_t t$) iff

$$traces(s) \subseteq traces(t) \wedge traces(t) \subseteq traces(s)$$

This corresponds to equivalence of languages in the automata world.

SIMULATION PREORDER AND EQUIVALENCE

Relation $R \subseteq S \times S$ is **simulation** iff $(s,t) \in R \implies \forall s'.s \xrightarrow{a} s' \, \exists t'.t \xrightarrow{a} t' \wedge (s',t') \in R$

States s and t are in **simulation preorder** ($s \leq_s t$) iff
there exists simulation R and $(s,t) \in R$.

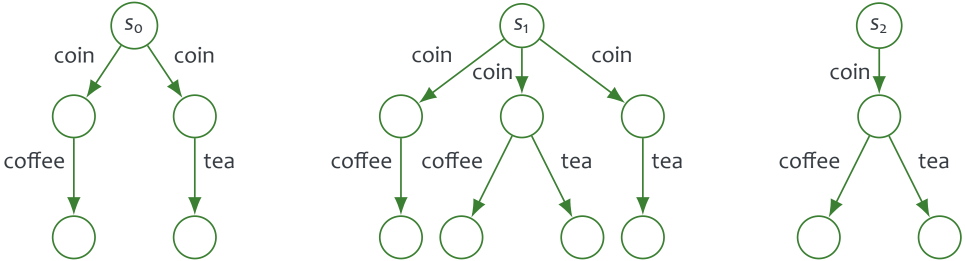
States s and t are **equivalent under simulation** ($s =_s t$) iff
$\exists R, Q.(s,t) \in R \wedge (t,s) \in Q$, and
R and Q are simulations.

R and Q can be the same relation or not.

LTS relations—Examples

Trace equivalence:

$$traces(s_0) = traces(s_1) = traces(s_2) = \{\epsilon, coin, coin.coffee, coin.tea\}$$
$$s_0 =_t s_1 =_t s_2$$

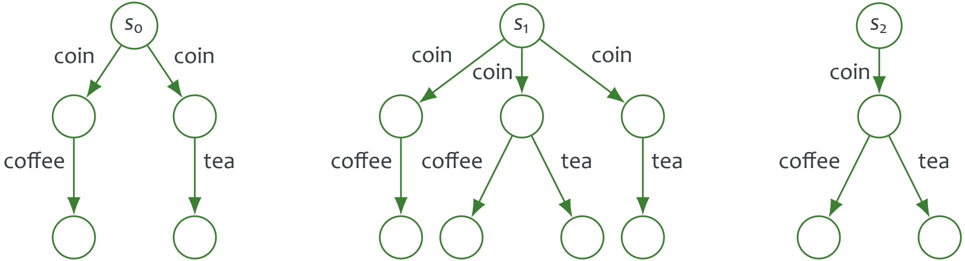


LTS relations—Examples

Simulation preorder and equivalence:

$$s_0 \leq_s s_1 \wedge \neg(s_1 \leq_s s_0) \implies s_0 \neq_s s_1$$

$$s_1 \leq_s s_2 \wedge s_2 \leq_s s_1 \implies s_1 =_s s_2$$



(STRONG) BISIMILARITY

Relation R is **bisimulation** iff $\forall s, t.(s, t) \in R \implies$

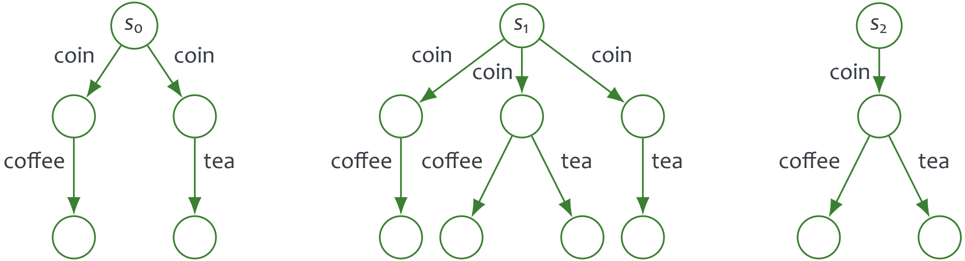
$$\forall s'.s \xrightarrow{a} s' \; \exists t'.t \xrightarrow{a} t' \wedge (s', t') \in R \wedge$$
$$\forall t'.t \xrightarrow{a} t' \; \exists s'.s \xrightarrow{a} s' \wedge (s', t') \in R$$

States s and t are **bisimilar** (equivalent under bisimulation) $(s \sim t)$ iff

$(s, t) \in R$ and R is bisimulation.

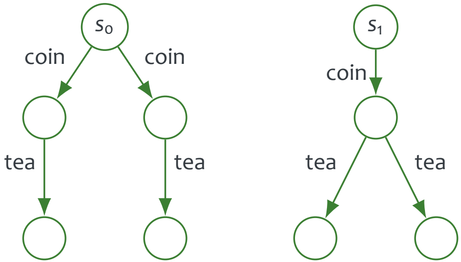
Bisimilarity—Example

$$s_0 \not\approx s_1$$
$$s_1 \not\approx s_2$$
$$s_0 \not\approx s_2$$



Bisimilarity—Example

$$s_0 \sim s_1$$



Deterministic vs. non-deterministic LTSs

For deterministic LTS, all the relations are equivalent: $a =_t b \leftrightarrow a =_s b \leftrightarrow a \sim b$.

Non-deterministic LTS cannot be transformed into equivalent deterministic LTS as in automata world.

- It can, of course, but its semantics changes!

Process Algebras (PA)

- Textual way for capturing LTS
- Various process algebras exist: CCS, CSP, ACP, π-calculus, μ-calculus, ...
- Equational reasoning—transformations of expressions usually to simplify them or to proof certain property
- Modelling in many areas: concurrent systems, communication protocols, electronic circuits, biochemical processes, ...

Algebra of Communicating Processes (ACP)

- Simple process algebra by Jan Bergstra and Jan Willem Klop (1982)
- Just few syntactical constructs:
  - Choice (+)
  - Sequencing (.)
  - Concurrency (||)
  - Process communication (γ)
  - Abstraction (τ)
- Example of processes:
  - $p : (gen_1 + gen_2).send$
  - $q : recv.proc$
  - Defining communication: $\gamma(send, recv) = trans$
  - Composition of processes: $p||q = (gen_1 + gen_2).trans.proc$
  - Hiding internal computation (abstraction): $\tau_{\{gen_1, gen_2, proc\}}(p||q) = \tau.trans.\tau$

ACP semantics

For process variables x, y

- $x + y = y + x$
- $(x + y) + z = x + (y + z)$
- $x + x = x$
- $(x + y).z = x.z + y.z$
- $(x.y).z = x.(y.z)$
- $x + \delta = x$
- $\delta.x = \delta$

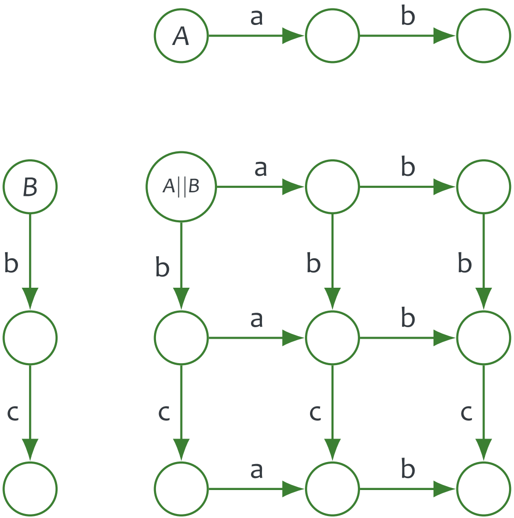
Note that $z.(x + y) = z.x + z.y$ is not included (non-deterministic choice)!

Parallel composition in ACP

Let $A = a.b.\epsilon$ and $B = b.c.\epsilon$

Parallel composition just “syntax sugar”:

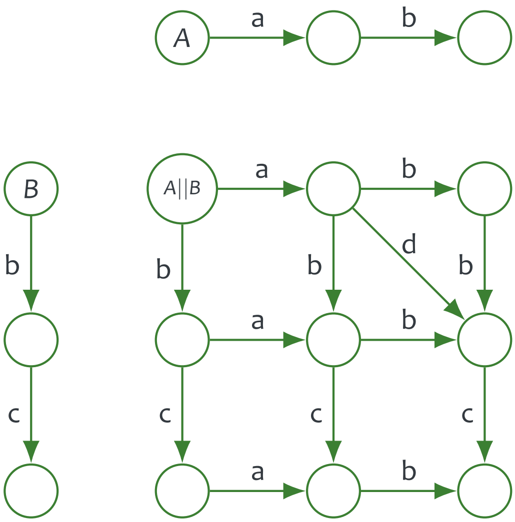
$$A||B = a.(b.b.c.\epsilon + b.(b.c.\epsilon + c.b.\epsilon)) + \\ b.(a.(b.c.\epsilon + c.b.\epsilon) + c.a.b.\epsilon)$$

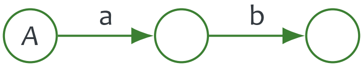


Parallel composition in ACP with communication

Let $A = a.b.\epsilon$ and $B = b.c.\epsilon$
Let $\gamma(b, b) = d$

Processes can perform the actions synchronously.

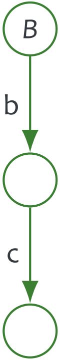


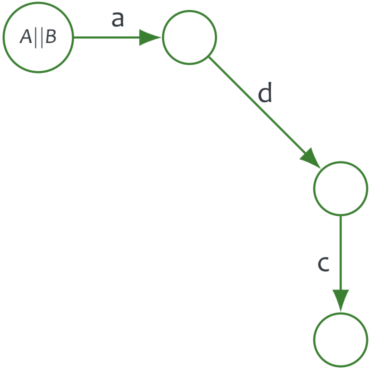


Let $A = a.b.\epsilon$ and $B = b.c.\epsilon$
Let $\gamma(b, b) = d$

Disabling (encapsulation) operator:
$\delta_{\{b\}}(A||B) = a.d.c.\epsilon$

Processes *must* perform actions synchronously.





Building concurrent systems

1. Specify particular components
2. Specify communication actions
3. Construct parallel compositions
4. Disable certain actions to enforce communication

Model-to-code correspondence

- LTS relations useful for verifying design of communication protocols, cryptography protocols, and algorithms in general
- They are also applicable for checking correspondence between code (implementation) and LTS (specification)
  - Inherently hard (undecidable) problem—models made of finite number of states while code usually induces infinite state space
  - Manual maintenance of model-to-code correspondence difficult—scalability issues
  - Preorder relation usually applied—specification to be implemented (can implement more)

Part III: Thesis topics

Thesis topics

- If you are interested in model checking (or verification in general), contact me for supervising bachelor or master thesis, software project, research project, or just contributing to one of our verification projects!
- For a (non-exhaustive) topic list, please visit

  `https://d3s.mff.cuni.cz/students/topics/`

- Surely we can devise a topic suiting your expectations!