

NSWI101: System Behaviour Models and Verification

Lab 01

Jan Kofroň; Klára Pernicová



FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

Department of
Distributed and
Dependable
Systems

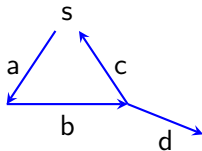


Traces

States p and q are in **trace preorder relation** ($p \leq_t q$) iff
 $traces(p) \subseteq traces(q)$

States p and q are **trace equivalent** ($p =_t q$) iff
 $traces(p) \subseteq traces(q) \wedge traces(q) \subseteq traces(p)$

1. Determine $traces(s)$
2. Draw two different LTSs where two states are trace equivalent
3. Draw two different LTSs where two states are in trace preorder relation but are not trace equivalent



Simulation

Relation $R \subseteq S \times S$ is **simulation** iff

$$(p, q) \in R \implies \forall p'. p \xrightarrow{a} p' \exists q'. q \xrightarrow{a} q' \wedge (p', q') \in R$$

States p and q are in **simulation preorder** ($p \leq_s q$) iff

there exists simulation R and $(p, q) \in R$.

States p and q are **equivalent under simulation** ($p =_s q$) iff

$\exists R, Q. (p, q) \in R \wedge (q, p) \in Q$, and
 R and Q are simulations.

1. Draw two different LTSs where two states are in simulation preorder
2. Draw two different LTSs where two states are equivalent under simulation but the two simulations are different relations
3. What property does it imply if one LTS is a subgraph of another LTS (starting from the same state)?

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.

1. Draw two different LTSs where two states are bisimilar
2. Draw two different LTSs where two states are trace equivalent but are not bisimilar
3. If two states are bisimilar, are they also trace equivalent?

1. Prove that trace preorder is transitive
2. Prove that simulation preorder is transitive
3. Prove that for deterministic LTSs, trace equivalence, simulation equivalence and bisimilarity are equivalent relations

- 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): $\tau_{\{gen_1, gen_2, proc\}}(p||q) = \tau.trans.\tau$

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)!

Mutex indicator in ACP

- Two lights operate continuously, each toggling between on and off.
- They must never be turned on at the same time.
- The lights communicate directly with each other to take turns.

Sender and receiver in ACP

- A sender continuously transmits messages to a receiver.
- They communicate directly.
- The sender cannot send the next message until the receiver has acknowledged the current one.

- A sender continuously generates and sends messages to a receiver.
- A thief can steal transmitted messages.
- If stolen, the receiver gets nothing, the sender times out, and the sender resends.
- If delivered, the receiver acknowledges, and the sender generates and sends a new message.
- Suppose for now that the thief cannot steal the acknowledgments.
- Let the timeout be a nondeterministic choice with receiving ack.

- Suppose the former situation, only the thief can also steal the acks.
- If the sender times out, it resends the old message.
- The receiver needs to know whether a new message is the old one or a new identical message.
- Both messages and acks are tagged with an alternating 1-bit tag.
- When receiver gets a message with the expected bit, its a new message and he delivers it.
- When receiver gets a message with a wrong bit, it's the old message, and receiver resends the old ack.