1.4 Structural Operational Semantics

In Examples 1.3.2 and 1.3.3 we have demonstrated the general idea on how to derive LTS from processes. This idea will now be formalised. Transitions are derived inductively over the syntactic structure of processes. In fact, rather than defining an LTS for each process separately, we just define *one big* LTS with the (infinite) state set $\mathbb{P}$. However, we always will only consider processes of which set of reachable states is finite (*cf.* Definition 1.2.7 and the comment following it).

1.4.1 Definition ($L_{\mathbb{P}}$) We define the transition system $L_{\mathbb{P}} = (S, Act \cup \{\tau\}, \rightarrow)$, where:

- state space $S = \mathbb{P}$, the processes;
- the actions set Act the same as used to define $\mathbb{P}$;
- $\rightarrow$ contains all and only those transitions that can be derived with the following rules:
  1. for all $a \in Act_\tau$, $p \in \mathbb{P}$: $a.p \xrightarrow{a} p$
  2. for $p, q \in \mathbb{P}$: if $p \xrightarrow{a} p'$ for $a \in Act_\tau$, then
     $$p + q \xrightarrow{a} p'$$
  3. Similarly: if $q \xrightarrow{a} q'$ for $a \in Act_\tau$, then
     $$p + q \xrightarrow{a} q'$$
  4. For $P \in \mathcal{P}$: if $P \hat{=} p$ and $p \xrightarrow{a} p'$, then
     $$P \xrightarrow{a} p'$$
  5. For $A \subseteq Act$ and $p\|_A q \in \mathbb{P}$: if $p \xrightarrow{a} p'$ and $a \notin A$, then
     $$p\|_A q \xrightarrow{a} p'\|_A q$$
  6. for q analogously
  7. if $p \xrightarrow{a} p'$, $q \xrightarrow{a} q'$, and $a \in A$, then
     $$p\|_A q \xrightarrow{a} p'\|_A q'$$

A more compact way to write the rules: SOS-Rules (slide #5)

$$1)\ \frac{}{a.p \xrightarrow{a} p} \qquad 2)\ \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'} \qquad 3)\ \frac{q \xrightarrow{a} q'}{p + q \xrightarrow{a} q'}$$

$$4)\ \frac{p \xrightarrow{a} p'}{P \xrightarrow{a} p'}(P \hat{=} p)$$

$$5)\ \frac{p \xrightarrow{a} p'}{p\|_A q \xrightarrow{a} p'\|_A q}(a \notin A) \quad 6)\ \frac{q \xrightarrow{a} q'}{p\|_A q \xrightarrow{a} p\|_A q'}(a \notin A)$$

$$7)\ \frac{p \xrightarrow{a} p' \ q \xrightarrow{a} q'}{p\|_A q \xrightarrow{a} p'\|_A q'}(a \in A)$$

Note: All material on the slides used in the lecture is in the script.

1.4.2 Example (Semantics of $\mathbb{P}$)
Three process equations:

$$\begin{aligned} X &\hat{=} a.b.X \\ Y &\hat{=} a.c.Y + a.a.Y \\ Z &\hat{=} X\|_{\{a\}}Y \end{aligned}$$

We begin simple: the transition following X:

$$4)\ \frac{a.b.X \xrightarrow{a} b.X}{X \xrightarrow{a} b.X}$$

More complicated: *one* transition from Y (we forget rule 4), when possible):

$$2)\frac{a.c.Y \xrightarrow{a} c.Y}{a.c.Y + a.a.Y \xrightarrow{a} c.Y}$$

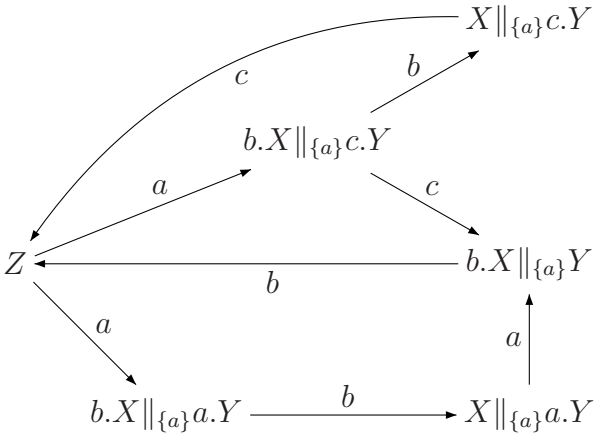
Another from Y:

$$3)\frac{a.a.Y \xrightarrow{a} a.Y}{a.c.Y + a.a.Y \xrightarrow{a} a.Y}$$

The big one: one transition from Z:

$$7)\frac{4)\dfrac{a.b.X \xrightarrow{a} b.X}{X \xrightarrow{a} b.X} \qquad 4)\dfrac{2)\dfrac{a.c.Y \xrightarrow{a} c.Y}{a.c.Y + a.a.Y \xrightarrow{a} c.Y}}{Y \xrightarrow{a} c.Y}}{X\|_{\{a\}}Y \xrightarrow{a} b.X\|_{\{a\}}c.Y}$$

The complete transitions system of Z (slide #6):



■ END EXAMPLE

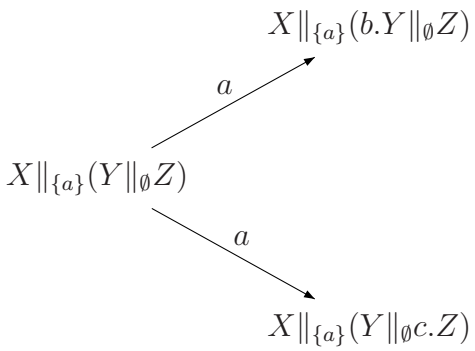
Note: All material on the slides used in the lecture is in the script.

1.4.3 Example (More fun with processes)

Parallel execution and Nondeterminism:

$$\begin{aligned} X &\hat{=} a.X \\ Y &\hat{=} a.b.Y \\ Z &\hat{=} a.c.Z \end{aligned}$$

Initial transitions $X\|_{\{a\}}(Y\|_{\emptyset}Z)$:



X, Y, Z are all deterministic. Parallelism causes apparently non-determinism.

Deadlock: $(a.\text{STOP}\|_{\{a,b\}}b.\text{STOP}) \not\rightarrow$ for all $c \in Act_\tau$.

■ END EXAMPLE

Note:

- The structure of $L_{\mathbb{IP}}$ depends actually on the given process definitions, which are part of the process specification.
- If there are no process definitions, then $L_{\mathbb{IP}}$ is still well defined (why?).

Note: All material on the slides used in the lecture is in the script.

Chapter 2

Differentiating Behaviour

The lecture is about a formal approach to testing, which, in our case, means that we are interested in establishing whether a given implementation is correct (or incorrect) with respect to a given specification. In the following we will introduce mathematical notions of correctness, the so-called *implementation relations*. In our setting here, implementation relations are relations $\leq \subseteq \mathbb{IP} \times \mathbb{IP}$. We say that, if $p \leq p'$, then p is an implementation of p'. There are a great many different definitions for implementation relations, each describing a different notion of correctness: some are more picky, some more lenient in comparing two processes. Quite frequently, implementation relations are *preorders*.

2.1 Preorders

2.1.1 Definition Let X be a set. $\leq \subseteq X \times X$ is called a *preorder*, iff

1. $(x, x) \in \leq$ (reflexivity)

2. $(x, y), (y, z) \in \leq \implies (x, z) \in \leq$ (transitivity)

We write $x \leq y$ for $(x, y) \in \leq$

2.1.2 Lemma Let $\leq$ be a preorder. Define $\sim_{\leq} \subseteq X \times X$ as $\{(x, y) |\ (x \leq y \text{ and } y \leq x\}$. $\sim_{\leq}$ is an equivalence relation, called the *kernel* of preorder $\leq$.

Proof: Reflexivity is inherited from $\leq$. The construction demands that $x \sim_{\leq} y \implies y \sim_{\leq} x$, thus symmetry holds as well. Transitivity: $x \sim_{\leq} y$ and $y \sim_{\leq} z$ imply $x \leq y$, $y \leq z$, and thus $x \leq z$. It also holds that $z \leq y$, $y \leq x$, thus also $z \leq x$, and $x \sim_{\leq} z$. ■ End Proof

- Preorders on $\mathbb{P}$ will play a role as so-called implementation relations: if $\leq$ is a preorder on processes, and $p \leq q$, then we say that p *is an implementation of* q.
- The kernels of preorders on $\mathbb{P}$ are then equivalences on processes: if $p \sim_{\leq} q$, then p, q behave *the same*, according of the chosen preorder.
- Usually it is easier to reason about preorders, rather than the respective kernels.

2.2 Some implementation relations

Until further notice we consider only processes that do not make any τ-steps!

Leaving out τ-steps has the following consequences:

1. $\Longrightarrow = \rightarrow \cup\{(p, \varepsilon, p) \mid p \in \mathbb{P}\}$
2. For consistency sake, we write then also $p \xrightarrow{\varepsilon} p'$ iff $p \stackrel{\varepsilon}{\Longrightarrow} p'$.

2.2.1 Definition (Trace preorder (trace inclusion)) Let p, q be processes. We define

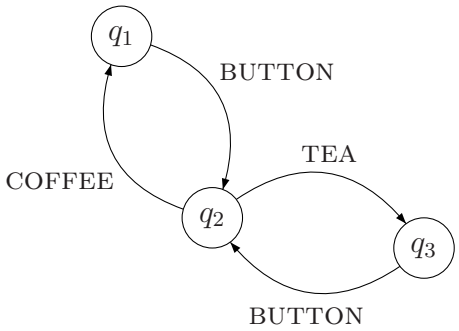
$$p \leq_{tr} q : \Longleftrightarrow \ traces(p) \subseteq traces(q)$$

- $\leq_{tr}$is a preorder: it follows from reflexivity and transitivity of $\subseteq$.
- if $p \leq q$ and p can execute $\sigma \in Act^*$, then q can as well.
- q describes the *legal* traces that an implementation is allowed to do.
- An implementation does not need to be complete.
- The kernel is *set equality* of the trace sets, called *trace equivalence*.

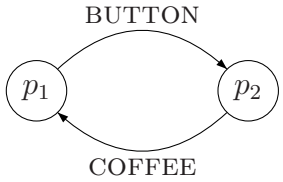
2.2.2 Example (Trace preorder)

1. State s_1 of the LTS in Example 1.2.9 (slide #4) and states q_1 and q_3 in the following LTS (slide #7) are trace equivalent (also state s_1 in Example 1.2.2, but that one contains τ-steps, which we agreed to neglect for the moment).

Note: All material on the slides used in the lecture is in the script.



2. Consider this LTS:



Apparently, p_1 is an implementation of q_1 above with respect to $\leq_{tr}$ (*i.e.*, $p_1 \leq_{tr} q_1$). Also, $p_2 \leq_{tr} q_2$. However, the opposite is not true: $q_1 \not\leq_{tr} p_1$, since *e.g.*, BUTTON · TEA $\in$ *traces*(q_1), but BUTTON · TEA $\notin$ *traces*(p_1).

■ END EXAMPLE

Another preorder, which is its own kernel.

2.2.3 Definition (Bisimulation equivalence) A bisimulation is a binary relation $R \subseteq \mathbb{P} \times \mathbb{P}$ on processes, satisfying for $a \in Act_\tau$:

- if pRq and $p \xrightarrow{a} p'$, then $\exists q' \in \mathbb{P} : q \xrightarrow{a} q'$ and $p'Rq'$.
- if pRq and $q \xrightarrow{a} q'$, then $\exists p' \in \mathbb{P} : p \xrightarrow{a} p'$ and $p'Rq'$.

We call p *bisimulation equivalent* to q ($\sim_B$) if there is a bisimulation R such that pRq.

The fact that we allow $a = \tau$ means that we consider τ an ordinary action. So even if we would consider LTS with internal steps, the definition would still be valid.

2.2.4 Example (Bisimulations (slide #8))
Consider the two LTS L_1 and L_2 in Figure 2.1 over action set $Act = \{a, b, c\}$. Let $S = \{p_1, p_2, p_3, q_1, \ldots, q_4\}$. The following relation $R \subseteq S \times S$ is a bisimulation:

$$p_1Rq_1 \quad p_2Rq_2 \quad p_3Rq_3 \quad p_1Rq_4$$

Note: All material on the slides used in the lecture is in the script.

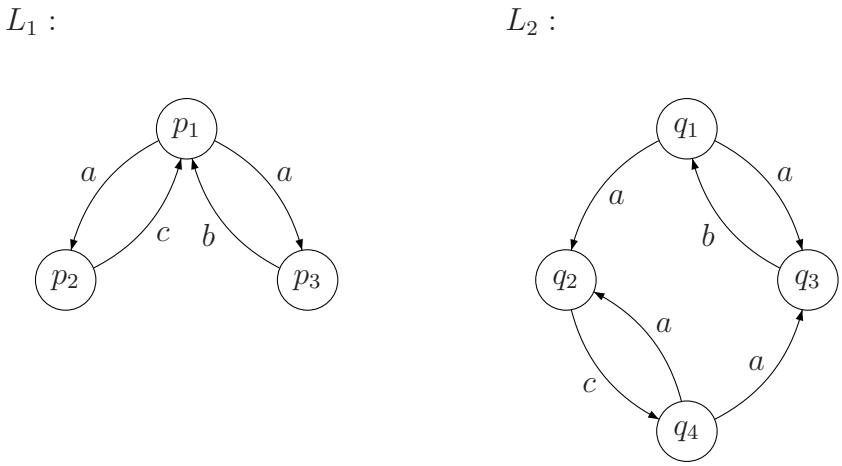


Figure 2.1: Example 2.2.4: Transition Systems

To prove this, we have to check that for all transitions in L_1, L_2 the conditions of Definition 2.2.3 are fulfilled. This is demonstrated in Figure 2.2. There, read $p_i \cdots\cdots q_j$ as $p_i R q_j$.

■ End Example

End of Lecture #2

Note: All material on the slides used in the lecture is in the script.