Recap of the previous class

LTS
- FSA
- Communicating machines
- Petri nets

A taste of denotational semantics
- RegExp

Term algebras

Transition System Specifications

"The first systematic study of TSSs may be found in [208], while the first study of TSSs with negative premises appeared in [57]." (Aceto et al.)
[208] R. d. Simone, Calculabilité et Expressivité dans l'Algèbre de Processus Parallèles Meije, thèse de 3 e cycle, Univ. Paris 7, 1984.
[57] B. Bloom, S. Istrail, and A. Meyer, Bisimulation can't be traced: preliminary report, in Conference Record 15th ACM Symposium on Principles of Programming Languages, San Diego, California, 1988, pp. 229–239. Preliminary version of Bisimulation can't be traced, J. Assoc. Comput. Mach., 42 (1995), pp. 232–268.

Fix a term algebra $\mathrm{Term}_{\Sigma,V}$ and a set of labels A

finite set of literals

A TSS with labels A is a set of (inference) rules $\dfrac{H}{\alpha}$

positive literals

LITERALS

$$\underbrace{t \xrightarrow{a} t' \quad \text{or} \quad t \in X}_{\text{positive}}$$

$$\underbrace{t \stackrel{a}{\not\rightarrow} \quad \text{or} \quad t \notin X}_{\text{negative}}$$

where $a \in A$, $t \in \mathrm{Term}_{\Sigma,V}$, $X \subseteq \mathrm{Term}_{\Sigma,V}$

Operational semantics of regular expressions

Operational semantics

$$(Act)\quad \frac{a \in A}{a \xrightarrow{a} 1}$$

$$(Cho_1)\quad \frac{x \xrightarrow{a} x' \quad x' \neq 1}{x + y \xrightarrow{a} x'} \qquad (Cho_2)\quad \frac{x \xrightarrow{a} 1}{x + y \xrightarrow{a} 1}$$

$$(Cho_3)\quad \frac{y \xrightarrow{a} y' \quad y' \neq 1}{x + y \xrightarrow{a} y'} \qquad (Cho_4)\quad \frac{y \xrightarrow{a} 1}{x + y \xrightarrow{a} 1}$$

$$(Seq_1)\quad \frac{x \xrightarrow{a} x' \quad x' \neq 1}{x \cdot y \xrightarrow{a} x' \cdot y} \qquad (Seq_1)\quad \frac{x \xrightarrow{a} 1}{x \cdot y \xrightarrow{a} y}$$

$$(Star_1)\quad \frac{}{x^* \xrightarrow{\varepsilon} 1} \qquad (Star_2)\quad \frac{x \xrightarrow{a} x'}{x^* \xrightarrow{a} x' \cdot x^*}$$

Note that
- x & y range over the set of reg exp
- these rules form a TSS
- there is a set of rules for each operator
- For 0, the set is empty!

Basic Process Algebras with $a \in A \cup \{\varepsilon\}$

Exercise 9
Simplify the TSS above (Hint: Think about the rules for choice)

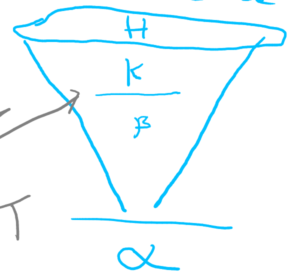
LTSs as proofs of TSSs

A proof in a TSS T of a closed transition rule H/α is an upwardly branching tree without infinite branches, whose

are the leaves

- nodes are labelled by literals
- the root is labelled by α, and
- if K is the set of labels of the nodes directly above a node with label β, then
  1. either K = ∅ and β ∈ H,
  2. or K/β is a closed substitution instance of a transition rule in T.

closed subst instance of a rule in T



If a proof of H/α from T exists, then H/α is provable from T , notation T |-- H/α.

> Exercise 10
> Formally define closed-term substitutions and their application to terms of a term algebra.

An example

Let's fix the alphabet $V = \{e, c, t, cl\}$

$$\dfrac{\dfrac{\dfrac{\dfrac{e \in V}{e \xrightarrow{e} 1}\ (act)}{e.c \xrightarrow{e} c}\ (seq_2) \qquad c \neq 1}{e.c + e.t \xrightarrow{e} c}\ (cho\ 1) \qquad c \neq 1}{(e.c + e.t).cl \xrightarrow{e} c.cl}\ (seq_1)$$

$$\begin{tikzcd}
(e.c+e.t).cl \arrow[r, "e"] \arrow[d, "e"'] & c.cl \arrow[r, "c"] & cl \arrow[r, "cl"] & 1 \\
t.cl \arrow[urr, "t"'] & & &
\end{tikzcd}$$

Exercise 10
Give the LTS of a*(b+c)

RegExp & their operational semantics

We saw that we can define the language of an FSA $M = (Q, \Sigma, q_0, \delta, F)$ as

$$\mathcal{L}_M = \{ a_1 \ldots a_n \in \Sigma^* \mid \exists q_1, \ldots, q_n \mid q_0 \xrightarrow{a_1} \ldots q_{n-1} \xrightarrow{a_n} q_n \xrightarrow{\checkmark} \bullet \}$$

where $\rightarrow$ is the relation of the LTS corresponding to M

This can be generalised to ANY LTS e.g.

Since the TSS of reg exp induces an LTS, we can use the very same definition to define the language $\mathcal{L}_E$ of a reg exp E; so

$$\mathcal{L}_E = \{ a_1 \ldots a_n \in \Sigma^* \mid \exists E_1, \ldots, E_n : E \xrightarrow{a_1} E_1 \ldots E_{n-1} \xrightarrow{a_n} E_n = 1 \}$$

where now $\xrightarrow{a_i}$ are transition to be proved by applying the rules of our TSS!

Example .

Show that $aab \in \mathcal{L}_E$ where $E = a^*(b + c)$ and $A = \{a, b, c, d\}$

1. find E_1 s.t. $E \xrightarrow{a} E_1$ & there are E_2, E_3 s.t. $E_2 \xrightarrow{a} E_3 \xrightarrow{b} 1$

- a candidate for E_3 is b since (Act) $\dfrac{b \in A}{b \xrightarrow{b} 1}$
- likewise a candidate for E_2 is ab WHY?

$$(\text{seq}_2)\ \dfrac{(\text{Act})\ \dfrac{a \in A}{a \xrightarrow{a} 1}}{ab \xrightarrow{a} b}$$

$$\text{seq}_2\ \dfrac{\text{act}\ \dfrac{a \in A}{a \xrightarrow{a} 1}}{aab \xrightarrow{a} ab} \qquad \dfrac{\dfrac{a \in A}{a \xrightarrow{a} 1}\ \text{act}}{\xrightarrow{a} b}\ \text{seq}_2 \qquad \dfrac{b \in A}{\xrightarrow{b} 1}\ \text{act}$$

A formal model of concurrency [Bergstra et al.]

From regular expressions to process algebras: a model of concurrency

$A_\tau = A \cup \{\tau\}$ $\quad \tau \notin A$

Note the different yet equivalent definition wrt [Bergstra et al.]

$E ::= \ \ldots \mid E \| E$ without Kleene star and 0

$$(Act)\ \frac{a \in A_\tau}{a \xrightarrow{a} 1}$$

$$(Cho_1)\ \frac{x \xrightarrow{a} x' \quad x' \neq 1}{x + y \xrightarrow{a} x'} \qquad (Cho_2)\ \frac{x \xrightarrow{a} 1}{x + y \xrightarrow{a} 1}$$

$$(Cho_3)\ \frac{y \xrightarrow{a} y' \quad y' \neq 1}{x + y \xrightarrow{a} y'} \qquad (Cho_4)\ \frac{y \xrightarrow{a} 1}{x + y \xrightarrow{a} 1}$$

$$(Seq_1)\ \frac{x \xrightarrow{a} x' \quad x' \neq 1}{x \cdot y \xrightarrow{a} x' \cdot y} \qquad (Seq_1)\ \frac{x \xrightarrow{a} 1}{x \cdot y \xrightarrow{a} y}$$

$$(Per_1)\ \frac{x \xrightarrow{a} x' \quad x' \neq 1}{x \| y \xrightarrow{a} x' \| y}$$

$$(Per_2)\ \frac{y \xrightarrow{a} y' \quad y' \neq 1}{x \| y \xrightarrow{a} x \| y'}$$

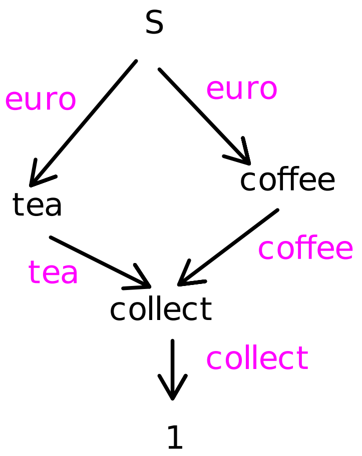
$$(Per_3)\ \frac{x \xrightarrow{a} 1}{x \| y \xrightarrow{a} y}$$

$$(Per_4)\ \frac{y \xrightarrow{a} 1}{x \| y \xrightarrow{a} x}$$

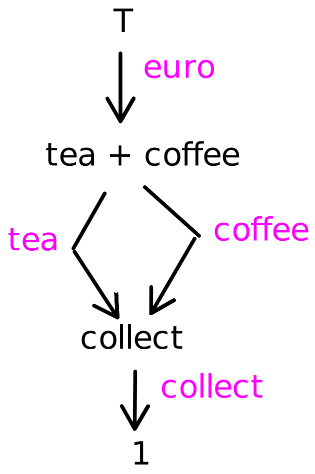
Interleaving semantics

Equivalences of concurrent programs

S = (euro.tea + euro.coffee).collect



T = euro.(tea + coffee).collect



- S and T have the same traces (words), but they differ if interpreted as reactive systems
- For reactive systems, bisimulation is a better notion of equivalence than language (trace) equivalence

Def. Given an LTS T, a binary relation B on the states of T is a bisimulation if whenever (s1,s2) are in B
- for all s1 --a--> s1' there is s2 --a--> s2' such that (s1',s2') is in B and
- for all s2 --a--> s2' there is s1 --a--> s1' such that (s1',s2') is in B

Exercise 11
Let S and T as in the example of the vending machine above. Show that there is no bisimulation containing the pair (S,T).