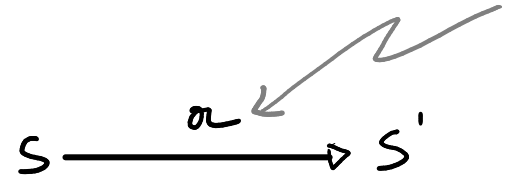
Another (important) variant of TS

A **labelled transition system** is a triple $(S, A, \rightarrow)$ where

- S is a set of states
- A is a set of labels (or actions, or operations, or events, ...)
- $\rightarrow \subseteq S \times A \times S$ $\quad (\rightarrow : S \rightarrow 2^{A \times S})$ transition relation



observable information about what happens during the transition
particularly handy to model communication / concurrency / distribution

Example An FSA $M = (Q, \Sigma, q_0, \delta, F)$ is an LTS:

$LTS_M = (S \cup \{\bullet\}, \Sigma \cup \{\checkmark\}, \rightarrow)$ where $s \xrightarrow{a} s' \Leftrightarrow a \in \Sigma \wedge s' \in \delta(s,a)$ or $a = \checkmark \wedge s' = \bullet \wedge s \in F$

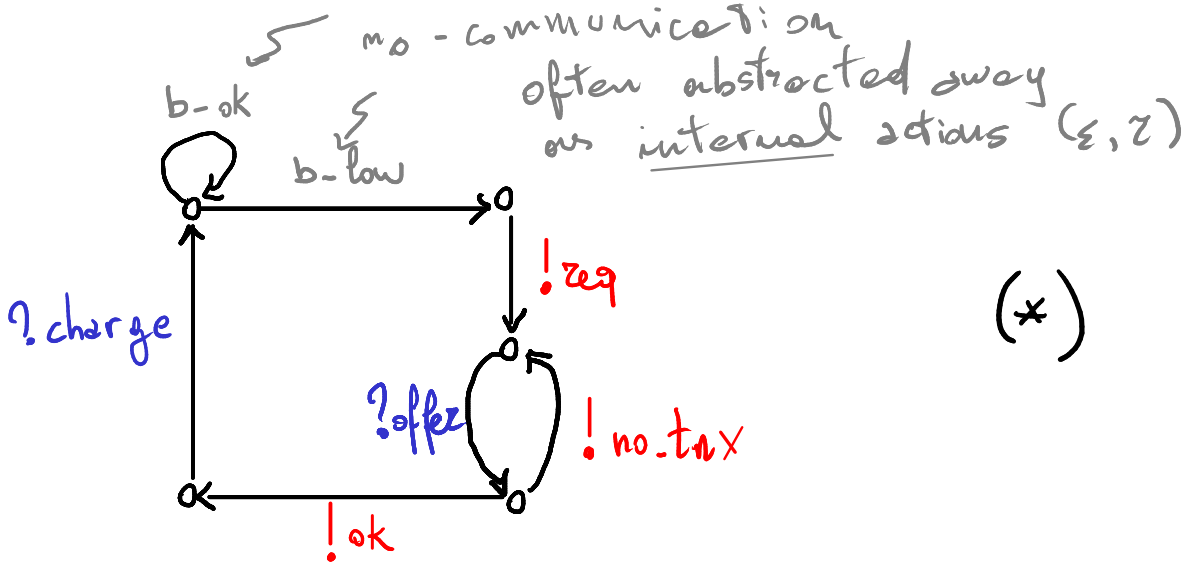
$$\mathcal{L}_M = \{a_1 \ldots a_n \in \Sigma^* \mid \exists q_1, \ldots, q_m \mid q_0 \xrightarrow{a_1} \ldots q_{n-1} \xrightarrow{a_n} q_n \xrightarrow{\checkmark} \bullet\}$$

Communication-based concurrency

A robotic scenario:
Some mobile robots need to manage their energy in order to accoplish their task (e.g., patrolling some premises).

- When their batteries deplete, robots look for a recharge.
- Recharges are offered by recharge stations or other robots.

We can model this behaviour using an LTS capturing the observable features we are interested in: in this case communication
For instance, the behaviour of a robot seeking for a recharge is



The set of labels is the union of

- {b_low, b_ok} internal actions
- {?charge, ?offer} input actions
- {!req, !no_tnx} output actions

Exercise 8
Give an LTS modelling the behaviour of a robot offering a recharge.
Reflect about the "compatibility" between your solution and the LTS (*) above

Transition System Specifications

Aceto, Fokkink, Verhof: Structural Operational Semantics

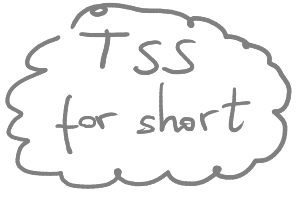
"The first systematic study of TSSs may be found in [208], while the first study of TSSs with negative premises appeared in [57]." (Aceto et al.)

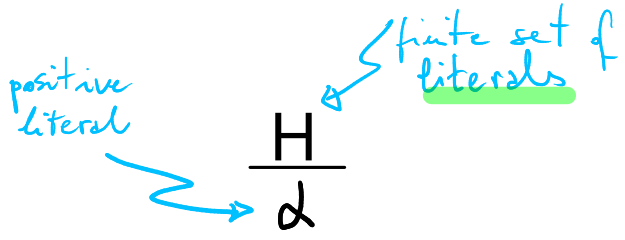
[208] R. d. Simone, Calculabilité et Expressivité dans l'Algèbre de Processus Parallèles Meije, thèse de 3 e cycle, Univ. Paris 7, 1984.

[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 $\mathrm{Term}_{\Sigma,\mathcal{V}}$ and a set of labels A

A transition system specification on A is a set of inference rules





LITERALS

positive $t \xrightarrow{a} t'$ or $t \in T$

negative $t \not\xrightarrow{a}$ or $t \notin T$

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

Operational semantics of regular expressions

A TSS

$$(\text{Act})\ \frac{a \in A}{a \xrightarrow{a} 1} \qquad (\text{Tic})\ \frac{}{1 \xrightarrow{\varepsilon} 1}$$

$$(\text{Seq}_1)\ \frac{x \xrightarrow{a} x' \quad x' \neq 1}{x \cdot y \xrightarrow{a} x' \cdot y} \qquad (\text{Seq}_2)\ \frac{x \xrightarrow{a} 1}{x \cdot y \xrightarrow{a} y}$$

$$(\text{Cho}_1)\ \frac{x \xrightarrow{a} x' \quad x' \neq 1}{x + y \xrightarrow{a} x'} \qquad (\text{Cho}_2)\ \frac{x \xrightarrow{a} 1}{x + y \xrightarrow{a} 1}$$

$$(\text{Cho}_3)\ \frac{y \xrightarrow{a} y' \quad y' \neq 1}{x + y \xrightarrow{a} y'} \qquad (\text{Cho}_4)\ \frac{y \xrightarrow{a} 1}{x + y \xrightarrow{a} 1}$$

$$(\text{Str}_1)\ \frac{}{x^* \xrightarrow{\varepsilon} 1} \qquad (\text{Str}_2)\ \frac{x \xrightarrow{a} x'}{x^* \xrightarrow{a} x' \cdot x^*}$$

Note that
- x & y range over the set of reg exp
- these rules form a TSS
- each operator has a set of rules (including 0, which has $\emptyset$!)

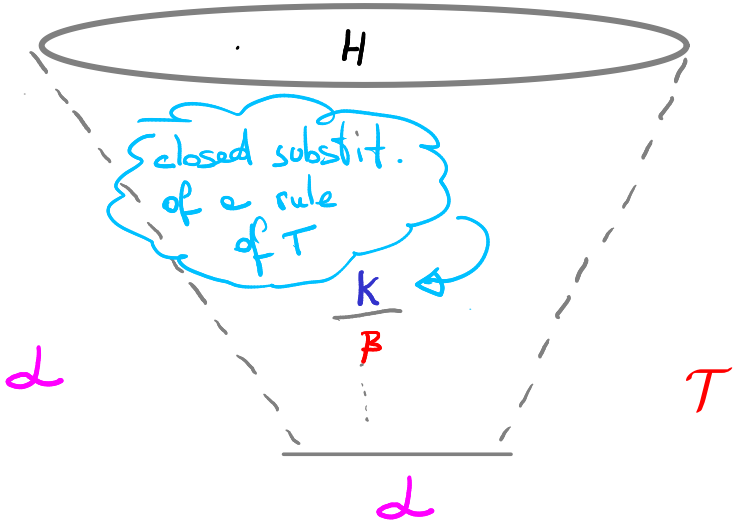
Basic Process Algebras with $a \in Act$

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

LTSs as proofs of TSSs

A proof in a TSS $\mathcal{T}$ of a closed transition rule $\frac{H}{\alpha}$ is an upwardly finitely-branching tree whose

- nodes are (labelled by) literals with the root (labelled by) α
- the leaves of the tree are the literals in H
- if K is the set of children of β then there is $\frac{H'}{\alpha'} \in \mathcal{T}$ and $\sigma : \mathcal{V} \to \mathrm{Term}_{\Sigma,\phi}$ s.t. $K = H'\sigma$ and $\beta = \alpha'\sigma$



H *provable* in $\mathcal{T}$, written $\mathcal{T} \vdash \frac{H}{\alpha}$, if there is a proof of $\frac{H}{\alpha}$ in $\mathcal{T}$

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

An example

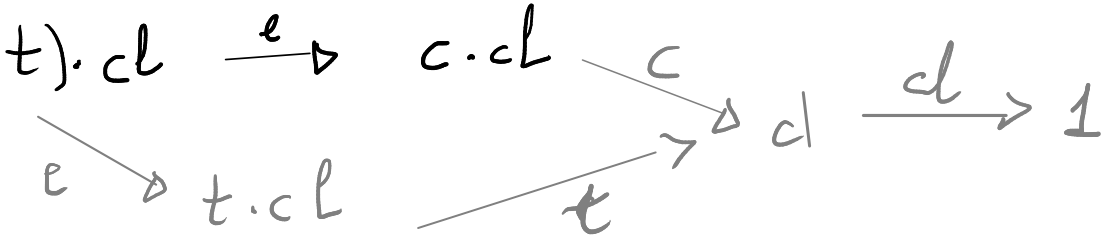
Let's fix the alphabet $V = \{e, c, t, cl\}$

$$(act)\ \frac{e \in V}{e \xrightarrow{e} 1}$$

$$(seq_2)\ \frac{e \xrightarrow{e} 1}{e.c \xrightarrow{e} c} \quad c \neq 1$$

$$(cho_1)\ \frac{e.c \xrightarrow{e} c \qquad c \neq 1}{e.c + e.t \xrightarrow{e} c}$$

$$(seq_1)\ \frac{e.c + e.t \xrightarrow{e} c \qquad c \neq 1}{(e.c + e.t).cl \xrightarrow{e} c.cl}$$



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

$$\mathcal{L}_M = \{ a_1 \dots a_n \in \Sigma^* \mid \exists q_1, \dots, q_m \mid q_0 \xrightarrow{a_1} \dots q_n \xrightarrow{a_n} q_n \xrightarrow{\checkmark} \bullet \}$$

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 $\bar{E}$; so

$$\mathcal{L}_E = \{ a_1 \dots a_n \in \Sigma^* \mid \exists \bar{E}_1, \dots, \bar{E}_n : \bar{E} \xrightarrow{a_1} \bar{E}_1 \dots \bar{E}_{n-1} \xrightarrow{a_n} E_n = 1 \}$$

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

Example

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

1. find E_1, E_2 s.t. $E_1 \xrightarrow{a} E_2$ & there is 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? $\rightsquigarrow$ (Act) $\dfrac{a \in A}{a \xrightarrow{a} 1}$, (seq_2) $\dfrac{a \xrightarrow{a} 1}{ab \xrightarrow{a} b}$

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

$$aab \xrightarrow{a} ab \xrightarrow{a} b \xrightarrow{b} 1$$

A formal model of concurrency [Bergstra et al.]

From regular expressions to process algebras: a model of concurrency

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

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

$E ::= \dots \mid E \parallel E$ without Kleene star and 0

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

$$(cho_1)\ \frac{x \xrightarrow{a} x' \quad x' \neq 1}{x + y \xrightarrow{a} x'} \qquad (cho_2)\ \frac{x \xrightarrow{a} 1}{x + y \xrightarrow{a} 1}$$

$$(cho_3)\ \frac{y \xrightarrow{a} y' \quad y' \neq 1}{x + y \xrightarrow{a} y'} \qquad (cho_4)\ \frac{y \xrightarrow{a} 1}{x + y \xrightarrow{a} 1}$$

$$(seq_1)\ \frac{x \xrightarrow{a} x' \quad x' \neq 1}{x \cdot y \xrightarrow{a} x' \cdot y} \qquad (seq_1)\ \frac{x \xrightarrow{a} 1}{x \cdot y \xrightarrow{a} y}$$

Interleaving semantics

$$(per_1)\ \frac{x \xrightarrow{a} x' \quad x' \neq 1}{x \parallel y \xrightarrow{a} x' \parallel y}$$

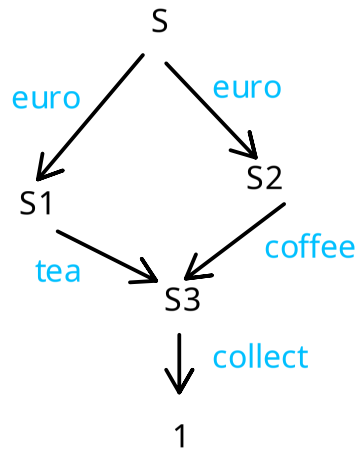
$$(per_2)\ \frac{y \xrightarrow{a} y' \quad y' \neq 1}{x \parallel y \xrightarrow{a} x \parallel y'}$$

$$(per_3)\ \frac{x \xrightarrow{a} 1}{x \parallel y \xrightarrow{a} y}$$

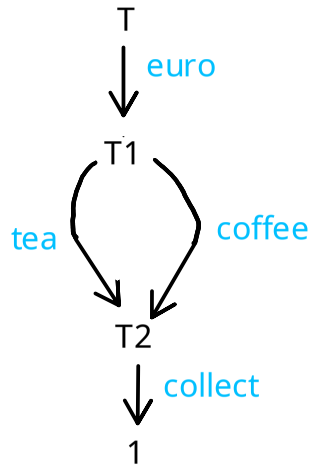
$$(per_4)\ \frac{y \xrightarrow{a} 1}{x \parallel y \xrightarrow{a} x}$$

Equivalences of concurrent programs

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



T = euro.(tea + coffee).collect



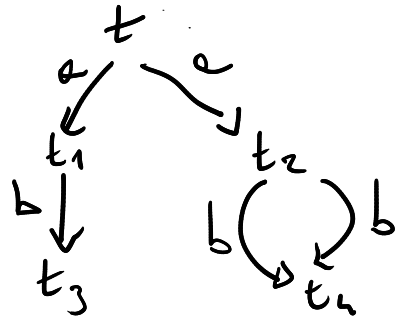
- S and T have the same traces (words), but they differ if interpreted as reactive systems
- For reactive systems, bisimulation is a better notion of equivalence than language (trace) equivalence

Def. Given an LTS T, a binary relation B on the states of T is a bisimulation if whenever q B r
- for all q --a--> q' there is r --a--> r' such that q' B r' and
- for all r --a--> r' there is q --a--> q' such that q' B r'

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).

Odd cases

Bisim. ?



$B = \{ (s,t) , (s_1,t_1) , (s_1,t_2) , (t_1,t_2) , (s_2,t_3) , (s_2,t_4) , (t_3,t_4) \}$