Modeling & Verification

Introduction & Labelled Transition Systems

Max Tschaikowski (tschaikowski@cs.aau.dk)
Slides courtesy of Giorgio Bacci

Aims of the Course

Present a general theory of *Reactive Systems* and its applications

Design + **Specification** + **Verification**

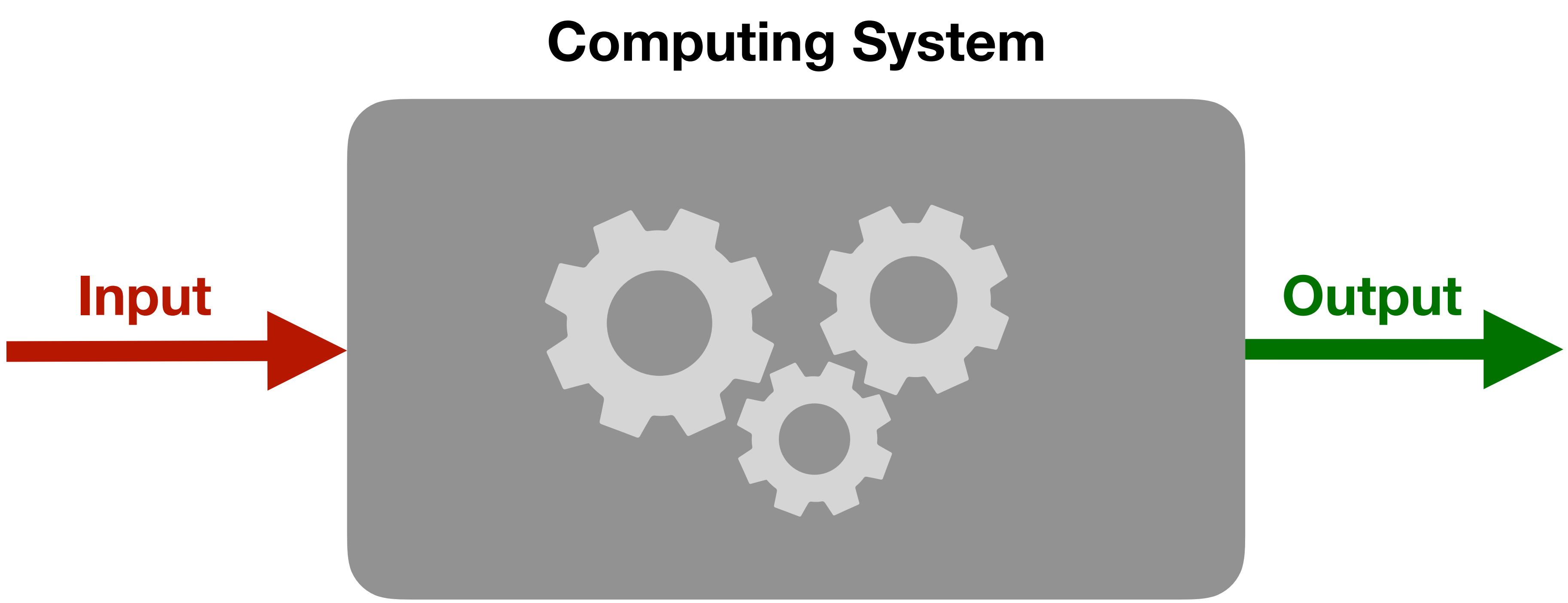
possibly automatic!

- Give the students practice in modelling systems in a formal framework
- Give the students skills in analysing behaviours of reactive systems
- Introduce Algorithms and Tools based on the modelling formalism

What is a Reactive System?

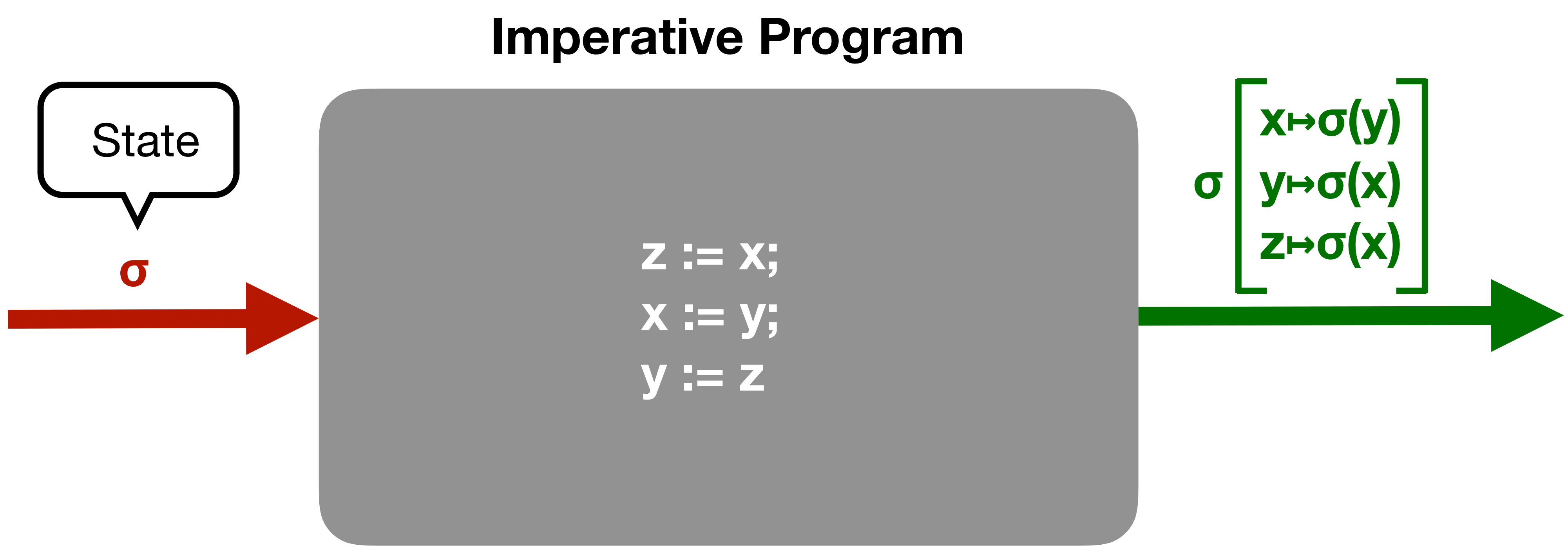
the "standard" view

A computing system at a high level of abstraction
can be considered as a black box



The meaning of programs

An algorithm is specified as a collection of legal inputs and, for each legal input it is associated an output



the semantics is a function
[States → States]

The meaning of programs

An algorithm is specified as a collection of legal inputs and, for each legal input it is associated an output

Imperative Program

the semantics is a partial function
[States → States]

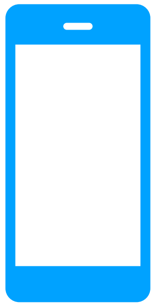
Semantics

- Each input is associated with an output value
- **Non-termination** = undefined output

is bad!

- In case of termination the result is unique

is this all we need?



mobile phones

operating systems

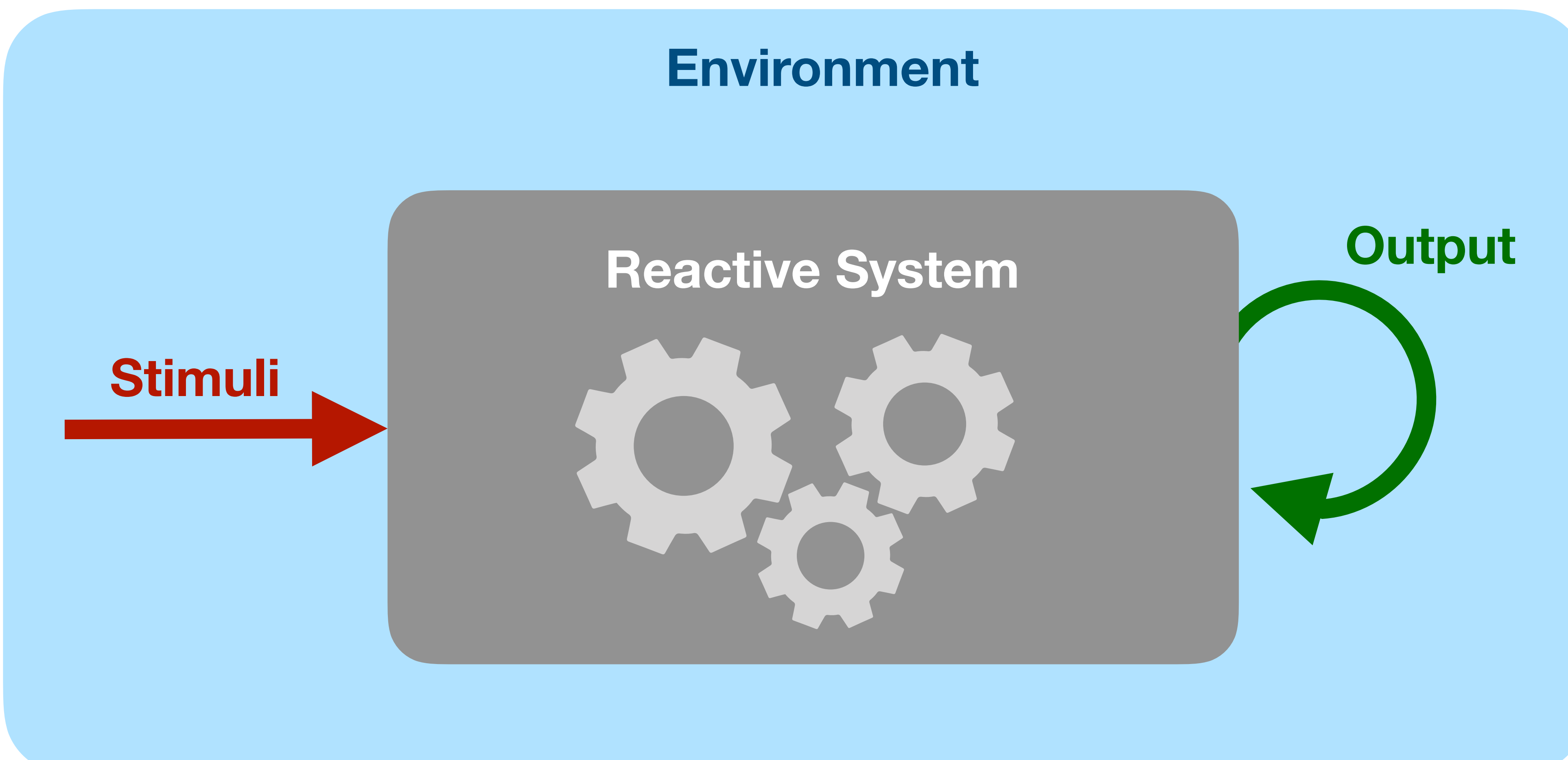
communication protocols

vending machines

Reactive System

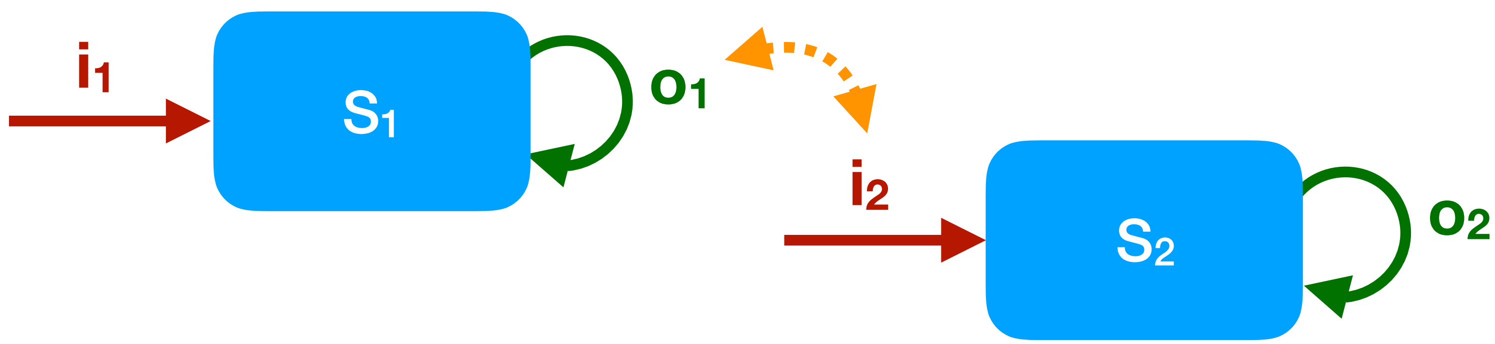
(Harel and Pnueli '85)

It is a computing system that reacts to stimuli from its environment



Key issues

Reactive systems have to deal with issues like:



- Communication & Interaction
- Parallelism (i.e., concurrency)
- The result (if any) does not have to be unique!

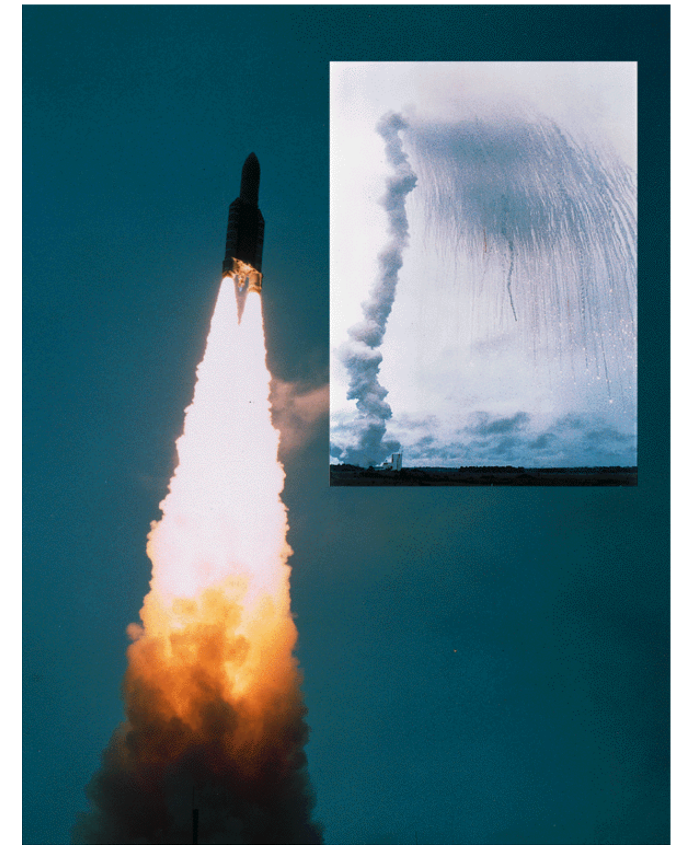
Sad but true...

Even small parallel systems
are hard to test and may be source of errors

Toyota Prius

- First mass-produced hybrid vehicle
- **February 2010**
  - software “glitch” found in Anti-lock Breaking System
  - in response of numerous complains/accidents
- **Eventually fixed via software update**
  - in total ~185k cars recalled —huge cost!
  - handling of the incident prompted criticism & bad publicity

Ariane 5



- ESA (European Space Agency) rocket designed to launch commercial payloads (e.g. satellites) into Earth orbit
- **First test flight (4th June 1996)**
  - it self-destructed 37 secs after launch
- **Uncaught Realtime Exception**
  - numerical overflow in a conversion routine resulted in incorrect altitude calculation by the on-board computer
  - Expensive, embarrassing…

How can we design/develop a system that works?

Modeling & Verification

How do we analyse such a system?

Labelled Transition Systems

Definition

A *Labelled Transition System* (LTS) is a tuple

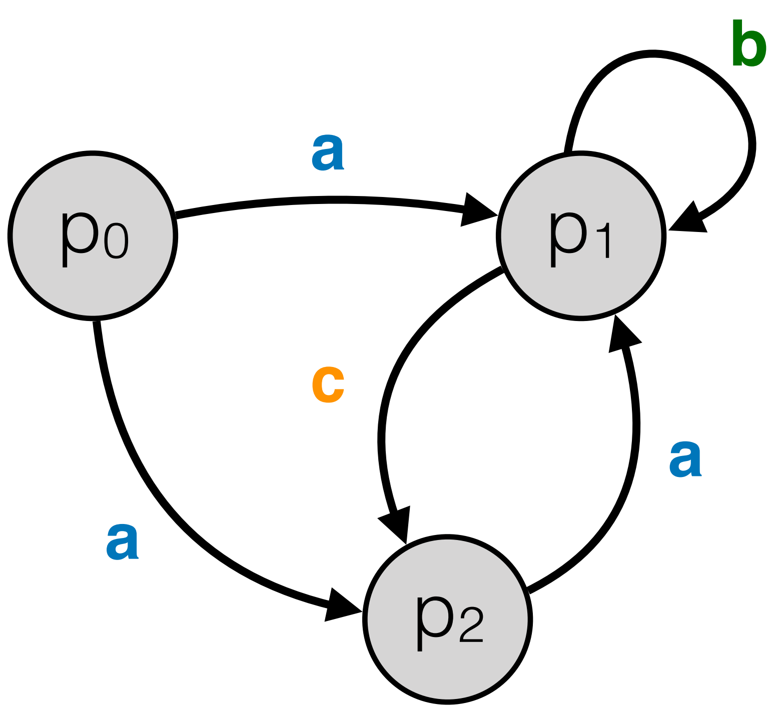
$$(\mathrm{Proc}, \mathrm{Act}, \{\xrightarrow{\alpha} \mid \alpha \in \mathrm{Act}\})$$

where

- Proc is a set of *states* (or *processes*)
- Act is a set of *actions* (or *labels*)
- for each $\alpha \in \mathrm{Act}$, $\xrightarrow{\alpha} \subseteq \mathrm{Proc} \times \mathrm{Proc}$ is a binary relation called *transition relation*

Sometimes we distinguish the *initial* (or *start*) state.

LTS (Example)



$\mathrm{Proc} = \{p_0, p_1, p_2\}$

$\mathrm{Act} = \{a, b, c\}$

$\xrightarrow{a} = \{(p_0, p_1), (p_0, p_2), (p_2, p_1)\}$

$\xrightarrow{b} = \{(p_1, p_1)\}$

$\xrightarrow{c} = \{(p_1, p_2)\}$

For convenience we will use the infix notation $p \xrightarrow{\alpha} p'$ meaning that $(p,p') \in \xrightarrow{\alpha}$

Recap of Binary Relations

Definition

A *binary relation on set A* is a subset $R \subseteq A \times A$.

Sometimes we write $a \; R \; a'$ instead of $(a,a') \in R$.

Properties

- R is *reflexive* if $(a,a) \in R$, for all $a \in A$,
- R is *symmetric* if $(a,b) \in R$ implies $(b,a) \in R$, for all $a,b \in A$
- R is *transitive* if $(a,b) \in R$ and $(b,c) \in R$ implies that $(a,c) \in R$, for all $a,b,c \in A$

Reflexive Closure

Let R, R', and R" be binary relations on a set A

Definition

R' is the *reflexive closure* of R if and only if

- R ⊆ R'
- R' is reflexive, and
- R' is the *smallest* relation satisfying the two conditions above, i.e., for any relation R", if R ⊆ R" and R" is reflexive, then R' ⊆ R"

Symmetric Closure

Let R, R', and R" be binary relations on a set A

Definition

R' is the *symmetric closure* of R if and only if

- $R \subseteq R'$
- R' is symmetric, and
- R' is the *smallest* relation satisfying the two conditions above, i.e., for any relation R", if $R \subseteq R''$ and R" is symmetric, then $R' \subseteq R''$

Transitive Closure

Let R, R', and R" be binary relations on a set A

Definition

R' is the *transitive closure* of R if and only if

- $R \subseteq R'$
- R' is transitive, and
- R' is the *smallest* relation satisfying the two conditions above, i.e., for any relation R",
  if $R \subseteq R''$ and R" is transitive, then $R' \subseteq R''$

LTS - Notation

Let (Proc, Act, $\{\xrightarrow{\alpha} \mid \alpha \in$ Act $\}$) be an LTS.

We can extend $\xrightarrow{\alpha}$ from labels $\alpha \in$ Act to words $w \in$ Act*

- $p \xrightarrow{\varepsilon} p$, for every $p \in$ Proc, and
- $p \xrightarrow{\alpha w} p'$, if there is $p'' \in$ Proc, such that $p \xrightarrow{\alpha} p''$ and $p'' \xrightarrow{w} p'$ for every $p,p' \in$ Proc, $\alpha \in$ Act, and $w \in$ Act*

Intuitively

If $w = \alpha_1\alpha_2\ldots\alpha_n$ then $p \xrightarrow{w} p'$ whenever

$$p = p_0 \xrightarrow{\alpha_1} p_1 \xrightarrow{\alpha_2} p_2 \xrightarrow{\alpha_3} \ldots \xrightarrow{\alpha_{n-1}} p_{n-1} \xrightarrow{\alpha_n} p_n = p'$$

LTS - Notation

Let (Proc, Act, $\{\xrightarrow{\alpha} \mid \alpha \in Act\}$) be an LTS.

- $\longrightarrow$ = $\{(p,p') \mid$ if $p \xrightarrow{\alpha} p'$, for some $\alpha \in Act\}$
- $\longrightarrow^*$ is the *reflexive* and *transitive* closure of $\longrightarrow$
- $p \xrightarrow{\alpha}$ if there exist $p' \in$ Proc such that $p \xrightarrow{\alpha} p'$
- $p \not\xrightarrow{\alpha}$ if there is no $p' \in$ Proc such that $p \xrightarrow{\alpha} p'$
- reachable states

Introduction to CCS

Calculus of Communicating Systems (Milner'89)

Towards a Process Algebra

Robin Milner (1989) observed that concurrent processes have an algebraic structure

$$P_1 * P_2 = P_1 * P_2$$

Process Algebra

Basic Principle

- Define a set of *atomic processes* modelling the simplest process behaviour
- Define *operators between processes*.
  These allow one to build complex behaviours from simple ones.

Example

Imperative Parallel Programs

- **Atomic instructions:**
  - skip
  - assignment (eg. x:=2 and x:=x+1)
- **Operators between programs:**
  - sequential composition (P_1 ; P_2)
  - parallel composition (P_1 || P_2)

(x:=1 || x:=2) ; x:=x+2 ; (x:=x-1 || x:=x+5)

is a parallel program

Semantics (small-step)

$$\frac{}{\langle x := e, \sigma\rangle \longrightarrow \langle \mathrm{skip}, \sigma[x \mapsto v(e)]\rangle}\ \textit{(Assignment)}$$

$$\frac{\langle P, \sigma\rangle \longrightarrow \langle P', \sigma'\rangle}{\langle P;Q, \sigma\rangle \longrightarrow \langle P';Q, \sigma'\rangle}\ \textit{(seq-1)}$$

$$\frac{\langle P, \sigma\rangle \not\longrightarrow}{\langle P;Q, \sigma\rangle \longrightarrow \langle Q, \sigma\rangle}\ \textit{(seq-2)}$$

$$\frac{\langle P, \sigma\rangle \longrightarrow \langle P', \sigma'\rangle}{\langle P \| Q, \sigma\rangle \longrightarrow \langle P' \| Q, \sigma'\rangle}\ \textit{(par-L)}$$

$$\frac{\langle Q, \sigma\rangle \longrightarrow \langle Q', \sigma'\rangle}{\langle P \| Q, \sigma\rangle \longrightarrow \langle P \| Q', \sigma'\rangle}\ \textit{(par-R)}$$

Features of the Semantics

- It is compositional (the behaviour of processes is described in terms of the behaviour of its constituents)
- It is non-deterministic
- It is terminating by construction! (usually we are not that "lucky")

CCS - Sequential Fragment

We would like to have language (process algebra) that is able to talk about *reactive systems* (LTS!)

- Nil (or 0) process (the only atomic process)
- action prefix (**a**.P)
- names and recursive definitions ($\stackrel{\text{def}}{=}$)
- non-deterministic choice (P+Q)

Any finite LTS can be described (up to isomorphism) by using this set of operations

What about communication and interaction?
... to be continued