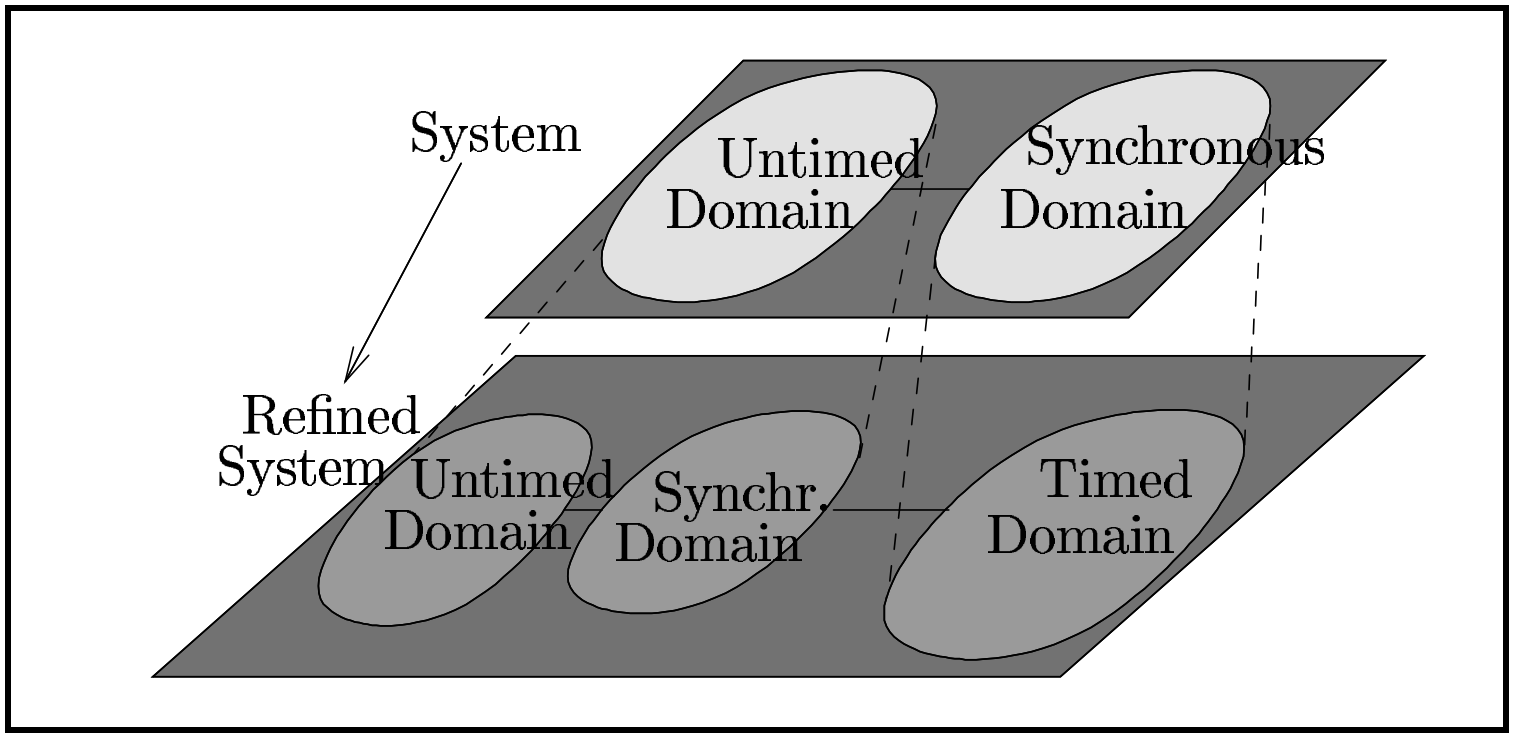
System Modeling

Introduction
Rugby Meta-Model
Finite State Machines
Petri Nets

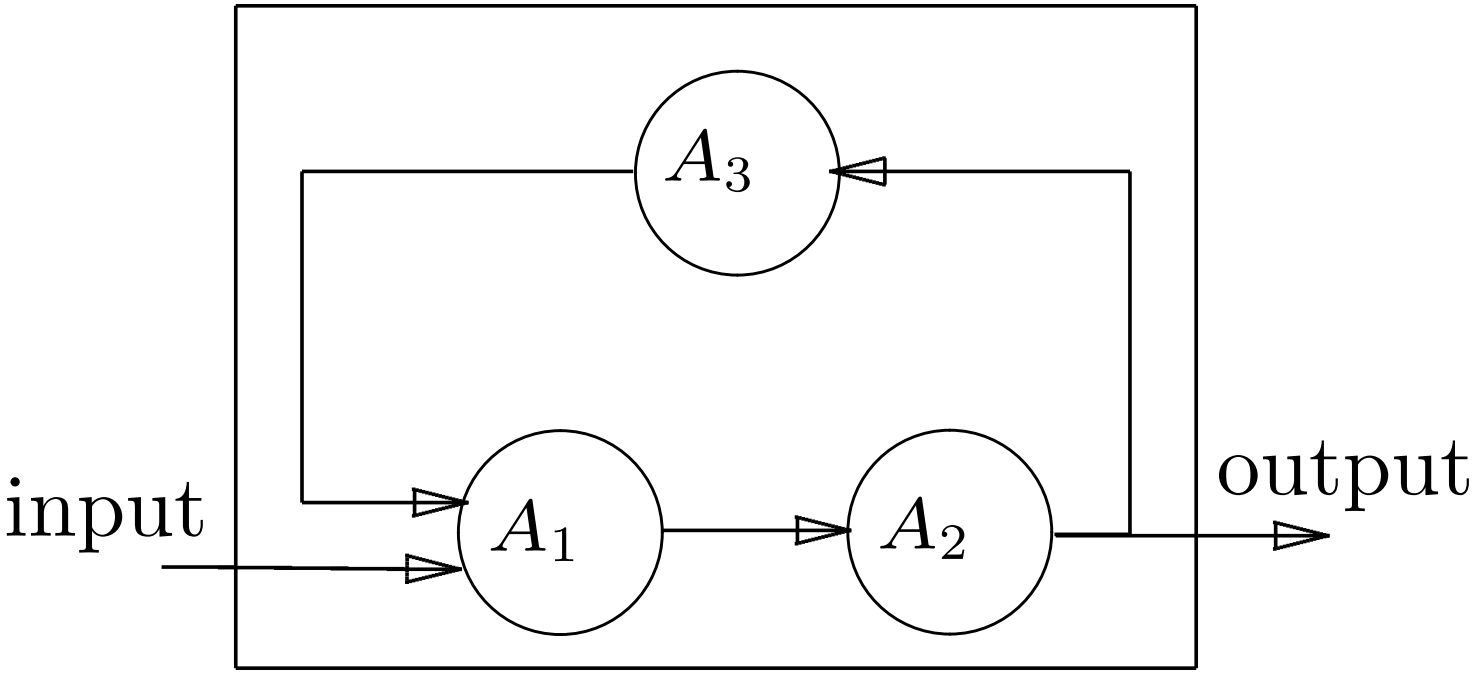
Untimed Model of Computation

Synchronous Model of Computation
Timed Model of Computation
Integration of Computational Models
Tightly Coupled Process Networks
Nondeterminism and Probability
Applications

Coexistence of Different MoC Domains



An Amplifier as a Network of Three Processes

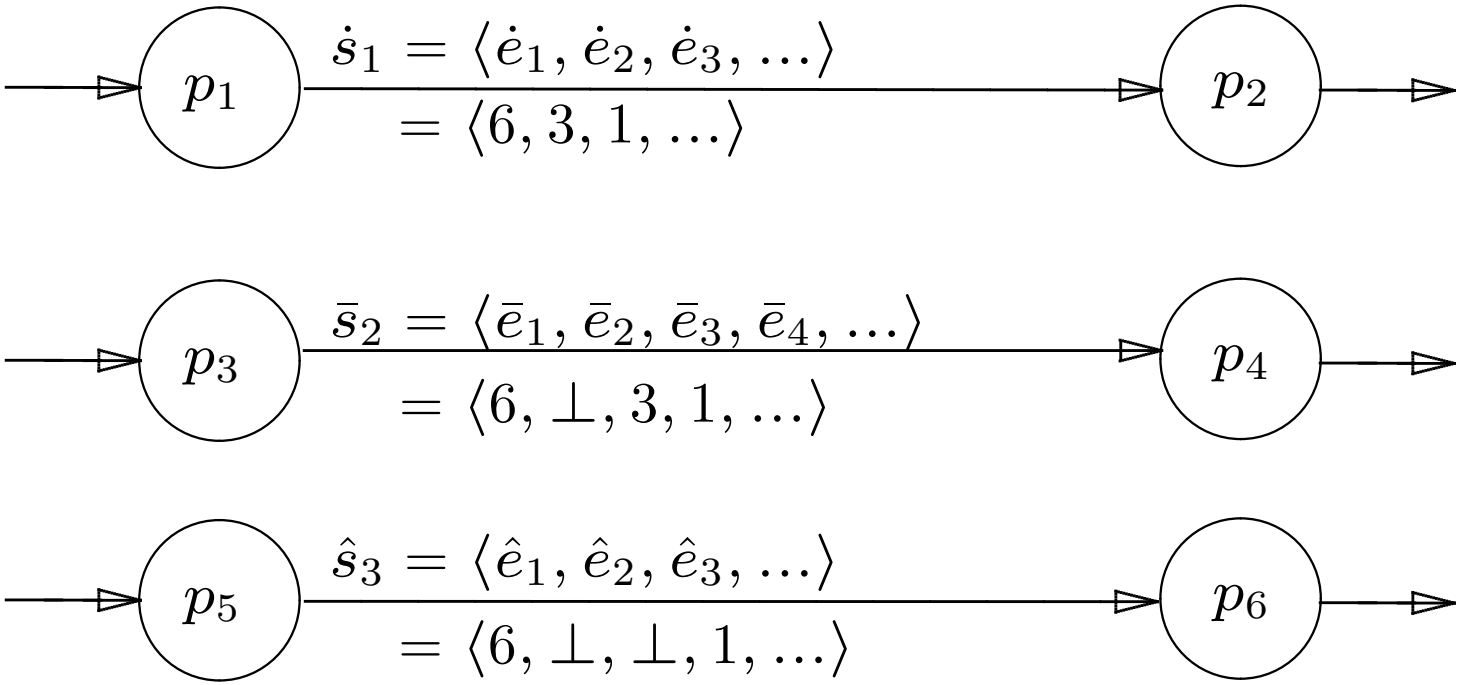


A_1 merges input data with a control signal;
A_2 amplifies the input signal by applying the control signal;
A_3 analyses the amplified signal and produces a new control value;

Events and Signals

- Untimed events $\dot{E} = V$;
- Synchronous events $\bar{E} = V \cup \{\sqcup\}$;
- Timed events $\hat{E} = \bar{E}$;
- Any kind of event $E = \dot{E} \cup \bar{E} \cup \hat{E}$;
- Untimed signals $\dot{S} : \dot{s} = \langle \dot{e}_0, \dot{e}_1, \dot{e}_2, ... \rangle, \dot{e}_i \in \dot{E}$;
- Synchronous signals $\bar{S} : \bar{s} = \langle \bar{e}_0, \bar{e}_1, \bar{e}_2, ... \rangle, \bar{e}_i \in \bar{E}$;
- Timed signals $\hat{S} : \hat{s} = \langle \hat{e}_0, \hat{e}_1, \hat{e}_2, ... \rangle, \hat{e}_i \in \hat{E}$;
- Any kind of signal $S : s = \langle e_0, e_1, e_2, ... \rangle, e_i \in E$;
- Empty signal $\langle\rangle$;

Processes Connected by Untimed, Synchronous and Timed Signals



Operations on Signals

Concatenation of signals $s_1 \oplus (s_2 \oplus s_3) = (s_1 \oplus s_2) \oplus s_3$,
$\langle\rangle \oplus s = s \oplus \langle\rangle = s$.
Length of a signal: $\#s$, e.g. $\#\langle e, e'\rangle = 2, \#\langle\rangle = 0$;
Indexing of signals: $[\,] : S \times \mathbb{N} \to E$,
e.g. $s = \langle e_1, e_2, e_3\rangle, s[2] = e_2$;

$$\text{take}(n, s) = \begin{cases} \langle e_0, ..., e_{n-1}\rangle & \text{if } \#s \geq n \\ s & \text{otherwise} \end{cases}$$

$$\text{drop}(n, s) = \begin{cases} \langle e_n, ..., e_{\#s-1}\rangle & \text{if } \#s \geq n \\ \langle\rangle & \text{otherwise} \end{cases}$$

$$\text{head}(s) = \begin{cases} e_0 & \text{if } s \neq \langle\rangle \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$\text{tail}(s) = \text{drop}(1, s)$$

Signal Partitioning

Definition: Let $\nu : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ be a function on natural numbers and $s \in S$ be a signal. The partition $\pi(\nu, s)$ is defined as follows.

$$\pi(\nu, s) = \mathsf{parts}(\nu, 0, s) \ \forall s \in S$$

We write $\pi(\nu, s) = \langle r_i \rangle$ with $i = 0, 1, 2, ...$

The remainder $\mathsf{rem}(\pi, \nu, s)$ is

$$s = (\bigoplus_{\langle r_i \rangle = \pi(\nu, s)} r_i) \oplus \mathsf{rem}(\pi, \nu, s), \ \ s \in S, i \in \mathbb{N}_0.$$

$$\mathsf{parts}(\nu, i, s) = \begin{cases} \langle\langle \mathsf{take}(\nu(i), s) \rangle\rangle \oplus \mathsf{parts}(\nu, i+1, \mathsf{drop}(\nu(i), s)) & \text{if } \#s \geq \nu(i) \\ \langle\rangle & \text{otherwise} \end{cases}$$

for all $s \in S, i \in \mathbb{N}_0$.

Signal Partitioning Example

$s = \langle e_0, e_1, e_2, e_3, e_4, e_5, ... \rangle$

$\pi(\nu, s) = \langle r_i \rangle$ for $\nu(i) = 3$ for all i

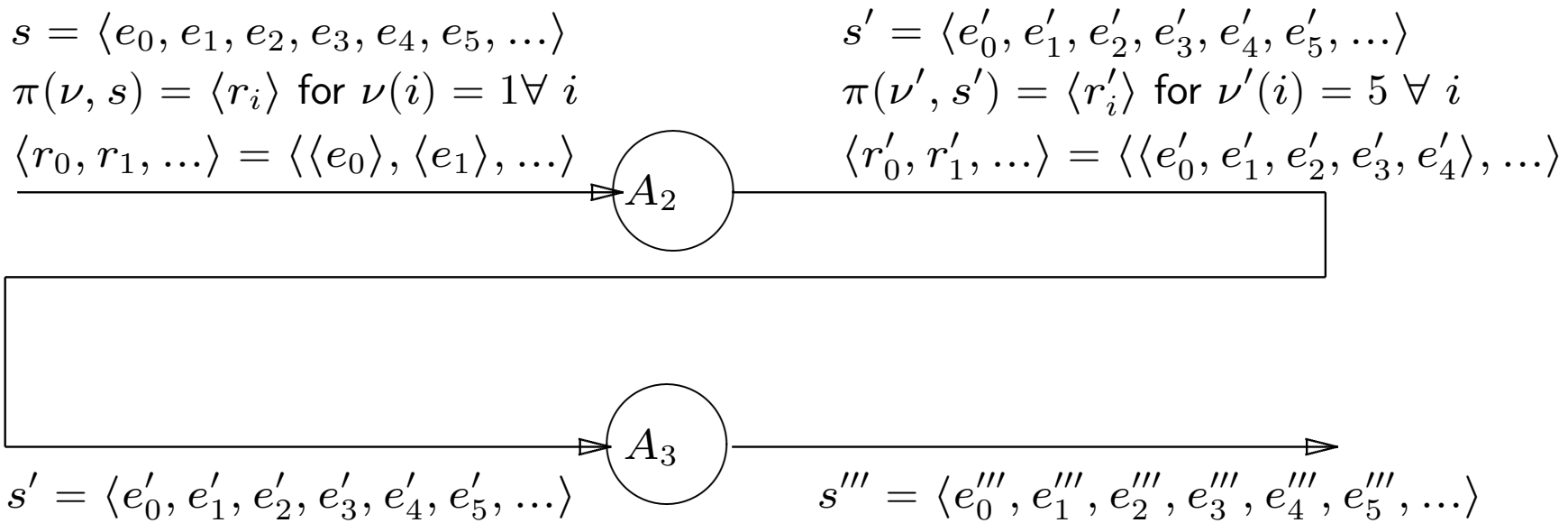
$\langle r_0, r_1, ... \rangle = \langle \langle e_0, e_1, e_2 \rangle, \langle e_3, e_4, e_5 \rangle, ... \rangle$

$s' = \langle e'_0, e'_1, e'_2, e'_3, e'_4, e'_5, ... \rangle$

$\pi(\nu', s') = \langle r'_i \rangle$ for $\nu'(i) = 2$ for all i

$\langle r'_0, r'_1, ... \rangle = \langle \langle e'_0, e'_1 \rangle, \langle e'_2, e'_3 \rangle, ... \rangle$

Signal Partitioning in the Amplifier

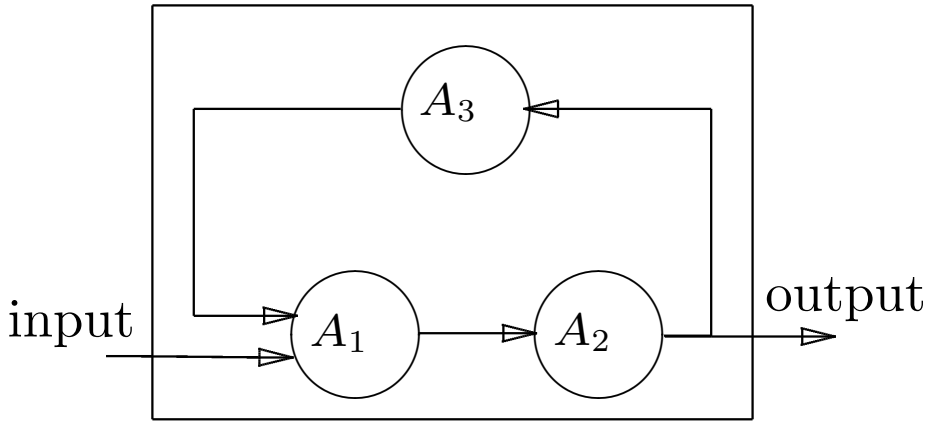


Process Constructors

Process constructors are templates to instantiate processes.

Level	Name	Constructor	Description
1	map	*mapU*	Processes without internal state.
2	scan	*scanU*	Processes with an internal state and a next-state function. The state is directly visible at the output.
3	moore	*mooreU*	Processes with a state; the output is a function of the state, but not directly of the input.
3	mealy	*mealyU*	Processes with a state; the output is a function of the state and the current input.

A Map-based Process



$$A_2 = \texttt{mapU}(c, f)$$

where $c = 1$

$$f((\langle x \rangle, \langle y_1, y_2, y_3, y_4, y_5 \rangle)) = \langle xy_1, xy_2, xy_3, xy_4, xy_5 \rangle$$

$$A_2(\langle (\langle 10 \rangle, \langle 1, 2, 3, 4, 5 \rangle), (\langle 10 \rangle, \langle 6, 7, 8, 9, 10 \rangle) \rangle)$$

$$= \langle 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 \rangle$$

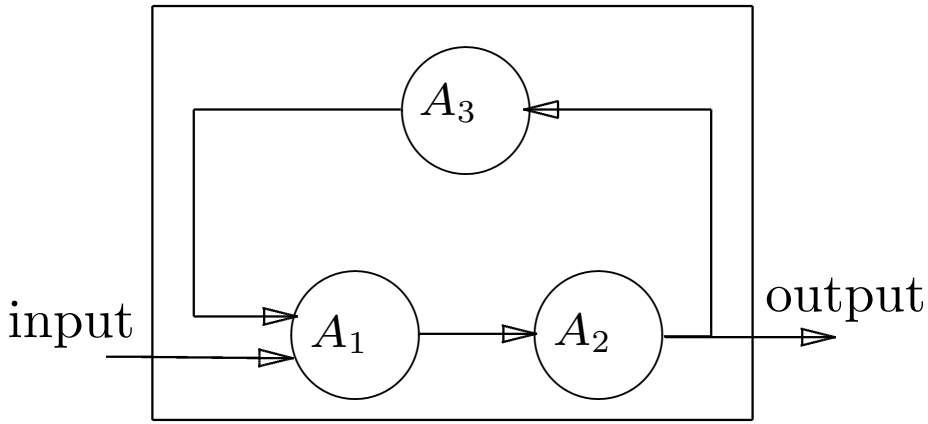
The *mapU* Process Constructor

$$
\begin{aligned}
\texttt{mapU}(c, f) \quad &= \quad p \\
\text{where} \quad & \quad p(\dot{s}) = \dot{s}' \\
& \quad f(\dot{a}_i) = \dot{a}'_i \\
& \quad \pi(\nu, \dot{s}) = \langle \dot{a}_i \rangle,\ \nu(i) = c \\
& \quad \pi(\nu', \dot{s}') = \langle \dot{a}'_i \rangle,\ \nu'(i) = \#f(\dot{a}_i)
\end{aligned}
$$

The *scanU* Process Constructor

$$
\begin{aligned}
\texttt{scanU}(\gamma, g, w_0) \quad &= \quad p \\
\text{where} \qquad & \quad p(\dot{s}) = \dot{s}' \\
& \quad g(\dot{a}_i, w_i) = w_{i+1} \\
& \quad \langle w_{i+1} \rangle = \dot{a}'_i \\
& \quad \pi(\nu, \dot{s}) = \langle \dot{a}_i \rangle,\ \nu(i) = \gamma(w_i) \\
& \quad \pi(\nu', \dot{s}') = \langle \dot{a}'_i \rangle,\ \nu'(i) = \#g(\dot{a}_i) = 1 \\
& \quad i \in \mathbb{N}_0
\end{aligned}
$$

A Scan-based Process



$$A_3 = \texttt{scanU}(\gamma, g, w_0)$$

where

$$w_0 = 10$$

$$\gamma(w_i) = 5 \; \forall i \in \mathbb{N}_0$$

$$g(w_i, \langle x_1, x_2, x_3, x_4, x_5 \rangle) = \begin{cases} w_i - 1 & \text{if } x_1 + x_2 + x3 + x4 + x5 > 500 \\ w_i + 1 & \text{if } x_1 + x_2 + x3 + x4 + x5 < 400 \\ w_i & \text{otherwise} \end{cases}$$

$$A_3(\langle 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 \rangle) = \langle 11, 11 \rangle.$$

The `mealyU` Process Constructor

$$\texttt{mealyU}(\gamma, g, f, w_0) = p$$

where

$$p(\dot{s}) = \dot{s}'$$

$$f(w_i, \dot{a}_i) = \dot{a}'_i$$

$$g(w_i, \dot{a}_i) = w_{i+1}$$

$$\pi(\nu, \dot{s}) = \langle \dot{a}_i \rangle,\ \nu(i) = \gamma(w_i)$$

$$\pi(\nu', \dot{s}') = \langle \dot{a}'_i \rangle,\ \nu'(i) = \#f(w_i, \dot{a}_i)$$

The `mooreU` Process Constructor

$$
\begin{aligned}
\texttt{mooreU}(\gamma, g, f, w_0) \quad &= \quad p \\
\text{where} \quad & p(\dot{s}) = \dot{s}' \\
& f(w_i) = \dot{a}'_i \\
& g(w_i, \dot{a}_i) = w_{i+1} \\
& \pi(\nu, \dot{s}) = \langle \dot{a}_i \rangle,\ \nu(i) = \gamma(w_i) \\
& \pi(\nu', \dot{s}') = \langle \dot{a}'_i \rangle,\ \nu'(i) = \#f(w_i)
\end{aligned}
$$

The *zipU* Process Constructor

$$
\begin{aligned}
\texttt{zipU}(\gamma_a, \gamma_b) \quad &= \quad p \\
\text{where} \quad & p(\dot{s}_a, \dot{s}_b, \dot{s}_c) = \dot{s}' \\
& \langle \dot{a}_i, \dot{b}_i \rangle = \dot{e}'_i \\
& \pi(\nu_a, \dot{s}_a) = \langle \dot{a}_i \rangle,\ \nu_a(i) = \gamma_a(\dot{c}_i) \\
& \pi(\nu_b, \dot{s}_b) = \langle \dot{b}_i \rangle,\ \nu_b(i) = \gamma_b(\dot{c}_i) \\
& \pi(\nu_c, \dot{s}_c) = \langle \dot{c}_i \rangle,\ \nu_c(i) = 1 \\
& \pi(\nu', \dot{s}') = \langle\langle \dot{e}'_i \rangle\rangle,\ \nu'(i) = 1
\end{aligned}
$$

The *unzipU* Process Constructor

$$
\begin{aligned}
\mathit{unzipU}() \quad &= \quad p \\
\text{where} \quad & \quad p(\dot{s}) = \langle \dot{s}', \dot{s}'' \rangle \\
& \quad \dot{e}_i = \langle \dot{a}'_i, \dot{a}''_i \rangle \\
& \quad \pi(\nu, \dot{s}) = \langle \dot{e}_i \rangle,\ \nu(i) = 1 \\
& \quad \pi(\nu', \dot{s}') = \langle \dot{a}'_i \rangle,\ \nu'(i) = \#\dot{a}'_i \\
& \quad \pi(\nu'', \dot{s}'') = \langle \dot{a}''_i \rangle,\ \nu''(i) = \#\dot{a}''_i
\end{aligned}
$$

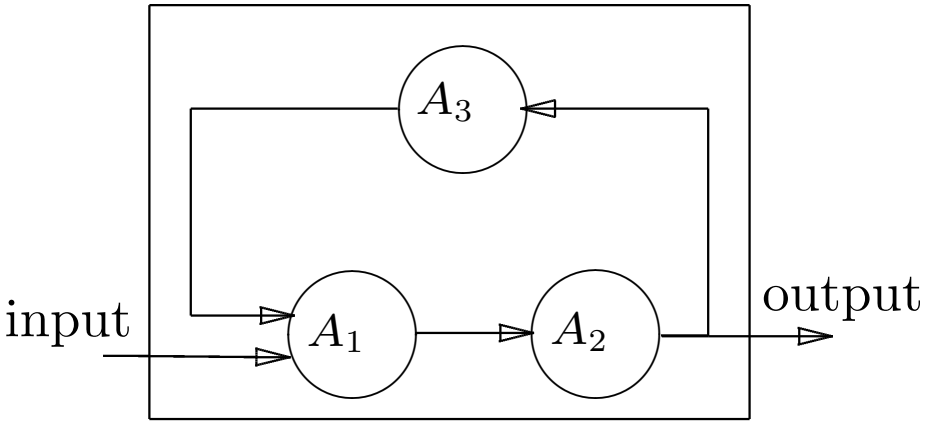
The *zipUs* Process Constructor

$$
\begin{aligned}
\texttt{zipUs}(c_1, c_2) \quad &= \quad p \\
\text{where} \qquad & \qquad p(\dot{s}_a, \dot{s}_b) = \dot{s}' \\
& \qquad (\dot{a}_i, \dot{b}_i) = \dot{e}'_i \\
& \qquad \pi(\nu_a, \dot{s}_a) = \langle \dot{a}_i \rangle,\ \nu_a(i) = c_1 \\
& \qquad \pi(\nu_b, \dot{s}_b) = \langle \dot{b}_i \rangle,\ \nu_b(i) = c_2 \\
& \qquad \pi(\nu', \dot{s}') = \langle\langle \dot{e}'_i \rangle\rangle,\ \nu'(i) = 1
\end{aligned}
$$

The *zipWithU* Process Constructor

$$
\begin{aligned}
\texttt{zipWithU}(c_1, c_2, f) \quad &= \quad p \\
\text{where} \qquad & p(\dot{s}_a, \dot{s}_b) = \dot{s}' \\
& f((\dot{a}_i, \dot{b}_i)) = \dot{c}_i \\
& \pi(\nu_a, \dot{s}_a) = \langle \dot{a}_i \rangle,\ \nu_a(i) = c_1 \\
& \pi(\nu_b, \dot{s}_b) = \langle \dot{b}_i \rangle,\ \nu_b(i) = c_2 \\
& \pi(\nu', \dot{s}') = \langle \dot{c}_i \rangle,\ \nu'(i) = \#\dot{c}_i
\end{aligned}
$$

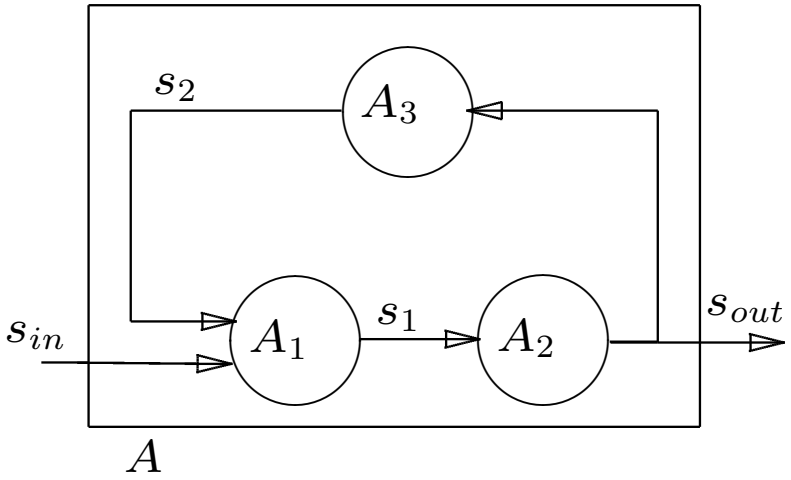
A Zip-based Process



$$A_1 = \texttt{zipUs}(1, 5)$$

$$A_1(\langle 10, 11\rangle, \langle 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\rangle) \;=\; \langle(\langle 10\rangle, \langle 1, 2, 3, 4, 5\rangle), (\langle 11\rangle, \langle 6, 7, 8, 9, 10\rangle)\rangle$$

The Amplifier Process Network



$$A(s_{in}) = s_{out}$$

where

$$s_{out} = A_2(s_1)$$

$$s_1 = A_1(s_2, s_{in})$$

$$s_2 = A_3(s_{out})$$

Process Constructors to Initialize Signals

$$\begin{aligned} \texttt{scandU}(\gamma, g, w_0) &= p \\ \text{where} \quad & p(\dot{s}) = \langle w_0 \rangle \oplus \texttt{scanU}(\gamma, g, w_0)(\dot{s}) \end{aligned}$$

$$\begin{aligned} \texttt{initU}(\dot{r}) &= p \\ \text{where} \quad & p(\dot{s}) = \dot{r} \oplus \dot{s} \\ & \nu = \nu' = 1 \\ & \dot{r}, \dot{s} \in \dot{S} \end{aligned}$$

Sink and Source Processes

$$
\begin{aligned}
\texttt{sourceU}(g, w_0) \quad &= \quad p \\
\text{where} \quad & p() = \dot{s}' \\
& w_i = e'_i \\
& g(w_i) = w_{i+1} \\
& \pi(\nu', \dot{s}') = \langle\langle e'_i \rangle\rangle,\ \nu'(i) = \#g(\dot{a}_i) = 1
\end{aligned}
$$

$$
\begin{aligned}
\texttt{sinkU}(\gamma, g, w_0) \quad &= \quad p \\
\text{where} \quad & p(\dot{s}) = \langle\rangle \\
& g(w_i) = w_{i+1} \\
& \pi(\nu, \dot{s}) = \langle \dot{a}_i \rangle,\ \nu(i) = \gamma(w_i)
\end{aligned}
$$

The Amplifier with Signal Initialization

$$A_4 = \texttt{initU}(\langle 10 \rangle)$$

$$A'(s_{in}) = s_{out}$$

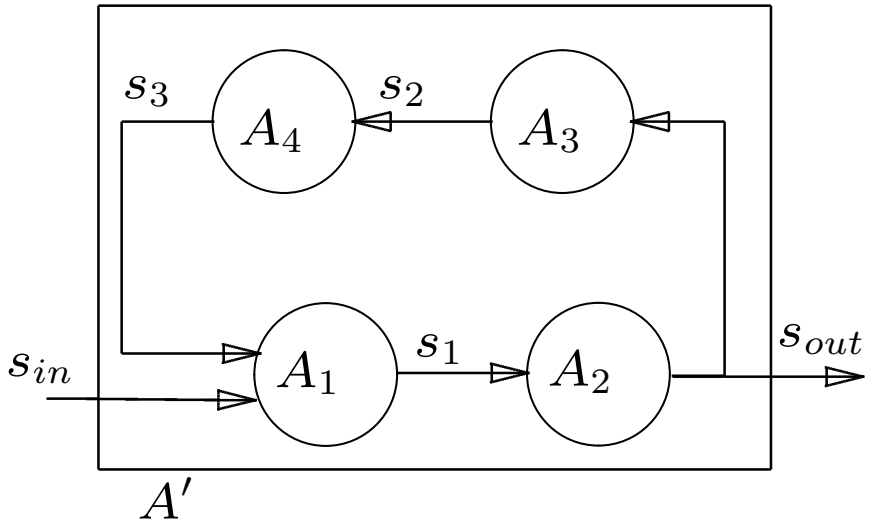
where

$$s_{out} = A_2(s_1)$$

$$s_1 = A_1(s_3, s_{in})$$

$$s_3 = A_4(s_2)$$

$$s_2 = A_3(s_{out})$$



$$s_{in} = \langle 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 18, 20 \rangle$$

$$s_{out} = \langle 10, 20, 30, 40, 50, 66, 77, 88, 99, 110,$$
$$121, 132, 143, 154, 165, 160, 170, 180, 190, 200 \rangle$$

$$s_3 = \langle 10, 11, 11, 10, 9 \rangle.$$

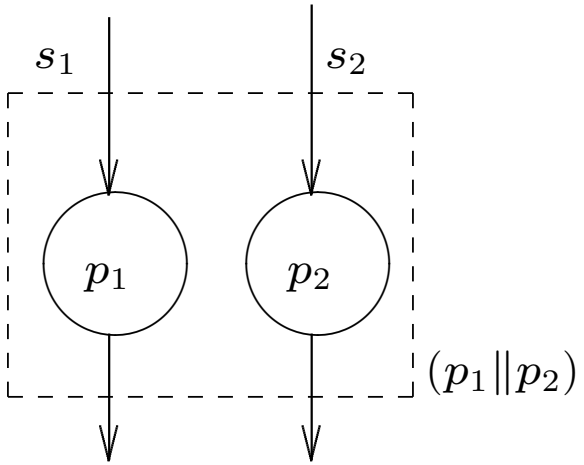
Process Composition

- Parallel Composition
- Sequential Composition
- Feedback Composition

Parallel Process Composition

p_1 and p_2 are two processes with one input and one output each.
$s_1, s_2 \in S$ are two signals.
Parallel process composition, denoted as $p_1 \parallel p_2$, is defined by

$$(p_1 \parallel p_2)(\langle s_1, s_2 \rangle) = \langle p_1(s_1), p_2(s_2) \rangle.$$



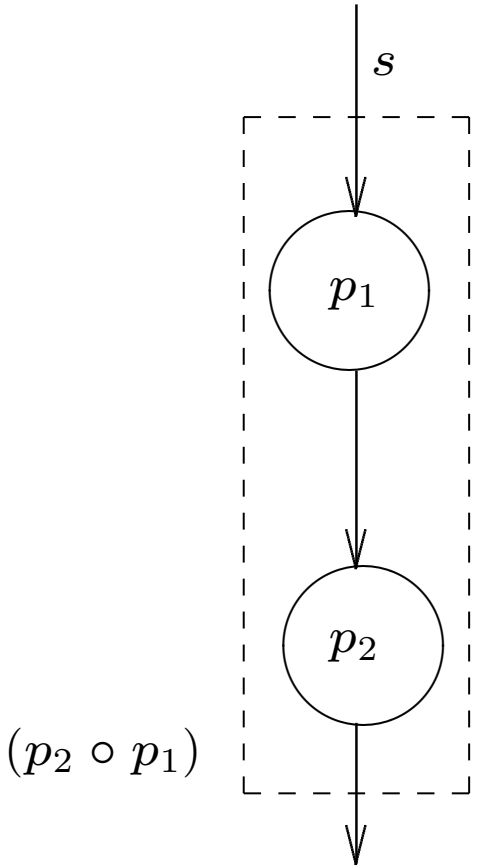
Example: $p(s_1, s_2) = (s_1, s_2)$ can be defined as

$$\begin{array}{lrcl} p(s_1, s2) & = & (p'(s1), p'(s2)) \\ \text{where} & p' & = & \texttt{mapU}(1, f) \\ & f(x) & = & x \end{array}$$

Sequential Process Composition

p_1 and p_2 are two processes with one input and one output each.
$s \in S$ is a signal.
Sequential process composition, denoted as $p_1 \circ p_2$, is defined by

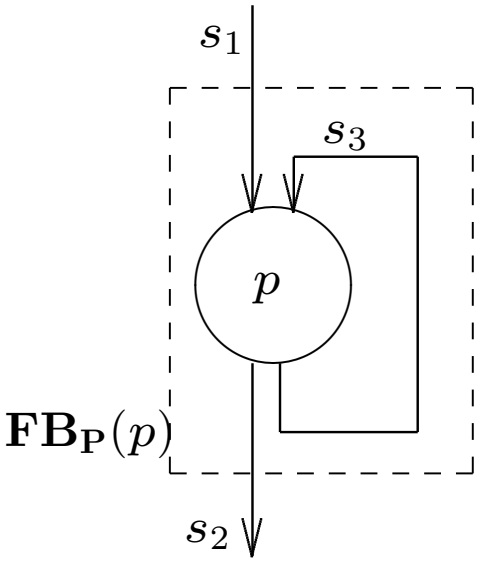
$$(p_2 \circ p_1)(s) = p_2(p_1(s)).$$



Feedback Process Composition

$p : (S \times S) \to (S \times S)$ is a process with two input signals and two output signals.
The operator $\mathbf{FB_P}$ is called the feedback operator. The process $\mathbf{FB_P}(p) : S \to S$ is defined by

$\mathbf{FB_P}(p)(s_1) = s_2$ where $p(s_1, s_3) = (s_2, s_3)$.



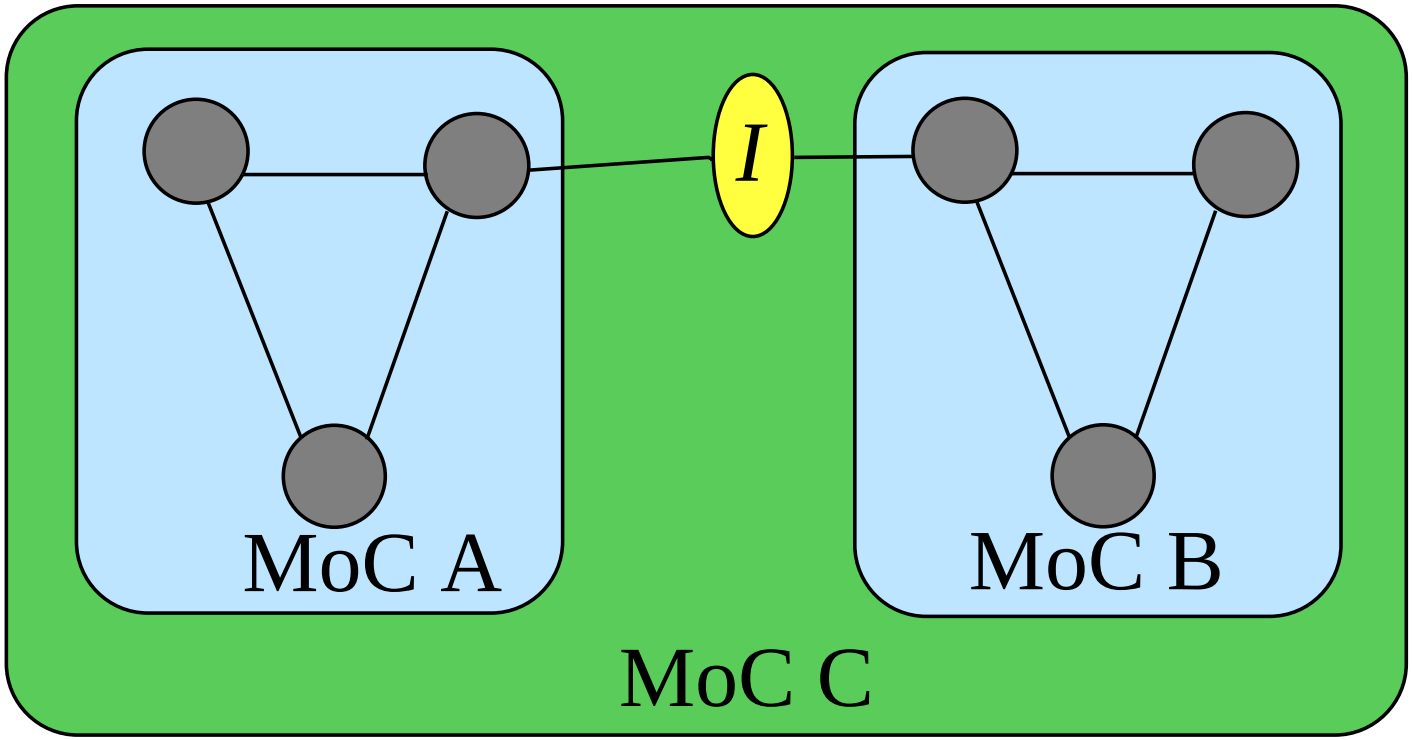
Models of Computation

Definition: A Model of Computation (MoC) is a 2-tuple MoC$= (C, O)$, where

C is a set of process constructors, each of which, when given constructor specific parameters, instantiates a process.

O is a set of process composition operators, each of which, when given processes as arguments, instantiates a new process.

Domains of different Models of Computation



The Untimed Model of Computation

Definition: The Untimed Model of Computation (Untimed MoC) is defined as Untimed MoC$=(C, O)$, where

$$C = \{\, \texttt{mapU}, \texttt{scanU}, \texttt{scandU}, \texttt{mealyU}, \texttt{mooreU},$$
$$\texttt{zipU}, \texttt{zipUs}, \texttt{zipWithU}, \texttt{unzipU},$$
$$\texttt{sourceU}, \texttt{sinkU}, \texttt{initU}\}$$
$$O = \{\|, \circ, \mathbf{FB_P}\}$$

In other words, a process or a process network belongs to the Untimed MoC Domain iff all its processes and process compositions are constructed either by one of the named process constructors or by one of the composition operators. We call such processes U-MoC processes.

Process Signature

Definition: The type of a process is a four-tuple $\langle TI, TO, NI, NO \rangle$, where

$TI = \{T_{I,1}, ..., T_{I,n}\}$	is the set of types of the n input signals
$TO = \{T_{O,1}, ..., T_{O,m}\}$	is the set of types of the m output signals
$NI = \{\nu i_1, ..., \nu i_n\}$	is the set of partitioning functions for the n input signals
$NO = \{\nu o_1, ..., \nu o_m\}$	is the set of partitioning functions for the m output signals

The pair $\langle NI, NO \rangle$ is the process signature.

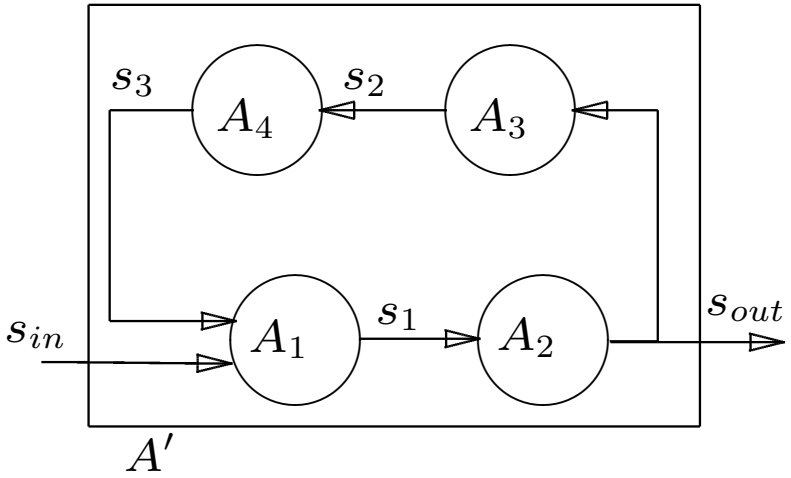
Process Matches

Process p has one output with type $T_{O,p}$ and partitioning ν'_p.
Process q has one input with type $T_{I,q} = TT_{O,q}$ and partitioning ν_q.
The match of the two processes is defined by

$$match(p,q) = \frac{\nu'_p(i)}{\nu_q(i)}.$$

$match(p,q) =$ constant 1 ... perfect match
$match(p,q) =$ constant rational number ... rational match
$match(p,q) =$ not constant ... varying match

Matches of the Amplifier Processes



$TYPE(A_1) = \langle TI, TO, NI, NO \rangle$ with $TI = \{\mathbb{Z}, \mathbb{Z}\}, TO = \{\mathbb{Z}\}, NI = \{1, 5\}, NO = \{1\}$.
$TYPE(A_2) = \langle TI, TO, NI, NO \rangle$ with $TI = \{\mathbb{Z}\}, TO = \{\mathbb{Z}\}, NI = \{1\}, NO = \{5\}$.
$TYPE(A_3) = \langle TI, TO, NI, NO \rangle$ with $TI = \{\mathbb{Z}\}, TO = \{\mathbb{Z}\}, NI = \{5\}, NO = \{1\}$.
$TYPE(A_4) = \langle TI, TO, NI, NO \rangle$ with $TI = \{\mathbb{Z}\}, TO = \{\mathbb{Z}\}, NI = \{1\}, NO = \{1\}$.

Process Up-rating

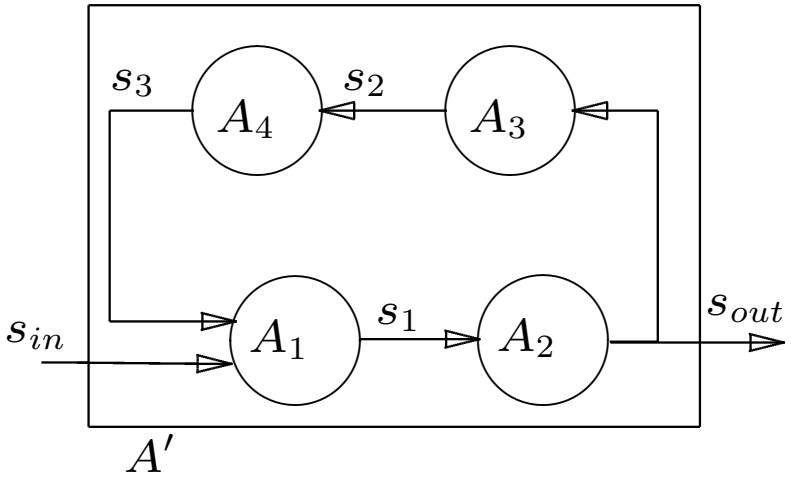
Definition: Let ϱ be a natural number $\varrho > 0$ and let p be a U-MoC process with one input and one output and the input signal s is partitioned $\pi(\nu_p, s) = \langle a_i \rangle$. Process p is up-rated by a factor ϱ resulting in another process q if q is continuous and the input signal partitioning of q is $\pi(\nu_q, s) = \langle b_j \rangle$ with

$$b_j = \bigoplus_{i=0}^{\varrho-1} a_{j\varrho+i} \qquad \forall j \in \mathbb{N}_0$$

and p behaves identical to q for all increasing prefixes of an input signal defined by b_j:

$$p(\bigoplus_{i=0}^{j-1} b_i) = q(\bigoplus_{i=0}^{j-1} b_i) \qquad \forall j \in \mathbb{N}_0.$$

Up-rating of Map Processes

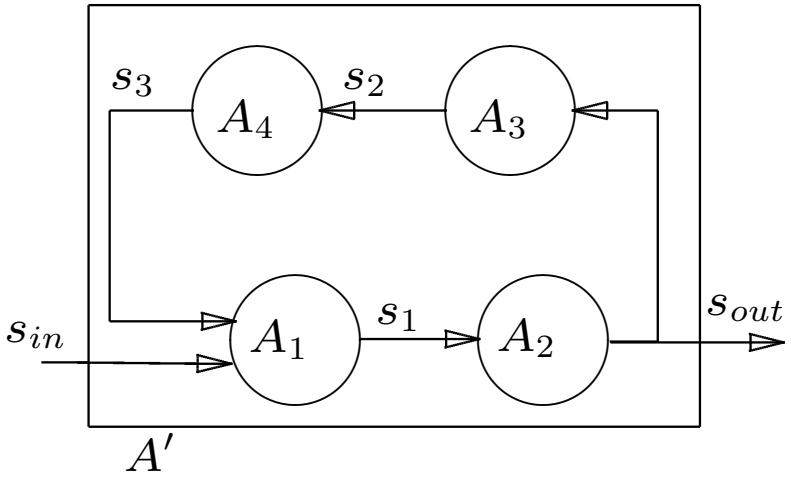


$$A_2' = \mathsf{uprate}(A_2, 2)$$

$$A_2 = \mathtt{mapU}(1, f)$$
where $f((\langle x \rangle, \langle y_1, y_2, y_3, y_4, y_5 \rangle)) = \langle xy_1, xy_2, xy_3, xy_4, xy_5 \rangle$

$$A_2' = \mathtt{mapU}(2, f')$$
where $f'(\langle x_1, x_2 \rangle) = f(x_1) \oplus f(x_2)$

Up-rating of Scan Processes



$$A_3 = \texttt{scanU}(\gamma, g, w_0)$$

where

$$\dot{s}' = A_3(\dot{s})$$

$$w_0 = 10$$

$$\gamma(w_i) = 5 \ \forall i \in \mathbb{N}_0$$

$$\pi(5, \dot{s}) = \langle \dot{a}_i \rangle \text{ and } \pi(1, \dot{s}') = \langle \dot{a}'_i \rangle$$

$$g(\langle x_1, x_2, x_3, x_4, x_5 \rangle, w_i) = \begin{cases} w_i - 1 & \text{if } x_1 + x_2 + x3 + x4 + x5 > 500 \\ w_i + 1 & \text{if } x_1 + x_2 + x3 + x4 + x5 < 400 \\ w_i & \text{otherwise} \end{cases}$$

Up-rating of Scan Processes - cont'd

$$A'_3 = \mathsf{uprate}(A_3, 2)$$

$$A'_3 = \texttt{mealyU}(2 \cdot 5, g', f', v_0)$$

where

$$\begin{aligned}
v_0 &= w_0 = 10 \\
\pi(10, \dot{s}) &= \langle u_i \rangle, u_i = \dot{a}_{2i} \oplus \dot{a}_{2i+1} \\
\pi(2, \dot{s}') &= \langle u'_i \rangle, u'_i = \dot{a}'_{2i} \oplus \dot{a}'_{2i+1} \\
f'(u_i, v_i) &= u'_i = g(\dot{a}_{2i}, v_i) \oplus g(\dot{a}_{2i+1}, g(\dot{a}_{2i}, v_i)) \\
g'(u_i, v_i) &= v_{i+1} = g(\dot{a}_{2i+1}, g(\dot{a}_{2i}, v_i))
\end{aligned}$$

$$\begin{aligned}
&A'_3(\langle 10{,}20, 30, 40, 50, 60, 70, 80, 90, 100 \rangle) \\
&\quad = f'(u_0, v_0) = f'(\langle 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 \rangle, 10) \\
&\quad = g(\langle 10, 20, 30, 40, 50 \rangle, 10) \oplus g(\langle 60, 70, 80, 90, 100 \rangle, g(\langle 10, 20, 30, 40, 50 \rangle, 10)) \\
&\quad = 11 \oplus g(\langle 60, 70, 80, 90, 100 \rangle, 11) = \langle 11, 11 \rangle.
\end{aligned}$$

Up-rating of Mealy Processes

$$
\begin{array}{rcl}
B & = & \texttt{mealyU}(\gamma, g, f, w_0) \\
\text{where} & & w_0 = \langle 0,0,0,0,0 \rangle \\
& & \gamma(w_i) = 2 \; \forall i \in \mathbb{N}_0 \\
& & g(\langle v_1, v_2, v_3, v_4, v_5 \rangle, \langle x_1, x_2 \rangle) = \langle v_3, v_4, v_5, x_1, x_2 \rangle \\
& & f(\langle v_1, v_2, v_3, v_4, v_5 \rangle, \langle x_1, x_2 \rangle) = \langle v_1 + v_2 + v_3 + v_4 + v_5 \rangle \\
\\
B' & = & \mathsf{uprate}(B, 2) \\
\\
B' & = & \texttt{mealyU}(2 \cdot 2, g', f', v_0) \\
v_0 & = & w_0 = \langle 0,0,0,0,0 \rangle \\
u_i & = & \dot{a}_{2i} \oplus \dot{a}_{2i+1} \\
u'_i & = & \dot{a}'_{2i} \oplus \dot{a}'_{2i+1} \\
f'(u_i, v_i) & = & f(\dot{a}_{2i}, w_{2i}) \oplus f(\dot{a}_{2i+1}, w_{2i+1}) \\
g'(u_i, v_i) & = & g(\dot{a}_{2i+1}, w_{2i+1}) \\
& & \text{with } w_{2i} = v_i \\
& & \text{and } w_{2i+1} = g(\dot{a}_{2i}, w_{2i}) \text{ for } i \in \mathbb{N}_0
\end{array}
$$

Up-rating of Mealy Processes - cont'd

For $\dot{s} = \langle 2, 4, 6, 8 \rangle$ we get

$$
\begin{array}{lcl}
B(\dot{s}) & = & \langle f(\langle 2,4\rangle, \langle 0,0,0,0,0\rangle), f(\langle 6,8\rangle, g(\langle 2,4\rangle, \langle 0,0,0,0,0\rangle))\rangle \\
 & = & \langle 0, 6\rangle \\
B'(\dot{s}) & = & \langle f'(u_0, v_0)\rangle \\
 & = & \langle f'(\langle 2,4,6,8\rangle, \langle 0,0,0,0,0\rangle)\rangle \\
 & = & f(\langle 2,4\rangle, \langle 0,0,0,0,0\rangle) \oplus f(\langle 6,8\rangle, g(\langle 2,4\rangle, \langle 0,0,0,0,0\rangle)) \\
 & = & \langle 0, 6\rangle
\end{array}
$$

Up-rating of Processes with Multiple Inputs

Definition: p is a U-MoC process with n inputs and m outputs.

The input signals $s_l, 1 \leq l \leq n$ are partitioned $\pi(\nu_{p,l}, s_l) = \langle a_{i,l} \rangle$.

Process p is up-rated by a factor $\varrho \in \mathbb{N}$ resulting in q if the input signal partitioning of q is $\pi(\nu_{q,l}, s_l) = \langle b_{j,l} \rangle$ with

$$b_{j,l} = \bigoplus_{i=0}^{\varrho-1} a_{j\varrho+i,l} \qquad \forall j \in \mathbb{N}_0, 1 \leq l \leq n$$

and p behaves identical to q for all increasing prefixes of input signals defined by $b_{j,l}$:

$$p((\bigoplus_{i=0}^{j-1} b_{i,1}), \cdots, (\bigoplus_{i=0}^{j-1} b_{i,n})) = q((\bigoplus_{i=0}^{j-1} b_{i,1}), \cdots, (\bigoplus_{i=0}^{j-1} b_{i,n})) \qquad \forall j \in \mathbb{N}_0.$$

Up-rating of Zip and Unzip Processes

- *zipU* processes cannot be up-rated in a simple way.
- $p = \textit{zipUs}(c_1, c_2)$, $q = \text{uprate}(p, \varrho)$ is defined by

$$\begin{aligned} q &= q_2 \circ q_1 \\ q_1 &= \textit{zipUs}(\varrho c_1, \varrho c_2) \\ q_2 &= \textit{mapU}(1, f) \end{aligned}$$

$$f((\dot{a}_1, \dot{a}_2)) = \begin{cases} \langle\rangle & \text{if } \#(\dot{a}_1) < c_1 \vee \#(\dot{a}_2) < c_2 \\ \langle(\text{take}(c_1, \dot{a}_1), \text{take}(c_2, \dot{a}_2))\rangle & \\ \oplus f(\text{drop}(c_1, \dot{a}_1), \text{drop}(c_2, \dot{a}_2)) & \text{otherwise} \end{cases}$$

- $p = \textit{unzipU}()$, $q = \text{uprate}(p, \varrho)$ is defined by

$$\begin{aligned} q &= q_2 \circ q_1 \\ q_1 &= \textit{mapU}(\varrho, f) \\ f(\langle(\dot{a}_1, \dot{b}_1), \cdots, (\dot{a}_\varrho, \dot{b}_\varrho)\rangle) &= (\dot{a}_1 \oplus \dot{a}_2 \oplus \cdots \oplus \dot{a}_\varrho, \dot{b}_1 \oplus \dot{b}_2 \oplus \cdots \oplus \dot{b}_\varrho) \\ q_2 &= \textit{unzipU}() \end{aligned}$$

Up-rating and Process Composition

- Sequential composition:

$$\text{uprate}((p \circ q), \varrho) = (\text{uprate}(p, \varrho)) \circ (\text{uprate}(q, \varrho))$$

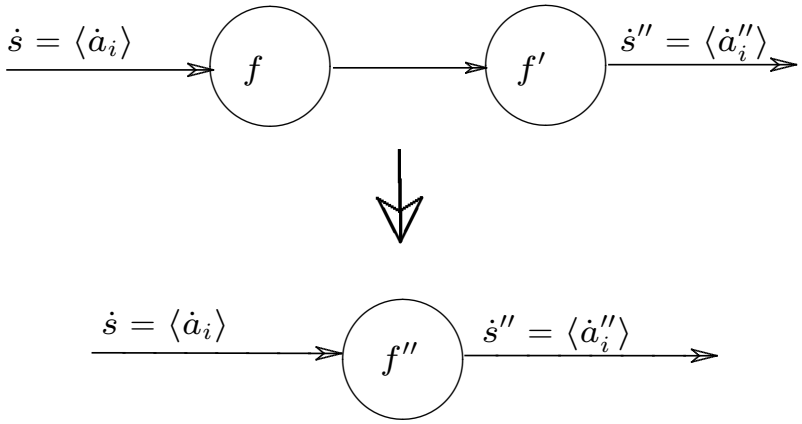
- Parallel composition:

$$\text{uprate}((p \parallel q), \varrho) = (\text{uprate}(p, \varrho)) \parallel (\text{uprate}(q, \varrho))$$

- Feedback composition:

$$\text{uprate}((\mathbf{FB_P}(p)), \varrho) \neq \mathbf{FB_P}(\text{uprate}(p, \varrho))$$

Merge of Map Processes

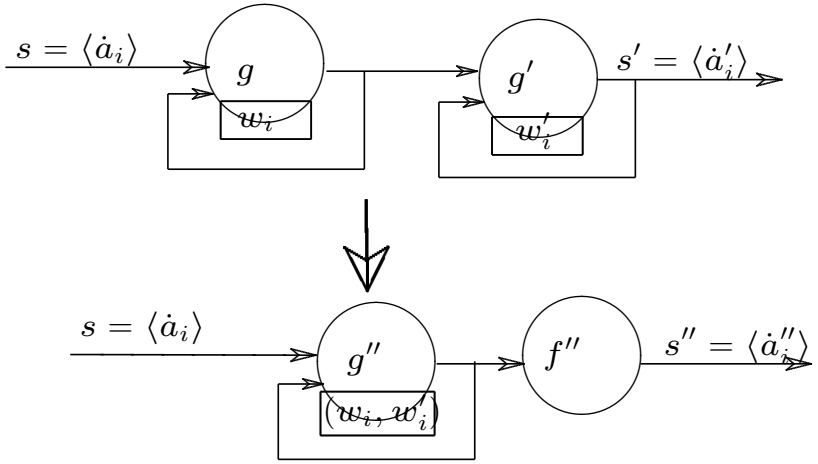


$$
\begin{array}{rcl}
p & = & \texttt{mapU}(c, f) \\
p' & = & \texttt{mapU}(c', f') \\
p'' & = & \texttt{mapU}(c'', f'')
\end{array}
$$

where

$$
\begin{array}{rcl}
f''(\dot{a}_i) & = & \dot{a}''_i = f'(f(\dot{a}_i)) \\
c'' & = & c \\
\nu'_{p''} & = & \nu'_{p'}
\end{array}
$$

Merge of Scan Processes

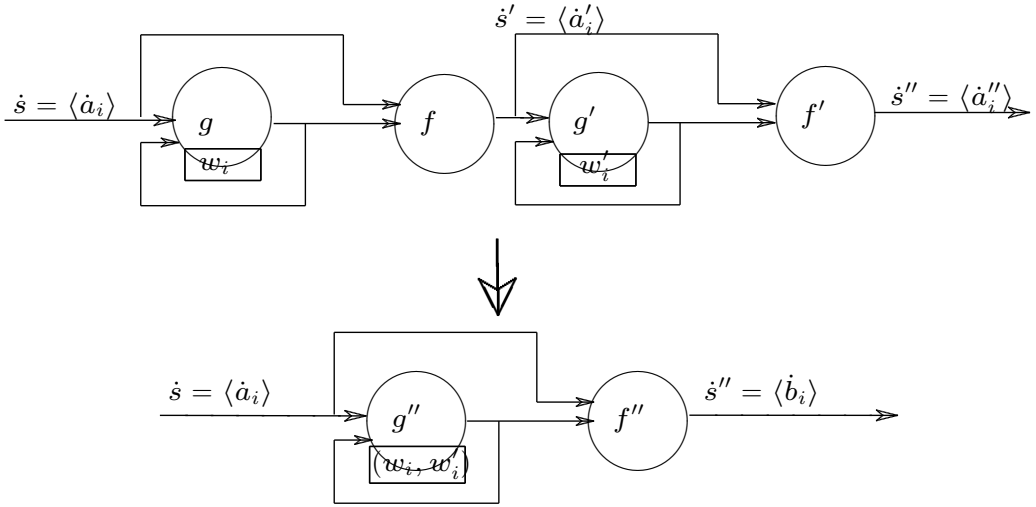


$$
\begin{array}{rcl}
p & = & \texttt{scanU}(\gamma, g, w_0) \\
p' & = & \texttt{scanU}(\gamma', g', w'_0) \\
p'' & = & \texttt{mealyU}(\gamma'', g'', f'', v''_0)
\end{array}
$$

where

$$
\begin{array}{rcl}
\gamma''(v''_i) & = & \gamma(w_i) \\
v''_0 & = & (v_0, v'_0) = (w_0, w'_0) \\
g''((v_i, v'_i)), \dot{a}_i) & = & (v_{i+1}, v'_{i+1}) = (g(v_i, \dot{a}_i), g'(v'_i, v_i)) \\
f''((v_i, v'_i), \dot{a}_i) & = & v'_i
\end{array}
$$

Merge of Mealy Processes

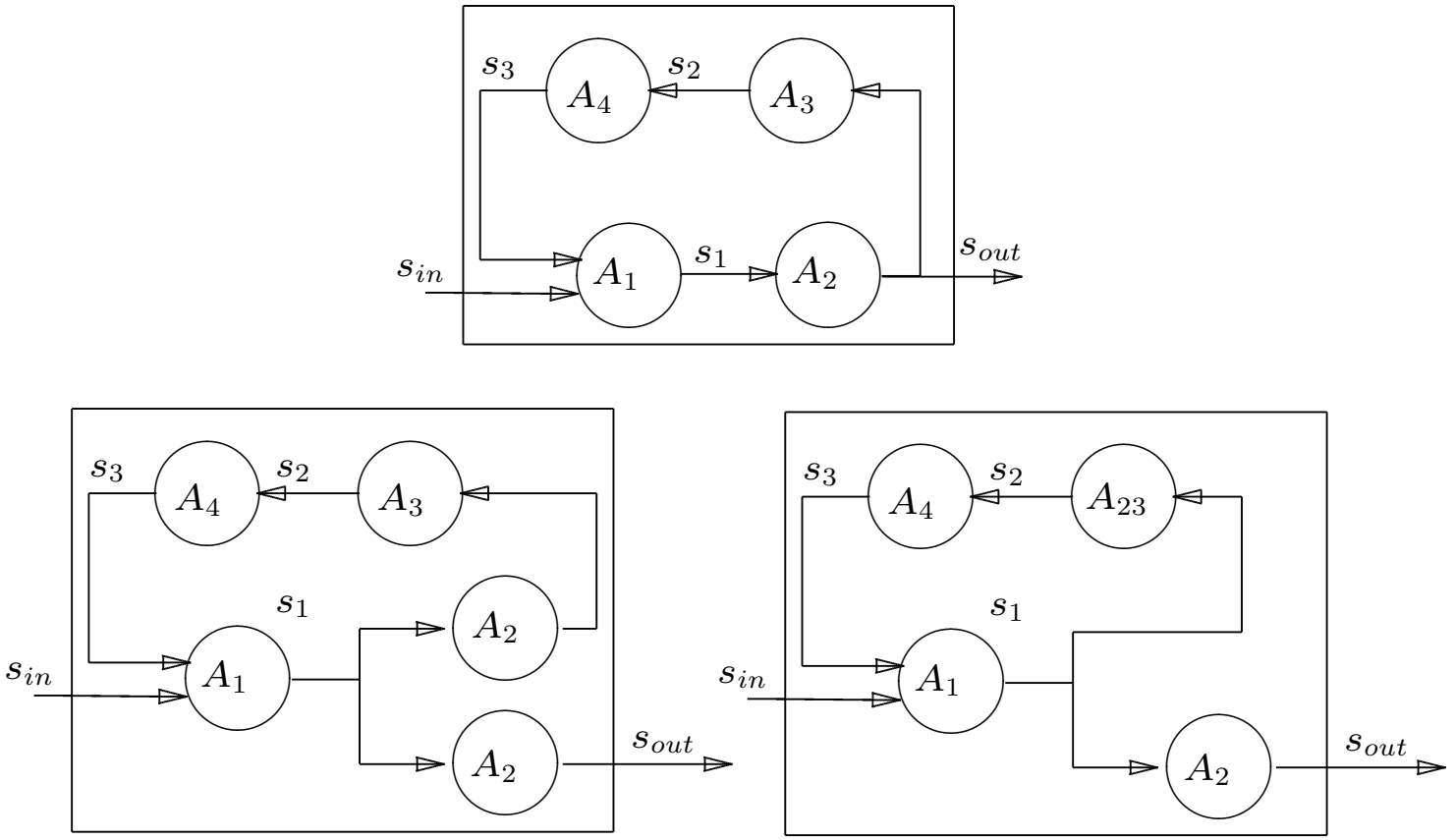


$$
\begin{array}{rcl}
p & = & \texttt{mealyU}(\gamma, g, f, w_0) \\
p' & = & \texttt{mealyU}(\gamma', g', f', w'_0) \\
p'' & = & \texttt{mealyU}(\gamma'', g'', f'', v''_0)
\end{array}
$$

where

$$
\begin{array}{rcl}
v''_0 & = & (v_0, v'_0) = (w_0, w'_0) \\
g''((v_i, v'_i), \dot{a}_i) & = & (v_{i+1}, v'_{i+1}) = (g(v_i, \dot{a}_i), g'(v'_i, f(v_i, \dot{a}_i))) \\
f''((v_i, v'_i), \dot{a}_i) & = & \dot{a}''_i = f'(v'_i, f(v_i, \dot{a}_i)) \\
\gamma''(v''_i) & = & \gamma(w_i)
\end{array}
$$

Process Merge Example



Process Merge Example - cont'd

$$A_2' = \texttt{mealyU}(\gamma_2, g_2, f_2, 0)$$

where

$$\gamma_2 = 1$$

$$f_2((\langle x\rangle, \langle y_1, y_2, y_3, y_4, y_5\rangle), 0) = \langle xy_1, xy_2, xy_3, xy_4, xy_5\rangle$$

$$g_2(x, 0) = 0$$

$$A_3' = \texttt{mealyU}(\gamma_3, g_3, f_3, w_0)$$

where

$$w_0 = 10$$

$$\gamma_3 = 5$$

$$f_3(x, w_i) = w_i$$

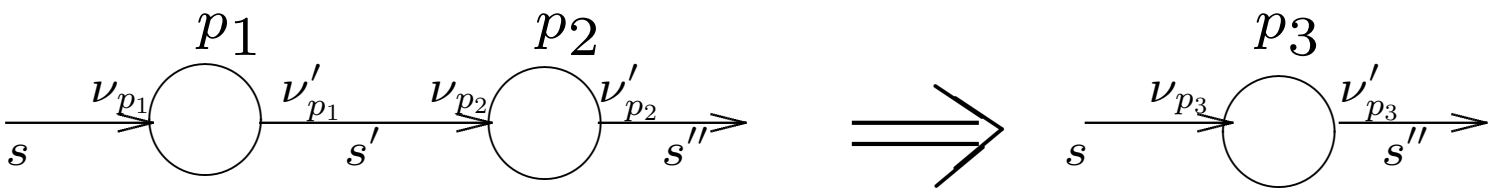
$$g_3(\langle x_1, x_2, x_3, x_4, x_5\rangle, w_i) = \begin{cases} w_i - 1 & \text{if } x_1 + x_2 + x_3 + x_4 + x_5 > 500 \\ w_i + 1 & \text{if } x_1 + x_2 + x_3 + x_4 + x_5 < 400 \\ w_i & \text{otherwise} \end{cases}$$

Process Merge Example - cont'd

$$
\begin{array}{rcl}
A_{23} & = & \texttt{mealyU}(\gamma, f, g, (0, w_0)) \\
\text{where} & & \\
\gamma((0, w_i)) & = & 1 \; \forall i \in \mathbb{N}_0 \\
g(\dot{a}_i, (0, w_i)) & = & (0, g_3(f_2(\dot{a}_i, 0), w_i)) \\
f(\dot{a}_i, (0, w_i)) & = & f_3(f_2(\dot{a}_i, 0), w_i)
\end{array}
$$

$$
\begin{array}{rcl}
A'_{23} & = & \texttt{mealyU}(\gamma', f', g', w_0) \\
\text{where} & & \\
\gamma(w_i) & = & 1 \; \forall i \in \mathbb{N}_0 \\
g'((\langle x \rangle, \langle y_1, y_2, y_3, y_4, y_5 \rangle), 0) = & & \\
& = & \begin{cases} w_i - 1 & \text{if } y_1 + y_2 + y_3 + y_4 + y_5 > 500/x \\ w_i + 1 & \text{if } y_1 + y_2 + y_3 + y_4 + y_5 < 400/x \\ w_i & \text{otherwise} \end{cases} \\
f'(x, w_i) & = & w_i
\end{array}
$$

Merge of Processes with a Rational Match



- Processes p_1 and p_2 for a rational match

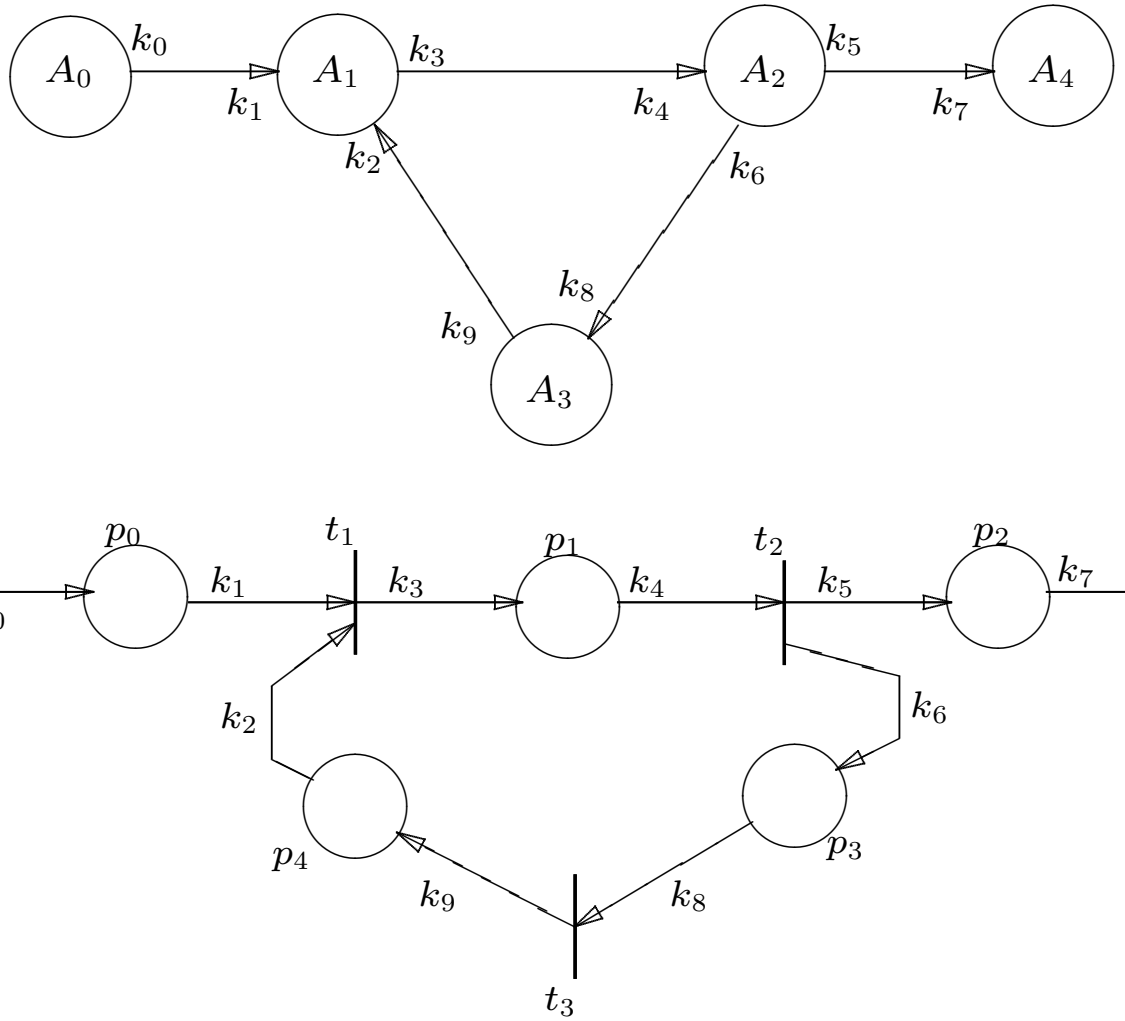
$$\text{match}(p_1, p_2) = \frac{\nu'_{p_1}(i)}{\nu_{p_2}(i)} = \frac{\varrho_2}{\varrho_1}$$

with $\varrho_2, \varrho_1 \in \mathbb{N}$ with no common integer divisor.

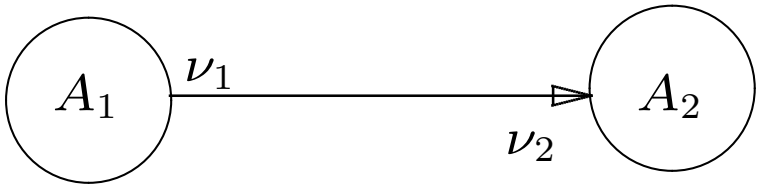
- Processes uprate(p_1, ϱ_1) and uprate(p_2, ϱ_2) form a perfect match.
- They can be merged: $p_3 = (\text{uprate}(p_2, \varrho_2)) \circ (\text{uprate}(p_1, \varrho_1))$ with

$$\nu_{p_3}(i) = \sum_{j=0}^{\varrho_1 - 1} \nu_{p_1}(i\varrho_1 + j) \qquad \nu'_{p_3}(i) = \sum_{j=0}^{\varrho_2 - 1} \nu'_{p_2}(i\varrho_2 + j).$$

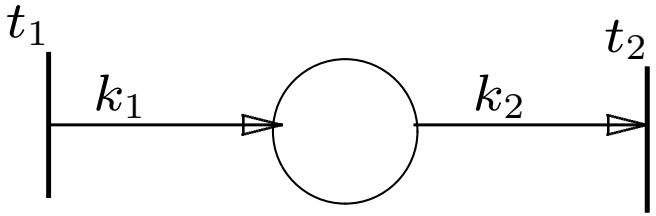
Untimed Process Networks and Petri Nets



Processes with Rational Matches as Petri Nets



(a)

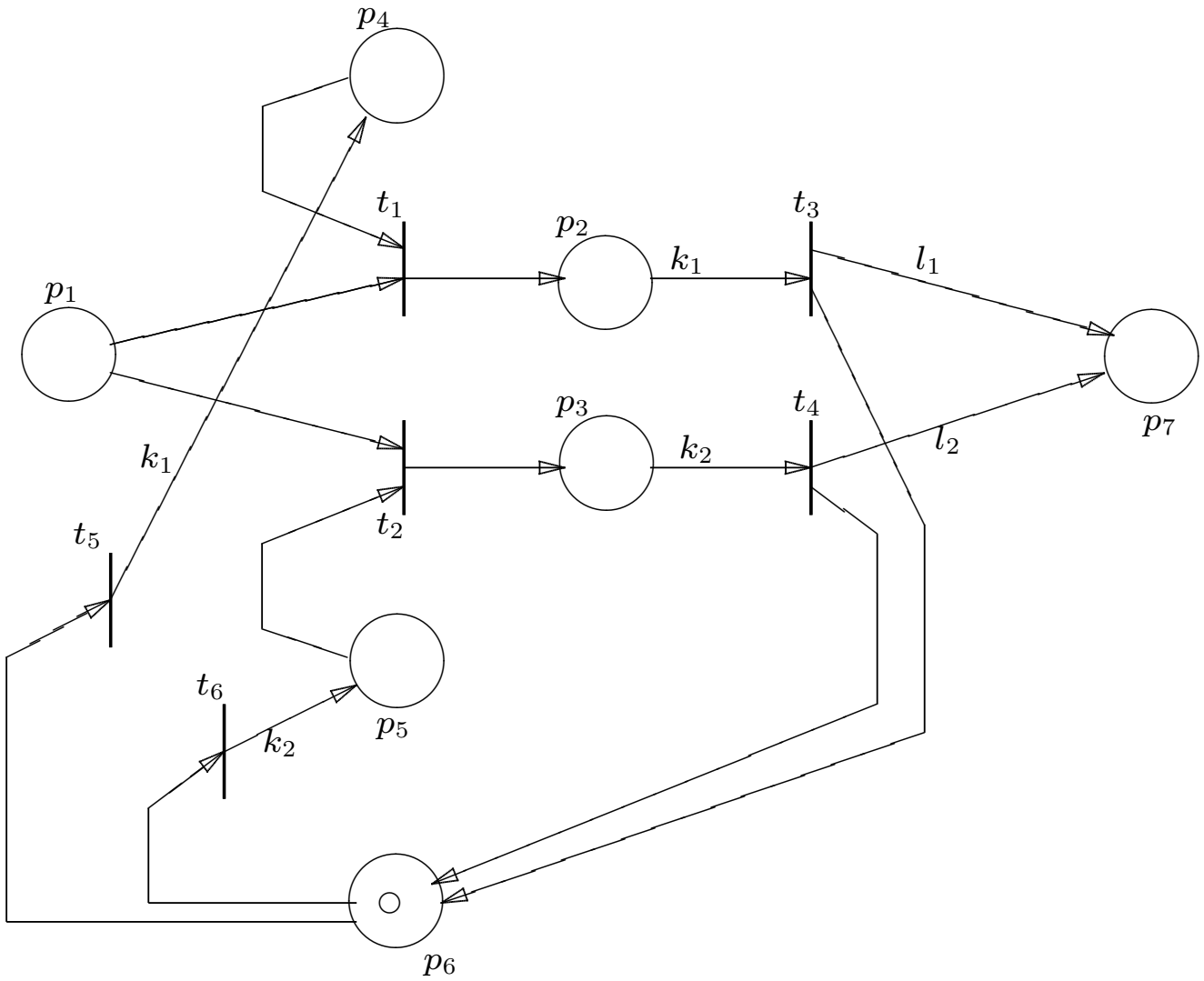


(b)

A process network in (a) can be represented by the Petri net in (b) if

$$\frac{\nu_1(i)}{\nu_2(i)} = \frac{k_1}{k_2} \text{ for all } i \in \mathbb{N}_0.$$

Processes with Data Dependent Partitioning as Petri Nets

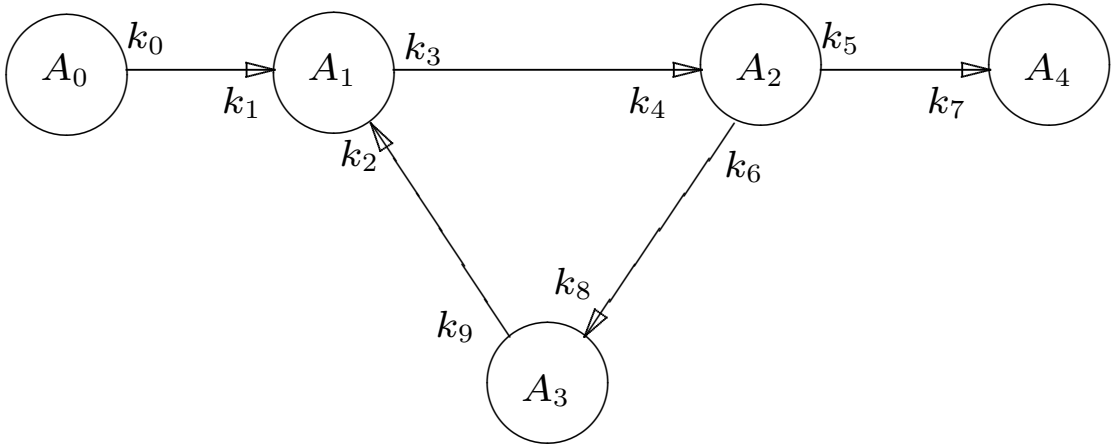


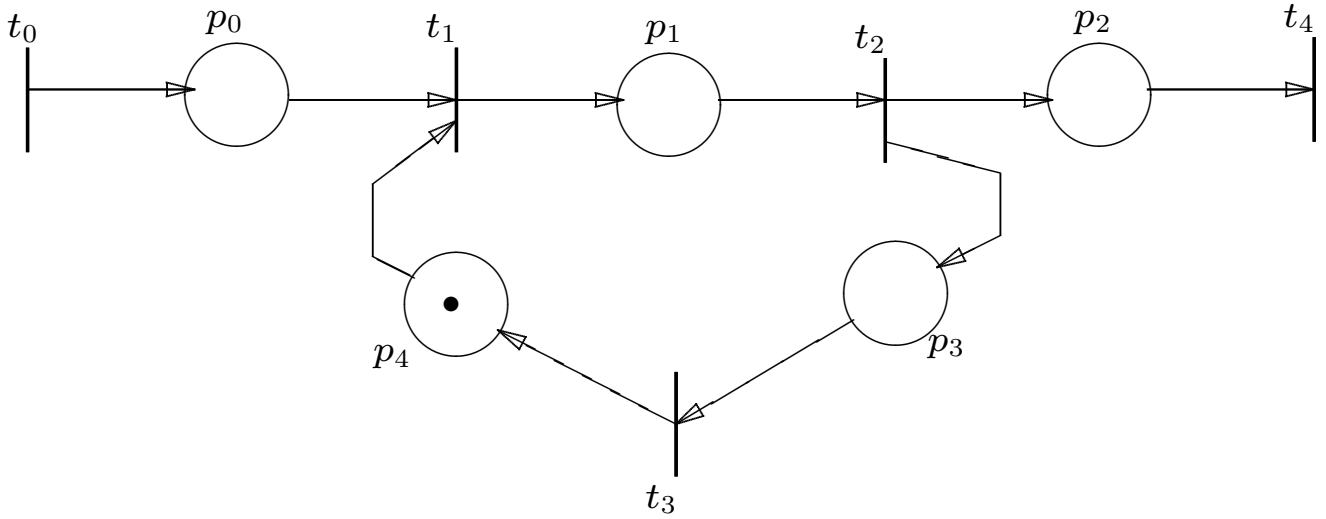
Synchronous Data Flow

Definition:

Synchronous Data Flow (SDF) is an untimed model of computation where all processes define only constant partitionings for all their input and output signals, i.e. all partitioning functions are constant; all process signatures are constant.

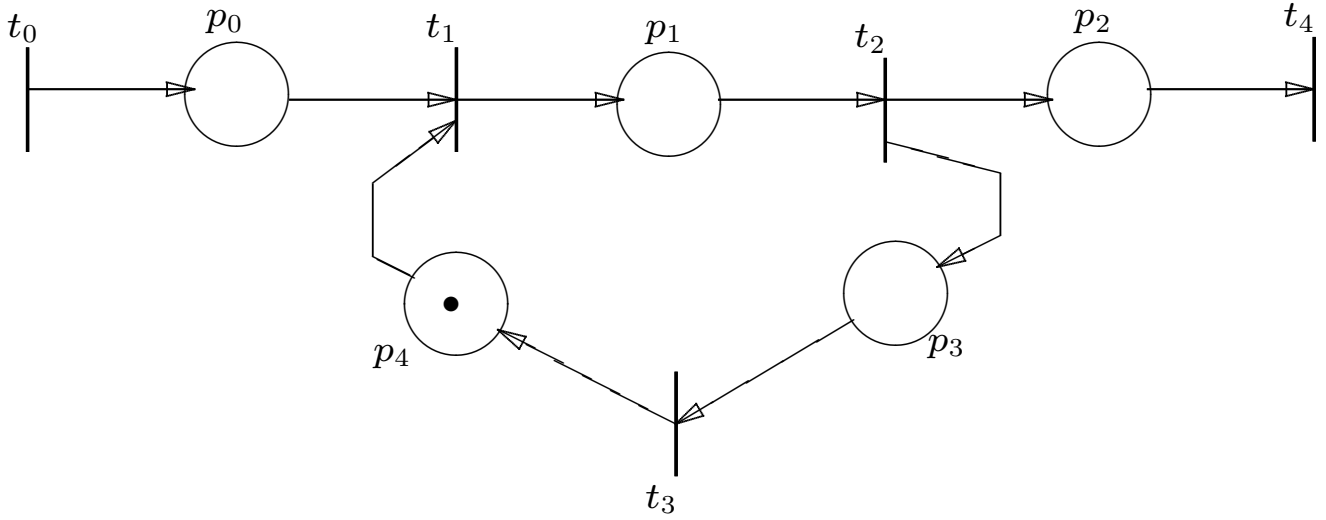
An SDF Network as Petri Net





$$k_i = 1 \ \forall i$$

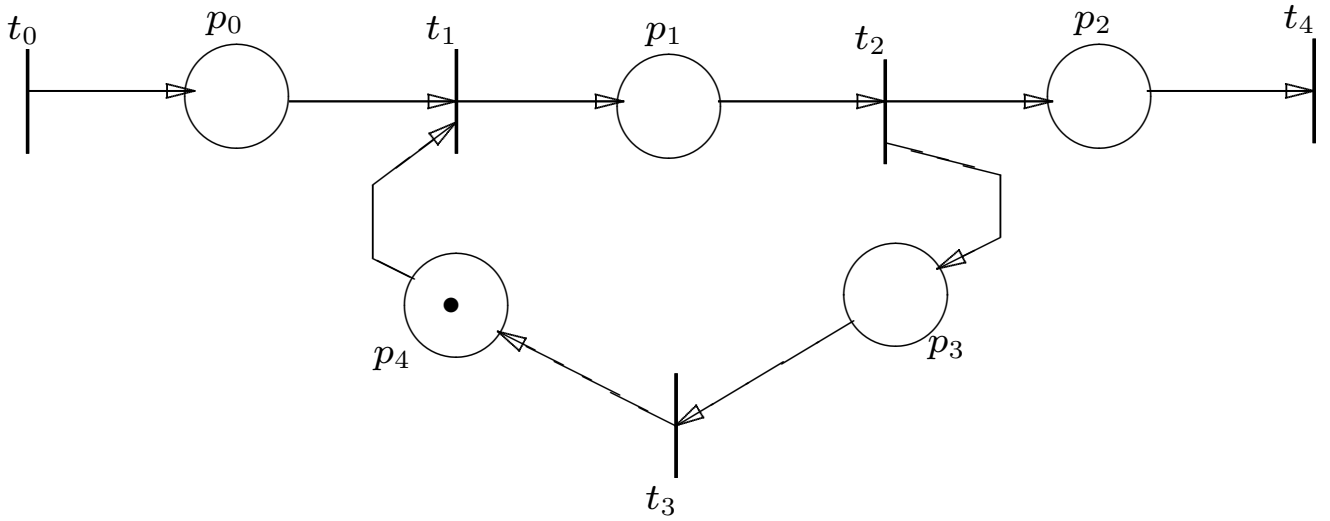
The Incidence Matrix of an SDF Network



$$\mathcal{A} = \begin{bmatrix} k_0 & 0 & 0 & 0 & 0 \\ -k_1 & k_3 & 0 & 0 & -k_2 \\ 0 & -k_4 & k_5 & k_6 & 0 \\ 0 & 0 & 0 & -k_8 & k_9 \\ 0 & 0 & -k_7 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

Rows correspond to transitions; columns to places.

Evolution of an SDF Network



$$\vec{x} = \vec{x}_0 + (\vec{u}_0 + \vec{u}_1 + \vec{u}_2 + \vec{u}_3 + \vec{u}_4)\,\boldsymbol{\mathcal{A}}$$

$$= [0,0,0,0,1] + [1,1,1,1,1] \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

$$= [0,0,0,0,1] + [0,0,0,0,0] = [0,0,0,0,1]$$

SDF Networks and Scheduling

- SDF represents an important class of applications;
- Important implementation problems are scheduling and buffer dimensioning;
- There exist necessary and sufficient conditions for the existence of periodic, static schedules;
- There exists an algorithm to construct a periodic, sequential schedule;
- There exists an algorithm to construct a periodic, parallel schedule.

Single Processor Schedule

Definition: s is a signal connecting the output of process A to the input of process B. The initial buffer condition of s is the number of events in s before A and B are executed the first time.

The number of events buffered in s is the number of events initially in s or produced by executions of A but not yet consumed by B.

An admissible, sequential schedule ϕ is a non-empty sequence of processes such that if the processes are executed in the sequence given by ϕ the number of events buffered in any signal will remain non-negative and bounded.

A periodic, admissible, sequential schedule (PASS) is a periodic and infinite admissible sequential schedule. It is specified by a list ϕ that is the list of processes executed in one period.

A Schedule as Transition Sequence

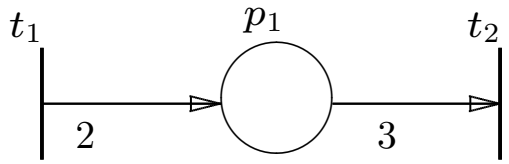
- A schedule ϕ is represented by $\phi = \langle t_i, t_k, ..., \rangle$.
- The firing of transition t_i is represented by vector $\vec{u}_i$.
- The firing vector $\vec{q}_\phi$ represents the schedule ϕ, i.e.

$$\vec{q}_\phi = \sum_{t_i \in \phi} \vec{u}_i.$$

- ϕ can only be a PASS if

$$\vec{q}_\phi \; \boldsymbol{\mathcal{A}} = \vec{0}$$

SDF Schedule Example 1



$$\boldsymbol{\mathcal{A}}_a = \begin{bmatrix} 2 \\ -3 \end{bmatrix}$$

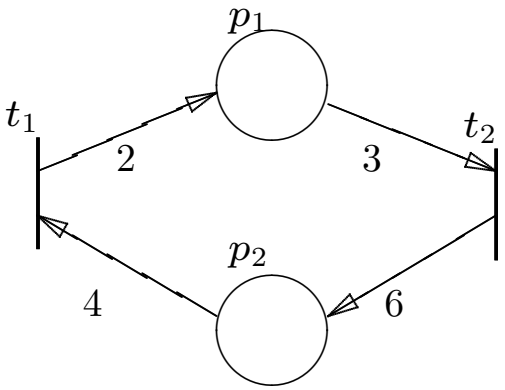
$$\vec{q}_\phi \, \boldsymbol{\mathcal{A}}_a = [q_1, q_2] \begin{bmatrix} 2 \\ -3 \end{bmatrix} = \vec{0} \quad \Rightarrow \quad 2q_1 - 3q_2 = 0$$

A periodic schedule requires an infinite number of solutions, which are represented by the smallest, strictly positive integer solutions for q_1 and q_2:

$$q_1 = 3 \text{ and } q_2 = 2$$

Possible schedules are:
$\phi = [t_1, t_1, t_1, t_2, t_2]$ or $\phi = [t_1, t_1, t_2, t_1, t_2]$.

SDF Schedule Example 2



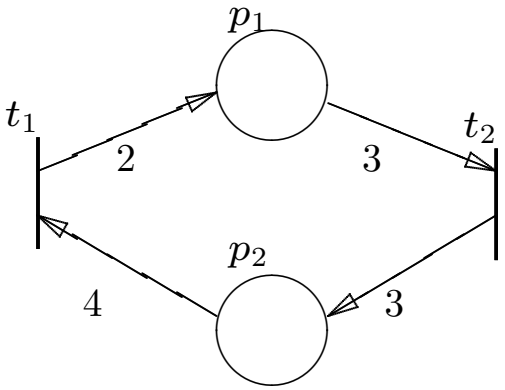
$$\boldsymbol{\mathcal{A}}_b = \begin{bmatrix} 2 & -4 \\ -3 & 6 \end{bmatrix}$$

$$\vec{q}_\phi \, \boldsymbol{\mathcal{A}}_b = [q_1, q_2] \begin{bmatrix} 2 & -4 \\ -3 & 6 \end{bmatrix} = \vec{0} \quad \Rightarrow \quad \begin{aligned} 2q_1 - 3q_2 &= 0 \\ -4q_1 + 6q_2 &= 0 \end{aligned}$$

These two equations are not independent and the smallest, strictly positive solution is

$$q_1 = 3 \quad \text{and} \quad q_2 = 2$$

SDF Schedule Example 3



$$\mathcal{A}_c = \begin{bmatrix} 2 & -4 \\ -3 & 3 \end{bmatrix}.$$

$$\vec{q}_\phi \, \mathcal{A}_c = [q_1, q_2] \begin{bmatrix} 2 & -4 \\ -3 & 3 \end{bmatrix} = \vec{0} \quad \Rightarrow \quad \begin{aligned} 2q_1 - 3q_2 &= 0 \\ -4q_1 + 3q_2 &= 0 \end{aligned}$$

The only solution is

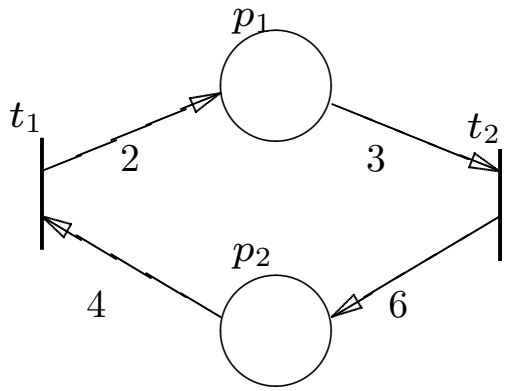
$$q_1 = 0 \quad \text{and} \quad q_2 = 0$$

The Rank Test

Theorem: For a connected SDF process network with N_t processes and its corresponding incidence matrix $\boldsymbol{\mathcal{A}}$, $\text{rank}(\boldsymbol{\mathcal{A}}) = N_t - 1$ is a necessary condition for a PASS to exist.

- The rank of a matrix gives the number of independent equations.
- If the rank would not be $N_t - 1$ any periodic schedule would accumulate tokens in one or more places.
- A positive rank test result guarantees the existence of a schedule with bounded token numbers in all places.
- A positive rank test does not guarantee that there are always sufficient tokens available for the next transition to fire.

Initial Buffer Conditions

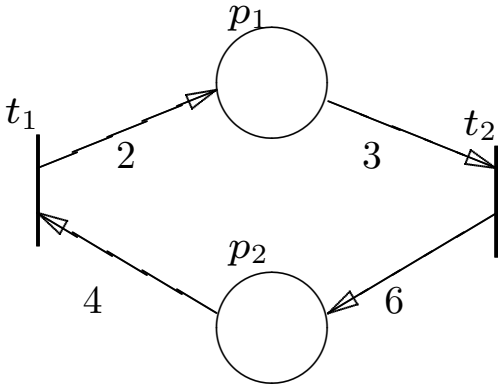


- The rank test is positive.
- $\vec{x}_0 = [0, 0]$: deadlock
- $\vec{x}_0 = [3, 0]$: deadlock after $\langle t_2, t_1 \rangle$
- $\vec{x}_0 = [6, 0]$: valid PASS: $\phi = [t_2, t_2, t_1, t_1, t_1, t_1]$

Computing the Initial Buffer Conditions

1. Find an arbitrary, positive, non-zero integer vector $\vec{q}$ such that $\vec{q}\,\boldsymbol{\mathcal{A}} = \vec{0}$.
   (a) There exists a vector $\vec{v} = [v_1, ..., v_{N_t}]$ with $\vec{v}\,\boldsymbol{\mathcal{A}} = \vec{0}$, $v_1 = 1$, all $v_i \geq 0, 1 \leq i \leq N_t$, and all v_i are rational numbers with their numerators and denominators are mutually prime.
   (b) Find $\eta \in \mathbb{N}_0$, the least common multiple of all denominators of v_i.
   (c) $\vec{q} = \eta\vec{v}$.
2. Select an arbitrary schedule ϕ with each transition firing as often as given by $\vec{q}$.
3. We execute one cycle of ϕ starting from an empty marking and allowing negative markings. We memorize the most negative marking of each place.
4. After the full cycle of ϕ the most negative marking, multiplied by -1, defines the initial buffer condition for each place.

Initial Buffer Conditions - Example



$$\mathcal{A}_b = \begin{bmatrix} 2 & -4 \\ -3 & 6 \end{bmatrix}$$

1. Find $\vec{q}$ such that $\vec{q}\,\mathcal{A} = \vec{0}$.
   (a) $2v_1 - 3v_2 = 0 \qquad -4v_1 + 6v_2 = 0$
   We set $v_1 = 1$ and derive $v_2 = 2/3$.
   (b) $\eta = 3$
   (c) $\eta\vec{v} = 3[1, 2/3] = [3, 2] = \vec{q}$
2. $\phi = \langle t_1, t_1, t_1, t_2, t_2 \rangle$
3. $\vec{x}_0 = [0, 0]$ $\qquad \vec{x}_3 = \vec{x}_2 + \vec{u}_1\,\mathcal{A} = [6, -12]$
   $\vec{x}_1 = \vec{x}_0 + \vec{u}_1\,\mathcal{A} = [2, -4]$ $\qquad \vec{x}_4 = \vec{x}_3 + \vec{u}_2\,\mathcal{A} = [3, -6]$
   $\vec{x}_2 = \vec{x}_1 + \vec{u}_1\,\mathcal{A} = [4, -8]$ $\qquad \vec{x}_5 = \vec{x}_4 + \vec{u}_2\,\mathcal{A} = [0, 0]$
4. Initial buffer conditions: 0 for p_1, 12 for p_2.

Multi Processor Schedule - Assumptions

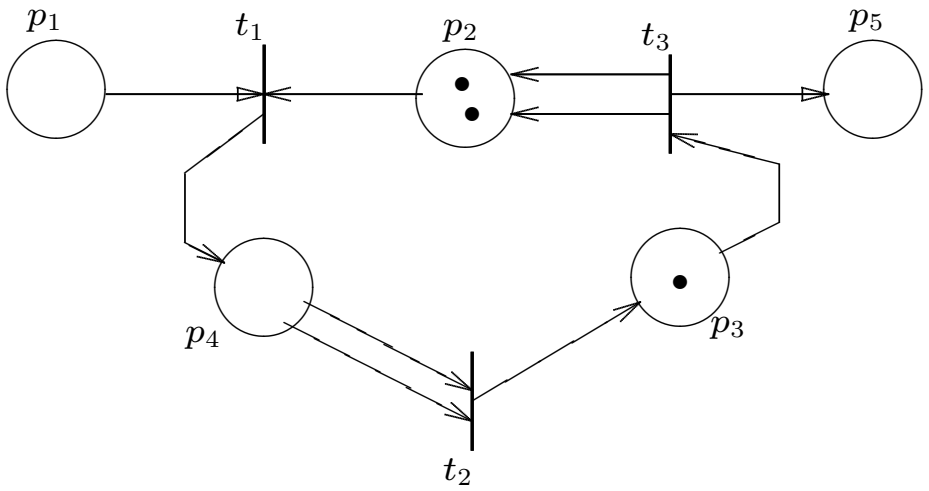
- Homogeneous resources;
- Constant number of resources;
- Communication is ignored;

Periodic, Admissible, Parallel Schedule

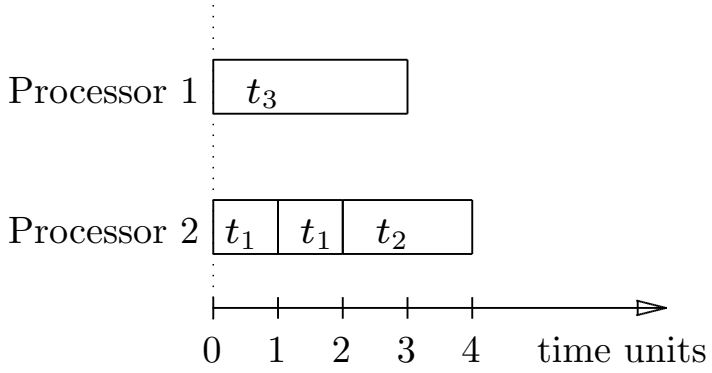
Definition: Given is an SDF process network and n resources. A periodic, admissible, parallel schedule (PAPS) is a periodic and infinite admissible sequential schedule for each of the resources available such that the data dependencies between the processes are respected. It is specified by a list $\{\psi_1, ..., \psi_n\}$, where ψ_i is a sequential schedule for processor i.

1. Compute a PASS schedule.
2. Determine a PASS "unroll factor" J, i.e. how many PASS cycles form a single PAPS cycle.
3. Construct a precedence graph.
4. Compute the PAPS based on the Hu-level algorithm.

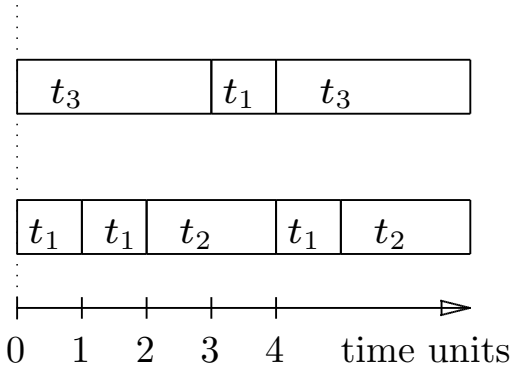
2. Determine the Unroll Factor J



A possible PASS is
$\phi = \langle t_1, t_1, t_2, t_3 \rangle$.

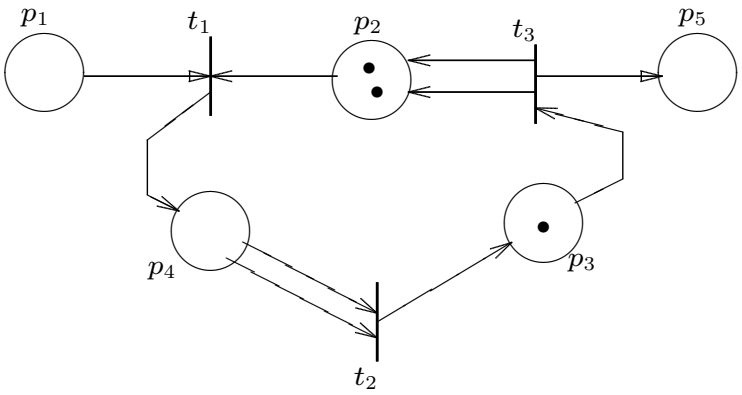


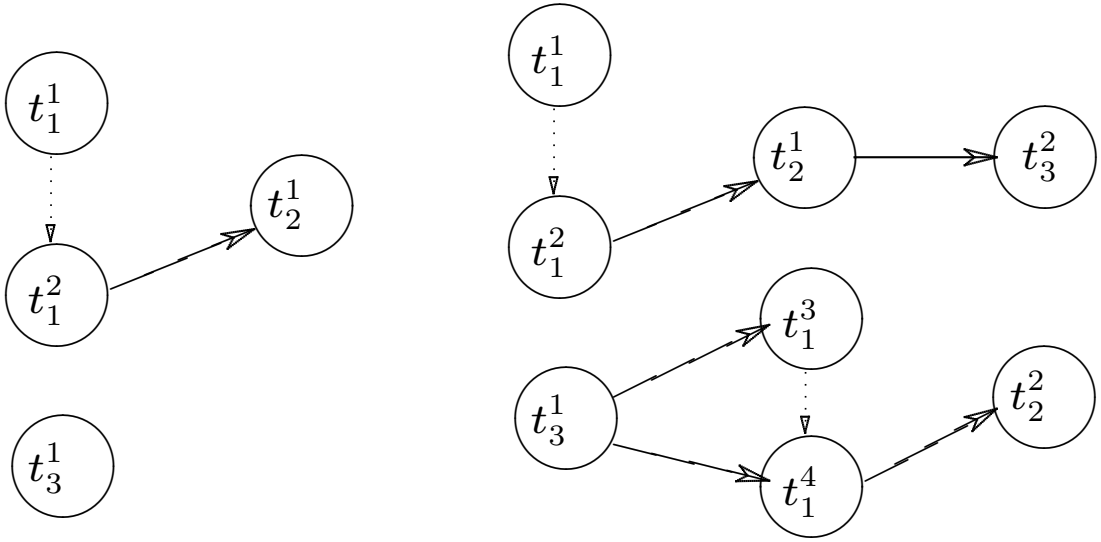
(a) $J = 1$



(b) $J = 2$

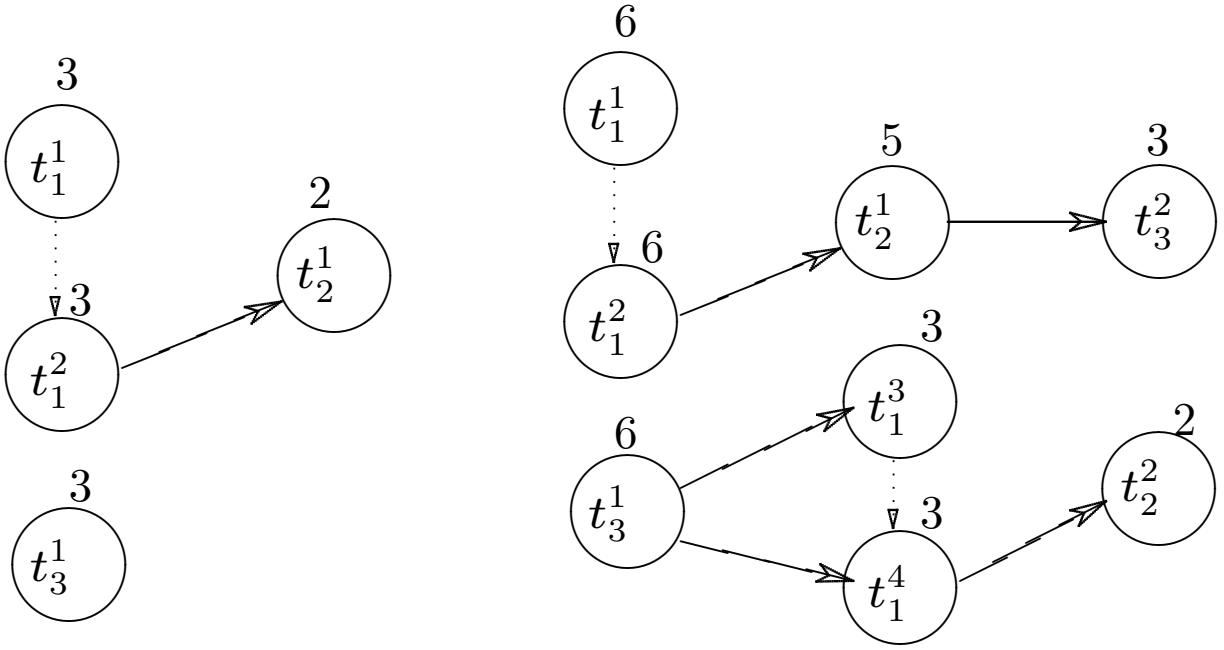
3. Construct Precedence Graph





(a) $J = 1$ (b) $J = 2$

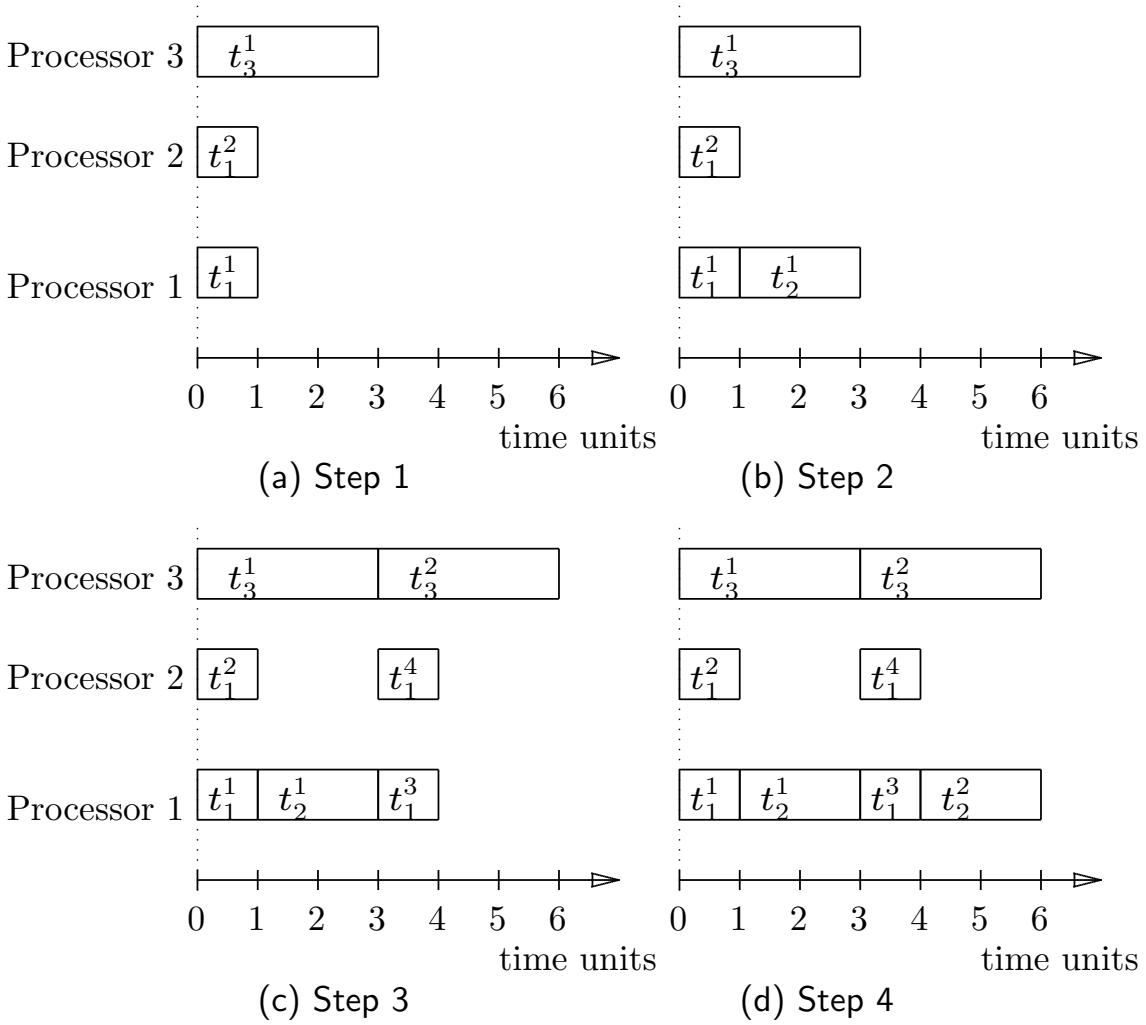
Precedence Graph Annotated with Hu-levels



(a) $J = 1$ (b) $J = 2$

Processors with runtime$(t_1) = 1$, runtime$(t_2) = 2$, runtime$(t_1) = 3$;

A Three Processor Schedule with $J = 2$



Summary

- Untimed Model of Computation
- Process instantiation
- Process composition
- Type and signature of processes
- Merging of processes
- Untimed process network as Petri net (SDF)
- Schedling and buffer analysis of SDF networks