Data-flow vs Control-flow

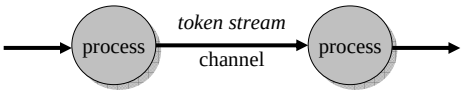
- Fuzzy distinction, yet useful for:
  - specification (language, model, ...)
  - synthesis (scheduling, optimization, ...)
  - validation (simulation, formal verification, ...)
- Rough classification:
  - **control**:
    - don't know when data arrive (quick reaction)
    - time of arrival often matters more than value
  - **data**:
    - data arrive in regular streams (samples)
    - value matters most

Data-flow vs Control-flow

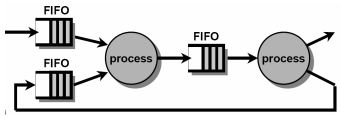
- Specification, synthesis & validation methods tend to emphasize …
- … for **control**:
  - event/reaction relation
  - response time
  - (Real Time scheduling for deadline satisfaction)
  - priority among events and processes
- … for **data**:
  - functional dependency between input and output
  - memory/time efficiency
  - (Dataflow scheduling for efficient pipelining)
  - all events and processes are equal

Process Networks

- Communicating processes with directed flow
  - communication: token "stream" between two processes
  - process: operations on tokens
  - host language: process description
  - coordination language: network description

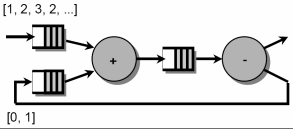


Kahn process networks (1974)



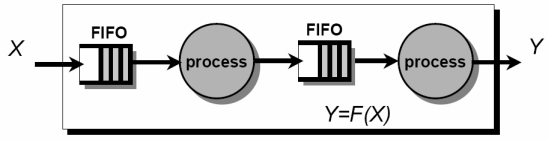
- special class of process networks
- stream is FIFO with unbounded capacity.
- process:
  - destructive read ("consumption") at process start,
  - non-destructive write ("production") at process end,
  - blocking read — process only executed if data available,
  - non-blocking write.

EXAMPLE



Kahn Process Networks: Formalism

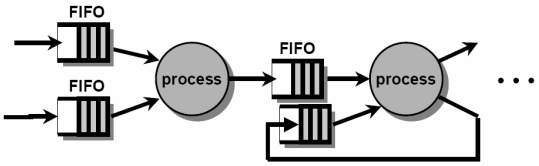
Sequence (a stream)	$X = [x_1, x_2, \ldots]$
Prefix ordering	$[x_1, x_2] \subseteq [x_1, x_2, x_3]$
Increasing chain of seq.	$\chi = \{X_0, X_1, \ldots\}$ where $X_0 \subseteq X_1$
Least upper bound	lub $\chi \subseteq Y$ where $X_i \subseteq Y$ for all $X_i \in \chi$



Continuous process F (lub χ)= lub F (χ)

Kahn Process Networks: Formalism

p-tuple of sequences	$\mathbf{X}=(X_0, X_1, \ldots X_p) \in S^p$
ordered set of seq.	$\mathbf{X} \subseteq \mathbf{X}'$ if $X_i \subseteq X_i'$ for each i
set of p-tuple of sequences	$\chi = \{\mathbf{X}_0, \mathbf{X}_1, \ldots\}$
functional process	$F : S^p \rightarrow S^q$



Continuous process F (lub χ)= lub F (χ)

Kahn Process Networks: Monotonicity

- Monotonicity
  - $\mathbf{X} \subseteq \mathbf{X}' \rightarrow F(\mathbf{X}) \subseteq F(\mathbf{X}')$
- It can be proved that…
  - ***a continuous process is monotonous***

➔ given a part of the input sequence it may be possible to compute part of the output sequence.

Monotonic does *not* imply continuous

- Consider F: $S \rightarrow S$

$$F(X) = \begin{cases} [0]; & \text{if } X \text{ is a finite sequence} \\ [0,1]; & \text{otherwise} \end{cases} . \quad (4)$$

Only two outputs are possible, both finite sequences. To show that this is monotonic, note that if the sequence X is infinite and $X \sqsubseteq X'$, then $X = X'$, so

$$Y = F(X) \sqsubseteq Y' = F(X') . \quad (5)$$

If X is finite, then $Y = F(X) = [0]$, which is a prefix of all possible outputs. To show that it is not continuous, consider the increasing chain

$$\chi = \{ X_0, X_1, \ldots \}, \text{where } X_0 \sqsubseteq X_1 \sqsubseteq \ldots , \quad (6)$$

where each X_i has exactly i elements in it. Then $\sqcup \chi$ is infinite, so

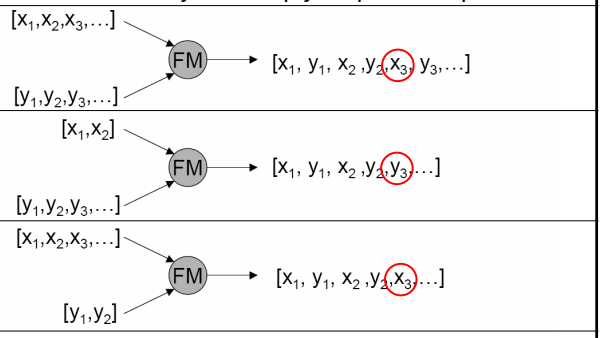
$$F(\sqcup \chi) = [0,1] \neq \sqcup F(\chi) = [0]. \quad (7)$$

Iterative computation of this function is clearly problematic.

A useful property is that a network of monotonic processes itself defines a monotonic process. This property is valid even for process networks with feedback loops, as is formally proven using induction by Panagaden and Shanbhogue [78]. It should not be surprising given the results so far that one can formally show that networks of monotonic processes are *determinate*.

Non-monotonic Processes

- "Canonical" non-monotonic process: *fair merge*
- **Fairness**: every non-empty sequence is processed



Non-monotonic Processes

- In the previous example, we have:

([x1,x2], [y1,y2,y3,…]) ⊆ ([x1,x2,x3,…], [y1,y2,y3,…])

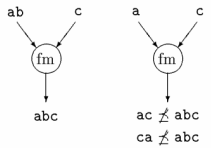
- but
  - [x1, y1, x2, y2, **x3**, y3, …] and
  - [x1, y1, x2, y2, **y3** , …]
- are incomparable

➔ The process is not monotonic (needs prediction of the future to be really fair).

➔The process is not continuous.

In fact the process is not even a (deterministic) function.

Fair Merge



Input a is a prefix of ab and c is a prefix of c, but neither of the possible outputs ac nor ca is a prefix of abc.

- Fair merge:
  - interleave input streams X_1 and X_2 to produce output stream Y

Least Fixed Point Semantics

Let X be the set of all sequences.

A network is a mapping F from the sequences to the sequences (where I represents the input sequence):

$$X = F(X, I)$$

The behavior of the network is defined as the unique *least fixed point* of the equation (*LFP*).

If F is continuous then the least fixed point exists

$$LFP = LUB(\{ F^n (\perp, I) : n \geq 0 \})$$