Dataflow Process Networks

An Introduction to a Mathematical Model of Dataflow

Thomas M. Parks

March 10, 1995

University of California, Berkeley

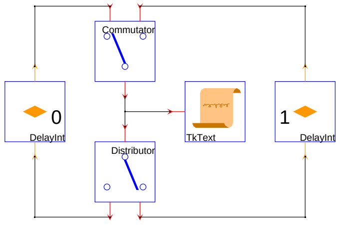
Ptolemy Conference

parks@eecs.berkeley.edu

http://www.eecs.berkeley.edu/∼parks

1 Process Networks

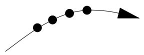
Produce a stream of 0's and 1's.
(from figure 2 in Kahn74)

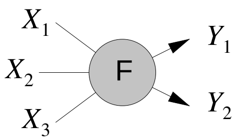


- concurrent processes
- FIFO communication channels, unbounded capacity
- non-blocking writes, blocking reads

1.1 Kahn's Formal Model

- A stream is a sequence of data tokens: $X_1 = [x_1, x_2, x_3, \ldots], \perp =$ empty.





- A process is a functional mapping from one set of sequences into another: $\mathrm{F}(\vec{X}) = \vec{Y}$
- blocking reads $\Longrightarrow$ determinate system

1.2 Streams

Stream:

$$\begin{aligned} X &= [x_1, x_2, \ldots] \in S \\ \perp &= [\,] \in S \end{aligned}$$

$$\begin{aligned} \vec{X} &= \{X_1, X_2, \ldots, X_p\} \in S^p \\ \vec{\perp} &= \{\perp, \perp, \ldots, \perp\} \in S^p \end{aligned}$$

Prefix Order:

$$\begin{aligned} [x_1, x_2] &\sqsubseteq [x_1, x_2, x_3] \\ \vec{X} &\sqsubseteq \vec{Y} \iff X_i \sqsubseteq Y_i \quad \forall X_i \in \vec{X} \end{aligned}$$

Increasing Chain:

$$\mathcal{X} = \{\vec{X}_1, \vec{X}_2, \dots\}$$
$$\vec{X}_1 \sqsubseteq \vec{X}_2 \sqsubseteq \cdots$$

Greatest Lower Bound:

$$\sqcup\mathcal{X} \sqsubseteq \vec{X}_i \qquad \forall \vec{X}_i \in \mathcal{X}$$

Least Upper Bound:

$$\vec{X}_i \sqsubseteq \sqcap\mathcal{X} \qquad \forall \vec{X}_i \in \mathcal{X}$$

1.3 Processes

Process:

$$\mathrm{F} : S^p \longrightarrow S^q$$

Continuous:

$$\mathrm{F}(\sqcap\mathcal{X}) = \sqcap\,\mathrm{F}(\mathcal{X})$$
$$\sqcap\,\mathrm{F}(\mathcal{X}) = \sqcap\{\mathrm{F}(\vec{X}_1), \mathrm{F}(\vec{X}_2), \dots\}$$

Monotonic:

$$\vec{X} \sqsubseteq \vec{Y} \implies \mathrm{F}(\vec{X}) \sqsubseteq \mathrm{F}(\vec{Y})$$

Sequential: F is continuous and

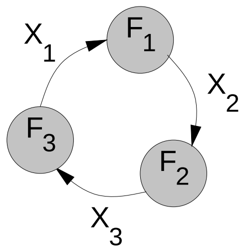
$$\forall \vec{X} \in S^p \qquad \exists X_i \in \vec{X}$$

such that

$$\mathrm{F}(\vec{X}) = \mathrm{F}(\vec{Y})$$
$$\forall \vec{Y}, \quad X_i = Y_i \quad \text{and} \quad \vec{X} \sqsubseteq \vec{Y}$$

sequential $\Longrightarrow$ continuous $\Longrightarrow$ monotonic

1.4 Determinism

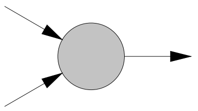


- S^p with $\sqsubseteq$ forms a complete partial order.
- Fix-point equations $\vec{X} = \vec{\mathrm{F}}(\vec{X})$ have a unique minimal solution when $\vec{\mathrm{F}}$ is continuous.
- Solution to fix-point equations corresponds to histories of channels.
- Histories will be the same for any correct implementation.

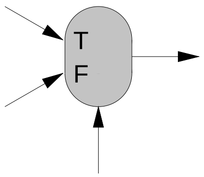
2 Dataflow Process Networks

2.1 Firing Rules

$$\mathrm{f} : S^p \longrightarrow S^q \qquad \mathcal{R} = \{\vec{R}_1, \vec{R}_2, \ldots, \vec{R}_N\}$$



$$\vec{R} = \{[*], [*]\}$$



$$\vec{R}_1 = \{[T], [*], \perp\}$$

$$\vec{R}_2 = \{[F], \perp, [*]\}$$

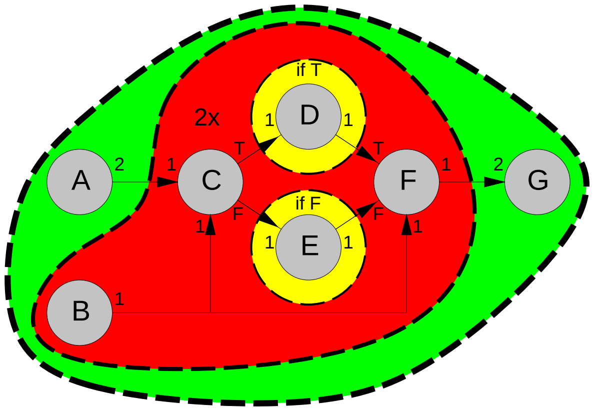
2.2 Higher Order Functions

2.2.1 $\mathrm{F} = \mathrm{map}(\mathrm{f})$

$$\begin{array}{rcll} \mathrm{map}(\mathrm{f})(\vec{R} : \vec{X}) & = & \mathrm{f}(\vec{R}) : \mathrm{map}(\mathrm{f})(\vec{X}) & \vec{R} \in \mathcal{R}_\mathrm{f} \\ \mathrm{map}(\mathrm{f})(\vec{Y}) & = & \vec{\perp} & \nexists \vec{R} \in \mathcal{R}_\mathrm{f} \quad \vec{R} \sqsubseteq \vec{Y} \end{array}$$

- f sequential $\Longrightarrow$ F sequential
- f continuous $\not\Longrightarrow$ F continuous

Sequential firing functions produce determinate process networks.



Loop: (conditionally) repeat an actor to match rates with neighbor

Merge: combine neighboring actors with same rate

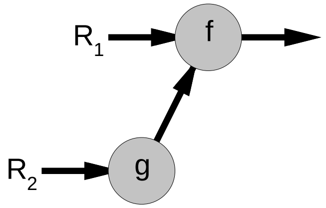
2.2.2 $\mathrm{g} = \mathrm{loop}(\mathrm{f}, N)$

$$\begin{array}{rcll} \mathrm{loop}(\mathrm{f}, N)(\vec{R} : \vec{X}) & = & \mathrm{f}(\vec{R}) : \mathrm{loop}(\mathrm{f}, N-1)(\vec{X}) & \vec{R} \in \mathcal{R}_\mathrm{f} \\ \mathrm{loop}(\mathrm{f}, N)(\vec{Y}) & = & \vec{\perp} & \nexists \vec{R} \in \mathcal{R}_\mathrm{f} \quad \vec{R} \sqsubseteq \vec{Y} \\ & = & \vec{\perp} & N = 0 \end{array}$$

What are the firing rules of $\mathrm{g} = \mathrm{loop}(\mathrm{f}, N)$?

$$\vec{R}_\mathrm{g} = \vec{R}_\mathrm{f} : \vec{R}_\mathrm{f} : \cdots \vec{R}_\mathrm{f}$$

2.2.3 $\mathrm{h} = \mathrm{merge}(\mathrm{f}, \mathrm{g})$



$$\mathrm{merge}(\mathrm{f}, \mathrm{g})(\{R_1, R_2\}) = f(\{R_1, \mathrm{g}(R_2)\})$$

What are the firing rules of $\mathrm{h} = \mathrm{merge}(\mathrm{f}, \mathrm{g})$?