Deterministic Receptive Processes are Kahn Processes

Stephen A. Edwards & Olivier Tardieu

Department of Computer Science, Columbia University

www.cs.columbia.edu/~{sedwards,tardieu}

{sedwards,tardieu}@cs.columbia.edu
Motivation

SHIM project: “Software/Hardware Integration Medium”

Want an asynchronous concurrent deterministic formalism for embedded systems.

I found two:

Kahn’s process networks (1974)


Are they “the same”?
Deterministic Merge

Each $a$ or $b$ input produces an $o$ output $\Leftrightarrow$ The number of $o$’s is the sum of the number of $a$’s and $b$’s.

$$
\begin{align*}
\epsilon & \quad \text{aaaaooo} & \quad \text{abaoao} & \quad \text{baaoao} & \quad \text{bbaoao} \\
ao & \quad \text{aaaaao} & \quad \text{aboaoa} & \quad \text{baaoao} & \quad \text{bbaoao} \\
b0 & \quad \text{aaoaoa} & \quad \text{aboaao} & \quad \text{baaoao} & \quad \text{bbaoao} \\
nao & \quad \text{aaoaao} & \quad \text{aobaao} & \quad \text{bbaaao} & \quad \text{bbaaao} \\
om & \quad \text{aaooao} & \quad \text{aboaao} & \quad \text{bbaaao} & \quad \text{bbaaao} \\
o & \quad \text{aaaoao} & \quad \text{aabaao} & \quad \text{babooo} & \quad \text{bbbooo} \\
aboo & \quad \text{aaboao} & \quad \text{abboaoo} & \quad \text{bbaaoo} & \quad \text{bbaboo} \\
aobo & \quad \text{aaaboo} & \quad \text{abobb} & \quad \text{bbaaoo} & \quad \text{bbbaoo} \\
asoo & \quad \text{aaoboo} & \quad \text{aboboo} & \quad \text{baaoo} & \quad \text{bbaboo} \\
boa & \quad \text{aaooab} & \quad \text{aboboo} & \quad \text{baaoo} & \quad \text{bbaboo} \\
booo & \quad \text{aaobbo} & \quad \text{abboob} & \quad \text{baabo} & \quad \text{bbabo} \\
boobo & \quad \text{aaaboo} & \quad \text{abboob} & \quad \text{baabo} & \quad \text{bbabo} \\
boobo & \quad \text{aaaboo} & \quad \text{abboob} & \quad \text{baabo} & \quad \text{bbabo} \\
boobo & \quad \text{aaaboo} & \quad \text{abboob} & \quad \text{baabo} & \quad \text{bbabo}
\end{align*}
$$
Deterministic Receptive Processes

In Mark Josephs’s formalism,

$$\varepsilon \quad \text{these traces are failures because the process fails to produce more outputs afterwards.}$$

$$ao$$

$$bo$$

$$aaoo$$

$$aoao$$

$$aboo$$

$$aobo$$

$$bboo$$

$$boao$$

$$boao$$

$$bboo$$

$$bobo$$

The set of failure traces characterizes one of Josephs’s deterministic receptive processes.

This process is deterministic and receptive according to Josephs.
Josephs’s Receptive Processes


Process: \((I, O, F)\)

\(I \cap O = \emptyset\) input/output alphabets

Set of failure traces: \(F \subseteq (I \cup O)^*\)

Divergences: \(F^\uparrow = \{s : \{t \in O^* : st \in F\} \text{ is infinite}\}\)

“When an infinite sequence of outputs is possible”

Traces: \(\hat{F} = \{s : \exists t \in O^* . st \in F\}\)

“When zero or more outputs are pending”
Receptive Process Axioms

\[ s \in F^\uparrow \Rightarrow st \in F^\uparrow \]
\[ F^\uparrow \subseteq F \]
\[ \epsilon \in \hat{F} \]
\[ st \in \hat{F} \Rightarrow s \in \hat{F} \]
\[ s \in \hat{F} \land t \in I^* \Rightarrow st \in \hat{F} \]

- Anything follows a divergence
- Divergences are failures
- Traces start from nothing
- Traces prefix-closed
- Input always possible = receptive

\( (io)^*i \)  
\( (io)^* \)  
\( (io)^*ii(i|o)^* \)

traces \( \hat{F} \)
failures \( F \)
divergences \( F^\uparrow \)
Problem: process can choose whether to output $o$ or $p$. 

\[(i(o|p))^*i\]

\[(i(o|p))^*\]

\[(i(o|p))^*ii(i|o|p)^*\]

traces $\hat{F}$

failures $F$

divergences $F^\uparrow$
Deterministic Receptive Processes


Four additional rules: one about inputs, three about outputs.

\[(\forall v, w. x = vw \Rightarrow svi \notin F^\uparrow) \land\]

\[i \in I \land sxiu \in F \Rightarrow sixu \in F\]

"An input that arrives early does not matter unless it causes divergence."
Deterministic Receptive Processes

\[ o \in O \land t \in (I \cup (O \setminus \{o\}))^* \land \]
\[ so \in \hat{F} \land st \in F \setminus F^{\uparrow} \quad \Rightarrow \quad \text{false} \]
\[ o \in O \land t \in (I \cup (O \setminus \{o\}))^* \land \]
\[ so \in \hat{F} \land st \in \hat{F} \quad \Rightarrow \quad sto \in \hat{F} \]
\[ o \in O \land t \in (I \cup (O \setminus \{o\}))^* \land \]
\[ so \in \hat{F} \land stou \in F \setminus F^{\uparrow} \quad \Rightarrow \quad sotu \in F \]

“If an output can occur now, it must be emitted before the process stops to wait for inputs.”

“An output may always be delayed.”

“Delaying an output does not affect long-term behavior.”
Kahn’s Networks

Alternating sequence of 0s and 1s along center channel

Emits a 1 then copies input to output

Copies alternately

Merges by alternating

Emits a 0 then copies input to output
Kahn’s Processes

“A Simple Language for Parallel Programming”

process f(in int u, in int v, out int w)
{
    int i; bool b = true;
    for (; ; ) {
        i = b ? wait(u) : wait(v);
        printf("%i\n", i);
        send(i, w);
        b = !b;
    }
}
Channels convey sequences of data values.

Sequences partially ordered: $aa \sqsubseteq aaaa$, but $aa \not\sqsubseteq ab$.

Each process a function on finite and infinite sequences

$$f : D_1^\omega \times D_2^\omega \times \cdots \times D_n^\omega \to D^\omega$$

$f$ is monotonic, $x \sqsubseteq y \Rightarrow f(x) \sqsubseteq f(y)$, and continuous

$$f(\bigsqcup X) = \bigsqcup f(X).$$

Continuity guarantees the function of a system

$F = (f_1, f_2, \ldots, f_k)$ has a unique least fixed point

$F(X) = X$. This is the (only) behavior of the system.
Deterministic Merge

As a Kahn process,

\[(\varepsilon, \varepsilon) = \varepsilon \quad (a, \varepsilon) = o \quad (aa, \varepsilon) = oo\]

\[(\varepsilon, b) = o \quad (a, b) = oo \quad (aa, b) = ooo\]

\[(\varepsilon, bb) = oo \quad (a, bb) = ooo \quad (aa, bb) = oooo \quad \cdots\]

\[(\varepsilon, bbb) = ooo \quad (a, bbb) = ooooo \quad (aa, bbb) = oooooo\]

\[(\varepsilon, bbbb) = oooo \quad (a, bbbb) = oooooo \quad (aa, bbbb) = ooooooo\]

\[\vdots\]

Clearly monotonic and continuous, hence deterministic.

*Cannot be described in Kahn's sequential language.*
A constructive proof that Deterministic Receptive Processes behave like Kahn processes
Projection selects a single event from a trace:

\[
\epsilon \downarrow e \quad = \quad \epsilon \\
\alpha s \downarrow e \quad = \quad \begin{cases} 
\alpha (s \downarrow e) & \text{if } \alpha = e, \text{ and} \\
 s \downarrow A & \text{otherwise.}
\end{cases}
\]

\[
i_1i_2o_1o_2i_1o_2o_1i_2i_1o_1o_2 \downarrow i_1 = i_1i_1i_1 \\
i_1i_2o_1o_2i_1o_2o_1i_2i_1o_1o_2 \downarrow i_2 = i_2i_2
\]
Input and Output Functions

\[ \mathcal{I}(f) = (f \downarrow i_1, f \downarrow i_2, \ldots, f \downarrow i_p) \]
\[ \mathcal{O}(f) = (f \downarrow o_1, f \downarrow o_2, \ldots, f \downarrow o_q) \]

Example: \( f = i_1 i_2 o_1 o_2 i_1 o_2 o_1 i_2 i_1 o_1 o_2 \)
\[ \mathcal{I}(f) = (i_1 i_1 i_1, i_2 i_2) \]
\[ \mathcal{O}(f) = (o_1 o_1 o_1, o_2 o_2 o_2) \]
The Central Lemma

The input/output relationship of a deterministic receptive process \( P = (I, O, F) \) with no divergence is monotonic, i.e., for \( f_1, f_2 \in F \), if \( I(f_1) \sqsubseteq I(f_2) \) then \( O(f_1) \sqsubseteq O(f_2) \).

**Proof** by contradiction. Assume \( I(f_1) \sqsubseteq I(f_2) \) but \( O(f_1) \nsubseteq O(f_2) \).

Reorder the events in \( f_1 \) and \( f_2 \) so that inputs appear first and the two share a common prefix.

There must be at least one more output that occurs less often in \( f_2 \) and hence in the reordered traces, but this contradicts the axiom of compulsory emission. QED.
An Illustration

\[ f_1 = i_1 o_1 i_2 o_2 i_2 o_2 i_1 \]
\[ f_2 = i_1 i_2 i_2 o_2 i_1 o_2 i_1 i_2 \]

\[ \mathcal{I}(f_1) = (i_1 i_1, i_2) \subseteq \mathcal{I}(f_2) = (i_1 i_1 i_1, i_2 i_2 i_2) \]
\[ \mathcal{O}(f_1) = (o_1, o_2 o_2) \nsubseteq \mathcal{O}(f_2) = (\epsilon, o_2 o_2) \]

Move inputs earlier (safe because no divergence)

\[ f'_1 = i_1 i_1 i_2 i_2 o_1 o_2 o_2 \]
\[ f'_2 = i_1 i_1 i_2 i_2 i_1 i_2 o_2 o_2 \]

\( f'_2 \) cannot be a failure because the output \( o_1 \) must eventually be emitted. Contradiction.
Technical point

Josephs only talks about finite traces
Kahn needs infinite traces because he takes limits
Unsurprising result: define the behavior of Josephs’s process as being its limit and everything works.
Conclusion

Kahn and Josephs deterministic for roughly same reason.

Big difference: Josephs models “don’t-cares” as divergences—no obvious analog in Kahn’s model.

Josephs’s axioms more complex, but more operational.

Not in the paper: we have found a more fundamental definition that makes Josephs’s axioms lemmas.

Ongoing work: developing the SHIM model and system built around a Kahn/Josephs-like model of computation.