Synchronous Hypotheses

Behind the scene …

Determinism & Reactivity

- Determinism:

  The same input sequence always yields

  The same output sequence

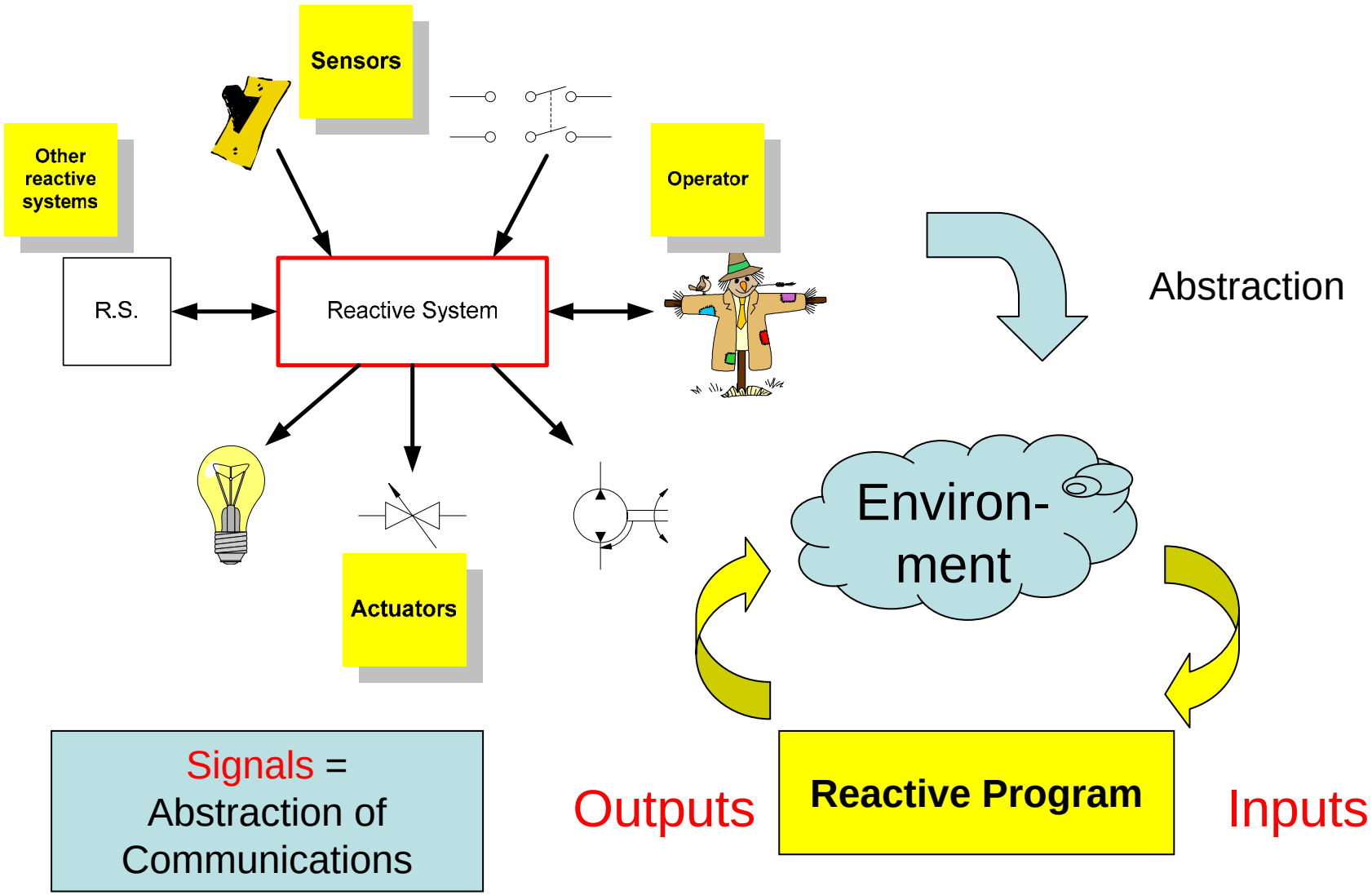
- Reactivity:

  The program must react(1) to any stimulus

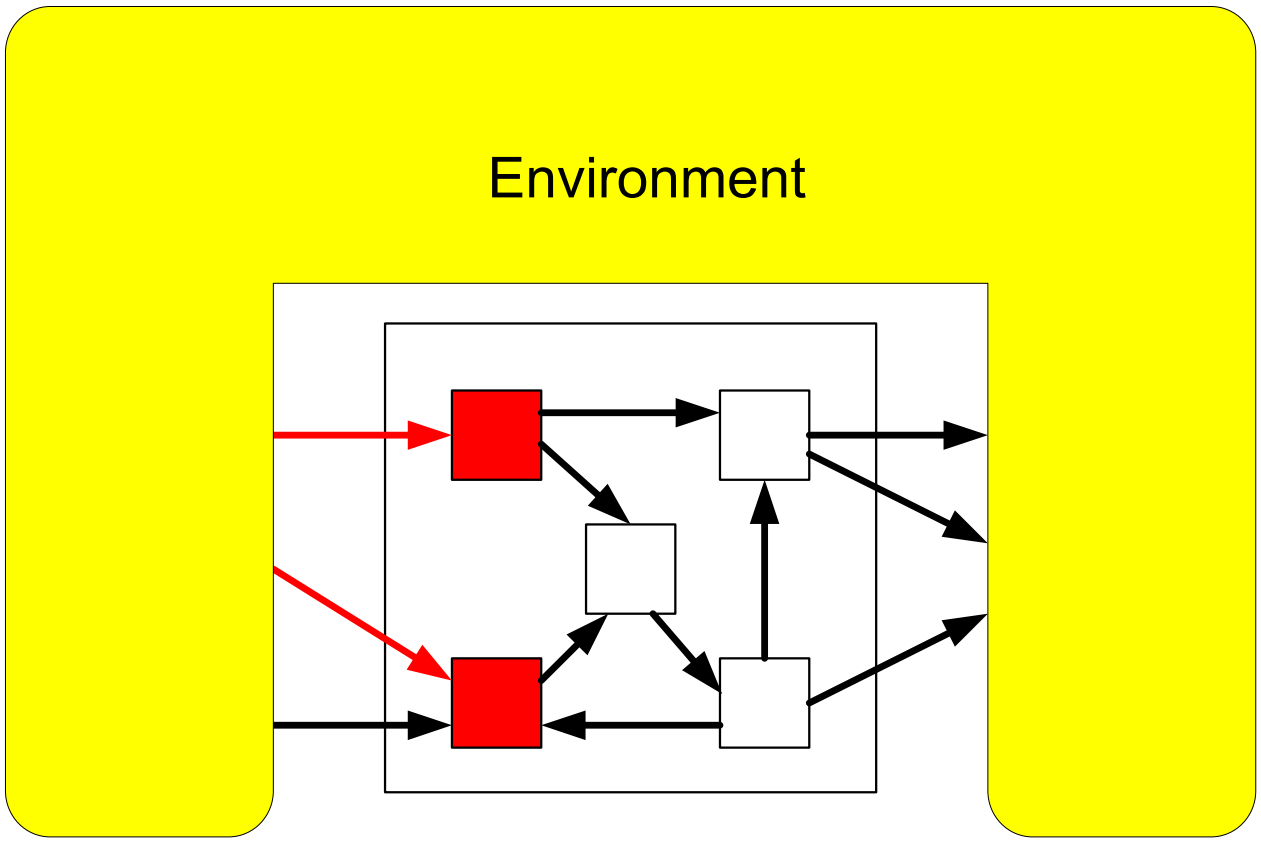
  Implies absence of deadlock

(1) Does not necessary generate outputs, the reaction may change internal state only.

Abstraction



Reactive Programs

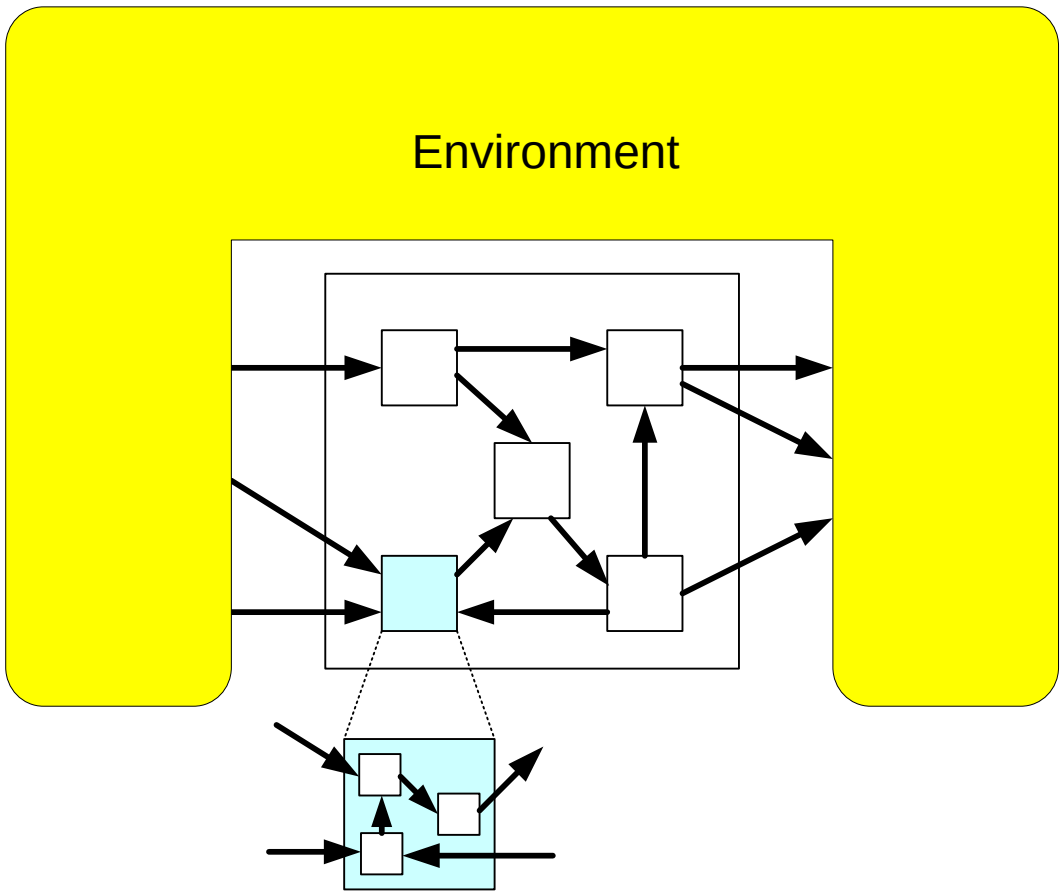


Incoming Communications

Actors' reactions and internal communications

Outgoing Communications

Reactive Programs



Refinement

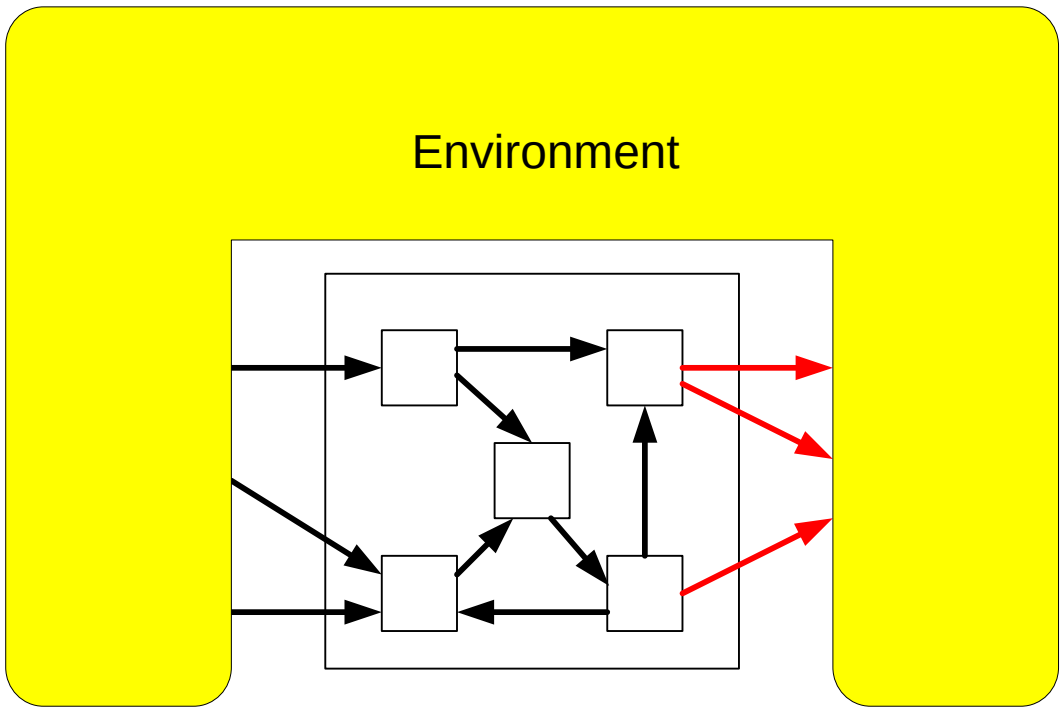
Demands for expressing

- Communication
  - Internal and with the environment
- Simultaneous activities
  - Independent/Concurrent/Parallel/ Sequential
- Modularity / Interfaces
- Hierarchy

Synchronous Reactive Programs (1)

Atomic execution:

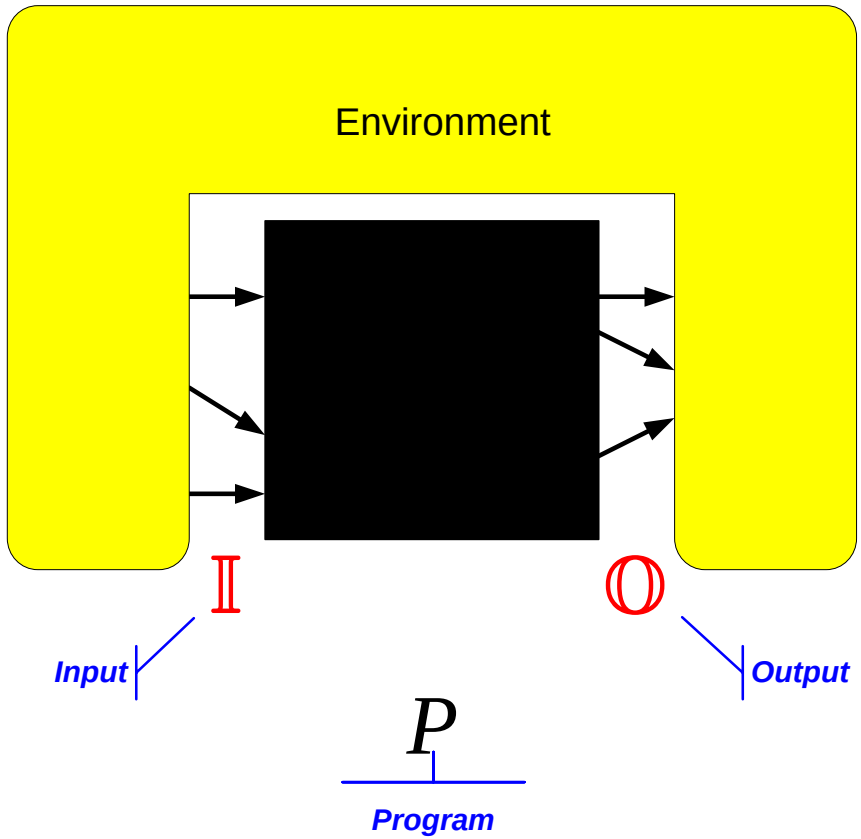
- Read
- Compute
- Write



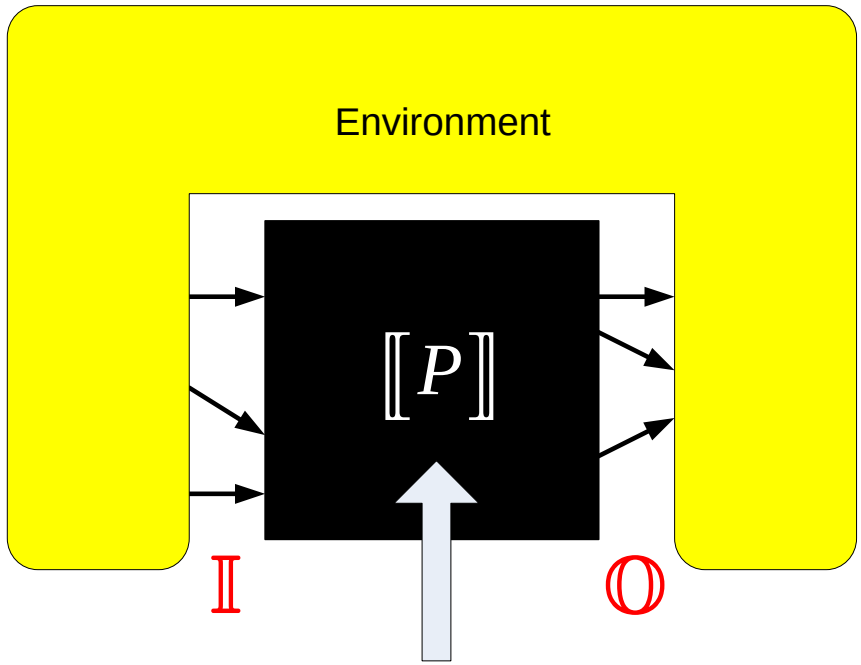
The Synchronous Approach

- Notion of logical instant
  - Simultaneity
  - Priority (reaction to absence)
- Signals
  - Abstraction of communications
  - Instantaneous broadcast
- Non-overlapping sequence of reactions
- Control
  - Deterministic, inter/intra instant
  - Preemption
- Possible instantaneous actions (0-duration)

Synchronous Reactive Programs (2)



Synchronous Reactive Programs (2)

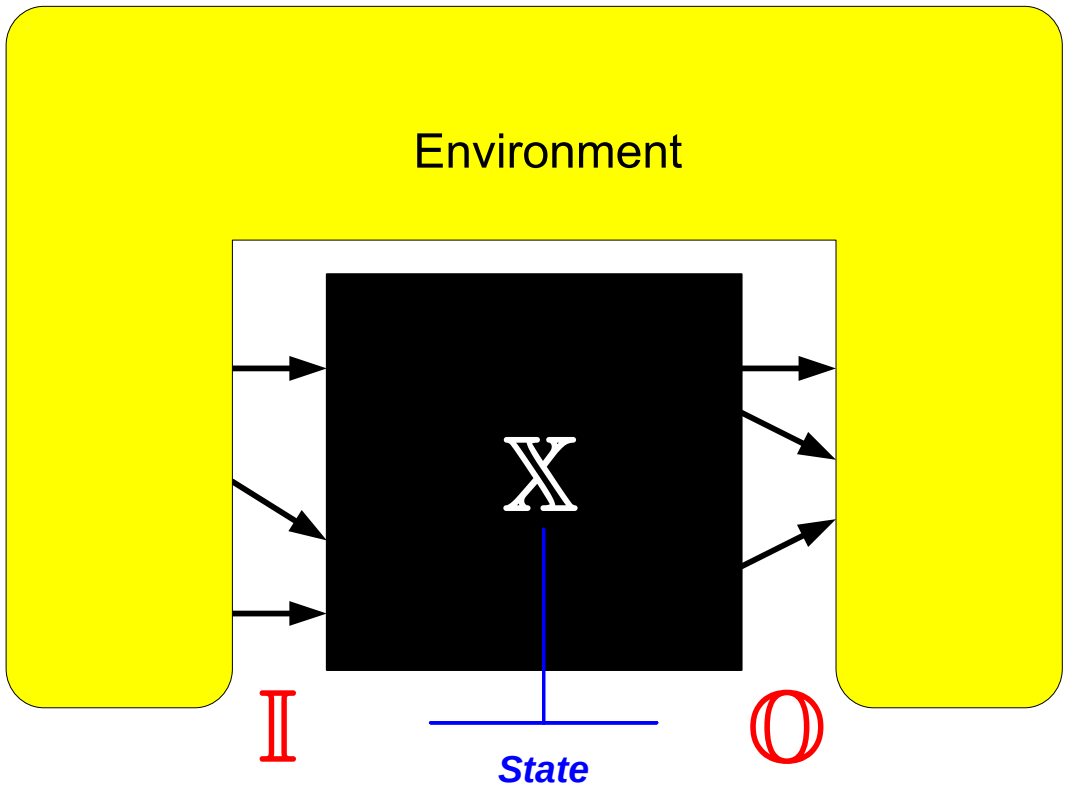


$$[\![P]\!] : \mathbb{I}^* \rightarrow \mathbb{O}^*$$

Deterministic transformation of input sequences into output sequences.

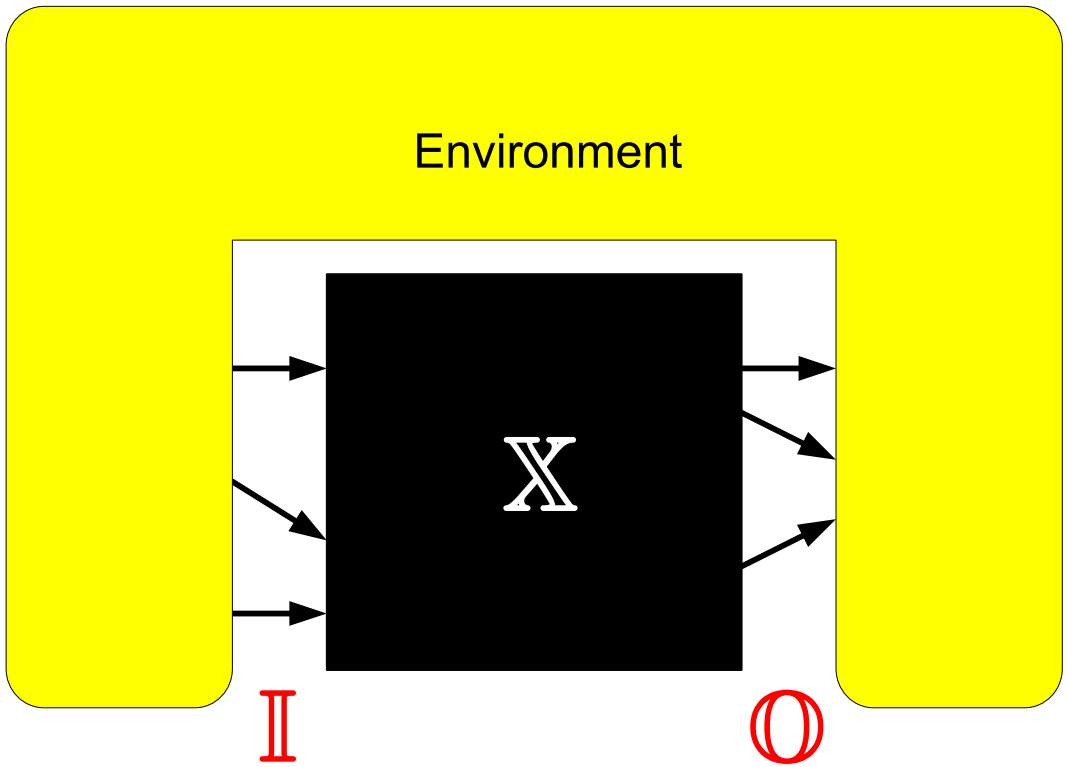
Sequences are indexed by logical instants.

Synchronous Reactive Programs (3)



Reaction : $(\mathbb{X}, \mathbb{I}) \mapsto (\mathbb{X}', \mathbb{O})$

Synchronous Reactive Programs (3)

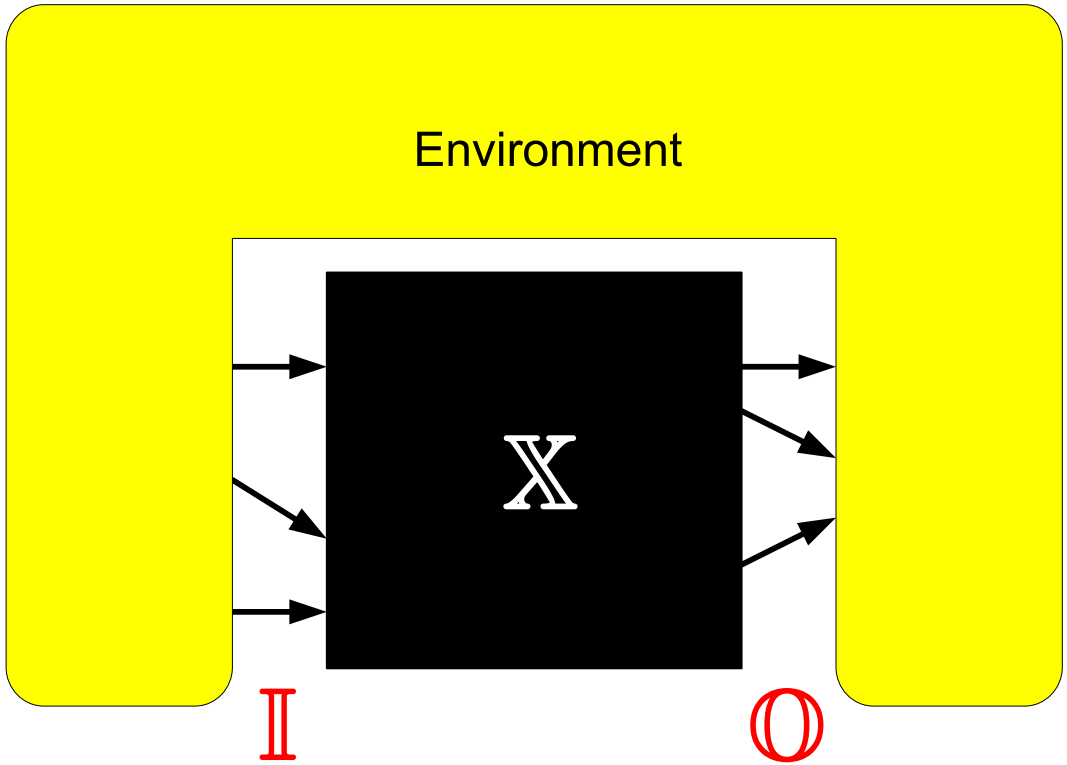


Reaction : $(\mathbb{X}, \mathbb{I}) \mapsto (\mathbb{X}', \mathbb{O})$

$\lambda : \mathcal{X} \times \mathcal{I} \to \mathcal{X}$ (next state function)

$\omega : \mathcal{X} \times \mathcal{I} \to \mathcal{O}$ (output function)

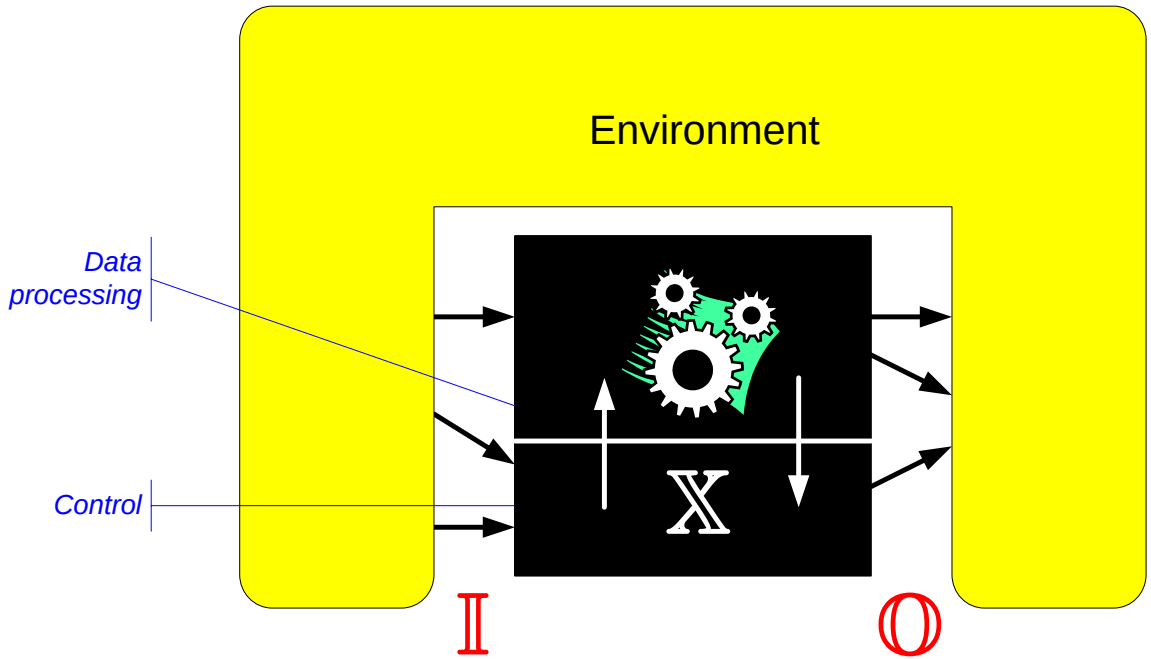
Synchronous Reactive Programs (3)



$$P \xrightarrow[\mathbb{I}]{\mathbb{O}} P' \text{ (rewriting)}$$

$$P = P_0 \xrightarrow[\mathbb{I}_1]{\mathbb{O}_1} P_1 \xrightarrow[\mathbb{I}_2]{\mathbb{O}_2} P_2 \cdots P_{n-1} \xrightarrow[\mathbb{I}_n]{\mathbb{O}_n} P_n \cdots$$

Synchronous Reactive Programs (4)

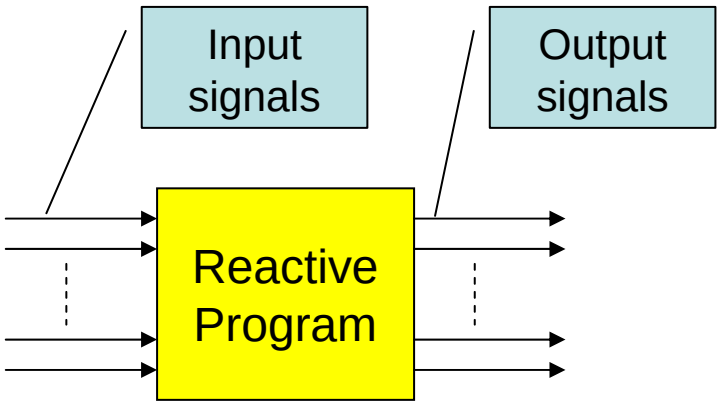


Reaction : $(\mathbb{X}, \mathbb{I}) \mapsto (\mathbb{X}', \mathbb{O})$ Control (finite state)

+ controlled data processing

$P \rightarrow$ a compiled scheduler

Hypotheses: Tuples of Signals



Tuples of signals
a.k.a. vectors

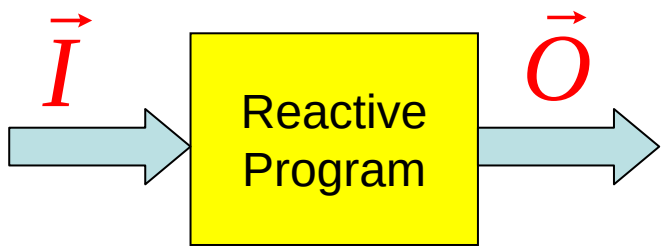
a1 a2 a3 a4 ….
b1 b2 b3 b4 ….
….
z1 z2 z3 z4 ….

Sequence of tuples
vs. tuples of sequences

But this is too strong a requirement!

Some signals may be absent.

a1 ⊥ a2 a3 ⊥ a4 ….
⊥ b1 b2 ⊥ b3 b4 ….
….
z1 z2 z3 ⊥ z4 ⊥ ….



Hypotheses: Non overlapping Reactions

- Actually, a synchronous model works on a logical time.
- The time is
  - Discrete
  - Total ordering of instants.

  } Use $\mathbb{N}$ as time base
- A reaction executes in one instant.
- Actions that compose the reaction may be partially ordered.

Hypotheses: Instantaneous broadcast

- Communications between actors are also supposed to be instantaneous.
- All parts of a synchronous model perceive exactly the same information (instantaneous broadcast).
- Outcome: Outputs are simultaneous with Inputs (they are said to be synchronous)
- Thanks to these strong hypotheses, parallel composition can be fully deterministic.

Hypotheses: Summary

- Signals are the unique support of communications
- The model deals with tuples of signals
- Any reaction has a 0-duration
- Logical time: instants, simultaneity
- Instantaneous broadcast of information

Synchrony and Real-Time

- Relationship between Logical time and "physical" time?
- Through distinguished signals bound to "time passing" (i.e., clock signals)
- A simple (and usual way) to execute a synchronous program:

```
do
   await Activation // a signal taken as a clock
   read inputs // acquisition
   execute actions // reaction
   write outputs // actuation
forever
```

Execution of the body of the loop =

A **reaction** ↔ 1 **instant**

Synchrony and Real-Time

- For automatic control and digital signal processing applications, a periodic (w.r.t. physical time) signal is chosen as activation signal: A form of time-triggered system.
- For event-triggered systems another execution mode may be taken:

```
do
    wait for an event
    execute actions // reaction
    write outputs // actuation
forever
```

Synchrony and Real-Time

- Since executions are fully deterministic, the longest (actual) execution time for a reaction δ can be computed accurately(1).
- For a given environment (contractual aspect) with a bounded and known response time τ, the synchronous implementation runs in real-time if δ is negligible w.r.t. τ.
- Precise meaning of negligible: refer to the system engineer

(1) Provided it can be computed for all elementary actions

Synchrony and Real-Time

- Remark:

  Digital circuits have been using a similar approach for long: known as “Synchronous circuits”

  Synchronous circuits are easier to develop

  In some special cases asynchrony is unavoidable.

- So are synchronous programs:
  - Easier to write, analyze, implement
  - Not applicable when underlying hypotheses are unrealistic for the application at hand.

Semantics of Synchronous Languages

- Synchronous languages (SL) are specialized languages, not general purpose ones (GPLs).
- SL are dedicated to reactive system programming.
- Compared to GPLs, Synchronous languages are both
  - Simpler (limited number of constructs)
  - Cleaner (based on mathematical semantics)
- May be more efficient (sophisticated compilation)