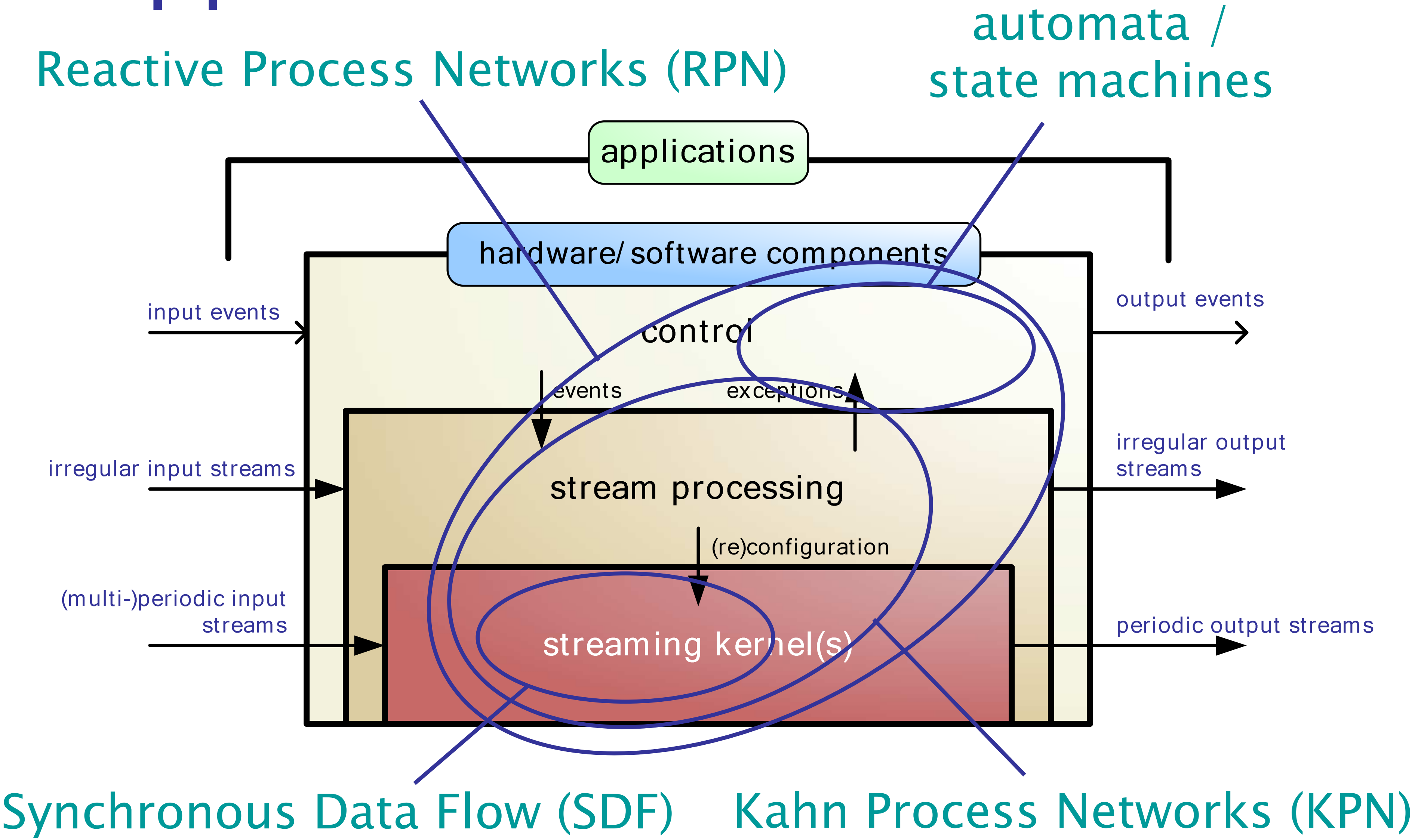
on Kahn process networks and reactive process networks

Marc Geilen and Twan Basten

Eindhoven University of Technology
Department of Electrical Engineering
Design Methodology for Electronic Systems

m.c.w.geilen@tue.nl

application domain



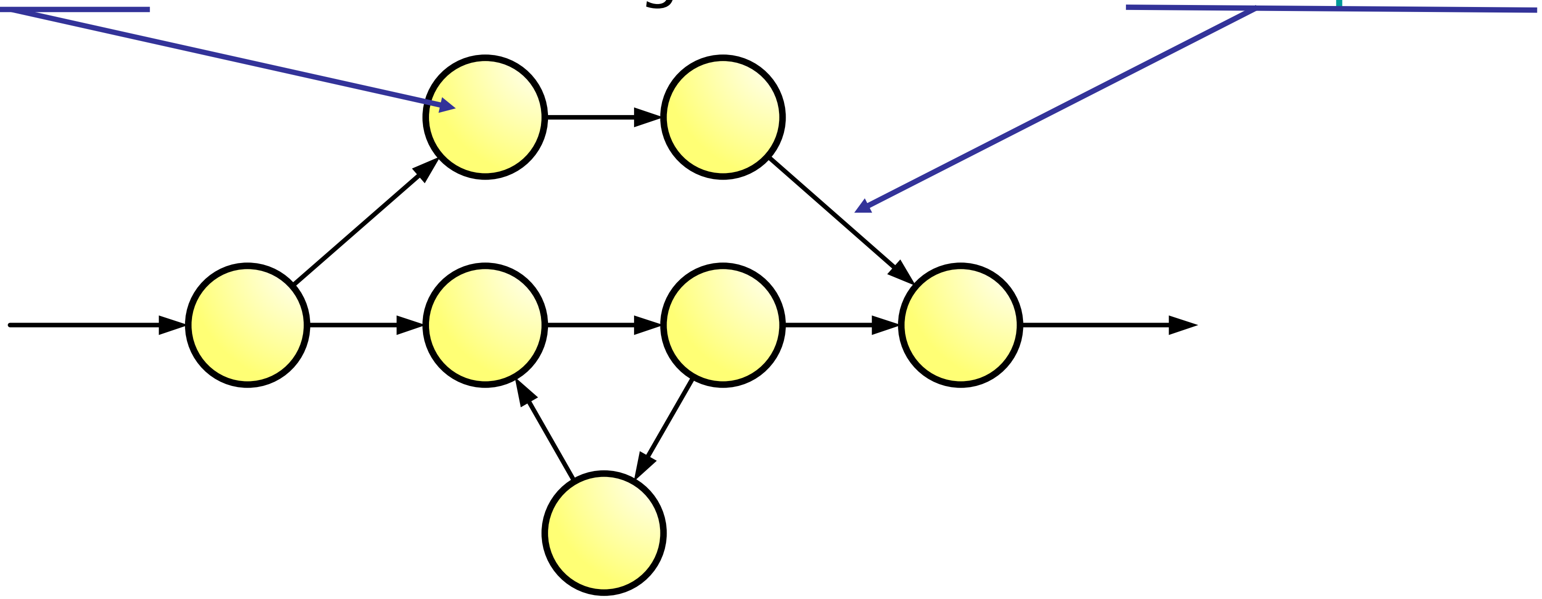
dataflow MoCs

- use (C)SDF where possible (presentations Sander Stuijk, Bart Theelen)
- use KPN for data-dependent streaming
- dynamic generation and reconfiguration of process networks
- interaction with non-streaming parts of an application

Kahn Process Networks

Kahn Process Networks

processes communicating via unbounded fifo queues



- reads block on empty queues
- writes may never block
- no global variables

[Kahn 1974]

denotational semantics

processes: functions from input strings to output strings

fifos: connect functions, hold (window on) strings

fixpoint semantics

continuous function of a network is the least fixpoint of a set of fixpoint equations

compositionality

if functions are continuous, then a network is also a continuous function

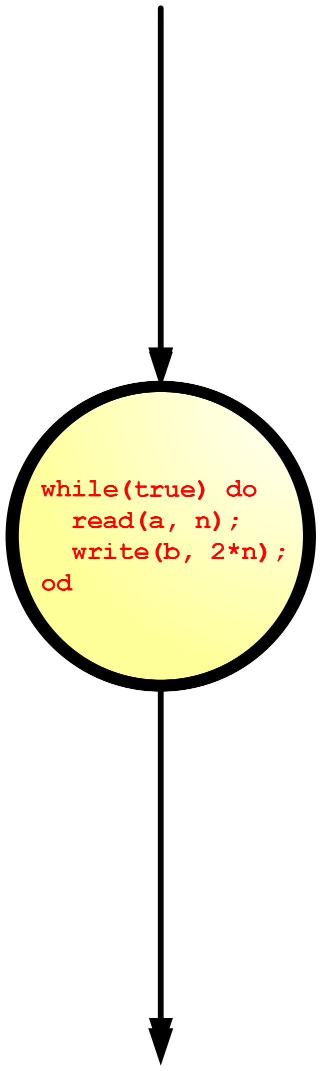
strengths & weaknesses

- compositionality
- determinacy (execution order and timing)
- explicit concurrency
- explicit communication
- captures data-dependent streaming behavior
- high abstraction level
- needs run-time resource management
- cannot capture asynchronous reactive behaviour
- undecidable, e.g., minimal buffer sizes
  (‘Turing complete’)

implementing KPNs

realizations of KPNs

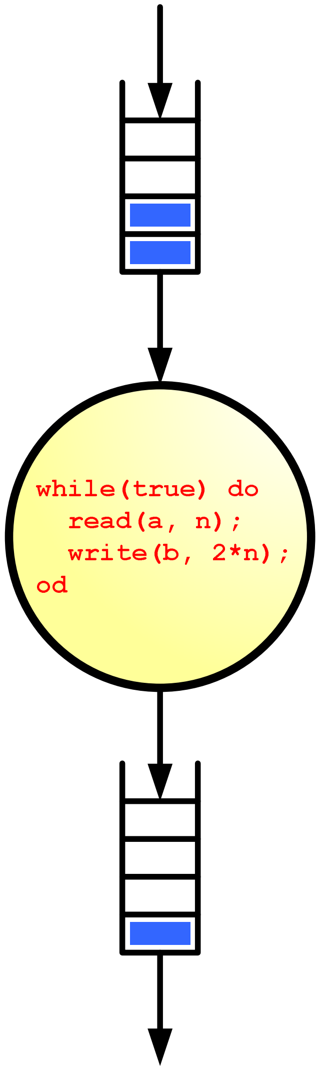
- functions: sequential programs
  e.g. C(++) or Java
- read and write operations



```
while(true) do
  read(a, n);
  write(b, 2*n);
od
```

realizations of KPNs

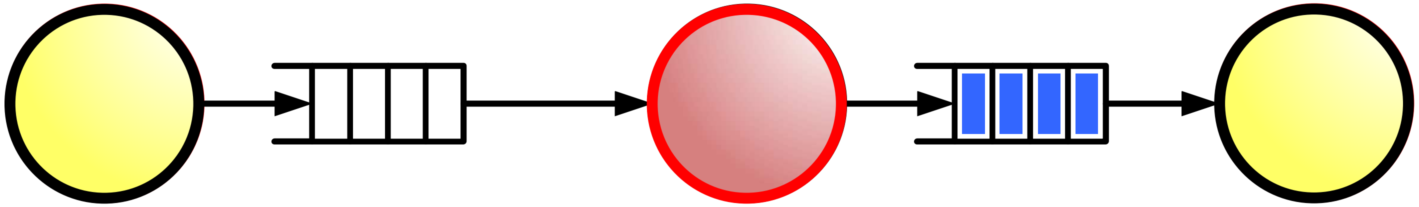
- functions: sequential programs
  e.g. C(++) or Java
- read and write operations
- arcs: FIFO queues
  store tokens that are written
  but not yet read



```
while(true) do
   read(a, n);
   write(b, 2*n);
od
```

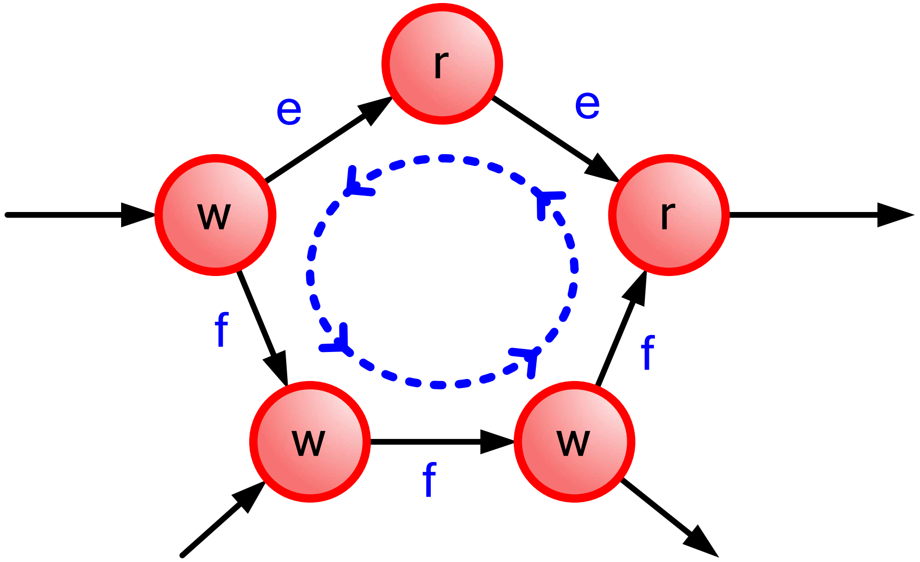
implementations of KPNs

- usually follow Thomas Parks' scheduling approach (YAPI, Ptolemy II, among others)
- bounded FIFOs combine aspects of data- and demand driven execution



implementations of KPNs

- FIFO bounds balance memory usage and context switching
- risk for artificial deadlocks
- run–time management of FIFO bounds



requirements [Parks, ESOP03]

boundedness

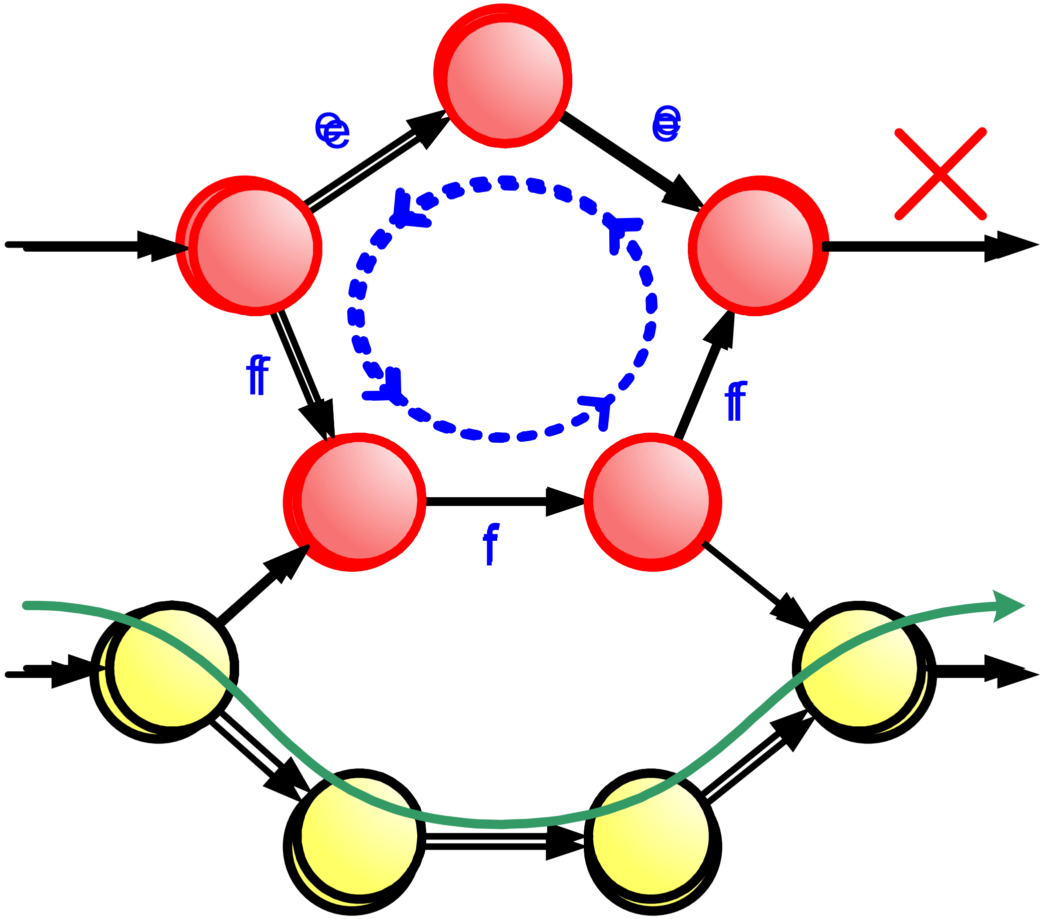
fifo bounds may not grow indefinitely if a bounded execution exists

completeness

progress must be made on ***all outputs*** *as prescribed by the denotational semantics*

scheduling KPNs

traditional execution model [Parks, 95] does not (always) follow Kahn’s semantics [ESOP03]



improved KPN scheduler

- a scheduler that is correct for every KPN cannot exist! [ESOP03]
- a scheduling algorithm has been defined which is correct for every bounded and "effective" KPN

  it is executed in bounded memory by our scheduler and produces the complete output
- prototype implementation in YAPI and by other (Olson and Evans, 2005)

improved KPN scheduler

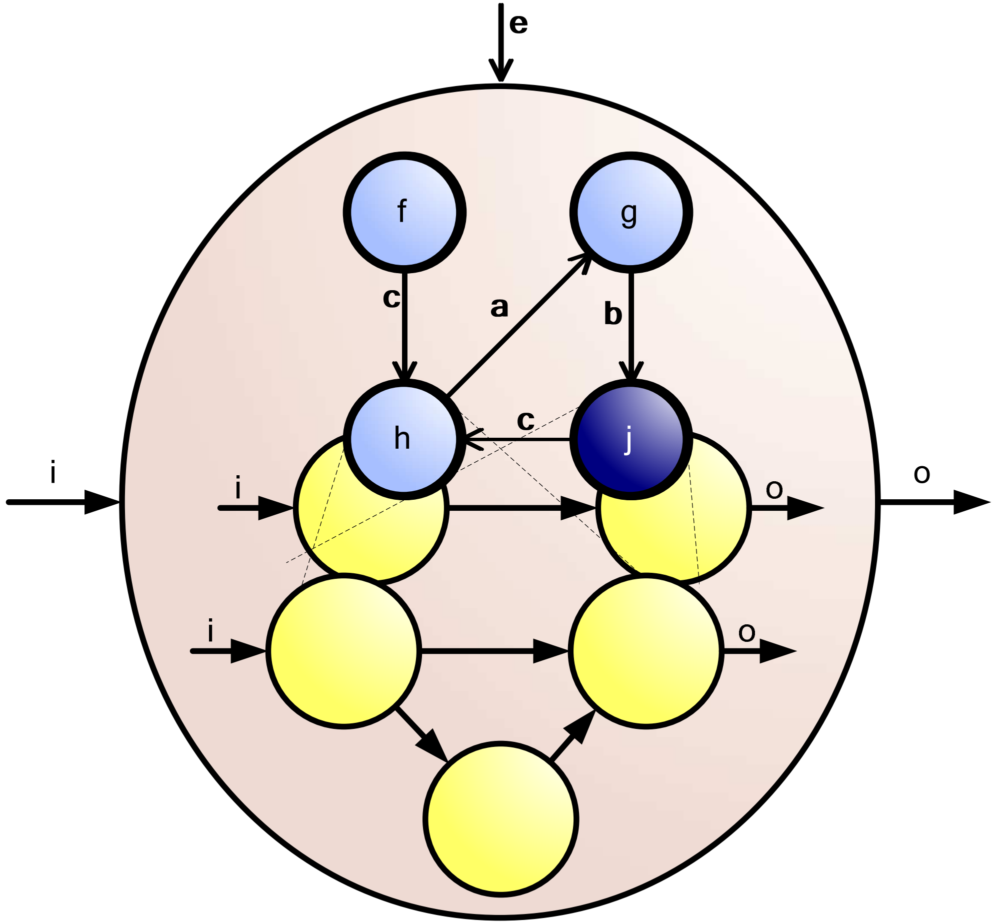
1. schedule enabled processes (in any fair way)
2. until (local) deadlock occurs
3. resolve deadlock if artificial by increasing smallest full FIFO

Reactive Process Networks

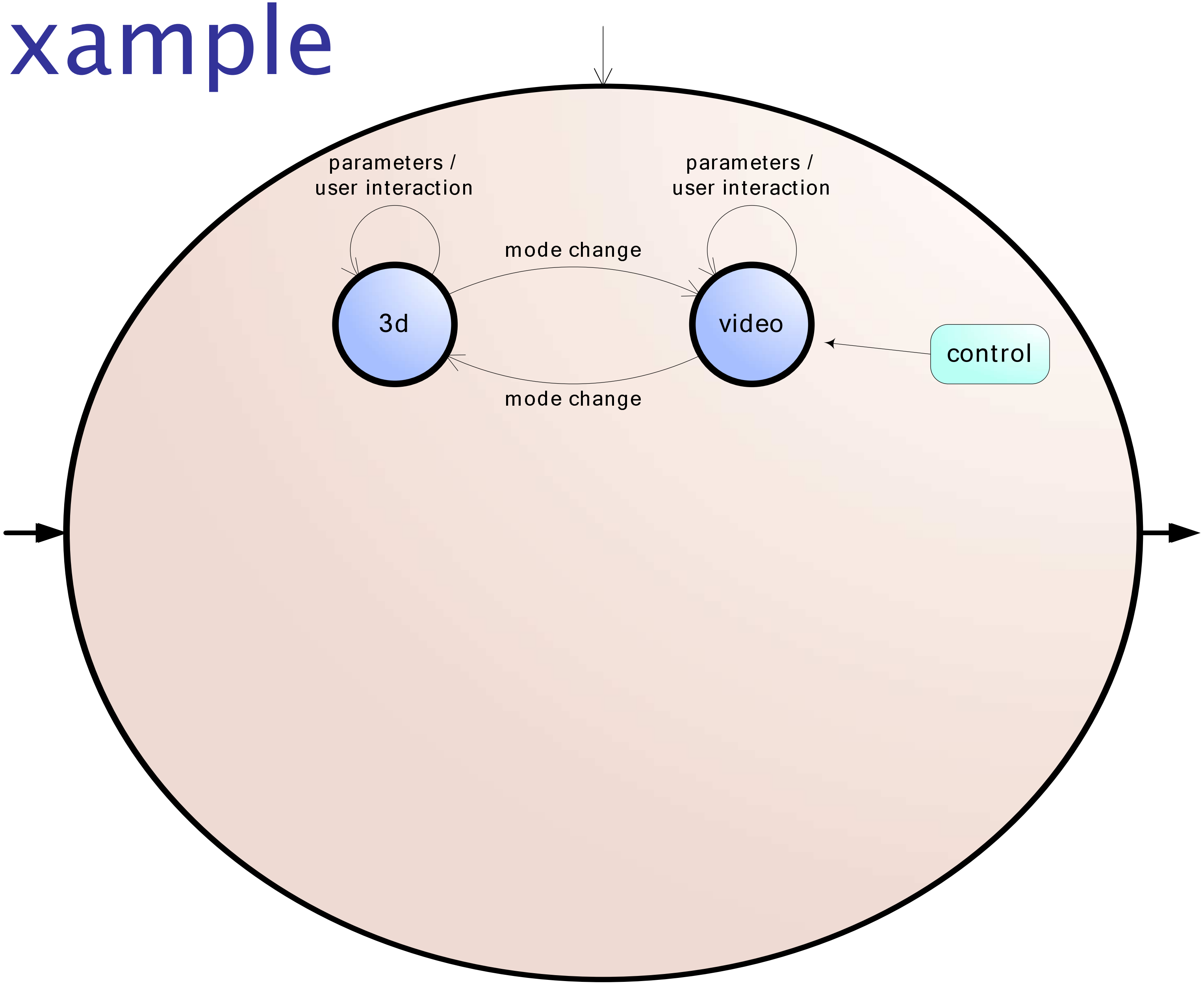
reactive behaviour and dataflow

- indeterminate behaviour
- non–functional input–output relations
- Brock–Ackerman anomaly
- ‘select’ primitive in YAPI
- predictability

the basic idea...

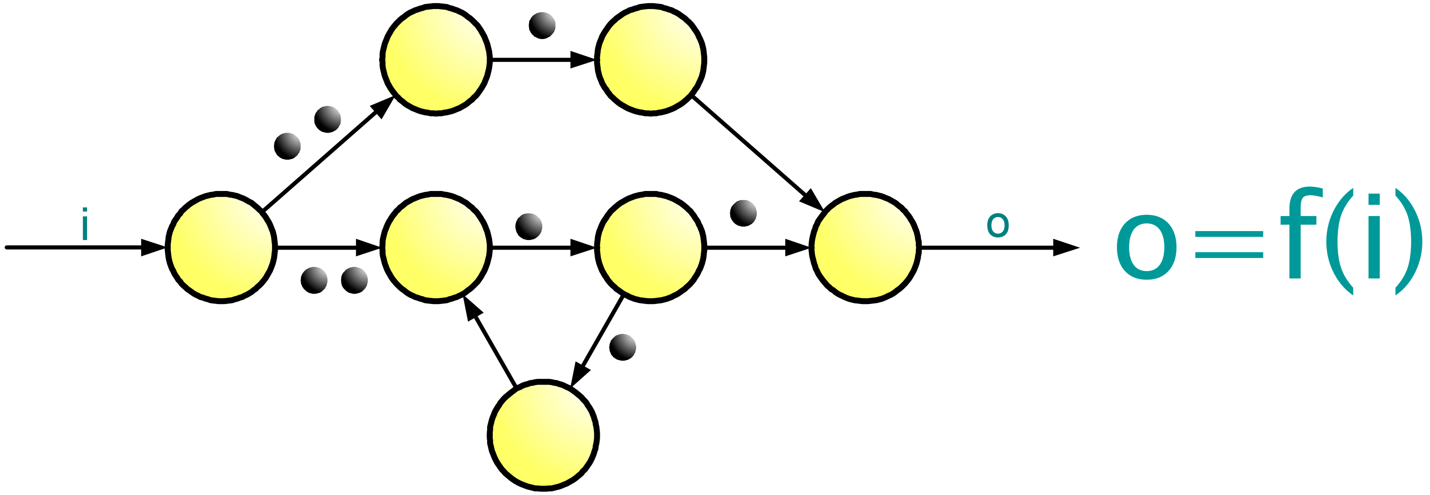


example


parameters /
user interaction
parameters /
user interaction
mode change
3d
video
control
mode change

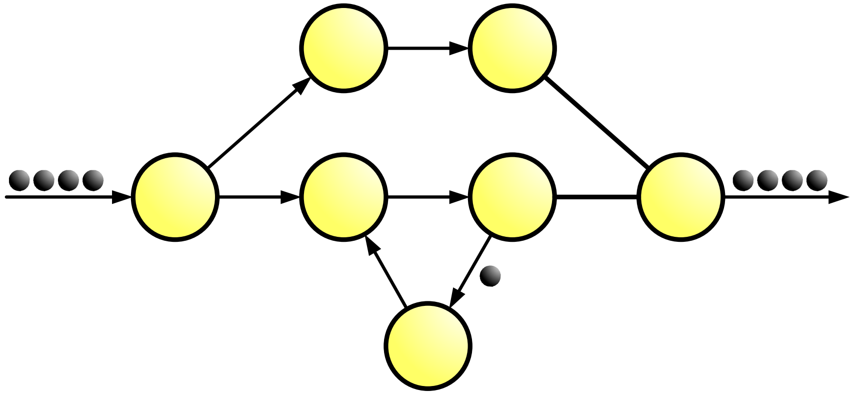
reconfiguration of streams

- PN computes functions on data streams
- conceptually, the response is immediate
- computation of the results takes time and is pipelined in practice
- **how to reconfigure with data in the pipeline?**



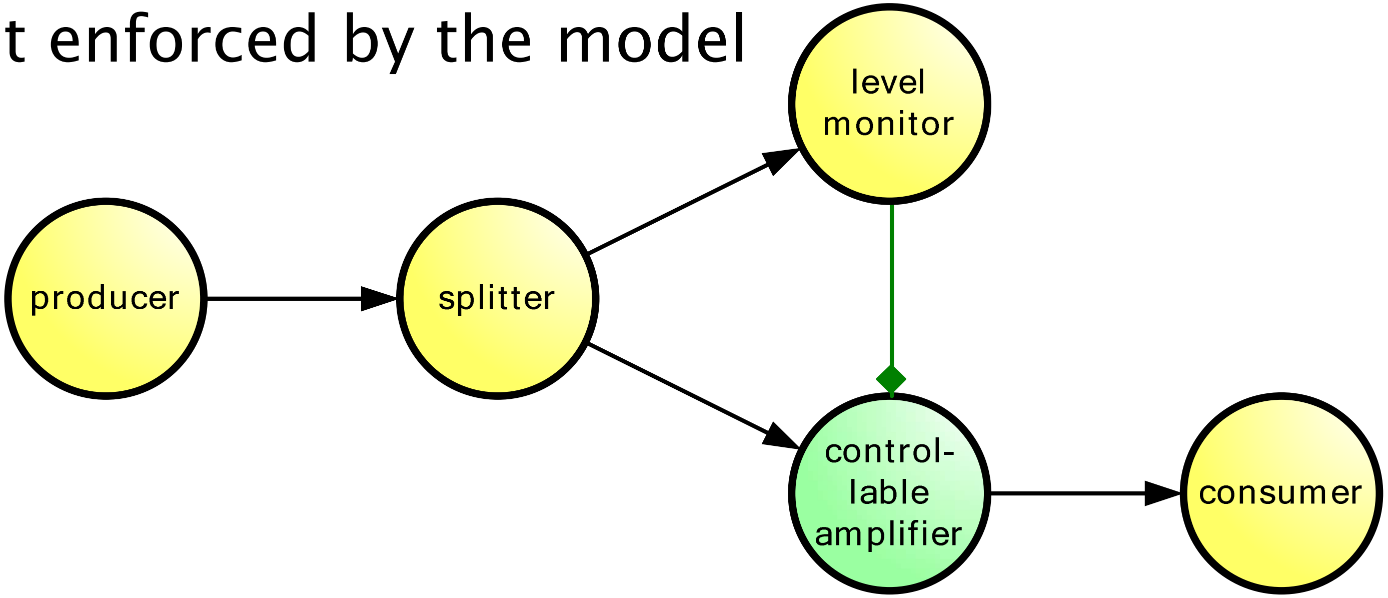
reconfiguration of streams

- reconfiguration should not have an effect on the outcome of the computation for input that has already been consumed
- reconfiguration should take place at ‘quiescent’ points
- but flushing the pipeline may be unaffordable



compositionality

- event handling encapsulated in a streaming component
- functional behaviour?
- this would lead to excessive synchronisation
- not enforced by the model



semantics of RPN

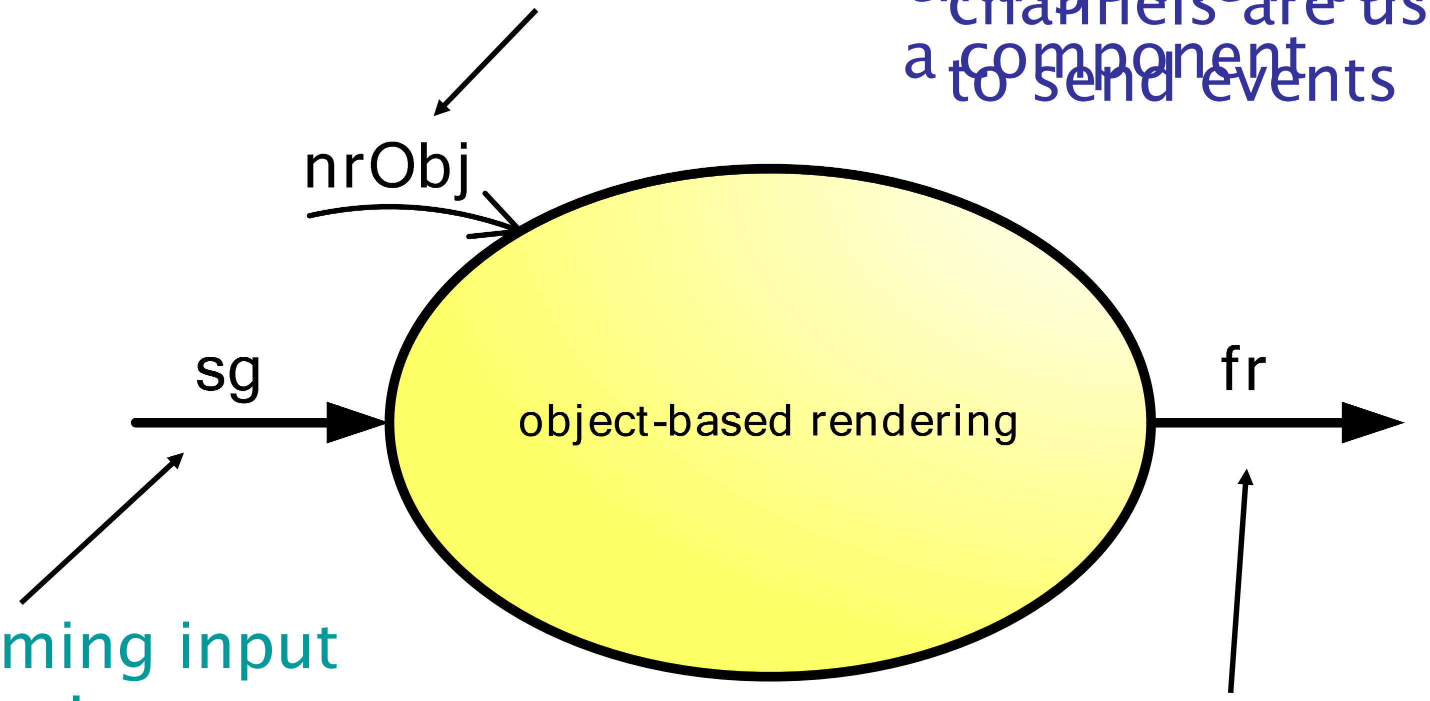
- denotational semantics as input/output relation doesn’t work (well) for indeterminate dataflow
- hence, we built an operational semantics as a Labelled Transition System using SOS rules
- hierarchical and compositional

interface

event input channels

reconfiguration cannot change the interface of a component

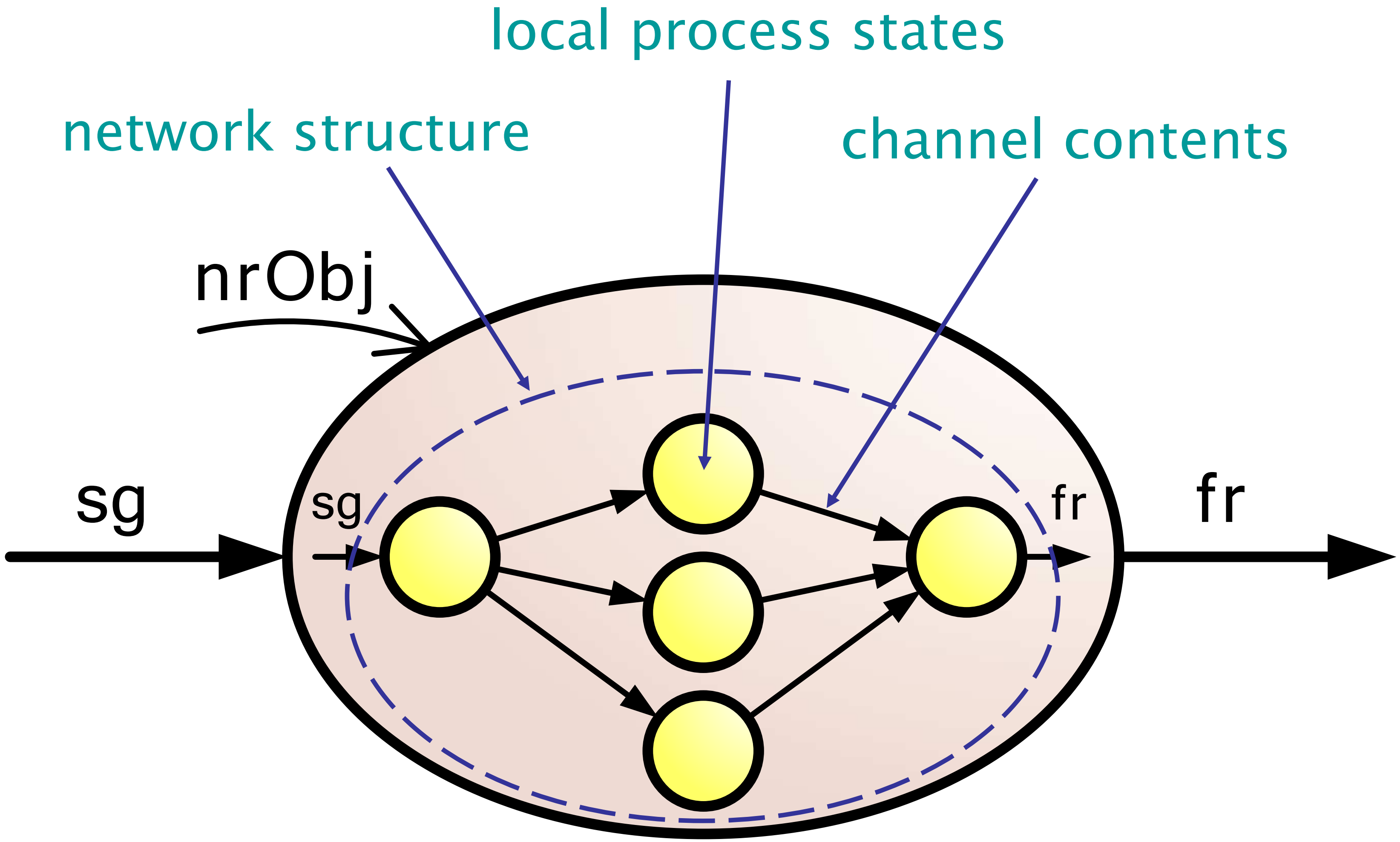
streaming output channels are used to send events



streaming input channels

streaming output channels

configurations

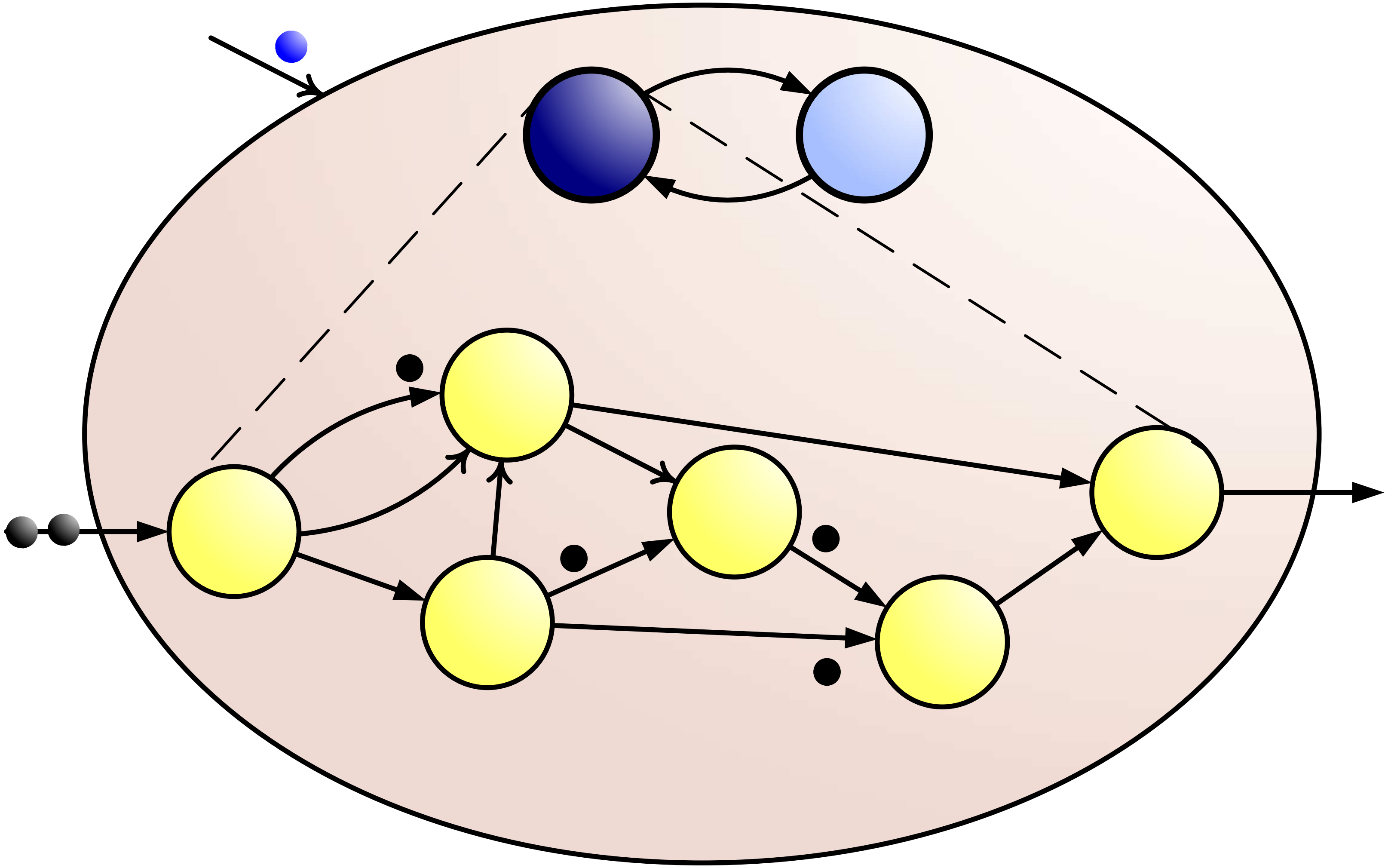


transitions

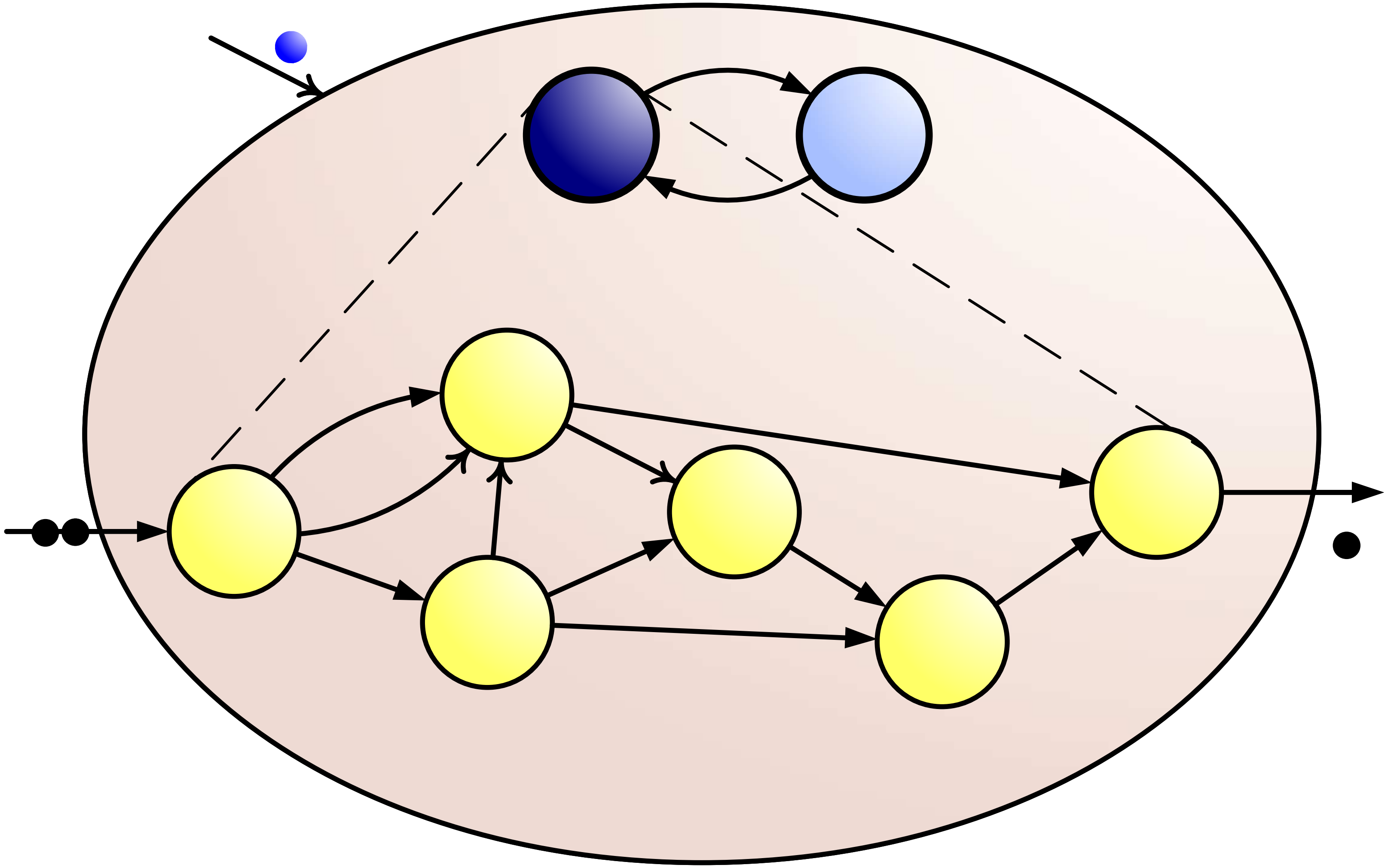
transitions from one configuration to the next caused by:

- reading tokens from streaming inputs
- writing tokens to streaming outputs
- reading tokens from event inputs, followed by a corresponding reconfiguration

operational semantics



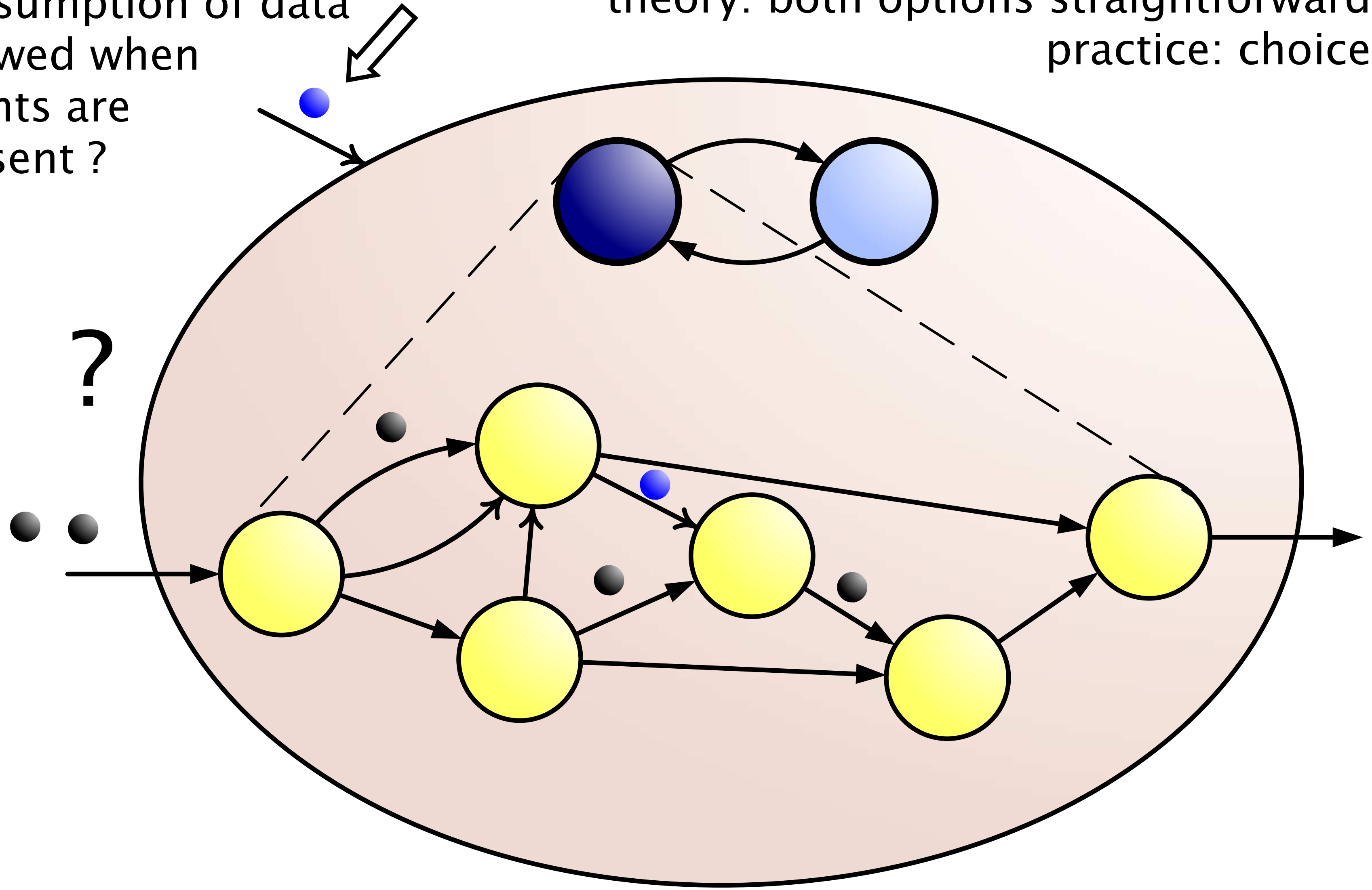
operational semantics



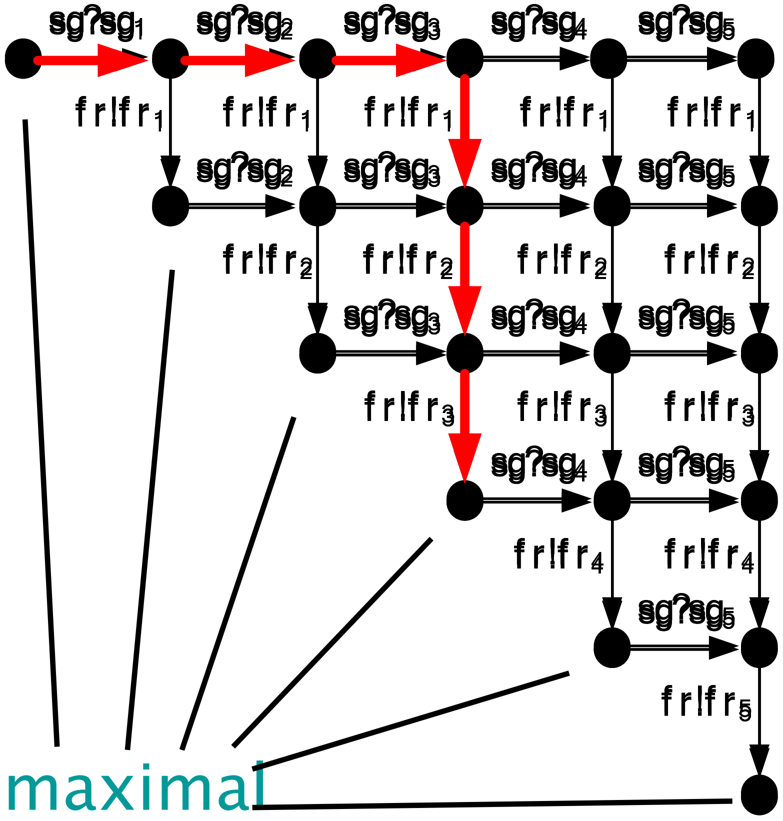
prioritizing events

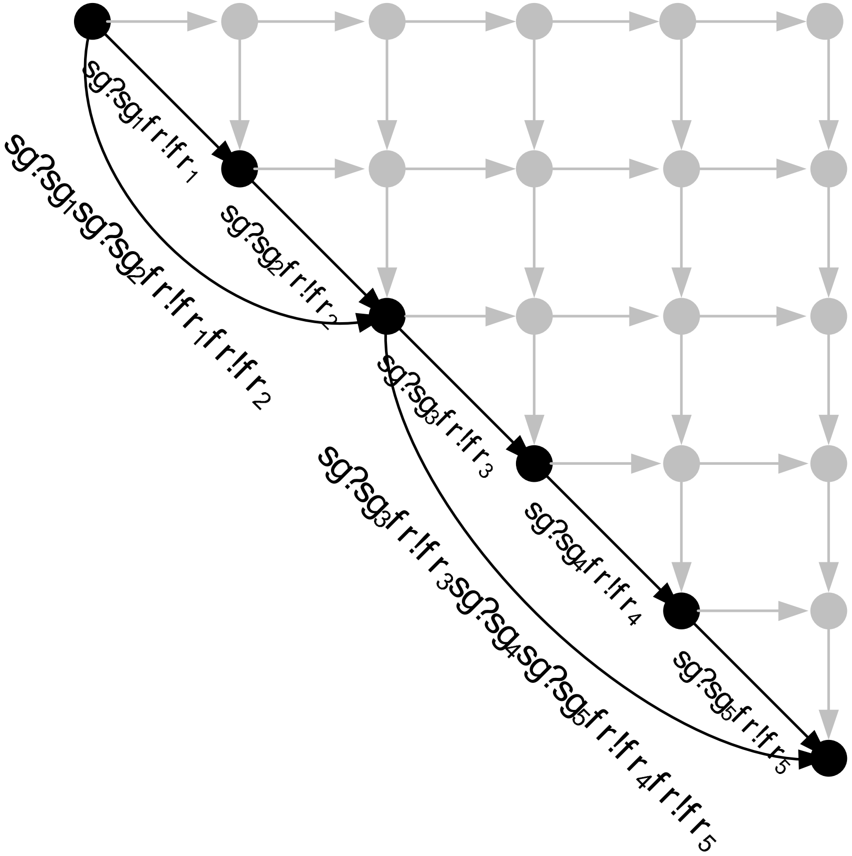
consumption of data allowed when events are present ?

theory: both options straightforward
practice: choice



operational semantics





prototype implementation

- based on YAPI, C++ implementation of KPN
- adds event input ports
- arriving events trigger functions
- runtime system ensures flushing of the component before calling event handler
- events have priority over consumption of new input data

open issues, future work

- case studies (ease of modelling, expressivity)
- distribution
- additional control over reconfiguration points
- timing prediction
  - reconfiguration/reaction times for events
- restricted models

  reactive BDF, reactive (C)SDF