SDL

Jian-Jia Chen
(slides are based on Peter Marwedel)
TU Dortmund,
Informatik 12

2016年 10 月 26 日

Models of computation considered in this course

Communication/ local computations	Shared memory	Message passing Synchronous | Asynchronous	
Undefined components	Plain text, use cases | (Message) sequence charts		
Communicating finite state machines	StateCharts		**SDL**
Data flow			Kahn networks, SDF
Petri nets		C/E nets, P/T nets, …	
Discrete event (DE) model	VHDL*, Verilog*, SystemC*, …	Only experimental systems, e.g. distributed DE in Ptolemy	
Von Neumann model	C, C++, Java	C, C++, Java with libraries CSP, ADA |	

* Based on implementation of VHDL, Verilog..

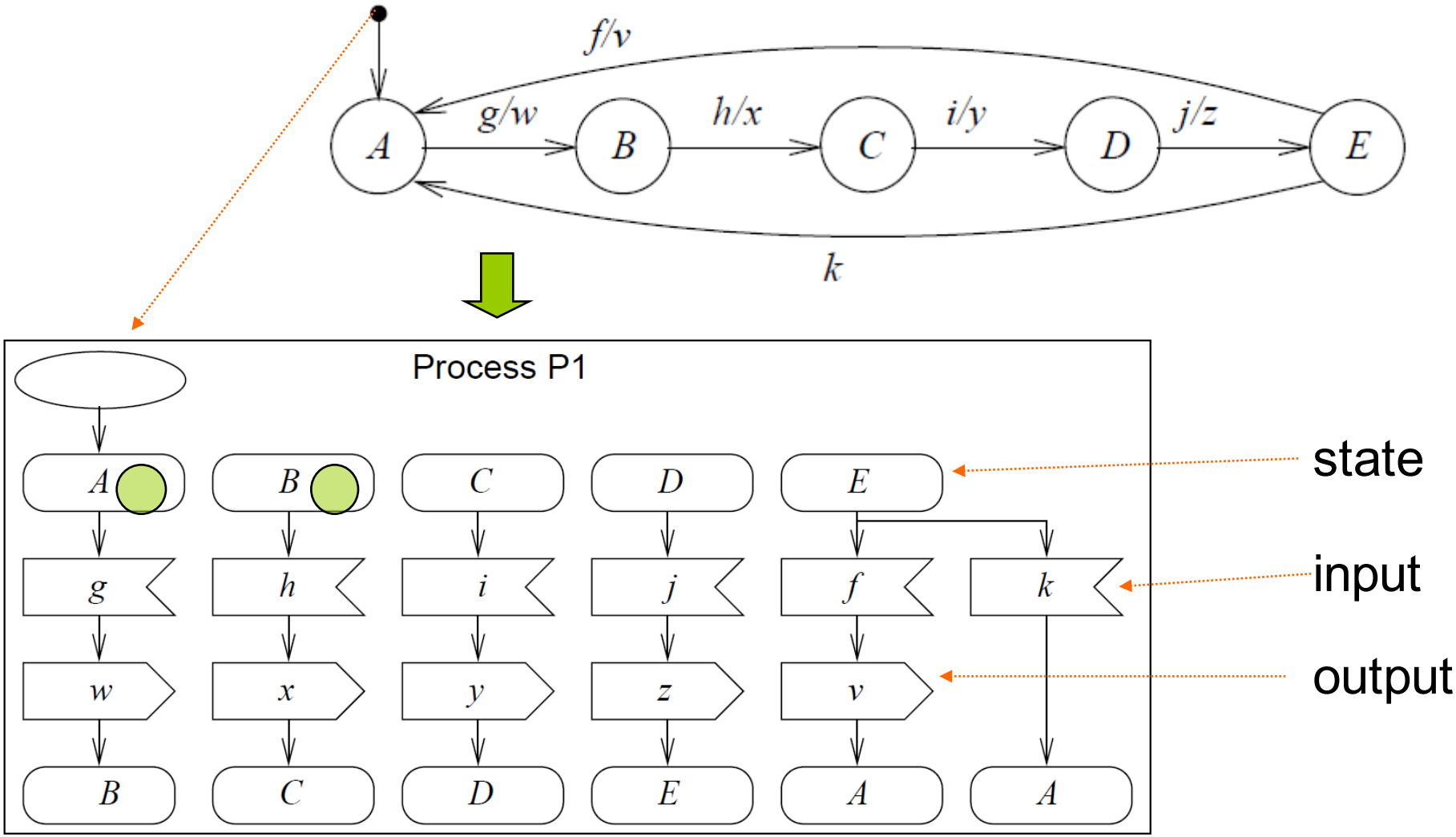
SDL

- SDL used here as a (prominent) example of a model of computation based on **asynchronous message passing communication**.

☞ SDL is appropriate also for distributed systems.

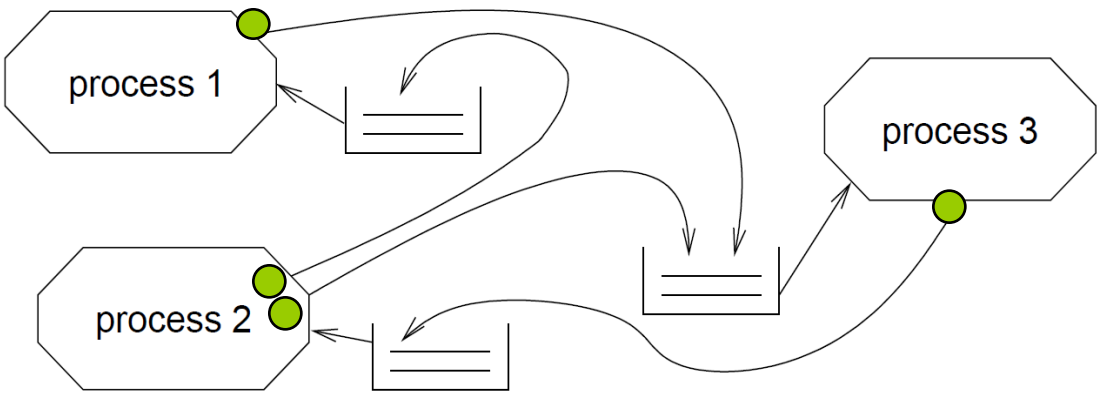
- Just like StateCharts, it is based on the CFSM model of computation; each FSM is called a **process**.

SDL-representation of FSMs/processes



Communication among SDL-FSMs

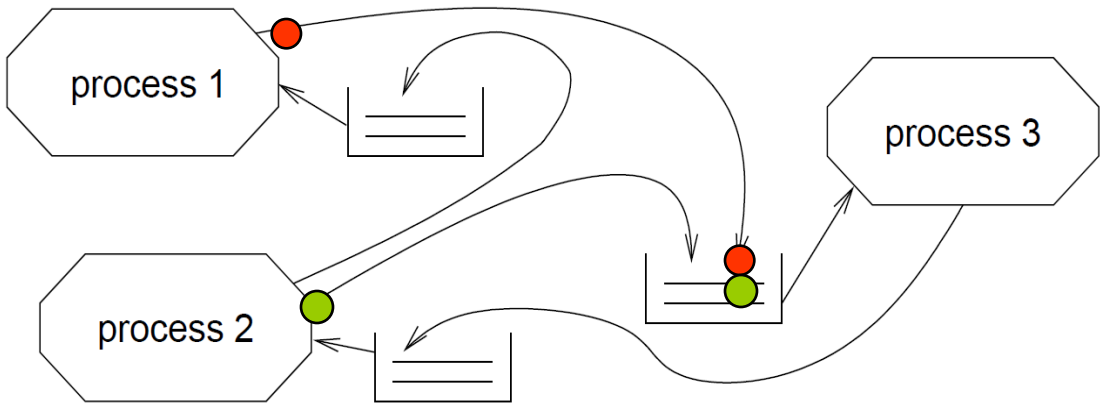
Communication is based on message-passing of *signals* (=inputs+outputs), assuming a **potentially indefinitely large FIFO-queue**.



- Each process fetches next signal from FIFO,
- checks if signal enables transition,
- if yes: transition takes place,
- if no: signal is ignored (exception: SAVE-mechanism).
- Implementation requires bound for the maximum length of FIFOs

Determinate?

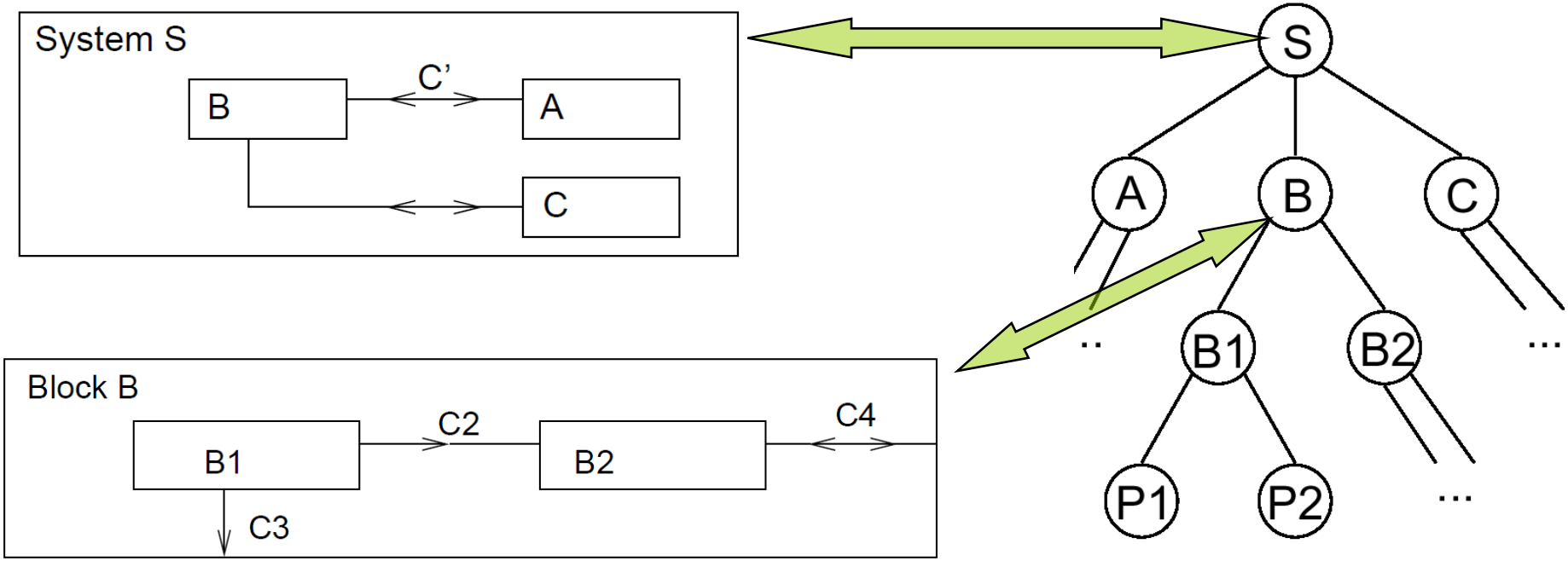
Let signals be arriving at FIFO at the same time:
☞Order in which they are stored, is unknown:



All orders are legal: ☞simulators can show different behaviors for the same input, all of which are correct.

Hierarchy in SDL

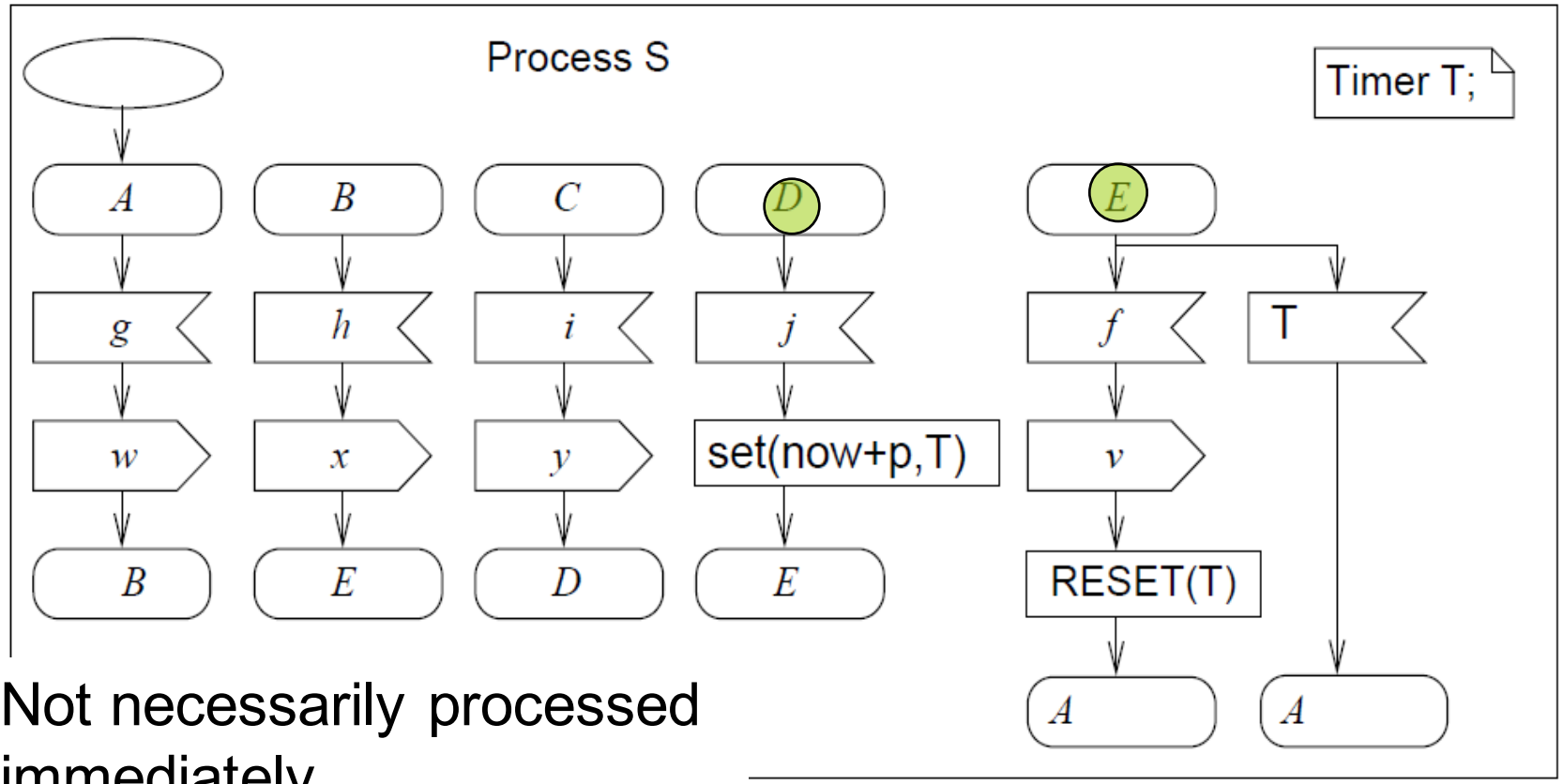
Process interaction diagrams can be included in **blocks.** The root block is called **system.**



Processes cannot contain other processes, unlike in StateCharts.

Timers

Timers can be declared locally. Elapsed timers put signal into queue. RESET removes timer (also from FIFO-queue).



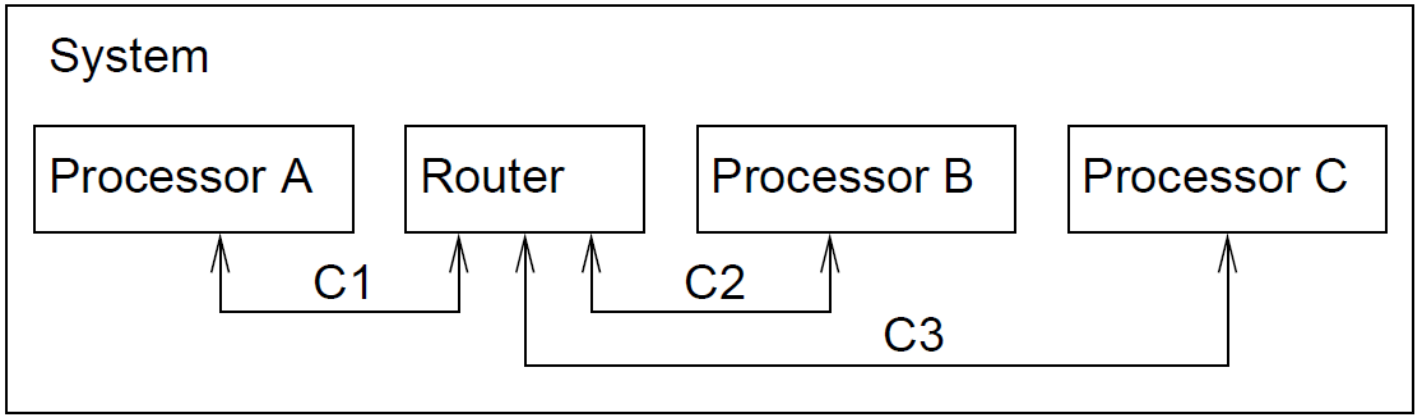
Not necessarily processed immediately.

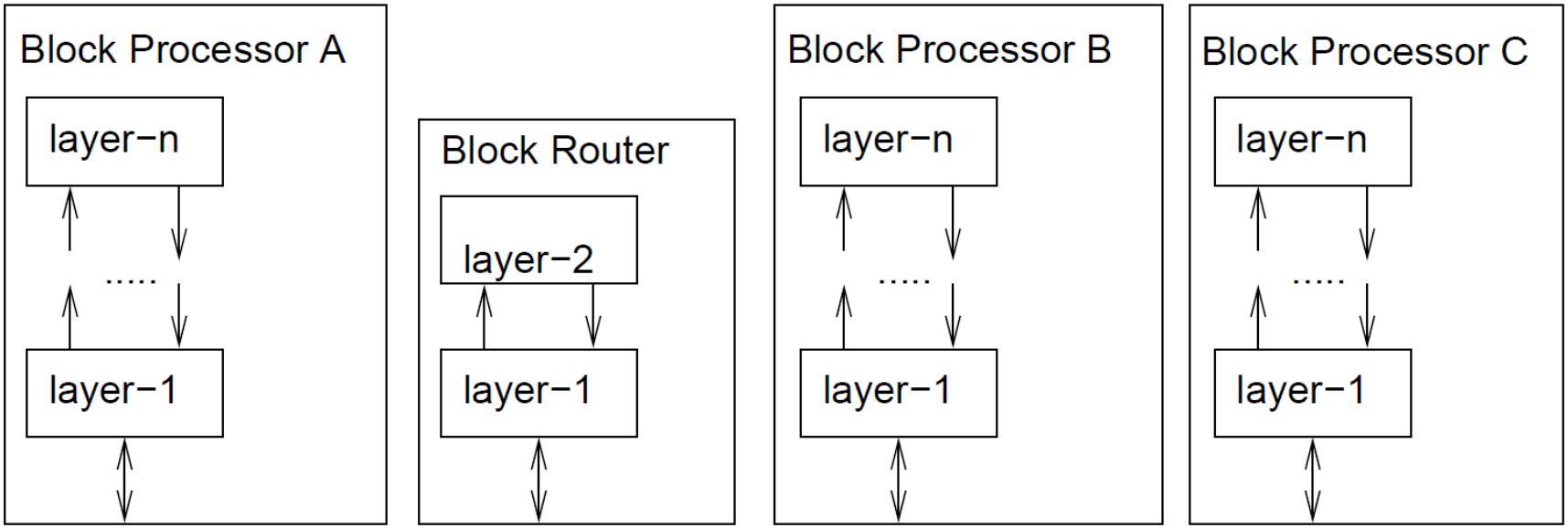
Additional language elements

SDL includes a number of additional language elements, like

- procedures
- creation and termination of processes
- advanced description of data
- More features added for SDL-2000

Application: description of network protocols





History

- Dates back to early 1970s,
- Formal semantics defined in the late 1980s,
- Defined by ITU (International Telecommunication Union):
  Z.100 recommendation in 1980
  Updates in 1984, 1988, 1992, 1996 and 1999
- SDL-2000 a significant update (not well accepted)
- Becoming less popular

Data flow models

Jian-Jia Chen
(Slides are based on
Peter Marwedel)
TU Dortmund,
Informatik 12

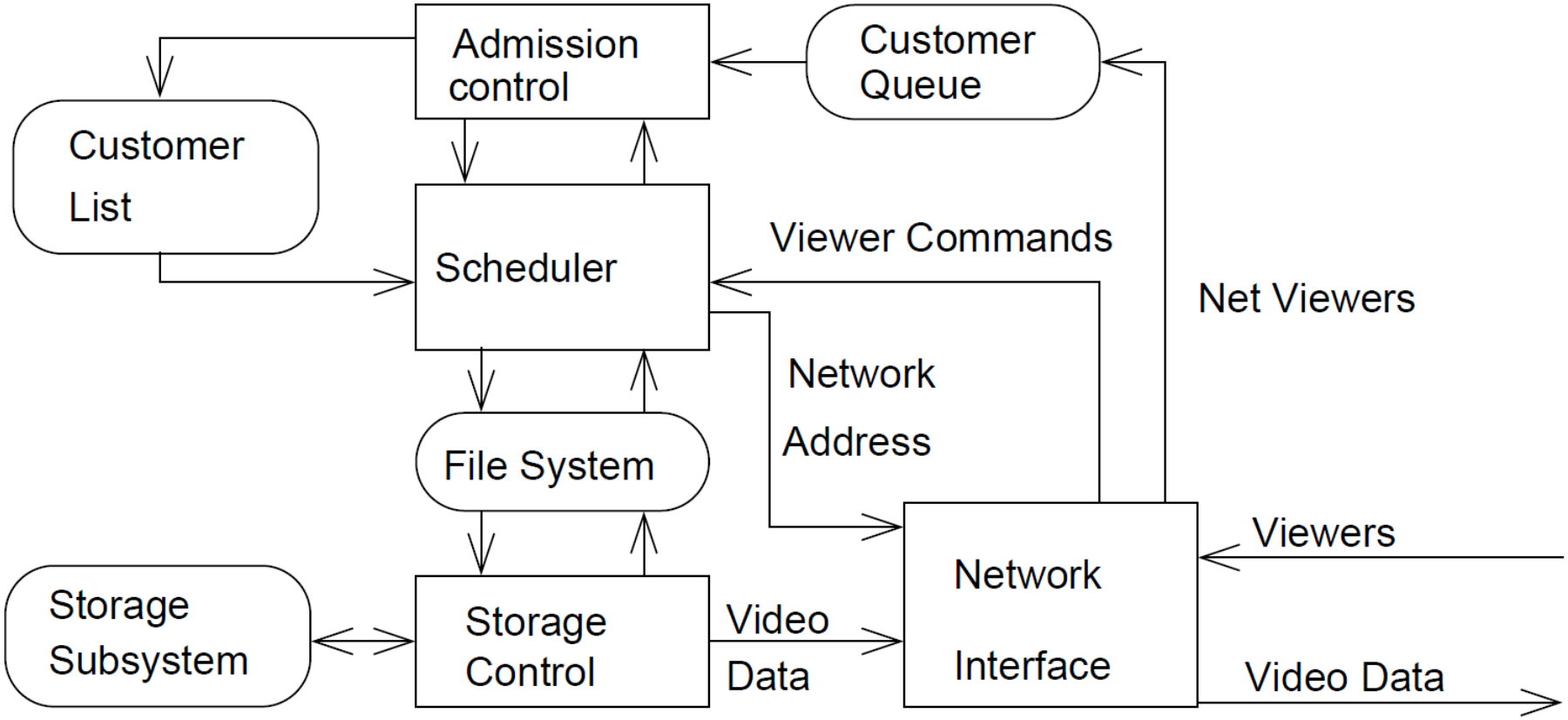
2016年 10 月 26 日

Models of computation considered in this course

Communication/ local computations	Shared memory	Message passing Synchronous | Asynchronous	
Undefined components	Plain text, use cases | (Message) sequence charts		
Communicating finite state machines	StateCharts		SDL
Data flow			Kahn networks, SDF
Petri nets		C/E nets, P/T nets, …	
Discrete event (DE) model	VHDL, Verilog, SystemC, …	Only experimental systems, e.g. distributed DE in Ptolemy	
Von Neumann model	C, C++, Java	C, C++, Java with libraries CSP, ADA |	

Data flow as a "natural" model of applications

Example: Video on demand system



www.ece.ubc.ca/~irenek/techpaps/vod/vod.html

Data flow modeling

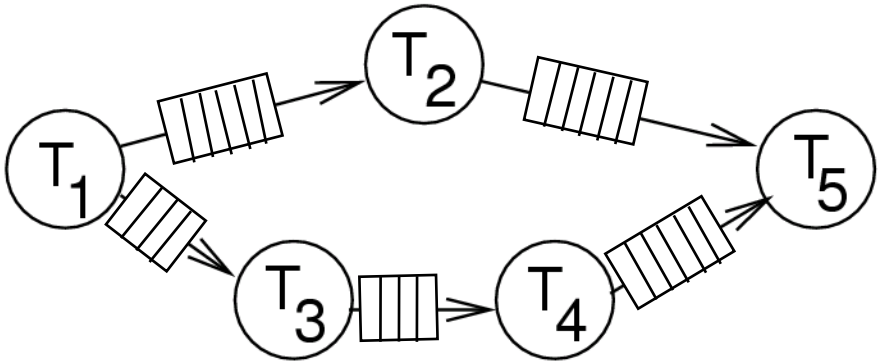
Definition: Data flow modeling is … "*the process of identifying, modeling and documenting how data moves around an information system.*
Data flow modeling examines

- *processes (activities that transform data from one form to another),*
- *data stores (the holding areas for data),*
- *external entities (what sends data into a system or receives data from a system, and*
- *data flows (routes by which data can flow)*".

[Wikipedia: Structured systems analysis and design method. *http://en.wikipedia.org/wiki/Structured Systems Analysis and Design Methodology,* 2010 (formatting added)].

Kahn process networks (KPN)

- Each component is modeled as a program/task/process
- Communication is by FIFOs
- Only one sender and one receiver per FIFO
  ☞ no SDL-like conflicts at FIFOs
- No overflow considered
  ☞ writes never have to wait,
  ☞ reads wait if FIFO is empty.

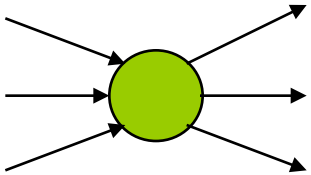


Example

```
Process f(in int u, in int v, out int w){
  int i; bool b = true;
  for (;;) {
   i= b ? read(u) : read(v);
                    //read returns next token in FIFO, waits if empty
   send (i,w);   //writes a token into a FIFO w/o blocking
   b = !b;
   }
```

Properties of Kahn process networks

- Communication is only via channels (no shared variables)
- Mapping from ≥1 input channel to ≥1 output channel possible;.

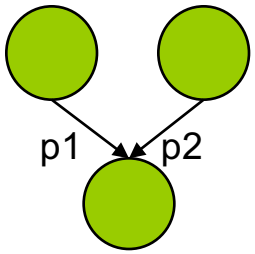


Key Property of Kahn process networks

- Processes have to commit to wait for data **from a particular port**;

  - A process cannot check for available data before attempting a read (wait).

    ~~if nonempty(p1) then read(p1)~~
    ~~else if nonempty(p2) then read(p2);~~



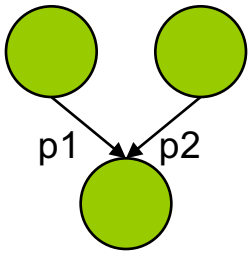
  - A process cannot wait for data for >1 port at a time.

    ~~read(p1|p2);~~

Key beauty of KPNs

- Therefore, the order of reads does not depend on the arrival time (but may depend on data).

- Therefore, Kahn process networks are determinate (!); for a given input, the result will be always the same, regardless of the speed of the nodes.



- Many applications in embedded system design: Any combination of fast and slow simulation & hardware prototypes always gives the same result.

Computational power and analyzability

- KPNs are Turing-complete
  - We can implement a Turing machine as a KPN
  - However, the difficulty of Turing machines is also inherited
  - Analyzing the maximum FIFO size is “undecidable”
  - Analyzing whether it terminates is “undecidable”
  - Deadlock may be possible
- Timing is not specified
- Number of processes is static (cannot change)

More information about KPNs

- http://ls12-www.cs.tu-dortmund.de/teaching/download/levi/index.html: Animation
- See also S. Edwards: http://www.cs.columbia.edu/~sedwards/classes/2001/w4995-02/presentations/dataflow.ppt

SDF

Less computationally powerful, but easier to analyze:

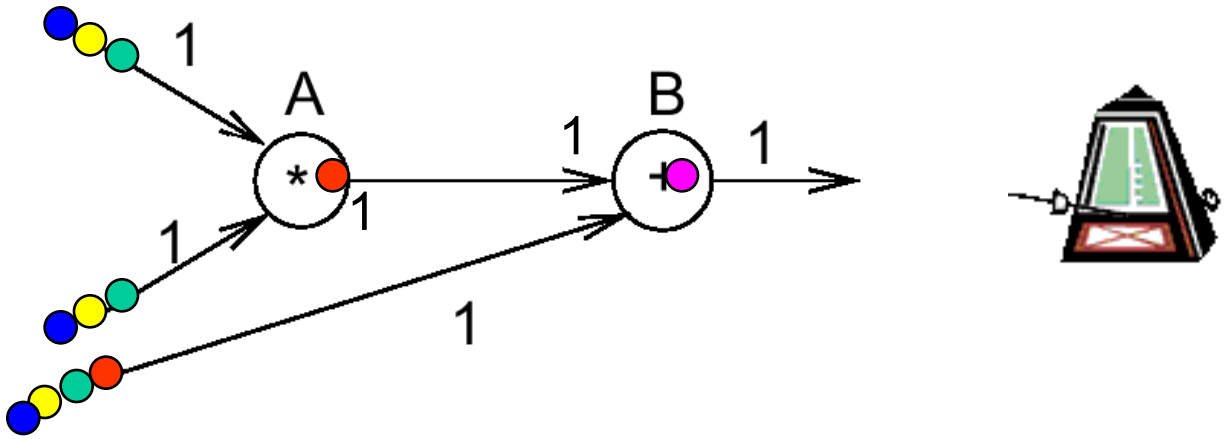
Synchronous data flow (SDF).

- **Synchronous**
  - all the tokens are consumed synchronously
  - one way to implement is to have a global clock controlling "firing" of nodes

- Again using ***a***synchronous message passing
  = tasks do not have to wait until output is accepted.

(Homogeneous-) Synchronous data flow (SDF)

- Nodes are called ***actors.***
- Actors are ***ready***, if the necessary number of input tokens exists and if enough buffer space at the output exists
- Ready actors ***can fire*** (be executed).

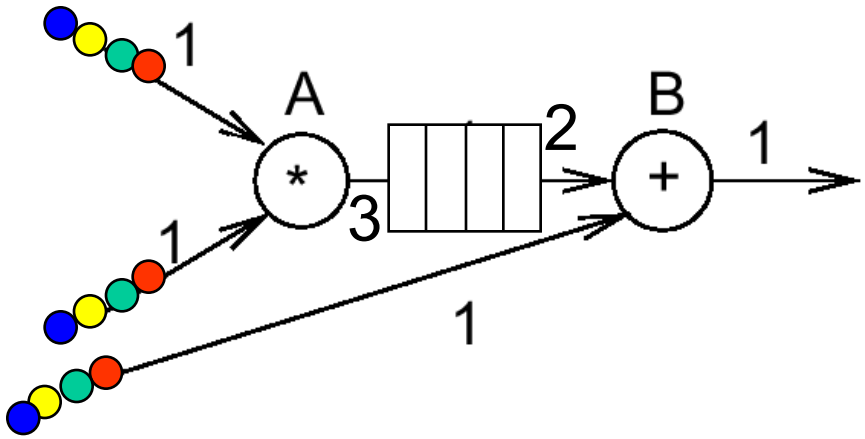


- Execution takes a ***fixed, known time***.

Actually, this is a case of ***homogeneous*** synchronous data flow models (HSDF): # of tokens per firing the same. ☞

(Non-homogeneous-) Synchronous data flow (SDF) (1)

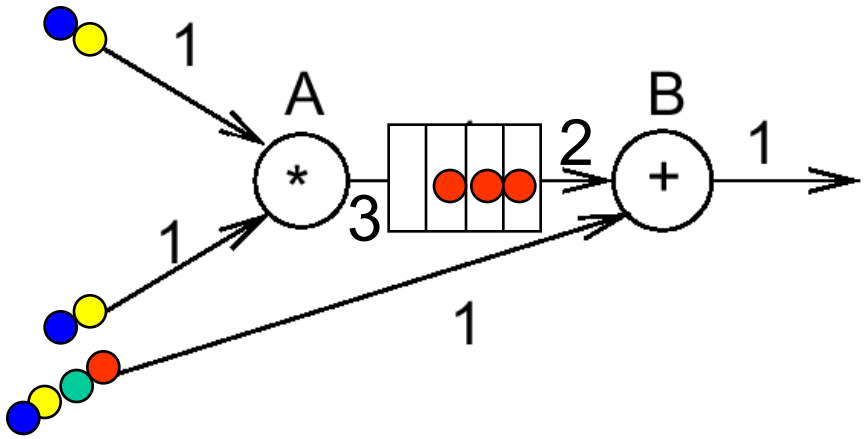
In the general case, a number of tokens can be produced/ consumed per firing



A ready, can fire (does not have to)

(Non-homogeneous-) Synchronous data flow (SDF) (2)

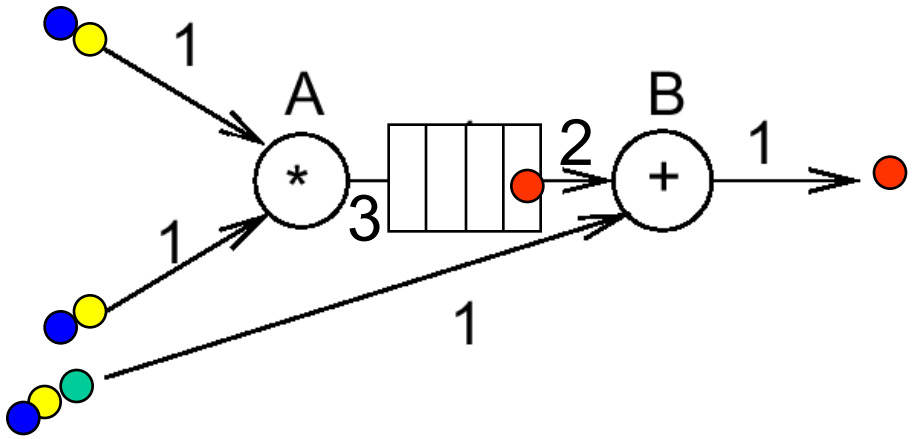
In the general case, a number of tokens can be produced/ consumed per firing



B ready, can fire

(Non-homogeneous-) Synchronous data flow (SDF) (3)

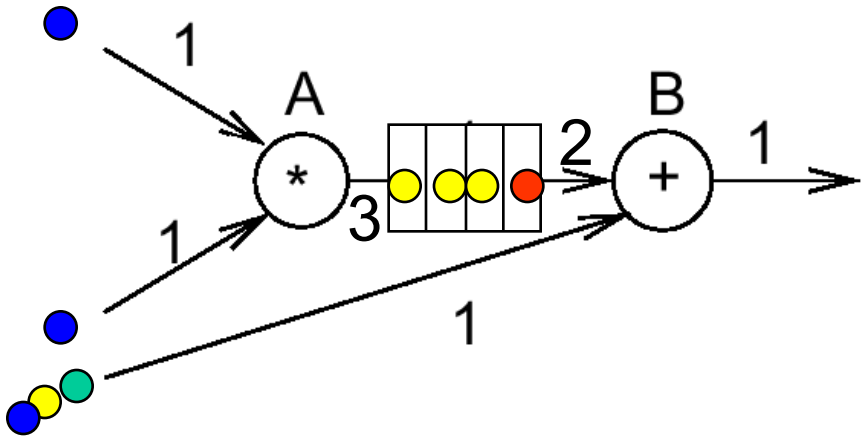
In the general case, a number of tokens can be produced/ consumed per firing



A ready, can fire

(Non-homogeneous-) Synchronous data flow (SDF) (4)

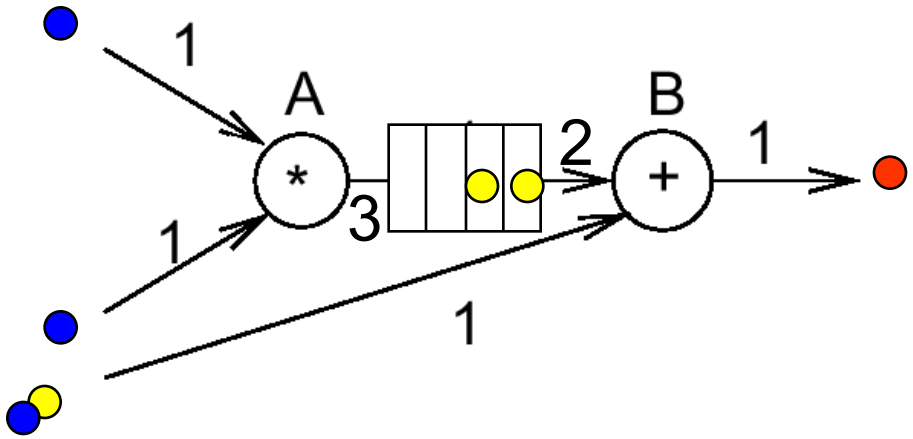
In the general case, a number of tokens can be produced/ consumed per firing



B ready, can fire

(Non-homogeneous-) Synchronous data flow (SDF) (5)

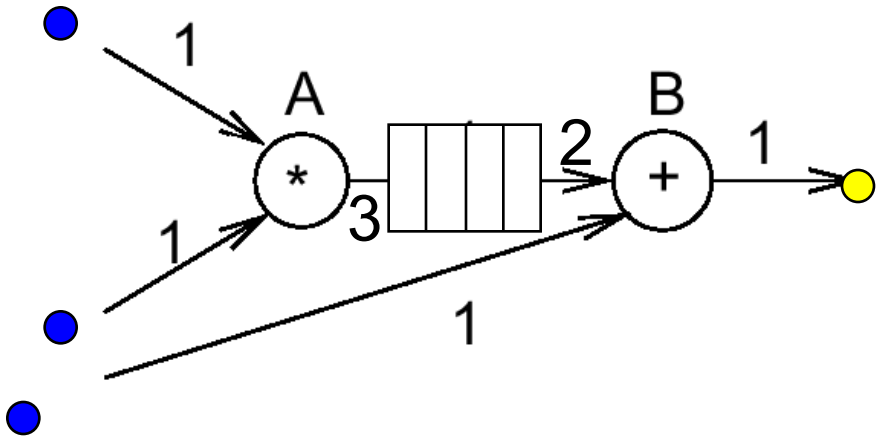
In the general case, a number of tokens can be produced/ consumed per firing



B ready, can fire

(Non-homogeneous-) Synchronous data flow (SDF) (6)

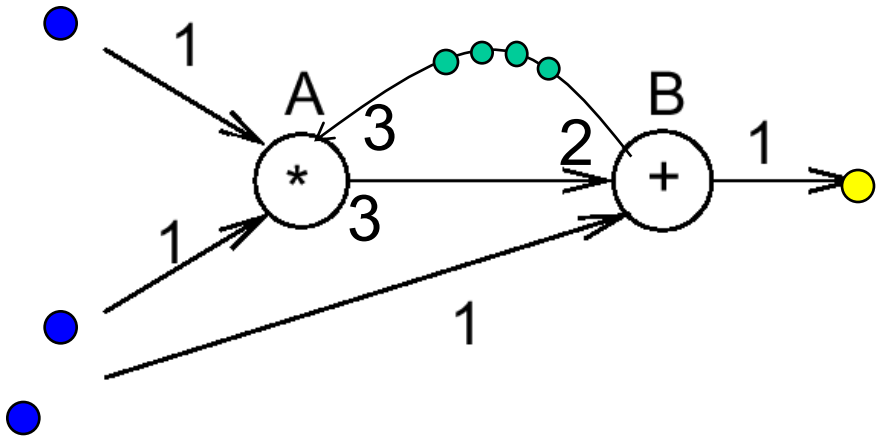
In the general case, a number of tokens can be produced/ consumed per firing



A ready, can fire

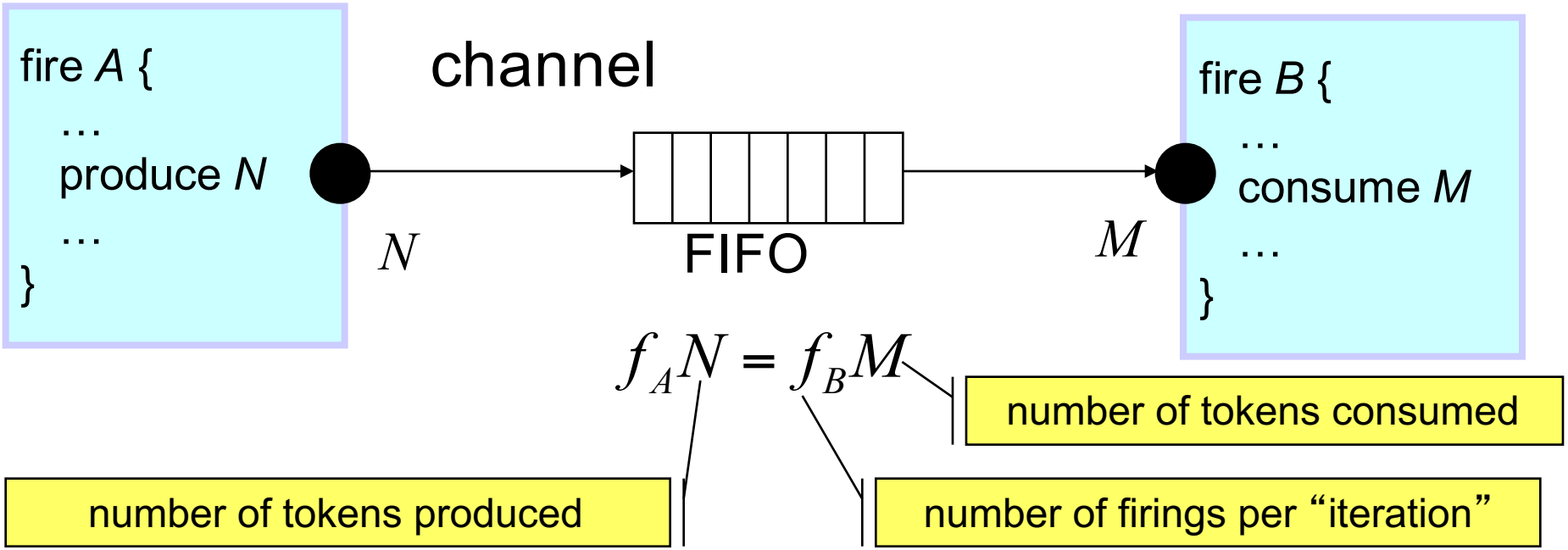
Actual modeling of buffer capacity

The capacity of buffers can be modeled easier:
as a ***backward*** edge where (initial number of tokens = buffer capacity).



Firing rate depends on # of tokens …

Multi-rate models & balance equations (one for each channel)



Determinate

Decidable:

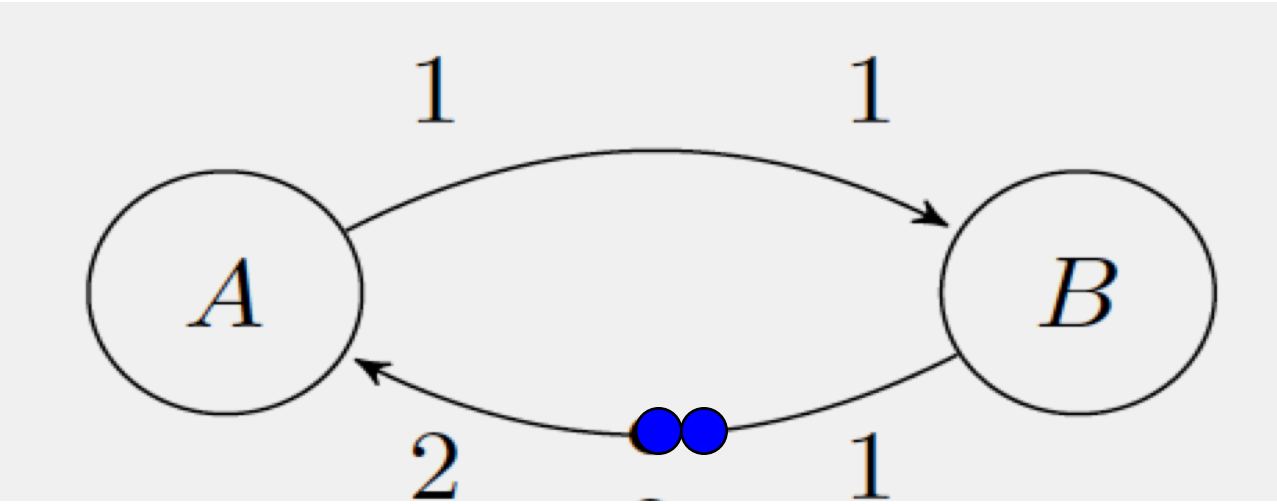
- buffer memory requirements
- deadlock

Schedulable statically

Adopted from: ptolemy.eecs.berkeley.edu/presentations/03/streamingEAL.ppt

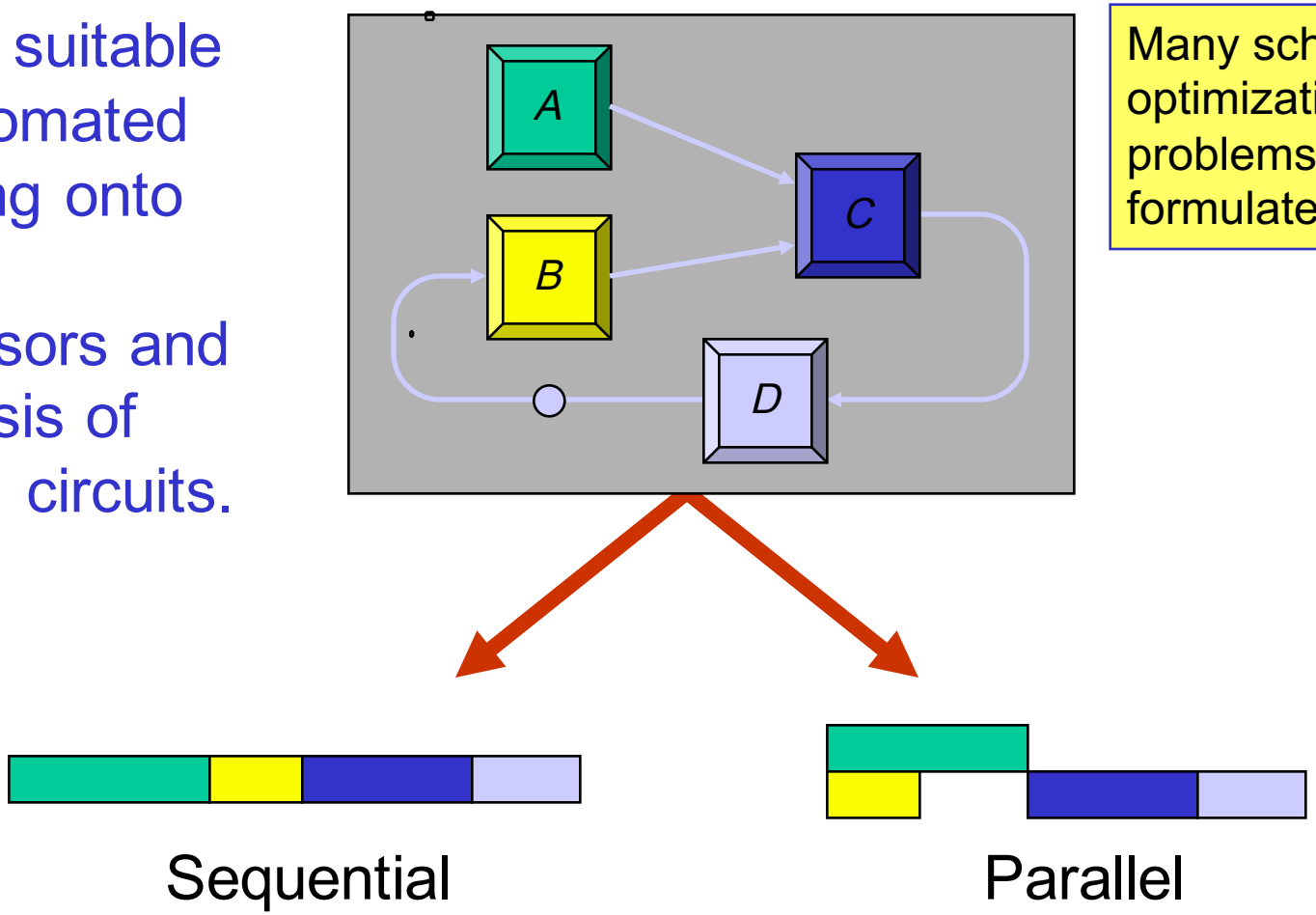
On-site Exercise: Behaviour Deadlock in SDF

Is it possible to have behavior deadlocks (all the actors are waiting to fire?) in SDF?



Parallel Scheduling of SDF Models

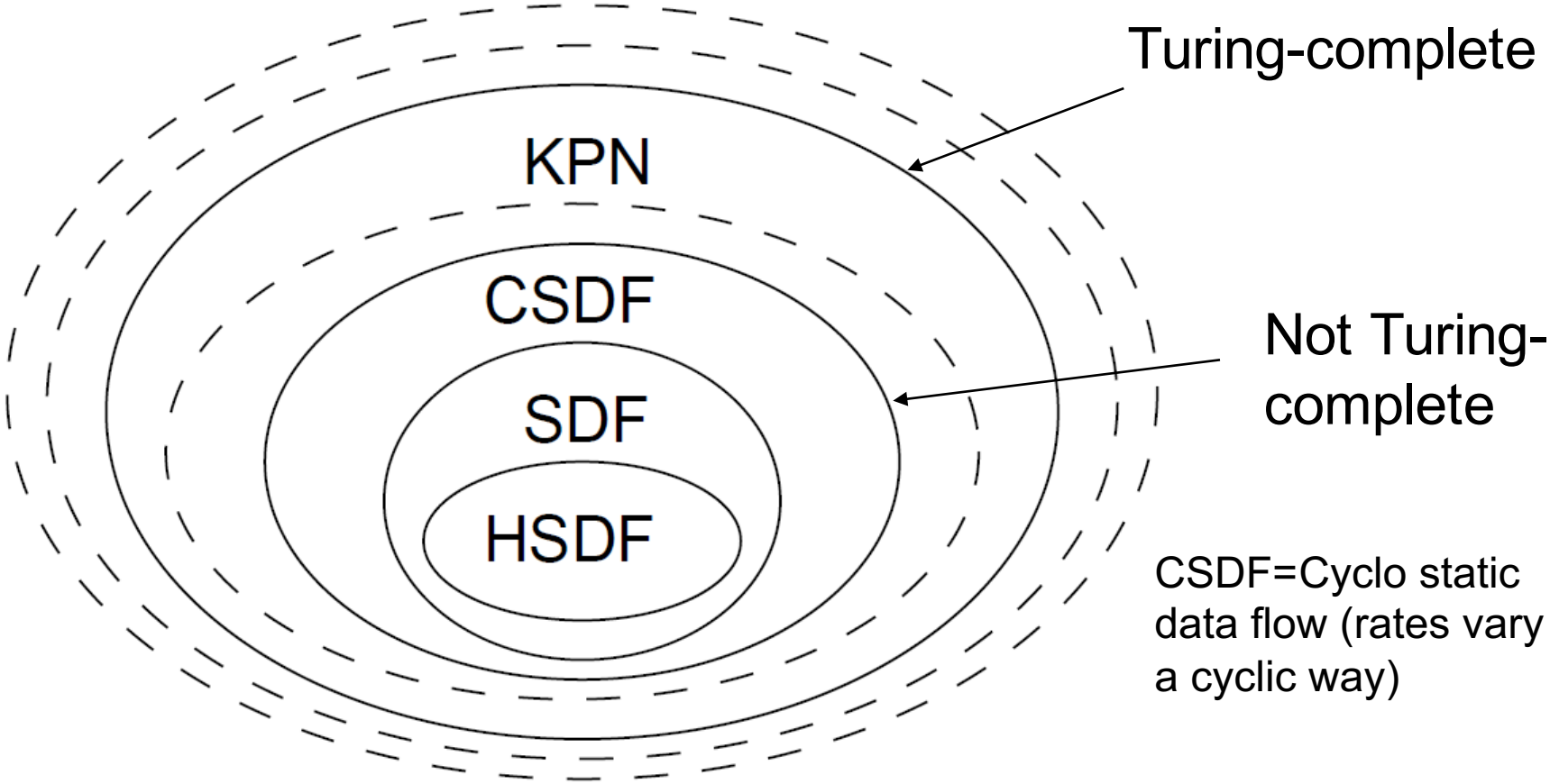
SDF is suitable for automated mapping onto parallel processors and synthesis of parallel circuits.



Many scheduling optimization problems can be formulated.

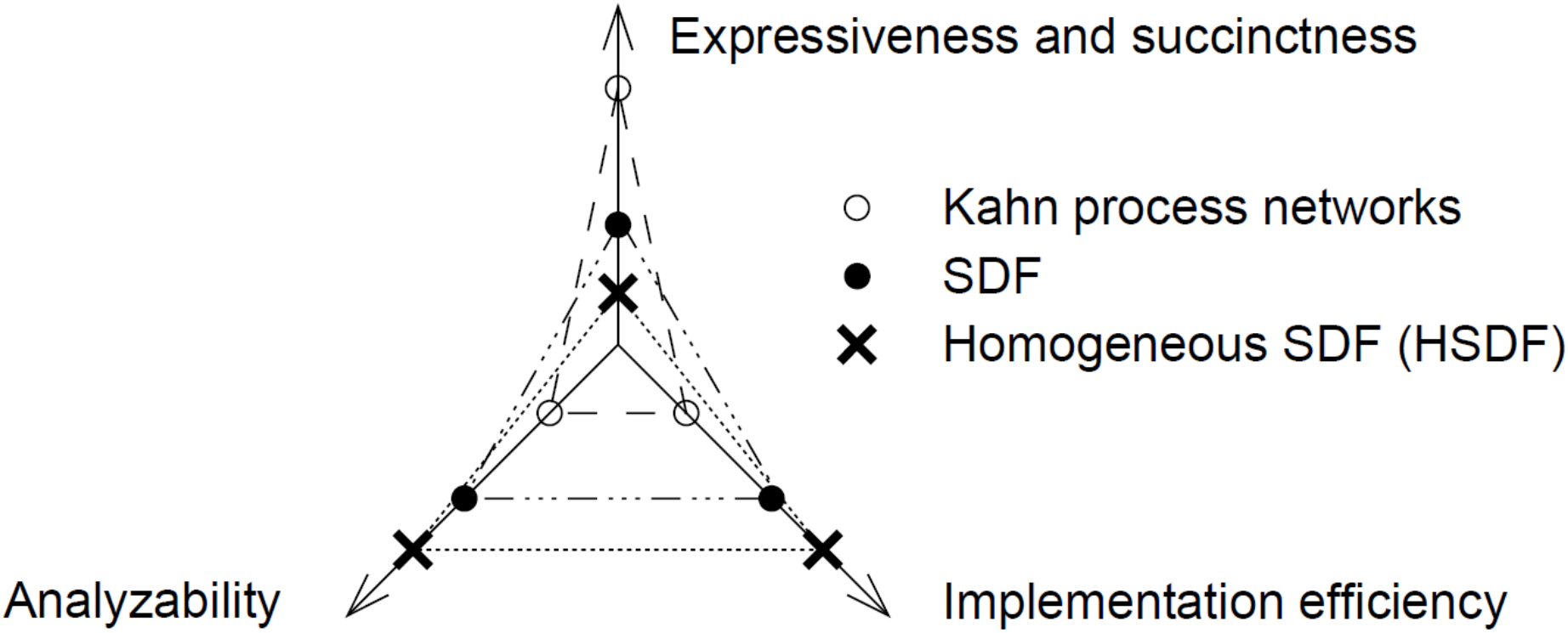
Source: ptolemy.eecs.berkeley.edu/presentations/03/streamingEAL.ppt

Expressiveness of data flow MoCs



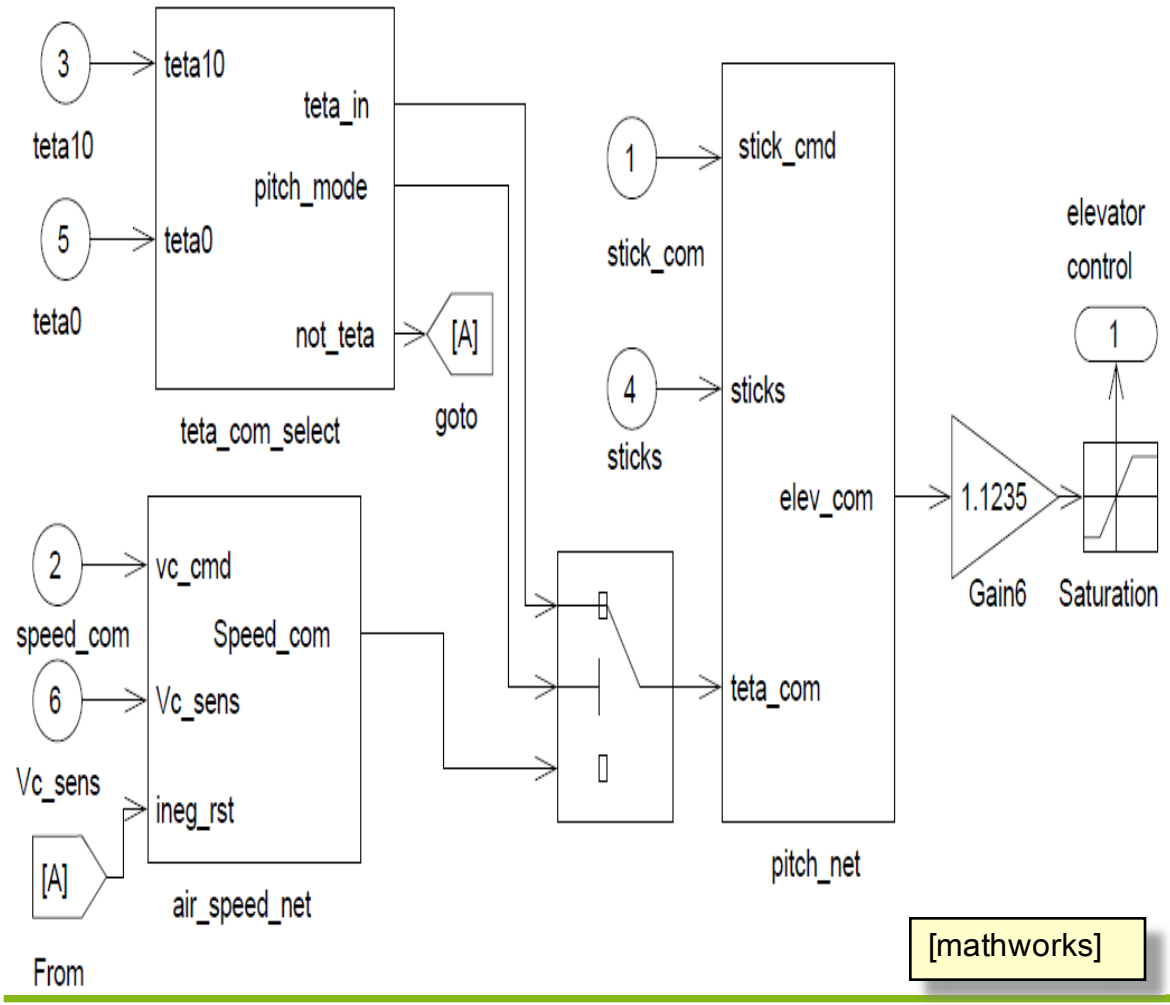
[S. Stuijk, 2007]

The expressiveness/analyzability conflict



[S. Stuijk, 2007]

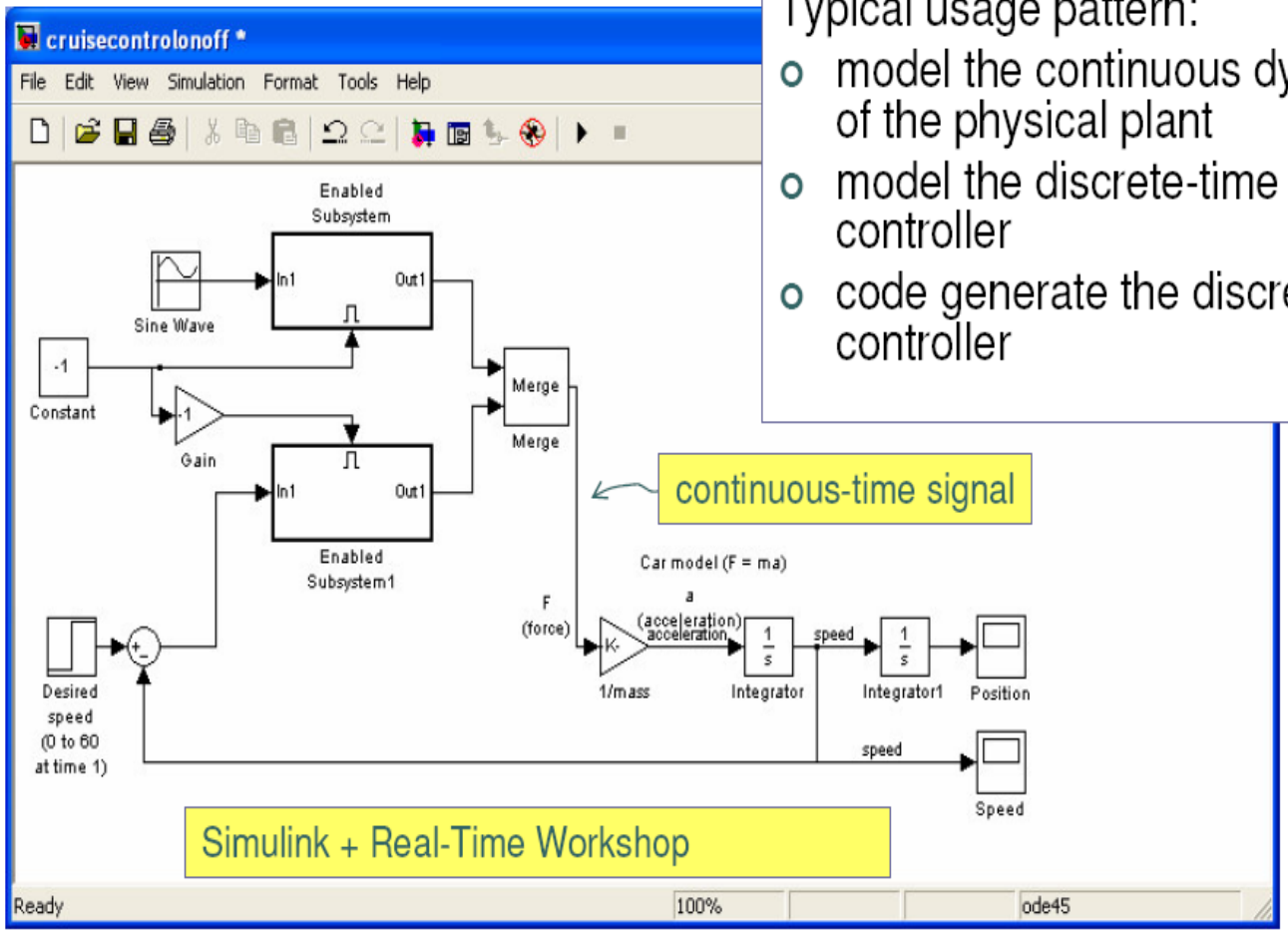
Similar MoC: Simulink
- example -



[mathworks]

Semantics? *"Simulink uses an idealized timing model for block execution and communication. Both happen infinitely fast at exact points in simulated time. Thereafter, simulated time is advanced by exact time steps. All values on edges are constant in between time steps."* [Nicolae Marian, Yue Ma]

Threads are Not the Only Possibility: 6th example: Continuous-Time Languages



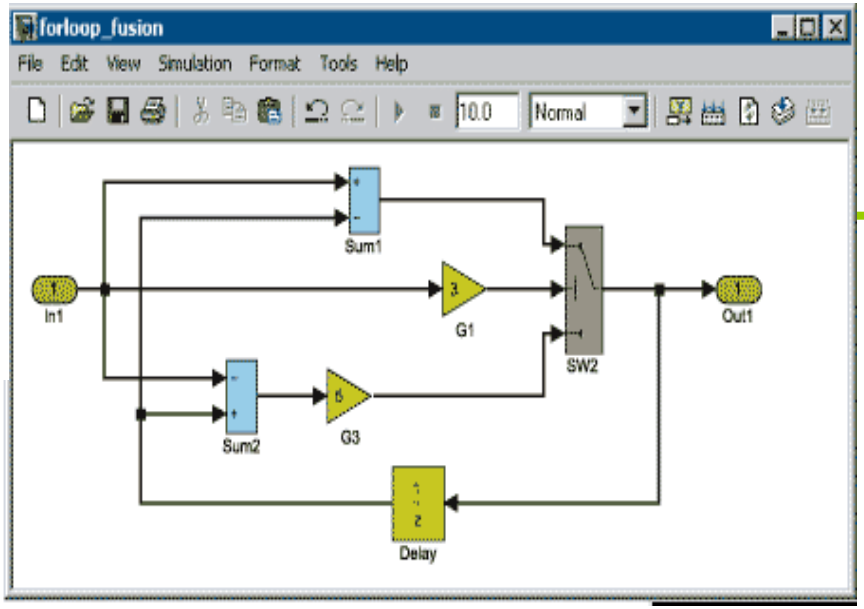
Simulink + Real-Time Workshop

continuous-time signal

Typical usage pattern:

- model the continuous dynamics of the physical plant
- model the discrete-time controller
- code generate the discrete-time controller

Starting point for “model-based design”



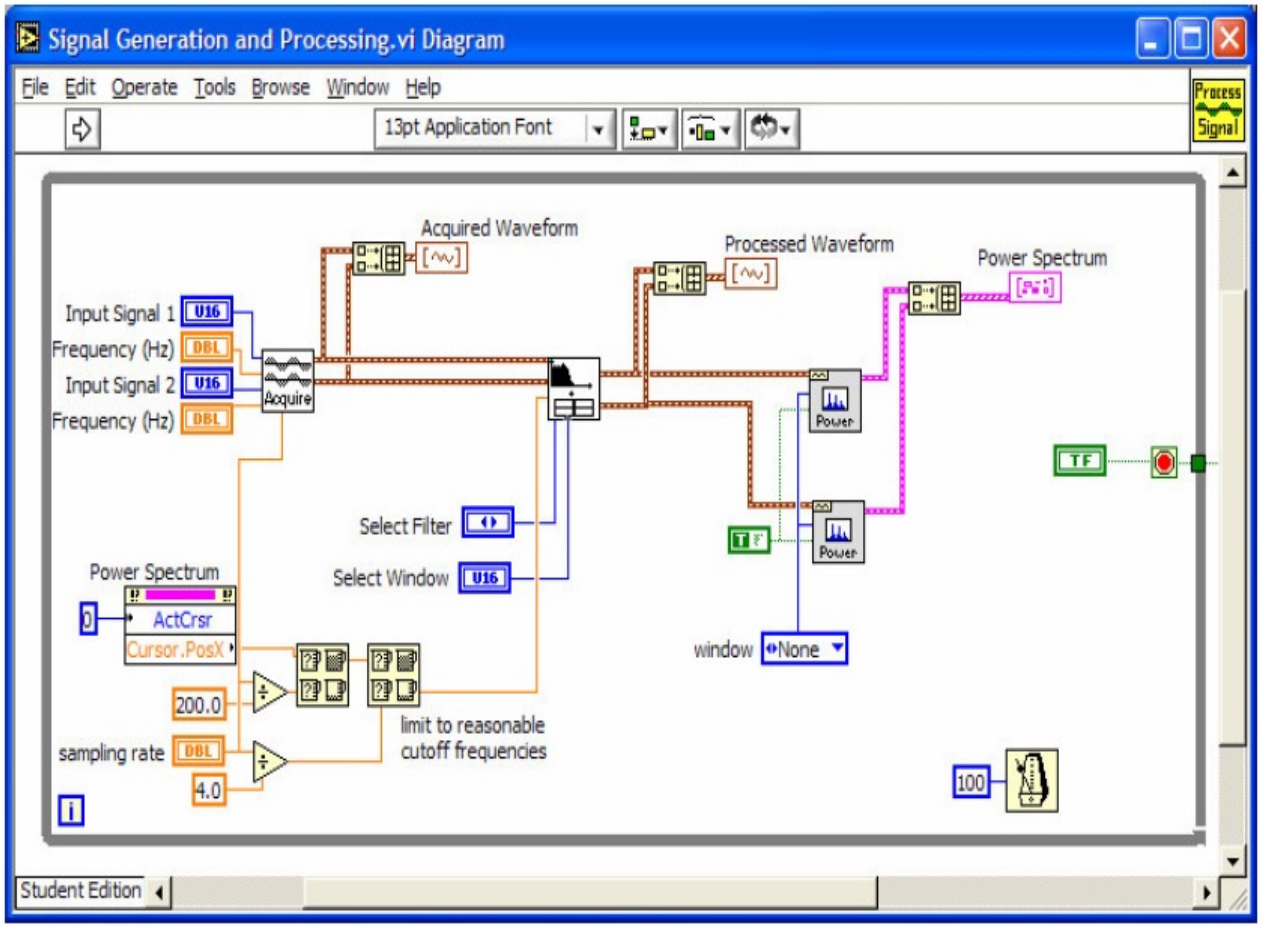

```
/* Switch: '<Root>/SW2' incorporates:
 *  Sum: '<Root>/Sum1'
 *  Gain: '<Root>/G1'
 *  Sum: '<Root>/Sum2'
 *  Gain: '<Root>/G3'
 */
for(i1=0; i1<10; i1++) {
  if(rtU.In1[i1] * 3.0 >= 0.0) {
    rtb_SW2_c[i1] = rtU.In1[i1] - rtDWork.Delay_DSTATE[i1];
  } else {
    rtb_SW2_c[i1] = (rtDWork.Delay_DSTATE[i1] - rtU.In1[i1]) * 5.0;
  }

  /* Outport: '<Root>/Out1' */
  rtY.Out1[i1] = rtb_SW2_c[i1];

  /* Update for UnitDelay: '<Root>/Delay' */
  rtDWork.Delay_DSTATE[i1] = rtb_SW2_c[i1];
}
```

Code automatically generated

Threads are Not the Only Possibility: 5th example: Instrumentation Languages

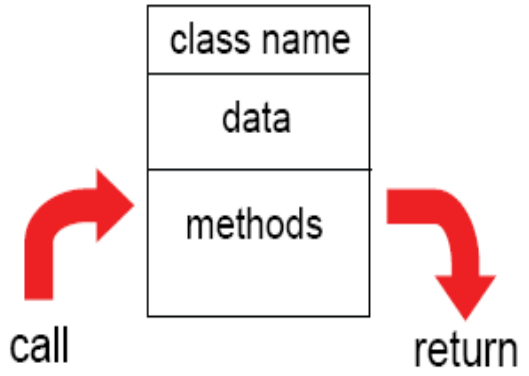


e.g. LabVIEW, Structured dataflow model of computation

Actor languages

-

The established: Object-oriented:

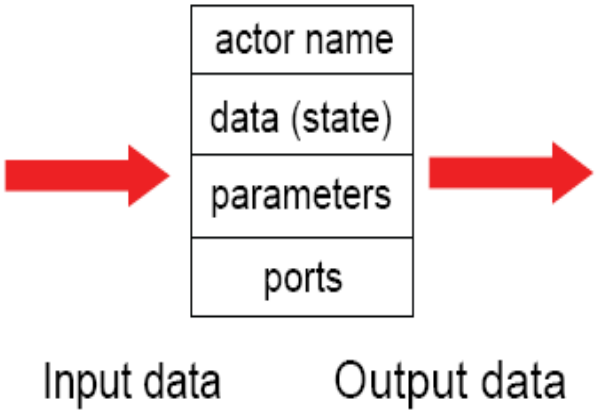


What flows through an object is sequential control

Things happen to objects

The alternative: Actor oriented:

Actors make things happen



What flows through an object is streams of data

© E. Lee, Berkeley

Summary

Data flow model of computation

- Motivation, definition
- Kahn process networks (KPNs)
- (H/C)SDF
- Visual programming languages
  - Simulink, Real Time Workshop, LabVIEW