Kahn Process Networks and a Reactive Extension

Twan Basten, Marc Geilen

Summer School on
Models for Embedded Signal Processing Systems

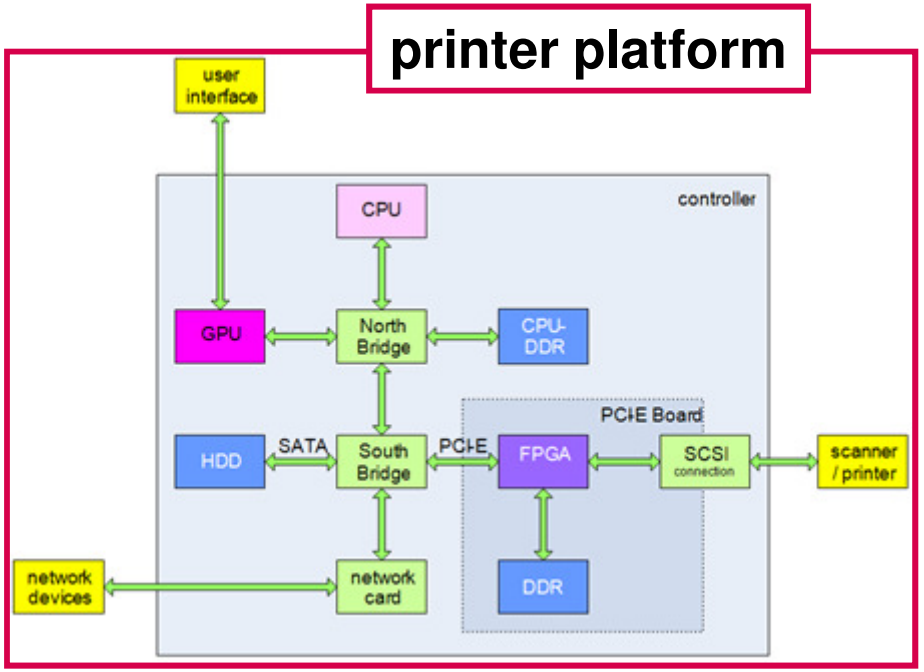
'Knowing is not understanding.'
Charles Kettering

The challenges of today

3 The problem: an example

- Use-cases to be mapped onto a professional printer
  - Copying (black&white, color, various paper sizes, zoom factors, …)
  - Printing
  - Scanning
  - Simultaneous printing and scanning

- How many CPUs, GPUs ?
- Processor speeds ?
- How to achieve X images/min ?
- Double resolution ?
- Run-time changes in speed, job type, power budget … ?



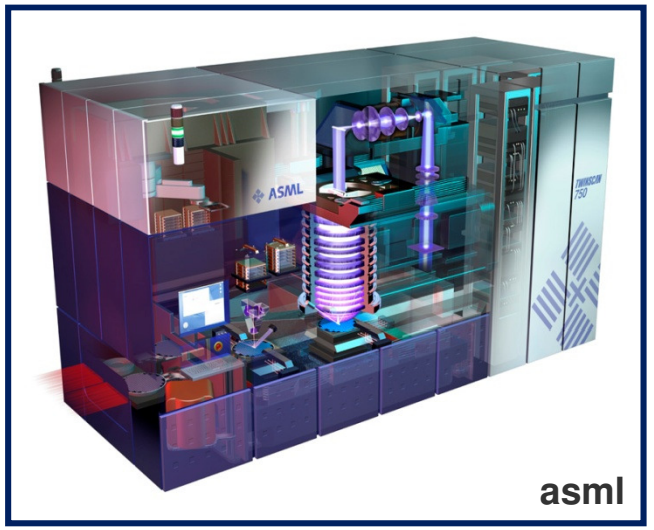
Objectives

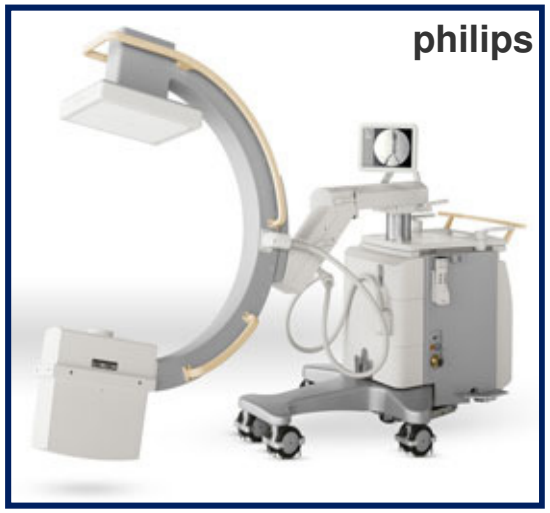
- Reliable, resource-efficient embedded systems
- Fast, predictable design trajectories

5 Objectives

- Reliable, resource-efficient embedded systems
- Fast, predictable design trajectories

High-tech professional systems

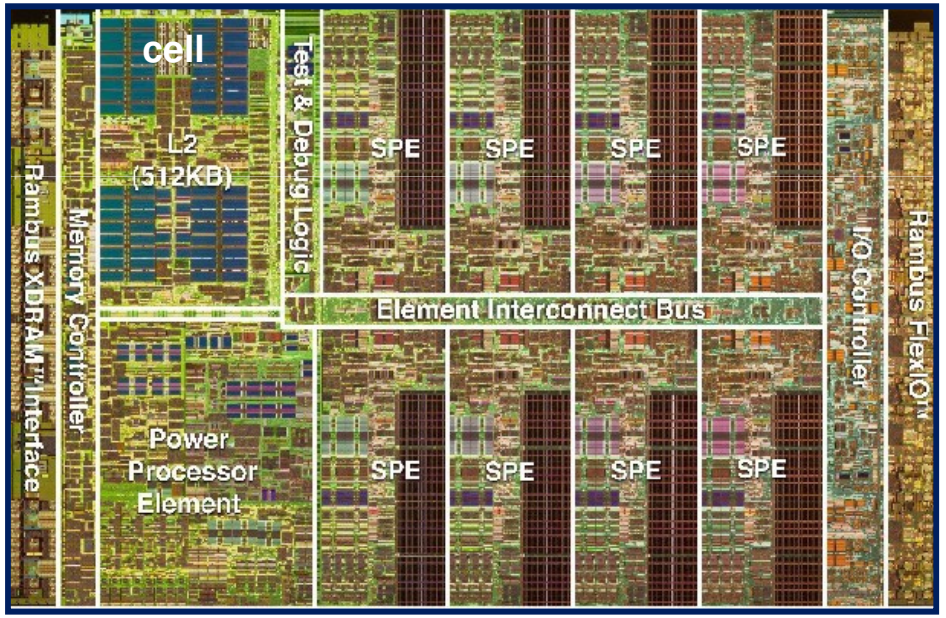


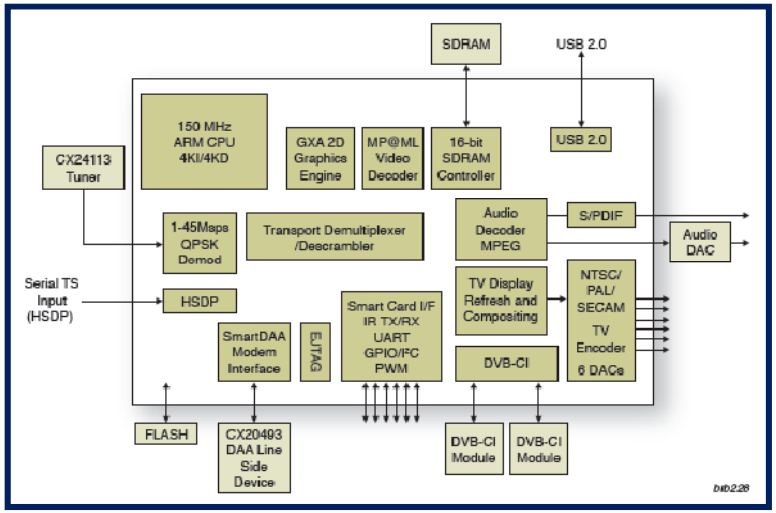


6 Objectives

- Reliable, resource-efficient embedded systems
- Fast, predictable design trajectories

Systems-on-chip

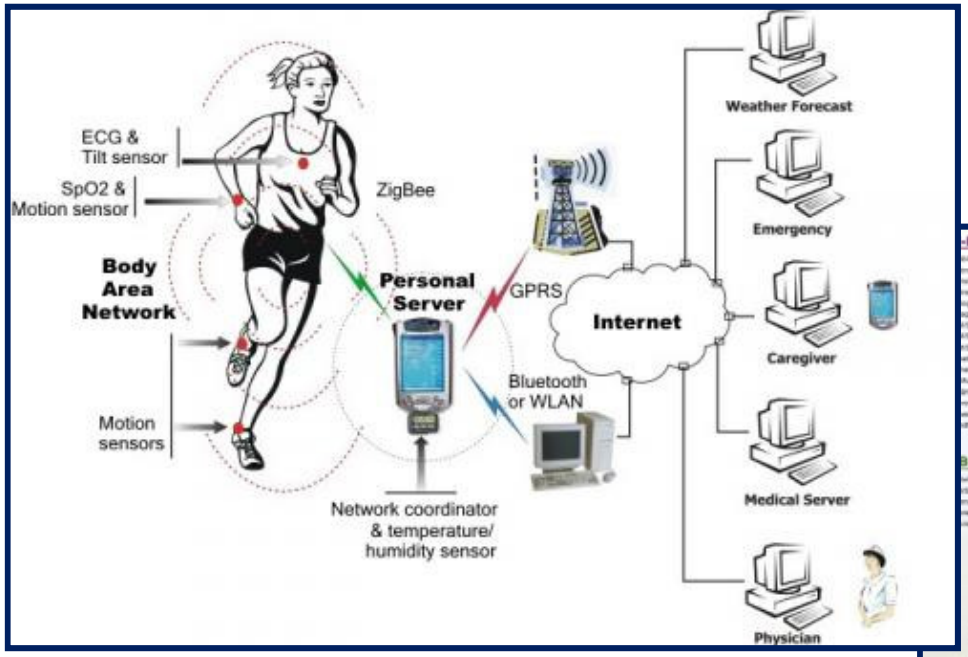




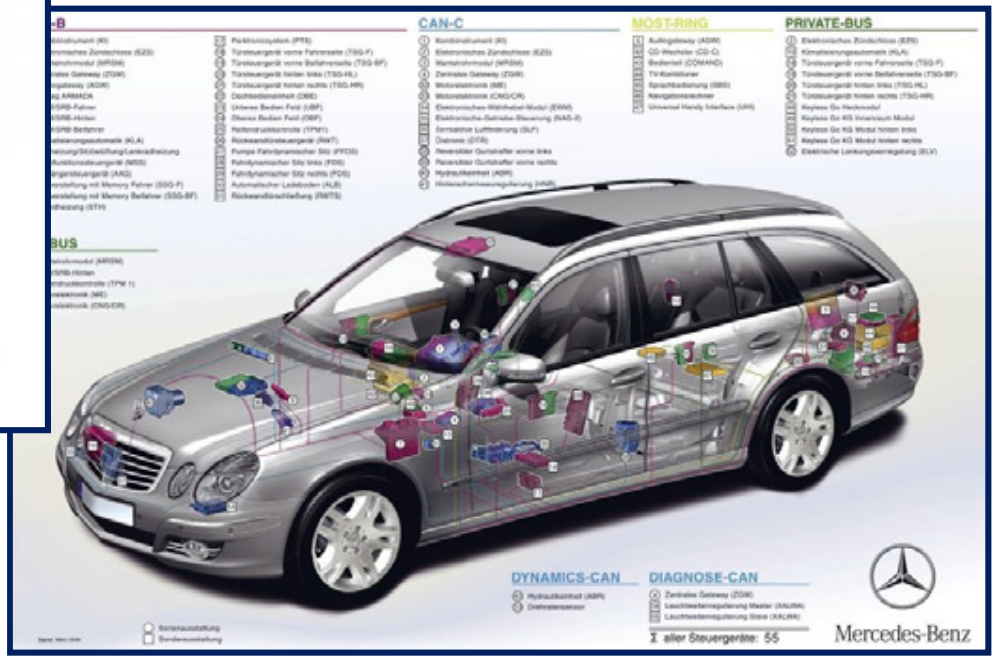
challenges → models → kpn → rpn → the hierarchy → the future

7 Objectives

- Reliable, resource-efficient embedded systems
- Fast, predictable design trajectories



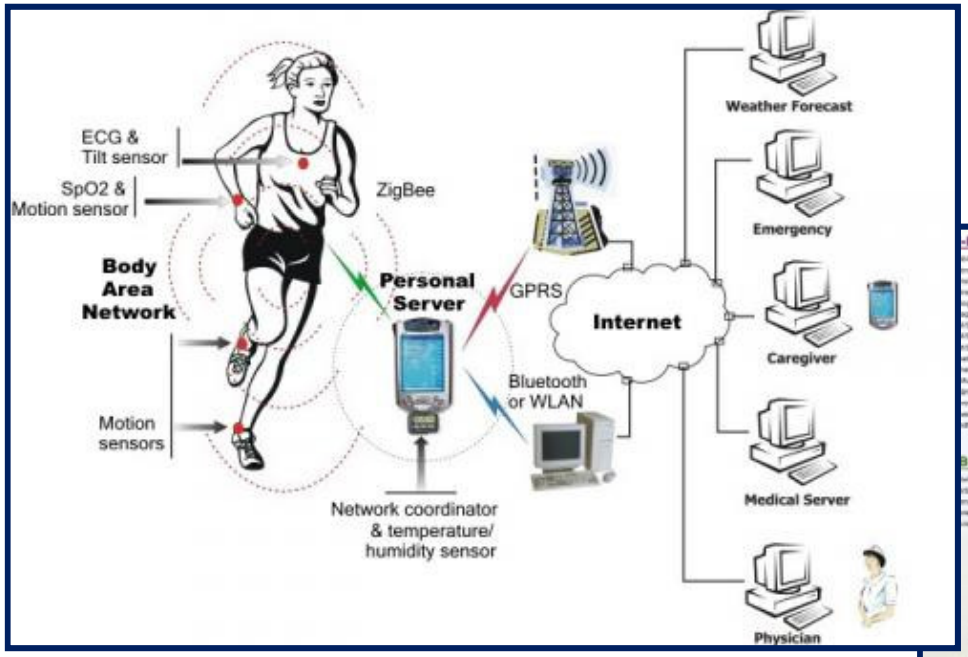
Networked embedded systems



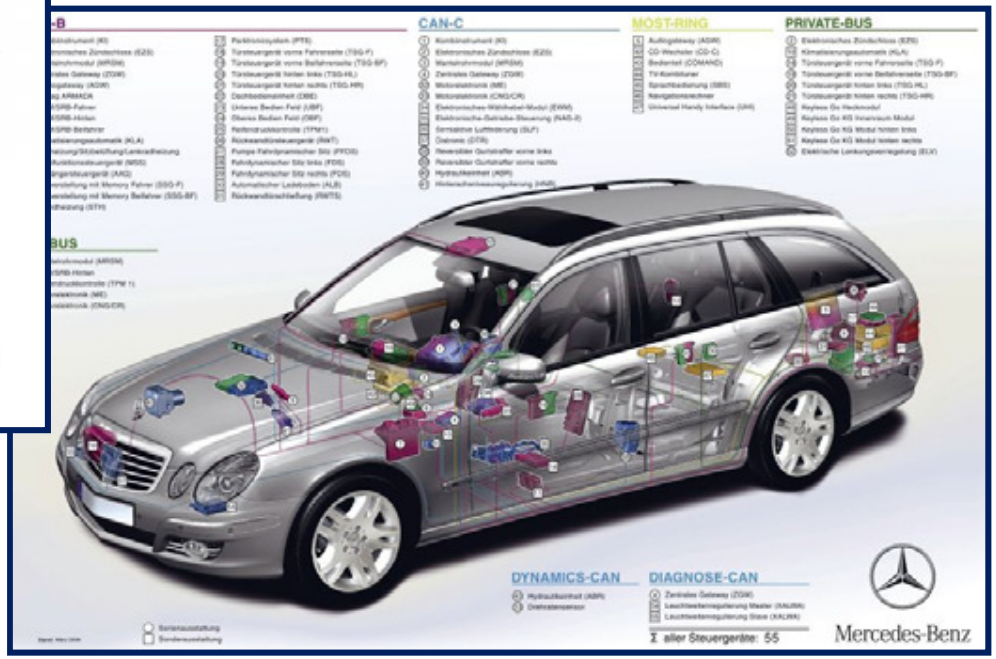
challenges → models → kpn → rpn → the hierarchy → the future

8 Trend: increasing complexity

- Multiprocessor systems / networking / concurrency
- Application convergence / application evolution
- Mixed data- and event processing
- Variability / dynamism



Networked embedded systems



9 Scientific challenges

- Reliable, resource-efficient embedded systems
  - Computational modeling and analysis
  - Model-driven synthesis and run-time management
- Fast, predictable design trajectories
  - Complexity reduction
  - (Semi-)automatic system synthesis

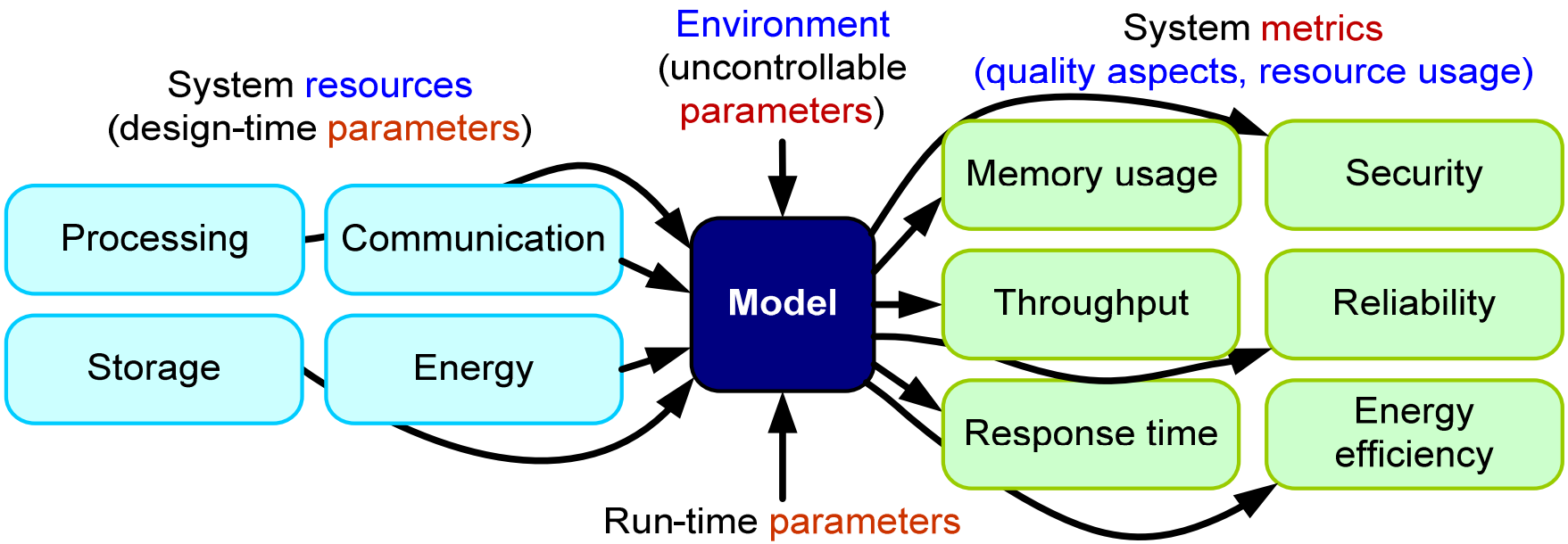
10 Outline

- The challenges of today
- Computational models
- Kahn Process Networks
- Reactive Process Networks
- The dataflow hierarchy
- Looking forward

Computational models

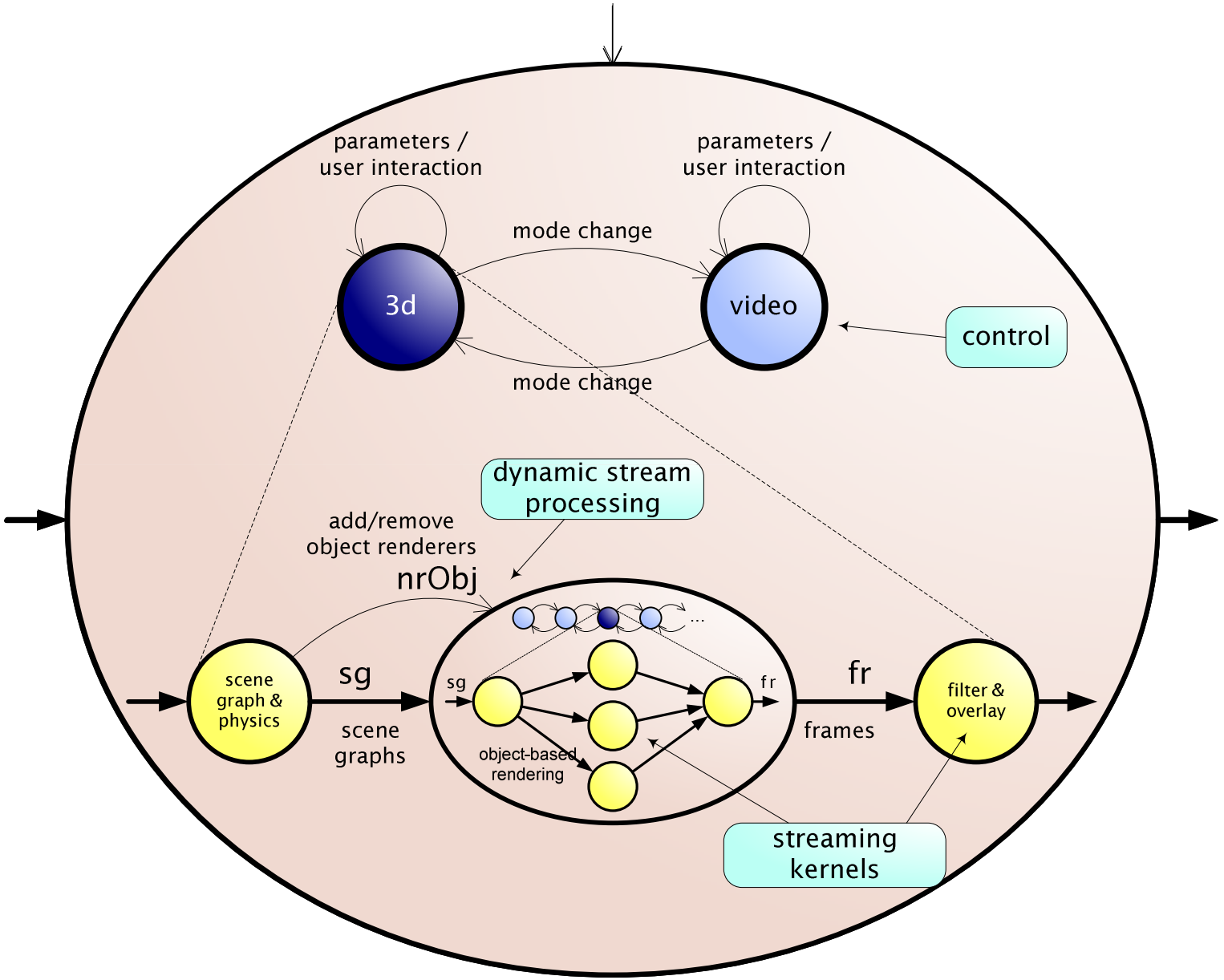
A model

… predicts system metrics given parameter values

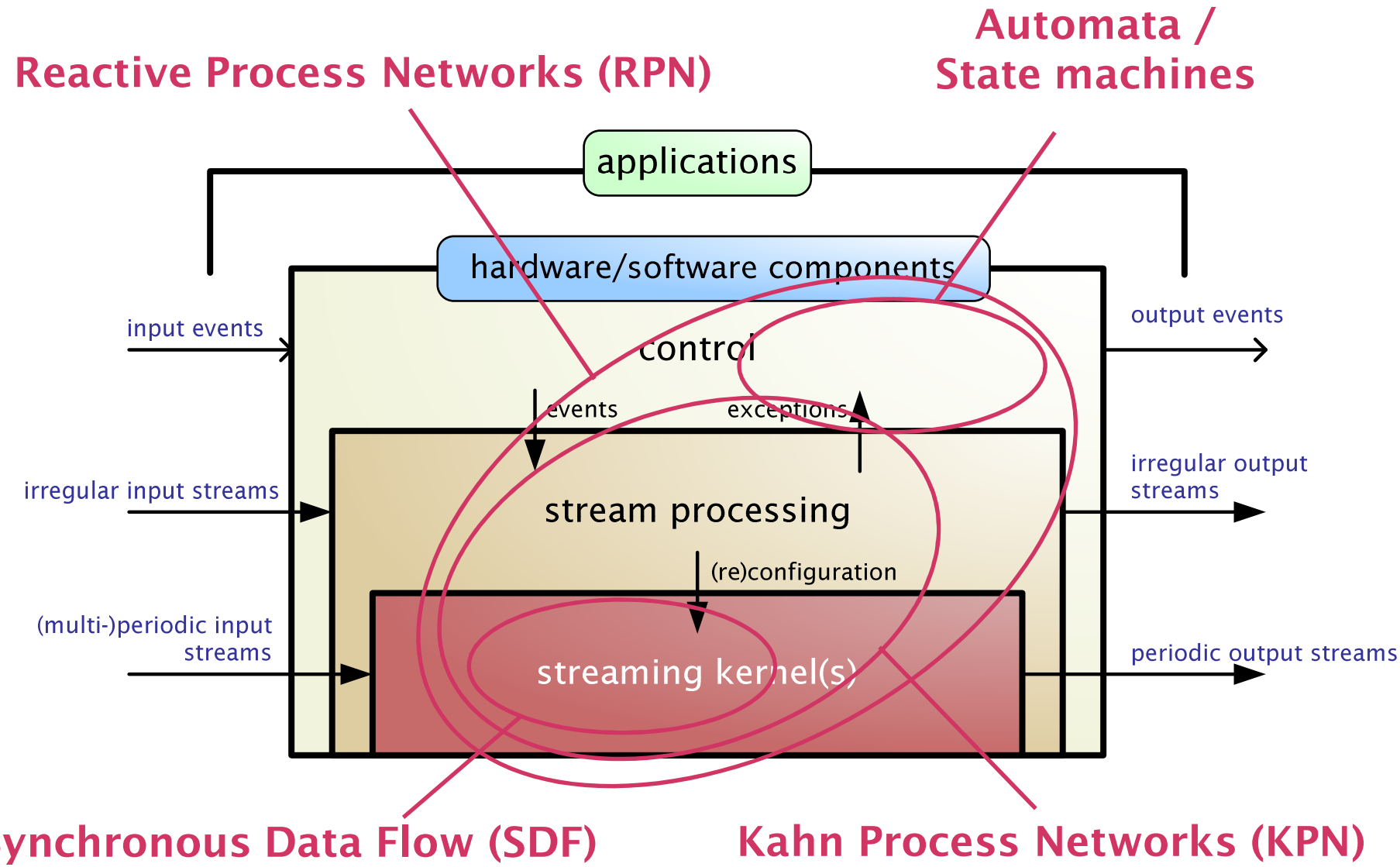


… should be targeted to the problem at hand

13 A gaming example



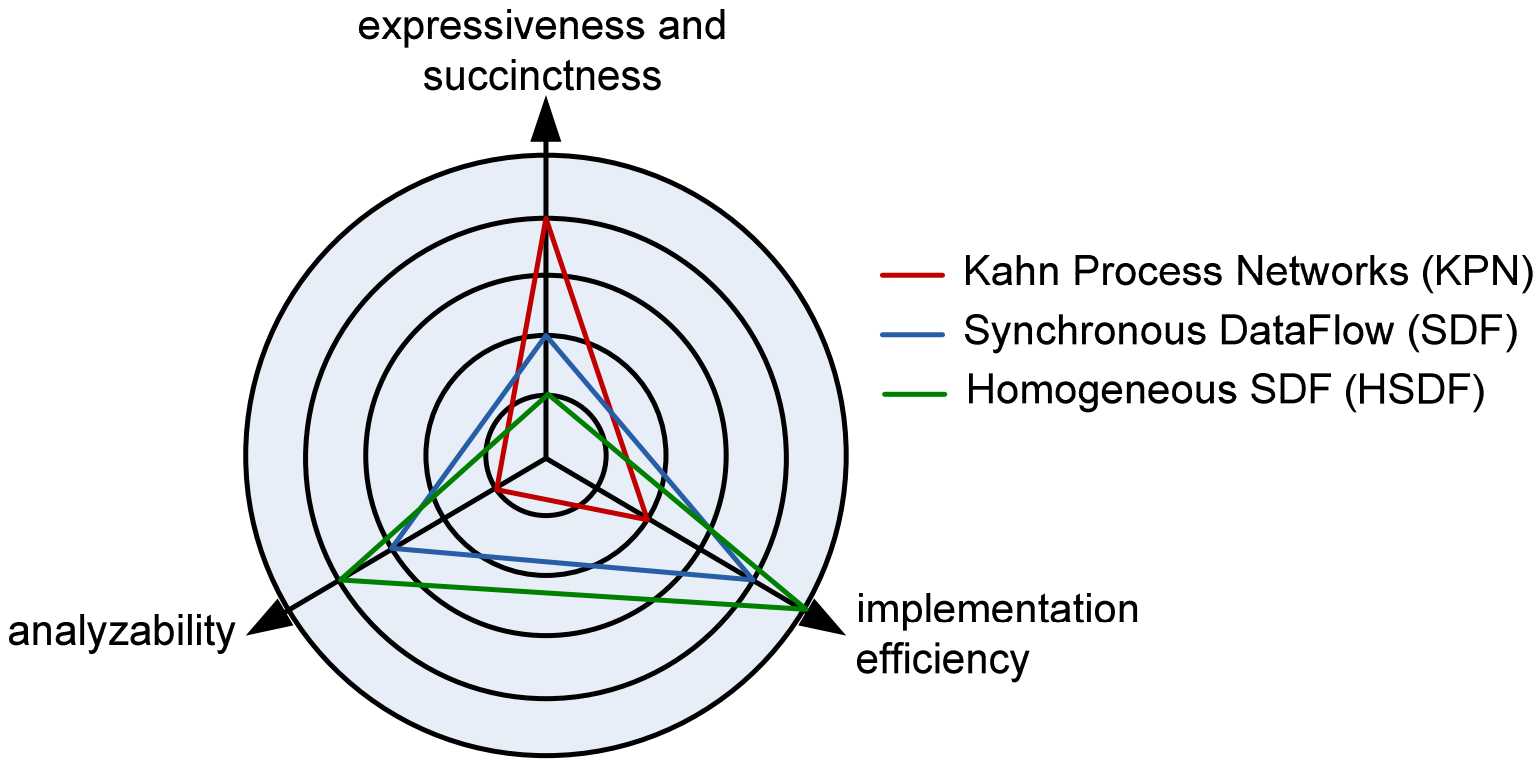
Application domain



Essential challenges

predictability and efficiency

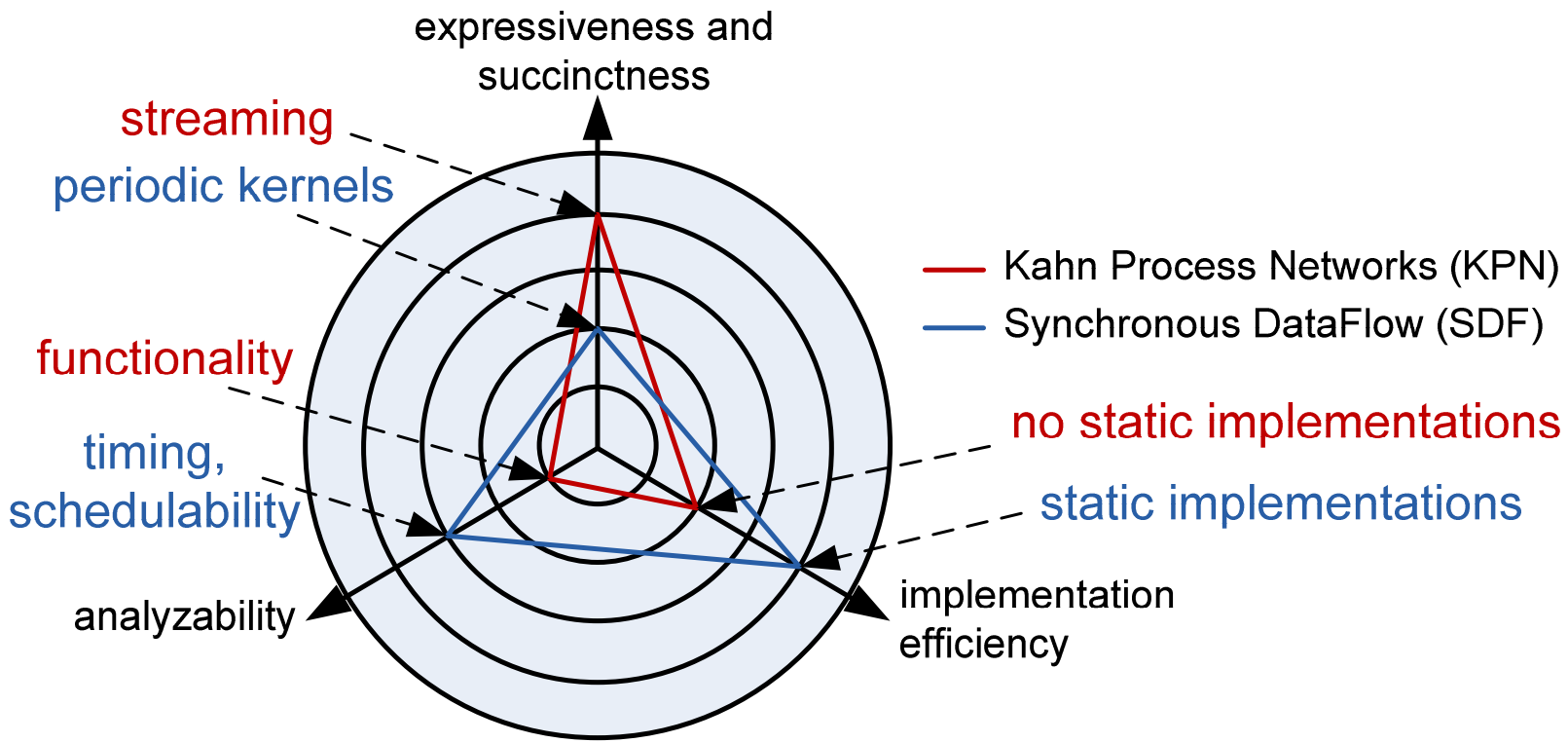
… are contradictory



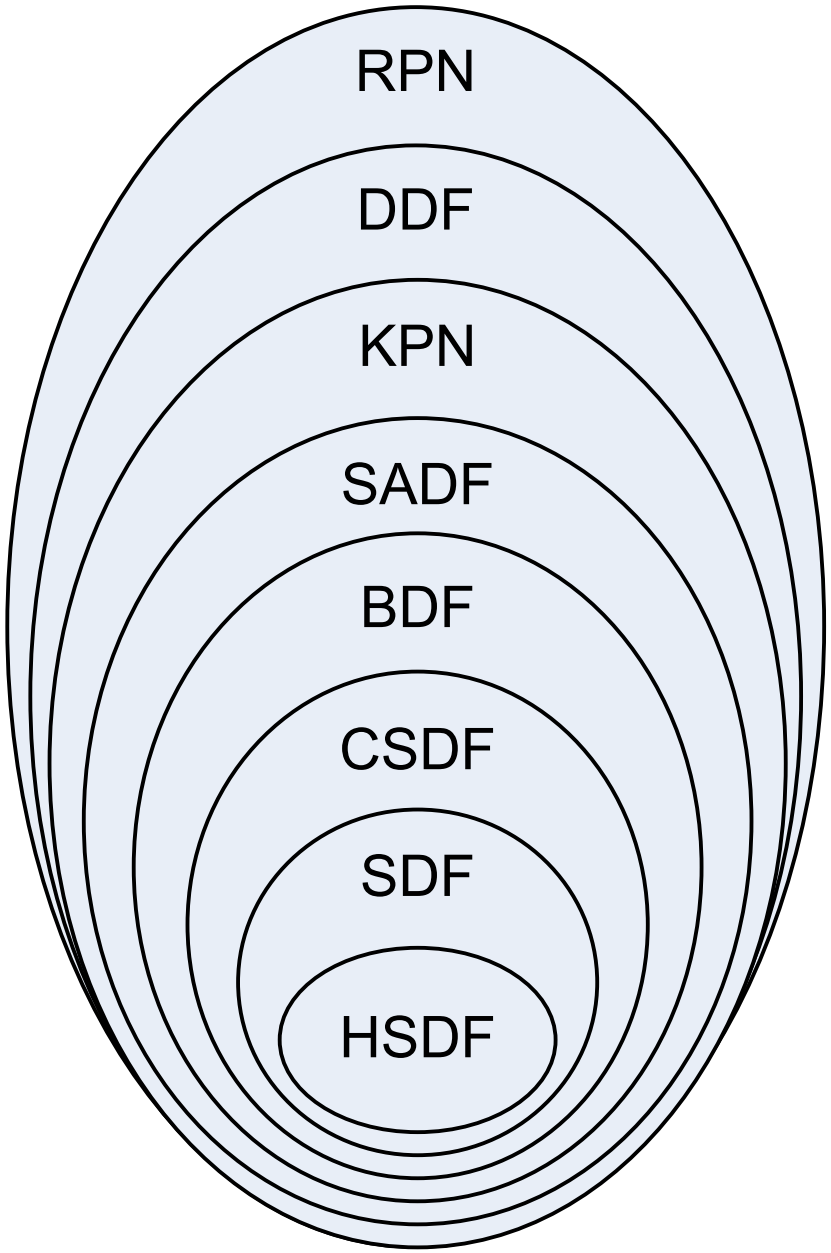
Essential challenges

predictability and efficiency

… are contradictory



17 Dataflow expressiveness hierarchy



RPN: Reactive Process Networks

KPN: Kahn Process Networks

DF: DataFlow

DDF: Dynamic DF

SADF: Scenario-Aware DF

BDF: Boolean DF

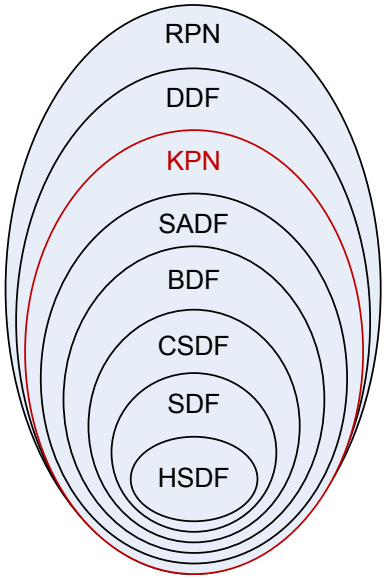
CSDF: Cyclo-Static DF

SDF: Synchronous DF

HSDF: Homogeneous SDF

BDF and larger: Turing complete

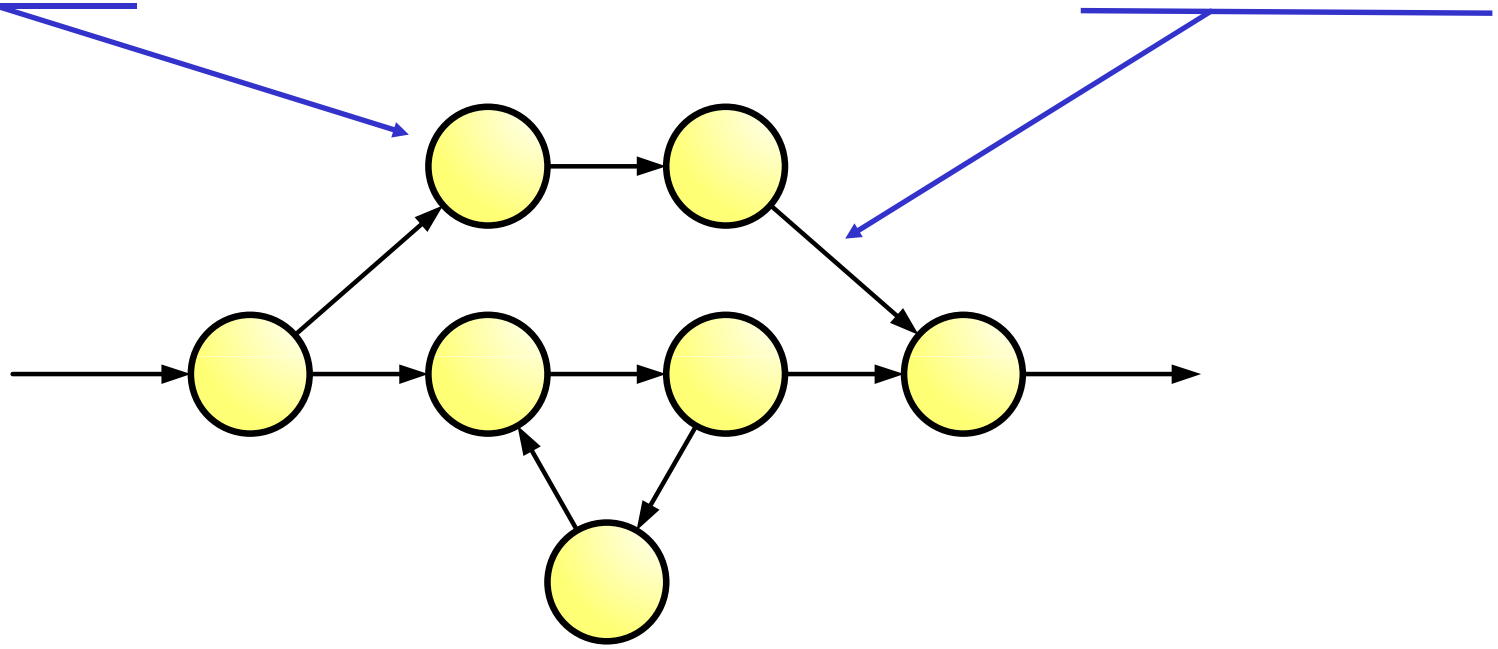
Kahn Process Networks



[G. Kahn. The semantics of a simple language for parallel programming. Proc. Information Processing 1974.]

19 Kahn Process Networks

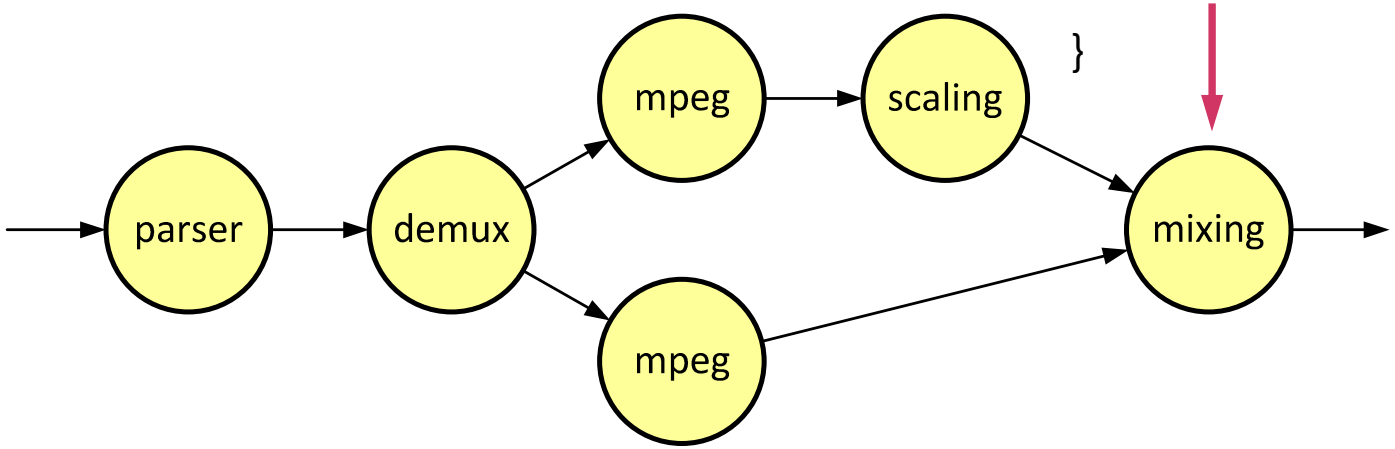
processes communicating via unbounded fifo queues



abstraction: only functionality, buffering
processes: need to be functional

Example KPN

Picture in Picture



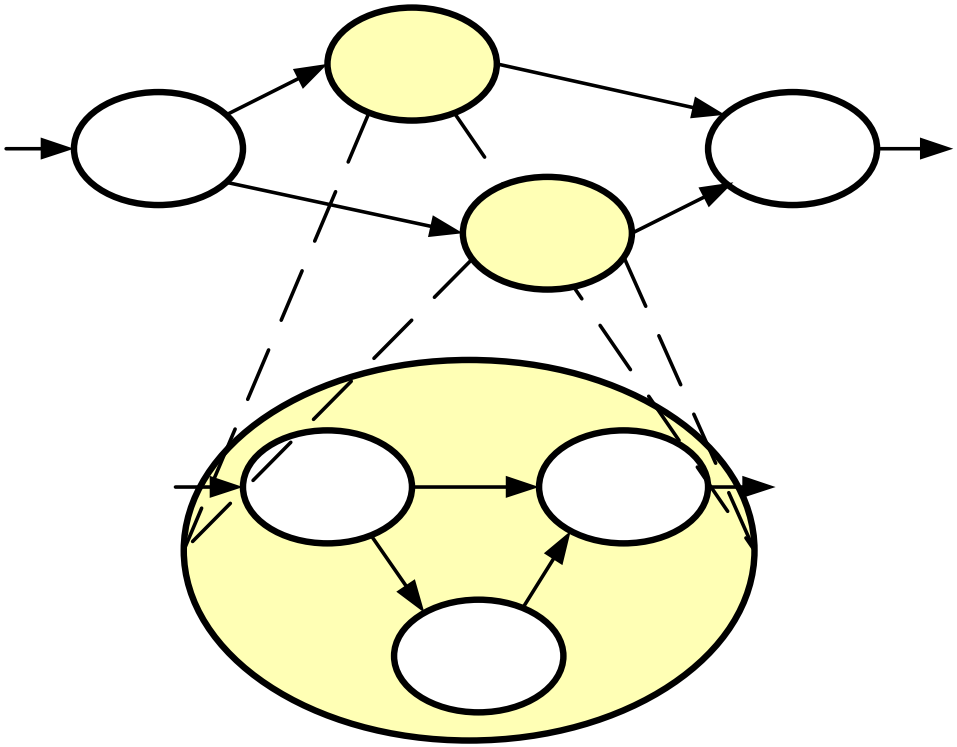

```
mixing()
while(true){
    read(in1, frame1);
    preprocess(frame1);
    read(in2, frame2);
    frame3 = combine(frame1, frame2);
    write(out, frame3);
}
```

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

21 Denotational semantics

processes = (mathematical) functions
fifo queues = strings, sequences of data tokens

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



compositional semantics

Continuity

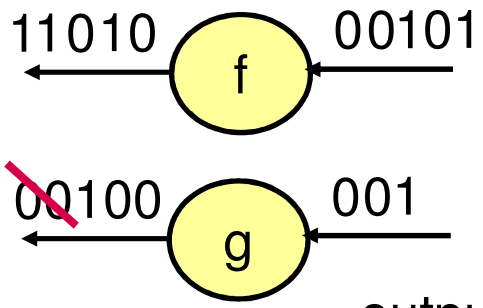
monotonicity

if input τ extends input σ
then output $f(\tau)$ extends output $f(\sigma)$

f(001)=110

f(00101)=11010

g(001)=100



bit-wise invert

reverse

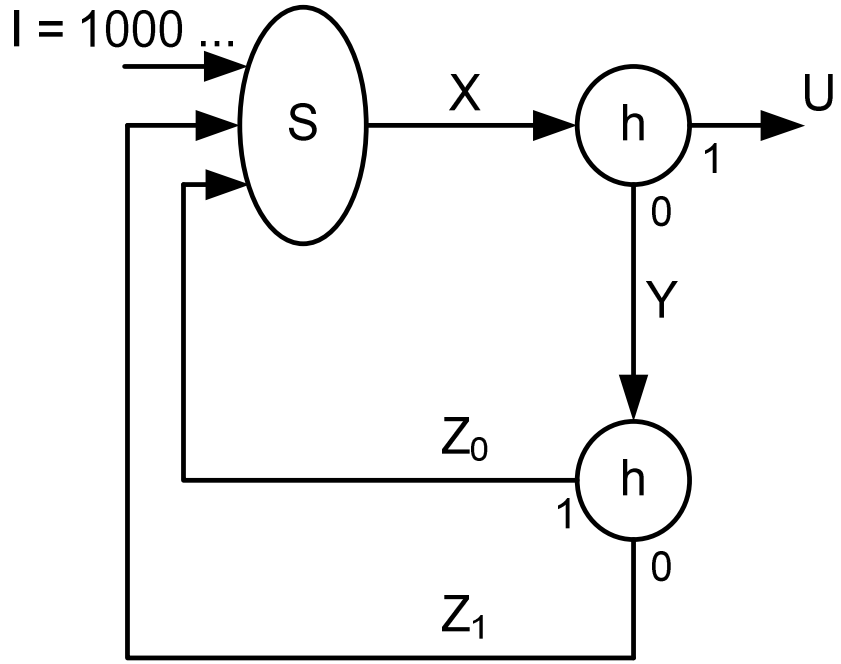
output must be taken back !! ☹

continuity

generalization of monotonicity to inputs of arbitrary, particularly infinite, length

“ continuity = streaming ”

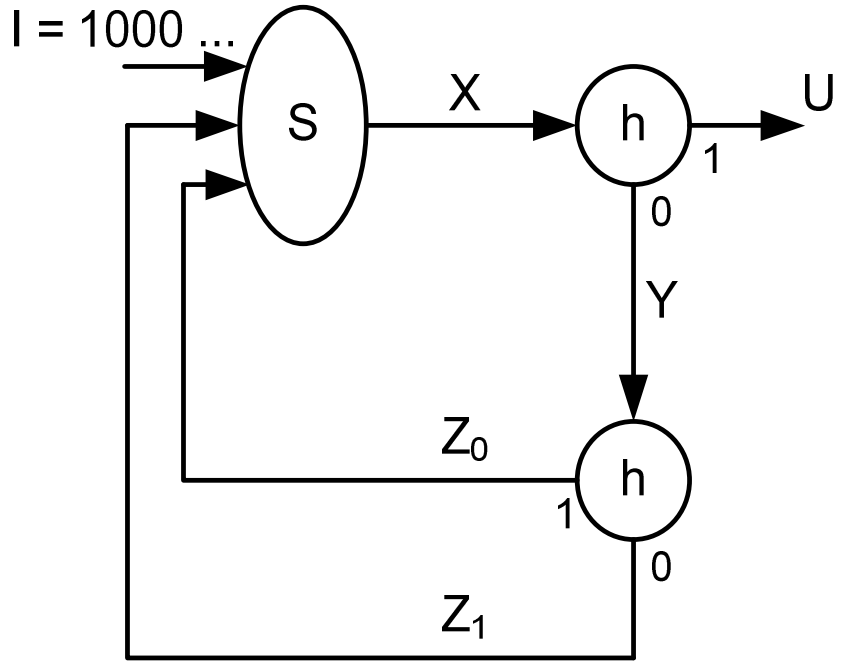
23 Computing the function of a network



S: elementwise sum
h_0: prefix a 0
h_1: copy

the continuous function of a network
is the least fixpoint
of a set of fixpoint equations
and can be computed via a
Kleene iteration

24 Fixpoint equations

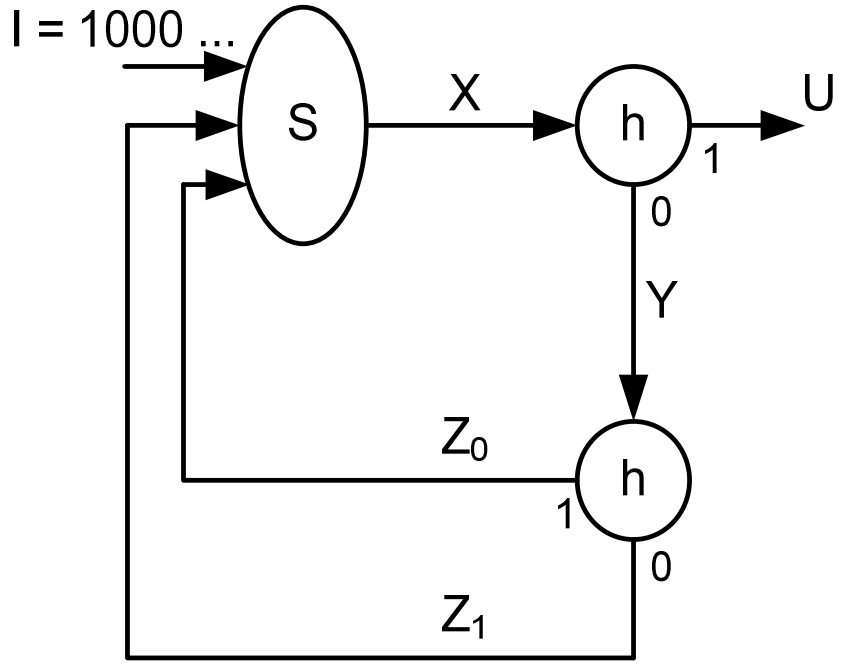


S: elementwise sum
h_0: prefix a 0
h_1: copy

$U = X$
$X = S(I,Z_0,Z_1)$
$Y = h_0(X)$
$Z_0 = Y$
$Z_1 = h_0(Y)$

$U = S(I,0U,00U)$

25 Kleene iteration



S: elementwise sum
h_0: prefix a 0
h_1: copy

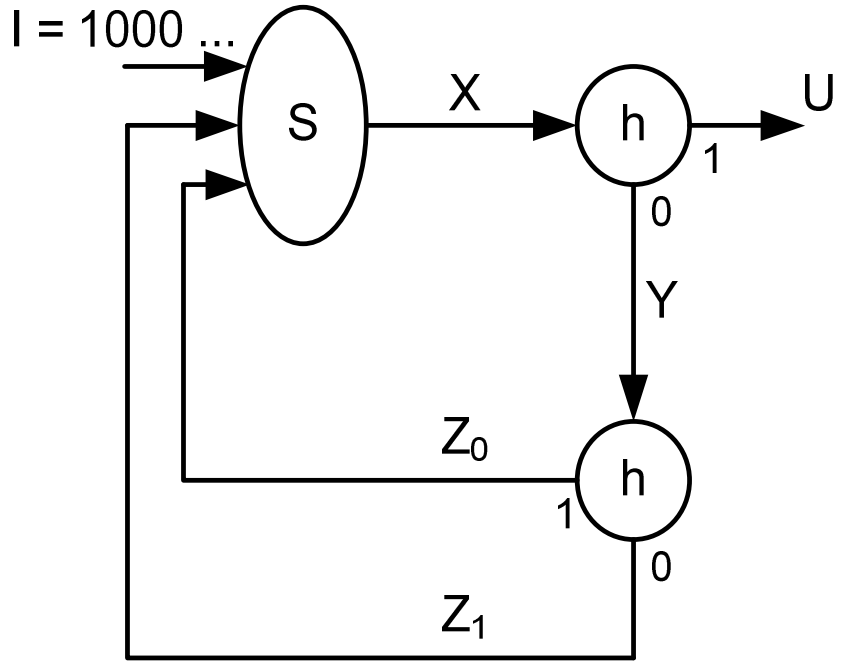
the continuous function of a network
is the least fixpoint
of a set of fixpoint equations
and can be computed via a
Kleene iteration

$U_0 = \varepsilon$
$U_{i+1} = S(I,0U_i,00U_i)$

$U = S(I,0U,00U)$

i =	0	1	2	3	4	5	6	7	8	9	10
U_i	ε										

Kleene iteration



S: elementwise sum
h_0: prefix a 0
h_1: copy

Scrap paper

U_1
$= S(I,0U_0,00U_0)$
$= S(1000\ldots,0\varepsilon,00\varepsilon)$
$= S(1000\ldots,0,00)$
$= 1+0+0$
$= 1$

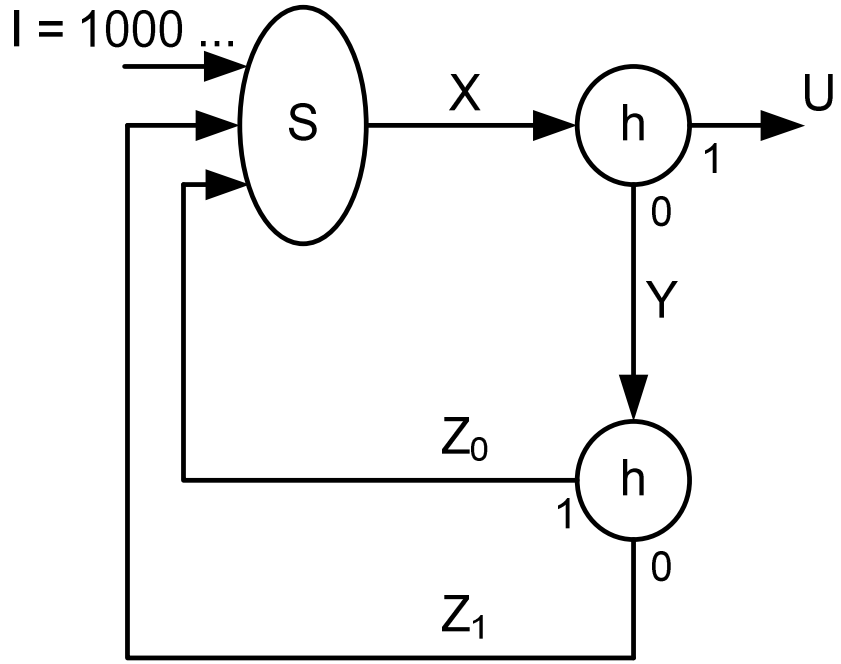
n of a network
point
equations
uted via a
tion

$U_0 = \varepsilon$
$U_{i+1} = S(I,0U_i,00U_i)$

i =	0	1	2	3	4	5	6	7	8	9	10
U_i	ε	1									

U(0)

27 Kleene iteration



S: elementwise sum
h_0: prefix a 0
h_1: copy

Scrap paper

U_2
$= S(I,0U_1,00U_1)$
$= S(1000\ldots,01,001)$
$= (1+0+0)(0+1+0)$
$= 11$

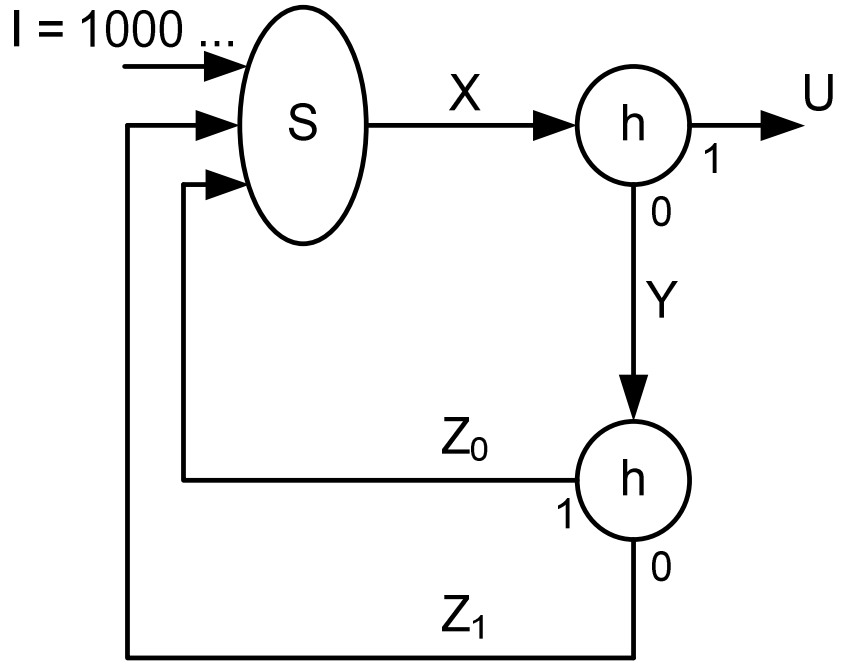
of a network
point
equations
ted via a
ion

$U_0 = \varepsilon$
$U_{i+1} = S(I,0U_i,00U_i)$

i =	0	1	2	3	4	5	6	7	8	9	10
U_i	ε	1	11								

U(0) U(1)

28 Kleene iteration



S: elementwise sum
h_0: prefix a 0
h_1: copy

the continuous function of a network
is the least fixpoint
of a set of fixpoint equations
and can be computed via a
Kleene iteration

$U_0 = \varepsilon$
$U_{i+1} = S(I,0U_i,00U_i)$

$U = S(I,0U,00U)$

i =	0	1	2	3	4	5	6	7	8	9	10
U_i	ε	1	11	112	1123	… 5	… 8	… 13	… 21	… 34	… 55
		↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
		U(0)	U(1)	U(2)	U(3)	U(4)	U(5)	U(6)	U(7)	U(8)	U(9)

29 Strengths & weaknesses

- compositionality
- determinacy (execution order and timing)
- explicit concurrency
- explicit communication (via fifos; no shared variables)
- captures data-dependent streaming behavior
- high abstraction level
- undecidable, e.g., minimal buffer sizes
- needs run-time resource management when directly implemented
- cannot capture asynchronous reactive behavior

An operational view

denotational semantics: what output is computed

operational semantics: how the output is computed

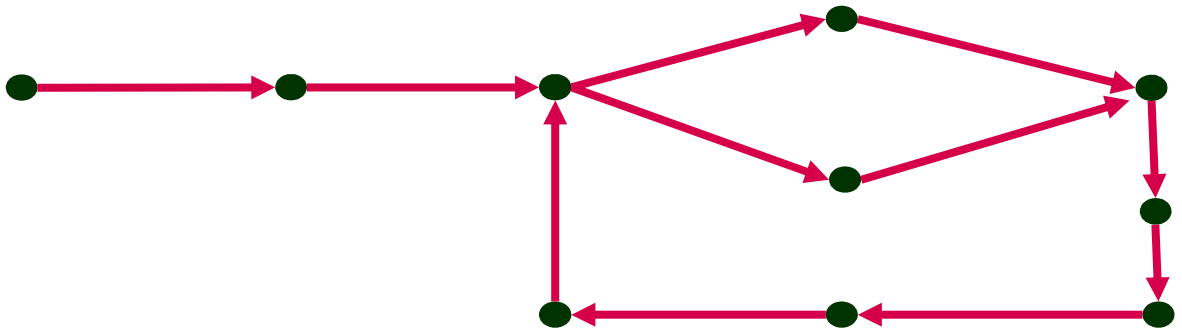
- strings are streams of tokens and fifo buffers of infinite capacity function as windows on those streams.
- an operational execution of functions conforms to (destructively) reading and writing tokens with internal computations in between.

31 Operational semantics

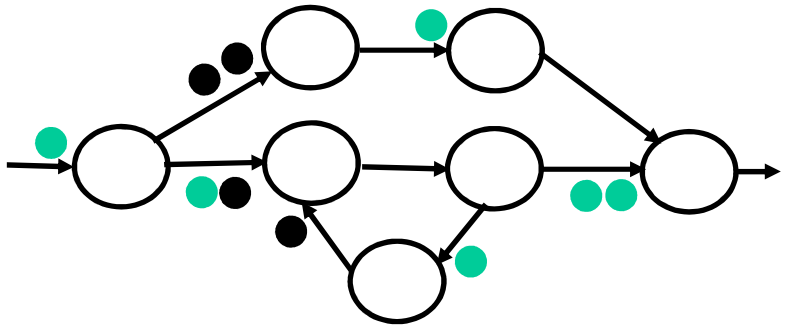
operational semantics captures all allowed ways to make behavior operational

based on
configurations
i.e., KPN + fifo fillings

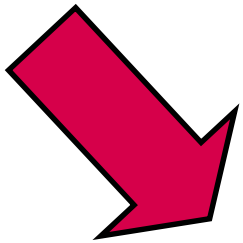
and
transitions
from one configuration to another one



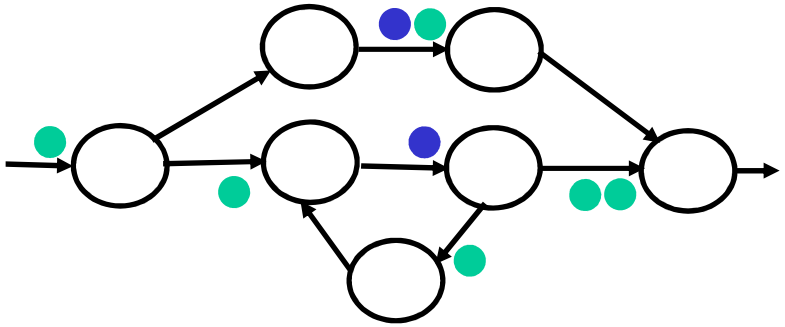
Operational semantics



Configuration A



transition



Configuration B

The Kahn Principle

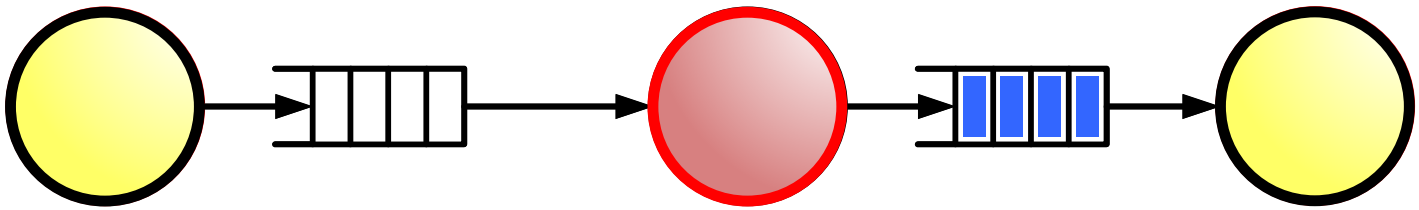
“ But hey, now we have two semantics of the model!
Are they the same? ”

The Kahn Principle:
Any fair execution in the operational model realizes the denotational semantics, the mathematical function

This provides the boundaries for realizations of KPNs

34 Implementations of KPNs

- **bounded FIFOs** combine aspects of data- and demand driven execution
- but **how large** should the FIFO buffers be?
- **undecidable**, FIFO sizes need to be **managed at run-time**



35 A run-time environment (RTE)

requirements

boundedness: fifo bounds may not grow indefinitely

completeness: progress must be made on all outputs towards the output prescribed by the semantics

there exists an RTE satisfying these requirements

prerequisites (undecidable in general!)

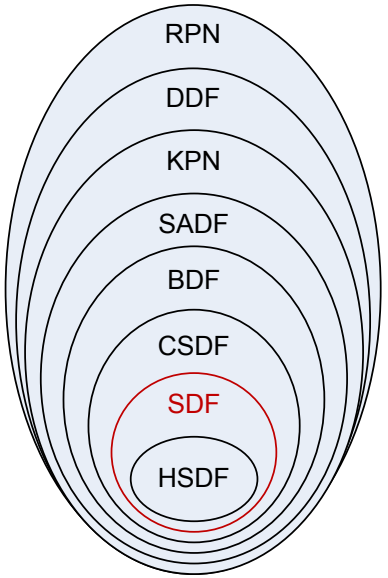
boundedness: there exists a fair execution within finite fifo bounds

effectiveness: every token that is produced is eventually also consumed

[M.C.W. Geilen and T. Basten. Requirements on the Execution of Kahn Process Networks. Proc. ESOP 2003.]

36 An analyzable subclass of KPN

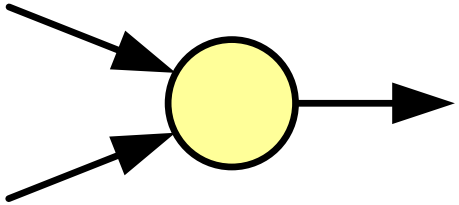
Synchronous DataFlow



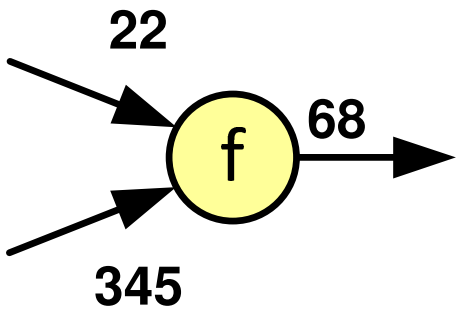
[E.A. Lee and D. Messerschmitt, Synchronous data flow, Proceedings of the IEEE, vol. 75, no. 9, pp. 1235–1245, 1987.]

Dataflow

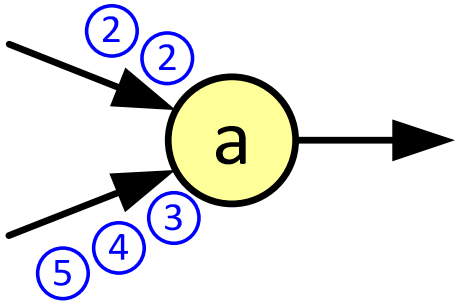
elementwise multiplication ?



function in a KPN



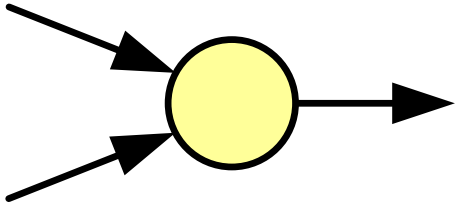
actor in dataflow



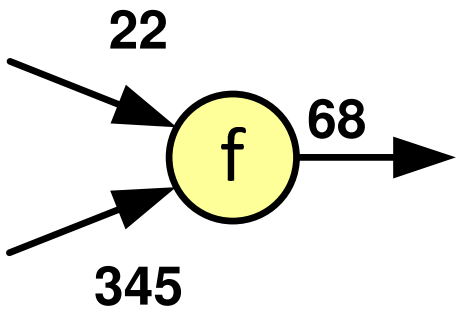
actor firing
(consumption and production of tokens)

Dataflow

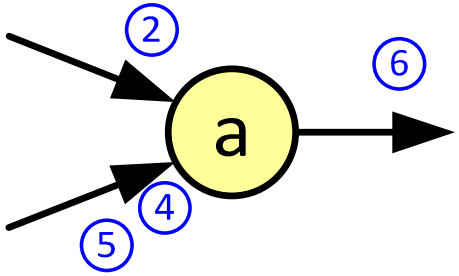
elementwise multiplication ?



function in a KPN



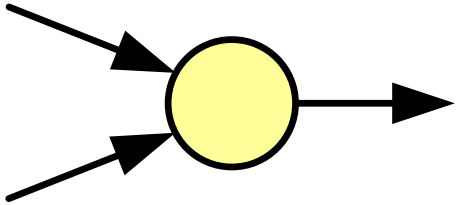
actor in dataflow



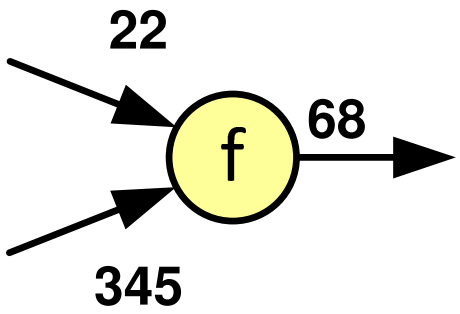
actor firing
(consumption and production of tokens)

Dataflow

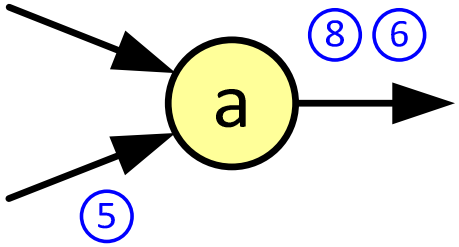
elementwise multiplication ?



function in a KPN

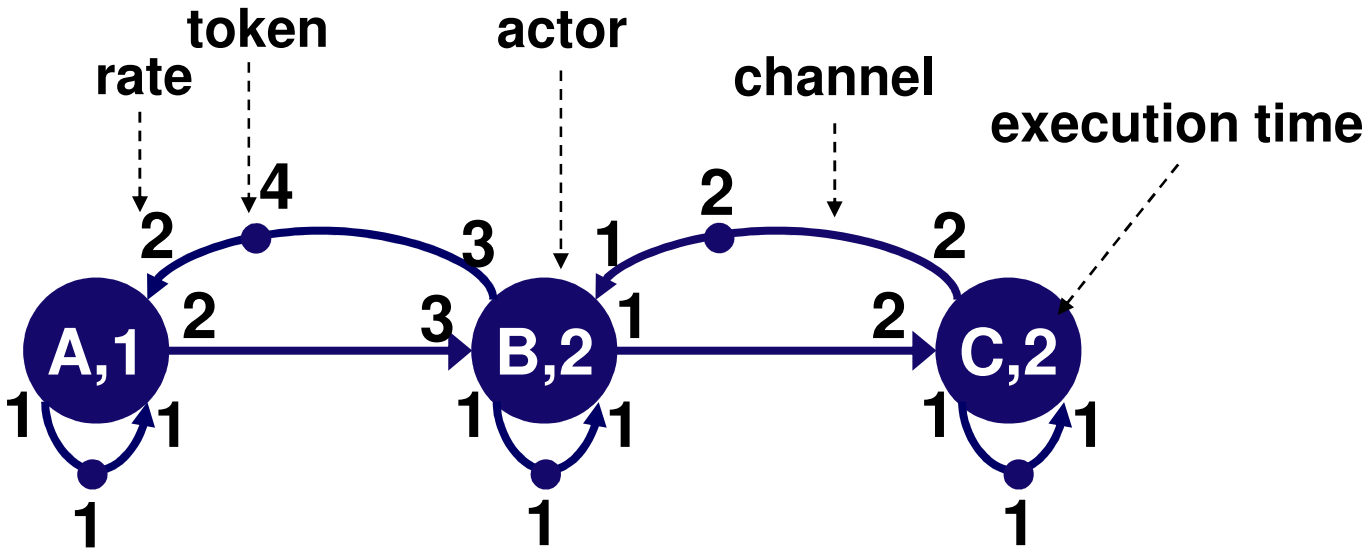


actor in dataflow



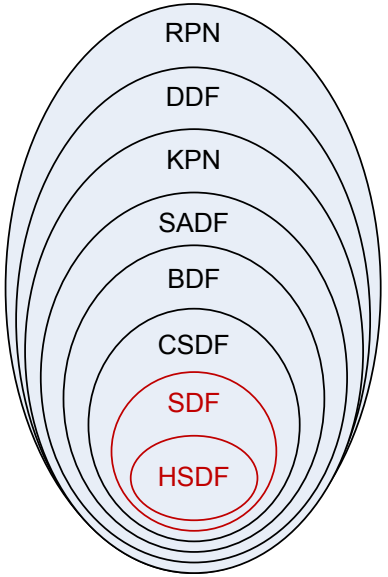
actor firing
(consumption and production of tokens)

Synchronous DataFlow

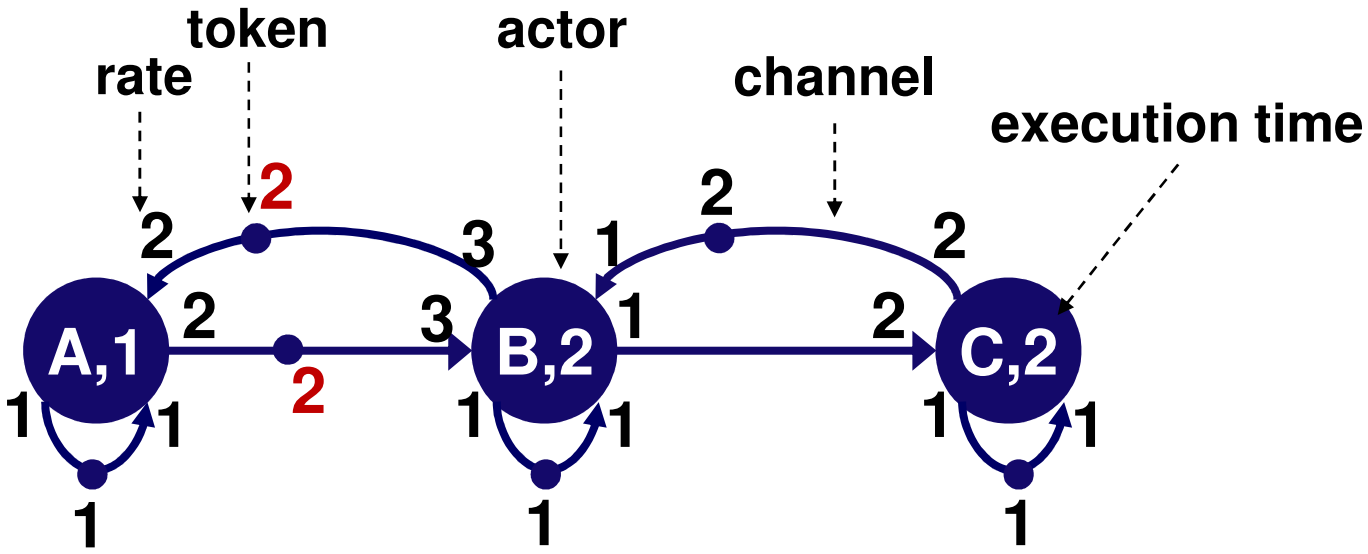


SDF: rates are fixed
(weighted marked graphs from Petri-net theory)

Homogeneous SDF: all rates 1
(marked graphs from Petri-net theory)

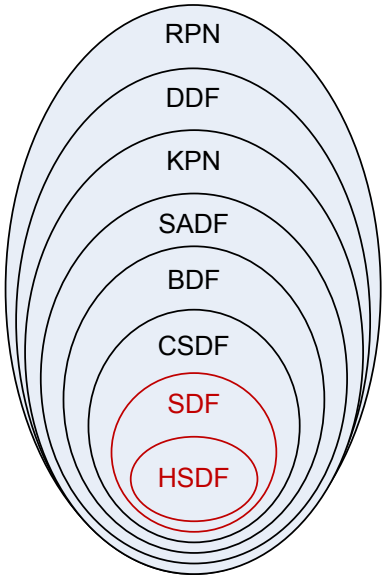


41 Synchronous DataFlow: actor firing



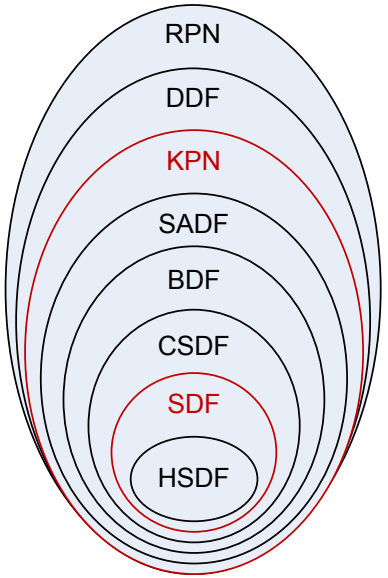
SDF: rates are fixed
(weighted marked graphs from Petri-net theory)

Homogeneous SDF: all rates 1
(marked graphs from Petri-net theory)



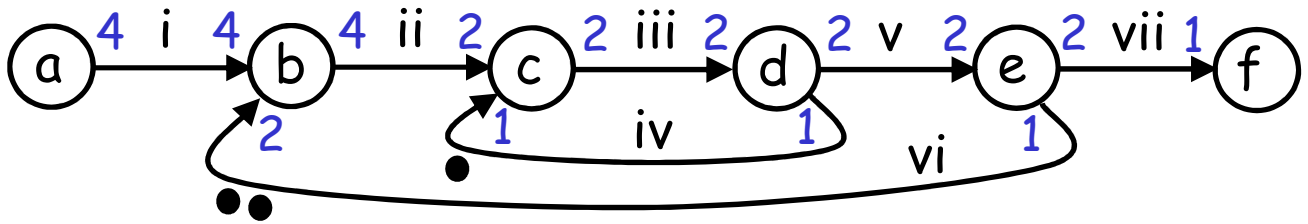
42 Every SDF is a KPN

The repeated firing of an actor can be captured as a recursive function that is continuous



Simple scheduling analysis

Example: pipeline with feedbacks



topology matrix
(consumption per actor per channel)

repetition vector

	a	b	c	d	e	f
i	-4	4	0	0	0	0
ii	0	-4	2	0	0	0
iii	0	0	-2	2	0	0
iv	0	0	0	-1	0	0
v	0	0	0	-2	2	0
vi	0	2	0	0	-1	0
vii	0	0	0	0	-2	1

$\bullet$

1	a
1	b
2	c
2	d
2	e
4	f

$=$

0	i
0	ii
0	iii
0	iv
0	v
0	vi
0	vii

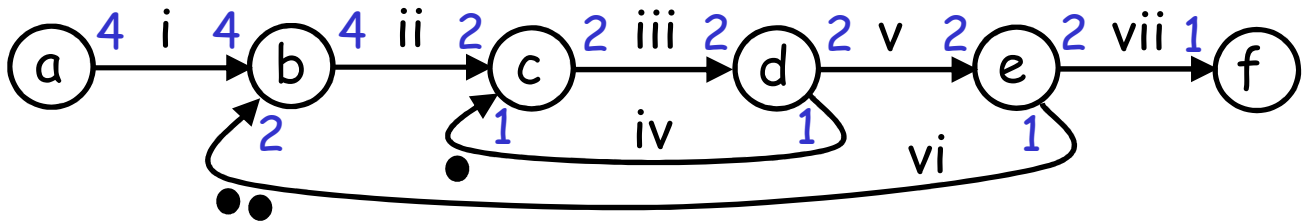
firing rates are consistent iff repetition vector exists

minimal (non-zero) number of actor firings that has no net effect on the initial token distribution

repetition vector can be computed by solving the so-called balance equations

Simple scheduling analysis

Example: pipeline with feedbacks



topology matrix
(consumption per actor per channel)

	a	b	c	d	e	f
i	-4	4	0	0	0	0
ii	0	-4	2	0	0	0
iii	0	0	-2	2	0	0
iv	0	0	0	-1	0	0
v	0	0	0	-2	2	0
vi	0	2	0	0	-1	0
vii	0	0	0	0	-2	1

• repetition vector

1	a
1	b
2	c
2	d
2	e
4	f

=

0	i
0	ii
0	iii
0	iv
0	v
0	vi
0	vii

iteration

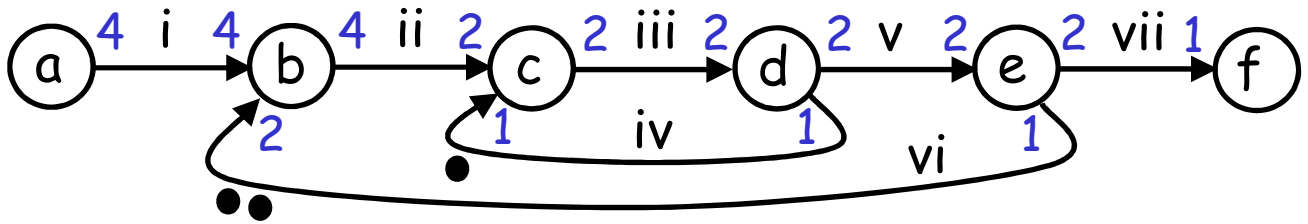
firing rates are consistent iff repetition vector exists

minimal (non-zero) number of actor firings that has no net effect on the initial token distribution

repetition vector can be computed by solving the so-called balance equations

Simple scheduling analysis

Example: pipeline with feedbacks



topology matrix
(consumption per actor per channel)

	a	b	c	d	e	f
i	-4	4	0	0	0	0
ii	0	-4	2	0	0	0
iii	0	0	-2	2	0	0
iv	0	0	0	-1	0	0
v	0	0	0	-2	2	0
vi	0	2	0	0	-1	0
vii	0	0	0	0	-2	1

• repetition vector

1	a
1	b
2	c
2	d
2	e
4	f

basis for static schedule

eg

$s = (a\ b\ (c\ d)^2\ (e\ f^2)^2)^\omega$

is one possible schedule

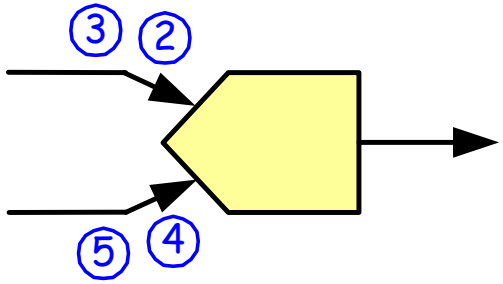
note

channel v requires a size of 4 tokens in s whereas a schedule exists in which v needs only a size of 2 tokens

determining minimal buffer sizes is NP-hard

46 KPN: Fundamental limitation

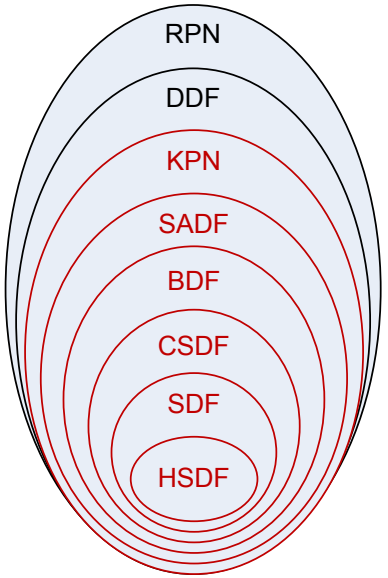
non-deterministic merge:



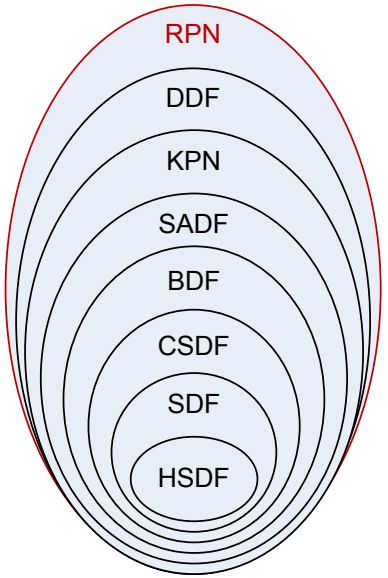
forwards tokens selecting input arbitrarily

outcomes: → 5 4 3 2 or
3 2 5 4 or
3 5 4 2 or …

i.e., behavior is not a function !

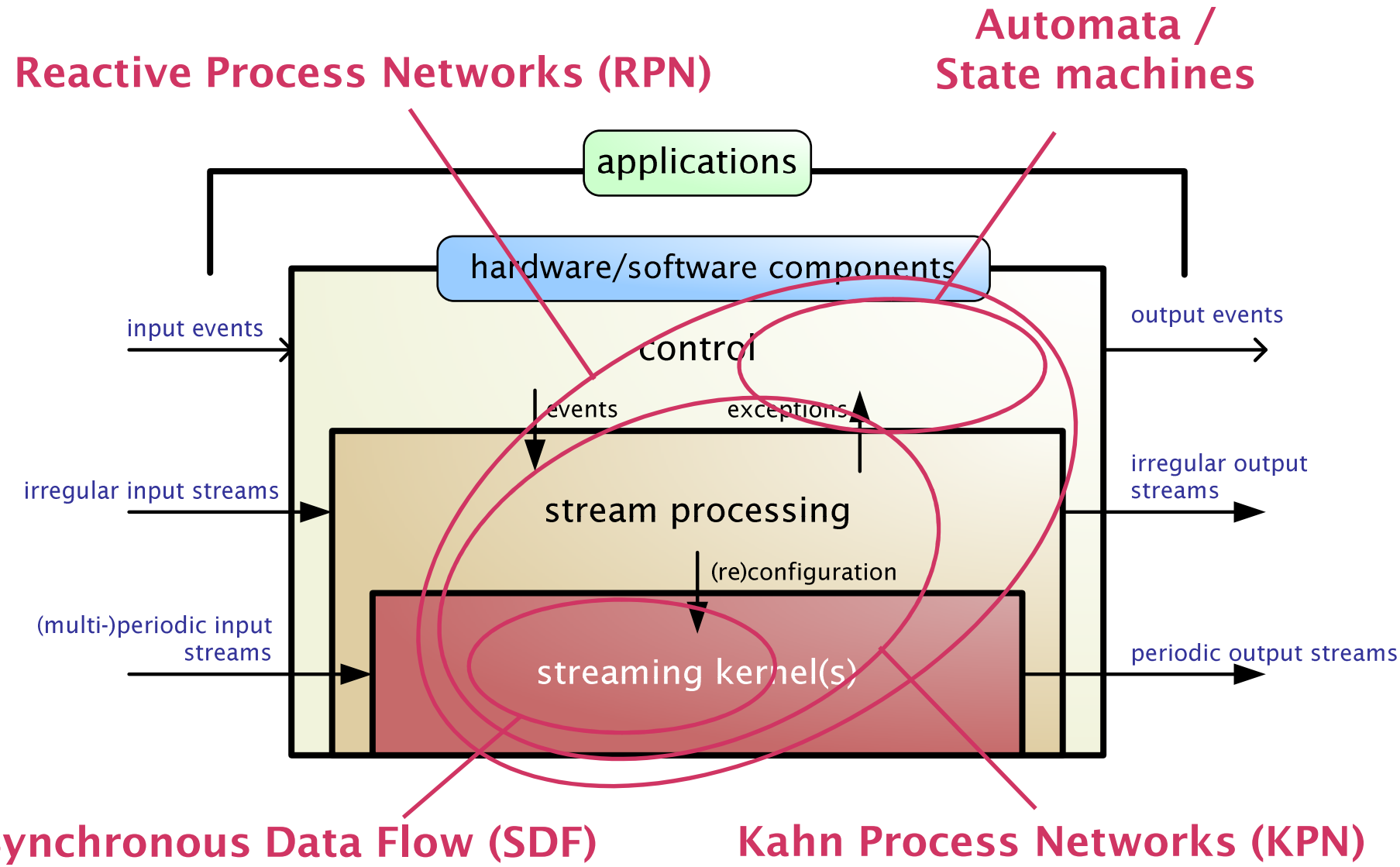


Reactive Process Networks

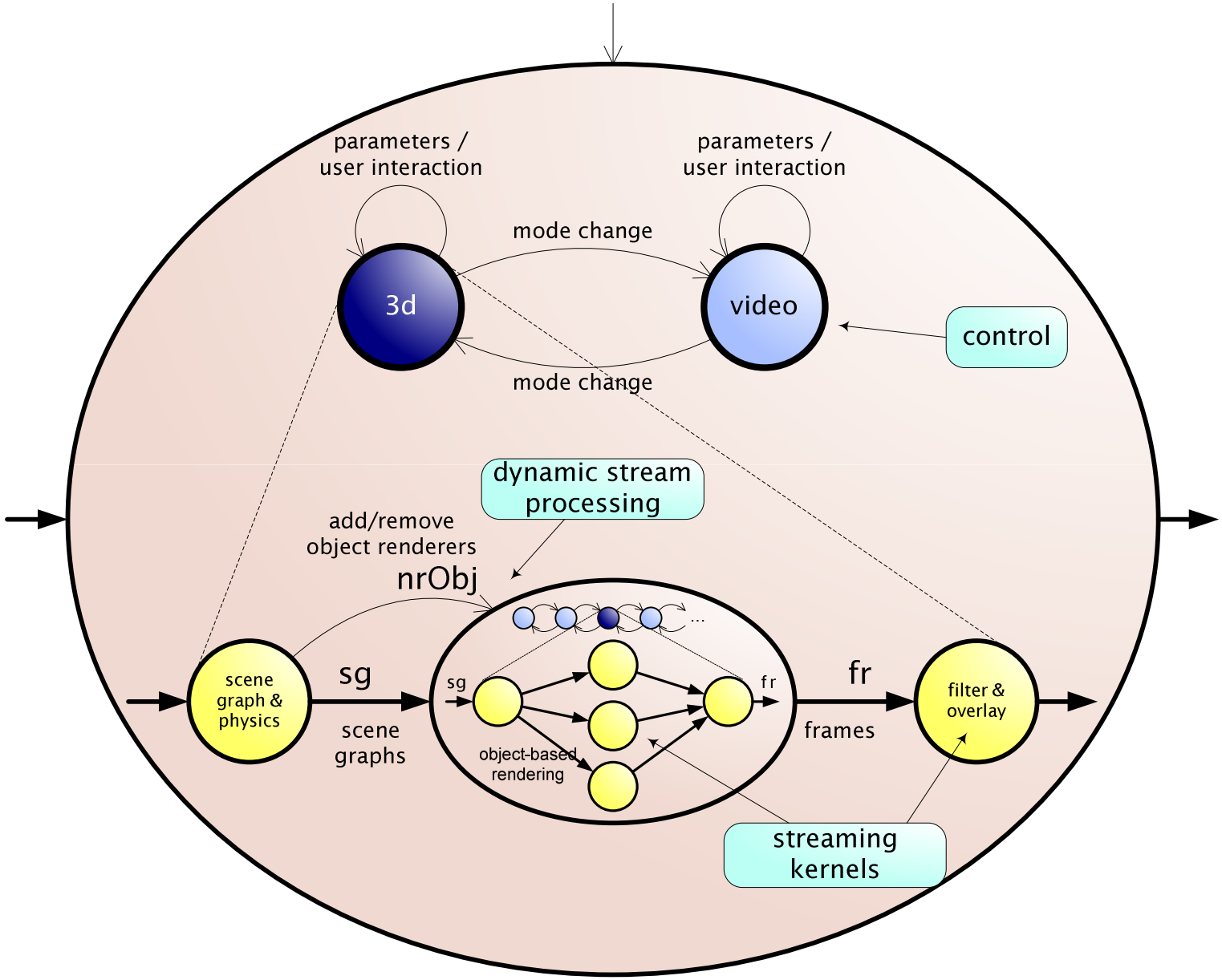


[M.C.W. Geilen and T. Basten. Reactive Process Networks. Proc. EMSOFT 2004.]

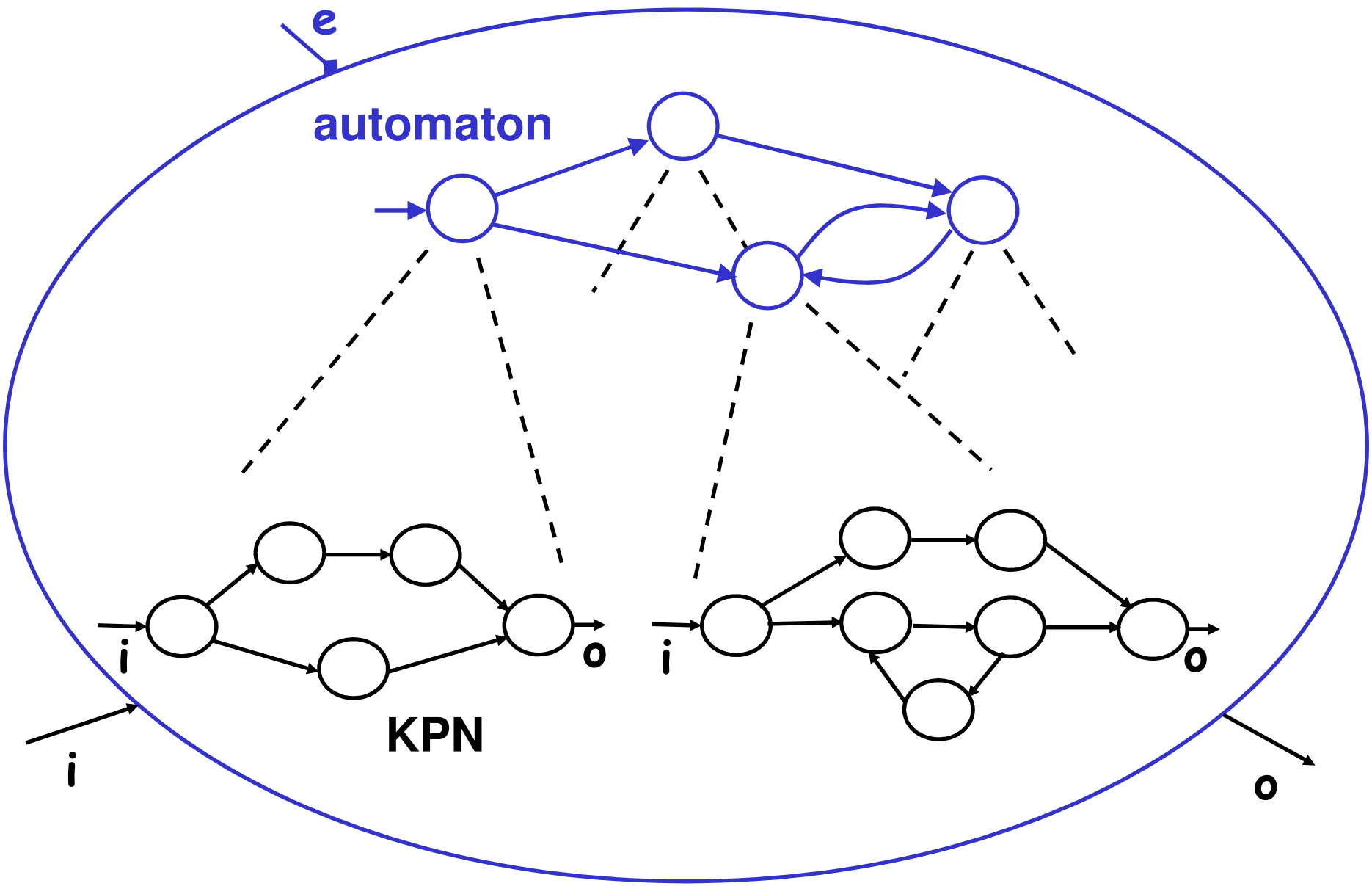
48 Application domain



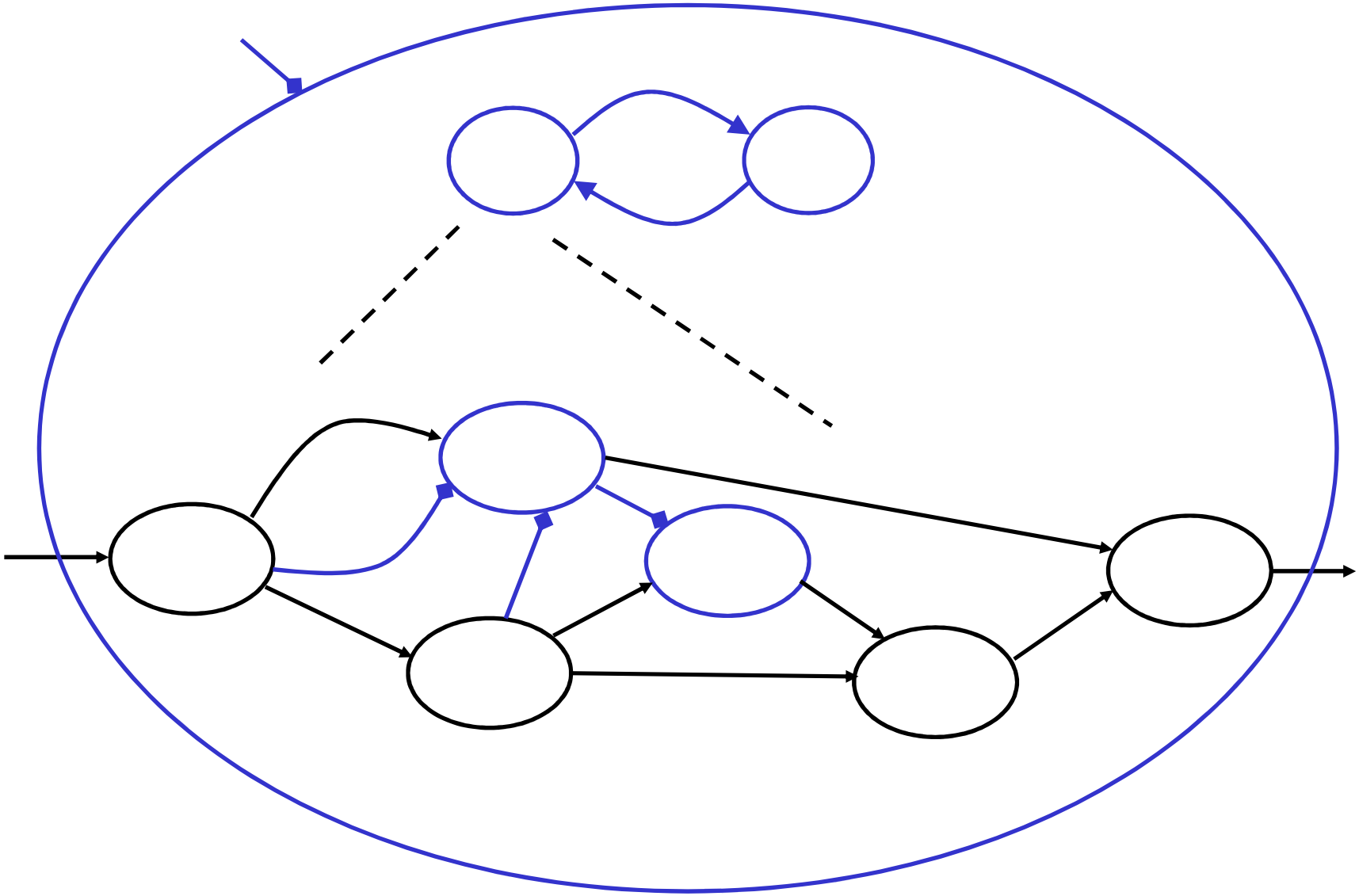
49 A gaming example



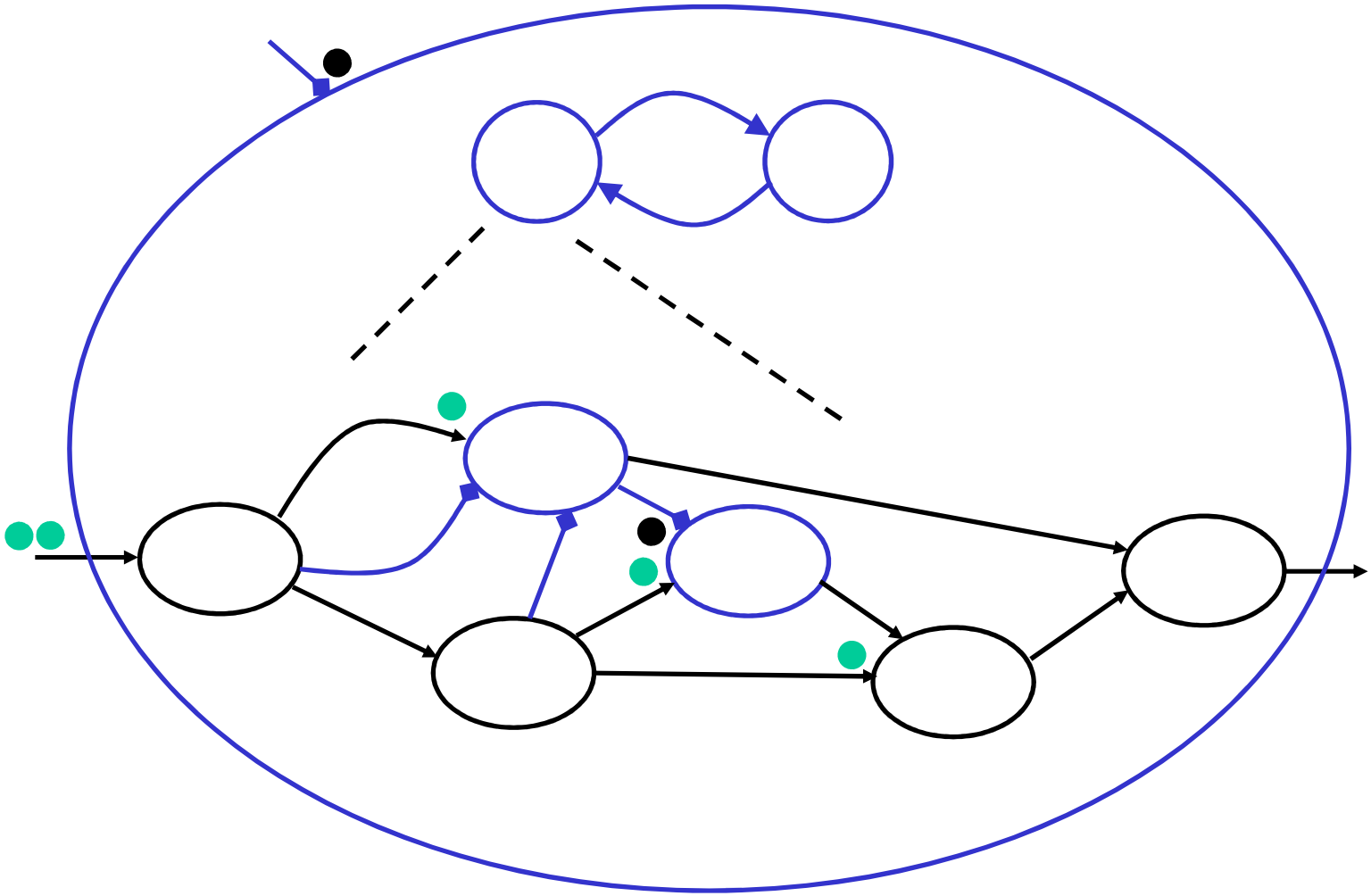
50 Reactive Process Networks



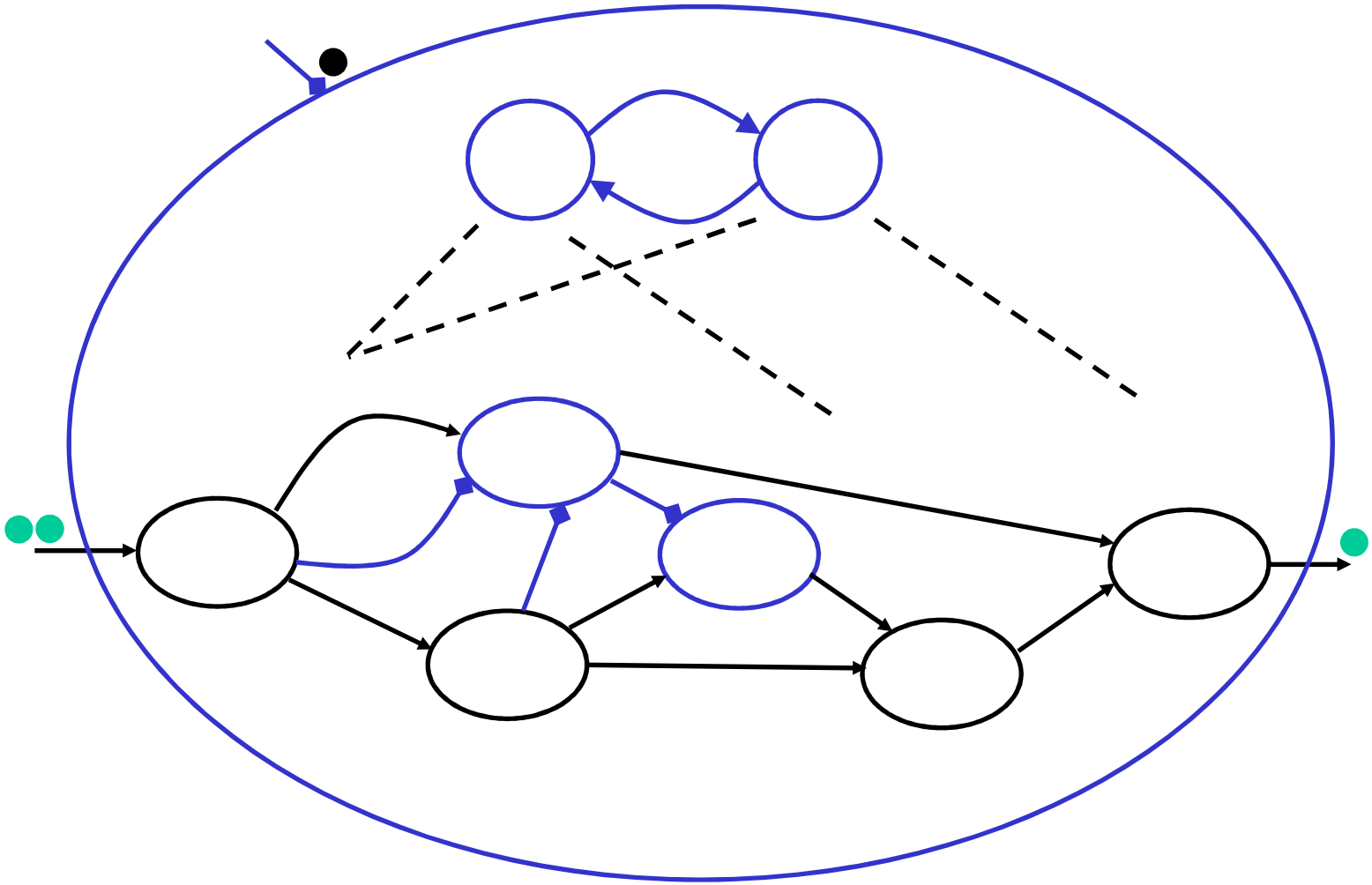
51 Reactive Process Networks



52 Operational semantics



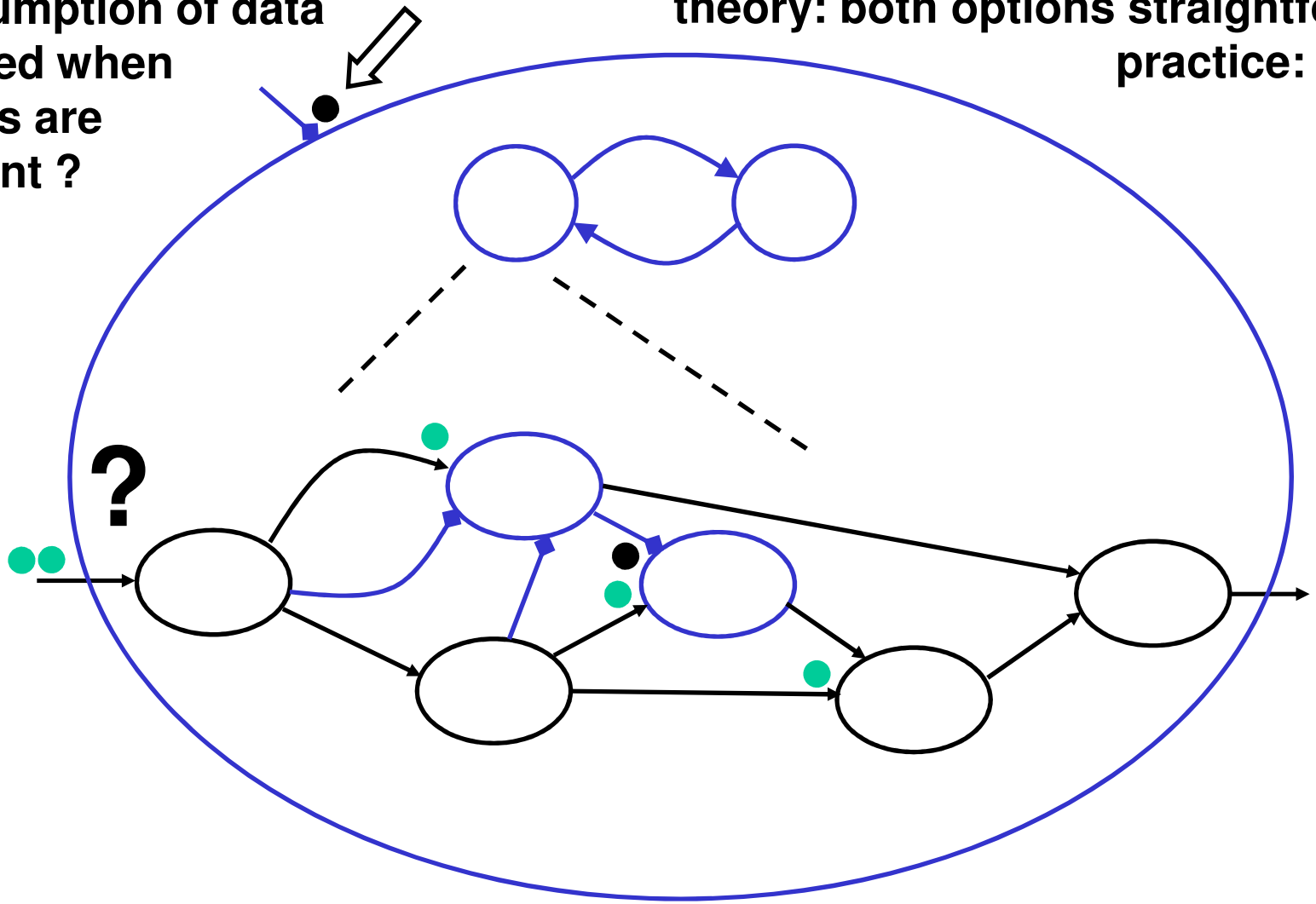
53 Operational semantics



54 Prioritizing events

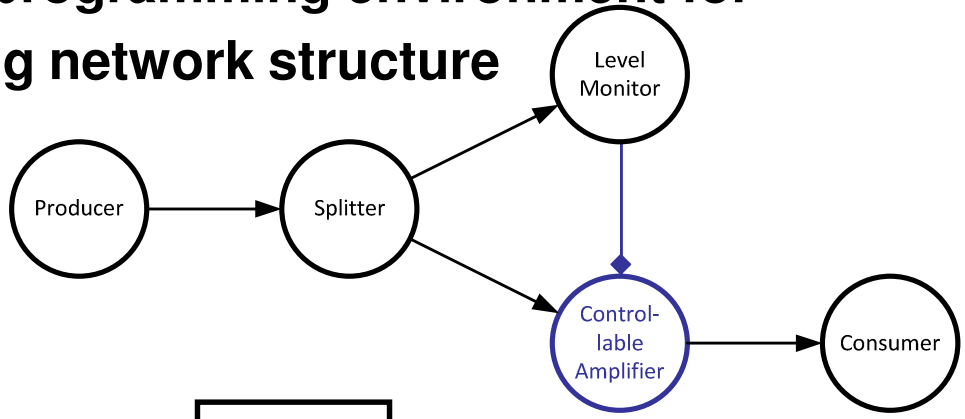
consumption of data allowed when events are present ?

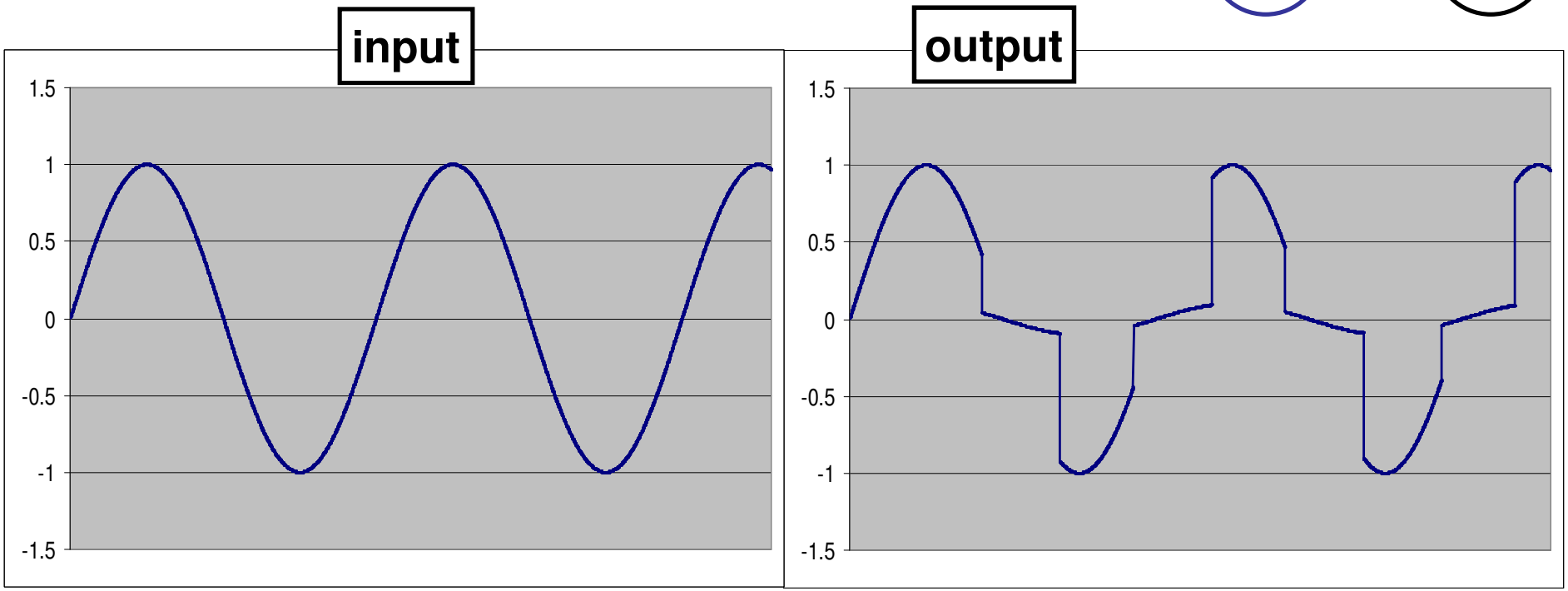
theory: both options straightforward
practice: choice



55 Prototype implementation

- prototype implementation of a programming environment for reconfigurations not affecting network structure
- events have priority over data



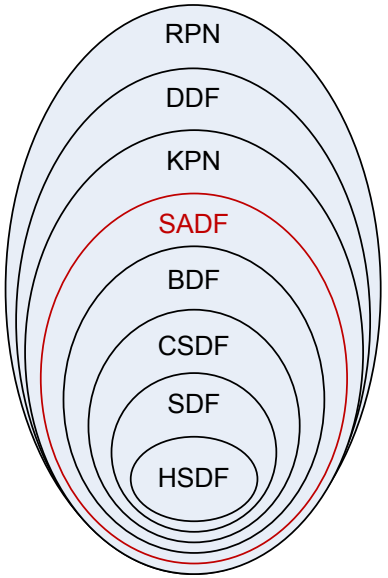


56 Open issues

- prototype implementation of arbitrary reconfigurations
- case studies (programming styles, expressivity)
- efficient implementation of reconfiguration
- deadlock analysis, scheduling, fifo management
- distribution
- **analyzable subclasses** (BDF, **SDF**)

An analyzable subclass of RPN

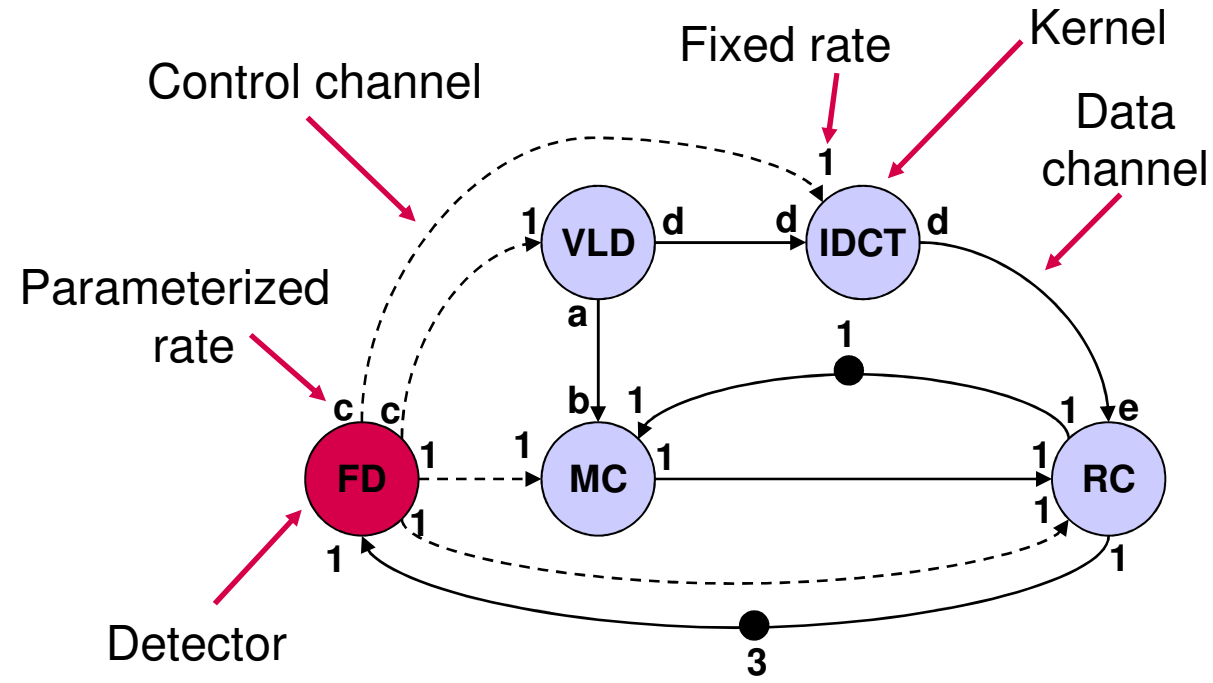
Scenario-Aware DataFlow

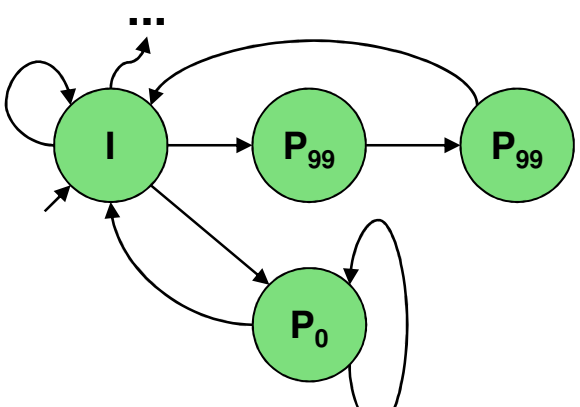


[B.D. Theelen et al., A Scenario-Aware Data Flow Model for Combined Long-Run Average and Worst-Case Performance Analysis. Proc. MEMOCODE, 2006.]

58 Scenario-Aware DataFlow (SADF)

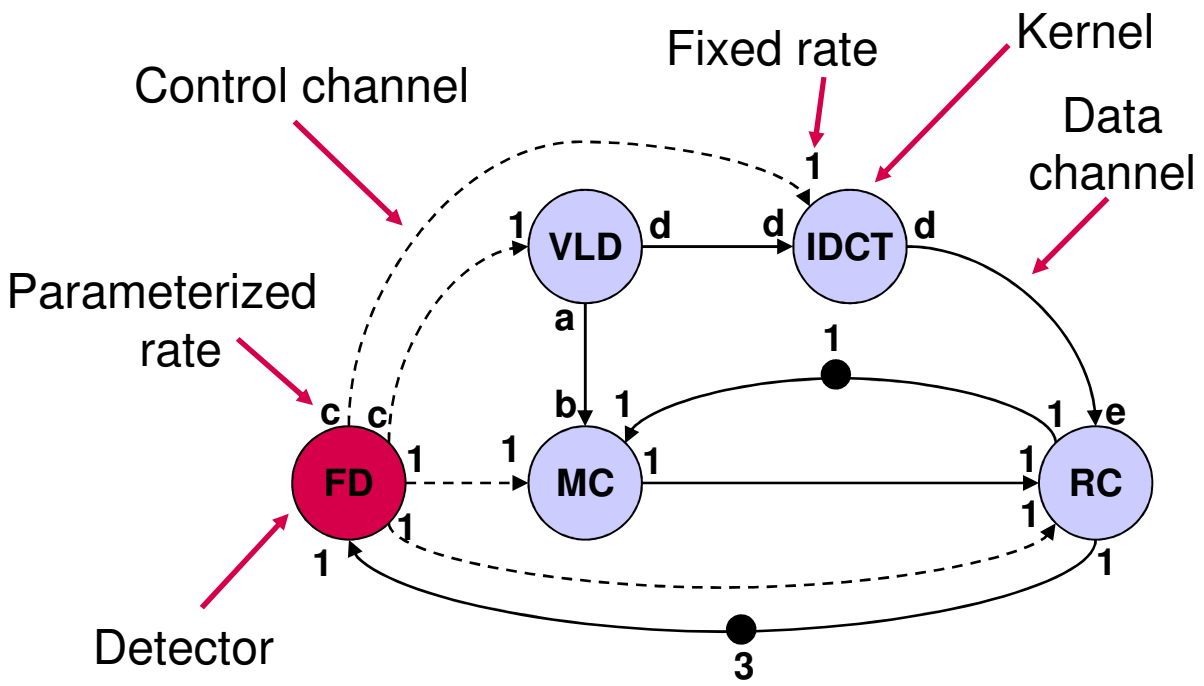
H.263 Decoder Model

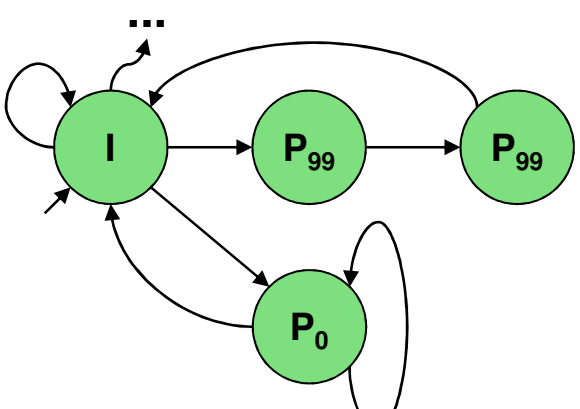




Scenario-Aware DataFlow (SADF)

H.263 Decoder Model





x={30,40,50,60,70,80,99}

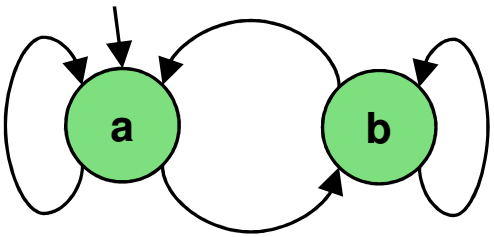
Rate	I	P_x	P_0
a	0	1	0
b	0	x	0
c	99	x	1
d	1	1	0
e	99	x	0

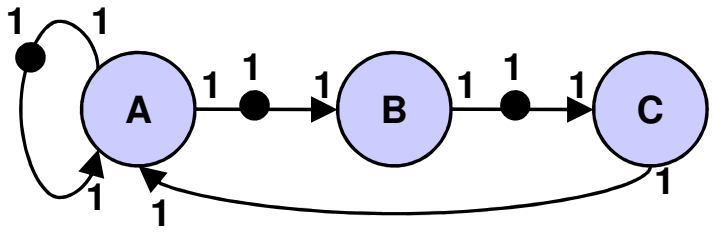
Execution time		
VLD	P_0	0
	others	40
IDCT	P_0	0
	others	17
MC	I, P_0	0
	P_{30}	90
	P_{40}	145
	P_{50}	190
	P_{60}	235
	P_{70}	265
	P_{80}	310
	P_{99}	390
RC	I	350
	P_0	0
	P_{30}, P_{40}, P_{50}	250
	P_{60}	300
	P_{70}, P_{80}, P_{99}	320
FD	All	0

Scenario-aware performance analysis

Simple SADF Model

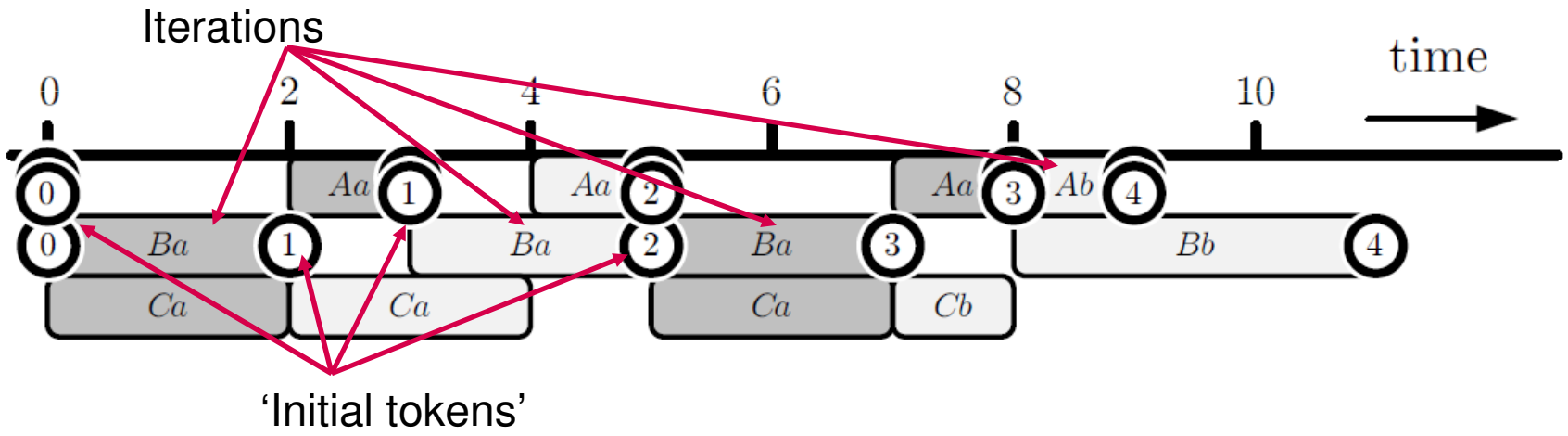
[Geilen&Stuijk, CODES+ISSS 2010]





Execution time	a	b
A	1	1
B	2	3
C	2	1

An execution, performing scenarios aaab



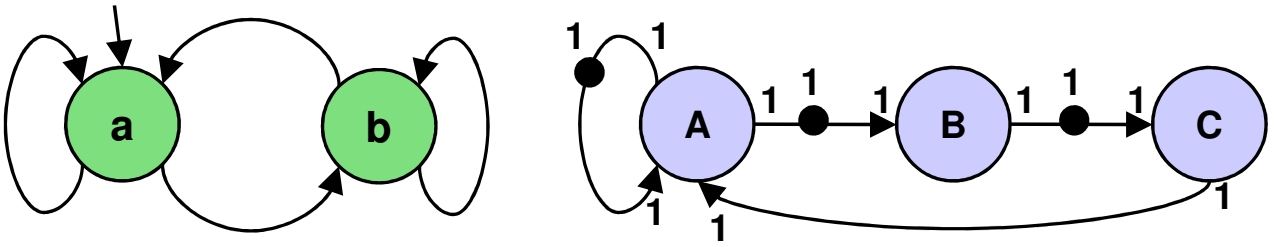
Semantics in (max,+) algebra, normalized vectors of token production times

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \quad \begin{pmatrix} -2 \\ -2 \\ 0 \end{pmatrix}$$

Scenario-aware performance analysis

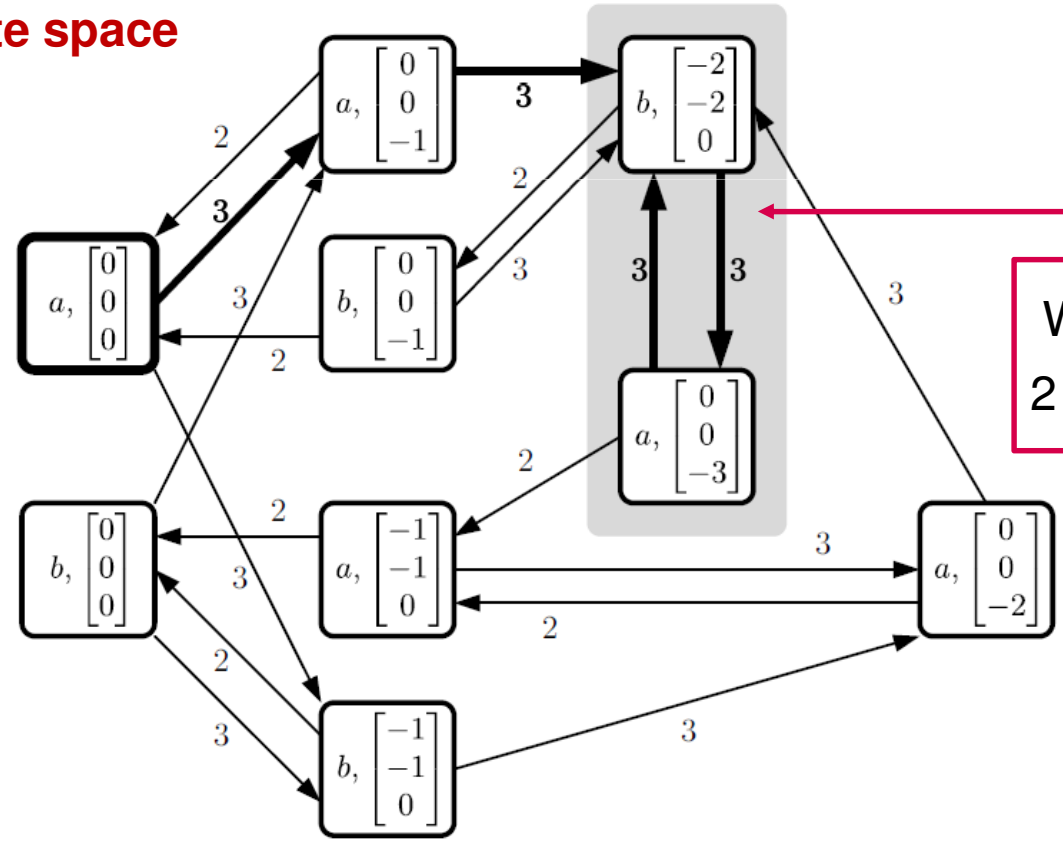
Simple SADF Model

[Geilen&Stuijk, CODES+ISSS 2010]



Execution time	a	b
A	1	1
B	2	3
C	2	1

State space



Critical cycle

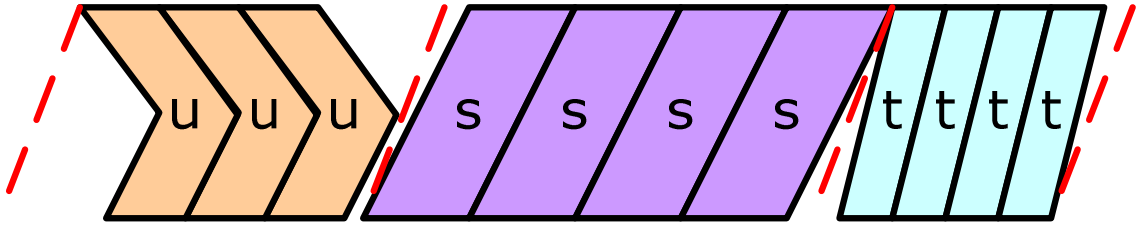
Worst-case throughput of
2 iterations per 6 time units

Scenario-aware performance analysis

[Geilen, ACM TECS 2010]

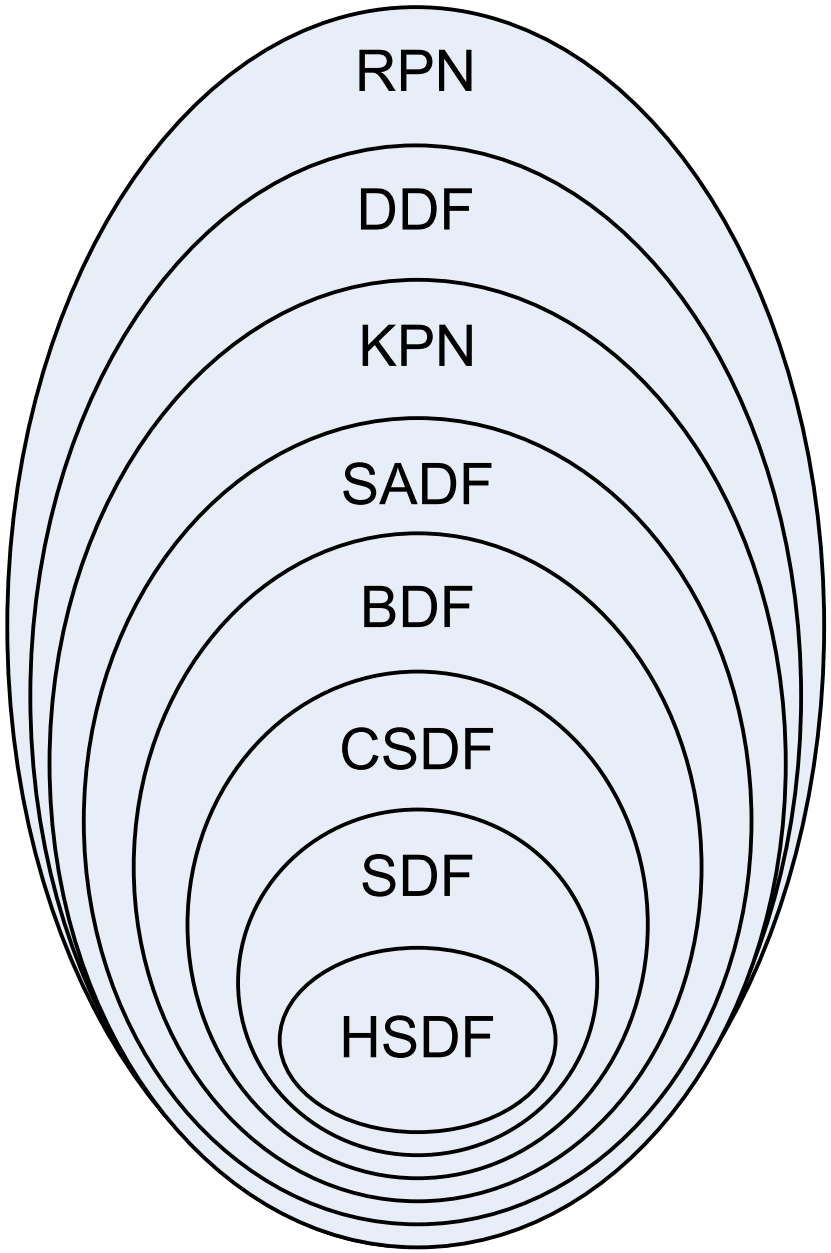
Analyzing all scenario transitions separately can be avoided

- Analyze scenarios in isolation
- Separate scenario transitions with an invariant reference schedule



The dataflow hierarchy

64 Dataflow expressiveness hierarchy



- **BDF and larger: Turing complete**
- **Better notions of expressiveness needed**
- **Unification needed**

RPN: Reactive Process Networks

KPN: Kahn Process Networks

DF: DataFlow

DDF: Dynamic DF

SADF: Scenario-Aware DF

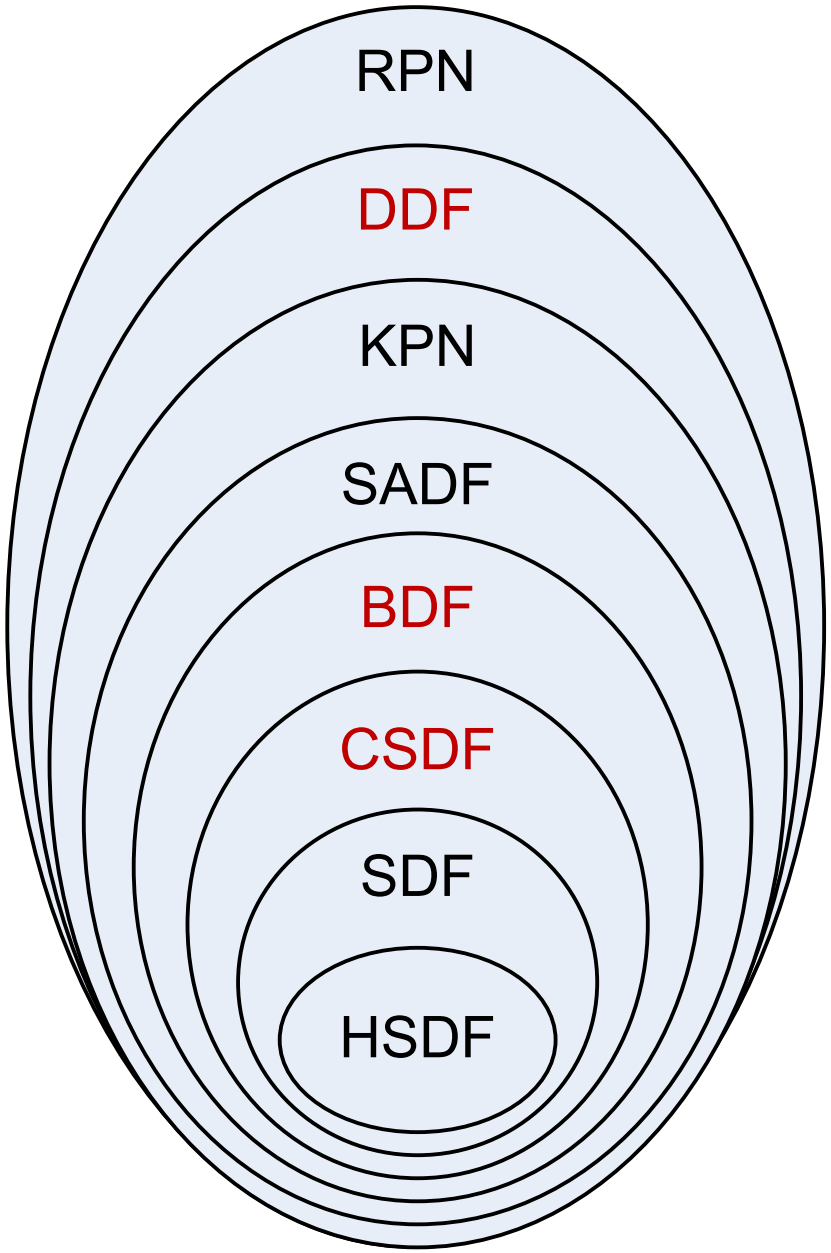
BDF: Boolean DF

CSDF: Cyclo-Static DF

SDF: Synchronous DF

HSDF: Homogeneous SDF

65 Dataflow expressiveness hierarchy



- **BDF and larger: Turing complete**
- **Better notions of expressiveness needed**
- **Unification needed**

RPN: Reactive Process Networks

KPN: Kahn Process Networks

DF: DataFlow

DDF: Dynamic DF

SADF: Scenario-Aware DF

BDF: Boolean DF

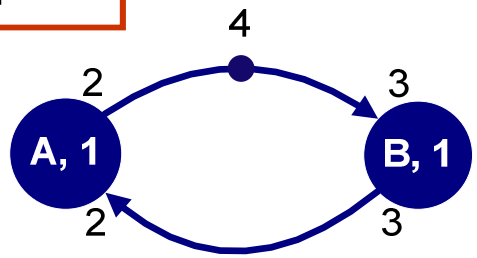
CSDF: Cyclo-Static DF

SDF: Synchronous DF

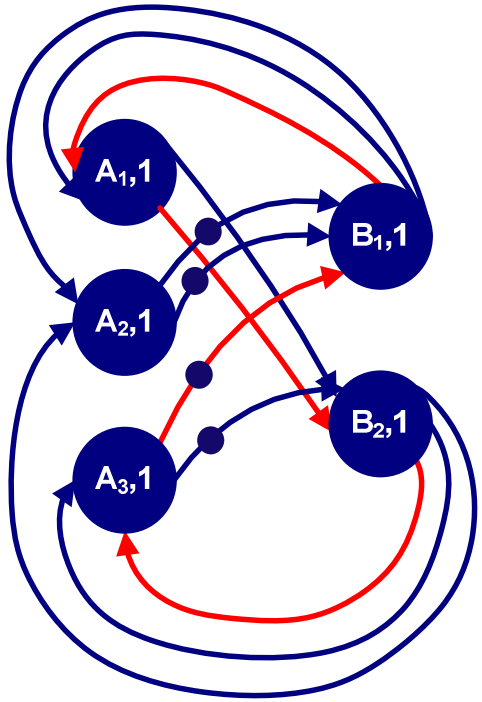
HSDF: Homogeneous SDF

66 SDF 2 ‘equivalent’ HSDF transformation

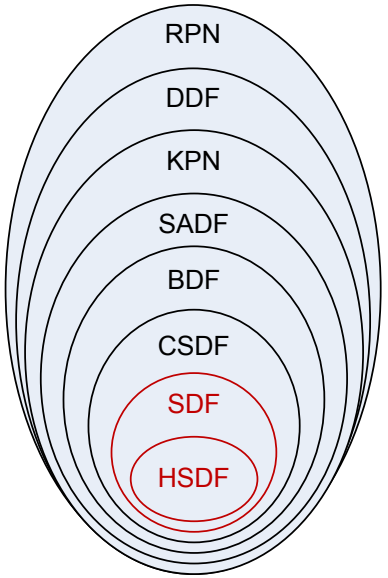
SDF



HSDF

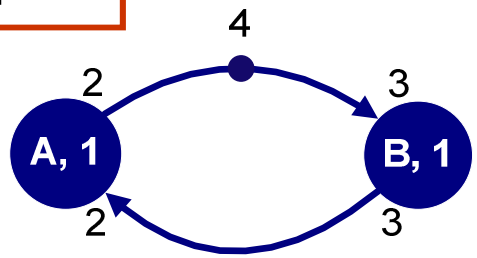


- Preserves actor firings, timing properties
- Does not preserve buffering aspects
- Algorithms operating directly on SDF may be more efficient and/or provide better results

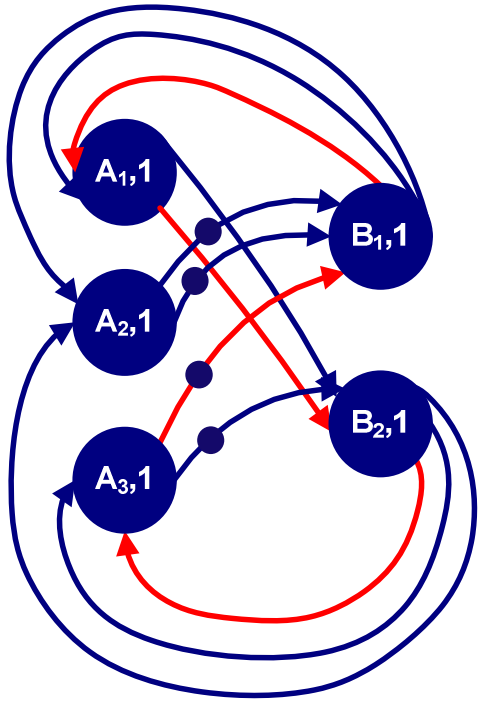


67 SDF 2 ‘equivalent’ HSDF transformation

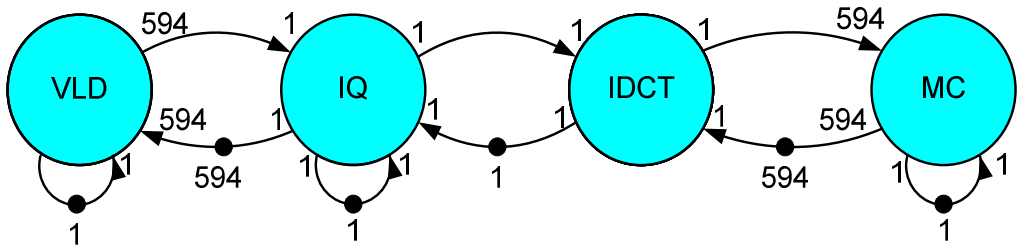
SDF



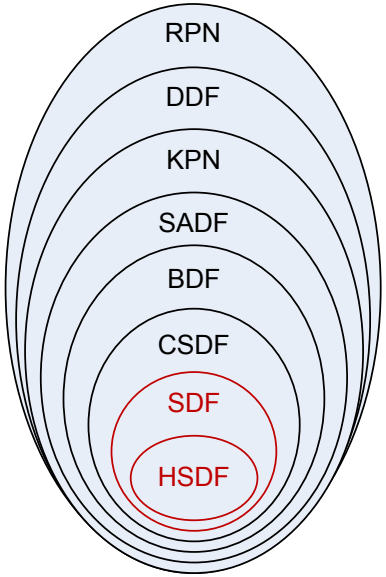
HSDF



Dataflow graph H.263 decoder

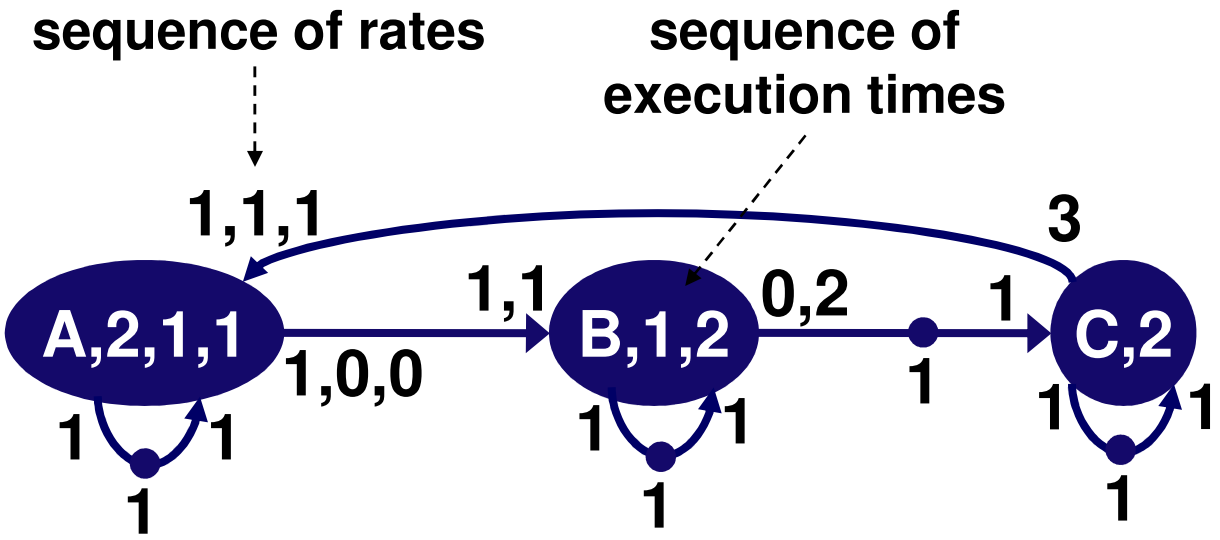


HSDF graph with 1190 actors

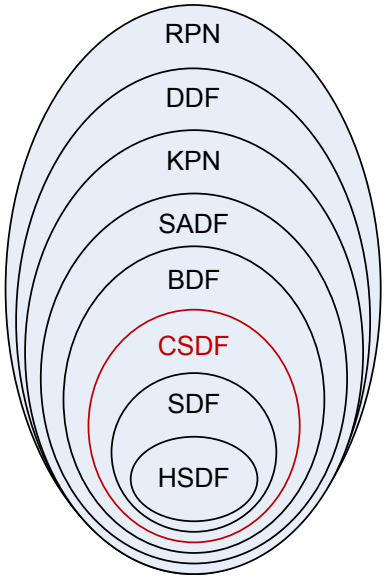


68 Cyclo-Static DataFlow

(CSDF [Bilsen et al 1996])



- Allows the specification of certain iterations
- Can be transformed into an ‘equivalent’ SDF
- Algorithms operating directly on CSDF may be more efficient and/or provide better results

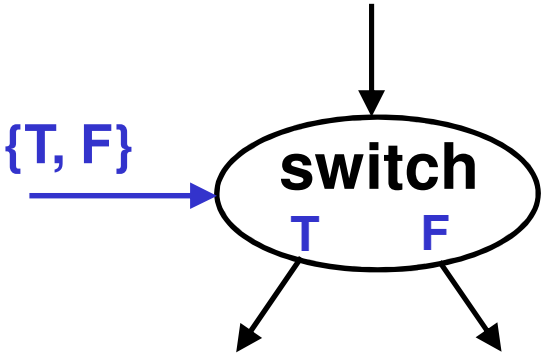


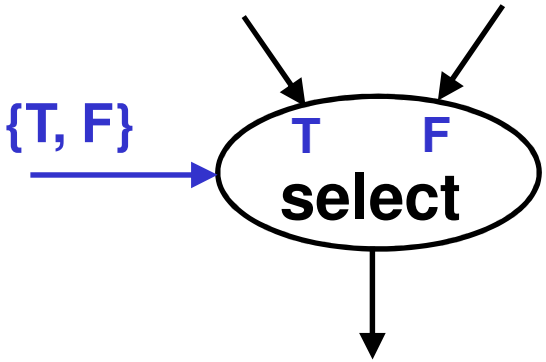
69 Boolean DataFlow

(BDF [Buck 1993])

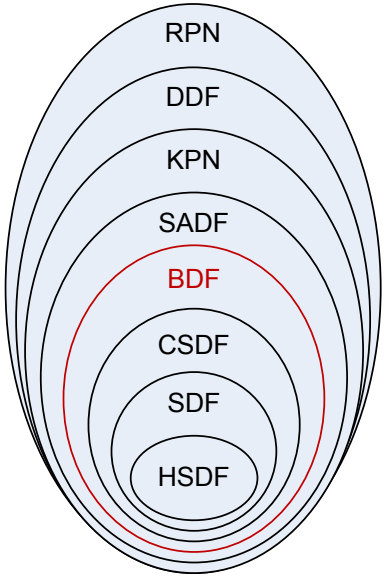
the numbers of tokens produced or consumed in a firing can be a two-valued function of a control token

for example:





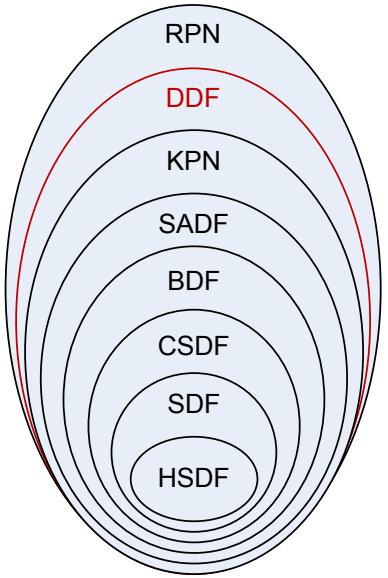
- Allows the specification of conditions and iterations
- Cannot, in general, be transformed into an ‘equivalent’ SDF
- Turing-complete



70 Dynamic DataFlow

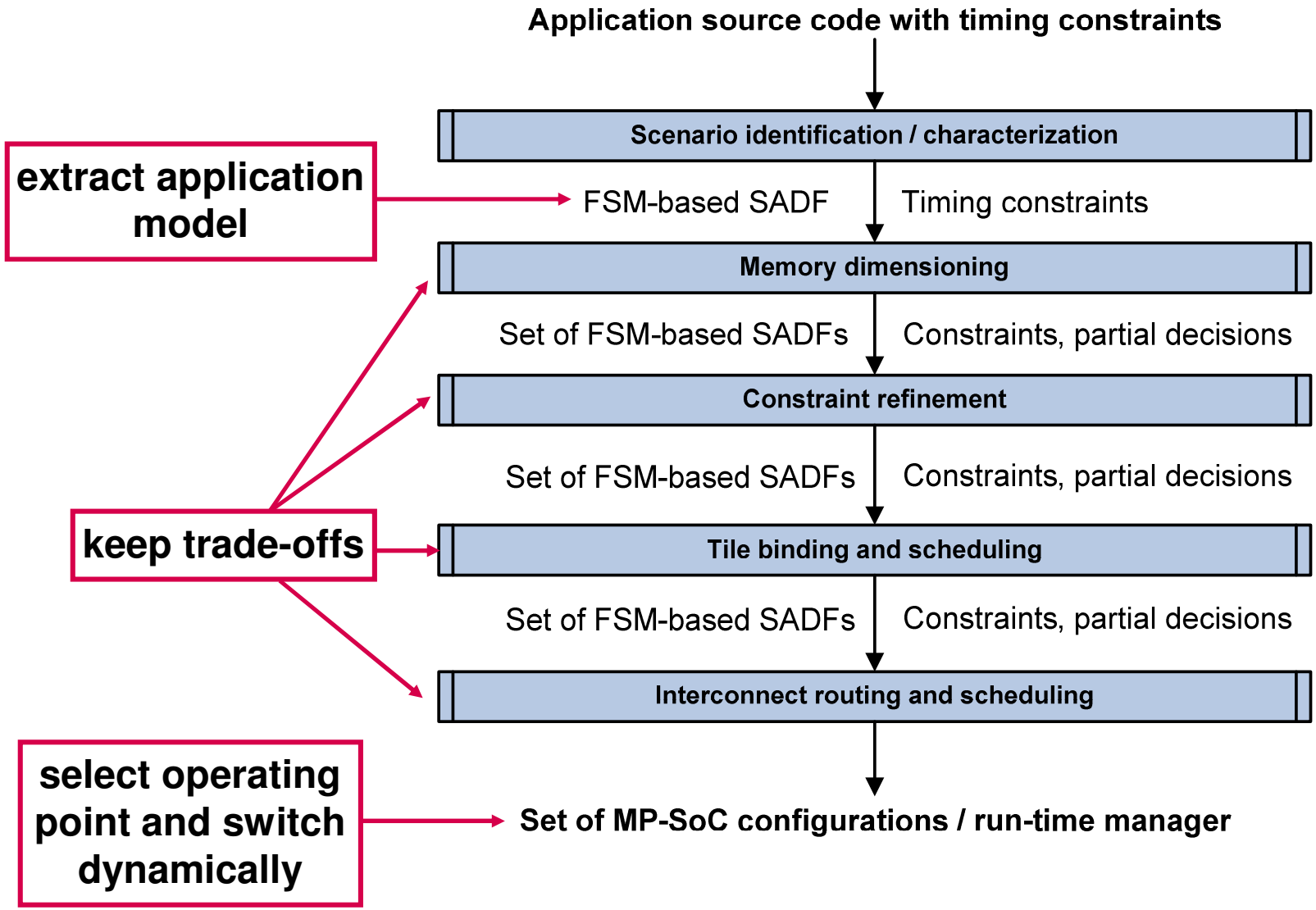
(DDF [Buck 1993])

the numbers of tokens produced or consumed in a firing may vary



Looking Forward

A model-driven scenario-aware design flow

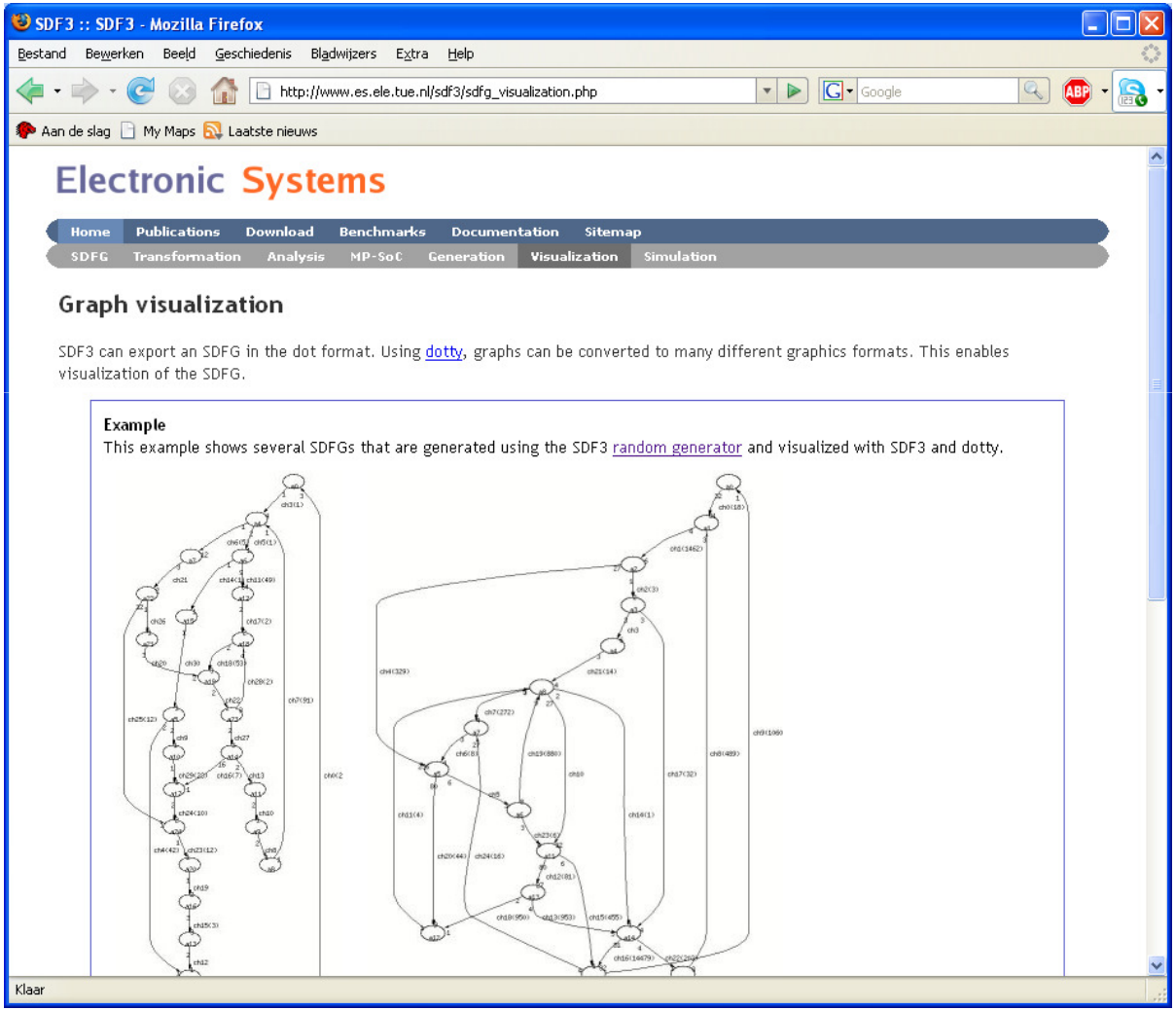


[Stuijk et al., DSD 2010]

SDF3 toolkit

Sander Stuijk (lead developer)

www.es.ele.tue.nl/sdf3



Hands-on this afternoon

[Stuijk et al., ACSD 2006]

Challenges

- Suitable notions of expressivity
- Expressivity while maintaining analyzability and synthesizability
- Abstraction without loosing accuracy

- Composability and compositionality
- MoCs for non-functional aspects
- Unification of MoCs

- Multi-objective trade-off analysis
- Parametric analysis

- Model-driven design and synthesis flows
- Model-driven run-time systems

Thank you !

Questions ?

More info: www.es.ele.tue.nl/~tbasten/
www.es.ele.tue.nl/~mgeilen/
Handbook of Signal Proc Systems, part 4

SDF3 toolkit: www.es.ele.tue.nl/sdf3/

‘An understanding of the natural world and what's in it
is a source of not only a great curiosity but great fulfillment.’
David Attenborough