VALIDATION & VERIFICATION OF SYSTEM DESIGN SPECIFICATIONS

Monika Heiner

BTU Cottbus,
Dep. of Computer Science

OUTLINE

- **V&V needs formal semantics**

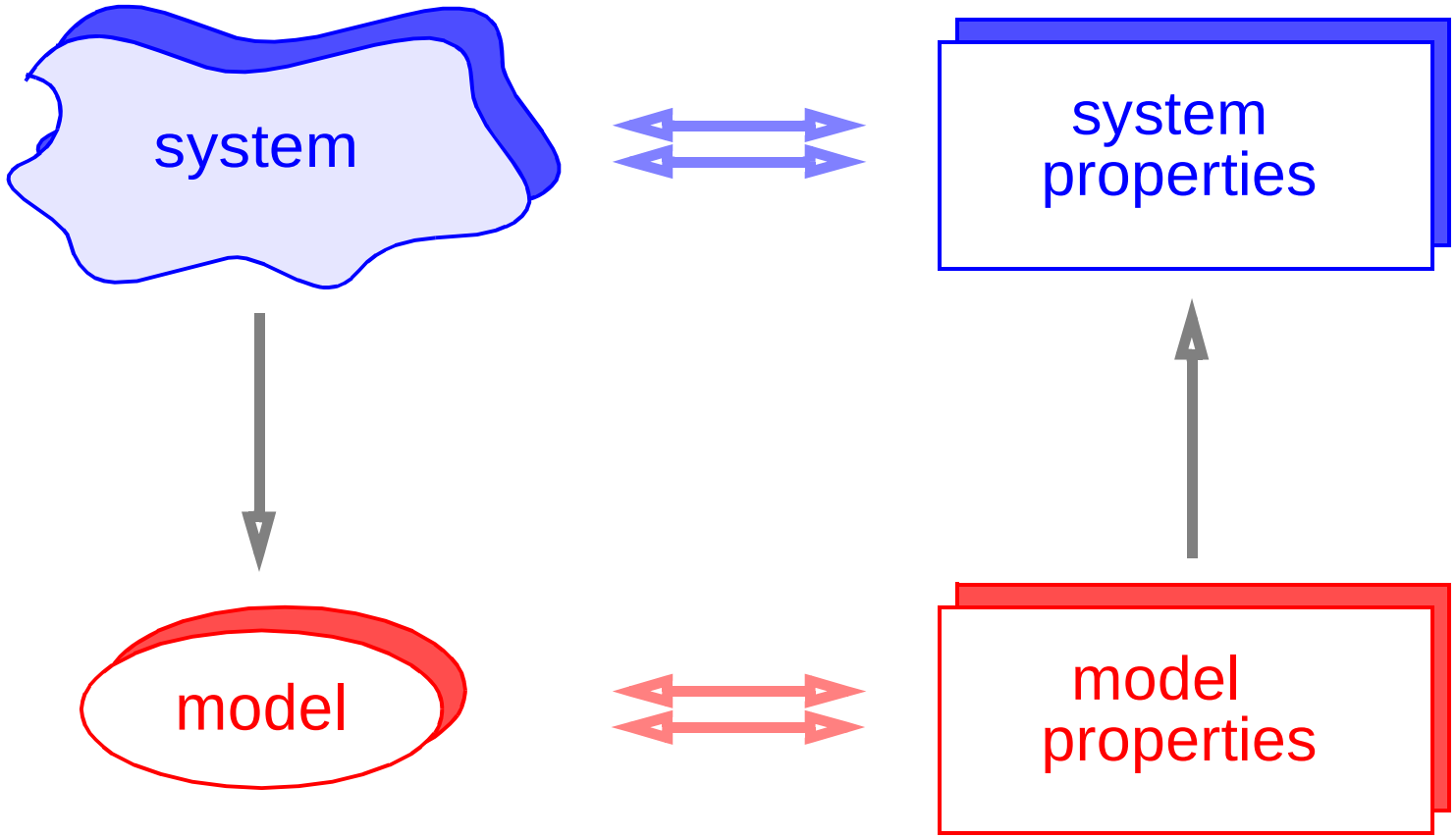
  -> *model-based system design*

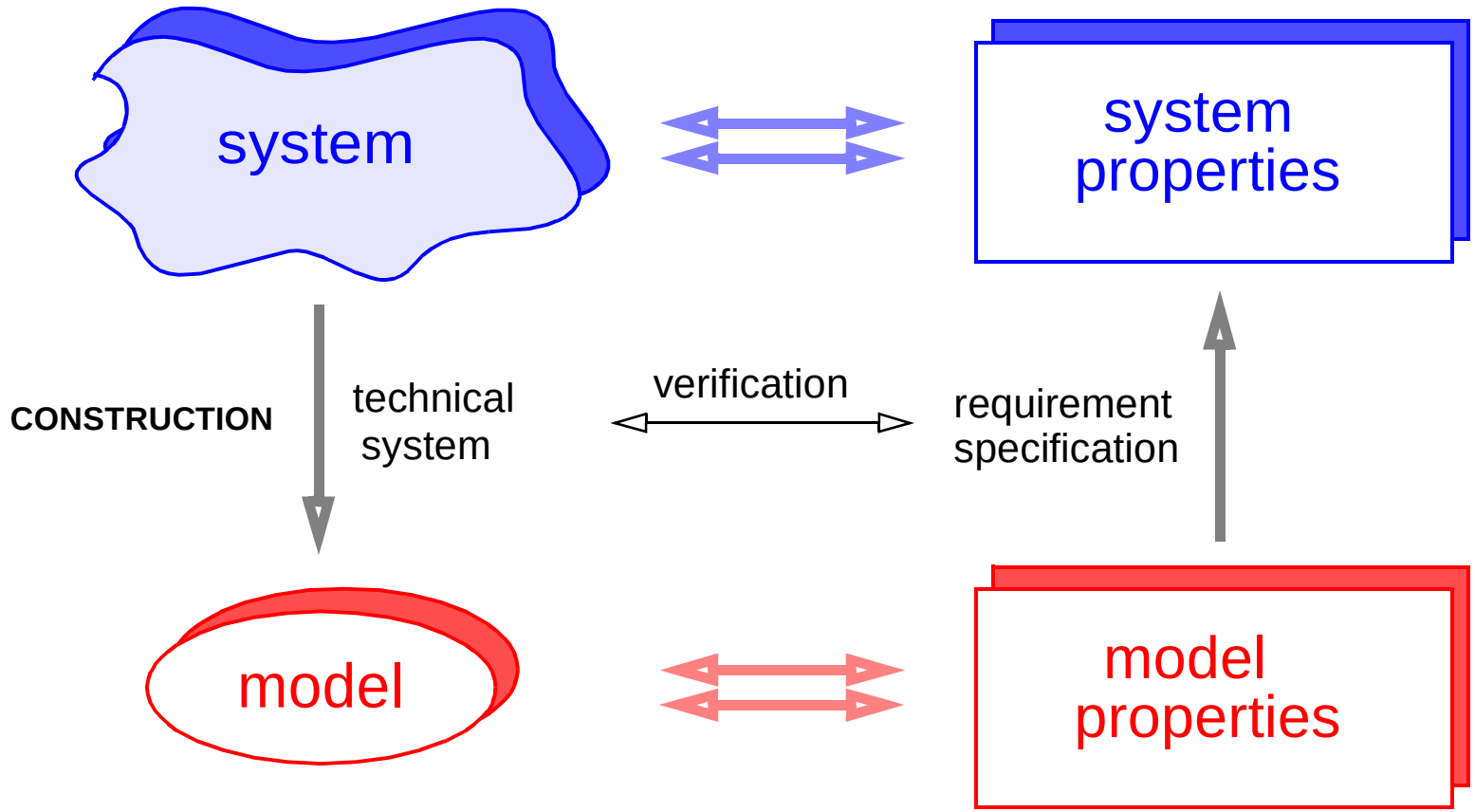
- **possible formal modelling techniques**

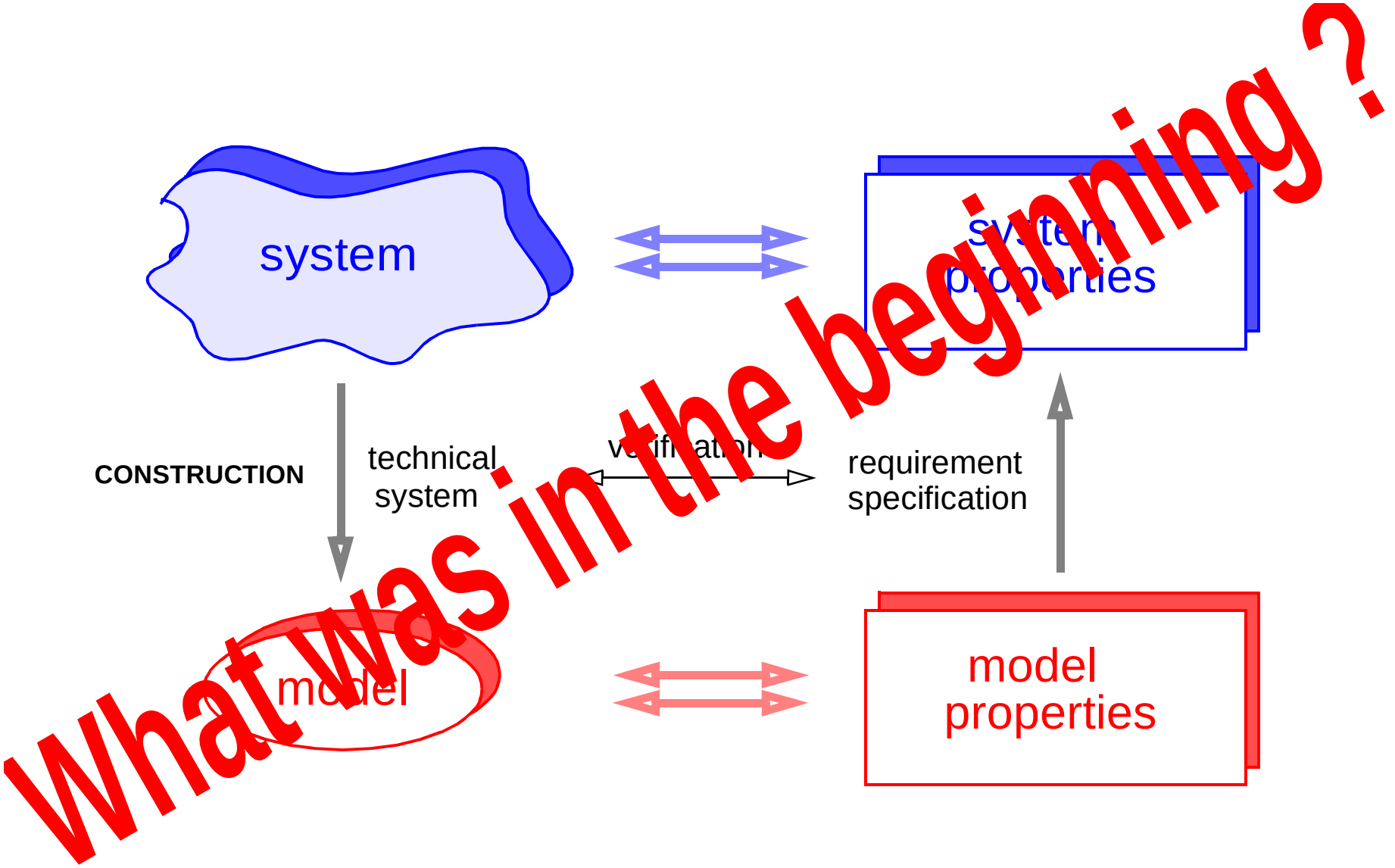
  . . . many . . .

OUTLINE

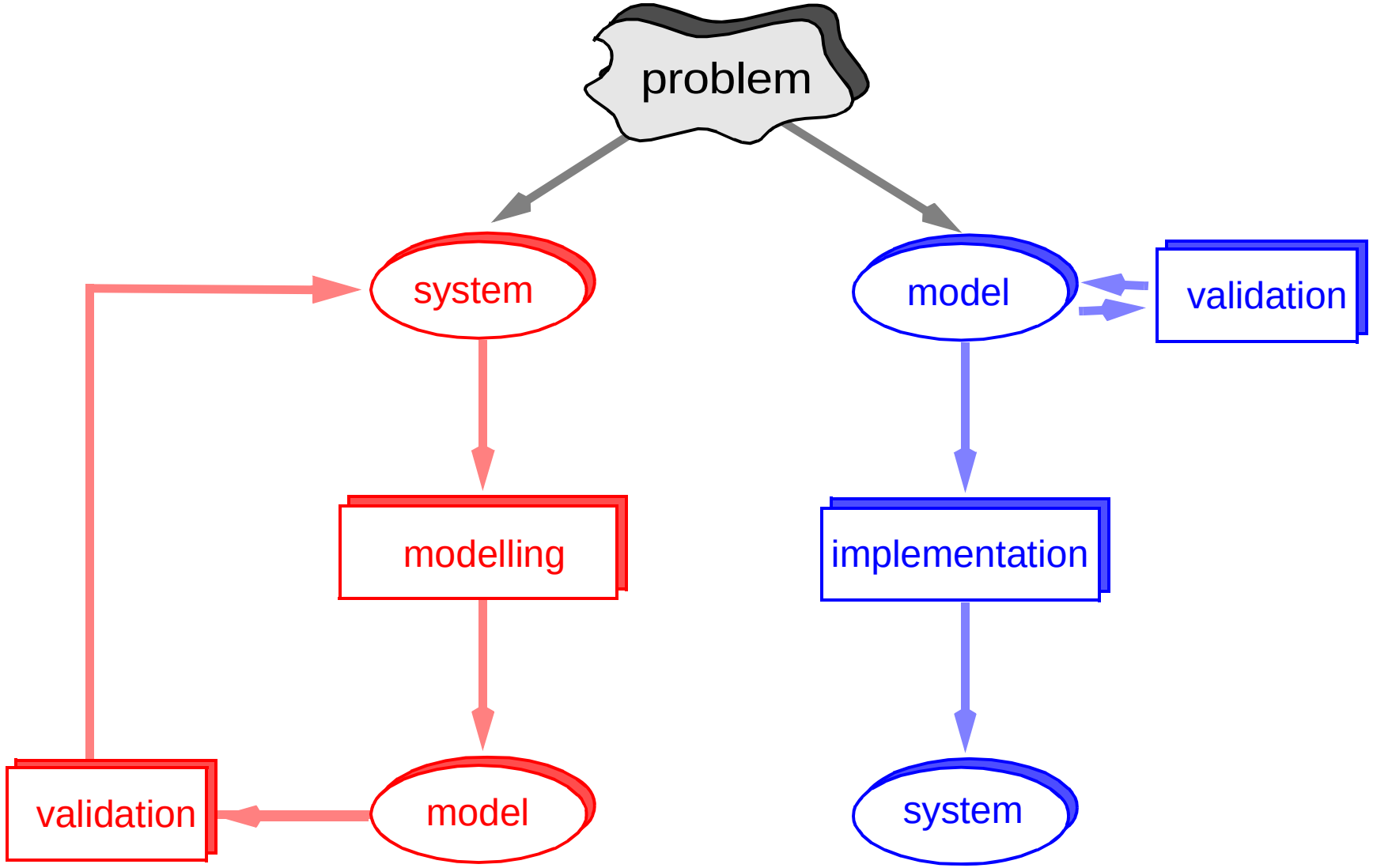
- **V&V needs formal semantics**
  -> *model-based system design*
- **possible formal modelling techniques**
  . . . many . . .
  -> *Petri nets*
- **introduction into Petri nets**
  -> *modelling and analysis*
  -> *typical Petri net properties*
  -> *typical Petri net analysis techniques*
- **applications**
  . . . many . . .
  -> *error-correcting Petri nets*


system
system properties
model
model properties


system
system properties
CONSTRUCTION
technical system
verification
requirement specification
model
model properties


system
system properties
CONSTRUCTION
technical system
verification
requirement specification
model
model properties
What was in the beginning ?

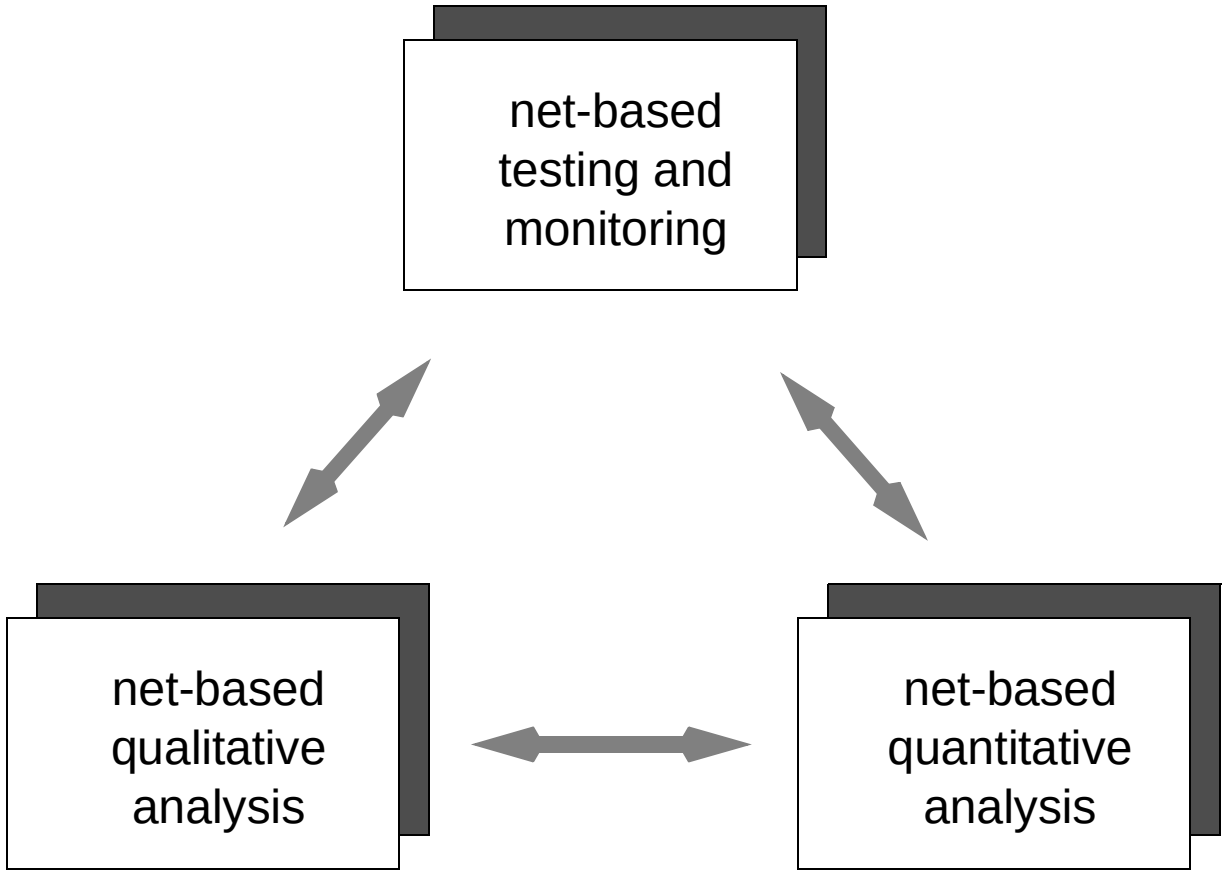
MODEL-BASED V&V, TWO APPROACHES



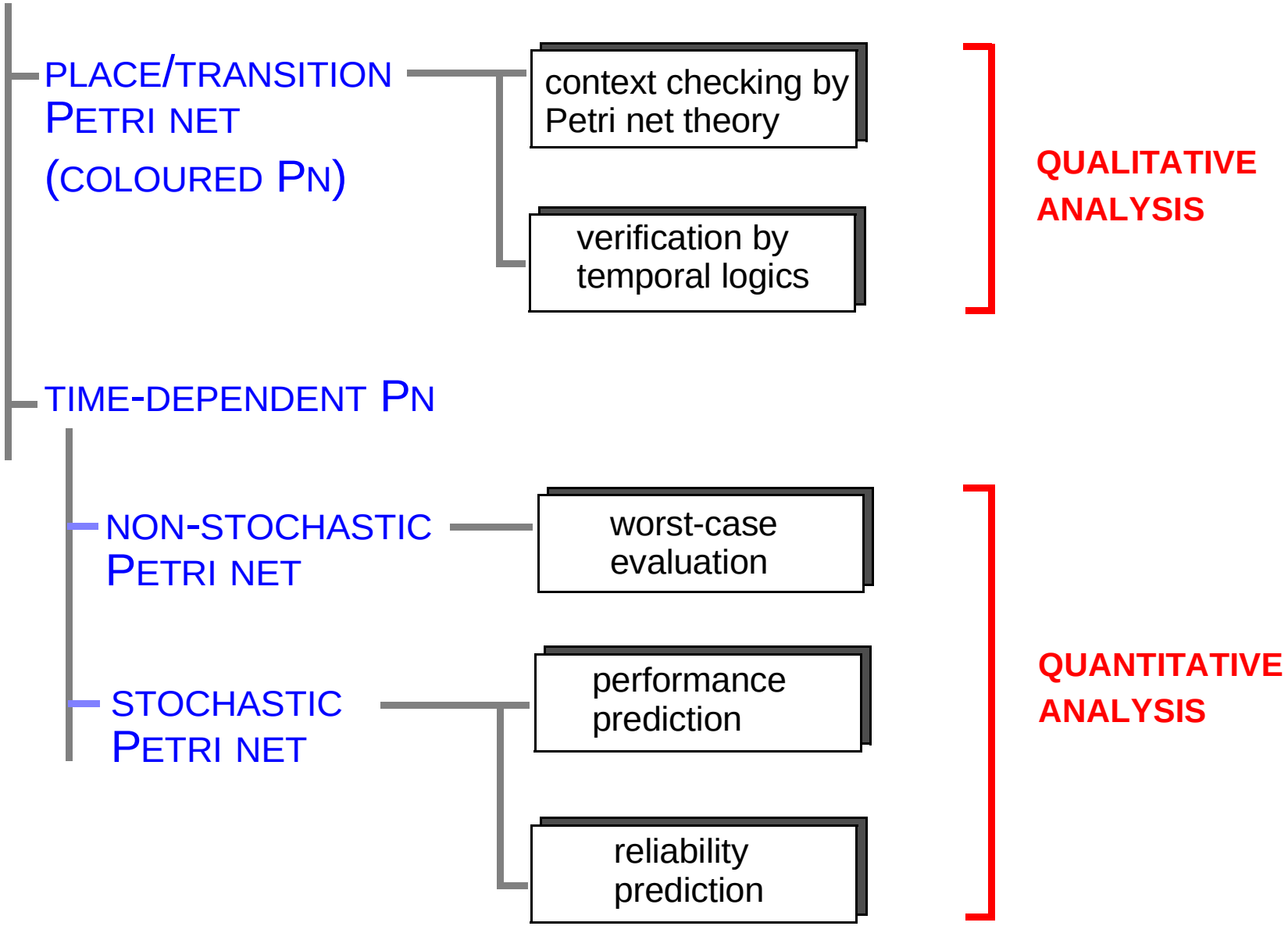
WHY PETRI NETS?

- **a suitable intermediate representation for**
  - -> *different (specification/programming) languages,*
  - -> *different phases of software development cycle,*
  - -> *different validation methods;*
- **modelling power**
  - -> *partial order (true concurrency) semantics*
  - -> *applicable on any abstraction level*
  - -> *specification of limited resources possible*
- **analyzing power**
  - -> *combination of static and dynamic analysis techniques*
  - -> *rich choice of methods, algorithms, tools*
- **BUT: modelling power <-> analyzing power**

INTEGRATION OF QUALITATIVE AND QUANTITATIVE ANALYSES



PETRI NETS, MODEL CLASSES

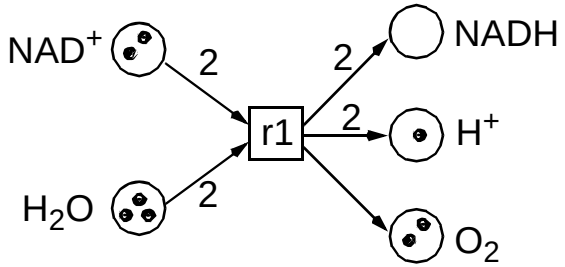


Petri nets,
a crash course

- **atomic actions -> Petri net transitions -> chemical reactions**

$$2\ NAD^+ + 2\ H_2O \rightarrow 2\ NADH + 2\ H^+ + O_2$$

input compounds

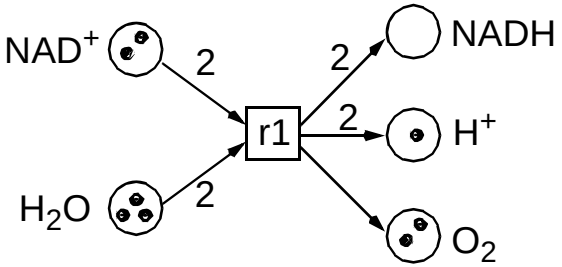


output compounds

- **atomic actions -> Petri net transitions -> chemical reactions**

$$2\ NAD^+ + 2\ H_2O \rightarrow 2\ NADH + 2\ H^+ + O_2$$

input compounds



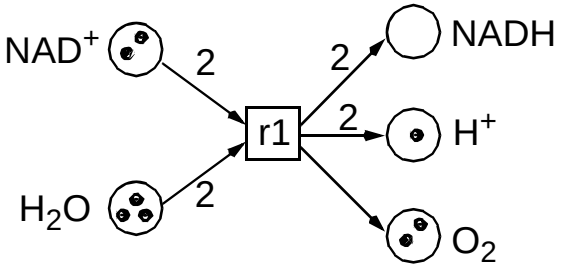
output compounds

- **local conditions -> Petri net places -> chemical compounds**

- **atomic actions -> Petri net transitions -> chemical reactions**

$$2\ NAD^+ + 2\ H_2O \rightarrow 2\ NADH + 2\ H^+ + O_2$$

input compounds



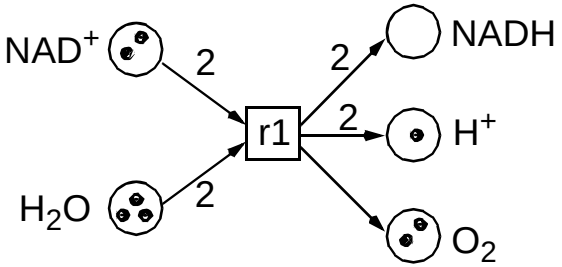
output compounds

- **local conditions -> Petri net places -> chemical compounds**
- **multiplicities -> Petri net arc weights -> stoichiometric relations**

- **atomic actions -> Petri net transitions -> chemical reactions**

$$2\ NAD^+ + 2\ H_2O \rightarrow 2\ NADH + 2\ H^+ + O_2$$

input compounds



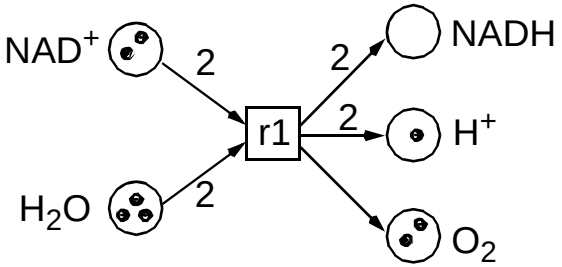
output compounds

- **local conditions -> Petri net places -> chemical compounds**
- **multiplicities -> Petri net arc weights -> stoichiometric relations**
- **condition's state -> token(s) in its place -> available amount (e.g. mol)**

- **atomic actions -> Petri net transitions -> chemical reactions**

$$2\ NAD^+ + 2\ H_2O \rightarrow 2\ NADH + 2\ H^+ + O_2$$

input compounds



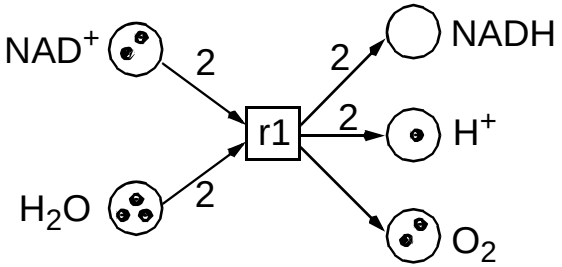
output compounds

- **local conditions -> Petri net places -> chemical compounds**
- **multiplicities -> Petri net arc weights -> stoichiometric relations**
- **condition's state -> token(s) in its place -> available amount (e.g. mol)**
- **system state -> marking -> compounds distribution**

- **atomic actions -> Petri net transitions -> chemical reactions**

$$2\ NAD^+ + 2\ H_2O \rightarrow 2\ NADH + 2\ H^+ + O_2$$

input compounds



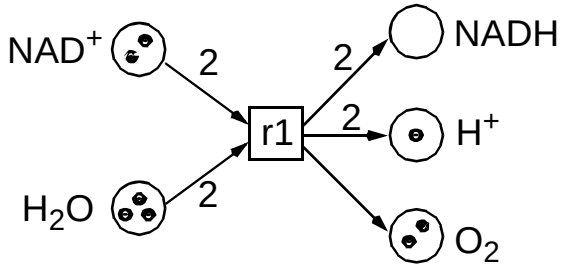
output compounds

- **local conditions -> Petri net places -> chemical compounds**
- **multiplicities -> Petri net arc weights -> stoichiometric relations**
- **condition's state -> token(s) in its place -> available amount (e.g. mol)**
- **system state -> marking -> compounds distribution**
- **PN = (P, T, F, m_0), F: (P x T) U (T x P) -> N_0, m_0: P -> N_0**

- **atomic actions -> Petri net transitions -> chemical reactions**

$$2\ NAD^+ + 2\ H_2O \rightarrow 2\ NADH + 2\ H^+ + O_2$$

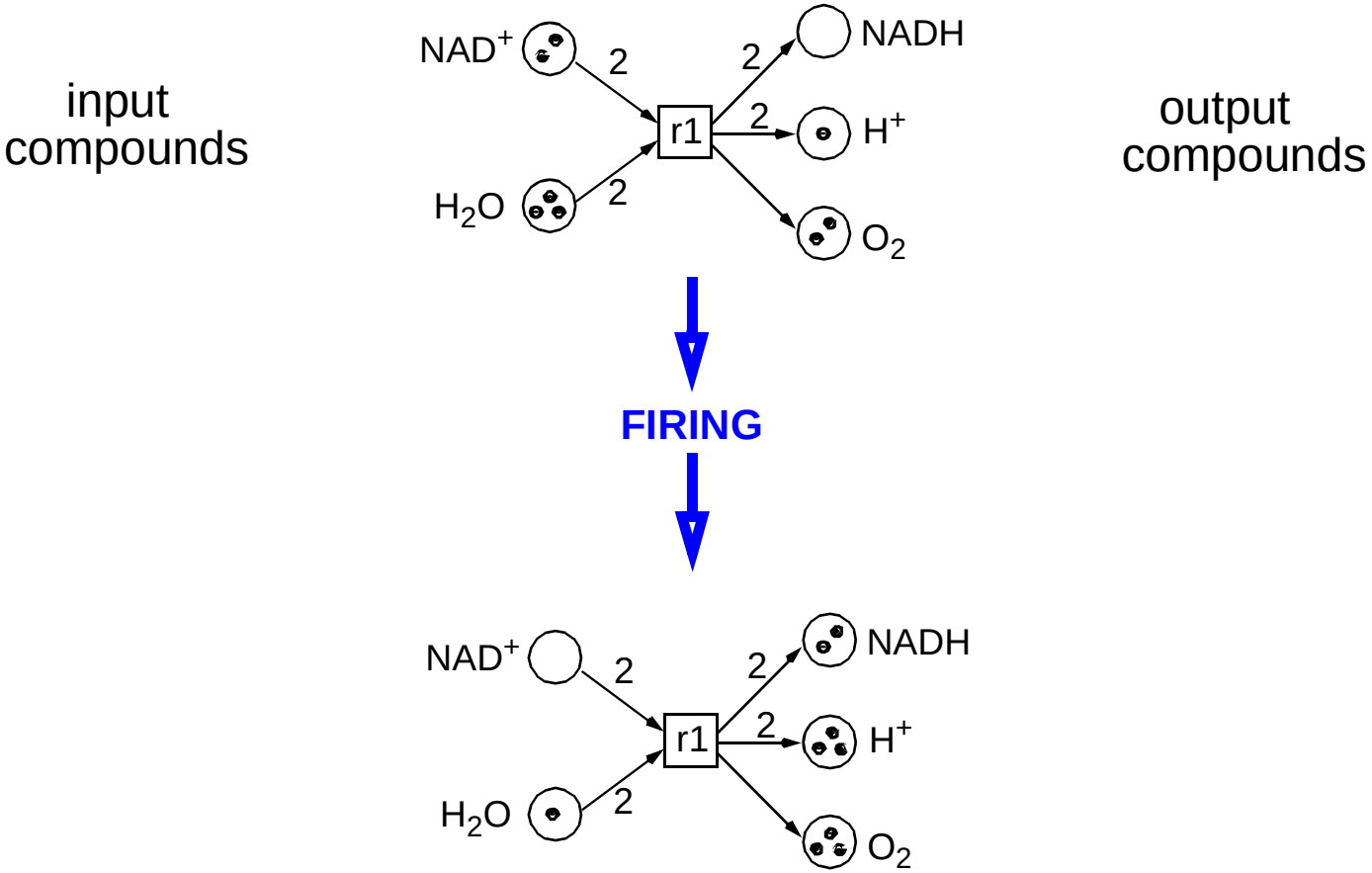
input compounds



output compounds

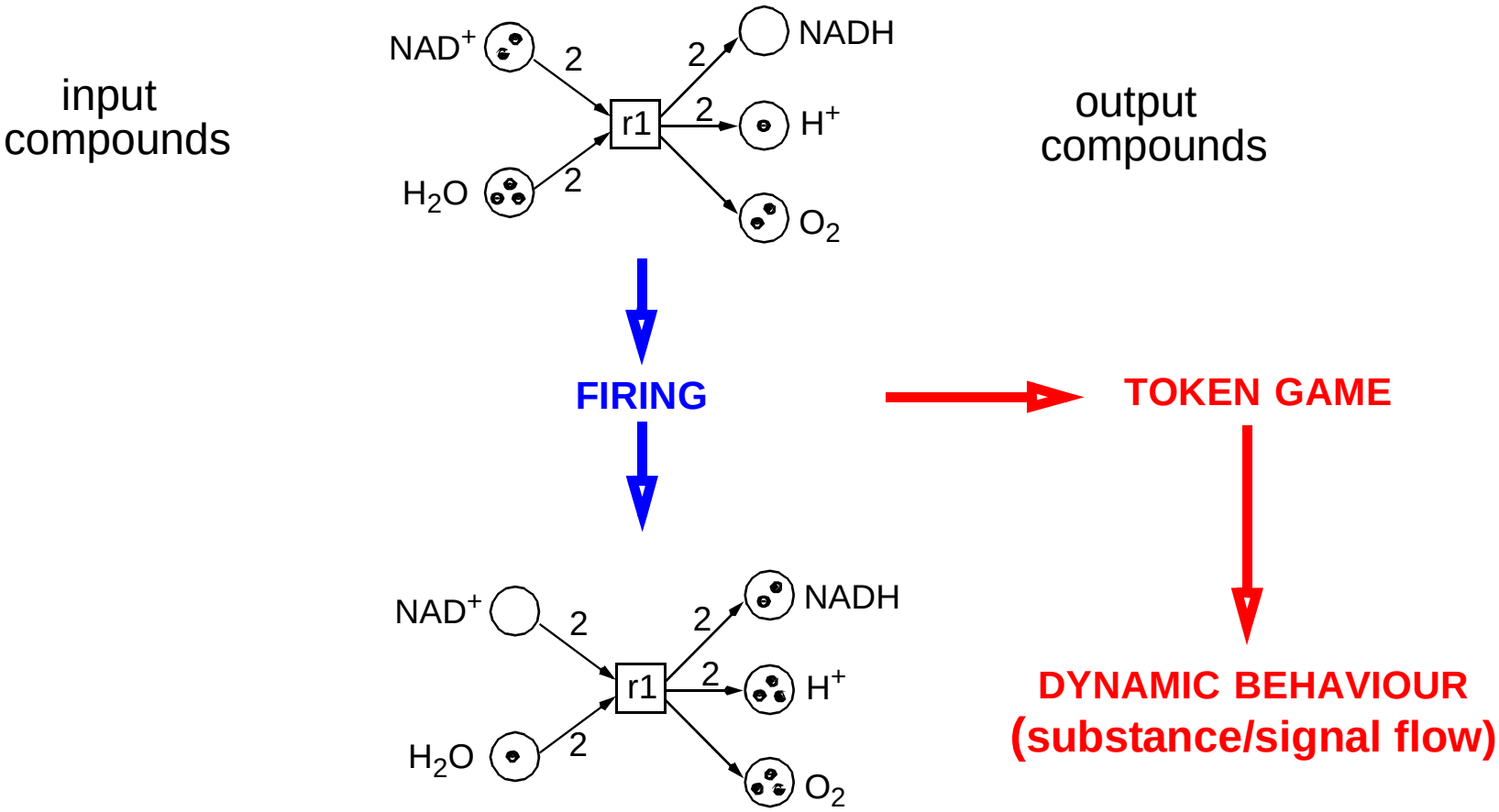
- **atomic actions -> Petri net transitions -> chemical reactions**

$$2\ NAD^+ + 2\ H_2O \rightarrow 2\ NADH + 2\ H^+ + O_2$$

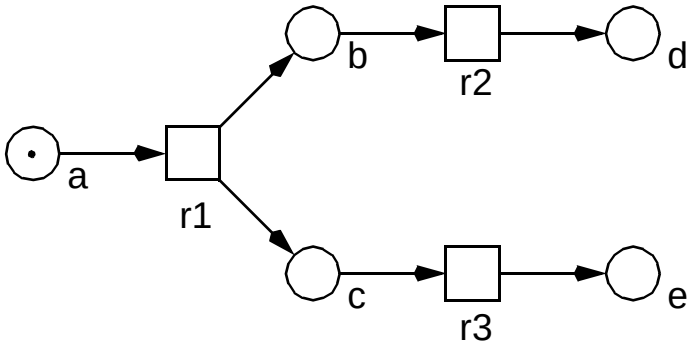


- **atomic actions -> Petri net transitions -> chemical reactions**

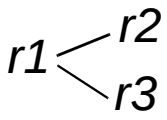
$$2\ NAD^+ + 2\ H_2O \rightarrow 2\ NADH + 2\ H^+ + O_2$$



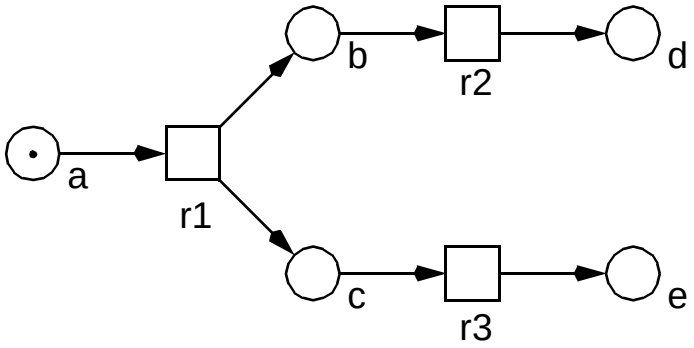
PARTIAL ORDER VERSUS INTERLEAVING SEMANTICS



- **order between r1 - r2 and r1 - r3**
  - -> *causality* $x < y$ [x-y]
  - -> *dependency*
- **no order between r2 , r3**
  - -> *concurrency* $x \,||\, y$
  - -> *independency*
- **partial order run**



PARTIAL ORDER VERSUS INTERLEAVING SEMANTICS



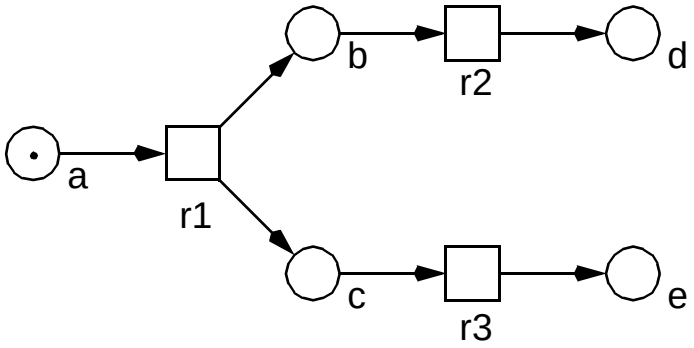
- **order between r1 - r2 and r1 - r3**
  - -> *causality* $x < y$ [x-y]
  - -> *dependency*
- **no order between r2 , r3**
  - -> *concurrency* $x \,||\, y$
  - -> *independency*
- **partial order run**

$$r1 \begin{cases} r2 \\ r3 \end{cases}$$

-> **PARTIAL ORDER SEMANTICS**
"true concurrency semantics"

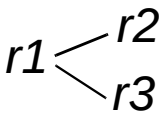
all partially ordered runs

PARTIAL ORDER VERSUS INTERLEAVING SEMANTICS



- **possible interleaving runs**
  - -> *r1 - r2 - r3*
  - -> *r1 - r3 - r2*
- **totally ordered runs**

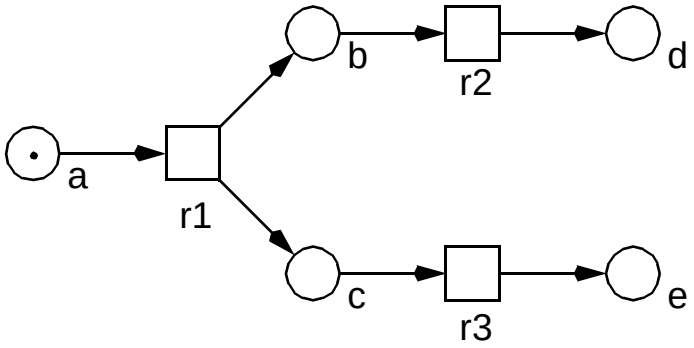
- **order between r1 - r2 and r1 - r3**
  - -> *causality* $x < y$ [x-y]
  - -> *dependency*
- **no order between r2 , r3**
  - -> *concurrency* $x \,||\, y$
  - -> *independency*
- **partial order run**



-> **PARTIAL ORDER SEMANTICS**
"true concurrency semantics"

all partially ordered runs

PARTIAL ORDER VERSUS INTERLEAVING SEMANTICS

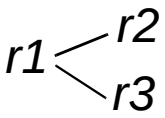


- **possible interleaving runs**
  - -> *r1 - r2 - r3*
  - -> *r1 - r3 - r2*
- **totally ordered runs**

-> **INTERLEAVING SEMANTICS**

all totally ordered runs

- **order between r1 - r2 and r1 - r3**
  - -> *causality* $x < y$ [x-y]
  - -> *dependency*
- **no order between r2 , r3**
  - -> *concurrency* $x \,||\, y$
  - -> *independency*
- **partial order run**



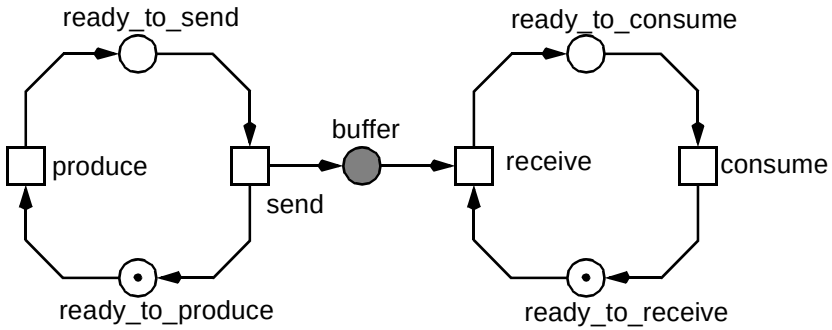
-> **PARTIAL ORDER SEMANTICS**

"true concurrency semantics"

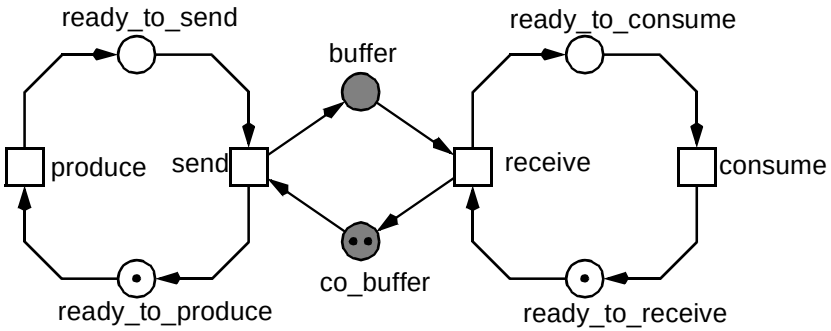
all partially ordered runs

PETRI NET, EXAMPLE 1

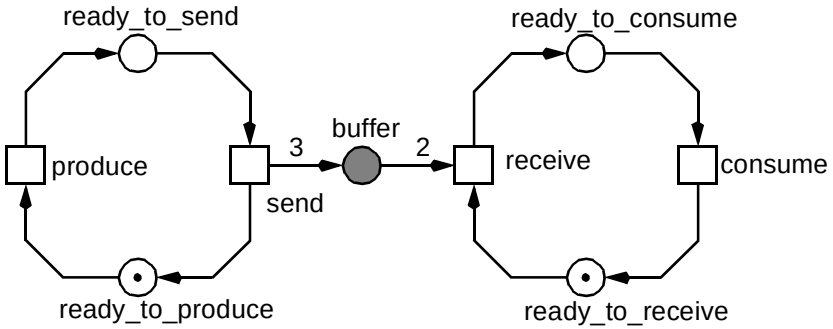
SYSTEM WITHOUT ARC WEIGHTS



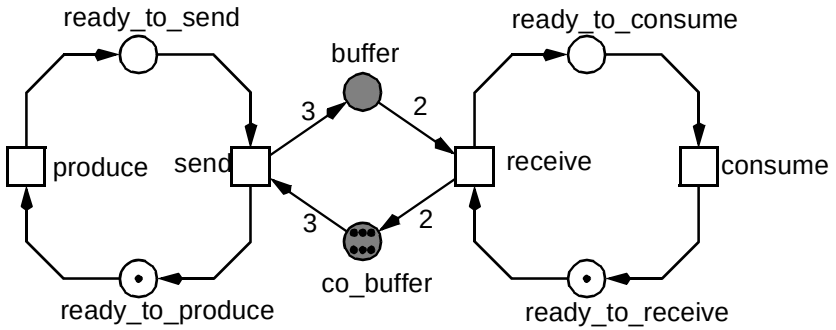
SYSTEM WITHOUT ARC WEIGHTS



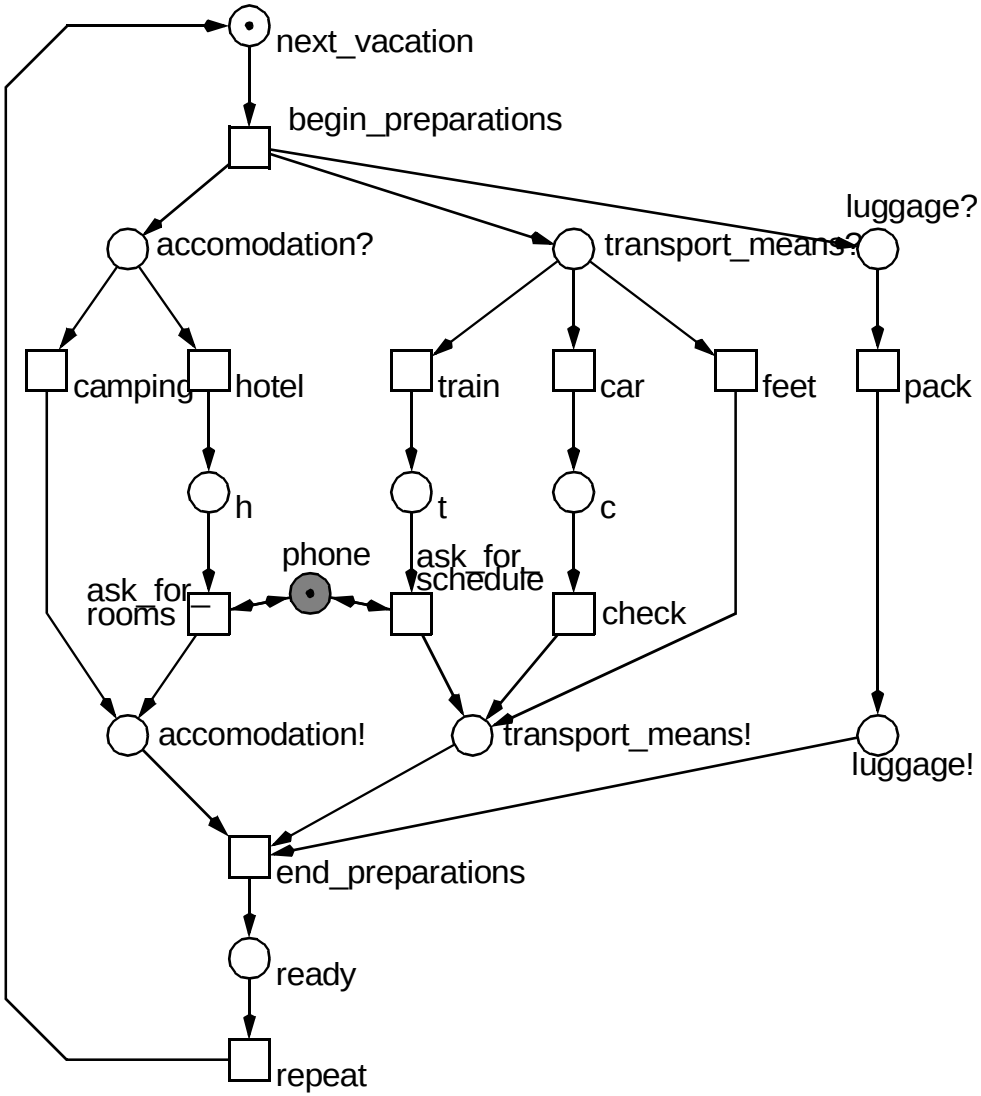
SYSTEM WITH ARC WEIGHTS



SYSTEM WITH ARC WEIGHTS



PETRI NET, EXAMPLE 2



TYPICAL PETRI NET QUESTIONS

- **How many tokens can reside at most in a given place ?**

  -> *(0, 1, k, oo)* -> ***BOUNDEDNESS***

TYPICAL PETRI NET QUESTIONS

- **How many tokens can reside at most in a given place ?**

  -> *(0, 1, k, oo)* -> ***BOUNDEDNESS***

- **How often can a transition fire ?**

  -> *(0-times, n-times, oo-times)* -> ***LIVENESS***

TYPICAL PETRI NET QUESTIONS

- **How many tokens can reside at most in a given place ?**
  - -> *(0, 1, k, oo)* -> ***BOUNDEDNESS***
- **How often can a transition fire ?**
  - -> *(0-times, n-times, oo-times)* -> ***LIVENESS***
- **How often can a system state be reached ?**
  - -> *never* -> ***UNREACHABLE -> SAFETY PROPERTIES***
  - -> *n-times* -> ***REPRODUCIBLE***
  - -> *always reachable again* -> ***REVERSIBLE (HOME STATE)***
  - -> *reversible initial state* -> ***REVERSIBILITY***

TYPICAL PETRI NET QUESTIONS

- **How many tokens can reside at most in a given place ?**
  - -> *(0, 1, k, oo)* -> ***BOUNDEDNESS***
- **How often can a transition fire ?**
  - -> *(0-times, n-times, oo-times)* -> ***LIVENESS***
- **How often can a system state be reached ?**
  - -> *never* -> ***UNREACHABLE -> SAFETY PROPERTIES***
  - -> *n-times* -> ***REPRODUCIBLE***
  - -> *always reachable again* -> ***REVERSIBLE (HOME STATE)***
  - -> *reversible initial state* -> ***REVERSIBILITY***
- **Are there behaviourally invariant subnet structures ?**
  - -> *token conservation* -> *P* - ***INVARIANTS***
  - -> *token distribution reproduction* -> *T* - ***INVARIANTS***
- **. . . and many more -> temporal logics (CTL, LTL)**

TYPICAL PETRI NET QUESTIONS

- **How many tokens can reside at most in a given place ?**
  - -> *(0, 1, k, oo)* -> ***BOUNDEDNESS***
- **How often can a transition fire ?**
  - -> *(0-times, n-times, oo-times)* -> ***LIVENESS***
- **How often can a system state be reached ?**
  - -> *never* -> ***UNREACHABLE -> SAFETY PROPERTIES***
  - -> *n-times* -> ***REPRODUCIBLE***
  - -> *always reachable again* -> **REVERSIBLE (HOME STATE)**
  - -> *reversible initial state* -> ***REVERSIBILITY***
- **Are there behaviourally invariant subnet structures ?**
  - -> *token conservation* -> *P - **INVARIANTS***
  - -> *token distribution reproduction* -> *T - **INVARIANTS***
- **. . . and many more -> temporal logics (CTL, LTL)**

GENERAL BEHAVIOURAL PROPERTIES

SPECIAL BEHAVIOURAL PROPERTIES

Petri nets,
typical analysis techniques

Petri nets, typical analysis techniques

MODEL ANIMATION (?)

Dynamic Analyses

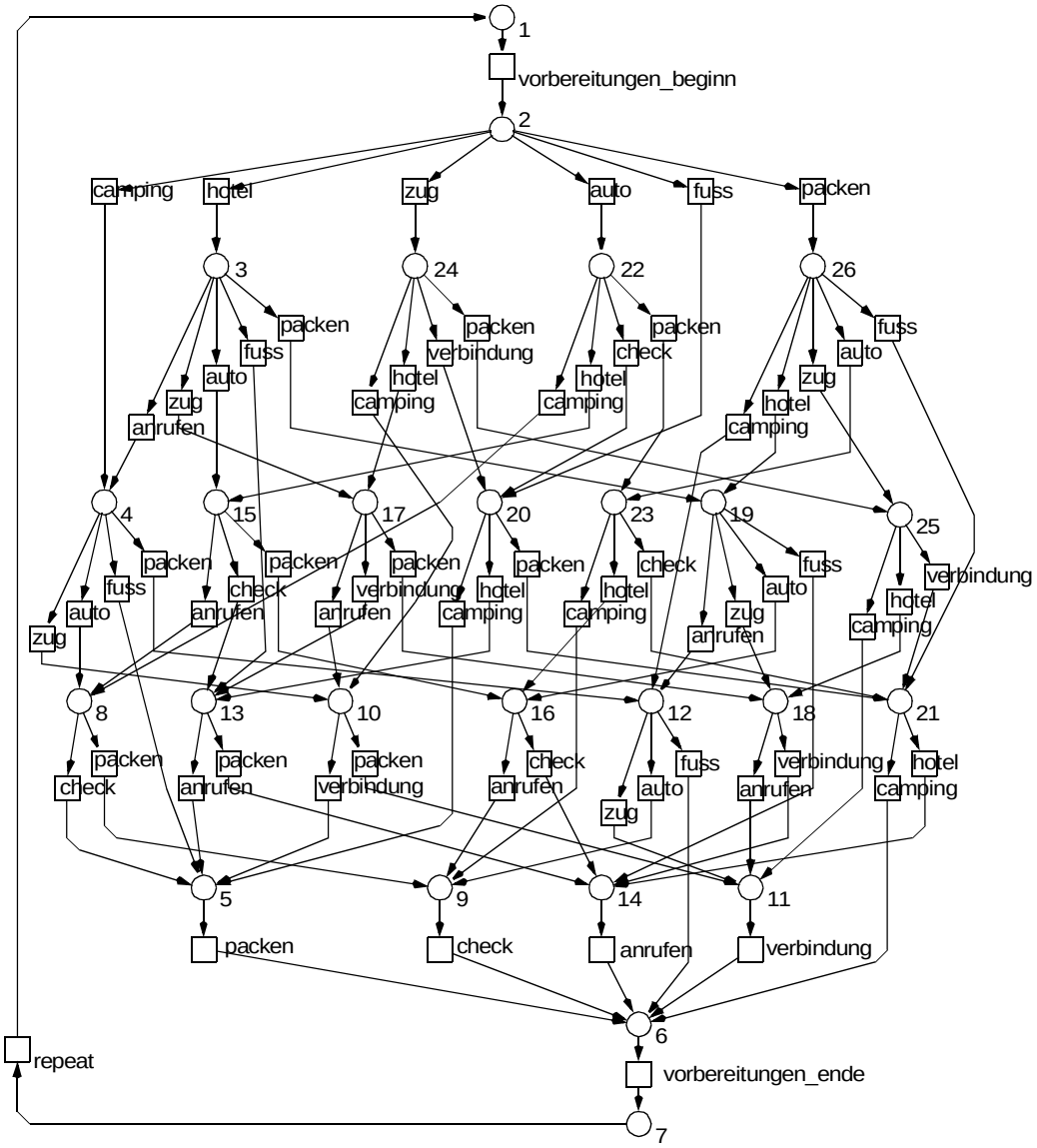
- **reachability / occurrence graph,**
  - -> *(labelled) state transition graph*
  - -> *Kripke structure, CTMC, . . .*
- **nodes**
  - -> *system states / markings*
- **arcs**
  - -> *the (single) firing transition*
  - -> *single step firing*
- **interleaving semantics**
  - -> *(sequential) finite automaton*
  - -> *concurrency == enumerating all interleaving sequences*
- **reachability graph construction - simple algorithm**

- **boundedness**
  -> *finite graph*
- **reversibility**
  -> *one Strongly Connected Component (SCC)*
- **liveness**
  -> *every transition contained in all terminal SCC*
- **dead states**
  -> *terminal nodes*

- **boundedness**
  -> *finite graph*
- **reversibility**
  -> *one Strongly Connected Component (SCC)*
- **liveness**
  -> *every transition contained in all terminal SCC*
- **dead states**
  -> *terminal nodes*

-> reachability graphs tend to be huge <-

REACHABILITY GRAPH, Example 2



- infinite for unbounded nets

REACHABILITY GRAPH, STATE SPACE COMPLEXITY

- infinite for unbounded nets
- worst-case for finite state spaces [Priese, Wimmel 2003]

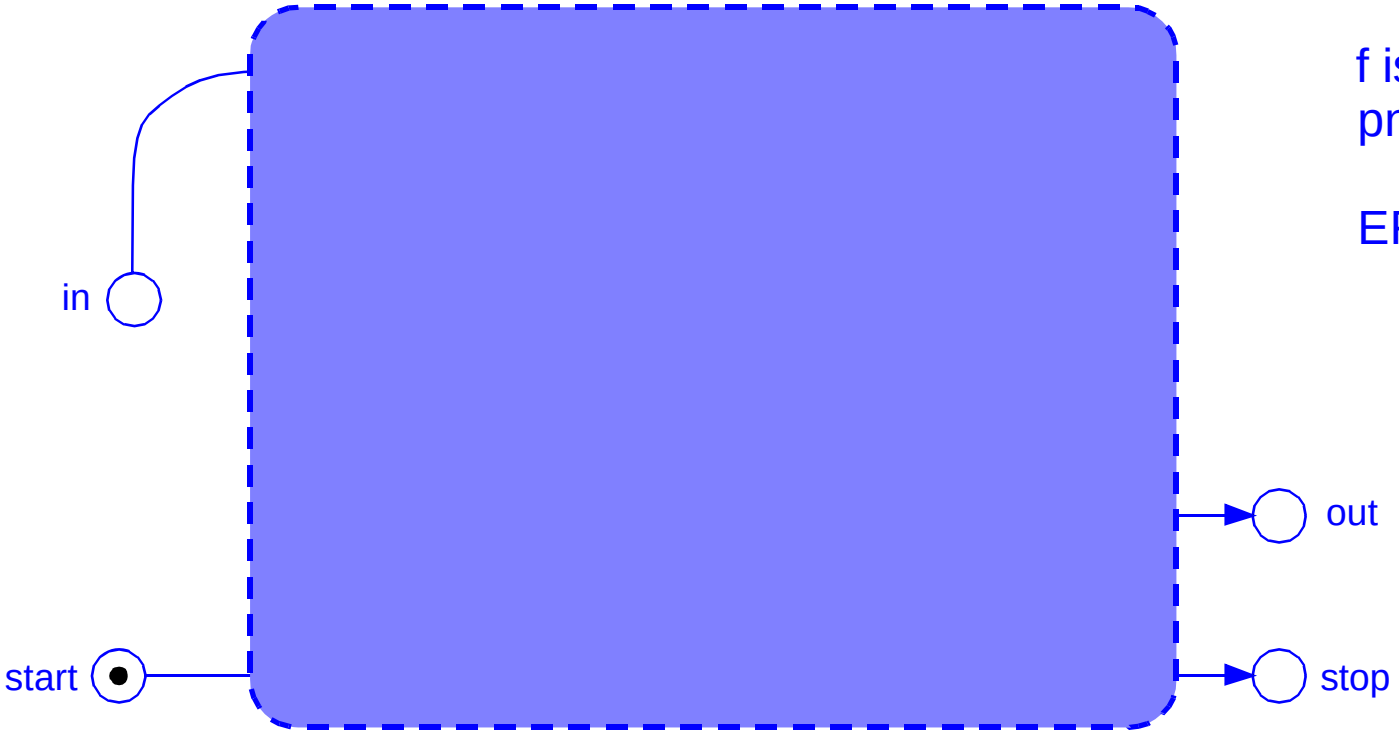
REACHABILITY GRAPH, STATE SPACE COMPLEXITY

- **infinite for unbounded nets**
- **worst-case for finite state spaces [Priese, Wimmel 2003]**

  ... cannot be bounded by a primitive recursive function ...

- **infinite for unbounded nets**
- **worst-case for finite state spaces** **[Priese, Wimmel 2003]**

  ... cannot be bounded by a primitive recursive function ...
- **proof** -> Petri net computer for a function **f: N_0^m -> N_0**

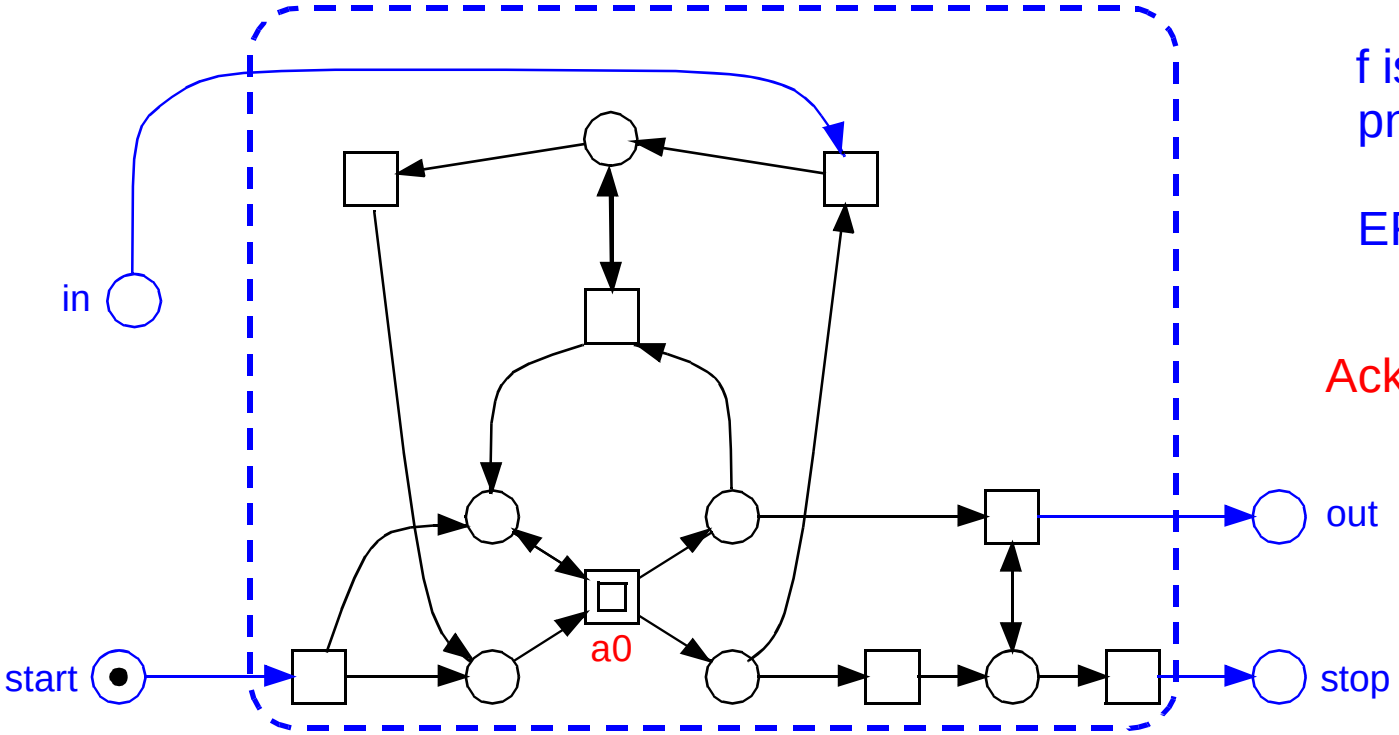


f is weakly pn-computable:

EF(out = f(in) &
stop = 1)

- **infinite for unbounded nets**
- **worst-case for finite state spaces** **[Priese, Wimmel 2003]**

  ... cannot be bounded by a primitive recursive function ...
- **proof** -> Petri net computer for a function **f: N_0^m -> N_0**



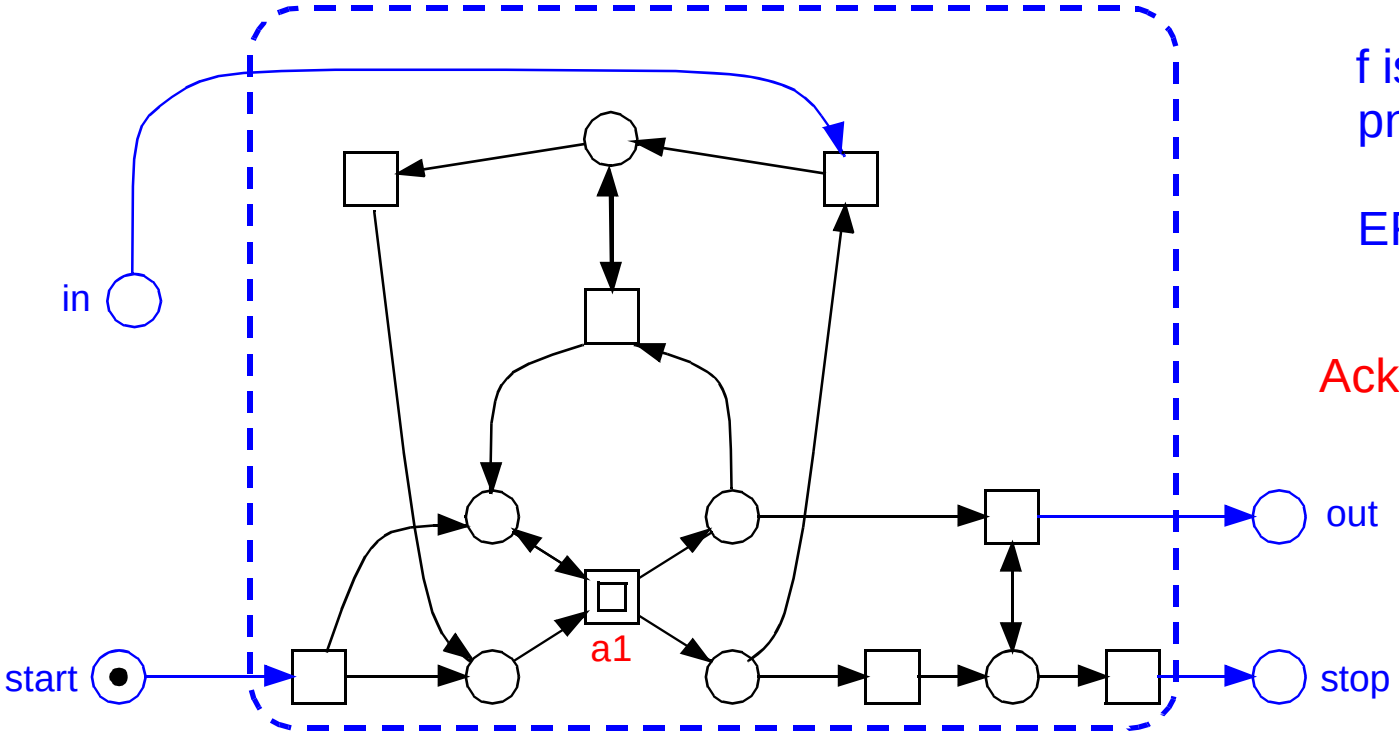
f is weakly pn-computable:

EF(out = f(in) &
stop = 1)

Ackermann function a1

- **infinite for unbounded nets**
- **worst-case for finite state spaces** **[Priese, Wimmel 2003]**

  ... cannot be bounded by a primitive recursive function ...
- **proof** -> Petri net computer for a function **f: N_0^m -> N_0**

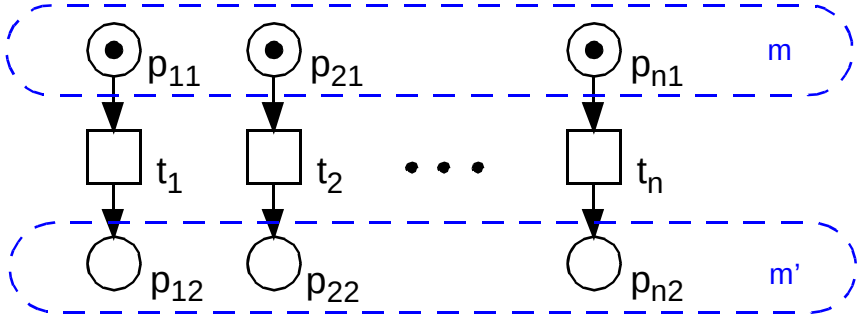


f is weakly pn-computable:

EF(out = f(in) &
stop = 1)

Ackermann function a2

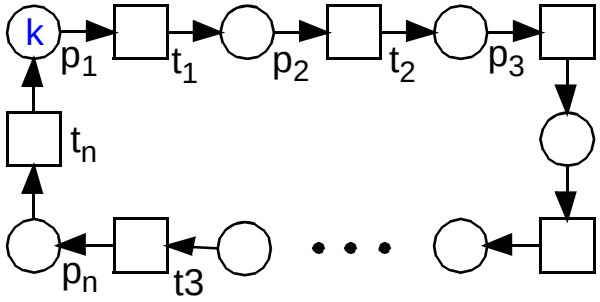
STATE SPACE COMPLEXITY, CAUSES



n! interleaving sequences

m -> m'

2^n - 2 intermediate states



$$\frac{(n + k - 1)!}{(n - 1)!\, k!}$$ states

(combination with repetition)

ANALYSIS TECHNIQUES

- **static analyses** -> **no state space construction**

- **dynamic analyses** -> **total / partial state space construction**

ANALYSIS TECHNIQUES

- **static analyses** **-> no state space construction**
  - -> *structural properties (graph theory)*
  - -> *P / T - invariants (linear algebra)*
  - -> *state equation*
- **dynamic analyses** **-> total / partial state space construction**

ANALYSIS TECHNIQUES

- **static analyses** **-> no state space construction**
  - -> *structural properties (graph theory)*
  - -> *P / T - invariants (linear algebra)*
  - -> *state equation*

- **dynamic analyses** **-> total / partial state space construction**

  -> *analysis of general behavioural system properties,*
  i.e. boundedness, liveness, reversibility

  -> *model checking of special behavioural system properties,*
  e.g. reachability of a given (sub-) system state (with constraints),
  reproducability of a given (sub-) system state (with constraints)

  => expressed in temporal logics (CTL / LTL),
  as very flexible & powerful query language

Static Analyses

STRUCTURAL ANALYSIS, SOME EXAMPLES

- **boundary nodes**
  - -> *input transitions -> not BND*
  - -> *input places -> not LIVE*
  - -> *LIVE & BND -> no boundary nodes*
- **conservative -> BND**
- . . .
- **Deadlock-Trap Property (DTP)**
  - -> *no structural deadlock -> live*
  - -> *ORD & DTP -> no dead states*
  - -> *ORD & ES & DTP -> LIVE*
  - -> *ORD & EFC & DTP <-> LIVE*

INCIDENCE MATRIX C

- **a representation of the net structure**

$C =$

P \ T	t1 . . .	tj	. . . tm
p1			
pi		cij	
: pn		Δtj	

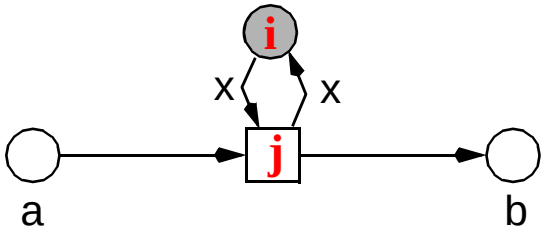
$cij = (pi, tj) = F(tj,pi) - F(pi, tj) = \Delta\, tj(pi)$

$\Delta tj = \Delta\, tj(*)$

- **matrix entry cij:**
  token change in place pi by firing of transition tj

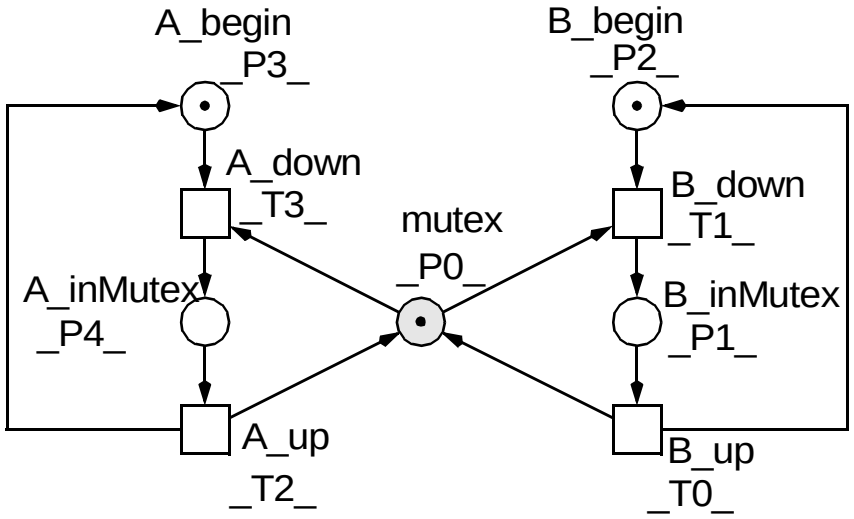
- **matrix column Δtj:**
  vector describing the change of the whole marking by firing of tj

- **side-conditions are neglected**

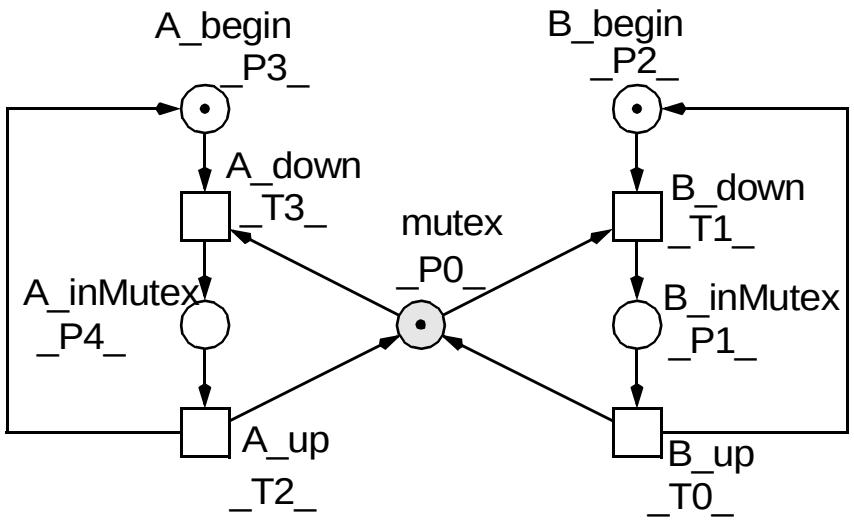


cij = 0

INCIDENCE MATRIX, EXAMPLE

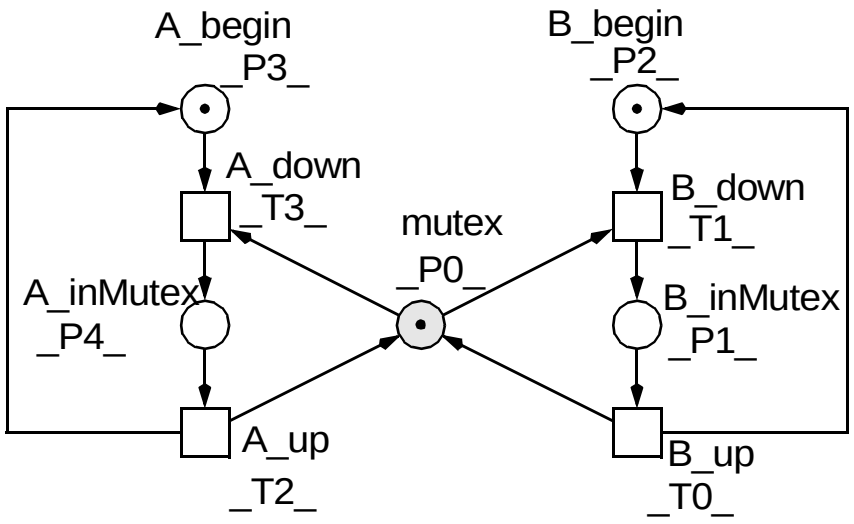


INCIDENCE MATRIX, EXAMPLE



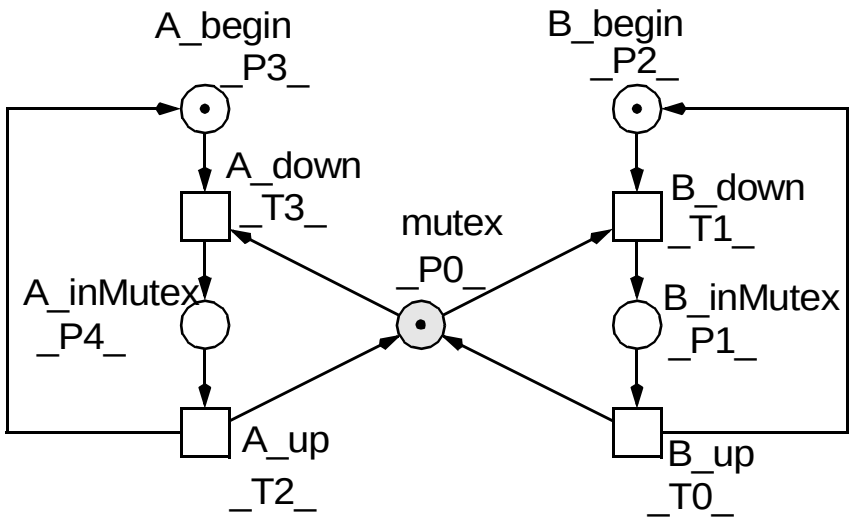
P \ T	B_up	B_down	A_up	A_down
mutex	+1	-1	+1	-1
B_inMutex	-1	+1	0	0
B_begin	+1	-1	0	0
A_begin	0	0	+1	-1
A_inMutex	0	0	-1	+1

INCIDENCE MATRIX, EXAMPLE



P \ T	B_up	B_down	A_up	A_down	m0
mutex	+1	-1	+1	-1	1
B_inMutex	-1	+1	0	0	0
B_begin	+1	-1	0	0	1
A_begin	0	0	+1	-1	1
A_inMutex	0	0	-1	+1	0

INCIDENCE MATRIX, EXAMPLE



P \ T	B_up	B_down	A_up	A_down
mutex	+1	-1	+1	-1
B_inMutex	-1	+1	0	0
B_begin	+1	-1	0	0
A_begin	0	0	+1	-1
A_inMutex	0	0	-1	+1

A_down

m0 →

1

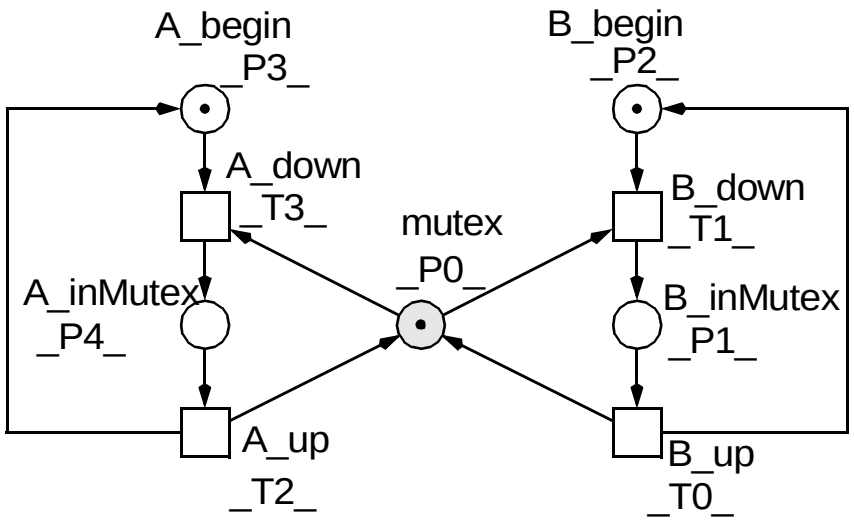
0

1

1

0

INCIDENCE MATRIX, EXAMPLE



P \ T	B_up	B_down	A_up	A_down
mutex	+1	-1	+1	-1
B_inMutex	-1	+1	0	0
B_begin	+1	-1	0	0
A_begin	0	0	+1	-1
A_inMutex	0	0	-1	+1

A_down

m0 →

1

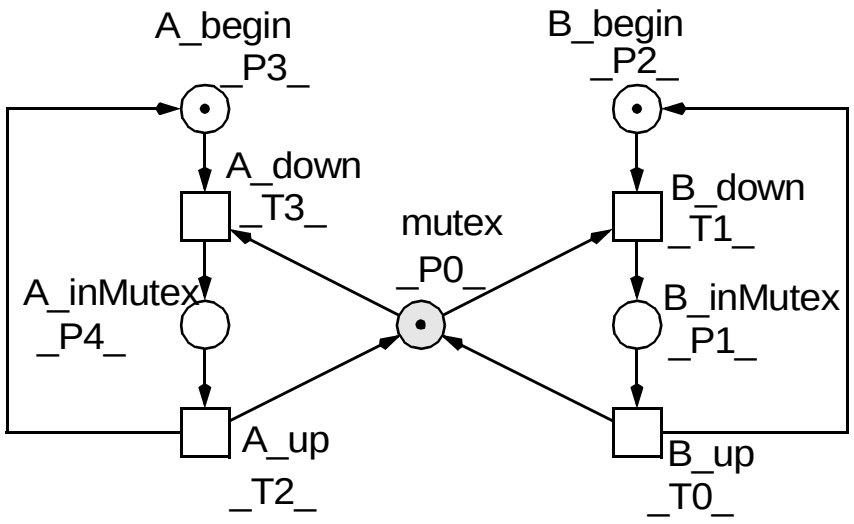
0

1

1

0

INCIDENCE MATRIX, EXAMPLE

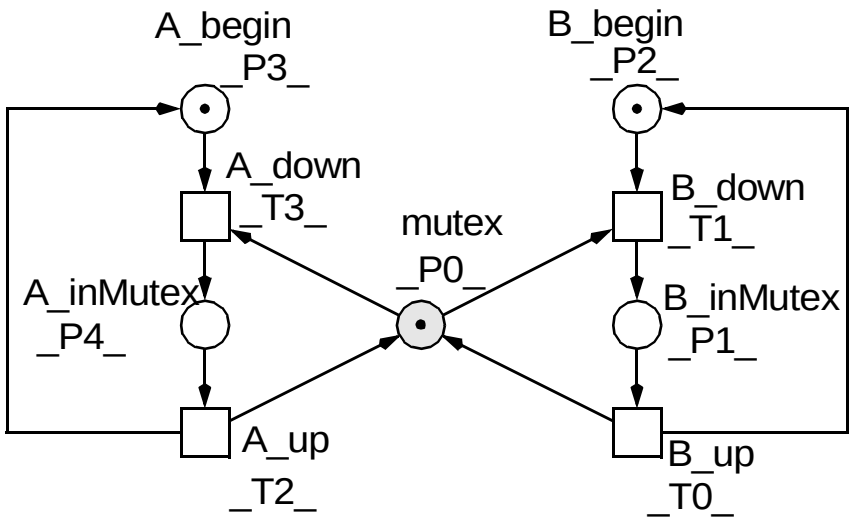


P \ T	B_up	B_down	A_up	A_down
mutex	+1	-1	+1	-1
B_inMutex	-1	+1	0	0
B_begin	+1	-1	0	0
A_begin	0	0	+1	-1
A_inMutex	0	0	-1	+1

A_down

m0	→	m1
1		0
0		0
1		1
1		0
0		1

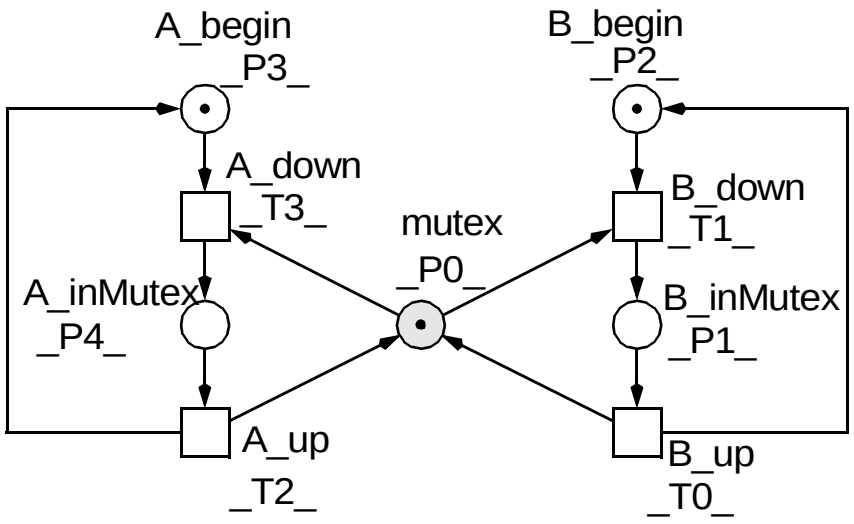
INCIDENCE MATRIX, EXAMPLE



P \ T	B_up	B_down	A_up	A_down
mutex	+1	-1	+1	-1
B_inMutex	-1	+1	0	0
B_begin	+1	-1	0	0
A_begin	0	0	+1	-1
A_inMutex	0	0	-1	+1

m0	A_down →	m1	A_up →
1		0	
0		0	
1		1	
1		0	
0		1	

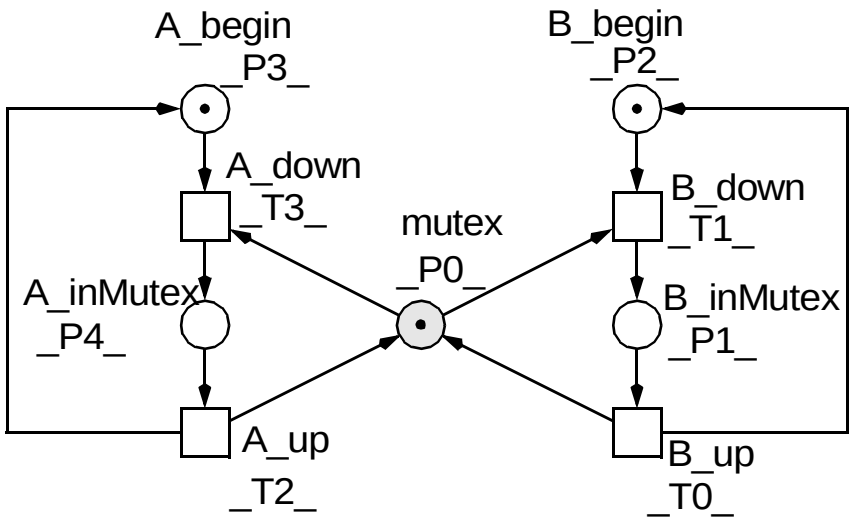
INCIDENCE MATRIX, EXAMPLE



P \ T	B_up	B_down	A_up	A_down
mutex	+1	-1	+1	-1
B_inMutex	-1	+1	0	0
B_begin	+1	-1	0	0
A_begin	0	0	+1	-1
A_inMutex	0	0	-1	+1

A_down		A_up	
m0	→	m1	→ m0
1		0	1
0		0	0
1		1	1
1		0	1
0		1	0

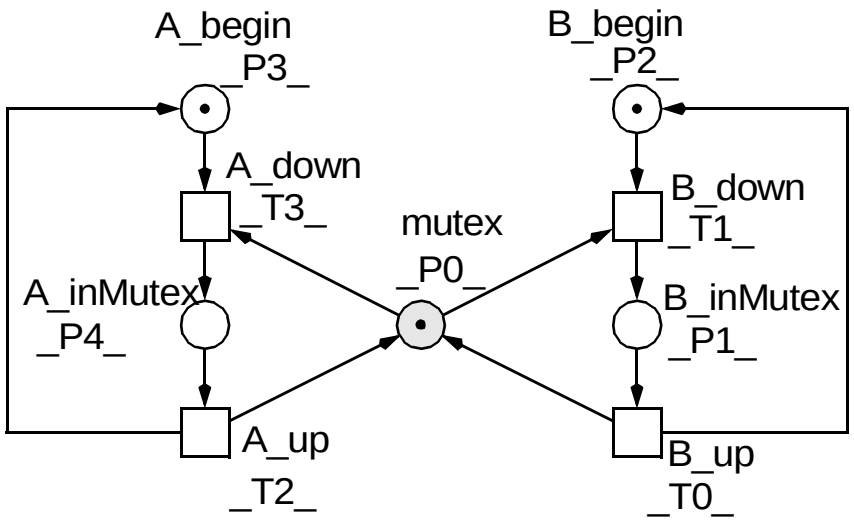
INCIDENCE MATRIX, EXAMPLE



P \ T	B_up	B_down	A_up	A_down
mutex	+1	-1	+1	-1
B_inMutex	-1	+1	0	0
B_begin	+1	-1	0	0
A_begin	0	0	+1	-1
A_inMutex	0	0	-1	+1

0
0
0
1

INCIDENCE MATRIX, EXAMPLE

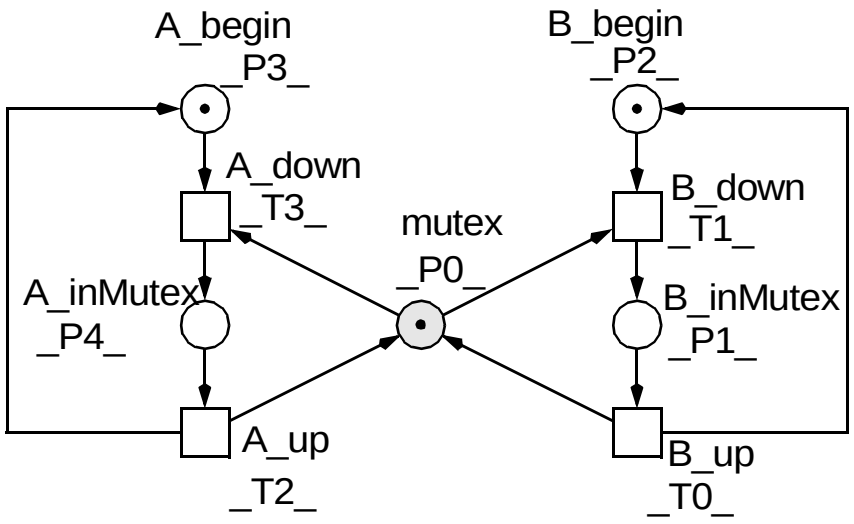


P \ T	B_up	B_down	A_up	A_down
mutex	+1	-1	+1	-1
B_inMutex	-1	+1	0	0
B_begin	+1	-1	0	0
A_begin	0	0	+1	-1
A_inMutex	0	0	-1	+1

0	
0	
0	
1	

-1	
0	
0	
-1	
1	

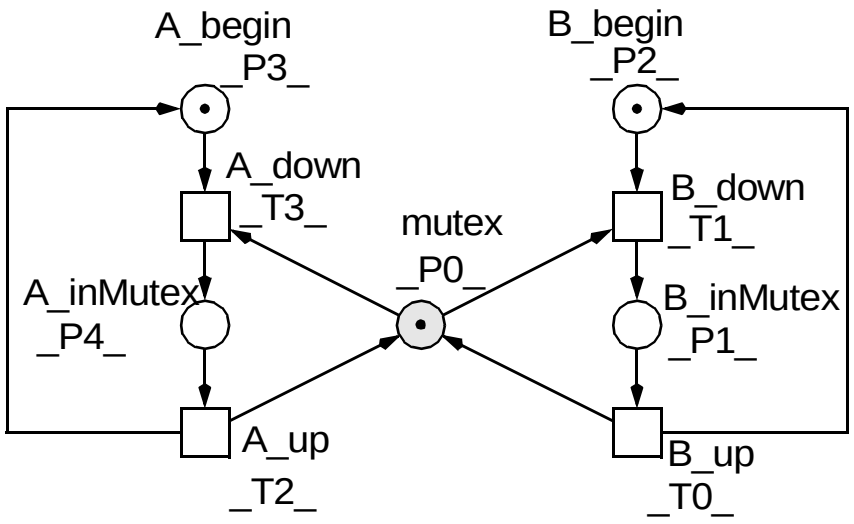
INCIDENCE MATRIX, EXAMPLE



P \ T	B_up	B_down	A_up	A_down
mutex	+1	-1	+1	-1
B_inMutex	-1	+1	0	0
B_begin	+1	-1	0	0
A_begin	0	0	+1	-1
A_inMutex	0	0	-1	+1

			m0 →	m1
0				
0				
0				
1				
-1		1		0
0		0		0
0	+	1	=	1
-1		1		0
1		0		1

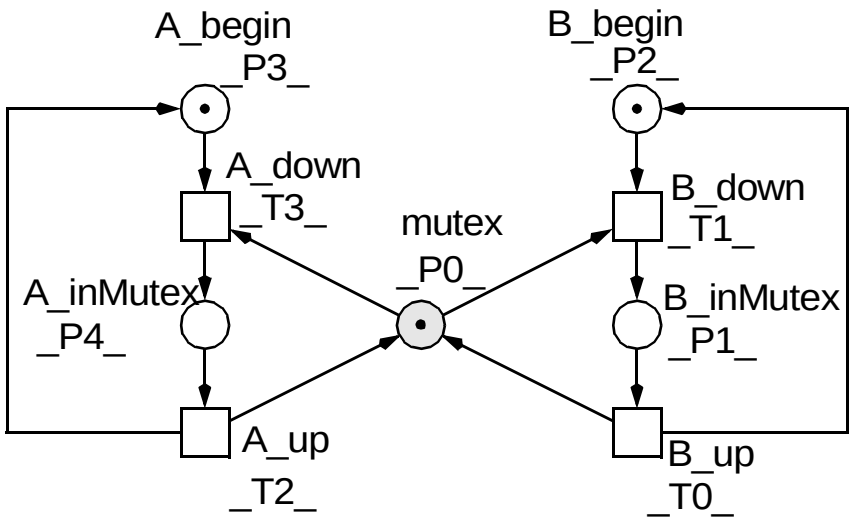
INCIDENCE MATRIX, EXAMPLE



P \ T	B_up	B_down	A_up	A_down
mutex	+1	-1	+1	-1
B_inMutex	-1	+1	0	0
B_begin	+1	-1	0	0
A_begin	0	0	+1	-1
A_inMutex	0	0	-1	+1

			m1 →	m0
0				
0				
1				
0				
1		0		1
0		0		0
0	+	1	=	1
1		0		1
-1		1		0

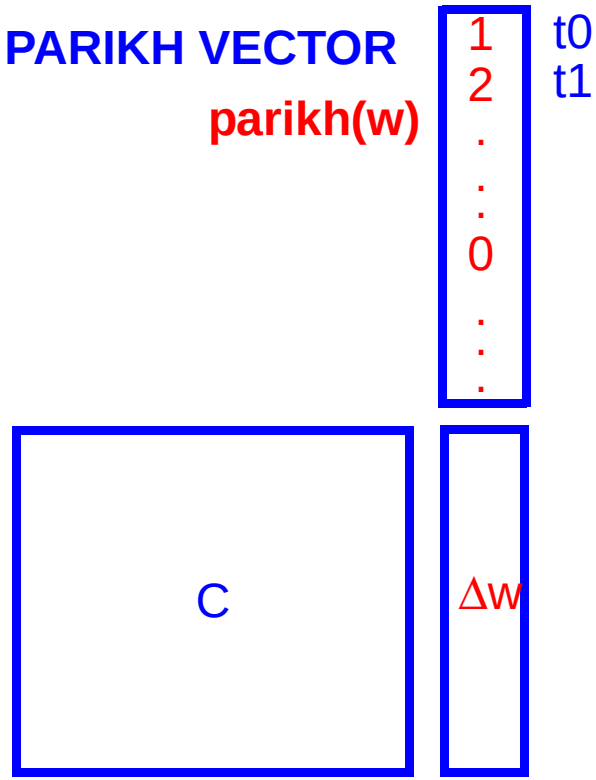
INCIDENCE MATRIX, EXAMPLE



P \ T	B_up	B_down	A_up	A_down
mutex	+1	-1	+1	-1
B_inMutex	-1	+1	0	0
B_begin	+1	-1	0	0
A_begin	0	0	+1	-1
A_inMutex	0	0	-1	+1

				m0 → m0
0		1		1
0		0		0
0	+	1	=	1
0		1		1
0		0		0

STATE EQUATION



Let the word w = t1-t0-t1-. . .
be a sequence of firing transitions.

The change of the marking

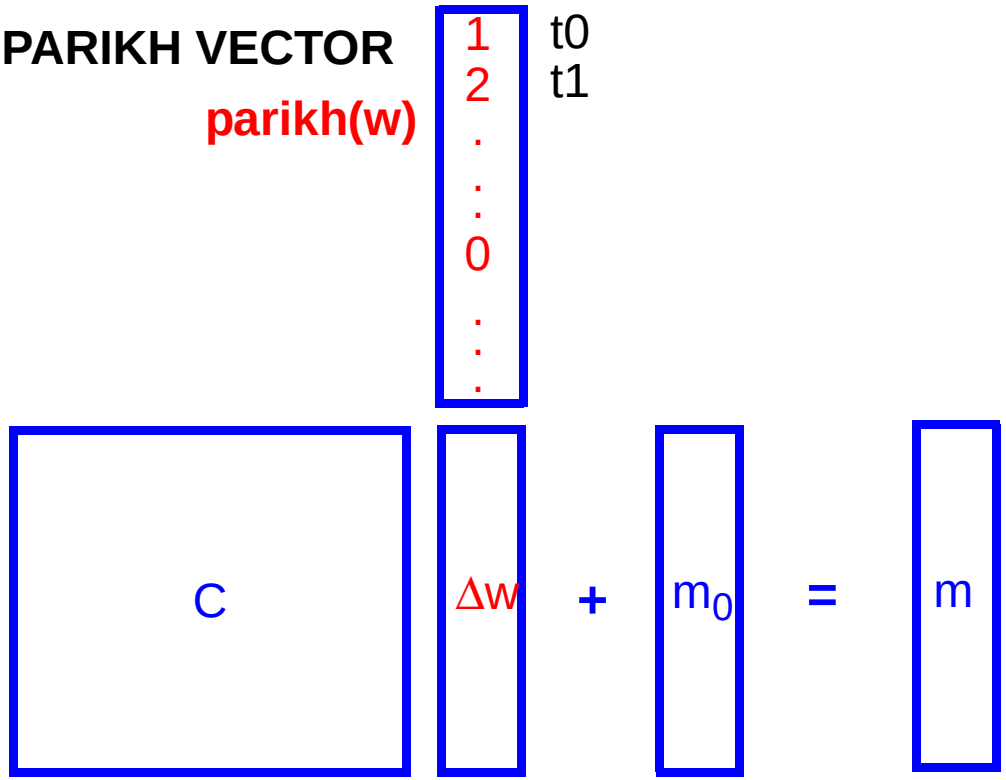
Δw

by firing that sequence
can be computed
by multiplying
the incidence matrix C
with the
Parikh vector parikh(w)
of that transition sequence.

STATE EQUATION

PARIKH VECTOR

parikh(w)



STATE EQUATION

- **linear programming problem**

$$m = m_0 + C\,x,$$
$$x >= 0$$

x **- T-vector**

- **There exists an integer solution for every reachable marking m (the Parikh vector of the transition sequence going to m).**

- **the integer solvability is a necessary condition for the reachability of a marking**

- **NON-REACHABILITY CHECK**

  -> *if there is no integer solution, then the marking is not reachable*

- **each reachable marking is a linear combination of**

  -> *initial marking*

  -> *columns of incidence matrix*

P-INVARIANTS, BASICS

- **Lautenbach, 1973**
- **P-invariants** -> *multisets of places*
  - -> *integer solutions y of* $yC = 0, y \neq 0, y \geq 0$
- **minimal P-invariants**
  - -> *there is no P-invariant with a smaller support* -> *sets of places*
  - -> *gcD of all entries is 1*
- **any P-invariant is a non-negative linear combination of minimal ones**
  - -> *multiplication with a positive integer*
  - -> *addition*
  - -> *Division by gcD*

$$ky = \sum_i a_i y_i$$

- **Covered by P-Invariants (CPI)**
  - -> *each place belongs to a P-invariant*
  - -> *CPI => BND (sufficient condition)*

T-INVARIANTS, BASICS

- **Lautenbach, 1973** -> **Schuster, 1993**
- **T-invariants** -> *multisets of transitions*
  - -> *integer solutions x of* $Cx = 0, x \neq 0, x \geq 0$ -> *Parikh vector*
- **minimal T-invariants**
  - -> *there is no T-invariant with a smaller support* -> *sets of transitions*
  - -> *gcD of all entries is 1*
- **any T-invariant is a non-negative linear combination of minimal ones**
  - -> *multiplication with a positive integer*
  - -> *addition*
  - -> *Division by gcD*

$$kx = \sum_i a_i x_i$$

- **Covered by T-Invariants (CTI)**
  - -> *each transition belongs to a T-invariant*
  - -> *BND & LIVE => CTI (necessary condition)*

Petri nets,
an application

-> error-correcting Petri nets