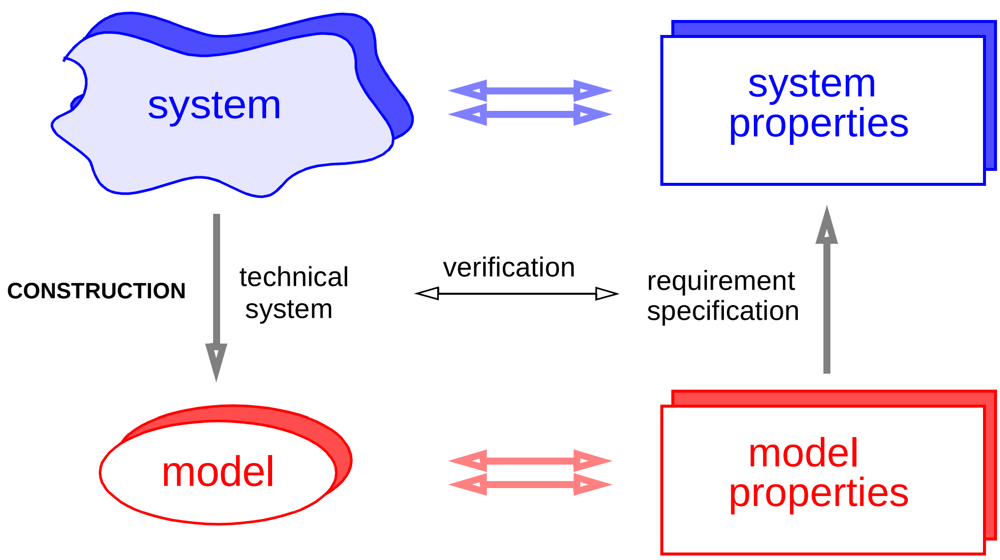
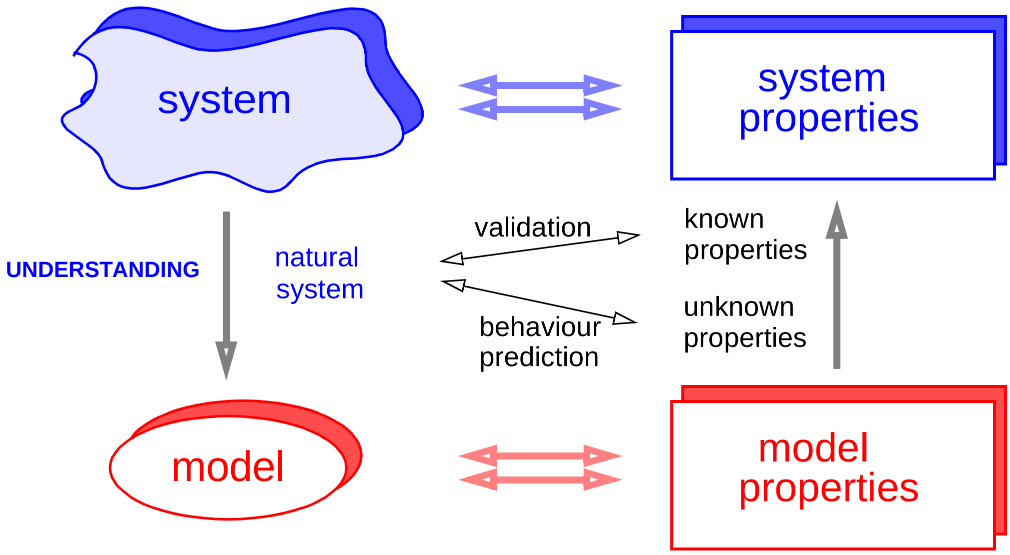
MODEL CHECKING OF CONCURRENT SYSTEMS - PART I -

Monika Heiner

**BTU Cottbus,
Computer Science Institute**

MODEL-BASED SYSTEM ANALYSIS





- **a language to model the system**
  - -> *formal semantics*
  - -> *many options, e.g.*
    Petri nets
- **a language to specify model properties**
  - -> *temporal Logics,*
  - -> *several options, e.g.*
    Computational Tree Logic (CTL)
- **an analysis approach to check a model against its properties**
  - -> *model checking,*
  - -> *various approaches (algorithms + data structures), e.g.*
    using reachability graph (RG)
    = labelled state transition system (STS) = Kripke structure
    ≈ Continuous Time Markov Chain (CTMC)

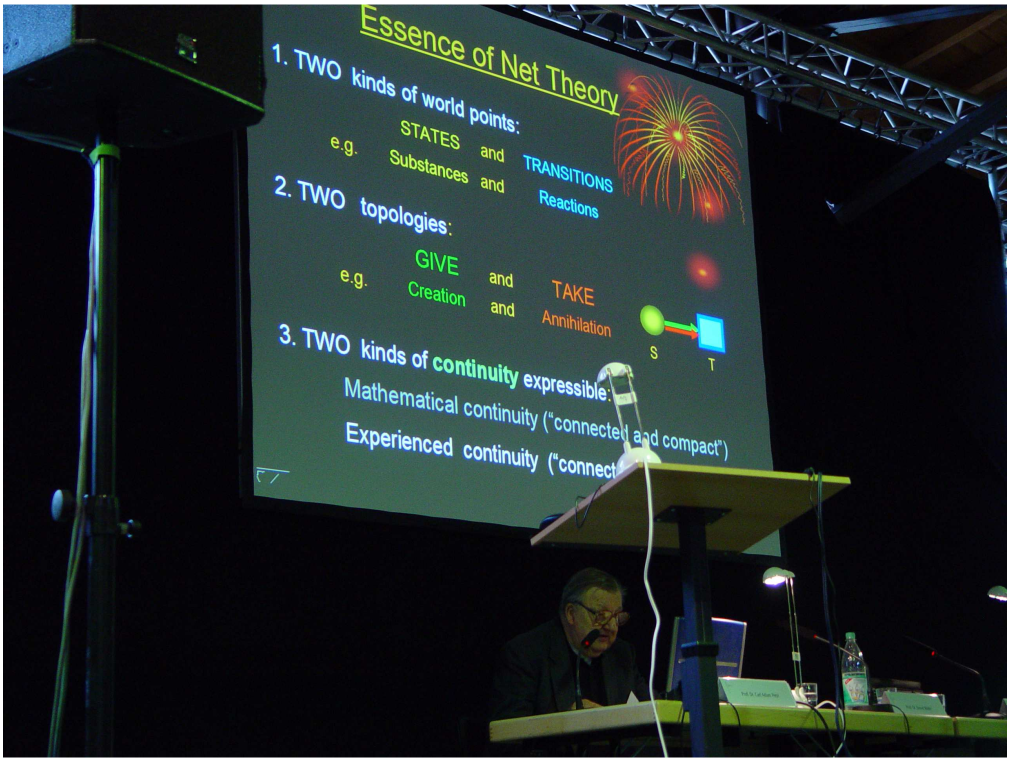
The modelling language -

Petri nets,
a crash course

A Bit of History

C. A. Petri, November 2006

A Bit of History

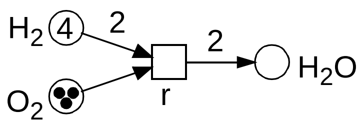


Petri Nets, Basics - the Structure

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

$2\ H_2 + O_2 \rightarrow 2\ H_2O$

input compounds



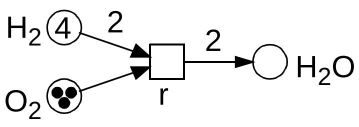
output compounds

- **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**
- **$PN = (P, T, F, m_0)$, $F: (P \times T) \cup (T \times P) \rightarrow N_0$, $m_0: P \rightarrow N_0$**

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

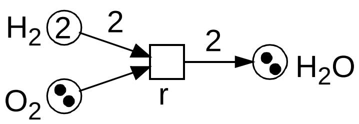
$$2\ H_2 + O_2 \rightarrow 2\ H_2O$$

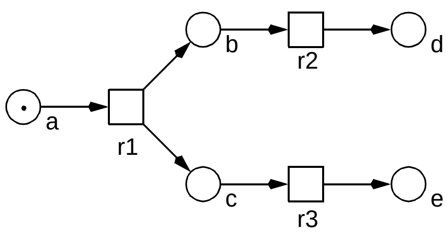
input compounds



output compounds

FIRING → **TOKEN GAME** → **DYNAMIC BEHAVIOUR (substance/signal flow)**



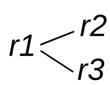


- **possible interleaving runs**
  - -> *r1 - r2 - r3*
  - -> *r1 - r3 - r2*
- **totally ordered runs**

-> 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**



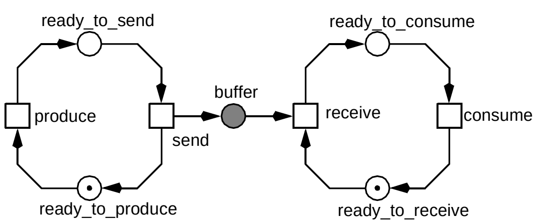
-> PARTIAL ORDER SEMANTICS
"true concurrency semantics"

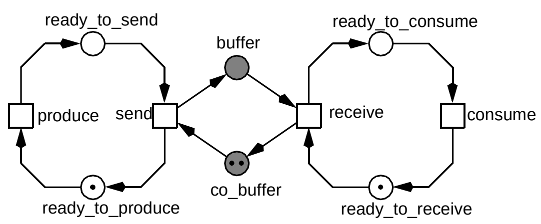
all partially ordered runs

Some examples

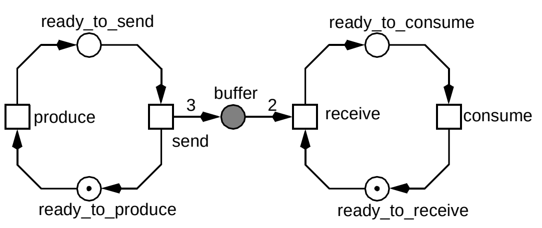
Example 1 - Producer/Consumer System in four Versions

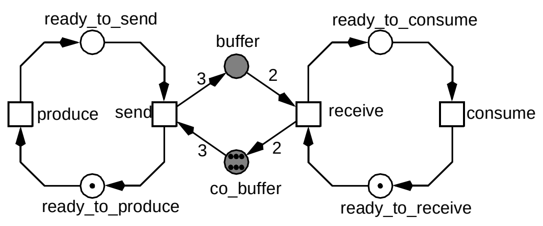
- **SYSTEMS WITHOUT ARC WEIGHTS**



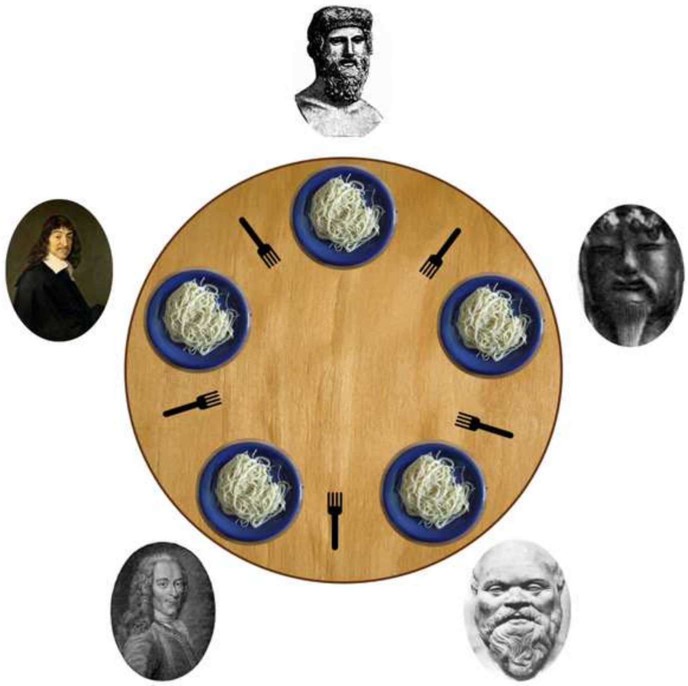


- **SYSTEMS WITH ARC WEIGHTS**



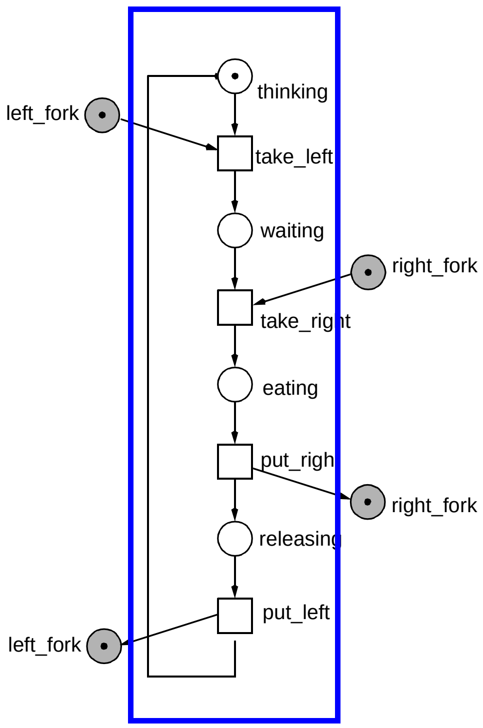


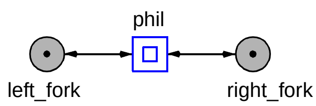
Example 2 - Dining Philosophers



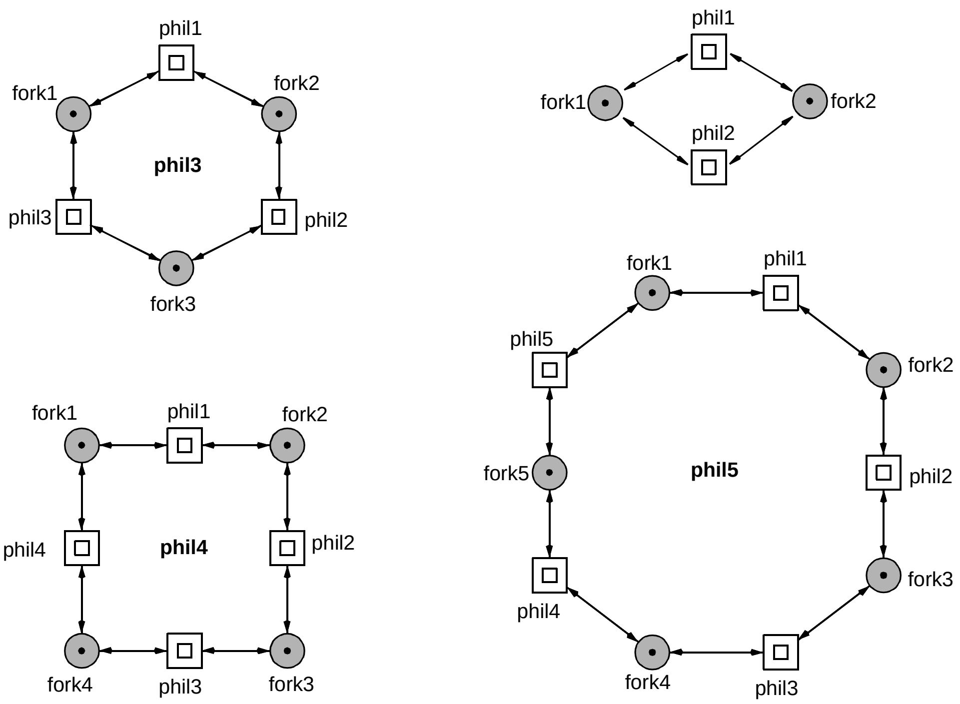
http://en.wikipedia.org/wiki/Dining_philosophers_problem

Example 2 - Dining Philosophers, One Philosopher

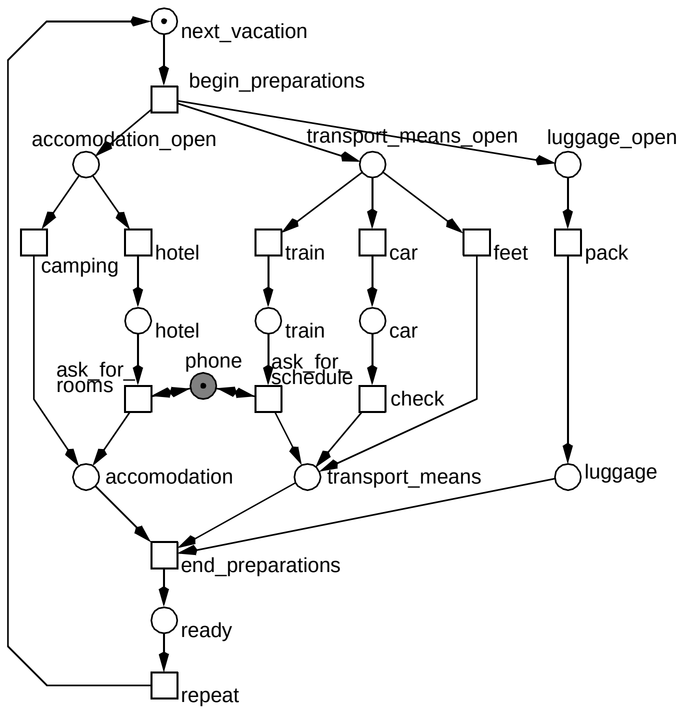




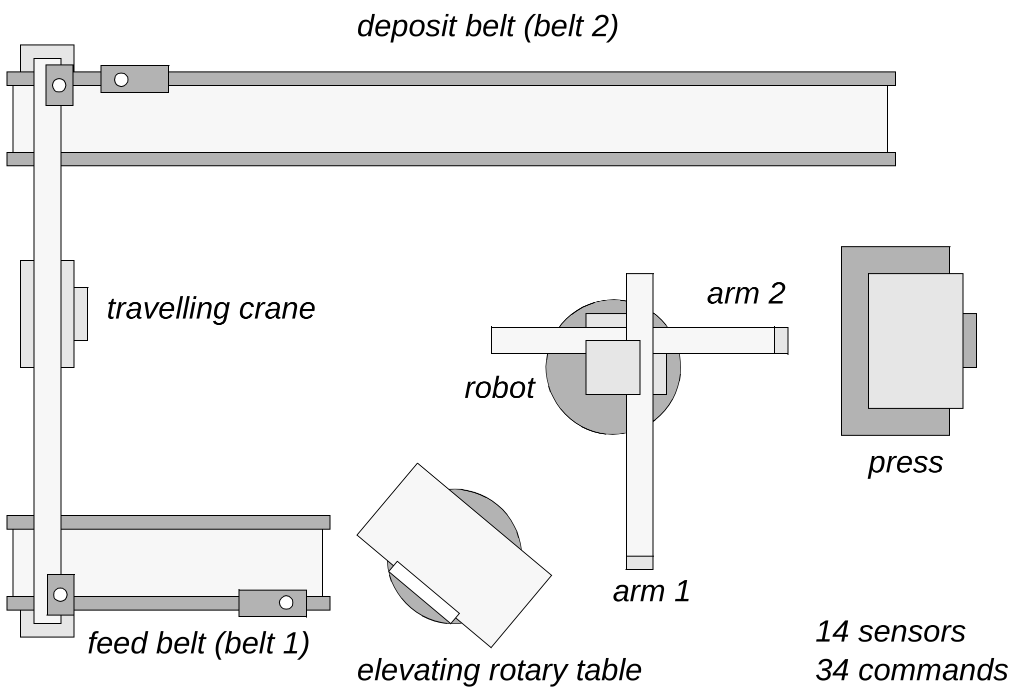
EXAMPLE 2 - SYSTEM OF N PHILOSOPHERS



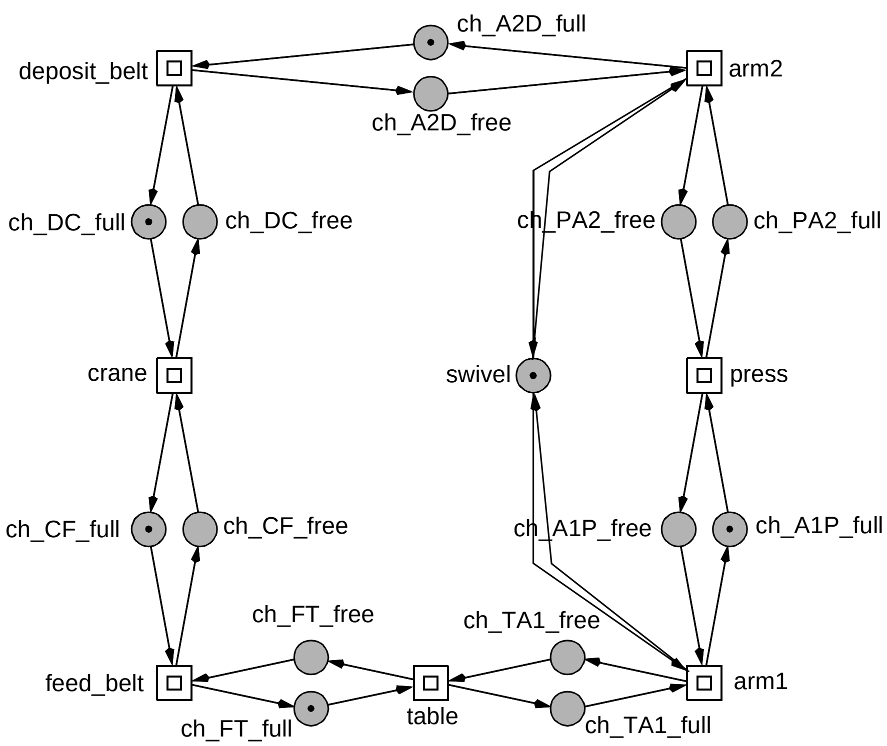
EXAMPLE 3 - TRAVEL PLANING



EXAMPLE 4 - PRODUCTION CELL



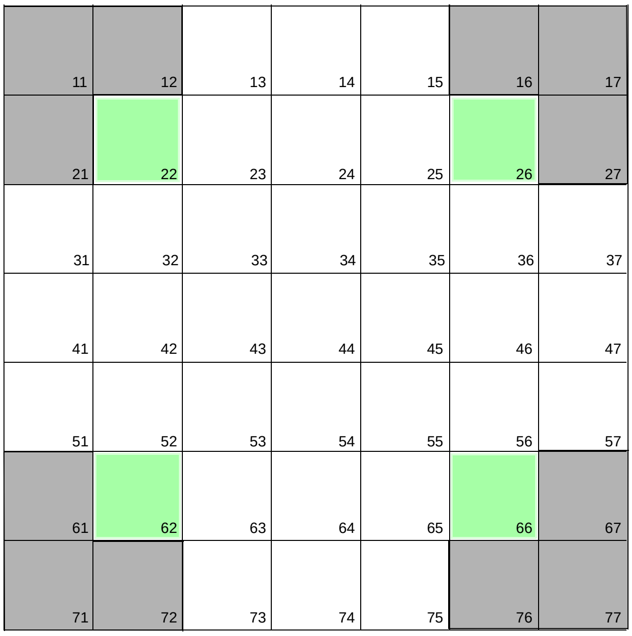
EXAMPLE 4 - CLOSED SYSTEM, COARSE STRUCTURE



231 P,
202 T,
65 PAGES

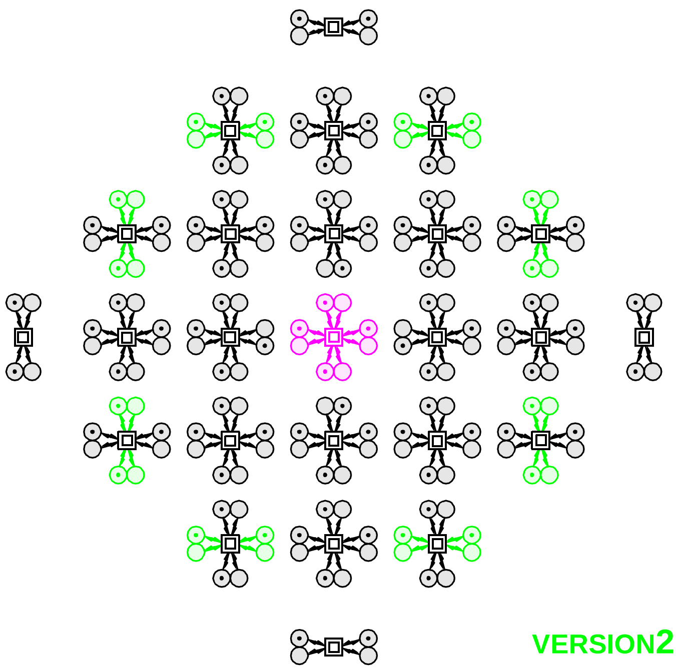
Example 5 - SOLITAIRE GAME

- two versions, green squares Y/N
- all but one squares carry tokens
- remove tokens by jumbing over them
- goal of the game: only one token left
- questions: is there a solution ?
- always ?



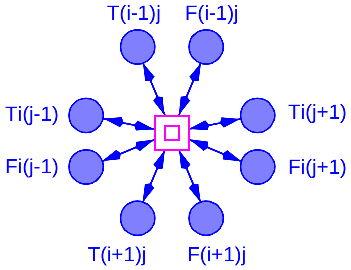
Example 5 - SOLITAIRE GAME

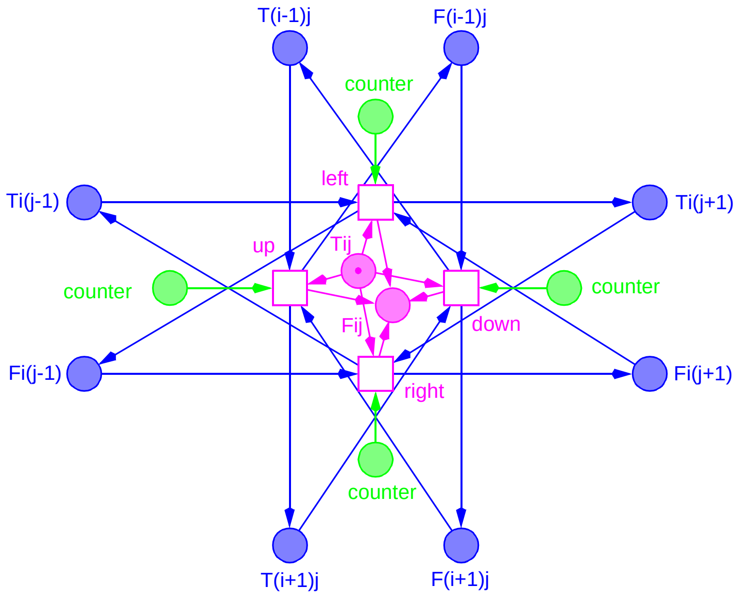
- two-level hierarchical pn
- only one square net component
- two states for each square i: T(i), F(i)
- goal of the game: dead state(s) with Σ T(i) = 1
- reachable ?
- for any initial marking ?



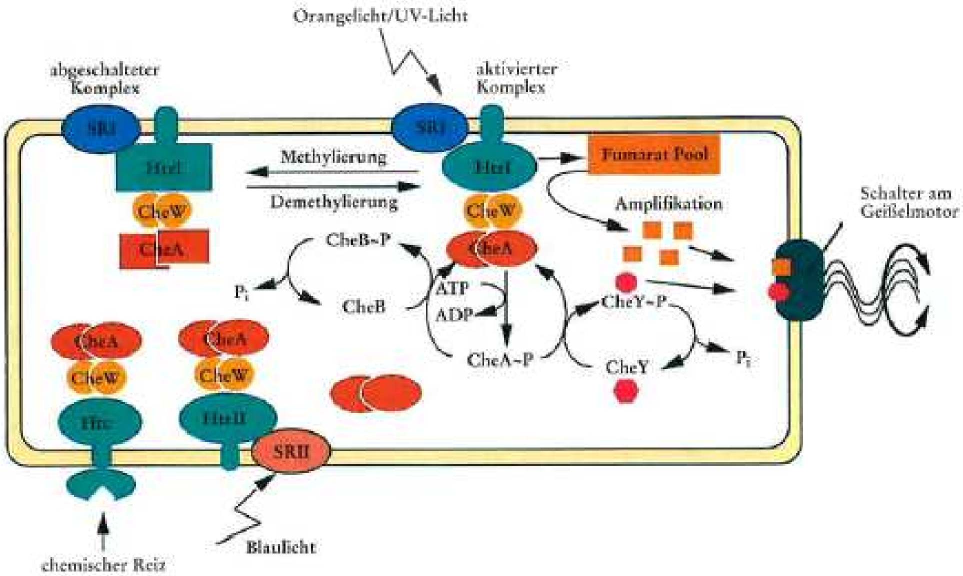
Example 5 - SOLITAIRE GAME

- square component
- counter facilitates reachbility question, but hinders analysis



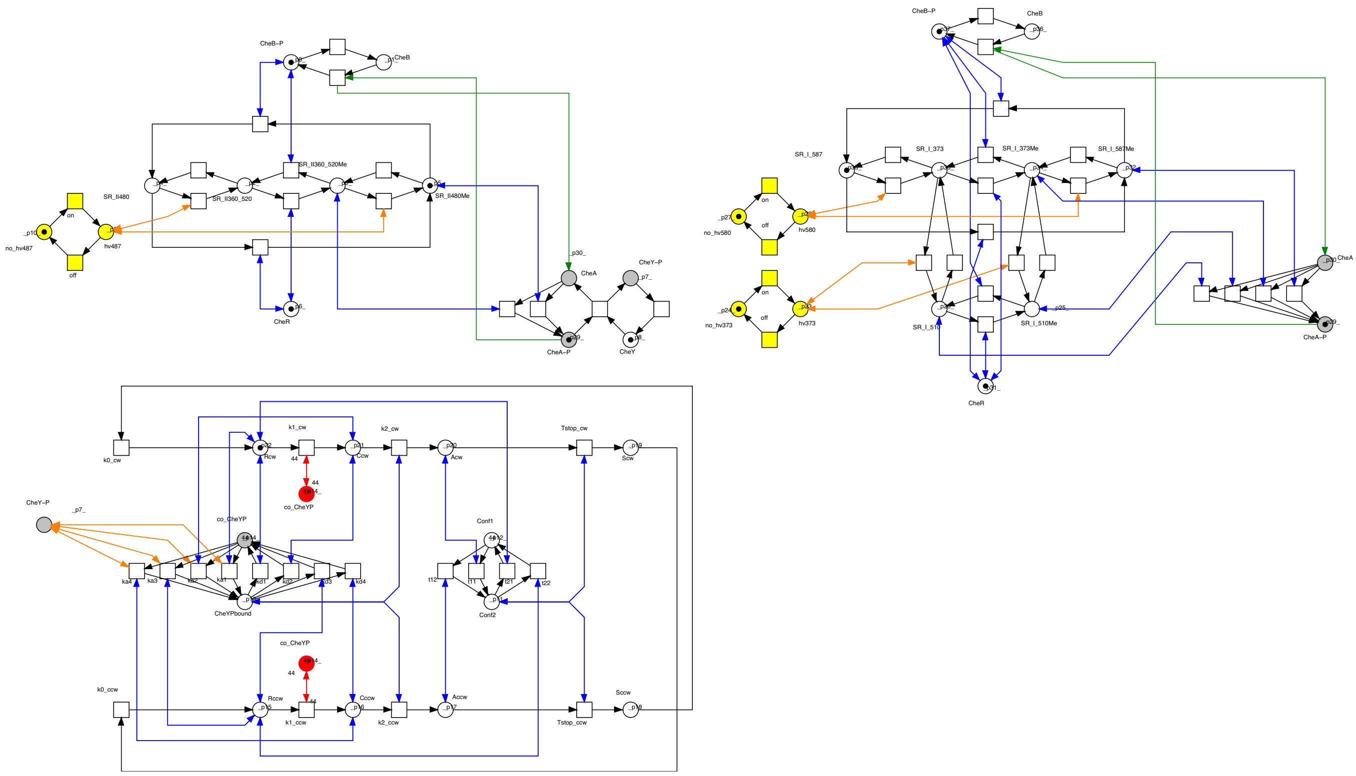


EXAMPLE 6 - HALOBACTERIUM SALINARUM



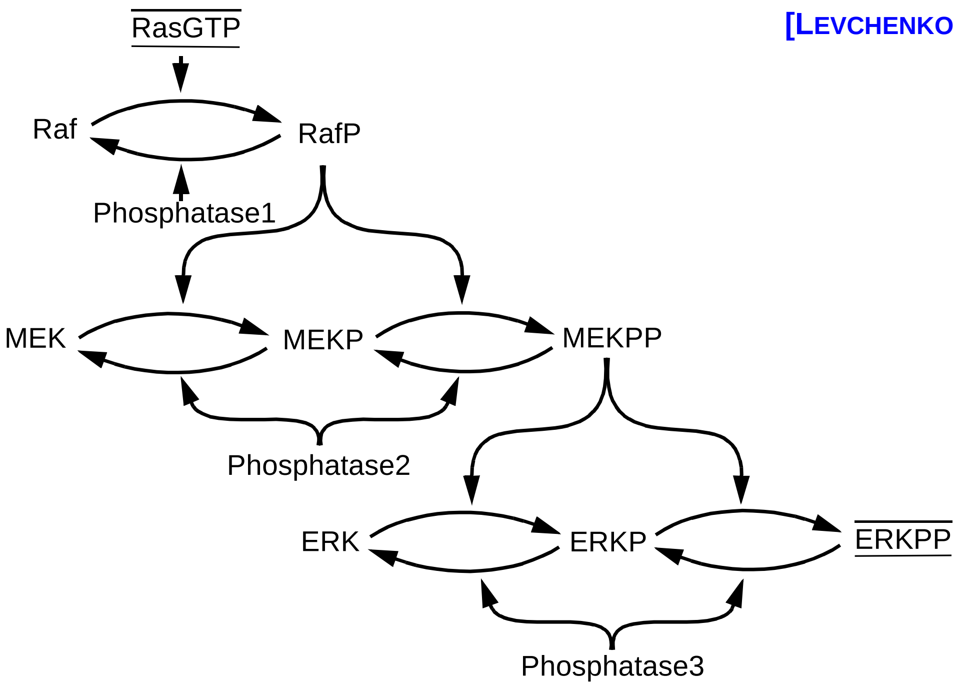
[Marwan; Oesterhelt 1999]

EXAMPLE 6 - HALOBACTERIUM SALINARUM

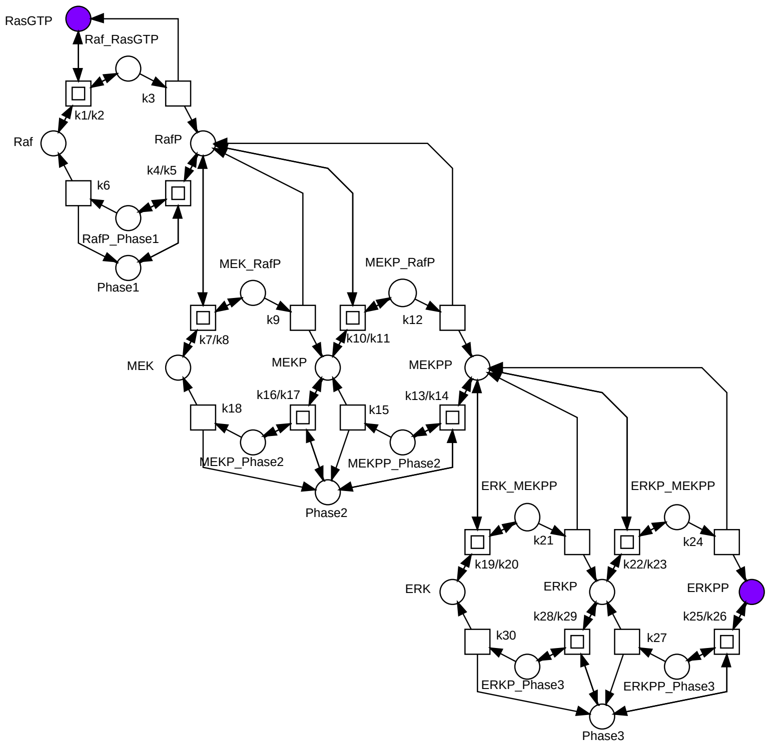


EXAMPLE - MAPK SIGNALLING CASCADE

[LEVCHENKO 2000]



Example - MAPK Signalling Cascade



[Gilbert, Heiner, Lehrack 2007]

[Heiner, Gilbert, Donaldson 2008]

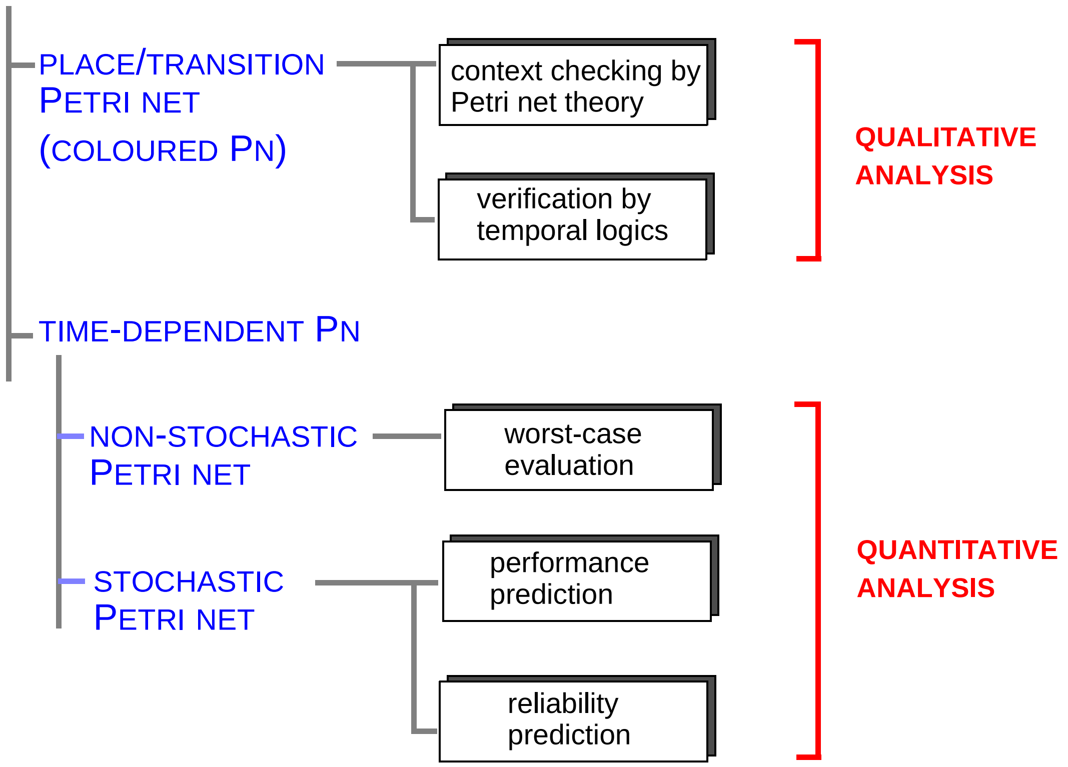
Petri nets, summary

WHY PETRI NETS?

- **a suitable intermediate representation for**
  - -> *different (specification/programming) languages,*
  - -> *different phases of software development cycle,*
  - -> *different validation methods;*
  - -> *technical & natural systems*
- **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**

PETRI NETS, MODEL CLASSES

dependability engineering



monika.heiner(at)b-tu.de
July 2017

Petri nets,
typical properties

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***

GENERAL BEHAVIOURAL PROPERTIES

- **Are there behaviourally invariant subnet structures ?**
  - -> *token conservation* -> *P* - ***INVARIANTS***
  - -> *token distribution reproduction* -> *T* - ***INVARIANTS***

SPECIAL BEHAVIOURAL PROPERTIES

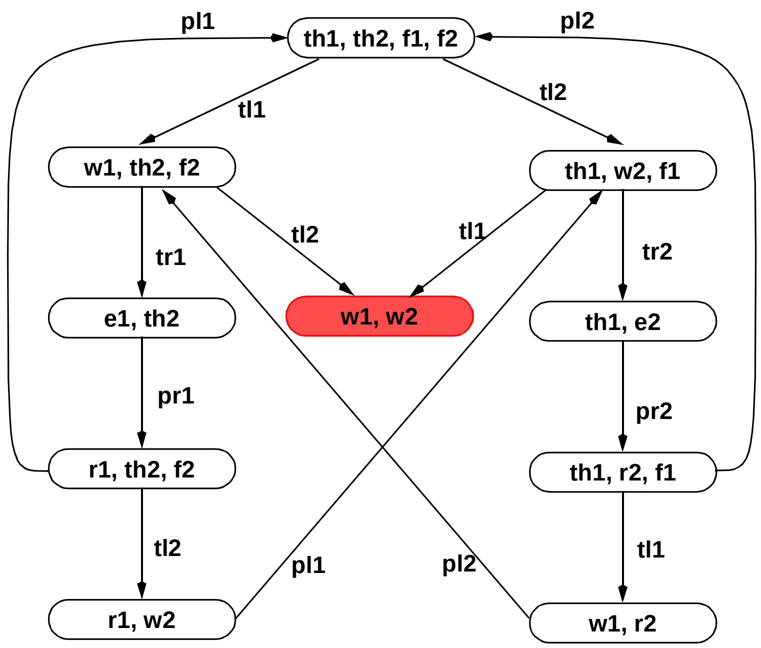
- **. . . and many more -> temporal logics (CTL, LTL)**

Petri nets, typical analysis techniques

MODEL ANIMATION (?)

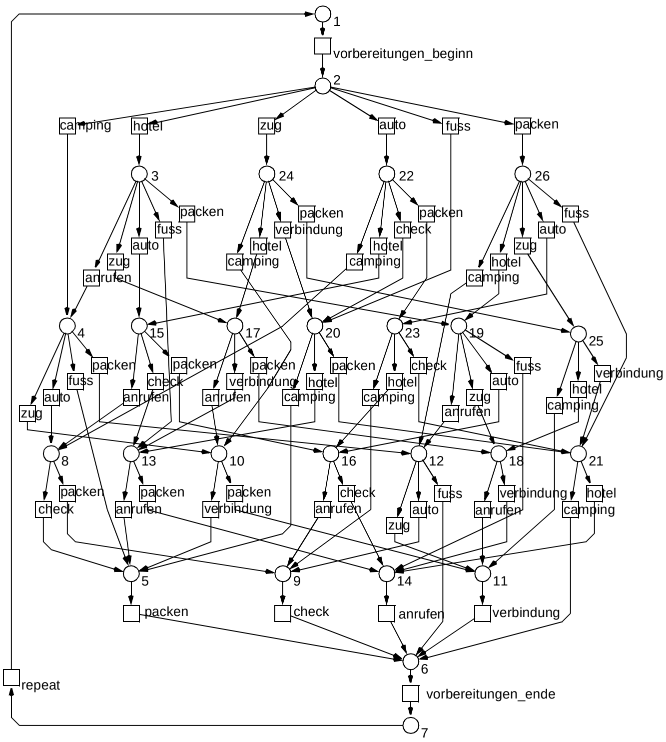
Dynamic analyses

- **reachability / occurrence graph,**
  -> *(labelled) state transition system (-> 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**

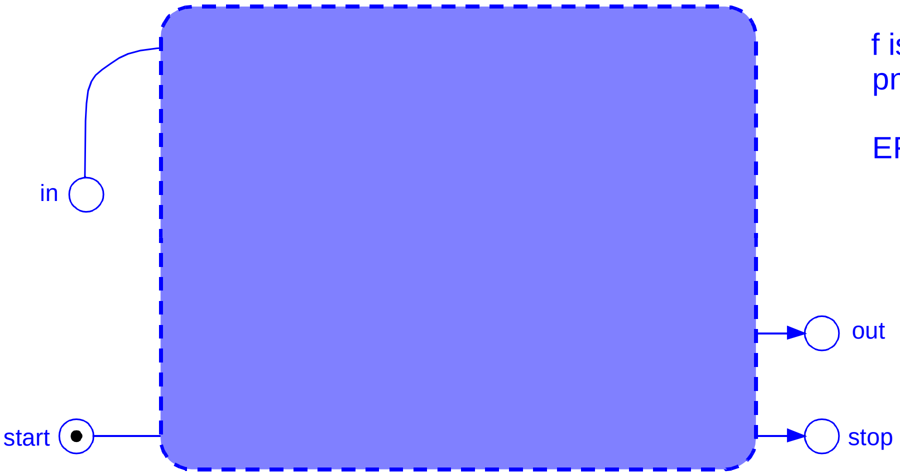


- **boundedness**
  -> *finite graph*
- **reversibility**
  -> *one Strongly Connected Component (SCC)*
- **liveness**
  -> *every transition contained in all terminal SCC*
- **dead states (deadlock)**
  -> *terminal nodes*

-> reachability graphs tend to be huge <-



- **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**

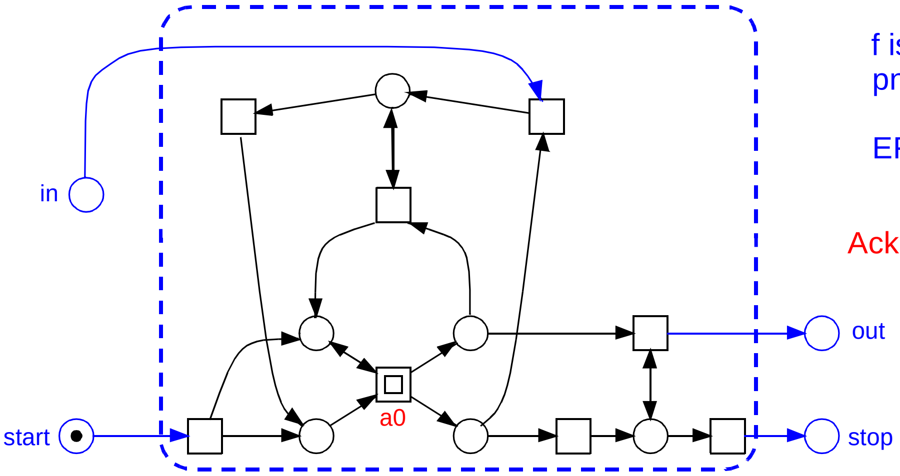


f is weakly pn-computable:

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

REACHABILITY GRAPH, STATE SPACE COMPLEXITY
dependability engineering

- **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:

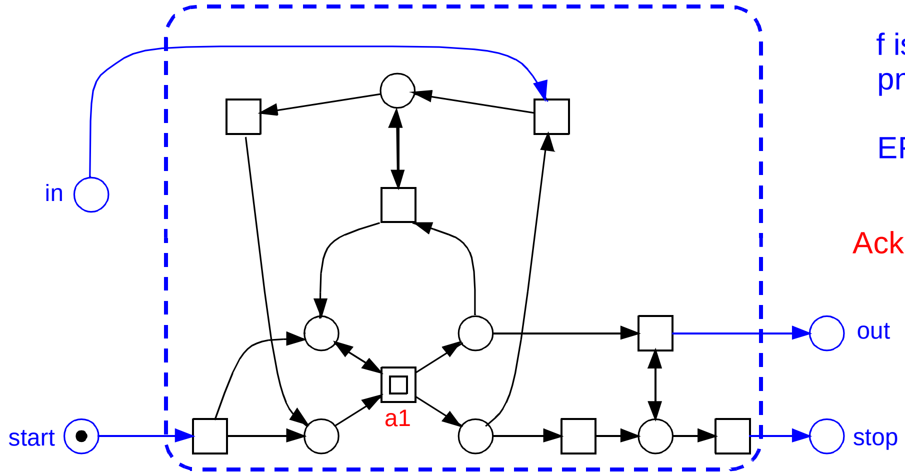
EF(out = f(in) &
stop = 1)

Ackermann function a1

monika.heiner(at)b-tu.de
July 2017

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 ...
- **proof** -> Petri net computer for a function **f: N_0^m -> N_0**

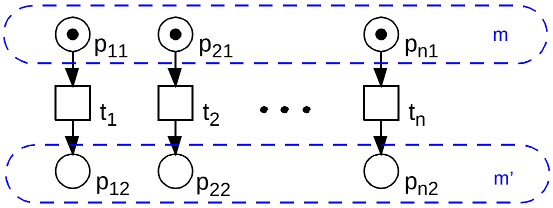


f is weakly pn-computable:

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

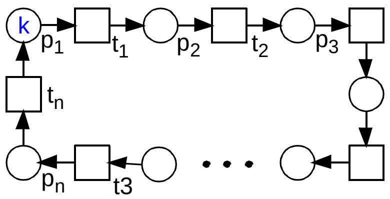
Ackermann function a2

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**
  - -> *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

ANALYSIS TOOLS

-> TOOLBOX <-

- **Petri net theory**
  - -> *INA (HU Berlin)*
  - -> *TINA (LAAS/CNRS)*
  - -> *Charlie*
- **model checking**

		CTL	LTL
-> *reachability graph*	->	*INA, Charlie* *PROD, MARIA*	*Charlie* *PROD, MARIA*
-> *lazy state spaces*			
- stubborn set reduction	->	*LoLA*	*PROD (LTL\X)*
- symmetry reduction	->	*LoLA*	
-> *compressed state spaces (BDD, NDD, ... , IDD)*	->	*bdd-CTL, SMART* *idd-CTL*	*bdd-LTL* *idd-LTL*
-> *Kronecker algebra*	->	*[Kemper]*	
-> *prefix*	->	*PEP (CTL_0)*	*QQ (LTL\X)*
-> *process automata*	->	*[pd]*	

to be continued:

Temporal Logics,

CTL - a crash cours