NoPain – Meeting

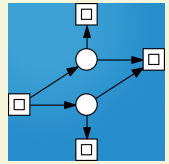
Monika Heiner Christian Rohr

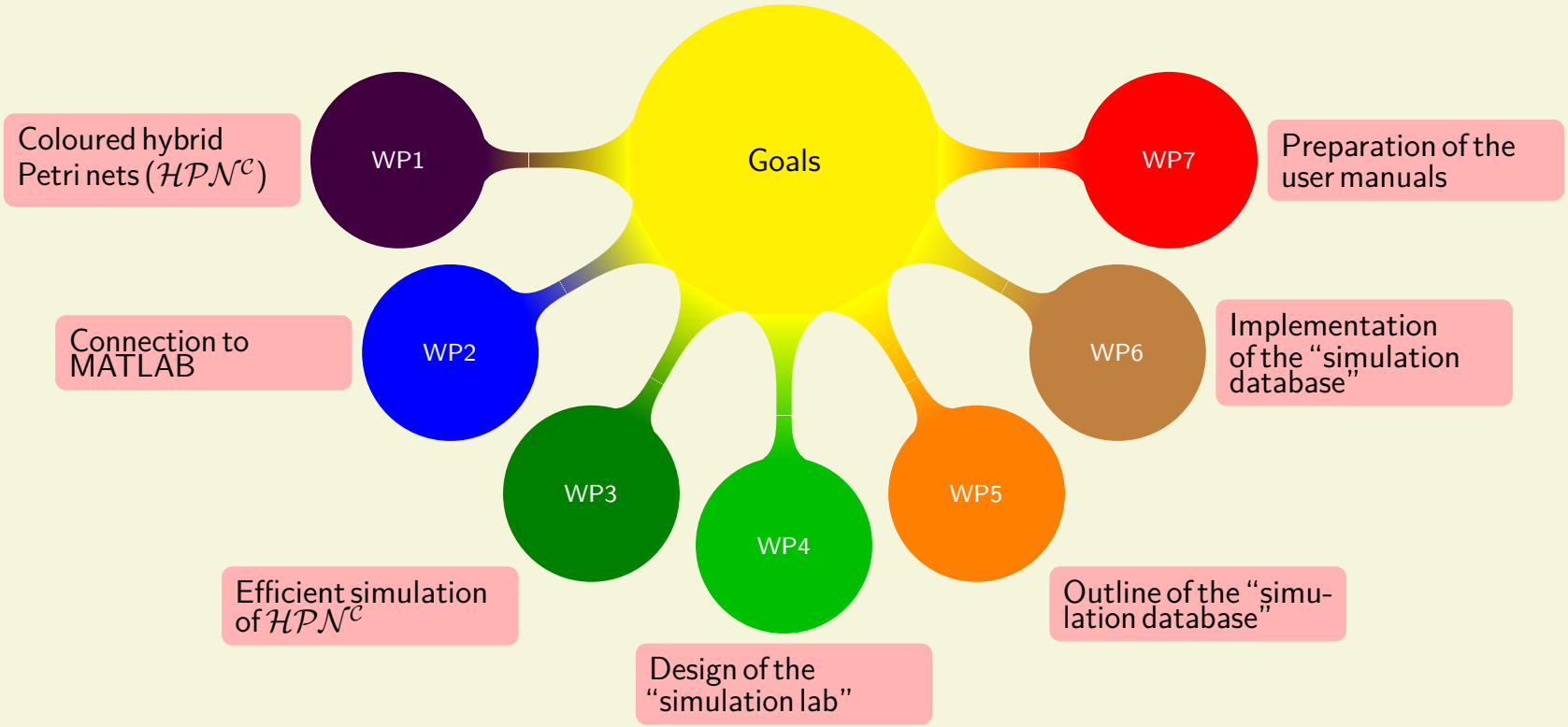
Department of Computer Science
Brandenburg University of Technology Cottbus

http://www-dssz.informatik.tu-cottbus.de

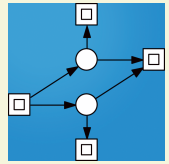
May 13, 2013

Work Packages





WP1



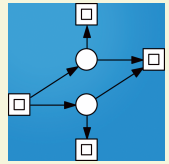
Coloured hybrid Petri nets ($\mathcal{HPN}^{\mathcal{C}}$)

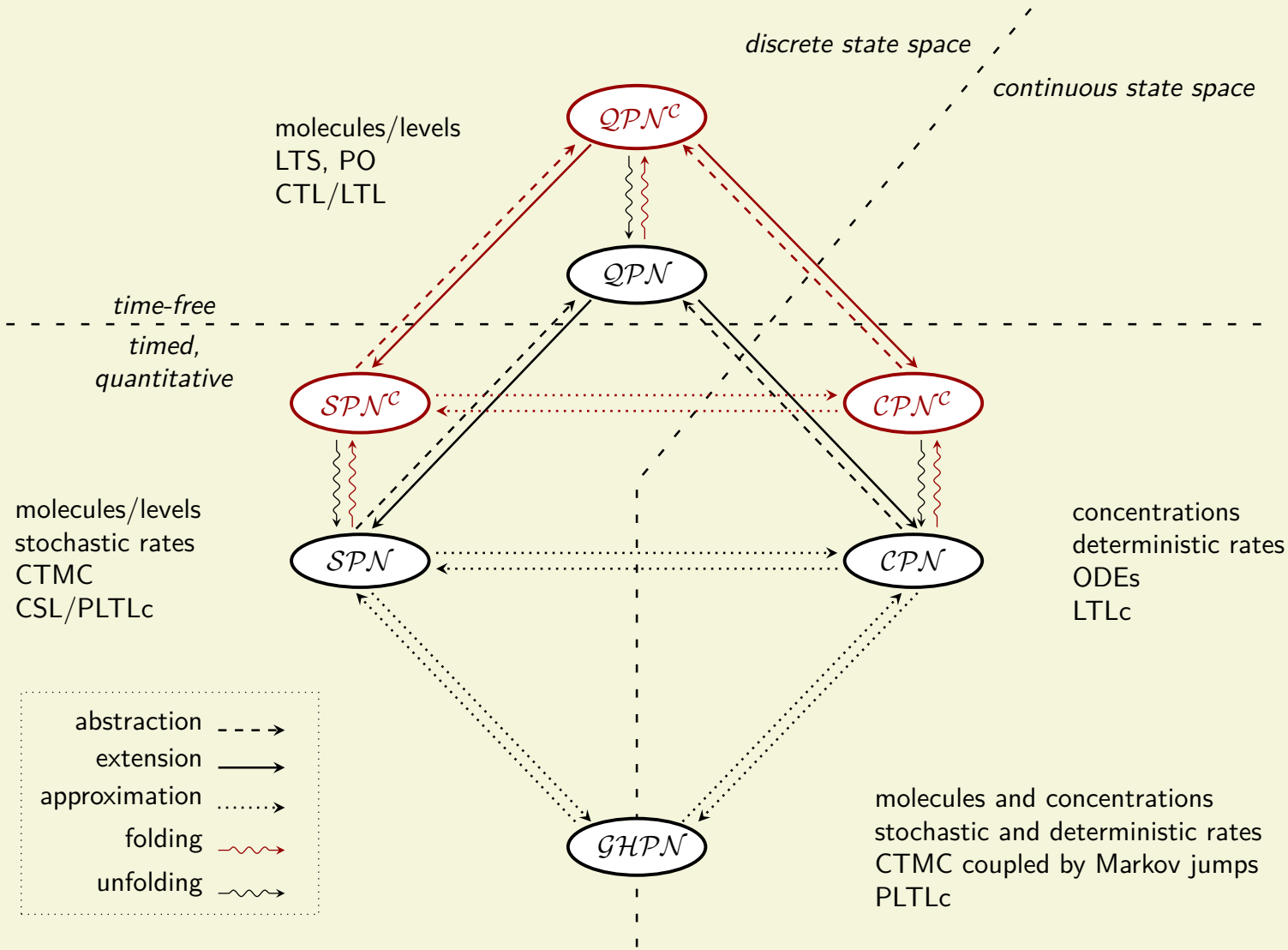
a. Predecessor WPs: -/-

b. Successor WPs: BTU-WP2, BTU-WP7

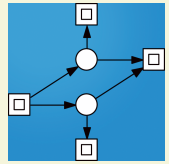
- Uniting the modelling concepts coloured and hybrid Petri nets in Snoopy
- Expansions around new special modelling features like constants and self-modifying edges with marking-dependent arc weights

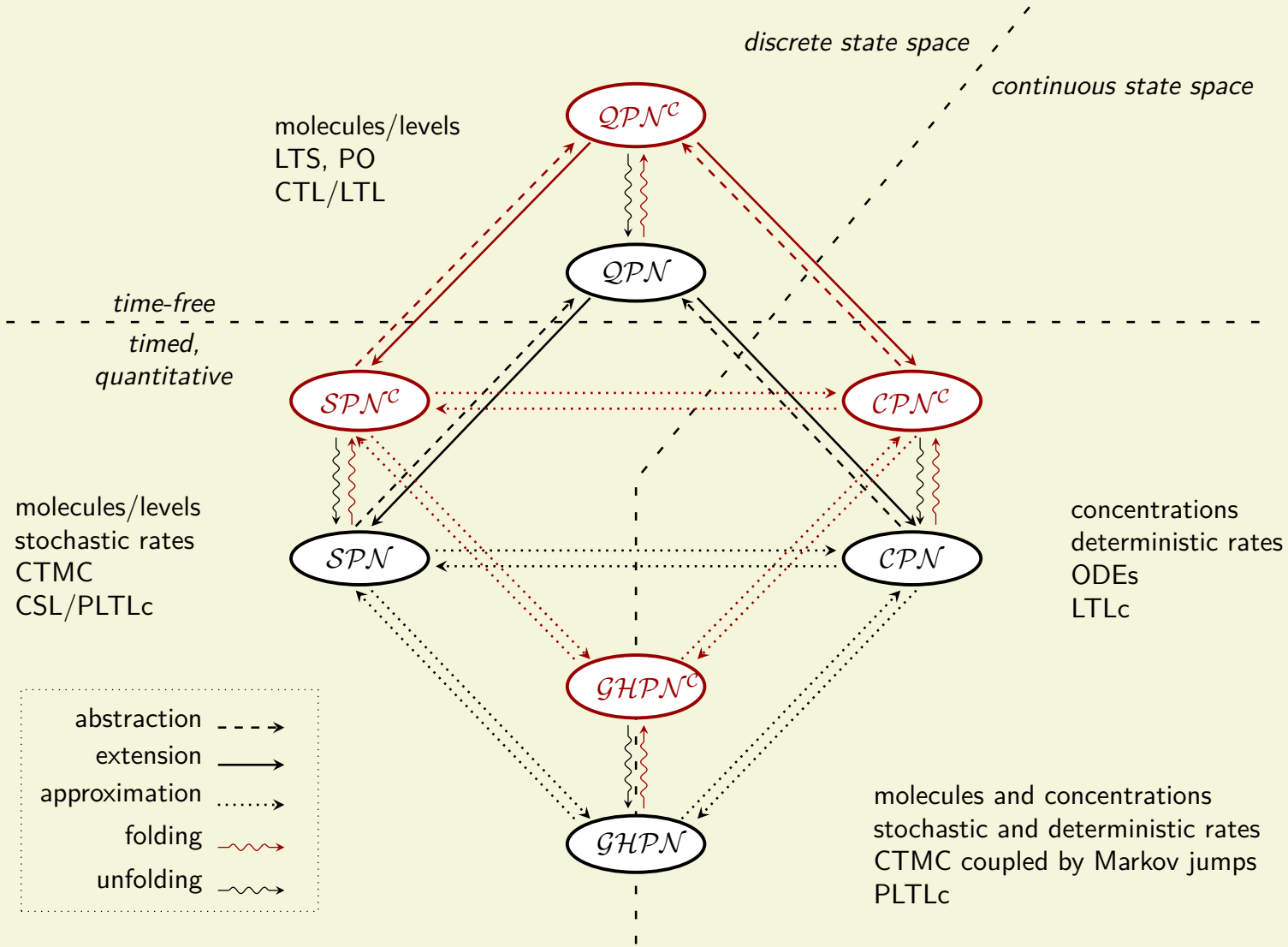
$\mathcal{HPN}^{\mathcal{C}}$



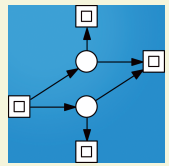


$\mathcal{HPN}^{\mathcal{C}}$

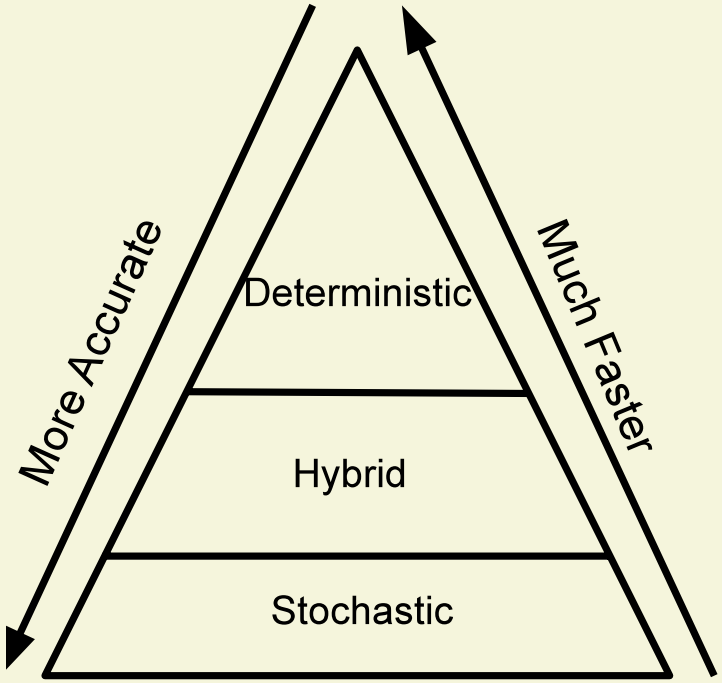




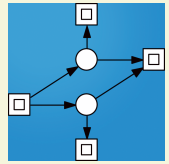
$\mathcal{HPN}^{\mathcal{C}}$



- an intermediate approximation between SPN and CPN
- level of approximation controlled by the number of transitions and places that belong to each category
- static partitioning, dynamic partitioning
- combining the modelling power of coloured Petri nets and hybrid Petri nets



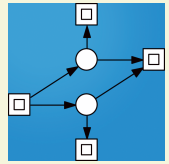
Constants



Disadvantages till now

- different kinds of value lists
  → marking list, function list, parameter list
- inconsistent use of value lists
  → not available in all net classes
- misused node class for parameters
  → parameters are not part of the actual Petri net
- no dependencies between different attributes of different net elements
  → user has to take care of dependencies manually
  → redundant data, error prone

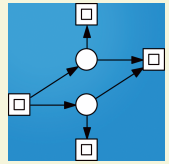
Constants



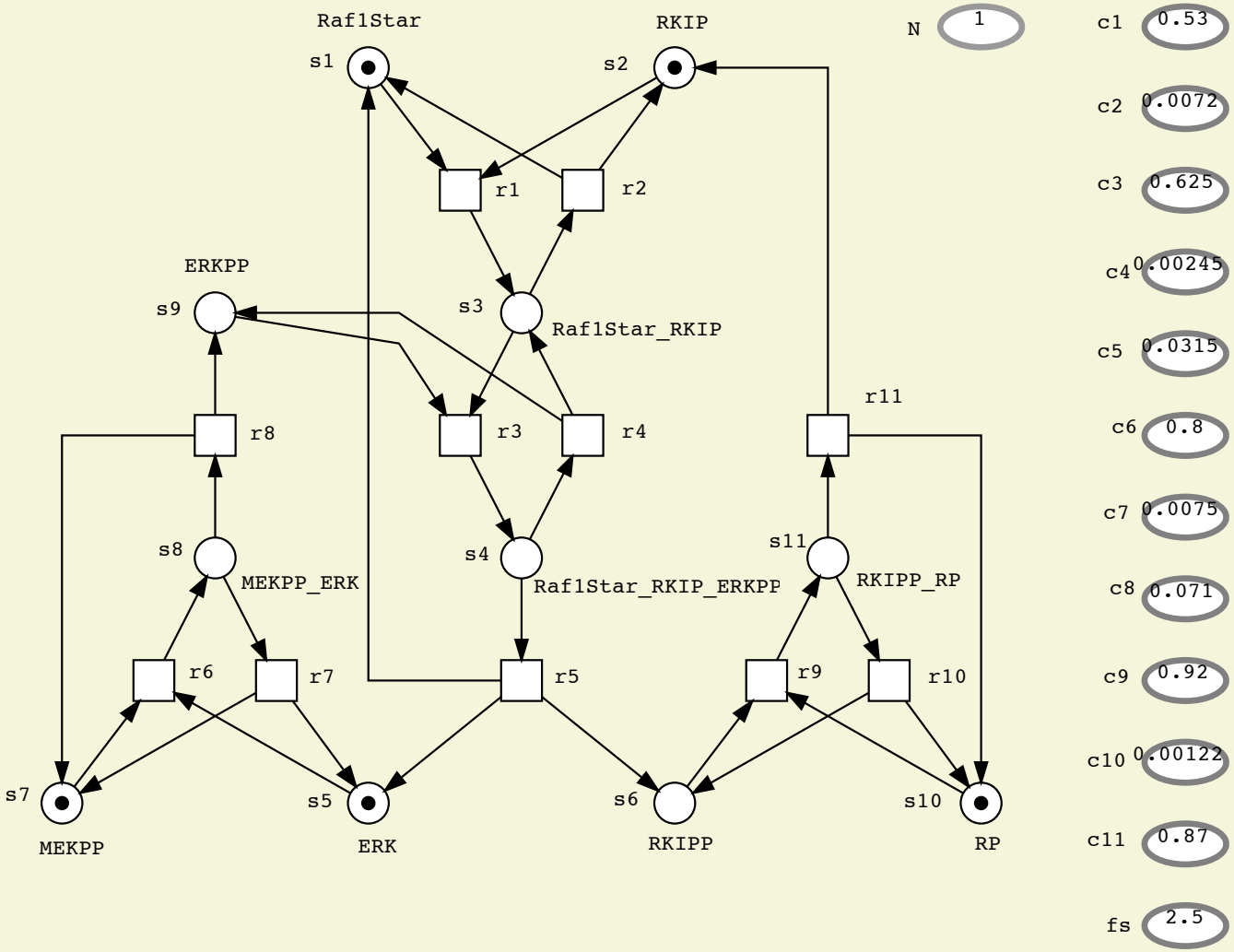
Attributes with value lists till now

	$\mathcal{PN}$	$\mathcal{XPN}$	$\mathcal{CPN}$	$\mathcal{SPN}$	$\mathcal{GHPN}$	$\mathcal{TPN}$
Marking			✓	✓	✓	✓
Arc weight						
Rate function			✓	✓	✓	
Weight				✓	✓	
Time delay				✓	✓	
Time interval				✓	✓	✓

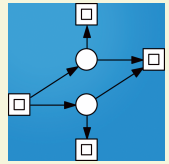
Constants



RKIP inhibited ERK Pathway



Constants



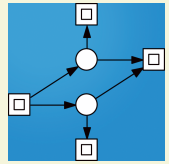
RKIP inhibited ERK Pathway

	Main	N5	N10	N100	N1000	N10000
Raf1Star	1	5	10	100	1000	10000
RKIP	1	5	10	100	1000	10000
Raf1Star_RKIP	0	0	0	0	0	0
ERKPP	0	0	0	0	0	0
MEKPP_ERK	0	0	0	0	0	0
Raf1Star_RKIP_ERKPP	0	0	0	0	0	0
RKIPP_RP	0	0	0	0	0	0
MEKPP	1	5	10	100	1000	10000
ERK	1	5	10	100	1000	10000
RKIPP	0	0	0	0	0	0
RP	1	5	10	100	1000	10000

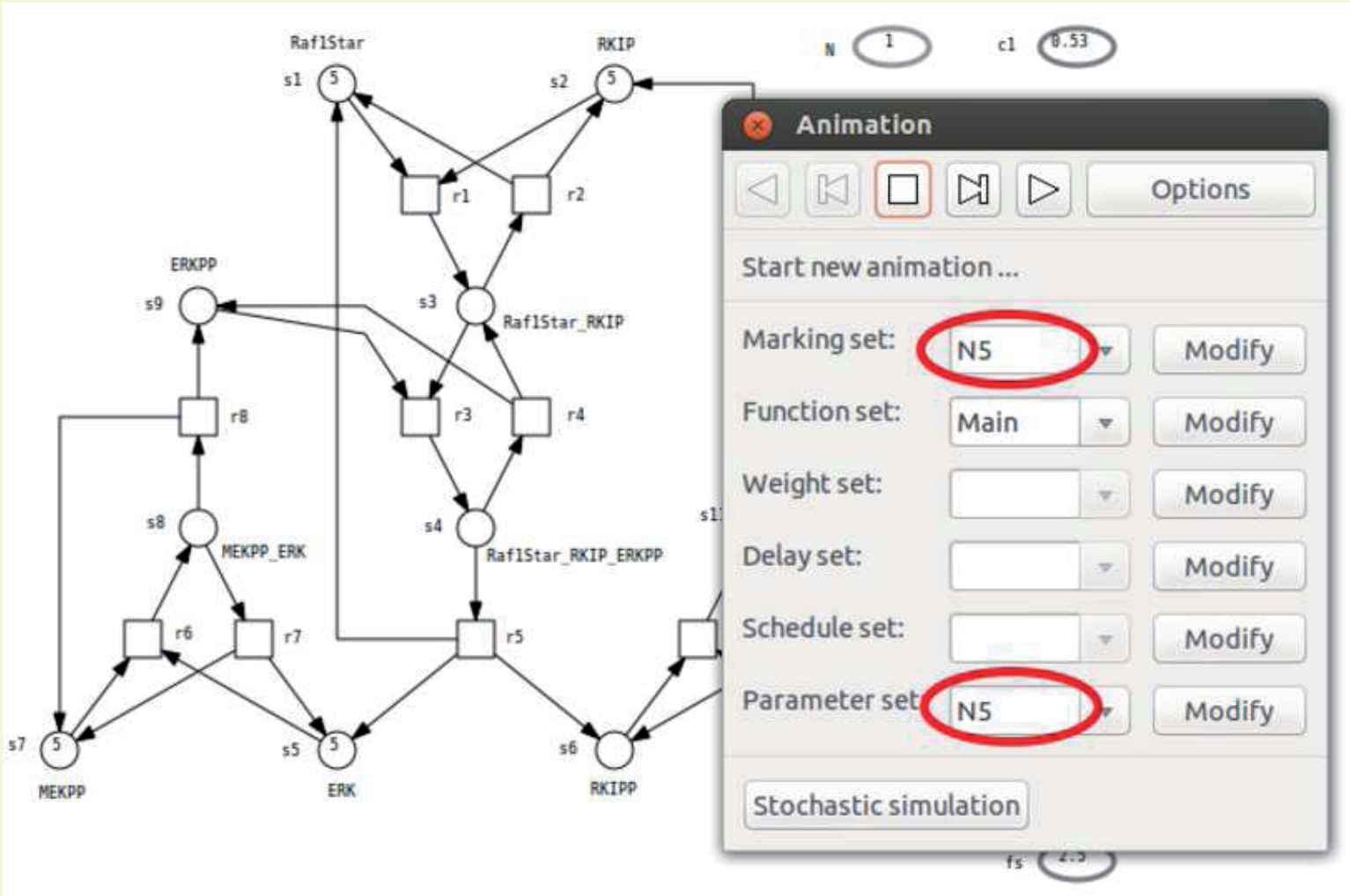
	Main	N5	N10	N100	N1000	N10000
c1	0.53	0.53	0.53	0.53	0.53	0.53
c2	0.0072	0.0072	0.0072	0.0072	0.0072	0.0072
c3	0.625	0.625	0.625	0.625	0.625	0.625
c4	0.00245	0.00245	0.00245	0.00245	0.00245	0.00245
c5	0.0315	0.0315	0.0315	0.0315	0.0315	0.0315
c6	0.8	0.8	0.8	0.8	0.8	0.8
c7	0.0075	0.0075	0.0075	0.0075	0.0075	0.0075
c8	0.071	0.071	0.071	0.071	0.071	0.071
c9	0.92	0.92	0.92	0.92	0.92	0.92
c10	0.00122	0.00122	0.00122	0.00122	0.00122	0.00122
c11	0.87	0.87	0.87	0.87	0.87	0.87
fs	2.5	2.5	2.5	2.5	2.5	2.5
N	1	5	10	100	1000	10000

	Main	F-Set 2	level	D
r1	MassAction(c1*fs/N)	MassAction(c1)	MassAction(c1/fs)	c1 * Raf1Star * RKIP
r2	MassAction(c2)	MassAction(c2)	MassAction(c2)	c2 * Raf1Star_RKIP
r3	MassAction(c3*fs/N)	MassAction(c3)	MassAction(c3/fs)	c3 * Raf1Star_RKIP * ERKPP
r4	MassAction(c4)	MassAction(c4)	MassAction(c4)	c4 * Raf1Star_RKIP_ERKPP
r6	MassAction(c6*fs/N)	MassAction(c6)	MassAction(c6/fs)	c6 * MEKPP *ERK
r7	MassAction(c7)	MassAction(c7)	MassAction(c7)	c7 * MEKPP_ERK
r9	MassAction(c9*fs/N)	MassAction(c9)	MassAction(c9/fs)	c9 * RP * RKIPP
r10	MassAction(c10)	MassAction(c10)	MassAction(c10)	c10 * RKIPP_RP
r5	MassAction(c5)	MassAction(c5)	MassAction(c5)	c5 * Raf1Star_RKIP_ERKPP
r8	MassAction(c8)	MassAction(c8)	MassAction(c8)	c8 * MEKPP_ERK
r11	MassAction(c11)	MassAction(c11)	MassAction(c11)	c11 * RKIPP_RP

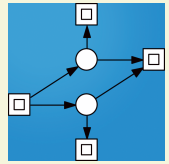
Constants



RKIP inhibited ERK Pathway



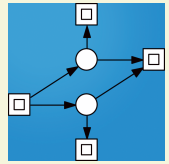
Constants



Advantages of new Constants

- new metadata class "Constants"
  $\rightarrow$ not part of the net structure any more
- central place for definition of constants
- consistent use of constants
  $\rightarrow$ available in all net classes
- dependencies between different attributes of different net elements possible
  $\rightarrow$ no redundant data

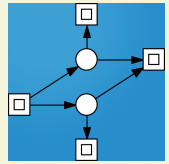
Constants



Attributes with constants

	$\mathcal{PN}$	$\mathcal{XPN}$	$\mathcal{CPN}$	$\mathcal{SPN}$	$\mathcal{GHPN}$	$\mathcal{TPN}$
Marking	✓	✓	✓	✓	✓	✓
Arc weight	✓	✓	✓	✓	✓	✓
Rate function			✓	✓	✓	
Weight				✓	✓	
Time delay				✓	✓	
Time interval				✓	✓	✓

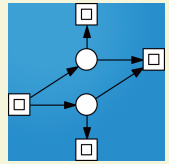
Constants



Advantages of new Constants

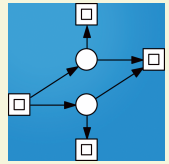
- integer and real valued constants
  $\rightarrow$ $\mathcal{PN}$, $\mathcal{XPN}$ only integer constants, other net classes both
- arithmetic expressions $(+, -, *, /)$
  $\rightarrow$ only values ≥ 0 allowed
- dependencies between constants
  $\rightarrow$ $c_1 = 1,\ c_2 = 2 * c_1,\ c_3 = c_2 + 1$
- new grouping concept for constants
  $\rightarrow$ combinatorial diversity
- change groups at run-time

Constants

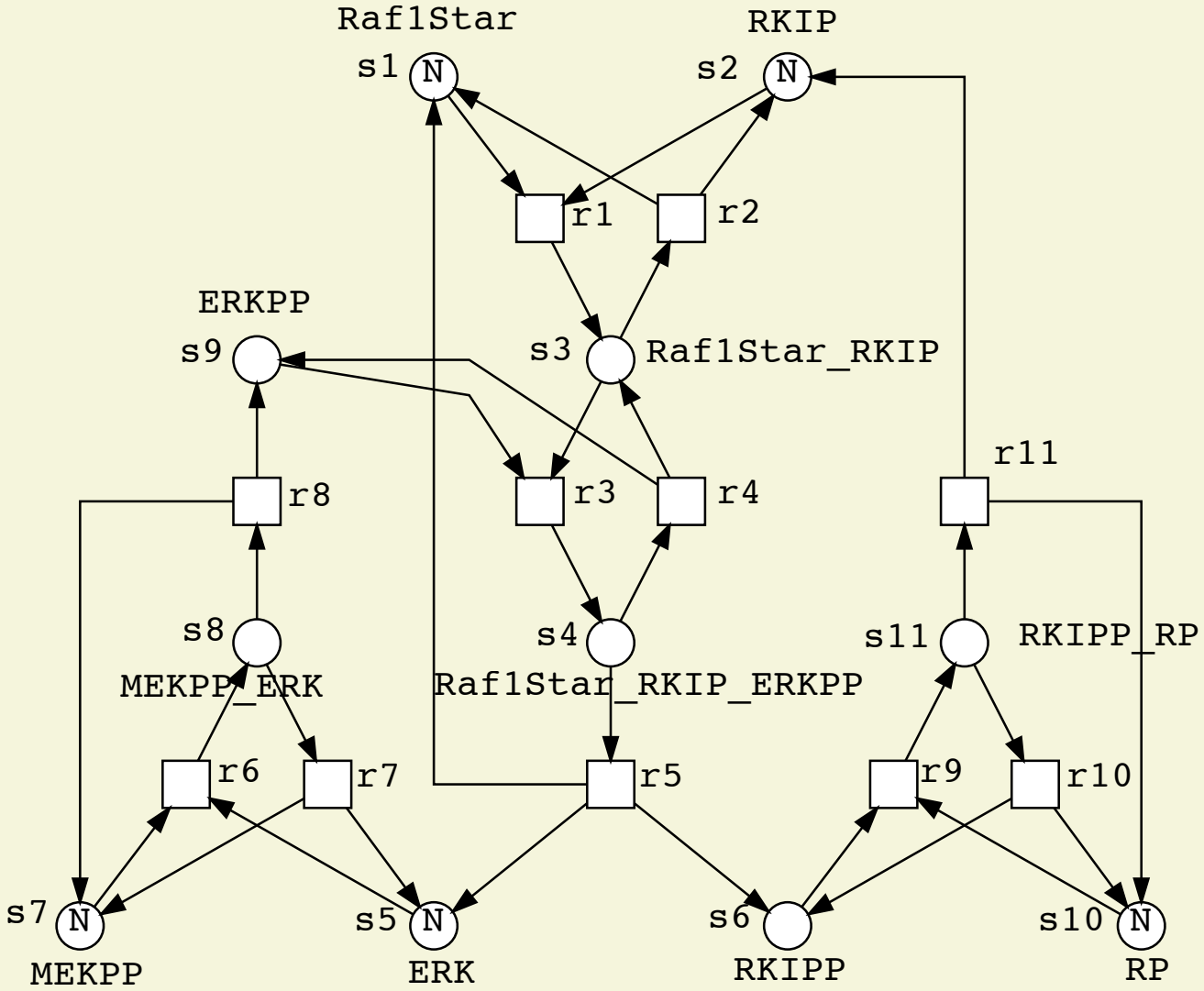


Advantages of new Constants

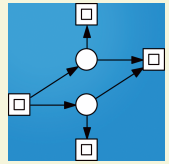
Constant	Group	Type	Main	V-Set 2	V-Set 3	V-Set 4
N1	marking	int	1	2	3	4
N2	marking	int	2*N1			
k1	parameter	double	0.15		0.45	
k2	parameter	double	0.3			0.6
c1	all	int	10			
c2	all	int	c1+10	c1+20	c1+30	c1+40



RKIP inhibited ERK Pathway



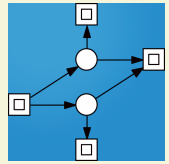
Constants



RKIP inhibited ERK Pathway

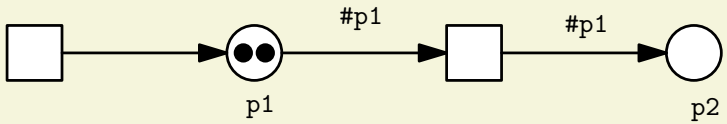
	Constant	Group	Type	Main	N5	N10	N100	N1000	N10000
1	N	marking	int	1	5	10	100	1000	10000
2	c1	parameter	double	0.53					
3	c2	parameter	double	0.0072					
4	c3	parameter	double	0.625					
5	c4	parameter	double	0.00245					
6	c5	parameter	double	0.0315					
7	c6	parameter	double	0.8					
8	c7	parameter	double	0.0075					
9	c8	parameter	double	0.071					
10	c9	parameter	double	0.92					
11	c10	parameter	double	0.00122					
12	c11	parameter	double	0.87					
13	fs	parameter	double	2.5					

Marking-dependent arc weights

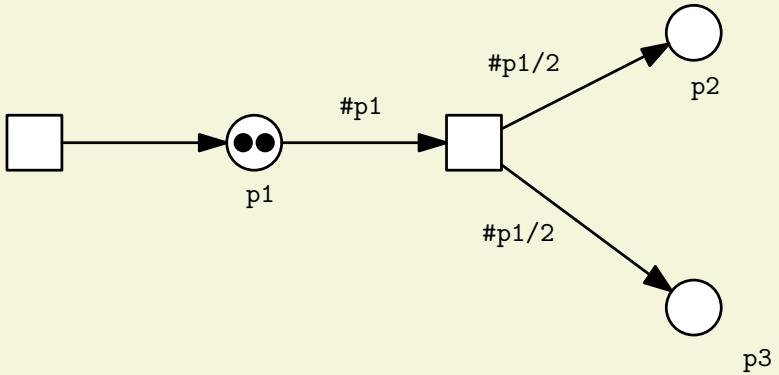


Reasons

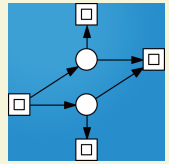
- transfer of tokens in one single step



- division of tokens, e.g. cell division

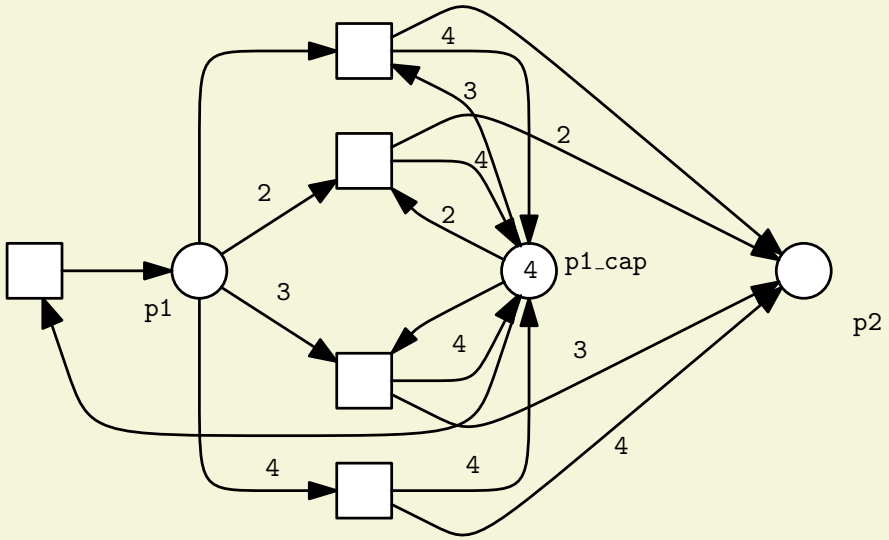


Marking-dependent arc weights

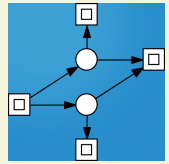


Possible with restrictions

- only bounded nets
- complementary place or inhibitor arcs
- need to model every possible marking
  $\rightarrow$ blow up net size, error prone

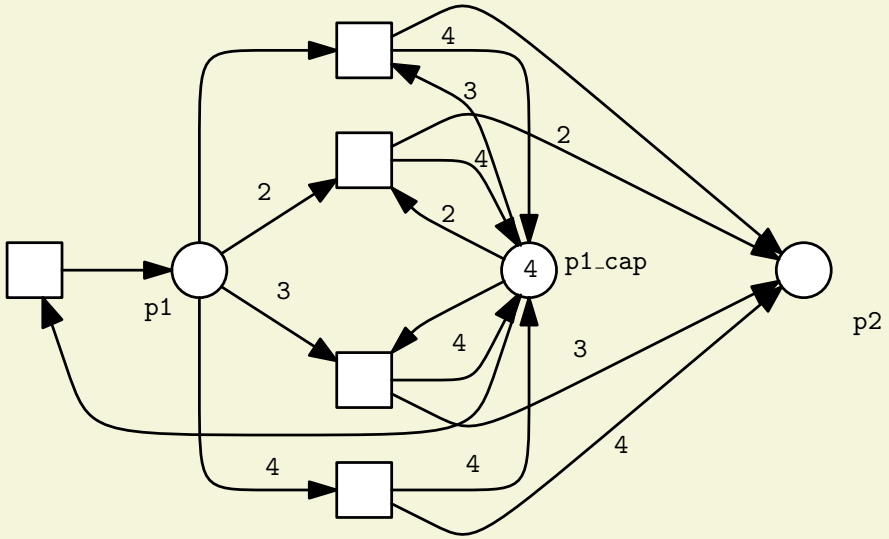


Marking-dependent arc weights



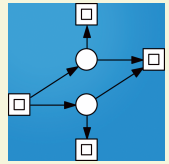
Possible with restrictions

- only bounded nets
- complementary place or inhibitor arcs
- need to model every possible marking
  $\rightarrow$ blow up net size, error prone



Not possible for unbounded Petri nets

Marking-dependent arc weights



Definition

Petri net $\mathcal{PN} = (P, T, f, m_0)$ is defined as followed

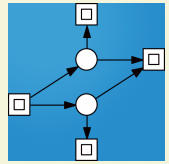
P a finite, non empty set of places

T a finite, non empty set of transitions

f: $((P \times T) \cup (T \times P)) \to \mathbb{N}_0$
(weighted directed arcs)

m_0: initial marking, whereby a marking $m \in \mathbb{N}^{|P|}$

Marking-dependent arc weights

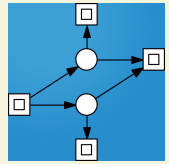


Definition

Petri net with marking dependent arc weights $\mathcal{PN} = (P, T, f, m_0)$ is defined as followed

- P a finite, non empty set of places
- T a finite, non empty set of transitions
- f: $((P \times T) \cup (T \times P)) \times \mathbb{N}_0^{|P|} \to \mathbb{N}_0$
  (directed arcs with marking dependent weights)
- m_0: initial marking, whereby a marking $m \in \mathbb{N}^{|P|}$

Marking-dependent arc weights



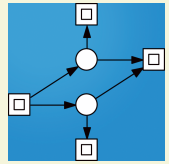
Definition

Petri net with marking dependent arc weights $\mathcal{PN} = (P, T, f, m_0)$ is defined as followed

- P a finite, non empty set of places
- T a finite, non empty set of transitions
- f: $((P \times T) \cup (T \times P)) \times \mathbb{N}_0^{|P|} \to \mathbb{N}_0$
  (directed arcs with marking dependent weights)
- m_0: initial marking, whereby a marking $m \in \mathbb{N}^{|P|}$

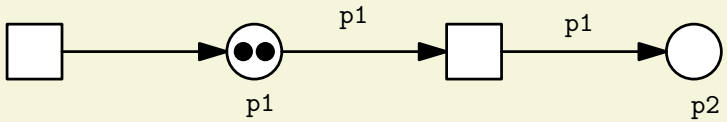
Arc weights can contain place names.

Marking-dependent arc weights

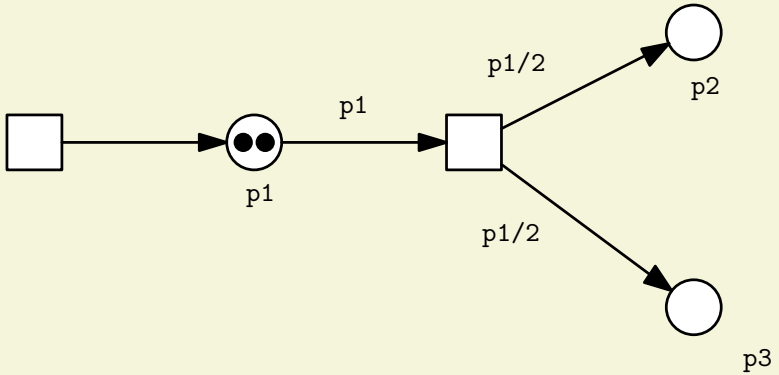


Examples

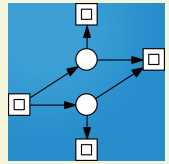
- transfer of tokens in one single step



- division of tokens, e.g. cell division

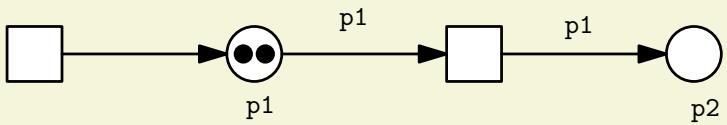


Marking-dependent arc weights

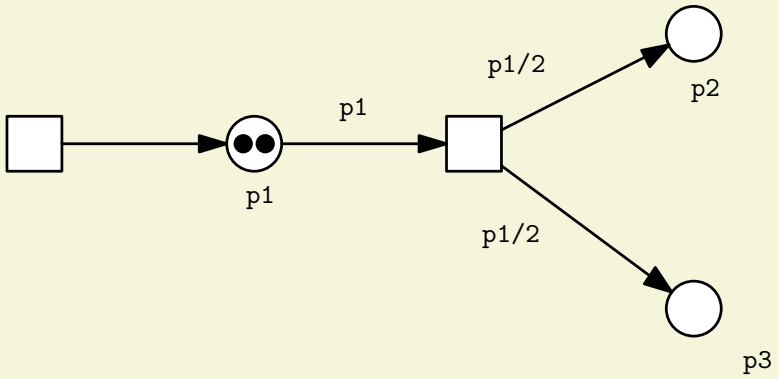


Examples

- transfer of tokens in one single step

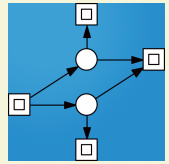


- division of tokens, e.g. cell division

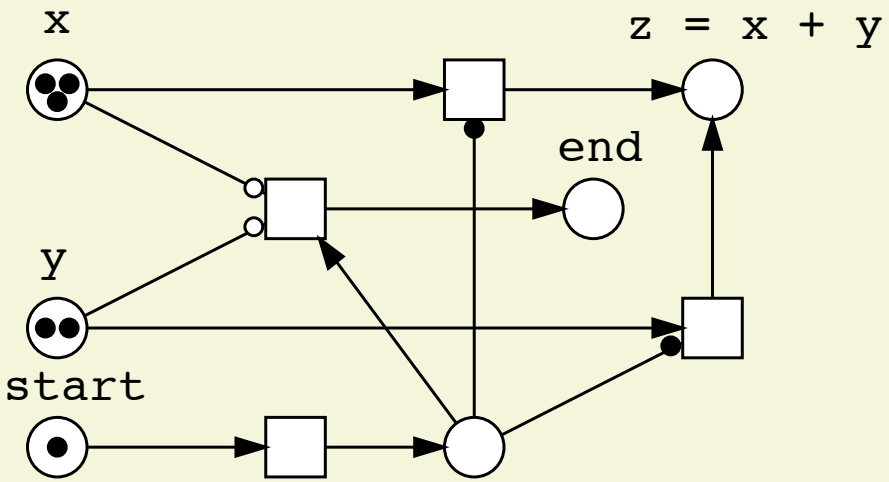


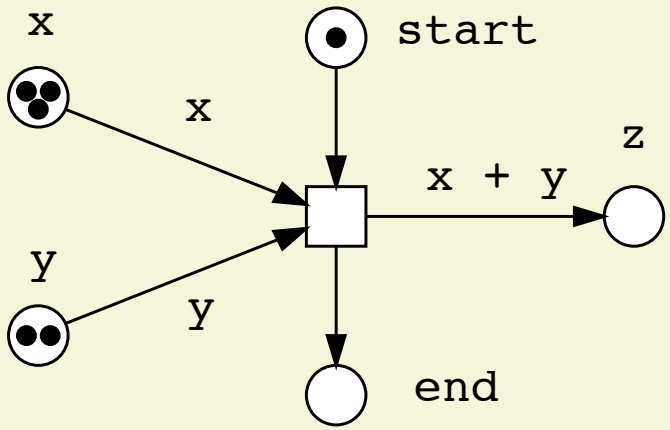
Pitfall: transition always enabled without further preconditions.

Marking-dependent arc weights

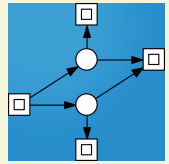


Addition of tokens



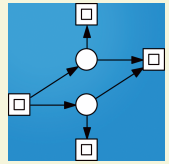


Milestones



	2013				2014				2015			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
WP1		M1										
WP2				M2								
WP3						M3						
WP4								M4				
WP5								M4				
WP6												M5
WP7												M6

Next steps...

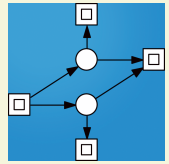


Connection to MATLAB

- Connection of the MATLAB software package for hybrid Petri nets as well as relevant net classes

1. Export the net structure in MATLAB format
   $\hookrightarrow$ readable by Octave too

2. Direct connection from Snoopy to MATLAB through its API, e.g. call MATLAB functions from within Snoopy



Thank you for your attention!