Decolourisation of Stochastic Symmetric Nets with Bags

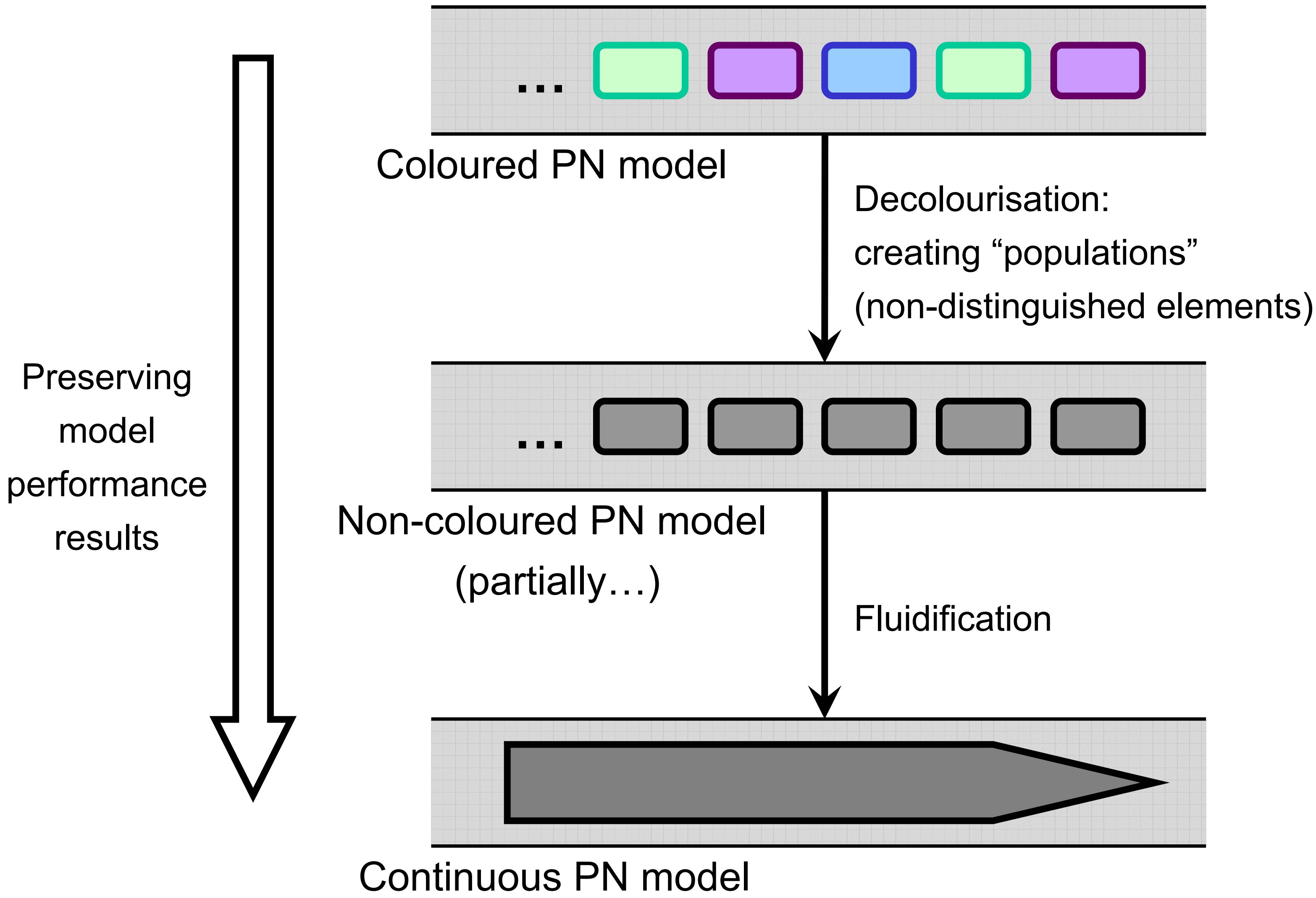
Michal Žarnay & Manuel Silva
(University of Žilina, University of Zaragoza)
18 March 2010

Motivation

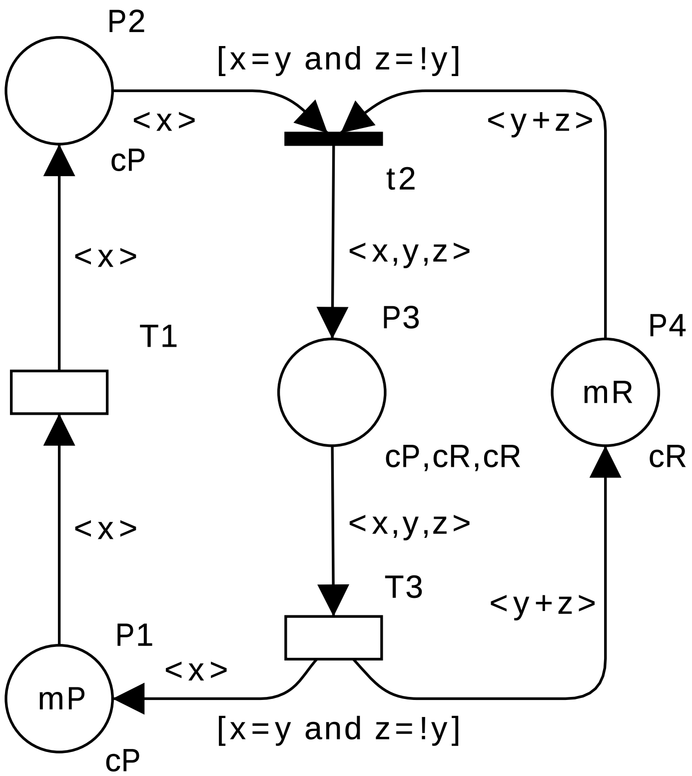
- Coloured Petri nets (CPN)
  - through “colours”, identities are modelled
  - natural description of systems with elements of various attributes
- Problem for highly populated systems: too large state space to be analyzed in reasonable time
- Continuous place/transition nets:
  - Lights: analysis of some highly populated systems
  - Shadows: Can any DES model be fluidified?
- How about transforming CPN-s to continuous P/T nets?
- Not interested in fluid-coloured nets, because identities lead to binary or small numbers

- Timed classes used
  - Coloured PN-s: Stochastic symmetric nets with bags (SSNB)
  - Non-coloured PN-s: Generalized stochastic Petri nets (GSPN)

Illustrating our desired procedure: Two steps from timed coloured to timed continuous Petri net



Getting the flavour: Dining philosophers (Dijkstra)

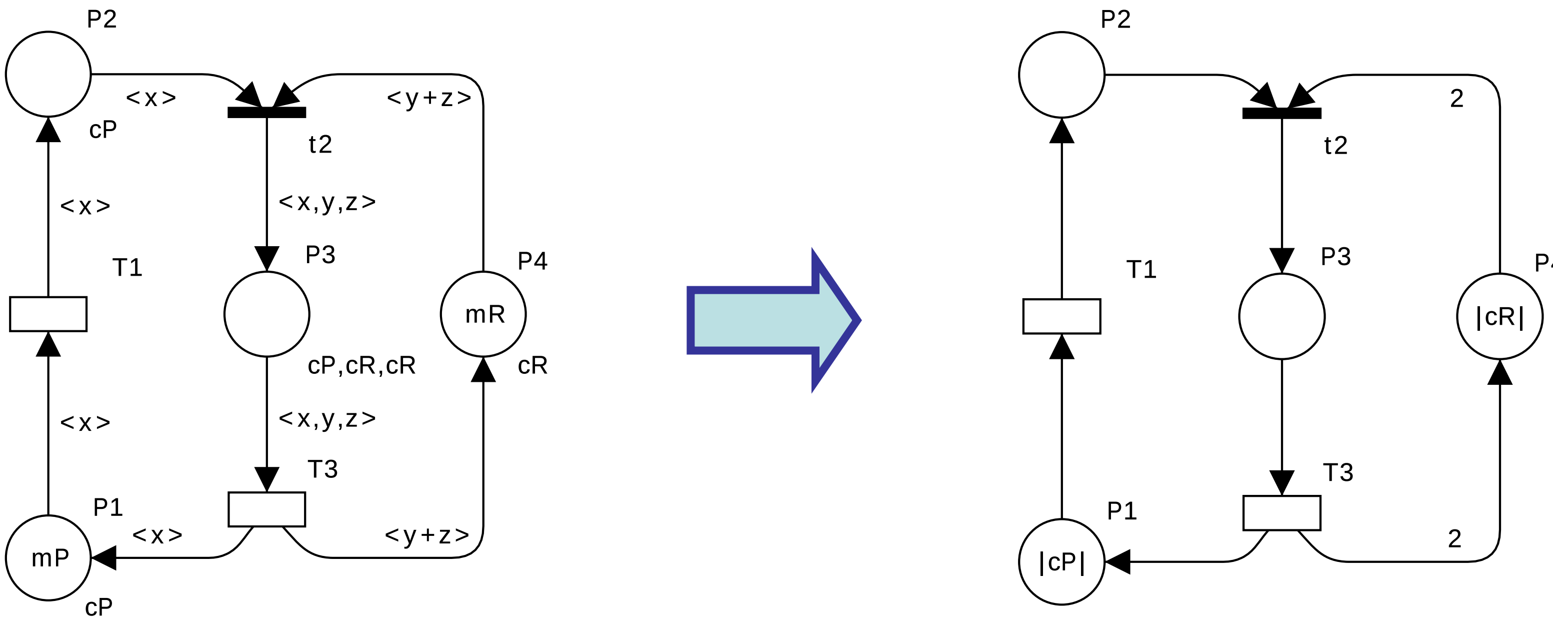


- *P* philosophers think (*P*1) and eat (*P*3) at one table sharing *P* forks.
- Philosopher *x* can use only forks *x* and *x*⊕1 ("!*x*").
- Philosophers decide to eat after some time of thinking (*T*1).
- They start eating only when their relevant forks are free (*P*4 enables *t*2). Otherwise they wait for them (*P*2).
- They keep forks until they finish eating (*T*3).

- **Structural and behavioural symmetries, but…**
- **This model <u>cannot</u> be decolourised due to the use of different resources**

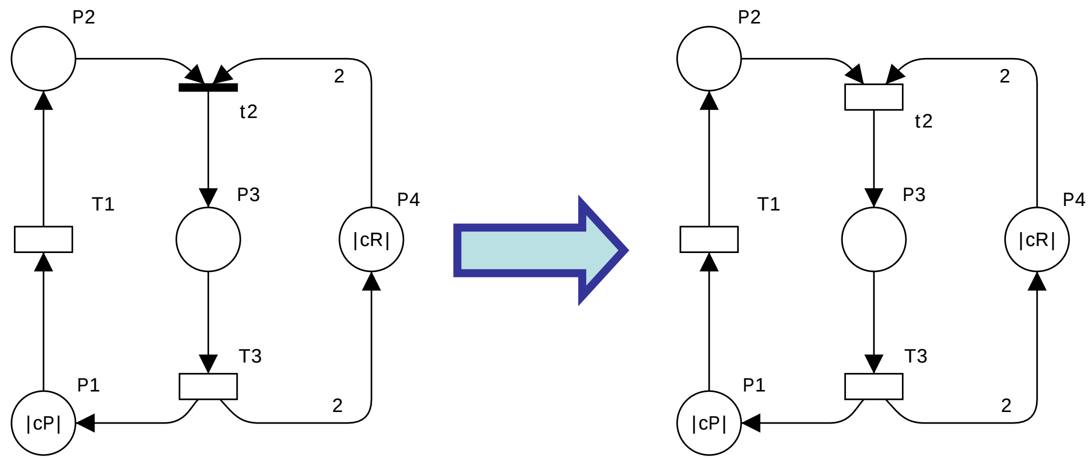
Getting the flavour: Dining philosophers – decolourisation

Now, let's assume that **resources become common** (non-Dijkstra): any philosopher can take any couple of forks (guards on *t*2 and *T*3 disappeared)



This model can be decolourised because of using common resources

Getting the flavour: DinPhilCommon – fluidification



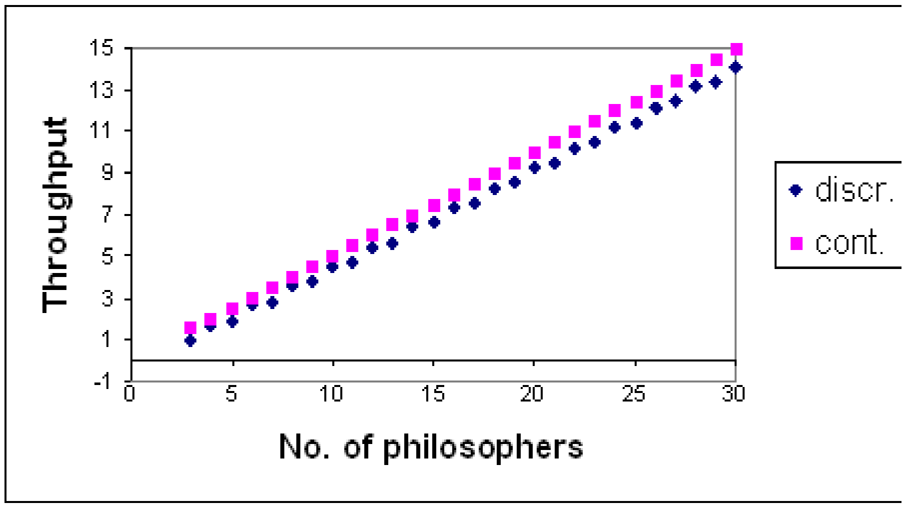
- t2 changed
- time delays:

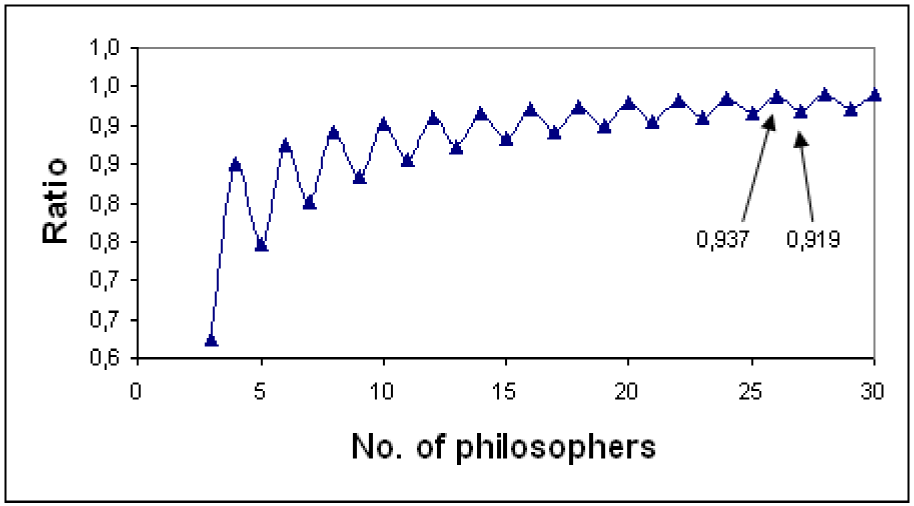
$\mathbf{w}(T1) = \mathbf{w}(T3) = 1$;

$\mathbf{w}(t2) = 0.001$

(in cont. model)

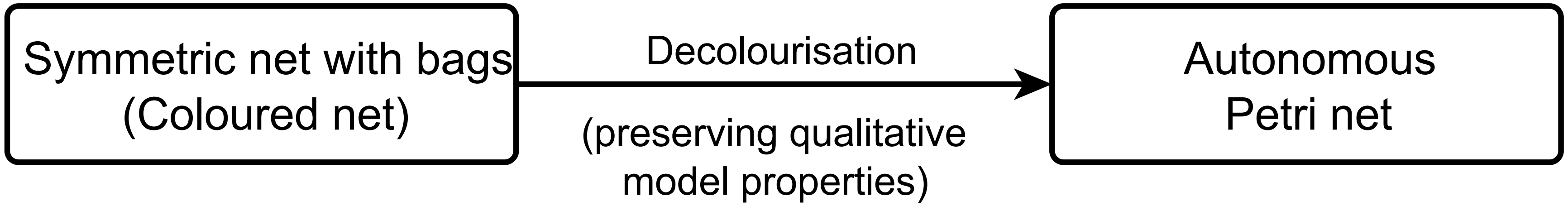
Discrete vs. continuous model: Throughput of T3 – difference and ratio





Contents

1. Decolourisation of autonomous nets: Basic idea



Limits of decolourisation based on net structure:

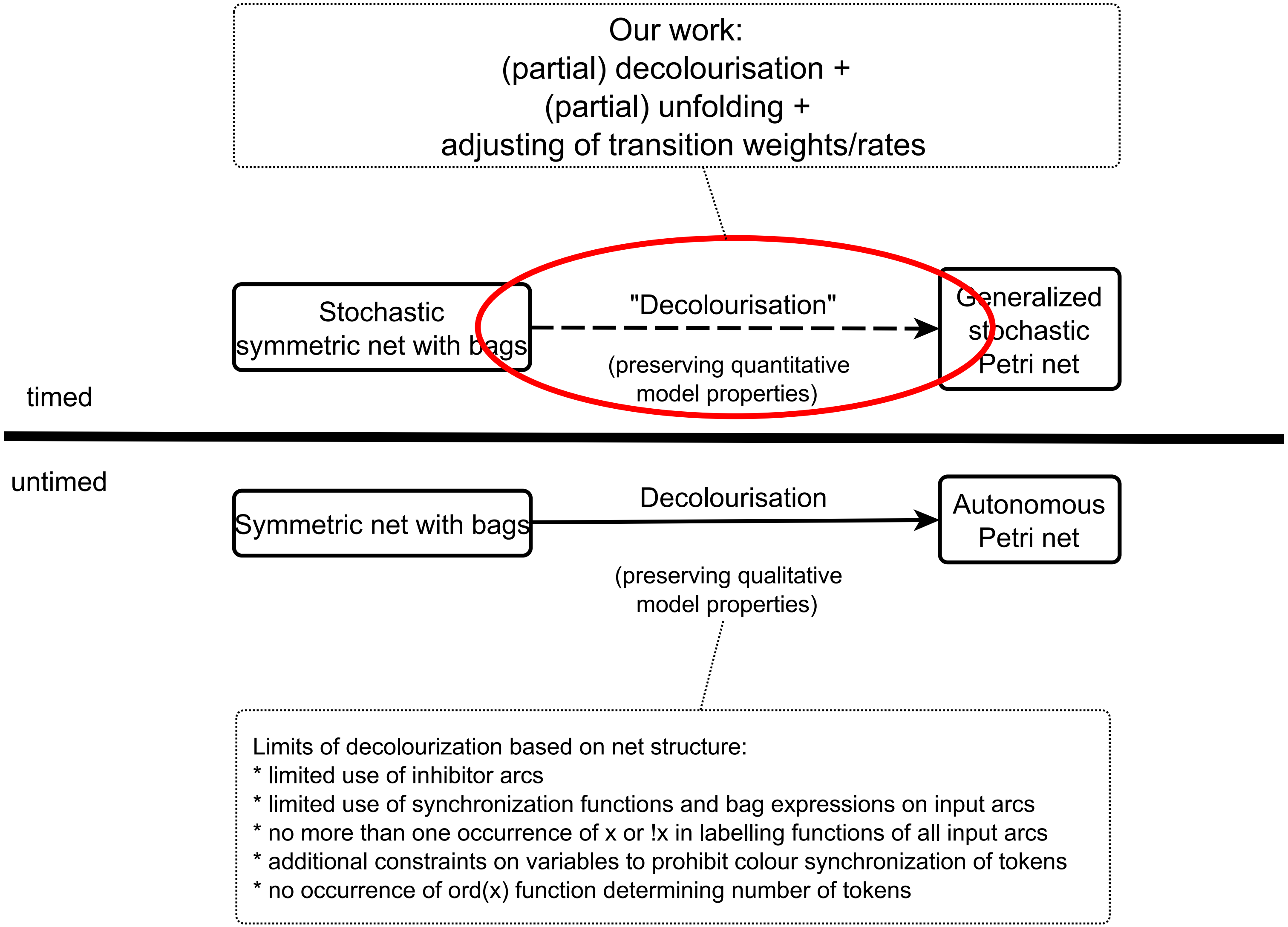
- limited use of inhibitor arcs
- limited use of synchronization functions and bag expressions on input arcs
- no more than one occurrence of x or !x in labelling functions of all input arcs
- additional constraints on variables to prohibit colour synchronization of tokens
- no occurrence of ord(x) function determining number of tokens

1. Decolourisation of autonomous nets: Previous work

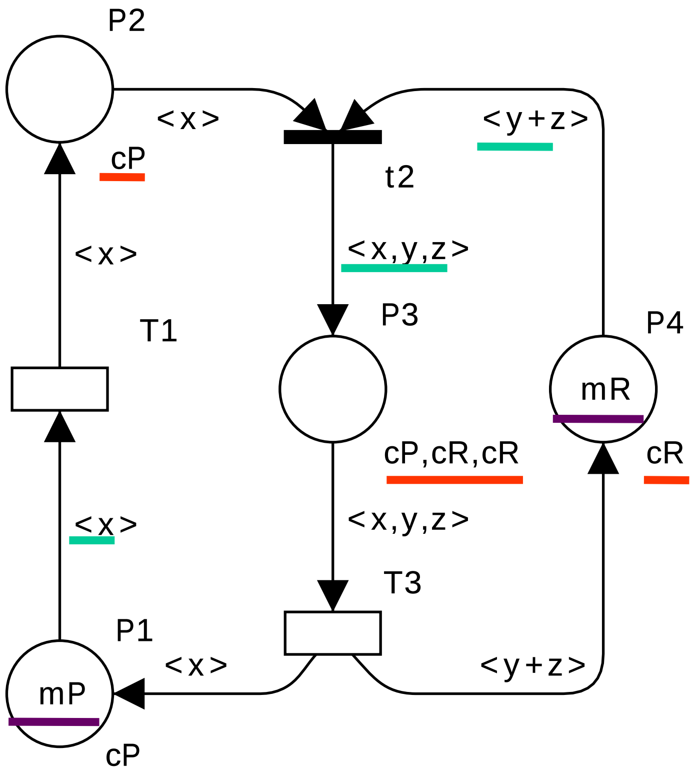
- Franceschinis – PhD thesis, 1993
- Chiola – Franceschinis, 1991
- Ajmone Marsan – Donatelli – Franceschinis – Neri, 1998
- Chiola – Dutheillet – Franceschinis – Haddad, 1991 & 1993
- Franceschinis – Ribaudo, 1996

- Decolourisation of symmetric nets (non-timed)
  - Based on reachability graph (behaviour)
  - **Based on net structure – this is what we look for!**
  - Timing issues mentioned partially in one paper
- Lumpability for stochastic symmetric net (timed)
  - Based on Symbolic Reachability Graph (SRG) is algorithmised
  - Here: we look for aggregation **at net level** (to keep the net structure) - it is **computationally more efficient**, but **less power in reduction**!

1. Decolourisation of autonomous nets: Basis for our work



1.1 SNBs: Dining philosophers with common res. (DinPhilCommon)



Symmetric Net with bags (SNB)

$\mathcal{N}$ = <P, T, **Pre**, **Post**, **Inh**, **pri**, Cl, $\mathcal{C}$, Φ)

Colour domains – from the set of basic colour classes Cl = *{cP, cR}*

Function defining colour domain

$\mathcal{C}(P3) = cP \times cR \times cR;$

$\mathcal{C}(t2) = \langle\langle x, y, z\rangle \in cP \times cR \times cR\rangle$

Arc functions

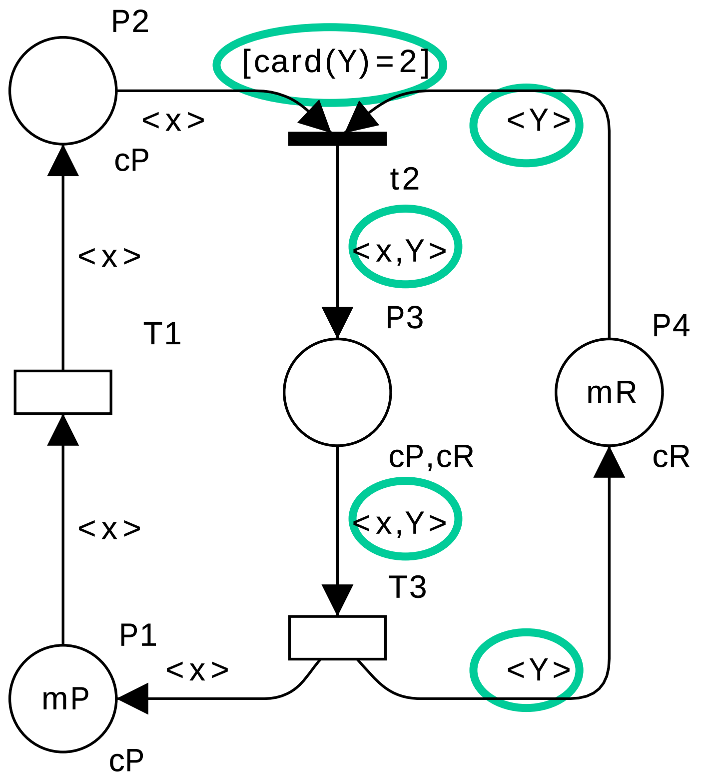
Initial marking: $mP = ph_1 \ldots ph_n$;

$mR = r_1 \ldots r_n$

Φ – mapping: guards on transitions

Inhibitor arcs (**Inh**) and priorities of transitions(**pri**) not used here

1.1 SNBs: DinPhilCommon with bags



- Resources are provided in a bag of 2 elements, not individually.
- Function *Y* represents a set – its cardinality is given in guard $\Phi(t2)$

1.1 Symmetric nets with bags: Relation to CPN

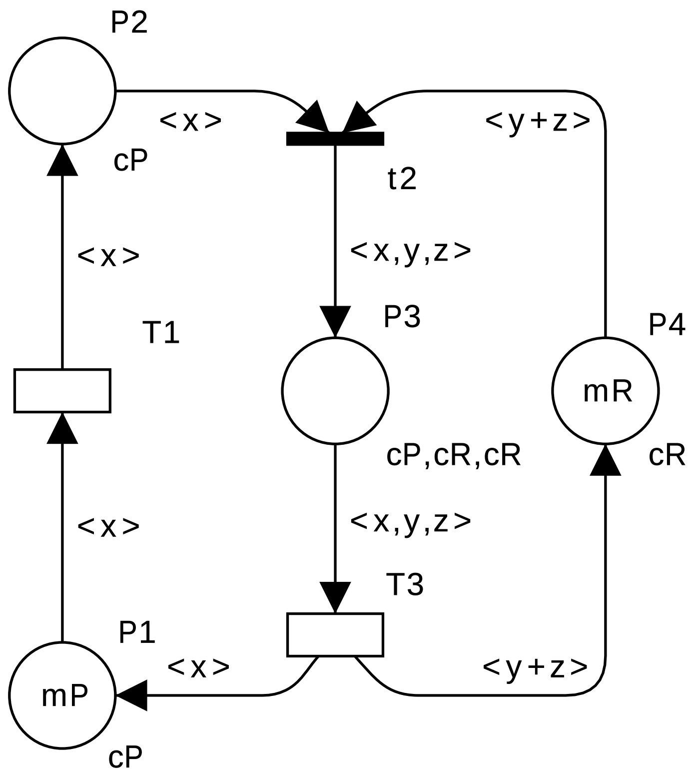
- Coloured Petri nets (CPN): tokens distinguished through colours
- Symmetric net
  - Has the same modelling power as CPN
  - Is subclass of CPN because it has **more strict definition** of **colour classes** (used in colour domains of places & transitions) and **colour functions** (in arc inscriptions & transition guards)
  - Colour classes and functions are written in more explicit (and parametric) form, using basic constructs of the formalism
- Symmetric net with bags
  - In addition to CPN: manipulation with bags of tokens

1.2 Decolourisation procedure of SNB: DinPhilCommon

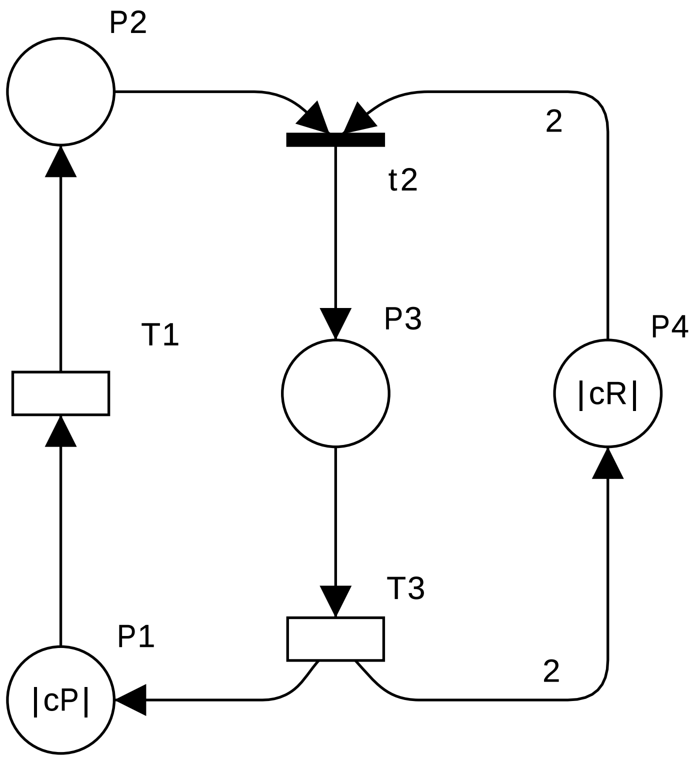
- **Flow 1** (philosophers):
  - *P*1, *P*2, *P*3, *T*1, *t*2, *T*3
  - colour domain *cP*, variable *x*
- **Colour shrinking function 1**:
  $cP \rightarrow cP'$: $\forall\ c \in cP$: $sh(c) = \bullet$

- **Flow 2** (resources):
  - *P*3, *P*4, *t*2, *T*3
  - colour domain *cR*, variables *y* and *z*
- **Colour shrinking function 2**:
  $cR \rightarrow cR'$: $\forall\ c \in cR$: $sh(c) = \bullet$

- Intuitively: It is not necessary
  to distinguish philosophers, nor resources

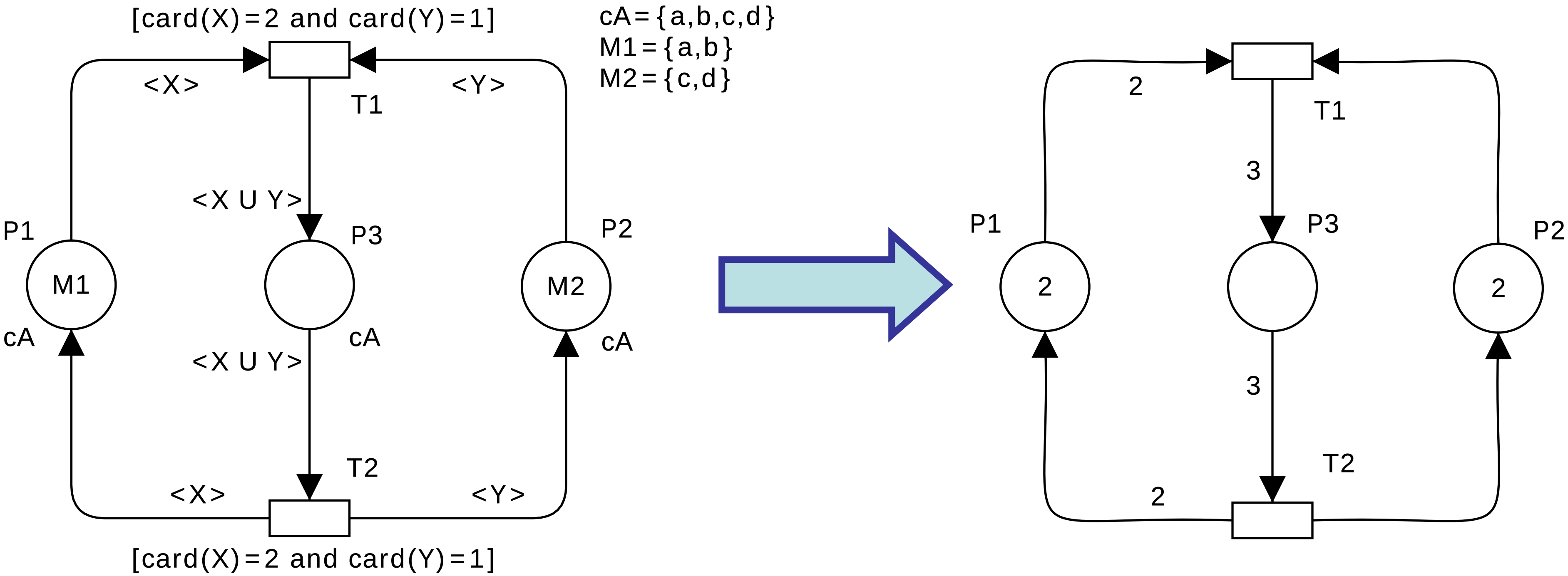


1.2 Decol. procedure of SNB: DinPhilCommon – decolourised net



- Modified version of Dining philosophers with common resources can be completely decolourised.
- Populations are created.

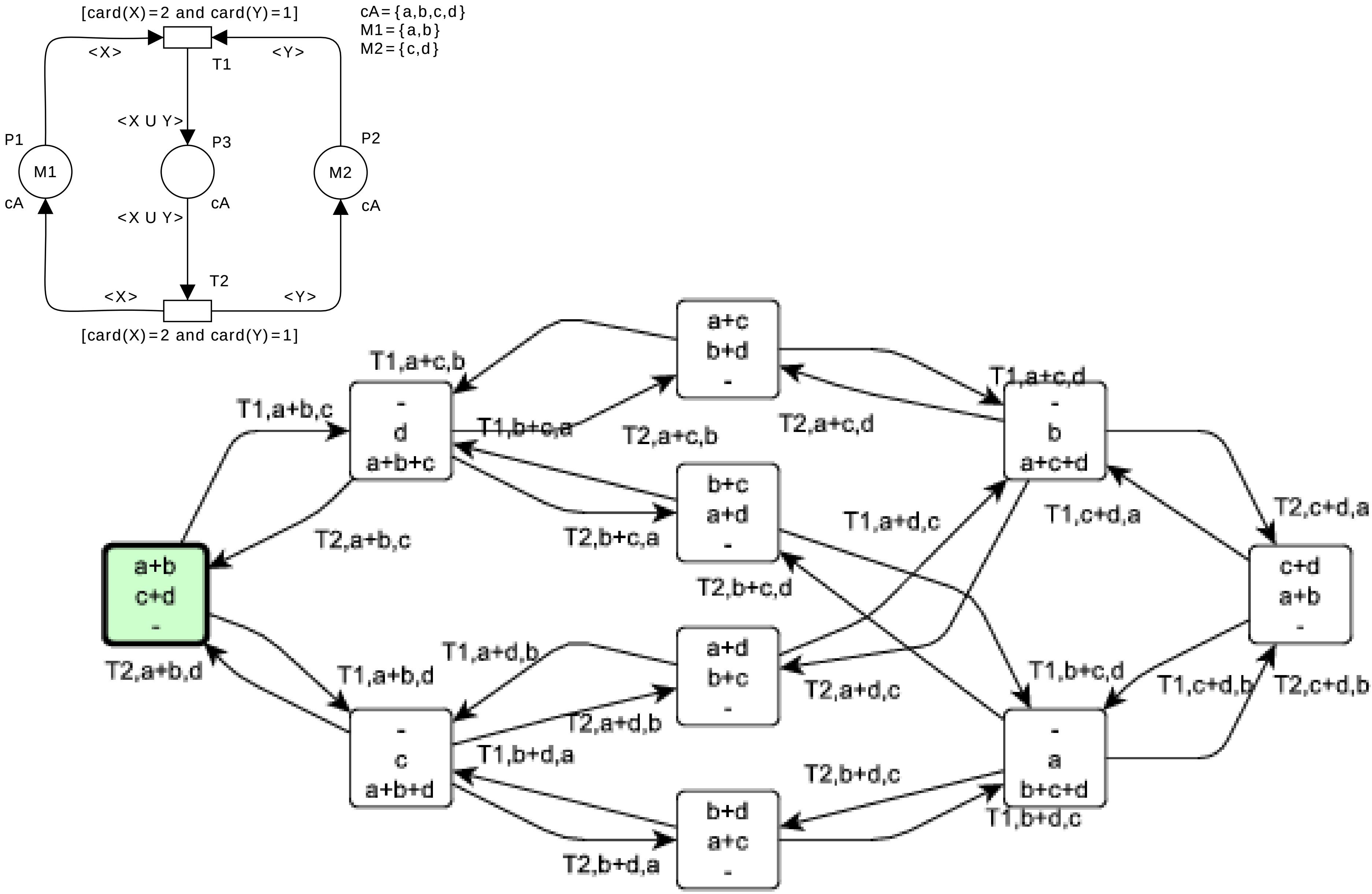
1.3 Decolourisation of bags: Union



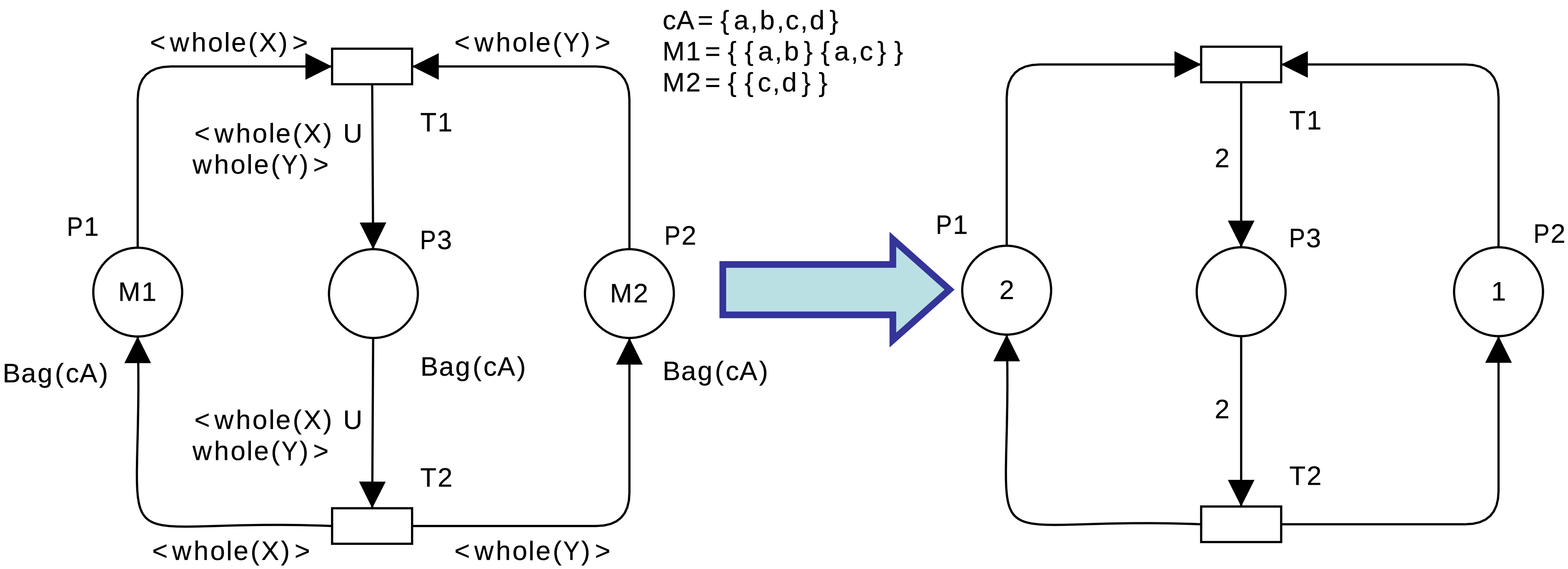
Three tokens introduced in *P*3 are equal ⇒ *T*2 has 3 instances in SNB

Bags ***X*** and ***Y*** have **prescribed** cardinalities ⇒
model can be decolourised.

1.3 Decolourisation of bags: Union – RG



1.3 Decolourisation of bags: Bags as wholes

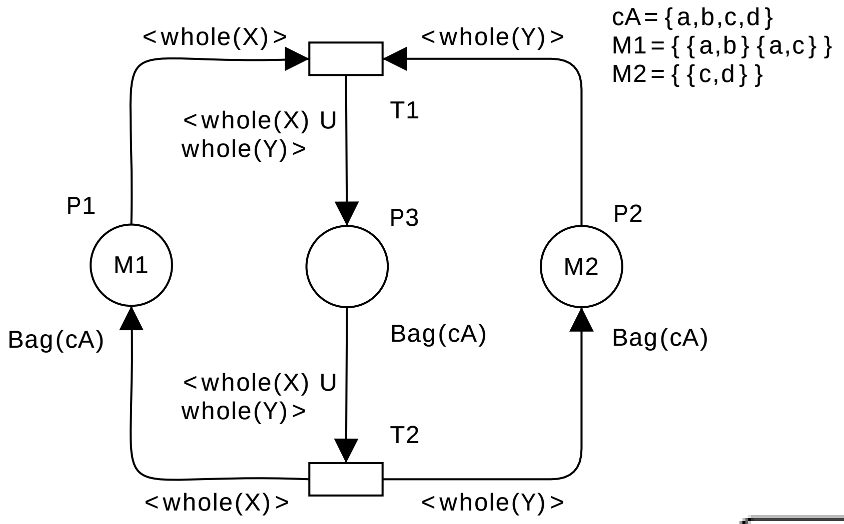


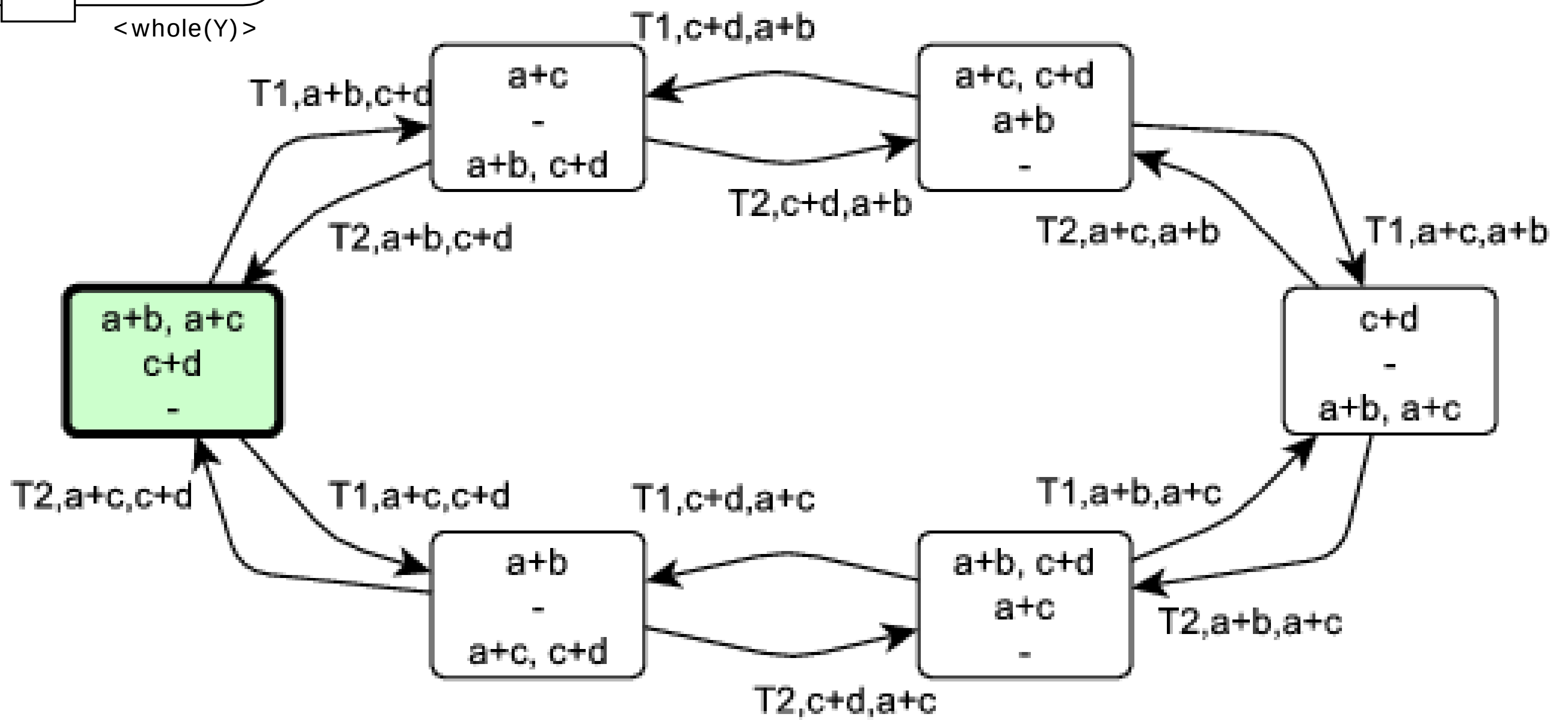
Every bag stays unchanged ⇒ **substitution**, e.g.:

k = {*a*,*b*}, *l* = {*a*,*c*}, *m* = {*c*,*d*}, *cB* = {*k*, *l*, *m*}

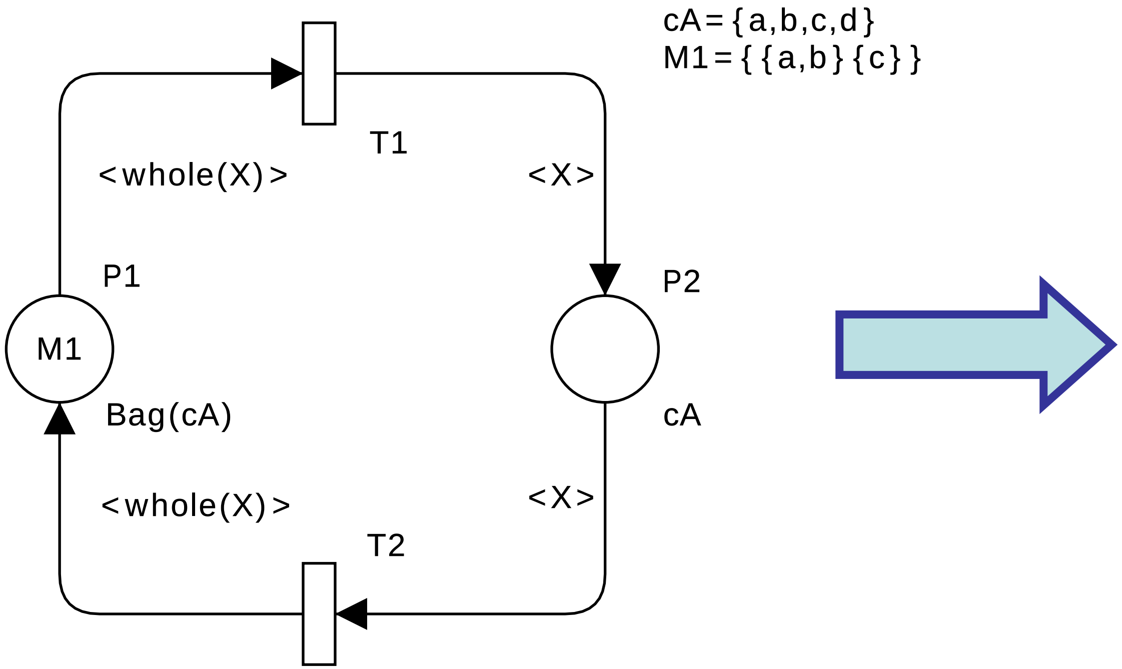
and the model can be decolourised like SN without bags

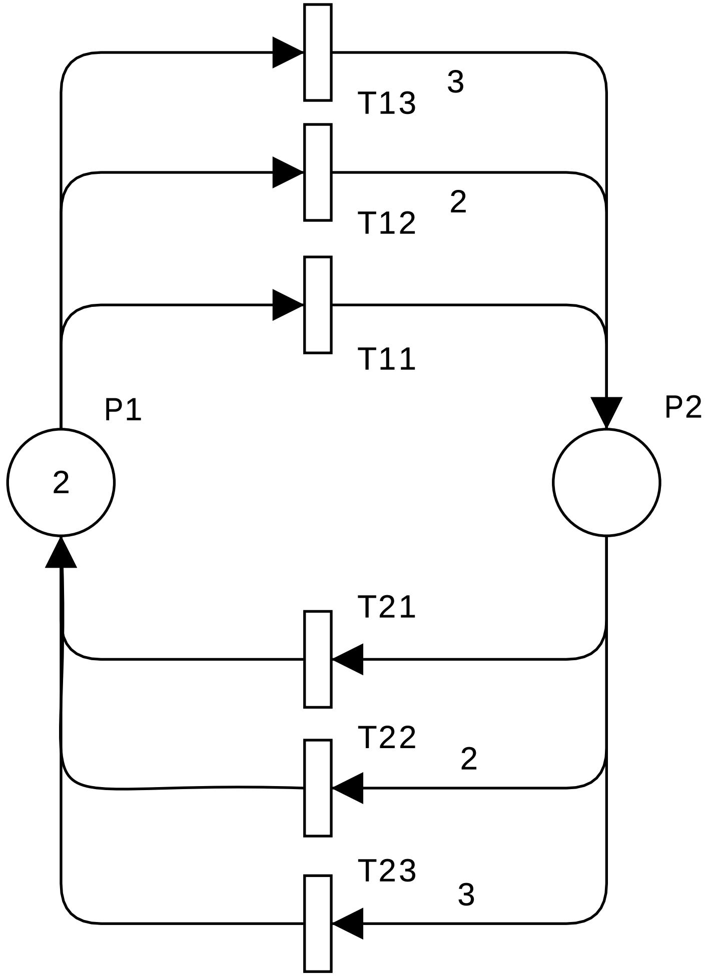
1.3 Decolourisation of bags: Bags as wholes – RG





1.3 Decolourisation of bags: Bags and elements

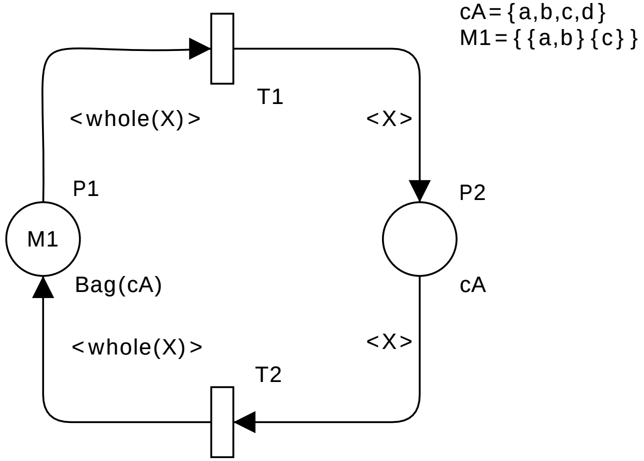


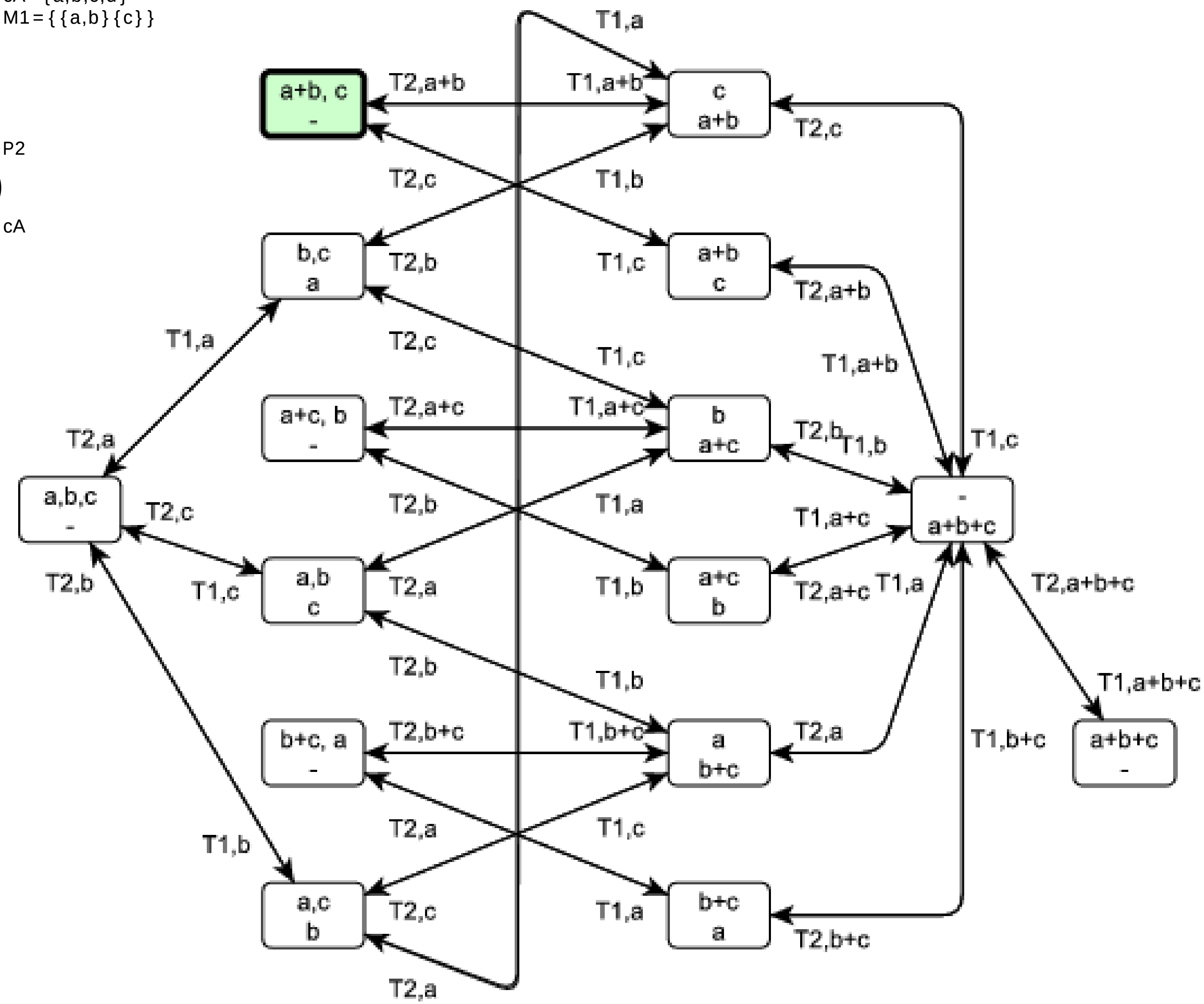


Bags as wholes (<whole(***X***)>) and groups of elements (***X***) , where size of *X* is **not determined.**

Net must be **bag-unfolded** first (*T*1 to *T*11, *T*12, *T*13, etc.) and then it **can be decolourised**.

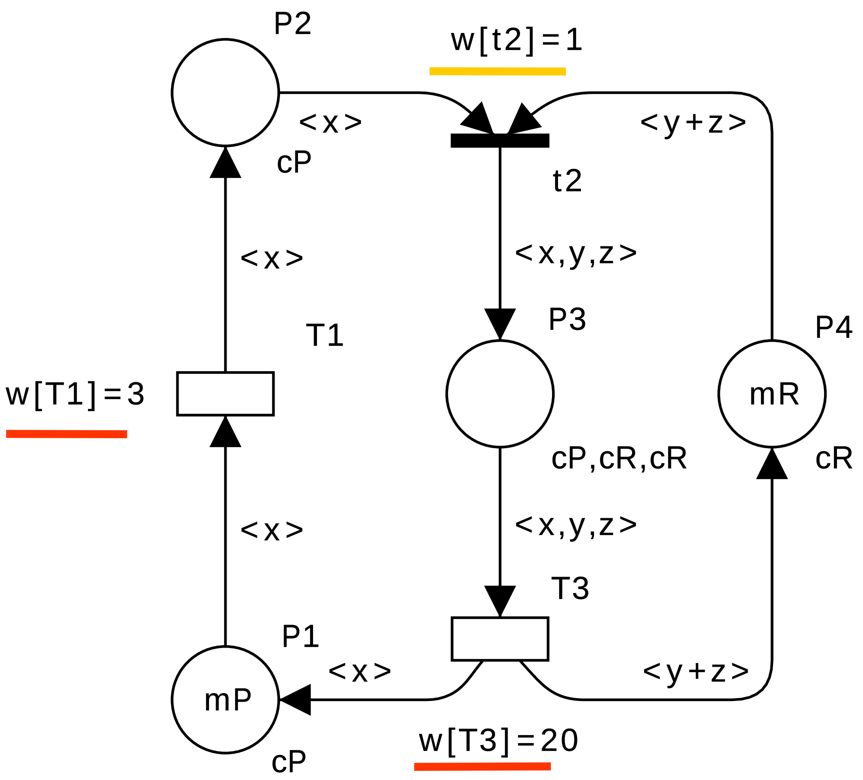
1.3 Decolourisation of bags: Bags and elements – RG





Contents

2.1 Stochastic SNBs: DinPhilCommon as an example



- adding
  - firing rates (timed t.)
  - weights (immediate t.)

- $\mathbf{w}(t2) = \sum \mathbf{w}(<t2, ph_i, r_j, r_k>)$, j<>k

  $<t2, ph_i, r_j, r_k>$ - transition instance of *t2* with colours of ph_i, r_j and r_k
  - For every *i*, there are $\mathbf{m}(P4)$. $(\mathbf{m}(P4) - 1)$ instances – philosopher *i* is deciding which couple of resources to pick up
  - since all variations of resources have equal chances, then

$$\mathbf{w}(<t2, ph_i, r_j, r_k>) = \mathbf{w}(t2) / (\mathbf{m}(P2) \,.\, \mathbf{m}(P4) \,.\, (\mathbf{m}(P4) - 1))$$

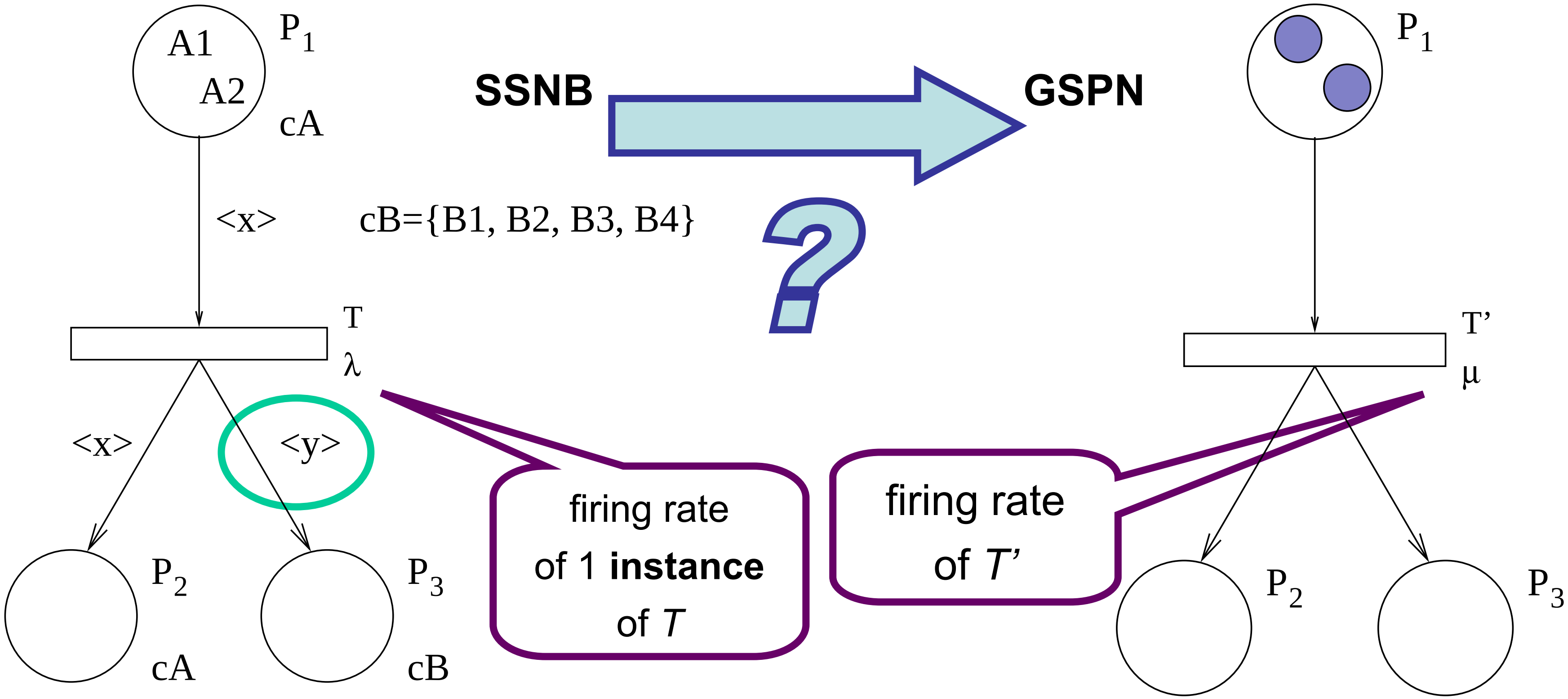
2.2.1 Decolourisation of SSNB: On used terminology

- *Extended conflict set (ECS)*:
  set of transitions that are in transitive closure of conflict relation (equivalence classes)
- *Colour-safe place*:
  in all possible markings, it contains at most one instance from each colour:
  - {A1, A2} – allowed
  - {A1, A1, A2} – not allowed

2.2.1 Decolourisation of SSNB: Overview of our approach

- Net transformation rules:
  we look for patterns not at behavioural level (symbolic Markov chain), but only on structural level: **from net to net**
- Steps of decolourisation procedure of SSNB:
  1. As autonomous net:
     a) Decolourisation of the net as SNB
     b) Where necessary: **unfolding** of colours or bags
     – it usually brings problems: population ↓, net size ↑
  2. As timed net:
     – **Adjusting** transition **firing rates / weights** according to existing extended conflict sets (ECS) for **transition instances** in the SSNB so that rates of underlying CTMC (Continuous-time Markov chain) stay preserved
- By default, we assume:
  – Infinite server semantics (ISS)
  – Bounded nets

2.2.2 TPA rule 1 – New variable on output (1)

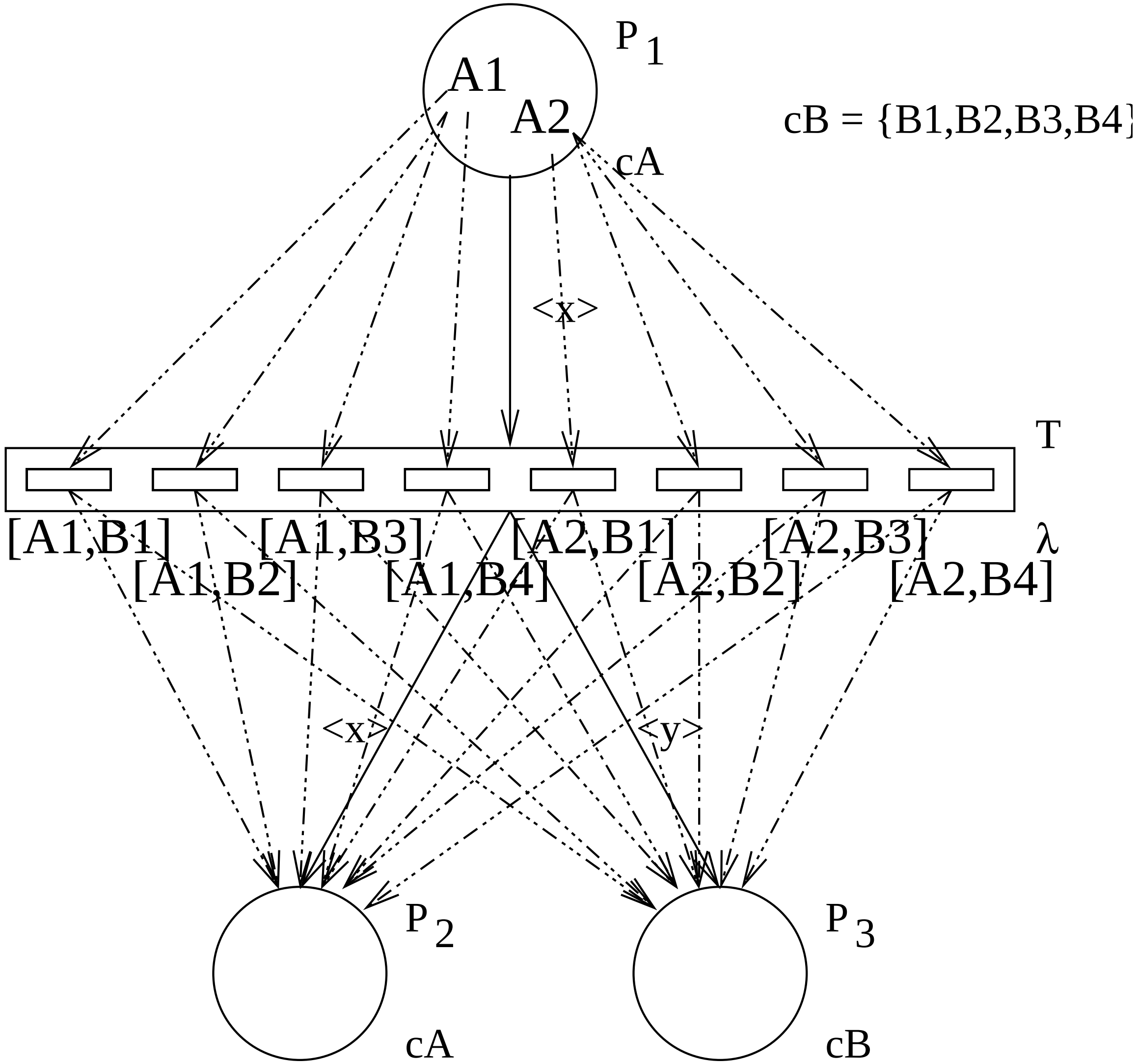


Variable *y* is not present on input, but on output arc only

It represents tokens from colour set *cB* with 4 possible values

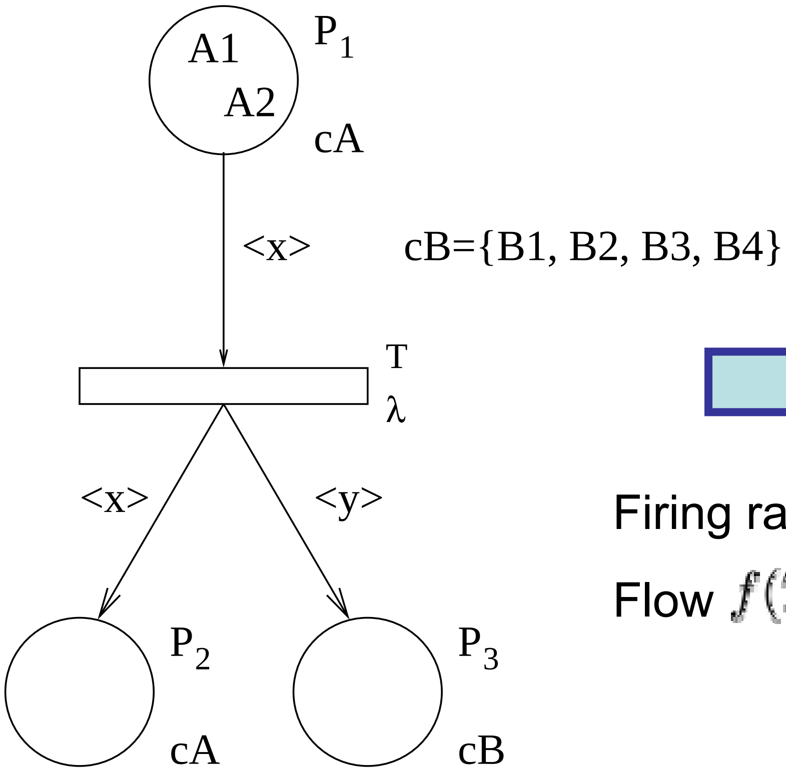
How does the firing rate of *T* change by decolourisation?

2.2.2 TPA rule 1 – New variable on output (2)



There are 8 transition instances for all combinations between 2 tokens in P_1 and 4 potential values of variable *y*.

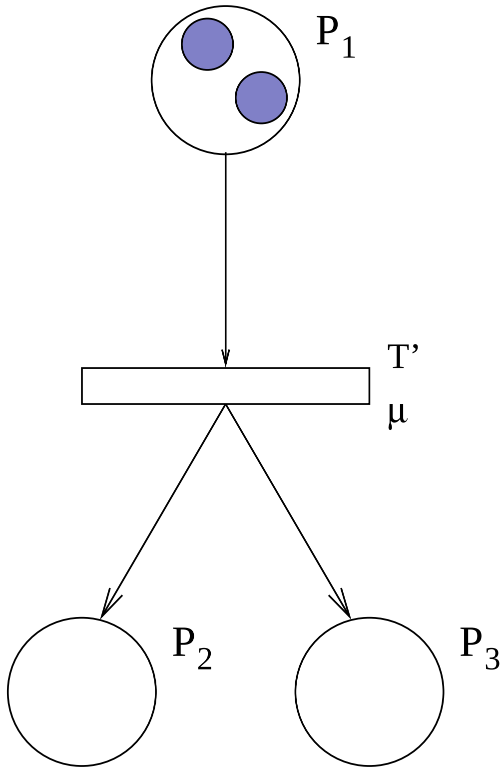
2.2.2 TPA rule 1 – New variable on output (3)



Firing rate $\mu = \lambda.|cB|$

Flow $f(T') = \mu.\mathrm{m}(P_1)$

Infinite SS

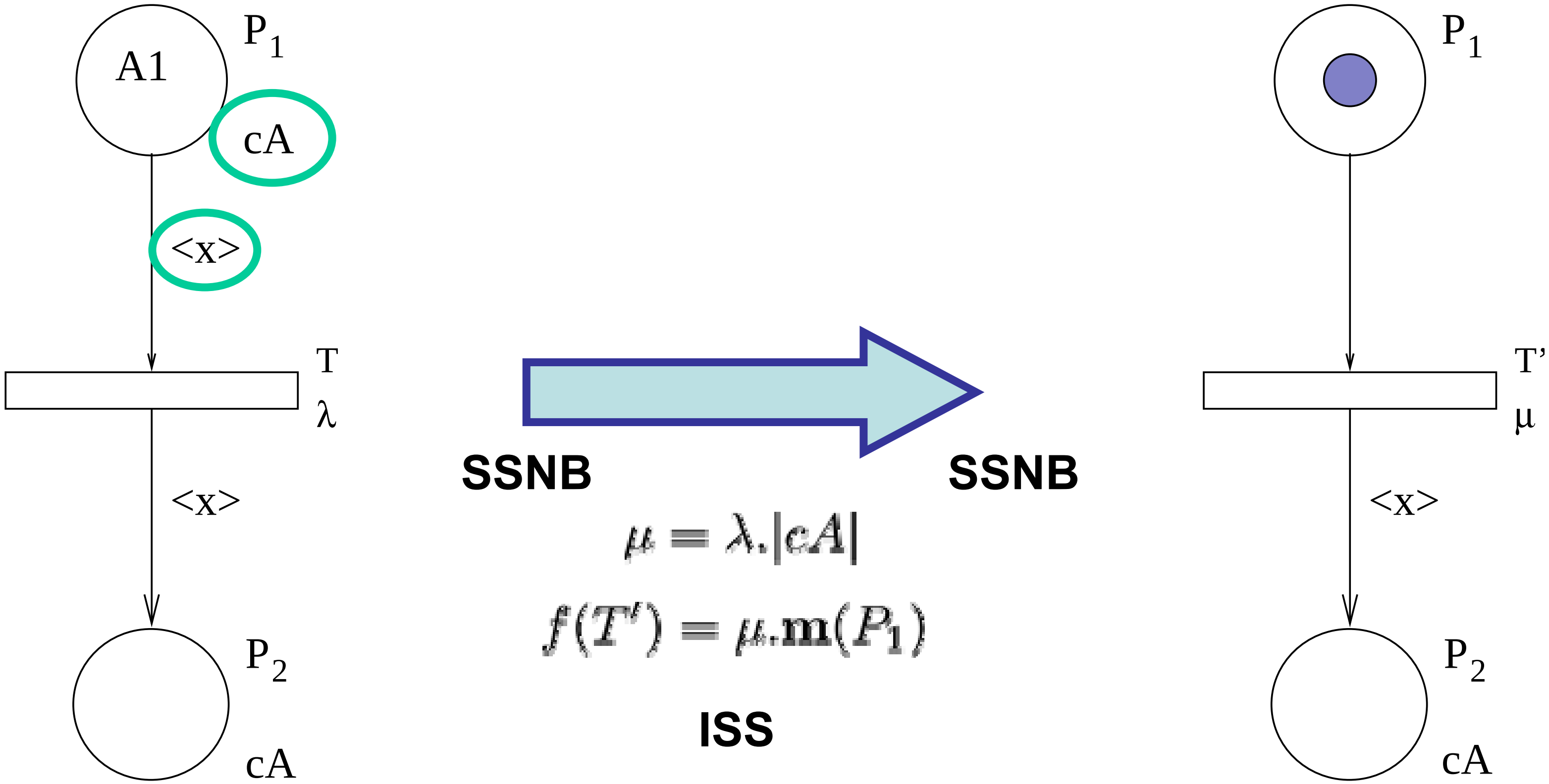


T in coloured model: 8 transition instances ⇒ firing rate by ISS: **8*λ***

T' in non-coloured model: enabling degree by ISS is 2 ⇒ **2*μ***

Difference: **|*cB*| = 4** … necessary multiplication: ***μ* = *λ*.4**

2.2.2 TPA rule 1bis – Decolourisation of input only

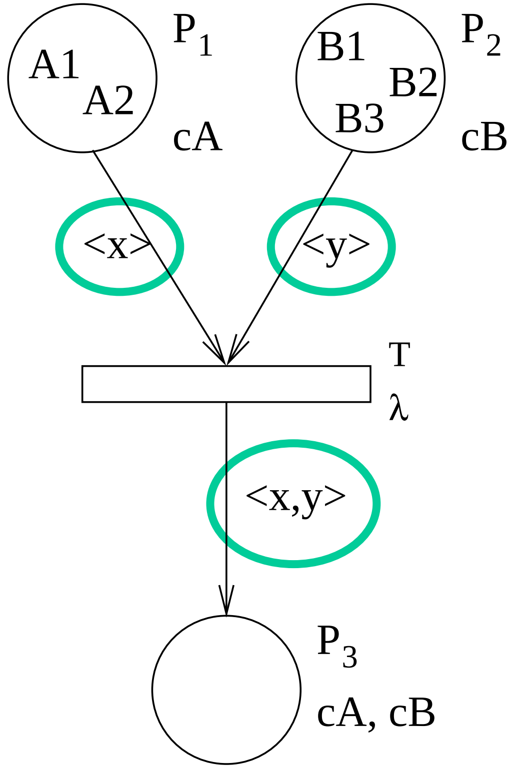


Only input place and arc are decolourised

value on output arc: not determined any more, but **random**

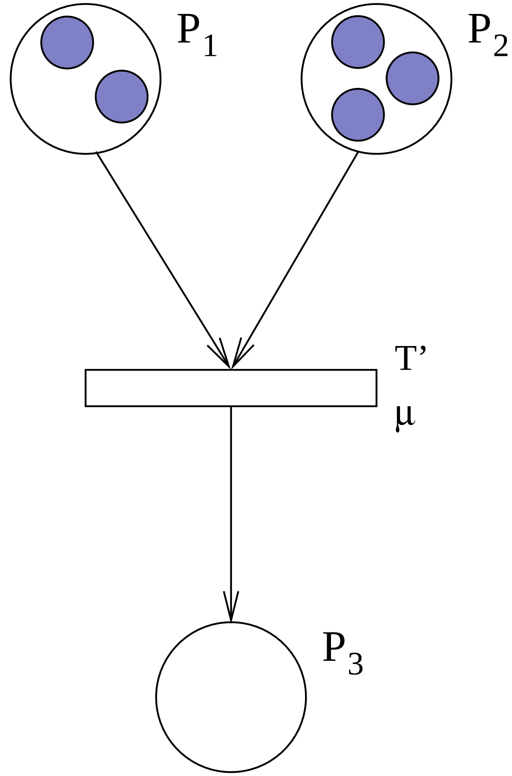
T': number of transition instances changes from 1 to **|*cA*|**

2.2.2 TPA rule 2 – Multiple input places



$$f(T') = \lambda.\mathbf{m}(P_1).\mathbf{m}(P_2)$$

Not ISS



T: tuple <*x*,*y*> on output composed from *x* and *y* on input

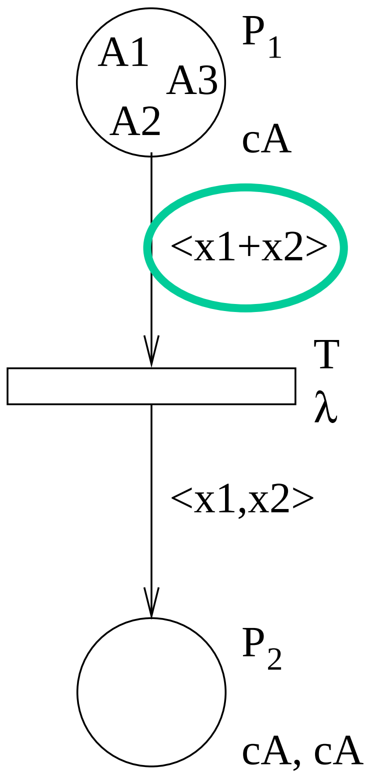
No. of transition instances: **product** of current marking of input places

T': **marking-dependent** flow containing the product

Products: – frequent in population dynamics (foxes, rabbits)

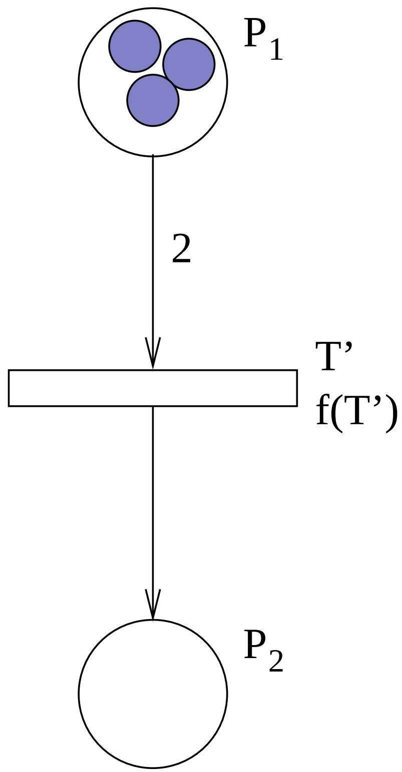
– clear non-linearity (≠ minimum operator)

2.2.2 TPA rule 3 – Addition on input arc



$$f(T') = \lambda.\mathbf{m}(P_1).(\mathbf{m}(P_1) - 1)$$

Not ISS

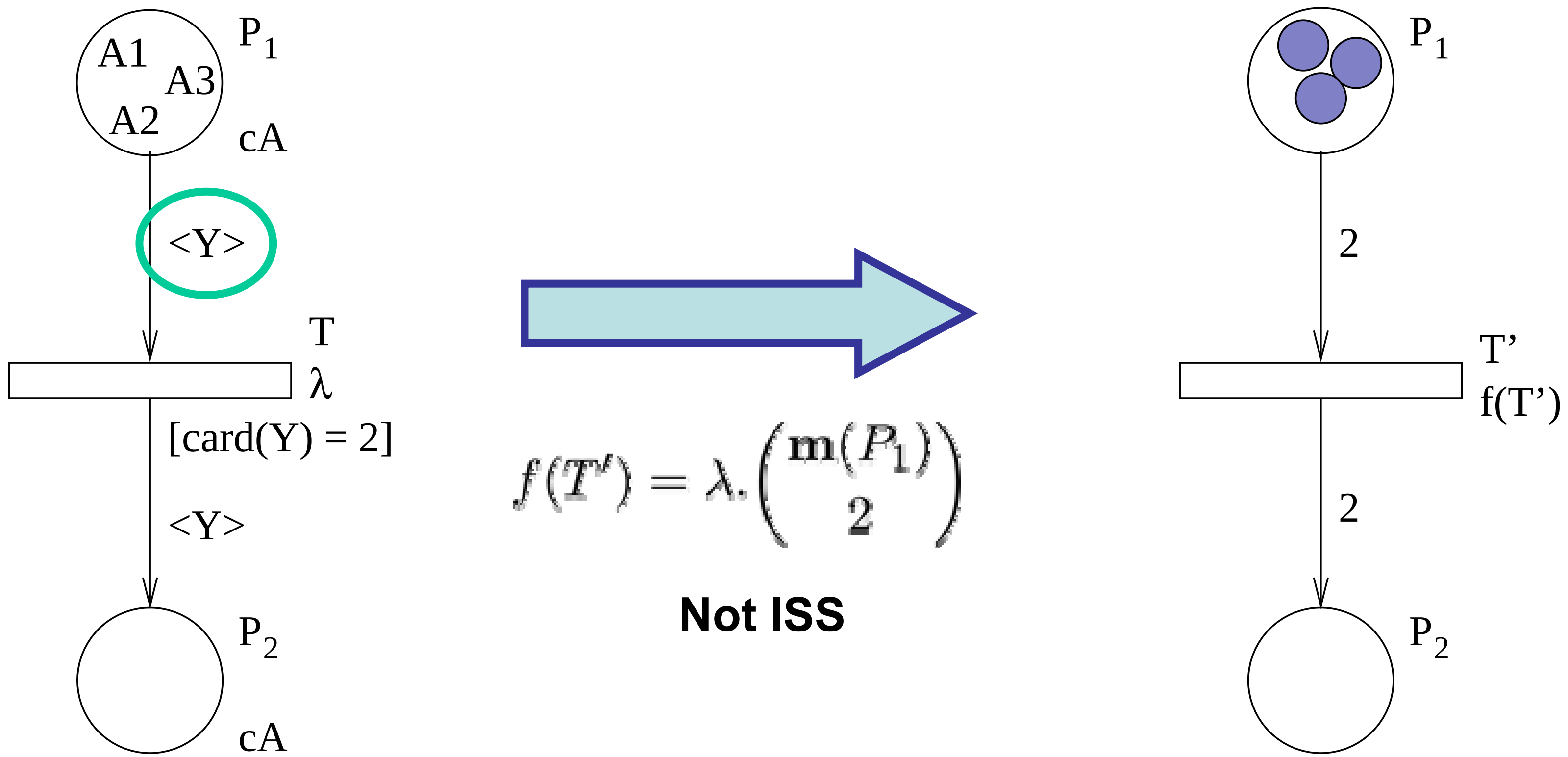


<*x*1+*x*2>: variations from current marking,

*x*1 = *A*1 and *x*2 = *A*2 **is different from** *x*1 = *A*2 and *x*2 = *A*1.

Result: **marking-dependent** firing rate of *T'*.

2.2.2 TPA rule 4 – Bag on input arc



$$f(T') = \lambda . \binom{\mathbf{m}(P_1)}{2}$$

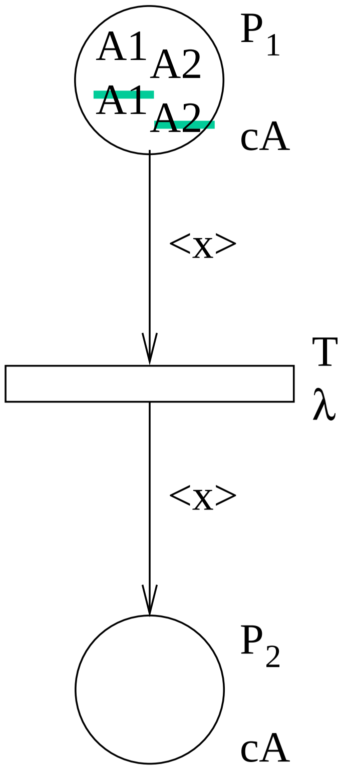
Not ISS

<*Y*>: combinations from current marking,

the order of colours in the bag is not important.

Result: **marking-dependent** firing rate of *T'*.

2.2.2 TPA rule 5 – Non-colour-safe input place

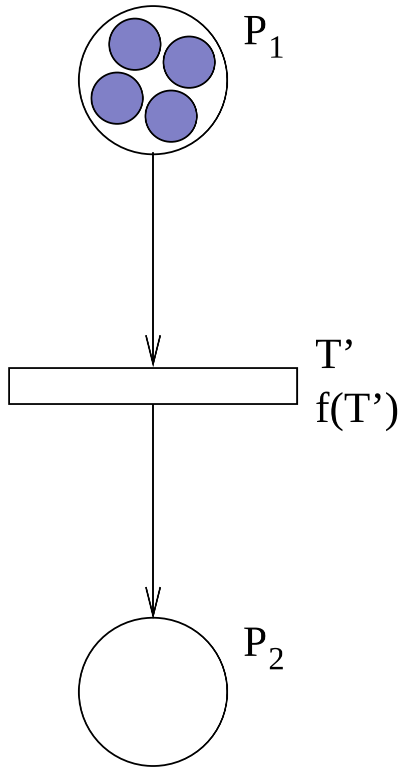


$$f(T') = \lambda.\mathbf{m}'(P_1)$$

$$\mathbf{m}'(P_i) = \sum_{j=1}^{|cd(P_i)|} l_j$$

$$l_j = \begin{cases} 1 & \text{if } |c_j| \geq 1 \\ 0 & \text{if } |c_j| < 1 \end{cases}$$

Not ISS



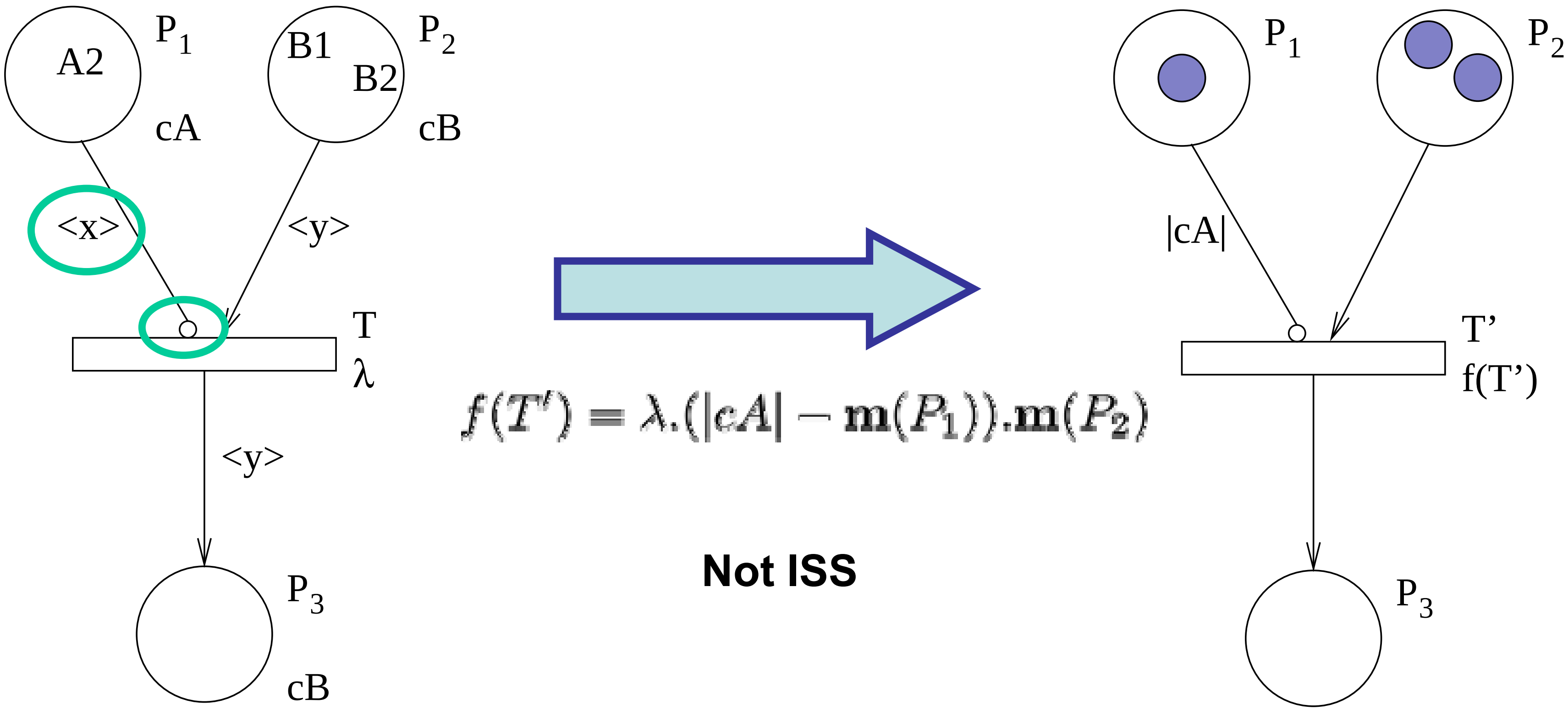
Place contains several tokens of the same value.

Transition instances: **one token per colour** is considered for enabling.

Result: **marking-dependent** flow **from unique tokens**.

2.2.2 TPA rule 6 – Inhibitor arc

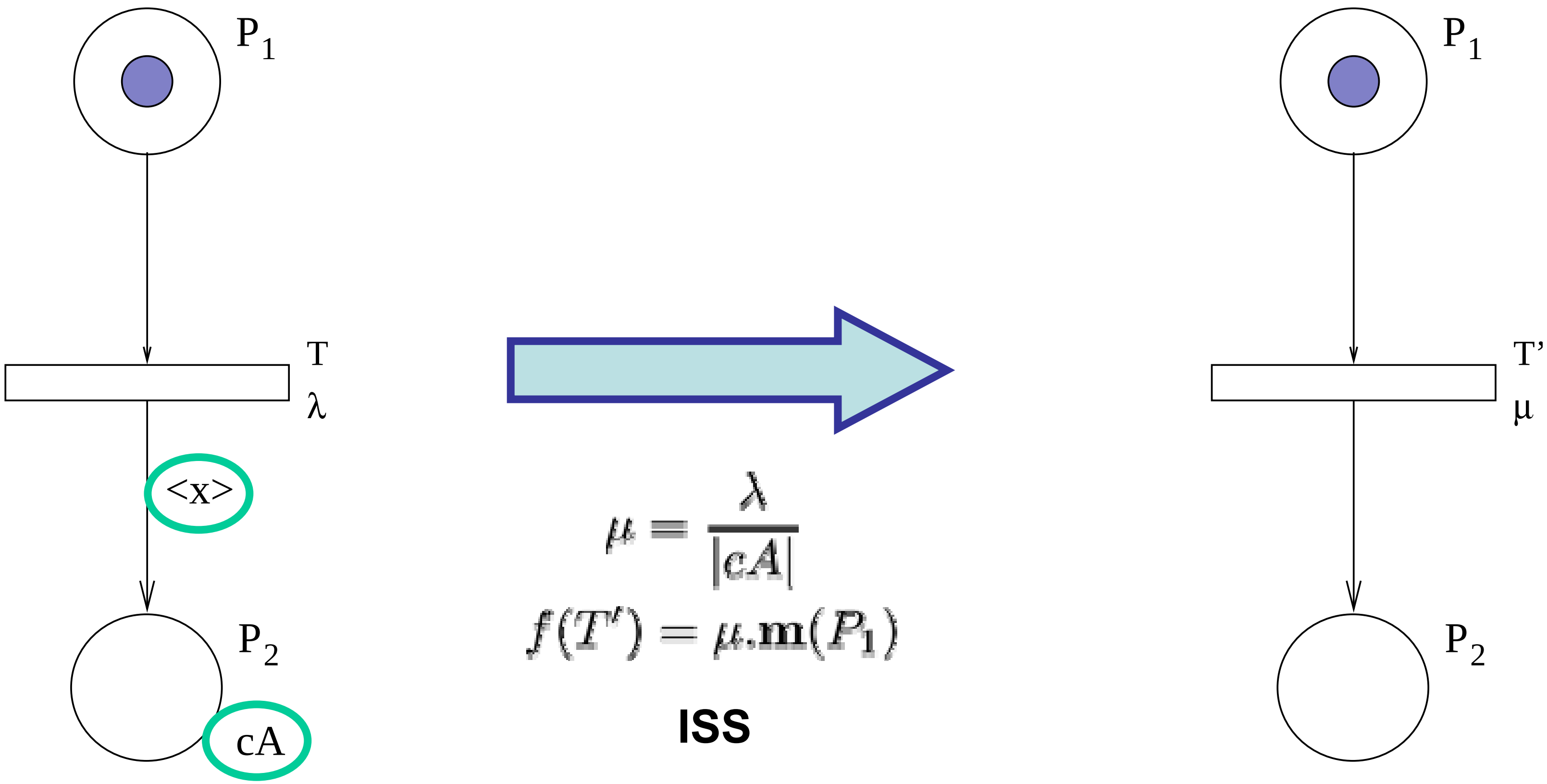
cA={A1, A2, A3}



<*x*> on inhibitor arc: number of absent colours considered.

Result: **marking-dependent** firing rate of *T’*.

2.2.2 TPA rule 7 – Decolourisation of output only



$$\mu = \frac{\lambda}{|cA|}$$

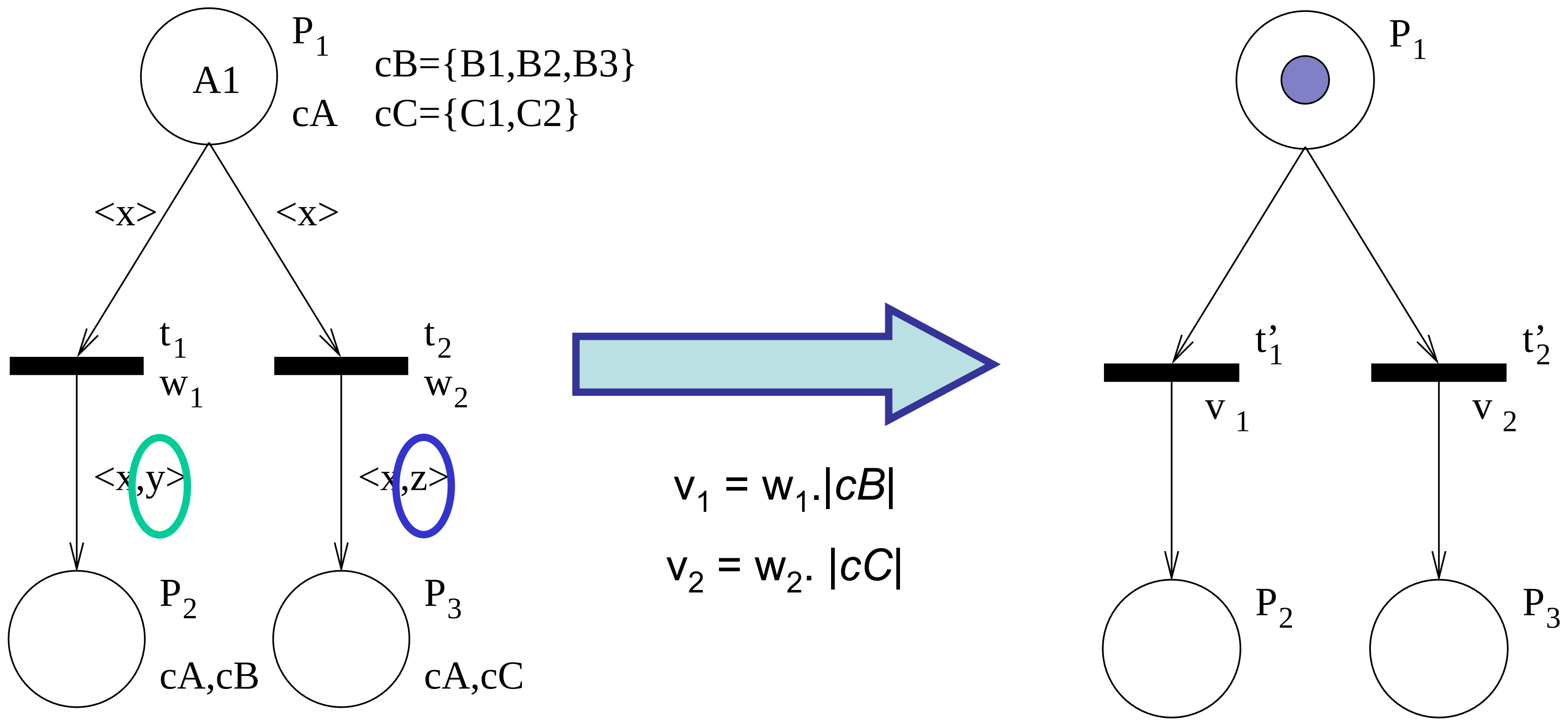
$$f(T') = \mu.\mathbf{m}(P_1)$$

ISS

Only output place and arc are decolourised in a partially decolourised net

T’: number of transition instances changes from |*cA*| to **1**

2.2.2 TPA rule 8 – Free-choice conflict with new variables



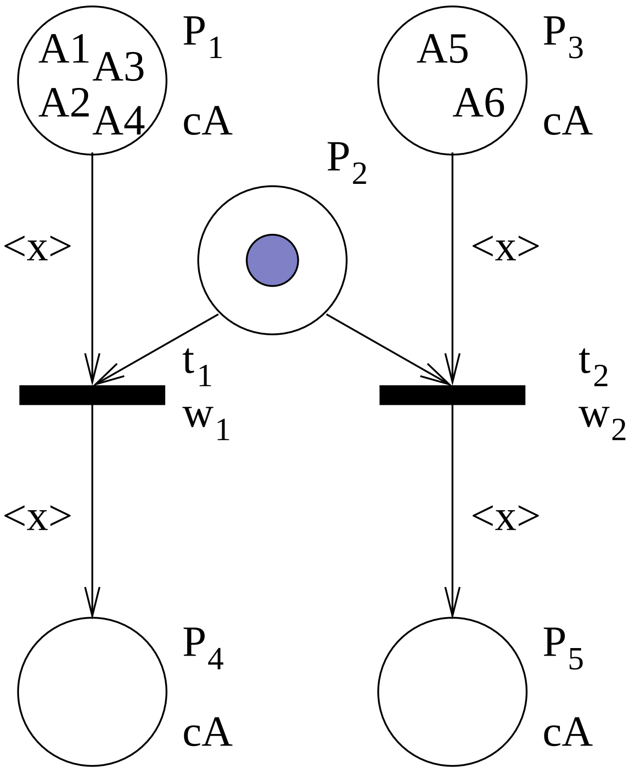
Coloured model: 3 instances of t_1 and 2 instances of t_2

$\Rightarrow \pi(t_1) = 3/2.(w_1/w_2).\ \pi(t_2)$

Non-coloured model: 1 t. i. of each transition $\Rightarrow \pi(t_1) = (v_1/v_2).\ \pi(t_2)$

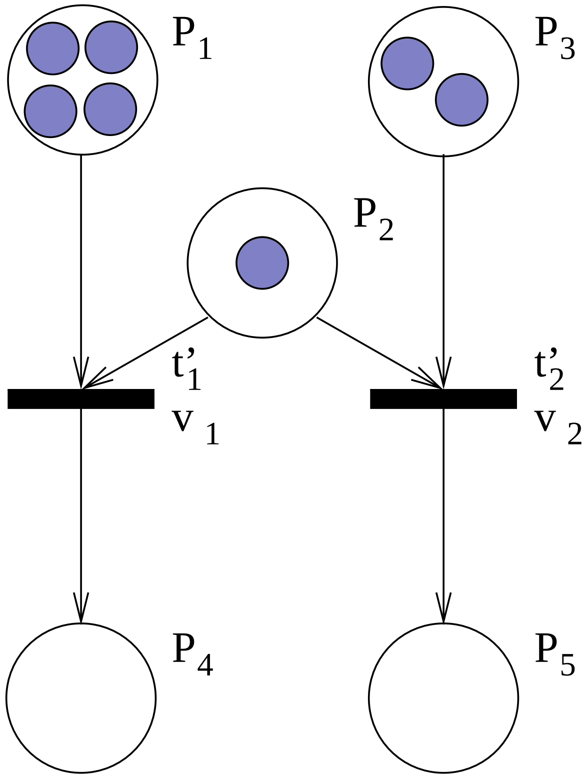
Result: **weight (firing rate) adjusted & fixed** for all markings

2.2.2 TPA rule 9 – Non-free-choice conflict



$v_1 = w_1 . \mathbf{m}(P_1)$

$v_2 = w_2 . \mathbf{m}(P_3)$



Analogous to previous case,

just the **weights/rates depend on current marking** here

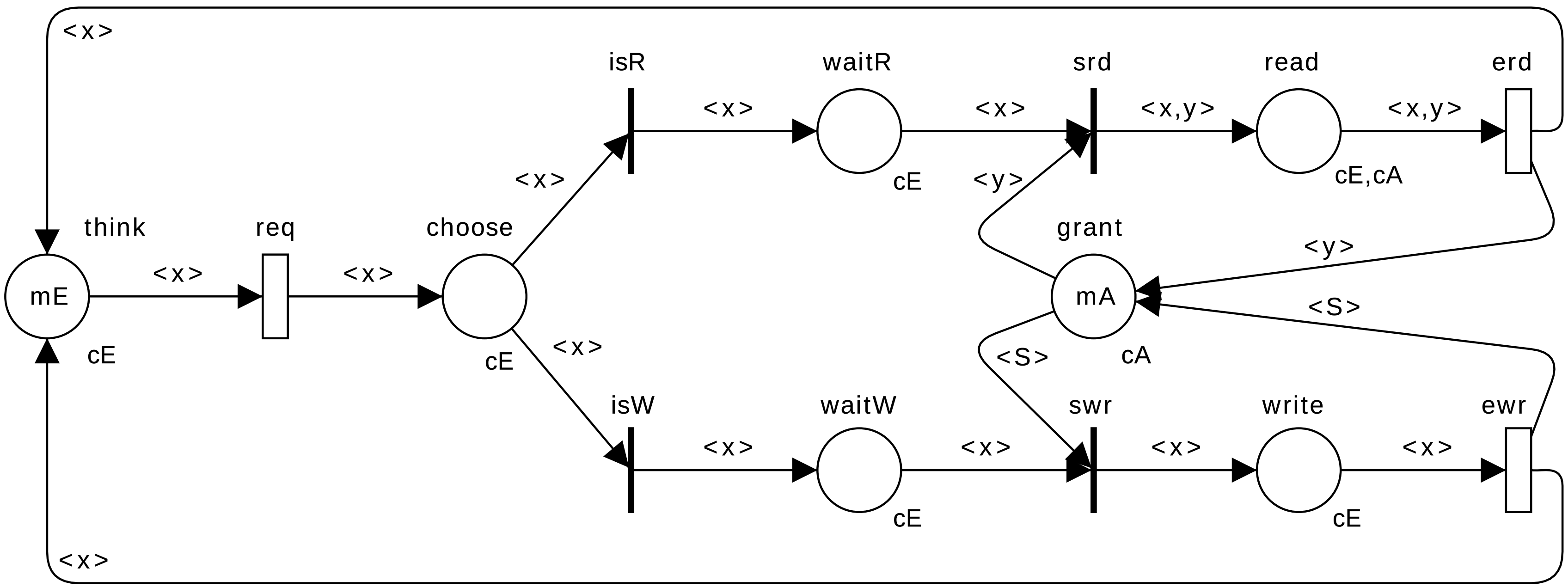
2.2.2 TPA rules: Summary

- Transition parameter adjustment rules for modification of
  - Firing rates of timed transitions:
    1) New variable on output

    1bis) Decolourisation of input only

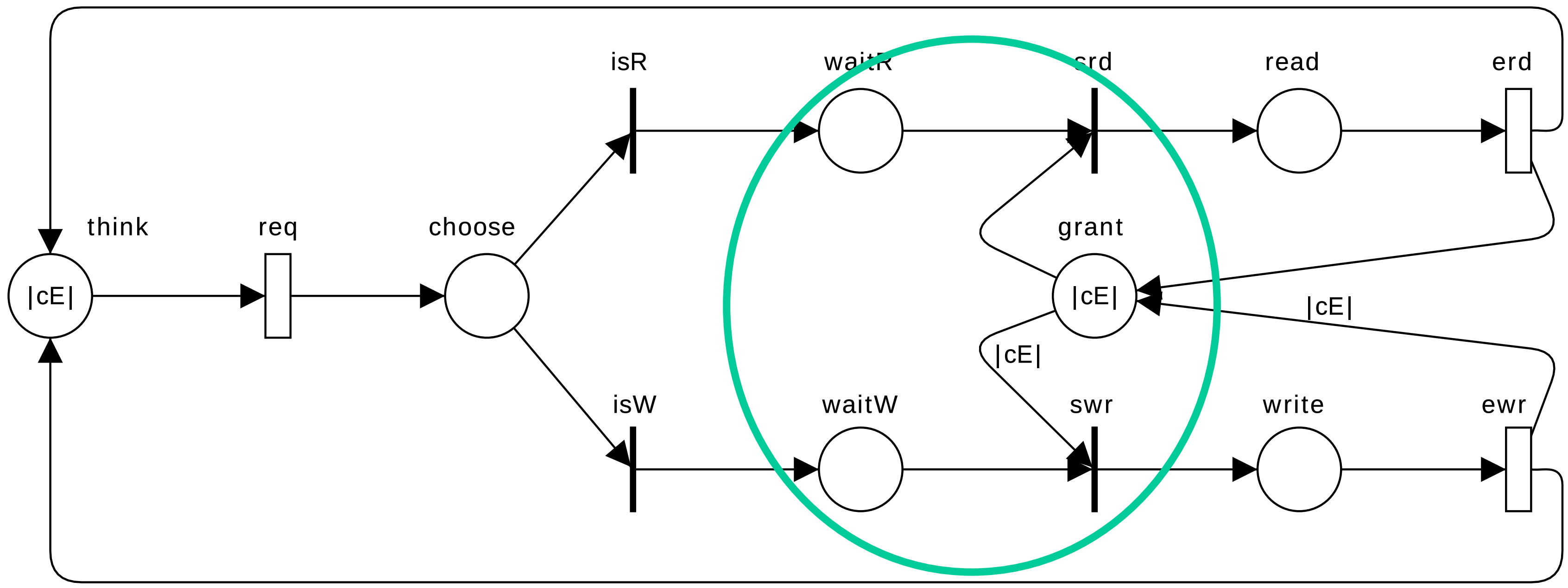
    2) Multiple input places
    3) Non-colour-safe input place
    4) Addition on input arc
    5) Bag on input arc
    6) Inhibitor arc
    7) Decolourisation of output only
  - Weights of immediate transitions or firing rates of timed transitions:
    8) Free-choice conflict with new variables
    9) Non-free-choice conflict
- Completeness? **– No! Rules ≈ tools**.

2.3 SSNB decolourisation examples: Concurrent Readers – Exclusive Writers (CREW)



- *E* entities access a shared space for reading concurrently (*read*) or writing exclusively (*write*)
- Access is granted (*srd*, *swr*) by access tokens (*grant*). Writing needs all of them (<*S*>), reading just one (<*y*>).
- If they are not available, entities wait (*waitR*, *waitW*).
- $|cA| = |cE|$

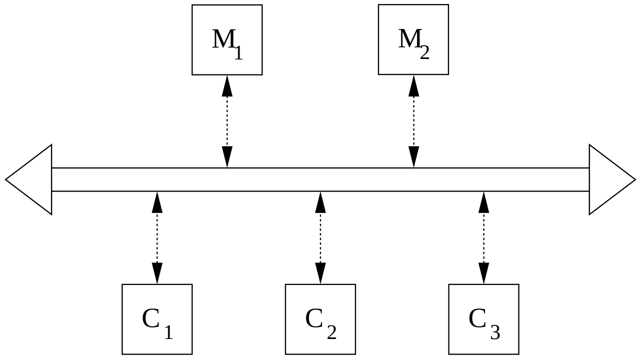
2.3 SSNB decol. examples: CREW – decolourized net



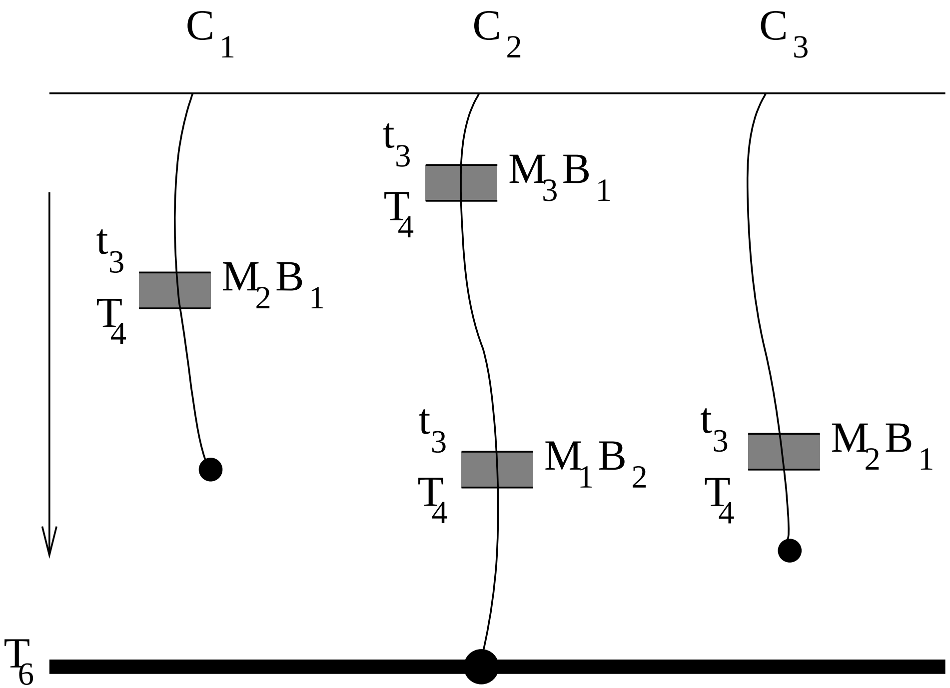
- This model can be completely decolourized - populations of entities and access tokens created
- Timing: **non-free-choice conflict of immed. transitions** (TPAR 9):
  - Conflict between *srd* and *swr* after firing of *ewr*
  - Their firing rates dependent of marking of their input places (*waitR*, *waitW*)

2.3 SSNB decolourisation examples:

Multi-computer programmable logic controller (MCPLC)



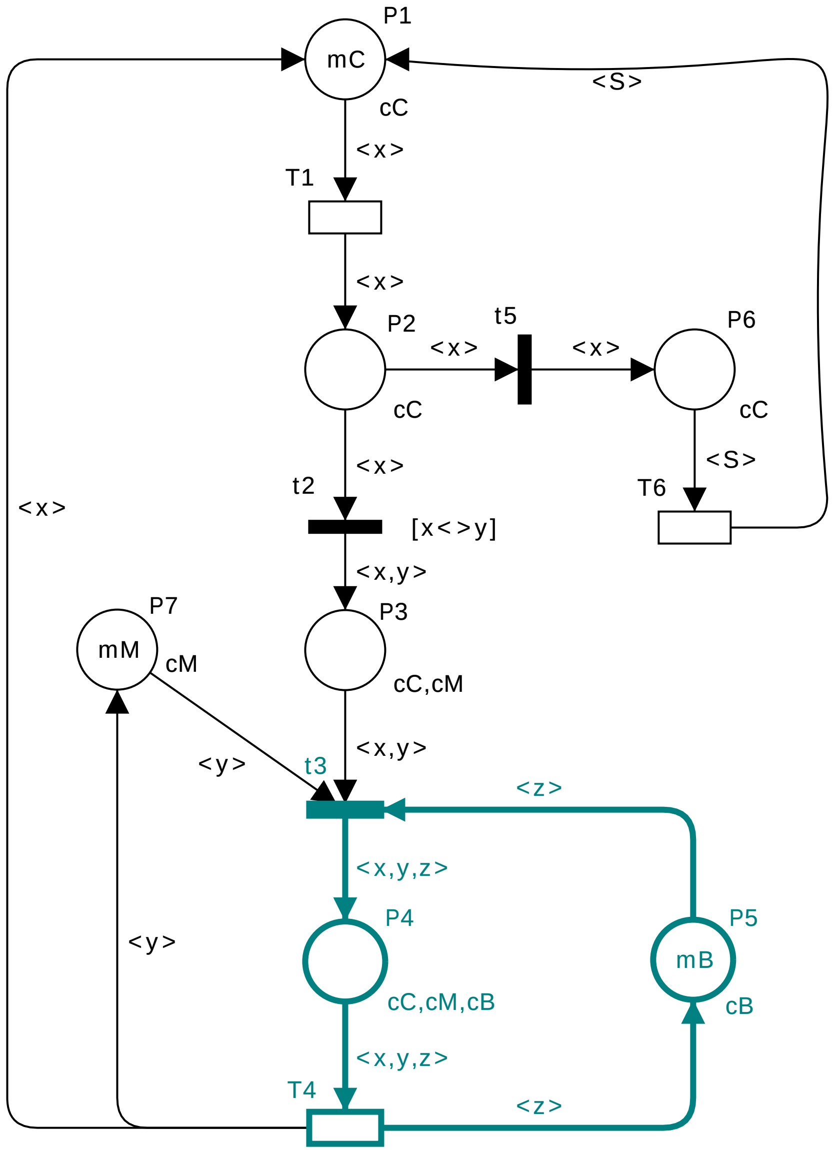
- *C* computers access memory modules of other computers over *B* buses
- Units compete for resources (t_3) and release them after their job (T_4)
- Cycles are synchronized (T_6)



Two kinds of synchronization:

- Use of resources (competition):
  - Buses
  - Memory modules
- Cycle (cooperation)

2.3 SSNB decolourisation examples: MCPLC



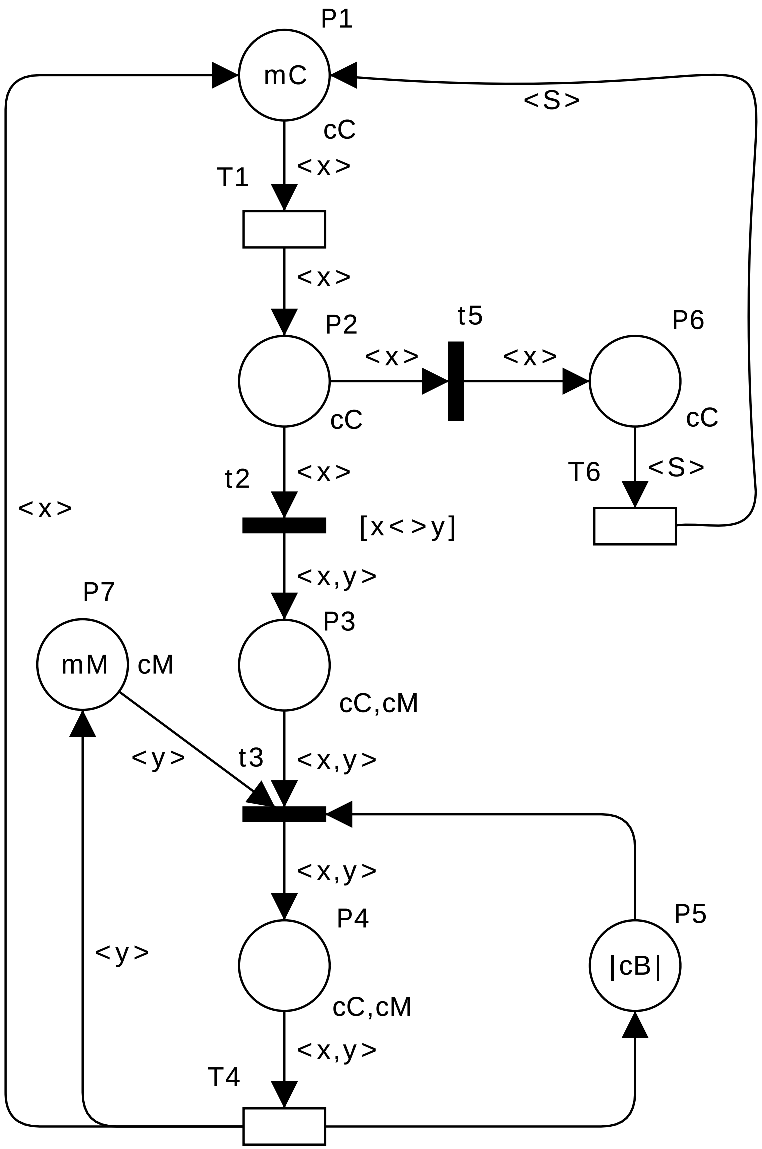
- **Flow** (buses):
  - *P*4, *P*5, *t*3, *T*4
  - colour domain *cB*, variable *z*
- **Colour shrinking function**: $cB \rightarrow cB'$: $\forall\ c \in cB$: $sh(c) = \bullet$

- Intuitively:

  Communication request does not ask for a specific bus (no condition on variable *z* in *t*3)

  ⇒ no reason to distinguish buses with colours

2.3 SSNB decol. examples: MCPLC – decolourised net



In general, only buses can be decolourised here
+ no timing changes necessary.

We only get a population of buses

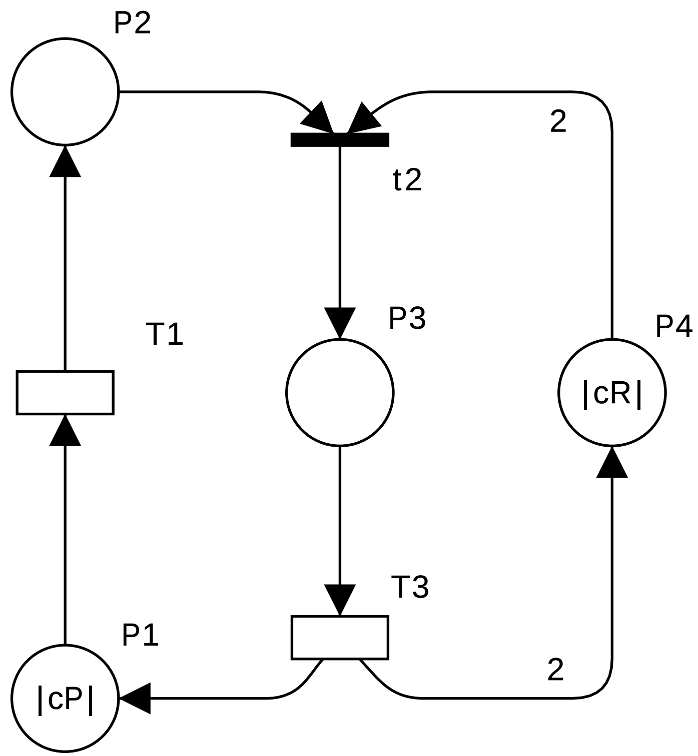
To get a P/T net,
unfolding is needed what means usually two kinds of problems:

- **population ↓**
- **net size ↑**

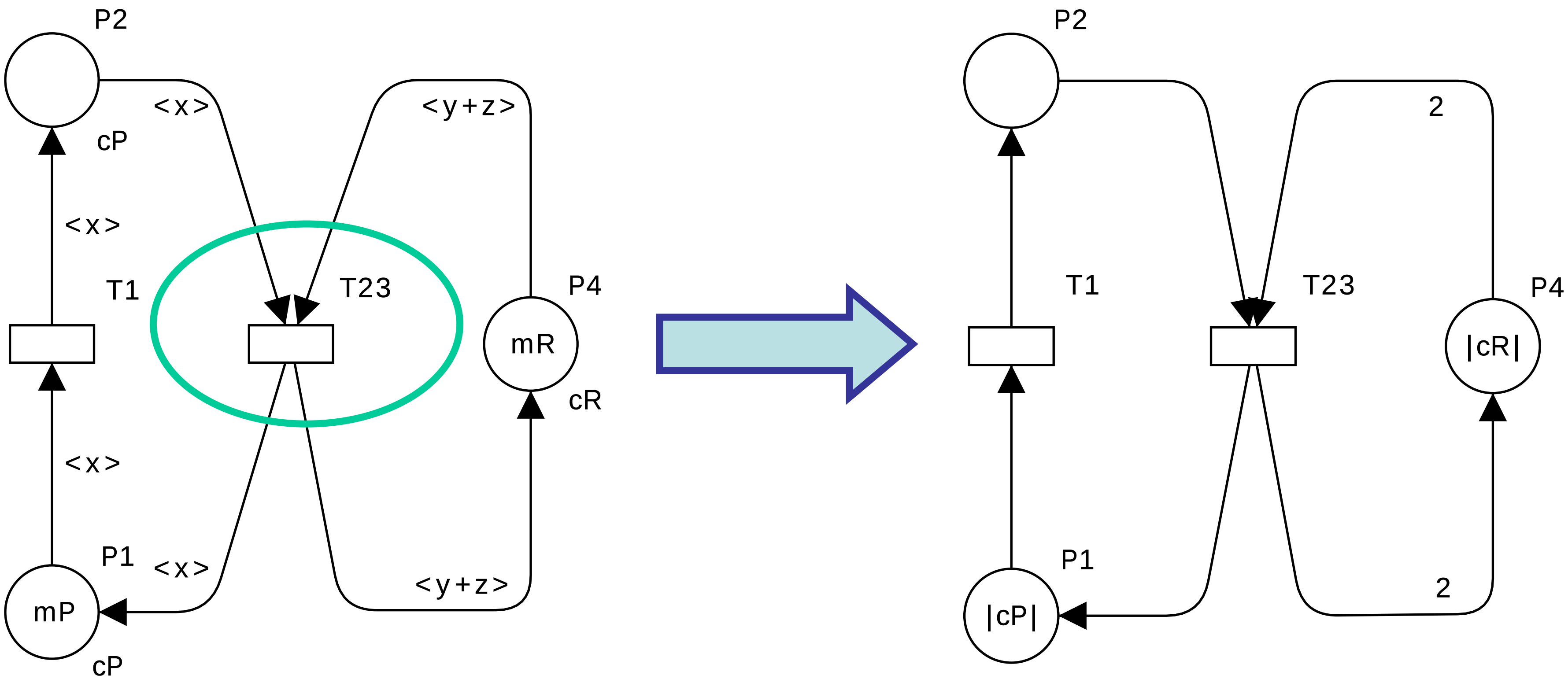
2.3 SSNB decolourisation examples: DinPhilCommon – decolourised net

- Completely decolourised model – the same for:
  - Resources assigned individually
  - Resources assigned in bag

- Timing: **no changes needed**



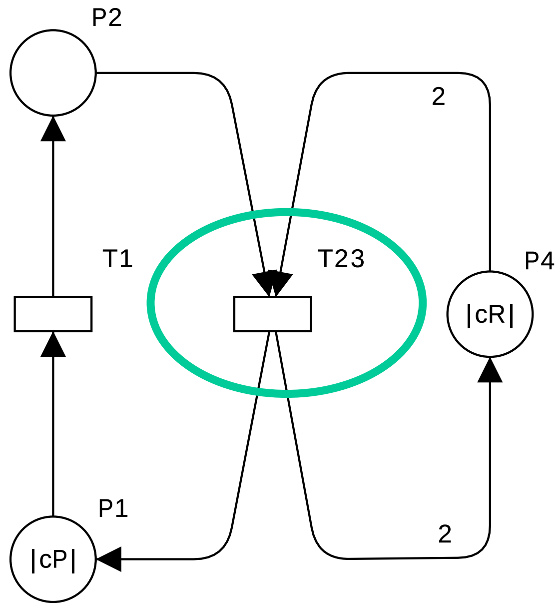
2.3 SSNB decolourisation examples: DinPhilCommon – net reduction



In the coloured net transitions *t*2 and *T*3 can be agglomerated to *T*23.

Modified meaning: *all waiting philosophers "eat at once" with the same resources and only the fastest one is fed up*

2.3 SSNB decol. examples: DinPhilCommon reduced



- Decolourisation on autonomous level: straightforward

- Decolourisation on timed level:
  - Modifying firing rate of T23
    - Using *TPAR2 Multiple input places* and *TPAR3 Addition on input arc*
    - Flow (not ISS): $$f(T23) = \lambda.\mathbf{m}(P2).\mathbf{m}(P4).(\mathbf{m}(P4) - 1)$$
  - If bags are used:
    - Using *TPAR2* and *TPAR4 Bag on input arc*
    - Flow (not ISS): $$f(T23) = \lambda.\mathbf{m}(P2).\binom{\mathbf{m}(P4)}{2}$$

Summary and Future Work

- Decolourisation of SSNB models
  - On autonomous level, procedure enhanced to include use of bags
  - On timed level, set of 9 rules for transition parameters adjustment defined
- Decolourisation process has got limits.
  - Where not applicable, unfolding must be used – that brings usually two kinds of problems:
    - **population** ↓
    - **net size** ↑
  - Non-colour-safe places and complicated operations with bags are obstacles in successful decolourisation of timed models. More research required.

Thank you for your attention

michal.zarnay@fri.uniza.sk, silva@unizar.es