# An Introduction to Petri nets and how to analyse them...

G. Geeraerts

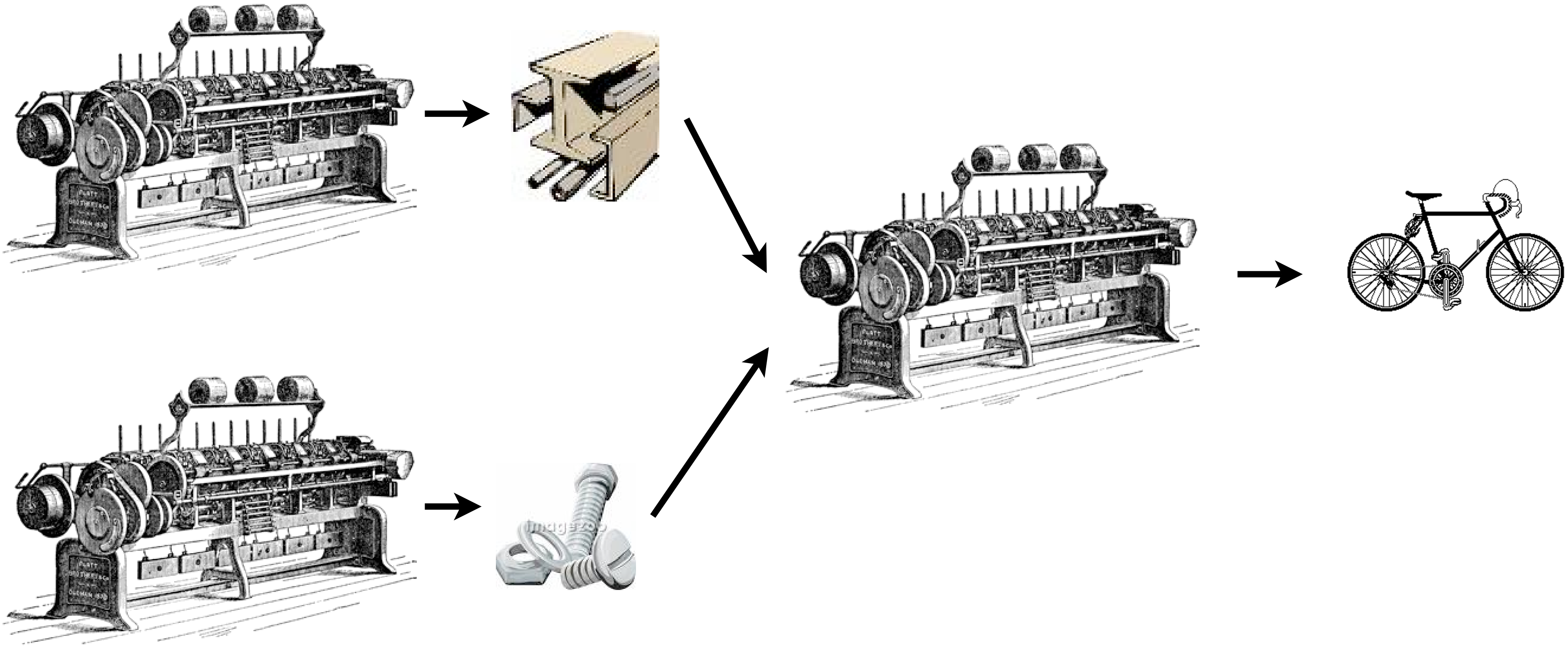
Groupe de Vérification - Département d'Informatique
Université Libre de Bruxelles

# Introduction

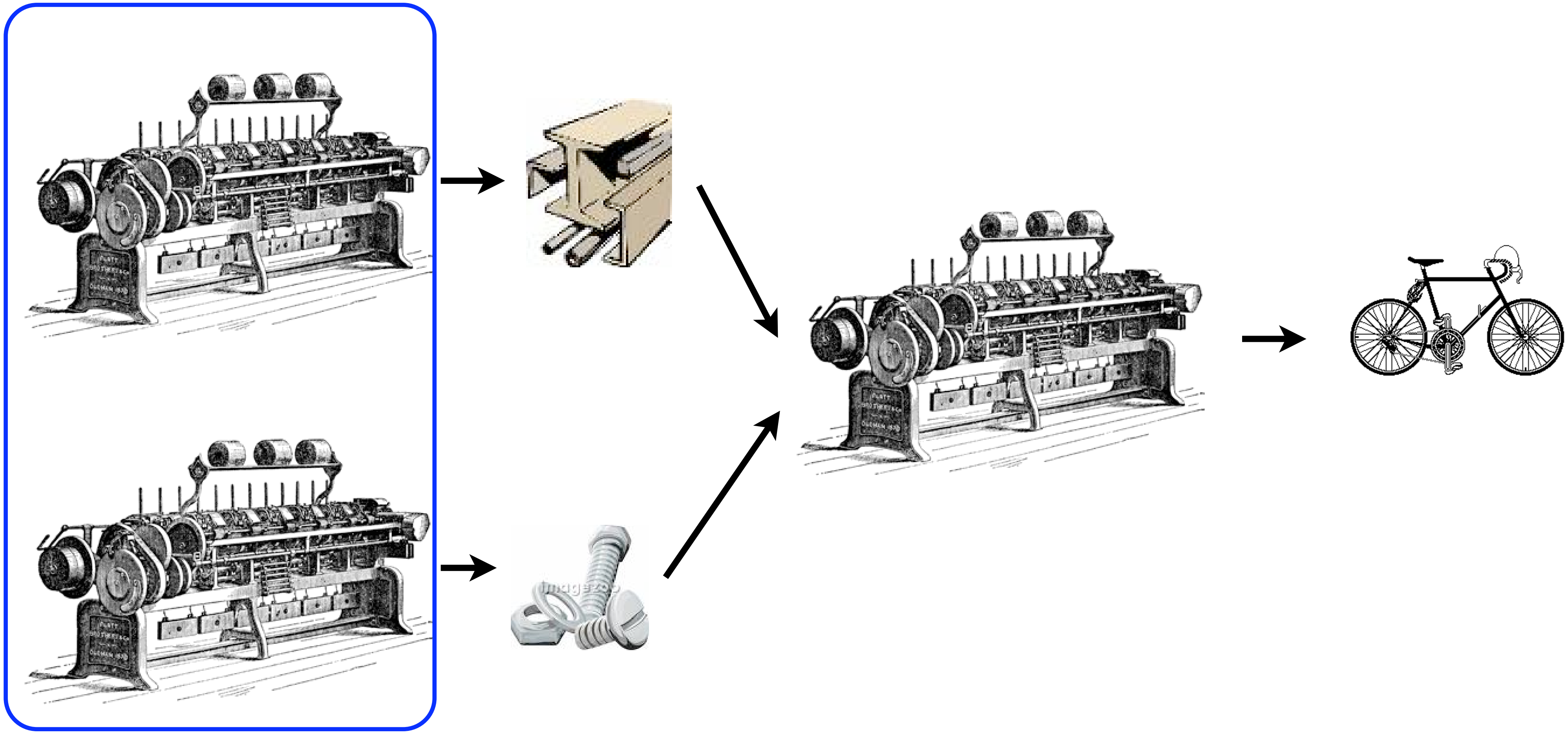
# Introduction

- Concurrency: property of a "system" in which many "entities" act at the same time and interact.
  - Often found in many application:
    - Computer science (e.g.: parallel computing)
    - Workflow
    - Manufacturing systems
    - ....

# Introduction
# Concurrency

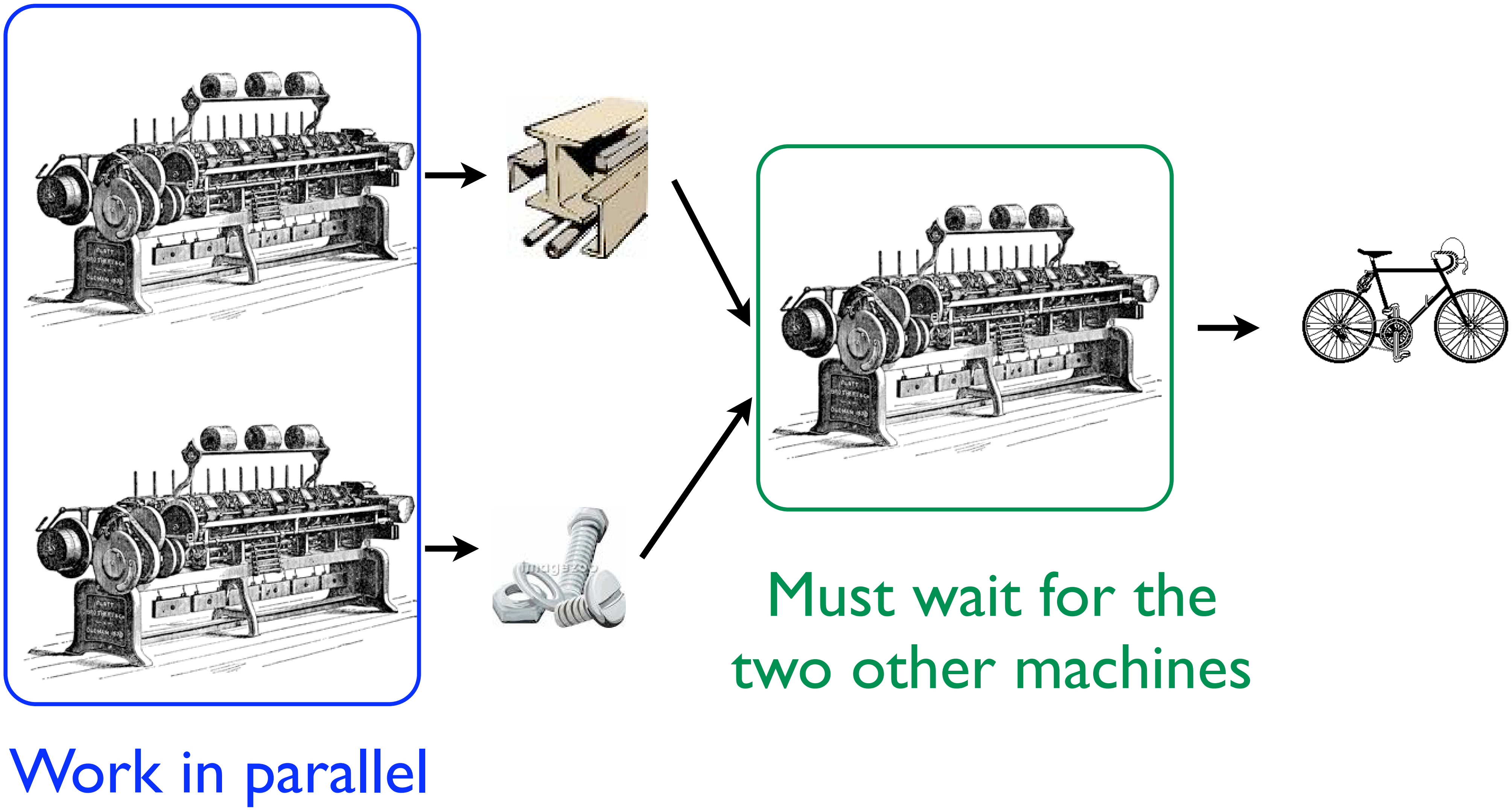


# Introduction
# Concurrency

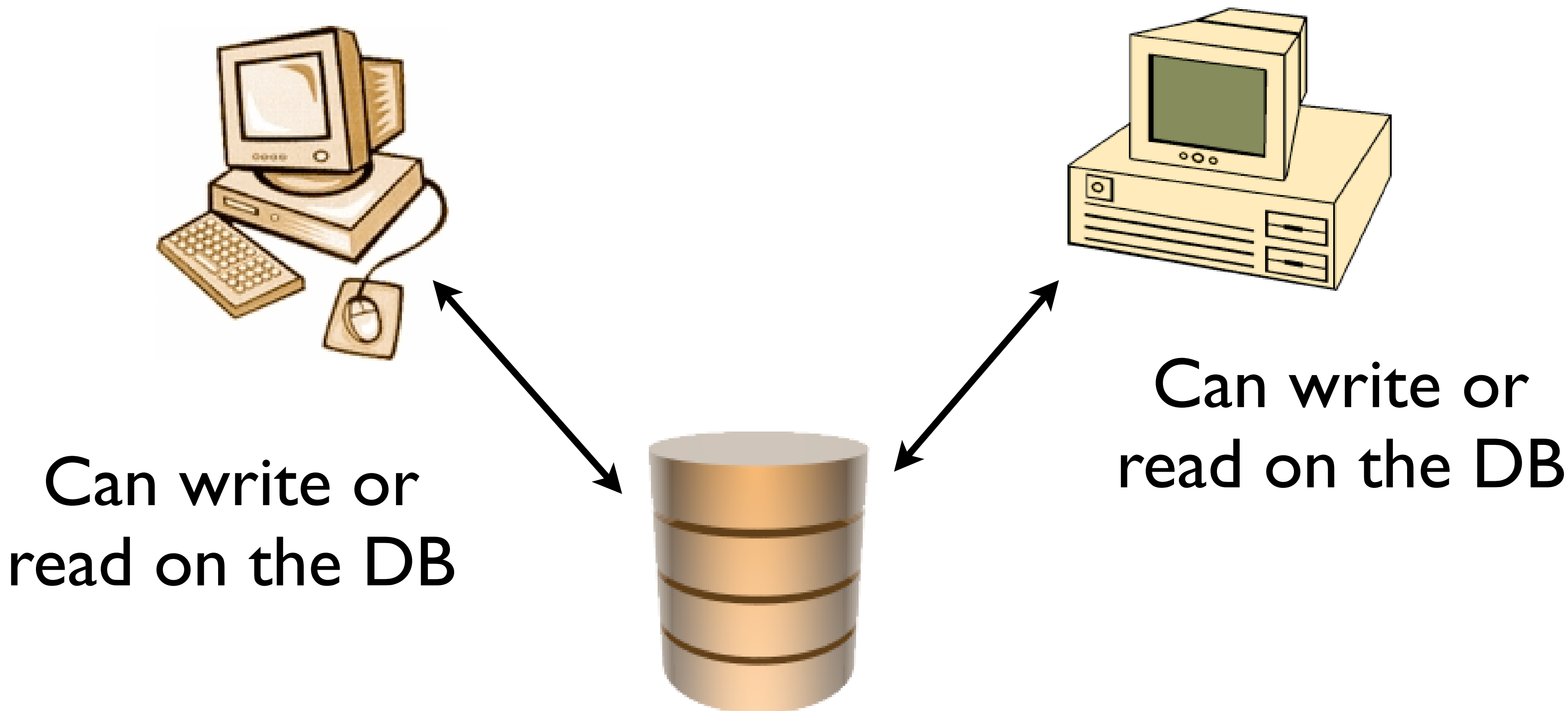


Work in parallel

# Introduction
# Concurrency



# Introduction
# Concurrency



# Introduction
# Concurrency

Boss

# Introduction
# Concurrency

Boss

# Introduction
# Concurrency

Boss

Employees: work in parallel

# Introduction
# Concurrency

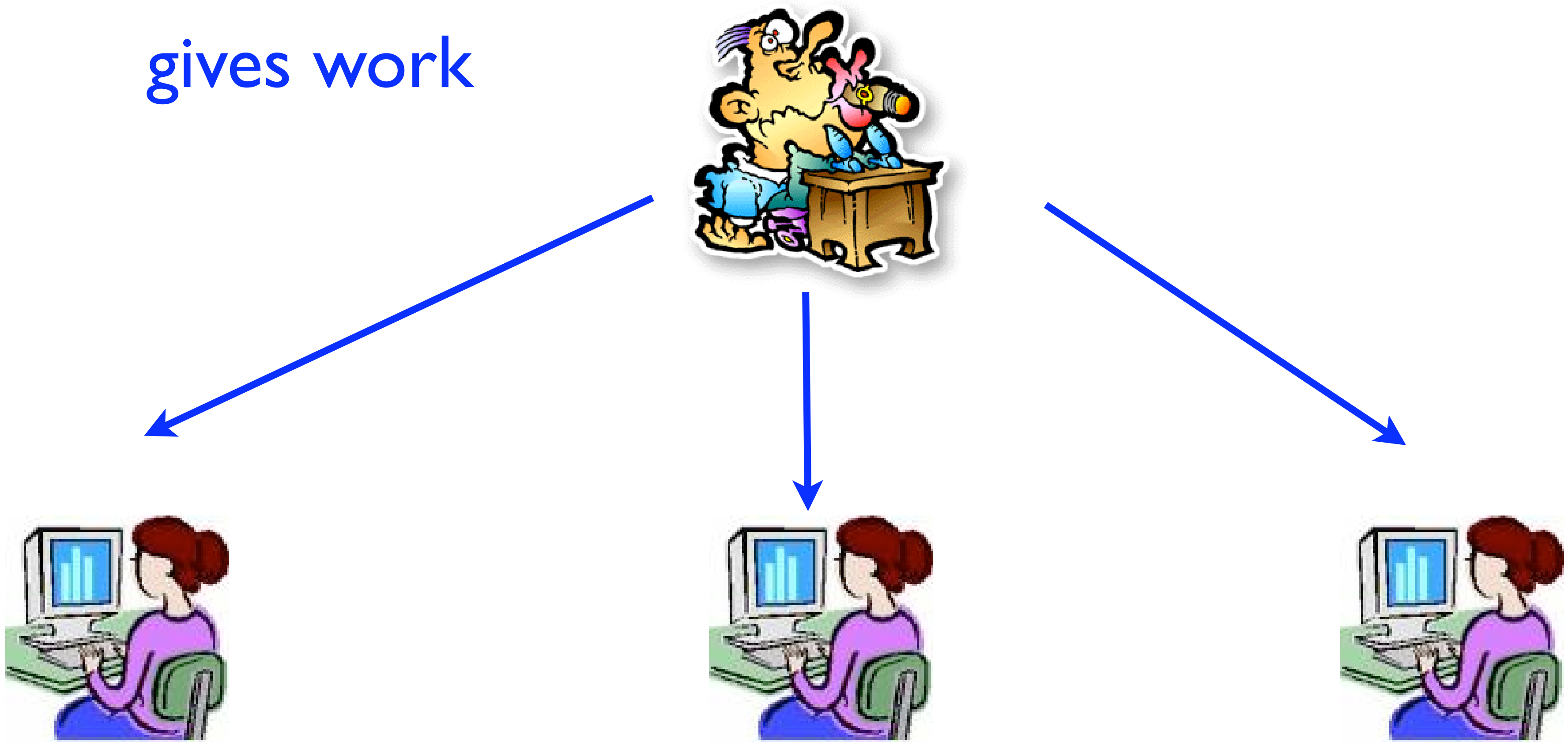
Boss

gives work

Employees: work in parallel

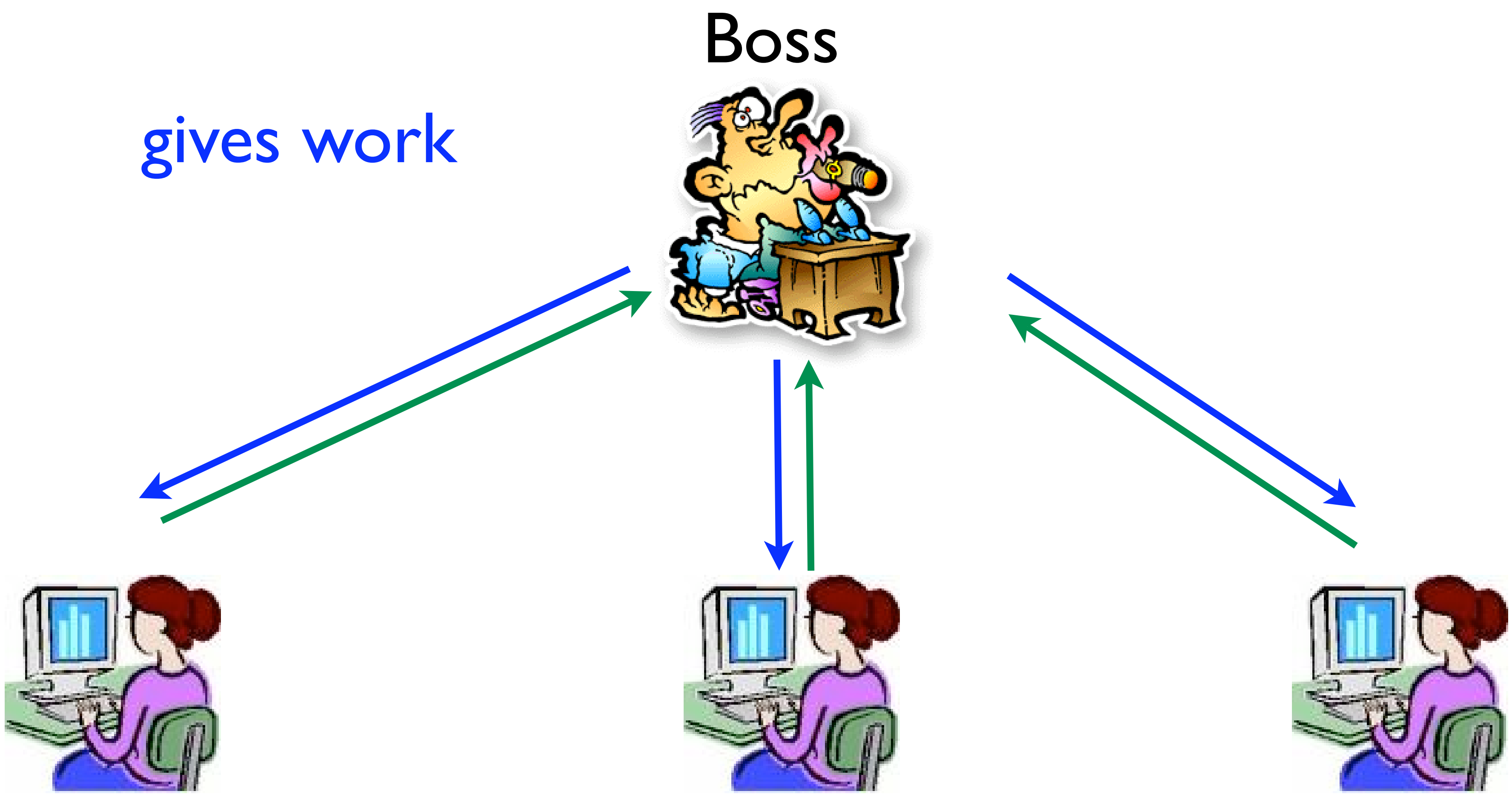
# Introduction
# Concurrency

Boss



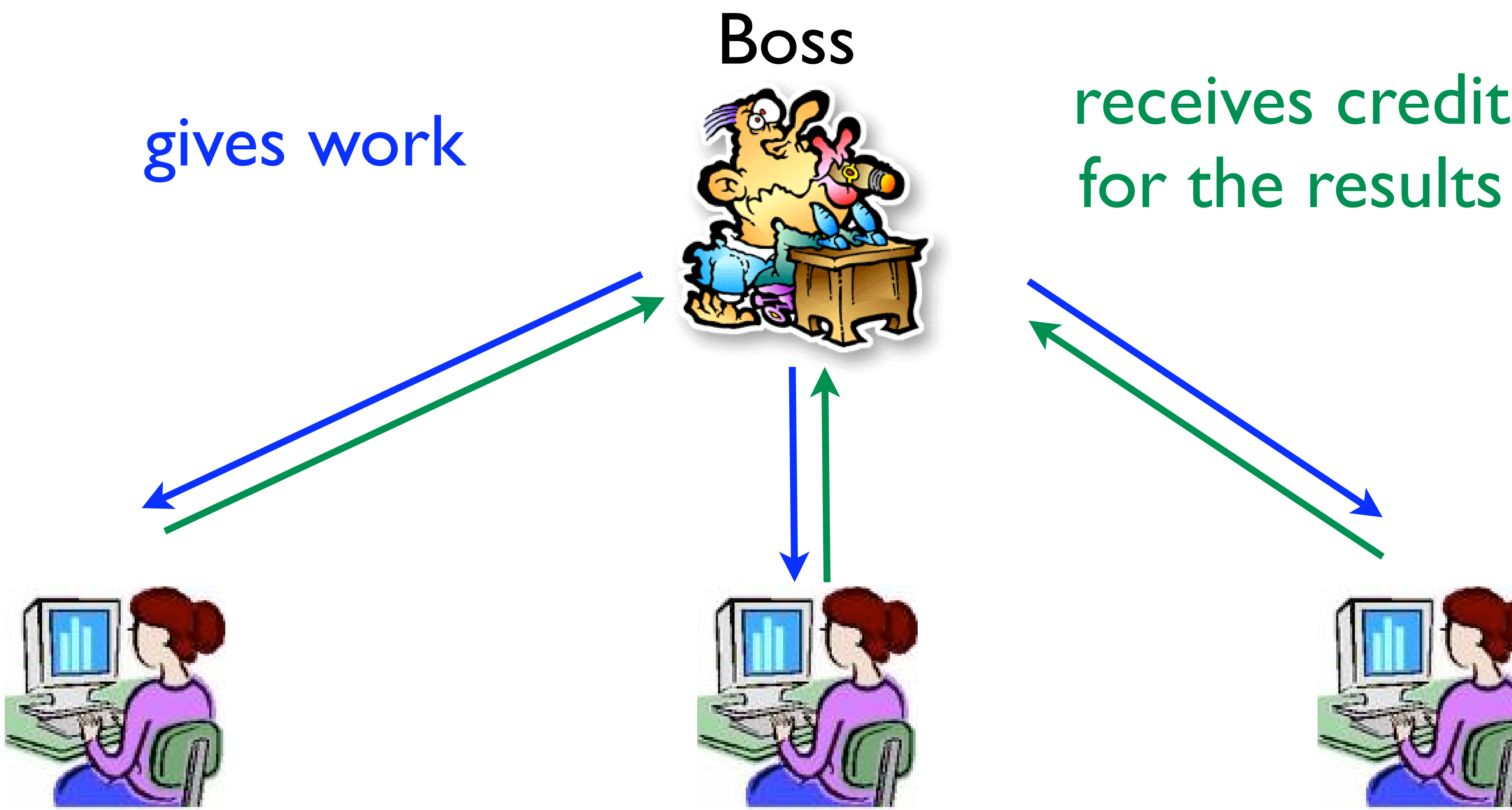

Employees: work in parallel

# Introduction
# Concurrency




Employees: work in parallel

# Introduction
# Concurrency




Employees: work in parallel

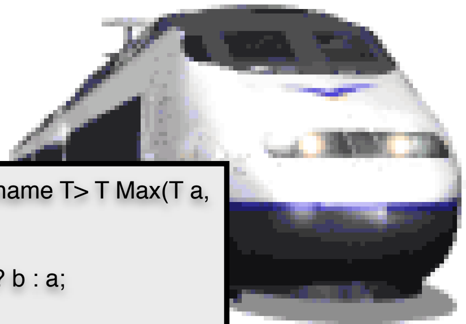
# Introduction

- Petri nets are a tool to model concurrent systems and reason about them.
- Invented in 1962 by C.A. Petri.

# The aim of the talk

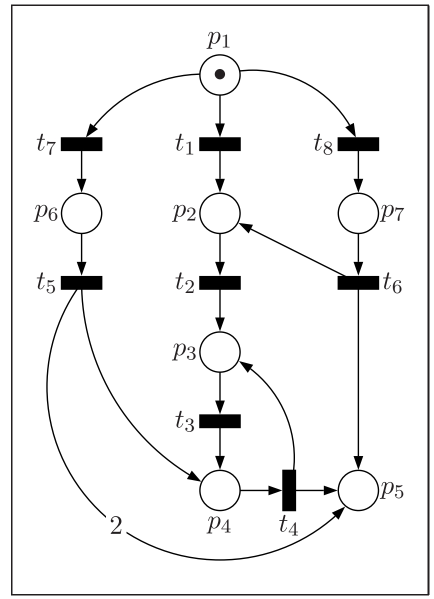
- Introduce you to Petri nets (and some of their extensions)
- Explain several analysis methods for PN
  - i.e., what can you ‘ask’ about a PN ?
- Give a rough idea of the research in the verification group at ULB...
  - ... and foster new collaborations ?

# How I use Petri nets



```
template <typename T> T Max(T a,
T b)
{
   return a < b ? b : a;
}

#include <string>
int main()  // fonction main
{
   int i = Max(3, 5);
   char c = Max('e', 'b');
   std::string s = Max(std::string
("hello"), std::string("world"));
   float f = Max<float>(1, 2.2f);
```

abstraction

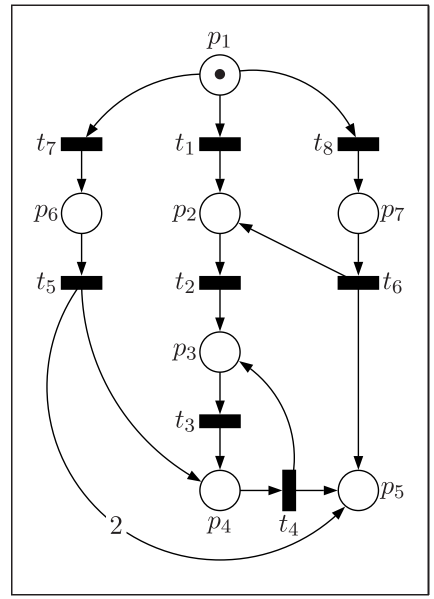



Analysis method
of PN

# How you might use PN

Your favorite application

abstraction

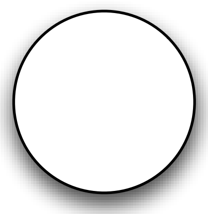


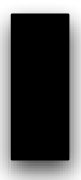

Analysis method of PN

# Intuitions

# Ingredients

A Petri net is made up of...

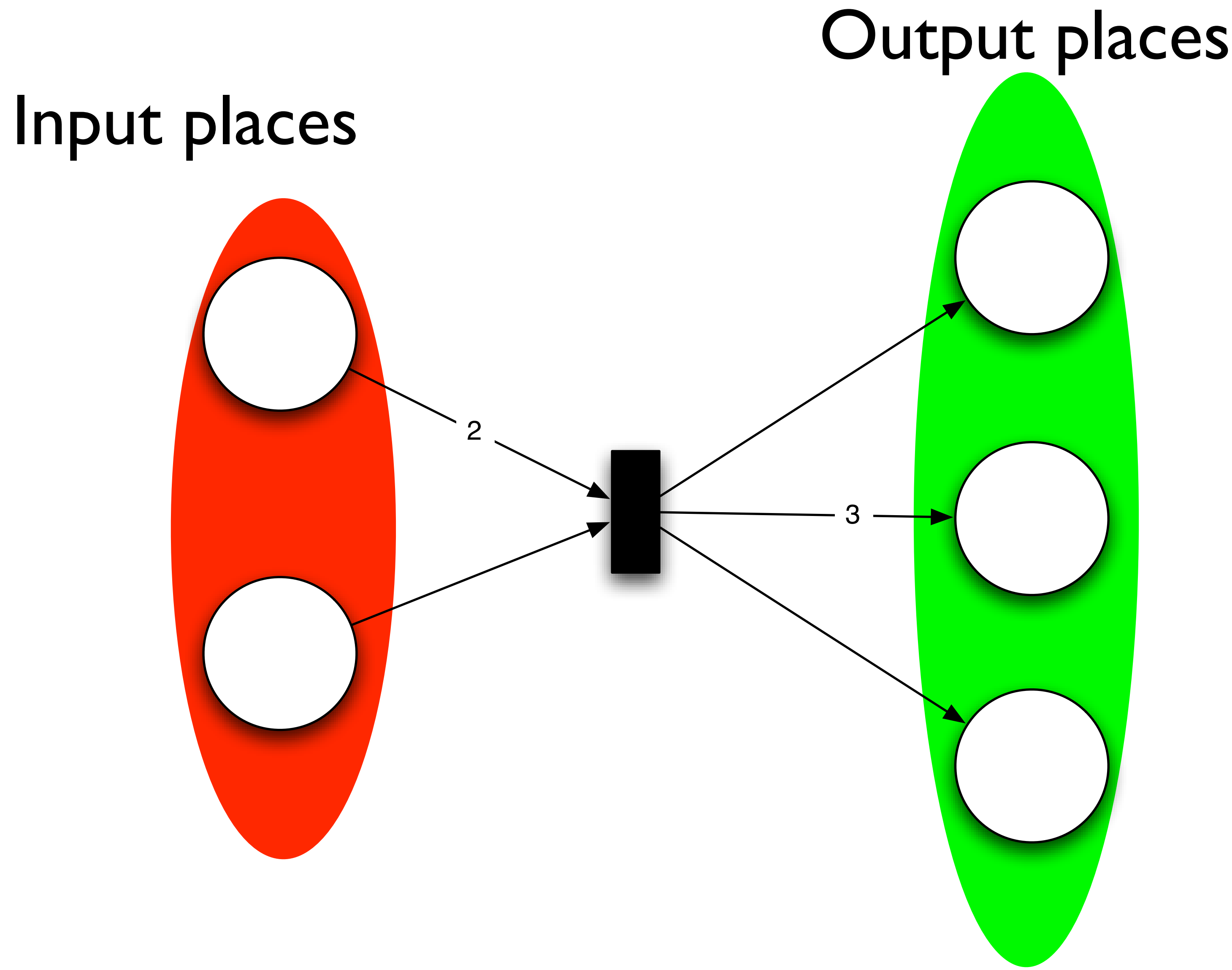
Places  = some type of resource

Transitions  consume and produce resources

Tokens  = one unity of a certain resource

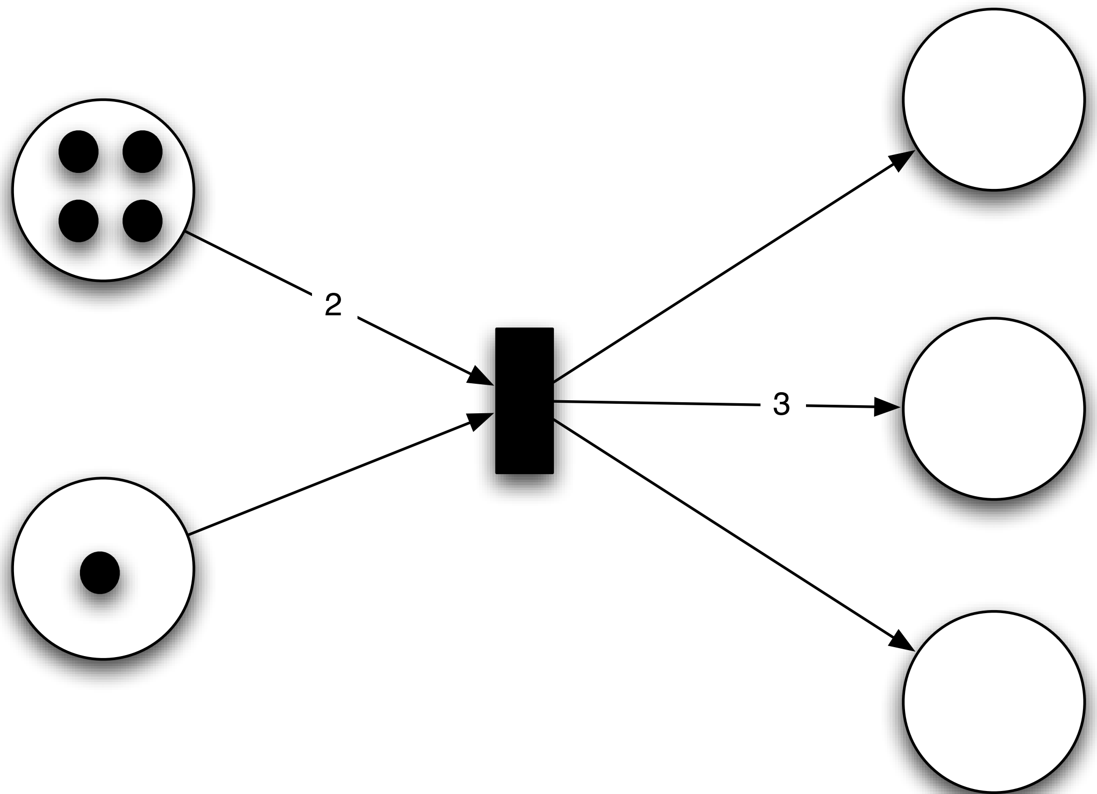
Tokens 'live' in the places

# Transitions



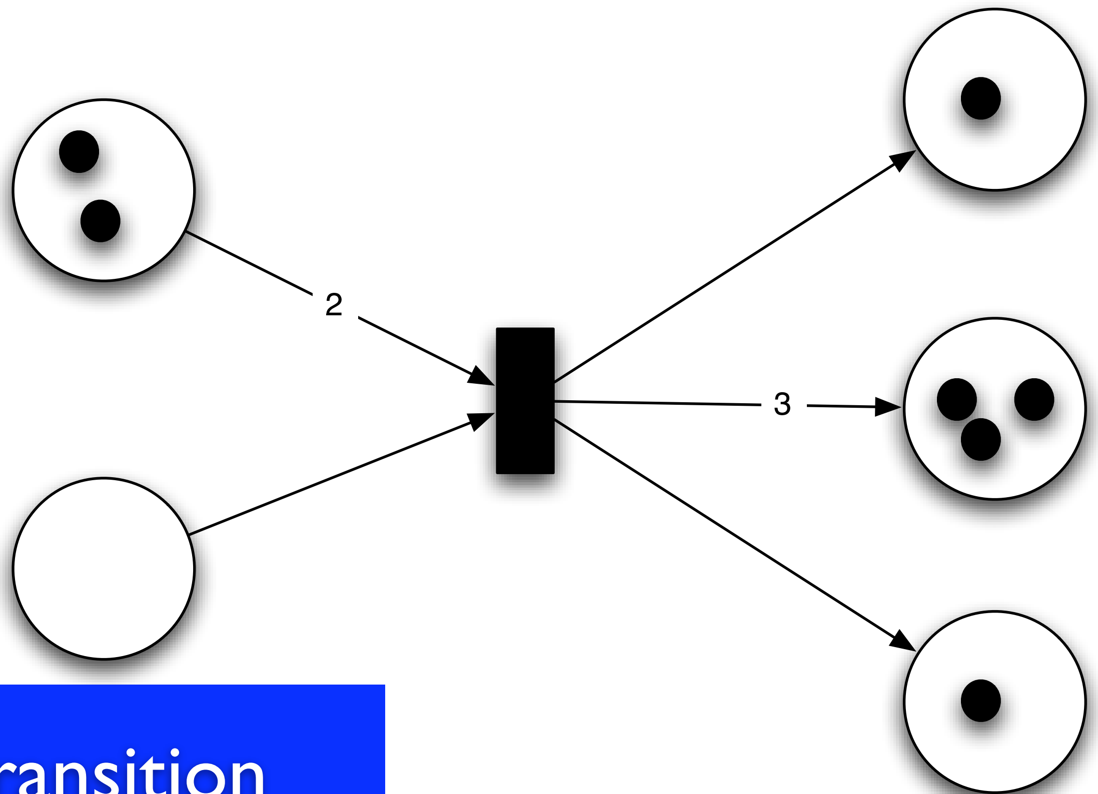
# Firing a transition

Transitions consume tokens from the input places and produce tokens in the output places



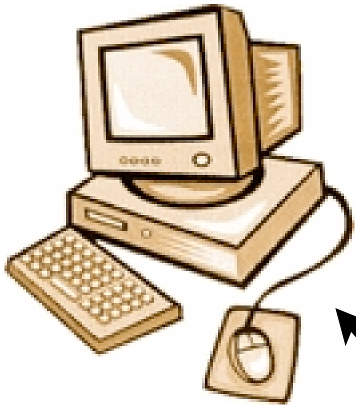
# Firing a transition

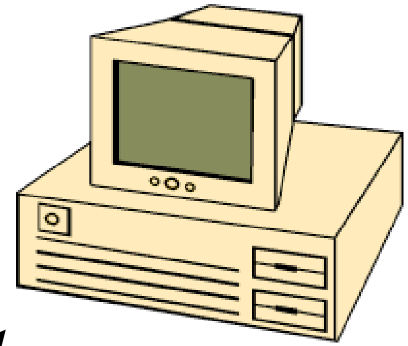
Transitions consume tokens from the input places
and produce tokens in the output places




Now, the transition
cannot be fired anymore

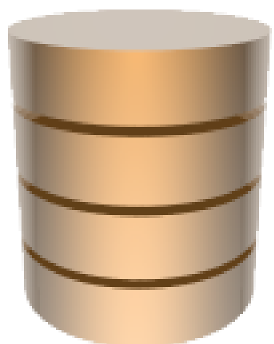
# Example 1





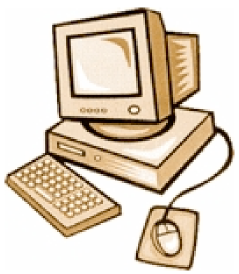
Can write or read on the DB

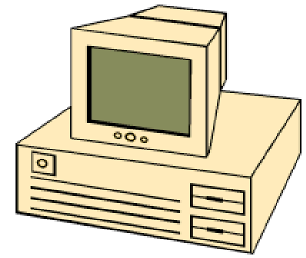
Can write or read on the DB

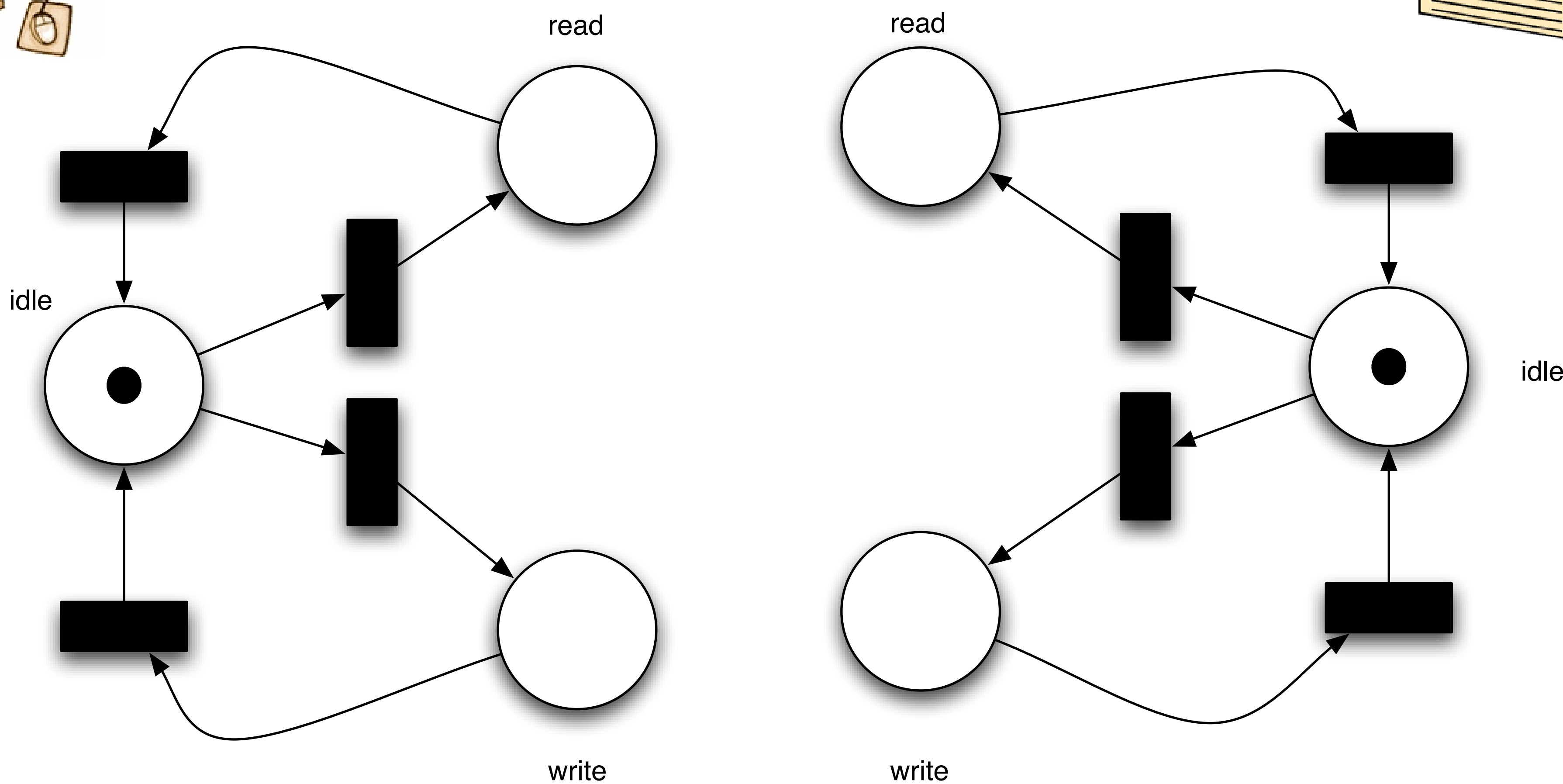


The two machines cannot write at the same time

# Example 1

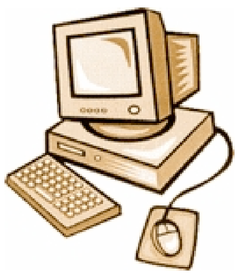


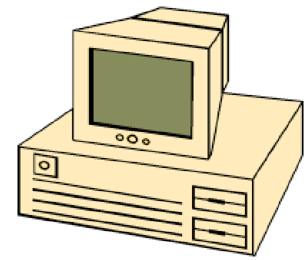


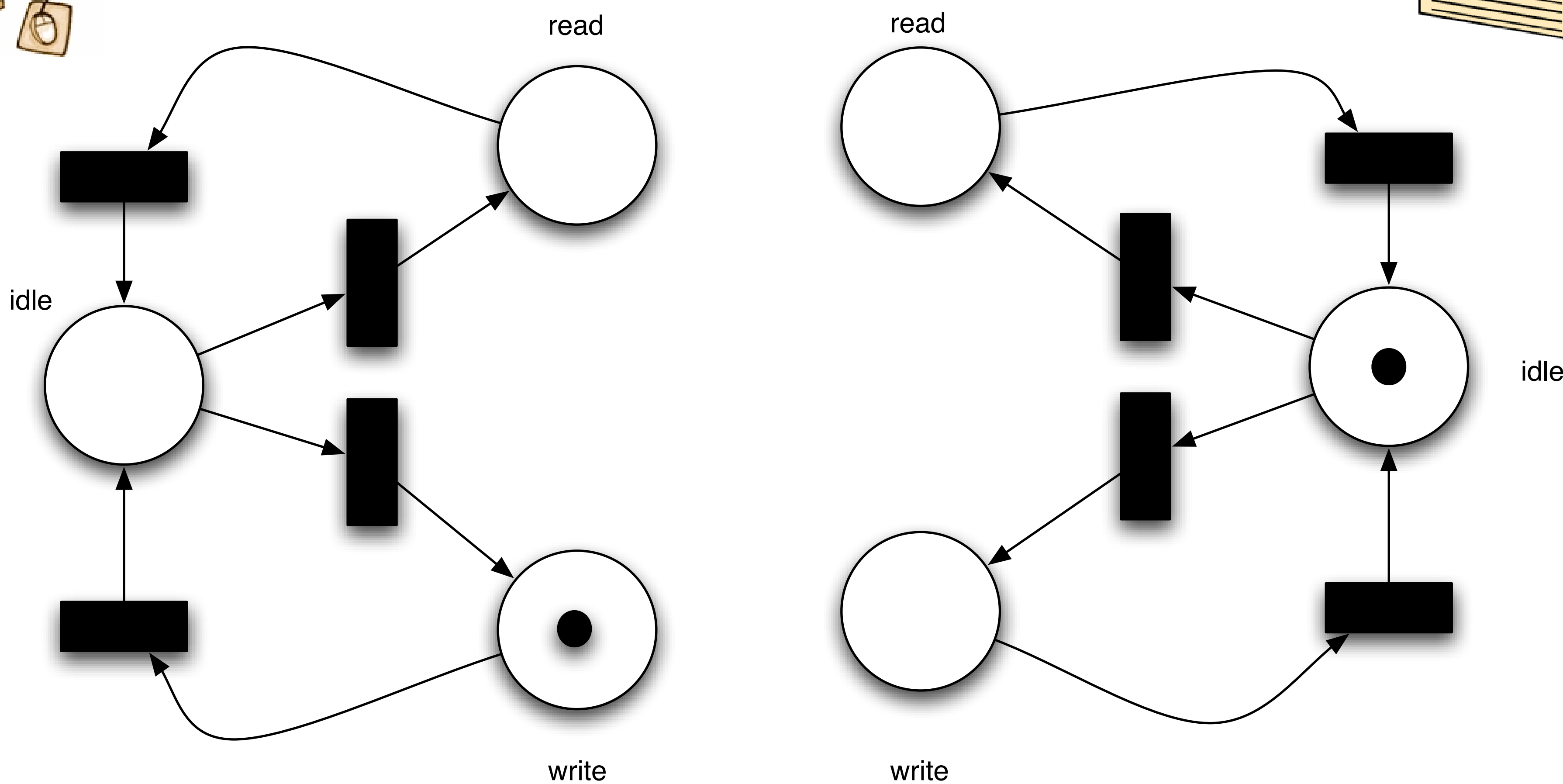



The token tells us the state of the process

# Example 1

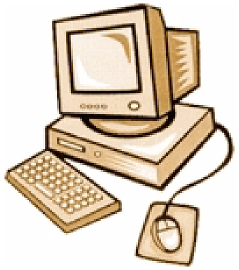


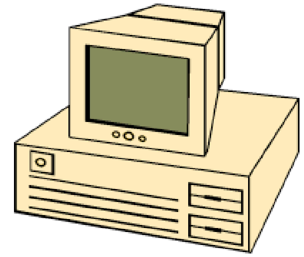


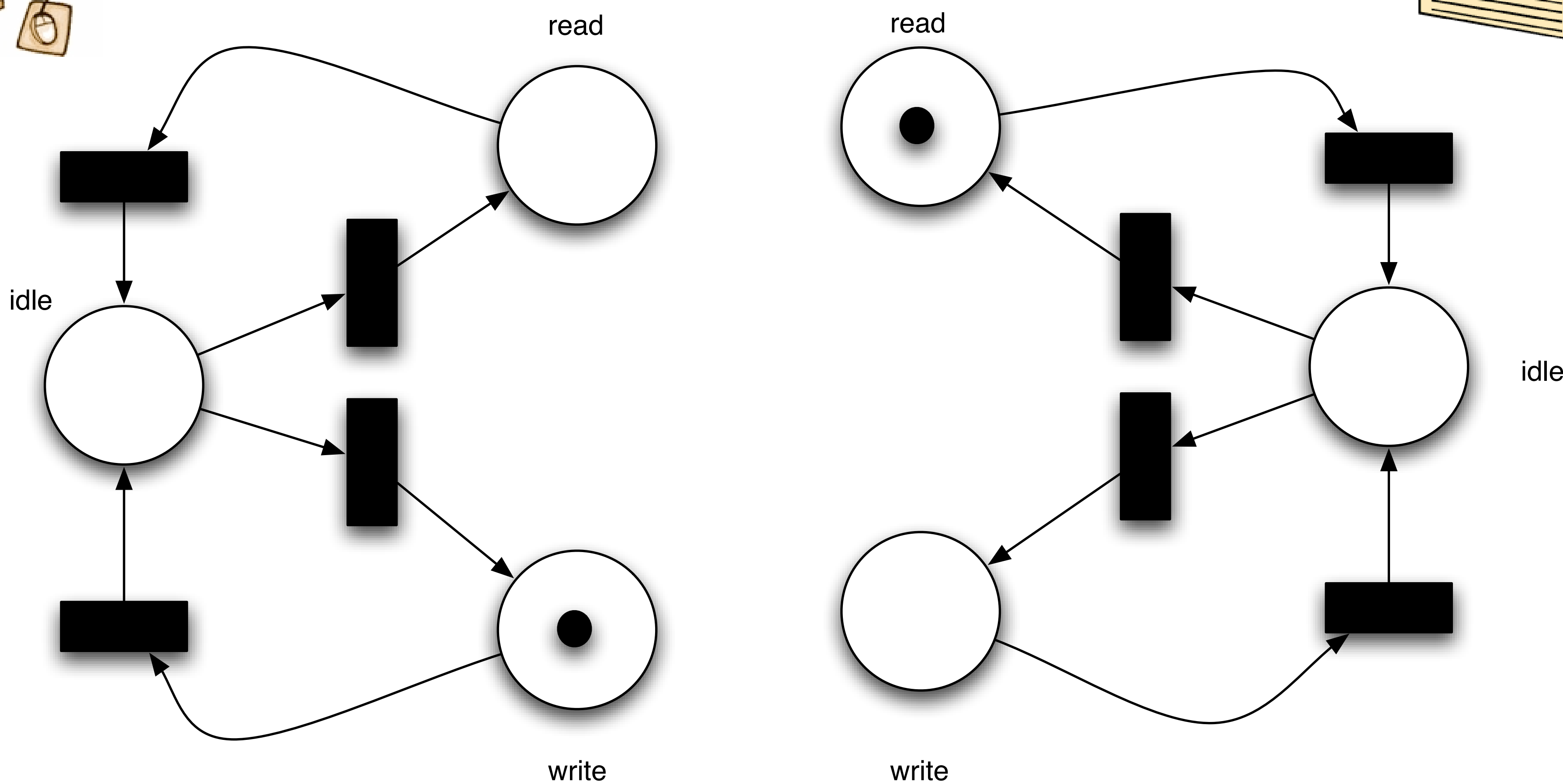



The token tells us the state of the process

# Example 1

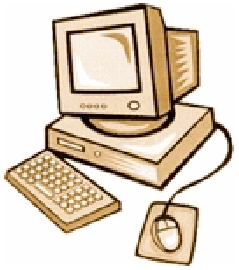


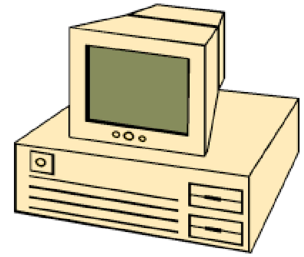


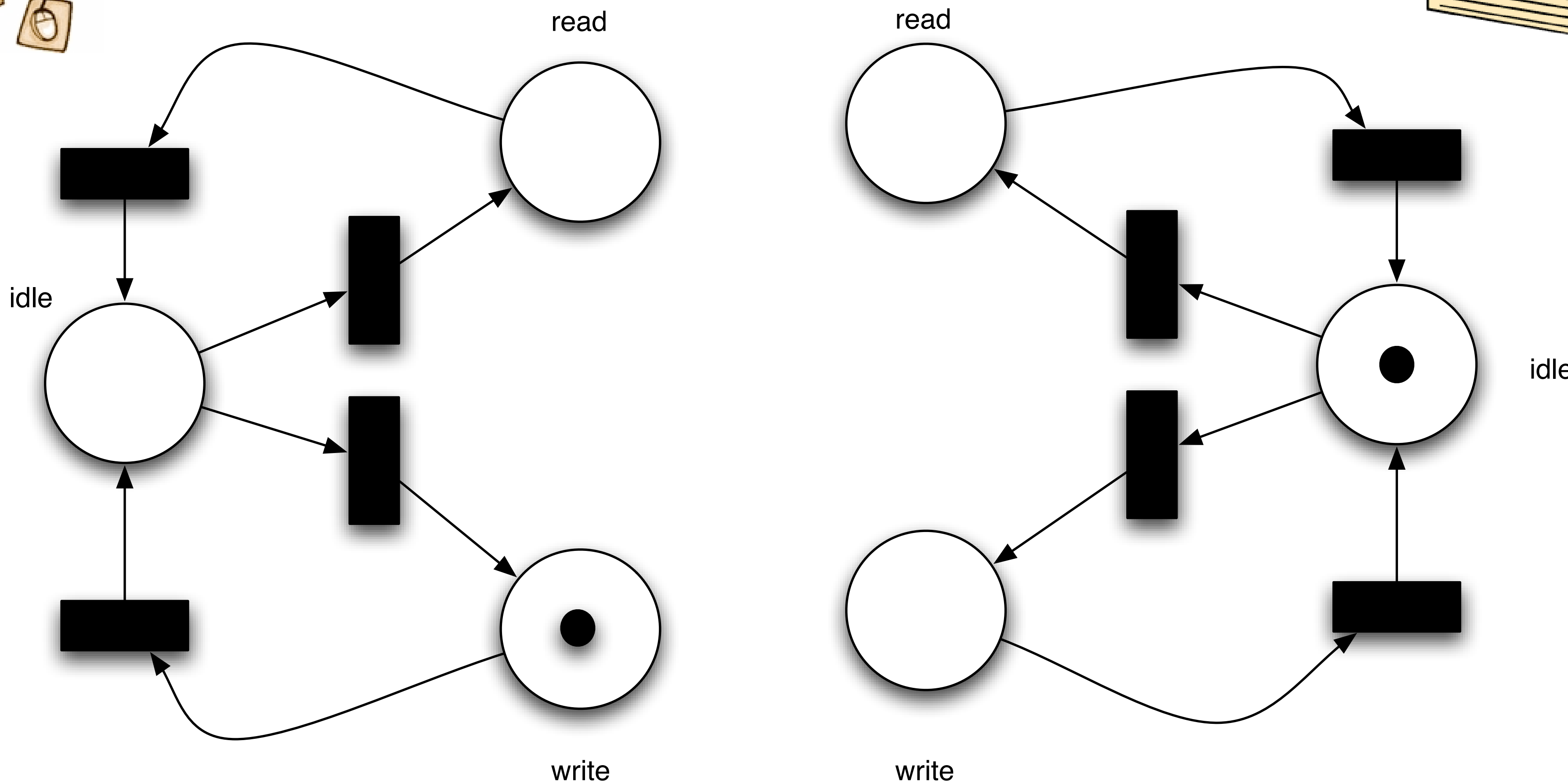



The token tells us the state of the process

# Example 1

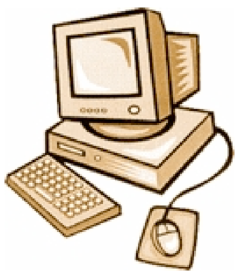


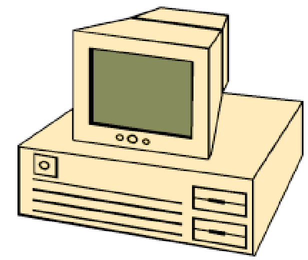


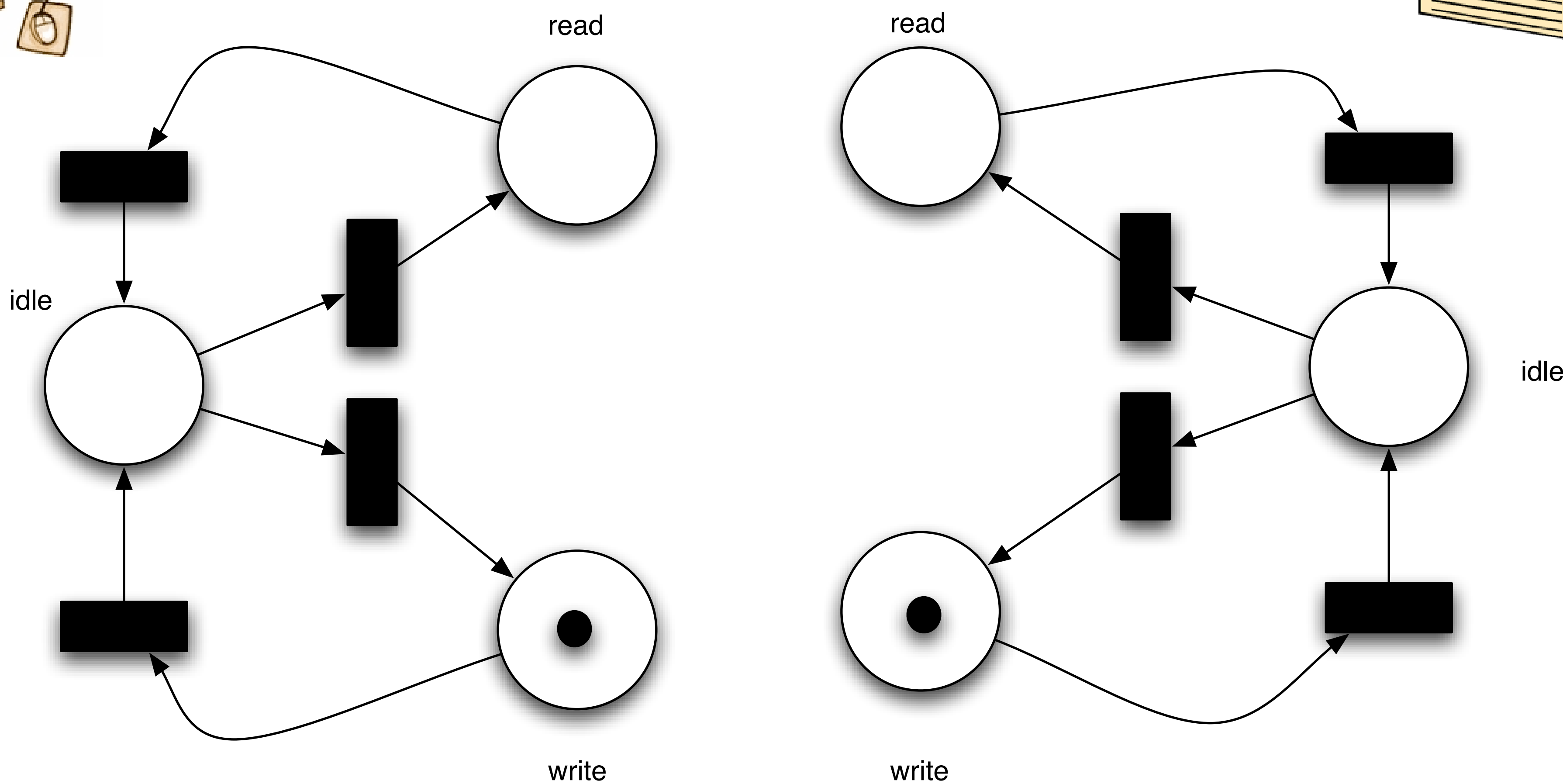



The token tells us the state of the process

# Example 1

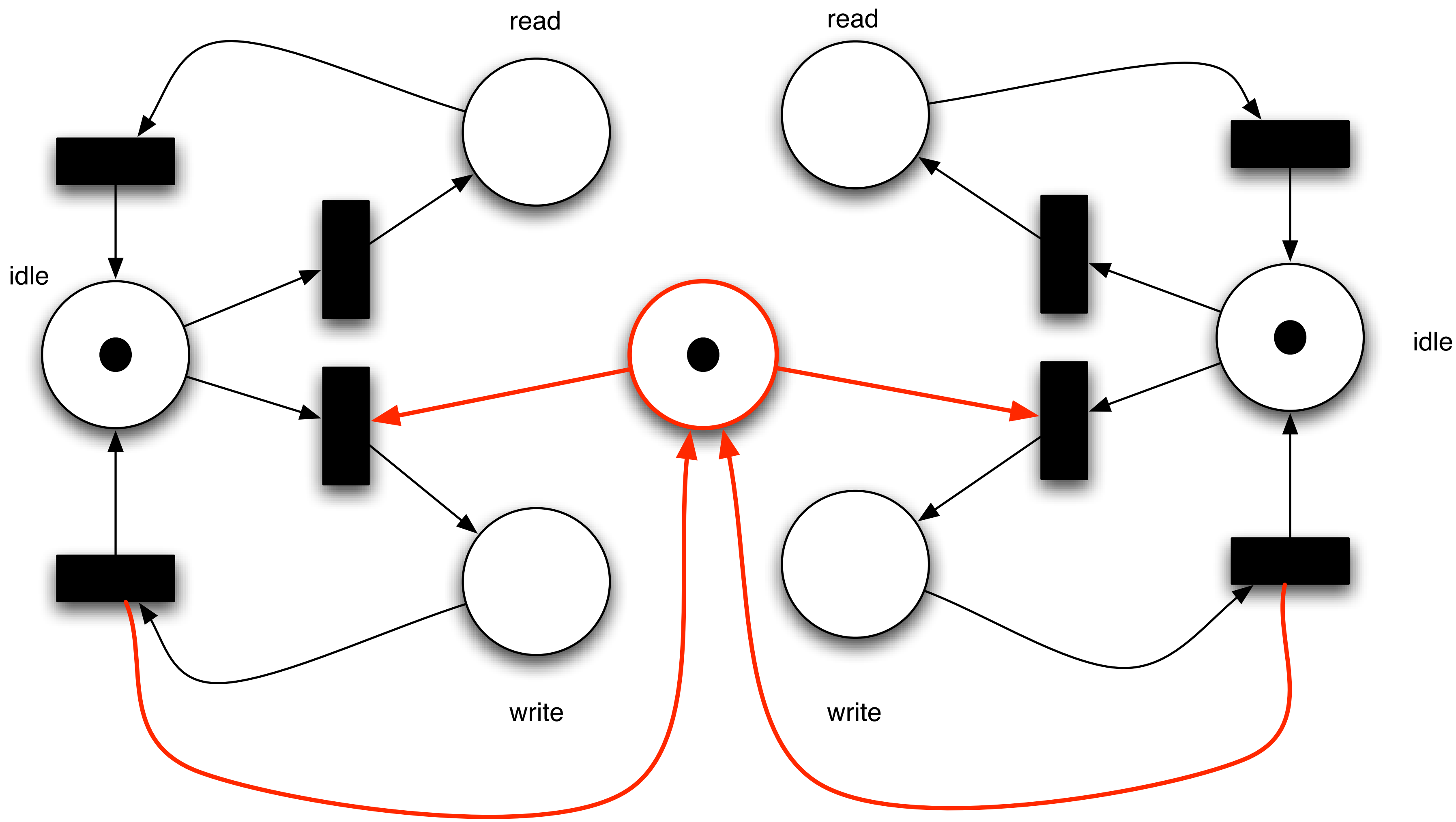






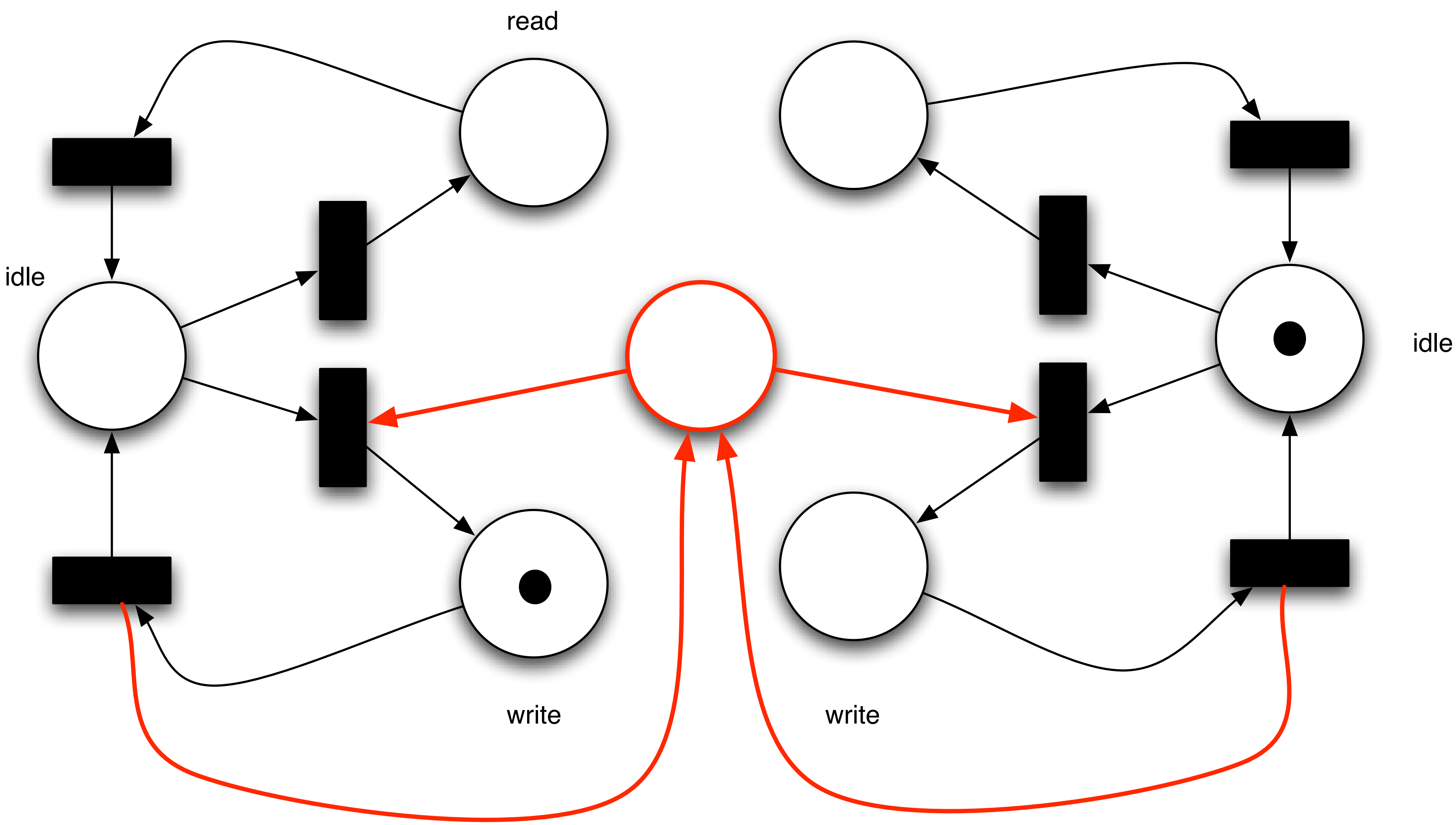

The token tells us the state of the process

# Example 1



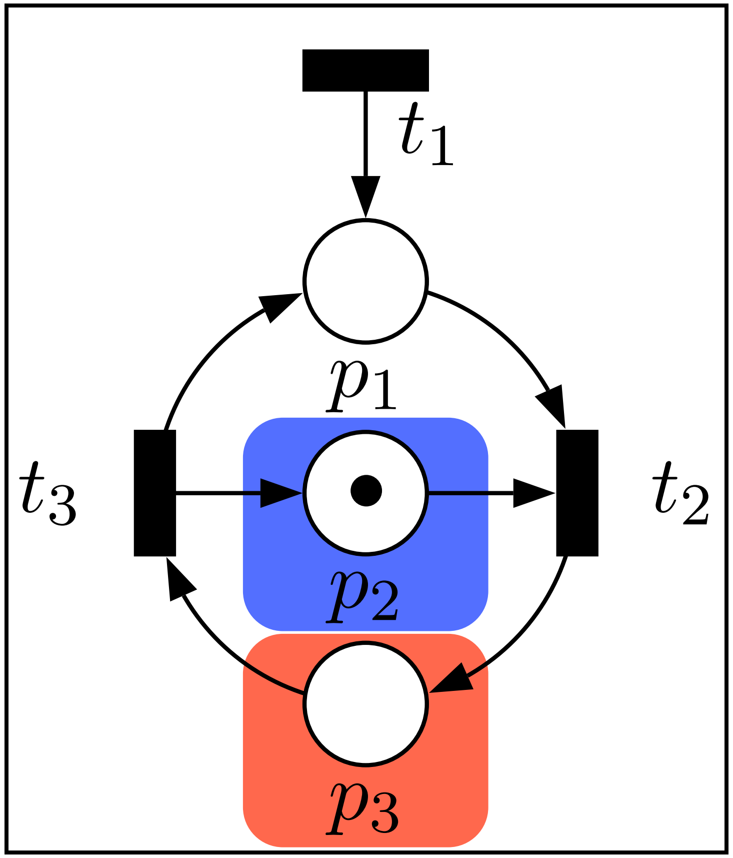

Add a lock to ensure mutual exclusion

# Example 1



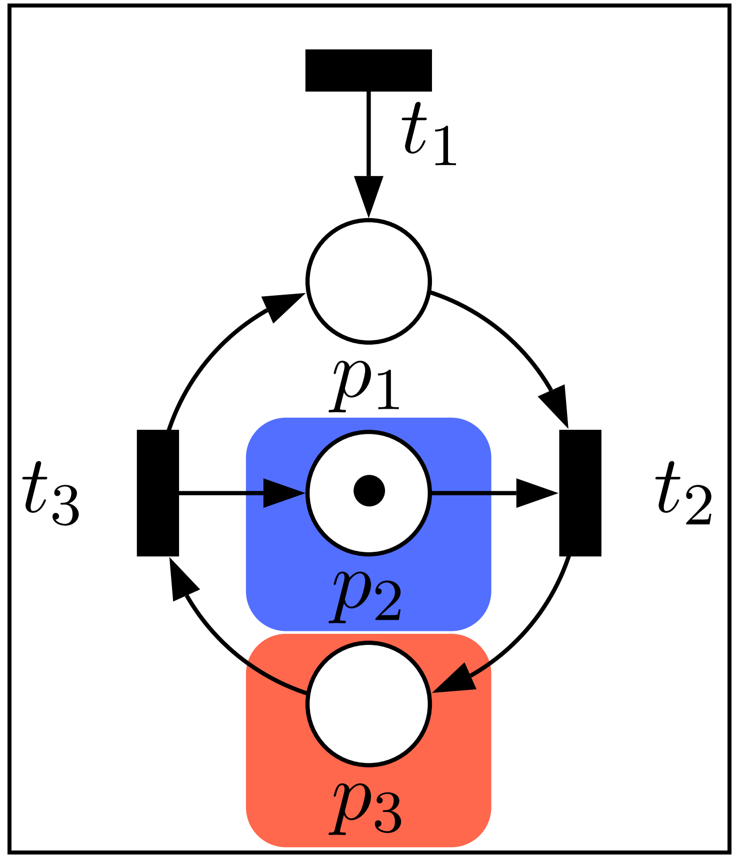
# Example 2

```
mutex M ;

Process P {
    repeat {
        take M ;
        critical ;
        release M ;
    }
}
```



# Example 2

```
mutex M ;

Process P {
    repeat {
        take M ;
        critical ;
        release M ;
    }
}
```




Here, we have applied a counting abstraction

# Plan of the talk

- Preliminaries
- Tools for the analysis of PN
  - reachability tree and reachability graph
  - place invariants
  - Karp & Miller and the coverability set
- The coverability problem
- More on PN: extensions...
- Conclusion

# Plan of the talk

Detailed coverage

- Preliminaries
- Tools for the analysis of PN
  - reachability tree and reachability graph
  - place invariants
  - Karp & Miller and the coverability set
- The coverability problem

Survey

- More on PN: extensions...
- Conclusion

# Preliminaries

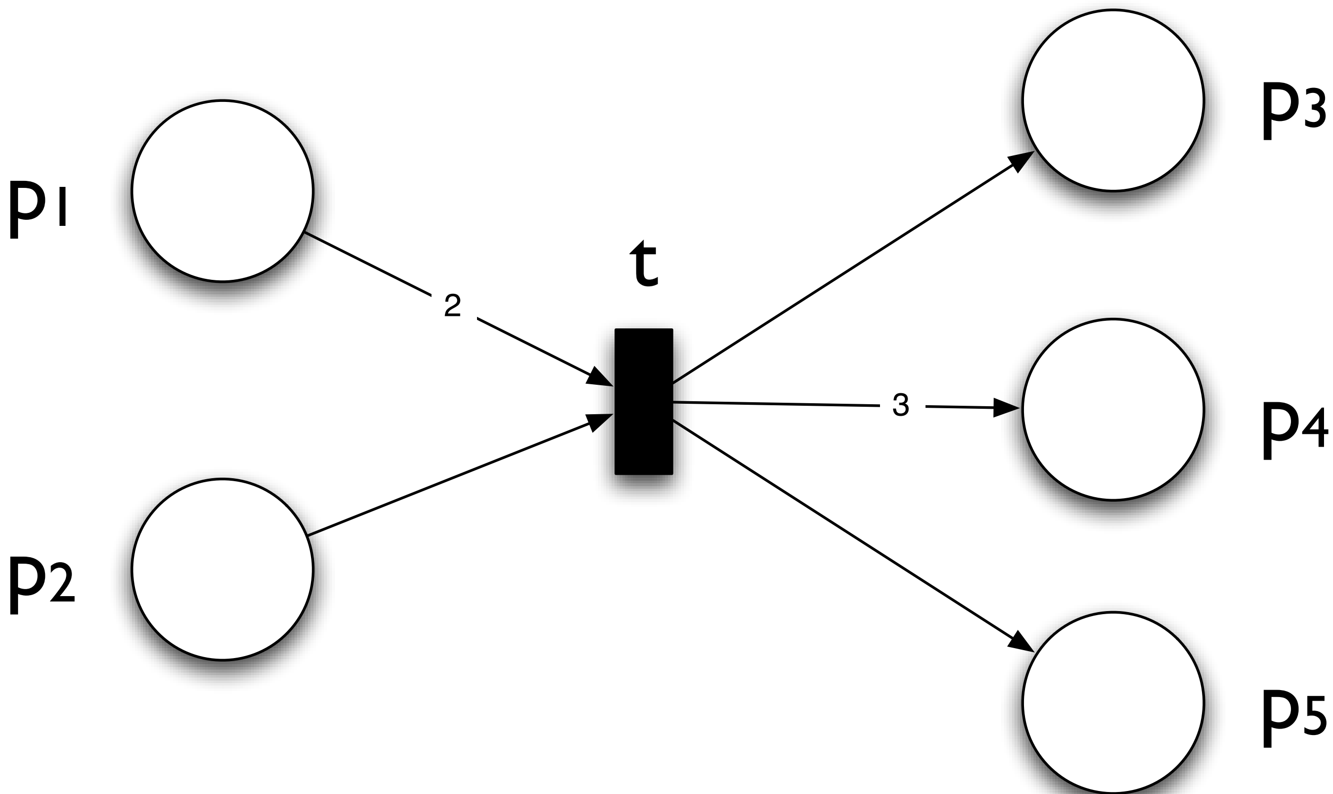
# Formal definition

- A Petri net is a tuple $\langle P, T \rangle$ where:
  - P is the (finite) set of places
  - T is the (finite) set of transitions. Each transition t is a tuple $\langle I, O \rangle$ where:
    - I: is a function s.t. t consumes I(p) tokens in each place p
    - O is a function s.t. t produces O(p) tokens in each place p

# Example

$I(p_1)=2$ $I(p_2)=1$ $I(p_3)=0$ $I(p_4)=0$ $I(p_5)=0$

$O(p_1)=0$ $O(p_2)=0$ $O(p_3)=1$ $O(p_4)=3$ $O(p_5)=1$



# Markings

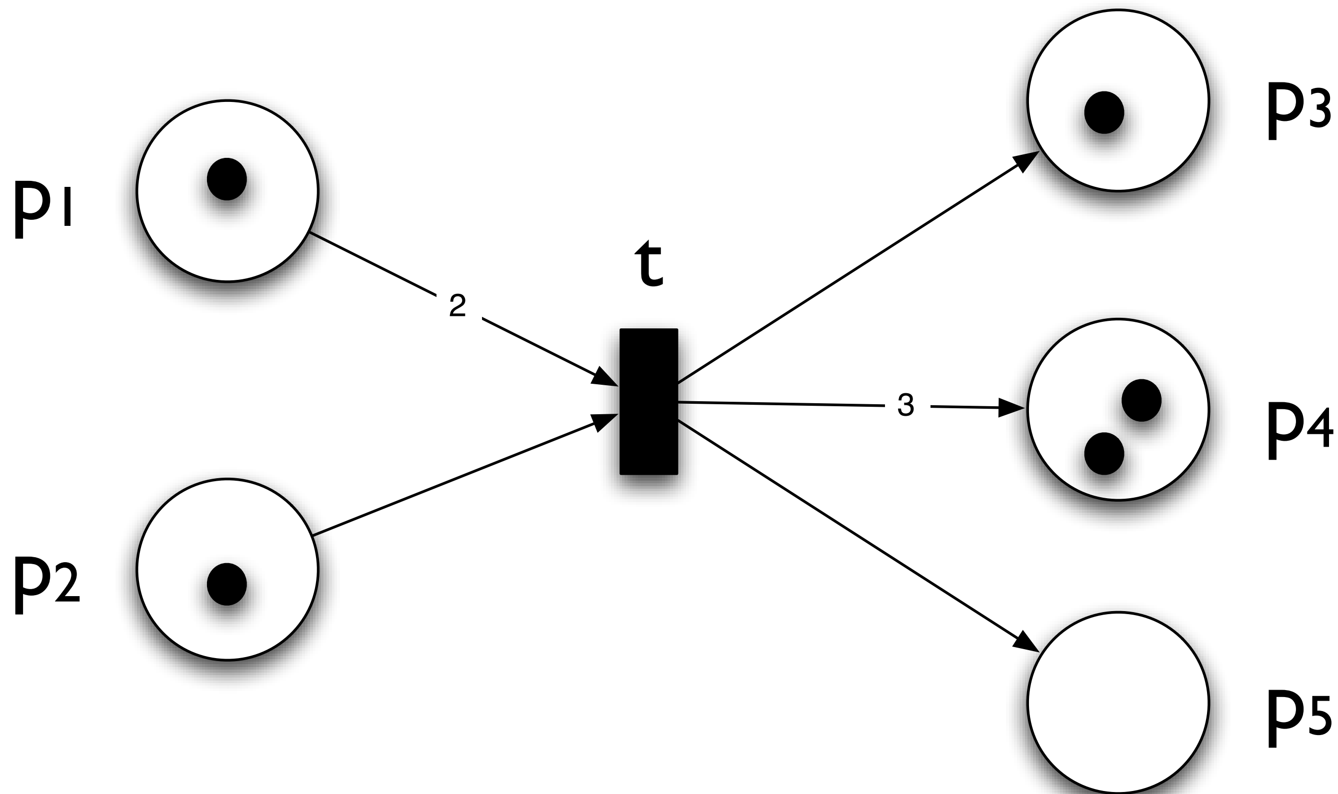
- The distribution of the tokens in the places is formalised by the notion of marking, which can be seen:
  - either as a function $m$, s.t. $m(p)$ is the number of tokens in place $p$
  - or as a vector $m = \langle m_1, m_2, \ldots m_n \rangle$ where $m_i$ is the number of tokens in place $p_i$

# Example

$$m = \langle 1,1,1,2,0 \rangle$$

$$m = \langle p_1, p_2, p_3, 2p_4 \rangle$$

$$m(p_1)=1, m(p_2)=1, m(p_3)=1, m(p_4)=2, m(p_5)=0$$



# Firing a transition

- A transition t = $\langle I, O \rangle$ can be fired from m iff for any place p:

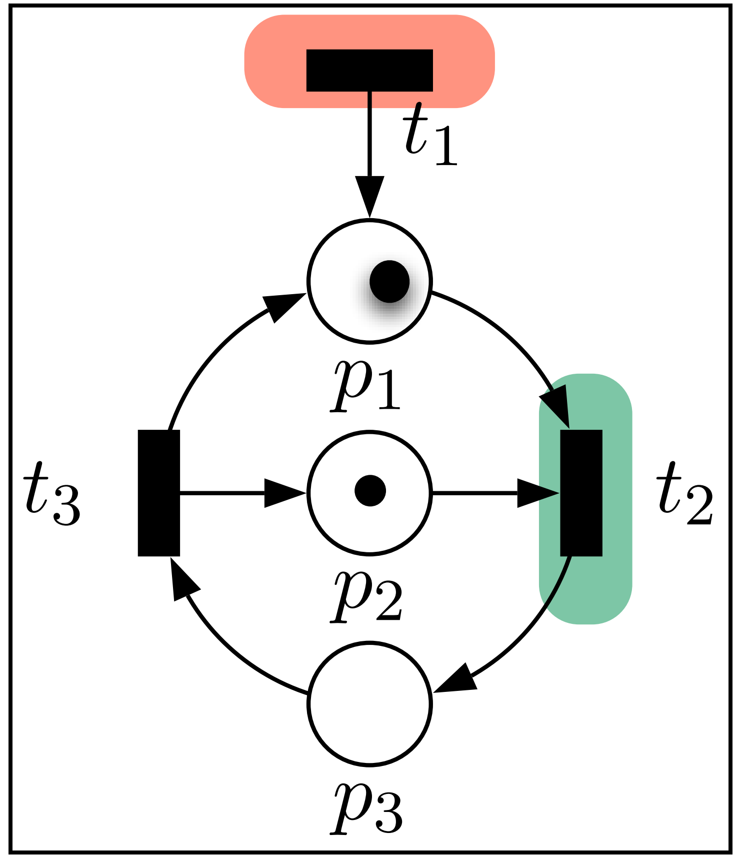
$$m(p) \geq I(p)$$

- The firing transforms the marking m into a marking m' s.t. for any place p:

$$m'(p) = m(p) - I(p) + O(p)$$

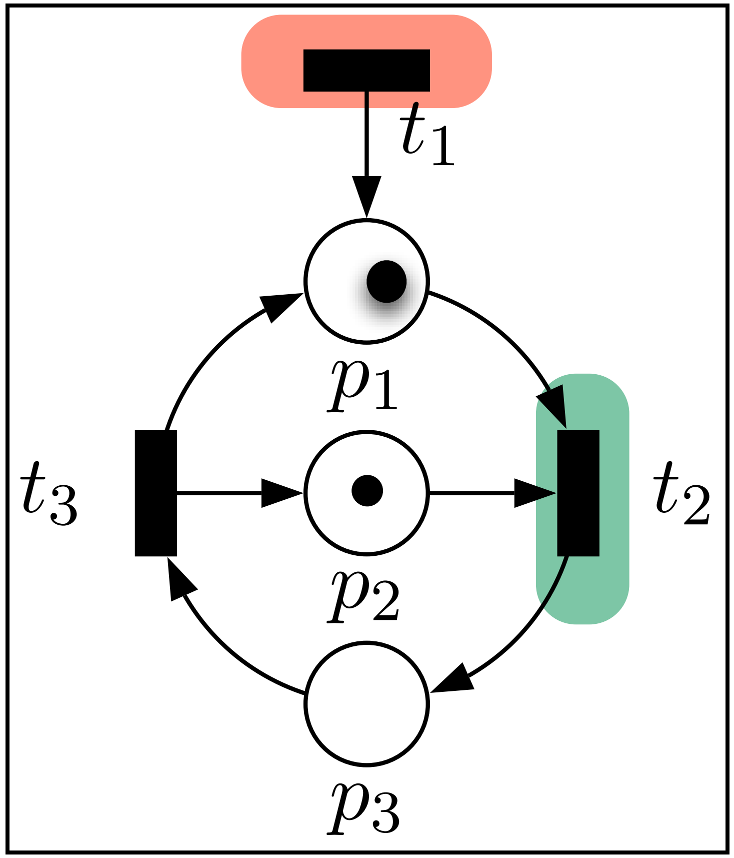
- Notation: $m \rightarrow m'$
- Notation: $Post(m) = \{m' \mid m \rightarrow m'\}$

# Example



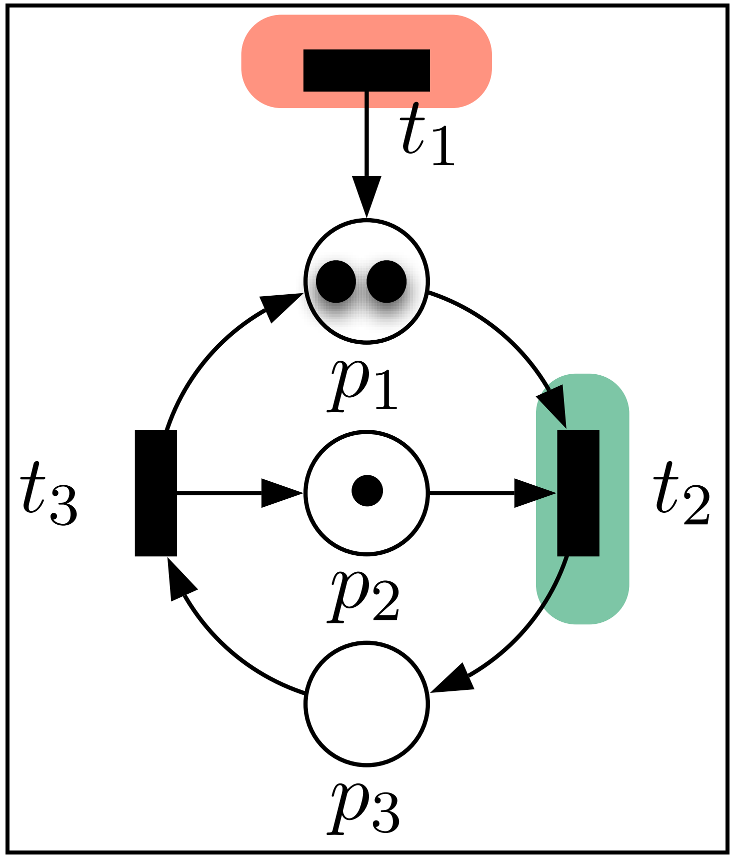
# Example

Post( $\langle 1, 1, 0 \rangle$ )=
{ $\langle 2, 1, 0 \rangle$ , $\langle 0, 0, 1 \rangle$ }



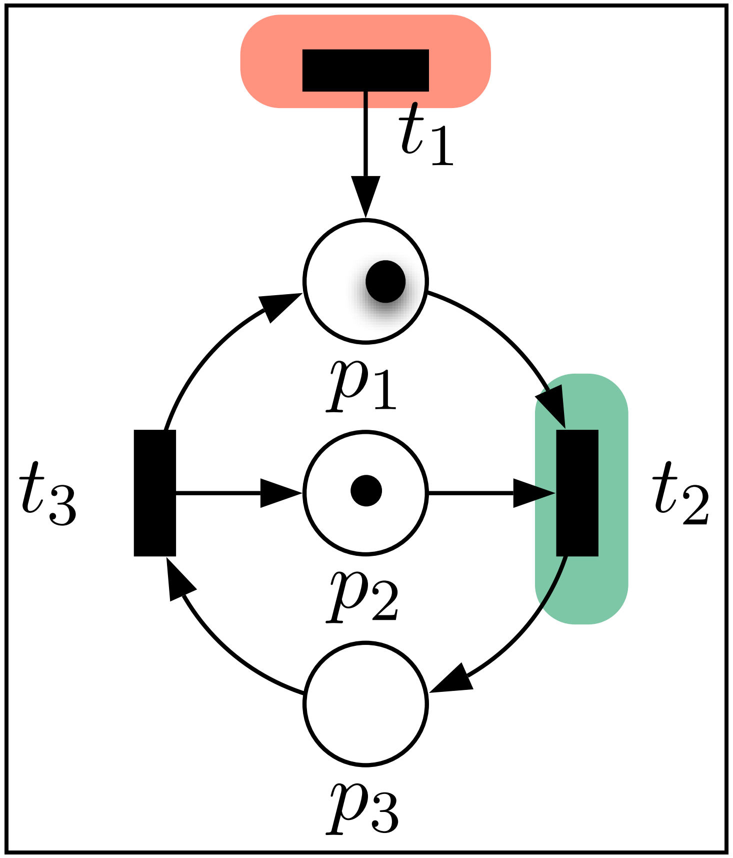
# Example

Post( $\langle 1, 1, 0 \rangle$ )=
{ $\langle 2, 1, 0 \rangle$ , $\langle 0, 0, 1 \rangle$ }



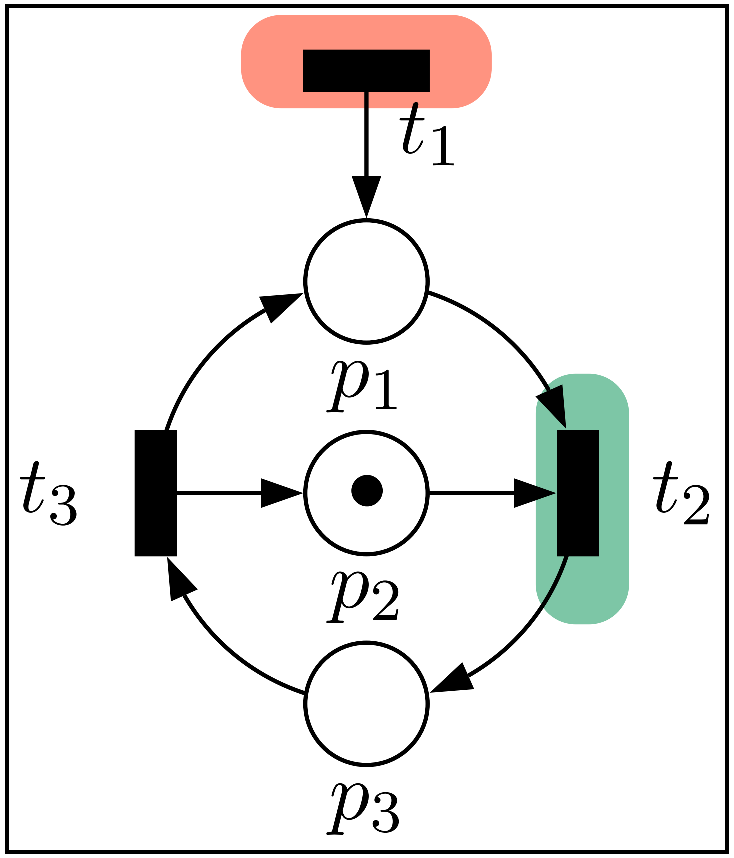
# Example

Post( $\langle 1, 1, 0 \rangle$ )=
{ $\langle 2, 1, 0 \rangle$ , $\langle 0, 0, 1 \rangle$ }



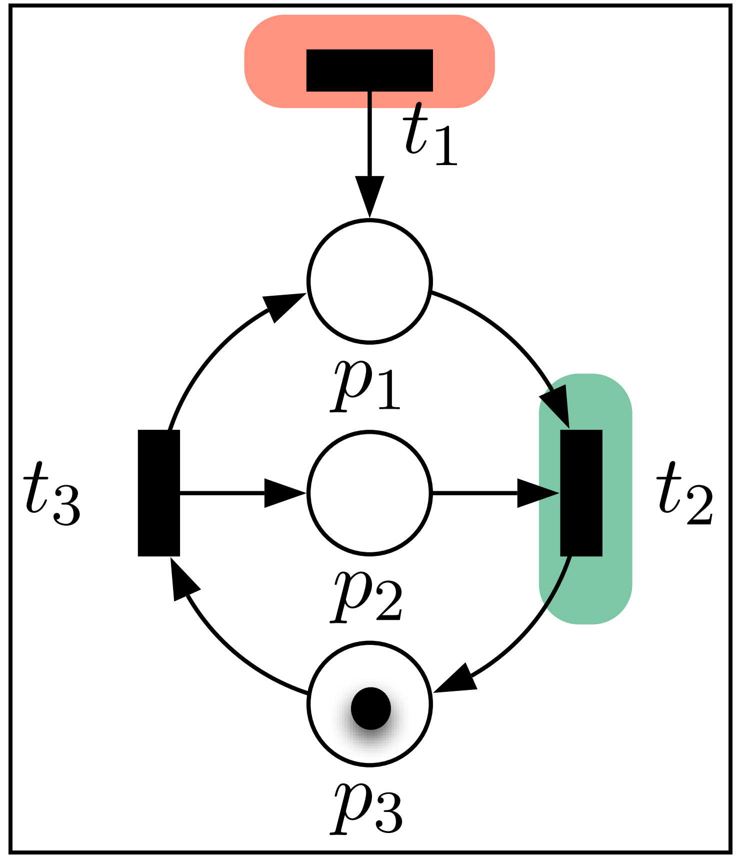
# Example

Post( $\langle 1, 1, 0 \rangle$ )=
{ $\langle 2, 1, 0 \rangle$ , $\langle 0, 0, 1 \rangle$ }



# Example

$$\text{Post}(\ \langle 1,1,0\rangle\ ) = \{\ \langle 2,1,0\rangle\ ,\ \langle 0,0,1\rangle\ \}$$



# Initial marking
# Reachable markings

- All PN are equipped with an initial marking $m_0$
- If two markings m and m' are s.t.:

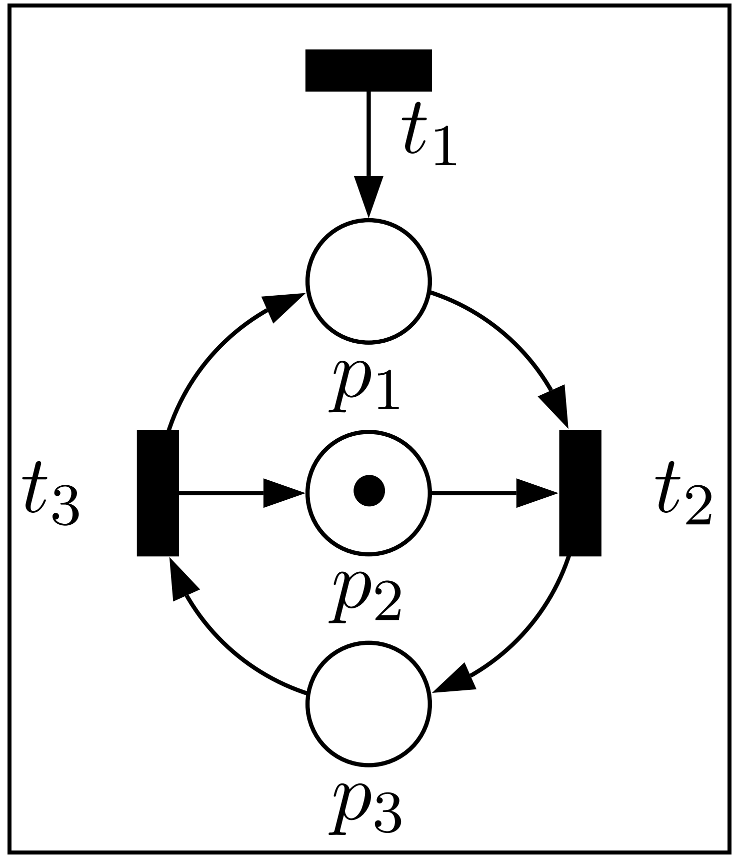
  $$m \rightarrow m_1 \rightarrow m_2 \rightarrow \cdots \rightarrow m'$$

  Then m' is reachable from m
- Let N be a PN with initial marking $m_0$:

  $$\text{Reach}(N) = \{m \text{ reachable from } m_0\}$$

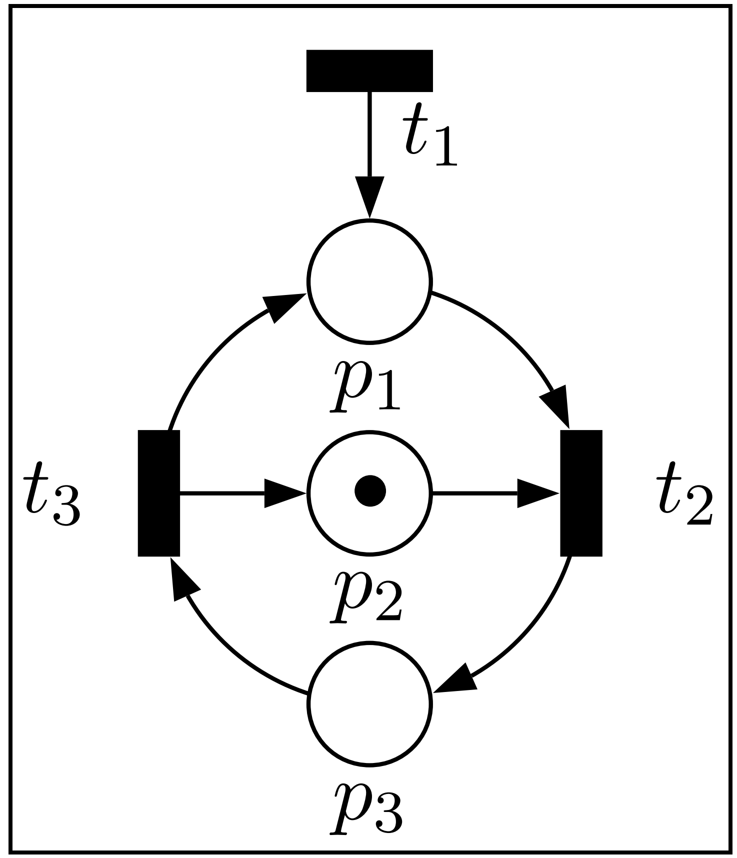
  is the set of reachable markings of N.

# Example



# Example

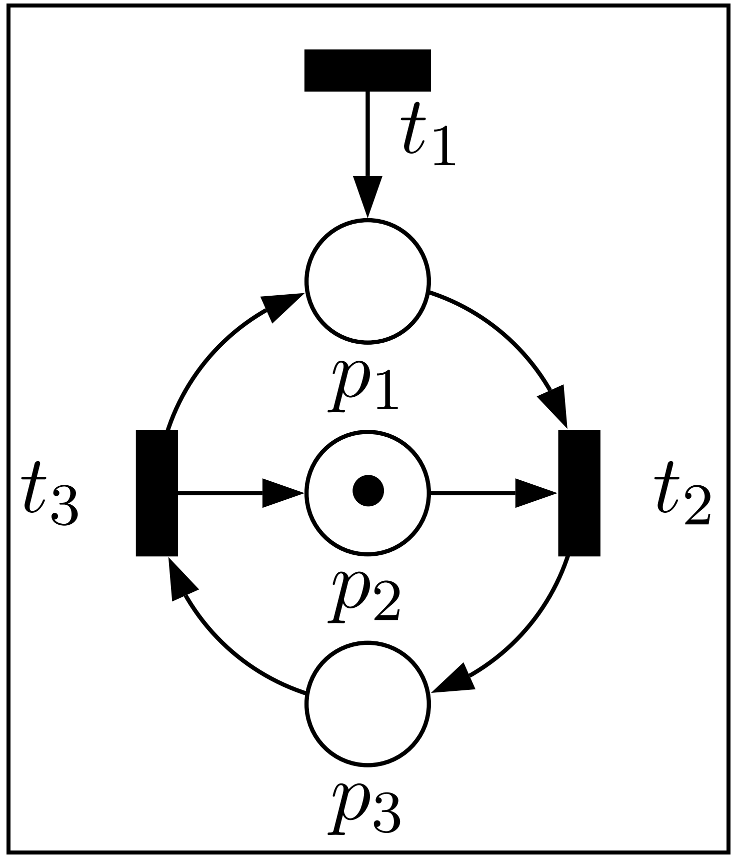
$$\mathrm{Reach}(\mathcal{N}) = \{\langle i,1,0\rangle \mid i \in \mathbb{N}\} \cup \{\langle i,0,1\rangle \mid i \in \mathbb{N}\}$$



# Example

$$\mathrm{Reach}(\mathcal{N}) = \{\langle i, 1, 0\rangle \mid i \in \mathbb{N}\} \cup \{\langle i, 0, 1\rangle \mid i \in \mathbb{N}\}$$

This set allows us to prove that the mutual exclusion is indeed enforced



# Ordering on markings

- Markings can be compared thanks to $\preccurlyeq$:

  $m \preccurlyeq m'$ iff for any place p: $m(p) \leqslant m'(p)$

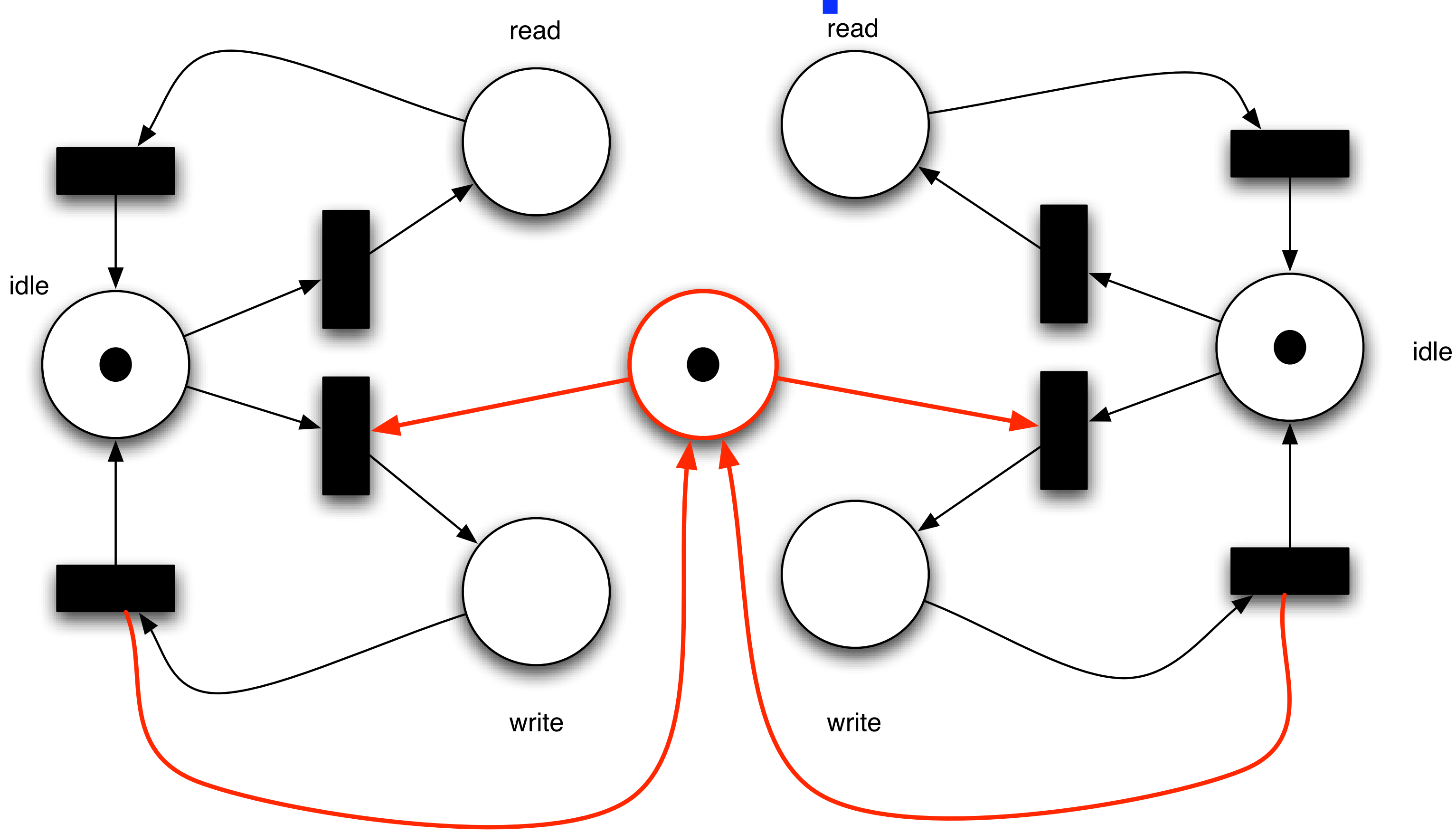
  $m \prec m'$ iff $m \preccurlyeq m'$ and $m \neq m'$

- Examples:
  - $\langle 1, 0, 0 \rangle \prec \langle 1, 1, 0 \rangle \preccurlyeq \langle 1, 1, 0 \rangle \preccurlyeq \langle 5, 7, 2 \rangle$
  - $\langle 1, 0, 0 \rangle$ is not comparable to $\langle 0, 1, 0 \rangle$

# Questions on PN

- Meaningful questions about PN include:
  - Boundedness: is the number of reachable markings bounded ?
  - Place boundedness: is there a bound on the maximal number of tokens that can be created in a given place ?
  - Semi-liveness: is there a reachable marking from which a given transition can fire ?
  - Coverability

# Example



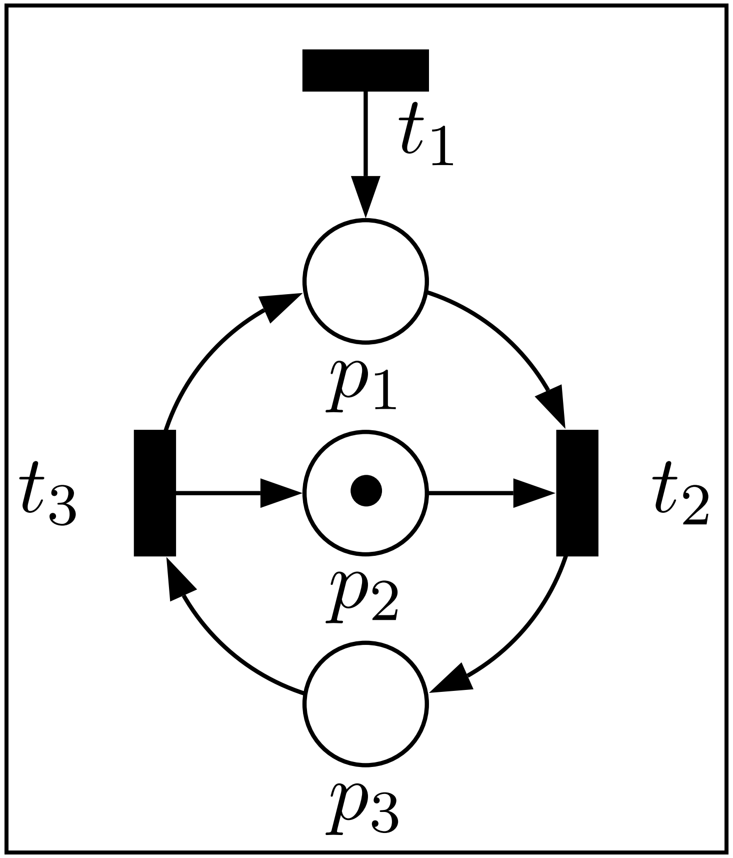

Bounded PN

All the places are bounded

All the transitions are semi-live

# Example

- Unbounded PN
- $p_2$ and $p_3$ are bounded
- $p_1$ is unbounded
- All the transitions are semi-live



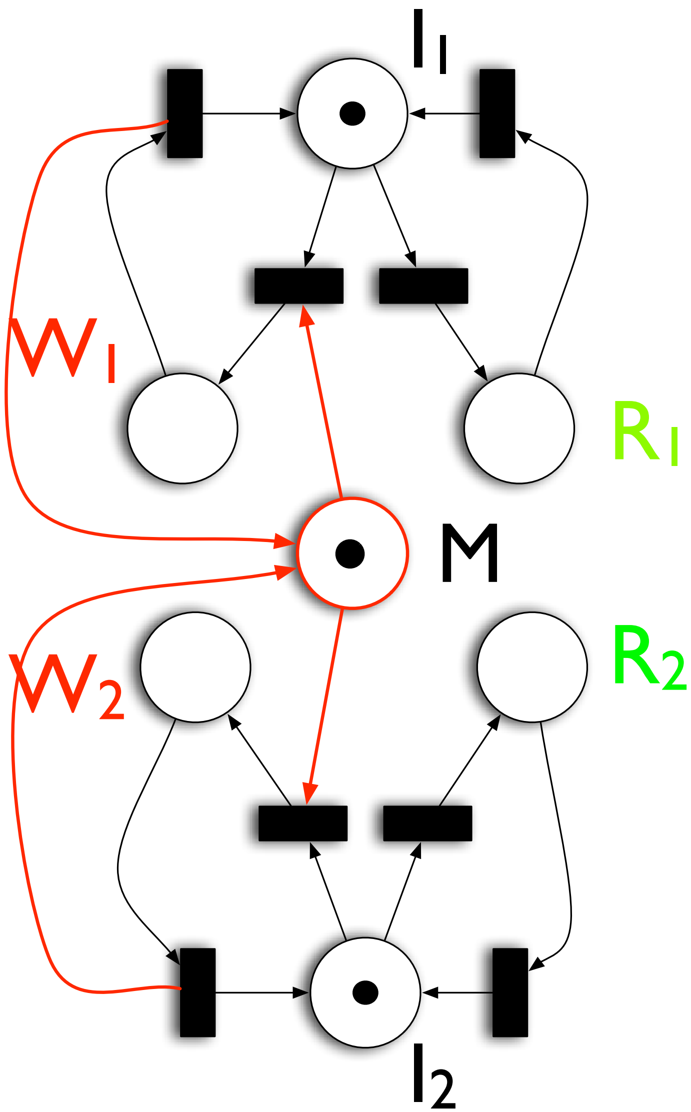
# Some tools for the analysis of PN

# Reachability tree and reachability graph

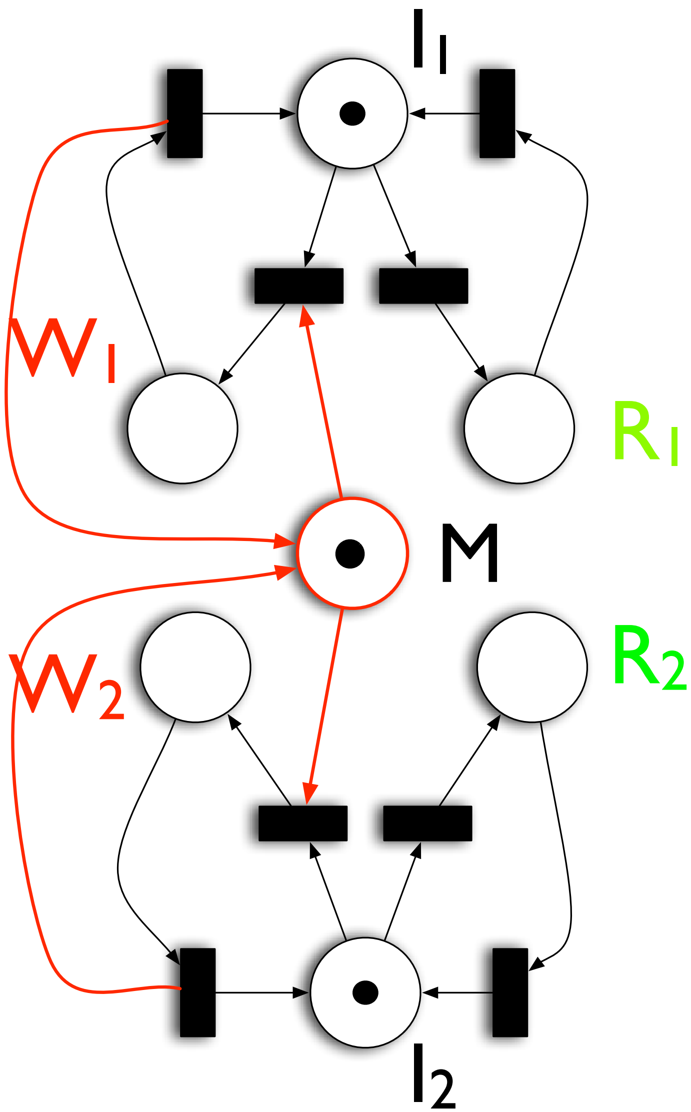
# Reachability Tree

- Idea:
  - the root is labeled by $m_0$
  - for each node labeled by m, create one child for each marking of Post(m)

# Reachability Tree

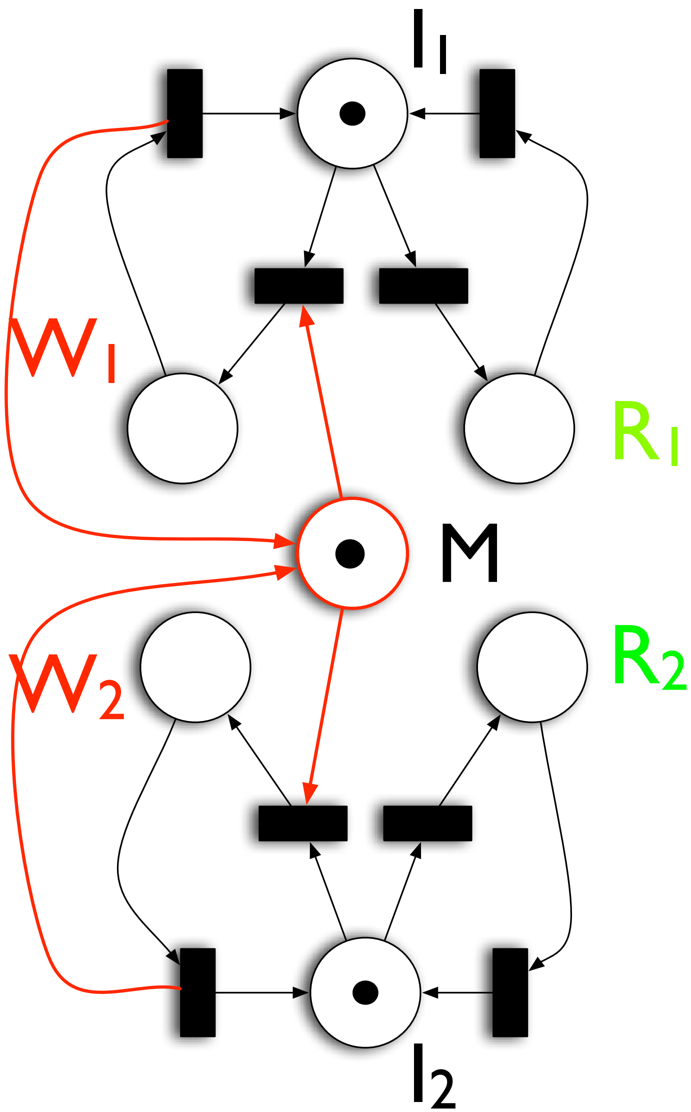


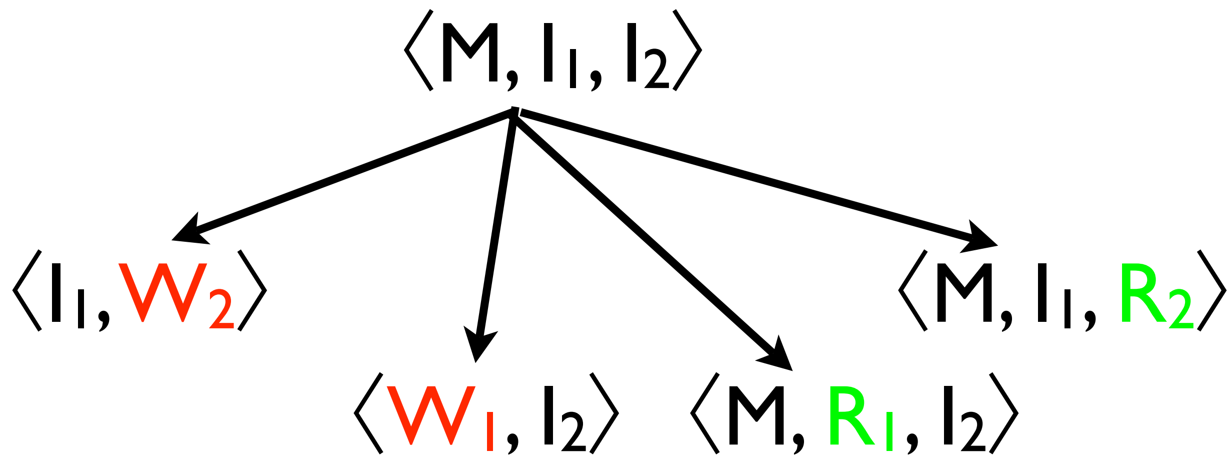
# Reachability Tree



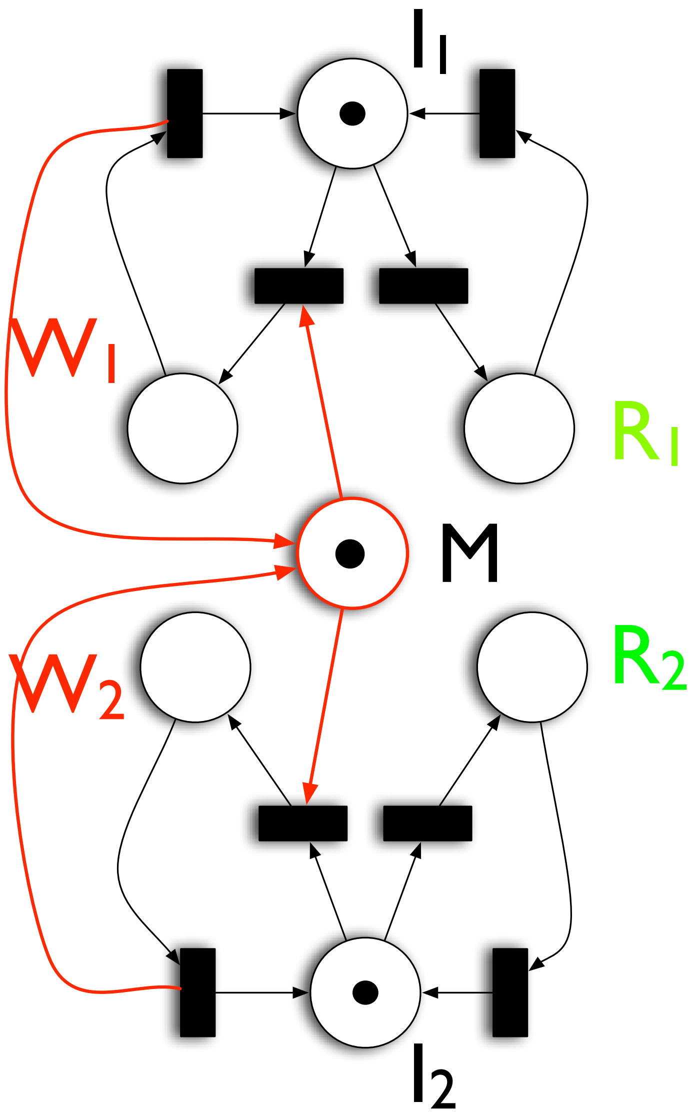

$\langle M, I_1, I_2 \rangle$

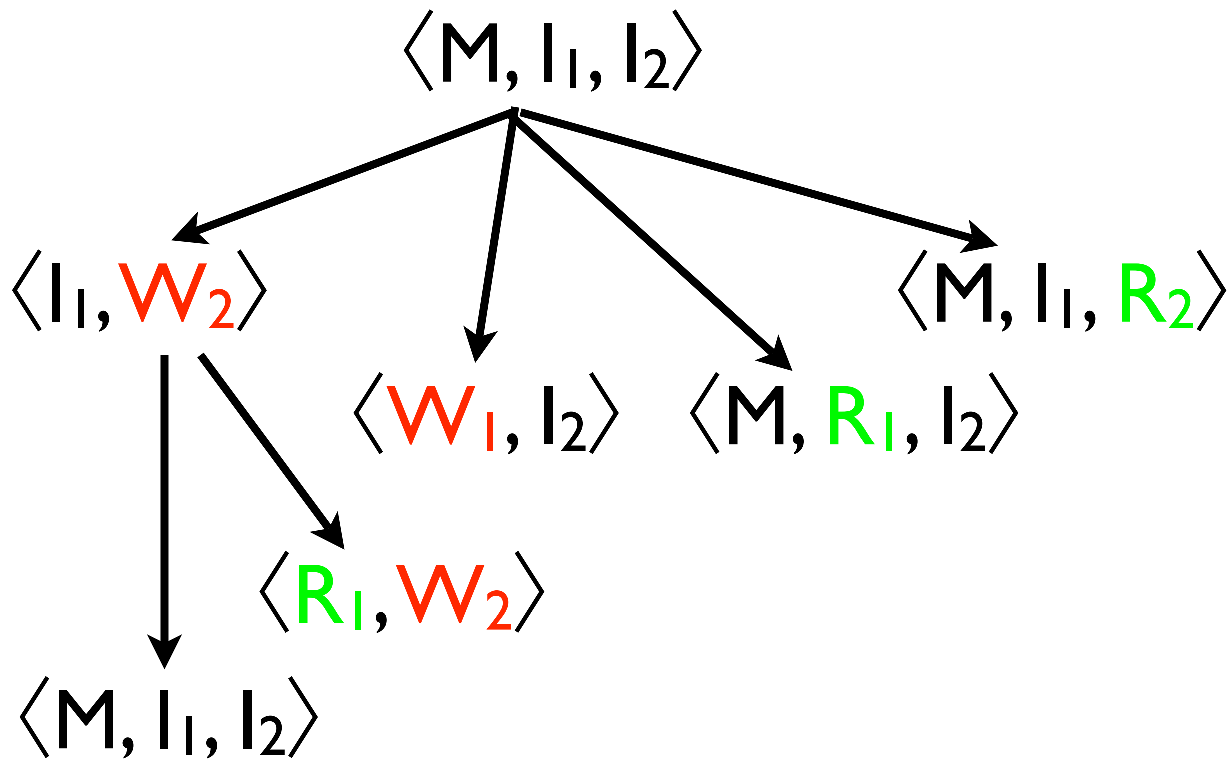
# Reachability Tree



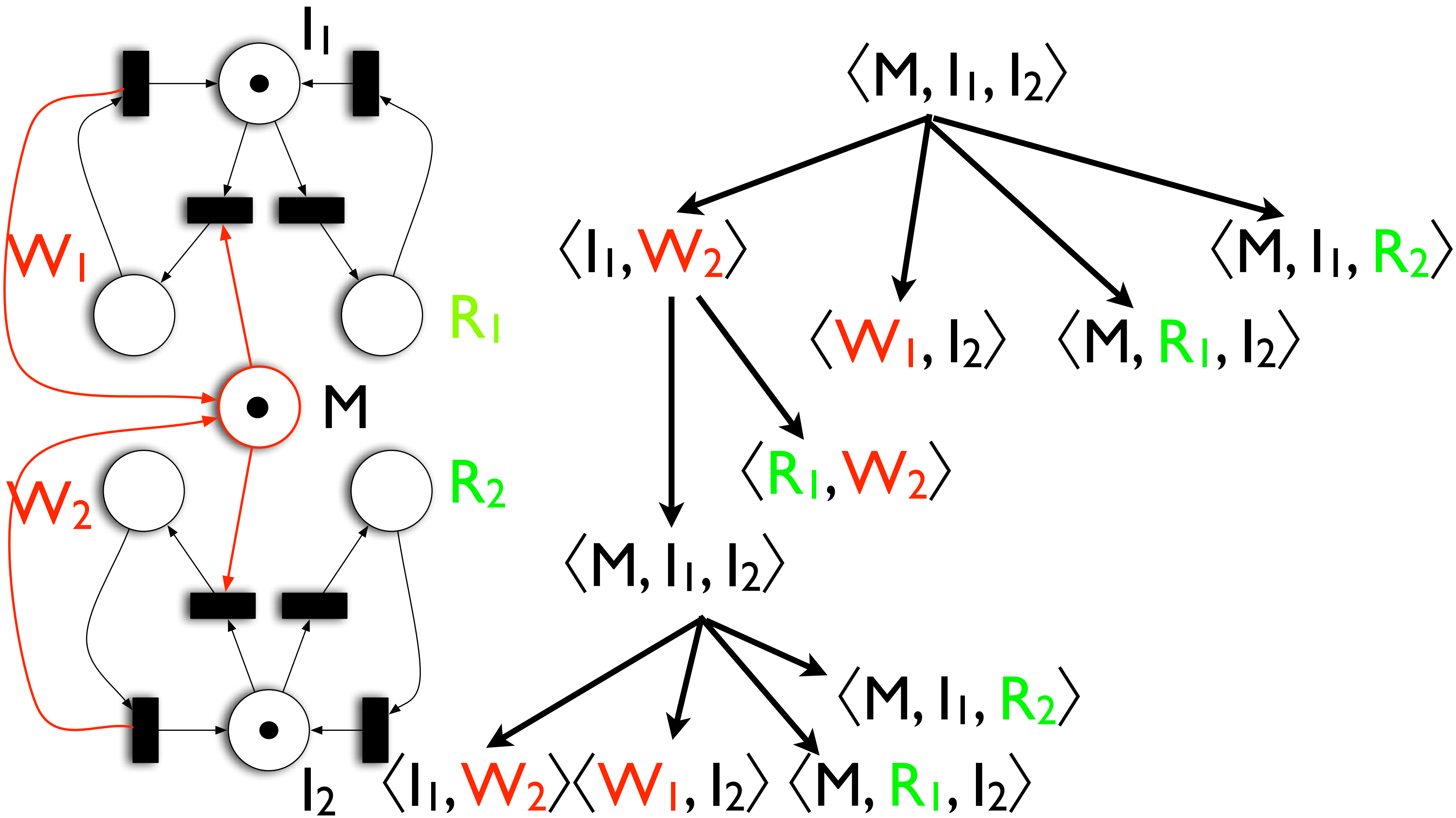



# Reachability Tree

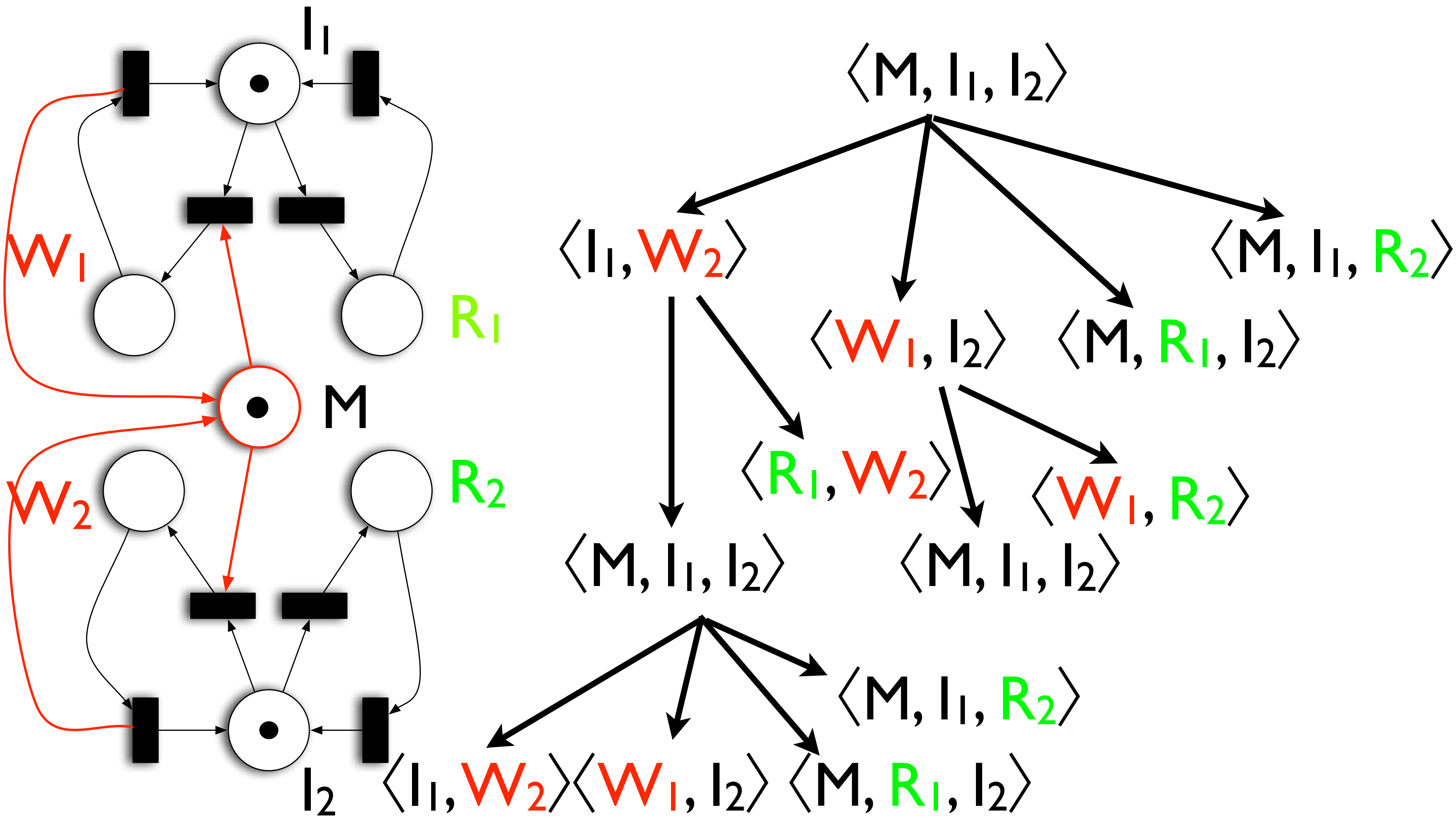


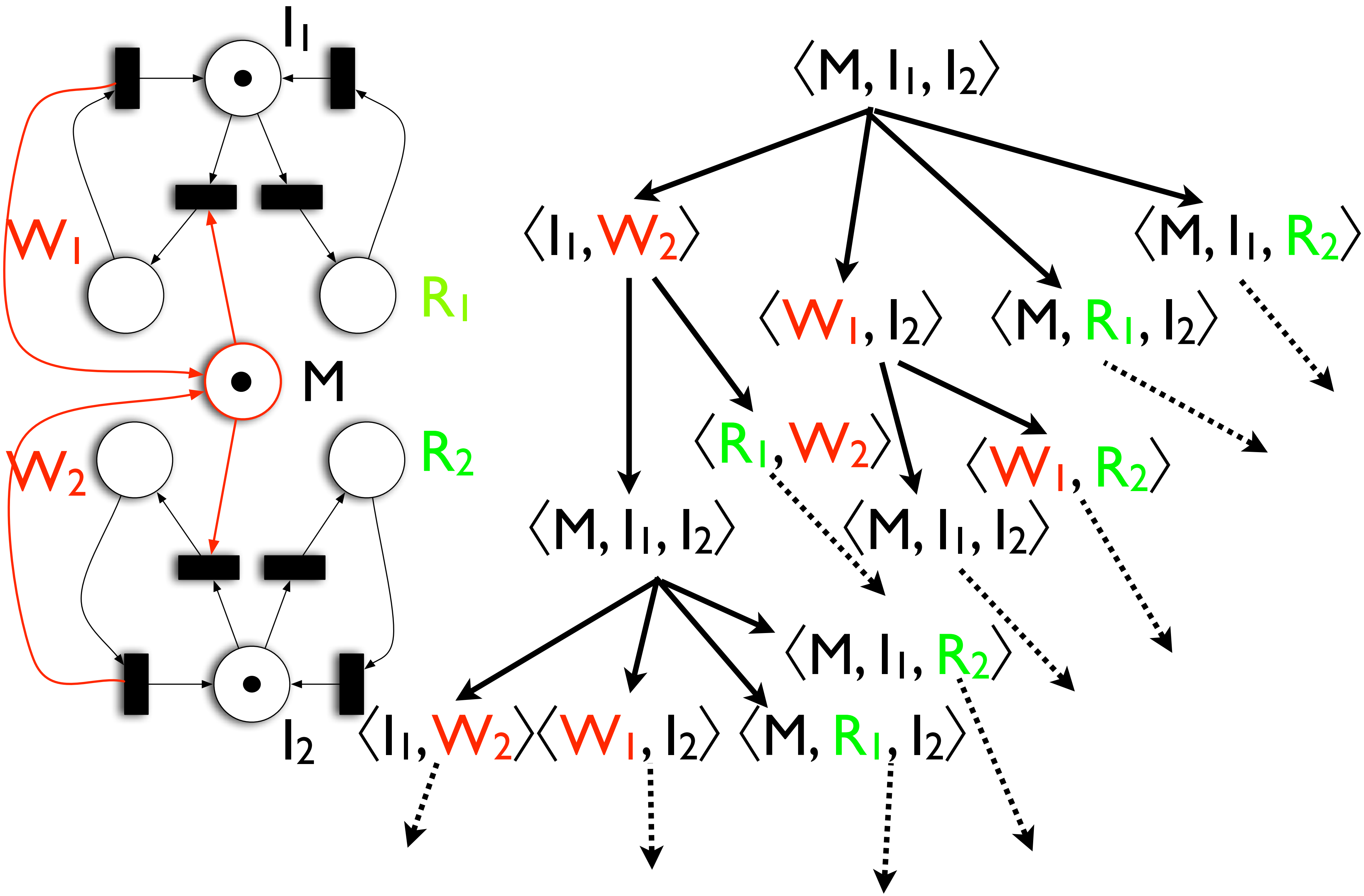
# Reachability Tree



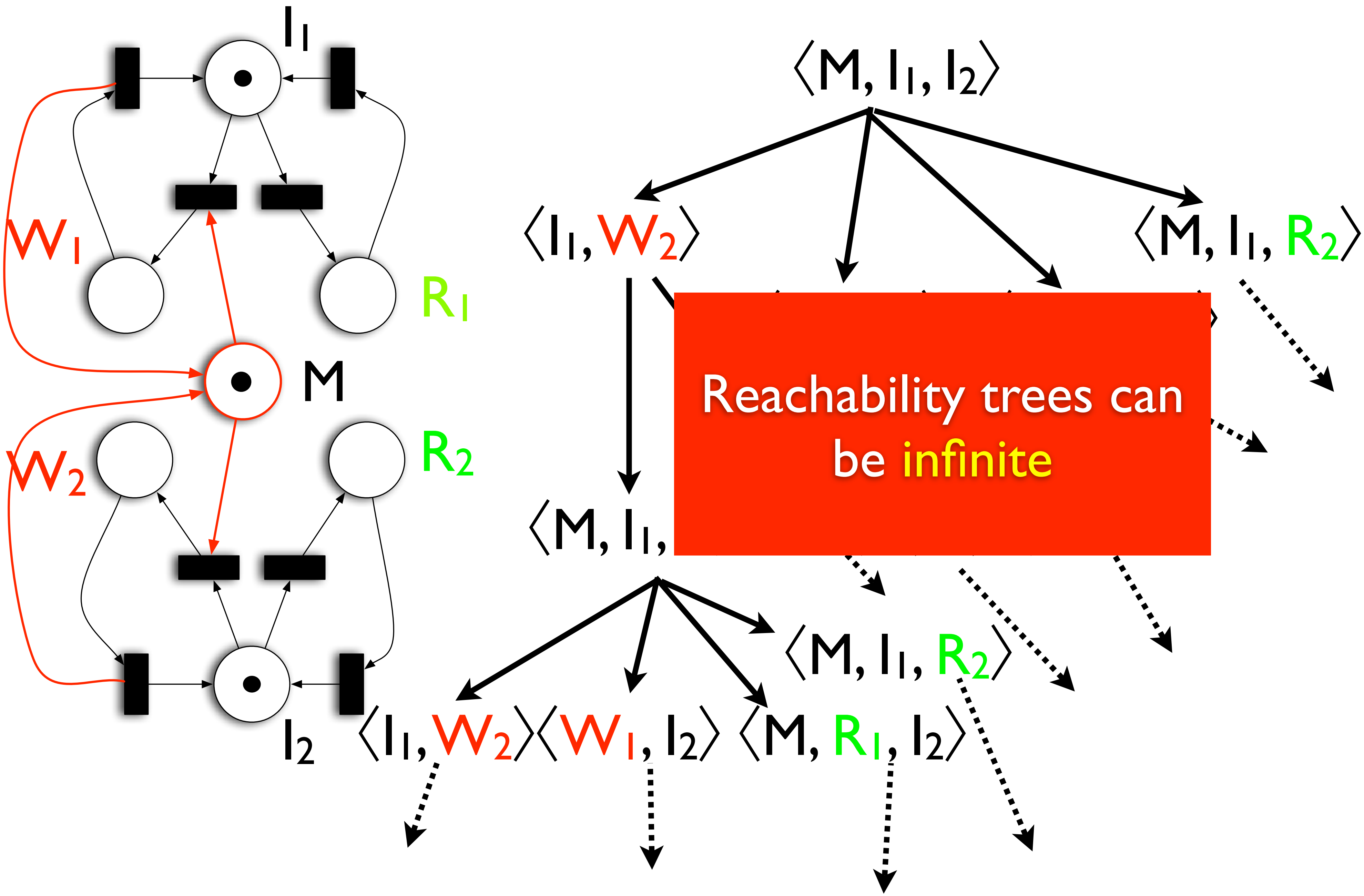
# Reachability Tree



# Reachability Tree



# Reachability Tree

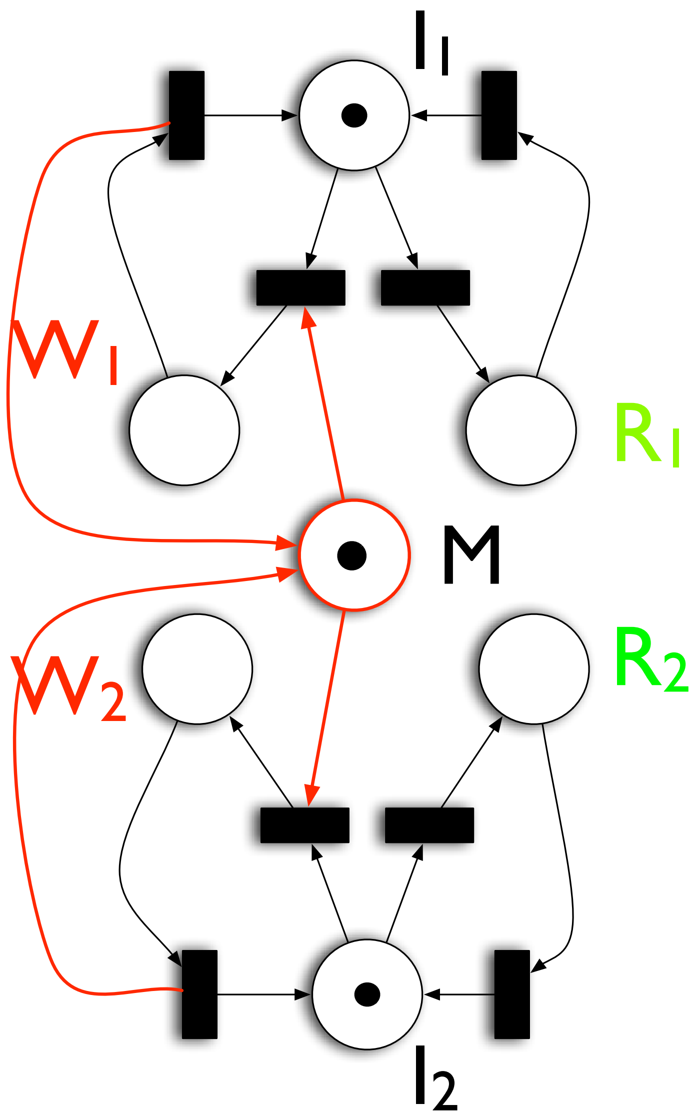



Reachability trees can be infinite

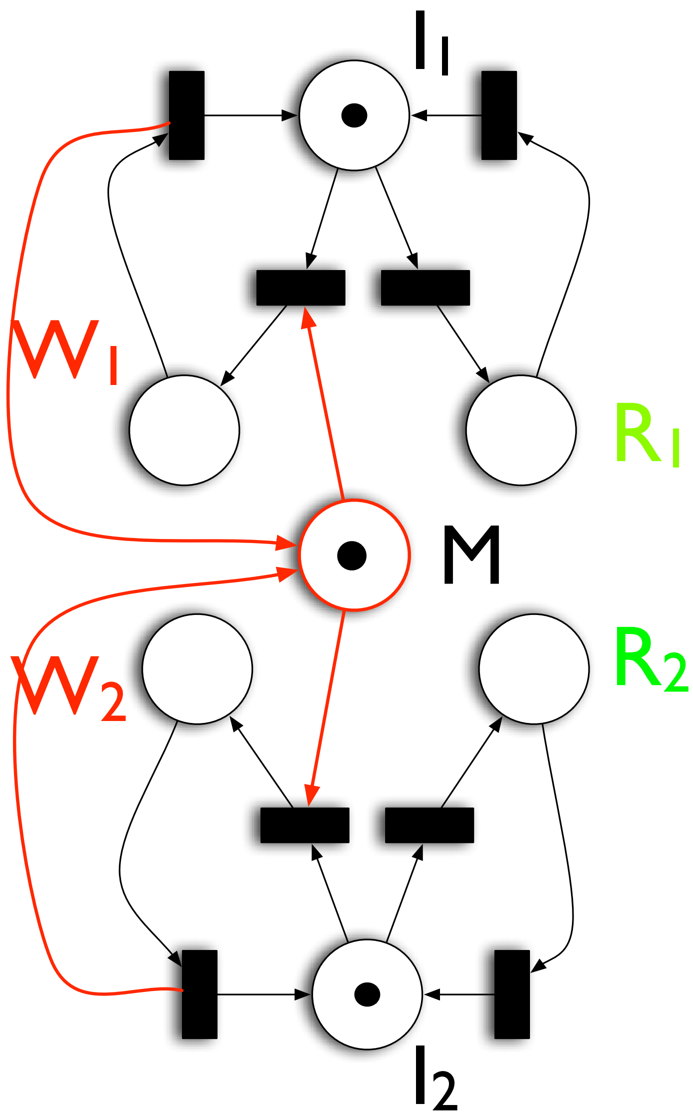
# Reachability graph

- Idea: build a node for each reachable marking and add an edge from m to m' if some transition transforms m into m'
  - remark: now, if we meet the same marking twice, we do not create a new node, but re-use the previously created node.

# Reachability graph

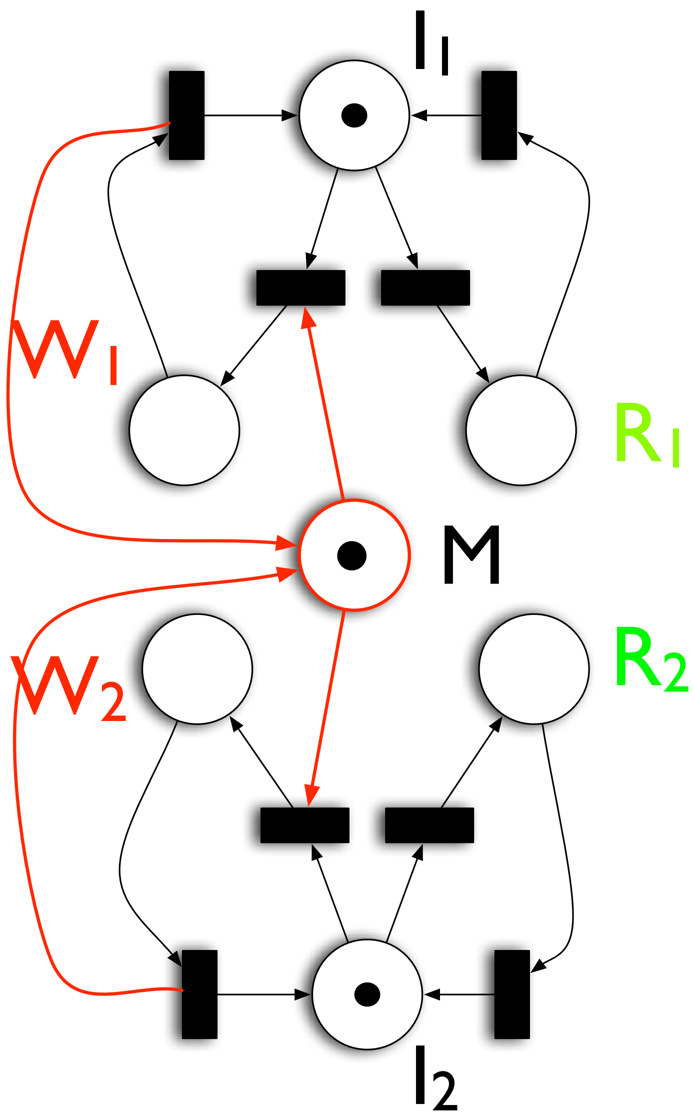


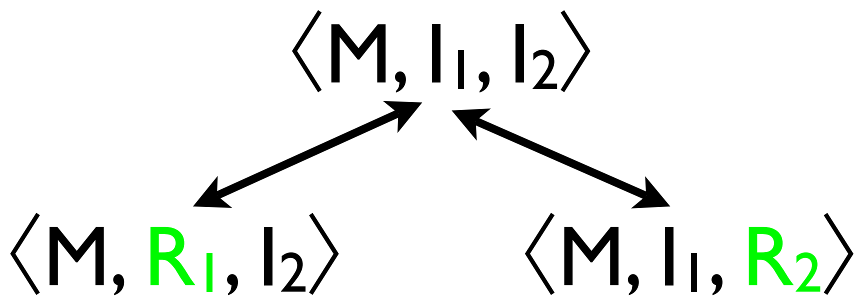
# Reachability graph



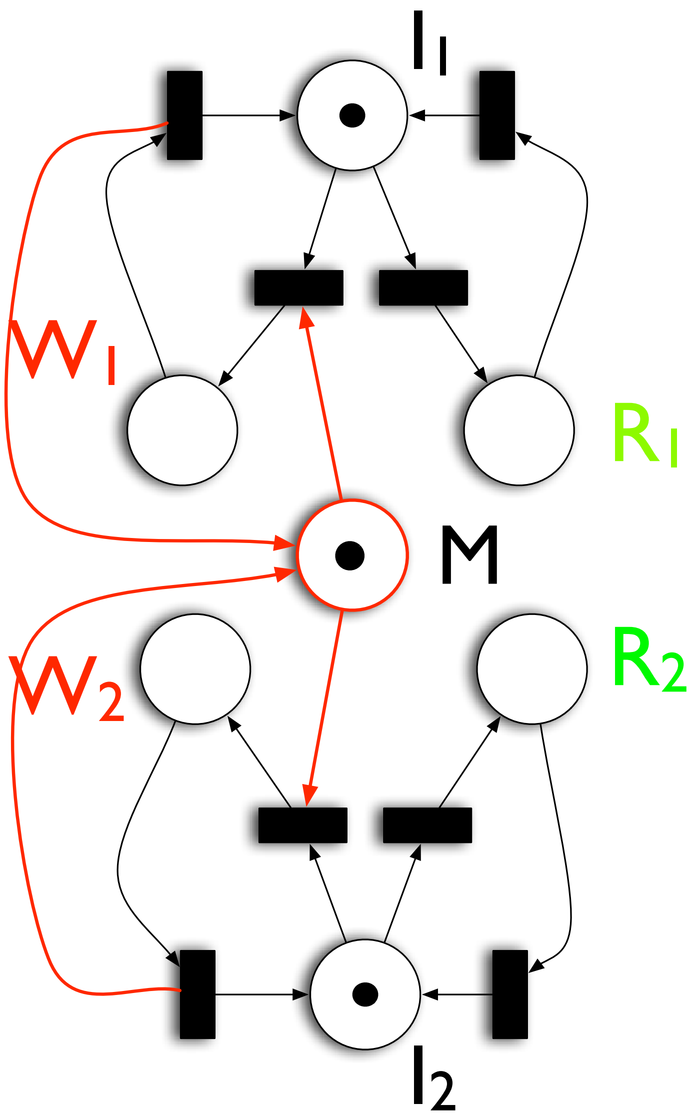

$\langle M, I_1, I_2 \rangle$

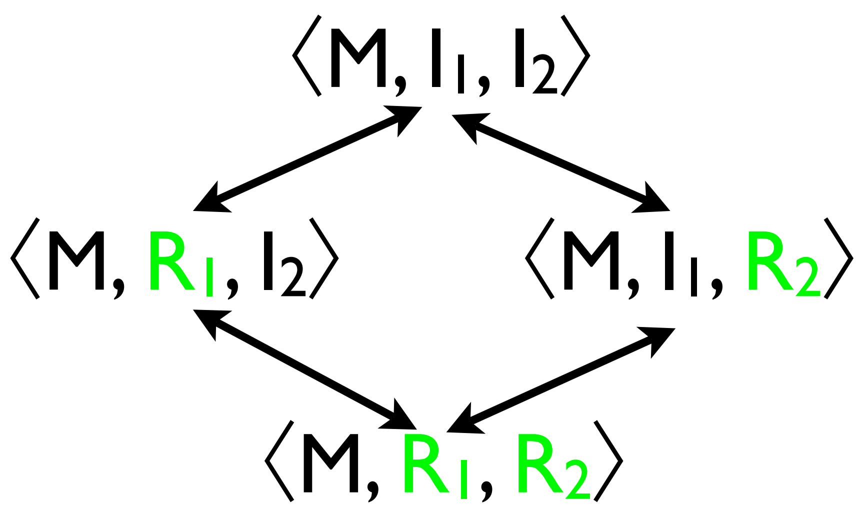
# Reachability graph



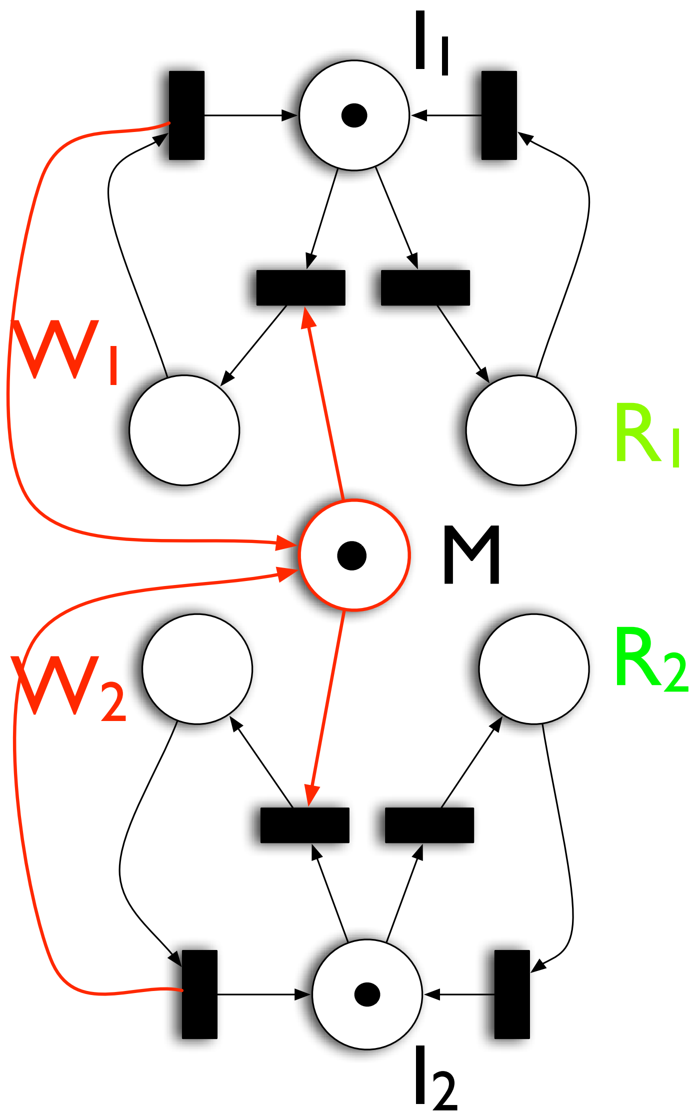



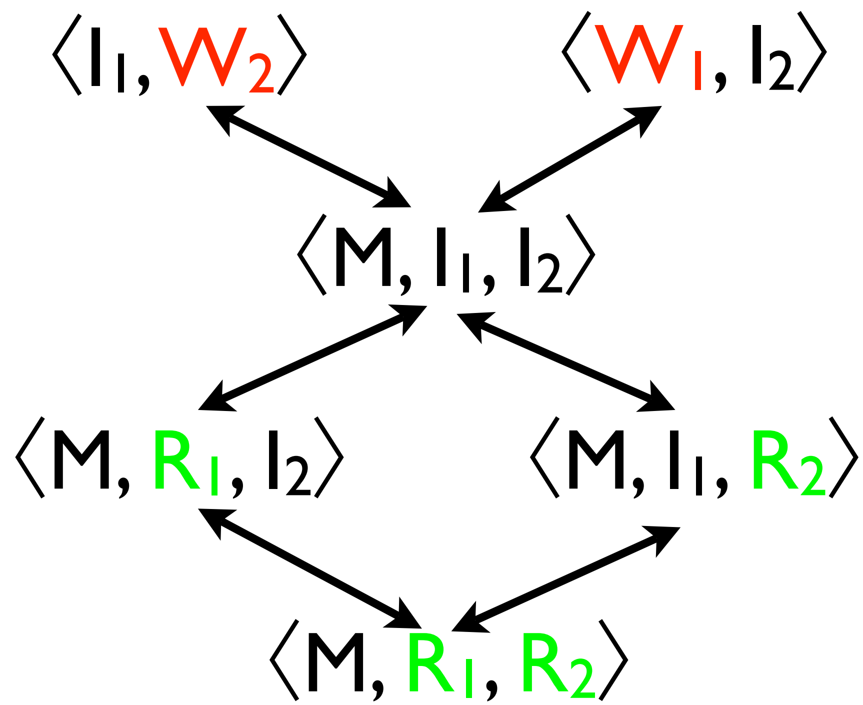
# Reachability graph



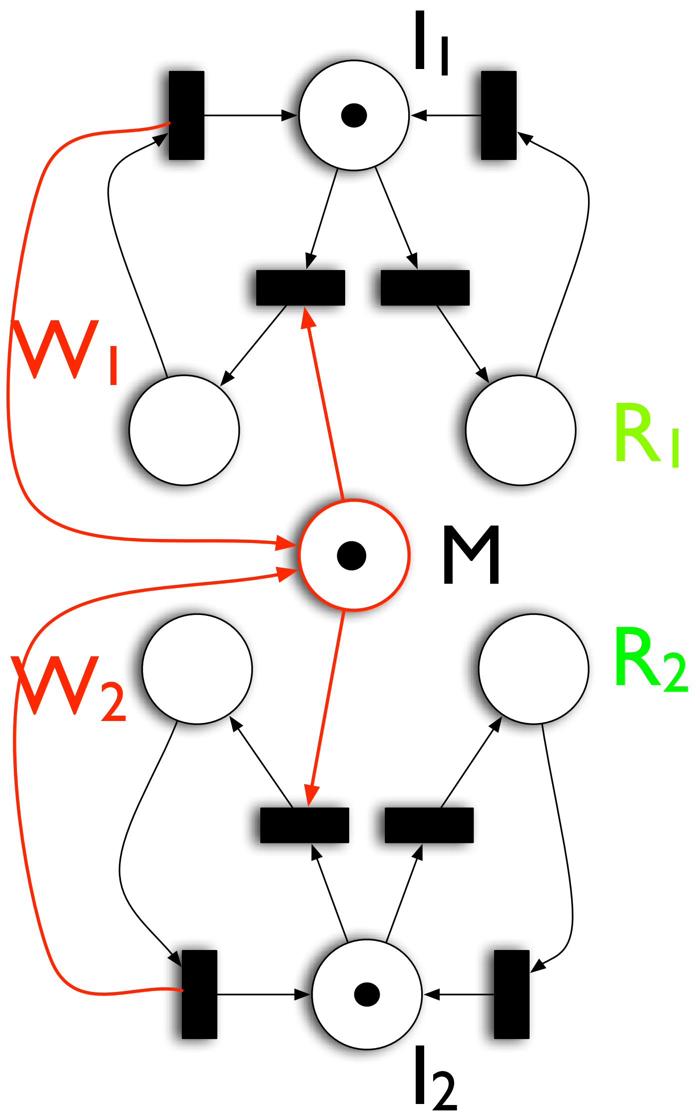



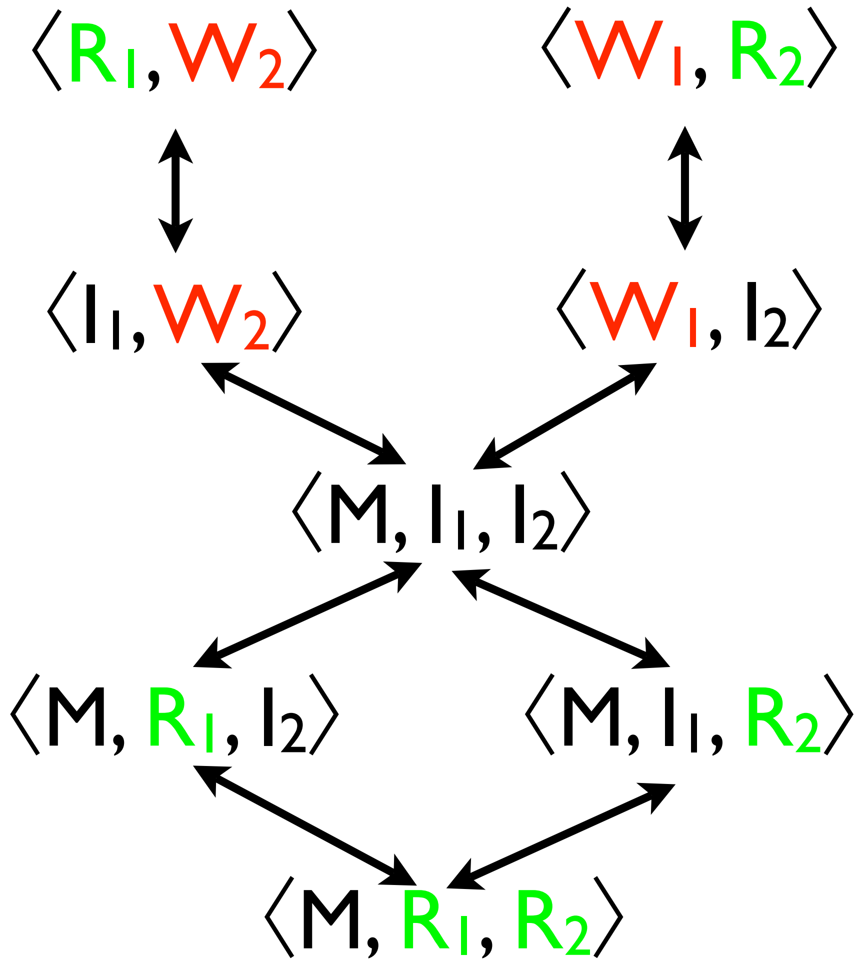
# Reachability graph



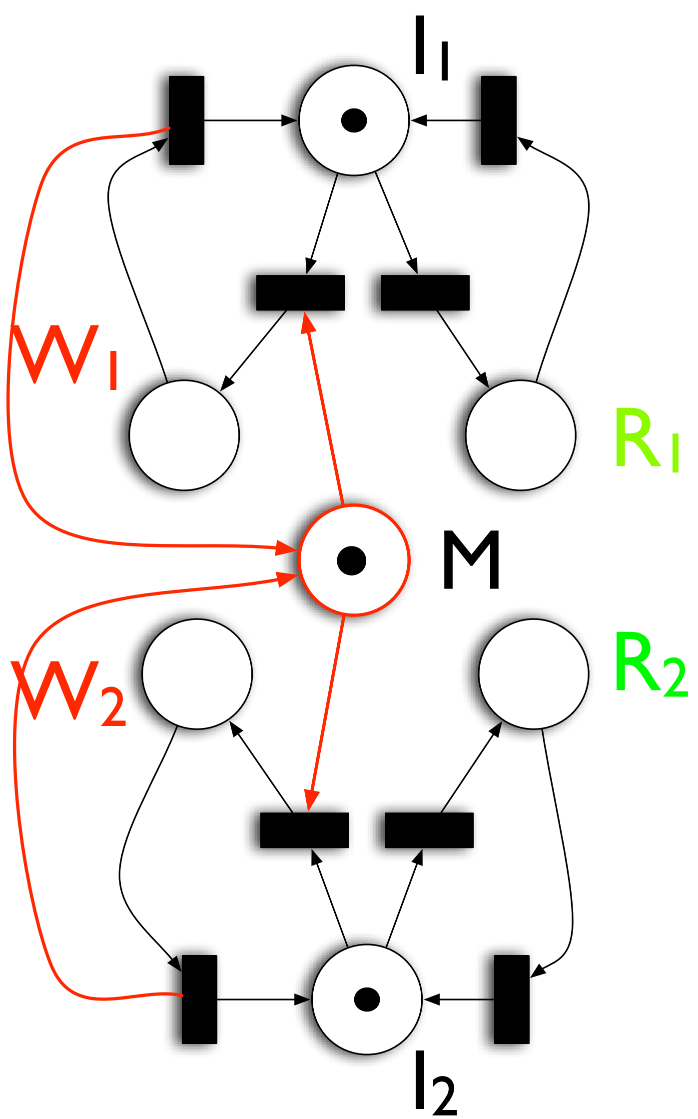



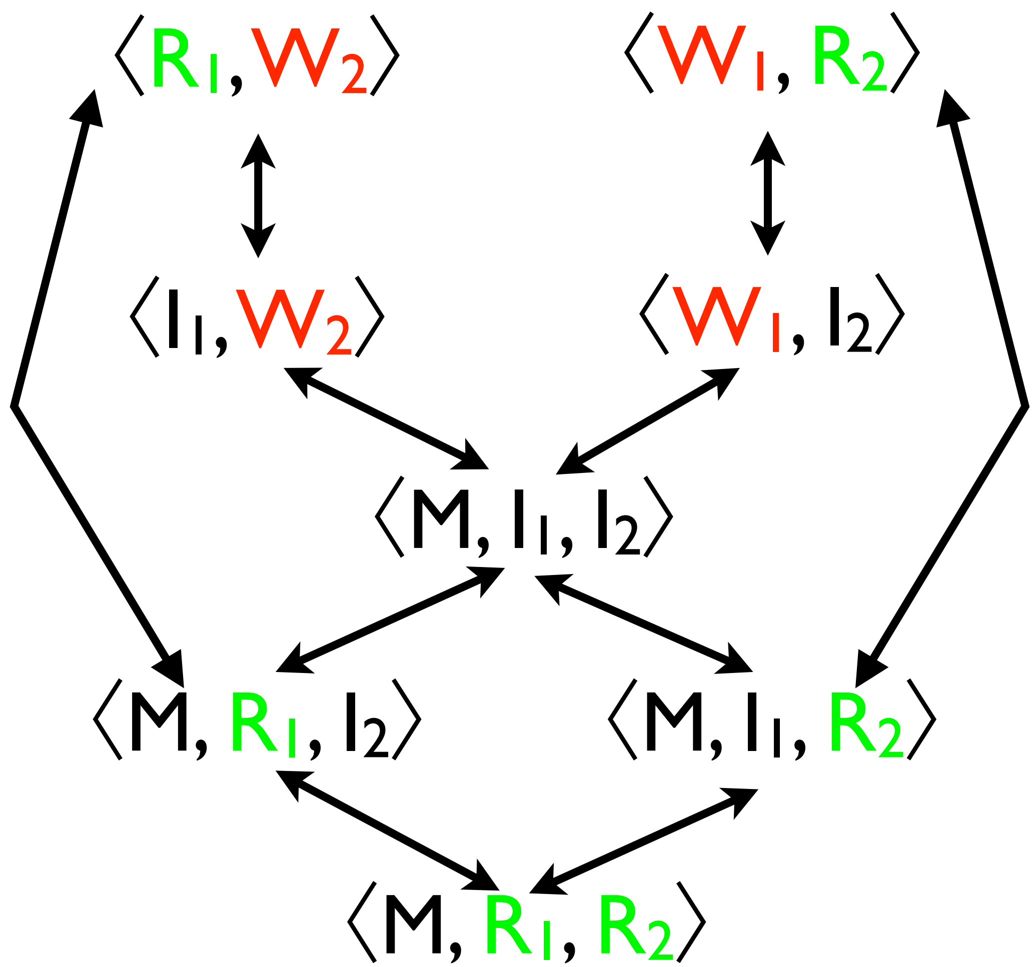
# Reachability graph



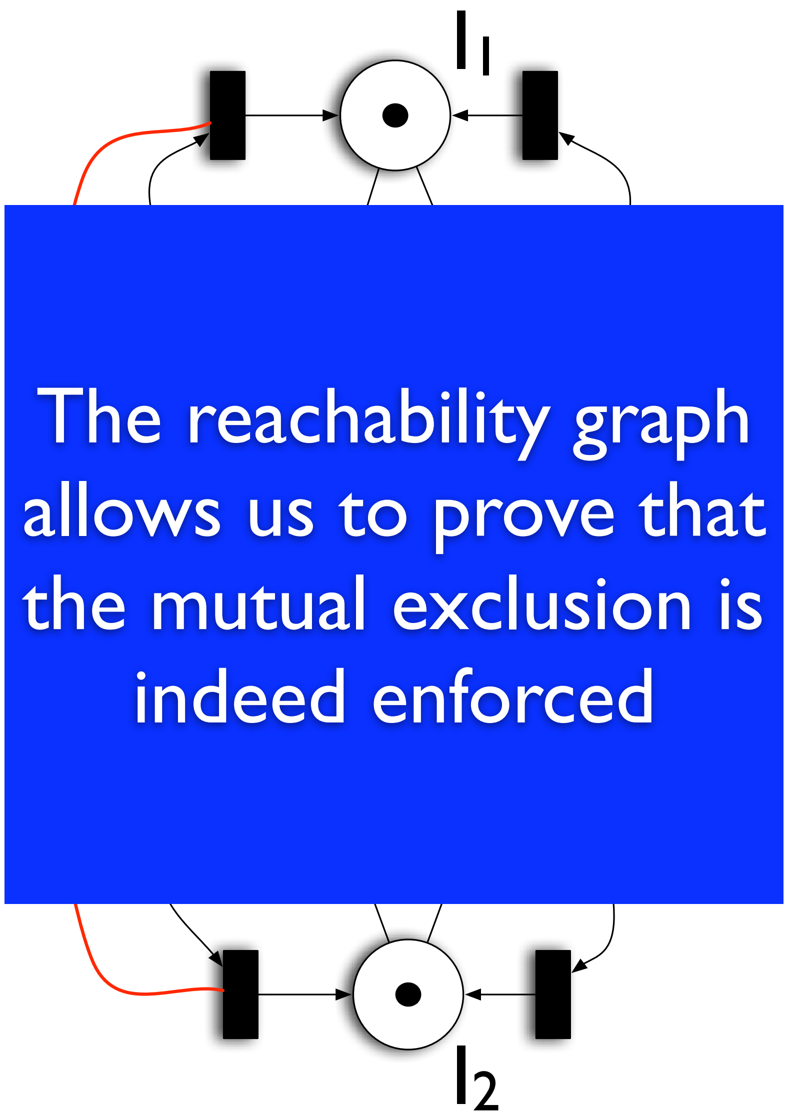



# Reachability graph

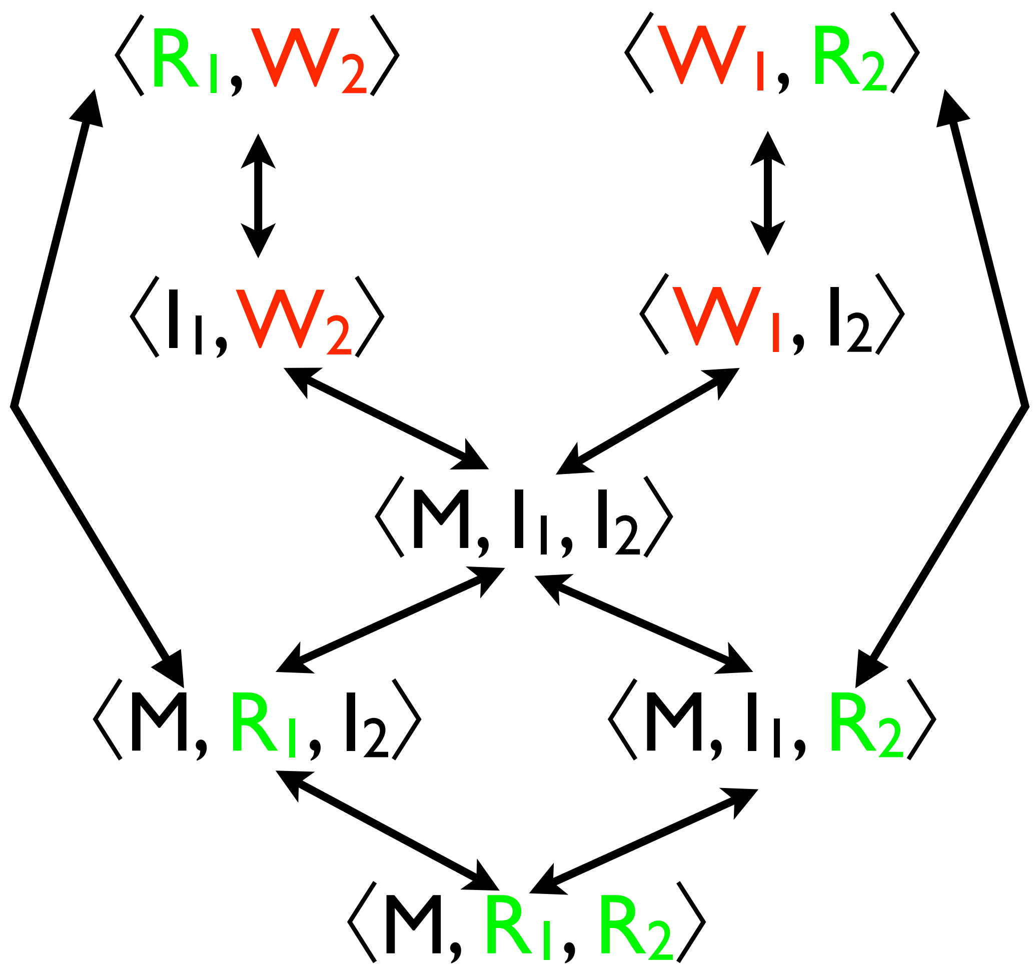


# Reachability graph




The reachability graph allows us to prove that the mutual exclusion is indeed enforced



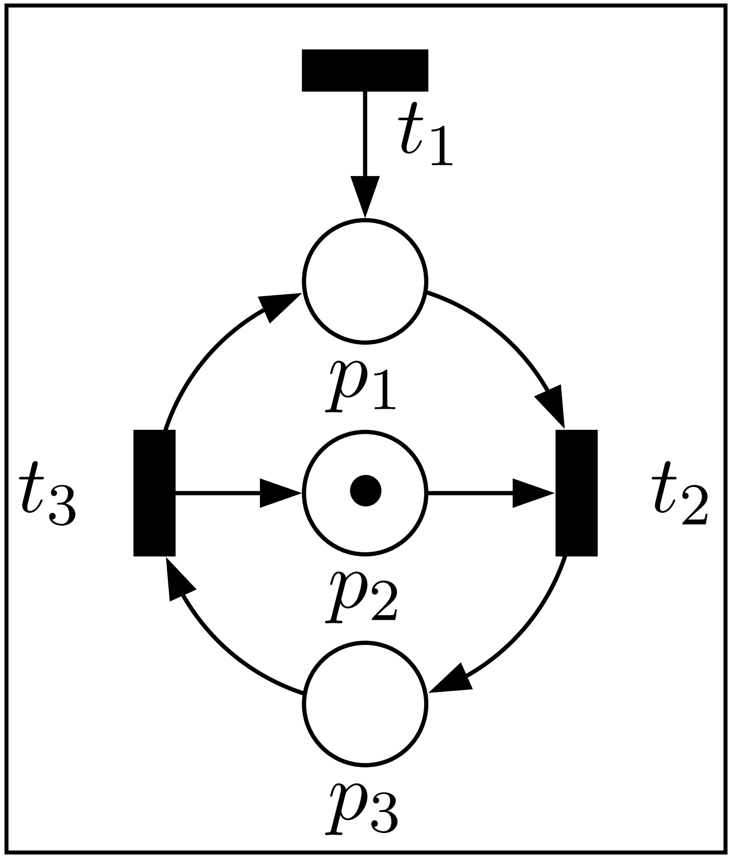
# Reachability graph

- The reachability graph of a PN contains all the necessary information to decide:
  - boundedness
  - place boundedness
  - semi-liveness
  - ...

# Reachability graph

- Unfortunately...

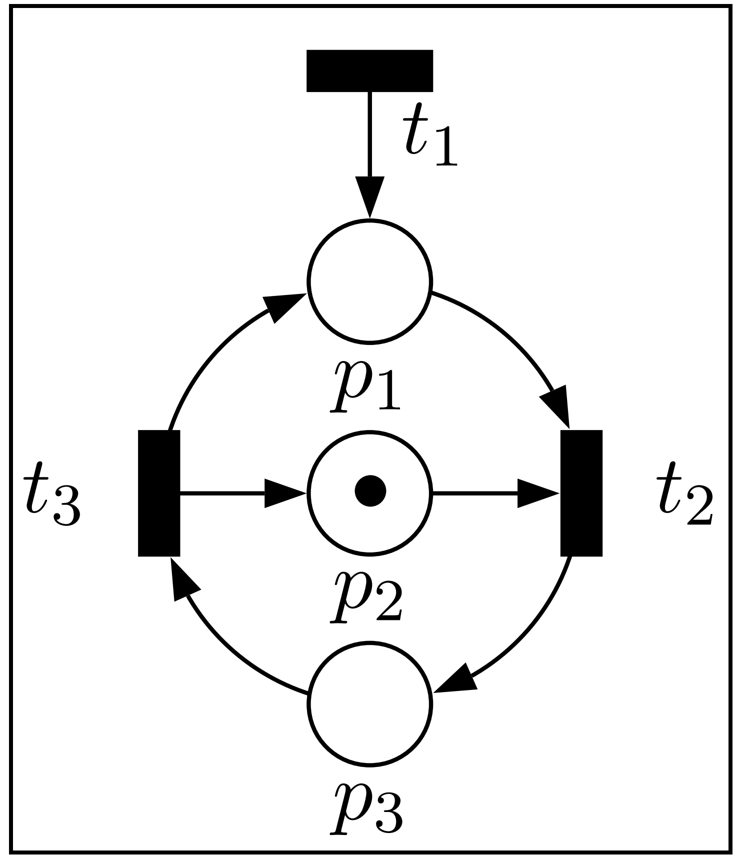
  $\langle p_2 \rangle$



# Reachability graph

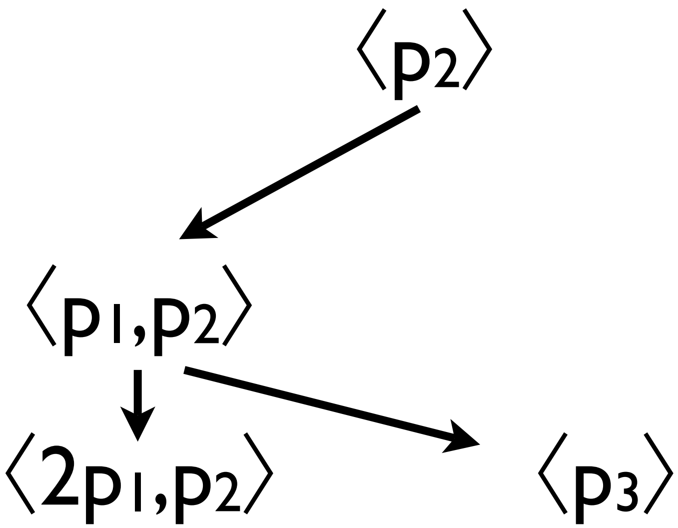
- Unfortunately...

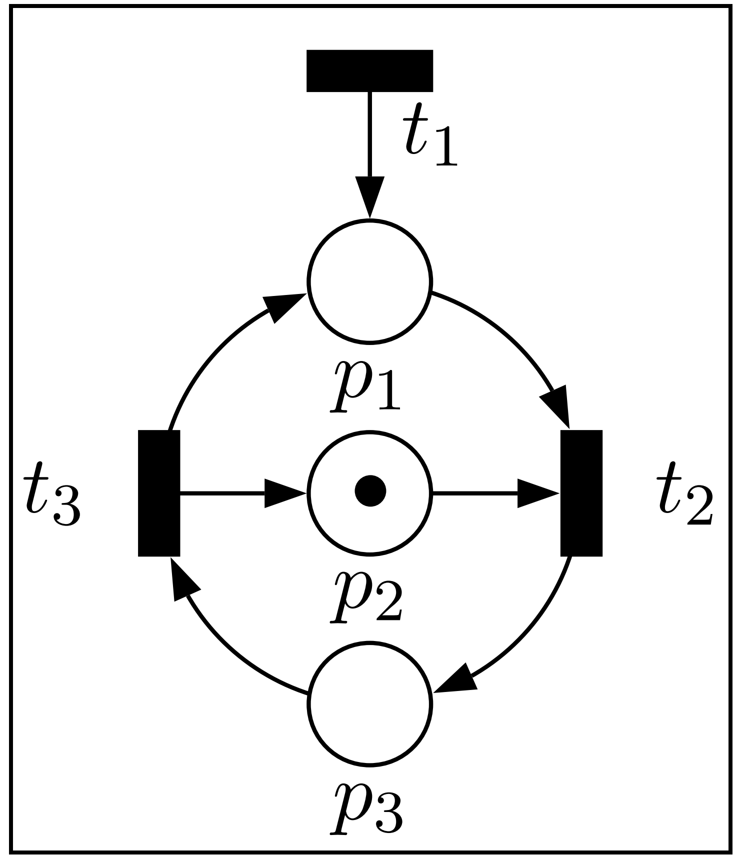
$$\langle p_2 \rangle \longrightarrow \langle p_1, p_2 \rangle$$



# Reachability graph

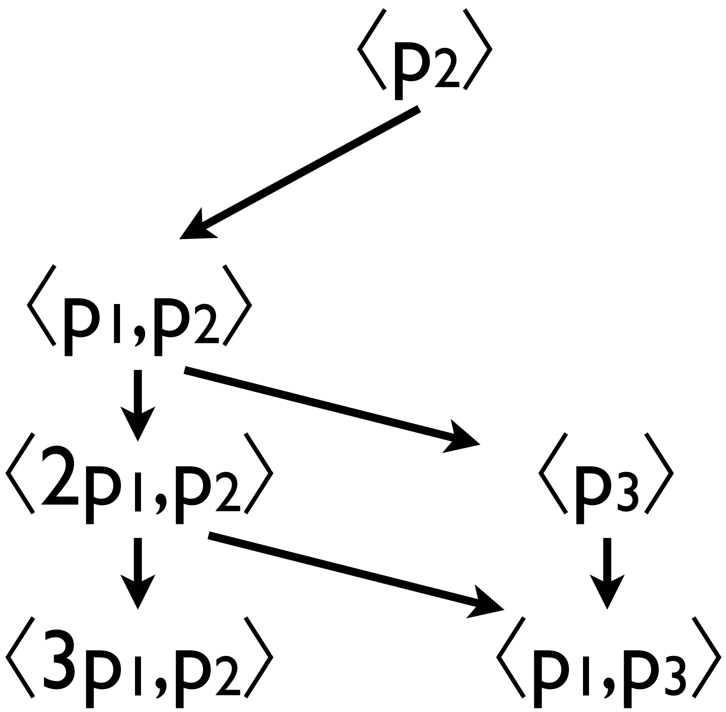
- Unfortunately...

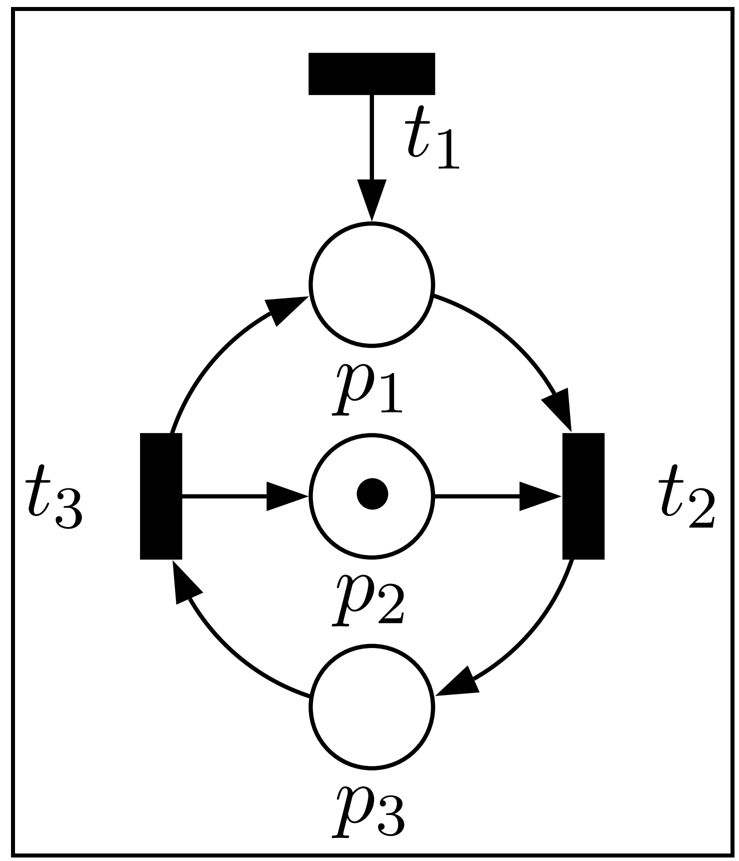


# Reachability graph

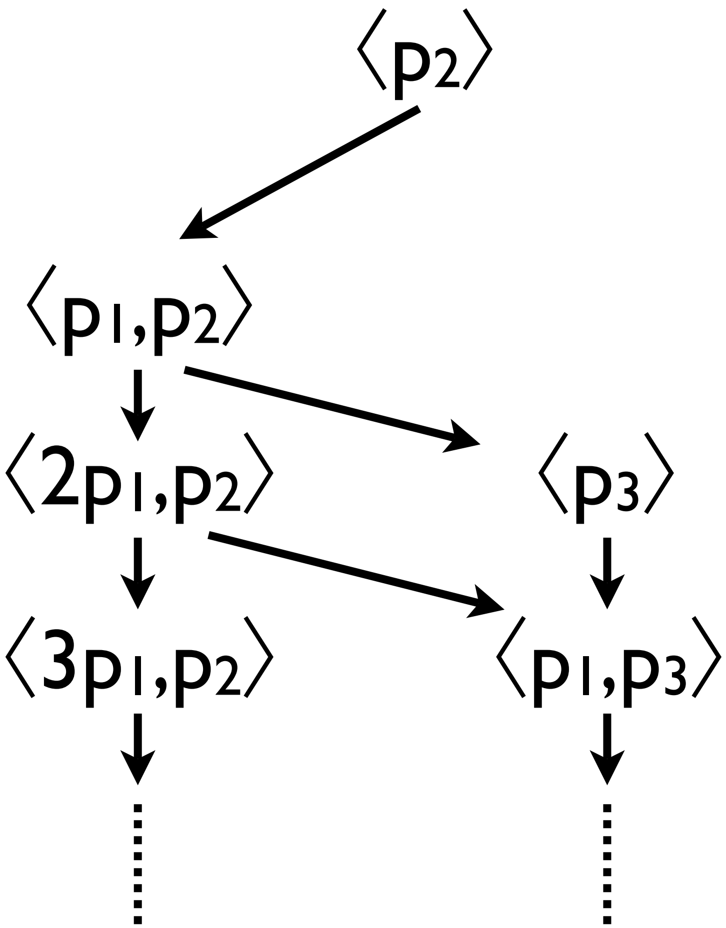
- Unfortunately...

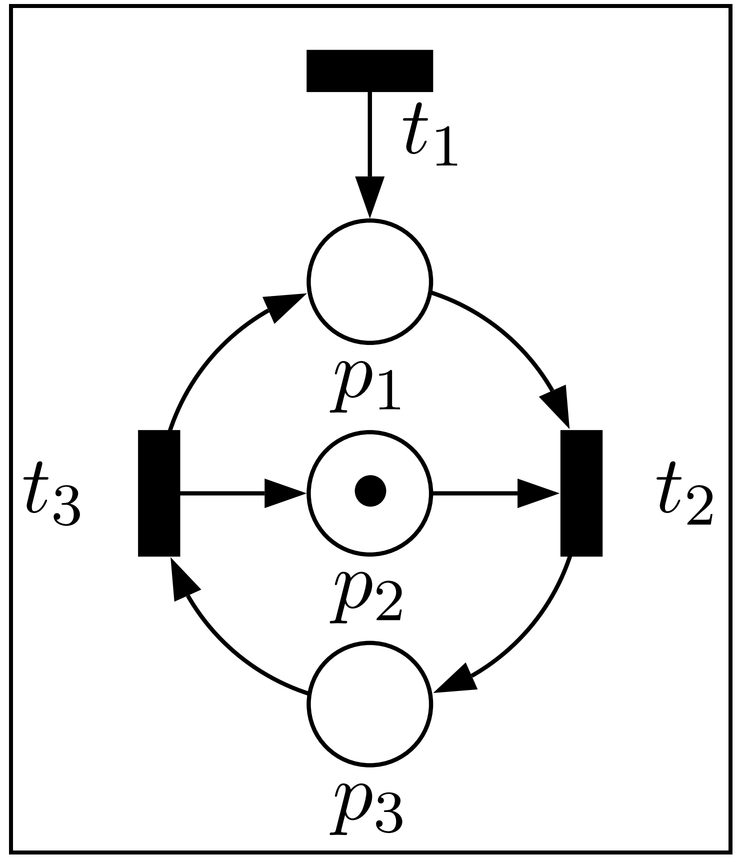


# Reachability graph

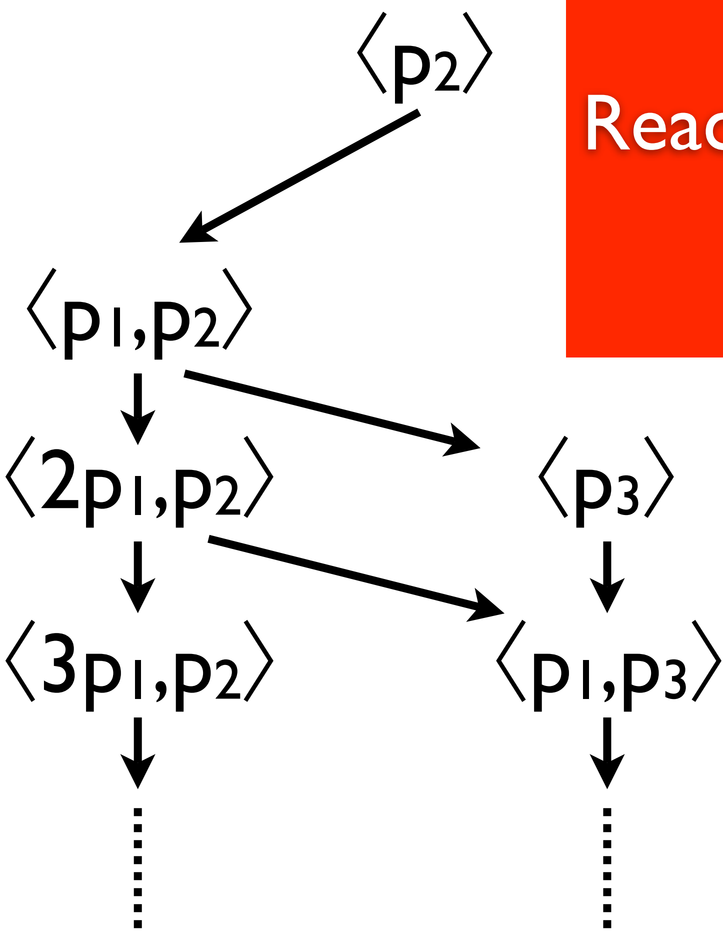
- Unfortunately...



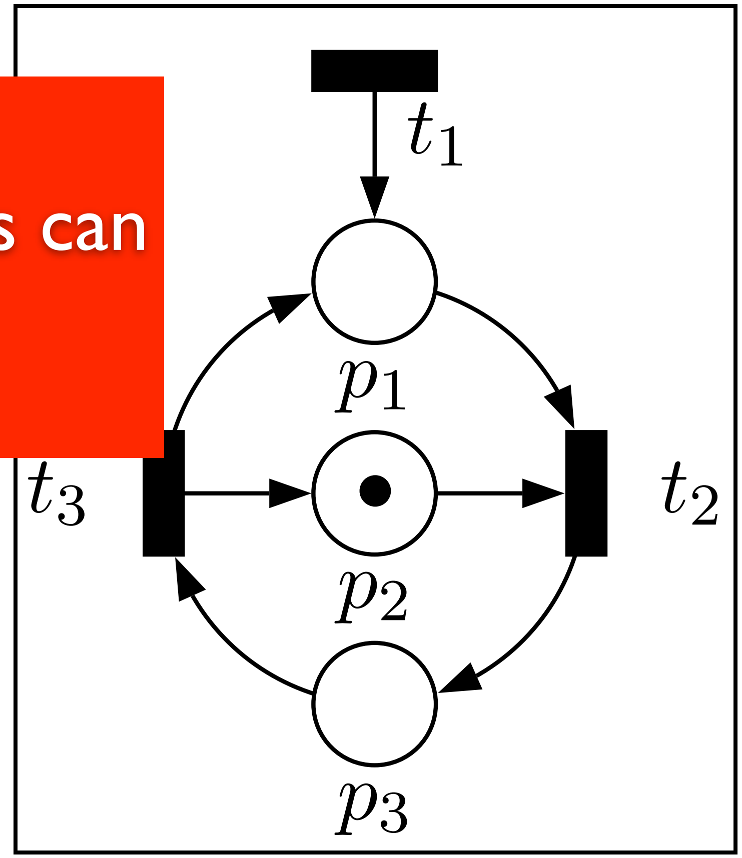



# Reachability graph

- Unfortunately...




Reachability graphs can be infinite



# The hard stuff...

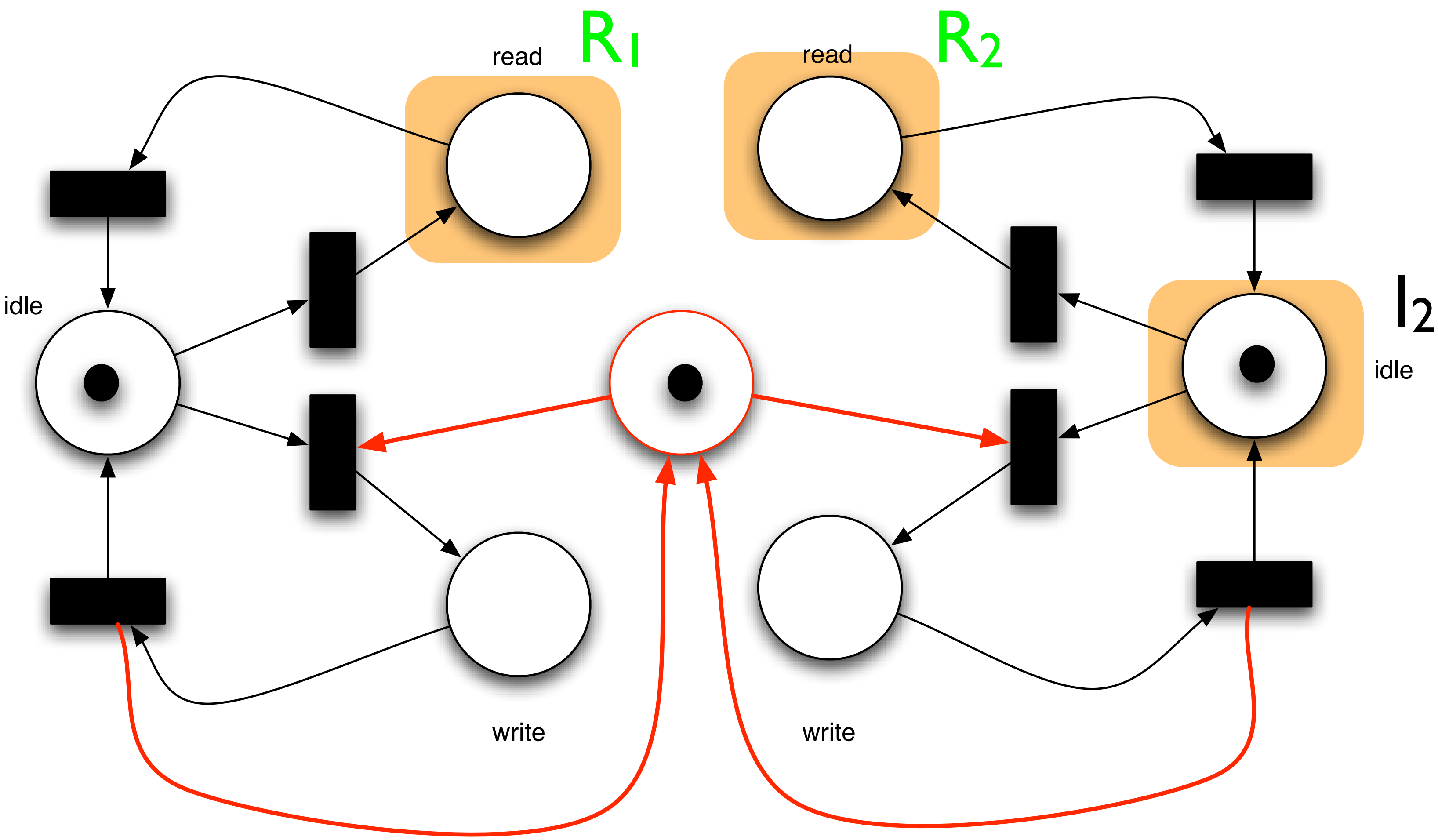
- The main difficulty in analysing Petri nets is due to the possibly infinite number of reachable markings.
  - We have to find techniques to deal with this infinite set.

# The hard stuff...

- Remark: finite doesn't mean easy
  - The set of reachable markings of a bounded net can be huge !
- Efficient techniques to deal with bounded nets have been developped.
  - e.g.: net unfoldings

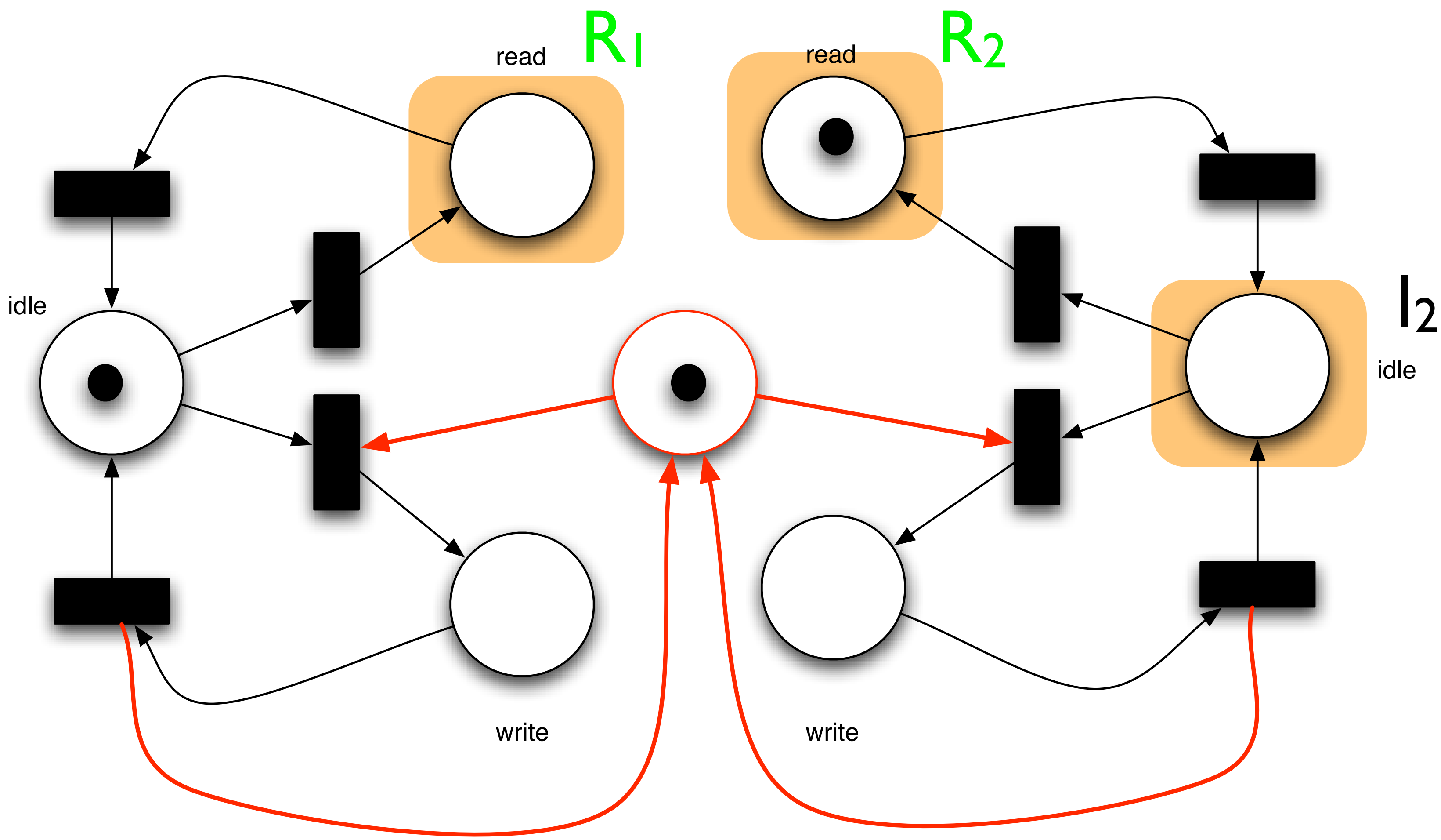
# Place invariants

# Place Invariants



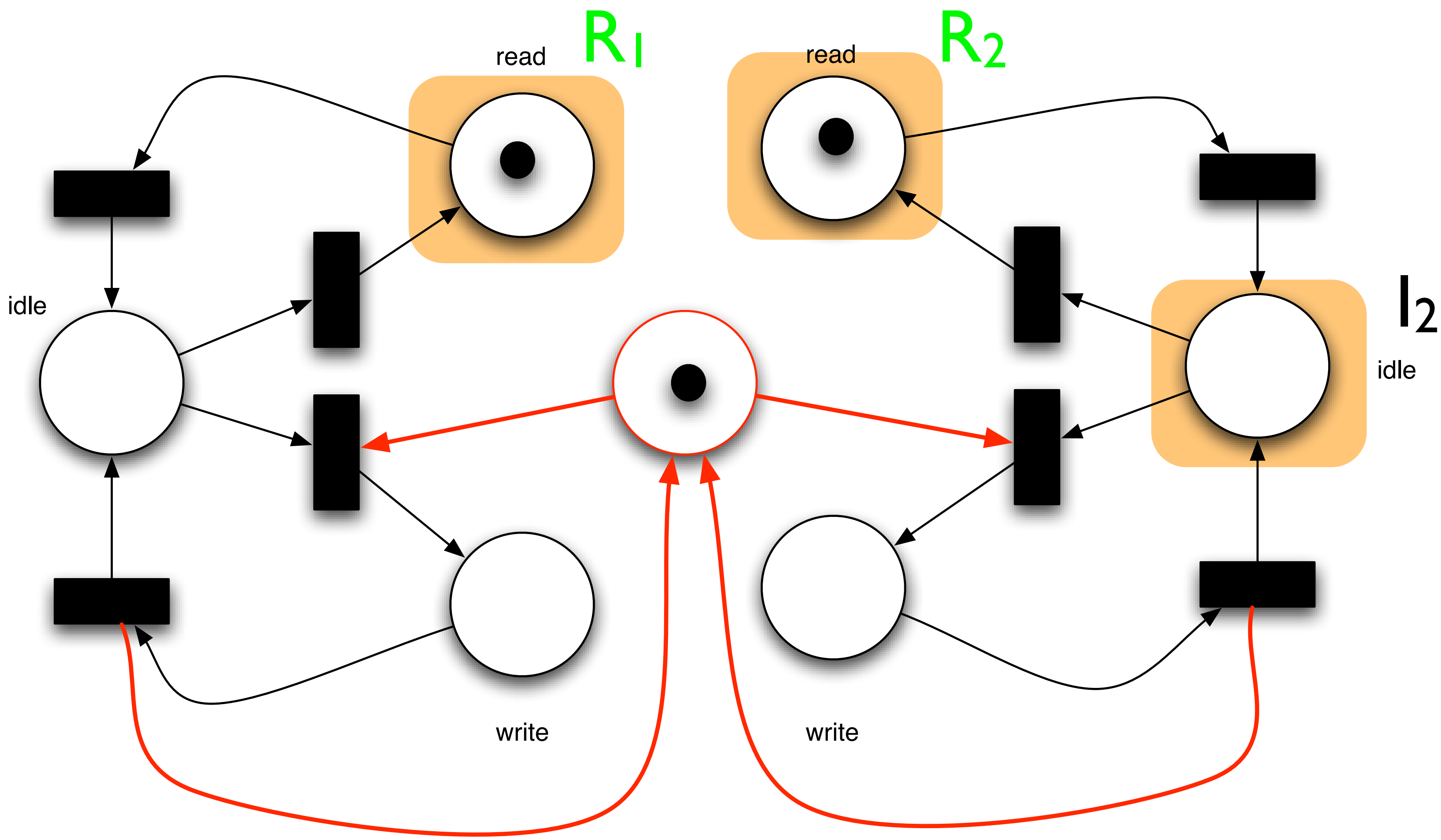

$$m(R_1) + m(R_2) + m(I_2) = 1$$

# Place Invariants



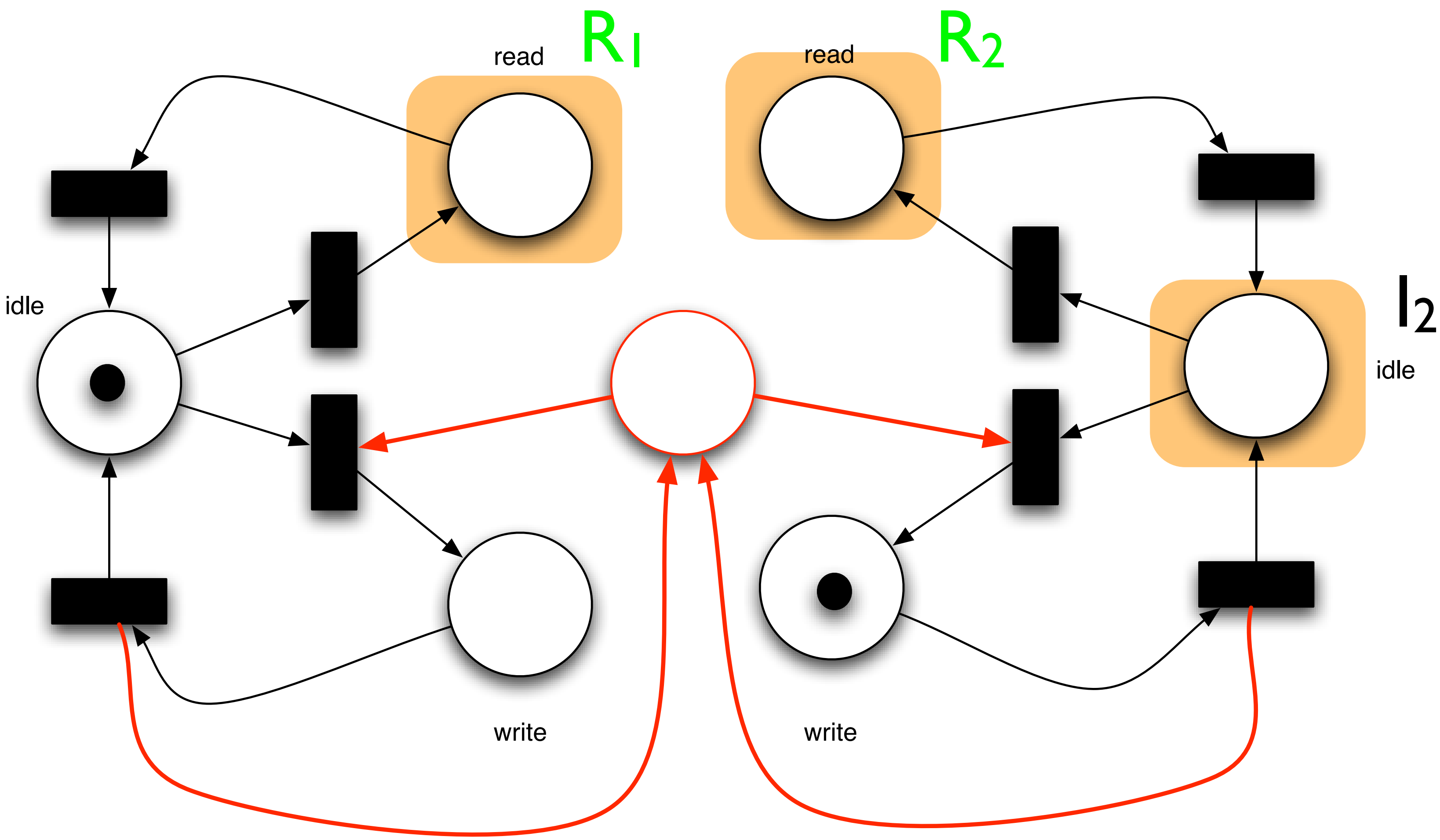

$$m(R_1) + m(R_2) + m(I_2) = 1$$

# Place Invariants



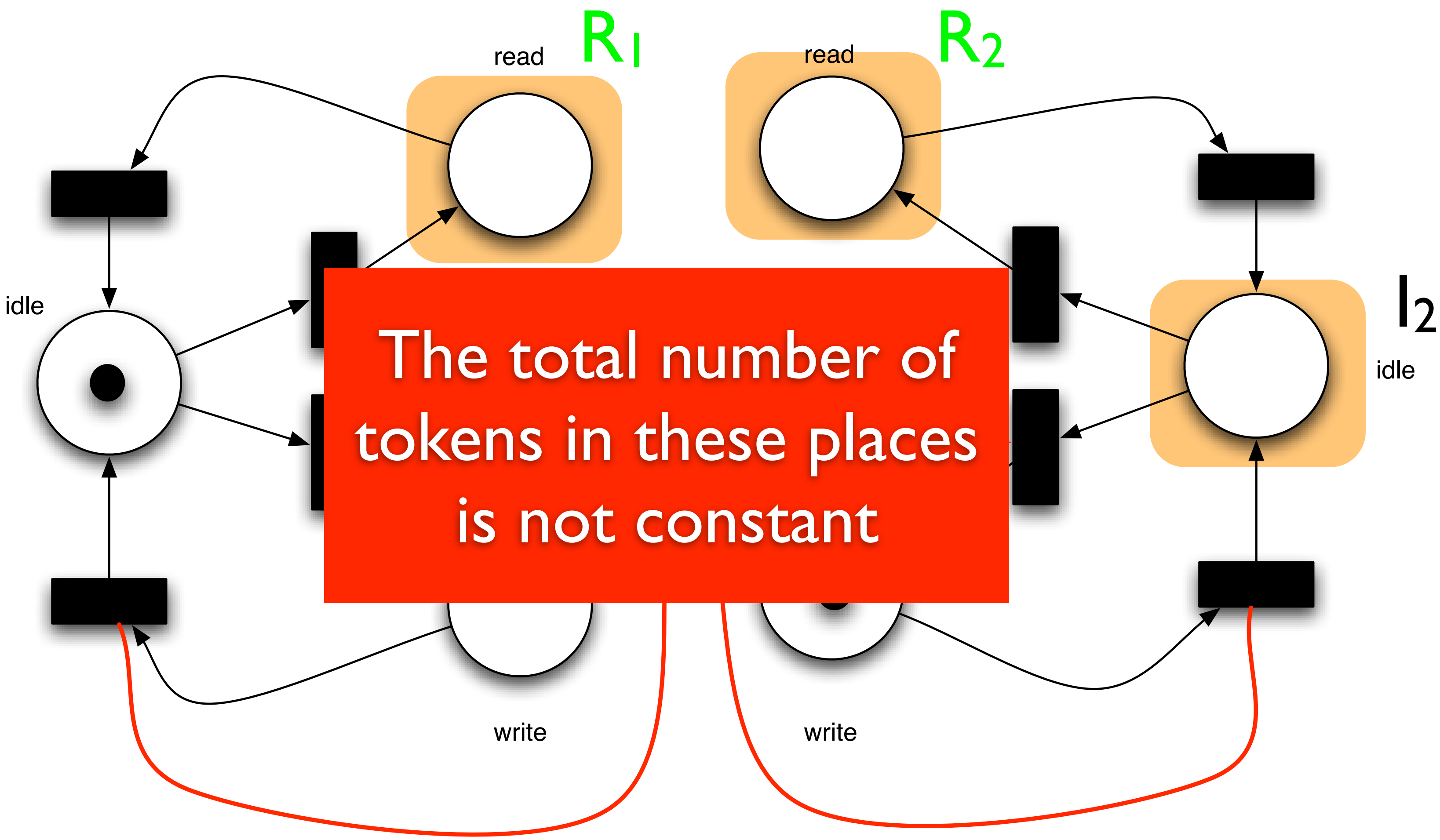

$$m(R_1) + m(R_2) + m(I_2) = 2$$

# Place Invariants



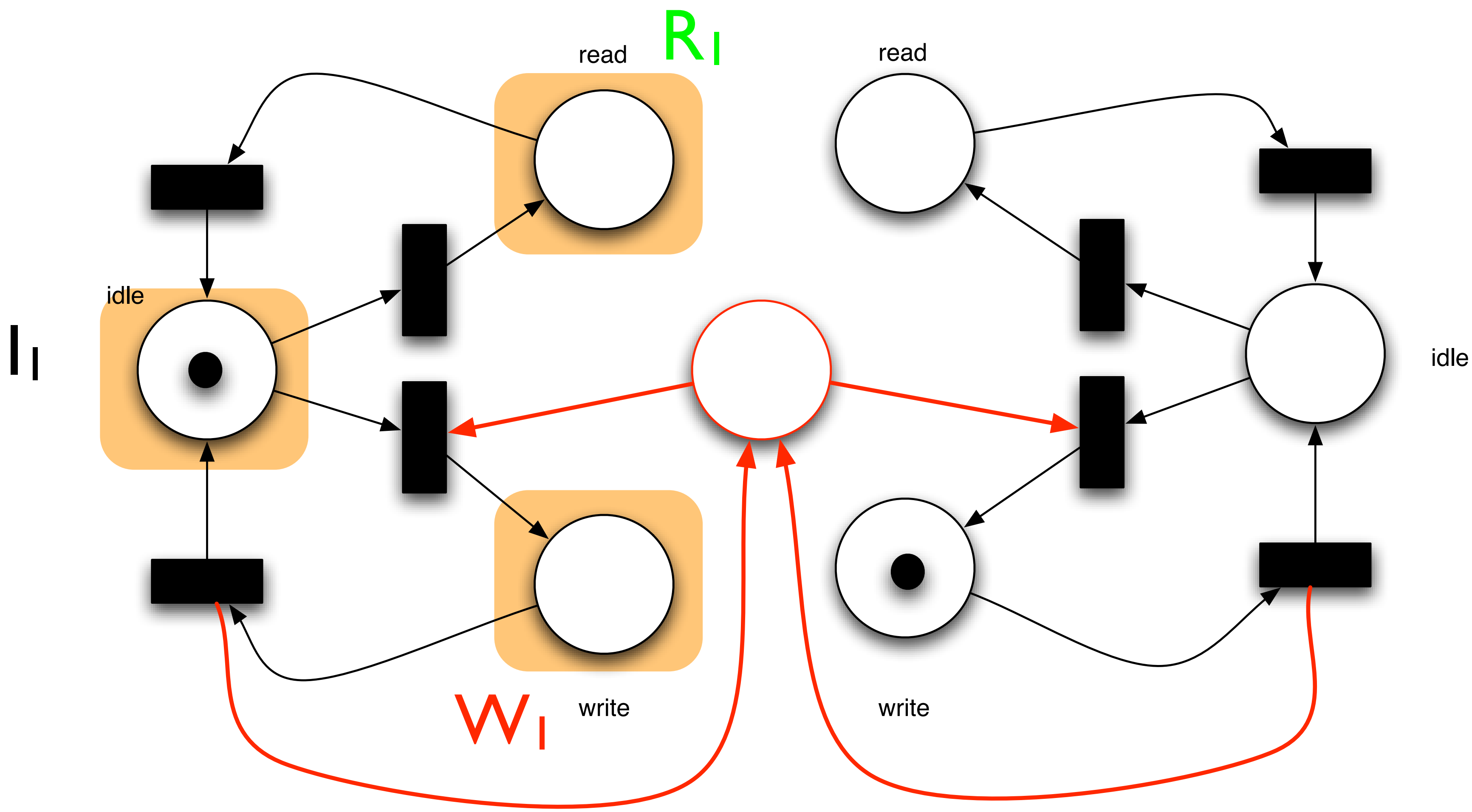

$$m(R_1) + m(R_2) + m(I_2) = 0$$

# Place Invariants



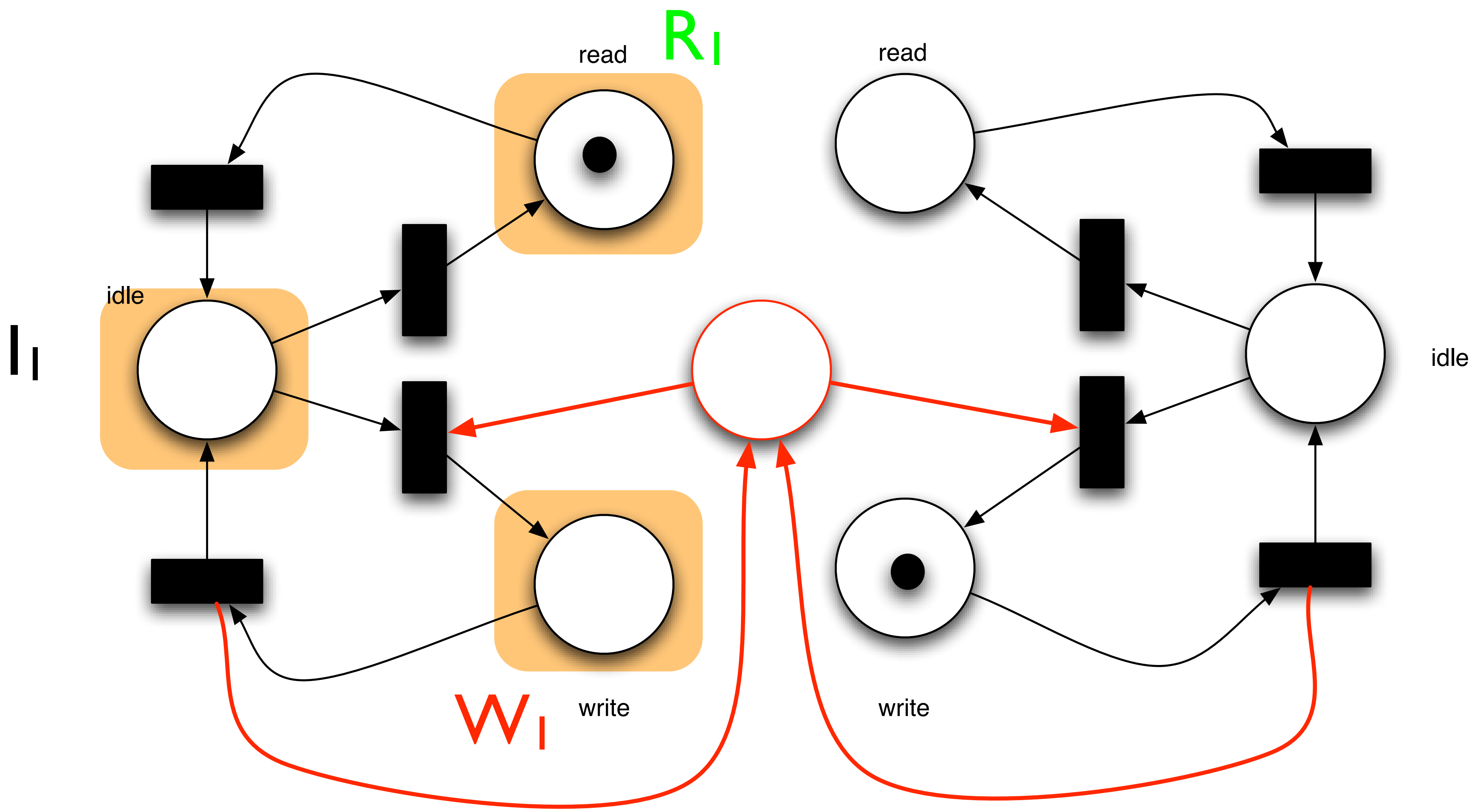

$$m(R_1) + m(R_2) + m(I_2) = 0$$

# Place Invariants



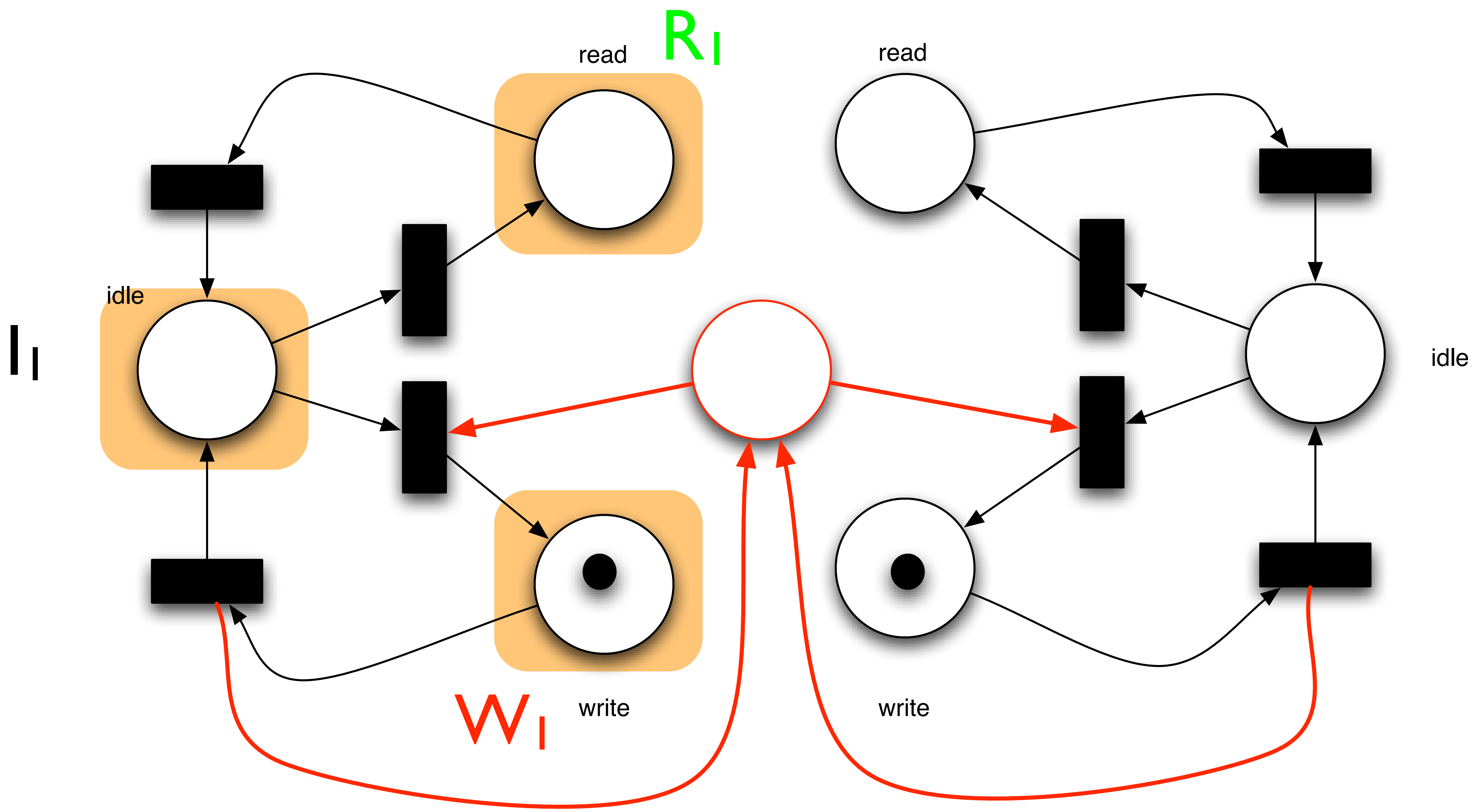

$$m(R_1) + m(W_1) + m(I_1) = 1$$

# Place Invariants



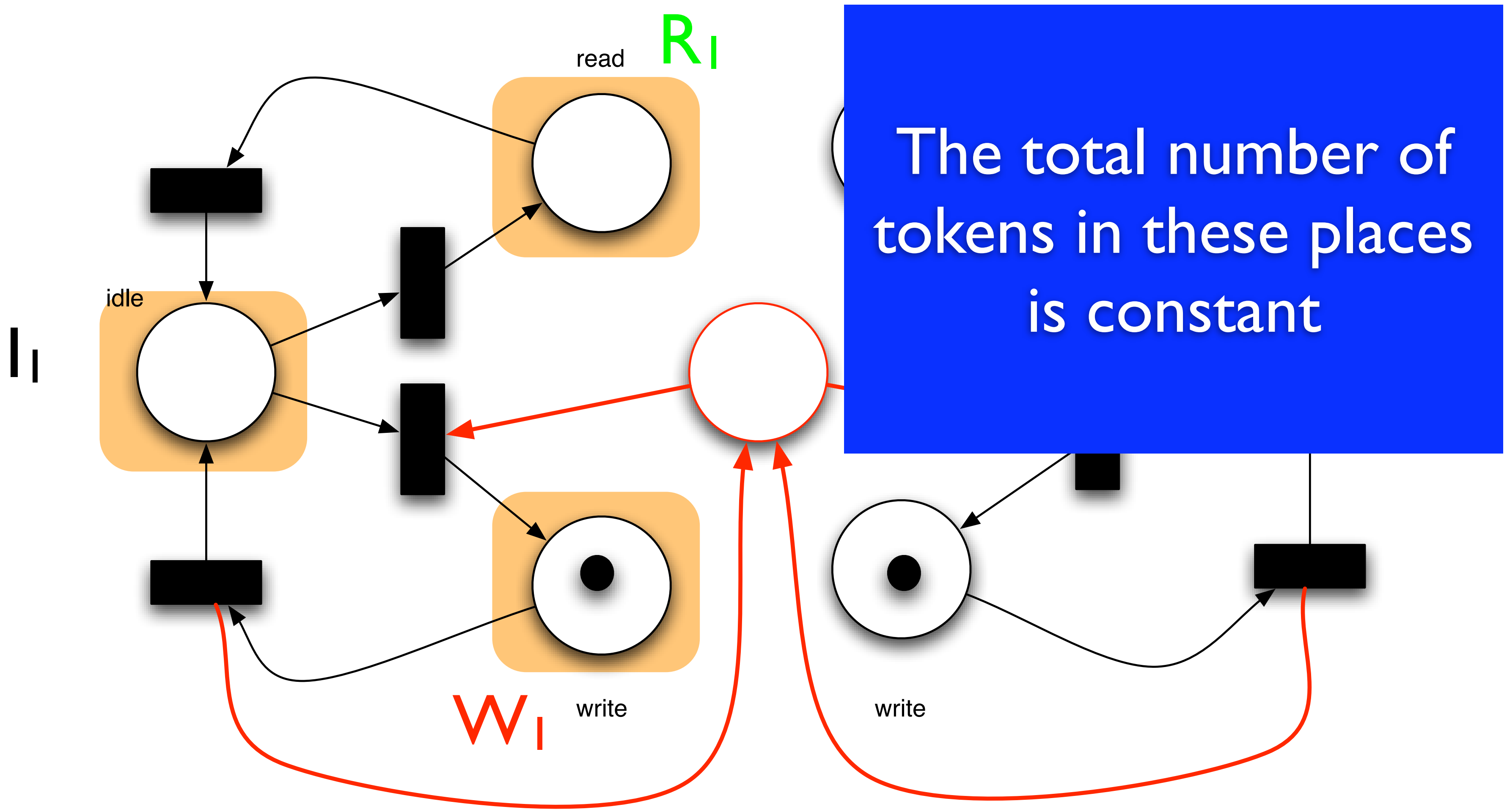

$$m(R_1) + m(W_1) + m(I_1) = 1$$

# Place Invariants



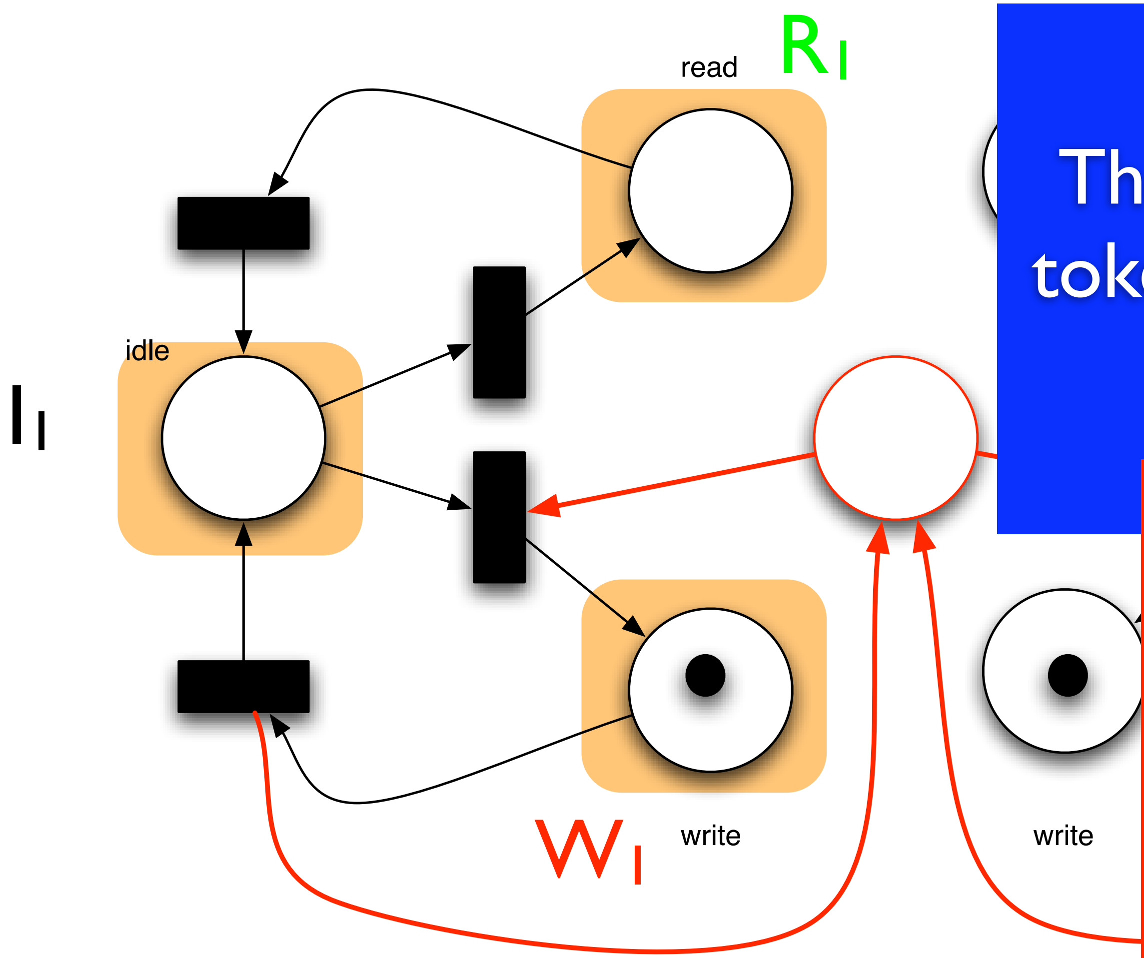

$$m(R_1) + m(W_1) + m(I_1) = 1$$

# Place Invariants




$$m(R_1) + m(W_1) + m(I_1) = 1$$

# Place Invariants

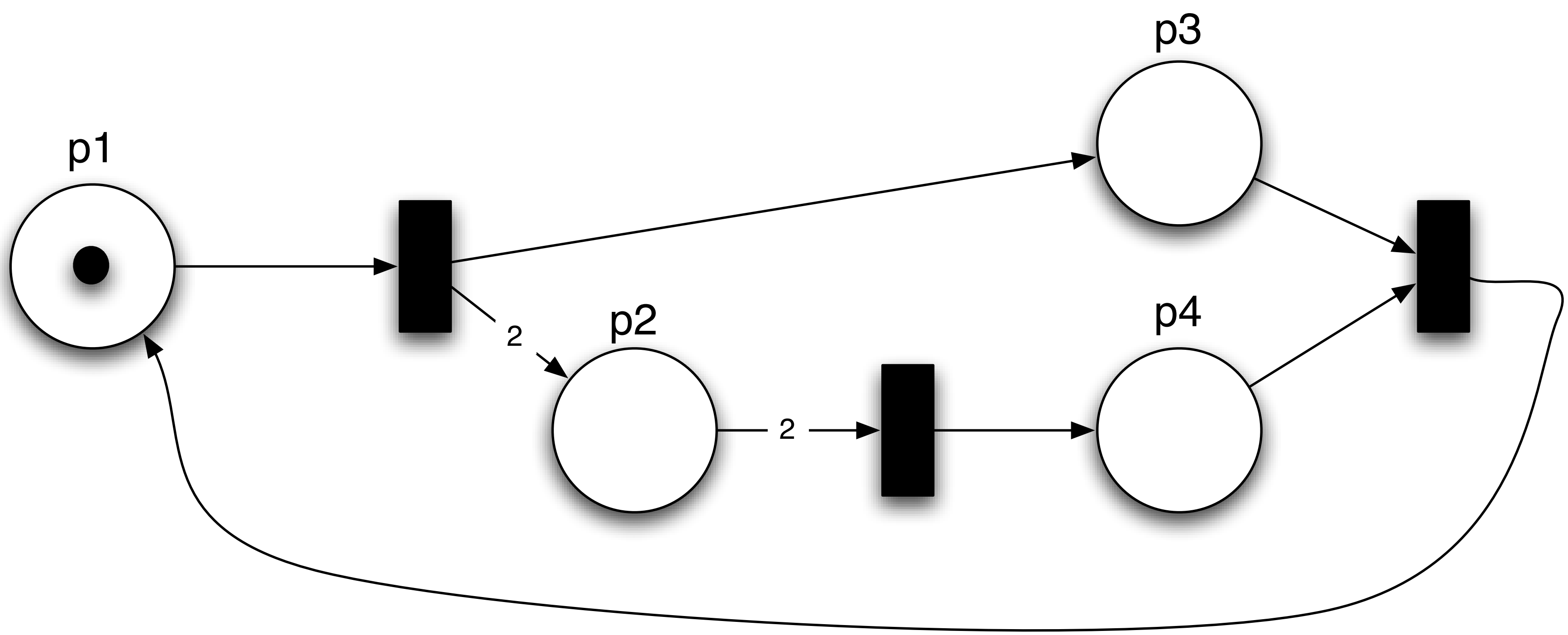



The total number of tokens in these places is constant

This provides meaningful information about the system: a process is either idle, or reading or writing

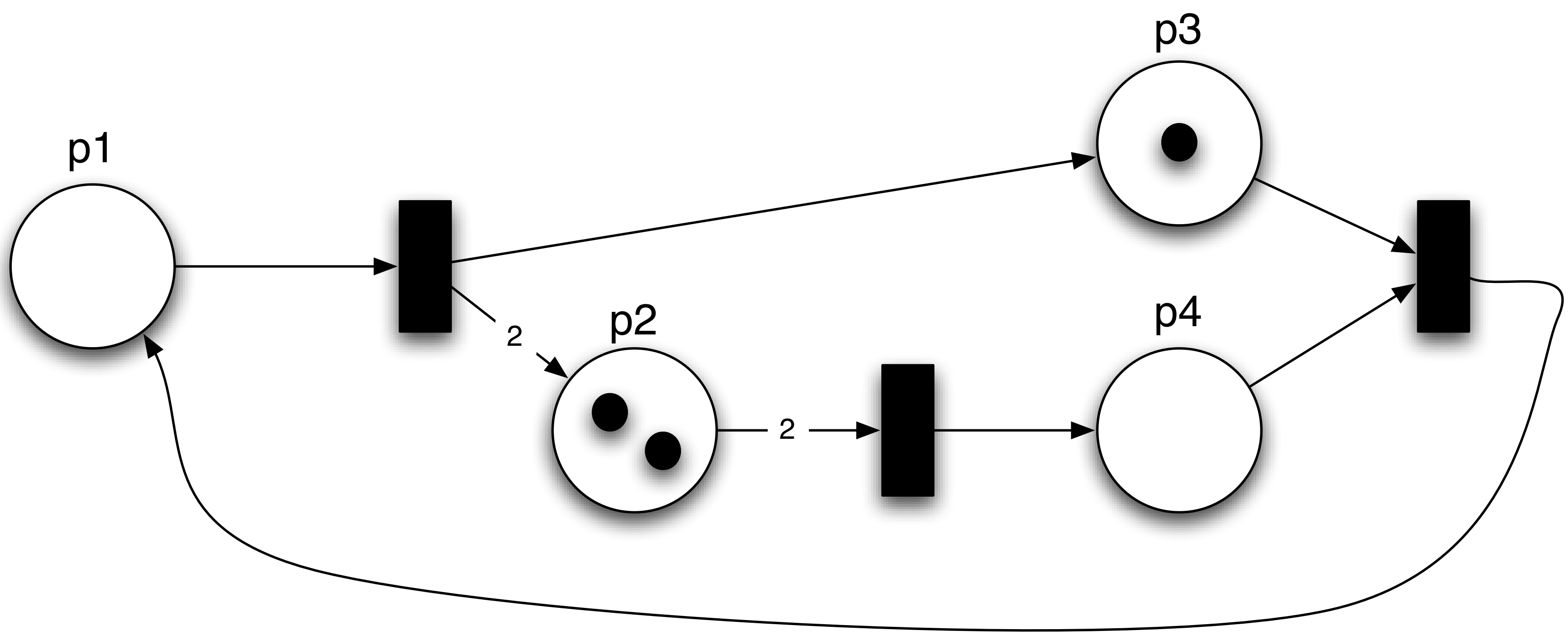
$$m(R_1) + m(W_1) + m(I_1) = 1$$

# Place Invariants



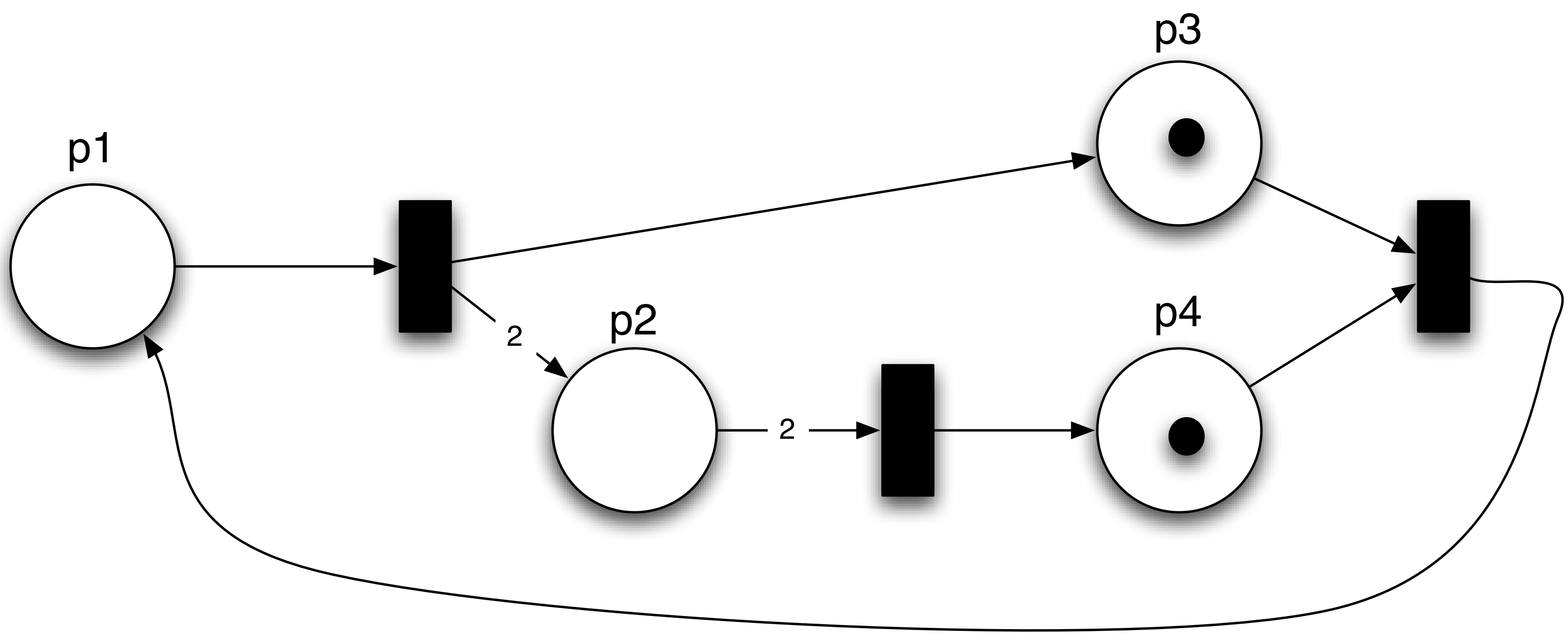

$$m(p_1) + m(p_2) + m(p_3) + m(p_4) = 1$$

# Place Invariants



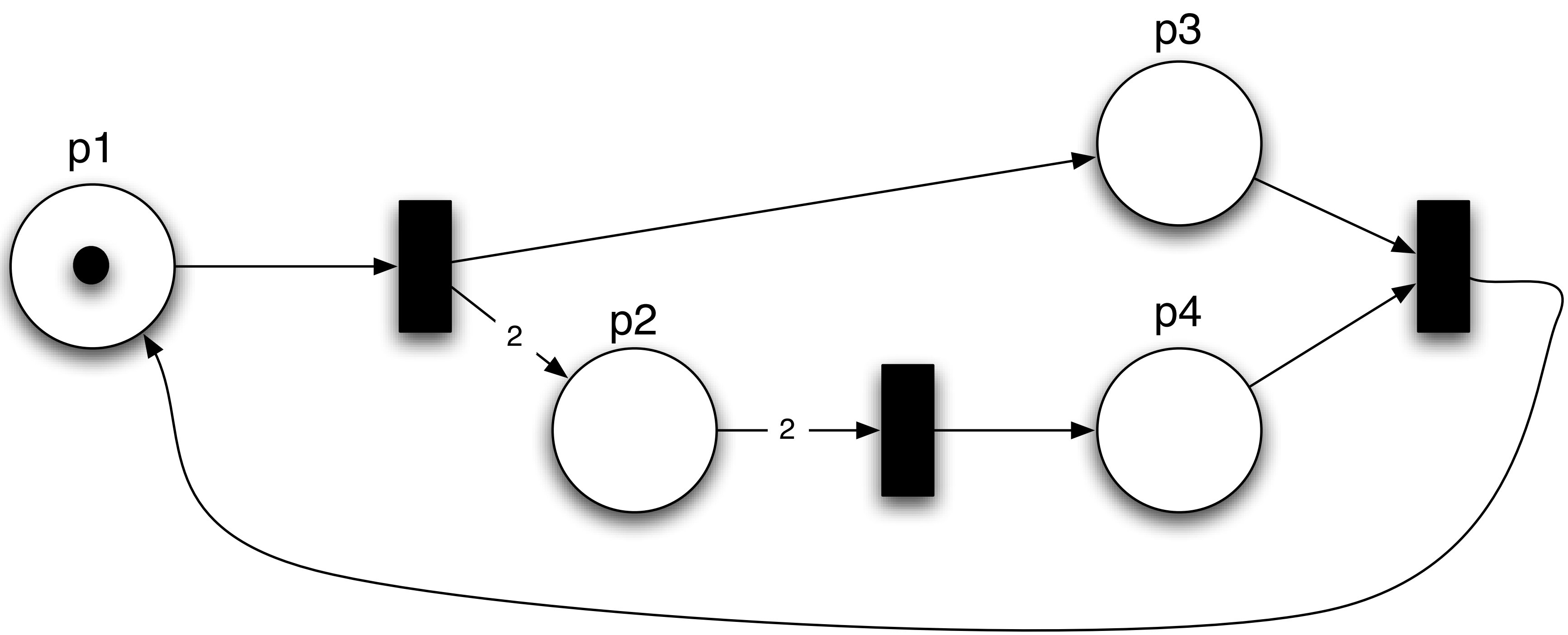

$$m(p_1) + m(p_2) + m(p_3) + m(p_4) = 3$$

# Place Invariants



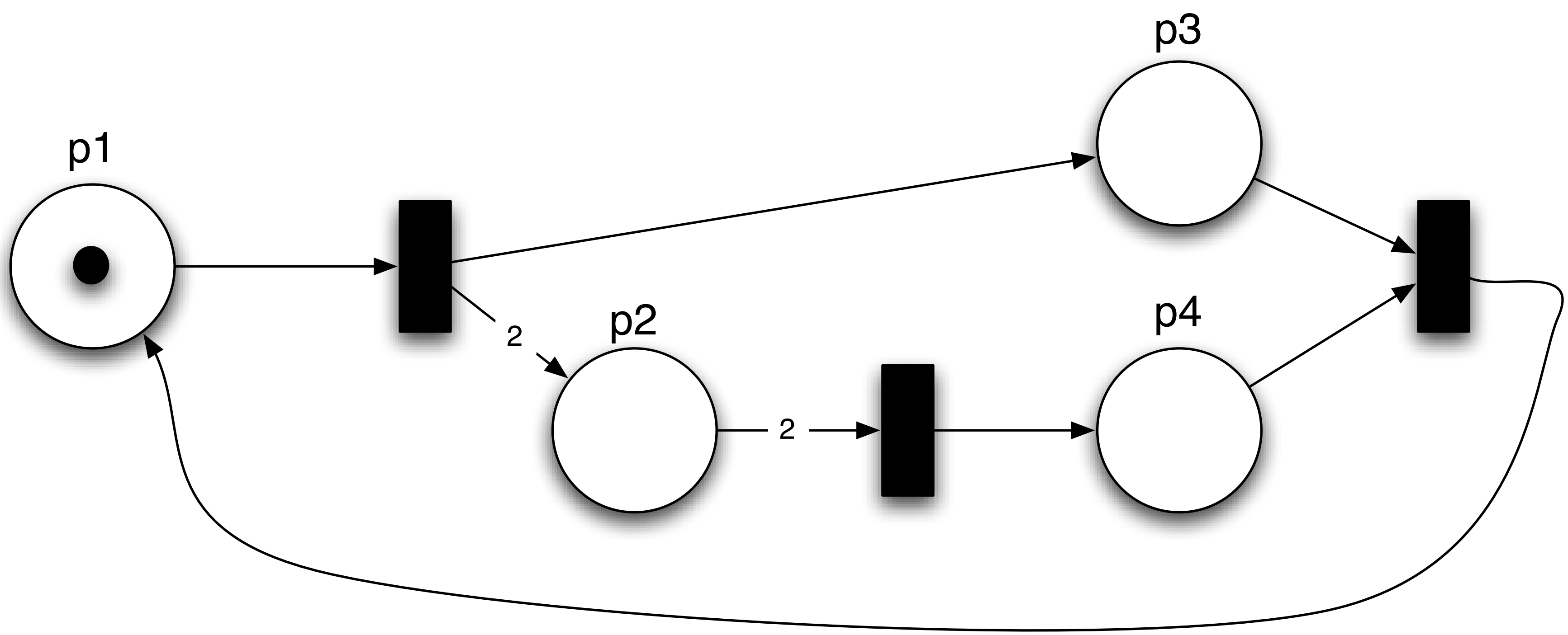

$$m(p_1) + m(p_2) + m(p_3) + m(p_4) = 2$$

# Place Invariants




$$m(p_1) + m(p_2) + m(p_3) + m(p_4) = 1$$

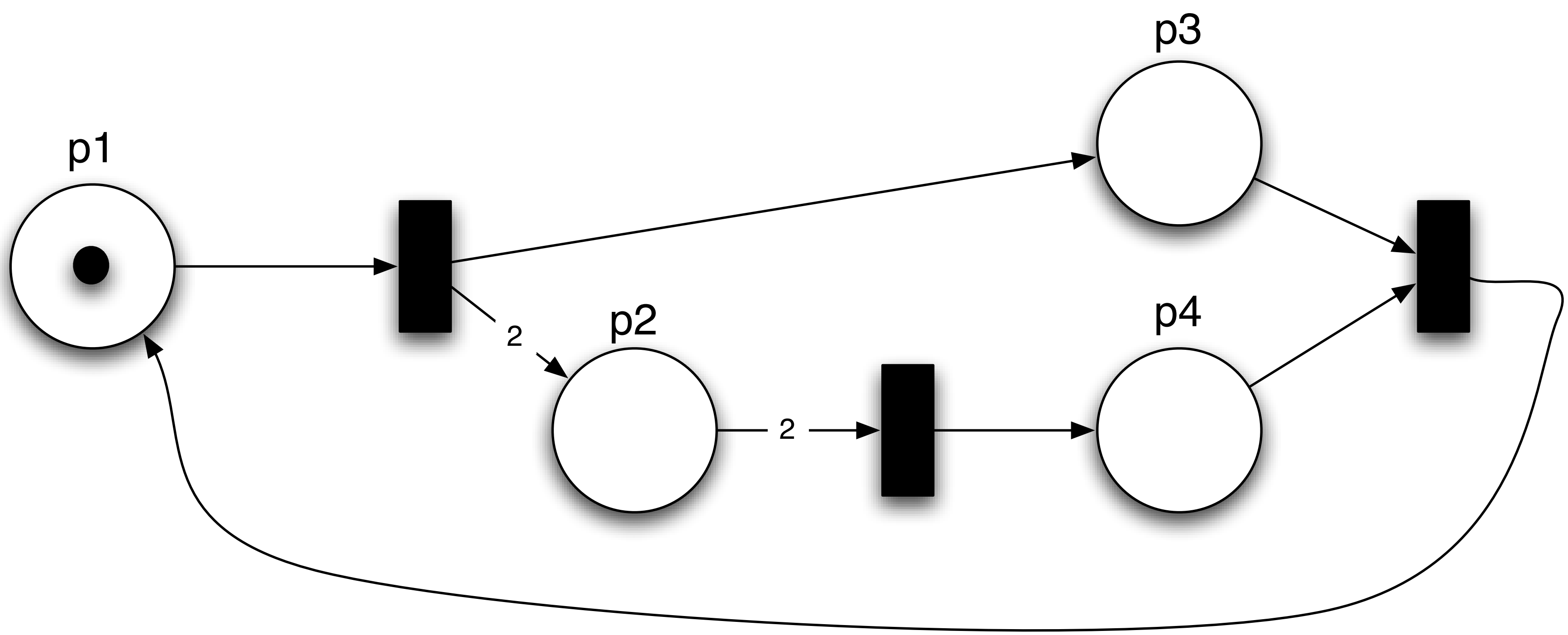
# Place Invariants




The total number of tokens in these places is not constant

$+ m(p_3) + m(p_4) = 1$

# Place Invariants




The total number of tokens in these places is not constant

+ m(p

In some sense, tokens in $p_1$ are heavier than those in $p_2$

# Place Invariants

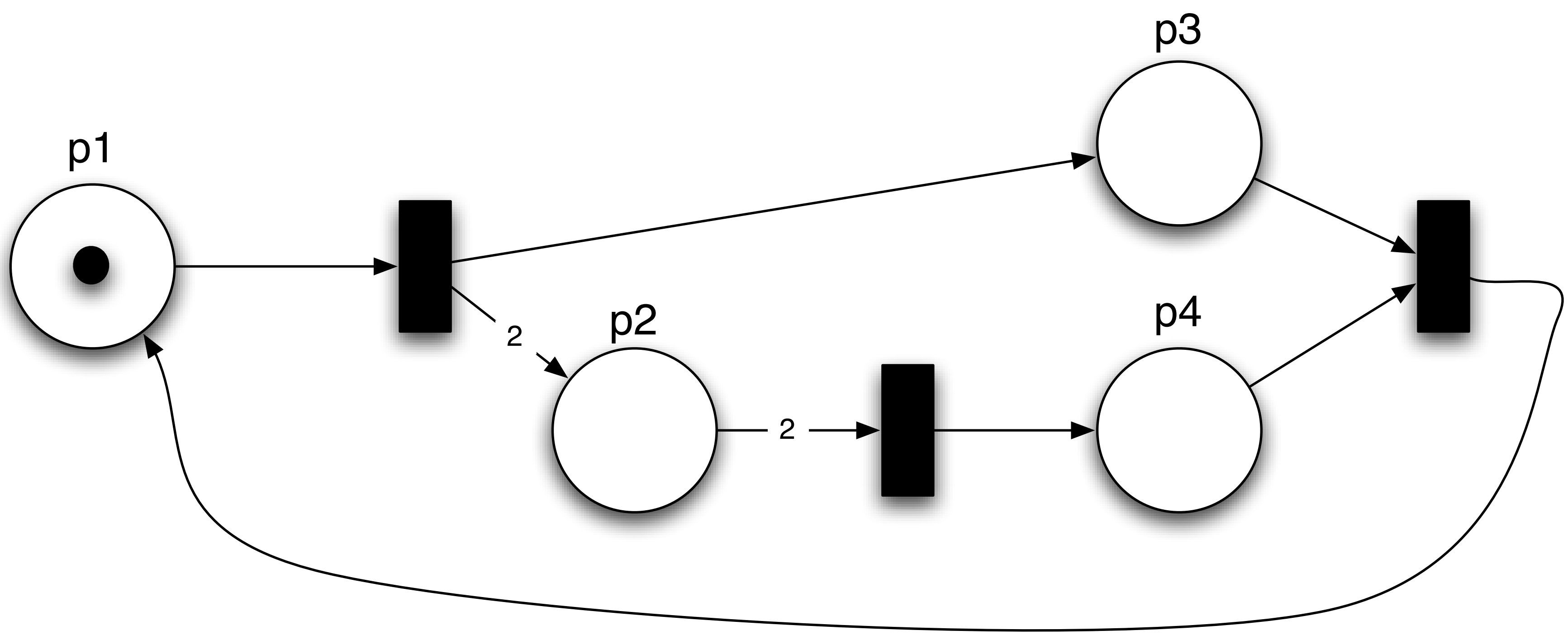

Let's add weights to the places !

The total number of tokens in these places is not constant

+ m(p

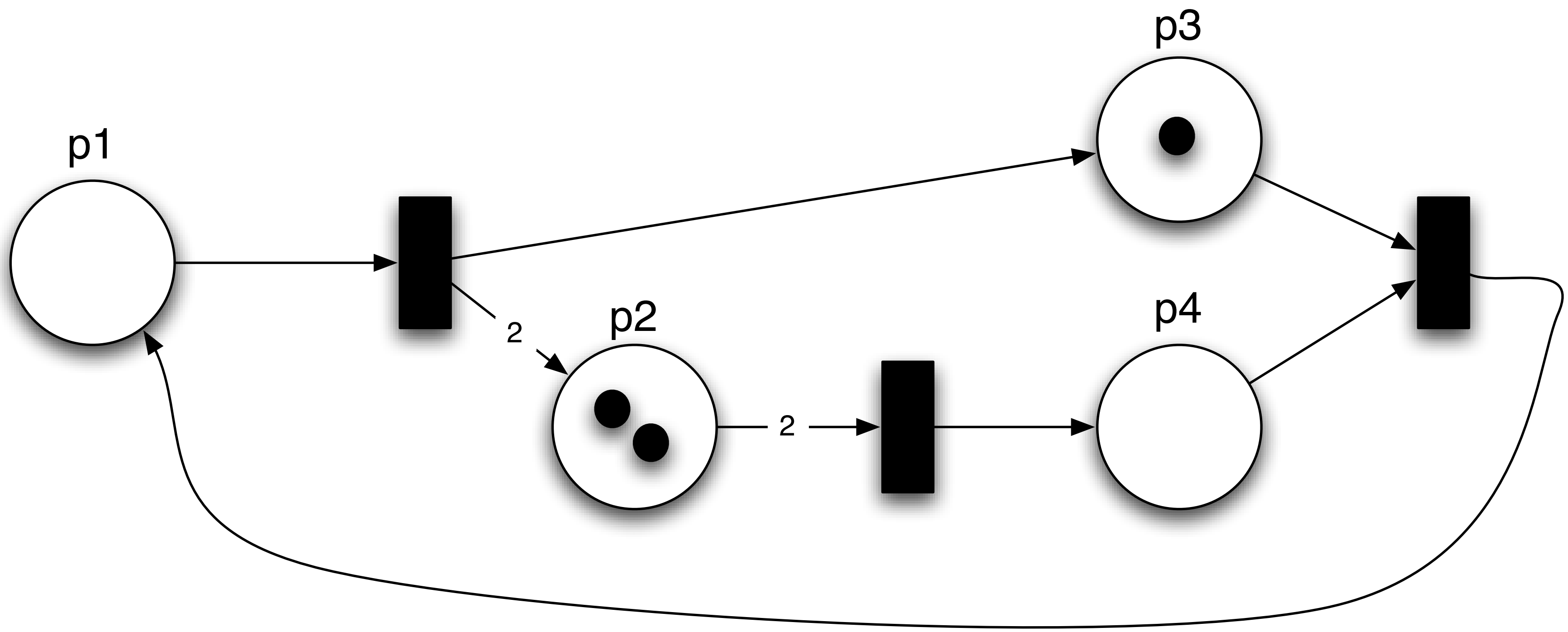
In some sense, tokens in $p_1$ are heavier than those in $p_2$

# Place Invariants



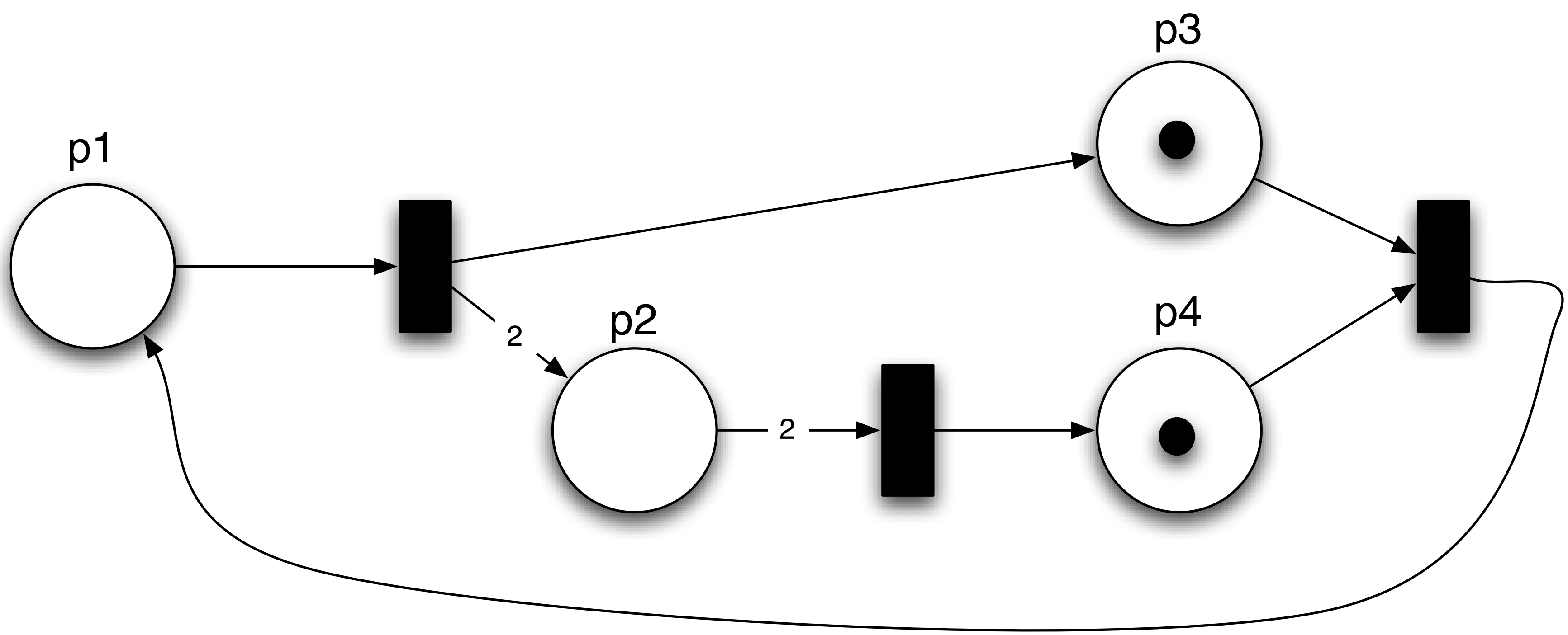

$$3\ m(p_1) + m(p_2) + m(p_3) + 2\ m(p_4) = 3$$

# Place Invariants




$$3\ m(p_1) + m(p_2) + m(p_3) + 2\ m(p_4) = 3$$

# Place Invariants




$$3\ m(p_1) + m(p_2) + m(p_3) + 2\ m(p_4) = 3$$

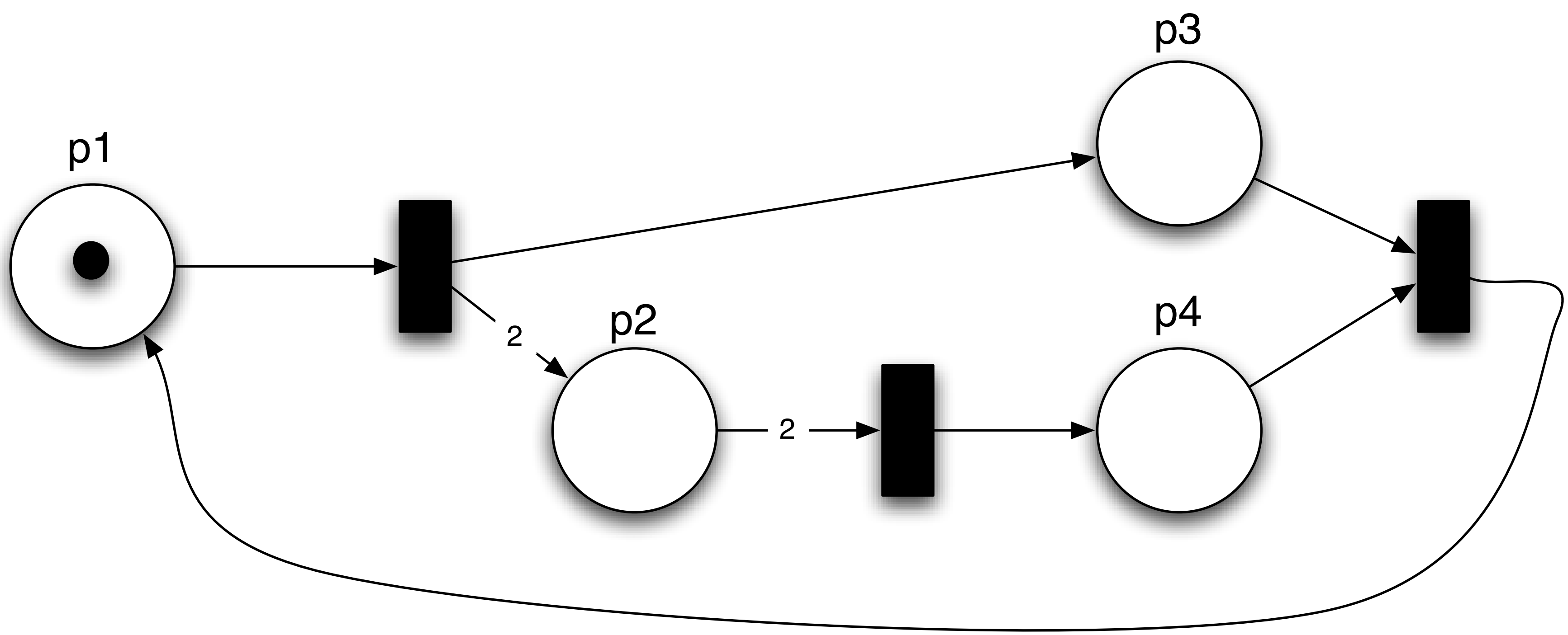
# Place invariant: Definition

- Definition: a place-invariant (or p-semiflow) is a vector i of natural numbers s.t. for any reachable marking m:

$$\sum_{p \in P} i(p) \times m(p) = \sum_{p \in P} i(p) \times m_0(p)$$

remark: there exists a trivial invariant i = $\langle 0, 0, .., 0 \rangle$

# Example: other invariants




$$m(p_1) + m(p_3) = 1$$

$$2\, m(p_1) + m(p_2) + 2\, m(p_4) = 2$$

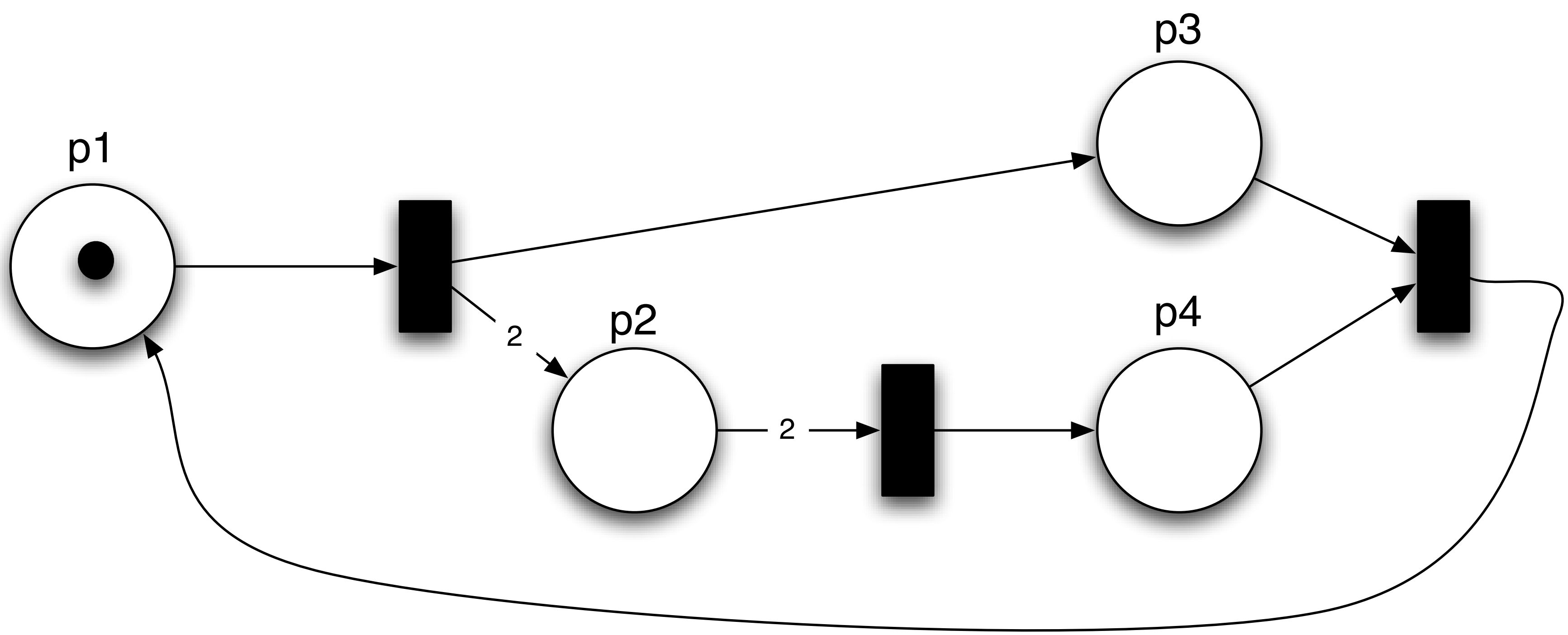
# Invariants as over-approximations

- A place-invariant expresses a constraint on the reachable markings.
  - If m is reachable and i is an invariant, then:

$$\sum_{p \in P} i(p) \times m(p) = \sum_{p \in P} i(p) \times m_0(p)$$

- The reverse is not true !

# Example




$$m(p_1) + m(p_3) = 1$$

is an invariant

but $\langle 1, 25, 0, 234 \rangle$ is not reachable

# Invariants as over-approximations

- Theorem: For any Petri net N:

$$\mathrm{Reach}(N)$$

$$\subseteq$$

$$\{m \mid m \text{ respects some invariant of } N\}$$

# Invariants as over-approximations

- Theorem: For any Petri net N:

$$\mathrm{Reach}(N) \subseteq \{m \mid m \text{ respects some invariant of } N\}$$

This set overapproximates the reachable markings

# Invariants as over-approximations

- Theorem: For any Petri net N:

Reach(N)

$\subseteq$

{m | m respects some invariant of N}

This set overapproximates the reachable markings

Place invariants are thus useful to finitely approximate the set of reachable markings

# Place invariant and boundedness

- Theorem: If there exists a place invariant i and a place p s.t. i(p)>0 then p is bounded.
- Remark: the reverse is not true.
  - One can find a bounded net that doesn't have a place invariant i with i(p)>0 for each place.

# Place invariant

- Question: how do we compute them ?

# Matrix characterisation

- The negative effect (consumption) of all the transitions on all the places can be summarised in one matrix:

$$W^- = \left(\begin{array}{cccc} I_1(p_1) & I_2(p_1) & \cdots & I_k(p_1) \\ I_1(p_2) & I_2(p_2) & \cdots & I_k(p_2) \\ \vdots & \vdots & \ddots & \vdots \\ I_1(p_n) & I_2(p_n) & \cdots & I_k(p_n) \end{array}\right) \begin{array}{l} \text{neg. eff. on } p_1 \\ \text{neg. eff. on } p_2 \\ \\ \\ \end{array}$$

where, for any i: $t_i = \langle I_i, O_i \rangle$

# Matrix characterisation

- The same can be done with the positive effects:

$$W^+ = \left(\begin{array}{cccc} O_1(p_1) & O_2(p_1) & \cdots & O_k(p_1) \\ O_1(p_2) & O_2(p_2) & \cdots & O_k(p_2) \\ \vdots & \vdots & \ddots & \vdots \\ O_1(p_n) & O_2(p_n) & \cdots & O_k(p_n) \end{array}\right) \begin{array}{l} \text{pos. eff. on } p_1 \\ \text{pos. eff. on } p_2 \\ \phantom{x} \\ \phantom{x} \end{array}$$

where, for any i: $t_i = \langle I_i, O_i \rangle$

# Incidence Matrix

- The global effect of every transition can be summarised as a single matrix:

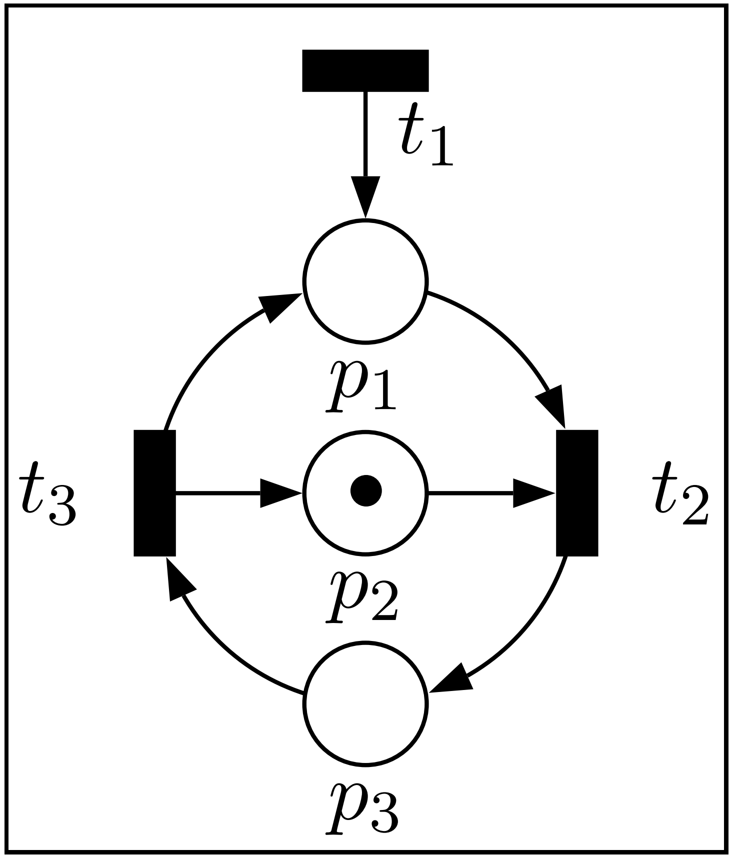
$$W = W^+ - W^-$$

W is called the incidence matrix of the net

# Example

$$W^+ = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad W^- = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

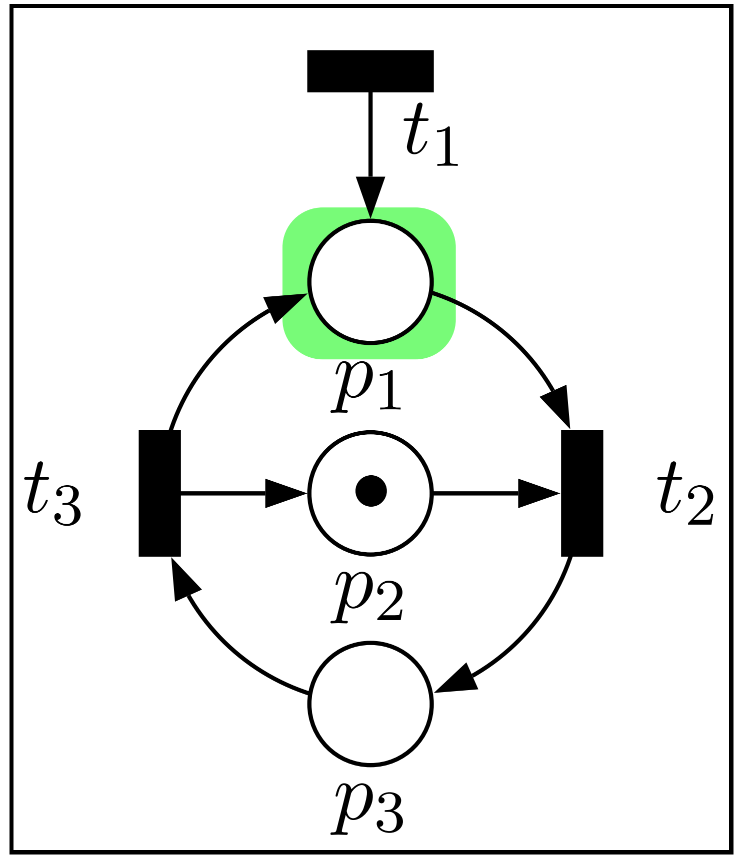
$$W = \begin{pmatrix} 1 & -1 & 1 \\ 0 & -1 & 1 \\ 0 & 1 & -1 \end{pmatrix}$$



# Example

$$W^+ = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad W^- = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

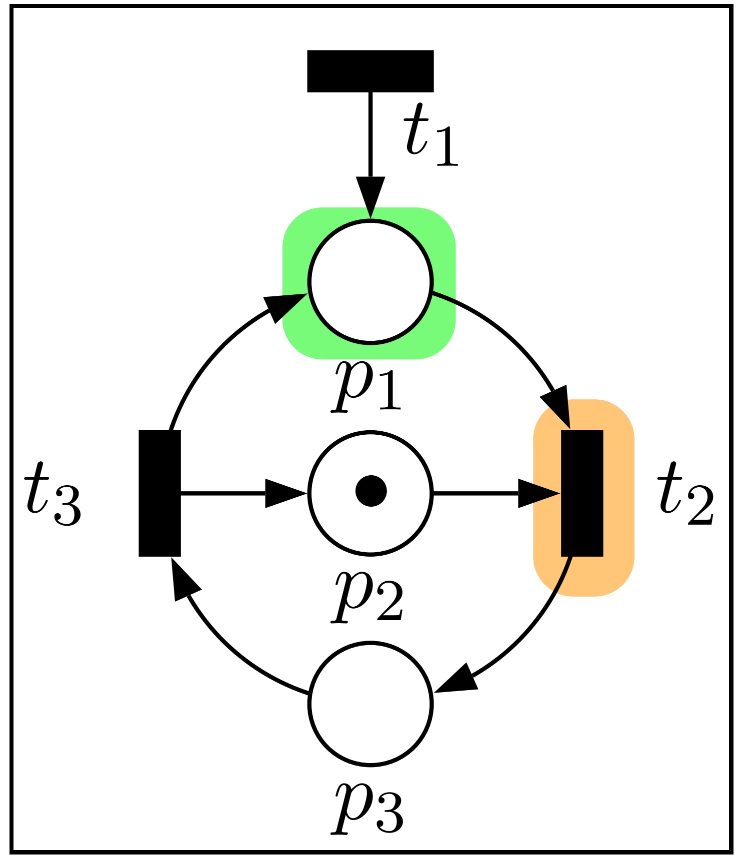
$$W = \begin{pmatrix} 1 & -1 & 1 \\ 0 & -1 & 1 \\ 0 & 1 & -1 \end{pmatrix}$$



# Example

$$W^+ = \begin{pmatrix} 1\,0\,1 \\ 0\,0\,1 \\ 0\,1\,0 \end{pmatrix} \quad W^- = \begin{pmatrix} 0\,1\,0 \\ 0\,1\,0 \\ 0\,0\,1 \end{pmatrix}$$

$$W = \begin{pmatrix} 1 & -1 & 1 \\ 0 & -1 & 1 \\ 0 & 1 & -1 \end{pmatrix}$$



# Computing place invariants

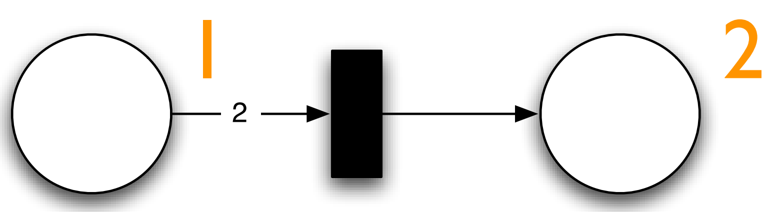
- Intuitively, if i is a place invariant it should assign weights to the places such that the positive and negative effects of every transition are balanced
- Thus, for any transition t = $\langle I, O \rangle$ we should have:

$$\sum_{p \in P} I(p) \times i(p) = \sum_{p \in P} O(p) \times i(p)$$

# Computing place invariants

- Intuitively, if i is a place invariant it should assign weights to the places such that the positive and negative effects of every transition are balanced
- Thus, for any transition t = ⟨I, O⟩ we should have:

$$\sum_{p \in P} I(p) \times i(p) = \sum_{p \in P} O(p) \times i(p)$$

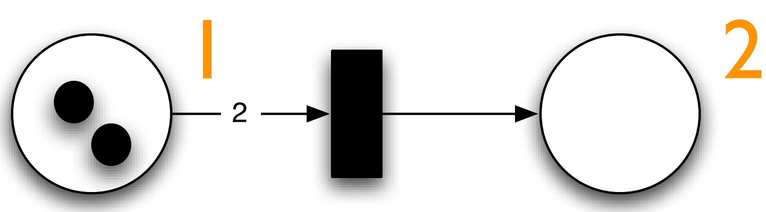


# Computing place invariants

- Intuitively, if i is a place invariant it should assign weights to the places such that the positive and negative effects of every transition are balanced

- Thus, for any transition t = $\langle I, O \rangle$ we should have:

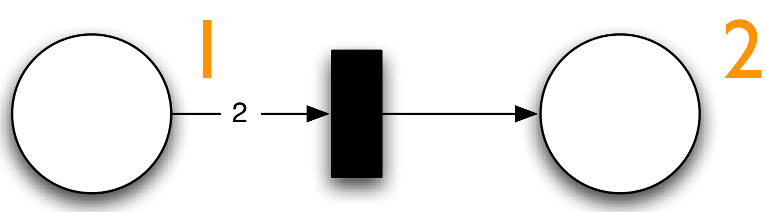
$$\sum_{p \in P} I(p) \times i(p) = \sum_{p \in P} O(p) \times i(p)$$



# Computing place invariants

- Intuitively, if i is a place invariant it should assign weights to the places such that the positive and negative effects of every transition are balanced
- Thus, for any transition t = $\langle I, O \rangle$ we should have:

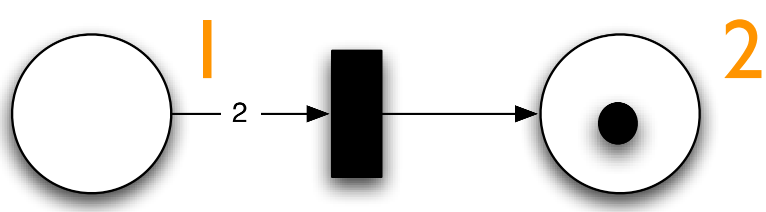
$$\sum_{p \in P} I(p) \times i(p) = \sum_{p \in P} O(p) \times i(p)$$



# Computing place invariants

- Intuitively, if i is a place invariant it should assign weights to the places such that the positive and negative effects of every transition are balanced
- Thus, for any transition t = ⟨I, O⟩ we should have:

$$\sum_{p \in P} I(p) \times i(p) = \sum_{p \in P} O(p) \times i(p)$$



# Computing place invariants

$$\sum_{p \in P} I(p) \times i(p) = \sum_{p \in P} O(p) \times i(p)$$

means

$$\sum_{p \in P} \big(O(p) - I(p)\big) \times i(p) = 0$$

# Computing place invariants

$$\sum_{p \in P} I(p) \times i(p) = \sum_{p \in P} O(p) \times i(p)$$

means

$$\sum_{p \in P} \big( O(p) - I(p) \big) \times i(p) = 0$$

t = $\langle$I, O$\rangle$

# Computing place invariants

$$\sum_{p \in P} I(p) \times i(p) = \sum_{p \in P} O(p) \times i(p)$$

means

$$\sum_{p \in P} \big(O(p) - I(p)\big) \times i(p) = 0$$

t = $\langle I, O \rangle$

$$W = \begin{pmatrix} \cdots & O(p_1) - I(p_1) & \cdots \\ \cdots & O(p_2) - I(p_2) & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & O(p_n) - I(p_n) & \cdots \end{pmatrix}$$

# Computing place invariants

$$\sum_{p \in P} I(p) \times i(p) = \sum_{p \in P} O(p) \times i(p)$$

means

$$\sum_{p \in P} \big(O(p) - I(p)\big) \times i(p) = 0$$

t = ⟨I, O⟩

$$W = \begin{pmatrix} \cdots & O(p_1) - I(p_1) & \cdots \\ \cdots & O(p_2) - I(p_2) & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & O(p_n) - I(p_n) & \cdots \end{pmatrix}$$

# Computing place invariants

$$\sum_{p \in P} \big( O(p) - I(p) \big) \times i(p) = 0$$

is thus the scalar product of i and the column of W that corresponds to transition t

# Computing place invariants

$$\sum_{p \in P} \big( O(p) - I(p) \big) \times i(p) = 0$$

is thus the scalar product of i and the column of W that corresponds to transition t

Since this must hold for any t, we obtain:

# Computing place invariants

$$\sum_{p \in P} \big( O(p) - I(p) \big) \times i(p) = 0$$

is thus the scalar product of i and the column of W that corresponds to transition t

Since this must hold for any t, we obtain:

Theorem: any solution i to the following system of equations is a place-invariant:

# Computing place invariants

$$\sum_{p \in P} \left( O(p) - I(p) \right) \times i(p) = 0$$

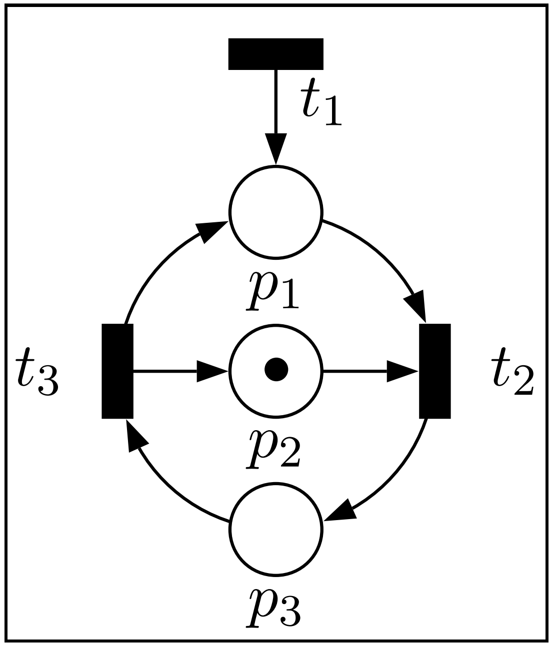
is thus the scalar product of i and the column of W that corresponds to transition t

Since this must hold for any t, we obtain:

Theorem: any solution i to the following system of equations is a place-invariant:

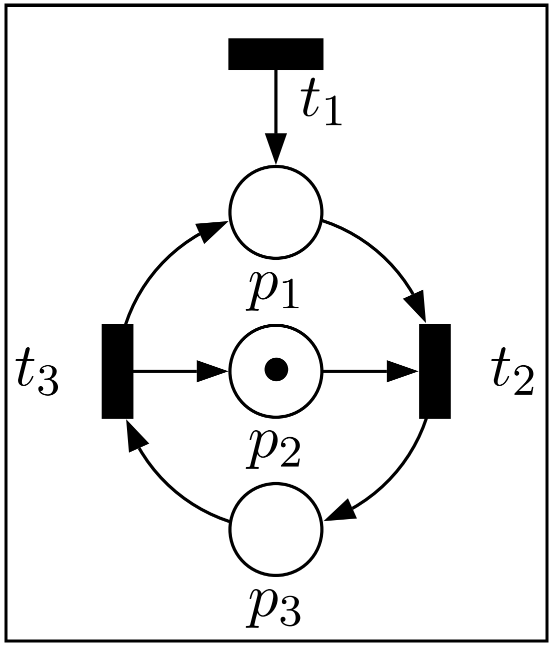
$$i \times W = 0$$

# Example




$$W = \begin{pmatrix} 1 & -1 & 1 \\ 0 & -1 & 1 \\ 0 & 1 & -1 \end{pmatrix}$$

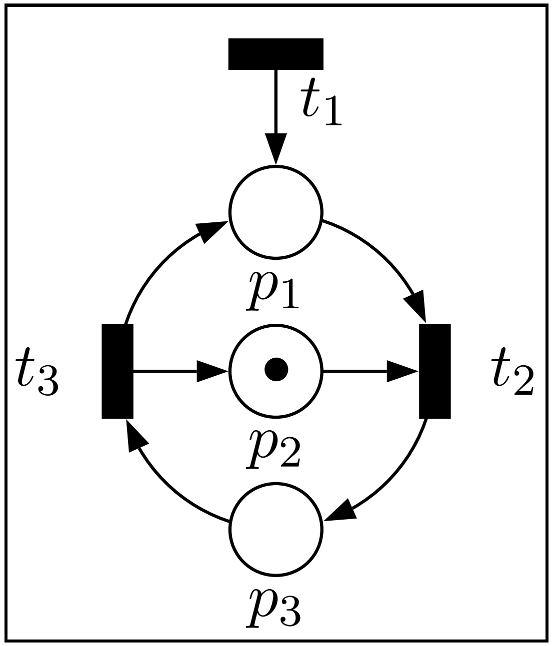
# Example




$$W = \begin{pmatrix} 1 & -1 & 1 \\ 0 & -1 & 1 \\ 0 & 1 & -1 \end{pmatrix}$$

$$\langle i_1, i_2, i_3 \rangle \times W = 0$$

# Example

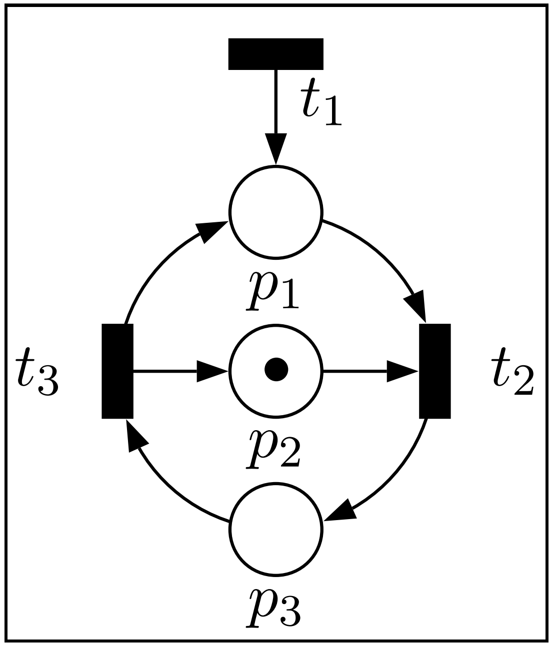



$$W = \begin{pmatrix} 1 & -1 & 1 \\ 0 & -1 & 1 \\ 0 & 1 & -1 \end{pmatrix}$$

$$\langle i_1, i_2, i_3 \rangle \times W = 0$$

$$\begin{cases} i_1 & = 0 \\ -i_1 - i_2 + i_3 & = 0 \\ i_1 + i_2 - i_3 & = 0 \end{cases}$$

# Example

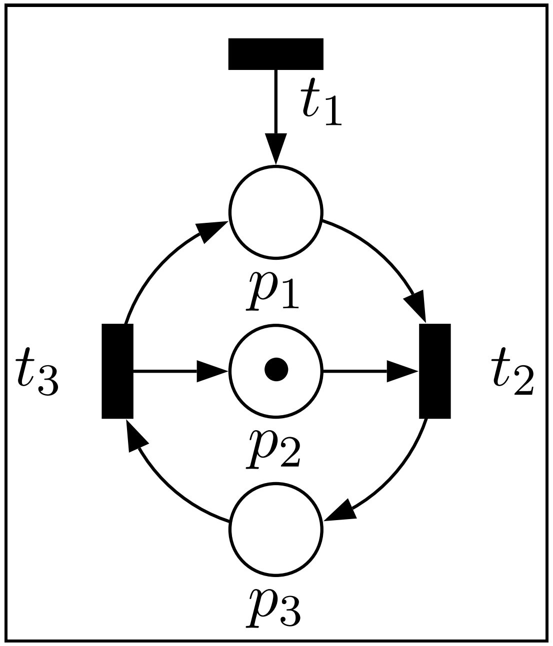



$$W = \begin{pmatrix} 1 & -1 & 1 \\ 0 & -1 & 1 \\ 0 & 1 & -1 \end{pmatrix}$$

$$\langle i_1, i_2, i_3 \rangle \times W = 0$$

$$\begin{cases} i_1 & = 0 \\ -i_1 - i_2 + i_3 & = 0 \\ i_1 + i_2 - i_3 & = 0 \end{cases} \qquad \begin{cases} i_1 & = 0 \\ -i_2 + i_3 & = 0 \\ +i_2 - i_3 & = 0 \end{cases}$$

# Example




$$W = \begin{pmatrix} 1 & -1 & 1 \\ 0 & -1 & 1 \\ & & -1 \end{pmatrix}$$

Any vector of the form

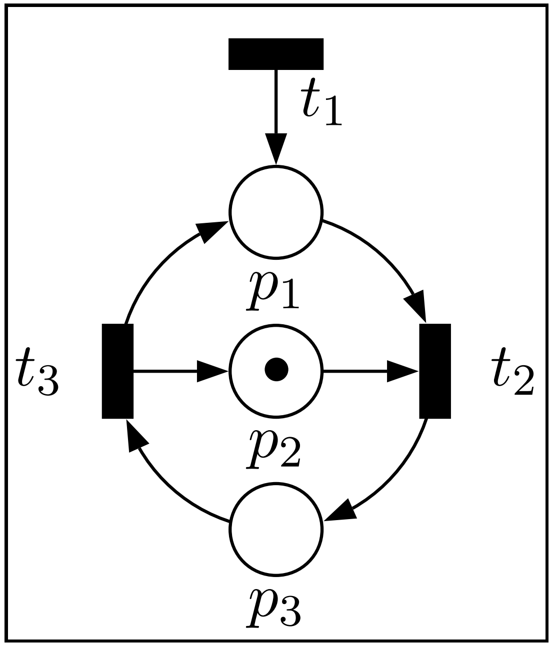
⟨0, i, i⟩

is a place invariant

$$\langle i_1, i_2, i_3\rangle \times W = 0$$

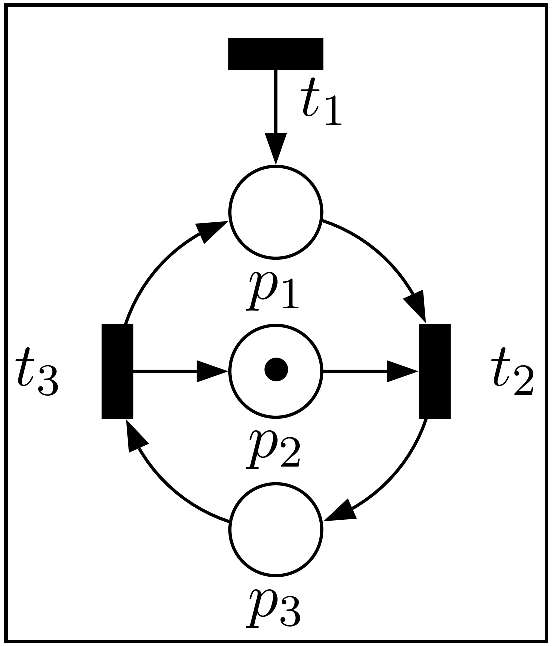
$$\begin{cases} i_1 = 0 \\ -i_1 - i_2 + i_3 = 0 \\ i_1 + i_2 - i_3 = 0 \end{cases} \qquad \begin{cases} i_1 = 0 \\ -i_2 + i_3 = 0 \\ +i_2 - i_3 = 0 \end{cases}$$

# Proving properties




Let us choose $\langle 0, 1, 1 \rangle$
as place-invariant

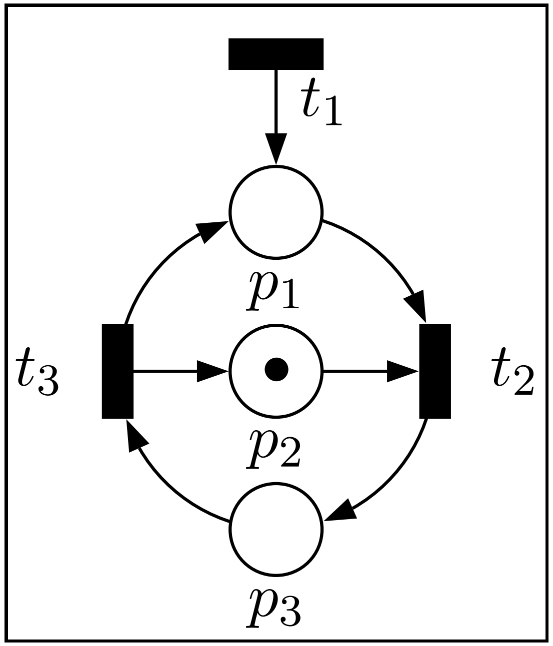
# Proving properties




Let us choose $\langle 0, 1, 1 \rangle$ as place-invariant

This means that $p_2$ and $p_3$ are bounded !

# Proving properties




Let us choose $\langle 0, 1, 1 \rangle$
as place-invariant

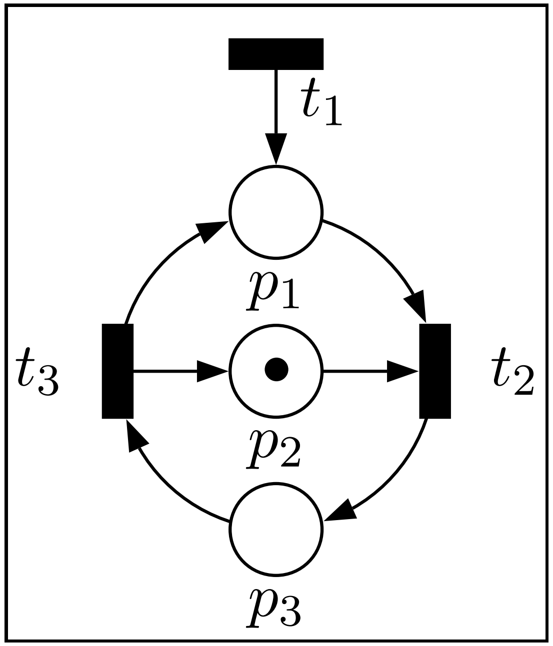
This means that $p_2$ and $p_3$ are
bounded !

For any reachable marking m:

$$0\ m(p_1) + 1\ m(p_2) + 1\ m(p_3) = 0\ m_0(p_1) + 1\ m_0(p_2) + 1\ m_0(p_3)$$

$$m(p_2) + m(p_3) = 1$$

# Proving properties




Let us choose $\langle 0, 1, 1 \rangle$ as place-invariant

This means that $p_2$ and $p_3$ are bounded !

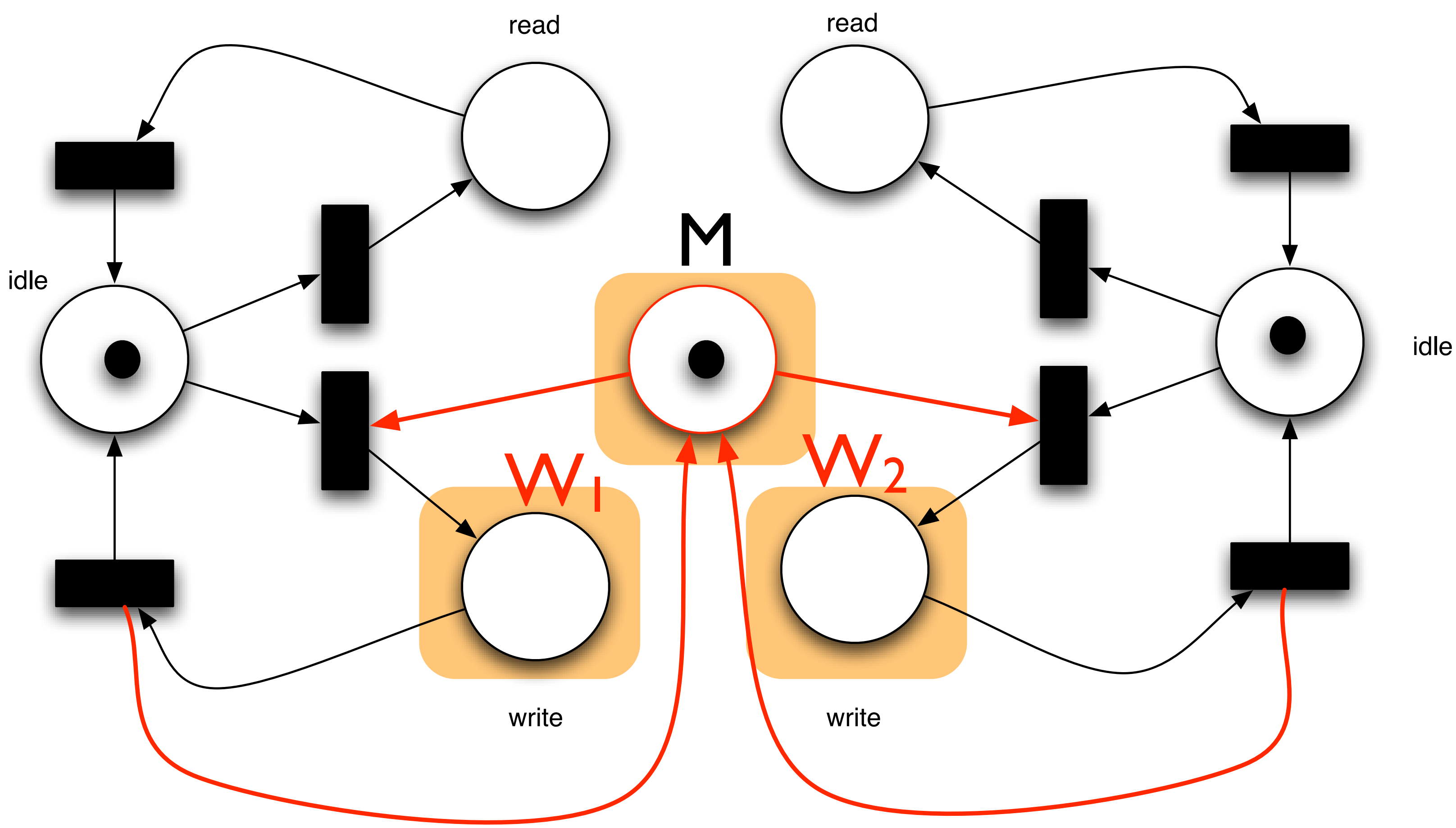
For any reachable marking m:

$$0\ m(p_1) + 1\ m(p_2) + 1\ m(p_3) = 0\ m_0(p_1) + 1\ m_0(p_2) + 1\ m_0(p_3)$$

$$m(p_2) + m(p_3) = 1$$

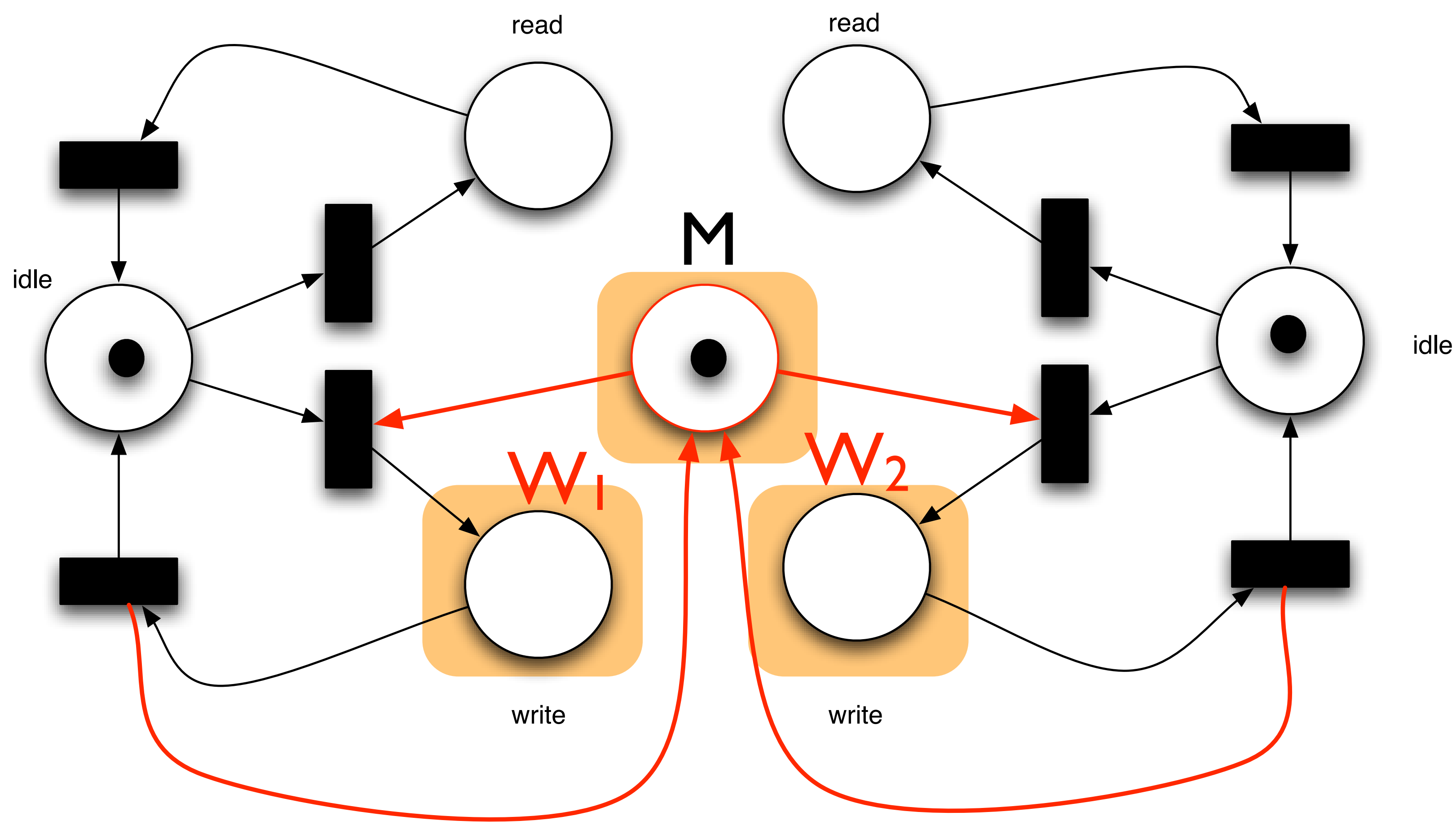
Hence, mutual exclusion is enforced !

# Proving properties




$i(M) = i(W_1) = i(W_2) = 1$ and $i(p) = 0$ otherwise
is a place invariant

# Proving properties




$i(M) = i(W_1) = i(W_2) = 1$ and $i(p) = 0$ otherwise

is a place invariant

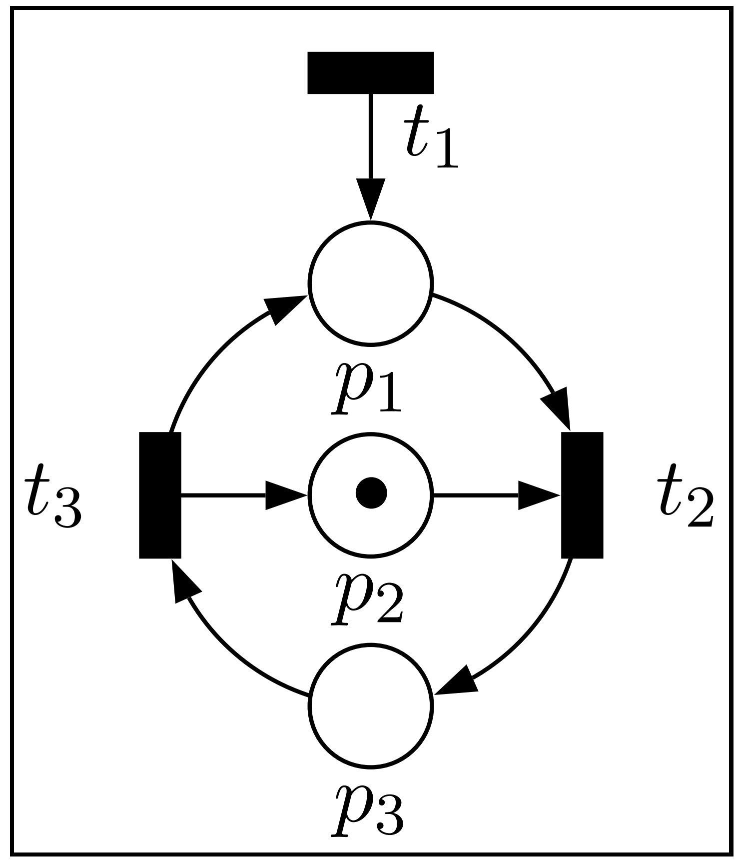
Hence, mutual exclusion is enforced !

# Karp & Miller
# and
# the coverability set

# The reachability tree revisited

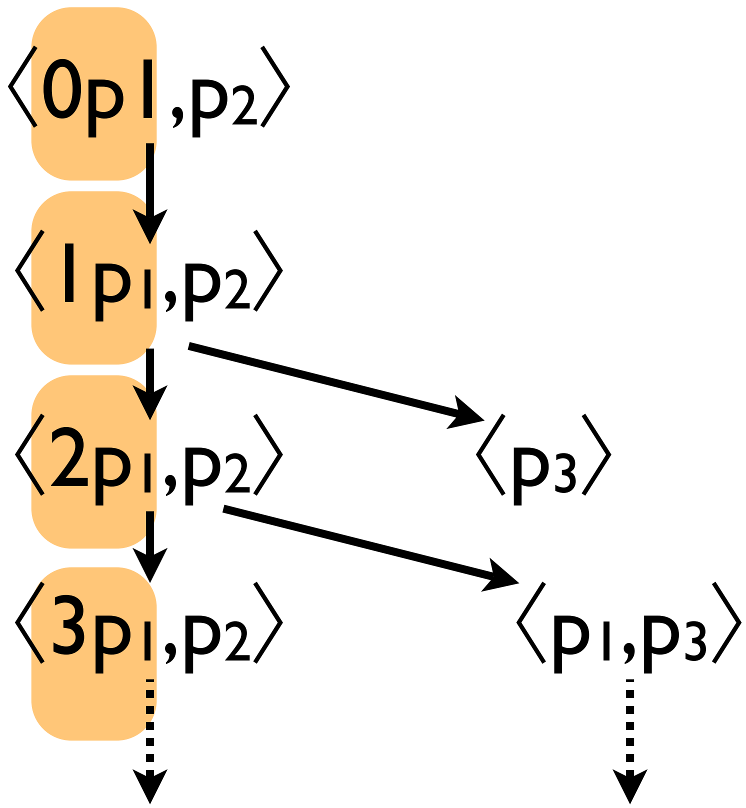
- Reminder: reachability trees can be infinite

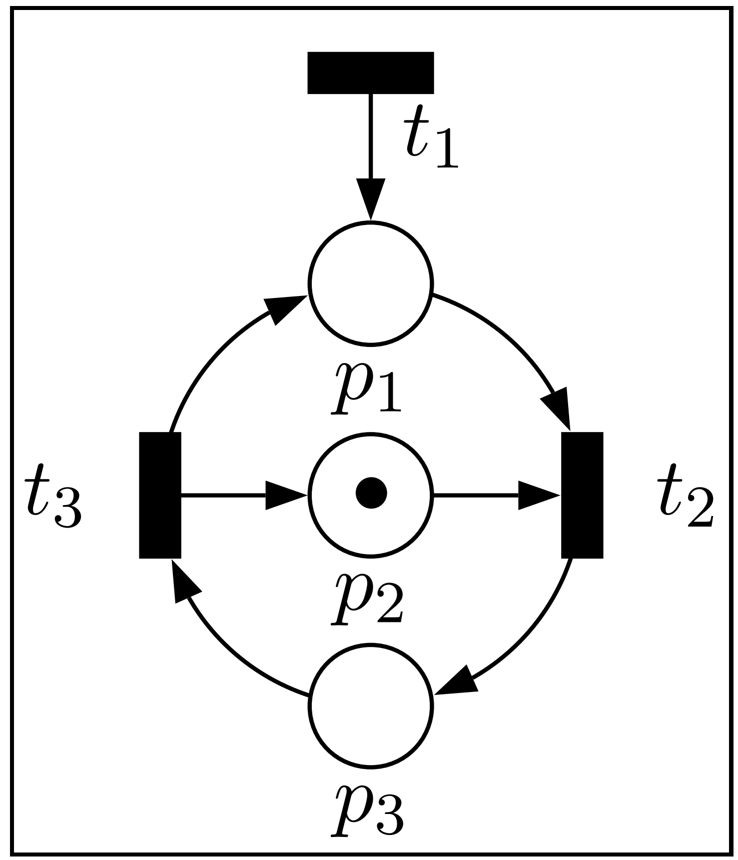
$$\begin{tikzcd}
\langle 0p_1,p_2\rangle \arrow[d] & \\
\langle 1p_1,p_2\rangle \arrow[d] \arrow[dr] & \\
\langle 2p_1,p_2\rangle \arrow[d] \arrow[dr] & \langle p_3\rangle \\
\langle 3p_1,p_2\rangle \arrow[d, dotted] & \langle p_1,p_3\rangle \arrow[d, dotted] \\
{} & {}
\end{tikzcd}$$



# The reachability tree revisited

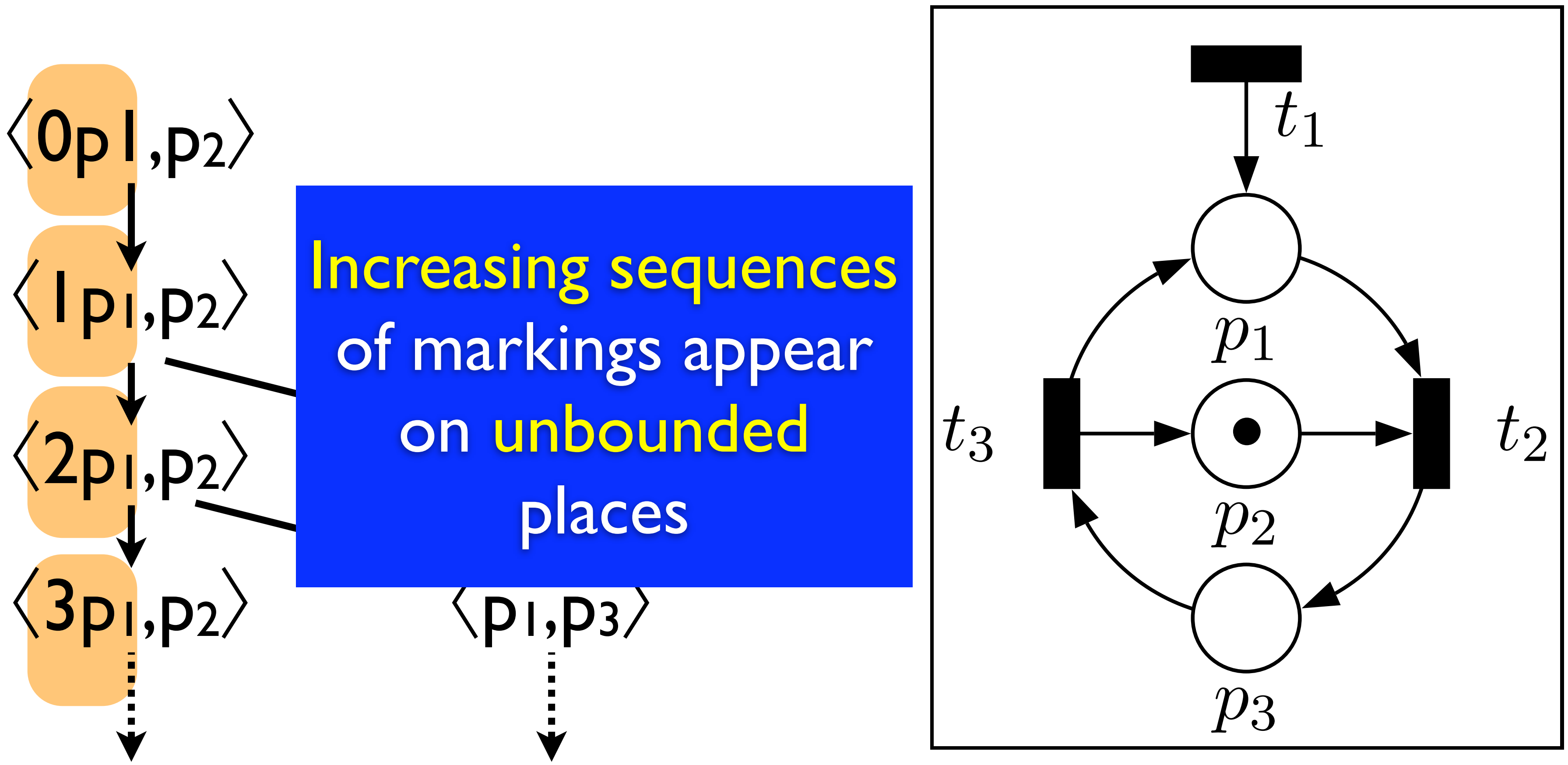
- Reminder: reachability trees can be infinite



# The reachability tree revisited

- Reminder: reachability trees can be infinite



# The reachability tree revisited

- Let us summarise this infinite sequence

$$\langle 0p_1, p_2 \rangle$$
$$\downarrow$$
$$\langle 1p_1, p_2 \rangle$$
$$\downarrow$$
$$\langle 2p_1, p_2 \rangle$$
$$\downarrow$$
$$\langle 3p_1, p_2 \rangle$$
$$\vdots$$

# The reachability tree revisited

- Let us summarise this infinite sequence

$$
\begin{array}{c}
\langle 0p_1, p_2 \rangle \\
\downarrow \\
\langle 1p_1, p_2 \rangle \\
\downarrow \\
\langle 2p_1, p_2 \rangle \\
\downarrow \\
\langle 3p_1, p_2 \rangle \\
\vdots
\end{array}
\quad \xrightarrow{\quad \text{limit} \quad}
$$

# The reachability tree revisited

- Let us summarise this infinite sequence

$$
\begin{array}{c}
\langle 0p_1, p_2 \rangle \\
\downarrow \\
\langle 1p_1, p_2 \rangle \\
\downarrow \\
\langle 2p_1, p_2 \rangle \\
\downarrow \\
\langle 3p_1, p_2 \rangle \\
\vdots
\end{array}
\quad \xrightarrow{\text{limit}} \quad \langle \omega p_1, p_2 \rangle
$$

# The reachability tree revisited

- Let us summarise this infinite sequence

$\omega$ must be regarded as: "any number of tokens"

$$\begin{array}{c} \langle 0p_1,p_2\rangle \\ \downarrow \\ \langle 1p_1,p_2\rangle \\ \downarrow \\ \langle 2p_1,p_2\rangle \\ \downarrow \\ \langle 3p_1,p_2\rangle \\ \vdots \end{array} \xrightarrow{\text{limit}} \langle \omega p_1,p_2\rangle$$

# The reachability tree revisited

- Let us summarise this infinite sequence

$$\langle 0p_1, p_2\rangle \to \langle 1p_1, p_2\rangle \to \langle 2p_1, p_2\rangle \to \langle 3p_1, p_2\rangle \dashrightarrow$$

$$\xrightarrow{\text{limit}} \langle \omega p_1, p_2\rangle$$

$\omega$ must be regarded as: "any number of tokens"

Main idea of the Karp and Miller algorithm

# Karp & Miller

- Propose in 1969 a solution to detect unbounded places of a Petri net

# Monotonicity

- Petri nets induce (strongly) monotonic transition systems:

$$\begin{tikzcd} m_3 \arrow[d, phantom, "\curlyvee"] & \\ m_1 \arrow[r, "t"] & m_2 \end{tikzcd}$$

- In particular:

if

# Monotonicity

- Petri nets induce (strongly) monotonic transition systems:

$$\begin{array}{ccc} m_3 & & m_4 \\ \succcurlyeq & & \\ m_1 & \xrightarrow{t} & m_2 \end{array}$$

- In particular:

if

# Monotonicity

- Petri nets induce (strongly) monotonic transition systems:

$$\begin{tikzcd} m_3 \arrow[r, "t"] & m_4 \\ m_1 \arrow[u, phantom, "\curlyvee" description] \arrow[r, "t"] & m_2 \arrow[u, phantom, "\curlyvee" description] \end{tikzcd}$$

- In particular:

if

# Monotonicity

- Petri nets induce (strongly) monotonic transition systems:

$$\begin{array}{ccc} m_3 & \xrightarrow{t} & m_4 \\ \curlyvee & & \curlyvee \\ m_1 & \xrightarrow{t} & m_2 \end{array}$$

- In particular:

if

# Monotonicity

- Petri nets induce (strongly) monotonic transition systems:

$$\begin{array}{ccc} m_3 & \xrightarrow{\quad t \quad} & m_4 \\ \curlyvee & & \curlyvee \\ m_1 & \xrightarrow{\quad t \quad} & m_2 \end{array}$$

- In particular:

if $\langle i_1, i_2, i_3 \rangle$

# Monotonicity

- Petri nets induce (strongly) monotonic transition systems:

$$\begin{tikzcd} m_3 \arrow[r, "t"] & m_4 \\ m_1 \arrow[r, "t"] \arrow[u, phantom, "\curlyvee"] & m_2 \arrow[u, phantom, "\curlyvee"] \end{tikzcd}$$

- In particular:

if $\langle i_1, i_2, i_3 \rangle$ $\langle i'_1, i'_2, i'_3 \rangle$

# Monotonicity

- Petri nets induce (strongly) monotonic transition systems:

$$\begin{tikzcd} m_3 \arrow[r, "t"] & m_4 \\ m_1 \arrow[r, "t"] \arrow[u, phantom, "\curlyvee"] & m_2 \arrow[u, phantom, "\curlyvee"] \end{tikzcd}$$

- In particular:

if $\langle i_1, i_2, i_3 \rangle \longrightarrow \langle i'_1, i'_2, i'_3 \rangle$

# Monotonicity

- Petri nets induce (strongly) monotonic transition systems:

$$
\begin{tikzcd}
m_3 \arrow[r, "t"] \arrow[d, phantom, "\rotatebox[origin=c]{-90}{$\succ$}"] & m_4 \arrow[d, phantom, "\rotatebox[origin=c]{-90}{$\succ$}"] \\
m_1 \arrow[r, "t"] & m_2
\end{tikzcd}
$$

- In particular:

if $\langle i_1, i_2, i_3 \rangle \longrightarrow \langle i_1', i_2', i_3' \rangle$

# Monotonicity

- Petri nets induce (strongly) monotonic transition systems:

$$\begin{tikzcd} m_3 \arrow[r, "t"] \arrow[d, phantom, "\rotatebox{-90}{$\succ$}"] & m_4 \arrow[d, phantom, "\rotatebox{-90}{$\succ$}"] \\ m_1 \arrow[r, "t"] & m_2 \end{tikzcd}$$

- In particular:

if $$\begin{tikzcd} \langle i_1, i_2, i_3 \rangle \arrow[r, "\prec"] \arrow[r, bend right, dash, "<"'] & \langle i'_1, i'_2, i'_3 \rangle \end{tikzcd}$$

# Monotonicity

- Petri nets induce (strongly) monotonic transition systems:

$$\begin{tikzcd} m_3 \arrow[r, "t"] & m_4 \\ m_1 \arrow[r, "t"] \arrow[u, phantom, "\curlyvee"] & m_2 \arrow[u, phantom, "\curlyvee"] \end{tikzcd}$$

- In particular:

if $\langle i_1, i_2, i_3 \rangle \xrightarrow{\ \prec\ } \langle i'_1, i'_2, i'_3 \rangle$

$<$

then $p_2$ is unbounded

# Example

$\langle 1, 0, 0, 0 \rangle$

# Example

$$
\begin{tikzcd}
\langle 1, 0, 0, 0 \rangle \arrow[dr, dotted] & \\
& \langle 0, 0, 1, 0 \rangle
\end{tikzcd}
$$

# Example

$\langle 1, 0, 0, 0 \rangle$

$\langle 0, 0, 1, 0 \rangle$

$\langle 0, 0, 0, 1 \rangle$

# Example

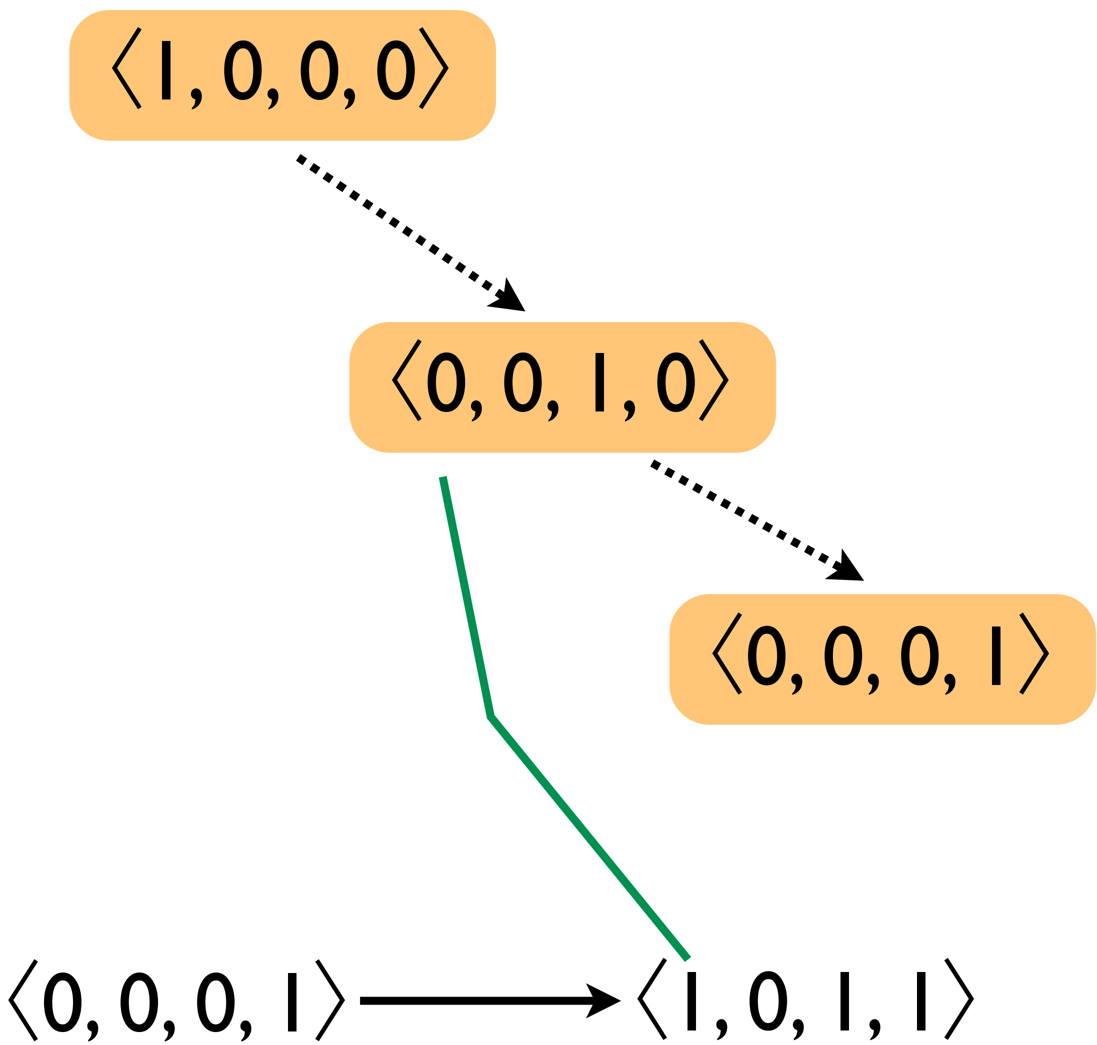
$$\langle 1, 0, 0, 0 \rangle$$

$$\dashrightarrow \langle 0, 0, 1, 0 \rangle$$

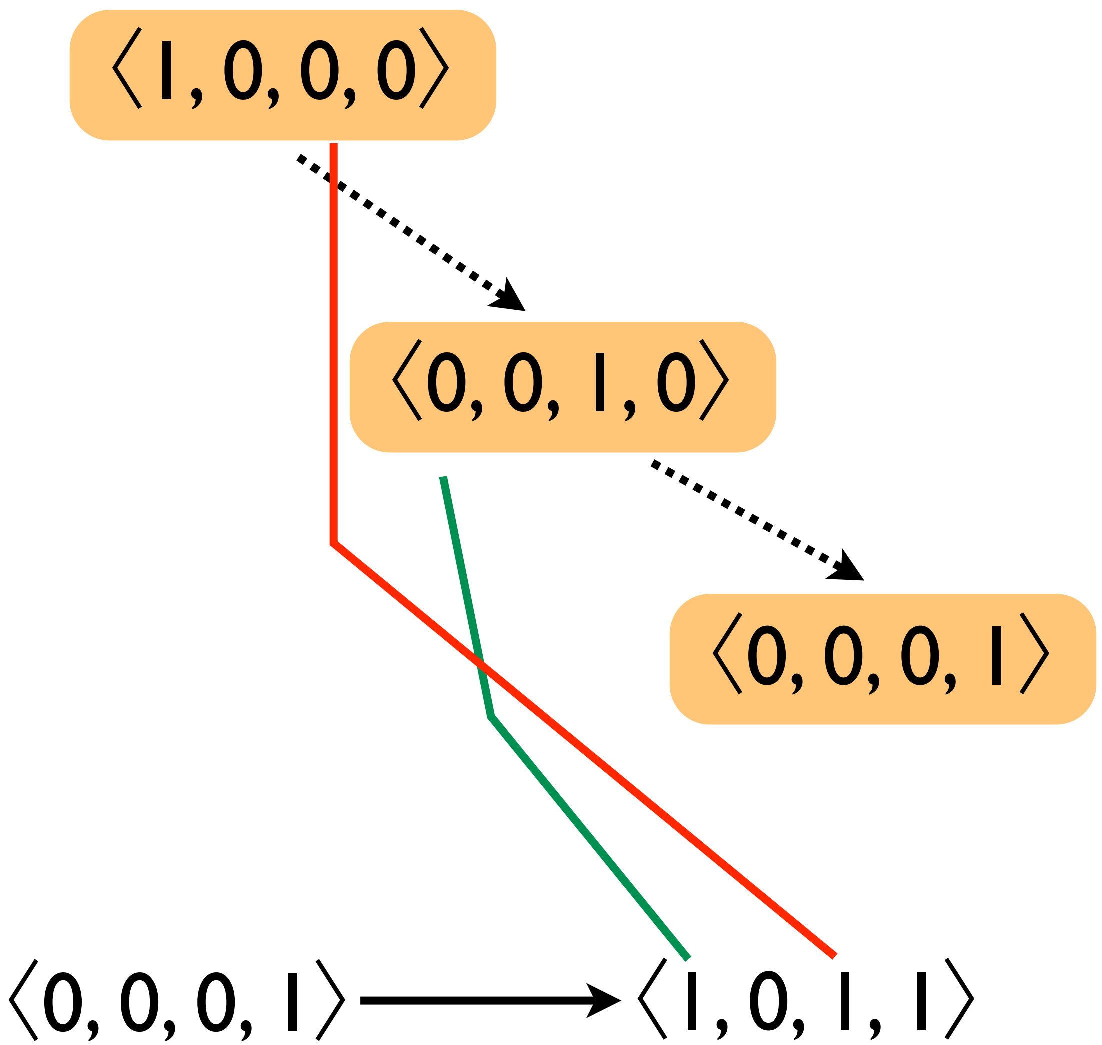
$$\dashrightarrow \langle 0, 0, 0, 1 \rangle$$

$$\langle 0, 0, 0, 1 \rangle \longrightarrow \langle 1, 0, 1, 1 \rangle$$

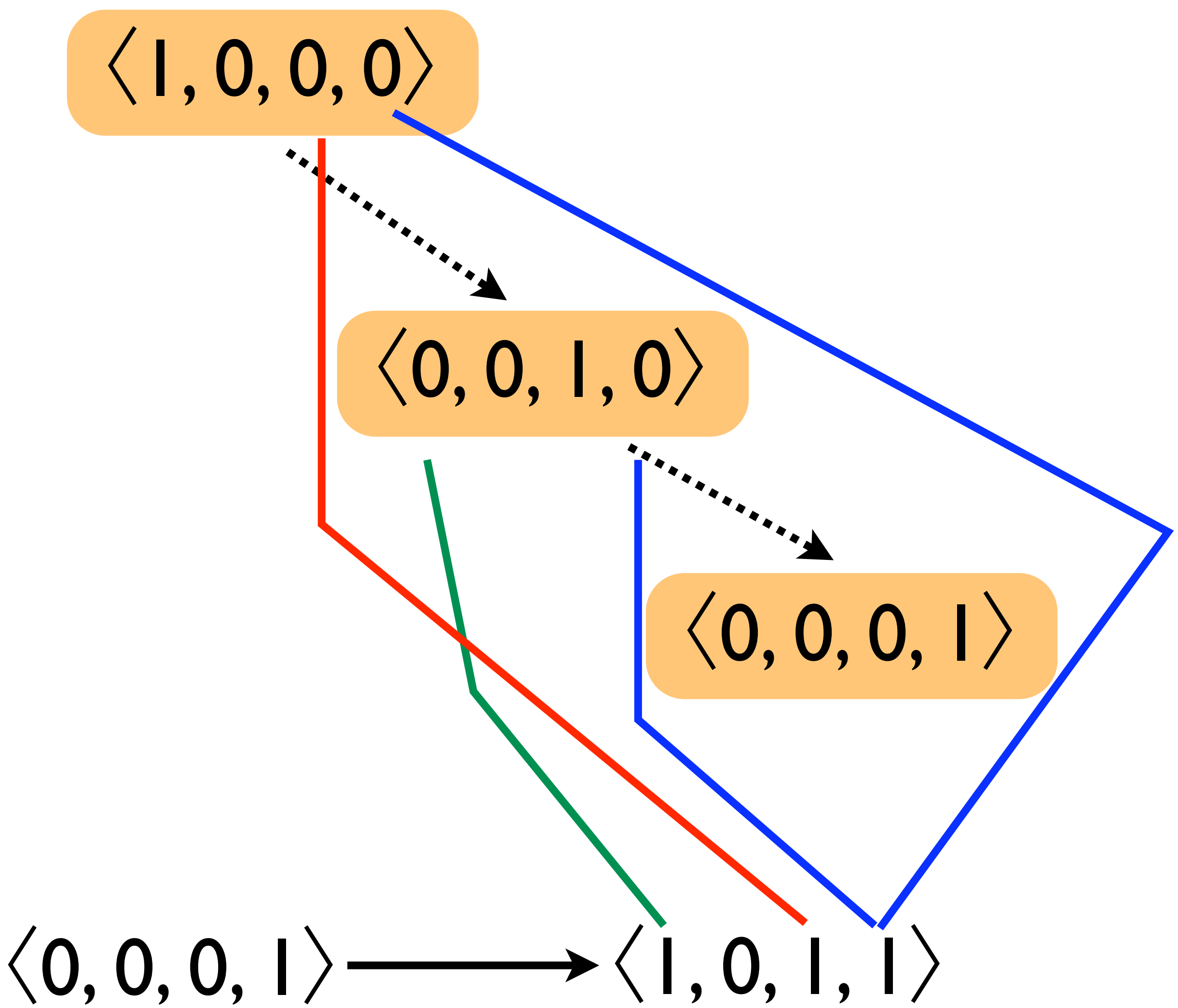
# Example



# Example

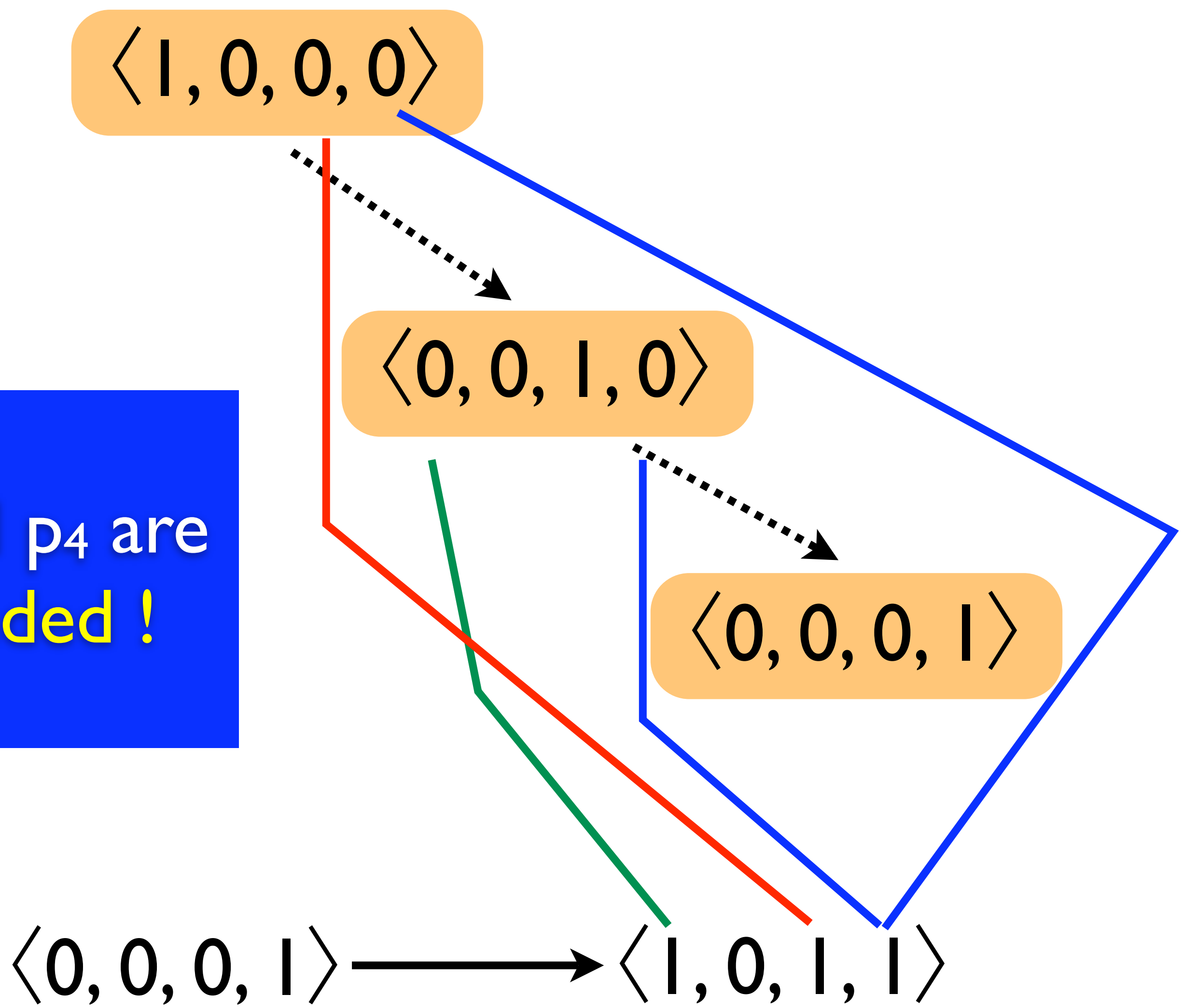


# Example



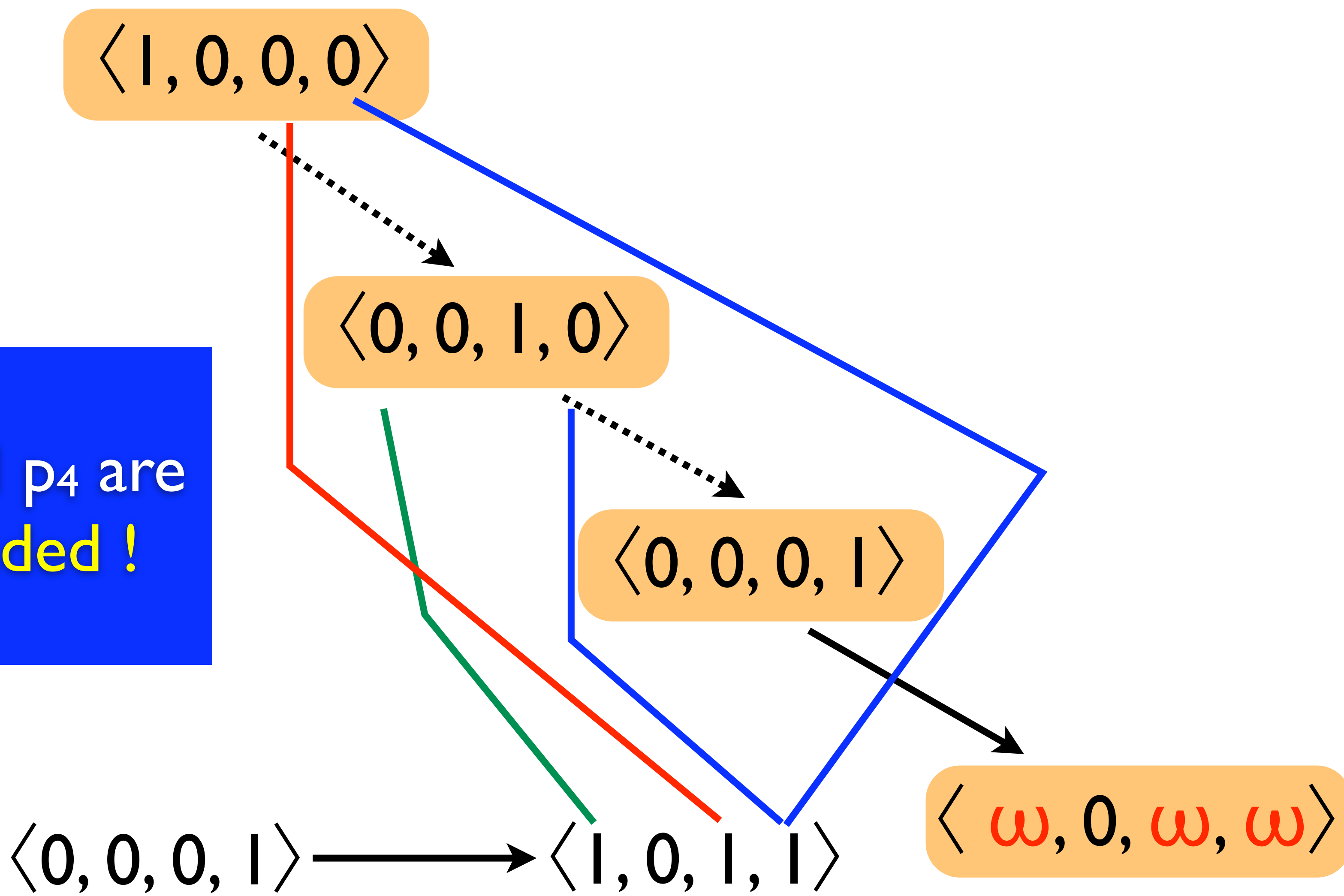
$\langle 1, 0, 0, 0 \rangle$
$\langle 0, 0, 1, 0 \rangle$
$\langle 0, 0, 0, 1 \rangle$
$\langle 0, 0, 0, 1 \rangle$
$\langle 1, 0, 1, 1 \rangle$

# Example



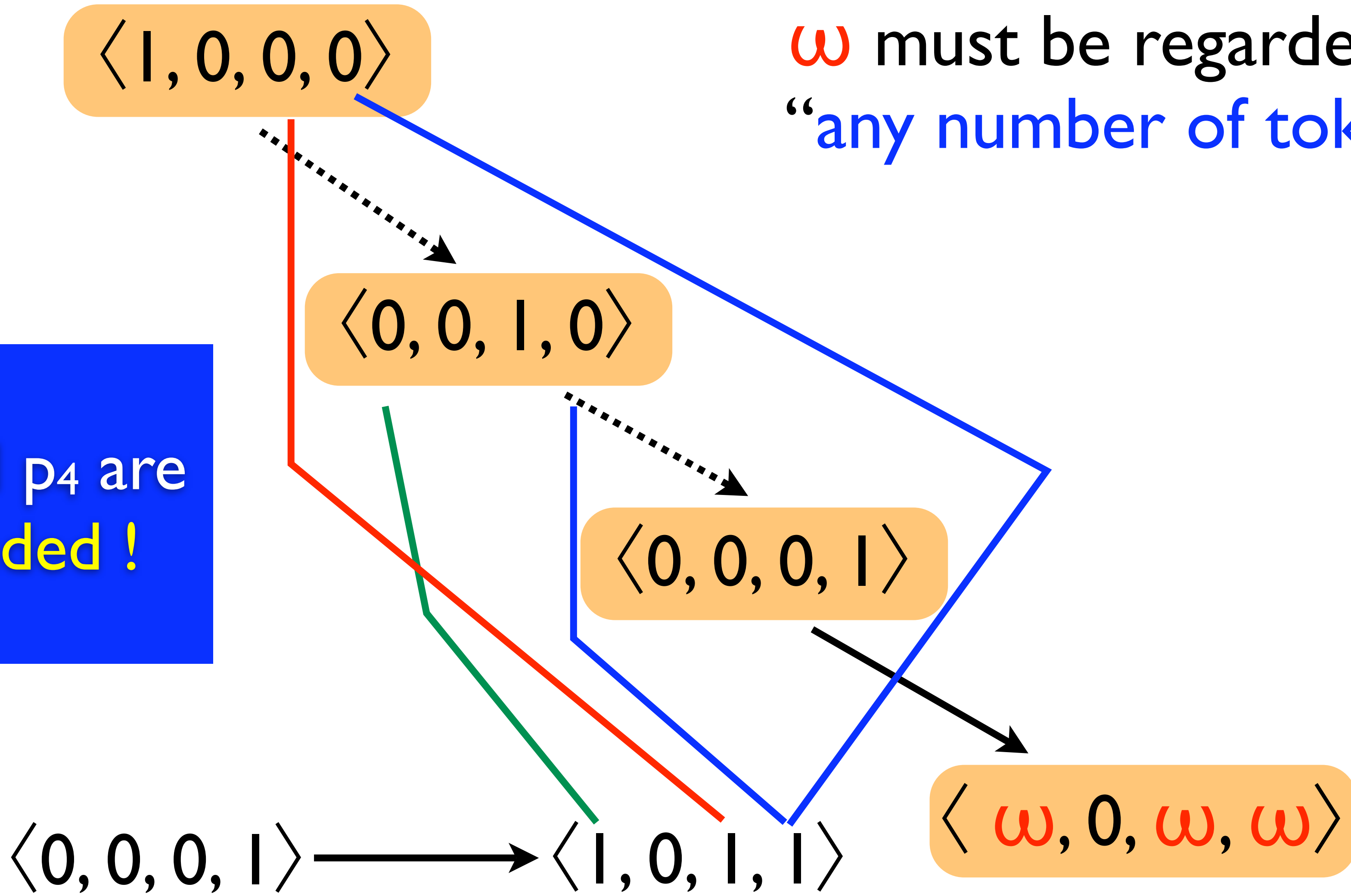

$p_1$, $p_3$ and $p_4$ are unbounded !

# Example




$p_1$, $p_3$ and $p_4$ are unbounded !

# Example



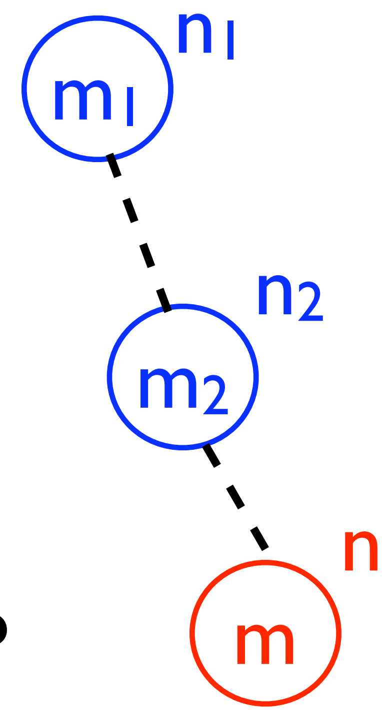

ω must be regarded as: "any number of tokens"

$p_1$, $p_3$ and $p_4$ are unbounded !

# Karp & Miller Acceleration

This is how we compute the successors of a node n:

**foreach** *Successor* $m'$ *of* $m$ **do**
$\quad m_\omega \leftarrow m'$;
$\quad$**foreach** *ancestor* $n_i$ *s.t.* $m_i \prec m'$ **do**
$\quad\quad$**foreach** *place* $p$ *s.t.* $m_i(p) < m'(p)$ **do**
$\quad\quad\quad m_\omega(p) \leftarrow \omega$;
$\quad$Add $m_\omega$ as child of $n$;



# Karp & Miller Acceleration

This is how we compute the successors of a node n:

**foreach** $\mathit{Successor}\ m'\ \mathit{of}\ m$ **do**

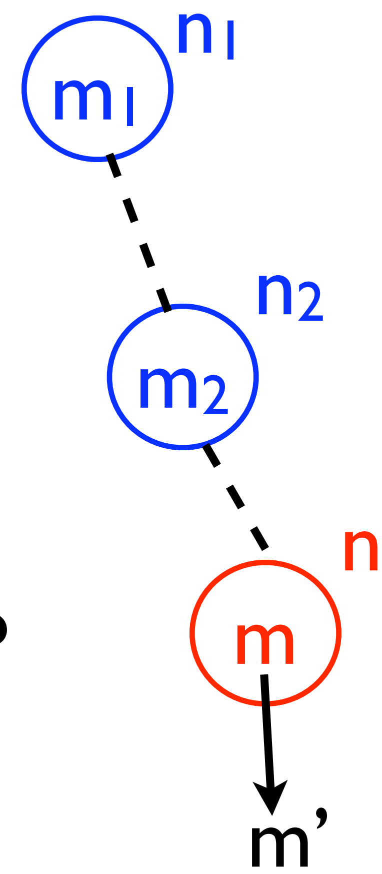
   $m_\omega \leftarrow m'$;

   **foreach** $\mathit{ancestor}\ n_i$ $s.t.\ m_i \prec m'$ **do**

      **foreach** $\mathit{place}\ p\ s.t.\ m_i(p) < m'(p)$ **do**

         $m_\omega(p) \leftarrow \omega$;

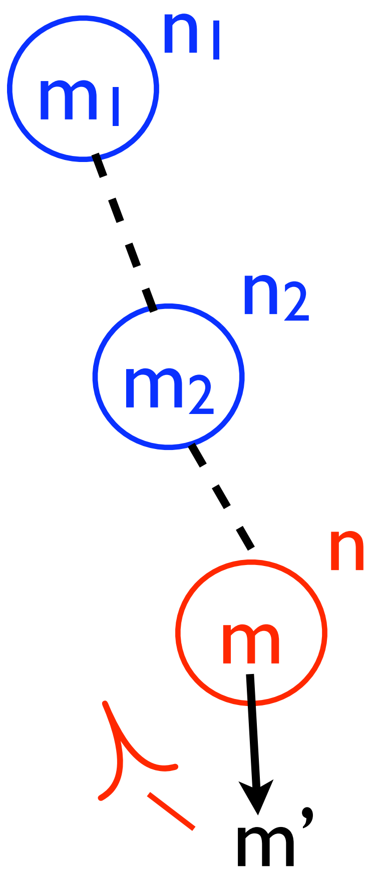
   Add $m_\omega$ as child of $n$;



# Karp & Miller Acceleration

This is how we compute the successors of a node n:

**foreach** *Successor* $m'$ *of* $m$ **do**
- $m_\omega \leftarrow m'$;
- **foreach** *ancestor* $n_i$ *s.t.* $m_i \prec m'$ **do**
  - **foreach** *place* $p$ *s.t.* $m_i(p) < m'(p)$ **do**
    - $m_\omega(p) \leftarrow \omega$;
- Add $m_\omega$ as child of $n$;



# Karp & Miller Acceleration

This is how we compute the successors of a node n:

**foreach** *Successor* $m'$ *of* $m$ **do**

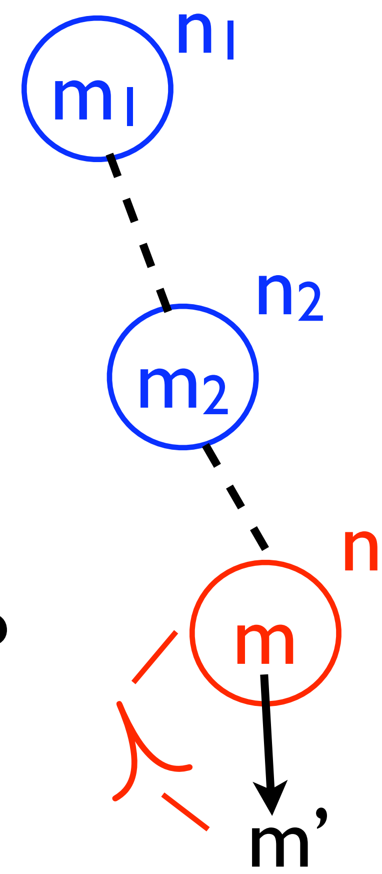
$\quad m_\omega \leftarrow m'$;

$\quad$ **foreach** *ancestor* $n_i$ *s.t.* $m_i \prec m'$ **do**

$\quad\quad$ **foreach** *place* $p$ *s.t.* $m_i(p) < m'(p)$ **do**

$\quad\quad\quad m_\omega(p) \leftarrow \omega$;

$\quad$ Add $m_\omega$ as child of $n$;



# Karp & Miller Acceleration

This is how we compute the successors of a node n:

**foreach** $Successor\ m'\ of\ m$ **do**

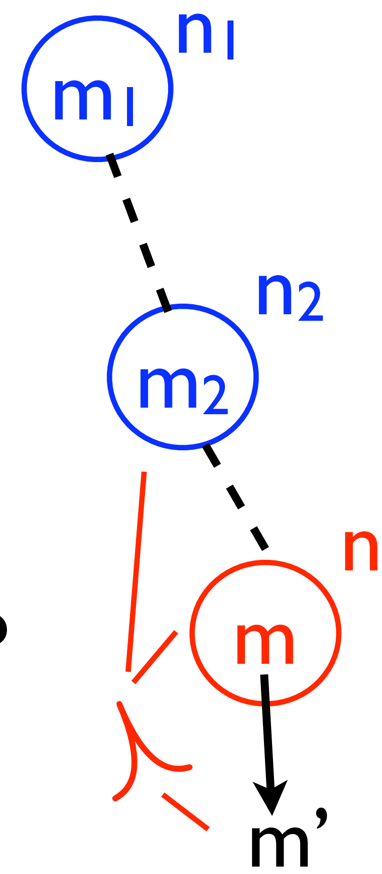
$\quad m_\omega \leftarrow m';$

$\quad$ **foreach** $ancestor\ n_i\ s.t.\ m_i \prec m'$ **do**

$\quad\quad$ **foreach** $place\ p\ s.t.\ m_i(p) < m'(p)$ **do**

$\quad\quad\quad m_\omega(p) \leftarrow \omega;$

$\quad$ Add $m_\omega$ as child of $n$;



# Karp & Miller Acceleration

This is how we compute the successors of a node n:

**foreach** *Successor $m'$ of $m$* **do**

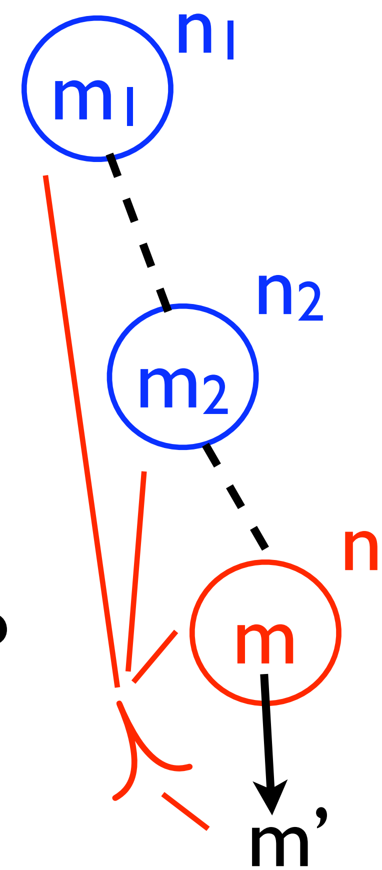
 $m_\omega \leftarrow m'$;

 **foreach** *ancestor $n_i$* *s.t.* $m_i \prec m'$ **do**

  **foreach** *place $p$ s.t.* $m_i(p) < m'(p)$ **do**

   $m_\omega(p) \leftarrow \omega$;

 Add $m_\omega$ as child of $n$;



# Karp & Miller Acceleration

This is how we compute the successors of a node n:

**foreach** $\mathit{Successor}\ m'\ \mathit{of}\ m$ **do**

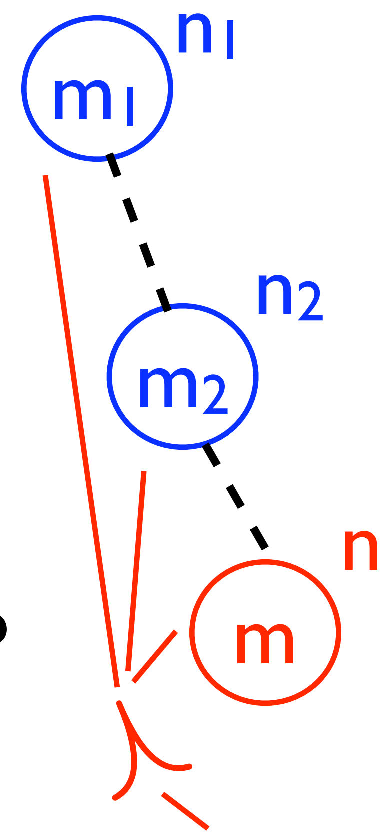
$\quad m_\omega \leftarrow m';$

$\quad$ **foreach** $\mathit{ancestor}\ n_i\ s.t.\ m_i \prec m'$ **do**

$\quad\quad$ **foreach** $\mathit{place}\ p\ s.t.\ m_i(p) < m'(p)$ **do**

$\quad\quad\quad m_\omega(p) \leftarrow \omega;$

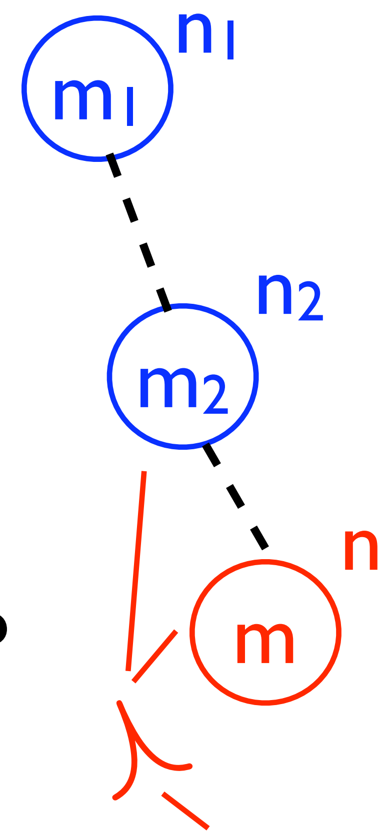
$\quad$ Add $m_\omega$ as child of $n$;



# Karp & Miller Acceleration

This is how we compute the successors of a node n:

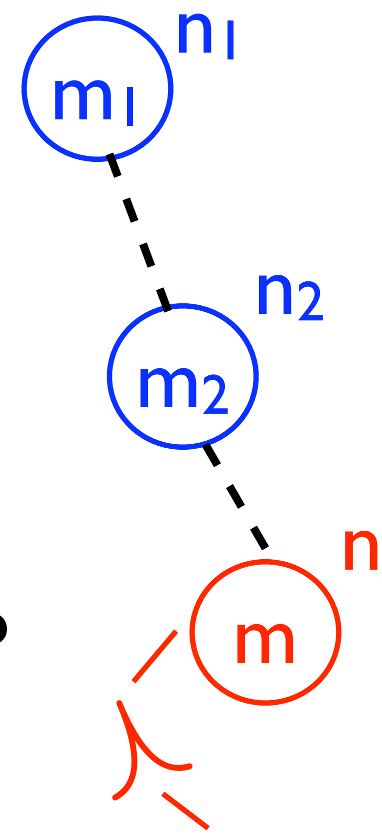
**foreach** *Successor* $m'$ *of* $m$ **do**
  $m_\omega \leftarrow m'$;
  **foreach** *ancestor* $n_i$ *s.t.* $m_i \prec m'$ **do**
    **foreach** *place* $p$ *s.t.* $m_i(p) < m'(p)$ **do**
      $m_\omega(p) \leftarrow \omega$;
  Add $m_\omega$ as child of $n$;



# Karp & Miller Acceleration

This is how we compute the successors of a node n:

**foreach** *Successor* $m'$ *of* $m$ **do**
- $m_\omega \leftarrow m'$;
- **foreach** *ancestor* $n_i$ *s.t.* $m_i \prec m'$ **do**
  - **foreach** *place* $p$ *s.t.* $m_i(p) < m'(p)$ **do**
    - $m_\omega(p) \leftarrow \omega$;
- Add $m_\omega$ as child of $n$;



# Karp & Miller Acceleration

This is how we compute the successors of a node n:

**foreach** *Successor* $m'$ *of* $m$ **do**

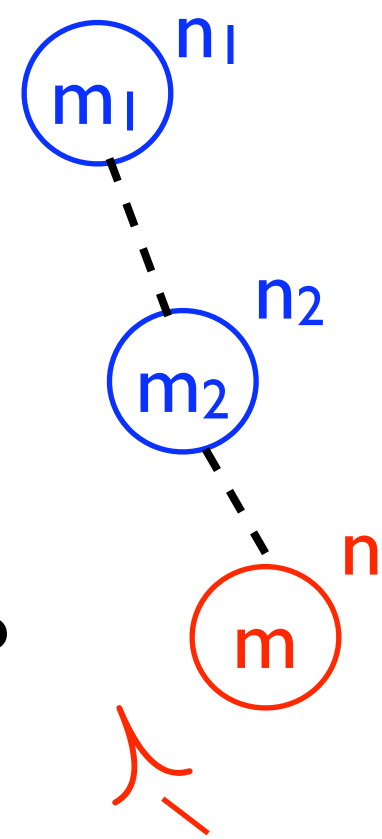
  $m_\omega \leftarrow m'$;

  **foreach** *ancestor* $n_i$ *s.t.* $m_i \prec m'$ **do**

    **foreach** *place* $p$ *s.t.* $m_i(p) < m'(p)$ **do**

      $m_\omega(p) \leftarrow \omega$;

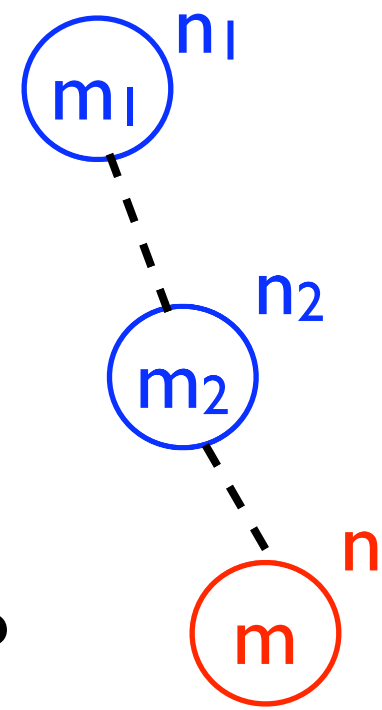
  Add $m_\omega$ as child of $n$;



# Karp & Miller Acceleration

This is how we compute the successors of a node n:

**foreach** *Successor* $m'$ *of* $m$ **do**
  $m_\omega \leftarrow m'$;
  **foreach** *ancestor* $n_i$ *s.t.* $m_i \prec m'$ **do**
    **foreach** *place* $p$ *s.t.* $m_i(p) < m'(p)$ **do**
      $m_\omega(p) \leftarrow \omega$;
  Add $m_\omega$ as child of $n$;



# Karp & Miller Acceleration

This is how we compute the successors of a node n:

**foreach** *Successor* $m'$ *of* $m$ **do**

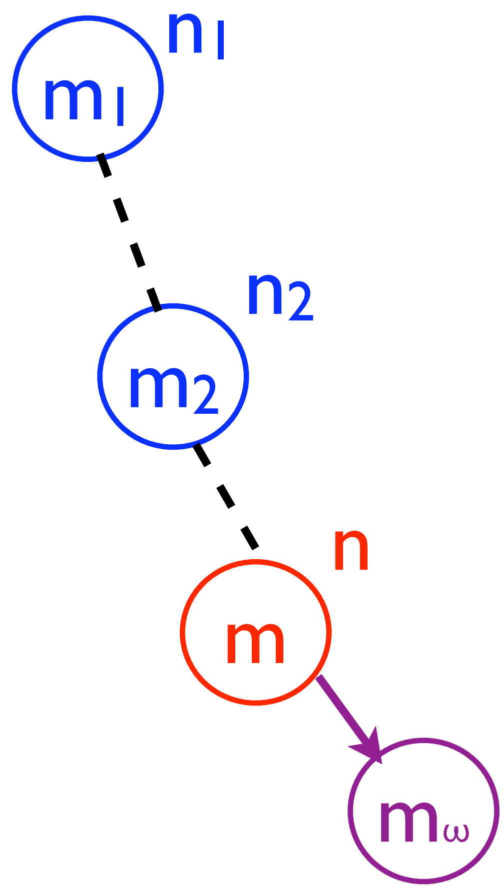
$\quad m_\omega \leftarrow m'$;

$\quad$ **foreach** *ancestor* $n_i$ $s.t.$ $m_i \prec m'$ **do**

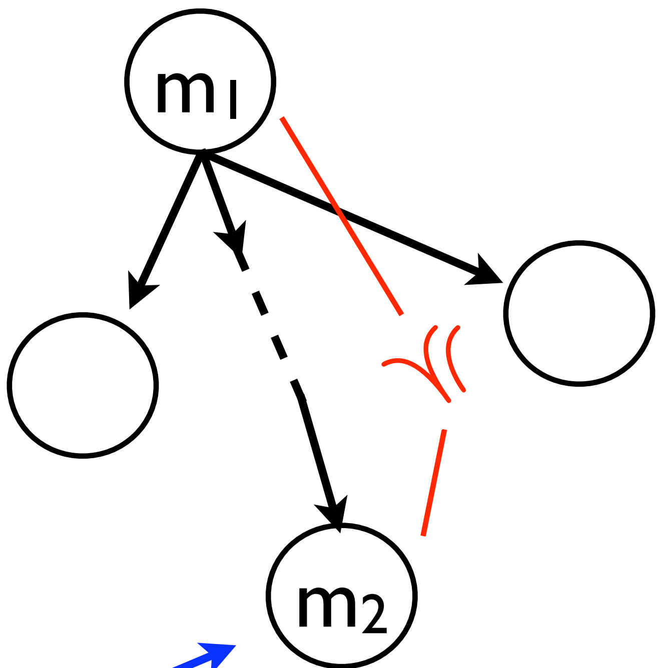
$\quad\quad$ **foreach** *place* $p$ $s.t.$ $m_i(p) < m'(p)$ **do**

$\quad\quad\quad m_\omega(p) \leftarrow \omega$;

$\quad$ Add $m_\omega$ as child of $n$;



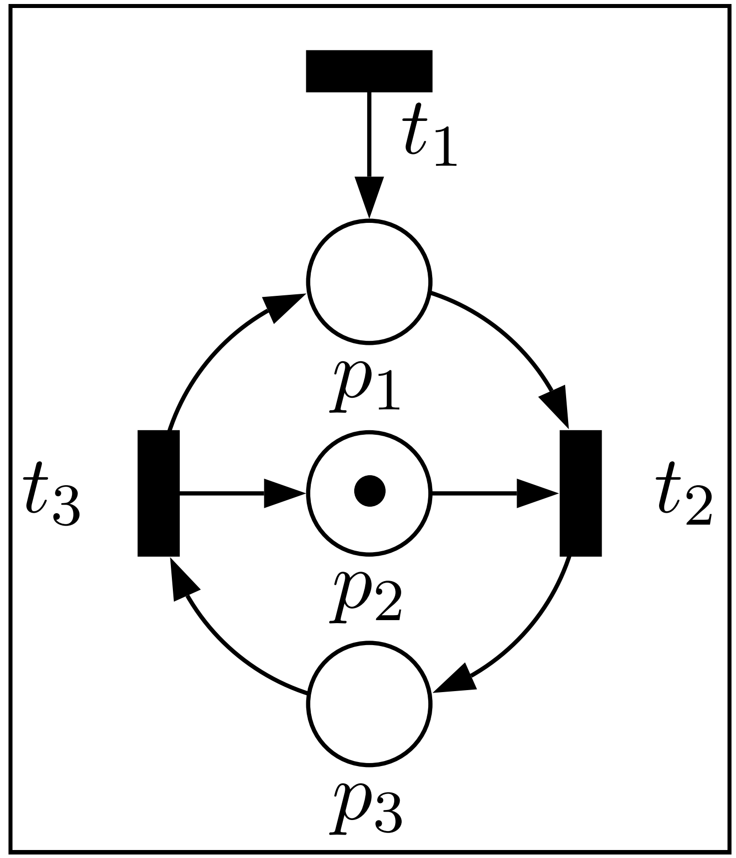
# Karp & Miller
# Stopping a branch




This node doesn't have to be developed

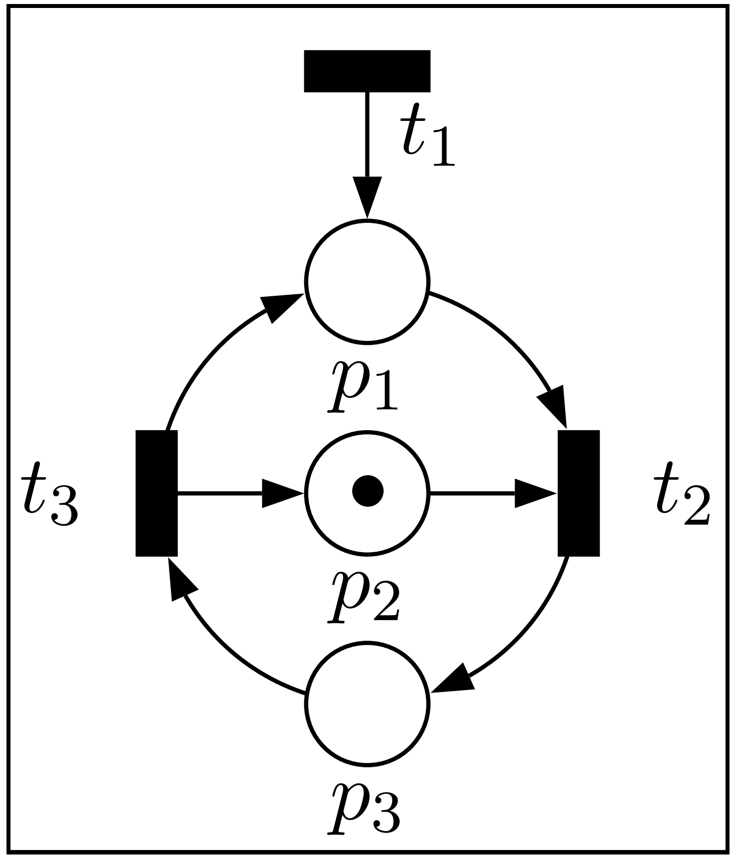
# Example of K&M tree

$\langle 0, 1, 0 \rangle$



# Example of K&M tree

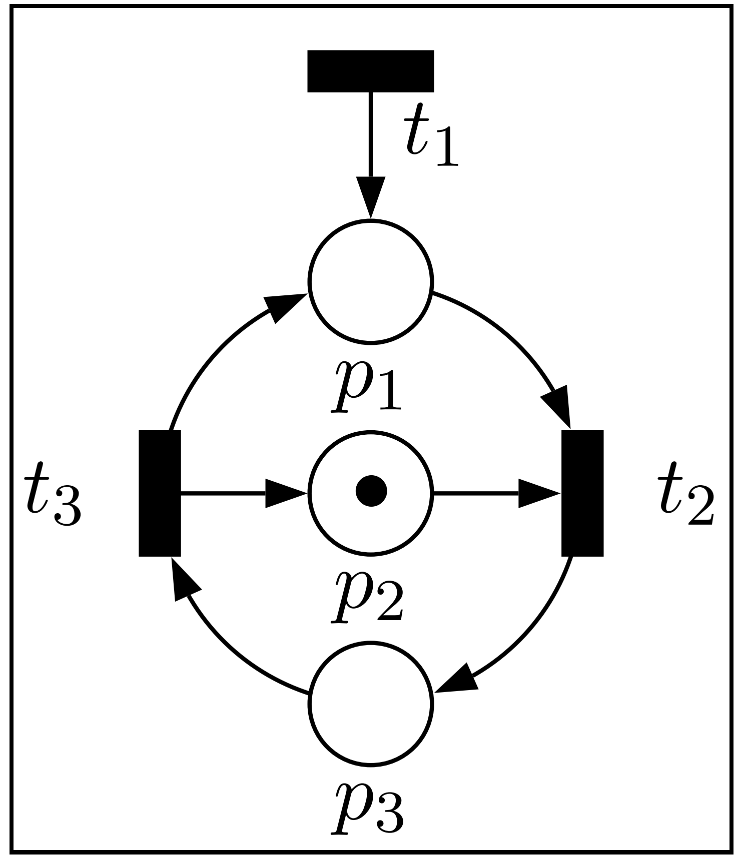
$\langle 0, 1, 0 \rangle$




$$(0,1,0) \xrightarrow{t_1} (1,1,0) \succ (0,1,0)$$

# Example of K&M tree

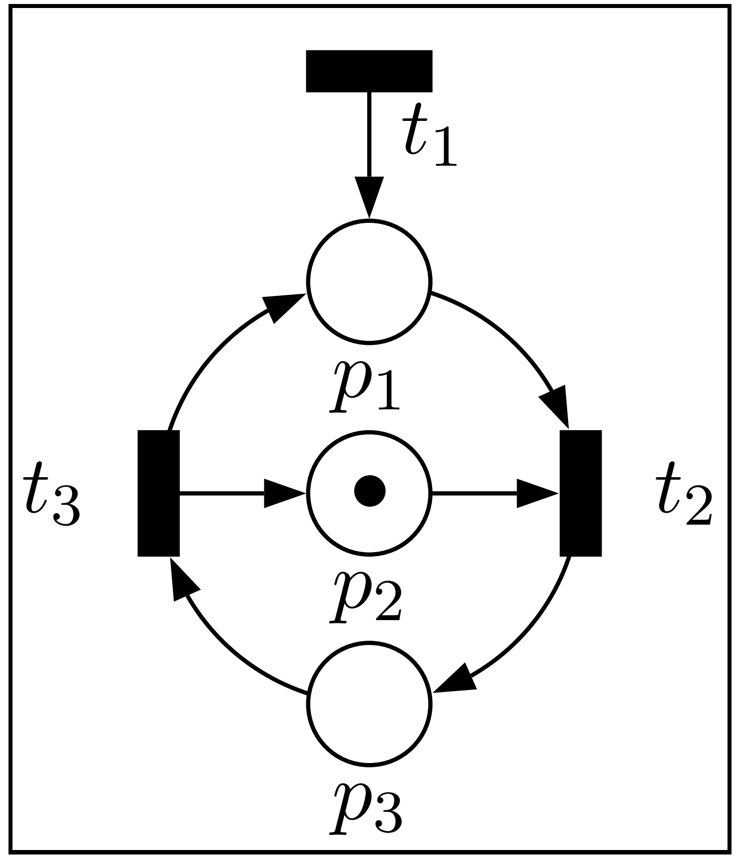
$$\langle 0, 1, 0\rangle$$
$$\Big|\ t_1$$
$$\langle \omega, 1, 0\rangle$$




$$(0,1,0) \xrightarrow{t_1} (1,1,0) \succ (0,1,0)$$

# Example of K&M tree

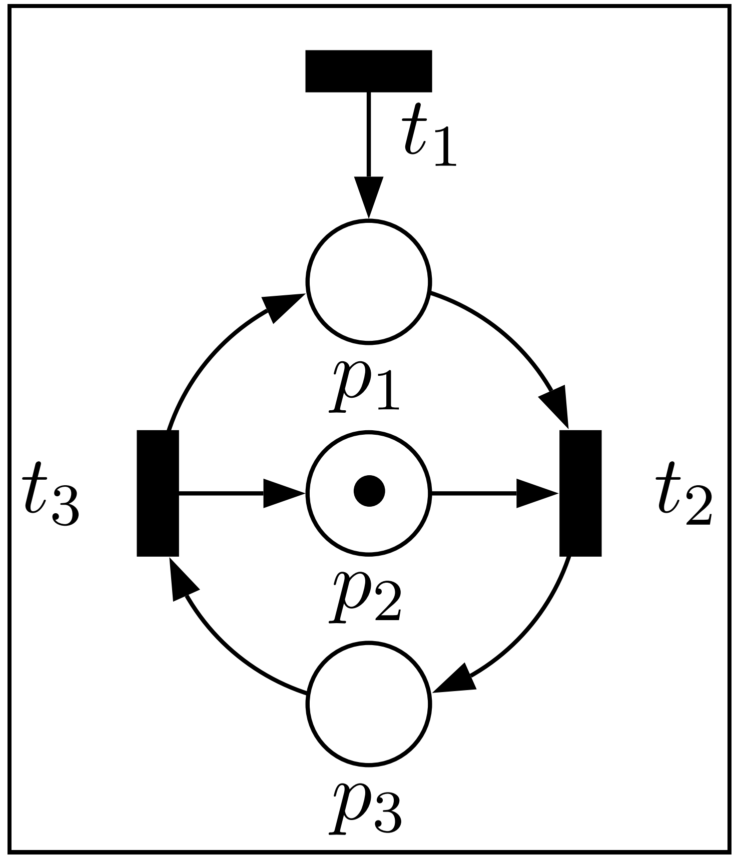
$$\begin{tikzcd}
& \langle 0, 1, 0 \rangle \arrow[d, dash, "t_1"] \\
& \langle \omega, 1, 0 \rangle \arrow[dl, dash, "t_1"'] \\
\langle \omega, 1, 0 \rangle &
\end{tikzcd}$$




$$(0,1,0) \xrightarrow{t_1} (1,1,0) \succ (0,1,0)$$

# Example of K&M tree

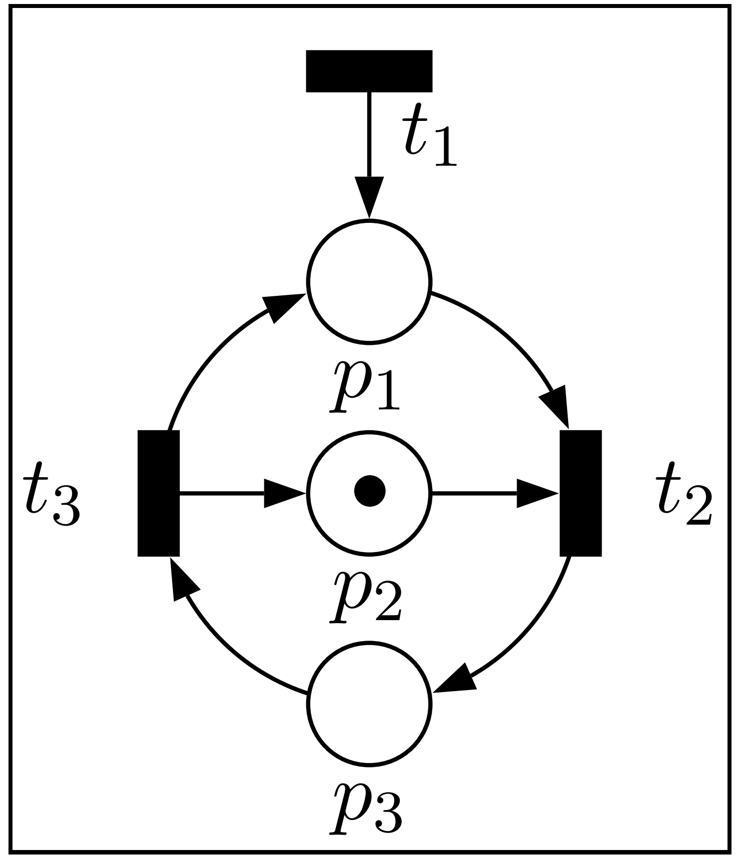
$$
\begin{tikzcd}[column sep=small]
& \langle 0, 1, 0 \rangle \arrow[d, dash, "t_1"] & \\
& \langle \omega, 1, 0 \rangle \arrow[dl, dash, "t_1"'] \arrow[dr, dash, "t_2"] & \\
\langle \omega, 1, 0 \rangle & & \langle \omega, 0, 1 \rangle
\end{tikzcd}
$$




$$(0,1,0) \xrightarrow{t_1} (1,1,0) \succ (0,1,0)$$

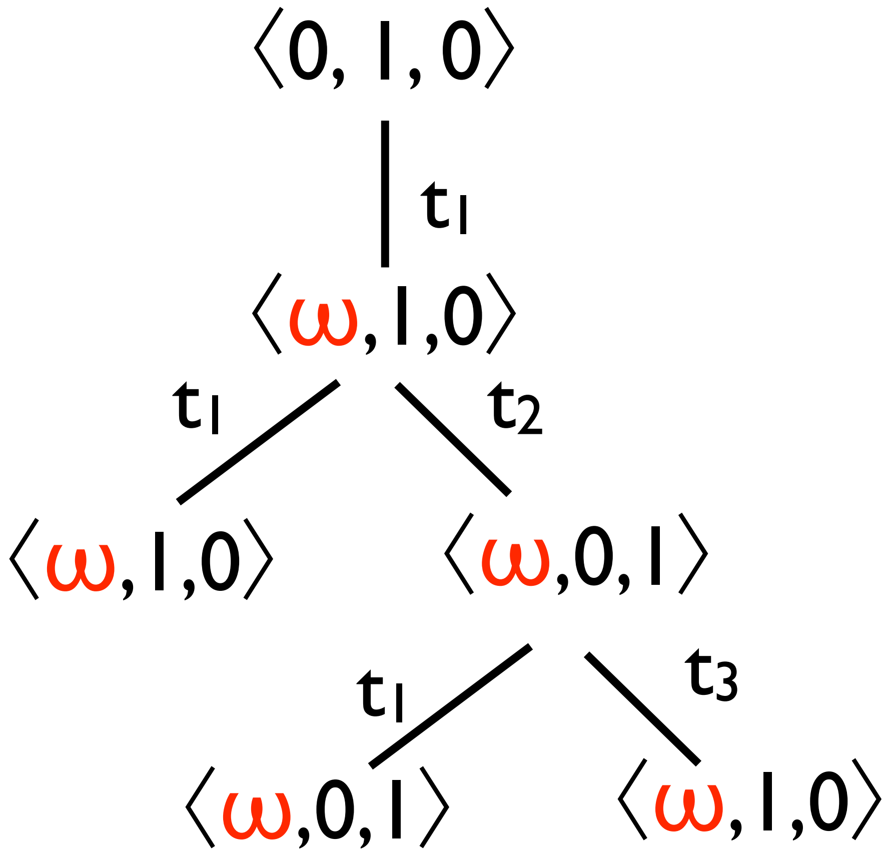
# Example of K&M tree

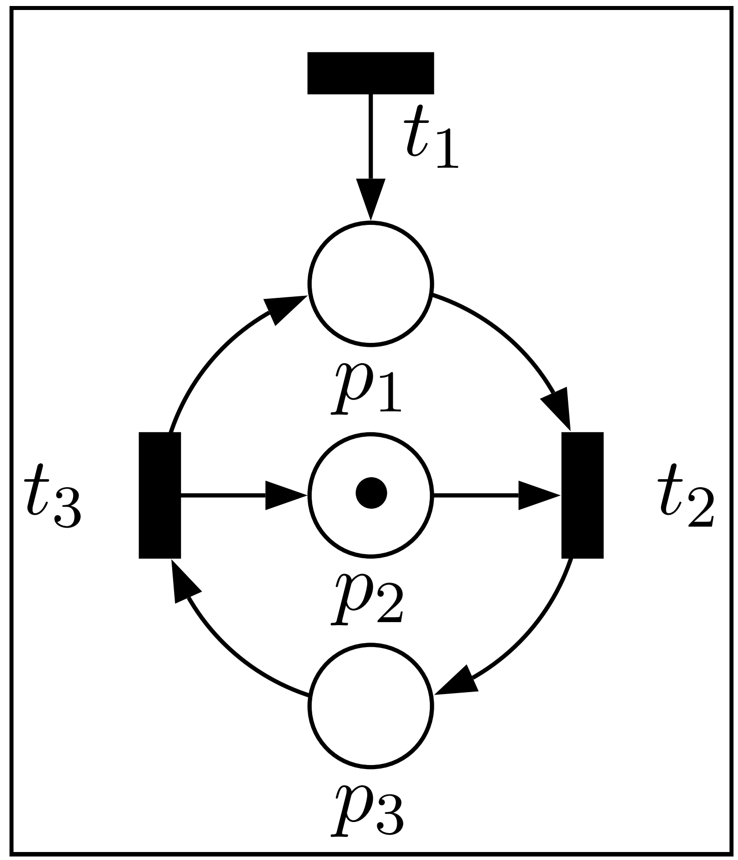
$$
\begin{tikzcd}[column sep=small]
 & \langle 0, 1, 0 \rangle \arrow[d, dash, "t_1"] & \\
 & \langle \omega, 1, 0 \rangle \arrow[dl, dash, "t_1"'] \arrow[dr, dash, "t_2"] & \\
\langle \omega, 1, 0 \rangle & & \langle \omega, 0, 1 \rangle \arrow[dl, dash, "t_1"'] \\
 & \langle \omega, 0, 1 \rangle &
\end{tikzcd}
$$




$$(0,1,0) \xrightarrow{t_1} (1,1,0) \succ (0,1,0)$$

# Example of K&M tree




$$(0,1,0) \xrightarrow{t_1} (1,1,0) \succ (0,1,0)$$

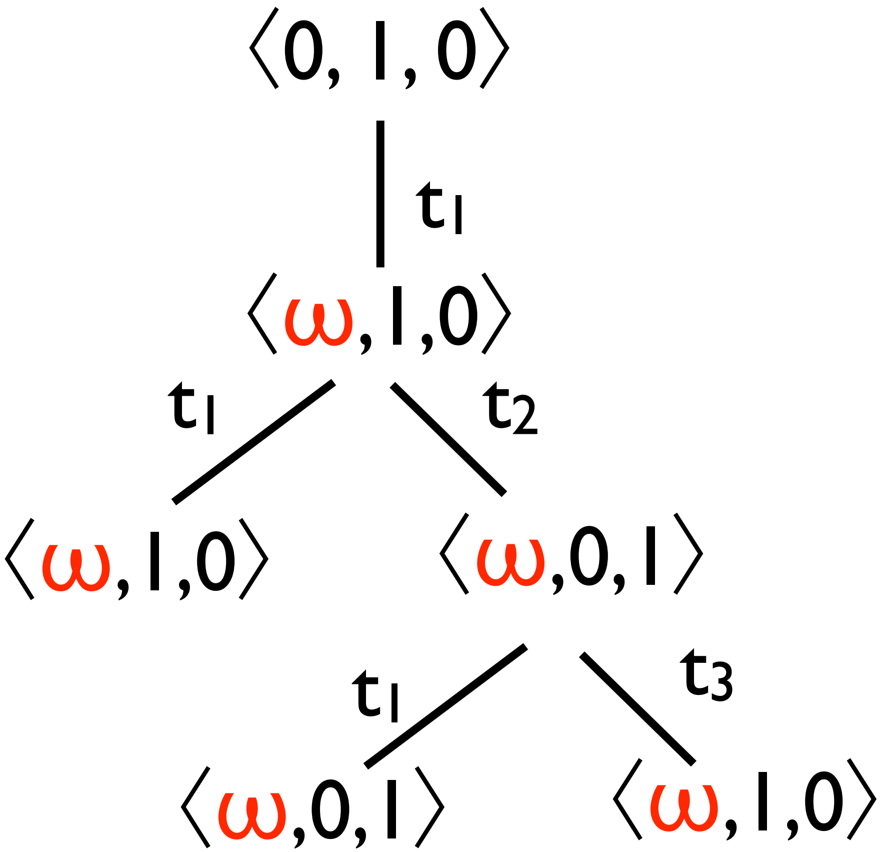
# Properties

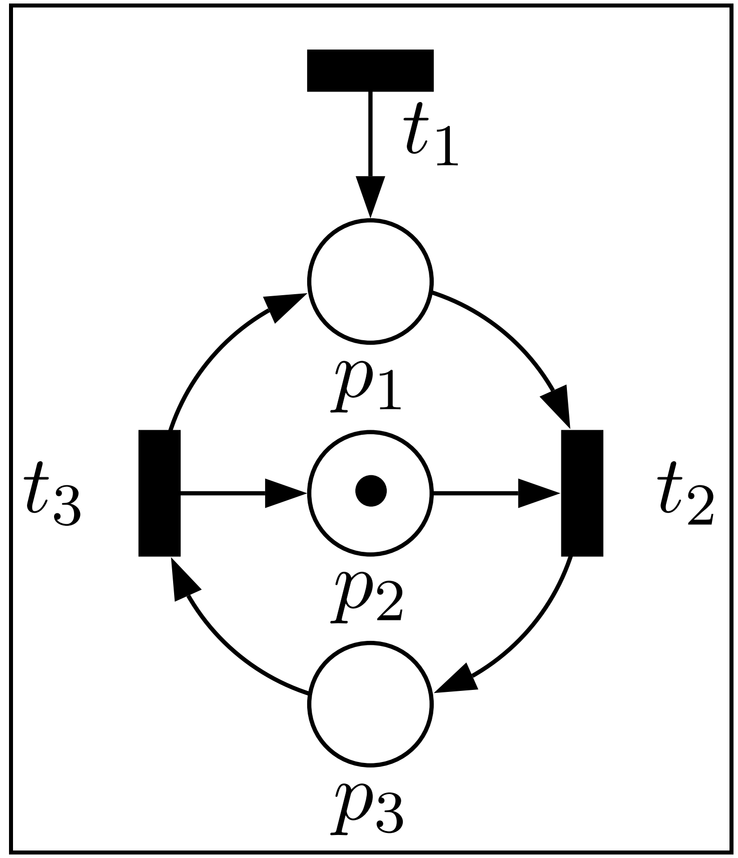
- Theorem: the K&M tree is always finite.
  - Idea of the proof:
    - if the net is not bounded, it is because of some infinite increasing sequence of markings.
    - such sequences are detected in a finite amount of time by adding $\omega$ in the unbounded places.

# Properties

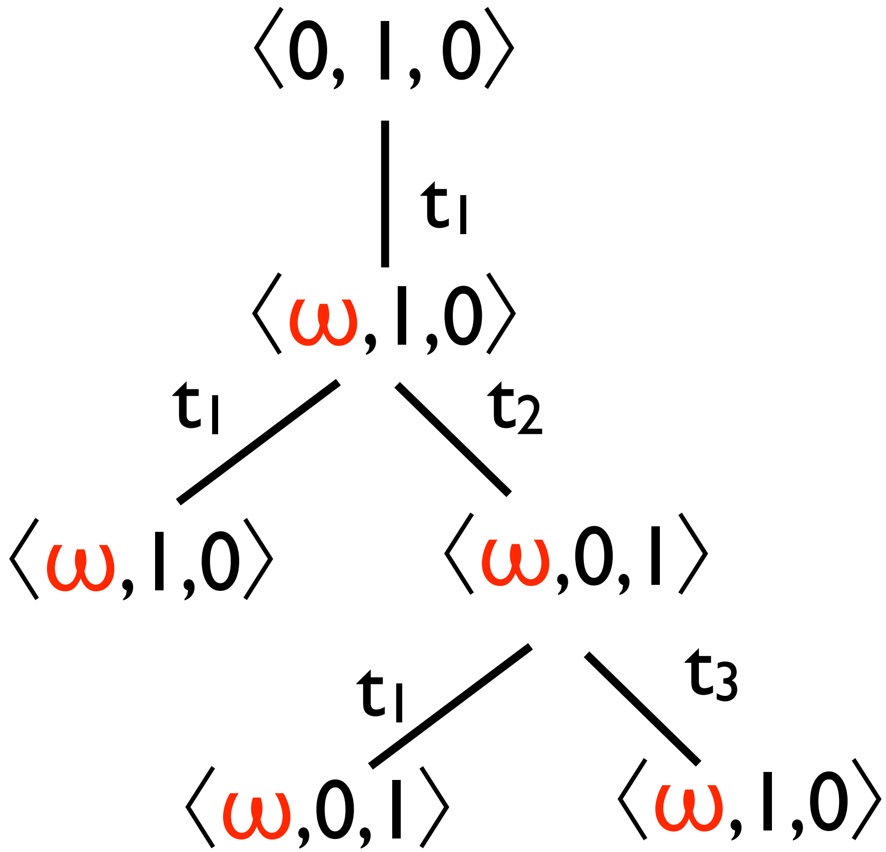
- Theorem: a net is bounded iff there is no node containing an $\omega$ in its K&M tree.
- Theorem: place p is unbounded iff there exists a node labeled by m in the K&M tree s.t. $m(p) = \omega$.
- Theorem: transition t is semi-live iff there exists a node labeled by m in the K&M tree s.t. t can fire in m.

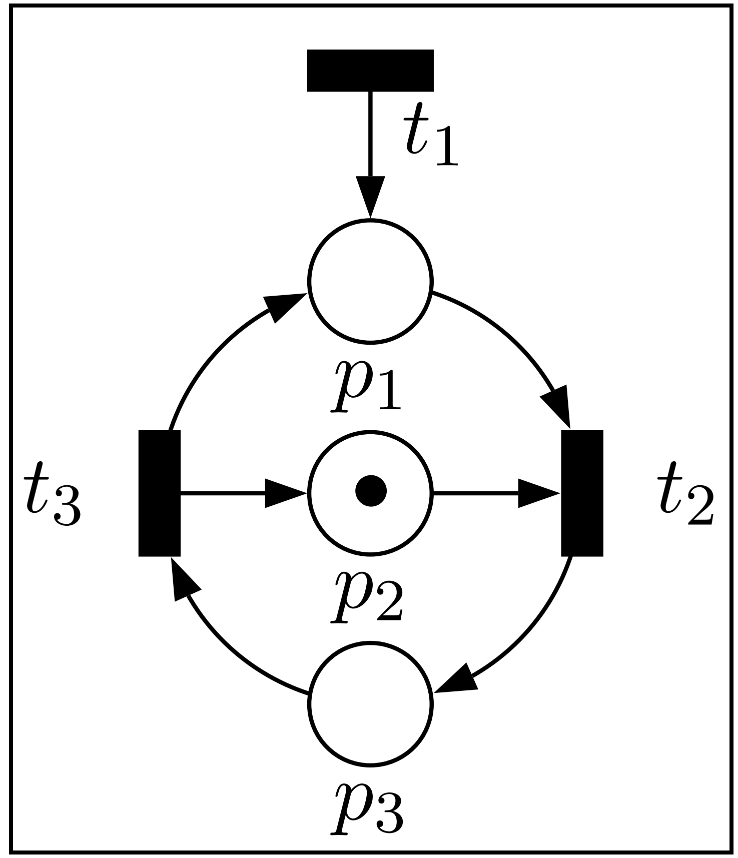
# Example



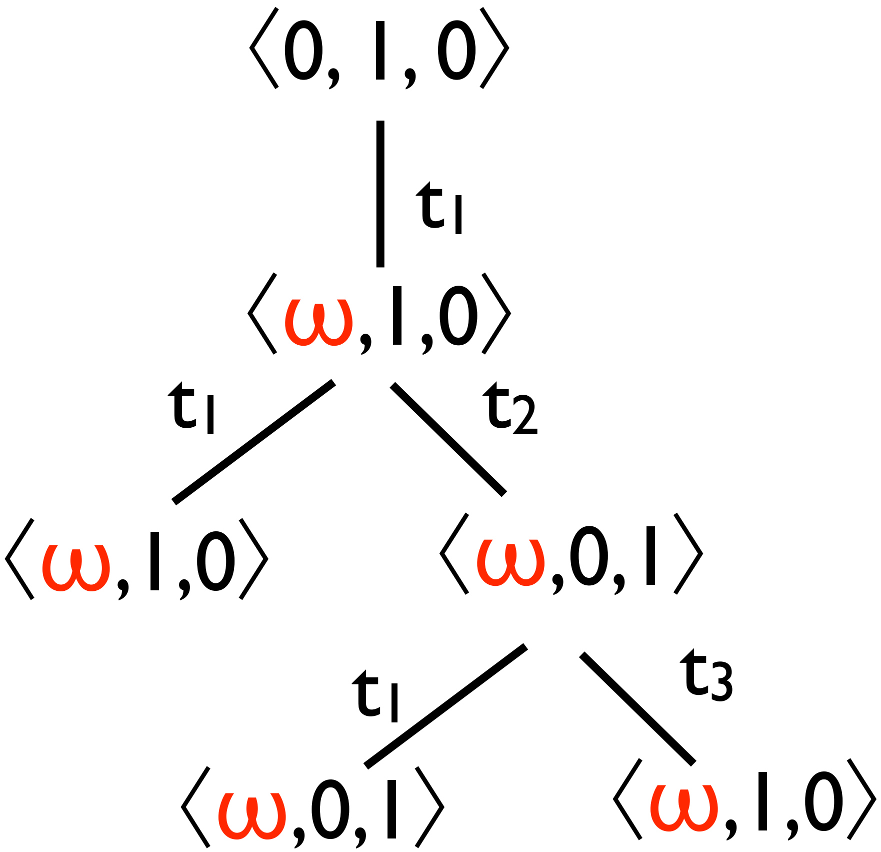
# Example

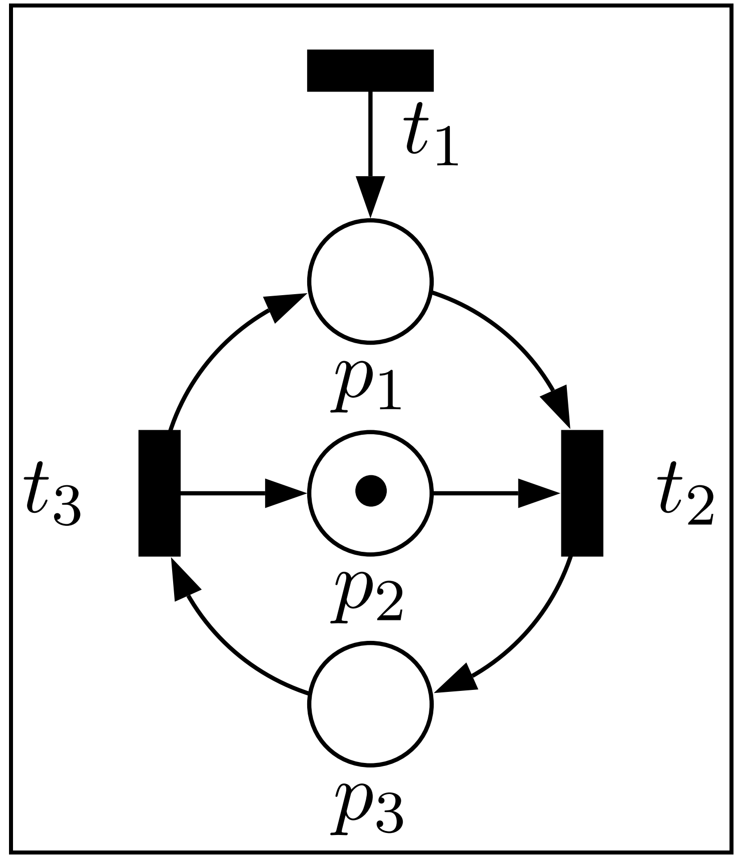



$t_2$ is semi-live

# Example

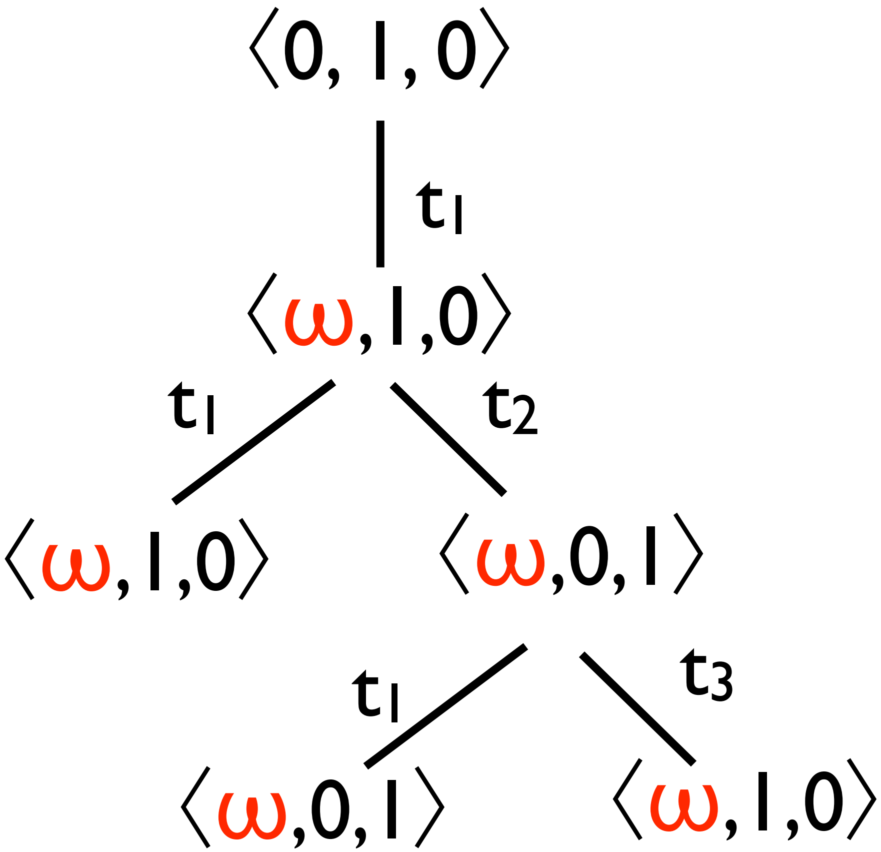


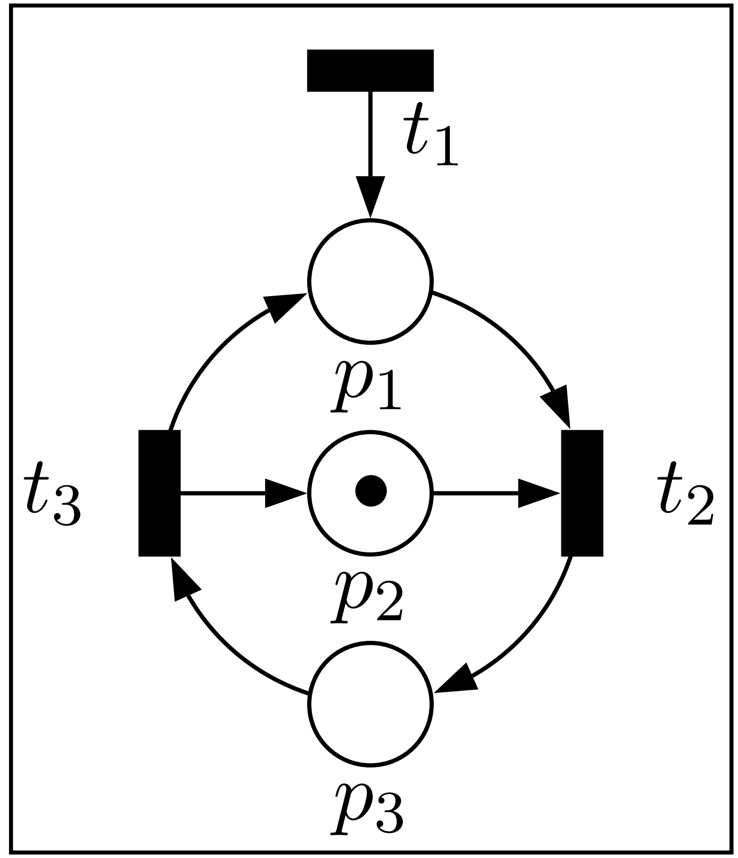




$t_2$ is semi-live

$p_2$ and $p_3$ are bounded

# Example



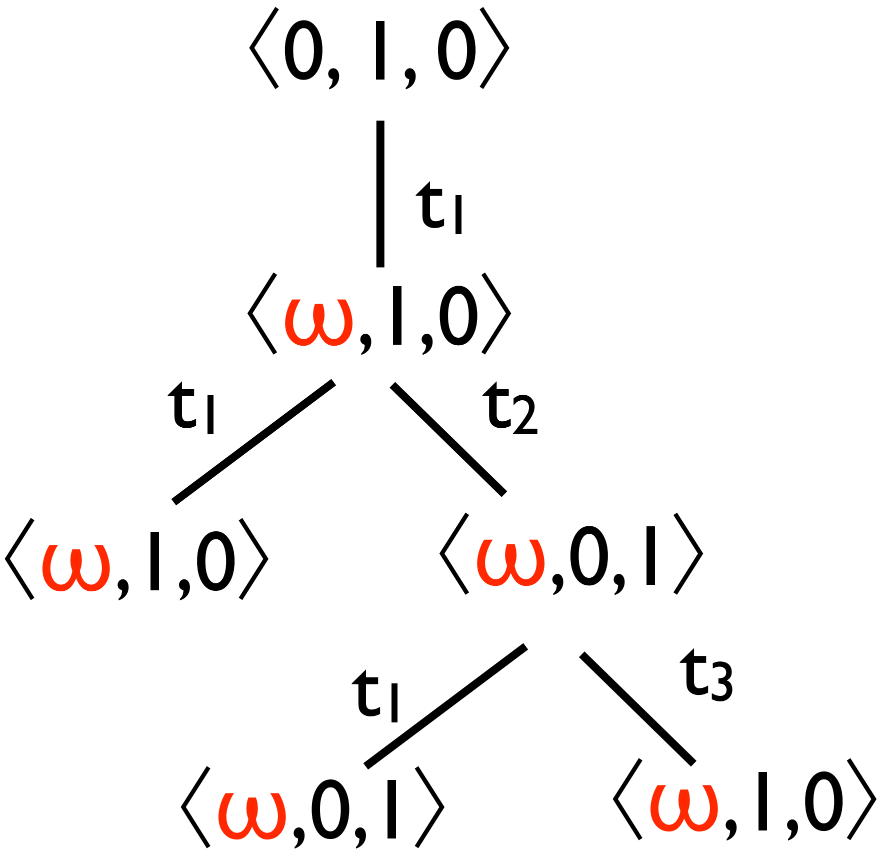



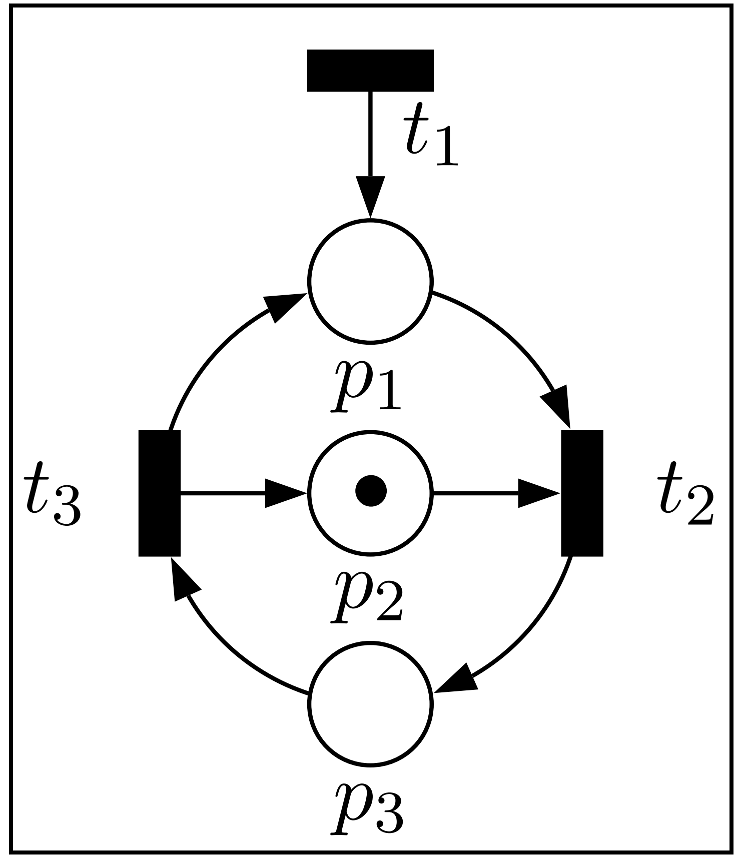

$t_2$ is semi-live

$p_1$ is unbounded

$p_2$ and $p_3$ are bounded

# Example




$t_2$ is semi-live

$p_1$ is unbounded

$p_2$ and $p_3$ are bounded

The net is unbounded

# Coverability set

- Question: what is the relationship between:
  - the set of reachable markings and
  - the set of labels of the nodes of the K&M tree ?

# Coverability set

might be infinite

- Question: what is the relationship between:
  - the set of reachable markings and
  - the set of labels of the nodes of the K&M tree ?

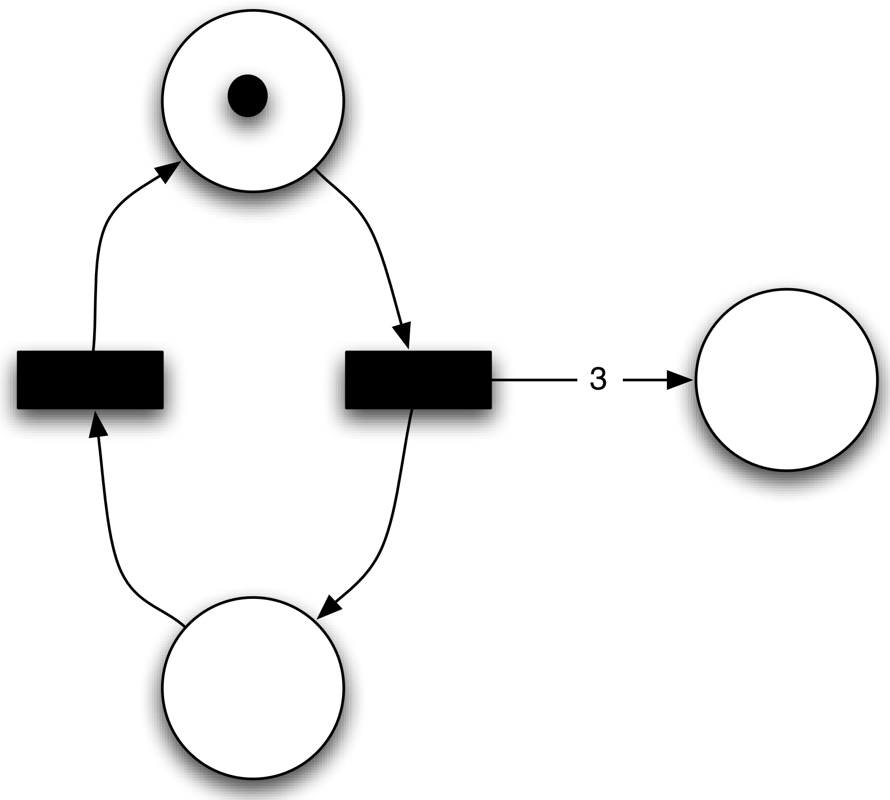
# Coverability set

might be infinite

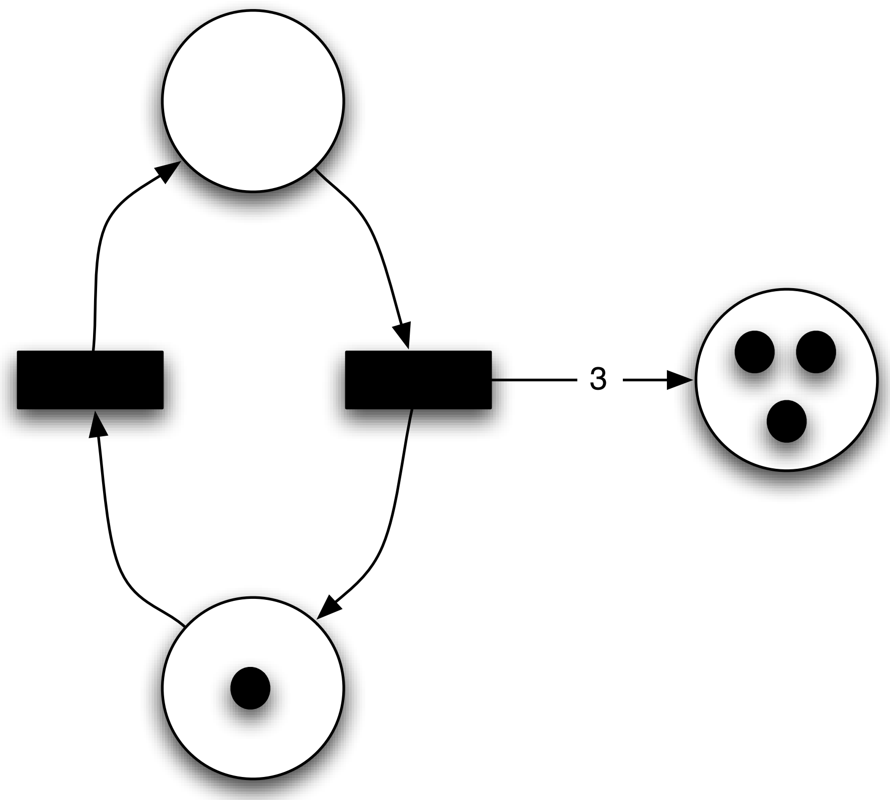
- Question: what is the relationship between:
  - the set of reachable markings and
  - the set of labels of the nodes of the K&M tree ?

always finite

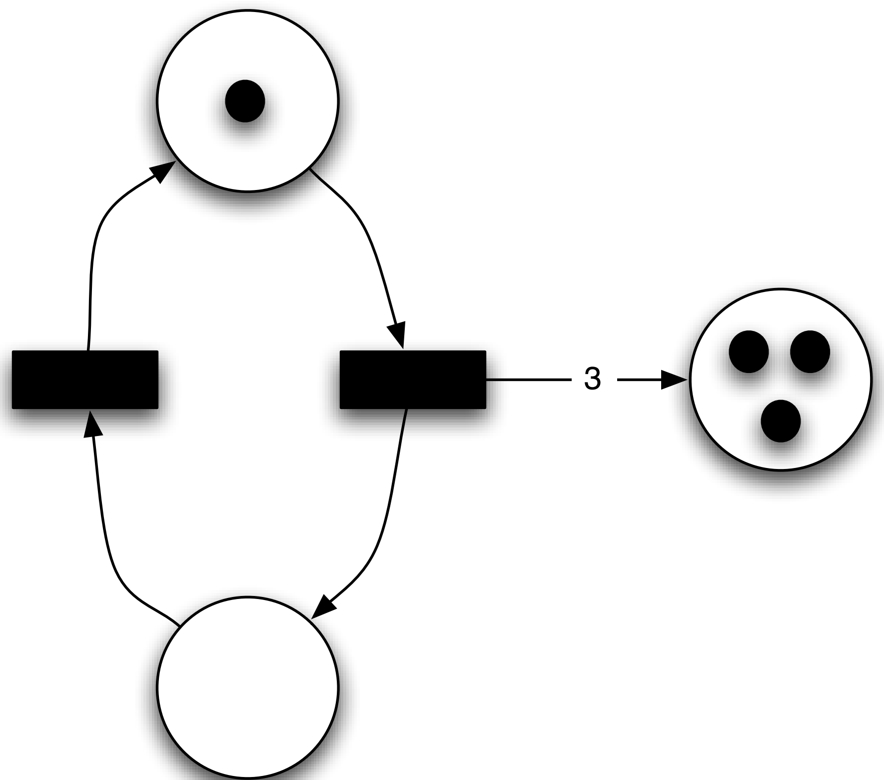
# Example



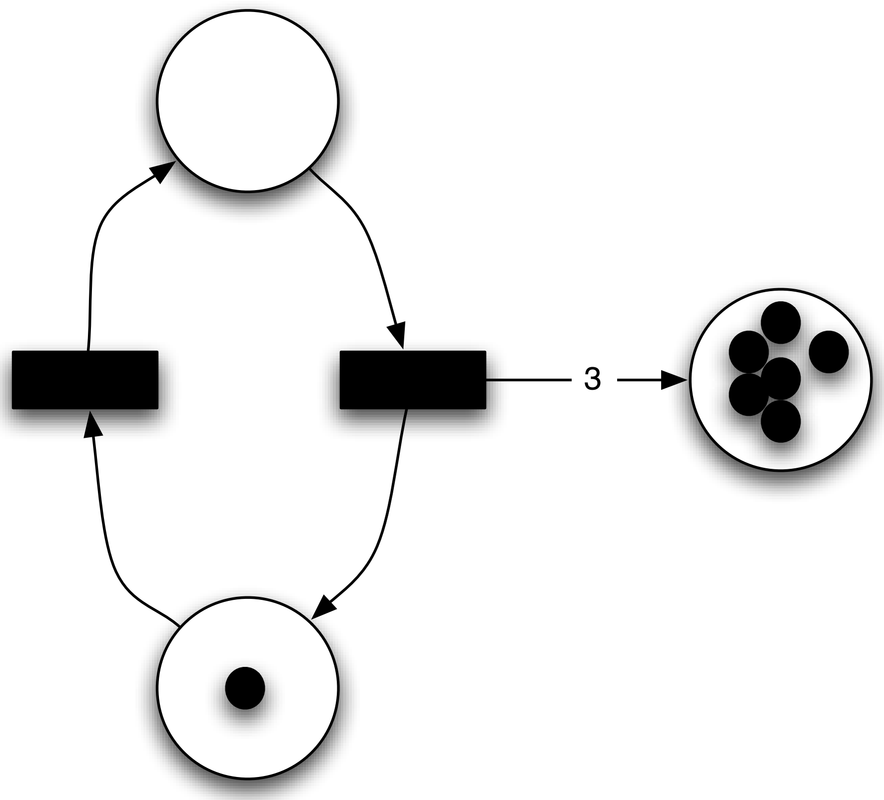
# Example



# Example



# Example



# Example

- Set of reachable markings:

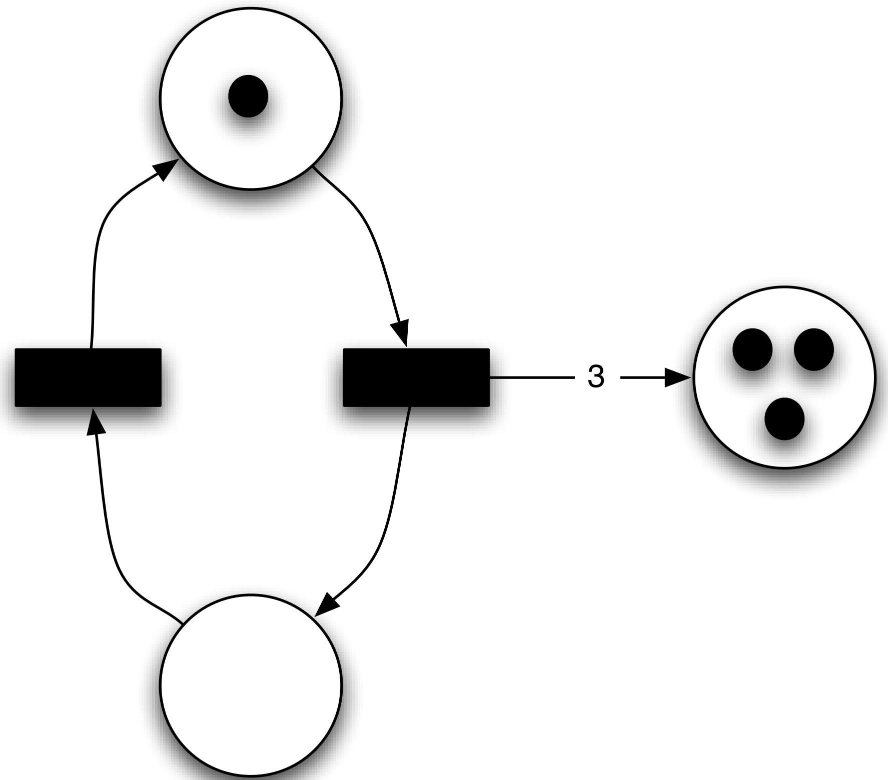
$\{ \langle 1, 0, 3.i \rangle \ , \ \langle 0, 1, 3.i \rangle \ \mid i \geqslant 0 \}$

- Set of nodes of the K&M tree:

$\{ \langle 1, 0, 0 \rangle \quad \langle 1, 0, \omega \rangle \ , \ \langle 0, 1, \omega \rangle \ \}$

- This set "represents":

$\{ \langle 1, 0, i \rangle \ , \ \langle 0, 1, i \rangle \ \mid i \geqslant 0 \}$



# Example

- Set of reachable markings:

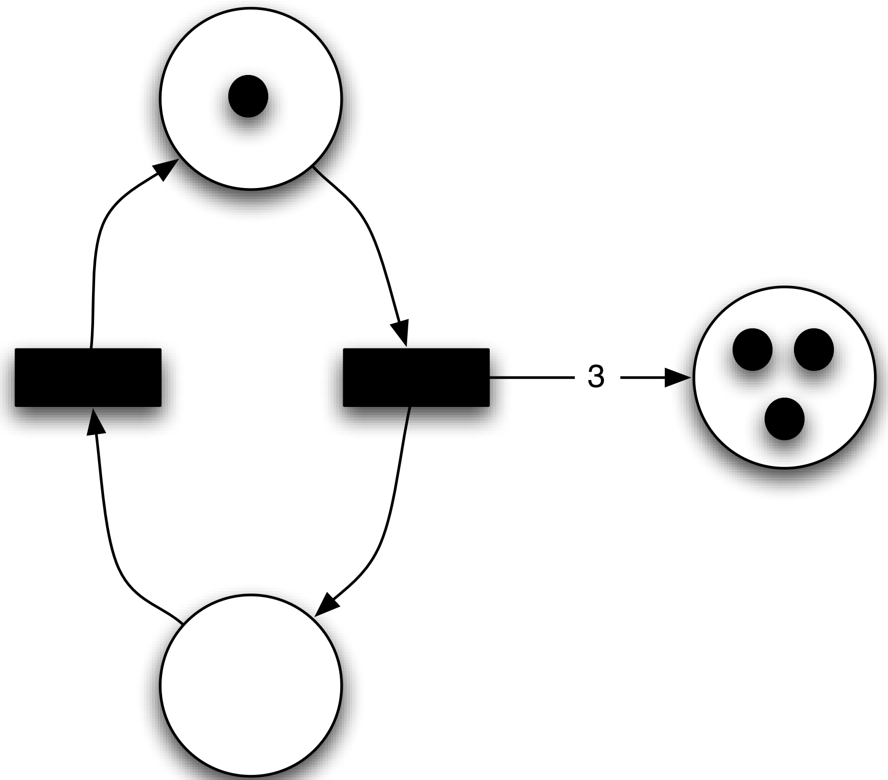
$\{ \langle 1, 0, 3.i \rangle \ , \ \langle 0, 1, 3.i \rangle \ \mid i \geqslant 0 \}$

- Set of nodes of the K&M tree:

$\{ \langle 1, 0, 0 \rangle \quad \langle 1, 0, \omega \rangle \ , \ \langle 0, 1, \omega \rangle \ \}$

- This set "represents":

$\{ \langle 1, 0, i \rangle \ , \ \langle 0, 1, i \rangle \ \mid i \geqslant 0 \}$




Clearly: $\neq$

# Example

Reach

$\{ \langle 1, 0, 3.i \rangle \ , \ \langle 0, 1, 3.i \rangle \ | \ i \geqslant 0 \}$

vs

K&M

$\{ \langle 1, 0, i \rangle \ , \ \langle 0, 1, i \rangle \ | \ i \geqslant 0 \}$

- Clearly, the K&M set contains more markings than the set of reachable markings:

$$\square \subseteq \square$$

- However, for every marking m in the K&M set, there exists a reachable marking m' s.t.:

$$m' \succcurlyeq m$$

# Example

Reach

$\{ \langle 1, 0, 3.i \rangle \;,\; \langle 0, 1, 3.i \rangle \mid i \geqslant 0 \}$

vs

K&M

$\{ \langle 1, 0, i \rangle \;,\; \langle 0, 1, i \rangle \mid i \geqslant 0 \}$

- Clearly, the K&M set contains more markings than the set of reachable markings:

$$\subseteq$$

- However, for every marking m in the K&M set, there exists a reachable marking m’ s.t.:

$$m' \succcurlyeq m$$

$$= \quad + \{m \mid \text{there is } m' \text{ in} \quad \text{with } m' \succcurlyeq m\}$$

# Downward-closure

- Let us assume that any natural number i is s.t.

$$i < \omega$$

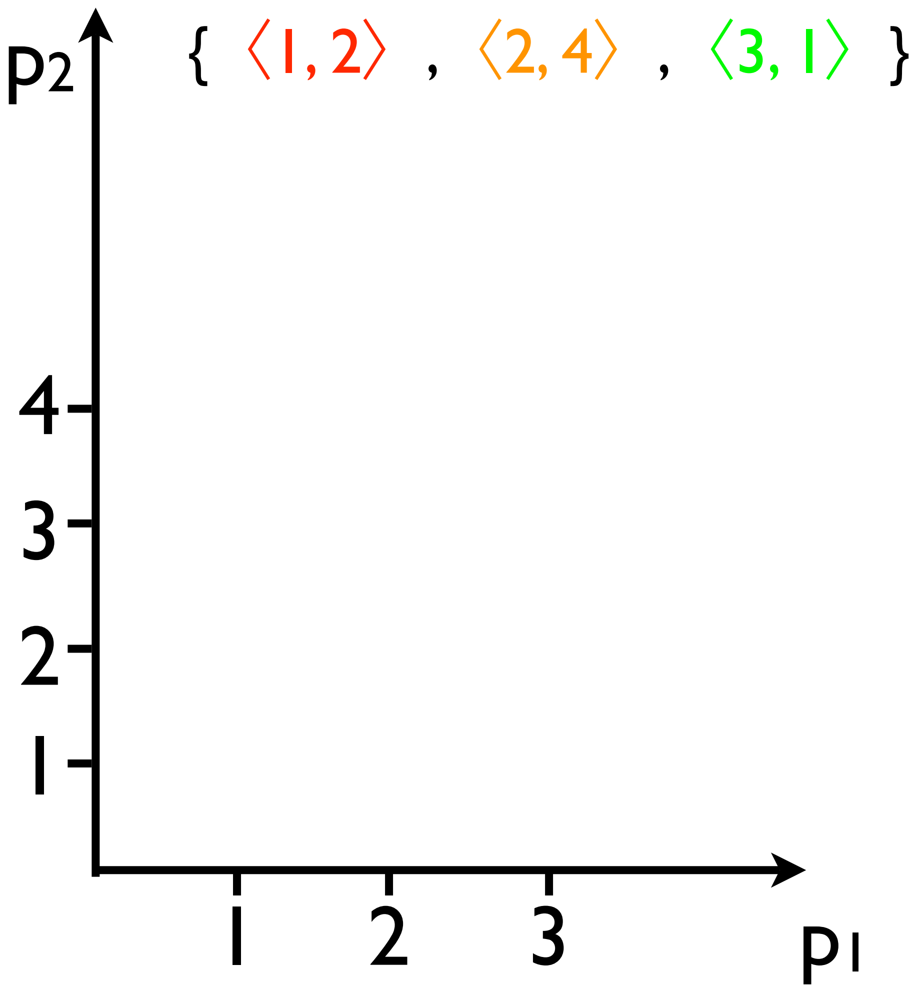
- Let $m$ be a marking (possibly with $\omega$), then its downward-closure is the set:

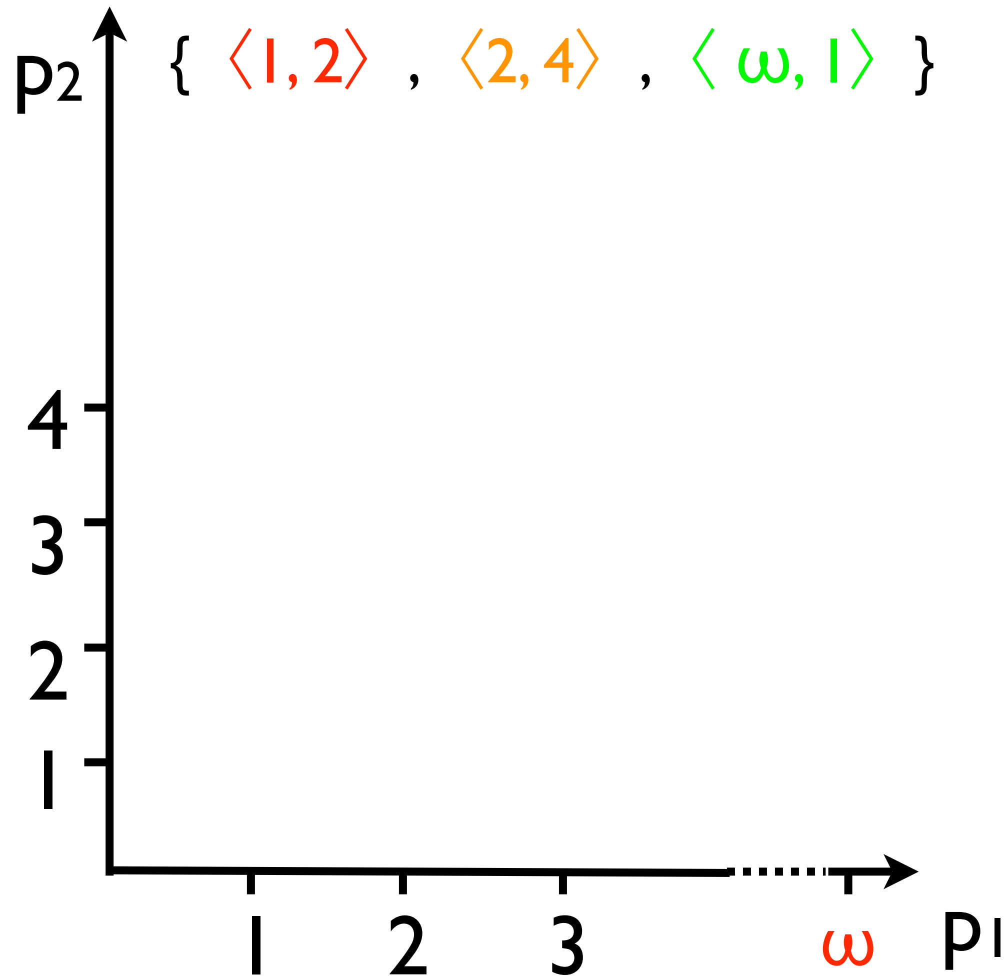
$$\downarrow m = \{m' \mid m' \preccurlyeq m\}$$

- Let $S=\{m_1, m_2, \ldots m_k\}$ be a set of markings, then:

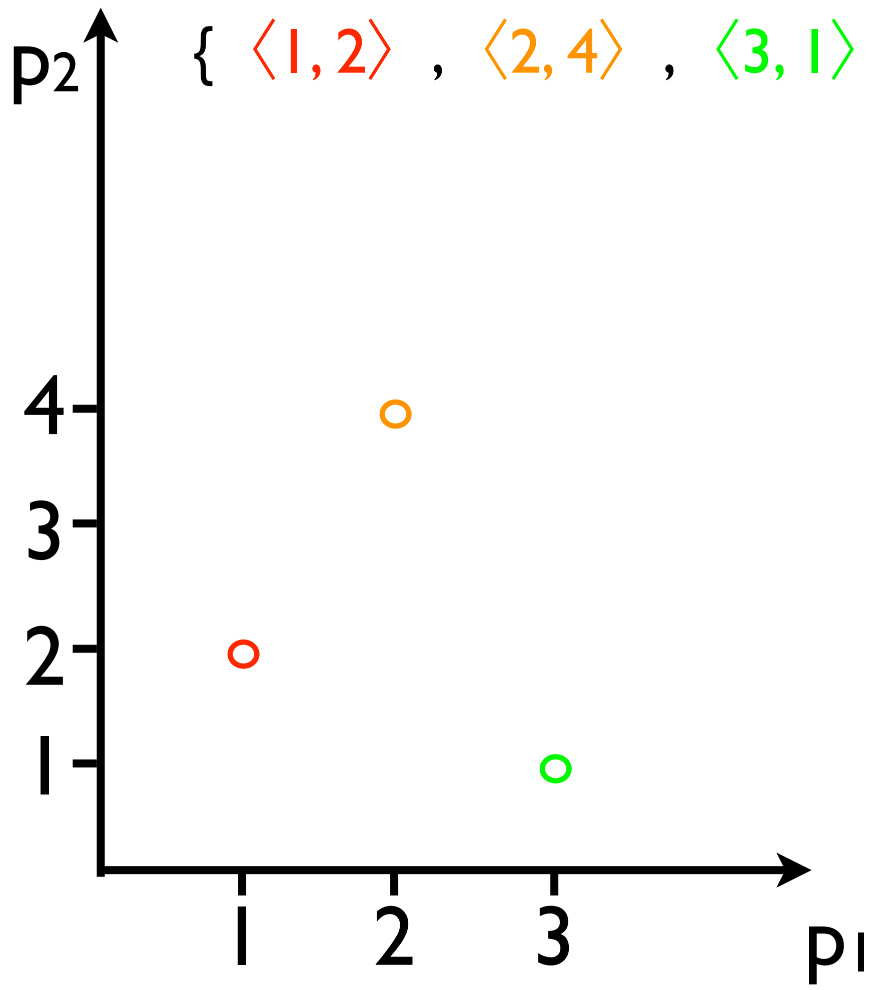
$$\downarrow S = \downarrow m_1 \cup \downarrow m_2 \cup \ldots \cup \downarrow m_k$$

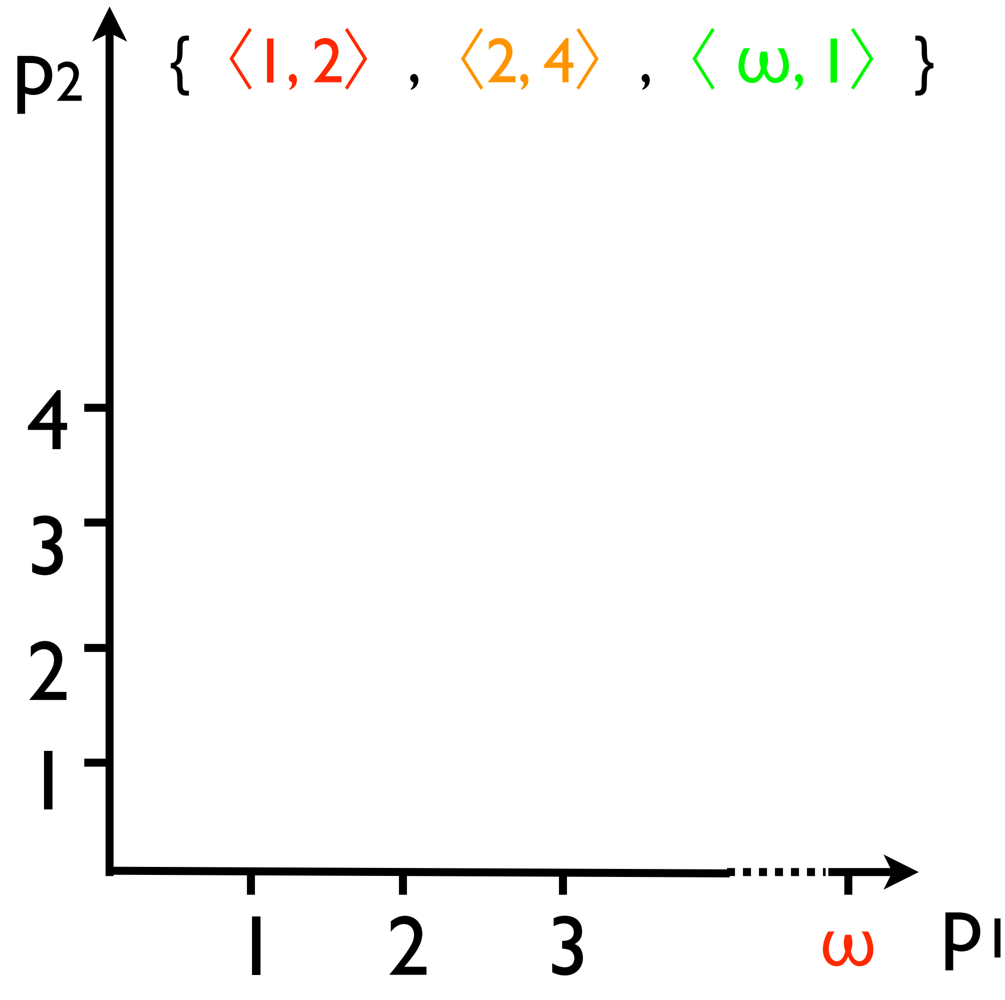
# Examples in 2 dim.



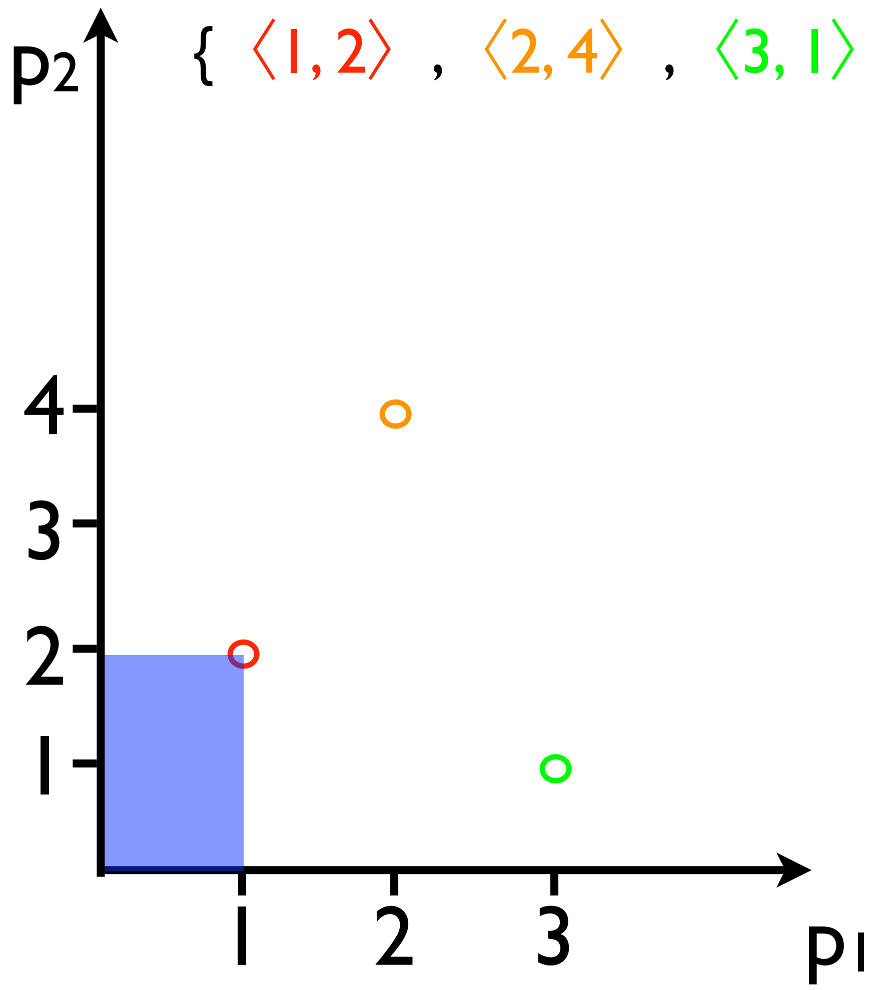



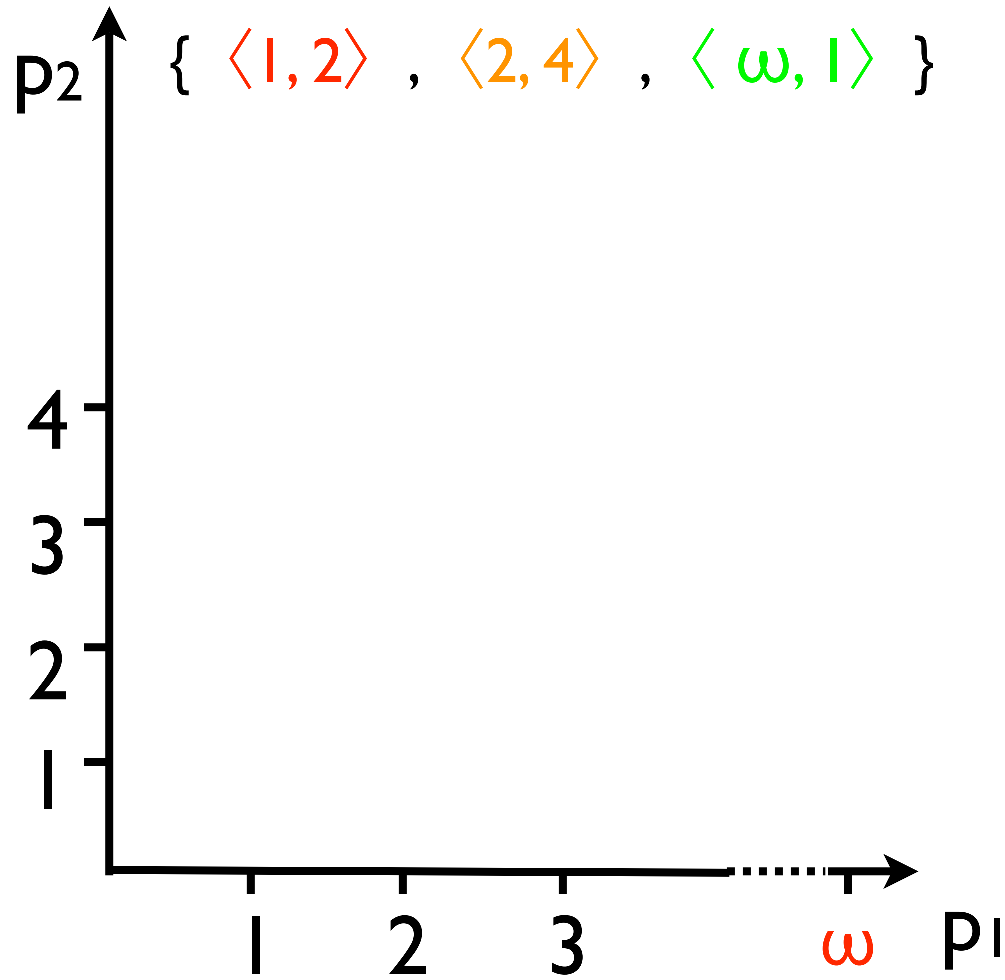
# Examples in 2 dim.



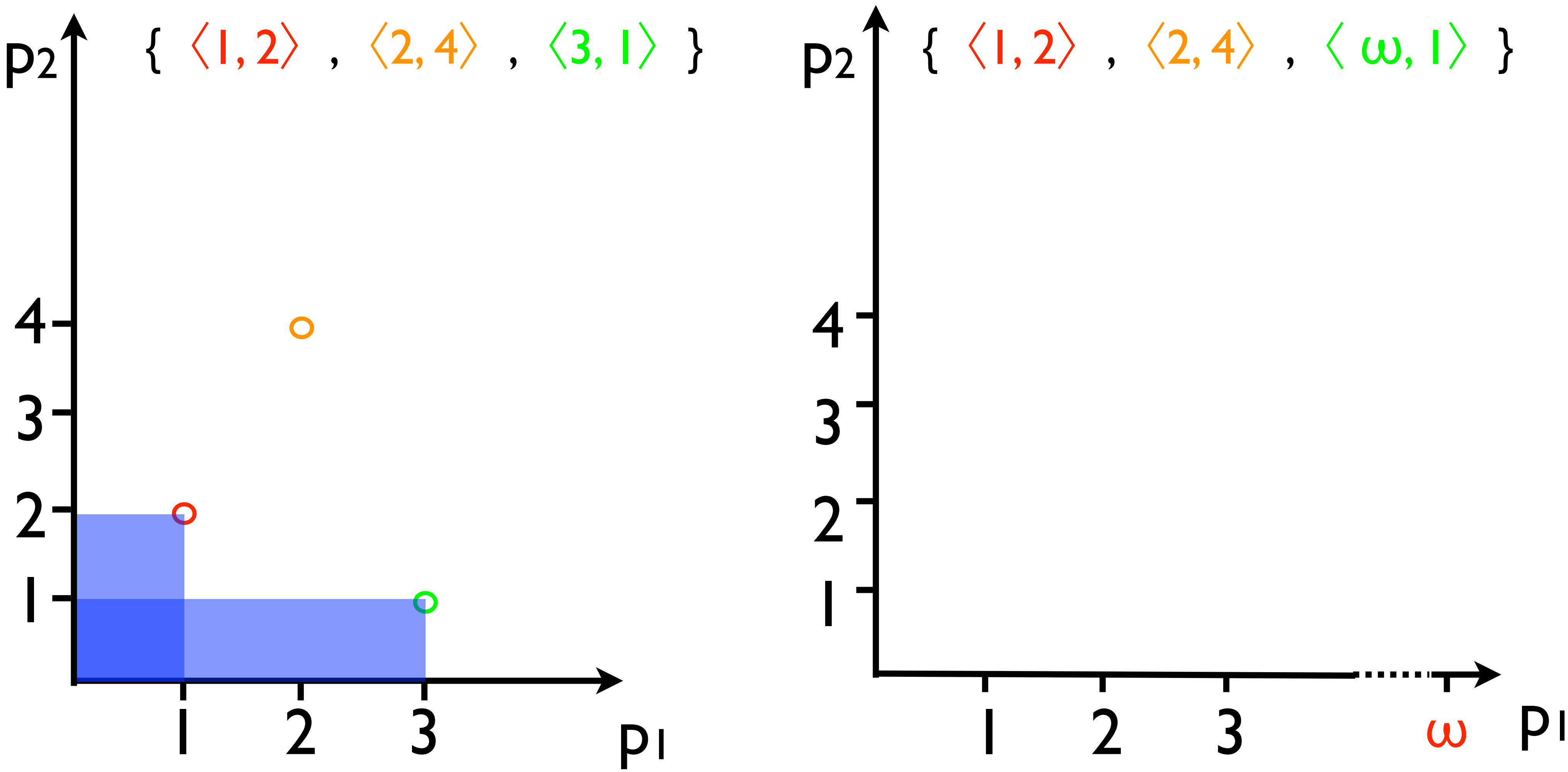



# Examples in 2 dim.

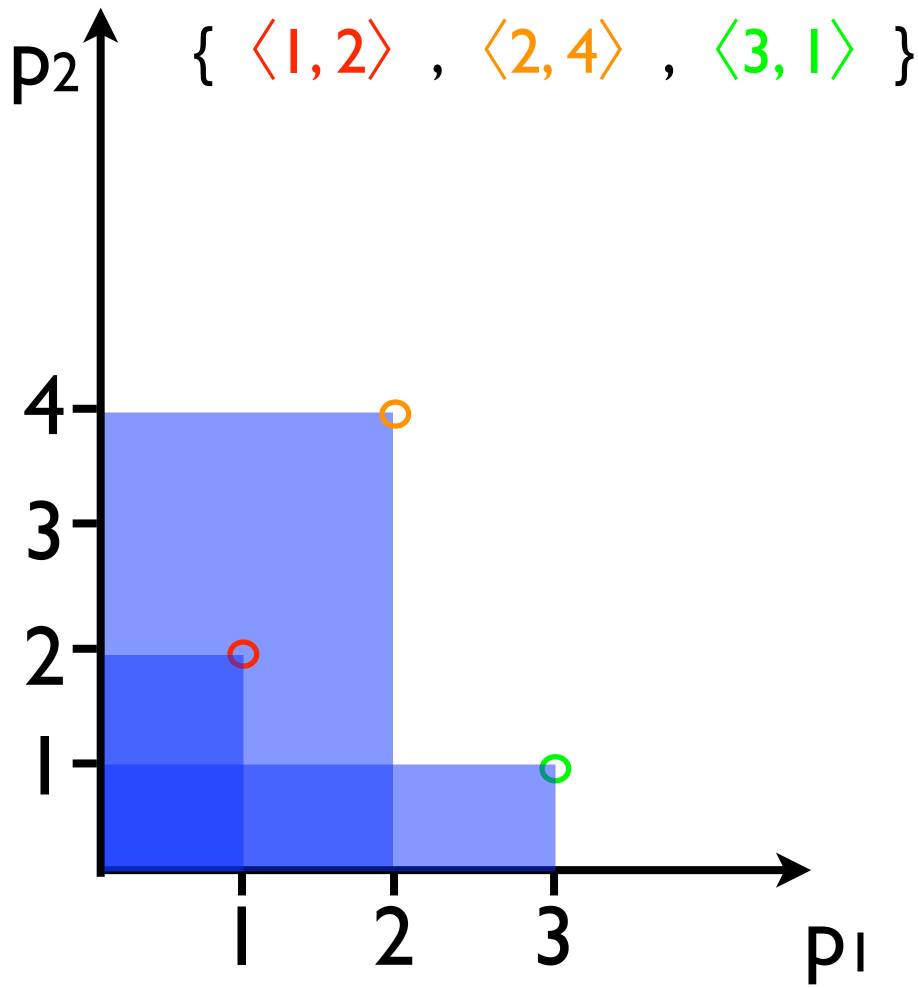


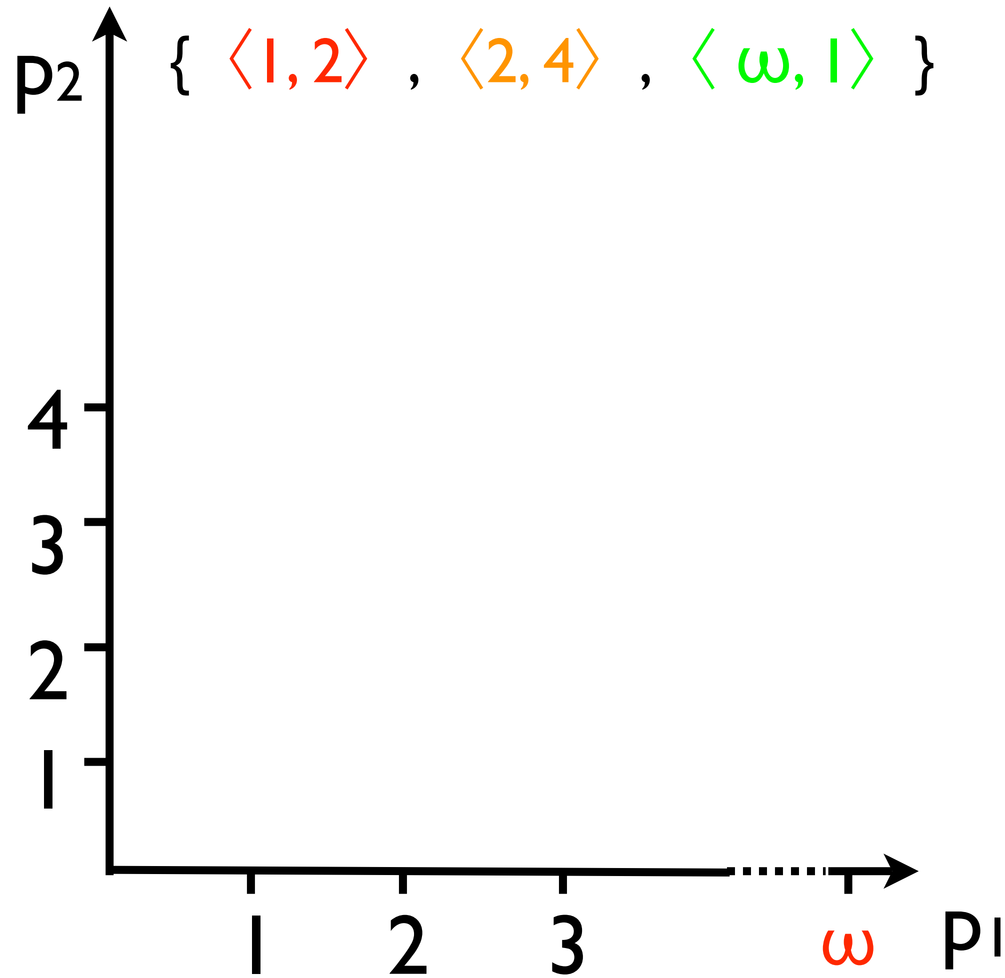



# Examples in 2 dim.

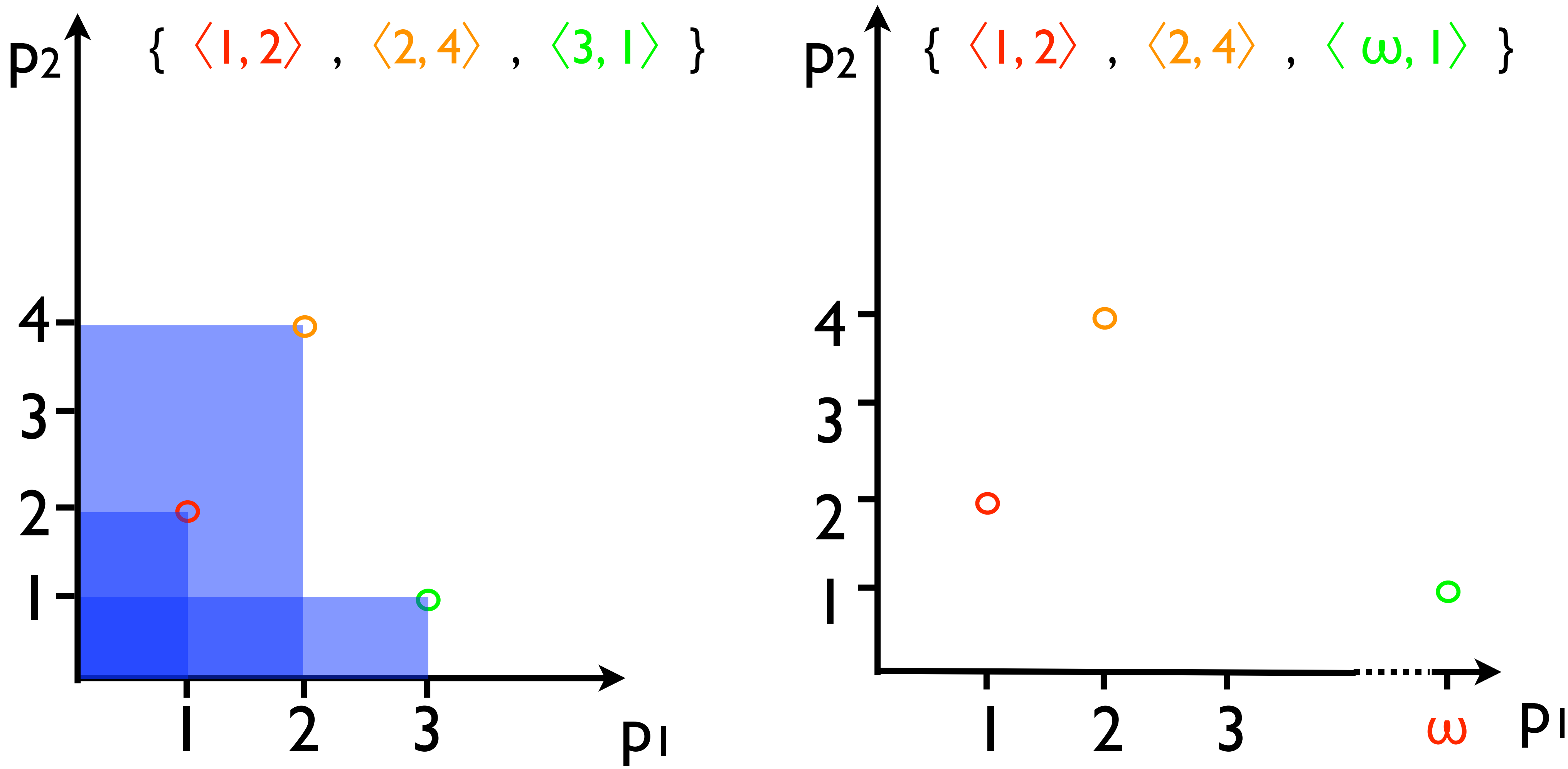


# Examples in 2 dim.

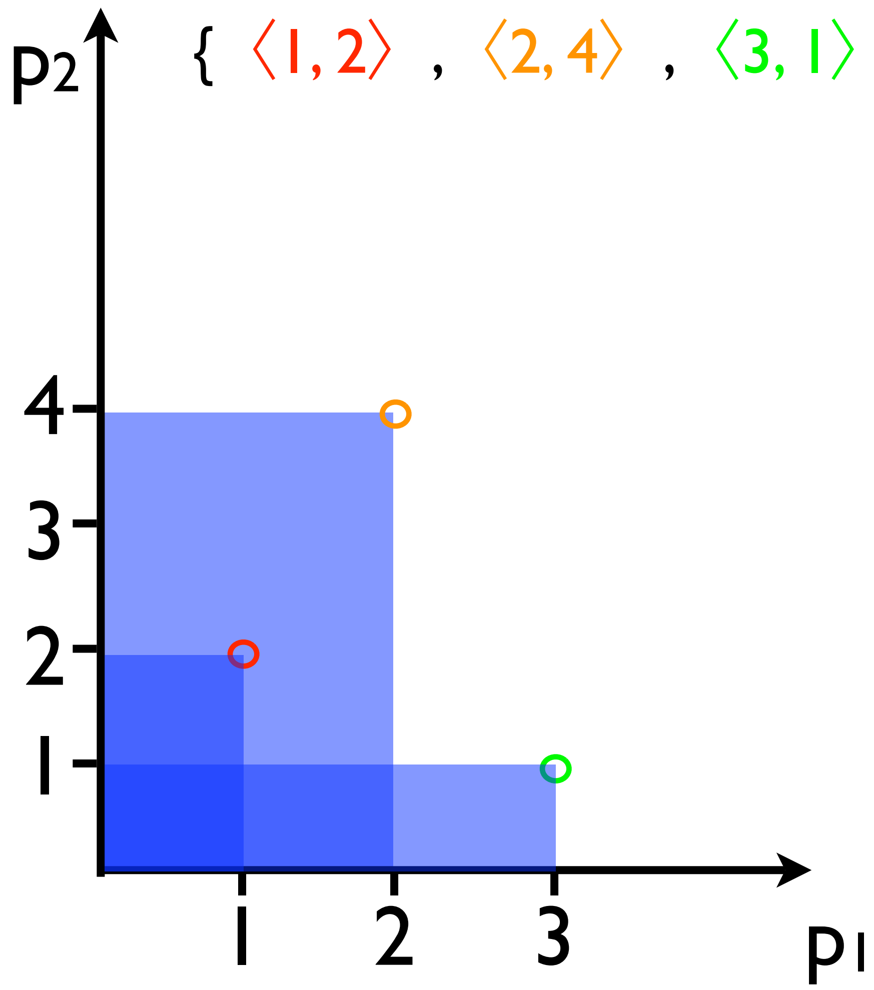


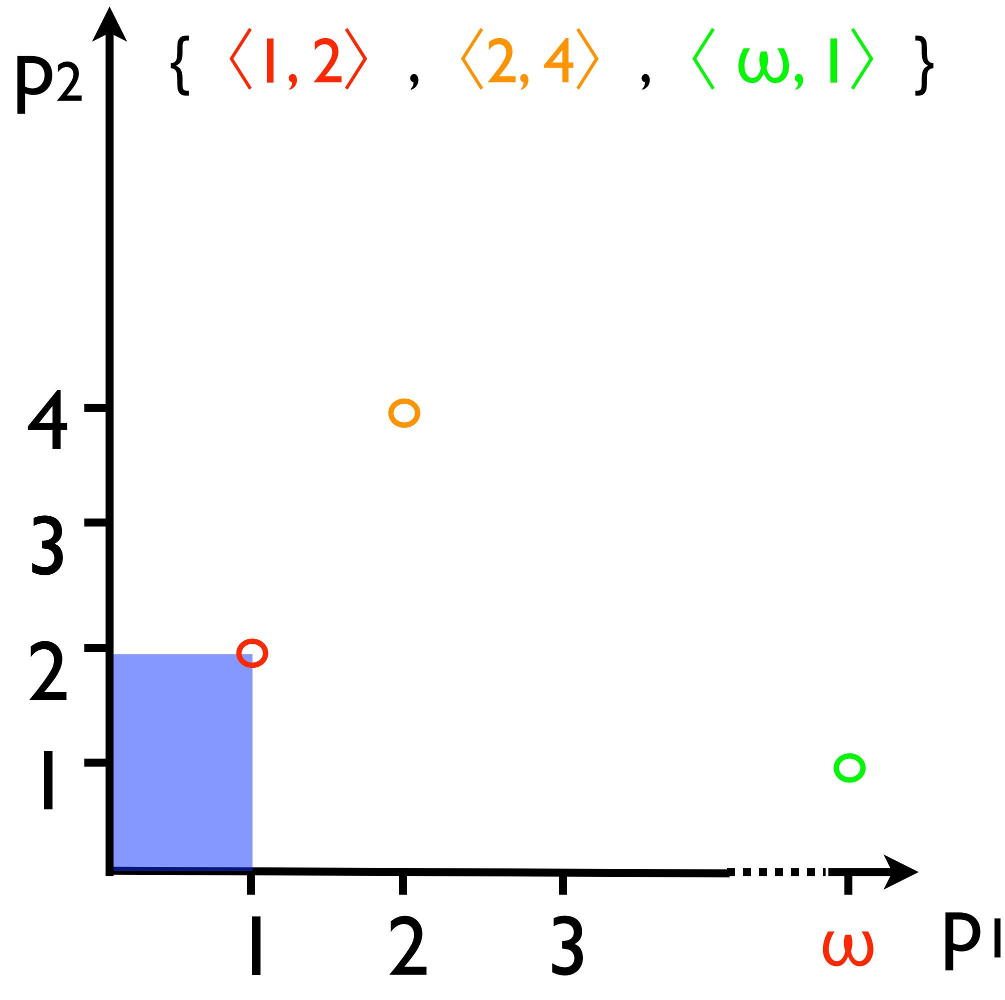



# Examples in 2 dim.

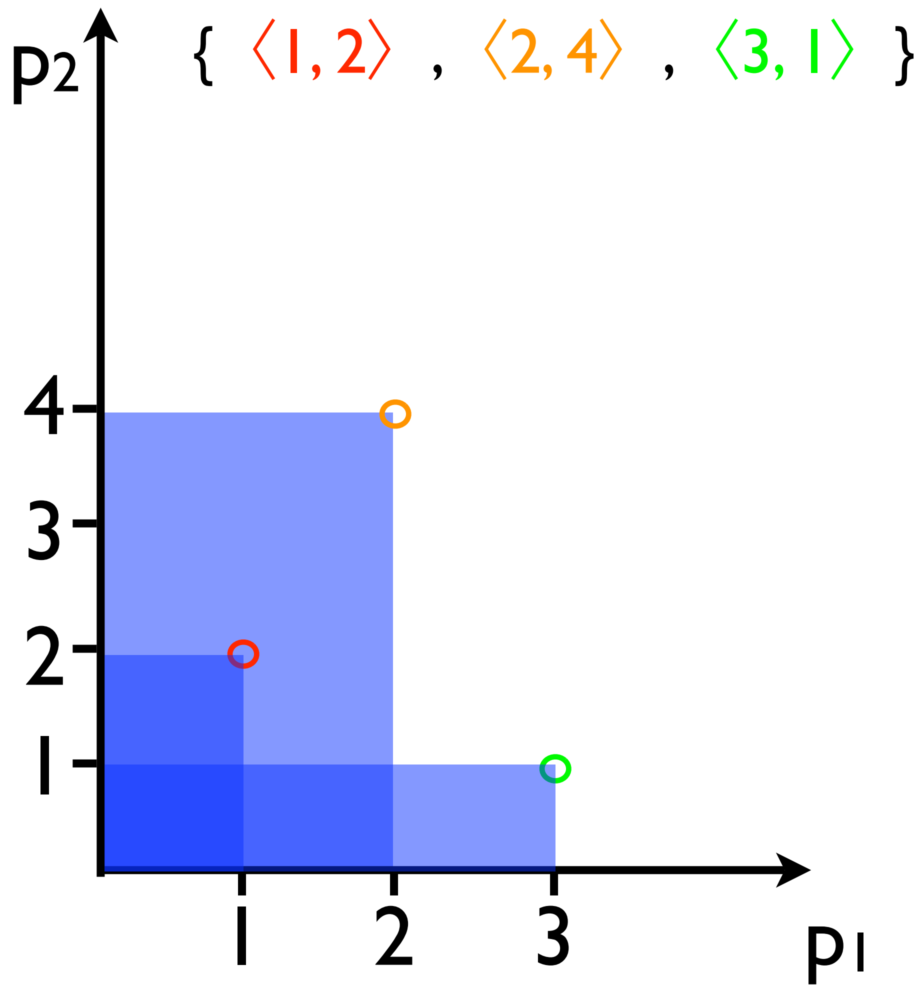


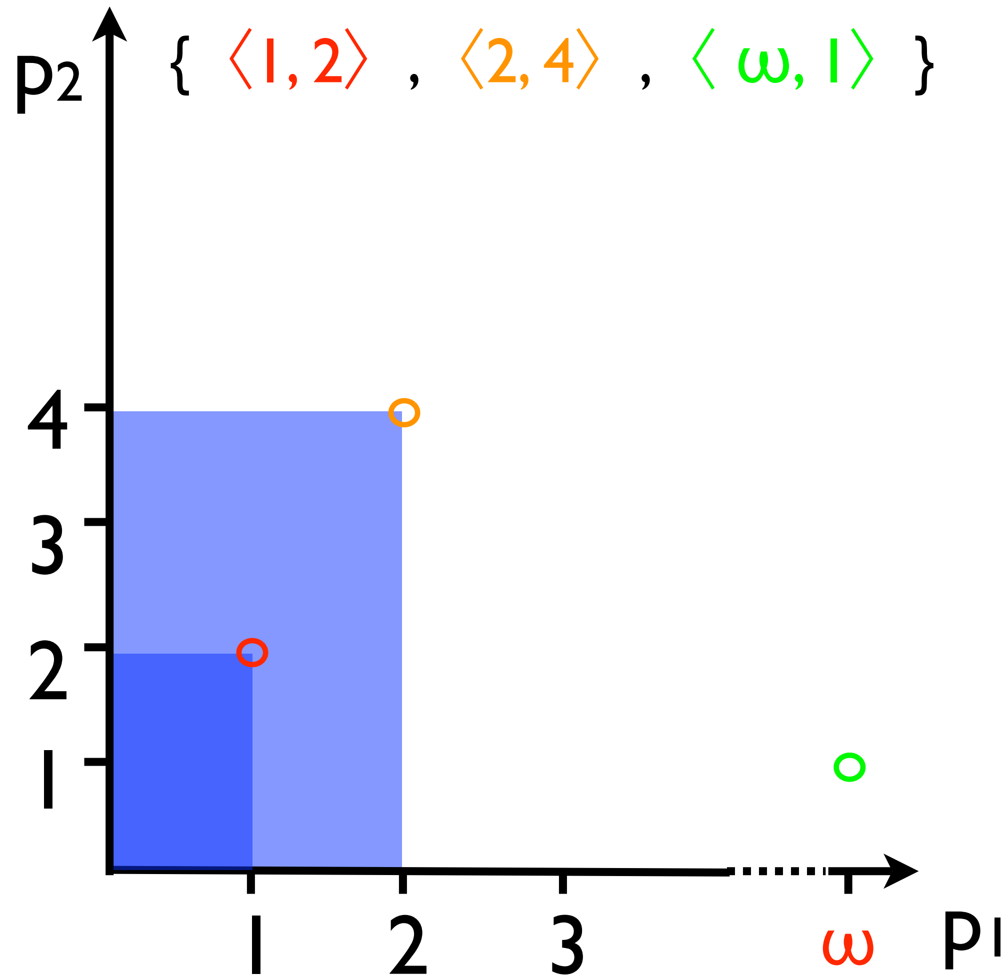
# Examples in 2 dim.



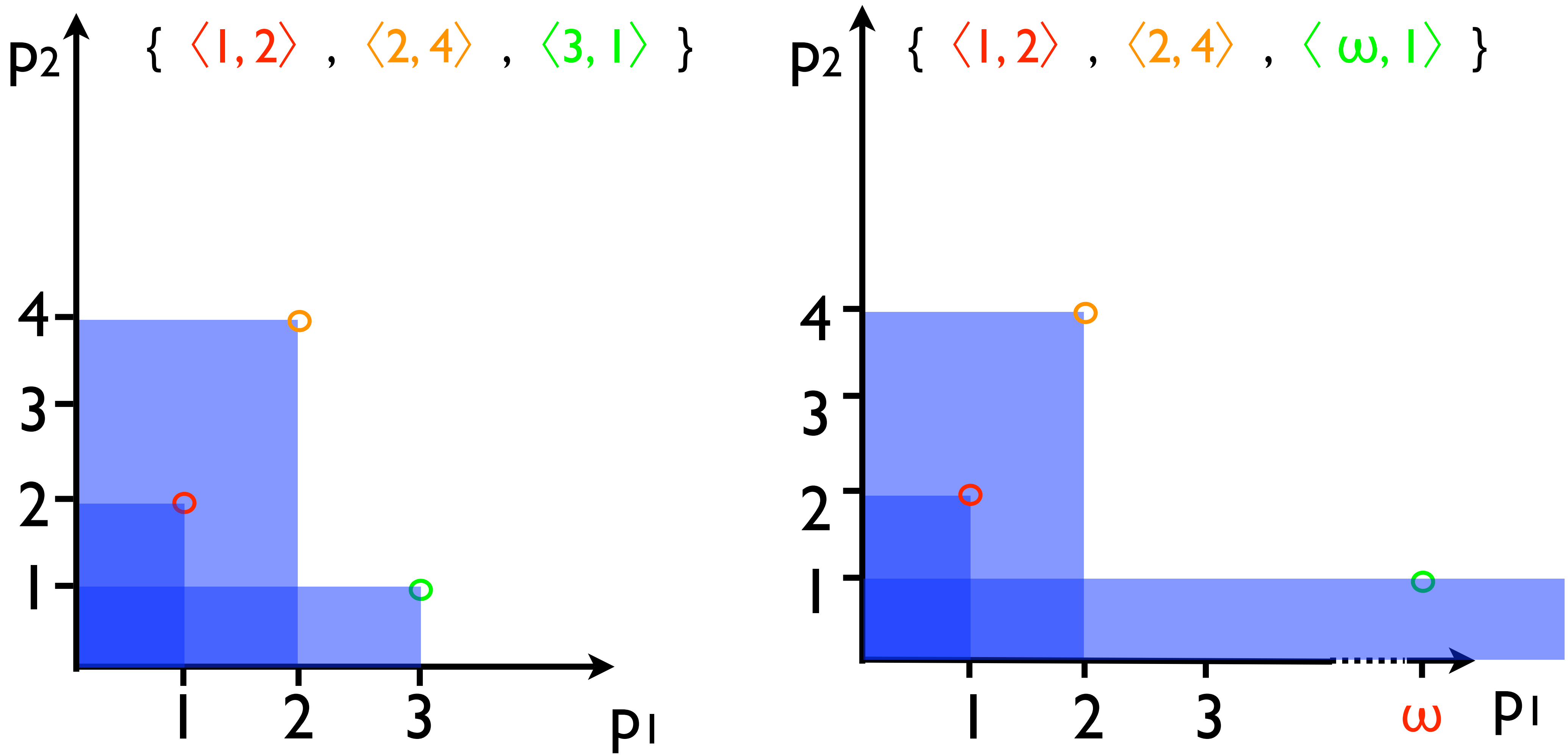



# Examples in 2 dim.



# Examples in 2 dim.



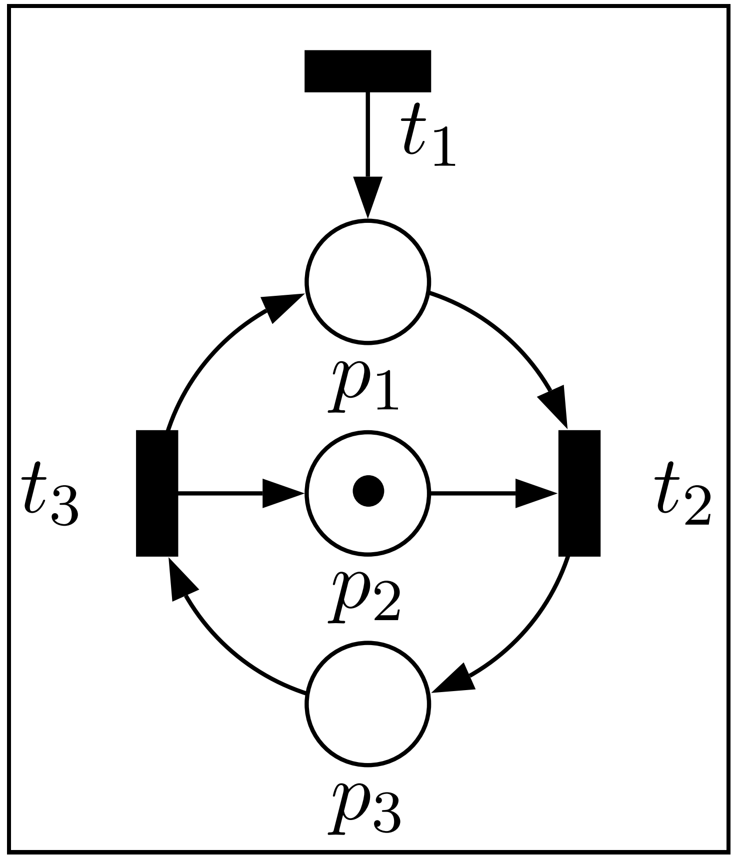
# Properties of the K&M tree

- The set of all the markings that appear in a K&M tree is called a coverability set of the net.
  - Notation: Cover(N)
- Theorem: $\downarrow$ Cover(N) = $\downarrow$ Reach(N)
- Theorem: Reach(N) $\subseteq$ $\downarrow$ Cover(N)
- Hence, $\downarrow$ Cover(N) is a finite over-approximation of Reach(N)

# Example

$$\mathrm{Reach}(N) = \{\ \langle i, 1, 0\rangle\ ,\ \langle i, 0, 1\rangle\ \mid i \geq 0\ \}$$

$$\mathrm{Cover}(N) = \downarrow\{\ \langle \omega, 1, 0\rangle\ ,\ \langle \omega, 0, 1\rangle\ \} = \mathrm{Reach}(N)\ \cup\ \{\ \langle 0, 0, 0\rangle\ \}$$




# Advertisement


- Recently, we have defined a new algorithm to compute the coverability set of a Petri net.

- It is several order of magnitudes more efficient than K&M

| Example | | | | | KM | | CovProc | | |
|---|---|---|---|---|---|---|---|---|---|
| Name | P | T | MCS | Tp | Nodes | Time | Max P. | Tot. P. | Time |
| CSM | 14 | 13 | 16 | U | $>2.40 \cdot 10^6$ | $\times$ | 178 | 248 | 0.34 |
| FMS | 22 | 20 | 24 | U | $>6.26 \cdot 10^5$ | $\times$ | 477 | 866 | 2.10 |
| PNCSA | 31 | 36 | 80 | U | $>1.02 \cdot 10^6$ | $\times$ | 2,617 | 13,408 | 113.79 |
| multipoll | 18 | 21 | 220 | U | $>1.16 \cdot 10^6$ | $\times$ | 14,034 | 14,113 | 365.90 |
| mesh2x2 | 32 | 32 | 256 | U | $>8.03 \cdot 10^5$ | $\times$ | 10,483 | 12,735 | 330.95 |

# The coverability problem

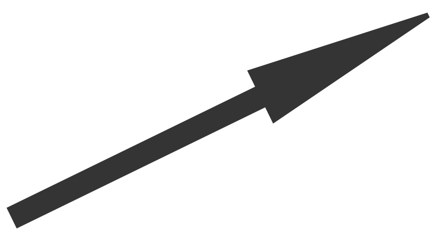
# Reachability: a natural question

- The reachability problem: given a marking $m$ is it reachable from $m_0$ ?

$m_0$

# Reachability: a natural question

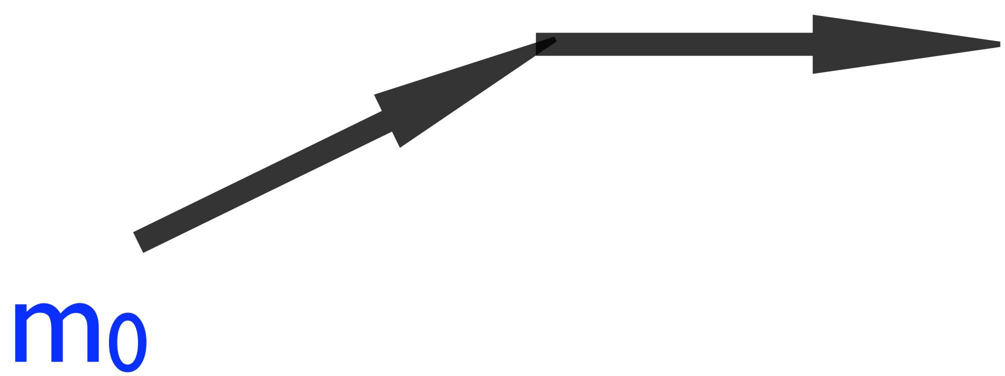
- The reachability problem: given a marking $m$ is it reachable from $m_0$ ?



$m_0$

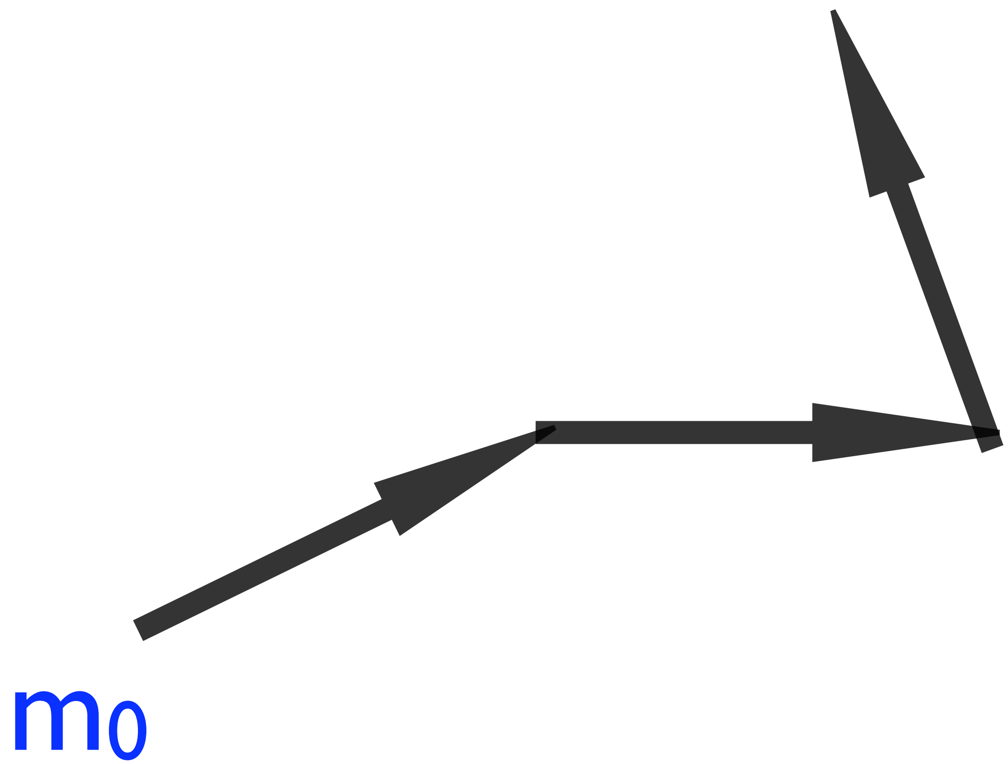
# Reachability: a natural question

- The reachability problem: given a marking m is it reachable from $m_0$ ?



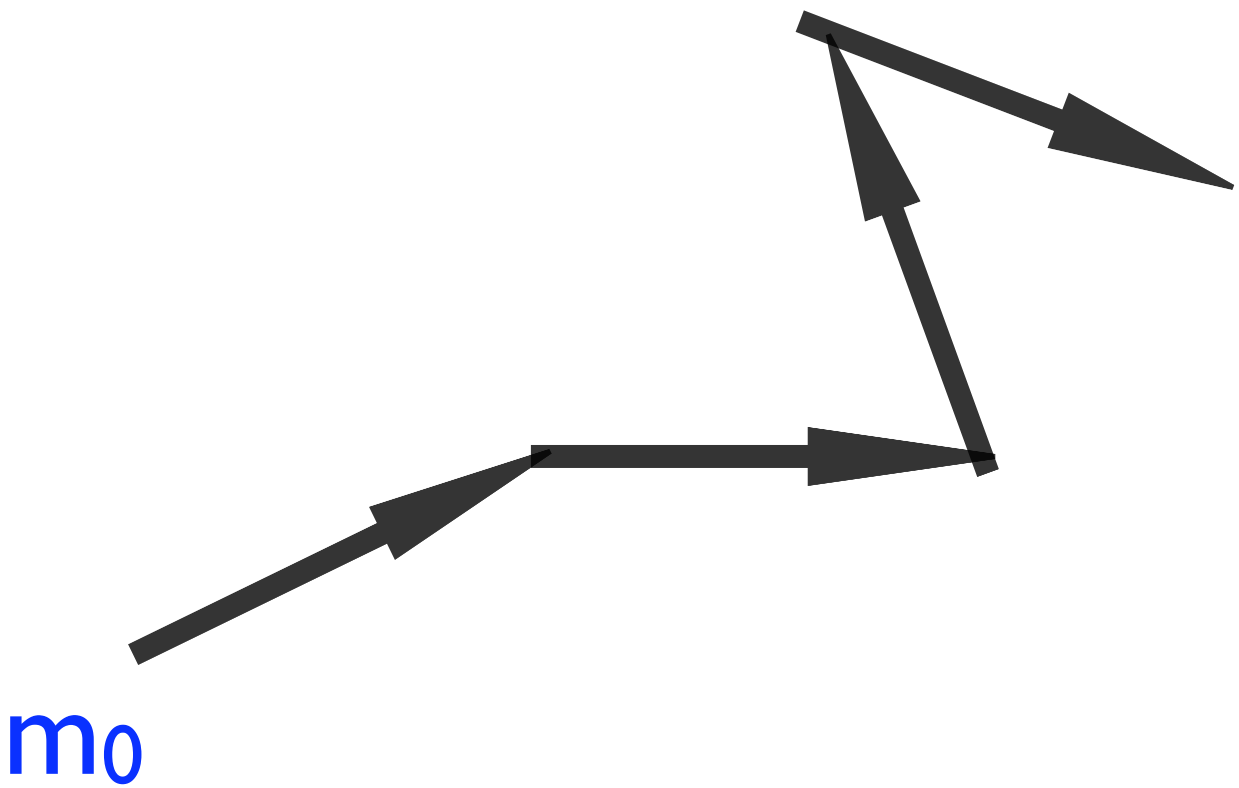
# Reachability: a natural question

- The reachability problem: given a marking $m$ is it reachable from $m_0$ ?



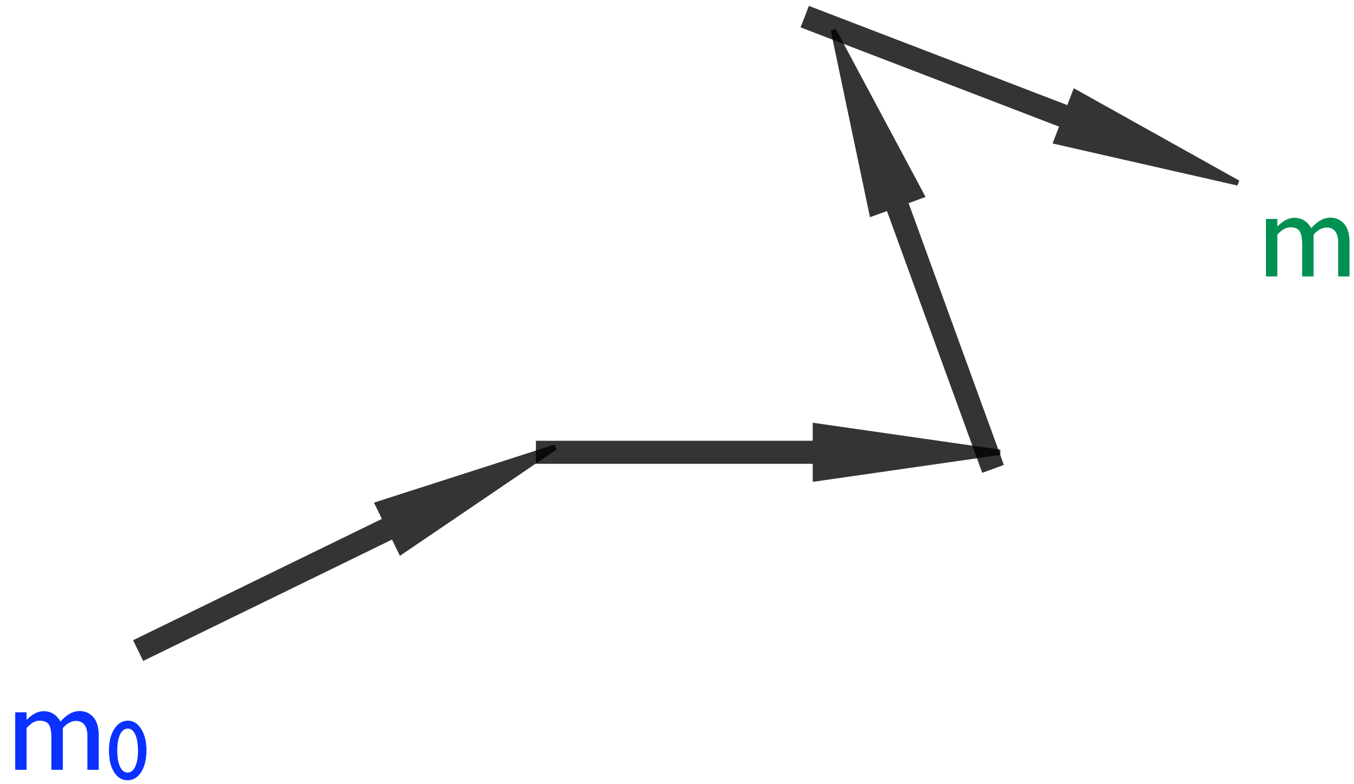
# Reachability: a natural question

- The reachability problem: given a marking $m$ is it reachable from $m_0$ ?



# Reachability: a natural question

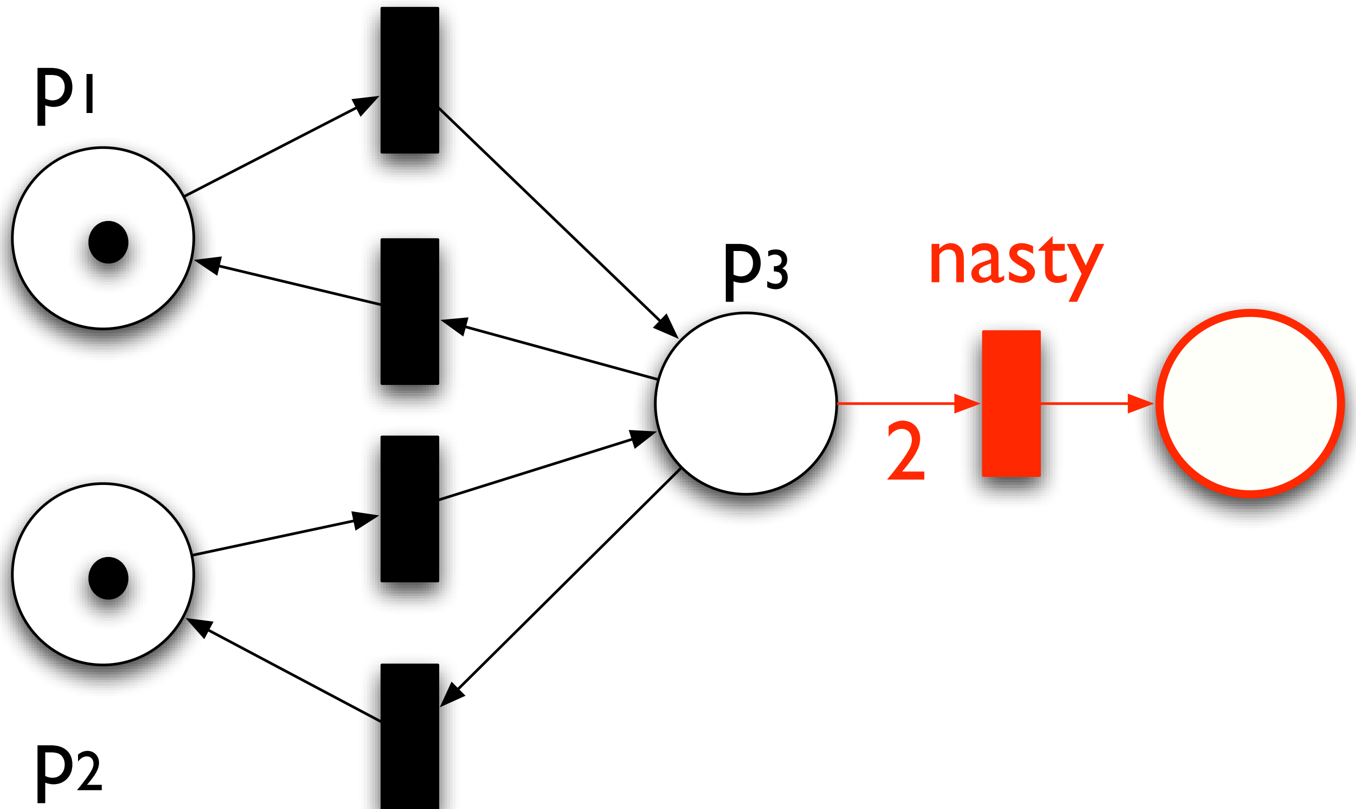
- The reachability problem: given a marking m is it reachable from $m_0$ ?



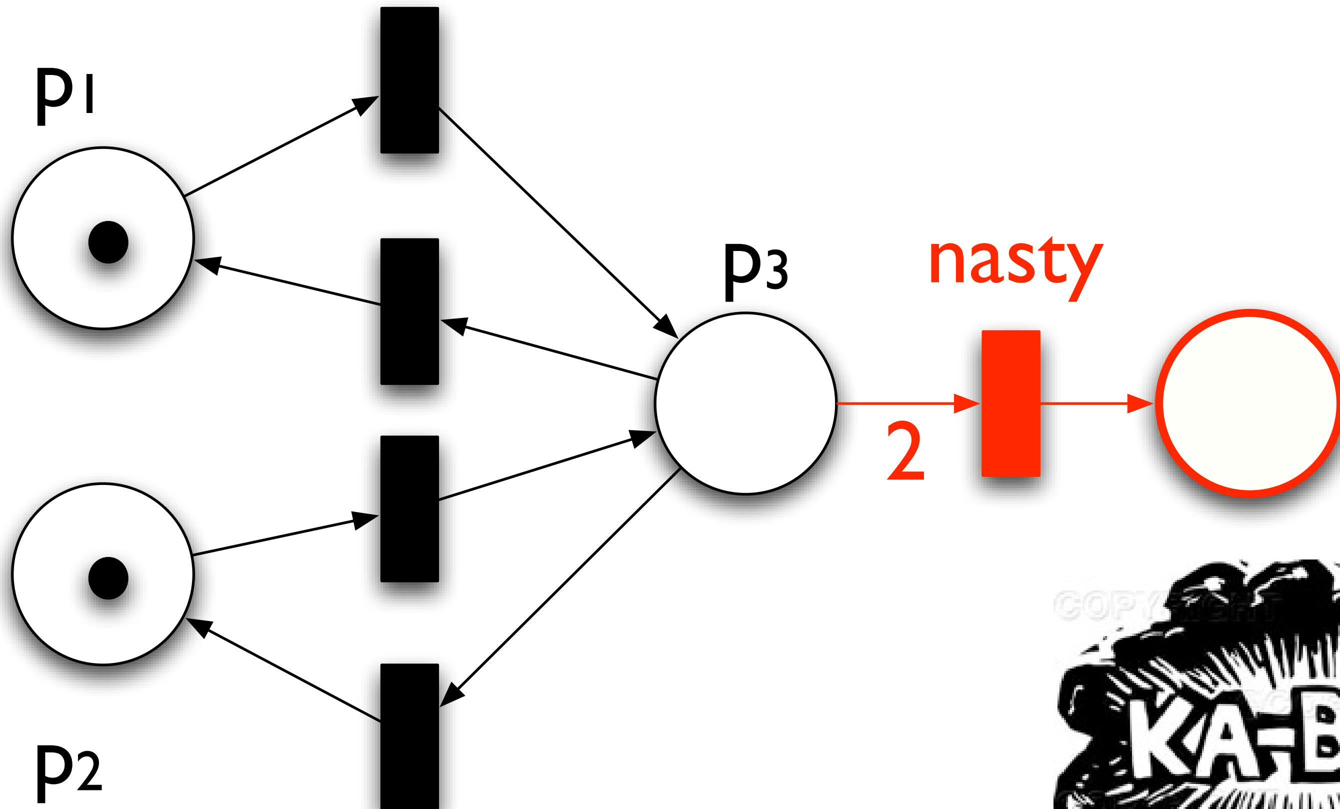
# Reachability: a natural question ??

- In the case  of Petri nets, asking whether a given marking is reachable does not always make sense...
- ... because Petri nets are monotonic

# Example



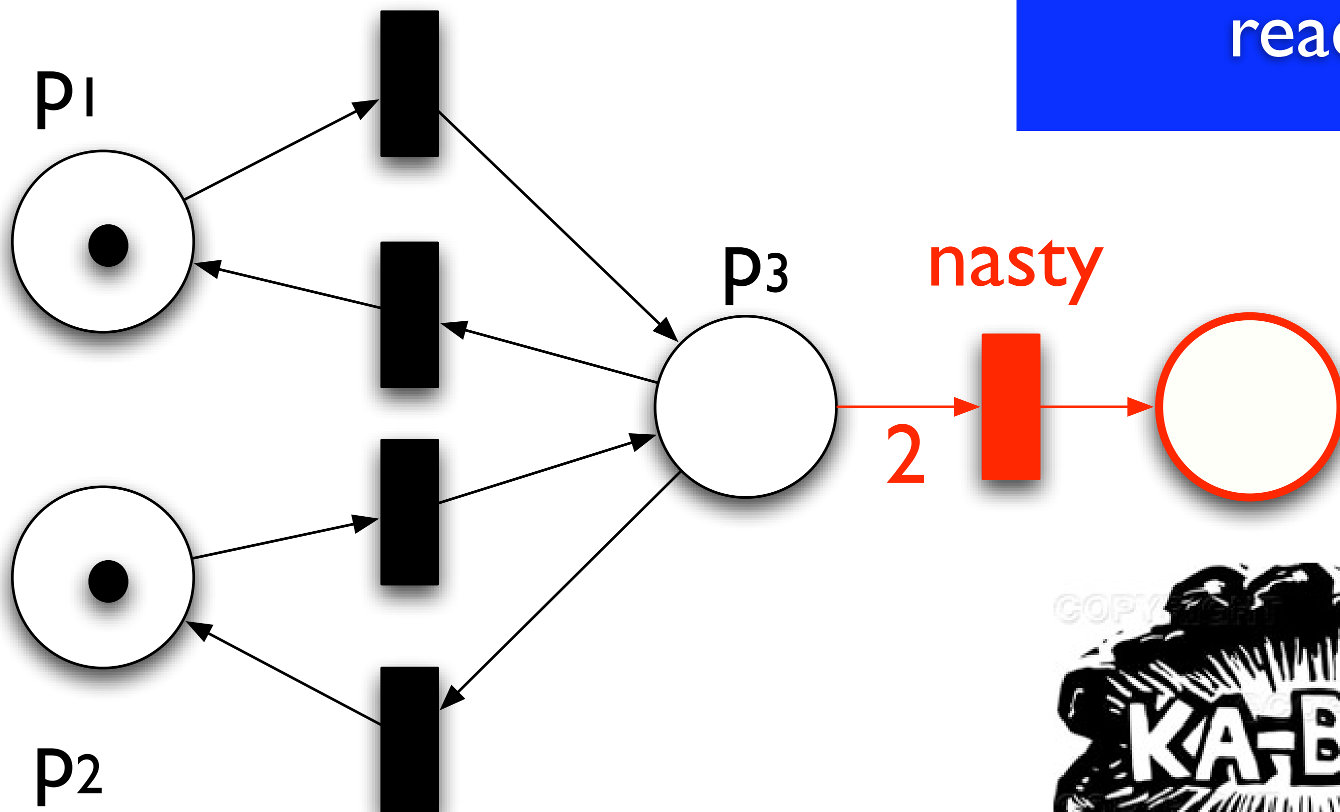
# Example




# Example

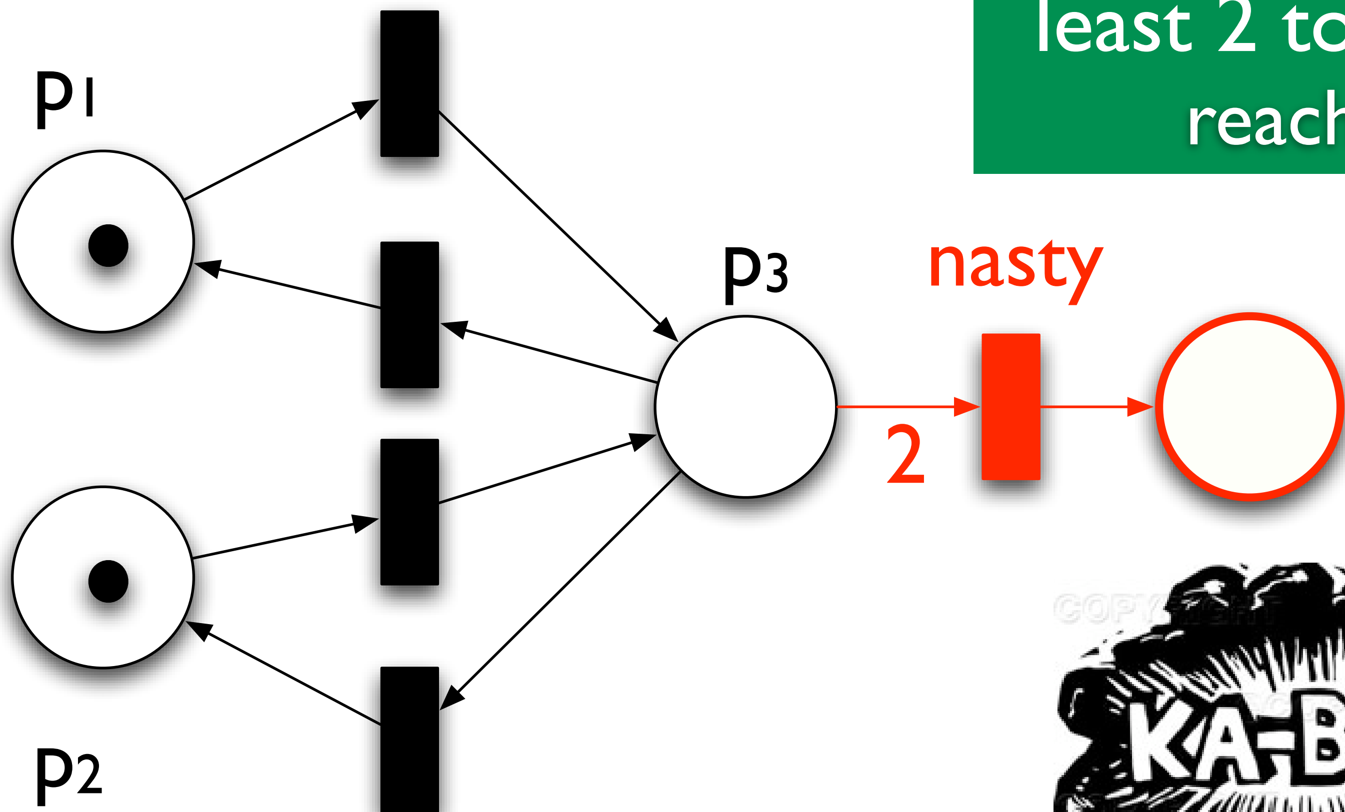
**Question**

is $\langle 0, 0, 2, 0 \rangle$ reachable ?



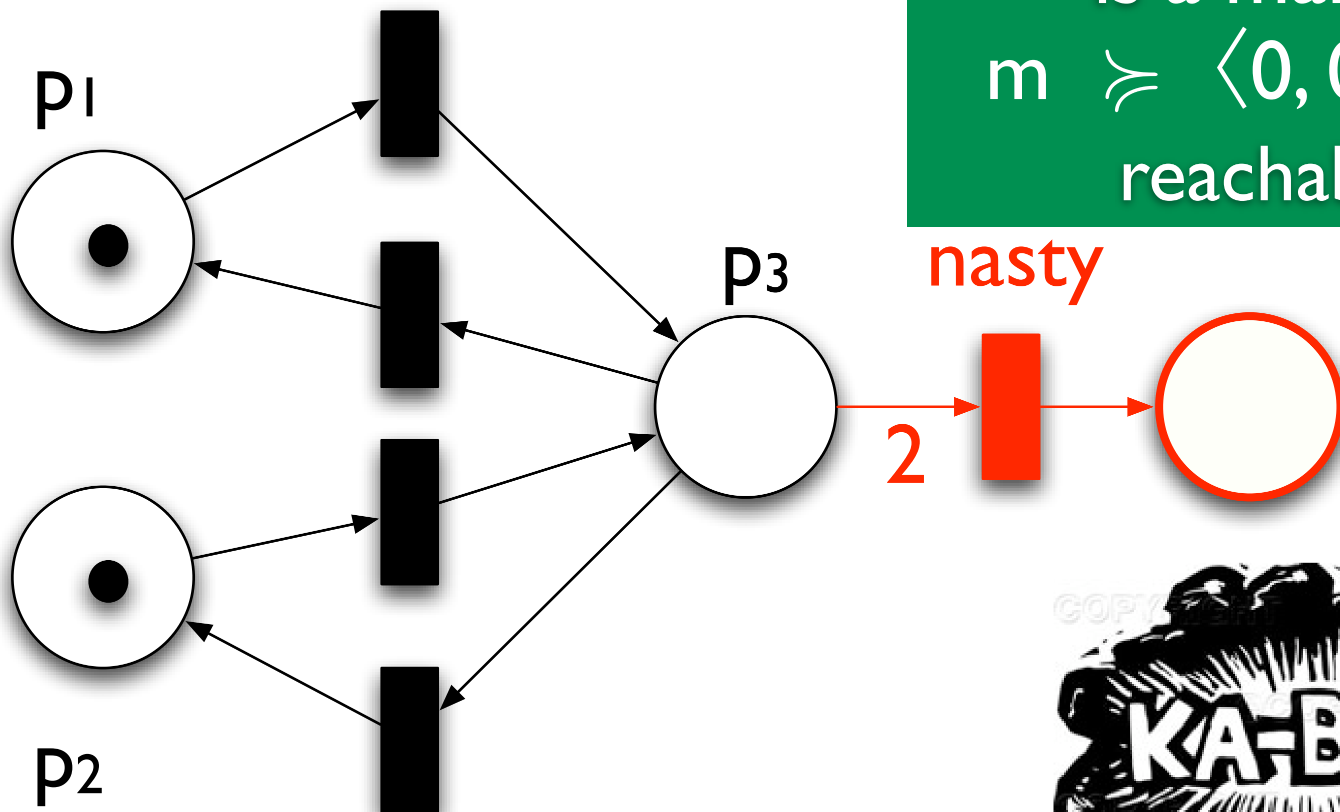

# Example

Better question
is a marking with at least 2 tokens in p3 reachable ?




# Example

Better question
is a marking
$m \succcurlyeq \langle 0, 0, 2, 0 \rangle$
reachable ?




# The coverability problem

Does there exist a reachable marking which is larger than some marking b ?

$b$

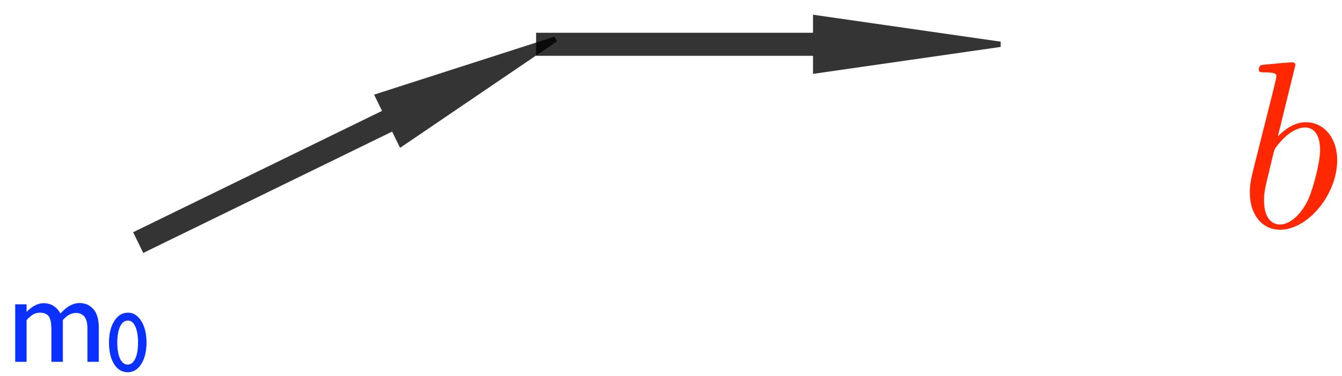
$m_0$

# The coverability problem

Does there exist a reachable marking which is larger than some marking b ?

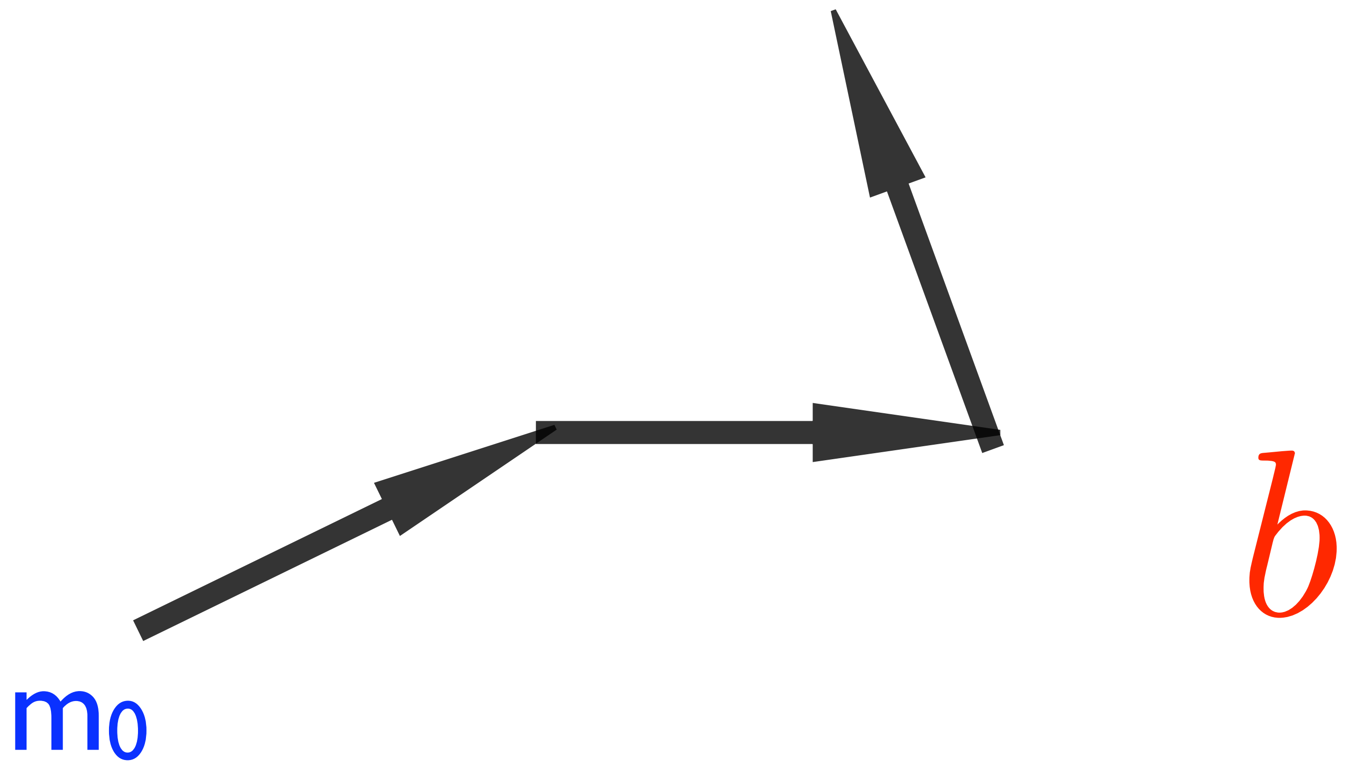
# The coverability problem

Does there exist a reachable marking which is larger than some marking b ?



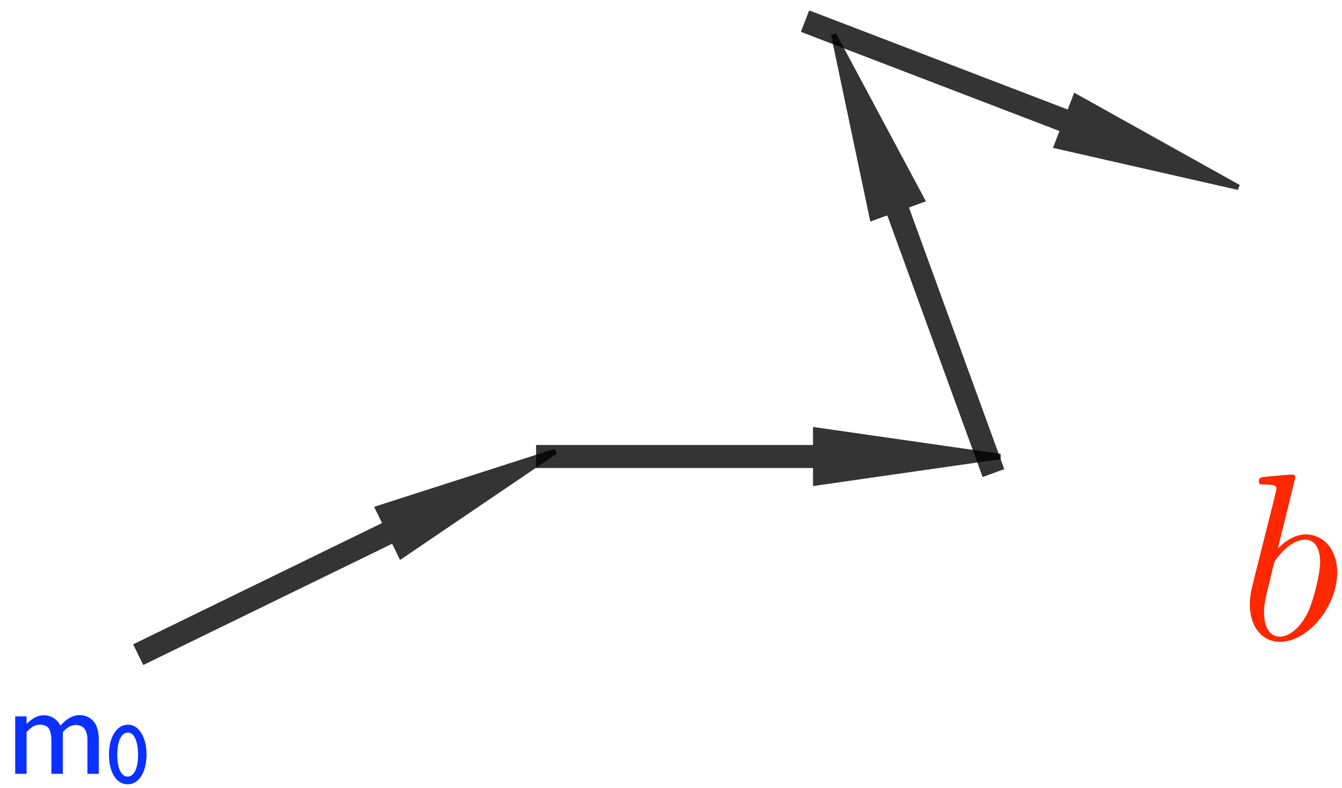
# The coverability problem

Does there exist a reachable marking which is larger than some marking b ?



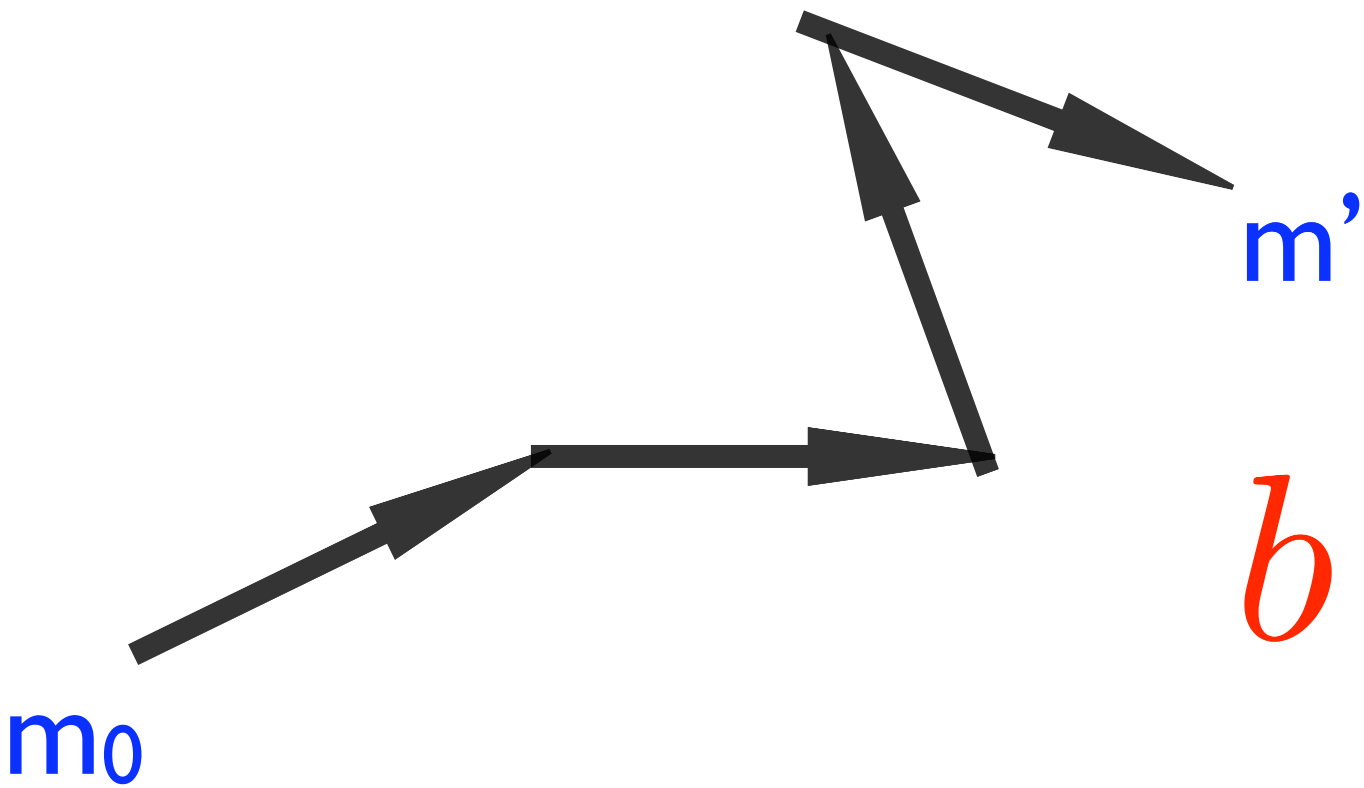
# The coverability problem

Does there exist a reachable marking which is larger than some marking b ?



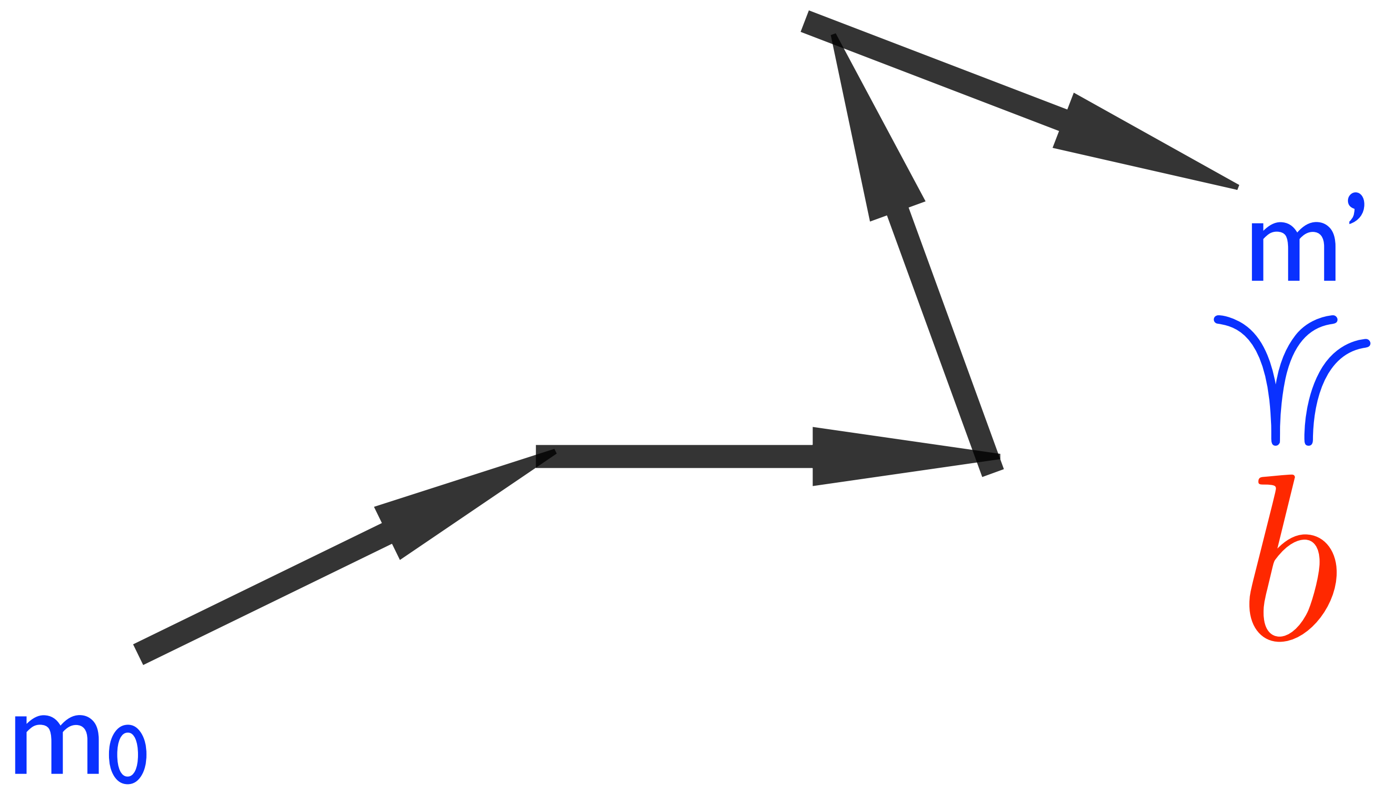
# The coverability problem

Does there exist a reachable marking which is larger than some marking b ?



# The coverability problem

Does there exist a reachable marking which is larger than some marking b ?



# The coverability problem

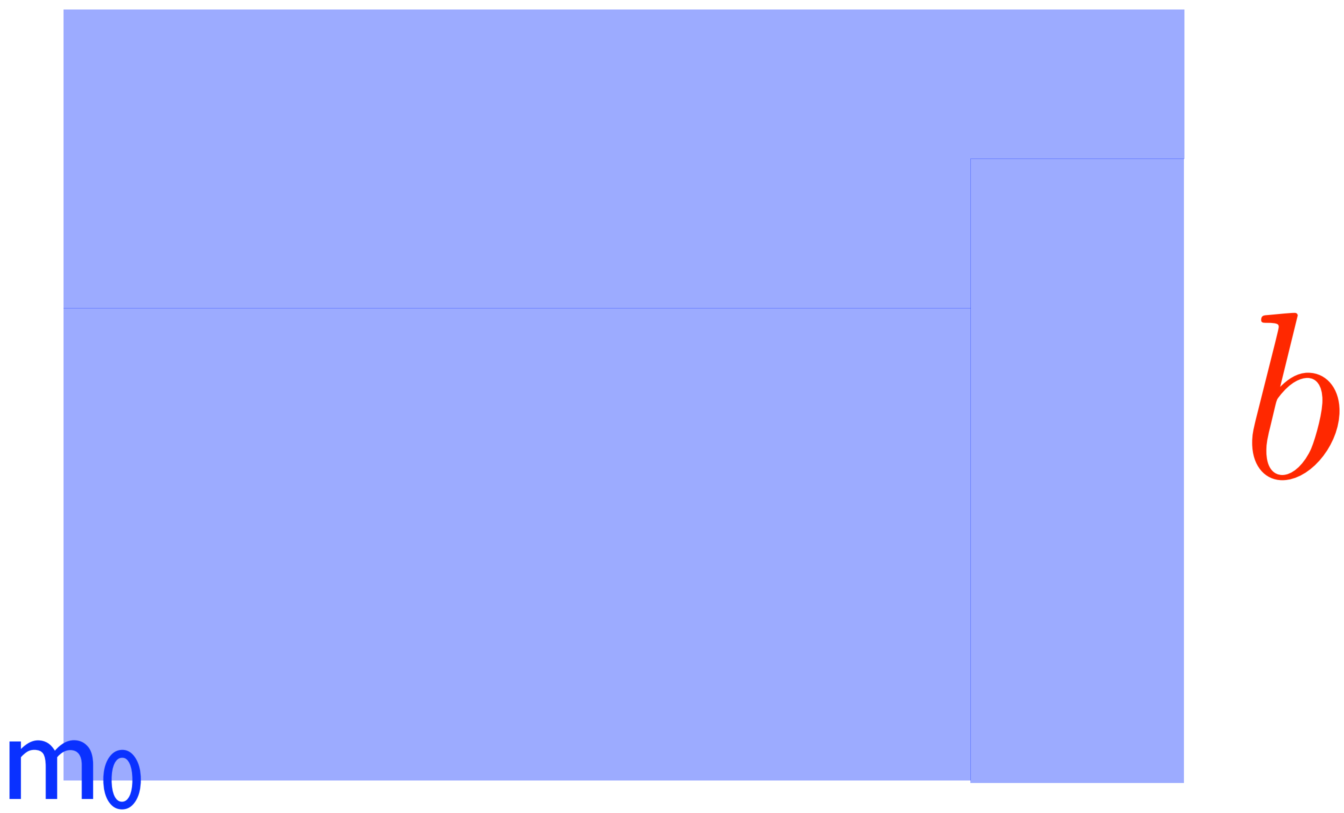
$b$

$m_0$

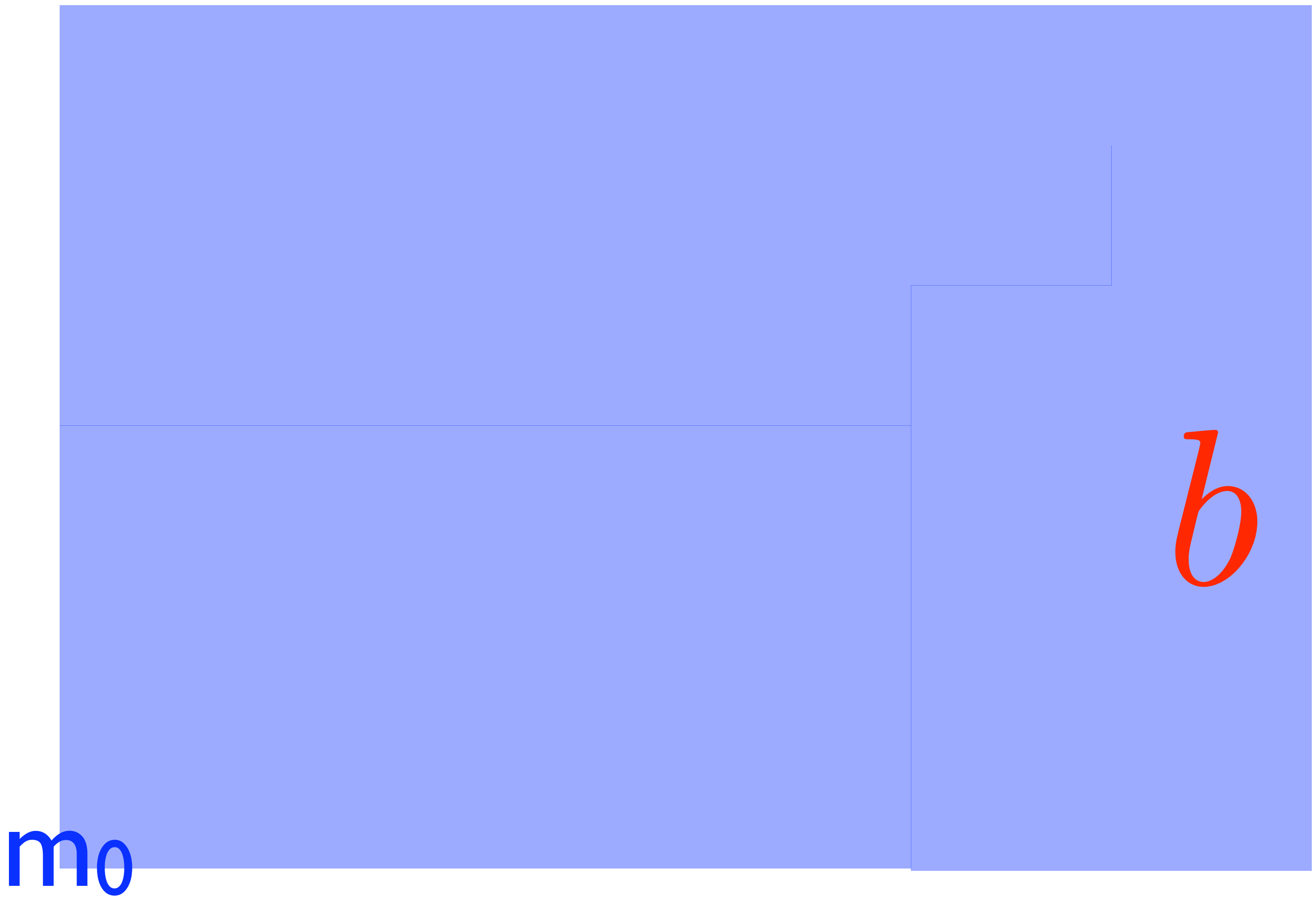
# The coverability problem

# The coverability problem

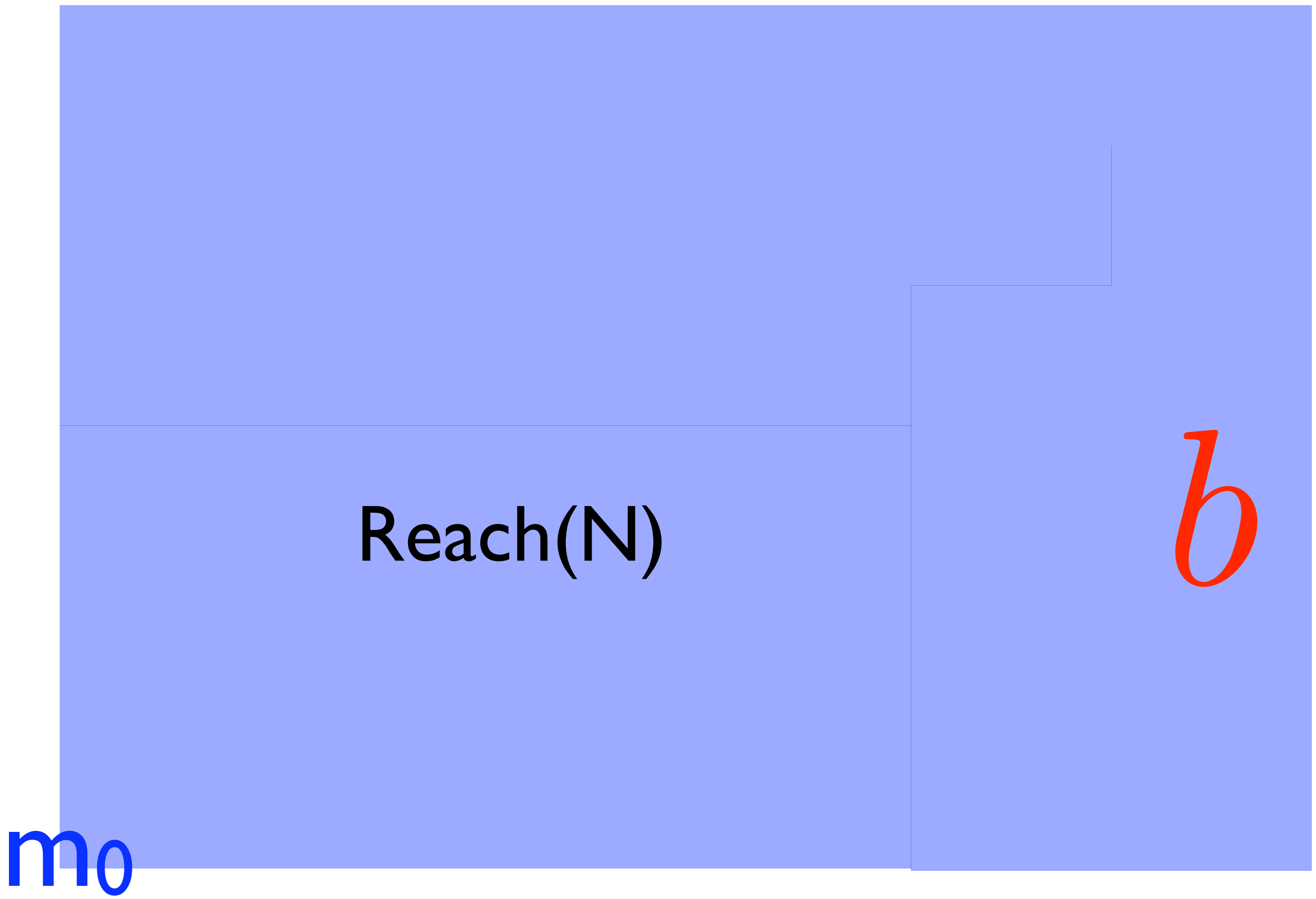
# The coverability problem



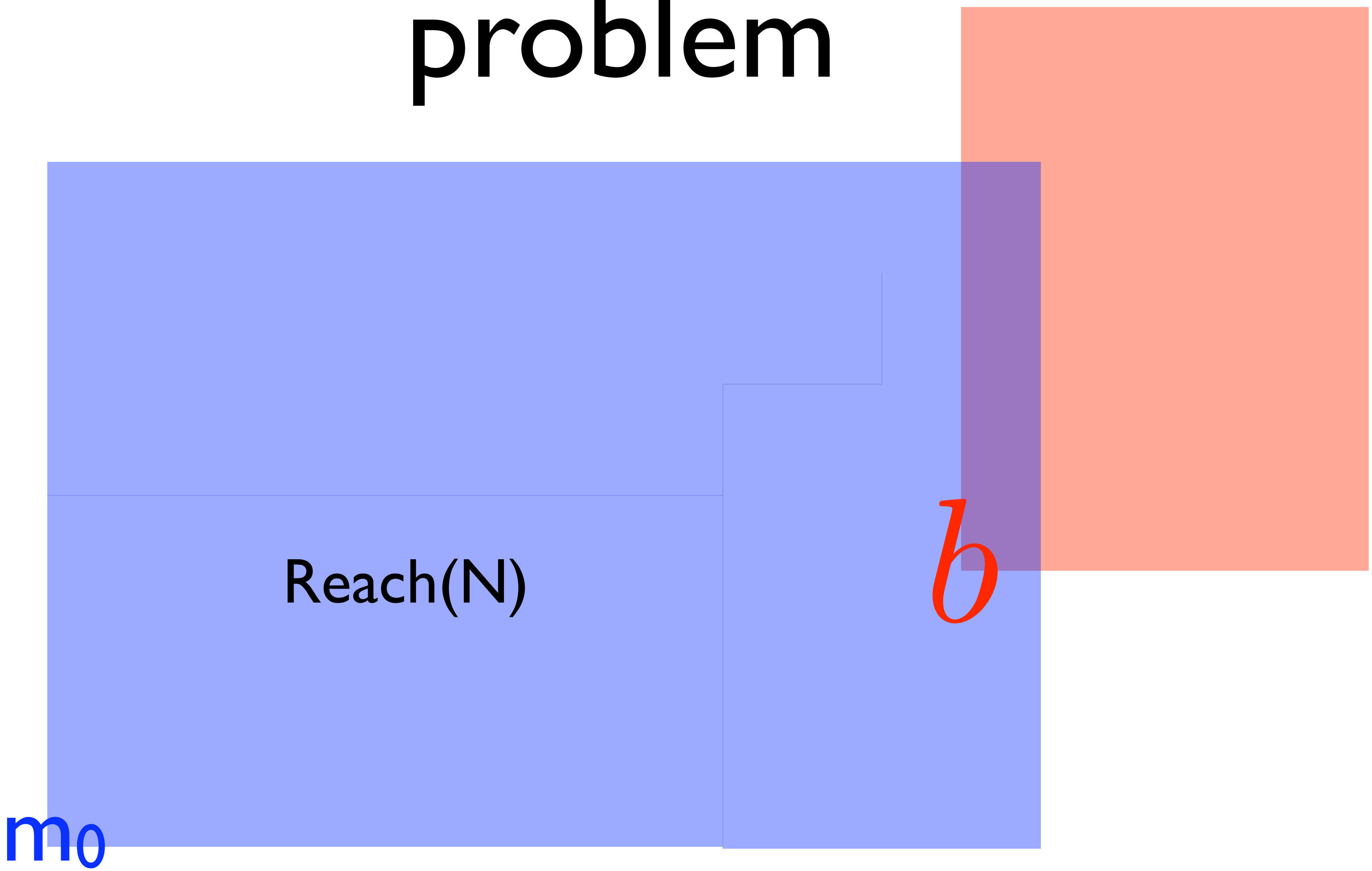
# The coverability problem



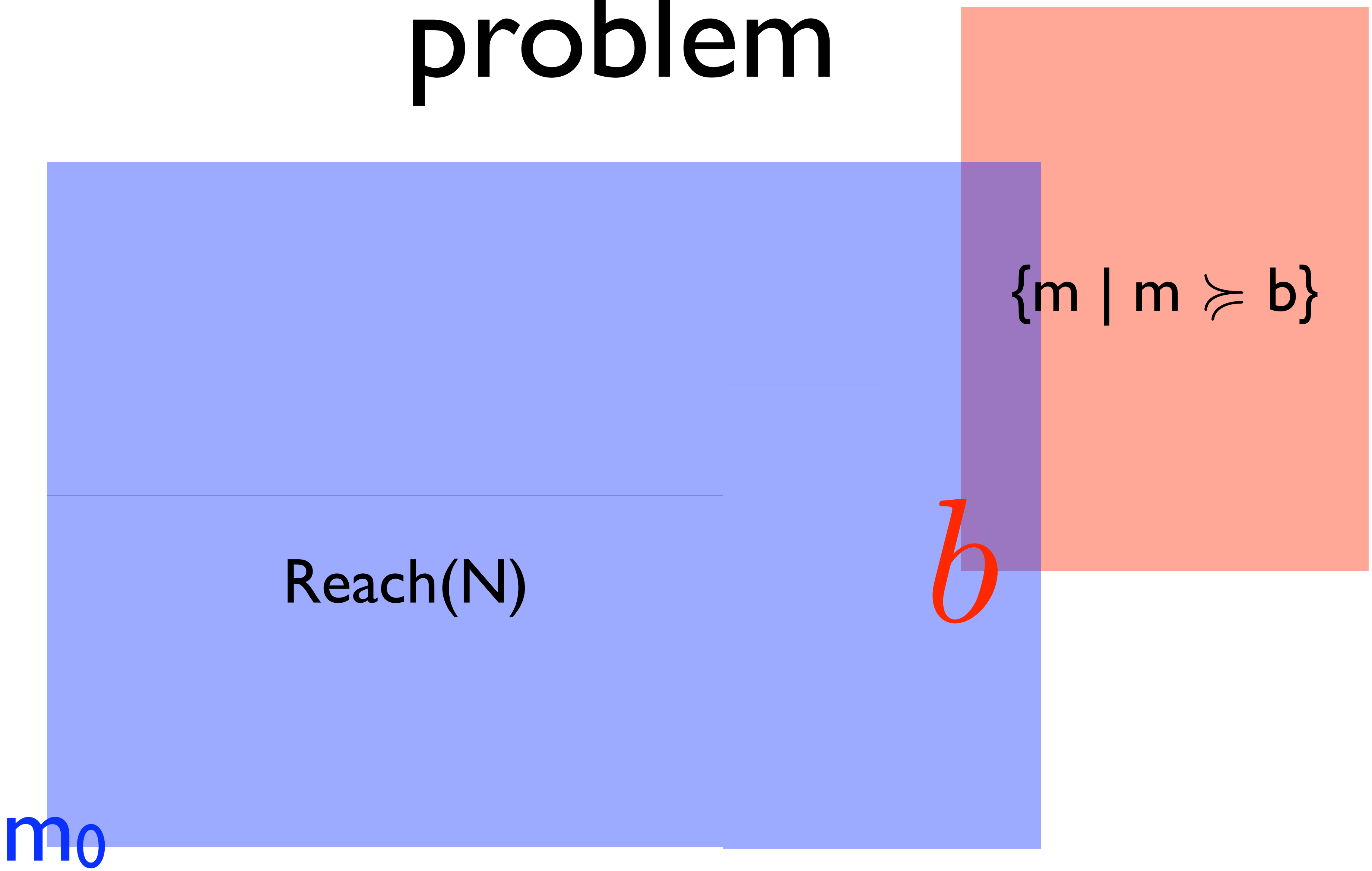
# The coverability problem



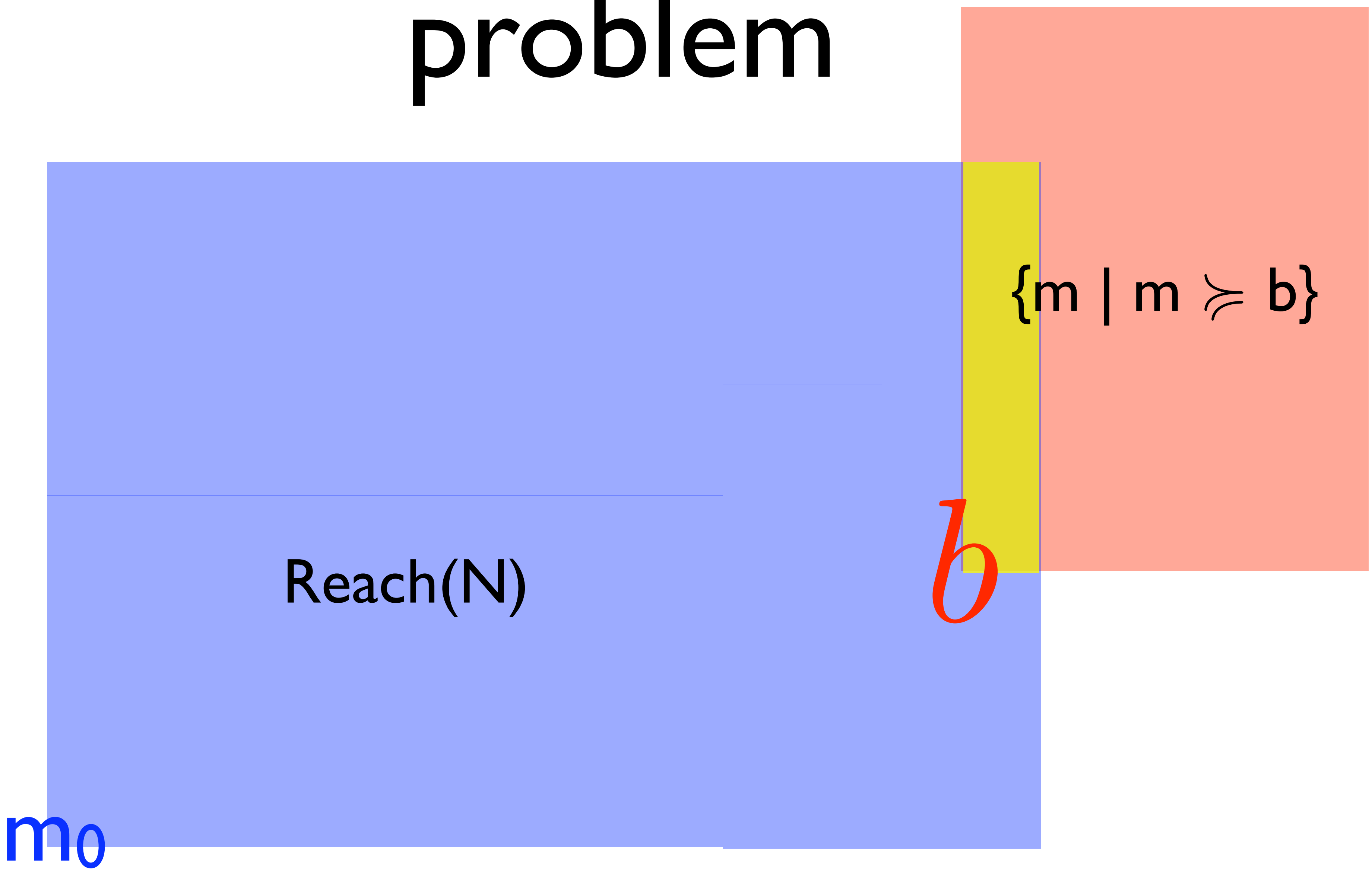
# The coverability problem



# The coverability problem



# The coverability problem



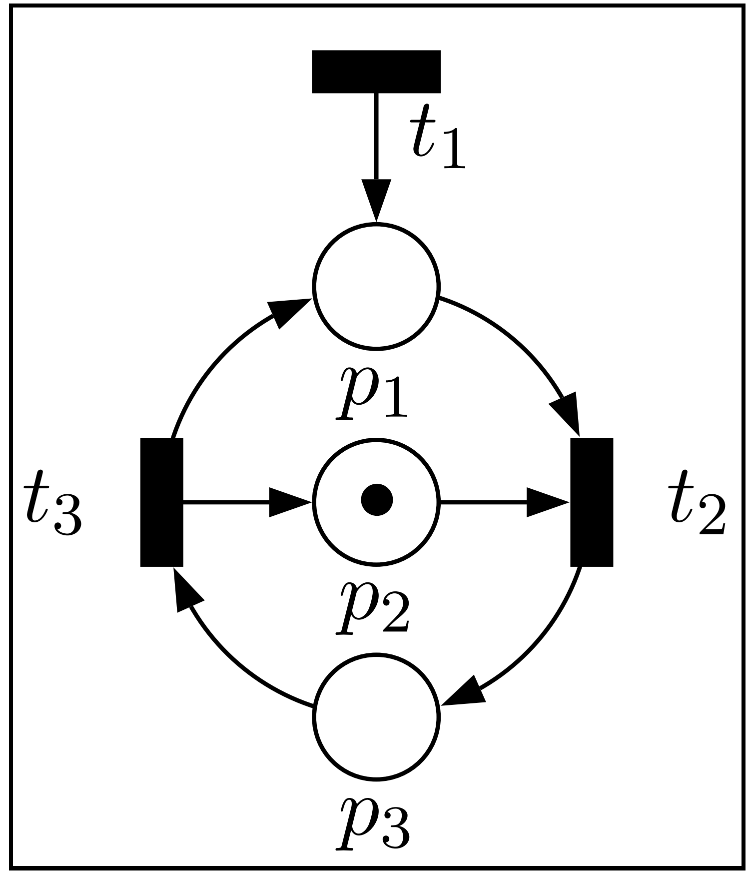
# The coverability problem

- Two alternative definitions:
  - Is there a reachable marking m s.t. $m \succcurlyeq b$ ?
  - Does $Reach(N) \cap \{m \mid m \succcurlyeq b\} \neq \phi$ ?

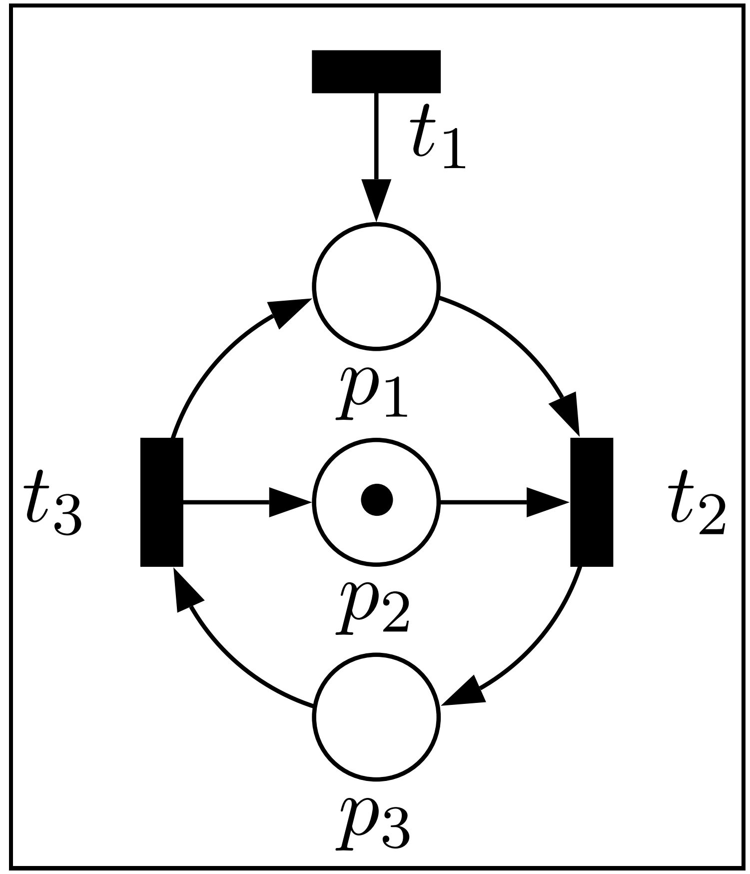
# Coverability: a natural question (indeed)

- Coverability might be regarded as the most natural reachability question in the framework of Petri nets
- Besides, coverability is much more easily solved than reachability

# Safety Properties



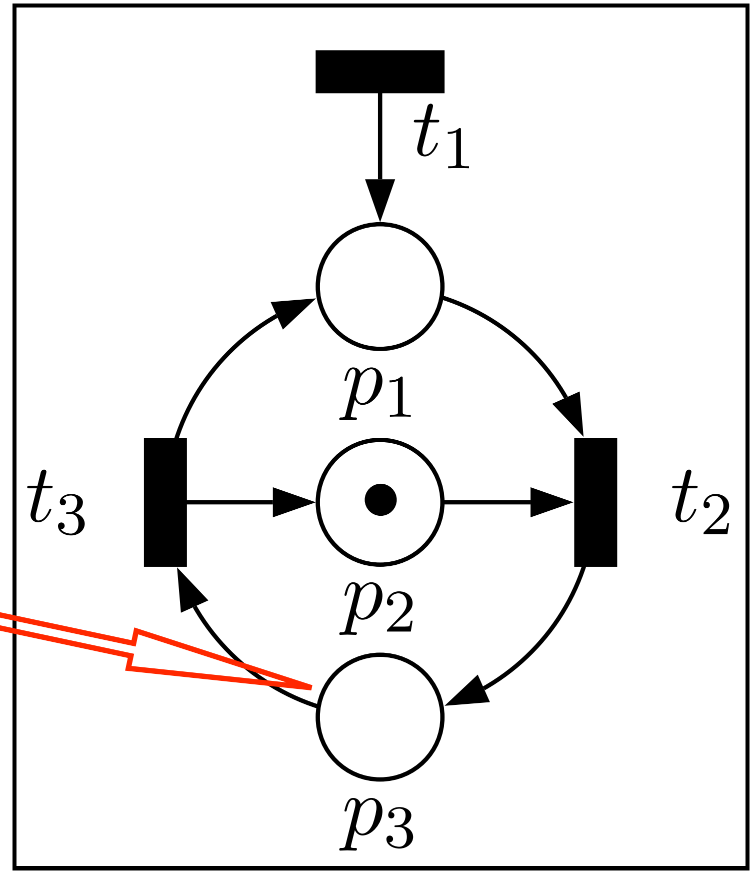
# Safety Properties




A marking m is unsafe when $m \succcurlyeq \langle 0, 0, 2, 0 \rangle$

# Safety Properties

No more than one token at a time in this place !!




A marking m is unsafe when $m \succcurlyeq \langle 0, 0, 2, 0 \rangle$

# First idea

- Use the coverability set !
- Remember: the coverability set over-approximates the reachable states:

$$\mathrm{Reach}(N) \subseteq \downarrow \mathrm{Cover}(N)$$

# First idea

- Use the coverability set !
- Remember: the coverability set over-approximates the reachable states:

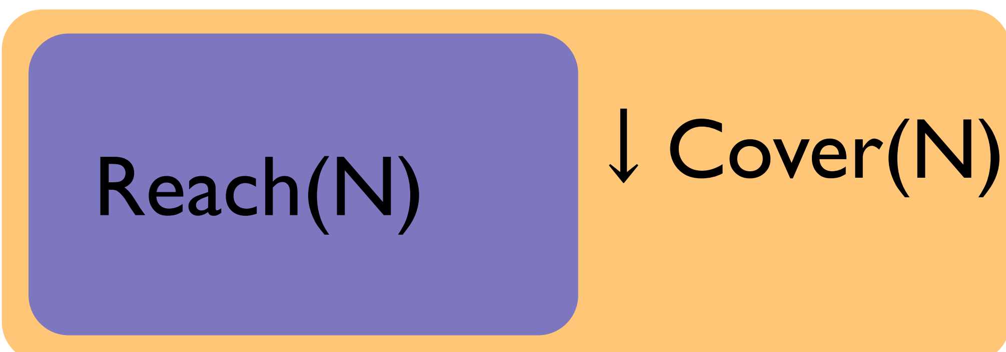
$$\text{Reach}(N) \subseteq \downarrow \text{Cover}(N)$$

Reach(N)

# First idea

- Use the coverability set !
- Remember: the coverability set over-approximates the reachable states:

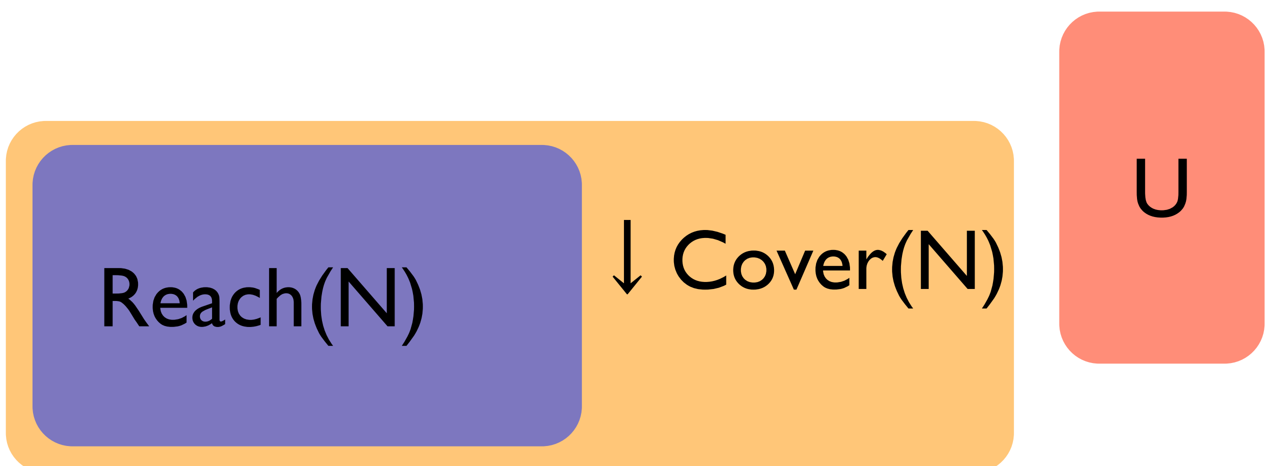
$$\mathrm{Reach}(N) \subseteq \downarrow \mathrm{Cover}(N)$$



# First idea

- Use the coverability set !
- Remember: the coverability set over-approximates the reachable states:

$$\mathrm{Reach}(N) \subseteq \downarrow \mathrm{Cover}(N)$$

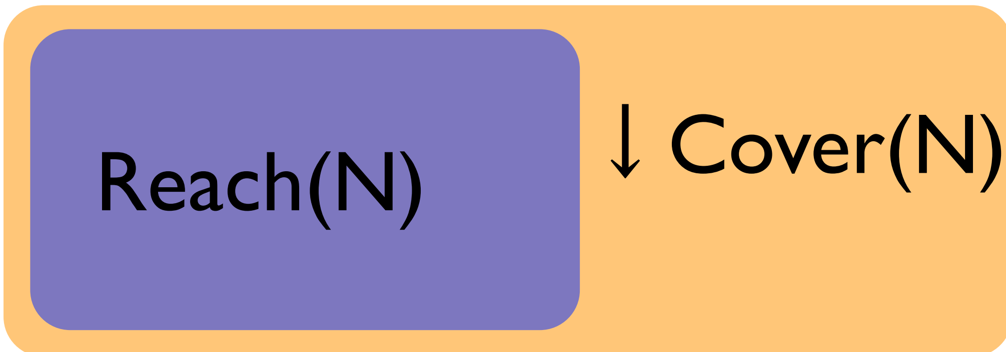


# First idea

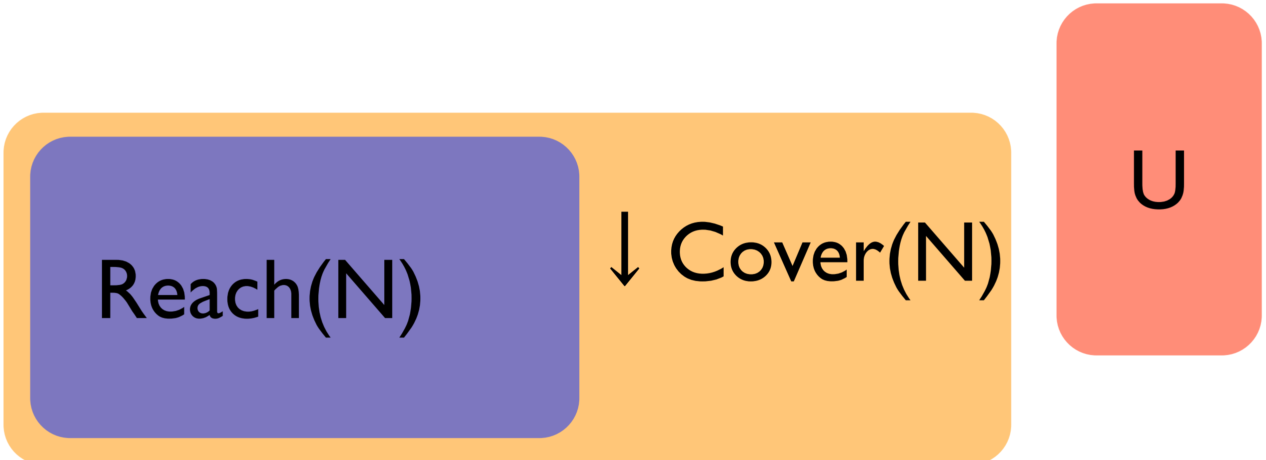
# First idea

Reach(N)

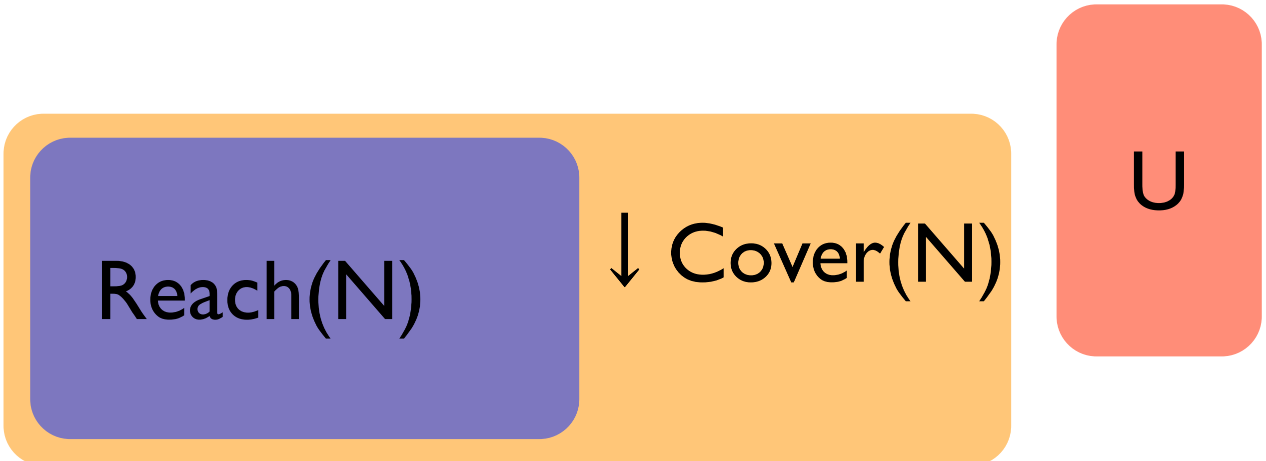
# First idea



# First idea



# First idea

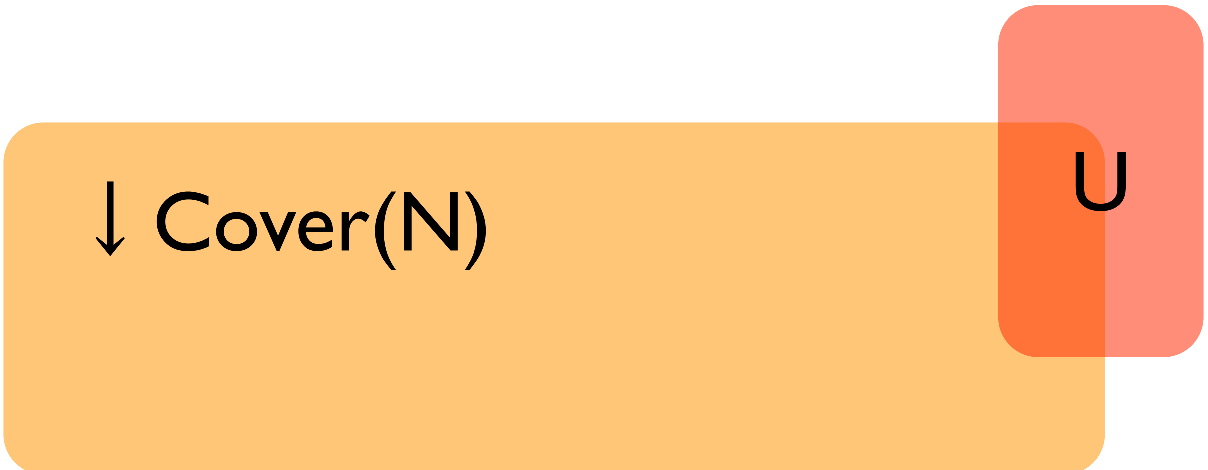



$$\downarrow \mathrm{Cover}(N) \cap U = \phi$$

implies

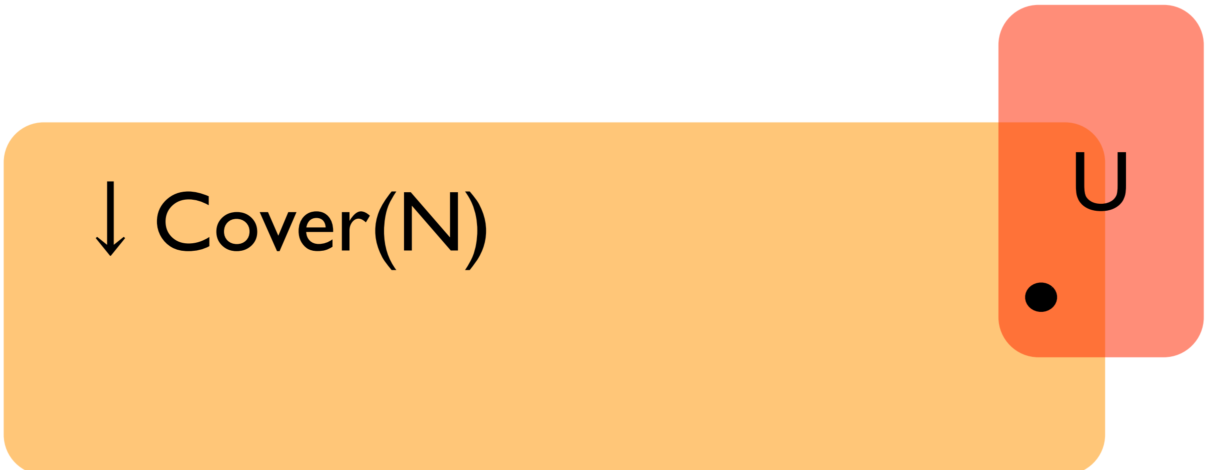
$$\mathrm{Reach}(N) \cap U = \phi$$

# What if ?



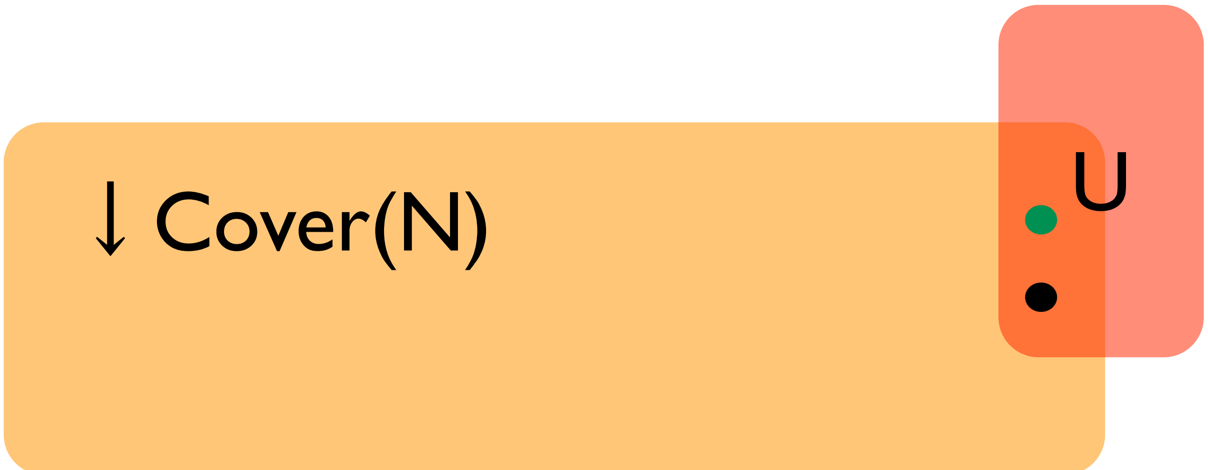

- There is m in $\downarrow$ Cover(N) $\cap$ U
- Hence, there is m' $\succcurlyeq$ m which is in Reach(N)
- However, any m' $\succcurlyeq$ m is also in U
- Thus, there is m' both in Reach(N) and U

# What if ?



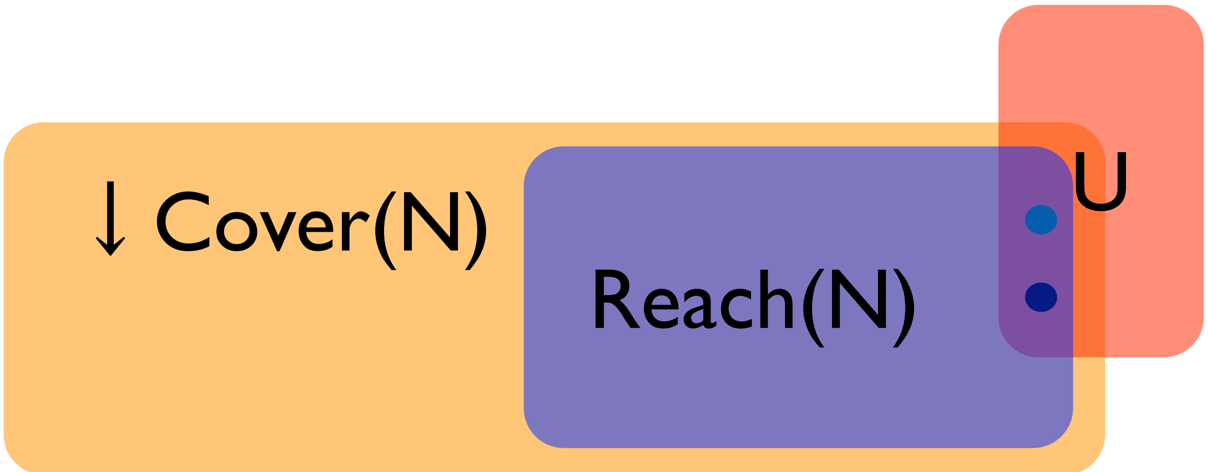

- There is m in $\downarrow$ Cover(N) $\cap$ U
- Hence, there is m' $\succcurlyeq$ m which is in Reach(N)
- However, any m' $\succcurlyeq$ m is also in U
- Thus, there is m' both in Reach(N) and U

# What if ?



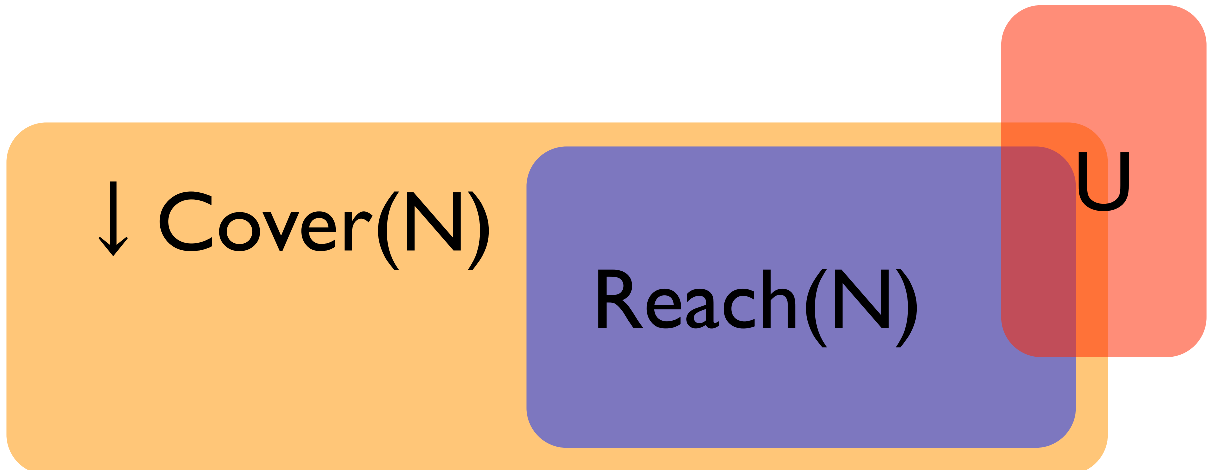

- There is m in ↓ Cover(N) ∩ U
- Hence, there is m' $\succcurlyeq$ m which is in Reach(N)
- However, any m' $\succcurlyeq$ m is also in U
- Thus, there is m' both in Reach(N) and U

# What if ?

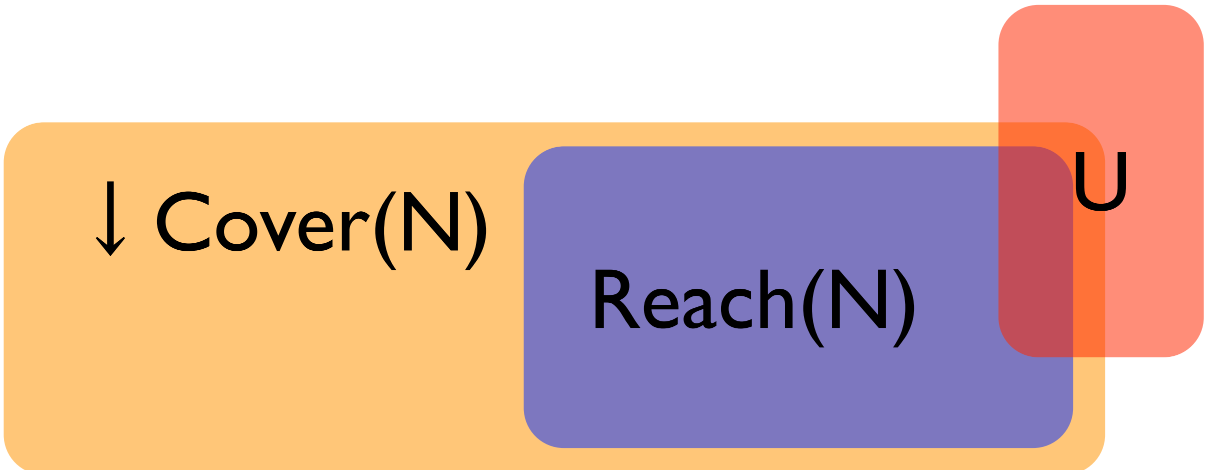



- There is m in ↓ Cover(N) ∩ U
- Hence, there is m' $\succcurlyeq$ m which is in Reach(N)
- However, any m' $\succcurlyeq$ m is also in U
- Thus, there is m' both in Reach(N) and U

# What if ?



# What if ?




$$\text{Reach}(N) \cap U = \emptyset$$

implies

$$\downarrow \text{Cover}(N) \cap U = \emptyset$$

# Coverability set and coverability problem

# Coverability set and coverability problem

- Theorem:

  $\mathrm{Reach}(N) \cap U = \phi$ iff $\downarrow \mathrm{Cover}(N) \cap U = \phi$

# Coverability set and coverability problem

- Theorem:

  $Reach(N) \cap U = \phi$ iff $\downarrow Cover(N) \cap U = \phi$

- Nice,...

# Coverability set and coverability problem

- Theorem:

  $Reach(N) \cap U = \phi$ iff $\downarrow Cover(N) \cap U = \phi$

- Nice,...
- ...but $U$ and $\downarrow Cover(N)$ might both be infinite !

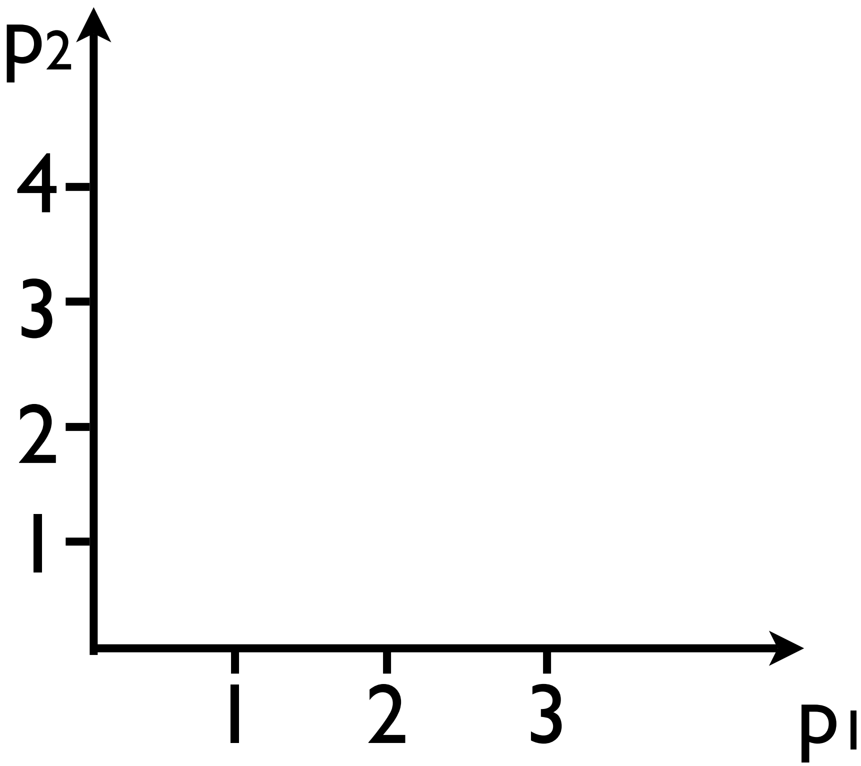
# Coverability set and coverability problem

- Theorem:

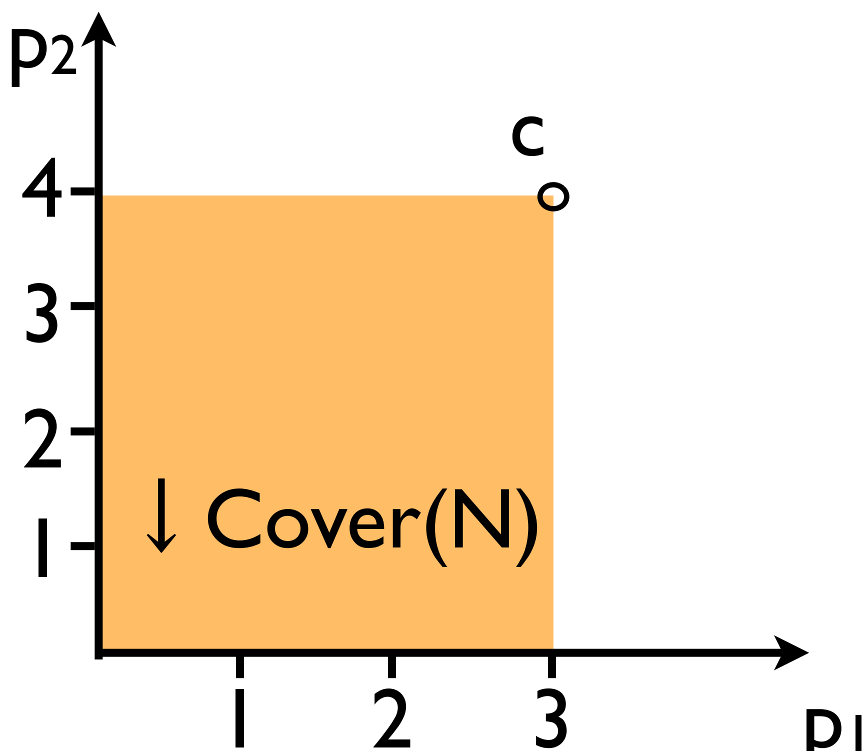
  $\mathrm{Reach}(N) \cap U = \emptyset$ iff $\downarrow \mathrm{Cover}(N) \cap U = \emptyset$

- Nice,...
- ...but $U$ and $\downarrow \mathrm{Cover}(N)$ might both be infinite !
- How do we test that $\downarrow \mathrm{Cover}(N) \cap U = \emptyset$??

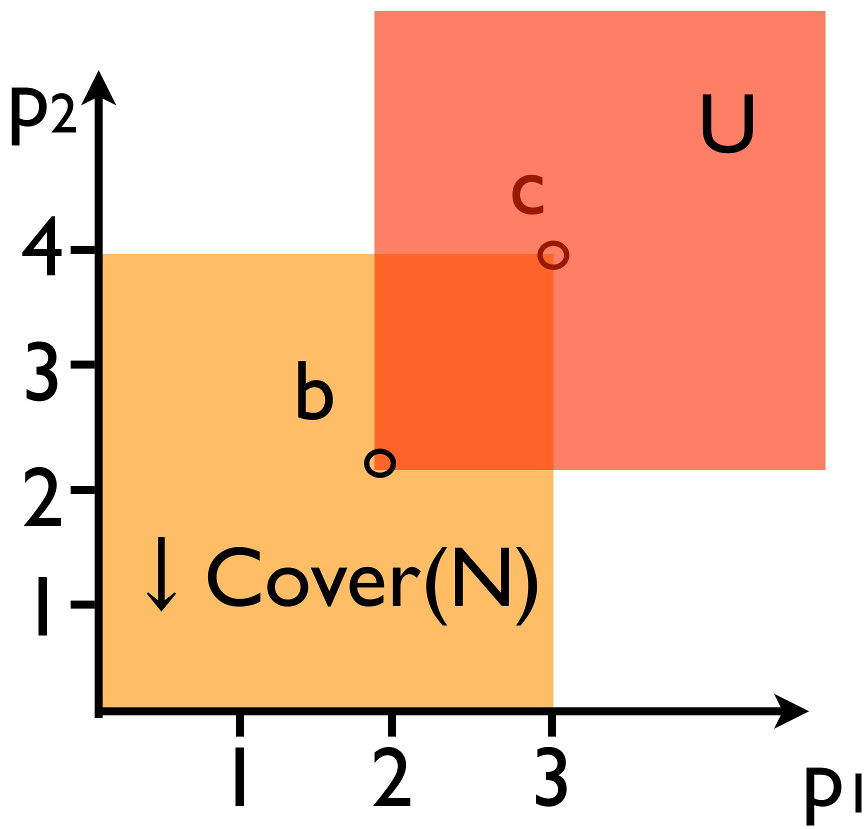
# Coverability set and coverability problem



# Coverability set and coverability problem

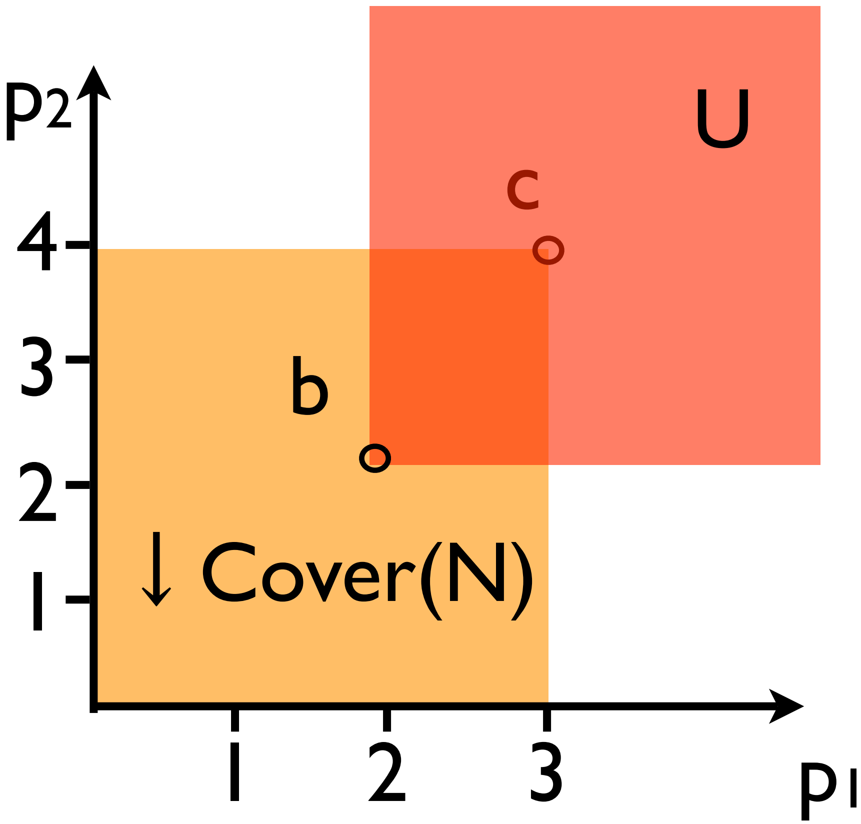


# Coverability set and coverability problem

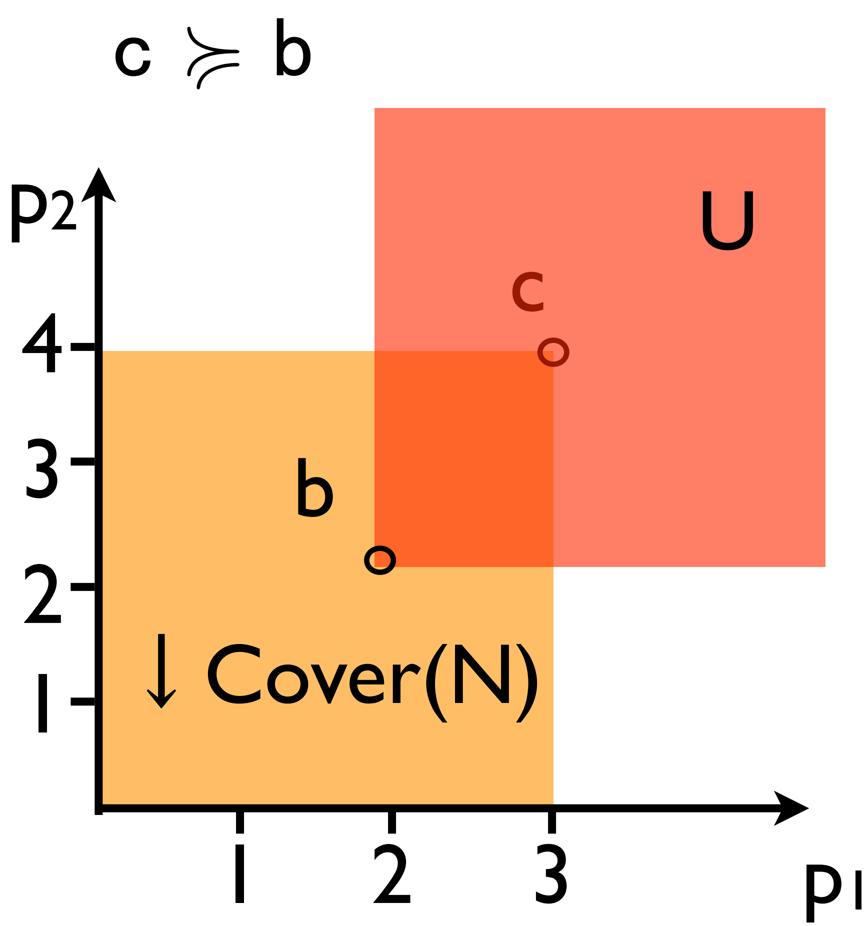


# Coverability set and coverability problem

$c \not\succcurlyeq b$

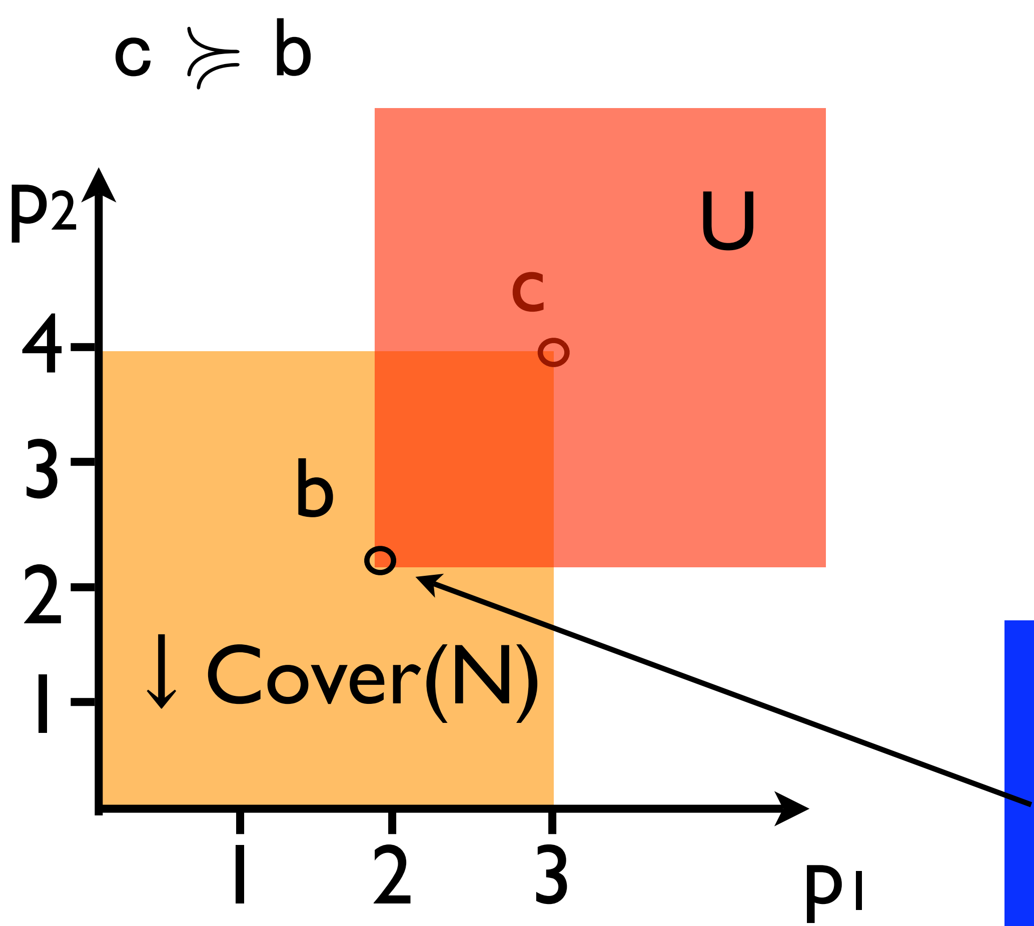


# Coverability set and coverability problem




All we need to remember is the (finite) set of minimal elements Min(U)

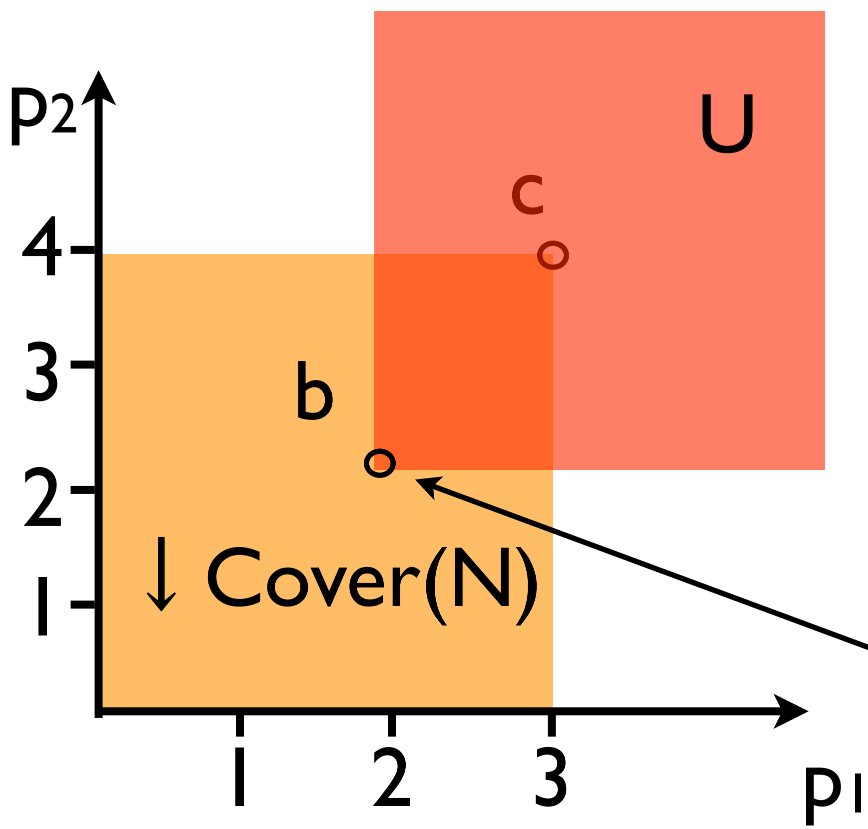
# Coverability set and coverability problem




All we need to remember is the (finite) set of minimal elements Min(U)

# Coverability set and coverability problem

$c \succcurlyeq b$



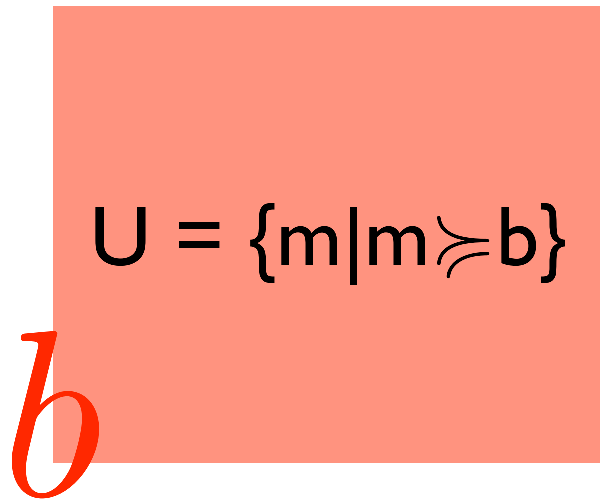

$\downarrow$ Cover(N) $\cap$ U $\neq \phi$

iff

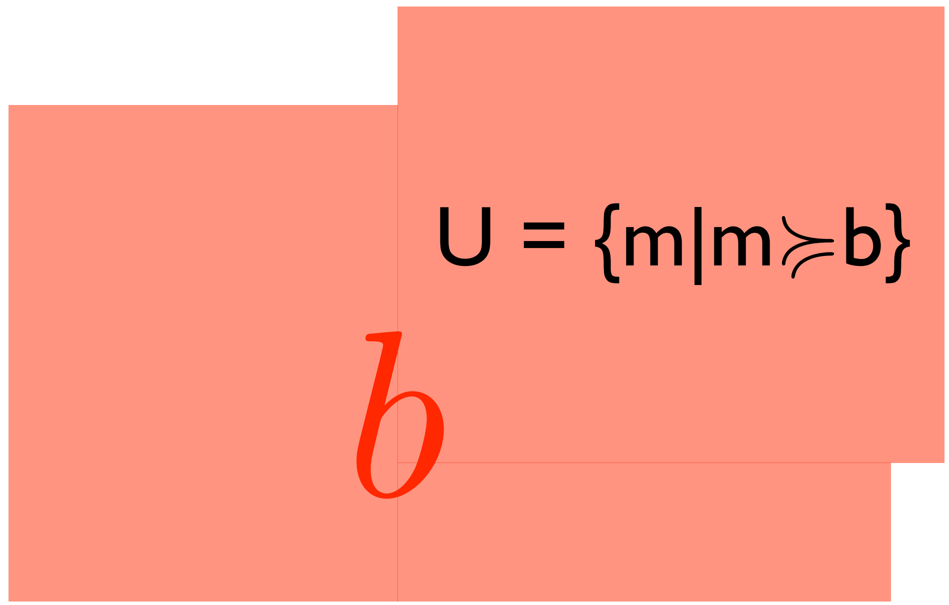
there is c in Cover(N) and b in Min(U) s.t. $c \succcurlyeq b$

All we need to remember is the (finite) set of minimal elements Min(U)

# Backward approach

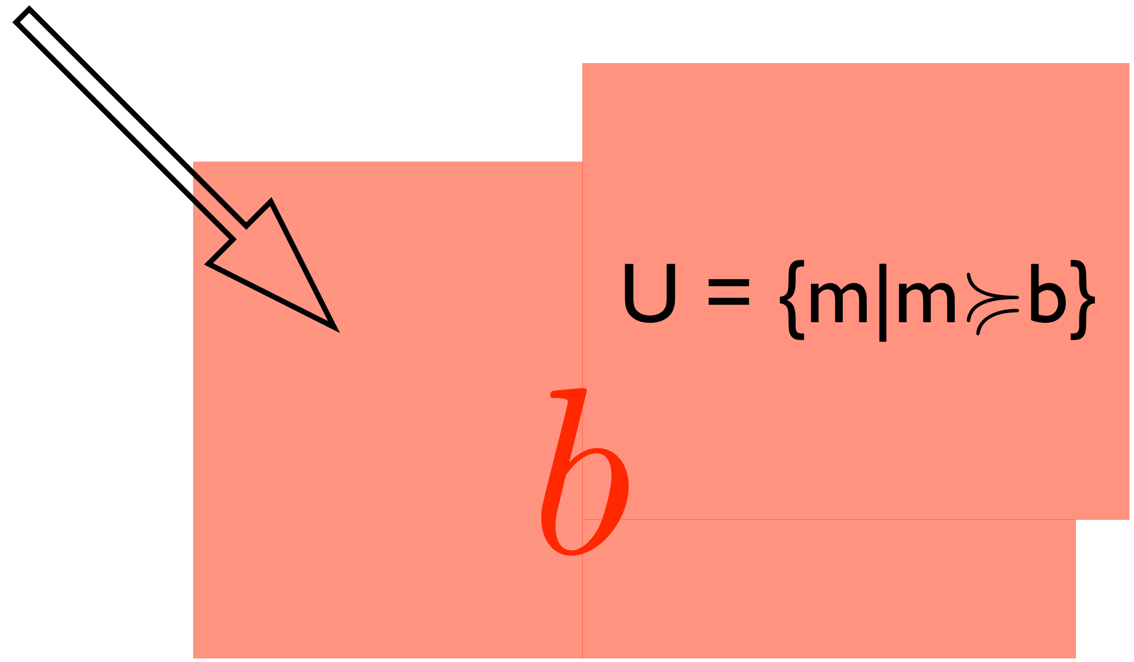


# Backward approach

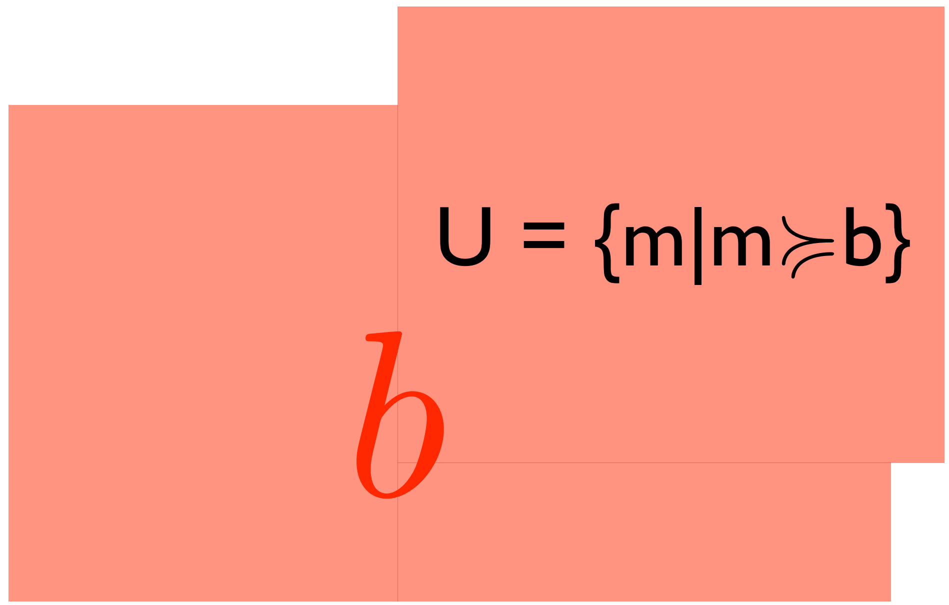


# Backward approach

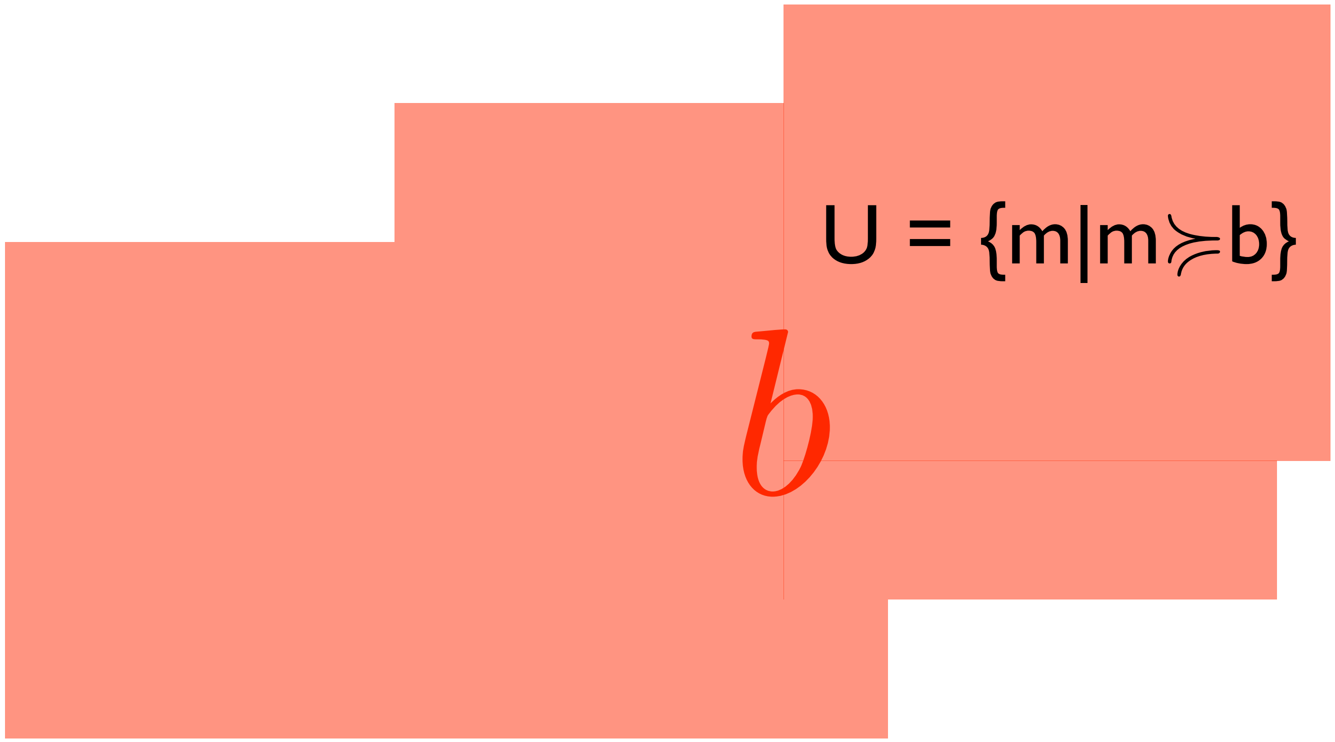
All the markings that can reach U in one step



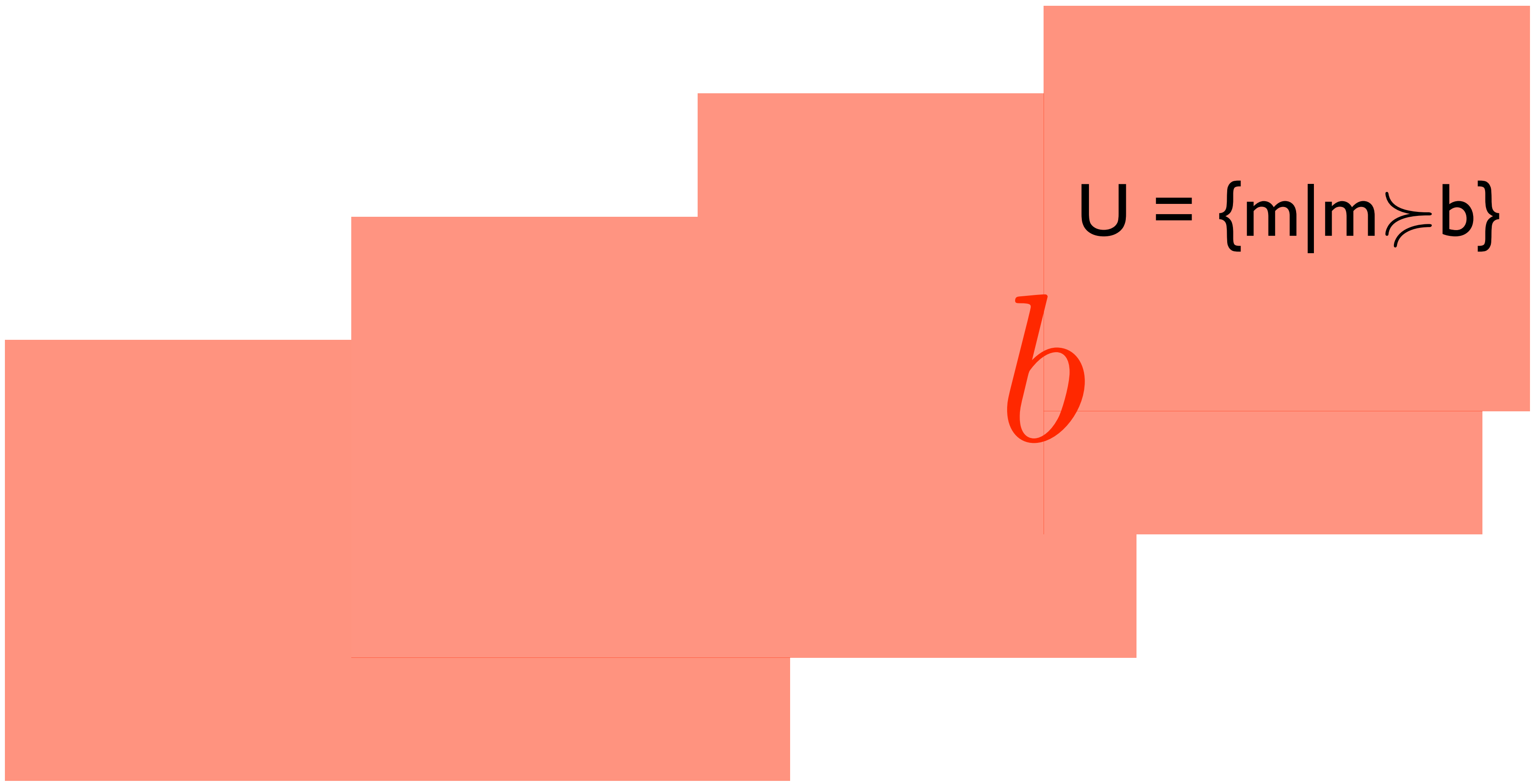
# Backward approach



# Backward approach

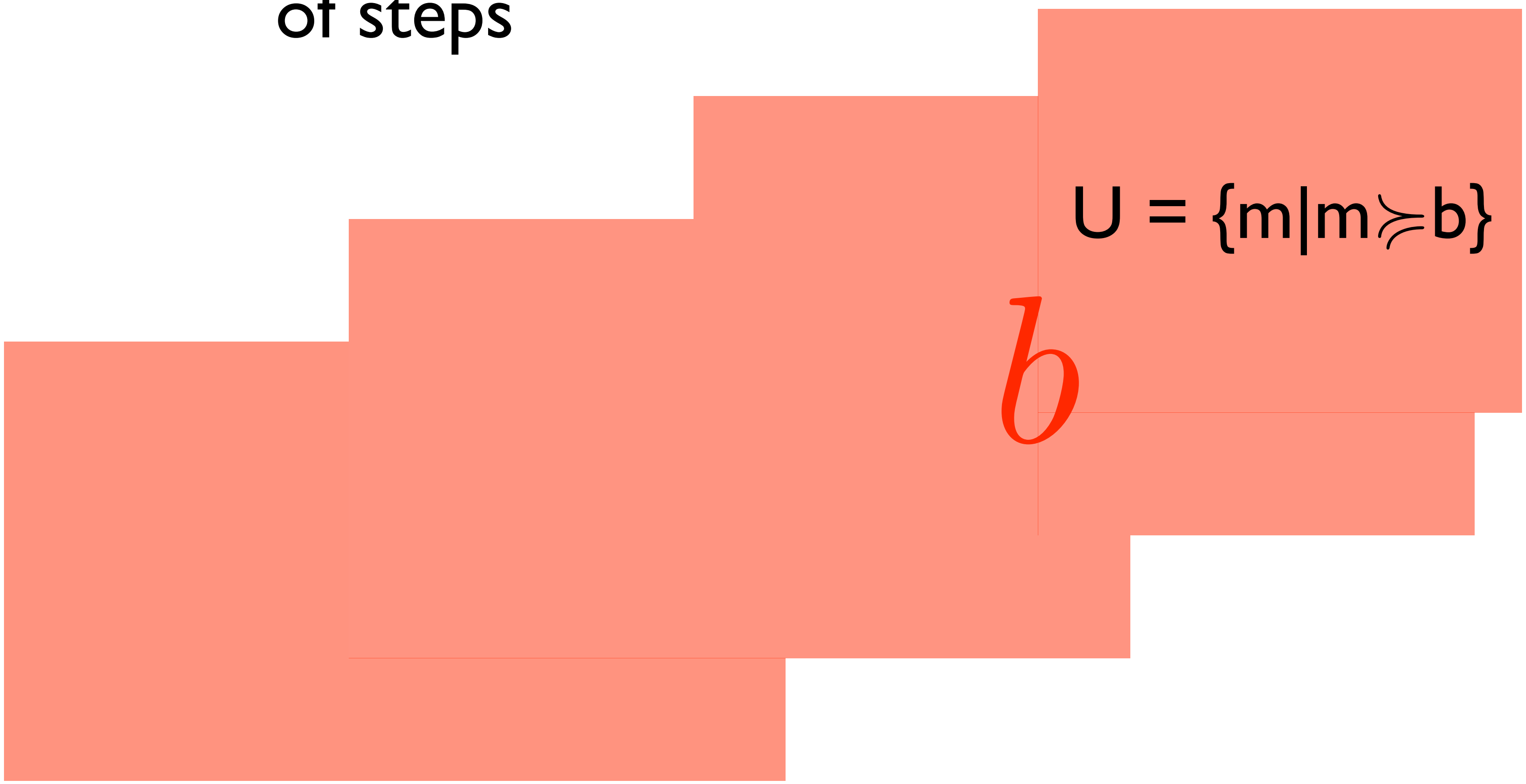


# Backward approach



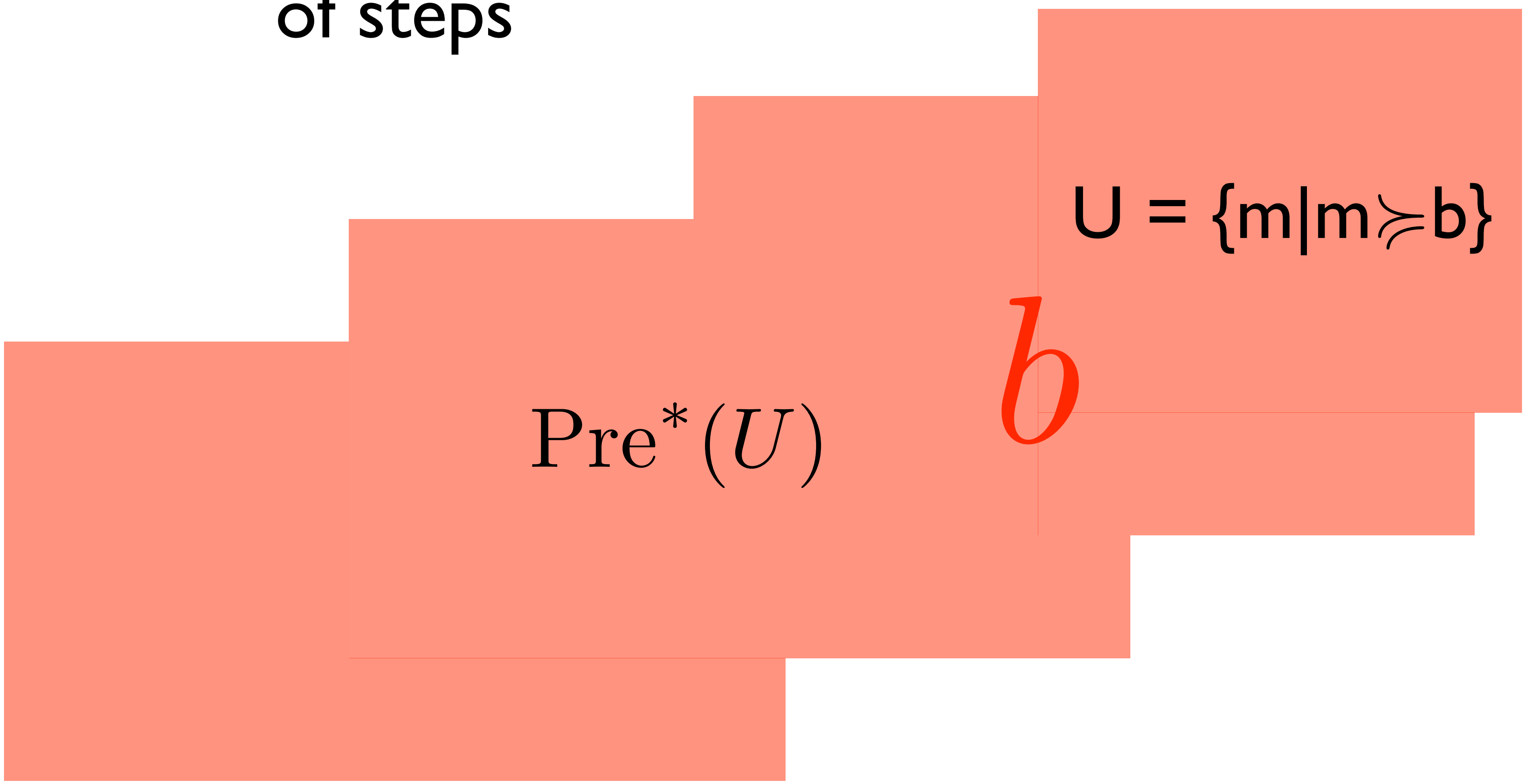
# Backward approach

In the end, we want to obtain all the markings that can reach U in any number of steps



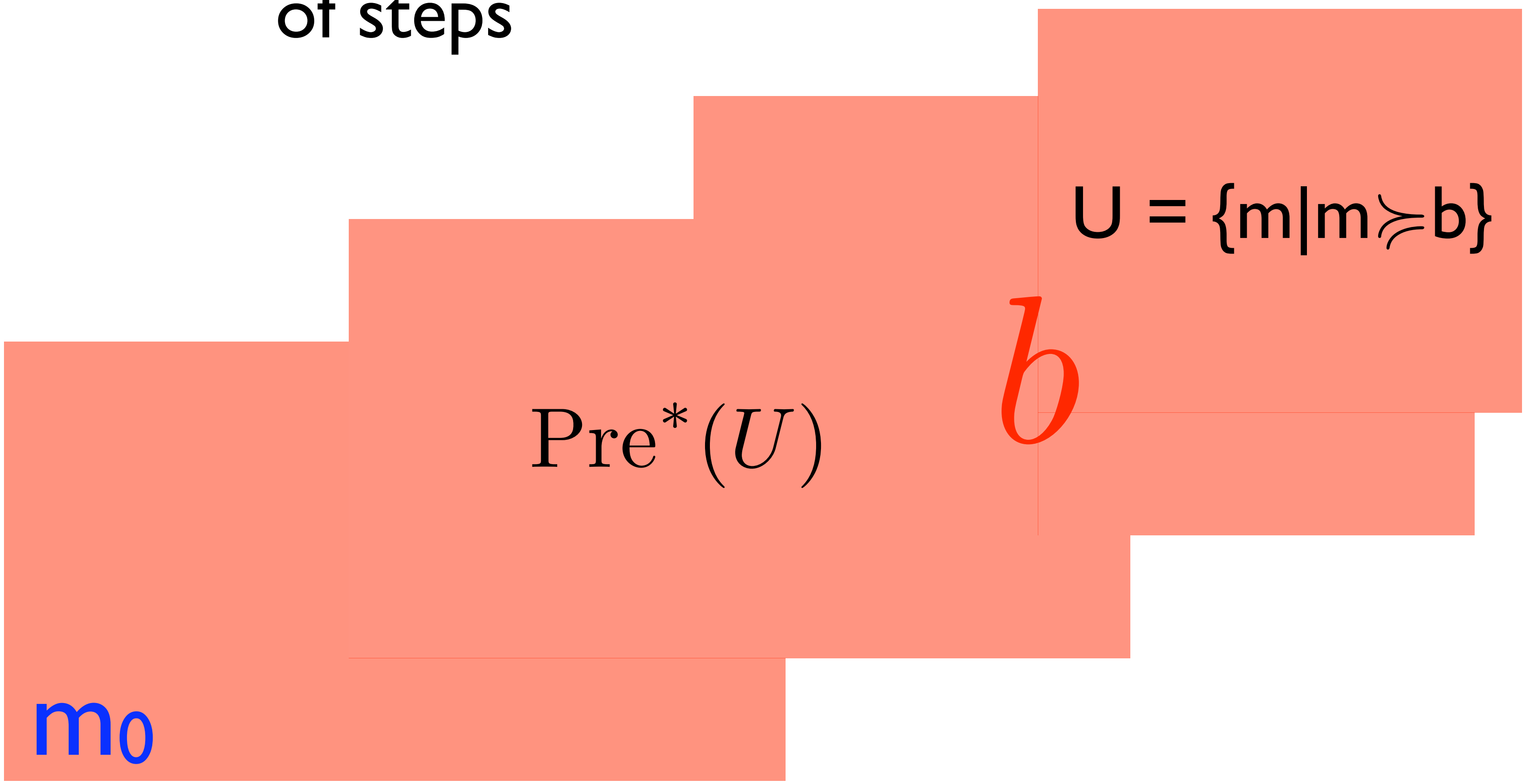
# Backward approach

In the end, we want to obtain all the markings that can reach U in any number of steps



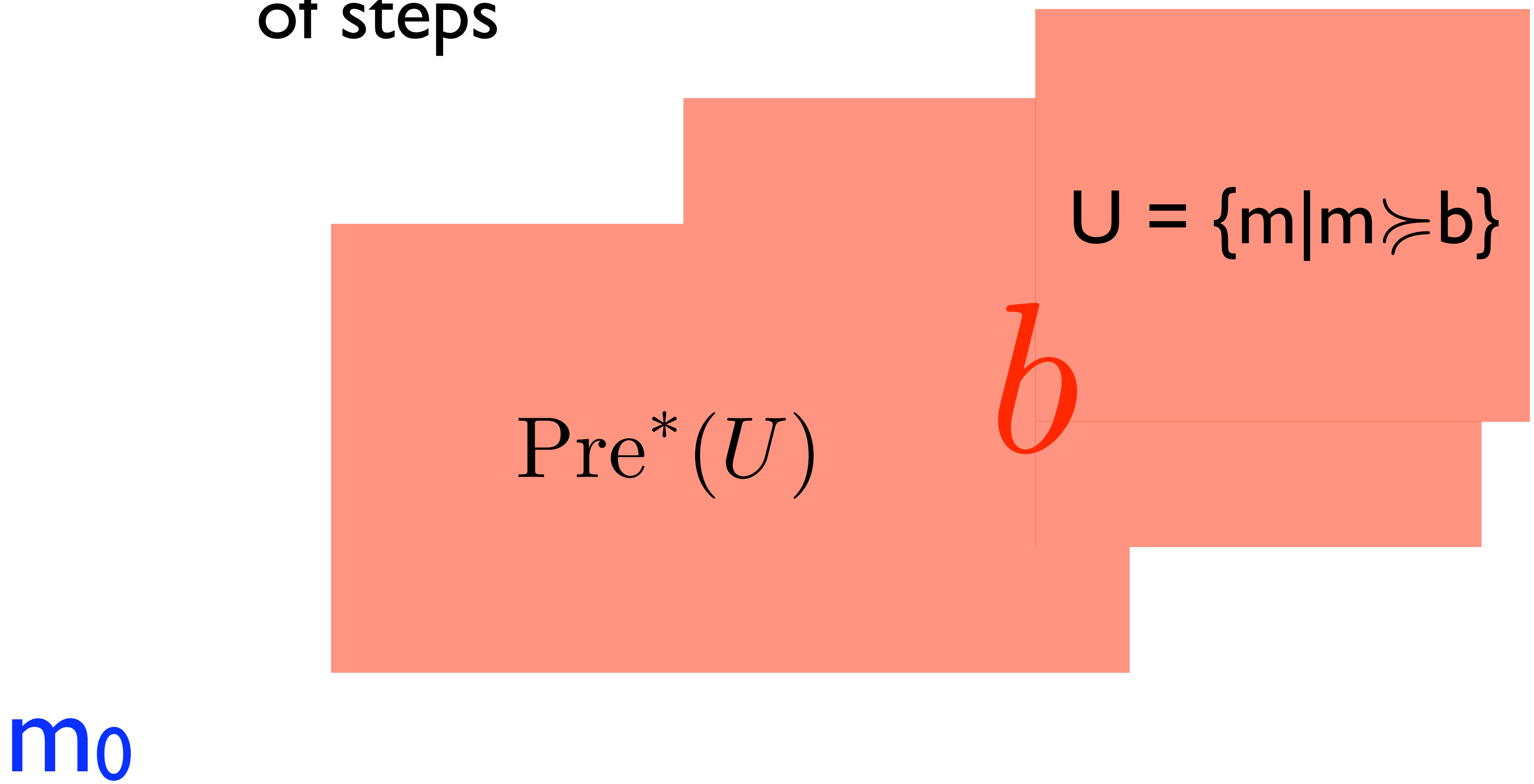
# Backward approach

In the end, we want to obtain all the markings that can reach U in any number of steps



# Backward approach

In the end, we want to obtain all the markings that can reach U in any number of steps



# Backward Approach

- Clearly:

  $m_0$ is in $Pre^*(U)$ iff $Reach(N) \cap U \neq \phi$

- Question: can we compute $Pre^*(U)$ ?
  - Yes !

# Predecessor operator

- Symmetrically to the Post, we define the predecessor operator:

$$\mathrm{Pre}(m) = \{m' \mid m \text{ is in } \mathrm{Post}(m')\}$$

- Let us consider the sequence

  U, Pre(U), Pre(Pre(U)), Pre(Pre(Pre(U))),...

- Theorem: After a finite amount of steps, the sequence stabilises, and we obtain $\mathrm{Pre}^*(U)$


# Advertisement

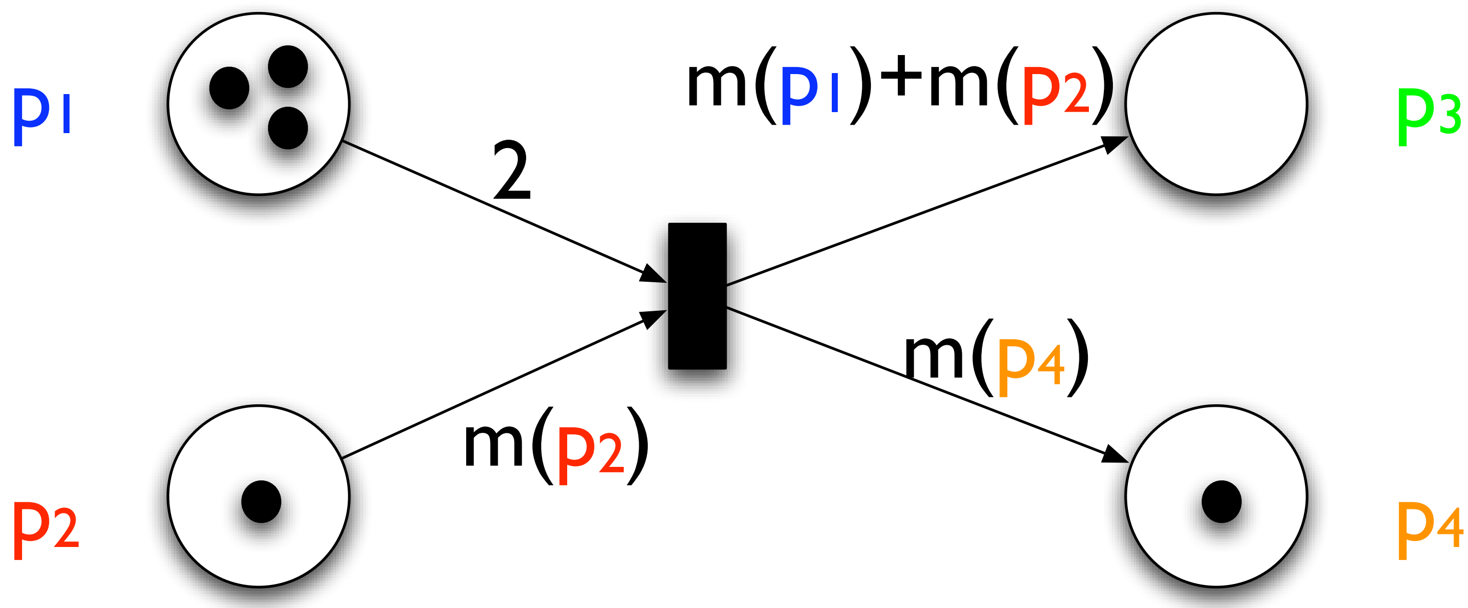

- Efficient datastuctures to implement this algorithm have been defined by researchers of the verification group at ULB.

# More on Petri nets

# Marking dependent effects

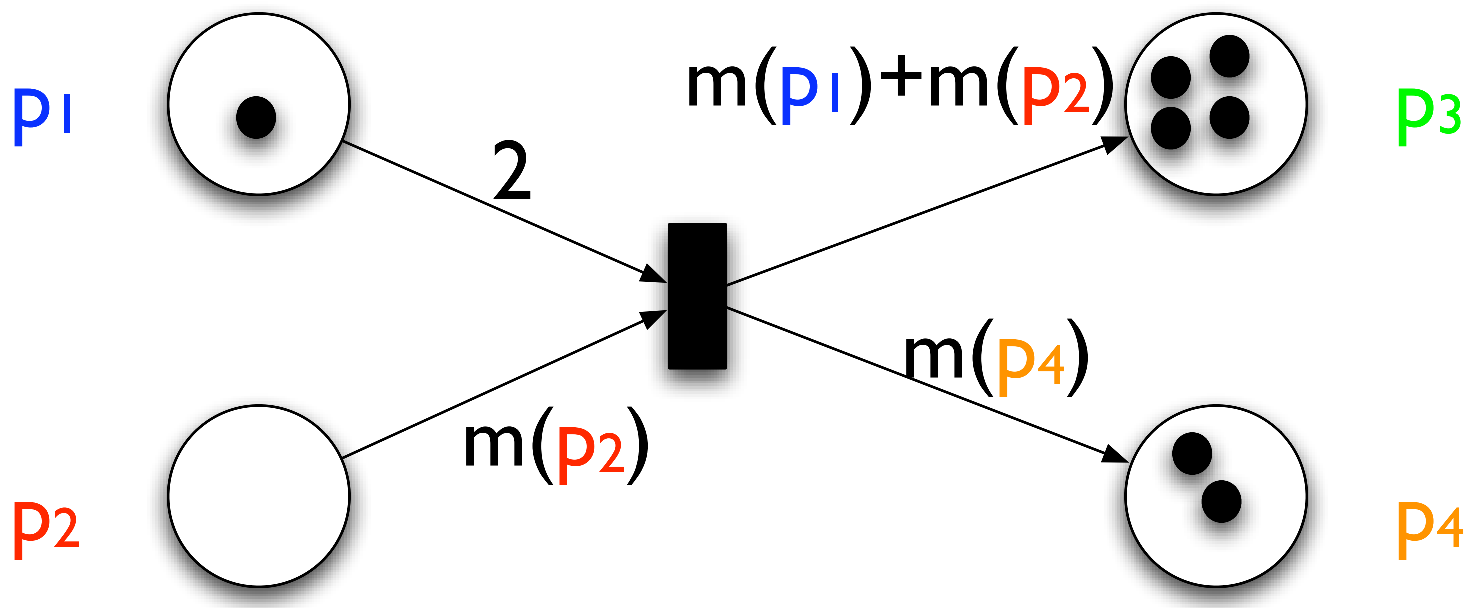
# Marking-dependent effect

- The effect of a transition is not constant anymore, but depends on the current marking.



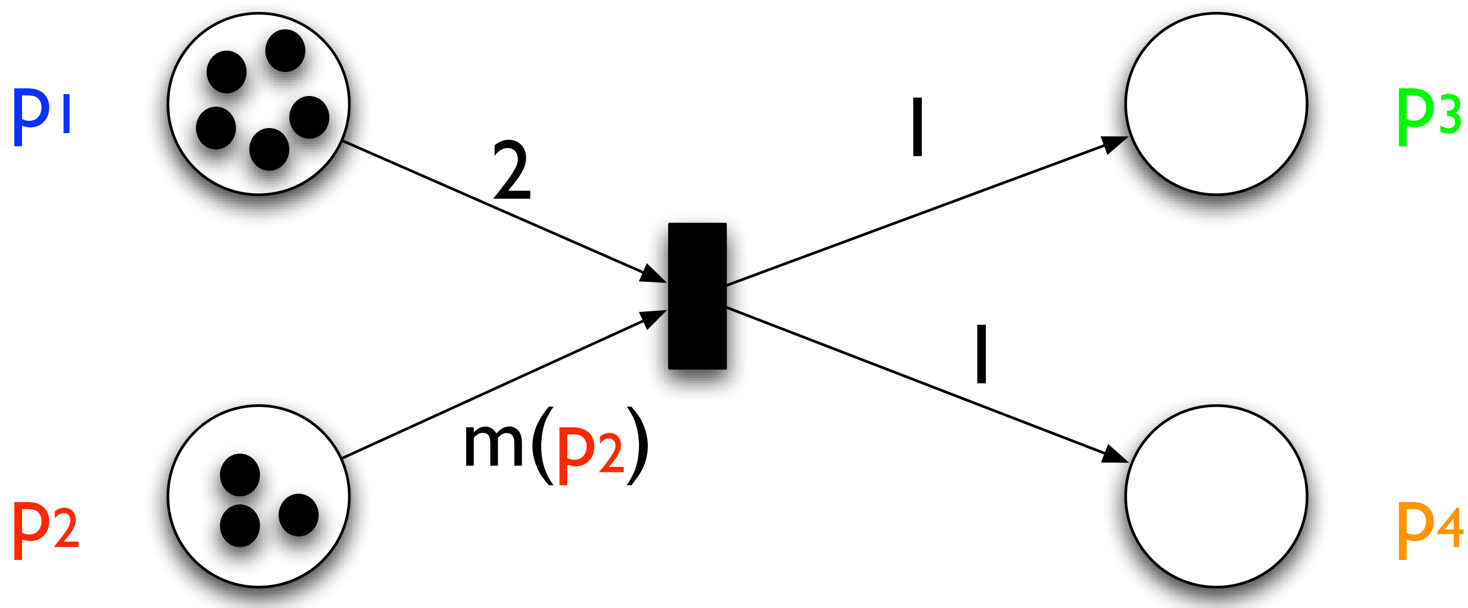
# Marking-dependent effect

- The effect of a transition is not constant anymore, but depends on the current marking.



# Marking-dependent effect - resets

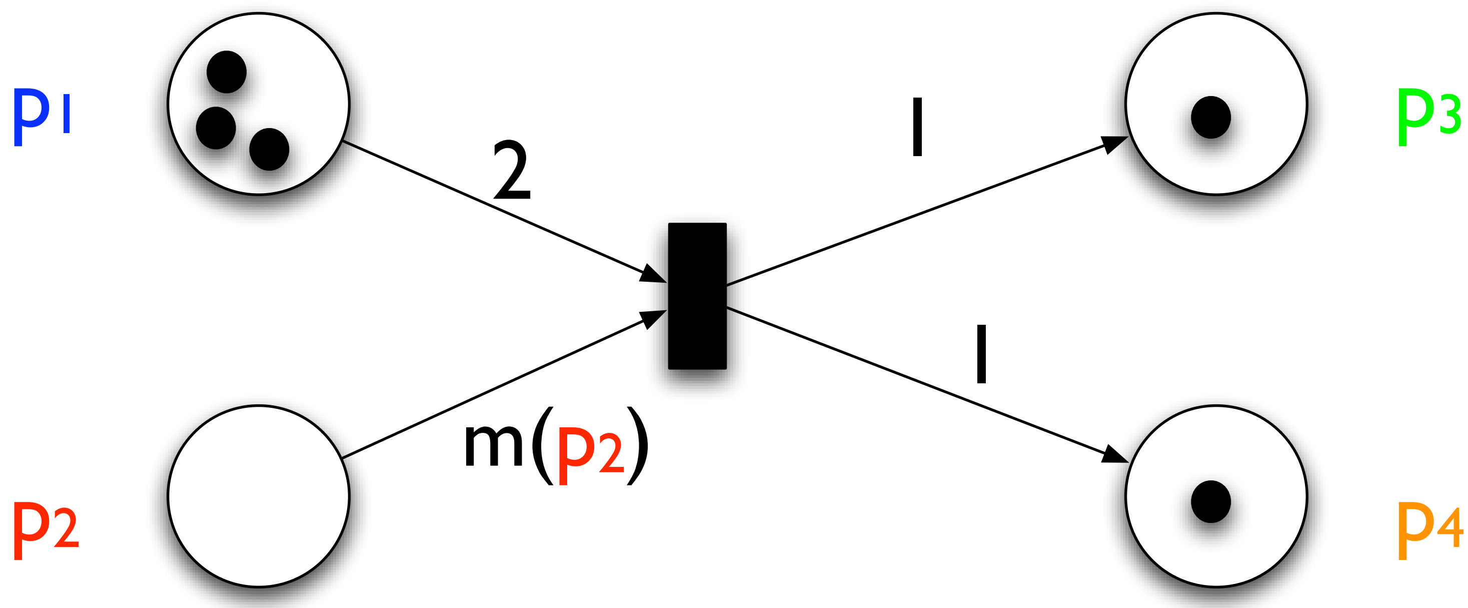
- In particular, we can define resets.




reset of p2

# Marking-dependent effect - resets

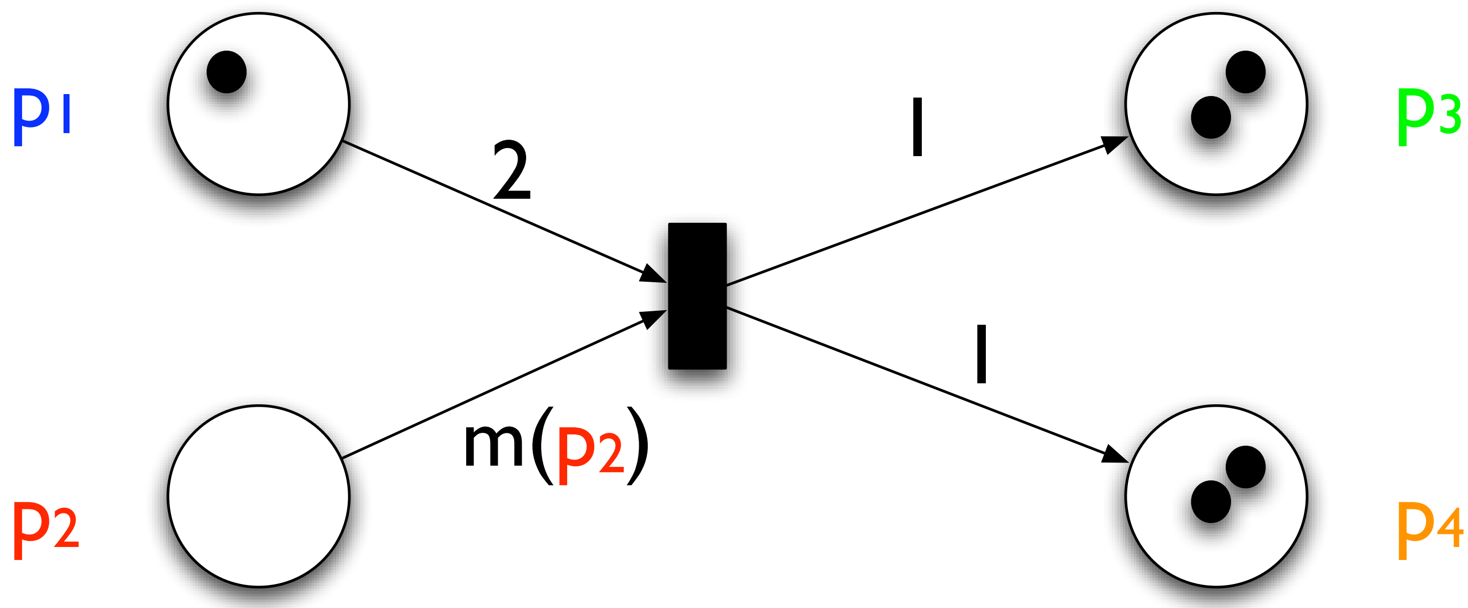
- In particular, we can define resets.




reset of p2

# Marking-dependent effect - resets

- In particular, we can define resets.



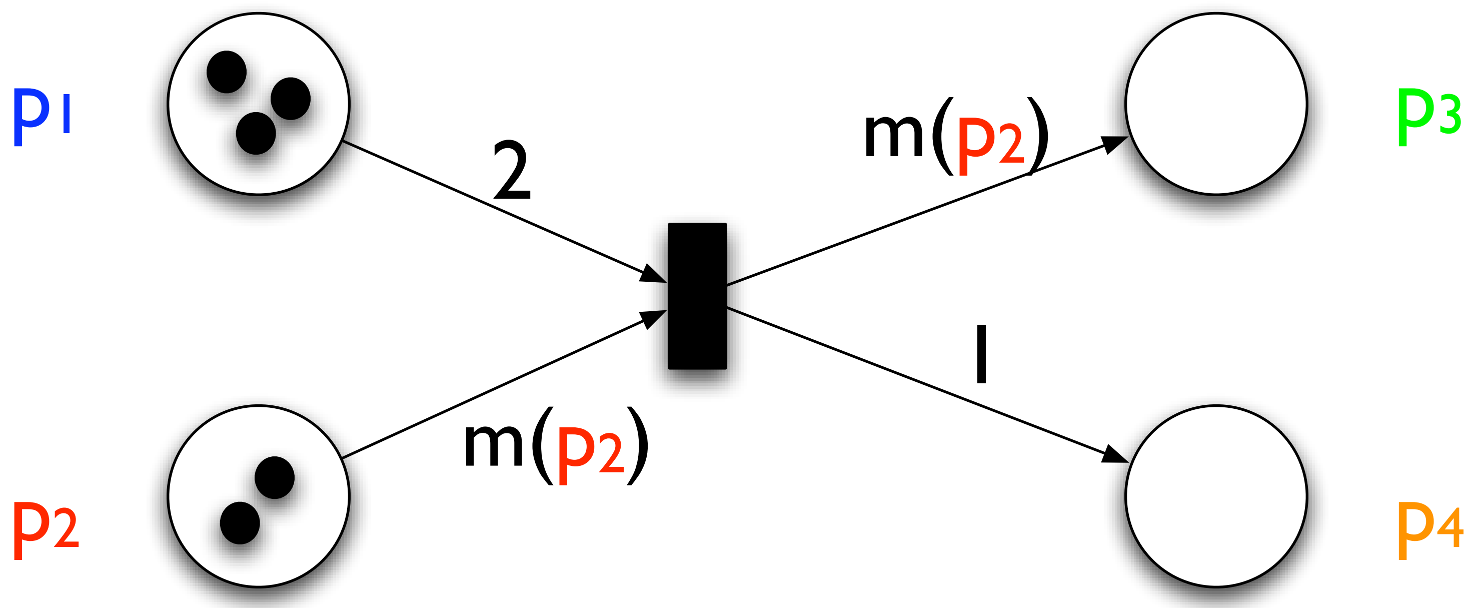

reset of p2

# Reset nets

- When we have only classical PN transitions + resets:
  - Coverability is decidable
  - Boundedness is decidable
  - Place boundedness is undecidable
  - The coverability set is not computable

# Marking-dependent effect - transfers

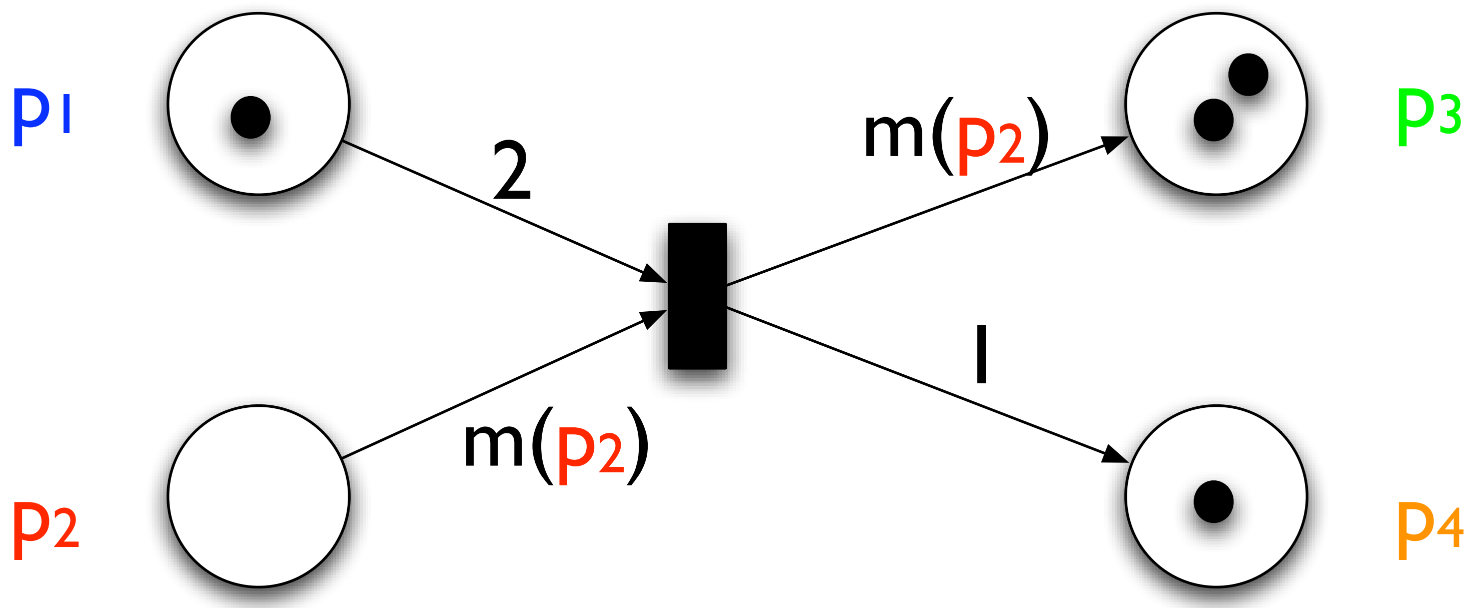
- In particular, we can define transfers.




transfer from $p_2$ to $p_3$

# Marking-dependent effect - transfers

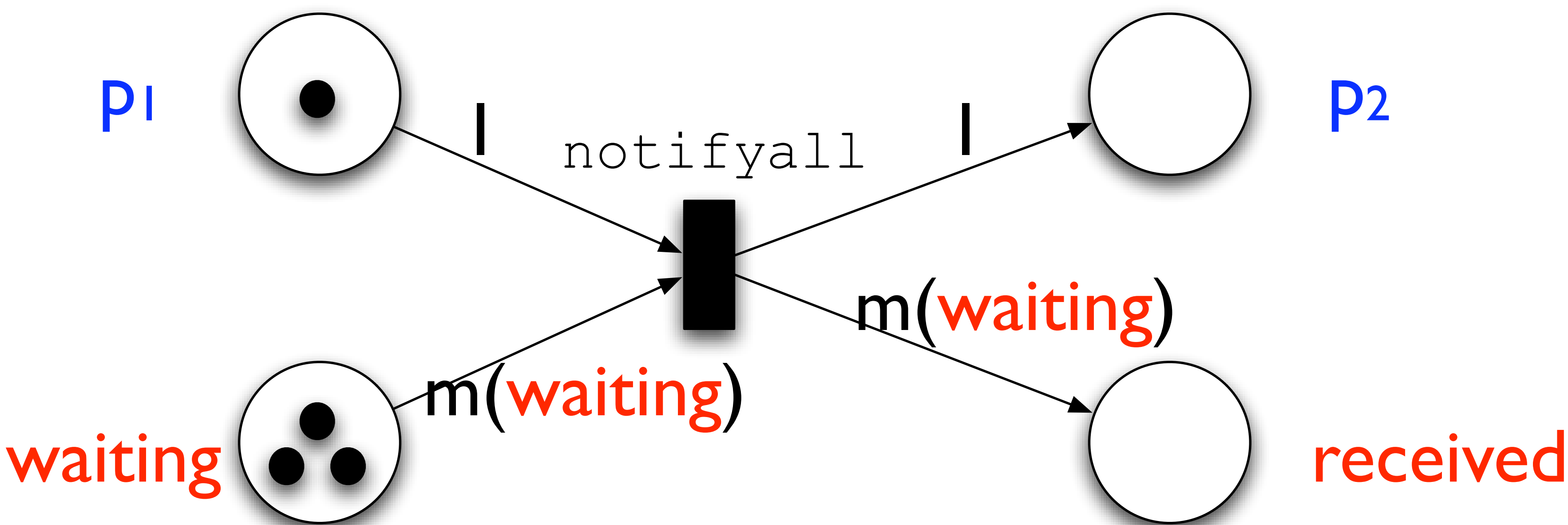
- In particular, we can define transfers.




transfer from p2 to p3

# Usefulness of transfers

- Modelisation of broadcasts :
  - A single message is sent to every process
  - Each process that receives the message moves to another state.

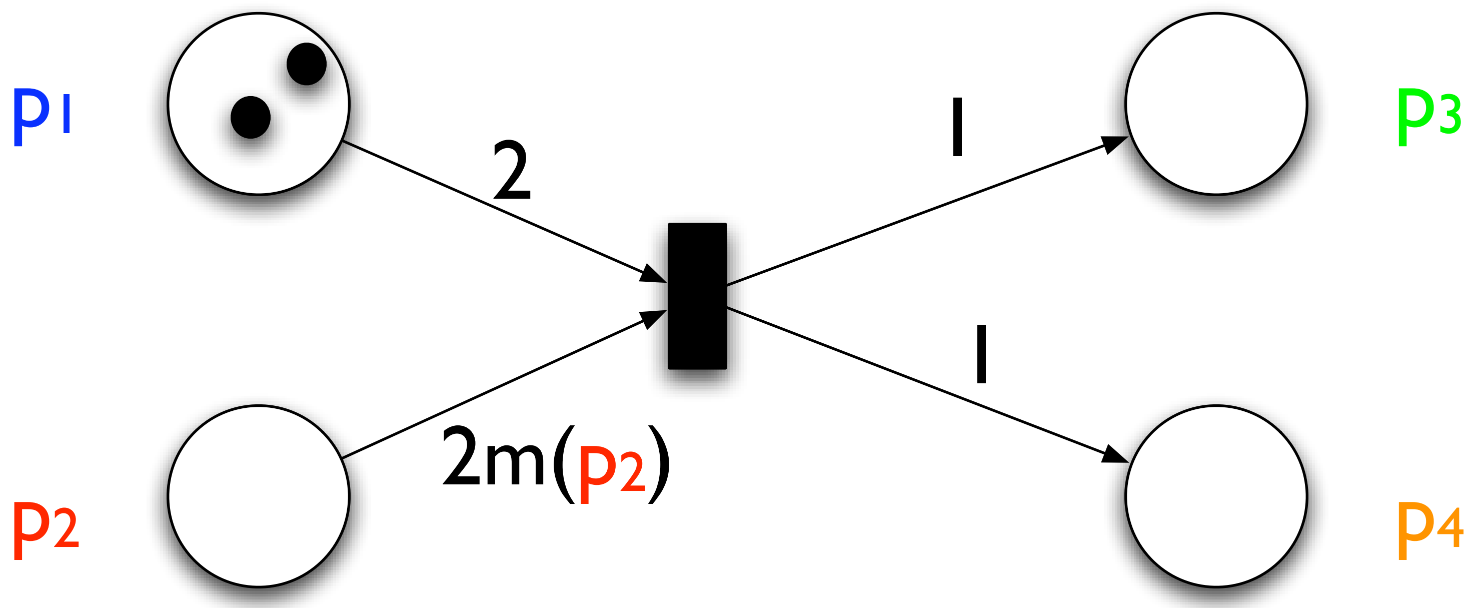


# Transfer nets

- When we have only classical PN transitions + transfers:
  - Coverability is decidable
  - Boundedness is decidable
  - Place boundedness is undecidable
  - The coverability set is not computable

# Marking-dependent effect - zero-test

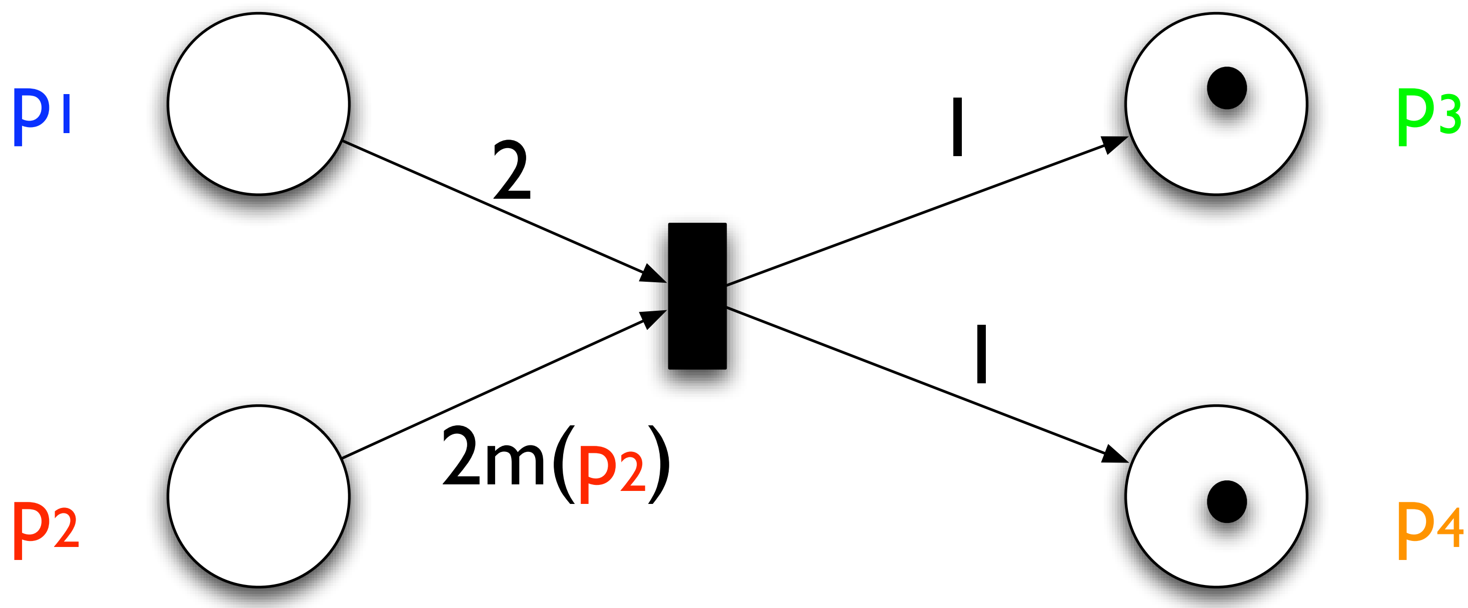
- In particular, we can define test for zero.




enabled only if p2 is empty

# Marking-dependent effect - zero-test

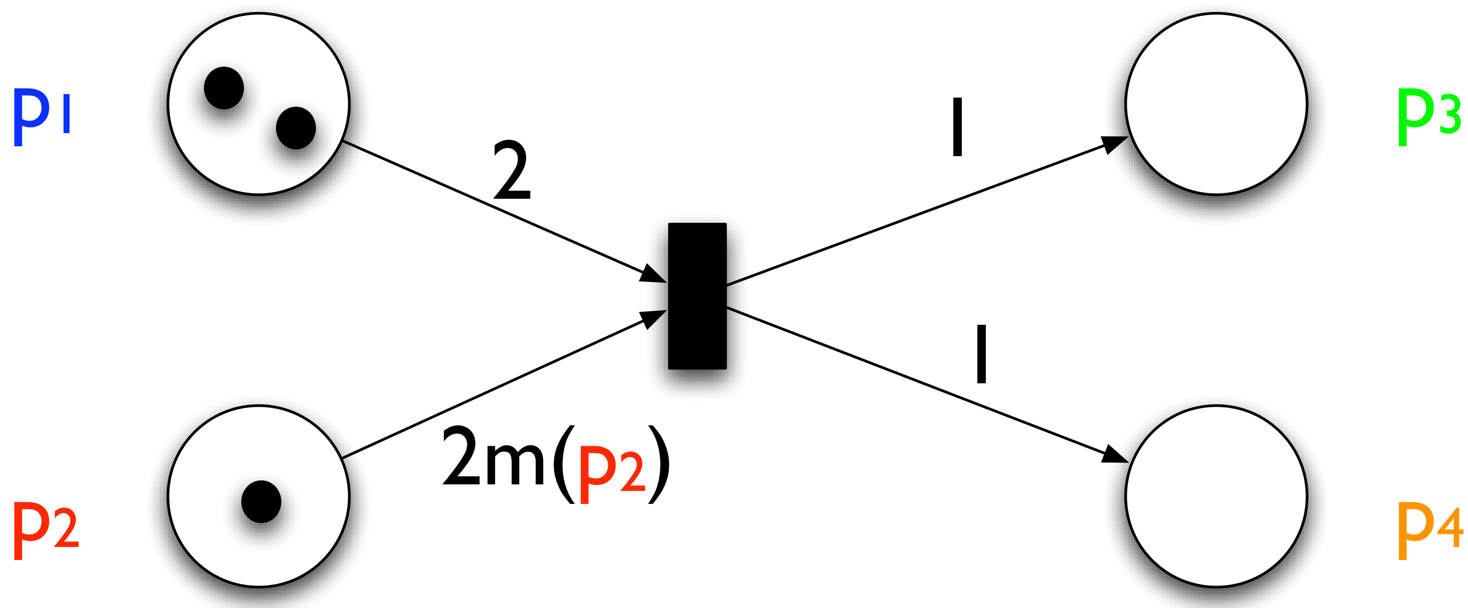
- In particular, we can define test for zero.




enabled only if $p_2$ is empty

# Marking-dependent effect - zero-test

- In particular, we can define test for zero.




enabled only if p2 is empty

# Test for zero

- Once we have test-for-zero everything becomes undecidable.

# Coloured Petri nets

# Coloured Petri nets

- Popular extension of the basic model.
  - Introduced by the team of Kurt Jensen, in the ‘80s
  - used in many applications

# Coloured Petri nets

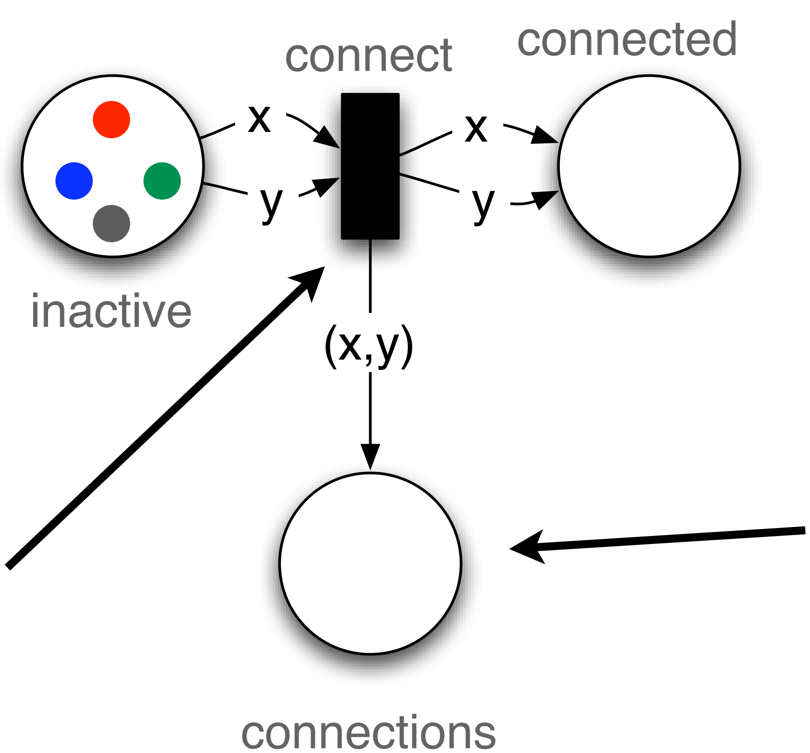
- Idea: add colours to the tokens
  - Allow to distinguish between different types of tokens
  - The colours can model data carried by the processes
  - Transitions are aware of the colours

# Phone example

- We have a set of customers:
  - Each customer is represented by a token.
  - Color of the token = Phone number.
  - A customer is either inactive or connected.

# Phone example

- A pair of inactive customers can establish a connection.
  - We want to distinguish between sender and receiver.

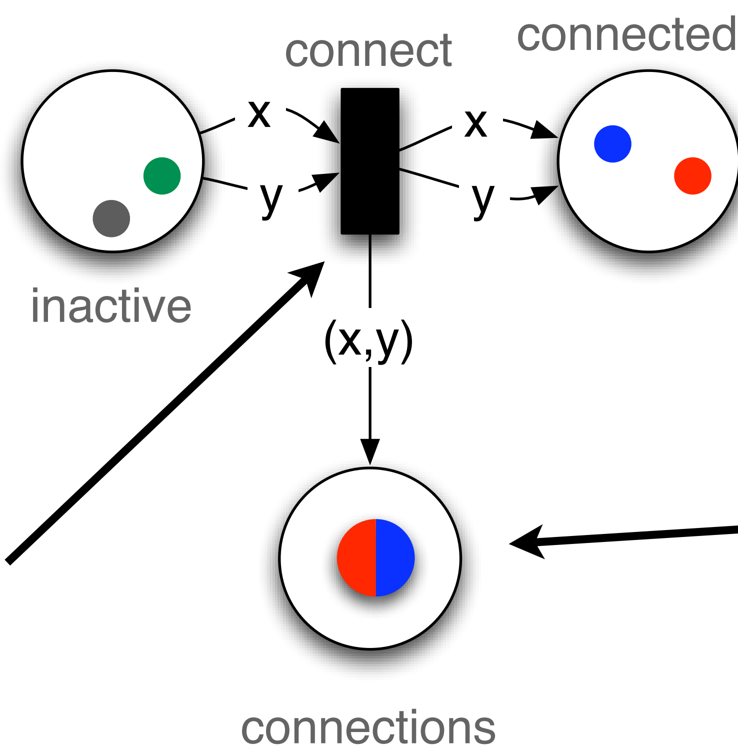



The transition consumes a sender x and a receiver y

Connections are recorded here as tokens whose color is a pair (snd, rcv)

# Phone example

- A pair of inactive customers can establish a connection.
  - We want to distinguish between sender and receiver.

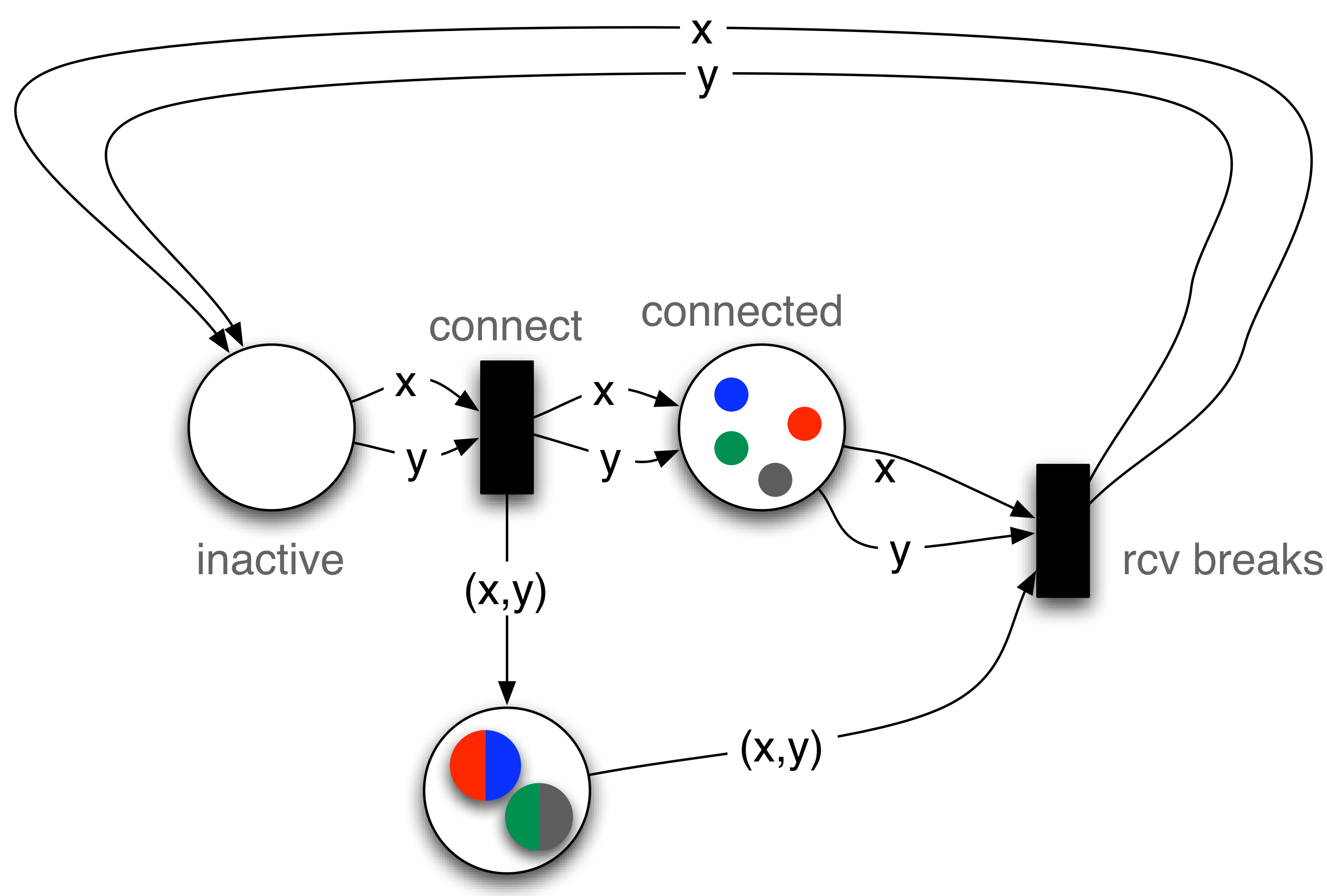



The transition consumes
a sender x and a
receiver y

Connections are
recorded here as
tokens whose color
is a pair (snd, rcv)

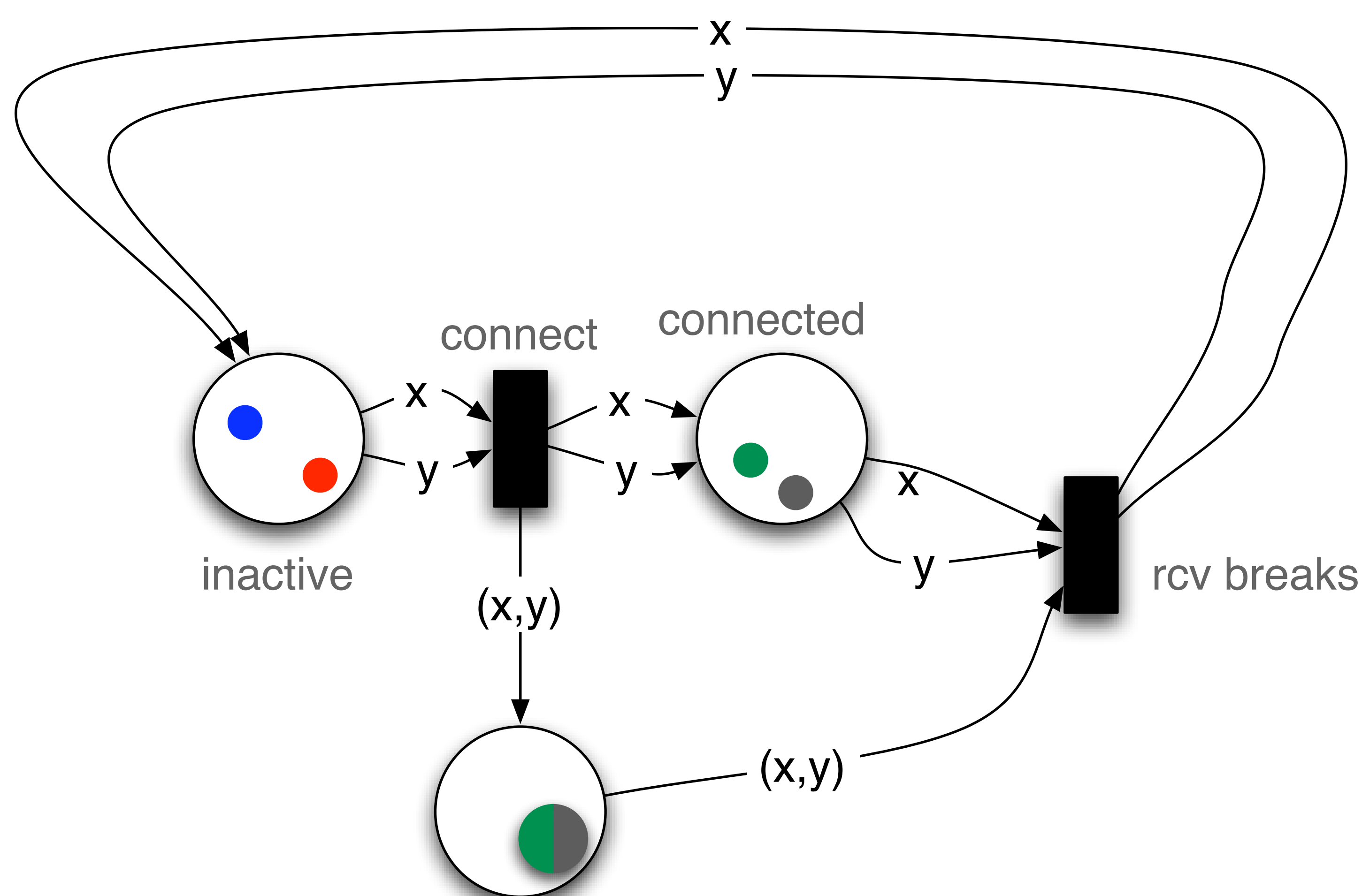
# Phone example

- The connection can be closed either by the sender or by the receiver.

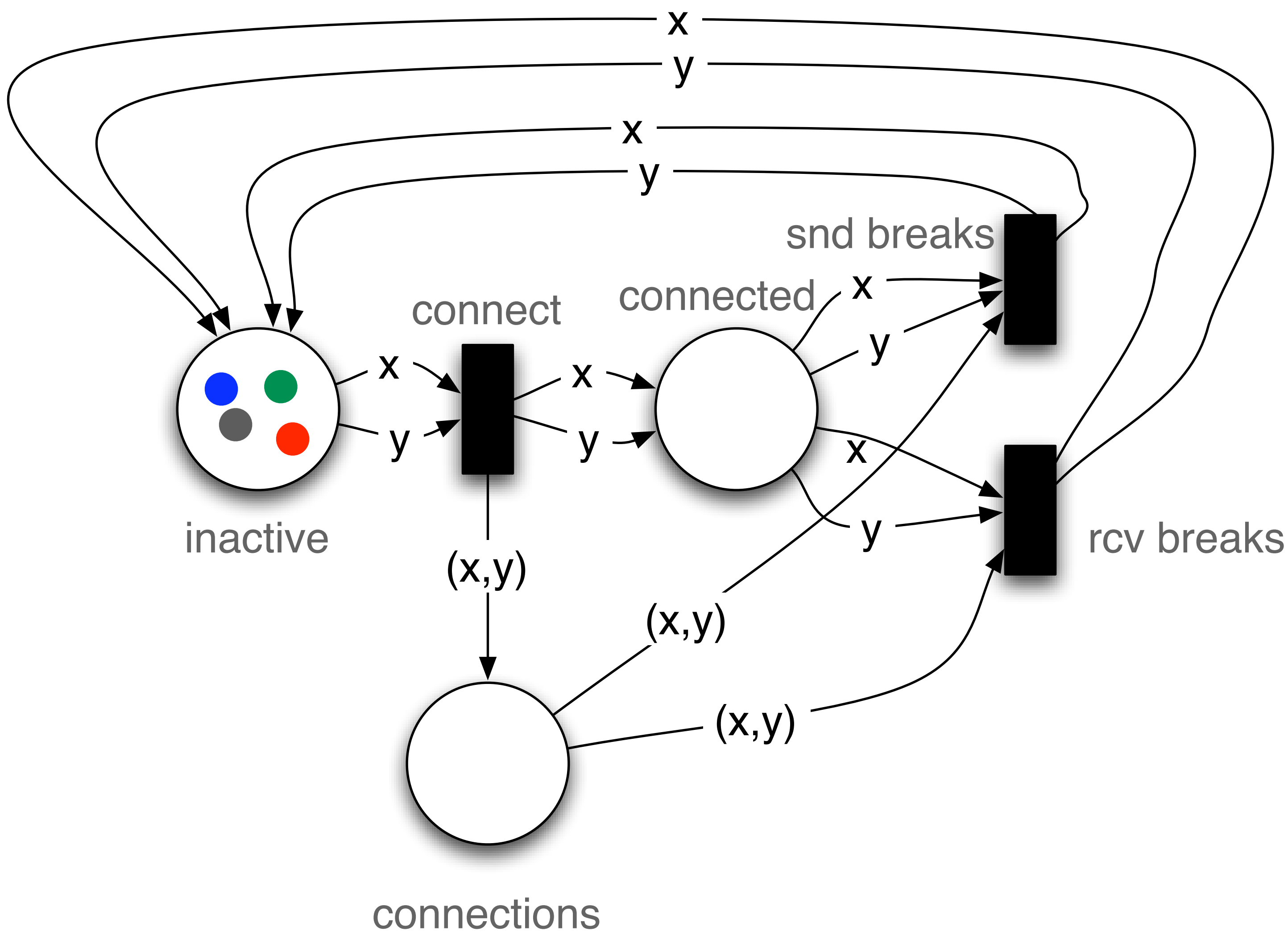


# Phone example

- The connection can be closed either by the sender or by the receiver.



# Phone example

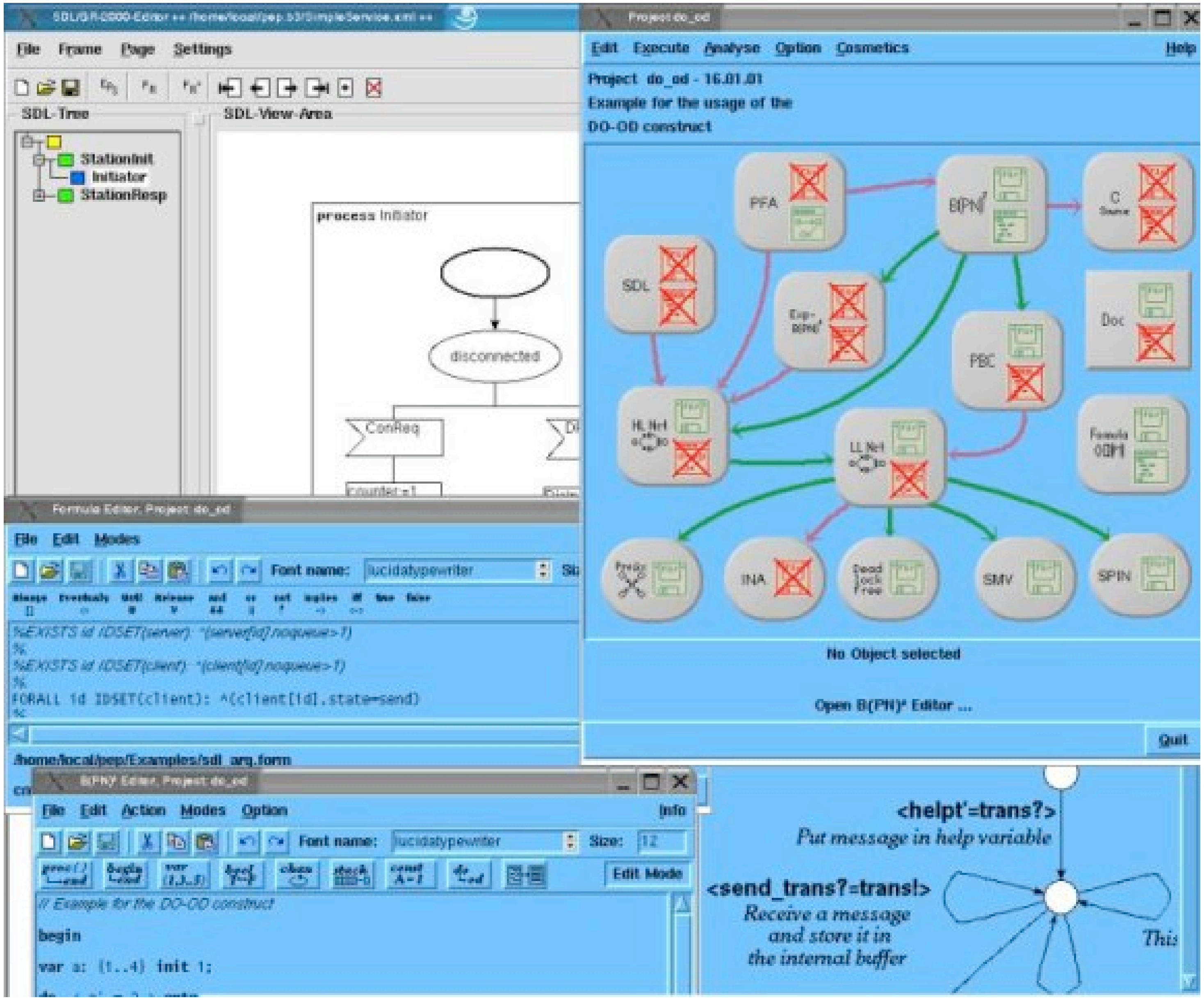


# Coloured Petri nets

- Several analysis methods have been developped for this model (finite number of colours)
    - e.g.: invariants
- Some results can be achieved when the colors have good properties

# Tools

# Practical Tools: Pep

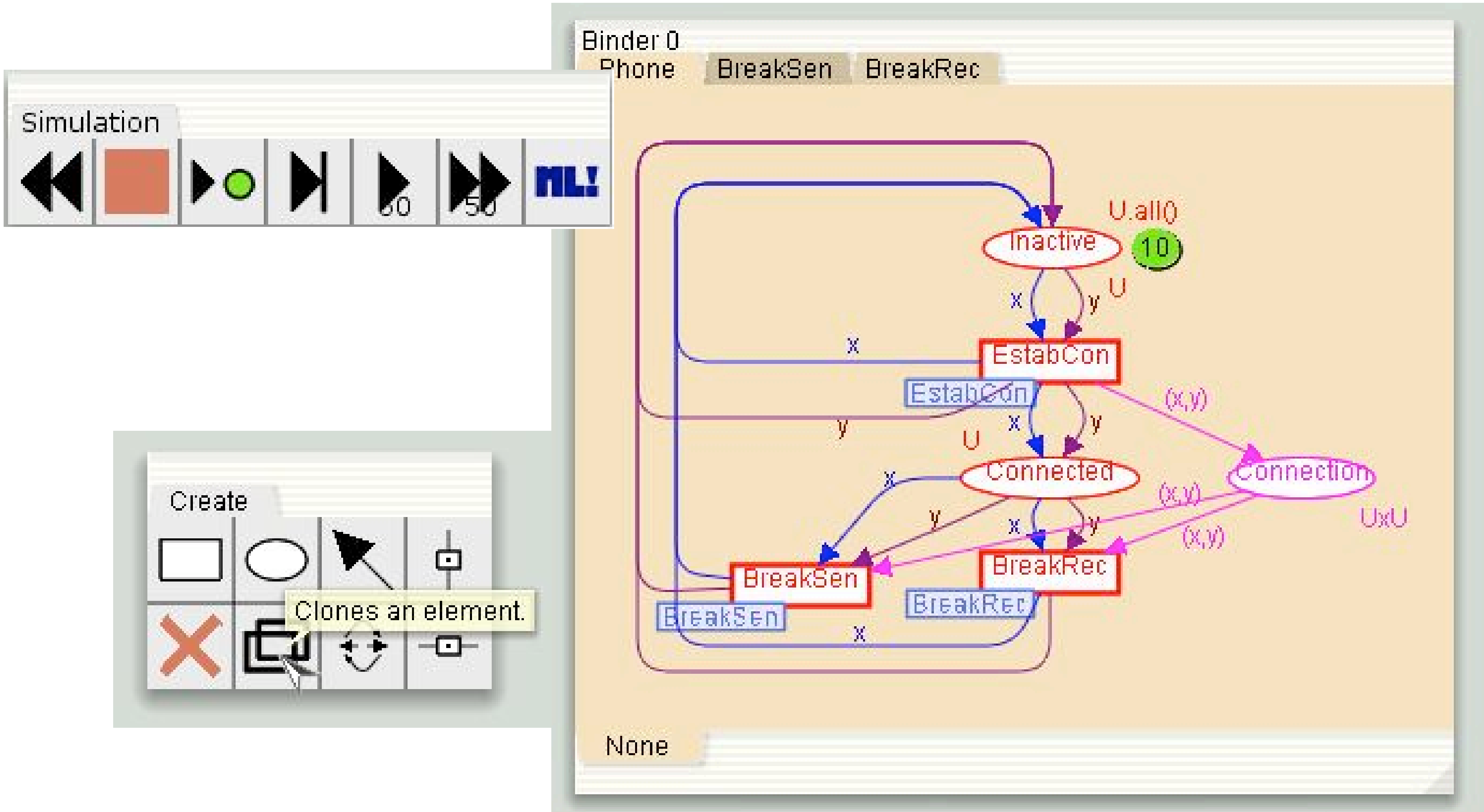


# Practical Tools: Pep

- = language to describe PN + a suite of tools to analyse them:
  - simulation
  - verification (SPIN, SMV)
  - translation from/to different formalisms
  - ...
- Everything can be accessed through a single graphical interface (Tcl/Tk)

http://theoretica.informatik.uni-oldenburg.de/~pep/

# Practical Tools: CPNTools



# Practical Tools: CPNTools

- Specialised in Coloured Petri nets
- Features similar to Pep:
  - modelisation
  - simulation
  - state space analysis
  - ...

http://wiki.daimi.au.dk/cpntools/cpntools.wiki

# Conclusion

# To conclude

- Petri nets (and their extensions) are a nice tool to reason about concurrent systems:
  - very popular
  - non-trivial decision problems are decidable
  - appealing graphical representation
  - tool supported

# To conclude

- There is still a lot to explore:
  - other extensions:
    - Time Petri nets
    - Timed Petri nets
    - Stochastic Petri nets,...

# To conclude

- There is still a lot to explore:
  - Subclasses of Petri nets:
    - 1-safe
    - marked graphs
    - free-choice
    - conflict free
    - ...
  - Some problems are easier to decide on these subclasses.

# To conclude

- There is still a lot to explore:
  - other problems:
    - liveness
    - deadlock freedom
    - semi-linearity
    - non-termination
    - ...

# To conclude

- Very active field of research !
  - Several conference and journals entirely dedicated to Petri nets
  - ... just hop in and join us !

http://www.informatik.uni-hamburg.de/TGI/PetriNets/

# Some references

- On Petri nets:
  - Reisig, W., Petri Nets: An introduction. Springer-Verlag, 1985.
  - Peterson, JL, Petri nets theory and modeling of systems, Prentice Hall, 1981
  - Girault, C. and Valk, R., Petri Nets for Systems Engineering - A Guide to Modeling, Verification, and Applications. Springer-Verlag, Berlin, Heidelberg, New York, 2003.
  - Javier Esparza, Mogens Nielsen, Decidability Issues for Petri Nets: a survey, Bulletin of the EATCS, 52:245--262, February 1994.
- On Petri nets with marking dependent effects:
  - Valk, R.: Self-Modifying Nets, a Natural Extension of Petri Nets. ICALP 1978: 464-476
  - G. Ciardo. Petri nets with marking-dependent arc multiplicity: properties and analysis. In R. Valette, editor, Application and Theory of Petri Nets 1994, Lecture Notes in Computer Science 815 (Proc. 15th Int. Conf. on Applications and Theory of Petri Nets, Zaragoza, Spain), pages 179-198. Springer-Verlag, June 1994.

# Some references

- On the coverability problem:
  - Richard M. Karp, Raymond E. Miller: Parallel Program Schemata. J. Comput. Syst. Sci. 3(2): 147-195 (1969)
  - Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Tsay Yih-Kuen. General Decidability Theorems for Infinite-State Systems. Proc. LICS'96, 11th IEEE Int. Symp. on Logic in Computer Science, New Brunswick, New Jersey, USA, 1996.
  - Finkel, A. and Schnoebelen, P. 2001. Well-structured transition systems everywhere!. Theor. Comput. Sci. 256, 1-2 (Apr. 2001), 63-92. DOI= http://dx.doi.org/10.1016/S0304-3975(00)00102-X
  - Geeraerts, Raskin, Van Begin, Expand, Enlarge and Check: new algorithms for the coverability problem of WSTS. Journal of Computer and System Sciences, volume 72(1), pp 180-203, Elsevier, 2005.
  - Giorgio Delzanno, Jean-François Raskin, Laurent Van Begin: Covering sharing trees: a compact data structure for parameterized verification. STTT 5(2-3): 268-297 (2004).
  - Geeraerts, Raskin, Van Begin. On the efficient Computation of the Minimal Coverability set of Petri nets. In Proceedings ATVA07, Lecture Notes in Computer Science, volume 4762, pages 98--113, Springer Verlag.

# Some references

- On Coloured Petri nets:
  - K. Jensen: A Brief Introduction to Coloured Petri Nets. In: E. Brinksma (ed.): Tools and Algorithms for the Construction and Analysis of Systems. Proceeding of the TACAS'97 Workshop, Enschede, The Netherlands 1997, Lecture Notes in Computer Science Vol. 1217, Springer-Verlag 1997, 203-208.
  - K. Jensen: An Introduction to the Theoretical Aspects of Coloured Petri Nets. In: J.W. de Bakker, W.-P. de Roever, G. Rozenberg (eds.): A Decade of Concurrency, Lecture Notes in Computer Science vol. 803, Springer-Verlag 1994, 230-272.
  - Jensen, K, Rozenberg, G. High-level petri nets : theory and application, Springer, 1991

# Some references

- On other extensions of Petri nets:
  - M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli and G. Franceschinis Modelling with Generalized Stochastic Petri Nets, Wiley Series in Parallel Computing, John Wiley and Sons
  - F. Bause, P. Kritzinger, Stochastic Petri Nets -- An Introduction to the Theory (2nd edition), Vieweg Verlag, Germany, 2002.
  - J. Wang, Timed Petri Nets, Theory and Application, Kluwer Academic Publishers 1998, ISBN: 0-7923-8270-6.
  - Louchka Popova. On time petri nets. Journal Information Processing and Cybernetics, EIK, 27(4):227–244, 1991.
- On net unfoldings:
  - J. Esparza, S. Römer, and W. Vogler. An improvement of mcmillan's unfolding algorithm. In Proc. TACAS '96, volume 1055 of Lecture Notes in Computer Science, pages 87-106. Springer-Verlag, 1997.
  - P. A. Abdulla, S. P. Iyer, and A. Nylén. Unfoldings of Unbounded Petri Nets. In Proc. of CAV '00, volume 1855 of Lecture Notes in Computer Science, pages 495-507. Springer-Verlag, 2000.
- More at:
  - http://www.informatik.uni-hamburg.de/TGI/PetriNets/introductions/

# Questions ?