Well-Structured Transition Systems and Extended Petri Nets —An Introduction—

Jean-François Raskin
ULB

AVACS Spring School - Oldenburg - March 2010

Plan of the talk

- Parametric systems - Parametric verification
- Well-quasi orders and well-structured transition systems
- Extended Petri nets
- Three algorithmic tools for WSTS:
  - The set saturation method
  - The finite unfolding (≠“Karp-Miller” tree)
  - The “Expand, Enlarge and Check” (EEC) algorithm
- Beyond this introduction - bibliography
- Conclusion

Introduction

Motivations

- Protocols are often designed to work for an arbitrary number of participants
- Multi-threaded programs may trigger the creation of an unbounded number of threads

- We need abstract models to reason about such systems
- We need techniques to establish correctness for an arbitrary number of participants/threads...
- We want parametric verification !

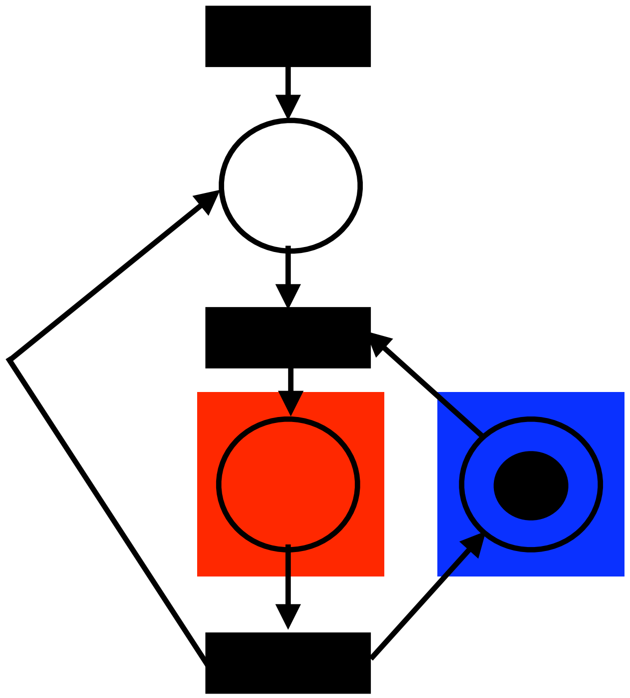
Parametric verification and PN

```
mutex M ;

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

Parametric verification and PN

```
mutex M ;

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

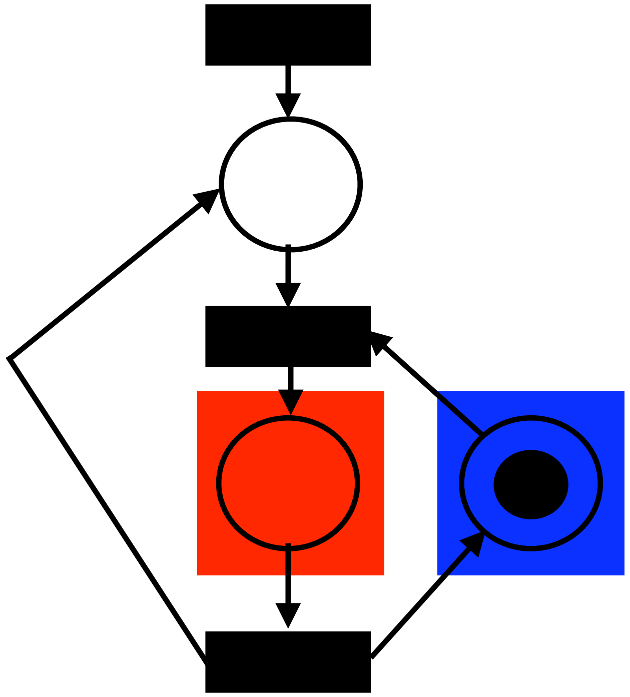
Counting abstraction



Parametric verification and PN

```
mutex M ;

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

Counting abstraction



Mutual exclusion is verified if there is no more than one token in the red place in any reachable marking.

Motivations

- Protocols are often designed to work for an arbitrary number of participants
- Multi-threaded programs may trigger the creation of an unbounded number of threads

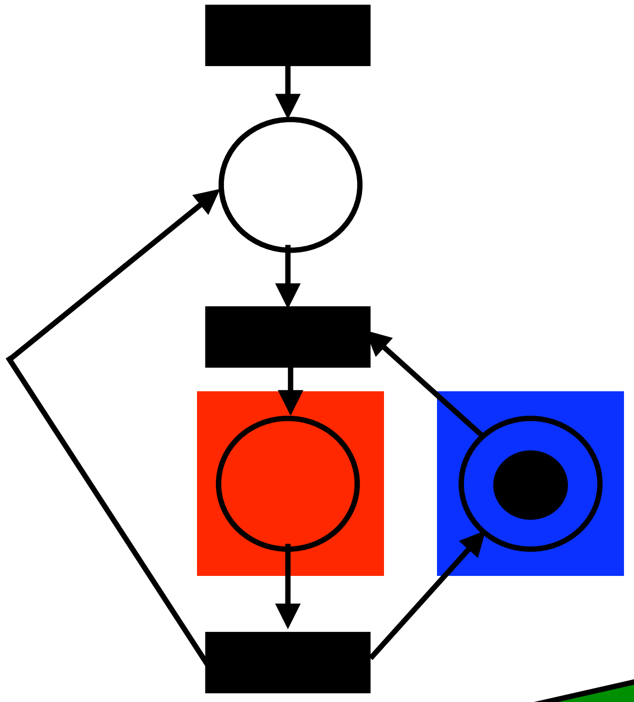
- We need abstract models to reason about such protocols/programs.

- Well structured transition systems (WSTS) are such abstract models.
- WSTS enjoy general decidability results.

Parametric verification and PN

```
mutex M ;

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

Counting abstraction



Mutual excl … no more than one token … ce in any reachable marking.

This is a **coverability** property !

Coverability properties are decidable for the class of WSTS !

Well quasi-orders
Well Structured Transition Systems

Well quasi-order

- Let S be a (possibly infinite) set, a relation $\leq \subseteq S \times S$ is
  - A pre-order iff $\leq$ is reflexive and transitive;
  - A partial-order iff $\leq$ is a pre-order and antisymmetric;
  - A total order iff $\leq$ is a partial-order and total.
- $(S, \leq)$ is an ordered set if $\leq$ is a pre-order on S.

Well quasi-order

- Let $(S,\leq)$ be an ordered set, $\leq$ is well-founded iff there is no infinite decreasing chains.

~~$s_1 > s_2 > s_3 > \dots > s_n > \dots.$~~

- Let $(S,\leq)$ be an ordered set, $\leq$ is a well-quasi ordering (WQO) iff in any infinite sequence $s_1 s_2 \dots s_i \dots$ there exist two positions $k<l$ s.t. $s_k \leq s_l$.

$$s_1\ s_2 \dots \underbrace{s_k \dots s_l}_{\leq} \dots$$

Well quasi-order

- (S,≤) is called a well-quasi ordered set if ≤ is a WQO.
- Clearly, all well-quasi ordered sets (S,≤) are well-founded sets.
- The set ($\mathbb{N}$,≤) is a well-quasi ordered set.

The set $(\mathbb{N},\leq)$ is a well-quasi ordered set

Indeed, consider for the sake of contradiction that it is not the case.

Then there exists a sequence of natural numbers $n_0 n_1 \dots n_i \dots$ such that for all $k<l : \neg(n_k \leq n_l)$.

But as $\leq$ is a total order, we have then for all $k<l : n_k > n_l$ i.e., an infinite strictly decreasing sequence of elements which is not possible.

Well quasi-order

Lemma. Let $(S,\leq)$ be a WQO set. From every infinite sequence $s_1 s_2 \ldots s_j \ldots$ in S we can extract an infinite subsequence which is increasing i.e., a subsequence $s_{f(1)} s_{f(2)} \ldots s_{f(j)} \ldots$ with $f(i)<f(i+1)$ for all $i \geq 1$, and such that $s_{f(i)} \leq s_{f(i+1)}$ for all $i \geq 1$.

from

$$s_1\ s_2\ s_3\ \ldots\ s_n\ \ldots$$

we can extract

$$s_{f(1)} \leq s_{f(2)} \leq \ldots \leq s_{f(i)} \leq \ldots$$

with

$$f(1) < f(2) < \ldots < f(i) < \ldots$$

$(\mathbb{N}^k, \preccurlyeq)$ is a well quasi-ordered set

- The set $(\mathbb{N}^k, \preccurlyeq)$, where $\preccurlyeq$ is the pointwise extension of $\leq$ on k-tuples of natural number i.e.,

  $(c_1, c_2, \ldots, c_k) \preccurlyeq (d_1, d_2, \ldots, d_k)$
  iff $c_i \leq d_i$ for all i, $1 \leq i \leq k$.

- .... is a well-quasi ordered set.

$(\mathbb{N}^k,\preccurlyeq)$ is a well quasi-ordered set

By induction on k. If k=1, the theorem holds as $(\mathbb{N},\leq)$ is a well-quasi ordered set.

Induction. Let k=i>1. By induction hyp. $(\mathbb{N}^{k-1},\preccurlyeq)$ is WQO set.

Assume for the sake of contradiction that $v_1 v_2 \ldots v_j \ldots$ is an infinite sequence of incomparable elements in $(\mathbb{N}^k,\preccurlyeq)$.

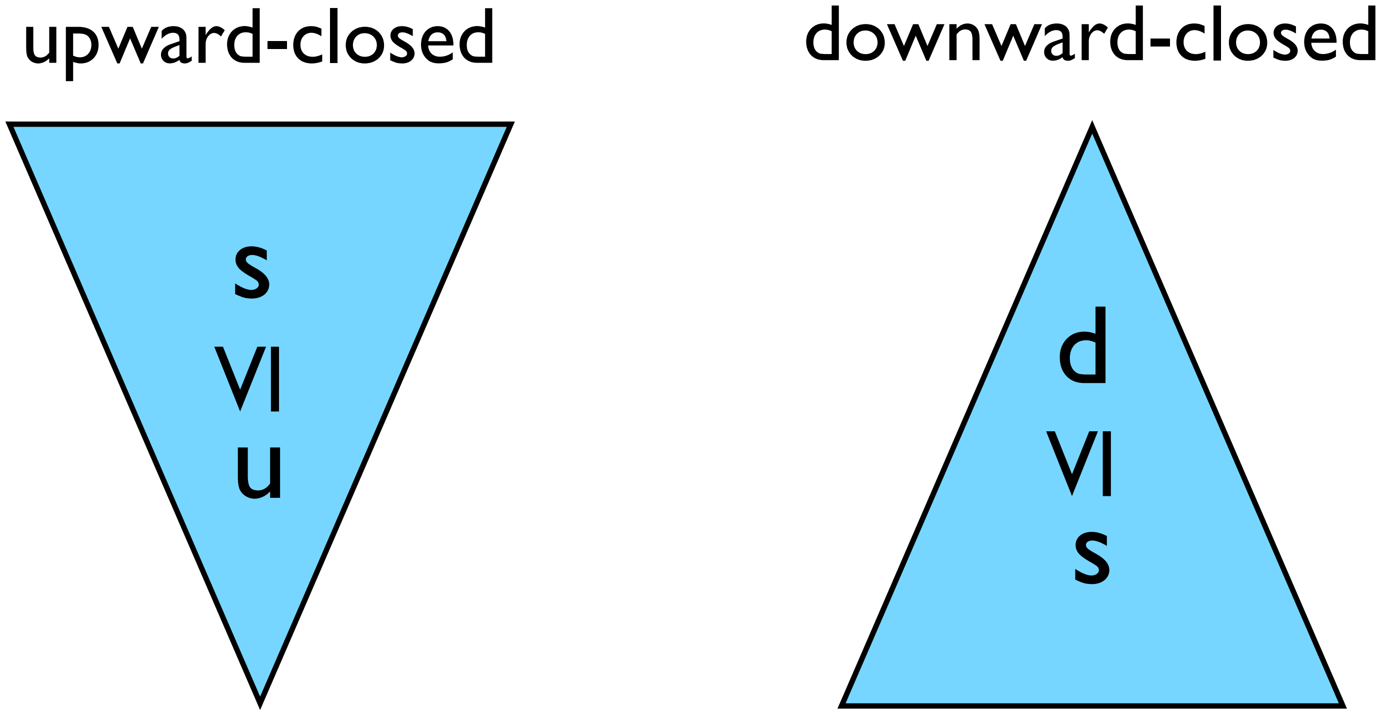
Let us consider the projection of this sequence on the dimensions 2,3,..,k : $v_1(2..i)\ v_2(2..i)\ldots v_j(2..i)\ldots$

By induction hypothesis $(\mathbb{N}^{k-1},\preccurlyeq)$ is WQO and so we can extract an infinite subsequence of increasing elements in $\mathbb{N}^{k-1}$. Let f(1)f(2)...f(j)... be the indices corresponding to this subsequence.

Clearly the sequence $v_{f(1)}(1) v_{f(2)}(1) \ldots v_{f(j)}(1) \ldots$ must be a sequence of pairwise incomparable elements. But this contradict the fact that $(\mathbb{N},\leq)$ is a WQO set.

Upward and downward closed sets

- Let (S,≤) be a ordered set.
- The set U⊆S is upward-closed
  iff for all u∈U for all s∈S : if u≤s then s∈U.
- The set D⊆S is downward-closed
  iff for all d∈D for all s∈S : if s≤d then s∈D.

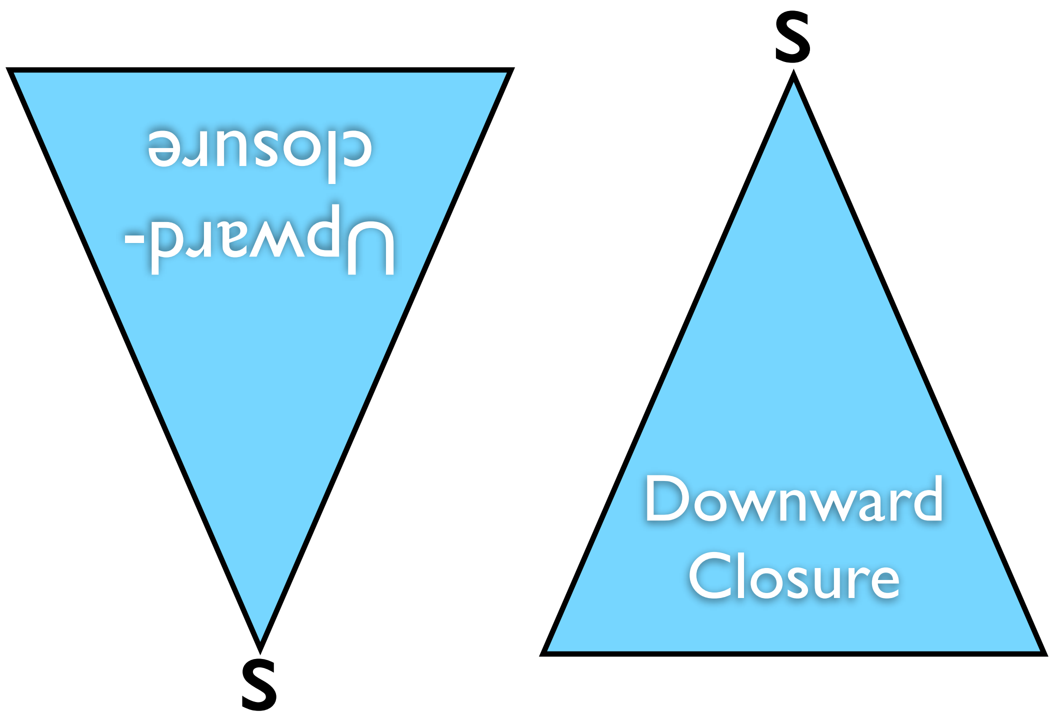


upward-closed

downward-closed

Upward and downward closed sets

- Let $(S,\leq)$ be a ordered set.
- Let $S' \subseteq S$. The upward-closure of S', noted $\uparrow S'$, is the set $\{ s \in S \mid \exists s' \in S' \bullet s' \leq s \}$.
- Let $S' \subseteq S$. The downward-closure of S', noted $\downarrow S'$, is the set $\{ s \in S \mid \exists s' \in S' \bullet s \leq s' \}$.

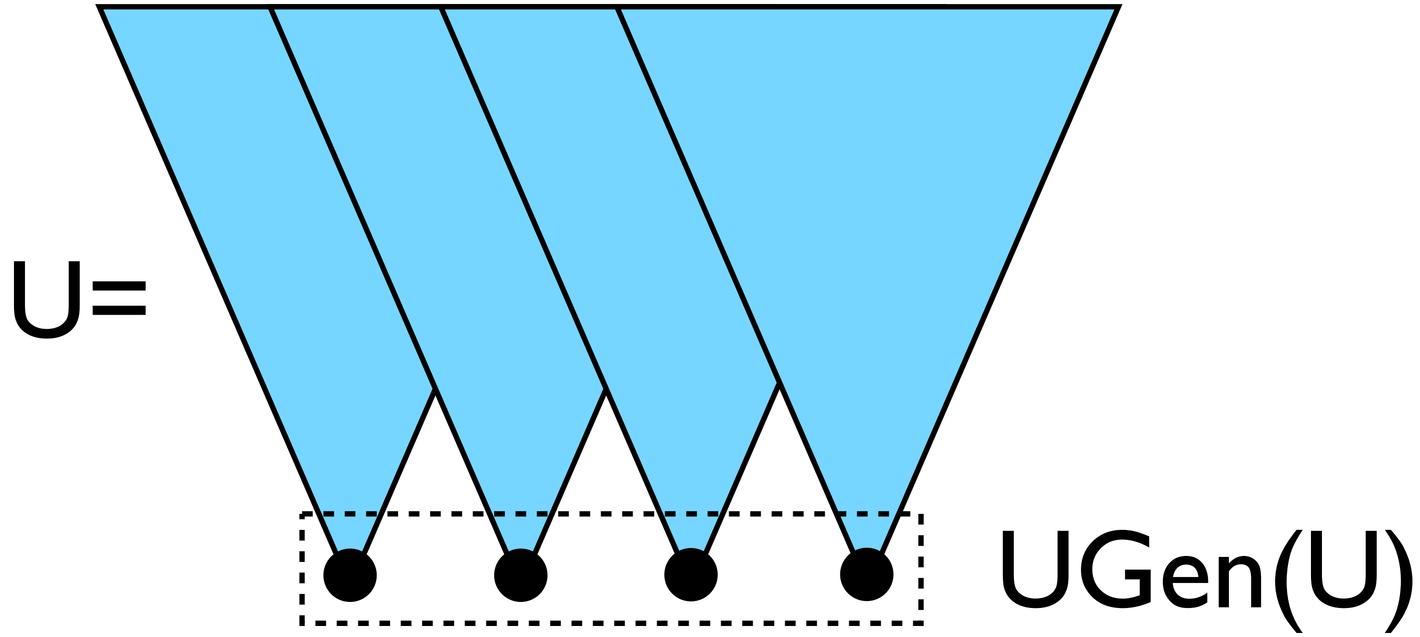


Generators of upward closed sets

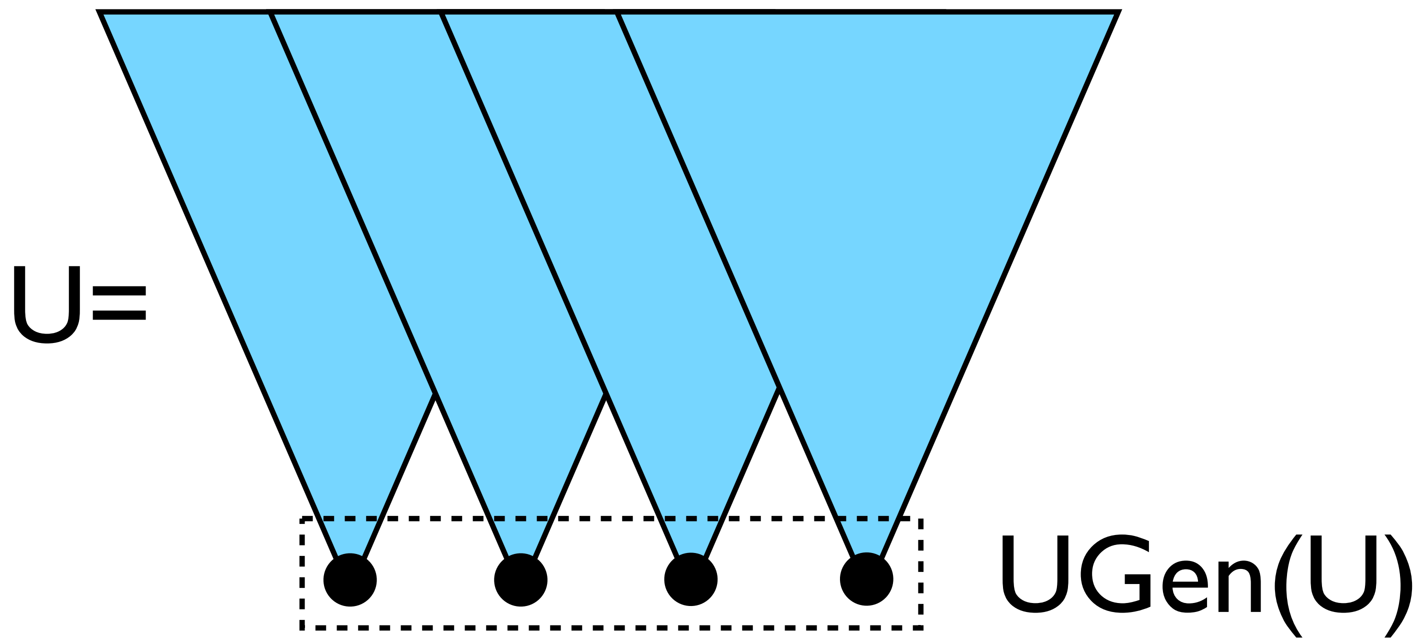
- Let $(S,\leq)$ be a ordered set.
- A set $A\subseteq S$ is an antichain if for all $a_1,a_2\in A$, if $a_1\neq a_2$ then neither $a_1\leq a_2$ nor $a_2\leq a_1$ i.e., a_1 and a_2 are incomparable.
- Let $U\subseteq S$ be an upward closed set. A set G is a generator for U if $\uparrow G=U$.
- Let $U\subseteq S$ be an upward closed set. Then UGen(U) is a set of elements of S such that:
    - $\mathrm{UGen}(U)\subseteq U$;
    - UGen(U) is a generator for U;
    - UGen(U) is an antichain.

Generators of upward closed sets

- Let U⊆S be an upward closed set. Then UGen(U) is a set of elements of S such that:
  - UGen(U)⊆U;
  - UGen(U) is a generator for U;
  - UGen(U) is an antichain.

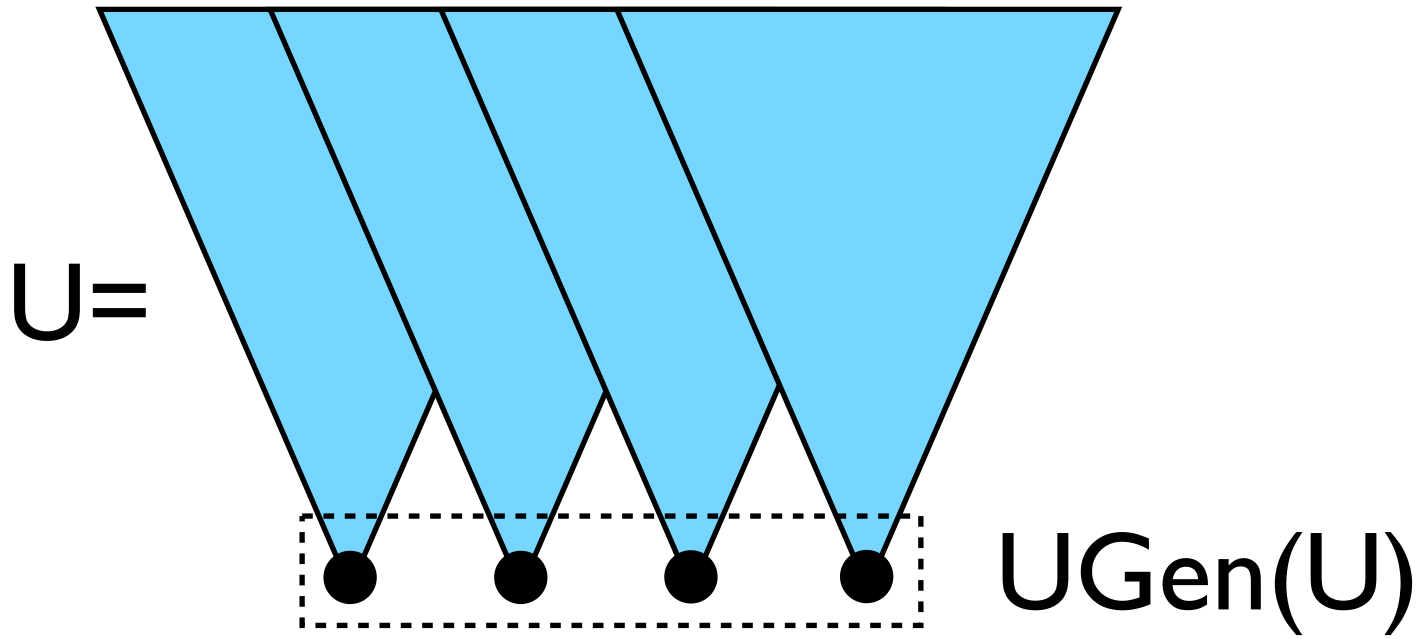


Generators of upward closed sets



- **Theorem**. Let (S,≤) be a WQO. Let U⊆S be an upward closed set. Then there exists a set A⊆U:
  - A is an antichain;
  - A is a generator of U.
  - A is finite.

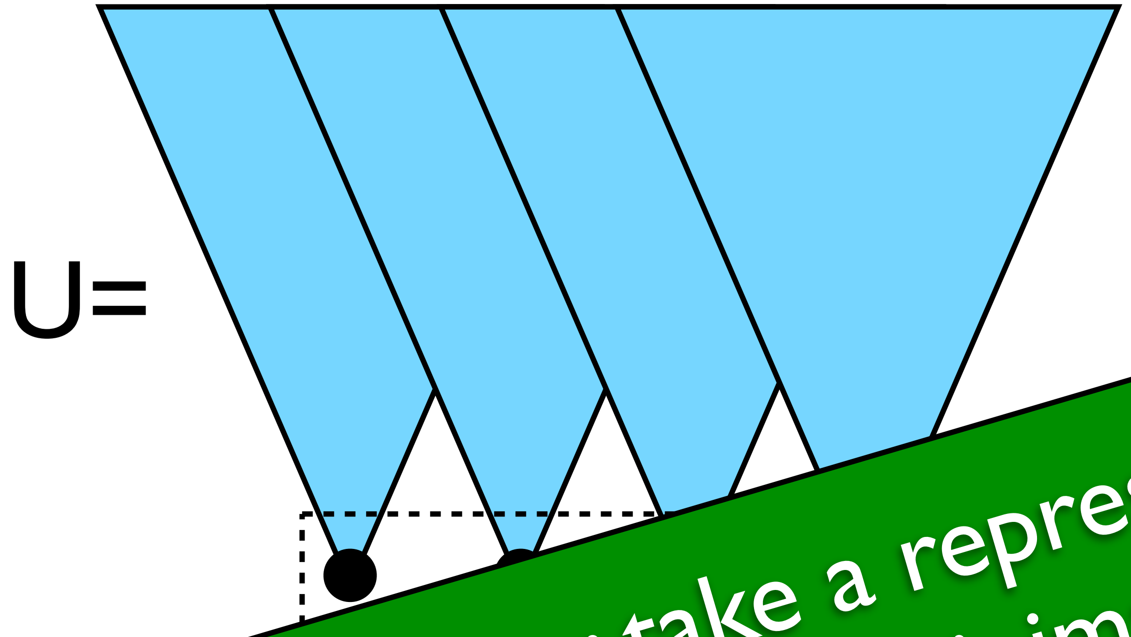
Generators of upward closed sets



- **Theorem**. Let (S,≤) be a WQO. Let closed set. Then there exi
  - A is an anti
  - A
  - A is

If ≤ is a partial order: take the finite set of minimal elements !

Generators of upward closed sets

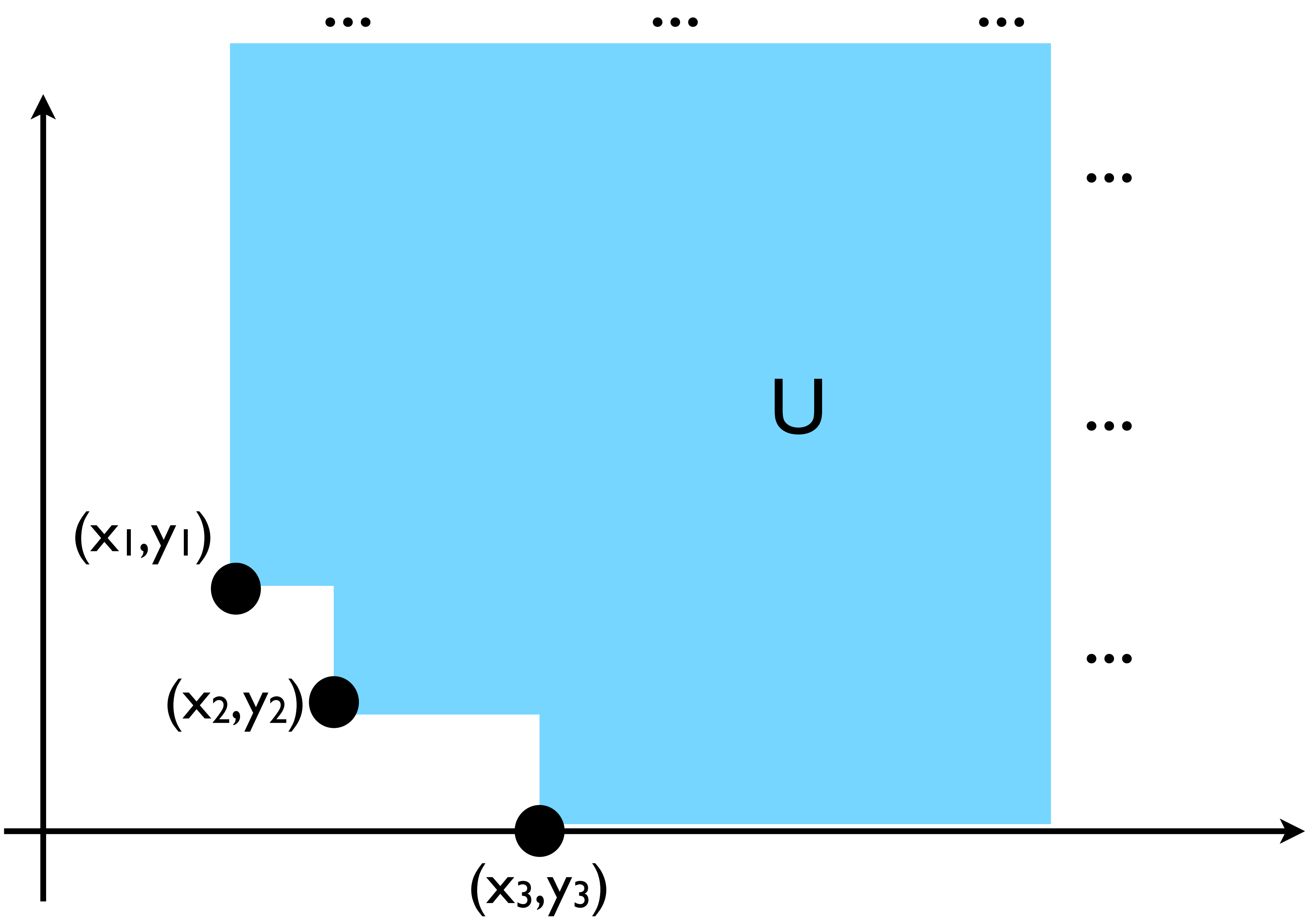


If ≤ is a pre-order: take a representative in each equivalence class of minimal elements !

If ≤ is a partial order: take the finite set of minimal elements !

- WQO. Le
  c there exi
  - A is an anti
  - A
  - A is

Upward closed sets in $(\mathbb{N}^k, \preccurlyeq)$



$\mathrm{Min}(U)=\{(x_1,y_1), (x_2,y_2)\ (x_3,y_3)\}$ is a finite generator for U.

Well Structured Transition Systems

Transition system

- A transition system is a tuple $T=(C,c_0,\Longrightarrow)$ where :
  - C is a (possibly infinite) set of configurations
  - $c_0 \in C$ is the initial configuration
  - $\Longrightarrow \subseteq C \times C$ is the transition relation

Well structured transition system

- A well-structured transition system is a tuple $T=(C,c0,\Longrightarrow,\leq)$ where:
  - $(C,c0,\Longrightarrow)$ is a transition system
  - $(C,\leq)$ is a well-quasi ordered set
  - $\Longrightarrow$ is monotonic: for all $c_1,c_2,c_3\in C$:
    if $c_1\Longrightarrow c_2$ and $c_1\leq c_3$
    then there exists c_4: $c_3\Longrightarrow c_4$ and $c_2\leq c_4$.

Well structured transition system

- A well-structured transition system is a tuple $T=(C,c0,\Longrightarrow,\leq)$ where:
  - $(C,c0,\Longrightarrow)$ is a transition system
  - $(C,\leq)$ is a well-quasi ordered set
  - $\Longrightarrow$ is monotonic: for all $c_1,c_2,c_3 \in C$:
    if $c_1 \Longrightarrow c_2$ and $c_1 \leq c_3$
    then there exists c_4: $c_3 \Longrightarrow c_4$ and $c_2 \leq c_4$.

$$\begin{tikzcd} & c_3 & \\ \forall & \rotatebox{90}{$\leq$} & \\ & c_1 \arrow[r, Rightarrow] & c_2 \end{tikzcd}$$

Well structured transition system

- A well-structured transition system is a tuple $T=(C,c0,\Longrightarrow,\leq)$ where:
  - $(C,c0,\Longrightarrow)$ is a transition system
  - $(C,\leq)$ is a well-quasi ordered set
  - $\Longrightarrow$ is monotonic: for all $c_1,c_2,c_3 \in C$:
    if $c_1 \Longrightarrow c_2$ and $c_1 \leq c_3$
    then there exists c_4: $c_3 \Longrightarrow c_4$ and $c_2 \leq c_4$.

$$\begin{tikzcd}
 & c_3 \arrow[r, Rightarrow, "\exists"] & c_4 \\
\forall & c_1 \arrow[u, phantom, sloped, "\leq"] \arrow[r, Rightarrow] & c_2 \arrow[u, phantom, sloped, "\leq"]
\end{tikzcd}$$

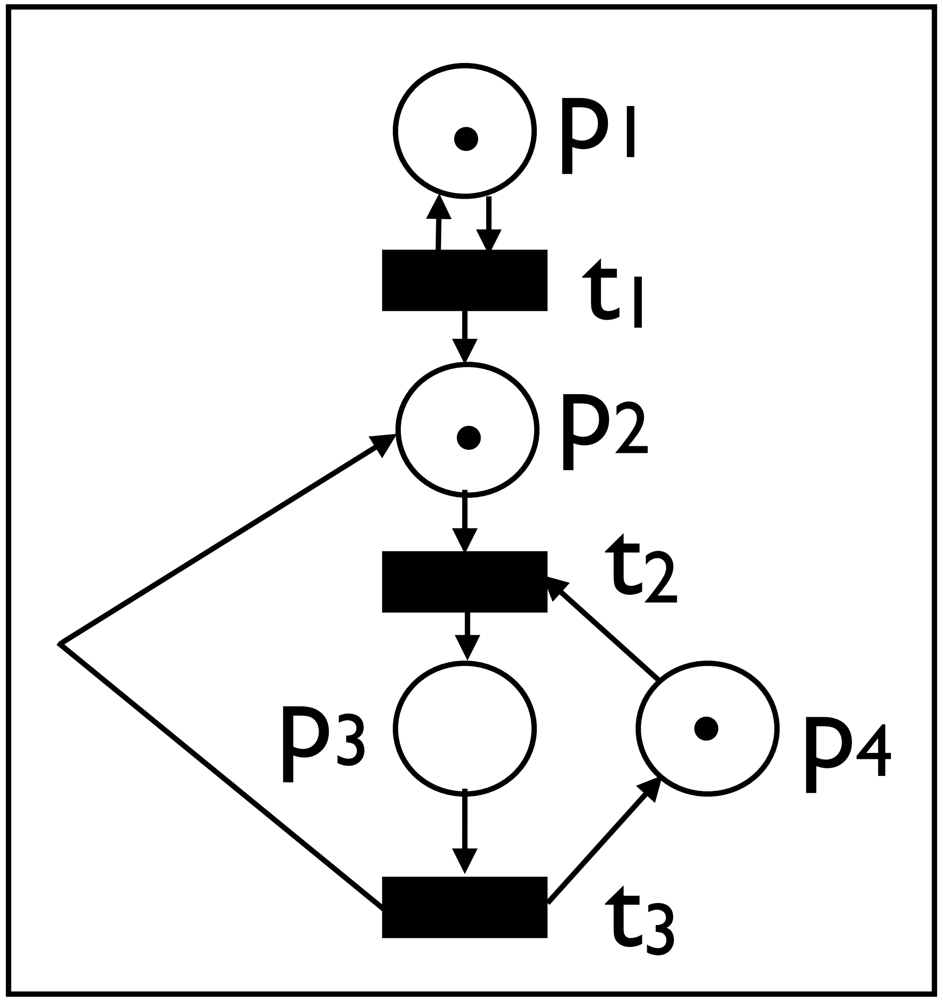
Predicate transformer for TS

- Predicate transformers:
  - Post(c)= { c' | c⟹c' }
    - As usual, for S⊆C, we write Post(S) for $\cup_{c\in S}$ Post(c).
    - $Post^1$=Post and $Post^i$=Post○$Post^{i-1}$ and $Post^*$=$\cup_{i\geq 0}$ $Post^i$.
    - Reach(T)=$Post^*(c_0)$.
  - Pre(c)= { c' | c'⟹c }
    - As usual, for S⊆C, we write Pre(S) for $\cup_{c\in S}$ Pre(c).
    - Pre^1=Pre and Pre^i=Pre○Pre^{i-1} and Pre^*=$\cup_{i\geq 0}$ Pre^i.

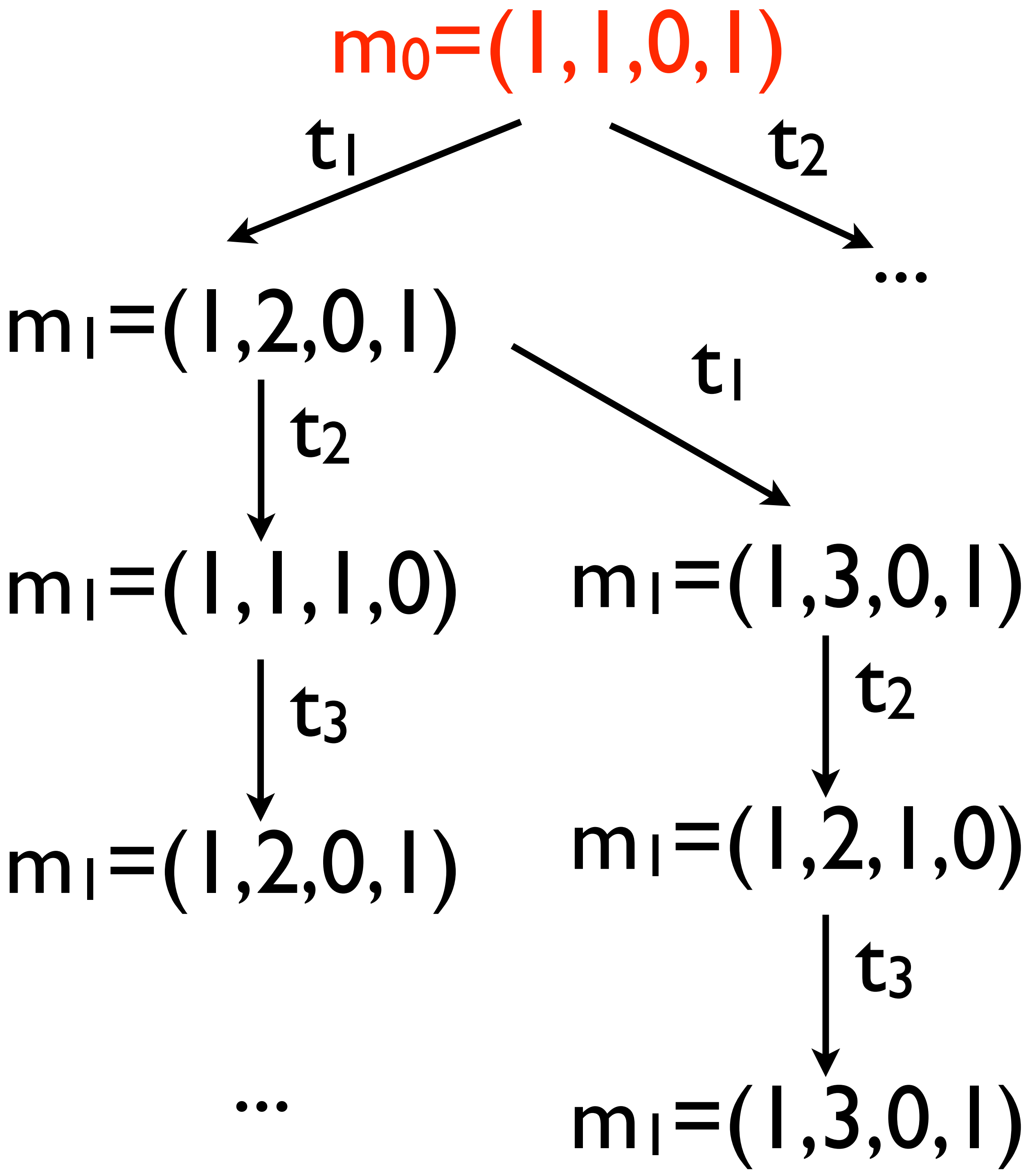
Petri nets and Extended Petri nets

Exemple of PN

Petri nets are an important and traditional model for modeling concurrent systems.



Exemple of PN



$m_0=(1,1,0,1)$

t_1

t_2

...

$m_1=(1,2,0,1)$

t_2

t_1

$m_1=(1,1,1,0)$

$m_1=(1,3,0,1)$

t_3

t_2

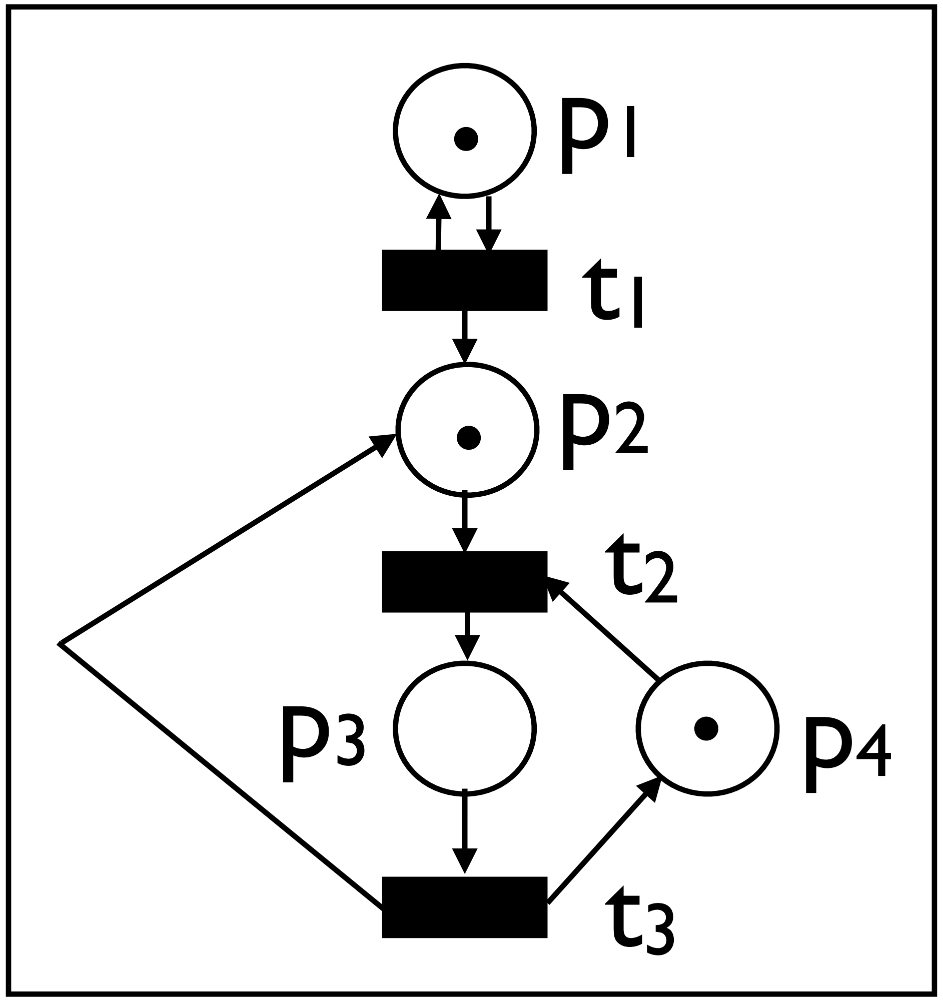
$m_1=(1,2,0,1)$

$m_1=(1,2,1,0)$

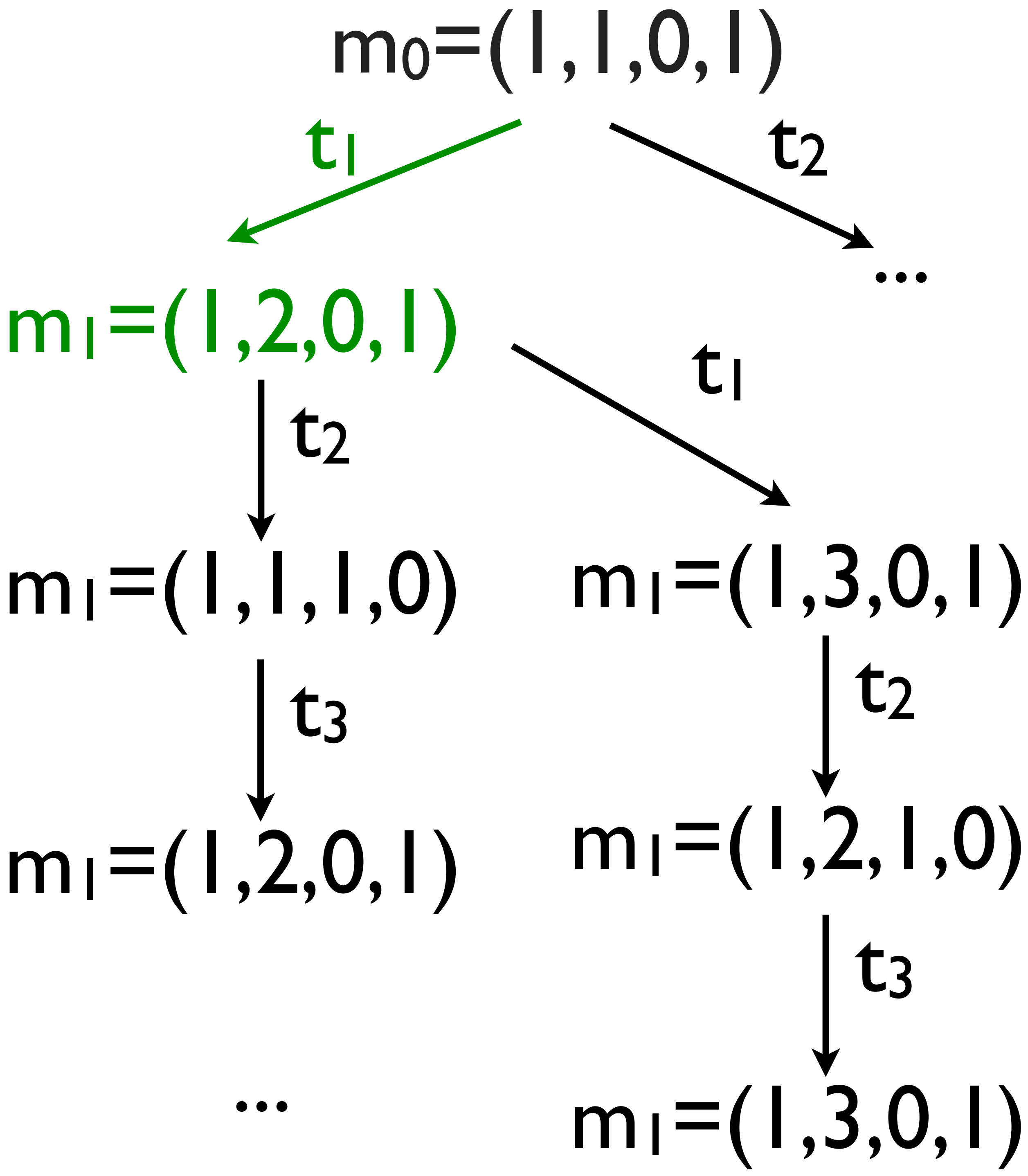
t_3

...

$m_1=(1,3,0,1)$



Exemple of PN



$m_0=(1,1,0,1)$

t_1 t_2

$m_1=(1,2,0,1)$ …

t_2 t_1

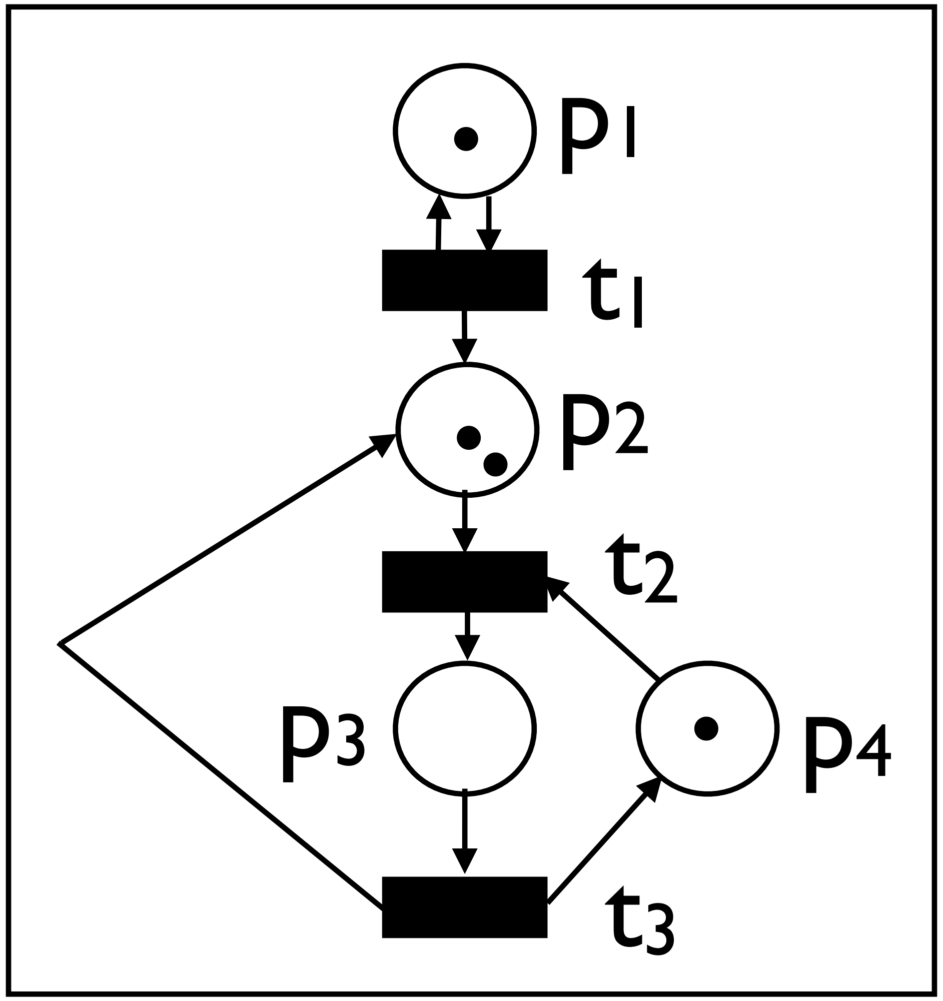
$m_1=(1,1,1,0)$ $m_1=(1,3,0,1)$

t_3 t_2

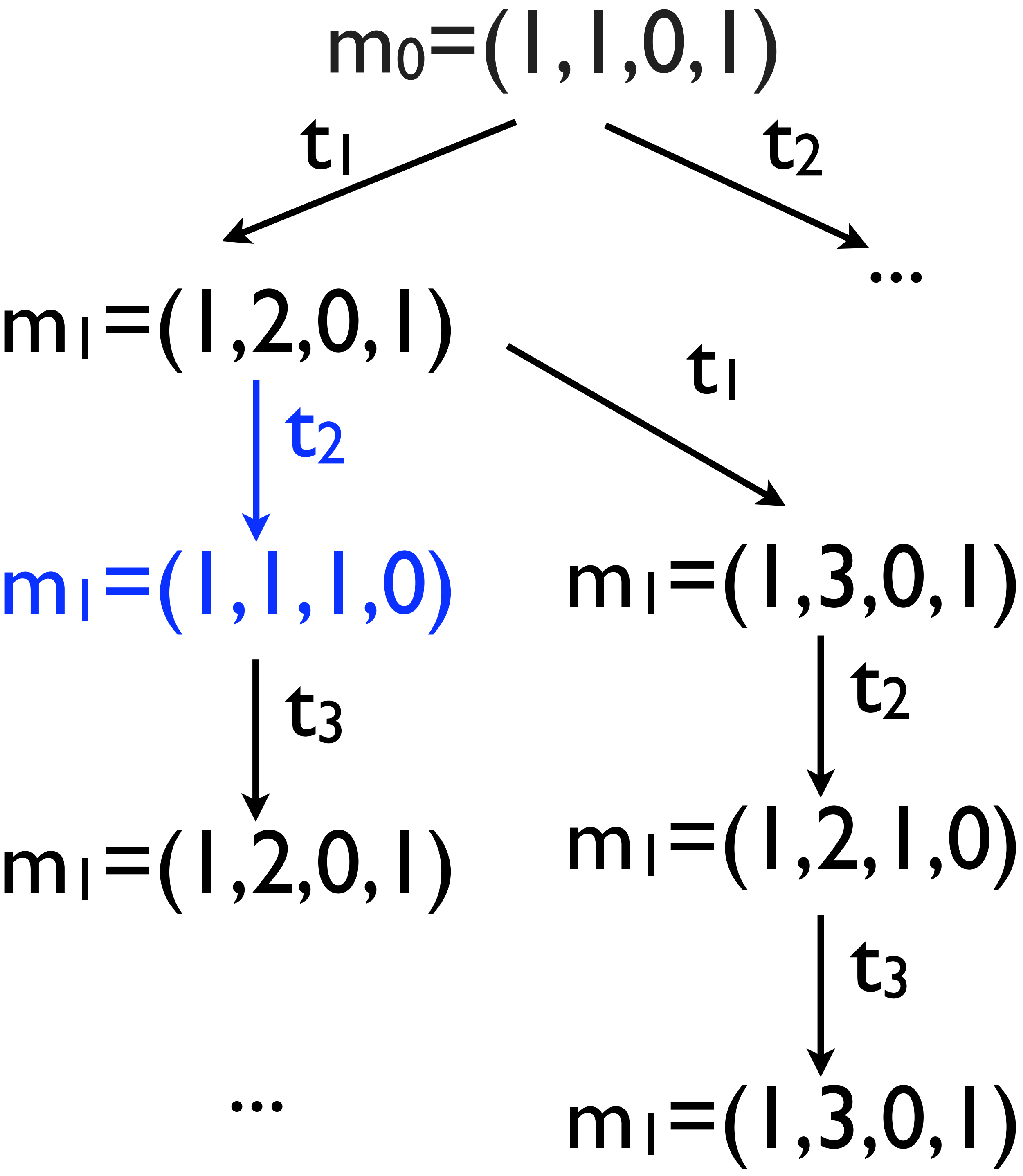
$m_1=(1,2,0,1)$ $m_1=(1,2,1,0)$

… t_3

$m_1=(1,3,0,1)$



Exemple of PN



$m_0=(1,1,0,1)$

t_1

t_2

$m_1=(1,2,0,1)$

...

t_2

t_1

$m_1=(1,1,1,0)$

$m_1=(1,3,0,1)$

t_3

t_2

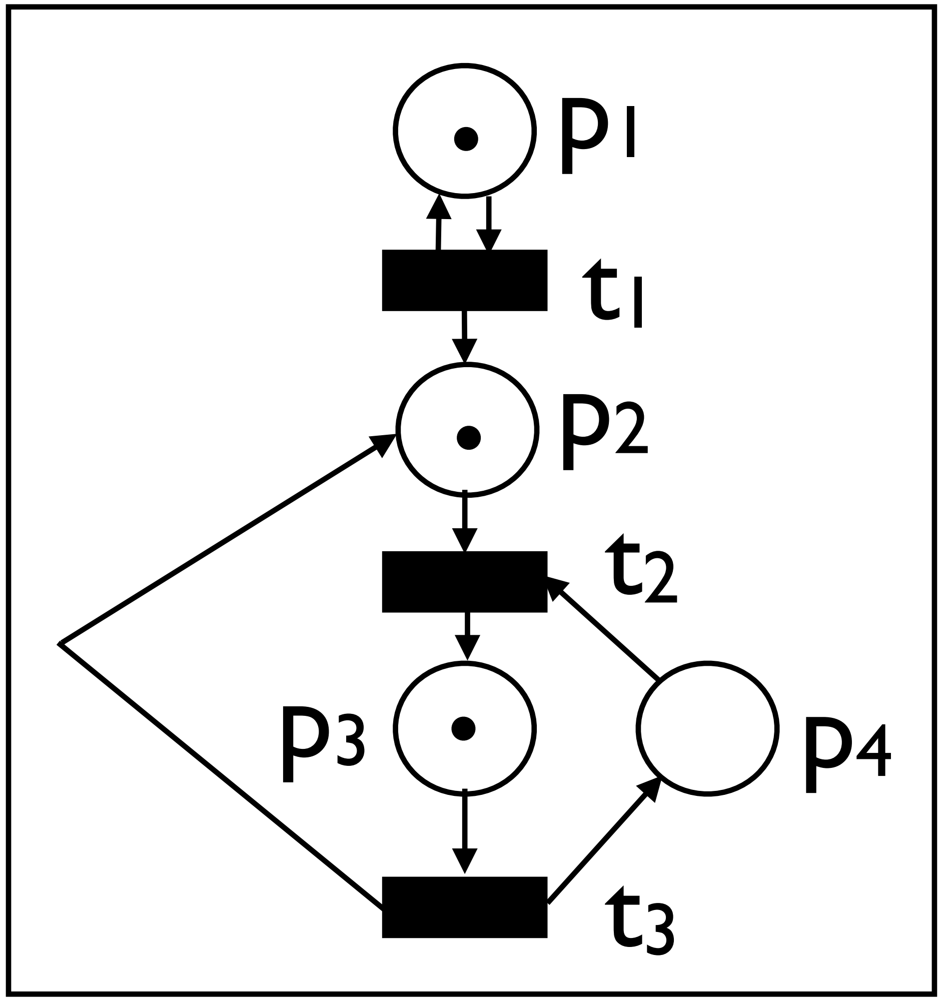
$m_1=(1,2,0,1)$

$m_1=(1,2,1,0)$

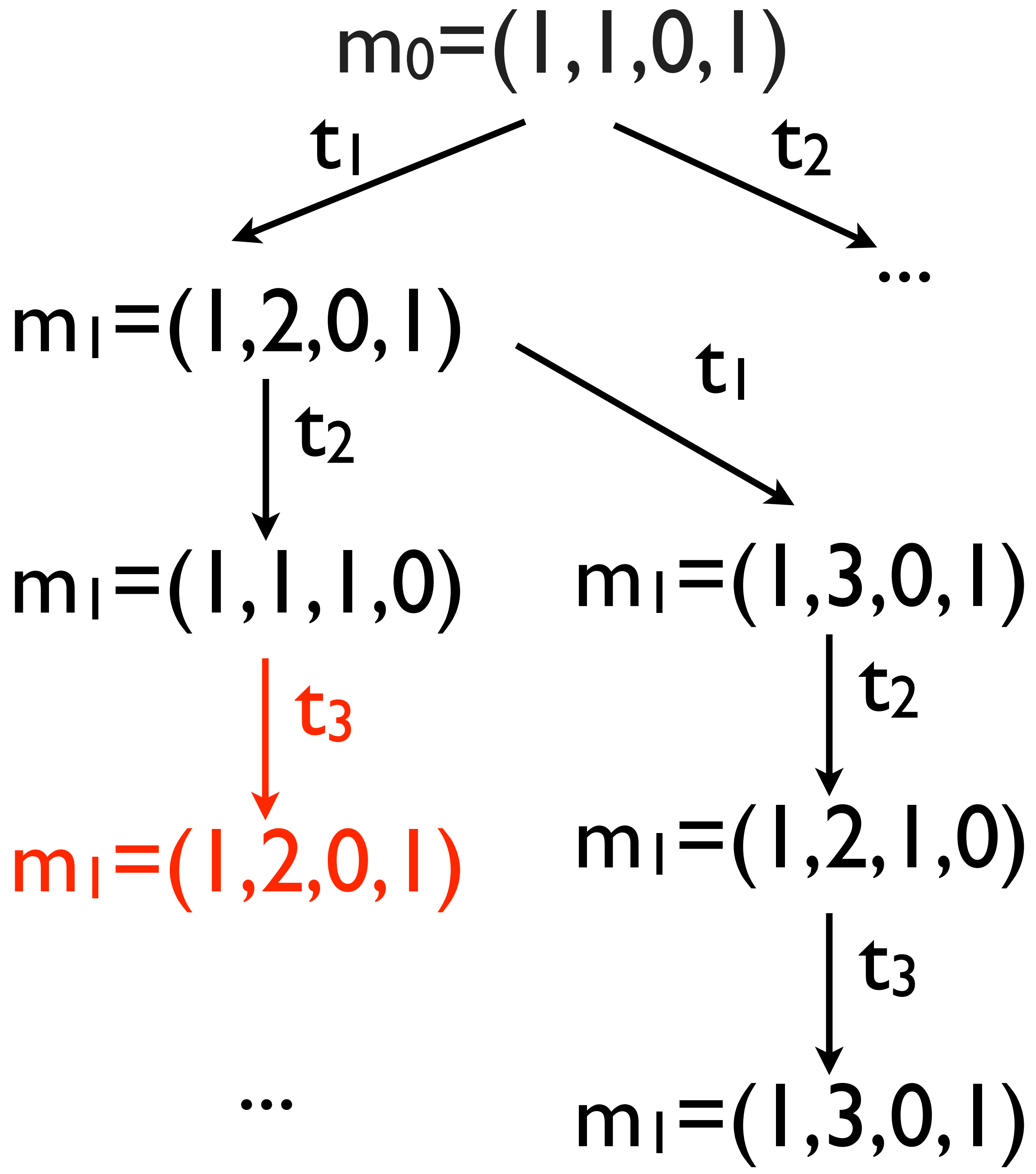
t_3

...

$m_1=(1,3,0,1)$



Exemple of PN



$m_0=(1,1,0,1)$

t_1 t_2

$m_1=(1,2,0,1)$ ...

t_2 t_1

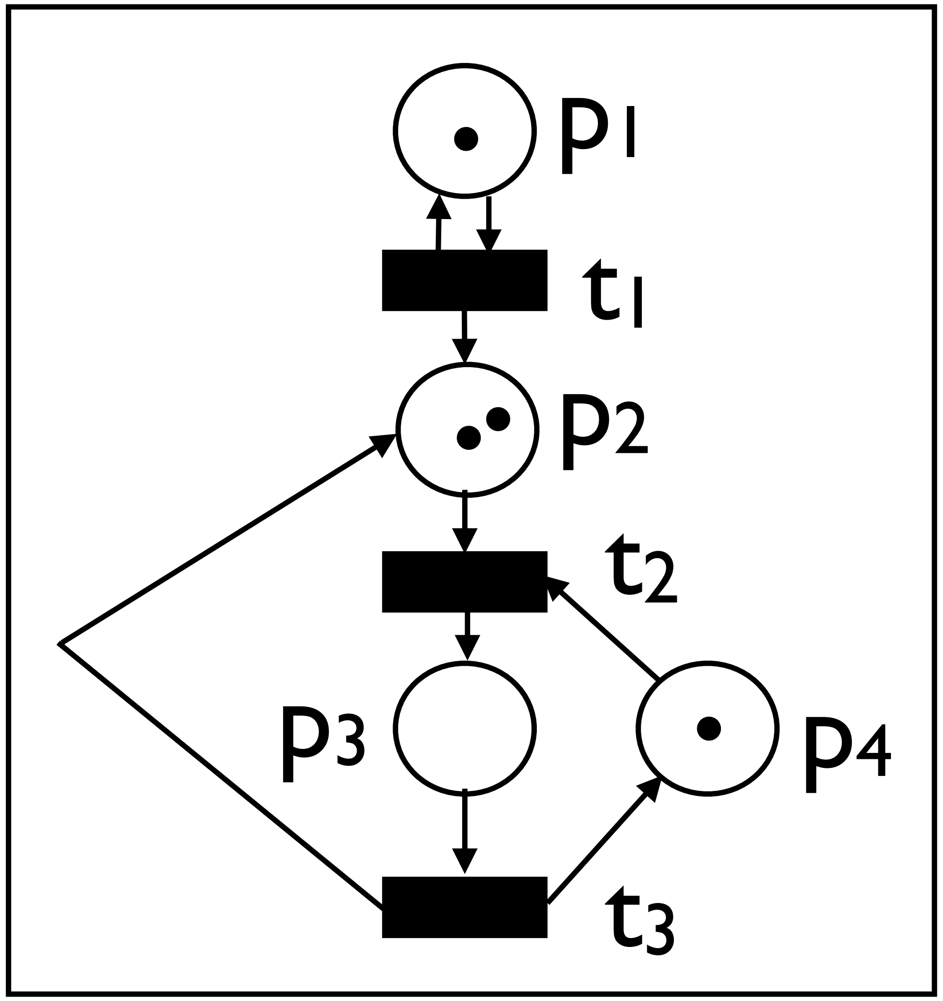
$m_1=(1,1,1,0)$ $m_1=(1,3,0,1)$

t_3 t_2

$m_1=(1,2,0,1)$ $m_1=(1,2,1,0)$

t_3

... $m_1=(1,3,0,1)$



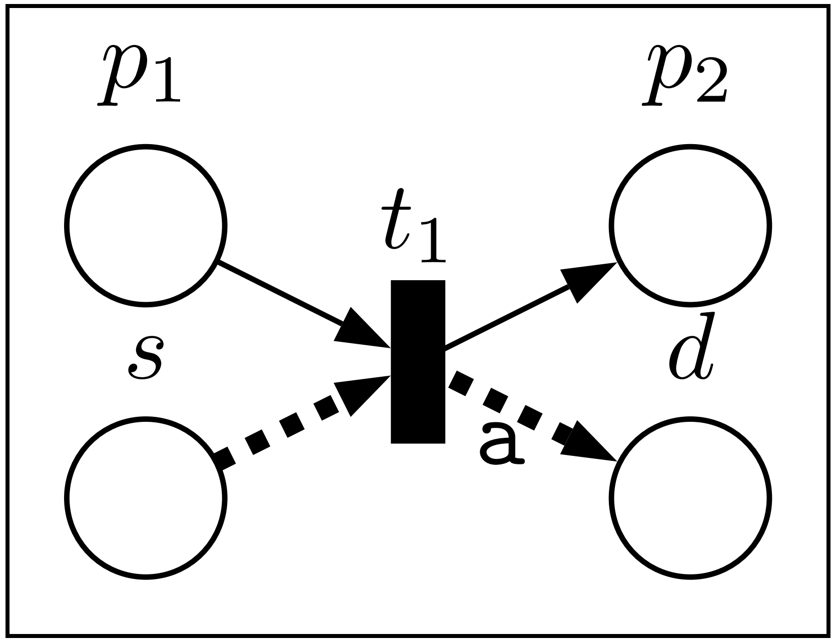
Extended Petri Nets

- A extended Petri net $N=(P,T,m_0)$ where :
  - $P=\{p_1,p_2,...,p_n\}$ is a finite set of places;
  - $T=\{t_1,t_2,...,t_m\}$ is a finite set of transitions, each of which is of the form (I,O,s,d,b) where :
    - ★ $I : P \to \mathbb{N}$ are multi-sets of input places, $I(p)$ represents the number of occurences of p in I.
    - ★ $O : P \to \mathbb{N}$ are multi-sets of output places.
    - ★ $s,d \in P\cup\{\perp\}$ are the source and destination places of a special arc and $b\in\mathbb{N}\cup\{+\infty\}$ is the bound associated to the special arc.
- We partition T into $T_r \cup T_e$ where T_r contains regular transitions where $s=d=\perp$ and $b=0$, and T_e contains extended transitions where $s,d\in P$ and $b\neq 0$.

Extended Petri Nets

- ➡ A Petri net (PN) is a EPN where $T_e=\varnothing$.
- ➡ A Petri net with transfer arcs (PN+T) is such that for all $t=(I,O,s,d,b)\in Te$, $b=+\infty$.
- ➡ A Petri net with non-blocking arcs (PN+NBA) is such that for all $t=(I,O,s,d,b)\in T_e$, $b=1$.
- ➡ Extended Petri nets are useful to model synchronization mechanisms in counting abstractions such as non-blocking synchronization, broadcast, etc.

Example of PN+NBA

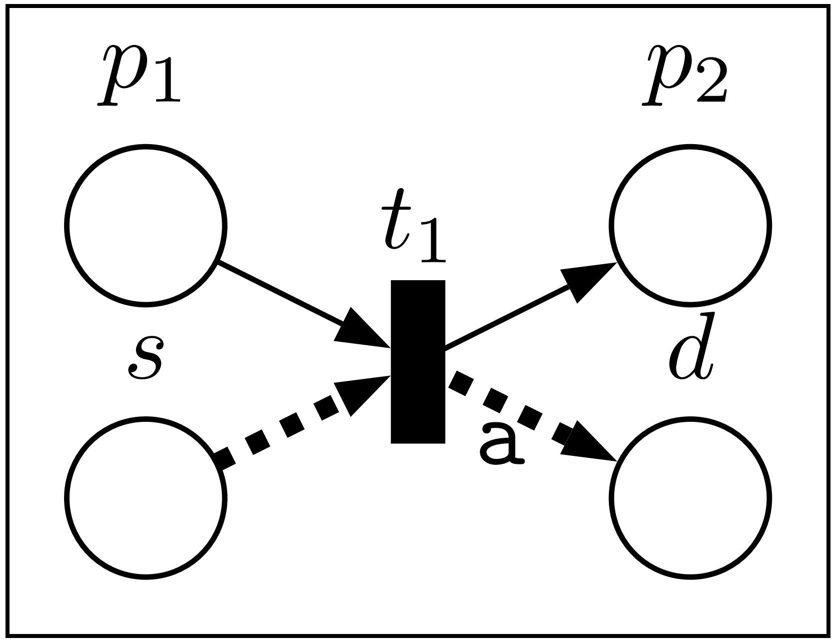


Example of PN+NBA

Non-blocking arcs

PN + NBA

At most one token gets moved from the source to the destination

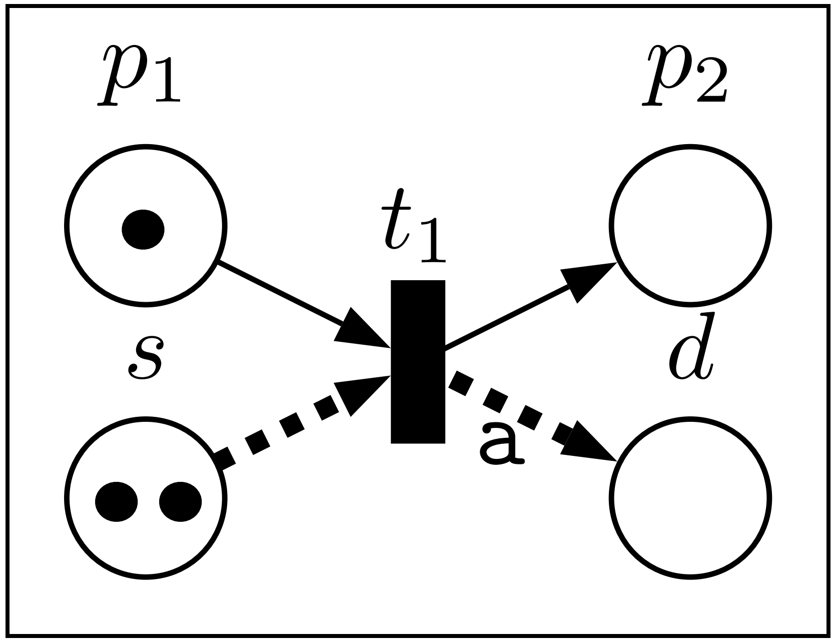


Example of PN+NBA

Non-blocking arcs

PN + NBA

At most one token gets moved from the source to the destination

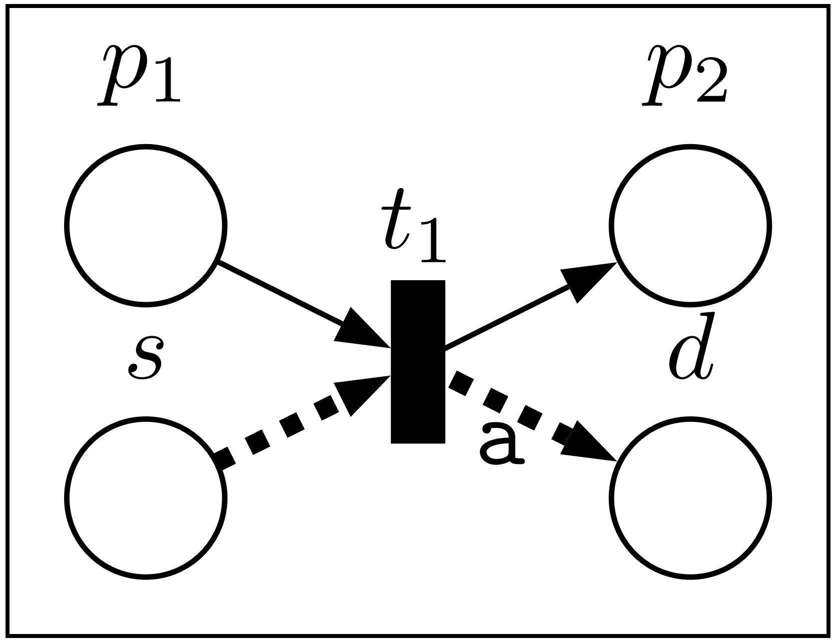


Example of PN+NBA

Non-blocking arcs

$\mathsf{PN} + \mathsf{NBA}$

At most one token gets moved from the source to the destination

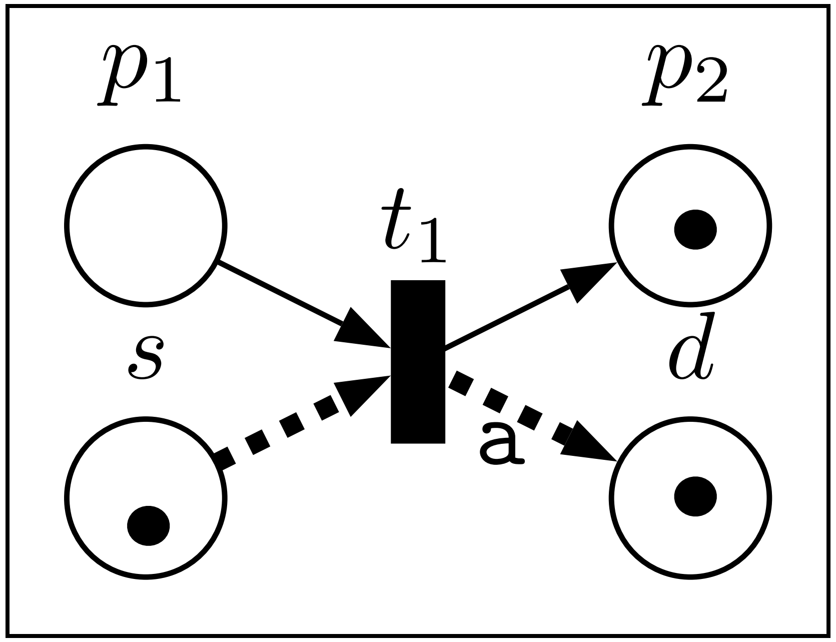


Example of PN+NBA

Non-blocking arcs

$\text{PN} + \text{NBA}$

At most one token gets moved from the source to the destination

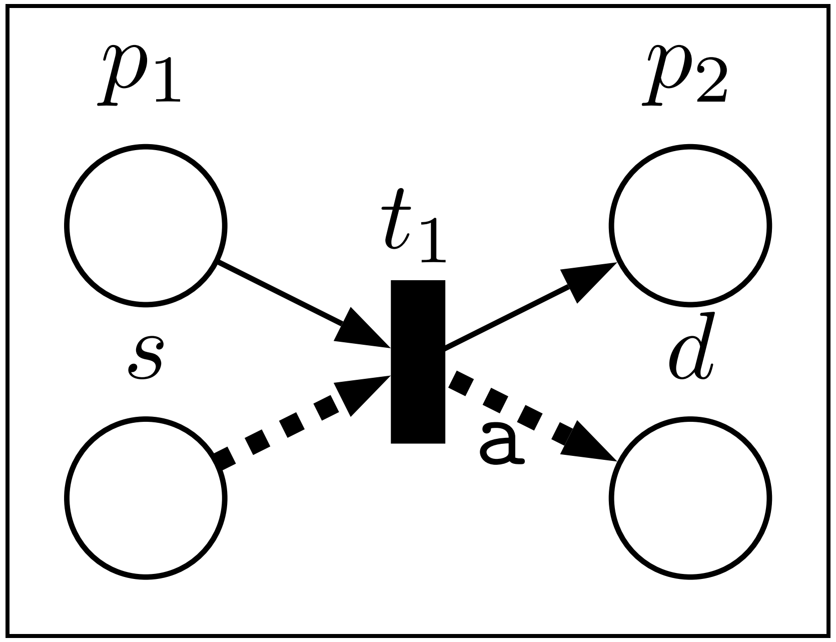


Example of PN+NBA

Non-blocking arcs

$\mathsf{PN} + \mathsf{NBA}$

At most one token gets moved from the source to the destination

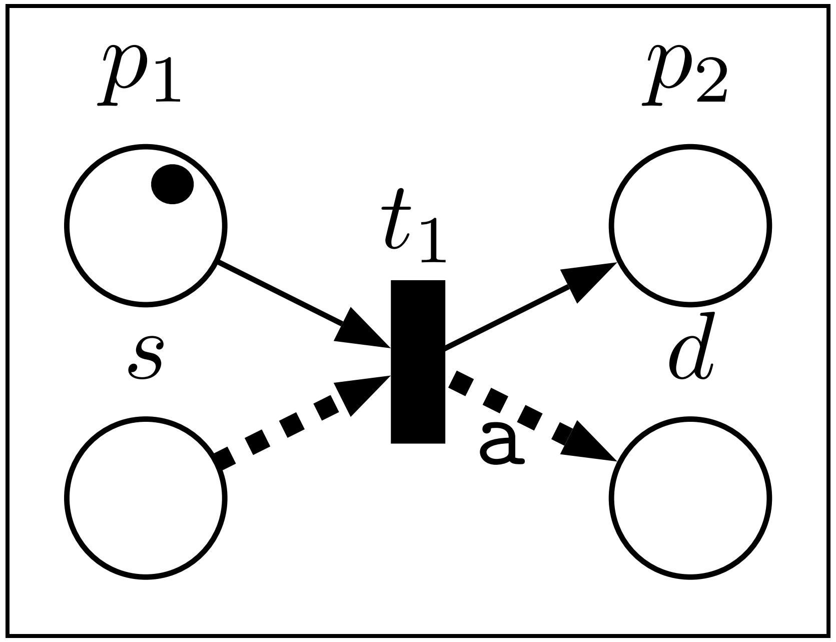


Example of PN+NBA

Non-blocking arcs

PN $+$ NBA

At most one token gets moved from the source to the destination

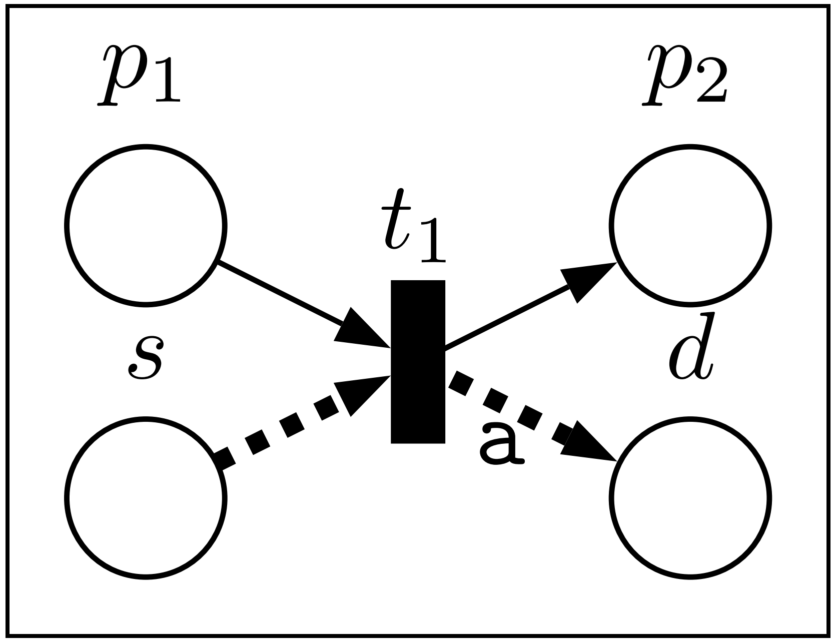


Example of PN+NBA

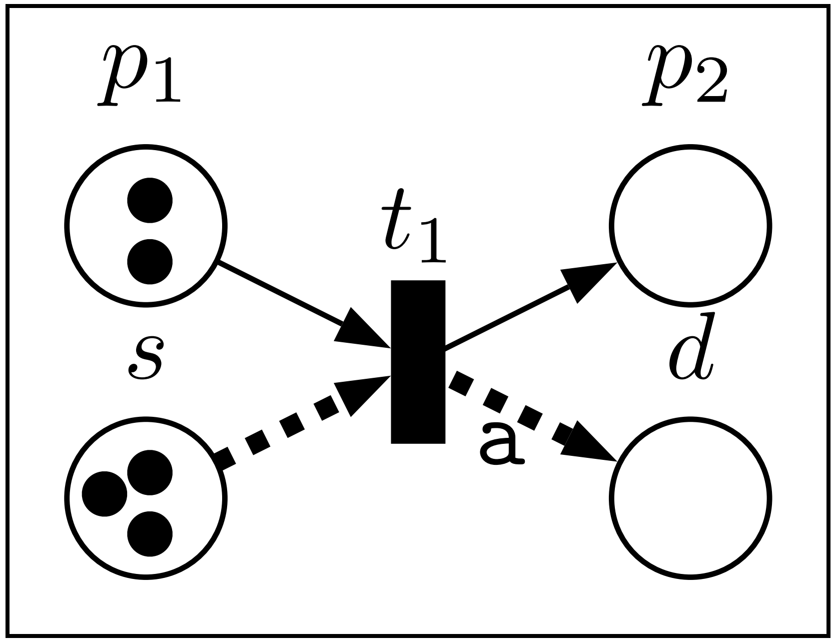
Non-blocking arcs

PN + NBA

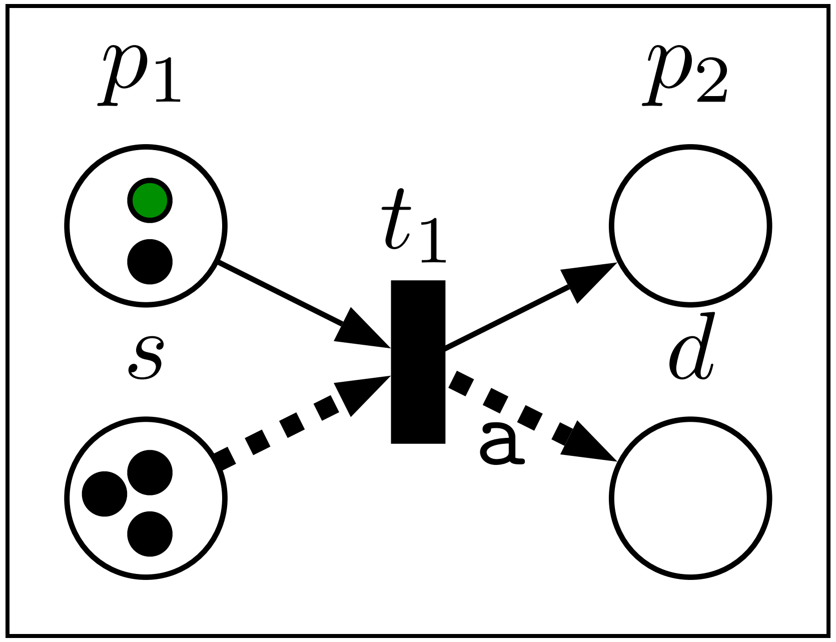
At most one token gets moved from the source to the destination



Example of PN+NBA

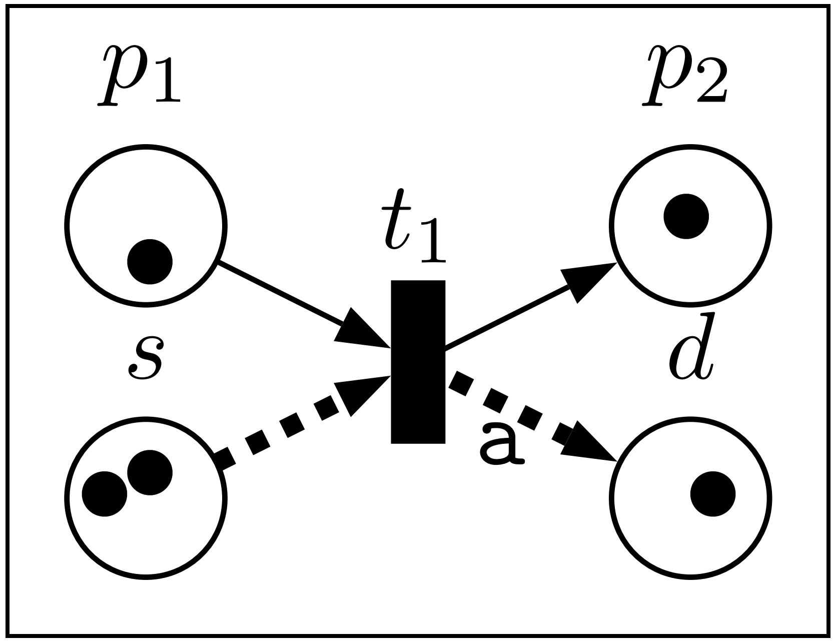


Example of PN+NBA



t₁ can be fired in this marking

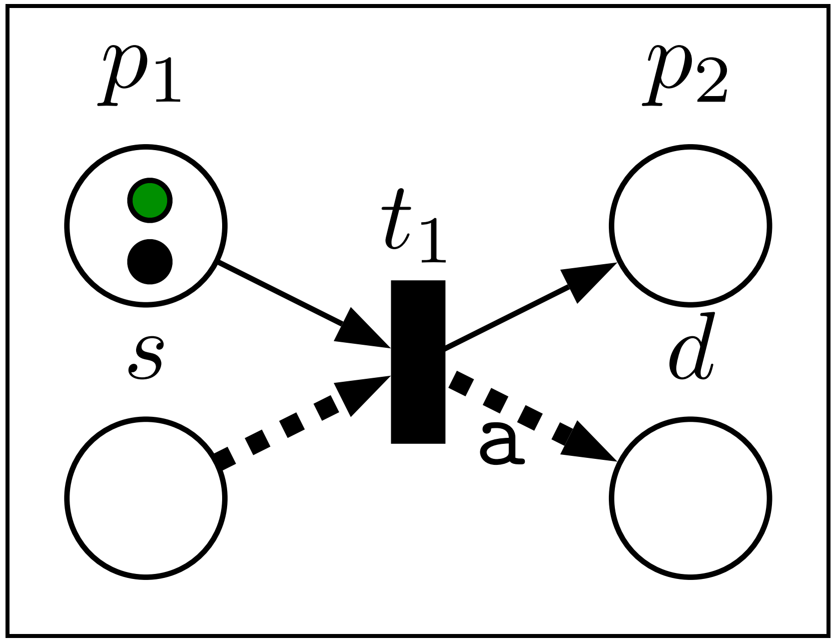
Example of PN+NBA



t_1 can be fired in this marking

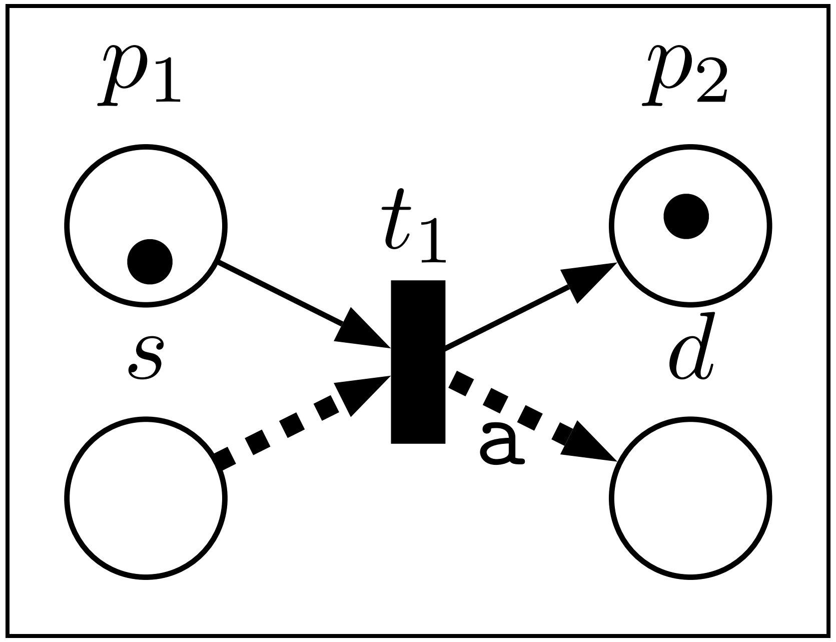
Firing t_1 removes one token in p_1, one token in s, add one token to p_2 and one token to d.

Example of PN+NBA



t_1 can be fired in this marking

Example of PN+NBA

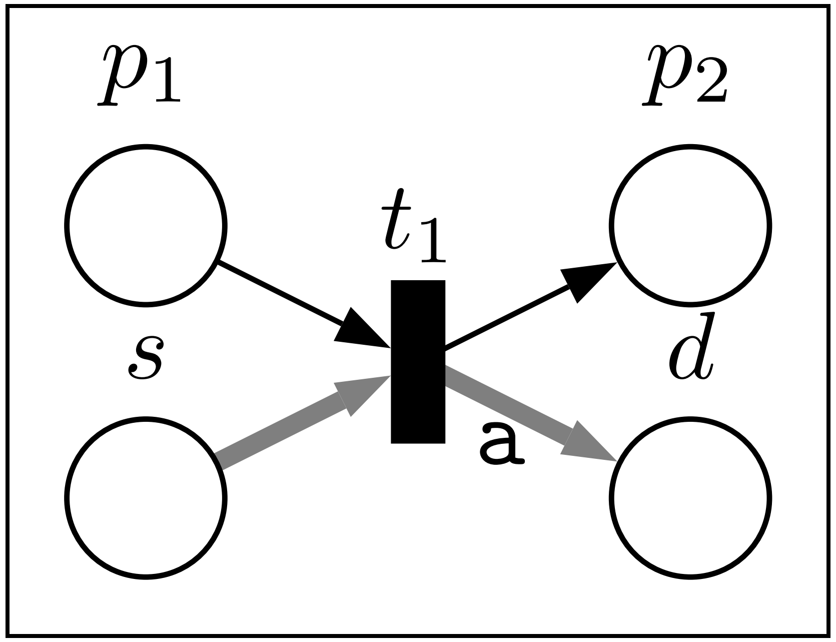


t_1 can be fired in this marking

Firing t_1 removes one token in p_1, add one token to p_2.

Example of PN+T

Example of PN+T

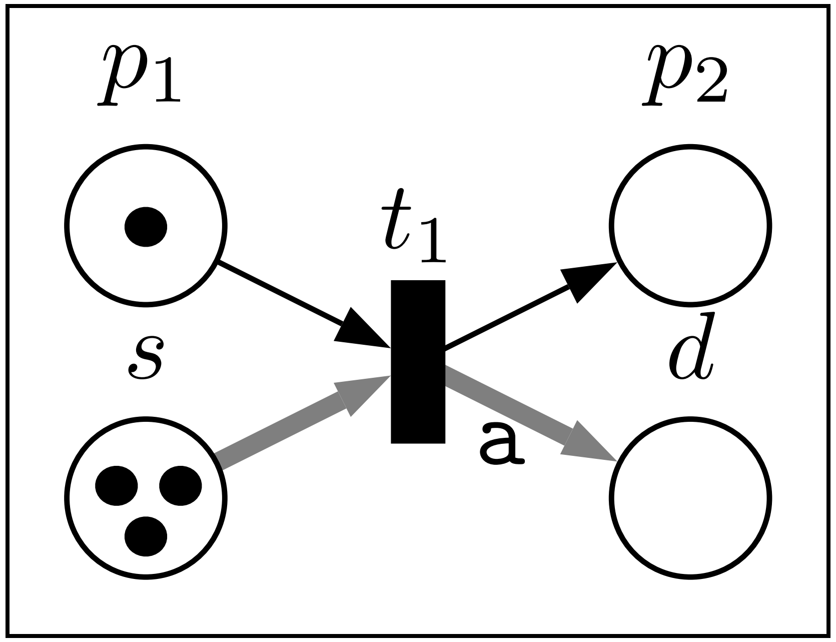


Transfer arcs

PN + T

All the tokens are moved from the source to the destination

Example of PN+T

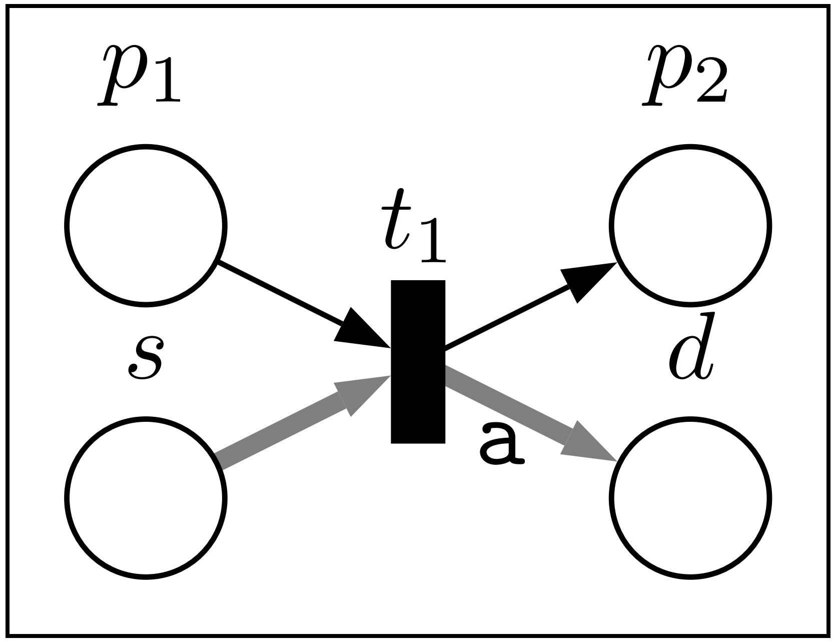


Transfer arcs

PN $+$ T

All the tokens are moved from the source to the destination

Example of PN+T

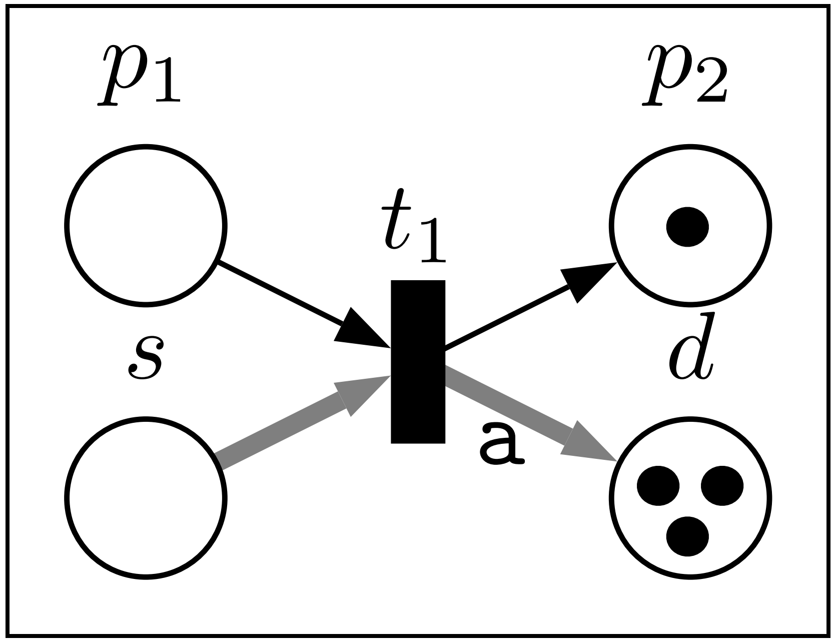


Transfer arcs

PN + T

All the tokens are moved from the source to the destination

Example of PN+T

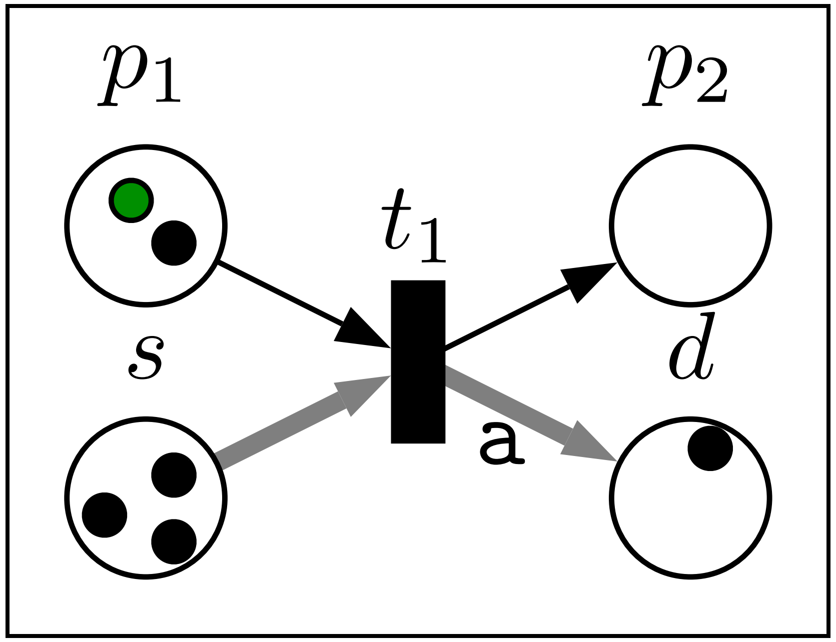


Transfer arcs

PN + T

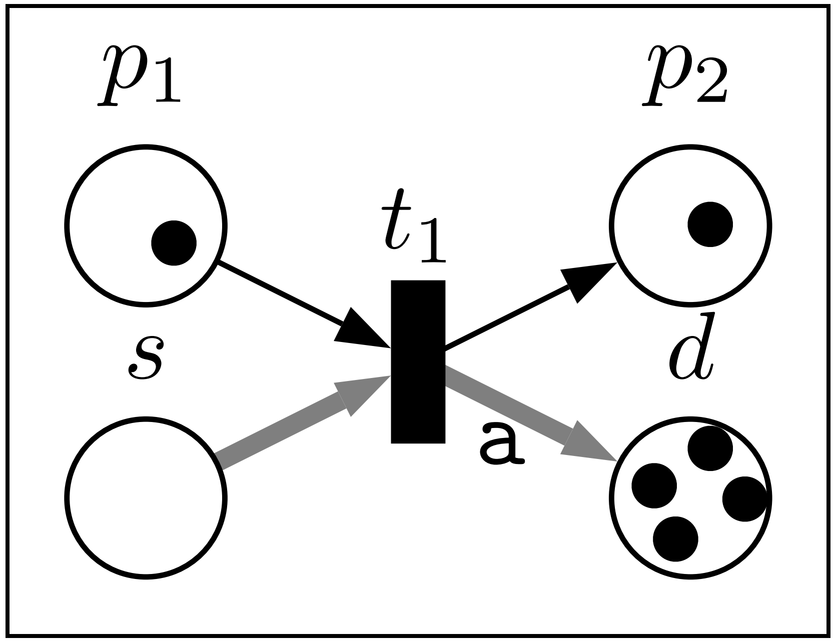
All the tokens are moved from the source to the destination

Example of PN+T



t₁ can be fired in this marking

Example of PN+T



t_1 can be fired in this marking

When firing t_1, one token is removed from p1 and added to p_2, and all the tokens in s are transfered to d.

Semantics of PN

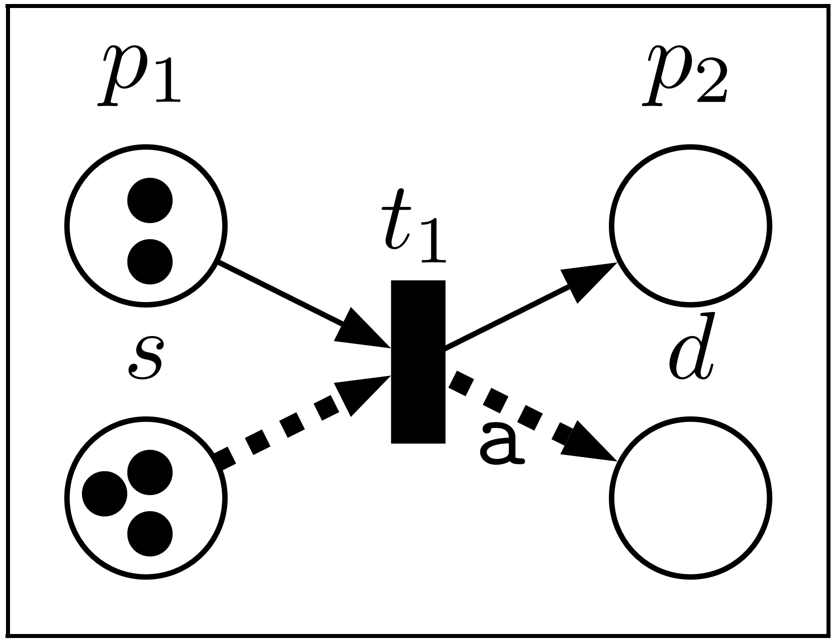
- Let N=(P,T,m0) be a Petri net.
- Its semantics is given by the following transition system Tr(N)=(C,c_0,⟹) where:
    - C={ m | m : P →$\mathbb{N}$ }
    - $c_0=m_0$
    - for all $m_1,m_2 \in C$, $m_1 \Longrightarrow m_2$ iff there exists t=(I,O)∈T:
        - $I \leq m_1$ and
        - $m_2=m_1-I+O$.

Semantics of Extended Petri nets

- Let $N=(P,T,m_0)$ be an extended Petri net.
- Its semantics is given by the following transition system $Tr(N)=(C,c_0,\Longrightarrow)$ where: $C=\{ m \mid m : P \rightarrow \mathbb{N} \}$, $c_0=m_0$, and:
  - for all $m,m' \in C$, $m \Longrightarrow m'$ iff there exists $t=(I,O,s,d,b) \in T$ and $I \leq m$, and m' is computed as follows: let $m_1=m-I$
    - Compute m_2 as follows: if $s=d=\perp$ then $m_2=m_1$ otherwise m_2 agrees with m_1 on all places but s and d where:
      - $m_2(s)=\max(0,m_1(s)-b)$
      - $m_2(d)=\min(m_1(d)+m_1(s),m_1(d)+b)$
    - Finally $m'=m_2+O$

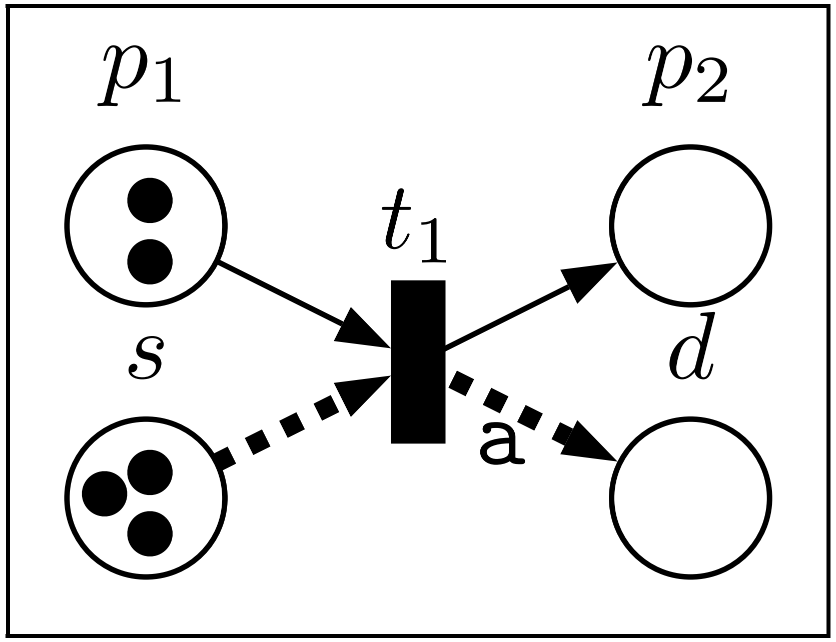
EPN are WSTS

- Let N=(P,T,m_0) be an extended Petri net. Its transition system Tr(N)=(C,c_0,⟹) is a WSTS (C,c_0,⟹,≼), where:
  - ≼ is the extension of $\leq \subseteq \mathbb{N} \times \mathbb{N}$ to tuples in $\mathbb{N}^{|P|}$, it is a WQO.
  - and ⟹ is monotonic w.r.t. ≼.



EPN are WSTS

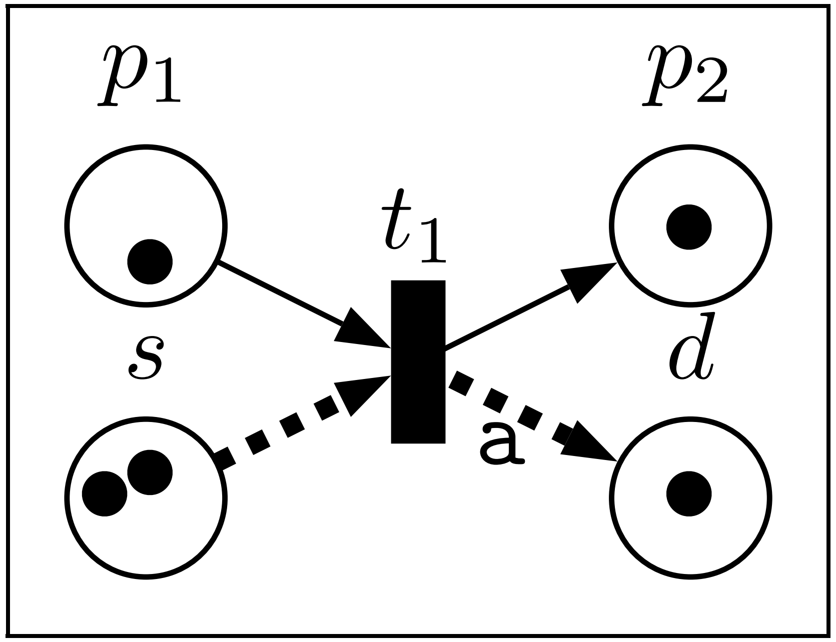
- Let N=(P,T,m_0) be an extended Petri net. Its transition system Tr(N)=(C,c_0,⟹) is a WSTS (C,c_0,⟹,≼), where:
  - ≼ is the extension of $\leq \subseteq \mathbb{N}\times\mathbb{N}$ to tuples in $\mathbb{N}^{|P|}$, it is a WQO.
  - and ⟹ is monotonic w.r.t. ≼.



m_I=(2,0,3,0)

EPN are WSTS

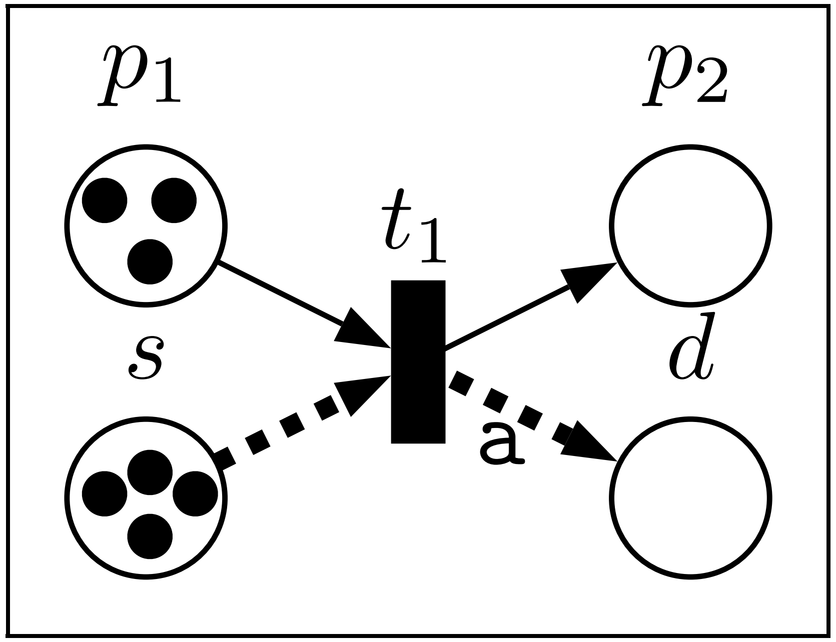
- Let $N=(P,T,m_0)$ be an extended Petri net. Its transition system $Tr(N)=(C,c_0,\Longrightarrow)$ is a WSTS $(C,c_0,\Longrightarrow,\preccurlyeq)$, where:
  - $\preccurlyeq$ is the extension of $\leq \subseteq \mathbb{N} \times \mathbb{N}$ to tuples in $\mathbb{N}^{|P|}$, it is a WQO.
  - and $\Longrightarrow$ is monotonic w.r.t. $\preccurlyeq$.



$m_1=(2,0,3,0) \longrightarrow m_2=(1,1,2,1)$

EPN are WSTS

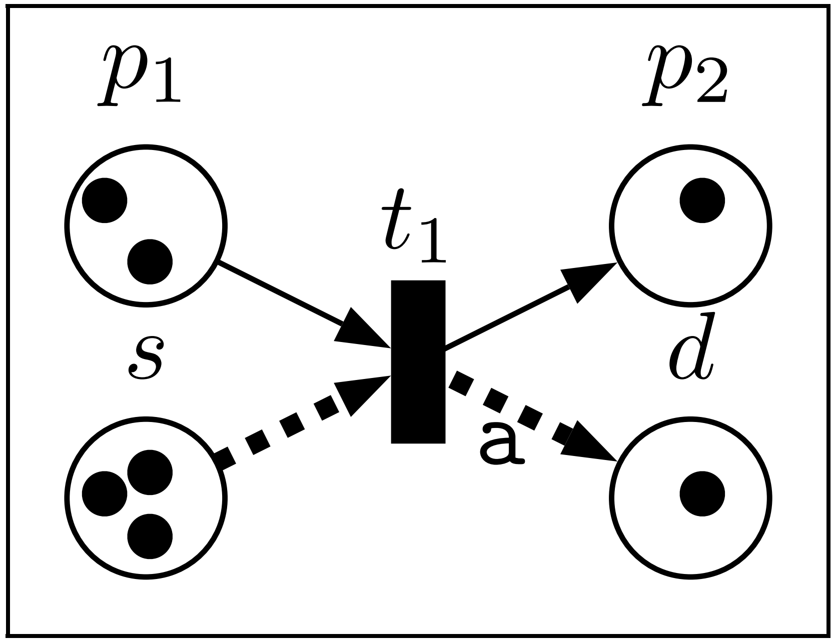
- Let N=(P,T,m_0) be an extended Petri net. Its transition system Tr(N)=(C,c_0,⟹) is a WSTS (C,c_0,⟹,≼), where:
  - ≼ is the extension of $\leq \subseteq \mathbb{N}\times\mathbb{N}$ to tuples in $\mathbb{N}^{|P|}$, it is a WQO.
  - and ⟹ is monotonic w.r.t. ≼.



$$\begin{tikzcd} m_3=(3,0,4,0) \arrow[d, phantom, "\curlyvee"] & \\ m_1=(2,0,3,0) \arrow[r] & m_2=(1,1,2,1) \end{tikzcd}$$

EPN are WSTS

- Let N=(P,T,m_0) be an extended Petri net. Its transition system Tr(N)=(C,c_0,⟹) is a WSTS (C,c_0,⟹,≼), where:
  - ≼ is the extension of $\leq \subseteq \mathbb{N}\times\mathbb{N}$ to tuples in $\mathbb{N}^{|P|}$, it is a WQO.
  - and ⟹ is monotonic w.r.t. ≼.



$$\begin{tikzcd}
m_3=(3,0,4,0) \arrow[r] & m_4=(2,1,3,1) \\
m_1=(2,0,3,0) \arrow[r] \arrow[u, phantom, "\preccurlyeq" sloped] & m_2=(1,1,2,1) \arrow[u, phantom, "\preccurlyeq" sloped]
\end{tikzcd}$$

Properties of extended Petri nets

- The reachability problem asks given a net $N=(P,T,m_0)$ and a marking m, if $m \in Post^*(m_0)$.
- The coverability problem asks given a net $N=(P,T,m_0)$ and a marking m, if there exists a marking $m' \geqslant m$ such that $m' \in Post^*(m_0)$.
- The non-terminating computation problem asks given a net $N=(P,T,m_0)$ if there exists an infinite computation in N starting from m_0.
- The place boundedness problem asks given a net $N=(P,T,m_0)$ and a place $p \in P$ if there exists a bound $n \in \mathbb{N}$ such that for all $m \in Reach(m_0)$, we have that $m(p) \leq n$.

Reachability is undecidable for EPN

Theorem. The reachability problem for PN+NBA (and for PN+T) is undecidable.

Reachability is undecidable for EPN

Theorem. The reachability problem for PN+NBA (and for PN+T) is undecidable.

Proof sketch. Given a 2CM machine M, we can construction a PN+NBA N and two markings m_0,m_1 such that m_1 is reachable from m_0 in N iff the machine M halts.

We associate to each counter and each control state of the 2CM a place of the net. We have an additional place p_{check}.

Initially, the place associated to the initial control state contains one token, all the other places (incluing p_{check} and the two counters) are empty.

Reachability is undecidable for EPN

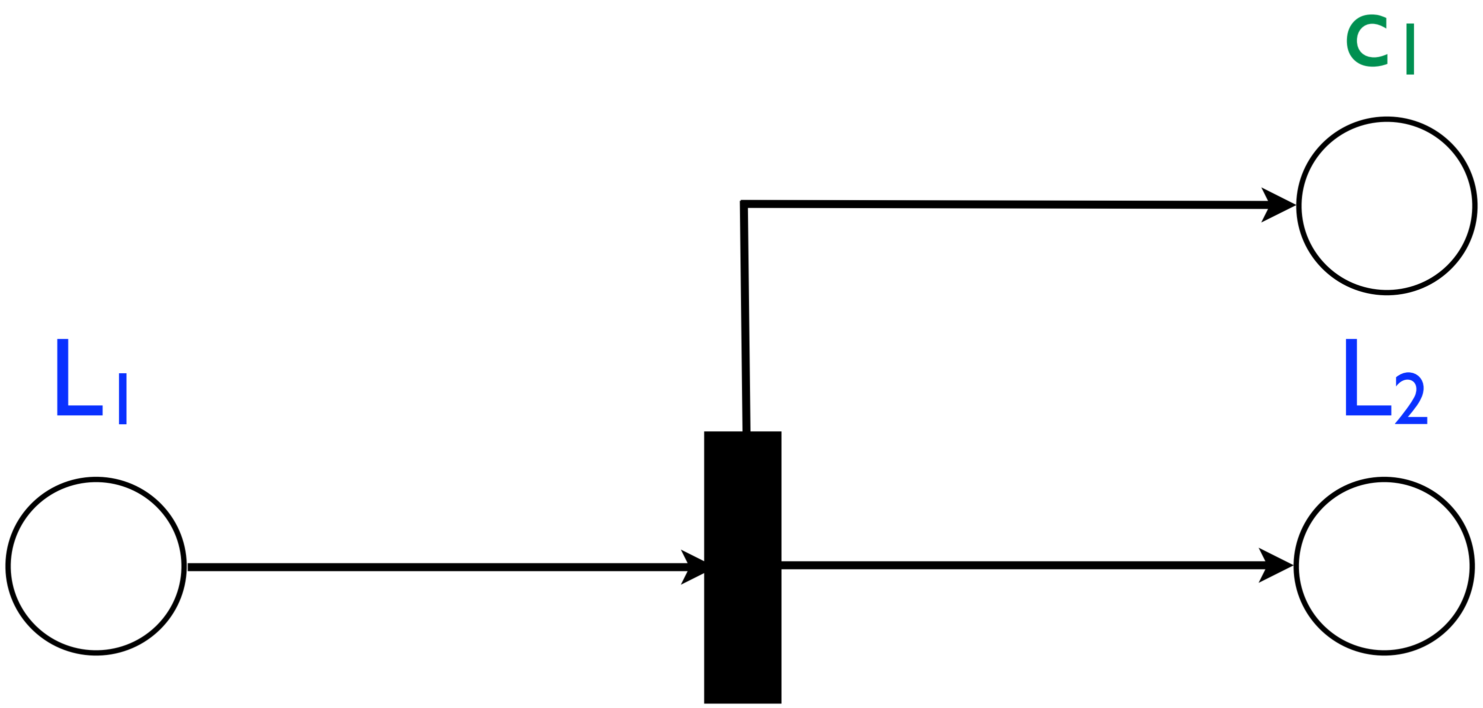
Theorem. The reachability problem for PN+NBA (and for PN+T) is undecidable.

Simulation of the instructions of a 2CM.

Reachability is undecidable for EPN

Theorem. The reachability problem for PN+NBA (and for PN+T) is undecidable.

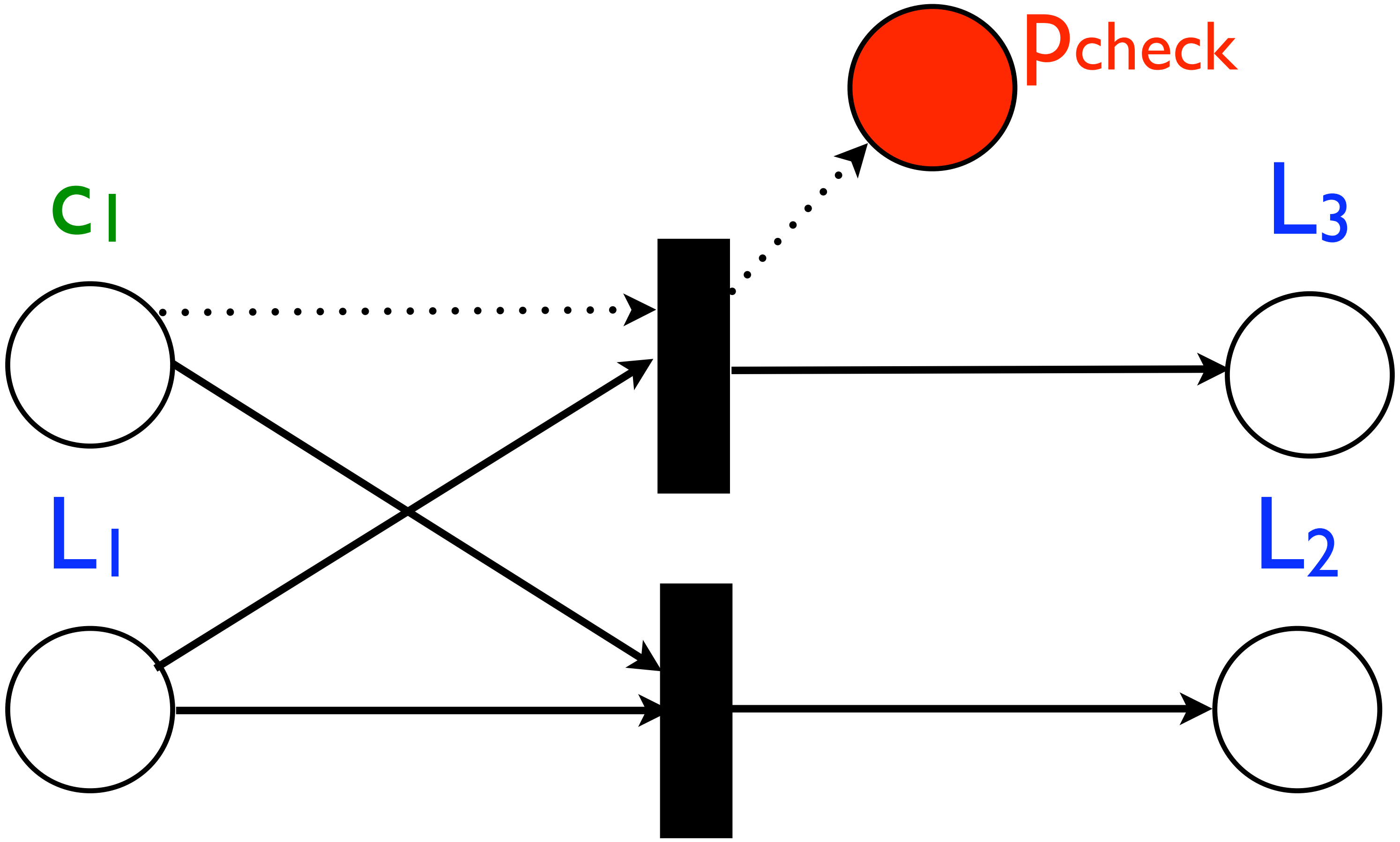
L_1: $c_1 := c_1 + 1$; goto L_2.



Reachability is undecidable for EPN

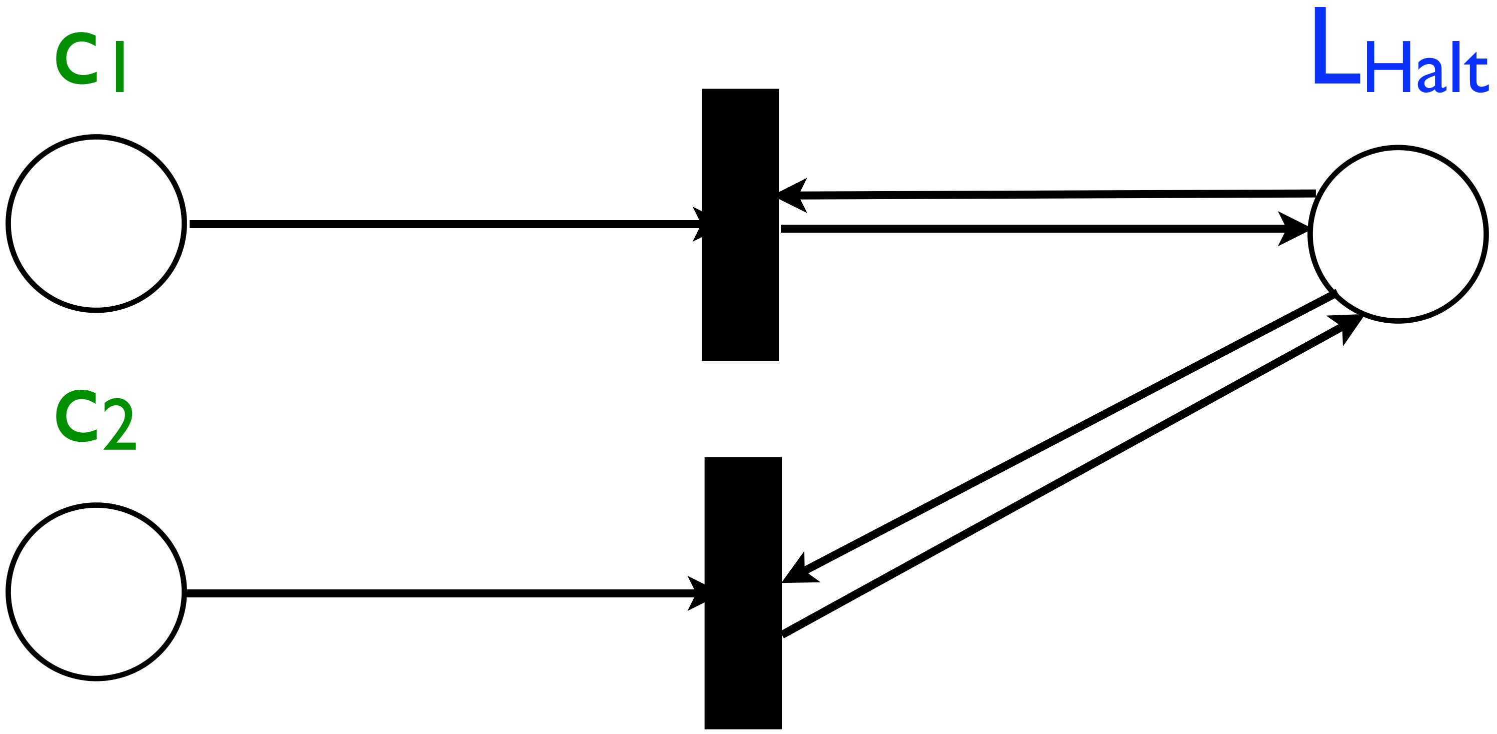
Theorem. The reachability problem for PN+NBA (and for PN+T) is undecidable.

L_1: if $c_1 \neq 0$ then $c_1 := c_1 - 1$; goto L_2 else goto L_3.



Reachability is undecidable for EPN

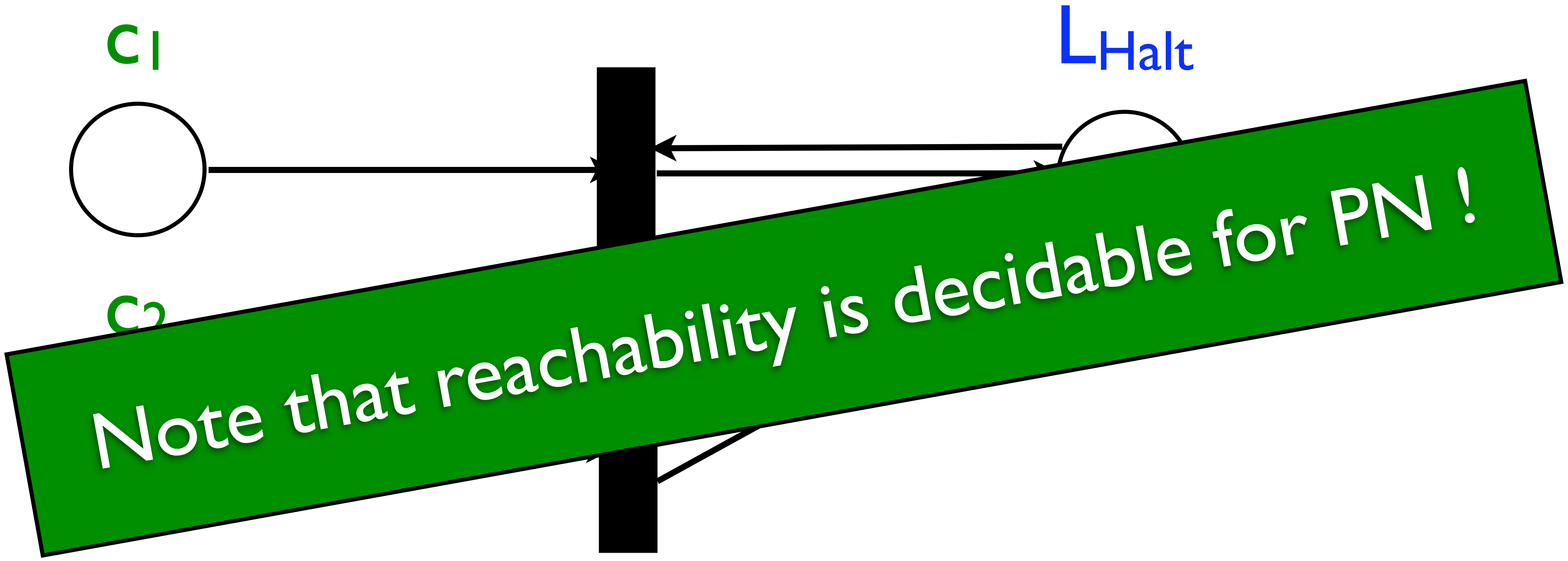
Theorem. The reachability problem for PN+NBA (and for PN+T) is undecidable.



With this additional gadget, it is clear that the machine M halts **iff** the marking "one token in halt and all other places empty" is reachable for the initial marking.

Reachability is undecidable for EPN

Theorem. The reachability problem for PN+NBA (and for PN+T) is undecidable.



Note that reachability is decidable for PN !

With this additional gadget, it is clear that the machine M halts **iff** the marking "one token in halt and all other places empty" is reachable for the initial marking.

Place boundedness

Theorem. The place boundedness problems for PN+NBA and PN+T are undecidable.

Place boundedness

Theorem. The place boundedness problems for PN+NBA and PN+T are undecidable.

To prove that we need a non-trivial extension of the proof idea in the previous undecidability result.

Three algorithmic techniques for WSTS

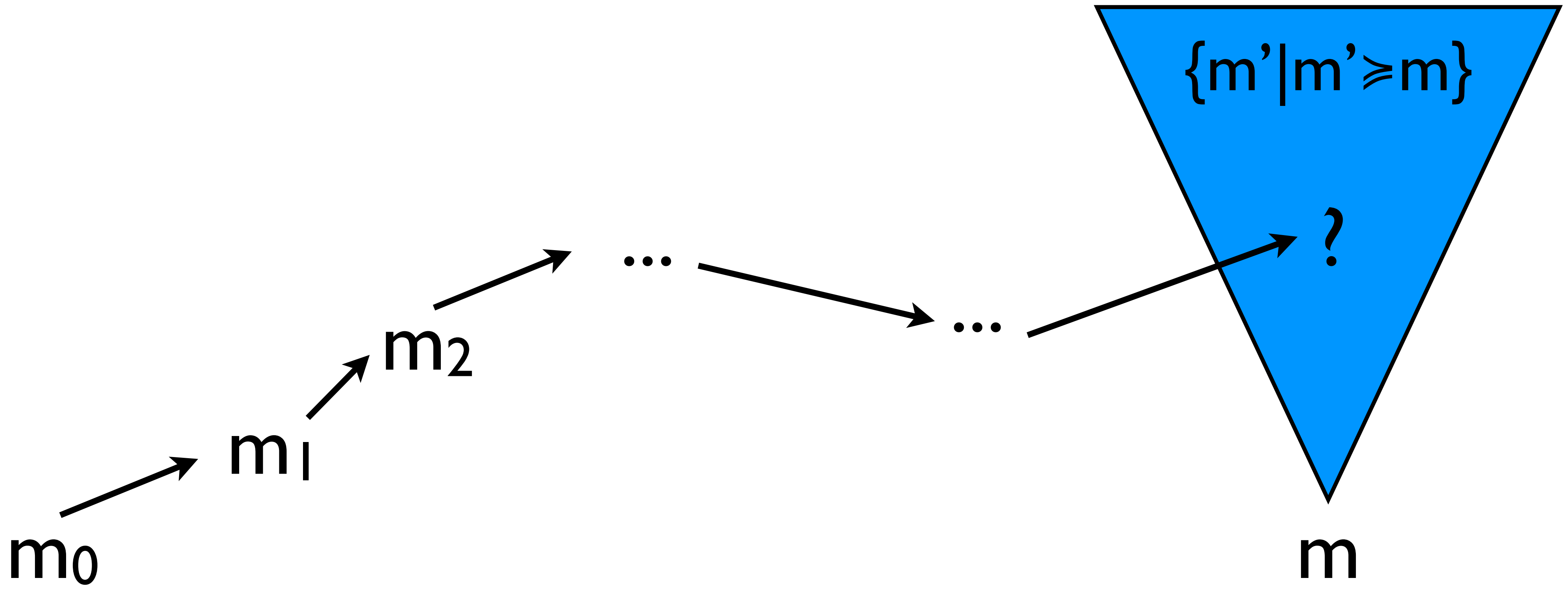
Technique 1:
set saturation

Backward algorithm for coverability

- The coverability problem asks given a net $N=(P,T,m_0)$ and a marking m, if there exists a marking $m' \geqslant m$ such that $m' \in Post^*(m_0)$.

Backward algorithm for coverability

- The coverability problem asks given a net $N=(P,T,m_0)$ and a marking m, if there exists a marking $m' \geqslant m$ such that $m' \in Post^*(m_0)$.

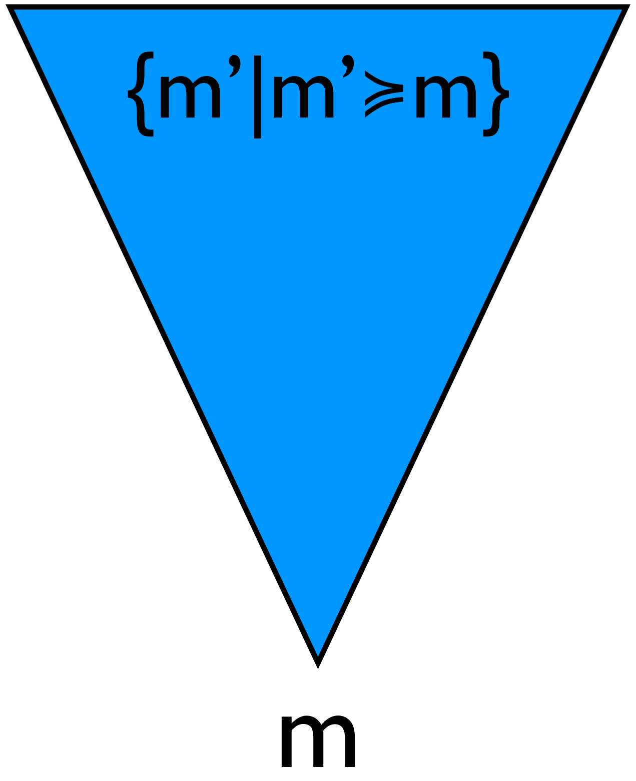


Backward algorithm for coverability

- The coverability problem asks given a net $N=(P,T,m_0)$ and a marking m, if there exists a marking $m' \geqslant m$ such that $m' \in Post^*(m_0)$.

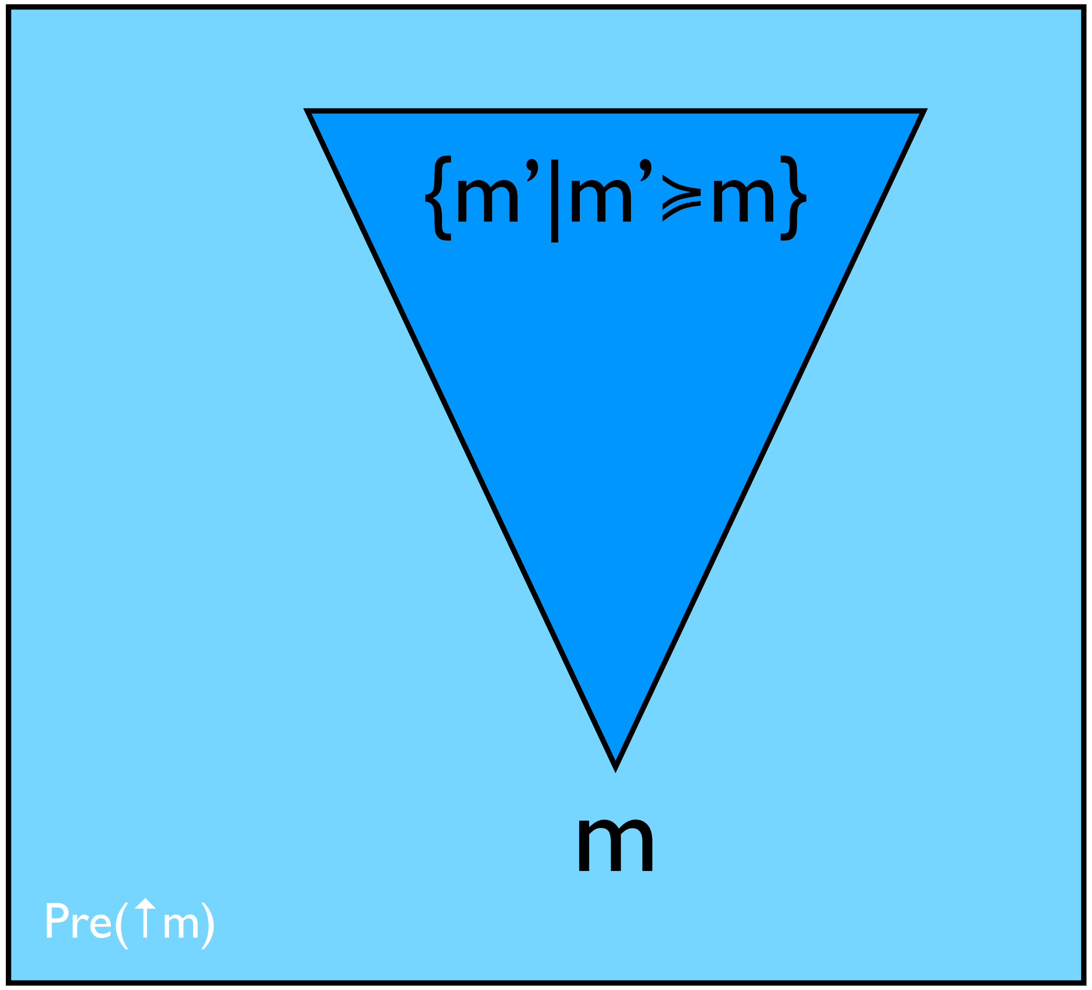
Backward algorithm for coverability

- The coverability problem asks given a net $N=(P,T,m_0)$ and a marking m, if there exists a marking $m' \geqslant m$ such that $m' \in Post^*(m_0)$.



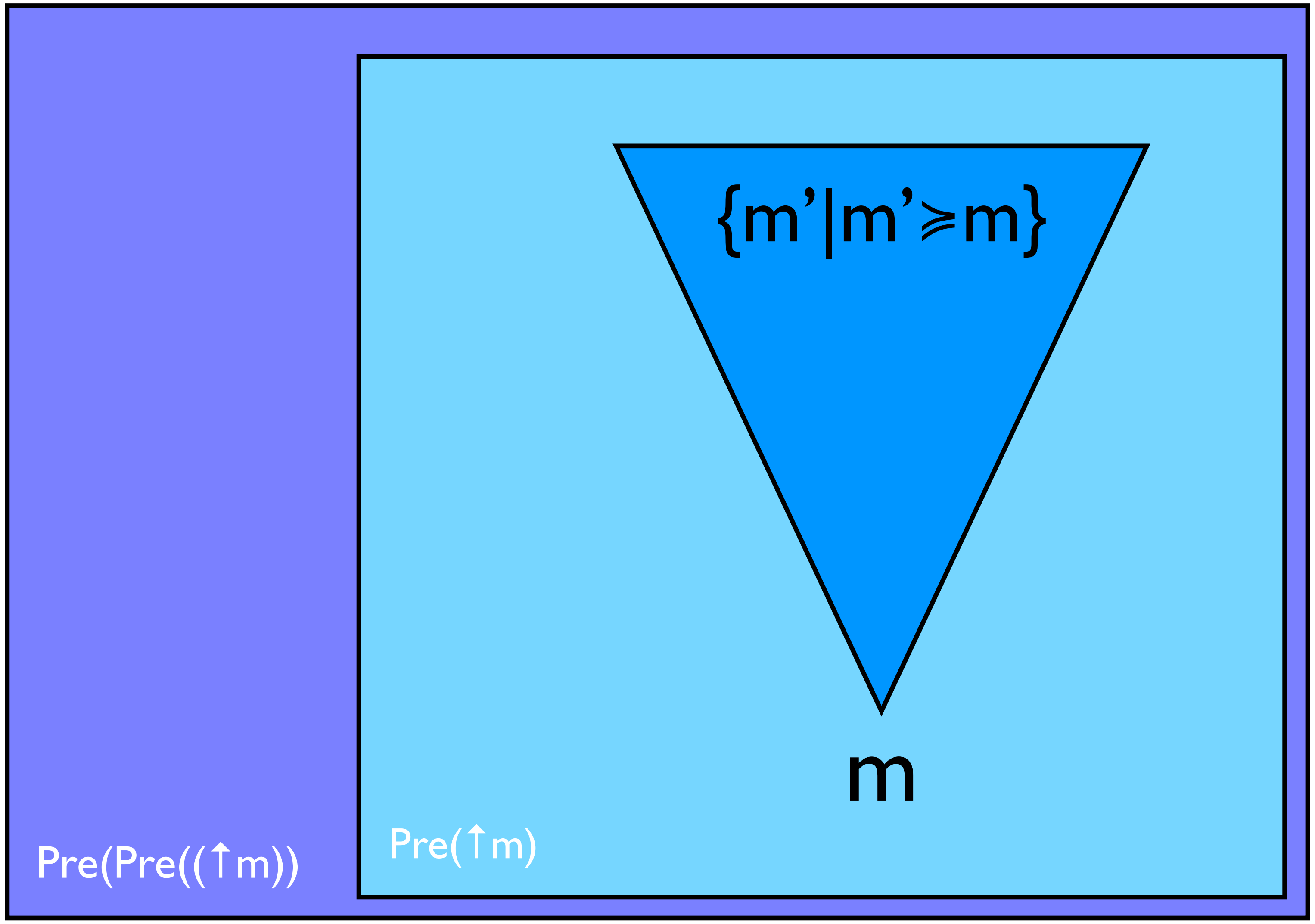
Backward algorithm for coverability

- The coverability problem asks given a net $N=(P,T,m_0)$ and a marking m, if there exists a marking $m' \geqslant m$ such that $m' \in Post^*(m_0)$.



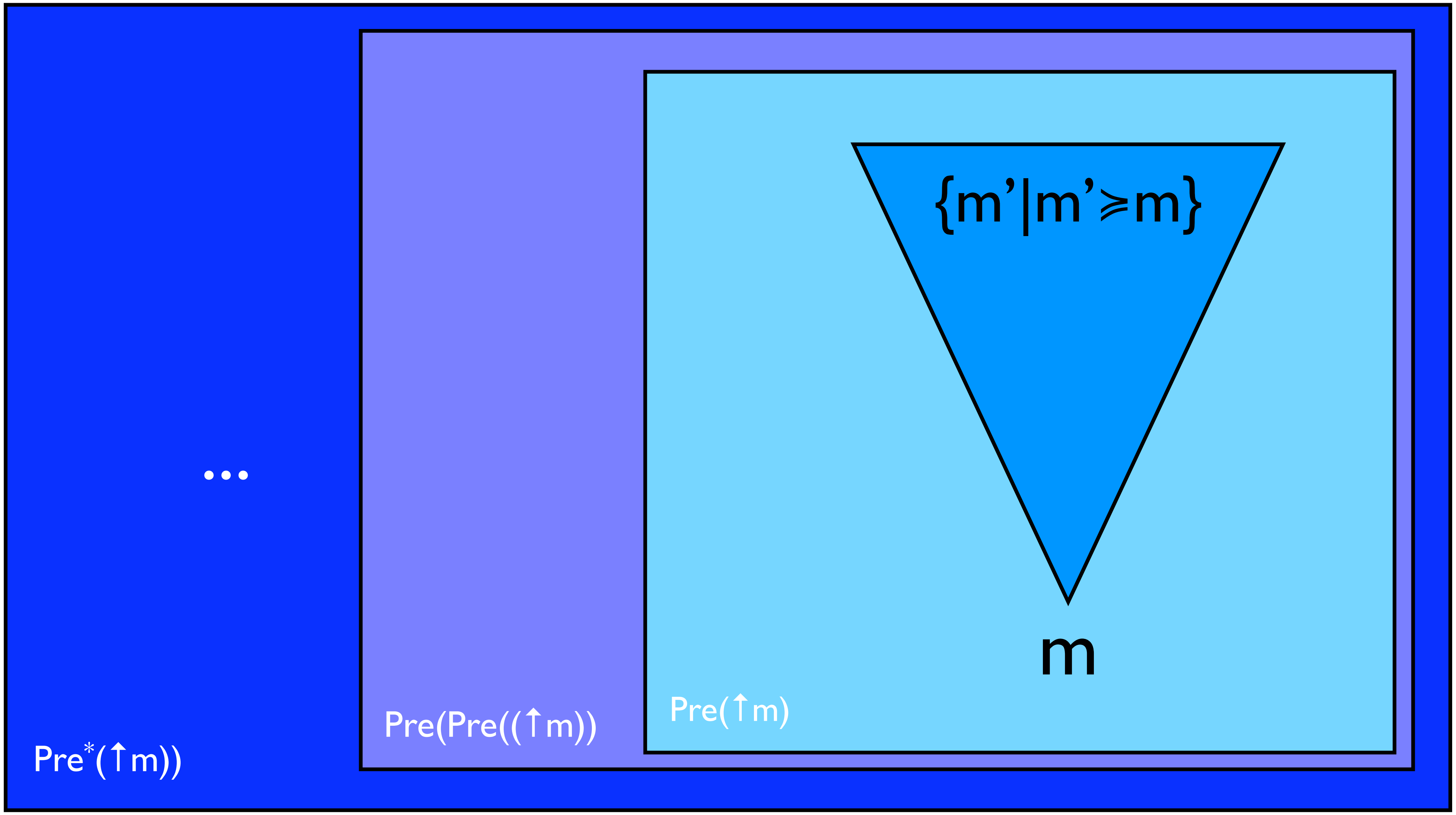
Backward algorithm for coverability

- The coverability problem asks given a net N=(P,T,m_0) and a marking m, if there exists a marking m'⩾m such that m'∈Post*(m_0).



Backward algorithm for coverability

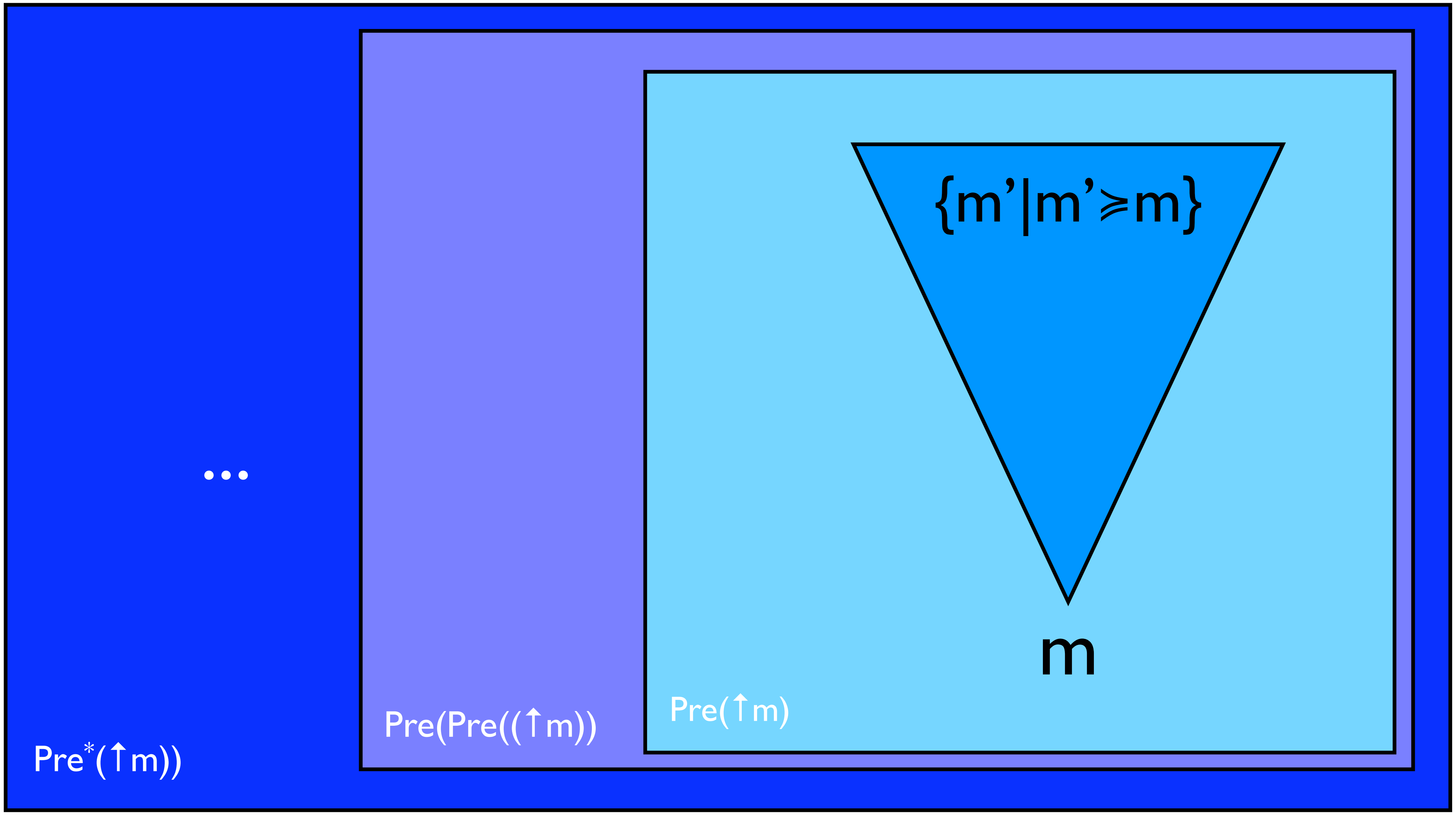
- The coverability problem asks given a net $N=(P,T,m_0)$ and a marking m, if there exists a marking $m' \geqslant m$ such that $m' \in Post^*(m_0)$.



Backward algorithm for coverability

- The coverability problem asks given a net $N=(P,T,m_0)$ and a marking m, if there exists a marking $m' \geqslant m$ such that $m' \in Post^*(m_0)$.

$m_0 \in^?$



Pre and upward-closed sets in WSTS

- **Lemma**. Let $T=(C,c_0,\Longrightarrow,\leq)$ be a WSTS and U be an $\leq$-upward closed set of configurations in T.
  Pre(U) is $\leq$-upward closed.

Pre and upward-closed sets in WSTS

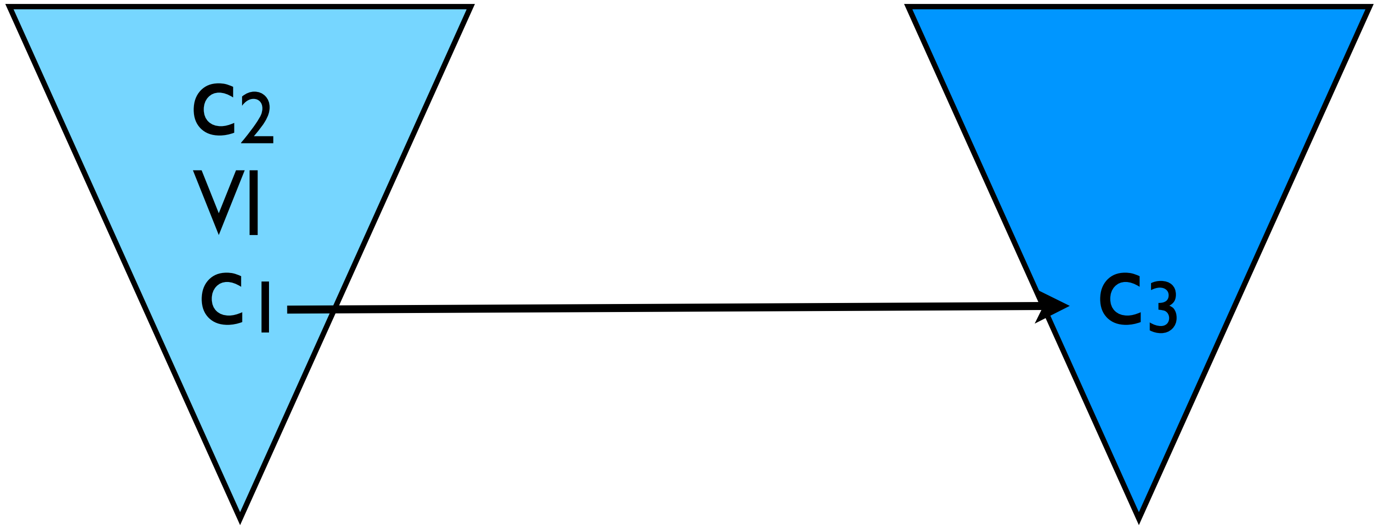
- **Lemma**. Let $T=(C,c_0,\Longrightarrow,\leq)$ be a WSTS and U be an $\leq$-upward closed set of configurations in T.
  Pre(U) is $\leq$-upward closed.

Proof. Let $c_1 \in Pre(U)$ and let us consider any c_2 such that $c_1 \leq c_2$.

We know that there exists $c_3 \in U$ and $c_1 \Longrightarrow c_3$.

By monotonicity, there exists c_4 such that $c_3 \leq c_4$ and $c_2 \Longrightarrow c_4$.

As U is upward closed, we have that $c_4 \in U$ and so $c_2 \in Pre(U)$.



Pre and upward-closed sets in WSTS

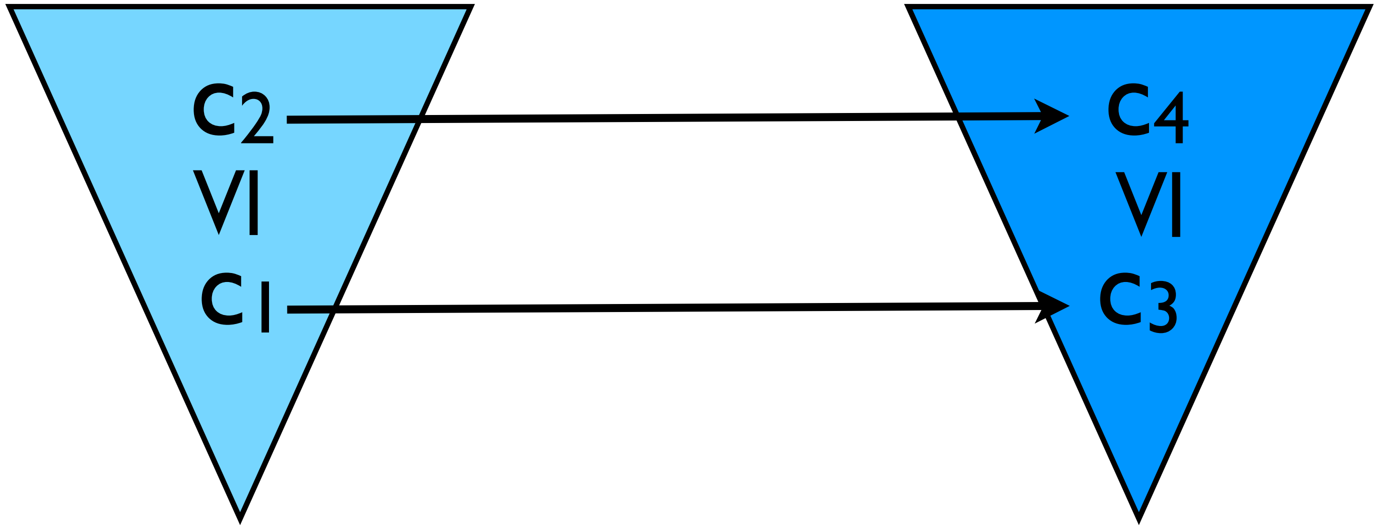
- **Lemma**. Let T=(C,c_0,⟹,≤) be a WSTS and U be an ≤-upward closed set of configurations in T.
  Pre(U) is ≤-upward closed.

Proof. Let $c_1 \in Pre(U)$ and let us consider any c_2 such that $c_1 \leq c_2$.

We know that there exists $c_3 \in U$ and $c_1 \Longrightarrow c_3$.

By monotonicity, there exists c_4 such that $c_3 \leq c_4$ and $c_2 \Longrightarrow c_4$.

As U is upward closed, we have that $c_4 \in U$ and so $c_2 \in Pre(U)$.



Effective WSTS

- PreUp(c) is the set of all configurations whose one-step successors by $\Longrightarrow$ are larger or equal to c i.e.:

  PreUp(c)={ c' | ∃ c'' : c' $\Longrightarrow$ c'' and c≤c'' }=Pre(↑c)

- A WSTS T=(C,c0,$\Longrightarrow$,≤) is effective (EWSTS) if:
  - given any pair of configurations c_1 and c_2 in C, one can decide if $c_1 \Longrightarrow c_2$ or not.
  - given any pair of configurations c_1 and c_2 in C, one can decide if $c_1 \leq c_2$ or not.
  - given any configuration c∈C, one can effectively compute UGen(PreUp(c)).

- If the set of successors Post(c) of a configuration c is finite and effectively computable, we say that the WSTS is forward effective (FEWSTS for short).

General backward for solving coverability in EWSTS

- Let $T=(C,c0,\Longrightarrow,\leq)$ be EWSTS. Let $U\subseteq C$ be an upward closed set and UGen(U) a finite generator for U.
- Consider now the sequence:
  $E_0=\mathrm{UGen}(U)$
  $E_i=\mathrm{UGen}(\mathrm{PreUp}(E_{i-1}) \cup \uparrow E_{i-1}))$, for $i\geq 0$.
  - First, note that all elements of this sequence are computable as T is an EWSTS.
  - Second, $\uparrow E_i$ is the set of configurations of T that can reach a configuration in U in i steps or less.
  - Third, there exists a position $k\geq 0$ such that for all $l\geq k$, $\uparrow E_l=\uparrow E_k$.

Termination

Assume that this is not the case.

Then, as the sequence $\uparrow E_i$ is increasing for $\subseteq$, there must exist a sequence of elements

$e_1\ e_2\ \dots\ e_n\ \dots$

such that for all $i<j$, $\neg(e_i \leq e_j)$.

But this is in contradiction with the fact that $(S,\leq)$ is a well-quasi ordered set !

General backward for solving coverability in EWSTS

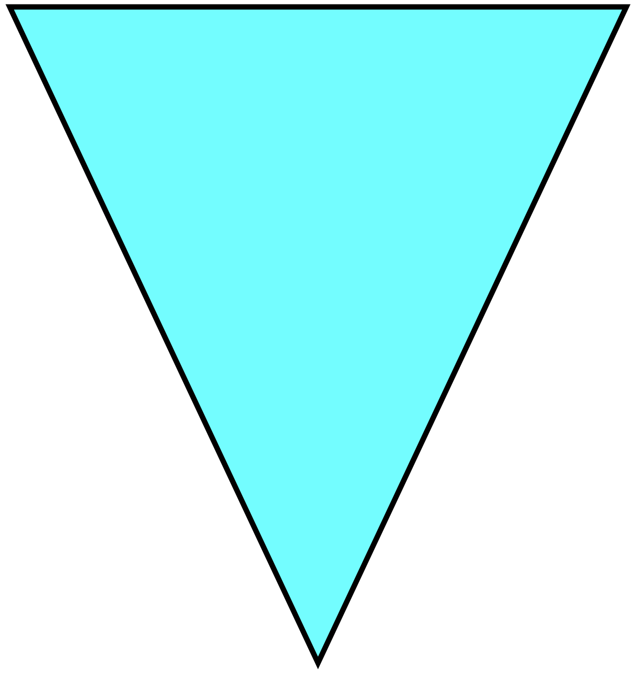
- Let $T=(C,c0,\Longrightarrow,\leq)$ be EWSTS. Let $U\subseteq C$ be an upward closed set and UGen(U) a finite generator for U.
- Consider now the sequence:
  E_0=UGen(U)
  E_i=UGen(PreUp(E_{i-1}) $\cup$ $\uparrow E_{i-1}$)), for $i\geq 0$.
  - First, note that all elements of this sequence are computable as T is an EWSTS.
  - Second, $\uparrow E_i$ is the set of configurations of T that can reach a configuration in U in i steps or less.
  - Third, there exists a position $k\geq 0$ such that for all $l\geq k$, $\uparrow E_l=\uparrow E_k$.
- This sequence is thus a **effective algorithm** to decide coverability in EWSTS.

Decidability of coverability for EWSTS

Theorem. The coverability problem is decidable for EWSTS.

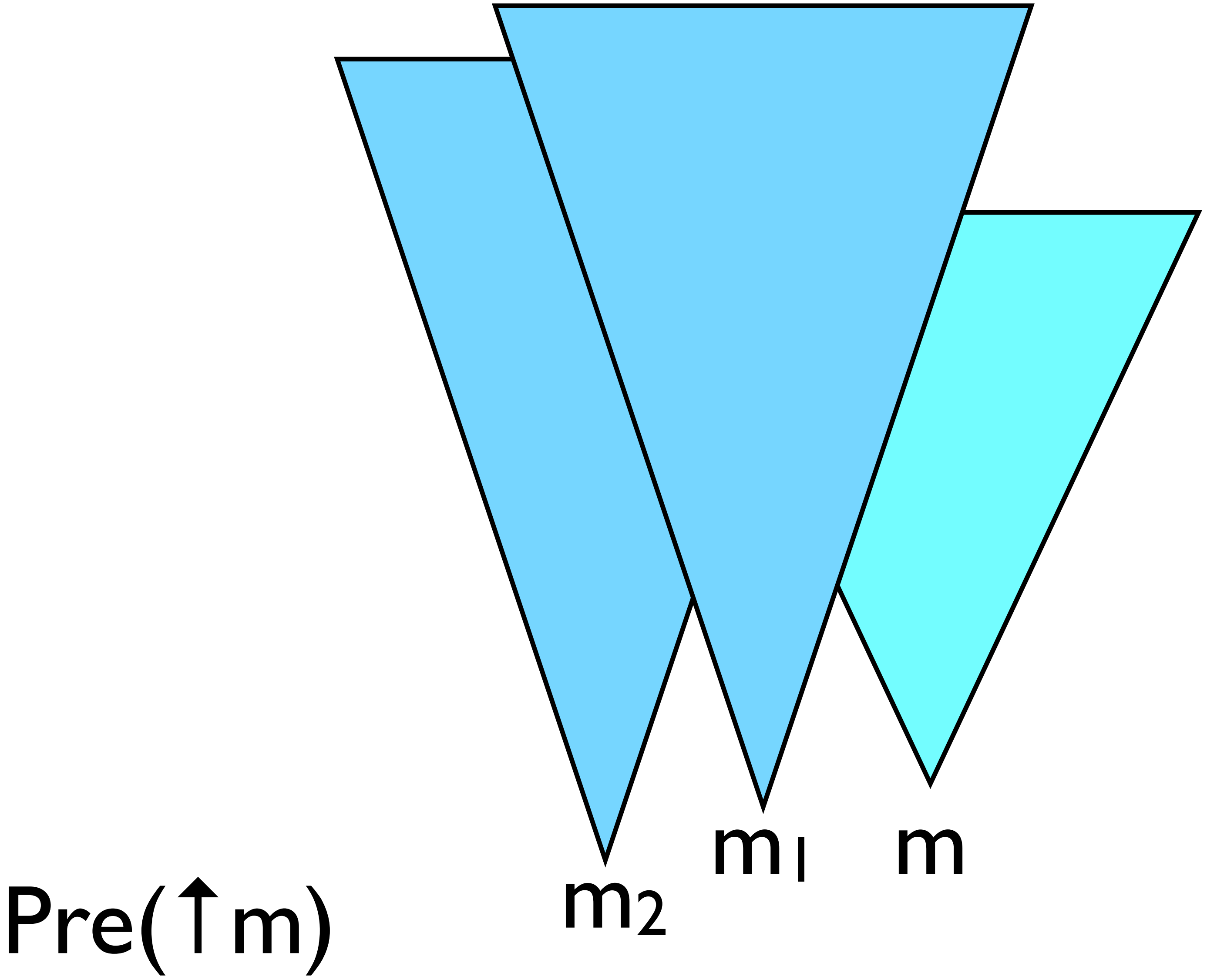
Backward algorithm for coverability

- The coverability problem asks given a net $N=(P,T,m_0)$ and a marking m, if there exists a marking $m' \geqslant m$ such that $m' \in Post^*(m_0)$.



Backward algorithm for coverability

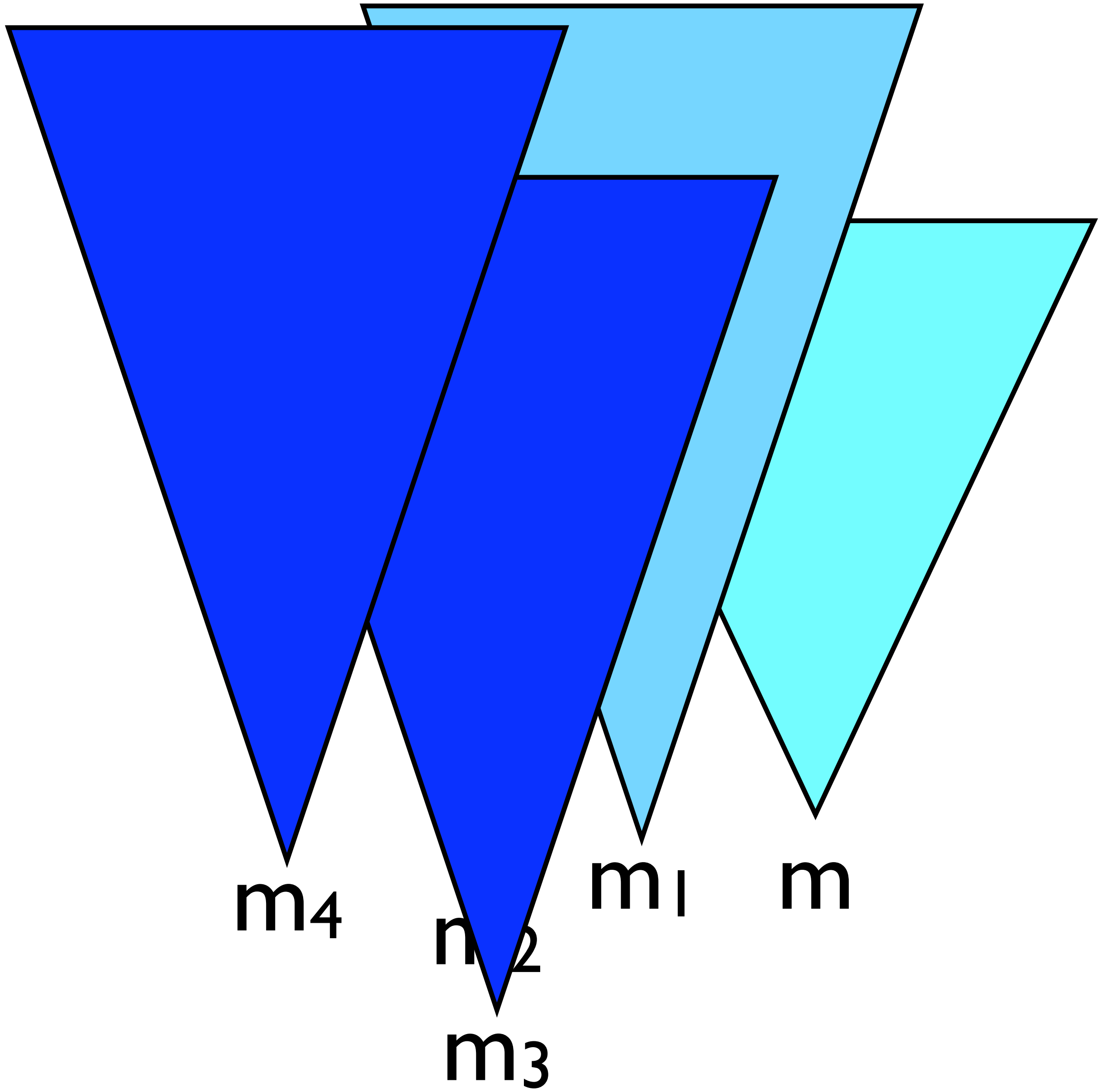
- The coverability problem asks given a net $N=(P,T,m_0)$ and a marking m, if there exists a marking $m' \geqslant m$ such that $m' \in Post^*(m_0)$.



Backward algorithm for coverability

- The coverability problem asks given a net $N=(P,T,m_0)$ and a marking m, if there exists a marking $m' \geqslant m$ such that $m' \in Post^*(m_0)$.

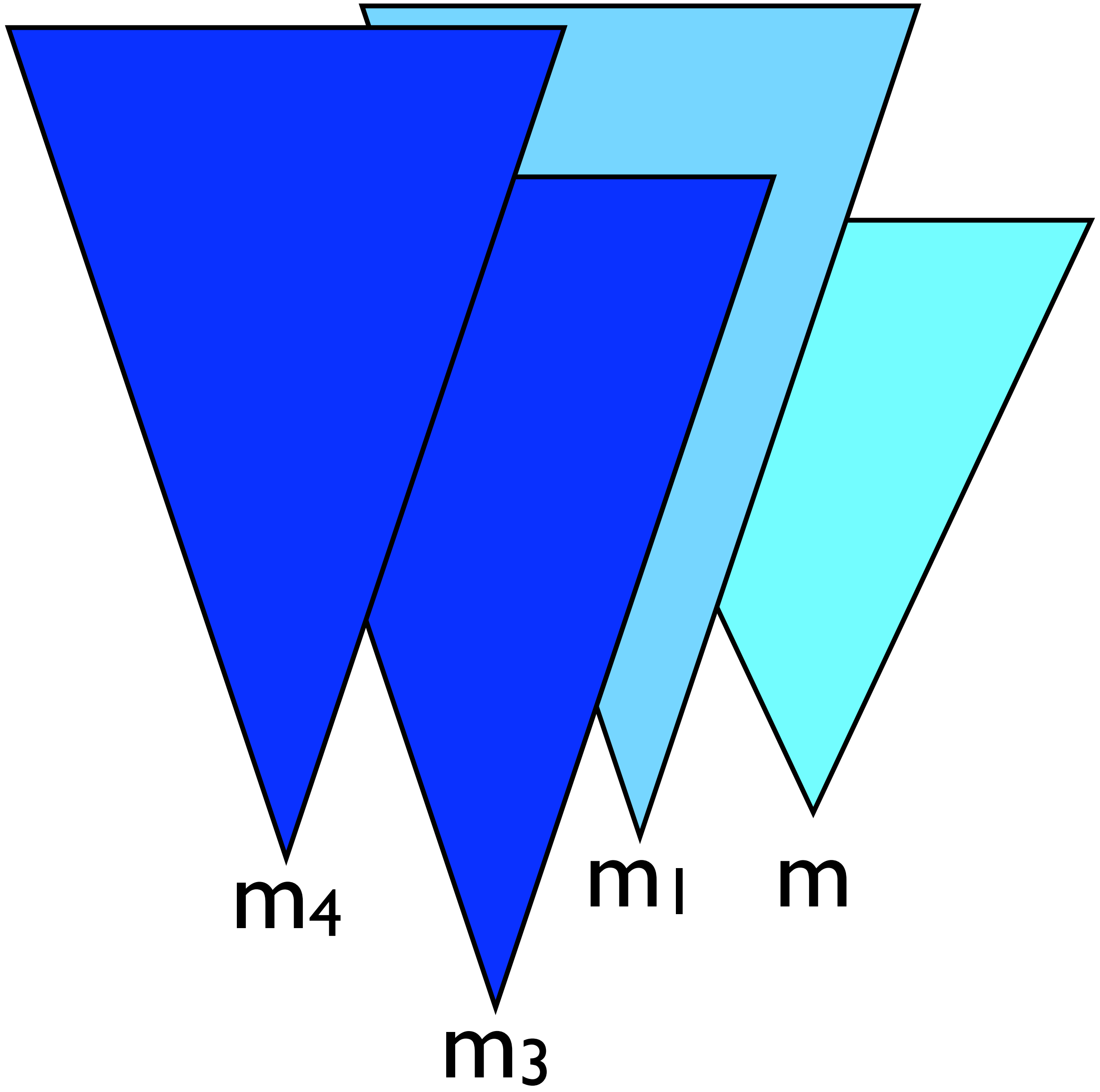
$Pre^2(\uparrow m)$



Backward algorithm for coverability

- The coverability problem asks given a net N=(P,T,m_0) and a marking m, if there exists a marking m’≥m such that m’∈$Post^*(m_0)$.

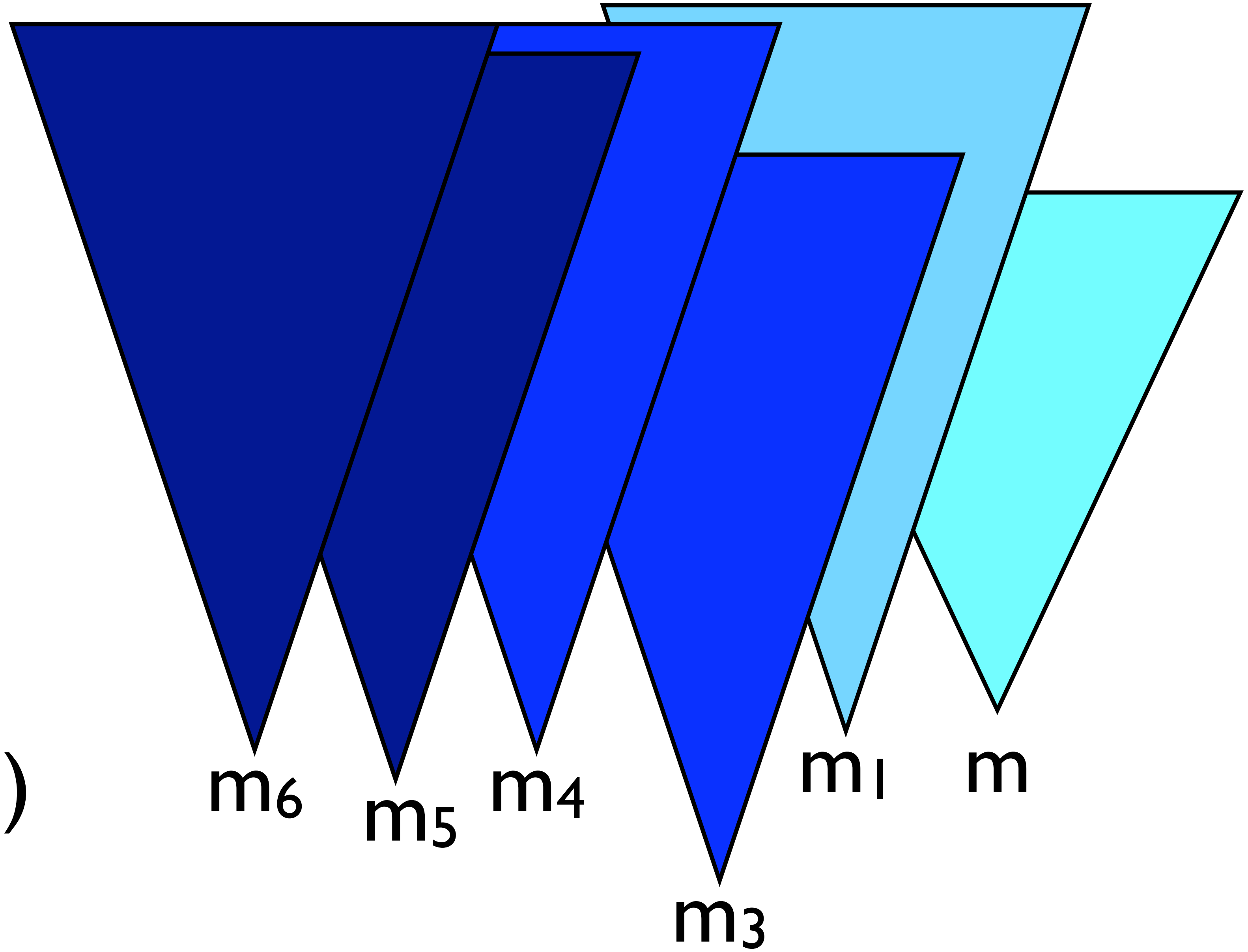
$Pre^2(\uparrow m)$



Backward algorithm for coverability

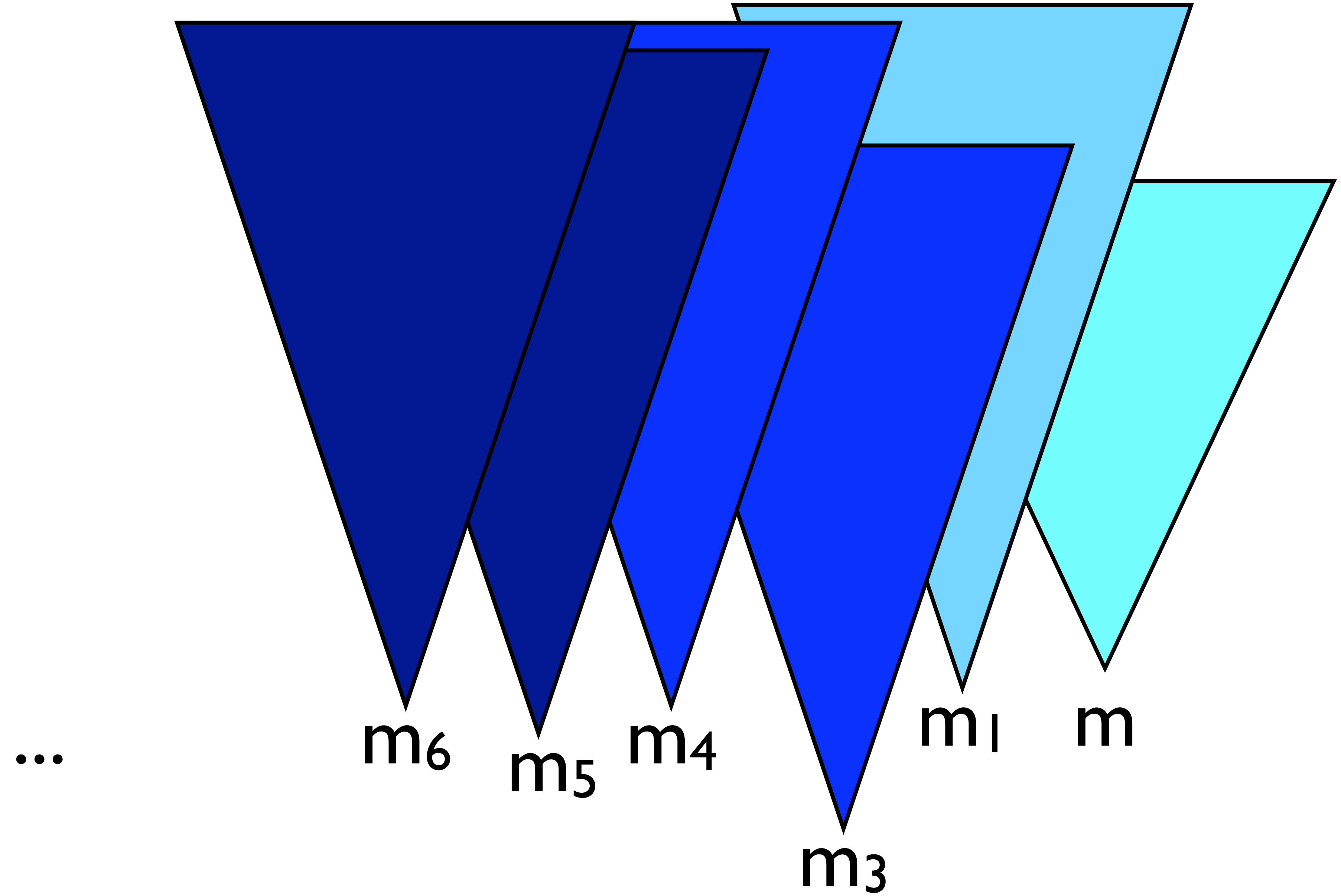
- The coverability problem asks given a net $N=(P,T,m_0)$ and a marking m, if there exists a marking $m' \geqslant m$ such that $m' \in Post^*(m_0)$.

$Pre^3(\uparrow m)$



Backward algorithm for coverability

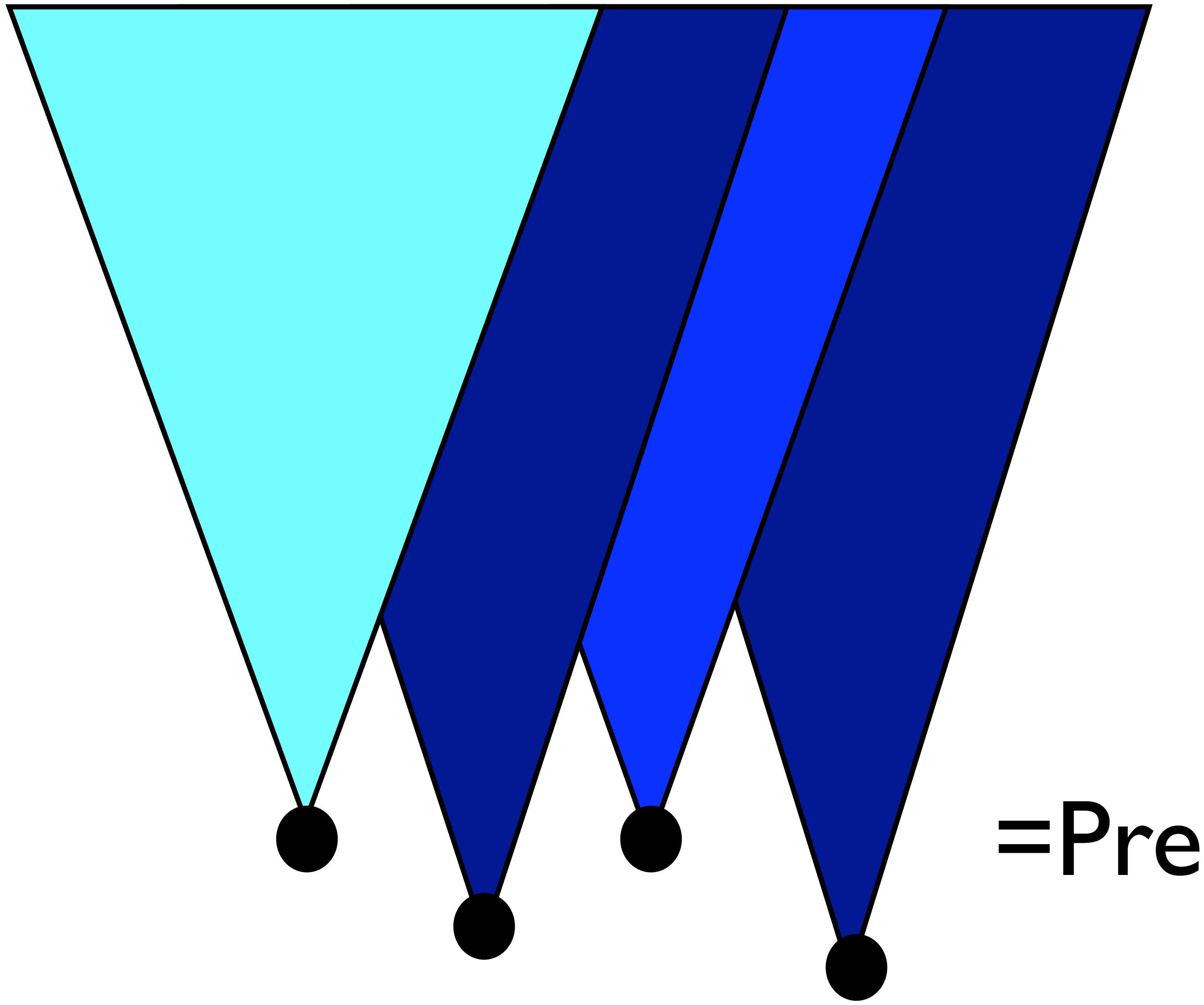
- The coverability problem asks given a net N=(P,T,m_0) and a marking m, if there exists a marking m'⩾m such that m'∈$Post^*(m_0)$.



Backward algorithm for coverability

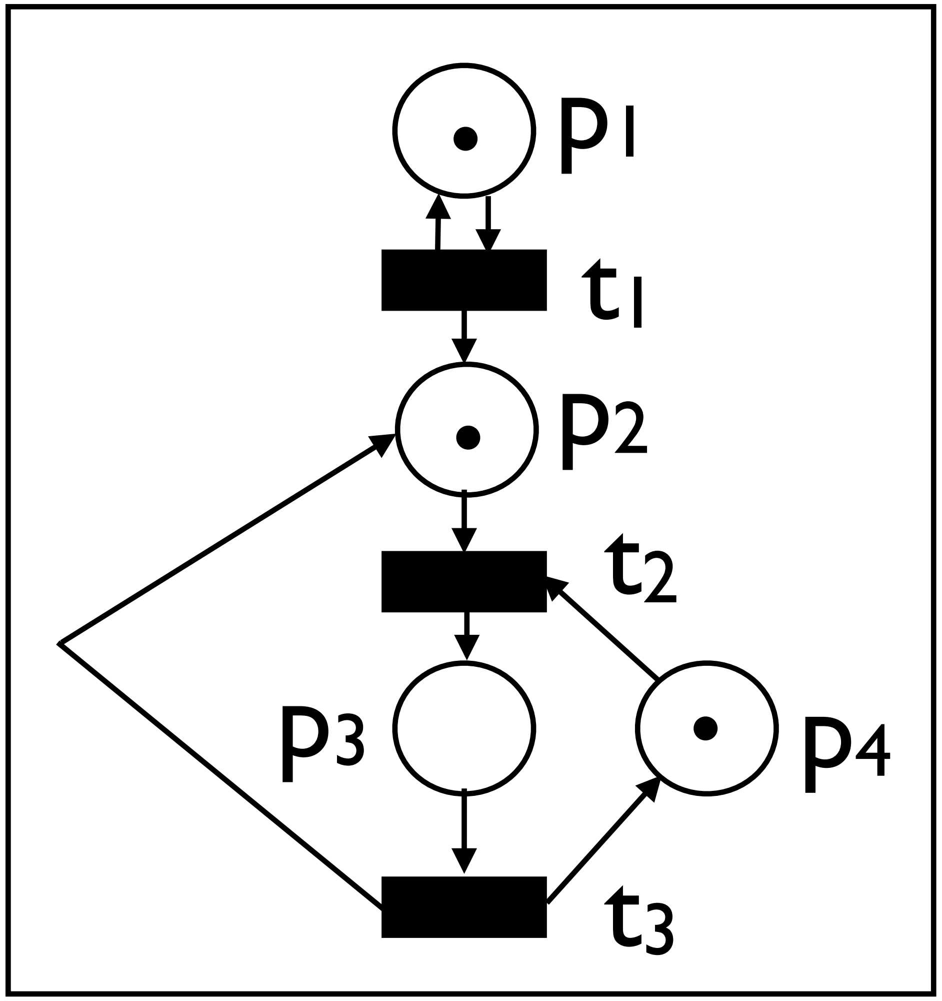
- The coverability problem asks given a net $N=(P,T,m_0)$ and a marking m, if there exists a marking $m' \geqslant m$ such that $m' \in Post^*(m_0)$.

After a finite number of iterations it stabilizes on a set of markings whose upward closure is equal to the set of markings that can reach a marking covering m.

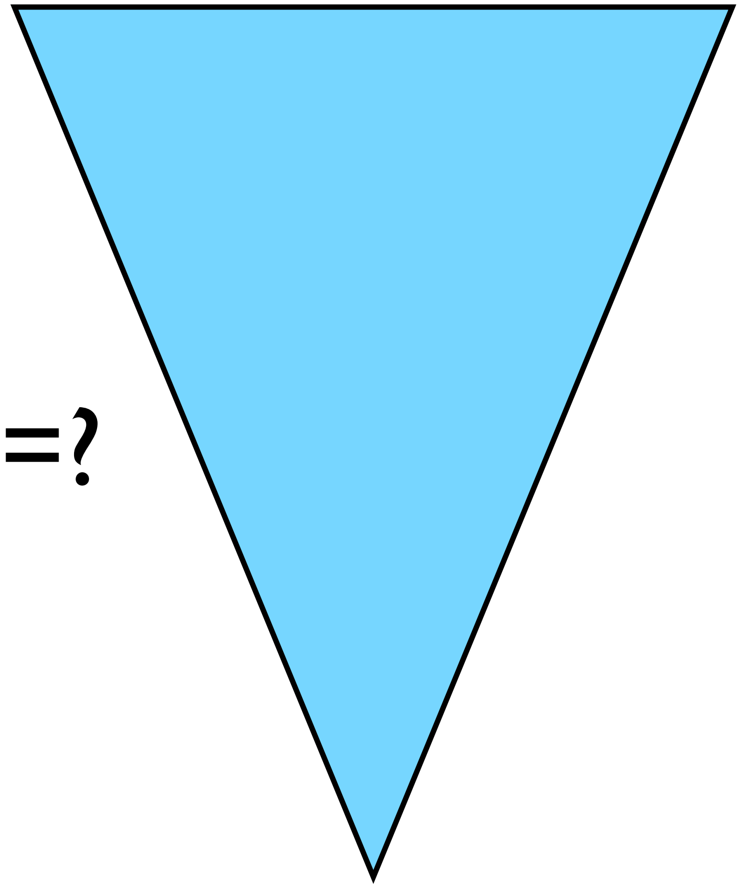


=$Pre^*(m)$

Example

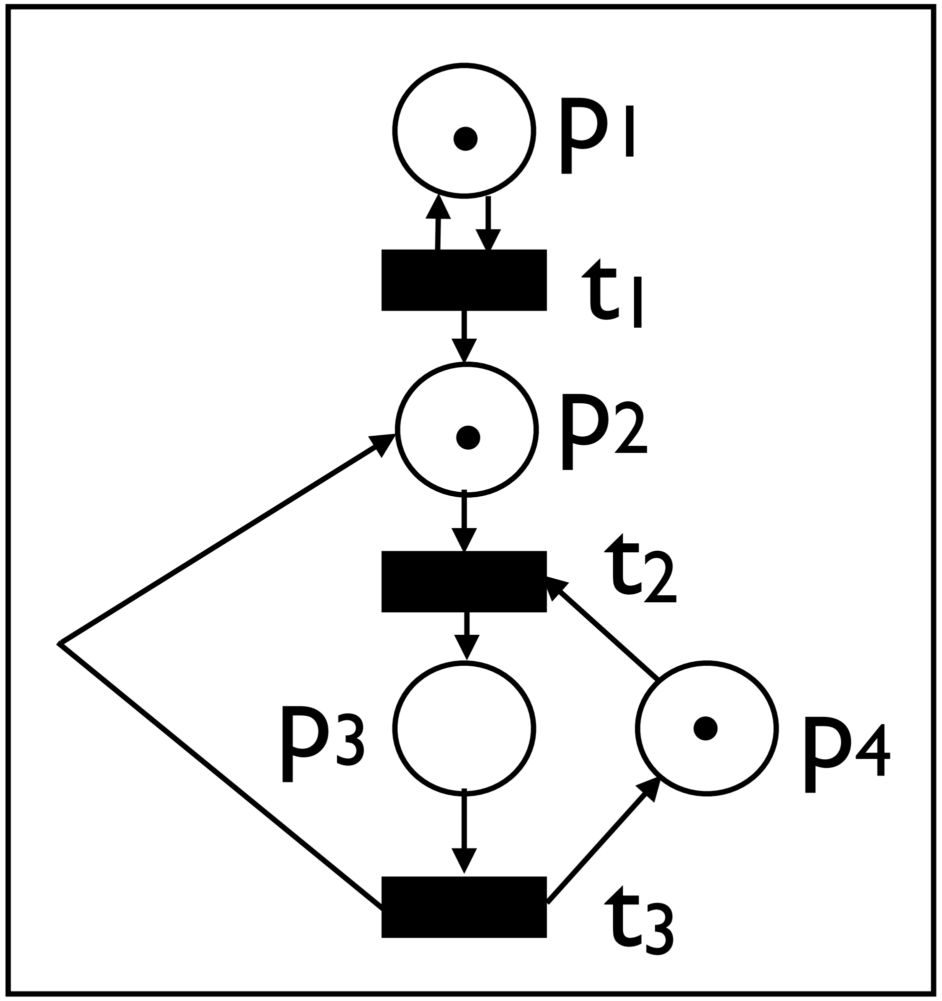


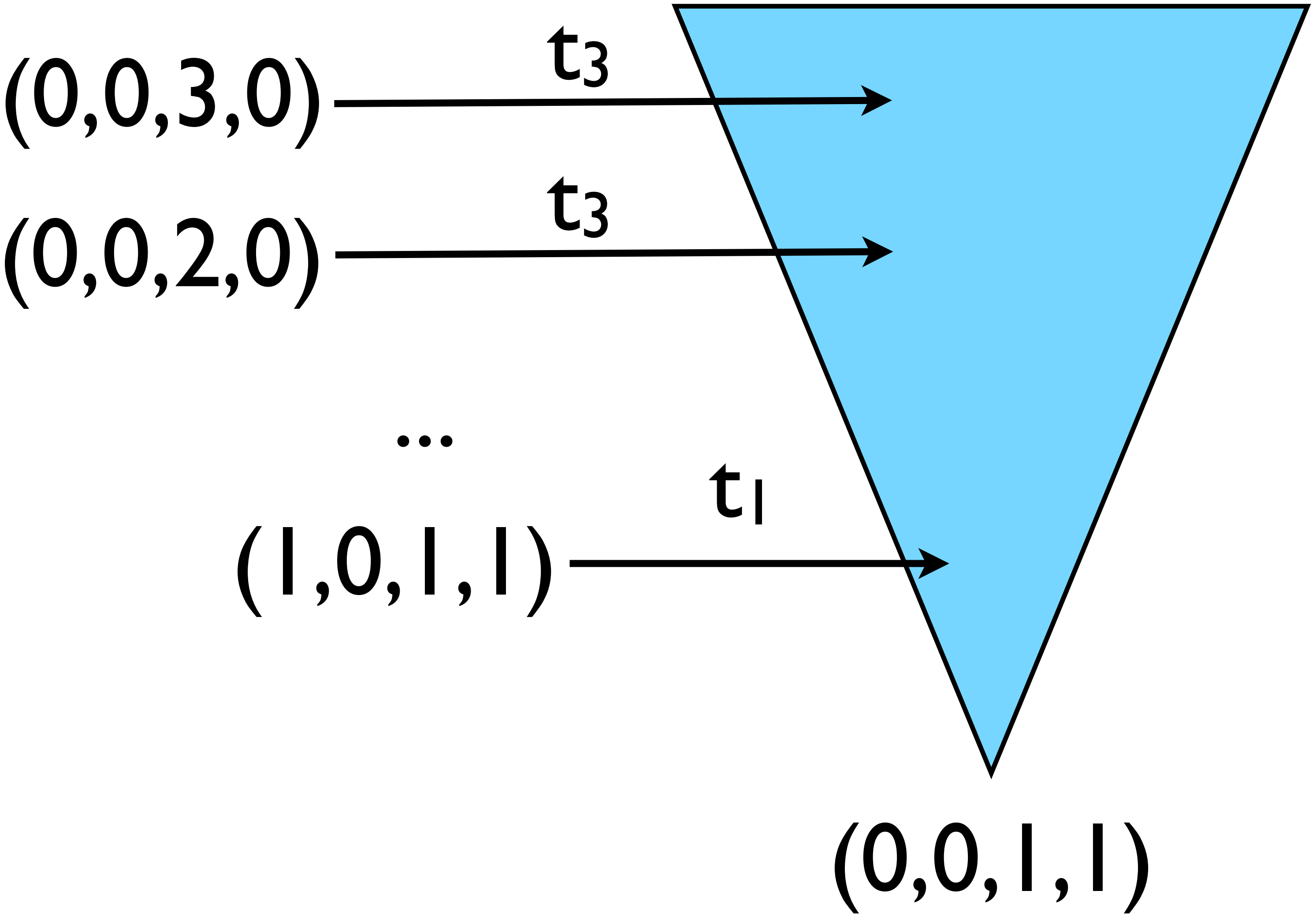
Pre(↑(0,0,1,1))=?



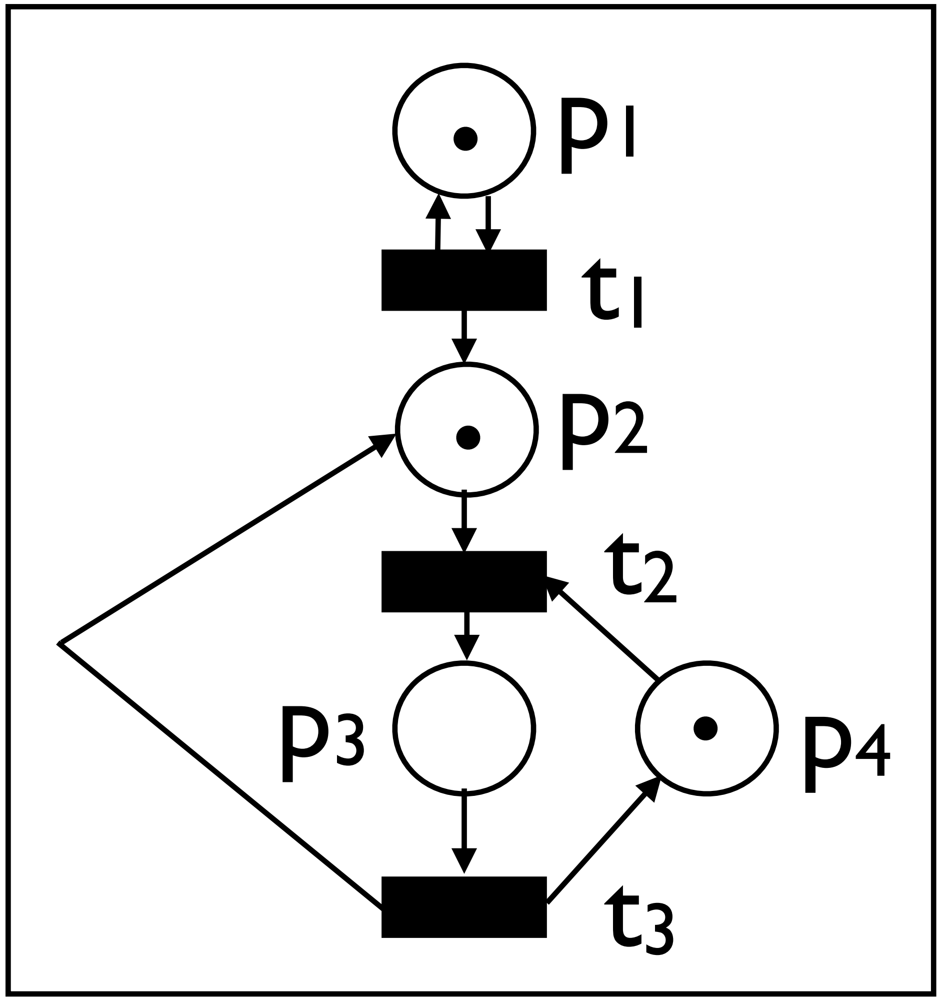
(0,0,1,1)

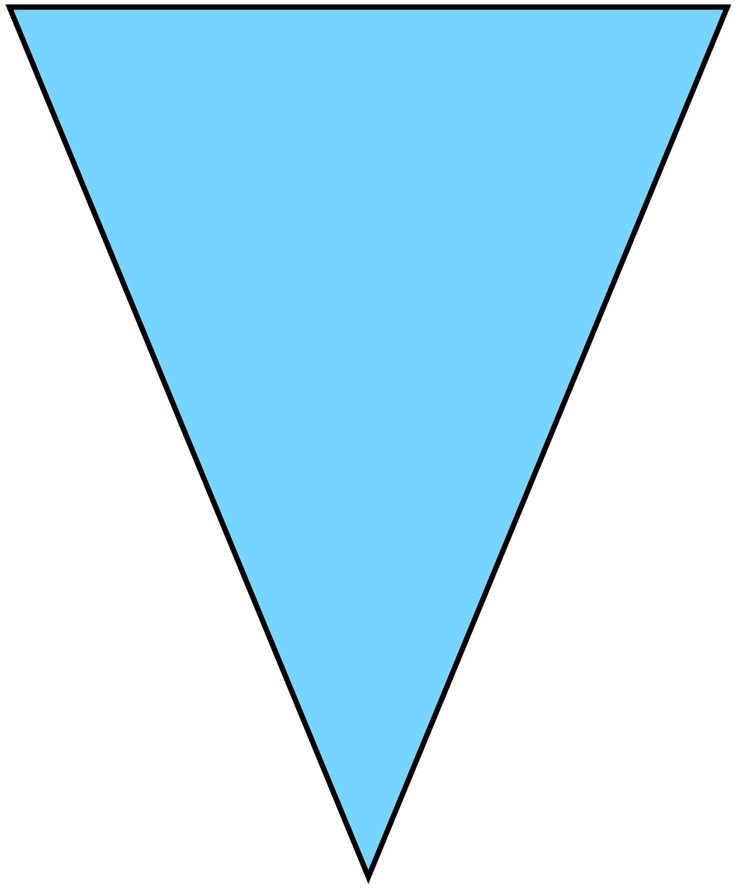
Example





Example

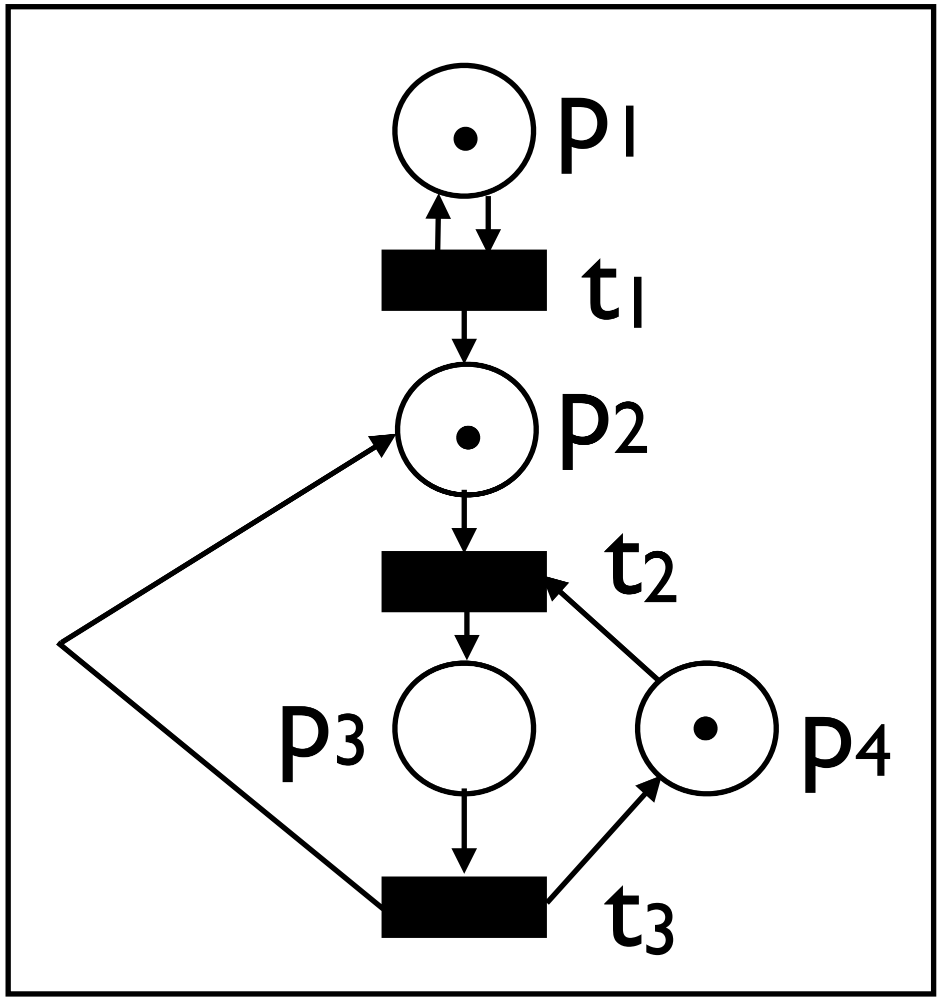


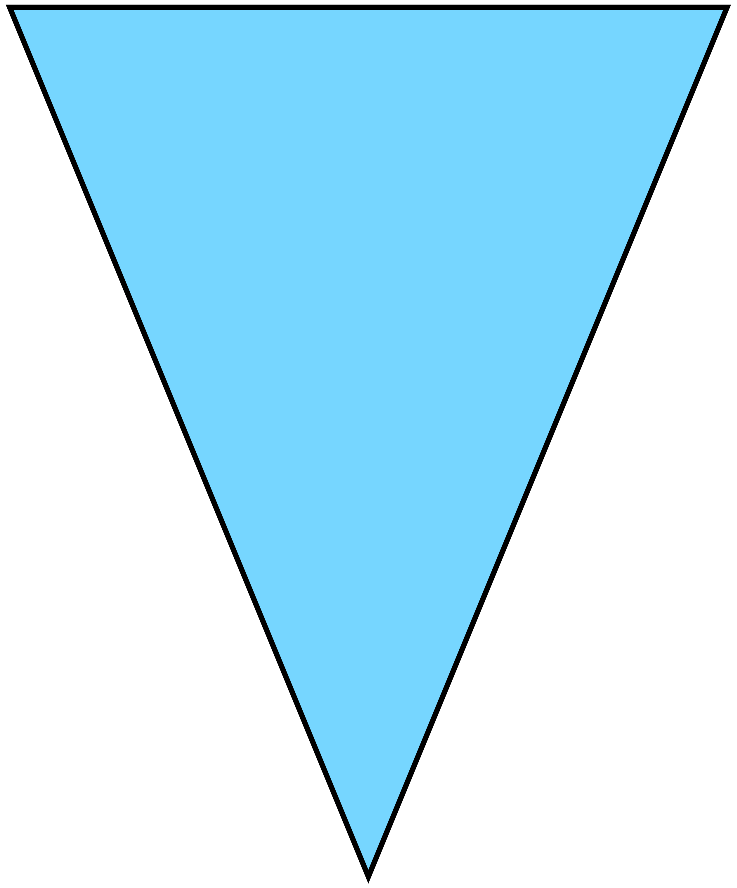


$\mathrm{UGen}(\mathrm{Pre}(\uparrow m))$

$= \mathrm{Min}\{ m' \in \mathbb{N}^{|P|} \mid m' \succcurlyeq I(t) \wedge m'-I(t)+O(t) \succcurlyeq m \}$ (0,0,1,1)

Example





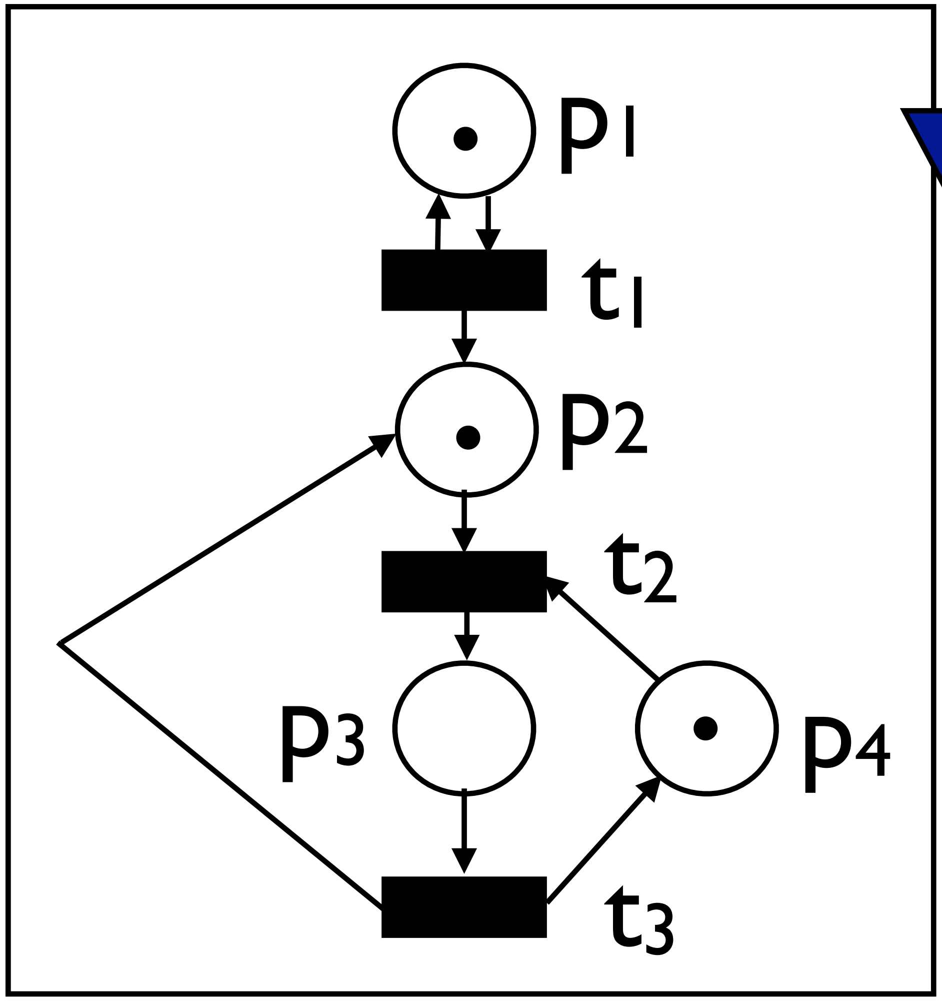
UGen(Pre(↑m))

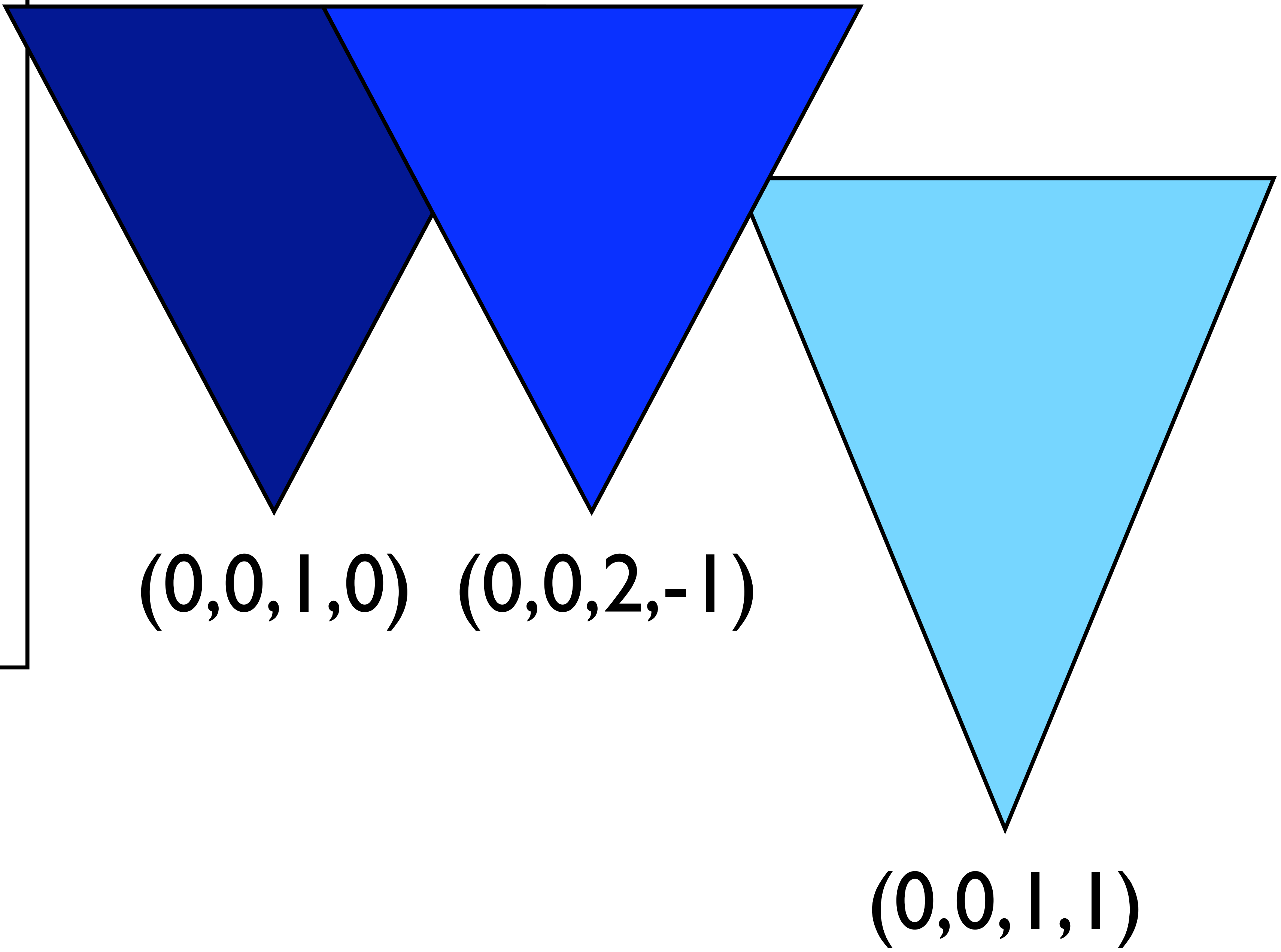
$=\mathrm{Min}\{\ m' \in \mathbb{N}^{|P|} \mid m' \geqslant I(t)\ \wedge m' - I(t) + O(t) \geqslant m\ \}$ $(0,0,1,1)$

=intersection of two upward-closed sets !

Example

For t_3

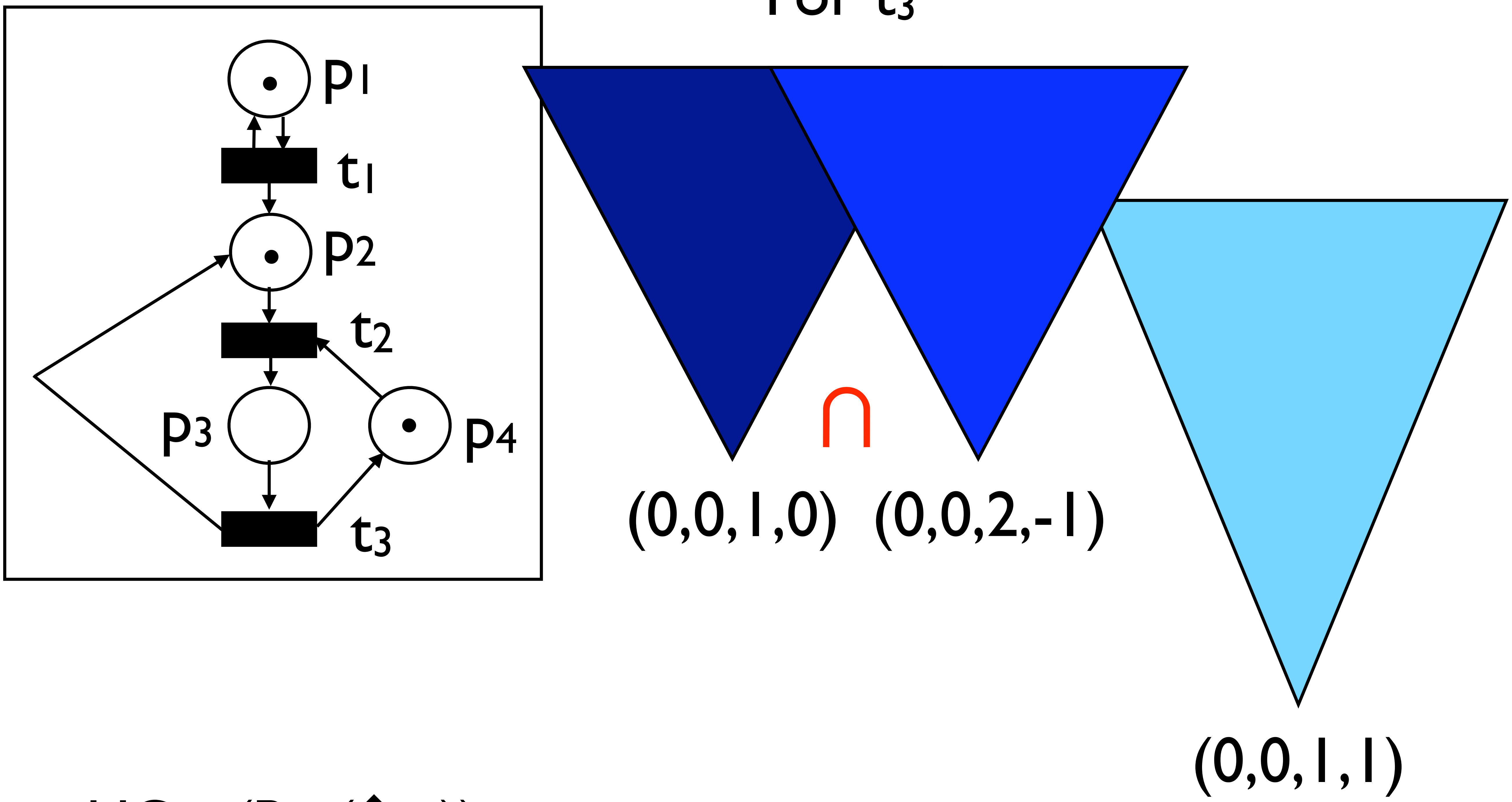




$$\mathrm{UGen}(\mathrm{Pre}(\uparrow m))$$
$$=\mathrm{Min}\{\ m' \in \mathbb{N}^{|P|} \mid m' \geqslant I(t)\ \wedge m'-I(t)+O(t) \geqslant m\ \}$$

Example

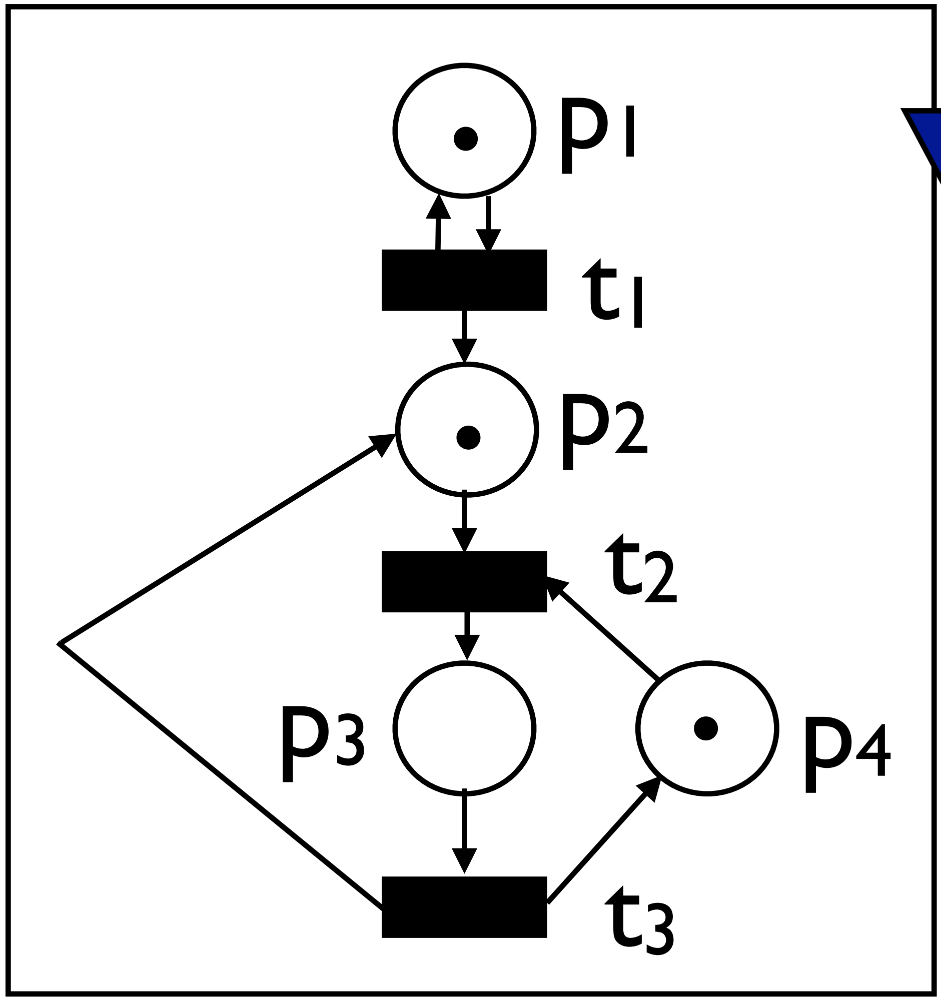
For t_3

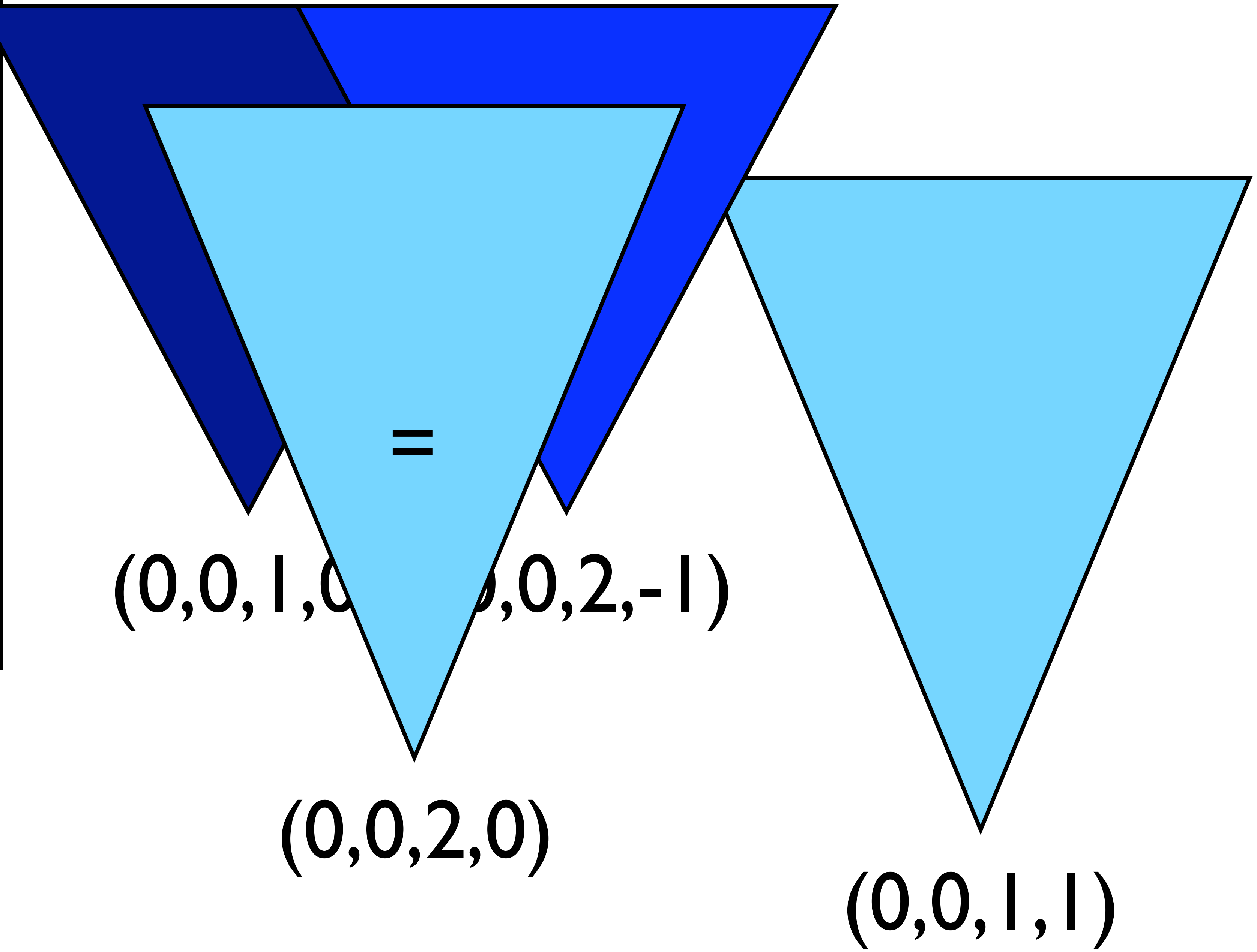


$$\mathrm{UGen}(\mathrm{Pre}(\uparrow m))$$
$$= \mathrm{Min}\{\, m' \in \mathbb{N}^{|P|} \mid m' \geqslant I(t) \wedge m' - I(t) + O(t) \geqslant m \,\}$$

Example

For t_3

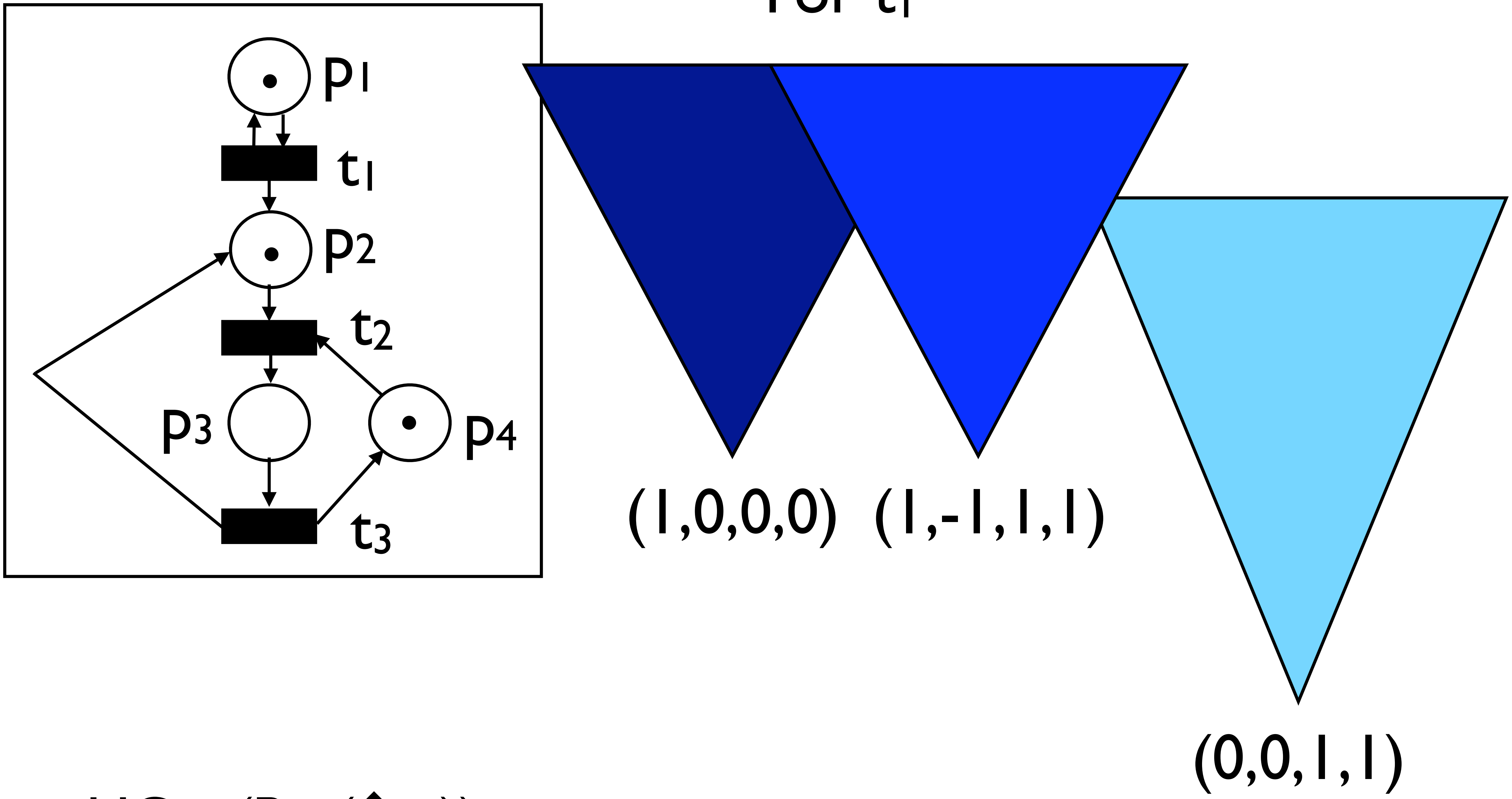




$$UGen(Pre(\uparrow m))$$
$$=Min\{\ m' \in \mathbb{N}^{|P|}\ |\ m' \geqslant I(t)\ \wedge m'-I(t)+O(t) \geqslant m\ \}$$

Example

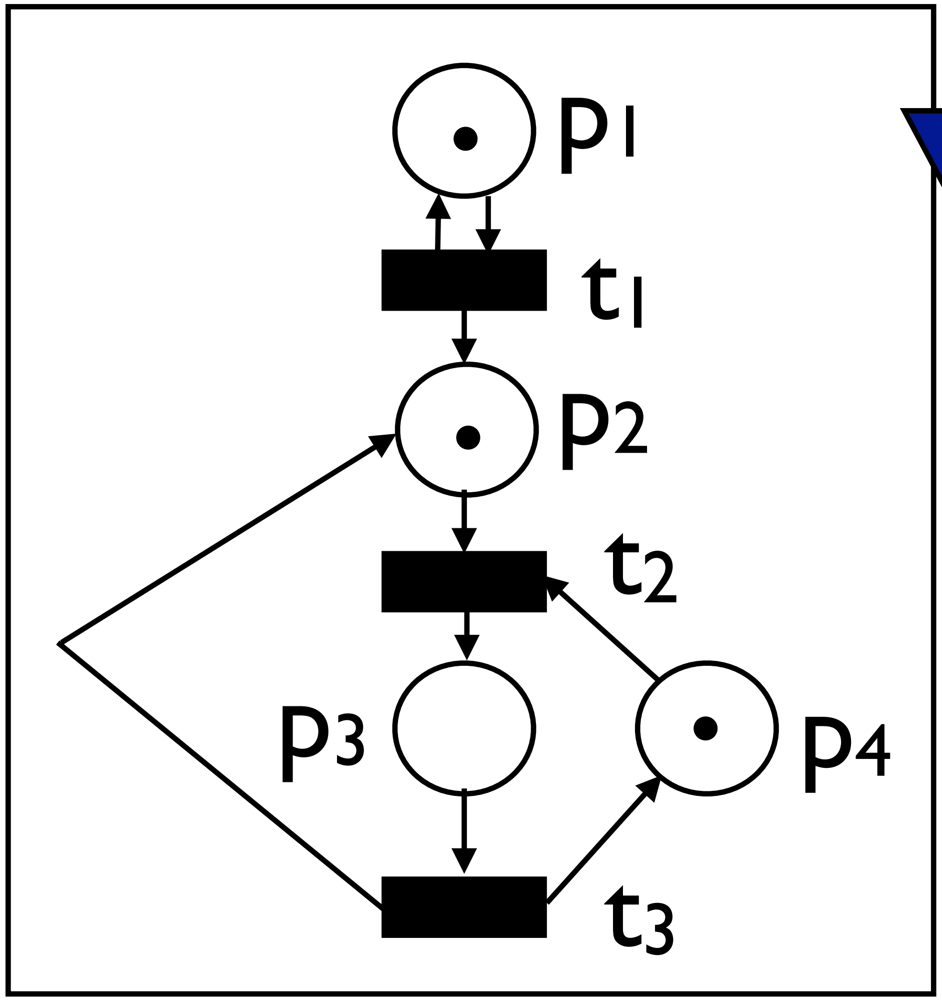
For t_1

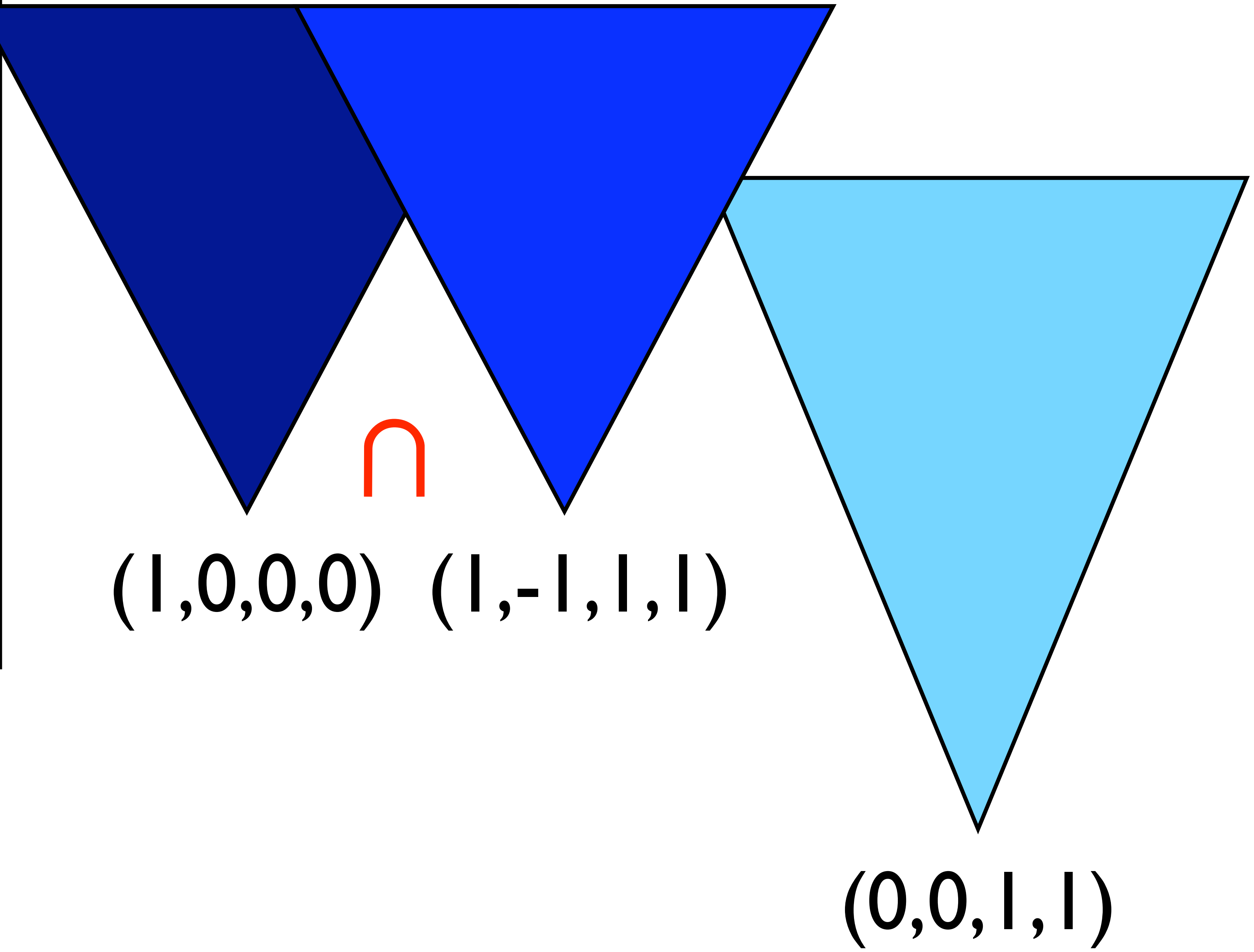


$$\mathrm{UGen}(\mathrm{Pre}(\uparrow m))$$
$$=\mathrm{Min}\{\ m' \in \mathbb{N}^{|P|}\ |\ m' \geqslant I(t)\ \wedge m' - I(t) + O(t) \geqslant m\ \}$$

Example

For t_1

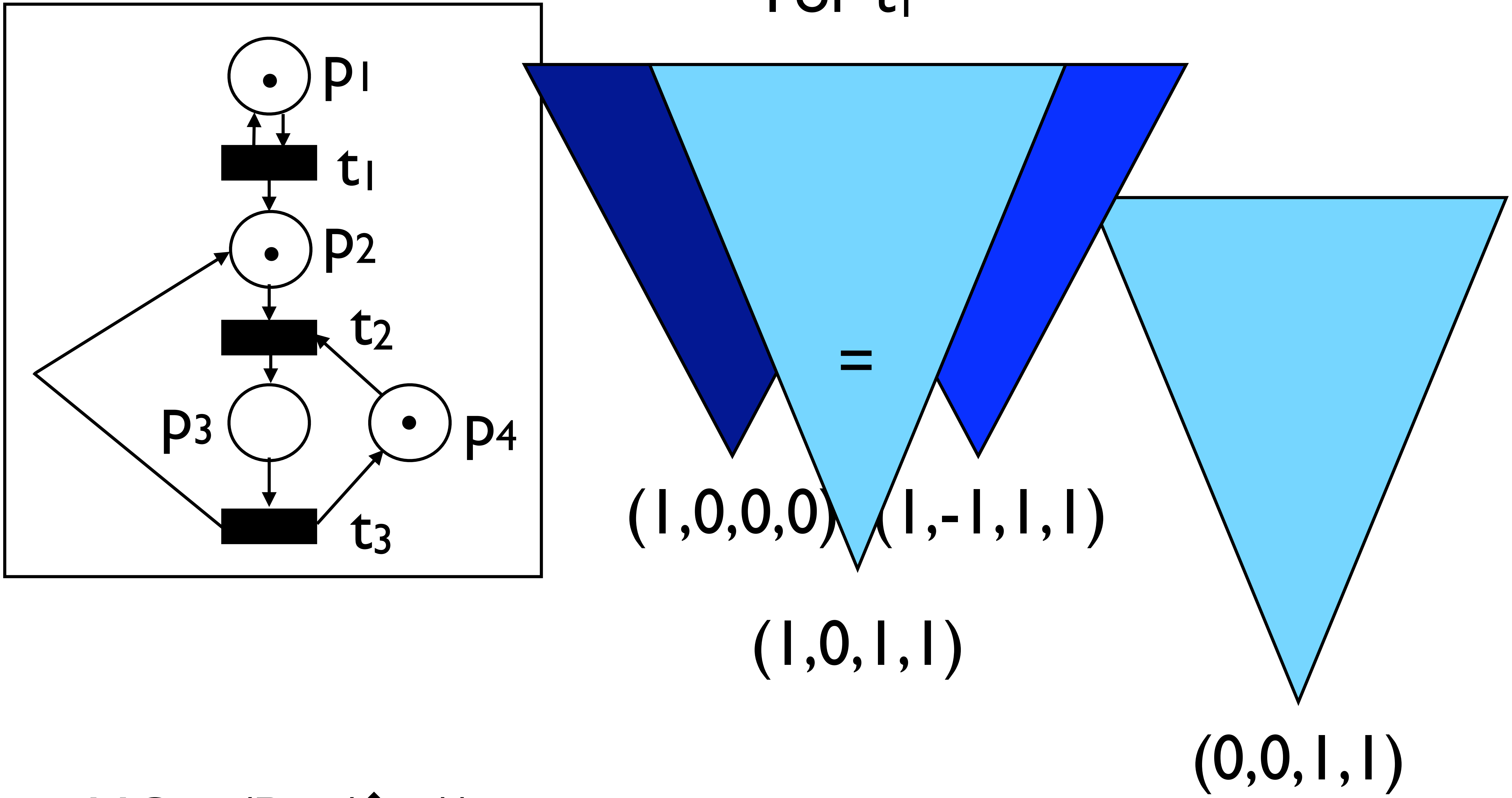




$$\mathrm{UGen}(\mathrm{Pre}(\uparrow m))$$
$$= \mathrm{Min}\{\, m' \in \mathbb{N}^{|P|} \mid m' \geqslant I(t) \wedge m' - I(t) + O(t) \geqslant m \,\}$$

Example

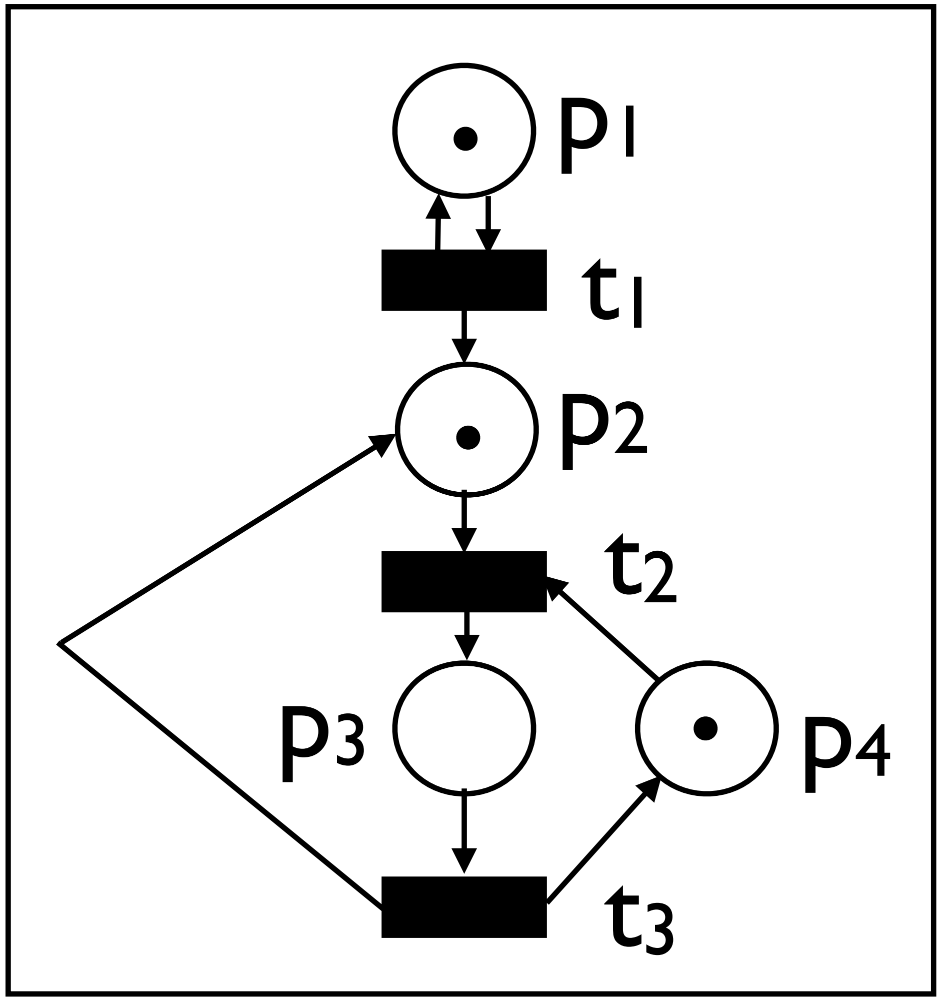
For t_1

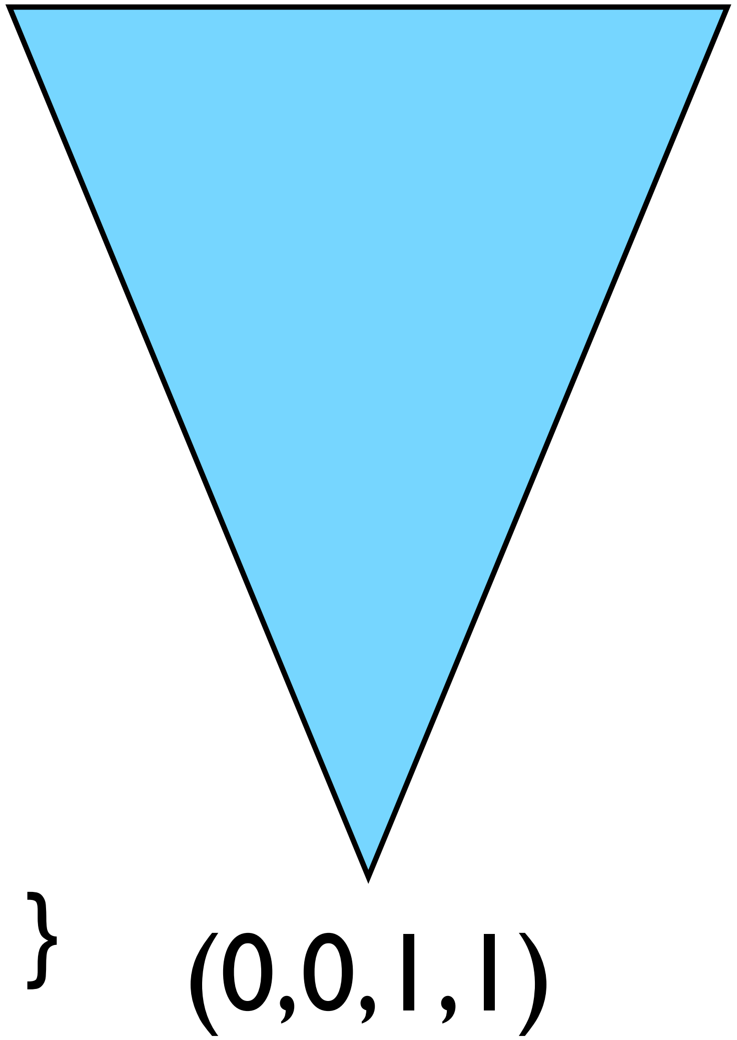


$UGen(Pre(\uparrow m))$

$= Min\{\ m' \in \mathbb{N}^{|P|} \mid m' \geqslant I(t)\ \wedge m' - I(t) + O(t) \geqslant m\ \}$

Example





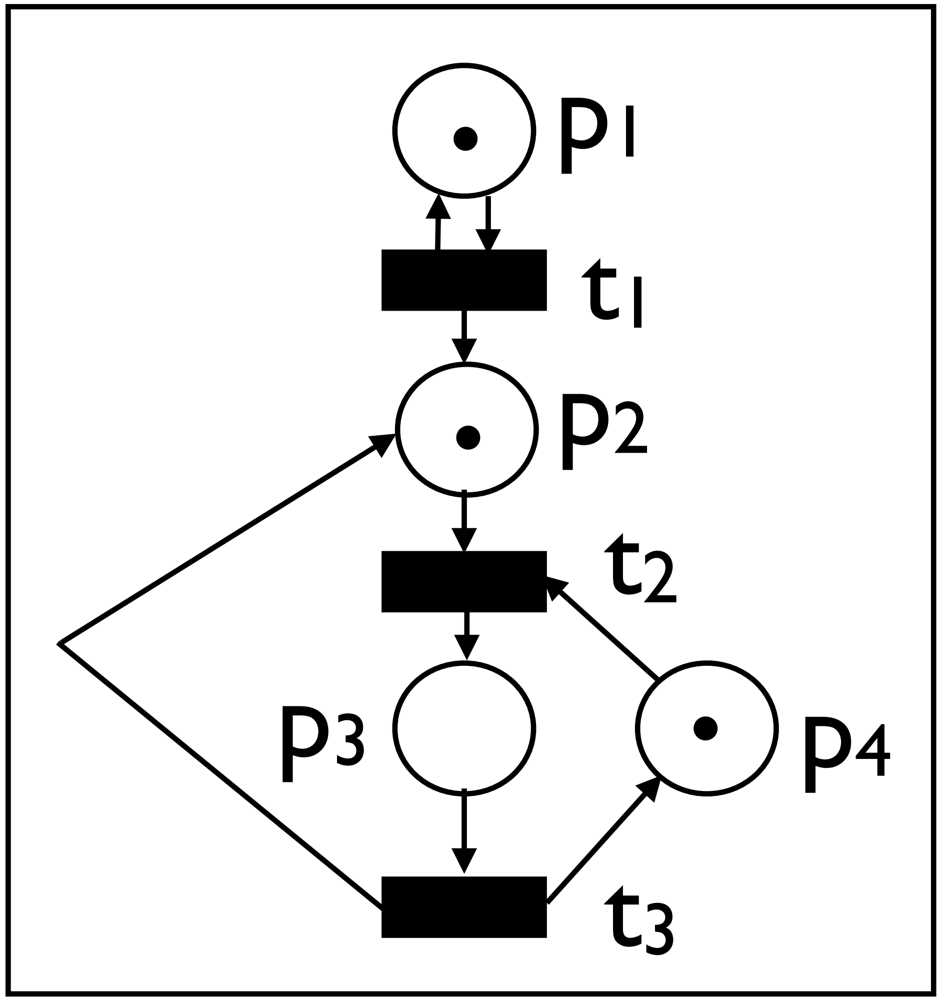
$\mathrm{UGen}(\mathrm{Pre}(\uparrow m))$

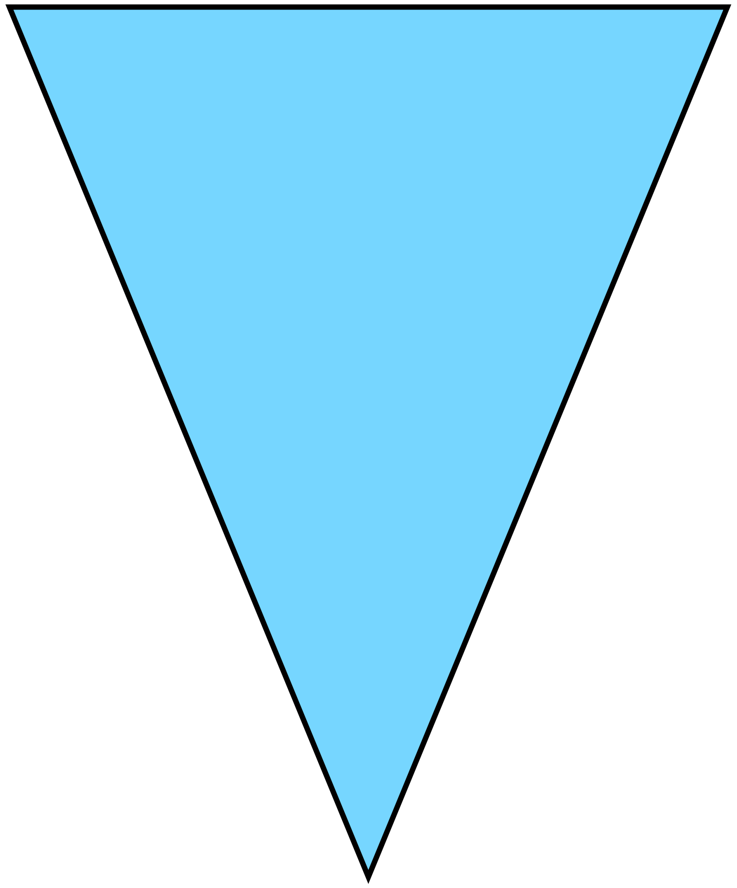
$= \mathrm{Min}\{\, m' \in \mathbb{N}^{|P|} \mid m' \succcurlyeq I(t) \wedge m' - I(t) + O(t) \succcurlyeq m \,\}$

$= \mathrm{Min}\{(1,0,1,1),(0,0,2,0),(0,1,0,1)\}$

$= \{(1,0,1,1),(0,0,2,0),(0,1,0,1)\}$

Example



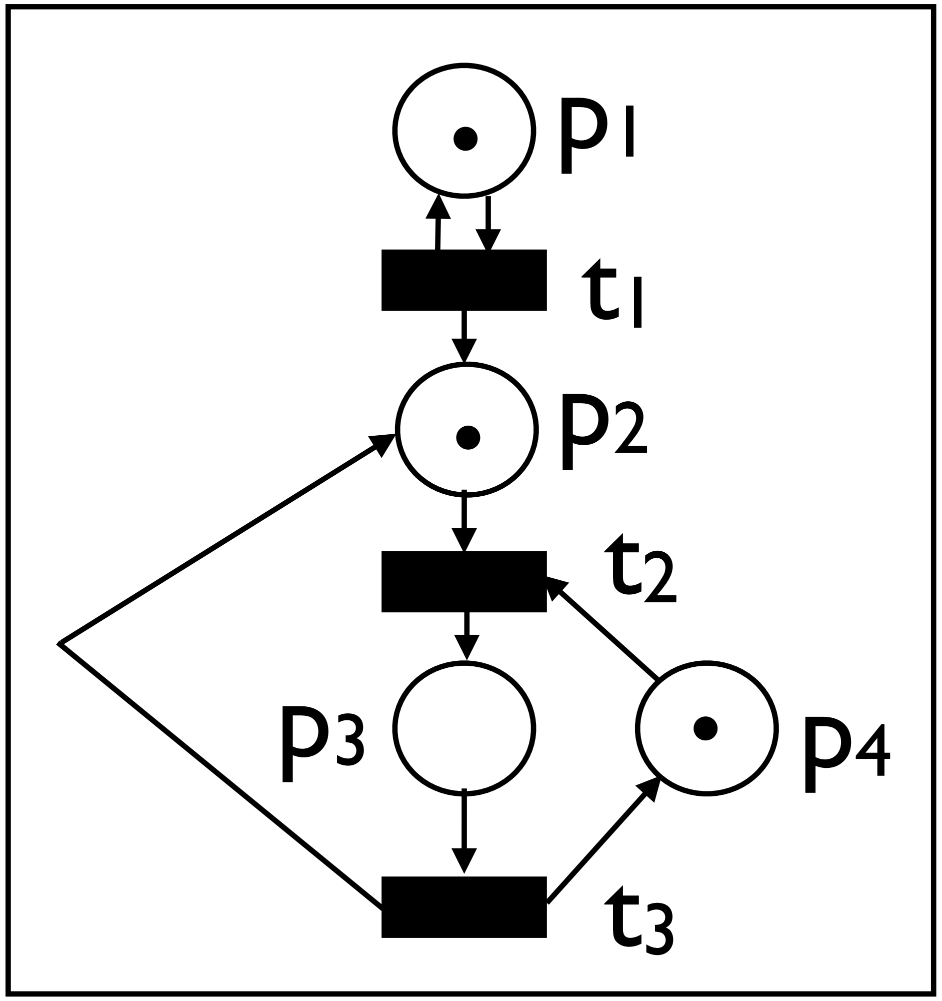


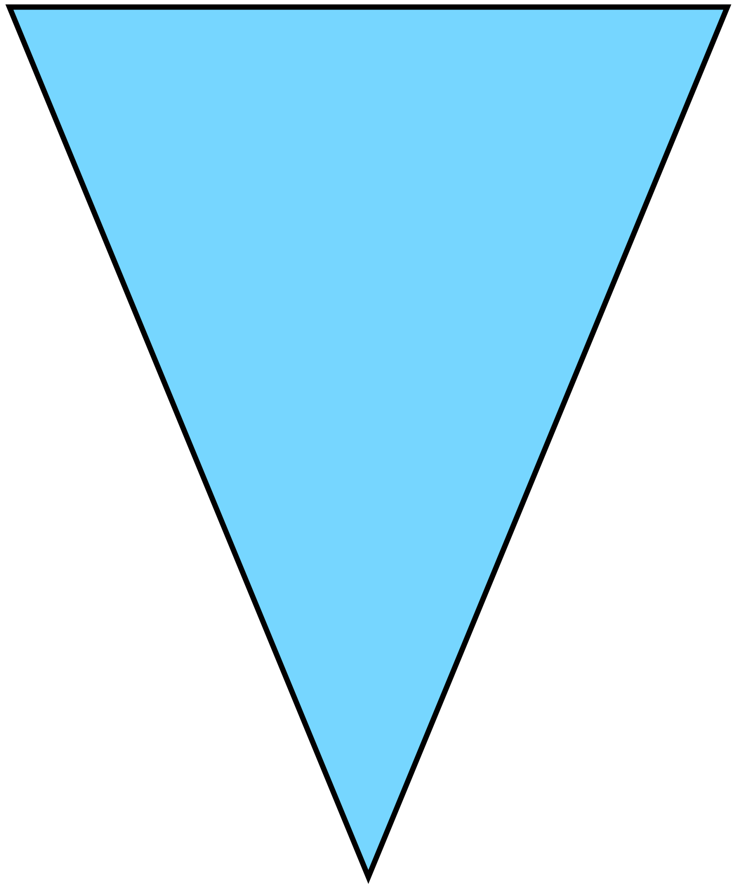
$\mathrm{UGen}(\mathrm{Pre}(\uparrow m) \cup \uparrow m)$

$= \mathrm{Min}(\{(1,0,1,1),(0,0,2,0),(0,1,0,1)\} \cup \uparrow\{(0,0,1,1)\}$

$= \{(0,0,2,0),(0,1,0,1),(0,0,1,1)\}$

Example





$\mathrm{UGen}(\mathrm{Pre}(\uparrow m) \cup \uparrow m)$

$= \mathrm{Min}(\{(1,0,1,1),(0,0,2,0),(0,1,0,1)\} \cup \uparrow \{(0,0,1,1)\}$

$= \{(0,0,2,0),(0,1,0,1),(0,0,1,1)\}$

• • •

Set saturation methods for EPN

- **Theorem.** The coverability problem for extended Petri net is decidable.

Set saturation methods for EPN

- **Theorem.** The coverability problem for extended Petri net is decidable.

Nevertheless, the worst case complexity is high:

- **Theorem.** The coverability problem is ExpSpace-C for Petri nets.
- **Theorem.** The coverability problem is non-primitive recursive for transfer/reset/NBA PN.

Technique 2:
Tree saturation

Tree saturation

Tree saturation

=

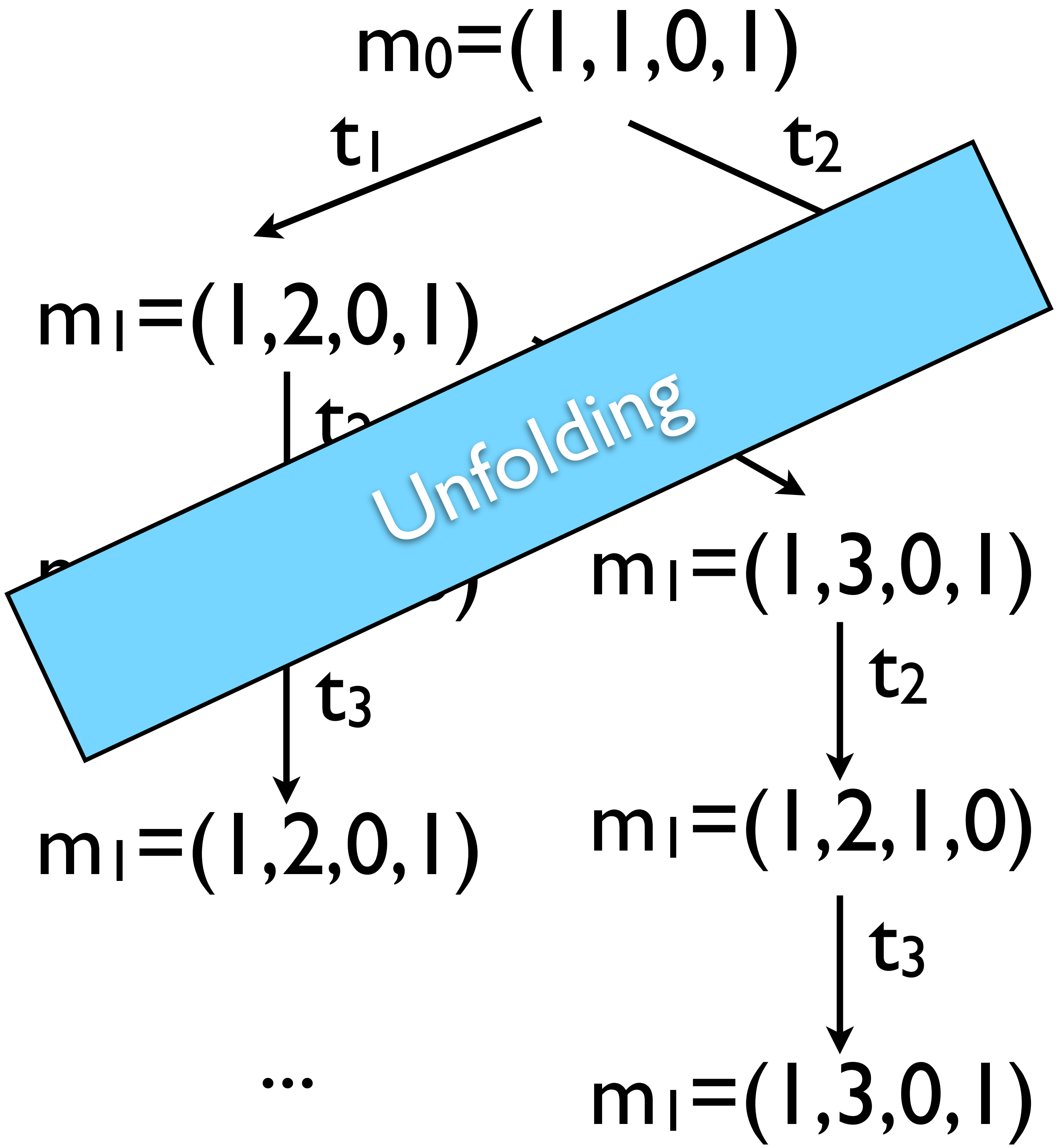
Unfolding

+

Rule to stop

Objective: construct a finite tree that represents (in some way) all the computations of the transition system.

Tree saturation for PN



$m_0=(1,1,0,1)$

t_1

t_2

$m_1=(1,2,0,1)$

t_3

$m_1=(1,3,0,1)$

t_2

$m_1=(1,2,0,1)$

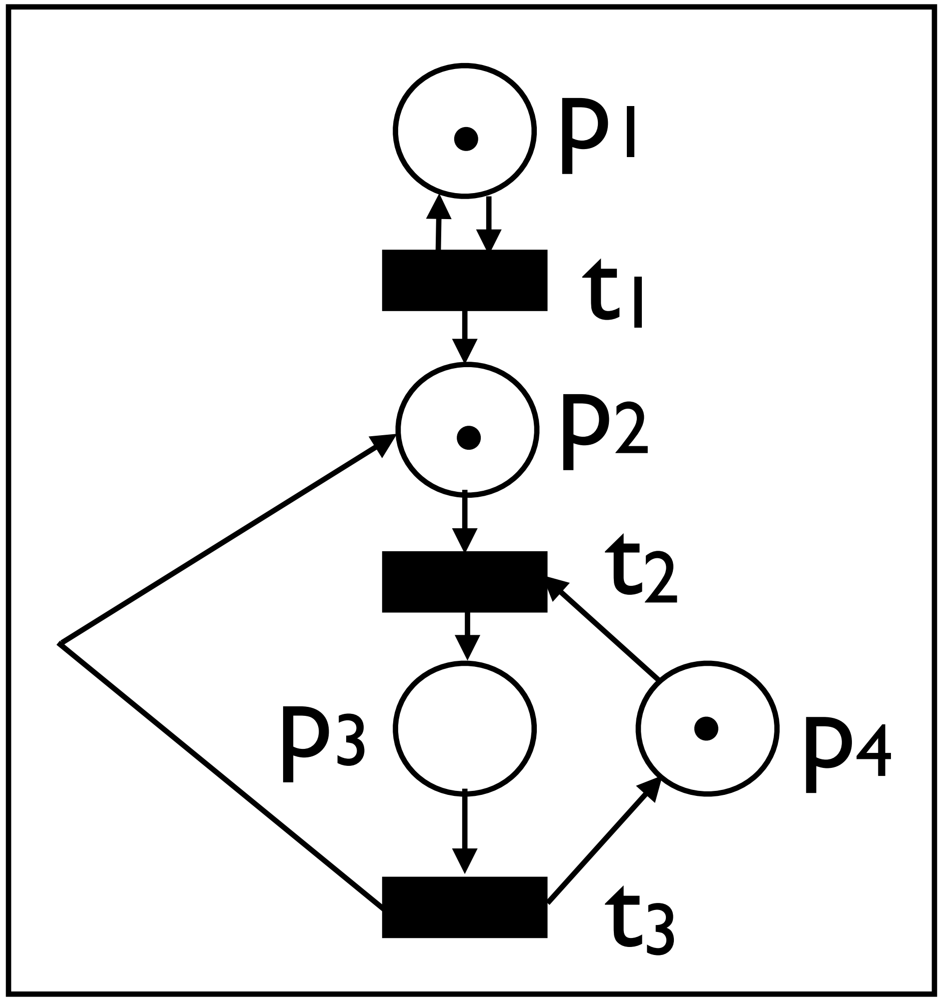
$m_1=(1,2,1,0)$

t_3

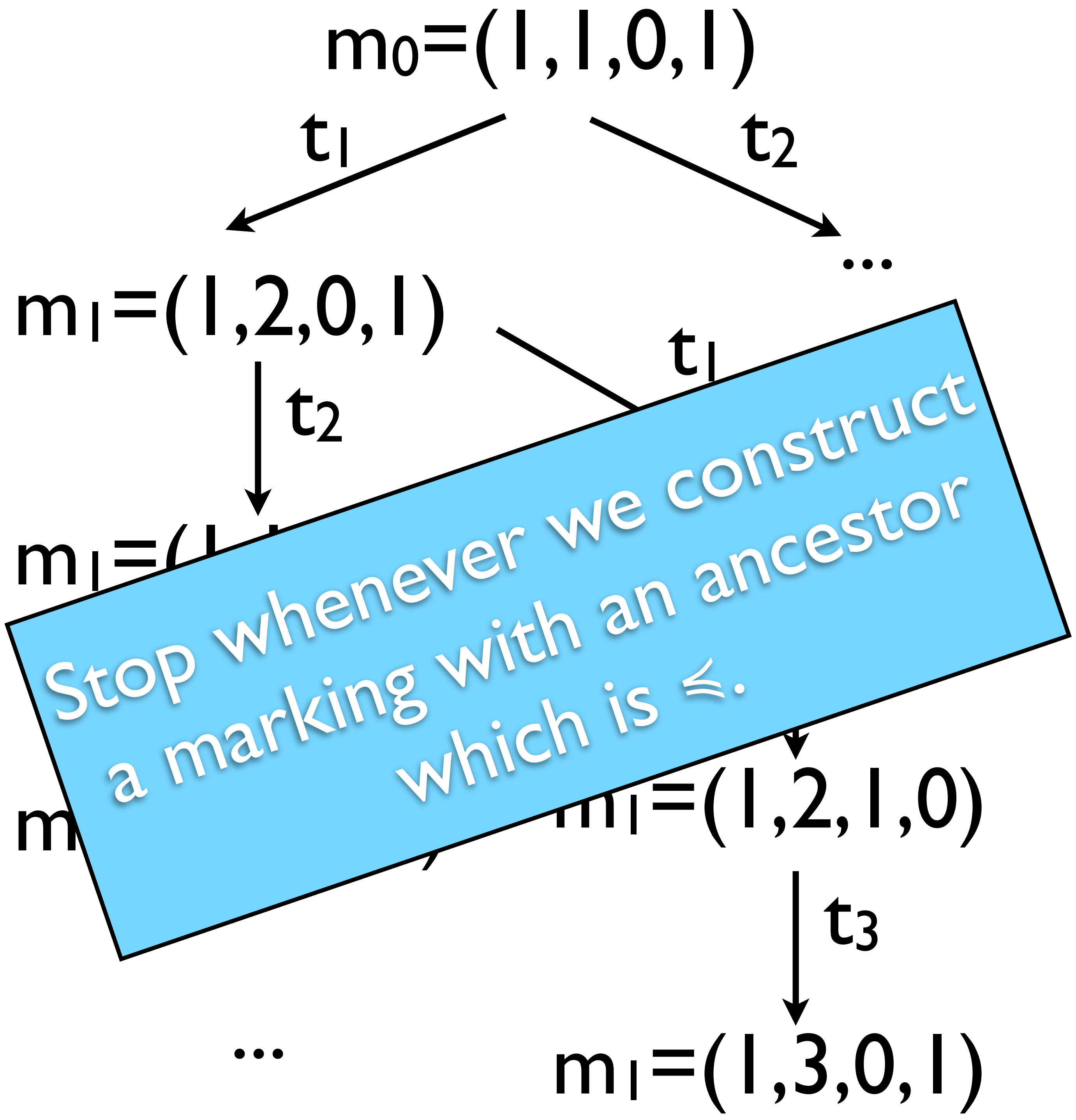
...

$m_1=(1,3,0,1)$

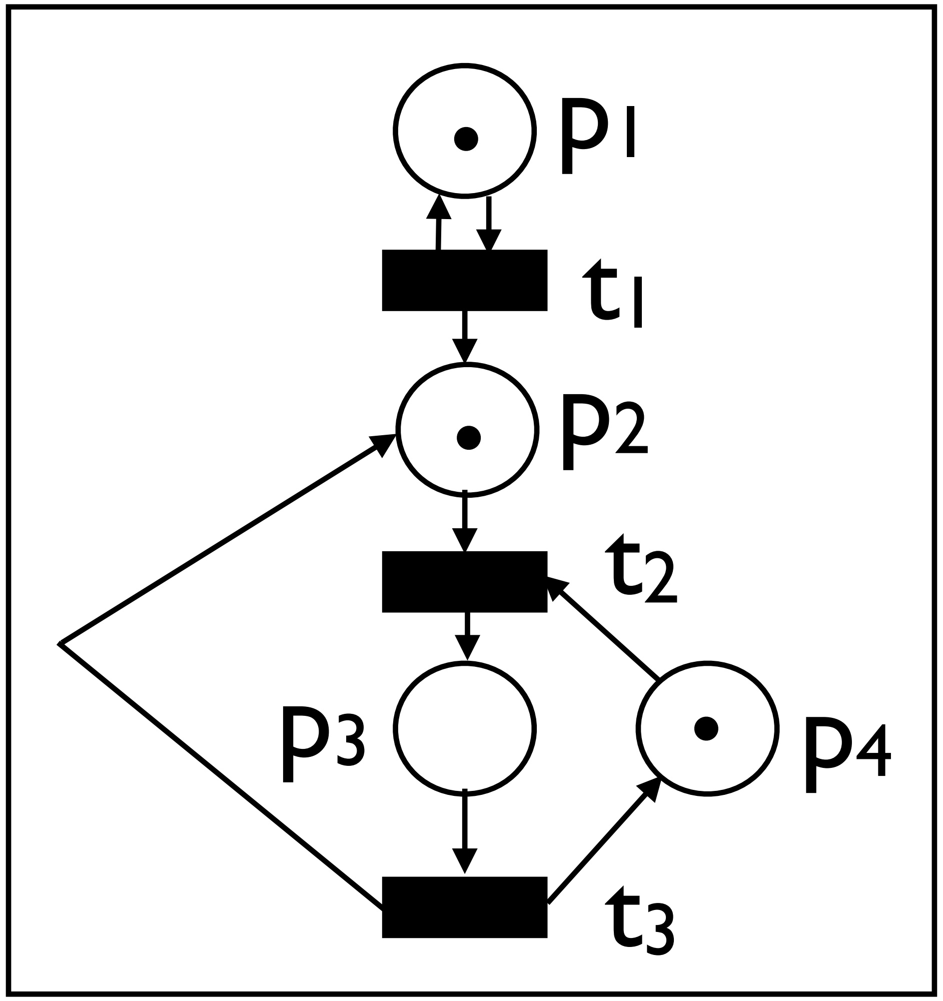
Unfolding



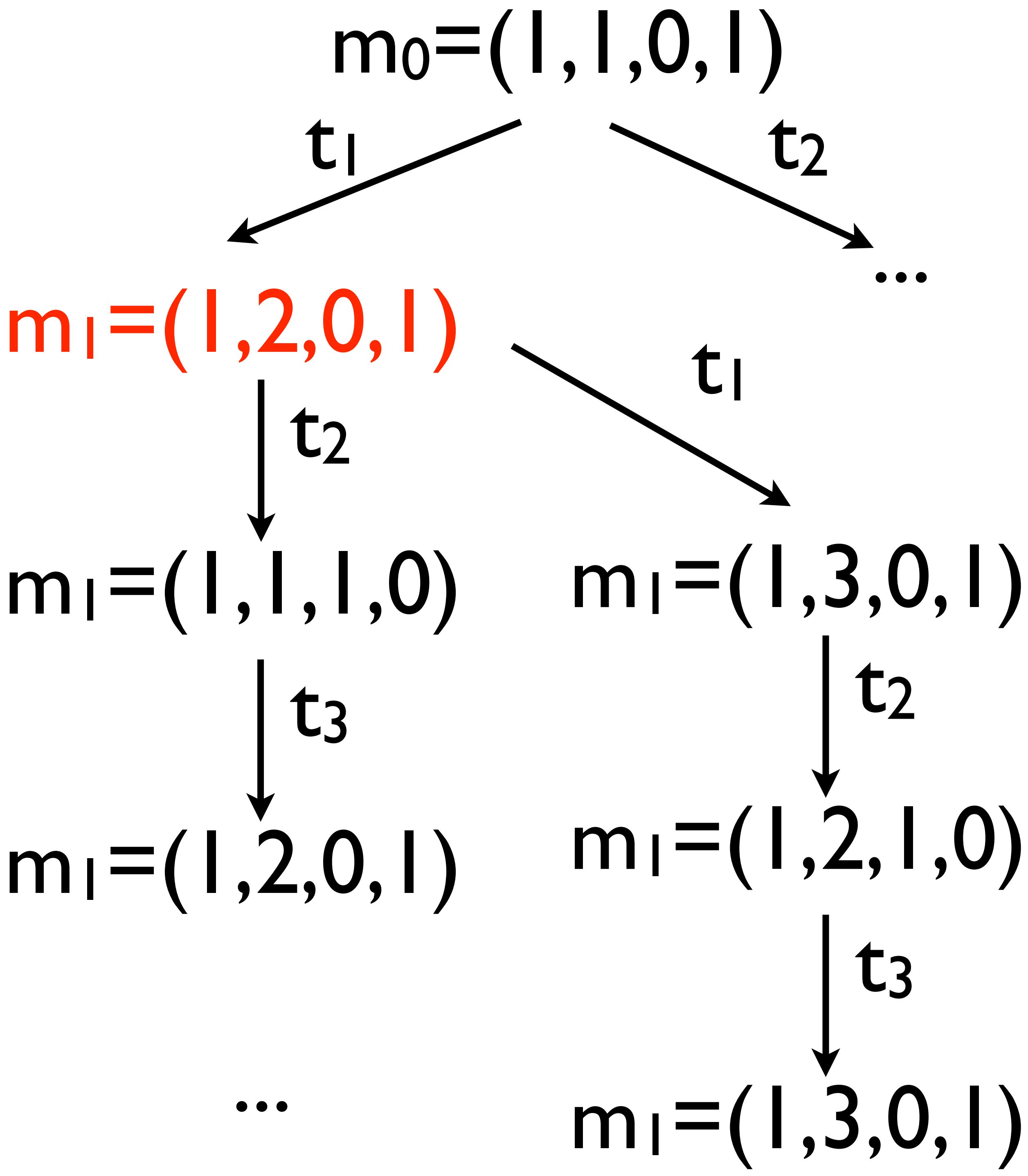
Tree saturation for PN



Stop whenever we construct a marking with an ancestor which is ≤.



Tree saturation for PN



$m_0=(1,1,0,1)$

t_1 t_2

$m_1=(1,2,0,1)$ ...

t_2 t_1

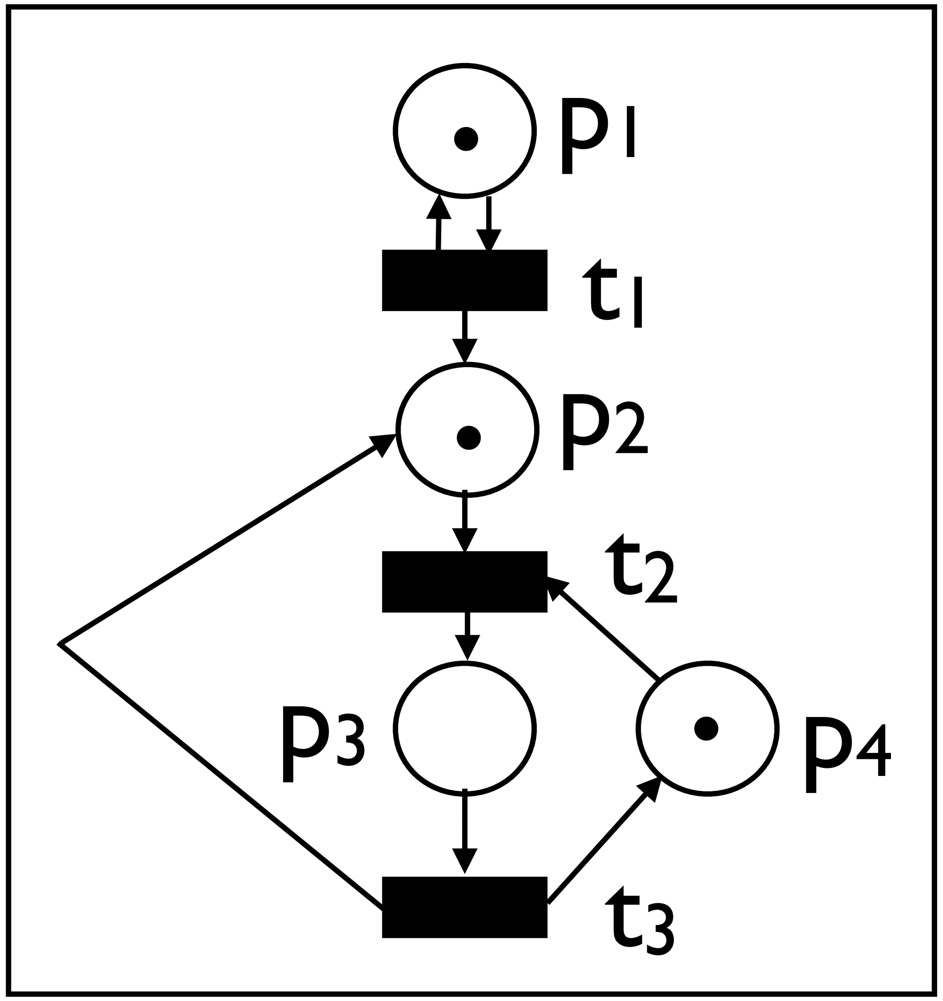
$m_1=(1,1,1,0)$ $m_1=(1,3,0,1)$

t_3 t_2

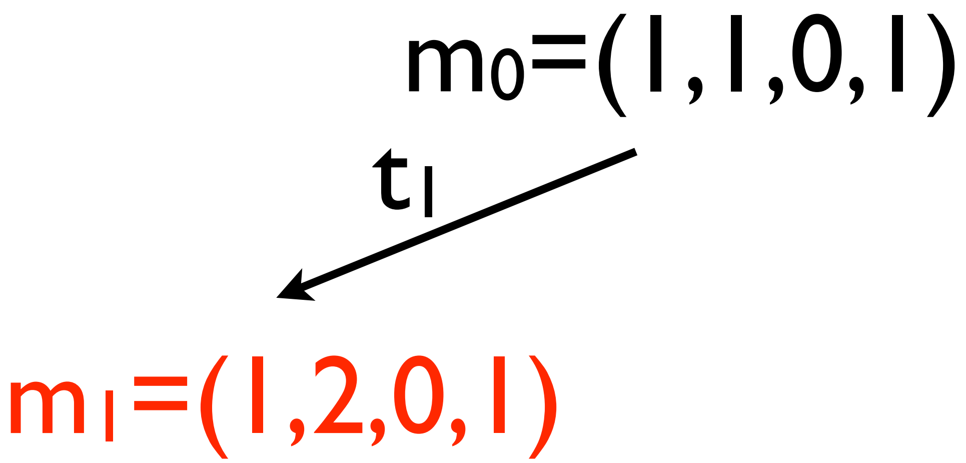
$m_1=(1,2,0,1)$ $m_1=(1,2,1,0)$

t_3

... $m_1=(1,3,0,1)$



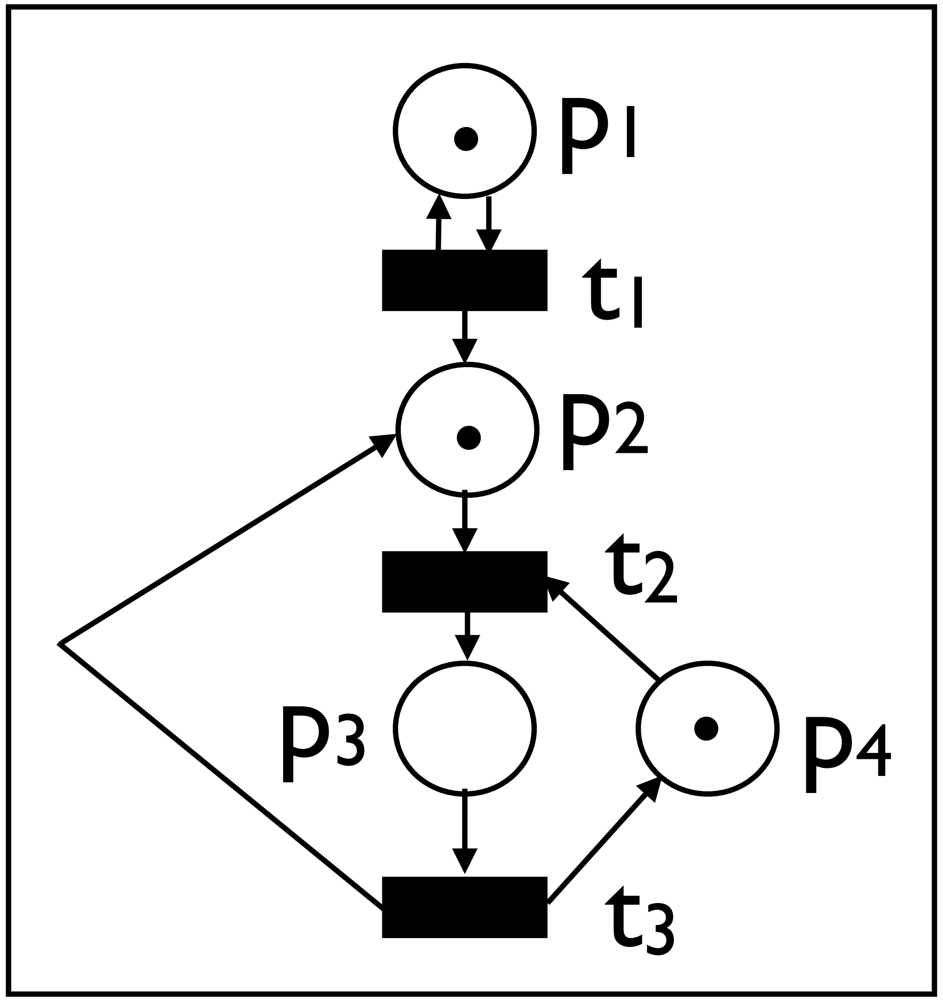
Tree saturation for PN



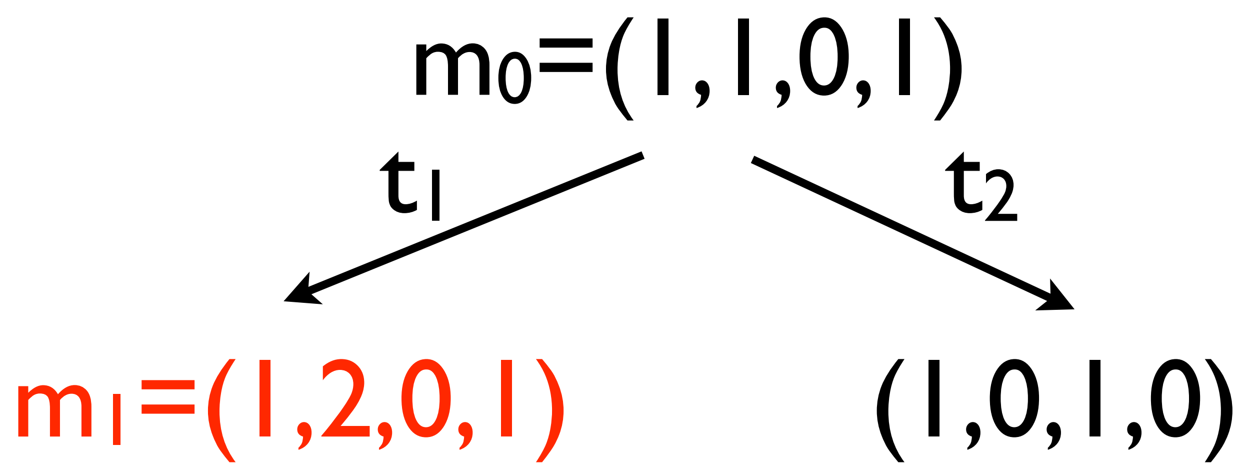
$m_0=(1,1,0,1)$

t_1

$m_1=(1,2,0,1)$



Tree saturation for PN



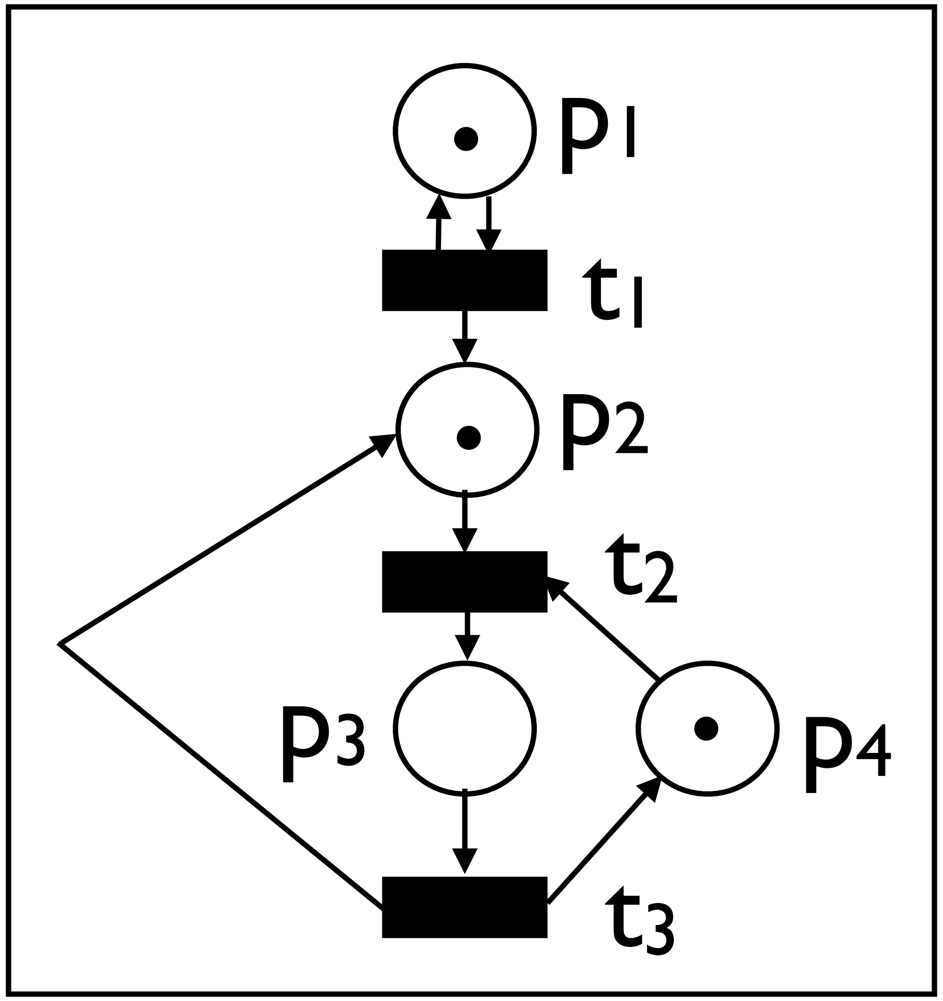
$m_0=(1,1,0,1)$

t_1

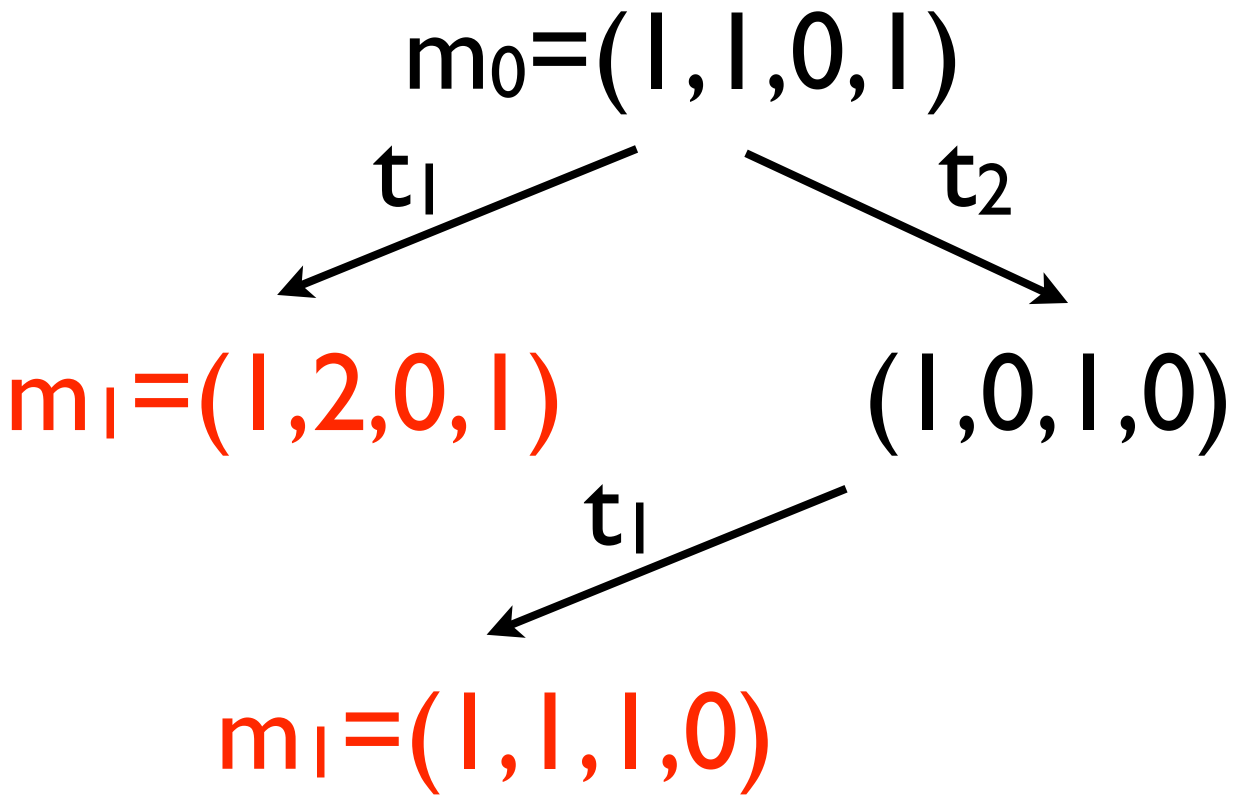
t_2

$m_1=(1,2,0,1)$

$(1,0,1,0)$



Tree saturation for PN



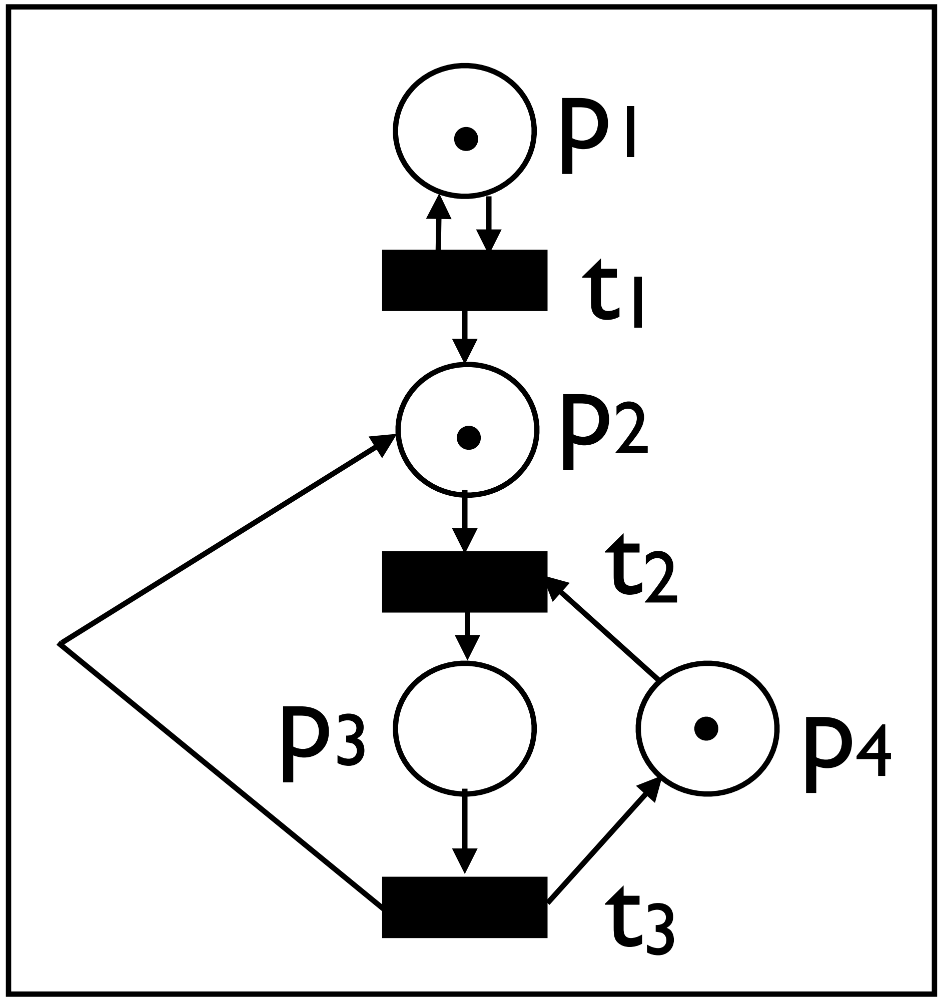
$m_0=(1,1,0,1)$

t_1 t_2

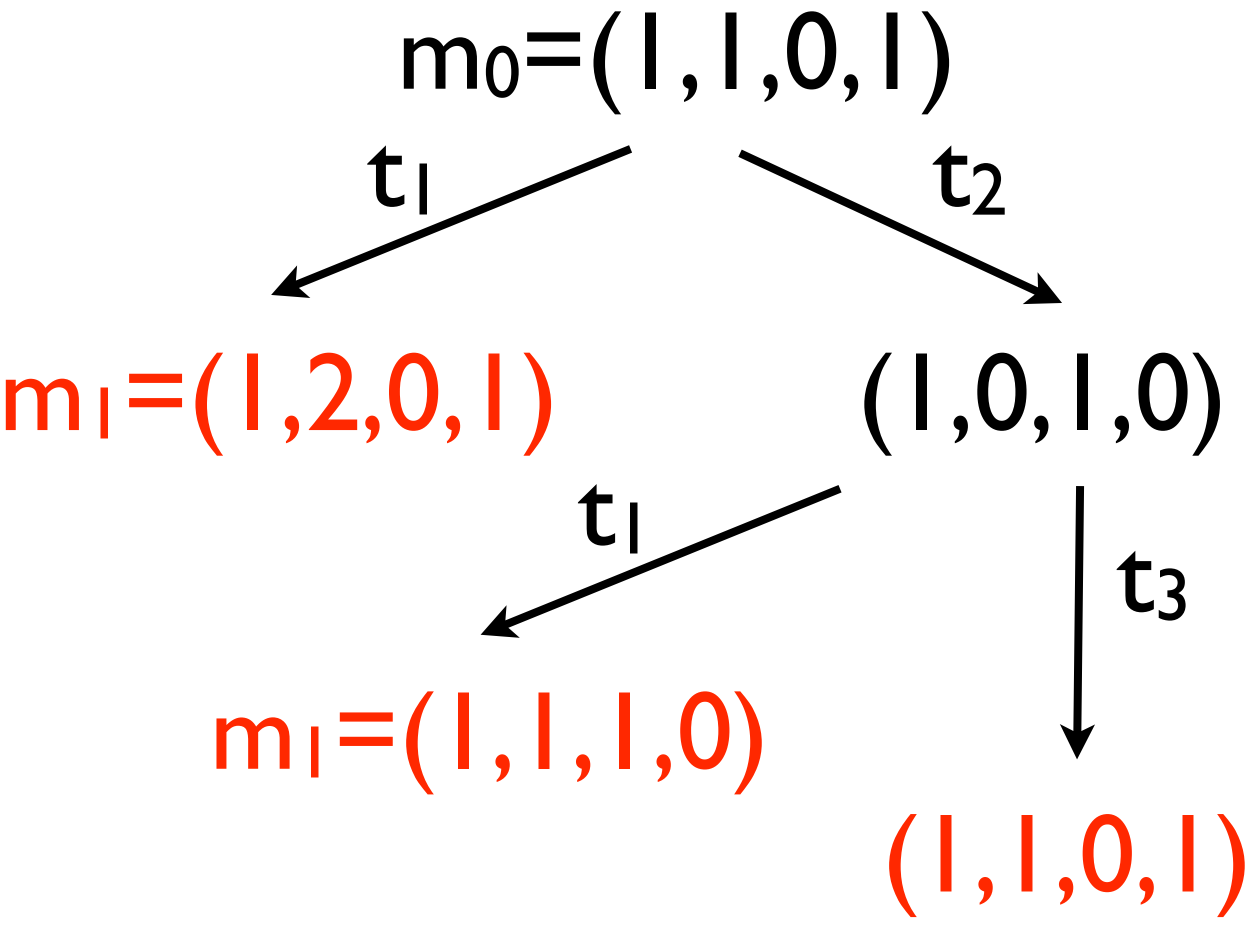
$m_1=(1,2,0,1)$ $(1,0,1,0)$

t_1

$m_1=(1,1,1,0)$



Tree saturation for PN



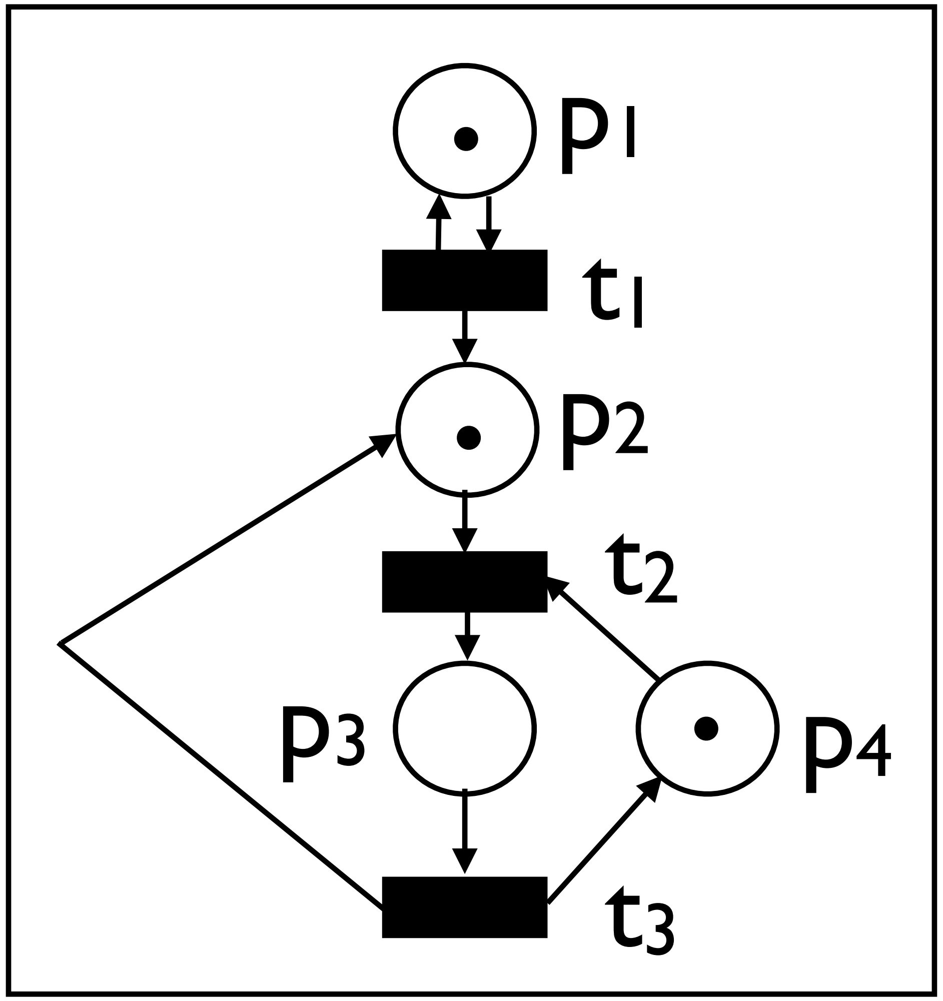
$m_0=(1,1,0,1)$

t_1 t_2

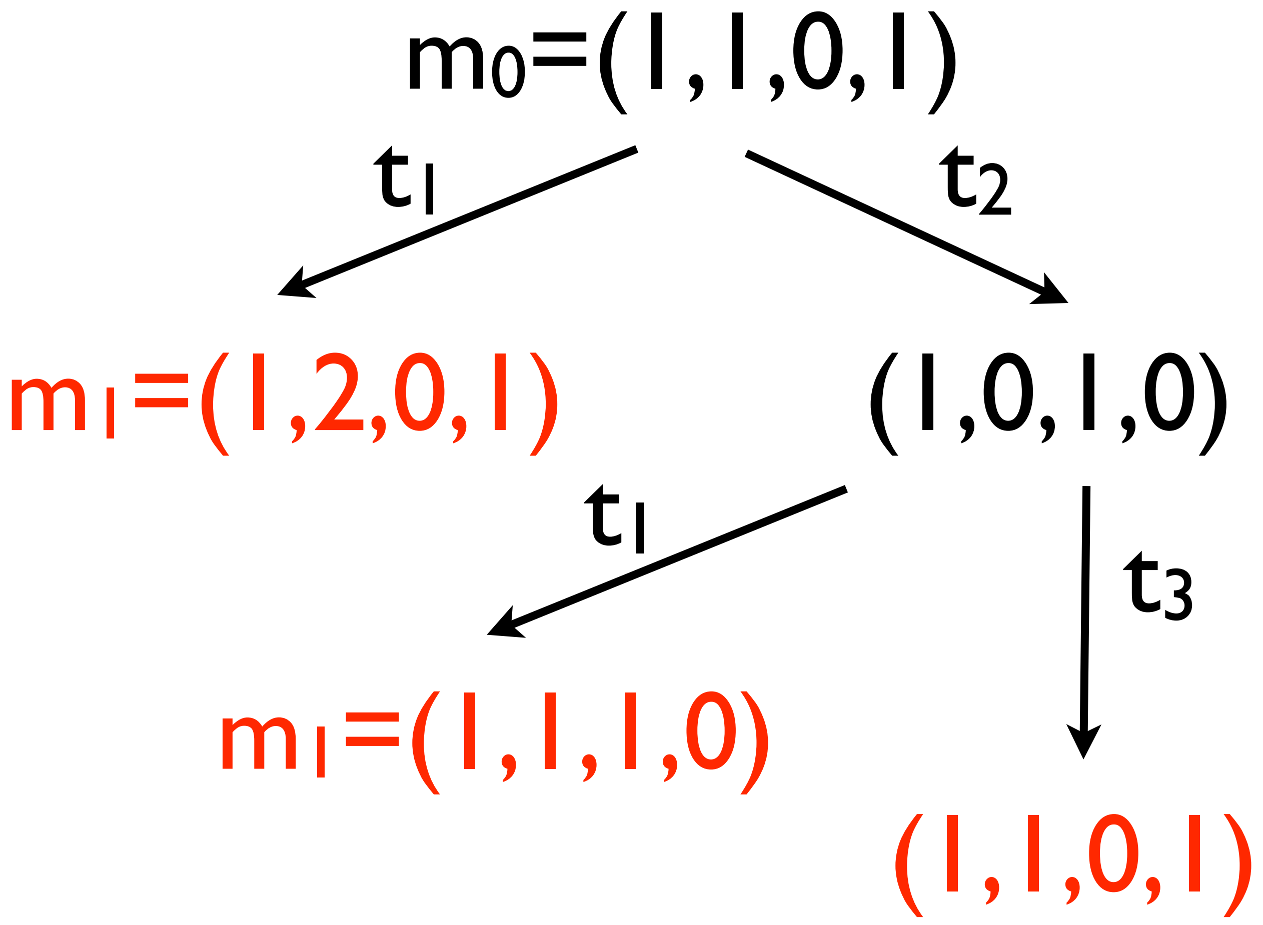
$m_1=(1,2,0,1)$ $(1,0,1,0)$

t_1 t_3

$m_1=(1,1,1,0)$ $(1,1,0,1)$



Tree saturation for PN



$m_0=(1,1,0,1)$

t_1

t_2

$m_1=(1,2,0,1)$

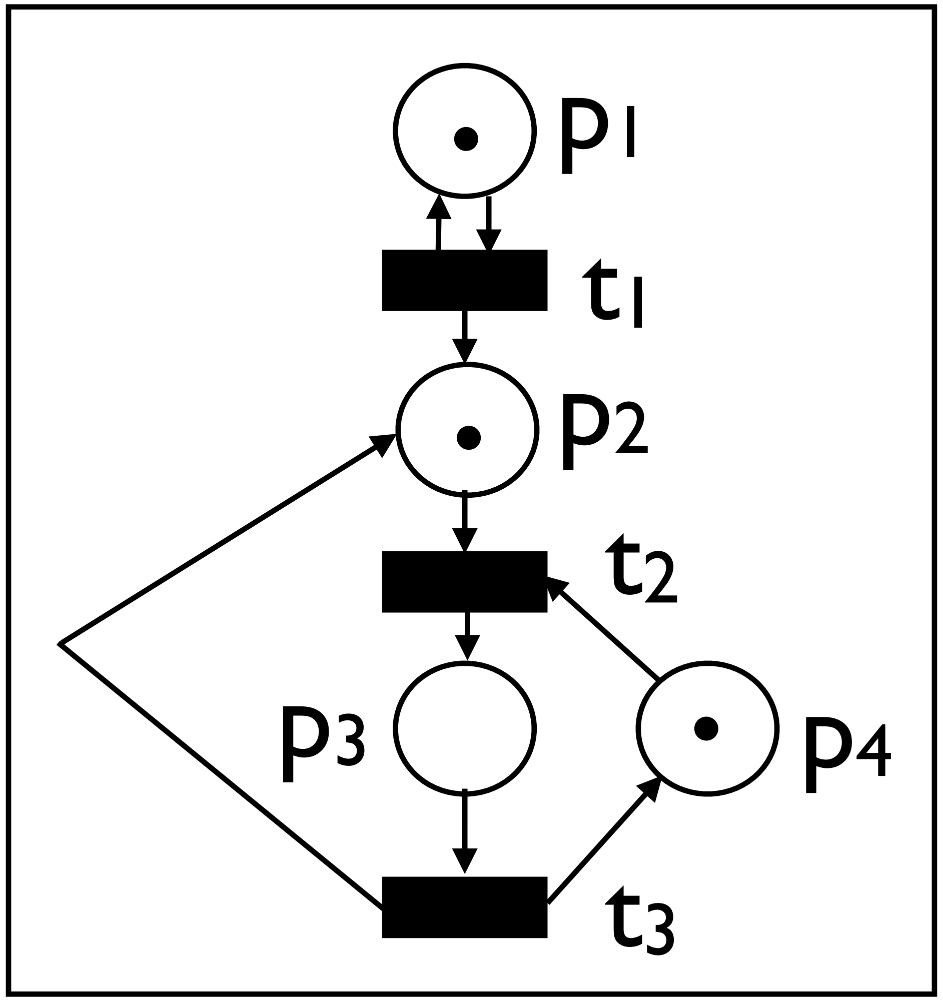
$(1,0,1,0)$

t_1

t_3

$m_1=(1,1,1,0)$

$(1,1,0,1)$



We are done !!!

Tree saturation for FEWSTS

- The stopping rule of the the tree saturation method is applicable to any FEWSTS.

  Indeed, on every infinite branch of the unfolding, we are guaranteed that there exist a node annotated with a state that is larger than one of its ancestor ! This is a direct consequence of WQO !

- So for every FEWSTS, there exists a finite tree, called the finite reachability tree, obtained by the tree saturation method:

  Theorem. A finite reachability tree exists and is effectively computable for any FEWSTS.

  (easy proof using WQO+König's lemma)

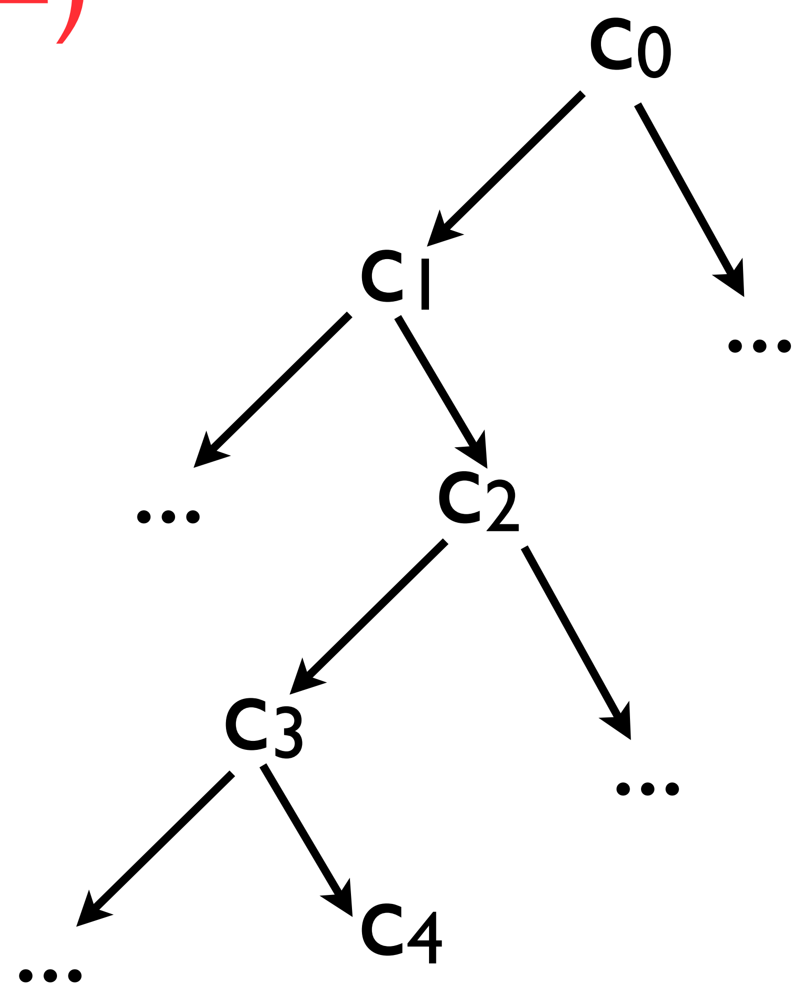
Properties of the finite reachability

- Clearly the leafs of the FRT(T) are nodes that either have no successors or contain a state which subsumes an ancestor. As a consequence, we have the following theorem.
- **Theorem.** T=(C,c_0,⟹≤) has a non-terminating computation starting in c_0 iff FRT(T) contains a subsumed node.

Properties of the finite reachability

- **Theorem.** T=(C,c_0,⟹≤) has a non-terminating computation starting in c_0 iff FRT(T) contains a subsumed node.

(⟸)



and $c_1 \leq c_4$

Then clearly $c_0(c_1c_2c_3c_4)^\omega$ is
an non-terminating computation in T

Properties of the finite reachability

- **Theorem.** $T=(C,c_0,\Longrightarrow\leq)$ has a non-terminating computation starting in c_0 iff FRT(T) contains a subsumed node.

$(\Longrightarrow)$

Let c_0 c_1 c_2 ... c_n ... be a non-terminating computation in T.

This computation has a prefix which labels a branch in FRT(T).

This branch must end in a node that subsumes an ancestor (it can not be a node with no successor).

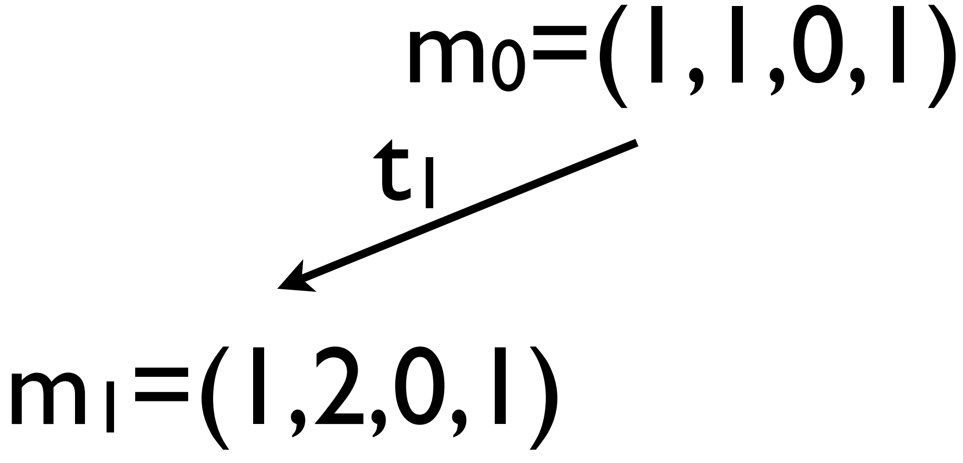
The non-terminating computation problem

- **Theorem.** The non-terminating computation problem is decidable for the entire class of FEWSTS.

Karp and Miller tree for PN

- The Finite Reachability Tree should not be confused with The Karp and Miller tree for Petri Net.
- KM Tree=Unfolding+Accelerations+Stopping rules.
- KM Tree is an procedure for computing an effective representation of the set ↓Reach(N) of a Petri net N.

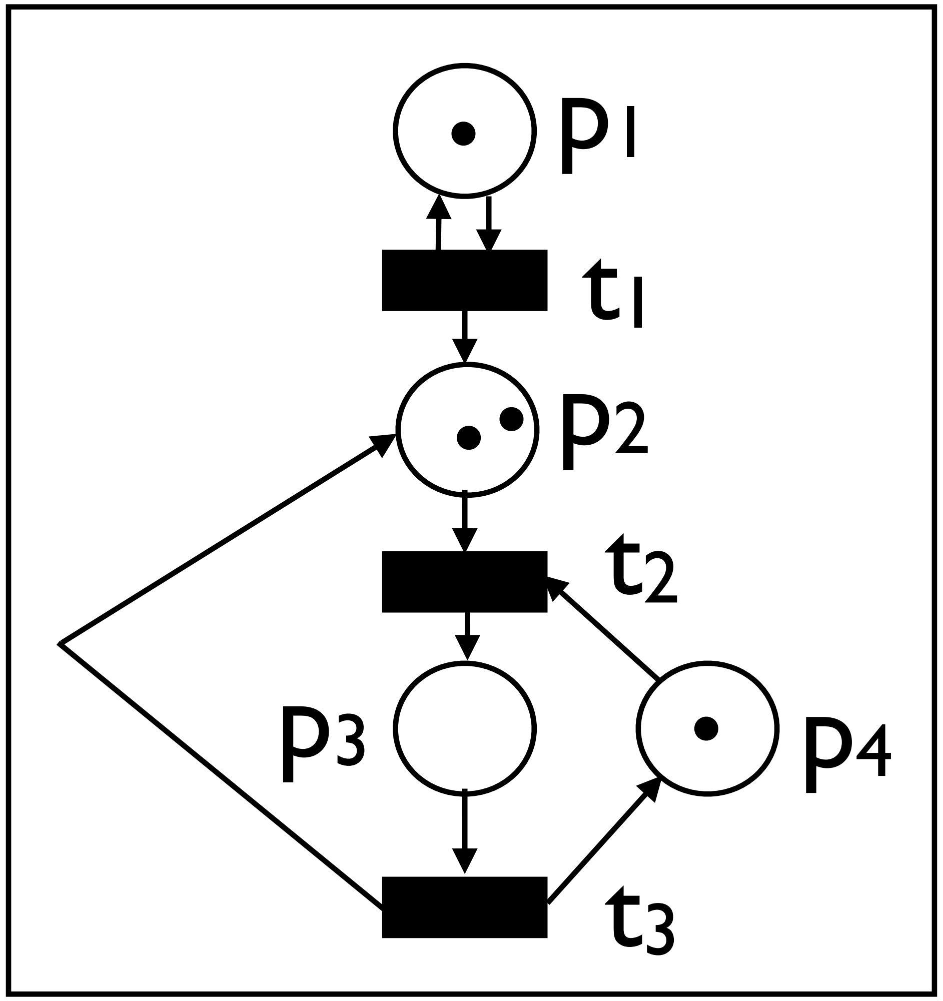
KM tree for PN



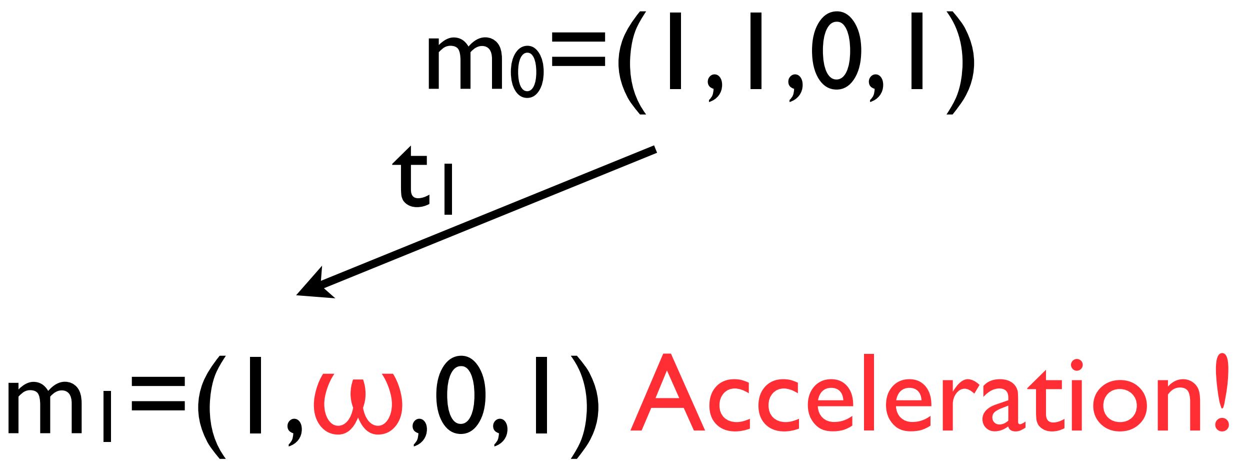
$m_0=(1,1,0,1)$

t_1

$m_1=(1,2,0,1)$



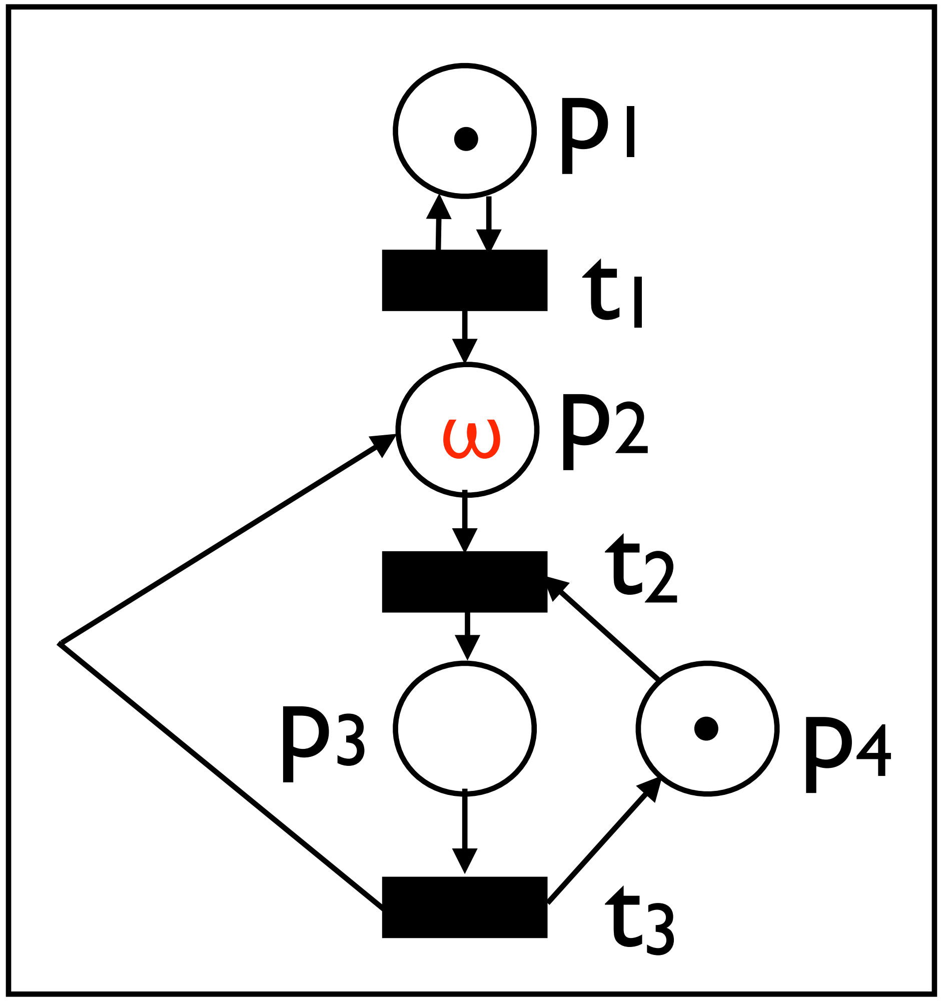
KM tree for PN



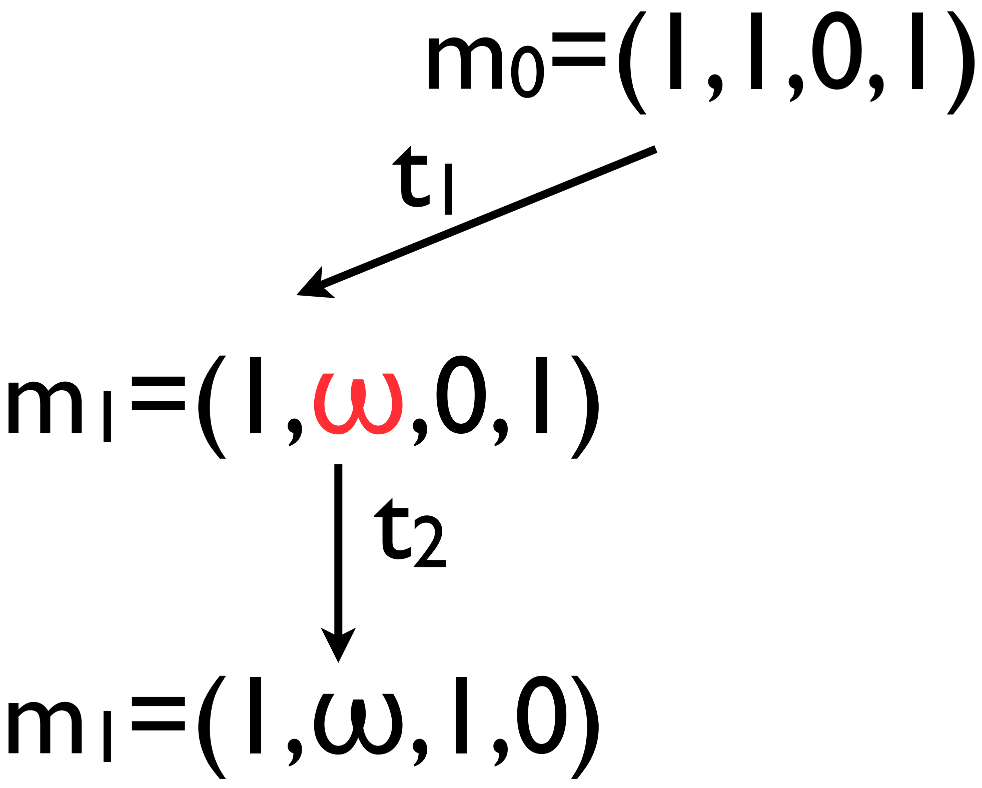
$m_0=(1,1,0,1)$

t_1

$m_1=(1,\omega,0,1)$ Acceleration!



KM tree for PN



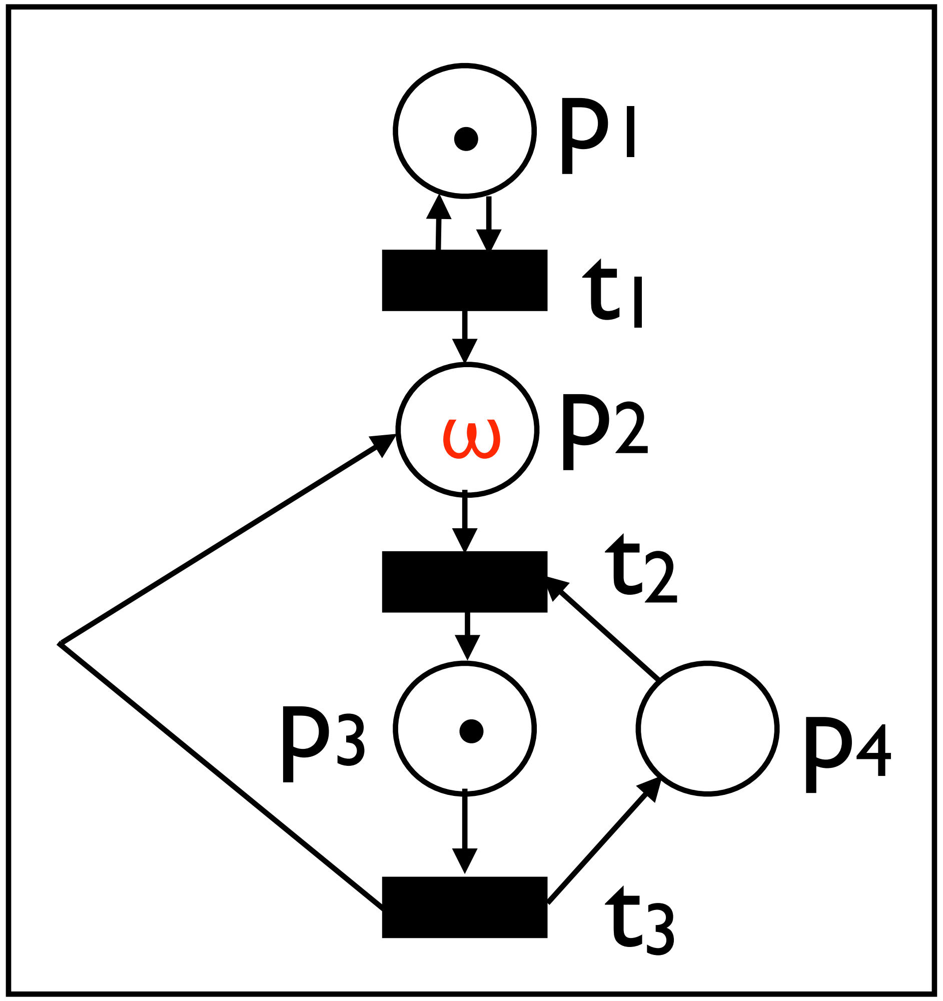
$m_0=(1,1,0,1)$

t_1

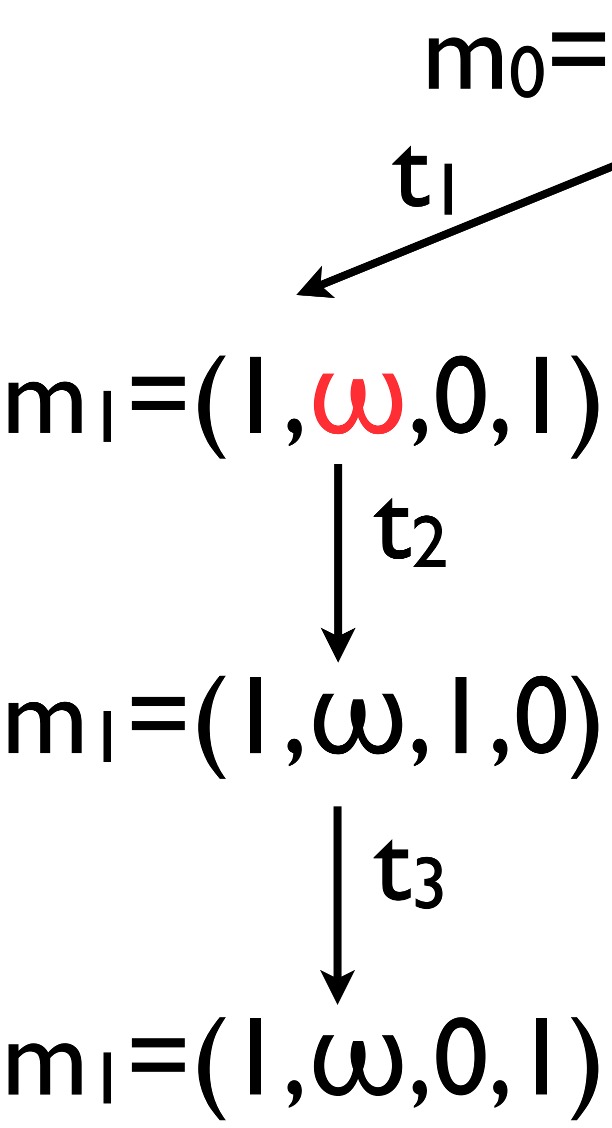
$m_1=(1,\omega,0,1)$

t_2

$m_1=(1,\omega,1,0)$



KM tree for PN



$m_0=(1,1,0,1)$

$\xrightarrow{t_1}$

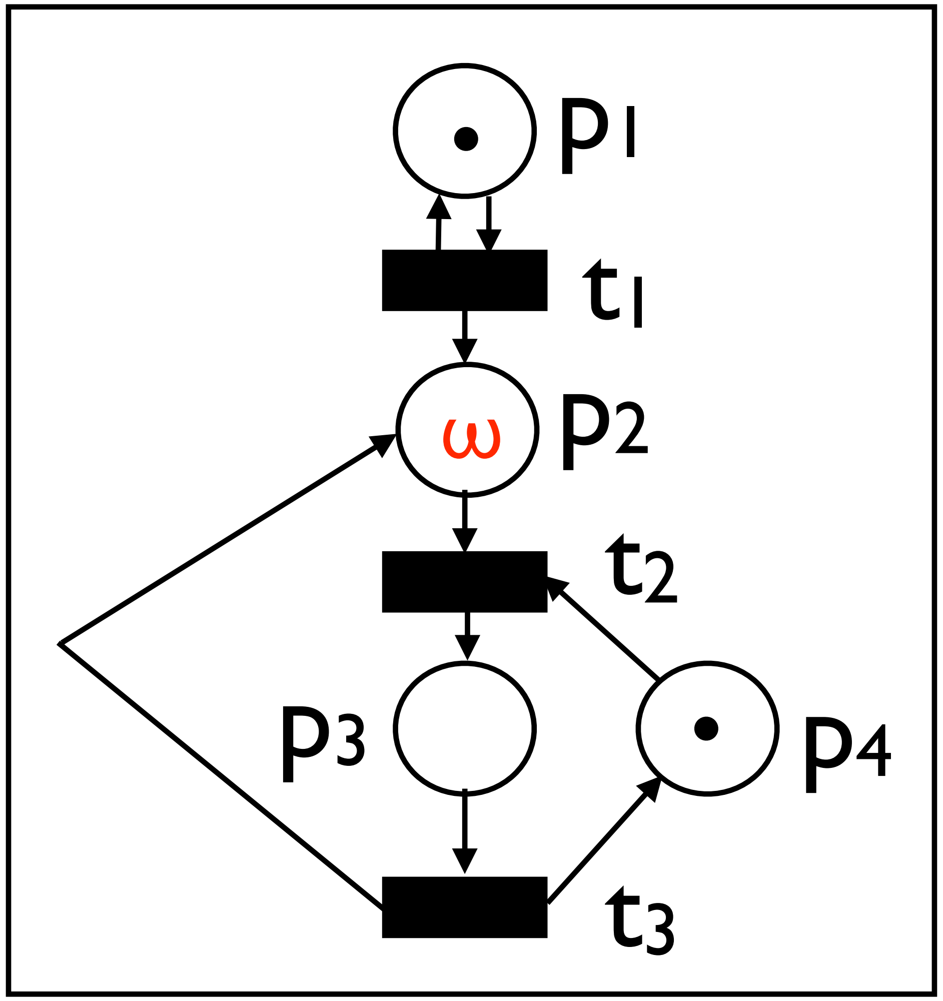
$m_1=(1,\omega,0,1)$

$\xrightarrow{t_2}$

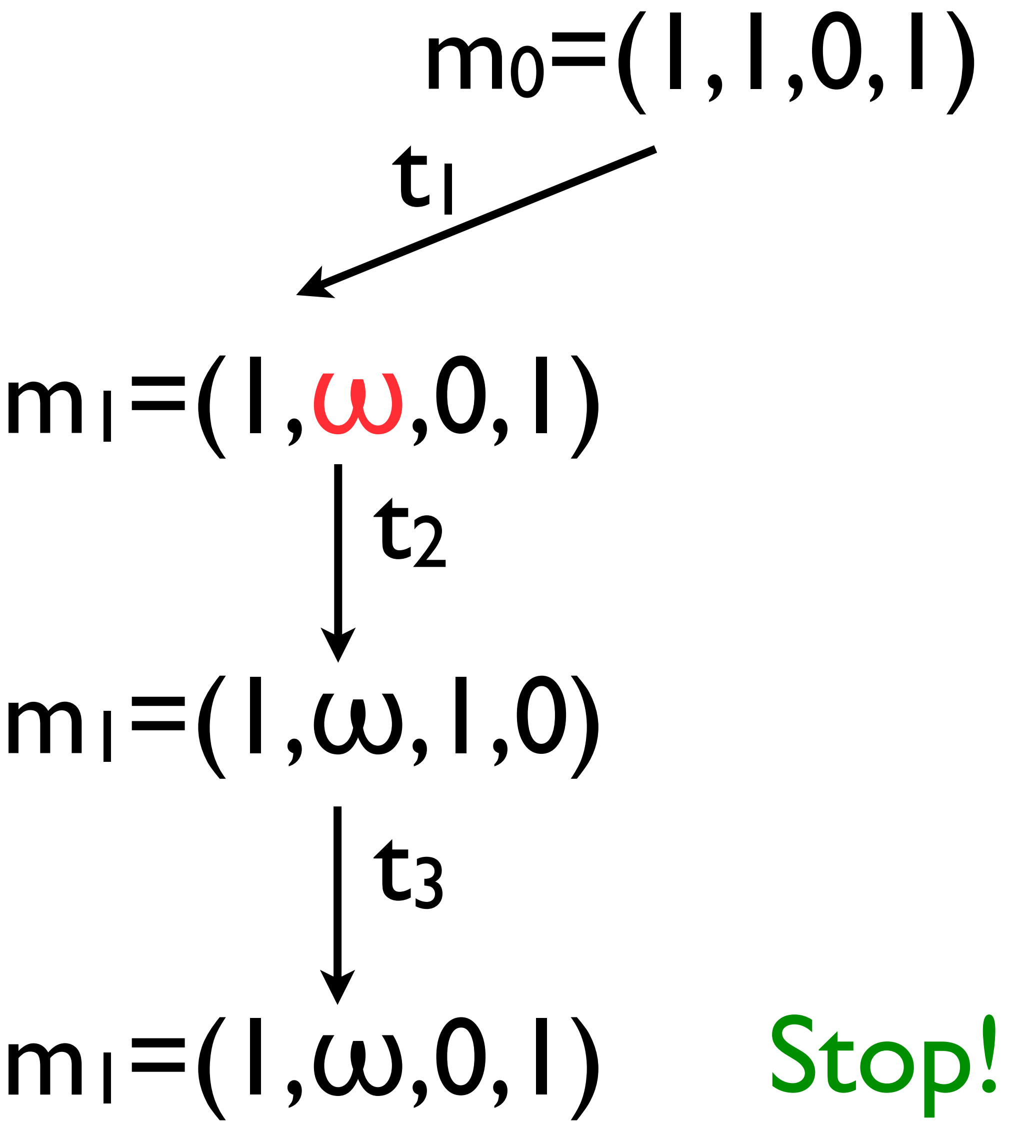
$m_1=(1,\omega,1,0)$

$\xrightarrow{t_3}$

$m_1=(1,\omega,0,1)$



KM tree for PN



$m_0=(1,1,0,1)$

t_1

$m_1=(1,\omega,0,1)$

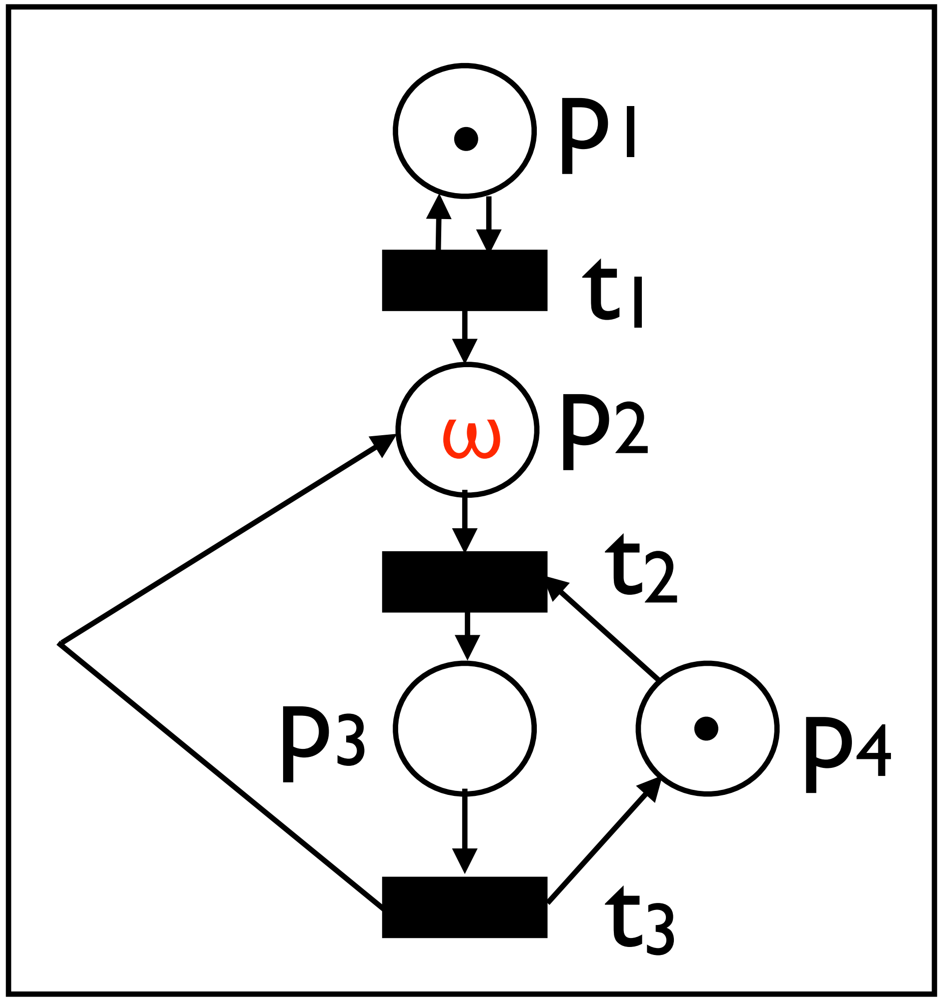
t_2

$m_1=(1,\omega,1,0)$

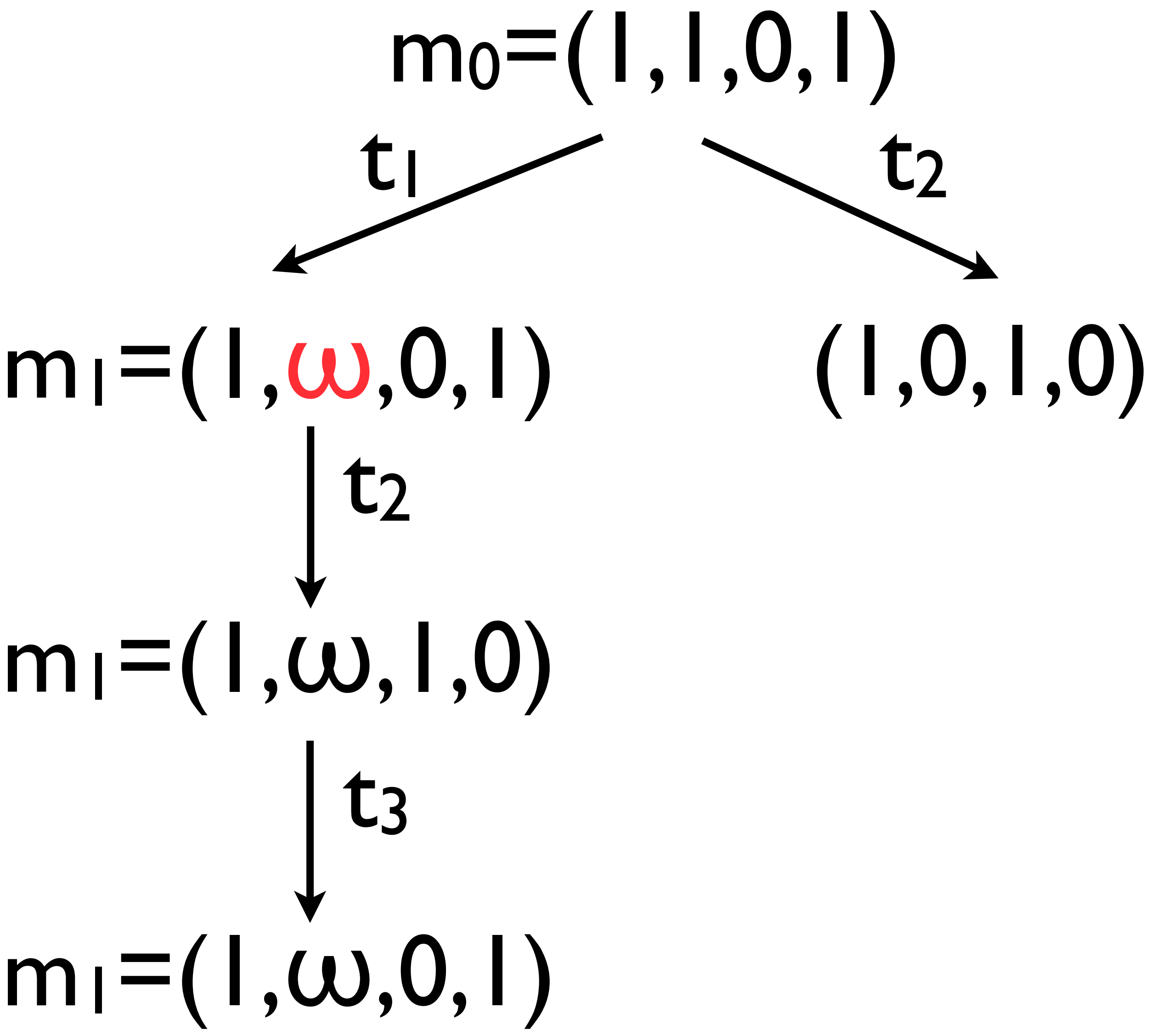
t_3

$m_1=(1,\omega,0,1)$

Stop!



KM tree for PN



$m_0=(1,1,0,1)$

t_1

t_2

$m_1=(1,\omega,0,1)$

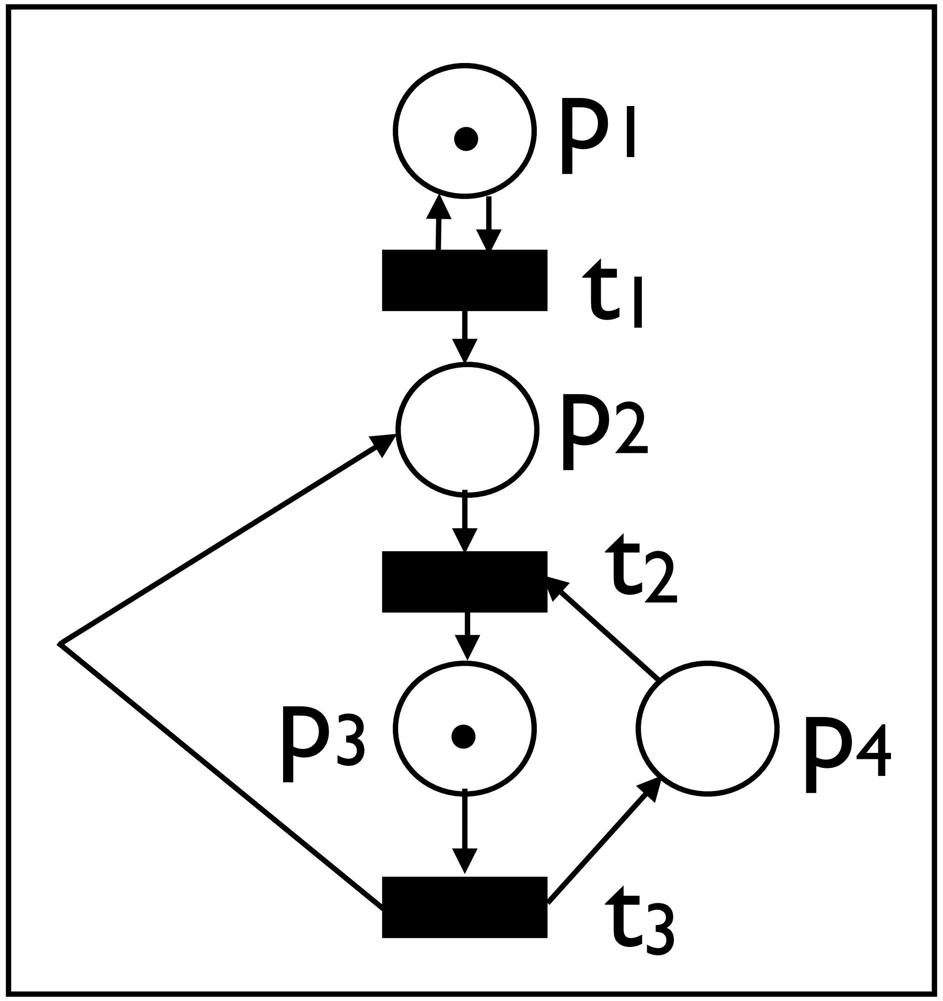
$(1,0,1,0)$

t_2

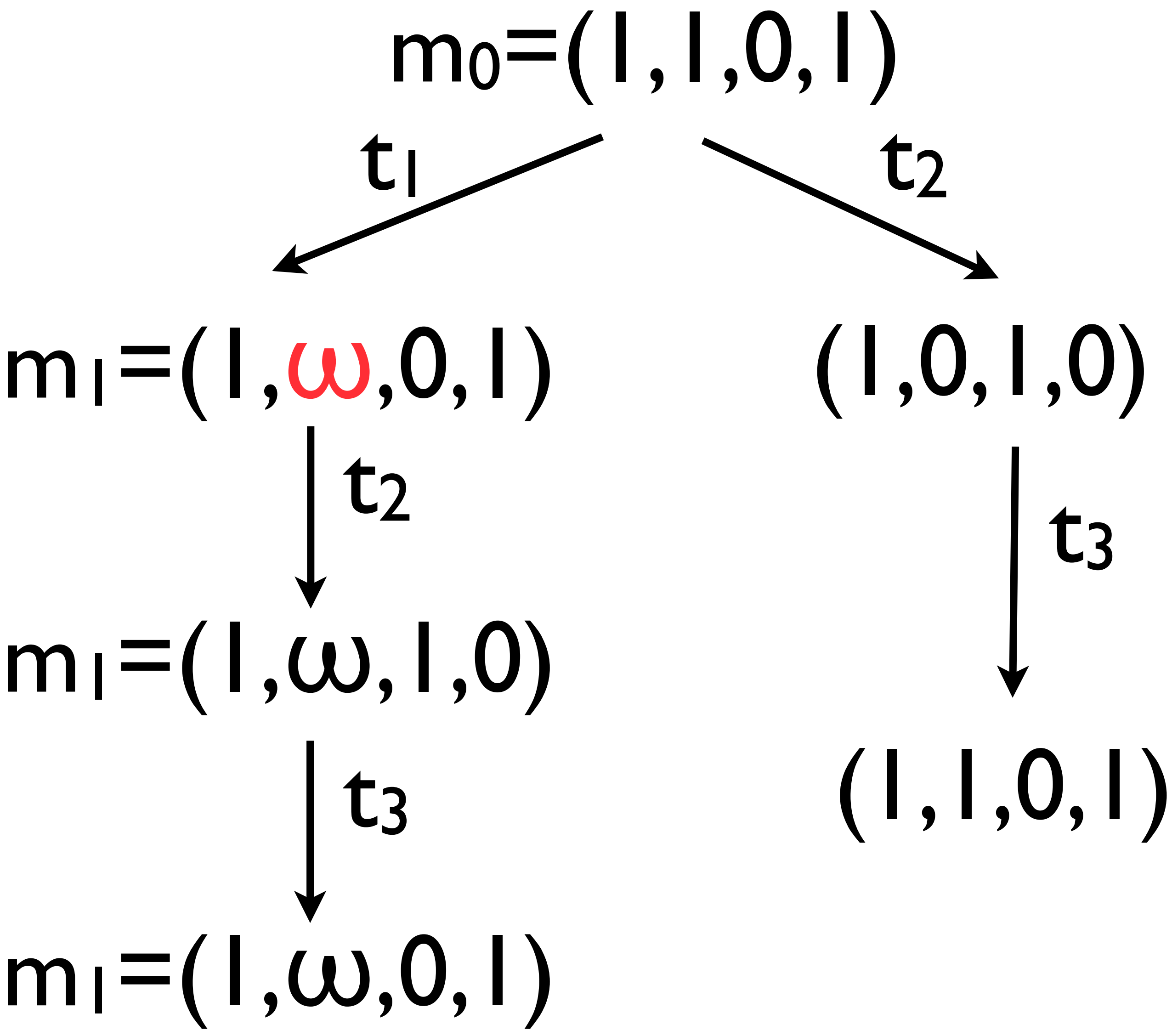
$m_1=(1,\omega,1,0)$

t_3

$m_1=(1,\omega,0,1)$



KM tree for PN



$m_0=(1,1,0,1)$

t_1

t_2

$m_1=(1,\omega,0,1)$

$(1,0,1,0)$

t_2

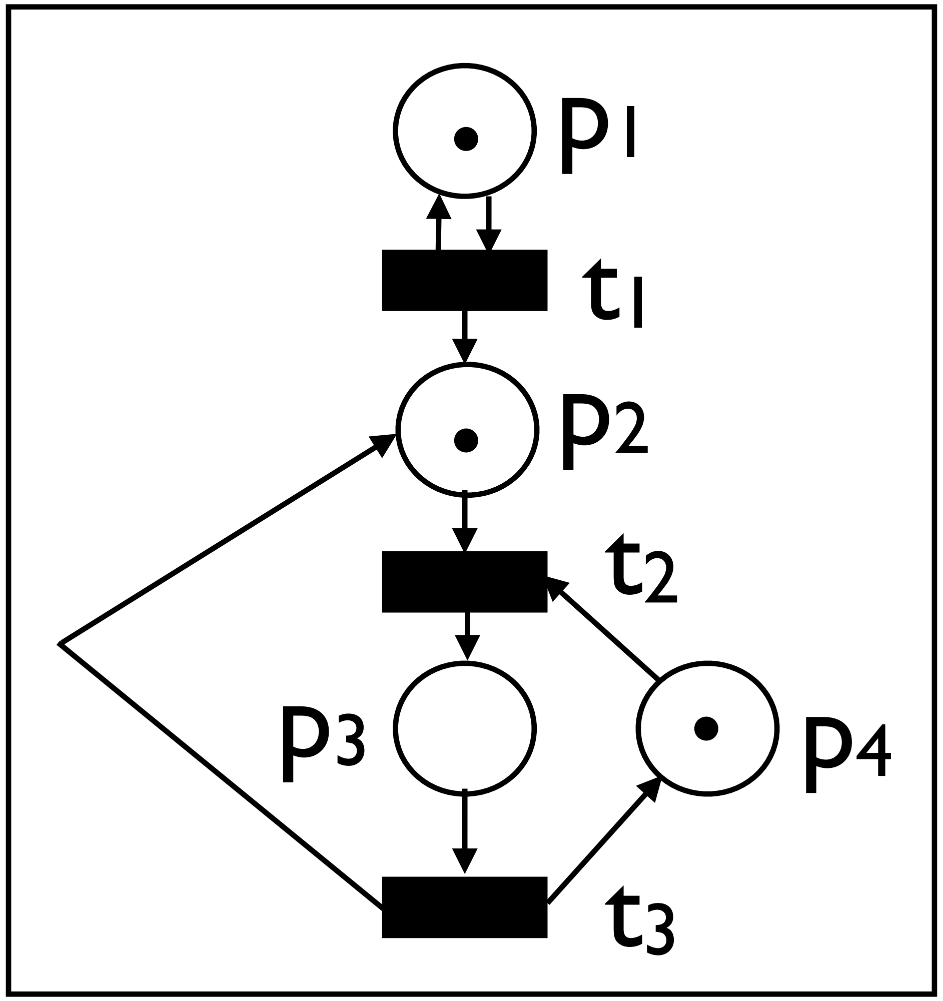
t_3

$m_1=(1,\omega,1,0)$

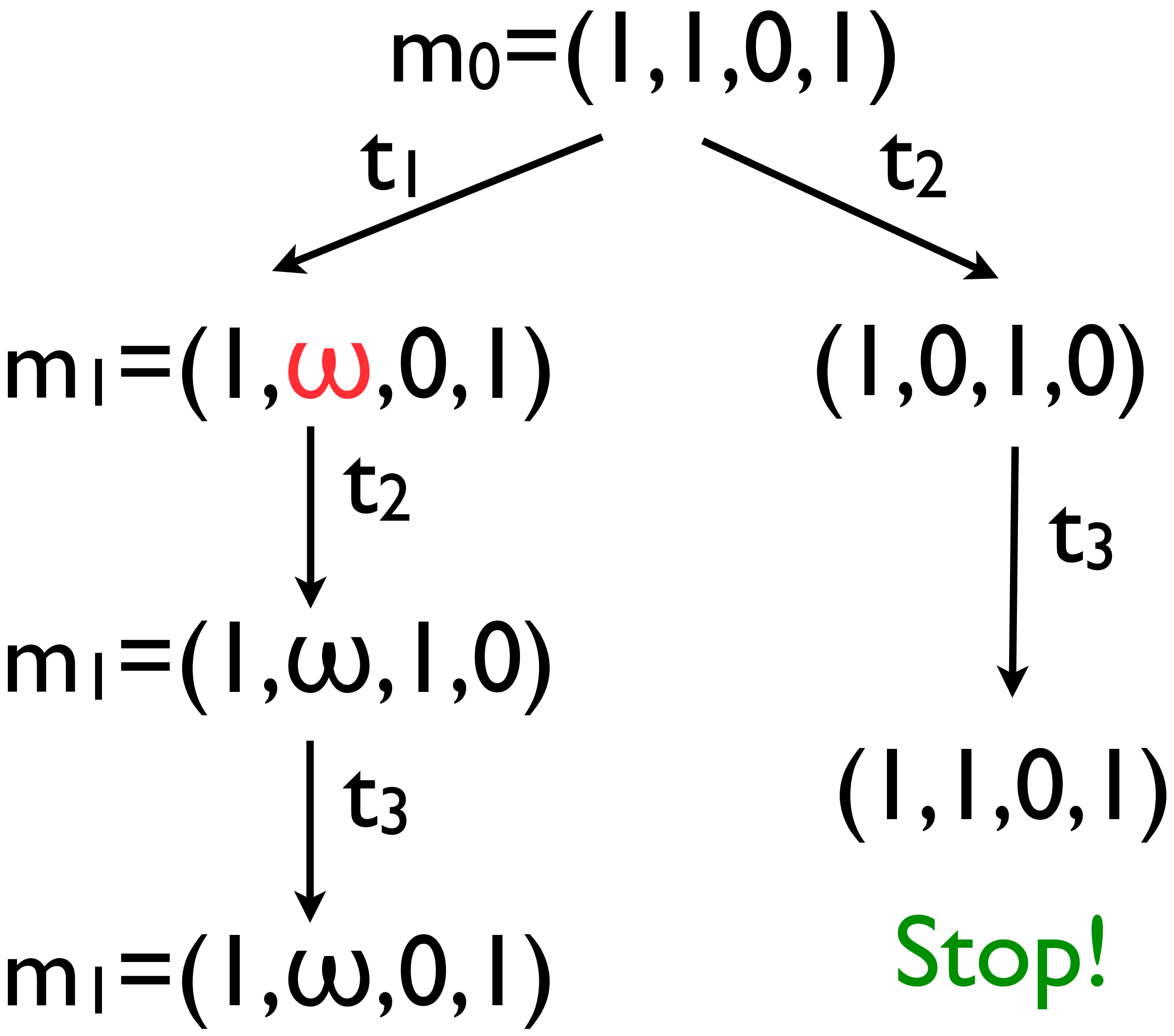
$(1,1,0,1)$

t_3

$m_1=(1,\omega,0,1)$



KM tree for PN



$m_0=(1,1,0,1)$

t_1 t_2

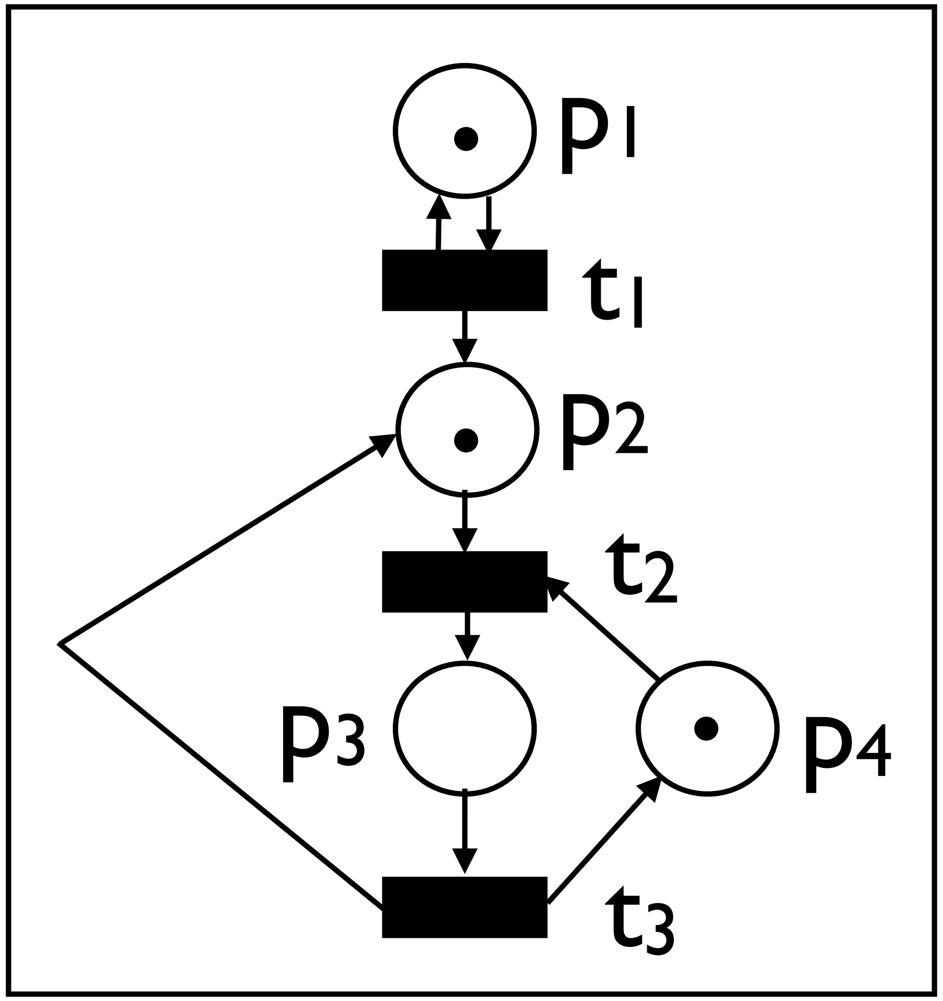
$m_1=(1,\omega,0,1)$ $(1,0,1,0)$

t_2 t_3

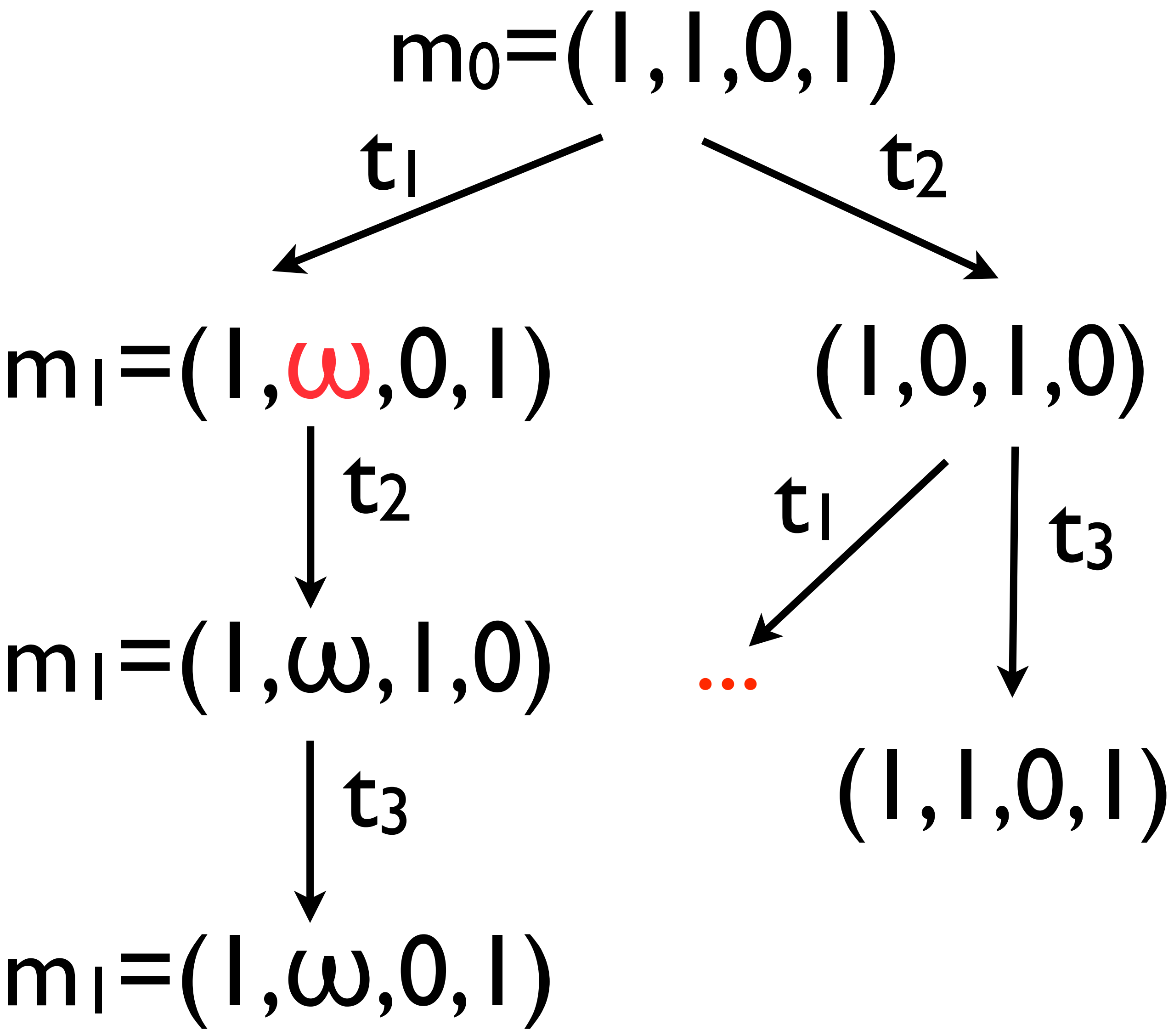
$m_1=(1,\omega,1,0)$ $(1,1,0,1)$

t_3 Stop!

$m_1=(1,\omega,0,1)$



KM tree for PN



$m_0=(1,1,0,1)$

t_1

t_2

$m_1=(1,\omega,0,1)$

$(1,0,1,0)$

t_2

t_1

t_3

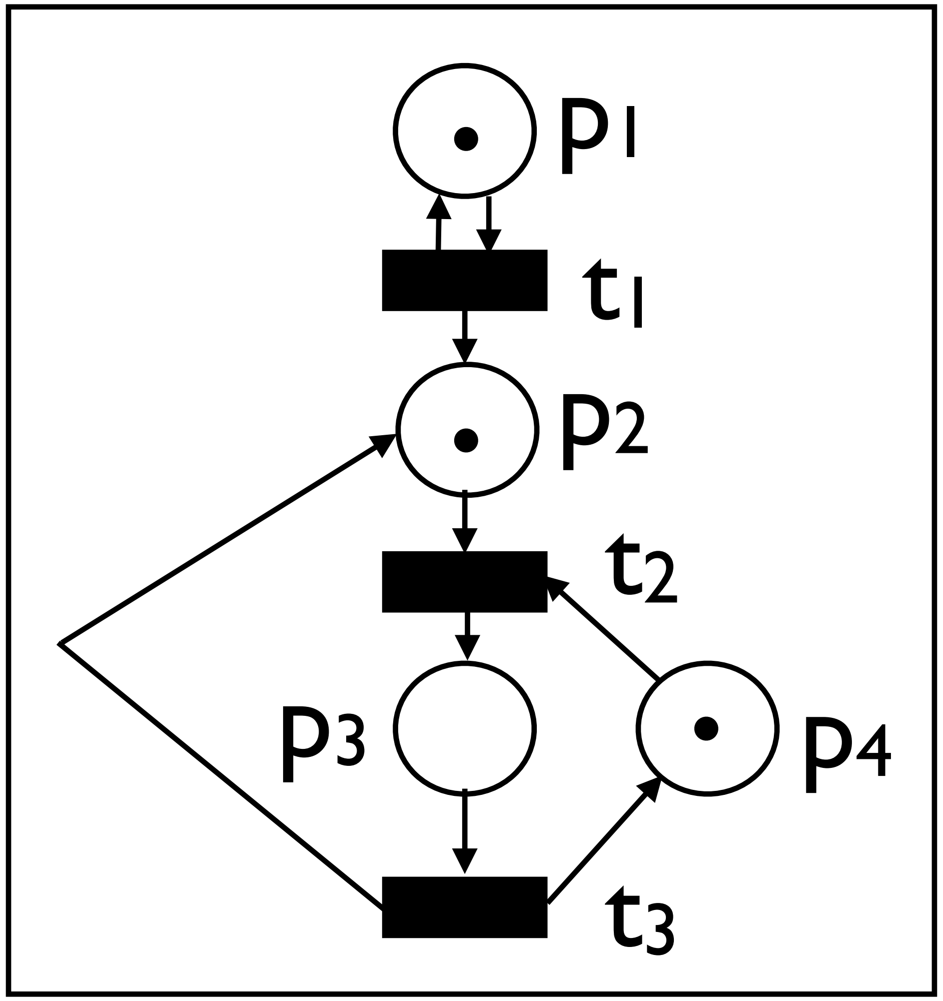
$m_1=(1,\omega,1,0)$

...

$(1,1,0,1)$

t_3

$m_1=(1,\omega,0,1)$



Karp and Miller tree for PN

- The Finite Reachability Tree should not be confused with The Karp and Miller tree for Petri Net.
- KM Tree=Unfolding+Accelerations+Stopping rules.
- KM Tree is an procedure for computing an effective representation of the set $\downarrow$Reach(N) of a Petri net N.
- $\downarrow$Reach(N) allows for deciding coverability:

  $\exists m' \geq m \bullet m' \in Post^{*}(m0)$ **iff** $m \in \downarrow Reach(N)$.

- $\downarrow$Reach(N) allows for deciding place boundedness:

  p is bounded in N **iff** $\exists k \in \mathbb{N} \bullet \forall m \in \downarrow Reach(N) \bullet m(p) \leq k$.

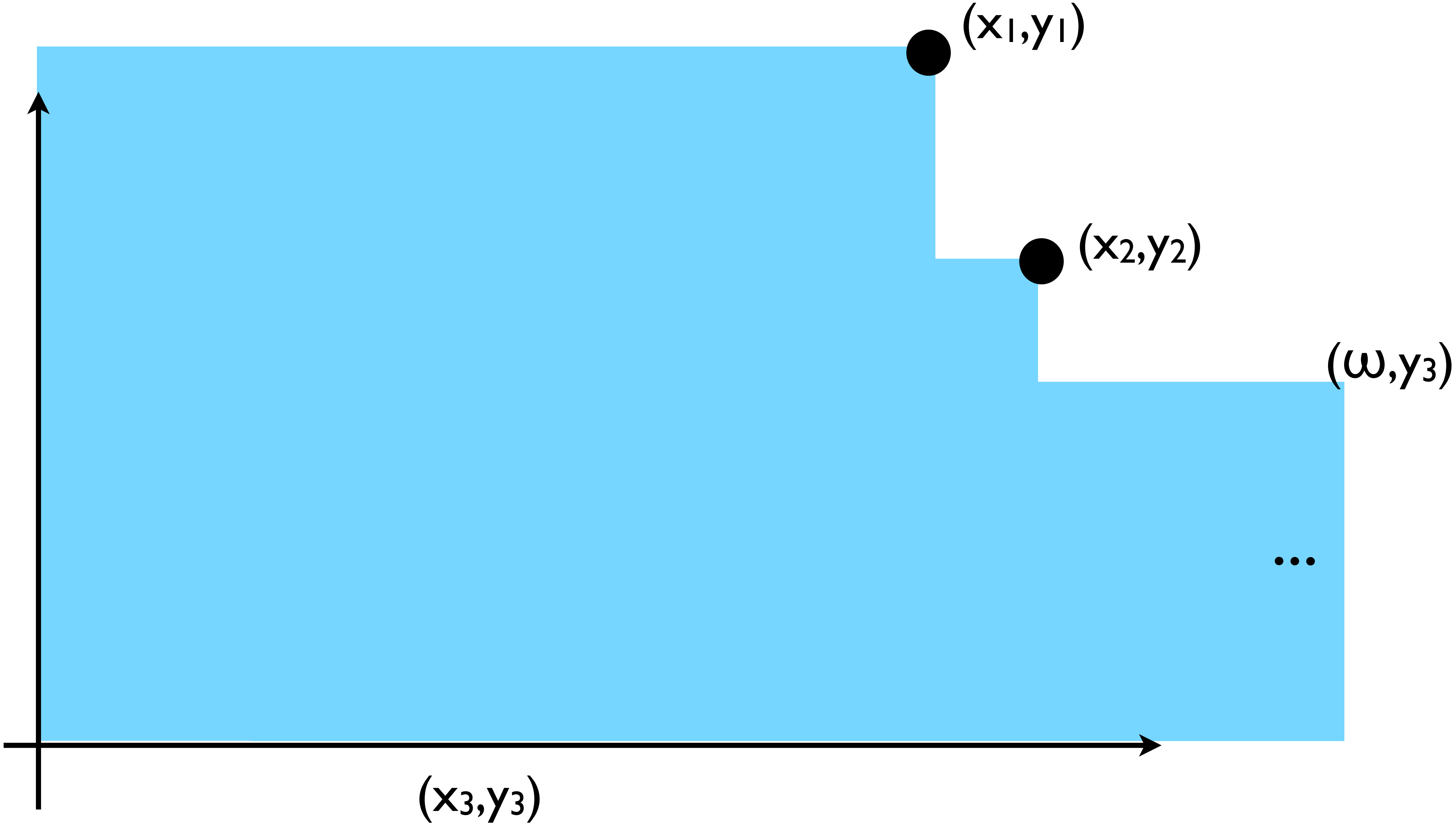
ω-Markings and downward closed sets in $(\mathbb{N}^k,\preccurlyeq)$

- A ω-marking is a function $m : P \rightarrow \mathbb{N} \cup \{\omega\}$.
- ω="any number of tokens".
- A ω-marking m represents a set of "plain" markings:

  Let m be an ω-marking

  $\downarrow m = \{\ m' \in [P \rightarrow \mathbb{N}] \mid \forall p \in P : m'(p) \leq m(p)\}$
- **Theorem.** For any downward-closed set of marking D, there exists a finite set of ω-marking M such that ↓M=D.

Downward-closed sets in $(\mathbb{N}^k,\preccurlyeq)$



DGen(D)=$\{(x_1,y_1), (x_2,y_2)\ (\omega,y_3)\}$ is a finite generator for D.

↓Reach(N) is not constructible for EPN

- We have seen that:
  - ↓Reach(N) is sufficient to decide place boundedness
  - Place boundedness is undecidable for EPN !
- So, ↓Reach(N) is not computable for EPN !

↓Reach(N) is not constructible for EPN

- We have seen that:
  - ↓Reach(N) is sufficient to decide place boundedness
  - Place boundedness is undecidable for EPN !
- So, ↓Reach(N) is not computable for EPN !

Still, can we have a forward algorithm for coverability ?

Expand-Enlarge and Check

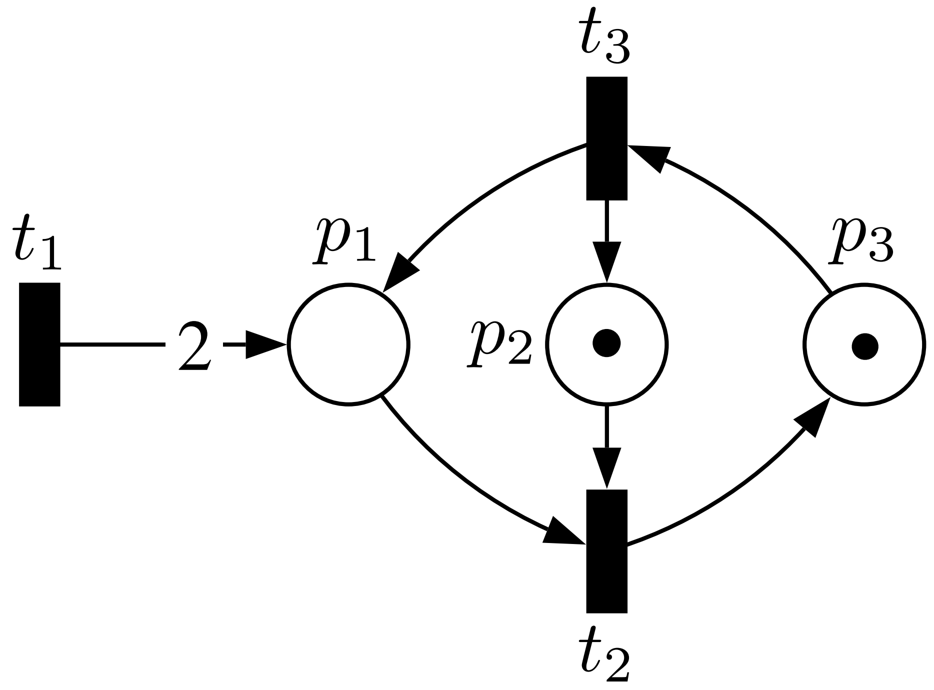
Forward algorithm for coverability of WSTS

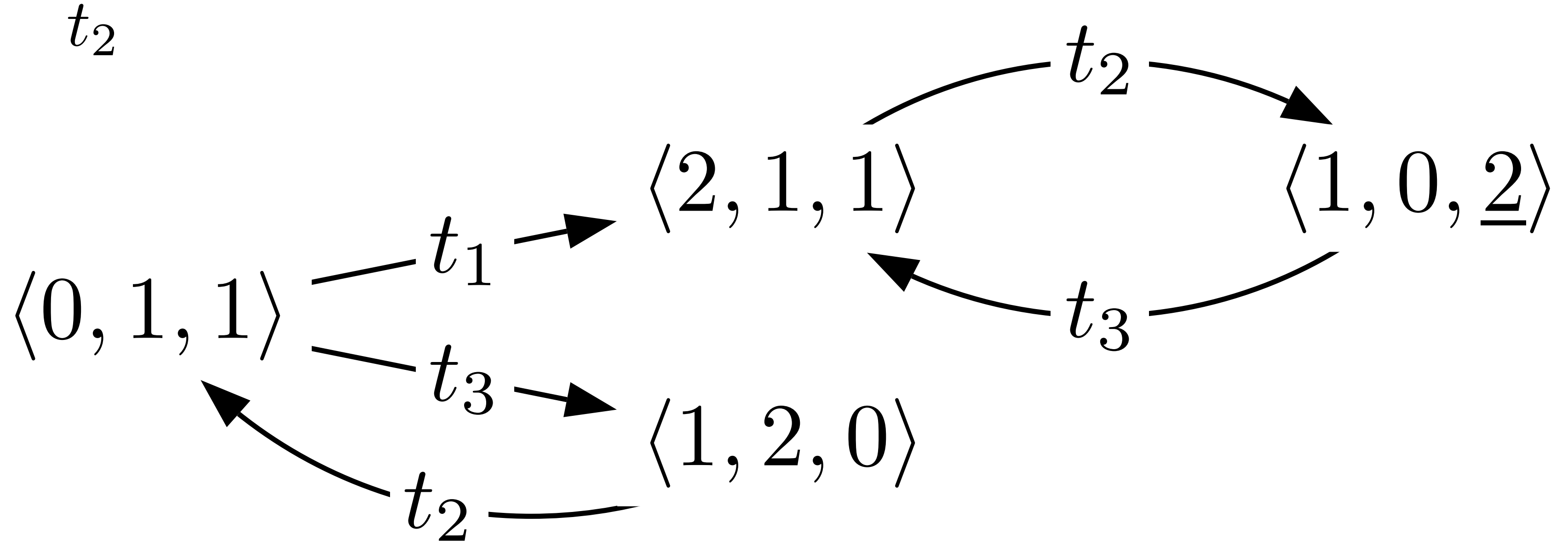
- We have just seen that ↓Reach(N) has always a finite representation but it is not effectively computable.
- Nevertheless, our solution for a forward algorithm for deciding coverability of EPN will rely on the existence of this finite representation.

Under-approx of $\downarrow$Reach(S)

- Let $N=(P,T,m_0)$ be an extended Petri net and $T(N)=([P\rightarrow\mathbb{N}],m_0,\Longrightarrow,\preccurlyeq)$ its associated WSTS.
- Let $k\in\mathbb{N}$, and the two following families of finite sets:
  C_k be the set of markings $\{ m \mid m \in P\rightarrow[0..k] \}\cup\{m_0\}$
  L_k be the set of ω-markings $\{ m \mid m \in P\rightarrow[0..k]\cup\{\omega\}\}\cup\{m_0\}$.
- $\text{UnderApprox}(N,k)=(C_k,m_0,\Longrightarrow_{\text{under}})$ where:
  - $\Longrightarrow_{\text{under}}=\Longrightarrow\cap C_k\times C_k$ i.e., transitions that leads to markings with more than k tokens are discarded.
- **Lemma**. $\downarrow\text{Reach}(\text{UnderApprox}(N,k))\subseteq\downarrow\text{Reach}(N)$.

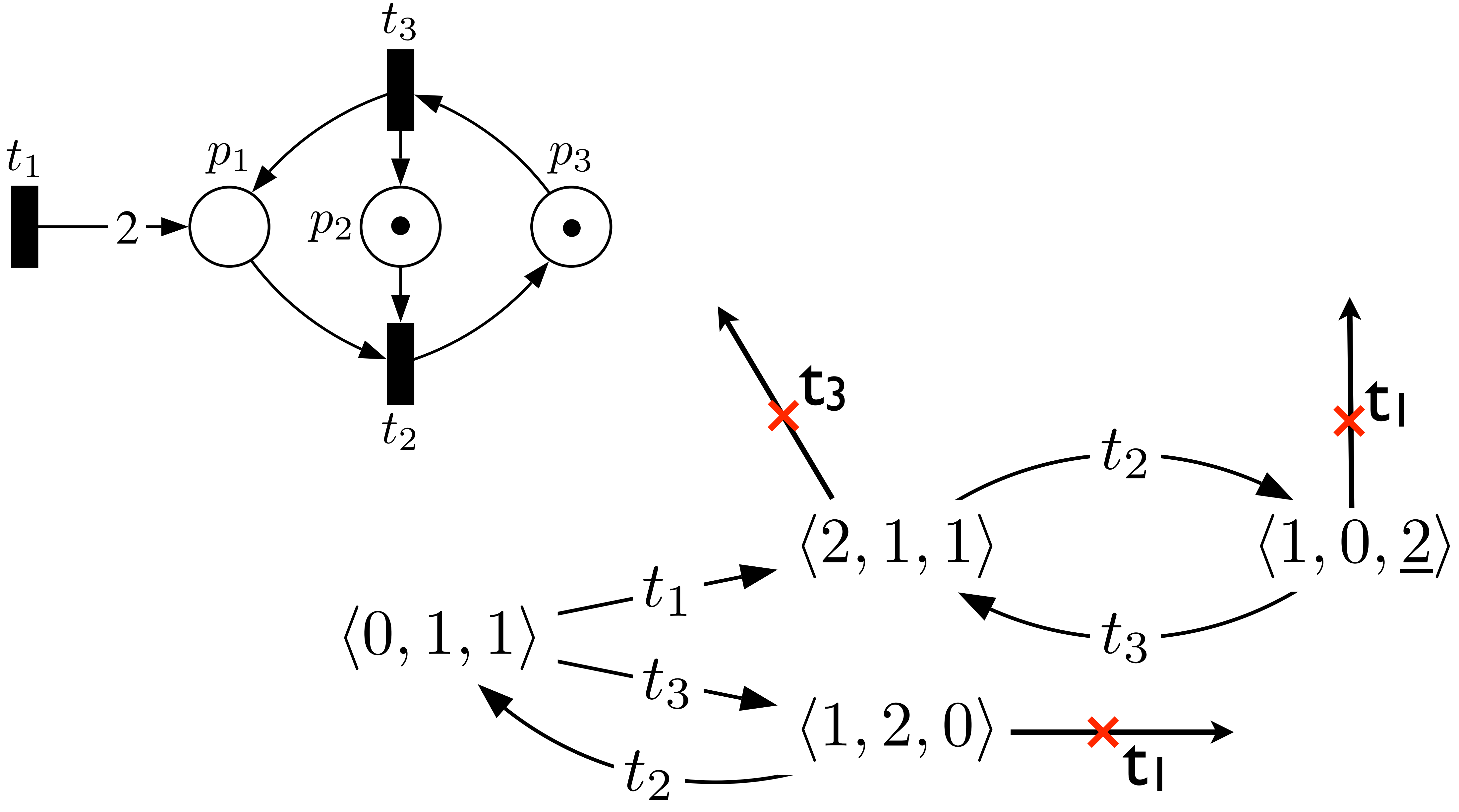
An example





Under(N,2)

An example



Under(N,2)

Over-approx of Cover(S)

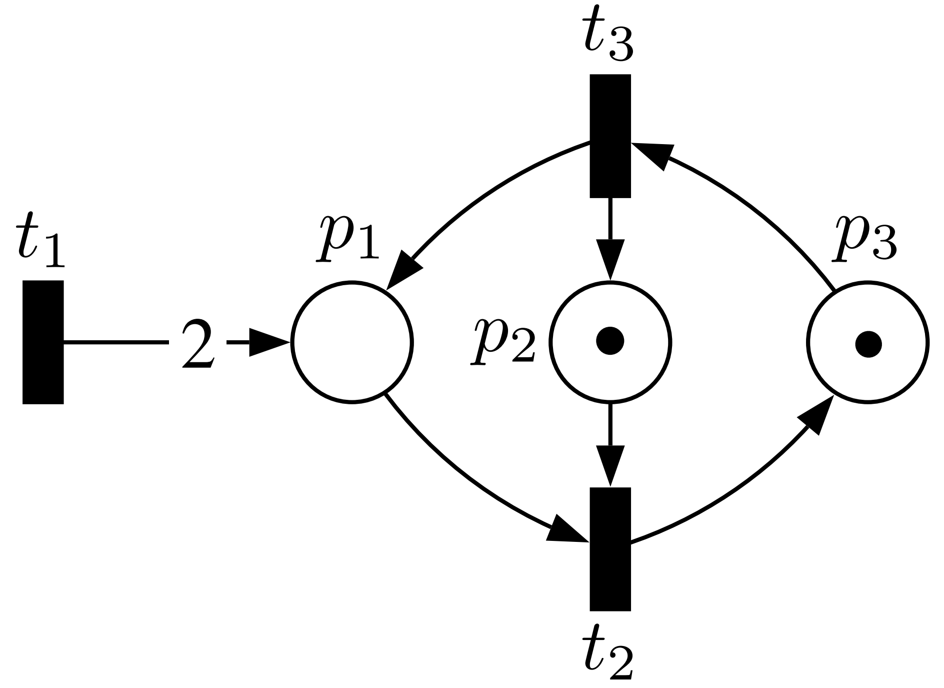
- We define Post$^{\#k}$: $L_k \rightarrow 2^{L_k}$ as follows:

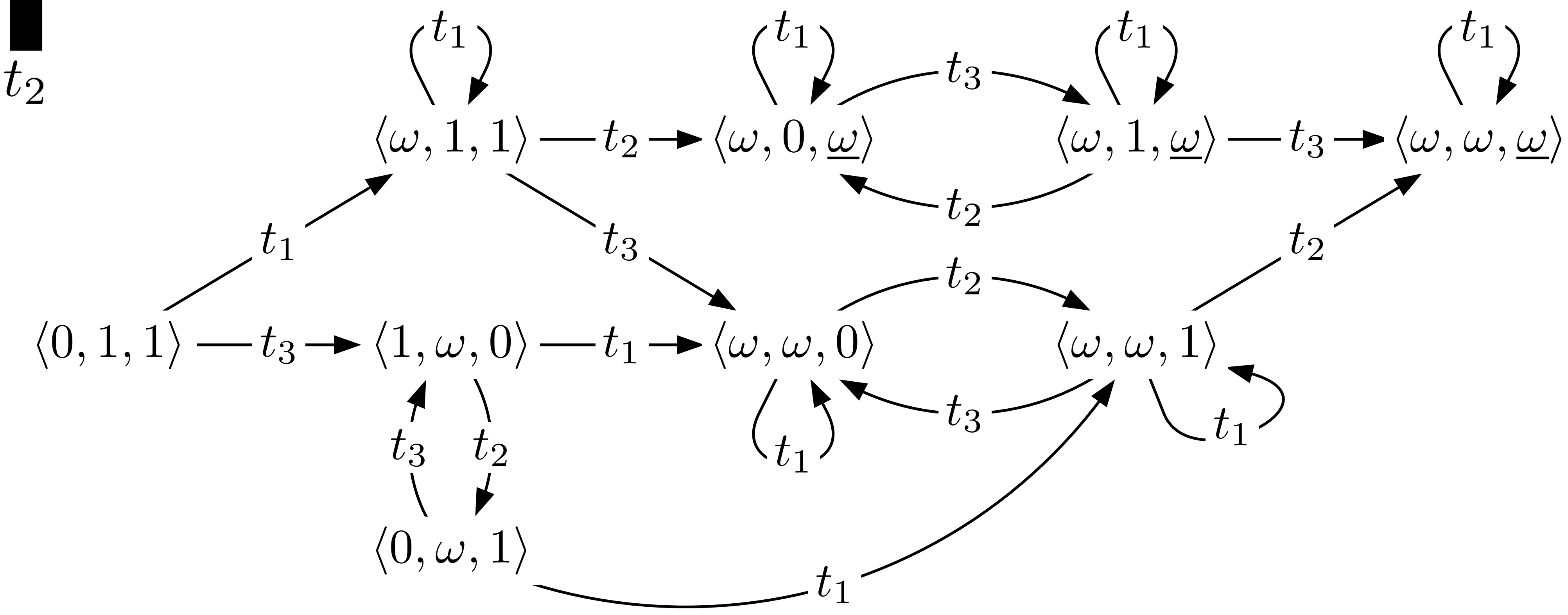
  Post$^{\#k}$(m)
  $=\{m' \in L_k \mid m \Longrightarrow_\omega m'$ or
  $\neg(m \Longrightarrow_\omega m')$ and $\exists m'' \bullet m \Longrightarrow_\omega m'' : m' = \textbf{enlarge}(m'',k)\}$

  where **enlarge**(m'',k)(p) = $\begin{cases} m''(p) & \text{if } m'(p) \leq k \\ \omega & \text{otherwise} \end{cases}$

- OverApprox(N,k)=(L_k,m_0,$\Longrightarrow_{over}$) where:
  - $(m_1,m_2) \in \Longrightarrow_{over}$ iff $m_2 \in$ Post$^{\#k}(m_1)$
- **Lemma.** $\downarrow$Reach(N)$\subseteq\downarrow$Reach(OverApprox(N,k)).

An example





$\langle 0,1,1\rangle$
$\langle \omega,1,1\rangle$
$\langle 1,\omega,0\rangle$
$\langle 0,\omega,1\rangle$
$\langle \omega,0,\underline{\omega}\rangle$
$\langle \omega,1,\underline{\omega}\rangle$
$\langle \omega,\omega,\underline{\omega}\rangle$
$\langle \omega,\omega,0\rangle$
$\langle \omega,\omega,1\rangle$
t_1
t_2
t_3

Over(N,1)

EEC Algorithm

```
k:=0;
Repeat:
“Expand”: Compute $D_{Under}$:=UnderApprox(N,k)
“Enlarge”: Compute $D_{Over}$:=OverApprox(N,k)
“Check” : if $D_{Under} \cap U \neq \varnothing$ return “positive”;
            else if $D_{Over} \cap U = \varnothing$ return “negative”
                        else k:=k+1;
```

EEC Algorithm

```
k:=0;

Repeat:

“Expand”: Compute $D_{Under}$:=UnderApprox(N,k)

“Enlarge”: Compute $D_{Over}$:=OverApprox(N,k)

“Check” : if $D_{Under} \cap U \neq \varnothing$ return “positive”;
              else if $D_{Over} \cap U = \varnothing$ return “negative”
                                     else k:=k+1;
```

Clearly this algorithm is sound as it uses:
-under-approximations to detect positive instances.
-over-approximations to detect negative instances.

EEC Algorithm

```
k:=0;

Repeat:

"Expand": Compute $D_{Under}$:=UnderA

"Enlarge": C

"Ch                              return "positive";
          else if $D_{Over} \cap U=\emptyset$ return "negative"
                                        else k:=k+1;
```

But does it always **terminate** ?

Clearly this algorithm is sound as it uses:
-under-approximations to detect positive instances.
-over-approximations to detect negative instances.

Termination of EEC

- Yes it does always **terminate** !
- **Lemma(Positive instances)**. Let $m_0m_1...m_n$ be an execution that reaches U. Let k be the maximal number of tokens in a place of a marking in this execution. Then UnderApprox(N,k)$\cap$U$\neq\varnothing$.
- **Lemma(Negative instances)**. Let k=max{ m(p)$\neq\omega$ | m$\in$DGen($\downarrow$Reach(N))}. $\downarrow$Post$^{\#k}$($\downarrow$Reach(N))=$\downarrow$Post($\downarrow$Reach(N)), and so $\downarrow$OverApprox(N,k)=$\downarrow$Reach(N).

Beyond this introduction Bibliography

Some interesting papers

- General papers
  - Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, Yih-Kuen Tsay: **General Decidability Theorems for Infinite-State Systems**. LICS 1996: 313-321
  - Alain Finkel, Ph. Schnoebelen: **Well-structured transition systems everywhere!** Theor. Comput. Sci. 256(1-2): 63-92 (2001)
  - Gilles Geeraerts, Jean-François Raskin, Laurent Van Begin: **Expand, Enlarge and Check: New algorithms for the coverability problem of WSTS**. J. Comput. Syst. Sci. 72(1): 180-203 (2006)

Some interesting papers

- More applications
  - Parosh Aziz Abdulla, Aurore Annichini, Ahmed Bouajjani: **Symbolic Verification of Lossy Channel Systems: Application to the Bounded Retransmission Protocol**. TACAS 1999: 208-222
  - Parosh Aziz Abdulla, Pritha Mahata, Richard Mayr: **Dense-Timed Petri Nets: Checking Zenoness, Token liveness and Boundedness**. Logical Methods in Computer Science 3(1): (2007)
  - Joël Ouaknine, James Worrell: **On the Language Inclusion Problem for Timed Automata: Closing a Decidability Gap**. LICS 2004: 54-63
  - Thomas Wies, Damien Zufferey, Thomas A. Henzinger: **Forward Analysis of Depth-Bounded Processes**. FOSSACS 2010: 94-10

Some interesting papers

- Relation with abstractions/Abstract interpretation/ Domain theory:
    - Pierre Ganty, Jean-François Raskin, Laurent Van Begin: **A Complete Abstract Interpretation Framework for Coverability Properties of WSTS**. VMCAI 2006: 49-64.
    - Rayna Dimitrova, Andreas Podelski: **Is Lazy Abstraction a Decision Procedure for Broadcast Protocols?** VMCAI 2008: 98-111
    - Alain Finkel, Jean Goubault-Larrecq: **Forward Analysis for WSTS, Part I: Completions**. STACS 2009: 433-444
    - Alain Finkel, Jean Goubault-Larrecq: **Forward Analysis for WSTS, Part II: Complete WSTS.** ICALP (2) 2009: 188-199

Some interesting papers

- PhD Thesis:
  - Gilles Geeraerts. **Coverability and Expressiveness Properties of WSTS**. PhD Thesis. ULB. 2007.
  - Laurent Van Begin. **Efficient Verification of Counting Abstraction for Parametric Systems**. PhD Thesis. ULB. 2003.
  - Pritha Mahata. **Model Checking Parameterized Timed Systems**. PhD Thesis, 2005.

Conclusion

Conclusion

- Well-structured transition systems are a general class of infinite state systems with decidable verification problems.
- They are useful to model:
  - parametric systems,
  - lossy channel systems,
  - broadcast protocols,
  - timed Petri nets,
  - complements of one-clock timed languages, etc.
- We have reviewed three algorithmic tools for their analysis.

Questions