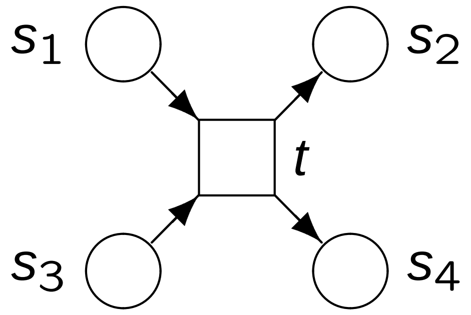
# Petri Nets

## Petri nets

Petri nets are a basic model of parallel and distributed systems, designed by Carl Adam Petri in 1962 in his PhD Thesis: “Kommunikation mit Automaten”. The basic idea is to describe state changes in a system with transitions.

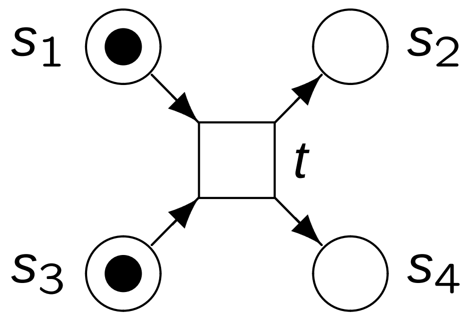


Petri nets contain places ○ (Stelle) and transitions □ (Transition) that may be connected by directed arcs.

Transitions symbolise actions; places symbolise states or conditions that need to be met before an action can be carried out.

## Behaviour of Petri nets

Places may contain tokens that may move to other places by executing (“firing”) actions.

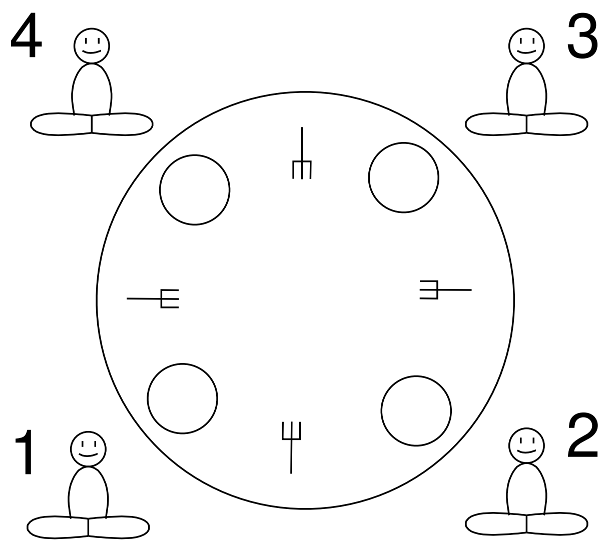


In the example, transition $t$ may “fire” if there are tokens on places $s_1$ and $s_3$. Firing $t$ will remove those tokens and place new tokens on $s_2$ and $s_4$.

## Example: Dining philosophers

There are philosophers sitting around a round table.

There are forks on the table, one between each pair of philosophers.



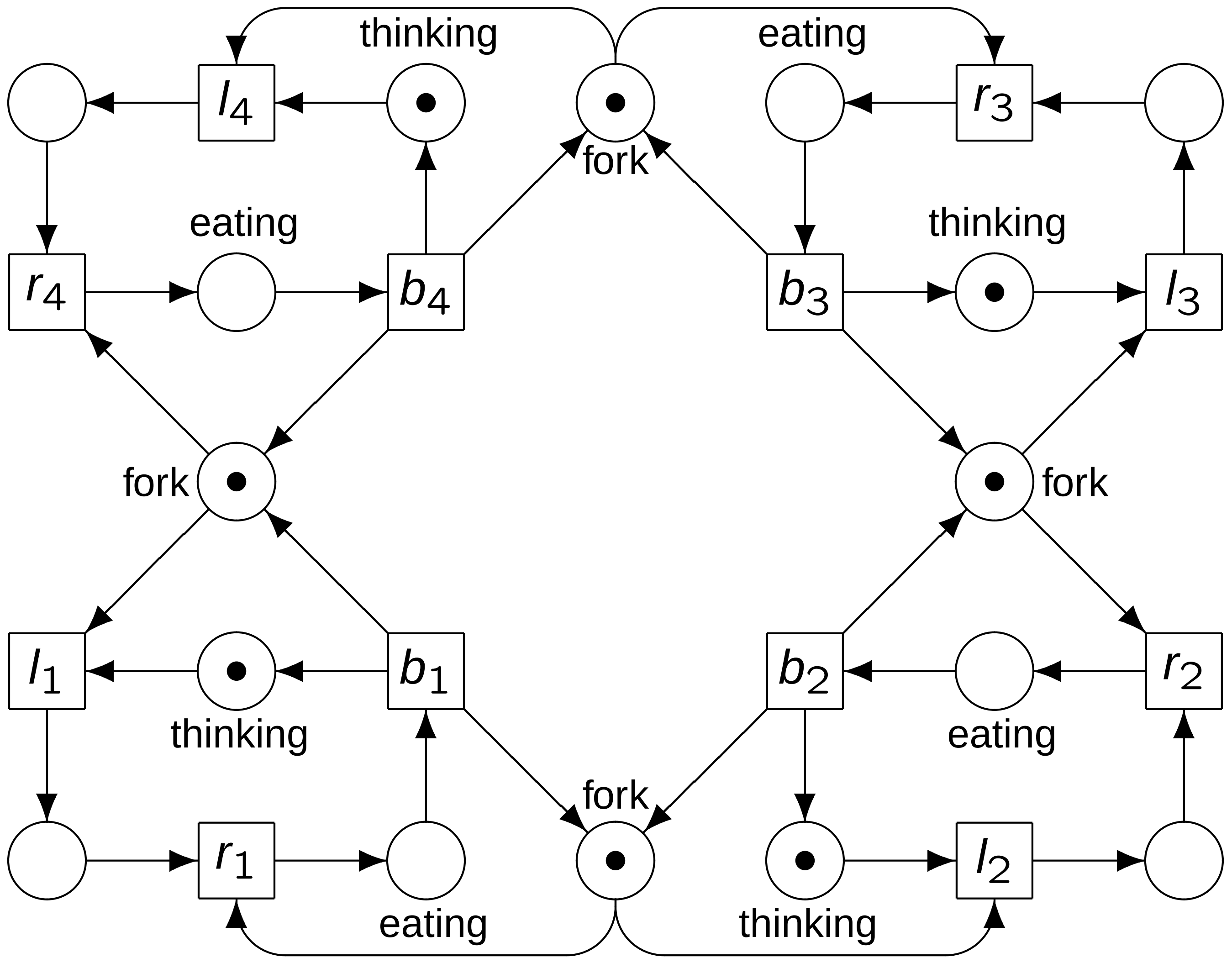
The philosophers want to eat spaghetti from a large bowl in the center of the table.

Unfortunately the spaghetti is of a particularly slippery type, and a philosopher needs both forks in order to eat it.

The philosophers have agreed on the following protocol to obtain the forks: Initially philosophers think about philosophy, when they get hungry they do the following: (1) take the left fork, (2) take the right fork and start eating, (3) return both forks simultaneously, and repeat from the beginning

How can we model the behaviour of the philosophers?

# Dining philosophers: Petri net



## Dining philosophers: Questions

Can two neighbouring philosophers eat at the same time?

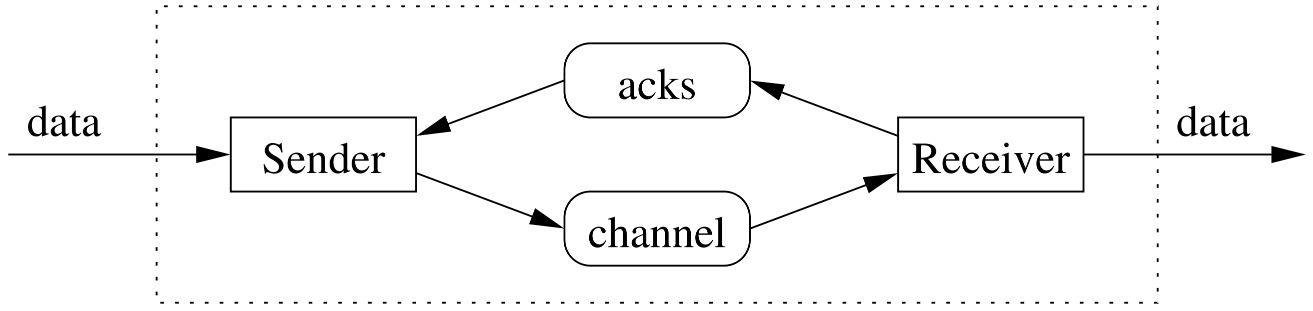
Can the philosophers starve to death?

Can an individual philosopher eventually eat, assuming he wants to?

## Example: Reliable connection on a lossy channel

Many data communication systems are based on unreliable connections that may distort, lose or duplicate messages. Distorted messages can be discarded by using checksums, and lost and duplicated messages can be detected by numbering the messages.

The basic solution assigns sequence numbers to the messages of transit in such a way that the recipient can detect a lost message and ask the sender to repeat previous messages. Whenever the recipient has obtained a contiguously numbered sequence of messages, it can relay them to the consumer.



## Alternating bit protocol

In the alternating bit protocol, there are two sequence numbers for messages: 0 and 1 (the alternating bit). Both the sender and the recipient have their own copy of the alternating bit. The sender holds it in the variable $s$, the recipient in $r$.

- Sender: send a message tagged with $s$.
  - if no acknowledgement tagged with $r$ arrives in due course, repeat the message;
  - if the recipient acknowledges with $s$, toggle $s$ and send next message.
- Recipient: receive a message tagged with $b$, i.e. the value of $s$ at sending time; acknowledge receipt of $b$.
  - if $b$ agrees with $r$, relay the message to consumer and toggle $r$;
  - otherwise discard the message.

## Alternating Bit protocol: Questions

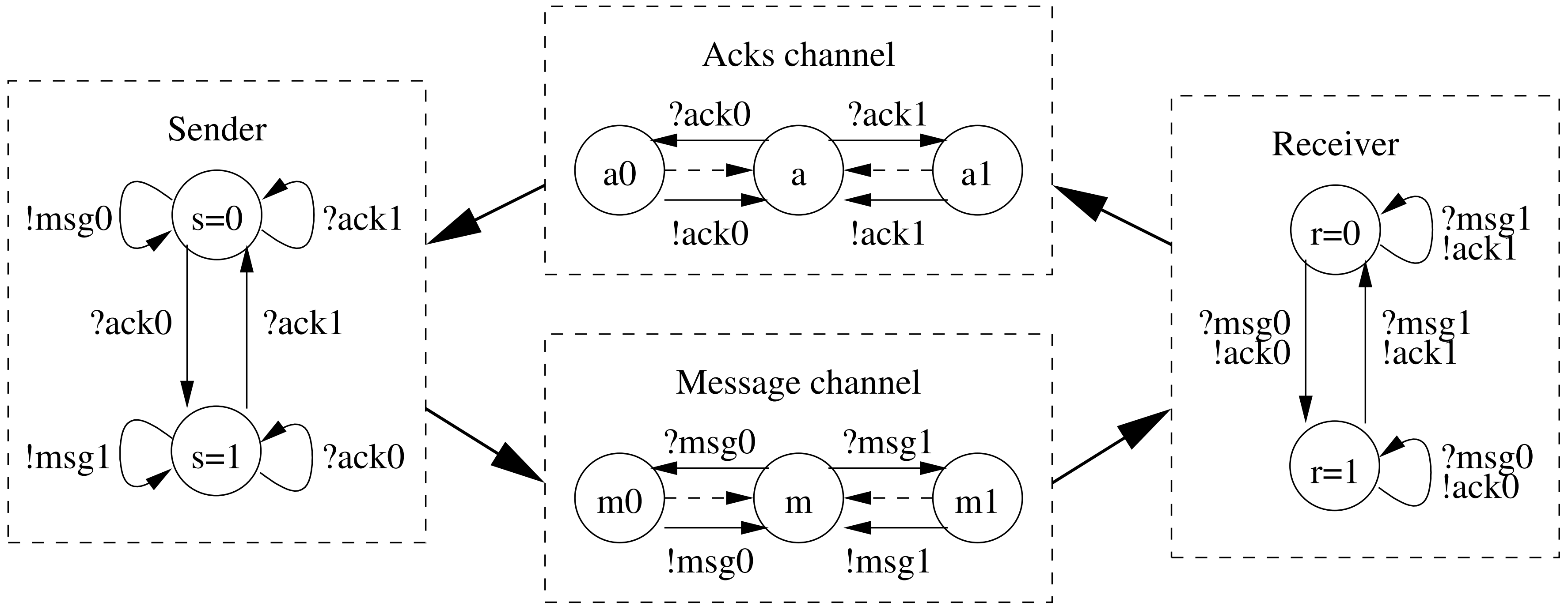
Is the protocol correct?

Is every message eventually delivered, assuming that the channels are not permanently faulty?

Are messages received in the right order?

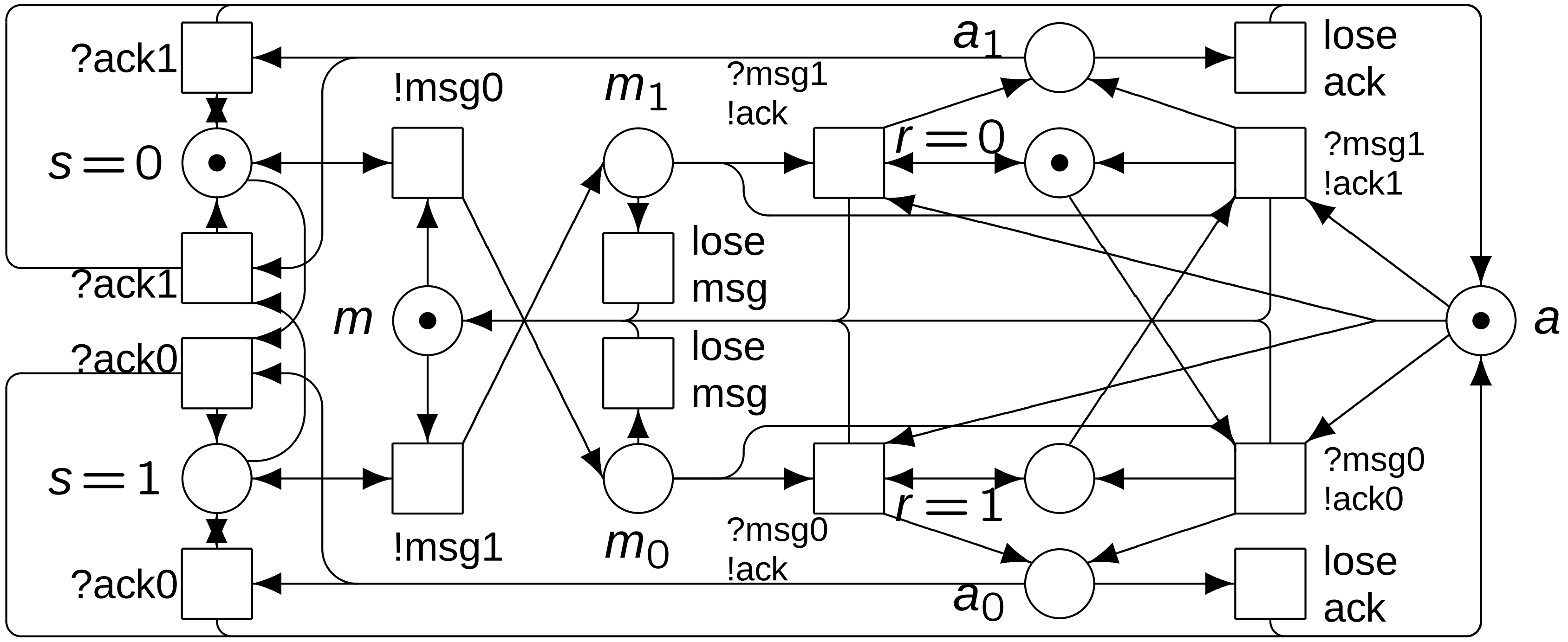
Can two messages with the same alternating bit value be confused?

## Alternating bit protocol: Communicating FA



The figure depicts the operation of the sender, the recipient and the channels as communicating finite state automata. Transmissions and receptions of messages are marked with exclamation marks and question marks. A transmission and a reception with the same name in neighbouring components are carried out simultaneously. The channels have capacity for at most one message at a time.

## Alternating bit protocol: Petri net



## Alternating bit protocol: state space

Even though the data transmitted by the protocol may be the main concern of end users, the data payload of the messages is irrelevant for observing the operation of the protocol. The less memory a model contains, the easier it can be verified, because a system with $b$ bits of memory can assume at most $2^b$ states.

In the Petri net model, the data has been abstracted away, as well as the value of the timeout (the timeout is assumed to be able to expire at any moment). The timeout could be handled by modelling a clock, but it would make verification harder (and the precise value of the timeout is not important here).

The state of a distributed system consists of the states of its component systems, for instance $\{s=0, s=1\} \times \{r=0, r=1\} \times \{m, m0, m1\} \times \{a, a0, a1\}$. In the beginning, each component system is in its initial state, which corresponds to the initial state to the whole system, e.g. $\langle s=0, r=0, m, a\rangle$.

## Constructing the state space (idea)

The state of a Petri system is formed by the distribution of tokens in the places.

The state changes when enabled transitions are fired.

A transition is enabled if each of its pre-places contains a token.

When an enabled transition fires, a token is removed from each of its pre-places and a token is inserted to each of its post-places.

The state space can be represented by a graph.

Nodes of the graph correspond to distributions of tokens.

Edges of the graph correspond to firing of transitions.

By constructing the state space, it is “fairly” easy to ensure that the alternating bit protocol works. There are at most $2 \cdot 2 \cdot 3 \cdot 3 = 36$ states.

It turns out that from $\langle s=0, r=0, m, a \rangle$, there are 18 reachable nodes and 40 edges.

## Place/Transition Nets

## Place/Transition Nets

Let us study Petri nets and their firing rule in more detail:

- A place may contain several tokens, which may be interpreted as resources.
- There may be several input and output arcs between a place and a transition. The number of these arcs is represented as the weight of a single arc.
- A transition is enabled if its each input place contains at least as many tokens as the corresponding input arc weight indicates.
- When an enabled transition is fired, its input arc weights are subtracted from the input place markings and its output arc weights are added to the output place markings.

## Place/Transition Net

A Place/Transition Net (P/T net) is a tuple $N = \langle P, T, F, W, M_0 \rangle$, where

- $P$ is a finite set of places,
- $T$ is a finite set of transitions,
- the places $P$ and transitions $T$ are disjoint ($P \cap T = \emptyset$),
- $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation,
- $W : F \to (\mathbb{N} \setminus \{0\})$ is the arc weight mapping, and
- $M_0 : P \to \mathbb{N}$ is the initial marking representing the initial distribution of tokens.

## P/T nets: Remarks

If $\langle p, t\rangle \in F$ for a transition $t$ and a place $p$, then $p$ is an input place of $t$,

If $\langle t, p\rangle \in F$ for a transition $t$ and a place $p$, then $p$ is an output place of $t$,

Let $a \in P \cup T$. The set ${}^\bullet a = \{a' \mid \langle a', a\rangle \in F\}$ is called the pre-set of $a$, and the set $a^\bullet = \{a' \mid \langle a, a'\rangle \in F\}$ is its post-set.

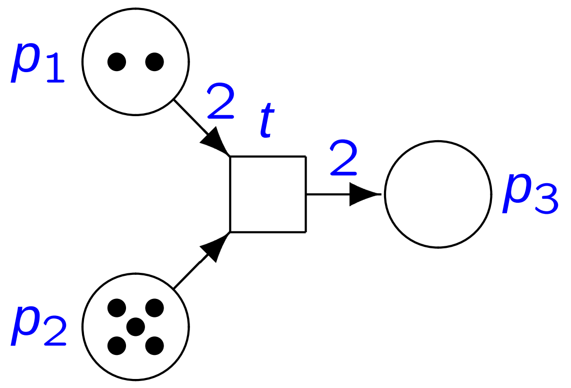
When drawing a Petri net, we usually omit arc weights of $1$. Also, we may either denote tokens on a place either by black circles, or by a number.

## Alternative definitions

Sometimes the notation $S$ (for Stellen) is used instead of $P$ (for places) in the definition of Place/Transition nets.

Some definitions also use the notion of a place capacity (the maximum number of tokens allowed in a place, possibly unbounded). Place capacities can be simulated by adding some additional places to the net (we will see how later), and thus for simplicity we will not define them in this course.

## Place/Transition Net: Example



The place/transition net $\langle P, T, F, W, M_0 \rangle$ above is defined as follows:

- $P = \{p_1, p_2, p_3\}$,
- $T = \{t\}$,
- $F = \{\langle p_1, t\rangle, \langle p_2, t\rangle, \langle t, p_3\rangle\}$,
- $W = \{\langle p_1, t\rangle \mapsto 2, \langle p_2, t\rangle \mapsto 1, \langle t, p_3\rangle \mapsto 2\}$,
- $M_0 = \{p_1 \mapsto 2, p_2 \mapsto 5, p_3 \mapsto 0\}$.

## Notation for markings

Often we will fix an order on the places (e.g., matching the place numbering), and write, e.g., $M_0 = \langle 2, 5, 0 \rangle$ instead.

When no place contains more than one token, markings are in fact sets, in which case we often use set notation and write instead $M_0 = \{p_5, p_7, p_8\}$.

Alternatively, we could denote a marking as a multiset, e.g. $M_0 = \{p_1, p_1, p_2, p_2, p_2, p_2, p_2\}$.

The notation $M(p)$ denotes the number of tokens in place $p$ in marking $M$.

## The firing rule revisited

Let $\langle P, T, F, W, M_0 \rangle$ be a Place/Transition net and $M : P \to \mathbb{N}$ one of its markings.

Firing condition:

Transition $t \in T$ is $M$-enabled, written $M \xrightarrow{t}$, iff $\forall p \in {}^{\bullet}t : M(p) \geq W(p, t)$.

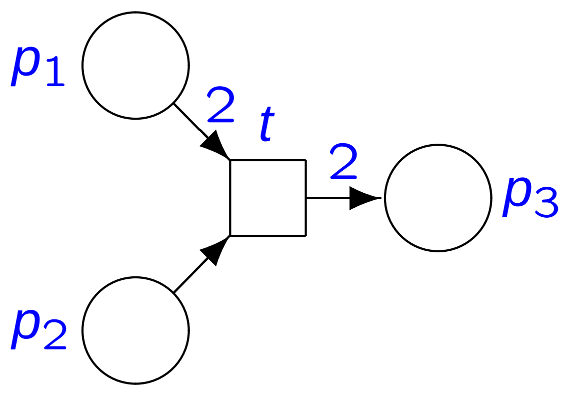
Firing rule:

An $M$-enabled transition $t$ may fire, producing the successor marking $M'$, written $M \xrightarrow{t} M'$, where

$$\forall p \in P : M'(p) = M(p) - \bar{W}(p, t) + \bar{W}(t, p)$$

where $\bar{W}$ is defined as $\bar{W}(x, y) := W(x, y)$ for $\langle x, y \rangle \in F$ and $\bar{W}(x, y) := 0$ otherwise.

# The firing rule of Place/Transition Nets: Example



| Marking $M$ | $M \xrightarrow{t}$ | $M'$ |
|---|---|---|
| $\{p_1 \mapsto 2, p_2 \mapsto 5, p_3 \mapsto 0\}$ | enabled | $\{p_1 \mapsto 0, p_2 \mapsto 4, p_3 \mapsto 2\}$ |
| $\{p_1 \mapsto 0, p_2 \mapsto 4, p_3 \mapsto 2\}$ | disabled | |
| $\{p_1 \mapsto 1, p_2 \mapsto 5, p_3 \mapsto 0\}$ | disabled | |

Note: If $M \xrightarrow{t} M'$, then we call $M'$ the successor marking of $M$.

## Reachable markings

Let $M$ be a marking of a Place/Transition net $N = \langle P, T, F, W, M_0 \rangle$.

The set of markings reachable from $M$ (the reachability set of $M$, written $reach(M)$) is the smallest set of markings, such that:

1. $M \in reach(M)$, and
2. if $M' \xrightarrow{t} M''$ for some $t \in T$, $M' \in reach(M)$, then $M'' \in reach(M)$.

Let $\mathcal{M}$ be a set of markings. The previous notation is extended to sets of markings in the obvious way:

$$reach(\mathcal{M}) = \cup_{M \in \mathcal{M}} reach(M)$$

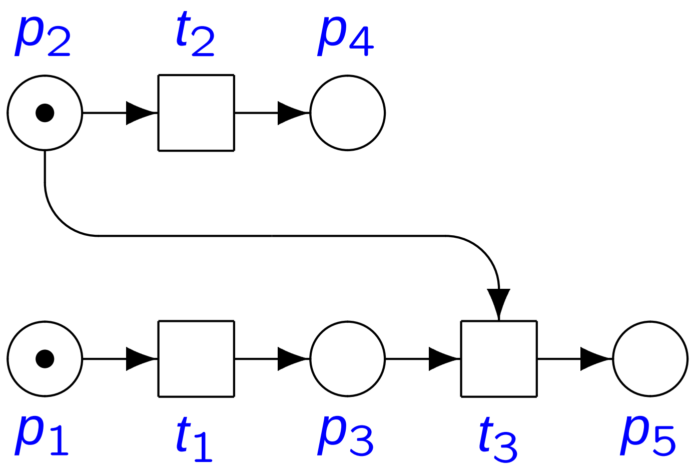
The set of reachable markings $reach(N)$ of a net $N = \langle P, T, F, W, M_0 \rangle$ is defined to be $reach(M_0)$.

# Reachability Graph

The reachability graph of a place/transition net $N = \langle P, T, F, W, M_0 \rangle$ is a rooted, directed graph $G = \langle V, E, v_0 \rangle$, where

- $V = \mathit{reach}(N)$ is the set of vertices, i.e. each reachable marking is a vertex;
- $v_0 = M_0$, i.e. the initial marking is the root node;
- $E = \left\{ \langle M, t, M' \rangle \;\middle|\; M \in V \text{ and } M \xrightarrow{t} M' \right\}$ is the set of edges, i.e. there is an edge from each marking (resp. vertex) $M$ to each of its successor markings, and the edge is labelled with the firing transition.

# Reachability Graph: Example



$$\begin{tikzcd}
 & \langle 0,0,1,1,0\rangle & & \langle 0,0,0,0,1\rangle \\
\langle 1,0,0,1,0\rangle \arrow[ur, "t_1"] & & \langle 0,1,1,0,0\rangle \arrow[ul, "t_2"'] \arrow[ur, "t_3"] & \\
 & > \langle 1,1,0,0,0\rangle \arrow[ul, "t_2"] \arrow[ur, "t_1"'] & &
\end{tikzcd}$$

- The weight of each arc is 1.
- The graph shows that $t_3$ cannot be fired if $t_2$ is fired before $t_1$. Thus, intuitively speaking, $t_1$ and $t_2$ are not independent, even though their presets and postsets are mutually disjunct.

## Computing the reachability graph

```
REACHABILITY-GRAPH($\langle P, T, F, W, M_0 \rangle$)
1   $\langle V, E, v_0 \rangle := \langle \{M_0\}, \emptyset, M_0 \rangle$;
2   $\mathit{Work}$ : set $:= \{M_0\}$;
3   while $\mathit{Work} \neq \emptyset$
4   do select $M$ from $\mathit{Work}$;
5       $\mathit{Work} := \mathit{Work} \setminus \{M\}$;
6       for $t \in \mathsf{enabled}(M)$
7       do $M' := \mathsf{fire}(M, t)$;
8           if $M' \notin V$
9              then $V := V \cup \{M'\}$
10                  $\mathit{Work} := \mathit{Work} \cup \{M'\}$;
11          $E := E \cup \{\langle M, t, M' \rangle\}$;
12  return $\langle V, E, v_0 \rangle$;
```

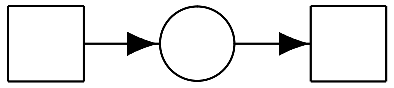
The algorithm makes use of two functions:

- $\mathsf{enabled}(M) := \{t \mid M \xrightarrow{t}\}$
- $\mathsf{fire}(M, t) := M'$
  if $M \xrightarrow{t} M'$

The set $\mathit{Work}$ may be implemented as a stack, in which case the graph will be constructed in a depth-first manner, or as a queue for breadth-first. Breadth first search will find the shortest transition path from the initial marking to a given (erroneous) marking. Some applications require depth first search.

## The size of the reachability graph

In general, the graph may be infinite, i.e. if there is no bound on the number tokens on some place. Example:



Definition: If each place of a place/transition net can contain at most $k$ tokens in each reachable marking, the net is said to be $k$-safe.

A $k$-safe net has at most $(k+1)^{|P|}$ markings; for $1$-safe nets, the limit is $2^{|P|}$.

## Use of reachability graphs

In practice, all analysis tools and methods for Petri nets compute the reachability graph in some way or other. The reachability graph can be effectively computed if the net is $k$-safe for some $k$.

If the net is not $k$-safe for any $k$, we may compute the coverability graph (see next lecture).

## Analysis with Place/Transition Nets

## Recap from last Session

We discussed the following topics:

Modeling: Petri nets (and others)

Specifying: reachability (and others, mostly informally)

Verifying: construction of reachability graph

## Programme

Example: modeling, specifying, and reachability analysis

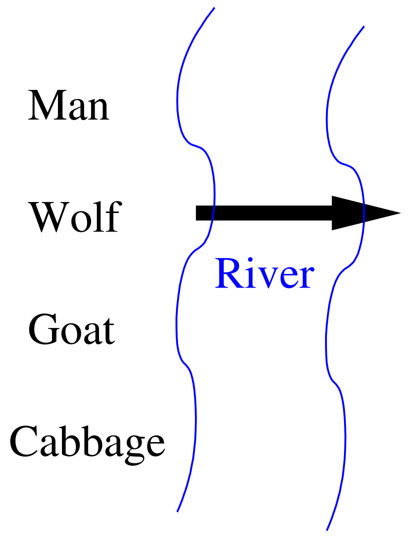
Coverability graphs

Analysis with reachability and coverability graphs

## Example: A logical puzzle

A man is travelling with a wolf, a goat, and a cabbage. The four come to a river that they must cross. There is a boat available for crossing the river, but it can carry only the man and at most one other object. The wolf may eat the goat when the man is not around, and the goat may eat the cabbage when unattended.

Can the man bring everyone across the river without endangering the goat or the cabbage? And if so, how?



## Example: Modeling

We are going to model the situation with a Petri net.

The puzzle mentions the following objects:

> Man, wolf, goat, cabbage, boat. Both can be on either side of the river.

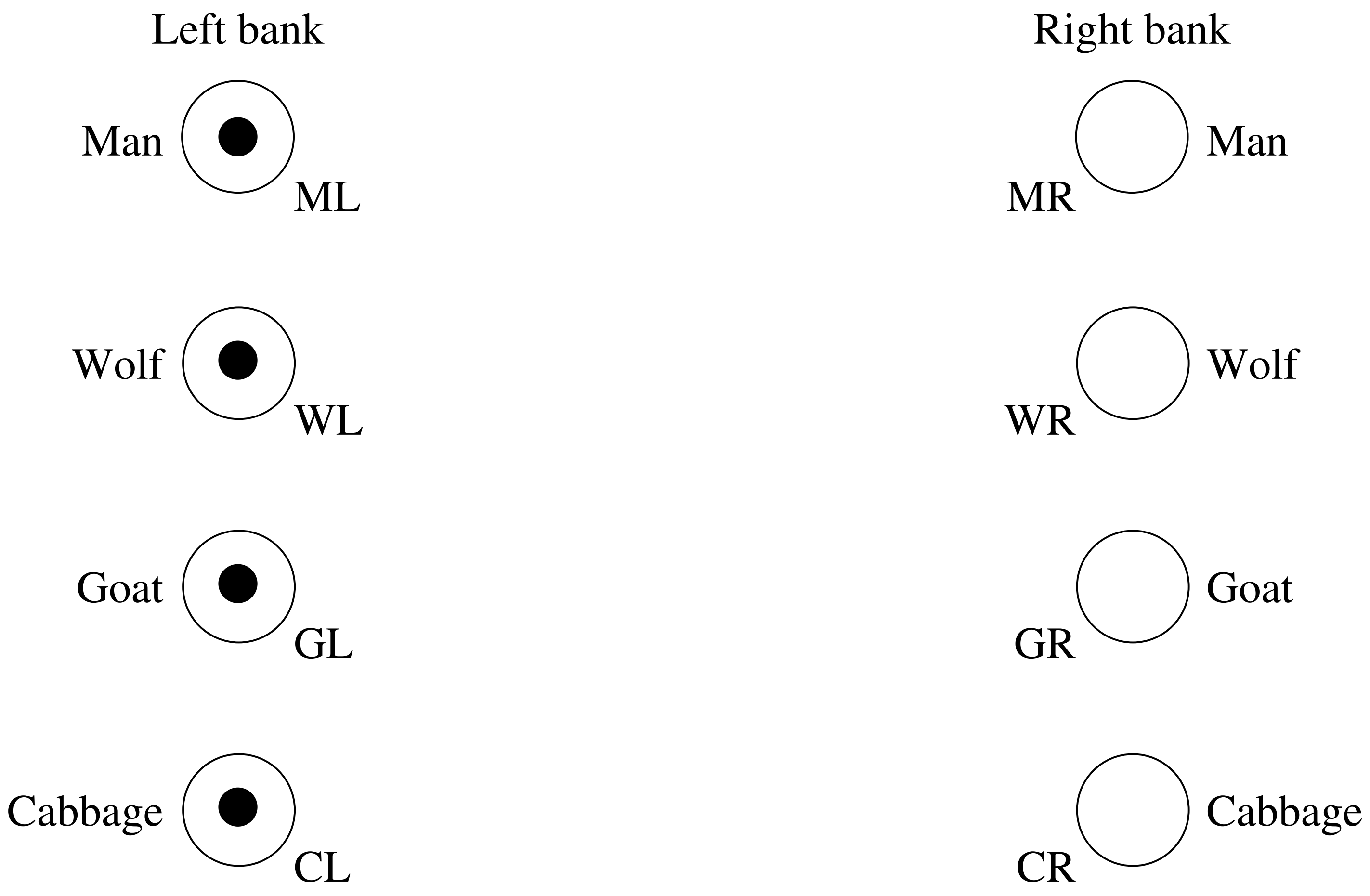
The puzzle mentions the following actions:

> Crossing the river, wolf eats goat, goat eats cabbage.

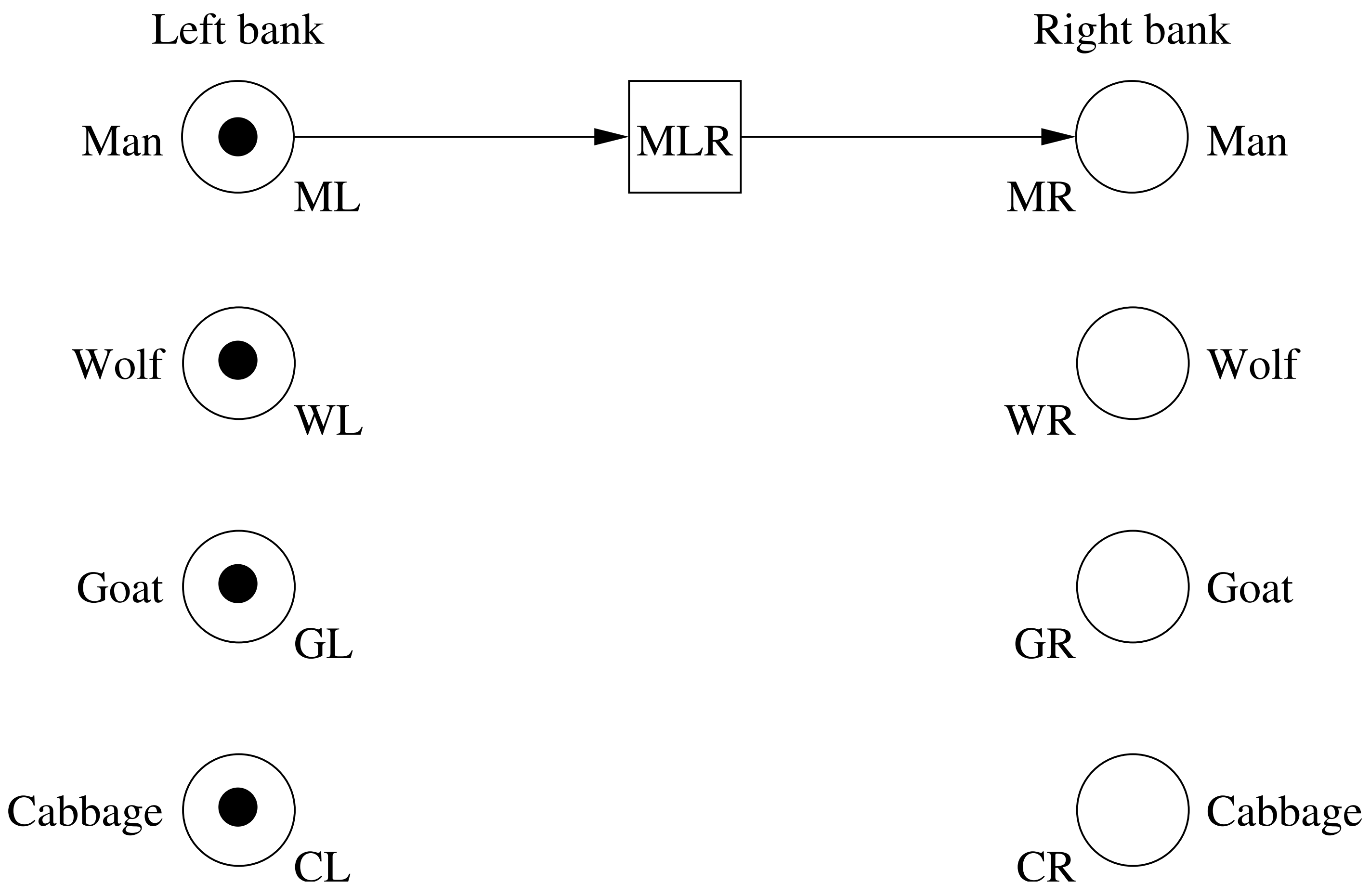
Objects and their states are modeled by places.
Actions are modeled by transitions.

Actually, we can omit the boat, because it is always going to be on the same side as the man.

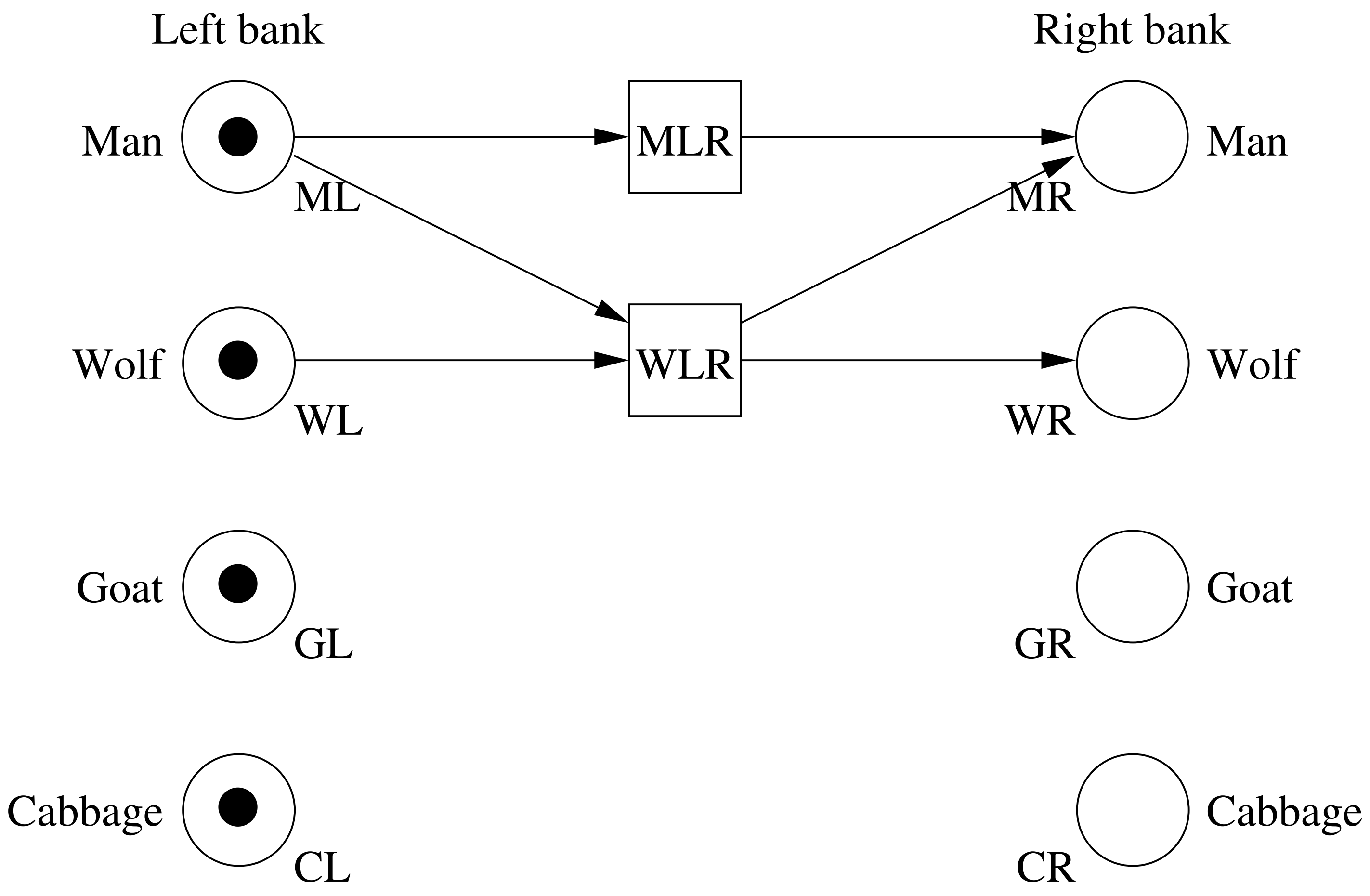
## Example: Places



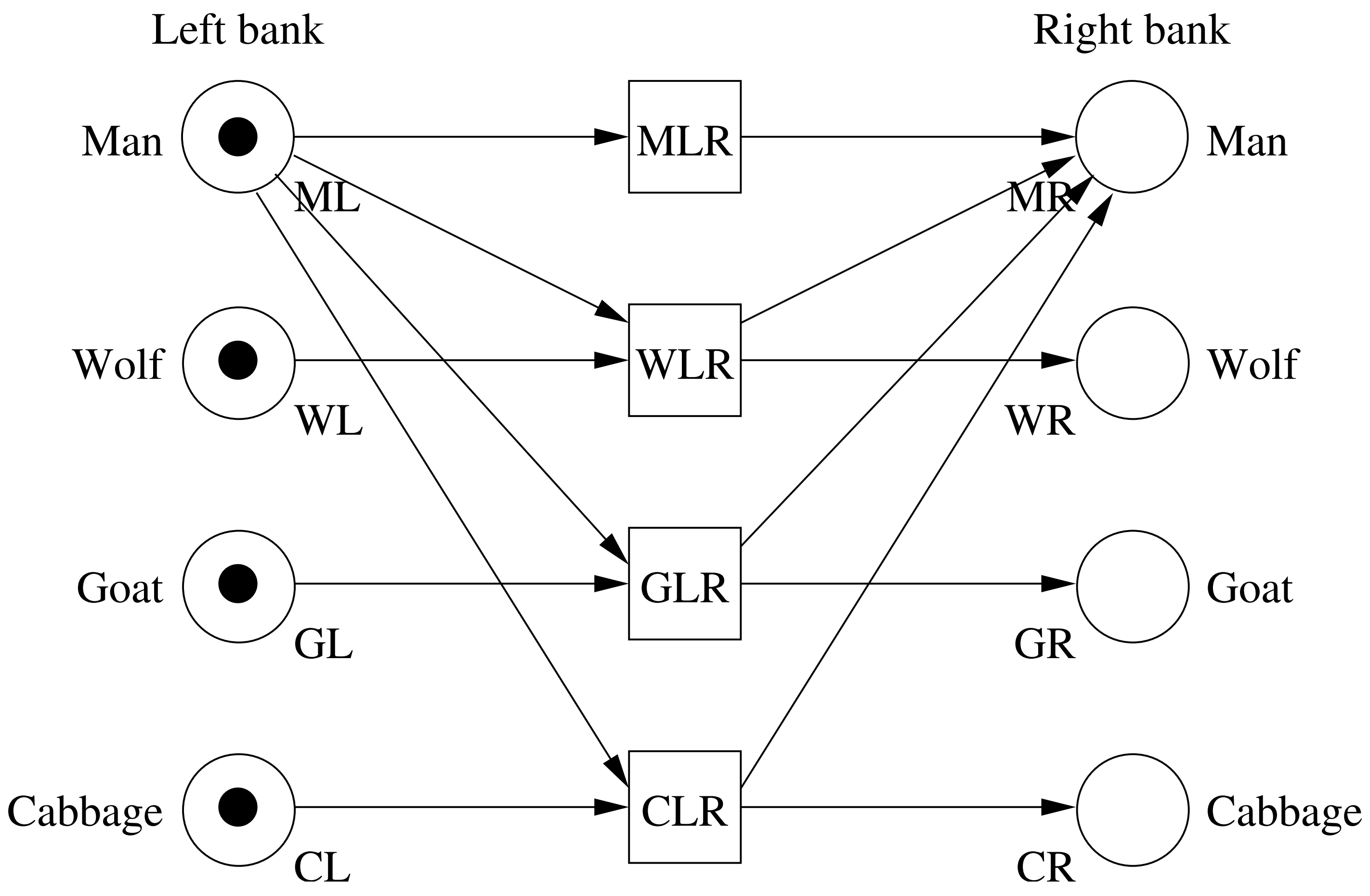
## Crossing the river (left to right)



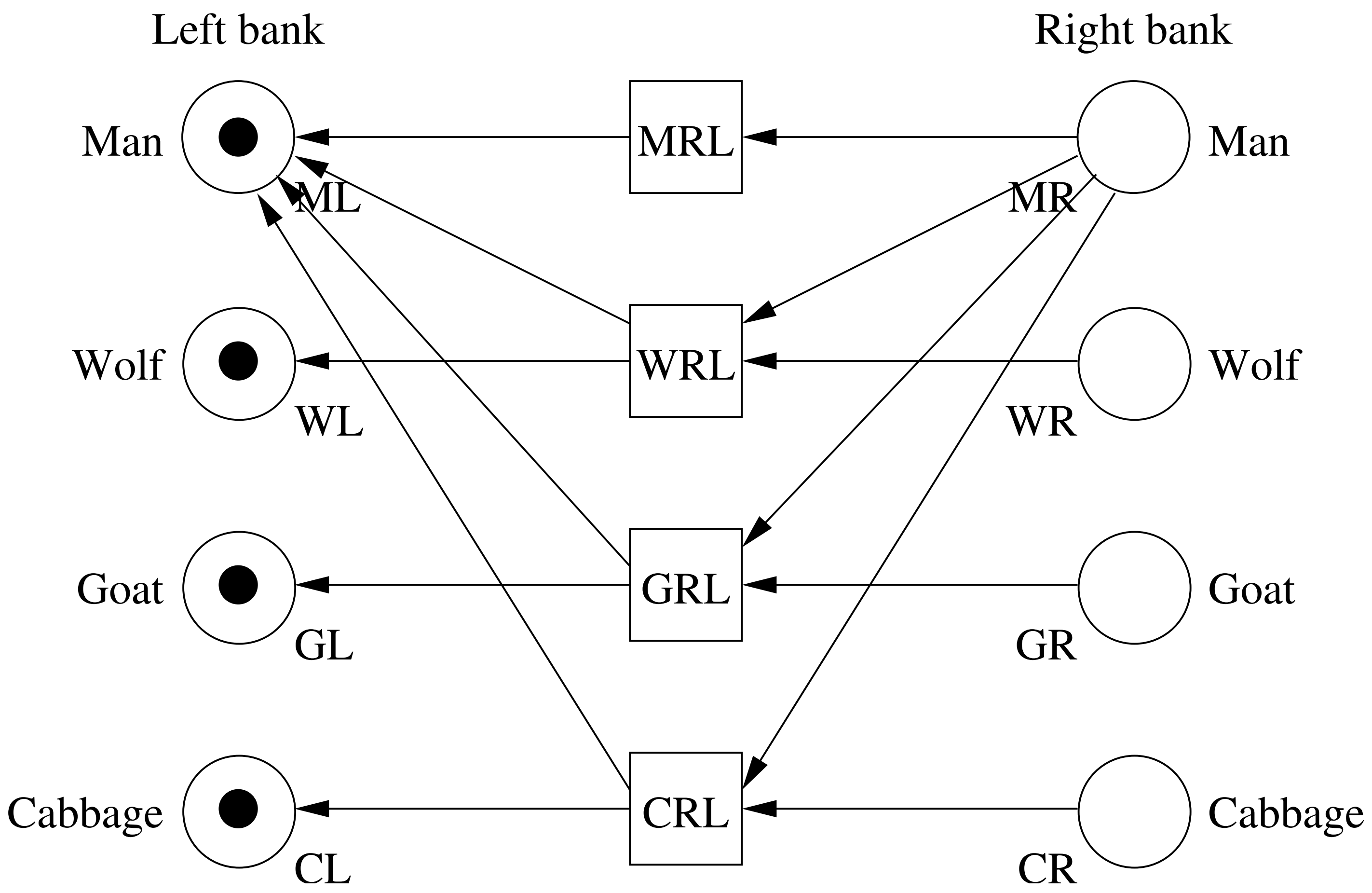
## Crossing the river (left to right)



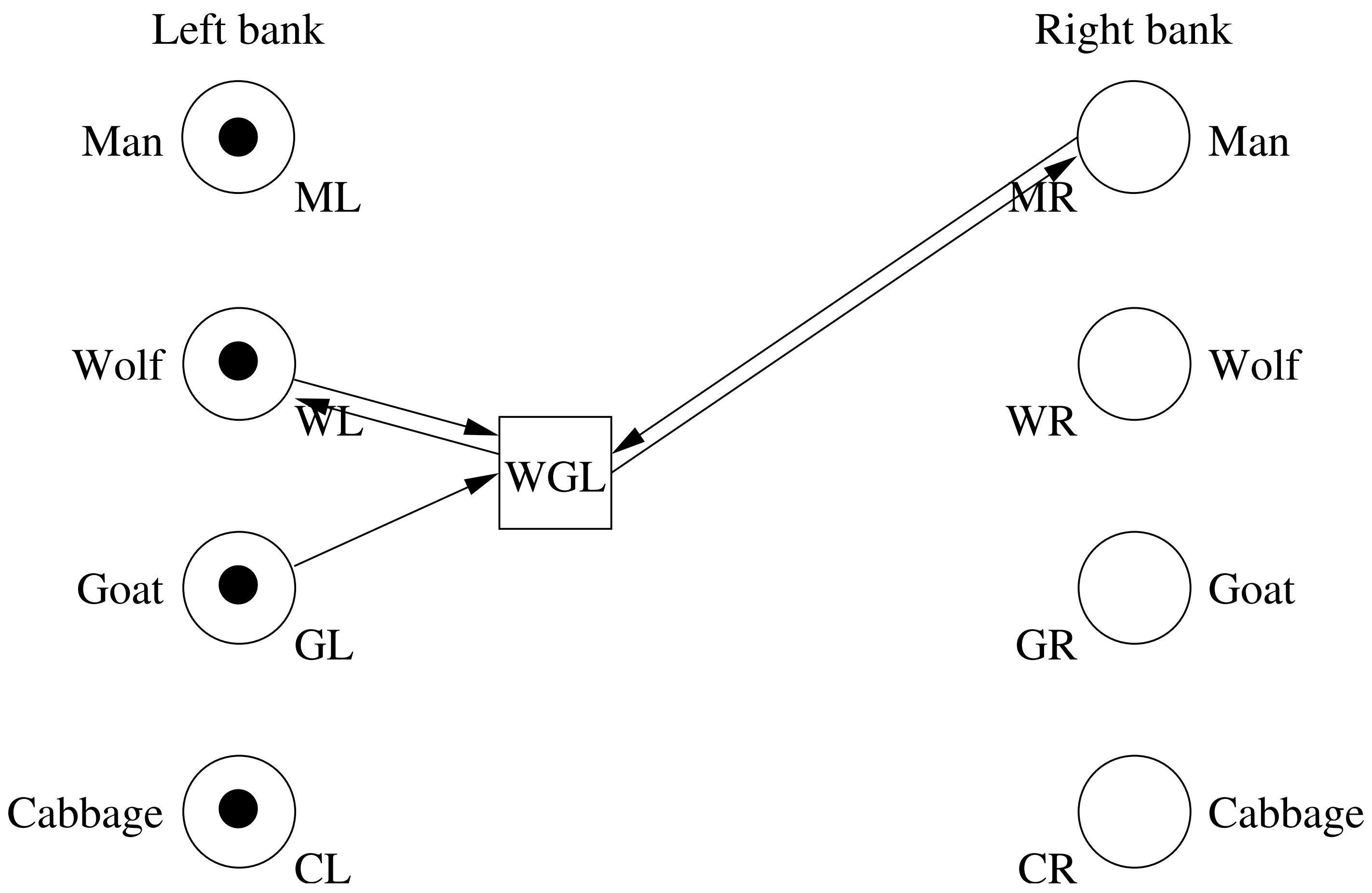
## Crossing the river (left to right)



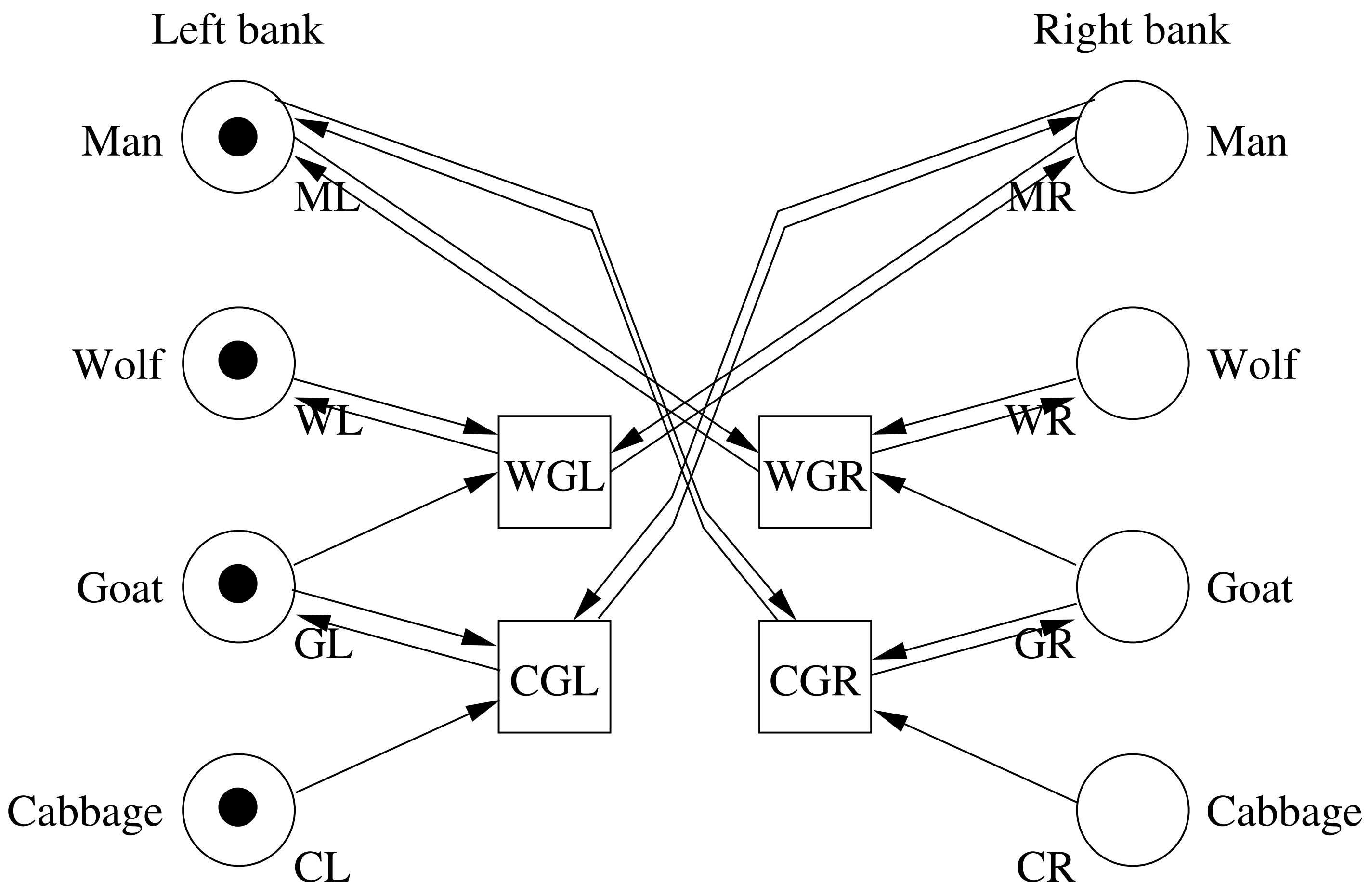
# Crossing the river (right to left)



## Wolf eats goat



# Wolf eats goat, goat eats cabbage



## Example: Specification

To solve the problem using the Petri net, we need to translate the questions *“Can the man bring everyone across the river without endangering the goat or the cabbage? And if so, how?”* into properties of the Petri net.

## Example: Specification

To solve the problem using the Petri net, we need to translate the questions *“Can the man bring everyone across the river without endangering the goat or the cabbage? And if so, how?”* into properties of the Petri net.

“Can the man bring everyone across the river?”

$\Rightarrow$ Is the marking $\{MR, WR, GR, CR\}$ reachable from $\{ML, WL, GL, CL\}$?

## Example: Specification

To solve the problem using the Petri net, we need to translate the questions *“Can the man bring everyone across the river without endangering the goat or the cabbage? And if so, how?”* into properties of the Petri net.

“Can the man bring everyone across the river?”

$\Rightarrow$ Is the marking $\{MR, WR, GR, CR\}$ reachable from $\{ML, WL, GL, CL\}$?

“$\ldots$ without endangering the goat or the cabbage?”

$\Rightarrow$ We need to avoid states in which one of the eating transitions is enabled.

## Example: Specification

To solve the problem using the Petri net, we need to translate the questions *“Can the man bring everyone across the river without endangering the goat or the cabbage? And if so, how?”* into properties of the Petri net.

“Can the man bring everyone across the river?”

$\Rightarrow$ Is the marking $\{MR, WR, GR, CR\}$ reachable from $\{ML, WL, GL, CL\}$?

“...without endangering the goat or the cabbage?”

$\Rightarrow$ We need to avoid states in which one of the eating transitions is enabled.

“How?”

$\Rightarrow$ Give a path that leads from one marking to the other. (Optionally: Find a shortest path.)

## Result

Constructing the reachability graph yields a graph with (at most) 36 nodes.

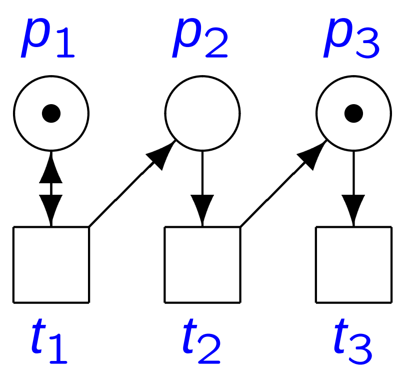
The marking $\{MR, WR, GR, CR\}$ *is reachable* without enabling an “eating” transition!

The transitions fired along a shortest path (there are two) are:

|  |  |
|---|---|
| $GLR$ | (man and goat cross the river), |
| $MRL$ | (man goes back alone), |
| $WLR$ | (man and wolf cross the river), |
| $GRL$ | (man and goat go back), |
| $CLR$ | (man and cabbage cross the river), |
| $MRL$ | (man goes back alone), |
| $GLR$ | (man and goat cross the river). |

## Coverability Graph Method

As we have mentioned before, the reachability graph of P/T-net can be infinite (in which case the algorithm for computing the reachability graph will not terminate). For example, consider the following net.



We will show a method to find out whether the reachability graph of a P/T-net is infinite or not. This can be done by using the coverability graph method.

## $\omega$-Markings

First we introduce a new symbol $\omega$ to represent “arbitrarily many” tokens.

We extend the arithmetic on natural numbers with $\omega$ as follows. For all $n \in \mathbb{N}$:
$n + \omega = \omega + n = \omega$,
$\omega + \omega = \omega$,
$\omega - n = \omega$,
$0 \cdot \omega = 0$, $\omega \cdot \omega = \omega$,
$n \geq 1 \Rightarrow n \cdot \omega = \omega \cdot n = \omega$,
$n \leq \omega$, and $\omega \leq \omega$.

Note: $\omega - \omega$ remains undefined, but we will not need it.

We will extend the notion of markings to $\omega$-markings. In an $\omega$-marking, each place $p$ will either have $n \in \mathbb{N}$ tokens, or $\omega$ tokens (infinitely many).

## Firing Rule and $\omega$-markings

The firing condition and firing rule (reproduced below) neatly extend to $\omega$-markings with the extended arithmetic rules:

> Firing condition:
>
> Transition $t \in T$ is $M$-enabled, written $M \xrightarrow{t}$, iff $\forall p \in {}^\bullet t : M(p) \geq W(p,t)$.
>
> Firing rule:
> An $M$-enabled transition $t$ may fire, producing the successor marking $M'$, where
>
> $$\forall p \in P : M'(p) = M(p) - \bar{W}(p,t) + \bar{W}(t,p).$$

Basically, if a transition has a place with $\omega$ tokens in its preset, that place is considered to have sufficiently many tokens for the transition to fire, regardless of the arc weight.

If a place contains an $\omega$-marking, then firing any transition connected with an arc to that place will not change its marking.

## Definition of Covering

An $\omega$-marking $M'$ covers an $\omega$-marking $M$, denoted $M \leq M'$, iff

$$\forall p \in P : M(p) \leq M'(p).$$

An $\omega$-marking $M'$ strictly covers an $\omega$-marking $M$, denoted $M < M'$, iff

$$M \leq M' \quad \text{and} \quad M' \neq M.$$

## Coverability and Transition Sequences (1/2)

Observation: Let $M$ and $M'$ be two markings such that $M \leq M'$.
Then for all transitions $t$, the following holds:

$$\text{If } M \xrightarrow{t} \text{ then } M' \xrightarrow{t}.$$

In other words, if $M'$ has at least as many tokens as $M$ has (on each place), then $M'$ enables at least the same transitions as $M$ does.

This observation can be extended to *sequences* of transitions:
Define $M \xrightarrow{t_1 t_2 \ldots t_n} M'$ to denote:

$$\exists M_1, M_2, \ldots, M_n : M \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \cdots \xrightarrow{t_n} M_n = M'.$$

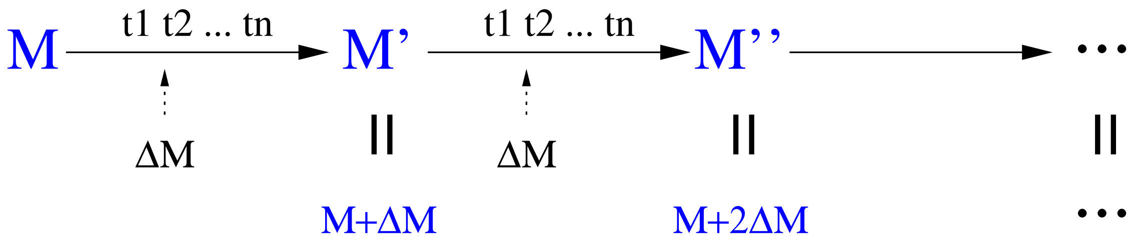
Now, if $M \xrightarrow{t_1 t_2 \ldots t_n}$ and $M \leq M'$, then $M' \xrightarrow{t_1 t_2 \ldots t_n}$.

## Coverability and Transition Sequences (2/2)

Assume that $M' \in \mathit{reach}(M)$ (with $M < M'$). Then clearly there is some sequence of transitions $t_1 t_2 \ldots t_n$ such that $M \xrightarrow{t_1 t_2 \ldots t_n} M'$. Thus, there is a marking $M''$ with $M' \xrightarrow{t_1 t_2 \ldots t_n} M''$.

Let $\Delta M := M' - M$ (place-wise difference). Because $M < M'$, the values of $\Delta M$ are non-negative and at least one value is non-zero.

Clearly, $M'' = M' + \Delta M = M + 2\Delta M$.



By firing the transition sequence $t_1 t_2 \dots t_n$ repeatedly we can “pump” an arbitrary number of tokens to all the places having a non-zero marking in $\triangle M$.

The basic idea for constructing the coverability graph is now to replace the marking $M'$ with a marking where all the places with non-zero tokens in $\triangle M$ are replaced by $\omega$.

## Coverability Graph Algorithm (1/2)

```
COVERABILITY-GRAPH($\langle P, T, F, W, M_0 \rangle$)
$\langle V, E, v_0 \rangle := \langle \{M_0\}, \emptyset, M_0 \rangle$;
$\mathit{Work}$ : set $:= \{M_0\}$;
while $\mathit{Work} \neq \emptyset$
do select $M$ from $\mathit{Work}$;
      $\mathit{Work} := \mathit{Work} \setminus \{M\}$;
      for $t \in \mathsf{enabled}(M)$
      do $M' := \mathsf{fire}(M, t)$;
            $M' := \mathsf{AddOmegas}(M, t, M', V, E)$;
            if $M' \notin V$
               then $V := V \cup \{M'\}$
                        $\mathit{Work} := \mathit{Work} \cup \{M'\}$;
            $E := E \cup \{\langle M, t, M' \rangle\}$;
return $\langle V, E, v_0 \rangle$;
```

The coverability graph algorithm is almost exactly the same as the reachability graph algorithm, with the addition of the call to subroutine $\mathsf{AddOmegas}(M, t, M', V, E)$, where all the details w.r.t. coverability graphs are contained. As for the implementation of $\mathit{Work}$, the same comments as for the reachability graph apply.

## Coverability Graph Algorithm (2/2)

The following notations are used in the AddOmegas subroutine:

- $M'' \rightarrow_E M$ iff $\langle M'', t, M\rangle \in E$ for some $t \in T$.
- $M'' \rightarrow_{E*} M$ iff
  $\exists n \geq 0 \colon \exists M_0, M_1, \ldots, M_n \colon M'' = M_0 \rightarrow_E M_1 \rightarrow_E M_2 \rightarrow_E \cdots \rightarrow_E M_n = M$.

```
AddOmegas($M, t, M', V, E$)
1   for $M'' \in V$
2   do if $M'' < M'$ and $M'' \rightarrow_{E*} M$
3           then $M' := M' + ((M' - M'') \cdot \omega)$;
4   return $M'$;
```

Line 3 causes all places whose marking in $M'$ is strictly larger than in the “parent” $M''$ to contain $\omega$, while markings of other places remain unchanged.

## Termination of the Coverability Graph Algorithm (1/2)

**Dickson's lemma**: Every infinite sequence $u_1 u_2 \ldots$ of $n$-tuples of natural numbers contains an infinite subsequence $u_{i_1} \leq u_{i_2} \leq u_{i_3} \leq \ldots$.

**Proof**: By induction on $n$.

**Base:** $n = 1$. Let $u_{i_1}$ be the smallest of $u_1 u_2 \ldots$, let $u_{i_2}$ be the smallest of $u_{i_1+1} u_{i_1+2} \ldots$ etc.

**Step:** $n > 1$. Consider the projections $v_1 v_2 \ldots$ and $w_1 w_2 \ldots$ of $u_1 u_2 \ldots$ onto the first $n - 1$ components and the last component, respectively. By induction hypothesis, there is an infinite subsequence $v_{j_1} \leq v_{j_2} \leq v_{j_3} \leq \ldots$.
Consider the infinite sequence $w_{j_1} \leq w_{j_2} \leq \ldots$. By induction hypothesis, this sequence has an infinite subsequence $w_{i_1} \leq w_{i_2} \leq \ldots$. So we have $u_{i_1} \leq u_{i_2} \leq u_{i_3} \leq \ldots$.

## Termination of the Coverability Graph Algorithm (2/2)

**Theorem:** The Coverability Graph Algorithm terminates.

**Proof**: Assume that the algorithm does not terminate. We derive a contradiction.

If the algorithm does not terminate, then the Coverability Graph is infinite. Since every node of the graph has at most $|T|$ successors, the graph contains an infinite path $\Pi = M_1 M_2 \ldots$.

If an $\omega$-marking $M_i$ of $\Pi$ satisfies $M_i(p) = \omega$ for some place $p$, then $M_{i+1}(p) = M_{i+2}(p) = \ldots = \omega$.
So $\Pi$ contains an $\omega$ marking $M_j$ such that all markings $M_{j+1}, M_{j+2}, \ldots$ have $\omega$'s at exactly the same places as $M_j$. Let $\Pi'$ be the suffix of $\Pi$ starting at $M_j$.

Consider the projection $\Pi'' = m_j m_{j+1} \ldots$ of $\Pi'$ onto the non-$\omega$ places. Let $n$ be the number of non-$\omega$ places. $\Pi''$ is an infinite sequence of distinct $n$-tuples of natural numbers.

By Dickson's lemma, this sequence contains markings $M_k, M_l$ such that $k < l$ and $M_k \leq M_l$. This is a contradiction, because, since $M_k \neq M_l$, when executing $\mathsf{AddOmegas}(M_{l-1}, t, M_l, V, E)$ the algorithm adds at least one $\omega$ to $M_{l-1}$

# Remarks on the Coverability Graph Algorithm

If the reachability graph is finite, the algorithm $\mathsf{AddOmegas}(M, t, M', V, E)$ will always return $M'$ as its output (i.e., the third parameter).

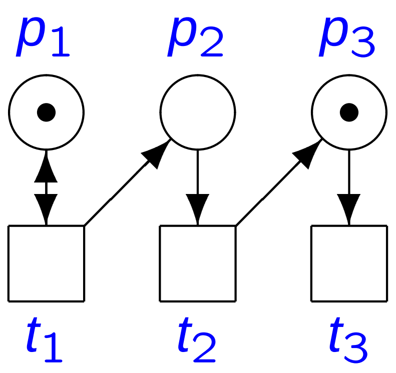
In this case the coverability graph algorithm will return the reachability graph (but it will run more slowly).

Implementations of the algorithm are bound to be slow because of the **for** loop in $\mathsf{AddOmegas}$, which has to traverse the potentially large size of the graph.

The result of the algorithm is not unique, e.g. it depends on the implementation of $Work$ and on the exact order of fired transitions on line 5 of the main routine.

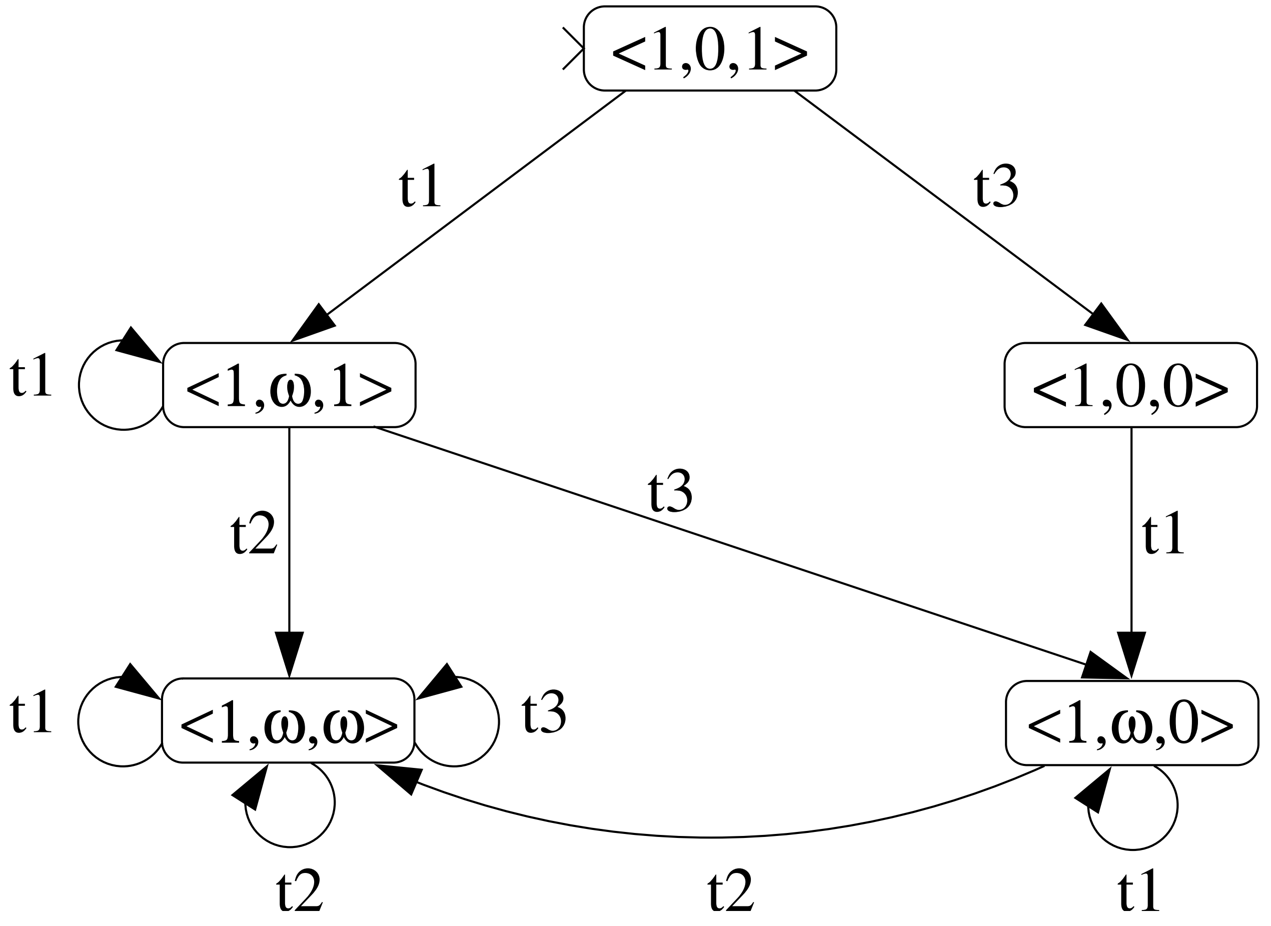
# Example 1: Coverability Graph

Recall the P/T-net example given in the previous lecture:



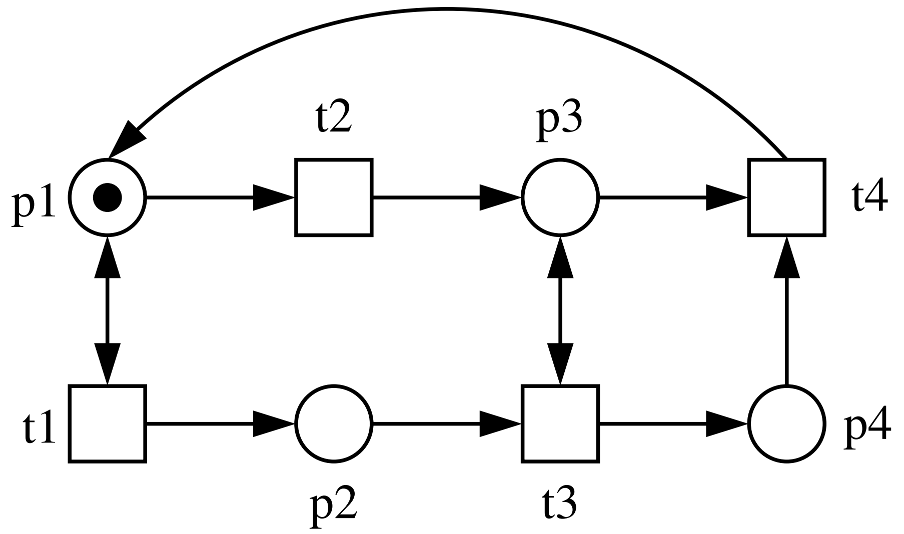
We will now compute the coverability graph for it.

# Example 1: Coverability Graph



## Example 2

Consider the following P/T-net. We will now compute a coverability graph for it.



## Example 2: Coverability graph

$$\begin{tikzcd}
 & \langle 1,0,0,0 \rangle \arrow[dl, "t1"'] \arrow[dr, "t2"] & \\
\langle 1,\omega,0,0 \rangle \arrow[loop left, "t1"] \arrow[d, "t2"'] & & \langle 0,0,1,0 \rangle \\
\langle 0,\omega,1,0 \rangle \arrow[d, "t3"'] & & \\
\langle 0,\omega,1,\omega \rangle \arrow[loop left, "t3"] \arrow[rr, bend right, "t2"'] & & \langle 1,\omega,0,\omega \rangle \arrow[loop right, "t1"] \arrow[ll, bend right, "t4"']
\end{tikzcd}$$

## Reachability and coverability graphs: Comparison (1)

Let $N = \langle P, T, F, W, M_0 \rangle$ be a net.

The reachability graph has the following fundamental property:

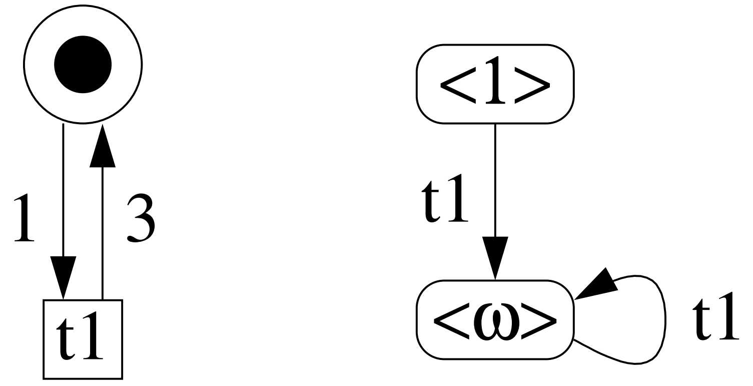
> A marking $M$ of $N$ is reachable *if and only if* $M$ is a vertex of the reachability graph of $N$.

The coverability graph has the following fundamental property:

> If a marking $M$ of $N$ is reachable, then $M$ is covered by some vertex of the coverability graph of $N$.

Notice that the first property is an equivalence, the second one an implication!

More specifically, the reverse implication *does not* hold: A marking that is covered by some vertex of the coverability graph is not necessarily reachable, as shown by the following example:



In the net, only markings with an odd number of tokens are reachable, but markings with an even number of tokens are also covered.

## Reachability and coverability graphs: Comparison (2)

The reachability graph captures exact information about the reachable markings (but its computation may not terminate).

The coverability graph computes an overapproximation
(but remains exact as long as the number of markings is finite).

## Summary: Which properties can we check so far?

Reachability: Given some marking $M$ and a net $N$, is $M$ reachable in $N$?
More generally: Given a set of markings $\mathcal{M}$, is some marking of $\mathcal{M}$ reachable?

Application: This is often used to check whether some ‘bad’ state can occur (classical example: violation of mutual exclusion property) if $\mathcal{M}$ is taken to be the set of ‘error’ states. Sometimes (as in the man/wolf/etc example), this analysis can check for the existence of a solution to some problem.

Using the reachability graph: Exact answer is obtained.

Using the coverability graph: Approximate answer. When looking for ‘bad’ states, this analysis is safe in the sense that bad states will not be missed, but the graph may indicate ‘spurious’ errors.

## Summary (cont'd)

Finding paths: Given a reachable marking $M$, find a firing sequence that leads from $M_0$ to $M$.

Application: Used to supplement reachability queries. If $M$ represents an error state, the firing sequence can be useful for debugging. When solving puzzles, the path represents actions leading to the solution.

Using the reachability graph: Find a path from $M_0$ to $M$ in the graph, obtain sequence from edge labels.

Using the coverability graph: Not so suitable – edges may represent 'shortcuts' (unspecified repetitions of some loop).

## Summary (cont'd)

Enabledness: Given some transition $t$, is there a reachable marking in which $t$ is enabled?
(Sometimes, $t$ is called dead if the answer is no. Actually, this is a special case of reachability.)

Application: Check whether some 'bad' action is possible. Also, is some desirable action is never enabled, a 'no' answer is an indication of some problem with the model.
In some Petri-net tools, checking for enabledness is easier to specify than checking for reachability. In that case, reachability queries can be framed as enabledness queries by adding 'artificial' transitions that can fire iff a given marking is reachable.

Using the reachability graph: Check whether there is an edge labeled with $t$.

Using the coverability graph: ?

## Summary (cont'd)

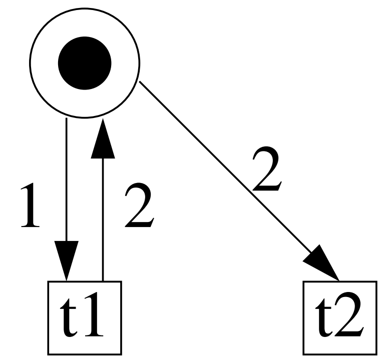
Deadlocks: Given a net $N$, is $N$ deadlock-free?

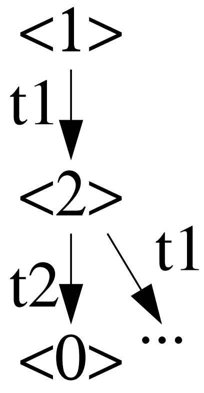
> A marking $M$ of a Place/Transition net $N = \langle P, T, F, W, M_0 \rangle$ is called a deadlock if no transition $t \in T$ is enabled in $M$. A net $N$ is deadlock-free if no reachable marking is a deadlock

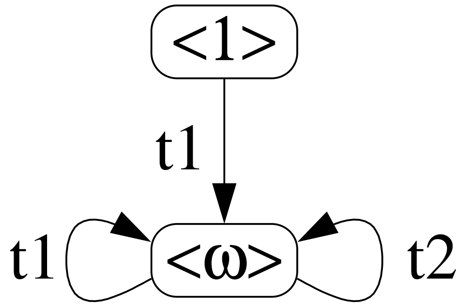
Application: Deadlocks tend to indicate errors (classical example: philosophers may starve).

Using the reachability graph: Check whether there is a vertex without an outgoing edge.

Using the coverability graph: Unsuitable – the graph may miss deadlocks!







## Summary (cont'd)

Boundedness: Given a net $N$, is there a constant $k$ such that $N$ is $k$-safe? Otherwise, which places can assume an unbounded number of tokens?

Application: If tokens represent available resources, unbounded numbers of tokens may indicate some problem (e.g. a resource leak). Also, this property should be checked *before* computing the reachability graph!

Using the reachability graph: Unsuitable, computation may not terminate.

Using the coverability graph:

> A place $p$ can assume an unbounded number of tokens iff the coverability graph contains a vertex $M$ where $M(p) = \omega$.
> Iff no vertex with an $\omega$ exists, then the net is $k$-safe, where $k$ is the largest natural number in a marking of the graph.

## What is missing? (Outlook)

Sometimes, properties mentioned in the summary can be checked even *without constructing the reachability graph* (which can be pretty large, after all).

Methods for doing this are collectively called structural analyses. These will be covered in the next lecture.

So far, we have not learnt how to express (and check) properties like these:

Marking $M$ can be reached infinitely often.

Whenever transition $t$ occurs, transition $t'$ occurs later.

No marking with some property $x$ occurs before some marking with property $y$ has occurred.

Properties like these can be expressed using temporal logic.

## Structural analysis of P/T nets

## Structural analysis of P/T nets

## Structural Analysis: Motivation

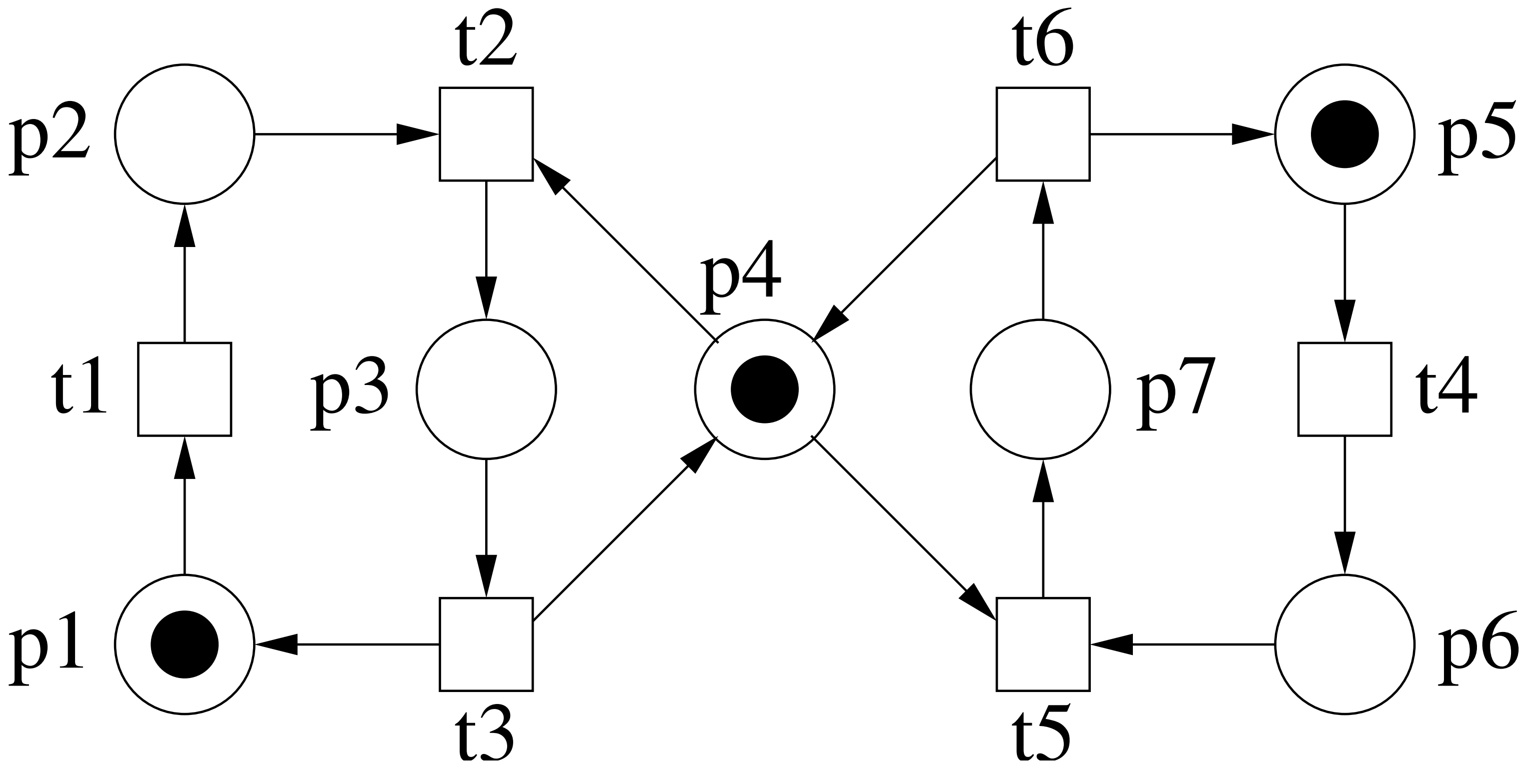
We have seen how properties of Petri nets can be proved by constructing the reachability graph and analysing it.

However, the reachability graph may become huge: exponential in the number of places (if it is finite at all).

Structural analysis makes it possible to prove some properties *without* constructing the reachability graph. The main techniques are:

- Place invariants
- Traps

## Example 1



# Incidence Matrix: Definition

Let $N = \langle P, T, F, W, M_0 \rangle$ be a P/T net. The corresponding incidence matrix $C_N : P \times T \to \mathbb{Z}$ is the matrix whose rows correspond to places and whose columns correspond to transitions. Column $t \in T$ denotes how the firing of $t$ affects the marking of the net: $C(t, p) = W(t, p) - W(p, t)$.

The incidence matrix of the example from the previous slide:

$$\begin{array}{cccccccc}
 & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & \\
\left(\right. & -1 & 0 & 1 & 0 & 0 & 0 & \left.\right) \; p_1 \\
 & 1 & -1 & 0 & 0 & 0 & 0 & p_2 \\
 & 0 & 1 & -1 & 0 & 0 & 0 & p_3 \\
 & 0 & -1 & 1 & 0 & -1 & 1 & p_4 \\
 & 0 & 0 & 0 & -1 & 0 & 1 & p_5 \\
 & 0 & 0 & 0 & 1 & -1 & 0 & p_6 \\
 & 0 & 0 & 0 & 0 & 1 & -1 & p_7
\end{array}$$

## Markings as vectors

Let us now write marking as column vectors. E.g., the initial marking is $M_0 = (1\ 0\ 0\ 1\ 1\ 0\ 0)^T$.

Likewise, we can write firing counts as column vectors with one entry for each transition. E.g., if $t_1$, $t_2$, and $t_4$ are to fire once each, we can express this with $u = (1\ 1\ 0\ 1\ 0\ 0)^T$.

Then, the result of firing these transitions can be computed as $M_0 + C \cdot u$.

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

## Caveat

Notice: Bi-directional arcs (an arc from a place to a transition and back) cancel each other out in the matrix!

Thus, when a marking arises as the result of a matrix equation (like on the previous slide), this does not guarantee that the marking is reachable!

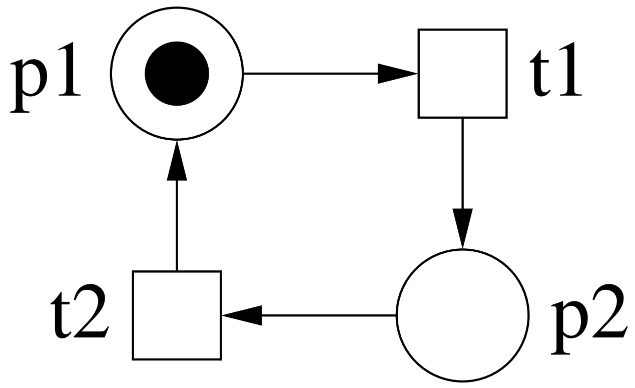
I.e., the markings obtained by the incidence markings are an over-approximation of the actual reachable markings (compare coverability graphs. . . ).

However, we *can* sometimes use the matrix equations to show that a marking $M$ is unreachable, i.e. if $M_0 + Cu = M$ has no natural solution for $u$.

Note: When we are talking about natural (integral) solutions of equations, we mean those whose components are natural (integral) numbers.

## Example 2

Consider the following net and the marking $M = (1\ 1)^T$.



$$\begin{pmatrix}1\\0\end{pmatrix} + \begin{pmatrix}-1 & 1\\1 & -1\end{pmatrix} \cdot \begin{pmatrix}u_1\\u_2\end{pmatrix} = \begin{pmatrix}1\\1\end{pmatrix}$$

has no solution, and therefore $M$ is not reachable.

## Invariants

The solutions of the equation $Cu = \mathbf{0}$ are called transition invariants (or: T-invariants). The natural solutions indicate (possible) loops.

For instance, in Example 2, $u = (1\ 1)^T$ is a T-invariant.

The solutions of the equation $C^T x = \mathbf{0}$ are called place invariants (or: P-invariants). A proper P-invariant is a solution of $C^T x = \mathbf{0}$ if $x \neq \mathbf{0}$.

For instance, in Example 1, $x_1 = (1\ 1\ 1\ 0\ 0\ 0\ 0)^T$, $x_2 = (0\ 0\ 1\ 1\ 0\ 0\ 1)^T$, and $x_3 = (0\ 0\ 0\ 0\ 1\ 1\ 1)^T$ are all (proper) P-invariants.

A P-invariant indicates that the number of tokens in all reachable markings satisfies some linear invariant (see next slide).

## Properties of P-invariants

Let $M$ be marking reachable with a transition sequence whose firing count is expressed by $u$, i.e. $M = M_0 + Cu$. Let $x$ be a P-invariant. Then, the following holds:

$$M^T x = (M_0 + Cu)^T x = M_0^T x + (Cu)^T x = M_0^T x + u^T C^T x = M_0^T x$$

For instance, invariant $x_2$ means that all reachable markings $M$ satisfy (reverting back to the function notation for markings):

$$M(p_3) + M(p_4) + M(p_7) = M_0(p_3) + M_0(p_4) + M_0(p_7) = 1 \tag{1}$$

As a consequence, a P-invariant in which all entries are either 0 or 1 indicates a set of places in which the number of tokens remains unchanged in all reachable markings.

Note that multiplying an invariant by a constant or component-wise addition of two invariants will again yield a P-invariant. That is, the set of all invariants is a *vector space*.

We can use P-invariants to prove mutual exclusion properties:

> According to equation 1, in every reachable marking of Example 1 exactly one of the places $p_3$, $p_4$, and $p_7$ is marked. In particular, $p_3$ and $p_7$ cannot be marked concurrently!

Another example: Mutual exclusion with token passing (demo)

## More remarks on P-invariants

P-invariants can also be useful as a *pre-processing step* for reachability analysis.

Suppose that when computing the reachability graph, the marking of a place is normally represented with $n$ bits of storage. E.g. the places $p_3$, $p_4$, and $p_7$ together would require $3n$ bits.

However, as we have discovered invariant $x_2$, we know that exactly one of the three places is marked in each reachable marking.

Thus, we just need to store in each marking *which* of the three is marked, which required just 2 bits.

## Algorithms for P-invariants

A basis of the set of all invariants can be computed using linear algebra.

There is an algorithm called “Farkas Algorithm” (by *J. Farkas*, 1902) to compute a set of so called minimal P-invariants (see the enxt slides). These are positive place invariants from which any other positive invariant can be computed by a linear combination.

Unfortunately there are P/T-nets with an exponential number of minimal P-invariants (in the number of places of the net). Thus the Farkas algorithm needs (at least) exponential time in the worst case.

The INA tool of the group of *Peter Starke* (Humboldt University of Berlin) contains a large number of algorithms for structural analysis of P/T-nets, including invariant generation.

# Farkas Algorithm

Input: the incidence matrix $C$ with $n$ rows (places), and $m$ columns (transitions).

$(C \mid E_n)$ denotes the augmentation of $C$ by a $n \times n$ identity matrix (last $n$ columns).

```
$D_0 := (C \mid E_n)$;
**for** $i := 1$ **to** $m$ **do**
    **for** $d_1$, $d_2$ rows in $D_{i-1}$ such that $d_1(i)$ and $d_2(i)$ have opposite signs **do**
        $d := |d_2(i)| \cdot d_1 + |d_1(i)| \cdot d_2$;    (* $d(i) = 0$ *)
        $d' := d/\text{gcd}(d(1), d(2), \ldots, d(m+n))$;
        augment $D_{i-1}$ with $d'$ as last row;
    **endfor**;
    delete all rows of the (augmented) matrix $D_{i-1}$ whose $i$-th component
      is different from $0$, the result is $D_i$;
**endfor**;
delete the first $m$ columns of $D_m$
```

## An example

Incidence matrix

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

$$D_0 = (C \mid E_5) = \left(\begin{array}{cccc|ccccc} -1 & 1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right)$$

Addition of the rows 1 and 2, 1 and 4, 2 and 5, 4 and 5:

$$D_1 = \left(\begin{array}{cccc|ccccc} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & -2 & 1 & 0 & 0 & 1 & 0 \\ 0 & -1 & -1 & 2 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right)$$

Addition of rows 3 und 4:

$$D_2 = \left(\begin{array}{cccc|ccccc} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{array}\right)$$

$$D_3 = D_4 = \left(\begin{array}{cccc|ccccc} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{array}\right)$$

Minimal P-invariants are $(1, 1, 0, 0, 0)$ and $(0, 0, 0, 1, 1)$.

## An example with many P-invariants

Incidence matrix for a net with $2n$ places:

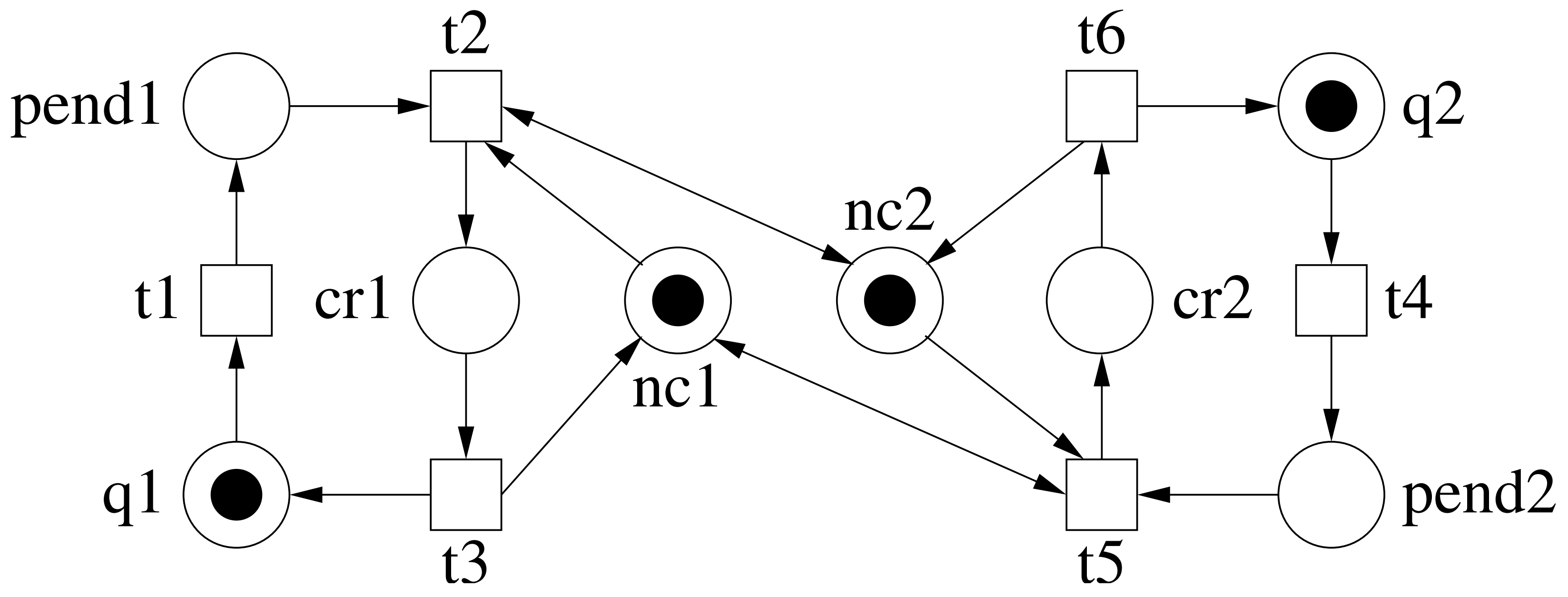
$$C = \begin{pmatrix} -1 & 0 & 0 & & 0 & 1 \\ -1 & 0 & 0 & & 0 & 1 \\ 1 & -1 & 0 & \cdots & 0 & 0 \\ 1 & -1 & 0 & & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \\ 0 & 0 & 0 & \cdots & 1 & -1 \\ 0 & 0 & 0 & \cdots & 1 & -1 \end{pmatrix}$$

$(y_1, 1-y_1, y_2, 1-y_2, \ldots, y_n, 1-y_n)$ is an invariant for every $y_1, y_2, \ldots, y_n \in \{0,1\}$, and so there are $2^n$ minimal P-invariants.

This example shows that the number of minimal P-invariants can be exponential in the size of the net. So Farkas algorithm may need exponential time.

## Example 3

Consider the following attempt at a mutual exlusion algorithm for $cr_1$ and $cr_2$:



The idea is to achieve mutual exclusion by entering the critical section only if the other process is not already there.

Thus, we want to prove that in all reachable markings $M$:

$$M(cr_1) + M(cr_2) \leq 1$$

The P-invariants we can derive in the net yield:

$$\begin{aligned} M(q_1) + M(pend_1) + M(cr_1) &= 1 && (2) \\ M(q_2) + M(pend_2) + M(cr_2) &= 1 && (3) \\ M(cr_1) + M(nc_1) &= 1 && (4) \\ M(cr_2) + M(nc_2) &= 1 && (5) \end{aligned}$$

But try as we might, we cannot show the desired property just with these four equations!

## Traps

Definition: Let $\langle P, T, F, W, M_0 \rangle$ be a P/T net.
A trap is a set of places $S \subseteq P$ such that $S^\bullet \subseteq {}^\bullet S$.

In other words, each transition which removes tokens from a trap must also put at least one token back to the trap.

A trap $S$ is called marked in marking $M$ iff for at least one place $s \in S$ it holds that $M(s) \geq 1$.

Note: If a trap $S$ is marked in $M_0$, then it is also marked in all reachable markings.

In Example 3, $S_1 = \{nc_1, nc_2\}$ is a trap.

> The only transitions that remove tokens from this set are $t_2$ and $t_5$. However, both also add new tokens to $S_1$.

$S_1$ is marked initially and therefore in all reachable markings $M$. Thus:

$$M(nc_1) + M(nc_2) \geq 1 \tag{6}$$

Traps can be useful in combination with place invariants to recapture information lost in the incidence matrix due to the cancellation of self-loop arcs.

Here: Adding (4) and (5) and subtracting (6) yields $M(cr_1) + M(cr_2) \leq 1$, which proves the mutual exclusion property.