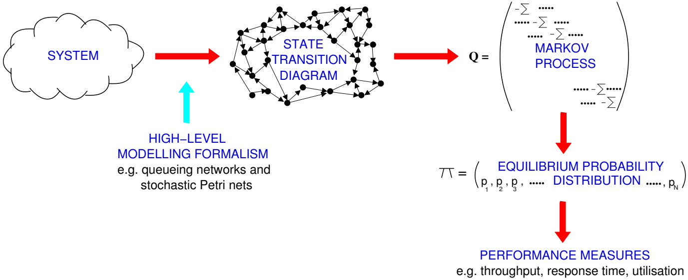
Performance Modelling — Lecture 7
Stochastic Petri Nets

Jane Hillston
School of Informatics
The University of Edinburgh
Scotland

6th February 2017

Motivation



As discussed in a previous lecture, high level modelling formalisms are used to make the job of constructing the state transition diagram and/or infinitesimal generator matrix easier. The second such language we consider is Stochastic Petri Nets.

Petri Nets

Petri nets are a formal notation designed for modelling concurrency, causality and conflict.

Petri Nets

Petri nets are a formal notation designed for modelling concurrency, causality and conflict.

A Petri net is a bipartite graph, and this graphical representation gives the formalism an easier intuitive interpretation than the Markov process .

Petri Nets

Petri nets are a formal notation designed for modelling concurrency, causality and conflict.

A Petri net is a bipartite graph, and this graphical representation gives the formalism an easier intuitive interpretation than the Markov process — at least for small or moderately sized models.

Petri Nets

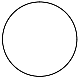
Petri nets are a formal notation designed for modelling concurrency, causality and conflict.

A Petri net is a bipartite graph, and this graphical representation gives the formalism an easier intuitive interpretation than the Markov process — at least for small or moderately sized models.

Petri nets were introduced in the 1960s for modelling a variety of concurrent systems, but their use for performance modelling originates from 1980s.

Basic Definitions

The primitives of the notation are the following:

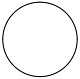


PLACES

Places are used to represent conditions or local system states, e.g. a place may relate to one phase in the behaviour of a particular component.

Basic Definitions

The primitives of the notation are the following:



PLACES

Places are used to represent conditions or local system states, e.g. a place may relate to one phase in the behaviour of a particular component.

TRANSITIONS

Transitions are used to describe events that occur in the system; these will usually result in a modification to the system state.

Basic Definitions

•

TOKENS

Tokens are identity-less markers that reside in places. The presence of a token in a place indicates that the corresponding condition or local state holds.

Basic Definitions

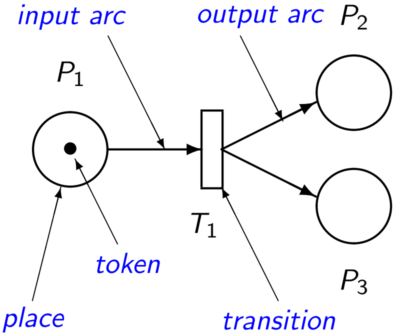
TOKENS

Tokens are identity-less markers that reside in places. The presence of a token in a place indicates that the corresponding condition or local state holds.

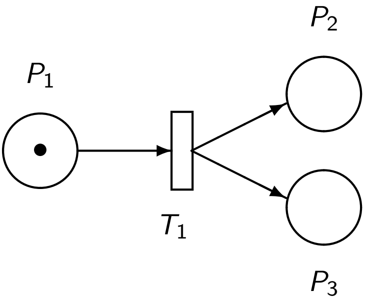
ARCS

Arcs specify the relationships between local states or conditions (places) and events (transitions). An arc from a place to a transition is termed an input arc. This indicates the local state in which the event can occur. An arc to a place from a transition is termed an output arc.

Anatomy of a Petri net

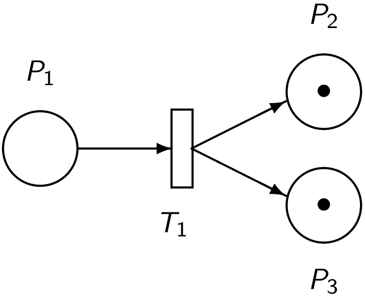


Example firing



When a transition fires tokens from input places are absorbed and tokens are created on each of the output places.

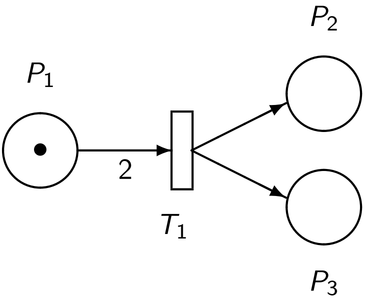
Example firing



When a transition fires tokens from input places are absorbed and tokens are created on each of the output places.

This is instantaneous.

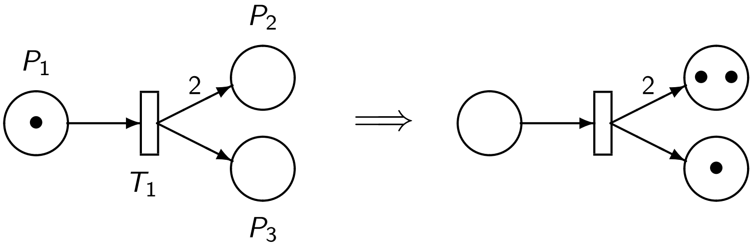
Arcs with multiplicities



If a multiplicity is associated with an arc, the firing rule is adjusted to reflect the multiplicity by altering the number of tokens required or produced.

For example, the net above cannot fire.

Arcs with multiplicities



If a multiplicity is associated with an arc, the firing rule is adjusted to reflect the multiplicity by altering the number of tokens required or produced.

Petri net marking

The state of the Petri net system at any time, is characterised by the distribution of tokens over the places, generally termed a marking: $m : P \longrightarrow \mathbb{N}_0$, where $m(p) = n$ means that there are n tokens on place p.

The firing rule more formally

A transition t is said to be enabled in a marking m, written as $m[t\rangle$, if all the pre-places of t (those connected by an input arc) have a marking that is greater than or equal to the multiplicity of that input arc.

The firing rule more formally

A transition t is said to be enabled in a marking m, written as $m[t\rangle$, if all the pre-places of t (those connected by an input arc) have a marking that is greater than or equal to the multiplicity of that input arc.

Otherwise t is said to be disabled.

The firing rule more formally

A transition t is said to be enabled in a marking m, written as $m[t\rangle$, if all the pre-places of t (those connected by an input arc) have a marking that is greater than or equal to the multiplicity of that input arc.

Otherwise t is said to be disabled.

A transition which is enabled in m may fire.

The firing rule more formally

A transition t is said to be enabled in a marking m, written as $m[t\rangle$, if all the pre-places of t (those connected by an input arc) have a marking that is greater than or equal to the multiplicity of that input arc.

Otherwise t is said to be disabled.

A transition which is enabled in m may fire.

When t in m fires, a new marking m' is reach, written as $m[t\rangle m'$.

Reachability

Starting from an initial marking and following the firing rule we can progress through the states of the model. This is sometimes called playing the token game.

Reachability

Starting from an initial marking and following the firing rule we can progress through the states of the model. This is sometimes called playing the token game.

Continuing in this way, we obtain the reachability set, all the possible states of the model.

Reachability

Starting from an initial marking and following the firing rule we can progress through the states of the model. This is sometimes called playing the token game.

Continuing in this way, we obtain the reachability set, all the possible states of the model.

Different initial markings might lead to different reachability sets which is why the initial marking is an important part of the model definition.

Reachability

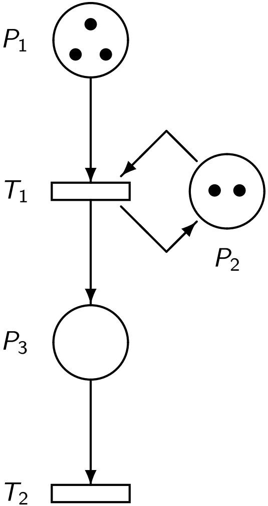
Starting from an initial marking and following the firing rule we can progress through the states of the model. This is sometimes called playing the token game.

Continuing in this way, we obtain the reachability set, all the possible states of the model.

Different initial markings might lead to different reachability sets which is why the initial marking is an important part of the model definition.

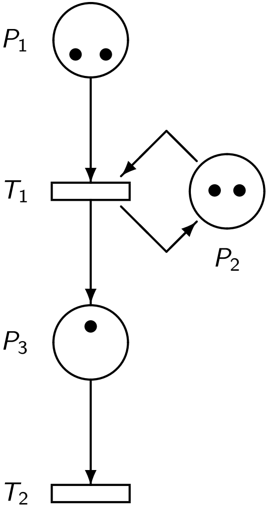
If, when playing the token game, as well as all the states we come across, we record the transitions between those states, we obtain the reachability graph.

Example: reachability graph



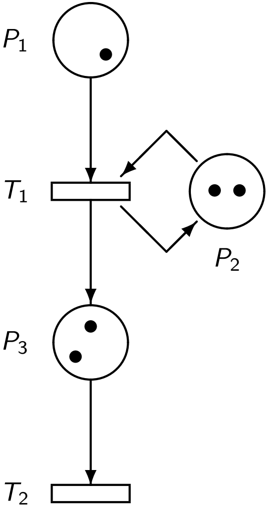
$(3, 2, 0)$

Example: reachability graph



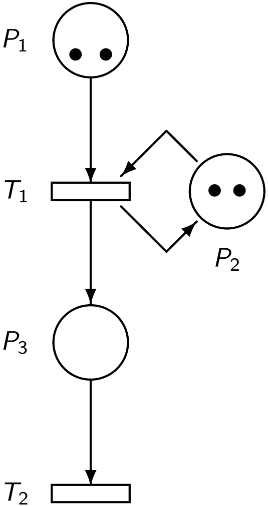
$$(3,2,0) \xrightarrow{T_1} (2,2,1)$$

Example: reachability graph



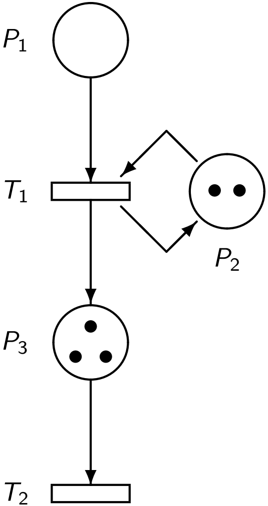
$$(3,2,0) \xrightarrow{T_1} (2,2,1) \xrightarrow{T_1} (1,2,2)$$

Example: reachability graph



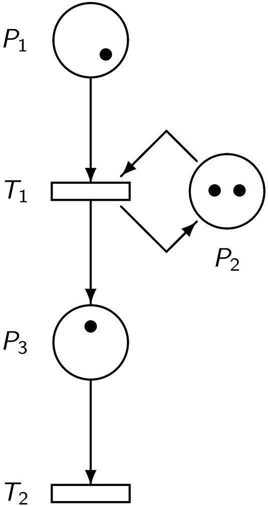
$$
\begin{array}{ccc}
(3,2,0) & & \\
\downarrow T_1 & & \\
(2,2,1) & \xrightarrow{T_2} & \color{red}{(2,2,0)} \\
\downarrow T_1 & & \\
(1,2,2) & &
\end{array}
$$

Example: reachability graph



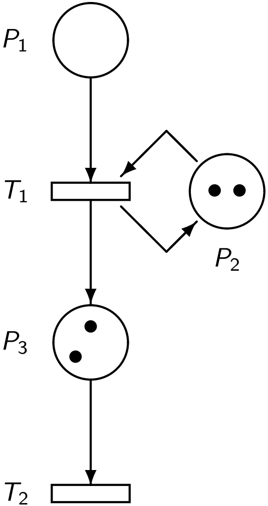
$$
\begin{array}{ccc}
(3,2,0) & & \\
\downarrow T_1 & & \\
(2,2,1) & \xrightarrow{T_2} & (2,2,0) \\
\downarrow T_1 & & \\
(1,2,2) & & \\
\downarrow T_1 & & \\
(0,2,3) & &
\end{array}
$$

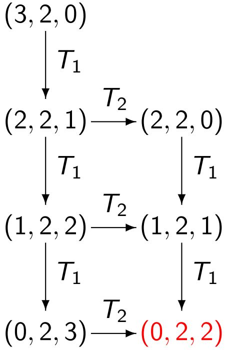
Example: reachability graph



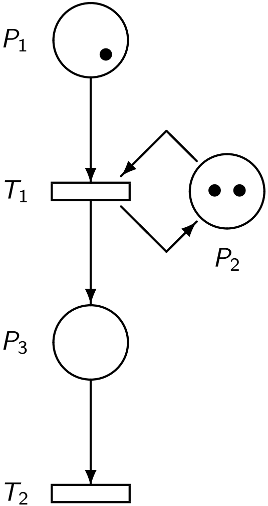
$$\begin{tikzcd}
(3,2,0) \arrow[d, "T_1"] & \\
(2,2,1) \arrow[r, "T_2"] \arrow[d, "T_1"] & (2,2,0) \arrow[d, "T_1"] \\
(1,2,2) \arrow[r, "T_2"] \arrow[d, "T_1"] & \color{red}{(1,2,1)} \\
(0,2,3) &
\end{tikzcd}$$

Example: reachability graph



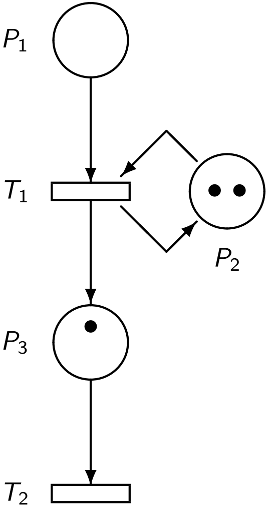


Example: reachability graph



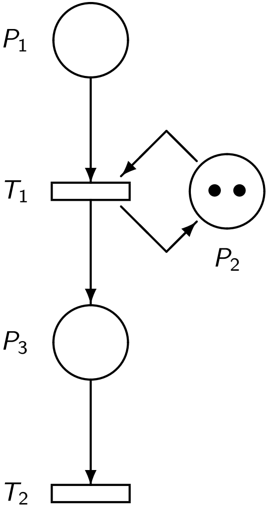
$$
\begin{tikzcd}
(3,2,0) \arrow[d, "T_1"] & & \\
(2,2,1) \arrow[r, "T_2"] \arrow[d, "T_1"] & (2,2,0) \arrow[d, "T_1"] & \\
(1,2,2) \arrow[r, "T_2"] \arrow[d, "T_1"] & (1,2,1) \arrow[r, "T_2"] \arrow[d, "T_1"] & \color{red}{(1,2,0)} \\
(0,2,3) \arrow[r, "T_2"] & (0,2,2) &
\end{tikzcd}
$$

Example: reachability graph



$$
\begin{array}{ccccc}
(3,2,0) & & & & \\
\downarrow T_1 & & & & \\
(2,2,1) & \xrightarrow{T_2} & (2,2,0) & & \\
\downarrow T_1 & & \downarrow T_1 & & \\
(1,2,2) & \xrightarrow{T_2} & (1,2,1) & \xrightarrow{T_2} & (1,2,0) \\
\downarrow T_1 & & \downarrow T_1 & & \downarrow T_1 \\
(0,2,3) & \xrightarrow{T_2} & (0,2,2) & \xrightarrow{T_2} & \color{red}{(0,2,1)}
\end{array}
$$

Example: reachability graph



$$
\begin{tikzcd}
(3,2,0) \arrow[d, "T_1"] & & & \\
(2,2,1) \arrow[r, "T_2"] \arrow[d, "T_1"] & (2,2,0) \arrow[d, "T_1"] & & \\
(1,2,2) \arrow[r, "T_2"] \arrow[d, "T_1"] & (1,2,1) \arrow[r, "T_2"] \arrow[d, "T_1"] & (1,2,0) \arrow[d, "T_1"] & \\
(0,2,3) \arrow[r, "T_2"] & (0,2,2) \arrow[r, "T_2"] & (0,2,1) \arrow[r, "T_2"] & \color{red}{(0,2,0)}
\end{tikzcd}
$$

Stochastic Petri Nets

Stochastic Petri nets (SPN) emerged as a modelling formalism for performance analysis in the early 1980s.

Stochastic Petri Nets

Stochastic Petri nets (SPN) emerged as a modelling formalism for performance analysis in the early 1980s.

In a stochastic Petri nets we associate an exponentially distributed delay with the firing of each transition.

Stochastic Petri Nets

Stochastic Petri nets (SPN) emerged as a modelling formalism for performance analysis in the early 1980s.

In a stochastic Petri nets we associate an exponentially distributed delay with the firing of each transition.

The delay occurs between when the transition becomes enabled and when it fires; indeed the instantaneous firing will only occur if the transition has remained enabled throughout the delay period.

Stochastic Petri Nets

Stochastic Petri nets (SPN) emerged as a modelling formalism for performance analysis in the early 1980s.

In a stochastic Petri nets we associate an exponentially distributed delay with the firing of each transition.

The delay occurs between when the transition becomes enabled and when it fires; indeed the instantaneous firing will only occur if the transition has remained enabled throughout the delay period.

It is straightforward to show that the reachability graph of a SPN forms the state transition diagram of an underlying Markov process.

Multi-processor example as a SPN

We consider again the simple multi-processor system, initially just with one processor.

We make a slight modification to the system: we now explicitly represent the processor requesting and gaining access to the common memory.

Multi-processor example as a SPN

We consider again the simple multi-processor system, initially just with one processor.

We make a slight modification to the system: we now explicitly represent the processor requesting and gaining access to the common memory.

In the previous model we used a higher level of abstraction in which this action was ignored.

Multi-processor example as a SPN

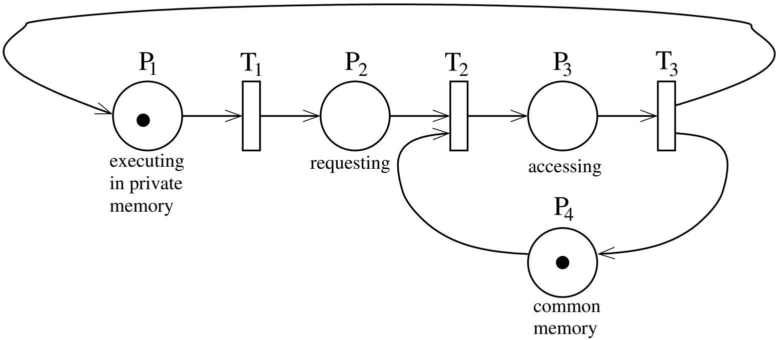
We consider again the simple multi-processor system, initially just with one processor.

We make a slight modification to the system: we now explicitly represent the processor requesting and gaining access to the common memory.

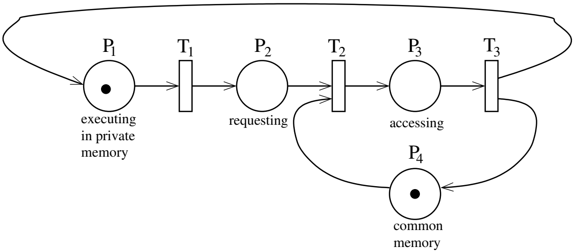
In the previous model we used a higher level of abstraction in which this action was ignored.

A processor executes locally for some time (mean duration $1/\lambda$), and then requests access to common memory (gaining access has mean duration $1/r$). Once it has gained access, the duration of common memory access is assumed to be $1/\mu$ on average.

Multi-processor example as a SPN

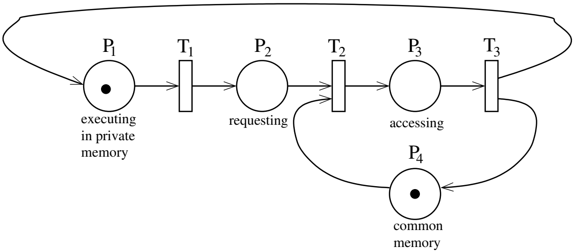


Multi-processor example as a SPN



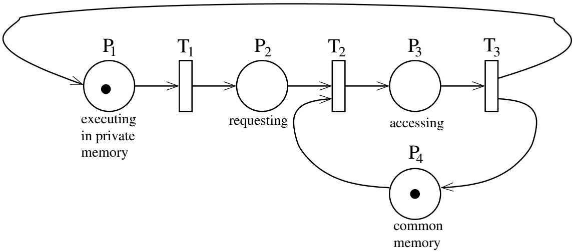
- P_1 represents the processor when it is executing privately;

Multi-processor example as a SPN



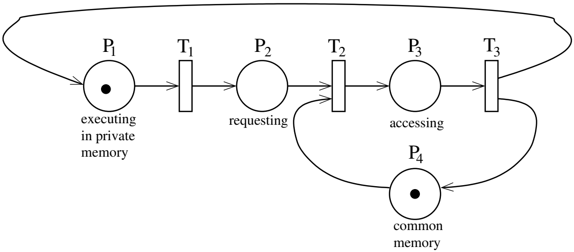
- P_1 represents the processor when it is executing privately;
- P_2 represents the processor when it is ready to access the common memory;

Multi-processor example as a SPN



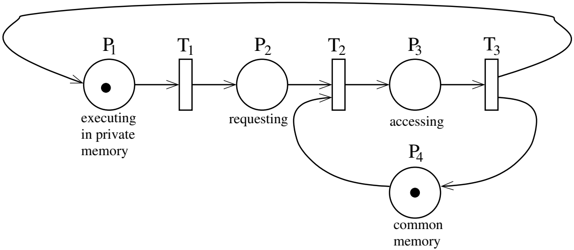
- P_1 represents the processor when it is executing privately;
- P_2 represents the processor when it is ready to access the common memory;
- P_3 represents the state when the process is using the common memory;

Multi-processor example as a SPN



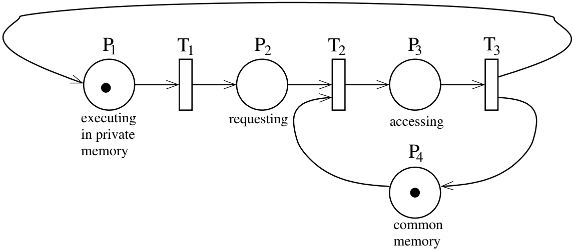
- P_1 represents the processor when it is executing privately;
- P_2 represents the processor when it is ready to access the common memory;
- P_3 represents the state when the process is using the common memory;
- P_4 represents the common memory when it is not in use;

Multi-processor example as a SPN



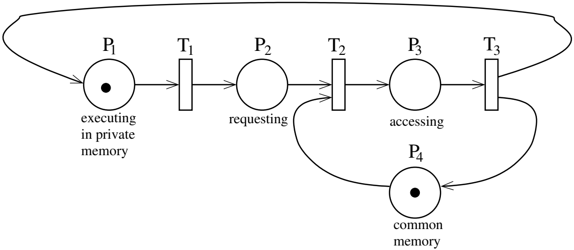
- T_1 represents the event of the processor executing privately; the rate of this transition is λ;

Multi-processor example as a SPN



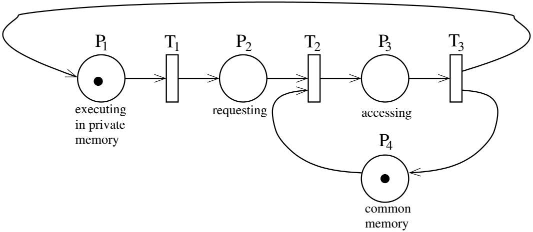
- T_1 represents the event of the processor executing privately; the rate of this transition is λ;
- T_2 represents the processor gaining access to the common memory; the rate of this transition is r;

Multi-processor example as a SPN



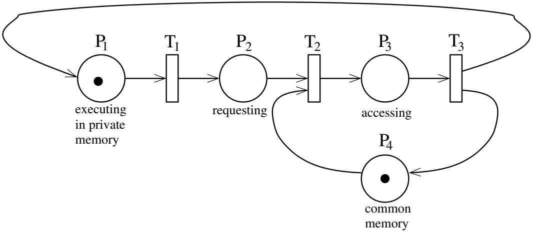
- T_1 represents the event of the processor executing privately; the rate of this transition is λ;
- T_2 represents the processor gaining access to the common memory; the rate of this transition is r;
- T_3 represents the processor accessing the common memory; the rate of this transition is μ.

Multi-processor example as a SPN

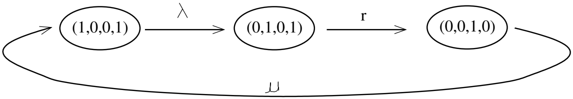


The reachability set of this model is
$\{(1,0,0,1),(0,1,0,1),(0,0,1,0)\}$,

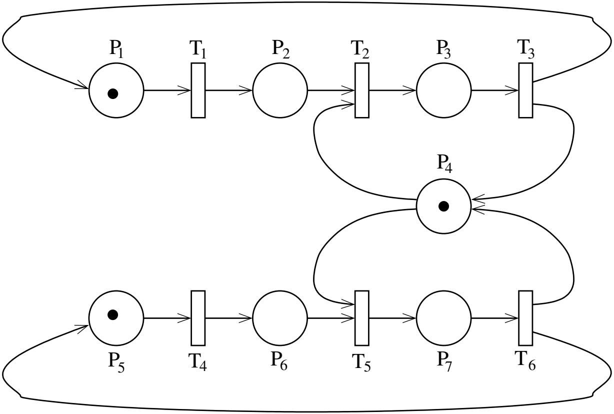
Multi-processor example as a SPN



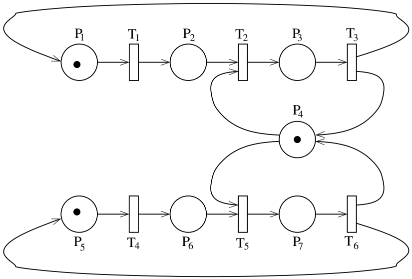
The reachability set of this model is
$\{(1,0,0,1),(0,1,0,1),(0,0,1,0)\}$,
and the reachability graph is



The two processor system

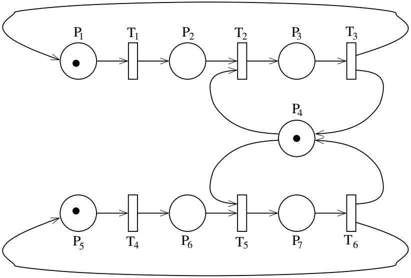


The two processor system



Now we see why the action of gaining access needs to be explicitly represented in the SPN model. Without it we could not enforce the necessary mutual exclusion between the processors.

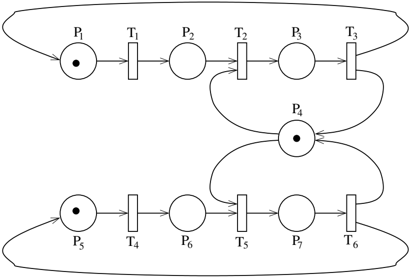
The two processor system



The reachability set is now

$$\left\{ \begin{array}{ccc} (1,0,0,1,1,0,0), & (1,0,0,1,0,1,0), & (1,0,0,0,0,0,1), \\ (0,1,0,1,1,0,0), & (0,1,0,1,0,1,0), & (0,1,0,0,0,0,1), \\ & (0,0,1,0,1,0,0), & (0,0,1,0,0,1,0) \end{array} \right\}$$

The two processor system

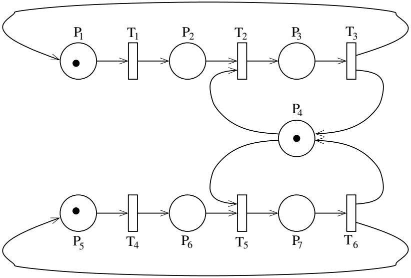


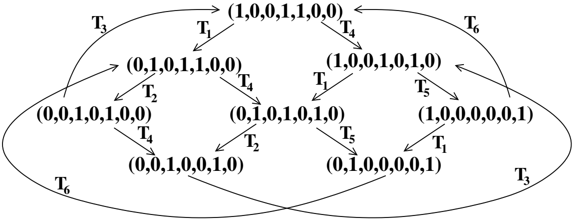
The reachability set is now

$$\left\{\begin{array}{ccc} (1,0,0,1,1,0,0), & (1,0,0,1,0,1,0), & (1,0,0,0,0,0,1), \\ (0,1,0,1,1,0,0), & (0,1,0,1,0,1,0), & (0,1,0,0,0,0,1), \\ & (0,0,1,0,1,0,0), & (0,0,1,0,0,1,0) \end{array}\right\}$$

Note, we have eight distinct states rather than the five states we had in the Markov process when we modelled the system directly.

The two processor system





Generalised Stochastic Petri Nets

SPN provide a clear and intuitive formalism for generating Markov processes.

Generalised Stochastic Petri Nets

SPN provide a clear and intuitive formalism for generating Markov processes.

However they do have the disadvantage that the models constructed in this way can soon become exceedingly large.

Generalised Stochastic Petri Nets

SPN provide a clear and intuitive formalism for generating Markov processes.

However they do have the disadvantage that the models constructed in this way can soon become exceedingly large.

One of the reasons for this is that actions which would not be explicitly represented if we were working directly at the Markov process level have to be represented by transitions.

Generalised Stochastic Petri Nets

SPN provide a clear and intuitive formalism for generating Markov processes.

However they do have the disadvantage that the models constructed in this way can soon become exceedingly large.

One of the reasons for this is that actions which would not be explicitly represented if we were working directly at the Markov process level have to be represented by transitions.

This has the effect of increasing the state space since we now have to consider the state of waiting for these actions to occur, whereas we would prefer to abstract away from these actions.

Generalised Stochastic Petri Nets

Generalised Stochastic Petri Nets (GSPN) represent an extension of the SPN formalism, which is designed to address this problem.

Generalised Stochastic Petri Nets

Generalised Stochastic Petri Nets (GSPN) represent an extension of the SPN formalism, which is designed to address this problem.

Two new primitives are added to the notation, all others remaining the same and keeping the same interpretation. These new primitives are immediate transitions and inhibitor arcs.

Immediate Transitions

Immediate transitions are used to describe events which are assumed to take no time.

IMMEDIATE TRANSITIONS

Immediate Transitions

IMMEDIATE TRANSITIONS

Immediate transitions are used to describe events which are assumed to take no time.

When they are enabled in a model they fire immediately, and they have priority over any enabled timed transitions.

Immediate Transitions

IMMEDIATE TRANSITIONS

Immediate transitions are used to describe events which are assumed to take no time.

When they are enabled in a model they fire immediately, and they have priority over any enabled timed transitions.

If two or more immediate transitions can be enabled at the same time, the probability that each of them is the one to fire must be declared in the model.

Immediate Transitions

The types of events represented by immediate actions usually fall into two categories: control actions and logical actions.

Immediate Transitions

The types of events represented by immediate actions usually fall into two categories: control actions and logical actions.

Control actions are necessary to ensure the correct behaviour of the model but which take negligible time to be executed (e.g. gaining access to the common memory in the multi-processor model)

Immediate Transitions

The types of events represented by immediate actions usually fall into two categories: control actions and logical actions.

Control actions are necessary to ensure the correct behaviour of the model but which take negligible time to be executed (e.g. gaining access to the common memory in the multi-processor model)

In these cases immediate actions provide an additional tool for abstraction within the model.

Immediate Transitions

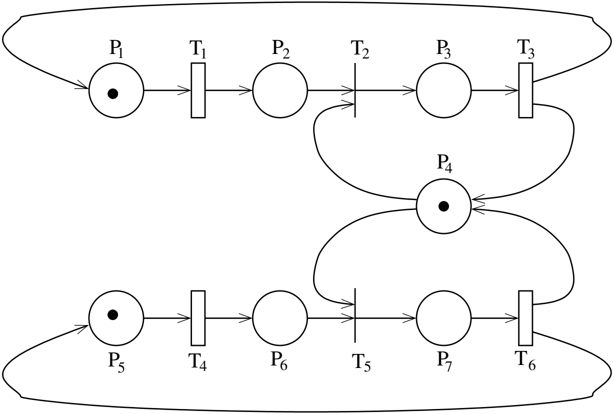
The types of events represented by immediate actions usually fall into two categories: control actions and logical actions.

Control actions are necessary to ensure the correct behaviour of the model but which take negligible time to be executed (e.g. gaining access to the common memory in the multi-processor model)

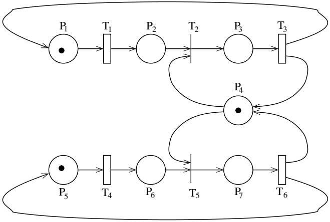
In these cases immediate actions provide an additional tool for abstraction within the model.

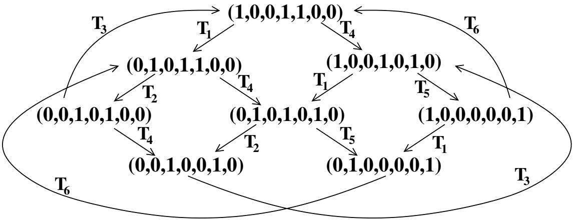
Logical actions arise when the system makes a choice between two or more alternatives. In an SPN these actions must be represented as actions which take time. However it is more natural to think of them occurring instantaneously.

Multi-processor example again

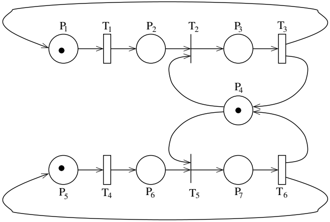


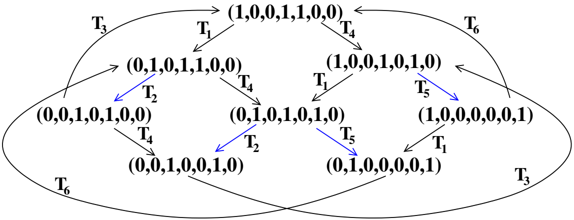
Multi-processor example again



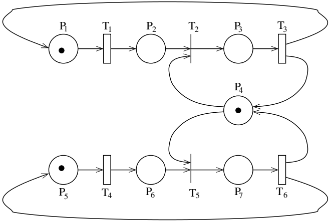


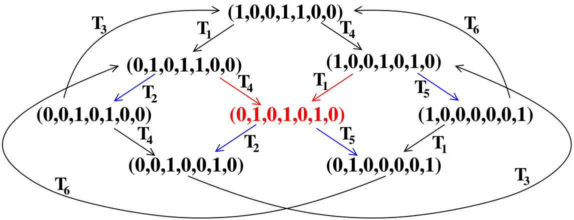
Multi-processor example again



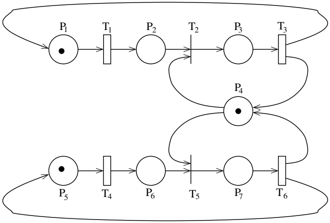


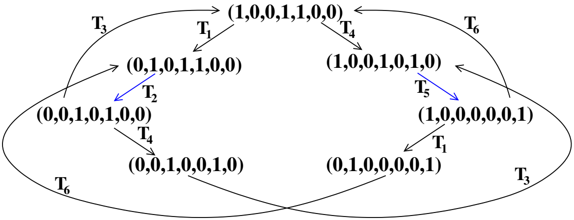
Multi-processor example again



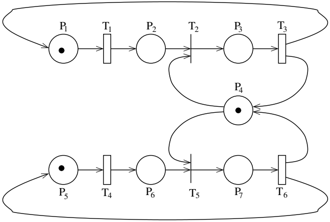


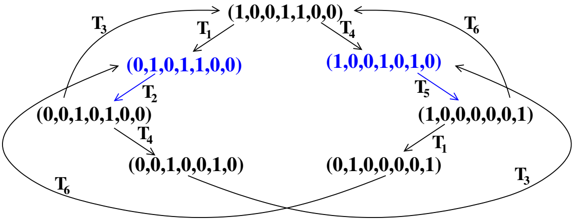
Multi-processor example again



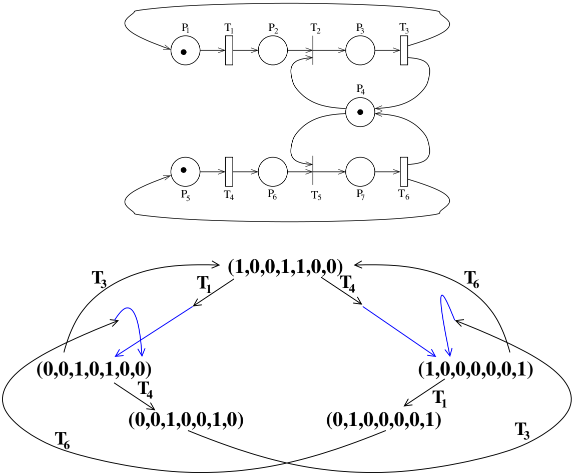


Multi-processor example again

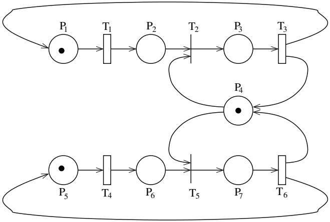


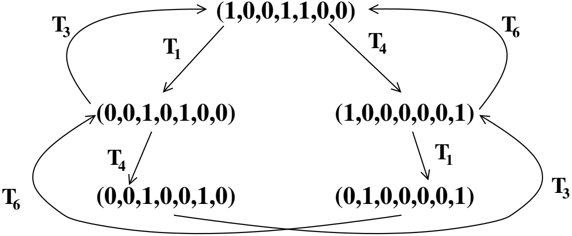


Multi-processor example again



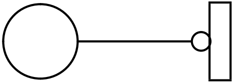
Multi-processor example again





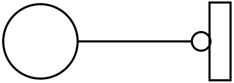
Inhibitor Arcs

An inhibitor arc is used to indicates when a local state disables a transition, rather than enables it.



INHIBITOR ARCS

Inhibitor Arcs

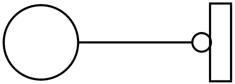


INHIBITOR ARCS

An inhibitor arc is used to indicates when a local state disables a transition, rather than enables it.

An inhibitor arc from a place to a transition, means the transition cannot fire if there is a token in the place;

Inhibitor Arcs



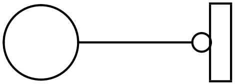
INHIBITOR ARCS

An inhibitor arc is used to indicates when a local state disables a transition, rather than enables it.

An inhibitor arc from a place to a transition, means the transition cannot fire if there is a token in the place;

It can fire when there is no token in the place if the places connected to its input arcs do contain tokens.

Inhibitor Arcs



INHIBITOR ARCS

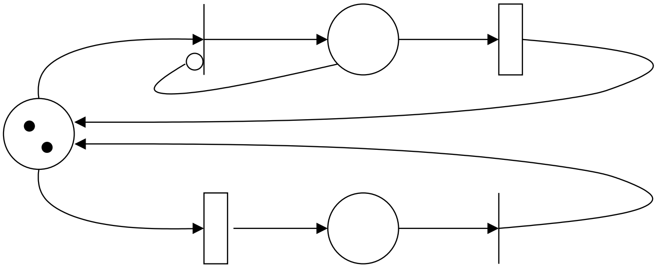
An inhibitor arc is used to indicates when a local state disables a transition, rather than enables it.

An inhibitor arc from a place to a transition, means the transition cannot fire if there is a token in the place;

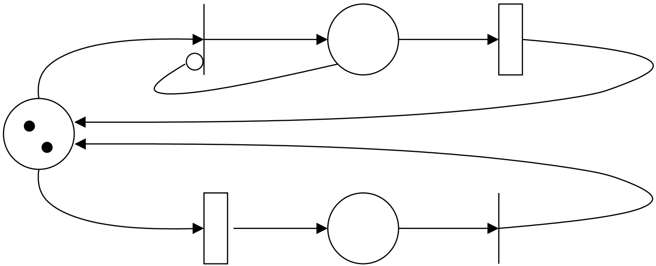
It can fire when there is no token in the place if the places connected to its input arcs do contain tokens.

The inhibitor arcs impose additional constraints to the usual firing rule.

Example with inhibitor arc



Example with inhibitor arc



Exercise: Construct the reachability graph for this GSPN with and without the inhibitor arc.