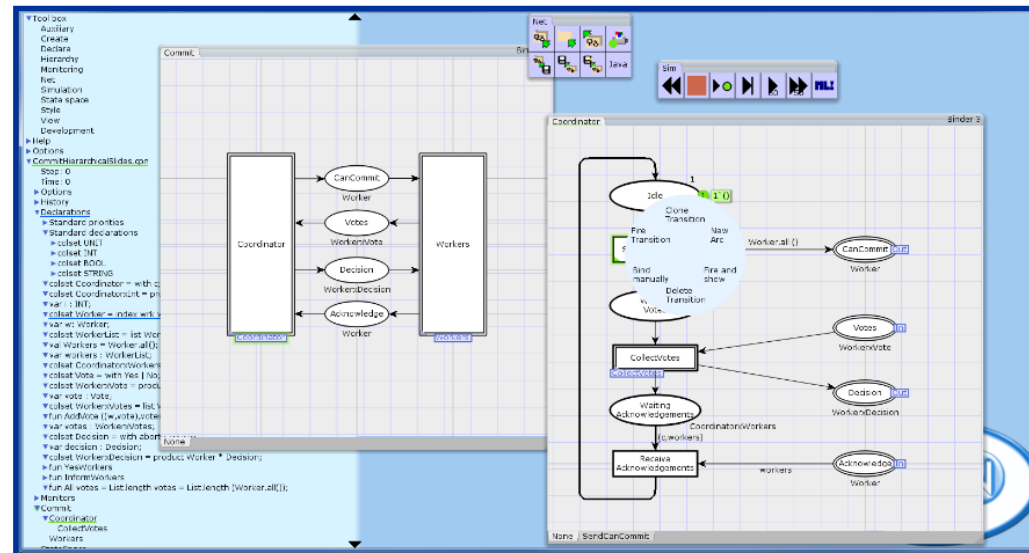


Lecture 6

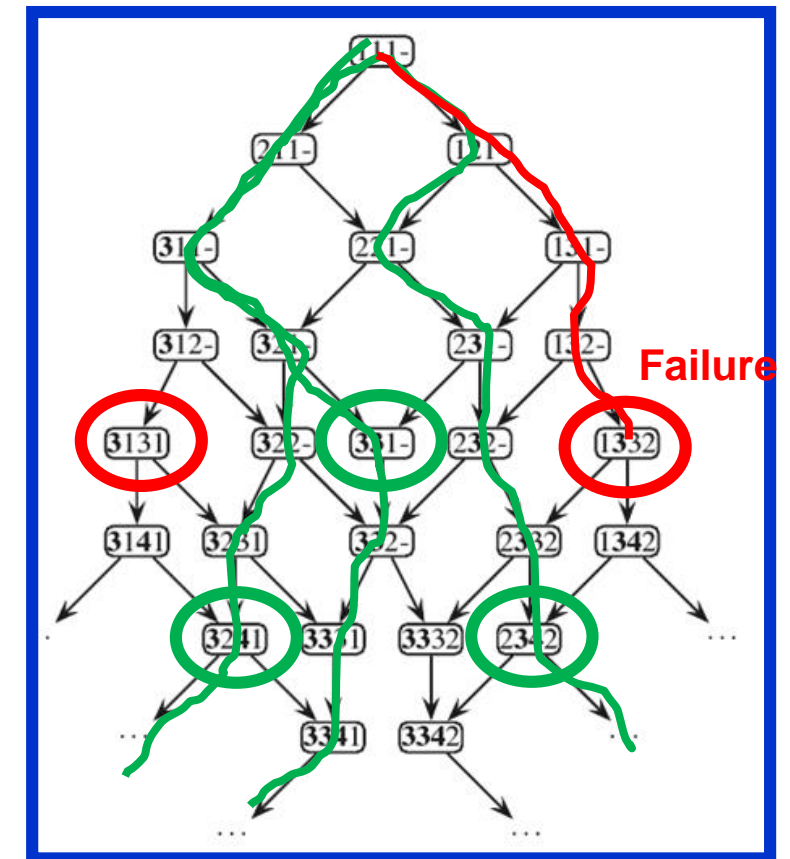
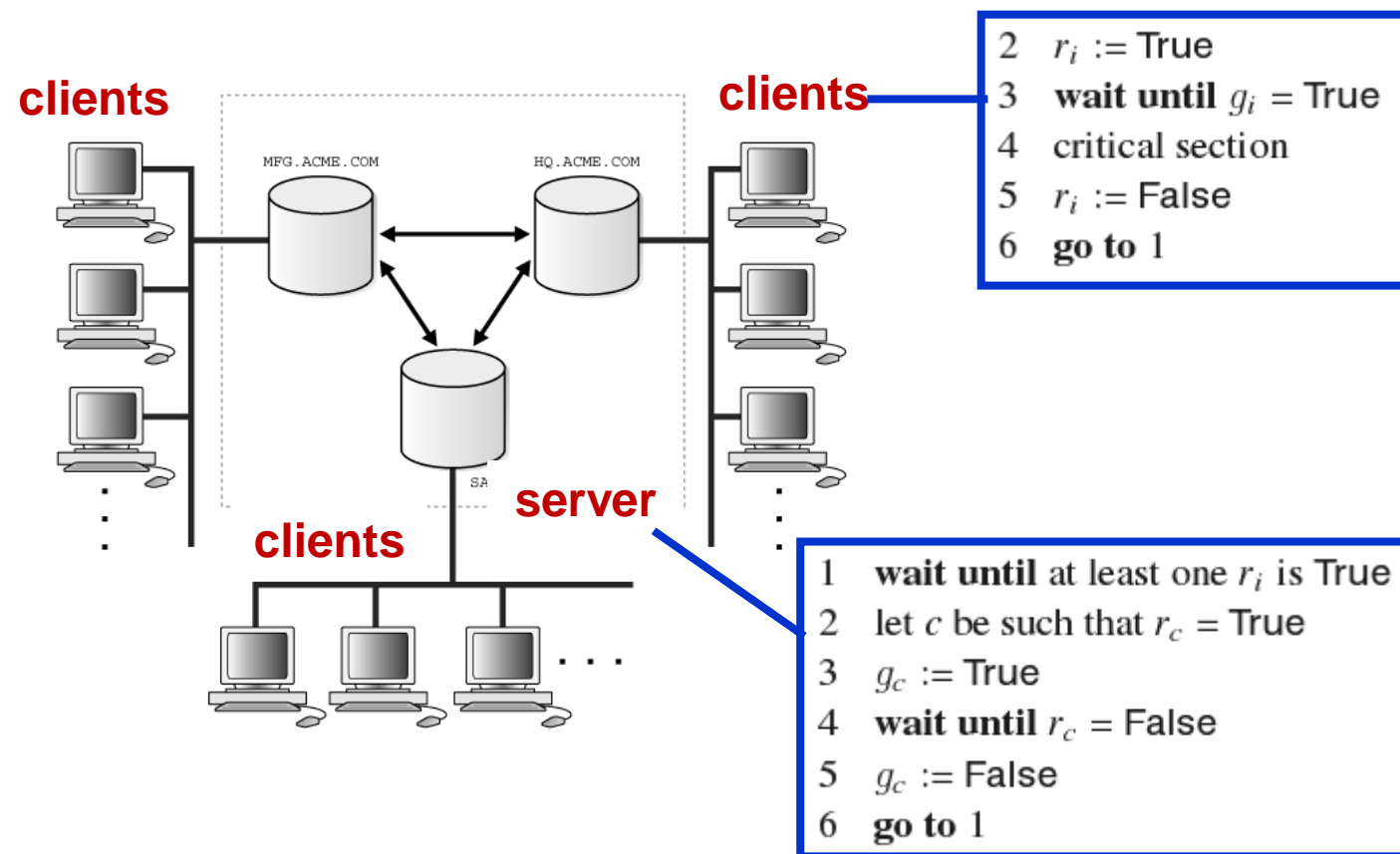
State Spaces Methods and Verification



Lars M. Kristensen
Department of Computer Science, Electrical Engineering, and Mathematical Sciences
Western Norway University of Applied Sciences
Email: lmkr@hvl.no / WWW: home.hib.no/ansatte/lmkr

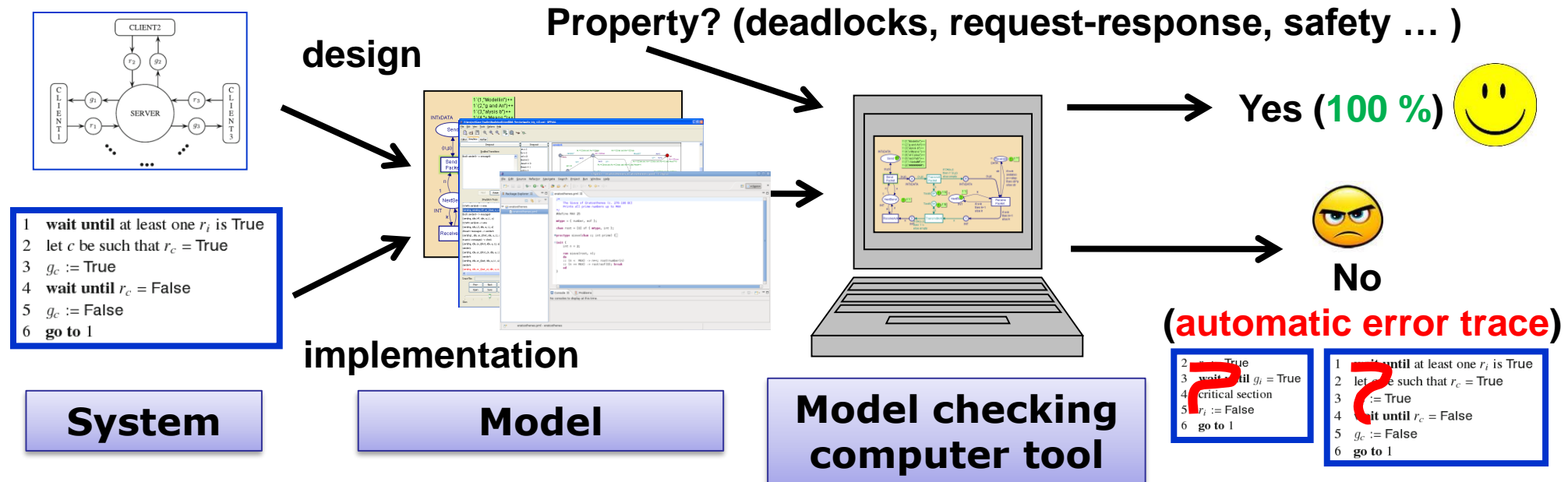
Complex behaviour

- A **server** controls access to databases which is to be **used by at most one client** at a time



Model checking and verification

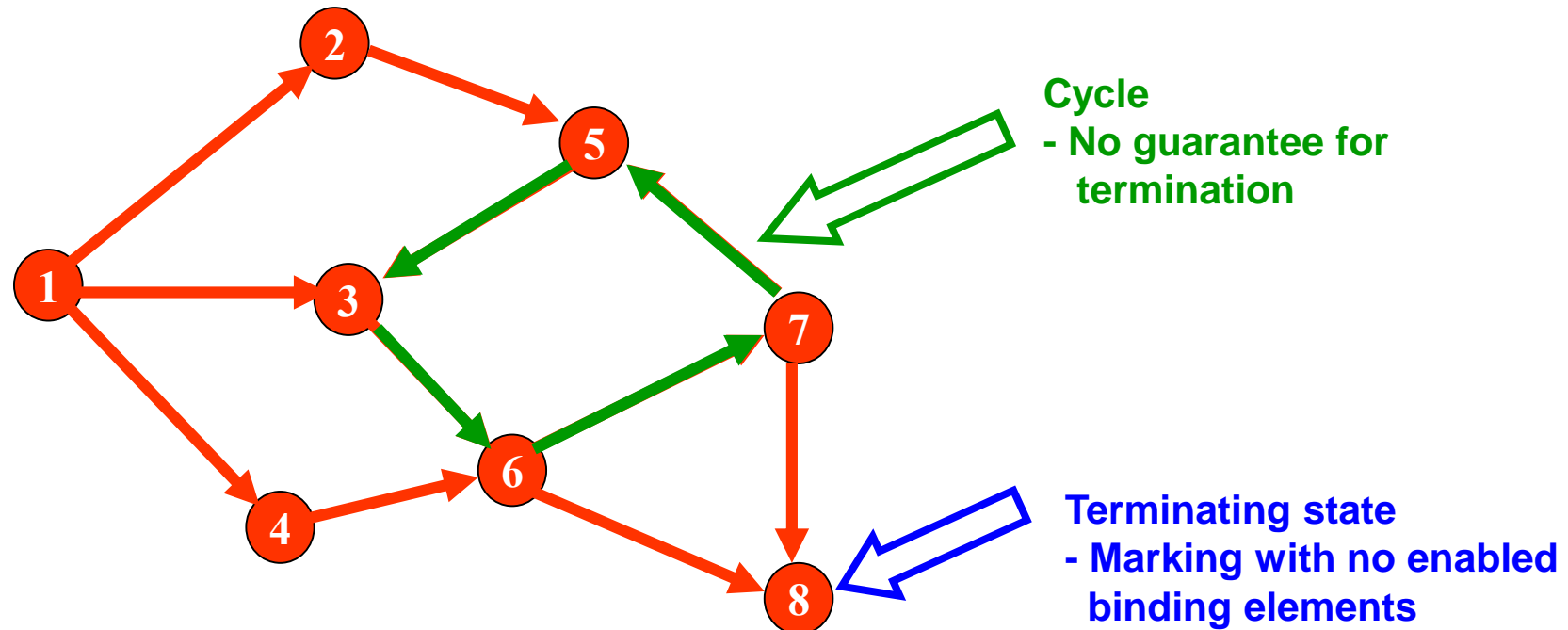
- Executable models can be automatically analysed by software tools



- Facilitates early **error-detection** and **verification** of components and systems

State spaces

- **The state space of a CPN is a directed graph with**
 - A **node** for each reachable **marking** (state)
 - An **arc** for each occurring **binding element**
- **State spaces can be used to investigate the behavioural properties of the a model**



State spaces

Definition 9.1. A **directed graph** with arc labels from a set L is a tuple $DG = (N, A)$, where:

1. N is a set of **nodes**.
2. $A \subseteq N \times L \times N$ is a set of **arcs**.



Definition 9.6. The **state space** of a Coloured Petri Net is a directed graph $SS = (N_{SS}, A_{SS})$ with arc labels from BE , where:

1. $N_{SS} = \mathcal{R}(M_0)$ is the set of **nodes**.
2. $A_{SS} = \{(M, (t, b), M') \in N_{SS} \times BE \times N_{SS} \mid M \xrightarrow{(t, b)} M'\}$ is the set of **arcs**.

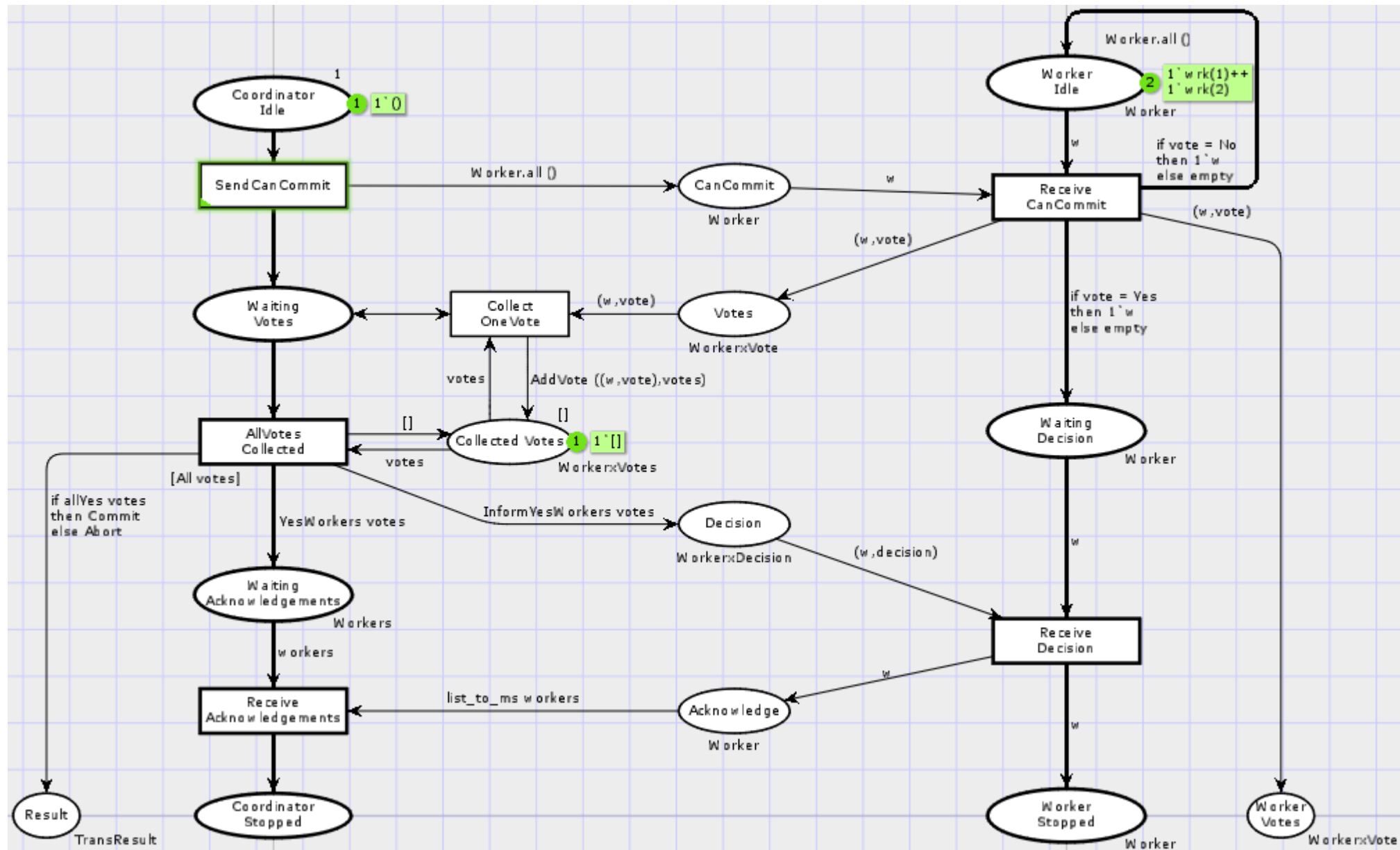
SS is **finite** if and only if N_{SS} and A_{SS} are finite.



CPN Tools state space demo

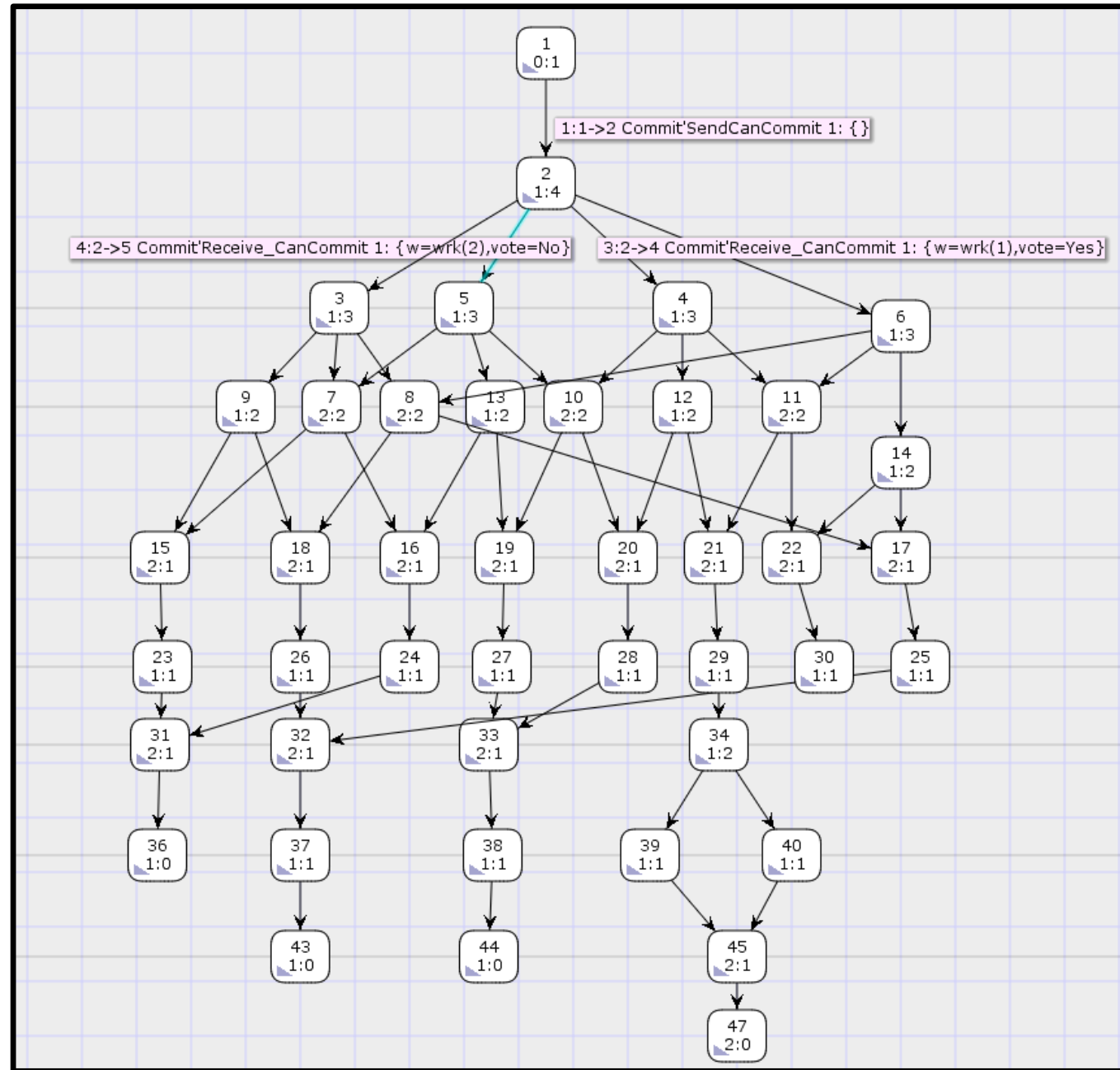
- **State space exploration of CPN models**
 - Variant of the two-phase commit protocol model where coordinator and workers do not return to their initial state
 - Calculation of state spaces
 - Stop options
 - Visualisation of fragments of state spaces





State Space Commit Protocol

$W = 2$



State space construction algorithm

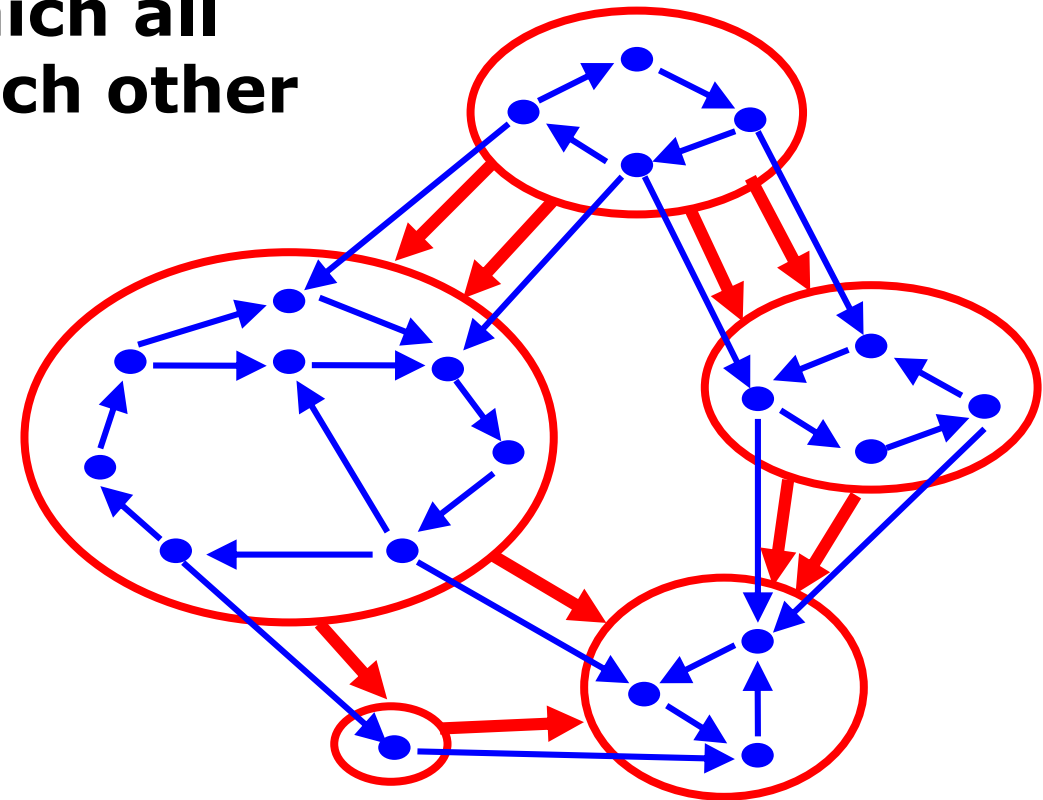
```
1: NODES  $\leftarrow \{M_0\}$ 
2: UNPROCESSED  $\leftarrow \{M_0\}$ 
3: ARCS  $\leftarrow \emptyset$ 
4: while UNPROCESSED  $\neq \emptyset$  do
5:   Select a marking  $M$  in UNPROCESSED
6:   UNPROCESSED  $\leftarrow$  UNPROCESSED  $- \{M\}$ 
7:   for all binding elements  $(t, b)$  such that  $M \xrightarrow{(t,b)}$  do
8:     Calculate  $M'$  such that  $M \xrightarrow{(t,b)} M'$ 
9:     ARCS  $\leftarrow$  ARCS  $\cup \{(M, (t, b), M')\}$ 
10:    if  $M' \notin$  NODES then
11:      NODES  $\leftarrow$  NODES  $\cup \{M'\}$ 
12:      UNPROCESSED  $\leftarrow$  UNPROCESSED  $\cup \{M'\}$ 
13:    end if
14:  end for
15: end while
```

Strongly Connected Components

- A **strongly connected component (SCC)** is a maximal subgraph in which all nodes are reachable from each other

- **SCC-graph contains**

- A **node** for each SCC
 - An **arc** from S_i to S_j for each state space arc from a node $n_i \in S_i$ to a node $n_j \in S_j$ ($i \neq j$)
- The SCCs are **mutually disjoint**
- Each node is in **exactly one SCC**
- The SCC-graph is **acyclic**

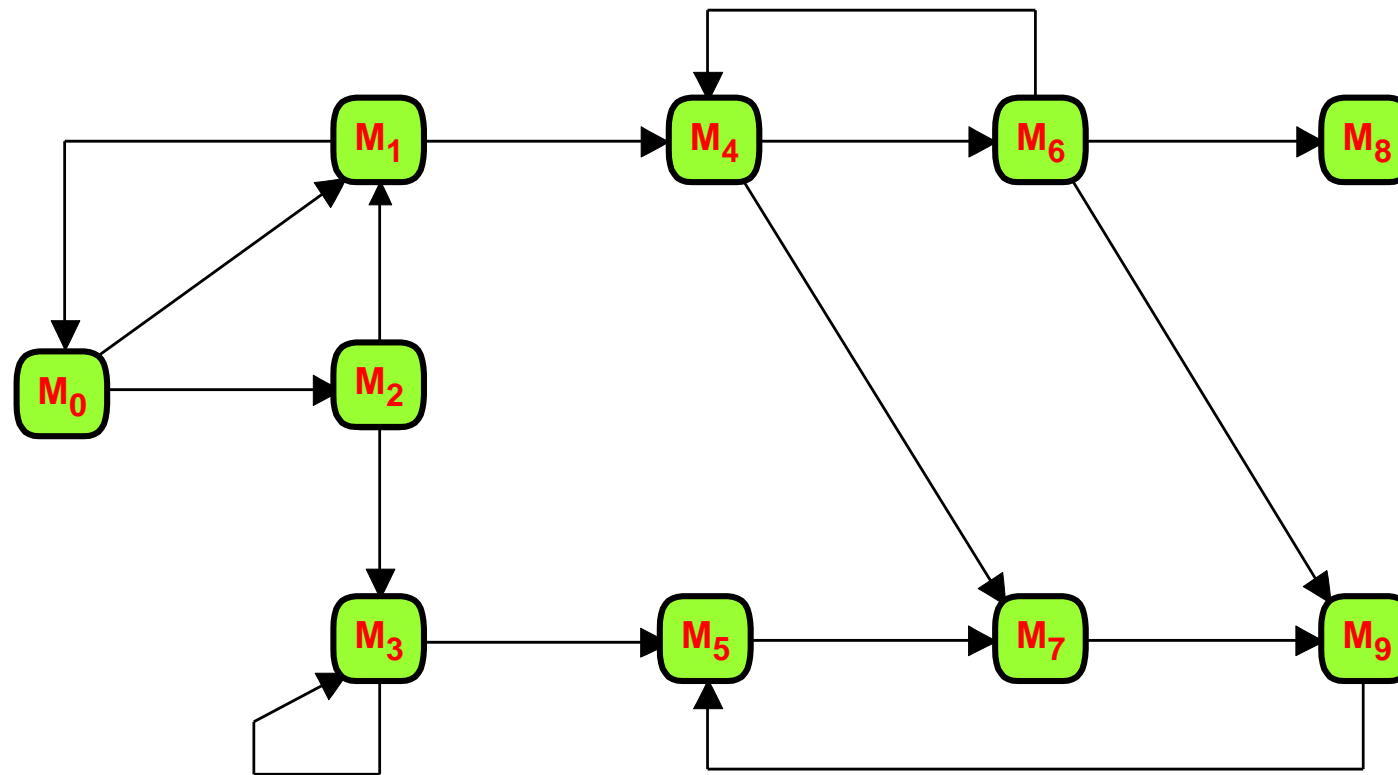


Trivial SCC
(one node and no arcs)

Terminal SCC
(no outgoing arcs)

Example: State space

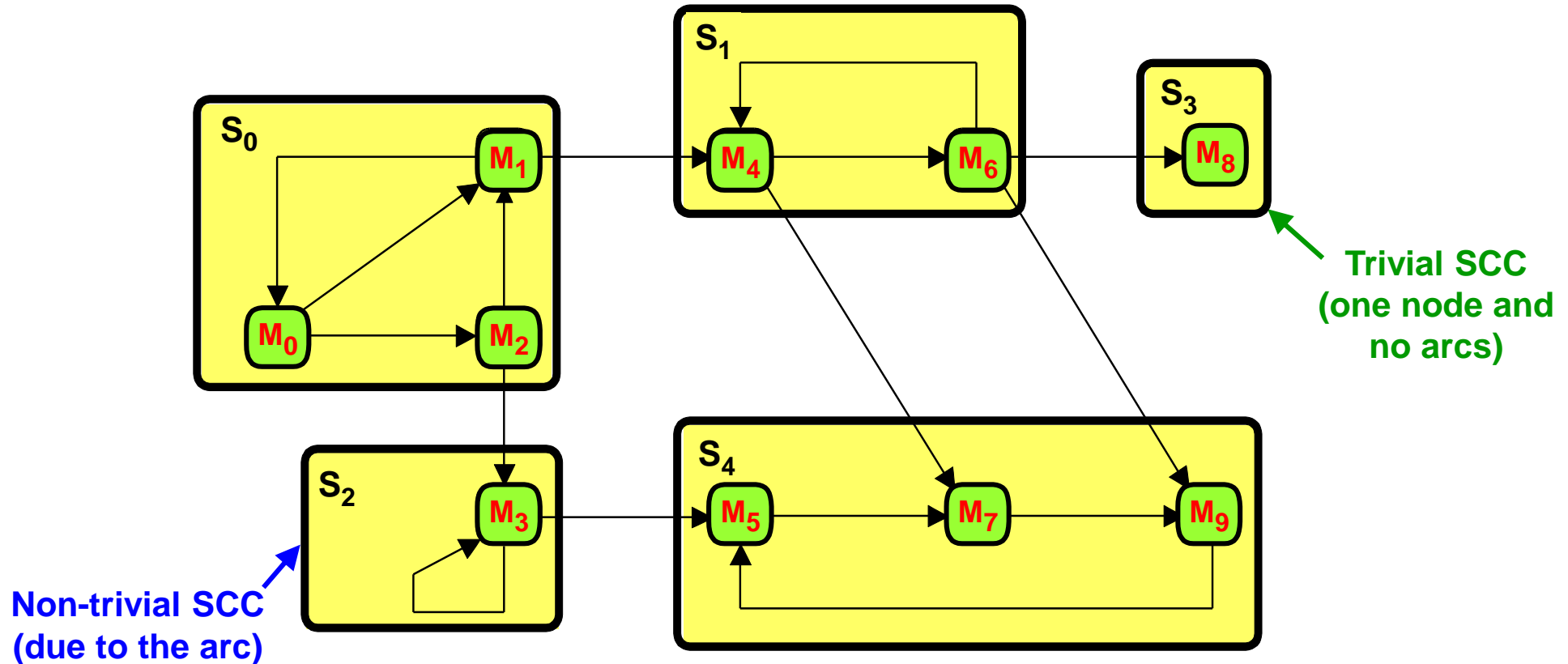
- 10 nodes and 16 arcs



Strongly connected components?

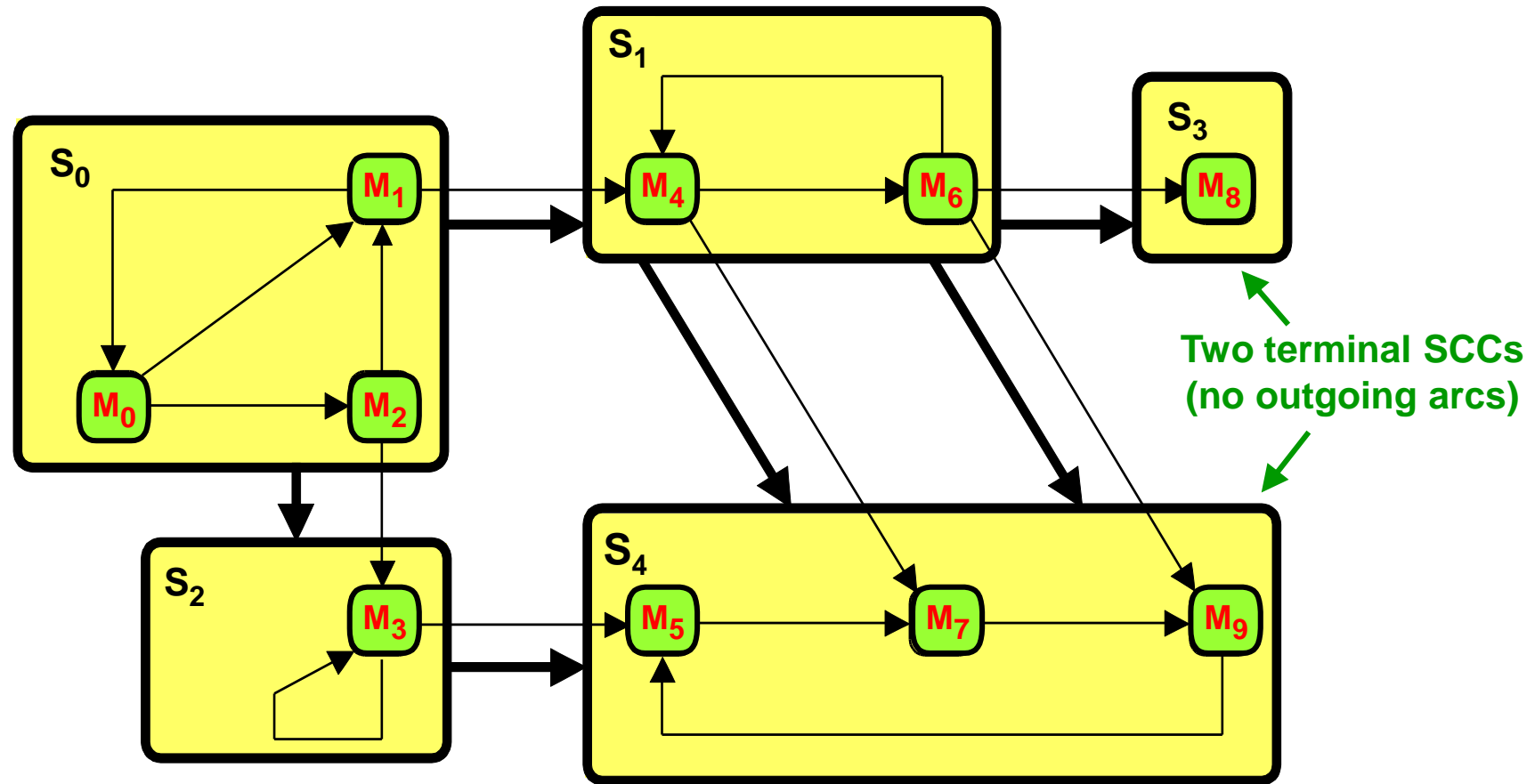
SCC Example

- Five SCCs



SCC graph

- 5 nodes (SCCs) and 6 arcs



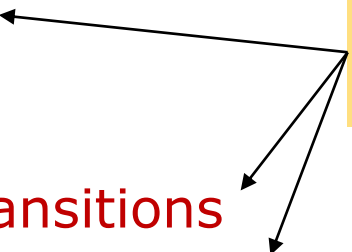
State space analysis

- **State spaces may be very large and we need computer tools to construct and analyse them**
- **Analysis of the state space typically starts with the generation of the **state space report****
 - Textual report that can be generated fully automatically
 - Contains information about standard behavioural properties of the CPN model
 - Useful for locating errors or increase confidence in the correctness of the system

State space report - content

- **The state space report contains information about **standard behavioural properties****
 - **Size** of the state space and the time used to generate it
 - **Bounds** for the number of tokens on each place and information about the possible token colours
 - **Home markings**
 - **Dead markings**
 - **Dead and live transitions**
 - **Fairness properties** for transitions
- **The state space report can be generated for any CPN models with a finite state space**

**Strongly connected components
used for checking these properties**



CPN Tools demo

- **State space analysis of CPN models**
 - Generation of the strongly connected components
 - Generation and inspection of the state space report



Size and exploration time

State Space Statistics

State Space

Nodes: 23,497

Arcs: 52,192

Secs: 76

Status: Full

Scc Graph

Nodes: 23,497

Arcs: 52,192

Secs: 1

- The state space has status **Full**: it **contains all reachable markings**
- The SCC graph and the state space has the same number of nodes and arcs: **no cycles**

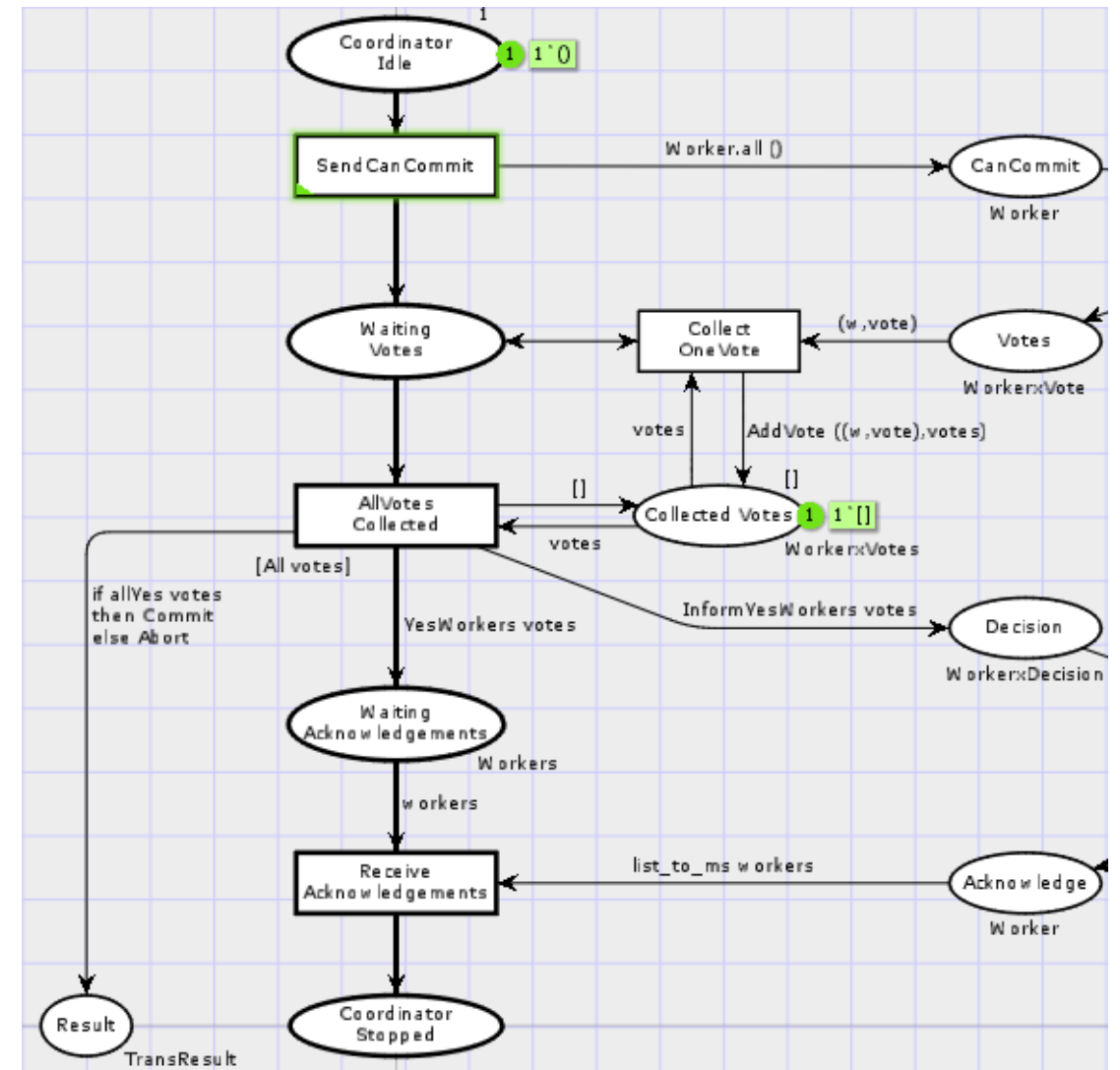
Integer bounds

- The integer bounds considers the **number of tokens** on a place
 - The **best upper integer bound** for a place is the **maximal number of tokens** on the place in a reachable marking
 - The **best lower integer bound** for a place is the **minimal number of tokens** on the place in a reachable marking
- Places with an upper integer bound are **bounded**
- Places with no upper integer bound are **unbounded**

Integer bounds - Coordinator

Best Integers Bounds

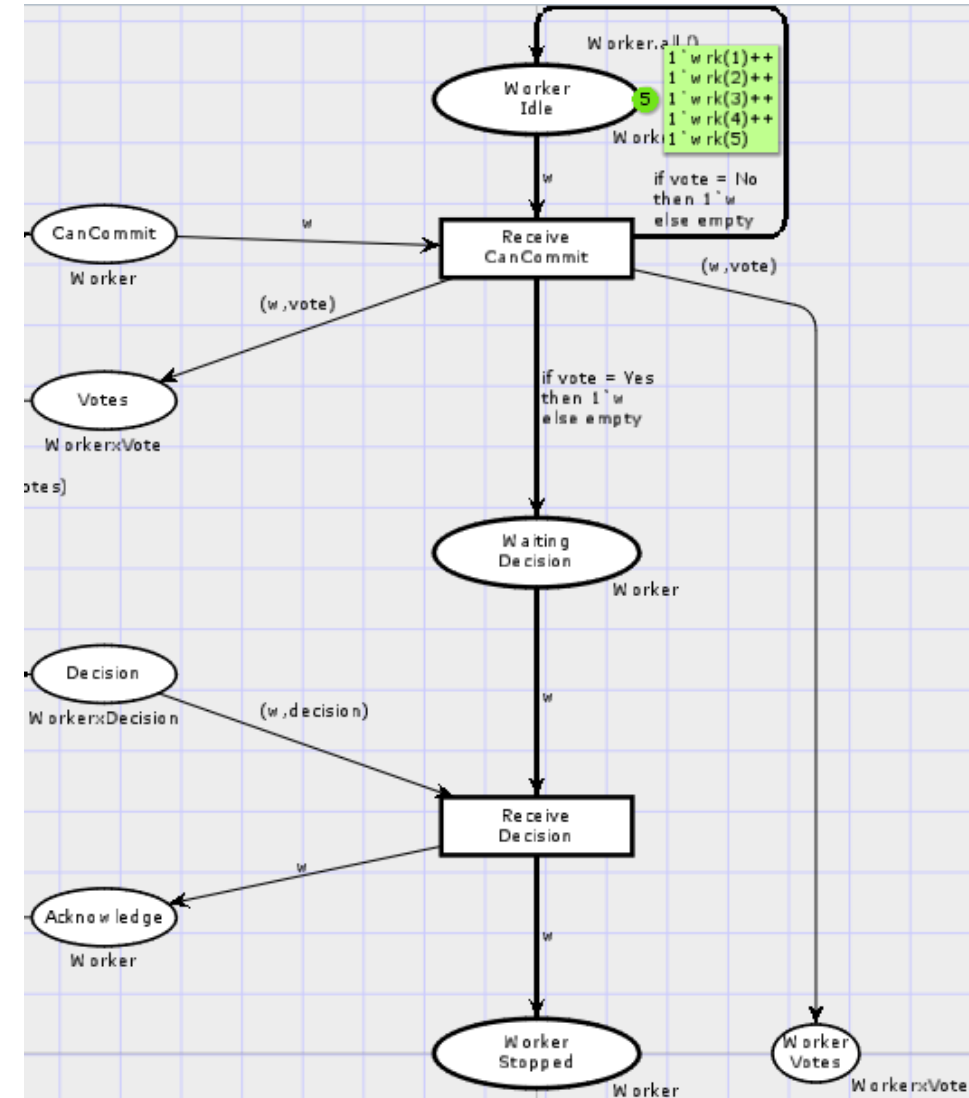
	Upper	Lower
CoordinatorIdle	1	0
CanCommit	5	0
WaitingVotes	1	0
CollectedVotes	1	1
Decision	5	0
Waiting Acknowledgements	1	0
CoordinatorStopped	1	0
Result	1	0



Integer bounds - Workers

Best Integers Bounds

	Upper	Lower
WorkerIdle	5	0
Votes	5	0
WaitingDecision	5	0
Acknowledge	5	0
WorkerStopped	5	0
WaitingDecision	5	0



Definition - Integer bounds

Definition 9.9. Let a place $p \in P$ and a non-negative integer $n \in \mathbb{N}$ be given.

1. n is an **upper integer bound** for p if and only if

$$\forall M \in \mathcal{R}(M_0) : |M(p)| \leq n$$

2. n is a **lower integer bound** for p if and only if

$$\forall M \in \mathcal{R}(M_0) : |M(p)| \geq n$$

3. p is **bounded** if and only if an upper integer bound for p exists. Otherwise, p is **unbounded**.

□

Multiset bounds

- Integer bounds considers the number of tokens
 - ignores the token colours
- Multiset bounds considers the possible token colours
- The best upper multiset bound for a place is a multiset over the colour set of the place
 - the coefficient for a colour c is the maximal number of occurrences of tokens with colour c in a reachable marking
- The best lower multiset bound for a place is a multiset over the colour set of the place
 - the coefficient for a colour c is the minimal number of occurrences of tokens with colour c in a reachable marking

Upper multiset - Coordinator

Best Upper Multiset Bounds

CoordinatorIdle 1`()

CanCommit 1`wrk(1) ++ 1`wrk(2) ++ 1`wrk(3) ++
 1`wrk(4) ++ 1`wrk(5)

Decision 1`(wrk(1),abort) ++ 1`(wrk(1),commit) ++
 1`(wrk(2),abort) ++ 1`(wrk(2),commit) ++
 1`(wrk(3),abort) ++ 1`(wrk(3),commit) ++
 1`(wrk(4),abort) ++ 1`(wrk(4),commit) ++
 1`(wrk(5),abort) ++ 1`(wrk(5),commit)

CoordinatorStopped 1`()

Result 1`Abort ++ 1`Commit

Upper Multiset - Workers

Best Upper Multiset Bounds

WorkersIdle, WaitingDecision, WorkerStopped, Acknowledge

1`wrk(1) ++ 1`wrk(2) ++ 1`wrk(3) ++ 1`wrk(4) ++ 1`wrk(5)

Votes, WorkerVotes

1` (wrk(1),Yes) ++ 1` (wrk(1),No) ++

1` (wrk(2),Yes) ++ 1` (wrk(2),No) ++

1` (wrk(3),Yes) ++ 1` (wrk(3),No) ++

1` (wrk(4),Yes) ++ 1` (wrk(4),No) ++

1` (wrk(5),Yes) ++ 1` (wrk(5),No)

Lower multiset bounds

Best Lower Multiset Bounds

CoordinatorIdle	empty	WorkerIdle	empty
CanCommit	empty	Votes	empty
WaitingVotes	empty	WaitingDecision	empty
CollectedVotes	empty	Acknowledge	empty
Decision	empty	WorkerVotes	empty
WaitingAcknowledgements	empty	WorkerStopped	empty
CoordinatorStopped	empty		
Result	empty		

- All are equal to the **empty multi-set**: all minimal coefficients are zero – no token is always present

Definition: multi-set bounds

Definition 9.11. Let a place $p \in P$ and a multiset $m \in C(p)_{MS}$ be given.

1. m is an **upper multiset bound** for p if and only if

$$\forall M \in \mathcal{R}(M_0) : M(p) \ll = m$$

2. m is a **lower multiset bound** for p if and only if

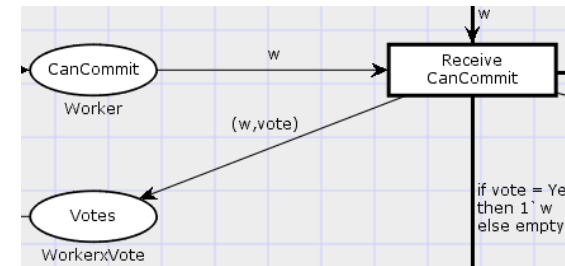
$$\forall M \in \mathcal{R}(M_0) : M(p) \gg = m$$



Integer and Multiset bounds

- The two kinds of bounds supplement each other and they provide different information
- Integer bounds

Votes	5	0
-------	---	---



Tells us that Votes has **at most five tokens** and **as little as zero tokens** - but gives us no information about the token colours

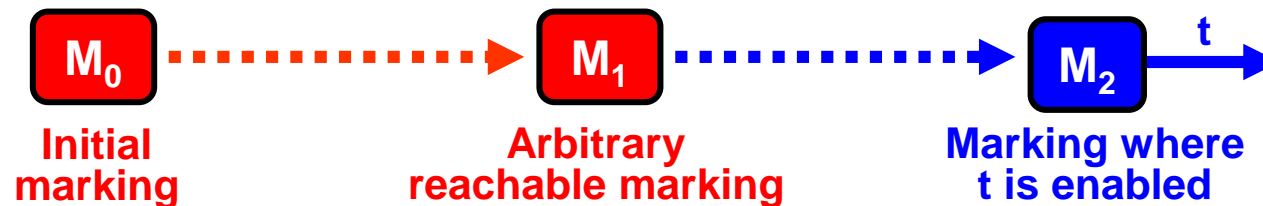
- Multi-set bounds

```
1` (wrk (1) , Yes) ++ 1` (wrk (1) , No) ++ 1` (wrk (2) , Yes) ++ 1` (wrk (2) , No) ++
1` (wrk (3) , Yes) ++ 1` (wrk (3) , No) ++ 1` (wrk (4) , Yes) ++ 1` (wrk (4) , No) ++
1` (wrk (5) , Yes) ++ 1` (wrk (5) , No)
```

Tells us that Votes can have **ten different tokens** and as little as zero of each - but not how many can be present simultaneously

Liveness properties

- A **marking M is dead** if M has no enabled transition bindings
- A **transition t is dead** if t never can occur
 - is disabled in all reachable markings
- A **transition t is live** if we from any reachable marking can reach a state where t is enabled



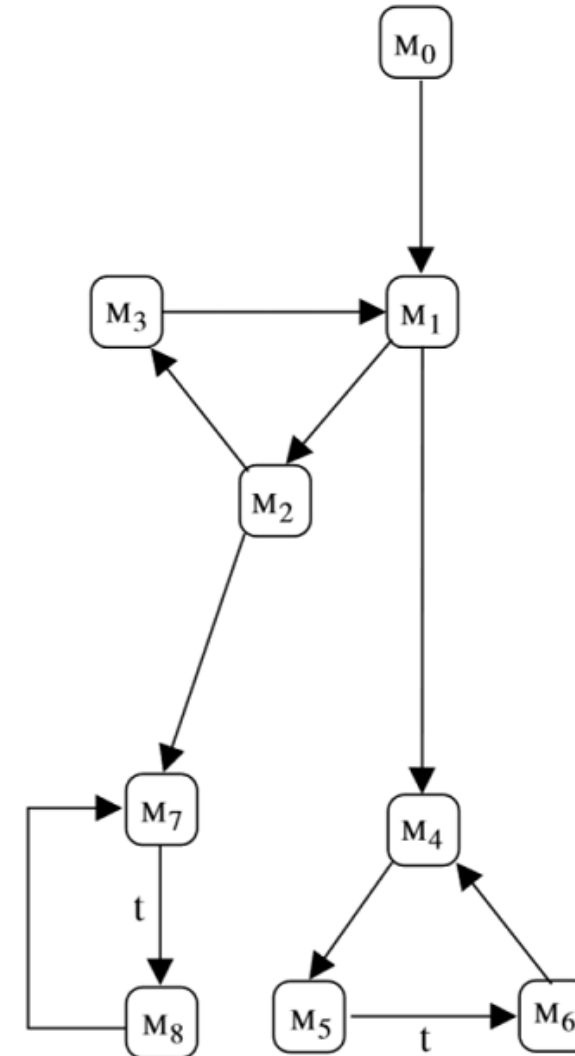
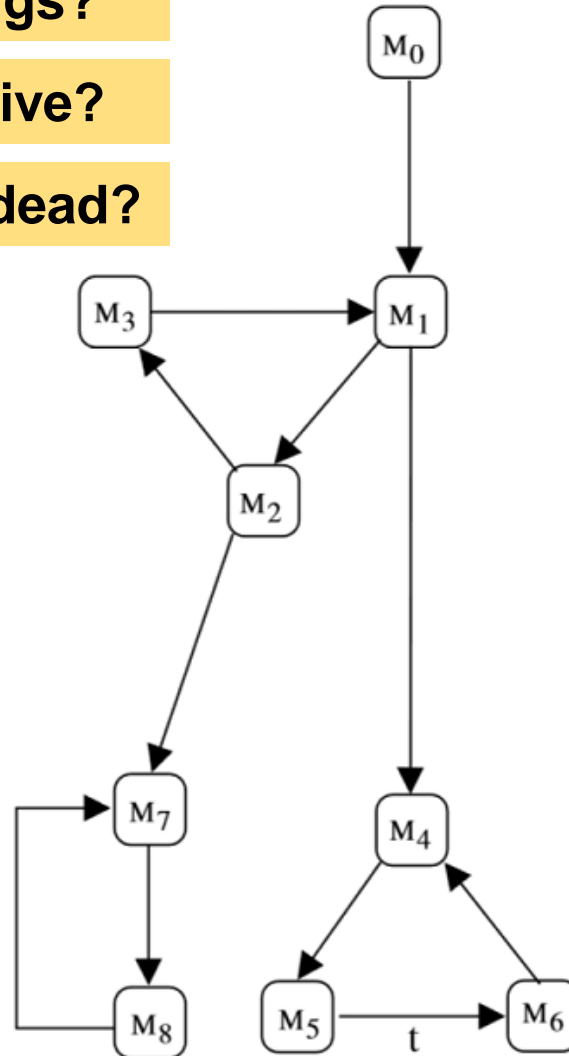
- Liveness tells that it is **possible** for t to occur
- **No guarantee** that this will happen
- It is possible to be non-dead without being live

Liveness Properties

Dead markings?

Transition t live?

Transition t dead?



Definition - Liveness properties

Definition 9.19. Let a transition $t \in T$ and a marking M be given.

1. M is a **dead marking** if and only if

$$\forall t \in T : \neg (M \xrightarrow{t})$$

2. t is **dead in** M_0 if and only if

$$\forall M \in \mathcal{R}(M_0) : \neg (M \xrightarrow{t})$$

3. t is **live in** M_0 if and only if

$$\forall M \in \mathcal{R}(M_0) \exists M' \in \mathcal{R}(M) : M' \xrightarrow{t}$$

□

Liveness properties

Liveness Properties

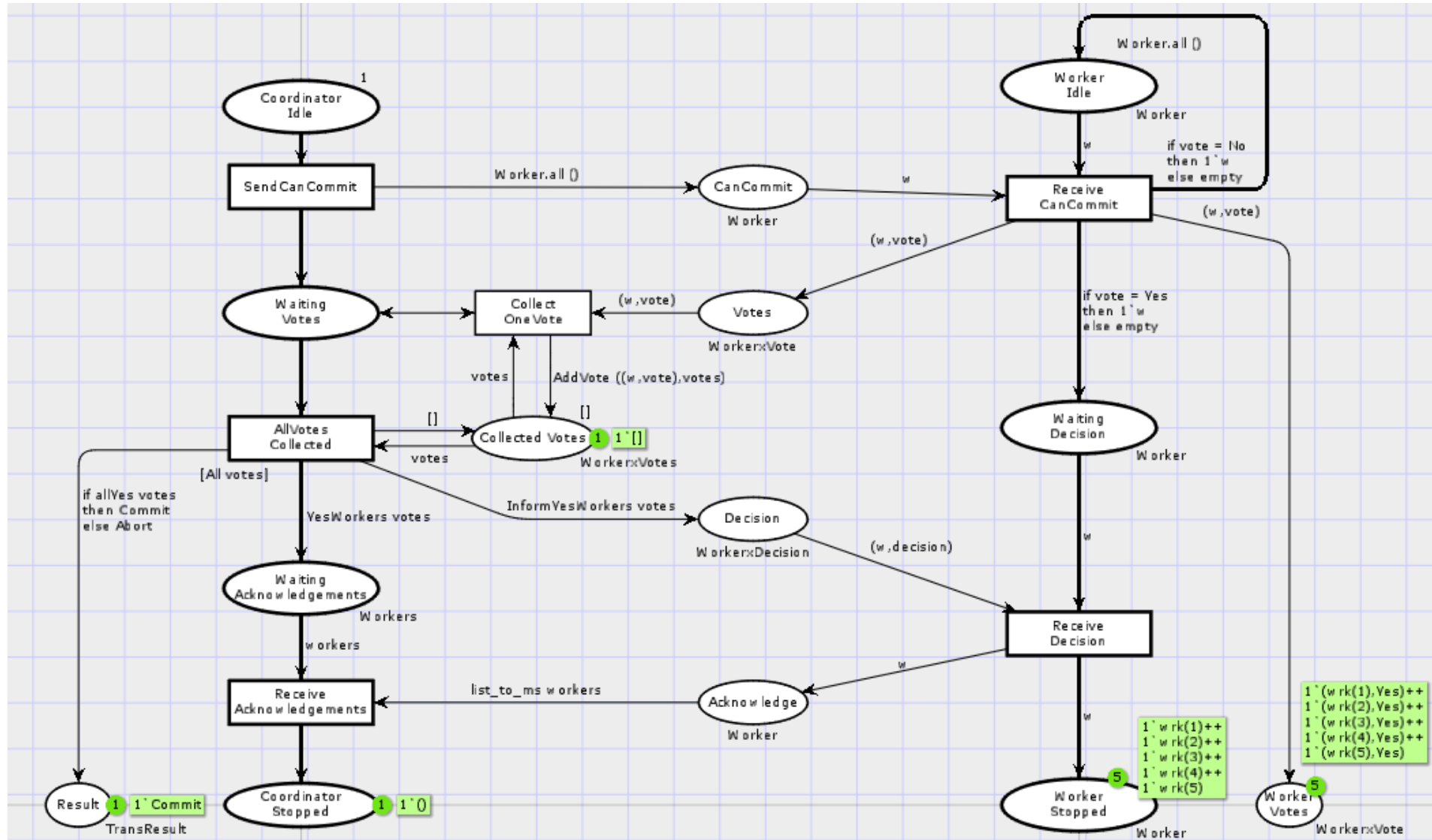
Dead Markings: 32 [23497, 23376, 23375, 23374, 23373, ...]

Dead Transitions: None

Live Transitions: None

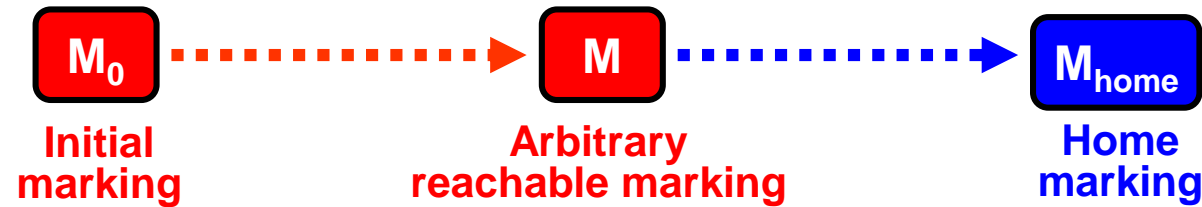
- **There are 32 dead markings represented by the nodes numbered: 23497, 23376, 23375,...**
- **There are no dead transitions**
 - All transitions have the possibility of becoming enabled
- **There are no live transitions**
 - Consequence of the existence of a dead markings

State 23497



Home properties

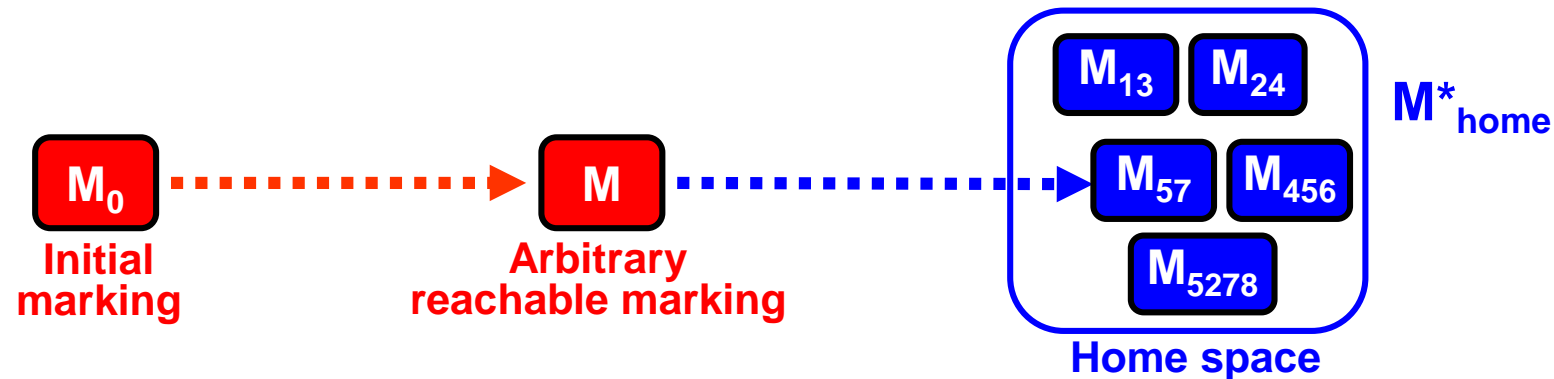
- A home marking is a marking M_{home} which can be reached from any reachable marking



- Impossible to have an occurrence sequence which cannot be extended to reach M_{home}
 - The home property tells that it is possible to reach M_{home}
 - No guarantee that this will happen

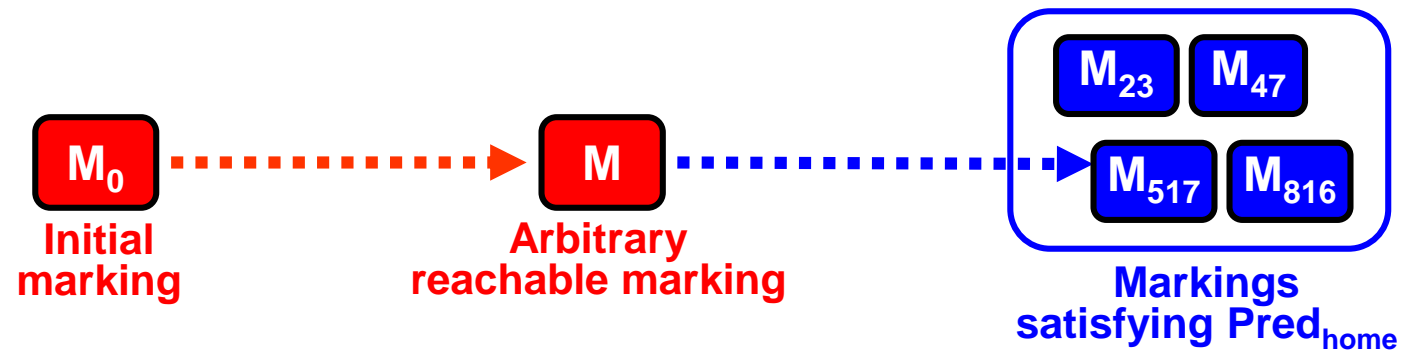
Home space

- A **home space** is a set of markings M^*_{home} such that at least one marking in M^*_{home} can be reached from any reachable marking



Home predicate

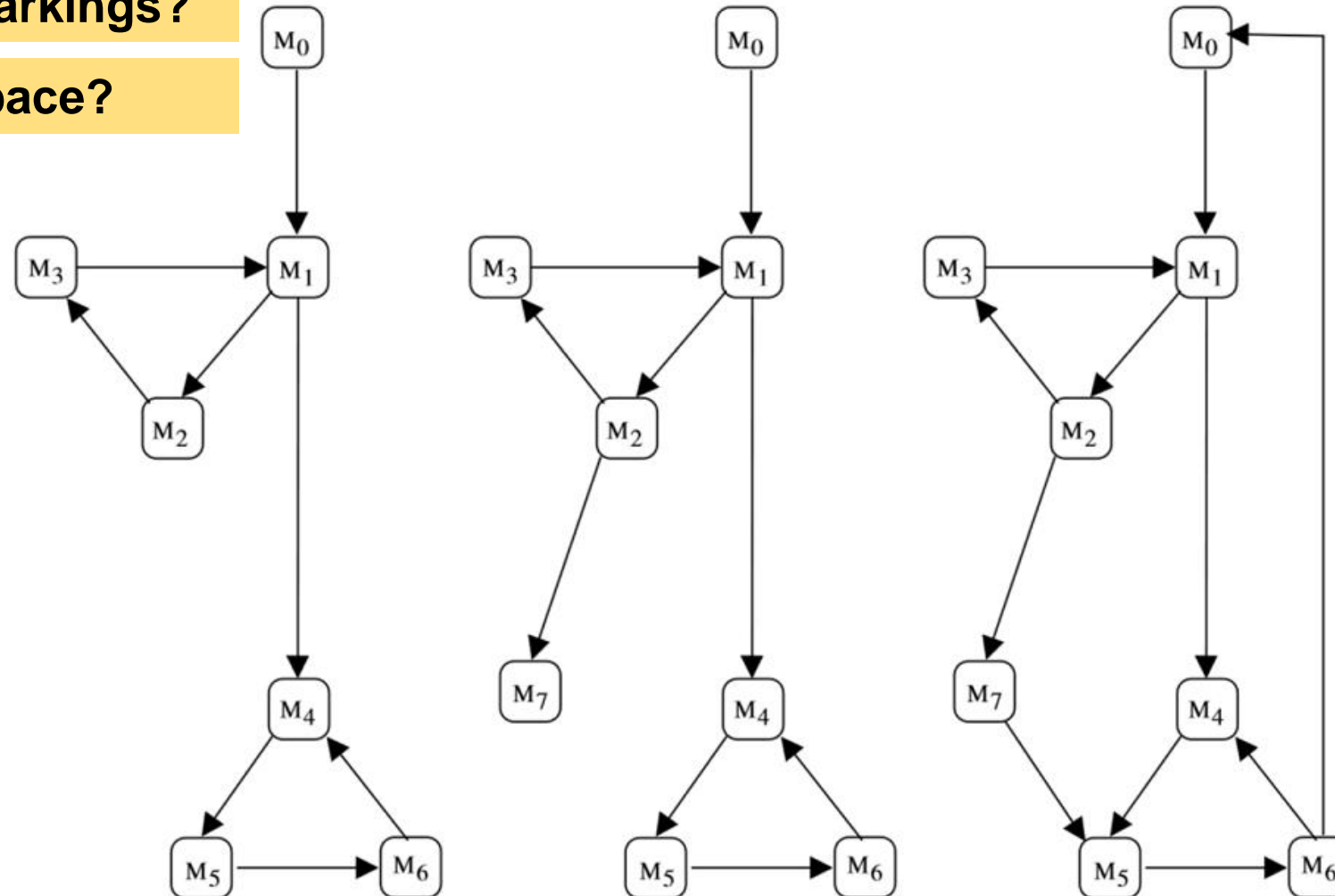
- A **home predicate** is a predicate on markings $\text{Pred}_{\text{home}}$ such that at least one marking satisfying $\text{Pred}_{\text{home}}$ can be reached from any reachable marking



Home properties

Home markings?

Home space?



Definition: home properties

Definition 9.17. Let M_{home} be a marking and M_{home}^* a set of markings.

1. M_{home} is a **home marking** if and only if

$$\forall M \in \mathcal{R}(M_0) : M_{home} \in \mathcal{R}(M)$$

2. M_{home}^* is a **home space** if and only if

$$\forall M \in \mathcal{R}(M_0) \exists M' \in \mathcal{R}(M) : M' \in M_{home}^*$$

3. A predicate ϕ on markings is a **home predicate** if and only if

$$\forall M \in \mathcal{R}(M_0) \exists M' \in \mathcal{R}(M) : \phi(M')$$



Home markings

- **There is are no home markings in the two-phase commit protocol CPN model**

Home Properties

Home Markings: None

- **A consequence of having more than one (32) dead markings**

Query functions

- **The state space report contains information about **standard behavioural properties****
 - Useful for validating the basic operation of model
 - Modelling errors are often evident from the state space report
- **Non-standard behavioural properties can also be investigated by means of queries written in the Standard ML language**
 - provide arguments to a predefined query function – e.g. to check whether a set of markings constitute a home space
 - write your own query functions using the Standard ML programming language

What does it mean for the two-phase commit protocol model to be correct?



Example – can we commit?

- Want to check whether one of the dead markings corresponds to a commit state
- Predicate checking the marking of the Result place in the coordinator

```
fun hasCommit n = Mark.Commit'Result 1 n == 1`Commit
```

- Check whether we have a commit in one of the dead markings

```
List.exists hasCommit (ListDeadMarkings ())
```

Consistent termination states

- The protocol commits if and only if all workers votes yes

```
fun correctTermination n =  
  let  
    val votes = Mark.Commit'Worker_Votes 1 n  
    val votecount = size votes  
    val yescount = List.length (yesVotes votes)  
    val result = ms_to_col (Mark.Commit'Result 1 n)  
  in  
    (votecount = W) andalso  
    (  
      ((result = Commit) andalso (yescount = W)) orelse  
      ((result = Abort) andalso (yescount < W))  
    )  
  end
```

```
List.all correctTermination (ListDeadMarkings ())
```

Ability to correctly terminate?

- Established that all terminating states of the protocol are consistent with the commit criteria
- Does the dead markings constitute a home space?

```
HomeSpace (ListDeadMarkings ())
```

- The SCC-graph and the state space are equal for the two-phase commit protocol
 - One of the dead markings will always be reached
- **Conclusion:** the protocol eventually terminates in a state consistent with the commit criteria

Error detection and counter examples

Property violations

- If a property is violated then state space methods can provide error-traces
- **Example:** two-phase commit with a wrong implementation of the allYes predicate

```
fun allYes votes = (List.length (yesVotes votes) = W-1)
```

- Results of running the queries

```
List.all correctTermination (ListDeadMarkings ())
```

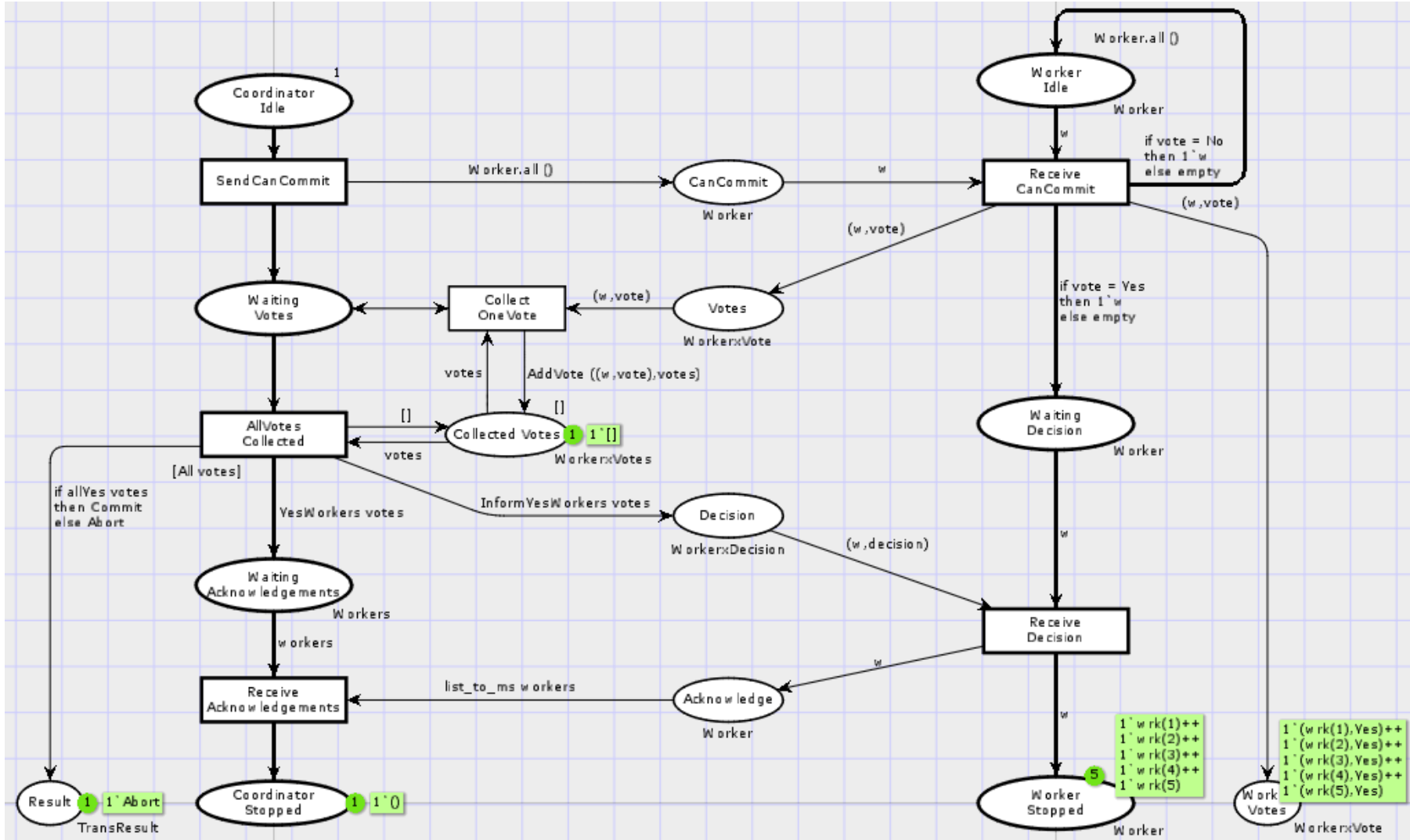
```
val it = false : bool
```

- Find a dead marking with inconsistent termination

```
List.find (not o correctTermination) (ListDeadMarkings ())
```

```
val it = SOME 23497 : Node option
```

State 23497



Counter example generation

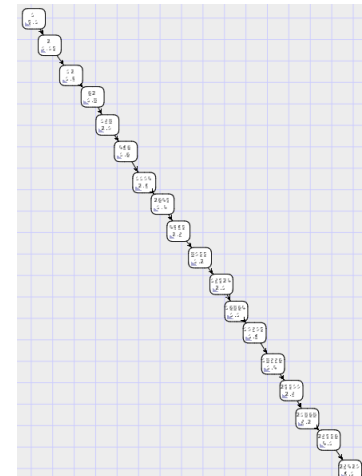
- **Find the arcs in a path from the initial state (node 1) to state 23497**

```
val path = ArcsInPath (1,23497)
```

```
val path = [1,11,93, 591, ... ] Arc list
```

- **Obtain the sequence of transition bindings corresponding to the arcs in the path**

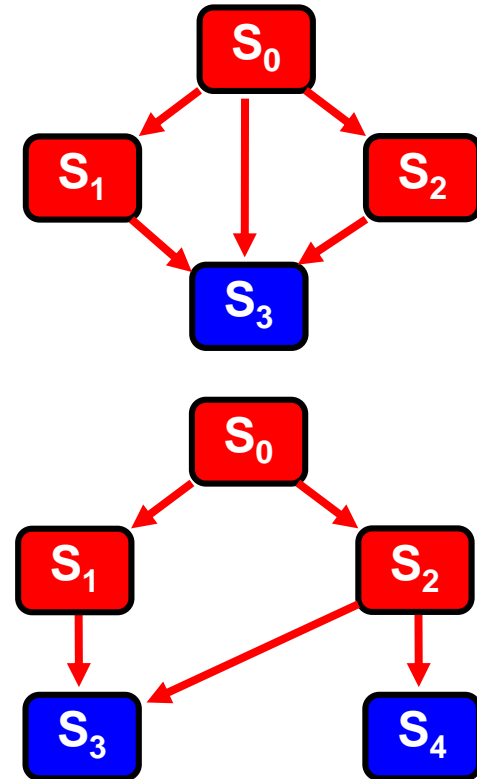
```
val errortrace = List.map ArcToBE path
```



Checking Home Properties using Strongly Connected Components

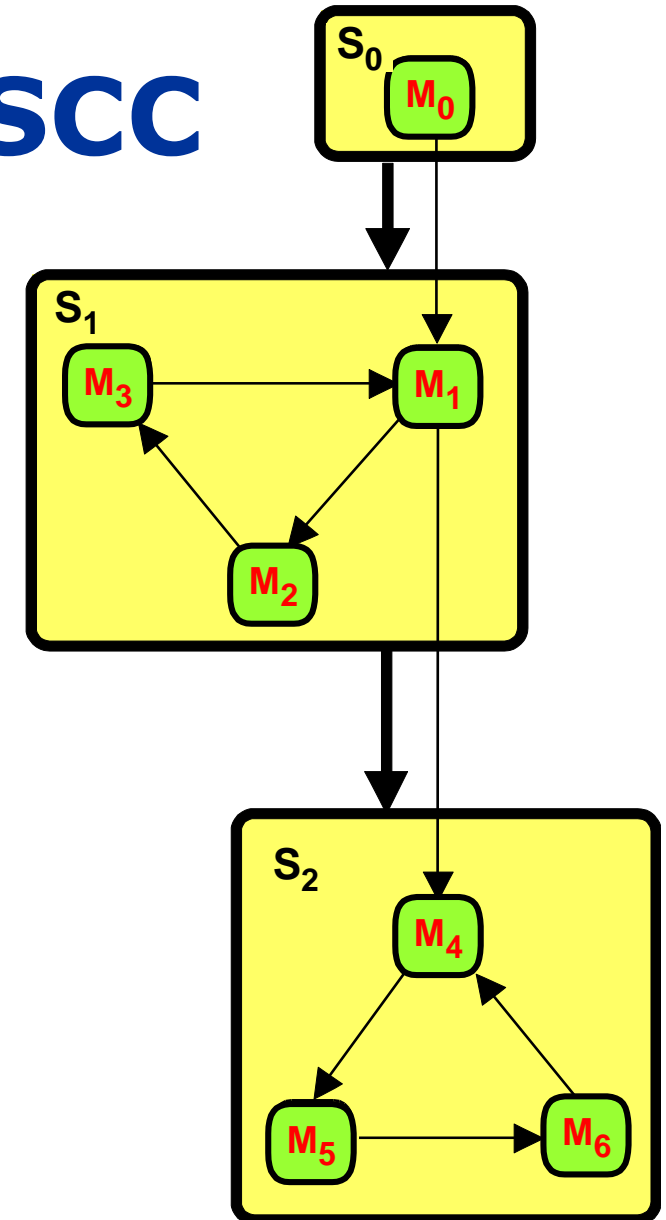
Home Markings and SCCs

- The existence of home markings can be determined from the number of terminal SCCs
- **Only one terminal SCC**
 - All markings in the terminal SCC are home markings
 - No other markings are home markings
- **More than one terminal SCC**
 - No home markings



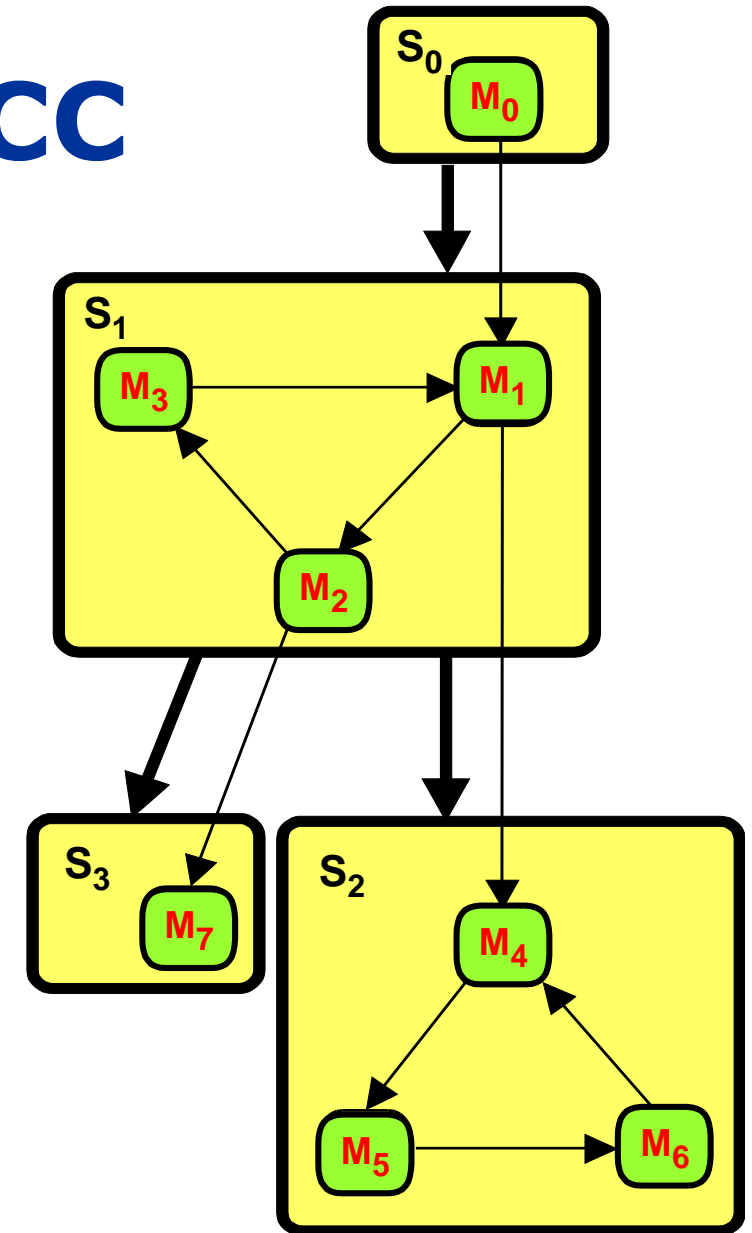
Example: Single Terminal SCC

- All markings in the terminal SCC S_2 are home markings
- No other markings are home markings



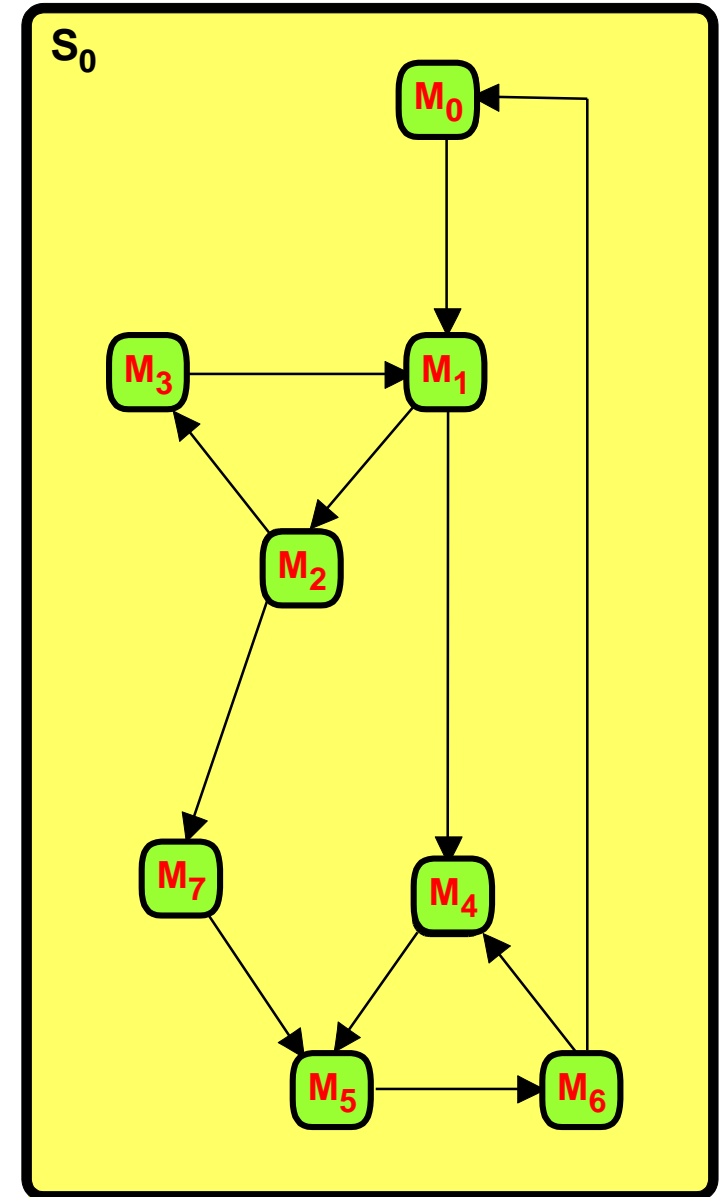
More than one terminal SCC

- No home markings
- When one of the terminal SCCs S_2 and S_3 has been reached, it is impossible to leave it again



Single SCC

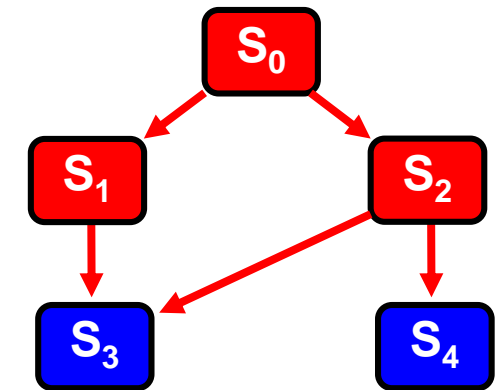
- All reachable markings are home markings
- They are mutually reachable from each other



Home Spaces and SCCs

- **Home spaces can also be determined from the terminal components in the SCC graph**

- A set of markings is a home space if and only if it contains a node from each terminal SCC
- Home spaces must have at least as many elements as there are terminal SCCs

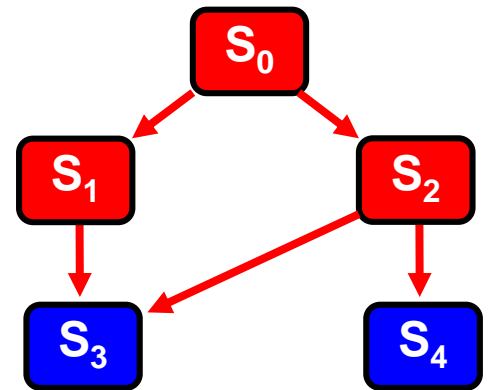


- A system may have home spaces without having home markings
- Each home marking is a home space with only one element

Checking Liveness Properties using Strongly Connected Components

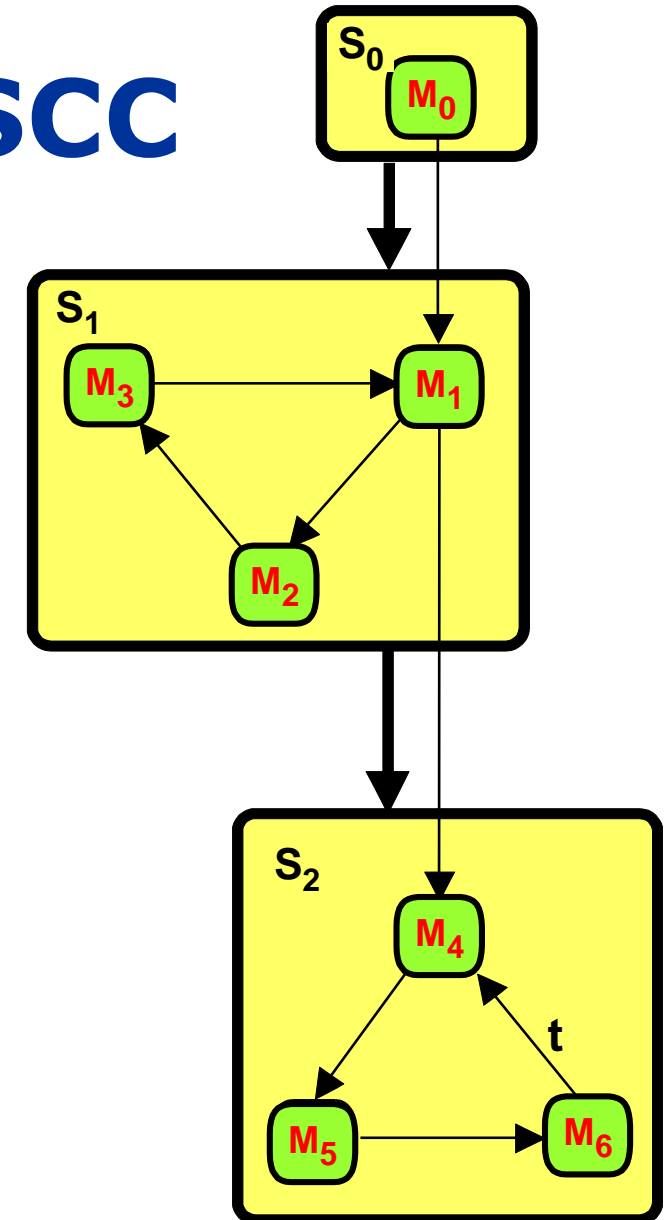
Liveness Properties and SCCs

- **Liveness properties of transitions can be determined from the SCC graph**
- A transition/binding element is live if and only if it appears on at least one arc in each terminal SCC



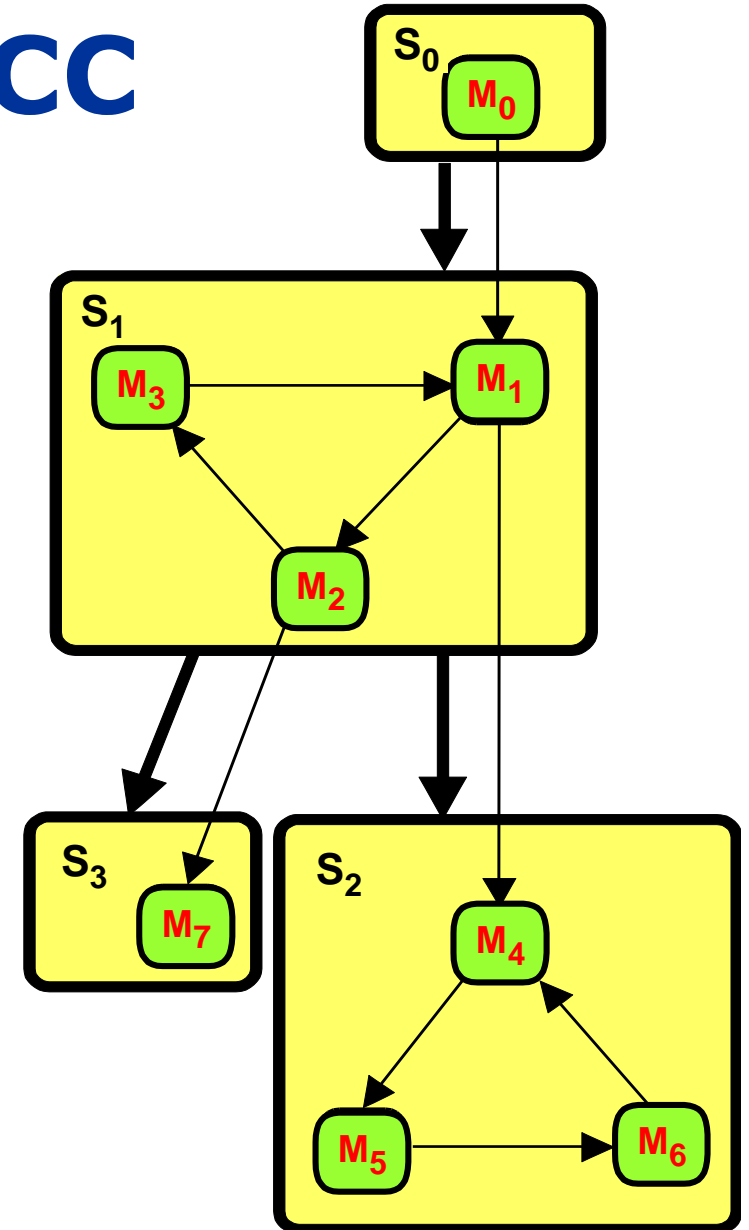
Example: Single Terminal SCC

- A transition is live if it appears on an arc in the terminal SCC S_2



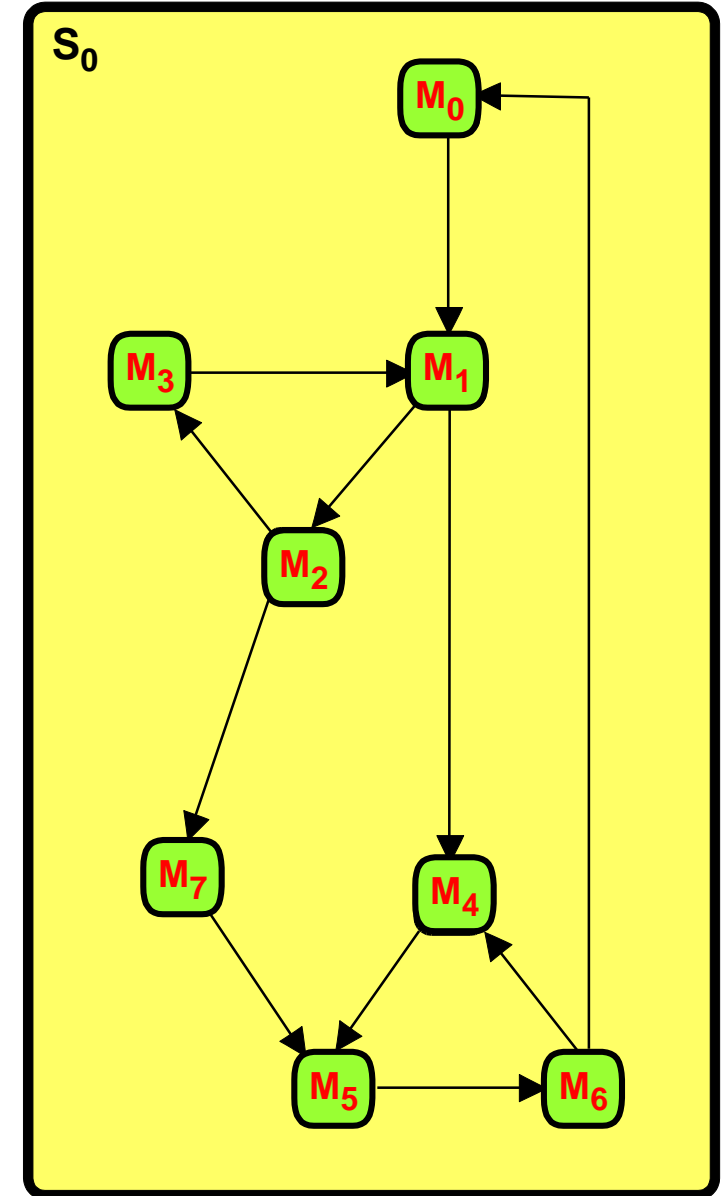
More than one terminal SCC

- A transition is live if it appears on an arc in each terminal SCC
- No live transitions
- S_3 is terminal and trivial
- M_7 is a dead marking



Single SCC

- A transition is live if it appears on an arc in the SCC
- In this case we have that a transition is live if and only if it is non-dead



State Spaces – Wrap up

System configurations

- **Basic state spaces analyses a system for a particular configuration of its parameters**
 - in practice it is often sufficient to consider a few rather small configurations
 - we cannot be totally sure that larger configurations will have the same properties
- **As system parameters increase the size of the state space often increases exponentially**
- **This phenomenon is also known as the state explosion problem**
 - one of the most severe limitations of state space methods

Is it worthwhile?

- **It takes many hours (or days) to generate the state spaces and verify the desired properties**
 - it is fully automatic and hence requires little human work
- **A relatively **small investment** compared to**
 - the total number of resources used in a system development project
 - cost of implementing, deploying and correcting a system with errors that could have been detected in the design phase

Summary

- **State spaces are powerful and easy to use**
 - construction and analysis can be automated
 - the user do not need to know the underlying mathematics
- **The main drawback and limitation for practical use is the state explosion problem**
 - the present CPN state space tool can handle state spaces with up to several million states
 - for many systems this is not sufficient and efficient state space methods is an active area of research
 - an abundance of sophisticated techniques exists for alleviating the state explosion problem