Coloured Petri Nets

Modelling and Validation of Concurrent Systems

Chapter 7: State Spaces and Behavioural Properties

Kurt Jensen & Lars Michael Kristensen {kjensen,lmkristensen}@cs.au.dk





Coloured Petri Nets Department of Computer Science Kurt Jensen Lars M. Kristensen

Complex Behaviour

aarhus

JNIVERSITY

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



Verification

 Executable models can be automatically analysed by computer tools:



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



- A state space 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 CPN model.





Coloured Petri Nets Department of Computer Science

Simple protocol

- SendPacket can occur an unlimited number of times producing an unlimited number of tokens on place A.
- This means that the state space becomes infinite.





Coloured Petri Nets Department of Computer Science Kurt Jensen Lars M. Kristensen

Simple protocol for state space analysis

 We add a new place Limit, which limits the total number of tokens on the buffer places A, B, C, and D.

This makes the state space finite.





Coloured Petri Nets Department of Computer Science



- Construction of the state space starts with the processing of node 1 which represents the initial marking.
- Node 1 has one enabled binding element:

SP1 = (SendPacket, <n=1, d="COL">)

- This gives us one new arc and one new node 2.
- Node 2 has one copy of data packet 1 on place A.
- Node 1 is now marked as processed (thick border line).



Kurt Jensen Lars M. Kristensen

- Next we process node 2.
- It has three enabled binding elements:



SP1 = (SendPacket, <n=1, d="COL">)

TP1- = (TransmitPacket, <n=1, d="COL", success=false>)

- This gives us three new arcs and two new nodes 3 and 4.
- Node 3 has one copy of data packet 1 on place B.
- Node 4 has two copies of data packet 1 on place A.
- Node 2 is now marked as processed (thick border line).



- Next we choose one of the unprocessed nodes: 3.
- It has two enabled binding elements:

SP1 = (SendPacket, <n=1, d="COL">)

RP1 = (ReceivePacket, <n=1, d="COL", k=1, data = "">)



- This gives us two new arcs and two new nodes 5 and 6.
- Node 5 has one copy of acknowledgement 2 on place C.
- Node 6 has one copy of packet 1 on place A and another on place B.
- Node 3 is now marked as processed (thick border line).





- This gives us a three new arcs and one new node 7.
- Node 7 has three copies of data packet 1 on place A.
- Node 4 is now marked as processed (thick border line).





- This gives us three new arcs and three new nodes 8, 9 and 10.
- Node 8 is identical to the initial marking except that NextRec and Data Received have been changed.
- Node 5 is now marked as processed (thick border line).







Coloured Petri Nets Department of Computer Science Kurt Jensen Lars M. Kristensen

- We continue to process the nodes one by one.
- If the state space is finite construction terminates when all reachable markings have been processed.
- Otherwise, we continue forever - obtaining a larger and larger part of the state space.
- This partial state space is visualised using the drawing facilities of the CPN state space tool.





Coloured Petri Nets Department of Computer Science

Kurt Jensen Lars M. Kristensen

Directed path

- A directed path is an alternating sequence of nodes and arcs.
- Each directed path in the state space corresponds to an occurrence sequence where all steps contain a single binding element.
- A directed path in the state space corresponds to one possible execution (simulation) of the model.





Coloured Petri Nets Department of Computer Science

Strongly connected components

- A strongly connected component (SCC) is a maximal subgraph in which all nodes are reachable from each other.
- The SCCs are mutually disjoint.
- Each node is in exactly one SCC.
- 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∈S_i to a node n_j∈S_j (i≠j).
- The SCC graph is acyclic.





Coloured Petri Nets Department of Computer Science

State space (example)

10 nodes and 16 arcs.





Coloured Petri Nets Department of Computer Science Kurt Jensen Lars M. Kristensen

Strongly connected components

• 5 different SCCs.





Coloured Petri Nets Department of Computer Science

SCC graph

• 5 nodes and 6 arcs.





Coloured Petri Nets Department of Computer Science

State space construction and analysis

- State spaces may be very large and hence we need computer tools to construct and analyse them.
- Analysis of the state space starts with the generation of the state space report.
- This is done totally automatic.
- The report contains a lot of useful information about the behavioural properties of the CPN model.
- The report is excellent for locating errors or increase our confidence in the correctness of the system.



Coloured Petri Nets Department of Computer Science

State space report

- The state space report contains information about standard behavioural properties which make sense for <u>all</u> CPN models:
 - 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.



Demo of State Spaces in CPN Tools

http://cs.au.dk/~cpnbook/models/chapter7/7-2LimitProtocol.cpn



Coloured Petri Nets Department of Computer Science Kurt Jensen Lars M. Kristensen

State space report: size and time

State Space Statistics				
State Space		Scc Grap	Scc Graph	
Nodes:	13,215	Nodes:	5,013	
Arcs:	52,784	Arcs:	37,312	
Secs:	53	Secs:	2	
Status:	Full			

- State space contains more than 13.000 nodes and more than 52.000 arcs.
- The state space was constructed in less than one minute and it is full – i.e. contains all reachable markings.
- The SCC graph is smaller. Hence we have cycles.
- The SCC graph was constructed in 2 seconds.



Reachability properties

 The standard query function below checks whether marking M₁₇ is reachable from M₁
– i.e. whether there is a path from node 1 to node 17.

Reachable (1,17); true

 We can also check whether M₁ is reachable from M₁₇:

Reachable (17,1);





Coloured Petri Nets Department of Computer Science

Reachability properties (SCC)

- It is also possible (and more efficient) to check reachability from the SCC graph.
- Then we check whether there exists a path from the SCC containing the first marking to the SCC containing the second marking.

SccReachable (1,17);

SccReachable (17,1);



21



Coloured Petri Nets Department of Computer Science Kurt Jensen Lars M. Kristensen

24

Desired terminal marking

 The following predicate checks whether node n represents a marking in which all data packets have been successfully received.





Coloured Petri Nets Department of Computer Science

Reachability of desired terminal marking

The following query checks whether the desired terminal marking is reachable:



 It is also possible to find the node(s) which represent the desired terminal marking:





Coloured Petri Nets Department of Computer Science

State space report: reachability properties

- The state space report does <u>not</u> contain information about reachability properties.
- The specific markings which it is of interest to investigate is highly model dependent – and there are too many to investigate all pairs.
- The statistics in the state space report for the protocol shows that there are more than one SCC.
- This implies that not all nodes in the state space are mutually reachable – as demonstrated above using standard query functions.



Coloured Petri Nets Department of Computer Science

Integer bounds

- Integer bounds counts 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.
- 0 is always a lower integer bound, but it may not be the best.



State space report: integer bounds

Best Integers Bounds	Upper	Lower	
PacketsToSend	6	6	
DataReceived	1	1	
NextSend, NextRec	1	1	
A, B, C, D	3	0	
Limit	3	0	

- PacketsToSend has exactly 6 tokens in all reachable markings.
- DataReceived, NextSend and NextRec have exactly one token each in all reachable markings.
- The remaining five places have between 0 and 3 tokens each in all reachable markings.



Coloured Petri Nets Department of Computer Science

Kurt Jensen Lars M. Kristensen

More general integer bounds

- It is also possible to find integer bounds for a set of places.
- As an example, we might investigate how many tokens we have simultaneously on places A and B.





Coloured Petri Nets Department of Computer Science Kurt Jensen Lars M. Kristensen

More general integer bounds

- It is also possible to investigate integer bounds which consider only certain token colours and places.
- As an example, we will investigate the minimal and maximal number of tokens with the colour (1,"COL") that can simultaneously reside on the places A and B:





Coloured Petri Nets Department of Computer Science Kurt Jensen Lars M. Kristensen

Multiset bounds

- Integer bounds count the number of tokens ignoring the token colours.
- Multiset bounds provide information about 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.



State space report: upper multiset bounds

Best Upper Multiset Bounds		
	PacketsToSend	1'(1,"COL")++1'(2,"OUR")++1'(3,"ED ")++ 1'(4,"PET")++1'(5,"RI ")++1'(6,"NET")
	DataReceived	1"""++1"COL"++1"COLOUR"++1"COLOURED "++ 1"COLOURED PET"++1"COLOURED PETRI "++ 1"COLOURED PETRI NET"
	NextSend, NextRec	1'1++1'2++1'3++1'4++1'5++1'6++1'7
	А, В	3'(1,"COL")++3'(2,"OUR")++3'(3,"ED ")++ 3'(4,"PET")++3'(5,"RI ")++3'(6,"NET")
	C, D	3'2++3'3++3'4++3'5++3'6++3'7
	Limit	3'()

 The upper bound for DataReceived is a multiset with seven elements although the place always has exactly one token.



State space report: lower multiset bounds

Best Lower Multiset Bounds		
PacketsToSend	1'(1,"COL")++1'(2,"OUR")++1'(3,"ED ")++ 1'(4,"PET")++1'(5,"RI ")++1'(6,"NET")	
DataReceived	empty	
NextSend, NextRec	empty	
A, B, C, D	empty	
Limit	empty	

 The lower bound for DataReceived is empty although the place always has exactly one token.



Coloured Petri Nets Department of Computer Science

More general multiset bounds

 Upper and lower multiset bounds can be generalised to sets of places in a similar way as described for integer bounds.





Coloured Petri Nets Department of Computer Science

More general multiset bounds

 Upper and lower multiset bounds can also be generalised to specific token colours residing on a set of places in a similar way as described for integer bounds.





Coloured Petri Nets Department of Computer Science
Integer and multiset bounds

 The two kinds of bounds supplement each other and provides different kinds of information.

DataReceived

Tells us that DataReceived has at most one token, but gives us no information about the token colours.

DataReceived 1""++1"COL"++1"COLOUR"++1"COLOURED "++ 1"COLOURED PET"++1"COLOURED PETRI "++ 1"COLOURED PETRI NET"

Tells us that DataReceived can have seven different token colours, but not whether they can be present simultaneously.



Home marking

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



- This means that it is 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}.
- However, there is no guarantee that this will happen.



Coloured Petri Nets Department of Computer Science Kurt Jensen Lars M. Kristensen

State space report: home markings

Home Properties

Home Markings: [4868]

- There is a single home marking represented by node number 4868.
- The marking of this node can be shown in the CPN simulator.







Coloured Petri Nets Department of Computer Science Kurt Jensen Lars M. Kristensen

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.



- This means that it is impossible to have an occurrence sequence which cannot be extended to reach a marking in M^{*}_{home}.
- The home property tells that it is possible to reach a marking in M*_{home}.
- However, there is no guarantee that this will happen.



Home predicate

 A home predicate is a predicate on markings Pred_{home} such that at least one marking satisfying Pred_{home} can be reached from any reachable marking.



- This means that it is impossible to have an occurrence sequence which cannot be extended to reach a marking satisfying Pred_{home}.
- The home property tells that it is possible to reach a marking satisfying Pred_{home}.
- However, there is no guarantee that this will happen.



Use of home predicate

 Instead of inspecting node 4868 in the CPN simulator we can check whether DesiredTerminal is a home predicate:







Coloured Petri Nets Department of Computer Science

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.





Coloured Petri Nets Department of Computer Science

Kurt Jensen Lars M. Kristensen





Single terminal SCC

- All markings in the terminal SCC S₂ are home markings.
- No other markings are home markings.





Coloured Petri Nets Department of Computer Science

More than one terminal SCC

- No home markings.
- When one of the terminal SCCs S₂ and S₃ has been reached, it is impossible to leave it again.





Coloured Petri Nets Department of Computer Science Kurt Jensen Lars M. Kristensen

Single SCC

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





Coloured Petri Nets Department of Computer Science Kurt Jensen Lars M. Kristensen

Calculation of home markings

The CPN state space tool uses the following query to calculate the set of all home markings:





Coloured Petri Nets Department of Computer Science Kurt Jensen Lars M. Kristensen

Home spaces and SCCs

- The size of home spaces can be determined from the number of 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.



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



Coloured Petri Nets Department of Computer Science

Liveness properties – being dead

- A marking M is dead if M has no enabled transitions.
- A transition t is dead if t never can occur i.e. is disabled in all reachable markings.
- Generalisations:
- A binding element is dead if it can never become enabled.
- A set of binding elements is dead if none of the binding elements can become enabled.
- A set of transitions is dead if the union of their binding elements is dead.



State space report: being dead

Liveness	Properties
----------	-------------------

Dead Markings: [4868]

Dead Transitions: None

Live Transitions: None

- There is a single dead marking represented by node number 4868.
 - Same marking as home marking.
- There are no dead transitions.



Marking no 4868

- We have seen that marking M₄₈₆₈ represents the state in which we have achieved successful completion of the transmission.
- M₄₈₆₈ is the <u>only</u> dead marking.
- Tells us that the system is partially correct. If execution terminates we will have the correct result.
- M₄₈₆₈ is a home marking.
- Tells us that it always is possible to reach the correct result
 independently of the number of losses and overtakings.



Being dead

- It is straightforward to check whether markings, transitions and binding elements are dead.
- A marking is dead if the corresponding state space node has no outgoing arcs.
- A transition is dead if it does not appear on an arc in the state space.
- A binding element is dead if it does not appear on an arc in the state space.
- A set of binding elements is dead if no binding element in the set appears on an arc in the state space.
- A set of transitions is dead if none of their binding elements appear on an arc in the state space.



Calculation of dead markings

The CPN state space tool uses the following query to calculate the set of all dead markings:

Maps a state space node
into its outgoing arcs
fun ListDeadMarkings () =
PredAllNodes (fn n => (OutArcs n) = []);

Standard query function:
- Searches through all
nodes in the state space
- Returns a list with those
that fulfil the predicate



Coloured Petri Nets Department of Computer Science

Calculation of dead transitions

The CPN state space tool uses the following query to check whether a transition instance is dead:





Coloured Petri Nets Department of Computer Science

Calculation of dead binding elements

 We want to check whether the Sender can receive an acknowledgement with sequence number 1.



Not possible to receive such acknowledgments.



Coloured Petri Nets Department of Computer Science

Liveness properties – being live

 A transition t is live if we from any reachable marking can find an occurrence sequence containing t.



- Liveness tells that it is possible for t to occur.
- However, there is no guarantee that this will happen.



Liveness is a strong property



- If the live transition t occurs in the marking M₂ we reach another reachable marking.
- We can use the new marking as M₁ and hence t is able to occur once more, and so on.
- This means that there exists infinite occurrence sequences in which t occurs infinitely many times.
- It is possible to be non-dead without being live.



Coloured Petri Nets Department of Computer Science

State space report: being live

Liveness Properties

Dead Markings: [4868] Dead Transitions: None

Live Transitions: None

- There are no live transitions
- Trivial consequence of the existence of a dead marking.



Generalisations of liveness

- A binding element is live if it can always become enabled.
- A set of binding elements is live if it is always possible to enable at least one binding element in the set.
- A set of transitions is live if the union of their binding elements is live.



Liveness properties and SCCs

 Liveness 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.
- A set of transitions/binding elements is live if and only if each of the terminal SCCs contains at least one arc with a transition/binding element from the set.



Coloured Petri Nets Department of Computer Science

Single terminal SCC

 A transition is live if it appears on an arc in the terminal SCC S₂.





Coloured Petri Nets Department of Computer Science Kurt Jensen Lars M. Kristensen

More than one terminal SCC

- A transition is live if it appears on an arc in each terminal SCC.
- No live transitions.
- S₃ is terminal and trivial.
- M₇ is a dead marking.





Coloured Petri Nets Department of Computer Science

Single SCC

- A transition is live if it appears on an arc in the SCC.
- In <u>this case</u> we have:
- A transition is live if and only is it is non-dead.





Coloured Petri Nets Department of Computer Science

Fairness properties

 A transition t is impartial if t occurs infinitely often in all infinite occurrence sequences.





Coloured Petri Nets Department of Computer Science

State space report: fairness properties



- SendPacket and TransmitPacket are impartial.
- If one of these are removed (or blocked by the guard false) the protocol will have no infinite occurrence sequences.
- The other three transitions are not impartial.
- If we remove the Limit place only SendPacket will be impartial.
- Adding the Limit place has changed the behavioural properties.



Generalisations of impartial

- A binding element is impartial if it occurs infinitely often in all infinite occurrence sequences.
- A set of binding elements is impartial if binding elements from the set occurs infinitely often in all infinite occurrence sequences.
- A set of transitions is impartial if the union of their binding elements is impartial.



Fairness properties and SCCs

- Impartiality of a transition/binding element X can be checked by means of an SCC graph:
 - Construct the pruned state space in which all arcs with X are removed.
 - Construct the SCC graph of the pruned state space.
 - X is impartial if and only if the two graphs have the same size (are isomorphic).
- Impartiality of a set of transitions/binding elements is checked in a similar way.



Coloured Petri Nets Department of Computer Science

Use of fairness properties

- As an example, we will investigate whether the set of binding elements corresponding to loss of data packets and acknowledgements is impartial.
- If the protocol does not terminate we expect this to be because the network keeps losing packets, and we therefore expect this set of binding elements to be impartial.



Use of fairness properties





Coloured Petri Nets Department of Computer Science

State space report / query functions

- The state space report contains information about standard behavioural properties which make sense for all CPN models.
- Non-standard behavioural properties can be investigated by means of queries.
- For some purposes it is sufficient to provide arguments to a predefined query function – e.g. to check whether a set of markings constitute a home space.
- For other more special purposes it is necessary to write your own query functions using the CPN ML programming language.



Example of user-defined query function

 We want to check whether the protocol obeys the stop-and-wait strategy – i.e. that the sender always sends the data packet expected by the receiver (or the previous one).

fun StopWait n =
 Converts a multiset 1`x with
 one element to the colour x
 val NextSend = ms_to_col (Mark.Protocol'NextSend 1 n);
 val NextRec = ms_to_col (Mark.Protocol'NextRec 1 n);
 in
 (NextSend = NextRec) orelse (NextSend = NextRec - 1)
 end;

val SWviolate = PredAllNodes (fn n => not (StopWait n));
Predefined search function
Negation
The stop-and-wait strategy is
not satisfied (7020 violations).
We check whether some states
violate the property.
This is easier than checking



73

that all states fulfil the property.


Violation of stop-and-wait strategy

- Acknowledgements may overtake each other on C and D.
- This means that it is possible for the sender to receive an old acknowledgement which decrements NextSend.





Coloured Petri Nets Department of Computer Science Kurt Jensen Lars M. Kristensen

Shortest counterexample

- We want to construct a shortest counterexample i.e. to find one of the shortest occurrence sequences leading from the initial marking to a marking where the predicate does not hold.
- The state space is generated in breadth-first order.
- Hence, we search for the lowest numbered node in the list SWviolate.



Lowest node in SWviolate



The function iterates over the list.

- In each iteration the combination function is applied to the pair consisting of the current element in the list and the value returned by the previous application of the combination function.
- In the first iteration, the initial value plays the role of the result from the previous application.

557 Violating marking (as close to M₀ as possible) Can be expected in the simulator



Coloured Petri Nets Department of Computer Science Kurt Jensen Lars M. Kristensen

Shortest counterexample



- The path can be visualised using the drawing facilities in the CPN state space tool.
- This is the same drawing facilities that were used to visualise the initial fragment of the state space (at the beginning of this lecture).



Coloured Petri Nets Department of Computer Science Kurt Jensen Lars M. Kristensen

Bindings elements in counterexample

 The binding elements in the shortest path can be obtained by the following query:





Coloured Petri Nets Department of Computer Science

Shortest counterexample

		1 (SendPacket, <d="col",n=1>)</d="col",n=1>
		2 (TransmitPacket, <n=1,d="col",success=true>)</n=1,d="col",success=true>
Packet no 1		3 (ReceivePacket, <k=1,data="",n=1,d="col">)</k=1,data="",n=1,d="col">
and its ack	•	4 (SendPacket, <d="col",n=1>)</d="col",n=1>
(5 (TransmitAck, <n=2,success=true>)</n=2,success=true>
		6 (ReceiveAck, <k=1,n=2>)</k=1,n=2>
(7 (SendPacket, <d="our",n=2>)</d="our",n=2>
		8 (TransmitPacket, <n=1,d="col",success=true>)</n=1,d="col",success=true>
Packet no 2	•	9 (TransmitPacket, <n=2,d="our",success=true>)</n=2,d="our",success=true>
and its ack	• 1	0 (ReceivePacket, <k=2,data="col",n=1,d="col">)</k=2,data="col",n=1,d="col">
(• 1 [′]	1 (ReceivePacket, <k=2,data="col",n=2,d="our">)</k=2,data="col",n=2,d="our">
	• 1:	2 (TransmitAck, <n=3,success=true>)</n=3,success=true>
	• 1:	3 (ReceiveAck, <k=2,n=3>)</k=2,n=3>
Packet no 3	• 1	4 (SendPacket, <d="ed ",n="3">)</d="ed>
NextRec = 4	• 1	5 (TransmitPacket, <n=3,d="ed ",success="true">)</n=3,d="ed>
Retrans-	• 1	6 (ReceivePacket, <k=3,data="colour",n=3,d="ed "="">)</k=3,data="colour",n=3,d="ed>
mission	• 1	7 (TransmitAck, <n=2,success=true>)</n=2,success=true>
NextSend = 2	• 18	8 (ReceiveAck, <k=3,n=2>)</k=3,n=2>



Kurt Jensen Lars M. Kristensen

Revised protocol

 Now we only send an acknowledgement when an expected packet is received.





Coloured Petri Nets Department of Computer Science

else emptv

Kurt Jensen Lars M. Kristensen

State space for revised protocol

- The state space contains 1,823 nodes and 6,829 arcs.
- Before we had 13,215 nodes and 52,874 arcs.
- As before there is a single dead marking which corresponds to the desired terminal marking, where all packets have been successfully transmitted.
- The new protocol is partially correct.



- Now there are <u>no</u> home markings.
- We can reach situations from which it is impossible to reach the desired terminal marking.



Coloured Petri Nets Department of Computer Science

Analysis of revised protocol

- The dead marking is no longer a home marking and hence we must have one or more terminal SCCs from which we cannot reach the dead marking.
- These terminal SCCs can be found by the following query which returns all SCCs that are terminal but not trivial:

PredAllSccs (fn scc => SccTerminal scc andalso
Standard query function: not (SccTrivial scc));
- Searches through all nodes in the SCC graph

- Returns those which fulfil the predicate
- The result of the query is a list with six SCCs.
- The state space nodes in the six SCCs can be obtained using the function SccToNodes.
- To get a shortest counterexample, we choose the lowest numbered node which is node 12.



Marking no 12





Coloured Petri Nets Department of Computer Science

What is wrong

- The analysis of marking no 12 has told us what the problem is.
- The sender continues to send wrong packets and the receiver never sends an acknowledgement which can correct the problem.
- We might also want to know how we arrived at this unfortunate situation.
- This is done by constructing an error trace / counterexample.



Counterexample

The query below returns a list with all the arcs in one of the shortest paths from node 1 (initial marking) to node number 12:

```
ArcsInPath(1,12);
```

The binding elements in the shortest path can be obtained by the following query:





Coloured Petri Nets Department of Computer Science Kurt Jensen Lars M. Kristensen

Counterexample

- The result of the query is the following list of binding elements:
 - 1 (SendPacket, <d="COL", n=1>)
 - 2 (TransmitPacket, <d="COL",n=1,success=true>)
 - 3 (ReceivePacket, <d="COL",n=1,k=1, data="">)
 - 4 (TransmitAck, <n=2,success=false>)
- We see that data packet no 1 was sent, successfully transmitted, and received.
- However, the acknowledgment requesting data packet no 2 was lost on the network.



System configurations

- With state space analysis we always investigate a system for a particular configuration of the system parameters.
- In practice it is often sufficient to consider a few rather small configurations – although we cannot be totally sure that larger configurations will have the same properties.
- As system parameters increase the size of the state space increases – often in an exponential way.
- This is called state space explosion, and it is one of the most severe limitations of the state space method.



Coloured Petri Nets Department of Computer Science

Different system configurations

Limit	Packets	Nodes	Arcs	Limit	Packets	Nodes	Arcs
1	10	81	110	5	1	217	760
1	20	161	220	5	2	2,279	10,645
1	50	401	550	5	3	17,952	97,963
1	100	801	1,100	5	4	82,260	483,562
1	600	4,801	6,600	5	5	269,680	1,655,021
2	1	26	53	7	1	576	2,338
2	5	716	1,917	7	2	11,280	64,297
2	10	3,311	9,062	7	3	148,690	1,015,188
2	20	14,276	39,402	10	1	1,782	8,195
2	50	93,371	258,822	10	2	76,571	523,105
3	1	60	159	12	1	3,276	15,873
3	5	7,156	28,201	12	2	221,117	1,636,921
3	10	70,131	286,746	13	1	4,305	21,294
3	15	253,656	1,047,716	13	2	357,957	2,737,878





Coloured Petri Nets Department of Computer Science

Is it worthwhile?

- State space analysis can be a time consuming process where it takes many hours to generate the state spaces and verify the desired properties.
- However, it is fully automatic and hence requires much less human work than lengthy simulations and tests.
- It may take days to verify the properties of a system by means of state spaces.
- However, this is still a relatively small investment:
 - compared to the total number of resources used in a system development project.
 - compared to the cost of implementing, deploying and correcting a system with errors that could have been detected in the design phase.



Partial state spaces

- It is sometimes impossible to generate the full state space for a given system configuration – either because it is too big or takes too long time.
- This means that only a partial state space i.e. a fragment of the state space is generated.
- Partial state spaces cannot in general be used to verify properties, but they may identify errors.
- As an example, an undesirable dead marking in a partial state space will also be present in the full state space.
- Partial state spaces can in that sense be viewed as being positioned between simulation and state spaces.
- The CPN state space tool has a number of parameters to control the generation of partial state spaces.



State spaces - summary

- State spaces are powerful and easy to use.
 - Construction and analysis can be automated.
 - The user do not need to know the mathematics behind the analysis methods.
- The main drawback is the state explosion i.e. the size of the state space.
 - The present CPN state space tool handles state spaces with up to one million states.
 - For many systems this is not sufficient.
 - Efficient state space methods is an active area of research and an abundance of sophisticated techniques exists for alleviating the state explosion problem.



Coloured Petri Nets Department of Computer Science

Reduced state spaces

- Fortunately, it is often possible to construct reduced state spaces – without losing analytic power.
- This is done by exploiting:
 - Progress measure.
 - Symmetries in the modelled system.
 - Other kinds of equivalent behaviour.
 - Concurrency between events.
- The reduction methods rely on complex mathematics.



Questions





Coloured Petri Nets Department of Computer Science Kurt Jensen Lars M. Kristensen

Assignment 3 – Task 2

Task 2:

Use state spaces to analyse the Sliding Window Protocol from task 1.

- To obtain a finite state space you should introduce a Limit place as shown in the lectures. Having done this you can construct the state space, the SCC graph, and the state space report. Start by setting the Limit to 1 or 2. Later you should increase it.
- Alternatively: consider a variant of the protocol with bounded retransmission of data packets. Available via: <u>http://www.hib.no/ansatte/lmkr/StateSpacesBounded.xml</u>
- Use the state space reports to investigate whether your protocol works correctly. You should check that the bounds are ok and that the dead markings and home markings/spaces are as expected.



Coloured Petri Nets Department of Computer Science