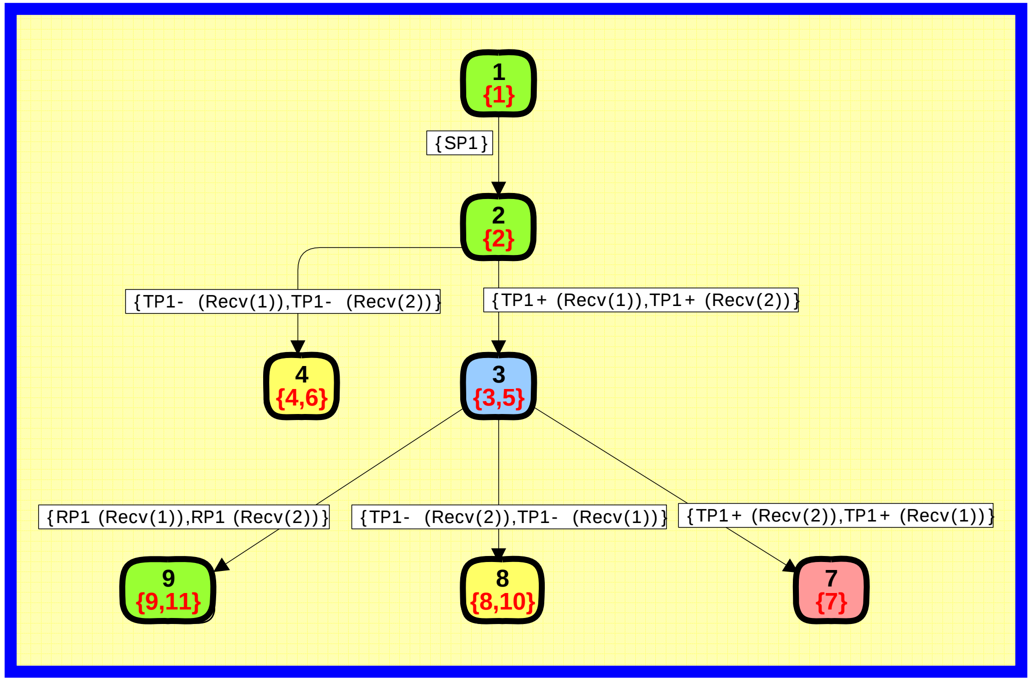
Coloured Petri Nets

Modelling and Validation of Concurrent Systems

Chapter 8: Advanced State Space Methods

Kurt Jensen &
Lars Michael Kristensen

{kjensen,lmkristensen}
@daimi.au.dk



State space methods

- The main limitation of using state spaces to verify behavioural properties of systems is the state explosion problem.

- State spaces of many systems have an astronomical number of reachable states.

- This means that they are too large to be handled with the available computing power:
  - Memory.
  - CPU speed.

State space reduction methods

- Methods for alleviating the state explosion problem is an active area of research. They allow:
  - faster construction,
  - more compact representation (less memory).

- A large collection of state space reduction methods exists.

- The reduction methods have significantly increased the class of systems that can be verified in practice.

- State spaces can now be used to verify systems of industrial size.

Independent of modelling language

- Most state space reduction methods are independent of the concrete modelling language and hence applicable for a large class of such languages (e.g. all transition systems).

- Some of the reduction methods have been developed within the context of the CPN modelling language:
  - Sweep-line method.
  - Symmetry method.
  - Equivalence method.

- Other reduction methods have been developed outside the context of the CPN modelling language.

Why different reduction methods?

- State space reduction methods typically exploit certain characteristics of the system under analysis.
- No single reduction method works well for all kind of systems.

- Furthermore, the methods often limit the verification questions that can be answered.

- When verifying a concrete system one must therefore choose a method that:
  - exploits characteristics present in the system.
  - preserves the behavioural properties to be verified.

On-the-fly verification

- Many reduction methods are based on the paradigm of on-the-fly verification.

- The verification question is stated before the exploration of the state space starts.
- The state space exploration is done relative to the provided verification question.

- This makes it possible to terminate the state space exploration as soon as the answer to the verification question has been obtained – ignoring irrelevant parts.

Modelling and Validation of Distributed Systems Group
Department of Computer Science

Model checking

- Many advanced state space reduction methods use temporal logic for stating the verification questions :
  - Linear-time temporal logic (LTL).
  - Computation tree temporal logic (CTL).
- The use of temporal logic for stating and checking verification questions is referred to as model checking.

State spaces are kept in main memory

- The amount of available main memory is often the limiting factor in the practical use of state spaces.

- During construction of the state space, the set of markings encountered are kept in main memory.

- This allows us to recognise already visited markings and thereby ensure that the state space exploration terminates.

Method 1: Sweep-line method

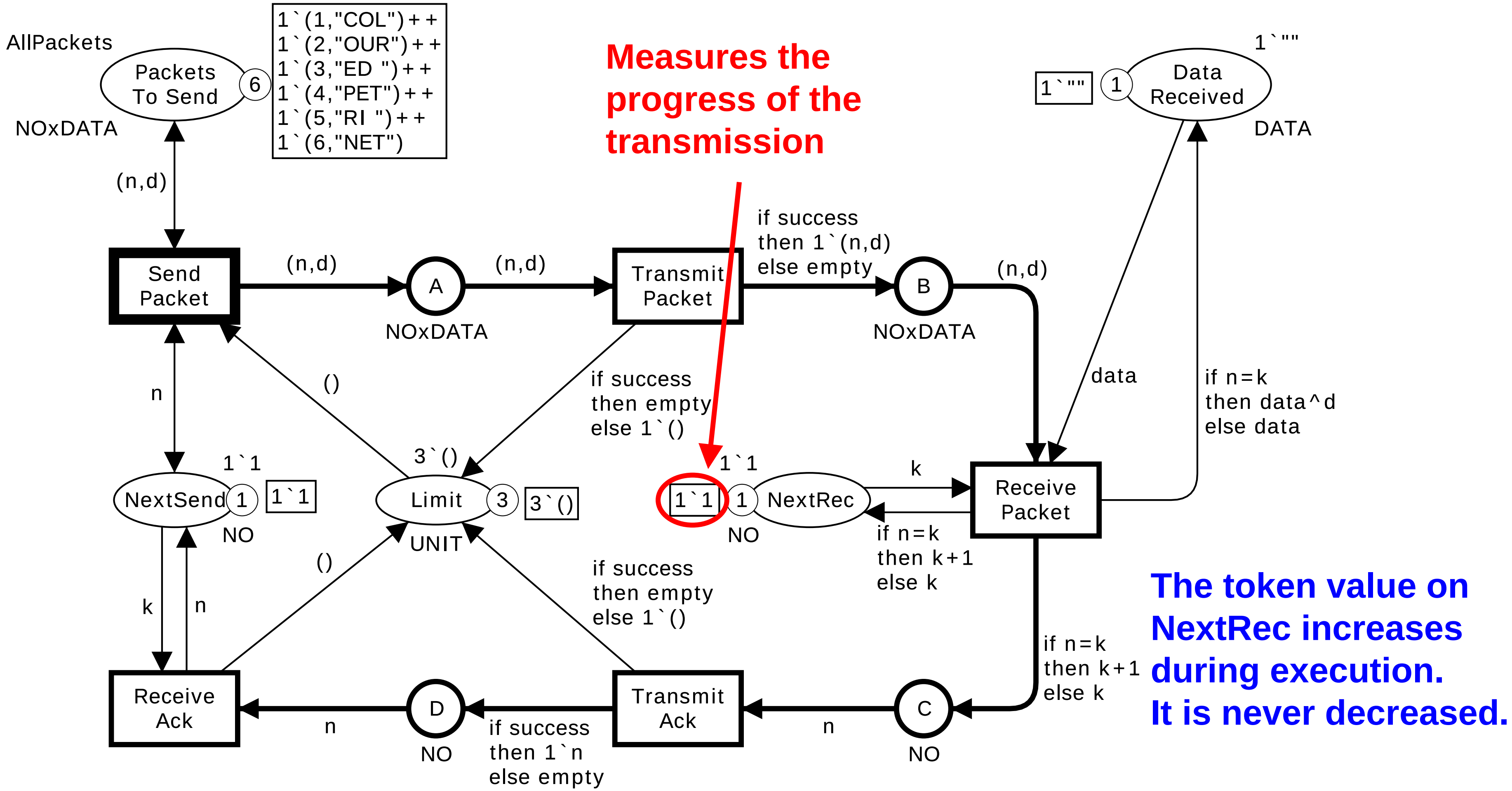
- The basic idea of the sweep-line method is to exploit a certain kind of progress exhibited by many systems.

- Exploiting progress makes it possible to explore all the reachable markings of a CPN model, while only storing small fragments of the state space in main memory at a time.

- This means that the peak memory usage is significantly reduced.

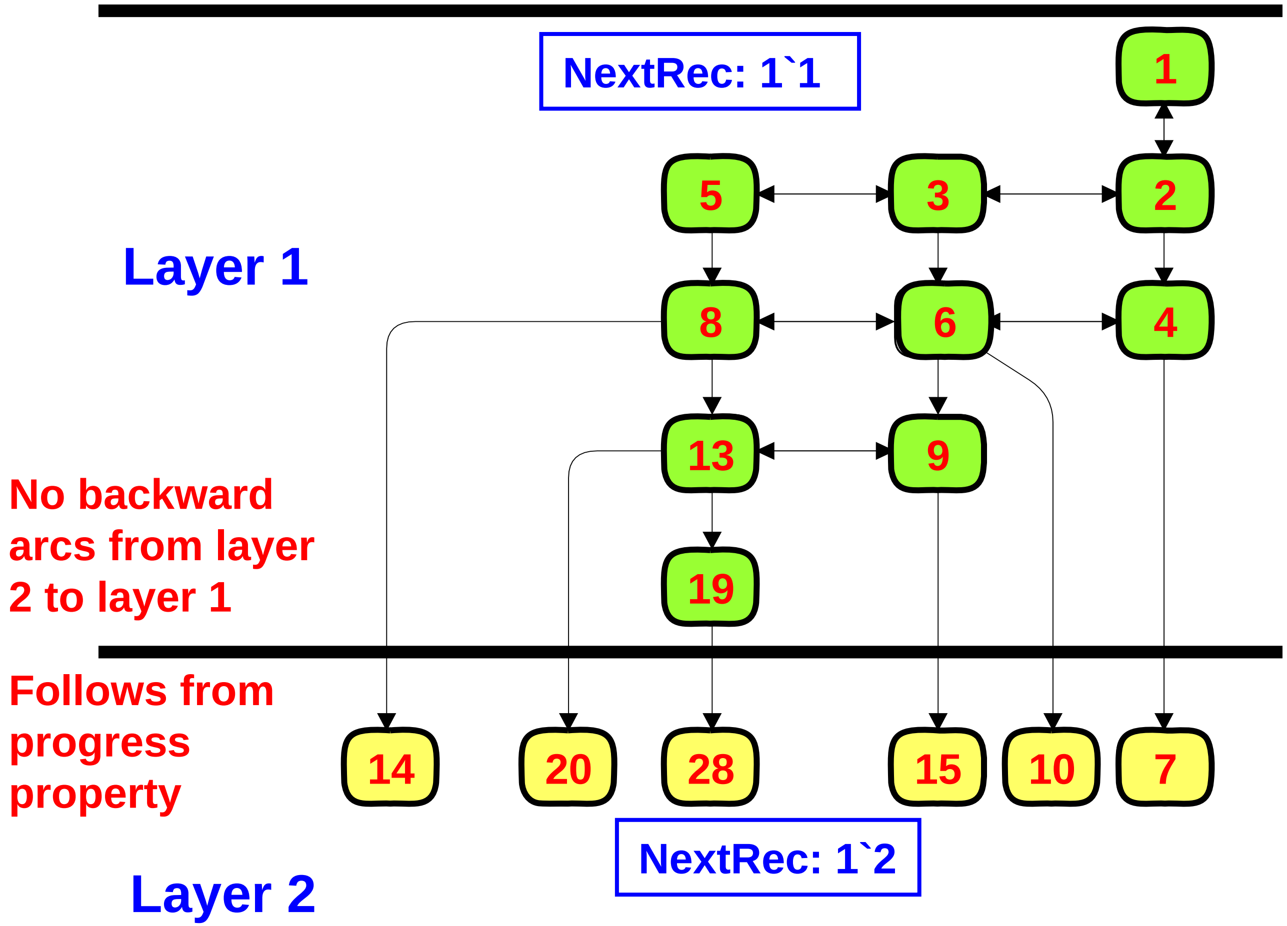
- The sweep-line method is aimed at on-the-fly verification of safety properties (e.g., determining whether a reachable marking exists satisfying a given predicate).

Simple protocol



Initial fragment of state space

- Each marking has successor markings either in the same layer or in higher layers – never in lower layers.



We process markings layer by layer

- We process the markings (i.e., calculate successor markings) one layer at a time.
- We only move from one layer to the next when all markings in the first layer have been processed.

- We can think of this as a sweep-line moving through the state space (layer by layer).

- At any time during state space exploration, the sweep-line corresponds to a single layer.
  - All markings in the layer are "on" the sweep-line.
  - All new markings calculated are either on the sweep-line or in front of the sweep-line (i.e. in a higher layer).

Progress measure

- The progress in the protocol system is captured by a progress measure which is a function mapping each marking into a progress value.

Converts a multi-set 1`x with one element to the colour x

```
fun ProtocolPM n = ms_to_col (Mark.Protocol'NextRec 1 n);
```

- Monotonic progress measure:

$$M' \in \Re(M) \Rightarrow \text{ProtocolPM } M \leq \text{ProtocolPM } M'$$

Statistics for sweep-line method

Limit	Packets	Nodes	Arcs	Nodes (peak)	Nodes	Time
1	4	33	44	33	1.00	1.00
2	4	293	764	134	2.19	1.00
3	4	1,829	6,860	758	2.41	1.00
4	4	9,025	43,124	4,449	2.03	1.78
5	4	37,477	213,902	20,826	1.80	1.65
6	4	136,107	891,830	82,586	1.65	1.51
4	5	20,016	99,355	8,521	2.35	1.95
4	6	38,885	198,150	14,545	2.67	2.19
4	7	68,720	356,965	22,905	3.00	2.27
4	8	113,121	596,264	33,985	3.33	2.41

Configuration — Standard method — Sweep-line — Gain

Summary for sweep-line method

- From the statistics on the previous slide, it can be seen that the sweep-line method yields a reduction in both space and time.
- The space reduction was expected since markings are deleted during state space exploration.
- The time reduction is because the deletion of states implies that there are fewer markings to compare with when determining whether a marking has been seen before.

Generalised sweep-line method

- Above we have used a monotonic progress measure:

$$M' \in \Re(M) \Rightarrow \text{ProtocolPM } M \leq \text{ProtocolPM } M'$$

- It is also possible to use a generalised sweep-line method where the monotonicity property only is satisifed by most steps.
- The generalised sweep-line method performs multiple sweeps of the state space, and it makes certain markings persistent which means that they cannot be deleted from memory.
- The sweep-line method has also been generalised to use external storage such that counter examples and diagnostic information can be obtained.
- This is not possible in the basic method since it deletes the markings from memory.

Method 2: Symmetry method

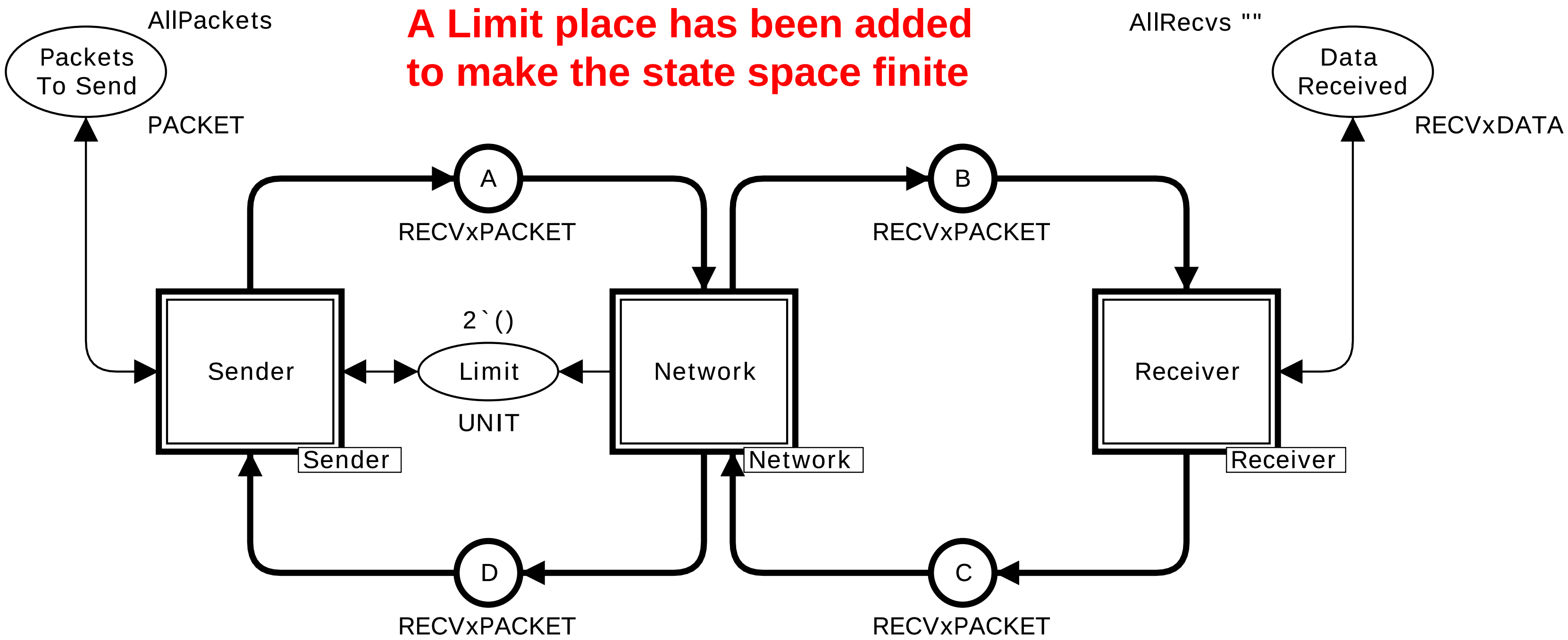
- Many concurrent systems possess a certain degree of symmetry.
- They may e.g. have similar components whose identities are interchangeable from a verification point of view.

- The basic idea in the symmetry method is to represent symmetric markings and symmetric binding elements using equivalence classes.
  - Each node represents a class of equivalent markings (instead of a single marking).
  - Each arc represents a class of equivalent binding elements (instead of a single binding element).

Construction and analysis

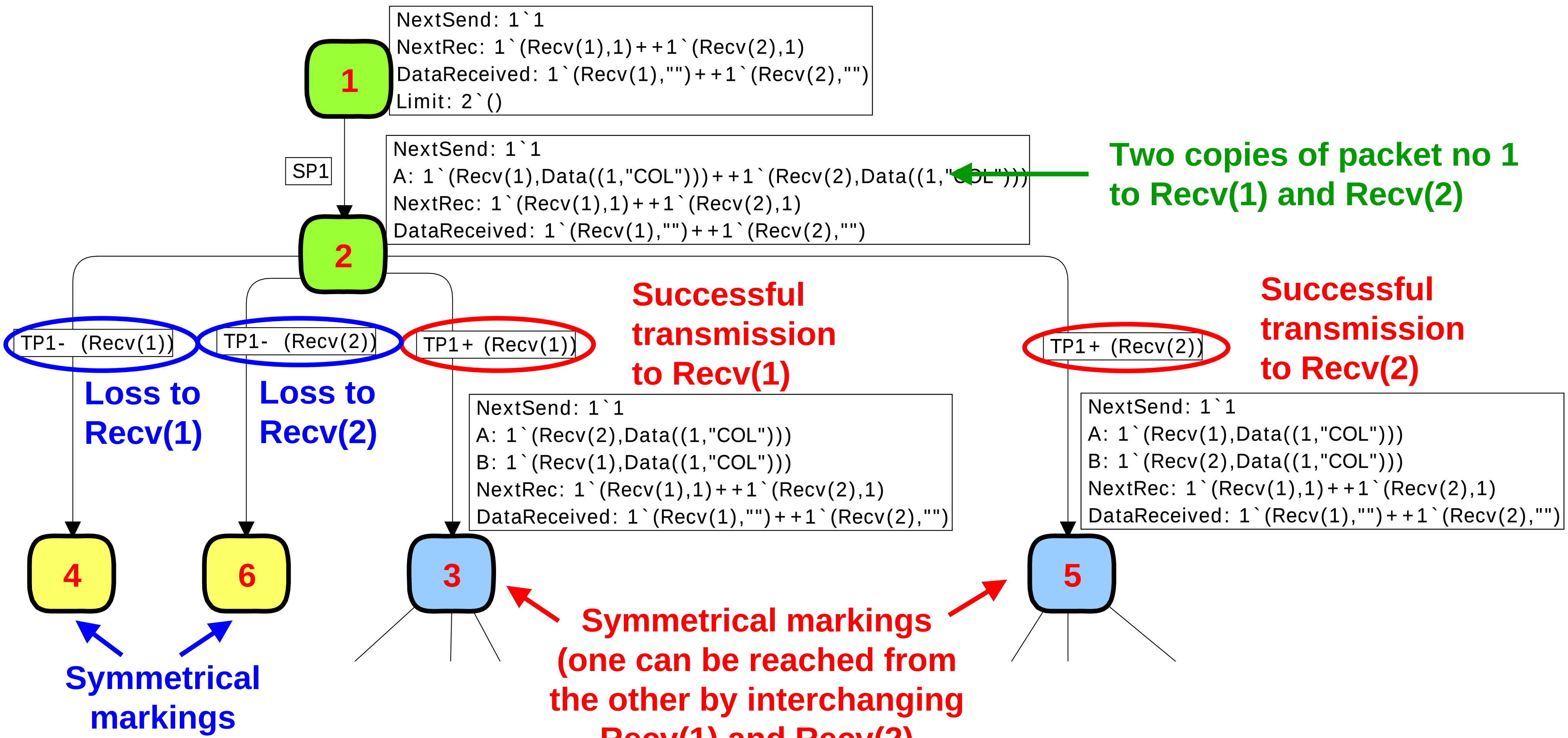
- Symmetry condensed state spaces are typically orders of magnitude smaller than the corresponding full state spaces.
- They can be constructed directly without first constructing the full state space and then grouping nodes and arcs into equivalence classes.
- Furthermore, behavioural properties can be verified directly on the symmetry condensed state space without unfolding to the full state space.

Protocol with multiple receivers

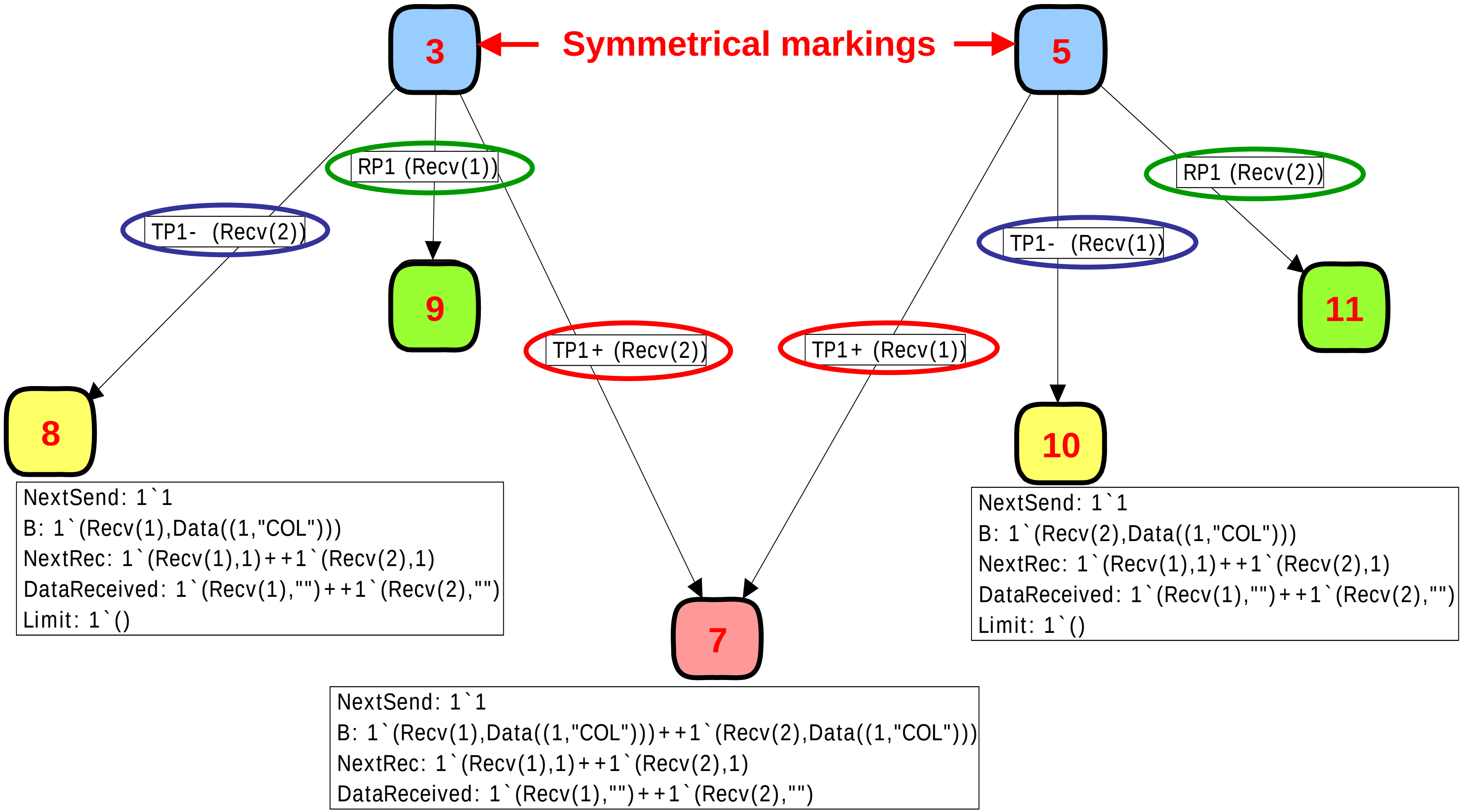


- The receivers in the protocol system are symmetric, in the sense that they all behave in the same way.
- They are only distinguishable by their identity.

State space (ordinary)



Symmetrical successors

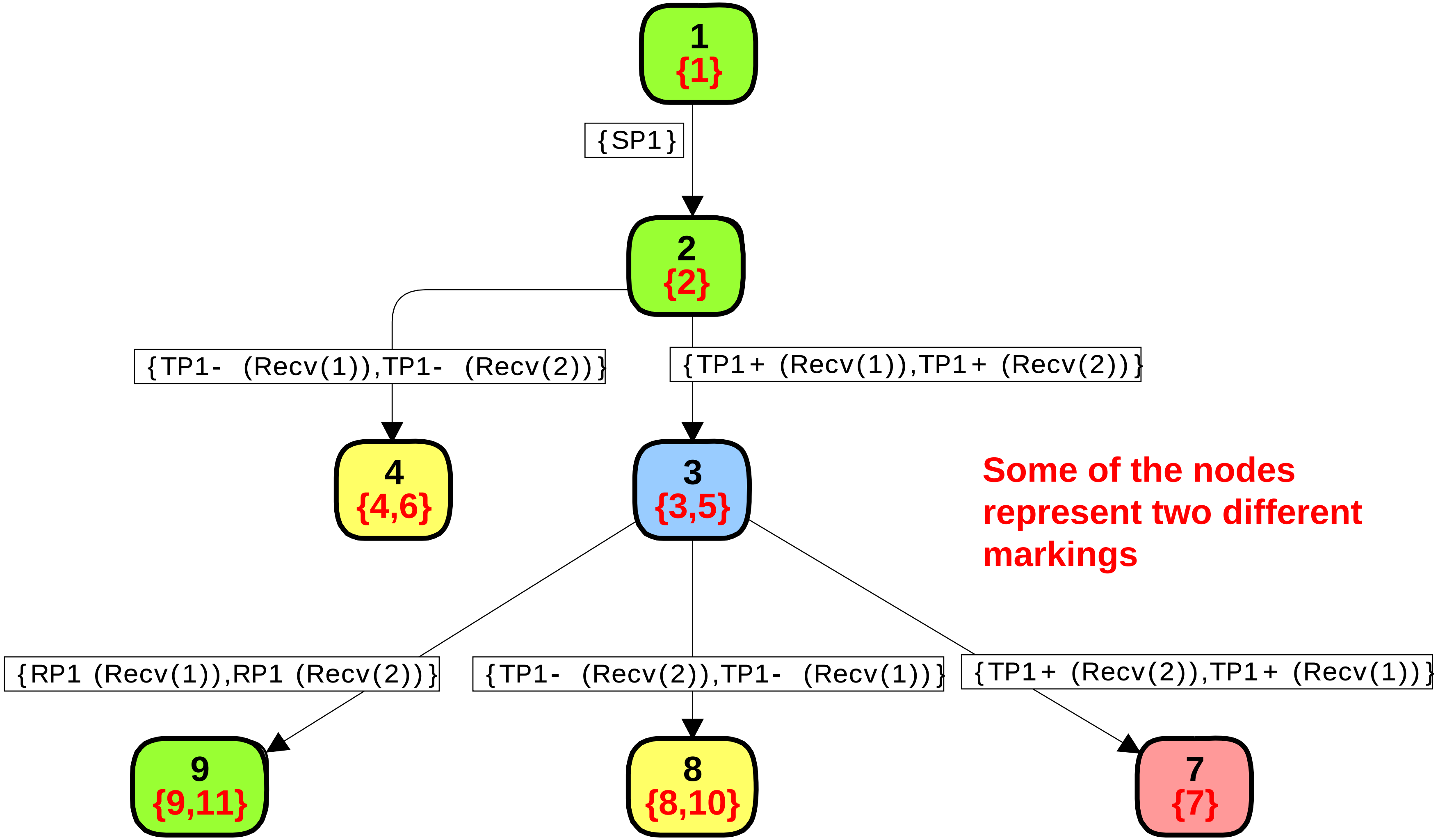


Symmetrical markings

- On the previous slide we saw that the two symmetric markings M_3 and M_5 have:
  - symmetric sets of enabled binding elements,
  - symmetric sets of direct successor markings.

- By induction this property can be extended to finite and infinite occurrence sequences:
  - For any occurrence sequence starting in a marking M and all markings M′ symmetric with M there exists a symmetric occurrence sequence starting in M′.
  - The things which can happen from M can also happen from M′ (up to symmetry).

Symmetry condensed state space



Some of the nodes represent two different markings

Soundness criteria

- The symmetries used to reduce the state space are required to be symmetries actually present in the CPN model:

- All initial marking inscriptions must be symmetric (applying a permutation to the initial marking does not change the initial marking).
- All guard expressions must be symmetric (evaluating the guard in a binding must give the same result as first permuting the binding element and then evaluating the guard).
- All arc expressions must be symmetric (evaluating the arc expression in a binding and then applying a permutation must give the same result as first permuting the binding element and then evaluating the arc expression).

Static checks by local examination of net inscriptions

Specification of symmetries

- Colour sets are divided into:
  - Atomic (Int, Bool, String, Unit, enumerations, indexed).
  - Structured (products, records, unions, lists, subsets).

- Each atomic colour set is associated with an algebraic group of allowed permutations.
- The structured colours sets inherits their permutations from the colour sets from which they are constructed.

- Examples of permutation groups are:
  - all permutations in the colour set,
  - all rotations in an ordered colour set,
  - identity element alone (no permutation allowed).

Protocol with multiple receivers

- Atomic colour sets:

```
colset NO   = int;
colset DATA = string;
colset RECV = index Recv with 1..NoRecv;
```

No permutations

All permutations

- Structured colour sets:

```
colset NOxDATA = product NO * DATA;
colset PACKET  = union Data : NoxDATA + Ack : NO;
colset RECVxDATA   = product RECV * DATA;
colset RECVxPACKET = product RECV * PACKET;
colset RECVxNO     = product RECV * NO;
```

No permutations

All permutations of Recv-component

Statistics for symmetry method

L P R	Nodes	Arcs	Nodes	Arcs	Nodes	Arcs	Time	R!
2 3 2	921	1,832	477	924	1.93	1.98	0.7	2
3 3 3	22,371	64,684	4,195	11,280	5.33	5.73	2.0	6
4 3 4	172,581	671,948	9,888	32,963	17.45	20.38	23.9	24
5 2 5	486,767	2,392,458	8,387	31,110	58.04	76.90	—	120
6 2 6	5,917,145	35,068,448	24,122	101,240	245.30	346.39	—	720

Configuration Standard method Symmetry Gain

L = Limit
P = Packets
R = Receivers

Number of possible permutations

Modelling and Validation of Distributed Systems Group
Department of Computer Science

Summary for symmetry method

- Significant reductions can be obtained as illustrated on the protocol with multiple receivers.
- The method can be sued to check all behavioural properties that are invariant under symmetry.

- Computation of the canonical representations of markings and binding elements is computational expensive.
  - At least as hard as the graph isomorphism problem for which no polynomial time algorithm is known.
  - The present algorithms exploits a number of advanced algebraic techniques and can efficiently deal with systems where the number of permutation symmetries are below 10!
  - This is usually sufficient in practice.

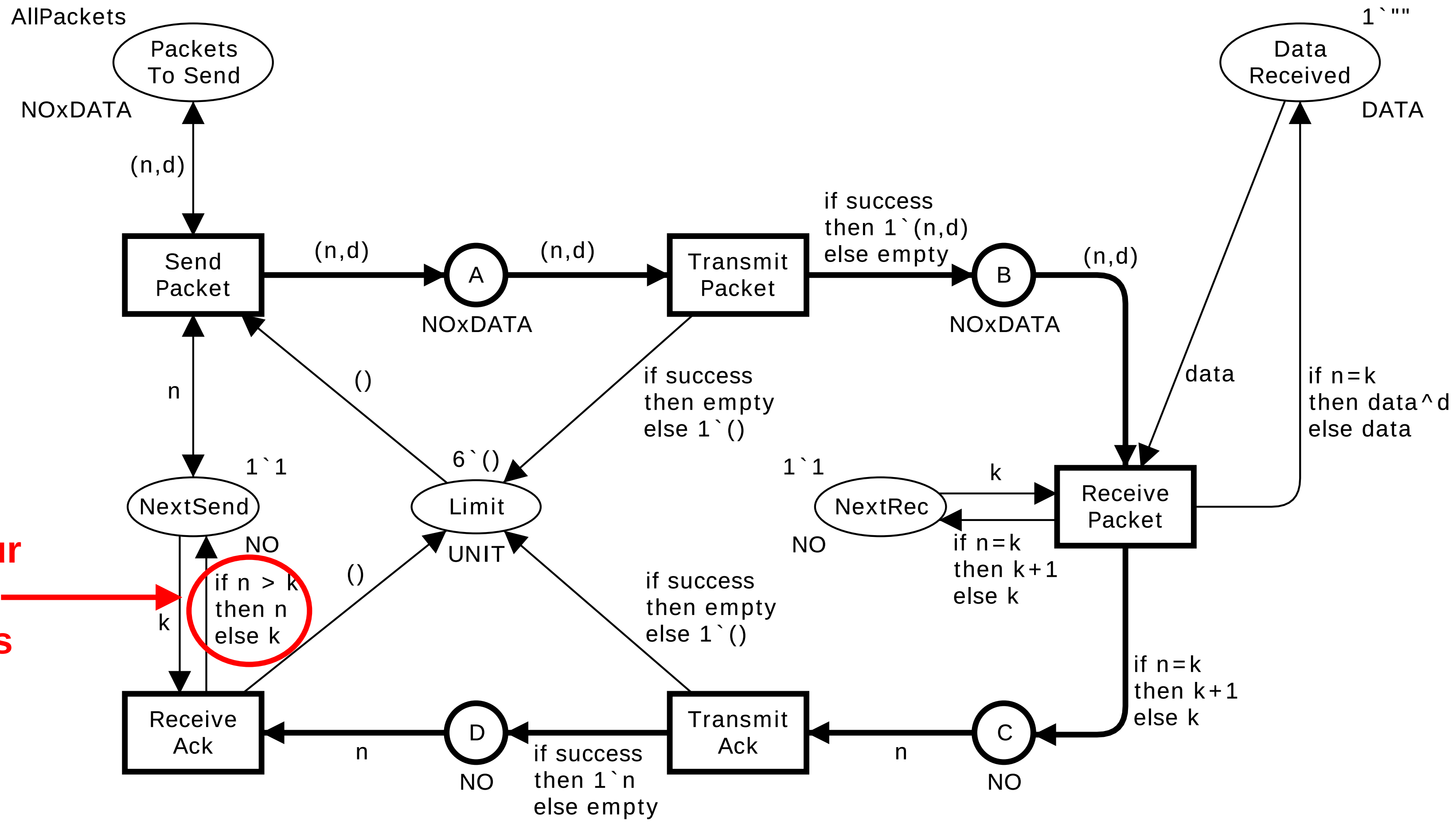
Method 3: Equivalence method

- The equivalence method is a generalisation of the symmetry method.

- In the symmetry method we have equivalence relations on the markings and on the binding elements.
- The equivalence relations are induced by the permutation symmetries.

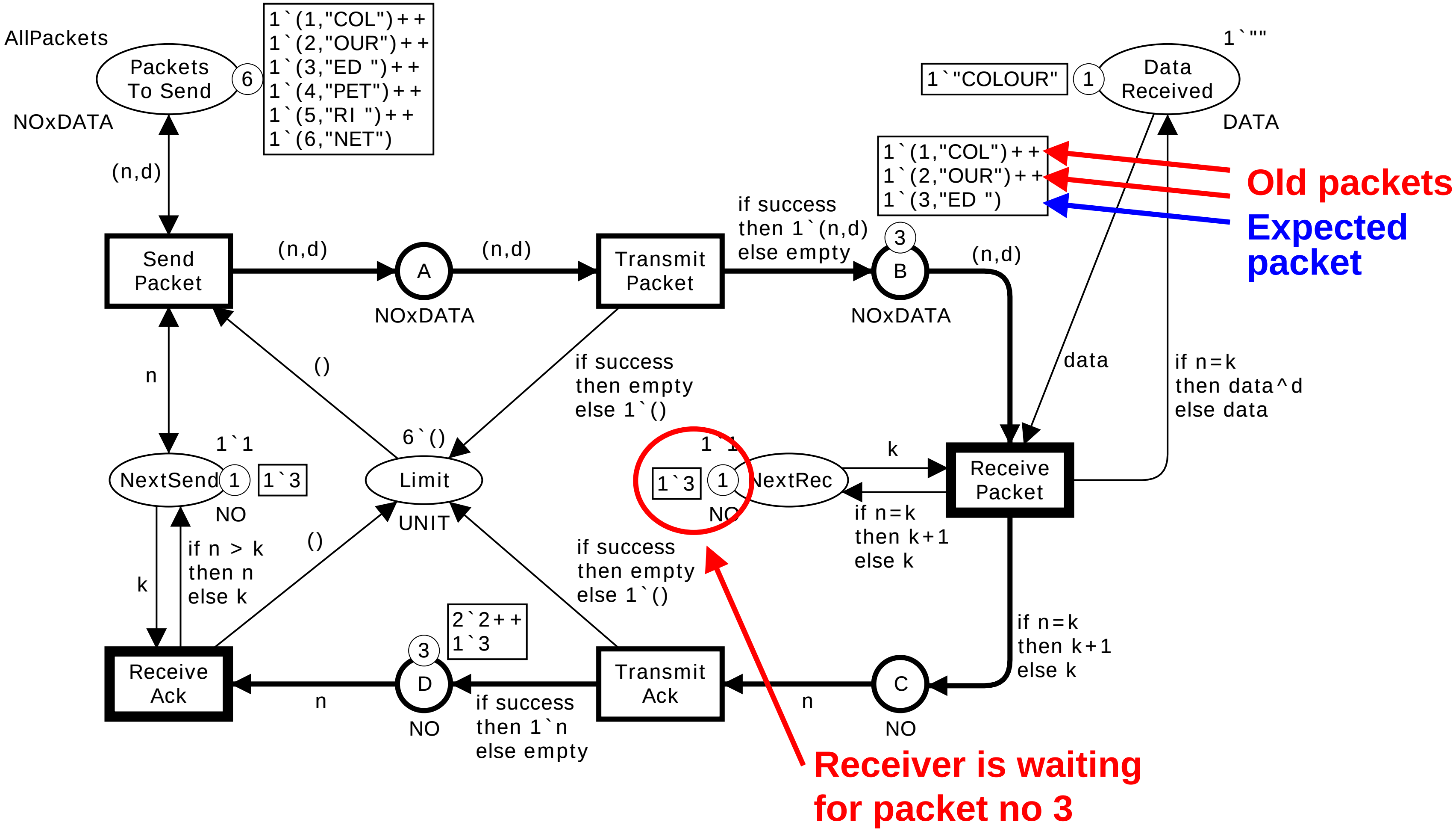
- In the equivalence method the equivalence relations are specified directly (without the use of symmetries).
- Soundness criteria: Equivalent markings must have equivalent sets of enabled binding elements and equivalent sets of successor markings.

Simple protocol (slightly modified)



The token colour on NextSend never decreases

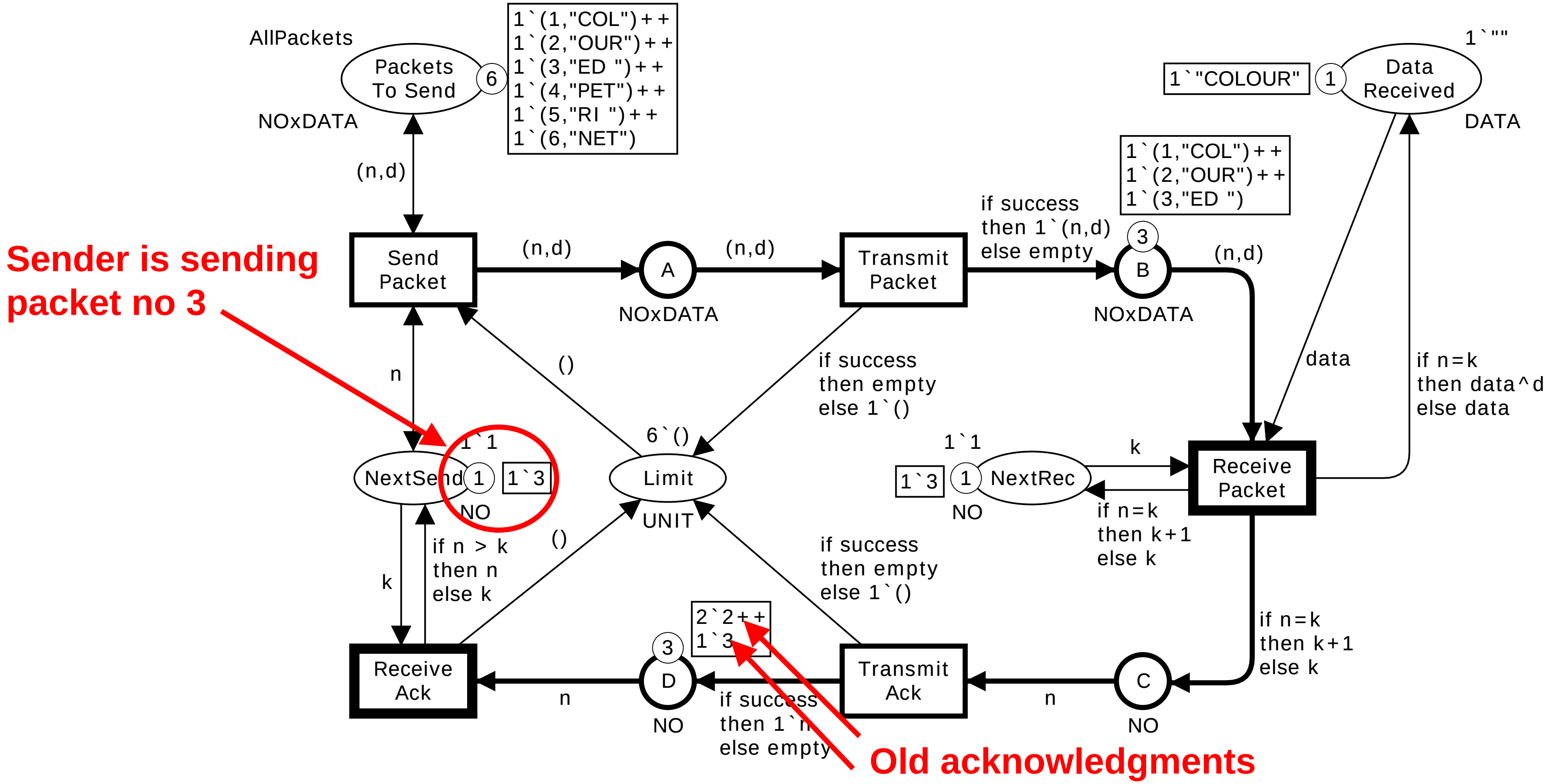
Old packets



Modelling and Validation of Distributed Systems Group
Department of Computer Science

Kurt Jensen and Lars M. Kristensen
Coloured Petri Nets

Old acknowledgments



Modelling and Validation of Distributed Systems Group
Department of Computer Science

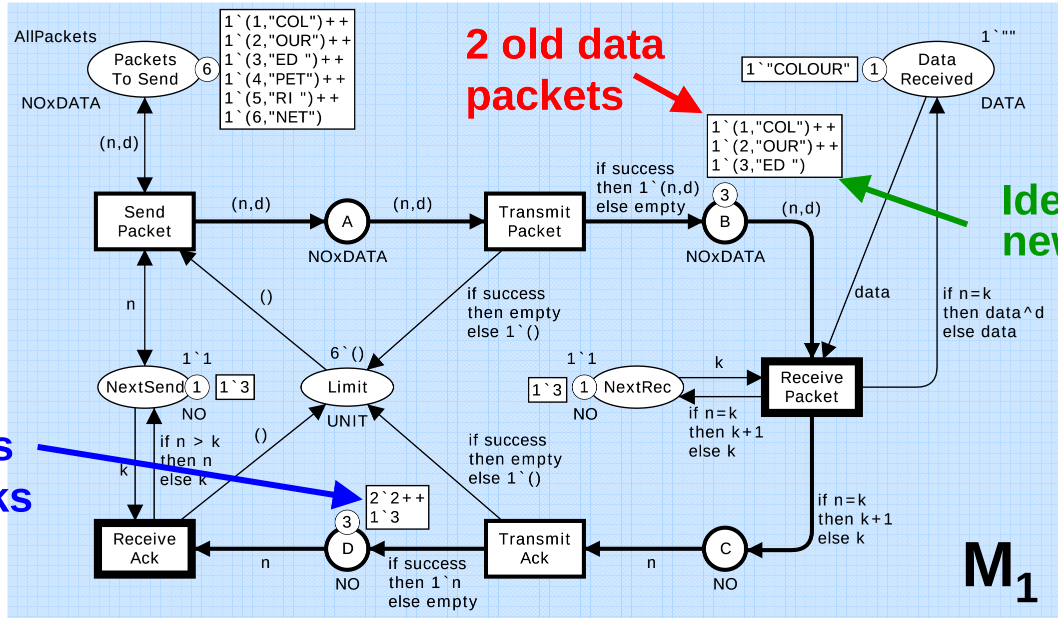
Kurt Jensen and Lars M. Kristensen

Equivalence relation for markings

- Basic idea:
  - Old data packets can be replaced by other old data packets.
  - Old acknowledgements can be replaced by other old acknowledgements.

- Two markings are equivalent if the following conditions hold:
  - Markings of A, B, C, and D: Identical non-old packets and the same number of old packets.
  - All other places must have identical markings.

Two equivalent markings

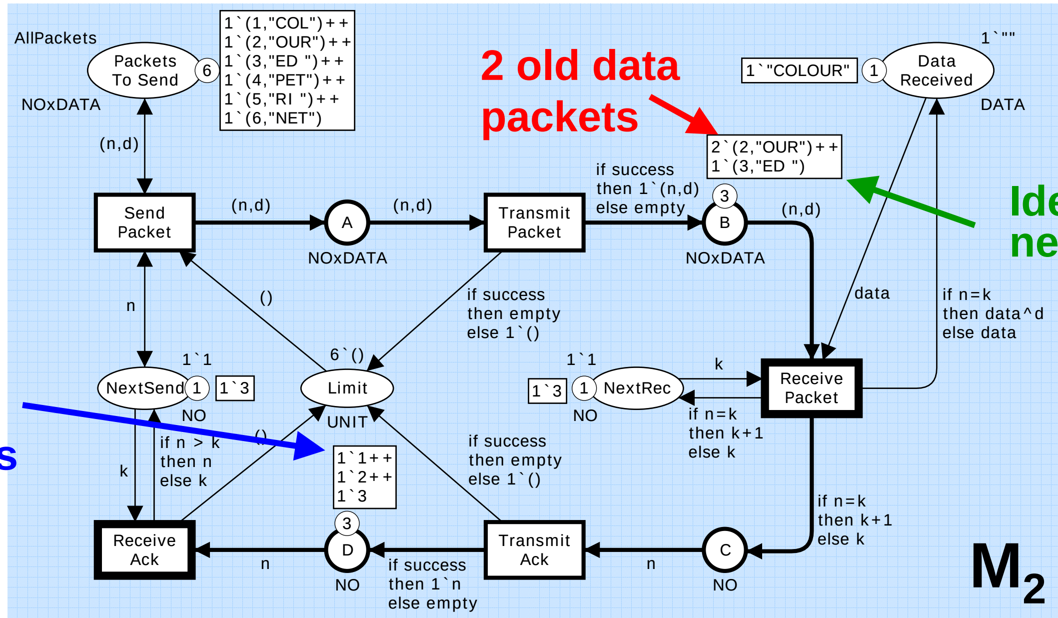


Identical new packet

3 old acks
0 new acks

All other places have identical markings

The two markings are equivalent to each other



Identical new packet

3 old acks
0 new acks

Equivalence relation for binding elements

- Two bindings of the same transition are equivalent to each other if they both involve old data packets or both involve old acknowledgements.
- All other binding elements are non-equivalent.

Statistics for equivalence method

Configuration		Standard method		Equivalence		Gain		
L	P	Nodes	Arcs	Nodes	Arcs	Nodes	Arcs	Time
1	4	33	44	33	44	1.00	1.00	1.00
2	4	293	764	155	383	1.89	1.99	1.00
3	4	1,829	6,860	492	1,632	3.72	4.20	0,90
4	4	9,025	43,124	1,260	5,019	7.16	8.59	1.56
5	4	37,477	213,902	2,803	12,685	13.37	18.86	4.09
6	4	136,107	891,830	5,635	28,044	24.15	31.80	13.58

L = Limit
P = Packets

Summary for equivalence method

- The equivalence method allows a more dynamic/general notion of equivalence than the symmetry method.
- Hence it can be used in situations where the symmetry method are of no use.

- The consistency proof must be done manually.
- The equivalence relations must be implemented manually (as ML functions).

- Later we shall see that the equivalence method can be used to reduce state spaces for timed CPN models (without manual consistency proof and with automatic implementation).

Multiple reduction methods

- It is often possible to simultaneously use two or more state space reduction methods.

- This leads to more reduction:
  - in CPU, and
  - memory usage

  than each method used in isolation.

- The sweep-line, symmetry, and equivalence methods can be used simultaneously with each other.

Modelling and Validation of Distributed Systems Group
Department of Computer Science