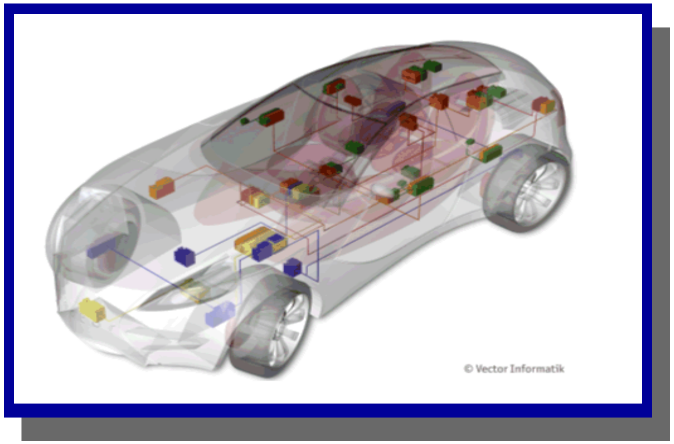
High-level Petri Nets

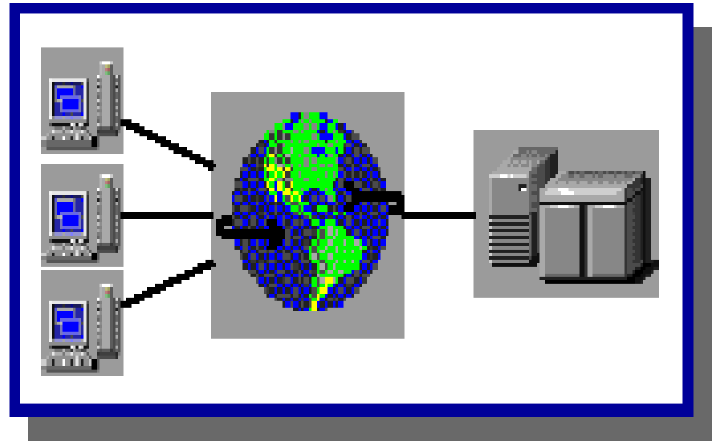
Model-based system development

Kurt Jensen
Aarhus University, Denmark

Presentation at the Carl Adam Petri Memorial Symposium, Berlin, February 4, 2011

Concurrent systems are very important





- A modern car contains a large number of micro processors connected via a dedicated network.
- The Internet with WWW and a lot of other distributed applications have an enormous impact on our daily life.

Concurrent systems are difficult to design

- They possess non-determinism.

- The execution may proceed in many different ways, e.g. depending on:
  - Whether messages are lost during transmission.
  - The scheduling of the individual processes.
  - The time at which input is received from the environment.

- Concurrent systems have an astronomical number of possible executions.
  - It is easy for the designer to miss important interaction patterns.
  - This may lead to gaps or malfunctions in the system design.

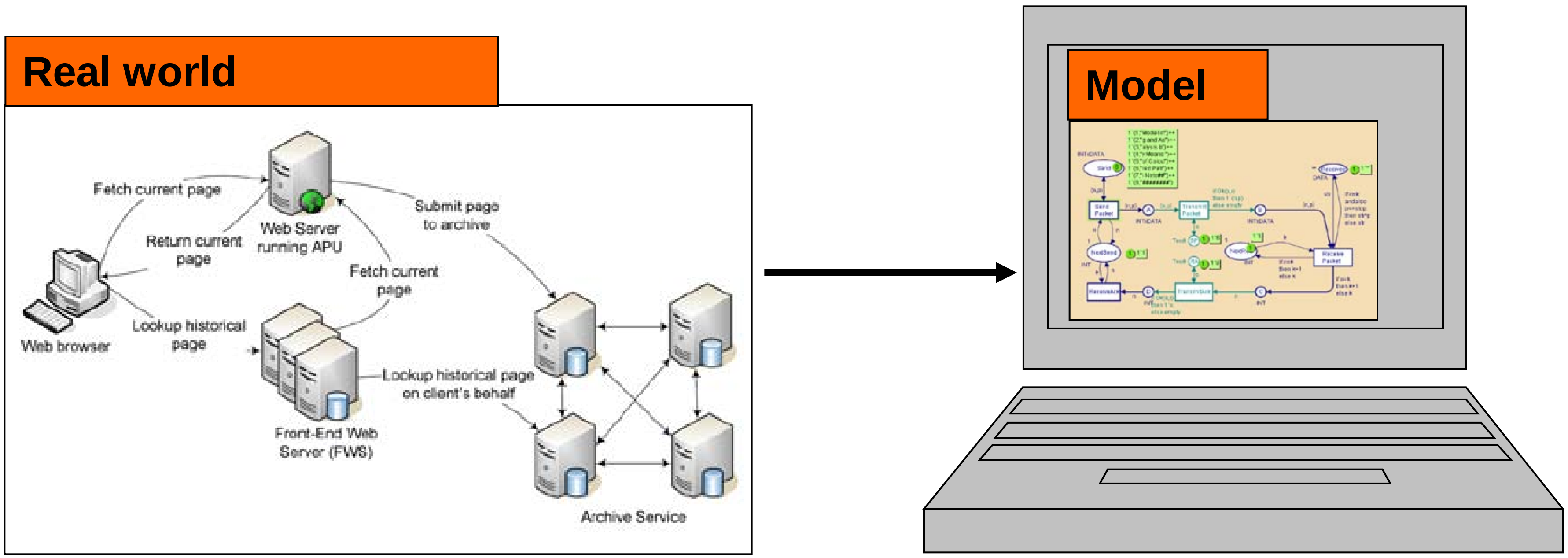
Concurrent systems are often critical

- For many concurrent systems it is essential that they work correctly from the very beginning:
  - Nuclear power-plants.
  - Aircraft control systems.
  - Hospital life support equipment.
  - Computer networks.
  - Banking systems.

- To cope with the complexity of modern concurrent systems, it is crucial to provide methods that enable debugging and testing of central parts of the system designs prior to implementation and deployment.

Model-based system development

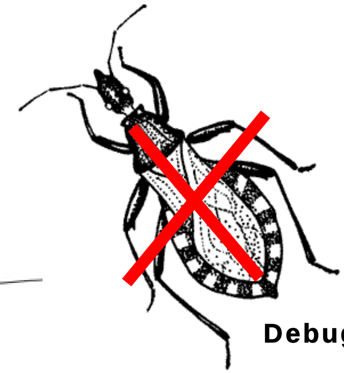
- One way to design a complex concurrent system is to build a model.
- This is an abstract mathematical representation which can be manipulated by means of a computer tool.
- We use the model to investigate how the system will behave and the properties it will have.



Why do we make models?

We make models to:

- gain insight in the system.
- get ideas to improve the design.
- locate design errors and other shortcomings.



Debugging

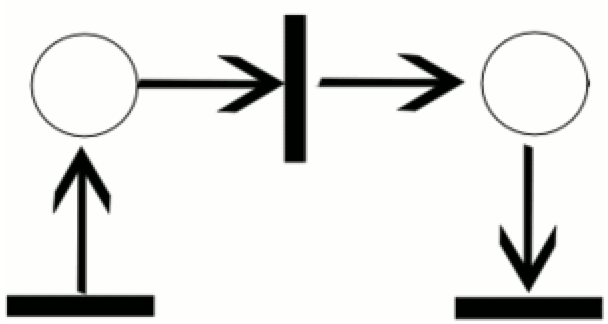
Correctness proof

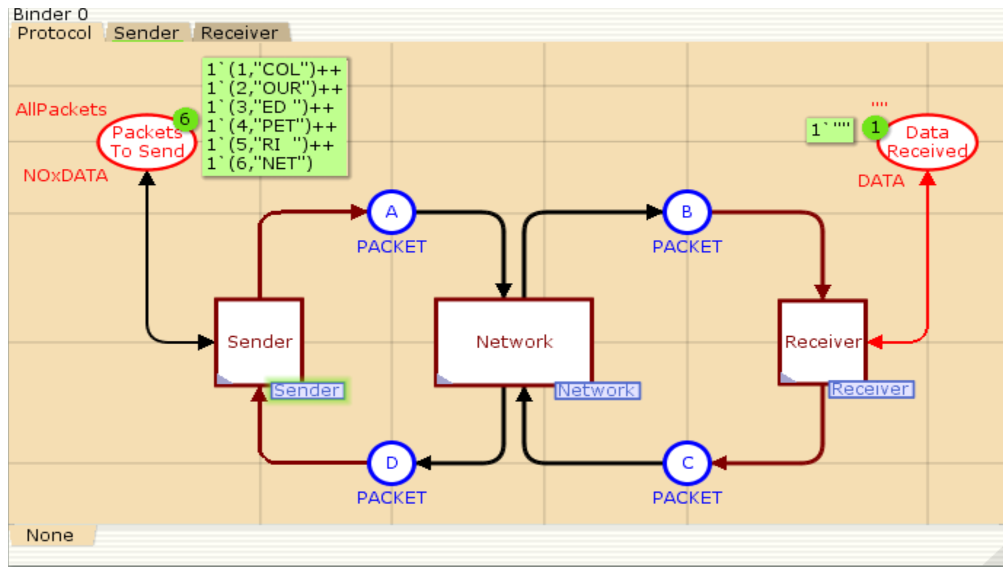
Models help us to:

- ensure completeness of the design.
- improve the correctness of the design.

Petri Net models

- In this talk we will – for obvious reasons – focus on models which are built by means of Petri Nets.
- Petri Net models have turned out to be excellent to investigate concurrent systems, where we typically have:
  - a number of processes which
  - communicate and synchronise with each other sharing
  - a set of common resources.





Remarkable foresight

- The importance of concurrency is quite obvious for us today.

- However, remember that Carl Adam Petri started his scientific work around 1960.

- That was far ahead of the time where:
  - distributed systems were invented,
  - computers started to have parallel processes.

- At that time programs and processing were considered to be sequential and deterministic.

- Hence, it was extremely visionary of Carl Adam Petri to predict the importance of being able to understand and characterise the basic concepts of concurrency.

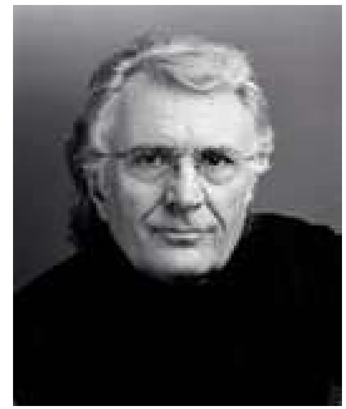
High-level Petri Nets

- In principle, we could build our models by means of elementary nets or place/transition nets.
- However, computer systems (and many other kinds of systems) contain complex data which influences the behaviour of the system.
- This means that we need a modelling language which makes it possible to represent data in an adequate and succinct way.

- This is offered by high-level Petri Nets.

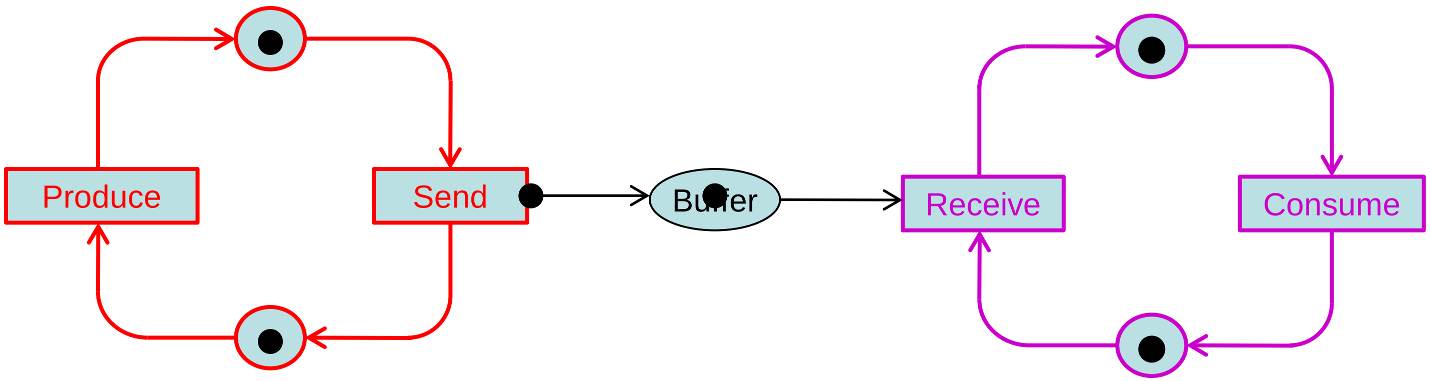
Birth of high-level Petri Nets

- The first successful type of high-level Petri Nets was called Predicate/Transition Nets (Pr/T-nets).
- This net class was developed by Hartmann Genrich and Kurt Lautenbach from Petri's group at Schloss Birlinghoven.

- The first paper was presented at a conference on Semantics of Concurrent Computation in 1979.
- The work was partly based on earlier work:
  - Transition nets with coloured tokens, Kurt Lautenbach & M. Schiffers 1977.
  - Transition nets with complex conditions, Robert Shapiro 1979.

Producer   Consumer



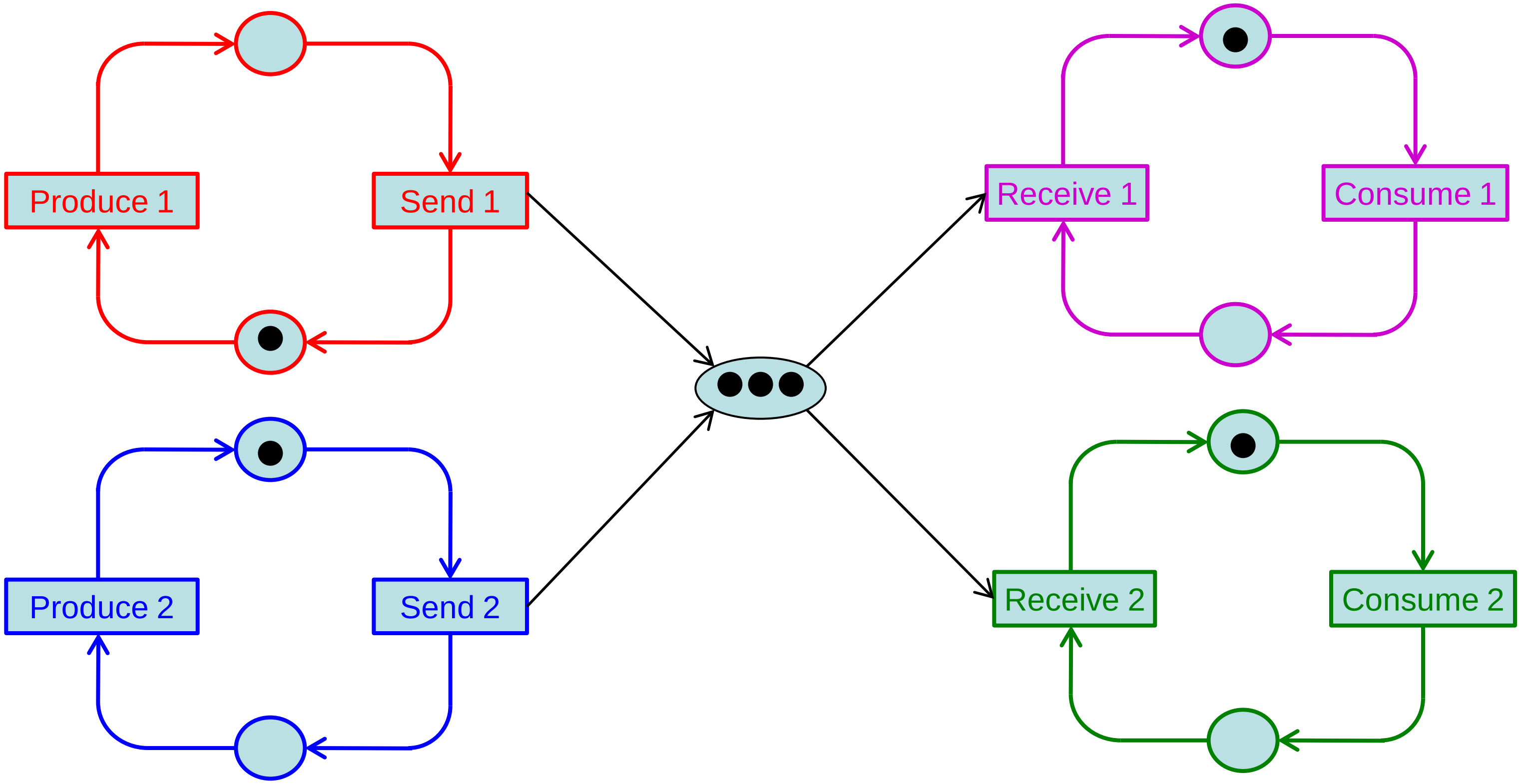
- Producer: Produces data which are Sent to a Buffer.
- Consumer: Receives data from the Buffer and Consumes them.

- Solid dots ● called tokens are used to represent:
  - The states of the producer and the consumer.
  - The data packets at the buffer.

- There is only one kind of tokens.

More producers and consumers

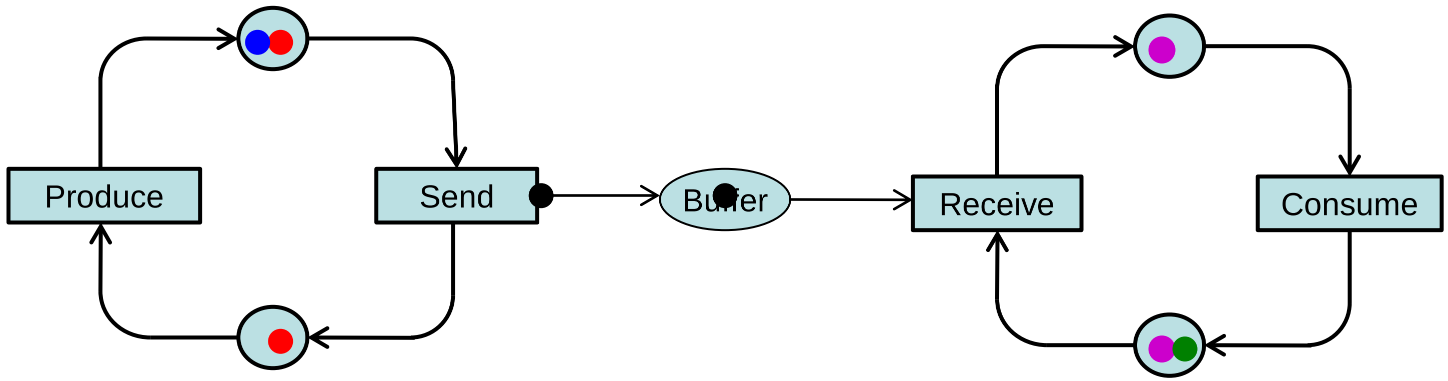
- Now we have two producers and two consumers communicating via a single buffer.



- What happens if we have 12 producers and 8 consumers?

Solution: use different tokens

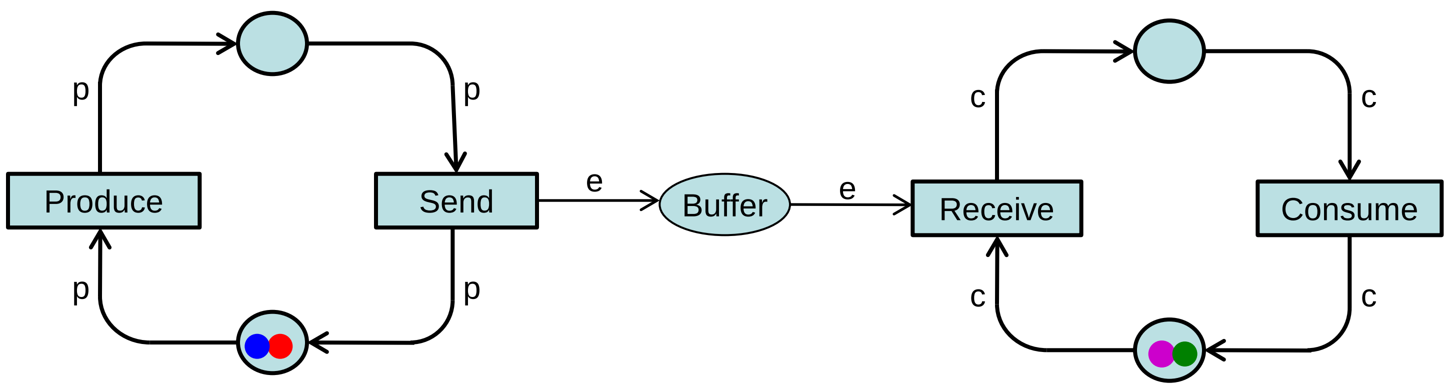
- Instead of having different nets for the producers, we use a single net with different kinds of tokens.
- Analogously for the consumers.



- It can be proved that this model has the same behaviour as the previous one (in which we duplicated the subnets for producers and consumers).
- Now we can easily change the number of producers and consumers (by adding or removing tokens).

Main idea behind Predicate/Transition Nets

- Tokens can be distinguished from each other.
- They are said to be coloured – in contrast to the tokens of place/ transition nets which are indistinguishable and drawn as black dots.



- Transitions can occur in many different ways – depending on the token colours of the available input tokens.
- Arc expressions and guards are used to specify enabling conditions and the effects of transition occurrences.

Limitations of Predicate/Transition Nets

- The invention of token colours was a gigantic step forward – but it still had some limitations.

- There was only one class of tokens colours – represented by a set D.
- D and Cartesian products of D (D x D, D x D x D, etc.) had to be used to represent all different kinds of data in the system, e.g.:
  - identity of producer processes.
  - identity of consumer processes.
  - detailed data in the packets.

More general colour sets

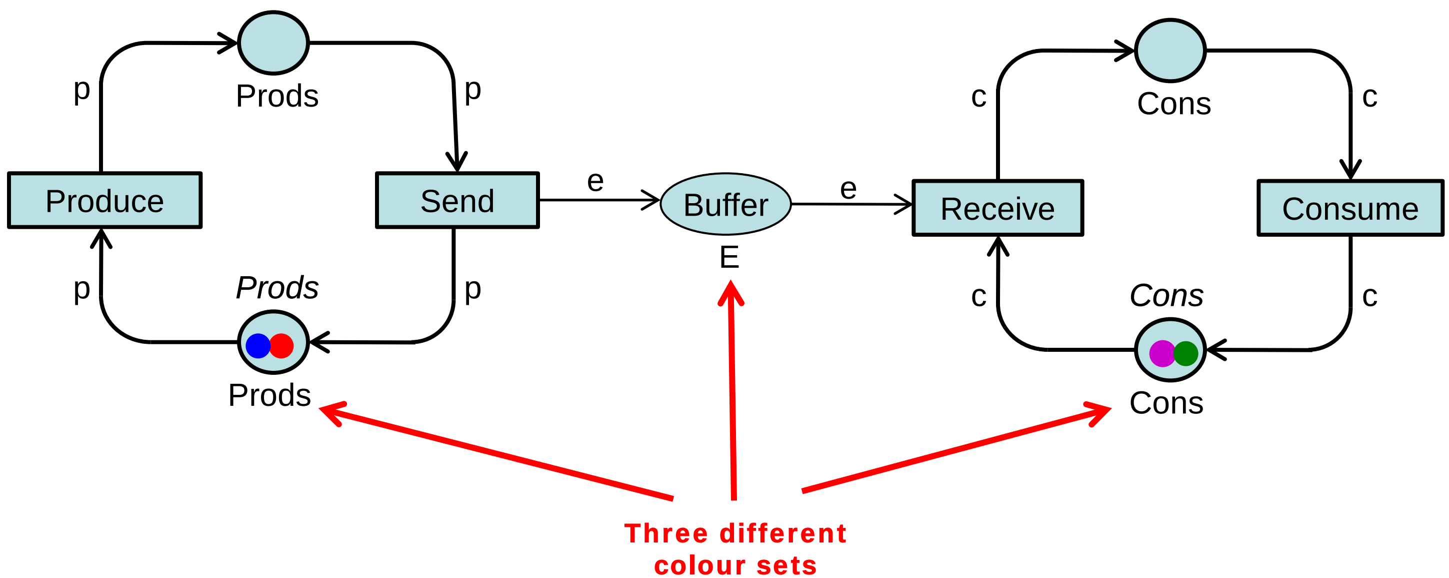
- The next step forwarded was achieved by the development of Coloured Petri Nets at Aarhus University, Denmark in 1979.
  - They allowed the use of a number of different sets of colours.
  - Hence it became possible, e.g., to distinguish between token colours used to model producers, consumers and packet data.

- It gradually turned out that it was convenient to define the colour sets by means of data types known from programming languages, such as products, records, lists, enumerations, etc.
  - Tokens colours became structured (and hence much more powerful).
  - Type checking became possible (making it much easier to locate modelling errors).
  - Colour sets, arc expressions and guards could be specified by the syntax and semantics known from programming languages.

Coloured Petri Nets

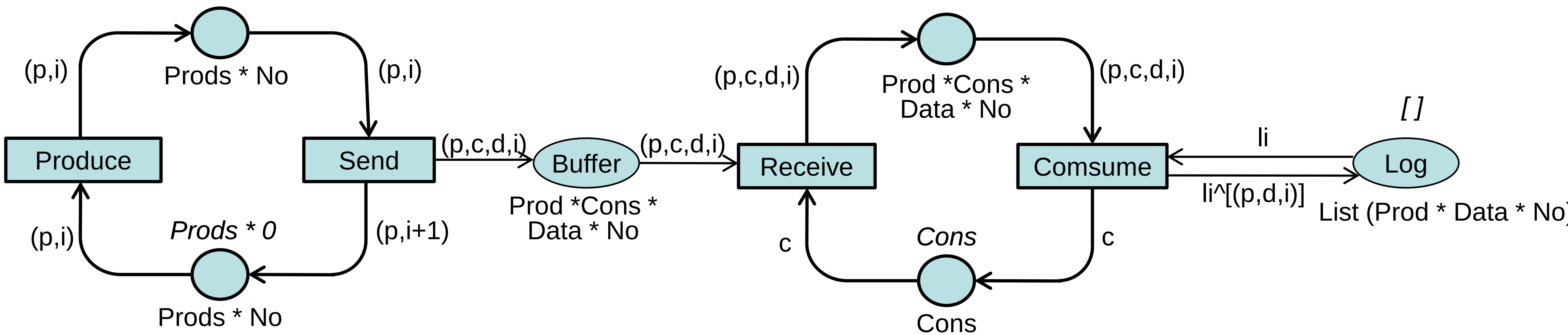
Main idea behind Coloured Petri Nets:

- Allow each place to have its own set of possible token colours.
- Use types to specify the colour sets.



Much more modelling power

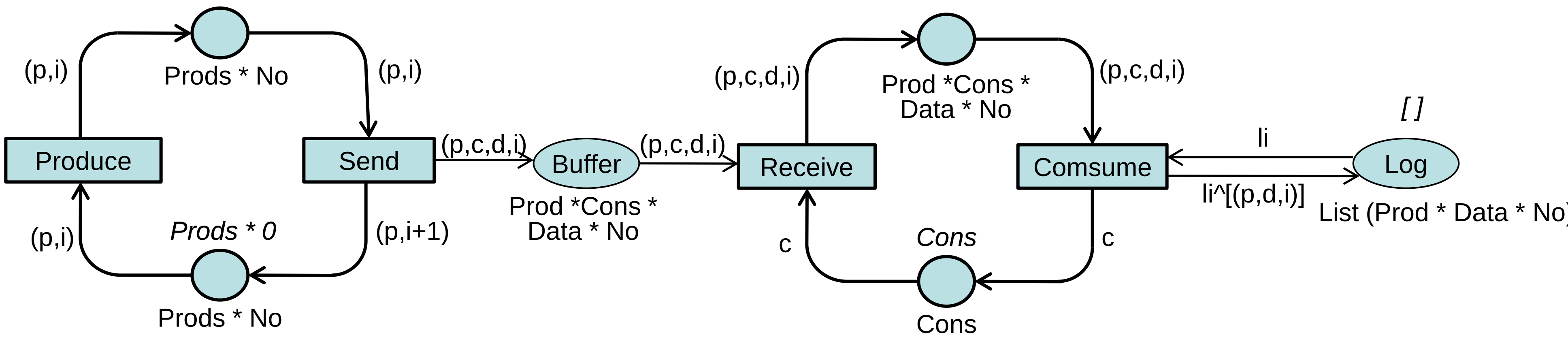
- Net structure and token positions specify the control flow.
- Token colours specify data values.



- Each message in the buffer contains four elements which specify:
  - the sending producer **p** and the receiving consumer **c**.
  - a text string to be transmitted as data **d**.
  - the sequence number of the packet **i**.
- The consumers keep a log with a list of all received messages.

Much more modelling power

- Net structure and token positions specify the control flow.
- Token colours specify data values.



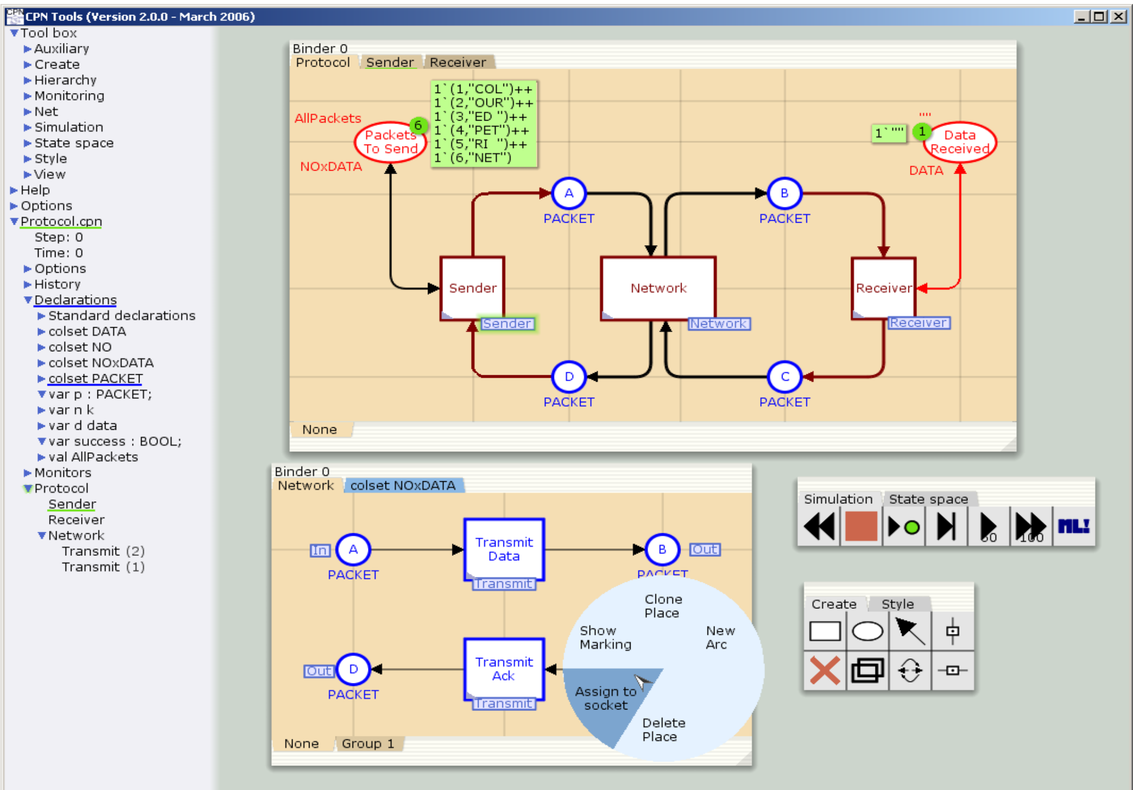
- In practice, it would be totally impossible to model all these details by means of elementary nets, place/transition nets or Predicate/Transition Nets.

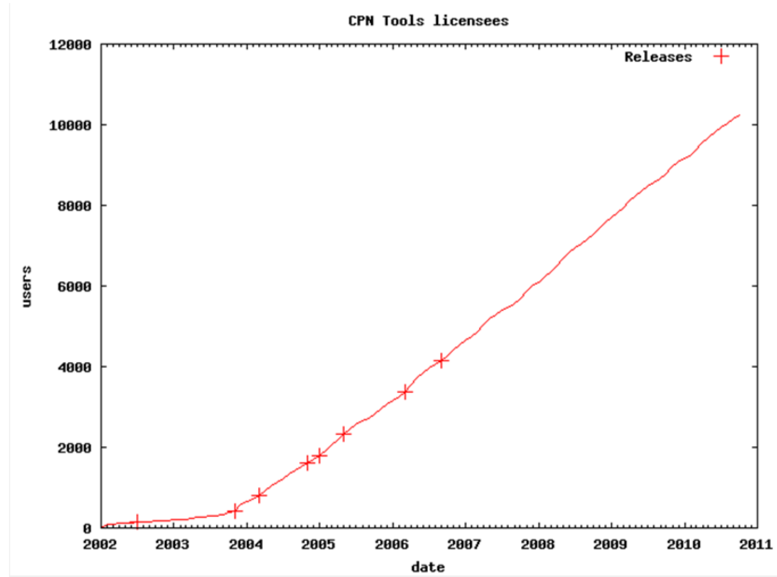
High-level Petri Nets

- The relationship between high-level Petri Nets and low-level Petri Nets is analogous to the relationship between a high-level programming language and assembly code.
- The high-level versions are obtained by adding types (token colours) and structuring facilitates (modules).
- In theory, the two levels have the same computational power.
  - Each high-level Petri Net can be translated into a behavioural equivalent low-level Petri Net (and vice versa).
  - This means that also high-level Petri Nets benefit from the work of Carl Adam Petri to establish the basic concepts of Petri Nets.
- In practice, the high-level languages have much more modelling power and hence they are much more convenient for human beings.

High-level Petri Nets have made a significant impact

- The most popular computer tool for Coloured Petri Nets have more than 10,000 licenses in nearly 150 countries.

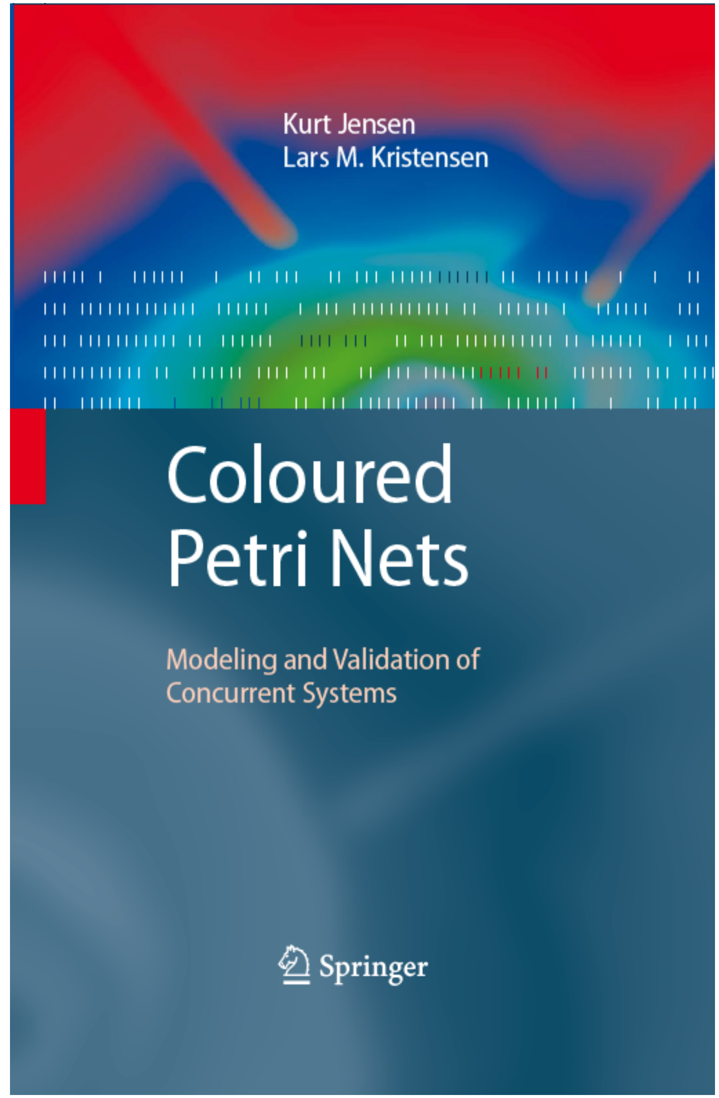




Impact (2)

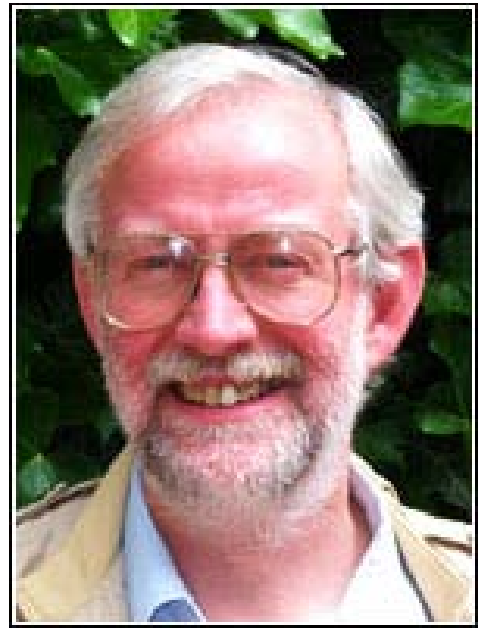
- The books and research papers defining Coloured Petri Nets have close to 10,000 citations.

Newest CPN book from 2009



Impact (3)

- High-level Petri Nets has become an international ISO/IEC standard.
  - Jonathan Billington: ISO/IEC 15909-1:2004, Software and System Engineering – High-level Petri Nets - Part 1: Concepts, Definitions and Graphical notation.



- High-level Petri Nets has been used in numerous industrial projects of which a considerable number have been documented in published peer-reviewed papers.
  - References to more than 100 papers can be found at http://cs.au.dk/cpnets/industrial-use/

Carl Adam Petri's contribution to high-level Petri Nets and computer tools for Petri Nets

- Carl Adam Petri was a theoretician and mathematician.
- However, he was extremely interested in practical applications and understood at a very early stage that these could only be done by building adequate tool support.
  - The first computer tool for Petri Nets was built in Carl Adam Petri's group at Schloss Birlinghoven by Robert Shapiro in 1982-83.
  - This tool became the ancestor and source of inspiration for all succeeding computer tools within the area.

Carl Adam Petri's impact on people

- Carl Adam Petri was a very positive and accommodating person.
- He has been a mentor and a spiritual farther for a lot of young students.
- He always had the time and patience to explain complicated mathematical concepts.

- Personally, I have benefitted enormously from his hospitality and open mindedness – both when I visited his group as a young PhD student and when I met him as an established researcher within the Petri Net area.
- I am very grateful for having had the opportunity to know Carl Adam Petri. Without that my professional career is likely to have taken a very different and probably less exciting direction.

I want to finish this talk with a very large and heartfelt thank-you to Carl Adam Petri

and a thank-you to all of you for your attention.