High-level Petri nets

Introduction to high-level Petri nets

Petri nets provide a framework for the design, specification, validation, and verification of discrete event systems. Petri nets have a wide range of application areas, and many Petri net projects have been carried out in industry, e.g., in the areas of communication protocols, operating systems, hardware design, business process re-engineering. Generally speaking the Petri nets can be classified into two classes: ordinary Petri nets and high-level Petri nets. High-level Petri nets have been used for modeling and simulation engineering and scientific problems with complex structures. Particularly, timed, stochastic and fuzzy Petri nets have been used for modeling, analysis, and simulation in intelligent task planning, dynamic knowledge representation, managing symbolic and numerical information, and artificial intelligence. Colored Petri nets (CP-nets or CPNs) represent another class of popular high-level Petri nets. The main advantage of CP-nets over the other Petri nets is that they can involve tokens with attached data values called colors, so that each token is associated with attributes. These attributes are called colors.

Creating large, intricate CPNs can be a cumbersome task. But similar to modular programming, the construction of CPN can be broken into smaller pieces by utilizing the facilities individually. Hierarchical CP-nets are nets with multiple layers of detail. You can have a somewhat simplified net that gives a broad overview of the system you are modeling. By substituting the object at the top-level net with more detailed subnets, you can bring more and more details into the model. Effective CPN-modeling requires the ability to distribute a CPN across multiple pages, so as to divide it into modules small enough to keep track.

Introduction to Colored Petri nets

Coloured Petri Nets (CP-nets or CPNs) is a modelling language developed for systems in which communication, synchronisation and resource sharing play an important role. CP-nets combine the strengths of ordinary Petri nets with the strengths of a high-level programming language. Petri nets provide the primitives for process interaction, while the programming language provides the primitives for the definition of data types and the manipulations of data values.

CP-nets has an intuitive, graphical representation which is appealing to human beings. A CPN model consists of a set of modules (pages) which each contains a network of places, transitions and arcs. The modules interact with each other through a set of well-defined interfaces, in a similar way as known from many modern programming languages. The graphical representation makes it easy to see the basic structure of a complex CPN model, i.e., understand how the individual processes interact with each other.

CP-nets also has a formal, mathematical representation with a well-defined syntax and semantics. This representation is the foundation for the definition of the different behavioural properties and the analysis methods. Without the mathematical representation it would have been totally impossible to develop a sound and powerful CPN language. However, for the practical use of CP-nets and their tools, it suffices to have an intuitive understanding of the syntax and semantics. This is analogous to programming languages which are successfully applied by users who are not familiar with the formal, mathematical definitions of the languages.

CPN models can be made with or without explicit reference to time. Untimed CPN models are usually used to validate the functional/logical correctness of a system, while timed CPN models are used to evaluate the performance of the system. There are many other languages
which can be used to investigate the functional/logical correctness of a system or the performance of it. However, it is rather seldom to find modelling languages that are well-suited for both kinds of analysis.

CP-nets can be simulated interactively or automatically. In an interactive simulation the user is in control. It is possible to see the effects of the individual steps directly on the graphical representation of the CP-net. This means that the user can investigate the different states and choose between the enabled transitions. An interactive simulation is similar to single-step debugging. It provides a way to "walk through" a CPN model, investigating different scenarios and checking whether the model works as expected. This is in contrast to many off-the-shelf simulation packages which often act as black boxes, where the user can define inputs and inspect the results, but otherwise have very little possibility to understand and validate the models on which the simulations build. It is our experience that the insight and detailed knowledge of a system, which the users gain during the development and validation of a simulation model, is often as important as the results that the users get from the actual simulation runs.

Automatic simulations are similar to program executions. Now the purpose is to be able to execute the CPN models as fast and efficient as possible, without detailed human interaction and inspection. However, the user still needs to interpret the simulation results. For this purpose it is often suitable to use animated, graphical representations providing an abstract, application-specific view of the current state and activities in the system.

CP-nets also offers more formal verification methods, known as state space analysis and invariant analysis. In this way it is possible to prove, in the mathematical sense of the word, that a system has a certain set of behavioural properties. However, industrial systems are often so complex that it is impossible or at least very expensive to make a full proof of system correctness. Hence, the formal verification methods should be seen as a complement to the more informal validation by means of simulation. The use of formal verification is often restricted to the most important subsystems or the most important aspects of a complex system.

**Why use CP-nets**

CP-nets are used for three different - but closely related - purposes. First of all, a CP-net model is a description of the modelled system, and it can be used as a specification (of a system to be built) or as a presentation (of a system to be explained to other people, or ourselves). By creating a model we can investigate a new system before we construct it. This is an obvious advantage, in particular for systems where design errors may jeopardise security or be expensive to correct. Secondly, the behaviour of a CPN model can be analysed, either by means of simulation (which is equivalent to program execution and program debugging) or by means of more formal analysis methods (which are equivalent to program verification). Finally, it should be understood that the process of creating the description and performing the analysis usually gives the modeller a dramatically improved understanding of the modelled system - and it is often the case that this is more valid than the description and the analysis results themselves. Below, we give a brief description of some of the main qualities of CP-nets.

1. **CP-nets have a graphical representation.** The graphical form is intuitively very appealing. It is extremely easy to understand and grasp - even for people who are not very familiar with the details of CP-nets. This is due to the fact that CPN diagrams resemble many of the informal drawings which designers and engineers make while they construct and analyse a system
2. **CP-nets have a well-defined semantics which unambiguously defines the behaviour of each CP-net.** The presence of the semantics makes it possible to implement simulators for CP-nets, and it also forms the foundation for the formal analysis methods.
3. **CP-nets are very general and can be used to describe a large variety of different systems.** The applications of CP-nets range from informal systems (such as the description of work processes) to formal systems (such as communication protocols). They also range from software systems (such as distributed algorithms) to hardware systems (such as VLSI chips).
4. **CP-nets have very few, but powerful, primitives.** The definition of CP-nets is rather short and it builds upon standard concepts which many system modellers already know from
mathematics and programming languages. This means that it is relatively easy to learn to use CP-nets. However, the small number of primitives also means that it is easy to develop strong analysis methods.

5. **CP-nets have an explicit description of both states and actions.** This is in contrast to most system description languages which describe either the states or the actions - but not both. Using CP-nets, the reader may easily change the point of focus - from states to actions or vice versa.

6. **CP-nets have a semantics which builds upon true concurrency, instead of interleaving.** In an interleaving semantics it is impossible to have two actions in the same step, and thus concurrency only means that the actions can occur after each other, in any order. A true concurrency semantics is easier to work with, because it is closer to the way human beings think about concurrent actions.

7. **CP-nets offer hierarchical descriptions.** This means that we can construct a large CP-net by relating smaller CP-nets to each other, in a well-defined way. The hierarchy constructs of CP-nets play a role similar to that of subroutines, procedures and modules of programming languages. The existence of hierarchical CP-nets makes it possible to model large systems in a manageable and modular way.

8. **CP-nets integrate the description of control and synchronisation with the description of data manipulation.** This means that on a single sheet of paper it can be seen what the environment, enabling conditions and effects of an action are. Many other graphical description languages work with graphs which only describe the environment of an action - while the detailed behaviour is specified separately (often by means of unstructured prose).

9. **CP-nets can be extended with a time concept.** This means that it is possible to use the same modelling language for the specification/validation of functional/logical properties (such as absence of deadlocks) and performance properties (such as average waiting times). The basic idea behind the time extension is to introduce a global clock and to allow each token to carry a time stamp - indicating when it is ready to be consumed by a transition.

10. **CP-nets are stable towards minor changes of the modelled system.** This is proved by many practical experiences and it means that small modifications of the modelled system do not completely change the structure of the CP-net. In particular, it is possible to add a new sequential process without changing the net structure representing existing processes.

11. **CP-nets offer interactive simulations where the results are presented directly on the CPN diagram.** The simulation makes it possible to debug a large model while it is being constructed - analogously to a good programmer debugging the individual parts of a program as they are finished. The data values of the moving tokens can be inspected.

12. **CP-nets have a number of formal analysis methods by which properties of CP-nets can be proved.** The two most important analysis methods are known as occurrence graphs and place invariants.

13. **CP-nets have computer tools supporting their drawing, simulation and formal analysis.** This makes it possible to handle even large nets without drowning in details and without making trivial calculation errors. The existence of such computer tools is extremely important for the practical use of CP-nets.

**Getting started with CP-nets**

We will study CP-nets introducing Resource Allocation Problem. Download and read resourceallocation.pdf to understand how CP-nets can be adapted to the structure of the problem. We will detail this example in the class.

**Resources**

Download cpntools_setup.exe software for getting prepared to work with CPNTools software. This manual explains how you can create, edit, simulate and analyse colored Petri net with CPNTools software. CPNTools Home Page maintained by CPN Group at University of Aarhus is excellent source to obtain broad information on theoretical aspects and practical applications of CPNTools software. CPN Tools uses the CPN ML language for declarations and net inscriptions.
Basics of CPN ML language can be found on http://wiki.daimi.au.dk/cpntools-help/cpn_ml.wiki. I strongly encourage you to visit this web page! There is an alternative way to obtain concise knowledge on CPN ML. Go to my web page. On the left hand side you will find CpnMLAll. Download the file and read it!

I also recommend to my students to visit Petri Nets World. The purpose of the Petri Nets World is to provide a variety of online services for the international Petri Nets community. The services constitute, among other things, information on the International Conferences on Application and Theory of Petri Nets, mailing lists, bibliographies, tool databases, newsletters, and addresses. Finally, for complete overview of Petri nets tools database click on http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/complete_db.html.