PNtalk – An Experimental System Based on Object-Oriented Petri Nets

Milan Češka, Vladimír Janoušek, and Tomáš Vojnar

Department of Computer Science and Engineering, Technical University of Brno
Božetěchova 2, CZ–612 66 Brno, Czech Republic
e-mail: {ceska, janousek, vojnar}@dcse.fee.vutbr.cz

PNtalk is an experimental tool demonstrating one possible way of combining objects and Petri nets. Its style of object-orientation as well as the appearance of its integrated development environment have been inspired by Smalltalk.

1 The Underlying Object-Oriented Petri Nets

PNtalk is based upon the so-called Object-Oriented Petri Nets (OOPNs). The OOPN formalism is characterized by a Smalltalk-based object orientation enriched with concurrency and polymorphic transition execution, which allow message sending, waiting for and accepting responses, creating new objects, and performing primitive computations. The OOPN formalism works with objects inside nets and also with nets inside objects.

An OOPN consists of Petri nets organized in classes. Every class consists of an object net describing the internal activity of its objects and of a set of method nets describing how these objects respond to messages. The inheritance of classes is defined by the inheritance of object nets, together with sets of method nets.

A running OOPN consists of objects, which are interconnected by references bound to tokens stored in places. Using these references, the objects can communicate via message sending. Every object is an instance of some class and consists of an instance of its object net and currently running instances of its method nets, which have been created as responses to received messages.

2 The PNtalk System

The system called PNtalk is intended to be a computer support tool for the work with the OOPNs mentioned above. It can be split into two separate parts each consisting of several subsystems. We can call these two parts front-end and back-end. Subsystems in the front-end and back-end are interconnected via a communication subsystem. This makes it possible to implement them in different environments and let them work on different nodes in a computer network.

A user of the PNtalk system communicates with the front-end of the system. It consists of several particular tools controlled from the central panel called the PNtalk Launcher. We can start the following tools from here: browser, debugger, post-mortem debugger, and breakpoint manager. The PNtalk Launcher also contains an area which is used for displaying messages coming from the

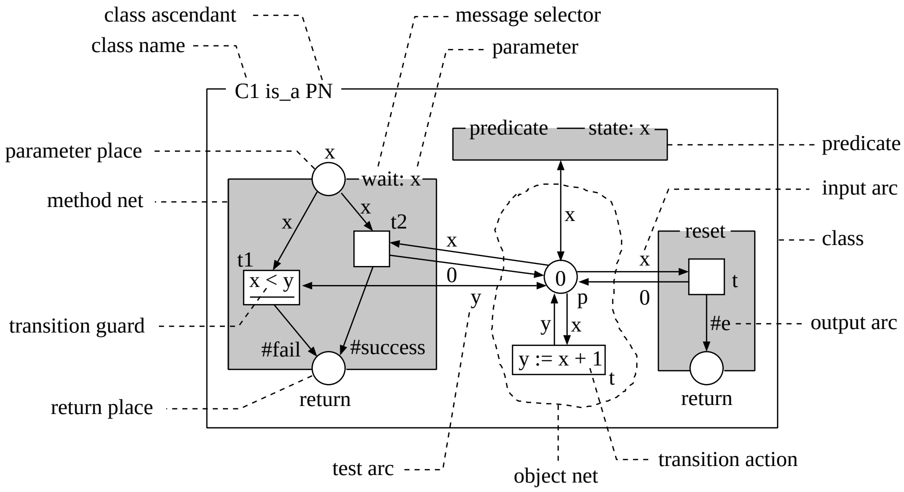

Fig. 1. A simple class demonstrating the notion of the OOPN formalism

back-end. These messages are either system messages or user-defined messages sent from transitions of the simulated OOPN.

The PNtalk browser is intended for creating, viewing, and editing OOPNs. It is a graphical tool, whose window has three different parts. The first one contains a list of classes, the second one a list of methods and constructors of the selected class. The last part displays the object net or the net of the selected method. The philosophy of this tool is similar to the Smalltalk browser. It organizes particular classes and their methods, and allows the users to concentrate individually on small parts of the system they are creating. However, the users should also have a general view of the system, which should be provided by another tool working with object diagrams.

The PNtalk debugger provides a possibility to trace the behaviour of an OOPN. It displays a tree of processes of methods and object nets reflecting calling sequences. Each node of this tree has a corresponding net, in which enabled transitions are highlighted. A user can choose an enabled transition and one of its bindings, and fire it. Inspecting the marking of particular places is possible, too. There can be even more debuggers running at the same time and displaying different parts of the OOPN.

The second part of the PNtalk system, the back-end, includes the PNtalk compiler and interpreter. The interpreter can work in many various modes. It can automatically fire enabled transitions and then an OOPN behaves as a running program. On the other hand, users can let the system inform them which transitions are enabled and with what bindings, and then they themselves decide what steps should be performed.

An interesting feature of PNtalk is a possibility of combining Smalltalk and PNtalk objects. We can describe the main structure of a system, which we are creating, by Petri nets and the corresponding scientific computations and visualisation realize in Smalltalk.

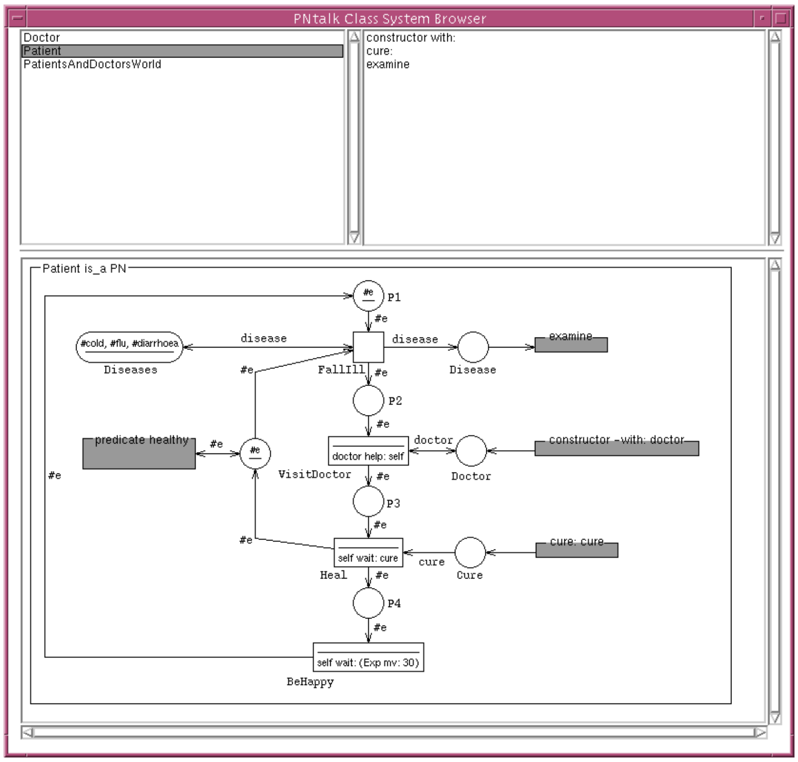

Fig. 2. PNtalk Browser – a tool for creating, displaying and modifying OOPNs

3 Extensions and Future Development

Up to now the first prototype version of PNtalk has been implemented. It is not intended for working with large models, but it can be used for getting acquainted with the OOPN formalism. To make PNtalk usable in praxis, it will be necessary to reimplement it to increase the stability of its GUI and to improve the efficiency of the simulator.

In addition to simulation, we are also working on methods of formal analysis of OOPNs. We have proposed a formal definition of OOPNs and of its state space. The notion of the state space has been defined in such a way that common

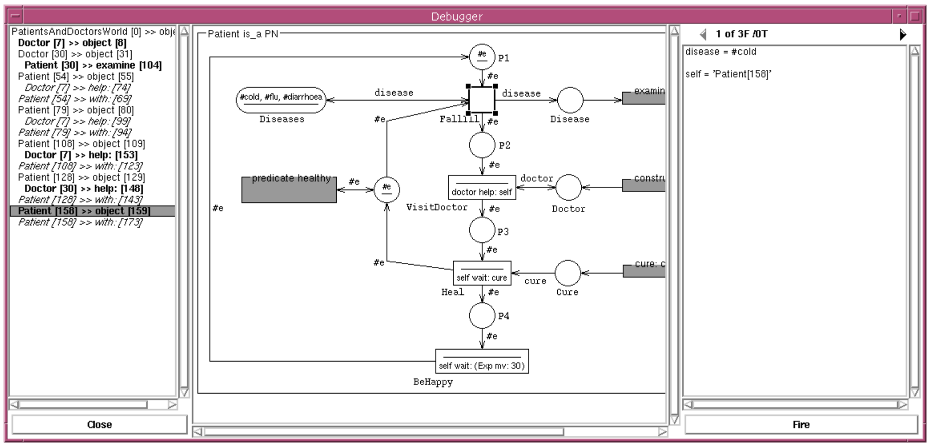

Fig. 3. PNtalk Debugger – a tool implementing a visualized token game

methods developed for the state space analysis of CPNs should be applicable also in the context of OOPNs. In future, we also want to examine the possibility of model checking over OOPNs. We would also like to try to exploit the modularity of OOPN based models for establishing some kind of modular analysis.

Recently we have extended the OOPN formalism with a notion of physical time, too. This is done by means of transition delays. Values of delays are specified by arbitrary functions. Therefore it is possible to use deterministic timing, interval timing, stochastic timing, as well as their various mixtures. Timed OOPNs can be analysed by means of simulation. Further, we would like to improve this situation by introducing more formal kinds of analysis for different restrictions of timed OOPNs containing e.g. only interval, only stochastic, or only deterministic delays.

Finally, the pure OOPN formalism has been enriched with a modified concept of subpages, which allow us to compose objects also in a static way.

References

[1] Češka, M., Janoušek, V., Vojnar, T.: *PNtalk — A Computarized Tool for Object-Oriented Petri Nets Modelling*, in proceedings of EUROCAST'97, Lecture Notes in Computer Science volume 1333, Springer-Verlag, 1997.

[2] Češka, M., Janoušek, V.: *A Formal Model for Object-Oriented Petri Nets Modeling*, Advances in Systems Science and Applications, An Official Journal of the International Institute for General Systems Studies, Special Issue, 1997.

[3] WWW pages of the Petri net research group at TU Brno (articles, software), `http://www.fee.vutbr.cz/~janousek/pntalk/pntalk.html`.