PNtalk – a Computerized Tool for Object Oriented Petri Nets Modelling

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

Abstract. The paper deals with a new formalism and a tool for concurrent systems modelling and prototyping. This formalism, Object Oriented Petri Net (OOPN), combines the descriptive power of Petri nets with the well known advantages of object oriented modelling techniques. In the paper there are presented the structure of system specifications by means of OOPNs, principles of their dynamic behaviour as well as some basic features of a computerized tool which supports practical use of OOPNs. This tool works with a language called PNtalk which is based on the OOPNs and the language Smalltalk.

1 Introduction

Nowadays there is clearly visible a movement in the field of new system design methodologies and computerized tools for systems analysis, prototyping, and implementation. These tools and methodologies often reflect paradigms and theoretical concepts from the System and Computer Science. Many examples of such a reflection can be recognized in current research projects as well as in some available tools for system engineering. They are connected to automata, different kinds of logic, grammars, neural nets etc., or to completely new concepts. The developed theories of the corresponding theoretical models can then extend and support applications of the resulting tools in the areas of system verification or optimization.

1.1 Petri Nets and Objects

During recent years several authors have been trying to combine the Petri nets formalism with object oriented modelling techniques in order to develop a formal environment for system specification and design which would preserve advantages of both the paradigms, i.e. natural ways of modelling parallelism, formal semantics and analysis methods provided by Petri nets and concepts of abstraction, encapsulation, reusability or inheritance provided by object orientation. Resulting formalisms can be characterized by the degree to which they support appearance of objects inside nets and nets inside objects.

For example the Thorn project [8] has lead to a formalism which allows us to bind C++ objects to tokens, but there is no possibility of having nets
inside objects here. The CodeSign project [14] deals with Petri nets in objects, but objects of classes described by Petri nets can be created only statically, i.e. before the start of execution. The LOOPN++ formalism [3] allows tokens to be bound both to C++ objects as well as to dynamically created objects described by particular Petri nets. However, there is no concept of methods described by Petri nets here.

In this article there is presented another approach to combining Petri nets and objects, which is somewhat related to the one described in [4]. This is because it also uses the RPC mechanism for objects communication. Apart from that, the developed formal model, Object Oriented Petri Net (OOPN), is based on the concept of net invocation [1]. The OOPN formalism is used as a base of the tool and language PNtalk, which are intended for executable formal specifications in the Smalltalk environment. It works with objects inside nets and also with nets inside objects. These nets describe independent objects activities as well as methods as reactions to messages sent from outside of objects.

When describing the OOPN formalism, we assume the reader to be familiar with basic concepts and notions of object oriented systems and Petri net models. The reader should also bear in mind that the description presented here is mostly informal.

1.2 Fundamental Principles of OOPNs and PNtalk

The proposed OOPN formalism is characterized by a Smalltalk-based object orientation enriched by concurrency and polymorphic transition execution which allow message sending, waiting for and accepting responses, creating new objects, and performing primitive computations.

OOPNs are based on viewing objects as active servers which offer reentrant services to other objects. Services provided by objects as well as independent activities of objects are described by Petri nets – services by method nets, object activities by an object nets. Tokens in nets represent references to objects.

An OOPN consists of Petri nets organized in classes. Every class consists of an object net describing the internal activity of objects of this class and a set of method nets describing how these objects respond to messages. All method nets share access to the object net (places of object net are accessible by transitions of method nets). Each method net is reentrant and has parameter places and a return place. Class inheritance is defined by the inheritance of object nets, together with sets of method nets.

Each object is an instance of some class and consists of an instance of its object net and currently running instances of its method nets. When an object receives a message, a new instance of the corresponding method net is created, parameters are put into the parameter places and the instance of the method net is executed concurrently with all other net instances until the return place receives a token. Then the value of the token in the return place is passed to the message sender, and the instance of the method net is deleted. Message sendings and object creations are specified as actions attached to transitions. It is a polymorphic modification of the above mentioned invocation – a message