Interactive Approach to Coloured Petri Nets Teaching

Štefan Korečko

Department of Computers and Informatics,
Faculty of Electrical Engineering and Informatics,
Technical University of Košice, Letná 9, 041 20 Košice, Slovakia
stefan.korecko@tuke.sk

Abstract. Formal methods belong to the techniques that, when used appropriately, can significantly contribute to the correctness of a software or hardware system under development. One of the suitable methods for systems with concurrent or non-deterministic behaviour is the Coloured Petri Nets modelling language. In this paper a teaching activity aimed at an explanation of the basic principles of the language and some of its functional programming-related features is described. The activity duration has been two and half hours and it involved an interactive model building with active audience participation.

1 Introduction

Considering the increasing dependency of the contemporary human society on computer systems, their correctness should be of utmost importance. And one of the approaches that can significantly contribute to the correctness is a utilization of formal methods during the software and hardware development. A formal method is a mathematically-based technique, which provides a formal language with unambiguously defined syntax and semantics and an apparatus, which allows performing verification, development and simulation tasks with system specifications, written in the language. One of the significant members of the formal methods family is the Coloured Petri Nets (CPN) modelling language. CPN [4, 3] combine the Petri nets formalism [1] with a functional language to handle data manipulation and decision procedures. The functional language is called CPN ML and it is a slightly modified version of Standard ML [2, 5]. The CPN language and corresponding specification, verification and simulation tasks are supported by the CPN Tools [6] software.

For more than a decade, CPN are a part of undergraduate courses related to formal methods, modeling and simulation at the home institution of the author. One of the methods, applied by the author when explaining CPN concepts is an interactive approach with an active audience participation. Here, the audience picks out the domain and process for which a CPN model will be designed and helps to create its selected parts. The experience from a particular implementation of this approach in a training activity for university teachers is described in the rest of this paper.

2 Training Activity with Interactive CPN Model Creation

The training activity was organized for about 10 participants, who were university teachers with certain functional languages background. The participants had limited to no previous knowledge of CPN. The total duration of the activity was about 2.5 hours, excluding breaks, and it was split into three phases.

The first phase took about 30 minutes and explained the basic principles of CPN. Namely, that CPN have a graphical form, a bipartite graph with two types of vertices: places, drawn as ellipses and transitions, drawn as rectangles. The places hold tokens, which represent a state of the net and the transitions can be understood as events, which change the state by consuming existing tokens and creating new ones.

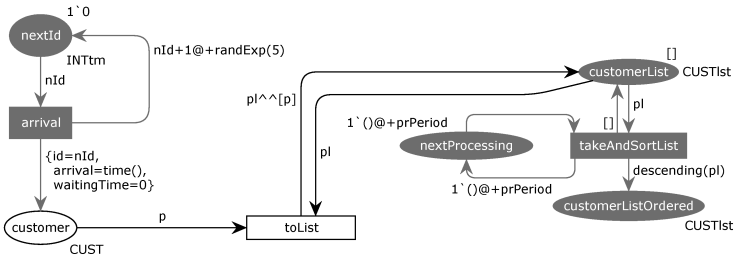

Fig. 1. The starter CPN model given to the activity participants

The second and third phase were devoted to a creation of a CPN model. As one of the goals of the activity was to show how some more advanced Standard ML concepts, namely structures and functors, can be used in CPN models, the participants had been given a starter CPN model, which already used the concepts, before the second phase begun. The starter model is shown in Fig. 1. The part consisting of the nodes `nextId`, `arrival` and `customer` represents an arrival of customers, which arrive one by one to be served. The serving itself is not presented in the starter model. Instead, there is the transition `toList`, which takes a token from `customer` and adds its value to a list, held in the place `customerList`. The transition `takeAndSortList` is fired at regular intervals, defined by the value `prPeriod`. Each firing of `takeAndSortList` empties the list in `customerList`, sorts its content and stores the ordered version in `customerListOrdered`. The place `nextProcessing` is auxiliary and ensures that `takeAndSortList` is fired only at the regular intervals. The sorting is provided by a function called `descending`, which implements the Quicksort algorithm. The function utilizes Standard ML structures and functors.

For the serving part, the activity participants decided to model a coffee vending machine. During the second phase they participated on a creation of a CPN

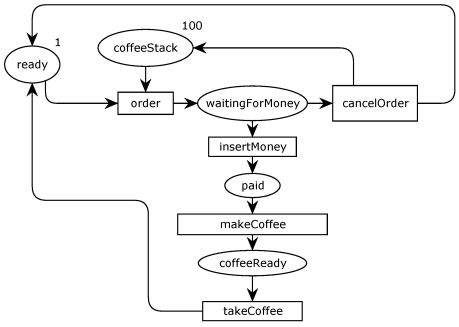

Fig. 2. CPN model of the serving part, created interactively during the second phase

model that captures the basic operation of the machine. The model is shown in Fig. 2. In its initial state the machine is ready to serve a customer (one token in the place `ready`) and is filled with 100 coffee doses (100 tokens in `coffeeStack`). The serving starts with a customer ordering a coffee by a firing of the transition `order`. Then the machine waits for the customer's next step (a token in `waitingForMoney`). The customer can insert money (by firing `insertMoney`) or cancel the order (by firing `cancelOrder`). The cancellation returns the machine to the "ready" state. If the money is inserted, the machine prepares the coffee (by firing `makeCoffee`). Finally, by a firing of `takeCoffee`, the customer takes the prepared coffee and the machine returns to the "ready" state.

After the second phase there was about 70 minutes long break. During the break the lecturer connected the model from the phase 2 to the parts of the starter model and added vertices and arcs describing the customer behaviour. He also corrected some inconsistencies in the model, pointed out by one of the participants. The resulting, final, CPN model can be seen in Fig. 3. The vertices taken from the starter model (Fig. 1) without any change are rendered in grey. The place `customer` is replaced by `customerQueue`, which holds a token with a list of values, representing a queue of customers waiting for the machine. Instead of the transition `toList` there is a serving part, created from the result of the second phase (Fig. 2). The serving part of the final model differs from Fig. 2 in three key aspects:

- The tokens carry information about the customer being served and arc expressions define durations of corresponding actions.
- Inconsistencies regarding the role of places and transitions are corrected. Now, all the transitions represent instantaneous events. For example, the transition `makeCoffee` from Fig. 2 is replaced by the place `makingCoffee` and the transition `takeCoffee` is replaced by the vertices `startTakingCoffee`, `takingCoffee` and `finishTakingCoffee`.
- Actions and states of the customers and the machine are modelled separately. The vertices `customerQueue`, `insertingDecidingEtc`, `waitingForCoffee`,

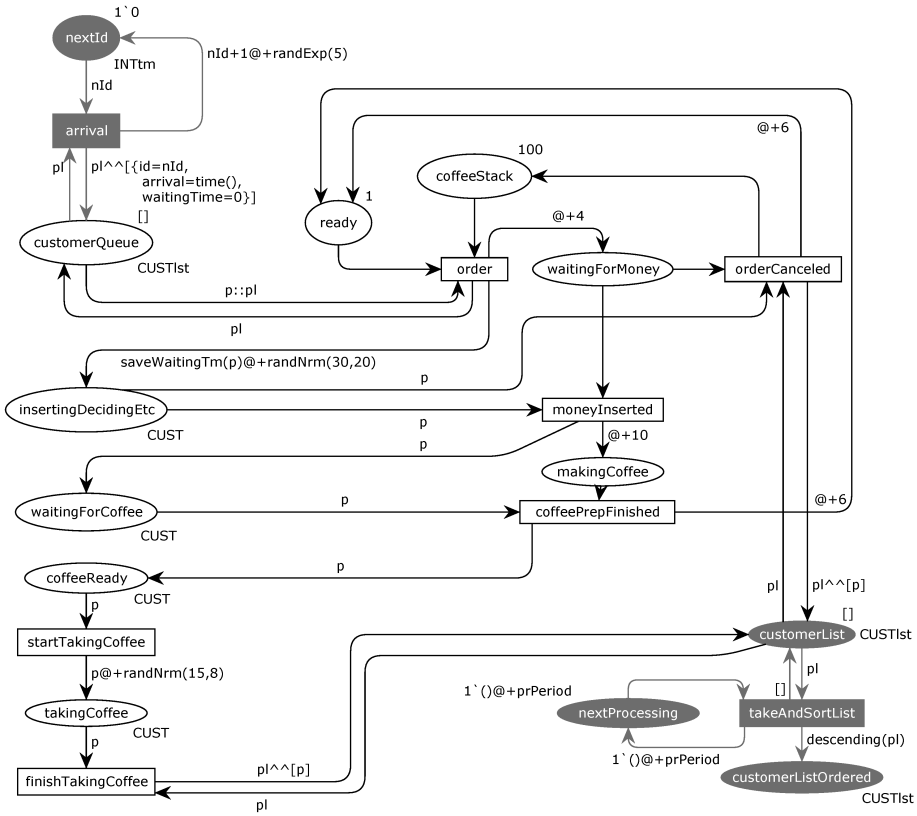

Fig. 3. The final form of the coffee machine CPN model

`coffeeReady`, `startTakingCoffee`, `takingCoffee` and `finishTakingCoffee` belong to the customers while the rest of the serving part represents the machine or both parties.

The third phase of the training activity has been devoted to the explanation of the final model and a discussion about the place of such models in the development of correct computer systems. It took about 30 minutes.

3 Conclusion

The interactive training activity, presented here, is suitable for short, intensive courses, which often take place during summer schools or other similar teaching events. The described test run of the activity revealed that the original time donation, which was 2 hours, was not sufficient. Therefore the third phase has been needed, where the lecturer presented the final model. Considering the time needed to construct the final model by the lecturer, it will require another at least two hours to perform the whole model creation process interactively with the auditory. All the CPN models presented or mentioned here can be obtained by request from the author.

Acknowledgement

This paper is part of the Intellectual Output O2 of the Erasmus+ Key Action 2 (Strategic partnership for higher education) project No. 2017-1-SK01-KA203-035402: "Focusing Education on Composability, Comprehensibility and Correctness of Working Software". The information and views set out in this paper are those of the author(s) and do not necessarily reflect the official opinion of the European Union. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use which may be made of the information contained therein.

References

1. Desel, J., Reisig, W.: Place/transition petri nets. In: Reisig, W., Rozenberg, G. (eds.) Lectures on Petri Nets I: Basic Models, Lecture Notes in Computer Science, vol. 1491, pp. 122–173. Springer Berlin Heidelberg. DOI: 10.1007/3-540-65306-6 (1998)
2. Harper, R.: Programming in Standard ML. Carnegie Mellon University (2011), `http://www.cs.cmu.edu/~rwh/smlbook/book.pdf`
3. Jensen, K.: An introduction to the theoretical aspects of coloured petri nets. In: A Decade of Concurrency, Reflections and Perspectives, REX School/Symposium. pp. 230–272. Springer-Verlag, London, UK. DOI: https://doi.org/10.1007/3-540-58043-3_21 (1994)
4. Jensen, K., Kristensen, L.M.: Coloured Petri Nets: Modelling and Validation of Concurrent Systems. Springer. DOI: 10.1007/b95112 (2009)
5. Milner, R., Tofte, M., Macqueen, D.: The Definition of Standard ML. MIT Press, Cambridge, MA, USA (1997), `http://sml-family.org/sml97-defn.pdf`
6. Cpn tools homepage (2018), `http://cpntools.org/`