# Leveraging the Pedagogical Potential of Tile-Based Games for Teaching Petri Net Modeling, the Sokoban Case

João-Paulo Barros<sup>1,3,\*</sup>, Luis Gomes<sup>2,3</sup>

<sup>1</sup>Polytechnic Institute of Beja, Beja, Portugal

<sup>2</sup>NOVA University Lisbon, School of Science and Technology, Portugal <sup>3</sup>Center of Technology and Systems (CTS), UNINOVA, Portugal

<sup>3</sup>Intelligent Systems Associate LAboratory (LASI), Portugal

#### Abstract

Students face unfamiliar abstract concepts when learning about formal methods and Petri net modeling. These concepts are challenging for students to understand fully and for teachers to design captivating modeling exercises that align with the intended learning outcomes. The proposed exercises should provide significant opportunities to apply abstract concepts in familiar domains, bridging the gap between the concrete and the abstract. Board games offer a range of modeling problems in an area that many students know. This paper introduces a model for Sokoban, a classic computer-based, one-player puzzle game. After, a sequence of possible Petri net modeling exercises focused on the game movement rules and model composition are proposed. The complete model is created by composing multiple instances of the previously defined models acting as modules. The complete model can then be used to verify game termination and obtain the net step sequence leading to the intended final net marking.

#### Keywords

Board games, Modeling, Formal methods, Education, Reachability graph, Petri nets

## 1. Introduction

Exercises are a core component of learning and assessment and must fulfill a dual role: they must align with the intended learning outcomes and be perceived by students as interesting and motivating.

Due to the well-known capacity to engage students, games are frequently used for pedagogical purposes (e.g., [1, 2]) and are often a way to introduce students to programming (e.g., [3]). Furthermore, games offer domains that are well-known by most students.

We propose using simple tile-based games, sometimes called grid-based games, as an adequate basis for teaching and learning Petri net modeling. More specifically, we argue that many of those games offer the following advantages:

- 1. Game domain, which should foster student engagement;
- 2. Huge variety, from very simple to very complex ones;
- 3. Graphical representations allowing a visual mapping between the physical board and the Petri net model structure;
- 4. A clear differentiation between the passive part (the board) and the active part (moving pieces and their movement rules, including restraints).

We present the Sokoban game as an illustrative example. This game, with its grid-based structure and movement rules, is a concrete application that exemplifies the above-listed advantages.

The paper is structured as follows. In the next section, we present the game Sokoban. In Section 3, we present a sequence of proposed exercises and the respective possible solutions focusing on the movement rules of the game and the subsequent composition that yields the complete model amenable to simulation or state-based exploration. Finally, we conclude with some proposals for future work.

*PeNGE 2024 - Petri Net games, examples and quizzes for education, contest and fun, Geneva, Switzerland, June 25, 2024* \* Corresponding author.

joao.barros@ipbeja.pt (J. Barros); lugo@fct.unl.pt (L. Gomes)

D 0000-0002-0097-9883 (J. Barros); 0000-0003-4299-8270 (L. Gomes)

<sup>© 0 2024</sup> Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

# 2. The Sokoban game

Sokoban has been named "the quintessential "block pushing" game " [4]. It is a single-player tile-based board game with simple rules. However, it can quickly become highly challenging. It is played on a "map" made of adjoining tiles. Each tile can be seen as a cell in a grid. The player moves in one of four directions: left to right, right to left, top to bottom, and bottom to top. The objective is for the player to push all the boxes around the grid to put them in the final marked locations. If a box is adjacent, the player can push the box to the next cell in the same direction. However, for that to happen, the ending box position must be free, which means no (other) box or "wall" can be in that cell. Each tile can be from one of three types:

Free cell A cell that can also contain the player or a box;

**End cell** A free cell that is marked as a final destination of any one box;

**Wall** A cell that cannot be occupied by the player or a box; these cells also serve to delimit the level board.

The Wikipedia webpage provides a simple and short introduction to the Sokoban game, illustrated with an animated gif [5]. Numerous Sokoban implementations and levels are readily available on the Internet. As a specially significant example, the website https://sokoban.info/ provides "more than 7500 levels". Fig. 1 illustrates one of the smallest and simplest levels available.



Figure 1: The game level "Mini Cosmos" at https://sokoban.info/?4.

Even Sokoban basic levels necessitate significant computational resources, as there are many branches and possible deadlocks, and a vast number of steps can quickly be needed to reach a solution [6]. More precisely, the game domain has been proven NP-Hard [7], and PSPACE-complete [8]. Sokoban has also been the object of several studies regarding possible algorithms (e.g., [9]). Here, our object is to use the modeling of the level (board) and movement rules as motivating exercises for model construction using Place/Transition Petri nets. The Petri net reachability graph can then be used to explore the existence of solutions and the respective paths to them.

The following section presents a sequence of possible student exercises. It corresponds to a step-wise and module-based approach for constructing a complete Sokoban level.

### 3. A Modular Petri net for a Sokoban level

Here, we present six sequential exercises toward a complete model for a Sokoban game level and its analysis using Place/Transition nets. First, we focus on the player movement, either by simply moving to an adjacent position or by pushing a box; secondly, we present how that model can be split into a passive and an active part; finally, using a textual description, we show how multiple instances of these two small modules can be composed to generate a level with all the possible movements in that level.

### 3.1. Modeling the player movement

When presenting the game to the students, the obvious challenge is to model the player's movements. Here, we split the player movement into two cases: (1) the end position is free; (2) the end position has a box.

#### 3.1.1. Free movement

Fig. 2a illustrates the cases that should be presented to the students as a first exercise. The player moves to a free position. The students have to consider the following aspects that are to be expressed in the model:

- 1. How the initial and free positions are modeled;
- 2. How do we know where the player is;
- 3. How the model moves the player from one position to the other;
- 4. How the model checks if the end position is free.





The student's first solution does not need to distinguish between an end position with a box or a wall. This distinction can be required for a second solution, as it should be clear that it will be needed for the case where the player has to move a box. Fig. 2b illustrates a solution with two places, for the begin and end positions, respectively, that need the end position to be free for the transition freeMove to be enabled. The transition firing models the player's movement.

### 3.1.2. Movement pushing a box

The cases to be presented to students in a second exercise are shown in Fig. 3a. The player moves to a position where a box is. The students have to consider the following aspects that are to be expressed in the model:

- 1. How the initial and end box position are modeled;
- 2. How to know where the player is;
- 3. How the model moves the player from one position to the other;
- 4. How the model moves the pushed box from one position to the other;
- 5. How the model moves the player and the pushed box synchronously;
- 6. How the model checks if the end position has a box;
- 7. How the model checks if the box's intended end position is free (from a box or a wall).



Figure 3: Box push: (a) game; (b) P/T model.

In this second exercise, it should be apparent from the beginning that the box and the player must change their positions. An additional complication is that a total of three positions must be considered, as the position after the box is also a possible restriction to the player's (and the box's) movement. The need to test and update the presence and the absence of a box presents a new difficulty that provides an opportunity to discuss the use of complementary places. In addition, it is helpful to discuss whether we need a complementary place for the wall and what information we need in each position.

Fig. 3b illustrates a solution. It is centered on the "push box" action, which is modeled by transition pushBox. It uses complementary places for boxes in the initial box position and in the position where the box is going to be pushed. It also checks whether or not there is a wall in the box end position. The chosen layout has all places related to the same cell (position) in the same row to highlight the number of positions involved and the data needed for each position state.

#### 3.1.3. Movement composition

A third, more straightforward exercise is to compose both movements into a single one. The solution is presented in Fig. 4.

#### 3.2. Position state and two-way movement

A fourth exercise is identifying all the state information needed for each position as a step toward modeling movement in both directions. Fig. 5a presents the solution presented in Fig. 4 with all the places needed to store each position's state. Their use in the two-way movement is illustrated in Fig. 5b.

#### 3.3. Creating the complete model using module composition

The complete model, for a whole level, requires multiple compositions. A two-step approach can be presented to students: (1) identify the modules to be composed; (2) express the multiple compositions.

#### 3.3.1. Modules identification

The modeling of the two movements in two opposite directions can be seen as the modeling of the "passage" between those two positions. The following exercise can be the identification of the model for each position (each cell in the grid). Next, both the "passage" and the "cell" should be transformed into composable modules by identifying the "interface" nodes. The "interface" nodes are the ones that will



**Figure 4:** Model for both movements of the player from position 1 to position 2: free move and move pushing a box.



Figure 5: Player movements: (a) from position 1 to position 2 (b) from position 2 to position 1.

be fused (merged) with nodes in other modules. It seems worth discussing the dual nature of "passages," which encapsulate the actions (moving from one position to another), and "cells," which encapsulate the game state. This duality is analogous to the one between Petri nets transitions and places. Hence, the module composition is particularly interesting when allowing and exhibiting this duality.

Figures 6a and 6b allow the composition of cell module instances with passages modules in between. Fig. 7 shows a high-level view of the whole game level of Fig. 1 as the composition of juxtaposed cell and passage instance modules.



**Figure 6:** Two Petri nets as modules: (a) cell and (b) passage with the respective abbreviated representations used in Fig. 7.



Figure 7: Module composition mimicking a game level "Mini Cosmos" at https://sokoban.info/?4.

#### 3.3.2. Complete model

The last modeling exercise can be the specification of all the necessary compositions. This exercise relies heavily on a tool that comprehensively supports model composition. Some textual language is probably the preferable approach. We present a possible pseudo-code for such composition (see Algorithm 1). The *Connect* procedure receives four-cell module instances and one passage module instance. A set of place fusions connects the five instances. The whole level is built by the *CreateBoard* procedure. This procedure calls the *Connect* procedure for each pair of positions (cells) with a passage between them. The function *ConnectionsList* returns the positions with a passage between them, thus effectively defining the game-level structure but not the initial state (marking). The *FourCells* function receives those two positions' coordinates and returns the four module instances needed for the composition. Functions *cell.create* and *passage.create* create new instances of module *cell* and module *passage* (identified by the respective parameters) if not yet created, and return the respective instance if already created.

If suitable tools are available to obtain the model for the complete level, using it for simulation or state-space analysis should be interesting. The state-space analysis is especially interesting for checking the reachability of the intended final marking (all boxes in the end cells) and the shortest path (step sequence) to reach it.

Algorithm 1 Create board (complete Petri net model)



## 4. Conclusions and future work

Using the Sokoban game as an example, we have shown how a classic tile-based game can be used as a basis for engaging P/T net modeling exercises. Considering low-level nets, the model also offers good opportunities to discuss and use test arcs, inhibitor arcs, or both to simplify the model. The game also provides a suitable ground for high-level Petri net modeling, including coloured Petri nets [10] and reference nets [11, 12, 13]. The presented exercise sequence can be adapted to the modeling of other tile-based games by P/T nets, high-level nets, or even non-autonomous nets.

In our future work, we recognize the necessity for a straightforward method to define and implement multiple module compositions. Specifically, this method should enable the composition of modular models in a declarative manner, utilizing a textual language.

## Acknowledgments

This work was financed by Portuguese Agency FCT – Fundação para a Ciência e Tecnologia, in the framework of project UIDB/00066/2020.

### References

- K. Sung, Computer games and traditional cs courses, Commun. ACM 52 (2009) 74–78. URL: https://doi.org/10.1145/1610252.1610273. doi:10.1145/1610252.1610273.
- [2] V. F. Martins, M. A. Eliseo, N. Omar, M. L. A. Castro, A. G. D. Corrêa, Using game development to teach programming, in: Handbook of Research on Immersive Digital Games in Educational Environments, IGI Global, 2019, pp. 450–485.
- [3] A. Vahldick, A. J. Mendes, M. J. Marcelino, A review of games designed to improve introductory computer programming competencies, in: 2014 IEEE Frontiers in Education Conference (FIE) Proceedings, 2014, pp. 1–7. doi:10.1109/FIE.2014.7044114.
- [4] M. J. Wolf (Ed.), The video game explosion: a history from Pong to Playstation®and beyond, Greenwood Press, 2008.
- [5] Wikipedia contributors, Sokoban Wikipedia, the free encyclopedia, https://en.wikipedia.org/w/ index.php?title=Sokoban&oldid=1219730853, 2024. [Online; accessed 21-April-2024].
- [6] T. Virkkala, Solving Sokoban, Ph.D. thesis, Masters thesis, University Of Helsinki, 2011.
- [7] D. Dor, U. Zwick, SOKOBAN and other motion planning problems, Computational Geometry 13 (1999) 215–228. URL: https://www.sciencedirect.com/science/article/pii/S0925772199000176. doi:https://doi.org/10.1016/S0925-7721(99)00017-6.
- [8] J. Culberson, Sokoban is PSPACE-complete, Technical Report, University of Alberta, 1997.
- [9] N. Froleyks, N. Froleyks, Using an algorithm portfolio to solve sokoban, Symposium on Combinatorial Search (2016). doi:10.5445/ir/1000073699.
- [10] K. Jensen, L. M. Kristensen, Coloured Petri Nets: Modelling and Validation of Concurrent Systems, 1st ed., Springer Publishing Company, Incorporated, 2009.
- [11] O. Kummer, F. Wienberg, M. Duvigneau, M. Köhler, D. Moldt, H. Rölke, Renew the Reference Net Workshop, in: E. Veerbeek (Ed.), Tool Demonstrations. 24th International Conference on Application and Theory of Petri Nets (ATPN 2003). International Conference on Business Process Management (BPM 2003)., Department of Technology Management, Technische Universiteit Eindhoven, Beta Research School for Operations Management and Logistics, 2003, pp. 99–102.
- [12] O. Kummer, Introduction to petri nets and reference nets, Sozionik Aktuell 1 (2001).
- [13] O. Kummer, F. Wienberg, M. Duvigneau, L. Cabac, M. Haustermann, D. Mosteller, D. M. Arbeitsbereich, Renew User Guide, Technical Report, University of Hamburg, Department for Informatics, Theoretical Foundations Group, 2022. Available online: https://www2.informatik.uni-hamburg.de/ TGI/renew/4.0/renew4.0.pdf (accessed on 2022-11-23).