Solving Puzzles using Coloured Petri Nets

S. Vanit-Anunchai

School of Telecommunication Engineering, Institute of Engineering, Suranaree University of Technology, Muang, Nakhon Ratchasima,Thailand

Abstract

This paper presents CPN models of three different puzzles: the Truth-tellers and liars problem; the Logical pairing problem; and the City tour problem. We attempt to explain design decisions and demonstrate how to use the models to solve the puzzles. The lessons learned is also discussed.

Keywords

State Space Analysis, Truth-tellers and liars problem, Logical pairing Problem, City tour problem.

1. Introduction

Teaching formal methods is hard but teaching them to the non-computer science students is harder. Because they lack not only sufficient knowledge in computer science but also specific background related to the assigned problems. This two-front trouble could be mitigated if we assign the problems that require exact thinking and no other specific background knowledge. An assignment category that matches these requirements is the logical puzzles. From my observation, students like playing logical puzzles and enjoy solving them. Because the logical puzzle questions are usually very clear to students, we often exploited them as toy examples in the class.

My inspiration of using Petri Nets to model games and logical puzzles started from modelling the "Game24" as discussed in [1]. We found that many logical puzzles can be easily modelled using Coloured Petri Nets (CPN) [2, 3]. In particular, we used PIPE2 (Platform Independent Petri Net Editor 2) [4] and CPN Tools [5] to create, edit, simulate and analyse the model. Because of its simplicity, I began teaching Grade 4-6 students to create games using PIPE2 and CPN Tools. Those students had created many games but [1] demonstrated only 3 simple examples: the stone picking puzzle; the egg fusion puzzle; and the river crossing puzzle. Continuing from [1], this paper illustrates selected CPN models of three different logical puzzles: Truth-tellers and liars problems, Logical pairing problems, and City tour problems. These CPN models are used to find the solutions of the puzzles. While others use programming languages such as C, Python, or Z3, we use state space tools searching for the answers. The major advantage of CPN Tools is that, to investigate the properties of the model, it has already included binding procedures, the state space generation and the query functions .

The rest of this paper is organised as follows. Section 2 discusses the related work. Section 3 discusses an example CPN model of the Truth-tellers and liars problem. Section 4 discusses an example CPN model of the Logical pairing problem. Section 5 presents an example CPN model of the City Tour problem. Section 6 presents lessons learned. Section 7 presents the conclusion.

2. Related work

The river crossing puzzles have long gained a lot of attention. The goal of the river crossing puzzle is to move objects across the river within the least number of trips. Usually the puzzle has constrains on how objects can be left together and how many objects can cross the river at the same time. There are four types of river crossing puzzles: 1) the dog, sheep, and cabbage problem 2) the jealous husbands problem 3) three missionaries and three cannibals problem 4) the bridge and torch problem. All four

somsav@sut.ac.th (S. V.)
0000-0003-4972-7713 (S. V.)

puzzles were modelled easily by grade 6 students but we do not discuss them in this paper due to space limitation. [6] applied dynamic programming to solve the missionaries and cannibals problem. [7] studied two river crossing problems: jealous husbands and missionaries and cannibals problems. She solved the problems using algebraic methods and studied the relationship between these problems using category theory. [8, 9] discussed various methods to solve the bridge and torch problems including dynamic programming.

Another important type of logic puzzles is the elimination grid problem. It typically presented with a grid filled with numbers or symbols. Given some rules or constraints, the task is to eliminate some possibilities so that the solution can be deduced. Well-known examples of the elimination grid problem are Sudoku [10] and Nonograms. Some logical puzzles involve pairing sets of objects and often are solved using the same elimination process. An elimination grid or table is a useful tool for solving the pairing problems. Therefore, the logical pairing problem sometimes called the elimination grid problem.

3. Truth-tellers and liars problems

Truth-tellers and liars problems are logic puzzles consisting of a set of statements provided by individuals who always tell the truth and individuals who always lie. These puzzles often involve a scenario presented with statements from several characters, each claiming to be the Truth-teller.

An example of the Truth-tellers and liars problems was modelled using CPN Tools as shown in Fig 1. In a crime scene, there are 4 people named Alice, Bob, Charlie and Danny. Some of them are the thieves. The innocents always tell the truth but the thieves always lie. After interrogation, four statements have been drawn as follows:

1) Alice said "Danny is the thief."
2) Bob said " Charlie is the thief."
3) Charlie said "Alice and Bob are the thieves."
4) Danny said "Bob is innocent."

We need to determine who are the thieves.

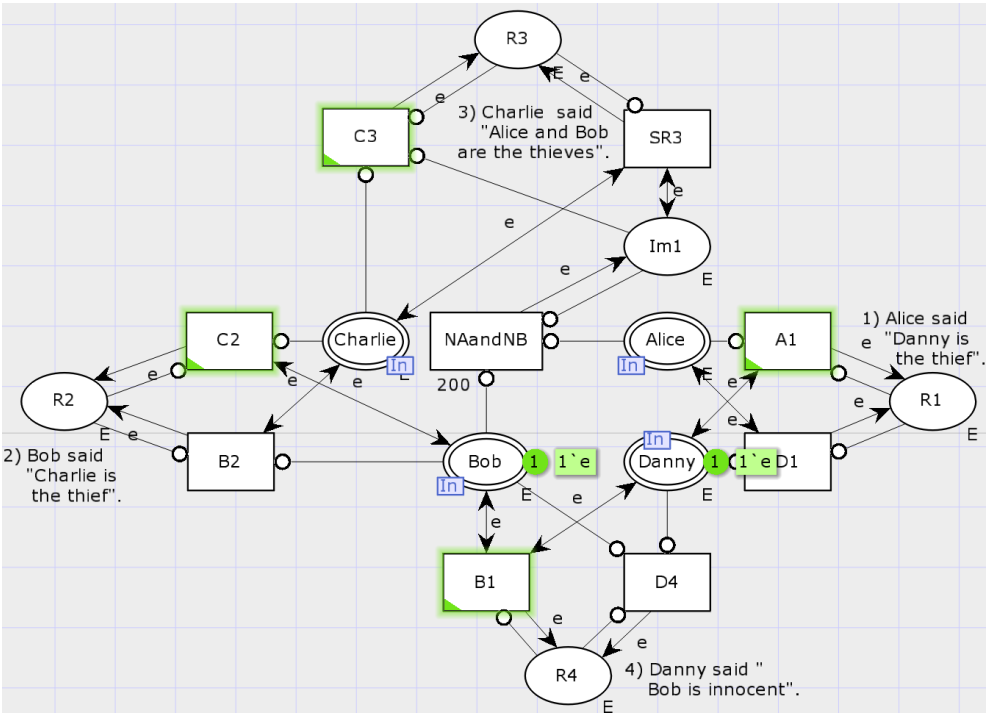

Figure 1: The CPN model of the Truth-tellers and liars problem.

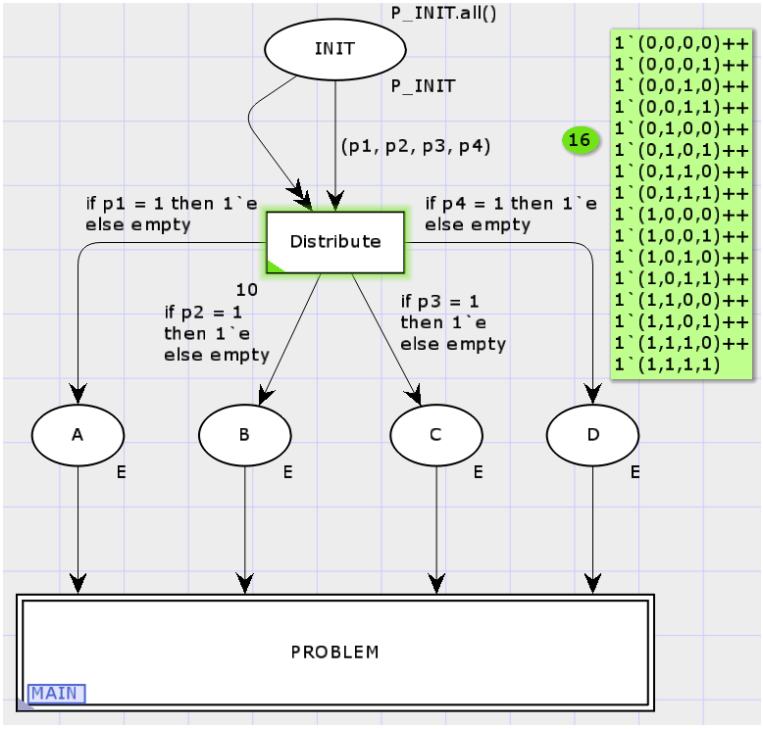

Figure 2: Initialization process.

```
val it = 16 : int
val evACC = fn : Node -> bool
val result = [88] : Node list
 Answer:   val it = 1 : int
```

```
size(ListDeadMarkings());
fun evACC  n =
if (ms_to_list(Mark.MAIN'R1  1 n)  <> []  andalso
ms_to_list(Mark.MAIN'R2  1 n)  <> []  andalso
ms_to_list(Mark.MAIN'R3  1  n)  <> [] andalso
ms_to_list(Mark.MAIN'R4  1 n)  <> [] )
then true else false;
val result=PredNodes
(ListDeadMarkings(), evACC, NoLimit);
val _ = print(" Answer:   ");
length(result);
```

```
88
4:0
```

```
MAIN'Alice 1: empty
MAIN'Bob 1: 1`e
MAIN'Charlie 1: empty
MAIN'Danny 1: 1`e
```

Figure 3: Using state space generation to find the solutions (the Truth-tellers and liars problem).

The existence of a token in the places in Fig 1, named after each person, means that he or she is innocent and always tells the truth. If the place is empty, that person is the thief and always lies. A token in places R1, R2, R3 or R4 means the statement is logically true.

This Truth-tellers and liars example can be simply solved using state space analysis. Figure 2 shows how we initially distribute a token (or empty) into the places Alice, Bob, Charlie and Danny. There are 16 possible initial markings. When an initial marking is selected, the Place Init is reset to empty by the double arrow reset arc. After generating the state space, we use ML query (Fig. 3) looking for the dead markings in which the statement places R1, R2, R3 and R4 are true or they are not empty.

On the other hand, this problem can also be solved simply by one transition binding as shown in Fig. 4. Four statements are coded in a guard function (matching). The binding of the transition is the answer of the problem. Although this model is very concise, it does not provide any insights. When the problem statements are incomplete or wrong, the model in Fig. 1 can provide more insights and help us fix the errors.

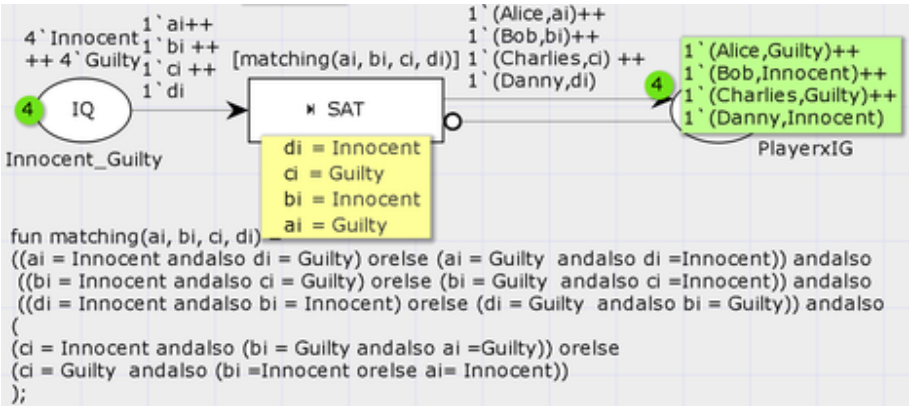

Figure 4: Another CPN model of the Truth-tellers and liars problem.

4. Logical pairing problems

Logical pairing problems are logic puzzles that require you to pair sets of objects into the correct groups. A well-known logical pairing puzzle is the Einstein's Riddle which was originally called the Zebra Puzzle. In this paper we shall consider the Five-house puzzle.

There are five houses located along a road. Each of the five houses is painted in a different colour (red, blue, yellow, green and white) and their inhabitants are of different nationality (British, Swedish, Danish, Norwegian and German), own different pets (cat, horse, dog, fish and bird), drink different beverages (water, rootbeer, coffee, milk and tea), smoke different brands of cigar (Blends, Dunhill, Blue Master, Prince and Pall Mall). Based on the following clues, the question is who owns the fish.

- The British lives in the red house.
- The Swedish owns a dog.
- The Danish drinks tea.
- The green house is next to the white house.
- The owner of the green house drinks coffee.
- The owner of the bird smokes Pall Mall cigar.
- The owner of the yellow house smokes Dunhill cigar.
- The owner of the center house drinks milk.
- The Norwegian owns the first house.
- The Blends smoker lives next to the cat owner.
- The DunHill smoker lives next to the horse owner.
- The Blue Master smoker drinks Root beer.
- The German smokes Prince.
- The Norwegian lives next to the blue house.
- The Blends smoker lives next to the one who drinks water.

The Five-house problems was modelled using CPN Tools shown in Fig. 5. Figure 6 lists the declaration of the model. To solve this particular problem we divided the clues into two groups. Each constraint of the first group is the clue related to single house but each constraint of the second group involves two

houses. After applying 10 constraints of the first group via the guard function chkn1 (Fig. 7), the number of possible solutions is reduced to 3,537. After applying another 5 constraints in Place Next_Cond, there are two only possible solutions as shown in Fig 8.

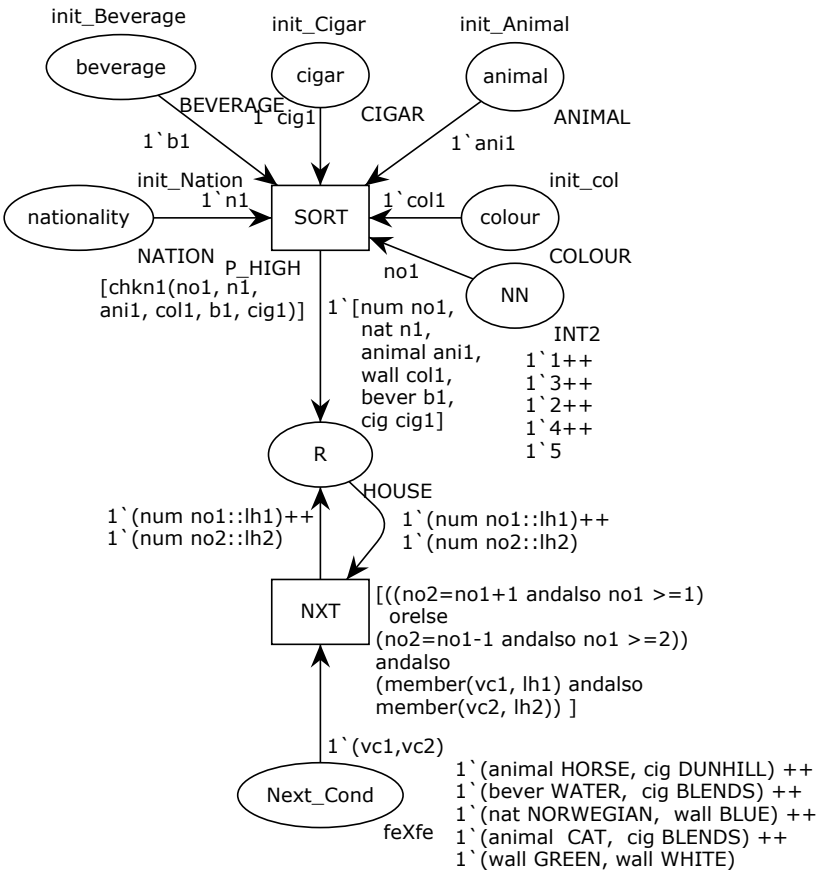

Figure 5: The CPN model of the Five-House puzzle.

5. City Tour problem

The objective of the city tour problems is to determine the chosen walking path from the starting point to the destination. The walking path is subject to some rules. Figure 9 shows a map of a 4x4 city blocks. The starting point is at Place P11 and the destination is at Place P55. The rules are following;

1. For every block you walk toward east direction, you gain two coins.
2. For every block you walk toward west direction, you lose two coins.
3. For every block you walk toward south direction, your money is double.
4. For every block you walk toward north direction, your money is half.
5. You shall not walk over the same paths but are allowed to pass the same intersection.

The objective of the game is that after you arrive at the destination Place P55, you must have zero coin. Which paths should you take?

Two types of substitution transitions are used to model walking east-west and north-south directions. The CPN subpages model the North-South and East-West walks are shown in Fig. 10 and Fig. 11 respectivly. The token is money which increases or reduces as we move along. When generating state space to find the solution, we encounter state explosion problem. Nevertheless we can play the game using the CPN Tools simulator. One of the possible solution to this game is shown in Fig. 12.

```
colset INT2   = int with 0..5;
colset NATION = with  BRITISH | SWEDISH | DANISH | NORWEGIAN | GERMAN;
colset ANIMAL = with  CAT | HORSE | DOG | FISH | BIRD;
colset BEVERAGE = with  WATER | ROOTBEER  | COFFEE | MILK | TEA;
colset COLOUR = with RED | BLUE | YELLOW  | GREEN | WHITE;
colset CIGAR = with BLENDS | DUNHILL | BLUE_MASTER | PRINCE | PALL_MALL;
var no1, no2:INT2;
var n1:NATION;
var ani1:ANIMAL;
var b1:BEVERAGE;
var col1:COLOUR;
var cig1:CIGAR;
colset FEATURE = union animal:ANIMAL + cig:CIGAR + bever:BEVERAGE
                       + wall:COLOUR+nat:NATION+num:INT;
colset HOUSE  = list FEATURE;
var vc1, vc2:FEATURE;
var lh1, lh2:HOUSE;
colset  feXfe= product FEATURE * FEATURE;
val init_Nation = 1`BRITISH ++ 1`SWEDISH ++ 1`DANISH ++ 1`NORWEGIAN ++ 1`GERMAN;
val init_Animal = 1`CAT ++ 1`HORSE ++ 1`DOG ++ 1`FISH ++ 1`BIRD;
val init_Beverage = 1`WATER ++ 1`ROOTBEER  ++ 1`COFFEE ++ 1`MILK ++ 1`TEA;
val init_col = 1`RED ++ 1`BLUE ++ 1`YELLOW ++ 1`GREEN ++1`WHITE;
val init_Cigar = 1`BLENDS ++ 1`DUNHILL ++ 1`BLUE_MASTER ++ 1`PRINCE ++ 1`PALL_MALL;
```

Figure 6: Declaration of the Five-House model.

```
fun chkn1(no1, n1, ani1, col1, b1, cig1) =
((n1 = BRITISH andalso col1 = RED) orelse (n1 <> BRITISH andalso col1 <> RED))
andalso
((n1 = SWEDISH andalso ani1 = DOG) orelse (n1 <> SWEDISH andalso ani1 <> DOG))
andalso
((n1 = DANISH andalso b1 = TEA) orelse (n1 <> DANISH andalso b1 <> TEA))
andalso
((col1 = GREEN andalso b1 = COFFEE) orelse (col1 <> GREEN andalso b1 <> COFFEE))
andalso
((cig1 = PALL_MALL andalso ani1 = BIRD) orelse (cig1 <> PALL_MALL andalso ani1 <> BIRD))
andalso
((col1 = YELLOW andalso  cig1= DUNHILL) orelse (col1 <> YELLOW andalso cig1<> DUNHILL))
andalso
((cig1=BLUE_MASTER andalso b1=ROOTBEER) orelse (cig1<>BLUE_MASTER andalso b1<>ROOTBEER))
andalso
((n1 = GERMAN andalso cig1 = PRINCE) orelse (n1 <> GERMAN andalso cig1 <> PRINCE))
andalso
((n1 = NORWEGIAN andalso no1 = 1) orelse (n1 <> NORWEGIAN andalso no1 <> 1))
andalso
((b1 = MILK andalso no1 = 3) orelse (b1<> MILK andalso no1 <> 3));
```

Figure 7: Ten constrains of the Five-House puzzle.

6. Lessons learned

1. The Truth-tellers and liars problems are in the subject of propositional logic. The problems are usually solved using proof by contradiction. When the problems involves too many characters and many statements, Logic deduction by hand could be very messy and prone to human errors.

Figure 1 shows the solution of the problem. With a token in places Bob and Danny transitions A1, B1, C2 and C3 are enabled. Thus, every statement (R1, R2, R3, and R4) is true. If the distribution of tokens in place Alice, Bob, Charlie and Danny, is not the solution, Fig. 1 will graphically show the

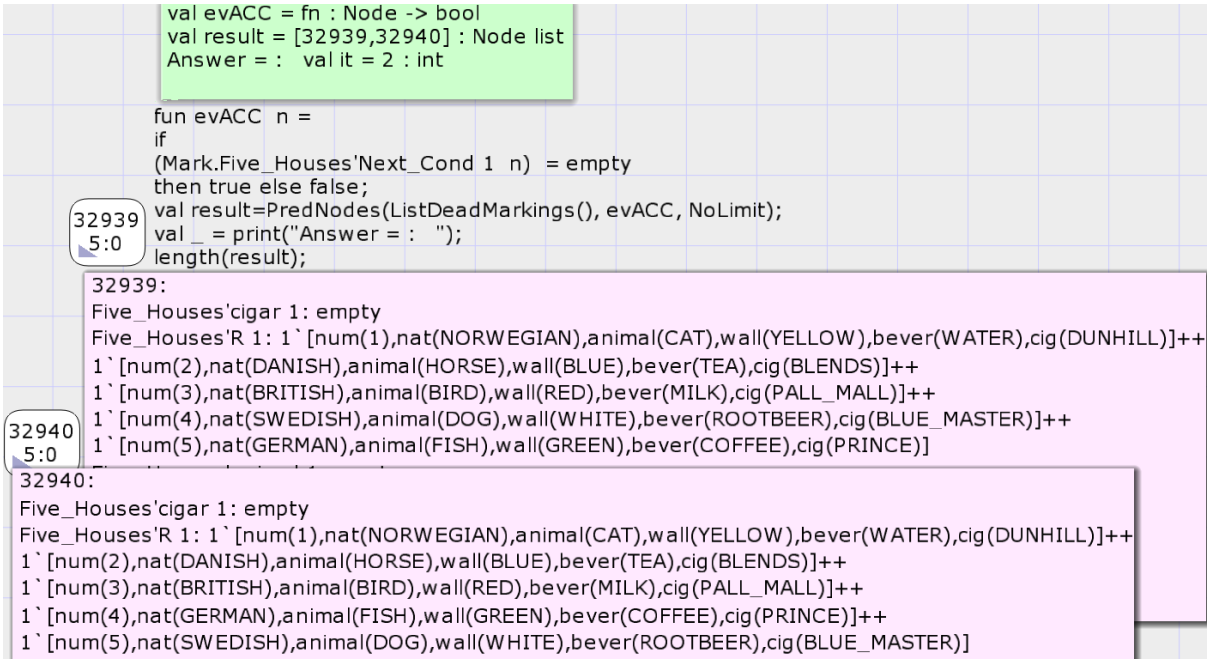

Figure 8: Using state space generation to find the solutions (the Five-House puzzle).

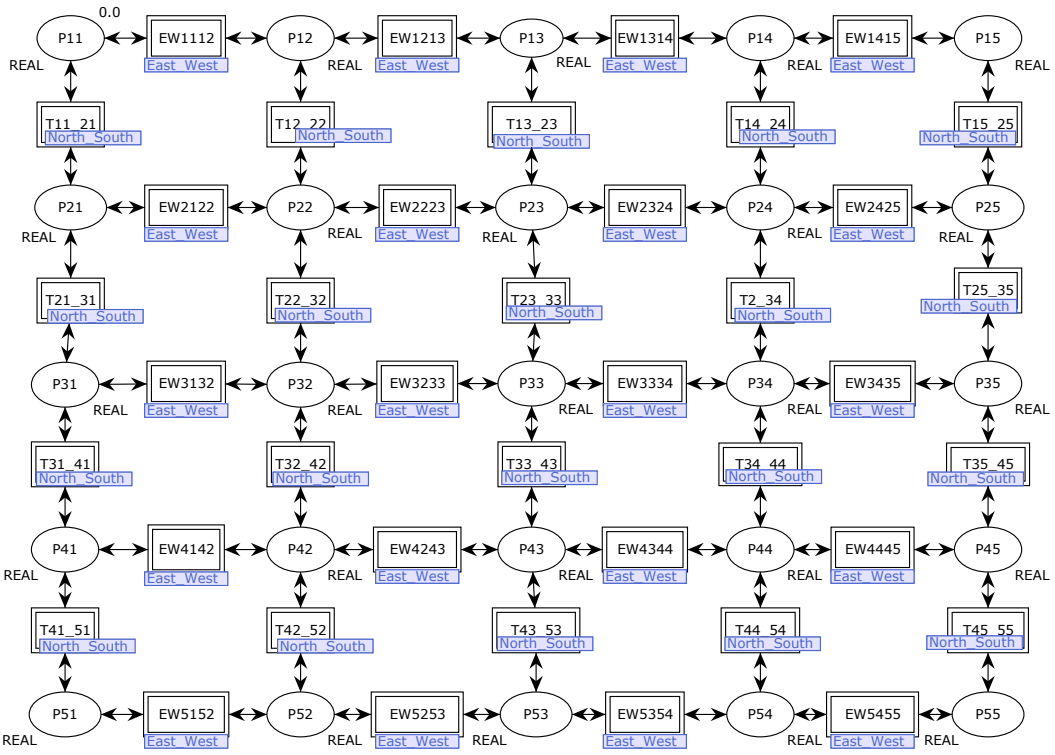

Figure 9: The CPN model of the City tour puzzle.

logic contradiction. When using other programming languages such as C or python, this graphical illustration would require more work done. Figure 4 also shows an elegant version of the CPN model in which the logic contradiction can be observed via the bindings.

2. The second puzzle is the Five-house problem (Fig. 5) which is more difficult to model and find the solutions. Because the five houses has totally six attributes (including locations) which is quite a large problem and some constraints are related to the two houses. The manual elimination using grid of such

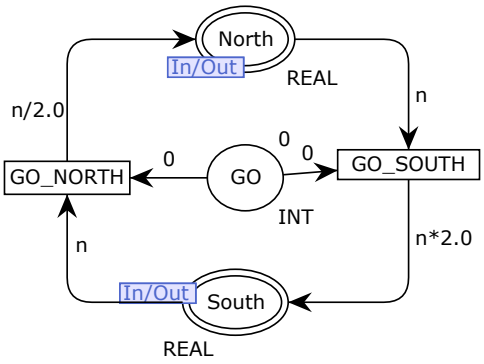

Figure 10: The CPN subpage models movement along North-South direction.

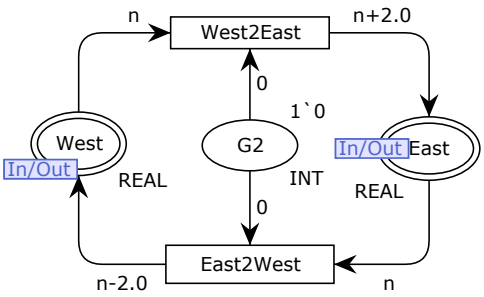

Figure 11: The CPN subpage models movement along East-West direction.

a large problems could be confusing and prone to errors. When dealing with a lot of constraints, we suggest to divide those constraints into various groups. We do not have to put every constraint into one transition but rather distribute them into a few transitions.

[11] discussed various methods to solve this problem. Using Z3 solver in Python is rather complicated, longer than using Constraint Solving Problems (CSPs). Solving by Haskell looks more elegant than the other two methods. From my observation, the CPN structure of the Five-house problem is similar to the net structure in [10] because both are solved with the same elimination process.

3. The third puzzle is the City tour problem which is quite different from the other problems. The characters in the first and second are not moving. But the character in the third puzzle is moving which is nicely captured by the token movement. The walking map is also nicely represented by the CPN net structure. Due to the state explosion, we can not find the solutions by the state space exploration. We use simulation and discover a solution.

4. In the class we used PIPE to model the first example: the dog, sheep, and cabbage problem (the river crossing puzzles). The jealous husbands problem and the three missionaries and three cannibals problem were left as students' execrcises using PIPE. After students spent time and effort creating complicated Place Transition models, we introduced CPN Tools in order to simplify their models. The bridge and torch puzzles was used to teach Timed Coloured Petri Nets. The City tour problem was used to teach hierarchical model. Following these steps and exercises, student's feedbacks were very positive.

7. Conclusion

This paper proposes the CPN model of three different type of puzzles: the Truth-tellers and liars; Logical pairing; and City tour puzzles. Games or puzzles are simpler version of many real applications. When we assign exercises to students using real applications, the students often do not understand the problems. Modeling games and puzzles instead could help them to get better acquaintance with the tools.

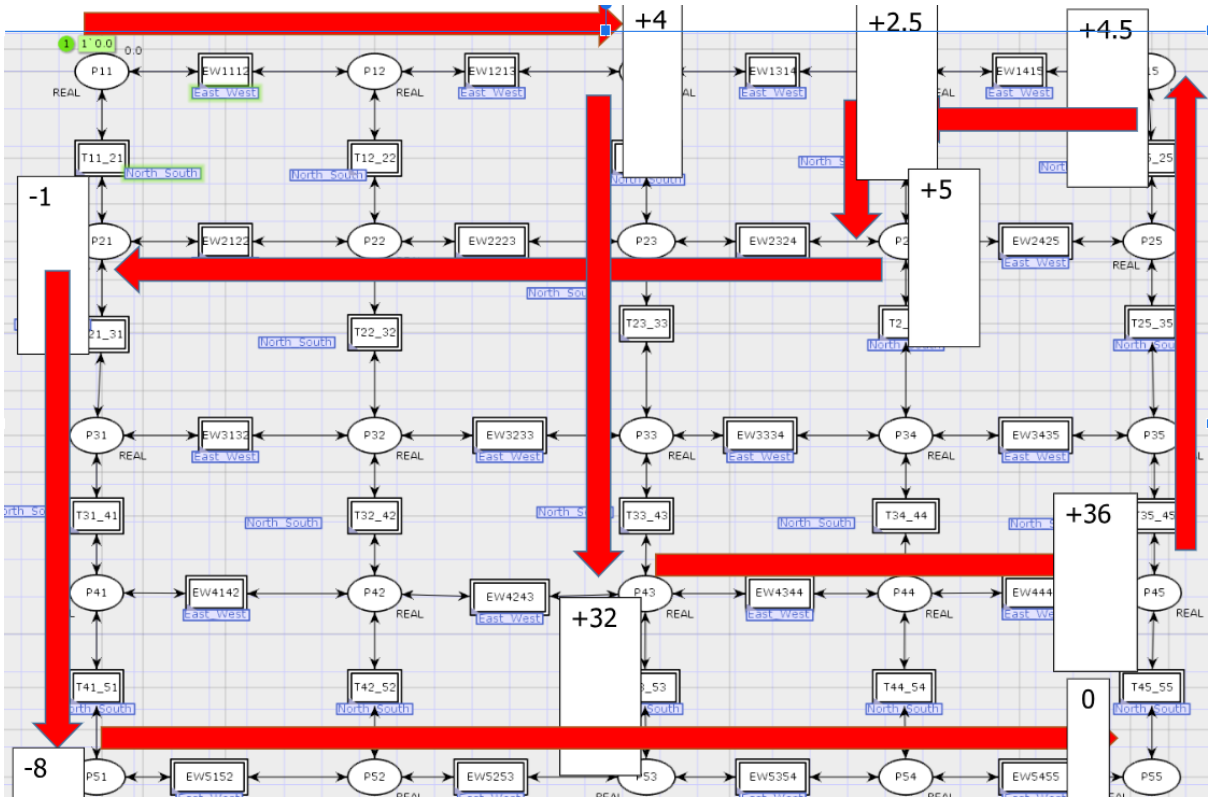

Figure 12: The solution to the City tour puzzle.

References

[1] S. Vanit-Anunchai, Teaching low-code formal methods with coloured petri nets, in: Formal Methods Teaching Workshop, Springer, 2023, pp. 96–104.

[2] K. Jensen, L. Kristensen, Coloured Petri Nets: Modelling and Validation of Concurrent Systems, Springer, Heidelberg, 2009.

[3] K. Jensen, L. M. Kristensen, Colored petri nets: a graphical language for formal modeling and validation of concurrent systems, Commun. ACM 58 (2015) 61–70. URL: http://doi.acm.org/10.1145/2663340. doi:10.1145/2663340.

[4] PIPE2, Platform Independent Petri net Editor 2, https://pipe2.sourceforge.net/, 2007.

[5] CPN Tools, CPN Tools home page, http://cpntools.org, 2018.

[6] R. Bellman, M. Sholander, Dynamic programming and "difficult crossing" puzzles, Mathematics Magazine 35 (1962) 27–29. doi:10.1080/0025570X.1962.11975287.

[7] E. A. Efimova, River crossing problems: Algebraic approach, 2018. arXiv:1802.09369.

[8] Crossing River Puzzles, Bridge and torch puzzle, https://stackoverflow.com/questions/1144207/bridge-crossing-puzzle, Access 2024.

[9] Crossing River Puzzles, Solving bridge and torch puzzle with dynamic programming, https://stackoverflow.com/questions/25399477/solving-bridge-and-torch-puzzle-with-dynamic-programming, Access 2024.

[10] E.G. Amparore, Model Sudoku: Colored Net, https://mcc.lip6.fr/2023/pdf/Sudoku-form.pdf, 2020.

[11] Einstein's riddle, How to solve Einstein's riddle with z3py the elegant way?, https://stackoverflow.com/questions/63790118/how-to-solve-einsteins-riddle-with-z3py-the-elegant-way, Access 2024.