Logical Reasoning with Object Diagrams in a UML and OCL Tool

Khanh-Hoang Doan(✉) and Martin Gogolla

University of Bremen, Computer Science Department, 28359 Bremen, Germany
{doankh,gogolla}@informatik.uni-bremen.de

Abstract. In this contribution, we introduce an approach to visualize and analyze logical reasoning problems in a UML and OCL tool by using logical puzzles represented with UML diagrams. Logical reasoning is formalized as a UML class diagram model enhanced by OCL restrictions. Puzzle rules and questions are expressed as either partial object diagrams or OCL formulas within the model. Solutions can be found and explored by a tool as object diagrams.

Keywords: Logical reasoning, UML and OCL model, Object diagram

1 Introduction

UML diagrams, such as class and object diagrams, are utilized to diagrammatically represent real-world system at an abstract level with some constraints formulated in OCL. Taking this as a basis, UML and OCL can be a promising solution for representing and visualizing logical reasoning problems. One approach for that will be introduced in this paper. A logical reasoning problem is represented as a UML class model enhanced by OCL restrictions. Rules and questions are formulated as either partial diagrams or OCL formulas within the model. The solutions can be found using a deduction system integrated in a tool and represented as object diagrams. This contribution focuses on representation and analysis of logical reasoning problems, in the context of the tool USE (Uml-based Specification Environment) [2].

Recently, the application of diagrammatic representation for reasoning has been a widely considered topic. For example, the approaches for reasoning with diagrams, i.e., Euler, Spider diagrams and Graphs, have been presented recently in [6], [5], and [4], respectively. In contrast to these approaches, our contribution uses UML diagrams, i.e. class and object diagrams, to visualize and analyze logical reasoning problems. In next section we will illustrate the idea of representing and visualizing a logical reasoning problem with our tool USE. In the last section, the paper ends with concluding remarks.

2 Solving and Representing Logical Reasoning Problems

Running example: In order to demonstrate our approach, an example of a logical reasoning problem is discussed. The deduction problem that we introduce

here is 'Einstein's Puzzle', a very well-known logic puzzle which sometimes is used as an example in teaching logic and formal methods [3].

Fig. 1. Einstein's puzzle.

The problem can be described as follows: Let us assume that there are five houses of different colors next to each other on the same road. In each house lives a man of a different nationality. Every man has his favorite drink, his favorite brand of cigarettes, and keeps a particular pet. There are fifteen clues of deduction that are listed below, and Fig. 1 illustrates the reasoning problem.

01. The Briton lives in the red house.
02. The Swede keeps dogs as pets.
03. The Dane drinks tea.
04. Looking from in front, the green house is just to the left of the white house.
05. The green house's owner drinks coffee.
06. The person who smokes Pall Malls raises birds.
07. The owner of the yellow house smokes Dunhill.
08. The man living in the center house drinks milk.
09. The Norwegian lives in the leftmost house.
10. The man who smokes Blends lives next to the one who keeps cats.
11. The man who keeps a horse lives next to the man who smokes Dunhill.
12. The owner who smokes Bluemasters also drinks beer.
13. The German smokes Prince.
14. The Norwegian lives next to the blue house.
15. The man who smokes Blends has a neighbor who drinks water

Construct a model: Firstly, a model corresponding to the problem being solved must be constructed. Depending on the problem, the model must include all necessary information, i.e., classes, attributes and associations, so that the model can simulate the problem.

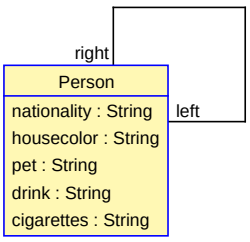

Fig. 2. Model of the logical reasoning problem.

For Einstein's Puzzle, we have formulate a model with five attributes: nationality, housecolour, pet, drink, cigarettes, and one association that represents the neighborhood relationship between the persons. Fig. 2 presents the class diagram of the model.

Formulate invariants: After constructing a suitable model, a collection of OCL invariants must be formulated on the model, one invariant for each clue (rule). As mentioned before, it is important that the model must cover all information from all clues. Therefore, we formulate 15 invariants corresponding to 15 rules of the puzzle. For

instance, the following listing is the invariant corresponding to the rule 08 “The man living in the center house drinks milk”.

```
context Person inv clue08:
  Person.allInstances()→one(p|Set{p} →closure(left)→size()=Set{p}
  →closure(right)→size() and p.drink='Milk')
```

In addition to the textual representation, some rules can be represented as a partial object diagram, which can enhance the understandability. For example, the diagrammatic visualization for the rule 08 is shown in Fig .3.

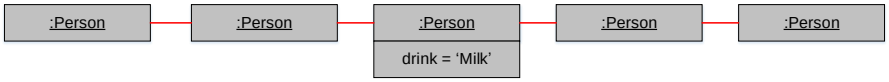

Fig. 3. The diagrammatic visualization of rule 08.

The other invariants are formulated analogously. The full model including all invariants is presented in [1].

Solving the problem with the model validator: After constructing a suitable model with all necessary invariants corresponding to all clues, we apply the model validator to solve the problem and find the answer. In the case of Einstein's puzzle, the model validator finds a solution as an object diagram, which is shown in Fig. 4. We arranged the 5 persons (objects) according to their neighborhood relationships from left to right. In the found solution we can easily check that some simple rules, e.g., rules 01, 02, 03, 13, are satisfied. Further analysis of the satisfaction of more complicated rules (invariants) on the found solution can be done with our tool with the 'Evaluation browser' functionality.

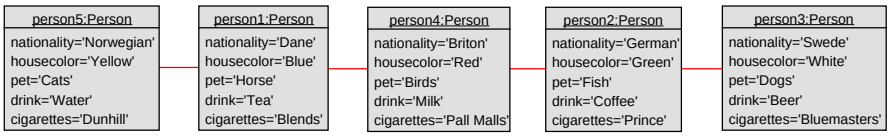

Fig. 4. Found solution for Einstein's Puzzle.

The search space of the model validator is defined by a configuration. Therefore we can speed up the solving process by setting a suitable configuration. Setting a proper configuration plays an essential role in the context of solving reasoning problem with the model validator. To archive this, one can go through the description of the problem and underline the number of objects (man/person) and the values which are given corresponding to the class attributes. As a result, the following list is the configuration for Einstein's puzzle.

```
Person_min_max = 5..5
Person_nationality =
  Set{'Norwegian','Dane','Briton','German','Swede'}
Person_housecolor = Set{'Red','Yellow','Blue','Green','White'}
Person_pet = Set{'Cats','Birds','Horse','Fish','Dogs'}
Person_drink = Set{'Water','Tea','Milk','Coffee','Beer'}
```

```
Person_cigarettes =
  Set{'Dunhill','Prince','Blends','Pall Malls','Bluemasters'}
Connecting_min_max = 4..4
```

Explore solution universe: In a case of having more than one solution, the model validator also provides an option to explore all of them. Naturally, this will be possible only if the solution universe is relatively small. To achieve all solutions we use the command `mv -scrollingAll <PropertyFile>`, and the additional succeeding command `mv -scrollingAll [prev|next|show(<N>)]` allows us to scroll through the solution interval and show each of them as an object diagram. For example, after executing the command `mv -scrollingAll show(1)`, the first (and only) solution is shown as an object diagram presented in Fig. 4.

3 Conclusion and Future Work

In this contribution we have described our method for visualizing and analyzing logical reasoning problems in the tool USE. We have used diagrammatic representations and puzzles as a cheap-prized entry to formal methods. Beside the Einstein's Puzzle, we have been applied the introduced approach for several popular logical reasoning examples, e.g. Sudoku puzzles or scheduling problems; the details can be seen in [1]. As future work we have identified to handle the various puzzle examples present in the literature in our approach. A further option is to develop a particular USE plugin particularly aiming at puzzle representation and to handle their solutions, as well as to allow the specification of OCL expressions with (partial object) diagrams.

References

1. Doan, K.H., Gogolla, M.: Addendum to Logical Reasoning with Object Diagrams in a UML and OCL Tool. Tech. rep., University of Bremen (2017), http://www.db.informatik.uni-bremen.de/publications/intern/ReasoningwithUSE.pdf
2. Gogolla, M., Hilken, F.: Model Validation and Verification Options in a Contemporary UML and OCL Analysis Tool. In: Proc. Modellierung (MODELLIERUNG'2016). pp. 203–218. GI, LNI 254 (2016)
3. Spichkova, M.: "Boring Formal Methods" or "Sherlock Holmes Deduction Methods"? In: STAF 2016 Workshops: DataMod, GCM, HOFM, MELO, SEMS, VeryComp. LNCS 9946. pp. 242–252 (2016)
4. Takemura, R.: A logical investigation of heterogeneous reasoning with graphs in elementary economics. In: Diagrammatic Representation and Inference - 9th International Conference, Diagrams 2016. pp. 98–104 (2016)
5. Urbas, M., Jamnik, M., Stapleton, G., Flower, J.: Speedith: A diagrammatic reasoner for spider diagrams. In: Diagrammatic Representation and Inference - 7th International Conference, Diagrams 2012. pp. 163–177 (2012)
6. Zelazek, F.: Diagrammatically explaining peircean abduction. In: Diagrammatic Representation and Inference - 8th International Conference, Diagrams 2014. pp. 308–310 (2014)