Considering Ill-Definedness of Problems from the Aspect of Solution Space

Nguyen-Thinh Le and **Wolfgang Menzel**
Department of Informatics
University of Hamburg, Germany
{le, menzel}@informatik.uni-hamburg.de

Niels Pinkwart
Department of Informatics
Clausthal University of Technology, Germany
niels.pinkwart@tu-clausthal.de

Abstract

Most researchers agree that there is a continuum between well-definedness and ill-definedness. However, positioning a specific problem within this continuum is not always easy. To determine the degree of ill-definedness of a problem, in this paper, we propose a classification into the following five classes with respect to the size of the problems's solution space: 1) one single solution, 2) one solution strategy which can be implemented in different variants, 3) a limited number of alternative solution strategies which can be implemented in different ways, 4) a great variety of possible solution strategies where solutions can be verified, and 5) the correctness of a solution cannot be verified.

Introduction

Recently, researchers in the area of intelligent tutoring systems have started considering ill-defined domains. Several definitions for the term "ill-definedness" have been proposed. Lynch et al. (2006) identified typical characteristics of an ill-defined domain. Simon (1973) considered well-definedness not as a property of a particular domain, but rather takes the perspective of the problem solving process which he casts as a heuristic search procedure. Regardless from which viewpoint ill-definedness of a domain or problem is observed, most researchers agree that there is a continuum between well-definedness and ill-definedness. Indeed, both well-defined and ill-defined problems may exist within a single domain (Mitrovic and Weerasinghe 2009). However, up to now a more detailed classification of this "continuum" has not been presented. How can we determine to which degree a problem is ill-defined? Since ITS systems typically pose problems to students that these have to solve, in this paper, we propose a scale of ill-definedness by considering the problems and their solution space. This classification may be helpful for choosing an appropriate modeling approach (e.g., model-tracing (Anderson et al. 1995), constraint-based modeling (Ohlsson 1994), or weighted constraint-based techniques (Le and Menzel 2009)) when building a tutoring system.

Two Levels of Solution Variability

Solution Strategy: Given a problem in a particular domain, an solver (e.g., a student) might have different solution strategies at her disposal. A solution strategy is based on the available means which can be used to deal with frequently occurring problem situations. E.g., in the domain of travel planning, a task could consist in finding a route between two places. Depending on the available means of transportation, different strategies can be applied: e.g., driving by car, taking a train, or taking a flight. In the domain of programming, almost always different solution alternatives are available. If a task is to write a Prolog predicate for calculating a return on investment, a direct analytical computation or an iteration can be used. The latter solution strategy can be further refined into normal recursion and tail recursion. Identifying the strategy underlying a solution is an important issue for tutoring: If feedback is meant to help the student improve her solution, it has to match the solution strategy the student is applying. Otherwise, the feedback may become useless or even confuse the student. This happens, for example, if the student has planned a travel using the train, but the tutoring system returns corrective hints in the context of a car-based strategy, since using a car might well be part of a strategy which predominantly relies on railway connections.

Implementation: Once a solution strategy has been selected for solving a specific problem, the problem solver is faced with the issue of how to use the available means for the implementation. That includes finding out how the different available constructs of a particular domain can be applied and arranged in the context of the chosen solution strategy. We illustrate this in the two domains mentioned above. If in the domain of travel planning the problem solver has chosen the strategy of using a car, she can find many different routes by combining different roads. Or, if traveling by train or airplane, the planner also can combine different train or flight connections to reach the desired destination. In the domain of programming, the programmer has to apply the primitives of the language being used. Usually, there are several options to realize a high level concept (e.g., using different combinations of arithmetic operators), and high level concepts can also be arranged in different ways.

Classification of Problem Tasks

Given a limited set of solution strategies and formal constructs of a domain, a potentially very large space of solutions can be derived for a problem. However, many tutoring systems constrict the student's freedom (Deek and McHugh

1999) due to the restricted ability of the underlying modeling approach, and thereby narrow down the possibilities of developing creative solutions. We classify problems into five classes according to increasing size of the solution space, thus defining a scale of ill-definedness.

Class 1: One solution strategy, one implementation. Problems of this level can be solved only according to a single solution strategy (possibly enforced by the user interface) and have only one solution. They are defined in a way that the solution is unique. Thus, they are suited to recall basic knowledge of the domain being taught. E.g., if the goal is to tutor the student to use a specific operator in Prolog correctly, a task might be: *"Please fill in the missing operator to sum the numbers 4 and 5. :"* `S ____ 4+5`.

Class 2: One solution strategy, alternative implementation variants. On the second level, problems can be solved according to a single solution strategy which, however, can be implemented in many different ways. Problems on this level are typically precisely specified so that the space of possible solutions is narrowed down to a single solution strategy, or the input is restricted by pre-specified solution templates, e.g.: *"Complete the following Prolog predicate in a way that it computes the return **R** of an investment **X** after **N** years for a fixed interest rate **Y** using an arithmetic expression"*. `invest(X, Y, N, R):- R is _____.` For such a task, the student is not allowed to implement other solution strategies except the analytic one which requires just a formulae for calculating compound interest.

Class 3: A limited number of alternative solution strategies. In this class of problems, the student is free to choose one of several known alternative solution strategies, and implements it according to her preferences, e.g.: *"Write a predicate to compute the return of investment after a period for a fixed interest rate."*. This kind of problem is more challenging to students than the two classes before because they have to make appropriate design decisions between solution strategies and implementation variants instead of simply applying a predefined solution template. In the case that a student solution does not satisfy the requirements of a given problem, appropriate feedback can only be given to the student if the system has a reasonable hypothesis about the underlying solution strategy most likely applied by the student.

Class 4: A great variability of possible solution strategies but the correctness of a solution can be verified. In this class, the problem is so complex that it needs to be solved by dividing it into sub-problems, where each of them can be solved using different solution strategies. Since the combination of solution strategies results in a new solution strategy for the overall task, the number of these combinations is not known a priori. A sample problem of this class might be represented by the following: *"Develop a calculator to calculate the return on investment"*. To solve this task, a number of design decisions have to be made concerning many issues, e.g.: The choice of data representations, a suitable heuristic search method, and defining helper predicates. Thus, the space of combinations of design decisions becomes large.

Class 5: Great variety of possible solution strategies and the correctness of solutions cannot be verified. Problems of this class typically require solutions not only to fulfil certain testable functional requirements (cf. class 4), but additionally their solutions should be considered "useful" and acceptable by a large number of stakeholders. The latter requirement usually results in controversial opinions which makes solutions not formally verifiable. The task *"Develop an investment simulation system"* is an example: The correctness of solutions of this problem relies on judgements and acceptance by different groups of end users.

Conclusion

We have identified two levels of solution variability for a problem task: solution strategy and implementation, based on which this paper has classified problems into five classes with respect to the size of the solution space. This classification represents a scale of ill-definedness.

References

Anderson, J. R.; Corbett, A. T.; Koedinger, K. R.; and Pelletier, R. 1995. Cognitive tutors: Lessons learned. *J. of the Learning Sciences* 4:167–207.

Deek, F. P., and McHugh, J. 1999. A survey and critical review of tools for learning programming. *Journal of Computer Science Education* 8(2):130–178.

Le, N.-T., and Menzel, W. 2009. Using weighted constraints to diagnose errors in logic programming- The case of an ill-defined domain. *Int. Journal of AI in Edu.* in press.

Lynch, C. F.; Ashley, K. D.; Aleven, V.; and Pinkwart, N. 2006. Defining ill-defined domains; a literature survey. In Ashley, K.; Aleven, V.; Pinkwart, N.; and Lynch, C., eds., *P. of the Workshop on ITSs for Ill-Defined Domains*, 1–10.

Mitrovic, A., and Weerasinghe, A. 2009. Revisiting ill-definedness and the consequences for ITSs. In Dimitrova, V.; Mizoguchi, R.; du Boulay, B.; and Graesser, A., eds., *P. of the 14th I. Conf. on AIED*, 375–382.

Ohlsson, S. 1994. Constraint-based student modelling. In Greer, J. E., and McCalla, G. I., eds., *Student Modelling: The Key to Individualized Knowledge-based Instruction*. Berlin: Springer. 167–189.

Simon, H. A. 1973. The structure of ill structured problems. *Artificial Intelligence* 4(3):181–201.