

Visual Representation of Algebraic Specifications: a User-Oriented Approach

Duncan S. Neary and Martin R. Woodward
Department of Computer Science
University of Liverpool
Chadwick Building, Peach Street
Liverpool, L69 7ZF, UK.
{dunc,mrw}@csc.liv.ac.uk

Abstract

This paper introduces a series of graphical notations for the visualisation of algebraic specifications, concentrating on a suitable representation of the equations that are a characteristic feature of such specifications. A user-oriented process was central both to the design of the notations and to the selection of a notation to implement. Details of this process are outlined.

1 Introduction

The inclusion of formal methods in the software development process can aid the delivery of high quality results. However, these deliverable advantages have not translated into widespread use of formal methods, which still remain primarily an academic field. The authors believe a factor in their limited use is difficulties encountered in understanding and applying formal methods. The experiment of Finney [1] for the Z specification notation appears to confirm this. The work presented here attempts to resolve this issue of understandability by the application of visual programming techniques to formal methods, specifically the algebraic specification language OBJ [2], through the creation of a series of visual notations.

OBJ, like other algebraic specification notations, is classified as a property-based technique, meaning that the behaviour or properties of an intended system are described without an explicit mathematical model being constructed. Equations are the key device by which properties are defined. In essence they describe the relationships between operations whose syntax is specified via their signatures, i.e. their domain and range types. One of the attractive features of algebraic specifications is that the equations may be used in a term rewriting process to reduce arbitrary, but valid, expressions involving compositions of operations to simpler canonical forms. This can be viewed as a kind of

executability or animation feature which in practice allows one to test algebraic specifications.

2 Profiling target users

The authors decided upon an approach that was primarily user-oriented. This involved first the selection of a target group for the project. Given the fact that one author has over twelve years of experience teaching formal methods, a great deal of observational research regarding students had been made prior to the project. This led to the identification of novice OBJ users as a suitable target group, since they had been regularly observed to have difficulty with the concepts behind, and the use of, OBJ.

The next step involved the creation of criteria for the application of visual techniques. Again, the requirements of the user were considered central to the creation of criteria; hence a survey of novice OBJ users was conducted. The observational research that had been collected helped to create a list of commonly occurring problems which were used as the basis for the survey, which confirmed the hypothesis that novice users did indeed experience difficulty when creating and using algebraic specifications.

3 Visual notations

The notations considered can be divided into two broad groups, namely 'structured' and 'unstructured'. The difference between these groups is in the layout of the notation: structured notations having rigid layout rules and unstructured having only general layout rules. This section aims to introduce the various notations.

3.1 Structured Notations

Three structured notations were considered, but only two are displayed in Fig 1 due to limitations on space, these are

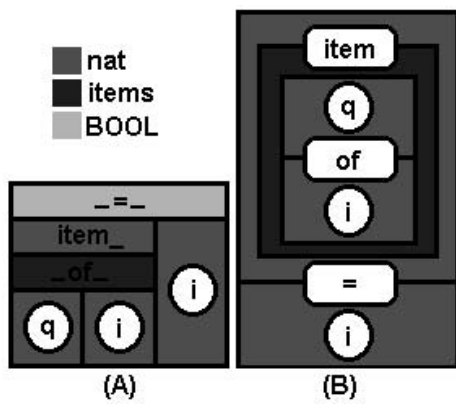


Figure 1. NS and VNB representations of equation $item(q\ of\ i) = i$

Nassi-Shneiderman (NS) (Fig 1 (A)) charts and the Vertical Nested Box (VNB) notation (Fig 1 (B)). Both are based on Nassi-Shneiderman (NS) charts [3], and utilise nesting to indicate the application of operations. They differ from traditional NS charts however, as circles are used to denote variables and colour to represent types. Further, the VNB notation adds a vertical aspect to complement the notation. This change in layout allows natural differentiation between prefix and infix syntax for operations.

As examples, the NS and VNB versions of the equation $item(q\ of\ i) = i$ are given in Fig 1. As can be seen, the operation $_of_$ is applied to q and i , and returns the type $items$; hence the $_of_$ rectangle contains the variables q and i , and is surrounded by the colour blue, which represents the type $items$.

3.2 Unstructured Notations

Two unstructured notations were considered. The first is based on GRASP [4] (Fig 2 (A)), a notation for representing the language PROLOG, and the second, operation relationship diagrams (ORD) (Fig 2 (B)) is inspired by entity relationship diagrams. Both utilise arrows to represent the relationship between the various entities. For example, in Fig 2 (A), the GRASP representation of $item(q\ of\ i) = i$, the result of the operation $_of_$ is passed to the operation $item_$. This is represented by connecting the two operations by an arrow which leaves $_of_$ and joins $item_$.

The display of operations and types differs between the notations. GRASP represents an operation by a rounded rectangle and surrounds it with a 'type' rectangle to indicate its type, whilst ORD distinguishes between user-defined operations and 'built-in' operations by surrounding the former with a diamond and the latter with a square.

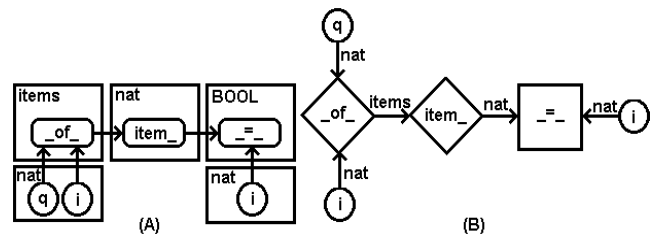


Figure 2. GRASP and ORD representations of equation $item(q\ of\ i) = i$

4 Conclusion

This paper has presented a number of notations intended to address commonly occurring problems which novice users experience when using OBJ. To judge their usefulness, the notations have been evaluated by means of the survey mentioned previously and by the application of comparative metrics [5]. Roughly 80% of survey participants thought that the visual notations were easier to understand than textual OBJ, with VNB being the most popular. This, in conjunction with the comparative metrics, led to VNB's selection for implementation in a pilot system. In addition, an experiment has been undertaken to compare the comprehensibility of VNB and NS notations to text [6]. The results of all these evaluation efforts are encouraging and bode well for the resultant system.

References

- [1] K. Finney. Mathematical Notation in Formal Specification: Too Difficult for the Masses? *IEEE Transactions on Software Engineering*, 22(2):158-159, 1996.
- [2] J.A. Goguen and J.J. Tardo. An Introduction to OBJ: A Language for Writing and Testing Formal Algebraic Program Specifications. *Proc. Conf. on Specification of Reliable Software*. IEEE Computer Society. 170-189, 1979.
- [3] I. Nassi and B. Shneiderman. Flowchart Techniques for Structured Programming. *ACM Sigplan Notices*, 8(8): 12-26, 1973.
- [4] Agustí, J., Puigsegur, J., and Robertson, D. A Visual Syntax for Logic and Logic Programming. *Journal of Visual Languages and Computing*, 9: 399-427, 1998.
- [5] J.D. Kiper, E. Howard and C. Ames. Criteria for Evaluation of Visual Programming Languages. *Journal of Visual Languages and Computing*. 8:175-192, 1997.
- [6] D.S. Neary and M.R. Woodward. Comparing the Comprehensibility of Textual and Visual Forms for Algebraic Specifications. *Journal of Visual Languages and Computing*, to appear.