Meaning Formulas for Syntax-Based Mathematical Algorithms*

Extended Abstract

William M. Farmer

McMaster University
Hamilton, Ontario, Canada
wmfarmer@mcmaster.ca

Many symbolic algorithms work by manipulating mathematical expressions in a mathematically meaningful way. A *meaning formula* for such an algorithm is a statement that captures the mathematical relationship between the input and output expressions of the algorithm. For example, consider a symbolic differentiation algorithm that takes as input an expression (say x^2), repeatedly applies syntactic differentiation rules to the expression, and then returns as output the final expression (say $2x$) that is produced. The intended meaning formula of this algorithm states that the function $(\lambda\, x : \mathbb{R} \;.\; 2x)$ represented by the output expression is the derivative of the function $(\lambda\, x : \mathbb{R} \;.\; x^2)$ represented by the input expression.

Meaning formulas of this kind are difficult to express in a traditional logic like first-order logic or simple type theory since there is no way to directly refer to the syntactic structure of the expressions in the logic. We argue that, to properly express meaning formulas, a logic is needed that has (1) an inductive type of syntactic values (e.g., syntax trees) that represent the syntactic structures of the expressions in the logic, (2) a quotation operator in the logic that maps expressions to syntactic values, and (3) an evaluation operator in the logic that maps syntactic values to expressions. We call this the *replete approach* for integrating reasoning about the syntax of the expressions with reasoning about what the expressions mean [5, 6].

Developing a logic that supports the replete approach is severely complicated by several thorny logical problems. The first of these is that the liar paradox can be expressed in the logic — which renders the logic inconsistent — if evaluation is unrestricted. This means that the evaluation operator must denote a partial operation, the use of evaluation must be significantly restricted, and expressions involving the evaluation operator may be undefined. The second problem is that syntactic notions, like whether a variable is free in an expression, can depend on the semantics of the expression as well as on the syntax of the expression. This means that, in the presence of the evaluation operator, syntactic operations like substitution are much more complicated than they are in a traditional logic. We have developed a version of simple type theory called $\mathcal{Q}_0^{\mathrm{uqe}}$ that supports the replete approach and solves these two problems [6].

In a logic supporting the replete approach it is possible not only to formally state meaning formulas, but also to formally prove meaning formulas. A proof of a meaning formula shows that the intended mathematical meaning of the algorithm follows from the computational behavior of the algorithm. That is, that the syntactic manipulations performed by the algorithm are mathematically correct. Of course, a formal proof of a meaning formula requires a proof system for the logic that has effective tools for reasoning about quotation and evaluation. $\mathcal{Q}_0^{\mathrm{uqe}}$ includes such a proof system, and we have proved in $\mathcal{Q}_0^{\mathrm{uqe}}$ meaning formulas for toy symbolic algorithms [6].

*This research is support by NSERC.

Meaning formulas have several applications. The first and simplest application is to use a meaning formula as a high-level, formal requirements specification of a symbolic algorithm. The algorithm satisfies its specification if the meaning formula is true. A second application is to use meaning formulas to construct *biform theories* [1, 4] in which mathematical knowledge is represented by a combination of axioms and algorithms. The knowledge embodied in a black-box algorithm would be represented by a meaning formula assumed as an axiom, while the knowledge embodied in an algorithm that has been proved correct would be represented by a meaning formula derived as a theorem.

A third application is to define a mathematical service as a pair consisting of a biform theory and a meaning formula for a mathematical algorithm [3]. The biform theory provides the context for the service, and the meaning formula specifies the service. An instance of the service is the relationship between an input expression given to the algorithm and the output expression returned by the algorithm. An instance of the service is obtained by instantiating the meaning formula with the input expression and then simplifying the resulting formula.

Acknowledgments

This research is part of the MathScheme project [2], a long-term initiative at McMaster University led by Dr. Jacques Carette and the author with the aim of producing a framework in which formal deduction and symbolic computation are tightly integrated.

References

[1] J. Carette and W. M. Farmer. High-level theories. In A. Autexier, J. Campbell, J. Rubio, M. Suzuki, and F. Wiedijk, editors, *Intelligent Computer Mathematics*, volume 5144 of *Lecture Notes in Computer Science*, pages 232–245. Springer-Verlag, 2008.

[2] J. Carette, W. M. Farmer, and R. O'Connor. Mathscheme: Project description. In J. H. Davenport, W. M. Farmer, F. Rabe, and J. Urban, editors, *Intelligent Computer Mathematics*, volume 6824 of *Lecture Notes in Computer Science*, pages 287–288. Springer-Verlag, 2011.

[3] J. Carette, W. M. Farmer, and J. Wajs. Trustable communication between mathematics systems. In T. Hardin and R. Rioboo, editors, *Calculemus 2003*, pages 58–68, Rome, Italy, 2003. Aracne.

[4] W. M. Farmer. Biform theories in Chiron. In M. Kauers, M. Kerber, R. R. Miner, and W. Windsteiger, editors, *Towards Mechanized Mathematical Assistants*, volume 4573 of *Lecture Notes in Computer Science*, pages 66–79. Springer-Verlag, 2007.

[5] W. M. Farmer. The formalization of syntax-based mathematical algorithms using quotation and evaluation. In J. Carette, D. Aspinall, C. Lange, P. Sojka, and W. Windsteiger, editors, *Intelligent Computer Mathematics*, volume 7961 of *Lecture Notes in Computer Science*, pages 35–50. Springer-Verlag, 2013.

[6] W. M. Farmer. Simple type theory with undefinedness, quotation, and evaluation. McSCert Report No. 13, McMaster University, 2014. Available at `http://imps.mcmaster.ca/doc/stt-with-uqe.pdf`.