J. Austral. Math. Soc. (*Series A*) 39 (1985), 101–106

GENERALIZED IMPLICATION EQUATION LANGUAGES

NORMAN Y. FOO and ROSLYN B. RILEY

(Received 17 January 1983; revised 21 December 1983)

Communicated by J. N. Crossley

Abstract

The calculus for equational implication languages given by Selman is generalized to handle the logical equivalent of the if... then... else... construct of high level programming languages. The relevance of these results to current investigations in the algebraic specifications of data types is discussed.

1980 *Mathematics subject classification* (*Amer. Math. Soc.*): 68 C 01.
Keywords and phrases: implication language, algebraic theory, decidability, completeness.

1. Introduction

Recent work by Goguen et al. [2] has shown that the algebraic specification of data types advocated by Guttag [3] and others is really an application of versions of the theory of varieties and quasi-varieties. The application however is not trivial. Questions relating to the computational equivalence, adequacy and efficiency are not yet completely resolved. Slightly variations in the syntax of the languages admitted seem to have a great effect on the succinctness of some data type specifications, as suggested by Majster [4], and proved by Thatcher *et al.* [8]. The general problem of showing that one specification is equivalent to another is recursively unsolvable, so good heuristics for transforming from one syntax to another and for eliminating redundant axioms are computationally useful and desirable.

The most general syntax considered by workers in the field of algebraic specifications is that adopted by Goguen *et al.* [2]. It admits the usual equations which axiomatize varieties, and in addition, incorporates the if...then... and if...then...else... constructs of programming languages like Algol and Pascal.

Hence axioms of the forms

$$E \to F \text{ and } (E \to F).(-E \to G)$$

are in effect included. (Here E, F, G are equations, and $\to, \cdot, -$, denote the logical connectives "implies", "conjunction" and "negation" respectively.) We are interested in devising valid manipulation rules for expressions in this syntax, with a view to obtaining the transformation heuristics alluded to earlier. Such a deductive calculus can be found by extending the complete calculus provided by Selman [7] for an equational language with an if...then... construct to one which also includes if...then...else.... It turns out that it is no more difficult to generalize the syntax even further to permit all expressions of the form $A \to B$, where A and B are either equations or the negations of equations. The calculus to be examined here therefore includes all the variants: if A then B, if A then B else C, in which A, B, C may be equations or their negations.

It is probable that results on different calculi for classes of specification languages will have a significant bearing on algorithms that implement them. A typical example of such an algorithm is the on-line equality prover of Samet [5], which deals with the more restricted class of an equational language without variable names. Entailment in that case is decidable but as indicated later the introduction of variables makes it undecidable. This undecidability property is obvious if the recursive unsolvability of the word problem for groups or semigroups is assumed, however it follows more simply from showing how to encode the halting problem for Turing machines in this generalized syntax.

2. The formal system

A generalized implication equation language (GIEL), L, is defined by a denumerable set of variables; a set of constants; and a set of function symbols (each of finite rank). From these the terms are formed by a finite number of applications of the following two rules.

1. All constants and variables are terms.

2. If the function symbol, f, has rank n, and $(t_1 \cdots t_n)$ are terms then $f((t_1 \cdots t_n))$ is a term. For any two terms, $t1$ and $t2$, the following are atomic formulas.

1. $t1 = t2$.

2. $-t1 \doteq t2$.

An equation with free variable x, say $t1(x) = t2(x)$ is implicitly universally quantified over this variable. Hence its negation is the existential statement that there is some x with $t1(x)$ not equal to $t2(x)$. The notation above, $-t1 = t2$ is

https://doi.org/10.1017/S1446788700022217 Published online by Cambridge University Press

used for this negative existential, since the alternative, $t1 \neq t2$, lends itself to a misinterpretation as a universal negation. For consistency, the following definitions of formulas also have the implicit universal quantification. The formulas are either atomic formulas or of the form $F \to G$, where F and G are atomic formulas. The only other symbols in L are those used above: $=$, $\to$, $-$, (, and). The axioms and rules of inference are given below, with p, q, $(t_1 \cdots t_n)$ for terms; E is an equation, each Fi is an atomic formula and f is a function symbol of rank n. In them opp is used as a metasymbol defined by opp $p = q$ is $-p = q$ and opp $-p = q$ is $p = q$. This is used to subsume four distinct rules in a common scheme and to remove the need to cope with constructions like $- - p = q$.

A1. $E \to E$

A2. $p = p$

A3. $p = q \to q = p$

R1. from $F1 \to F2$ infer opp $F2 \to$ opp $F1$ (called rule of contraposition)

R2. from $F1$ infer $F2 \to F1$

R3. from $F1$ and $F1 \to F2$ infer $F2$

R4. from $F1 \to F2$ and $F2 \to F3$ infer $F1 \to F3$

R5. from $F \to p = q$ and $F \to q = r$ infer $F \to p = r$

R6. from $F \to p_1 = q_1$, and $\ldots, F \to p_n = q_n$ infer $F \to f(p_1,\ldots,p_n) = f(q_1,\ldots,q_n)$

R7. from $B(z)$ infer $B(p)$: the result of replacing all occurrences of variable z in formula B by p.

It should be observed that these rules are patterned after those in Selman [7], in particular the last six are virtually the same, and in all cases we have as far as poosible retained his notation, nomenclature and proof style to facilitate ease of understanding and reference for those familiar with his results. However, in R7 when the formula $B(z)$ is of the form $-t1(z) = t2(z)$, the substituted p must have a previously unused name.

A first generalization of an if...then... construct, particularly in computer science usage, would be an if...then...else... construct. Since "if $E1$ then $E2$ else $E3$" is equivalent to "if $E1$ then $E2$ and if $-E1$ then $E3$", this is obviously subsumed in the language given above. The inclusion is strict: while contraposition of the if...then...else... will give formulas of the forms, if $E1$ then $E2$, if $-E1$ then $E2$, and if $-E2$ then $-E1$, it will not provide the scheme, if $E1$ then $-E2$, as is permitted in the generalized form admitted in our language.

It should not be thought that the scheme, if $E1$ then $E2$ else $E3$, enforces the joint existence of $E1$ and $-E1$ as antecedents in provable formulas for, if $-E1$ is an axiom, the if...then...else... reduces to $-E1 \to E3$ alone, since the if part will never be true.

https://doi.org/10.1017/S1446788700022217 Published online by Cambridge University Press

3. Completeness

A formula is said to be closed if it contains no variable symbols. Given a set of formulas, S, in L, a standard deduction from it is a deduction in which all applications of the *modus ponens* rule (R3) has as its minor premise either an axiom, a formula from S, or a formula derived from S by a finite number of applications of R7 (the rule permitting substitution of constants).

The following lemmas and theorem require that if there is a deduction of F from S, then there is a standard deduction of F from S. In order for this to be true the following derived rules of inference must be added:

R8. from $p = q$ and $q = r$ infer $p = r$

R9. from $p_1 = q_1$ and $\ldots p_n = q_n$ infer $f(p_1 \cdots p_n) = f(q_1 \cdots q_n)$

R10. from $F1 \to F2$ and opp $F2$ infer opp $F1$.

It is easily seen that these are implied by R1 to R7. Using these rules, a deduction is made standard in two steps: first the substitutions of constants are placed as soon as possible in the deduction pattern, and then the appropriate substitutions are made in all uses of R3. For example,

$$\dfrac{\dfrac{F1 \to F2}{\text{(applying R1)}}\qquad \text{opp } F2}{\text{(applying R3)}}\quad \text{opp } F2 \to \text{opp } F1 \qquad \text{becomes} \qquad \dfrac{\text{opp } F2, \quad F1 \to F2}{\text{(R10)}}$$

$$\text{opp } F1 \qquad\qquad\qquad\qquad \text{opp } F1.$$

Both deductions prove the formula, opp $F1$, but the second is standard, the first is not.

Analogously with Selman [7] we have the following lemmas.

LEMMA 1. *If $F1$ and $F2$ are atomic formulas and $F1$ is closed, and $S \cup \{F1\} \vdash F2$ then $S \vdash (F1 \to F2)$.*

LEMMA 2. *If, F, $F1$, $F2$ are atomic formulas, $F1$ and $F2$ are closed and $S \cup \{F1\} \vdash F$ and $S \cup \{F1 \to F2\} \vdash F$ then $S \vdash F$.*

LEMMA 3. *If G is a formula containing only the variables $x_1, \ldots, x_n$ and $L+$ is the extension of the language L formed by enlarging the set of constants to include the new symbols $c_1 \cdots c_n$ then $G+$ represents the fomrula in $L+$ formed by replacing each x_i in G by c_i. With this notation we have $S \vdash G+$ in $L+$ implies $S \vdash G$ in L.*

https://doi.org/10.1017/S1446788700022217 Published online by Cambridge University Press

The proof for each of these lemmas is a straightforward modification of the corresponding proof by Selman [7] for his language and is therefore omitted. It is in these proofs that several of the rules of inference must be invoked.

COMPLETENESS THEOREM. *Let L be a GIEL as defined above, S be any set of formulas of L, and let F be any particular formula of L. Then*

(i) *for any algebra, A, if S is valid in A then F is valid in A, (that is, if* $A \models S$ *then* $A \models F$)

implies

(ii) F *is provable from* S, *using the axioms and rules of inference given earlier* ($S \vdash F$).

PROOF. We use a standard Henkin-type argument.

(1) Suppose F is a closed atomic formula. We wish to form $S1$, the maximal set of formulas containing S but not entailing F. If the set of function symbols is countable, this can be done by enumerating the formulas and then enlarging S in steps, by adding successively those formulas which do not make the current set entail F. Where this set is not countable, Zorn's lemma ensures that such an $S1$ exists even where its construction is not known.

Form the interpolation, I, which has domain all the closed terms. In it each constant is interpreted as itself and the binary relation R is defined by pRq if and only if $p = q$ is in $S1$.

The algebra we shall construct, in which S is valid but F is not, will be I/R. It is easy to see that by R1 and R2 no formula and its negation can both be in $S1$. Now we obviously require that pRq fails to hold if and only if $-p = q$ is in $S1$, that is we must know that either $p = q$ is in $S1$ or $-p = q$ is in $S1$. Assume the contrary, that there is some closed equation, E, with neither E nor $-E$ in $S1$. Then since $S1$ is a maximal set of formulas not entailing F, we have both

(i) $S1 \cup E \vdash F$,

(ii) $S1 \cup -E \vdash F$.

Since $S1$ does not entail $-E$, we have $S1$ does not entail $(p = p \rightarrow -E)$ from *modus ponens* and R2. Therefore $S1 \cup \{p = p \rightarrow -E\}$ entails F by maximality of $S1$, and by contraposition this yields

(iii) $S1 \cup E \rightarrow -p = p \vdash F$.

Now Lemma 2 with these parts (i) and (iii) together imply the contradiction, $S1$ entails F.

(2) Suppose F is any nonatomic closed formula, $F1 \rightarrow F2$. From Lemma 1 $S \cup F1$ does not entail $F2$, so we are done because of part 1 of this theorem.

(3) Suppose F is not closed and $S \nvdash F$ in L. Using Lemma 3 we have $S \nvdash F+$ in $L+$, and by 1 or 2 above, we are done.

https://doi.org/10.1017/S1446788700022217 Published online by Cambridge University Press

Thus in all possible cases we have constructed an algebra in which our unprovable formula is not valid.

Having established the completeness of the given calculus for GIEL formulas the obvious question which follows is the decidability of the entailment relation. It is not hard to see that even at the pure equational level (that is without the if...then...) the entailment relation is already undecidable and so entailment in GIEL systems is also. We outline the first reduction.

By using Skolem functions to eliminate existential quantifiers the group axioms can be formulated as equational formulas. The celebrated unsolvability of the word problem immediately means that the problem of deciding if A can be deduced from S is unsolvable.

References

[1] J. R. Buchi, 'Weak second order arithmetic and finite automata', *Z. Math. Logik Grundlag. Math.* **6** (1960), 66–92.

[2] J. A. Goguen, J. W. Thatcher and E. G. Wagner, 'An initial algebra approach to the specification, correctness and of abstract data types', *Current Trends in programming methodology* IV, R. T. Yeh (ed.), pp. 80–149 (Prentice Hall, New Jersey, 1978).

[3] J. V. Guttag, 'Some extensions to algebraic specifications', pp. 63–67, *Proceedings of ACM conference* (Sigplan Notices 12, 1977).

[4] M. E. Majster, 'Limits of the 'algebraic' specification of abstract data types', (ACM-Sigplan Notices 12, 1977).

[5] H. Samet, 'Effective on-line proofs of equalities and inequalities of formulas', pp. 28–32, *IEEE Transactions on Computers* **29** (1980).

[6] A. Selman, 'Completeness of calculi for axiomatically defined classes of algebras', *J. Symbolic Logic* **37** (1972), 433.

[7] A. Selman, 'Completeness of calculi for axiomatically defined classes of algebras', *Algebra Universalis* **2(1)** (1972), 20–32.

[8] J. W. Thatcher, E. Wagner and J. Wright, 'Data types specification: parametrization and the power of specification techniques', pp. 119–132, *10th ACM Symposium on the Theory of Computing*, 1978.

Basser Department of Computer Science
University of Sydney
Sydney, N.S.W. 2006
Australia