A CANONICAL FORM ALGORITHM FOR PROVING EQUIVALENCE OF CONDITIONAL FORMS

Hanan SAMET
Computer Science Department, University of Maryland, College Park, Maryland 20742

Received 31 August 1977; revised version received 8 November 1977

Canonical form, equivalence, conditional forms

1. Introduction

In [1] axioms and canonical form algorithms for proving equivalence for the theory of conditional forms are presented. These algorithms form the foundation of [2] where they are extended to enable proving the correctness of compilation. The algorithms are distinguished on the basis of whether or not strong or weak equivalence is desired. In the case of strong equivalence, an additional set of axioms was introduced. In this note we prove that the additional axioms were unnecessary, and that there is essentially no difference in the method of proof for strong and weak equivalence. We also present a simpler algorithm for proving strong equivalence.

2. Preliminaries

The basic entity is a generalized boolean form (gbf), formed as follows:

(1) Variables are divided into propositional variables p, q, r, etc. and general variables x, y, z, etc.

(2) $(p \rightarrow x, y)$ is called an elementary conditional form of which p, x, and y are called the premise, conclusion, and alternative respectively.

(3) A variable is a gbf, and if it is a propositional variable, then it is called a propositional form (pf).

(4) If p is a pf, and x and y are gbf's, then $(p \rightarrow x, y)$ is a gbf. If, in addition, x and y are pf's, so is $(p \rightarrow x, y)$.

The value of a gbf x for given values (T, F, or undefined) of the propositional variables will be T or F in case x is a pf and a general variable otherwise. This value is determined for a gbf $(p \rightarrow x, y)$ according to Table 1.

Two gbf's are said to be *strongly equivalent* (denoted by $=$) if they have the same value for all values of the propositional variables in them including the case of undefined propositional variables. The gbf's are *weakly equivalent* (denoted by $=_w$) if they have the same values for all values of the propositional variables when these are restricted to T and F.

There are two equivalence rules which enable the use of equivalences to generate other equivalences. These rules hold for both weak and strong equivalences.

(a) If $x = y$ and x_1 and y_1 are the results of uniformly substituting any gbf for any variable in x and y, then $x_1 = y_1$. This is known as the *Rule of Substitution*. This enables the use of the about-to-be-presented axioms as schemas.

(b) If $x = y$ and x is a subexpression of z and w is the result of replacing occurrences of x in z by an occurrence of y, then $z = w$. This is known as the *Rule of Replacement*. Note the similarity to substitution of equals for equals.

Table 1
Conditional form values

value (p)	value ($(p \rightarrow x, y)$)
T	value (x)
F	value (y)
undefined	undefined

Equivalence can be tested by the method of truth tables as in propositional calculus, and also by using the following eight equations as axioms to transform any gbf into an equivalent one. This transformation is aided by using the rules of substitution and replacement.

(1) $(p \to a, a) =_w a$

(2) $(T \to a, b) = a$

(3) $(F \to a, b) = b$

(4) $(p \to T, F) = p$

(5) $(p \to (p \to a, b), c) = (p \to a, c)$

(6) $(p \to a, (p \to b, c)) = (p \to a, c)$

(7) $((p \to q, r) \to a, b) = (p \to (q \to a, b), (r \to a, b))$

(8) $(p \to (q \to a, b), (q \to c, d)) =$

$$(q \to (p \to a, c), (p \to b, d)).$$

Note that all are strong equivalences with the exception of the first which is a weak equivalence. Thus our previous statement about transforming a gbf into an equivalent one should be reworded to preclude the use of axiom (1) in proving strong equivalence.

In fact these rules and axioms can be used to transform any gbf into a weak canonical form defined as follows:

If $p_1, p_2, ..., p_n$ are the variables of the gbf, p, taken in some arbitrary order, then p can be transformed into the form:

$(p_1 \to a_0, a_1)$, where each a_i has the form:

$a_i = (p_2 \to a_{i0}, a_{i1})$ and in general for each $k = 1, ..., n-1$

$a_{i_1 ... i_k} = (p_{k+1} \to a_{i_1 ... i_k 0}, a_{i_1 ... i_k 1})$

and each $a_{i_1 ... i_n}$ is a truth value or a general variable. Thus in this canonical form, the 2^n cases of the truth or falsity of $p_1, p_2, ..., p_n$ are explicitly exhibited. Another way of viewing the canonical form is to think of it as a binary tree whose non-terminal nodes are propositional variables and whose terminal nodes represent computations.

3. Equivalence algorithms

The algorithm for obtaining a canonical form for weak equivalence is as follows:

(1) Use axiom (7) repeatedly until in every subexpression the p in $(p \to x, y)$ consists of a single propositional variable. Also apply axioms (2) and (3) whenever possible.

(2) The propositional variable p_1 is moved to the front by repeated application of axiom (8). There are three cases:

(a) $(q \to (p_1 \to a, b), (p_1 \to c, d))$ to which axiom (8) is directly applicable.

(b) $(q \to a, (p_1 \to c, d))$ where axiom (8) is applicable after axiom.

(c) $(q \to (p_1 \to a, b), c)$ which is handled in the same manner as case (b) – i.e., axiom (8) is applicable after axiom (1) is used to yield the form $(q \to (p_1 \to a, b), (p_1 \to c, c))$.

(3) Once the main expression has the form $(p_1 \to x, y)$, then all p_1's which occur in x and y are moved to the front of x and y by using the same procedure. The p_1's which have been moved are then eliminated by using axioms (5) and (6). p_2 is then moved to the front of x and y, using axiom (1) if necessary to insure at least one occurrence of p_2 in each of x and y. This process is continued until the canonical form is achieved.

There is also a canonical form for strong equivalence. The difference is that the propositional variable p_1 may not be chosen arbitrarily, but instead must be an *inevitable variable* of the gbf a. An inevitable variable of a gbf $(p \to x, y)$ is defined to be either the first propositional variable or else an inevitable variable of both x and y. Note that once again the canonical form is of the form $(p_1 \to x, y)$ where x and y do not contain p_1 and are themselves in canonical form.

The algorithm for the derivation of the canonical form for strong equivalence is identical to the algorithm given for weak equivalence. This statement is in contrast with the algorithm given in [1] where two axioms were added in addition to the restriction that axiom (1) could not be used. The axioms that were unnecessarily added were:

(9) $(p \to (q \to a, b), c) = (p \to (q \to (p \to a, a),$
$(p \to b, b)), c)$

(10) $(p \to a, (q \to b, c)) = (p \to a, (q \to (p \to b, b),$
$(p \to c, c)))$.

The algorithm for obtaining a canonical form for weak equivalence was modified to be valid for strong equivalence by use of these axioms so that occurrences of an inevitable variable, say p_1, in the conclusion or alternative can be eliminated when substitution and replacement are used. This was given as an alternate solution to the obvious use of the general rule that any occurrence of the premise in the conclusion can be replaced by T and occurrences in the alternative by F. The motivation behind the proposed solution is a possible reluctance to make use of the meta-notion of T and F and to work strictly by using formulas not involving the introduction of T or F. The revised algorithm stated that it is desired to replace all occurrences of the premise in the conclusion by T, and occurrences in the alternative by F. This is accomplished by finding the clause (i.e., conclusion or alternative) which contains the objectionable atom. If it is in the conclusion, then axiom (9) is used; and if it is in the alternative, then axiom (10) is used. Next, axioms (9) and (10) are applied in the manner described in the previous statement until the objectionable atom, say p, occurs as the inner p of one of the forms $(p \to (p \to a, b), c)$ or $(p \to a, (p \to b, c))$. In either case the objectionable p is removed by using axioms (5) or (6) and p's that were introduced by applications of axioms (9) and (10) are removed by repeated application of axiom (8). Notice that the solution characterized by replacement with T and F is much simpler in a computational sense as will be seen in a subsequent example.

Actually, the algorithm differs from that given for the weak equivalence case in that step (2) now states: choose any inevitable variable, say p_1, and put the gbf in the form $(p_1 \to x, y)$ by using axiom (8). Note that axioms (9) and (10) were added for the specific reason that axiom (1) could not be used. In fact, there is no need at all for axioms (9) and (10) since, for example, axiom (9) can be shown to be true in the following manner:

$$(p \to (q \to (p \to a, a), (p \to b, b)), c) =$$
$$= (p \to (p \to (q \to a, b), (q \to a, b)), c)$$
by axiom (8)
$$= (p \to (q \to a, b), c) \quad \text{by axiom (5).}$$

The same can be said for axiom (10).

Therefore, the algorithm is revised as follows:

(1) Use axiom (7) to get all premises as propositional variables.

(2) Choose any inevitable variable, say p_1, and put the gbf in the form $(p_1 \to x, y)$ by using axiom (8).

(3) Eliminate occurrences of p_1 in x and y. If p_1 occurs in any conclusion part of a gbf is x or y, say conc, then introduce in the alternative clause, say alt, $(p_1 \to \text{alt}, \text{alt})$. Similarly, if p_1 occurs in any alternative clause in x or y, then introduce in the conclusion clause, say conc, $(p_1 \to \text{conc}, \text{conc})$. Next, apply axiom (8). Repeat step (3) until one of axioms (5) or (6) is applicable.

Proof of the validity of the change in Step (3). Since we have passed the point where p_1 is undefined, axiom (1) – i.e., $(p_1 \to a, a) = a$ is valid since p_1 is now defined. Remember the latter was the only reason that axiom (1) was not applicable to strong equivalence. The use of the equivalence $(p_1 \to a, a) = a$ is valid according to the rule of replacement.

Thus it is seen that actually there is no difference in the method of proof for strong equivalence and weak equivalence and that the canonical forms can be the same if inevitable variables are used. This leads to the following:

... *If two gbfs are strongly equivalent, then they are also weakly equivalent.*

Proof. Order the variables according to inevitability. This is one of the acceptable orders and weak equivalence follows. q.e.d.

As an example of the process of determining the equivalence of two gbf's we show $(p \to (q \to (p \to x, y), b), c) = (p \to (q \to x, b), c)$ by means of axioms (9) and (10) and also by means of the revised algorithm.

Axioms (9) and (10) method:

$$(p \to (q \to (p \to x, y), b), c)$$
$$= (p \to (q \to (p \to (p \to x, y), (p \to x, y)),$$
$$(p \to b, b)), c) \text{ by axiom (9)}$$
$$= (p \to (q \to (p \to x, (p \to x, y)), (p \to b, b)), c)$$
by axiom (5)

$= (p \to (q \to (p \to x, y), (p \to b, b)), c)$

by axiom (6)

$= (p \to (p \to (q \to x, b), (q \to y, b)), c)$

by axiom (8)

$= (p \to (q \to x, b), c)$ by axiom (5)

Revised algorithm method:

$(p \to (q \to (p \to x, y), b), c)$

$= (p \to (q \to (p \to x, y), (p \to b, b)), c)$

by axiom (1)

$= (p \to (p \to (q \to x, b), (q \to y, b)), c)$

by axiom (8)

$= (p \to (q \to x, b), c)$ by axiom (5)

The canonical form algorithm for strong equivalence can be further simplified by revising steps (2) and (3) as follows. Once an inevitable propositional variable, say p_1, of the gbf $(p \to x, y)$ has been found, we simply replace the gbf by an application of axiom (1) – i.e., $(p \to x, y) = (p_1 \to (p \to x, y), (p \to x, y))$. To the quantity on the right we apply axioms (5) and (6) until they can be applied no further. It should be clear that this procedure is equivalent to the previously given algorithm. The only difference is that the latter proceeds to propagate the inevitable variable out from inside the gbf; while the former first brings the variable out and then applies the redundant predicate removal axioms (i.e., (5) and (6)).

References

[1] J. McCarthy, A Basis for a mathematical theory of computation, in: Braffort and Hirschberg, Eds., Computer Programming and Formal Systems (North-Holland, Amsterdam, 1963).

[2] H. Samet, Automatically Proving the Correctness of Translations Involving Optimized Code, Ph.D. Thesis, Stanford Artificial Intelligence Project Memo AIM-259, Computer Science Department, Stanford University (1975).