

TEMPORAL REASONING AND TERMINATION OF PROGRAMS

Luis Fariñas-del-Cerro

Laboratoire des Langages et Systèmes Informatiques
Université Paul Sabatier, Toulouse, France

ABSTRACT

This paper deals with the relationship between the termination of programs and the validity of certain modal formulas. We give a complete proof procedure for these formulas, which will allow to bring the correctness of these programs back to a problem of automated deduction in modal logic.

Many modal logics of programs have been developed during these last years [PR, HKP, M2]. Particular attention has been paid to their ability to express many properties of programs [HKP, M2].

This paper is concerned with the relationship of termination problem for regular programs to the validity of certain formulas in modal logic.

For these formulas we present a proof procedure, very close to the resolution procedure for first-order predicate calculus. We will use this procedure to prove the termination of programs. As in Lucid [AW] we can consider the set of modal formulas with the proof procedure as a programming language. This way of doing things permits direct reasoning about programs, from their direct manipulation, rather than indirectly via another language.

I. Preliminaries

We shall consider as in [M2] the modal system S4, for reasoning about programs.

For our formalization of quantificational S4, we start with denumerably infinite lists of individual variables $x_1, x_2, x_3, \dots$, n -adic function symbols $f^n, g^n, h^n, \dots$, and n -adic predicate symbols $p^n, q^n, r^n, \dots$.

Atomic formulas as well as negations of atomic formulas are called literals. We adopt the prime symbols $\&$ (conjunction), $\sim$ (negation) L (necessity) and (x_i) (universal quantification). We use the usual notions of terms and well-formed formulas. Let $A, B, C, \dots$ be arbitrary formulas. For each A , MA is defined as usual as $\sim L \sim A$. The axiomatization of quantificational S4 is obtained by

adding to a habitual formalization of lower predicate calculus (LPC) the following axiom schemes and the following rule of inference :

- . $L(A \rightarrow B) \rightarrow .LA \rightarrow LB$
- . $LA \rightarrow A$
- . $LA \rightarrow LLA$
- . $(x)LA \rightarrow L(x) A$
- . A / LA

Following the example of Kripke, we define a quantificational S4 modal structure as an ordered triple (G, K, R) , where K is a set, R is a relation on K and G is a distinguished element of K , together with a function Ψ which assigns to each H a set $\Psi(H)$, called domain in H . We shall further specify that all domains are identical. The interpretation Γ of A at H is defined as in LPC by induction on the number of logical symbols in A . And $\Gamma(LB, H) = t$ iff $\Gamma(B, H') = t$ for every $H' \in K$ such that HRH' , otherwise $\Gamma(LB, H) = f$.

A sentence A is said to be true in a model U associated with a modal structure (G, K, R) if $\Gamma(A, G) = t$; it is said to be false in that model if $\Gamma(A, G) = f$. A is said to be valid iff it is true in all its models (for every modal structure), and unsatisfiable iff it is false in all its models.

A set of formulas is said to be consistent if there is no finite subset such that the disjunction of negations of its elements is a theorem.

II. Programs and modalities

2.1. Assume that a program is represented by a directed graph whose nodes are the labels of this program and whose arcs represent transitions between labels. In the graphs there is only a start node (l_s) and a terminal node (l_t) [M1]. For an arc (l_1, l_2) the transition has the general form as follows :

$$C_{(l_1, l_2)}(\bar{x}, \bar{y}) \rightarrow (z := f_{(l_1, l_2)}(\bar{x}, \bar{y}))$$

$$l_1 \xrightarrow{\hspace{10em}} l_2$$

where $\bar{x} = (x_1, \dots, x_n)$ is the input variables,

$\bar{y} = (y_1, \dots, y_m)$ is the program variables and $\bar{z} = (z_1, \dots, z_h)$ is the output variables. $C_{(l_1, l_2)}(\bar{x}, \bar{y})$ is a condition under which the assignment $\bar{z} := f_{(l_1, l_2)}(\bar{x}, \bar{y})$ is realized. We assumed that each node l_i , one and only one of the condition, corresponding to the arcs leaving l_i , is true.

Since modal logic, like classical logic [M1], does not provide a direct tool for speaking about programs, such as the logics of programs [HO, S, PR], we must associate to each arc of the graph a modal formula :

$$I((at(l_1) \& C_{(l_1, l_2)}(\bar{x}, \bar{y}) \& p(\bar{x}, \bar{y})) \rightarrow M(at(l_2) \& p(\bar{x}, f_{(l_1, l_2)}(\bar{x}, \bar{y}))))$$

where $at(l_i)$ is a proposition corresponding to the label l_i . This formula is relative to the forward assignment.

2.2. If P is a program and F_P the set of modal formulas associated to the graph of P , then F_P is satisfiable.

This is true since the set of formulas of F_P with the general form $M(at(l) \& p)$ is consistent [HC].

2.3. A program P is said to be totally correct with respect to the specification (p, q) if for every input $\bar{x}_0$, satisfying $p(\bar{x}_0)$, the computation of P terminates and the final values $(\bar{z}_0)$, upon termination, satisfy $q(\bar{z}_0)$. This property is expressed by the formula (noted by F_{TC}) :

$$(at(l_s) \& p(\bar{x}_0)) \rightarrow M(at(l_t) \& q(\bar{z}_0))$$

In general, in order to verify this property, we need the following informations :

- . the formulas describing the execution of program P ,
- . the formulas (noted by F_R) concerning conditions, assignments and inputs (frame axioms in [M2]) are as follows :

$$L(x > 0) \\ L(x = 0) \vee M(x \neq 0)$$

Then we can consider these formulas that express the material properties, (depending on the labels) and formal properties not depending on the labels).

2.4. Let P be a program. P terminate iff $F_P \& F_R \& \neg F_{TC}$ is unsatisfiable.

The proof can be obtained by a similar reasoning to [M1].

III. Deduction

3.1. We will now give a deduction method for the formulas of quantificational S_4 , whose variables in the scope of the modal operators are free (noted by S_4^+).

We consider the formulas in the Skolem normal form :

$$(x_1), \dots, (x_n) (A_1 \& \dots \& A_m)$$

- where each (x_i) is an universal quantifier (the existential quantifiers have been eliminated by Skolem functions in the usual way [RS]) and :

$$- m > 1$$

$$- n \geq 0$$

- each A_i $1 \leq i \leq m$ (clause in the classical terminology [R]) is a disjunction of the form :

$$A_i = LD_i \vee \dots \vee LD_k \vee MF_1 \vee \dots \vee MF_{k'} \vee P_1 \vee \dots \vee P_{k''}$$

- where each P_i $1 \leq i \leq k''$ is a literal, each F_i $1 \leq i \leq k'$ is a conjunction of terms possessing the general form of the clauses, and each D_i $1 \leq i \leq k$ is a disjunction that possesses the general form of the clauses.

3.2. There is an effective procedure, for constructing, for any given formula of S_4^+ , an equivalent formula in Skolem normal form.

3.3. Let $E = \{E_1, \dots, E_n\}$ $1 \leq n < \omega$, be a finite set of expressions. A substitution σ is called a unifier of the set E if $E_i \sigma = E_j \sigma$ for $i, j \in \{1, \dots, n\}$. A unifier σ is called the most general unifier if for any unifier σ' of E there is a substitution σ'' such that $\sigma' = \sigma \cdot \sigma''$ [R].

3.4. We consider the set of rules Σ_i $i=1, \dots, 4$ and Γ_i $i=1, 2$ defined recursively as follows:

1. Elementary rule :

$$\Sigma_1[(D_1.D_2) ; F] = (\Sigma_1[D_1 ; F] . D_2) \\ \text{provided } . \text{ is } \vee \text{ or } \&$$

2. Extending rules :

$$\Sigma_2[LD ; \Delta F] = \Delta(\Sigma_1[D ; F]) \text{ provided } \Delta \text{ is } L, \\ M \text{ or the empty symbol}$$

$$\Gamma_1[A\{M(D \& F)\}] = A\{M(\Sigma_1[D ; F])\}$$

$A\{M(D \& F)\}$ means that $M(D \& F)$ is a subformula of A .

3. Transforming rule :

$$\Sigma_3[LD ; F] = \Sigma_1[LLD ; F]$$

4. Classical rule :

$$\Sigma_4[p ; \neg p] = \emptyset$$

5. Elimination of $\emptyset$:

$$\Gamma_2[\emptyset \& F] = \emptyset$$

$$\Gamma_2[\emptyset \vee D] = D$$

$$\Gamma_2 \Delta[\emptyset] = \emptyset \text{ provided } \Delta \text{ is } L \text{ or } M.$$

Let A_1 and A_2 be two clauses. We say that A_1 and A_2 are resolvable if $\Sigma_i[A_1; A_2]$ is defined (i.q. the classical rule is used).

Let A_i be a clause. We say that A_i is resolvable if $\Gamma_i[A_i]$ is defined.

The remaining term obtained from A_1 (A_1 and A_2) after application of Γ_i and (or Σ_i) is represented by $\Gamma_1[A_1]$ ($\Sigma_i[A_1; A_2]$)

3.5. Let $A \vee A_1$ and $A' \vee A'_1$ be two clauses with no variables in common. The inference rules :

$$1. \frac{A \vee A_1, A' \vee A'_1}{\sigma \Sigma_i[A; A'] \vee \sigma A_1 \vee \sigma A'_1}$$

$$2. \frac{A \vee A_1 \vee A_2}{\sigma \Sigma_i[A] \vee \sigma A_1}$$

will be applied if A and A' (or A) are (is) resolvable. Where σ is the most general unifier used in the classical rule.

3.6. Let $A \vee A_1 \vee A_2$ be a clause.

The inference rule :

$$3. \frac{A \vee A_1 \vee A_2}{A \sigma \vee A_i \sigma}$$

will be applied if A_1 and A_2 have most general unifier σ .

3.4. Let S be a finite set of clauses, a deduction of A from S is a finite sequence of clauses $A_1, \dots, A_n$, such that :

- A_n is A
- A_i $1 \leq i < n$ is
 - an element of S , or
 - a factor (rule 3) of A_j , $j < i$, or
 - a resolvent (rules 1 and 2) of A_j and A_k or $j, k < i$

3.5. A set S of clauses is unsatisfiable iff there is a deduction of the empty clause from S .

The proof can be obtained as in $[F]$ or $[0]$.

IV. Example

Consider the program P (fig.1) over the integers, that adds an integer x_1 to a natural num-

ber x_2 .

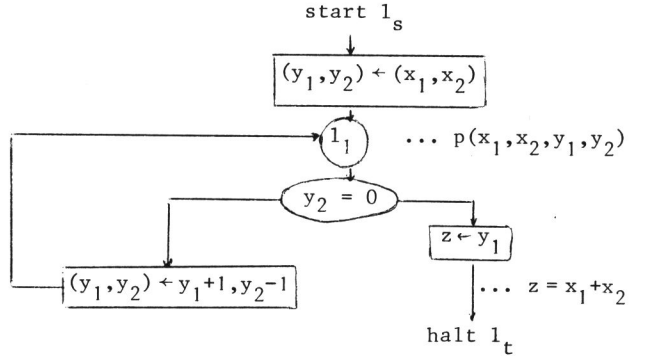


Fig. 1

The clauses as set out in 3. corresponding to this program are of the form :

1. $L((\text{nat}(l_s) \vee \neg p(x_1, x_2, x_1, x_2) \vee M(\text{at}(l_1) \& p(x_1, x_2, y_1, y_2)))$
2. $L((\text{nat}(l_1) \vee \neg y_2 > 0 \vee \neg p(x_1, x_2, y_1 + 1, y_2 - 1)) \vee M(\text{at}(l_1) \& p(x_1, x_2, y_1, y_2)))$
3. $L((\text{nat}(l_1) \vee \neg y_2 = 0 \vee \neg p(x_1, x_2, y_1, y_2)) \vee M(\text{at}(l_t) \& p(x_1, x_2, y_1, 0)))$

where $p(x_1, x_2, y_1, y_2)$ is the assertion $x_1 + x_2 = y_1 + y_2$.

Using the induction formula :

4. $M(\text{at}(l_1) \& p(x_1, x_2, y_1, 0) \& L(\text{nat}(l_t) \vee \neg p(x_1, x_2, y_1, 0)) \vee M((\text{nat}(l_1) \vee \neg p(x_1, x_2, y_1, y_2)) \& y_2 > 0 \& p(x_1, x_2, x_1 + 1, y_2 - 1)) \vee L(\text{nat}(l_1) \vee \neg p(x_1, x_2, y_1, y_2) \vee M(\text{at}(l_t) \& p(x_1, x_2, y_1, 0)))$

And the frame clauses :

5. $M(x \neq 0) \vee L(x = 0)$
6. $L(\neg 0 \neq 0)$

The program considered will be totally correct if this set of clauses with the following formula :

7. $\neg((\text{nat}(l_s) \vee \neg p(x_1, x_2, x_1, x_2)) \vee M(\text{at}(l_t) \& p(x_1, x_2, y_1, 0)))$

is unsatisfiable.

Using the method given in 3. we obtain many steps, but here we will give the alternate steps as follows :

8. $MM(at(1_t) \& p(x_1, x_2, y_1, 0) \& L(\neg at(1_t) \vee p(x_1, x_2, y_1, 0))) \vee M((at(1_1) \vee \neg p(x_1, x_2, y_1, y_2)) \& at(1_1) \& y_2 > 0 \& p(x_1, x_2, x_1+1, y_2-1)) \vee L(\neg at(1_1) \vee \neg p(x_1, x_2, y_1, y_2)) \vee M(at(1_t) \& p(x_1, x_2, y_1, 0))$
from 4., 3., 5 and 6.
9. $M((\neg at(1_1) \vee \neg p(x_1, x_2, y_1, y_2)) \& at(1_1) \& y_2 > 0 \& p(x_1, x_2, x_1+1, y_2-1)) \vee L(\neg at(1_1) \vee \neg p(x_1, x_2, y_1, y_2)) \vee M(at(1_t) \& p(x_1, x_2, y_1, 0))$
from 8.
10. $MM(\neg at(1_1) \vee y_2 > 0 \vee \neg p(x_1, x_2, y_1+1, y_2-1) \& at(1_1) \& y_2 > 0 \& p(x_1, x_2, x_1+1, y_2-1)) \vee L(\neg at(1_1) \vee \neg p(x_1, x_2, y_1, y_2)) \vee M(at(1_t) \& p(x_1, x_2, y_1, 0))$
from 9. and 2.
11. $\neg at(1_s) \vee \neg p(x_1, x_2, y_1, y_2)$
from 10., 7. and 1.

And from 11. and $at(1_s) \& p(x_1, x_2, y_1, y_2)$ we obtain the empty clause.

This proof is very close to the Lucid [AW] version of intermittent assertion proofs [MW].

The different rules (concatenation, consequence...) can be obtained by putting together several steps of our method.

In general, we can consider the set of modal formulas with the proof procedure as a programming language, that permits direct reasoning about the programs, from their direct manipulation.

BIBLIOGRAPHY

- [AW] Ascroft, E., Wadge, W. "Intermittent assertion proofs in Lucid". in IFIP 77, pp. 723-726.
- [F] Fariñas-del-Cerro, L. "A deduction method for modal logic". European AI Conference, 11-14 July 1982, Orsay.
- [HKP] Harel, Kozen, Parikh, "Process logic, expressiveness, decidability, completeness". in FOCS 80, pp. 129-142.
- [HO] Hoare, "An axiomatic basic of computer programming". in Com. ACM, 12, 10, (1969).
- [HC] Hugues, Cresswell, "An introduction to modal logic", Mathuem & Co., London, (1978).
- [M1] Manna, Z. "Properties of programs and first order predicate calculus", in J.A.C.M., 16, 2, (1969), pp. 244-255.
- [M2] Manna, Z. "Logics of Programs", in Proc. IFIP 80, North-Holland, pp. 41-52.
- [MW] Manna, Z., Waldinger, R. "Is 'sometime' sometime better than 'always' ? : Intermittent assertions in proving program correctness", in Com. ACM, 21, 2, (1978), pp. 159-172.
- [O] Orłowska, E. "Resolution systems and their applications", I, II, Fundamenta Informaticae, (1980), pp. 235-267, 333-362.
- [PR] Pratt, V. R. "Semantical Considerations on Floy's-Hoare Logic", in Proc. 17th Ann. IEEE Symp. on Foundations of Comp. Sci., (1976); pp. 109-121.
- [RS] Rasiowa, Sikorski, "The mathematics of mathematics", Warszawa, (1963).
- [R] Robinson, "A machine oriented logic based on the resolution principle", in J.A.C.M., 12, (1965), pp. 23-41.
- [S] Salwicki, "Formalized algorithm language", in Bul. Ac. Pol. Sci., 18, 5, (1970), pp. 227-232.