RAIRO. INFORMATIQUE THÉORIQUE

LUIS FARIÑAS DEL CERRO

Les modalités de la correction totale

RAIRO. Informatique théorique, tome 16, n° 4 (1982), p. 349-363

<http://www.numdam.org/item?id=ITA_1982__16_4_349_0>

R.A.I.R.O. Informatique théorique/Theoretical Informatics
(vol. 16, n° 4, 1982, p. 349 à 363)

LES MODALITÉS DE LA CORRECTION TOTALE (*)

par Luis FARIÑAS DEL CERRO (1)

Communiqué par Erwin ENGELER

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 automatic theorem-proving in modal logic.*

Résumé. — *Cet article concerne la relation entre la terminaison de programmes et la validité de certaines formules modales. Une procédure complète de démonstration pour ces formules est présentée, ce qui nous permettra de ramener la correction de ces programmes à un problème de démonstration automatique en logique modale.*

I. INTRODUCTION

Il est bien connu, depuis quelques années, que les logiques modales sont un outil très intéressant pour raisonner sur les programmes [AW1, B, HKP, PR]. Un grand nombre de systèmes (formalismes ou semi-formalismes) ont été développés [H, PR, K, E, EP] et un intérêt particulier a été consacré à leur facilité à exprimer un grand nombre de propriétés relatives au comportement de programmes [M2, MP]. Ceci est réalisé grâce aux opérateurs modaux qui permettent de raisonner sur des séquences de temps (suite de situations).

Sommairement, une formule modale est une expression construite à partir de symboles de propositions, de prédicats, de fonctions, de constantes d'individus, de variables, de symboles logiques classiques, de quantificateurs et de l'opérateur modal nécessaire $\mathscr{L}$ (comme d'habitude l'opérateur modal possible $\mathscr{M}$ est défini comme $\sim \mathscr{L} \sim$). Un modèle U d'une formule modale consiste en un ensemble de situations S et en une relation binaire R (relation d'accessibilité) sur l'ensemble des situations. Dans chaque situation nous

(*) Reçu en septembre 1981.
(1) Langages et systèmes informatiques, Équipe compréhension du raisonnement naturel, Université de Toulouse Le Mirail.

R.A.I.R.O. Informatique théorique/Theoretical Informatics, 0399-0540/1982/349 /$ 5.00

disposons d'un domaine et d'une interprétation classique sur le domaine de tous les symboles du premier ordre. La signification de l'opérateur modal dans une situation s, (A/s note que la valeur de vérité de la formule A dans la situation s est vraie) est la suivante :

$$\mathscr{L}\, A/s \text{ si et seulement si } A/s', \text{ pour tout } s' \text{ tel que } sRs'.$$

La valeur de vérité associée à une formule modale dans un état consiste à répéter la règle ci-dessous pour les opérateurs modaux, et à évaluer de façon classique dans le même état les sous-formules classiques.

Une formule qui est vraie pour chaque univers et pour chaque état est appelée valide et insatisfaisable si elle est la négation d'une formule valide (*voir* [HC] pour plus de détails).

Considérer les programmes comme générateurs de séquences d'exécutions, pour lesquels on passe d'un état à l'autre par l'exécution d'une instruction, suggère immédiatement l'utilisation de la logique modale pour raisonner sur les programmes, en faisant correspondre d'une part la notion de situation à celle de l'état du programme considéré et d'autre part la relation d'accessibilité à celle de dérivabilité entre les différents états du programme (*voir* [MP]).

Puisque nous nous intéressons à l'étude de la correction totale (qui est donc une propriété dite *éventuelle* [MP], il n'est pas nécessaire, comme nous le verrons par la suite, d'envisager un formalisme modal qui nous permette de décrire exactement l'exécution des programmes. Ceci nous conduit, comme en [MP], à considérer comme système de base pour raisonner sur les programmes le système modal $S4$ avec des quantificateurs dont l'axiomatique est une extension de toute formalisation habituelle du calcul classique aux schemas d'axiomes et à la règle d'inférence suivante :

- $\mathscr{L}(A \to B). \to . \mathscr{L} A \to \mathscr{L} B$;
- $\mathscr{L} A \to A$;
- $\mathscr{L}\mathscr{L} A \to \mathscr{L} A$;
- $A/\mathscr{L} A$;
- $\forall x\, \mathscr{L} A \to \mathscr{L} \forall x A$.

Du point de vue sémantique, cette axiomatique contraint d'une part R à être une relation reflexive et transitive, et les domaines associés à chaque situation d'autre part à être identiques [HC].

Un ensemble de formules est dit consistant si et seulement si il ne possède pas un sous-ensemble fini de formules telles que la disjonction de leurs négations est un théorème.

II. DESCRIPTION DE PROGRAMMES A L'AIDE DE MODALITÉS

2.1. Définitions préliminaires

Un graphe orienté et fini G est composé :

– d'un ensemble fini X;

– d'une application multivoque Γ de $x (x \in X)$ dans X.

Les éléments x de X sont appelés sommets, les éléments $\Gamma(x)$ sont appelés successeurs de x qui est leur antécédent.

Un ensemble de deux éléments b et c de X, tels que $c \in \Gamma(b)$ (noté (b, c)) est appelé arc du graphe. b est l'origine de l'arc et c l'extrémité terminale. Le demi-degré intérieur (resp. extérieur) d'un sommet x est le nombre d'arcs ayant x comme extrémité terminale (origine).

Un chemin du graphe est une séquence d'arc dont l'extrémité terminale de l'un est l'origine de l'autre.

Les programmes considérés sont définis par :

– un graphe orienté et fini tel que :

• il possède un seul sommet d de X, dont le demi-degré intérieur est égal à zéro, appelé sommet de départ,

• il possède un seul sommet a de X dont le demi-degré extérieur est égal à zéro, appelé sommet de sortie;

– des ensembles de variables :

• $\bar{x} = (x_1, \ldots, x_n)$ variables d'entrée,

• $\bar{y} = (y_1, \ldots, y_m)$ variables du programme,

• $\bar{z} = (z_1, \ldots, z_l)$ variables de sortie;

– et à chaque arc (b, c) est associé :

• une formule sans quantifieurs $C_{(b,c)}(\bar{x}, \bar{y})$ appelée test ou condition,

• une affectation $(\bar{y} := f_{(b,c)}(\bar{x}, \bar{y}))$;

– et pour chaque sommet b, différent du sommet a, l'ensemble des conditions $\{C_{(b,x)}(\bar{x}, \bar{y}) : x \in \Gamma(b)\}$ vérifie ce qui suit :

• la disjonction des conditions $C_{(b,x)}(\bar{x}, \bar{y})$ pour $x \in \Gamma(b)$ est une formule valide,

• et la conjonction de ces conditions prises deux par deux est toujours fausse.

En conséquence chaque arc (b, c) :

$$b \; \underline{C_{(b,c)}(\bar{x}, \bar{y}) \rightarrow (\bar{y} := f_{(b,c)}(\bar{x}, \bar{y}))} \; c$$

représente une instruction du programme de la forme générale :

$$C_{(b,c)}(\bar{x}, \bar{y}) \rightarrow (\bar{y} := f_{(b,c)}(\bar{x}, \bar{y})),$$

qui exprime que, sous la condition $C_{(b,c)}(\bar{x}, \bar{y})$ (éventuellement sans condition), l'affectation $\bar{y} := f_{(b,c)}(\bar{x}, \bar{y})$ est réalisée.

La majeure partie des systèmes déductifs qui permettent de prouver des propriétés de programmes (logique de Hoare, logique dynamique, logique algorithmique...) donne un instrument pour parler directement des programmes (logiques exogènes). Dans toutes ces logiques il existe des variables de programmes. Puisque la logique modale, ainsi que la logique classique ne nous donnent pas un tel instrument directement (logiques endogènes) il faut construire des formules, à partir du graphe, associées au programme [M1].

Pour chaque arc (b, c) nous construisons une formule :

$$(1) \quad \mathcal{M}(\text{at}(l_b) \,\&\, C_{(b,c)}(\bar{x}, \bar{y}) \,\&\, p(\bar{x}, \bar{y})) \rightarrow \mathcal{M}(\text{at}(l_c) \,\&\, p(\bar{x}, f(\bar{x}, \bar{y})),$$

où at (l_b) est une variable propositionnelle qui indique que l_b est une étiquette du programme (relative au sommet b du graphe) et où la formule atomique $p(\bar{x}, f(\bar{x}, \bar{y}))$ dénote la substitution de $f(\bar{x}, \bar{y})$ dans toutes les occurrences de $\bar{y}$.

La formule (1) est interprétée ainsi : s'il existe un instant (maintenant ou plus tard) dans l'exécution du programme, tel que le contrôle se trouve dans l'état b et qu'une certaine condition est vérifiée, alors il existe un instant (maintenant ou plus tard) tel que le contrôle se trouve dans l'état c et que la substitution de $\bar{y}$ par $f(\bar{x}, \bar{y})$ aura été réalisée.

Si nous utilisons l'axiome $A \rightarrow \mathcal{M} A$, nous pouvons déduire de la formule (1) la formule suivante :

$$\text{at}(l_b) \,\&\, C_{(b,c)}(\bar{x}, \bar{y}) \,\&\, p(\bar{x}, \bar{y}) \rightarrow \mathcal{M}(\text{at}(l_c) \,\&\, p(\bar{x}, f(\bar{x}, \bar{y}))).$$

C'est-à-dire, si $\text{at}(l_b) \,\&\, C_{(b,c)}(\bar{x}, \bar{y}) \,\&\, p(\bar{x}, \bar{y})$ est vrai, alors $\text{at}(l_c) \,\&\, p(\bar{x}, f(\bar{x}, \bar{y}))$ sera vrai (maintenant ou plus tard).

Par conséquent, à chaque programme on peut associer un ensemble de formules modales, une pour chaque arc du graphe correspondant, ce qui nous permet de décrire le comportement du programme. En outre, la sémantique

de la logique modale nous donne la possibilité de représenter les étapes de l'exécution des programmes [HKP, M2, MP].

2.2. *Si P est un programme et F est l'ensemble des formules modales représentant la description du programme, alors F est satisfaisable.*

Puisque les formules associées au programme P sont des formules conditionnelles (schématiquement $\mathcal{M}(\text{at}(l_1) \& C \& p) \rightarrow \mathcal{M}(\text{at}(l_2) \& p')$), nous considérons l'ensemble des conséquences de ces formules, qui est consistant, en conséquence satisfaisable [HC]; l'ensemble de formules associé au programme P est donc satisfaisable.

2.3. Etant donné que les programmes calculent dans des domaines déterminés, et peuvent, en particulier, posséder des boucles, nous aurons besoin de formules nous permettant de caractériser ceci [M1, M2]. Ainsi nous aurons des formules relatives aux données (par exemple les variables prennent leurs valeurs sur les entiers), ou des formules relatives aux transformations ou aux conditions (par exemple, la nécessité de disposer d'un axiome d'induction) ([2]).

2.3.1. Le principe d'induction peut être une induction implicite sur les séquences de calculs comme chez Floyd, ou bien une induction explicite sur les données comme chez Burstall. Ici, nous considérons, puisque nous nous intéressons à la correction totale, une instance modale du principe d'induction sur les données.

Soit le principe d'induction sur les entiers positifs :

$$p(0) \& \forall n \geqq 0\, (p(n) \rightarrow p(n+1)) \rightarrow \forall k \geqq 0\, p(k).$$

(De façon similaire, nous pouvons disposer d'un principe d'induction sur un ensemble bien ordonné.)

Nous considérons l'instance modale de ce principe [MP] :

$$(\mathcal{M} p(0) \rightarrow \mathcal{M} q) \& \forall n ((\mathcal{M} p(n) \rightarrow \mathcal{M} q) \rightarrow (\mathcal{M} p(n+1) \rightarrow \mathcal{M} q)) \rightarrow (\exists k\, \mathcal{M} p(k) \rightarrow \mathcal{M} q),$$

c'est-à-dire que p est réalisé au plus k fois et q est assuré.

2.3.2. Il existe un autre type de formules (par exemple, celles relatives à l'imput) qui sont indépendantes des états du programme et qui, par

([2]) Il faut remarquer que le système $S4$ est axiomatisable de façon classique [McT], ce qui veut dire qu'on peut donner une axiomatisation de $S4$ ayant modus ponens comme seule règle d'inférence, ce qui nous permet d'obtenir le théorème de déduction comme corollaire, et par conséquent de suivre une démarche parallèle à celle de la logique classique [M1].

conséquent, sont vraies à tous les états de l'éxécution. Exemples de ce type de formules :

- $\mathscr{L}(x > 0)$;
- $\mathscr{L}(x = 0) \vee \mathscr{M}(x \neq 0)$.

Dans ce type de formules la proposition at (l) n'apparaît pas. En résumé, nous pourrons dire que les propriétés exprimées par les formules 2.1 et 2.3 pourront être de deux types: celles qui expriment des propriétés dites matérielles dont la description dépend de la situation (état), et celles qui expriment des propriétés dites formelles, dont la description est indépendante de l'état [HC].

2.4. Correction totale

La correction totale d'un programme relatif aux spécifications p et p' est assurée si pour chaque imput qui vérifie p la terminaison est garantie et si, pour la valeur obtenue, p' est vérifiée. Ceci peut être représenté la formule modale suivante :

$$\mathscr{M}(\text{at}(l_d) \,\&\, p) \rightarrow \mathscr{M}(\text{at}(l_a) \,\&\, p').$$

Cette formule modale qui exprime la correction totale sera appelée *formule d'arrêt* (et notée F_A).

2.5. *Soient P un programme, F_P l'ensemble de formules modales associé au graphe correspondant à P, et F_I l'ensemble des formules relatives aux données, aux conditions et aux transformations. P se termine si et seulement si la formule $F_P\, F_I \rightarrow F_A$ est une formule valide.*

Démonstration : (a) Si P se termine, quand ce programme arrive au sommet de sortie a, une certaine formule $\mathscr{M}(\text{at}(l) \,\&\, C \,\&\, p) \rightarrow \mathscr{M}(\text{at}(l_a) \,\&\, p')$ lui est associée, et $\mathscr{M}(\text{at}(l) \,\&\, C \,\&\, p)$ doit être vraie; donc $\sim \mathscr{M}(\text{at}(l) \,\&\, C \,\&\, p)$ est fausse. Et, puisque $\mathscr{M}(\text{at}(l_a) \,\&\, p')$ est inconsistante avec la négation de la formule d'arrêt, $F_P \,\&\, F_I \rightarrow F_A$ est une formule valide.

(b) Soit F'_P l'ensemble des formules obtenu à partir de F_P en effaçant les expressions du type $\mathscr{M}(\text{at}(l_d) \,\&\, C \,\&\, p) \rightarrow$ et $\mathscr{M}(\text{at}(l_a) \,\&\, p)$. On peut voir facilement que si $F_P \,\&\, F_I \rightarrow F_A$ est une formule valide, alors $F'_P \,\&\, F_I$ est insatisfaisable. Par ailleurs $F_P \,\&\, F_I$ est satisfaisable, puisque F_P est satisfaisable (2.2) et les formules apparaissant dans F_I sont soit consistantes avec F_P, soit des formules où les variables propositionnelles at (l) n'apparaissent pas. Soit U un modèle qui satisfait $F_P \,\&\, F_I$ et qui assigne la valeur vraie à toutes les

formules du types $\mathcal{M}(\text{at}(l) \& p)$ (*voir* 2.2). Alors U doit falsifier F'_P, puisque $F'_P \& F_I$ est insatisfaisable, et U falsifie, en particulier, une formule $\mathcal{M}(\text{at}(l) \& C \& p) \rightarrow$, de F'_P, obtenue à partir d'une formule

$$\mathcal{M}(\text{at}(l) \& C \& p) \rightarrow \mathcal{M}(\text{at}(l_a) \& p),$$

de F_P, en effaçant $\mathcal{M}(\text{at}(l_a) \& p)$. $\mathcal{M}(\text{at}(l) \& C \& p)$ doit donc être vraie, et le contrôle doit être transmis au sommet de sortie; en conséquence le programme termine.

Ceci nous permet de ramener le test de la correction totale à un problème de démonstration en logique modale. Il faut remarquer que toutes les formules que nous avons considérées ne possèdent pas d'opérateurs modaux dans le champ d'autres opérateurs modaux (ce qui revient à dire que les formules sont de degré 1 [P]), et que les quantifieurs n'apparaissent pas dans le champ des opérateurs modaux. Dans ce qui suit nous donnons une méthode de déduction automatique pour ce type de formules et par conséquent nous ramenons le test de la correction totale d'un programme donné à un problème de démonstration automatique en logique modale.

En suivant une démarche parallèle à celle qui consiste à considérer les logiques modales comme des sous-ensembles de formules d'une logique du temps grammatical [Mc] nous pouvons considérer les formules de degré 1 comme l'ensemble des formules de $S5$, puisque toute formule dans $S5$ est réductible à une formule de degré 1 [HC]. Dans ce sens et non dans celui de la sémantique, le système $S5$, considéré par Burstall pour raisonner sur les programmes, ne contredit pas notre idée intuitive sur le déroulement séquentiel des programmes [PR].

III. DEUX MÉTHODES DE DÉDUCTION

3.1. Dans ce paragraphe nous donnons deux méthodes de déduction relatives aux formules modales de degré 1 pour lesquelles les variables dans le champ de leurs opérateurs modaux sont libres.

Ces formules, comme on peut le constater (*voir* paragraphe II et [MP]) nous permettent d'exprimer les contraintes relatives aux données ainsi que les étapes de l'exécution des programmes dans le but de tester leur correction totale.

Nous considérons les formules en forme normale de Skolem :

$$Q_1 \ldots Q_n (A_1 \& \ldots \& A_m),$$

où $m > 1$, $n \geqq 0$, les $Q_i\, 1 \leqq i \leqq n$ sont des quantificateurs universels et chaque A_i (clauses dans la terminologie classique [R]) est une disjonction de formules modales. L'élimination des quantificateurs existentiels est réalisée de façon usuelle comme en [RS].

Nous pouvons considérer plusieurs méthodes de déduction qui dépendent de la forme des formules modales A_i. Nous définissons les deux suivantes :

(*a*) Les $A_i\, 1 \leqq i \leqq n$ sont de la forme :

$$A_i = \mathscr{L} D_1 \vee \ldots \vee \mathscr{L} D_k \vee \mathscr{M} F \vee p_1 \vee \ldots \vee p_l,$$

où chaque $D_i\, 1 \leqq i \leqq k$ est une disjonction de littéraux (formules atomiques, variables propositionnelles, négation de formule atomique où négation de variable propositionnelle). F est une conjonction de disjonction de littéraux et les $p_i\, 1 \leqq i \leqq l$ sont des littéraux (*voir* [C]).

Pour ce type de clauses (que nous appellerons *a*-clauses) nous définissons la méthode de déduction (variante de celle qui nous a été communiquée par Minc [MI], basée sur les règles de résolution et de factorisation suivantes : soient $B \vee \mathscr{L}(D \vee p)$ et $C \vee \Delta((D' \vee p') \& F)$ deux *a*-clauses sans variables communes, p et p' deux littéraux et Δ qui peut être un des opérateurs modaux $\mathscr{L}$ ou $\mathscr{M}$, ou le symbole vide. Les règles de résolution :

$$\frac{B \vee \mathscr{L}(D \vee p); C \vee \Delta((D' \vee \sim p') \& F)}{B\sigma \vee C\sigma \vee \Delta((D \vee D') \& F)\sigma},$$

$$\frac{B \vee p,\ C \vee \sim p'}{B\sigma \vee C\sigma},$$

$$\frac{B \vee \mathscr{M}((D \vee p) \& (D' \vee \sim p') \& F)}{B\sigma \vee \mathscr{M}((D \vee D') \& F)\sigma},$$

sont appliquées s'il existe un unifieur le plus simple σ [R] tel que $p\sigma = p'\sigma$. Si $(D \vee D')$ est une disjonction vide alors il faut effacer $\Delta((D \vee D') \& F)$.

La règle de factorisation :

$$\frac{B \vee C \vee C'}{B\sigma \vee C\sigma}$$

est appliquée s'il existe un unifieur le plus simple σ tel que $C\sigma = C'\sigma$.

(*b*) Les $A_i\, 1 \leqq i \leqq m$ sont des disjonctions de formules indécomposables. Par formule indécomposable nous entendons une formule modale de l'un des types suivants :

- un littéral;
- une formule atomique, ou la conjonction de formules atomiques

(possédant éventuellement une variable propositionnelle) précédée de $\mathscr{M}$ ou de $\sim \mathscr{M}$;

— une disjonction de littéraux, ayant au moins un littéral positif, précédée de $\mathscr{L}$ ou de $\sim \mathscr{L}$.

Pour ce type de clauses (que nous appellerons *b*-clauses) nous définissons la méthode de déduction basée sur les règles suivantes :

Soient $B_1 \vee C, \ldots B_n \vee C_n$ $n \geqq 1$, n *b*-clauses sans variables communes. La règle de résolution :

$$\frac{B_1 \vee C_1; \ldots; B_n \vee C_n}{C_1 \vee \ldots \vee C_n},$$

est appliquée s'il existe un unifieur le plus simple σ (extension triviale de l'unification classique) tel que $\{B_1\ \sigma\ \&\ldots\&\ B_n\ \sigma\}$ est inconstante ([3]) et aucun sous-ensemble des B_i $1 \leqq i \leqq n$ ne vérifie cette propriété.

La règle de factorisation est identique à celle définie auparavant.

A ces deux règles il faut ajouter celle qui consiste à recopier des *b*-clauses, copies ne possédant pas de variables communes avec les autres *b*-clauses.

3.2. Soit S un ensemble fini de *a*-clauses (*b*-clauses); une *a-déduction* (*b-déduction*) à partir de S est une séquence de *a*-clauses (*b*-clauses) obtenues en appliquant les règles définies dans (*a*) et (*b*). Une *a-réfutation* (*b-réfutation*) à partir de S est une *a*-déduction (*b*-déduction) de la clause vide à partir de S.

3.2.1. *Soit S un ensemble de a-clauses (b-clauses). S est insatisfaisable si et seulement si il existe une a-réfutation (b-réfutation) à partir de S.*

La démonstration est obtenue de façon similaire à celle proposée en [F] ou [O].

Exemple : Soit l'ensemble insatisfaisable de *a*-clauses :

(1) $\mathscr{L}(\sim p(x) \vee q(y)) \vee \mathscr{M} r(x)$;

(2) $\mathscr{L} \sim r(c)$;

(3) $\mathscr{M} q(b)$;

(4) $\mathscr{L} p(a)$,

où les *b*-clauses équivalentes :

(1)′ $\sim \mathscr{M}(p(x) \& q(y)) \vee \mathscr{M} r(x)$;

(2)′ $\sim \mathscr{M} r(a)$;

(3)′ $\mathscr{M} q(b)$;

(4)′ $\mathscr{L} p(a)$.

([3]) *Voir* Carnap [C] pour une méthode très simple de décision pour *S*5.

Par le premier système de règles, nous obtenons successivement :

(5) $\mathscr{L}(\sim p(x) \vee q(y))$, à partir de (1) et (2);

(6) $\mathscr{M} \sim p(x)$, à partir de (5) et (3);

(7) la clause vide, à partir de (6) et (4).

Par le deuxième système de règles nous obtenons :

(5)′ $\sim \mathscr{M}(p(a) \& q(y))$, à partir de (1)′ et (2)′;

(6)′ la clause vide, à partir de (5)′, (3)′ et (4)′.

La règle de résolution utilisée pour obtenir (6)′ est représentée ainsi :

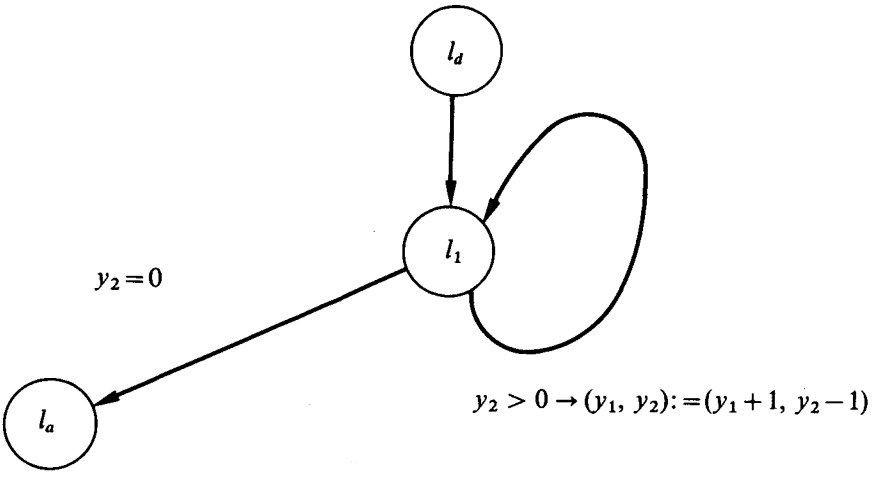
$$\frac{\sim \mathscr{M}(p(a) \& q(y)); \mathscr{M} q(b); \mathscr{L} p(a)}{\text{clause vide}}.$$

3.3. Dans le but d'obtenir des démonstrations proches de celles réalisées dans les logiques des programmes [MP] nous utiliserons, dans ce qui suit, la deuxième méthode (*b*) de déduction, puisque la première (*a*) associe, d'une façon générale, plusieurs pas de démonstration à chaque fois qu'on applique la règle de résolution de la deuxième méthode. Par la suite, chaque fois que nous parlerons de clause nous nous réfèrerons aux *b*-clauses.

Il faut remarquer que puisque les symboles de prédicats ou les variables propositionnelles qui apparaissent dans les formules modales associées aux programmes (2.1 et 2.3) sont tous différents, il ne sera pas nécessaire d'utiliser la règle nous permettant de recopier des *b*-clauses.

IV. CORRECTION TOTALE. UN EXEMPLE

Considérons le programme représenté par le diagramme suivant :



tel que pour x_1 et x_2, le programme calcule $x_1 + x_2$.

Les clauses correspondant au programme sont (F_P) :

α: $\mathscr{M}(\text{at}(l_d) \& p(x_1, x_2, x_1, x_2)) \rightarrow \mathscr{M}(\text{at}(l_1) \& p(x_1, x_2, y_1, y_2))$,

β: $\mathscr{M}(\text{at}(l_1) \& y_2 > 0 \& p(x_1, x_2, y_1+1, y_2-1)$
$\rightarrow \mathscr{M}(\text{at}(l_1) \& p(x_1, x_2, y_1, y_2)$,

γ: $\mathscr{M}(\text{at}(l_1) \& y_2 = 0 \& p(x_1, x_2, y_1, y_2)) \rightarrow \mathscr{M}(\text{at}(l_a) \& p(x_1, x_2, y_1, 0))$,

où $p(x_1, x_2, y_1, y_2)$ est l'assertion $x_1 + x_2 = y_1 + y_2$.

L'ensemble des clauses correspondant au principe d'induction introduit auparavant est :

I1: $\mathscr{M}(\text{at}(l_1) \& p(x_1, x_2, y_1, 0)) \vee \mathscr{M}(\text{at}(l_1) \& y_2 > 0 \&$,
$0 \leqq y_2 \leqq n+1 \& p(x_1, x_2, y_1+1, y_2-1)) \vee \sim \mathscr{M}(\text{at}(l_1) \&$,
$0 \leqq y_2 \leqq k \& p(x_1, x_2, y_1, y_2)) \vee \mathscr{M}(\text{at}(l_a) \& p(x_1, x_2, y_1, 0))$,

I2: $\mathscr{M}(\text{at}(l_1) \& p(x_1, x_2, y_1, 0)) \vee \sim \mathscr{M}(\text{at}(l_1) \&$,
$0 \leqq y_2 - 1 \leqq n \& p(x_1, x_2, y_1, y_2)) \vee \mathscr{M}(\text{at}(l_1) \&$,
$0 \leqq y_2 \leqq k \& p(x_1, x_2, y_1, y_2)) \vee \mathscr{M}(\text{at}(l_a) \& p(x_1, x_2, y_1, 0))$.

Nous ajoutons aussi les clauses suivantes :

1: $\mathscr{M}(x \neq 0) \vee \mathscr{L}(x = 0)$,

2: $\sim \mathscr{M}(0 \neq 0)$,

3: $\mathscr{L}(0 \leqq x-1 \leqq n) \vee \sim \mathscr{M}(x > 0 \& 0 \leqq x \leqq n+1)$,

4: $\mathscr{L}(0 \leqq x \leqq x)$,

qui sont indépendantes de l'état du programme, puisque la variable propositionnelle at (l) n'apparaît pas dans ces clauses. Donc elles expriment des propriétés formelles; (comme auparavant, nous notons par F_I, la conjonction de clauses correspondant à l'induction et aux propriétés formelles).

Pour établir la correction totale, il faut prouver que la formule composée de la conjonction des clauses :

$$F_P \& F_I \& \sim (\mathscr{M}(\text{at}(l_d) \& p(x_1, x_2, x_1, x_2)) \rightarrow \mathscr{M}(\text{at}(l_a) \& p(x_1, x_2, y_1, 0)))$$

est insatisfaisable.

Par résolution, nous obtenons successivement, à partir de α et

$\mathscr{M}(\text{at}(l_d) \& p(x_1, x_2, x_1, x_2))$:

5 : $\mathscr{M}(\text{at}(l_1) \& p(x_1, x_2, x_1, x_2))$,

à partir de I1, γ et 1 :

6 : $\mathscr{M}(0 \neq 0) \vee \mathscr{M}(\text{at}(l_1) \& y_2 > 0 \& 0 \leqq y_2 \leqq n+1 \&$,
$p(x_1, x_2, y_1 - 1, y_2 + 1)) \vee \mathscr{M}(\text{at}(l_1) \& 0 \leqq y_2 \leqq k \&$,
$p(x_1, x_2, y_1, y_2)) \vee \sim \mathscr{M}(\text{at}(l_a) \& p(x_1, x_2, y_1, 0))$,

donc, la règle de résolution sera représentée schématiquement (moyennant l'unification) ainsi :

$$\frac{C \vee \mathscr{M} A,\ D \vee \sim \mathscr{M}(A \& B),\ E \vee \mathscr{L} B}{C \vee D \vee E};$$

— à partir de 6 et 2 :

7 : $\mathscr{M}(\text{at}(l_1) \& y_2 > 0 \& 0 \leqq y_2 \leqq n+1 \&$,
$p(x_1, x_2, y_1 - 1, y_2 + 1)) \vee \mathscr{M}(\text{at}(l_1) \& 0 \leqq y_2 \leqq k \&$,
$p(x_1, x_2, y_1, y_2)) \vee \sim \mathscr{M}(\text{at}(l_a) \& p(x_1, x_2, y_1, 0))$;

— à partir de 7 et β :

8 : $\mathscr{M}(\text{at}(l_1) \& p(x_1, x_2, y_1, y_2) \vee \mathscr{M}(\text{at}(l_1) \& 0 \leqq y_2 \leqq k \&$,
$p(x_1, x_2, y_1, y_2)) \vee \mathscr{M}(\text{at}(l_a) \& p(x_1, x_2, y_1, 0))$;

— à partir de 8, 12 et 3, puis factorisation :

9 : $\mathscr{M}(\text{at}(l_1) \& p(x_1, x_2, y_1, 0)) \vee \sim \mathscr{M}(y_2 > 0 \& 0 \leqq y_2 \leqq n+1) v$,
$\mathscr{M}(\text{at}(l_1) \& 0 \leqq y_2 \leqq k \& p(x_1, x_2, y_1, y_2)) \vee \mathscr{M}(\text{at}(l_a) \&$,
$p(x_1, x_2, y_1, 0))$;

— à partir de 9, γ, 1 puis factorisation :

10 : $\mathscr{M}(0 \neq 0) \vee \sim \mathscr{M}(y_2 > 0 \ \& \ 0 \leqq y_2 \leqq n+1) \vee \mathscr{M}(\text{at}(l_1) \ \& \ 0 \leqq y_2 \leqq k$,
$\& \ p(x_1, x_2, y_1, y_2)) \vee \mathscr{M}(\text{at}(l_a) \ \& \ p(x_1, x_2, y_1, 0))$;

– à partir de 10 et 6 puis factorisation :

11 : $\mathcal{M}(0 \neq 0) \vee \sim \mathcal{M}(\text{at}(l_1) \& 0 \leqq y_2 \leqq k \& p(x_1, x_2, y_1, y_2)),$
$\vee \mathcal{M}(\text{at}(l_a) \& p(x_1, x_2, y_1, 0)),$

donc, la règle de résolution sera représentée schématiquement ainsi :

$$\frac{\mathcal{M}(A \& B) \vee C, \sim \mathcal{M} A \vee D}{C \vee D};$$

– à partir de 11 et 2 :

12 : $\sim \mathcal{M}(\text{at}(l_1) \& 0 \leqq y_2 \leqq k \& p(x_1, x_2, y_1, y_2)) \vee,$
$\sim \mathcal{M}(\text{at}(l_a) \& p(x_1, x_2, y_1, 0));$

– à partir de 12, 5 et 4 :

13 : $\mathcal{M}(\text{at}(l_a) \& p(x_1, x_2, y_1, 0));$

– à partir de 13 et $\sim \mathcal{M}(\text{at}(l_a) \& p(x_1, x_2, y_1, 0))$ nous obtenons la clause vide. Donc la correction totale est assurée.

Le déroulement de la démonstration de cet exemple est très proche de celui qui découle de la méthode des *assertions intermittentes* [MW] et, en particulier, de sa formulation en Lucid [AW1, AW2].

L'application des différentes règles de déduction [MP] peut être simulée par l'application de la règle de résolution. Ainsi, on montre facilement que l'utilisation de la règle d'induction, chez Manna et Pnueli, est très proche de celle donnée en Lucid, et, par conséquent, simulable dans notre formalisme. D'autres règles, comme la *concaténation* et la *conséquence* (qui sont des règles dérivées dans *S*4), sont simulables directement par la règle de résolution, en l'appliquant une fois pour la concaténation, et deux fois de suite pour la conséquence. La règle dite de *Frame* [MP] est utilisée implicitement dans la règle de résolution de façon plus intuitive. L'application de cette règle met en jeu plus de deux clauses, ce qui revient à dire que les clauses exprimant des propriétés dites matérielles, ou indépendantes de l'état où le programme se trouve, seront utilisées.

Par conséquent, nous pouvons considérer l'ensemble de formules de notre système modal, muni du principe de résolution, comme un langage de programmation, nous permettant ainsi de raisonner sur des programmes à partir de leur manipulation directe, plutôt qu'indépendamment, *via* une logique séparée. Dans ce sens, cette démarche est très proche de celle présentée par Lucid (pour la correction totale) [AW2] ou de celle présentée par

van Emden [VE] dans laquelle les formules qui représentent les conditions de vérification (dans le sens de Floyd) et qui nous permettent d'établir la correction partielle, sont interprétées directement comme des programmes.

BIBLIOGRAPHIE

[AW1] E. ASHCROFT et W. WADGE, *Lucid, a Non Procedural Language with Iteration*, Com. A.C.M., vol. 20, n° 7, July 1977, p. 519-526.

[AW2] E. ASHCROFT et W. WADGE, *Intermittent Assertion Proofs in Lucid*, I.F.I.P. 77, p. 723-726.

[B] BURSTALL, *Program Proving as Hand Simulation with a Little Induction*, I.F.I.P. 74, p. 308-312.

[C] R. CARNAP, *Modalities and Quantification*, J.S.L., vol. 11, n° 2, 1946, p. 33-64.

[E] E. ENGELER, *Algoritmic Properties of Structures*, Math. Sys. The., vol. 1, n° 3, p. 183-185.

[EP] P. ENJALBERT, *Systèmes de déduction pour les arbres et les schémas de programmes*, R.A.I.R.O. Inform. Théor., vol. 14, n° 3, 1980, p. 247-278.

[F] FARIÑAS DEL CERRO, *Un principe de résolution en logique modale* (à paraître).

[FL] FLOYD, *Assigning Meaning to Programs*, Proc. Amer. Math. Soc. Symp. in App. Math., vol. 19, 1967, p. 19-31.

[H] D. HAREL, *First-Order Dynamic Logic*, Lectures Notes in Computer Science, Springer Verlag, n° 68.

[HKP] HAREL, KOZEN et PARIKH, *Process Logic, Expressiveness, Decidebility, Completeness*, F.O.C.S. 80, p. 129-142.

[HO] HOARE, *An Axiomatic Basic of Computer Programming*, Com. A.C.M., vol. 12, n° 10, 1969.

[HC] HUGHES et CRESSWELL, *An Introduction to Modal Logic*, Mathuem et Co., London, 1978.

[K] KRÖGER, *LAR: A Logic for Algorithmic Reasoning*, Acta Informatica, 1977, p. 243-266.

[M1] Z. MANNA, *Properties of Programs and First Order Predicate Calculus*, J.A.C.M., vol. 16, n° 2, 1969, p. 244-255.

[M2] Z. MANNA, *Logics of Programs*, Proc. I.F.I.P. 80, North-Holland, p. 41-52.

[MP] Z. MANNA et A. PNUELI, *The Modal Logic of Programs*, Memo AIM-330 Stanford A.I. Laboratory, Sept. 1979.

[NW] Z. MANNA et R. WALDINGER, *Is "sometime" sometime better than "always"?: Intermittent Assertions in Proving Program Correcteness*, Com. A.C.M., vol. 21, n° 2, 1978, p. 159-172.

[Mc] MCARTHUR, *Tense Logic*, Reidel Publ., 1976.

[McT] MCKINSEY et TARSKI, *Some Theorems about the Sentential Calculi of Lewis and Heyting*, J.S.L., vol. 13, 1948, p. 1-15.

[MI] MINC, Communication personnelle.

[O] ORLOWSKA, *Resolution Systems and their Applications*, I, II Fundamenta Informaticae, p. 235-267, p. 333-362.

[P] W. T. PARRY, *Modalities in the Survey System of Strict Implication*, J.S.L., vol. 4, 1939, p. 131-154.

[PR] V. R. PRATT, *Semantical Considerations on Floyd-Hoare Logic*, Proc. 17th Ann. I.E.E.E. Symp. on Foundations of Comp. Sc., 1976, p. 109-121.

[RS] RASIOWA et SIKORSKI, *The Mathematics of Metamathematics*, Warszowa, 1963.

[R] ROBINSON, *A Machine Oriented Logic Based on the Resolution Principle*, J.A.C.M., vol. 12, 1965, p. 23-41.

[S] SALWICKI, *Formatized Algorithme Language*, Bull. Ac. Pol. Sc., vol. 18, n° 5, 1970, p. 227-232.

[VE] M. VAN EMDEN, *Verification Conditions as Programs, Automate Languages and Programming*, MICHEALSON and MILNER, éd., Edinburg Univ. Press, 1976, p. 99-119.