ЛИТЕРАТУРА

[1] Fejér L., *Über die Lage der Nullstellen von Polynomen die aus Minimumforderungen gewisser Art entspringen*. Math., Ann., **85**, 41—48 (1922).
[2] Гребенюк D. Г., Полиномы найлучшего приближения, коэффициенты которых связаны линейными зависимостями. Ташкент 1960.
[3] Marden M., *Location af the zeros of infrapolynomials*. Amer. Math. Monthly, **70**, *4*, 361—371 (1963).
[4] Maruşciac I., *Sur certains infrapolynomes conditionnés*. Mathematica (Cluj), *4* **(27)**, *1*, 33—52 (1957).
[5] Maruşciac I., *Complèment au travail „Sur certains infrapolynomes conditionnés"*. Mathematica (Cluj), **7 (30)**, *2*, 283—285 (1965).
[6] Марущак И., *Обобщённые инфраполиномы*. Mathematica (Cluj), **7** (30), *2*, 263—282 (1965).

Поступило 1. X. 1966.

MATHEMATICA VOL. 9 (32), 1, 1967, pp. 111—128

ANALYSE LOGIQUE DES ALGORITHMES (I)

par

E. MUNTEANU

à Cluj

1. Introduction

Les succès obtenus ces derniers temps dans l'automation de la programmation, par la création des langages de programmation du type ALGOL et des translateurs de ces langages, ont conduit à l'allègement essentiel du travail de programmation.

L'utilisation des programmes de programmation dispense le programmateur d'un travail pénible qui n'est pas directement relié à l'essence même de la résolution du problème, et dont l'apparition est due aux particularités de la machine.

Toutefois, ainsi que le montre l'expérience, l'écriture de l'algorithme en un langage de programmation comporte certaines difficultés caractéristiques, même alors que son exposition en langage mathématique habituel ne laisse rien à désirer sous le rapport de la clarté. Telles sont les difficultés caractéristiques à la mise de l'algorithme dans une forme commode pour la programmation et elles ont trait aux caractéristiques générales des machines électroniques à calculer. Pour écarter complètement ces difficultés, nous estimons nécessaire d'élaborer des langages de programmation aux possibilités nouvelles sous le rapport de la notion d'algorithme.

La description du problème dans un tel langage à la différence des langages existants jusqu'à ce jour, ne doit pas coïncider avec la description détaillée de la succession des opérations. En d'autres mots la description du problème dans un tel langage ne doit pas être nécessairement un algorithme, au sens habituel du mot. Le travail d'élaboration de l'algorithme proprement dit doit rester à la charge de la machine.

On sait qu'en certains cas, la description d'un algorithme simple peut être remplacée par la description du système de relations logiques existentes entre l'information initiale du problème et le résultat obtenu à la suite de la réalisation de l'algorithme. Il est donc naturel qu'un tel langage contienne la possibilité de décrire le problème au moyen d'un système de conditions logiques. Cependant, les possibilités d'une telle description doivent être bornées vu qu'au cas contraire nous serions amenés à prétendre à un programme de traduction d'un tel langage d'élaborer l'algorithme de résolution pour n'importe quel problème mathématique. En réalité, il s'agit de laisser à la charge de à la machine seulement les éléments les plus simples de construction d'un algorithme, notamment ceux qui présentent un caractère exclusivement technique et n'ont pas trait à l'essence du problème.

C'est pourquoi il est naturel de poser le problème suivant:

Trouver une méthode qui permette dans les cas les plus simples de construire automatiquement les algorithmes, en partant du système de relations logiques existentes entre l'information initiale et le résultat final.

Ce problème conduit à un nouveau problème d'analyse des algorithmes. La caractéristique générale des méthodes d'analyse connues jusqu'à ce jour [3], [6], [4] consiste dans la détermination des contradictions d'ordre formel, qui existent dans la construction de l'algorithme. Qu'il n'y ait pas de telles contradictions il reste cependant le problème suivant, qui n'a pas été résolu jusqu'à présent: l'algorithme analysé correspond-il au problème pour lequel il a été construit?

Il est donc naturel que l'analyse des algorithmes soit traitée du point de vue de la correspondance entre celui-ci et le système de conditions logiques qui décrit l'énoncé du problème qui est résolu à l'aide de l'algorithme respectif. En d'autres mots, le problème de l'analyse des algorithmes doit être formulé dans le sens de l'élaboration de certaines méthodes de résolution du système de conditions logiques qui décrit l'énoncé du problème, en partant de la seule description formelle de l'algorithme et de la vérification du fait si ce système coïncide avec le système donné initialement par l'énoncé du problème.

Une résolution de problème d'analyse des algorithmes dans le sens décrit ci-dessus pourrait constituer un premier pas dans la voie de trouver les méthodes de construction automatique des algorithmes. De plus elle peut devenir un instrument très utile dans le contrôle des programmes pour les machines électroniques à calcul.

En effet, en disposant d'une méthode pour établir le système des conditions logiques qui décrit l'énoncé du problème en partant du seul programme de résolution de ce problème et en comparant ensuite ce système avec celui initialement donné, on pourra établir si le programme relatif au problème est correct.

Les difficultés considérables, qui surgissent au contrôle des programmes, tant en programmation manuelle qu'en programmation automatique,

ainsi que l'absence de programmes universels de contrôle justifient complètement l'importance de ces idées, dans la résolution du problème de l'analyse des algorithmes.

2. Énoncé du probleme de l'analyse des algoritmes

Nous considérons le problème $\mathcal{P}$ et désignerons par X^0 ses données initiales et par X^1 ses résultats finaux.

Nous supposerons que pour la résolution du problème $\mathcal{P}$ on a construit l'algorithme $\mathfrak{A}$, décrit dans un langage formel. En ce qui suit nous considérerons à titre de langage formel pour la description des algorithmes le langage des schémas graphes.

Définition 1. *La formule* $F(X^0, X^1)$ *sera dite* formule attachée *à l'algorithme* $\mathfrak{A}$ *si elle est vraie alors que l'algorithme* $\mathfrak{A}$ *appliqué aux données initiales* X^0 *les transforme dans la résultat final* X^1.

Définition 2. *La formule.* $F(X^0, X^1)$ *sera dite* formule complètement attachée *à l'algorithme* $\mathfrak{A}$ *si elle est vraie si et seulement si en appliquant l'algorithme aux données initiales* X^0 *celui-ci les transforme dans le résultat final* X^1.

La résolution du problème de l'analyse des algorithmes nécessite, à côte de l'élaboration d'une méthode de construction de la formule attachée, de la formule complètement attachée à un algorithme, aussi la formulation d'un langage dans lequel ces formules doivent être écrites, ainsi que le système de relations logiques initiales, qui représentent l'enoncé du problème $\mathcal{P}$. Nous allons décrire par la suite toutes ces formules dans le langage logiqo-mathématique.

Nous supposerons que l'énoncé du problème $\mathcal{P}$ est décrit par la formule logique $\mathcal{F}(X^0, X^1)$ qui est un prédicat défini sur l'ensemble des données initiales et des résultats finaux. Par exemple pour trouver le maximum des nombres $x_1, x_2, \ldots, x_n$ le prédicat correspondent sera

$$(1) \qquad [(x=x_1)\vee(x=x_2)\vee\ldots\vee(x=x_n)]\ \&\ (x\geqslant x_1)\&(x\geqslant x_2)\ \&\ldots\&\ (x\geqslant x_n),$$

où x est le maximum recherché. La formule (1) n'est autre que la définition du maximum des nombres $x_1, x_2, \ldots, x_n$ elle ne représente pas l'algorithme qui permet de trouver le maximum, attendu qu'elle ne nous fournit aucune information relative à la succession d'opérations, qui doivent être effectuées pour trouver le maximum des nombres $x_1, x_2, \ldots, x_n$.

Si nous sommes parvenus à construire une formule $F(X^0, X^1)$ qui satisfait aux définitions 1 ou 2 alors nous dirons que l'algorithme $\mathfrak{A}$ est *correct* relativement au problème $\mathcal{P}$ si la formule

$$(2) \qquad \forall X^0 \forall X^1 [F(X^0, X^1) \supset \mathcal{F}(X^0, X^1)],$$

est vraie.

Énoncé de la sorte, le problème de l'analyse de l'algorithme $\mathfrak{A}$ se réduit à la construction d'une formule, qui satisfait aux définitions 1 ou 2 et à la vérification de la véracité de la formule (2).

Dans ce travail nous présenterons une méthode pour la construction de la formule complètement attachée à un algorithme schéma-graphe sans cycles.

3. Schémas—graphes à mémoire

Dans ce qui suit nous attacherons à un schéma-graphe Γ défini par A. L. KALUJNIN en [2] les notions de mémoire et d'état de mémoire.

Un ensemble fini d'objets $M = \{x_1, x_2, \ldots, x_n\}$ sera appelé mémoire d'un schéma-graphe. Les éléments de l'ensemble seront appelés *cellules* de la mémoire. Nous considérons un ensemble

$$S = \{x_1^0, x_2^0, \ldots, x_n^0, \ldots, x_1^j, x_2^j, \ldots, x_n^j, \ldots\},$$

dont les éléments seront appelés *états*. L'état x_i^j sera appelé contenu de la cellule x_i.

L'image $x_i^j = U(x_i)$ de l'ensemble M dans l'ensemble S sera appelée *état de la mémoire* du schéma-graphe Γ.

La vecteur $\xi = (x_1^j, x_2^j, \ldots, x_n^j)$ sera appelé vecteur complet de l'état de la mémoire. Nous désignerons par $\mathcal{S} = \{\xi\}$ l'ensemble des vecteurs complets de tous les états de la mémoire.

Soit $X = (x_1, x_2, \ldots, x_n)$ un vecteur complet de la mémoire, et f un symbole fonctionnel quelconque, enfin p un symbole arbitraire de relation.

Définition 3. *Une expression de la forme* $y := f(X)$, *où* $y \in M$ *sera appelée* opérateur d'attribution.

Définition 4. *Une expression de la forme* $p(X)$ *sera appelée opérateur logique.*

Dans les définitions des opérateurs d'attribution et de l'opérateur logique les fonctions f et p sont définies sur tout l'ensemble de la mémoire schéma-graphe, mais à l'ordinaire les expressions $f(X)$ et $p(X)$ ne comprennent pas toutes les cellules de la mémoire. Les cellules qui ne figurent pas dans ces expressions ne sont pas essentielles pour les opérateurs respectifs.

Définition 5. *Une expression de la forme* $\eta = f(\xi)$ *où* $\eta \in S$ *et* $\xi \in \mathcal{S}$ *sera dite* formule élémentaire attachée à l'opérateur *d'attribution* $y := f(X)$.

Définition 6. *Une expression de la forme* $p(\xi)$, *où* $\xi \in \mathcal{S}$ *sera appelée formule élémentaire attachée à l'opérateur logique.*

Au cas où le fait qu'une formule élémentaire est attachée à un opérateur d'attribution ou à un opérateur logique n'est pas essentiel, nous la désignerons simplement pas les termes „formule élémentaire".

Pour les formules élémentaires vaut l'observation faite ci-dessus à propos des opérateurs.

Nous entendrons par interprétation d'un schéma-graphe à mémoire défini par l'ensemble $T = \{T_1, T_2, \ldots, T_k\}$ des opérateurs de transformation et par l'ensemble $R = \{R_1, R_2, \ldots, R_m\}$ des opérateurs de ramification ce qui suit.

Nous ferons correspondre à chaque élément T_i de T un opérateur d'attribution f_i (la correspondence n'étant pas nécessairement biunivoque) et à chaque élément R_j de R un opérateur logique p_j (la correspondence n'étant pas nécessairement biunivoque).

Dans le dernier cas on suppose la décomposition de l'ensemble $\mathcal{S}$ en deux ensembles disjoints $\mathcal{S}_{p_j}^+$ dont les éléments satisfont à la propriété p_j et $\mathcal{S}_{p_j}^-$ dont les éléments ne satisfont pas à la propriété p_j. Nous désignerons un telle interprétation par $\{M, \mathcal{S}; T \to f; R \to p\}$.

Si un schéma-graphe Γ à mémoire est donné, ainsi qu'une interprétation $\{M, \mathcal{S}; T \to f; R \to p\}$ alors nous dirons qu'est donné un algorithme schéma-graphe $\mathfrak{A}$ dont l'action est la suivante.

1. Nous supposerons que le vecteur des données initiales $X^0 \in \mathcal{S}$ en parcourant les noeuds du schéma-graphe Γ s'est transformé dans le vecteur $\xi = (x_1^S, \ldots, x_n^S)$ qui est arrivé au noeud T_i.

Au noeud T_i correspond par l'interprétation donnée l'opérateur d'attribution $x_h := f_i(X)$. Le procédé de calcul est poursuivi selon l'unique flèche qui part de ce noeud, par le vecteur, $\xi' = (x_1^{S'}, x_2^{S'}, \ldots, x_h^{S'}, x_{h+1}^{S'}, \ldots, x_n^{S'})$ où $x_i^{S'} = x_i^S$ pour $i = 1, 2, \ldots, h, h+1, \ldots, n$ tandis que

$$x_h^{S'} = f_i(\xi), \tag{3}$$

ce qui est la formule élémentaire attachée à l'opérateur d'attribution $x_h := f_i(X)$.

2. Nous supposerons que le vecteur ξ, transformé du vecteur X^0 est arrivé dans un noeud R_j. Au noeud R_j correspond par le truchement de l'interprétation donnée, l'opérateur logique p_j. En ce cas le procès de calcul se poursuit par le même vecteur ξ, selon la flèche marquée par le signe $+$, si la formule élémentaire $p_j(\xi)$ est vraie ou bien selon la flèche marquée par le signe $-$ au cas contraire.

3. Si à une certaine étape du procédé de calcul l'élément X^1 de l'ensemble $\mathcal{S}$ arrive à l'issue du schéma-graphe, alors nous dirons que X^1 est le résultat de l'application de l'algorithme $\mathfrak{A}$, défini par le schéma-graphe

Γ par le truchement de l'interprétation, $\{M, \mathcal{S}; T \to f; R \to p\}$ ce que nous désignerons par la notation suivante

$$\mathfrak{A}(X^0) = X^1.$$

Si dans le schéma-graphe de l'algorithme $\mathfrak{A}$ est essentiel non le type de l'opérateur, mais son nombre, nous désignerons cet opérateur par le symbole O_i. Si l'opérateur O_i fait partir une flèche vers l'opérateur O_j nous appellerons l'opérateur O_j *successeur* de l'opérateur O_i, et ce dernier *prédécesseur* de l'opérateur O_j.

3.1. Distribution de la mémoire d'un schéma-graphe

Considérons l'algorithme schéma-graphe $\mathfrak{A}$ défini par le schéma-graphe Γ par l'interprétation $\{M, \mathcal{S}, T \to f; R \to p\}$.

Nous désignerons par $M_v = \{x_{i_1}, x_{i_2}, \ldots, x_{i_r}\}$ le sous-ensemble de l'ensemble M qui est caractérisé par le fait que durant la réalisation de l'algorithme $\mathfrak{A}$ le contenu des cellules x_{i_k} change. L'ensemble M_v est composé des cellules de la mémoire qui figurent au second membre des opérateurs d'attribution. L'ensemble des autres cellules de la mémoire sera désigné par $M_c = \{c_1, c_2, \ldots, c_s\}$ $(s + r = n$, où n est le nombre des cellules de la mémoire). Nous désignerons par c_i^0 l'état initial de la cellule c_i et par $M_c^0 = \{c_1^0, c_2^0, \ldots, c_s^0\}$ l'ensemble des états initiaux des cellules de l'ensemble M_c. Il est évident que M_c, est inclus dans S.

Nous désignerons par $\{x_{j_1}, x_{j_2}, \ldots, x_{j_m}\}$ les cellules de l'ensemble M_v dont les états initiaux sont les données initiales pour l'algorithme $\mathfrak{A}$. Nous supposerons pour simplifier que ce sont les premières m cellules de la mémoire et nous désignerons à partir de ce moment leur ensemble par $N = \{x_1, x_2, \ldots, x_m\}$.

Leur ensemble peut être vide. Nous désignerons par N^0 l'ensemble des états initiaux des cellules de l'ensemble N. Les éléments de l'ensemble $I^0 = N^0 \cup M_c^0$ seront alors les données initiales de l'algorithme $\mathfrak{A}$.

Si l'ensemble N ne coïncide pas avec l'ensemble M_v c'est que pendant le temps d'action de l'algorithme $\mathfrak{A}$ certaines cellules de l'ensemble sont utilisées pour conserver (garder) des résultats intermédiaires ou des résultats finaux. Les cellules qui sont utilisées seulement pour la conservation des résultats intermédiaires seront désignées par le nom *cellules actives*.

Il est évident que l'algorithme $\mathfrak{A}$ doit être construit de telle manière que le résultat final obtenu ne dépende pas de l'état initial des cellules actives.

Après que le procès de calcul ait été terminé, certaines cellules de l'ensemble M_v (et ce peut être toutes) contiennent le résultat de l'application de l'algorithme $\mathfrak{A}$.

Nous désignerons l'ensemble de ces cellules par $R = \{x_{i_1}, x_{i_2}, \ldots, x_{i_p}\}$. Si nous désignons par x^1 l'état final de la cellule x alors l'ensemble $R^1 = \{x_{i_1}^1, x_{i_2}^1, \ldots, x_{i_p}^1\}$ est l'ensemble des résultats de l'application de l'algorithme $\mathfrak{A}$.

En vue de la description d'un état intermédiaire d'une cellule quelconque, nous introduirons la notion de *rang*.

Les symboles x_i^j qui figurent dans une formule élémentaire seront désignés par le terme de *variables*. Le nombre j sera donc le rang de la variable x_i^j. Le rang de chaque variable est un nombre entier, positif, qui se rapporte à la cellule dont l'état est la variable respective.

Il résulte de la notation introduite pour l'état final d'une cellule que toutes les variables du premier rang représentent des résultats finaux de l'algorithme respectif.

Si l'algorithme schéma-graphe $\mathfrak{A}$ ne contient pas de cycles et si nous analysons le schéma-graphe correspondant en partant de l'issue vers l'entrée, alors chaque fois que le contenu d'une cellule change le rang de la variable correspondante augmente (en général d'une unité). Si x_i'' est l'état initial de la cellule x_i nous conviendrons de désigner le nombre n par le terme de *rang maximal* de la variable x_i''. Si x_i'' est une variable à rang maximal, nous désignerons cette variable par x_i^0.

4. Construction de la formule complètement attachée à un algorithme schéma-graphe sans cycles

Nous entendrons par la suite par algorithme schéma-graphe $\mathfrak{A}$ un algorithme sans cycles [1].

Nous supposerons que le schéma-graphe de l'algorithme $\mathfrak{A}$ contient $n - 1$ opérateurs, l'entrée et la sortie. Nous allons numéroter de la manière suivante les opérateurs de l'algorithme $\mathfrak{A}$:

1.° Nous désignerons par O_0 l'entrée du schéma-graphe de l'algorithme $\mathfrak{A}$.

2.° Si le successeur de l'opérateur d'attribution O_i ou de l'entrée est l'opérateur O_j alors nous conviendrons que $j < i$. Si les successeurs de l'opérateur logique O_k sont les opérateurs O_{j_1} et O_{j_2} alors nous conviendrons que $j_1 > k$ et $j_2 > k$.

3.° Nous désignerons par O_n la sortie du schéma-graphe de l'algorithme $\mathfrak{A}$.

Les opérateurs de tout algorithme sans cycles peuvent être toujours numérotés de telle manière que les conditions 1°, 2°, et 3° soient satisfaites.

Pour démontrer cette affirmation, nous indiquerons le procédé suivant de réalisation d'une telle manière de numéroter :

a) Nous désignerons l'entrée du schéma-graphe par O_0

b) Nous supposerons que les opérateurs $O_0, O_1, \ldots, O_i$, ont été numérotés de telle maniére que les conditions suivantes, 1°, 2°, et 3° soient satisfaites et que dans aucun de ces opérateurs il n'entre de flèches en partance d'opérateurs non numérotés. Si l'algorithme ne contient pas d'opérateurs non numérotés, alors la numérotation est finie et satisfait aux conditions 1°, 2°, et 3°.

c) Si l'algorithme contient des opérateurs non numérotés, alors nous chercherons parmi ceux-ci l'un dans lequel il n'entre aucune flèche en partance d'opérateurs non numérotés.

Il résulte de l'hypothèse que l'algorithme $\mathfrak{A}$ ne contient pas de cycles qu'on peut toujours trouver un tel opérateur. Nous supposerons qu'il n'existe pas un tel opérateur. Nous désignerons par O un opérateur non numéroté.

L'opérateur O possède au moin un prédécesseur non numéroté, qui possède à son tour un prédécesseur non numéroté.

Vu que le nombre des opérateurs non numérotés est fini, il résulte que l'opérateur O appartient à un cycle.

Cette conclusion contredit l'hypothèse selon laquelle l'algorithme ne contient pas de cycles.

Nous désignerons par O_{i+1} l'opérateur dans lequel il n'entre aucune flèche en partance d'opérateurs non numérotés.

Après un nombre fini de pas, la numérotation de l'algorithme est finie et satisfait aux condition 1°, 2° et 3°.

4.1. Construction de la formule élémentaire attachée aux opérateurs d'attribution

Le formule élémentaire attachée à un opérateur d'attribution telle qu'elle a été définie (définition 5) n'est pas unique.

Nous montrerons qu'étant donnée une formule logique φ qui satisfait à une certaine condition et un opérateur d'attribution, $z := g(X)$, alors relativement à φ on peut toujours construire d'une manière, unique la formule élémentaire attachée à l'opérateur $z := g(X)$. Nous désignerons par $A_\varphi[z := g(X)]$ l'expression du second nombre de la formule élémentaire attachée à l'opérateur $z := g(X)$ relativement à φ. Par la notation $[\psi]^x_\omega$ nous entendrons le résultat de l'opération de remplacement de chaque entrée libre de la variable x en ψ, à l'expression ω.

Condition α. Si la variable x_i^s entre librement dans la formule ψ et, $s \geqslant 2$ alors elle est l'unique variable de rang au moins égal à deux relativement à la cellule x_i qui entre librement en ψ.

Soit donné l'opérateur $z := g(X)$ et la formule φ qui satisfait à la condition α. L'expression $A_\varphi[z := g(X)]$ se construit de la manière suivante:

C.1.1 Si z est une composante du vecteur X et s'il existe la variable z^s, où $s \geqslant 2$ qui entre librement dans φ alors nous définirons

$$A^1_\varphi[z := g(X)] :: = [g(X)]^z_{z^{s+1}}. \tag{4}$$

C.1.2 Si z est une composante du vecteur X et z^1 est sa seule variable relative à z qui entre librement dans φ où φ ne contient pas de variables relatives à z alors nous définirons.

$$A^1_\varphi[z := g(X)] :: = [g(X)]^z_{z^1}. \tag{5}$$

C.1.3 Si z n'est pas une composante du vecteur X nous définirons

$$A^1_\varphi[z := g(X)] :: = g(X). \tag{6}$$

C.2 Pour toutes les cellules c qui entrent dans l'expression $A^1_\varphi[z := g(X)]$ nous définirons

$$A^2_\varphi[z := g(X)] :: = [A^1_\varphi[z := g(X)]]^{C_i}_{C_i^0}. \tag{7}$$

C.3 Pour toutes les cellules x_i qui entrent dans l'expression $A^2_\varphi[Z := g(X)]$ et relativement auxquelles il n'existe aucune variable qui entre dans φ nous définissons.

$$A^3_\varphi[Z := g(X)] :: = [A^2_\varphi[Z := g(X)]]^{x_i}_{x_i^1}. \tag{8}$$

C.4 Pour toutes les cellules x_i qui entrent dans l'expression $A^3_\varphi[z := g(X)]$ et relativement auxquelles il existe la variable $x_i^{s_i}$, où $s_i \geqslant 2$, qui entre librement dans φ, nous définissons.

$$A^4_\varphi[z := g(X)] :: = [A^3_\varphi[z := g(X)]]^{x_i}_{x_i^{s_i}}. \tag{9}$$

C.5 Pour toutes les cellules x_i qui entrent dans l'expression $A^4_\varphi[z := \varphi(X)]$ et relativement auxquelles existent les variables x_i^1 qui entrent librement dans le second nombre d'une formule élémentaire attachée à un opérateur d'attribution ou dans une formule élémentaire attachée à un opérateur logique, contenue dans φ, nous définirons

$$A^5_\varphi[z := g(X)] :: = [A^4_\beta[z := g(X)]]^{x_i}_{x_i^1}. \tag{10}$$

C.6 Pour toutes les cellules x_i qui entrent dans l'expression $A_\varphi^5[z:=g(X)]$ et relativement auxquelles la formule φ contient les formules élémentaires de la forme $x_i^1 = f(\xi)$ où x_i^1 n'entrent pas dans l'expression $f(\xi)$ nous définissons

$$A_\varphi^6[z:=g(X)]:=[A_\varphi^5[z:=g(X)]]_{x_i^2}^{x_i}. \tag{11}$$

L'expression $A_\varphi^6[z:=g(X)]$ est le membre droit d'une formule élémentaire, relative à φ attachée à l'opérateur d'attribution $z:=g(X)$ c'est-à-dire $A_\varphi[z:=g(X)]$. Dans le processus de construction de l'expression $A_\varphi[z:=g(X)]$, les conditions décrites ci-dessus sont toujours vérifiées dans l'ordre C.1, C.2, C.3, C.4, C.5 et C.6.

Il résulte de ces conditions que si l'expression $A_\varphi[z:=g(X)]$ contient une variable relative à la cellule x alors elle en contient une seule relativement à cette cellule.

4.2. Construction de la formule élémentaire attachée à l'opérateur logique

Si l'on donne une formule φ et l'opérateur logique $p(X)$ alors de même qu'au cas de l'opérateur d'attribution on peut toujours construire d'une seule manière une formule élémentaire relativement à φ attachée à l'opérateur $p(X)$.

Nous désignerons cette formule par $L_\varphi[p(X)]$.

La construction de la formule $L_\varphi[p(X)]$ s'effectue d'une manière semblable à l'exception seulement de la condition C.4 qui est remplacée par la condition suivante.

C'.4. Pour toutes les cellules x_i, qui entrent dans l'expression $L_\varphi[p(X)]$ et relativement auxquelles il existe les variables $x_i^{s_1}, \ldots, x_i^{s_p}$ qui entrent librement dans φ et pour lesquelles $s_k \geqslant 2$ $(k = 1, 2, \ldots, r)$ nous définirons:

$$L_\varphi^4[p(X)]::=[L_\varphi^3[p(X)]]_{x_i^s}^{x_i}, \tag{12}$$

où $s = \max(s_1, s_2, \ldots, s_r)$.

Il est évident que pour la construction de la formule, $L_\varphi[p(X)]$ il n'y a aucune raison d'utiliser la condition C.1. Pour la formule $L_\varphi[p(X)]$ on a la propriété:

Si $L_\varphi[p(X)]$. contient une variable relative à la cellule x alors elle en contiendra une seulement relativement à cette cellule.

4.3. L'opération d'égalisation des rangs (ER)

L'opérateur (ER) ne s'applique qu'à deux formules φ_1 et φ_2 qui satisfont à la condition α. A la suite de l'application de cet opérateur nous obtiendrons les deux formules $E(\varphi_1, \varphi_2)$ et $E(\varphi_2, \varphi_1)$ de la manière suivante. Si les variables x^s et $x^{s'}$ entrent librement dans la formule φ_1 respectivement φ_2 et $s \geqslant 2$, $s < s'$ alors nous remplacerons chaque entrée de la variable $x^{s'}$ dans la formule φ_2 par la variable x^s. Si $s' = 1$ le remplacement se fera au seul cas où x^1 entre dans l'expression du second membre d'une formule élémentaire attachée à un opérateur d'attribution ou dans une formule élémentaire attachée à un opérateur logique contenu dans φ_2.

En effectuant toutes les substitutions possibles nous obtiendrons une formule que nous désignerons par la lettre $E(\varphi_2, \varphi_1)$ En permutant les formules φ_1 et φ_2 et en effectuant toutes les substitutions possibles dans les mêmes conditions nous obtiendrons le second résultat de l'opérateur c'est-à-dire une formule que nous désignerons par la notation $E(\varphi_1,\varphi_2)$.

Après l'application de l'opération ER toutes les variables qui entrent dans les formules $E(\varphi_1, \varphi_2)$ et $E(\varphi_2, \varphi_1)$ et qui satisfont aux conditions d'application de cette opération ont le même rang.

4.4. Construction de la formule complètement attachée à l'algorithme $\mathfrak{A}$

Est donné l'algorithme $\mathfrak{A}$ ainsi que les ensembles N, R et M_C. La construction de la formule complètement attachée à l'algorithme $\mathfrak{A}$ s'effectue par induction de la manière suivante:

1°. Nous attachons à l'opérateur O_n c'est-à-dire à la sortie la formule;

$$F_n::=(0=0),$$

qui satisfait évidemment à la condition α.

2°. Nous supposons que à chaque opérateur $O_j (j = i+1, \ldots, n)$ nous avons attaché la formule F_j qui satisfait à la condition α.

Pour construire la formule F_i, attachée à l'opérateur O_i nous distinguerons deux cas.

2.1. L'opérateur O_i est l'opérateur d'attribution $y:=f(X)$. En ce cas nous supposerons que l'opérateur O_t est le successeur de l'opérateur O_i. Il résulte de l'hypothèse d'inductivité que la formule F_t est construite et qu'elle satisfait à la condition α.

2.1.1 Si la formule F_t ne contient pas de variables relativement à la cellule y et si y appartient à R alors

$$F_i::=[(y^1 = A_{F_t}[y:=f(X)] \ \&\ F_t]. \tag{I}$$

Par contre, si y n'appartient pas à R, alors

(II) $$F_i :: = F_t.$$

En effet si la formule F_t ne contient pas de variables relativement à la cellule y c'est que pendant la réalisation du processus de calcul, à partir de l'opérateur O_t et jusqu'à la fin, le contenu de la cellule y reste inchangé.

Par conséquent, si y appartient à R, l'état de la cellule y après réalisation de l'opérateur O_t est un état final, ce qui est exprimé par la formule (I).

Si y n'appartient pas à R, c'est que le résultat intermédiaire obtenu après réalisation de l'opérateur O_i est un résultat inutile.

En ce cas l'opérateur O_i est un opérateur inutile, ce qui est exprimé par la formule II.

2.1.2 Si la formule F_t contient la variable y^s, où $s \geqslant 2$, alors

(III) $$F_i :: = [F_t]^{y^s}_{A_{F_t}[y:=f(X)]}.$$

En ce cas le contenu de la cellule y obtenu après réalisation de l'opérateur O_i continue à être utilisé à titre de résultat intermédiaire.

2.1.3 Si la formule F_t contient une formule élémentaire attachée à un opérateur logique, où entre librement la variable y^1 ou une formule élémentaire attachée à un opérateur d'attribution dont l'expression du second membre contient la variable y^1 également, et y appartient à R, alors

(IV) $$F_i :: = [y^1 = A_{F_t}[y: = f(X)] \ \& \ [F_t]^{y_1}_{A_{F_t}[y:=f(X)]}] \ .$$

Si $y \bar{\in} R$, alors

(V) $$F_i :: = [F_t]^{y^1}_{A_{F_t}[y:=f(X)]}.$$

Il résulte de la condition 2.1.3 que l'état de la cellule y, obtenu après exécution de l'opérateur O_i, continue à être utilisé dans le proccesus de calcul à titre de résultat intermédiaire.

Par conséquent si $y \in R$, alors l'état respectif est en même temps un résultat final ce qui est exprimé par la formule (IV).

Si $y \bar{\in} R$, alors l'état correspondant de la cellule y ne représente qu'un résultat intermédiaire (formule V)).

2.2 L'opérateur O_i est l'opérateur logique $p(X)$. Nous supposerons que l'opérateur O_{i_1} est le successeur de l'opérateur O_i, suivant la flèche qui part de celui-ci et est marquée par le signe „+" et que l'opérateur O_{i_2} est

le successeur de l'opérateur O_i suivant la flèche qui part de celui-ci et est marquée par la digne „—". Il résulte de l'hypothèse d'inductivité que les formules F_{t_1} et F_{t_2} attachées aux opérateurs O_{i_1}, respectivement O_{i_2} sont construites et qu'elles satisfont à la condition α. Nous désignerons par φ la formule F_{t_1} & F_{t_2} et nous définissons.

(VI) $$F_i :: = [L_\varphi[p(X)] \supset E(F_{t_1}, F_{t_2})] \ \& \ [\neg L_\varphi[p(X)] \supset E(F_{t_2}, F_{t_1})].$$

Nous démontrerons que la formule F_i construite par la méthode indiquée, satisfait à la condition α.

Nous supposerons que la formule F_i a la forme (1). En ce cas l'expression $A_{F_t}[y: = f(X)]$ peut s'obtenir à l'aide des formules (9) ou (11), Nous supposerons que F_t ne satisfait pas à la condition. α.

Il existe alors les variables x^s et, $x^{s'}$ qui entrent dans F_i et pour lesquelles nous avons $s \geqslant 2$, $s' \geqslant 2$ et $s \neq s'$. Vu que par l'hypothèse inductive nous avons supposé que F_t satisfait à la condition α, il résulte que dans la formule F_t il entre seulement une de ces variables.

Nous supposerons que $x^{s'}$ entre dans F_t et que x^s entre dans l'expression $A_{F_t}[y: = f(X)]$.

Mais si l'expression $A_{F_t}[y: = f(X)]$ a été construite à l'aide de la formule (9) alors $s = s'$, tandis que si cette expression a étè construite à l'aide de la formule (11), alors $s = 2$ et $s' = 1$. Mais les conclusions auxquelles nous sommes arrivés contredisent l'hypothèse suivant laquelle $s' \neq s$ et $s' \geqslant 2$. Par conséquent, la formule F_i de la forme (I) satisfait à la condition α.

Nous supposerons que la formule F_i a la forme (III). En ce cas l'expression $A_{F_t}[y: = f(X)]$ s'obtient par application des formules de construction de cette expression, à partir de la formule (4). Il en résulte que la seule variable relativement à la cellule y et qui entre dans l'expression $A_{F_t}[y: = f(X)]$ est y^{s+1}, où $s \geqslant 2$. Alors la seule variable de rang $r \geqslant 2$ qui entre dans la formule F_t relativement à la cellule y, est y^{s+1}.

Par suite, relativement aux états de la cellule y, la formule (III) satisfait à la condition α.

La démonstration que la formule F_i de la forme (III) satisfait à la condition α par rapport aux autres variables également se fait de la même manière qu'au cas où la formule F_i est de la forme (I).

Nous supposerons que la formule F_i a la forme (IV) ou (V). En ce cas l'expression $A_{F_t}[y: = f(X)]$ s'obtient par application des formules correspondantes, à partir de la formule (V). Il en résulte que la seule variable, relative à la cellule y, et qui entre dans l'expression $A_{F_t}[y: = f(X)]$ est y^2.

Par suite la formule F_i ne contient pas d'autres variables relatives à la cellule y, que y^2 et y^1. Ceci signifie que relativement aux états de la cellule y, la formule F satisfait à la condition α. La démonstration du

fait que la formule F_i satisfait à la condition α par rapport aux autres variables, se fait de la même manière qu'au cas où F_i est de la forme (I).

Nous supposerons que F_i est de la forme (VI) et ne satisfait pas à la condition α. Il s'ensuit qu'il existe les variables x^s et $x^{s'}$ qui entrent dans la formule F_i et pour lesquelles $s \neq s'$, $s \geqslant 2$ et $s' \geqslant 2$. Il résulte de la définition de l'opération ER que chaqune des formules $E(F_{t_1}, F_{t_2})$ et $E(F_{t_2}, F_{t_3})$ ne peut contenir qu'une seule de ces variables.

Supposons par exemple que la variable x^s est contenue dans la formule élémentaire $L_\varphi(p(X))$ et que la variable $x^{s'}$ est contenue dans la formule $E(F_{t_1}, F_{t_2})$. Il résulte aussi de la définition de l'opération ER que la variable $x^{s'}$ n'entre que dans la formule F_{t_1} et qu'il n'existe pas d'autre variable $x^{s''}$, où $s'' < s'$, qui entre dans la formule F_{t_2}.

Il résulte de l'application de la formule (12) pour la construction de l'expression $L_\varphi[p(X)]$ que $s = s'$, ou que si $s = 2$, alors la formule $L_\varphi[p(X)]$ ne peut contenir la variable x^s que comme résultat de l'application de la formule (11), auquel cas $s' = 1$. Les deux cas contredisent l'hypothèse selon laquelle la formule F_i de la forme (VI) ne satisfait pas à la condition α.

3°. Après $n+1$ pas, nous obtenons la formule F_0 qui satisfait à la condition α. Du fait que la formule F_0 satisfait à la condition α il résulte que si elle contient deux variables relativement à la même cellule, alors l'une d'elles possède un rang égal à l'unité. Par conséquent, si $x_{i_1}^{\rho_1}, \ldots, x_{\rho_p}^{\rho_p}$, ou $\rho_j \geqslant 2 (j = 1, \ldots, p)$ sont toutes des variables qui entrent dans F_0 alors ρ_j est le rang maximal de la variable $x_{i_j}^{\rho_j}$. Conformément à la convention adoptée, $x_{i_j}^{\rho_j}$ représente le contenu initial de la cellule x_{i_j} et ce contenu est désigné par $x_{i_j}^0$. Nous remplaçons dans la formule F_0 toutes les variables $x_{i_k}^{\rho_k}$, où $\rho_k \geqslant 2$ par les variables $x_{i_k}^0$.

Après cette substitution la formule obtenue ne contient que les variables à rang égal à zéro ou à un. Nous désignerons cette formule par $F(X^0, X^1)$.

Toutes les variables de rang égal à l'unité qui entrent dans $F(X^0, X^1)$ représentent des résultats finaux de l'algorithme $\mathfrak{A}$. En effet, si la variable x^1 figurait dans l'expression du second membre d'une formule élémentaire attachée à un opérateur d'attribution ou dans une formule élémentaire attachée à un opérateur logique, alors $x \in M_c$. Cette conclusion ne peut être vraie, parce que il résulte des régles de construction des expressions $A_\varphi[z := g(X)]$ et $L_\varphi[p(X)]$ que dans ce cas la formule F_0 doit contenir x^0 et non x^1 ainsi que nous l'avions supposé. Par conséquent l'entrée de la variable x^1, dans la formule $F(X^0, X^1)$ est le premier membre d'une formule élémentaire attachée à un opérateur d'attribution. Les composantes du vecteur X^1 sont donc des éléments de l'ensemble R^1, et les composantes du vecteur X^0 sont des éléments de l'ensemble I^0.

THÉORÈME. *La formule $F(X^0, X^1)$ est une formule complètement attachée à l'algorithme $\mathfrak{A}$.*

Pour démontrer ce théorème, nous introduirons les notations suivantes.

Nous désignerons par $\mathfrak{A}_i$ l'algorithme schéma-graphe donné par le schéma-graphe Γ_i par l'interprétation $\{M_i, S_i; T \to f : R \to p\}$ où M_i est cette partie de la mémoire (qui peut constituer même la mémoire M qui est utilisée à la réalisation de l'algorithme $\mathfrak{A}$, à partir de l'exécution de l'opérateur O_i.

Le schéma-graphe Γ_i, consiste dans les opérateurs $O_i, \ldots, O_{i_r}$, où O_{i_r} est la sortie de l'algorithme $\mathfrak{A}$ et chaque opérateur $O_{i_k} (k = 1, 2, \ldots, r)$ est le successeur d'au moins un des opérateurs $O_i, O_{i_1}, \ldots, O_{i_{k-1}}$. Si le schéma-graphe de l'algorithme $\mathfrak{A}$ est donné, alors on peut toujours construire le schéma-graphe Γ_i par la suppression de tous les opérateurs de l'algorithme $\mathfrak{A}$, à l'exception des opérateurs $O_i, O_{i_1}, \ldots, O_{i_r}$ et des flèches qui partent des opérateurs supprimés.

Nous désignerons par ξ_i^0 et par ξ_i^1 les données initiales, respectivement les résultats finaux de l'algorithme $\mathfrak{A}_i$.

À titre de données initiales pour l'algorithme $\mathfrak{A}_i$ nous considérons les données initiales de l'algorithme $\mathfrak{A}$, et les résultats intermédiares, qui ont été obtenus par la réalisation du processus de calcul jusqu'à l'exécution de l'opérateur O_i, qui sont utilisés à obtenir les résultats finaux à partir de l'exécution de l'opérateur O_i. À titre de résultats finaux de l'algorithme $\mathfrak{A}_i$ nous considérons les résultats finaux de l'algorithme $\mathfrak{A}$ qui s'obtiennent par la réalisation de cet algorithme, à partir de l'exécution de l'opérateur O_i Nous désignerons par $F_i^*(\xi_i^0, \xi_i^1)$ la formule obtenue après le remplacement dans F_i des données initiales de l'algorithme $\mathfrak{A}_i$ par le vecteur ξ_i^0 et des résultats finaux par le vecteur ξ_i^1.

Nous allons démontrer que la formule $F_i^*(\xi_i^0, \xi_i^1)$ est une formule complètement attachée à l'algorithme $\mathfrak{A}_i$.

A. La formule F_n^* est une formule complètement attachée à l'algorithme $\mathfrak{A}_n$. En effet l'algorithme $\mathfrak{A}_n$ est l'algorithme vide, qui consiste en la réalisation de l'opérateur O_n, c'est-à-dire de la sortie de l'algorithme $\mathfrak{A}$.

La formule complètement attachée à cet algorithme est $\xi_n^0 = \xi_n^1$, où $\xi_n^1 := \xi_n^1 := 0$ et par suite la formule $F_n^n ::= (0 = 0)$ est complètement attachée à l'algorithme $\mathfrak{A}_n$.

B. Nous supposerons que la formule $F_j^* (j = i+1), \ldots, n)$ est complètement attachée à l'algorithme $\mathfrak{A}_j$.

B.1. Nous supposerons que F_i a la forme (I), où F_i ne contient pas de variables relativement à la cellule y. Alors y^1 est un résultat final de

l'algorithme $\mathfrak{A}$.Les résultats finaux de l'algorithme $\mathfrak{A}_i$ seront y^1, et ξ_t^1, où ξ_t^1 sont les résultats finaux de l'algorithme $\mathfrak{A}_t$. Nous désignerons par $\bar{\xi}_i^0$ les données initiales de l'algorithme $\mathfrak{A}_i$ qui sont utilisées pour l'obtention du résultat y^1. La formule $F_i^*(\xi_i^0, \xi_i^1)$ prendra la forme suivante

$$[y^1 = f(\bar{\xi}_i^0)] \;\&\; F_t^*(\xi_t^0, \xi_t^1). \tag{13}$$

Nous supposerons que la formule (13) est vraie, Alors $y' = f(\bar{\xi}_i^0)$ et $F_t^*(\xi_t^0, \xi_t^1)$ sont des formules vraies et par suite $\mathfrak{A}_t(\xi_t^0) = \xi_t^1$ et par la réalisation de l'opérateur O_i les données initiales $\bar{\xi}_i^0$ se transforment dans le résultat y^1. Par conséquent $\mathfrak{A}_i(\xi_i^0) = \xi_i^1$.

Nous supposerons que $\mathfrak{A}_i(\xi_i^0) = \xi_i^1$. Alors par la réalisation de l'opérateur O_i, les données initiales $\bar{\xi}_i^0$ seront transformées dans le résultat y^1, et nous avons également $\mathfrak{A}_t(\xi_t^0) = \xi_t^1$. Par conséquent les formules $y^1 = f(\bar{\xi}_i^0)$ et $F_t^*(\xi_t^0, \xi_t^1)$ sont vraies. Par conséquent la formule (13) est vraie.

B.2. Nous supposerons que la formule F_i est de la forme III où F_t contient la variable y^s, $s \geqslant 2$. En ce cas y^s est un résultat intermédiaire de l'algorithme $\mathfrak{A}_i$. Nous supposerons que la formule F_i^* est vraie. En ce cas ces formules $y^s = f(\bar{\xi}_i^0)$ et F_t^* sont vraies. Par suite $\mathfrak{A}_t(\xi_t^0) = \xi_t^1$ où la composante y^s du vecteur ξ_t^0 a été remplacée par l'expression $f(\bar{\xi}_i^0)$.

Il en résulte que $\mathfrak{A}_i(\bar{\xi}_i^0) = \xi_i^1$ où $\xi_i^1 = \xi_t^1$, et les données initiales de l'algorithme $\mathfrak{A}_i$ sont $\bar{\xi}_i^0$ et ξ_t^0.

Nous supposerons que $\mathfrak{A}_i(\xi_i^0) = \xi_i^1$. Alors $\mathfrak{A}_t(\xi_t^0) = \xi_t^1$ et après exécution de l'opérateur O_i, $\bar{\xi}_i^0$ se transforme en y^s.

Par conséquent, les formules $F_t^*(\xi_t^0, \xi_t^1)$ et $y^s = f(\bar{\xi}_t^0)$ sont vraies. Il en résulte que la formule F_i^* est vraie.

Si la formule F_i est de la forme IV ou V, la démonstration du théorème se fait de manière analogue aux cas 1 et 2.

B.3. Nous supposerons que F_i est de la forme VI où F_{t_1} et F_{t_2} sont des formules complètement attachées aux algorithmes $\mathfrak{A}_{t_1}$, respectivement $\mathfrak{A}_{t_2}$.

Nous supposerons que la formule F_i^* correspondante à F_i est vraie.

Alors sont vraies les formules $p(\xi_i^0) \supset E(F_{t_1}, F_{t_2})$ et $\neg p(\xi_i^0) \supset E(F_{t_2}, F_{t_1})$. Soit $p(\xi_i^0)$ une formule vraie.

Alors $E(F_{t_1}, F_{t_2})$ est vraie et par suite F_{t_1} est aussi vraie. Il résulte que $\mathfrak{A}_{t_1}(\xi_{t_1}^0) = \xi_{t_1}^1$.

Alors $\mathfrak{A}_i(\xi_i^0) = \xi_i^1$, ou $\xi_i^1 = \xi_{t_1}^1$, et les données initiales de l'algorithme $\mathfrak{A}_i$ sont $\bar{\xi}_i^0$ et $\xi_{t_1}^0$. Si nous supposons que $\neg p(\xi^\circ)$ est vraie, nous aboutissons à la conclusion que $\mathfrak{A}_i(\xi_i^0) = \xi_i^1$, ou $\xi_i^1 = \xi_{t_2}^1$ et les données initiales de l'algorithme $\mathfrak{A}_i$ sont $\bar{\xi}_i^0$ et $\xi_{t_2}^0$.

Nous supposerons que $\mathfrak{A}_i(\xi_i^0) = \xi_i^1$. En ce cas ou bien la formule $p(\bar{\xi}_i^0)$ est vraie, et $\mathfrak{A}_{t_1}(\xi_{t_1}^0) = \xi_{t_1}^1$ ou bien la formule $\neg p(\bar{\xi}_i^0)$ est vraie, et $\mathfrak{A}_{t_2}(\xi_{t_2}^0) = \xi_{t_2}^1$. Dans les deux cas, il en résulte immédiatement que la formule, F_i^* est vraie.

Nous avons démontré de la sorte le théorème énoncé pour la formule $F_i^*(\xi_i^0, \xi_i^1)$ et l'algorithme $\mathfrak{A}_i$. Le théorème est donc vrai aussi pour la formule F_0 et l'algorithme $\mathfrak{A}_0$, qui est l'algorithme $\mathfrak{A}$.

Exemple. Considérons l'algorithme schéma-graphe suivant:

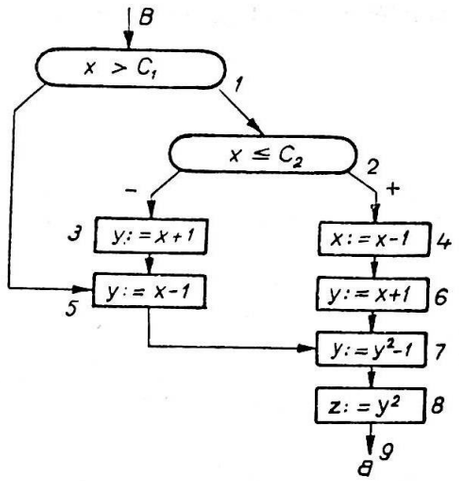

Fig. 1

en ce cas

$$N = \{x\},\ R = \{z\},\ M_c = \{c_1, c_2\}.$$

En appliquant la méthode de construction de la formule complètement attachée à cet algorithme, nous obtiendrons

$$F_9 :: = (O = O)$$

$$F_8 :: = (z^1 = (y^1)^2)$$

$$F_7 :: = (z^1 = ((y^2)^2 - 1)^2)$$

$$F_6 :: = (z^1 = ((x^1 + 1)^2 - 1)^2)$$

$$F_5 :: = (z^1 = ((x^1 - 1)^2 - 1)^2)$$

$$F_4 :: = (z^1 = ((x^2)^2 - 1)^2)$$

$$F_3 :: = (z^1 = ((x^1 - 1)^2 - 1)^2)$$

$$F_2 :: = [(x^2 \leqslant c_2^0) \supset F_4] \ \& \ [\neg (x^2 \leqslant c_2^0) \supset [F^3]_{x^2}^{x^1}]$$

$$F_1 :: = [(x_1^2 > c_1^0) \supset F_2] \ \& \ [\neg (x_1^2 < c_1^0) \supset [F_5]_{x_2}^{x_1}].$$

Après remplacement de la variable x^2 par la variable x^0, nous obtiendrons la formule

$$\{(x^0 > c_1^0) \supset [[(x^0 \leqslant c_2^0) \supset (z^1 = ((x^0)^2 - 1)^2)] \ \& \ [\neg (x^0 \leqslant c_2) \supset$$
$$\supset (z^1 = ((x^0 - 1)^2 - 1)^2]]\} \ \& \ \{\neg (x^0 < c_1^0) \supset (z^1 = ((x^0)^2 - 1)^2)\},$$

qui est la formule complètement attachée à l'algorithme considéré.

BIBLIOGRAPHIE

[1] Berge C., Théorie des Graphes et ses applications. Dunod, Paris, 1958.
[2] Калужнин Л. А., *Об алгоритмизации математических задач*. Проблемы кибернетики, вып. 3 (1959).
[3] Karp M. R., *A nota on the application of graph theory to digital computer programming*. Inf. and Control, **3**, *2*, 170—190 (1960).
[4] Мартинюк В. В., *О некаторых методах анализа операторных схем*. Автореферат МГУ, 1963.
[5] Мунтяну Э., *Метод аналива граф-схемных алгорифмов*. Mathematica, **5(28)**, *2*, (1965).
[6] Prosser T. R., *Application of Boolean matrices to the analysis of flow diagrams*. Proc. Eastern Joint Computer Conf., 1959.

Reçu le 27. VI. 1966.

MATHEMATICA VOL. 9 (32), 1, 1967, pp. 129—140

Professor TIBERIU POPOVICIU
zu seinem 60. Geburtstag gewidmet

BEMERKUNGEN ZUM PROBLEM DER FERTIGUNGSPROGRAMMIERUNG

von

L. NÉMETI

Cluj

1. In den Arbeiten [2] und [3] wurde gezeigt, wie man das Reihenfolgeproblem in der Fertigungsprogrammierung als lineare Planungsaufgabe mit logischen Bedingungen aufschreibt. Hierbei wurde, besonders in [3], die Problemstellung verschiedenen, in der Praxis vorkommenden technologischen Sachverhalten gerecht. Dieselben gehen über die, in solchen Untersuchungen üblicherweise behandelten Bedingungen hinaus: von jeder Maschinenart gibt es mehrere, im Wesentlichen identische Exemplare, und ein gegebener Arbeitsgang kann auf verschiedenen Maschinenarten gleicherweise durchgeführt werden.

Die in obligen Arbeiten verwendete Bezeichnungsweise benützt für verschiedene Konstante und Unbekannte zwei Indices. Wir wollen zunächst darstellen, wie man mit einem einzigen Zeiger auskommt, wobei man ausser der einfacheren Schreibweise auch noch andere Vorteile erhält.

Es seien m verschiedene Maschinenarten vorhanden. Dieselben seien, zur Identifizierung, numeriert (von 1 bis m). Von der Maschinenart i gibt es μ_i identische Exemplare, $i \in M = \{1, 2, \ldots, m\}$.

Es seien n verschiedene Arbeitsgänge auf den gegebenen Bearbeitungsmaschinen durchzuführen (in dieser Arbeit wird im Gegensatz zu [2] und [3], der Begriff des Erzeugnisses nicht verwendet; es wird nur mit Arbeitsgängen operiert); wir setzen $N = \{1, \ldots, n\}$. Jeder Arbeitsgang kann auf verschiedenen Maschinenarten durchgeführt werden. Einen bestimmten Arbeitsgang, durchgeführt auf einer bestimmten Maschinenart, nennen wir eine (technologische) Variante. Die Varianten seien durchlaufend numeriert, von 1 bis $\overline{n}$ $(\overline{n} \geqslant n)$; wir setzen $\overline{N} = \{1, \ldots, \overline{n}\}$. Die Zeiger (Lauf-