and the branch of arccos $\frac{1}{r}$ lies in the interval $\left(0, \frac{\pi}{2}\right)$. The eliminating of ψ from (24) and (25) yields the above considered equation $h(r, \rho) = 0$. For $\rho = 0$ we have $\psi^*(0) \approx 1{,}44$ and $r^*(0) = r_s \approx 2{,}83$. The function $\psi^*(\rho)$ decreases with increasing ρ and we have $\lim_{\rho \uparrow 1} \psi^*(\rho) = \tilde{\psi} \approx 1{,}218$ and $\lim_{\rho \uparrow 1} r^*(\rho) = \tilde{r} = r^*(1) \approx 1{,}50$.

REFERENCES

[1] Goluzin G. M., *Interior problems of the theory of schlicht functions*, Uspehi Mat. Nauk, **6**, 26–89 (1939).

[2] Mocanu P. T., *Asupra razei de stelaritate a funcţiilor univalente*, Studii şi cercet. de mat., Cluj, **XI**, 337–341 (1960).

[3] — *Sur le rayon d'étoilement et le rayon de convexité de fonctions holomorphes*, Mathematica **4** (**27**), 1, 57–63 (1962).

[4] — *Asupra razei de convexitate a funcţiilor olomorfe*, Studia Univ. Babeş-Bolyai, Cluj, series Math.-Phys. **2**, 31–33 (1964).

[5] Pólya G. and Szegő G., *Aufgaben und Lehrsätze aus der Analysis* I, Berlin 1925.

[6] Rényi A., *On the geometry of conformal mapping*, Acta sci. Math. Szeged, **12** B, 215–222 (1950).

Received 22. I. 1966.

MATHEMATICA VOL. 8 (31), 1, 1966, pp. 103–108

En hommage à mon Professeur
TIBERIU POPOVICIU
à l'occasion de son 60-e anniversaire

CONTRÔLE AUTOMATIQUE DES PROGRAMMES POUR LES MACHINES ÉLECTRONIQUES À CALCUL

par

E. MUNTEANU

à Cluj

Le contrôle de la correctitude d'un programme, ayant son exécution, constitue une étape importante du processus de résolution des problèmes à l'aide d'une machine électronique à calcul.

La pratique a démontré combien grandes sont les difficultés que le programmateur doit affronter et combien de temps-machine on perd pour la réalisation de cette étape. C'est pour quoi, conjointement avec le développement des machines à calcul et avec, l'élargissement de leur sphère d'applicabilité, le problème devient toujours plus actuel de la vérification automatique des programmes.

Dans ce travail, on décrit l'algorithme d'un programme de contrôle basé sur des principes totalement différents de ceux qui sont à la base de la réalisation de la majorité des programmes de ce genre.

L'idée de l'algorithme proposé consiste dans l'établissement du système de conditions logiques qui décrivent l'énoncé du problème dont on vérifie le programme, en partant seulement du programme respectif et de la représentation des données initiales et des résultats finaux du problème.

L'énoncé exact du problème est le suivant.

Considérons le programme P construit pour la résolution du problème $\mathcal{P}$. Nous supposerons que l'énoncé du problème $\mathcal{P}$ peut être décrit par une formule logique $\mathcal{F}(X_0, X_1)$ où X_0 est le vecteur des données initiales, tandis que X_1 est le vecteur des résultats finaux.

Définition 1. La formule $F(X_0, X_1)$ est dite formule *attachée* au programme P, si elle est vraie alors que la réalisation du programme P avec les données initiales X_0 entraîne l'obtention des résultats finaux X_1.

Définition 2. La formule $F(X_0, X_1)$ est dite formule *complètement attachée* au programme P, si elle est vraie ou cas et seulement au cas où la réalisation du programme P, à l'aide des données initiales X_0 entraîne l'obtention des résultats finaux X_1.

Si l'on peut construire une formule $F(X_0, X_1)$ que satisfait à la définition 1 ou à la définition 2, alors on dira que le programme P est correct par rapport au problème $\mathcal{P}$ si la formule

$$\forall X_0 \forall X_1 [F(X_0, X_1) \supset \mathcal{F}(X_0, X_1)]$$

est vraie.

Dans ce travail, nous décrirons une méthode de construction de la formule attachée, à un programme au cas où celui-ci est écrit sous forme de schéma-graphe.

Pour décrire un programme à l'aide d'un schèma-graphe, composé d'opérateurs d'attribution et d'opérateurs logiques [1], nous allons attacher à celui-ci les notions de mémoire et d'état de la memoire.

Un ensemble fini d'objects $M = \{x_1, x_2, \ldots, x_n\}$ est appelé *mémoire*. Les *éléments* de la mémoire s'appellent les *cellules*.

Considérons un ensemble $S = \{{}^0x_1, {}^0x_2, \ldots, {}^0x_n, \ldots, {}^jx_1x_2 \ldots {}^jx_n, \ldots\}$ dont les éléments seront appelés *états*. L'état ${}^jx_i$ s'appelle *le contenu* de la cellule x_i.

L'image ${}^jx_i = U(x_i)$ de l'ensemble M dans l'ensemble S s'appelle *état de la mémoire*. Le vecteur $\xi = ({}^jx_1, {}^jx_2, \ldots, {}^jx_n)$ s'appelle *vecteur complet* de l'état de la memoire. Nous désignerons par $\mathcal{S} = (\zeta)$ l'ensemble des vecteurs complets de tous les états possibles de la mémoire.

Soit $X = (x_1, x_2, \ldots, x_n)$, f, un symbole fonctionnel quelconque et p un symbole de relation. Nous entendrons par le terme *d'opérateur d'attribution* une expression de la forme

$$y := f(x)$$

où

$$y \in M.$$

Une expression de la forme $p(x)$ s'appelle *opérateur logique*. Dans les définitions des deux opérateurs, f et p sont définis sur toute la mémoire, mais à l'ordinaire dans les expressions $p(x)$ et $f(x)$ n'interviennent pas toutes les cellules de la mémoire.

Les cellules qui ne sont pas écrites dans ces expressions ne présentent pas d'intérêt pour l'analyse du programme considéré.

Définition 3. Une expression de la forme

$$\eta = f(\xi)$$

où $\eta \in S$ et $\xi \in \mathcal{S}$ s'appelle *formule élémentaire attachée* à l'opérateur d'attribution.

Définition 4. Une expression de la forme $p(\xi)$ où $\xi \in \mathcal{S}$ s'appelle *formule élémentaire attachée* à l'opérateur logique.

Pour les formules élémentaires, la même observation vaut que pour les opérateurs, c'est-à-dire que les états qui n'interviennent pas dans les expressions $f(\xi)$ et $p(\xi)$ ne présentent pas d'intérêt pour l'analyse du programme considéré.

À l'aide de la notion de formule élémentaire attachée à un opérateur, on démontre que pour un programme qui ne contient pas de cycles on peut toujours construire la formule complètement attachée au programme respectif.

L'algorithme de construction d'une telle formule est décrit dans le travail [2] et consiste dans l'analyse du programme de la sortie vers l'entrée, en construisant pour chaque opérateur la formule élémentaire attachée et en reliant entre elles, ces formules par les symboles des relations logiques qui correspondent à la structure logique du programme.

L'étude du programme qui contient des cycles à structure logique compliquée présente des difficultés considérables et on ne peut pas toujours construire une formule complètement attachée au programme. Il est évident que chaque formule identiquement vraie est une formule attachée à tout programme.

Une telle formule ne nous fournit cependant aucune indication sur les transformations qui s'effectuent par la réalisation du programme respectif. C'est pourquoi nous tendrons à obtenir une formule attachée au programme qui nous donne le plus d'information possible concernant celui-ci.

Dans ce qui suit nous proposerons une méthode de construction de la formule attachée à un programme au cas où le schéma-graphe correspondent peut être ramené à la forme normale.

Définition 5. On dira qu'un cycle possède une structure simple s'il est de la forme

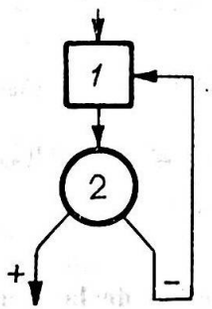

Fig. 1.

où le bloc 1 ne contient pas de cycles et l'opérateur 2 est l'opérateur logique de direction du cycle.

Nous dirons qu'un schéma-graphe est ramené à la forme normale s'il est une répétition et une superposition de cycles à structure simple.

Définition 6. Un cycle C d'un schéma-graphe ramené à la forme normale sera dit *maximal* si le schéma-graphe correspondant à ce cycle n'est pas intérieur à un autre cycle.

Nous allons considérer dans ce qui suit chaque cycle maximal comme un élément unitaire du schéma et l'appeler *opérateur* cyclique. Relativement aux opérateurs cycliques, chaque programme ramené à la forme normale ne contient pas de cycles.

Nous désignerons par ξ_0 le vecteur des données initiales du cycle considéré C, par ξ_1, le vecteur des résultats finaux et par ξ et ξ' l'état courant respectivement l'état suivant du cycle.

Définition 7. La formule $\varphi(\xi_0, \xi_1)$ s'appelle formule élémentaire *attachée au cycle* C, si elle est vraie au cas si l'on réalise le programme P par l'exécution du cycle C, les données initiales ξ_0 se transforment dans le résultat final ξ_1 du cycle.

Définition 8. La formule $\Phi(\xi, \xi')$ sera dite formule élémentaire attachée à un pas du cycle C si elle est vraie au cas où si on réalise le programme P par l'exécution d'un pas du cycle C, l'état courant ξ se transforme en l'état suivant ξ'.

La construction de la formule attachée à un cycle C à structure simple se fait de la manière suivante : nous désignerons par $p(\xi)$ la formule élémentaire attachée à l'opérateur logique 2. Nous construisons la formule complètement attachée au sous — programme représenté fig. 1 par le bloc 1.

Ce sous — programme ne contient pas de cycle, cela conformément à la définition du cycle à structure simple. La construction de cette formule s'effectue en appliquant la méthode décrite dans le travail [2] pour programmes sans cycles. Nous désignerons cette formule par $\Phi(\xi, \xi')$.

En ce cas si nous réussissons à construire une telle formule Ω de telle sorte que la formule

$$\forall\xi\forall\xi_0\forall\xi'[\neg p(\xi)\&\Phi(\xi, \xi')\&\Omega(\xi_0, \xi) \supset \Omega(\xi_0, \xi')]$$

soit vraie, alors

$$\varphi(\xi_0, \xi_1) \equiv p(\xi_1)\&\Omega(\xi_0, \xi_1)$$

est la formule attachée au cycle C.

Construction de la formule Ω

Nous désignerons par i l'état courant du paramètre du cycle C et par i' l'état suivant.

Nous supposerons que le paramètre du cycle varie suivant une progression arithmétique de raison h. La construction de la formule Ω s'effectue à l'aide d'un tableau T de la forme suivante :

Tableau T.

$\Phi(y, y')$	$\Omega({}^0y, y')$
$\alpha_1(y, y')$	$\beta_1({}^0y, y')$
$\alpha_2(y, y')$	$\beta_2({}^0y, y')$
. . . . .	. . . . .
$\alpha_r(y, y')$	$\beta_r({}^0y, y')$

où nous avons désigné par y l'état courant d'une cellule quelconque, par y', l'état suivant de la même cellule et par ${}^0y$, son état initial. Les formules de ce tableau sont construites de manière telle que la formule

$$(i' = i + h)\ \&\ \alpha_j(y, y')\ \&\ \beta_j({}^0y, y)\ \&\ \beta_j({}^0y, y')$$

où $j = 1, 2, \ldots, r$ soit vraie.

Un exemple pratique d'un tel tableau est le suivant :

$\Phi(y, y')$	$\Omega({}^0y, y')$
$y' > y$	$y' > {}^0y$
$y' = y$	$y' = {}^0y$
$y' = {}^0a_{i'}\, y + {}^0b_{i'}$	$y' = {}^0y \prod_{k=1}^{p} {}^0a_k + \sum_{j=1}^{p} \left({}^0b_j \prod_{k=j+1}^{p} {}^0a_k \right)$
$y' = f(i')y + g(i')$	$y' = {}^0y \prod_{k=1}^{p} f(i_0 + kh) + \sum_{j=1}^{p} g(i_0 + jh) \prod_{k=j+1}^{p} f(t_0 + kh)$
$\Phi_{i'}(y, y')$	$\forall[(1 \leqslant k \leqslant p) \supset \Phi_{i_0+kh}(y, y')]$

où $p = \frac{i' - i_0}{h}$, ${}^0a$, ${}^0b$ sont des constantes arbitraires et $\{{}^0a_k\}$ et $\{{}^0b_k\}$ des massifs arbitraires.

Si l'on se donne les formules $\Phi(\xi, \xi')$ et $p(\xi)$ ainsi que le tableau T, alors la formule Ω se construit de la manière suivante :

1°. $\Omega_1 = (0 = 0)$

2°. Nous supposerons que nous avons construit la formule Ω_k de telle sorte que la formule

(1) $$\neg p(\xi)\ \&\ \Phi(\xi, \xi')\ \&\ \Omega_k(\xi_0, \xi) \supset \Omega_k(\xi_0, \xi')$$

soit vraie.

Pour $k = 1$ cette formule est évidemment vraie.

)8

3°. Nous cherchons au tableau T une formule α_j telle que après substitution respective des variables qui figurent dans α_j par des variables ui figurent dans la formule (1) la formule

2) $$\urcorner p(\xi)\&\Phi(\xi, \xi')\&\Omega_k(\xi_0, \xi)\&\Omega_k(\xi_0, \xi') \supset \alpha_j(x, x')$$

oit vraie. Si nous trouvons une telle formule au tableau T, alors nous posons

$$\Omega_{k+1}(\xi_0, \xi') \equiv \Omega_k(\xi_0, \xi')\&\beta_j({}^0x, x')$$

où ${}^0x, x, x'$ sont des variables qui figurent dans la formule (1).

On démontre facilement que la formule obtenue de la formule (1) après remplacement de k par $k+1$ est une formule vraie.

Le processus de construction de la formule Ω continue aussi longtemps qu'il existe au tableau T des formules qui satisfassent à la relation (2).

La difficulté qui surgit à ce stade est de démontrer que la formule (2) est vraie. Pour vérifier la véracité de cette formule, on applique le calcul séquentiel G_1.

En utilisant cette méthode de construction de la formule élémentaire attachée à un cycle, la construction de la formule attachée à un programme s'effectue simplement, si l'on tient compte que, par rapport aux opérateurs cycliques tout programme à forme normale ne contient pas de cycles.

Le tableau introduit dans notre travail présente les cas les plus fréquemment rencontrés et il peut évidemment être élargi.

BIBLIOGRAPHIE

[1] Kalujnin L. A., *Ob algoritmizații matematiceschih zadaci*, Problemî kibernetiki, **2**, (1959).

[2] Munteanu E., *Metod analiza graf-shmîh algoritmov*, Mathematica **5 (28)**, 247–260 (1963).

[3] Prosser R. T., *Application of Boolean matrices to the analysis of flow diagrams*, Proc, Eastern Joint Compaceter Conf. 1959.

Reçu le 25. VI. 1966.

MATHEMATICA VOL. 8 (31), 1, 1966, pp. 109–115

En hommage au Professeur TIBERIU POPOVICIU
à l'occasion de son 60-e anniversaire

SUR LA PROGRAMMATION TEMPORELLE DE LA FABRICATION

par

L. NÉMETI, F. RADÓ

à Cluj

1. Le problème de la programmation temporelle de la fabrication (scheduling), dénommé encore problème d'ordonnancement (sequencing) apparaît dans la littérature mathématique en 1954, avec l'article de S. M. JOHNSON [9], dans lequel il a minimisé la durée totale de fabrication pour n produits et deux machines. Le problème plus général énoncé pour n produits et m machines, a été posé par plusieurs auteurs [19], [2], [14], [6], [4], [12]; dans leurs travaux ce problème a été transformé de diverses manières en des problèmes de programmation linéaire discrète, tous à un très grand nombre d'inconnues. En tenant compte que les méthodes de la résolution de la programmation discrète (par ex. [7]) comportent des difficultés de calcul, les auteurs mêmes de ces travaux reconnaissent que leurs méthodes ne sont pas pratiquement réalisables à l'étape actuelle de la technique du calcul.

L. NÉMETI a résolu ce problème dans [15] en suivant un autre principe: le problème de la programmation temporelle est transformé en un problème de programmation linéaire à conditions logiques. Un algorithme général pour ce problème-ci a été donné par F. RADÓ [16], [17]; cet algorithme ramène la programmation linéaire à conditions logiques à une suite de problèmes fondamentaux, qui sont des problèmes de programmation linéaire ordinaire. Dans le cas de la programmation temporelle, les problèmes fondamentaux prennent une forme spéciale, la même à laquelle ont été appliquées les méthodes: PERT [13], chemin critique [10], cheminement dans les graphes [18], l'algèbre de l'ordonnancement [3], [5].

La construction de la suite des problèmes fondamentaux, donnée dans [16], [17], utilise un principe dont l'importance pour divers problèmes