REVISTA DE ANALIZĂ NUMERICĂ ȘI TEORIA APROXIMAȚIEI
Volumul 1, Fascicola 1, 1972, pp. 49—52

NOI CONSIDERAȚII CU PRIVIRE LA LIMBAJELE DE PROGRAMARE

de

E. MUNTEANU

(Cluj)

Sistemele de operare și programare ale calculatoarelor electronice din generația a 3-a conțin fiecare diverse limbaje de programare orientate. Uneori, un sistem de operare și programare conține mai multe versiuni ale aceluiași limbaj, urmînd ca la generarea sistemului să se aleagă versiunea corespunzătoare aplicației care urmează a se realiza. Cele mai utilizate limbaje orientate sînt în prezent cele de tip FORTRAN, PL—1 și COBOL. Utilizarea acestor limbaje în activitatea de prelucrare automată a datelor a condus la o ușurare esențială a muncii de programare.

Experiența arată că există totuși unele greutăți în scrierea algoritmilor într-un limbaj de programare orientat chiar și atunci cînd expunerea algoritmului este făcută clar și precis. Aceste greutăți sînt pe de o parte legate de specificul calculatoarelor electronice în general. De exemplu necesitatea ca informația să aibă o structură asemănătoare cu structura în formă de adrese a memoriei calculatorului, este o particularitate specifică calculatoarelor electronice.

Aducerea unui algoritm la o formă comodă pentru scrierea lui într-un

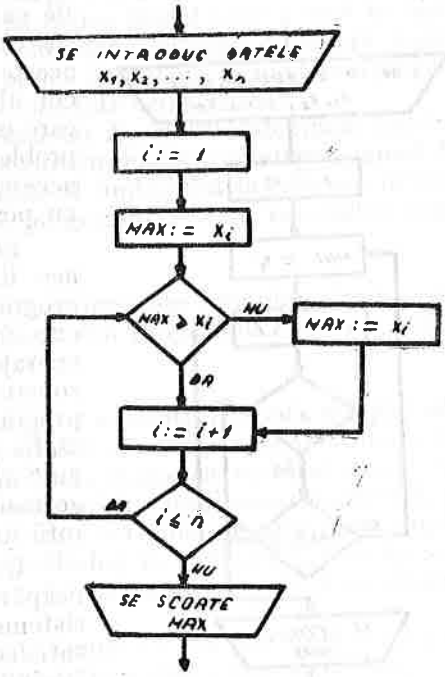

Fig. 1

limbaj de programare înseamnă o formalizare excesivă a acestuia, înseam
introducerea unor detalii tehnice care nu sînt legate de esenţa algo
mului, care sînt caracteristice limbajului în care urmează să se scrie prog
mul. De exemplu, pentru găsirea maximului numerelor $x_1, x_2, \ldots, x_n$ sche
logică a algoritmului este cea din fig. 1.

Iniţializarea indicelui i, introducerea variabilei MAX, mărirea indicel
cu o unitate precum şi condiţia de sfîrşit a ciclului sînt detalii tehnice c
nu ţin de esenţa problemei dar fără de care nu se poate ajunge comod la
program scris într-un limbaj de programare orientat. În acest caz esen
problemei constă în a descrie definiţia maximului care este un predicat
gic de forma :

$$[(1)\ [(\mathrm{MAX} := x_1) \vee (\mathrm{MAX} := x_2) \vee \ldots \vee (\mathrm{MAX} := x_n)] \,\&\, (\mathrm{MAX} \geqslant x_1) \,\&\, (\mathrm{MAX} \geqslant x_2) \,\&\, \ldots \,\&\, (\mathrm{MAX} \geqslant x_n)$$

Formula (1) nu reprezintă algoritmul de găsire a maximului. Din aceas
formulă se pot construi mai mulţi algoritmi cu acelaşi rezultat. Cel prezenta
în fig. 1. este o variantă, următoarea schemă logică (fig. 2) reprezintă
altă variantă.

Diferenţa între cele două scheme logice constă tocmai în detaliile d
ordin tehnic necesare pentru programare : iniţializări, alegerea modulu
de variaţie a indicelor, etc. Exemplul de ma
sus este un algoritm simplu. Deocamdată î
aceste cazuri simple este interesant de a înlo
cui algoritmul prin sistemul de relaţii logic
care există între datele iniţiale şi rezultatel
problemei. În această idee considerăm că est
necesar să se elaboreze limbaje de programar
cu posibilităţi principial noi.

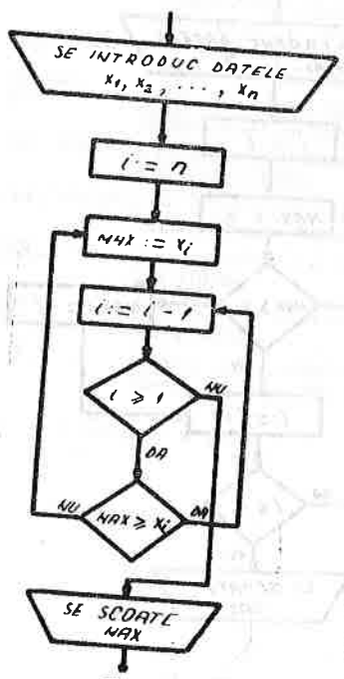

Fig. 2

Scrierea unui program într-un asemenea limbaj, spre deosebire de limbajele de programare orientate, nu trebuie să coincidă cu descrierea precisă a succesiunii de operaţii care trebuie realizate pentru rezolvarea problemei. Cu alte cuvinte un program într-un asemenea limbaj nu trebuie să fie neapărat un algoritm în înţelesul obişnuit al cuvîntului. Munca de elaborare a algoritmului propriu-zis urmează a fi transferată în sarcina compilatorului.

Un asemenea limbaj trebuie să conţină neapărat descrierea formulării problemei prin sistemul de condiţii logice care trebuie să le satisfacă datele iniţiale şi rezultatele finale. Posibilitatea descrierii unei probleme printr-un sistem de condiţii logice trebuie să fie

mitată deoarece în caz contrar ar însemna ca un compilator corespunzător unui asemenea limbaj să fie capabil să construiască algoritmul de rezolvare al oricărei probleme.

Astfel devine actuală formularea următoarei probleme : Să se elaboreze un limbaj de programare şi un compilator care să permită, în cazurile cele mai simple construirea în mod automat a programului pornindu-se de la sistemul de relaţii logice existent între datele iniţiale şi rezultatele executării programului.

Avantajul unor asemenea limbaje este pe de o parte o uşurare a muncii de programare iar pe de altă parte posibilitatea de a rezolva mult mai efficient analiza unui program. Se ştie că toate compilatoarele, care există în prezent în funcţiune, realizează în cadrul etapei de analiză sintactică şi o verificare a existenţei contradicţiilor de ordin formal într-un program. De exemplu, un compilator FORTRAN semnalează ca eroare sintactică prezenţa unor etichete într-un program, cărora nu le corespunde nici o instrucţiune de salt.

Greutăţile întîlnite la depanarea programelor nu sînt cele de ordin sintactic ci cele de odin semantic. Pentru acestea nu există pînă în prezent o metodă de semnalare a lor. Analiza semantică a unui program poate fi tratată astfel:

Pornind de la programul scris într-un limbaj de programare se construieşte sistemul de relaţii logice între datele iniţiale şi rezultatele finale. Alături de acest sistem există sistemul de relaţii logice iniţial care a stat la baza scrierii programului. Prin compararea celor două sisteme de relaţii logice se poate determina corectitudinea din punct de vedere semantic a programului. Formularea exactă a acestei probleme este următoarea :

Fie X_0, datele iniţiale ale unei probleme iar X_1, rezultatele finale. Pentru rezolvarea problemei respective s-a construit în mod automat programul P. Aceasta înseamnă că există un compilator al limbajului în care s-a descris algoritmul de rezolvare a problemei şi programul P este rezultatul compilării.

D e f i n i ţ i a 1. Formula $F(X_0, X_1)$ se numeşte formulă *ataşată* programului P, dacă ea este adevărată atunci cînd programul P, se execută cu datele iniţiale X_0 şi are ca rezultat datele X_1.

D e f i n i ţ i a 2. Formula $F(X_0, X_1)$ se numeşte formulă *complet ataşată* programului P, dacă ea este adevărată atunci şi numai atunci cînd programul P se execută cu datele iniţiale X_0 şi are ca rezultat datele X_1.

Rezolvarea problemei de analiză semantică a unui program constă, pe lîngă elaborarea unei metode de construire a formulei ataşate sau complet ataşate unui program şi în elaborarea unui limbaj în care urmează să fie scris programul P şi sistemul de relaţii logice dintre datele iniţiale şi rezultatele finale.

Presupunem că pentru scrierea programului P s-a pornit de la un algoritm de rezolvare a problemei la baza căreia a stat predicatul $F(X_0, X_1)$ definit pe mulţimerea datelor iniţiale şi a rezultatelor finale.

Dacă reuşim să construim o formulă $F(X_0, X_1)$ care să satisfacă finiţia 1 sau 2, atunci vom spune că programul P este corect atît sinta cît şi semantic dacă formula

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

este adevărată.

Într-o lucrare ulterioară vom prezenta un limbaj de programare c să răspundă cerinţelor enunţate în acestă notă.

NOUVELLES CONSIDÉRATIONS RELATIVES AUX LANGAGES D PROGRAMMATION

RÉSUMÉ

Dans cette note on présente un point de vue nouveau relatif à l'éla boration des langages de programmation. La nouvelle manière d'aborde les langages de programmation permet une automatisation de la con struction des algorithmes de même qu'une étude du problème d'analyse sémantique des programmes.

BIBLIOGRFIE

[1] Manna Zohar, *The correctness of programs*. J. Comput. System. Sci. 3, 119–127 (1969).

Primit la 30. XII. 1971.