REVISTA DE ANALIZĂ NUMERICĂ ŞI TEORIA APROXIMAŢIEI
Volumul 3. Fascicola 2, 1974, pp. 187—201

RELAŢII DE INTERACŢIUNE INTR-UN SISTEM DE CORUTINE

de

A. PRODAN şi T. RUS

(Cluj)

§ 1. Introducere

În activitatea de programare a sistemelor software, deseori se caută posibiltatea divizării întregului sistem în părţi componente, în aşa fel ca întregul program să se exprime ca o compunere a părţilor lui, iar interacţiunea dintre aceste părţi să fie minimă ; altfel spus, părţile componente ale sistemului să aibă o independenţă funcţională. Prin standardizarea unor *astfel de părţi* de programe se ajunge la noţiunea de *modul*. După cum este firesc, posibilitatea definirii unor astfel de module standard şi de descompunere a unui program în module este o propietate a nivelului lingvistic în care se exprimă programul.

Ridicarea eficienţei muncii de proiectare şi implementare a sistemelor software este strîns legată de posibilitatea definirii unor tehnologii de obţinere a acestor produse. Acestea la rîndul lor sînt legate de o analiză profundă a noţiunii de modul şi a nivelelor lingvistice care posedă proprietăţi ce permit definirea modulelor şi operaţiilor cu module. Primele încercări relative la definirea modulelor în scopul notat mai sus se găsesc în lucrările [3,4] folosindu-se noţiunea de corutină.

În iulie 1968 a avut loc la Boston un simpozion asupra programării modulare în care s-au exprimat diferite idei relative la programarea modulară mergîndu-se de la principii vagi pînă la definiţii formale. În Cursul avansat de Sotware Engineering ţinut la München în februarie 1973, J. DENNIS [1] a reluat noţiunea de modul şi a precizat modul în care modulul poate fi considerat o ,,cărămidă'' la baza tehnologiilor de sistem software.

În lucrarea de faţă pornind de la o înţelegere intuitivă a noţiunii de modul considerăm cazul particular al corutinelor şi studiem atît relaţiile

definitorii ale unui astfel de concept cît şi relaţiile care se pot stabili între diferitele corutine într-un sistem. Aceste relaţii ne vor conduce la considerarea unor structuri definite asupra corutinelor, cum sînt acelea de sisteme echilibrate de corutine, care pot constitui concepte de bază în elaborarea unei tehnologii de sistem software.

Conceptul de bază de la care pronim este acela de *modul*, considerat intuitiv ca fiind acea sevenţă de program într-un sistem de programare, care satisface următoarele cerinţe:

(i) Orice modul trebuie să poată fi testat şi corectat individual, independent de contextul în care apare.

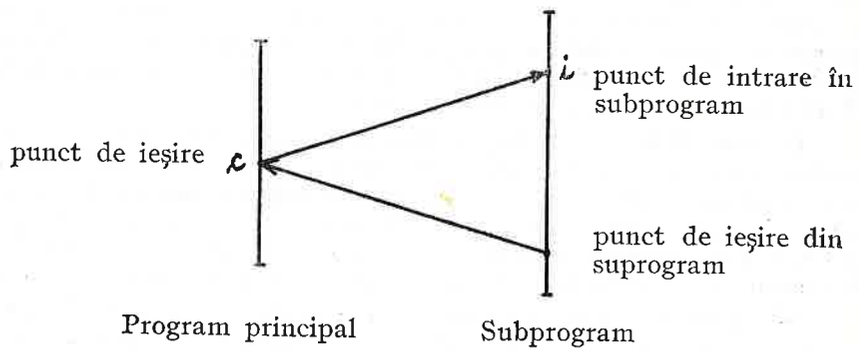
(ii) Diferitele module de program se pot asambla împreună independent de activitatea lor internă.

Intr-un sistem de module un lucru foarte important este modul în care aceste module se activează reciproc, întrucît de această interacţiune funcţională depinde indeplinirea sarcinii generale a sistemului prin integrarea funcţională a sarcinilor particulare ale modulelor componente.

În general există următoarea relaţie între două module M_1 şi M_2 în cazul în care M_1 apelează pe M_2:

Modulul M_1 avînd controlul se execută pînă se ajunge la *punctul de ieşire c*. În acest punct se transmite controlul modului M_2 într-un punct „*i*" numit punct de intrare. Punctul de intrare „*i*" poate fi orice punct din modulul M_2, dar el este bine determinat în punctul de ieşire *c* al modului apelant M_1. Prin urmare punctul de intrare „*i*" nu depinde cu nimic de activitatea anterioară a modulului M_2, ci numai de punctul de ieşire *c*.

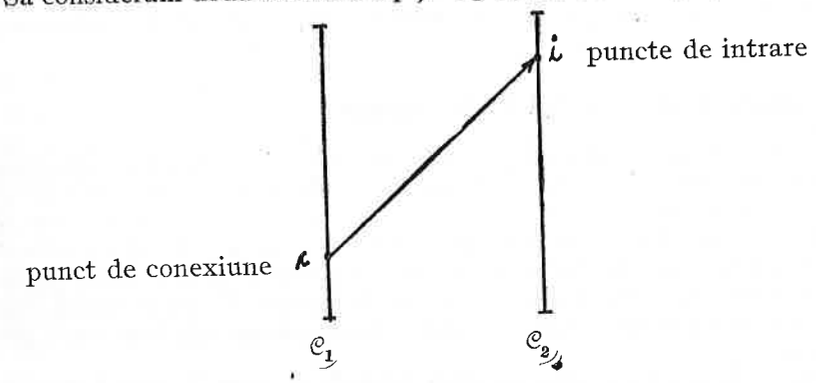
În cazul subprogramelor avem exact aceeaşi situaţie în ce priveşte punctul „*i*" de intrare în subprogram:



Bineînţeles că în acest caz, una din sarcinile dificile este determinarea precisă a unui punct de intrare „*i*" în fiecare punct de ieşire „*c*".

Corutinele sînt module care au anumite particularităţi atît în privinţa structurii lor cît mai ales a interacţiunii dintre ele.

Să considerăm două corutine $\mathcal{C}_1$ şi $\mathcal{C}_2$ astfel încît $\mathcal{C}_1$ apelează pe $\mathcal{C}_2$:



În punctul de ieşire „*c*" al corutinei $\mathcal{C}_1$ nu se determină decît corutina $\mathcal{C}_2$ căreia i se transmite controlul, nu şi punctul de intrare „*i*" în care $\mathcal{C}_2$ primeşte controlul. Acest punct de intrare „*i*" este o problemă internă a corutinei $\mathcal{C}_2$ şi el nu se poate preciza în punctul „*c*" al corutinei $\mathcal{C}_1$. Punctul de intrare fiind *o problemă intrinsecă*, el depinde de structura corutinei şi de activitatea sa anterioară. Mai precis dacă o corutină $\mathcal{C}$ a ajuns în timpul execuţiei la un punct de ieşire „*c*" în care ea apelează o altă corutină, atunci următorul punct în care va fi apelată corutina $\mathcal{C}$ este chiar punctul „*c*" în care ea a avut ultima dată controlul.

Acest lucru este valabil pentru orice corutină $\mathcal{C}$ dintr-un sistem $\mathcal{H}$ de corutine.

În cazul relaţiei program principal-suprogram, afirmaţia de mai sus este valabilă numai pentru programul principal.

Intr-adevăr, revenirea dintr-un subprogram în programul principal se face de obicei în punctul de ieşire *c* din care el a fost apelat; dacă eventual subprogramul doreşte să revină în alt punct al programului principal, el poate obţine acel punct *numai cu ajutorul lui c*, în caz contrar el îşi pierde calitatea de suprogram. Bineînţeles că un subprogram poate apela alte subprograme faţă de care el este program principal şi în punctele de ieşire respective sînt adevărate afirmaţiile de mai sus.

Dar un program nu are absolut nici o pretenţie în privinţa punctului în care va fi apelat de către programul principal.

După cum am arătat mai sus, punctul de intrare în subprogram este precizat în punctul *c* al programului principal.

În fiecare apel, subprogramul primeşte sarcini concrete şi *discrete* din partea programului principal, sarcini independente între ele. Aceste *sarcini discrete* sînt integrate de către programul principal în sarcina generală a sistemului.

§ 2. Studiul intern al corutinei

În cazul unui sistem de corutine $\mathfrak{K}$, o corutină $\mathcal{C} \in \mathfrak{K}$ nu este nici program-principal nici subprogram în raport cu celelalte corutine din sistem. În tot timpul execuţiei, activitatea unei corutine are următoarele caracteristici:

a) Independenţa relativă în funcţionare

Această independenţă rezultă din faptul că în timpul funcţionării o corutină primeşte controlul într-un punct de intrare care este precizat de către ea însăşi şi nu de către corutina apelantă. După cum s-a mai spus, punctul de intrare într-o corutină la un moment dat al execuţiei, este chiar punctul de ieşire în care acea corutină a transmis ultima dată controlul unei alte corutine. Prin urmare punctele de intrare într-o corutină coincid cu punctele de ieşire ale acelei corutine şi în continuare le vom numi *puncte de conexiune.*

Totuşi independenţa este relativă întrucît o corutină este legată funcţional de celelalte corutine din sistem.

b) Continuitatea în funcţionare

Relaţiile unei corutine cu celelalte corutine din sistem se stabilesc prin punctele de conexiune. Aceste puncte au atît calitatea de puncte de intrare cît şi de puncte de ieşire. Din modul cum se manifestă aceste două calităţi ale punctelor de conexiune în timpul funcţionării, rezultă continuitatea în funcţionare a unei corutine. Dacă în timpul execuţiei o corutină $\mathcal{C}$ a ajuns la punctul de conexiune c, acel punct de conexiune are mai întîi rolul de punct de ieşire cînd corutina $\mathcal{C}$ transmite controlul unei alte corutine, apoi are rolul de punct de intrare cînd corutina $\mathcal{C}$ primeşte din nou controlul.

Activitatea corutinei $\mathcal{C}$ după reprimirea controlului în punctul c este strict legată de activitatea sa de dinainte de a ajunge în punctul c. Acest lucru fiind valabil pentru oricare punct de conexiune din $\mathcal{C}$, în timpul execuţiei, rezultă de aici continuitatea în funcţionare a corutinei $\mathcal{C}$.

O corutină se poate reprezenta cu ajutorul unui graf în felul următor:

— punctele de conexiune se reprezintă prin nodurile grafului;

— secvenţa de instrucţiuni dintre două puncte de conexiune (privită sub aspect funcţional, adică respectînd instrucţiunile de ramificare) se reprezintă printr-o săgeată (arc orientat) care leagă cele două noduri corespunzătoare punctelor de conexiune respective. Aceste secvenţe reprezentate prin săgeţi se numesc *acţiuni ale corutinei.*

Dintr-un punct de conexiune pot pleca mai multe acţiuni şi într-un punct de conexiune pot sosi mai multe acţiuni.

Dacă o corutină $\mathcal{C}$ ajunge în timpul execuţiei la punctul de conexiune c, spunem că $\mathcal{C}$ se află în starea c.

Notînd starea corutinei $\mathcal{C}$ prin $s(\mathcal{C})$ se poate scrie:

$$s(\mathcal{C}) = c$$

În timpul funcţionării unui sistem de corutine, o corutină din sistem se poate afla în următoarele două situaţii:

— în activitate, şi atunci corutina execută una din acţiunile sale;

— în aşteptare, şi atunci corutina se află în unul din punctele de conexiune, care se numeşte starea corutinei în acel moment.

O corutină care se află în aşteptare în starea c, în momentul cînd va deveni activă va primi controlul chiar în punctul c. Apoi se va executa una din acţiunile care pleacă din punctul c şi se va ajunge la capătul acelei acţiuni la un alt punct de conexiune c_1. Aici corutina va trece în aşteptare în starea c_1.

După cum se vede mulţimea punctelor de conexiune ale unei corutine coincide cu mulţimea stărilor acelei corutine. Noţiunea de *punct de conexiune* este legată de relaţiile unei corutine cu celelalte corutine din sistem, iar cea de *stare* este legată de activitatea internă a corutinei.

Dintre punctele de conexiune ale unei corutine, două se pot deosebi de celelalte:

— unul care constituie *starea iniţială* a corutinei, adică starea în care aceasta se află la începutul execuţiei;

— unul care constituie *starea finală* a corutinei, adică starea de la sfîrşitul execuţiei.

Din proprietatea de *continuitate în funcţionare* a corutinei, rezultă următoarele proprietăţi pentru graful ataşat ei (în continuare nodurile grafului le vom numi stări sau puncte de conexiune, iar săgeţile le vom numi acţiuni):

1. În orice stare diferită de cea iniţială intră cel puţin o acţiune şi din orice stare diferită de cea finală iese cel puţin o acţiune.

Într-adevăr datorită proprietăţii de continuitate în funcţionare, dacă într-o stare nu intră nici o acţiune, starea respectivă este inaccesibilă.

De asemenea dacă dintr-o stare nu pleacă nici o acţiune, cînd corutina ajunge în acea stare nu mai are sens să fie apelată din nou întrucît nu-şi poate continua activitatea; rezultă de aici că o stare din care nu pleacă nici o acţiune nu poate fi decît starea finală.

2. În starea iniţială se poate să nu intre nici o acţiune şi din starea finală se poate să nu plece nici o acţiune, dar acest lucru nu este obligatoriu. Dacă în starea iniţială nu intră nici o acţiune, corutina se află în această stare numai la începutul execuţiei. De asemenea dacă din starea finală nu pleacă nici o acţiune, corutina se află în această stare numai la sfîrşitul execuţiei.

3. Oricare ar fi o stare c a corutinei $\mathcal{C}$, există cel puţin un drum între starea iniţială şi cea finală, care trece prin c.

Într-adevăr, dacă n-ar exista cel puţin un drum, pe baza continuităţii în funcţionare, starea c ar fi inaccesibilă pentru corutina $\mathcal{C}$.

Caracteristicile de *independenţă relativă în funcţionare şi continuitatea în funcţionare* pot fi ridicate la rangul de *principii interne de funcţionare a unei corutine*, întrucît ele sînt respectate de orice corutină.

Trebuie totuşi observat că al doilea principiu este mai tare decît primul şi prin urmare respectarea celui de al doilea principiu implică respectarea primului. Intr-adevăr continuitatea în funcţionare a unei corutine presupune determinarea succesiunii stărilor în timpul execuţiei prin înlănţuirea acţiunilor.

Corutina îşi determină noua stare s_1 din vechea stare s prin execuţia unei acţiuni. Noua stare s_1 va fi în acelaşi timp următorul punct de intrare în care va primi controlul corutina $\mathcal{C}$. Întrucît s_1 a fost determinat în interiorul corutinei $\mathcal{C}$ prin execuţia unei acţiuni care este un element aparţinînd corutinei $\mathcal{C}$, rezultă că principiul independenţei relative în funcţionare este respectat.

Dar respectarea independenţei relative în funcţionare nu implică respectarea continuităţii în funcţionare.

Într-adevăr, o corutină îşi poate determina punctele de intrare în care va fi apelată de celelalte corutine în alt mod decît respectînd principiul continuităţii.

În general o corutină execută o anumită sarcină ori de cîte ori o altă corutină îi cere acest lucru. Pentru execuţia acestei sarcini, poate să apară una din următoarele situaţii:

— se poate executa una sau mai multe acţiuni de către corutina respectivă;

— corutina poate cere altor corutine execuţia altor sarcini care vor fi integrate în sarcina sa.

De aceea se impune ca punctele de conexiune ale unei corutine să fie de două feluri:

— puncte de cerere

— puncte de transmitere.

Prin urmare o corutină C este caracterizată de următoarele elemente:

(*i*) o mulţime finită de *puncte de conexiune*, care se mai numeşte şi mulţimea *stărilor* corutinei respective şi care se notează cu S.

Această mulţime se descompune în două clase: mulţimea punctelor de *cerere*, notată cu C şi mulţimea punctelor de *transmitere* notată cu T astfel ca:

$$S = C \cup T \; ; C \cap T = \emptyset$$

Starea iniţială se notează cu s_0, iar cea finală cu s_F.

(ii) o mulţime finită de acţiuni notată cu A, care leagă între ele punctele de conexiune două cîte două. Mulţimea A stabileşte o relaţie a mulţimii S cu ea însăşi şi prin urmare această mulţime A se poate identifica cu o submulţime a produsului cartezian $S \times S$:

$$A \subseteq S \times S$$

După cum s-a mai spus, datorită continuităţii în funcţionare a unei corutine $\mathcal{C}$, mulţimea sa de acţiuni A are următoarea proprietate:

$$\forall\, s \in S \quad \exists\, s_1 \in S \quad \text{astfel încît} \quad (s, s_1) \in A\,;$$
$$\forall\, s \in S \quad \exists\, s_2 \in S \quad \text{astfel încît} \quad (s_2, s) \in A\,.$$

O corutină $\mathcal{C}$ care are mulţimea punctelor de conexiune $S = C \cup T$ şi mulţimea acţiunilor A, se poate nota astfel:

$$\mathcal{C} = (C, T, A)$$

Definiţia 1.

O corutină $\mathcal{C} \in \mathfrak{K}$ *este iniţială, dacă* $C = \emptyset$.

O corutină $\mathcal{C} \in \mathfrak{K}$ *este finală, dacă* $T = \emptyset$.

§ 3. Relaţii între corutine.

Pentru a studia amănunţit comportamentul unei corutine în timpul funcţionării sistemului este necesară o definire mai precisă a relaţiilor dintre corutine.

Relaţiile unei corutine $\mathcal{C} = (C, T, A)$ cu celelalte corutine din sistemul $\mathfrak{K}$ se stabilesc prin intermediul punctelor sale de conexiune $S = C \cup T$; $C \cap T = \emptyset$.

Prin fiecare punct de cerere $c \in C$ corutina $\mathcal{C}$ comunică unei anumite corutine $\mathcal{C}_1$ din sistem necesitatea îndeplinirii unei sarcini.

Corutina $\mathcal{C}_1$ căreia i se adresează cererea este unic determinată pentru un punct $c \in C$ pentru tot timpul funcţionării sistemului. Prin urmare cerinţele unei corutine $\mathcal{C} = (C, T, A)$ faţă de celelalte corutine din sistemul $\mathfrak{K}$ sînt stabilite printr-o funcţie pe care o notăm cu f:

$$f : C \to \mathfrak{K}$$
$$\forall\, c \in C\ \exists\,!\, \mathcal{C}_1 \in \mathfrak{K} \text{ astfel încît } f(c) = \mathcal{C}_1.$$

Prin intermediul funcţiei f se transmite controlul din punctul c al corutine $\mathcal{C}$ în punctul $s(\mathcal{C}_1)$ al corutinei $\mathcal{C}_1$ (care este starea corutinei $\mathcal{C}_1$ în acel monet).

Prin fiecare punct de transmitere $t \in T$ corutina $\mathcal{C}$ comunică unei anumite corutine $\mathcal{C}_2 \in \mathfrak{K}$ îndeplinirea sarcinii pe care a avut-o de îndeplinit. Corutina $\mathcal{C}_2$ căreia i se transmite acest lucru este unic determinată pentru un $t \in T$. Rezultă că şi transmiterea îndeplinirii sarcinilor de către o corutină se face pe baza unei funcţii pe care o notăm cu g;

$$g : T \to \mathfrak{K}$$
$$\forall\, t \in T\ \exists\,!\, \mathcal{C}_2 \in \mathfrak{K} \quad \text{astfel încît} \quad g(t) = \mathcal{C}_2.$$

Prin intermediul funcţiei g se transmite controlul din punctul t al corutinei $\mathcal{C}$ în punctul $s(\mathcal{C}_2)$ al corutinei $\mathcal{C}_2$.

Principiul continuităţii în funcţionare care este respectat de către toate corutinele unui sistem, implică anumite restricţii în relaţiile dintre corutine, faţă de relaţiile generale dintre module. Aceste restricţii sînt necesare pentru *coerenţa în funcţionare* a sistemului de corutine şi ele trebuie să fie bine definite pe baza unor principii.

Aceste principii ale relaţiilor dintre corutine la vom defini mai jos şi respectarea lor va fi condiţia necesară şi suficientă ca o mulţime de corutine să alcătuiască un *sistem de corutine.*

§ 4. Principiul coerenţei în funcţionare

Acest principiu precizează tipul stării în care trebuie să se găsească o corutină (stare de cerere sau de transmitere) în momentul în care ea este apelată de o altă corutină.

Dacă o corutină $\mathcal{C}$ cere unei alte corutine $\mathcal{C}_1$ îndeplinirea unei sarcini, $\mathcal{C}_1$ trebuie să se găsească în momentul apelului într-o stare de transmitere.

Dacă o corutină $\mathcal{C}$ transmire unei alte corutine $\mathcal{C}_2$ îndeplinirea unei sarcini, $\mathcal{C}_2$ trebuie să se găsească în momentul apelului într-o stare de cerere.

Dacă notăm $\mathcal{C} = (C, T, A, f, g)$, $\mathcal{C}_1 = (C_1, T_1, A_1, f_1, g_1)$, $\mathcal{C}_2 = (C_2, T_2, A_2, f_2, g_2)$, principiul de mai sus se poate exprima astfel:

Dacă o corutină $\mathcal{C} \in \mathfrak{K}$ a ajuns în timpul execuţiei într-un punct $c \in C$ şi $f(c) = \mathcal{C}_1$, trebuie ca în acel moment $s(\mathcal{C}_1) \in T_1$.

Dacă o corutină $\mathcal{C} \in \mathfrak{K}$ a ajuns în timpul execuţiei într-un punct $t \in T$ şi $g(t) = \mathcal{C}_2$, trebuie ca în acel moment $s(\mathcal{C}_2) \in C_2$. În continuare vom păstra notaţiile de mai sus.

Fie $\mathfrak{K}$ un sistem de corutine.

Să definim *relaţia de cerere* între corutinele sistemului în felul următor:

Dacă $\mathcal{C}_1, \mathcal{C}_2 \in \mathfrak{K}$,

spunem că $\mathcal{C}_1$ cere lui $\mathcal{C}_2$ execuţia unei sarcini şi notăm aceasta cu $\mathcal{C}_2 \succ\!\!- \mathcal{C}_1$, dacă $\exists\, c \in C_1$ astfel încît $f_1(c) = \mathcal{C}_2$.

Această relaţie se poate reprezenta printr-o submulţime a produsului cartezian $\mathfrak{K} \times \mathfrak{K}$ pe care o notăm cu $C(\mathfrak{K})$:

$$C(\mathfrak{K}) = \{(\mathcal{C}_1\ \mathcal{C}_2) \mid \exists\, c \in C_1 \text{ astfel încît } f_1(c) = \mathcal{C}_2\}$$

Relaţia de transmitere în sistemul $\mathfrak{K}$ se defineşte astfel:

Spunem că $\mathcal{C}_1$ transmite lui $\mathcal{C}_2$ îndeplinirea unei sarcini şi notăm aceasta prin $\mathcal{C}_1 \to \mathcal{C}_2$, dacă

$$\exists\, t \in T_1 \text{ astfel încît } g_1(t) = \mathcal{C}_2.$$

Această relaţie se poate reprezenta printr-o submulţime a prodului cartezian $\mathfrak{K} \times \mathfrak{K}$ pe care o notăm cu $T(\mathfrak{K})$:

$$T(\mathfrak{K}) = \{(\mathcal{C}_1, \mathcal{C}_2) \mid \exists\, t \in T_1 \text{ astfel încît } g_1(t) = \mathcal{C}_2\}$$

Fiind dată o corutină $\mathcal{C} \in \mathfrak{K}$, se notează cu $c(\mathcal{C})$ mulţimea corutinelor care pot să-i ceară lui $\mathcal{C}$ îndeplinirea unei sarcini:

$$c(\mathcal{C}) = \{\mathcal{C}_1 \mid \mathcal{C}_1 \in \mathfrak{K},\ (\mathcal{C}_1, \mathcal{C}) \in C(\mathfrak{K})\}$$

Cu $t(\mathcal{C})$ se notează mulţimea corutinelor cărora $\mathcal{C}$ le poate transmite îndeplinirea unei sarcini:

$$t(\mathcal{C}) = \{\mathcal{C}_1 \mid \mathcal{C}_1 \in \mathfrak{K},\ (\mathcal{C}, \mathcal{C}_1) \in T(\mathfrak{K})\}$$

Propoziţia 1:

$$\forall\, \mathcal{C} \in \mathfrak{K},\ \operatorname{card} |t(\mathcal{C})| \leqslant \operatorname{card} |T|$$

Demonstraţie:

Pentru a demonstra această relaţie, punem în evidenţă următoarea corespondenţă:

$$\forall\, \mathcal{C}_1 \in t(\mathcal{C}),\ (\mathcal{C}, \mathcal{C}_1) \in T(\mathfrak{K}) \Rightarrow \exists\, t \in T \text{ astfel încît } g(t) = \mathcal{C}_1.$$

Elementului $\mathcal{C}_1 \in t(\mathcal{C})$ facem să-i corespundă elementul $t \in T$ pentru care $g(t) = \mathcal{C}_1$.

Corespondenţa fiind realizată prin intermediul funcţiei $g: T \to \mathfrak{K}$, se ştie că pentru două corutine $\mathcal{C}_1 \neq \mathcal{C}_2$ din $t(\mathcal{C})$ există două puncte $t_1, t_2 \in T$, $t_1 \neq t_2$ astfel încît $g(t_1) = \mathcal{C}_1$ şi $g(t_2) = \mathcal{C}_2$. Prin urmare cardinalul mulţimii T este mai mare sau cel puţin egal cu cardinalul mulţimii $t(\mathcal{C})$.

Pentru a avea egalitatea $\operatorname{card} |T| = \operatorname{card} |t(\mathcal{C})|$ este necesar ca pentru oricare două puncte $t_1, t_2 \in T$, $t_1 \neq t_2$ să corespundă două corutine diferite:

$$g(t_1) = \mathcal{C}_1,\ g(t_2) = \mathcal{C}_2,\ \mathcal{C}_1 \neq \mathcal{C}_2.$$

Acest lucru se întîmplă numai atunci cînd g este o aplicaţie biunivocă.

Definiţia 2.

O corutină $\mathcal{C} \in \mathfrak{K}$ este echilibrată în transmitere, dacă $t(\mathcal{C}) = c(\mathcal{C})$.

Definiţia 3.

Un sistem de corutine $\mathfrak{K}$ este echilibrat în transmitere, dacă el conţine numai corutine echilibrate în transmitere.

Pentru o corutină $\mathcal{C} \in \mathfrak{K}$, se notează cu $\bar{c}(\mathcal{C})$ mulţimea corutinelor cărora corutina $\mathcal{C}$ poate să le ceară îndeplinirea unei sarcini:

$$\bar{c}(\mathcal{C}) = \{\mathcal{C}_1 \mid \mathcal{C}_1 \in \mathfrak{K},\ (\mathcal{C}, \mathcal{C}_1) \in C(\mathfrak{K})\}$$

Cu $\bar{t}(\mathcal{C})$ se notează mulţimea corutinelor care pot transmite corutinei $\mathcal{C}$ îndeplinirea unei sarcini:

$$\bar{t}(\mathcal{C}) = \{\mathcal{C}_1 \mid \mathcal{C}_1 \in \mathfrak{K},\ (\mathcal{C}_1,\ \mathcal{C}) \in T(\mathfrak{K})\}$$

P r o p o z i ţ i a 2.

$$\forall\, \mathcal{C} \in \mathfrak{K},\ \text{card}\,|\,\bar{c}(\mathcal{C})\,| \leqslant \text{card}\,|\,C\,|$$

Demonstraţie

Pentru $\forall\, \mathcal{C}_1 \in \bar{c}(\mathcal{C}),\ (\mathcal{C},\ \mathcal{C}_1) \in C(\mathfrak{K}) \Rightarrow \exists\, c \in C$ astfel încît $f(c) = C_1$. Intrucît f este o funcţie $f: C \to \mathfrak{K}$, pentru două corutine $\mathcal{C}_1 \neq \mathcal{C}_2$ din $\bar{c}(\mathcal{C})$ vor exista două puncte $c_1, c_2, \in C$, $c_1 \neq c_2$ astfel încît $f(c_1) = \mathcal{C}_1$ şi $f(c_2) = \mathcal{C}_2$.

Prin urmare cardinalul mulţimii C este mai mare sau cel puţin egal cu cardinalul mulţimii $\bar{c}(\mathcal{C})$.

Pentru a avea loc egalitatea card $|C|$ = card $|\,\bar{c}(\mathcal{C})\,|$ este necesar ca pentru oricare două puncte $c_1, c_2 \in C$, $c_1 \neq c_2$ să corespundă două corutine diferite: $f(c_1) = \mathcal{C}_1$, $f(c_2) = \mathcal{C}_2$, $\mathcal{C}_1 \neq \mathcal{C}_2$

Acest lucru se întîmplă numai atunci cînd f este o aplicaţie biunivocă.

D e f i n i ţ i a 4.

O corutină $\mathcal{C} \in \mathfrak{K}$ este echilibrată în cerere, dacă $\bar{c}(\mathcal{C}) = \bar{t}(\mathcal{C})$.

D e f i n i ţ i a 5.

Un sistem de corutine $\mathfrak{K}$ este echilibrat în cerere, dacă conţine numai corutine echilibrate în cerere.

D e f i n i ţ i a 6.

Relaţia de cooperare $R(\mathfrak{K})$ în sistemul $\mathfrak{K}$ se defineşte prin egalitatea:

$$R(\mathfrak{K}) = C(\mathfrak{K}) \cap T(\mathfrak{K})$$

P r o p o z i ţ i a 3

Fie $\mathfrak{K}$ un sistem de corutine.

$\mathfrak{K}$ este echilibrat în transmitere $\Leftrightarrow$ $\mathfrak{K}$ este echilibrat în cerere.

Demonstraţie

Vom demonstra mai întîi următoarea implicaţie:

$\mathfrak{K}$ este echilibrat în transmitere $\Rightarrow$ $\mathfrak{K}$ este echilibrat în cerere, adică:

$$\forall\, \mathcal{C} \in \mathfrak{K},\ t(\mathcal{C}) = c(\mathcal{C}) \Rightarrow \forall\, \mathcal{C} \in \mathfrak{K},\ \bar{t}(\mathcal{C}) = \bar{c}(\mathcal{C})$$

Acesta este echivalent cu a demonstra:

$$\exists\, \mathcal{C} \in \mathfrak{K},\ \bar{t}(\mathcal{C}) \neq \bar{c}(\mathcal{C}) \Rightarrow \exists\, \mathcal{C} \in \mathfrak{K},\ t(\mathcal{C}) \neq c(\mathcal{C})$$

Trebuie considerate următoarele două cazuri:

a) $\exists\, \mathcal{C}_1 \in \bar{t}(\mathcal{C})$ şi $\mathcal{C}_1 \notin \bar{c}(\mathcal{C})$

b) $\exists\, \mathcal{C}_1 \in \bar{c}(\mathcal{C})$ şi $\mathcal{C}_1 \notin \bar{t}(\mathcal{C})$

a) Dacă $\mathcal{C}_1 \in \bar{t}(\mathcal{C}) \Rightarrow (\mathcal{C}_1, \mathcal{C}) \in T(\mathfrak{K}) \Rightarrow \mathcal{C} \in t(\mathcal{C}_1)$ şi dacă $\mathcal{C}_1 \notin \bar{c}(\mathcal{C}) \Rightarrow$ $\Rightarrow (\mathcal{C},\ \mathcal{C}_1) \notin C(\mathfrak{K}) \Rightarrow \mathcal{C} \notin c(\mathcal{C}_1)$

Prin urmare în acest caz pentru corutina $\mathcal{C}_1$ este adevărată inegalitatea:

$$t(\mathcal{C}_1) \neq c(\mathcal{C}_1)$$

b) $\mathcal{C}_1 \in \bar{c}(\mathcal{C}) \Rightarrow (\mathcal{C},\ \mathcal{C}_1) \in C(\mathfrak{K}) \Rightarrow \mathcal{C} \in c(\mathcal{C}_1)$

$\mathcal{C}_1 \in \bar{t}(\mathcal{C}) \Rightarrow (\mathcal{C}_1,\ \mathcal{C}) \notin T(\mathfrak{K}) \Rightarrow \mathcal{C} \notin t(\mathcal{C}_1)$

Prin urmare şi în acest caz:

$$c(\mathcal{C}_1) \neq t(\mathcal{C}_1)$$

Implicaţia inversă se demonstrează în mod analog.

Prin urmare dacă un sistem $\mathfrak{K}$ este echilibrat în transmitere, este echilibrat şi în cerere şi invers. De aceea în continuare vom folosi noţiunea de *sistem echilibrat*, fără a mai preciza în transmitere sau în cerere.

P r p o z i ţ i a 4

Fie $\mathfrak{K}$ un sistem de corutine.

$\mathfrak{K}$ este echilibrat $\Leftrightarrow$ $C^{-1}(\mathfrak{K}) = T(\mathfrak{K})$

Demonstraţie

$\Rightarrow$ Fie $\mathfrak{K}$ un sistem echilibrat. Vom demonstra egalitatea, arătînd că sînt adevărate următoarele două incluziuni: $T(\mathfrak{K}) \subseteq C^{-1}(\mathfrak{K})$ şi $C^{-1}(\mathfrak{K}) \subseteq T(\mathfrak{K})$.

Fie $(\mathcal{C}_1,\ \mathcal{C}_2) \in T(\mathfrak{K})$. Rezultă că $\mathcal{C}_2 \in t(\mathcal{C}_1)$.

În acelaşi timp este adevărată egalitatea: $t(\mathcal{C}_1) = c(\mathcal{C}_1)$, $\mathfrak{K}$ fiind echilibrat.

Prin urmare $\mathcal{C}_2 \in c(\mathcal{C}_1) \Rightarrow (\mathcal{C}_2, \mathcal{C}_1) \in C(\mathfrak{K})' \Rightarrow (\mathcal{C}_1, \mathcal{C}_2) \in C^{-1}(\mathfrak{K})$

Rezultă deci că: $T(\mathfrak{K}) \subseteq C^{-1}(\mathfrak{K})$.

Incluzunea inversă se demostrează pe baza următoarelor implicaţii:

$$(\mathcal{C}_1, \mathcal{C}_2) \in C^{-1}(\mathfrak{K}) \Rightarrow (\mathcal{C}_2, \mathcal{C}_1) \in C(\mathfrak{K}) \Rightarrow \mathcal{C}_1 \in c(\mathcal{C}_1)$$

$$\left.\begin{array}{l} c(\mathcal{C}_1) = t(\mathcal{C}_1) \\ \mathcal{C}_2 \in c(\mathcal{C}_1) \end{array}\right\} \Rightarrow \mathcal{C}_2 \in t(\mathcal{C}_1) \Rightarrow (\mathcal{C}_1, \mathcal{C}_2) \in T(\mathfrak{K})$$

$\Leftarrow$ Fie $\mathfrak{K}$ un sistem de corutine care verifică egalitatea

$$T(\mathfrak{K}) = C^{-1}(\mathfrak{K})$$

Vom demonstra că pentru $\forall\, \mathcal{C} \in \mathfrak{K}$ este verificată egalitatea $t(\mathcal{C}) = c(\mathcal{C})$.

Dacă $\mathcal{C}_1 \in t(\mathcal{C})$, atunci $(\mathcal{C},\ \mathcal{C}_1) \in T(\mathfrak{K})$.

Pe baza ipotezei rezultă că $(\mathcal{C}, \mathcal{C}_1) \in C^{-1}(\mathcal{K})$ deci $(\mathcal{C}_1, \mathcal{C}) \in C(\mathcal{K})$ şi prin urmare $\mathcal{C}_1 \in c(\mathcal{C})$. Am demonstrat că $t(\mathcal{C}) \subseteq c(\mathcal{C})$. Incluziunea inversă se demonstrează analog.

§ 5. Aplicaţii ale sistemelor de corutine

După cum am notat în introducere, aplicaţia majoră a corutinelor este legată de metodologia de elaborare a componentelor software mari, cum ar fi de exemplu compilatoarele.

În lucrarea [3] se indică efectiv o astfel de aplicaţie pentru cazul unui compilator COBOL.

În cazul general putem presupune că obiectivul urmărit printr-un sistem de corutine este transformarea unui text sursă care aparţine unui limbaj de programare L1, într-un text obiect, care aparţine unui limbaj de programare L2.

Un astfel de sistem de corutine va constitui în acest caz un compilator din limbajul sursă L1 în limbajul obiect L2. Dacă L2 este limbajul intern al unui calculator atunci sistemul de corutine reprezintă o implementare a limbajului L1 pe calculatorul a cărui limbaj intern este L2.

Datorită faptului că memoria unui calculator este limitată, iar compilatoarele sînt programe de dimensiuni mari în general se adoptă tehnica de împărţire a compilatorului în mai multe module, la un moment dat fiind prezente în memorie un număr cît mai mic din aceste module. Sub aspect funcţional acest lucru se concretizează prin faptul că traducerea textului sursă în text obiect se face în mai mulţi paşi. Comunicarea între două module în acest caz se realizează prin texte intermediare care sînt memorate temporar pe benzi magnetice. În figura 1 se indică modul de funcţionare al unui compilator în patru paşi.

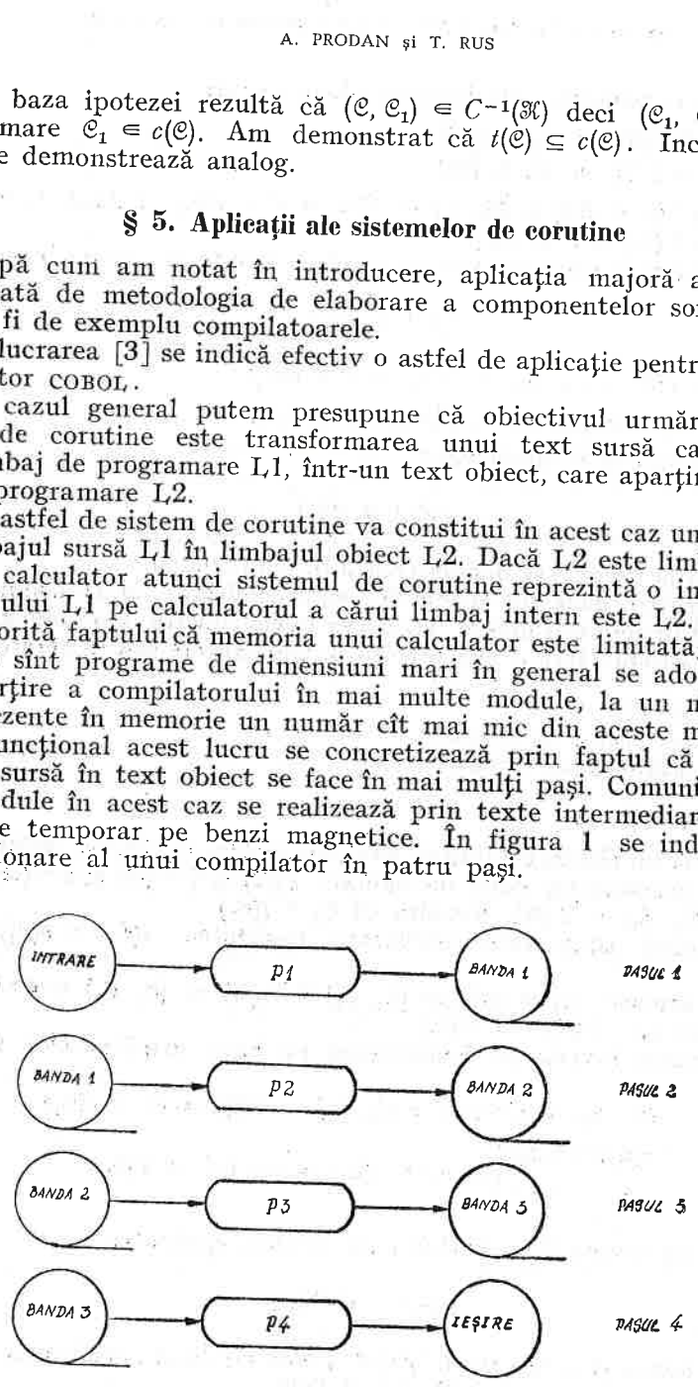


Figura 1

Numărul de paşi ai unui compilator este cu atît mai mare cu cît memoria calculatorului este mai mică.

Corutinele reprezintă o tehnică de programare deosebit de utilă dacă vrem să creăm un compilator care să se poată implementa pe mai multe calculatoare, care diferă între ele prin capacitatea memoriei.

În acest caz componentele compilatorului rămîn aceleaşi pentru toate calculatoarele. Ceea ce diferă de la un calculator la altul este interfaţa dintre aceste componente.

De exemplu dacă vrem să implementăm compilatorul din figura 1 pe un calculator în memoria căruia încap simultan P1 cu P2, atunci comunicarea dintre P1 şi P2 se va face direct în felul următor: cînd P1 vrea să transmită o informaţie apelează pe P2, iar cînd P2 are nevoie de o informaţie apelează pe P1, apelurile făcîndu-se astfel încît să se respecte principiul continuităţii în funcţionare atît pentru P1 cît şi pentru P2.

Dacă şi P3 încape împreună cu P4 în memorie, atunci compilatorul va funcţiona pe noul calculator în doi paşi, aşa cum se vede în figura 2:

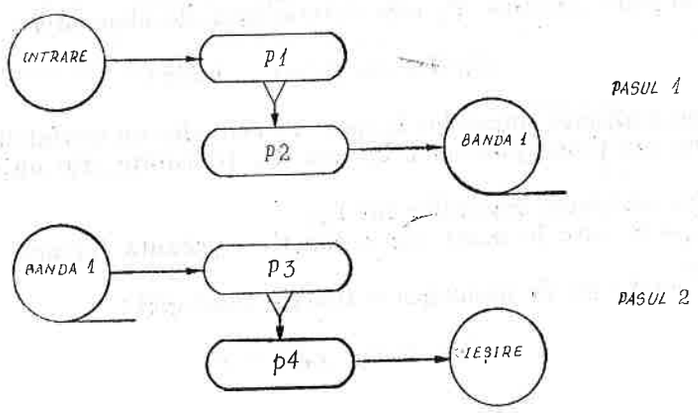


Figura 2

Dacă vrem să implementăm compilatorul pe un calculator în memoria căruia încap simultan P1, P2, P3, şi P4, atunci el va funcţiona într-un singur pas, ca în figura 3.

În acest caz compilatorul este format dintr-un sistem de patru corutine pe care îl notăm cu $\mathcal{Q}$.

Săgeata $\rightarrowtail$ nu semnifică pur şi simplu un salt de la un program la altul, ci chiar relaţia de cooperare între două corutine. În cazul sistemului $\mathcal{Q}$, relaţia de cooperare este stabilită de următoarea mulţime:

$$R(\mathcal{Q}) = \{(P_1, P_2), (P_2, P_3), (P_3, P_4)\}$$

Pe baza definiţiilor din paragrafele 3 şi 4, fiecare corutină din sistem este determinată de cinci elemente.

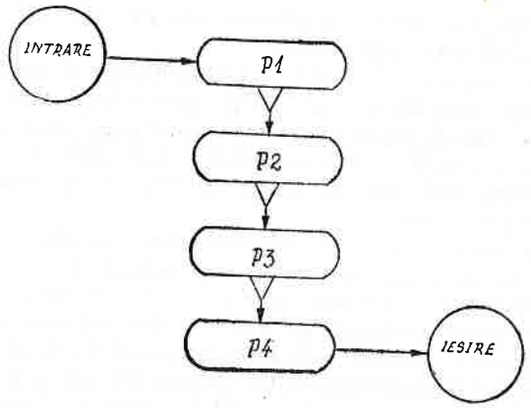


Figura 3

De exemplu corutina P_2 este determinată de elementele.

$$(C_2,\ T_2,\ A_2,\ f_2,\ g_2), \quad \text{unde:}$$

C_2 este mulţimea punctelor în care P_2 cere cîte un cuvînt lui P1;
T_2 este mulţimea punctelor în care P_2 transmite cîte un cuvînt lui P_3;

A_2 este mulţimea acţiunilor lui P_2;

$f_2 : C_2 \to \mathfrak{A}$ este în acest caz o funcţie constantă întrucît $\forall c \in C_2$, $f_2(c) = P_1$.

$g_2 : T_2 \to \mathfrak{A}$ este de asemenea o funcţie constantă:

$$\forall t \in T_2,\ g_2(t) = P_3$$

INTERACTION RELATIONS IN A COROUTINE SYSTEM

SUMMARY

The modularity is a main feature used in the programming techniques, by means of which the programer is able to define his program as a collection of independent units of actions. A coroutine is a particular case of module.

The paper is devoted to an algebrical study of the notions of coroutine and coroutine system. The purpose of the paper is to find the relations between the functional principles of the coroutines and their structural characteristics on different levels of coroutine system organization.

In the first section of the paper the notions of module, coroutine and coroutine system are briefly discussed. The section two is devoted to a

formal definition of the functional principles of a coroutine system. The main distinction between a coroutine system and a module system is discussed. On the base of the given functional principles the properties of the coroutine structure are given. The sections three and four contain some of the interaction relations between coroutines in a coroutine system The properties of these relations and their functional and structural consequences in a system are studied. În section five an example of used coroutine for a compiler implementation is given.

BIBLIOGRAFIE

[1] * * * *Advanced Course on Software Engineering*. Edit. F.L. Bauer, Springer — Verlag (Berlin), Heidelberg, New York, 1973.

[2] Dennis J. B., *Modularity. Project MAC*, Massachusetts Institute of Technology. Cambridge, Massachusetts, U.S.A.

[3] Conway M. E., *Design of a Separable transition — diagram compiler*. Comm. of the ACM, **6**, 7, pp 396—408 (1963),

[4]. Dijkstra, E. W., *Co — operating sequential processes. Programming Lnguages*, F. Genuys, Ed. Academic Press, New York, 1968.

Institutul de Tehnică de Calcul,
Filiala Cluj

Primit la 22. XI. 1973.