

RENDICONTI *del* SEMINARIO MATEMATICO *della* UNIVERSITÀ DI PADOVA

GIORGIO CASADEI

ROBERTO RIGHI

ANTONIO G. B. TEOLIS

**Una teoria algebrica per i « guarded
commands » di Dijkstra**

Rendiconti del Seminario Matematico della Università di Padova,
tome 59 (1978), p. 269-283

http://www.numdam.org/item?id=RSMUP_1978__59__269_0

© Rendiconti del Seminario Matematico della Università di Padova, 1978, tous droits réservés.

L'accès aux archives de la revue « Rendiconti del Seminario Matematico della Università di Padova » (<http://rendiconti.math.unipd.it/>) implique l'accord avec les conditions générales d'utilisation (<http://www.numdam.org/conditions>). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM

*Article numérisé dans le cadre du programme
Numérisation de documents anciens mathématiques*
<http://www.numdam.org/>

Una teoria algebrica per i « guarded commands » di Dijkstra.

GIORGIO CASADEI (*) - ROBERTO RIGHI (**)
ANTONIO G. B. TEOLIS (***)

0. Introduzione.

In questo lavoro si considera una teoria algebrica che permette, analogamente a quella costruita originariamente da Kott [0, 1, 2], di associare a qualsiasi flow-chart un sistema di equazioni algebriche e di pensare il programma ad essa corrispondente come la soluzione del sistema medesimo. La differenza con le precedenti teorie è che si introducono gli analoghi algebrici (anzichè dell'if-then-else e del do-while) delle strutture di controllo definite da Dijkstra [3], cioè:

- i) *if* $\langle \text{guarded-command list} \rangle$ *fi*
- ii) *do* $\langle \text{guarded-command list} \rangle$ *od*

ove

$$\begin{aligned} \langle \text{guarded-command list} \rangle &:= \langle \text{guarded-command} \rangle \mid \\ &\quad \langle \text{guarded-command list} \rangle \parallel \\ &\quad \parallel \langle \text{guarded-command} \rangle \end{aligned}$$

$$\langle \text{guarded-command} \rangle := \langle \text{predicate} \rangle \rightarrow \langle \text{statement list} \rangle$$

per la cui interpretazione si rimanda all'originale [3] ⁽¹⁾.

(*) Indirizzo dell'A.: Università di Padova, Istituto di Matematica Applicata.

(**) Indirizzo dell'A.: Università di Bologna, Istituto di Geometria.

(***) Indirizzo dell'A.: C.N.E.N., Centro di Calcolo di Bologna.

⁽¹⁾ La notazione qui impiegata è leggermente diversa dall'originale; in particolare è usato « list » invece di « set » per ragioni che saranno immediatamente evidenti.

Per comodità chiameremo le suddette due strutture di controllo rispettivamente IF e DO.

Nel paragrafo 1 sono richiamate le principali definizioni che riguardano l'ambiente teorico al quale appartiene il formalismo usato; nel paragrafo 2 si costruisce la opportuna algebra a più tipi e si chiarifica la relazione che essa ha con quella introdotta in [2]; nel paragrafo 3 si mostra come con le suddette strutture sia possibile scrivere programmi corrispondenti al più a flow-chart pathwise equivalenti ([4]) a una D-chart. Le strutture di Dijkstra, qualora il loro uso venga ristretto allo studio di flow-charts, pur non essendo più potenti dello if-then-else e del do-while, sono vantaggiose in quanto in taluni casi si viene ad evitare la duplicazione di parti di programma (node-splitting) non altrimenti evitabile usando le altre due strutture. Contrariamente alla situazione esposta in [2], in questo caso la espressione della soluzione del sistema algebrico non è unica e in questo lavoro non viene data una procedura risolutiva generale per la soluzione in una forma desiderata (per esempio quella più compatta, cioè con il minor numero di parti di programma duplicate); si definiscono invece casi particolari di ottimizzazione locale che, in certe condizioni, producono anche una ottimizzazione globale.

La teoria esposta è facilmente generalizzabile al caso di predicati a più (di due) valori; anche in questo caso le strutture di Dijkstra si rivelano di grande potenza (come del resto le strutture di Zahn).

1. Preliminari algebrici.

Con riferimento al formalismo introdotto in [5] si riprendono (in maniera non rigorosa) le seguenti definizioni.

Sia S un insieme, non vuoto, i cui elementi sono detti tipi. Sia $\mathcal{A}$ una famiglia di generici insiemi $\mathcal{A}_s$ con $s \in S$. Con w si indichi una parola su S , cioè $w \in S^*$; posto $w = s_1 s_2 \dots s_n$ ($s_i \in S$) la lunghezza n di w è indicata con $|w|$. Per ogni w e per ogni s , $\Sigma_{w,s}$ rappresenti un insieme di simboli (di operazioni da $\mathcal{A}_{s_1} \times \mathcal{A}_{s_2} \times \dots \times \mathcal{A}_{s_n}$ ad $\mathcal{A}_s$) la cui arità è $|w|$.

Una segnatura Σ è una famiglia di $\Sigma_{w,s}$. Un elemento di un generico $\Sigma_{w,s}$ di una segnatura viene indicato con σ .

Dato Σ una Σ -algebra si ottiene specificando gli insiemi $\mathcal{A}_s$ (cioè assegnando ogni sostegno $\mathcal{A}_s$ di tipo s) e associando ad ogni simbolo $\sigma \in \Sigma_{w,s}$ una effettiva operazione $\sigma_A: \mathcal{A}_{s_1} \times \mathcal{A}_{s_2} \times \dots \times \mathcal{A}_{s_n} \rightarrow \mathcal{A}_s$. Se λ rap-

presenta la parola vuota su S , $\Sigma_{\lambda,s}$ è l'insieme delle operazioni 0-arie di tipo s . Tutti gli elementi di A_s ottenibili come risultati di operazioni appartenenti a $\Sigma_{\lambda,s}$ si dicono costanti di A_s : ciascuna operazione 0-aria può essere indicata con la rispettiva costante.

Fissata una segnatura Σ si dice algebra delle parole (word-algebra) la particolare Σ -algebra, indicata con T_Σ , che si ottiene nel modo seguente. Sia O_Σ l'insieme di tutti i simboli di operazioni della segnatura Σ ($O_\Sigma = \bigcup_{w,s} \Sigma_{w,s}$); sia $(T_{\Sigma_s})_{s \in S}$ la più piccola famiglia di insiemi contenuti in $(O_\Sigma \cup \{ (,) \})^*$ tale che:

$$0) \Sigma_{\lambda,s} \subseteq T_{\Sigma_s};$$

$$1) \text{ se } \sigma \in \Sigma_{w,s}, w = s_1 s_2 \dots s_n, n > 0, \text{ e } t_i \in T_{\Sigma_{s_i}}, \text{ allora } \sigma(t_1 \dots t_n) \in T_{\Sigma_s}.$$

La T_Σ si ottiene prendendo la famiglia dei T_{Σ_s} come famiglia dei sostegni e come operazioni le seguenti:

$$0) \sigma \in \Sigma_{\lambda,s} \sigma_{T_\Sigma} = \sigma \in T_{\Sigma_s},$$

$$1) \sigma \in \Sigma_{w,s}, w = s_1 s_2 \dots s_n, t_i \in T_{\Sigma_{s_i}}: \sigma_{T_\Sigma}(t_1, t_2, \dots, t_n) = \sigma(t_1 t_2 \dots t_n) \in T_{\Sigma_s}.$$

Nel seguito, per la particolare segnatura Σ utilizzata in questo lavoro, si considera la Σ -algebra CT_Σ , che è sostanzialmente la Σ -word-algebra ai cui supporti sono state aggiunte parole (stringhe o alberi) di lunghezza infinita. Per maggiori dettagli si rimanda a [5].

2. Costruzione dell'ambiente.

Quando si applica questo formalismo allo studio di flow-charts, ad ogni tipo può essere data una ben precisa interpretazione. In particolare, nel presente lavoro, l'insieme dei tipi è

$$S = \{\alpha, \gamma, b, \pi, p, a, g\}$$

dove:

1) il tipo α individua genericamente parti di programma (che non contengono nè lo « start » nè lo « stop ») che hanno una sola entrata e una sola uscita: le costanti di tipo α sono le azioni elementari;

2) il tipo γ individua le code di programma (cioè parti di programma ad una sola entrata che terminano tutte con lo « stop »): la costante di tipo γ è Ω (cioè lo « stop »);

- 3) il tipo b individua i predicati (funzioni booleane);
- 4) il tipo π individua solo lo « start »;
- 5) il tipo p individua i programmi;
- 6) il tipo a individua guarded-command lists quando i predicati sono a guardia di oggetti di tipo α ;
- 7) il tipo g individua guarded-command lists quando i predicati sono a guardia di oggetti di tipo γ .

La segnatura Σ che si utilizza è:

$\Sigma_{\lambda,\gamma} = \{\Omega\}$	costante del tipo γ ,
$\Sigma_{\lambda,\pi} = \{\pi\}$	costante del tipo π ,
$\Sigma_{\lambda,p} = \emptyset$	costanti del tipo p ,
$\Sigma_{\lambda,b} = \{A, B, \dots\}$	costanti di tipo b (predicati),
$\Sigma_{\lambda,\alpha} = \{I, a, b, \dots\}$	costanti di tipo α (azioni elementari),
$\Sigma_{\lambda,g} = \emptyset$	costanti del tipo g ,
$\Sigma_{\lambda,a} = \emptyset$	costanti del tipo a ,
$\Sigma_{b,b} = \{-\}$	operazione unitaria not su predicati,
$\Sigma_{g,\gamma} = \{[\]\}$	operazione <i>if ... fi</i> su guarded-command list di tipo g ,
$\Sigma_{a,\alpha} = \{[\]\}$	operazione <i>if ... fi</i> su guarded-command list di tipo a ,
$\Sigma_{\pi\gamma,p} = \{\cdot\}$	} concatenazioni,
$\Sigma_{\alpha\alpha,\alpha} = \{\cdot\}$	
$\Sigma_{a\gamma,\gamma} = \{\cdot\}$	
$\Sigma_{b\alpha,a} = \{\rightarrow\}$	} concatenazione tra un predicato ed un oggetto di tipo α (γ) che dà un guarded-command di tipo a (g),
$\Sigma_{b\gamma,g} = \{\rightarrow\}$	
$\Sigma_{ga,g} = \{+\}$	} concatenazioni tra guarded-command lists ($[\]$).
$\Sigma_{aa,a} = \{+\}$	

Nella CT_{Σ} associata alle precedenti definizioni, per uniformità di notazione con la precedente letteratura [0, 1, 2, 3], si usano le scritture riportate di seguito nella colonna di sinistra anzichè quelle con-

formi alla notazione standard sopra definita (e scritte nella colonna di destra):

<i>notazione usata</i>	<i>notazione standard</i>
$t_1 \cdot t_2$ (oppure $t_1 t_2$)	$\cdot (t_1 t_2)$
$t_1 + t_2$	$+ (t_1 t_2)$
$[t]$	$[] (t)$
$\bar{t}$	$-(t)$
$B \rightarrow t$ (oppure Bt)	$\rightarrow (Bt)$

Il simbolo $\cdot$ che indica la concatenazione, che può interessare i vari tipi, è di solito omissso. Per semplicità si ometterà anche il simbolo $\rightarrow$ caratteristico del guarded-command nel quale associa ad ogni predicato l'azione che deve essere eseguita se questo è vero.

Si consideri la varietà equazionale V definita dalle seguenti equazioni, cioè l'insieme delle Σ -algebre che le soddisfano:

- | | |
|---|---|
| i) $(t_1 t_2) t_3 = t_1 (t_2 t_3)$ | t_1 di tipo α ,
t_2 di tipo α ,
t_3 di tipo α, γ ; |
| ii) $(t_1 + t_2) + t_3 = t_1 + (t_2 + t_3)$ | t_i di tipo g, a ($i = 1, 2, 3$); |
| iii) $It = t$ | t di tipo α, γ ; |
| iv) $At_1 + \bar{A}[Bt_2 + \bar{B}t_3] =$
$= At_1 + Bt_2 + \bar{B}t_3$ | t_1, t_2, t_3 di tipo α, γ ,
A, B di tipo b ; |
| v) $[At_1\xi + \dots + A_n t_n \xi] =$
$= [A_1 t_1 + \dots + A_n t_n] \xi$ | A_i di tipo b ($i = 1, 2, \dots, n$),
t_i di tipo α ($i = 1, 2, \dots, n$),
ξ di tipo α, γ . |

Come diretta conseguenza della iv) si ha:

$$\begin{aligned} & [A_1 t_1 + \bar{A}_1 [A_2 t_2 + \bar{A}_2 [\dots [A_n t_n + \bar{A}_n t_{n+1}] \dots]]] = \\ & = [A_1 t_1 + A_2 t_2 + \dots + A_n t_n + \bar{A}_n t_{n+1}]. \end{aligned}$$

Il considerare tale varietà equazionale è suggerito dal particolare uso che si farà dell'algebra costruita: le precedenti equazioni rispecchiano caratteristiche tipiche degli oggetti che costituiscono un programma: per esempio la i) afferma la associatività della concatenazione di statements, mentre la iii) identifica la particolare azione I con la identità (la azione che lascia inalterate le variabili del programma). La iv) assume un particolare significato: qualora si consideri

l'operazione - come l'operazione not, essa permette di identificare la struttura [... + ... + ... + ...] con l'IF, ove però il + non è commutativo a differenza di $\sqcup$ (perciò si è usato «list» invece di «set») e con la seguente ipotesi aggiuntiva:

I.A. Quando nella lista di guarded-command c'è più di un predicato vero, viene eseguita l'azione più a sinistra (nella somma) tra quelle il cui predicato guardia risulta vero.

In analogia con quanto fatto in [2] si definisce la $\Sigma(X)$ come la segnatura ottenuta dalla precedente Σ ponendo:

$$\Sigma_{\lambda, \gamma}(X) = \{\Omega\} \cup X,$$

$$\Sigma_{\lambda, x}(X) = \{x_0\},$$

ove X e $\{x_0\}$ sono insiemi disgiunti da Σ , i cui elementi verranno chiamati variabili (x_0 variabile principale) ⁽²⁾.

I sostegni di $CT_{\Sigma(X)}$ possono essere pensati come i sostegni di una Σ -algebra che chiameremo $CT_{\Sigma}(X)$, (in cui le x , appunto, sono scomparse come operazioni 0-arie e rimangono solo nei sostegni).

Come per le Σ -algebre è particolarmente significativa l'algebra CT_{Σ} così per una certa varietà V è particolarmente significativa l'algebra $CT_{\Sigma V}$, ottenuta introducendo in CT_{Σ} la relazione di equivalenza Q corrispondente in maniera naturale al sistema di equazioni che definisce la varietà V e ponendo $CT_{\Sigma V} = CT_{\Sigma}/Q$; si costruisce in maniera analoga $CT_{\Sigma V}(X)$ ⁽³⁾.

Si può quindi considerare un sistema E su $CT_{\Sigma V}(X)$ come una funzione $E: X_n \rightarrow CT_{\Sigma V}(X_n)$ ove X_n è una n -pla di elementi distinti di $X \cup \{x_0\}$, uno dei quali supporremo sempre sia x_0 .

DEF. 1.0. Un sistema è detto uniforme se ogni membro destro delle equazioni ha una delle due forme

$$x_i = ax_j \quad \text{oppure} \quad x_i = [Ax_j + \bar{A}x_k]$$

dove $a \in \Sigma_{\lambda, \alpha}$ e $A \in \Sigma_{\lambda, b}$.

Sia data una flow-chart. Ad essa è associabile un sistema uniforme secondo la seguente procedura: si pone una variabile x_j , $j = 1, \dots, n$,

⁽²⁾ Non si confondano queste variabili con le variabili del programma.

⁽³⁾ Per semplicità si indicherà un elemento di $CT_{\Sigma V}$ con un qualunque elemento equivalente di CT_{Σ} .

all'ingresso di ogni box della flow-chart, x_0 sullo start, Ω sullo stop, e si scrive per ogni x_j , $j = 0, 1, \dots, n$, una equazione ove il membro destro è così costruito:

$x_j = ax_k$ se il cammino computazionale tra x_j e x_k passa solo per il box rappresentativo dell'azione a ;

$x_j = [Ax_k + \bar{A}x_i]$ se x_j è all'entrata del predicato A e se il cammino computazionale va da x_j a x_k se A è vero e da x_j a x_i se A è falso ($\bar{A}$ è vero).

Ci si può facilmente convincere che, dato un sistema uniforme, è possibile associargli una flow-chart invertendo, per così dire, la procedura appena detta.

Si ricorda che anche nella teoria algebrica introdotta originariamente da Kott [0] si faceva una simile associazione (si veda anche [1, 2]). Dunque si può dare attraverso la flow-chart una corrispondenza tra sistemi di equazioni nell'algebra costruita dalla segnatura sopra introdotta — che chiamiamo A_D — e quelli nell'algebra introdotta in [2] — che chiameremo A_K .

DEF. 2.1. Siano E_1 ed E_2 due sistemi uniformi rispettivamente in A_D e A_K . E_1 ed E_2 sono detti equivalenti ($E_1 \leftrightarrow E_2$) se ad E_1 ed E_2 è associata la stessa flow-chart.

Una tale corrispondenza non si limita ai sistemi considerati globalmente, ma essendo uguale il modo di assegnare le variabili, essa può essere estesa prima ai sottinsiemi e poi ai singoli termini. Risulta in particolare

$$(2.0) \quad \begin{aligned} & \text{i) } [Ax_j + \bar{A}x_k] \leftrightarrow A(x_j, x_k). \\ & \text{ii) } ax_j \leftrightarrow ax_j. \end{aligned}$$

La i) esprime il fatto che la struttura di controllo if è un caso particolare di struttura IF quando i due predicati-guardia sono uno la negazione dell'altro. Inoltre dalla equazione iv) si ha ⁽⁴⁾:

$$(2.1) \quad \begin{aligned} [A_1\alpha_1 + A_2\alpha_2 + \dots + A_n\alpha_n + \bar{A}_n\alpha_{n+1}] = \\ = [A_1\alpha_1 + \bar{A}_1[A_2\alpha_2 + \dots [A_n\alpha_n + \bar{A}_n\alpha_{n+1}] \dots]] \leftrightarrow \\ \leftrightarrow A_1(\alpha_1, A_2(\alpha_2, \dots, A_n(\alpha_n, \alpha_{n+1}) \dots)). \end{aligned}$$

⁽⁴⁾ Questa equivalenza rispecchia anche la equivalenza fra struttura « case » e annidamento di if.

Si noti che la corrispondenza di cui sopra è strettamente legata al fatto di avere introdotto l'ipotesi aggiuntiva I.A. sulle strutture IF; d'altra parte il carattere di indeterminazione della IF è superfluo in un contesto di flow-charts deterministiche.

Inoltre, se si fosse considerata in [2] anche una particolare azione ω (l'« abort ») tale che $\omega t = \omega$, $\forall t$ di tipo α e $\omega \xi = \Omega$, $\forall \xi$ di tipo γ , allora si avrebbe anche la equivalenza

$$(2.2) \quad [At] \leftrightarrow A(t, \omega).$$

Così, in virtù delle (2.1) e (2.2) ad ogni struttura IF può essere fatta corrispondere un if o un annidamento di if, e ovviamente ad ogni if corrisponde una struttura IF.

Essendosi stabilita una corrispondenza tra le strutture elementari, questa dà la possibilità di confrontare sistemi di qualsiasi tipo scritti nelle due differenti notazioni, cioè in A_D e A_K . Inoltre, ricordando le (2.0) e le definizioni di sistema uniforme date per A_D e A_K , si ha che a sistemi uniformi in una delle due algebre corrispondono sistemi uniformi nell'altra.

3. Principali risultati.

Si consideri la equazione al punto unito:

$$(3.0) \quad x_i = [A\alpha x_i + \bar{A}x_k]$$

a cui corrisponde, per quanto detto nel paragrafo precedente, la equazione in A_K :

$$(3.1) \quad x_i = A(\alpha x_i, x_k).$$

Come ben noto da [5] la soluzione della (3.0) in $CT_{\Sigma^V}(X)$ è unica (considerando x_k come parametro). Si conviene di scrivere tale soluzione nella seguente maniera:

$$x_i = [A\alpha]^* x_k^{(5)}.$$

(⁵) Una simile notazione non è eccessivamente arbitraria in quanto la espressione di cui sopra può essere pensata scritta come

$$x_i = [A\alpha][A\alpha] \dots x_k$$

l'asterisco quindi ha un significato in qualche maniera analogo a quello delle espressioni regolari.

Considerando la flow-chart associata al (3.0) riportata in fig. 1,

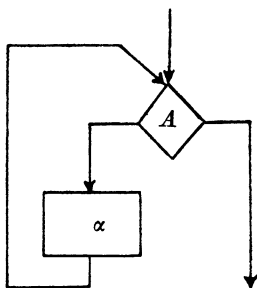


Figura 1

ci si può facilmente convincere che la notazione $[A\alpha]^*$ rappresenta la struttura DO ove un solo guarded-command è coinvolto. Tale struttura, per l'interpretazione datane da Dijkstra, corrisponde ad un do-while, il quale è effettivamente soluzione della equazione (3.1), che in $[0, 1, 2]$ è stata indicata con:

$$x_j = A^*(\alpha) x_k.$$

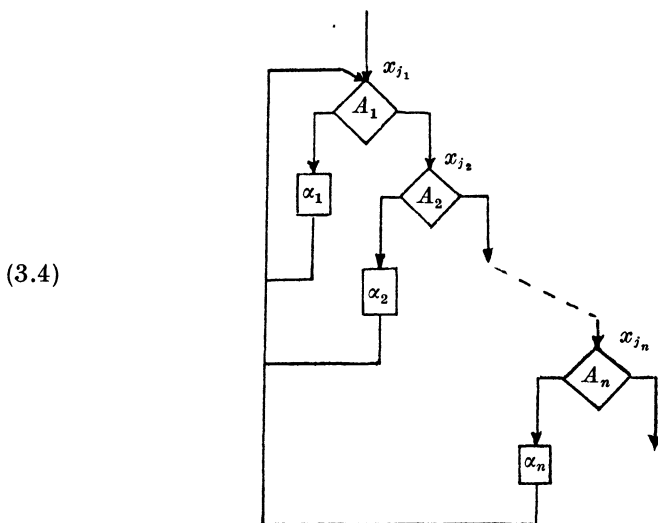
Si consideri ora il sottosistema:

$$(3.2) \quad \begin{cases} x_{j_1} = [A_1 \alpha_1 x_{j_1} + \bar{A}_1 x_{j_1}] , \\ x_{j_2} = [A_2 \alpha_2 x_{j_1} + \bar{A}_2 x_{j_2}] , \\ \vdots \\ x_{j_n} = [A_n \alpha_n x_{j_1} + \bar{A}_n x_{j_{n+1}}] , \end{cases}$$

a cui corrisponde il sottosistema scritto nell'altra notazione:

$$(3.3) \quad \begin{cases} x_{j_1} = A_1(\alpha_1 x_{j_1}, x_{j_2}), \\ x_{j_2} = A_2(\alpha_2 x_{j_1}, x_{j_2}), \\ \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \\ x_{j_n} = A_n(\alpha_n x_{j_1}, x_{j_{n+1}}), \end{cases}$$

e la sottoflow-chart seguente:



Come ben noto da [5] la soluzione è unica (considerando $x_{j_{n+1}}$ come parametro). Si conviene di scrivere la soluzione rispetto a x_{j_1} nella seguente maniera:

$$x_{j_1} = [A_1\alpha_1 + A_2\alpha_2 + \dots + A_n\alpha_n]^* x_{j_{n+1}}$$

ove la precedente scrittura indica una struttura DO di Dijkstra ove sia stata mantenuta l'ipotesi aggiuntiva I.A. Questo è facilmente verificabile confrontando la flow-chart (3.4) e la definizione del DO.

Si noti che anche il sottosistema costituito dalle prime $n-1$ equazioni del (3.2) ha la stessa struttura del (3.2) e la sua soluzione può essere scritta nella stessa maniera:

$$x_{j_1} = [A_1\alpha_1 + \dots + A_{n-1}\alpha_{n-1}]^* x_{j_n},$$

$$x_{j_n} = [A_n\alpha_n x_{j_1} + \bar{A}_n x_{j_{n+1}}],$$

da cui, sostituendo e risolvendo l'equazione al punto unito, si ottiene:

$$x_{j_1} = [A_1\alpha_1 + \dots + A_{n-1}\alpha_{n-1}]^* [A_n\alpha_n [A_1\alpha_1 + \dots + A_{n-1}\alpha_{n-1}]^*] x_{j_{n+1}}.$$

Analogamente è possibile considerare dapprima le prime $n-l$ equa-

di un sistema sia da preferire, e se eventualmente esiste un metodo per ottenerla. Un criterio possibile sarebbe quello di preferire sempre la espressione « più compatta » cioè quella ove predicati ed azioni sono ripetuti il minor numero di volte possibile, ottenuta cioè con il minor numero possibile di node-splitting. Sfortunatamente questo criterio non è applicabile in generale dato che non è sempre possibile stabilire quale tra due espressioni è quella più compatta, anche confrontandole direttamente. Inoltre soluzioni parziali che minimizzano la ripetizione di simboli danno poi origine a soluzioni che non godono più di una simile caratteristica.

Tuttavia una procedura può essere data in qualche caso particolare. Dato un sistema si segua il processo di soluzione seguente:

a) sostituire in tutte le equazioni tutte le variabili i cui membri destri dipendono solo da una variabile o da Ω ;

b) quando nessuna sostituzione è più possibile esaminare i sottosistemi del tipo

$$\begin{aligned}x_{j_1} &= [A_1\alpha_1x_{j_1} + \bar{A}_1x_{j_2}], \\x_{j_2} &= [A_2\alpha_2x_{j_1} + \bar{A}_2x_{j_2}], \\&\dots\dots\dots \\x_{j_n} &= [A_n\alpha_nx_{j_1} + \bar{A}_nx_{j_{n+1}}],\end{aligned}$$

ove la equazione di $x_{j_{n+1}}$ non è del tipo

$$x_{j_{n+1}} = [A_{n+1}\alpha_{n+1}x_{j_1} + \bar{A}_{n+1}x_{j_{n+1}}]$$

e, se x_{j_i} , $i = 2, \dots, n$ non compare in nessun membro destro delle rimanenti equazioni in modo che il sottosistema può essere risolto solamente rispetto ad x_{j_1} , rimpiazzare il sottosistema con la soluzione:

$$x_{j_1} = [A_1\alpha_1 + A_2\alpha_2 + \dots + A_n\alpha_n]^*x_{j_{n+1}}.$$

c) riprendere la procedura da capo finchè possibile.

Vale il seguente risultato di minimizzazione globale:

se la procedura a)-c) genera una soluzione, la soluzione ottenuta minimizza la duplicazione di predicati e azioni.

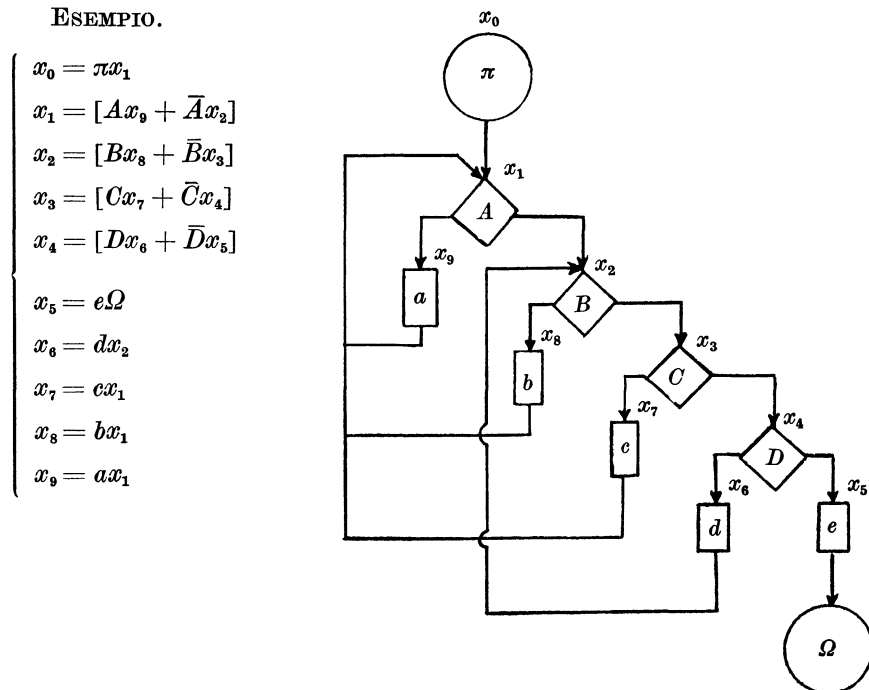
Infatti la espressione della soluzione parziale per x_{j_1} ottenuta con questa procedura è certamente, tra le tante possibili, la più compatta in quanto non vi è alcuna ripetizione di alcun simbolo. Se solamente

x_i viene chiamato dalle altre equazioni, l'averlo impostato la forma più compatta per essa assicura che la soluzione finale sarà costruita da blocchi i più compatti possibile.

Il risultato non è vero in generale in quanto, se anche qualche variabile intermedia del sottosistema compare in qualche membro destro, la massima compattezza per la espressione che riguarda la variabile di testa non assicura la massima compattezza per quel che riguarda tale variabile intermedia; il grado di compattezza della espressione finale dipende da fattori esterni al sottosistema in esame e quindi non è possibile determinare qual è l'espressione più conveniente dalla sola osservazione di questo sottosistema.

Comunque vale il seguente risultato di minimizzazione locale: quando nel procedimento di soluzione si incontra un sottosistema che soddisfa le ipotesi del punto b) è certamente conveniente l'espressione della soluzione suggerita dalla procedura in quanto si dà della sottoflow-chart corrispondente al sottosistema, che avendo una entrata e una uscita compare nella soluzione come un unico blocco, l'espressione più compatta.

ESEMPIO.



1° modo. Sostituendo x_5, x_6, x_7, x_8, x_9 e considerando il sottosistema x_1, x_2, x_3 si ottiene:

$$\begin{cases} x_0 = \pi x_1 \\ x_1 = [Aa + Bb + Ce]^* x_4 \\ x_2 = [Bb[Aa + Bb + Ce]^* x_4 + \bar{B}x_3] \\ x_3 = [Ce[Aa + Bb + Ce]^* x_4 + \bar{C}x_4] \\ x_4 = [Ddx_2 + \bar{D}e\Omega] \end{cases}$$

$$\begin{cases} x_0 = \pi[Aa + Bb + Ce]^* x_4 \\ x_4 = [Dd[Bb[Aa + Bb + Ce]^* + Ce[Aa + Bb + Ce]^* + \bar{C}I]x_4 + \bar{D}e\Omega] \end{cases}$$

$$\begin{cases} x_0 = \pi[Aa + Bb + Ce]^* x_4 \\ x_4 = [Dd[Bd[Aa + Bb + Ce]^* + Ce[Aa + Bb + Ce]^* + \bar{C}I]^* e\Omega] \end{cases}$$

2° modo. Sostituendo x_5, x_6, x_7, x_8, x_9 e considerando l'equazione per x_1 si ottiene:

$$\begin{cases} x_0 = \pi x_1 \\ x_1 = [Aa]^* x_2 \\ x_2 = [Bb[Aa]^* x_2 + Bx_3] \\ x_3 = [Ce[Aa]^* x_2 + \bar{C}x_4] \\ x_4 = [Ddx_2 + \bar{D}e\Omega] \end{cases}$$

e considerando poi il sottosistema x_2, x_3, x_4 :

$$\begin{cases} x_0 = \pi[Aa]^* x_2 \\ x_2 = [Bb[Aa]^* + Ce[Aa]^* + Dd]^* e\Omega \end{cases}$$

In questo caso il 2° modo risulta più conveniente anche se alla prima applicazione del punto unito sembrerebbe più compatta la soluzione del 1° modo.

BIBLIOGRAFIA

- [0] J. KOTT, *Remarque sur la structure des schemes de programmes*, in Automata Languages and Programming, M. NIVAT (ed.), North Holland Pub. Co., Amsterdam (1972), pp. 265-271.
- [1] A. CHIARINI - A. G. B. TEOLIS, *Mathematical Foundations for Formal Programming*, C.N.E.N. RT/EDP (76)2.
- [2] R. RIGHI - A. G. B. TEOLIS, *Program Transformations: D- and BJ_n -charts*, C.N.E.N. RT/EDP (78) 5.
- [3] E. W. DIJKSTRA, *A Discipline of Programming*, Prentice Hall, 1976.
- [4] H. F. LEDGARD - M. MARCOTTY, *A Genealogy of Control Structures*, Comm. ACM, **18**, no. 11 (1975), pp. 829-639.
- [5] J. A. GOGUEN - J. W. THATCHER - E. G. WAGNER - J. B. WRIGHT, *Initial Algebra Semantics and Continuous Algebras*, J. ACM, **24**, no. 1 (1977), pp. 68-95.

Manoscritto pervenuto in redazione il 9 gennaio 1979.