UNIVERSITE DE NANCY FACULTE DES SCIENCES

APPLICATION DE LA NOTION DE PILE A DES PROBLEMES PORTANT SUR LES CHEMINS DES GRAPHES

THESE

pour l'obtention du

DOCTORAT de SPECIALITE MATHEMATIQUES (3ème CYCLE)

Soutenue devant le Jury le 30 octobre 1965

par

Albert EMOND

Jury : Mr J. LEGRAS Président

Mr C. PAIR

Mr M. DEPAIX Examinateurs

UNIVERSITE DE NANCY FACULTE DES SCIENCES

APPLICATION DE LA NOTION DE PILE A DES PROBLEMES PORTANT SUR LES CHEMINS DES GRAPHES

° ° °

pour l'obtention du

DOCTORAT de SPECIALITE MATHEMATIQUES (3ème CYCLE)

Soutenue devant le Jury le 30 octobre 1965

par

Albert EMOND

° ° °

Jury : Mr J. LEGRAS Président

Mr C. PAIR

Mr M. DEPAIX Examinateurs

NIVERSITE DE NANCY

FACULTE DES SCIENCES

Doyen : M. AUBRY

Assesseur : M. GAY

oyens honoraires : MM. CORNUBERT - DELSARTE - URION - ROUBAULT.

rofesseurs honoraires : MM. CROZE-RAYBAUD-LAFITTE-LERAY-JOLY-LAPORTE-EICHHORN-GODEMENT-DUBREIL-SCHWARTZ-DIEUDONNE-DE MALLEMANN-LONGCHAMBON-LETORT-DODE-GAUTHIER-GOUDET-ULMER-CORNUBERT-HAPELLE-GUERIN-CHEVALLIER-WAHL.

aîtres de conférences honoraires : MM. LIENHART - PIERRET.

Professeurs

M.URION	Chimie bio.	FRUHLING	Physique	GARNIER	Agronomie
DELSARTE	Analyse sup.	SUHNER	Physique expérim.	*WEPPE	Minéralogie Appliquée
ROUBAULT	Géologie	HILLY	Géologie	BERNARD	Géologie appliquée
VEILLET	Biologie animale	LE GOFF	Génie chimique	*CHAMPIER	Physique
BARRIOL	Chimie théorique	CHAPON	Chimie biologique	*REGNIER	Physico-chimie
BIZETTE	Physique	HEROLD	Chimie miné. ind.	*GAY	Chimie biologique
GUILLIEN	Electronique	SCHWARTZ	Exploit. minière	*WERNER	Botanique
GIBERT	Chimie Physique	GAYET	Physio. animale	*GONDE	Zoologie
LEGRAS	Mécanique Ration.	*MALAPRADE	Chimie	STEPHAN	Zoologie
BOLLY	Minéralogie	HADNI	Physique	EYMARD	Calcul dif. et intégr.
NICLAUSE	Chimie	BONVALET	Mécan. Appliquée	LEVISALLES	Chimie organique
FAVRE	Physique appl.	*KERN	Minéralogie	CAPELLE	Mécanique rationnelle
AUBRY	Chimie minérale	*BASTICK	Chimie	MANGENOT	Botanique
DUVAL	Chimie	DUCHAUFOUR	Pédologie	Mme HERVE	Méthodes math. physiqu
COPPENS	Radiogéologie	NEEL	Chimie org. indus.	BASTICK	Chimie M.P.C. Epinal

Maîtres de conférences

M.COSSE	Mécanique physique	BLAZY	Minéralogie appliquée (ENSG)
ROCCI	Géologie	BALESDENT	Thermodynamique chimique appliquée
VUILLAUME	Psychophysiologie	JANOT	Physique M.P.C. (Epinal)
GUDEFIN	Physique	JACQUIN	Pédologie et chimie agricoles
HORN	Physique propédeutique	CACHAN	Entomologie appliquée (ENSA)
FRENTZ	Biologie animale	N...	Probabilités et Statistiques
AUROUZE	Géologie	N...	Mécanique (I.S.I.N.)
MARI	Chimie (I.S.I.N.)	N...	Physique M.G.P.
LAFON	Physique (I.S.I.N.)	N...	Mathématiques
FELDEN	Physique théorique et nucl.	N...	Mécanique expérimentale
FLECHON	Physique M.P.C.	N...	Génie chimique
VIGNES	Métallurgie	N...	Mathématiques propédeutiques
HUET	Mathématiques S.P.C.N.	N...	Phytopathologie
DEVIOT	Physique du solide	N...	Mathématiques S.P.C.N.

Je tiens à exprimer ma profonde gratitude à Monsieur le Professeur J. LEGRAS, Directeur du Centre de Calcul Automatique de Nancy, pour la bienveillante sollicitude qu'il n'a cessé de me témoigner au cours de ces deux années.

Je suis heureux d'adresser mes plus vifs remerciements à Monsieur C. PAIR, pour l'attention continuelle avec laquelle il a suivi et dirigé ce travail.

Je remercie Messieurs les Professeurs qui ont bien voulu accepter de former le jury.

Mes remerciements vont également à Mademoiselle C. COURTOIS, qui m'a fait aimablement profiter de son expérience "Gamma 60".

Je remercie enfin Mesdames les Secrétaires du Centre de Calcul pour l'empressement avec lequel elles m'ont toujours rendu service.

SOMMAIRE

NOTA

Dans les organigrammes, les tests sont encadrés de la façon suivante :

Si la relation est vraie, le branchement se fait dans le sens de la flèche ; si la relation est fausse, le branchement se fait vers le bas.

CHAPITRE I

DEFINITIONS

Dans les définitions qui suivent, nous adopterons les notations suivantes : lettres d'imprimerie majuscules pour les relations, minulcules pour les éléments et les termes des matrices ; lettres rondes majuscules pour les matrices.

D. 1. Matrice associée à une relation binaire dans un ensemble.

Soit une relation binaire R définie sur l'ensemble E. La matrice associée à la relation R (on dit aussi matrice de la relation) est une matrice carrée booléenne $\mathcal{R}$ dont le terme r_j^i vaut 1 si, et seulement si, $e_i R e_j$, 0 sinon (e_i, $e_j \in E$).

D. 2. Réunion et produit de deux relations binaires.

Soient R' et R'' deux relations binaires définies sur E, x et y deux éléments de E. On appelle réunion de R' et R'', et on note $R' \vee R''$ la relation binaire R : xRy si, et seulement si, $xR'y$ ou $xR''y$.

On appelle produit de R' et R'', et on note $R' \wedge R''$ la relation binaire P : xPy si, et seulement si, il existe un élément z de E tel que $xR'z$ et $zR''y$.

T. 2. Les matrices $\mathcal{R}$ et $\mathcal{P}$ des relations binaires R' et P' sont respectivement égales à la somme et au produit booléens * des matrices $\mathcal{R}'$ et $\mathcal{R}''$ associées à R' et R''.

Dans ce qui suit, somme et produit booléens de matrices sont respectivement notés $+$ et $\times$.

---

* La somme et le produit booléens sont définis à partir de la somme $\vee$ et du produit $\wedge$ comme la somme et le produit de deux matrices réelles à partir des opérations $+$ et $\times$.

D. 3. Puissance d'une relation binaire.

Soit R une relation binaire. Sa puissance $n^{ième}$ est une relation binaire P^n définie par :

$$P^n = P \wedge P \ldots P \wedge P \quad (n \text{ facteurs})$$

T. 3. La matrice $\mathcal{S}_n$ de la relation P^n est égale à la puissance booléenne $n^{ième}$ de la matrice $\mathcal{R}$ associée à R.

D. 4. Fermeture transitive d'une relation binaire.

Soit Γ une relation binaire. On appelle fermeture transitive de Γ une relation T' réunion de toutes les puissances Γ^n de Γ $(n = 0, 1, \ldots)$, Γ^0 étant la relation d'égalité :

$$T' = \Gamma^0 \vee \Gamma^1 \vee \Gamma^2 \vee \ldots$$

T. 4. La matrice associée à la relation de fermeture transitive de Γ est la matrice $\mathcal{C}' = I + \gamma + \gamma^2 + \ldots$

$\mathcal{C}'$ est aussi appelée fermeture transitive de la matrice Γ.

D. 5. Fermeture transitive stricte d'une relation binaire.

C'est la relation T définie par :

$$T = \Gamma \vee \Gamma^2 \vee \Gamma^3 \vee \ldots$$

T. 5. La matrice associée à la relation de fermeture transitive stricte de Γ est la matrice : $\mathcal{C} = \gamma + \gamma^2 + \gamma^3 + \ldots$

D. 6. Graphe.

On appelle graphe un couple (E, Γ) où E est un ensemble de points et Γ une relation binaire définie sur E.

L'ensemble des points x tels que $y \Gamma x$ sera appelé famille de y et noté $\Gamma(y)$.

D. 7. Chemins, circuits d'un graphe.

Soit un graphe (E, Γ). On appelle chemin une suite $(e_0, e_1, \ldots, e_n)$ de points du graphe telle que $n \geqslant 1$ et que $e_{i-1} \Gamma e_i$ pour $i = 1, 2, \ldots, n$. n est appelé longueur du chemin. e_0 est l'origine et e_n l'extrémité du chemin. Un chemin de longueur 1 est appelé arc.

Un chemin qui ne contient pas deux fois le même point est dit élémentaire. On appelle circuit un chemin dont l'extrémité coïncide avec l'origine.

Un circuit qui ne contient pas deux fois le même point (à l'exclusion de l'origine extrémité) est dit élémentaire.

D. 8. Matrice associée à un graphe [BERGE].

T. 8. La matrice associée à un graphe (E, Γ) est égale à la matrice associée à la relation Γ du graphe.

D. 9. Préordre associé au graphe.

C'est une relation de préordre Q telle que si x, y sont des points du graphe, $x Q y$ si, et seulement si, $x = y$ ou il existe un chemin d'origine x et d'extrémité y.

T. 9. La relation de préordre associée à un graphe (E, Γ) est la relation de fermeture transitive de Γ.

Dans ce qui suit la relation de préordre ou fermeture transitive sera notée R'.

D. 10. Relation d'équivalence associée au graphe.

C'est l'équivalence définie par le préordre R' associé au graphe. On note $x \sim y \iff x R' y$ et $y R' x$.

T. 10. x et y sont équivalents si, et seulement si, x égale y, ou il existe un circuit contenant les points x et y.

D. 11. Pile, entrées et sorties [PAIR].

Soit un ensemble E. On appelle pile sur E toute suite finie $U = (u_0, u_1, \ldots, u_n)$ de mots sur l'ensemble E telle que :

- $u_0 = u_n = \Lambda$ (mot vide)
- pour $i = 1, 2, \ldots, n$ on a soit $u_i = u_{i-1} e$
  soit $u_{i-1} = u_i e$

u_i est appelé état de la pile à l'instant i. Le dernier élément du mot u_i est appelé sommet de la pile à l'instant i (on dit aussi sommet de u_i).

On appelle entrée de e et on note $\mathcal{E}(e)$ l'instant i tel que $u_i = u_{i-1}e$. On appelle sortie de e et on note $\sigma(e)$ l'instant i tel que $u_{i-1} = u_i e$.

Si tout élément entre une fois et une seule dans la pile, celle-ci est dite pile simple. Il en résulte que tout élément sort une fois et une seule.

D. 12. Ordre associé à une pile simple.

C'est une relation N définie sur l'ensemble E : $x N y$ si, et seulement si, il existe un instant i tel que y est au sommet de la pile et $x \in u_i$.

D. 13. Pile attachée à un graphe.

C'est une pile simple sur l'ensemble E des points du graphe telle que l'on passe de l'état u_{i-1} à l'état u_i (pour $i > 0$) par le processus suivant :

a) $u_{i-1} = \Lambda$

- s'il existe un élément $z \in E$ n'ayant pas d'entrée $< i$ alors i est l'entrée de z ;
- s'il n'existe pas d'élément $z \in E$ n'ayant pas d'entrée $< i$ alors $u_i = u_n$.

b) $u_{i-1} \neq \Lambda$ et a pour sommet x.

- s'il existe un élément $z \in \Gamma(x)$ n'ayant pas d'entrée $< i$ alors i est l'entrée de z ;
- s'il n'existe pas d'élément $z \in \Gamma(x)$ n'ayant pas d'entrée $< i$ alors i est la sortie de x.

Les éléments n'ayant qu'une entrée et une sortie dans la pile attachée, on peut leur associer un ordre d'entrée P : $x P y$ si, et seulement si, $\mathcal{E}(x) \leq \mathcal{E}(y)$, et un ordre de sortie S : $x S y$ si, et seulement si, $\sigma(x) \leq \sigma(y)$.

CHAPITRE II

DETERMINATION DE LA FERMETURE TRANSITIVE STRICTE DU GRAPHE

On se propose dans ce chapitre de déterminer la fermeture transitive stricte du graphe à l'aide d'une pile annexe V dont seuls certains états seront identiques à ceux de la pile attachée.

On notera Γ^+ la relation du graphe, R sa relation de fermeture transitive stricte, u_i l'état de la pile attachée à l'instant i, N l'ordre associé à la pile U, P et S l'ordre d'entrée et de sortie des éléments dans la pile U. Parallèlement, v_i sera l'état de la pile annexe à l'instant i.

Nous ferons d'abord le choix d'une pile attachée (celui de la pile annexe en découlant), nous exposerons et justifierons ensuite la méthode dans le cas du graphe sans circuit, puis dans le cas général.

Choix de la pile attachée.

Les piles attachées se diférencient les unes des autres par l'ordre dans lequel les éléments entrent dans la pile. Le graphe étant mémorisé sous la forme de sa matrice associée, les éléments sont codés implicitement. C'est dans cet ordre que se feront les entrées dans la pile.

Pile annexe à un graphe.

C'est une pile V sur l'ensemble E des points du graphe telle que l'on passe de l'état v_{i-1} à l'état v_i par le processus suivant :

a) $v_{i-1} = \Lambda$

alors i est l'entrée du premier $z \in E$ n'ayant pas d'entrée inférieure à i s'il existe. Sinon, v_{i-1} est l'état final de V.

b) $v_{i-1} \neq \Lambda$ et a pour sommet x.

1) i-1 est une sortie de y.

si y n'est pas le dernier élément de $\Gamma(x)$ alors i est une entrée de l'élément z qui suit y dans $\Gamma(x)$,

si y est le dernier élément de $\Gamma(x)$ alors i est une sortie de x.

2) i-1 est la première entrée de x.

si $\Gamma(x) \neq \emptyset$ alors i est une entrée du premier élément z de $\Gamma(x)$

si $\Gamma(x) = \emptyset$ alors i est une sortie de x.

3) i-1 est une autre entrée de x,

alors i est une sortie de x.

Entrée et sortie vraies de la pile annexe.

On appelle entrée vraie de x et on note $\Delta(x)$ la première entrée de x dans la pile V.

On appelle sortie vraie de x et on note $\Omega(x)$ l'instant i tel que $v_{\Delta(x)} = v_{i-1}$ et i est une sortie de x.

Propriété de la pile annexe.

Nous allons montrer que l'ordre d'entrée vraie dans V (resp. de sortie vraie de V) est l'ordre P d'entrée dans U (resp. de sortie de U) par une récurrence dont l'hypothèse est la suivante :

$u_i = v_{f(i)}$ (1) et pour $i > 0$: f(1), f(2) ... f(i) (2) est la suite croissante des entrées et sorties vraies inférieures ou égales à f(i).

Posons f(0) = 0 : $u_0 = v_{f(0)} = \Lambda$. L'hypothèse est vraie à l'instant i = 0. Montrons que si elle est vraie pour i-1 elle l'est pour i.

a) $u_{i-1} = \Lambda$. D'après (1) $v_{f(i-1)} = \Lambda$.

- Tous les éléments ont une entrée dans U inférieure à i :

u_{i-1} est l'état final de U. D'après (2), tous les éléments ont une entrée vraie dans V inférieure ou égale à f(i-1) :

f(i-1) est l'état final de V.

- i est l'entrée dans U du premier z n'ayant pas d'entrée inférieure à i. Tout z' (s'il existe) précédant z possède une entrée h dans U inférieure à i. D'après (2) : f(h) est inférieur ou égal à f(i-1) et f(h) est une entrée vraie de z'. z est donc le premier élément n'ayant pas d'entrée vraie dans V inférieure ou égale à f(i-1), et f(i-1)+1 est son entrée vraie :

$$f(i) = f(i-1)+1,\ u_i = v_{f(i)}\ .$$

b) $u_{i-1} \neq \Lambda$. Soit x le sommet de u_{i-1}. D'après (1) x est sommet de $v_{f(i-1)}$.

1) i-1 est la sortie de y, i est la sortie de x de U.

- Supposons qu'il existe p éléments z'_j $(1 \leqslant j \leqslant p)$ suivant y dans $\Gamma(x)$. Tout z'_j possède dans U une entrée h_j inférieure à i. D'après (2) tout z'_j possède dans V une entrée vraie $f(h_j)$ inférieure ou égale à f(i-1). On a donc :

f(i-1)+1 = entrée de z'_1, f(i-1)+2 = sortie de z'_1 autre que la sortie vraie

.....

f(i-1)+2p-1 = entrée de z'_p, f(i-1)+2p = sortie de z'_p autre que la sortie vraie

f(i-1)+2p+1 = sortie vraie de x :

$$f(i) = f(i-1)+2p+1,\ u_i = v_{f(i)} \quad (p \text{ entier positif})$$

- S'il n'existe aucun z'_j, alors f(i-1)+1 est la sortie vraie de x :

$$f(i) = f(i-1)+1,\ u_i = v_{f(i)}\ .$$

2) i-1 est la sortie de y, i est l'entrée de z dans U.

- Supposons qu'il existe p éléments z'_j $(1 \leqslant j \leqslant p)$ compris entre y et z dans $\Gamma(x)$. Tout z'_j possède dans U une entrée h_j inférieure à i. D'après (2) tout z'_j possède dans V une entrée vraie $f(h_j)$ inférieure ou égale à f(i-1). Le même raisonnement qui ci-dessus nous conduit à :

f(i-1)+2p+1 est l'entrée vraie de z :

$$f(i) = f(i-1)+2p+1,\ u_i = v_{f(i)} \quad (p \text{ entier positif}).$$

- S'il n'existe aucun z'_j, alors

f(i-1)+1 est l'entrée vraie de z :

$$f(i) = f(i-1)+1,\ u_i = v_{f(i)}\ .$$

3) i-1 est l'entrée de x, i est la sortie de x de U.

- $\Gamma(x) \neq 0$. Soient z'_j $(1 \leqslant j \leqslant p)$ les p éléments de $\Gamma(x)$.

Un raisonnement analogue conduit à :

f(i-1)+2p+1 est la sortie vraie de x.

$f(i) = f(i-1)+2p+1$, $u_i = v_{f(i)}$ (p entier positif).

- $\Gamma(x) = 0$: f(i-1)+1 est la sortie vraie de x.

$f(i) = f(i-1)+1$, $u_i = v_{f(i)}$.

4) i-1 est l'entrée de x, i est l'entrée de z dans U.

- Soient z'_j $(1 \leqslant j \leqslant p)$ les p éléments précédant z dans $\Gamma(x)$ s'ils existent.

Le même raisonnement entraîne :

f(i-1)+2p+1 est l'entrée vraie de z d'où :

$f(i) = f(i-1)+2p+1$, $u_i = v_{f(i)}$ (p entier positif).

- S'il n'existe aucun z'_j, alors

f(i-1)+1 est l'entrée vraie de z :

$f(i) = f(i-1)+1$, $u_i = v_{f(i)}$.

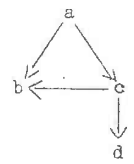
Par récurrence, si n est l'état final de la pile U :

$u_i = v_{f(i)}$, $0 \leqslant i \leqslant n$ et $f(0), f(1), \ldots, f(n)$ est la suite des entrées et sorties vraies de la pile V.

Théorème.

L'ordre P d'entrée dans U (resp. S de sortie de U) est l'ordre d'entrée vraie dans V (resp. de sortie vraie de V).

Exemple.



Soit le graphe ci-contre. Nous donnons ci-dessous les différents états de la pile annexe et de la pile attachée. On suppose que les éléments sont rangés dans l'ordre alphabétique.

						b		d		
Pile annexe		b		c	c	c	c	c		
	a	a	a	a	a	a	a	a	a	a
Instant	0	1	2	3	4	5	6	7	8	9	10

					d				
Pile attachée		b		c	c	c			
	a	a	a	a	a	a	a	a	
Instant	0	1	2	3	4	5	6	7	8

Nous donnons ci-dessous les instants d'entrées et de sortie vraies des éléments.

$\Delta(a) = 1$	$\Omega(a) = 10$
$\Delta(b) = 2$	$\Omega(b) = 3$
$\Delta(c) = 4$	$\Omega(c) = 9$
$\Delta(d) = 7$	$\Omega(d) = 8$

Ordre associé à la pile annexe.

Par définition, ce sera l'ordre N associé à la pile attachée.

CAS DU GRAPHE SANS CIRCUIT.

Méthode.

A tout point x du graphe, on associe un ensemble K(x,j) tel que à l'instant 0 on ait $K(x,0) = \{ z \text{ tel que } x \Gamma z \}$ et qu'à l'instant de sortie vraie de x, on ait $K(x, \Omega(x)) = \{ z \text{ tel que } x R z \}$.

Pour cela, on passe de K(x,j-1) à K(x,j) par le processus suivant :

- x est le sommet de v_j

si j est une sortie de z alors :

$K(x,j) = K(x,j-1) \cup K(z,j-1)$ (α)

si j est une entrée

$K(x,j) = K(x,j-1)$ (β)

- x n'est pas le sommet de v_j

$K(x,j) = K(x,j-1)$ (β)

Justification.

1) Montrons par une récurrence sur l'instant j que $y \in K(x,j)$ entraîne $x R y$.

Supposons qu'à l'instant j-1, on ait $y \in K(x,j-1)$ entraîne $x R y$ pour tout x. Alors à l'instant j, on a :

soit $K(x,j) = K(x,j-1)$ (β) d'où

$y \in K(x,j)$ entraîne $y \in (x,j-1)$ entraîne $x R y$;

soit $K(x,j) = K(x,j-1) \cup K(z,j-1)$ (α) avec $x \Gamma z$ d'où

$y \in K(x,j)$ entraîne $y \in K(x,j-1) \Longrightarrow x R y$

ou $y \in K(z,j-1) \Longrightarrow z R y \Longrightarrow x R y$.

Or, à l'instant 0, $y \in K(x,0) \Longrightarrow x \Gamma y \Longrightarrow x R y$.

2) Montrons que $x R y$ entraîne $y \in K(x, \Omega(x))$.

Si $x R y$, alors il existe un chemin élémentaire $x \Gamma x_1 \ldots \Gamma x_n = y$. Comme $x_{i-1} \Gamma x_i$, il existe une entrée h de x_i inférieure à la sortie vraie de x_{i-1}. Supposons que la sortie vraie de x_{i-1} soit inférieure ou égale à la sortie vraie de x_i : $\Delta(x_i) \leqslant h < \Omega(x_{i-1}) \leqslant \Omega(x_i)$. On a donc $x_i N x_{i-1} \Gamma x_i$ d'où $x_{i-1} \sim x_i$ ou $x_{i-1} = x_i$, ce qui est contraire à l'hypothèse graphe sans circuit, ou chemin élémentaire. La sortie vraie de x_i est donc inférieure à la sortie vraie de x_{i-1}. D'autre part, à l'instant de la sortie de x_i, on a (α).

$$K(x_i, \Omega(x_i)) \subseteq K(x_{i-1}, \Omega(x_i)) \subseteq K(x_{i-1}, \Omega(x_{i-1}))$$

or $y \in K(x_{n-1}, 0)$ entraîne $y \in K(x_{n-1}, \Omega(x_{n-1}))$ d'où : xRy entraîne $y \in K(x, \Omega(x))$.

CAS GENERAL.

La différence essentielle entre cas général et cas du graphe sans circuit réside dans le fait que $x \Gamma y$ n'entraîne pas forcément $\Omega(y) < \Omega(x)$. Pour rétablir une propriété analogue, nous allons définir un élément de classe x tel que $x \sim y \Longrightarrow \Omega(y) < \Omega(x)$.

Element premier de classe.

C'est le premier dans l'ordre P des éléments appartenant à une même classe. D'après la propriété de la pile annexe si x est premier de classe et $y \sim x$, on a : $\Delta(x) <$ toute entrée de y.

Pour démontrer que x premier de classe et $y \sim x$ entraîne $\Omega(y) < \Omega(x)$, nous allons d'abord établir le lemme suivant :

Lemme.

$x = x_0 \Gamma x_1 \ldots \Gamma x_n$ et $x P x_i$ pour $i = 0,1,2 \ldots n$ entraîne $x_n S x$.

Soit $j = \Delta(x)$ et $k = \Omega(x)$. $x P x_i$ entraîne $x_i \notin u_j$ et $u_k x = u_j$ entraîne $x_i \notin u_k$. Comme $x_i \Gamma x_{i+1}$, il existe une entrée h de x_{i+1} inférieure à la sortie vraie de x_i : $\Delta(x_{i+1}) \leqslant h < \Omega(x_i)$.

Faisons une récurrence sur i :

Supposons $x_i S x_0$. Alors $\Delta(x_{i+1}) \leqslant k$ et $x_{i+1} \notin u_k$ entraîne $x_{i+1} S x_0$.

Or $x_0 S x_0$ par récurrence $x_n S x_0$.

Théorème 14.

Si x est premier de classe et si y est équivalent à x : $y S x$.

$y \sim x$ entraîne il existe un chemin $x \Gamma x_1 \ldots \Gamma x_n = y$ avec $x_i \sim x$ pour tout i. Or, x premier de classe entraîne $x P x_i$ pour tout i. D'après le lemme : $y S x$.

Méthode.

A tout point x du graphe, on associe un ensemble K(x,j) tel qu'à l'instant 0 on ait $K(x,0) = \{z \text{ tel que } x \Gamma z\}$ et qu'à l'instant final m on ait $K(x,m) = \{z \text{ tel que } x R z\}$. Pour cela, on passe de K(x,j-1) à K(x,j) par le processus suivant :

- j est une sortie vraie de z_0 premier de classe.

  $K(z_k,j) = \bigcup_k K(z_k,j-1)$ pour tout k tel que $z_k \sim z_0$ (α)

  si x est sommet de v_j

  $K(x,j) = K(x,j-1) \cup K(z_0,j)$ (β)

  $K(y,j) = K(y,j-1)$ pour tout $y \neq x$ et $y \not\sim z_0$ (γ)

- j est une sortie de z supérieure à la sortie du premier de sa classe.

  si x est sommet de v_j

  $K(x,j) = K(x,j-1) \cup K(z,j-1)$ (δ)

  $K(y,j) = K(y,j-1)$ pour tout $y \neq x$ (ε)

- j est une entrée ou une sortie de z inférieure à la sortie du premier de sa classe.

  $K(y,j) = K(y,j-1)$ (μ)

Justification.

1) Une récurrence sur l'instant j va nous permettre d'établir :

K(x,h) contient y entraîne x R y (1).

Supposons que (1) soit vrai à l'instant j-1.

- Si $K(x,j) = K(x,j-1)$ (opérations γ, ξ, μ de la méthode générale)

  $K(x,j)$ contient y entraîne

  $K(x,j-1)$ contient y d'où x R y.

- Si $K(z_k,j) = \bigcup_k K(z_k,j-1)$ (α)

  $K(z_k,j)$ contient y entraîne $K(z_i,j-1)$ contient y d'où $z_i R y$. Comme z_k est équivalent à z_i, on a $x R y$.

- Si $K(x,j) = K(x,j-1) \cup K(z_0,j)$ (β)

  $y \in K(x,j) \Longrightarrow$ { $K(x,j-1)$ contient y $\Longrightarrow x R y$
  ou $K(z_0,j)$ contient y $\Longrightarrow z_0 R y$ or $x \Gamma z_0$ d'où $x R$

- Si $K(x,j) = K(x,j-1) \cup K(z,j-1)$ (δ)

  $y \in K(x,j) \Longrightarrow$ { $K(x,j-1)$ contient y $\Longrightarrow x R y$
  ou $K(z,j-1)$ contient y $\Longrightarrow z R y$ or $x \Gamma z$ d'où $x R y$.

  A l'instant 0, $K(x,0)$ contient y entraîne $x \Gamma y$ d'où $x R y$.

  Par récurrence :

  $\boxed{K(x,h) \text{ contient y entraîne } x R y}$

2) Montrons que x R y et p premier de la classe de x entraîne $K(x, \Omega(p))$ contient y.

Si y appartient à $K(p, \Omega(p))$ alors (α) entraîne $K(x, \Omega(p))$ contient y.

Il suffit donc de montrer que si x R y et x premier de classe alors :

$K(x, \Omega(x))$ contient y (1).

a) $x \sim y$

Il existe un chemin $x \Gamma x_1 \ldots \Gamma x_q = y$ avec $x_i \sim x$.

D'autre part, $y \in K(x_{q-1}, 0)$. A l'instant $\Omega(x)$, (α) entraîne $y \in K(x, \Omega(x))$.

b) $x \not\sim y$

Il existe un chemin $x \Gamma x_1 \ldots \Gamma x_q \Gamma z \; R \; y$, $x_q \not\sim z$.

Comme $x_q \Gamma z$, $x_q \not\sim z$, il existe une entrée h et une sortie k de z inférieures à $\Omega(x_q)$ et x_q est le sommet de v_k. Soit z' le premier de la classe de z. Montrons que $k \geq \Omega(z')$. S'il n'en était pas ainsi, on aurait les inégalités suivantes :

$\Delta(z') \leq \Delta(z) \leq h < k < \Omega(z')$.

A l'instant k, $z' \in v_k$ alors : $z' R x_q \Gamma z$ d'où $z \sim x_q$, ce qui est contraire à l'hypothèse. On a donc $k \geq \Omega(z')$ (2).

Faisons ensuite une récurrence sur l'ordre de sortie des éléments premiers de classe :

Pour cela, supposons que pour tout z' premier de classe tel que $\Omega(z') < \Omega(x)$ on ait : $K(z', \Omega(z'))$ contient y si $z' R y$.

A l'instant k de sortie de z ($k \geq \Omega(z')$ d'après (2)) :

$K(v,k)$ est inclus dans $K(x_q,k)$.

A l'instant $\Omega(x) > k$, on a :

$K(x_q,k) \subseteq K(x_q, \Omega(x)) \subseteq K(x, \Omega(x))$.

D'après l'hypothèse de récurrence :

$K(x, \Omega(x))$ contient y.

Problèmes posés par la méthode.

Ils sont au nombre de deux :

1) Savoir lorsqu'un élément sort (sortie vraie) s'il est premier de classe ;

2) Connaître tous les éléments équivalents au premier de classe lors de la sortie vraie de celui-ci.

Pour résoudre ces problèmes, nous allons définir une fonction $\ell(y,i)$ appelée lien de l'élément y à l'instant i, nous démontrerons ensuite deux théorèmes, puis nous exposerons et justifierons la méthode employée.

Fonction lien.

Le lien d'un élément y est un élément $\ell(y,i)$ tel que :

a) $\ell(y,i) \sim y$;

b) Pour i compris entre $\Delta(y)$ et $\Omega(x)$, x premier de la classe de y : $\ell(y,i) \in v_i$;

c) Si N est l'ordre associé à la pile U : $\ell(y,i) \; N \; \ell(y,i-1)$.

Existence de la fonction lien.

Soit la fonction $\ell(y,i)$ telle que : $\ell(y,0) = y$.

Si i est sortie vraie de z non premier de classe et z' est le sommet de v_i, alors on a :

$\ell(y,i) = z'$ pour tout y tel que $\ell(y,i-1) = z$ (1)

sinon $\ell(y,i) = \ell(y,i-1)$. (2)

- Montrons par une récurrence sur i que l'on a $\ell(y,i) \sim y$. Pour cela, supposons que l'on ait $\ell(y,i-1) \sim y$. A l'instant i on a soit (1), ce qui entraîne $z \sim y$, comme d'autre part $z' \sqcap z$ et i est une sortie de z non premier de classe, le théorème 14 entraîne : x premier de la classe de z appartient à v_i. On a donc : $x\ N\ z' \sqcap z$ ce qui entraîne $z' \sim z$, or, par hypothèse, $z \sim y$, d'où $\ell(y,i) = z' \sim y$.

Si on a (2), l'hypothèse entraîne $\ell(y,i) \sim y$.

Or, à l'instant 0, $\ell(y,0) = y$. Par récurrence :

$\ell(y,i) \sim y$ pour tout i.

- Montrons par une récurrence sur l'instant i compris entre $\Delta(y)$ et $\Omega(x)$ que v_i contient $\ell(y,i)$.

Pour cela, supposons que $\ell(y,h) \in v_h$ pour $\Delta(y) < h < i$ et montrons que $\ell(y,i) \in v_i$ ou $i = \Omega(x)$.

* si i est une entrée :

$\ell(y,i) = \ell(y,i-1)$. D'après l'hypothèse, i étant une entrée $\ell(y,i) \in v_i$.

* si i est une sortie vraie de z non premier de classe et $\ell(y,i-1) \neq z$, alors $\ell(y,i) = \ell(y,i-1)$. D'après l'hypothèse i n'étant pas sortie de $\ell(y,i-1)$: v_i contient $\ell(y,i-1)$.

* si i est une sortie vraie de z non premier de classe et $\ell(y,i-1) = z$, alors $\ell(y,i)$ est le sommet de v_i : v_i contient $\ell(y,i-1)$.

* si i est une sortie vraie de z premier de classe

  ⊠ $z = x$ alors $i = \Omega(x)$

  ⊠ $z \neq x$ alors z n'étant pas équivalent à y n'est le lien d'aucun y. $\ell(y,i) = \ell(y,i-1)$. D'après l'hypothèse v_i contient $\ell(y,i)$.

* i est une sortie de z autre que la première. D'après la propriété de la pile annexe, i-1 est une entrée de z d'où $u_{i-2} = u_i$. D'autre part, $\ell(y,i-2) = \ell(y,i-1) = \ell(y,i)$ d'après l'hypothèse v_i contient $\ell(y,i)$.

A l'instant $\Delta(y)$, $\ell(y,\Delta(y)) = y$ d'où $\ell(y, \Delta(y)) \in v_{\Delta(y)}$

par récurrence v_i contient $\ell(y,i)$ pour tout i compris entre $\Delta(y)$ et $\Omega(x)$.

- Montrons que l'on a $\ell(y,i)\ N\ \ell(y,i-1)$.

* si $\ell(y,i) = \ell(y,i-1)$, cela est évident.

* si $\ell(y,i) = z' \neq \ell(y,i-1) = z$, alors i est sortie vraie de z et z' est sommet de v_i. A l'instant i-1, $z' \in v_{i-1}$ et z est sommet de v_{i-1}. On a donc : $z'\ N\ z$ ou $\ell(y,i)\ N\ \ell(y,i-1)$.

La fonction $\ell(y,i)$ est bien une fonction lien.

Théorème 15.

Si y n'est pas premier de classe, il existe $z \in E$ et $z' \in E$ tels que : $y\ N\ z \sqcap z'$, $z'\ P\ y\ P\ z$, $z' \sim y$ et $z' \neq y$.

Soit x l'élément premier de la classe de y. $y \sim x$ entraîne qu'il existe un chemin élémentaire $y \sqcap x_1 \ldots \sqcap x_n = x$ avec $x_i \sim y$ pour tout i. Soit x_k le premier des x_i tels que $x_i\ P\ y$. On a alors : $y\ P\ x_i$ pour $i = 1,2,\ldots,k-1$ et $y \sqcap x_1 \ldots \sqcap x_{k-1}$, d'après le lemme : $y\ N\ x_{k-1}$.

Posons $x_k = z'$ et $x_{k-1} = z$, on a bien :

$y\ N\ z \sqcap z'$, $z'\ P\ y\ P\ z$ avec $z' \sim y$ et $z' \neq y$.

Théorème 16.

y non premier de classe $\Longleftrightarrow$ il existe $z' \in E$, un instant k et α, β, γ mots sur E tels que :

$v_k = \alpha\ \ell(z',k)\ \beta\ y\ \gamma\ z'$.

1) Montrons que $v_k = \alpha\ \ell(z',k)\ \beta\ y\ \gamma\ z'$ entraîne y n'est pas premier de classe, en effet $\ell(z',k) \sim z'(a)$ entraîne $y \sim \ell(z',k)$ or, $\ell(z',k)\ P\ y$ entraîne y n'est pas premier de classe.

2) Montrons que y non premier de classe entraîne il existe z' et k tels que $v_k = \alpha\ \ell(z',k)\ \beta\ y\ \gamma\ z'$. En effet, le théorème 15 entraîne il existe

z et z' tel que $y N z \Gamma z'$ et $z' P y P z$ avec z' différent de y et $z' \sim y$, d'où $\Delta(z') < \Delta(y) < \Delta(z)$. D'autre part $z \Gamma z'$ entraîne il existe un instant k tel que z est sommet de v_{k-1} et k est une entrée de z'. D'où $\Delta(z) < k < \Omega(z)$. En définitive, si on appelle x le premier de la classe de y, on a :

$$\Delta(z') < \Delta(y) < \Delta(z) < k < \Omega(z) \leqslant \Omega(x).$$

D'après la propriété (b) de la fonction lien, on a : $\ell(z', \Delta(y)) \in v_{\Delta(y)}$ et $\ell(z', k) \in v_k$. D'après la propriété (c) de la fonction lien $k > \Delta(y)$ entraîne $\ell(z', k) N \ell(z', \Delta(y))$ d'où $\ell(z', k) N y$. On a donc :

$$v_k = \alpha\ \ell(z', k)\ \beta\ y\ \gamma\ z'.$$

Détermination des éléments équivalents.

On repère les éléments non premiers de classe par le signe $\sim$. Pour cela, si k est une entrée de z avec $\ell(z,k) \in v_k$, alors on met le signe $\sim$ aux éléments y de la pile compris entre $\ell(z,k)$ et z.

Justification.

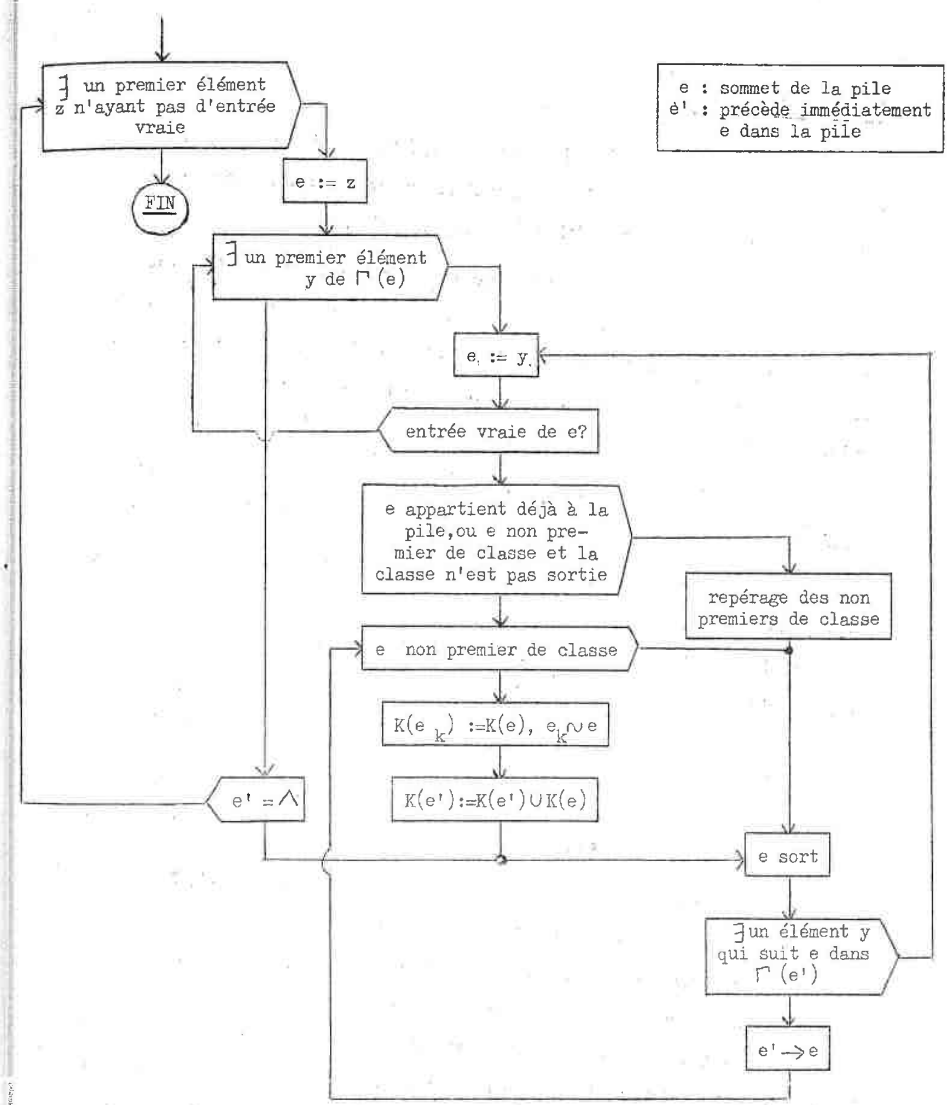
1) Le théorème 16 entraîne y est non premier de classe ;

2) Montrons que l'on a tous les éléments non premiers de classe lors de la sortie du premier de classe :

Soit y non premier de classe. Le théorème 16 entraîne : il existe z et k tels que $v_k = \alpha\ \ell(z,k)\ \beta\ y\ \gamma\ z$. A l'instant k inférieur à la sortie du premier de classe puisque y non premier de classe appartient à v_k, on met le signe $\sim$ aux éléments de la pile compris entre $\ell(z,k)$ et z, donc à y.

Détermination du premier de classe.

C'est un élément x qui ne possède pas le signe $\sim$ à l'instant $\Omega(x)$



PROGRAMMATION.

La programmation de cette méthode exige un emplacement réservé à la matrice associée au graphe, un emplacement réservé à sa matrice de fermeture transitive stricte, une pile.

D'autre part, il faut savoir à chaque instant i si un élément x appartient à u_i, si $\Omega(x) < i$, si un élément est non premier de classe, et dans ce cas quel est son lien. Or, à chaque élément x correspond une ligne de la matrice associée. Il paraît donc logique "d'ajouter" à la matrice associée les colonnes nécessaires à ces renseignements.

C'est ce que nous faisons pour noter l'entrée et la sortie vraiesde l'élément ;

Nous ajoutons une colonne T ainsi définie (x_α et x_β étant les éléments de code α et β)

- à tout instant inférieur à l'entrée vraie de x_α : $T(\alpha) = 0$;
- à tout instant compris entre l'entrée et la sortie vraies de x_α :
  $T(\alpha) = \beta$, x_β est le dernier élément de $\Gamma(x_\alpha)$ ayant eu une entrée dans la pile V ;
- à tout instant supérieur à la sortie vraie de x_α :
  $T(\alpha)$ = valeur négative.

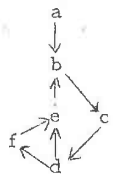
En fait, ce procédé est commode pour noter si, à l'instant i, un élément x a déjà eu sa sortie vraie, mais il ne convient pas très bien pour noter si un élément est non premier de classe. En effet, à l'instant k défini dans le théorème 16, on met le signe $\sim$ et un lien à tout élément y dans la colonne correspondante de la matrice associée ; quand un élément z sort, il faut savoir s'il est le lien d'un élément y et dans ce cas changer de lien y. Ces deux opérations nécessitent de nombreux tests sur les termes de la colonne additionnelle.

Pour ces deux raisons, il nous a paru plus simple de constituer une pile parallèle à la pile annexe dans laquelle on met les signes $\sim$ en regard des éléments y non premiers de classe. Lorsqu'un élément non premier de classe sort, au lieu de lui mettre le lien défini plus haut, on conserve l'élément dans la pile en dessous de son lien avec le signe $\lceil$ qui signifie élément non premier de classe passif pour la pile V.

De ce fait, à l'instant de sortie vraie d'un élément z non premier de classe, il suffit de lui mettre le signe $\lceil$ et de l'échanger dans la pile avec le nouveau sommet x pour donner le lien x aux éléments qui avaient le lien z.

Lors de la sortie du premier de classe x, on a tous les éléments équivalents à x en dessous de lui avec le signe $\lceil$. En effet, x est le dernier lien de tous les éléments qui lui sont équivalents.

Exemple :



Soit à rechercher la fermeture transitive du graphe ci-contre. On supposera que les éléments d'une même famille sont pris dans l'ordre alphabétique. Nous donnons ci-dessous les différents états de la pile annexe et de la pile parallèle.

								e						
						b			f	f ∿f	∿d	∿c	b	
				e	e	∿e	∿d	∿d	∿d	∿d	∿d	⌈f	⌈f	⌈f
			d	d	d	∿d	⌈e	⌈e	⌈e	⌈e	⌈e	⌈e	⌈e	
		c	c	c	c	∿c	∿c	∿c	∿c	∿c	∿c	∿c	⌈d	⌈d
	b	b	b	b	b	b	b	b	b	b	b	b	b	⌈c
∧	a	a	a	a	a	a	a	a	a	a	a	a	a	a ∧

Représentation des ensembles K(x,i).

Il est intéressant de mettre les ensembles K(x,j) sous forme d'une matrice booléenne $\mathcal{H}$ fonction de l'instant j ainsi définie : $k^{\alpha}_{\beta}(j) = 1$ si à l'instant j l'élément x_α a été trouvé en relation avec x_β ($x_\alpha R x_\beta$). La méthode exposée plus haut nécessite l'initialisation de la matrice $\mathcal{H}(0)$ par la matrice associée au graphe $\mathcal{G}$.

Il est intéressant de faire coïncider les matrices $\mathcal{H}(0)$ et $\mathcal{G}$, c'est-à-dire de faire varier $\mathcal{G}$ en fonction de j. Cela exige que l'on puisse distinguer les éléments y tels que $x \Gamma y$ et les éléments z tels que $x \Gamma z$ et $x R z$. Pour cela, la matrice booléenne $\mathcal{G}$ est remplacée par une matrice $\mathcal{H}(j)$ dont les termes peuvent prendre trois valeurs :

- $a^{\alpha}_{\beta}(j) = 0$ si à l'instant j on a $x_\alpha \Gamma x_\beta$ et $K(x_\alpha, j)$ ne contient pas x_β
- $a^{\alpha}_{\beta}(j) = 1$ si on a $x_\alpha \Gamma x_\beta$ et à l'instant j $K(x_\alpha, j)$ ne contient pas x_β
- $a^{\alpha}_{\beta}(j) = 2$ si à l'instant j $K(x_\alpha, j)$ contient x_β.

Les opérations booléennes (α), (β) et (δ) sont transformées en les opérations suivantes (*) où $z[k]$ est élément de la classe de z_0 qui contient $\ell + 1$ éléments, et n est le nombre de points du graphe :

(α) devient

pour k:=1 pas 1 jusqu'à ℓ faire
pour i:=1 pas 1 jusqu'à n faire
si $a[y[k],i] > 0$ alors $a[y[\Gamma],i] := 2$; (1)
puis : pour k:=1 pas 1 jusqu'à ℓ faire
pour i:=1 pas 1 jusqu'à n faire
$a[y[k],i] := a[y[0],i]$; (2)

(β) et (δ) deviennent :

pour i:=1 pas 1 jusqu'à n faire
si $a[z,i] > 0$ alors $a[x,i] := 2$ (3)

(*) Remarque : en programmation l'instant j n'intervient que sur la succession des opérations.

Mais ces opérations modifient la matrice de poids 1 initialement matrice $\mathcal{G}$ associée au graphe :

En effet, lors de l'opération (3), si on a simultanément $a[z,i] > 0$ et $a[x,i] = 1$, on obtient à la fin de l'opération $a[x,i] = 2$ ce qui équivaut à supprimer l'élément x_i de $\Gamma(x)$.

Or, on effectue (3) dans deux cas :

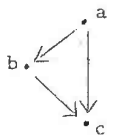
- lors de la sortie vraie de z premier de classe,
- lors d'une sortie de z supérieure à la sortie du premier de sa classe.

Dans ces deux cas, si j est l'instant où l'opération s'effectue on a : $K(z,j) \subseteq K(x,j)$, $j \geq \Omega(z_0)$, z_0 premier de la classe de z. D'autre part, si $a[z,i] > 0$, alors $z R x_i$. Comme $j \geq \Omega(z_0)$, on a : $\Omega(x_i) < \Omega(z_0) \leq j$ et $K(x_i,j) \subseteq K(z,j)$ d'où $K(x_i,j) \subseteq K(x,j)$.

Soit h l'entrée de x_i quand x est sommet. Si $h < j$, alors le fait de supprimer l'élément x_i au moment j n'a pas d'importance puisque x_i a déjà eu une entrée. Si $h > j$ alors $h+1$ est sortie de x_i et provoque (δ) $K(x,h+1) = K(x,h) \cup K(x_i,h)$ inutile puisque $K(x_i,h) = K(x_i,j) \subseteq K(x,j)$.

Exemple :

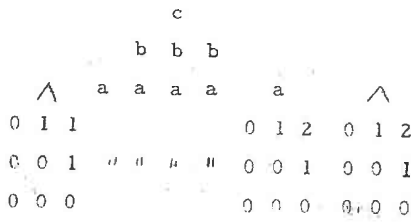
Soit le graphe suivant :



La matrice associée $\mathcal{H}$ est la suivante :

	a	b	c
a	0	1	1
b	0	0	1
c	0	0	0

Nous donnons ci-dessous les états u_i de la pile et la matrice $\mathcal{H}(i)$ correspondante.



COMPARAISON DES METHODES MATRICIELLE ET PILE ANNEXE.

Nous allons rappeler la méthode matricielle [WARSHALL] [NOLIN] d'obtention de la fermeture transitive stricte du graphe, nous donnerons ensuite le programme ALGOL employé et comparerons les méthodes au point de vue rapidité d'exécution et encombrement des mémoires.

Rappel de la méthode matricielle.

La procédure ALGOL est la suivante :

```
procédure FTSM(n,a) ; valeur n ; booléen tableau a ;
début entier i,j,k ;
        pour k:=1 pas 1 jusqu'à n faire
        pour i:=1 pas 1 jusqu'à n faire
        pour j:=1 pas 1 jusqu'à n faire
        a[i,j]:= a[i,j] ∨ a[i,k]∧a[k,j] fin ;
```

où n est l'ordre du graphe et a le tableau initialisé par la matrice associée au graphe.

Nous avons utilisé la procédure sous la forme équivalente suivante :

```
procédure FTSM(n,a) ; valeur n ; booléen tableau a ;
début entier i,j,k ;
        pour k:=1 pas 1 jusqu'à n faire
        pour i:=1 pas 1 jusqu'à n faire
        si a[i,k] alors
            pour j:=1 pas 1 jusqu'à n faire
        a[i,j] := a[i,j] ∨ a[k,j] fin ;
```


Ce qui nous permet de comparer le nombre d'opérations

```
pour j:=1 pas 1 jusqu'à n faire
    a[i,j] := a[i,j] ∨ a[k,j]
```

et le nombre d'opérations analogues (1) et (3) de la méthode de la pile annexe, nécessaires pour obtenir la fermeture transitive stricte d'un même graphe.

Temps d'exécution.

Nous avons testé trois graphes d'ordre 40 :
un graphe sans circuit, le même graphe auquel on avait ajouté quelques arcs pour former des classes simples (classes d'équivalence formées d'un seul circuit), le même graphe auquel on avait ajouté de nombreux arcs supplémentaires donnant des classes complexes (classes d'équivalence comportant plusieurs circuits).

Nous avons obtenu les résultats suivants :

Nombre d'opérations	Graphe sans circuit	Graphe avec circuits simples	Graphes avec circuits complexes
Méthode matricielle	108	187	259
Méthode de la pile annexes	22	38	38

La simple considération du nombre d'opérations ne suffit pas pour décider de la méthode la plus rapide : en effet, la méthode de la pile annexe nécessite des "entrées" et "sorties" de la pile, des échanges d'éléments à l'intérieur de la pile dans les cas de graphe possédant des circuits. Aussi le temps effectif d'exécution dépend-il essentiellement de l'ordinateur utilisé.

Les essais effectués sur 650 IBM (en langage PASØ) où les opérations se faisaient par des sous-programmes, ont donné un très net avantage à la méthode de la pile annexe .

Pour différents graphes allant de l'ordre 40 à l'ordre 100, le rapport des temps d'exécution a varié de 5 à 10 en faveur de la méthode de la pile annexe .

Les essais effectués sur Γ 60 BULL (en langage C7) où les opérations se faisaient directement dans le calculateur logique mais où le travail sur la pile était très fastidieux, ont donné pour des graphes d'ordre 100 :

Temps d'exécution	Graphe sans circuit	Graphes avec circuits simples	Graphes avec circuits complexes
Méthode matricielle	5 s	5,5 s	7 s
Méthode de la pile annexe	18 s	20 s	17,5 s

La méthode matricielle se révèle ici plus rapide dans le rapport moyen 3.

Enfin, les essais effectués sur CAE 510 (en langage ALGOL) ont donné les résultats suivants que l'on pourra comparer utilement avec les nombres d'opérations relatifs aux mêmes graphes.

Temps d'exécution	Graphe sans circuit	Graphes avec circuits simples	Graphes avec circuits complexes
Méthode matricielle	85 s	133 s	180 s
Méthode de la pile annexe	30 s	100 s	105 s

Remarques :

1) Tous les temps donnés ci-dessus excluent les temps d'impression des résultats, d'entrée des données ;

2) La méthode de la pile annexe donne comme "sous-produits" les classes d'équivalence : en effet, lorsqu'un élément premier de sa classe et appartenant à un circuit sort, tous les éléments qui lui sont équivalents se trouvent sous lui avec le signe [en regard. Il suffit à ce moment de sortir la partie de pile intéressée ;

3) Nous avons choisi la matrice associée comme donnée du graphe, forme la mieux adaptée à la méthode matricielle.
Mais ce n'est pas la seule méthode de mémorisation :
En effet, on peut se donner pour chaque élément x sa famille Γ (x).
Cette mémorisation avantageuse pour la méthode de la pile annexe (puisque, lorsqu'un élément est au sommet, on teste les éléments de sa famille) nécessite la transformation en matrice associée pour être traité par la méthode matricielle.

Encombrement des mémoires. (*)

Ici, l'avantage de la méthode matricielle est évident :

- programme :
les différents programmes écrits en langage machine ont donné un rapport de 1 à 4 en faveur de la méthode matricielle.
Les programmes ALGOL comportent respectivement (entrées-sorties comprises) : 14 lignes pour la méthode matricielle, et
51 pour la méthode de la pile annexe .
(voir annexe).

(*) Nous ne parlons pas de l'encombrement dû au fait que la matrice associée est déclarée booléen tableau, alors que la matrice a utilisée est déclarée entier tableau, car dans le compilateur ALGOL que nous possédons, le booléen occupe une mémoire tout comme l'entier. S'il n'en était pas ainsi, il serait plus économique de travailler sur deux tableaux booléens (matrice associée et matrice de la relation R) que sur un seul tableau entier.

- mémoires auxiliaires :

la méthode pile annexe nécessite en plus de la matrice associée une pile de n mémoires (la pile parallèle pouvant être inclue dans cette dernière pile), une colonne additionnelle à la matrice associée de n mémoires, soit au total 2n mémoires supplémentaires.

Conclusions.

Les résultats donnés ici demandent à être complétés par d'autres plus systématiques, pour en tirer une conclusion définitive. Néanmoins, on peut affirmer que si la méthode de la pile annexe n'est pas supérieure à la méthode matricielle, elle lui est quand même comparable.

D'autre part, la méthode de la pile annexe donnée ici est susceptible d'améliorations permettant d'éviter les échanges à l'intérieur de la pile, et la pose des signes $\sim$ et $[$ qui se révèlent assez longs :

Lorsque h est une entrée de z appartenant déjà à v_{h-1}, il suffit de donner à y (sommet de v_{h-1}) le lien z.

A l'instant i sortie de l'élément z qui est son propre lien, alors si z possède un autre lien, on donne cet autre lien à tout x qui avait z comme lien. Si z n'a pas d'autre lien, on fait l'opération :

$K(x,i) = K(z,i-1)$ pour tout x ayant le lien z.

Les liens peuvent alors se mettre dans la matrice $\mathcal{A}$ et sont transférés lors des opérations (α), (β) (γ) de la méthode générale.

PROGRAMME ALGOL.

Les signes $\sim$ et $[$ sont respectivement un neuf en cinquième position décimale du mot courant de la pile annexe et un neuf en cinquième et quatrième positions décimales (le passage du signe $\sim$ au signe $[$ se fait en ajoutant l'entier 9 000). La procédure VALEUR(I) permet de restituer l'élément du niveau I de la pile, débarassée des signes $\sim$ ou $[$ éventuels.

Le terme $T(\alpha)$ de la colonne additionnelle peut avoir les valeurs suivantes :

$T(\alpha) = 0$: x_α n'a pas encore eu d'entrée dans V ;
$T(\alpha) > 0$: x_α a eu une entrée vraie mais n'a pas encore eu de sortie vrai
$T(\alpha) = -10$: x_α a déjà eu sa sortie vraie de V ;
$T(\alpha) = -1$: $\Gamma(x_\alpha) = \emptyset$.

L'indicateur L permet de savoir si l'instant précédent était une entrée ($L \neq 0$) ou une sortie ($L = 0$) :
Si on a simultanément $L \neq 0$, $T(\alpha) > 0$, x_α vient d'entrer dans V alors x_α qui était déjà dans V appartient à un circuit.

L'opération (1) de la méthode générale est faite dans le même bloc que l'opération (3). Pour cela, le test $P[I] > 99000$ permet d'exploiter les éléments non premiers de classe lors de la sortie du premier de classe, par un branchement sur l'opération (3). Après chaque opération, la pile est baissée d'un niveau et le programme se branche au test $P[I] > 99000$. L'indicateur D permet de déclencher l'opération (2) après la sortie du dernier non premier de classe.

CHAPITRE III

APPLICATION DE LA FERMETURE TRANSITIVE A LA CONSTRUCTION DE DEUX MATRICES DE PRIORITE

DEFINITIONS.

D. 20. Vocabulaire.

Soit T et N deux ensembles disjoints respectivement appelés terminal et non terminal. Le vocabulaire est l'ensemble V tel que $V = T \cup N$.

D. 21. Grammaire.

C'est un ensemble de couples X, λ tels que $X ::= \lambda$ avec X élément non terminal, λ mot sur le vocabulaire V (éventuellement vide) ::= relation de Bacchus. $X ::= \lambda$ est appelée règle de grammaire, X en est le premier membre, λ le second.

Dans les définitions qui suivent, on notera A, B et C des éléments du vocabulaire, X, Y et Z des éléments non terminaux, x,y,z des éléments terminaux, λ et μ des mots sur le vocabulaire V.

D. 22. Relations "a pour successeur droit" et "a pour successeur gauche".

On dit que $X \in N$ a pour successeur droit (respectivement a pour successeur gauche) $A \in V$ si, et seulement si, il existe une règle de grammaire du type : $X ::= \mu A$ (respectivement $X ::= A \mu$). On note $X \mathcal{S}^* A$ (resp. $X {}^*\mathcal{S} A$).

D. 23. Initiales, finales. Initiales, finales strictes. Initiales, finales terminal

- Les relations de fermeture transitive de S^* et ${}^*S$ seront notées respectivement F' et I'.

Tout élément A appartenant à V tel que $XF'A$ (resp. $XI'A$) est appelé finale (resp. initiale) de X.

- Les finales (resp. initiales) de X appartenant à T sont appelées finales terminales (resp. initiales terminales) de X.

Si x est finale terminale (resp. initiale terminale) de A, on note : $AF'_t x$ (resp. $AI'_t x$).

- Les relations de fermeture transitive stricte de S^* et ${}^*S$ seront notées respectivement F et I.

Tout élément A appartenant à V tel que XFA (resp. XIA) est appelé finale stricte (resp. initiale stricte) de X.

D. 24. Descendants droits et gauches.

$x \in T$ est dit descendant droit (resp. descendant gauche) de X si, et seulement si, il existe une règle de grammaire du type : $X' ::= Y x \mu$ ou $X' ::= x \mu$ avec X' initiale non terminale de X (resp. $X' ::= \mu x Y$ ou $X' ::= \mu x$ avec X' finale non terminale de X).

On note XGx (resp. XDx).

D. 25. Relations de priorités définies sur une grammaire.

On peut définir les trois relations suivantes sur les grammaires les plus générales :

$A \approx B$ si, et seulement si, il existe une règle du type $X ::= \lambda A B \mu$;

$A \lessdot B$ si, et seulement si, il existe une règle du type $X ::= \lambda A Y \mu$
avec B initiale stricte de Y ;

$A \gtrdot x$ si, et seulement si, il existe une règle du type $X ::= \lambda Y x \mu$
avec A finale stricte de Y
ou il existe une règle du type $X ::= \lambda Y B \mu$
avec A finale stricte de Y et x initiale terminale de B.

D. 26. Grammaire d'opérateurs [FLOYD].

C'est une grammaire n'ayant pas de règle du type $X ::= \lambda Y Z \mu$ $(Y, Z \in N)$.

D. 27. Relations de priorité sur une grammaire d'opérateurs [FLOYD].

On définit les trois relations suivantes sur une grammaire d'opérateurs :

$x \doteq y$ si, et seulement si, il existe une règle du type $X ::= \lambda x y \mu$
ou il existe une règle du type $X ::= \lambda x Y y \mu$;

$x \lessdot y$ si, et seulement si, il existe une règle du type $X ::= \lambda x Y \mu$
avec y descendant gauche de Y ;

$x \gtrdot y$ si, et seulement si, il existe une règle du type $X ::= \lambda X y \mu$
avec x descendant droit de X.

CONSTRUCTION DES MATRICES DE PRIORITE.

Dans ce qui suit, nous adopterons les notations suivantes :

$M\approx$, $M\lessdot$ et $M\gtrdot$ matrices des relations $\approx$, $\lessdot$, $\gtrdot$.

$M\doteq$, $M\lessdot$ et $M\gtrdot$ matrices des relations $\doteq$, $\lessdot$, $\gtrdot$.

$\mathfrak{I}$, $\mathfrak{F}$, $\mathfrak{I}_t$, matrices des relations I, F, I_t.

$\mathfrak{U}$ matrice unité.

Première matrice.

Remarquons tout d'abord que :

$A \lessdot B$ est équivalent à : il existe Y tel que $A \approx Y$ et $Y I B$. D'après D. 2, la relation $\lessdot$ est le produit des relations $\approx$ et I fermeture transitive stricte de ${}^*S$.

On en déduit l'égalité matricielle suivante :

$$M\lessdot = M\approx \dot{x} \mathfrak{I} .$$

De même $A \gtrdot x$ est équivalent à : il existe Y et B tels que YFA et $Y \approx B$ et $BI_t x$.

On en déduit l'égalité matricielle suivante :

$M\gtrdot = \left[(\mathfrak{F})^t \dot{x} M\approx \dot{x} I_t \right]$ où $(\mathfrak{F})^t$ est la matrice transposée de $\mathfrak{F}$.

Deuxième matrice.

Les mêmes remarques que précédemment conduisent à :
$M \lessdot = M \simeq \dot{x} G$ avec $M \simeq$ matrice de la relation $\simeq$ telle que $x \simeq Y$ si, et seulement si, il existe une règle du type $X ::= \lambda x Y \mu$ de même :
$M \gtrdot = (\mathcal{D})^t \dot{x} (M \simeq)^t$ avec $(M \simeq)^t$ matrice transposée de $M \simeq$.

PROGRAMMATION.

Première matrice.

Les règles de grammaire permettent d'obtenir directement $M \approx$, $\mathcal{G}^*$ et ${}^*\mathcal{G}$. Le sous-programme de fermeture transitive stricte donne $\mathcal{F}$ et $\mathcal{J}$.

Il est plus rapide d'obtenir $M \lessdot$ et $M \gtrdot$ à partir des règles de grammaire, de $\mathcal{F}$ et de $\mathcal{J}_t$ que de faire les produits matriciels de la méthode citée plus haut.

On donne le poids 1 aux termes de la matrice $M \lessdot$, le poids 2 aux termes de la matrice $M \gtrdot$ et le poids 4 aux termes de la matrice $M \approx$. En faisant $M1 = M \lessdot + M \gtrdot + M \approx$ on peut alors facilement voir si deux ou trois relations sont compatibles.

Deuxième matrice.

Les règles de grammaire donnent directement les matrices ${}^*\mathcal{L}$ et $\mathcal{L}^*$ associées aux relations ${}^*T$ et $'T^*$:
$X {}^*T x$ (resp. $X T^* x$) si, et seulement si, il existe une règle de grammaire du type
$X ::= Y x$ ou $X ::= x \mu$ (resp. $X ::= \mu x Y$ ou $X ::= \mu x$).

Le sous-programme de fermeture transitive donne $\mathcal{J}$ et $\mathcal{F}$.
On obtient alors immédiatement $\mathcal{D}$ et G.

On remarque que $(M \gtrdot)^t = M \simeq \dot{x} \mathcal{D}$ et que $\mathcal{D}$ est égal à la matrice G d'une grammaire d'opérateurs "symétrique" de la grammaire d'opérateurs initiale, c'est-à-dire où chaque règle $X ::= \lambda$ est déduite de la règle $X ::= \lambda'$ de la grammaire initiale, par $\lambda =$ mot dont les éléments sont deux de λ' pris en sens inverse.

Il suffit donc de reprendre le programme avec les règles symétriques pour obtenir $(M \gtrdot)^t$.

On donne le poids 1 à $M \lessdot$, le poids 2 à $M \gtrdot$ et le poids 4 à $M \doteq$. La matrice $M2 = M \lessdot + M \gtrdot + M \doteq$ permet de voir facilement si deux ou trois relations sont compatibles.

Nous avons programmé les deux problèmes sur 650 IBM. La première matrice nécessite la mémorisation des matrices des initiales strictes, des finales, de la matrice $M \approx$ et enfin de la matrice résultat, soit au total 4 matrices carrées dont la dimension est égale au nombre d'éléments du vocabulaire. Sur 650 IBM, il restait environ 1000 mémoires pour ces matrices (500 mémoires étant occupées par le programme et 500 autres par le sous-programme de fermeture transitive et les règles de grammaire). Ceci conduit à un nombre d'éléments total de 50. Il n'est donc pas question de traiter une grammaire ALGOL dont le nombre d'éléments avoisine 150. Nous avons donc traité une "sous-grammaire" ALGOL, celle de l'expression arithmétique simple où la grammaire légèrement modifiée est donnée ci-dessous, avec la matrice résultat.

EXEMPLES.

Règles de grammaire :

⟨Expression Arithmétique Simple⟩ ::=
⟨Expression Arith. Simple⟩ ⟨Opérateur additif⟩ ⟨Terme⟩
⟨Opérateur additif⟩ ⟨Terme⟩
⟨Terme⟩
⟨Opérateur additif⟩ ::= +
-

⟨Terme⟩ ::= ⟨Terme⟩ ⟨Opérateur multiplicatif⟩ ⟨Facteur⟩
⟨Facteur⟩
⟨Opérateur multiplicatif⟩ ::= x
-
/

⟨Facteur⟩ ::= ⟨Facteur⟩ ↑ ⟨Primaire arithmétique⟩
⟨Primaire arithmétique⟩

⟨Primaire arithmétique⟩ ::= ⟨Variable⟩
⟨Nombre sans signe⟩

⟨Nombre sans signe⟩ ::= ⟨Nombre décimal⟩ ⟨Facteur de cadrage⟩
⟨Facteur de cadrage⟩
⟨Nombre décimal⟩

⟨Nombre décimal⟩ ::= ⟨Entier sans signe⟩ ⟨Partie décimale⟩
⟨Partie décimale⟩
⟨Entier sans signe⟩

⟨Partie décimale⟩ ::= . ⟨Entier sans signe⟩

⟨Entier sans signe⟩ ::= ⟨Entier sans signe⟩ ⟨Chiffre⟩
⟨Chiffre⟩

⟨Facteur de cadrage⟩ ::= 10 ⟨Entier⟩

⟨Entier⟩ ::= ⟨Signe⟩ ⟨Entier sans signe⟩
⟨Entier sans signe⟩

⟨Signe⟩ ::= +
-

⟨Variable⟩ ::= ⟨Identificateur⟩

⟨Identificateur⟩ ::= ⟨Identificateur⟩ ⟨Chiffre⟩
⟨Identificateur⟩ ⟨Lettre⟩
⟨Lettre⟩

La seconde matrice ne nécessite à un instant donné que la mémorisation de deux matrices, l'une rectangulaire (matrice des descendants), l'autre carrée (matrice résultat). Sur 650 IBM, les 1000 mémoires disponibles pour ces matrices permettent de traiter des grammaires opérateur de l'importance d'ALGOL, c'est-à-dire comportant environ 50 éléments terminaux et 150 éléments non terminaux.

Nous avons modifié la grammaire ALGOL de façon à la rendre grammaire opérateur. Nous donnons ci-dessous les règles de la grammaire modifiée et la référence du rapport ALGOL 60 des règles correspondantes.

	Exp. Arith. Simple	Opérateur additif	Terme	Opérateur multiplicatif	Facteur	Primaire arithmétique	Nombre sans signe	Nombre décimal	Partie décimale	Entier sans signe	Facteur de cadrage	Entier	Signe	Variable	Identificateur	+	-	x	÷	/	↑	.	10	0 1 ... 8 9	aAbB ... zZ
Exp. arit. simple	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
Opérateur additif	0	0	5	0	1	1	1	1	1	1	1	0	0	1	1	0	0	1	1	0	0	0	0	1	1
Terme	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	2	2	1	1	1	0	0	0	0	0
Opérat. multiplicat.	0	0	0	0	5	1	1	1	1	1	1	0	0	1	1	0	0	0	0	0	0	1	1	1	1
Facteur	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	4	0	0	0	0
Primaire arithmét.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	0	0	0	0
Nombre sans signe	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	0	0	0	0
Nombre décimal	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	2	2	2	2	2	2	0	1	0	0
Partie décimale	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	0	2	0	0
Entier sans signe	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	2	2	2	2	2	2	1	2	4	0
Facteur de cadrage	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	0	0	0	0
Entier	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	0	0	0	0
Signe	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Variable	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	0	0	0	0
Identificateur	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	0	0	4	4
+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2
x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2
÷	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2
/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2
↑	0	0	0	0	0	4	1	1	1	1	1	0	0	1	1	0	0	0	0	0	0	1	1	1	1
.	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
10	0	0	0	0	0	0	0	0	0	1	0	4	1	0	0	1	1	0	0	0	0	0	0	1	0
0 1 ... 8 9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2
aAbB ... zZ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	0	2	2

1 : Relation $\lessdot$
2 : Relation $\gtrdot$
3 : Relations $\lessdot$ et $\gtrdot$
4 : Relation $\approx$
5 : Relations $\lessdot$ et $\approx$
6 : Relations $\gtrdot$ et $\approx$
7 : Relations $\lessdot$, $\gtrdot$ et $\approx$.

Nous avons supprimé les éléments non terminaux suivants :

⟨partie paramètre effectif⟩ , ⟨délimiteur de paramètre⟩

⟨opérateur de relation⟩

⟨lettre⟩ , ⟨chiffre⟩

⟨opérateur additif⟩ , ⟨opérateur multiplicatif⟩ , ⟨facteur de cadrage⟩

⟨partie décimale⟩ .

Nous avons ajouté un élément non terminal que nous avons appelé

⟨partie complémentaire⟩ ,

et six éléments terminaux :

α signifiant $< \leqslant = \geqslant > \neq$

β signifiant + ou -

γ signifiant x / ou ÷

δ signifiant 0 1 2 ... 8 ou 9

ε signifiant a A b B ... z ou Z

et type signifiant réel booléen ou entier

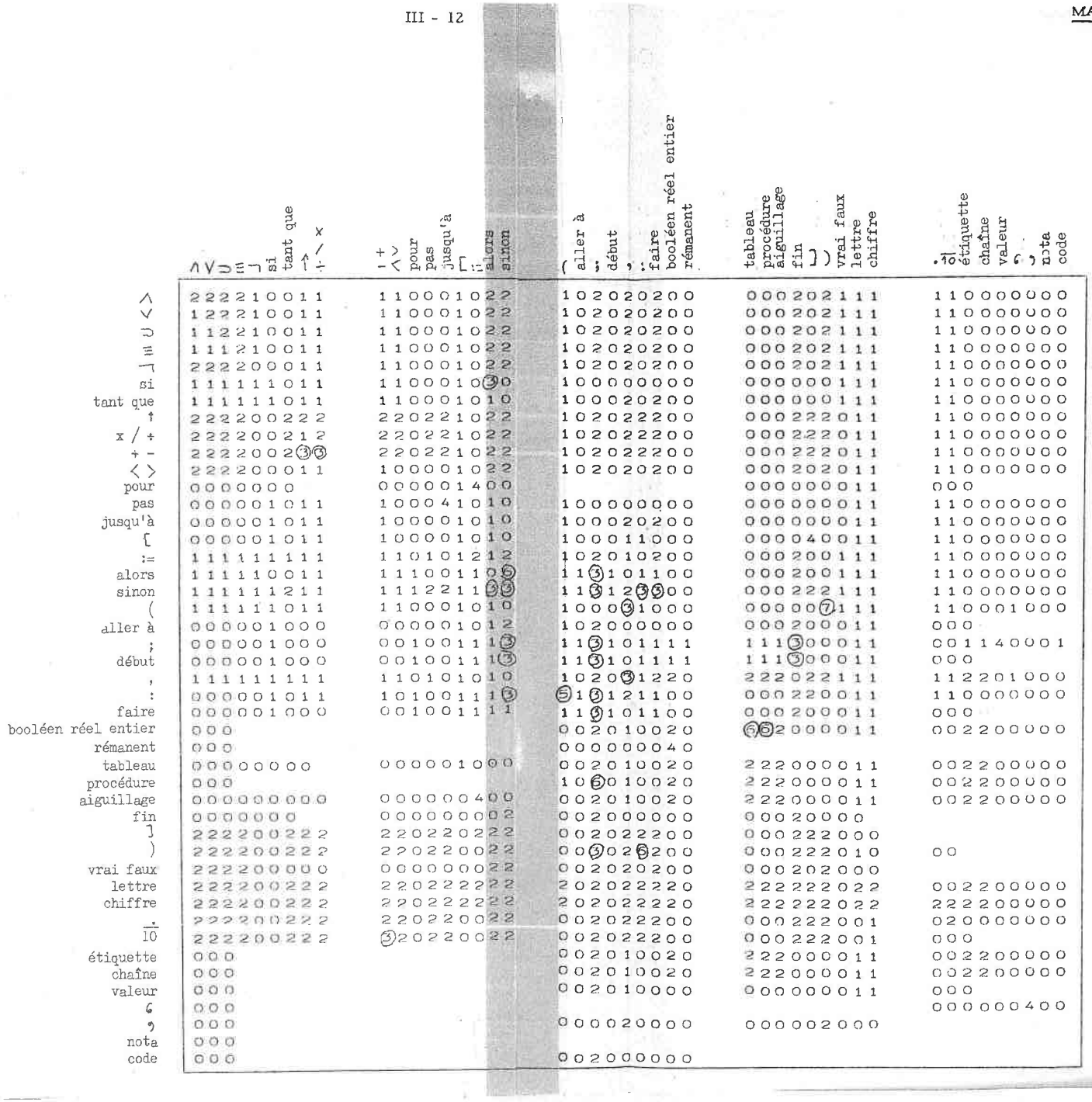
Le concept ⟨vide⟩ a été supprimé.

Paragraphe du rapport ALGOL	Règle ALGOL modifiée
2. 4. 1.	⟨Identificateur⟩ ::= ⟨identificateur⟩ δ ⟨identificateur⟩ ε ε
2. 5. 1.	⟨Entier sans signe⟩ ::= ⟨entier sans signe⟩ δ δ
2. 5. 1.	⟨Entier⟩ ::= ⟨entier sans signe⟩ β ⟨entier sans signe⟩
2. 5. 1.	⟨Nombre décimal⟩ ::= ⟨entier sans signe⟩ . ⟨entier sans signe⟩ . ⟨entier sans signe⟩ ⟨entier sans signe⟩
2. 5. 1.	⟨Nombre sans signe⟩ ::= ⟨nombre décimal⟩ 10 ⟨entier⟩ 10 ⟨entier⟩ ⟨Nombre décimal⟩

Paragraphe du rapport ALGOL	Règle ALGOL modifiée
2. 6. 1.	⟨Chaîne propre⟩ ::= nota
2. 6. 1.	⟨Chaîne ouverte⟩ ::= nota nota nota ⟨chaîne ouverte⟩ ⟨chaîne ouverte⟩ nota ⟨chaîne propre⟩
2. 6. 1.	⟨Chaîne⟩ ::= ' ' '⟨chaîne ouverte⟩'
3. 2. 1.	⟨Indicateur de fonction⟩ ::= ⟨identificateur de procédure⟩ (⟨liste de paramètres effectifs⟩)
3. 3. 1.	⟨Terme⟩ ::= ⟨terme⟩ γ ⟨facteur⟩ ⟨facteur⟩
3. 3. 1.	⟨Expression arithmétique simple⟩ ::= ⟨expression arithmétique simple⟩ β ⟨terme⟩ β ⟨terme⟩ ⟨terme⟩
3. 3. 1.	⟨Expression arithmét.⟩ ::= ⟨proposition si⟩ alors ⟨E. A. S.⟩ sinon ⟨E. A. ⟨E. A. S.⟩
3. 4. 1.	⟨Relation⟩ ::= ⟨E. A. S.⟩ α ⟨E. A. S.⟩
3. 4. 1.	⟨Expression booléenne⟩ ::= ⟨proposition si⟩ alors ⟨E. B. S.⟩ sinon ⟨E. B. ⟨E. B. S⟩
3. 5. 1.	⟨Expression de désignation⟩ ::= ⟨proposition si⟩ alors ⟨E. D. S.⟩ sinon ⟨E.D. ⟨E. D. S.⟩
4. 2. 1.	⟨Partie gauche⟩ ::= ⟨variable⟩ ⟨identificateur de procédure⟩
4. 2. 1.	⟨Liste de parties gauches⟩ ::= ⟨liste de parties gauches⟩ := ⟨partie gauche⟩ ⟨partie gauche⟩

Paragraphe du rapport ALGOL	Règle ALGOL modifiée
4. 2. 1.	⟨Instruction d'affectation⟩ ::= ⟨liste de parties gauches⟩ := ⟨E. B.⟩
	⟨liste de parties gauches⟩ := ⟨E. A.⟩
4. 5. 1.	⟨Proposition si⟩ ::= si ⟨expression booléenne⟩
4. 5. 1.	⟨Instruction si⟩ ::= ⟨proposition si⟩ alors ⟨instruction inconditionnelle⟩
4. 5. 1.	⟨Instruction condition.⟩ ::= ⟨étiquette⟩ : ⟨instruction condition.⟩
	⟨instruction si⟩
	⟨instruction si⟩ sinon ⟨instruction⟩
	⟨proposition si⟩ alors ⟨instruction pour⟩
4. 6. 1.	⟨Proposition pour⟩ ::= pour ⟨variable⟩ := ⟨liste de pour⟩
4. 6. 1.	⟨Instruction pour⟩ ::= ⟨étiquette⟩ : ⟨instruction pour⟩
	⟨proposition pour⟩ faire ⟨instruction⟩
4. 7. 1.	⟨Liste de paramètres effectifs⟩ ::=
	⟨liste de param. effectifs⟩) ⟨chaîne de lettres⟩ : (⟨partie param. effectifs
	⟨liste de param. effectifs⟩ , ⟨paramètre effectif⟩
	⟨paramètre effectif⟩
4. 7. 1.	⟨Instruction procédure⟩ ::= ⟨identificateur de procédure⟩ (⟨liste de param. effectif⟩)
	⟨identificateur de procédure⟩
5. 1. 1.	⟨Type⟩ ::= type
5. 1. 1.	⟨Déclaration de type⟩ ::= type ⟨liste de types⟩
	rémanent type ⟨liste de types⟩
5. 4. 1.	⟨Partie valeur⟩ ::= ; valeur ⟨liste d'identificateurs⟩

Paragraphe du rapport ALGOL	Règle ALGOL modifiée
5. 4. 1.	⟨Partie spécification⟩ ::= ⟨partie sp.⟩ aiguillage ⟨liste d'ident.⟩
	⟨partie sp.⟩ étiquette ⟨liste d'ident.⟩
	⟨partie sp.⟩ chaîne ⟨liste d'ident.⟩
	⟨partie sp.⟩ tableau ⟨liste d'ident.⟩
	⟨partie sp.⟩ procédure ⟨liste d'ident.⟩
	⟨partie sp.⟩ type ⟨liste d'ident.⟩
	⟨partie sp.⟩ type tableau ⟨liste d'ident.⟩
	⟨partie sp.⟩ type procédure ⟨liste d'id.⟩
	aiguillage ⟨liste d'identificateurs⟩
	étiquette ⟨liste d'identificateurs⟩
	chaîne ⟨liste d'identificateurs⟩
	tableau ⟨liste d'identificateurs⟩
	procédure ⟨liste d'identificateurs⟩
	type ⟨liste d'identificateurs⟩
	type tableau ⟨liste d'identificateurs⟩
	type procédure ⟨liste d'identificateurs⟩
5. 4. 1	⟨Partie complémentaire⟩ ::= ⟨partie valeur⟩ ; ⟨partie spécification⟩
	; ⟨partie spécification⟩
	⟨partie valeur⟩
5. 4. 1	⟨Tête de procédure⟩ ::= ⟨identif. de procédure⟩ (⟨liste de param. formels⟩) ⟨partie compl.⟩
	⟨identif. de procédure⟩ (⟨liste de param. formels⟩)
	⟨identif. de procédure⟩
5. 4. 1.	⟨Déclaration de procédure⟩ ::= ⟨type⟩ procédure ⟨tête de procéd.⟩ ; ⟨corps de procéd.⟩
	procédure ⟨tête de procéd.⟩ ; ⟨corps de procéd.⟩


∧ ∨ ⊃ ≡ ⌐ si tant que ↑ x / ÷ + − < > pour pas jusqu'à [:= alors sinon (aller à ; début , : faire booléen réel entier rémanent tableau procédure aiguillage fin]) vrai faux lettre chiffre . 10 étiquette chaîne valeur ' ' nota code

ALGOL II (a)

Une grammaire ALGOL modifiée par [CUSEY] pour les besoins du compilateur ALGOL de 1620 IBM a donné la matrice de priorité suivante :

1 ⟶ <
2 ⟶ >
4 ⟶ ≈

	∧ ∨ ⊃ ≡ ¬	si	tant que	↑	x / ÷	+ −	< >	pour	pas	jusqu'à	[	:=	al	ors	sinon	(	aller à	;	début	,	:	faire	type	rémanent	tableau	procédure	aiguillage	fin	]	)	vrai faux	identificateur	entier sans signe	.	10	étiquette	chaîne	valeur	chaîne propre	‘	’
∧	2	2	2	2	1	0	0	1	1	0	0	0	1	1	0	2	1	0	2	0	2	0	2	0	0	0	0	2	0	2	1	1	1	1	0	0	0	0	0	0
∨	1	2	2	2	1	0	0	1	1	0	0	0	1	0	2	0	1	0	2	0	2	0	2	0	0	0	0	2	0	2	1	1	1	1	0	0	0	0	0	0
⊃ ≡	1	1	2	2	1	0	0	1	1	0	0	0	1	0	2	0	1	0	2	0	2	0	2	0	0	0	0	2	0	2	1	1	1	1	0	0	0	0	0	0
¬	1	1	1	2	1	0	0	1	1	0	0	0	1	0	2	0	1	0	2	0	2	0	2	0	0	0	0	2	0	2	1	1	1	1	0	0	0	0	0	0
si	2	2	2	2	0	0	0	1	1	0	0	0	1	1	0	2	1	0	2	0	2	0	2	0	0	0	0	2	0	2	1	1	1	1	0	0	0	0	0	0
tant que	1	1	1	1	1	1	0	1	1	0	0	0	1	0	4	1	1	0	2	0	2	0	2	0	0	0	0	2	0	2	1	1	1	1	0	0	0	0	0	0
↑	2	2	2	2	0	0	2	2	2	0	2	2	1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
x / ÷	2	2	2	2	0	0	2	1	2	2	2	2	1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
+ −	2	2	2	2	0	0	2	3	3	2	2	2	1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
< >	2	2	2	2	0	0	0	1	1	0	0	0	1	0	2	0	1	0	2	0	2	0	2	0	0	0	0	2	0	2	1	1	1	1	0	0	0	0	0	0
pour	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
pas	0	0	0	0	0	1	0	1	1	0	0	0	2	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
jusqu'à	0	0	0	0	0	1	0	1	1	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
[	0	0	0	0	0	1	0	1	1	0	0	0	1	0	0	1	1	1	0	0	0	2	0	2	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
:=	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	0	1	1	0	0	0	4	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0
al	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	③	1	0	2	0	1	0	2	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0
ors	1	1	1	1	0	0	1	1	1	0	0	1	1	0	0	2	0	0																						
sinon	1	1	1	1	1	1	2	1	1	1	2	2	1	1	2	1	2	1	1	0	1	1	0	0	0	0	0	2	0	0	1	1	1	1	0	0	0	0	0	0
(	1	1	1	1	1	1	0	1	1	0	0	1	0	0	1	③	1	1	2	1	2	③	③	0	0	0	0	2	2	1	1	1	1	1	0	0	0	0	0	0
aller à	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	1	1	0	0	0	1	0	0	0	4	1	1	0	0	0	0	1	1	1	0	0	0	0	1	0
;	0	0	0	0	0	1	0	0	0	1	0	0	1	1	0	③	1	0	2	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
début	0	0	0	0	0	1	0	0	0	1	0	0	1	1	0	1	1	1	①	1	0	1	1	1	1	1	2	0	0	0	1	1	1	1	4	0	0	0	1	0
,	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	0	1	1	1	1	1	0	1	1	1	1	4	0	0	0	1	1	1	1	0	0	0	0	0	0
:	0	0	0	0	0	1	0	1	1	0	0	1	1	0	1	1	1	0	2	0	③	1	2	2	0	0	2	0	2	2	1	1	1	2	2	0	0	1	0	
faire	0	0	0	0	0	1	0	0	0	1	0	0	1	1	0	③	1	1	2	1	2	1	1	0	0	0	2	2	0	0	1	1	1	1	0	0	0	0	0	0
type	0	0	0													1	1	1	2	1	0	1	1	0	0	0	0	1	0	0	1	1	1	1	0	0	0	0	0	0
rémanent	0	0	0													0	0	0	2	0	1	0	0	2	0	1	1	2	0	0	0	0	1	0	0	0	2	2	0	0	0
tableau	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	1																
procédure	0	0	0													0	0	0	2	0	1	0	0	2	0	0	0	2	0	0	0	0	1	0	0	0	2	2	0	0	0
aiguillage	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	①	0	1	0	0	2	0	0	0	2	0	0	0	0	1	0	0	0	2	2	0	0
fin	0	0	0													0	0	0	2	0	1	0	0	2	0	0	0	2	0	0	0	0	1	0	0	0	2	2	0	0	0
]	2	2	2	2	0	0	2	2	2	0	2	2	0	2	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
)	2	2	2	2	0	0	2	2	2	0	2	2	0	0	0	2	0	0	2	0	2	2	2	0	0	0	0	2	2	0	0	0	0	0	0	0				
vrai faux	2	2	2	2	0	0	0	0	0	0	0	0	0	2	0	0	0	0	2	0	2	2	2	0	0	0	0	2	2	0	0	0	0	0	0	0				
identificateur	2	2	2	2	0	0	2	2	2	0	2	2	2	2	0	2	0	0	2	0	2	2	2	0	0	0	0	2	2	0	0	0	0	0	0	0				
entier sans signe	2	2	2	2	0	0	2	2	2	0	2	2	0	2	0	2	0	0	2	0	2	2	2	0	0	0	0	2	2	0	0	0	0	0	0	0				
.	0	0	0																																					
10	2	2	2	2	0	0	2	2	2	③	2	0	2	2	0	0	2	0	0	2	0	2	2	2	0	0	0	2	2	0	0	0	0	0	0	0				
étiquette	0	0	0													2	0	0	2	0	2	2	2	0	0	0	0	2	2	0	0	0	0	0	0	0					
chaîne	0	0	0													0	0	0	2	0	1	0	0	2	0	0	0	2	0	0	0	0	1	0	0	0					
valeur	0	0	0													0	0	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0					
chaîne propre	0	0	0																																					
‘	0	0	0																																					
’	0	0	0													0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					

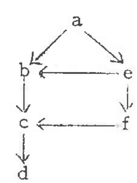
CHAPITRE IV

DETERMINATION DES CHEMINS ELEMENTAIRES DU GRAPHE

Nous nous proposons dans ce dernier chapitre de déterminer tous les chemins élémentaires du graphe en utilisant la pile annexe définie au second chapitre. Pour cela, nous allons essayer de suivre une méthode analogue à celle employée pour la recherche de la fermeture transitive stricte du graphe :

A l'instant de la sortie d'un élément y, x étant le nouveau sommet, on obtiendra les chemins élémentaires d'origine x en faisant précéder les chemins élémentaires d'origine y du point x. On conçoit que cette méthode soit bonne dans le cas du graphe sans circuit car lors de la sortie vraie d'un élément y de la pile V, on connaît tous les chemins élémentaires d'origine cet élément. Un exemple va nous permettre de le constater.

Exemple : Soit le graphe ci-contre auquel nous appliquons la méthode donnée plus haut. Les états successifs de la pile annexe sont les suivants :



				d								c				
			c	c	c				b		f	f	f			
		b	b	b	b	b		e	e	e	e	e	e	e		
∧	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	∧
					1	2	3			4			5	6	7	8

d^* 1 cd

c^* 2 bc, bcd

b^* 3 ab, abc, abcd

4 eb, ebc, ebcd

5 fc, fcd,

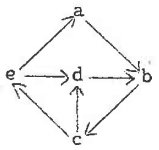
e^* 6 ef, efc, efcd

f^* 7 ae, aeb, aebc, aebcd, aef, aefc, aefcd

a^* 8

Nous avons numéroté les sorties de 1 à 8. La sortie vraie de l'élément x est repérée par x^*. En face de chaque sortie on indique les chemins élémentaires déterminés à cet instant. On constate qu'à la sortie vraie de chacun des éléments tous les chemins élémentaires d'origine cet élément ont été déterminés. Nous allons examiner maintenant le cas d'un graphe comportant des circuits et constater qu'il n'en est plus ainsi :

Exemple : en appliquant la méthode au graphe ci-contre, on obtient :



				b				d		a				
			d	d	d		e	e	e	e	e			
		c	c	c	c	c	c	c	c	c	c	c		
	b	b	b	b	b	b	b	b	b	b	b	b	b	
∧	a	a	a	a	a	a	a	a	a	a	a	a	a	a
					1	2			3		4	5	6	7

1 db

d^* 2 cd, cdb

3 ed, edb

4 ea

e^* 5 ce, ced, cedb, cea

c^* 6 bc, bcd, bce, bced, bcea

b^* 7 ab, abc, abcd, abce, abced

a^* 8

Un chemin tel que bdcea n'a pas été trouvé : en effet, la sortie vraie de d a eu lieu avant que l'on ait déterminé tous les chemins élémentaires d'origine b car la première sortie de b n'était pas la sortie vraie. Par contre, lors de la sortie vraie de b, on avait déterminé le chemin : bcea. Ceci suggère, lors de la sortie vraie de tout élément tel que b, de construire les chemins élémentaires non encore déterminés tel que dbcea. Nous allons dans le paragraphe qui suit définir de façon plus précise ce que nous venons de constater.

Elément fermeture de circuit.

On appelle élément fermeture de circuit un élément possèdant une sortie inférieure à la sortie vraie.

Dans l'exemple ci-dessus b était élément fermeture de circuit. Le circuit était le suivant : bcdb. Il en était de même de a avec le circuit abcea.

Nous allons maintenant définir des ensembles générateurs qui joueront un rôle analogue à celui des ensembles K(x,i) définis au chapitre II. Ensuite, nous définirons une opération de concaténation contractée élémentaire qui jouera le rôle de l'union logique puis une réunion contractée élémentaire qui sera l'analogue de la réunion usuelle des ensembles.

Ensembles générateurs.

On appelle ensemble générateur d'origine x et on note K(x) un ensemble de chemins élémentaires du graphe possèdant la propriété suivante :

Si $x, x_1, \ldots x_n$ appartient à K(x) alors $x, x_1, \ldots x_j$ $(0 < j < n)$ est contenu dans K(x).

Concaténation contractée élémentaire.

Soient $C = x_0, x_1, \ldots x_n$ et $C' = y_0, y_1, \ldots y_m$ deux chemins élémentaires de $K(x_0)$ et $K(y_0)$. On appelle concaténation contractée élémentaire de C et C' et on note $\underset{c}{\vee}$ l'opération suivante :

- $x_n \neq y_0$ $\quad C \underset{c}{\vee} C' = C$
- $x_n = y_0$ $\quad C \underset{c}{\vee} C' = (x_0, x_1, \ldots, x_n = y_0, y_1, \ldots, y_{i-1})$,

y_i étant le premier des y_k égal à l'un des x_j s'il en existe, sinon $y_{i-1} = y_m$.

Il résulte de cette définition que $C \underset{c}{\vee} C'$ est un chemin élémentaire d'origine x_0.

Réunion contractée élémentaire.

Soient K(x) et K(y) deux ensembles générateurs. On appelle réunion contractée élémentaire de K(x) et K(y) l'ensemble F des chemins élémentaires d'origine x obtenus par concaténation contractée de tous les chemins élémentaires C de K(x) et C' de K(y) :

$K(x) \underset{c}{\cup} K(y) = F$. D'après la définition de $\underset{c}{\vee}$, $K(x) \subseteq F$.

Montrons que F est un ensemble générateur. Pour cela, il suffit de montrer que F possède la propriété de tels ensembles :

Soit $S = C \underset{c}{\vee} C' = x, x_1, \ldots, x_n = y, y_1, \ldots, y_p$ un chemin élémentaire de F n'appartenant pas à K(x). Alors pour k inférieur à n, $x, x_1, \ldots, x_k$ appartient à K(x) inclus dans F. De même pour tout h inférieur à $C'' = y, y_1, \ldots, y_h$ appartient à K(y). $C \underset{c}{\vee} C''$ est donc contenu dans F.

METHODE.

Soient K(x,i) des ensembles générateurs tels qu'à l'instant initial de la pile V : $K(x,0) = \{$arcs d'origine x$\}$ et qu'à l'instant final m : $K(x,m) = \{$ chemins élémentaires d'origine x$\}$. Pour cela, on passe de K(x,i-1) à K(x,i) par le processus suivant :

- i est la sortie vraie de z fermeture de circuit :

  $K(a,i) = K(a,i-1) \underset{c}{\cup} K(z,i-1)$ pour tout élément a tel que z est extrémité d'un chemin élémentaire de K(a,i-1).

  Si x est le sommet de v_i :

  $K(x,i) = K(x,i-1) \underset{c}{\cup} K(z,i)$

  $K(y,i) = K(y,i-1)$ pour tout y différent de x et a.

- i est une sortie de z non fermeture de circuit, ou i est une sortie de z fermeture de circuit supérieure à la sortie vraie :

  $K(x,i) = K(x,i-1) \underset{c}{\cup} K(z,i-1)$

  $K(y,i) = K(y,i-1)$ pour tout y différent de x.

- i est une entrée, ou i est une sortie de z fermeture de circuit inférieure à la sortie vraie :

  $K(y,i) = K(y,i-1)$ pour tout y.

Avant de passer à la justification de la méthode, nous allons établir un lemme que nous utiliserons par la suite.

Lemme.

Soit $y_0, y_1, \ldots, y_p$ un chemin élémentaire, et appelons G_j l'hypothèse suivante :

Pour tout h compris entre 0 et p (p exclu), $K(y_h, j)$ contient $y_h, y_{h+1}, \ldots, y_k$, y_k étant égal à y_p ou y_k étant élément fermeture de circuit dont la sortie vraie est supérieure à j.

Montrons que G_j entraîne G_{j+1} :

- Si $k = p$ alors $K(y_h, j+1)$ contient $y_h, y_{h+1}, \ldots, y_p$.
- Si $k \neq p$:
- j+1 est sortie vraie de y_k. Alors G_j entraîne $K(y_k, j)$ contient $y_k, y_{k+1}, \ldots, y_k$ $y_{k'}$ étant égal à y_p ou $y_{k'}$ étant élément fermeture de circuit dont la sortie vraie est supérieure à j. j+1 n'étant pas sortie vraie de $y_{k'}$ $(k' > k)$: $\Omega(y_{k'}) > j+1$.

  A l'instant j+1 : $K(y_h, j+1) = K(y_h, j) \underset{c}{\cup} K(y_k, j)$:

  $K(y_h, j+1)$ contient $y_h, y_{h+1}, \ldots, y_{k'}, y_{k'}$ répondant à l'hypothèse.
- j+1 n'est pas sortie vraie de y_k : $\Omega(y_k) > j+1$.

  $K(y_h, j+1)$ contient $y_h, y_{h+1}, \ldots, y_k, y_k$ répondant à l'hypothèse.

JUSTIFICATION.

1°) Tous les chemins de K(x,i) sont élémentaires :

Cela résulte de la définition des opérations $\underset{c}{\vee}$ et $\underset{c}{\cup}$.

2°) Nous allons montrer par une récurrence sur l'indice i de l'élément x_i, que si $x_0, x_1, \ldots, x_n$ est un chemin élémentaire alors $K(x_0, m)$ contient ce chemin.

Appelons H_i l'hypothèse suivante :

Pour tout t compris entre i et n (n exclu)

$K(x_t, \mathcal{R}(x_t)-1)$ contient $x_t, x_{t+1}, \ldots, x_k$, x_k étant égal à x_n ou x_k étant élément fermeture de circuit dont la sortie vraie est supérieure à $\mathcal{R}(x_t)$.

Montrons que H_i entraîne H_{i-1}. Pour cela, il suffit de montrer que $K(x_{i-1}, \mathcal{R}(x_{i-1})-1)$ contient un chemin répondant à l'hypothèse.

a) Si l'on a $\mathcal{R}(x_i)$ supérieur à $\mathcal{R}(x_{i-1})$ alors $K(x_{i-1}, \mathcal{R}(x_{i-1})-1)$ contient $x_{i-1}x_i$ et il existe une sortie h de x_i inférieure à $\mathcal{R}(x_{i-1})$ donc inférieure à $\mathcal{R}(x_i)$: x_i est élément fermeture de circuit.

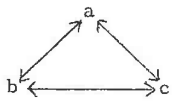
b) $\mathcal{R}(x_i)$ inférieure à $\mathcal{R}(x_{i-1})$. Il existe une sortie h de x_i inférieure à $\mathcal{R}(x_{i-1})$ telle que x_{i-1} est sommet de v_h. Si h est inférieur à la sortie vraie de x_i alors à l'instant $\mathcal{R}(x_{i-1})$, x_i appartient à la pile et $\mathcal{R}(x_i) > \mathcal{R}(x_{i-1})$ ce qui est contraire à (b). On a donc $\mathcal{R}(x_i) \leqslant h < \mathcal{R}(x_{i-1})$.

L'hypothèse H_i étant la même que l'hypothèse $G_{\mathcal{R}(x_i)-1}$ pour le chemin élémentaire $x_i, x_{i+1}, \ldots, x_n$, h-1 supérieur ou égal à $\mathcal{R}(x_i)-1$ entraîne d'après le lemme :

$K(x_i, h-1)$ contient $x_i, x_{i+1}, \ldots, x_{k'}, x_{k'}$ répondant à l'hypothèse. A l'instant h : $K(x_i, h) = K(x_{i-1}, h-1) \underset{c}{\cup} K(x_i, h-1)$ d'où :

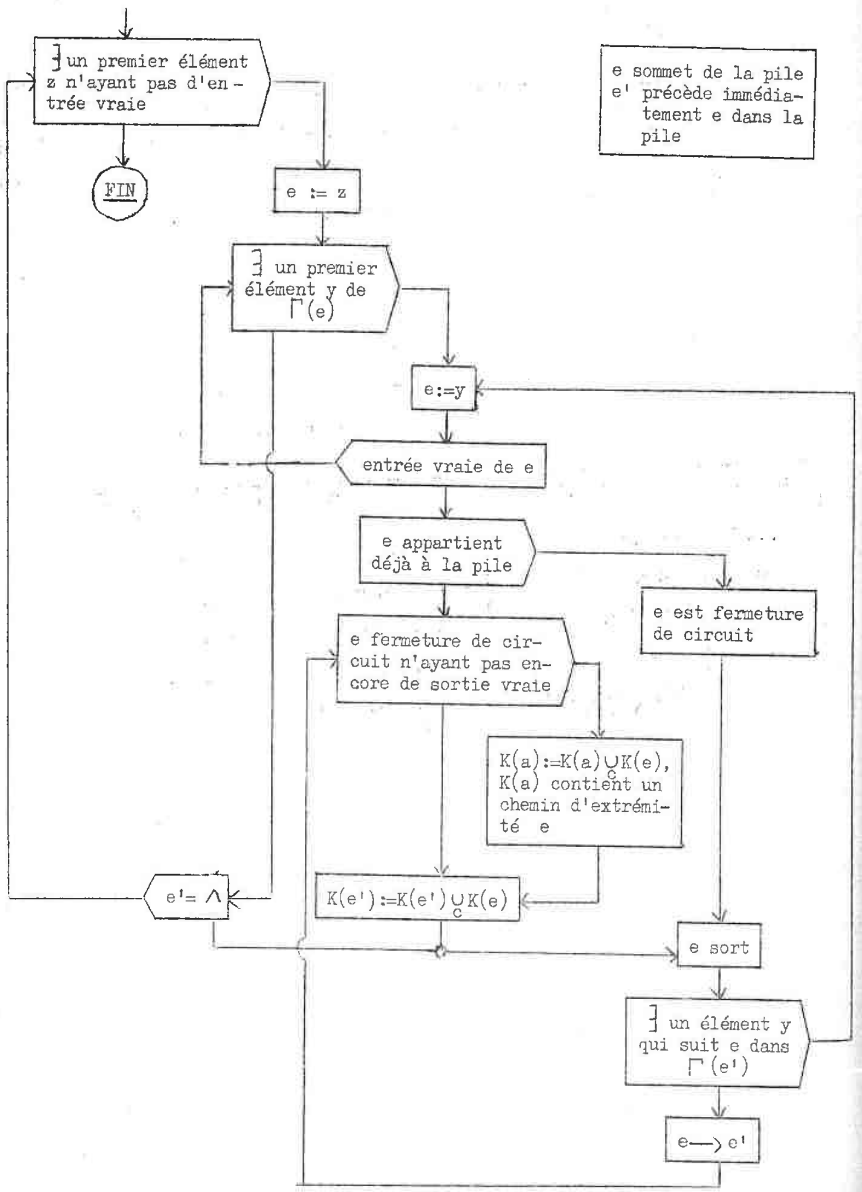
$K(x_{i-1}, h)$ contient un chemin répondant aux hypothèses. On a alors l'hypothèse G_h pour le chemin élémentaire $x_{i-1}, x_i, \ldots, x_{k'}$. A l'instant $\mathcal{R}(x_{i-1})-1 \geqslant h$, $K(x_{i-1}, \mathcal{R}(x_{i-1})-1)$ contient le chemin $x_{i-1}, x_i, \ldots, x_{k''}$ avec $x_{k''} = x_n$ ou $x_{k''}$ fermeture de circuit dont la sortie vraie est supérieure à $\mathcal{R}(x_{i-1})-1$ donc à $\mathcal{R}(x_{i-1})$. Or, H_{n-1} est vérifiée puisque $K(x_{n-1}, \mathcal{R}(x_{n-1})-1)$ contient $x_{n-1}x_n$. On a donc l'hypothèse $G_{\mathcal{R}(x_0)-1}$ pour le chemin élémentaire $x_0, x_1, \ldots, x_q$. D'après le lemme : $K(x_0, m)$ contient le chemin : $x_0, x_1, \ldots, x_{q'}$ avec $x_{q'} = x_n$ ou $x_{q'}$ est élément fermeture de circuit dont la sortie vraie est supérieure à m. Or, tous les éléments ont une sortie vraie inférieure ou égale à m $\Longrightarrow x_{q'} = x_n$.

Exemple : Nous donnons ci-dessous les états de la pile annexe et le contenu des ensembles K(x,i) pour le graphe ci-contre. Les éléments fermeture de circuit dont repérés dans la pile par le signe - .



						a		b						
		a			c	c	c	c	c					
	b	b	b	b	b	b	b	b	-b	-b		c		
a	a	a	-a	-a	-a	-a	-a	-a	-a	-a	-a	-a	-a	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Instant	K(a,i)	K(b,i)	K(c,i)
0 ... 9	ab,ac	ba,bc	ca,cb
10	ab,ac	ba,bc,bca	ca,cb
11 12	ab,abc,ac	ba,bc,bca	ca,cb,cba
13	ab,abc,ac,acb	ba,bc,bca	ca,cb,cba
14	ab,abc,ac,acb	ba,bac,bc,bca	ca,cab,cb,cba



PROGRAMMATION.

Lors de la programmation, nous avons rencontré deux problèmes principaux : la mémorisation des chemins élémentaires et la réalisation de l'opération $\cup_c$, le second problème dépendant essentiellement du premier. Nous nous sommes attachés à la recherche du moyen le plus économique de "stocker" les résultats.

Représentation des ensembles K(x,i).

Nous avons tout d'abord remarqué qu'il n'était pas utile de conserver en mémoire tous les chemins élémentaires, mais uniquement les chemins maximaux, ces premiers s'en déduisant aisément.

Lors de la recherche de la fermeture transitive stricte, les ensembles K(x,i) comportaient au plus n éléments occupant chacun une mémoire, ce qui donnait un encombrement de n^2 mémoires (n étant l'ordre du graphe). Ici, chaque ensemble K(x,i) comporte au plus (n-1) . (n-2) ... 1 chemins élémentaires maximaux de longueur n, ce qui représente n . n ! mémoires pour les n ensembles K(x,i). Pour la fermeture transitive, nous disposions d'environ 2000 mémoires pour les résultats, ce qui permettait de traiter des graphes d'ordre 40 à 45. En supposant que pour ce programme nous disposions du même nombre de mémoires (ce qui se révèlera approximativement exact), nous ne pouvons traiter que des graphes d'ordre 5 : (6 . 6 ! = 4320). Il n'est donc pas question de faire une réservation statique. Mais le compilateur ALGOL que nous possédons n'accepte pas les variables rémanentes, il est donc impossible de faire une réservation dynamique. Il nous faut trouver un mode de réservation statique qui nous permette de mémoriser les chemins élémentaires maximaux sous la forme la plus condensée possible. La méthode des suites espacées par un séparateur et rangées dans un tableau à une dimension se révèle lourde, tant pour la programmation de l'opération $\cup_c$, que pour la simulation de la réservation dynamique. D'autre part, si deux chemins élémentaires maximaux ne diffèrent que par leur extrémité, nous sommes tenus de ranger en mémoire deux suites de longueur $l+1$

dont la "partie intéressante" ne représente qu'une fraction de chaque suite. Enfin, les ensembles K(x,i) sont essentiellement variables au cours du programme, ce qui nécessite des transferts de suites longs et peu élégants.

Ce sont ces trois raisons primordiales qui nous ont conduits à représenter les ensembles K(x,i) par des arbres de racine x, mémorisés sous la forme de leur matrice d'enchaînement.

Matrice d'enchaînement [IVERSON].

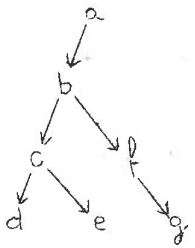
Les éléments de chaque famille $\Gamma(x)$ étant rangés dans l'ordre implicite des codes, la matrice d'enchaînement se définit comme suit :

C'est une matrice à trois colonnes dont le nombre de lignes est égal au nombre de sommets de l'arbre. Dans la première colonne se trouve le nom du sommet (ou plus exactement son code), dans la seconde un lien vertical, dans la troisième un lien horizontal ainsi définis :

Un élément x possède un lien vertical si, et seulement si, $\Gamma(x)$ n'est pas vide. Dans ce cas, le lien vertical est l'indice de la ligne où se trouve le premier élément de $\Gamma(x)$.

Un élément x appartenant à $\Gamma(y)$ possède un lien horizontal si, et seulement si, x n'est pas le dernier élément de $\Gamma(y)$. Dans ce cas, le lien horizontal est l'indice de la ligne où se trouve l'élément qui suit x dans $\Gamma(y)$.

Exemple.



L'arbre ci-contre a pour matrice d'enchaînement la matrice suivante :

indice de la ligne	nom du sommet	lien vertical	lien horizontal
1	a	2	
2	b	3	
3	c	4	6
4	d		5
5	e		
6	f		7
7	g		

Réalisation de l'opération $\underset{c}{\cup}$ sur les matrices d'enchaînement.

Si on représente chaque ensemble K(x,i) par sa matrice d'enchaînement, on se trouve de nouveau devant un problème de réservation des matrices. Pour éviter ceci, nous rangeons toutes les matrices dans la même et faisons une réservation maximale pour cette dernière. A l'instant origine, nous avons dans les n premières lignes les n éléments, origine des chemins élémentaires.

1°) Sortie d'un élément qui n'est pas fermeture de circuit :

Nous faisons une concaténation simple de l'élément x et des chemins d'origine z (opération $K(x,i) = K(x,i-1) \underset{c}{\cup} K(z,i-1)$). En effet, x n'étant pas fermeture de circuit aucun chemin d'origine z ne peut contenir x. Nous nous trouvons en face d'un simple problème de transfert d'un certain nombre de lignes de la matrice dans une partie non encore utilisée de celle-ci. Les noms des sommets se translatent sans changement alors que les liens verticaux et horizontaux doivent évoluer pour s'appliquer aux nouvelles lignes de la matrice. Nous reviendrons sur ce problème après avoir vu les cas suivants.

2°) Sortie d'un élément fermeture de circuit :

Nous devons faire deux opérations :

- La première $K(a,i) = K(a,i-1) \underset{c}{\cup} K(z,i-1)$,
- La seconde $K(x,i) = K(x,i-1) \underset{c}{\cup} K(z,i-1)$.

Pour la première, nous faisons réellement une opération de concaténation contractée élémentaire en testant successivement chacun des sommets de K(z,i-1) par rapport aux sommets du chemin d'origine a, d'extrémité z. Nous renouvelons cette opération pour tous les chemins de K(a,i-1) d'extrémité z.

Pour la seconde opération, nous faisons une concaténation simple analogue à celle de 1°) en testant de plus chacun des sommets des chemins de K(z,i-1) par rapport à x.

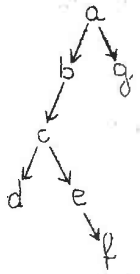
Nous rencontrons de nouveau deux problèmes :

- La translation d'une partie de la matrice d'enchaînement avec évolution des liens,
- Les tests sur les éléments et la suppression des lignes correspondantes lors de la translation.

Translation sans test.

Elle se fait ligne par ligne en "parcourant" la partie de matrice intéressée. Cette opération n'est possible que si tout élément possède un lien horizontal ou vertical, car il n'est pas possible de "remonter" les lignes de la matrice.

Exemple :

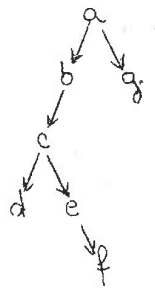


Lorsque l'on a "parcouru" la matrice jusqu'au sommet f, f ne possèdant ni lien horizontal ni lien vertical, on ne peut aller en g. (dans cet exemple, les lignes de la matrice se suivent dans l'ordre des sommets, mais il n'en est plus ainsi dans le cas général après de nombreuses translations).

Pour remédier à cette impossibilité, nous avons donné à tout élément sans lien un pseudo-lien horizontal indice de la ligne du premier élément (s'il existe, sinon le "parcours" est terminé) possèdant un lien horizontal. Ce pseudo-lien se distingue du lien véritable par le fait qu'il est négatif. Dans ce qui suit, nous noterons M(i), V(i) et H(i) le nom de l'élément, le lien vertical et le lien ou pseudo-lien horizontal de la ligne i.

Nous donnons la matrice d'enchaînement et les indices successifs de ligne lors du "parcours" de la matrice.

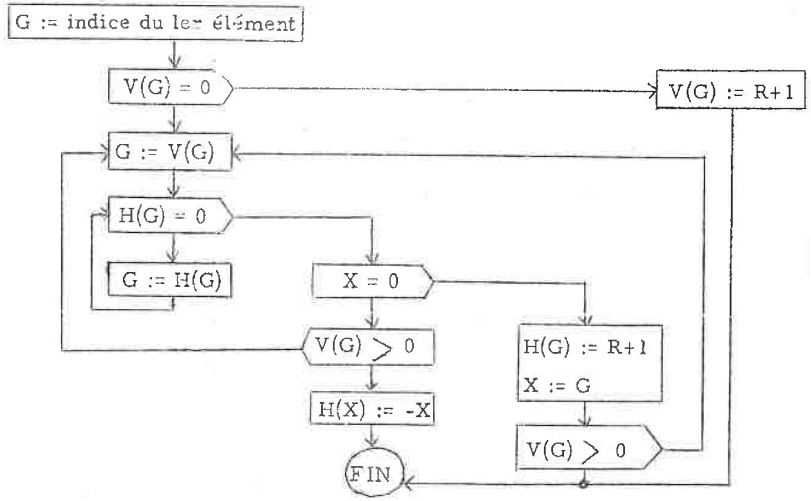
Exemple :



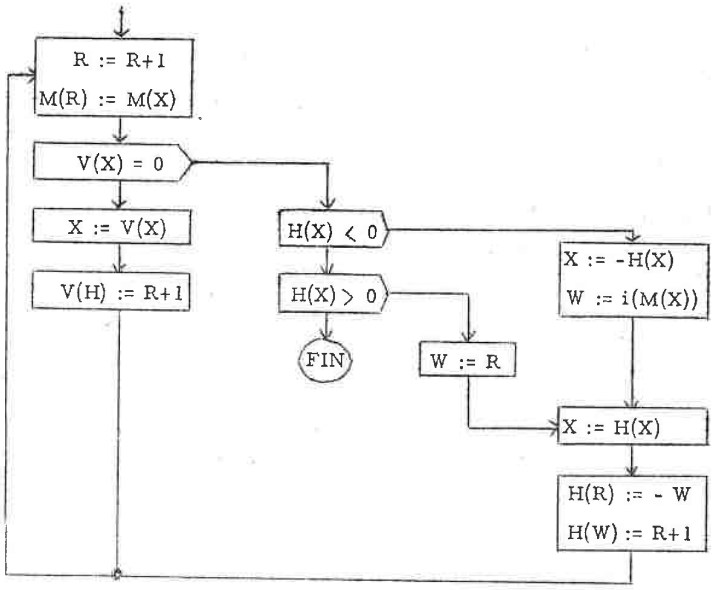
i	M(i)	V(i)	H(i)
1	a	2	
2	b	3	7
3	c	4	
4	d		5
5	e	6	
6	f		-2
7	g		

Indices successifs : 1 2 3 4 5 6 (-2) 7.

Lorsqu'à la sortie d'un élément z nous ajoutons à K(x,i) chemins x, z, ..., nous devons "raccorder" les chemins déjà existants aux nouveaux chemins par un lien ou pseudo-lien horizontal. Pour cela, nous devons trouver le dernier élément y de $\Gamma(x)$, lui mettre comme lien horizontal l'indice R de la première ligne libre de la matrice. S'il n'est pas le dernier élément de la matrice d'origine x, on cherche celui-ci, et on lui met le pseudo-lien horizontal suivant : l'indice de la ligne de y. L'organigramme correspondant à ce "raccord" est le suivant : (X est un indicateur permettant de connaître l'indice de y).



Nous sommes alors en mesure de faire la translation proprement dite, dont l'organigramme est le suivant :



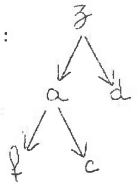
X est la variable d'indice relative à la matrice K(z,i-1),
R est la dernière ligne occupée de la matrice d'enchaînement.

Pour translater le pseudo-lien horizontal, on recherche dans les lignes translatées (en partant de la ligne R) l'indice W de l'élément correspondant au pseudo-lien. C'est cette opération qui est symbolisée par l'affectation :

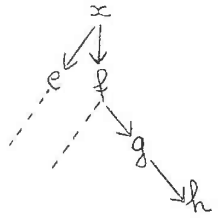
W := i (M(X)) puis H(R) := - W .

Exemple :

K(z,i) :



K(x,i) :



La matrice d'enchaînement avant translation est la suivante :

i	M(i)	V(i)	H(i)
j	x	j+1	
j+1	e	...	k
k	f	ℓ	
ℓ	g	ℓ+1	
ℓ+1	h		
ℓ+2	z	ℓ+3	
ℓ+3	a	ℓ+4	ℓ+6
ℓ+4	b		ℓ+5
ℓ+5	c		-(ℓ+3)
ℓ+6	d		

Après translation :

i	M(i)	V(i)	H(i)	Observations
j	x	j+1		
j+1	e	...	k	
k	f	ℓ	R+1	lien de "raccor
ℓ	g	ℓ+1		
ℓ+1	h		-k	pseudo-lien de "raccord"
ℓ+2	z	ℓ+3		
ℓ+6	d			
R+1	z	R+2		
R+2	a	R+3	R+5	partie ajoutée
R+3	b		R+4	
R+4	c		-(R+2)	
R+5	d			

Translation avec tests.

Les éléments sur lesquels doivent porter les tests sont rangés dans une pile. La procédure TEST(A) permet de faire successivement chacun des tests sur les éléments de la pile. Cette pile a été choisie comme partie haute de la pile annexe pour gagner le maximum de place en mémoire (le nombre d'éléments total n'étant jamais supérieur à n+2).

Nous allons maintenant envisager les différents cas dans lesquels nous aurons à faire un test :

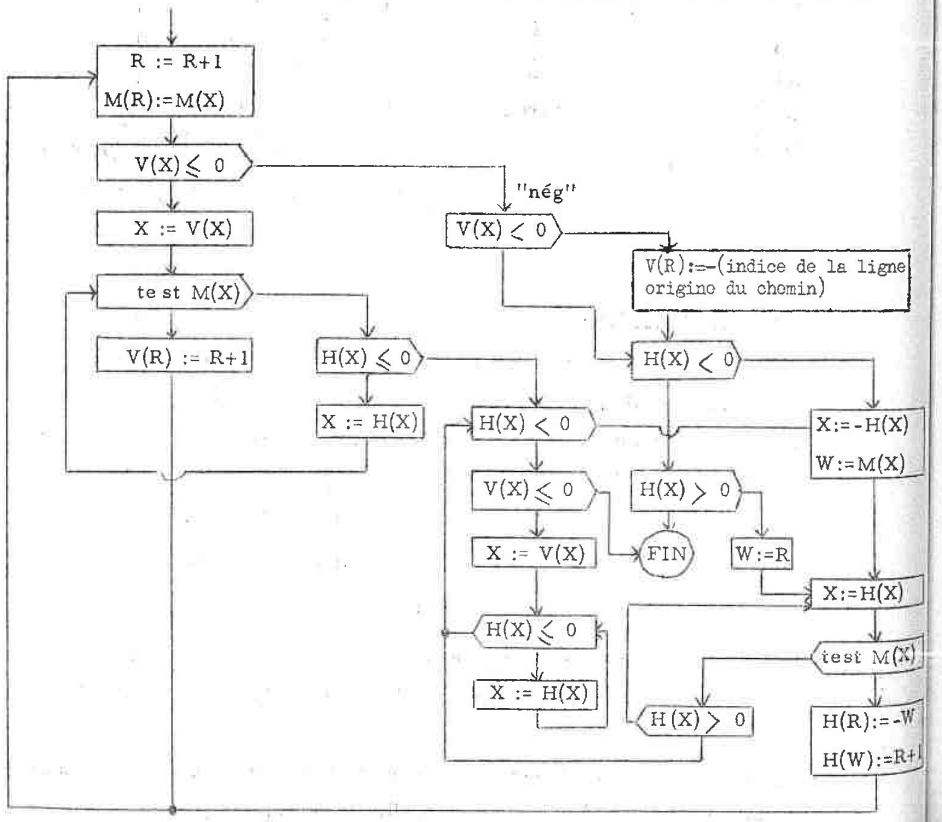
Nous supposerons que l'élément a est déjà translaté :

a possède un lien vertical et l'élément b correspondant se trouve déjà dans le chemin :

- b possède un lien horizontal correspondant à l'élément c. Tout se passe comme si on avait l'indice de c pour lien vertical.

- b ne possède ni lien ni pseudo-lien horizontal. Alors on "parcourt" la matrice à partir de b jusqu'à rencontre du dernier élément précédant b. Si cet élément n'a pas de lien horizontal, nous avons terminé la translation, si cet élément a un pseudo-lien horizontal, nous sommes conduits au cas suivant :
- b possède un pseudo-lien horizontal. On recherche l'élément c qui suit b dans la matrice d'enchaînement, et on est amené au cas suivant :
- a possède un lien horizontal et l'élément correspondant b se trouve déjà dans le chemin : nous sommes amenés à l'un des trois cas précédents.

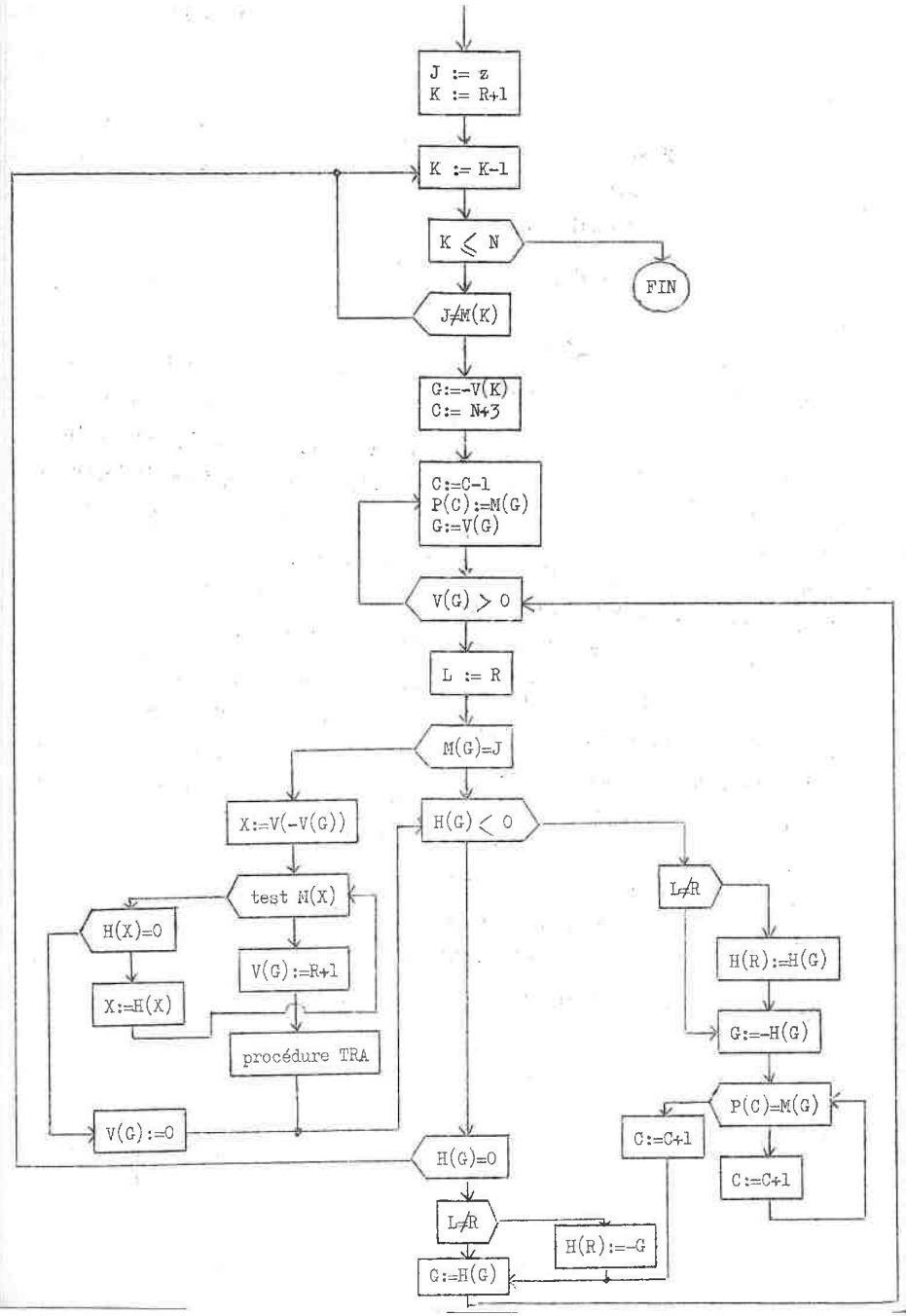
L'organigramme de translation avec tests est le suivant :



C'est cette translation avec test mise sous forme de procédure TRA qui représente l'opération $\bigcup_c$ (les tests sont sautés automatiquement lorsqu'ils sont inutiles).

Lors de l'opération $K(a,i) = K(a,i-1) \bigcup_c K(z,i-1)$, nous devons faire cette translation avec tests pour tous les chemins d'extrémité z. Il est facile de trouver les éléments z extrémité de chemin dans la matrice d'enchaînement : ce sont ceux qui n'ont pas de lien vertical. Mais pour pouvoir faire les tests et par conséquent ranger dans le haut de la pile annexe les sommets du chemin, il faut en connaître son origine. Nous indiquons l'indice de la ligne d'origine du chemin par un pseudo-lien vertical qui se distingue du lien véritable parce qu'il est négatif. Ce pseudo-lien vertical n'est mis qu'aux éléments fermeture de circuit extrémité d'un chemin. Lors de toute translation, ce pseudo-lien doit évoluer. On réalise ceci par un test du lien dans la branche "neg" de l'organigramme translation avec test.

La recherche des chemins élémentaires d'extrémité z se fait alors en recherchant les éléments z ayant un pseudo-lien vertical, ce qui donne l'origine a du chemin. Puis, en "parcourant" l'arbre dont cet élément est origine, on range dans le haut de la pile annexe les sommets successifs. Quand on rencontre un élément z ayant un pseudo-lien vertical, on supprime la ligne de z et on passe par la procédure TRA. Lorsque tous les chemins de a ont été analysés, on cherche de nouveau s'il existe des éléments z ayant un pseudo-lien vertical. L'organigramme correspondant à cette "résolution" est le suivant :

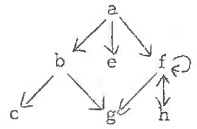


Pour restreindre l'encombrement des mémoires, nous nous donnons le graphe sous une autre forme que sa matrice associée : en effet, dans notre compilateur, le booléen occupe la même place que l'entier, soit une mémoire. Il est donc plus économique de se donner le graphe par les éléments de chaque famille $\Gamma(x)$. Pour cela, nous rangeons dans un tableau tous les éléments des différents ensembles $\Gamma(x)$ dans l'ordre de codage implicite des éléments. Un second tableau nous permet de connaître pour tout x la position (par son indice) du premier élément de $\Gamma(x)$.

Le passage de la forme matrice associée, à la forme donnée ci-dessus se fait par un petit programme ALGOL indépendant du programme principal. Nous profitons de cette transformation pour supprimer les boucles (circuits de longueur unité) du graphe qui n'interviennent pas dans la recherche des chemins élémentaires.

Lors de l'entrée du premier élément y de $\Gamma(x)$, son emplacement dans le tableau sert de compteur d'entrée pour les éléments z qui suivent y dans $\Gamma(x)$. Lors de la sortie vraie de x, ce compteur est forcé à zéro. Pour ne pas confondre les éléments de $\Gamma(x)$ avec le compteur, nous donnons à ce dernier des valeurs négatives.

Exemple : Les deux tableaux associés au graphe ci-dessous sont les suivants :



	1	2	3	4	5	6	7	8
D	b	e	f	c	g	g	h	f

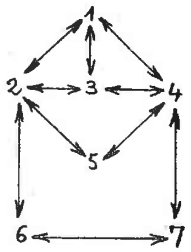
	a	b	c	e	f	g	h
T	1	4	6	6	6	8	8	9

Programme Algol.

Pour économiser la place, nous avons mis les deux premières colonnes de la matrice d'enchaînement dans le même tableau MV, ce qui explique les deux fonctions procédures M(X), V(X) permettant de restituer l'élément et son lien vertical à partir d'une seule mémoire.

Le programme généré occupe environ 1250 mémoires, ce qui laisse environ 2100 mémoires pour les données et les résultats.

Nous donnons un exemple de résultat obtenu pour un graphe d'ordre 7 symétrique qui nous a demandé environ le tiers de la capacité maximum.



GRAPHE D'ORDRE 7
CHEMINS ELEMENTAIRES MAXIMAUX

1 2 3 4 5
1 2 3 4 7 6
1 2 5 4 3
1 2 5 4 7 6
1 2 6 7 4 3
1 2 6 7 4 5
1 3 2 5 4 7 6
1 3 2 6 7 4 5
1 3 4 5 2 6 7
1 3 4 7 6 2 5
1 4 3 2 5
1 4 3 2 6 7
1 4 5 2 3
1 4 5 2 6 7
1 4 7 6 2 3
1 4 7 6 2 5
2 1 3 4 5
2 1 3 4 7 6
2 1 4 3
2 1 4 5
2 1 4 7 6
2 3 1 4 5
2 3 1 4 7 6
2 3 4 1
2 3 4 5
2 3 4 7 6
2 5 4 1 3
2 5 4 3 1
2 5 4 7 6
2 6 7 4 1 3
2 6 7 4 3 1
2 6 7 4 5

3 1 2 5 4 7 6
3 1 2 6 7 4 5
3 1 4 5 2 6 7
3 1 4 7 6 2 5
3 2 1 4 5
3 2 1 4 7 6
3 2 5 4 1
3 2 5 4 7 6
3 2 6 7 4 1
3 2 6 7 4 5
3 4 1 2 5
3 4 1 2 6 7
3 4 5 2 1
3 4 5 2 6 7
3 4 7 6 2 1
3 4 7 6 2 5
4 1 2 3
4 1 2 5
4 1 2 6 7
4 1 3 2 5
4 1 3 2 6 7
4 3 1 2 5
4 3 1 2 6 7
4 3 2 1
4 3 2 5
4 3 2 6 7
4 5 2 1 3
4 5 2 3 1
4 5 2 6 7
4 7 6 2 1 3
4 7 6 2 3 1
4 7 6 2 5

5 2 1 3 4 7 6
5 2 1 4 3
5 2 1 4 7 6
5 2 3 1 4 7 6
5 2 3 4 1
5 2 3 4 7 6
5 2 6 7 4 1 3
5 2 6 7 4 3 1
5 4 1 2 3
5 4 1 2 6 7
5 4 1 3 2 6 7
5 4 3 1 2 6 7
5 4 3 2 1
5 4 3 2 6 7
5 4 7 6 2 1 3
5 4 7 6 2 3 1
6 2 1 3 4 5
6 2 1 3 4 7
6 2 1 4 3
6 2 1 4 5
6 2 1 4 7
6 2 3 1 4 5
6 2 3 1 4 7
6 2 3 4 1
6 2 3 4 5
6 2 3 4 7
6 2 5 4 1 3
6 2 5 4 3 1
6 2 5 4 7
6 7 4 1 2 3
6 7 4 1 2 5
6 7 4 1 3 2 5
6 7 4 3 1 2 5
6 7 4 3 2 1
6 7 4 3 2 5
6 7 4 5 2 1 3
6 7 4 5 2 3 1

7 4 1 2 3
7 4 1 2 5
7 4 1 2 6
7 4 1 3 2 5
7 4 1 3 2 6
7 4 3 1 2 5
7 4 3 1 2 6
7 4 3 2 1
7 4 3 2 5
7 4 3 2 6
7 4 5 2 1 3
7 4 5 2 3 1
7 4 5 2 6
7 6 2 1 3 4 5
7 6 2 1 4 3
7 6 2 1 4 5
7 6 2 3 1 4 5
7 6 2 3 4 1
7 6 2 3 4 5
7 6 2 5 4 1 3
7 6 2 5 4 3 1

A N N E X E

```
'DEBUT' 'ENTIER' I , N ;
DEB:    EXL(&#GRAPHE_D'ORDRE$@) ; IMPR ; LICLAV(N) ;

'DEBUT' 'ENTIER' 'TABLEAU' R.(1:N,1,N). , S.(1:N). , P.(1:N+1). ;
        LIRT(R) ;
        'POUR'I:=1'PAS'1'JUSQUA'N'FAIRE'S.(I).:=0 ;

'DEBUT' 'ENTIER' 'PROCEDURE' VALEUR(I) ; 'ENTIER' I ;
                 VALEUR:='SI'P.(I).'SUP'99000'ALORS'
                 P.(I).-99000'SINON'
                 'SI'P.(I).'SUP'90000'ALORS'
                 P.(I).-90000'SINON'P.(I). ;
        'ENTIER' C , D , B , G , H , J , K , L ;
        B:=C:=D:=0 ; P.(1).:=I:=1 ; G:=1 ;
        'SI'CLE(6)'ALORS''DEBUT'
        EXL(&#MATRICE_ASSOCIEE$@) ; IMPR ; 'ALLERA' SORT 'FIN' ;
DEBCL:  K:=VALEUR(I) ;
        'SI'P.(I).'SUP'99000'ALORS''ALLERA' BOUSOR ;
        J:=I-1 ;
        'SI'S.(K).'EGAL'-1'ALORS''ALLERA' BOENF ;
        'SI'S.(K).'INF'0'ALORS''ALLERA' TRANS ;
        'SI'S.(K).'DIF'0'ET'L'EGAL'1'ALORS''ALLERA' BOUENT ;
        'POUR'G:=S.(K).+1'PAS'1'JUSQUA'N'FAIRE'
        'SI'R.(K,G).'EGAL'1'ALORS''ALLERA' PROGI ;
        'SI'S.(K).'DIF'0'ALORS''ALLERA' SORTIE ;
        S.(K).:=-1 ;
        'SI'I'EGAL'1'ALORS''ALLERA' EXFIN ; 'ALLERA' BOENF ;
PROGI:  S.(K).:=G ; I:=I+1 ; P.(I).:=G ; L:=1 ;
        'SI'P.(I).'DIF'K'ALORS''ALLERA' DEBCL ; C:=C+1 ;
        EXL(&#BOUCLE$@) ; EXE(4,C) ; IMPR ;
        EXE(4,K) ; IMPR ; 'ALLERA' BOENF ;
BOUENT: 'POUR'G:=I-1'PAS'-1'JUSQUA'1'FAIRE''DEBUT'
        'SI'P.(I).'EGA'VALEUR(G)'ALORS''ALLERA' BOENF ;
        'SI'P.(G).'INF'90000'ALORS'
        P.(G).:=P.(G).+90000 'FIN' ; IMPR ;
BOENF:  L:=0 ; I:=I-1 ; 'ALLERA' DEBCL ;
SORTIE: 'SI'P.(I).'INF'90000'ALORS''ALLERA' SU3 ;
        'POUR'J:=I-1'PAS'-1'JUSQUA'1'FAIRE'
        'SI'P.(J).'INF'99000'ALORS''ALLERA' TESTSOM ; IMPR ;
```

```
TESTSOM:'SI'P.(J).'INF'90000'ALORS''ALLERA' TESTFC ;
        H:=P.(J). ; P.(J).:=P.(I).+9000 ;
        P.(I).:=H ; 'ALLERA' DEBCL ;
SU3:    'SI'D'EGA'0'ALORS''ALLERA' BOUSOR ;
        'POUR'G:=I+1'PAS'1'JUSQUA'D'FAIRE''DEBUT'
        B:=B+1 ;
        'POUR'H:=1'PAS'1'JUSQUA'N'FAIRE'
        R.(VALEUR(G),H).:=R.(VALEUR(I),H). 'FIN' ;
        D:=0 ; 'ALLERA' BOUSOR ;
RACINE: P.(1).:=G ; 'ALLERA' DEBCL ;
TESTFC: 'POUR'G:=S.(VALEUR(J)).+1'PAS'1'JUSQUA'N'FAIRE'
        'SI'R.(VALEUR(J),G).'EGAL'1'ALORS''DEBUT'
        K:=P.(J). ; 'ALLERA' PROGI 'FIN' ;
        D:=I ; C:=C+1 ;
        EXL(&#CIRCUIT$@) ; EXE(4,C) ; IMPR ;
        'POUR'G:=J'PAS'1'JUSQUA'I'FAIRE''DEBUT'
        EXE(4,VALEUR(G)) ;
        'SI'(G-J+1)%20"20'EGAL'G-J+1'ALORS'IMPR 'FIN' ;
        IMPR ; 'ALLERA' BOUSOR ;
BOUSOR: S.(K).:=-10 ; 'SI'I'EGAL'1'ALORS''ALLERA' EXFIN ;
TRANS:  B:=B+1 ; 'POUR'G:=1'PAS'1'JUSQUA'N'FAIRE'
        'SI'R.(K,G).'SUG'1'ALORS'R.(VALEUR(J),G).:=2 ;
        'ALLERA' BOENF ;
EXFIN:  'POUR'G:=1'PAS'1'JUSQUA'N'FAIRE'
        'SI'S.(G).'SUG'0'ALORS''ALLERA' RACINE ;
        EXE(8,B) ; IMPR ;
        EXL(&#MATRICE_DE_FERMETURE_TRANSITIVE!$@) ; IMPR ;
SORT:   'POUR'G:=1'PAS'1'JUSQUA'N'FAIRE''DEBUT'
        EXE(3,G) ; ESPACE(2) ;
        'POUR'J:=1'PAS'1'JUSQUA'N'FAIRE'
        'SI'R.(G,J).'EGAL'0'ALORS'EXL(&0_@)'SINON'EXL(&1_@) ;
        IMPR 'FIN' ;
        'SI''NON'CLE(6)'ALORS''ALLERA' DEB ;
        PAUSE(1) ; 'ALLERA' DEBCL ;
'FIN'

'FIN'

'FIN'#
```



```
'DEBUT' 'ENTIER' N , I , J , K , C ;
DEB:    LICLAV(N) ;

'DEBUT' 'ENTIER''TABLEAU' M.(1:N,1:N). ;
        LIRT(M) ; K:=1 ;
        'SI'CLE(6)'ALORS''DEBUT'
        EXL(&#MATRICE_ASSOCIEE$@) ; IMPR ;
        'ALLERA' PF 'FIN' ;
        C:=0 ;
CA:     'POUR'I:=1'PAS'1'JUSQUA'N'FAIRE'
        'POUR'J:=1'PAS'1'JUSQUA'N'FAIRE'
        'SI'M.(J,I).'DIF'0'ALORS''DEBUT'
        C:=C+1 ;
        'POUR'K:=1'PAS'1'JUSQUA'N'FAIRE'
        M.(J,K).:=M.(J,K).+M.(I,K). 'FIN' ;
        EXE(8,C) ; IMPR ;
        EXL(&#MATRICE_DE_FERMETURE_TRANSITIVE$@) ;
        IMPR ; K:=0 ;
PF:     'POUR'I:=1'PAS'1'JUSQUA'N'FAIRE''DEBUT'
        EXE(3,I) ; ESPACE(1) ;
        'POUR'J:=1'PAS'1'JUSQUA'N'FAIRE'
        'SI'M.(I,J).'EGAL'0'ALORS'EXL(&_0@)'SINON'EXL(&_1@) ;
        IMPR 'FIN' ;
        'SI'K'DIF'0'ALORS''ALLERA' CA ;
        'ALLERA' DEB
'FIN'

'FIN'#
```

```
'DEBUT' 'ENTIER' X , N , MA , C , W , I , J , K , G , L , R ;
ET:     LIRE(N,L,MA) ;

'DEBUT' 'ENTIER''TABLEAU' P.(1:N+2). , T.(1:N+1). , D.(1:L). ,
        MV , H.(1:MA). ;
        'ENTIER''PROCEDURE' V(X) ;
                'ENTIER' X ; V:=MV.(X).%100 ;
        'ENTIER''PROCEDURE' M(X) ;
                'ENTIER' X ; M:=MV.(X).-100*V(X) ;
        'PROCEDURE' TEST(A) ; 'ETIQUETTE' A ;
                'DEBUT' 'ENTIER' J ;
                'POUR'J:=C'PAS'1'JUSQUA'N+2'FAIRE'
                'SI'ABS(M(X))'EGAL'P.(J).'ALORS'
                'ALLERA' A 'FIN' ;
        'PROCEDURE' TRA ;
                'DEBUT'
        TD:     R:=R+1 ; MV.(R).:=M(X) ; H.(R).:=0 ;
                'SI'V(X)'ING'0'ALORS''DEBUT'
                'SI'MV.(X).'INF'0'ALORS'
                MV.(R).:=MV.(R).-P.(N+2).*100 ;
                'ALLERA' TP 'FIN' ;
                X:=V(X) ;
        TF:     TEST(TK) ; MV.(R).:=MV.(R).+100*(R+1) ;
                'ALLERA' TD ;
        TK:     'SI'H.(X).'SUP'0'ALORS''DEBUT'
                X:=H.(X). ; 'ALLERA' TF 'FIN' ;
        TG:     'SI'H.(X).'INF'0'ALORS''ALLERA' TE ;
                'SI'V(X)'ING'0'ALORS''ALLERA' FINTRA ; X:=V(X) ;
        TH:     'SI'H.(X).'ING'0'ALORS''ALLERA' TG ;
                X:=H.(X). ; 'ALLERA' TH ;
        TP:     'SI'H.(X).'EGAL'0'ALORS''ALLERA' FINTRA ;
                'SI'H.(X).'INF'0'ALORS''ALLERA' TE ; W:=R ;
        TL:     X:=H.(X). ; TEST(TZ) ; H.(R).:=-W ;
                H.(W).:=R+1 ; 'ALLERA' TD ;
        TZ:     'ALLERA''SI'H.(X).'ING'0'ALORS' TG 'SINON' TL ;
        TE:     X:=-H.(X). ;
                'POUR'W:=R'PAS'-1'JUSQUA'1'FAIRE'
                'SI'M(X)'EGAL'M(W)'ALORS''ALLERA' TL ;
        FINTRA: 'FIN' ;
```



```
                LIRT(D) ; LIRT(T) ; R:=N ;
                'POUR'I:=1'PAS'1'JUSQUA'N'FAIRE'MV.(I).:=I ;
                I:=MA:=1 ; L:=0 ;
        EXFIN:  'POUR'MA:=MA'PAS'1'JUSQUA'N'FAIRE'
                'SI'T.(MA).'DIF'T.(MA+1).'ALORS''DEBUT'
                'SI'D.(T.(MA).).'SUP'0'ALORS''ALLERA' RACINE 'FIN' ;
                'ALLERA' CHEMINS ;
        RACINE: P.(1).:=MA ;
        DEBCL:  K:=ABS(P.(I).) ; X:=T.(K). ;
                'SI'X'EGAL'T.(K+1).'ALORS''ALLERA' TRAN ; G:=D.(X). ;
                'SI'G'EGAL'0'ALORS''ALLERA' TRAN ;
                'SI'G'SUP'0'ALORS''DEBUT'
                D.(X).:=-X-1 ; P.(I+1).:=G ; 'ALLERA' DE 'FIN' ;
                'SI'L'DIF'0'ALORS''ALLERA' CIRENT ;
        PROG:   'SI'G'EGAL'-T.(K+1).'ALORS''ALLERA' TESTSOR ;
        ENTRE:  D.(X).:=G-1 ; P.(I+1).:=D.(-G). ;
        DE:     L:=I:=I+1 ; 'ALLERA' DEBCL ;
        CIRENT: 'POUR'G:=I-1'PAS'-1'JUSQUA'1'FAIRE'
                'SI'P.(I).'EGAL'P.(G).'ALORS'P.(G).:=-P.(G). ;
                'ALLERA' TRANS ;
        TESTSOR:D.(X).:=0 ; 'SI'P.(I).'INF'0'ALORS''ALLERA' RESOL ;
        TESTI:  'SI'I'EGAL'1'ALORS''ALLERA' EXFIN ;
        TRAN:   L:=0 ;
        TRANS:  G:=P.(N+2).:=ABS(P.(I-1).) ; X:=0 ;
                'SI'P.(I-1).'INF'0'ALORS'C:=N+2'SINON'C:=N+3 ;
                'SI'V(G)'EGAL'0'ALORS''DEBUT'
                MV.(G).:=MV.(G).+100*(R+1) ; 'ALLERA' TESTL 'FIN' ;
        TA:     G:=V(G) ;
        TB:     'SI'H.(G).'DIF'0'ALORS''DEBUT'
                G:=H.(G). ; 'ALLERA' TB 'FIN' ;
                'SI'X'EGAL'0'ALORS''ALLERA' TC ;
                'SI'V(G)'SUP'0'ALORS''ALLERA' TA ;
                H.(G).:=-X ; 'ALLERA' TESTL ;
        TC:     H.(G).:=R+1 ; X:=G ;
                'SI'V(G)'SUP'0'ALORS''ALLERA' TA ;
        TESTL:  'SI'L'DIF'0'ALORS''DEBUT'
                R:=R+1 ; MV.(R).:=-P.(I).-100*P.(N+2). ;
                H.(R).:=0 ; L:=0 ; I:=I-1 ; 'ALLERA' DEBCL 'FIN' ;
                X:=ABS(P.(I).) ; TRA ;
                I:=I-1 ; 'ALLERA' DEBCL ;
        RESOL:  J:=P.(I). ; K:=R+1 ;
        RA:     'POUR'K:=K-1'TANTQUE'K'SUP'N'FAIRE'
                'SI'J'EGAL'M(K)'ALORS''ALLERA' RB ; L:=0 ; 'ALLERA' TESTI ;
        RB:     G:=-V(K) ;
        RJ:     C:=N+3 ;
        RD:     'SI'V(G)'SUP'0'ALORS''DEBUT'
                C:=C-1 ; P.(C).:=M(G) ;
                G:=V(G) ; 'ALLERA' RD 'FIN' ;
                'SI'J'SUP'0'ALORS''ALLERA' PERFO ; L:=R ;
                'SI'M(G)'DIF'J'ALORS''ALLERA' RE ;
                X:=V(-J) ; MV.(G).:=-J ;
        RL:     TEST(RK) ; MV.(G).:=MV.(G).+100*(R+1) ; TRA ;
```

```
RE:       'SI'H.(G).'INF'0'ALORS''ALLERA' RF ;
          'SI'H.(G).'EGAL'0'ALORS''ALLERA' RA ;
          'SI'L'DIF'R'ALORS'H.(R).:=-G ;
RG:       G:=H.(G). ; 'ALLERA' RD ;
RK:       'SI'H.(X).'EGAL'0'ALORS''ALLERA' RE ;
          X:=H.(X). ; 'ALLERA' RL ;
RF:       'SI'L'DIF'R'ALORS'H.(R).:=H.(G). ;
RM:       G:=-H.(G). ;
RH:       C:=C+1 ; 'SI'P.(C-1).'DIF'M(G)'ALORS''ALLERA' RH ;
          'ALLERA' RG ;
CHEMINS:EXL(&#!GRAPHE_D'ORDRE$@) ; EXE(3,N) ; IMPR ;
          EXL(&#CHEMINS_ELEMENTAIRES_MAXIMAUX!$@) ; IMPR ; I:=0 ;
CA:       'POUR'I:=I+1'TANTQUE'I'ING'N'FAIRE''DEBUT'
          J:=G:=I ; 'ALLERA' RJ 'FIN' ;
          'ALLERA' ET ;
PERFO:    C:=C-1 ; P.(C).:=M(G) ;
          'POUR'J:=N+2'PAS'-1'JUSQUA'C'FAIRE'
          EXE(3,P.(J).) ; IMPR ;
          'SI'H.(G).'SUP'0'ALORS''DEBUT'
          C:=C+1 ; 'ALLERA' RG 'FIN' ;
          'SI'H.(G).'INF'0'ALORS''ALLERA' RM ; 'ALLERA' CA ;
'FIN'

'FIN' #
```

REFERENCES

[BERGE] Théorie des graphes et ses applications - Dunod -

[FLOYD] Syntactic Analysis and operator precedence - Journal ACM - Octobre 1963 - Pages 316-333 -

[IVERSON] A programming language - John Willey and Sons, Inc. New-York London - Page 126 -

[NOLIN] Traitement des données groupées - Institut Blaise Pascal - Page 94 -

[PAIR] Arbres, piles et compilation - Revue "Chiffres" - N° 3 -

[PAIR] Etude de la notion de pile ; Application à l'analyse syntaxique - Thèse -

[WARSHALL] A theorem on booleen matrices - Journal ACM - Janvier 1962 - Pages 11-12 -

Vu et Approuvé
Nancy, le
Le Doyen de la Faculté
des Sciences de NANCY,

M. AUBRY

Vu et Permis d'imprimer
Nancy, le
le Recteur :
Président du Conseil de
l'Université,

P. IMBS