UNIVERSITE DE NANCY I
U.E.R. DE MATHEMATIQUES

UTILISATION DES HYPERGRAPHES POUR L'ÉVALUATION AUTOMATIQUE DES TAILLES DE DONNÉES

REALISATION DES LOGICIELS CORRESPONDANTS

POUR LE PROJET REMORA

THÈSE

pour l'obtention
du doctorat de spécialité
de mathématiques appliquées
(informatique)

soutenue le 30 Septembre 1974
par Marie-Claude PORTMANN

Jury : Président : M. J. LEGRAS

Examinateurs : Mme C. ROLLAND
M. C. PAIR
M. J.C. DERNIAME

UNIVERSITE DE NANCY I
U.E.R. DE MATHEMATIQUES

UTILISATION DES HYPERGRAPHES POUR L'ÉVALUATION AUTOMATIQUE DES TAILLES DE DONNÉES

REALISATION DES LOGICIELS CORRESPONDANTS

POUR LE PROJET REMORA

THÈSE

pour l'obtention
du doctorat de spécialité
de mathématiques appliquées
(informatique)

soutenue le 30 Septembre 1974
par Marie-Claude PORTMANN

Jury : Président : M. J. LEGRAS

Examinateurs : Mme C. ROLLAND
M. C. PAIR
M. J.C. DERNIAME

Je remercie chaleureusement Monsieur le Professeur LEGRAS, guide bienveillant, qui m'a encouragée et conseillée tout au long de cette étude.

Ce travail a été réalisé sous la direction de Madame ROLLAND, dont j'ai mis la patience à rude épreuve et qui m'a amicalement dirigée et aidée, tout particulièrement pour la rédaction de ce document.

Je remercie vivement Messieurs les Professeurs PAIR et DERNIAME qui me font l'honneur de participer au jury.

Que soient aussi remerciés l'équipe de l'exploitation et l'atelier de perforation de l'I.U.C.A. qui m'ont toujours accueillie avec le sourire quelles que soient mes exigences.

Ma gratitude va également aux services de secrétariat et tout particulièrement à Madame ZANNAD et à Madame LECLERC pour le soin et la gentillesse dont elles ont fait preuve et qui ont permis la réalisation matérielle de cette thèse en un temps très court.

SOMMAIRE (*)

(*) Un plan détaillé est fourni au début de chaque partie

INTRODUCTION

Ce travail fait partie d'un ensemble : le "projet Remora" dont l'objectif est de concevoir et de réaliser un système de pilotage des projets informatiques.

Ce système a la particularité de s'appuyer sur un ensemble d'outils que nous qualifions "d'orthogonaux" parce qu'ils peuvent, suivant les cas, être utilisés isolément, indépendamment les uns des autres ou au contraire imbriqués selon les besoins et la volonté de l'utilisateur.

Les "outils de datamatique" que nous développons ici sont un de ces éléments. Sous l'hypothèse de description de données en format fixe, ils ont un double objectif :

- Lors de la création des descriptions de données, ils permettent d'attribuer des tailles aux mémoires intermédiaires de travail ;

- Lors de la maintenance, ils donnent à l'utilisateur les arguments de l'évaluation des impacts d'une modification à effectuer et réalisent de manière automatique les corrections des tailles de toutes les données "influencées".

Ces deux problèmes nous ont conduits à formaliser la notion de "liaison" entre les données d'un module de traitements en terme de complexe d'influences. En lui associant un hypergraphe orienté, nous montrerons que ces problèmes se ramènent à la recherche d'une loi de cheminement sur l'hypergraphe faisant appel à des règles d'évaluation de tailles.

Respectant l'objectif de l'équipe du projet Remora, nous avons réalisé les outils pour les tester sur des cas concrets.

PREMIERE PARTIE :

PRESENTATION GENERALE

PLAN DE LA PREMIERE PARTIE

- Chapitre 1

Principes des opérateurs de datamatique

- Section 1

Champ d'application des opérateurs

- Paragraphe 1

Analyse et conception d'une chaîne de traitements dans le projet "Remora"

- Paragraphe 2

Maintenance d'une chaîne de traitements

- Section 2

Complexes d'influences et hypergraphes

- Section 3

Manipulation des données en format statique et cheminements dans le graphe des influences

- Paragraphe 1

Données en format statique

- Paragraphe 2

Calcul des tailles lors de la maintenance

- Paragraphe 3

Calcul des tailles des données non déclarées

- Section 4

Standardisation des relations d'influence

- Chapitre 2

Choix de la réalisation pratique.

Première partie

Chapitre 1 :

Principe des opérateurs de datamatique

(1.1)

première partie
chapitre 1

section 1 :

Champ d'application des opérateurs

paragraphe 1

Analyse et conception
d'une chaîne de traitements
dans le projet "Remora"

(1.1.1.1)

Le projet "Remora" (Rolland, 1974) en général et les outils de datamatique en particulier interviennent dans la conduite et la maintenance de l'automatisation d'une application.

Nous donnerons d'une application la définition qu'en propose (Mallet, 1971) à partir du concept de procédure.

Définitions :

On appelle procédure un ensemble de traitements d'informations donnant réponse à un événement, tendant à une fin définie utile et indispensable au fonctionnement de l'entreprise, et d'exécution assez fréquente pour qu'on ait pu en arrêter les modalités.

(exemples : la paie du personnel, la gestion d'un stock de marchandises.)

On appelle événement un ensemble d'informations à l'origine d'une ou plusieurs procédures.

(exemples : la fin du mois, une rupture de stock, un bon de commande.)

On appelle famille de procédures un ensemble de procédures, appartenant à un même domaine d'activité de l'entreprise, traitant d'informations concernant une même population, déclenchées par des événements de même nature et nécessaires pour l'obtention des mêmes sorties.

Une application est constituée de la partie de la famille de procédures qui est automatisée.

.../...

Le projet Remora propose au concepteur un ensemble d'outils visa
à faciliter la conception et la mise en oeuvre d'une automatisation d'applic
c'est-à-dire lui permettant de passer d'une famille de procédures qu'il a
recensées à une chaîne de traitements.

Ce passage s'effectue traditionnellement en plusieurs étapes :

- l'étude d'opportunité
- l'analyse fonctionnelle
- l'analyse organique
- la programmation

La chaîne de traitements, résultat de l'étape "programmation", est
ainsi la traduction dans un langage exécutable sur ordinateur des traitemen
de la famille de procédures.

Pour satisfaire aux contraintes techniques imposées par l'exploitati
sur l'ordinateur, la chaîne apparait physiquement comme une succession
d'unités de traitements (U. T.) ou programmes.

La première des étapes ou étude d'opportunité (Bauvin, 1968 ; Reix,
1971 ; Yvon et Semin, 1970) est une étude des procédures existantes, comp
tée par une évaluation des volumes et des coûts. Elle permet de prendre l
décision d'automatiser ou non les différentes procédures.

Le projet Remora se place après cette phase. Il s'intéresse par co
aux autres étapes de la transformation, en apportant au concepteur un ensem
d'outils qui, s'ils sont utilisés suivant un ordre imposé, lui permettent de
passer sans heurt de l'analyse à la programmation puis à la maintenance,
le libérant d'une partie importante de son travail par l'automatisation des
tâches.

Tout le système s'appuie sur une idée force, sa pierre d'achoppem
une tentative de normalisation des faits de gestion dont l'aboutissement se
leur traduction automatique.

On peut constater en effet que si des procédures telles que la paie
la facturation varient d'une entreprise à une autre, il semble anormal que l
automatisation conduise à recommencer chaque fois le même travail en se
heurtant aux mêmes difficultés sans bénéficier de l'expérience acquise.

.../..

Il a semblé possible au groupe "Remora" de mettre en évidence,
pour une très large classe de problèmes, d'une part des traitements communs
à l'ensemble des fonctions de la classe et d'autre part des traitements spé-
cifiques à chacune de leur réalisation ou paramètres : terme qu'il faut
prendre au sens large, un paramètre pouvant être un mot, une expression
arithmétique ou booléenne ou un quelconque ensemble de traitements non
immédiatement spécifié.

Les traitements communs sont connus a priori, seuls les paramètres
ont besoin d'être spécifiés. C'est le rôle que Remora assigne aux descripteurs
documents adaptés à la saisie des paramètres par des gestionnaires, c'est-à-
dire dans un langage se rapprochant autant que faire se peut de la langue
naturelle. Les descripteurs sont ainsi et des outils de communication et des
outils d'analyse.

Ce sont en effet des outils d'analyse, car le gestionnaire y décrit les
traitements suivant un mode déclaratif (CODASYL, 1972) c'est-à-dire dans
l'ordre de leur conception et non de leur exécution : c'est alors le mode
impératif, celui des langages de programmation. Le mode déclaratif s'associe
à une méthode de conception descendante (Crocus, 1974) c'est-à-dire à une
description par étapes successives de la suite des transformations qui permet-
tent de passer des résultats aux données du problème. A chaque étape
apparaissent des données agrégées ou résultats intermédiaires.

Les descripteurs sont aussi des outils de programmation :

En effet des générateurs, utilisant, d'une part, la charpente pré-
établie que constituent les traitements communs, d'autre part, les descrip-
teurs, génèrent les modules de traitements correspondants.

Nous en verrons une utilisation dans notre propre problème lors de
la définition des règles de calcul de tailles (chapitre 2. 2) *

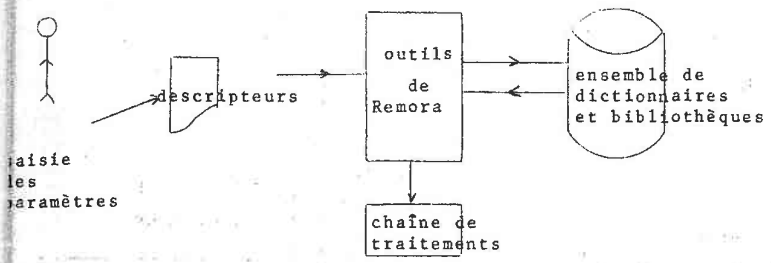
---

* note : toute référence à une autre partie de cet ouvrage pourra être
indiquée par une suite de chiffres :

1er chiffre : partie
2è " : chapitre
3è " : section
4è " : paragraphe

.../...

Une vue synoptique du fonctionnement du système est ainsi :



La figure 1 fait apparaître les principaux outils du système et les composantes de l'ensemble des dictionnaires et bibliothèques.

.../...

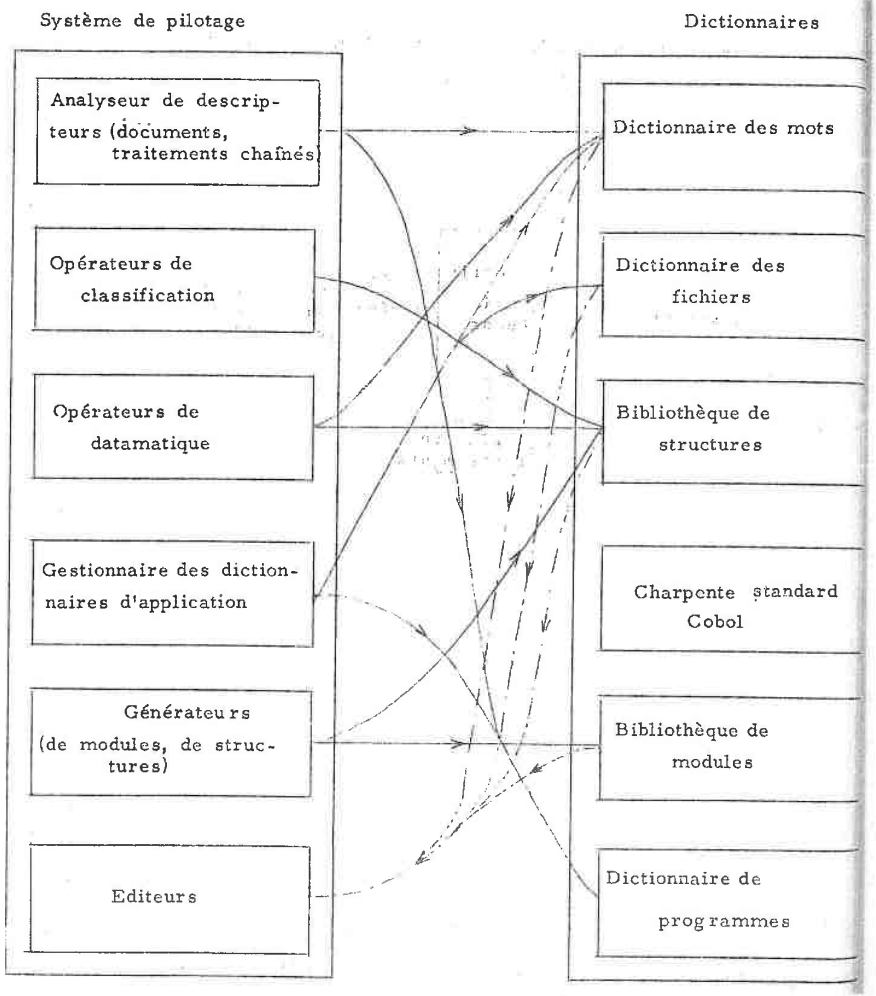


Figure 1

A partir d'entrées limitées (les descripteurs) exprimées dans un véritable langage d'analyse le système génère les différents éléments qui constituent la chaîne des traitements :

- Les analyseurs dépouillant les descripteurs font apparaître résultats, résultats intermédiaires et données.

- Les opérateurs de classification peuvent organiser les données en classes et les générateurs de structure en définissent la description automatique.

- Les opérateurs de datamatique jouent ici leur premier rôle :

Si les données et les résultats ont généralement une taille imposée (la taille peut être considérée, pour le moment, comme l'occupation mémoire d'une donnée), les résultats intermédiaires apparaissant à chaque étape de l'analyse des paramètres sur les descripteurs ont des tailles inconnues qu'il est possible de déterminer automatiquement en tenant compte des relations où ils interviennent.

- Les générateurs traduisent les descripteurs en modules de traitements.

- Le superviseur utilisant bibliothèques et dictionnaires constitue la chaîne des traitements.

Notons que les outils qui apparaissent ici comme s'enchaînant les uns aux autres dans un ordre imposé peuvent, grâce à leur propriété d'orthogonalité, s'utiliser indépendamment du contexte où nous les avons présentés.

Ainsi, par exemple, on peut concevoir que les opérateurs de datamatique puissent s'appliquer, avec le rôle que nous venons de leur assigner, à un programme déjà écrit où les résultats intermédiaires n'auraient pas été déclarés.

S'ils apparaissent ici dans leur aide conceptuelle à la conduite d'un projet informatique ; certains d'entre eux interviendront aussi au niveau de la maintenance de la chaîne de traitements qu'ils auront permis de construire ; d'autres (certains dictionnaires en particulier) seront spécifiques de cette étape dont l'importance justifie qu'on s'y intéresse particulièrement.

première partie

chapitre 1

section 1 :

Champ d'application des opérateurs

paragraphe 2

Maintenance d'une chaîne de traitements

(1, 1, 1, 2)

Contrairement aux applications scientifiques, qui par leur nature évoluent peu, les applications de gestion, auxquelles s'applique tout particulièrement le système Remora, sont liées aux activités humaines et à leurs fluctuations. Aussi ont-elles un caractère évolutif tout au long de leur existence.

Pendant sa conception, comme ensuite pendant sa vie de "routine", une chaîne de traitements ne cesse de subir des modifications plus ou moins importantes, d'origines extérieures, aussi bien qu'intérieures à l'entreprise. Par exemple des impératifs économiques ou sociaux conduisent quelquefois à remanier complètement les programmes : on peut imaginer quelles ont été les conséquences de la réforme de la TVA ou de la mensualisation sur les chaînes de facturation ou de paie.

Une chaîne de traitements est donc sans cesse remise en cause. Pour tenir compte des changements, il convient, dans les cas les plus simples, de corriger les programmes ; mais généralement il faut reprendre aussi l'analyse.

Ces tâches de mise à jour continuelle constituent la dernière phase de la conduite d'une application : sa maintenance.

Le volume de la maintenance est considérable. Certaines statistiques (IRIA, 1973 ; Berger, 1973) affirment que les travaux de mises à jour absorbent jusqu'à 60 % du potentiel de programmation d'une grosse équipe et il nous semble instructif de citer Mr Bourras, Directeur au Centre d'Automatisation du Management, qui, lors du colloque d'Orsay, sur la recherche en informatique de gestion, insista sur l'importance du coût de la maintenance : "chaque fois que l'on budgète un milliard d'études on se prépare à en dépenser encore deux dans les cinq années qui suivent. Ce qui veut dire que tout effort, quelque soit son contenu, destiné à améliorer le problème de la maintenance, est très rentable".

Le problème de la maintenance a été longtemps sous-estimé et ceci se comprend un peu :

- Les délais trop courts abrègent souvent la phase de conception : l'objectif majeur étant d'obtenir rapidement une chaîne correcte qui donne une satisfaction immédiate sans prévoir les possibilités d'évolution ultérieure.

.../...

- La recherche d'une rentabilité immédiate conduit, au moment du découp de la chaîne en unités de traitements, à alourdir ces U. T. alors que leu conception, pour une maintenance facilitée, pousse à les spécialiser. Il certain qu'une trop forte spécialisation diminue les performances de la ch

Un pas en avant a été fait dans le domaine par les méthodes d'ana qui ont propo sé une documentation fournie sur la chaîne et insisté sur l'i tance d'une tenue à jour de cette documentation.

Cert aines méthodes proposent même des "logiciels" spécialisés p gérer des répertoires (Minos) destinés spécifiquement aux problèmes de maintenance.

Les gestionnaires des dictionnaires d'application de Remora ont u rôle analogue, gérant la documentation plus complexe du système, consti par les dictionnaires et les bibliothèques.

Il est certain que, grâce à l'automatisation des tâches que Remor propose, une solution simple au problème de la maintenance consiste à f appel à la succession des outils : c'est-à-dire, par exemple, à saisir les nouveaux paramèt res sur les descripteurs pour générer les modules de traitements correspondants.

Il est évident que ce procédé serait trop coûteux pour être systém En particulier, la fréquence des corrections portant sur les tailles de don et dont les incidences sont limitées à l'ensemble de leurs déclarations just que l'on recherche un outil adapté.

C'est le deuxième objectif que nous avons assigné aux opérateurs datamatique : la succession des impacts d'une modification de la taille d'u (ou plusieurs) donnée (s) peut être définie automatiquement de même que corrections à effectuer sur les déclarations des données incidentes.

Du fait de l'orthogonalité de l'outil, il pourra être utilisé avec la fonction précédente dans n'importe quelle chaîne de programmes.

.../..

Les outils de datamatique ont donc deux objectifs :

1 - Evaluer la taille des résultats intermédiaires :

Si ceux-ci apparaissent naturellement comme variables agrégés dans le processus de conception descendante que propose le système Remora et sont comme tels des aides à l'analyse de l'application étudiée, il semble anormal d'exiger du programmeur qu'il ait à estimer au mieux la taille de ces variables afin de les déclarer. Leur taille est le résultat d'une analyse complexe de la suite de transformations qui les a fait apparaître et peut être définie automatiquement.

2 - Corriger les tailles des variables "influencées" par la modification de la taille d'une (ou plusieurs) variable (s) incidente (s) :

La modification de taille d'une donnée peut avoir une "cascade" d'impacts sur d'autres données, qu'il est souhaitable, compte tenu de la difficulté du problème, de définir automatiquement.

Les calculs de tailles apparaissant dans ces deux problèmes sont des cas particuliers d'une étude plus générale qui nous a conduit à la notion de complexe d'influences que nous aborderons à la section suivante (1. 1. 2). Le cas des tailles sera abordé ensuite à la section 1. 1. 3.

section 2 :

Complexes d'influences et hypergraphes

(1.1.2)

Chaque entité manipulée dans une application sera désignée par son mot.

L'ensemble des mots constitue le vocabulaire de l'application.

Dans Remora, ce vocabulaire est répertorié dans le dictionnaire des mots.

Un mot peut être défini en extension, par la liste des valeurs qu'il peut avoir, ou en compréhension, par le processus permettant de calculer ses valeurs. (Peccoud, 1971).

L'interprétation que fait un individu d'un mot porté sur un document en fait une information.
Son stockage en mémoire d'ordinateur le rend "donnée" (Warnier, 1972)

Dans le cadre de ce travail, nous n'accorderons pas une importance fondamentale à ces distinctions.

Un mot a une désignation unique ou référence qui permet de le trouver dans le dictionnaire des mots.

Il peut être manipulé dans les programmes par des noms ou identificateurs différents ; la liste d'identificateurs ainsi que leur localisation dans les programmes figure dans le dictionnaire.

Nous appelons caractéristique toute information associée à un mot : l'ensemble des caractéristiques de chaque mot est stocké dans le dictionnaire.

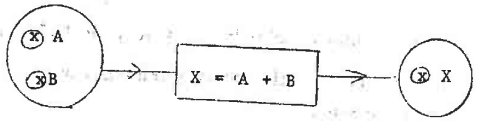
La taille, définie au paragraphe précédent est une caractéristique de mot.

.../...

Les traitements de l'application portent sur les "mots". Ils peuv
être regroupés en familles de telle sorte que certains mots : les résulta
se déduisent d'autres : les opérandes par exécution des traitements de
famille.

Nous appelons groupe opératoire le triplet (opérandes, famille de
traitements, résultats)

exemple : ({ A, B }, X = A + B, { X }) que l'on peut aussi schématise
la forme :



A partir des traitements d'une application, on peut obtenir des gr
opératoires de complexité croissante selon que la famille de traitements
constituée d'une opération élémentaire (comme ci-dessus) d'un ensemble
restreint d'opérations élémentaires (comme par exemple les traitement
décrits sur un descripteur de Remora) ou de tous les traitements d'un p
me ou même d'une chaîne de programmes.

Il convient, dans chaque cas particulier, de préciser le niveau de
complexité des groupes opératoires constitués.

Aux traitements d'un groupe opératoire, on peut associer des règ
qui déterminent la valeur d'une caractéristique des résultats en fonction
celles des opérandes.

Ces règles dépendent de la famille de traitements du groupe opér
et de la nature de la caractéristique étudiée.

Elles devront être explicitées dans chaque cas particulier.

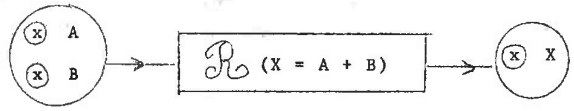
.../...

Nous appelons relation d'influence (liée à une caractéristique et à
un groupe opératoire) le triplet (opérandes, règles, résultats).

exemple :

({ A, B } , règle de calcul de la taille de X avec X = A + B , { X })

que l'on peut encore schématiser



remarques :

Nous avons choisi le terme "relation d'influence" pour exprimer
d'une part que opérandes et résultats sont liés par une "relation" et d'autre
part que les caractéristiques des résultats sont "influencées" par celles des
opérandes par l'intermédiaire de la règle.

La complexité des règles croît évidemment avec celle des groupes
opératoires auxquelles elles sont associées.

.../...

Dans ce travail, nous nous intéressons à l'ensemble des relation d'influence associées aux groupes opératoires qui constituent l'ensemble traitements d'une application :

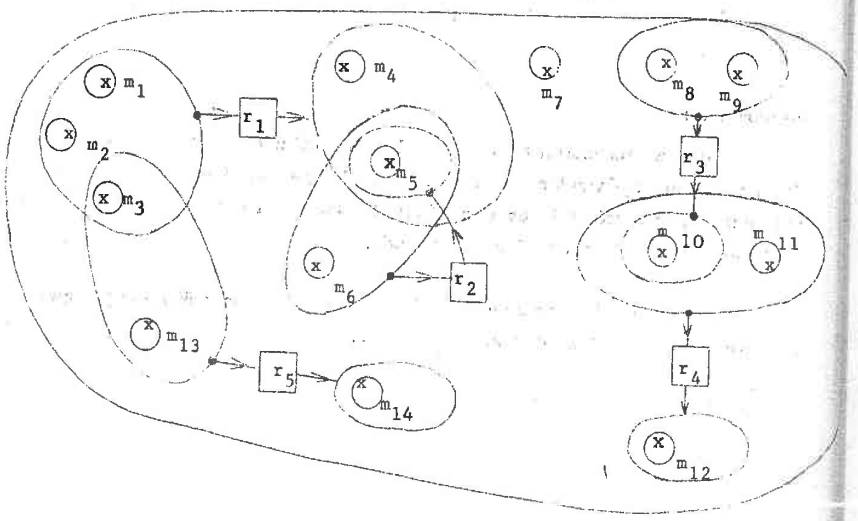
Nous appelons complexe d'influence s un couple $(\mathcal{M}, \mathcal{R}o)$ où

$\mathcal{M}$ est un ensemble de mots munis d'une caractéristique ;

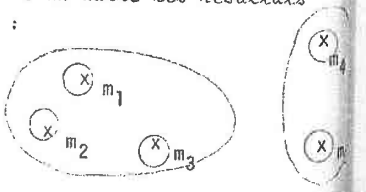
$\mathcal{R}o$ est un ensemble de relations d'influence dont les règles portent sur caractéristique.

A chaque application peuvent ainsi être associés plusieurs comple d'influences suivant le choix que l'on a fait, d'une part des mots et de le caractéristique, d'autre part des groupes opératoires et des règles corres dantes.

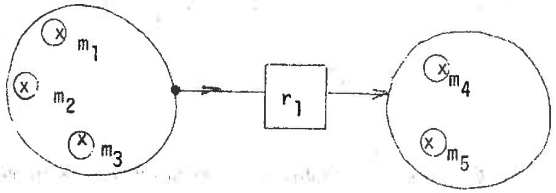
En conservant la représentation schématique des relations d'influ proposée précédemment, on peut schématiser un complexe d'influences : par exemple



- *Chaque mot n'est représenté qu'une fois par* $(x)_{m_i}$
- *Un trait entoure les opérandes et un autre les résultats de chaque relation d'influence :*



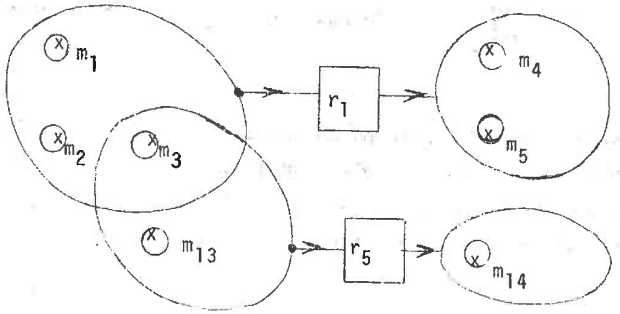
- *Chaque règle est représentée par* $\boxed{r_j}$
- *Une flèche joint l'ensemble des opérandes à la règle et une autre joint la règle à l'ensemble des résultats :*



Cet exemple permet de faire quelques constatations :

Certains mots appartiennent à plusieurs relations d'influences :

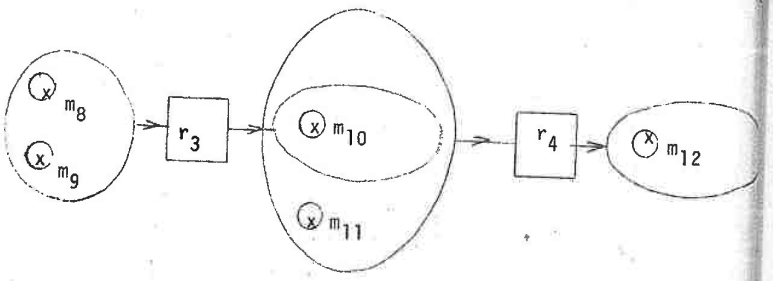
On peut noter en particulier une intersection entre ensembles d'opérandes :



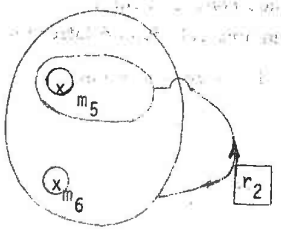
(m_3 *est opérande de* $\mathcal{R}o_1$ *et* $\mathcal{R}o_5$)

.../...

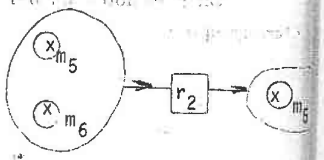
Une intersection entre résultats d'une relation d'influence et opér
d'une autre suggère une notion de succession entre les relations d'influen



Une intersection entre opérandes et résultats d'une même relation correspond à des traitements usuels en programmation, par exemple "affecter à m_5 la valeur de $m_5 + m_6$" :



que l'on peut développer :



Nous verrons l'importance de ces intersections pour la résolution des problèmes posés au paragraphe précédent.

Dans ce cas les ri sont les règles de calcul de la taille des rés d'un groupe opératoire en fonction de celle des opérandes.

Compte tenu de ce choix, le schéma de la page 18 peut ainsi représenter différents complexes d'influences :

Par exemple celui qui est associé à un programme où les mi sont les identificateurs du programme et où les groupes opératoires corres pondent à des opérations élémentaires.

.../...

Ou bien, si l'on considère l'analyse d'une application par les outils de Remora, les mi sont les mots et chaque groupe opératoire peut être constitué à partir des traitements décrits sur un descripteur.

De même un dossier d'analyse quelconque comporte des traitements qui peuvent être assemblées en groupes opératoires où les mots:opérandes et résultats sont les données décrites dans le dossier.

Enfin, si chaque programme est considéré comme une seule famille de traitements, les entrées du programme, le programme et ses sorties constituent un groupe opératoire auquel correspond une relation d'influence. Un complexe d'influence est alors obtenu en assemblant les données "globales", entrées ou sorties de programmes, d'une part, et les relations d'influence obtenues à partir de chaque programme, d'autre part.

Il apparait ainsi que tout raisonnement effectué sur l'entité complexe d'influences au niveau de sa définition générale sera valable pour chacune de ces occurrences ; c'est ce que nous exploiterons dans les problèmes de calcul de la caractéristique taille.

En outre la notion de complexe d'influences permet une généralisation des solutions que nous développons dans le cadre de notre travail ; car en effet, si nous changeons la caractéristique et/ ou les règles, nous pouvons à partir des mêmes groupes opératoires obtenir d'autres complexes d'influences et résoudre par les mêmes procédés des problèmes de natures différentes.

Ainsi, tout problème traité sur un cas particulier peut par l'intermédiaire de la notion générale de complexe d'influences être transposé à un autre cas particulier. (Nous songeons à la détermination automatique de jeux d'essais permettant des tests de possibilités, c'est-à-dire testant toutes les "branches" d'un programme).

.../...

La définition du complexe d'influences et sa représentation schématique présentent des analogies avec la notion de graphe et surtout avec la notion d'hypergraphe :

Définition (Berge, 1970)

Soit $X = \{x_1, x_2, \ldots, x_n\}$ un ensemble fini, et $\mathcal{E} = (E_i / i \in I)$ une famille de parties de X. On dira que $\mathcal{E}$ constitue un hypergraphe sur X si l'on a

$$(1)\quad E_i \neq \emptyset \quad \forall_i \in I$$

$$(2)\quad \bigcup_{i \in I} E_i = X$$

L'hypergraphe est représenté par le couple $(X, \mathcal{E})$

Les x_i sont les points de l'hypergraphe.

Les E_i sont les arêtes de l'hypergraphe.

On peut obtenir un hypergraphe à partir d'un complexe d'influences.

Soit $(\mathcal{M}, \mathcal{R})$ un complexe d'influences où

$$\mathcal{M} = \{m_1, m_2, \ldots, m_n\}$$
$$\mathcal{R} = \{\mathcal{R}_i ;\ i \in I\}$$

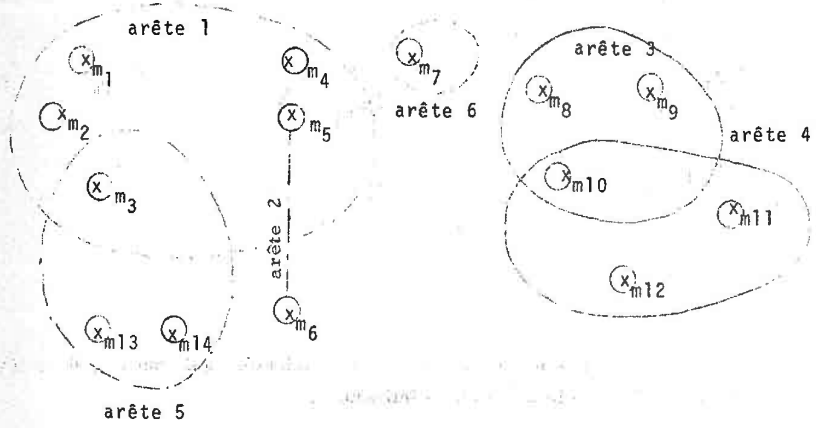
Soient $\mathcal{O}_{p_i}$ les opérandes et $\mathcal{R}_{s_i}$ les résultats d'une relation d'influence quelconque $\mathcal{R}_i$.

Posons $E_i = \mathcal{O}_{p_i} \cup \mathcal{R}_{s_i}$

Alors, si $\mathcal{E} = (E_i / i \in I)$, $\mathcal{E}$ est une famille de parties de $\mathcal{M}$ et si tout point de $\mathcal{M}$ appartient au moins à une relation $\mathcal{R}_i$ (condition (2)) il est évident que le couple $(\mathcal{M}, \mathcal{R})$ est un hypergraphe.

.../...

L'exemple de la page 18 peut alors avoir pour représentation schématique :



- *Chaque mot est représenté une fois par* $(x)\ m_i$
- *Un trait -.-.-.- entoure les points qui constituent chaque arête.*

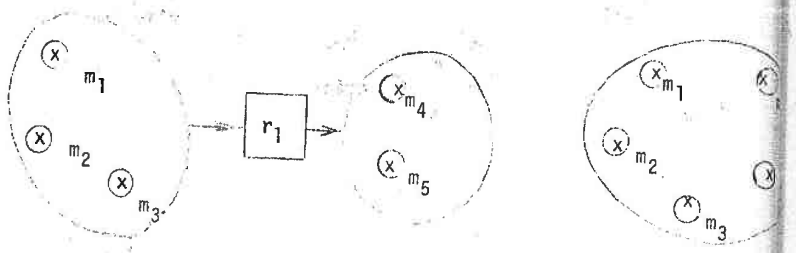
(l'arête 6 a été ajoutée pour vérifier la condition (2), celle-ci n'est pas fondamentale pour nous)

Si toute arête ne comporte que 2 éléments, comme l'arête 2, on retrouve les graphes simples non orientés comme cas particulier des hypergraphes.

.../...

Comparons le schéma précédent avec celui de la page 13 :

complexe d'influences : hypergraphe :



L'hypergraphe est une notion non orientée, qui contient, de ce fait, d'information que le complexe d'influences.

Nous proposons une extension des hypergraphes en définissant des hypergraphes orientés :

Définition

Soit $X = \{x_1, x_2, \ldots, x_n\}$ un ensemble fini, et $\mathcal{E} = (E_i / i \in I)$ une famille de parties de $\mathcal{P}(X) \times \mathcal{P}(X)$. On dira que $\mathcal{E}$ constitue un hypergraphe orienté sur X si l'on a

(1) $E_i \neq (\emptyset, \emptyset)\ \forall_i \in I$

(2) si $E_i = \left(\left\{ o_{j_i} ; j_i \in J_i \right\}, \left\{ e_{j'_i} ; j'_i \in J'_i \right\} \right)$:

$$\bigcup_{i \in J} \left(\bigcup_{j_i \in J_j} o_{j_i} \bigcup \bigcup_{j'_i \in J'_i} e_{j'_i} \right) = X$$

.../...

L'hypergraphe orienté est représenté par le couple $(X, \mathcal{E})$

Les xi sont les points de l'hypergraphe orienté :

Les Ei sont les arcs de l'hypergraphe orienté :

Les origines de l'arc Ei sont les o_{j_i},

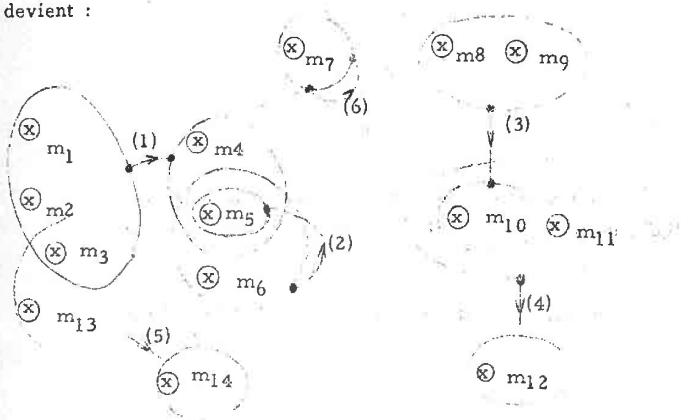
Les extrémités de l'arc Ei sont les e_{j_i}. (1)*

Soit $(\mathcal{M}, \mathcal{R})$ le complexe d'influences défini précédemment

Posons $Ei = (\mathcal{O}_{p_i}, \mathcal{R}_{s_i})$ et $\mathcal{E} = (E_i / i \in I)$

alors $(\mathcal{M}, \mathcal{E})$ est un hypergraphe orienté.

Sur l'exemple précédent, la schématisation de l'hypergraphe orienté devient :



- *Chaque mot est représenté une fois par* $(x)_{m_\ell}$
- *Un trait entoure les origines et un autre les extrémités de chaque arc. Une flèche rejoint ces deux ensembles entourés, des origines vers les extrémités.*

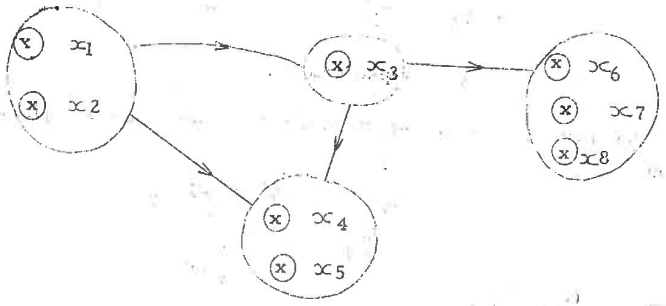
(1) * Dans tout ce travail, nous appelons "origines" d'un arc ses "extrémités initiales" et "extrémités" ses "extrémités terminales", nous utilisons cette terminologie aussi bien pour les graphes orientés que pour les hypergraphes orientés.

.../...

(L'arc (6) permet, ici encore, de vérifier la condition (2))

Considérons deux ensembles quelconques x et y origines e extrémités d'arc :

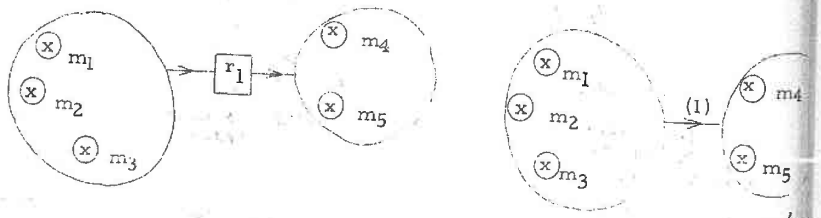
si $\forall x$ et $\forall y$: $x \cap y = \emptyset$ ou $x = y$, alors on ob comme cas particulier des hypergraphes orientés, des graphes orienté dont les points sont des parties d'un ensemble.



Comparons le schéma de l'hypergraphe orienté avec celui du c d'influences de la page 18 :

par exemple :

complexe d'influences : hypergraphe orienté



Un arc de l'hypergraphe orienté obtenu à partir du complexe d'influences est un couple (opérandes, résultats), alors qu'une relation d'influence est un triplet (opérandes, règles, résultats) ; la notion de complexe d'influences est plus riche que celle d'hypergraphe orienté : c'est un hypergraphe orienté où chaque arc est muni d'informations ; on pourrait donner un nom à la notion d'hypergraphe orienté accompagné d'une application de l'ensemble des arcs dans un ensemble quelconque, le complexe d'influences pourrait alors être considéré comme un cas particulier de cette notion très générale.

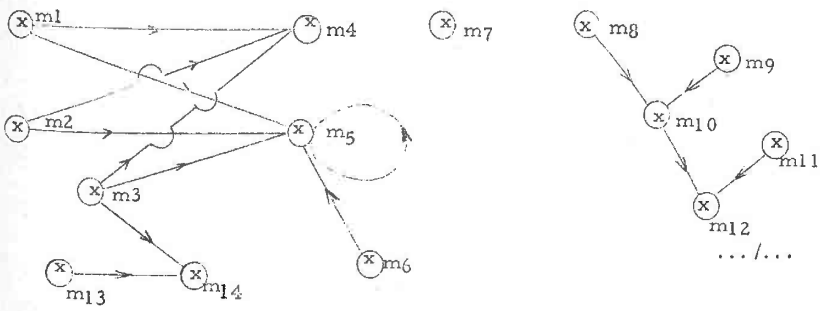
En fait, nous constaterons dans la deuxième partie que la seule généralisation fondamentale, pour notre travail, est celle qui permet d'associer à un complexe d'influences un hypergraphe orienté. En effet la définition d'un cheminement sur les hypergraphes orientés permet de résoudre élégamment notre problème. Ce cheminement s'effectuera sur le graphe représentatif des arcs de l'hypergraphe orienté, sous-graphe du graphe des noeuds d'influences.

Toutefois, pour aborder simplement les problèmes dans cette première partie, nous utiliserons un graphe plus intuitif : le graphe des influences.

Définition :

Nous appelons graphe des influences associé à un complexe d'influences, le graphe dont les sommets sont les mots ($\mathcal{M}$) et dont les arcs traduisent l'existence de relations d'influence où l'origine x de l'arc est opérande de l'extrémité y résultat.

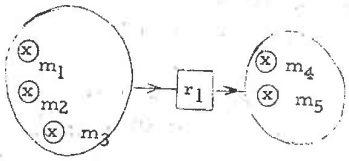
En prenant le même exemple, nous obtenons comme représentation schématique de ce graphe :



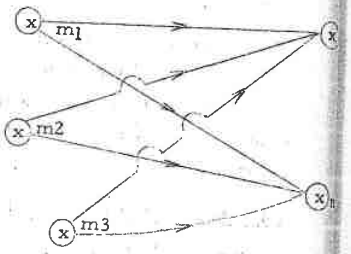
Ainsi, un arc joint pour chaque relation d'influences tout opéran à tout résultat.

Par exemple :

complexe d'influences :



graphe des influences



Avant d'aborder les problèmes de cheminement, il nous faut précis et justifier l'hypothèse de base de notre travail qui est celle de la manipu tion des données en format statique.

(1.1.3.1)

(1.1.3.1.1) Nécessité des données numériques en caractères

Dans les applications de gestion, auxquelles nous pensons tout particulièrement, et malgré tout es les difficultés que cela entraîne, la manipulation du caractère au niveau des calculs, et donc des langages de programmation, est indispensable.

Lors des calculs scientifiques, réalisés par exemple en Algol ou en Fortran, les nombres sont mémorisés uniquement en format interne binaire, virgule fixe ou flottante ; ils ont un encombrement mémoire constant (1 ou 2 mots) et des capacités extrêmes fixes, dépendant du nombre de bits d'un mot mémoire et de la représentation interne choisie par le constructeur. Selon les valeurs possibles du nombre l'utilisateur choisira simplement entre les représentations entières ou réelles en simple ou en double précision ; ce qui simplifie à l'extrême la déclaration des données, laquelle peut à la limite devenir implicite (fortran).

De tels formats ne peuvent suffire pour les calculs de gestion : ils permettent mal de maîtriser les arrondis, ils ont une précision constante et surtout ne permettent pas des traitements comparatifs avec des chaînes de caractères.

Le problème des arrondis intervient souvent en gestion dans des problème s tels que ceux de la comptabilité, où le résultat donné au centime près est obtenu par application de règles précises sur les arrondis. Ces calculs seraient très délicats avec des formats binaires en virgule flottante (où, par exemple, 1 peut être mémorisé 1.000 000 ou 0.999 999).

La précision constante donne à tous les nombres les mêmes capacités extrêmes et surtout le même nombre théorique de chiffres significatifs sans que l'on y distingue le nombre de chiffres ayant effectivement un sens.

.../...

Le dernier point est, sans doute, le plus important et nous allo
l'illustrer sur deux exemples :

Considérons, tout d'abord, un programme de contrôle qui lit de
données sur cartes : un certain champ de la carte contient théoriqueme
un nombre, c'est-à-dire une suite de caractères numériques (ex : 1025
mais peut évidemment contenir par erreur une suite de caractères no
numériques (ex : UØ25).

Quels pourront être alors les comportements à l'exécution d'un
programme de contrôle écrit dans un langage où tout nombre est conve
en binaire (exemple : fortran) :

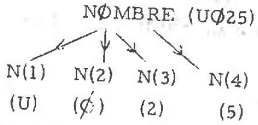
- Si le programmeur n'a pas prévu d'ordres spécifiques, il y a en géné
une impression de la carte erronée, d'un message standard d'erreur d
conversion, et les calculs sont poursuivis avec une valeur binaire fauss

- Si le programmeur a prévu une intervention spécifique simple telle qu
l'arrêt du programme sur erreur de conversion ; il y aura une perte de
temps considérable. Une telle solution est de ce fait inacceptable pour u
programme de contrôle à haute fréquence d'utilisation puisqu'il faudrait
exécuter le programme autant de fois qu'il y a de champs erronés sur l
cartes.

- S'il a choisi de programmer une récupération de l'erreur de conversio
un traitement adéquat, il devra faire appel à une programmation délicate
faisant obligatoirement intervenir les chaînes de caractères.

Au contraire, tout langage de gestion, comme Cobol ou PL1, pré
la manipulation des chaînes de caractères quelconques et des nombres e
caractères, facilite la programmation d'un tel problème.

Dans le cas de cet exemple, il suffit de considérer UØ 25 comm
une chaîne de caractères, de tester son type (est-elle numérique ?) et en
d'erreur d'examiner isolément chacun de ses chiffres (grâce à une struc
hierarchisée)

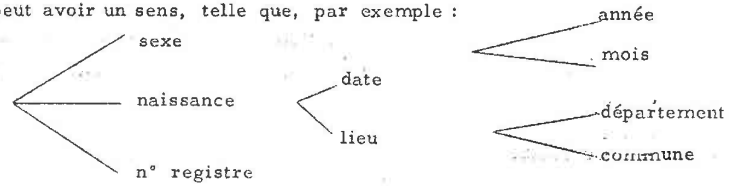


ce qui conduit à isoler l'erreur (U et Ø ne sont pas numériques).

.../...

Un simple message d'erreur peut suffir ; mais on peut aussi chercher à réaliser une correction automatique : par exemple en remarquant que U et 1 sont sur la même touche de la perforatrice et que Ø et O sont souvent confondus, on peut imaginer la correction automatique de UØ25 en 1025.
Si la valeur numérique obtenue est plausible, l'exécution des calculs pourra avoir lieu.

Considérons, maintenant, une chaîne de caractères numériques dont la structure est hierarchisée et où chaque élément ou groupe d'éléments (champ) peut avoir un sens, telle que, par exemple :



On peut, par exemple, utiliser "date" comme une seule valeur numérique pour calculer l'âge ou "n° insee" comme un nombre qui peut servir, à la limite, de clé d'accès. En outre, un langage tel que Cobol, permet facilement des éditions variées de chaque item ou groupe d'items.

Il est évident que de tels traitements sont difficiles à réaliser si les données sont représentées en binaire.

Enfin remarquons que tout document à usage humain doit obligatoirement, pour être compréhensible, être exprimé en caractères. Si la représentation interne conserve les caractères, leur représentation sur le document n'en sera qu'améliorée. Au contraire, si les données sont stockées autrement, il faudra opérer des conversions pouvant introduire des "erreurs de chute".

Les données numériques en caractères sont donc bien nécessaires dans les langages de gestion ; leur emploi nécessite généralement une description minutieuse.

(1.1.3.1.2) **Définition de la taille**

Les données en caractères n'ayant pas un format standard, il f
obligatoirement les déclarer. La déclaration doit être faite avec préci
sinon les calculs risquent d'être erronés.

Les renseignements fournis au moment de la déclaration consti
une caractéristique du mot. Cette caractéristique est un n-uple consti
principalement de la taille totale, la taille décimale et la taille entière

Définitions

Nous appelons taille totale l'encombrement en caractères de la
représentation d'un mot en mémoire.
Pour les données numériques :

La taille décimale est le nombre de chiffres après la virgule déc
conservés en mémoire.

La taille entière est de même le nombre de chiffres devant la vi
décimale.

Remarque

La somme de la taille décimale et de la taille entière ne donne p
toujours la taille totale : par exemple, si la donnée est "éditée" et conti
en outre des caractères insérés au moment de l'édition ou encore lorsq
la représentation interne est condensé par rapport à la représentation ex
par exemple, en décimal condensé (Cobol CII) un chiffre est codé sur u
demi-caractère, alors

$$\text{taille totale} = \left\lceil \frac{\text{taille entière} + \text{taille décimale}}{2} \right\rceil + 1 \left(\text{signe} + \begin{cases} 1 \text{ chiffre} \\ \text{ou} \\ 4 \text{ bits perd} \end{cases} \right.$$

(partie entière de)

.../...

Donnons quelques exemples de déclarations de données :

a) en Cobol :

77 F PICTURE S9(8)V99.

est la déclaration d'un nombre décimal étendu (1 chiffre par caractère) signé (le signe est superposé à droite). Les tailles totale, décimale et entière sont respectivement 10, 2 et 8.

77 G PICTURE S9(8)V99 COMP-3. (Cobol C.I.I.)

est la déclaration d'un nombre décimal condensé (4 bits pour un chiffre) signé (4 bits pour le signe).
Les tailles sont : taille totale 6, taille décimale 2, taille entière 8.
(4 bits sont perdus).

b) en PL1, la clause PICTURE peut permettre les mêmes déclarations qu'en Cobol. En outre

DCL G DEC FIXED (10,2) ; est une déclaration équivalente à celle de G faite ci-dessus en Cobol.

Si l'on suppose que la déclaration est statique à l'exécution (ce qui permet une réservation statique), un problème important se pose au programmeur quant à la précision de la taille déclarée : en effet, toute erreur peut être fatale puisqu'une taille trop courte entraîne la perte de chiffres significatifs. (par exemple, si l'on range 9880 dans une zone de taille totale 3, on obtient peut être 988 ou 880 mais en aucun cas 9880 et la suite des calculs est évidemment erronée).

Ce même problème se retrouve chaque fois que la taille d'une donnée déclarée est modifiée, soit directement, soit lorsque les incidences successives d'une modification de taille d'une autre donnée l'atteignent, ce qu'il n'est pas toujours aisé de déterminer.

Devant de tels inconvénients, certains langages ont proposé d'autres solutions parmi lesquelles, a priori, certaines peuvent s'adapter aux calculs de gestion.

(1.1.3.1.3) Autres modes de déclaration

Même en gestion on peut certes envisager d'utiliser, au cours d[es] calculs, les formats internes binaires de manière à limiter le nombre d[e] déclarations délicates.

Les traitements sont alors découpés en trois phases : des lectur[es] accompagnées de conversions, des calculs en binaire, des écritures aprè[s] conversions.

Un premier avantage apparent de cette méthode est une optimisati[on] temps de calcul : les calculs sont plus rapides en binaire surtout dans l'h[y] où l'arithmétique décimale n'est pas câblée sur l'ordinateur utilisé.

Néanmoins, une caractéristique des traitements en gestion étant l[e] pourcentage des opérations d'entrées-sorties par rapport aux calculs ; il n['] évidemment intéressant de convertir les données en binaire que si le temp[s] supplémentaire dû aux conversions (décimal ⟶ binaire et binaire ⟶déc[.] est inférieur au gain de temps obtenu sur les calculs.

D'autre part l'utilisation des calculs binaires ne dispense pas en gé[n] de la déclaration des formats des zones d'entrée-sortie (par exemple for[.] Fortran ou structures des fichiers Cobol) et des difficultés de mise à jou[r] ces formats.

Un autre type d'amélioration peut être apporté à la maintenance de[s] formats par l'utilisation de variables de tailles :

ex : 77 F PICTURE $9(P_1)V9(P_2)$.

où P_1 et P_2 sont des variables dont la valeur peut être calculée.

Les tailles peuvent être variables à deux niveaux : pendant la comp[.] ou à l'exécution.

La première solution suppose que le langage utilisé est en fait un métalangage : Des métavariables peuvent être manipulées pendant la comp[.] et prendre des valeurs calculées que l'on affecte, par exemple, aux taille[s]

.../...

Le programme compilé obtenu correspond à un nouveau texte source où les déclarations de données, en particulier, définies par des métavariables dans le programme initial, ont à présent des valeurs constantes. (ex : les instructions % en PL1).

La seconde solution impose une gestion dynamique de la mémoire ; les variables de tailles étant des variables ordinaires, les réservations de place en mémoire ont lieu de manière dynamique à l'exécution. C'est le cas en PL1 : (Berthet, 1971)

1 - pour les tableaux de tailles variables dont la place est réservée lors de l'activation du bloc qui les déclare ;

2 - pour les variables contrôlées dont la place est réservée par l'instruction ALLOCATE et libérée par l'instruction FREE (avec une gestion en pile si plusieurs réservations se succèdent sans libération).

Remarquons que si cette solution est élégante parce qu'elle permet par exemple de mettre en facteur des paramètres de la taille :

ex : M, N et P auront P_1 chiffres avant la virgule

et T, U, V et W auront P_2 caractères alphanumériques.

En fait, le problème des tailles n'est pas entièrement résolu, puisqu'il faut calculer P_1 et P_2 et que si ce calcul est mal fait, cela peut entraîner des erreurs au cours des traitements effectués sur ces données.

Certains langages, qu'on peut envisager d'utiliser en gestion, p
une manipulation des données plus sophistiquée :

Ainsi en APL, toute donnée change automatiquement de type e
fonction des valeurs qu'elle prend à l'exécution. Cette souplesse est
grâce à un adressage indirect (Armingaud, 1972)

SNOBOL, conçu spécifiquement pour la manipulation des chaîne
caractères (Griswold Poage Polonsky, 1972) dispense le programmeur
déclarations de type. Le "système" fait les calculs en binaire et se cha
automatiquement des conversions. On retrouve donc les avantages et le
inconvénients, déjà cités, des calculs en binaire avec l'inconvénient s
mentaire d'une diminution de performance due au fait que les conversi
ne peuvent être choisies a priori mais seulement au moment de l'exécut
(par exemple : la chaîne de caractères 1225 sera convertie en virgule
(entier) mais 10.5 sera convertie en virgule flottante (réel)).

Remarquons, d'autre part, qu'au niveau des éditions, le problè
entier, le programmeur ayant à préparer le format intégral de ces lig
d'impression, problème important en gestion où l'on édite souvent sur d
papiers préimprimés ; SNOBOL fournit l'encombrement des résultats
en décimal et le programmeur, pour cadrer ses résultats, doit calcul
nombre de blancs nécessaires et constituer ses lignes par concaténati
de chaînes. On retrouve, ici encore, le problème de l'évaluation des
formats, au moyen de traitements exécutables prévus et choisis par le
programmeur.

(1.1.3.1.4) Conclusions

Il apparait que la manipulation des données en caractères (numériques ou non numériques) est une nécessité pour tout langage de gestion et leur déclaration statique telle que celle de Cobol est une hypothèse de travail pour la plus grande majorité des informaticiens de gestion (Les sondages révèlent que 80 % des programmes de gestion sont écrits en Cobol ; Bertini, 1973).

Elle risque de le demeurer longtemps puisque CODASYL prévoit dans le LDD (langage de définitions des données) que les déclarations d'items du "schéma" (représentation de la structure virtuelle de la base) soient en format statique (CODASYL, 1972).

On peut remarquer d'autre part que les inconvénients de telles déclarations n'ont pas été complétement supprimés dans les solutions répertoriées précédemment ; car, à un moment ou à un autre, le concepteur est amené à résoudre le problème de l'évaluation des tailles, problème difficile auquel il ne peut apporter une solution fiable que par une analyse détaillée de toutes les transformations où interviennent les données.

Nous conserverons donc l'hypothèse des déclarations statiques en acceptant en outre la possibilité de déclarations incomplètes ; mais nous apportons au concepteur, par l'intermédiaire des opérateurs de datamatique, un outil assurant automatiquement et de manière fiable la gestion des tailles des données :

Les opérateurs complètent ou corrigent les déclarations de données sur le texte source (phase de précompilation) sans nécessiter de préparation particulière de la part du programmeur. Ils permettent une gestion de la mémoire statique à l'exécution, donc sans diminution des performances des programmes.

Les opérateurs effectuent ces tâches en cheminant dans un graphe issu du complexe d'influences considéré, c'est ce que nous allons montrer sur des exemples dans les paragraphes suivants.

section 3 :

Manipulation des données en format statique et cheminements dans le graphe des influences

paragraphe 2

Calcul des tailles lors de la maintenance

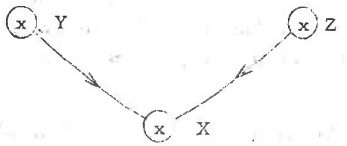
Plaçons nous dans un complexe d'influence associé à un programme.

Soit l'exemple simple de 3 mots nommés X, Y et Z de tailles respectives (représentées par les triplets (taille totale, taille entière, taille décimale)), (3, 2, 1), (2, 1, 1) et (2, 1, 1).

Supposons que ces trois nombres soient liés par une seule relation d'influence :

$$(\{Y, Z\}, \mathcal{R}_T \ (X = Y + Z), \{X\})$$

Il existe alors dans le graphe des influences les arcs :



Supposons encore que pour une raison externe la taille de Z devienne (3, 1, 2).

Logiquement, si aucune modification n'intervient sur la déclaration de X, X ne peut ne pas avoir la valeur exacte attendue par suite de la perte d'un chiffre significatif ; le résultat conservé de l'opération sera donc faux.

Pour corriger convenablement X, il faut connaitre les relations d'influence où X est résultat et Z, l'opérande modifié ; mais surtout, il faut disposer des "règles" de la relation d'influence qui permettent de calculer les tailles des résultats en fonction des tailles des opérandes.

Ici, il semble naturel de donner à X la nouvelle taille (4, 2, 2) ; c'est-à-dire d'appliquer la règle :

Accorder au résultat la plus grande des tailles décimales des opérandes.

.../...

Cette règle a été choisie en fonction du type de la relation d'inf
ici une addition.

Ce n'est pas la seule règle plausible pour calculer la taille déc
du résultat d'une addition : par exemple, on peut imaginer que le progr
meur avait initialement prévu des tailles qui donnaient lieu à une tronc
volontaire du résultat. On peut alors chercher à conserver cette tronc
en complétant la règle précédente par :

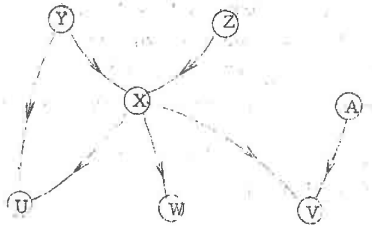
ne pas accorder au résultat une variation de taille déc
supérieure à celle des opérandes modifiés.

Cet exemple simple montre, d'une part qu'il est nécessaire de
une typologie des relations d'influence pour définir des règles propres
chaque type de relations : d'autre part que le choix des règles de calc
tailles n'est pas simple et qu'il dépend vraisemblablement des problèm
traités.

Le chapitre 2.2 sera consacré au développement de ces point

Il est évident enfin que le processus de correction de tailles é
sur l'exemple n'est pas terminé :

Supposons qu'il existe entre X et les mots nommés U, V et W
des relations d'influence apportant au graphe des influences les arcs
suivants :



(par ex : U = X * Y ; V = X - A ; W = - X)

.../...

Toute correction effectuée sur X doit être propagée sur U, V et W, c'est-à-dire les successeurs de X dans le graphe des influences.

De même, toute correction apportée à U, V ou W nécessite la correction de leurs successeurs dans le graphe ...

Il appa rait ainsi une succession de modifications qui font "boule de neige", chaque nouvelle correction en entraînant éventuellement plusieurs autres, nous désignons ce phénomène par l'expression : modifications en cascade .

Les points corrigés sont tous sur un chemin du graphe des influences issu de Z puisque les modifications se propagent d'un point vers ses successeurs dans le graphe :

Les modifications en cascade utilisent les cheminements dans le graphe des influences.

Plusieurs algorithmes de cheminement peuvent convenir pour résoudre ce problème de recherche de tous les chemins issus d'un point fixe ou d'un sous-graphe donné (Derniame - Pair 1971) ; cependant si nous ne voulons pas conserver en mémoire l'image de tous les chemins trouvés, nous devons obtenir les points dans l'ordre de la propagation des modifications.

Ces problèmes de cheminement seront étudiés à la section 2.1.1

Le problème de l'automatisation des modifications de tailles en cascade peut donc être résolu par un algorithme de cheminement dans le graphe des influences en appliquant pour chaque arc des chemins, la règle de calculs des tailles associée à la relation d'influence qui a donné naissance à l'arc.

.../...

Si l'on se place au niveau d'une chaîne de programmes, une solut
consiste à se ramener au problème précédent en regroupant toutes les rela
d'influences dues aux groupes opératoires élémentaires de toutes les unité
traitement pour ne construire qu'un seul graphe des influences. Cette sol
ne nous agrée pas car elle ne permet pas de comprendre et de juger glob
ment les modifications obtenues ; nous préférons choisir la solution suiva

Nous considérons le complexe d'influences où les mots sont restre
mots de l'application qui ont une portée globale (c'est-à-dire valable pou
la chaîne) et où chaque relation d'influence correspond à une unité de trai
elle comporte des opérandes qui sont les entrées de l'U. T. et des résulta
qui en sont les sorties ; les résultats intermédiaires n'interviennent pas d
tement comme mot dans ces relations.

En fait, si d'un point de vue formel, ces relations d'influence cor
pondent à la définition que nous avons donnée, on peut en réalité les consid
comme des macro-influences définies par le triplet (opérandes, proces
de calcul de tailles, résultats), parce que la liaison qu'elles fournissent en
entrées et sorties est obtenue comme résultat d'un processus du type déjà
exposé portant sur le complexe d'influence de niveau élémentaire(obtenu à
partir de groupes opératoires élémentaires) associable à l'U. T

Au complexe d'influences constitué par ces macro-influences on pe
associer le "graphe des macro-influences".

Les modifications en cascade au niveau d'une chaîne de traitement
peuvent donc être résolues par un algorithme de cheminement dans le grap
des ma cro-influcences en appliquant pour chaque arc des chemins un algo
de modifications en cascade au sein de l'U. T. qui a donné naissance à l'ar

L'avantage de cette solution est de spécifier le rôle de chaque U. T
dans les modifications des mots d'intérêt global à toute la chaîne.

section 3 :

Manipulations des données en format statique
et
cheminements dans le graphe des influences

paragraphe 3.

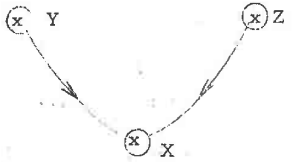
Calcul des tailles des données non déclarées

Plaçons nous de nouveau dans un complexe d'influences associé à un programme.

Considérons toujours les 3 mots X, Y et Z liés par la relation d'influence :

$$(\{X, Z\}, \mathcal{R}_I (X = Y + Z), \{X\})$$

soient dans le graphe des influences les arcs :



Supposons cette fois que X n'ait pas été déclaré et que sa taille soit donc inconnue.

Si Y et Z ont les tailles respectives (3, 2, 1) et (3, 1, 2), pour que X puisse contenir la valeur exacte Y + Z sans jamais perdre de chiffres significatifs, il faut donner à X la taille (5, 3, 2) c'est-à-dire appliquer la règle :

Accorder au résultat la plus grande des tailles entières des opérandes augmentée de 1 pour une éventuelle retenue.
Accorder au résultat la plus grande des tailles décimales des opérandes. Sommer taille décimale et taille entière pour obtenir la taille totale.

Comme pour les modifications en cascade on peut imaginer d'autres règles.

Il n'est pas question cette fois d'utiliser des tailles initiales prévues par le programmeur puisque ces mots ne sont pas déclarés.

Toutefois on peut, par exemple pour les problèmes de comptabilité, borner la taille décimale ou, par suite de contraintes du langage par exemple, borner la taille totale.

Pour résoudre le problème de la détermination des tailles, il est donc

.../...

nécessaire de choisir une règle pour chaque type de relation ; ce choix dépend ici encore des problèmes traités.

La taille de X n'est pas forcément définie par une seule relation d'influence :

Il est fréquent, en programmation, qu'un identificateur soit résultat de plusieurs groupes opératoires ; par exemple, le calcul de X peut être conditionné :

si condition 1 alors X : = Y + Z
sinon si condition 2 alors X := A * B ...

La taille de X est définie par l'ensemble des relations où X est résultat.

Ainsi, si les tailles de A et de B sont respectivement (3, 2, 1) et (2, 2, 0), l'étude de la relation X = A * B peut conduire à choisir pour X la taille (5, 4, 1).

Pour que X puisse contenir à la fois les valeurs exactes de I et A * B, il lui faut pour taille (6, 4, 2).

En examinant les relations les unes après les autres, on est ainsi amené à définir pour X une taille temporaire obtenue à partir de toutes les relations déjà étudiées ; la règle de la relation d'influence utilise, entre les tailles des opérandes, la taille temporaire actuelle du résultat et lui donne éventuellement une nouvelle valeur.

La taille définitive de X sera sa taille temporaire après examen de toutes les relations d'influence où X est résultat.

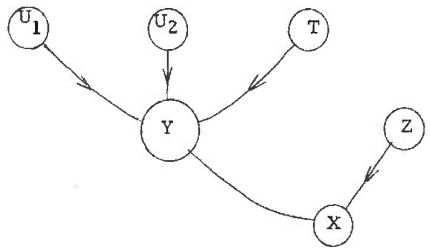
.../...

Le problème peut être plus complexe. En supposant connues les tailles de Y et Z ou de A et B, nous nous sommes placés dans un cas très favorable.

Supposons donc cette fois que la taille de Y soit également non déclarée.

On ne peut évaluer la taille de X qu'après celle de Y.

Il faut pour cela examiner les relations d'influence où Y est résultat ; elles donnent, par exemple, naissance dans le graphe des influences aux arcs suivants :



Mais parmi les opérandes U_1, U_2 ou T, certains ont peut être aussi des tailles inconnues...

On voit ainsi apparaître un nouveau cheminement dans le graphe des influences :

Il consiste à rechercher des chemins ayant pour extrémité un point fixe.

Or chercher les chemins ayant pour extrémité un point fixe dans un graphe est équivalent à chercher les chemins issus d'un point fixe dans le graphe inverse.

On constate ainsi que le calcul des tailles de données non déclarées se résoudra par les mêmes moyens que celui des modifications de tailles, il suffira d'appliquer suivant les cas l'algorithme de cheminement au graphe des influences ou à son inverse.

.../...

section 4 :

Standardisation des relations d'influence

Les algorithmes que nous venons de suggérer sont indépendants du complexe d'influences considéré :

La recherche du cheminement est la même quelle que soit l'origine des relations d'influence : descripteurs de Remora, programmes dans un langage quelconque...

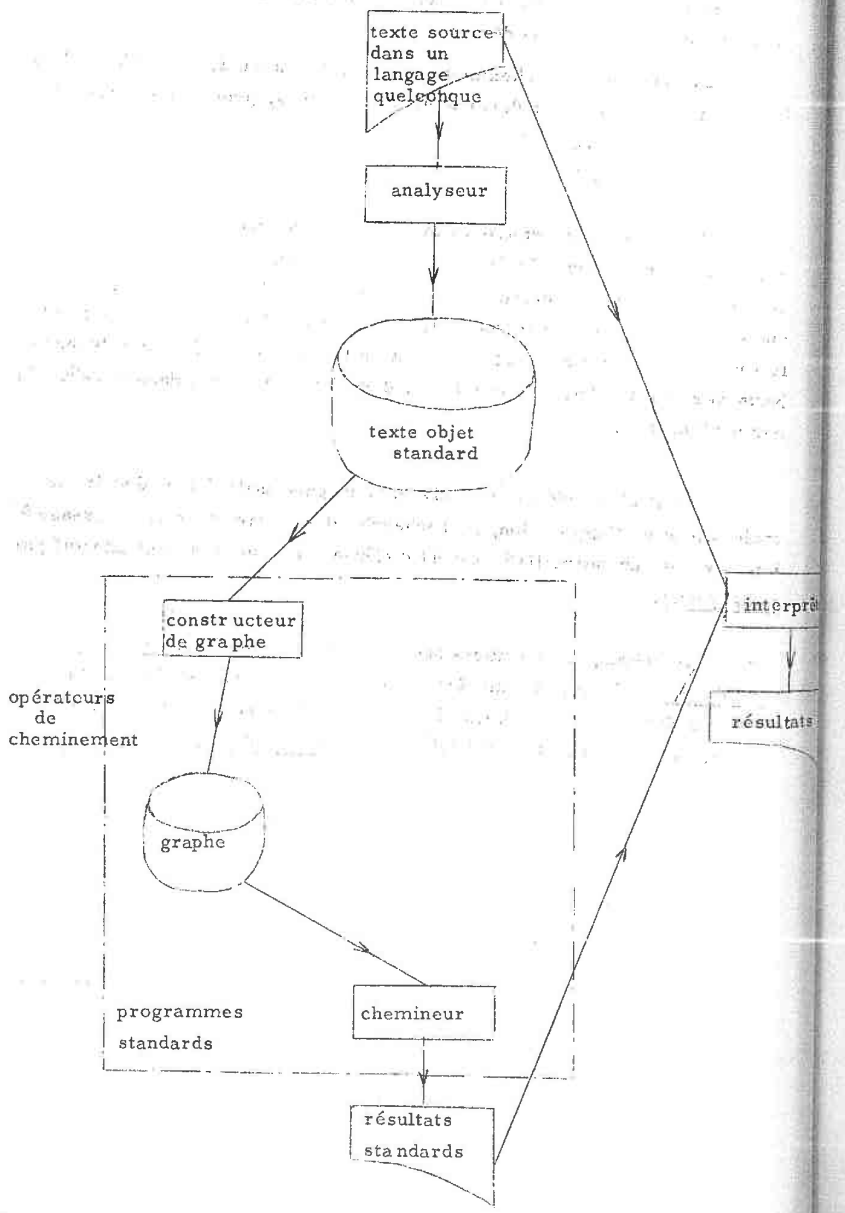
Si on veut faire du programme de cheminement un programme standard (nous l'appelerons "opérateur de cheminement") valable dans chaque cas particulier de complexe, il est nécessaire de le paramétrer ; c'est ce que nous avons fait en proposant une forme normalisée des paramètres : le vocabulaire des mots contenant leur taille et les relations d'influence Nous verrons (troisième partie) leur description physique dans le cadre de nos réalisations.

Il est bien évident que suivant le langage utilisé pour décrire les traitements de l'application, la recherche des paramètres et le passage à leur expression normalisée seront différents ; ce passage est effectué par l'analyseur.

De même, les résultats obtenus à la sortie de l'opérateur de cheminement étant sous une forme standard il est nécessaire pour les restituer dans le langage initial à l'intérieur du texte source, d'effectuer une transformation ; elle est réalisée par l'interprêteur.

.../...

Le fonctionnement de cette partie de logiciel est ainsi schématisé par :



Remarquons que, afin de dissocier la construction du graphe et la recherche du cheminement, l'opérateur de cheminement se compose en fait de deux programmes : le constructeur de graphe et le chemineur.

Analyseur et interprêteur sont spécifiques du langage source, ils devraient en principe être réécrits dans chaque cas de complexe considéré ; toutefois, si certains détails de leur réalisation pratique sont fonction de chaque cas particulier, nous avons, dans un souci de normalisation, pu mettre en évidence une charpente commune qu'il est inutile d'expliciter dans chaque cas. Cette charpente peut être écrite une fois pour toute sous forme d'un sous-programme contenu dans les opérateurs de datamatique. les "paramètres" propres à chaque cas pourront être saisis sur des descripteurs et générés par le système Remora en modules de traitements incorporables par simple copie dans le sous-programme charpente.

La réalisation pratique des générateurs de Remora se faisant en parallèle avec notre propre travail, pour nous permettre de tester les outils de datamatique, nous avons mis au point un analyseur de programmes écrits en Cobol.

Chapitre 2 :

Choix de la réalisation pratique

(1.2)

Les opérateurs de datamatique ont été programmés en Cobol comme tous les outils du système Remora.

Nous avons choisi un langage évolué couramment utilisé pour :

- sa facilité : contrairement aux programmes écrits en langage d'assemblage, les programmes Cobol sont plus lisibles donc plus accessibles ;

- sa portabilité : les programmes peuvent être repris sur un autre ordinateur que le 10070 moyennant peu de modifications.

Ce choix est celui de la majorité des SCI pour l'écriture des packages (60 % selon des enquêtes de 01 informatique) y compris de certains softwares de gestion de base de données ou SGDB.

Malgré certaines controverses Cobol est encore le langage de gestion le plus courant et l'on peut même remarquer que les langages de manipulation de données dans une base prévoient en général une interface Cobol (Socrate, IDS, Codasyl).

Il offre certaines possibilités tels que le verbe perform "qui n'existe dans aucun autre langage" (Berthet, 1971) et la sous-compilation (Ory, 1972) qui favorisent la modularité de la programmation.

.../...

Les noyaux de cheminement standards sont opérationnels et, comme nous l'avons annoncé précédemment, afin de tester les outils sur des cas concrets, nous avons été amenés à rédiger un analyseur de texte Cobol.

L'ensemble permet de calculer les tailles de n'importe quel programme Cobol : la seule ressource software demandée étant un compilateur Cobol.

Les opérateurs de datamatique constitue un outil agréable dont nous avons ressenti le besoin maintes fois pendant la réalisation de ce logiciel. La première phase de ce travail étant opérationnelle, les outils de datamatique nous serviront désormais à leur propre mise à jour.

L'enchaînement automatique des corrections ou calculs de tailles niveau d'une chaîne n'est que partiellement réalisé (absence des gestion de répertoires). Nous en verrons certains aspects dans la troisième p

DEUXIEME PARTIE :

ANALYSE DES PRINCIPES DE CONCEPTION DES OPERATEURS

PLAN DE LA DEUXIEME PARTIE

Chapitre 1

Graphe s et cheminements

- Section 1

Modifications en cascade

- Paragraphe 1

Définitions et caractéri stiques du graphe des noeuds d'influence

- Paragraphe 2

Caractéristiques et choix de cheminement

- Section 2

Evaluation de tailles inconnues

- Paragraphe 1

Graphe pseudo-inverse

- Paragraphe 2

Algorithme du cheminement

Chapitre 2

Règles de calcul des taill es

- Section 1

Typologie

- Section 2

Biais systématique

- Section 3

Utilisation des descripteurs de "Remora"

- Paragraphe 1

Description des "paramètres"

- Paragraphe 2

Charpente spécifique de la définition des formules de calcul de tailles

Chapitre 3

Concept ion de l'analyseur

Il résulte de la partie précédente que tout problème de calcul ou de maintenance des tailles dans un contexte déterminé passe par la définition du complexe d'influences, c'est-à-dire des mots et de leur caractéristique (la taille), des groupes opératoires et de leur relation d'influence (opérandes, règles, résultats).

Le complexe d'influences peut être représenté par un graphe et des lois de cheminement adéquates sur ce graphe permettront de résoudre le problème.

Nous montrerons au chapitre 2.1 quelle est la définition des graphes les mieux appropriés au problème (paragraphes 2.1.1.1. et 2.1.2.1) et quel est le cheminement le plus satisfaisant (paragraphes 2.1.1.2 et 2.1.2.2).

Nous supposerons dans ce chapitre le complexe défini sous sa forme normalisée. Les chapitres suivants reprendront les problèmes du passage au complexe normalisé, le chapitre 2.2 en particulier développera les problèmes associés au choix et à la description des règles propres à chaque application.

deuxième partie

Chapitre 1

Graphes et Cheminements

(2.1)

Pour clarifier l'exposé, et bien que les deux problèmes de création et de maintenance des tailles soient très voisins, nous étayerons les démonstrations de la section 2.1.1 par des exemples tirés du problème des modifications en cascade. Nous compléterons les éléments théoriques de cette section pour le cas de l'évaluation des tailles inconnues à la section 2.1.2.

deuxième partie
chapitre 1

section 1:

Modifications en cascade

paragraphe 1.

Définitions et caractéristiques du graphe des noeuds d'influence

(2.1.1.1)

(2.1.1.1.1) Inconvénients du graphe des influences

Donnons en tout d'abord une définition mathématique.

Soient $\mathcal{M}$ l'ensemble des mots munis d'une caractéristique et $\mathcal{R}$ l'ensemble des relations d'influence, dont les règles portent sur cette caractéristique, du complexe d'influences normalisé (que nous noterons C.I.N.) considéré.

Définitions

Soit $\overline{\Phi}$ une relation binaire sur $\mathcal{M}$ définie par :

$\forall X$ et $Y \in \mathcal{M}$ on a :

$$X \overline{\Phi} Y \Longleftrightarrow \left\{ \begin{array}{l} \exists\, r \in \mathcal{R} \text{ tel que} \\ X \text{ est opérande de } r \\ Y \text{ est résultat de } r \end{array} \right\}$$

On dira aussi que
X "influence directement" Y
ou que
Y "est directement influencé par " X.

$(\mathcal{M}, \overline{\Phi})$, graphe dont les points sont les éléments de $\mathcal{M}$ et les arcs sont définis par la relation $\overline{\Phi}$, est le graphe des influences.

Ce graphe permet de trouver tous les mots susceptibles d'être modifiés lorsque l'on corrige la caractéristique d'un mot quelconque X :

En effet, $\overline{\Phi}^{*}(X)$, qui selon les notations usuelles (Derniame-Pair, 1971) désigne les successeurs de X dans le graphe $(\mathcal{M}, \overline{\Phi}^{*})$ où Φ^{*} est la fermeture transitive de la relation $\overline{\Phi}$, est la solution de ce problème :

.../...

L'étude de la suite des impacts d'une modification de la caractéristique de X conduit à cette solution :

- $\Phi^{0}(X) = \{X\}$ par convention

- Les modifications dues à l'influence directe de X portent sur

$\Phi^{1}(X) = \Phi(X)$

- Ce nouvel ensemble influence directement les éléments de

$\Phi^{2}(X) = \Phi(\Phi(X))$

.
.
.

- Les modifications des caractéristiques de $\Phi^{i}(X)$ entraînent celles de

$\Phi^{i+1}(X) = \Phi(\Phi^{i}(X))$

.
.
.

En résumé, tout mot modifié appartient à la réunion de ces ensembles :

$$\bigcup_{n \geqslant 0} \Phi^{n}(X)$$

C'est-à-dire $\Phi^{*}(X)$

Il était normal de trouver cette solution, en effet par définition et convention :

(1) Y appartient à $\Phi^{*}(X)$

est équivalent à

(2) $X \; \Phi^{*} \; Y$

qui est équivalent à

(3) $\left\{ \begin{array}{l} \exists X_1, X_2, \ldots, X_n \; (n \geqslant 1) \text{ tels que} \\ X_1 = X ; X_n = Y \text{ et} \\ X_1 \; \Phi \; X_2 ; X_2 \; \Phi \; X_3 \ldots X_{n-1} \; \Phi \; X_n \end{array} \right\}$

ou encore à

(4) $\left\{ \begin{array}{l} X = Y \text{ ou} \\ \text{il existe un chemin dans } (\mathcal{M}, \Phi), \\ \text{d'origine } X \text{ et d'extrémité } Y \end{array} \right\}$

.../...

Ainsi les successeurs de X dans le graphe $(\mathcal{M}, \Phi^{*})$ sont les points appartenant aux chemins issus de X dans le graphe $(\mathcal{M}, \Phi)$ et nous retrouvons les problèmes de cheminement suggérés dans la première partie.

La fermeture transitive d'un graphe permet de ramener certains problèmes de cheminement à des problèmes très simples (ici : successeurs de X).

Mais la connaissance de $\Phi^{*}(X)$, ensemble des mots "influencés" par X, est insuffisante si l'on veut aussi calculer les nouvelles valeurs de leur caractéristique.

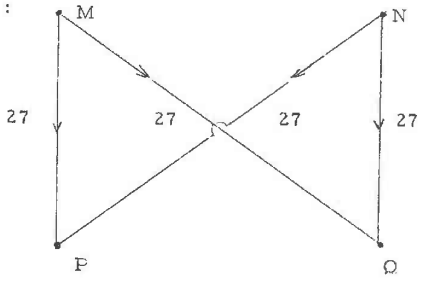
Pour résoudre ce problème en conservant le graphe des influences une solution consiste à compléter les arcs par des informations sur les règles correspondantes :

Par exemple, en associant à tout arc la liste des références aux relations d'influence correspondantes, c'est-à-dire telles que l'origine de l'arc appartienne aux opérandes et l'extrémité aux résultats.

On trouvera dans l'annexe I figure 3a) un exemple de graphe des influences, ainsi complété, obtenu à partir du complexe d'influences de la figure 2 c) associé au programme de la figure 1 a) et b) ; les relations d'influence ont été numérotées et ces numéros sont portés sur les arcs.

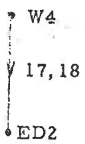
On peut remarquer sur cette figure :

- qu'à plusieurs arcs correspondent une même relation d'influence ; par exemple, les quatre arcs (M, P), (N, P), (M, Q) et (N, Q) correspondent à la relation 27 :



.../...

- qu'à un arc peut correspondre plusieurs relations d'influence ; c'est-à-
qu'il faut effectivement associer à un arc une liste de références ; par
exemple, l'arc (W4, ED2) correspond aux relations d'influence 17 et 18 ;



Bien que le graphe des influences ainsi modifié, (ses arcs sont c
tués par les triplets : origine, liste de références, extrémité), permette
résoudre le problème de la maintenance des tailles, les remarques préc
dentes font ressortir un certain nombre d'inconvénients (répétition, liste
variable de références). Ces inconvénients, surtout lorsque l'on cherche
automatiser le procédé, sont tels que nous avons été conduits à abandonn
graphe au profit d'un graphe mieux adapté.

Ce nouveau graphe, sera appelé graphe des noeuds d'influence
que ses principaux noeuds ou sommets sont associés aux relations d'influ
du C. I. N., nous le définirons en deux étapes, après introduction d'un a
graphe que nous appelons graphe bialterné.

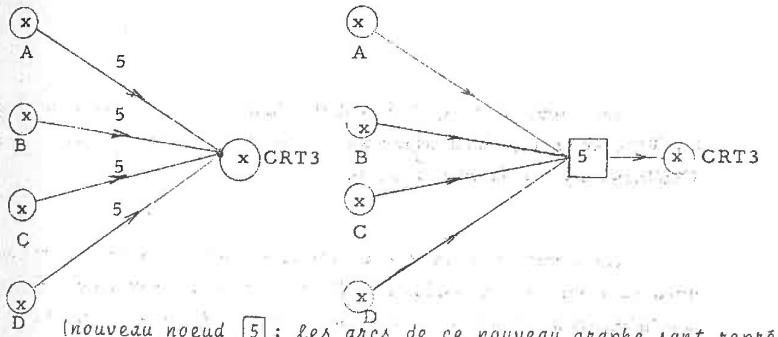
Remarquons que ces étapes sont facultatives dans la mesure où e
n'ont pour objectif que de développer le raisonnement logique qui nous a
d'aboutir à la définition du graphe des noeuds d'influence. Une fois obten
cette définition (2.1.1.1.3) le passage du C. I. N. au graphe pourra se fai
directement.

(2.1.1.1.2) **Définition et transformations du graphe bialterné**

Pour éviter la redondance précédente tout en conservant les informations attachées aux arcs, qui sont indispensables, une solution consiste à les attacher à de nouveaux noeuds : c'est l'idée du graphe bialterné :

par exemple :

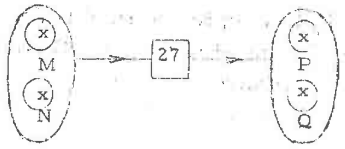
graphe des influences — graphe bialterné



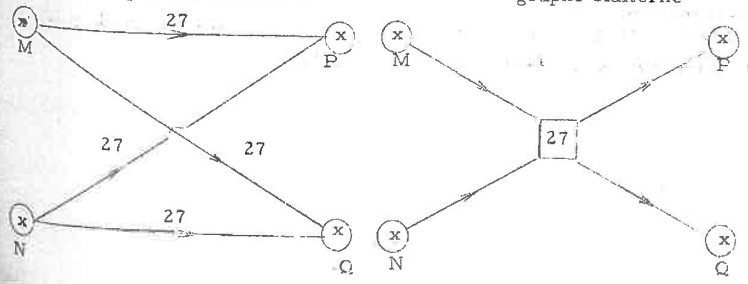
(nouveau noeud [5] ; les arcs de ce nouveau graphe sont représentés par les flèches)

La ressemblance de la représentation schématique de ce graphe avec celle de l'hypergraphe orienté, que nous avons défini dans la première partie, est frappante :

exemple : hypergraphe orienté



graphe des influences — graphe bialterné



On peut ainsi fournir une règle permettant d'obtenir la représentation schématique du graphe bialterné à partir de celle de l'hypergraphe orienté associé au C. I. N. :

supprimer le contour encerclant les opérandes (respectivement résultats) d'une relation d'influence et remplacer la flèche qui joignait ce contour à la règle par autant de flèches de même sens joignant chaque opérande (respectivement chaque résultat) à la règle de la relation .

Les figures 2 c), 3 a) et 4 de l'annexe 1 permettent de comparer les différentes représentations sur l'ensemble des mots et des relations d'influence de cet exemple d'appui.

On remarquera que les noeuds de ce nouveau graphe proviennent d'une part des mots du C. I. N. et ils correspondent aux noeuds du graphe des influences (nous les appelerons noeuds de 1ère espèce), d'autre part des règles des relations d'influence et ils correspondent alors aux informations associées aux arcs du graphe des influences (nous les appelerons noeuds de 2ème espèce).

A tout noeud de 1ère espèce ne succèdent que des noeuds de 2ème espèce et réciproquement ; aussi tout chemin (1) est constitué de noeuds pris alternativement dans chacune des deux espèces. C'est la raison de la dénomination : " graphe bialterné "

(1) un chemin peut être défini par la suite de ses arcs ou par la suite de ses points pour un 1-graphe (Berge, 1970)

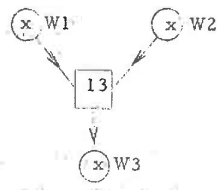
. . . /. . .

Ce graphe est un graphe biparti selon la définition très générale donnée par Berge, mais pas selon la définition plus restrictive formulée par d'autres auteurs (Roy, 1969 ; Kaufmann, 1968).

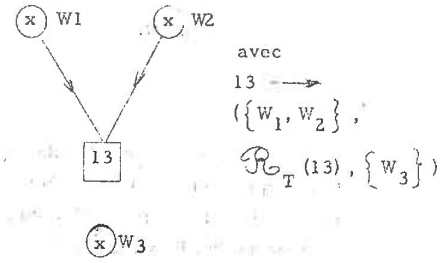
Un noeud de 2ème espèce est, en fait, une référence à un triplet (opérandes, règles, résultats) ; il donne donc la liste de ses successeurs de 1ère espèce (les résultats) ; les arcs ayant pour origine un noeud de 2ème espèce et pour extrémité un noeud de 1ère espèce sont, sous ce point de vue, des informations redondantes.

Si l'on songe à une construction sur ordinateur d'un tel graphe, il semble normal de supprimer cette redondance en choisissant une représentation comme celle de l'exemple suivant :

graphe bialterné

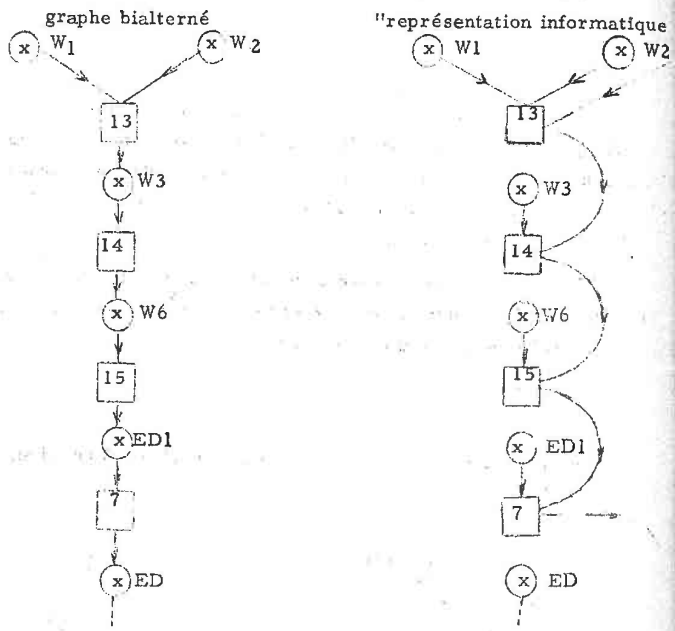


"représentation informatique"



. . . /. . .

Pour des commodités de réalisation, il est également intéressa relier par des arcs les noeuds de 2ème espèce qui se "succèdent" par l'intermédiaire d'un noeud de 1ère espèce, comme le montre l'exemple ci-après :



Notons que les arcs du type (x)→□ qui relient les noeuds de 1ère espèce aux noeuds de 2ème espèce doivent être maintenus pour une seule raison : l'initialisation du processus de cheminement. Ensuite ils deviennent inutiles puisque l'on passera toujours d'un noeud de 2ème es à un autre noeud de 2ème espèce :

exemple : on modifie la taille de W1 ;
W1 est opérande dans [13] , car [13] succède à W1 ;
on exécute la règle [13] et la taille du résultat W3 est alors modifiée ;
la règle [14] succède à la règle [13] ;
on exécute la règle [14] et la taille de W6 est modifiée
la règle [15] succède à la règle [14] ;
on exécute la règle [15] et ED1 est modifiée
...

L'idée vient donc de limiter la représentation des noeuds de 1ère espèce au sous-ensemble $\mathcal{N}_6'$ indispensable au démarrage des cheminements que l'on est susceptible d'envisager :

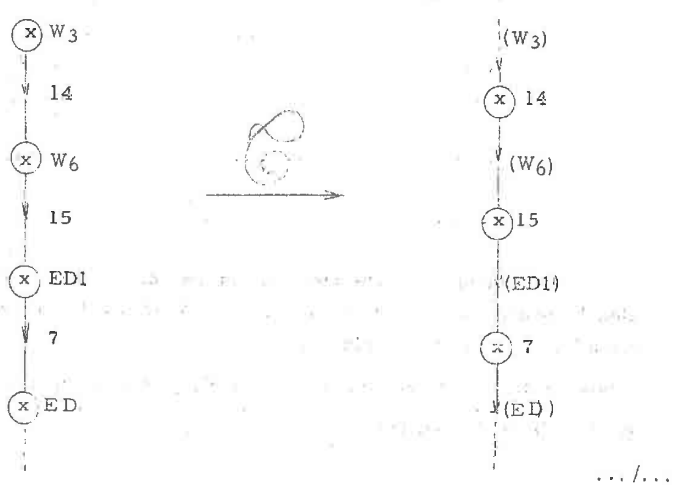
La représentation du graphe pourra alors être réduite à celle du sous-graphe dont les points sont les éléments de $\mathcal{N}_6'$ et $\mathcal{R}$.

Il est évident qu'une telle réduction prend tout son intérêt (économie de place) dans un système automatique.

La figure 6a) de l'annexe I présente un tel sous-graphe où $\mathcal{N}_6'$ est réduit à la liste des variables d'entrée.

Indépendamment des noeuds de $\mathcal{N}_6'$ on remarque pour les noeuds restants (relations d'influence) que le passage du graphe des influences au sous-graphe de la "représentation informatique" se fait par une transformation où l'on associe aux arcs du premier graphe des noeuds dans le second et aux noeuds du premier des arcs du second :

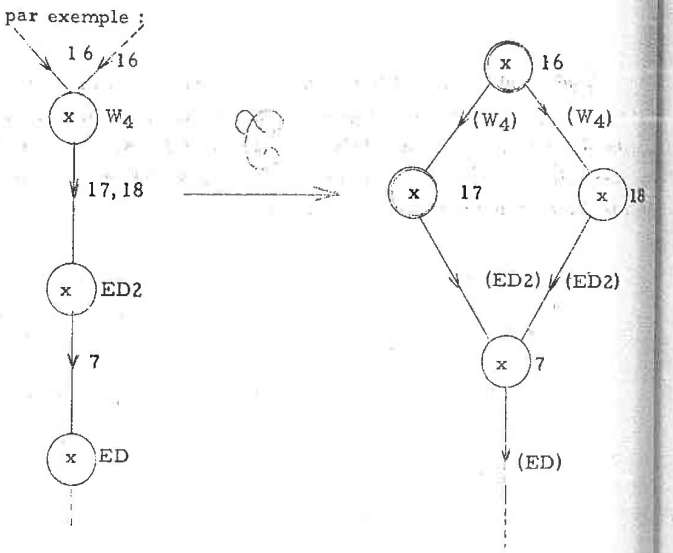
par exemple :



.../...

Kaufmann présente une transformation du même type pour calcul
la classe chromatique d'un graphe en montrant qu'elle est égale au nomb
chromatique du graphe transformé (Kaufmann tome 2 page 284).

La transformation précédente en diffère par le fait, que Kaufma
l'applique aux arêtes d'un graphe non orienté, mais surtout que plusieu
arcs associés à la même relation ne donnent qu'un point dans le nouveau
graphe et qu'un seul arc peut donner plusieurs points s'il porte une liste
de références :

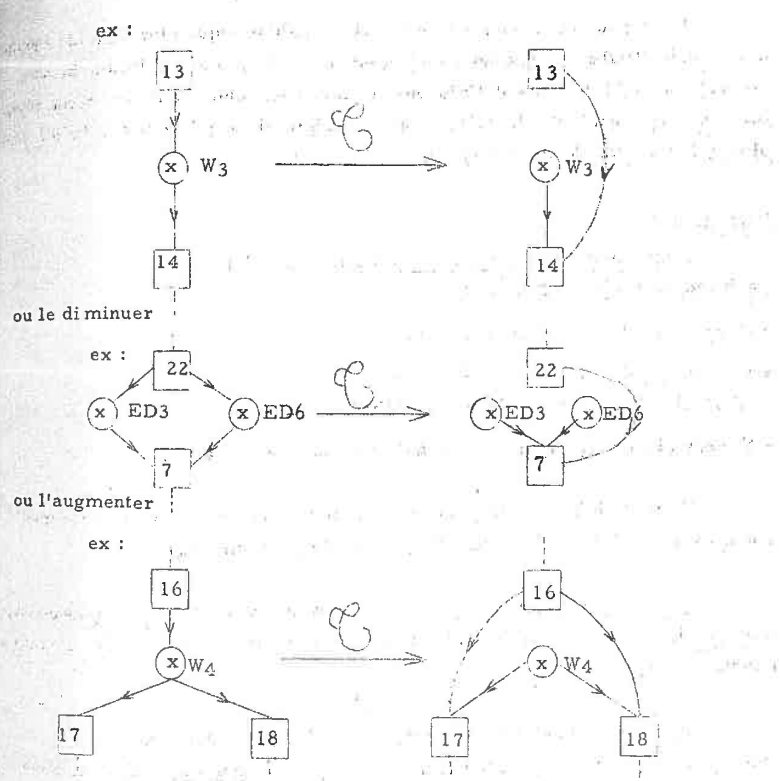


Le graphe que nous appelons graphe de la "représentation infor
tique" auquel nous sommes parvenus ($\mathcal{M}' \neq \emptyset$) est le résultat d'une
transformation du graphe bialterné :

- laissant le nombre de points, c'est-à-dire, l'ordre du graphe inchangé
($\mathcal{M}' = \mathcal{M}$) ou inférieur ($\mathcal{M}' \subset \mathcal{M}$),

.../...

- modifiant éventuellement le nombre d'arcs ; la transformation peut en
effet le laisser invariant



Les figures 4 et 5 de l'annexe 1 permettent une comparaison
plus globale.

Nous appelons le graphe de la "représentation informatique" ainsi
obtenu, graphe des noeuds d'influence et nous allons en donner une définition
mathématique qui permettra sa construction directe à partir du complexe
d'influences.

(2.1.1.1.3) **Définition du graphe des noeuds d'influence**

Pour pouvoir donner une définition mathématique classique du gra des noeuds d'influence par un couple (ensemble de points, relation binair sur cet ensemble), nous définissons de nouveaux objets qui ont même st ture que les noeuds de deuxième espèce (triplets) et qui sont destinés à r placer les noeuds de première espèce (mots) :

Définitions

Nous appelons noeud de définition tout triplet (opérandes, règles, résultats) où

- l'ensemble des opérandes est vide ;
- l'exécution des règles est sans effet : il s'agit d'une règle spéciale de définition ;
- l'ensemble des résultats est réduit à un seul mot m de $\mathcal{M}$.

Cette définition permet de ne manipuler qu'un seul type de noeuds constitués par des triplets (opérandes, règles, résultats).

Soit $\mathcal{D}$ l'ensemble de tous les noeuds de définition différents cons à partir des mots $\mathcal{M}$ d'un C. I. N. Il est évident que $\mathcal{D}$ et $\mathcal{M}$ sont en cor pondance biunivoque.

Si $\mathcal{R}$ est l'ensemble des relations d'influence du C. I. N., soit $\mathcal{N}$ l'union de $\mathcal{R}$ et de $\mathcal{D}$; : nous appelons noeuds d'influence les éléments

Définissons la relation Γ sur l'ensemble $\mathcal{N}$ des noeuds d'influen associés à un C. I. N. par :

$\forall n$ et $m \in \mathcal{N}$

$n \Gamma m$ est équivalent à $\left\{ \begin{array}{l} \exists\ x \in \mathcal{M} \text{ tel que} \\ x \text{ est résultat de } n \\ x \text{ est opérande de } m \end{array} \right\}$

.../...

$(\mathcal{N}, \Gamma)$, graphe dont les points sont les éléments de $\mathcal{N}$ et les arcs sont définis par la relation Γ est le graphe des noeuds d'influence.

On peut facilement vérifier que ce graphe est isomorphe (Steen, 1968) au graphe dit de la "représentation informatique" que nous avons obtenu au paragraphe précédent par une suite de transformations logiques à partir du graphe bialterné :

En effet, d'une part les deux ensembles de points sont isomorphes puisque les mots ont simplement été remplacés par les noeuds de définition ; d'autre part, si un arc relie deux points dans le premier graphe, un arc relie les points correspondants dans l'autre graphe, et réciproquèment :

- Un premier type d'arc ($x \xrightarrow[W_6]{} 15$) joint tout mot x de $\mathcal{M}$ aux relations d'influence où il est opérande ; or à tout mot x correspond un noeud de définition dx où x est résultat.

Comme $d_x \Gamma Y \Longleftrightarrow \left\{ \begin{array}{l} \exists\ x \in \mathcal{M} \text{ tel que} \\ x \text{ est résultat de } dx \\ x \text{ est opérande de } Y \end{array} \right\}$

et comme x est le seul résultat de dx : un arc joint toute définition dx aux relations d'influence où x est opérande ($d(W_6) \rightarrow 15$)

- Un deuxième type d'arc joint deux noeuds de deuxième espèce (ou relations d'influence) s'ils se "succédaient" dans le graphe bialterné par l'intermédiaire d'un noeud de 1ère espèce (ou mot) c'est-à-dire, si un mot est résultat d'une relation d'influence et opérande de l'autre, ou encore s'ils sont reliés par la relation Γ, qui précisémment définie les arcs du 2ème graphe.

- Il n'y a pas d'arcs ayant pour extrémité un mot ni un noeud de définition, car l'ensemble des opérandes de tout noeud de définition est vide ; de même il n'y a pas d'arcs joignant deux mots ni deux noeuds de définition.

Les deux graphes sont donc isomorphes et auront la même représentation schématique.

(2.1.1.1.4) Graphe représentatif des arcs d'un hypergraphe orienté

Le graphe des noeuds d'influence est une représentation du C.I

Nous avons montré dans la première partie (1.1.2) qu'une certa représentation schématique d'un complexe d'influence était en fait celle hypergraphe orienté.

Peut-on lier le graphe des noeuds d'influence aux graphes repré tatifs des hypergraphes ? La réponse à cette question est l'objet de ce paragraphe.

Rappelons les différentes définitions que donne Berge (1970) p aboutir à la notion de graphe représentatif des arêtes d'un hypergraphe.

Définition

Soit $X = \{ x_1 , x_2 , \ldots , x_n \}$ un ensemble fini, et

$\mathcal{E} = (E_i / i \in I)$ une famille de parties de X.

On dira que $\mathcal{E}$ constitue un hypergraphe sur X si l'on a

(1) $E_i \neq \emptyset \quad \forall i \in I$

(2) $\bigcup_{i \in I} E_i = X$

L'hypergraphe est représenté par le couple $(X, \mathcal{E})$.

Les x_i sont les points de l'hypergraphe.

Les E_i sont les arêtes de l'hypergraphe.

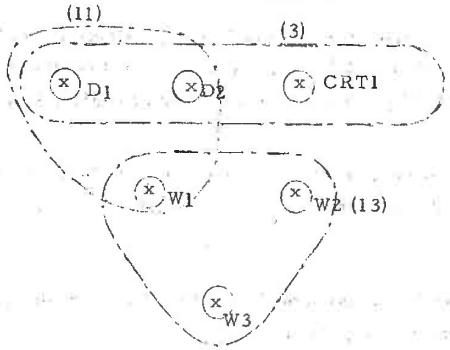
.../...

Définitions

Deux sommets d'un hypergraphe sont adjacents s'il existe une arête E_i qui les contient tous les deux.

Deux arêtes E_i et E_j sont adjacentes si leur intersection est non vide.

exemple :



Les sommets D_1 et D_2 sont adjacents parce qu'ils appartiennent tous deux à l'arête 3 ou à l'arête 11.

Les arêtes 3 et 11 sont adjacentes, il en est de même pour 11 et 13 ; mais 3 et 13 ne le sont pas, elles n'ont aucun point commun.

Définition

Soit $H = (X , \mathcal{E})$ un hypergraphe.

On appelle graphe représentatif des arêtes d'un hypergraphe le couple $G = (E, \Phi)$ où

- $E = \{ e_i ; i \in I \}$: chaque point e_i de E correspond à l'arête E_i de $\mathcal{E}$.
- la relation binaire Φ est définie par :

pour $i \neq j$: $e_i \Phi e_j \Longleftrightarrow E_i \cap E_j \neq \emptyset$

.../...

Le graphe représentatif des arêtes d'un hypergraphe H est do
graphe dont chaque point correspond à une arête de H et où une arê
relie deux points qui correspondent à deux arêtes adjacentes de H.

On peut remarquer que dans le cas particulier où l'hypergraphe
un graphe simple, le graphe représentatif des arêtes d'un graphe est iso
au graphe transformé présenté par Kaufmann mentionné au paragraphe
(2.1.1.1.2).

La figure 8 de l'annexe 1 présente le graphe représentatif des a
de l'hypergraphe de la figure 7.

Reprenons l'extension de la notion d'hypergraphe que nous avon
proposé dans la première partie.

Définition

Soit $X = \{x_1, x_2, \ldots, x_n\}$ un ensemble fini, et $\mathcal{E} = (E_i / i$
une famille de parties de $\mathcal{P}(X) \times \mathcal{P}(X)$.

On dira que $\mathcal{E}$ constitue un hypergraphe orienté sur X si l'on

(1) $E_I \neq (\emptyset, \emptyset) \quad \forall i \in I$

(2) si $E_i = \left(\{ o_{j_i} ; j_i \in J_i \}, \{ e_{j'_i} ; j'_i \in J'_i \} \right)$

$$\bigcup_{i \in J} \left(\bigcup_{j_i \in J_i} o_{j_i} \bigcup \bigcup_{j'_i \in J'_i} e_{j'_i} \right) = X$$

L'hypergraphe orienté est représenté par le couple $(X, \mathcal{E})$
Les x_i sont les points de l'hypergraphe orienté.
Les E_i sont les arcs de l'hypergraphe orienté :
Les origines de l'arc E_i sont les o_{j_i}
Les extrémités de l'arc E_i sont les $e_{j'_i}$.

.../...

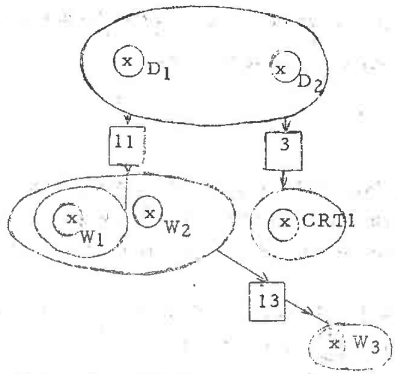
Compte tenu de cette définition étendons la notion de graphe représentatif des arêtes d'un hypergraphe à celle de graphe représentatif des arcs d'un hypergraphe orienté.

Définitions

Deux sommets d'un hypergraphe orienté sont adjacents s'il existe un arc E_i qui les contient tous les deux, comme origines et/ou comme extrémités.

Deux arcs E_i et E_j sont adjacents s'ils ont au moins un point commun, comme origine et/ou comme extrémité.

exemple :



Les sommets D_1 et D_2 sont adjacents.
Les arcs 3 et 11 sont adjacents, il en est de même pour 11 et 13, mais 3 et 13 ne le sont pas.

Si l'hypergraphe considéré est un graphe, on retrouve bien la notion d'adjacence dans les graphes orientés (Berge, 1970 ; Kaufmann, 1968).

.../...

Définition

Nous dirons que l'arc E_j "succède" à l'arc E_i si un point e… de E_i est origine de E_j.

Par exemple, l'arc I3 succède à l'arc I1

Définition

Soit $H = (X, \mathcal{E})$ un hypergraphe orienté où chaque arc E_i es… au couple $(\mathcal{O}_i, \mathcal{E}_i)$.

Nous appelons graphe représentatif des arcs d'un hypergraphe or… le couple $G = (E, \Phi)$ où

- $E = \{ e_i ; i \in I \}$: chaque point e_i de E correspond à l'arc…
- la relation binaire Φ est définie par

$$e_i \Phi e_j \Leftrightarrow \mathcal{E}_i \cap \mathcal{O}_j \neq \emptyset$$

Le graphe représentatif des arcs d'un hypergraphe orienté H e… le graphe orienté dont chaque point correspond à un arc de H et où un … relie 2 points s'ils correspondent à 2 arcs qui se "succèdent" dans H. (1)

Compte tenu des définitions précédentes, il est immédiat que le sq… graphe du graphe des noeuds d'influence $(\mathcal{R}_\mathcal{O}, \Gamma)$ est isomorphe du … représentatif des arcs de l'hypergraphe orienté de même représentation … schématique que le complexe d'influences $(\mathcal{M}, \mathcal{R}_\mathcal{O})$ (cf. 1.1.2) :

En effet, les points de ces deux graphes sont associés aux éléme… de $\mathcal{R}_\mathcal{O}$ et la relation "succède" entre les arcs de l'hypergraphe corresp… la relation Γ entre les relations d'influence du complexe d'influences de … représentation schématique.

(1) La généralisation aux hypergraphes orientés de la notion de graphe re… tatif aurait pu comporter des arcs correspondants non seulement aux arc… "succèdent" dans H, mais plus généralement aux arcs qui sont "adjacents" … dans H par analogie avec la définition non orientée, mais cette généralis… ne convenait pas ici.

.../...

Il est intéressant de constater que le graphe des noeuds d'influence obtenu par un raisonnement logique a, en fait, un fondement théorique attaché aux notions de complexe d'influences et d'hypergraphe associé.

Nous aurons besoin dans la suite de la notion de cheminement dans les hypergraphes orientés.

Donnons en une définition :

Définition

Nous dirons qu'une suite d'arcs $(E_1, E_2, \ldots, E_n)$ $(n \geq 1)$ d'un hypergraphe $(X, \mathcal{E})$ constitue un chemin dans l'hypergraphe si $\forall i, 2 \leq i < n$, E_i "succède" à E_{i-1}.

Si l'hypergraphe considéré est un graphe, on retrouve bien la notion de chemin dans un graphe.

En outre, et c'est ce qui est le plus intéressant, l'ensemble des chemins d'un hypergraphe et l'ensemble des chemins du graphe représentatif des arcs de l'hypergraphe, chemins définis, cette fois, par une suite de points (1), sont isomorphes :

$$(E_1, E_2, \ldots, E_n) \longleftrightarrow (e_1, e_2, \ldots, e_n)$$

$$E_j \text{ "succède à " } E_i \longleftrightarrow e_i \Phi e_j$$

Ainsi, tout problème de cheminement dans un hypergraphe orienté peut être ramené à celui du cheminement dans le graphe représentatif des arcs de l'hypergraphe.

(1) pour un graphe simple, il est indifférent de définir les chemins par des suites d'arcs ou par des suites de points.

.../...

Le cheminement pressenti dans la première partie est en fait
chemin ement sur l'hypergraphe associé au complexe d'influences, il e
naturel d'utiliser le graphe représent atif de cet hypergraphe étendu p
démarrage des che minements au grap he des noeuds d'influence.

Le problème pratique de calcul ou de maintenance de tailles fie
partie d'une classe beaucoup plus générale de problèmes faisant app
notion s de c omplexe d'influences et de cheminement généralisé à l'hy
graphe orienté associé.

(2.1.1.1.5) **Caractéristiques du graphe**

oucles et circuits

Le graphe des noeuds d'influence, défini au paragraphe 2.1.1.1.3, par
e couple (ω, Γ) peut comporter des boucles et des circuits :

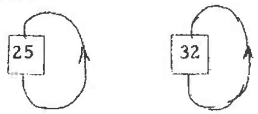
Il comporte des boucles chaque fois qu'un noeud n vérifie la relation
$n \Gamma n$, c'est-à-dire chaque fois qu'il existe un mot x qui est à la fois résultat
et opérande de n . Des relations d'influence de ce type apparaissent
ans des C.I.N. issus de programmes contenant des instructions du type :

I : = I + 1

ou

S : = S + Q

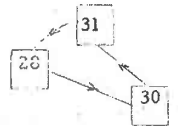
où des boucles du type de celles que l'on peut remarquer sur l'exemple, figure 5
ans l'annexe 1 :



Nous proposerons une solution à ce problème (2.1.1.2) lorsque les boucles
ortent sur des "instructions", mais nous pouvons déjà remarquer qu'il est plus
élicat au niveau d'un C.I.N. constitué de macro-influences : par exemple, si un
rogramme manipule des données "reprises" (Reix, 1971) appartenant à un fi-
hier mis à jour, de telles données étant à la fois opérande et résultat d'une macro-
nfluence, c'est au niveau du programme qu'il y a "boucle". C'est donc à ce
iveau qu'il faudra prévoir un test d'arrêt pour les itérations du cheminement.

Le graphe comporte aussi des circuits : un circuit est un chemin dont
'origine du premier arc et l'extrémité du dernier sont confondues.

Ainsi, sur l'exemple d'appui de l'annexe 1 :



st un circuit dû à la séquence d'instruction

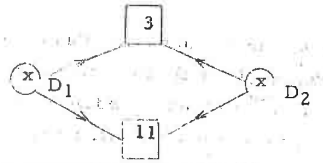
28. P ⟶ TRV
30. TRV-5000 ⟶ Q
31. Q ⟶ P

En général, il y a circuit lorsque le calcul de la nouvelle valeur d'un
ot décomposé en plusieurs groupes opératoires fait intervenir son ancienne
aleur ; c'est-à-dire lorsqu'on manipule des variables au sens strict (Pair, 1974).

Cycles

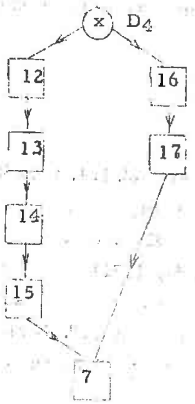
Un cycle est une chaîne fermée, c'est-à-dire un circuit du graphe orienté. Les boucles et les circuits sont des cas particuliers de cycles.

Il y a de nombreux cycles sur la figure de l'exemple d'appui, par exemple :



En fait, pour le problème de la maintenance des tailles, un seul de cycle a une incidence sur la recherche du cheminement ; ce sont les où deux chemins issus d'un même point ont un second point en commun.

Par exemple, sur la figure 5 de l'annexe 1 :



Définitions

Nous dirons que deux chemins distincts sont en dérivation s'ils ont même origine et même extrémité.

Nous appelerons doublet tout cycle constitué par deux chemins en dérivation et fermeture de doublet l'extrémité commune à ces deux chemins.

Le doublet pose en effet le problème de la propagation simultanée des modifications de tailles sur les deux chemins en dérivation avant d'atteindre la fermeture du doublet.

(2.1.1.2.1) Caractéristiques du cheminement

Notion d'ordre

Définition

Le préordre associé à un graphe (Kaufmann, 1968) est la relation binaire réflexive et transitive définie par

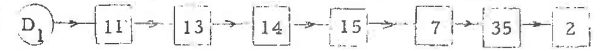
$x_i \leq x_j \Leftrightarrow x_i = x_j$ ou il existe un chemin de x_i à x_j.

(cf. la définition de la fermeture transitive de la relation binaire qui définit le graphe).

Sur un chemin sans circuit, ce préordre permet de classer les points de l'origine vers l'extrémité dans l'ordre de leur position : tout point n'apparaît qu'après ses prédécesseurs.

Dans le problème des modifications en cascade, c'est sur le graphe muni de son préordre que le cheminement [1] doit avoir lieu ; en effet, seule l'exécution des règles de calcul des tailles en un point où un opérande a été lui-même modifié (directement ou comme résultat du noeud d'influence qui précède) provoque la modification de tailles d'un résultat, c'est-à-dire d'un opérande des points qui "suivent".

Par exemple, considérons sur l'exemple d'appui le chemin



et supposons que la taille de D_1 soit modifiée ; pour obtenir une

(1) cheminement = action de cheminer (Larousse)
problème de cheminement = problème portant sur les chemins d'un graphe (Derniame -Pair, 1971)
Nous appelerons ici cheminement, l'action d'examiner tour à tour les points de certains chemins d'un graphe

.../...

taille correcte en [2] il faut exécuter les règles dans l'ordre de l
vers la droite ; ainsi en supposant que le calcul des tailles soit simul
la recherche des chemins (1), tout cheminement qui examinerait, pa
[14] avant [11] et [13] ne modifierait pas convenablem
tailles, puisque : aucun opérande n'ayant été modifié, la règle [14]
sans effet ; aucun opérande de [15] n'est donc modifié ; les règ
et [13] propagent normalement les modifications issues de D1,
règles appliquées en [15] et en ses successeurs sont sans effet pu
les opérandes n'y sont pas modifiés.

Nous retiendrons l'hypothèse que le calcul des tailles est sim
la recherche des chemins et nous énonçons alors la règle de chemine
suivante :

Règle de cheminement 1 :

Tout noeud doit être considéré avant ses successeurs

(1) C'est évidemment cette hypothèse que nous conserverons puisqu'elle évite de stocker en mémoire les images des chemins construits comme le nécessitent d'autres problèmes de cheminement.

Traitements des cycles

Par définition, la règle énoncée précédemment ne s'applique qu'à des graphes sans circuit. Or les graphes sur lesquels nous travaillons peuvent comporter, nous venons de le voir, des circuits. Le cheminement automatique doit donc les détecter sous peine de n'avoir pas de terminaison ; c'est ce que nous résumerons par la règle :

Règle de cheminement 2 :

Tout cheminement doit détecter les boucles et les circuits.

En général, pour éliminer ce problème, les algorithmes de cheminement transforment le graphe en un graphe sans circuit "par ouverture des fermetures de circuit", en supprimant l'arc qui "ferme le circuit" ou en ajoutant des points synonymes (Gadefait, 1972) ; l' "ouverture" ne modifiant pas nécessairement l'image du graphe : par exemple, adjonction d'un booléen à chaque point du graphe pour indiquer les points déjà rencontrés par le cheminement (Derniame-Pair, 1971).

Nous conserverons ce dernier type d'ouverture et tenant compte de la spécificité du problème de la maintenance des tailles nous répercuterons en partie le traitement des circuits sur la définition des règles.

Etudions les divers cas particuliers :

Les boucles

Elles sont dues généralement à des cumuls, par exemple à l'affectation $S := S + Q$.

La taille de S ne peut être parfaitement définie que si l'on connaît le nombre de fois où l'instruction est exécutée pour une exploitation du programme ; chose que l'on ignore en général sauf dans quelques cas particuliers (boucles fixées à l'avance) à l'examen du C.I.N. ou à la lecture du texte source.

.../...

On peut supposer cependant (et ceci seulement dans le problè
maintenance des tailles) que le programmeur a choisi, après une es
du volume des données manipulées, une définition initiale "correcte"
Si l'estimation des volumes changent, S risque d'être choisi corre
de modifications en cascade et sa nouvelle taille est imposée ; sinon
modifié, l'idée vient, pour les cumuls de cette espèce, d'opérer par
"variation relative" des tailles, c'est-à-dire de calculer les nouvelle
des résultats à partir des anciennes et des écarts de variation des
opérandes.

Il suffit alors d'appliquer une fois les règles qui comportent
tement spécifique des boucles : le cheminement ignore donc les boucl

Remarquons que cette solution devra être revue dans le prob
des tailles inconnues.

Les circuits

De multiples combinaisons de groupes opératoires peuvent en être la
par exemples

1 - permutation de 2 valeurs

```
TRV:=A
A:=B
B:=TRV
```

2 - cumul "caché"

```
SP:=S+Q
S:=SP
```

3 - mémoire de travail utilisée à plusieurs reprises

etc...

Ce problème se rapproche du précédent et nous conserverons
même solution ("variation relative" au sein des règles) : il suffit alors
parcourir une fois chaque circuit, puis de propager les modifications
les successeurs de la fermeture du circuit.

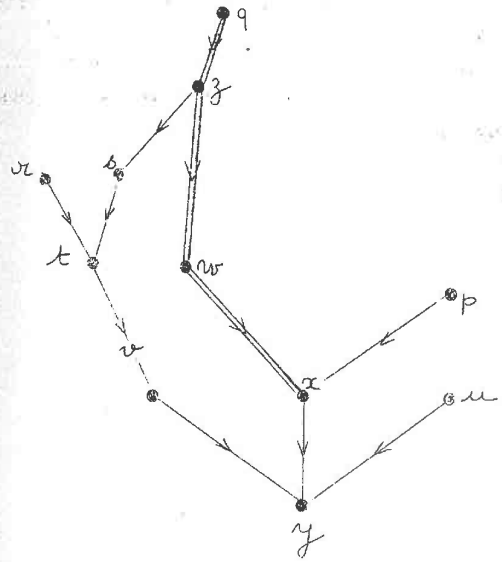
Cette solution appelle la même remarque que précédemment.

...

Les doublets

La solution dans ce cas paraît simple, il suffit de s'assurer, avant d'exécuter les règles associées à la fermeture du doublet, que l'on a bien parcouru les deux chemins en dérivation. Si cette méthode est évidente en apparence en particulier lorsque l'on chemine manuellement sur la représentation graphique en repérant "à l'oeil" les doublets, elle se complique lorsque l'on passe au traitement automatisé puisqu'elle exige la connaissance à tout moment de tous les chemins ayant pour extrémité le point examiné susceptible d'être fermeture de doublet.

Par exemple, supposons que l'on examine le point y comme successeur de α ; le dessin ci-dessous présente les chemins ayant pour extrémité y :



arcs suivis pour atteindre α

.../...

Avant d'appliquer les règles associées au noeud y, il faut vé
que parmi les chemins ayant pour extrémité y, il n'y en a aucun qui,
comporter α , ont des points communs avec des chemins ayant pour
trémité α : c'est le cas ici pour le chemin (q, z, s, t, v, y) qui a
points communs q et z avec le chemin (q, z, w, α) ; y est ferm
doublet et il faut assurer les propagations sur le morceau du deuxièm
chemin en dérivation non encore examiné (s, t, v), puis appliquer le
en y.

Un tel traitement demande la représentation en mémoire de to
chemins joignant deux points quelconques du graphe et, bien qu'il exis
nombreuses possibilités pour effectuer cette construction de chemins
me-Pair, 1971 ; Roy, 1969), nous avons préféré une solution plus éco
par "marquage", commune aux doublets et aux circuits, que nous dév
au paragraphe suivant.

(2.1.1.2.2) Choix du cheminement

Parmi tous les algorithmes de recherches de chemins (Warshall ; Dantzig ; Ford...), notre choix est fort limité par les caractéristiques que nous venons de préciser ; d'une part la présence de circuits en élimine la plus grande partie (par exemple : algorithme fondé sur un ordre des arcs ; Derniame-Pair, 1971) et d'autre part les algorithmes existants satisfaisant à la règle de cheminement 2 doivent être adaptés à notre problème (par exemple pour le traitement des doublets).

C'est le cas de deux algorithmes que nous citons :

- "La recherche de chemins de p arcs d'un sous-graphe X' à tout point du graphe " (Derniame-Pair pages 20 et 120) ; l'adaptation de cet algorithme, basé sur le principe de récurrence, à notre problème, nous a conduit à utiliser une structure de file : nous l'appelerons "algorithme de la file".

- "L'emploi d'une pile pour étudier l'existence d'un chemin joignant un point α donné aux divers points y " (Derniame-Pair page 121) ; adapté à notre cheminement, cet algorithme sera appelé, sans ambiguité dans notre contexte , "algorithme de la pile".

L'adaptation nécessite dans les deux cas la technique du "marquage" (*) que nous avons choisie pour traiter les circuits et les doublets :

(*) du type de celle employée en recherche opérationnelle, par exemple pour les problèmes de flots :
Ford et Fulkerson, 1967 ; Faure, 1971 ; Desbazeille, 1972)

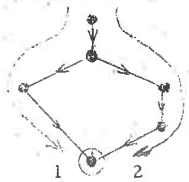
.../...

Le marquage

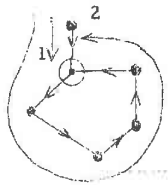
Il consiste à associer à chaque point du graphe une "marque" (par exemple sous forme d'un tableau d'entiers). Cette marque indique l'état du point qui peut être :

- non encore examiné ;
- examiné une fois ;
- fermeture de doublet ou de circuit examinée deux fois ou plus mais la propagation finale n'a pas été plus loin ;
- fermeture de doublet ou de circuit dont les successeurs ont été examinés après la dernière exécution des règles associées.

Initialement tous les points sont "non encore examinés" ; puis ils peuvent devenir "examinés" ; si le cheminement les rencontre à nouveau c'est que deux chemins issus de l'origine du cheminement se rejoignent en ces points :
ils sont fermetures de doublet :



ou de circuit :



On y applique une nouvelle fois les règles sans poursuivre une nouvelle le cheminement vers les successeurs, ce qui ferait "boucler" l'algorithme en cas de circuit. Cependant on affecte à ces points une marque spéciale pour retenir que des modifications doivent être propagées à partir des successeurs.

.../...

Lorsque la première itération de l'algorithme ou itération principale est terminée, on réalise de nouvelles itérations "secondaires" dont les origines sont les fermetures de doublet et de circuit précédemment détectées et marquées. On modifie alors la marque pour indiquer que les modifications ont été propagées au delà des successeurs et que ces points ne doivent plus être origines de cheminement dans les autres itérations.

Nous utiliserons en fait deux tableaux, l'un d'entiers, l'autre de booléens :

- l'entier t (i) indique que le point i a été "examiné" t (i) fois au cours de l'itération considérée ;
- le booléen m (i) a la valeur "vrai" si le point a été origine de cheminement lors d'une itération "secondaire" : nous dirons dans ce cas que le point est "marqué".

Les points qui seront origines de cheminement lors des itérations ultérieures sont stockés dans une "file d'attente" ou "file d'origines".

Pour permettre des comparaisons rapides, nous présentons les algorithmes adaptés à notre problème, sur un exemple sans circuit, comportant un doublet, extrait de l'exemple d'appui.

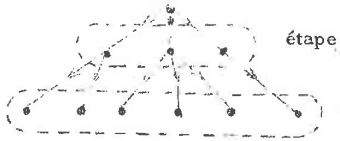
Algorithme de la file

Il procède par récurrence :

A chaque étape on examine les points successeurs des points examinés à l'étape précédente.

Après un point on examine tous ses successeurs

aussi pourrait-on l'appeler "cheminement en éventail" ou "par niveaux"



Rappelons que le marquage permet de traiter le problème des et des doublets par des itérations secondaires.

Redonnons, pour mémoire, la définition d'une file :

Une file est une suite ordonnée d'informations qui peut subir modifications suivantes :

- suppression d'un élément en "tête" de file (le plus ancien) :
- adjonction d'un nouvel élément en "queue" de file.

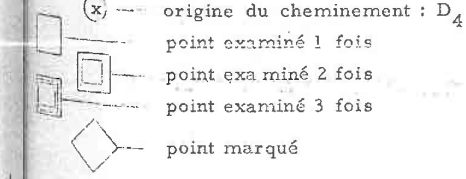
On utilise la "file" de la manière suivante :

.../...

- Le point, dont on examine les différents successeurs est toujours en tête de "file" :
  on ajoute les successeurs non encore examinés au cours de l'itération en queue de la "file" et les successeurs déjà examinés une fois et non marqués en queue de la "file des origines" ;
- Si la tête de la "file" n'a plus de successeurs à considérer, on supprime cet élément de la "file"
- Si la "file" est vide, une itération est terminée ; alors si la file des origines" contient des éléments, sa tête est supprimée et devient la tête de la "file" pour une nouvelle itération, si la file des origines était vide elle aussi, le cheminement est terminé.

En utilisant les figures 6a) et 6b), effectuons le cheminement consécutif à une modification de D_4 (1)

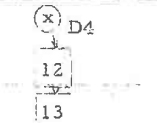
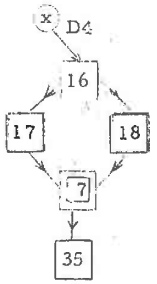
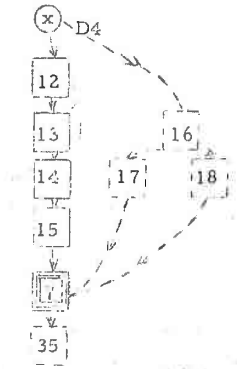
(1) Symboles :



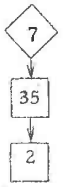
analyse	cheminement	variations des files
		"file"
initialisations		vide
première itération		
étape 0	ⓧ D_4	D_4
étape 1		
successeurs de D_4 : 4	ⓧ D_4 → 4	D4, 4
12	ⓧ D_4 → 12	D4, 4, 12
16	ⓧ D_4 → 16	D4, 4, 12, 16
fin		4, 12, 16
étape 2 :		
successeurs de 4 : néant		12, 16
successeurs de 12 : 13	ⓧ D_4 → 12 → 13	12, 16, 13
fin		16, 13
successeurs de 16 : 17	ⓧ D_4 → 16 → 17	16, 13, 17
18	ⓧ D_4 → 16 → 17, 18	16, 13, 17, 18
fin		13, 17, 18

analyse	cheminement	variations des files	
		" file "	file des origines
J	ⓧ D_4		
sseurs de 13 : 14	ⓧ D_4 → 12 → 13 → 14	13, 17, 18, 14	
fin		17, 18, 14	
sseurs de 17 : 7	ⓧ D_4 → 16 → 17 → 7	17, 18, 14, 7	
fin		18, 14, 7	
sseurs de 18 : 7	ⓧ D_4 → 16 → 17, 18 → 7		7
fin		14, 7	

fermeture de doublet 7 est examinée ici dans un cas favorable : 7 est encore ns la "file", ses successeurs n'ont pas encore été examinés.

analyse	cheminement	variation des files	
		" file "	file des origines
étape 4			
successeurs de 14 : 15		14, 7, 15	
fin		7, 15	
successeurs de 7 : 35		7, 15, 35	
fin		15, 35	
successeurs de 15 : 7		7	
fin		35	

la fermeture de doublet 7 est examinée ici dans un cas défavorable : 7 n'est p
dans la file et ses successeurs sont déjà examinés, ceci est dû au fait que 7
pas au même niveau sur les différents chemins en dérivation.

analyse	cheminement	variation des files	
		" file "	file des origines
successeurs de 35 : 2		35, 2	
fin		2	
étape 6 successeurs de 2 : néant		vide	7
nième itération			
étape 0		7	vide
étape 1 successeurs de 7 35		7, 35	
fin		35	
étape 2 successeurs de 35 : 2		35, 2	
fin		2	
étape 3 successeurs de 2 : néant		vide	vide

Remarques

1 - Ce processus itératif est convergent car le graphe est un graphe fini (n points) :

a) Chaque itération converge puisque tout point n'est mis dans la "file" que s'il est non examiné (il devient alors examiné) et qu'il disparaît de la "file" lorsque l'on a considéré tous ses successeurs : l'itération est terminée lorsque la "file" est vide.

b) Le nombre d'itérations est fini : un point n'entre dans la "file des origines" que s'il est examiné pour la deuxième fois dans une itération et non marqué et il disparaît de la "file des origines" lorsqu'il est pris comme origine d'une itération (il devient alors marqué).

2 - Les noeuds examinés à l'étape n d'une itération sont contenus dans Γ^n (origine de l'itération) ; néanmoins le découpage en étapes qui correspond au passage des chemins de i arcs aux chemins de i + 1 arcs n'intervient pas pour la gestion de la "file" et on peut décrire une phase d'itération de l'algorithme sans distinguer les étapes :

- La "file" étant vide, mettre en tête de "file" un élément ôté à la tête de la "file des origines".
- Tant que la "file" n'est pas vide :
  - Tant que la tête de la "file" a un successeur à considérer :
    - Exécuter les règles de ce successeur ;
    - si on l'examine pour la première fois
      alors le mettre en queue de "file"
      finsi ;
    - si on l'examine pour la seconde fois et qu'il est non marqué
      alors le mettre en queue de "file des origines"
      finsi.
  - fintantque ;
  - enlever la tête de la "file".
- fintantque.

.../...

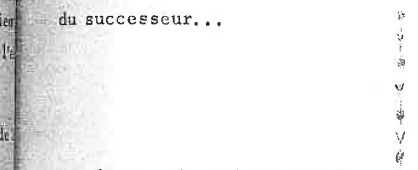
Algorithme de la pile

Il repose sur le principe suivant :

Dès que l'on atteint un point, on étudie ses successeurs avant d'étudier les autres successeurs non encore considérés de son prédécesseur.

En termes imagés : le "fils" d'un point passe toujours avant les "oncles" non examinés avant le "père".

Après un point, on examine un de ses successeurs, puis un successeur du successeur...



aussi pourrait-on l'appeler "cheminement vertical".

Rappelons que le marquage permet de traiter le problème des circuits et des doublets au moyen d'itérations secondaires.

Redonnons, pour mémoire, la définition d'une pile :
Une pile est une suite ordonnée d'informations qui peut subir les modifications suivantes :

- suppression d'un élément au "sommet" de la pile ;
- adjonction d'un nouvel élément au "sommet" de la pile.

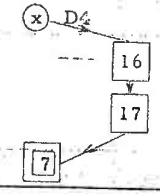
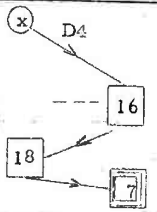
On utilise la pile (et la file des origines) de la manière suivante :

.../...

- Le point, dont on prend un successeur, est toujours le "sommet" d
  ajoute tout successeur non encore examiné au cours de l'itérati
  "sommet" de la pile qui devient alors le nouveau sommet et les suc
  déjà examinés une fois et non marqués en queue de la file des origi
- Si le "sommet" de la pile n'a plus de successeurs à considérer, on
  cet élément de la pile et on retrouve un 'ancien" sommet.
- La file des origines fournit, comme précédemment l'origine de chaq

Reprenons le même exemple en utilisant la même présentation.

analyse	cheminement	variations de la pile	variations de la file des origines
isations		vide	D4
ère itération			
e du cheminement	(x) D4	D_4	vide
sseur de D_4 : 4	(x) D4	D4,4	
sseur de 4 : néant	4	D_4	
	(x) D_4		
sseur de D_4 : 12	12	D4, 12	
sseur de 12 : 13	13	D4, 12, 13	
sseur de 13 : 14	14	D4, 12, 13, 14	
sseur de 14 : 15	15	D4, 12, 13, 14, 15	
sseur de 15 : 7	7	D4, 12, 13, 14, 15, 7	
sseur de 7 : 35	35	D4, 12, 13, 14, 15, 7, 35	
sseur de 35 : 2	2	D4, 12, 13, 14, 15, 7, 35, 2	
sseur de 2 ; néant		D4, 12, 13, 14, 15, 7, 35	
sseur de 35 : fin		D4, 12, 13, 14, 15, 7	
sseur de 7 : fin		D4, 12, 13, 14, 15	
sseur de 15 : fin		D4, 12, 13, 14	
sseur de 14 : fin		D4, 12, 13	
sseur de 13 : fin		D4, 12	
sseur de 12 : fin		D4	

successeur de D4 : 16 successeur de 16 : 17 successeur de 17 : 7		D4, 16 D4, 16, 17	
la fermeture de doublet 7 est examinée dans un cas défavorable : 7 n'est plus d pile et ses successeurs sont déjà examinés, ce cas est le seul possible pour l'a rithme de la pile où les successeurs d'un point considéré sont examinés avant ses			
successeur de 17 : fin successeur de 16 : 18 successeur de 18 : 7		D4, 16 D4, 16, 18	
même remarque que ci-dessus			
successeur de 18 : fin successeur de 16 : fin successeur de D4 : fin		D4, 16 D4 vide	7
deuxième itération ==========			
origine du cheminement		7	vide
successeur de 7 : 35 successeur de 35 : 2 successeur de 2 : néant successeur de 35 : fin successeur de 7 : fin		7, 35 7, 35, 2 7, 35 7 vide	vide

Remarques :

1- Pour les mêmes raisons que précédemment l'algorithme est convergent :

a) Au cours d'une itération, tout point entre au plus une fois dans la pile d'où l'itération converge (points "examinés").

b) Le nombre d'itérations est fini puisque tout point ne peut être plus d'une fois origine de cheminement (points "marqués").

2- Une description modulaire d'une itération est :

- La pile étant vide, mettre en son sommet un élément ôté à la tête de la file des origines.
- Tant que la pile n'est pas vide :
  - Tant que son sommet possède un successeur à considérer :
    - Exécuter les règles de ce successeur ;
    - si on l'examine pour la première fois
      alors le mettre au sommet de la pile
      finsi ;
    - si on l'examine pour la deuxième fois et qu'il est non marqué
      alors le mettre en queue de la file des origines
      finsi
  - fintant que ;
  - enlever le sommet de la pile.
- fintant que

Etude comparative

Si nous superposons les descriptions modulaires d'une itération de ces deux algorithmes dans une présentation simultanée, il apparaît une similitude frappante

- La { file / pile } étant vide,

mettre { en queue de file / au sommet de la pile } un élément ôté à la tête de la "file des origines" ;

.../...

- Tant que la { file / pile } n'est pas vide :

  - Tant que { la tête de la file / le sommet de la pile } possède un successeur à considé

    - Exécuter les règles de ce successeur ;
    - si on l'examine pour la première fois
      alors le mettre { à la queue de la file / au sommet de la pile }
      finsi ;
    - si on l'examine pour la deuxième fois et qu'il est non marqué
      alors le mettre en queue de la "file des origines"
      finsi.

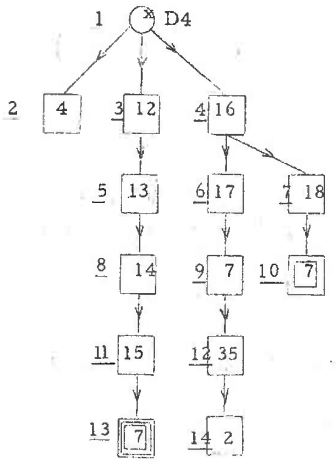
  - fintant que ;
  - enlever { la tête de la file / le sommet de la pile. }

- fintant que

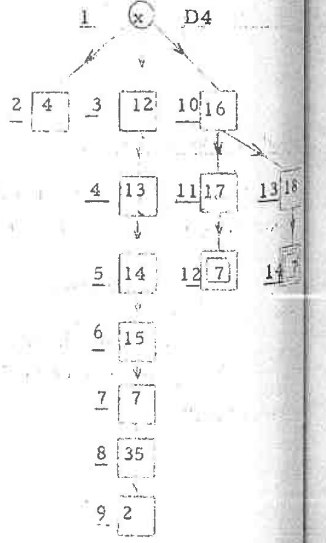
Ainsi, le nombre d'opérations ou de traitements élémentaires est le m pour l'exécution de ces deux algorithmes de cheminement.

Bien que respectant tous deux, en absence de circuits et de doublets, l notion d'ordre, les algorithmes accèdent aux noeuds dans un ordre différ comme le montre une numérotation de la progression sur l'exemple précéd

file ou cheminement "par niveaux"



pile ou "cheminement vertical"



La détection des doublets et des circuits par la méthode du marquage, que nous avons imaginée indépendamment du contexte de l'un ou l'autre des algorithmes est efficace dans les deux cas.

Cependant l'algorithme de la pile présente sur ce point deux avantages : il permet

- une différenciation des doublets et des circuits
- la connaissance de l'image des circuits.

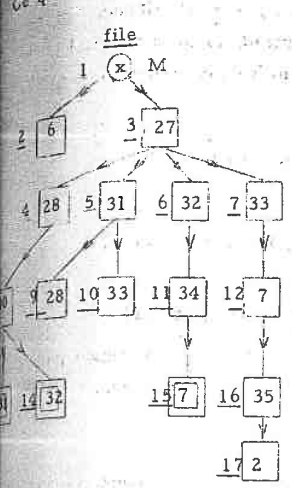
Prenons, pour mettre ces avantages en évidence, un exemple de cheminement, rencontrant à la fois des doublets et un circuit, en modifiant M (figure 6a et 6 b de l'annexe 1) .

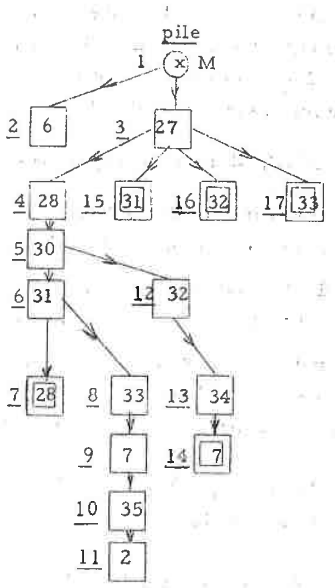
Les états successifs de la file et de la pile au cours de la première itération sont alors : (1)

file	n°	pile
M	1	M
M,6	2	M,6
M,6,27	3	M
6,27	4	M,27
27	5	M,27, 28
27,28	6	M,27, 28, 30
27,28,31	7	M,27, 28, 30,31
27,28,31,32	8	M,27, 28, 30,31(,28)!
27,28,31,32,33	9	M,27, 28, 30,31, 33
28,31,32,33	10	M,27, 28, 30,31, 33, 7
28,31,32,33,30	11	M,27, 28, 30,31, 33, 7, 35
31,32,33,30	12	M,27, 28, 30,31, 33, 7, 35,2
31,32,33,30(,28)!	13	M,27, 28, 30,31, 33, 7, 35
31,32,33,30(,33)!	14	M,27, 28, 30,31, 33, 7
32,33,30	15	M,27, 28, 30,31, 33
32,33,30, 34	16	M,27, 28, 30,31
33,30, 34	17	M,27, 28, 30
33,30, 34, 7	18	M,27, 28, 30,32
30, 34, 7	19	M,27, 28, 30,32, 34
30, 34, 7 (,31)!	20	M,27, 28, 30,32, 34'(,7)!
30, 34, 7 (,32)!	21	M,27, 28, 30,32
34, 7	22	M,27, 28, 30
34, 7 (,7)!	23	M,27, 28
7	24	M,27
7, 35	25	M,27(,31)!
35	26	M,27(,32)!
35, 2	27	M,27(,33)!
2	28	M
nil	29	nil

(1) Nous indiquons les fermetures de doublet ou de circuit qui ne peuvent pas avoir de seconde entrée dans la file ou la pile par les symboles (,n°)!

Ce qui correspond aux cheminements :





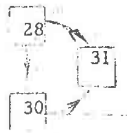
Considérons, dans le cas de la pile, les états 7 et 8 :

P_7 = (M, 27, 28, 30, 31)

et à l'état 8 on voudrait faire entrer 28 alors qu'il est encore sur la pile : on en déduit immédiatement que c'est une fermeture de circuit.

En outre la pile contenant toujours l'image d'un chemin d'origine M, il est possible d'en extraire la composition du circuit détecté soit :

P_7 = (M, 27, 28, 30, 31) d'où



De même il est facile de reconnaître les doublets aux états 20, 25, 26 et 27, les points respectifs 7, 31, 32 et 33 n'étant plus sur la pile au moment où on les examine pour la deuxième fois sont des fermetures de doublet (il ne reste sur la pile qu'un des chemins en dérivation, le dernier parcouru).

.../...

Au contraire, les états de la file ne permettent pas de distinguer les fer
de circuit des fermetures de doublet et, ne contenant généralement pas l'ima
chemins mais seulement une fraction correspondant à deux niveaux, ils ne fo
sent pas l'image des circuits.

Ainsi, si nous considérons les états 14 et 20 de la file :

33, fermeture de doublet, est dans la file lorsqu'on le
rencontre pour la deuxième fois : F14 = (31, 32, 33, 30) (idem pour 7)

et 31, fermeture de doublet n'est plus dans la file
lorsqu'on le rencontre à nouveau : F20 = (30, 34, 7) (idem pour 32)

si nous ne le constations pas sur la figure 6a) ou si nous ne l'avions pas
découvert par l'algorithme de la pile, rien ne permet d'affirmer que ce sont
des fermetures de doublet et non de circuit.

De même, l'état de la file F13 =(31, 32, 33, 30) ne permet aucune conclusi
sur le noeud 28.

D'autres études (Derniame-Pair, 1971) menées sur l'évaluation des perf
ces des algorithmes à piles en comparaison d'algorithmes dérivés, pour déte
les circuits et optimisés par des tests, d'algorithmes classiques (Warshall,
Dantzig...) ont montré l'intérêt des algorithmes à piles. L'algorithme de la
demande moins d'opérations élémentaires (l'opération élémentaire est l'unité
mesures faites) dans la majorité des exemples traités, en particulier lorsque
graphe a "peu" [1] de fermetures de circuit ce qui est le cas du graphe des
noeuds d'influence.

Ces mesures, ajoutées aux avantages déjà cités de l'algorithme de la pil
l'algorithme de la file (équivalent par contre en nombre d'opérations éléments
expliquent notre choix de l'algorithme de la pile.

---

(1) l'algorithme de la pile est le plus performant :
- toujours quelque soit l'ordre du graphe en absence de circuits
- jusqu'à 15 fermetures de circuits pour un graphe d'ordre 20
- jusqu'à 25 fermetures de circuits pour un graphe d'ordre 30

...

Algorithme "brut" de la pile

Derniame-Pair (page 122) proposent de l'algorithme de la pile sous sa forme "brute" une description par une procédure Algol 60 qui donne à la fois l'analyse du procédé et sa méthode de résolution sur ordinateur.

Cette procédure utilise :

- une matrice carrée a(n, n), c'est la matrice associée au graphe d'ordre n (a(i, j) = vrai s'il existe un arc de i à j, fausse sinon),
- une pile p rangée dans un tableau,
- un tableau de booléen t qui permet de noter l'existence d'un chemin d'un point donné x à tout point i (t(i) = vrai s'il existe un chemin de x à i, faux sinon)

```
procédure pilexist (a, n, x, t) ; valeur n, x ; booléen tableau a, t ;
          entier n, x ;
début     entier k, i, j, r ; entier tableau p [1:n+1] ; booléen s ;
          pour i := 1 pas 1 jusqua n faire t [i] := faux ;
          k := 1 ; p [1] := x ; t (x) := vrai ; i := x ;
test :    pour j := 1 pas 1 jusqua n faire
          si a [i, j] alors allera entr ;
sor :     si k = 1 alors allera exit ;
          k := k-1 ; r := i + 1 ; i := p [k] ;
          pour j := r pas 1 jusqua n faire
          si a [i, j] alors allera entr ;
          allera sor ;
entr :    k := k + 1 ; p [k] := i := j ; s := t (i) ; t [j] := vrai ;
          commentaire : j entre dans la pile et on note qu'il existe un chemin de
          x à j ;
          allera si s alors sor sinon test ;
exit :    fin ;
```

.../...

description de l'algorithme utilisé

Nous préférons, pour notre part, exposer l'analyse du processus
damment de ses contraintes de programmation dans un langage "modul
du même type que celui que nous avons déjà utilisé aux pages 98, 103

On retrouvera en particulier dans cette description celle d'une itér
du processus donnée page 103.

Il nous faut au préalable préciser certaines notations. Par rapport
procédure Algol, nous remplacerons le tableau de booléens t par un
tableau d'entiers t qui permet de compter, au cours d'une itération,
nombre d'examen de chaque point et de s'assurer que chaque point n'a
entrée sur la pile p.

Nous remplacerons l'entier r (il permettait d'éliminer les succes
déjà considérés du sommet de la pile) par une fonction entière nouv-suc
telle que, initialisée au début de chaque itération, au $j^{ième}$ appel :

$$\text{nouv-succ}(i) = \begin{cases} \text{n° du } j^{ième} \text{ successeurs de } i \text{ s'il existe ;} \\ -1 \text{ si } i \text{ a moins de } j \text{ successeurs.} \end{cases}$$

Nous ajouterons, pour réaliser les itérations nécessitées par la te
du marquage, une file f, qui au départ contiendra le numéro no d
de définition associé au mot modifié origine des modifications en cascad
un tableau de booléens m qui indique si les points sont marqués ou no

Nous éditerons les circuits, au cours des itérations secondaires,
la fermeture de circuit est le point marqué situé à la base de la pile (cr
du cheminement), ce qui évite une exploration de la pile pour trouver le
du circuit (cf page 107)

Enfin, en remarquant qu'il n'est intéressant d'examiner les succes
d'un point que s'il y a effectivement des modifications de tailles à prop
il faut conditionner le cheminement par le résultat de l'exécution des r
c'est le rôle donné au booléen mod prenant la valeur vrai si au moins
résultat a eu sa taille modifiée par l'exécution de la règle et la valeur
dans le cas contraire.

Compte tenu de ces notations, l'algorithme peut être ainsi décrit :

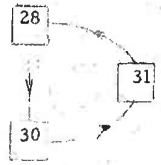
```
prendre une file f vide ; mettre no en queue de f ;
mettre tous les éléments de m à faux ;
prendre une pile p vide ;
tant que la file f n'est pas vide :
- enlever la tête x de f et la mettre au sommet de p ;
- donner à tous les éléments de t la valeur zéro ; t(x)=1;m(x):= vrai ;
- initialiser la fonction nouv-succ ;
- tant que la pile p n'est pas vide :
   - tant que j = nouv-succ (sommet (p)) > 0 :
        - exécution des règles j ;
        - si mod = vrai
          alors
                si t (j) = 0
                alors mettre j au sommet de p
                sinon
                     si t (j) = 1 et m (j) = faux
                     alors mettre j en queue de f
                     sinon
une                      si j = base de p
itération :              alors imprimer le circuit j...j
cheminement                    ( = contenu de p)
l'origine x              finsi
                     finsi
                finsi
          t(j) : = t(j) + 1
          finsi
   - fintant que ;
   - enlever le sommet de p.
- fintant que ;
- tant que m (tête de f) = vrai :
   - enlever la tête de f
- fin tant que.
fin tant que.
```

On trouvera, dans l'annexe 2 (figure 1) les itérations secondaires, l'algorithme de la pile, de l'exemple avec circuit de la page 106 et dans la troisième partie, les résultats obtenus sur ordinateur.

Remarques sur ces itérations secondaires :

- le tableau m pourrait être utilisé autrement : m (j) = vrai si j a déjà mis dans la file des origines et faux dans le cas contraire, cela ne change aux résultats de l'algorithme.

- pour le circuit



28 et 31 sont deux fermetures de circuit au delà desquelles il faut propager les modifications dues au circuit ;
30 par contre appartient au circuit, mais n'a pas de successeurs en dehors de lui, il ne joue aucun rôle dans les itérations secondaires.

- les extrémités des chemins : (7, 35, 2) en particulier, sont parcourues plusieurs fois, chaque fois que l'on propage de nouvelles modifications à partir des fermetures de circuits.

- le fait que le cheminement ne se poursuive au delà d'un point que s'il y a effectivement des modifications à propager diminue le nombre de points examinés, d'où en conséquence les itérations secondaires (qui partent de points examinés deux fois).

(2.1.2.1.1) **Définitions**

Les objets qui font la base de cette partie sont les mêmes que ceux du paragraphe 2.1.1. Nous considérons donc un complexe d'influences normalisé (en abrégé C.I.N.) défini par le couple $(\mathcal{M}, \mathcal{R})$ où la caractéristique associée est la taille des mots.

Le problème posé est le suivant :

L'ensemble des mots étant partitionné en deux sous ensembles $\mathcal{J}$ et $\mathcal{M} - \mathcal{J}$ où chaque élément a dans le premier une taille indéterminée ou inconnue et dans le second au contraire une taille déterminée ou connue, il faut calculer les tailles des mots de $\mathcal{J}$ à partir de celles des mots de $\mathcal{M} - \mathcal{J}$ en appliquant les règles issues de $\mathcal{R}$.

Nous avons pressenti dans la 1è partie que ce problème découle des mêmes principes que le problème des modifications en cascade : la construction d'un graphe et un cheminement dans ce graphe en appliquant les règles de calcul des tailles.

L'analyse rapide que nous avions ébauchée utilisait le graphe inverse du graphe des influences.

Les mêmes arguments que ceux développés au paragraphe 2.1.1.1. conduisent à abandonner ce graphe dont la représentation est complexe (liste d'informations sur chaque arc) au profit d'un graphe plus simple pour l'automatisation des cheminements et que nous avons appelé pseudo-inverse du graphe des noeuds d'influence ou graphe P.I.G.N.I.

Nous pouvons, comme nous l'avons fait au paragraphe 2.1.1.1. pour le graphe des noeuds d'influence introduire ce graphe de trois façons :

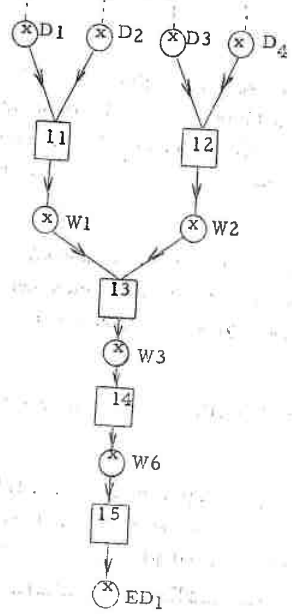
- par transformations successives à partir de l'inverse du graphe bialterné,
- par sa définition ;
- comme extension du graphe représentatif des arcs de l'hypergraphe inverse d'un hypergraphe orienté.

.../...

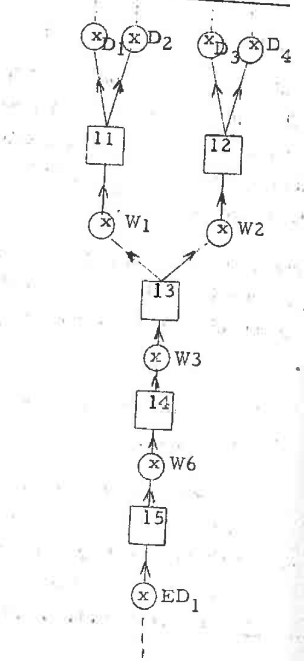
Reprenons rapidement chacun de ces points :

Pour développer le premier, prenons un exemple extrait de la figure de l'annexe 1 :

graphe bialterné



inverse du graphe bialterné

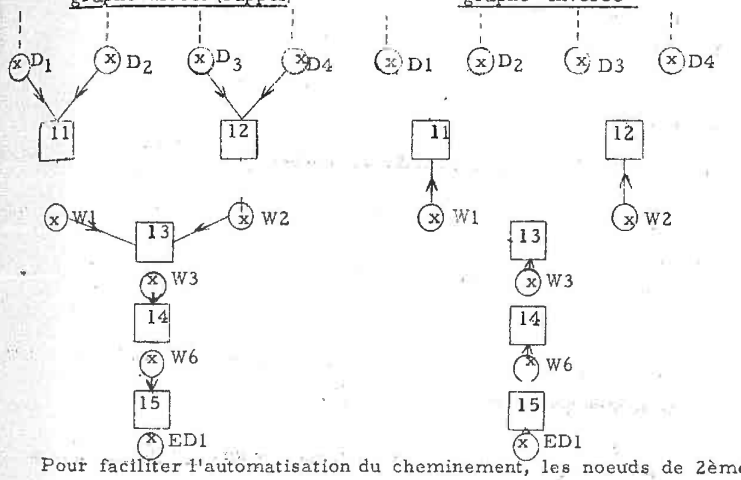


Les noeuds de deuxième espèce sont des références à des triplets (Opérandes, Règles, Résultats), ils fournissent eux-mêmes la liste de leurs successeurs (les opérandes) et les arcs joignant les noeuds de 2ème espèce aux noeuds de première espèce constituent une information redondante qui peut être supprimée de la "représentation informatique" du graphe :

" représentation informatique "

graphe direct (rappel)

graphe "inverse"

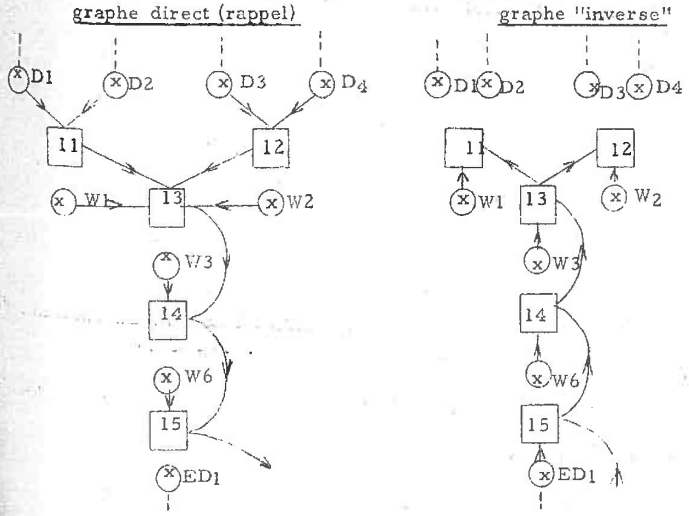


Pour faciliter l'automatisation du cheminement, les noeuds de 2ème espèce qui se "succèdent" dans le graphe inverse du graphe bialterné sont reliés entre eux :

représentation informatique

graphe direct (rappel)

graphe "inverse"



Il est facile de montrer, comme nous l'avons fait au paragraphe 2, que le graphe ainsi obtenu est isomorphe au graphe que nous allons maintenant définir par une définition mathématique, puisqu'ils ont même représentation graphique (Steen, 1968).

Introduisons, cette fois, des pseudo-noeuds de définition en correspondance biunivoque avec les mots de $\mathcal{M}$:

Définitions

Nous appelons pseudo-noeud de définition le triplet (opérandes, règles, résultats) où
l'ensemble des opérandes est réduit à un mot de $\mathcal{M}$, l'ensemble des règles est vide,
les règles sont des règles sans effet appelées "règles de définition".

Soit $\mathcal{P}$ l'ensemble des pseudo-noeuds de définition ; $\mathcal{P}$ est en correspondance biunivoque avec $\mathcal{M}$ ou avec $\mathcal{D}$.

Soit $\tilde{\mathcal{N}} = \mathcal{R} \cup \mathcal{P}$; nous appelerons encore noeuds d'influence les éléments de $\tilde{\mathcal{N}}$.

Soit $\tilde{\Gamma}$ la relation binaire définie sur $\tilde{\mathcal{N}}$ par $\forall n,\ m \in \tilde{\mathcal{N}}$

$$n\ \tilde{\Gamma}\ m \Longleftrightarrow \left\{ \begin{array}{l} \exists\ x \in \mathcal{M} \text{ tel que} \\ x \text{ est opérande de } n \\ x \text{ est résultat de } m \end{array} \right\}$$

Il est évident que sur $\mathcal{R}$: $\tilde{\Gamma} = \Gamma^{-1}$

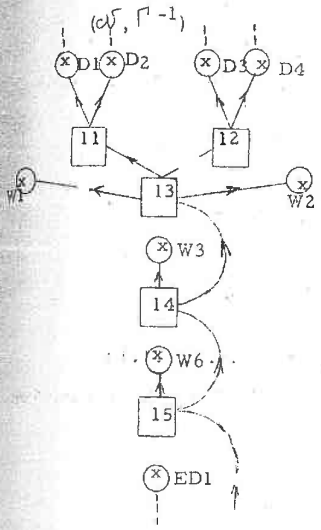
Le graphe $(\tilde{\mathcal{N}}, \tilde{\Gamma})$, dont les points sont les éléments de $\tilde{\mathcal{N}}$ et les arcs définis par la relation $\tilde{\Gamma}$, est le pseudo-inverse du graphe des noeuds d'influence (en abrégé graphe pseudo-inverse ou P.I.G.N.I.)

Le sous-graphe $(\mathcal{R}, \tilde{\Gamma})$ est le graphe inverse du sous-graphe $(\mathcal{R}, \Gamma)$ du graphe $(\mathcal{N}, \Gamma)$.

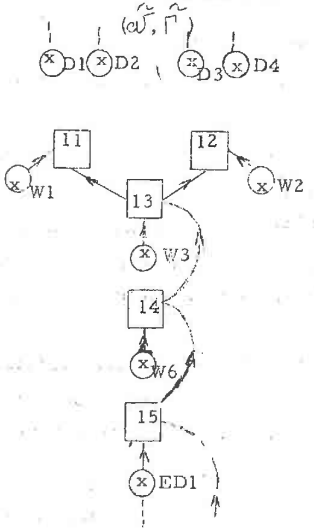
Cependant, $(\tilde{\mathcal{N}}, \tilde{\Gamma})$ n'est pas isomorphe à $(\mathcal{N}, \Gamma^{-1})$ graphe inverse de $(\mathcal{N}, \Gamma)$:

On peut le constater rapidement sur un exemple en traçant la représentation graphique du graphe $(\mathcal{N}, \Gamma^{-1})$ à partir de celle du graphe direct de la page 116 par inversion du sens des arcs :

inverse du graphe direct



graphe "inverse"



Ces deux représentations diffèrent uniquement par les arcs adjacents aux noeuds (respectivement pseudo-noeuds) de définition.

.../...

La définition de ces noeuds, différente dans chaque cas, en est la ca

Dans le graphe direct :

A tout mot x est associé un noeud de définition dx composé du triplet ($\emptyset$, def., x) ; aussi comme

$$n \Gamma m \Leftrightarrow \left\{ \begin{array}{l} \exists y \in \mathcal{M} \text{ tel que} \\ y \text{ est résultat de } n \\ y \text{ est opérande de } m \end{array} \right\} ;$$

+ il n'existe aucun noeud n tel que $n \Gamma d_x$ puisque l'ensemble des opér de dx est vide (c'est-à-dire qu'aucun arc n'a pour extrémité d_x)

+ les noeuds n tels que $dx \Gamma n$ ont x comme opérande puisque x est seul résultat de dx (c'est-à-dire qu'un arc joint dx à chaque noeud où x opérande)

Il en résulte que dans le graphe inverse du graphe direct :

+ il n'y a aucun arc ayant pour origine dx
+ un arc joint tout noeud où x est opérande à dx.

Par contre dans le graphe pseudo-inverse (ou P.I.G.N.I.) :

A tout mot x est associé un pseudo-noeud de définition dx composé du triplet (x, def., $\emptyset$) ; aussi comme

$$n \tilde{\Gamma} m \Leftrightarrow \left\{ \begin{array}{l} \exists y \in \mathcal{M} \text{ tel que} \\ y \text{ est opérande de } n \\ y \text{ est résultat de } m \end{array} \right\} ;$$

+ les noeuds n tels que $dx \tilde{\Gamma} n$ ont x comme résultat puisque x est opérande de dx (c'est-à-dire qu'un arc joint dx à chaque noeud où x est résultat)

+ il n'existe aucun noeud n tel que $n \tilde{\Gamma} dx$ puisque l'ensemble des résul dx est vide (c'est-à-dire qu'aucun arc n'a pour extrémité dx).

Ce qui peut être résumé par le tableau suivant : $\forall x$ de $\mathcal{M}$:

arcs	inverse	pseudo-inverse
origine en dx		extrémité aux noeuds où x est résultat
extrémité en dx	origine aux noeuds où x est opérande	

Ces différences justifient le terme de "pseudo-inverse" que nous avons employé. Elles sont une raison supplémentaire à la construction directe du graphe pseudo-inverse associé à un C.I.N. à partir de sa définition ($\tilde{\mathcal{U}}$, $\tilde{\Gamma}$) de même que nous avions convenu de construire le graphe des noeuds d'influence à partir de sa définition (N, Γ).

Il est certain d'autre part que l'on peut aussi limiter la construction du graphe à celle d'un sous-graphe en réduisant l'ensemble $\mathcal{P}$ des pseudo-noeuds de définition au sous-ensemble $\mathcal{P}'$ des pseudo-noeuds qui seront origines de cheminement, c'est-à-dire associés à l'ensemble $\mathcal{J}$ des mots dont la taille est indéterminée.

Il est enfin immédiat de montrer que le graphe ($\mathcal{B}$, $\tilde{\Gamma}$) est le graphe représentatif des arcs de l'hypergraphe "inversé" de l'hypergraphe orienté présenté au paragraphe 2.1.1.1. :

Si l'on définit l'hypergraphe inverse d'un hypergraphe orienté comme étant l'hypergraphe obtenu en inversant le sens des arcs : à tout arc ($\mathcal{O}_i$, $\mathcal{C}_i$) on associe l'arc ($\mathcal{C}_i$, $\mathcal{O}_i$), on déduit aisément que le graphe représentatif des arcs de l'inverse d'un hypergraphe orienté est l'inverse du graphe représentatif des arcs de cet hypergraphe.

Les problèmes de calcul et de maintenance des tailles dans un contexte donné peuvent être formalisés grâce à la notion de complexe d'influences dont une représentation sous forme d'hypergraphe orienté permet de ramener la recherche d'une solution à celle d'algorithmes de cheminement sur son graphe représentatif.

(2.1.2.1.2) **Caractéristiques**

Le graphe $(\tilde{\mathcal{A}}, \tilde{\Gamma})$ contient le sous-graphe $(\mathcal{B}, \tilde{\Gamma})$, est inverse du sous-graphe $(\mathcal{B}, \Gamma)$; aussi, comme lui, comporte-t-il des boucles, des circuits et des doublets.

(2.1.2.2.1) Récursivité du processus

Au paragraphe 2.1.1.2., nous avons été amenés à utiliser l'algorithme de la pile pour la recherche de chemins dans le graphe des noeuds d'influence.

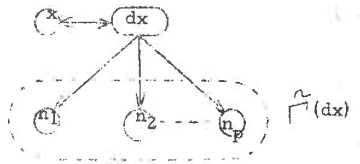
Nous pouvons justifier a posteriori ce choix, car les recherches de cheminement considérées s'introduisent naturellement comme des processus récursifs : considérons sous cet angle l'évaluation des tailles inconnues :

Soit x un mot de taille inconnue. Nous noterons dx son pseudo-noeud de définition.

Si nous supposons que nous disposons d'une procédure évalue-taille (n) qui calcule les tailles inconnues des opérandes du noeud n, il est évident que évalue - taille (dx) calcule la taille de x puisque x est le seul opérande de dx.

Quel peut être le contenu de cette procédure ?

dx est un point de $(\tilde{\alpha}, \tilde{\Gamma})$ qui a pour successeurs tous les noeuds où x est résultat :



$\tilde{\Gamma}(dx)$ est l'ensemble des noeuds d'influence qu'il faut examiner pour évaluer la taille de x :

En effet, chaque noeud n_i de $\tilde{\Gamma}(dx)$ fournit une évaluation de la taille de x, qui peut être différente ou non de la taille précédente. Nous avions convenu, au cours de la première partie, de désigner ces tailles intermédiaires par le terme "taille temporaire".

La taille définitive ou évaluée de x est la taille temporaire obtenue après exécution de toutes les règles associées aux relations d'influence où x est résultat.

.../...

En absence de circuits ou de boucles dans $(\hat{\omega}, \tilde{\Gamma})$, la taille évalu
est indépendante de l'ordre dans lequel on considère ces relations d'influ

Si pour tout noeud n_i de $\tilde{\Gamma}$ (dx), la taille des opérandes est co
déjà évaluée, il suffit d'appliquer successivement les règles associées au
différents noeuds n_i et la taille temporaire finale de x constitue sa tai
évaluée.

Si, au contraire, pour au moins un noeud n_i de $\tilde{\Gamma}$(dx), il existe
opérandes de taille inconnue il faut, avant celle de x, évaluer leurs taille
pour cela utiliser la procédure évalue -taille (n_i), ce qui constitue un pro
récursif dont on peut donner une définition sous la forme :

évalue - taille (n) :
- pour chaque ni de $\tilde{\Gamma}$(n)
  - si au moins un opérande de n_i est de taille inconnue
  - alors évalue - taille (n_i)
  - finsi ;
  - exécution des règles associées à n_i ;
- fin pour chaque ;
- les tailles temporaires des opérandes de n deviennent des taille
  évaluées .

Une utilisation classique des piles est la traduction des processus
récursifs, il est donc normal, pour une programmation en un langage n
tant pas les procédures récursives, de développer des algorithmes utilis
pile.

La solution que nous avions choisie au paragraphe 2.1.1.2. de tr
le problème des cycles non pas par une transformation du graphe mais p
spécification des règles de calcul inhérente aux modifications en cascad
oblige ici à reprendre ces points.

Décrivons auparavant, de la même façon, le problème de la maintenance des tailles à l'aide d'une procédure récursive : modifie - taille (n) qui effectue les modifications consécutives à celles subies par les résultats du noeud n et dont le fonctionnement est :

modifie - taille (n) :
- pour chaque n_i de Γ (n)
  - exécution des règles associées à n_i ;
  - modifie - taille (n_i).

On notera les positions différentes de l'exécution des règles associées à n_i :

- après évalue - taille (n_i) pour que les opérandes de n_i aient des tailles évaluées ;
- avant modifie - taille (n_i) pour que les résultats de n_i soient modifiés.

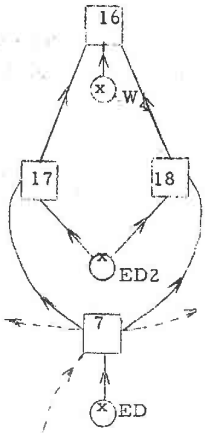
(2.1.2.2.2) traitements des cycles

Les doublets

Le problème se simplifie ici comme nous allons le vérifier sur un ex
en appliquant le processus énoncé précédemment :
Supposons que pour l'exemple d'appui de l'annexe 1, l'ensemble $\mathcal{M} - \mathcal{Y}$ des
dont la taille est connue soit :

$\mathcal{M} - \mathcal{Y} = \{ D_1, D_2, D_3, D_4, D_5, A, B, C, D, M, N \}$ et cherchons à évaluer la ta
de ED2.

L'environnement de ED2 dans le graphe P.I.G.N.I. est :



ED2 est résultat dans 17 , et 17 comprend un opérande W_4 de taille inconnue ;
En respectant le principe de la récursivité :

on cherche d'abord la taille des opérandes de 17 qui sont résultats de la
tion d'influence 16 puisque 17 $\tilde{\Gamma}$ 16 . Les opérandes de 16 (D_2
ayant des tailles connues, les règles associées à 16 permettent d'affecter
nouvelle taille temporaire (différente de l'état nul) au résultat W_4. Or
étant le seul successeur de 17 , la taille temporaire de W_4 devient
évaluée. On peut alors appliquer les règles associées à 17 ce qui permet
calculer la taille temporaire de ED2.

.../...

ED2 est aussi résultat dans 18 lequel a pour opérande W_4 qui est résultat dans 16 , ce qui conduit, en principe, à détecter une fermeture de doublet dans le graphe.

Mais l'étude du premier chemin en dérivation a conduit à évaluer la taille de tous les opérandes des noeuds de ce chemin (excepté l'origine), en particulier celle de W4 opérande commune à 17 et à 18 (et de ce fait cause de la fermeture du doublet). Or le cheminement n'étudie les successeurs d'un point que si celui-ci possède des opérandes de tailles inconnues (les tailles devenant connues à la suite de cette étude) ;
ainsi le (ou les) opérande (s) commune (s) (W4) aux points prédesseurs 17 et 18 sur chaque chemin dérivé, de la fermeture de doublet (16) ayant toujours sa (leur) taille évaluée lors de l'étude du premier chemin en dérivation, le cheminement n'examinera pas les points où il (s) (W4) est (sont) résultat (s) (16) lors du parcours des autres chemins dérivés.

Toutefois si 18 avait d'autres opérandes et qu'ils soient également de tailles inconnues, le cheminement se poursuivrait naturellement vers les successeurs de 18 (dont 16 mais cela est sans importance puisque les tailles sont connues).

Si le cheminement est réalisé par la procédure récursive définie précédemment : les doublets ne nécessitent plus une procédure spéciale pour les détecter lors du cheminement, bien plus, le processus de marquage déjà utilisé, n'intervenant que pour les points "examinés" deux fois au moins, ils n'interviennent absolument plus dans l'algorithme que nous développons.

Les boucles et les circuits

Le problème des boucles et des circuits se pose dans les mêmes termes que pour la maintenance des tailles.

La solution que nous avions proposé dans ce cas, s'appuyant sur définition existante des tailles ne peut évidemment pas être transposée à extenso. Devant l'inexistence d'une solution réellement satisfaisante, nous avons cependant conservé le principe d'un seul passage par boucle ou ci en faisant appel à l'utilisateur et en lui fournissant une documentation lui permettant de prendre la "meilleure" décision.

Un tel choix implique la possibilité de "conversation" entre l'hom et l'outil, ce que nous avons prévu dans notre réalisation par la possib d'insertion d'une procédure conversationnelle mais non exécuté pour de simples raisons techniques.

Pour pallier l'absence d'une telle procédure, nous avons prévu la possibilité, pour l'utilisateur, d'intervenir en choisissant lui-même les tailles des mots provoquant les boucles et les fermetures de circuit (le cheminement s'arrêtant sur les tailles connues, on ne parcourt plus les circuits). Il pourra réaliser plusieurs essais avec des tailles différentes essais ne demandent que l'exécution répétée du seul "cheminour" qui fo les boucles et les fermetures de circuit rencontrées lors des cheminem

(2.1.2.2.3) Description de l'algorithme

Par rapport à la procédure Algol 60 de l'algorithme de la pile (Derniame-Pair, 1971) rappelée page 109, la description que nous donnons fait apparaître les différences suivantes :

- Le cheminement est conditionné par la présence ou non d'opérandes de tailles inconnues : nous noterons inconnue (n) la fonction booléenne qui a la valeur "vrai" si au moins un opérande de n est de taille inconnue et "faux" dans le cas contraire.

- Il est accompagné d'exécution de règles de calcul des tailles.

- L'algorithme est répété tant qu'il y a des mots de tailles inconnues : nous noterons nouv - inconnue la fonction entière qui reçoit la référence positive à un pseudo-noeud de définition associée à un mot de taille inconnue s'il en existe et qui prend la valeur - 1 sinon.

- Il fait intervenir, comme pour les modifications en cascade, le tableau d'entiers t et la fonction nouv - succ (n).

```
prendre une pile p vide ;
mettre t à la valeur zéro ;
initialiser la fonction nouv - succ ;
tant que x = nouv-inconnue> 0 ;
        mettre x au sommet de p ; t(x) = 1 ;
        tant que la pile p n'est pas vide ;
            tant que j = nouv-succ(sommet (p)) > 0 ;
                si inconnue (j) = "vrai"
                alors
                    si t (j) ≠ 0
                    alors édition boucle ou circuit
                    sinon (t(j) = 0)
                            mettre j au sommet de p ; t(j) = 1
                    finsi
                sinon (inconnue (j) = "faux") exécution des règles j
                finsi
            fintant que ;
            la taille temporaire des opérandes de sommet (p) devient l
            taille évaluée ;
            exécution des règles associées à sommet de p ;
            enlever le sommet de p
        fintant que.
fintant que.
```

On trouvera dans l'annexe 2 (figure 2) un exemple de cheminement avec

Chapitre 2

Règle de calcul des tailles

(2.2)

deuxième partie

chapitre 2

Section 1 :

Typologie

(2.2.1)

La théorie développée au chapitre précédent supposait d'une part que l'on disposait des règles de calcul de tailles, d'autre part que l'on travaillait au niveau d'un complexe d'influences normalisé.

Ce chapitre a pour objet de montrer comment les règles peuvent être élaborées, le suivant étudie la transformation par l'analyseur d'un contexte donné en G.I.N. associé.

Dans tout langage d'analyse ou de programmation traitant des données simples ou complexes on trouve principalement quatre grandes familles de groupes opératoires créant des "influences" entre les tailles des données :

- influences "hiérarchiques" ou de "concaténation"
- influences de "redéfinition" ou "synonymes"
- influences "arithmétiques"
- influences de "transfert" ou d' "affectation simple"

Ainsi le langage Cobol sur lequel nous travaillons comporte :

- Les influences hiérarchiques (dont les règles sont référencées par le code H) qui proviennent des déclarations de structures. Elles autorisent la définition d'une zone résultat par concaténation ou juxtaposition des zones opérandes.
- Les influences de redéfinition (code des règles : R) qui définissent des synonymes : un ou plusieurs opérandes et le résultat occupent le même emplacement mémoire.
- Les influences arithmétiques : additives (code +), soustractives (code -), multiplicatives (code *), de division (code /) ou d'exponentiation (code E), correspondent à une opération élémentaire liant opérandes et résultats.
- Les influences de transfert (code M) dues au déplacement du contenu de une ou plusieurs zone (s) dans une ou plusieurs autre (s).

Cette typologie permet de limiter la recherche des règles à celles de chaque type de groupes opératoires.

Section 2 :
Biais systématique

(2. 2. 2)

Considérons un type de groupes opératoires, par exemple une influence de division, et cherchons s'il existe un procédé pour définir la règle qui calcule la taille des résultats en fonction de celles des opérandes :

Soit la relation d'influence

$$\left(\left\{ A, B \right\} , \mathcal{R}_T \, (C := A/B) , \left\{ C \right\} \right)$$

où A et B ont pour tailles (6, 5, 1) et (5, 3, 2).

Pour calculer la taille de C on peut imaginer de chercher les valeurs extrêmes positives que peut prendre C :

$$\text{valeur} \left\{ \begin{array}{l} \text{maximale} \\ \text{minimale non nulle} \end{array} \right\} \quad \text{pour A} = \left\{ \begin{array}{l} 99999,9 \\ 00000,1 \end{array} \right\}$$

$$\text{valeur} \left\{ \begin{array}{l} \text{maximale} \\ \text{minimale non nulle} \end{array} \right\} \quad \text{pour B} = \left\{ \begin{array}{l} 999,99 \\ 000,01 \end{array} \right\}$$

d'où pour C = A/B :

$$\text{valeur} \left\{ \begin{array}{l} \text{maximale} \\ \text{minimale non nulle} \end{array} \right\} \quad \text{pour C} = \left\{ \begin{array}{l} \dfrac{99999,9}{000,01} = 9999990 \\ \dfrac{00000,1}{999,99} > \dfrac{1}{10000} = 0,0001 \end{array} \right.$$

Ceci permet de donner à C la taille (11, 7, 4), ce qu'exprime la règle

la taille entière de C est la somme de celle du dividende A et de la taille décimale du diviseur B ; la taille décimale de C est la somme de celle du dividende A et de la taille entière du diviseur B .

.../...

Remarquons que la taille de C ainsi choisie n'exclue pas les arrondis :

exemple : A=1 et B = 3 alors C = 0, 3333...

ceci est naturel puisque l'ensemble des nombres décimaux est strictement contenu dans l'ensemble des nombres rationnels : on ne peut donc pas (de même pour les calculs scientifiques traditionnellement effectués en binaire et qui ne peuvent manipuler qu'un autre sous-ensemble des rationnels) supprimer toute possibilité d'erreur de chute.

Toutefois, la taille trouvée pour C est satisfaisante puisqu'elle évite les "débordements par le haut" (perte de chiffres de fort poids) et les "débordements par le bas" (un zéro n'est obtenu que si A = 0).

Notons que dans le cas où l'on cherche à calculer la taille de C (taille non définie), C peut être résultat de plusieurs relations d'influence il faut alors faire intervenir sa taille temporaire dans la formulation des règles :

Soit TET_C et TDT_C (1) les tailles entière et décimale temporaires de C ; si l'on ne veut pas risquer de perdre des chiffres significatifs, l'examen du noeud de division pousse à prendre comme nouvelles tailles temporaires de C la plus grande des tailles entières et la plus grande des tailles décimales) c'est-à-dire le triplet :

$(Sup\ (TET_C, 7) + Sup\ (TDT_C, 4),\ Sup\ (TET_C, 7),\ Sup\ (TDT_C, 4))$

où $7 = TE_A + TD_B$ et $4 = TD_A + TE_B$

ainsi pour C := A/B

$$\mathcal{R}_{GT} = \left\{ \begin{array}{l} TET_C := Sup\ (TET_C,\ TE_A + TD_B) \\ TDT_C := Sup\ (TDT_C,\ TD_A + TE_B) \\ TTT_C := TET_C + TDT_C \end{array} \right\}$$

(1) pour l'évaluation des tailles inconnues tout sigle désignant des tailles ainsi constitué :

T pour "taille"

E pour "entière" ; D pour "décimale" ou T pour "totale"

T pour "temporaire"

l'indice en bas à droite indique le mot dont on considère la taille.

constitue les règles de calcul de la nouvelle taille temporaire de C en fonction de son ancienne valeur et des tailles évaluées des opérandes.

Le développement logique que nous venons de faire pour la division peut être appliqué aux autres types de groupes opératoires ; nous appelons ce type de construction des règles : "construction par biais systématique" ; le tableau 1 de l'annexe 3 fournit l'ensemble des règles systématiques, pour la typologie Cobol, dans le cas de l'évaluation des tailles inconnues.

Rappelons que le choix que nous avons fait dans la loi du cheminement implique qu'au cours des modifications en cascade, on procède par "variation", c'est-à-dire que l'on tienne compte des volontés de troncature contenues dans les tailles initiales.

On est alors conduit à la formule : (1)

$$\mathcal{R}_{GT}(C:=A/B) = \left\{ \begin{array}{l} TEE_C := Sup(TEE_C, inf(TEE_A + TDE_B, TEI_C + (TEE_A - TEI_A) + (TDE_B - TDI_B)) \\ TDE_C := Sup(TDE_C, inf(TDE_A + TEE_B, TDI_C + (TDE_A - TDI_A) + (TEE_B - TEI_B)) \\ TTE_C := TEE_C + TDE_C \end{array} \right\}$$

Toutefois, si ces formules sont "techniquement" acceptables, elles présentent lorsqu'on les applique dans le contexte d'applications de gestion un certain nombre de lacunes, la principale étant sans doute un "surdimensionnement" inutile :

(1) pour les modifications en cascade les sigles de tailles sont ainsi constitués :

T pour "taille"

E, D ou T pour "entière", "décimale" ou "totale"

I pour "initiale" ou E pour "évolutive"

l'indice en bas à droite indique le mot dont on considère la taille.

Ainsi, si l'on sait que $B < 1$ est une relation impossible (ce qui peut se produire, par exemple, dans toute application bancaire où on interdit les manipulations inférieures à un franc) alors :

$$\mathcal{R}_T\ (C=A/B) = \left\{ \begin{array}{l} TET_C := Sup\ (TET_C, TEA) \\ TDT_C := Sup\ (TDT_C, TD_A + TE_B) \\ TTT_C := TET_C + TDT_C \end{array} \right\}$$

est une nouvelle formulation des règles de division pour l'évaluation des tailles inconnues qui minimise l'occupation mémoire.

Ainsi, la taille temporaire calculée pour C devient (9, 5, 4) à la place de (11, 7, 4) ; ce gain de place (2 caractères) est d'autant plus appréciable que la fréquence d'apparition de C est plus grande (par exemple : élément de table).

Un autre exemple où l'on est naturellement conduit à définir des règles particulières au type de l'application que l'on traite est celui de la comptabilité où l'on sait par avance que la taille décimale vaut au plus deux.

Enfin des conditions propres au langage ou à l'ordinateur sont susceptibles d'intervenir dans la formulation des règles : ainsi sachant qu'en Cobol, le nombre total de chiffres des opérandes est borné, il est logique de proposer :

$$\mathcal{R}_T\ (C:=A/B) = \left\{ \begin{array}{l} TET_C := Sup(TET_C, TE_A + TD_B) \\ TDT_C := Sup(TDT_C, TD_A + TE_B) \\ TTT_C := TET_C + TDT_C \\ \underline{si}\ \ TTT_C > 18 \\ \underline{alors}\ \ \underline{si}\ \ TET_C > 18 \\ \qquad \underline{alors}\ \ TET_C := 18 ; TDT_C := 0 \\ \qquad \underline{sinon}\ \ TDT_C := 18 - TET_C \\ \qquad \underline{finsi}\ ; \\ \qquad TTT_C := 18 \\ \underline{finsi} \end{array} \right\}$$

Cette règle n'est d'ailleurs pas la seule possible : ici, on a convenu d'accorder le "plafond" (18 pour CII) à la taille entière et de "sacrifier" la taille décimale, ce qui est évidemment une option parmi d'autres.

.../...

Malgré l'existence de "formules systématiques", l'analyse précédente montre qu'en fonction des objectifs poursuivis, des problèmes traités, des langages utilisés... etc... on est conduit à particulariser la formulation des règles.

Aussi, nous a-t-il semblé préférable (contrairement aux problèmes du graphe et des cheminements où une solution théorique précise a pu être proposée) de laisser à l'utilisateur le soin de choisir les formules les mieux appropriées à son problème.

Ce choix suppose que l'on mette à la disposition du gestionnaire un langage qui lui soit facilement accessible, ce langage est celui que propose "Remora" pour l'analyse d'un problème à partir de descripteurs de traitements.

La section suivante sera consacrée à la présentation d'une description de règles sur ces documents et à leur utilisation pour "paramétrer" les opérateurs de datamatique.

deuxième partie

chapitre 2

Section 3 :

Utilisation des descripteurs de "Remora"

Paragraphe 1

Description des "paramètres"

(2.2.3.1)

Les descripteurs de traitements de " Rémora " sont des outils du système ayant

. un objectif : limiter la saisie des traitements d'une fonction à automatiser aux seuls "paramètres" ;

. et trois caractéristiques : ce sont des outils d'analyse, de communication et de programmation,

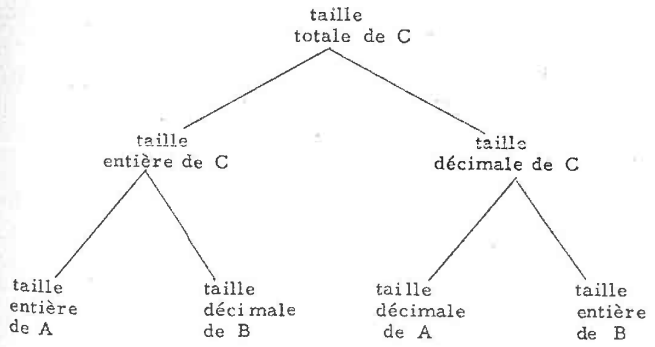
points que nous avons déjà faits apparaître dans la première partie (on peut se reporter à Remora 1, 2, 3 (Rolland, 1974))

Les descripteurs permettent de décrire les traitements variables ou "paramètres" (par opposition aux traitements figés de la "charpente") dans un mode que nous appelons "déclaratif" (Codasyl, 1972) ou "procédural" (Le Moigne, 1973).

Contrairement au mode impératif qui est celui des langages actuels de programmation dans lequel tous les détails du traitement sont imposés localement, dans l'ordre de leur exécution, sur le texte source, le mode déclaratif satisfait les désirs de l'analyste qui souhaite ignorer les détails techniques de réalisation du problème au moment où il l'analyse et le décrit .

Le mode déclaratif s'allie à une méthode de conception descendante (Crocus, 1974) "on part du résultat que l'on souhaite obtenir et on définit par étapes une solution conduisant à ce résultat".

Ainsi, par exemple, pour décrire un traitement arborescent tel que le calcul de la taille entière du résultat de la division ébauché page 135



.../...

l'analyste procédera naturellement par étapes successives de la racine vers
feuilles :

désignation des éléments	...	identificateur
taille totale de C		TT_C
taille entière de C		$+TE_C$
taille entière de A		$+TE_A$
taille décimale de B	...	$+TD_B$
taille décimale de C		$+TD_C$
taille décimale de A		$+TD_A$
taille entière de B		$+TE_B$

Remarques

- Les niveaux de l'arborescence sont représentés par simple décalage graphiq
- Les sommes algébriques telles que taille entière de C = taille entière de A + décimale de B sont traduites simplement en signant (+ ou -) tout élément d être additionné ou soustrait aux noeuds de même niveau pour constituer le noeud "père" du niveau supérieur.

Au fur et à mesure qu'il procède à cette description, l'analyste la co en exprimant, si c'est nécessaire, le conditionnement associé à l'élément dé et en définissant les transformations qui permettent de passer d'un noeud à u autre.

...

Ainsi, la formule plus complète de calcul des tailles de la page 138 sera explicitée sous la forme suivante :

désignation des éléments	condition	calcul	données	identificateur
t. totale temporaire de C				TTT_C
t totale intermédiaire				WTT_C
t. entière intermédiaire		sup(TET_C, TER)		$+WTE_C$
ancienne t. e. t. de C				TETC
t. entière de A/B				TER
t. entière de A				+TEA
t. décimale de B				+TDB
t. décimale intermédiaire		sup(TDT_C, TDR)		+WTDC
ancienne t. d. t. de C				TDTC
t. décimale de A/B				TDR
t. décimale de A				+TDA
t. entière de B				+TEA
nouvelle t. entière temporaire de C				$+TET_C$
sans majoration	$WTE_C<18$			$+WTE_C$
avec majoration	$WTE_C>18$		18	+BORNE
nouvelle t. décimale temp. de C				$+TDT_C$
sans majoration	$WTT_C<18$			+WTDC
avec majoration	$WTT_C>18$	18-TETC		+MEM
mémoire de travail				MEM

Remarques

- Des éléments non signés peuvent figurer dans la description parce qu'ils sont utiles au calcul mais n'entrent pas directement dans la somme algébrique définissant un noeud de niveau supérieur (ex : TER et TDR)
- Une condition portée sur une ligne de niveau i est évidemment valable pour toutes les lignes de niveau inférieur qui en dépendent : la taille décimale ne sera bornée que si la taille totale calculée précédemment est supérieure
- Les calculs s'expriment au moyen d'expressions arithmétiques respectant les règles de priorité habituelles sur les opérateurs (ex : 18 - TET_C) ;
- Ils peuvent comporter des fonctions standards telle que la fonction sup qui a la signification habituelle :
  sup (x, y) = x si x > y sinon y.
- La décomposition d'un calcul jusqu'au niveau le plus élémentaire fait apparaître des éléments dont la valeur est une donnée du problème (ex : 18 ou TE_A, TE_B, TD_A, TD_B) qui peut éventuellement être notée dans la colonne

Le mode déclaratif impose enfin un système de description modulaire (Dkjistra, 1972), qui calque les mécanismes de la pensée : un esprit humain ne peut en effet maîtriser un problème qu'en le décomposant en problèmes partiels plus élémentaires, chacun d'eux pouvant être, à son tour et pour une meilleure compréhension, partagé en tâches plus élémentaires.
A cette image, la description des paramètres sur un descripteur (le "père") pourra se faire en plusieurs étapes, simplement en désignant par un nom de descripteur (le "fils") un ensemble partiel de traitements qui y seront décrits plus tard.
par exemple

désignation des éléments	...	appel
majoration à 18	...	MAJØRTAIL

remplace les traitements de majoration sur le descripteur précédent et reporte leur définition à plus tard ; en outre, ces traitements intervenant pour tous les types de groupes opératoires à résultats numériques, auront été décrits une seule fois sur le descripteur "fils" MAJØRTAIL et appelés dans plusieurs descripteurs "pères".

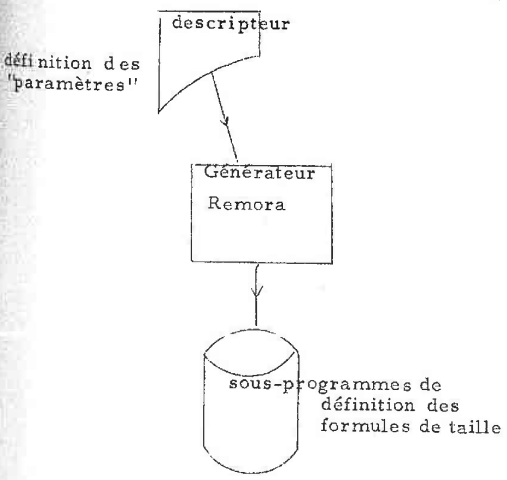
Les générateurs (Ory, 1972, Lamy, en cours de réalisation) de "Remora" traduisent les descripteurs en un langage objet, qui est un langage de programmation classique ; aussi doivent-ils assurer le passage du mode déclaratif au mode impératif : pour cela, ils s'appuient sur la structure du descripteur, structure figée et préétablie qu'il est possible d'interpréter automatiquement pour reconnaître la nature des traitements décrits et les remettre dans l'ordre où ils seront normalement exécutés.

Ainsi les traitements décrits page 142 seront traduits dans l'ordre suivant :

- calcul de la taille entière de C
- calcul de la taille décimale de C
- calcul de la taille totale de C

c'est-à-dire en remontant l'arborescence qui a été définie du haut en bas.

Le résultat de la génération est ainsi un sous-programme incorporable au cheminer :



deuxième partie
chapitre 2

Section 3 :
Utilisation des descripteurs de "Remora"

paragraphe 2

Charpente spécifique
de la définition
des formules de calcul de tailles

(2.2.3.2)

On peut encore améliorer la solution proposée précédemment pour la définition des formules de calcul de taille.

Il est possible, en effet, de mettre en évidence une charpente spécifique de ce problème, (donc qui ne peut être prise en compte par le générateur Remora)ce qui permet de limiter encore la saisie des "paramètres".

L'existence de cette charpente apparaît sur un exemple plus complexe tel que celui de la figure 1 décrivant la règle associée à un noeud de additif ou soustractif.

DESIGNATION DES ELEMENTS	Condition	Indicateur ou occurrence	CALCUL	Contrôle	ELEMENT DE RESULTAT	ANNEAU
initialisations						
cumul taille entière				0	CTE	
cumul taille décimale				0	CTD	
compteur d'opérandes				0	CPTØ	
compteur de signe				0	REPS	
boucle sur les opérandes		*				
modification de CTE			sup(CTE,TEEØ)		CTE	TEEØ
t.entière évolutive de l'opé.						
modification de CTD			sup(CTD,TDEØ)		CTD	TDEØ
t.décimale évolutive de l'opé.						
modification de CPTØ			CPTØ + 1		CPTØ	
modification de REPS			REPS + 1		REPS	
repère de signe de l'opé.	SGNØ = 'S'				SGNØ	
calcul de la retenue					RET	
soustract. à opé. non signés	codus='-' et reps = 0		CPTØ-1		CPTØ	CØDUS
code de l'influence						+CPTØ
1 opérande	CPTØ=0					+ 1
moins de 10 opérandes	CPTØ ≤10					+ 2
au delà de 10 opérandes	CPTØ >10					
boucle sur les résultats						
résultat numérique	TDER numérique	*			TDER	
taille déc.évolut. du rés.					TTER	+TEER
t.totale évolutive du rés.						+TDER
nouv. t. entière	TDER = 0		sup(CTE+RET,TEER)			
nouv. t. décimale	TDER ≠ 0		0			
"	TTER > 18		sup(CTD,TDER)			+TDER
majoration à 18						
résultat non numérique	TDER non numérique					MAJØRTAIL
nouv. taille totale			sup(CTE+CTD+RET,TTER)		TTER	

FIGURE 1.

On constate en effet que cette description est faite suivant un schéma dont on vérifie qu'il est commun à tous les types de noeud :

initialisations

pour chaque opérande

exécution des calculs relatifs à l'opérande

fin pour chaque

traitements intermédiaires

pour chaque résultat

exécution des calculs relatifs au résultat

fin pour chaque

traitements finaux

Remarque :

Les traitements initiaux et finaux sont souvent facultatifs (ex : cas de la division)

Les traitements intermédiaires permettent de faire des calculs relatifs à l'ensemble des opérandes (ex : calcul de la retenue)

Les impressions en italiques correspondent aux traitements communs à toutes les définitions de tous les types de règle.

Les impressions normales correspondent aux "paramètres" réels du problème.

On peut encore simplifier la tâche du concepteur en définissant une fois pour toute la charpente sous forme d'un sous-programme et en limitant les descriptions qu'il doit faire à celle des paramètres spécifiques de chaque cas particulier.

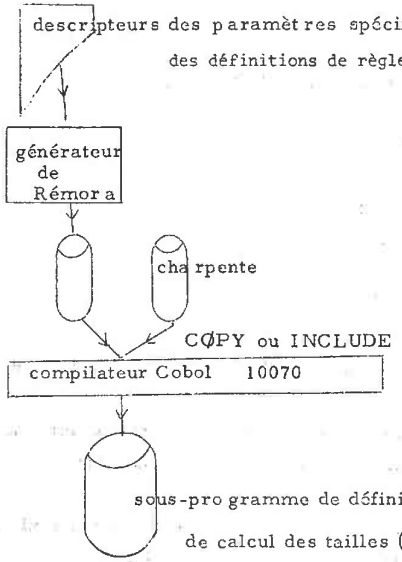
Il le fera sur des descripteurs dont le nom est imposé et la fonction bien définie :

INITAS pour l'initialisation
ØPERAS pour le traitement d'un opérande
INTERAS pour le calcul de la retenue
RESULAS pour le traitement d'un résultat

} d'un noeud additif ou soustractif

(voir le tableau 3 de l'annexe 3)

L'insertion des règles dans les opérateurs de datamatique se fera alors suivant le schéma :



En fait, le gestionnaire aura à décrire en général, pour un problème donné, plusieurs descriptions du type INITAS, ØPERAS... (une pour chaque type de noeuds).

Afin d'avoir un sous-programme standard indépendant du nombre de types de noeuds traités dans le problème considéré, nous limitons le nombre d'appels de descripteurs depuis la charpente à quatre appels principaux dont nous imposons les noms (INIT, ØPER, INTER, RESUL). Ces appels correspondent à des descripteurs "pères" aiguillant vers des descripteurs "fils" spécifiques de chaque type de noeuds et dont le nom est choisi, cette fois, par le concepteur (colonne "appel").

Par exemple RESUL est un descripteur "père" qui va appeler selon le code du noeud (par exemple le noeud additif) le descripteur "fils" correspondant (par exemple un descripteur du type RESULAS pour l'addition).

On trouvera en annexe 3 un autre exemple : le noeud multiplicatif (tableau 3) et l'ensemble des règles arithmétiques programmées en Cobol en utilisant la charpente présentée ci-dessus (figure 4)

Remarque :

On peut constater une sorte d'étreinte fatale :

- D'une part les descripteurs de Remora et leurs générateurs sont nécessaires pour "paramétrer" les opérateurs de datamatique.
- D'autre part les générateurs utilisent les opérateurs de datamatique pour calculer la taille des mémoires intermédiaires des descripteurs.

Ainsi les formules obtenues à partir du biais systématique et contenues dans les opérateurs de datamatique deviennent indispensables pour initialiser le processus.

2ème partie

Chapitre III

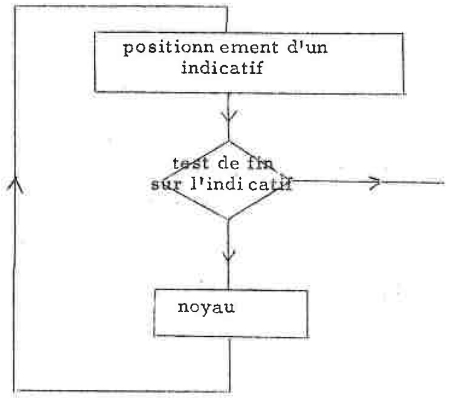
Conception de l'analyseur

(3.2)

Suivant le même principe qu'au paragraphe précédent et comme nous l'avons suggéré dans la première partie, il est possible de mettre en évidence une charpente spécifique des travaux de l'analyseur, traduisible en un sous-programme écrit une fois pour toutes, dans lequel viendront s'imbriquer des modules générés à partir de descripteurs .

En fait les travaux de l'analyseur assurant d'une part la "reconnaissance" des mots et de leurs tailles et des groupes opératoires et d'autre part le passage de leur expression "source" à leur expression normalisée peuvent être décomposés en charpente et paramètres.

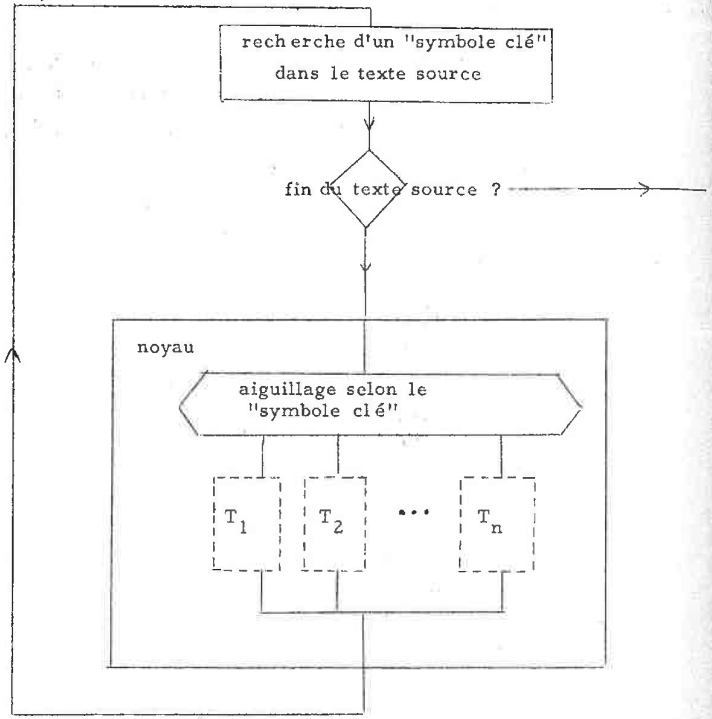
Ainsi reconnaissance est un traitement itératif dont on sait que la structure est standard :



Dans notre cas, l'indicatif est un "symbole clé". Ce symbole peut etre un mot réservé du langage (par exemple les verbes Cobol) ou un caractère spécial ayant un rôle précis (par exemple le signe d'affectation = ou : = en fortran ou en PL1).
Le test est celui de l'existence d'un tel symbole avant la fin du texte source que l'on analyse.
Il est évident qu'il faudra disposer d'une table de ces symboles.

.../...

Le noyau qui assure à partir d'un symbole à la fois la reconnaissance des éléments du groupe opératoire ou du mot et leur normalisation peut lui-même structuré de façon standard, à partir d'un aiguillage sur le symbole clé :



Sur ce schéma la charpente est en traits pleins et les paramètres en traits pointillés.

En fait les traitements spécifiques sélectionnés par l'aiguillage comportent encore une racine commune pour tous qu'il sera donc possible d'écrire une fois et d'appeler par son nom dans chacun des pavés.

.../...

Cette racine correspond au passage à la forme normalisée, une fois reconnue la liste des éléments (liste des opérandes, liste des résultats ou mot déclaré et liste de ses tailles).

Ainsi lorsqu'à partir du groupe opératoire

ADD A B C GIVING R1 R2.

reconnu par le mot clé ADD, un traitement spécifique a permis de reconnaître les listes A, B, C et R1, R2 ;

A B C	+	R1 R2

est standard (voir le paragraphe 3.1.1.3).

En fait, bien qu'elle soit un paramètre, la procédure de reconnaissance des listes (A, B, C et R1, R2 sur l'exemple) peut être résolue de la même façon pour tous les groupes opératoires d'un même langage et quelque soit ce langage, en effet elle consistera en général à définir des zones délimitées par des mots clés secondaires. (exemple : la reconnaissance de GIVING et du délivre les blocs | A B C | et | R1 R2 R3 |, la reconnaissance du blanc les éléments individuels A, B, C, D, R1, R2.

Notons que cette possibilité de "génération" d'analyseurs spécifiques, qui reprend l'idée force de "Remora" : charpente de traitements communs et "paramètres", n'était pas la seule envisageable ; en utilisant la théorie de la compilation (Fair, 1972) et en particulier l'analyse syntaxique on pouvait concevoir un constructeur d'analyseurs spécifiques par exemple en utilisant FACE : (Maroldt, Bellegarde, 1972) dont les données principales sont la grammaire du langage source et la "sous-grammaire" des groupes opératoires générateurs d'influences.

Troisième partie :

Réalisations pratiques

et résultats

plan de la troisième partie

Chapitre 1
l'analyseur
Section 1
fonctionnement
paragraphe 1
structure physique du C. I. N.
paragraphe 2
étude morphologique
paragraphe 3
remarques de programmation
Section 2
résultats
Section 3
éditions complémentaires

Chapitre 2
les opérateurs de cheminement
Section 1
utilisation
Section 2
analyse et résultats
paragraphe 1
constructeur du graphe
paragraphe 2
chemineur

troisième partie

Chapitre 1 :
l'analyseur

(3.1)

troisième partie

chapitre 1

Section 1 :

fonctionnement

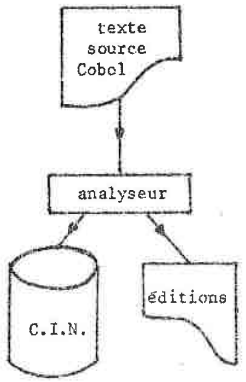
(3.1.1)

Rappelons que pour pouvoir tester immédiatement les opérateurs de cheminement sur des cas concrets, nous avons réalisé un analyseur de programmes écrits en Cobol.

Comme nous l'avons mis en évidence au chapitre 2.3. pour le logiciel "standard", l'analyseur a deux rôles distincts :

- la reconnaissance des éléments du complexe d'influences dans le texte source Cobol ;
- leur codage pour obtenir le complexe d'influences normalisé (ou C.I.N.), accessible aux opérateurs de cheminement.

Ces deux rôles sont assumés par le même programme, si bien que son fonctionnement peut être schématisé par :



Nous définirons au paragraphe 1 la structure physique du complexe d'influences normalisé pour reprendre ensuite au paragraphe 2 chacun des deux rôles de l'analyseur du point de vue Cobol et compléter l'étude au paragraphe 3 par des remarques sur la programmation.

troisième partie

chapitre 1

Section 1 :

fonctionnement

paragraphe 1

structure physique du C. I. N.

(3. 1. 1. 1)

Travaillant sur une exploitation classique, qui ne propose, pour gérer les données, qu'un système de gestion de fichiers (ou S.G.F.), nous représentons le C.I.N. sous forme de deux fichiers :

- un fichier des mots munis de leur caractéristique (la taille)
- un fichier de noeuds d'influence.

A chaque mot du C.I.N. est associé une référence unique (un numéro) qui donne accès à l'article correspondant du fichier.

De même, la numérotation des noeuds d'influences sert de clé d'accès.

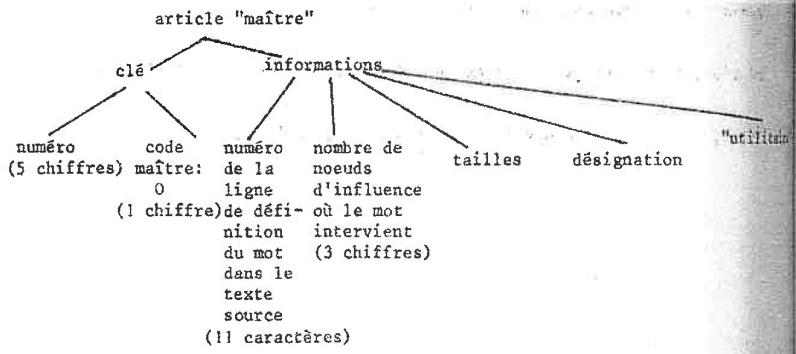
Remarque : ces fichiers sont organisés en séquentiel indexé.

(3.1.1.1.1.) **Fichier des mots**

Pour simplifier la tâche du constructeur du graphe, le fichier des mots comporte deux types d'articles :

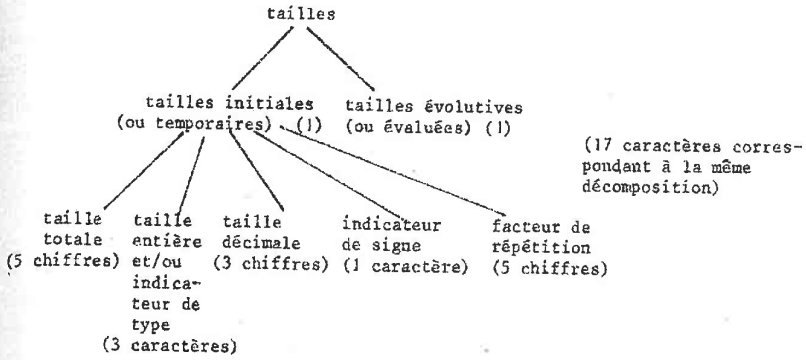
- Les articles "maîtres" qui comportent les informations fondamentales (exemple : les tailles) ou les informations fixes (exemple : nombre de noeuds adjacents contenant ce mot, "utilitaire") : il y a un article maître pour chaque mot.
- Les articles "secondaires" qui contiennent les informations variables utilisées pour optimiser la construction du graphe des noeuds d'influence (mais non au cours du cheminement sauf pour le noeud (ou pseudo-noeud) de définition) : il peut éventuellement y avoir plusieurs articles secondaires pour un mot (*).

Un article "maître" a la structure suivante :



(*) en fait, il suffirait d'un article de longueur variable pour chaque mot, mais la manipulation des fichiers séquentiels indexés à enregistrement variable étant "aléatoire" sur CII 10070, nous avons préféré choisir la sécurité. (Notre solution est d'ailleurs plus rapide et sa "portée", en cas de changement de machine, est supérieure).

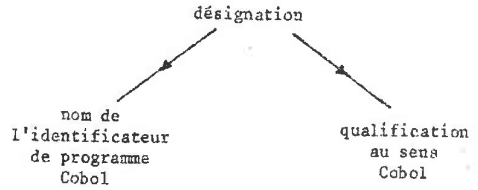
où la sous-structure tailles se décompose en :



(17 caractères correspondant à la même décomposition)

Le tableau 2 de l'annexe 4 donne un échantillon des contenus possibles de cette zone taille.

L'analyseur garnit la zone "désignation" (qui est destinée à recevoir un texte explicatif de chaque entité dans le répertoire des mots de REMORA) par :



La qualification au sens Cobol (ou PL1) consiste à préciser le nom d'un élément d'une structure par celui de son père et/ou celui du grand-père....
Ceci permet en particulier de pouvoir donner, sans ambiguité, le même nom à deux zones distinctes puisque la qualification par des "ancêtres" différents permettra de les différencier (2).

(1) selon que l'on considère les modifications en cascade ou l'évaluation des tailles non déclarées.

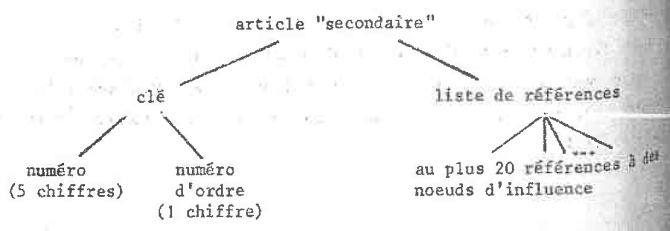
(2) donner le même nom à deux zones distinctes permet d'utiliser les verbes Cobol ADD, SUBTRACT et MOVE avec l'option CORRESPONDING (ou CORR).

Nous limitons l'utilisation de la qualification à la racine de l'arborescence, c'est-à-dire que nous imposons donc que tout qualificateur soit décrit par un niveau 01 (limite que nous avons estimée peu contraignante pour le programmeur). La zone "utilitaire" est initialisée à des valeurs utiles au constructeur du graphe (pointeurs).

La structure d'un article maître est décrite en Cobol par :

```
00064                   01  ID-IDEN.
00065                       02  CLEID PIC X(6).
00066                       02  NBLIGNEDEF PIC X(11).
00067                       02  NBREF PIC 9(3).
00068                       02  TAILLES.
00069                           03  TAILLE-I.
00070                               04  TTI PIC 9(5).
00071                               04  TEI PIC 9(3).
00072                               04  TEIX REDEFINES TEI.
00073                                   05  TEIX1  PIC X.
00074                                   05  FILLER  PIC XX.
00075                               04  TDI PIC 9(3).
00076                               04  REPI PIC 9(5).
00077                               04  SGNI PIC X.
00078                           03  TAILLE-E PIC X(17).
00079                       02  NOM-IDEN PIC X(25).
00080                       02  QUALIFICATION PIC X(24).
00081                       02  ZTRAV.
00082                           03 ZPOINT PIC X.
00083                           03 ZREFT PIC S9(5) COMP-3.
00084                           03 ZREFO PIC S9(5) COMP-3.
```

Un article "secondaire" est hiérarchisé suivant le schéma :



Ce qui correspond à la description Cobol :

```
01  ID-SUITE.
    02  CLEID-CRE.
        03  NOIDN PIC 9(5).
        03  NOID REDEFINES NOIDN.
            04 TID PIC X OCCURS 5.
        03  SUITE PIC 9.
    02  REFERENCES.
        03  REFUS PIC S9(5) OCCURS 20.
    02  FILLER PIC X(4).
FD  TABID LABEL RECORD STANDARD DATA RECORD ID-IDEN.
```

On peut remarquer que la zone répétitive REFUS qui reçoit les références est une zone signée :

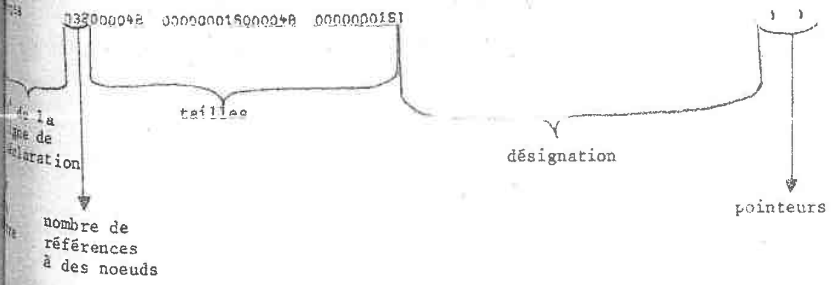
En effet, nous conviendrons

- que la référence à un noeud d'influence où le mot est opérande reçoit un signe positif ;
- que la référence à un noeud d'influence où le mot est résultat reçoit un signe négatif.

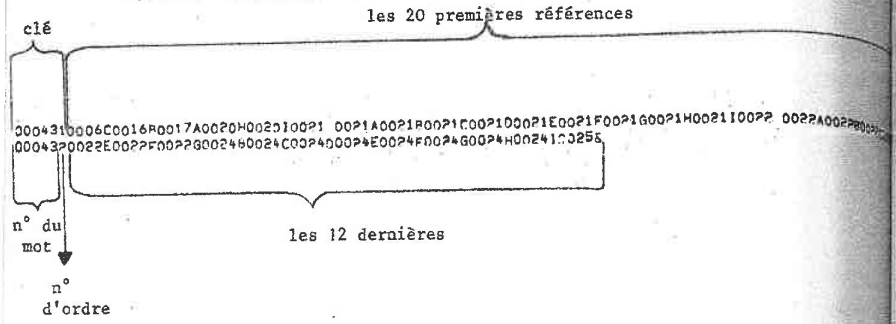
Il y a à cette codification une exception : la référence au noeud de définition du mot est toujours positive ; on la reconnaît comme étant la première référence du premier article secondaire ; elle est la seule à être utilisée à la fois par le constructeur du graphe et par le chemineur.

Exemples d'articles du fichier des mots :

article "maître" :



articles "secondaires" :



Remarque : Compte tenu de la représentation Cobol choisie, le signe se trouve superposé à droite :
par exemple :

0006C ⟺ + 00063
0002N ⟺ - 00205

La correspondance est résumée dans le tableau ci-dessous :

valeur du chiffre	0	1	2	3	4	5	6	7	8	9
signe + superposé	␣	A	B	C	D	E	F	G	H	I
signe - superposé	δ	J	K	L	M	N	O	P	Q	R

(3.1.1.1.2.) Fichier des noeuds d'influence

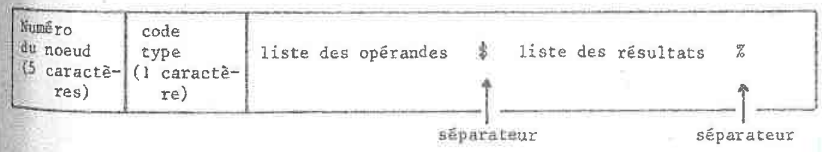
Un noeud d'influence est un triplet :
(opérandes, règles, résultats)
où

- les opérandes sont constituées d'une liste de références : une référence étant
  - soit I (pour identificateur) suivi du numéro du mot
    (ou R dans les noeuds hiérarchiques pour les synonymes)
  - soit L (pour libellé) suivi de la taille totale de cette constante non numérique.
  - soit N (pour numérique) suivi des informations de tailles de cette constante numérique.

- les règles sont réduites :
  - soit à un code de type pour les influences élémentaires,
  - soit à une référence d'U.T. pour les macro-influences

- les résultats sont formés d'une liste de références à des variables (I numéro).

Pour des commodités pratiques, les informations fixes sont groupées au début, suivies des informations variables encadrées par des séparateurs :



Ce qui correspond à la description Cobol (*)

```
00046                01  US.
00047                    02  EFUS PIC X.
00048                    02  CLEUS PIC 9(5).
00049                    02  REGLEUS.
00050                        03  CODUS PIC X.
00051                        03  DONNEESUS.
00052                            04  ELUS PIC X OCCURS 278.
```

(*) EFUS désigne un caractère d'effacement, dans nos résultats il a toujours la valeur arbitraire A.

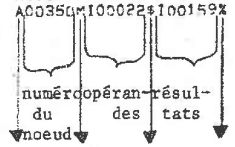
troisième partie
chapitre 1

Section 1 :
fonctionnement
paragraphe 2
étude morphologique

(3.1.1.2)

Exemples de noeuds d'influence codés :

```
A00311MN00001001000 $I00079%
A00312MI00006$I00076%
A00313+I00088I00095$I00095%
A00314MI00060$I00079%
A00315MN00001001000 $I00069%
A00316/I00069I00095$I00097%
A00317+I00097I00060$I00060%
A00318+I00097I00060$I00060%
A00319-I00097I00097I00069$I00069%
A00320+I00097I00097I00060$I00060%
A00321-I00097I00097I00069$I00069%
A00322+I00069I00060$I00060%
A00323MI00060$I00079%
A00324ML00006          $I00098%
A00325/I00069I00090$I00097%
A00326+I00097I00060$I00060%
A00327+I00097I00060$I00060%
A00328-I00097I00097I00069$I00079%
A00329+I00097I00060$I00060%
A00330-I00097I00069$I00079%
A00331MI00069$I00079%
A00332MN00001001000 $I00069%
A00333MN00001001000 $I00162%
A00334MI00184$I00124%
A00335MI00088$I00081%
A00336ML00003          $I00096%
A00337ML00003          $I00096%
A00338MI00052$I00122%
A00339MI00053$I00123%
A00340MI00054$I00142%
A00341MI00055$I00151%
A00342MI00056$I00153%
A00343MI00052$I00122%
A00344MI00053$I00123%
A00345MI00017$I00144%
A00346MI00018$I00146%
A00347MI00019$I00148%
A00348MI00020$I00155%
A00349MI00021$I00157%
A00350MI00022$I00159%
```



EFUS code sépa-rateur sépa-rateur

Tout texte source Cobol est composé de quatre divisions :

- une pour donner les coordonnées du programme (numéro, auteur, dates, ...)
- une pour décrire l'ordinateur et les liens avec l'extérieur
- une pour la description des données
- une pour la description des traitements.

Seules les deux dernières : la DATA division et la PROCEDURE division, fournissent des éléments au complexe d'influences.

La data division est partagée en sections spécialisées où sont déclarées et décrites les données. Une section joue toutefois un rôle particulier : "la report section" où sont décrits les états de l'éditeur Cobol et qui comporte à la fois des descriptions de données et des descriptions de traitements.

(3.1.1.2.1.) **Décodage de la data division : sections classiques**

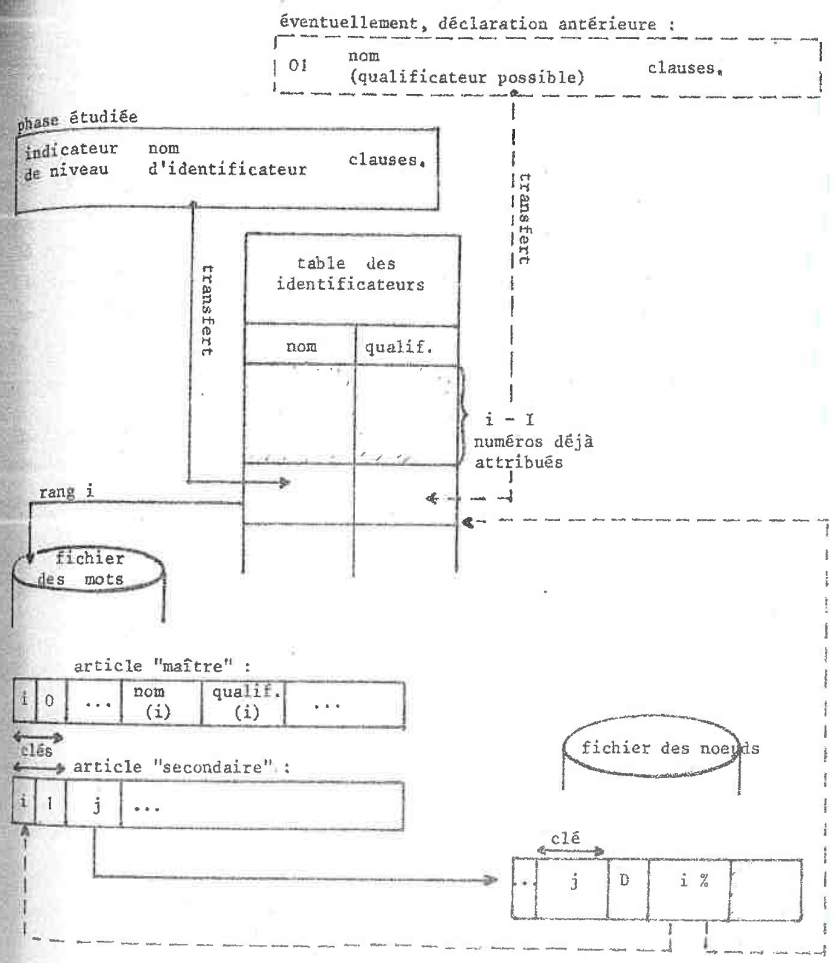
Le décodage de la data division s'effectue phrase par phrase puisqu'en effet, elle est composée de sections apparaissant comme une succession de phrases qui ont toutes la structure :

indicateur de niveau ␣ nom d'identificateur ␣ liste des clauses,

ce qui constitue la déclaration d'un mot. Le décodage d'une phrase entraîne donc :

- d'une part l'introduction dans le fichier des noeuds du noeud de définition associé,
- d'autre part la création dans le fichier des mots de l'article maître correspondant, ainsi que du premier article secondaire qui contient la référence au noeud de définition.

Pour faciliter les techniques d'accès aux fichiers, on remplace, dans ce travail, la désignation Cobol d'un mot au moyen de son identificateur (ou plusieurs identificateurs réunis par le symbole de qualification), par un numéro qui devient sa référence unique.
Ceci suppose la gestion d'un répertoire des mots ou "table des identificateurs" assurant la correspondance entre l'identificateur (éventuellement qualifié) et le numéro qui lui est associé.
Une première approche du décodage peut ainsi être schématisée par :



Pour garnir complètement l'article "maître" du fichier des mots, on a besoin des informations sur la taille du mot ce qui, selon le cas, est plus ou moins complexe : pour les éléments indépendants n'appartenant pas à une structure, l'étude de la liste des clauses (en particulier la clause PICTURE) fournit immédiatement les informations nécessaires (qui peuvent être "taille non déclarée").

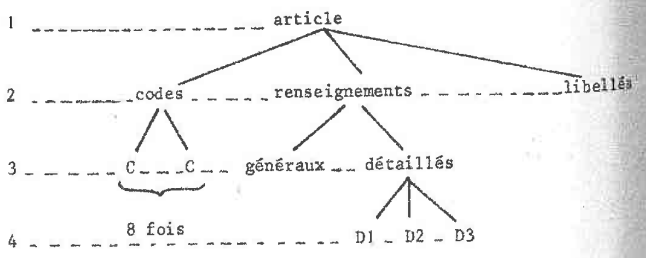
Au contraire, pour les éléments qui appartiennent à une structure, deux problèmes se posent :

- il faut, d'une part, construire les noeuds hiérarchiques correspondants, les introduire dans le fichier des noeuds et compléter les références à ces noeuds dans les articles secondaires des mots qui y sont opérandes ou résultats.
- il faut, d'autre part, calculer la taille des mots "structurés" à partir de celle des mots de niveau supérieur. On peut remarquer que ce problème peut être résolu par la méthode de détermination des tailles inconnues que nous avons développée ; en fait, nous utilisons ici une autre méthode qui consiste à calculer les tailles en simultanéité avec la constitution des noeuds hiérarchiques.

Ceci suppose l'exploitation regroupée de plusieurs phases consécutives (toutes celles qui appartiennent à une même description de structure).

Le schéma du décodage proposé, page172 est dans ce cas incomplet ; il doit être complété par le travail suivant :

Soit la structure



décrite en Cobol par :


```
01  ARTICLE.
    02  CØDES.
        03  C  PIC[TURE]  X ØCCURS  8.
    02  RENSEIGNEMENTS.  (*)
        03  GENERAUX PIC X (25).
        03  DETAILLES.
            04  D1  PIC  9(5)V99.
            04  D2  PIC    S9(8).
            04  D3  PIC  X(15).
    02  LIBELLES  PIC  X(70).
```

Supposons que la première partie du décodage remplisse ainsi la table des identificateurs :

	nom	qualification
1		
.		
.		
.		
20	ARTICLE	
21	CØDES	ARTICLE
22	C	"
23	RENSEIGNEMENTS	"
24	GENERAUX	"
25	DETAILLES	"
26	D1	"
27	D2	"
28	D3	"
29	LIBELLES	"

Les noeuds hiérarchiques à construire sont :

n° H 21 23 29 $ 20 % ("codes", "renseignements" et "libellés" constituent "article")
n° H 22 $ 21 % ("c" constitue "codes")
n° H 24 25 S 23 % ("généraux", "détaillés" constituent "renseignements")
n° H 26 27 28 $ 25 % ("D1", "D2", "D3" constituent "détaillés").

(*) PICTURE peut être abrégé et noté PIC

Pour les obtenir, il faut analyser l'arborescence représentée en Cobol par la technique des "nombres niveaux" ; un algorithme à pile permet cette fois encore de résoudre le problème ; on peut le résumer ainsi :

- prendre une pile p vide ;
- mettre un niveau fictif 0 au sommet de la pile ;
- NVSPA : = 0 ; FIN : = "faux" ;
- *WTT : = 0 ;*
- tant qu'il existe des lignes de déclaration (alors EXIST = "vrai") ou que FIN = "faux" : (1)
  - si EXIST = "faux"
  - alors FIN : = "vrai" ; NBNIV : = 0
  - finsi ;
  - ART : = "vrai" ;
  - tantque NBNIV < niveau (sommet (p)) ou que ART = "faux" : (2)
    - si niveau (sommet (p)) ≥ NVSPA
    - alors mettre la référence du sommet (= une opérande) dans la zone "noeud" ;
      NVSPA : = niveau (sommet (p)) ;
      oter le sommet de p ;
      WTT := WTT + taille (sommet (p)) ;
      - si niveau (sommet (p)) = NVSPA
      - alors ART : = "faux"
      - finsi
    - sinon mettre $, la référence du sommet (= le résultat) et l
      dans la zone "noeud" ; enregistrer le noeud ;
      attribuer la taille WTT à ce résultat et la
      au sommet de p ; WTT := 0 ;
      NVSPA : = 0
    - finsi
  - fintantque
  - mettre la nouvelle déclaration au sommet de p
    (y compris la taille pour les mots élémentaires)
- fintantque

(1) le booléen FIN permet de répéter les actions du premier tantque pour chaque de déclaration et encore une fois avec une valeur fictive NBNIV : = 0

(2) le booléen ART permet d'exécuter les actions du deuxième tantque si niveau (sommet (p)) < NBNIV et de les répéter jusqu'à ce que niveau (sommet (p)) NBNIV (une dernière exécution étant effectuée sous cette condition).

Ce qui donne sur l'exemple :

déclaration considérée n° du mot NBNIV		Niveau du sommet		NVSPA	pile p (variations) (élément = (n° mot, niveau))	Noeuds hiérarchiques (= enregistré)	WTT
		0		0	(?,0)		0
20	1 >	0		0			
		1		0	(?,0)(20,1)		0
21	2 >	1		0			
		2		0	(?,0)(20,1)(21,2)		0
22	3 >	2		0			
		3		0	(?,0)(20,1)(21,2)(22,3) (1x8) (*)		0
23	2 <	3	>	0			
		2	!	3	(?,0)(20,1)(21,2)	22	8
	2 =	2	<	3			
		2		0	(?,0)(20,1)(21,2)(23,2) (8)	22$21%	0
24	3 >	2		0			
		3		0	(?,0)(20,1)(21,2)(23,2)(24,3) (25)		0
25	3 =	3		0			
		3		0	(?,0)(20,1)(21,2)(23,2)(24,3)(25,3)		0
26	4 >	3		0			
		4		0	(?,0)(20,1)(21,2)(23,2)(24,3)(25,3)(26,4) (7)		0
27	4 =	4		0			
		4		0	(?,0)(20,1)(21,2)(23,2)(24,3)(25,3)(26,4)(27,4) (8)		0
28	4 =	4		0			
		4		0	(?,0)(20,1)(21,2)(23,2)(24,3)(25,3)(26,4) (27,4)(28,4) (15)		0
29	2 <	4	>	0			
		4	!	4	(?,0)(20,1)(21,2)(23,2)(24,3)(25,3)(26,4)(27,4)	28	15
	2 <	4	=	4			
		4	!	4	(?,0)(20,1)(21,2)(23,2)(24,3)(25,3)(26,4)	28 27	23
	2 <	4	=	4			
		3	!	4	(?,0)(20,1)(21,2)(23,2)(24,3)(25,3)	28 27 26	30
	2 <	3	<	4			
		3	!	0	(30)	28 27 26 $25%	
	2 <	3	>	0			
		3	!	3	(?,0)(20,1)(21,2)(23,2)(24,3)	25	30
	2 <	3	=	3			
		2	!	3	(?,0)(20,1)(21,2)(23,2)	25 24	55
	2 =	2	<	3	(55)		
		2		0	(?,0)(20,1)(21,2)(23,2)(29,2) (70)	25 24$23%	0
0	0 <	2	>	0			
		2	!	2	(?,0)(20,1)(21,2)(23,2)	29	70
	0 <	2	=	2			
		2	!	2	(?,0)(20,1)(21,2)	29 23	125
	0 <	2	=	2	(?,0)(20,1)	29 23 21	133
		1	!	2			
	0 <	1	<	2		29 23 21 $20%	0
		1		0	(133)		
	0 <	1	>	0	(?,0)	20	133
		0	!	1			
	0 =	0	<	1		20$?%	0

On remarque qu'il faut un test d'arrêt après enregistrement de la règle dont le résultat est un niveau 01 si on ne veut pas créer une dernière règle anormale.

Il est évident que les noeuds étant obtenus dans l'ordre nécessaire à l'exécution des règles de calcul des tailles, il suffit de compléter l'analyse de l'arborescence par le calcul des tailles ; calcul réduit à de simples sommations (indications en italiques).

Un dernier type de noeud peut être rencontré lors du décodage de ces sections : il s'agit des noeuds de redéfinition qui sont dûs à la clause REDEFINES permettant la redéfinition d'un "synonyme" du point de vue de l'espace mémoire occupé. Ainsi

```
02   TAB1   REDEFINES   TAB2...
```

indique que TAB1 et TAB2 occupent le même emplacement mémoire ; le décodage de ces noeuds ne présente pas de difficulté : si nous supposons que le numéro de ces mots est respectivement 105 et 112, il faut créer les noeuds de redéfinition

```
n°  R 105 $ 112 %
et  n°  R 112 $ 105 %
```

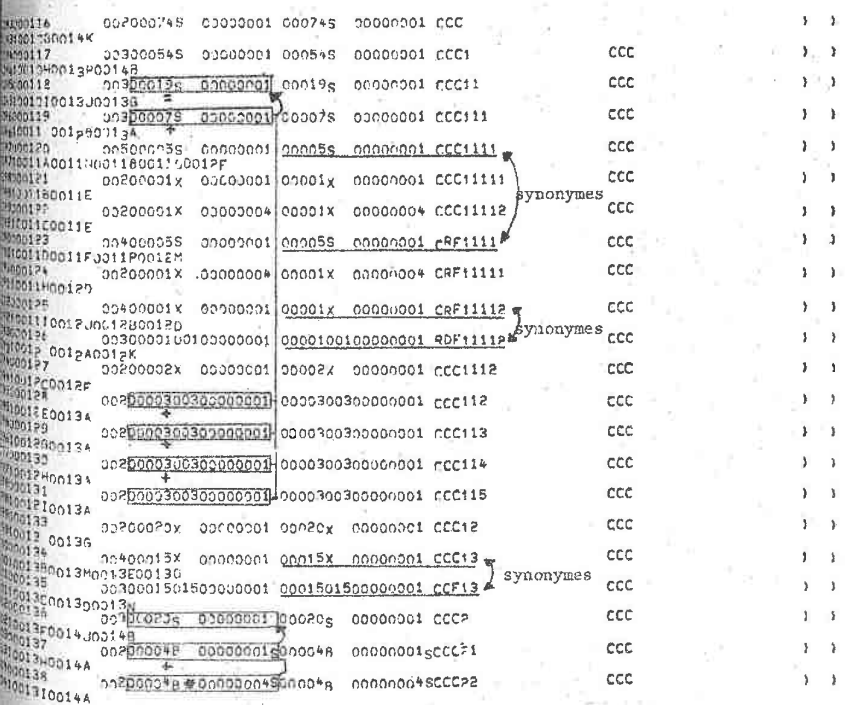
Remarquons enfin qu'en Cobol, le code F permet de distinguer une influence spéciale qui est l'appartenance d'articles à un fichier (décrit par un niveau FD ou SD) : tous les articles occupent en effet le même emplacement mémoire (redéfinition implicite). Le résultat de cette règle est le nom du fichier ; nous lui accordons une taille totale égale à la plus grande de celle de ses articles (tous les articles n'ont pas forcément la même taille) ; cette définition est indispensable puisque le nom du fichier intervient comme opérande dans les lectures accompagnées de transfert (READ INTØ).

(*) pour ne pas alourdir la représentation des états de la pile avec les informations sur les tailles, nous indiquons simplement ces informations au moment de leur entrée au sommet de la pile.

Exemples : la structure Cobol suivante :

```
01  CCC.
    02  CCC1.
        03  CCC11.
            04  CCC111.
                05  CCC1111.
                    06  CCC11111 PIC X VALUE 'I'.
                    06  CCC11112 PIC X OCCURS 4.
                05  CRF1111 REDEFINES CCC1111.
                    06  CRF11111 PIC X OCCURS 4.
                    06  CRF11112 PIC X.
                    06  RDF11112  REDEFINES CRF11112  PIC 9.
                05  CCC1112 PIC X(2).
            04  CCC112 PIC 9(3) .
            04  CCC113 PIC 9(2)9 .
            04  CCC114 PIC 9(2)9.
            04  CCC115 PIC 999
        .
        03  CCC12 PIC X(20).
        03  CCC13 PIC X(15).
        03  CCF13 REDEFINES CCC13 PIC 9(15).
    02  CCC2.
        03  CCC21 COMP.
        03  CCC22 COMP OCCURS 4.
```

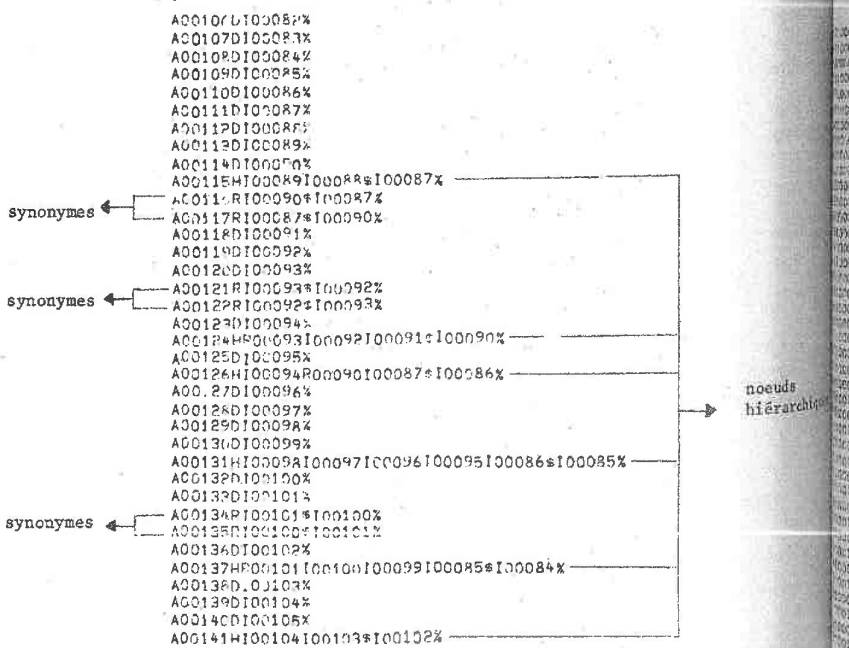
donne lieu à la création des articles du fichier des mots :



On peut constater :

- que les "synonymes" ont la même taille totale
  (exemple : CCC13 et CCF13) ;
- que dans une structure la somme des tailles totales des éléments des niveaux supérieurs fournit la taille totale de l'élément de niveau inférieure
  (exemple : CCC111, CCC112, CCC113, CCC114, CCC115 et CCC11) ;
- que cette somme est calculée en tenant compte du facteur de répétition
  (exemple : CCC21, CCC22 et CCC2)

et des articles du fichier des noeuds d'influences :



D'autre part la structure d'édition :

```
00087                  01  EDITE.
00088                      02  ED1 PICTURE
00089                      99.
00090                      02  ED2 PIC S9(5).
00091                      02  ED3 PIC AABAA.
00092                      02  ED4 PIC AxBxA.
00093                      02  ED5 PIC 9CR.
00094                      02  ED6 PIC 99V9(3).
00095                      02  ED7 PIC 99.9.
00096                      02  ED8 PIC 99.9 .
00097                      02  ED9 PIC P(5)99.
00098                      02  ED10 PIC 9(6)PP.
00099                      02  ED11 PIC SPP9.
00100                      02  ED12 PIC S99PPP.
00101                      02  ED13 PIC ++9.99.
00102                      02  ED14 PIC -(6)9.
00103                      02  ED15 PIC *(8).
00104                      02  ED16 PIC $$$.
00105                      02  ED17 PIC +ZZZ.99.
00106                      02  ED18 PIC ZZ,ZZZ.99-
00107                      02  ED19 PIC  CR999.
00108                      02  ED20 PIC  9990B.
```

fournit un échantillon de déclarations de tailles dont on peut voir le codage correspondant dans le fichier des mots :

```
00087  0020001055  05000001 00105S  00000001 EDITE                          ) )
00088  0030000200200000001 0000200200000001 ED1              EDITE          ) )
00089  0040000500500000001S0000500500000001SED2              EDITE          ) )
00090  0020000SX  C0000001 0000SX  00000C01 ED3              EDITE          ) )
00091  0020000SX  00000001 00005X  00000C01 ED4              EDITE          ) )
00092  0030000300300000001 0000300300000001 ED5              EDITE          ) )
00093  0020000500200300001 0000500200300001 ED6              EDITE          ) )
00094  0020000400200100001 00004E0200100001 ED7              EDITE          ) )
00096  0020000400200100001 00004E0200100001 ED8              EDITE          ) )
00097  0030000800300800001 0000800080080001 ED9              EDITE          ) )
00098  0020000800800800001 0000800800800001 ED10             EDITE          ) )
00099  0020000300000300001S0000300000300001SED11             EDITE          ) )
00100  0020000500500000001S0000500500000001SED12             EDITE          ) )
00101  0020000600300200001S0000600300200001SED13             EDITE          ) )
00102  0020000700700000001S0000700700000001SED14             EDITE          ) )
00103  0020000800800000001 0000800800000001 ED15             EDITE          ) )
00104  0020000300300000001 0000300300000001 ED16             EDITE          ) )
00105  0020000700400200001S0000700400200001SED17             EDITE          ) )
00106  0020000900500200001 0000900500200001 ED18             EDITE          ) )
00107  0020000400300000001S0000400300000001SED19             EDITE          ) )
00108  0020000400300000001S0000400300000001SED20             EDITE          ) )
```

(3.1.1.2.2.) **Décodage de la data division : report section**

Une phrase de la report section a la même structure que les autres phrases de la data division :

indicateur de niveau ␣ [nom d'identificateur] liste des clauses

(1)

L'originalité de cette section vient des clauses spéciales qui y sont utilisées par exemple :

1) 02 LINE 6 CØLUMN 20 PIC 9 (4) VALUE 1974.

signifie que la 6ème ligne de l'état décrit doit être garnie au moment de l'écriture à partir de la colonne 20 par la valeur 1974 en format 9 (4). L'opérande étant une constante et le résultat n'étant pas utilisé, une phrase de ce type est ignorée par l'analyseur.
Par contre, si cette ligne était nommée, il faudrait procéder à la première étape du décodage (page172). (Elle pourrait aussi être garnie par un traitement décrit dans les sections déclaratives de la division des traitements).

2) 02 LINE 12 CØLUMN 10 PIC Z (3) . 99 SØURCE MØNTANT

signifie que la 12ème ligne doit être garnie à partir de la colonne 10 avec le contenu d'un autre mot montant "édité" en format Z (3) . 99 ;
Cette phrase comprend donc un ordre de transfert dont l'opérande est MONTANT et dont le résultat est un mot, ici sans désignation, qui correspond à la donnée à imprimer : il a une taille (due à la clause PICTURE) qui peut être

(1) le crochet [ ] indique que le nom d'identificateur est facultatif.

modifiée par le processus de modification en cascade, aussi pour le désigner à l'utilisateur, nous garnirons la zone "désignation", en absence de nom d'identificateur, avec LIG : suivi du numéro de la ligne dans le texte source. Une telle ligne de déclaration demande donc, comme toute ligne de la data (cf. page172) :

- une entrée dans la table des identificateurs,
- un article "maître" et un article "secondaire" dans le fichier des mots,
- un noeud de définition dans le fichier des noeuds,

et en outre

- un noeud de transfert :

n° M | n° de mot associé à montant | $ | n° de mot attribué à cette ligne | %

3) 02 TOTAL LINE 20 CØLUMN 10 PIC ZZ,Z (3) . 99 SUM MØNTANT.

Le mot TØTAL est alors le résultat d'un cumul :
MØNTANT est additionné à TØTAL à chacune de ses impressions (TØTAL est remis à zéro sur "rupture de contrôle" (voir notice d'utilisation du Cobol CII 10070)).

Cette ligne demande donc les entrées précédentes communes à toutes les lignes de data dans les différents fichiers et table,

et en outre

- un noeud additif (qui traduit total : = total + montant) :

n° + | n° du mot MØNTANT | n° du mot TØTAL | $ | n° du mot TØTAL | %

Exemple :

A partir du texte source Cobol :

```
00155          01 FIN-D-UN-ETAT TYPE IS CONTROL FOOTING
00156             CHANGEMENT.
00157             02 LINE 45.
00158             03 COLUMN 51 PIC X(6) VALUE 'TOTAUX'.
00159             03 TOTAL-BOURSES COLUMN 58 PIC ZZZBZZZBZZZ
00160             SUM MONTANT-BOURSE.
00161             03 TOTAL-TERMES COLUMN 69 PIC ZZZBZZZBZZZ
00162             SUM MONTANT.
00163             02 LINE 48.
00164             03 COLUMN 10 PIC X(93) SOURCE SOMME-LITTERALE.
00165             02 LINE 49.
00166             03 COLUMN 49 PIC 99 SOURCE JOUR.
00167             03 COLUMN 53 PIC X(9) SOURCE MOISF.
00168             03 COLUMN 74 PIC 99 SOURCE ANNEE.
00169          01 REPORT-BAS-PAGE TYPE IS PAGE FOOTING.
00170             02 LINE 58.
00171             03 COLUMN 48 PIC X(10) SOURCE REPORT-TEXTE.
00172             03 COLUMN 58 PIC ZZZBZZZBZZZ SOURCE
00173             TOTAL-BOURSES.
00174             03 COLUMN 69 PIC ZZZBZZZBZZZ SOURCE
00175             TOTAL-TERMES.
```

la table des identificateurs est complétée par :

```
 numéro              désignation                                  n° ligne de
                                                                  déclaration
+0000000181 LIG: 00138                  ENTETE                    00138
+0000000182 TOTAL-BOURSES               FIN-D-UN-ETAT             00159
+0000000183 LIG: 00140                  ENTETE                    00140
+0000000184 TOTAL-TERMES                FIN-D-UN-ETAT             00161

+0000000196 FIN-D-UN-ETAT                                         00155
+0000000197 LIG: 00157                  FIN-D-UN-ETAT             00157
+0000000198 LIG: 00158                  FIN-D-UN-ETAT             00158
+0000000199 LIG: 00163                  FIN-D-UN-ETAT             00163
+0000000200 LIG: 00164                  FIN-D-UN-ETAT             00164
+0000000201 LIG: 00165                  FIN-D-UN-ETAT             00165
+0000000202 LIG: 00166                  FIN-D-UN-ETAT             00166
+0000000203 LIG: 00167                  FIN-D-UN-ETAT             00167
+0000000204 LIG: 00168                  FIN-D-UN-ETAT             00168
+0000000205 REPORT-BAS-PAGE                                       00169
+0000000206 LIG: 00170                  REPORT-BAS-PAGE           00170
+0000000207 LIG: 00171                  REPORT-BAS-PAGE           00171
+0000000208 LIG: 00172                  REPORT-BAS-PAGE           00172
+0000000209 LIG: 00174                  REPORT-BAS-PAGE           00174
```

Remarque :
total-bourses et total-termes ont été utilisés, donc numérotés, avant d'être déclarés.

Le fichier des mots comporte :

```
00500011E0900000000 00011E0900000000 TOTAL-BOURSES
00221 0025N0025EC027R
00200010E0800000000 00010E0800000000 LIG: 00140                ENTETE
0023K
00600011E0900000000 00011E0900000000 TOTAL-TERMES
0023R0025C0025F0027000330

0010000000000000000000000000000000000FIN-D-UN-ETAT
0010000000000000000000000000000000000LIG: 00157                FIN-D-UN-ETAT
0010000000000000000000000000000000000LIG: 00158                FIN-D-UN-ETAT
0010000000000000000000000000000000000LIG: 00163                FIN-D-UN-ETAT
00200093X  00000000 00093X  00000000 LIG: 00164                FIN-D-UN-ETAT
0010000000000000000000000000000000000LIG: 00165                FIN-D-UN-ETAT
00200020020000000 000020020000000 LIG: 00166                   FIN-D-UN-ETAT
00200009X  00000000 00009X  00000000 LIG: 00167                FIN-D-UN-ETAT
00200020020000000 000020020000000 LIG: 00168                   FIN-D-UN-ETAT
0010000000000000000000000000000000000REPORT-BAS-PAGE
0010000000000000000000000000000000000LIG: 00170                REPORT-BAS-PAGE
00200010X  00000000 00010X  00000000 LIG: 00171                REPORT-BAS-PAGE
00200011E0900000000 00011E0900000000 LIG: 00172                REPORT-BAS-PAGE
00200011E0900000000 00011E0900000000 LIG: 00174                REPORT-BAS-PAGE
```

(seules les tailles des données élémentaires ont été calculées, ce sont les seules utiles)

et le fichier des noeuds d'influences :

```
A00252DI00196%
A00253DI00197%
A00254DI00198%
A00255+I00047I00182$I00182%  ┐
A00256+I00079I00184$I00184%  ┘──> influences dues à SUM
A00257DI00199%
A00258DI00200%
A00259MI00125$I00200%  ┐
A00260DI00201%         │
A00261DI00202%         │
A00262MI00108$I00202%  ┤
A00263DI00203%         │
A00264MI00110$I00203%  ┤
A00265DI00204%         ├──> influences dues à SØURCE
A00266MI00109$I00204%  ┤
A00267DI00205%         │
A00268DI00206%         │
A00269DI00207%         │
A00270MI00093$I00207%  ┤
A00271DI00208%         │
A00272MI00182$I00208%  ┤
A00273DI00209%         │
A00274MI00184$I00209%  ┘
```

(3.1.1.2.3.) **Décodage de la procédure division**

La procédure division ne comporte pas de déclaration de données ; son décodage ne génère donc pas (sauf une exception : CØMPUTE) de nouvelles entrées, ni dans la table des identificateurs, ni dans le fichier des mots ; seuls les articles "secondaires" de ce dernier seront complétés par les références des nouveaux noeuds d'influence dus à la procédure.
La procédure division alimente en effet le fichier des noeuds puisqu'elle est composée d'instructions de la forme :

verbe ␣ attributs "fin" (1)

qui au moins, pour un certain nombre de verbes, sont génératrices d'influences.

L'analyse se limite donc, dans une première approche, à la reconnaissance des verbes créateurs d'influences, puis la constitution des noeuds d'influence par la recherche des opérandes et des résultats dans la liste des attributs et leur codification.

Par exemple ADD A B GIVING C où A, B et C ont respectivement les numéros 12, 15 et 13 donne le noeud additif :

n° + 12 15 $ 13 %

Le tableau 1 de l'annexe 4 fournit la liste des différents verbes Cobol, les multiples formats de leurs écritures, les groupes opératoires et les noeuds d'influence correspondants.

Signalons en particulier le verbe CØMPUTE qui nécessite l'utilisation d'un traducteur d'expression arithmétique et qui demande des entrées dans les fichiers et la table pour de nouveaux mots : les mémoires intermédiaires nécessaires au stockage du résultat des opérations élémentaires.

(1) on reconnaît la fin d'une instruction, soit par la présence d'un point, soit par la rencontre du verbe de l'instruction suivante.

De même les opérations "corresponding" sont délicates à traduire puisqu'elles nécessitent l'étude des arborescences correspondant aux structures de la data (qui doivent donc être conservées, par exemple sous forme de matrices d'enchaînement, les noeuds hiérarchiques ne permettant que de remonter l'arborescence des feuilles vers la racine). Elles n'ont pas été traitées.

Exemples :

a) texte source Cobol :

```
                2EME  SECTION.
                DEBUT. INCLUDE TRT1.
                TRUC. MOVE RENS1 TO DDD1. MOVE RENS2 TO DDD2.
                NOTES. NOTE  ON SAUTE  CE  QUI  SUIT
                    ADD  W  TO 7 .
                    SUBTRACT  A  FROM W.
                SUITE. MOVE  JOUR TO  JR  MS  AE .
                    MOVE  JOUR  TO  JR
                    MS  AE.
                    MOVE  ALI  '/'  TO  NAIS.
                    MOVE  ' '  TO NAIS  TP1 TP2
                    .
                SUITE1.
                    WRITE ECR11  FROM LUE.
```

b) table des identificateurs

(fraction nécessaire à la correspondance entre les numéros et les noms)

```
0002? LUE                                          00016

0002? JOUR              LUE                        00020

00011 RENS1             LUE                        00024.00002
00012 RENS2             LUE                        00024.00003
00013 ECRI                                         00025
00014 ECRI1                                        00026

00020 JR                ECRI2                      00031
00021 FILLER            00032                      00032
00022 MS                ECRI2                      00033
00023 FILLER            00034                      00034
00024 AE                ECRI2                      00035
00025 TEMP                                         00037
00026 TP1                                          00039
00027 TP2                                          00054

00106 DDD1              DDD                        00139.00002
00107 DDD2              DDD                        00139.00003
```

c) fichier des noeuds d'influences

```
A00199MI00011%I00106%
A00199MI00012%I00107%
A00200MI00006%I00020I00022I00024%
A00201MI00006%I00020I00022I00024%
A00202ML00001        %I00013I00026I00027%
A00203MI00002%I00014%
```

Notons le libellé '␣' de 1 caractère qui est codé L00001.
Notons aussi l'absence d'influence du verbe de transfert
MØVE dans MØVE ALL '/' TØ NAIS qui signifie garnir toute la zone NAIS
avec le caractère / .

Remarquons surtout que, bien qu'il ne crée pas d'influences, le verbe NØTE
(qui annonce un commentaire) doit être soigneusement étudié lors de l'analyse
du texte source :

- Si note n'est pas le premier verbe d'un paragraphe : les commentaires cessent au premier point suivi d'un blanc
- Sinon ils se poursuivent jusqu'à la fin du paragraphe.
  Il faut donc reconnaître les débuts et les fins de paragraphes, traitement qui à priori, n'était pas justifié.

(3.1.1.3)

Bien que nous n'ayons pu utiliser dans cette version de l'analyseur les principes développés au chapitre 2.3., nous nous sommes efforcés de rendre la programmation aussi modulaire que possible.
Ainsi, bon nombre de "sous-programmes" pourront être utilisés pour l'écriture de la charpente citée page 154; nous nous limiterons ici à une liste explicative de quelques-uns d'entre eux :

ligne-suivante (*) : fournit la ligne suivante après avoir éjecté les anomalies (par exemple, les messages d'erreur pour déclaration incomplète).

mot-suivant (*) : met dans une zone déterminée le mot suivant pris éventuellement sur deux lignes consécutives.

attribue-iden : réalise la partie de codage présentée page 172 :
- attribution d'un numéro et rangement dans la table des identificateurs
- créations de l'article maître
  du noeud de définition
  du premier article secondaire

garni-iden : complète en prenant les informations dans une zone déterminée le contenu de l'article maître (par exemple, lorsqu'on vient de calculer la taille d'une donnée structurée).

maj-iden (ou maj-iden-res) :
ajoute le numéro d'un noeud d'influence signé positivement (ou négativement) à la liste des références des articles secondaires d'un mot.

travail-au-sommet traduit la partie centrale de l'algorithme de la page 175.

graphe-iden sort en fin d'analyse le graphe des influences à partir des tables et fichiers codés (voir section 3.1.3.).

range-op et pose-dollard :
permettent de ranger un à un les opérandes, les résultats et $ dans le noeud d'influence en cours de construction.

prendre-opérande (*) permet, lors du décodage des traitements, de trouver le numéro de l'opérande suivant et/ou sa taille (constantes numériques) etc...

(*) Ces trois sous-programmes fondamentaux demandent un paramètrage dépendant du langage décodé :
Comment reconnaît-on les lignes d'erreurs ? (par exemple ***... message)
Quelle est la définition d'un mot ? (par exemple : tableau des caractères permis et/ou tableau des séparateurs)
Quels sont les types d'opérandes permis (constantes numériques, qualification) ?

troisième partie

chapitre 1

Section 2 :

résultats

(3.1.2)

Le passage de l'analyseur sur le texte Cobol donné à l'annexe 1 et correspondant à l'exemple utilisé dans la deuxième partie donne les résultats suivants :

a) table des identificateurs

TABLE DES IDENTIFICATEURS

référence (numéro du mot)	nom	qualification	numéro de la ligne de définition
00000001	CARTES		00014
00000002	CRTE		00015
00000003	LIGNES		00016
00000004	LGNE		00017
00000005	M1		00019
00000006	M2		00020
00000007	M3		00021
00000008	M4		00022
00000009	M6		00023
00000010	M1		00024
00000011	M2		00025
00000012	X		00026
00000013	P		00027
00000014	Q		00028
00000015	TRV		00029
00000016	I		00030
00000017	S		00031
00000018	CRT1		00032
00000019	D1	CRT_1	00033
00000020	D2	CRT_1	00034
00000021	CRT2		00035
00000022	D3	CRT2	00036
00000023	D4	CRT2	00037
00000024	D5	CRT2	00038
00000025	CRT3		00039
00000026	A	CRT3	00040
00000027	B	CRT3	00041
00000028	C	CRT3	00042
00000029	D	CRT3	00043
00000030	CRT4		00044
00000031	M	CRT4	00045
00000032	N	CRT4	00046
00000033	ED		00047
00000034	ED1	ED	00048
00000035	ED2	ED	00049
00000036	ED3	ED	00050
00000037	ED4	ED	00051
00000038	ED5	ED	00052
00000039	ED6	ED	00053
00000040	TALLY*		00054

(*) TALLY est un identificateur réservé, résultat d'une instruction Cobol (EXAMINE), il peut être manipulé par n'importe quelle autre instruction sans être déclaré (déclaration implicite).

b) fichier des mots :

```
00001000014      00600080FD 00000001 00080FD 00000001 CARTES
00001100004A0000K00004H00041000 5 0006F
00002000015      002C000E0X 00000001 00080X 00000001 CRTF
000021000000000OR
00003000016      00200132FD 00000001 00132FD 00000001 LIGNES
00003100000000000N
00004000017      00300132X 00000001 00132X 00000001 LGNF
000041C000F00000E0007N
00005000019      00300007007000000001 00007007000000001 W1
000051000G0005J0005C
00006000020      003000060060000000001 00006006000000001 W2
00006100000H0005K0005C
00007000021      0030000900700200001 0000900700200001 W3
000071000C10005L0005D
00008000022      00400005005000000001 00005005000000001 W4
000081000 1 00058000 5G0005H
00009000023      00300012E0700200001 00012E0700200001 W6
00009100001A0005M0005E
00010000024      00300004F 00000001S00004F 00000001SM1
000101000190005R0006A
00011000025      00300004F 00000001S00004F 00000001SM2
000111000100006R0006A
00012000026      00300004F 00000001S00004F 00000001SX
0001210001D0006J0006B
00013000027      0060001501300200001 0001501300200001 P
000131000 1E0006P0006H0006R0007J0007C
00014000028      0050001501300200001 0001501300200001 Q
000141000 1F0006P0007B0007A0007B
00015000029      0030001501300200001 0001501300200001 TRV
000151000 1G000600007
00016000030      0050000100100000001 0000100100000001 I
000161000 1H0006I0006M0006E00061
00017000031      0050001801600200001 0001801600200001 S
000171000 1K0004M0007K0007L0007D
00018000032      00300011S 00000001 00011S 00000001 CRT1
000181000 2 0002M0004G
00019000033      00300005005000000001 00005005000000001 D1                CRT1
0001910002A0002P0005A
00020000034      00300006006000000001 00006006000000001 D2                CRT1
00020100002B0002D0005A
00021000035      00300015S 00000001 00015S 00000001 CRT2
000211000 2C0002R0004R
00022000036      0030000500400100001 0000500400100001 D3                  CRT2
0002210002F0002I0005B
00023000037      0040000500500000001 0000500500000001 D4                  CRT2
000231000 2F0002I00058 0005F
00024000038      0030000500500000001 0000500500000001 D5                  CRT2
000241000 2 30002I0005F
00025000039      00300028S 00000001 00028S 00000001 CRT3
0002510002H0003N0005A
00026000040      0030000700500200001 0000700500200001 A                   CRT3
000261000 3 0003F0005I
00027000041      0030000700500200001 0000700500200001 B                   CRT3
000271000 3A0003E0005I
00028000042      0030000700500200001 0000700500200001 C                   CRT3
0002810003B0003E0006
00029000043      0030000700500200001 0000700500200001 D                   CRT3
00029100003C0003E0006
00030000044      00300024S 00000001 00024S 00000001 CRT4
000301000 3D0003R0006B
00031000045      0030001401200200001 0001401200200001 M                   CRT4
00031100003F0003I0006G
[illegible]                                                               CRT4
0003210003G0003I0006G
00033000047      00300090S 00000001 00090S 00000001 ED
0003310003H0004R0007E
00034000048      00300013X 00000001 00013X 00000001 ED1                   ED
000341000 4 0004F0005N
00035000049      00400006L0600000001S00006E0600000001SED2                 ED
00035100004A0004F0005P0005Q
00036000050      00300018E1500200001 00018E1500200001 ED3                 ED
000361000 4B0004F0006K
00037000051      00300017E1400200001 00017E1400200001 ED4                 ED
0003710004C0004F0007L
00038000052      00300018E1800000001 00018E1800000001 ED5                 ED
00038100004D0004F0007M
00039000053      00300018E1500200001 00018E1500200001 ED6                 ED
0003910004E0004F0006K
00040000053      00100004B 00000001S00004B 00000001STALLY
0004010004G...
```


c) fichier des noeuds d'influence :

```
A00001I I00001%
A00002F I00002$I00001%
A00003D I00002%
A00004D I00003%
A00005F I00004$I00003%
A00006D I00004%
A00007D I00005%
A00008D I00006%
A00009D I00007%
A00010D I00008%
A00011D I00009%
A00012D I00010%
A00013D I00011%
A00014D I00012%
A00015D I00013%
A00016D I00014%
A00017D I00015%
A00018D I00016%
A00019D I00017%
A00020D I00018%
A00021D I00019%
A00022D I00020%
A00023D I00021%
A00024H I00020I00019$I00018%
A00025D I00022%
A00026D I00023%
A00027D I00024%
A00028D I00025%
A00029H I00024I00023I00022$I00021%
A00030D I00026%
A00031D I00027%
A00032D I00028%
A00033D I00029%
A00034D I00030%
A00035H I00029I00028I00027I00026$I00025%
A00036D I00031%
A00037D I00032%
A00038D I00033%
A00039H I00032I00031$I00030%
A00040D I00034%
A00041D I00035%
A00042D I00036%
A00043D I00037%
A00044D I00038%
A00045D I00039%
A00046H I00039I00038I00037I00036I00035I00034$I00033%
A00047D I00040%
A00048M I00001$I00018%
A00049M I00001$I00021%
A00050M I00001$I00025%
A00051,I00019I00020$I00005%
A00052+I00022I00023$I00006%
A00053/I00005I00006$I00007%
A00054M I00007$I00009%
A00055M I00009$I00034%
A00056-I00024I00023$I00008%
A00057M I00008$I00035%
A00058-I00008N0000100100 0 $I00035%
A00059+I00026I00027$I00010%
A00060+I00028I00029$I00011%
A00061+I00010I00011$I00012%
A00062M I00012$I00036I00039%
[illegible]
A00064M N00001001000 $I00017%
A00065,N00001001000 I00016$I00016%
A00066M I00001$I00030%
A00067+I00031I00032$I00013I00014%
A00068M I00013$I00015%
A00069M I00016$I00013%
A00070-N00004004000 I00015$I00014%
A00071M I00014$I00013%
A00072,I00014I00017$I00017%
A00073M I00013$I00037%
A00074M I00017$I00038%
A00075M I00033$I00004%
```

La numérotation effectuée par l'analyseur n'étant pas conforme à celle que nous avons arbitrairement donnée sur les figures de l'annexe 1, nous proposons au lecteur les correspondances sur les figures 1, 2 et 3 de l'annexe 5.

Section 3 :

Editions complémentaires

(3.1.3)

L'analyseur a, du texte source Cobol, une connaissance qu'il nous a paru souhaitable d'exploiter pour fournir au programmeur un document d'aide soit à la mise au point, soit à la maintenance. Il s'agit de la "liste des références croisées", inspirée de la "Cross reference" Cobol.

Pour donner à l'utilisateur un moyen de contrôle efficace du processus automatique (qui peut d'ailleurs s'imposer dans certains cas litigieux), nous lui proposons deux documents édités donnant :

- les relations d'influence ;

- le graphe des influences (plus commode, nous le rappelons, pour dérouler le processus de modification de tailles manuellement que le graphe des noeuds d'influence).

Edition des relations d'influences :

Ce document fournit, pour chaque numéro de noeuds d'influence (à l'exception des noeuds de définition), un texte qui exprime "en clair" (dans un langage simplifié) le contenu de chaque groupe opératoire (les mots y sont nommés par les identificateurs Cobol éventuellement qualifiés). Ainsi, à partir de l'exemple d'appui dont le texte Cobol constitue la figure 1 de l'annexe 1, on obtient :

n° ligne	texte explicatif					
00200014	FD CARTES	DATA RECORD CRTE				
00500016	FD LIGNES	DATA RECORD LGNE				
02400035	REGLE HIERARCHIQUE 01 CRT1				02 D2	
02900039	REGLE HIERARCHIQUE 01 CRT2				02 D5	
03500044	REGLE HIERARCHIQUE 01 CRT3				02 D	
03900047	REGLE HIERARCHIQUE 01 CRT4				02 N	
04600054	REGLE HIERARCHIQUE 01 ED				02 ED6	
04800056	LIRE	CARTES	DANS	CRT1		
04900057	LIRE	CARTES	DANS	CRT2		
05000058	LIRE	CARTES	DANS	CRT3		
05100059	D1	D2	SOMMES DANS	W1		
05200060	D3	D4	SOMMES DANS	W2		
05300061	W1	DIVISE PAR	W2	RANGE DANS	W3	
05400061	DEPLACE	W3	DANS	W6		
05500061	DEPLACE	W6	DANS	ED1		
05600062	D5	SOUSTRAIT(S) A	D4	RANGE DANS	W4	
05700063	DEPLACE	W4	DANS	ED2		
05800064	W4	SOUSTRAIT(S) A	N00001001000	RANGE DANS	ED2	
05900065	A	B	SOMMES DANS	M1		
06000066	C	D	SOMMES DANS	M2		
06100066	M1	MULTIPLIE PAR	M2	RANGE DANS	X	
06200067	DEPLACE	X	DANS	ED3	ED6	
06300068	DEPLACE	N00001001000	DANS	I		
06400063	DEPLACE	N00001001000	DANS	S		
06500069	N00001001000	I	SOMMES DANS	I		
06600071	LIRE	CARTES	DANS	CRT4		
06700072	M	N	SOMMES DANS	P	Q	
06800073	DEPLACE	P	DANS	TRV		
06900074	DEPLACE	I	DANS	P		
07000075	N00004004000	SOUSTRAIT(S) A	TRV	RANGE DANS	Q	
07100076	DEPLACE	Q	DANS	P		
07200077	Q	S	SOMMES DANS	S		
07300078	DEPLACE	P	DANS	ED4		
07400079	DEPLACE	S	DANS	ED5		
07500080	ECRIRE	ED	PAR ENR	LGNE		

ou encore sur un exemple mettant en évidence le traitement de la qualification :

texte Cobol

```
01  LUE.
    02  NOM PIC X(30).
    02  PRENOM PIC X(30).

01  ECRI1.
    02  NOM PIC X(30).
    02  PRENOM PIC X(30).


DEBU. MOVE NOM OF LUE TO NOM OF ECRI1. WRITE ECRI1 .
    READ LECT INTO ECRI2
    AT END GO TO FIN.
```

relations correspondantes "en clair" et codées

```
00172 DEPLACE   NOM     DE LUE   DANS    NOM     DE ECRI1
00172 LIRE      LECT    DANS     ECRI2

M00003%00017%
M00001%00015%
```

numéro des mots : extrait de la table des identificateurs

```
00001 LECT                              00015

00003 NOM               LUE             00017
00004 PRENOM            LUE             00018

00015 ECRI2                             00026
00016 ECRI3                             00036
00017 NOM               ECRI1           00027
00018 PRENOM            ECRI1           00028
```

Le graphe des influences :

Cette édition comporte pour chaque mot, la liste des mots successeurs et pour chaque successeur, la liste des numéros des noeuds d'influence qui "donne naissance" à l'axe.

L'extrait ci-dessous du graphe des influences sur l'exemple d'appui permet de remarquer en particulier :
- que X a deux successeurs ED3 et ED6 dus au même noeud 62 ;
- que l'arc (W4, ED2) est dû à deux relations d'influence : 57 et 58.

Cette représentation est à rapprocher du graphique donné en figure 2, annexe 1.

```
                                          **********          CARTES
        SUCCESSEURS:                      NUMEROS DES INFLUENCES:
   CRT1                                          00048
   CRT2                                          00049
   CRT3                                          00050
   CRT4                                          00066
*****************************************************************************
                                          **********          CRTF
        SUCCESSEURS:                      NUMEROS DES INFLUENCES:
   CARTES                                        00002
*****************************************************************************
                                          **********          LGNF
        SUCCESSEURS:                      NUMEROS DES INFLUENCES:
   LIGNES                                        00005
*****************************************************************************
                                          **********          W1
        SUCCESSEURS:                      NUMEROS DES INFLUENCES:
   W3                                            00053
*****************************************************************************
                                          **********          W2
        SUCCESSEURS:                      NUMEROS DES INFLUENCES:
   W3                                            00053
*****************************************************************************
                                          **********          W3
        SUCCESSEURS:                      NUMEROS DES INFLUENCES:
   W6                                            00054
*****************************************************************************
                                          **********          W4
        SUCCESSEURS:                      NUMEROS DES INFLUENCES:
   ED2                ED                         00057  00058
*****************************************************************************
                                          **********          W6
        SUCCESSEURS:                      NUMEROS DES INFLUENCES:
   ED1                ED                         00055
*****************************************************************************
                                          **********          X1
        SUCCESSEURS:                      NUMEROS DES INFLUENCES:
   X                                             00061
*****************************************************************************
                                          **********          X2
        SUCCESSEURS:                      NUMEROS DES INFLUENCES:
   X                                             00061
*****************************************************************************
                                          **********          X
        SUCCESSEURS:                      NUMEROS DES INFLUENCES:
   ED3                ED                         00062
   ED6                ED                         00062
*****************************************************************************
                                          **********          P
        SUCCESSEURS:                      NUMEROS DES INFLUENCES:
   TRV                                           00068
   ED4                ED                         00073
*****************************************************************************
```


Liste des références croisées :

Elle comporte en fait trois sous-listes portant sur :

- les identificateurs
- les identificateurs d'étiquettes de paragraphe
- les identificateurs d'étiquettes de section.

Chaque ligne d'une sous-liste précise pour l'identificateur concerné (éventuellement complété par sa qualification) : le numéro de ligne de définition et la liste des numéros de lignes où l'identificateur apparaît.

Ainsi, sur l'exemple d'appui, on obtient :

```
FERENCES CROISEES ***

TIFICATEURS

      A                CRT3        00065
      B                CRT3        00065
      C                CRT3        00065
      CARTES                       00055   00056   00057   00058   00071
                                   00082
      CRTF                         00014
      CRT1                         00056
      CRT2                         00057
      CRT3                         00058
      CRT4                         00071
      D                CRT3        00065
      D1               CRT1        00059
      D2               CRT1        00059
      D3               CRT2        00059
      D4               CRT2        00059   00062
      D5               CRT2        00062
      ED               ED          00080
      ED1              ED          00061
      ED2              ED          00063   00064
      ED3              ED          00067
      ED4              ED          00079
      ED5              ED          00079
      ED6              ED          00067
      I                            00068   00069   00070   00074
      LGNE                         00016   00080
      LIGNES                       00055   00082
      M                CRT4        00072
      M1                           00065   00066
      M2                           00065   00066
      N                CRT4        00072
      P                            00072   00073   00074   00076   00079
      Q                            00072   00075   00076   00077
      S                            00068   00077   00079
      TRV                          00073   00074   00075   00076
      V1                           00059   00060
      W2                           00059   00060
      W3                           00060   00061   00063
      W4                           00062   00063   00064
      W6                           00061   00061
      X                            00066   00067

FERENCES CROISEES ***

ETTES DE PARAGRAPHES

      BOUCLE                       00077
      DEBUT
      ECRITURE                     00070
      FIN                          00054   00057   00058   00071
      LECTURES                     00081

ligne                              nom     qualification     n° ligne de références
tion
```

Et pour un exemple comportant des sections :

```
*** REFERENCES CROISEES ***

ETIQUETTES DE PARAGRAPHES

00140          BBU             1ERE
00139          DEBUT           1ERE          00150
00153          DEBUT           2EME
00203*00002    DEBUT           3EME
00151          FIN             1ERE          00139     00142     00145     00148
00154          NOTES           2EME
00146          SUITE           1ERE
00157          SUITE           2EME
00163          SUITE1          2EME
00171          SUITE2          2EME
```

```
*** REFERENCES CROISEES ***

ETIQUETTES DE SECTIONS

00138          1ERE
00152          2EME                          00149
00203          3EME
```


Chapitre 2 :

Les opérateurs de cheminements

Ce chapitre a pour objectif l'étude du fonctionnement des opérateurs de cheminement. Rappelons que leur fondement théorique a fait l'objet de la deuxième partie de ce travail.

troisième partie

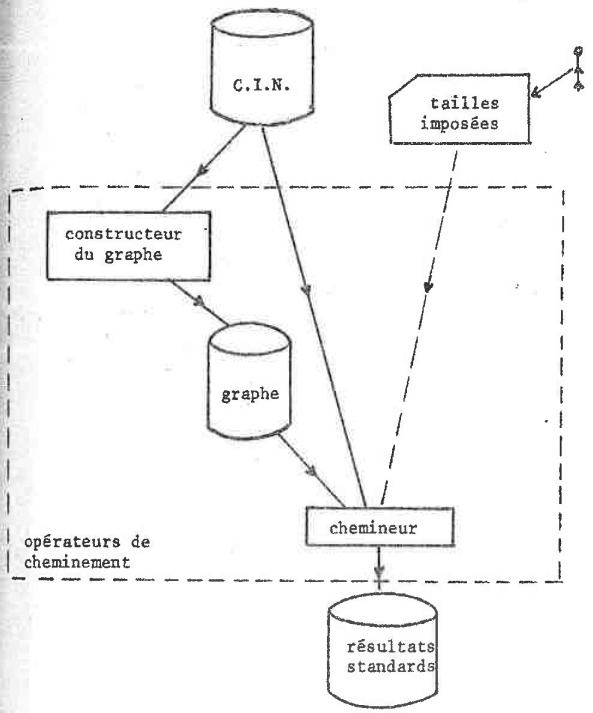
chapitre 2

Section 1 :

utilisation

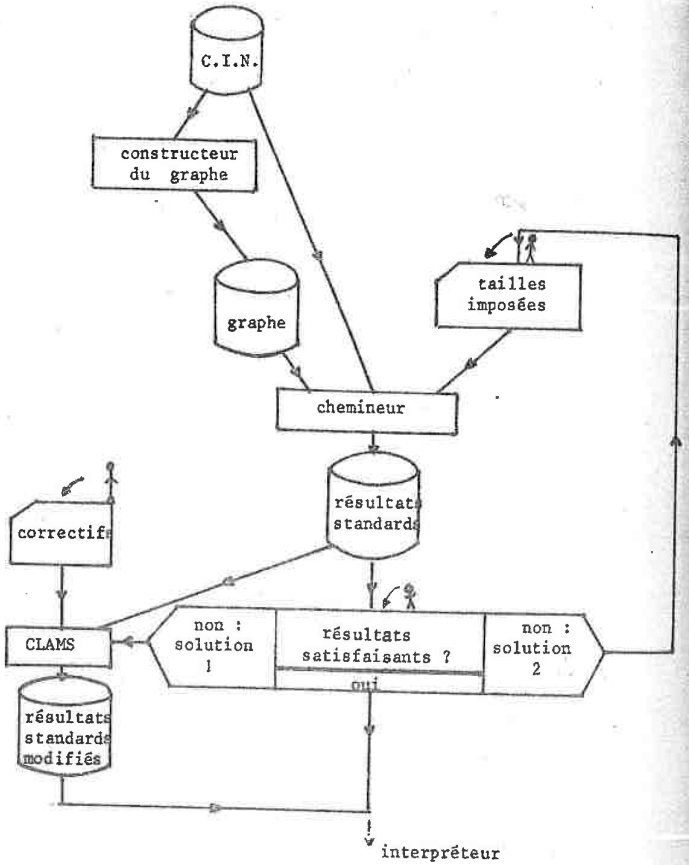
(3. 2. 1)

Nous avons déjà proposé une vue synoptique de l'enchaînement des composantes des opérateurs de cheminement sous la forme :



Remarque :
Les tailles imposées sont une entrée facultative du chemineur ; elles correspondent aux mots initialement modifiés déclenchant le processus de modifications en cascade et au choix de tailles de cumuls ou de variables au sens strict pour l'évaluation de tailles inconnues.

Le fonctionnement des opérateurs est en fait plus complexe puisque nous souhaitons donner à l'utilisateur la possibilité d'infléchir les décisions automatiques dans le sens où il le veut :



Nous prévoyons deux possibilités :

1 - ou bien le choix impératif par le responsable des tailles de certains "mots", ce qui l'oblige à assurer les corrections correspondantes dans le fichier des mots. Il pourra pour ce faire utiliser un programme standard de mise à jour des fichiers (CLAMS).

2 - ou bien de nouveaux essais avec de nouvelles valeurs de tailles imposées.

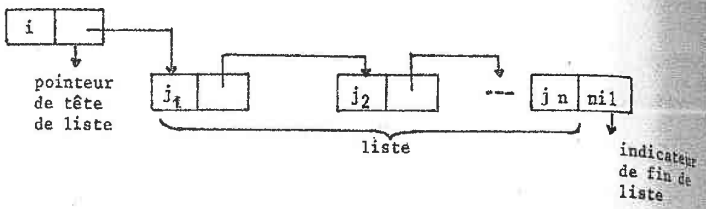
(3.2.2.1.1.) **Représentation du graphe en mémoire**

Parmi les représentations usuelles des graphes quelconques, outre les représentations graphiques, on trouve chez les auteurs (BERGE, ROY, KAUFMANN...) :

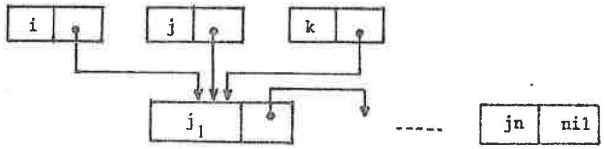
- la matrice booléenne ou associée A (n, n) (n = ordre du graphe) où A (i, j) = 1 s'il existe un arc de i à j et 0 dans le cas contraire.
- la matrice d'incidence B (n, m) (n = ordre du graphe ou nombre de points, m = nombre d'arcs) où B (i,j) vaut 1 si le point i est origine de l'arc j, - 1 si le point i est extrêmité de l'arc j et 0 dans les autres cas.
- le dictionnaire du graphe (ROY, 1969) ou listes des successeurs de chaque point.

La matrice associée permet un accès direct à toutes les informations, elle convient pour tous les traitements. Mais son encombrement croît avec n^2 quel que soit le nombre d'arcs m.
La matrice d'incidence ne permet pas de représenter les boucles, elle est encombrante (la dimension est n x m) alors que seule deux éléments dans chaque colonne de longueur n sont non nuls. Elle se prête bien à la formulation matricielle de certains problèmes (par exemple les problèmes de flot : FORD et FULKERSON, 1967).
Le dictionnaire du graphe privilégie certains types d'accès : on obtient par exemple très facilement les successeurs d'un point les uns après les autres.
Son encombrement, par exemple, est n + 2 x m (un pointeur de liste pour chaque point et deux encombrements élémentaires pour chaque extrêmité d'arc) pour une représentation chaînée des listes.

C'est cette dernière représentation que nous avons choisie puisqu'elle correspond à l'accès dont nous avons besoin et est économique en espace mémoire. Il s'agit de listes de successeurs avec une implémentation chaînée ; ainsi, si i a pour successeurs j_1, j_1, ..., j_n, on aura :



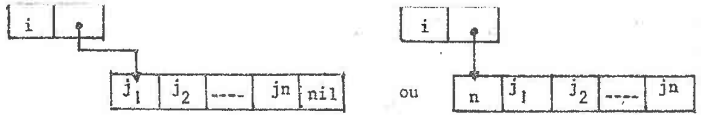
En remarquant que le graphe est invariant pendant son utilisation, on peut optimiser l'emplacement mémoire utilisé : si deux points possèdent les mêmes successeurs, on leur associe la même liste :



Ce cas n'est pas rare pour le graphe des noeuds d'influence où, en particulier, deux noeuds peuvent avoir les mêmes résultats et donc les mêmes successeurs.

En fait, nous n'avons programmé qu'un cas particulier de cette optimisation pour les noeuds ayant un seul et même résultat.

Remarque :
Comme ces listes sont invariantes après la construction du graphe, on pourrait utiliser l'implémentation chaînée pendant la construction, puis réorganiser les listes en implantation consécutive :



On gagnerait ainsi m - 1 places tout en facilitant les accès.

(3.2.2.1.2.) **Construction du graphe**

Le fichier des noeuds d'influence construit par l'analyseur donne toutes les informations nécessaires à la construction du graphe. Ainsi l'algorithme de construction le plus immédiat, sans optimisation de l'emplacement mémoire utilisé notamment, serait :

```
pour chaque noeud n :
    pour chaque noeud m :
        pour chaque résultat i de n :
            tant qu'il existe un opérande j de m non examiné
            et que m n'a pas encore été reconnu successeur de m :
                si i = j
                alors noter dans une liste associée à n que m est successeur
                finsi
            fintantque
        fin pour chaque
    fin pour chaque
fin pour chaque.
```

Mais la préparation du travail que nous avons faite en alimentant, par l'analyseur, les articles secondaires du fichier des mots avec les références positives ou négatives des noeuds où les mots sont opérandes ou résultats, permet de construire le graphe ou son inverse beaucoup plus rapidement suivant l'algorithme (nous y avons inséré un test qui permet en outre d'optimiser la place : à deux noeuds ayant le même résultat, on associe la même liste de successeurs) :

pour le graphe direct :

(3.2.2.1.2.) Construction du graphe

Le fichier des noeuds d'influence construit par l'analyseur donne toutes les informations nécessaires à la construction du graphe. Ainsi l'algorithme de construction le plus immédiat, sans optimisation de l'emplacement mémoire utilisé notamment, serait :

```
pour chaque noeud n :
    pour chaque noeud m :
        pour chaque résultat i de n :
            tant qu'il existe un opérande j de m non examiné
            et que m n'a pas encore été reconnu successeur de m :
                si i = j
                alors noter dans une liste associée à n que m est successeur
                finsi
            fintantque
        fin pour chaque
    fin pour chaque
fin pour chaque.
```

Mais la préparation du travail que nous avons faite en alimentant, par l'analyseur, les articles secondaires du fichier des mots avec les références positives ou négatives des noeuds où les mots sont opérandes ou résultats, permet de construire le graphe ou son inverse beaucoup plus rapidement suivant l'algorithme (nous y avons inséré un test qui permet en outre d'optimiser la place : à deux noeuds ayant le même résultat, on associe la même liste de successeurs) :

pour le graphe direct :


```
pour chaque noeud n :
    si n n'a qu'un résultat et
    si ce résultat est associé à un noeud de définition déjà examiné (pointeur
       dans l'article maître)
    alors attribuer à n la même liste que pour le noeud de définition du résulta
    sinon
        pour chaque résultat i de n :
            pour chaque élément positif (sauf le premier) des
            articles secondaires de clé "i|chiffre" :
                noter cet élément dans une liste associée à n
            fin pour chaque
        fin pour chaque
    finsi
fin pour chaque.
```

et pour le graphe pseudo-inverse :

```
pour chaque noeud n :
        si n n'a qu'un opérande et
        si cet opérande est associé à un pseudo-noeud de définition déjà
        examiné
        alors attribuer à n la même liste que pour le pseudo-noeud de
             définition de l'opérande
        sinon
             pour chaque opérande j de n :
                 pour chaque élément négatif des articles secondaires
                 de clé "j|chiffre" :
                     noter cet élément dans une liste associée à n
                 fin pour chaque
             fin pour chaque
        finsi
fin pour chaque .
```

On constate que les deux algorithmes se déduisent l'un de l'autre en échangeant les notions :

opérande ⟷ résultat
négatif ⟷ positif

Le constructeur de graphe est donc un seul programme qui construit en fonction de la valeur d'un booléen soit le graphe direct, soit le graphe pseudo-inverse.

```
pour chaque noeud n :
    si n n'a qu'un résultat et
    si ce résultat est associé à un noeud de définition déjà examiné (pointeurs
       dans l'article maître)
    alors attribuer à n la même liste que pour le noeud de définition du résulta
    sinon
        pour chaque résultat i de n :
                pour chaque élément positif (sauf le premier) des
                articles secondaires de clé "i|chiffre" :
                        noter cet élément dans une liste associée à n
                fin pour chaque
        fin pour chaque
    finsi
fin pour chaque.
```

et pour le graphe pseudo-inverse :

```
pour chaque noeud n :
        si n n'a qu'un opérande et
        si cet opérande est associé à un pseudo-noeud de définition déjà
        examiné
        alors attribuer à n la même liste que pour le pseudo-noeud de
             définition de l'opérande
        sinon
             pour chaque opérande j de n :
                  pour chaque élément négatif des articles secondaires
                  de clé "j|chiffre" :
                        noter cet élément dans une liste associée à n
                  fin pour chaque
             fin pour chaque
        finsi
fin pour chaque .
```

On constate que les deux algorithmes se déduisent l'un de l'autre en échangeant les notions :

opérande ⟷ résultat

négatif ⟷ positif

Le constructeur de graphe est donc un seul programme qui construit en fonction de la valeur d'un booléen soit le graphe direct, soit le graphe pseudo-inverse.

(3.2.2.1.3.) **Résultats**

Il s'agit ici d'une édition "in extenso" à partir du contenu du fichier magnétique représentant le graphe ; ceci explique que l'édition reste "technique".

On y voit apparaître pour chaque noeud ayant des successeurs :

- l'adresse relative (dans la zone chaînée contenant le graphe) de la tête de la liste (TETE)
- l'adresse relative (s'il y a plus d'un successeur) de la queue de la liste (QUEUE ≠ TETE)
- pour chaque successeur :

    + son numéro

    (+ le contenu du pointeur de liste)

On peut remarquer :

- que deux noeuds ayant le même résultat "pointent" sur la même liste ; par exemple : 1 et 2, 15 et 69, 18 et 63, etc....
- que deux noeuds n'ayant pas le même résultat ne "pointent" pas sur la même liste, même s'ils ont les mêmes successeurs ; par exemple : 40, 41, 42, 43, 44 et 45.

```
SUCC. 048 00002+
SUCC. 049 00003+
SUCC. 050 00004+
SUCC. 066 00001-
NBEUD N8 +0000000002 TETE +0000000001 QUEUE +0000000004
SUCC. 018 00002+
SUCC. 049 00003+
SUCC. 050 00004+
SUCC. 066 00001-
NBEUD N8: +0000000003 UN SUCC. 002 TETE=QUEUE +0000000005
NBEUD N8: +0000000006 UN SUCC. 005 TETE=QUEUE +0000000006
NBEUD N8: +0000000007 UN SUCC. 053 TETE=QUEUE +0000000007
NBEUD N8: +0000000008 UN SUCC. 053 TETE=QUEUE +0000000008
NBEUD N8: +0000000009 UN SUCC. 054 TETE=QUEUE +0000000009
NBEUD N8 +0000000010 TETE +0000000010 QUEUE +0000000011
SUCC. 057 00011+
SUCC. 058 00001-
NBEUD N8: +0000000011 UN SUCC. 055 TETE=QUEUE +0000000012
NBEUD N8: +0000000012 UN SUCC. 061 TETE=QUEUE +0000000013
NBEUD N8: +0000000013 UN SUCC. 061 TETE=QUEUE +0000000014
NBEUD N8: +0000000014 UN SUCC. 062 TETE=QUEUE +0000000015
NBEUD N8 +0000000015 TETE +0000000016 QUEUE +0000000017
SUCC. 068 00017+
SUCC. 073 00001-
NBEUD N8 +0000000016 TETE +0000000018 QUEUE +0000000019
SUCC. 071 00019+
SUCC. 072 00001-
NBEUD N8: +0000000017 UN SUCC. 070 TETE=QUEUE +0000000020
NBEUD N8 +0000000018 TETE +0000000021 QUEUE +0000000022
SUCC. 065 00022+
SUCC. 069 00001-
NBEUD N8 +0000000019 TETE +0000000023 QUEUE +0000000024
SUCC. 072 00024+
SUCC. 074 00001-
NBEUD N8 +0000000021 TETE +0000000025 QUEUE +0000000026
SUCC. 024 00026+
SUCC. 051 00001-
NBEUD N8 +0000000022 TETE +0000000027 QUEUE +0000000028
SUCC. 024 00028+
SUCC. 051 00001-
NBEUD N8 +0000000025 TETE +0000000029 QUEUE +0000000030
SUCC. 029 00030+
SUCC. 052 00001-
NBEUD N8 +0000000026 TETE +0000000031 QUEUE +0000000033
SUCC. 029 00032+
SUCC. 052 00033+
SUCC. 056 00001-
NBEUD N8 +0000000027 TETE +0000000034 QUEUE +0000000035
SUCC. 029 00035+
SUCC. 046 00001-
NBEUD N8 +0000000030 TETE +0000000036 QUEUE +0000000037
SUCC. 035 00037+
SUCC. 059 00001-
NBEUD N8 +0000000031 TETE +0000000038 QUEUE +0000000039
SUCC. 035 00039+
SUCC. 059 00001-
NBEUD N8 +0000000032 TETE +0000000040 QUEUE +0000000041
SUCC. 035 00041+
SUCC. 060 00001-
NBEUD N8 +0000000033 TETE +0000000042 QUEUE +0000000043
SUCC. 035 00043+
SUCC. 060 00001-
NBEUD N8 +0000000036 TETE +0000000044 QUEUE +0000000045
```



```
SUCC. 067 00001-
NBEUD N8 +0000000037 TETE +0000000046 QUEUE +0000000047
SUCC. 039 00047+
SUCC. 067 00001-
NBEUD N8: +0000000033 UN SUCC. 075 TETE=QUEUE +0000000048
NBEUD N8: +0000000040 UN SUCC. 046 TETE=QUEUE +0000000049
NBEUD N8: +0000000041 UN SUCC. 046 TETE=QUEUE +0000000050
NBEUD N8: +0000000042 UN SUCC. 046 TETE=QUEUE +0000000051
NBEUD N8: +0000000043 UN SUCC. 046 TETE=QUEUE +0000000052
NBEUD N8: +0000000044 UN SUCC. 046 TETE=QUEUE +0000000053
NBEUD N8: +0000000045 UN SUCC. 046 TETE=QUEUE +0000000054
NBEUD N8: +0000000046 UN SUCC. 075 TETE=QUEUE +0000000048
NBEUD N8: +0000000051 UN SUCC. 053 TETE=QUEUE +0000000007
NBEUD N8: +0000000052 UN SUCC. 053 TETE=QUEUE +0000000008
NBEUD N8: +0000000053 UN SUCC. 054 TETE=QUEUE +0000000009
NBEUD N8: +0000000054 UN SUCC. 055 TETE=QUEUE +0000000012
NBEUD N8: +0000000055 UN SUCC. 046 TETE=QUEUE +0000000049
NBEUD N8 +0000000056 TETE +0000000010 QUEUE +0000000011
SUCC. 057 00011+
SUCC. 058 00001-
NBEUD N8: +0000000057 UN SUCC. 046 TETE=QUEUE +0000000050
NBEUD N8: +0000000058 UN SUCC. 046 TETE=QUEUE +0000000050
NBEUD N8: +0000000059 UN SUCC. 061 TETE=QUEUE +0000000013
NBEUD N8: +0000000060 UN SUCC. 061 TETE=QUEUE +0000000014
NBEUD N8: +0000000061 UN SUCC. 062 TETE=QUEUE +0000000015
NBEUD N8: +0000000062 UN SUCC. 046 TETE=QUEUE +0000000055
NBEUD N8 +0000000063 TETE +0000000021 QUEUE +0000000022
SUCC. 065 00022+
SUCC. 069 00001-
NBEUD N8 +0000000064 TETE +0000000023 QUEUE +0000000024
SUCC. 072 00024+
SUCC. 074 00001-
NBEUD N8 +0000000065 TETE +0000000021 QUEUE +0000000022
SUCC. 065 00022+
SUCC. 069 00001-
NBEUD N8 +0000000067 TETE +0000000056 QUEUE +0000000059
SUCC. 058 00057+
SUCC. 073 00058+
SUCC. 071 00059+
SUCC. 072 00001-
NBEUD N8: +0000000068 UN SUCC. 070 TETE=QUEUE +0000000020
NBEUD N8 +0000000069 TETE +0000000016 QUEUE +0000000017
SUCC. 068 00017+
SUCC. 073 00001-
NBEUD N8 +0000000070 TETE +0000000018 QUEUE +0000000019
SUCC. 071 00019+
SUCC. 072 00001-
NBEUD N8 +0000000071 TETE +0000000016 QUEUE +0000000017
SUCC. 068 00017+
SUCC. 073 00001-
NBEUD N8 +0000000072 TETE +0000000023 QUEUE +0000000024
SUCC. 072 00024+
SUCC. 074 00001-
NBEUD N8: +0000000073 UN SUCC. 046 TETE=QUEUE +0000000052
NBEUD N8: +0000000074 UN SUCC. 046 TETE=QUEUE +0000000053
NBEUD N8: +0000000075 UN SUCC. 005 TETE=QUEUE +0000000006
```

troisième partie
chapitre 2

Section 2
Analyse et résultats

paragraphe 2
chemineur

(3. 2. 2. 2)

La programmation du chemineur correspond à l'étude faite dans la deuxième partie et, en particulier, à l'énoncé de l'algorithme donné au chapitre 2.1.

Rappelons toutefois qu'au lieu d'être générés, les sous-programmes de définition de calcul des tailles utilisées par l'algorithme ont été, pour les essais cités, programmés directement.

Nous donnerons du fonctionnement du chemineur 2 types de résultats :

- le premier permet de vérifier le seul déroulement de la pile ;
- les deux suivants se rapportent à des exemples complets de modifications en cascade.

Ces résultats correspondent au traitement par les opérateurs de cheminement de l'exemple d'appui ; aussi peuvent-ils être contrôlés à partir des tableaux donnés dans la 2ème partie et en annexe 2.
Comme précédemment pour le graphe, ces résultats n'ayant pas été traités par l'interpréteur sont édités sous la forme d'un "suivi d'exécution" qui n'est évidemment pas la forme définitive à l'intention des utilisateurs.

(3.2.2.2.1.) **Cheminements**

Cette édition permet de tester le noyau de cheminement indépendamment des règles.
Elle fait apparaître les différentes itérations et mouvements de la pile
(on peut y remarquer un circuit de 3 points imprimés 2 fois car il a 2 fermetures).
Pour l'obtenir, il a suffi de remplacer l'exécution des règles par le positionnement à "vrai" du booléen mod (RESULMØD = Ø).


```
...ER: 000030 MODIFICATION 0000600000000001
...F DU CHMNT: +0000000026
...DE: +0000000026 ; 029
...0 = 0
...PILE
...S SUCC. DE: +0000000029
...0029 DEPILE
...DE: +0000000026 ; 052
...0 = 0
...PILE
...DE: +0000000052 ; 053
...0 = 0
...PILE
...DE: +0000000053 ; 054
...0 = 0
...PILE
...DE: +0000000054 ; 055
...0 = 0
...PILE
...DE: +0000000055 ; 046
...0 = 0
...PILE
...DE: +0000000046 ; 075
...0 = 0
...PILE
...DE: +0000000075 ; 005
...0 = 0
...PILE
...S SUCC. DE: +0000000005
...0005 DEPILE
...S SUCC. DE: +0000000075
...0075 DEPILE
...S SUCC. DE: +0000000046
...0046 DEPILE
...S SUCC. DE: +0000000055
...0055 DEPILE
...S SUCC. DE: +0000000054
...0054 DEPILE
...S SUCC. DE: +0000000053
...0053 DEPILE
...S SUCC. DE: +0000000052
...0052 DEPILE
...DE: +0000000026 ; 056
...0 = 0
...PILE
...DE: +0000000056 ; 057
...0 = 0
...PILE
...DE: +0000000057 ; 046
...0 = 0
...FERMETURE DE DOUBLET OU DE CIRCUIT
...S SUCC. DE: +0000000057
...0057 DEPILE
...DE: +0000000056 ; 058
```

```
...ESULMOD = 0
* 058 EMPILE
* SUCC. DE: +0000000058 ; 046
RESULMOD = 0
* FIN DES SUCC. DE: +0000000058
* +0000000058 DEPILE
* FIN DES SUCC. DE: +0000000056
* +0000000056 DEPILE
* FIN DES SUCC. DE: +0000000026
* +0000000026 DEPILE
***** FIN ITER
*** ORIGINE DU CHMNT: +0000000046
* SUCC. DE: +0000000046 ; 075
RESULMOD = 0
* 075 EMPILE
* SUCC. DE: +0000000075 ; 005
RESULMOD = 0
* 005 EMPILE
* FIN DES SUCC. DE: +0000000005
* +0000000005 DEPILE
* FIN DES SUCC. DE: +0000000075
* +0000000075 DEPILE
* FIN DES SUCC. DE: +0000000046
* +0000000046 DEPILE
***** FIN ITER
***** FIN D UNE MODIF
IDEN. NUMERO: 000310 MODIFICATION 0001701300400001
*** ORIGINE DU CHMNT: +0000000036
* SUCC. DE: +0000000036 ; 039
RESULMOD = 0
* 039 EMPILE
* FIN DES SUCC. DE: +0000000039
* +0000000039 DEPILE
* SUCC. DE: +0000000076 ; 067
RESULMOD = 0
* 067 EMPILE
* SUCC. DE: +0000000067 ; 068
RESULMOD = 0
* 068 EMPILE
* SUCC. DE: +0000000068 ; 070
RESULMOD = 0
* 070 EMPILE
* SUCC. DE: +0000000070 ; 071
RESULMOD = 0
* 071 EMPILE
* SUCC. DE: +0000000071 ; 068
RESULMOD = 0
* 068 FERMETURE DE DOUBLET OU DE CIRCUIT
* SUCC. DE: +0000000071 ; 073
RESULMOD = 0
* 073 EMPILE
* SUCC. DE: +0000000073 ; 046
RESULMOD = 0
* 046 EMPILE
* SUCC. DE: +0000000046 ; 075
RESULMOD = 0
* 075 EMPILE
* SUCC. DE: +0000000075 ; 005
RESULMOD = 0
* 005 EMPILE
* FIN DES SUCC. DE: +0000000005
* +0000000005 DEPILE
* FIN DES SUCC. DE: +0000000075
* +0000000075 DEPILE
* FIN DES SUCC. DE: +0000000046
```

```
* FIN DES SUCC. DE: +0000000074
* +0000000073 DEPILE
* FIN DES SUCC. DE: +0000000071
* +0000000071 DEPILE
* SUCC. DE: +0000000070 ; 072
RESULMAD = 0
* 072 EMPILE
* SUCC. DE: +0000000072 ; 074
RESULMAD = 0
* 074 EMPILE
* SUCC. DE: +0000000074 ; 046
RESULMAD = 0
* 046 FERMETURE DE DOUBLET OU DE CIRCUIT
* FIN DES SUCC. DE: +0000000074
* +0000000074 DEPILE
* FIN DES SUCC. DE: +0000000072
* +0000000072 DEPILE
* FIN DES SUCC. DE: +0000000070
* +0000000070 DEPILE
* FIN DES SUCC. DE: +000000006R
* +0000000068 DEPILE
* SUCC. DE: +0000000067 ; 073
RESULMAD = 0
* 073 FERMETURE DE DOUBLET OU DE CIRCUIT
* SUCC. DE: +0000000067 ; 071
RESULMAD = 0
* 071 FERMETURE DE DOUBLET OU DE CIRCUIT
* SUCC. DE: +0000000067 ; 072
RESULMAD = 0
* 072 FERMETURE DE DOUBLET OU DE CIRCUIT
* FIN DES SUCC. DE: +0000000067
* +0000000067 DEPILE
* FIN DES SUCC. DE: +0000000036
* +0000000036 DEPILE
***** FIN ITER
*** ORIGINE DU CHAMT: +000000006R
* SUCC. DE: +0000000068 ; 070
RESULMAD = 0
* 070 EMPILE
* SUCC. DE: +0000000070 ; 071
RESULMAD = 0
* 071 EMPILE
* SUCC. DE: +0000000071 ; 068
RESULMAD = 0
***** CIRCUIT:
00068+
00070+
00071+
***** FIN CIRCUIT
* SUCC. DE: +0000000071 ; 073
RESULMAD = 0
* 073 EMPILE
* SUCC. DE: +0000000073 ; 046
RESULMAD = 0
* 046 EMPILE
* SUCC. DE: +0000000046 ; 075
RESULMAD = 0
* 075 EMPILE
* SUCC. DE: +0000000075 ; 005
RESULMAD = 0
* 005 EMPILE
* FIN DES SUCC. DE: +0000000005
* +0000000005 DEPILE
* FIN DES SUCC. DE: +0000000075
* +0000000075 DEPILE
* FIN DES SUCC. DE: +0000000046
```



```
* +0000000046 DEPILE
* FIN DES SUCC. DE: +0000000073
* +0000000073 DEPILE
* FIN DES SUCC. DE: +0000000071
* +0000000071 DEPILE
* SUCC. DE: +0000000070 ; 072
RESULMAD = 0
* 072 EMPILE
* SUCC. DE: +0000000072 ; 074
RESULMAD = 0
* 074 EMPILE
* SUCC. DE: +0000000074 ; 046
RESULMAD = 0
* 046 FERMETURE DE DOUBLET OU DE CIRCUIT
* FIN DES SUCC. DE: +0000000074
* +0000000074 DEPILE
* FIN DES SUCC. DE: +0000000072
* +0000000072 DEPILE
* FIN DES SUCC. DE: +0000000070
* +0000000070 DEPILE
* FIN DES SUCC. DE: +000000006R
* +0000000068 DEPILE
***** FIN ITER
*** ORIGINE DU CHAMT: +0000000046
* SUCC. DE: +0000000046 ; 075
RESULMAD = 0
* 075 EMPILE
* SUCC. DE: +0000000075 ; 005
RESULMAD = 0
* 005 EMPILE
* FIN DES SUCC. DE: +0000000005
* +0000000005 DEPILE
* FIN DES SUCC. DE: +0000000075
* +0000000075 DEPILE
* FIN DES SUCC. DE: +0000000046
* +0000000046 DEPILE
***** FIN ITER
*** ORIGINE DU CHAMT: +0000000073
* SUCC. DE: +0000000073 ; 046
RESULMAD = 0
* 046 EMPILE
* SUCC. DE: +0000000046 ; 075
RESULMAD = 0
* 075 EMPILE
* SUCC. DE: +0000000075 ; 005
RESULMAD = 0
* 005 EMPILE
* FIN DES SUCC. DE: +0000000005
* +0000000005 DEPILE
* FIN DES SUCC. DE: +0000000075
* +0000000075 DEPILE
* FIN DES SUCC. DE: +0000000046
* +0000000046 DEPILE
* FIN DES SUCC. DE: +0000000073
* +0000000073 DEPILE
***** FIN ITER
*** ORIGINE DU CHAMT: +0000000071
* SUCC. DE: +0000000071 ; 068
RESULMAD = 0
* 068 EMPILE
* SUCC. DE: +0000000068 ; 070
RESULMAD = 0
* 070 EMPILE
* SUCC. DE: +0000000070 ; 071
RESULMAD = 0
***** CIRCUIT:
```

```
00071+
00068+
00070+
***** FIN CIRCUIT
* SUCC. DE: +0000000070 ; 072
RESULMAD = 0
* 072 EMPILE
* SUCC. DE: +0000000072 ; 074
RESULMAD = 0
* 074 EMPILE
* SUCC. DE: +0000000074 ; 046
RESULMAD = 0
* 046 EMPILE
* SUCC. DE: +0000000046 ; 075
RESULMAD = 0
* 075 EMPILE
* SUCC. DE: +0000000075 ; 005
RESULMAD = 0
* 005 EMPILE
* FIN DES SUCC. DE: +0000000005
* +0000000005 DEPILE
* FIN DES SUCC. DE: +0000000075
* +0000000075 DEPILE
* FIN DES SUCC. DE: +0000000046
* +0000000046 DEPILE
* FIN DES SUCC. DE: +0000000074
* +0000000074 DEPILE
* FIN DES SUCC. DE: +0000000072
* +0000000072 DEPILE
***** FIN ITER
***** FIN D UNE MODIF
```

(3.2.2.2.2.) **Modifications en cascade**

On retrouve sur les deux exemples suivants le déroulement de la pile précédente complété par l'analyse des calculs de tailles en chaque noeud.

Les principaux éléments de cette analyse sont :

- le texte du noeud d'influence codé
- le numéro et les tailles de chaque opérande
- le séparateur @
- le numéro et les tailles de chaque résultat après exécution des règles.

On peut constater en particulier que l'on n'édite plus le circuit : le cheminement s'arrêtant chaque fois qu'il n'y a pas de modifications à prépager (RESULMOD = N).

Pour compléter ces éditions, nous donnons également le nouvel état du fichier des mots.


```
IDEN. NUMERO: 000230 MODIFICATION 0000600000000001
*** ORIGINE DU CHANT: +0000020026
* SUCC. DE: +0000000026 ; 029
REGLE 00029 CODE + TEXTE I00024I000231000228I00021%
(BP.AU RES.) 00024
TAILLES OP.: 0000500500000001 0000500500000001
(BP.AU RES.) 00028
TAILLES OP.: 0000500500000001 0000600000000001
(BP.AU RES.) 00022
TAILLES OP.: 0000500400100001 0000500400100001
(BP.AU RES.) 0
(BP.AU RES.) 00021
RES. NO: 000210 TAILLES: 00015S 00000001 00016S 00000001
RESULMOD = 0
* 029 EMPILE
* FIN DES SUCC. DE: +0000000029
* +0000000028 DEPILE
* SUCC. DE: +0000000026 ; 052
REGLE 00052 CODE + TEXTE I00022I00023%I00006%
(BP.AU RES.) 00022
TAILLES OP.: 0000500400100001 0000500400100001
(BP.AU RES.) 00023
TAILLES OP.: 0000500500000001 0000600000000001
(BP.AU RES.) 0
(BP.AU RES.) 00006
RES.NO: 000060 TAILLES: 0000600600000001 0000600600000001
(BP.AU RES.) 0
RESULMOD = N
* SUCC. DE: +0000000026 ; 056
REGLE 00056 CODE - TEXTE I00024I00023%I00008%
(BP.AU RES.) 00024
TAILLES OP.: 0000500500000001 0000500500000001
(BP.AU RES.) 00023
TAILLES OP.: 0000600500000001 0000600000000001
(BP.AU RES.) 0
(BP.AU RES.) 00008
RES.NO: 000080 TAILLES: 0000500500000001 0000500500000001
(BP.AU RES.) 0
RESULMOD = N
* FIN DES SUCC. DE: +0000000026
* +0000000026 DEPILE
***** FIN ITER
***** FIN D UNE MODIF
IDEN. NUMERO: 000310 MODIFICATION 0001701300400001
*** ORIGINE DU CHANT: +0000030036
* SUCC. DE: +0000000036 ; 039
REGLE 00039 CODE + TEXTE I00032I00031%I00030%
(BP.AU RES.) 00032
TAILLES OP.: 0001000800200001 0001000800200001
(BP.AU RES.) 00031
TAILLES OP.: 0001401200200001 0001701300400001
(BP.AU RES.) 0
(BP.AU RES.) 00030
RES. NO: 000300 TAILLES: 000245 00000001 000275 00000001
RESULMOD = 0
* 039 EMPILE
```

```
* 10000000037 DEPILE
* SUCC* DE: +0000000035 ; 067
REGLE 00067 CODE = TEXTE I00031I00032%I00013I00014%
(BP*AU RES*) 00031
TAILLES BP*: 0001401P00P00001 0001701R004000D1
(BP*AU RES*) 00032
TAILLES BP*: 0001000P00P00001 0001000A00200001
(BP*AU RES*) 0
(BP*AU RES*) 00013
RES*NB: 000130 TAILLES: 0001501300P00001 0001R014004000D1
(BP*AU RES*) 00014
RES*NB: 000140 TAILLES: 0001501300200001 0001R014004000D1
(BP*AU RES*) 0
RESULMOD = 0
* 067 EMPILE
* SUCC* DE: +0000000067 ; 068
REGLE 00068 CODE = TEXTE I00013%I00015%
(BP*AU RES*) 00013
TAILLES BP*: 0001501300200001 0001R014004000D1
(BP*AU RES*) 0
(BP*AU RES*) 00015
RES*NB: 000150 TAILLES: 0001501300P00001 0001R014004000D1
(BP*AU RES*) 0
RESULMOD = 0
* 068 EMPILE
* SUCC* DE: +0000000068 ; 070
REGLE 00070 CODE = TEXTE N00004004000 I00015%I00014%
(BP*AU RES*) 00004004000
TAILLES BP*: 0030004004000       00004004000
(BP*AU RES*) 00015
TAILLES BP*: 0001501300P00001 0001R013004000D1
(BP*AU RES*) 0
(BP*AU RES*) 00014
RES*NB: 000140 TAILLES: 0001501300200001 0001R014004000D1
(BP*AU RES*) 0
RESULMOD = 0
* FIN DES SUCC* DE: +0000000068
* +0000000068 DEPILE
* SUCC* DE: +0000000067 ; 073
REGLE 00073 CODE = TEXTE I00013%I00037%
(BP*AU RES*) 00013
TAILLES BP*: 0001501300P00001 0001R014004000D1
(BP*AU RES*) 0
(BP*AU RES*) 00037
RES*NB: 000370 TAILLES: 00017E1400P00001 0001RE1400200001
(BP*AU RES*) 0
RESULMOD = 0
* 073 EMPILE
* SUCC* DE: +0000000073 ; 045
REGLE 00046 CODE = TEXTE I00039I00036I00037I00035I00038%I00033%I00033%
(BP*AU RES*) 00039
TAILLES BP*: 0001RE1500200001 0001RE1500200001
(BP*AU RES*) 00038
TAILLES BP*: 0001RE1800000001 0001RE1800000001
(BP*AU RES*) 00037
TAILLES BP*: 00017E1400200001 0001RE1400200001
(BP*AU RES*) 00036
TAILLES BP*: 0001RE1500200001 0001RE1500200001
(BP*AU RES*) 00035
TAILLES BP*: 00006E0600000001S00006E0600000001S
(BP*AU RES*) 00034
TAILLES BP*: 00013X  00000001 00013X  00000001
(BP*AU RES*) 0
(BP*AU RES*) 00033
RES* NB: 000330 TAILLES: 00003X  00000001 00021S  00000001
```



```
TAILLES
* FIN DES SUCC* DE: +0000000046
* +0000000046 DEPILE
* SUCC* DE: +0000000073 ; 075
REGLE 00075 CODE = TEXTE I00033%I00034%
(BP*AU RES*) 00033
TAILLES BP*: 00003S  00000001 00031S  00000001
(BP*AU RES*) 0
(BP*AU RES*) 00034
RES*NB: 000340 TAILLES: 0013PX  00000001 0013PX  00000001
(BP*AU RES*) 0
RESULMOD = N
* FIN DES SUCC* DE: +0000000046
* +0000000046 DEPILE
* FIN DES SUCC* DE: +0000000073
* +0000000073 DEPILE
* SUCC* DE: +0000000067 ; 071
REGLE 00071 CODE = TEXTE I00014%I00013%
(BP*AU RES*) 00014
TAILLES BP*: 0001501300200001 0001801400400001
(BP*AU RES*) 0
(BP*AU RES*) 00013
RES*NB: 000130 TAILLES: 0001501300200001 0001801400400001
(BP*AU RES*) 0
RESULMOD = N
* SUCC* DE: +0000000067 ; 072
REGLE 00072 CODE = TEXTE I00014%I00017%I00017%
(BP*AU RES*) 00014
TAILLES BP*: 0001501300200001 0001801400400001
(BP*AU RES*) 00017
TAILLES BP*: 0001801600200001 0001801600200001
(BP*AU RES*) 0
(BP*AU RES*) 00017
RES*NB: 000170 TAILLES: 0001801600200001 0001801700100001
(BP*AU RES*) 0
RESULMOD = 0
* 072 EMPILE
* SUCC* DE: +0000000072 ; 074
REGLE 00074 CODE = TEXTE I00017%I00038%
(BP*AU RES*) 00017
TAILLES BP*: 0001801600200001 0001801700100001
(BP*AU RES*) 0
(BP*AU RES*) 00038
RES*NB: 000380 TAILLES: 0001RE1800000001 0001RF1800000001
(BP*AU RES*) 0
RESULMOD = N
* FIN DES SUCC* DE: +0000000072
* +0000000072 DEPILE
* FIN DES SUCC* DE: +0000000067
* +0000000067 DEPILE
* FIN DES SUCC* DE: +0000000036
* +0000000036 DEPILE
***** FIN ITER
***** FIN D UNE MODIF
```

```
00001000014        00600080FD 00000001 00080FD 00000001 CARTES
00001100000AC000K0004H00041 0005 0006F
00002000015        002000EDX  00000001 00080X  00000001 CRTF
00002100000C00000H
00003000016        00200132FD 00000001 00132FD 00000001 LIGNES
0000310000D00000N
00004000017        0030C132X  00000001 00132X  00000001 LGNF
000041000F0000E0007N
00005000019        003000070070000001 000070070000001 W1
00005100000G0005J0005C
00006000020        003000060060000001 000060060000001 W2
00006100000H0005K0005C
00007000021        003000090070300001 000090070020001 W3
0000710000I0005L0005D
00008000022        004000050050000001 0000500500000.. W4
0000810001 0005M0005G0005H
00009000023        003000012E070020001 00012E070020001 W6
000091001A0005M0005E
00010000024        0030004F   00000001S00004F  00000001SM1
000101001B0005R0006A
00011000025        0030004F   00000001S00004F  00000001SM2
000111001C0006A0006A
00012000026        0030004F   00000001S00004F  00000001SX
0001210001D0006J0006B
00013000027        006000150130020001 000180140040001 P
000131001E0006P0006H0006P0007J0007C
00014000028        005000150130020001 000180140040001 Q
000141001F0006P0007A0007A0007B
00015000029        003000150130020001 000180140040001 TPV
0001510001G0006Q0007
00016000030        005000010010000001 000010010000001 I
0001610001H0006L0006M006E000A1
00017000031        005000180160020001 000180170010001 S
00017100011I0006M0007 0007B0007D
00018000032        0030011S   00000001 00011S   00000001 CRT1
0001810002 0002K0004G
00019000033        003000050050000001 000050050000001 D1                  CRT1
000191002A0002D0005A
00020000034        003000060060000001 000060060000001 D2                  CRT1
000201002B0002D0005A
00021000035        00300015S  00000001 00016S   00000001 CRT2
000211002C0002R0004R
00022000036        003000030040010001 000050040010001 D3                  CRT2
000221002E0002I0005B
00023000037        004000050050000001 000060000000001 D4                  CRT2
000231002F0002I0005B0005F
00024000038        003000050050000001 000050050000001 D5                  CRT2
000241002G0002I0005F
00025000039        0030028S   00000001 00028S   00000001 CRT3
000251002H0003N0005G
00026000040        003000070050020001 000070050020001 A                   CRT3
0002610003 0003F0005I
00027000041        003000070050020001 000070050020001 B                   CRT3
000271003A0003F0005I
00028000042        003000070050020001 000070050020001 C                   CRT3
000281003B0003F0006
00029000043        003000070050020001 000070050020001 D                   CRT3
000291003C0003F0006
00030000044        00300024S  00000001 00027S   00000001 CRT4
00030100003D0003R00068
00031000045        003000140120020001 000170130040001 M                   CRT4
000311003F0003I0006G
00032000046        003000100080020001 000100080020001 N                   CRT4
000321003G0003I0006G
00033000047        0050090S   00000001 00091S   00000001 ED
00033 0C3H0004H0007E
00034000048        0030013X   00000001 00013X   00000001 FD1                 ED
0003410004 0004F0005N
00035000049        00400006E0600000001S00006E0600000001SFD2                 ED
000351004A0004F0005P0005Q
00036000050        0030018E150020001 00018E150020001 FD3                 ED
000361004B0004F0006K
00037000051        0030017E140020001 00018E140020001 ED4                 ED
000371004C0004F0007L
00038000052        0030018E130000001 00018E180000001 FD5                 ED
000381004D0004F0007M
00039000053        0030018E150020001 00013E150020001 FD6                 ED
000391004E0004F0006K
00040000053        0010048    00000001P00004B  00000001STALLY
000401004G
```



```
IDEN. NUMERO: 000310 MODIFICATION: 00015013000200001
*** ORIGINE OU COMMIT: +0000000036
* SUCC. DE: +0000000036 ; 039
REGLE 00039 CODE H TEXTE I00032I00031*I00030%
(BP.AU RES.) 00032
TAILLES BP.: 00010008002000001 00010008000200001
(BP.AU RES.) 00031
TAILLES BP.: 0001401200200001 0001501300200001
(BP.AU RES.) 8
(BP.AU RES.) 00030
RES. NO: 000300 TAILLES: 00024S  00000001 00025S  00000001
RESULMOD = 9
* 039 EMPILE
* FIN DES SUCC. DE: +0000000039
* +0000000039 DEPILE
* SUCC. DE: +0000000036 ; 067
REGLE 00067 CODE + TEXTE I00031I00032*I00013I00014%
(BP.AU RES.) 00031
TAILLES BP.: 0001401200200001 0001501300200001
(BP.AU RES.) 00032
TAILLES BP.: 0001000800200001 0001000800200001
(BP.AU RES.) 8
(BP.AU RES.) 00013
RES.NO: 000130 TAILLES: 0001501300200001 0001601400200001
(BP.AU RES.) 00014
RES.NO: 000140 TAILLES: 0001501300200001 0001601400200001
(BP.AU RES.) 8
RESULMOD = 8
* 067 EMPILE
* SUCC. DE: +0000000067 ; 068
REGLE 00068 CODE M TEXTE I00013MI00015%
(BP.AU RES.) 00013
TAILLES BP.: 0001501300200001 0001601400200001
(BP.AU RES.) 8
(BP.AU RES.) 00015
RES.NO: 000150 TAILLES: 0001501300200001 0001601400200001
(BP.AU RES.) 8
RESULMOD = 8
* 068 EMPILE
* SUCC. DE: +0000000068 ; 070
REGLE 00070 CODE - TEXTE N00004004000 I00015*I00014%
(BP.AU RES.) 00004004000
TAILLES BP.: 00004004000      00004004000
(BP.AU RES.) 00015
TAILLES BP.: 0001501300200001 0001601400200001
(BP.AU RES.) 8
(BP.AU RES.) 00014
RES.NO: 000140 TAILLES: 0001501300200001 0001601400200001
(BP.AU RES.) 8
RESULMOD = N
* FIN DES SUCC. DE: +0000000068
* +0000000068 DEPILE
* SUCC. DE: +0000000067 ; 073
REGLE 00073 CODE M TEXTE I00013*I00037%
(BP.AU RES.) 00013
TAILLES BP.: 0001501300200001 0001601400200001
```

```
(BP+4U RES+) $
(BP+4U RES+) 00037
RES+N8: 000370 TAILLES: 0C017E1400200001 00017E1400200001
(BP+4U RES+) $
RESULMBD = N
* SUCC+ DE: +0000000067 ; 071
REGLE 00071 CODE M TEXTE I00011 00013%
(BP+4U RES+) 00014
TAILLES BP+: 0001501300200001 0001601400200001
(BP+4U RES+) 0
(BP+4U RES+) 00013
RES+N8: 000130 TAILLES: 0001501300200001 0001601400200001
(BP+4U RES+) $
RESULMBD = N
* SUCC+ DE: +0000000067 ; 072
REGLE 00072 CODE + TEXTE I00014I00017$I00017%
(BP+4U RES+) 00014
TAILLES BP+: 0001501300200001 0001601400200001
(BP+4U RES+) 00017
TAILLES BP+: 0001801600200001 0001801700100001
(BP+4U RES+) $
(BP+4U RES+) 00017
RES+N8: 000170 TAILLES: 0001801600200001 0001801700100001
(BP+4U RES+) $
RESULMBD = 0
* 072 EMPILE
* SUCC+ DE: +0000000072 ; 074
REGLE 00074 CODE M TEXTE I00017$I00038%
(BP+4U RES+) 00017
TAILLES BP+: 0001801600200001 0001801700100001
(BP+4U RES+) 0
(BP+4U RES+) 00038
RES+N8: 000380 TAILLES: 0001RE1800000001 0001RE1400000001
(BP+4U RES+) $
RESULMBD = N
* FIN DES SUCC+ DE: +0000000072
* +0000000074 DEPILE
* FIN DES SUCC+ DE: +0000000067
* +0000000067 DEPILE
* FIN DES SUCC+ DE: +0000000036
* +0000000036 DEPILE
***** FIN ITER
***** FIN D UNE MBDIF
IDEN+ NUMERB: 000310 MBDIFICATION 00017013004000001
*** ORIGINE DU CHMNT: +000000003A
* SUCC+ DE: +0000000036 ; 039
REGLE 00039 CODE M TEXTE I00032I00031$I00030%
(BP+4U RES+) 00032
TAILLES BP+: 0001000800200001 0001000800200001
(BP+4U RES+) 00031
TAILLES BP+: 0001401200200001 0001701300400001
(BP+4U RES+) $
(BP+4U RES+) 00030
RES+N8: 000300 TAILLES: 00C24S 00000001 00027S 00000001
RESULMBD = 0
* 039 EMPILE
* FIN DES SUCC+ DE: +0000000039
* +0000000039 DEPILE
* SUCC+ DE: +0000000036 ; 067
REGLE 00067 CODE + TEXTE I00031I00032$I00013I00014%
(BP+4U RES+) 00031
TAILLES BP+: 0001401200200001 0001701300400001
(BP+4U RES+) 00032
TAILLES BP+: 0001000800200001 0001000800200001
(BP+4U RES+) 0
(BP+4U RES+) 00013
```



```
RES+N8: 000140 TAILLES: 0C015C13C0200001 0001801400400001
(BP+4U RES+) $
RESULMBD = 0
* 067 EMPILE
* SUCC+ DE: +0000000067 ; 068
REGLE 00068 CODE M TEXTE I00013$I00015%
(BP+4U RES+) 00013
TAILLES BP+: 0001501300200001 0001801400400001
(BP+4U RES+) $
(BP+4U RES+) 00015
RES+N8: 000150 TAILLES: 0001501300200001 0001801400400001
(BP+4U RES+) $
RESULMBD = 0
* 068 EMPILE
* SUCC+ DE: +0000000065 ; 070
REGLE 00070 CODE + TEXTE N0C004004000 I00015$I00014%
(BP+4U RES+) 00004004000
TAILLES BP+: 00004004000 00004004000
(BP+4U RES+) 00015
TAILLES BP+: 0001801300200001 0001801400400001
(BP+4U RES+) $
(BP+4U RES+) 00014
RES+N8: 000140 TAILLES: 0001501300200001 0001801400400001
(BP+4U RES+) $
RESULMBD = N
* FIN DES SUCC+ DE: +0000000068
* +0000000070 DEPILE
* SUCC+ DE: +0000000067 ; 073
REGLE 00073 CODE M TEXTE I0C0134I00037%
(BP+4U RES+) 00013
TAILLES BP+: C0015013J0200001 0001801400400001
(BP+4U RES+) $
(BP+4U RES+) 00037
RES+N8: 000370 TAILLES: 0C017E1400200001 0001RE1400200001
(BP+4U RES+) $
RESULMBD = 0
* 073 EMPILE
* SUCC+ DE: +0000000073 ; 046
REGLE 00046 CODE M TEXTE I0C038I00038I00037I00036I00035I00034$I00033%
(BP+4U RES+) 00039
TAILLES BP+: 0001RE1500200001 0001RE1500P00001
(BP+4U RES+) 00038
TAILLES BP+: 0001RE1800000001 0001RE1400000001
(BP+4U RES+) 00037
TAILLES BP+: 0001721400200001 0001RE1400P00001
(BP+4U RES+) 00036
TAILLES BP+: 0001RE1500200001 0001RE1500P00001
(BP+4U RES+) 00035
TAILLES BP+: 00004E0600000001800006E0600000001S
(BP+4U RES+) 00034
TAILLES BP+: 00014X 00000001 00013X 00000001
(BP+4U RES+) 0
(BP+4U RES+) 00033
RES+N8: 000330 TAILLES: 00090S 00000001 00091S 00000001
RESULMBD = 0
* 046 EMPILE
* SUCC+ DE: +0000000046 ; 075
REGLE 00075 CODE M TEXTE I00033$I00034%
(BP+4U RES+) 00033
TAILLES BP+: 00090S 00000001 00091S 00000001
(BP+4U RES+) 0
(BP+4U RES+) 00034
RES+N8: 000340 TAILLES: 00132X 00000001 00133X 00000001
(BP+4U RES+) 0
RESULMBD = N
* FIN DES SUCC+ DE: +0000000046
* +0000000046 DEPILE
* FIN DES SUCC+ DE: +0000000073
* +0000000073 DEPILE
* SUCC+ DE: +0000000067 ; 071
REGLE 00071 CODE M TEXTE I00014$I00013%
(BP+4U RES+) 00014
TAILLES BP+: 0001501300200001 0001801400400001
(BP+4U RES+) 0
(BP+4U RES+) 00013
RES+N8: 000130 TAILLES: 0001801300200001 0001801400400001
(BP+4U RES+) $
RESULMBD = N
* SUCC+ DE: +0000000067 ; 072
REGLE 00072 CODE + TEXTE I00014I00017$I00017%
(BP+4U RES+) 00014
TAILLES BP+: 0001501300200001 0001801400400001
(BP+4U RES+) 00017
TAILLES BP+: 0001801600200001 0001801700100001
(BP+4U RES+) 0
(BP+4U RES+) 00017
RES+N8: 000170 TAILLES: 0001801600200001 0001801700100001
(BP+4U RES+) $
RESULMBD = N
* FIN DES SUCC+ DE: +0000000067
* +0000000067 DEPILE
* FIN DES SUCC+ DE: +0000000036
* +0000000036 DEPILE
***** FIN ITER
***** FIN D UNE MBDIF
```

CRT1
CRT1
CRT2
CRT2
CRT2
CRT3
CRT3
CRT3
CRT3
CRT4
CRT4
CRT4
ED
ED
ED
ED
ED
ED

CONCLUSION

L'évolution actuelle de la conception de l'automatisation des applications tend à faire disparaître la notion de chaîne de traitements au profit de ce que nous appelons dans Remora "algorithme programmé" ; la formalisation de la notion de programme par le complexe d'influences nous semble une étape indispensable de cette évolution et nous envisageons en particulier d'utiliser des complexes d'influences portant sur une autre caractéristique que la taille.

Dans une telle hypothèse, les outils de datamatique deviennent indispensables puisqu'il s'agit de rendre la manipulation des variables intermédiaires totalement transparente à l'utilisateur : en effet, si ces variables sont utiles à la description modulaire de l'algorithme dans le langage procédural, elles doivent pouvoir être ignorées du concepteur dans toutes les autres étapes.

Ils conservent bien évidemment leur rôle essentiel dans la maintenance d'un "jeu" de programmes.

Bien que la réalisation pratique de ces outils ait demandé un important travail de programmation d'ailleurs encore insuffisant, elle nous paraît fondamentale dans ce type de recherche afin de fournir effectivement un outil opérationnel preuve de la viabilité des principes énoncés.

ANNEXES

Annexe 1 : Complexe d'influences et graphes (exemple d'appui)

Annexe 2 : Cheminements sur l'exemple d'appui

Annexe 3 : Règles de calcul des tailles

Annexe 4 : Décodage Cobol

Annexe 5 : Correspondance entre la numérotation des objets de l'annexe 1 et celle des résultats de l'ordinateur.

ANNEXE 1

Complexes d'influences et Graphes (exemple d'appui)

figure 1 - programme Cobol

a) description des données

```
00012          DATA DIVISION.
00013          FILE SECTION.
00014          FD  CARTES LABEL RECORD OMITTED DATA RECORD CRTE.
00015          01  CRTE PIC X(80).
00016          FD  LIGNES LABEL RECORD OMITTED DATA RECORD LGNE.
00017          01  LGNE PIC X(132).
00018          WORKING-STORAGE SECTION.
00019          77  W1 PIC 9(7).
00020          77  W2 PIC 9(6).
00021          77  W3 PIC 9(7)V9(2).
00022          77  W4 PIC 9(5).
00023          77  W6 PIC 7.ZZZ.ZZZ.99.
00024          77  M1 COMP-1.
00025          77  M2 COMP-1.
00026          77  X COMP-1.
00027          77  P PIC 9(13)V9(2).
00028          77  Q PIC 9(13)V9(2).
00029          77  TRV PIC 9(13)V9(2).
00030          77  I PIC 9.
00031          77  S PIC 9(16)V9(2).
00032          01  CRT1.
00033              02  D1 PIC 9(5).
00034              02  D2 PIC 9(6).
00035          01  CRT2.
00036              02  D3 PIC 9(4)V9.
00037              02  D4 PIC 9(5).
00038              02  D5 PIC 9(5).
00039          01  CRT3.
00040              02  A PIC 9(5)V9(2).
00041              02  B PIC 9(5)V9(2).
00042              02  C PIC 9(5)V9(2).
00043              02  D PIC 9(5)V9(2).
00044          01  CRT4.
00045              02  M PIC 9(12)V9(2).
00046              02  N PIC 9(8)V9(2).
00047          01  ED.
00048              02  ED1 PIC X(13).
00049              02  ED2 PIC +(6).
00050              02  ED3 PIC Z(15).Z(2).
00051              02  ED4 PIC Z(14).9(2).
00052              02  ED5 PIC Z(14).
00053              02  ED6 PIC Z(16).Z(2).
```

figure 1 - programme Cobol

b) description des traitements

```
00054          PROCEDURE DIVISION.
00055          DEBUT. OPEN INPUT CARTES OUTPUT LIGNES.
00056          LECTURES. READ CARTES INTO CRT1 AT END GO TO FIN.
00057              READ CARTES INTO CRT2 AT END DISPLAY 'ER2' GO TO FIN.
00058              READ CARTES INTO CRT3 AT END DISPLAY 'ER3' GO TO FIN.
00059              ADD D1 D2 GIVING W1 ADD D3 D4 GIVING W2.
00060              DIVIDE W2 INTO W1 GIVING W3
00061              MOVE W3 TO W6 MOVE W6 TO ED1.
00062              SUBTRACT D5 FROM D4 GIVING W4.
00063              IF W3 > W4 MOVE W4 TO ED2  ELSE
00064              SUBTRACT W4 FROM 0 GIVING ED2.
00065              ADD A B GIVING M1 ADD C D GIVING M2
00066              MULTIPLY M1 BY M2 GIVING X.
00067              MOVE X TO ED3 ED6.
00068              MOVE 0 TO I. MOVE 0 TO S.
00069          BOUCLE. ADD 1 I.
00070              IF I > 8 GO TO ECRITURE.
00071              READ CARTES INTO CRT4 AT END DISPLAY 'ER4' GO TO FIN.
00072              ADD M N GIVING P Q.
00073              MOVE P TO TRV.
00074              IF TRV NOT > 5000 MOVE I TO P ELSE
00075              SUBTRACT 5000 FROM TRV GIVING Q.
00076              IF TRV > 500 MOVE Q TO P.
00077              ADD Q TO S GO TO BOUCLE.
00078          ECRITURE.
00079              MOVE P TO ED4 MOVE S TO ED5.
00080              WRITE LGNE FROM ED.
00081              GO TO LECTURES.
00082          FIN. CLOSE CARTES LIGNES STOP RUN.
```


désignation	t. totale	t. entière	t. décimale	type
A	7	5	2	)'
B	7	5	2	)'
C	7	5	2	)'
CARTES	80	/	/	fichier
CRTE	80	/	/	enregistrement
CRT1	11	/	/	structure
CRT2	15	/	/	structure
CRT3	28	/	/	structure
CRT4	24	/	/	structure
D	7	5	2	)'
D1	5	5	0	)'
D2	6	6	0	)'
D3	5	4	1	)'
D4	5	5	0	)'
D5	5	5	0	)'
ED	90	/	/	structure
ED1	13	/	/	-X)'
ED2	6	6	0	)'édité
ED3	18	15	2	)'édité
ED4	17	14	2	)'édité
ED5	18	18	0	)'édité
ED6	18	15	2	)'édité
I	1	1	0	)'
LGNE	132	/	/	enregistrement
LIGNES	132	/	/	fichier
M	14	12	2	)'
M1	4	/	/	réel
M2	4	/	/	réel
N	10	8	2	)'
P	15	13	2	)'
Q	15	13	2	)'
S	18	16	2	)'
TRV	15	13	2	)'
W1	7	7	0	)'
W2	6	6	0	)'
W3	9	7	2	)'
W4	5	5	0	)'
W6	12	9	2	)'édité
X	4	/	/	réel

fig 2 - complexe d'influences a) les mots et leur caractéristique

1 - CRTE "constitue" CARTES (fichier)
2 - LGNE "constitue" LIGNES (fichier)
3 - D1 et D2 constituent CRT1
4 - D3, D4 et D5 constituent CRT2
5 - A, B, C et D constituent CRT3
6 - M et N constituent CRT4
7 - ED1, ED2, ED3, ED4, ED5 et ED6 constituent ED
8 - CARTES ⟶ CRT1
9 - CARTES ⟶ CRT2
10 - CARTES ⟶ CRT3
11 - D1 + D2 ⟶ W1
12 - D3 + D4 ⟶ W2
13 - W1/W2 ⟶ W3
14 - W3 ⟶ W6
15 - W6 ⟶ ED1
16 - D4-D5 ⟶ W4
17 - W4 ⟶ ED2
18 - W4 ⟶ ED2
19 - A+B ⟶ M1
20 - C+D ⟶ M2
21 - M1 x M2 ⟶ X
22 - X ⟶ ED3, ED6
23 - 0 ⟶ I
24 - 0 ⟶ S
25 - I + 1 ⟶ I
26 - CARTES ⟶ CRT4
27 - M + N ⟶ P, Q
28 - P ⟶ TRV
29 - I ⟶ F
30 - TRV - 5000 ⟶ Q
31 - Q ⟶ F
32 - S + Q ⟶ S
33 - P ⟶ ED4
34 - S ⟶ ED5
35 - ED ⟶ LGNE

Figure 2 - complexe d'influences b) les groupes opératoires

dûs à la déclaration des données

dûs aux traitements

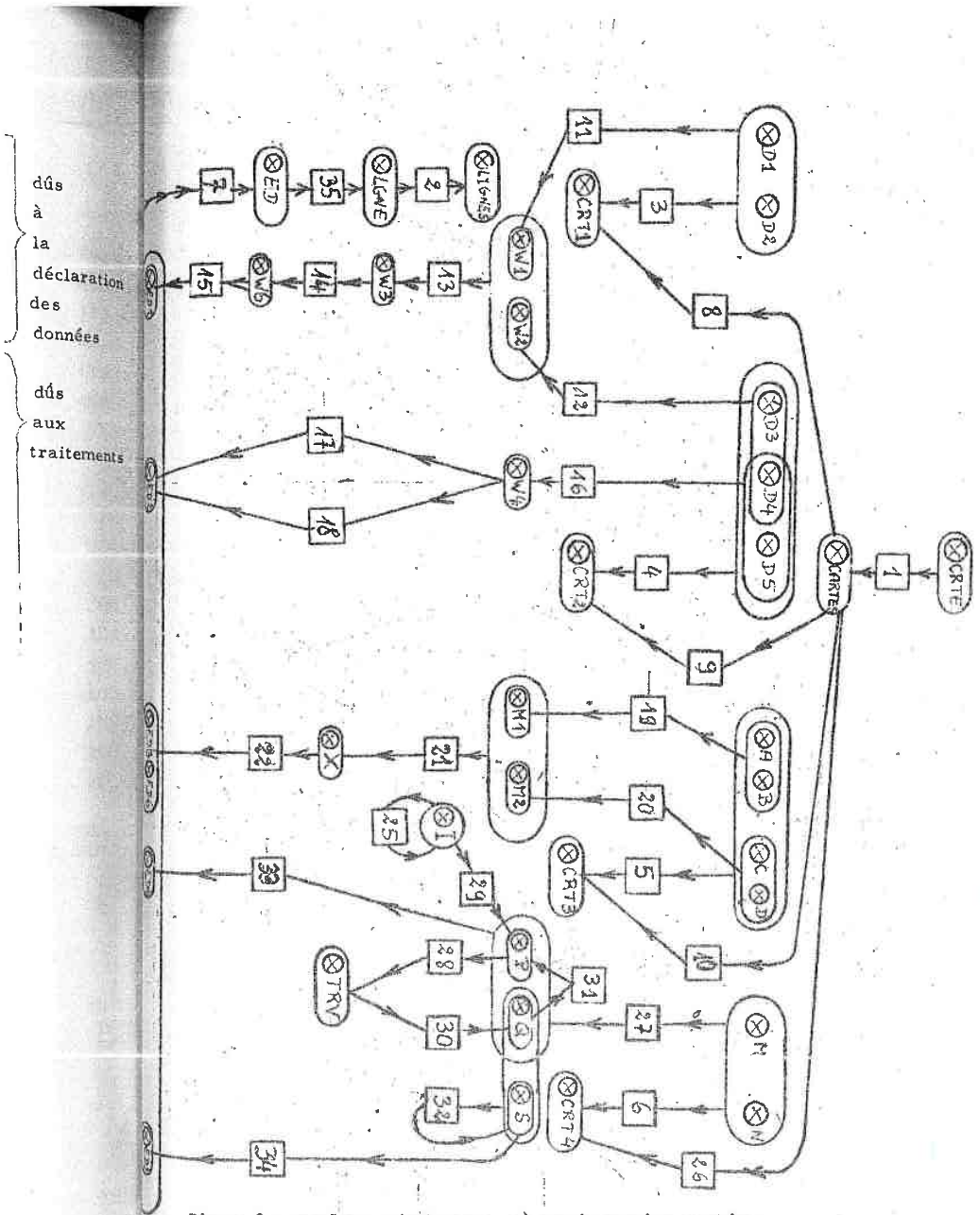


Figure 2 : complexe d'influences c) représentation graphique

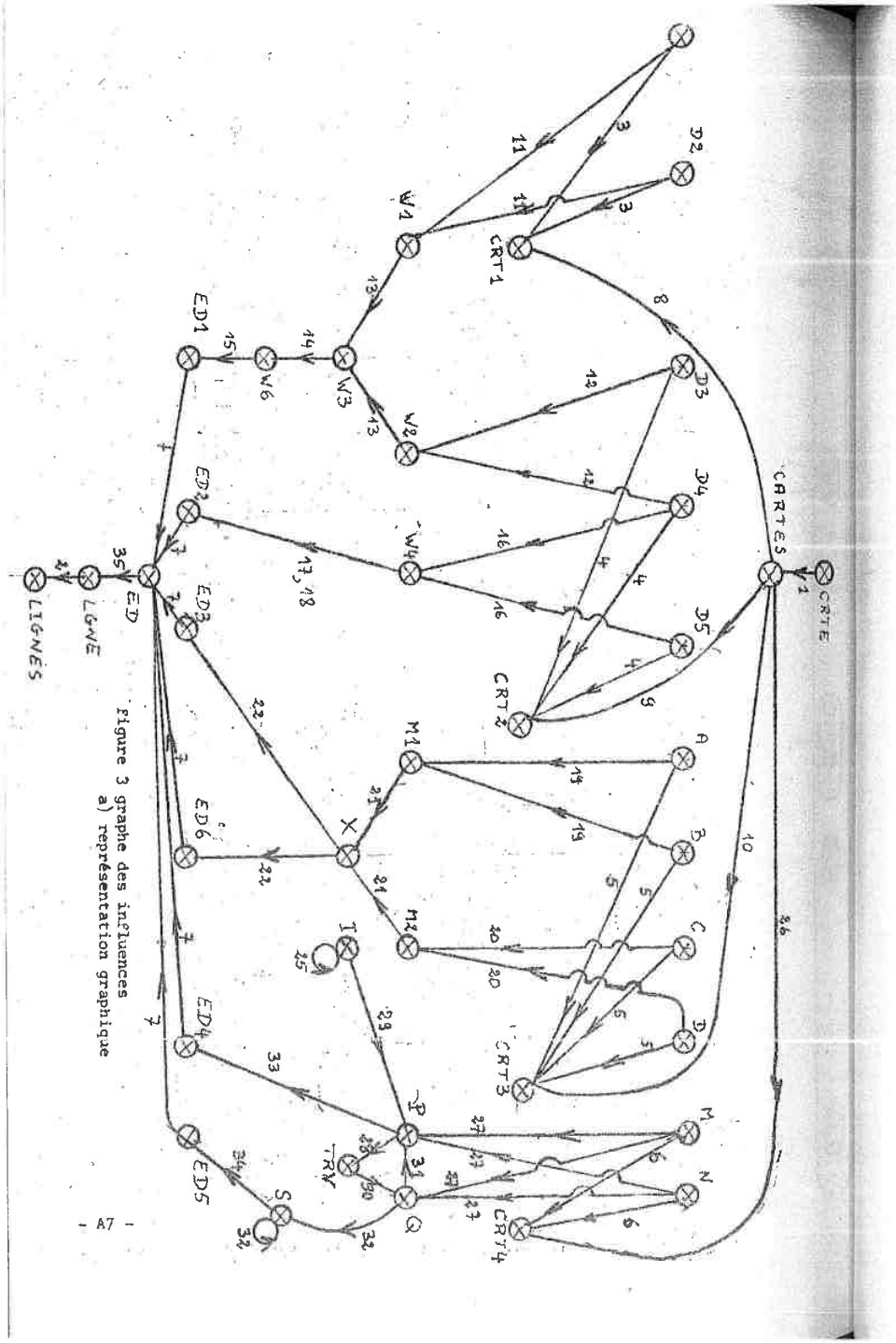


figure 3 graphe des influences
a) représentation graphique

mots	mots successeurs avec, entre parenthèses, les numéros des relations d'influence correspondantes
A	CRT3(5);M1(19)
B	
C	CRT3(5);M2(20)
CARTES	CRT1(8);CRT2(9);CRT3(10);CRT4(26)
CRTE	CARTES (1)
CRT1	
CRT2	
CRT3	
CRT4	
D	cf. C
D1	CRT1(3);W1(11)
D2	
D3	CRT2(4) ;W2(12)
D4	CRT2(4) ; W2(12);W4(16)
D5	CRT2(4) ; W4(16)
ED	LGNE(35)
ED1	
ED2	
ED3	
ED4	ED (7)
ED5	
ED6	
I	I (25) ; P (29)
LGNE	LIGNES (2)
LIGNES	/
M	CRT4(6) ; P(27) ; q Q (27)
M1	X (21)
M2	
N	cf M
P	TRV(28) ; ED4 (33)
Q	P(31) ; S (32)
S	S(32) ; EDS (34)
TRV	Q(30)
W1	W3 (13)
W2	
W3	W6 (14)
W4	ED2 (17,18)
W6	ED1 (15)
X	ED3 (22) ; ED6(22)

figure 3 graphe des influences b) représentation sous forme de listes de successeurs

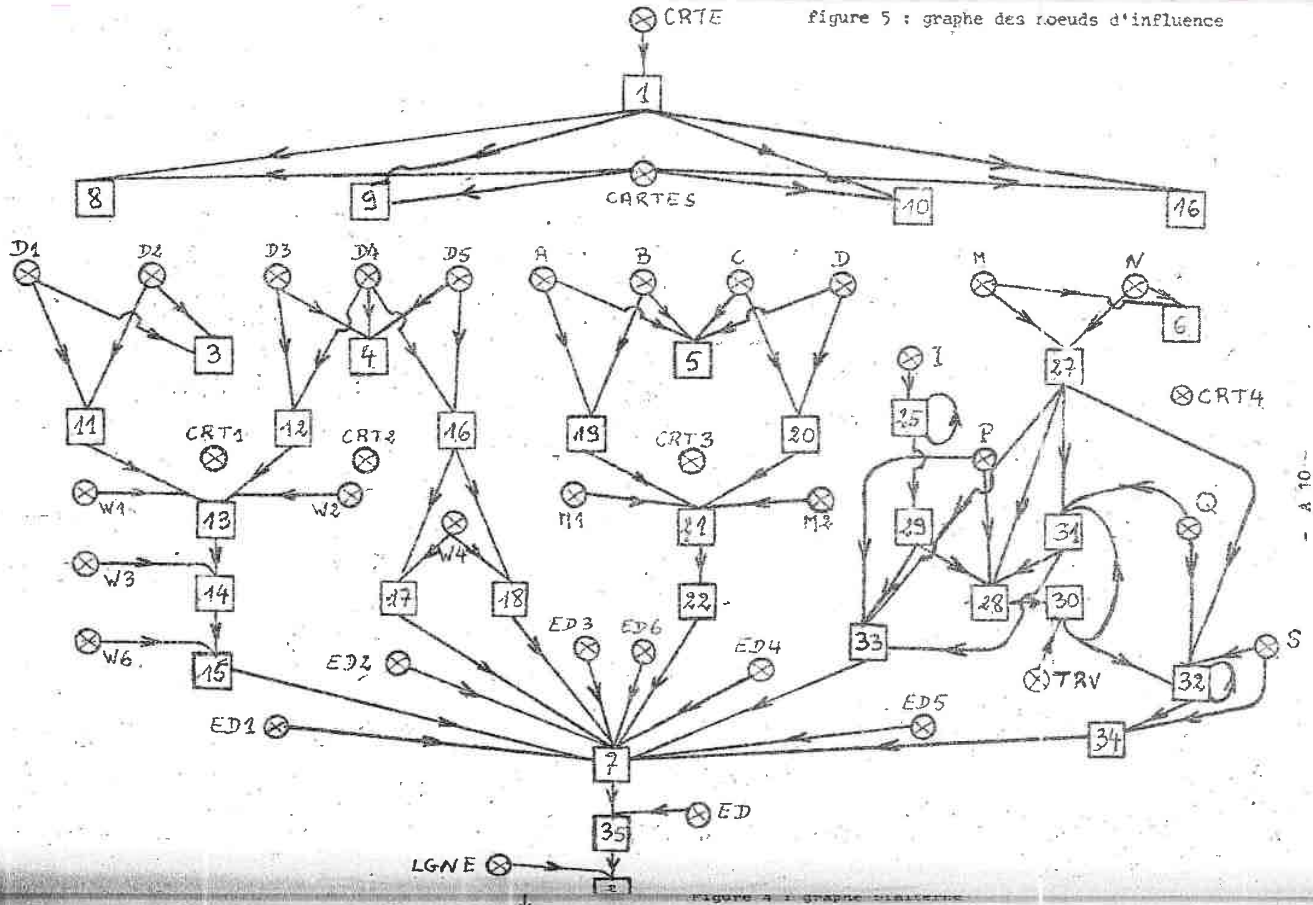
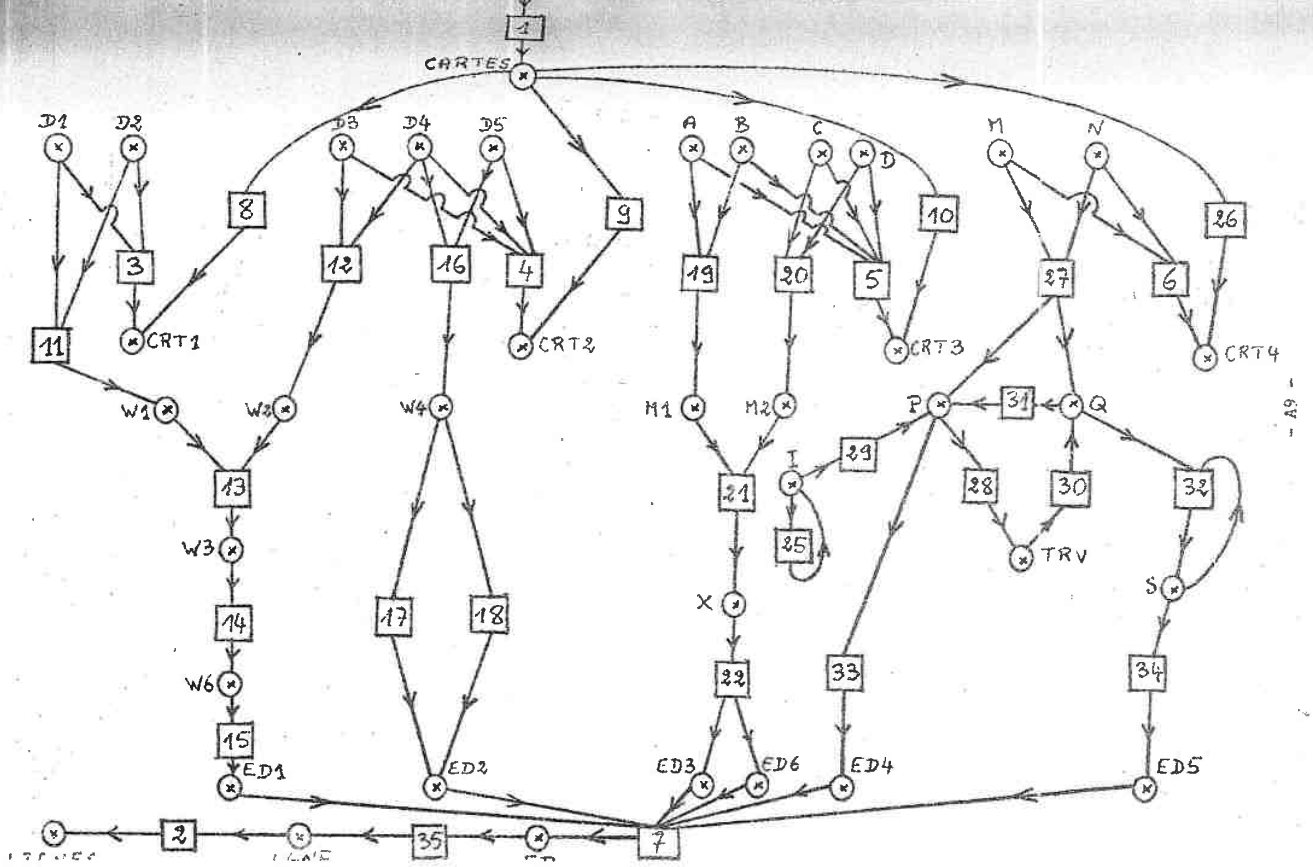


Figure 5 : graphe des noeuds d'influence

- A 10 -

Figure 4 : graphe didactique



- A9 -

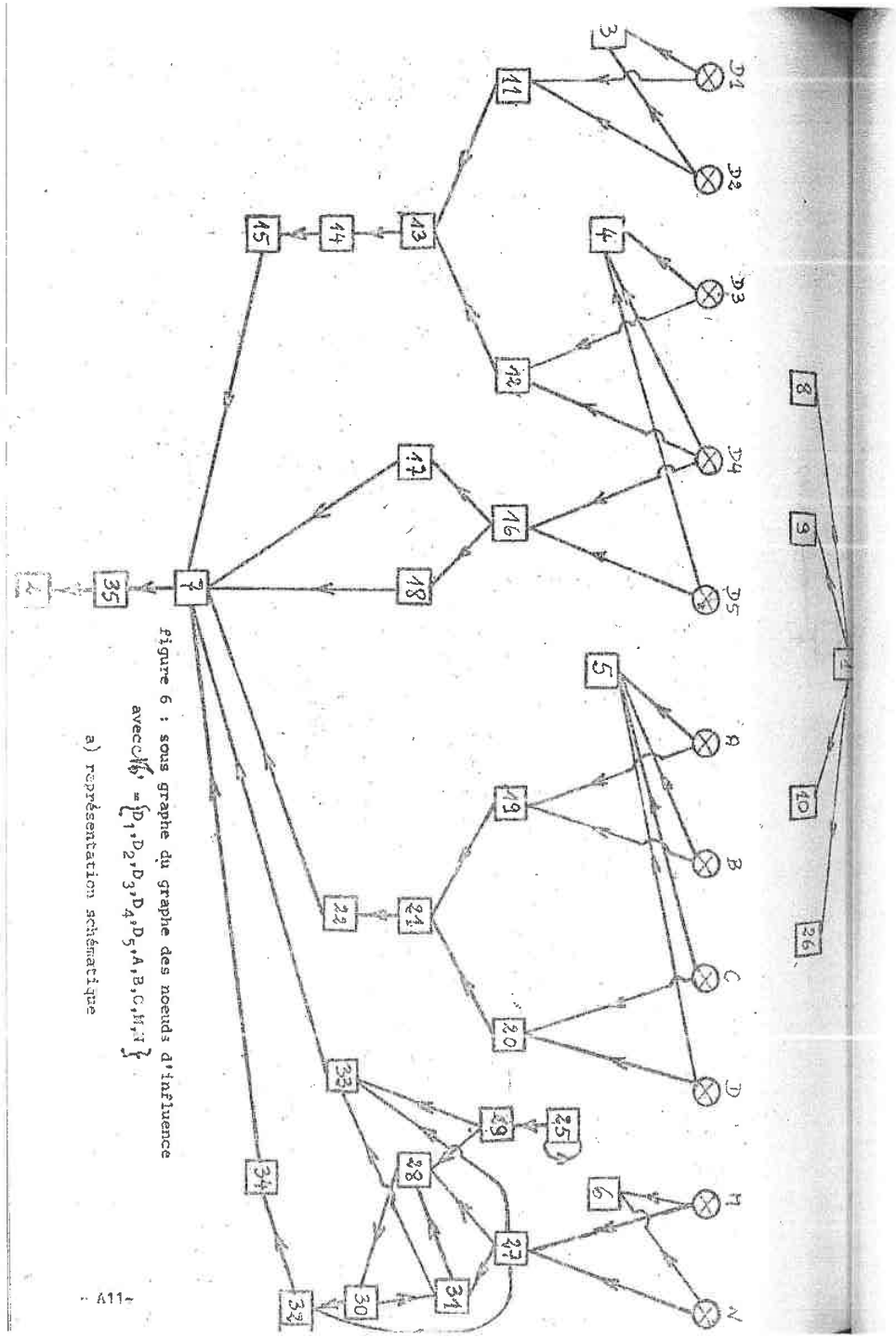


figure 6 : sous graphe du graphe des noeuds d'influence avec $\Phi = \{D_1, D_2, D_3, D_4, D_5, A, B, C, H, N\}$

a) représentation schématique

n° relation	liste des successeurs
1	8 9 10 26
2	nil
3	
4	
5	
6	
7	35
8	nil
9	
10	
11	13
12	
13	14
14	15
15	7
16	17 18
17	7
18	
19	21
20	
21	22
22	7
25	25 29
26	nil
27	28 31 32 33
28	30
29	28 33
30	31 32
31	28 33
32	32 34
33	7
34	
35	2

figure 6 - Sous-graphe $(\mathcal{R}, \Gamma)$ du graphe des noeuds d'influence
b) sous forme de liste de successeurs

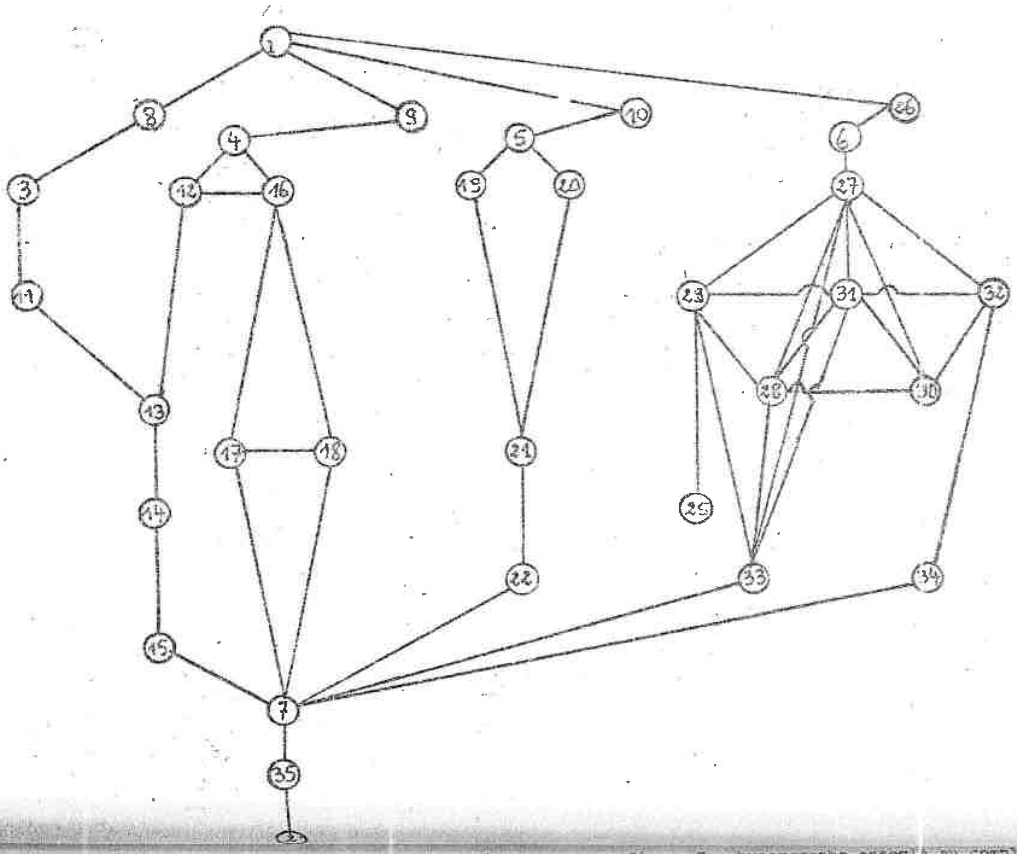
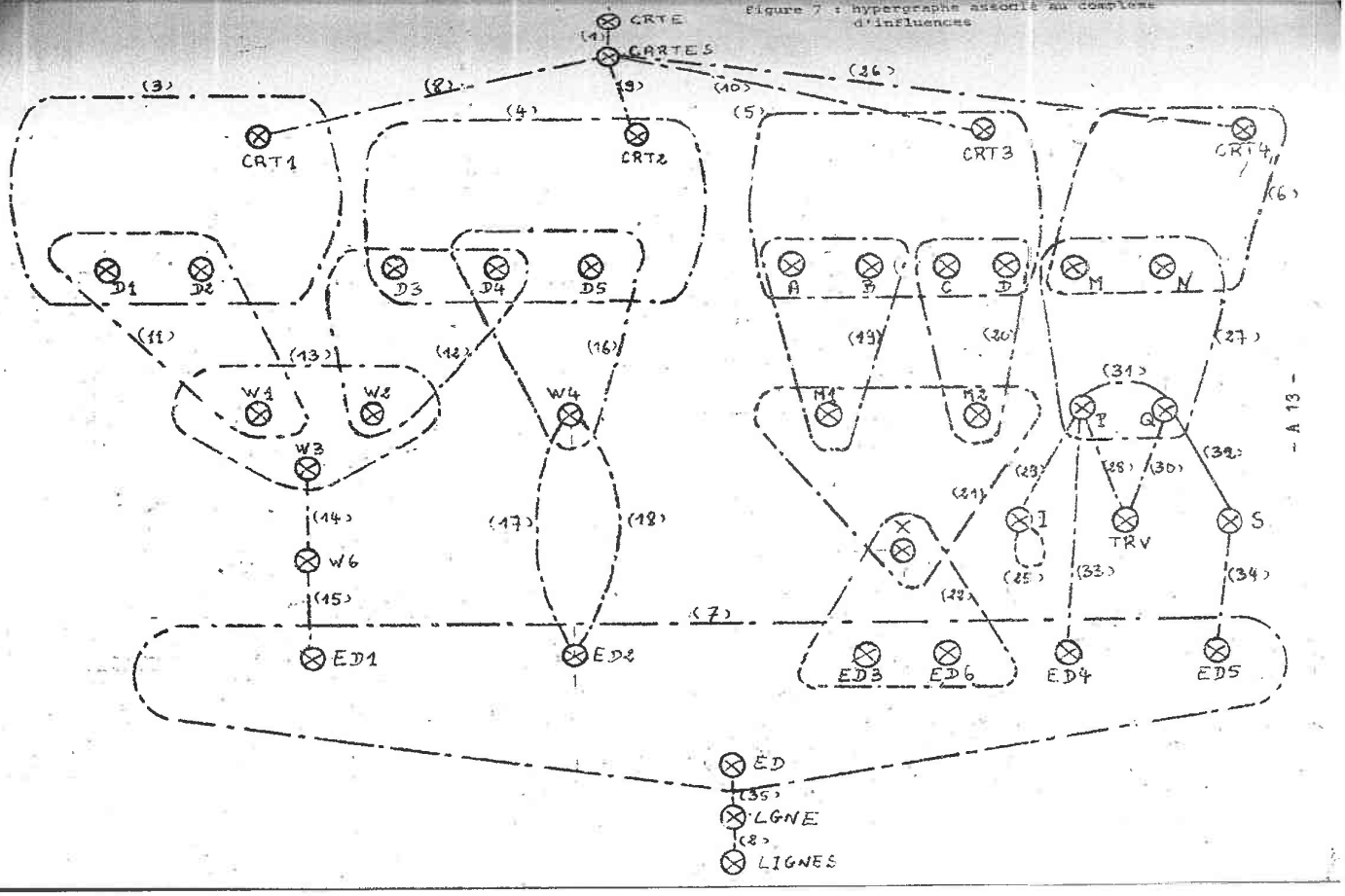


figure 8 : graphe représentatif des arêtes de l'hypergraphe associé au complexe d'influences

figure 7 : hypergraphe associé au complexe d'influences



ANNE XE 2

Cheminements sur

l'exemple d'appui

Figure 1 : modifications en cascade (itérations secondaires)

Figure 2 : évaluation des tailles inconnues

Figure 1 : itérations secondaires de l'exemple avec circuit et doublet : suivie en pas à pas (sans conditionnement)

résultats de la première itération :

m(M) : = vrai

pile p vide

file f = (28, 7, 31, 32, 33)

deuxième itération

f = (7, 31, 32, 33) ; p_o=(28) ; t(28) := 1 ; m (28) : = vrai

t (x) : = 0 pour x ≠ 28

p_1 = (28, 30) ; t (30) : = 1

p_2 = (28, 30, 31) ; t (31) : = 1

28 successeur de 31 = base de p :

édition du circuit | 28, 30, 31 | ; t (28) : = 2

p_3 = (28, 30, 31, 33) ; t (33) : = 1

p_4 = (28, 30, 31, 33, 7) ; t (7) : = 1

p_5 = (28, 30, 31, 33, 7, 35) ; t (35) : = 1

p_6 = (28, 30, 31, 33, 7, 35, 2) ; t (2) : = 1

p7 = P5

P8 = P4

P9 = P3

p10= p2

p11= p1=(28, 30)

p12=(28, 30, 32)

p13=(28, 30, 32, 34)

file f= (7, 31, 32, 33, 7) ; t(7) = 2

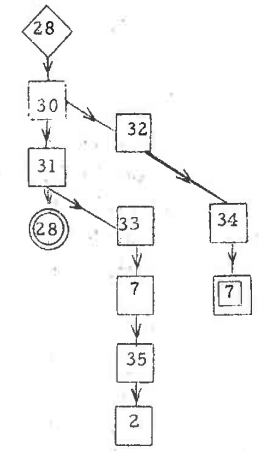
p14=p12

p15=p11

p16=(28)

p17=vide

cheminement



.../...

troisième itération

f=(31, 32, 33, 7) ; po=(7) ;

t(x):= o pour x ≠ 7 ; t(7):=1 ; m(7):= vrai

p1=(7, 35) ; t(35):=1

p2=(7, 35, 2) ; t (2):= 1

p3=p1

p4=po

p5=vide

quatrième itération

f=(32, 33, 7) ; po = (31) ;

t(x):= o pour x ≠ 31 ; t(31):=1 ; m(31) := vrai

p1=(31, 28) ; t(28):=1

p2=(31, 28, 30);t(30):=1

31 successeur de 30 = base de p

édition du circuit 31, 28, 30 ; t(31):=2

p3=(31, 28, 30, 32) ; t(32) :=1

p4=(31, 28, 30, 32, 34);t(34):=1

p5=(31, 28, 30, 32, 34, 7);t(7):=1

p6=(31, 28, 30, 32, 34, 7, 35);t(35):=1

p7=(31, 28, 30, 32, 34, 7, 35, 35, 2) ; t(2):=1

p8=p6

p9=p5

p10=p4

p11=p3

p12=p2

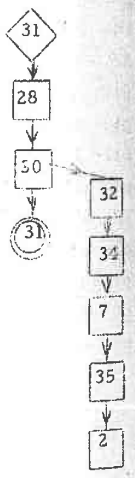
p13=p1

p14=po

p15=vide

cheminement

cheminement



cinquième itération

f=(33, 7);po=32 ;

t(x):=o pour x ≠ 32 ; t(32):=1 ; m(32) :=vrai

p1=(32, 34) ; t(34):=1

p2=(32, 34, 7) ; t(7):=1

p3=(32, 34, 7, 35);t(35):=1

p4=(32, 34, 7, 35, 2) ; t(2) :=1

p5=p3

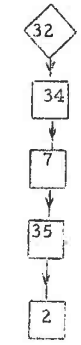
p6=p2

p7=p1

p8=po

p9=vide

cheminement



sixième itération :

f=(7) ; po=33 ;

t(x) :=o pour x ≠ 33 ; t(33):=1 ; m(33) := vrai

p1=(33, 7) ; t(7):=1

p2=(33, 7, 35);t(35):=1

p3=(33, 7, 35, 2);t(2):=1

p4=p2

p5=p1

p6=po

p7=vide

f=vide (car m(7) = vrai)

cheminement

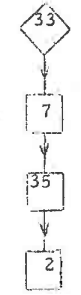


figure 2 évaluation des tailles inconnues :
exemple avec boucle et circuit

Hypothèses
exemple de l'annexe 1
$\mathcal{M}_0 = \{D_1, D_2, D_3, D_4, D_5, A, B, C, D, M, N\}$
(on cherche à évaluer la taille de ED_4)

analyse	règles exécutées	cheminement	t(...) : 0→1	variation de la pile	taille évaluée
initialisation	-	nul	0...0	vide	$\mathcal{M}_0 - \mathcal{J}$
		⊗ ED4	t(ED4)	ED4	
successeur de ED4 : 33		⊗ ED4			
(taille de P ?)		33	t(33)	ED4, 33	
successeur de 33 : 27	27	27			
successeur de 33 : 29		29			
(taille de I ?)			t(29)	ED4, 33, 29	
successeur de 29 : 25		25			
(taille de I ?)			t(25)	ED4, 33, 29, 25	
successeur de 25 : 25		boucle en 25			
(t(25) ≠ 0 !)					
successeur de 25 : fin	25			ED4, 33, 29	I
successeur de 29 : fin	29			ED4, 33	I
successeur de 33 : 31		⊗ ED4			
(taille de Q ?)		33	t(31)	ED4, 33, 31	
successeur de 31 : 27	27	31 / 27			
successeur de 31 : 30		30			
(taille de TRV?)			t(30)	ED4, 33, 31, 30	
successeur de 30 : 28		28			
(taille de P ?)			t(28)	ED4, 33, 31, 30, 28	
successeur de 28 : 27	27	27			
successeur de 28 : 29	29	29			
successeur de 28 : 31		31			
(t(31) ≠ 0 !)		circuit 31, 30, 28			
successeur de 28 : fin	28			ED4, 33, 31, 30	P
successeur de 30 : fin	30			ED4, 33, 31	TRV
successeur de 31 : fin	31			ED4, 33	Q
successeur de 33 : fin	33			ED4	P
successeur de ED4 : fin	def.			vide	ED4

ANNEXE 3 :

formules de calcul des tailles

tableau 1 : évaluation des tailles inconnues (en pseudo-algol)

tableau 2 : modifications en cascade (en pseudo-algol)

tableau 3 : évaluation des tailles inconnues (exemples de formulation sur descripteurs de traitements)

figure 4 : modifications en cascade (programmation des règles arithmétiques selon les principes du chapitre 2.3)

tableau 1 : évaluation des tailles inconnues

(règles construites par le biais systématique exprimées en pseudo-algol)

hypothèse : caractères alphanumériques pour toutes les données(numériques dilatés de Cobol)

notations (rappel)

tout sigle désignant un élément de l'information "taille" est ici constitué par:

1ère lettre : T pour taille

2ème lettre : T, D ou E pour totale, décimale ou entière

3ème lettre : T pour temporaire ou E pour évaluée

indice : Op pour opérande ou Res pour résultat

sous indice : i pour désigner le $i^{ème}$ opérande

règles hiérarchiques

de type "structure" : Op_1 , Op_2 , . . . , Op_n constituent Res

Res est une structure : pas de tailles entière et décimale

$$TTT_{Res} := TTE_{op_1} * REP_{op_1} + \ldots + TTE_{op_n} * REP_{op_n}$$

note :

REP_{op_i} est le facteur de répétition de l'opérande op_i ;

en outre, si plusieurs opérandes sont synonymes, un seul parmi eux intervient dans cette somme.

règles hiérarchiques

de type "fichier" : Op_1 , Op_2 , . . . , Op_n sont les articles du fichier Res

$$TTE_{Res} := \sup_i TTE_{op_i}$$

note :

on obtient directement la taille évaluée car c'est la seule règle où Res est résultat.

règles de redéfinition : Op synonyme de Res

les règles de redéfinition allant toujours par paires :
O_p synonyme de Res et Res synonyme de O_p créent un circuit dans le graphe des noeuds d'influence, aussi accepte-t-on ici des calculs où les tailles des opérandes sont encore temporaires :

TTT_{Res} := sup (TTT_{Res}, TTT_{op})

TTT_{op} := TTT_{Res}

note : on aligne toujours la taille totale des synonymes sur la plus grande taille totale.

règles d'addition : $Op_1 + Op_2 + \ldots + Op_n$ dans Res

retenue : = p + 1 avec $10^p < n \leqslant 10^{p+1}$ et p entier $\geqslant -1$

si Res est numérique (option par défaut)
alors TET_{Res} := sup (TET_{Res}, $\sup_i(TEE_{op_i})$ + retenue)

TDT_{Res} := sup (TDT_{Res}, $\sup_i(TDE_{o_{p_i}})$)

TTT_{Res} := TET_{Res} + TDT_{Res}

sinon (déclaration partiellement réalisée dans le texte source : PIC X (?) par exemple)

TTT_{Res} := sup(TTT_{Res}, $\sup_i(TEE_{o_{p_i}})$ + $\sup_i(TDE_{o_{p_i}})$ + retenue)

finsi

note :
on peut affiner le calcul de la retenue en tenant compte des $TEE_{o_{p_j}} < \sup_i (TEE_{o_{p_i}})$ et en les regroupant en classes, cela donne une formule trop complexe pour sa faible utilisation dans les cas concrets.

règles de soustraction : $o_{p_1} - (o_{p_2} + \ldots + o_{p_n})$ dans Res

si i : op_i non signé
alors retenue = p+1 avec $10^p < n-1 \leqslant 10^{p+1}$ et p entier $\geqslant -1$
sinon retenue = p+1 avec $10^p < n \leqslant 10^{p+1}$ et p entier $\geqslant -1$
finsi
suite des calculs analogue à ceux de l'addition.
note :
c'est le seul cas où le signe intervient (dans l'hypothèse où le signe est superposé, donc ne nécessite pas une position dans la taille totale), pour le traiter convenablement, il faut compléter les règles d'évaluation par l'évaluation de l'existence ou non d'un signe, l'intérêt "relatif" de ce calcul "affiné" de la retenue, mais qui pourrait l'être encore plus (voir note sur l'addition), ne justifie pas, à notre avis, un alourdissement manifeste des règles.

règles de multiplication : $o_{p_1} * o_{p_2}$ dans Res

si Res est numérique (option par défaut)
alors TET_{Res} := sup(TET_{Res}, $TEEo_{p_1} + TEEo_{p_2}$)

TDT_{Res} := sup(TDT_{Res}, $TDEo_{p_1} + TDEo_{p_2}$)

TTT_{Res} := TET_{Res} + TDT_{Res}

sinon

TTT_{Res} := sup(TTT_{Res}, $TEEo_{p_1} + TEEo_{p_2} + TDEo_{p_1} + TDEo_{p_2}$)

finsi

règles de division : op_1 / o_{p_2} dans Res

si Res est numérique (option par défaut)
alors TET_{Res} := sup(TET_{Res}, $TEE_{op_1} + TDEop_2$)

TDT_{Res} := sup(TDT_{Res}, $TDE_{o_{p_1}} + TEEo_{p_2}$)

TTT_{Res} := TET_{Res} + TDT_{Res}

sinon TTT_{Res} := sup(TTT_{Res}, $TEE_{o_{p_1}} + TEE_{o_{p_2}} + TDE_{o_{p_1}} + TDE_{o_{p_2}}$)

finsi

règles d'exponentiation $o_{p_1} \uparrow o_{p_2}$ dans Res

si o_{p_2} = constante

alors EX := partie entière de $(o_{p_2} + 0,5)$

sinon EX := $10 \uparrow TEE o_{p_2}$

finsi

si Res numérique (option par défaut)

alors $TET_{Res} := sup\ (TET_{Res}, EX * TEE_{o_{p_1}})$

$TDT_{Res} := sup\ (TDT_{Res}, EX * TDE_{o_{p_1}})$

$TTT_{Res} := TET_{Res} + TDT_{Res}$

sinon

$TTT_{Res} := sup(TTT_{Res}, EX * TTE_{o_{p_1}})$

finsi

règles de transfert : op dans Res

si o_p et Res numérique

alors $TET_{Res} := sup(TET_{Res}, TEE_{o_p})$

$TDT_{Res} := sup(TDT_{Res}, TDE_{o_p})$

$TTT_{Res} := TET_{Res} + TDT_{Res}$

sinon

$TTT_{Res} := sup\ (TTT_{Res}, TTT_{o_p})$

note :

le cas où op est numérique est analogue au cas de l'addition en prenant n = 1

ableau 2 : modifications en cascade

ègles programmées pour nos essais exprimées en pseudo-algol)

ous ensemble correspondant à la même hypothèse que le tableau précédent

notations (rappel)

out sigle désignant un élément de l'information "taille" est ici constitué par :

ère lettre : T pour taille

ème lettre : T, D ou E pour totale, décimale ou entière

ème lettre : I ou E pour initiale ou évolutive

ndice : O_p pour opérande ou Res pour résultat

ous-indice : i pour désigner le $i^{ème}$ opérande

règles hiérarchiques : $O_{p1}, \ldots, O_{pn}$ constituent Res de type "structure"

Res est non numérique (structure)

$TTE_{Res} := TTE_{o_{p1}} * REP_{o_{p1}} + \ldots + TTE_{o_{p_n}} * REP_{o_{pn}}$

ote :

$REP_{o_{pi}}$ est le facteur de répétition de l'opérande o_{p_i} ;

n outre seuls les opérandes n'étant pas des redéfinitions d'autres opérandes nter viennent dans cette somme (il ne faut compter qu'une fois la place commune occupée par les synonymes)

règles hiérarchiques de type "fichier" $o_{p1}, \ldots, o_{pn}$ sont les articles du fichier Res

$TTE_{Res} := \sup_i TTE_{o_{pi}}$

règles de redéfinition : Op synonyme de Res

$WT := \sup(TTE_{o_p}, TTE_{Res})$

si $TTE_{Res} < WT$

alors $TTE_{Res} := WT$

si Res numérique

alors $TEE_{Res} := TTE_{Res} - TDE_{Res}$

finsi

finsi

si $TTE_{o_p} < WT$

alors $TTE_{op} := WT$;

si op numérique

alors $TEEop := TTEop - TDEop$

finsi

finsi

note :

pour être sûr de respecter la loi des synonymes, on modifie toujours la plus petite taille, normalement celle de Res, et on la rend égale à la plus grande.

règles d'addition : $o_{p1} + \ldots + o_{pn}$ dans Res

retenue := p+1 avec $10^p < n \leqslant 10^{p+1}$ et p entier $\geqslant -1$

$MAXTE := \sup_i TEE_{o_{pi}}$; $MAXTD := \sup_i TDE_{o_{pi}}$

$MAXVARE := \sup_i (TEE_{o_{pi}} - TEI_{o_{pi}})$; $MAXVARD := \sup_i (TDE_{o_{pi}} - TDI_{o_{pi}})$

si Res numérique

alors $TEE_{Res} := \sup(TEE_{Res}, \inf(MAXTE+retenue, TEI_{Res} + MAXVARE))$

$TDE_{Res} := \sup(TDE_{Res}, \inf(MAXTD, TDI_{Res} + MAXVARD))$

$TTE_{Res} := TEE_{Res} + TDE_{Res}$

règles d'addition (suite)

majoration à 18 :

si $TTE_{Res} > 18$

alors $TTE_{Res} := 18$

si $TEE_{Res} > 18$

alors $TEE_{Res} := 18$

finsi

$TDE_{Res} := 18 - TEE_{Res}$

finsi

sinon $TTE_{Res} := \sup(TTE_{Res}, \inf(MAXTE+retenue+MAXTD, TTI_{Res} + MAXVARE + MAXVARD))$

finsi

règles de soustraction : $Op_1 - (O_{p2} + \ldots + O_{pn})$ dans Res

si $\forall i$: Op_i non signé

alors retenue := P +1 avec $10^p < n-1 \leqslant 10^{p+1}$ et p entier $\geqslant -1$

sinon retenue := p+1 avec $10^p < n \leqslant 10^{p+1}$ et p entier $\geqslant -1$

finsi

suite des calculs analogue à ceux de l'addition

règles de multiplication et de division : $O_{p_1} \left\{ \begin{matrix} * \\ / \end{matrix} \right\} O_{p2}$ dans Res

si multiplication

alors $TEW := TEE_{op_1} + TEE_{o_{p_2}}$; $TDW := TDE_{o_{p1}} + TDE_{o_{p2}}$

$VEW := TEW - TEI_{o_{p1}} - TEI_{o_{p2}}$; $VDW := TDW - TDI_{o_{p1}} - TDI_{o_{p2}}$

sinon (division)

$TEW := TEE_{o_{p1}} + TDE_{o_{p2}}$; $TDW := TDE_{o_{p1}} + TEE_{o_{p2}}$

$VEW := TEW - TEI_{o_{p1}} - TDI_{o_{p2}}$; $VDW := TDW - TDI_{o_{p1}} - TEI_{o_{p2}}$

finsi

si Res numérique

alors $TEE_{Res} := \sup(TEE_{Res}, \inf(TEW, TEI_{Res} + VEW))$

$TDE_{Res} := \sup(TDE_{Res}, \inf(TDW, TDI_{Res} + VDW))$

$TTE_{Res} := TEE_{Res} + TDE_{Res}$; majoration à 18

sinon $TTE_{Res} := \sup(TTE_{Res}, \inf(TEW+TDW, TTI_{Res} + VEW + VDW))$

finsi

règles d'exponentiation : $Op_1 \uparrow Op_2$ dans Res

si Op_2 = constante
alors Ex:= partie entière (Op_2 + 0,5)
VEX:=0
sinon EX:= $10 \uparrow TEEop_2$
VEX:=EX-$10 \uparrow TEIop_2$
finsi
si Res numérique
alors TEE_{Res}:=sup(TEE_{Res}, inf(EX* $TEEop_1$, TEI_{Res} +EX* ($TEEop_1$ - $TEIop_1$) +VEX* $TEIop_1$))
TDE_{Res}:=sup(TDE_{Res}, inf(EX* $TDEop_1$, TDI_{Res} +VEX* ($TDEop_1$ - $TDIop_1$) +VEX*$TDIop_1$))
TTE_{Res}:=TEE_{Res}+TDE_{Res} ; majoration à 18
sinon TTE_{Res}:=sup(TTE_{Res}, inf(EX*($TTEop_1$, TTI_{Res} +EX*($TTEop_1$-$TTIop_1$) +VEX*$TTIop_1$))

règles de transfert

si Op et Res numérique
alors TEE_{Res}:= sup(TEE_{Res}, inf(TEEop, TEI_{Res}+(TEEop-TEIop))
TDE_{Res}:=sup(TDE_{Res}, inf(TDEop, TDI_{Res}+(TDEop-TDIop))
TTE_{Res}:=TEE_{Res}+TDE_{Res}
sinon TTE_{Res}:=sup(TTE_{Res}, inf(TTEop, TTI_{Res}+(TTEop-TTIop))
si Res numérique
alors TEE_{Res}:=TTE_{Res}-TDE_{Res}
finsi
finsi

remarque :
dans beaucoup de règles ; les formules de calcul de la taille entière, de la taille décimale et de la taille totale (si le résultat est non numérique) ne diffèrent que par la deuxième lettre (E, D et T) des sigles utilisés :
multiplication, exponentiation, transfert.

Tableau 3 :

évaluation des tailles inconnues :
exemples de formulation sur descripteurs de traitements.

hypothèse :
une charpente est complétée par les descripteurs "pères" INIT, ØPER, INTER et RESUL ceux-ci appelent, si nécessaire, des descripteurs "fils" adaptés à chaque type de noeuds d'influence :

- pour l'addition et la soustraction : figures A, B, C, D
- pour la multiplication : figures E et F

ØPERAS (figure B)

DESIGNATION DES ELEMENTS	Condition	indicatif ou occurrence	CALCUL	donnée	IDENTIFICATEUR ou RESULTAT	Appel
modification de CTE taille entière évolutive de l'op.			sup(CTE, TEEØ)		CTE TEEØ	
modification de CTD taille décimale évolutive de l'op.			sup(CTD, TDEØ)		CTD TDEØ	
modification de REPS repère de signe de l'opérande	SGNØ = 'S'		REPS + 1		REPS SGNØ	

DESIGNATION DES ELEMENTS	Condition	indicatif ou occurrence	CALCUL	donnée	IDENTIFICATEUR ou RESULTAT	Appel
cumul taille entière				0	CTE	
cumul taille décimale				0	CTD	
repère de signe				0	REPS	

RESULAS (figure D)

DESIGNATION DES ELEMENTS	condition	indicatif ou occurrence	CALCUL	donnée	IDENTIFICATEUR ou RESULTAT	Appel
résultat numérique	TDER numérique					
t. décimale évolutive du res.					TDER	
t. totale évolutive du résultat					TTER	
nouvelle taille entière			sup(CTE+RET, TEER)		+TEER	
nouvelle taille décimale	TDER = 0		0		+TDER	
" " "	TDER ≠ 0		sup(CTD, TDER)		+TDER	
majoration à 18	TTER > 18					MAJØRTAIL
résultat non numérique	TDER non numérique					
nouvelle taille totale			sup(CTE+CTD+RET, TTER)		TTER	

INTER (figure C)

DESIGNATION DES ELEMENTS	Condition	indicatif ou occurrence	CALCUL	donnée	IDENTIFICATEUR ou RESULTAT	Appel
retenue					RET	
soustraction à op. non signés	codus :'-' et reps = 0		CPTØ - 1		CPTØ	
code de l'influence					CØDUS	
1 opérande	CPTØ = 0				+CPTØ	
moins de 10 opérandes	CPTØ ≤ 10			1	+constante 1	
au delà de 10 opérandes	CPTØ > 10			2	+constante 2	

RESULMUL (figure F)

DESIGNATION DES ELEMENTS	condition	indicatif ou occurrence	CALCUL	donnée	IDENTIFICATEUR ou RESULTAT	APPEL
résultat numérique	TDER num.					
taille décimale évolutive du res					TDER	
taille totale					TTER	
taille entière			sup(TEER, TEEØ1+ TEEØ2)		+ TEER	
taille décimale					+ TDER	
entier	TDER = 0			0	TDER	
rationnel	TDER ≠ 0		sup(TDER, TDEØ1+ TDEØ2)		TDER	
majoration à 18	TTER > 18					MAJØRTAI L
résultat non numérique	TDER non num.					
nouvelle taille totale			sup(TTER, TTEØ1+ TTEØ2)		TTER	

- A 36 -

DESIGNATION DES ELEMENTS	Condition	indicatif ou occurrence	CALCUL	donnée	INDICATEUR ou RESULTAT	Appel
premier opérande	CPTØ =1					
sauvegarde des tailles évolutives						
taille totale			TTEØ		TTEØ1	
taille entière			TEEØ		TEEØ 1	
taille décimale			TDEØ		TDEØ 1	
deuxième opérande	CPTØ=2					
sauvegarde des tailles évolutives						
taille totale			TTEØ		TTEØ 2	
taille entière			TEEØ		TEEØ 2	
taille décimale			TDEØ		TDEØ 2	

- A 35 -

figure 4 :

modifications en cascade

(programmation des règles arithmétiques selon les principes du chapitre 2.3)

```
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
```

```
ARITHMETIQUE SECTION.
DEBUT. PERFORM INITAR.
    MOVE NUL TO CPTO.
BOUOP.
    PERFORM PREND-IDEN.
    IF FINUS = '*' GO TO ENTROPRES.
    IF FINUS = 'X' DISPLAY 'REGLE SANS RESULTAT' GO TO FIN.
    ADD UN CPTO.
    IF FINUS = 'I' READ TABID INVALID KEY DISPLAY 'ER ID',CLEW
    GO TO BOUOP.
    DISPLAY 'TAILLES OP.:',TAILLEW.
    IF TEEX NOT NUMERIC DISPLAY 'OPERANDE NON NUM' GO TO FIN.
    PERFORM OPREPAR.
    GO TO BOUOP.
ENTROPRES. PERFORM ENTRAR.
BOURES.
    PERFORM PREND-IDEN.
    IF FINUS = 'X' GO TO FIN.
    IF FINUS = '*' DISPLAY '2E DANS 1 REGLE' GO TO FIN.
    IF FINUS NOT = 'I' DISPLAY 'RESULTAT CONSTANT' GO TO BOURES.
    READ TABID INVALID KEY DISPLAY 'ER ID',CLEW GO TO BOURES.
    IF TEEX NOT NUMERIC PERFORM RNONNUMAR
    ELSE PERFORM RNUMAR.
    PERFORM TESTMOD GO TO BOURES.
FIN. EXIT.
```



```
DEBUT. MOVE ALL '0' TO TAILINT VARTAIL.
    MOVE NUL TO REPS.
OPERAR SECTION.
DEBUT. IF CODUS = '+' OR '-' GO TO OPAS.
    IF CPTO = UN OR CODUS = '*'
    ADD TEE TO TAILINTE VARTAILE SUBTRACT TEI FROM VARTAILE
    ADD TDE TO TAILINTD VARTAILD SUBTRACT TDI FROM VARTAILD
    ELSE
    ADD TDE TO TAILINTE VARTAILE SUBTRACT TDI FROM VARTAILE
    ADD TEE TO TAILINTD VARTAILD SUBTRACT TEI FROM VARTAILD.
    GO TO FIN.
OPAS.
    IF TEE > TAILINTE MOVE TEE TO TAILINTE.
    IF TDE > TAILINTD MOVE TDE TO TAILINTD.
    SUBTRACT TEI FROM TEE GIVING TEE1.
    SUBTRACT TDI FROM TDE GIVING TDE1.
    IF TEE1 > VARTAILE MOVE TEE1 TO VARTAILE.
    IF TDE1 > VARTAILD MOVE TDE1 TO VARTAILD.
    IF SGNI = 'S' ADD UN REPS.
FIN. EXIT.
ENTRAR SECTION.
DEBUT. IF CODUS NOT = '+' AND NOT = '-' GO TO FIN.
    IF CODUS = '-' AND REPS = NUL SUBTRACT UN FROM CPTO.
    IF CPTO = NUL NEXT SENTENCE
    ELSE IF CPTO NOT > DIX ADD UN TAILINTE
    ELSE ADD DEUX TAILINTE.
FIN. EXIT.
RNUMAR SECTION.
DEBUT.
    ADD TEI VARTAILE ADD TDI VARTAILD.
    IF VARTAILE < TAILINTE MOVE VARTAILE TO TAILINTE.
    IF VARTAILD < TAILINTD MOVE VARTAILD TO TAILINTD.
    IF TEE > TAILINTE MOVE TEE TO NVTAILE
    ELSE MOVE TAILINTE TO NVTAILE.
    IF TDE > TAILINTD MOVE TDE TO NVTAILD
    ELSE MOVE TAILINTD TO NVTAILD.
    IF TDE = 0 AND TDI = 0 MOVE NUL TO NVTAILD.
    COMPUTE NVTAILT = NVTAILE + NVTAILD.
RNONNUMAR SECTION.
DEBUT. MOVE ZONETAILE TO NVTAIL.
    ADD TTI VARTAILD VARTAILE GIVING VARTAILT.
    ADD TAILINTD TAILINTE GIVING TAILINTT.
    IF VARTAILT < TAILINTT MOVE VARTAILT TO TAILINTT.
    IF TAILINTT > TTE MOVE TAILINTT TO NVTAILT
    .
TESTMOD SECTION.
DEBUT.
    IF TEIX NUMERIC
    PERFORM MAX18 THRU F-MAX18.
    IF NVTAIL = TAILE GO TO FIN.
    IF NVTAILT < TTE GO TO FIN.
    IF TEIX NUMERIC MOVE NVTAIL TO TAILE
    ELSE MOVE NVTAILT TO TTE.
    MOVE 'B' TO RESULMOD
    REWRITE IDW INVALID KEY DISPLAY 'ERREUR PROG.'.
    GO TO FIN.
MAX18.
    IF NVTAILT NOT > 18 GO TO F-MAX18.
    IF NVTAILC NOT NUMERIC GO TO F-MAX18.
    MOVE 18 TO NVTAILT.
    IF NVTAILE > 18 MOVE 18 TO NVTAILE MOVE 0 TO NVTAILD
    ELSE SUBTRACT NVTAILE FROM 18 GIVING NVTAILD.
F-MAX18. EXIT.
FIN.
    DISPLAY 'RES.NB:',CLEW,'TAILLES:',TAILLEW.
PREND-IDEN SECTION.
DEBUT.
    MOVE '*' TO FINUS.
    PERFORM PROGIUS.
    MOVE ELUS (IUS) TO FINUS.
    MOVE ' ' TO ZONETAILI.
    IF FINUS = '*' OR 'S' MOVE '0' TO ZONETAILI GO TO FIN.
    IF FINUS = 'I' OR 'R' PERFORM PR-IDEN GO TO FIN.
    IF FINUS = 'C' PERFORM TR-LIB GO TO FIN.
    IF FINUS = 'N' PERFORM TR-NUM GO TO FIN.
    IF FINUS = ' ' GO TO DEBUT.
    DISPLAY 'ERREUR PG:'.
    GO TO FIN.
PROGIUS.
    ADD UN IUS. IF IUS > VALUSMAX GO TO FIN.
PROGID.
    MOVE ELUS (IUS) TO WI (IW).
PR-IDEN.
    PERFORM PROGIUS THRU PROGID VARYING IW FROM UN BY UN
    UNTIL IW = SIX. MOVE ZONETAILI TO CLEW.
    MOVE 0 TO SUITEW.
TR-LIB.
    MOVE '     F' TO ZONETAILI.
    PERFORM PROGIUS THRU PROGID VARYING IW FROM UN BY UN
    UNTIL IW = SIX. MOVE ZONETAILI TO ZONETAILE.
TR-NUM.
    PERFORM PROGIUS THRU PROGID VARYING IW FROM UN BY UN
    UNTIL IW = NV. MOVE ZONETAILI TO ZONETAILE.
FIN. DISPLAY '(OP.OU RES.)',ZONETAILI.
```

ANNEXE 4 : décodage Cobol

tableau 1 :
verbes Cobol :

différents formats,
traitements et noeuds d'influence correspondants

tableau 2 :
description des tailles
dans le texte source Cobol
(exemples)

TABLEAU 1

format d'écriture	calculs correspondants	(nombre de) noeuds d'influence
A1,A2,...,An	An:=An+(A1+A2+...+A_{n-1})	(1) n°+A1 A2...An $An%
A1,...,ApTØAp+1,...An	Ap+1:=Ap+1 Ap+2:=Ap+2 ⋮ An:=An } +(A_1+...+Ap)	(n-p) n°+A_1...ApAp+1$Ap+1% n°+$A_1$...ApAp+2$Ap+2% ⋮ n°+A1...ApAn$An%
A_1,...,A_p GIVING A_{p+1},...,An	An:=An-1:=...:=Ap+2:=Ap+1:= A_1+A_2+...+A_p	(1) n°+A1...Ap$Ap+1...An%
CORR A TØ B ons A_i et B_i les q ents en correspon- ce dans les uctures A et B	B_1:=B_1+A_1 B_2:=B_2+A_2 ⋮ Bq:=Bq+Aq	(q) n°+$A_1B_1$$B_1% n°+$A_2B_2$$B_2% ⋮ n°+AqBq$Bq%
TRACT		
TRACT A_1,...,Ap FRØM Ap+1,...,An	Ap+1:=Ap+1 Ap+2:=Ap+2 ⋮ An:=An } -(A_1+...+Ap)	(n-p) n°-A_1...ApAp+1$Ap+1% n°-$A_1$...ApAp+2$Ap+2% ⋮ n°-A_1...ApAn$An%
TRACT A_1,...,FRØM Ap+1 GIVING Ap+2,...,An	An:=An-1:=...:=Ap+2:= Ap+1-(A_1+A_2+...+Ap)	(1) n°-A_1...ApAp+1$Ap+2...An%
TRACT CORR A FROM B ons A_i et Bi les q ents en correspon- ce dans les struc- es A et B	B_1:=B_1-A_1 B_2:=B_2-A_2 ⋮ Bq:=Bq-Aq	(q) n°-$A_1B_1$$B_1% n°-$A_2B_2$$B_2% ⋮ n°-AqBq$Bq%
LTIPLY		
TIPLY A1 BY A2,...,An	A2:=A2 A3:=A3 } * A_1 An:=An	(n-1) n° * $A_1A_2$$A2% n° * $A_1A_3$$A3% n° * A_1An$An%
TIPLY A1 BY A2 GIVING A3,...An	An:=An-1:=...:=A4:=A3:= A_1 * A_2	(1) n° * $A_1A_2$$A_3$...$A_n$%

formats d'écriture	calculs correspondants	(nombre de) noeuds d'influence
DIVIDE		
1) DIVIDE A1 INTØ $A_2,\ldots,An$	$\left\{\begin{array}{l}A2:=A2\\A3:=A3\\An:=An\end{array}\right\}$ /A_1	(n-1) n°/$A_2A_1$$A_2% n°/$A_3A_1$$A_3% ⋮ n°/AnA1$An%
2) DIVIDE A_1 INTØ A_2 GIVING $A_3,\ldots,A_n$	$An:=An-1:=\ldots:=A_4:=A_3:=$ A_2 / A_1	(1) n°/$A_2A_1$$A_3\ldots A_n$%
3) DIVIDE A_1 BY A2 GIVING $A_3,\ldots,An$	$An:=An-1:=\ldots:=A_4:=A_3:=$ A_1 /A_2	(1) n°/$A_1A_2$$A_3\ldots An$%
CØMPUTE		
CØMPUTE $A_1,\ldots,An$ = expression arithmétique exemple : CØMPUTE X,Y = (A+B)* (C+D)-(E+F+G)	An:=An-1:=...:=A_1 = valeur de l'expression mem1:=A+B mem2:=C+D mem2:=mem1*mem2 x:=y:=mem2-(E+F+G)	(x) n°+AB$mem1% n°+CD$mem2% n°*mem1mem2$mem2% n°-EFG mem2$ x y %
MØVE		
1) MØVE A1 TØ A2,...,An	An:=An-1:=An-2...:=A2:=A1	(1) n° M A_1 $A_2\ldots A_n$%
2) MØVE CØRR A TØ B,C... soient A_i et B_i , $i\in I$ A_j et C_j , $j\in J$ ⋮ les éléments en correspondance dans les structures A et B, A et C, ⋮	$\forall i\in I$: B_i :=A_i $\forall j\in J$: C_j:=A_j ⋮	(x) n° M A_i B_i % $\forall i\in I$ n° M A_j C_j % $\forall j\in J$
WRITE/REWRITE/RELEASE		
{WRITE REWRITE RELEASE} A FRØM B...	A:=B (écriture ou réécriture de A)	(1) n° M B $ A %
READ / RETURN		
{READ RETURN} A INTØ B...	(lecture dans A) B:=A	(1) n° M A $ B %

TABLEAU 2

description Cobol	tailles (codées) totale	entière	décimale	signe	facteur de répétition
Données alphanumériques simples :					
PIC XXX	3	X	0		1
PIC X(20)	20	X	0		1
PIC X(2)A(3)	5	X	0		1
PIC X(3) ØCCURS 5	3	X	0		5
Données alphanumériques "édités" :					
PIC X(3)BX(2)0	7	X	0		1
Données numériques simples :					
PIC 999	3	3	0		1
PIC S9(3)	3	3	0	S	1
PIC 9(2)V99 ØCCURS 7	4	2	2		7
PIC 99P(3)	5	5	0		1
PIC PP9(8)	10	0	10		1
Données numériques condensées					
PIC S9(3) CØMP-3	2	3	0	S	1
PIC S9(3)V9(2) CØMP-3	3	3	2	S	1
Données numériques binaires					
entier : CØMP	4	B	0	S	1
CØMP ØCCURS 100	4	B	0	S	100
réel simple précision : CØMP-1	4	F	0	S	1
réel double précision : CØMP-2	8	F	0	S	1
Données numériques "éditées"					
PIC 999B	4	E3	0		1
PIC +(3)	3	E3	0	S	1
PIC Z(5).9(3)	9	E5	3		1
PIC *(8)9.99	11	E9	2		1
PIC +99,999,999.99	14	E8	2	S	1
etc					

ANNEXE 5

Correspondance entre la numérotation des objets de l'annexe 1 et celle des résultats de l'ordinateur

figure 1 : tableau de correspondance

figure 2 : graphe des influences "renuméroté"

figure 3 : graphe des noeuds d'influence "renuméroté"

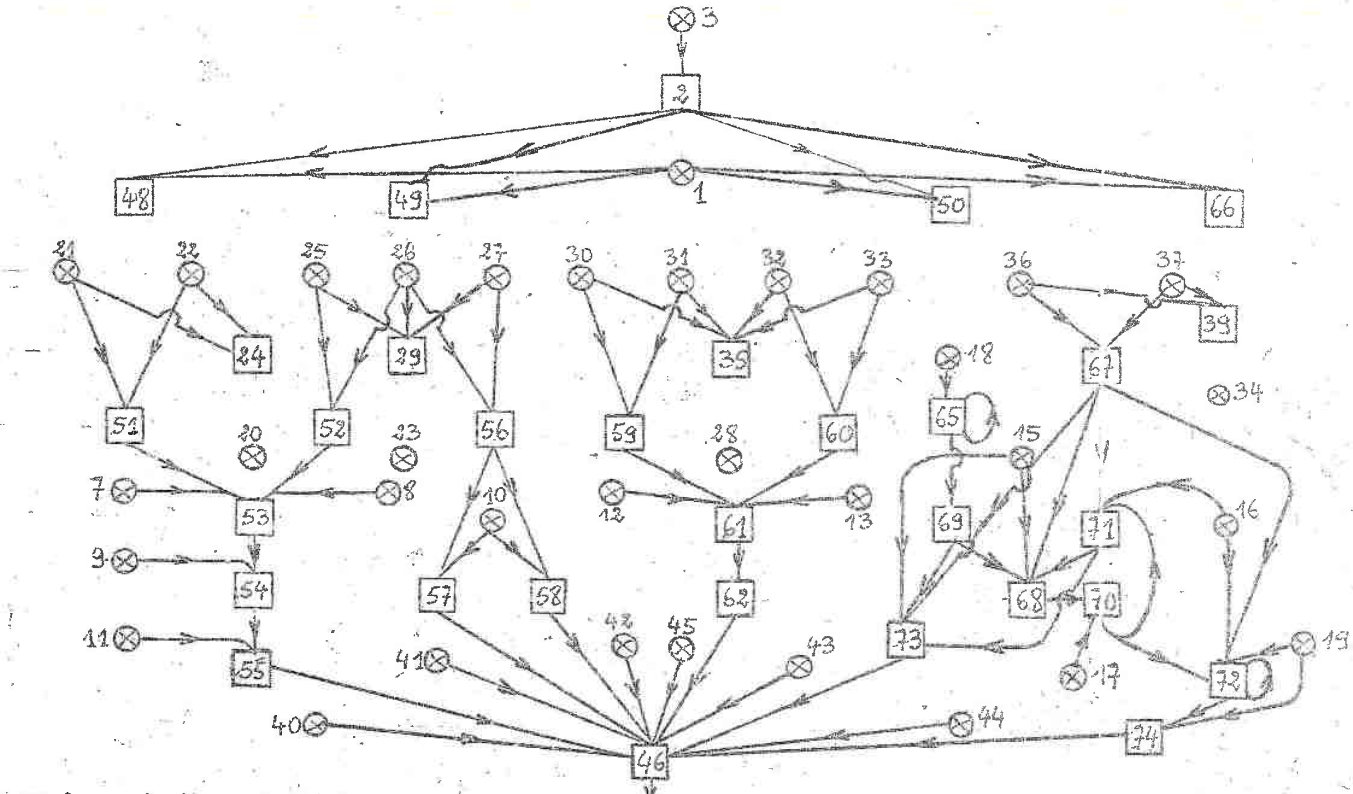
figure 1 a) tableau de correspondances pour les mots :

nom	numéro	numéro du noeud de définition
A	26	30
B	27	31
C	28	32
CARTES	1	1
CRTE	2	3
CRT1	18	20
CRT2	21	23
CRT3	25	28
CRT4	30	34
D	29	33
D1	19	21
D2	20	22
D3	22	25
D4	23	26
D5	24	27
ED	33	38
ED1	34	40
ED2	35	41
ED3	36	42
ED4	37	43
ED5	38	44
ED6	39	45
I	16	18
LGNE	4	6
LIGNES	3	4
M	31	36
M1	10	12
M2	11	13
N	32	37
P	13	15
Q	14	16
S	17	19
TRV	15	17
W1	5	7
W2	6	8
W3	7	9
W4	8	10
W6	9	11
X	12	14

b) tableau de correspondances pour les relations d'influences

numérotation manuelle	numérotation automatique
1	2
2	5
3	24
4	29
5	35
6	39
7	46
8	48
9	49
10	50
11	51

puis toujours un décalage de 40 dû aux 40 noeuds de définition qui ont été numérotés simultanément



igure 3 : graphe des noeuds d'influence
(numérotation des résultats
de l'ordinateur)

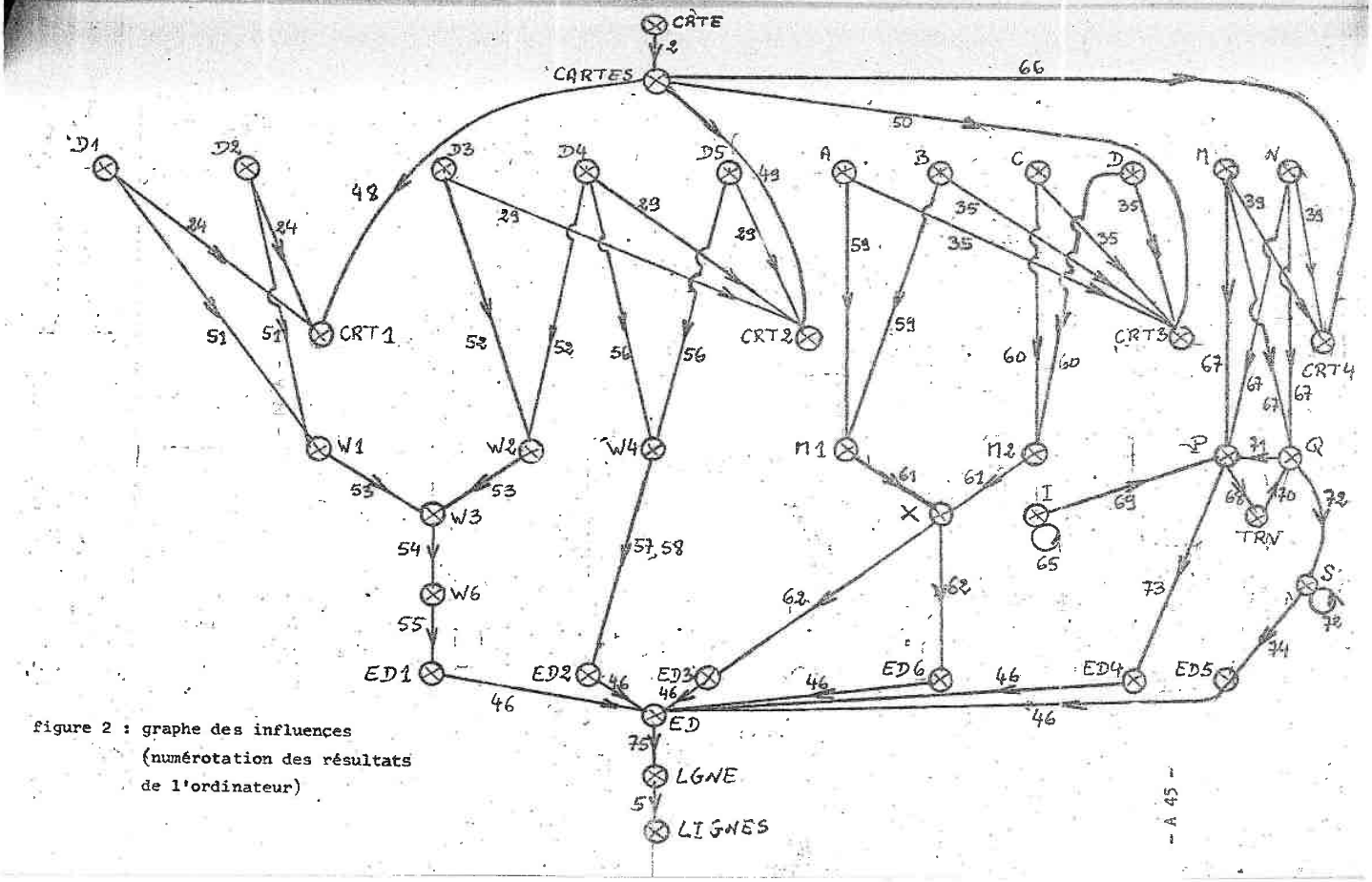


figure 2 : graphe des influences
(numérotation des résultats
de l'ordinateur)

REFERENCES BIBLIOGRAPHIQUES

Graphes

1 BERGE "Graphes et hypergraphes" - DUNOD - 1970

2 BERGE "Sur certains hypergraphes généralisant les graphes bipartis" - Combinatorial Theory and its applications - Balatonfüred - Amsterdam-Londres - 1970

3 BERGE "Hypergraphs generalizing bibartite graphs" - Integer and non linear programming - chap 26 (Ed. J .Abadie) Amsterdam - 1970

4 DERNIAME " Etude d'algorithmes pour les problèmes de cheminement dans les graphes finis" - thèse de 3è cycle - NANCY - 1966

5 DERNIAME-PAIR "Problèmes de cheminement dans les graphes" DUNOD - 1971

6 DESBAZEILLE "Exercices et problèmes de recherche opérationnelle" DUNOD - 1972

7 EMOND "Application de la notion de pile à des problèmes portant sur les chemins des graphes" - thèse de 3è cycle - NANCY - 1965

8 FAURE "Eléments de la recherche opérationnelle" - GAUTHIER-VILLARS - 1971

9 FØRD-FULKERSON "Flots dans les graphes" - GAUTHIER-VILLARS - 1967

10 KAUFMANN "Méthodes et modèles de la recherche opérationnelle" - DUNOD - 1968 - tomes 1 et 2

11 NASH-WILLIAMS "An application of matroids to Graphs théory"

12 ROY "Cheminement et connexité dans les graphes - Application aux problèmes d'ordonnancement" - METRA - série spéciale - 1962

13 ROY "Algèbre moderne et théorie des graphes" DUNOD - 1969 - tomes 1 et 2

14 STEEN "Algorithme de recherche d'un isomorphisme entre deux graphes" - thèse de 3è cycle - LILLE - 1968

2 - Langages

1 APL ARMINGAUD -
"Gestion des données en APL"
Ecole d'été d'informatique
Neuchâtel - 1972

2 CØBØL 65 CII sous Siris 7/Siris 8
manuel d'utilisation et manuel d'opération

3 CØDASYL "Report of the CODASYL Data Task Group"
(DBTG) ACM
New-York - April 1971

4 PL1 VEILLON - CAGNAT -
"Cours de programmation en langage PL/1"
ARMAND COLIN - Collection U - Tomes 1, 2 et 3

BERTHET -
"Le langage de programmation PL/1" -
DUNOD - 1972

5 SNØBØL JARAY - PISTRE
"Représentations particulières des chaînes de caractères, généralisation et application à l'implémentation de SNOBOL 4"
IUCA - NANCY

GRISWOLD-POAGE-POLONSKY
"SNOBOL 4 - The programming language" -
Prentice - Hall - INC.-Engelwood Cliffs
NEW JERSEY

- Analyse et programmation

1 - BAUVIN "L'informatique de Gestion"
Hommes et Techniques - 1968

2 - BELLEGARDE "Face - Langage d'écriture de Compilateurs"
thèse de 3è cycle - 1972 - NANCY

3 - BERGER "Recherche en informatique de gestion" -
Informatique et gestion n° 49
Juillet - Août 1973

4 - BERTINI - Tallineau "Une base pour l'optimisation : le Cobol structuré"
J I I A - 1974 -

5 - CØDASYL
(1) Report of the CODASYL Data Task Group (DBTG),
ACM, New York - 1971
(2) A survey of generalized Data Base Management Systems,
CODASYL Systems Comittee - 1969
(3) Feature Analysis of Generalized Data Base Management System,
CODASYL Systems Comittee Technical Report, ACM, New York - 1971
(4) Systèmes de gestion des bases de données : les recommandations du CODASYL -
01 Informatique mensuel - avril et mai 1973

6 - CROCUS "Principes de conception d'un système d'exploitation" -
DUNOD - 1974

7 - DKJISTRA "Notes on structured programming in structured programming" -
APIC Studies in DATA PROCESSING - 1972 -
ACADEMIC PRESS

8 - GADEFAIT "Un générateur automatique de fichiers de tests" -
01 informatique mensuel - juin 1972

9 - HARMANT - PORTMANN - VILLARD
"CLAMS : package de création et de mise à jour de fichier" - NANCY -
janvier 1973

.../...

10 - ISDOS Research Project

(1) TEICHROEW - CARLSON
"A Model of the System Building Process"
ISDOS Working paper n° 64 - 1973
(University of Michigan) :

(2) TEICHROEW - RATAJ - HERSHEY
"An introduction to computer - aided documentation of user requirements for computer based information processing systems"
ISDOS working paper n° 72 - 1973

11 - LE GALLOU "Informatique et téléinformatique" -
l'Informatèque - EME - 1974

12 - LE MOIGNE "Les systèmes d'information dans les organisations"
Presses Universitaires de France - 1973

13 - MALLET "La méthode informatique" -
HERMANN - 1971

14 - MAROLDT "Définition de face - Langage pour l'écriture des compilateurs" -
thèse de docteur ingénieur 1972 - NANCY

15 - PAIR Cours de compilation
Ecole d'été d'informatique - Neuchâtel - 1972

16 - PAIR Séminaires de recherche : "programmation déductive" - NANCY - 197

17 - PECCOUD Cours d'analyse
Ecole d'Eté d'informatique d'Alès - 1971 -

18 - REIX "L'analyse en informatique de gestion"
DUNOD - 1971 - tomes 1 et 2

19 - WARNIER "Les procédures de traitements et leurs données" -
Ed. d'organisation - 1973

20 - YVON-SEMIN "Comment concevoir un système intégré de gestion" -
Entreprise moderne d'édition 1970

21 - Colloque de Grenoble : "Les outils de l'analyse en informatique de gestion"
1971

22 - Colloque d'Orsay (IRIA) : "La recherche en informatique de gestion" - 1973

4 - Méthodes d'analyse

1 -	ARIANE	Gamma groupe Bossard
2 -	CORIG	C.G.I.
3 -	MINOS	C.E.G.O.S.
4 -	PAC	C.G.I.
5 -	PROTEE	S.I.S.

5 - Projet Remora

1 - LAMY "Descripteurs de traitements et générateur associé dans le projet Remora" -
thèse de 3è cycle - NANCY - à paraître

2 - ORY "Software d'aide à l'analyse et la programmation" -
thèse de 3è cycle - NANCY - 1972

3 - ROLLAND (1) "Engineering of management informatic systems"
INTERNET - 1974 -

(2) "Système d'édition dans le projet Remora"
BIGEN - 1974

(3) "Le projet Remora : un système de pilotage des projets informatiques"
INFORMATIQUE ET GESTION - 1974

4 - THIERY "Méthode de conception d'algorithmes programmés".
Thèse de 3è cycle - NANCY - à paraître

NOM DE L'ETUDIANT : PORTMANN M. Claude

NATURE DE LA THESE : Doctorat de Spécialité en Mathématiques Appliquées

VU, APPROUVE

& PERMIS D'IMPRIMER

NANCY, le 23 Septembre 1974

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

J.R. HELLUY