Projet SOCRATE - (2-3) Interprétation et compilation

Robert Morin

N° D'ORDRE

T H E S E

présentée à

LA FACULTE DES SCIENCES DE L'UNIVERSITE DE GRENOBLE

Pour Obtenir

LE GRADE DE DOCTEUR-INGENIEUR

par

Robert MORIN
Ingénieur I. M. A. G.

PROJET SOCRATE

(2-3) Interprétation et Compilation

Thèse soutenue le 22 Décembre 1970 devant la commission d'examen

Monsieur	J. KUNTZMANN	Président
Monsieur	L. BOLLIET	Examinateurs
Monsieur	N. GASTINEL	
Monsieur	J.C. BOUSSARD	
Monsieur	F. SALLE	

L I S T E D E S P R O F E S S E U R S

Doyen honoraire : Monsieur M. MORET
Doyen : Monsieur E. BONNIER

PROFESSEURS TITULAIRES

MM.	NEEL Louis	Physique Expérimentale
	KRAVTCHENKO Julien	Mécanique Rationnelle
	CHABAUTY Claude	Calcul différentiel et intégral
	BENOIT Jean	Radioélectricité
	CHENE Marcel	Chimie Papetière
	FELICI Noël	Electrostatique
	KUNTZMANN Jean	Mathématiques Appliquées
	BARBIER Reynold	Géologie Appliquée
	SANTON Lucien	Mécanique des Fluides
	OZENDA Paul	Botanique
	FALLOT Maurice	Physique Industrielle
	KOSZUL Jean-Louis	Mathématiques
	GALVANI Octave	Mathématiques
	MOUSSA André	Chimie Nucléaire
	TRAYNARD Philippe	Chimie Générale
	SOUTIF Michel	Physique Générale
	CRAYA Antoine	Hydrodynamique
	REULOS René	Théorie des Champs
	BESSON Jean	Chimie Minérale
	AYANT Yves	Physique Approfondie
	GALLISSOT François	Mathématiques
Melle.	LUTZ Elisabeth	Mathématiques
MM.	BLAMBERT Maurice	Mathématiques
	BOUCHEZ Robert	Physique Nucléaire
	LLIBOUTRY Louis	Géophysique
	MICHEL Robert	Minéralogie et pétrographie
	BONNIER Etienne	Electrochimie et Electrométallurgie
	DESSAUX Georges	Physiologie animale
	PILLET Emile	Physique Industrielle-Electrotechnique
	YOCCOZ Jean	Physique Nucléaire théorique
	DEBELMAS Jacques	Géologie Générale
	GERBER Robert	Mathématiques
	PAUTHENET René	Electrotechnique
	MALGRANGE Bernard	Mathématiques Pures
	VAUQUOIS Bernard	Calcul Electronique
	BARJON Robert	Physique Nucléaire

MM.	BARBIER Jean-Claude	Physique
	SILBER Robert	Mécanique des Fluides
	BUYLE-BODIN Maurice	Electronique
	DREYFUS Bernard	Thermodynamique
	KLEIN Joseph	Mathématiques
	VAILLANT François	Zoologie et Hydrobiologie
	ARNAUD Paul	Chimie
	SENGEL Philippe	Zoologie
	BARNOUD Fernand	Biosynthèse de la Cellulose
	BRISSONNEAU Pierre	Physique
	GAGNAIRE Didier	Chimie Physique
Mme.	KOFLER Lucie	Botanique
MM.	DEGRANGE Charles	Zoologie
	PEBAY-PEROULA Jean-Claude	Physique
	RASSAT André	Chimie Systématique
	DUCROS Pierre	Cristallographie Physique
	DODU Jacques	Mécanique Appliquée I. U. T.
	ANGLES D'AURIAC Paul	Mécanique des Fluides
	LACAZE Albert	Thermodynamique
	GASTINEL Noël	Analyse numérique
	GIRAUD Pierre	Géologie
	PERRET René	Servo-mécanisme
	PAYAN Jean-Jacques	Mathématiques Pures

PROFESSEURS SANS CHAIRE

MM.	GIDON Paul	Géologie
Mme.	BARBIER Marie-Jeanne	Electrochimie
Mme.	SOUTIF Jeanne	Physique
	COHEN Joseph	Electrotechnique
	DEPASSEL R.	Mécanique des Fluides
	GLENAT René	Chimie
	BARRA Jean	Mathématiques Appliquées
	COUMES André	Electronique
	PERRIAUX Jacques	Géologie et Minéralogie
	ROBERT André	Chimie Papetière
	BIARREZ Jean	Mécanique Physique
	BONNET Georges	Electronique
	CAUQUIS Georges	Chimie Générale
	BONNETAIN Lucien	Chimie Minérale
	DEPOMIER Pierre	Physique Nucléaire-Génie Atomique
	HACQUES Gérard	Calcul numérique
	POLOUJADOFF Michel	Electrotechnique
Mme.	KAHANE Josette	Physique
Mme.	BONNIER Jane	Chimie
MM.	VALENTIN Jacques	Physique
	REBECQ Jacques	Biologie
	DEPORTES Charles	Chimie
	SARROT-REYNAULD Jean	Géologie
	BERTRANDIAS Jean-Paul	Mathématiques Appliquées
	AUBERT Guy	Physique

PROFESSEURS ASSOCIES

MM.	RODRIGUES Alexandre	Mathématiques Pures
	MORITA Susumu	Physique Nucléaire
	RADHAKRISHNA	Thermodynamique

MAITRES DE CONFERENCES

MM.	LANCIA Roland	Physique Atomique
Mme.	BOUCHE Liane	Mathématiques
MM.	KAHANE André	Physique Générale
	DOLIQUE Jean Michel	Electronique
	BRIERE Georges	Physique
	DESRE Georges	Chimie
	LAJZEHOWICZ Joseph	Physique
	LAURENT Pierre	Mathématiques Appliquées
Mme.	BERTRANDIAS Françoise	Mathématiques Pures
MM.	LONGEQUEUE Jean-Pierre	Physique
	SOHM Jean-Claude	Electrochimie
	ZADWORNY François	Electronique
	DURAND Francis	Chimie Physique
	CARLIER Georges	Biologie végétale
	PFISTER Jean-Claude	Physique
	CHIBON Pierre	Biologie animale
	IDELMAN Simon	Physiologie animale
	BLOCH Daniel	Electrotechnique I. P.
	MARTIN-BOUYER Michel	Chimie (C. S. U. Chambéry)
	SIBILLE Robert	Construction mécanique (I. U. T.)
	BRUGEL Lucien	Energétique I. U. T.
	BOUVARD Maurice	HYdrologie
	RICHARD Lucien	Botanique
	PELMONT Jean	Physiologie animale
	BOUSSARD Jean-Claude	Mathématiques Appliquées (I. P. G.)
	MOREAU René	Hydraulique I. P. G.
	ARMAND Yves	Chimie I. U. T.
	BOLLIET Louis	Informatique I. U. T.
	KUHN Gérard	Energétique I. U. T.
	PEFFEN René	Chimie I. U. T.
	GERMAIN Jean-Pierre	Mécanique
	JOLY Jean-René	Mathématiques Pures
Melle.	PIERY Yvette	Biologie animale
	BERNARD Alain	Mathématiques Pures
	MOHSEN Tahsin	Biologie (C. S. U. Chambéry)
	CONTE René	Mesures Physiques I. U. T.
	LE JUNTER Noël	Génie Electrique Electronique I. U. T.
	LE ROY Philippe	Génie Mécanique I. U. T.
	ROMIER Guy	Techniques Statistiques quantitatives I. U. T.
	VIALON Pierre	Géologie
	BENZAKEN Claude	Mathématiques Appliquées
	MAYNARD Roger	Physique

MM.	DUSSAUD René	Mathématiques (C. S. U. Chambéry)
	BELORIZKY Elie	Physique (C. S. U. Chambéry)
Mme.	LAJZEROWICZ Jeannine	Physique (C. S. U. Chambéry)
M.	JULLIEN Pierre	Mathématiques Pures
Mme.	RINAUDO Marguerite	Chimie
MM.	BLIMAN Samuel	E. I. E.
	BEGUIN Claude	Chimie Organique
	NEGRE Robert	I. U. T.

MAITRES DE CONFERENCES ASSOCIES

MM.	YAMADA Osamu	Physique du Solide
	NAGAO Makoto	Mathématiques Appliquées
	MAREZIO Massimo	Physique du Solide
	CHEECKE John	Thermodynamique
	BOUDOURIS Georges	Radioélectricité
	ROZMARIN Georges	Chimie Papetière

Je tiens à remercier en premier lieu, puisqu'il s'agit d'une thèse de groupe, Monsieur J.R. ABRIAL qui fut l'initiateur du projet SOCRATE et qui en reste l'âme et le moteur,

Messieurs J. BAS, G. BEAUME, G. HENNERON, D. TISSEAU, G. VIGLIANO qui formaient l'équipe si sympathique et si brillante du projet SOCRATE.

Je voudrais maintenant remercier,

Monsieur le Professeur J. KUNTZMANN, qui a bien voulu me faire l'honneur de présider le jury,

Monsieur le Professeur L. BOLLIET, qui a toujours montré beaucoup d'intérêt pour nos travaux,

Monsieur le Professeur N. GASTINEL, qui a accepté de faire partie du jury,

Monsieur le Professeur J.C. BOUSSARD qui a accepté de le juger,

Monsieur F. SALLE dont nous avons pu apprécier le compétence en ce domaine et qui a bien voulu se déplacer pour faire partie du jury,

et ceux, ou plutôt celle qui a contribué à la réalisation pratique de ce papier.

R.M.

L'exposé qui suit est un complément aux spécifications générales du projet SOCRATE.

Les définitions données dans les spécifications générales sont supposées connues.

Les références à cet ouvrage sont notées [1].

Les autres références du projet SOCRATE sont notées :

[2.1.] *Langage de Requêtes*

[2.2.] *Gestion des mémoires*

[2.3.] *Interprétation et Compilation.*

L'ensemble de ces quatre ouvrages constitue une thèse de groupe.

oOoOoOoOoOoOoOoOo

S O M M A I R E

1 - RAPPEL SUR LA TECHNIQUE D'INTERPRETATION ---------- 1

2 - METHODOLOGIE UTILISEE ---------- 2

3 - L'INTERPRETEUR COMME GENERATEUR DE CODE ---------- 4

Définitions ---------- 4

Détermination des BI ---------- 5

Propriétés de C ---------- 6

BI associée à un appel récursif ---------- 7

BI associée au retour ---------- 8

BI associée aux variables statiques ---------- 8

4 - EXEMPLES ---------- 8

Algorithme de I ---------- 9

Liste des BI ---------- 10

Algorithme de C ---------- 10

5 - CARACTERISTIQUES DE I_{QS} ---------- 12

Taille de I_{QS} ---------- 12

Performances des I_{QS} ---------- 12

6 - METHODE DE SIMULATION ---------- 14

7 - CONCLUSION ---------- 15

ANNEXE 1 ---------- 16

- Organigramme de (I) ---------- 17

- Organigramme de(I_{QS}) ---------- 22

ANNEXE 2 ---------- 24

I. RAPPEL SUR LA TECHNIQUE D'INTERPRETATION

La technique d'interprétation utilisée dans le projet SOCRATE a déjà été décrite ([1] § 5.2.1. et §.5.2.2.). Nous aborderons ici cette technique sous un jour nouveau.

Rappelons que l'interpréteur (I) est un programme général qui est capable à l'aide d'une question (Q) et d'une structure interne (S) de balayer un fichier (F) pour donner une réponse (R).

Soit : R = I(Q,S,F)

De par cette généralité, le programme I est assez complexe, en effet il doit envisager toutes les possibilités. Si nous classons toutes les instructions qui le composent par rapport aux trois éléments (Q,S,F), nous pouvons déterminer six catégories d'instructions.

a) Instructions conditionnelles portant sur la structure interne.

Par exemple : Quel est le type d'un objet, est-ce une entité ou une caractéristique ?

b) Instructions conditionnelles portant sur la question.

Par exemple : Y a-t-il un filtre, si oui est-il introduit par AYANT ou TELQUE ?

c) Instructions inconditionnelles qui permettent d'explorer la question.

Par exemple : BV = BV - 1, si BV est l'indice de parcours sur la question.

./.

d) Instructions inconditionnelles sur la structure interne qui sont en général des instructions de lecture pour initialiser des variables de travail de (I).

e) Instructions conditionnelles portant sur le fichier.

Par exemple : une classe d'entités est-elle vide ?

f) Instructions inconditionnelles qui travaillent sur le fichier.

Par exemple : les instructions qui décodent la chaîne de bits associée à une classe d'entités.

Les instructions de type a), b), c), d) peuvent être évaluées ou effectuées, indépendamment du fichier pour une question donnée, sur une structure donnée. On a alors, construit un interpréteur spécialisé I_{QS} équivalent à I dans un contexte donné. Il est évident que le programme I_{QS} est beaucoup plus performant que I, puisqu'il minimise le nombre d'instructions à exécuter. D'où l'idée d'écrire un programme C, qui n'est rien d'autre qu'un compilateur, pour générer I_{QS}.

$$C(Q,S) \rightarrow I_{QS}$$

2. METHODOLOGIE UTILISEE

A ce stade de recherche, il était possible d'envisager l'écriture d'un compilateur, en utilisant les techniques classiques de compilation. Cette solution n'a pas été retenue essentiellement pour 2 raisons.

./.

Un compilateur (Algol ou PL/1 par exemple) produit à partir d'un programme source généralement long un programme objet, ceci d'une seule façon possible. En effet, l'ensemble des règles syntaxiques et l'ensemble des règles sémantiques du langage sont des constantes pour le compilateur. Ici le problème est différent. Le programme source est un programme très court puisqu'il s'agit d'une question, mais auquel on peut faire correspondre plusieurs programmes objets, en effet la structure du fichier qui est une grammaire particulière, est un paramètre du compilateur.

Par exemple, sur les deux structures suivantes :

```
ENTITE PERSONNE                    ENTITE PERSONNE
        ENTITE VOITURE                     VOITURE REFERENCE VOITURE
                NUMERO             ENTITE VOITURE
                                           NUMERO
```

On peut poser une question formellement identique :

NUMERO DE VOITURE DE PERSONNE ?

Sur cette question on aura deux interprétations différentes.

Il nous faut donc un compilateur qui utilise à la fois la grammaire du langage et la grammaire du fichier. Ceci nous semble présenter quelques difficultés techniques.

La deuxième raison est une raison qui relèverait plutôt de l'architecture des systèmes. La solution précédente suppose l'écriture d'un programme neuf et donc l'abandon définitif de l'interpréteur, qui, bien que moins performant, est opérationnel. Nous avons préféré construire à partir de l'acquis, c'est-à-dire construire un compilateur directement issu de l'interpréteur. C'est cette méthode que nous allons essayer de décrire.

./.

3. L'INTERPRETEUR COMME GENERATEUR DE CODE

DEFINITION

Nous appellerons boîte d'instructions (BI) une séquence d'instructions de I du type e) ou f) décrites au paragraphe 1.

Par exemple :

```
if NBR = 0 then dø ;        /* NBR nombre de réalisations d'une classe
   RET = 1 ;                                              d'entités */
   goto FINCIT ;
                end ;
```

Si à la place de l'instruction RET = 1 ; nous avions l'instruction BV = BV-1 où BV est l'indice de parcours sur la forme normalisée associée à la question, la suite d'instructions n'est plus une BI.

Le programme I se compose entre autres d'un certain nombre de BI et pour Q et S données il n'exécute que certaines BI. L'assemblage de ces BI constitue I_{QS}. Le problème se décompose donc en deux sous-problèmes :

- isoler toutes les BI contenues dans I,
- savoir assembler les BI nécessaires à l'évaluation d'une question.

D'autre part, dans un souci de simplification au niveau de l'assemblage des BI, il est nécessaire que les BI ne possèdent pas entre elles d'autres relations que la possession de variables communes. Nous verrons que cette règle souffre cependant quelques exceptions.

Nous appellerons C le programme d'assemblage des BI. C ne peut se composer que d'instructions des 4 premiers types. Nous verrons que pour la plupart ces instructions sont issues de I.

DETERMINATION DES BI

La détermination des BI est très simple. Il suffit de grouper ensemble les instructions de type e) ou f) tant qu'on ne rencontre pas d'instructions d'autres types. Il n'est pas toujours possible de procéder de cette façon dans le cas où une instruction de type e) introduit des instructions de type a), b), c), d). Deux cas peuvent alors se présenter :

1) soit une séquence possible de I :

```
if NBR = 0 then BV = BV-1 ;
else BV = BV-2 ;
```

On ne peut alors placer ces instructions ni dans une BI, ni dans C ce qui met en échec la méthode. Il est nécessaire alors de modifier I pour éliminer cette situation, ce qui a toujours été possible dans notre cas particulier, mais ne l'est pas nécessairement dans le cas général.

2) par contre dans l'exemple suivant :

```
if NBR = 0 then dø ;
     if BV = 1 then goto A ;
     else goto B ;
                  end ;
else dø ;
  if BV = 1 then goto B ;
  else goto A ;
     end ;
```

l'instruction de type e) n'introduit qu'<u>apparemment</u> les instructions de type b) puisqu'on peut écrire la séquence en sens inverse.

```
if BV = 1 then dø ;
    if NBR = 0 then goto A ;
    else goto B ;
                end ;
else dø ;
    if NBR = 0 then goto B ;
    else goto A ;
     end ;
```

Nous pouvons alors déterminer facilement deux (BI).

PROPRIETES DE C

* C doit avoir le même comportement que I vis-à-vis d'une question, il comporte donc toutes les instructions de I indépendantes du fichier.

* On remplace chaque BI de I, par une instruction "d'empilement" des BI. Ces empilements successifs formeront finalement le programme généré I_{QS}. Avant d'empiler une boîte il faudra éventuellement évaluer des constantes qui dépendent de S ou de Q, en effet, on ne peut trouver dans une BI, une variable liée à Q ou à S.

Exemple :

La BI :

K4 = K4 + M(BROU(BV)) + NI
où NI, BROU(BV) dépendent de Q
et M de S

sera remplacée par

$$K4 = K4 + C^{ste}$$

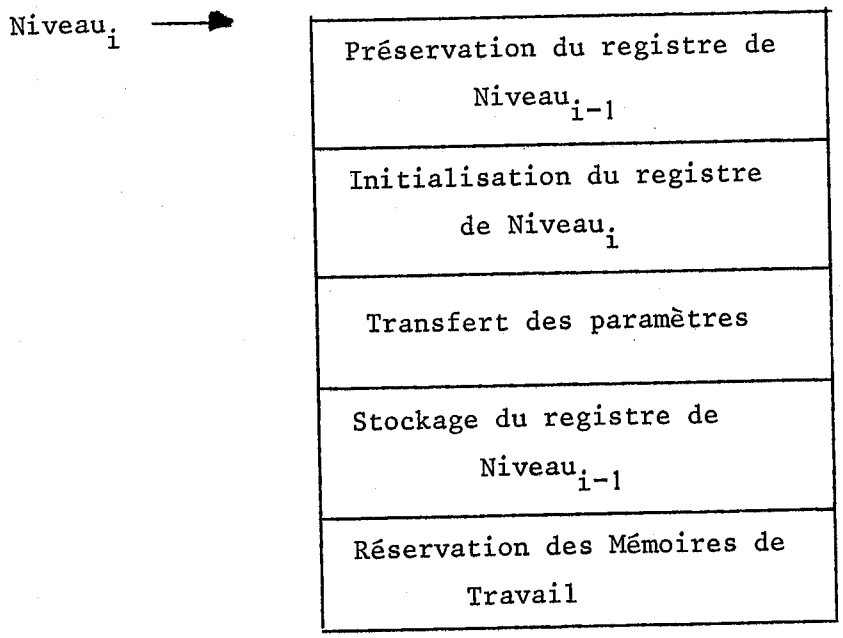
/.

* Certaines BI sont étiquetées. On note dans C leurs adresses d'empilement, pour que C puisse placer les adresses correctes dans les BI qui font références à ces étiquettes.

* Le programme (I) étant récursif, (C) l'est aussi, mais (I_{QS}) n'est pas récursif ceci dans un but de simplification et d'optimisation des performances. A chaque appel récursif de C, C empilera une BI spéciale que nous allons décrire, de même qu'à chaque retour de procédure.

BI ASSOCIEE A UN APPEL RECURSIF

Cette BI permet d'initialiser un registre de base qui sera utilisé dans toutes les BI de même niveau, de transmettre les paramètres du niveau supérieur, de stocker le registre de base du niveau supérieur, de réserver une zone de mémoire pour les variables de travail.



Cette BI sera notée dans les exemples suivants BI_{CALL}.

./.

BI ASSOCIEE AU RETOUR

Cette BI permet de récupérer le registre de base du $Niveau_{i-1}$ Ceci est possible puisqu'il a été stocké dans une variable de travail du $Niveau_i$, il est donc accessible par le registre de base dont on dispose au $Niveau_i$.

BI ASSOCIEE AUX VARIABLES STATIQUES

Toutes les variables statiques utilisées dans l'interpréteur, sont rassemblées dans une BI placée en tête du programme généré (I_{QS}). Cette BI possède un registre de base spécial ce qui permet d'adresser toutes les variables statiques à l'aide de ce registre.

4. EXEMPLES

Nous allons donner sur une grammaire réduite, l'exemple d'un interpréteur (I) puis la détermination des (BI) et la construction de (C). Soit une grammaire de citation réduite :

<CIT> → <IDENTIF> <ENTITE>
<ENTITE> → DE UN <IDENTIF> <ENTITE>
DE TOUT <IDENTIF> <ENTITE>
Ø
<IDENTIF> Chaîne alphanumérique

On peut générer avec cette grammaire des questions telles que :

NUMERO DE UN PIECE DE TOUTE VOITURE DE UN PERSONNE ?

Supposons que la citation ait été mise sous forme normalisée.(I) interprète cette forme.

./.

ALGORITHME DE I

```
Iréduit : procédure (X2) récursive ;
    /* X2 adresse de l'entité mère */
    if BV = 1 then dø ;
            /* écrire la caractéristique fille de X2 */
            goto fin ;
                end ;
    /* calcul de NBR nombre d'entités */
      I = 0 ;
      BW = BV ;
BOUCLE : I = I + 1 ;
        if I > NBR then goto fin ;
                /* calcul de ADF adresse i^ème entité fille de X2 */
        BV = BW ;
        BV = BV - 2 ;
        CALL Iréduit (ADF) ;
        if QU = '∃' then goto fin ;
        else goto boucle ;
fin :
end Iréduit ;
```

Les commentaires notés dans le programme remplacent des séquences d'instructions. BV est l'indice de parcours sur la forme normalisée, il est placé au départ sur la fin de la citation. BV = 1 est le test de début de citation. QU est le quantificateur d'une entité.

./.

LISTE DES BI

BI_1 → /* écrire la caractéristique fille de X2 */

BI_2 → /* calcul de NBR nombre d'entités */
I = 0

BI_3 → boucle : I = I + 1
if I > NBR then goto fin ;
/* calcul de ADF adresse $i^{ème}$ entité fille de X2 */

BI_4 → goto fin ;

BI_5 → goto boucle ;

On dispose de plus des BI_{CALL} et BI_{RETOUR}.

Il est alors possible d'écrire C.

ALGORITHME DE C

```
Créduit : procédure récursive ;
    if BV = 1 then dø ;
                /* empiler BI1 */
               goto fin ;
                    end ;
                /* empiler BI2 */
                /* empiler BI3 */    /* noter l'adresse de BI3 */
       BV = BV - 2 ;
                /* empiler BICALL */
       CALL  Créduit ;
     if QU = '∃' then  /* empiler BI4 */   /* noter l'adresse de BI4 */
     else /* empiler BI5 */    /* charger dans BI5 l'adresse de BI3 */
fin ;  /* empiler BIRETOUR */
       /* charger dans BI3 et BI4 l'adresse de BIRETOUR */
end Créduit ;                                                    ./.
```

Nous voyons sur cet exemple que (C) se déduit de façon systématique de (I). Le paramètre formel de (I) disparaît puisque c'est un 'paramètre fichier'.

Les seules instructions qu'il est nécessaire d'ajouter sont les instructions qui créent des liaisons entre (BI).

Soit la question :

NUMERO DE UN VOITURE DE TOUTE PERSONNE ?

Nous allons voir sur cet exemple ce que génère (C).

1	BI_{CALL}	
2	BI_2	
3	BI_3	13
4	BI_{CALL}	
5	BI_2	
6	BI_3	11
7	BI_{CALL}	
8	BI_1	
9	BI_{RETOUR}	
10	BI_4	11
11	BI_{RETOUR}	
12	BI_5	3
13	BI_{RETOUR}	

Nous avons noté dans la partie droite les adresses de liaison entre BI.

5. CARACTERISTIQUES DE I_{QS}

• Taille de I_{QS}

Le programme I_{QS} est entièrement développé, puisqu'on a supprimé la récursivité. Sa taille est donc fonction de la longueur de la question. Connaissant la taille de chaque (BI) il est facile de déterminer la taille de I_{QS}. Nous donnons en annexe la liste des (BI) avec leur taille respective.

Pour une question du type :

NOM DE TOUTE PERSONNE AYANT EXISTE PROFESSION ; ?

on obtient une taille d'environ 600 octets.

Cette taille risque cependant d'être prohibitive même si les (BI) sont optimisées au mieux, par exemple dans le cas où on filtre une entité à l'aide de nombreux critères. Il faut environ 80 octets par critère. Ceci nous impose soit une limitation dans la longueur des questions (limitation qui n'existe pas avec (I)), soit l'utilisation de (S) au niveau de I_{QS} (difficulté au niveau de la multiprogrammation), soit le partage des (BI) par plusieurs tâches (cf.3).

• Performances des I_{QS}

Nous avons cherché à comparer pratiquement les performances de (I) et de (I_{QS}). Le programme (C) n'étant pas encore opérationnel, nous n'avons encore que des comparaisons fragmentaires obtenues en simulant (I_{QS}).

Sur une question où on filtre une entité par des critères simples séparés par des OU nous avons obtenu les gains de temps suivants :

./.

Nombre de critères	1	2	3
% gain temps	30 %	42 %	49 %

On constate une amélioration du gain en fonction de la complexité de la question, ce qui était prévisible.

Nous avons aussi cherché à comparer théoriquement les performances des deux techniques, ceci en calculant le nombre de tests effectués dans les deux méthodes.

Les résultats sont les suivants :

- pour la question simple : NOM DE TOUTE PERSONNE ?

si NBR est le nombre des personnes du fichier

avec (I) $N_{test} = 16\ NBR$

(I_{QS}) $N_{test} = 3\ NBR$

- pour : NOM DE TOUTE PERSONNE AYANT EXISTE PROFESSION ?

Il y a deux paramètres NBR et P nombre de personnes qui vérifient le filtre.

(I) $N_{test} = 20\ NBR + 13\ P$

(I_{QS}) $N_{test} = 4\ NBR + 2\ P$

pour P = 0 le rapport est de 5, pour P = NBR le rapport est de 5,5.

Ces résultats sont évidemment insuffisants pour qu'on puisse en tirer des conclusions définitives. Une étude complète sur ce sujet est en

cours de réalisation. Il s'agit de savoir évaluer à priori le temps d'exécution d'une question, en fonction des caractéristiques formelles de la requête et des caractéristiques du fichier, ces dernières caractéristiques étant connues soit exactement, soit statistiquement. Cette évaluation serait intéressante à plusieurs titres :

- optimisation de l'interprétation donc du code généré,
- calcul à priori du coût d'une requête.

6. METHODE DE SIMULATION

Pour pouvoir tester facilement (C) et (I_{QS}) nous avons utilisé PL/1 au maximum.

En effet, on peut considérer que chaque (BI) est une chaîne de bits au sens PL/1. Nous pouvons donc définir ces (BI) écrites en code comme des constantes du programme (C).

Toutes les opérations d'empilement des (BI) se ramènent à des opérations de concaténation, les opérations d'évaluation des paramètres des (BI) à des opérations d'union ou d'intersection sur des chaînes de bits. (I_{QS}) étant alors totalement construit il suffit de passer le contrôle à cette chaîne de bits.

./.

7. CONCLUSION

Les relations qui existent entre (I) et (C) permettent d'envisager l'automatisation du passage de l'un à l'autre. On peut alors concevoir l'écriture d'un programme "compilateur de compilateur" qui accepterait (I) comme entrée et génèrerait (C). Dans l'hypothèse où (I) ne comporte pas de "conditions fichier" devant des "instructions question ou structure" cela ne semble pas présenter de difficultés.

Cette méthode aurait l'intérêt de faciliter les modifications du langage, car on modifierait d'abord (I) pour tester le nouveau langage et on aurait ensuite le programme (C) de façon automatique.

En dehors de l'amélioration des performances obtenue grâce à (C), nous voyons un autre avantage à cette technique par rapport à celle de (I), c'est la possibilité d'implémenter à peu de frais le système sur des machines différentes. En effet (C) peut être écrit en langage évolué sans pour cela altérer beaucoup les performances du système, donc sans nécessiter une reprogrammation, seules les (BI) doivent être programmées en code, or elles ne comportent que peu d'instructions ...

A N N E X E 1

Nous donnons l'organigramme de (I). Il s'agit de l'interpréteur du projet SOCRATE, correspondant à la grammaire complète du langage de citation, mais réduit à la requête d'interrogation. On remarque que les conditions fichier (entourées d'un cercle gras) n'introduisent jamais d'autres conditions, si ce n'est la condition "JJ > NBR", correspondant à la fin d'une boucle sur une classe d'entités, ce qui permet facilement le passage à (C). Cette restriction introduit quelque lourdeur, par exemple à la page 18 de l'organigramme la condition "KK = 1" est répétée cinq fois alors qu'on pourrait ne l'avoir qu'une fois.

* Notes sur les conditions :

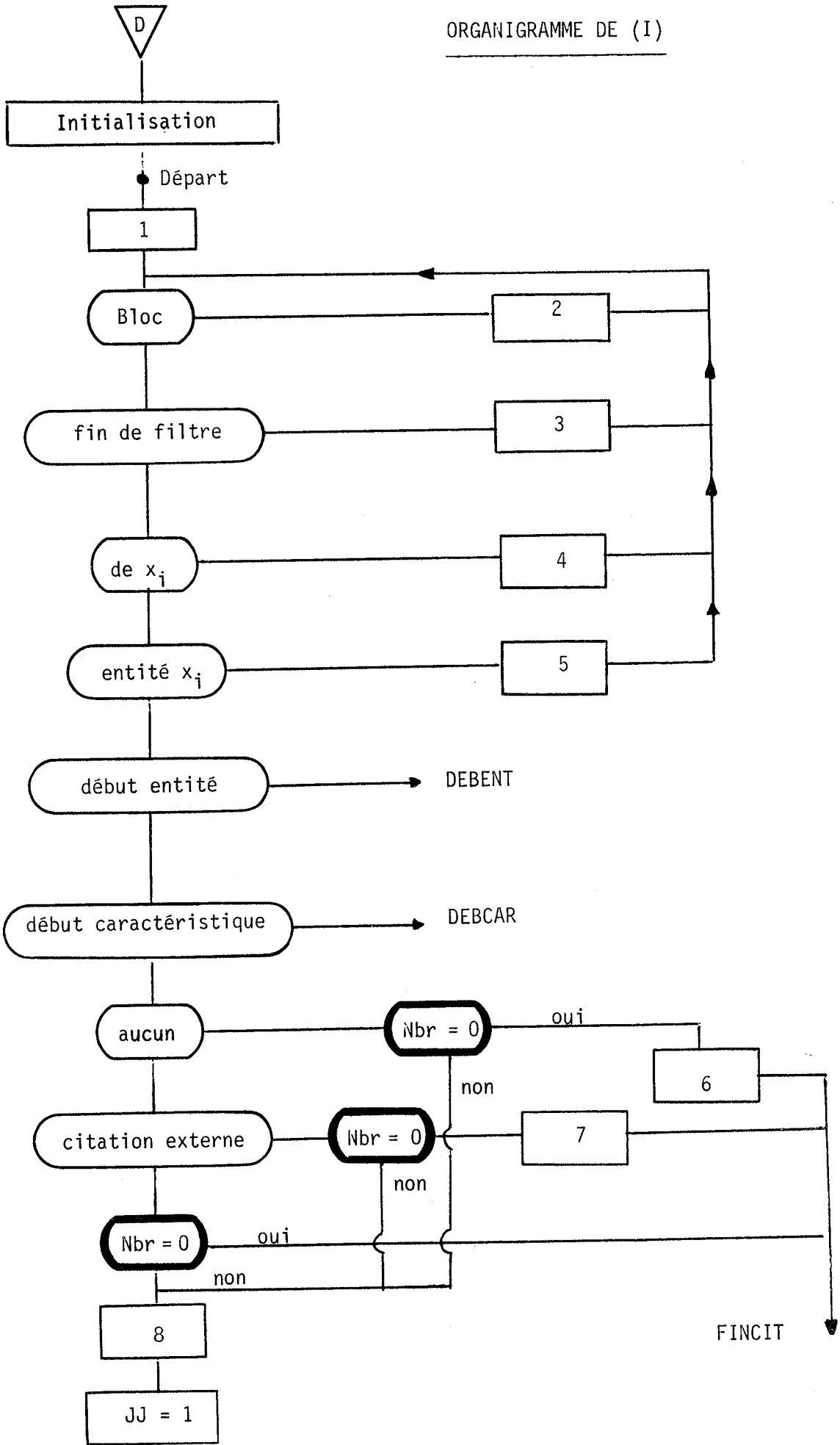
- NBR = 0	est le test de classe d'entités vide
- KK = 1	est le test de classe d'entités dont aucun représentant ne vérifie le filtre
- RET = 0	évaluation précédente est fausse
- RET = 1	évaluation précédente est vraie
- RET = 2	évaluation précédente est indéfinie.

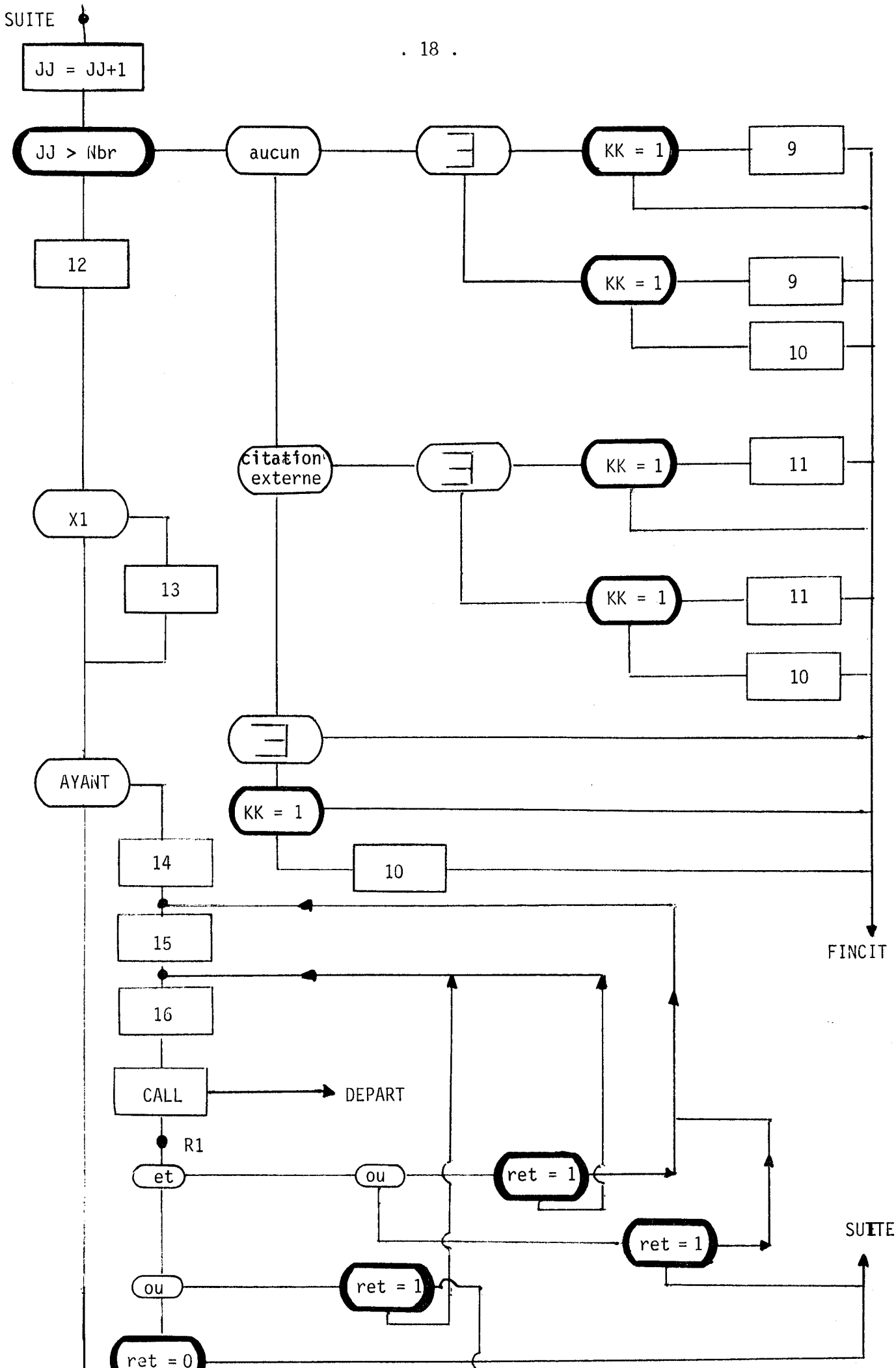
Nous donnons ensuite (page 22) l'organigramme de (I_{QS}) déduit de (I) par (C) pour la question :

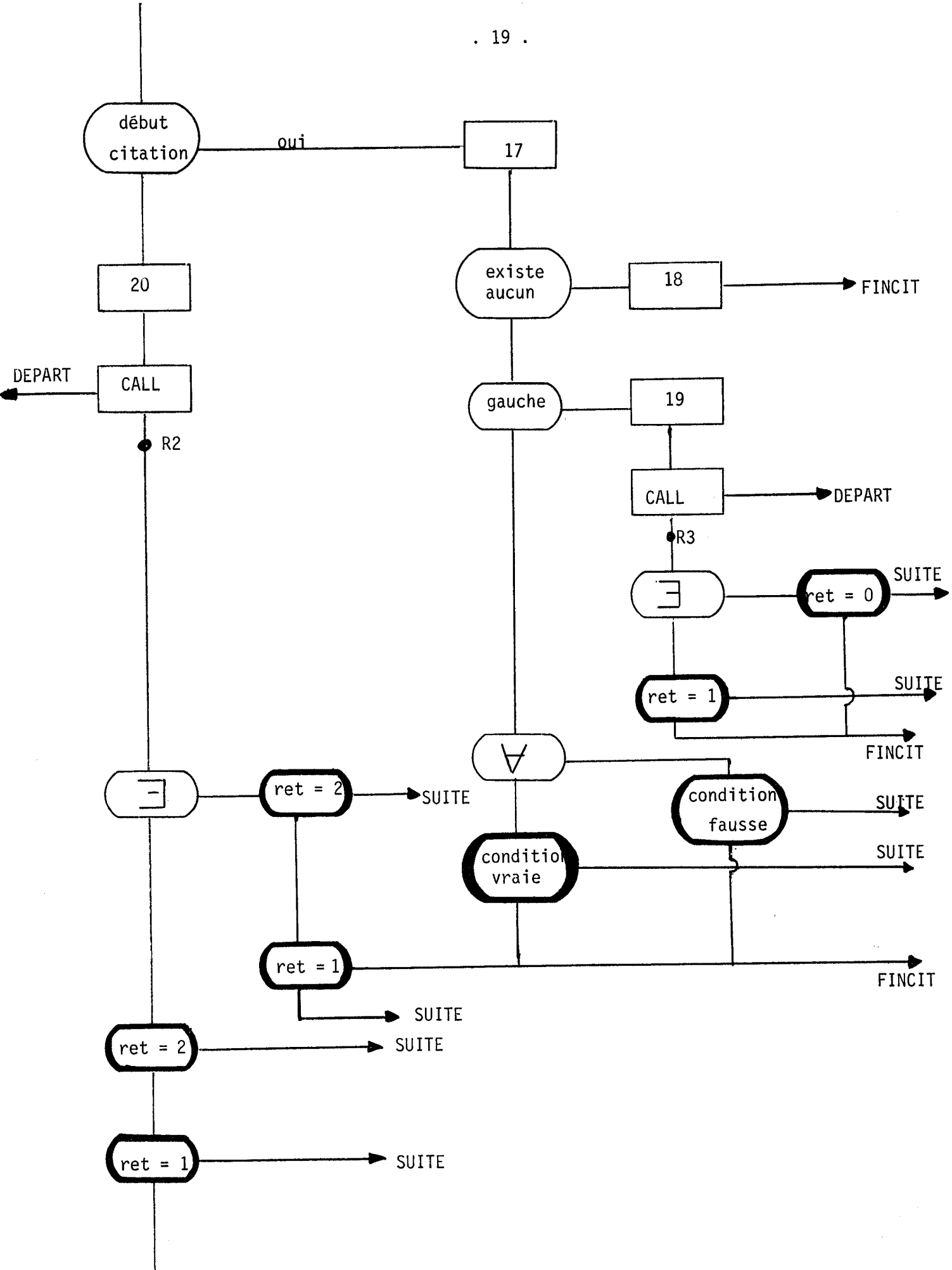
NUMERO DE TOUT DOCUMENT AYANT VALEUR DE UN MOT-CLE = 500 ; ?

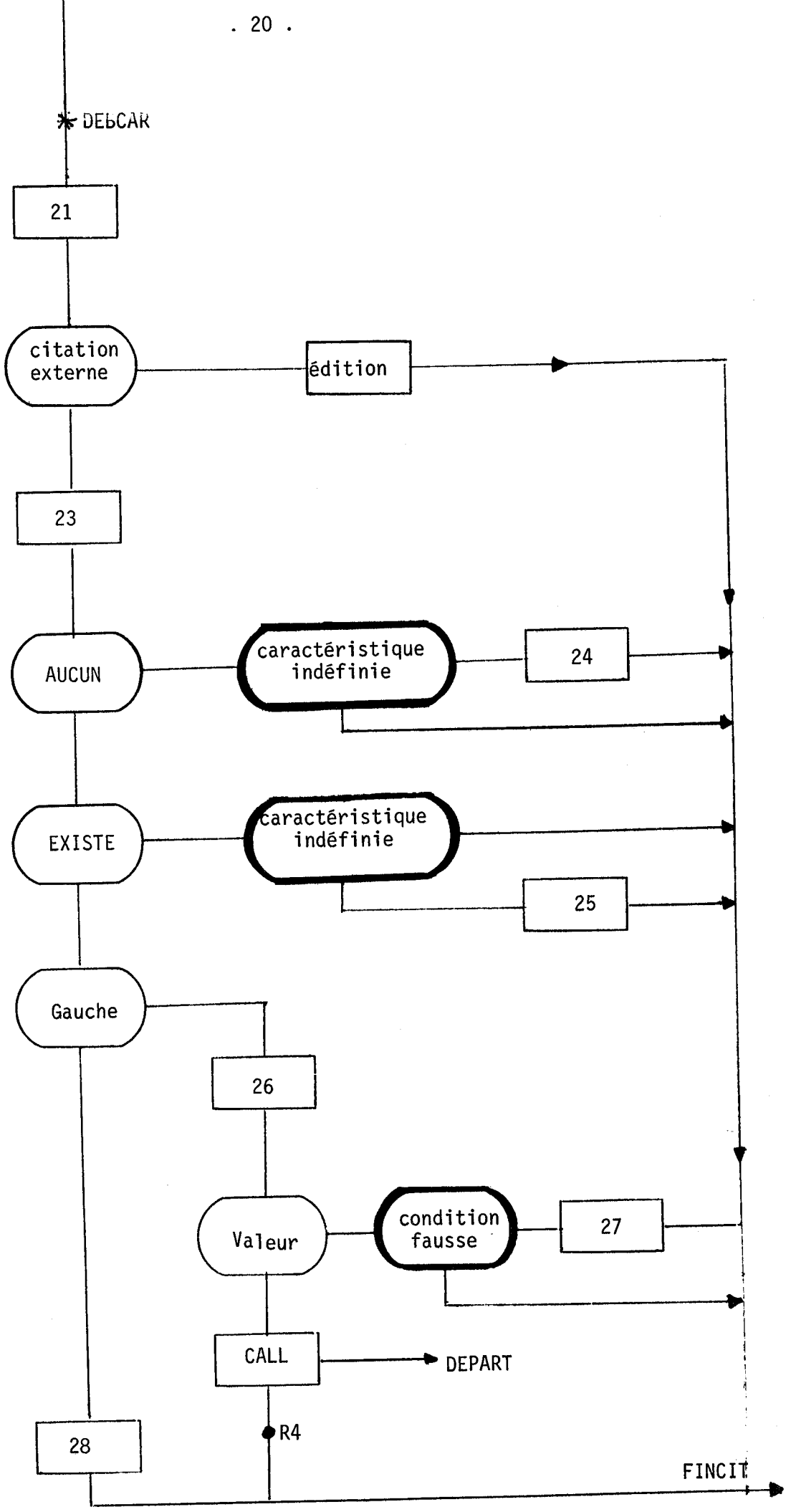
./.

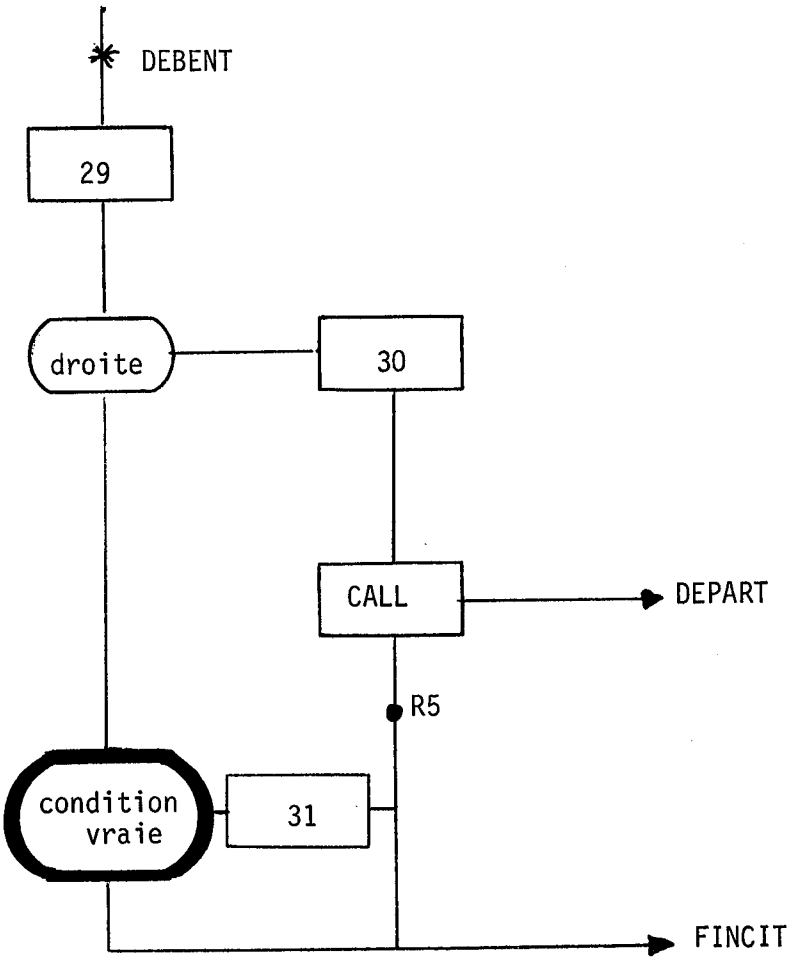
ORGANIGRAMME DE (I)

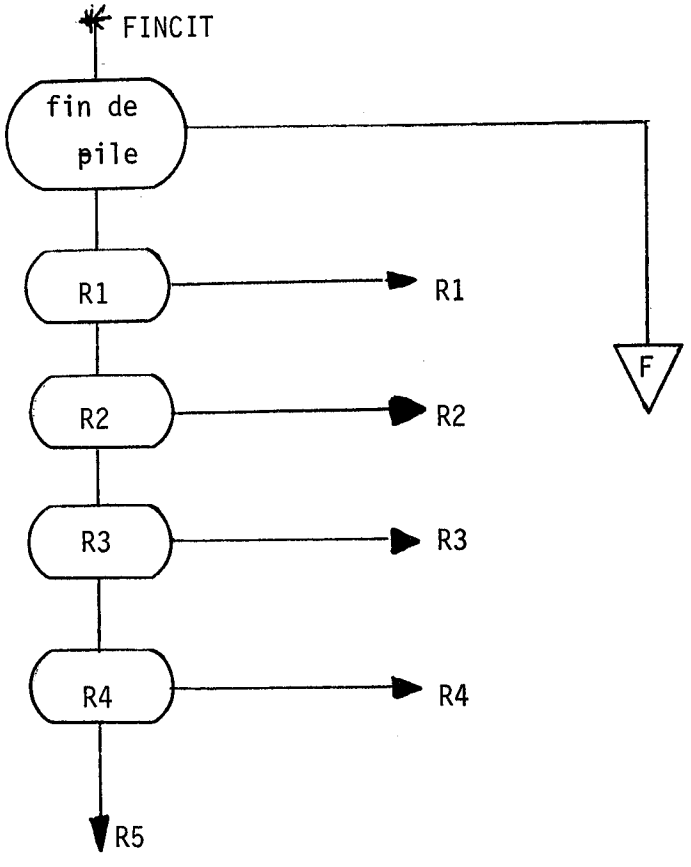



SUITE
JJ = JJ+1
JJ > Nbr
aucun
∃
KK = 1
9
12
KK = 1
9
10
citation externe
∃
KK = 1
11
X1
13
KK = 1
11
10
∃
AYANT
KK = 1
14
10
15
FINCIT
16
CALL
DEPART
R1
et
ou
ret = 1
ret = 1
SUITE
ou
ret = 1
ret = 0

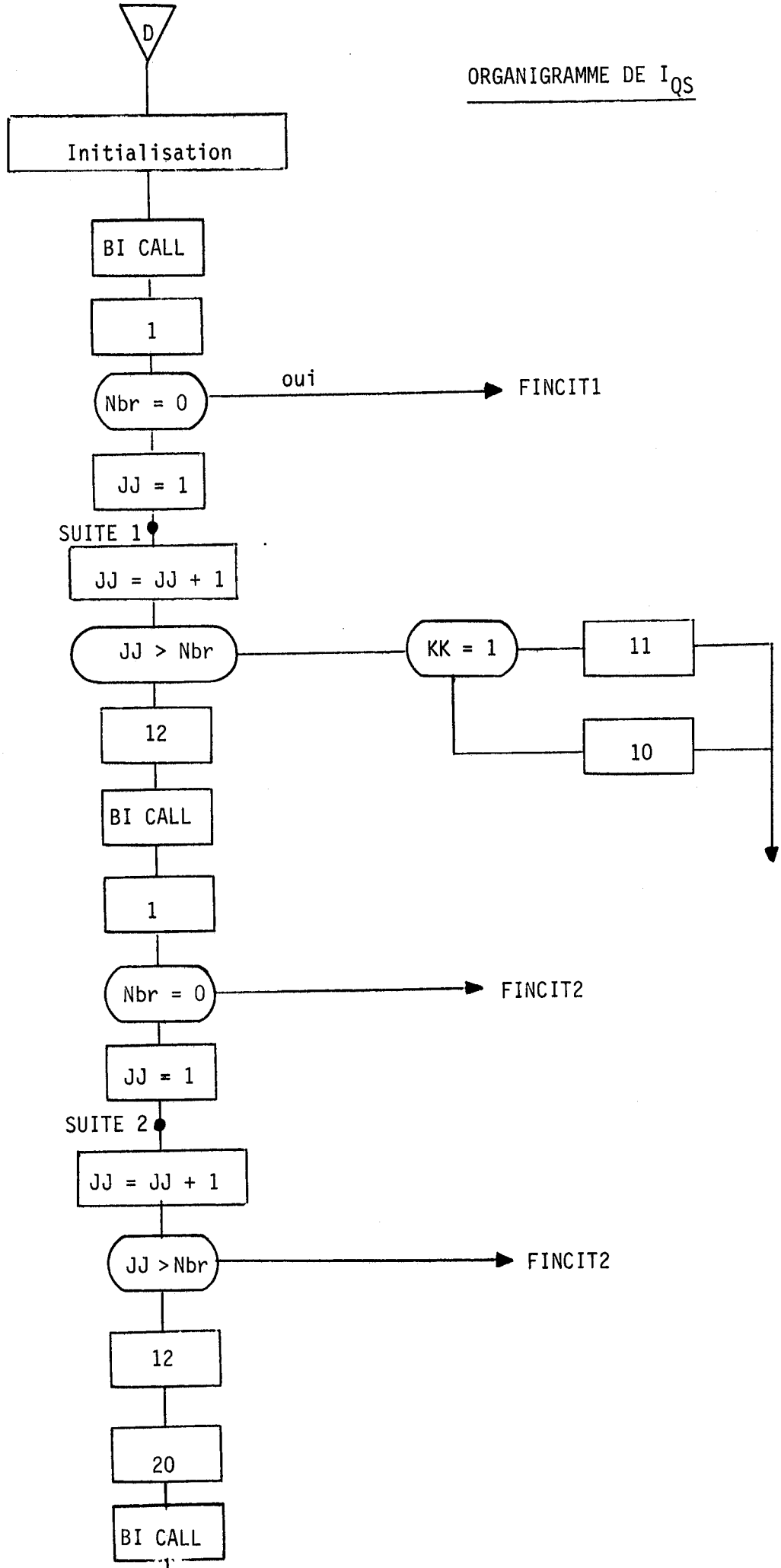

début citation
oui
17
20
existe aucun
18
FINCIT
DEPART
CALL
gauche
19
R2
CALL
DEPART
R3
SUITE
ret = 0
SUITE
ret = 1
FINCIT
ret = 2
SUITE
condition fausse
SUITE
condition vraie
SUITE
ret = 1
FINCIT
SUITE
ret = 2
SUITE
ret = 1
SUITE

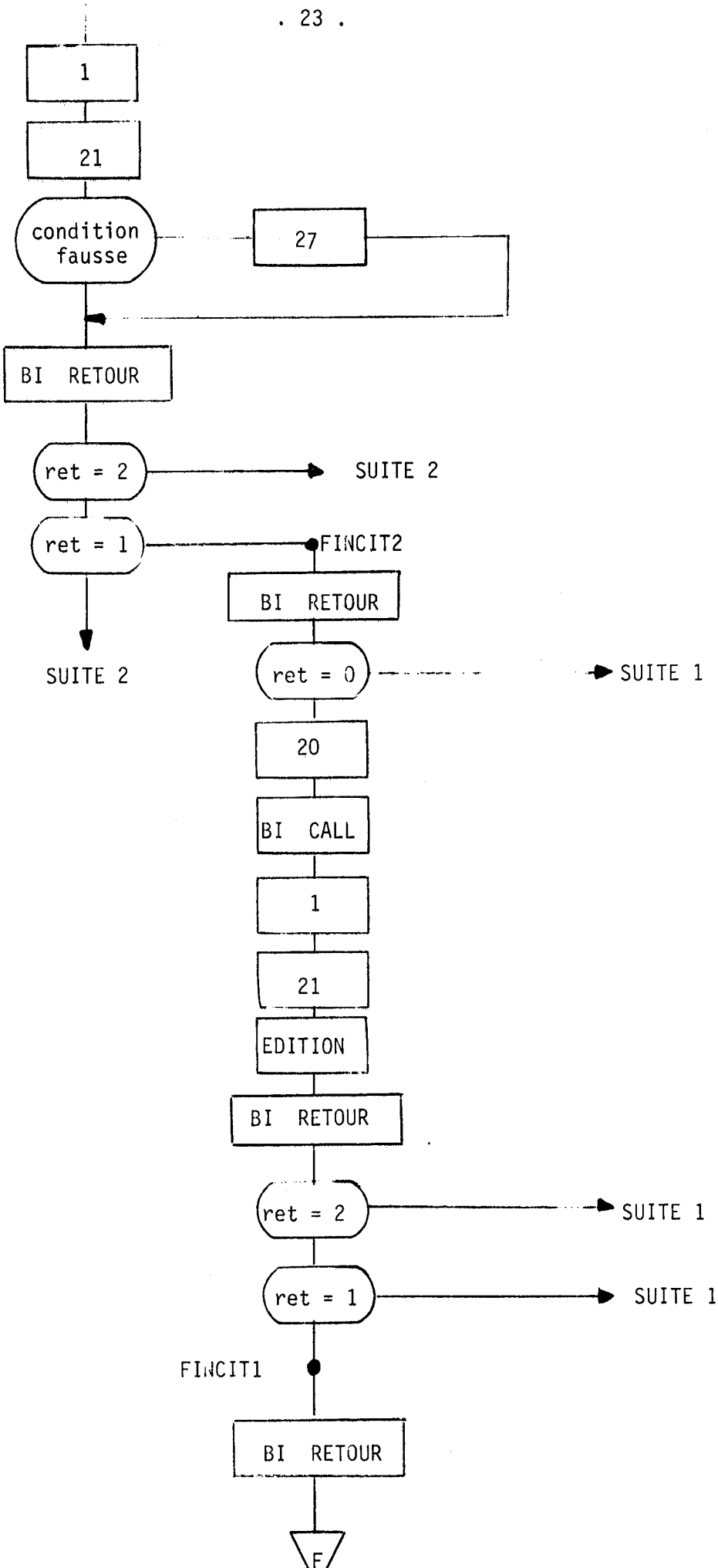

* DEBCAR
21
citation externe
édition
23
AUCUN
caractéristique indéfinie
24
EXISTE
caractéristique indéfinie
25
Gauche
26
Valeur
condition fausse
27
CALL
DEPART
R4
28
FINCIT


DEBENT
29
droite
30
CALL
DEPART
R5
condition vraie
31
FINCIT


FINCIT
fin de pile
R1
R1
F
R2
R2
R3
R3
R4
R4
R5

ORGANIGRAMME DE I_{QS}




1
21
condition fausse
27
BI RETOUR
ret = 2
SUITE 2
ret = 1
FINCIT2
BI RETOUR
SUITE 2
ret = 0
SUITE 1
20
BI CALL
1
21
EDITION
BI RETOUR
ret = 2
SUITE 1
ret = 1
SUITE 1
FINCIT1
BI RETOUR
F

A N N E X E 2

Nous avons donné au paragraphe 4 quelques performances de (I_{QS}), obtenues en simulation. Nous comparions deux programmes (I) et (I_{QS}) écrits en PL/1. Ceci permettait une comparaison entre les deux méthodes indépendamment du type de programmation.

Dans la réalité, il faut comparer (I_{QS}) écrit en code avec (I) écrit en PL/1.

Sur un exemple de question équivalent à celui donné au paragraphe 5, nous avons obtenu les performances suivantes :

Nombre de critères	1	2	3	4
I en secondes	37,3	59,1	80,9	102,5
I_{QS} en secondes	14,1	16,4	18,9	21,0
% gain	62 %	72 %	77 %	80 %

On voit que l'ajout d'un critère avec (I) coûte environ 21 secondes alors qu'il ne coûte que 2 secondes avec (I_{QS}).

B I B L I O G R A P H I E

[1] *JR. ABRIAL, J. BAS, G. BEAUME, G. HENNERON, R. MORIN, G. VIGLIANO*

Projet SOCRATE
(I) Spécifications générales

communication I.M.A.G. Août 1970

[2] *G. VIGLIANO*

Projet SOCRATE
(2.1.) Langage de requêtes

Thèse présentée à l'Université de Grenoble, déc. 1970

[3] *G. BEAUME*

Projet SOCRATE
(2.2.) Gestion des mémoires

Thèse présentée à l'Université de Grenoble, déc. 1970

[4] *R. MORIN*

Projet SOCRATE
(2.3.) Compilation et Interprétation

Thèse présentée à l'Université de Grenoble, déc. 1970

VU,

Grenoble, le

Le Président de la Thèse

Vu,
Grenoble, le

Le Doyen de la Faculté des Sciences

Vu, et permis d'imprimer

Le Recteur de l'Académie
de Grenoble