Interprétation de graphes pour l'enchaînement des requêtes dans un système de gestion de bases de données réparties : projet Polyphème

Christian Euzet

THESE

présentée à

Institut National Polytechnique de Grenoble

pour obtenir le grade de

DOCTEUR DE 3ème CYCLE

«Génie Informatique»

par

EUZET Christian

INTERPRETATION DE GRAPHES POUR L'ENCHAINEMENT DES REQUETES DANS UN SYSTEME DE GESTION DE BASES DE DONNEES REPARTIES.

◇ ◇

PROJET POLYPHEME

Thèse soutenue le 25 septembre 1979 devant la commission d'examen

L. BOLLIET	Président
E. ANDRE	
C. DELOBEL	Examinateurs
S. KRAKOWIAK	
E. PICHAT	

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1977-1978

Président : M. Philippe TRAYNARD

Vice-présidents : M. René PAUTHENET

M. Georges LESPINARD

PROFESSEURS TITULAIRES

MM.	BENOIT Jean	Electronique - automatique
	BESSON Jean	Chimie minérale
	BLOCH Daniel	Physique du solide - cristallographie
	BONNETAIN Lucien	Génie chimique
	BONNIER Etienne	Métallurgie
*	BOUDOURIS Georges	Electronique - automatique
	BRISSONNEAU Pierre	Physique du solide - cristallographie
	BUYLE-BODIN Maurice	Electronique - automatique
	COUMES André	Electronique - automatique
	DURAND Francis	Métallurgie
	FELICI Noël	Electronique - automatique
	FOULARD Claude	Electronique - automatique
	LANCIA Roland	Electronique - automatique
	LONGEQUEUE Jean-Pierre	Physique nucléaire corpusculaire
	LESPINARD Georges	Mécanique
	MOREAU René	Mécanique
	PARIAUD Jean-Charles	Chimie - physique
	PAUTHENET René	Electronique - automatique
	PERRET René	Electronique - automatique
	POLOUJADOFF Michel	Electronique - automatique
	TRAYNARD Philippe	Chimie - physique
	VEILLON Gérard	Informatique fondamentale et appliquée

* en congé pour études

PROFESSEURS SANS CHAIRE

MM.	BLIMAN Samuël	Electronique - automatique
	BOUVARD Maurice	Génie mécanique
	COHEN Joseph	Electronique - automatique
	GUYOT Pierre	Métallurgie physique
	LACOUME Jean-Louis	Electronique - automatique
	JOUBERT Jean-Claude	Physique du solide - cristallographie

.../...

MM.	ROBERT André	Chimie appliquée et des matériaux
	ROBERT François	Analyse numérique
	ZADWORNY François	Electronique - automatique

MAITRES DE CONFERENCES

MM.	ANCEAU François	Informatique fondamentale et appliquée
	CHARTIER Germain	Electronique - automatique
	CHIAVERINA Jean	Biologie, biochimie, agronomie
	IVANES Marcel	Electronique - automatique
	LESIEUR Marcel	Mécanique
	MORET Roger	Physique nucléaire - corpusculaire
	PIAU Jean-Michel	Mécanique
	PIERRARD Jean-Marie	Mécanique
	SABONNADIERE Jean-Claude	Informatique fondamentale et appliquée
Mme	SAUCIER Gabrielle	Informatique fondamentale et appliquée
M.	SOHM Jean-Claude	Chimie Physique

CHERCHEURS DU C.N.R.S. (Directeur et Maîtres de Recherche)

M.	FRUCHART Robert	Directeur de Recherche
MM.	ANSARA Ibrahim	Maître de Recherche
	BRONOEL Guy	Maître de Recherche
	CARRE René	Maître de Recherche
	DAVID René	Maître de Recherche
	DRIOLE Jean	Maître de Recherche
	KLEITZ Michel	Maître de Recherche
	LANDAU Ioan-Doré	Maître de Recherche
	MATHIEU Jean-Claude	Maître de Recherche
	MERMET Jean	Maître de Recherche
	MUNIER Jacques	Maître de Recherche

Personnalités habilitées à diriger des travaux de recherche (décision du Conseil Scientifique) E.N.S.E.E.G.

MM.	BISCONDI Michel	Ecole des Mines St. Etienne (dépt. Métallurgie)
	BOOS Jean-Yves	Ecole des Mines St. Etienne (Métallurgie)
	DRIVER Julian	Ecole des Mines St. Etienne (Métallurgie)

.../...

MM.	KOBYLANSKI André	Ecole des Mines St. Etienne (Métallurgie)
	LE COZE Jean	Ecole des Mines St. Etienne (Métallurgie)
	LESBATS Pierre	Ecole des Mines St. Etienne (Métallurgie)
	LEVY Jacques	Ecole des Mines St. Etienne (Métallurgie)
	RIEU Jean	Ecole des Mines St. Etienne (Métallurgie)
	SAINFORT	C.E.N. Grenoble (Métallurgie)
	SOUQUET	U.S.M.G.
	CAILLET Marcel	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	COULON Michel	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	GUILHOT Bernard	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	LALAUZE René	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	LANCELOT Francis	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	SARRAZIN Pierre	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	SOUSTELLE Michel	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	THEVENOT François	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	THOMAS Gérard	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	TOUZAIN Philippe	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	TRAN MINH Canh	Ecole des Mines St. Etienne (Chim. Min. Ph.)

E.N.S.E.R.G.

MM.	BOREL	Centre d'études nucléaires de Grenoble
	KAMARINOS	Centre national recherche scientifique

E.N.S.E.G.P.

M.	BORNARD	Centre national recherche scientifique
Mme	CHERUY	Centre national recherche scientifique
MM.	DAVID	Centre national recherche scientifique
	DESCHIZEAUX	Centre national recherche scientifique

Je voudrais tout d'abord remercier Monsieur Louis BOLLIET, Directeur du Département Informatique de l'IUT-B de Grenoble, de m'avoir fait l'honneur de présider le jury de cette thèse.

Je veux aussi exprimer ma sincère reconnaissance à Monsieur Claude DELOBEL, Professeur à l'Université Scientifique et Médicale de Grenoble, pour sa confiance, sa patience et ses conseils qui m'ont permis de mener à bien ce travail.

Je remercie également Monsieur Edouard ANDRE, Ingénieur au Centre Scientifique CII-HB de Grenoble, Monsieur Sacha KRAKOWIAK, Professeur à l'Université Scientifique et Médicale de Grenoble et Monsieur Etienne PICHAT, Professeur au Conservatoire National des Arts et Métiers de Paris, pour leur participation au jury et la rigueur constructive de leurs critiques.

A Monsieur Michel ADIBA, Maître-Assistant à l'IUT de Grenoble et animateur du Projet Polyphème, toute ma gratitude pour les fastidieuses lectures et critiques de manuscrit dont il a bien voulu se charger.

Merci également aux membres de l'Equipe Polyphème ainsi qu'à Gilles BOGO, Paul DECITRE, Jacques PULOU et Mohamed MOALLA pour leurs nombreux conseils.

Un grand merci aussi à Madame Marie-José DOREL pour tout le soin souriant apporté à la frappe d'un manuscrit pas toujours gai.

Merci enfin à Monsieur IGLESIAS et à son équipe pour leur disponibilité et leur compétence au tirage de ce document.

Parviendrons-nous, dans le respect des libertés individuelles, à mettre en place les mécanismes cybernétiques de régulation en temps réel qui font si cruellement défaut à nos systèmes sociaux, alors qu'ils constituent le fondement des systèmes biologiques ?

LE MACROSCOPE
Vers une vision globale
Joël de ROSNAY

A mes parents,

A ma femme,

A ... mes filles !

SOMMAIRE

CHAPITRE 0

INTRODUCTION

Il est des termes - et des thèmes de réflexion - qui surgissent au fil de l'évolution de l'Informatique ; parmi ces sujets récents, pour ne pas dire "à la mode", on peut classer celui dans lequel se situe ce travail :

Les Systèmes de Gestion de Bases de Données Réparties (SGBDR) [ADIBCO].

Dans un SGBDR, les données qui composent la base sont physiquement stockées dans des calculateurs reliés par un Réseau de communication [BERGLU, ELZI73, EUZENT, POUZIN].

L'usager ignore en fait l'emplacement de stockage de ces données et la notion même de répartition ; pour les manipuler, il s'adresse au SGBDR dans les mêmes termes que pour une base de données centralisée, c'est-à-dire sous forme de *requête*.

Toute requête se compose de deux ou trois parties :

- une partie "caractéristiques" définissant avec précision les données manipulées ;
- une partie "commandes d'accès-mise à jour" :
  - . interrogation,
  - . modification,
  - . création,
  - . suppression ;
- éventuellement, une partie "traitement" des ensembles de données obtenus :
  - . union,
  - . intersection,
  - . composition ....

Une requête de l'utilisateur du SGBDR sera appelée *Requête Globale*.

Une requête globale doit, avant d'être exécutée dans l'environnement réparti considéré, être *"décomposée"* par le système :

- dans une première phase, la requête est localisée pour donner un ensemble de requêtes locales définies sur les calculateurs du réseau [ADIBA - CALECA] ;
- dans un deuxième temps, il s'agit d'enchaîner les différentes requêtes locales en vue d'élaborer sur le réseau la réponse finale.

Le sujet de notre travail est d'étudier les paramètres influençant cet enchaînement de requêtes locales et de fournir un outil permettant de le décrire.

Dans un premier chapitre, nous situerons le contexte de ce travail qui est le *Projet POLYPHEME* et nous détaillerons l'exécution d'une requête globale en une phase de Décomposition-Localisation [CALECA - NGUYEN] et une phase d'Enchaînement.

Le deuxième chapitre est une étude des différents choix entrant en ligne de compte pour l'élaboration d'une stratégie d'exécution répartie dans un SGBDR :

- compilation suivie d'une exécution ou interprétation ;
- "granularité" des données prises en compte à l'exécution ; cette étape conduit à définir le plus petit ensemble de données manipulable et transportable ;
- combinaisons possibles au niveau de la localisation des données, des opérateurs et du système de commande.

Cette étude générale des types possibles d'exécution de requêtes dans un SGBDR permet de situer les choix de Polyphème.

Dans le chapitre trois, nous énumérerons les objectifs que doit satisfaire une représentation de requête globale susceptible de prendre en compte les différents critères cités dans le chapitre précédent ; nous donnerons alors la représentation graphique qui, selon nous, répond le mieux à ces divers besoins :

les Réseaux de Petri Interprétés.

Enfin, dans un quatrième chapitre, nous décrirons la réalisation faite dans le projet Polyphème :
Implantation d'un interpréteur de "Réseaux de Petri Interprétés" représentant des requêtes globales.

CHAPITRE I

PRESENTATION DE POLYPHEME

Le projet POLYPHEME qui a servi de cadre à ce travail est mené à l'Université de Grenoble, en liaison avec l'Equipe CII-HB et le Projet SIRIUS de l'IRIA.

Dans ce projet, on envisage un SGBDR comme un *Système de Coopération de Bases de Données Réparties*. La Base de Données Répartie (BDR) vue par l'Utilisateur Global est construite de manière *ascendante* à partir de bases de données locales pré-existantes (par opposition à une approche descendante (voir Figure I.1)).

D'autre part, les bases de données pré-existantes peuvent être différentes d'un point de vue modèle et/ou d'un point de vue SGBD [DATE] ; on s'intéresse donc à un SGBDR *hétérogène* dans Polyphème.

Dans la suite de ce chapitre, nous allons rapidement décrire les *architecture logiques et physiques* de ce type de SGBDR en précisant les hypothèses dans lesquelles se place notre étude.

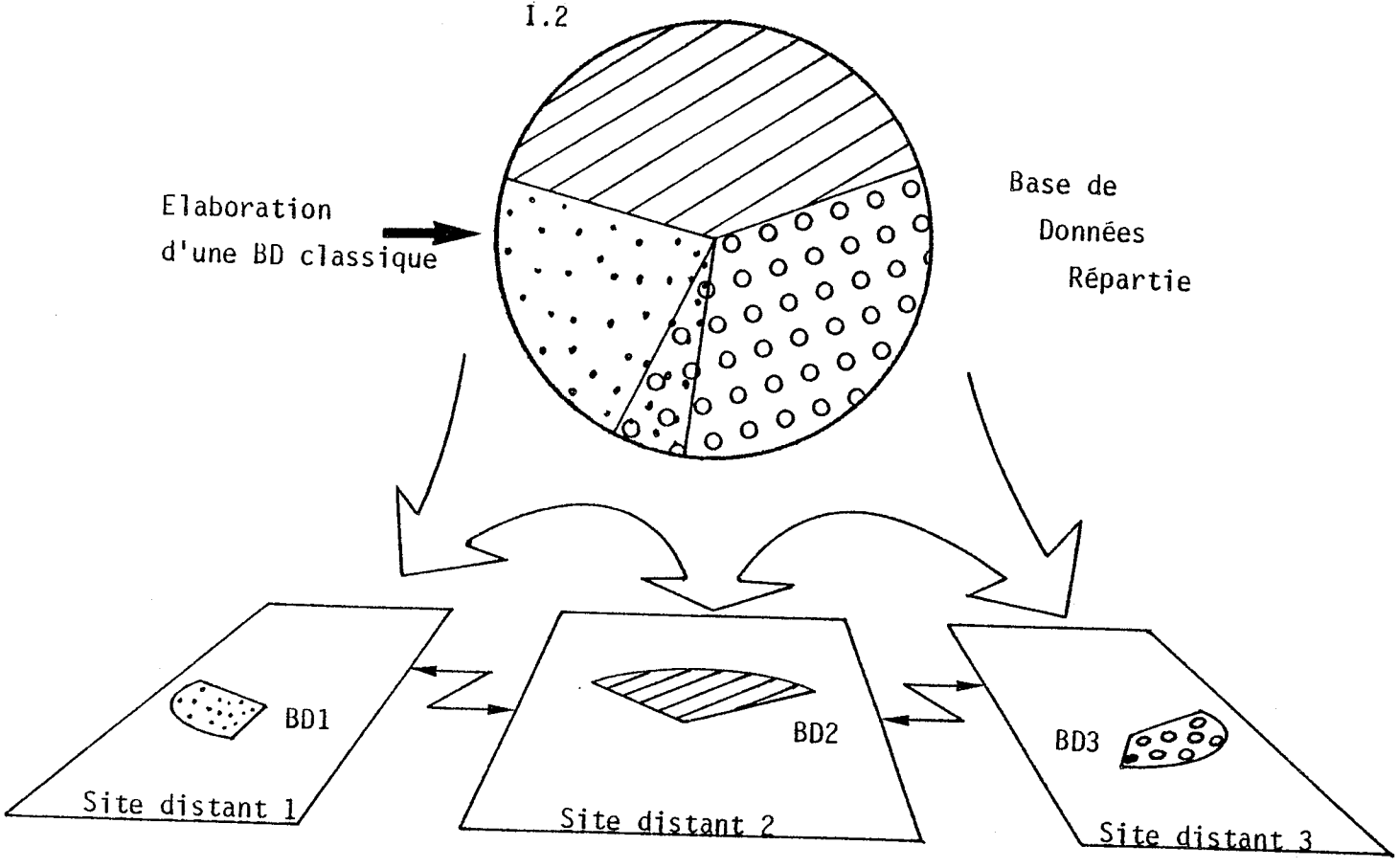
Nous aborderons également le découpage logique d'une Requête Globale afin de distinguer une partie opérative qui précise "QUOI" faire sur les divers éléments du SGBDR d'une partie commande qui indique "COMMENT" exécuter et synchroniser ces exécutions [LITWIN].

I-1 Architecture Logique de Polyphème

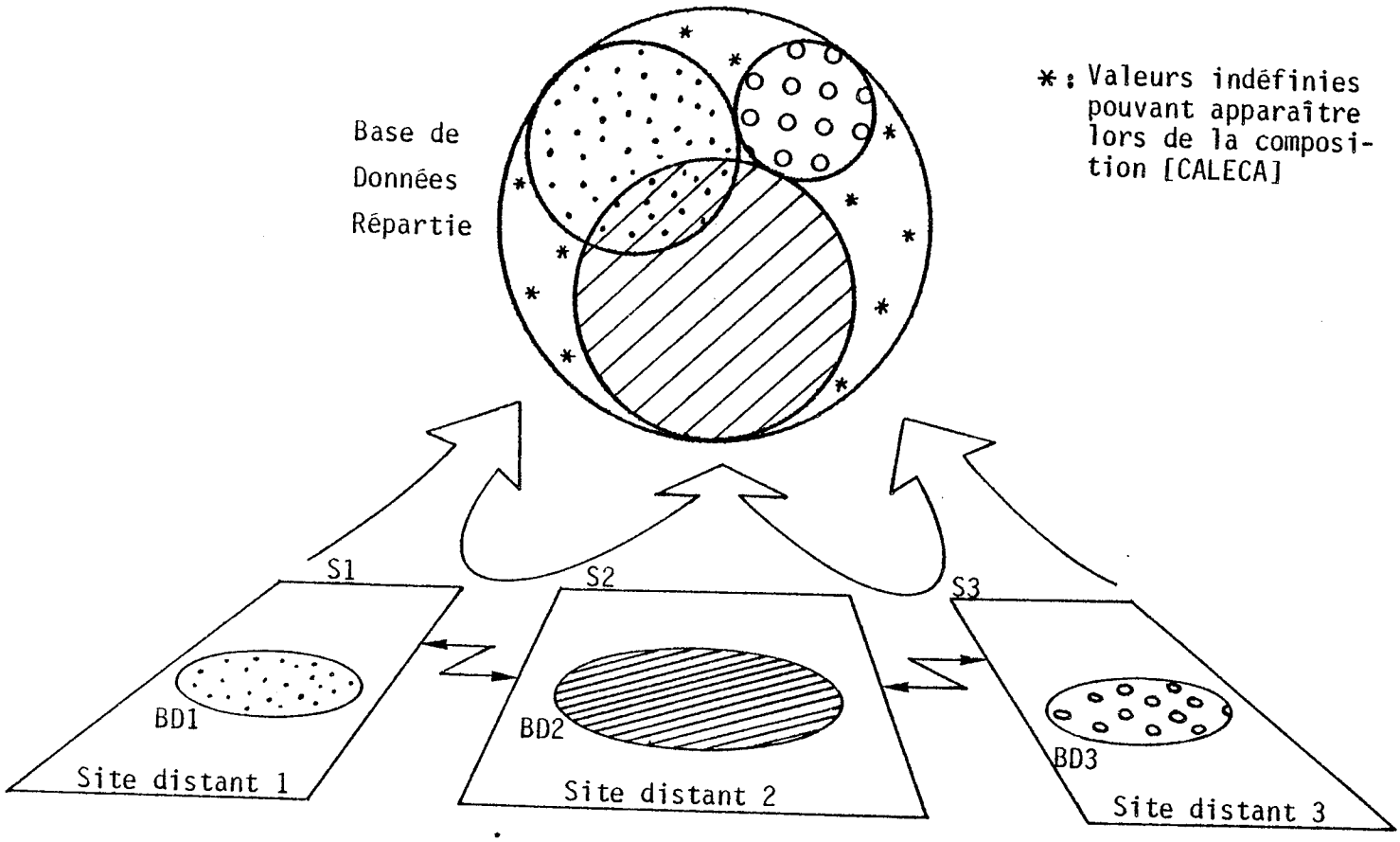
Elle est basée sur le concept de Machines Relationnelles Logiques (la terminologie utilisée est celle des modèles relationnels de données, s'y reporter [ADIDEL, CODD]).

Une Machine Relationnelle Logique est un automate répondant aux commandes suivantes [ADCAE1 et 2] :

- les quatre opérations élémentaires sur des relations :
  - . obtenir un n-uplet d'une relation,
  - . insérer un n-uplet dans une relation,
  - . supprimer un n-uplet d'une relation,
  - . modifier un n-uplet d'une relation.
- les opérations de l'Algèbre Relationnelle :



Approche Descendante



Approche Ascendante

Figure I.1 - Stratégies de conception de Bases de Données Réparties

. sélection,
. projection,
. union,
. composition ....

Cette architecture utilise deux types de Machines Relationnelles Logiques :

- la *Machine Locale* qui assure l'interface avec les bases coopérantes homogénéisant ainsi leur comportement ;
- la *Machine Globale* qui permet à plusieurs utilisateurs de décrire et d'utiliser une base de données répartie.

I-1-1 Machine Locale

Elle permet d'avoir ce que l'on appellera une *Vue Locale* relationnelle de la base locale sous-jacente ; on peut qualifier une vue locale comme une vue virtuelle de la base de données locale.

Cette vue locale correspond à un ensemble de relations ou de "tables de valeurs" n'ayant pas d'existence réelle dans la base ; toute interrogation ou mise à jour de cette vue locale est répercutée dans la base de donnée locale par un ensemble de *Programmes Locaux Précompilés* stockés dans la base, c'est-à-dire que toute opération portant sur une relation locale se traduira en termes de programmes locaux.

L'avantage de programmes locaux précompilés est double :

- pas de transfert de code sur le réseau (coûteux) ;
- limitation des droits d'accès aux bases locales ;

en effet, ces programmes locaux accédant via des sous-schémas à la base locale, le soin est laissé au SGBD local d'assurer sa propre cohérence et ses protections qui ne peuvent pas être ainsi "court-circuitées" par un utilisateur distant.

D'autre part, l'absence même d'un ou plusieurs types de programmes locaux pour une relation contient en elle-même une protection implicite de cette relation ; par exemple, s'il n'existe pas de programme local de suppression de n-uplet pour une relation d'une vue locale, cela signifie qu'au travers de cette vue locale, il sera impossible de supprimer des n-uplets de cette relation.

En complément des programmes locaux qui permettent les opérations élémentaires sur les relations locales, la machine relationnelle doit être capable d'exécuter les opérateurs de l'Algèbre Relationnelle (Figure I.2) [ADIBA].

Ainsi, une requête locale se présentera comme une expression relationnelle d'interrogation ou de mise à jour sur les relations qui composent la vue locale en utilisant les opérateurs relationnels.

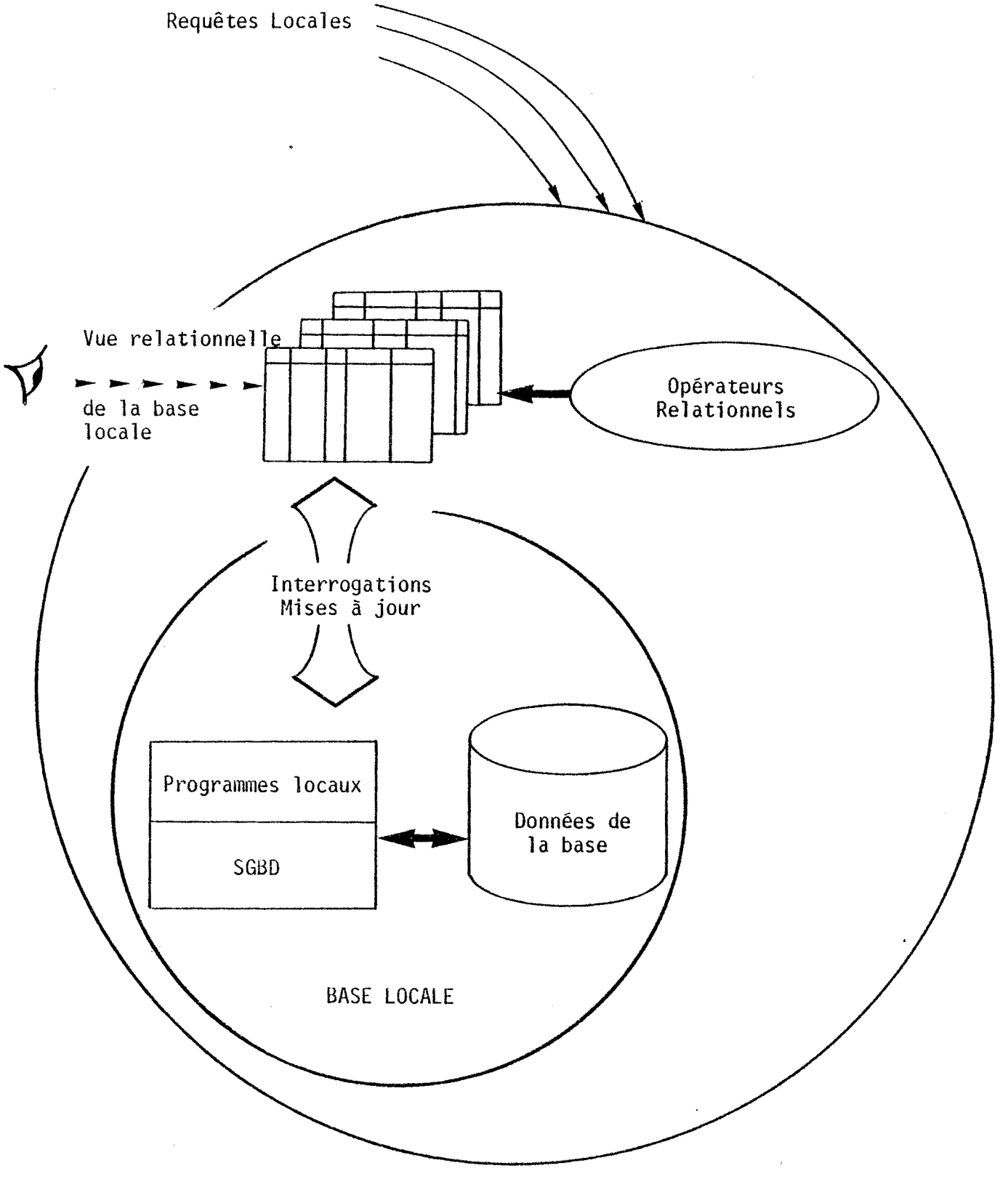


Figure I.2 - Machine Locale

I-1-2 Machine Globale

C'est la machine globale qui va donner à l'utilisateur de la BDR l'image d'une base unique réalisant ainsi la *transparence de la répartition*.

Cette image de la BDR est appelée *Vue Globale* ; elle décrit un ensemble de relations dites "relations globales", chacune construite au moyen des relations locales.

Les opérandes d'une requête globale sont donc des relations globales ; le rôle de la phase de "décomposition" est de remplacer ces opérations sur des relations globales par des opérations concernant uniquement des relations locales [ADICAL].

Pour ce faire, la Machine Globale utilise des catalogues décrivant la Vue Globale, les différentes Vues Locales, ainsi que la correspondance entre les deux [NEUBIL].

Nous verrons plus en détail au paragraphe I-3 comment peut s'exécuter une requête globale.

On a donc la structure logique présentée en Figure I-3.

I-2 Architecture physique de Polyphème

L'architecture logique que nous avons vue fait ressortir deux niveaux fonctionnels, à savoir :

- le niveau local de stockage et d'accès à un ensemble d'informations mémorisées dans un calculateur ;
- le niveau global de définition et de manipulation d'un ensemble d'informations réparties.

Pour une implantation réelle, ces deux niveaux fonctionnels peuvent ou non cohabiter sur chaque calculateur qui compose le Réseau. En effet, une Machine Globale peut se trouver sur le même site que zéro, une ou plusieurs Machines Locales (Figure I.4) [ADCAE1 et 2].

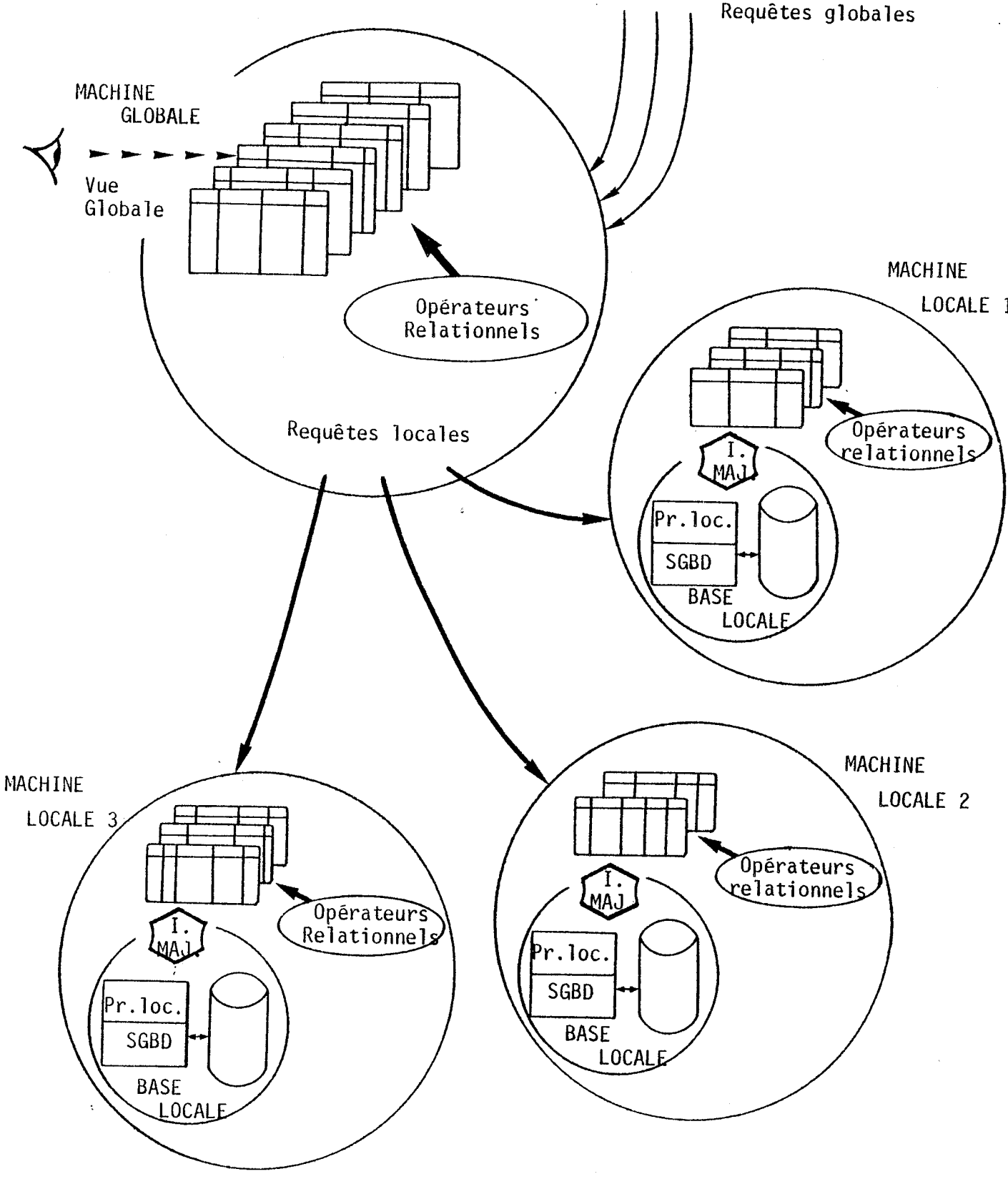


Figure I.3 - Architecture Logique de Polyphème

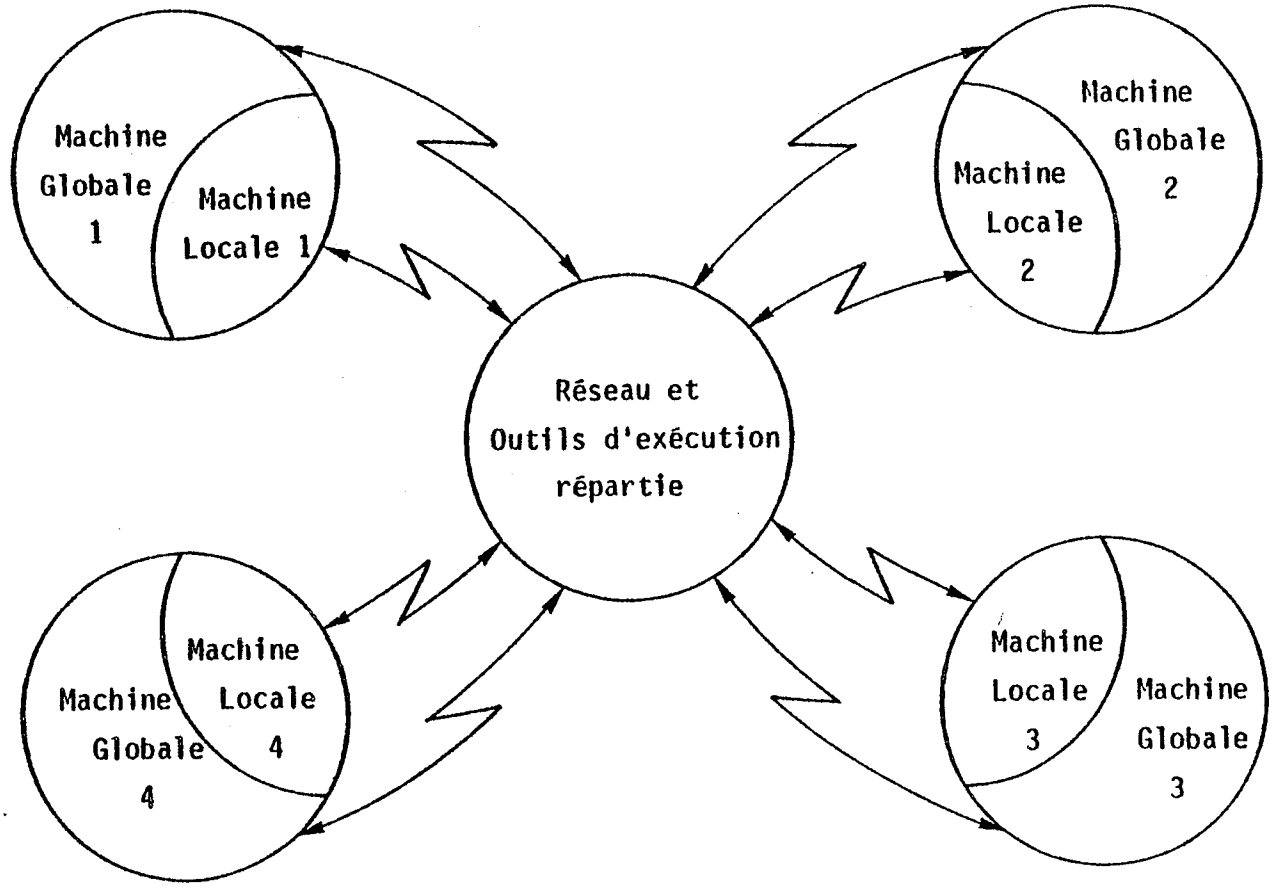


Figure I.4 - Exemple d'architecture physique
où chaque calculateur comporte une Machine Globale et une Machine Locale

De manière à limiter le nombre de problèmes, nous avons choisi pour notre travail l'architecture physique suivante :

- une seule Machine Globale (mono-utilisateur),
- plusieurs Machines Locales accessibles uniquement via la Machine Globale.

Il n'en demeure pas moins un bon nombre de questions que nous aborderons dans la suite de cette thèse.

I-3 Découpage logique d'une Requête Globale

Le rôle de la Machine Globale est de transformer toute manipulation de la vue globale en une ou plusieurs manipulation(s) des vues locales correspondantes.

Cette transformation appelée aussi "décomposition" est fonction de nombreux paramètres tels que [GOOROT] :

- le type de la requête (interrogation ou mise à jour),
- le mode de répartition des données concernées,
- l'optimisation possible au niveau des opérateurs algébriques,
- la nature du réseau de communication ....

Cette partie de décomposition indique en fait au SGBDR les opérations relationnelles (obtenir un n-uplet,projection, sélection, ...) à effectuer et les opérandes concernés (relations locales ou résultats d'opération) sous forme d'un "*graphe de données*".

Exemple :
Soient trois relations implantées sur les machines (M_i) du réseau de la manière suivante :

M1 : ETUDIANT (NE, NOM, AGE, VILLE) ;
M2 : INSCRIPTION (NE, NUV, NOTE) ;
M3 : UV (NUV, TITRE, CYCLE, RESPONSABLE).

Cela signifie donc qu'un ETUDIANT est identifié par son numéro d'étudiant NE avec comme renseignement son NOM, son AGE et la VILLE de résidence.

La relation INSCRIPTION indique les UV auxquelles est inscrit un ETUDIANT avec la NOTE obtenue pour chaque UV et chaque ETUDIANT.

Enfin, une UV est identifiée par son numéro, NUV ; pour chaque UV, on peut connaître son TITRE, le CYCLE dans lequel elle se situe et le professeur qui en est le RESPONSABLE.

Dans un but de simplification des exemples, nous n'abordons pas les problèmes de partitionnement et duplications de relations ; nous considérons que les vues locales correspondent directement aux bases locales ainsi décrites ; notre vue globale est l'union de ces vues locales.

On se donne d'autre part la structure Polyphème d'implantation des machines locales et globale décrite en Figure I-5.

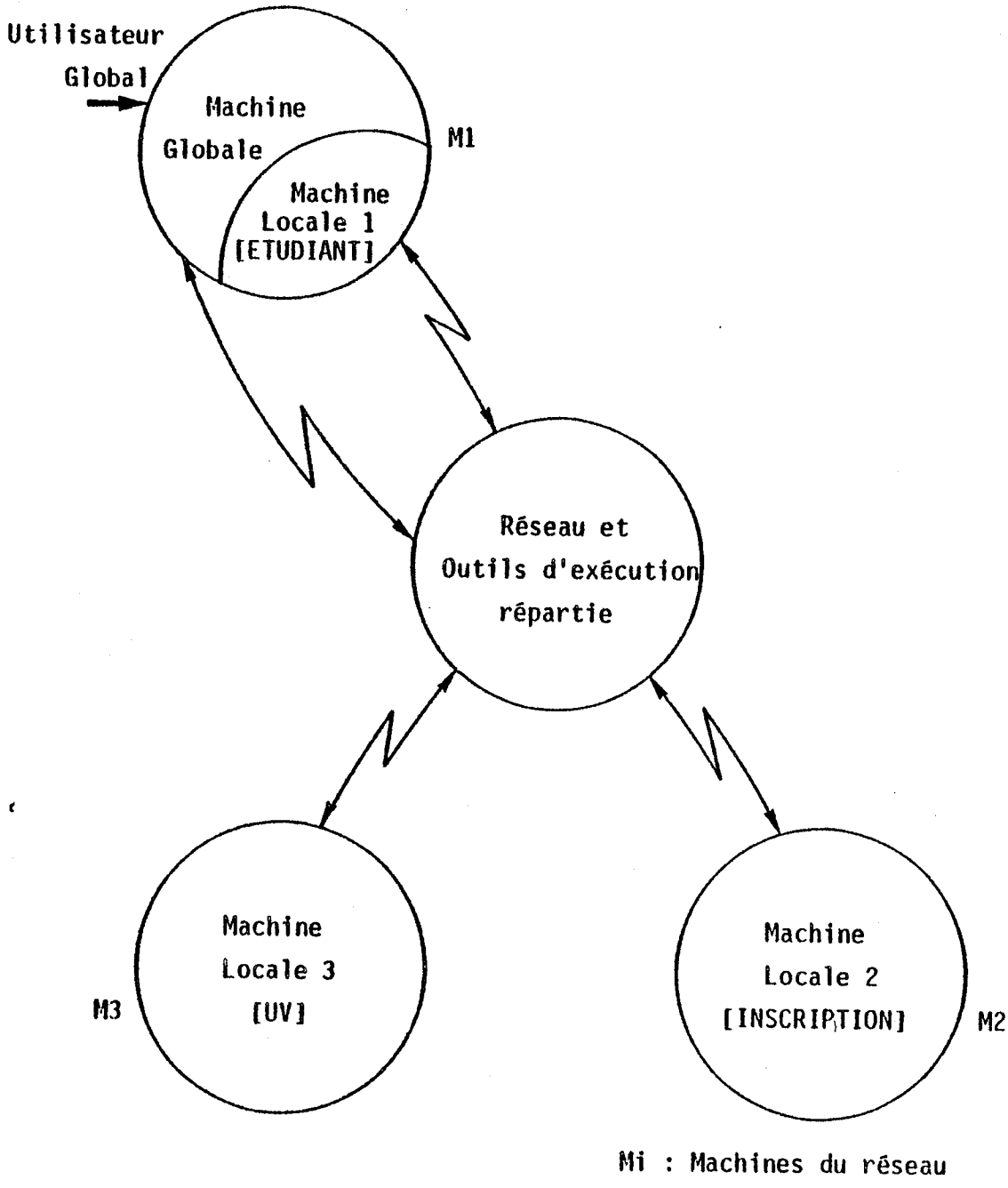


Figure I.5 - Architecture Polyphème
correspondant à l'exemple

Soit la question suivante posée par l'utilisateur de la BDR :

R1 = "Nom des ETUDIANTS de Grenoble INSCRITS aux UV de troisième cycle ?"

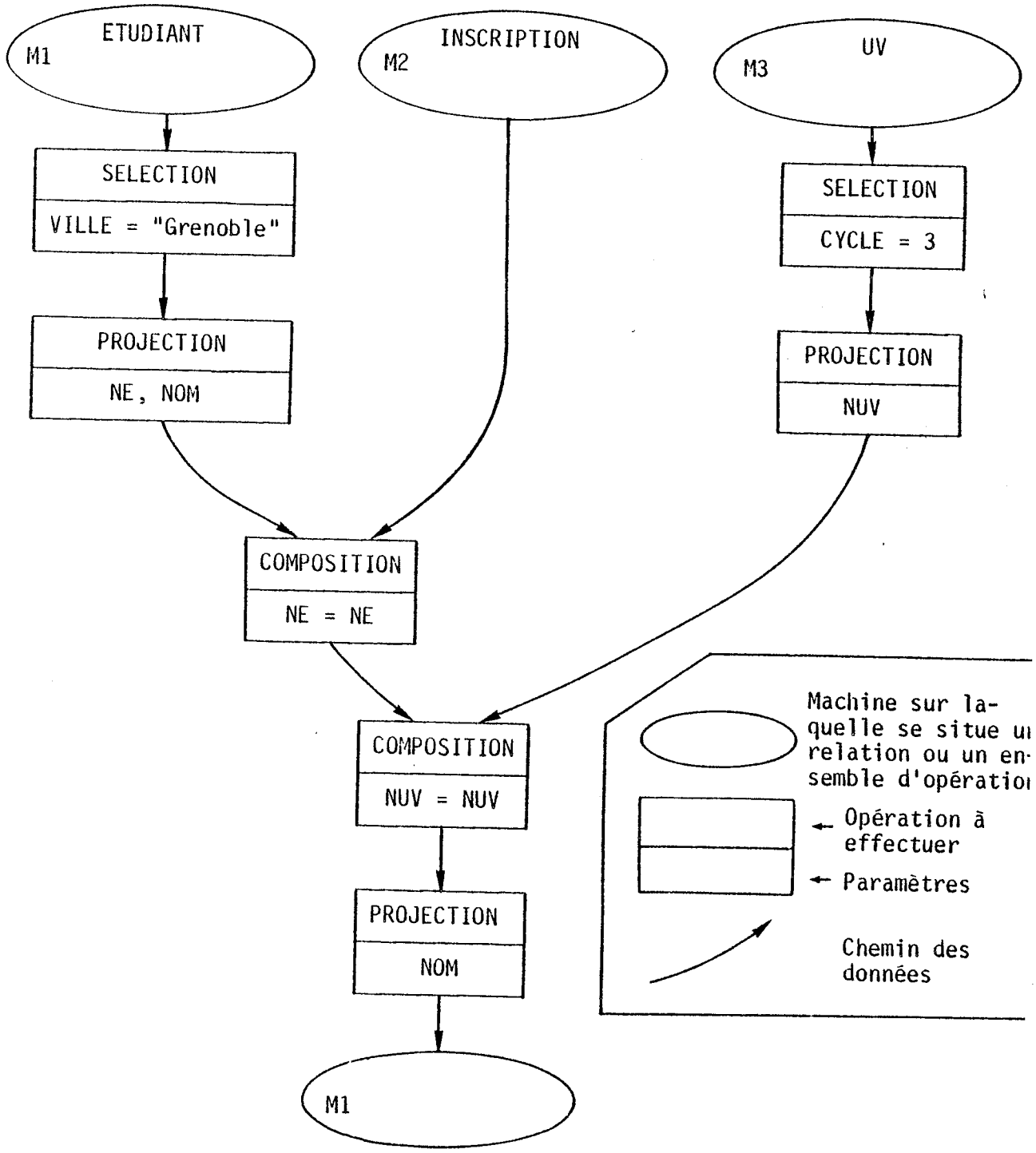


Figure I.6 - Graphe de données correspondant à la requête :
R1 = "Nom des Etudiants de Grenoble inscrits aux UV de troisième cycle"

Toute requête peut être représentée sous forme d'un arbre syntaxique du type du "graphe de données" décrivant R1 en Figure I.6.

La façon de construire cet arbre et son analyse ne sont pas l'objet du présent travail et le lecteur doit se reporter à [CALECA] pour une étude complète.

Cet exemple nous permet de présenter les principaux problèmes que l'on va rencontrer ; en effet, sur ce premier "graphe de données", seules sont localisées les relations d'origine qui sont les feuilles de l'arbre.

Il s'agit donc de localiser les opérations intermédiaires avec les différentes relations résultats en cherchant à minimiser les transferts réseaux tout en exploitant au maximum les possibilités de parallélisme.

Cette localisation peut se faire de manière dynamique au fur et à mesure de l'exécution de l'arbre par le SGBDR en fonction principalement du volume des relations à transférer. Dans notre exemple, la "COMPOSITION-NE=NE" entre "INSCRIPTION" et la relation résultante de la PROJECTION ("RP") aura lieu soit sur M1 si le volume de "RP" est supérieur à celui de "INSCRIPTION", soit sur M2 dans le cas contraire [NGUYEN].

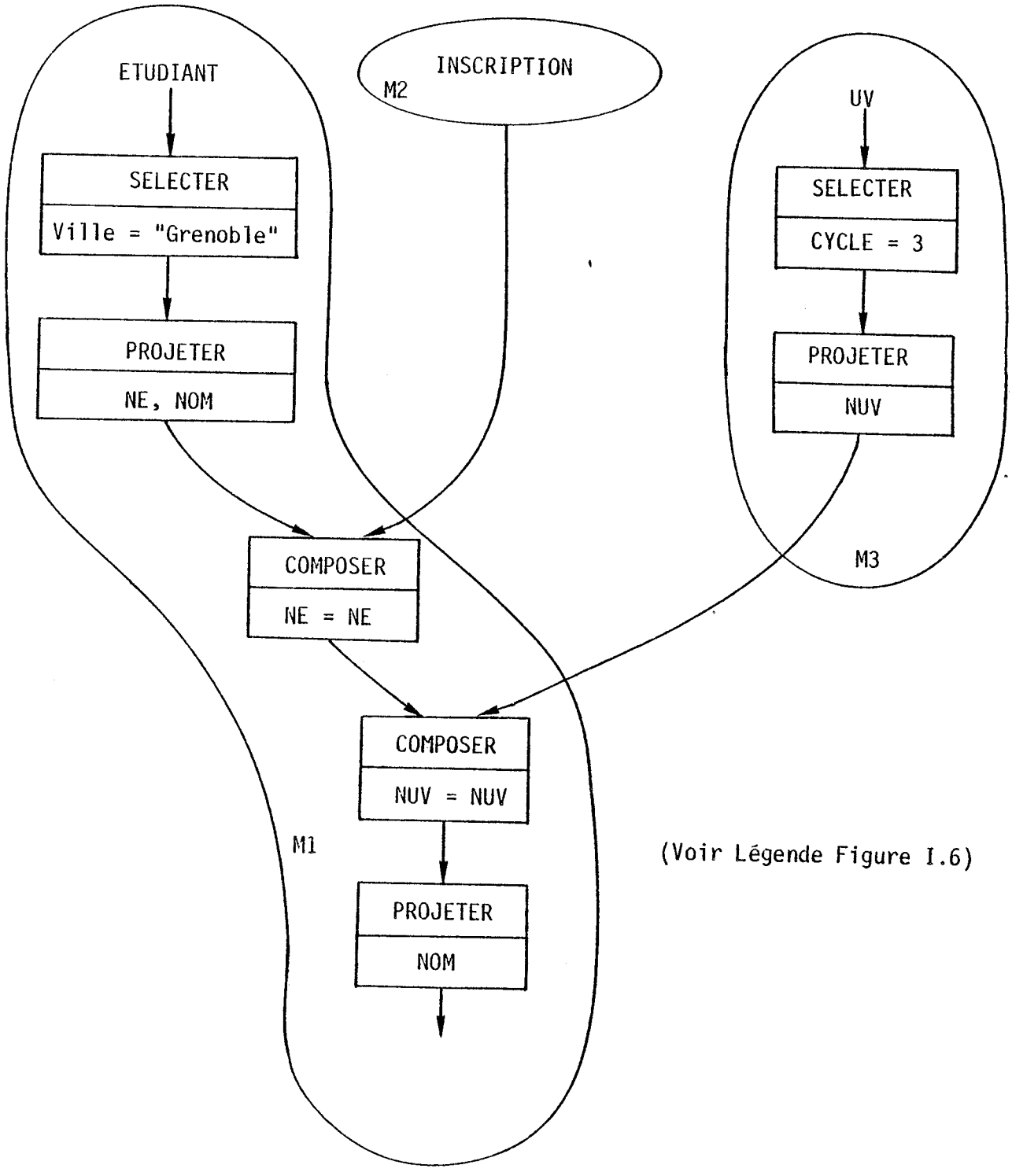
Le problème principal de cette localisation dynamique réside dans la réalisation de l'algorithme qui présidera au choix des machines à utiliser. De plus, des paramètres tels que la saturation d'un noeud du réseau ou le niveau de charge d'une machine peuvent difficilement être pris en compte à ce niveau.

La "localisation statique" consiste, par contre, en une pré-détermination des sites d'exécution en fonction des volumes des relations d'origine [CALECA].

Ce type de localisation a l'avantage d'être beaucoup plus simple à mettre en oeuvre que la localisation dynamique. Il a, par contre, le défaut de ne pas pouvoir s'adapter en temps réel aux évolutions des volumes de données.

C'est cette hypothèse de localisation statique que nous prendrons pour la suite de ce travail.

Sur l'exemple précédent, une des localisations possibles est donnée en Figure I.7.



(Voir Légende Figure I.6)

Figure I.7 - Exemple de Graphe de données totalement localisé

Une fois la localisation effectuée, il faut indiquer le "COMMENT" des actions en donnant l'ordre des différentes opérations sous-jacentes et leur synchronisation.

Pour ce faire, nous introduisons le concept de "*Graphe de Commande*" dont un exemple est donné à titre d'illustration par la Figure I-8.

Dans un graphe de commande, les noeuds correspondent aux opérations à effectuer et aux points de synchronisation ; les arcs indiquent le chemin de contrôle.

Notre travail a consisté tout d'abord à étudier les différents problèmes qui se posent lorsque l'on veut exécuter une requête globale en utilisant comme support un graphe de commande ; à partir de ces observations, nous avons ensuite défini le type de graphe qui nous a paru le plus adapté à la résolution de ces problèmes.

Nous n'avons pas étudié le passage automatique du graphe de données au graphe de commande ; nous montrerons cependant que dans certaines stratégies d'exécution, le graphe de commande correspond directement au graphe de données (Chapitre IV).

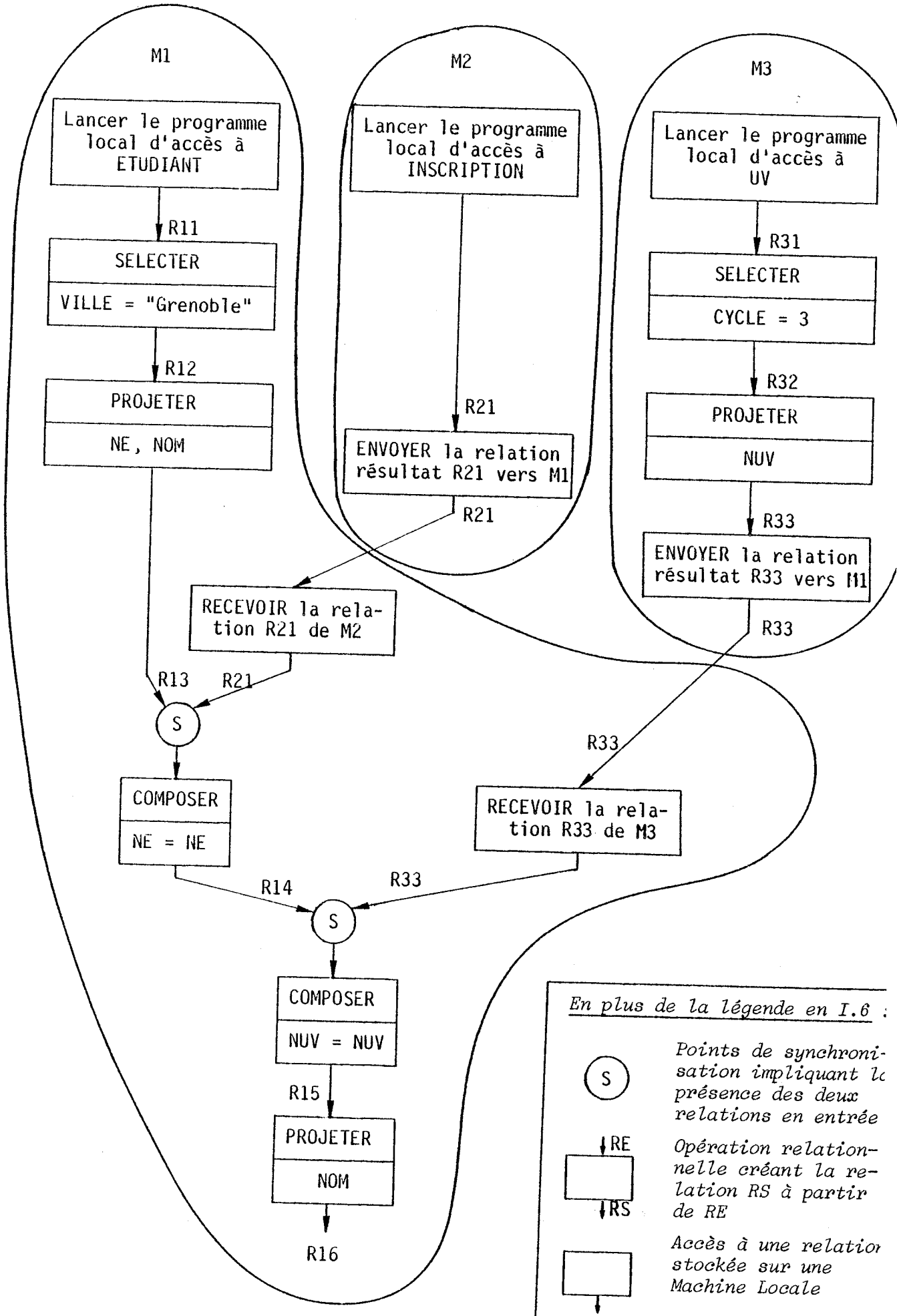


Figure I.8 - Graphe de commande correspondant à l'exemple Figure I.7

CHAPITRE II

STRATEGIES D'EXECUTION DE REQUETE DANS UN SGBDR

Ce chapitre analyse les principaux critères utilisables pour définir une stratégie d'exécution de requête au sein d'un SGBDR et obtenir ainsi différents types d'exécution à partir d'un même graphe de données localisé (Fig. II-1).

Pour qu'une exécution puisse avoir lieu dans un système réparti, elle doit être exprimée sous forme d'un programme réparti écrit dans un langage d'application répartie [CHUPIN], souvent un langage de haut niveau auquel sont ajoutés des outils de synchronisation répartie.

Ce programme réparti correspond à un ensemble de programmes monosites mutuellement synchronisés.

Nous allons donc voir d'abord sur quel site peut avoir lieu la "*traduction*" (soit interprétation, soit génération, compilation et exécution) du graphe de commande localisé.

Nous considérons ensuite la *granularité d'exécution* qui correspond au choix de l'unité de donnée indivisible qui sera traitée au niveau des machines locales et du réseau.

Enfin, notre troisième et dernier critère est lié au triptyque : *localisation des données, des opérateurs et du système de commande* ; les différentes combinaisons seront étudiées.

Il est important de se souvenir tout au long de ce chapitre des hypothèses de base dans lesquelles nous situons cette étude :

- hétérogénéité des SGBD locaux
- terminologie relationnelle
- unicité logique de la Machine Globale
- un seul utilisateur global.

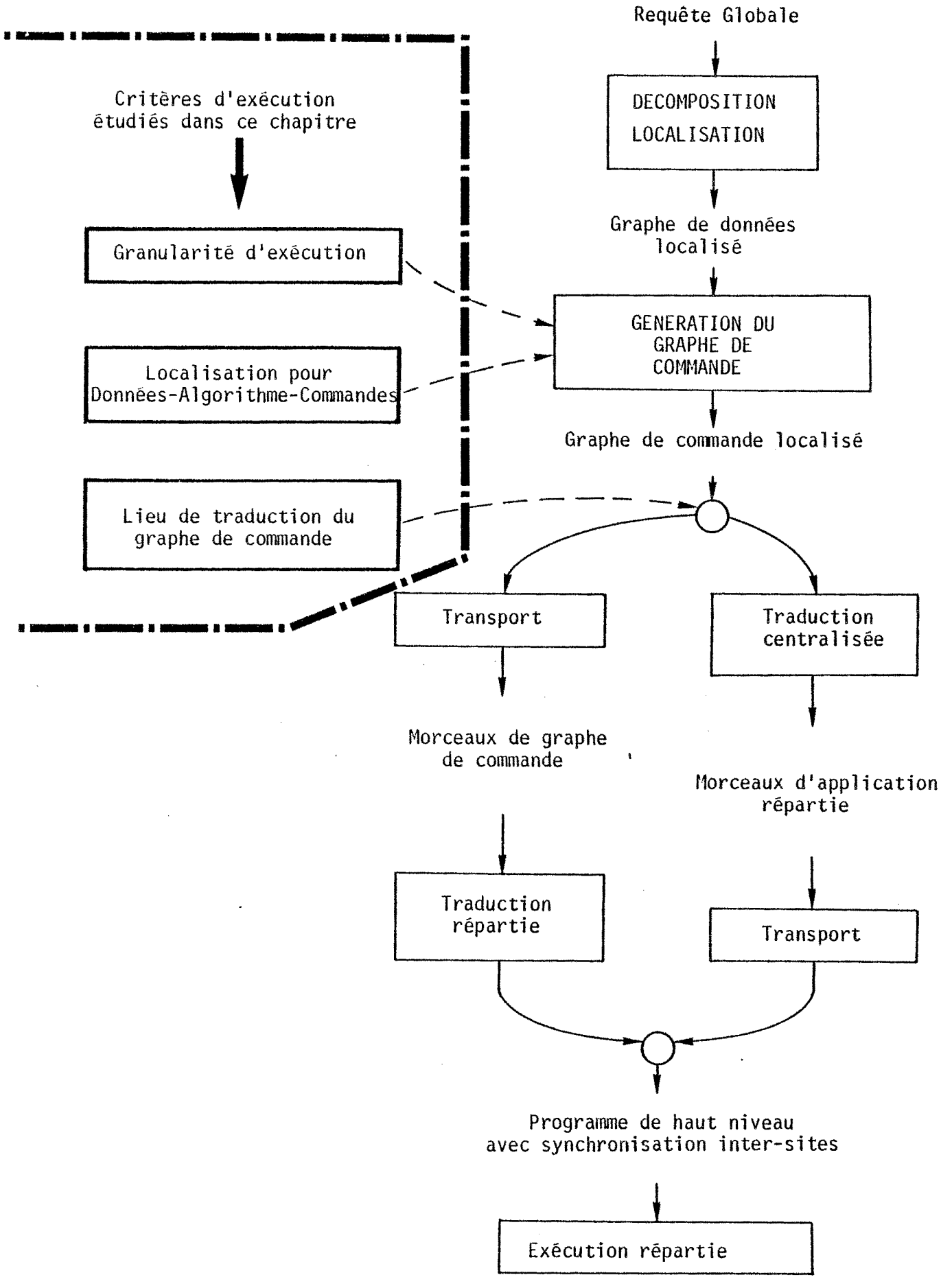


Figure II.1 - Présentation des critères d'exécution étudiés dans ce chapitre

II-1 Lieu de traduction des graphes de commande localisés

Le traitement d'une requête par le SGBDR peut être réalisé de deux manières différentes :

- soit *traduction centralisée* regroupant successivement une traduction au niveau global en un ensemble de programmes en langage de haut niveau, leur éventuel transport sur les machines adéquates et leur activation ;
- soit *traduction répartie*, c'est-à-dire successivement : découpage en un ensemble synchronisé de sous-graphes de commande localisés, leur transport sur les différentes machines, la traduction de ces morceaux sur leur site et leur activation.

Ce terme de traduction recouvre en fait deux types possibles d'exécution :

- soit une *interprétation* du graphe de commande localisé,
- soit une *génération* (compilation d'un graphe de commande localisé en un programme réparti) suivie de sa *compilation* et de son *activation*.

Nous allons étudier ces différentes combinaisons.

II-1-1 Interprétation du graphe de commande localisé

Que cette interprétation soit centralisée ou répartie, elle repose de toute manière sur un ensemble de programmes implantés sur le système réparti ; ces programmes écrits dans le langage d'application répartie du réseau réalisent :

- le transport des données, des commandes et des graphes ;
- l'activation de ces commandes avec retour de compte rendus d'exécution ;
- l'interprétation des graphes de commande localisés.

Cette interprétation revient à faire évoluer une ou plusieurs "marques" dans ce graphe avec des conventions du type suivant : (III.2.3.1)

- lorsqu'une marque arrive dans un noeud, la commande correspondante est activée ;
- quand l'exécution d'une commande est achevée, la marque du noeud correspondant suit l'arc sortant pour aller vers le noeud suivant.

Nous détaillerons plus loin les conventions d'interprétation en fonction des différents types de graphes de commande utilisés (Chapitre III).

En fonction du volume de code d'interprétation admissible sur les différentes machines locales, les deux cas de figure se présentent :

- soit le graphe de commande localisé est analysé par un interpréteur unique sur le site initial qui lance des commandes prédéfinies sur les sites distants (Figure II.2.a) ; c'est la traduction centralisée avec interprétation du graphe de commande localisé ;

- soit ce graphe de commande localisé est fractionné en plusieurs sous-graphes synchronisés entre eux correspondant chacun à un site donné ; ces graphes de commande sont alors véhiculés sur le réseau pour être analysés par des interpréteurs locaux (Figure II.2.b) ; c'est la traduction répartie avec interprétation.

II-1-2 Génération-compilation-exécution

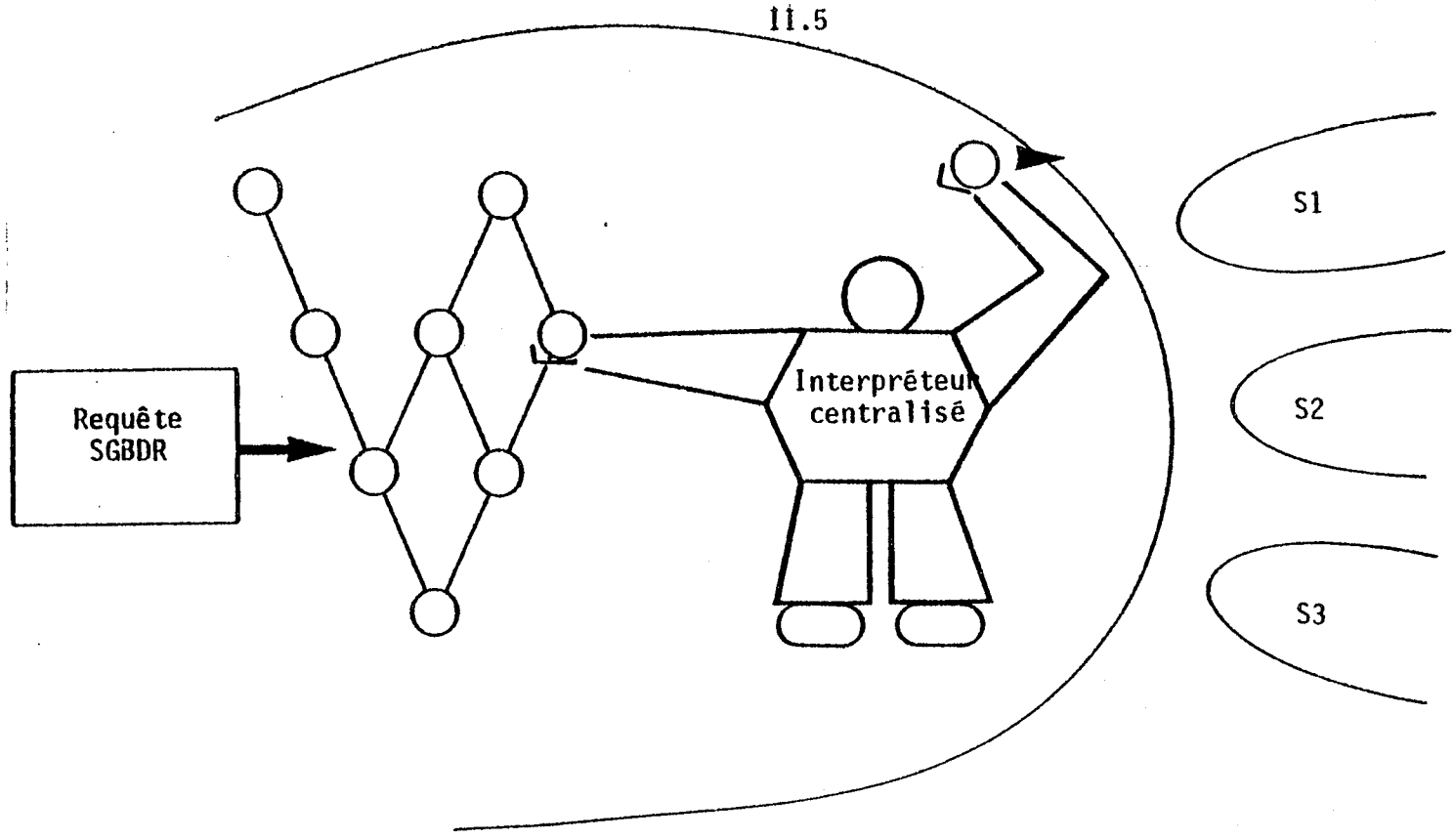
A partir d'un même graphe de commande localisé, il est possible d'envisager plusieurs manières de générer, compiler et activer le programme réparti correspondant sur le système réparti.

Le tableau II-1 donne un aperçu des différentes combinaisons (centralisée ou répartie) qui sont commentées par la suite.

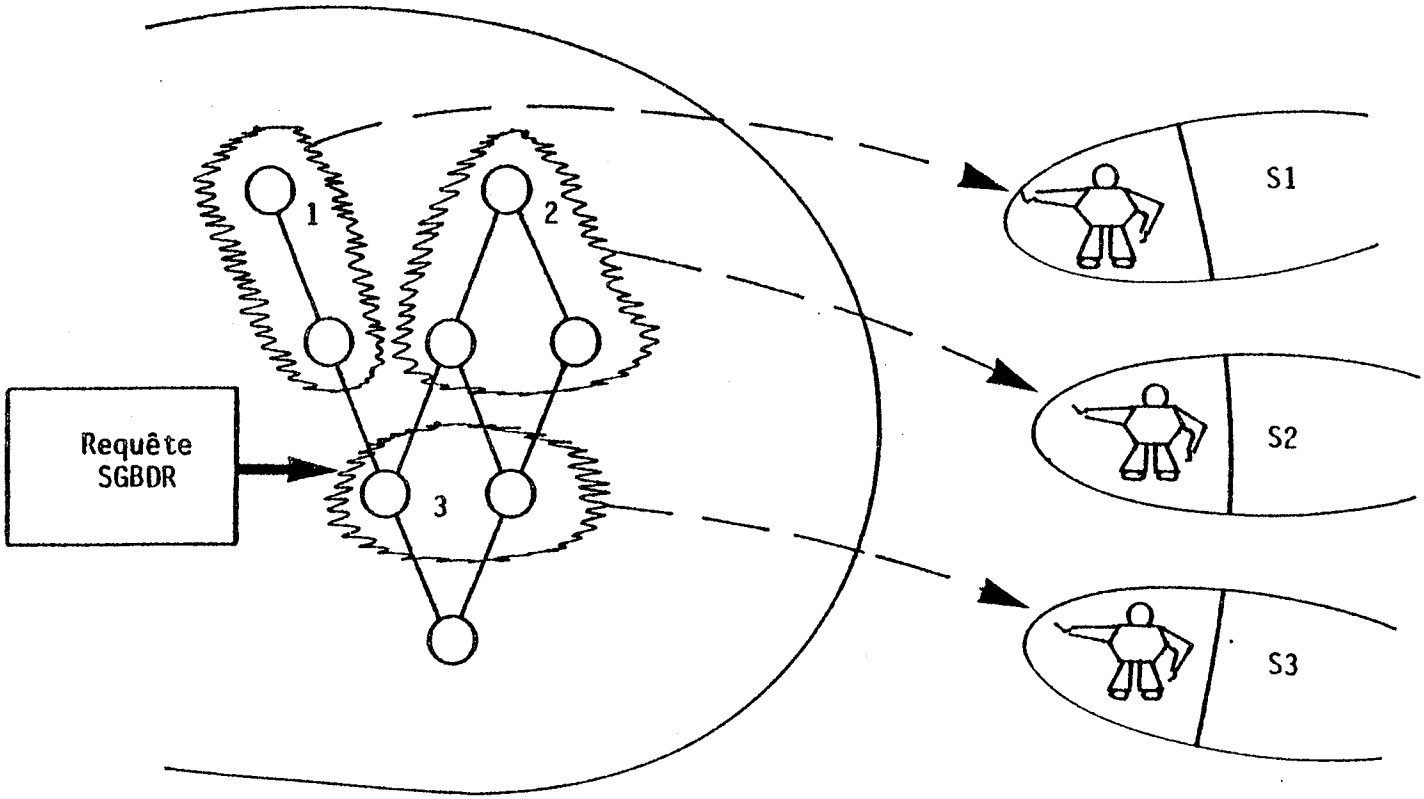
Combinaison	Génération	Compilation	Exécution
1	Répartie	Répartie	Répartie
2	Centralisée	Répartie	Répartie
3	Centralisée	Centralisée	Répartie
4	Centralisée	Centralisée	Centralisée

Tableau II-1 - Combinaisons "centralisé-réparti" pour la Génération-Compilation-Exécution d'un Graphe de commande localisé

Combinaison 1 - Le graphe de commande localisé est découpé en sous-graphes monosites au niveau global et comme dans le cas de l'interprétation répartie vue précédemment, ces sous-graphes sont véhiculés sur les sites concernés ; là a lieu la génération du morceau de programme réparti correspondant, puis sa compilation suivie de son activation.



a) L'interpréteur lance une commande après l'autre vers chaque site distant



b) Envoi des morceaux de graphe de commande localisé aux interpréteurs locaux

Figure II.2 - Les deux manières d'interpréter un graphe de commande localisé

Combinaison 2 - Génération centralisée du programme réparti, transport du code en langage de haut niveau (par exemple PL/1 [DEC 781 et 2] ou le "Langage Intermédiaire" de SIGOR [DANSER]) puis compilation locale et activation (en fait interprétation pour le LI de SIGOR).

Combinaison 3 - Génération et compilation centralisées suivies du transport du code objet correspondant pour une activation sur les sites locaux ; on perd ici l'avantage d'un langage de haut niveau compilé localement qui permettait d'avoir des machines différentes (hétérogènes) d'un site à l'autre.

Combinaison 4 - Génération, compilation et activation centralisées du programme résultant ; les commandes concernant les sites distants font alors l'objet d'activations de procédures via le réseau de la même manière que dans l'interprétation centralisée présentée au paragraphe précédent.

Dans les trois premiers cas où l'activation est effectivement répartie, la synchronisation entre les modules des différentes machines se fait au moyen des outils fournis par le système réparti :

- principe de "Rendez-vous" entre deux modules qui s'attendent mutuellement [DANSER] ;
- attente d'activation de procédures à distance [DEC 781 et 2].

Ce système de génération pose de gros problèmes de mise au point d'un *générateur automatique* lorsque la syntaxe du langage d'application répartie est celle d'un langage de haut niveau (type PL/1).

De plus, la gestion des erreurs doit être prévue dans le code généré en fonction des problèmes particuliers que peut poser l'exécution sur chaque machine locale.

D'autre part, l'hypothèse d'hétérogénéité des SGBD locaux dans laquelle nous nous sommes placés implique que le générateur d'application répartie soit effectivement capable de prendre en compte des modes de manipulation différents d'un SGBD à l'autre.

Enfin, tous les langages d'application répartie ne sont pas conçus comme le LI (Langage Intermédiaire) de SIGOR pour être véhiculés sous forme compacte au travers du réseau [DANSER].

II-2 Granularité d'exécution

Le transport inter-site des données et leur manipulation sur les différentes machines implique le choix d'une unité atomique d'information qui fixe ce que nous appelons la *Granularité d'exécution*.

On détermine trois niveaux de granularité :

- une expression relationnelle par site impliquant plusieurs relations de cette machine ;
- une opération par relation ou paire de relation ;
- une opération par n-uplet ou paire de n-uplet.

Le niveau de granularité choisi implique une finesse plus ou moins grande du graphe de commande correspondant aux différents modes d'exécution :

- le niveau "Expression relationnelle par site" est assimilable au mode transactionnel ;
- le niveau "Une opération par relation" correspond au mode total ;
- le niveau "Une opération par n-uplet" est à associer au mode sériel (ou "pipe-line") ; il faut toutefois bien remarquer que, comme nous l'analysons plus loin, certaines opérations ne sont possibles qu'en mode mixte (une relation complète est disponible sur les deux), voire même uniquement en mode total [PAIK]. (II.2.3).

Nous allons détailler les caractéristiques de chacun de ces niveaux.

II-2-1 Une expression relationnelle par site

Il s'agit donc de transporter sur le réseau toute une expression relationnelle impliquant l'exécution d'opérateurs portant sur des relations locales déjà sur ce site ou venant d'autres sites.

Lorsqu'il s'agit d'une interrogation, le résultat est une relation créée temporairement sur le site local alors que dans le cas d'une mise à jour, le résultat est un booléen indiquant que l'opération a été ou non correctement exécutée localement.

Sur le plan de la synchronisation, on se ramène, entre le site demandeur et le site local, à une association de type producteur-consommateur.

Une telle granularité d'exécution revient, au niveau de chaque site, à l'enchaînement présenté dans la figure II.3.

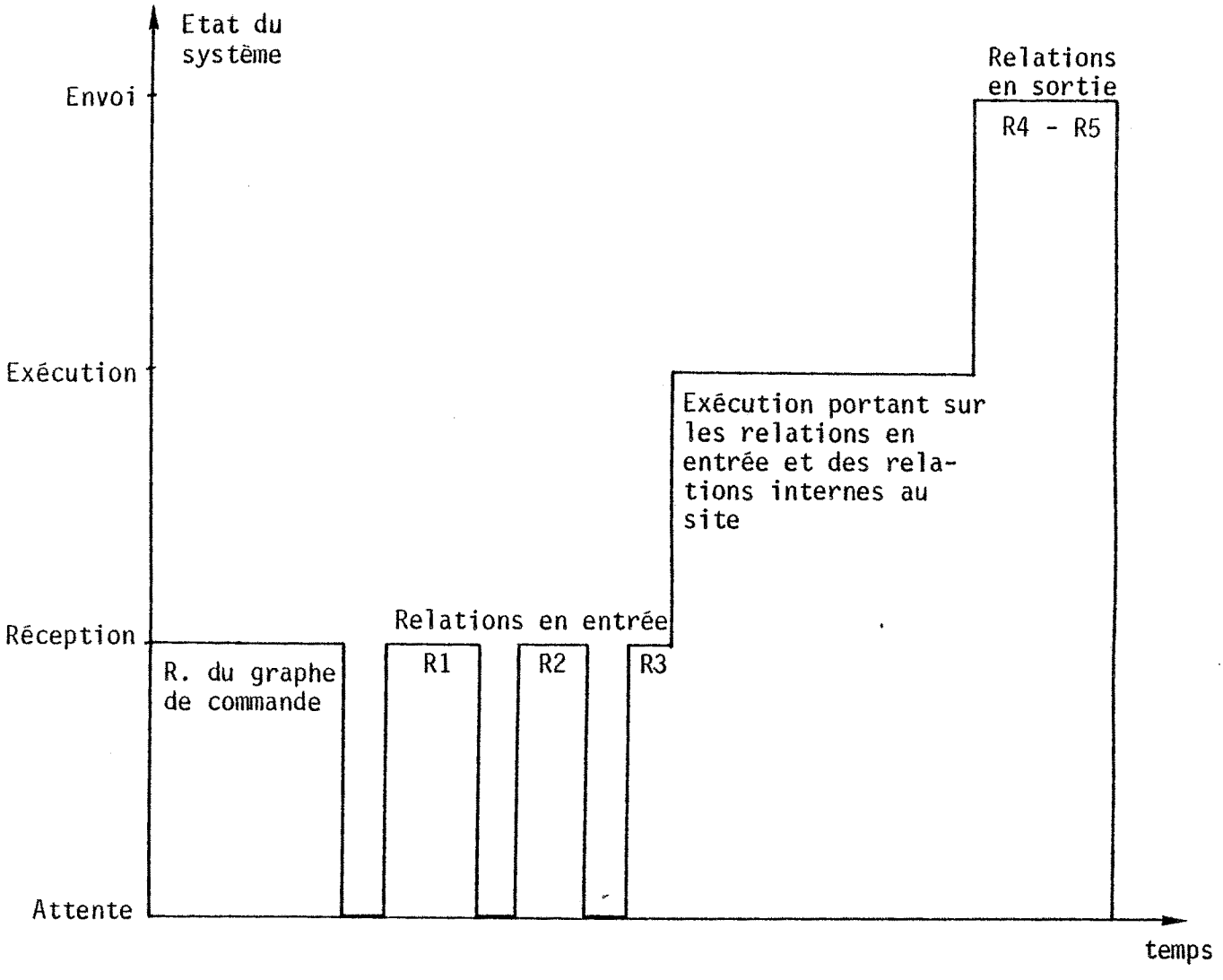


Figure II.3 - Chronogramme schématisant l'exécution d'une expression relationnelle sur un site local

Dans cette stratégie, les relations en entrée venant d'autres sites ne peuvent être utilisées que lorsqu'elles sont totalement disponibles sur le site ; la synchronisation se fait donc sur la notion de "fin de relation".

Lorsque le système de transport a annoncé "Fin de relation" pour toutes celles qui sont en entrée, alors, et alors seulement, peut commencer l'exécution proprement dite.

Il est facile de voir qu'avec un tel système, un interblocage entre deux sites peut aisément se produire.

Supposons l'exemple suivant :

Site 1 :
Relation en entrée :
A
Relation en sortie :
B

Site 2
Relation en entrée :
B
Relation en sortie :
A

Ce type d'imbrication est parfois difficile à déceler quand il se situe dans des requêtes complexes. Il faut cependant remarquer qu'un tel cas ne peut apparaître dans le contexte mono-utilisateur de la machine globale dans lequel nous raisonnons par ailleurs ; en effet, l'utilisateur global qui s'exprime en terme de "chemin de données arborescent" dans ses requêtes relationnelles au SGBDR ne peut *dans une seule requête* créer une telle imbrication :
les deux requêtes locales seront sérialisées dans un sens ou dans l'autre :

Requête 1 puis Requête 2
ou Requête 2 puis Requête 1

le résultat final dépendant bien sûr de l'ordre choisi par l'utilisateur global.

II-2-2 Une opération par relation

La description du graphe de commande localisé est ici plus fine que dans le cas précédent :

- les relations intermédiaires sont prédéfinies par la Machine Globale ;
- les synchronisations ne se font plus au niveau d'un ensemble de relations d'entrée, mais plutôt par opérateur attendant des relations extérieures (voir figure II.4) ;
- de même, les relations de sortie peuvent être envoyées vers les sites distants dès qu'elles sont entièrement prêtes.

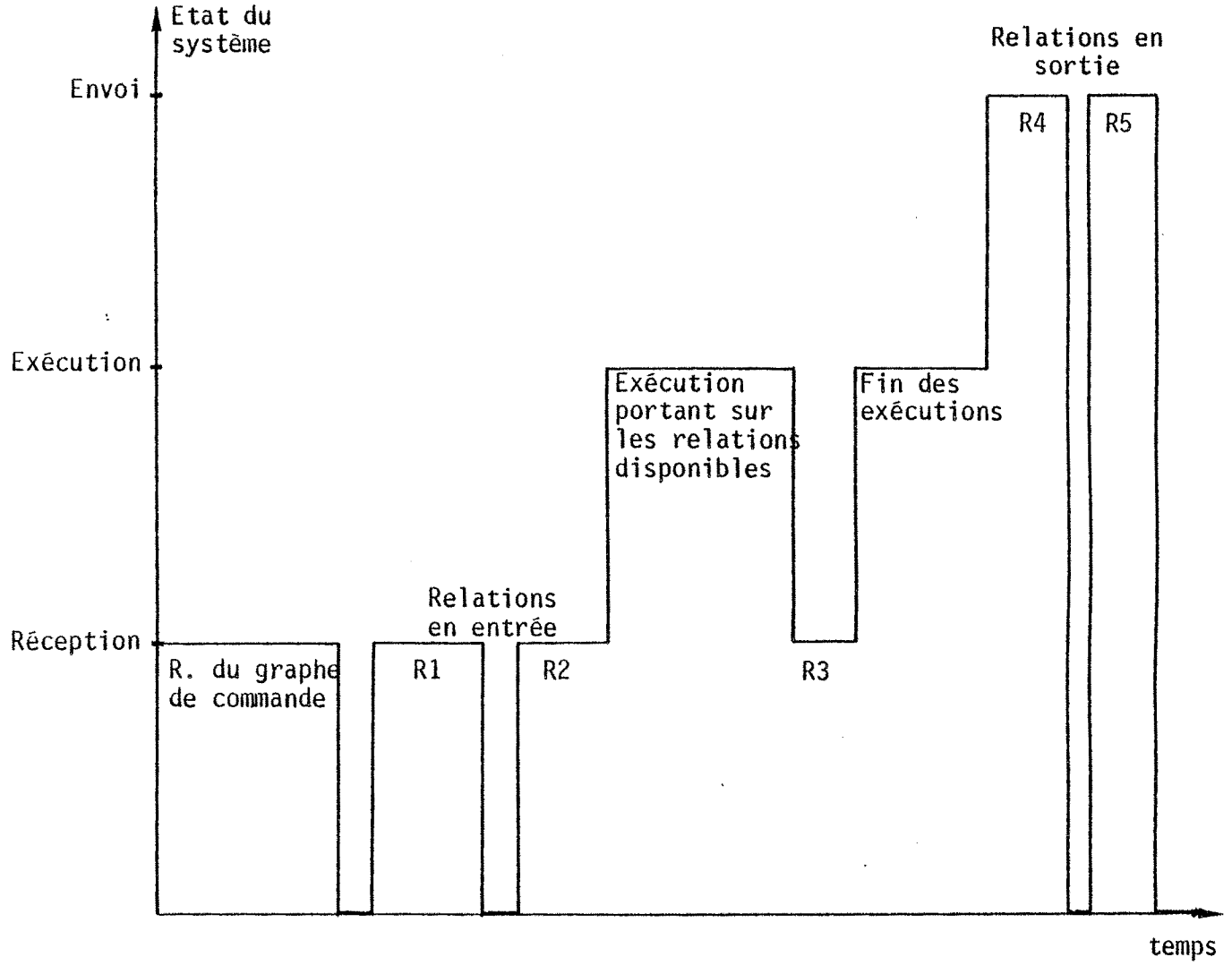


Figure II.4 - Chronogramme schématisant l'exécution d'un graphe de commande localisé composé d'opérations portant sur des relations complètes

II-2-3 Une opération par n-uplet

C'est le niveau de granularité le plus fin qui puisse se concevoir ; le graphe de commande localisé contient alors des opérations relationnelles et de synchronisation qui concernent des n-uplets et des "fin de relation".

Des n-uplets des relations de sortie peuvent être produits alors que le graphe de commande consomme toujours des n-uplets des relations d'entrée. Ce type d'exécution est aussi appelé "mode sériel" (ou "pipe-line").

Toutes les opérations relationnelles ne peuvent être exécutées en mode sériel. Ainsi certains opérateurs binaires nécessitent en entrée la totalité de l'une des relations pour produire les n-uplets de sortie en consommant, au fur et à mesure de leur arrivée, les éléments de la deuxième relation (mode mixte en entrée, sériel en sortie) ; par exemple, on peut citer l'intersection de deux relations.

D'autres opérateurs pourront consommer au fur et à mesure les n-uplets d'entrée mais le résultat n'apparaîtra qu'en fin de relation (mode sériel en entrée, total en sortie) ; par exemple, le cardinal d'une relation.

Enfin, la division de deux relations est l'exemple d'une opération nécessitant la présence des deux entrées complètes pour fournir le résultat : c'est le mode total en entrée et donc aussi en sortie.

Une récapitulation des opérateurs relationnels avec pour chacun le mode le plus fin pouvant s'appliquer est donnée dans le tableau II-2.

Opérations relationnelles		Mode applicable en entrée	Mode applicable en sortie
DIVISION (X,Y)		Total	Total
COMPOSITION (X,Y)	R-JOIN (X,Y)	Mixte	Sériel
	L-JOIN (X,Y)	Mixte	Sériel
	-JOIN (X,Y)	Mixte	Sériel
	-JOIN (X,Y)	Mixte	Sériel
DIFFERENCE : X-Y, Y présent		Mixte	Sériel
INTERSECTION (X,Y)		Mixte	Sériel
UNION (X,Y)		Sériel	Sériel
SELECT (X)		Sériel	Sériel
PROJECT (X)		Sériel	Sériel
CARDINAL (X)		Sériel	Total

Tableau II-2 - Modes d'exécution applicables aux différents opérateurs relationnels

Il est évident que tout opérateur qui peut être sériel est utilisable en mode mixte ou total ; de même, un opérateur de type mixte est exécutable en mode total.

En fin d'exécution n-uplet par n-uplet, il faut savoir si toute la relation résultante est effectivement disponible. Pour cela, il faut *répercuter* les fins d'opération ; en effet, une opération du graphe de commande ne sera totalement terminée que si *toutes* les exécutions en amont ont annoncé

la fin de leur relation de sortie (voir figure II.5).

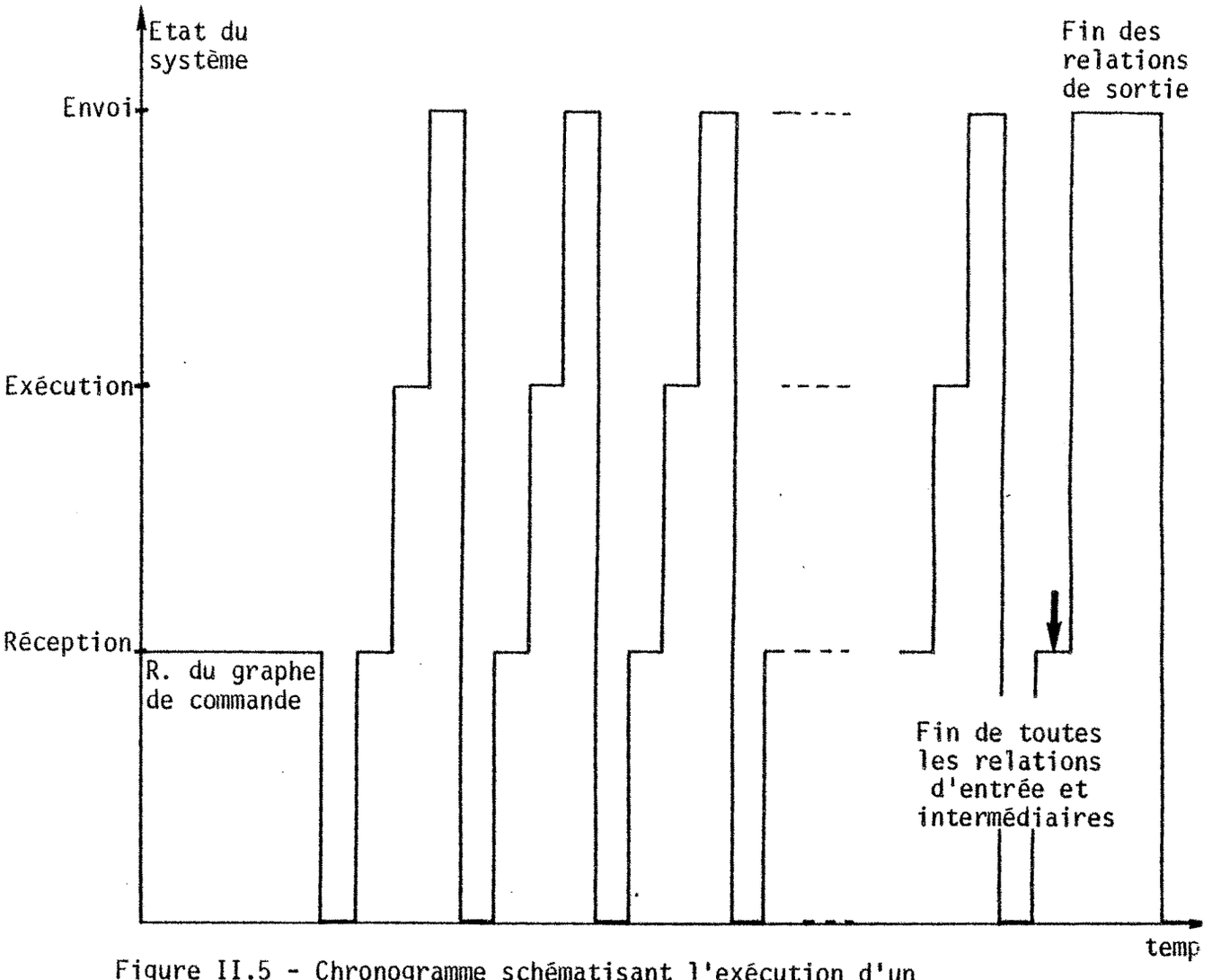


Figure II.5 - Chronogramme schématisant l'exécution d'un graphe de commande localisé avec des opérations travaillant n-uplet par n-uplet en mode mixte ou sériel

Il est important de noter que si ce type d'exécution semble satisfaisant pour l'esprit - "Dès qu'un n-uplet est disponible, on s'en sert !" - il est en fait très complexe à mettre en oeuvre. En particulier, dans le cas de pannes dans le SGBDR, le retour à un état cohérent de la BDR sera d'autant plus difficile à réaliser car le nombre de relations concernées à un instant donné par ce type d'exécution peut correspondre à l'ensemble des relations du graphe de commande.

II-3 Répartition et ordonnancement des algorithmes

Un des choix essentiels dans l'architecture d'un SGBDR se situe entre les deux extrêmes suivants [CHUPIN] :

- transporter les données vers les algorithmes qui doivent les traiter ;
- transporter les algorithmes vers les données à manipuler.

Entre ces deux extrêmes, il est possible de faire de multiples choix pour la construction et l'exécution des graphes de commande localisés. Nous allons en présenter quelques-uns dans la suite de ce paragraphe dans une progression allant du transfert total des données vers le transfert complet des algorithmes.

Nous inclurons également le problème du système de commande qui peut être centralisé ou non.

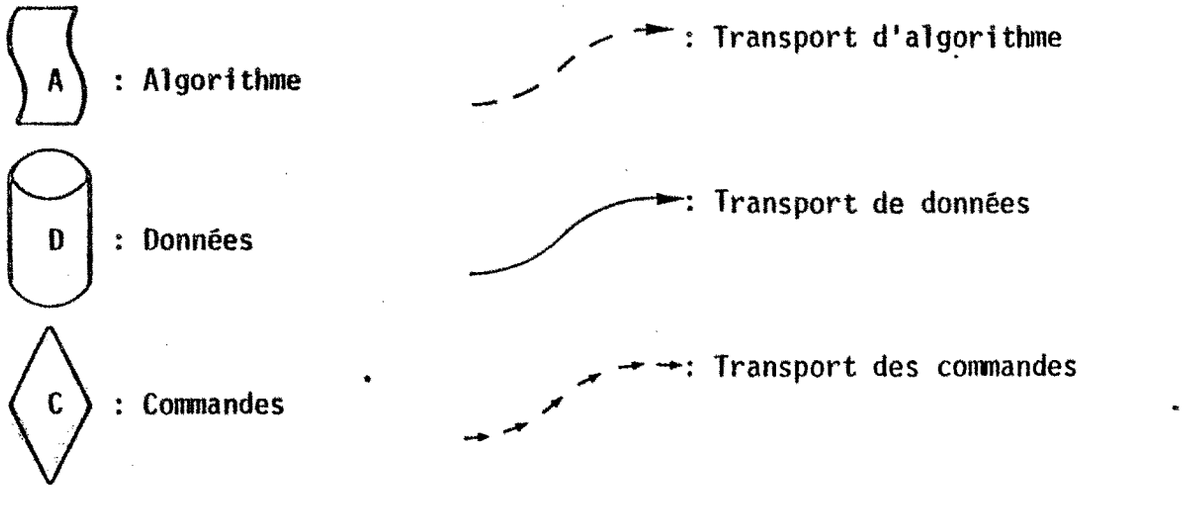
Lors de l'exécution d'une requête concernant la BDR, le système doit combiner les éléments suivants :

- algorithmes (opérateurs et commandes d'accès et de manipulation des données) ;
- données (structures et occurrences à manipuler) ;
- commandes (lancement d'opérations et récupération des comptes rendus d'exécution pour enchaîner éventuellement d'autres opérations).

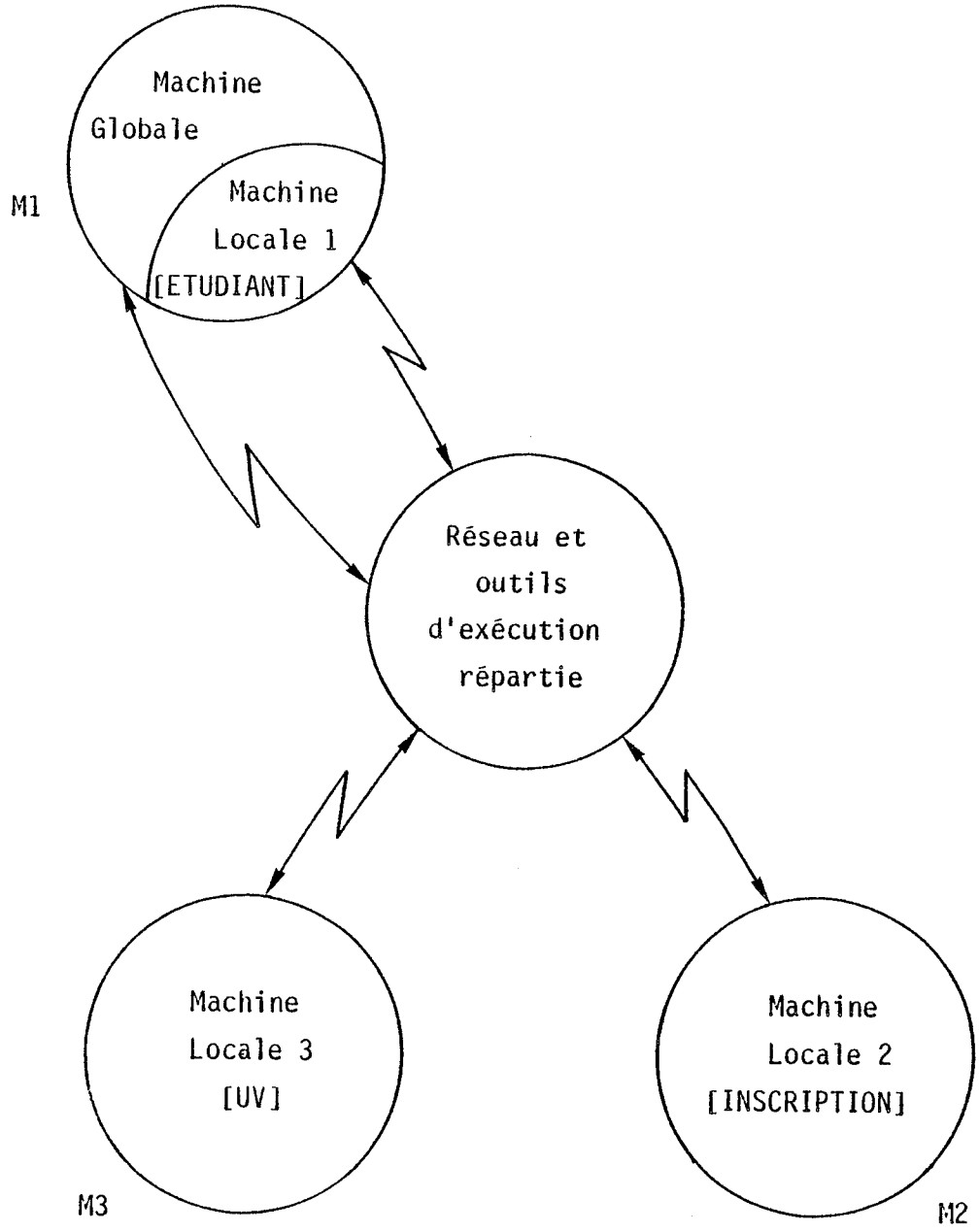
De plus, pour chaque élément, il faut exprimer un éventuel transfert.

La légende suivante sera utilisée pour la description des différents cas d'ordonnancement.

Légende :



L'exemple du chapitre 1 sera utilisé pour concrétiser les différents types d'ordonnancement que nous allons décrire. Nous le rappelons brièvement :



ETUDIANT (NE, NOM, AGE, VILLE)
INSCRIPTION (NE, NUV, NOTE)
UV (NUV, TITRE, CYCLE, RESPONSABLE)

Figure II.6 - Rappel de l'exemple

Nous ne représenterons plus le "réseau et les outils d'exécution répartie" pour simplifier les schémas suivants.

Nous allons traiter la requête globale suivante : "Le Responsable "DUPONT" veut rajuster les notes de ses étudiants en rajoutant deux points à chacune".

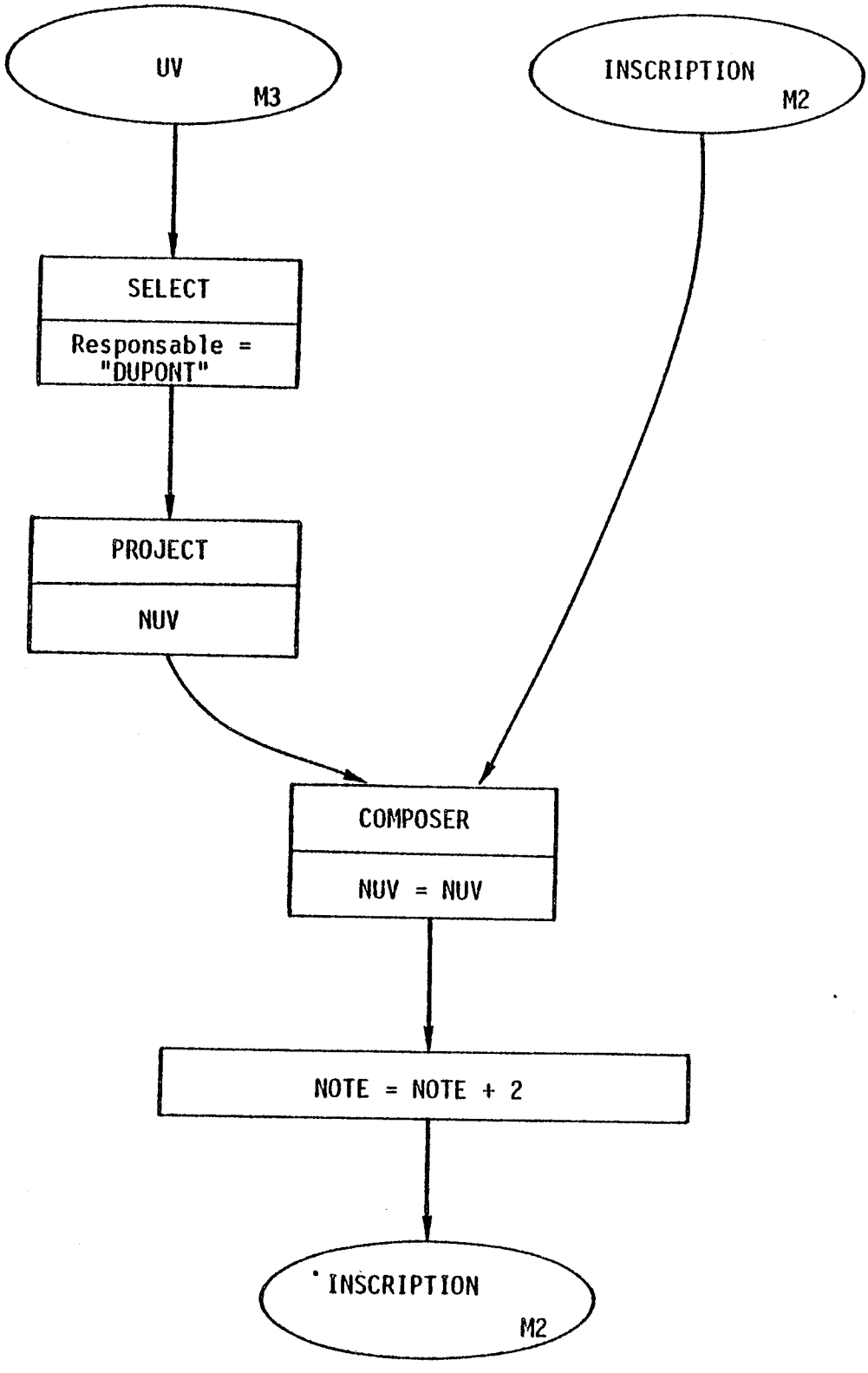


Figure II.7 - Graphe de données correspondant à l'incrémentation de 2 des notes des étudiants dont le responsable d'UV est "Dupont"

Cas 1 : Algorithmes (centralisés) + Commandes (centralisés) + Données (centralisées)

Tous les algorithmes correspondant à une requête se trouvent sur le site ayant reçu cette requête.

Toutes les données concernées par la requête sont alors centralisées sur ce site ; l'exécution effective des algorithmes peut ensuite avoir lieu.

Evidemment, seules les données modifiées sont retournées à leur site d'origine.

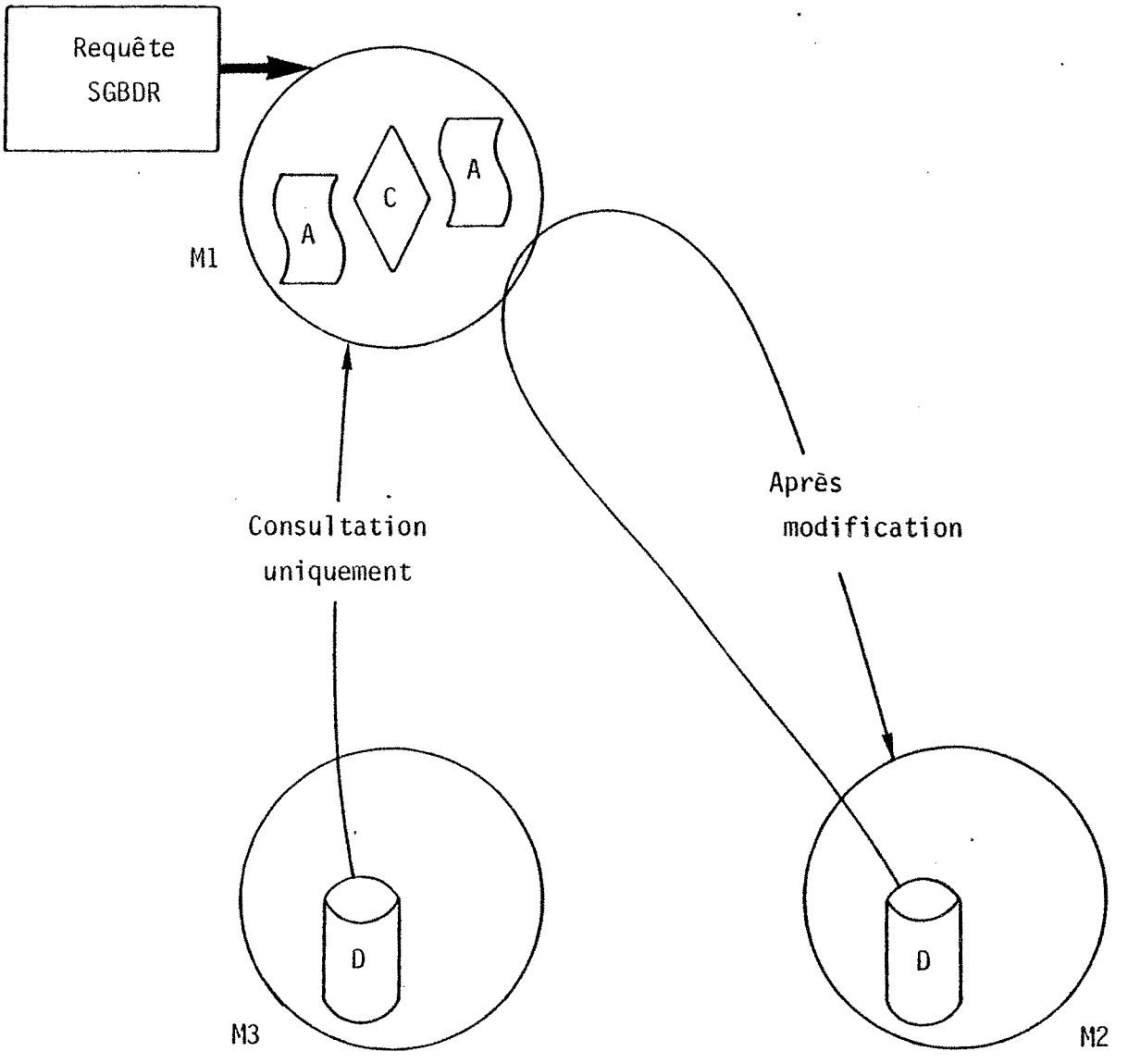


Figure II.8.a - Illustration du Cas 1 sur l'exemple

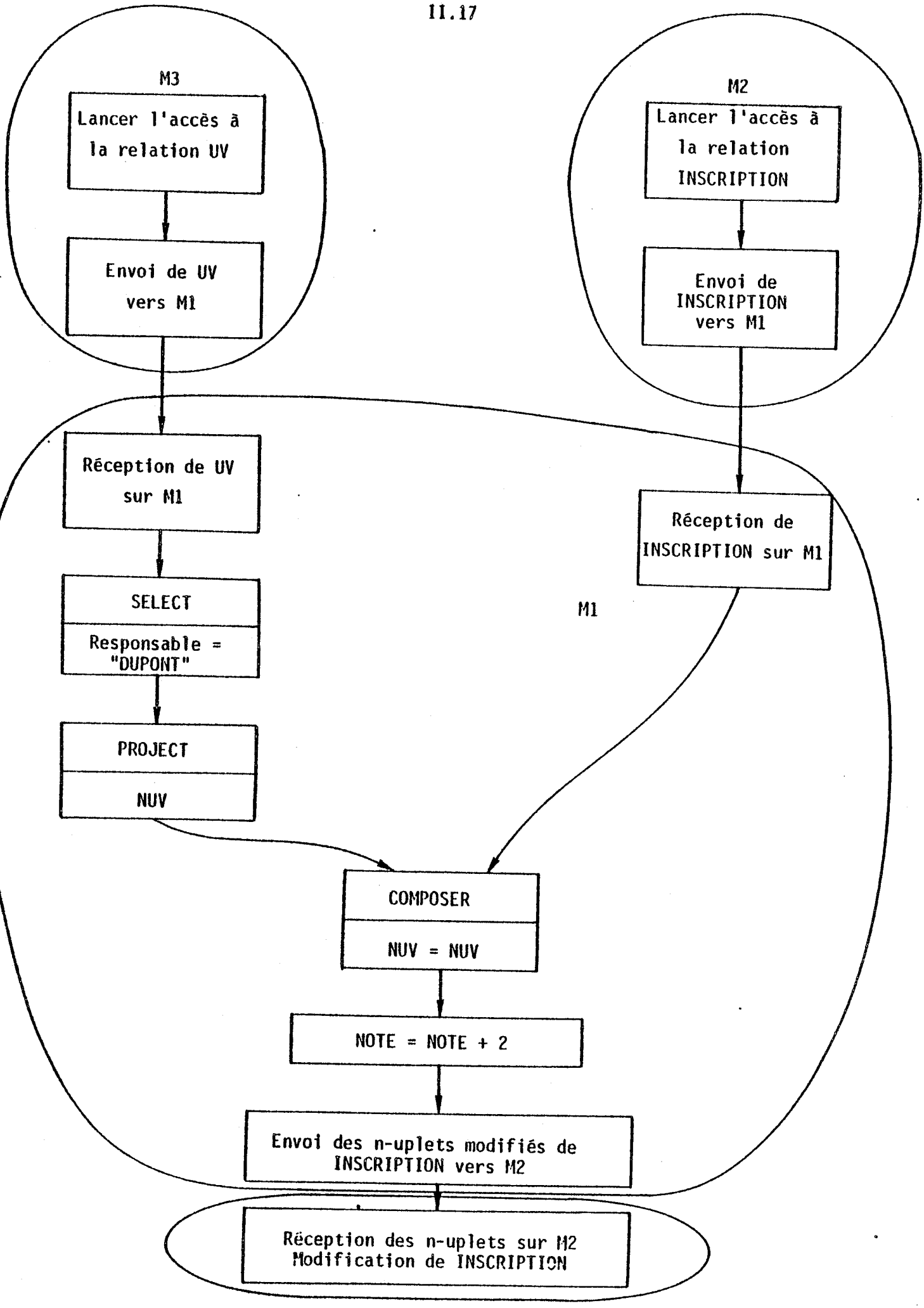


Figure II.8.b - Conséquences du Cas 1 sur l'exécution du graphe de commande localisé

Cas 2 : Algorithmes (transportés) + Commandes (centralisées) + Données (centralisées)

Les algorithmes correspondant à la requête sont envoyés vers les sites et les données résultantes transitent par le site récepteur de la requête ; c'est ce site qui gère la synchronisation des différentes exécutions.

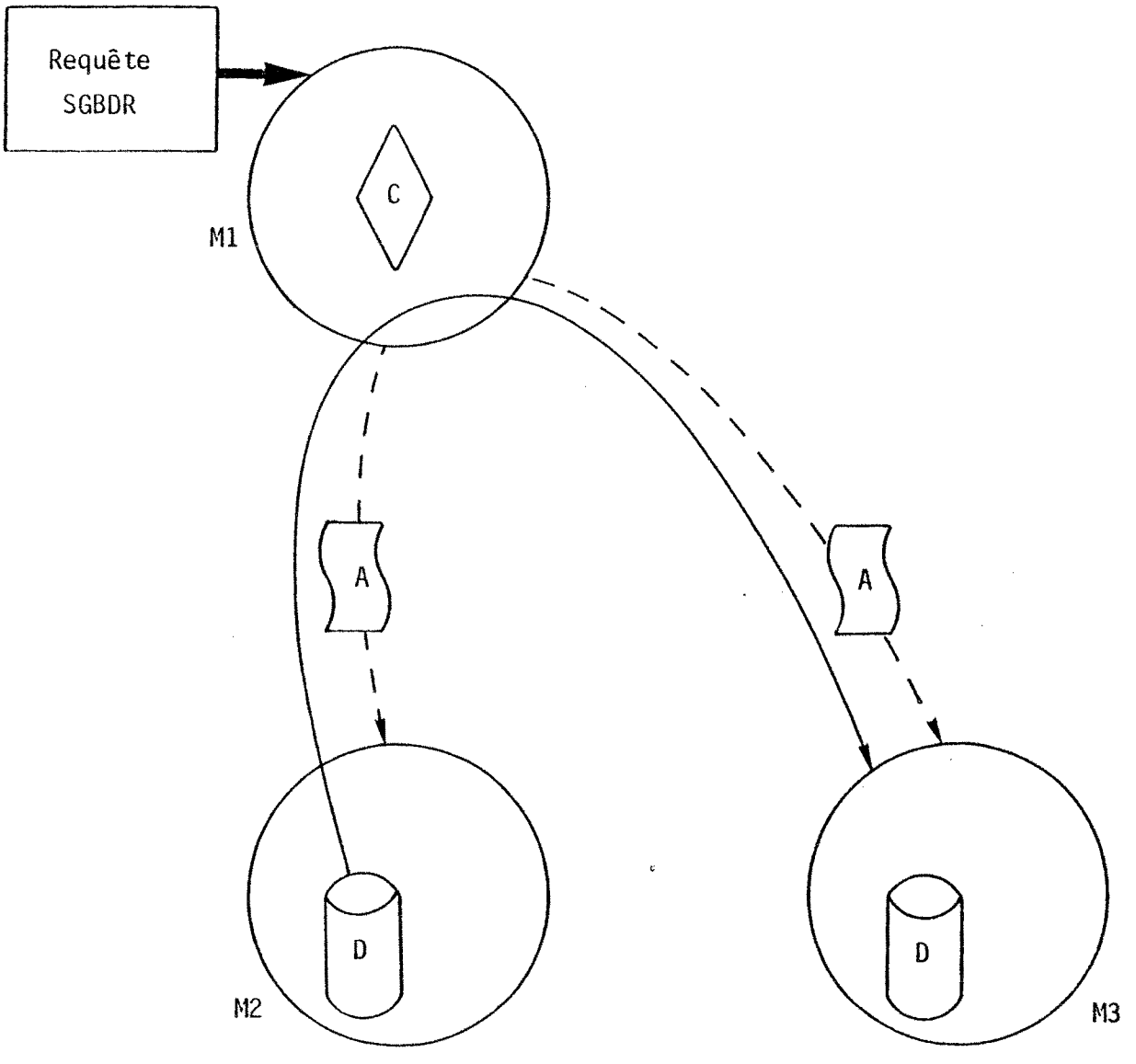


Figure II.9.a - Illustration du Cas 2 sur l'exemple

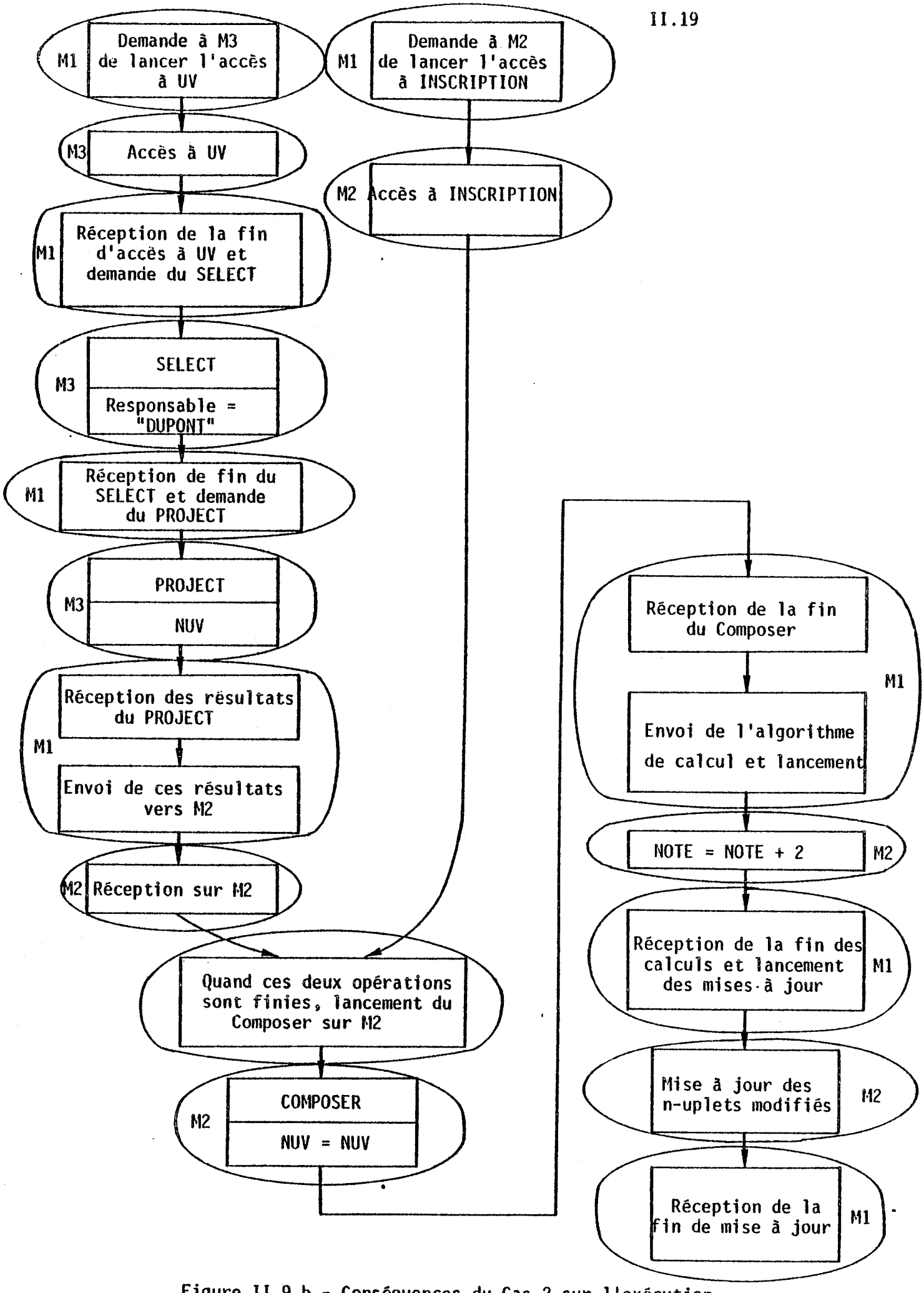


Figure II.9.b - Conséquences du Cas 2 sur l'exécution du graphe de commande localisé

Cas 3 : Algorithmes (transportés) + Commandes (centralisées) + Données (non centralisées)

Comme dans le Cas 2, les algorithmes sont envoyés sur les sites distants mais les données passent directement de machine à machine, selon les besoins ; le système de commande est toujours centralisé sur le site origine.

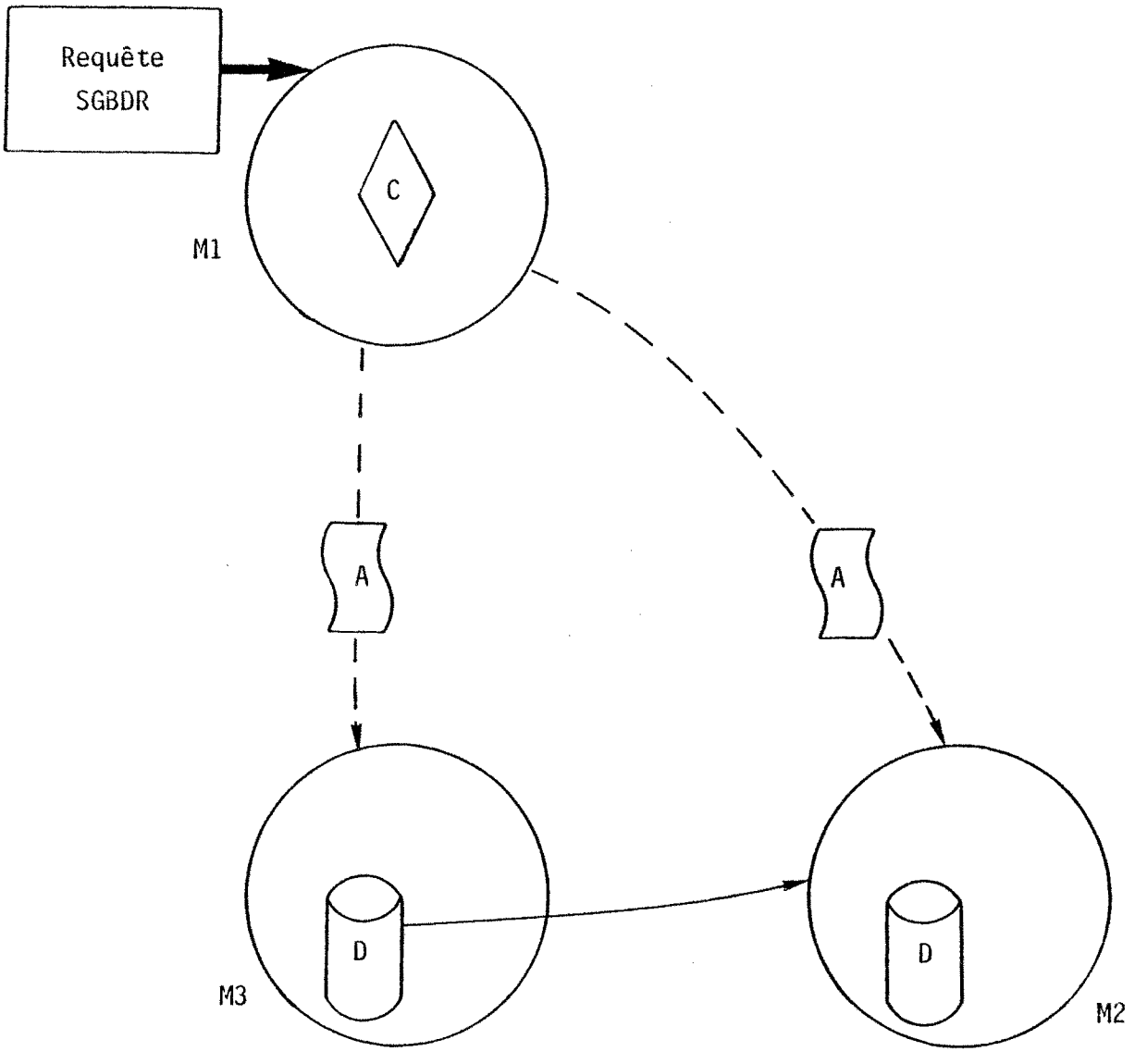


Figure II.10.a - Illustration du Cas 3 sur l'exemple

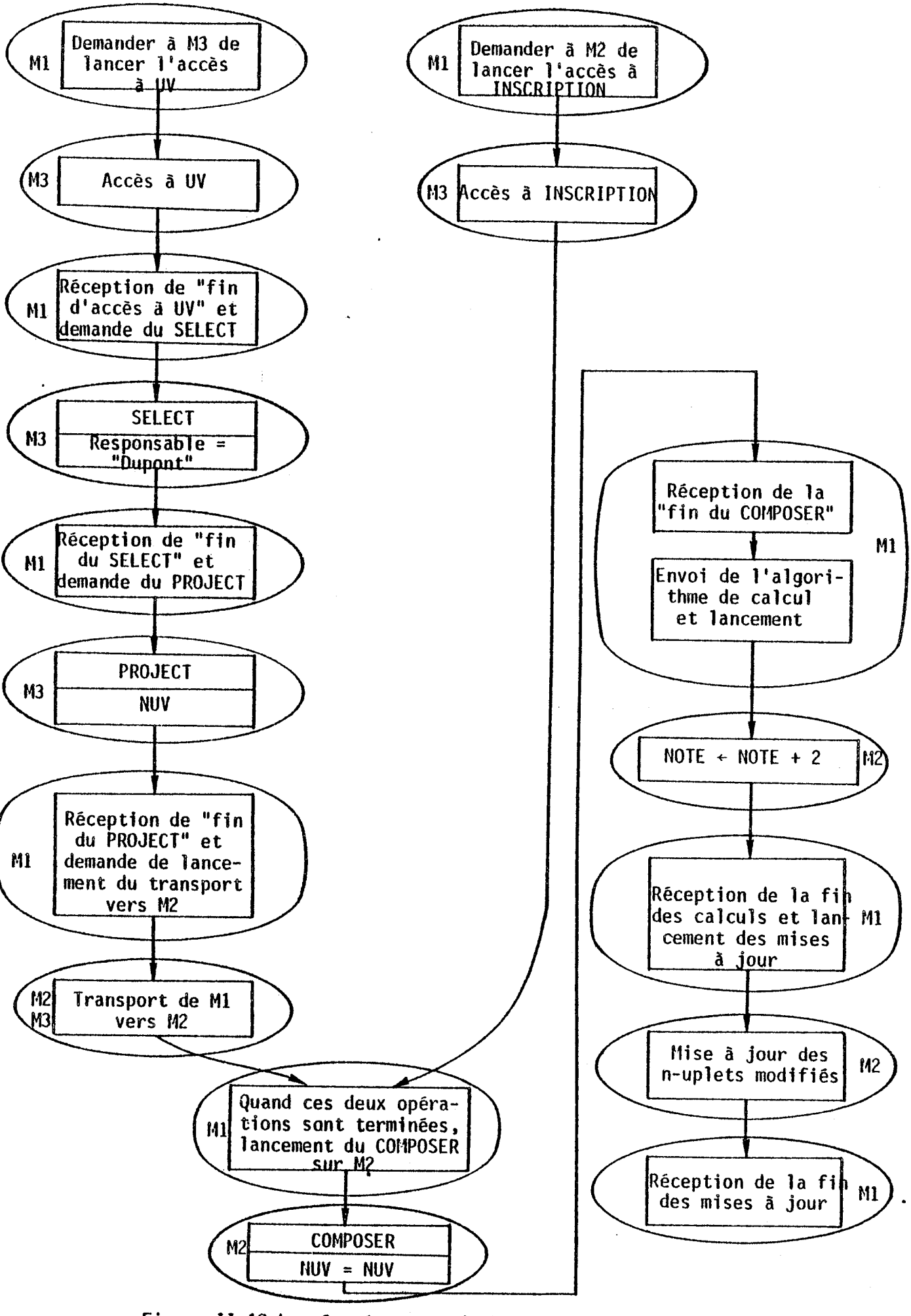


Figure II.10.b - Conséquences du Cas 3 sur l'exécution du graphe de commande localisé

Cas 4 : Algorithmes (implantés) + Commandes (transportées) + Données (centralisées)

C'est un cas qui correspond directement au Cas 2, la différence étant que les algorithmes sont implantés sur les sites distants à la création du SGBDR ; c'est le principe d'activation de programmes locaux dans Polyphème. Le transfert de donnée ainsi que le système de commande sont centralisés sur la machine réceptrice de la requête SGBDR.

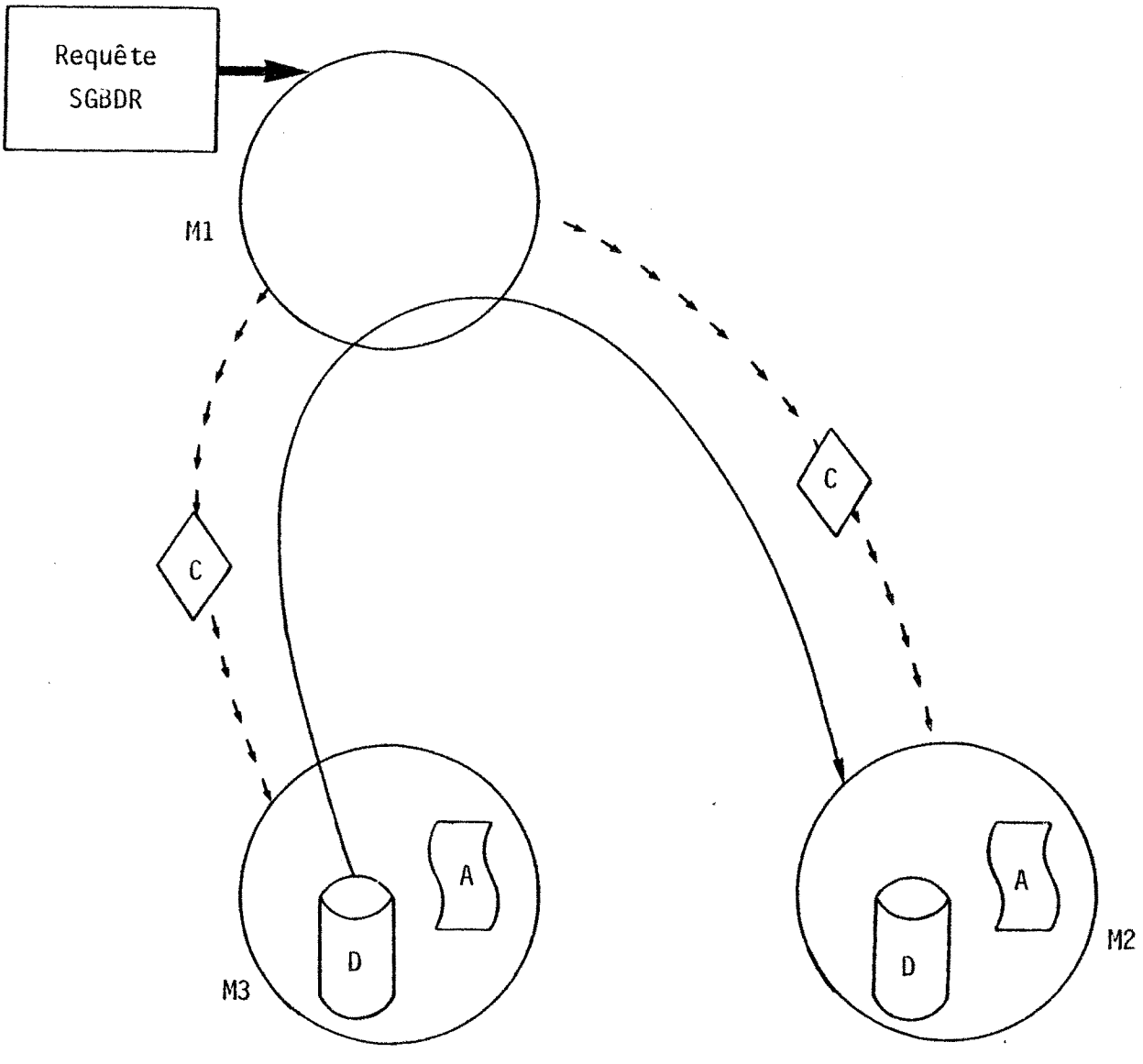


Figure II.11.a - Illustration du cas 4 sur l'exemple

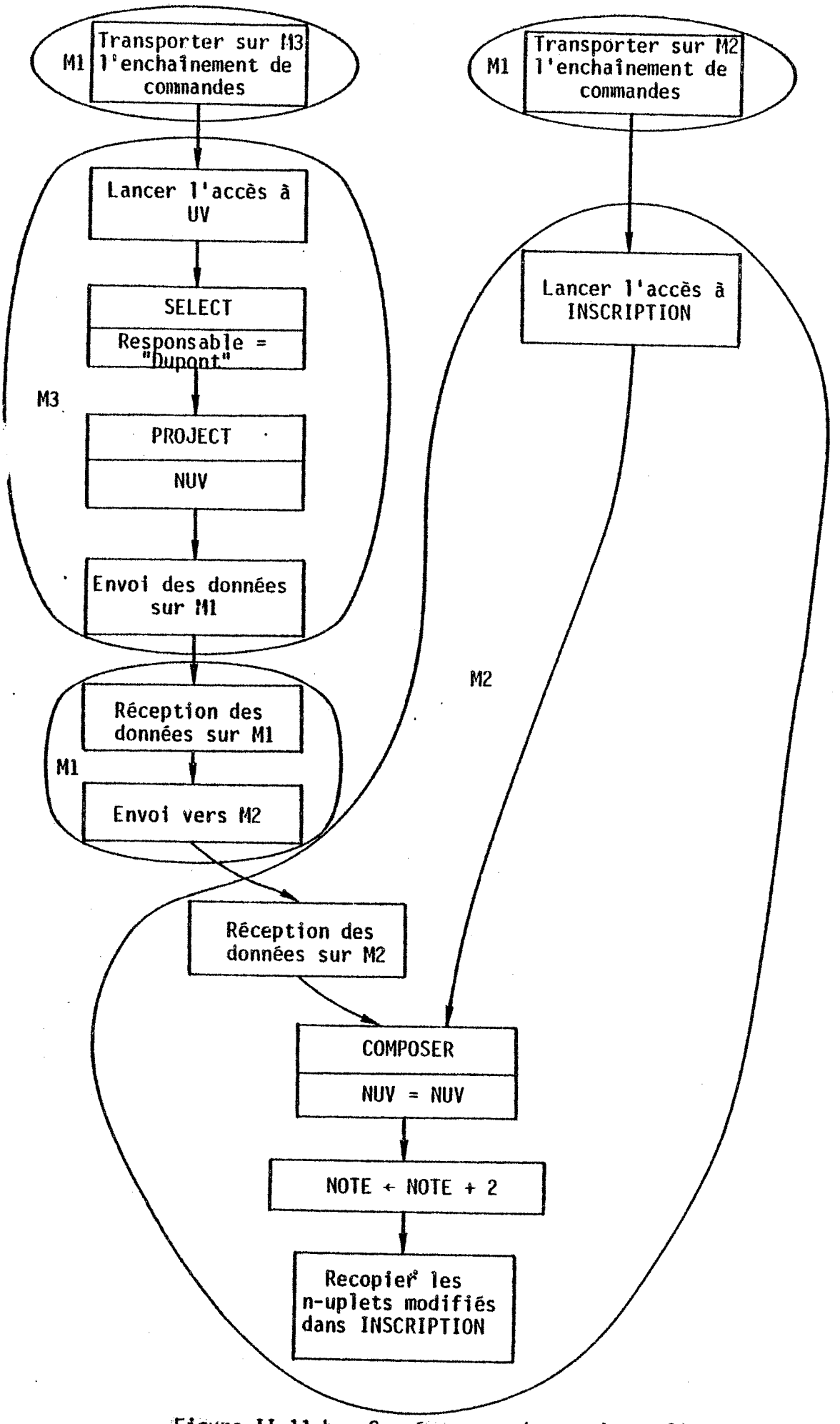


Figure II.11.b - Conséquences du cas 4 sur l'exécution du graphe de commande localisé

Cas 5 : Algorithmes (transportés) + Commandes (transportées) + Données (non centralisées)

Les algorithmes transmis à partir du site origine vers les sites distants contiennent eux-mêmes les synchronisations nécessaires à l'exécution de la requête.

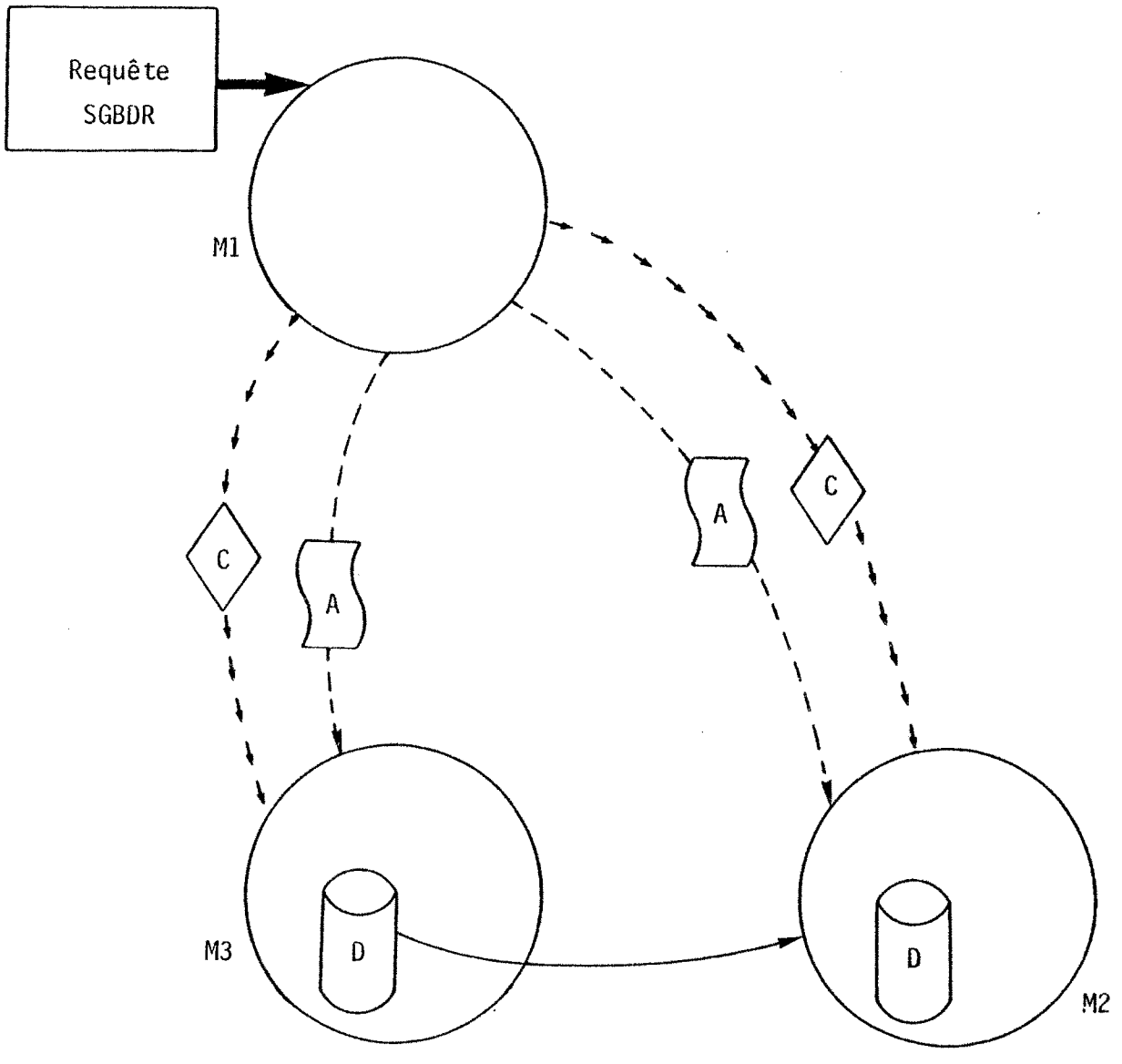


Figure II.12.a - Illustration du Cas 5 sur l'exemple

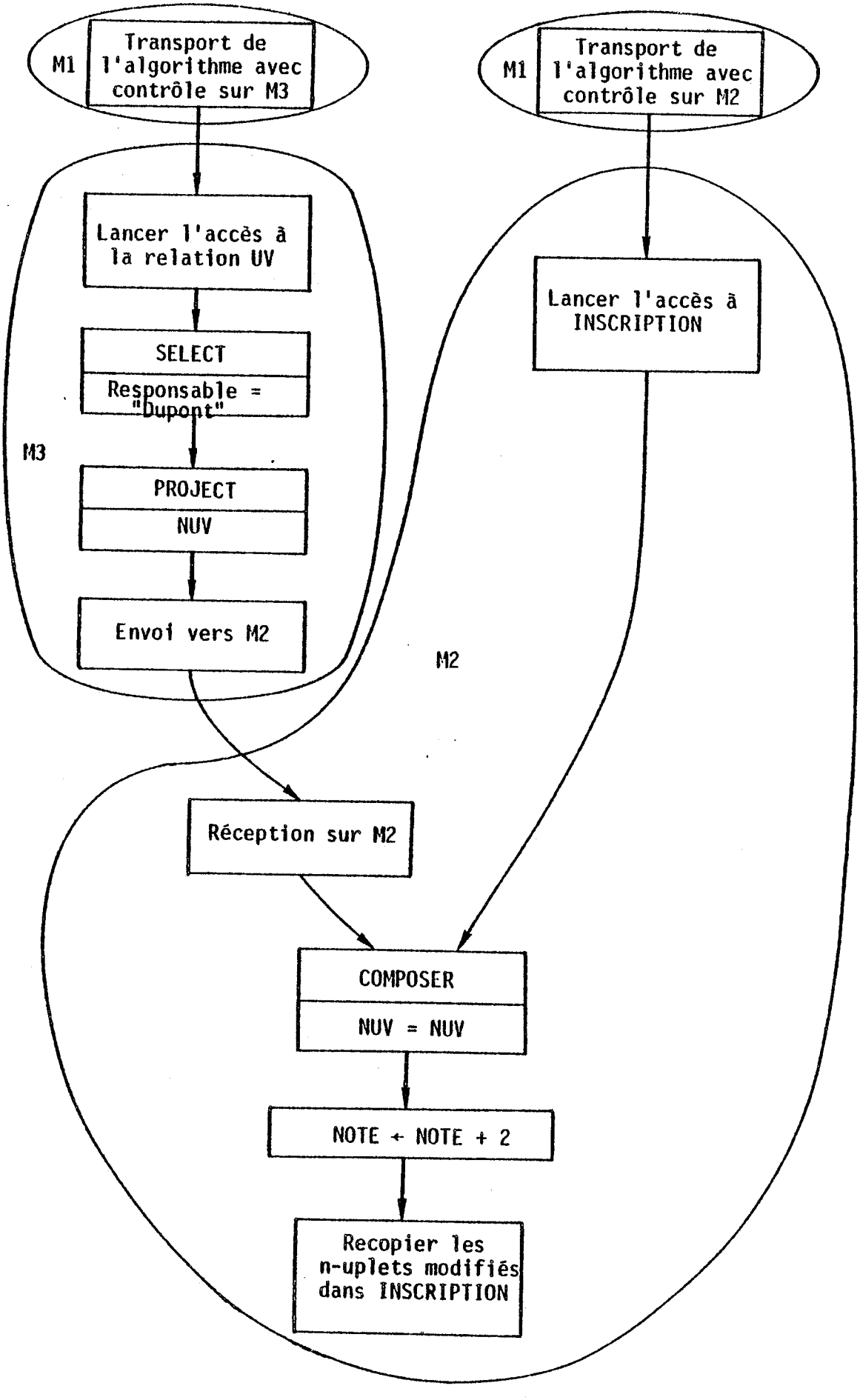


Figure II.12.b - Conséquences du Cas 5 sur l'exécution du graphe de commande localisé

Cas 6 : Algorithmes (Implantés) + Système de commande (réparti) + Données (non centralisées)

Les différents sites à la création du SGBDR contiennent leur enchaînement logique et les synchronisations correspondantes. Cela permet de considérer le SGBDR comme un ensemble de Machines Logiques Relationnelles commandées par des ordres venus du réseau et pouvant elles-mêmes lancer des opérations sur d'autres machines du SGBDR

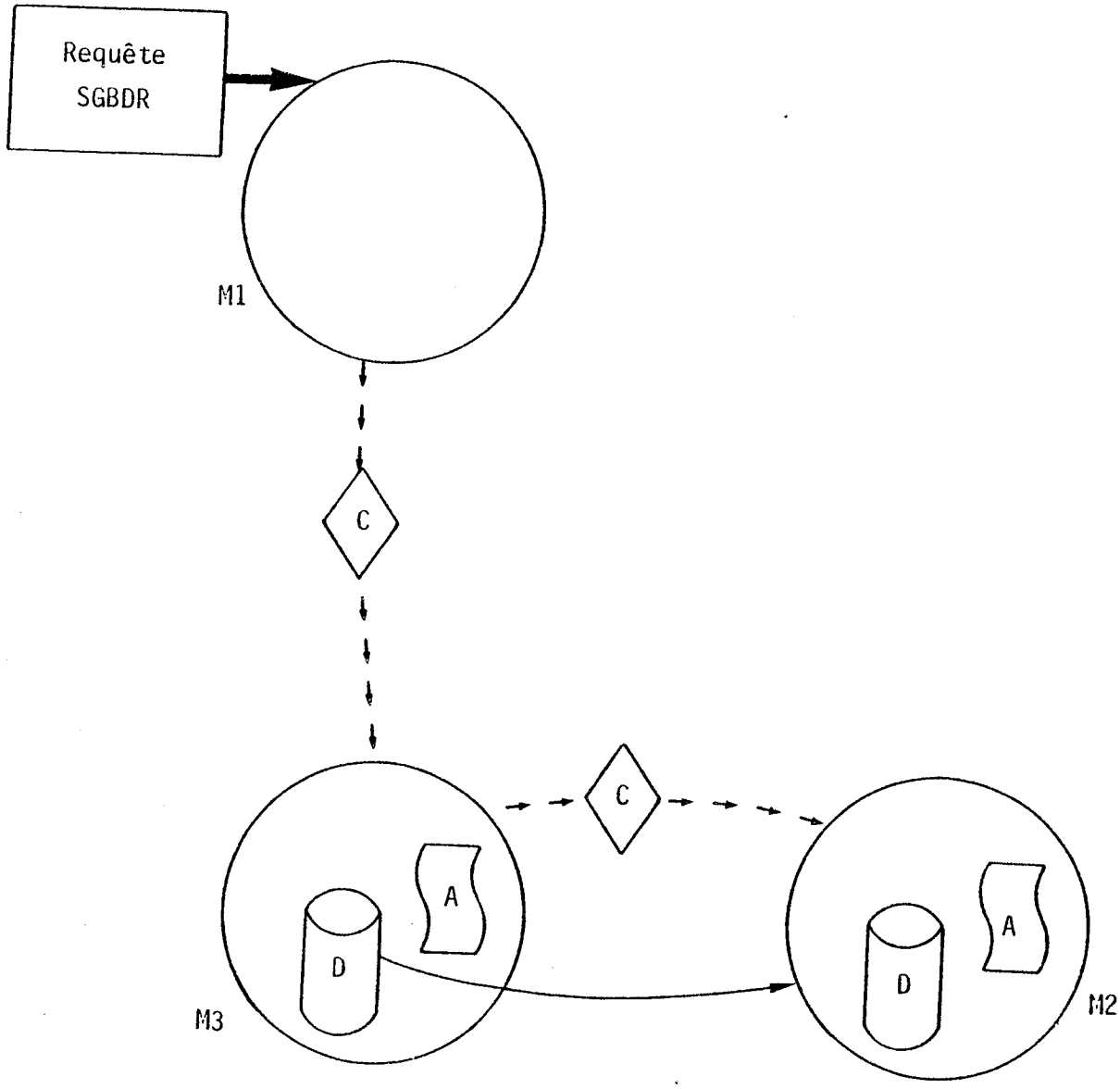


Figure II.13.a - Illustration du Cas 6 sur l'exemple

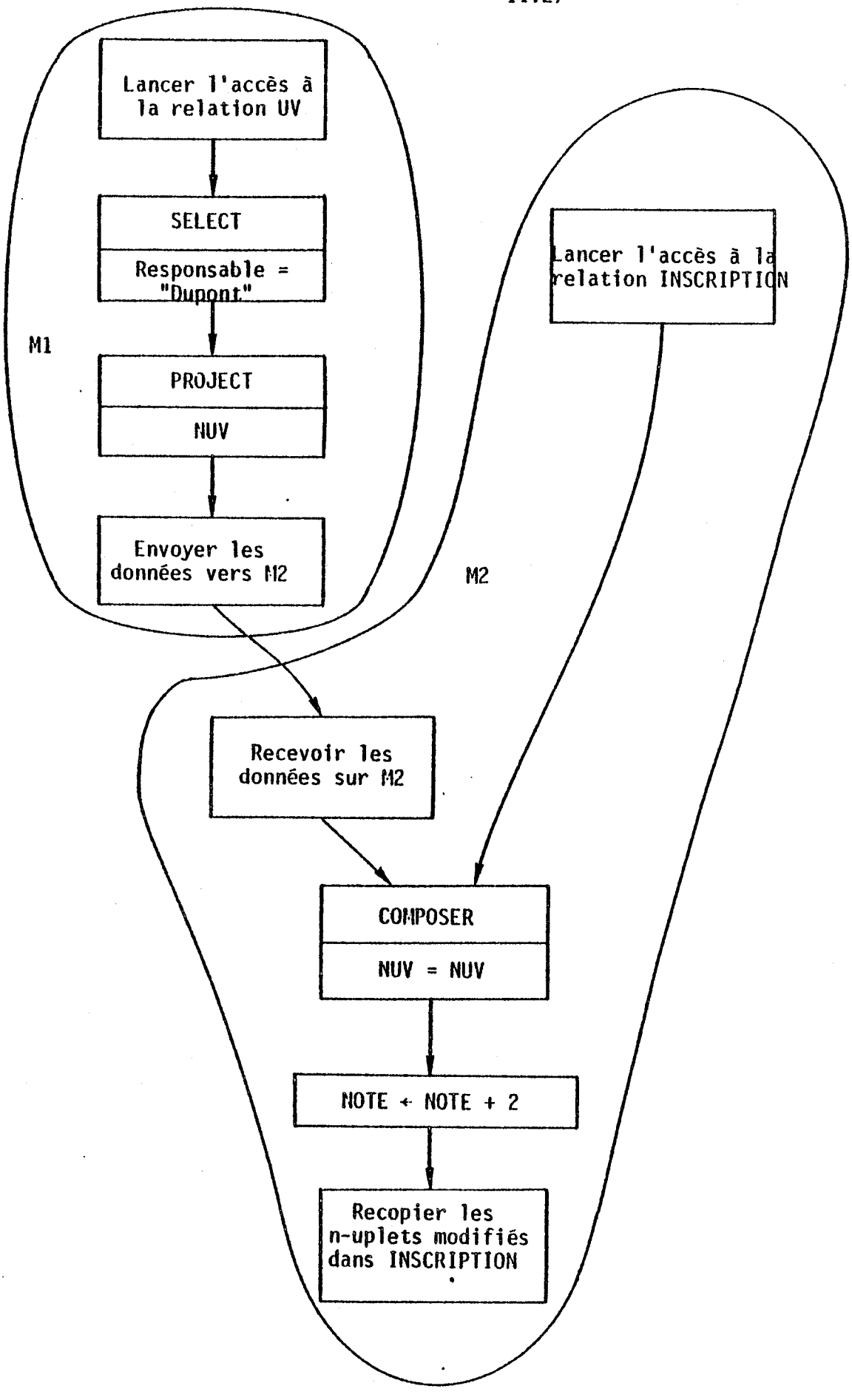


Figure II.13.b - Conséquences du Cas 6 sur l'exécution du graphe de commande localisé

On trouvera ici un tableau récapitulant les centralisations-répartitions dans les différents domaines.

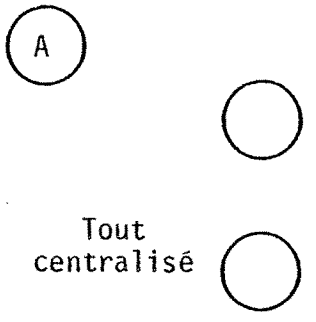
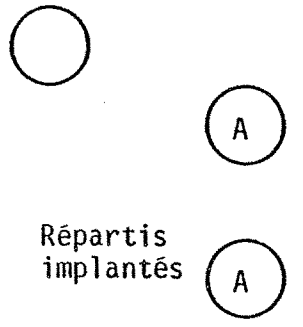
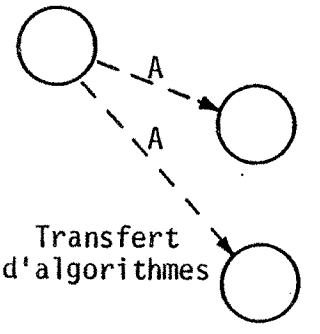
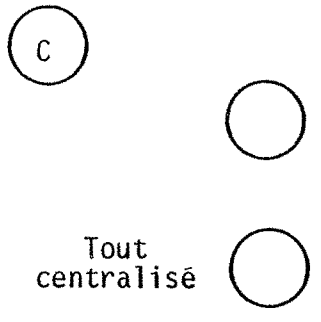
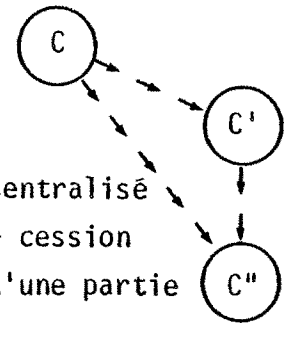
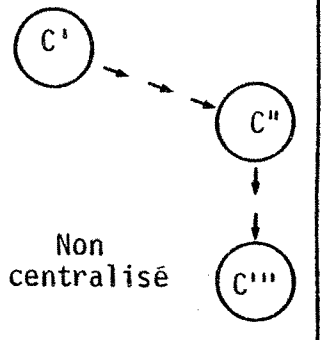
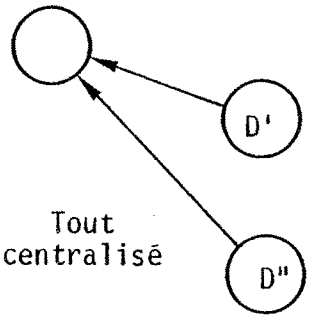
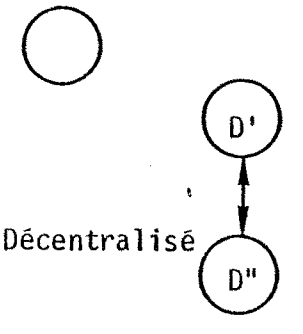
	A1	A2	A3
ALGORITHMES			
	C1	C2	C3
COMMANDES			
	D1	D2	
DONNEES			

Tableau II.3

On remarque aisément que toutes les combinaisons n'ont pas obligatoirement de sens.

Par exemple, A1-D2 ne peut exister puisque les données et les algorithmes doivent obligatoirement être ensemble à l'exécution.

Le diagramme suivant donne l'automate permettant de créer des combinaisons cohérentes.

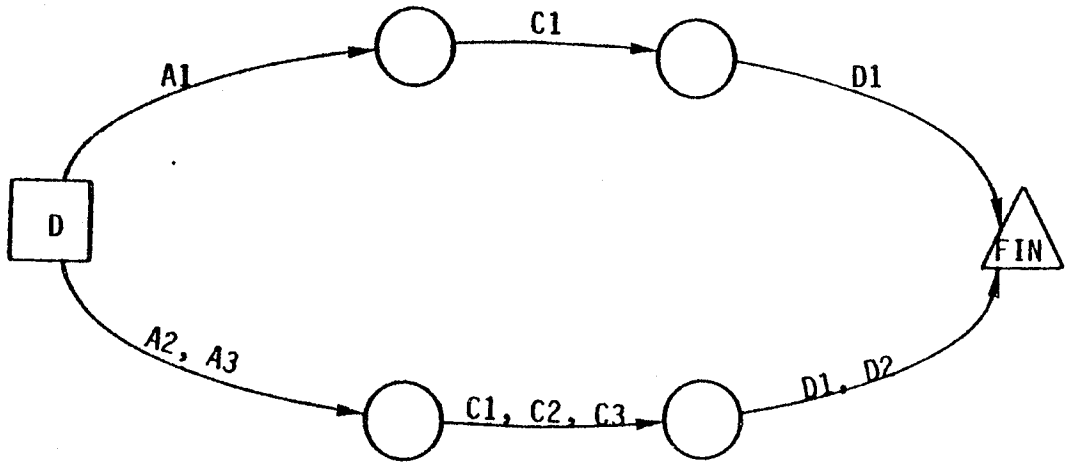


Figure II.14 - Automate donnant les compositions possibles du Tableau II.3

Cela nous donne donc pour les cas précédemment cités :

CAS 1 : A1 C1 D1	CAS 2 : A3 C1 D1
CAS 3 : A3 C1 D2	CAS 4 : A2 C2 D1
CAS 5 : A3 C2 D2	CAS 6 : A2 C3 D2

Le Projet Polyphème correspond à A2 C3 D2, c'est-à-dire : les algorithmes étant déjà implantés (programmes locaux et interpréteurs), les graphes de commande véhiculent les commandes avec transmission des données directe d'une machine à l'autre.

Chapitre III

Exécution d'une Requête globale dans Polyphème

Nous avons énuméré au chapitre précédent les différents critères entrant en ligne de compte pour le choix d'une stratégie d'exécution dans un SGBDR ; ce chapitre présente les motivations en faveur des Réseaux de Petri Interprétés [MOSISI], comme outil algorithmique réparti choisi dans Polyphème.

Tout d'abord, nous énumérons les qualités requises d'un tel outil algorithmique ; ensuite, nous présentons diverses réalisations de cet outil en détaillant les Réseaux de Petri Interprétés (RDPI) et en expliquant les raisons de notre choix moyennant certaines modifications.

III-1 Qualités requises pour un outil algorithmique réparti

Ces qualités peuvent être présentées en trois volets :

- qualités générales ;
- avantages pour l'Utilisateur de la BDR ;
- avantages pour l'Administrateur de la BDR.

a) Qualités générales :

* simple
  à lire et à utiliser : on peut rapidement synthétiser l'algorithme sous-jacent.

* modulaire
  pour pouvoir découper l'algorithme global en sous-algorithmes locaux.

* aisément transportable
  si l'on doit transporter des morceaux d'algorithmes d'une machine à une autre, le protocole de transport doit pouvoir être le plus simple possible.

* sémantiquement dense
  un volume minimum pour un maximum d'information sur l'algorithme.

* extensible
  tout d'abord, pour pouvoir rajouter facilement des opérateurs d'un nouveau type ;
  ensuite pour enrichir un algorithme préexistant soit en ajoutant de nouvelles opérations, soit en substituant à une opération élémentaire un sous-algorithme complet.

* adapté à l'expression du contrôle et du parallélisme
  afin d'utiliser effectivement simultanément les différents processeurs du réseau tout en synchronisant les diverses exécutions.

* permettre une pré-analyse des blocages
  en donnant la possibilité d'un parcours fictif de l'algorithme avant qu'il ne soit effectivement lancé au niveau réseau afin de détecter les éventuels interblocages pouvant survenir.

b) Avantages pour l'Utilisateur de la BDR :

* transparence
  En effet, l'Utilisateur de la BDR est sensé ignorer tous ces problèmes algorithmiques de contrôle et de synchronisation : il s'exprimera en terme de données à manipuler et absolument pas au niveau de l'algorithme réparti.

c) Avantages pour l'Administrateur de la BDR :

* exprimer les contraintes d'intégrité
  L'Administrateur doit pouvoir écrire, en fonction des différents cas de répartition des données, quelles sont les contraintes d'intégrité qu'il veut voir vérifier.
  Par exemple, lors d'une mise à jour d'objets dupliqués, tous les duplicata doivent effectivement avoir été modifiés. Dans le cas contraire, un algorithme de retour arrière doit pouvoir être mis en route pour restaurer la BDR à son état initial.

* réversibilité à tout état stable de l'exécution
  On vient de voir que l'impossibilité de satisfaire aux contraintes d'intégrité obligeait à un retour arrière pour restaurer l'état de la base avant cette dernière modification ; l'idéal est un outil algorithmique réversible permettant automatiquement l'exécution de l'inverse de ce qui a déjà été réalisé.

* tolérance aux pannes et optimisation
  Si l'on exprime plusieurs moyens d'atteindre la solution demandée, et qu'à l'exécution certaines machines du réseau sont absentes, il est alors possible de trouver une des branches de l'algorithme qui donne la solution avec les machines restantes.

Cette redondance de solutions est aussi utilisable pour optimiser l'exécution de l'algorithme en ajoutant une pondération à chaque sous-algorithme. Cette pondération est évaluée soit dynamiquement en fonction du volume des informations à transporter et des caractéristiques des différentes machines, soit statiquement selon ce dernier critère.

III-2 Outils algorithmiques répartis

Nous présentons ici deux types d'outils selon le critère décrit en II-1 :

- deux sont du type génération-exécution
- le troisième est de type graphique interprétable.

Un tableau récapitulant leurs qualités respectives est donné en Annexe 1.

III-2-1 SIGOR

C'est un "Système d'Interprétation Général Orienté Réseau réalisé à Grenoble [DANSER] permettant de définir une Machine Réseau Logique [CHUPIN] grâce à un langage de type procédural. Ce langage unique sur le réseau, avec un interpréteur sur chaque site, est conçu afin d'être transportable (compact). Il permet de gérer les synchronisations entre machines de deux manières :

- par appel de procédures créant une hiérarchie d'exécution, le processus appelant étant le "Père" du processus "Fils" correspondant à la procédure appelée ;
- par variable de type "Rendez-vous" ; ces variables globales aux fils d'un même père leur permettent de se "retrouver" pour échanger des informations.

Si, malgré sa définition, nous ne plaçons pas ce système avec ceux dits "interprétables", c'est que d'un point de vue SGBDR, la requête de l'utilisateur de la BDR doit être analysée pour que soit *généré* un code dont l'*exécution* ensuite est une interprétation au niveau des différentes machines locales.

III-2-2 PERE - MERE

C'est également un outil basé sur l'appel de procédures distantes. MERE est le "Moniteur d'Exécution REpartie" véhiculant lancements de procédures et paramètres au moyen du "Protocole d'Exécution REpartie" [ANDRE, DEC781 et 2]

Utilisant les principes de synchronisation par activation de programmes du système VCAM-SIRIS8 [CIHOBU], MERE en est une double adaptation :

- d'une part au réseau
- d'autre part aux langages de haut niveau.

Un moniteur MERE se trouve sur chaque site du réseau avec un groupe de procédures appartenant à l'application répartie [CHUPIN] déjà implantée.

Une de ces procédures a un rôle très particulier ; c'est la procédure "INITIALE" qui est le début de la chaîne d'exécution : chaque moniteur lance la procédure INITIALE de son site qui lance éventuellement des procédures distantes avec divers paramètres. Pour répondre, ces procédures à distance activeront alors d'autres procédures paramétrées et ainsi de suite (figure III.2). Tous ces échanges entre MERE se font au travers de PERE qui correspond à l'ensemble des règles d'utilisation du réseau pour ces moniteurs.

Il est bien entendu possible de transmettre des noms de procédures en paramètres.

Chaque lancement distant est assorti d'une procédure d'acquittement sur le site de lancement ; cette procédure sera activée lorsque le moniteur du site distant aura reçu la "fin d'exécution" de la procédure désirée.

Le programmeur de l'exécution répartie peut et doit gérer toutes les synchronisations nécessaires. Un outil supplémentaire est disponible pour ces synchronisations ; c'est l'*Interaction* [DEC781, GALY].

L'Interaction permet de contrôler le nombre d'appels de procédures (ou *Actions*) entre deux sites au même instant.

Ce contrôle est mis en oeuvre par les MERE correspondants qui limitent les appels sur le site de lancement s'ils ne peuvent être pris en compte sur l'autre site ; c'est donc un double système d'asservissement du type de la figure III.1.

Une interaction est schématiquement composée des éléments suivants :

- deux *files d'attente* contenant les Actions à lancer (une par site) ;
- deux *compteurs d'anticipation* donnant la place restante dans la file de l'autre site (si le compteur est nul, il est inutile d'envoyer une activation de procédure qui ne pourra être mise en file d'attente ; la procédure ayant requis cette activation est alors mise en attente, le contrôle étant donné à une autre interaction de ce site) ;
- deux *variables d'état* correspondant aux deux directions de l'interaction ($S_i \rightarrow S_j$ et $S_j \rightarrow S_i$) et pouvant prendre les valeurs suivantes :
  - A . active (une action est en train de s'exécuter)
  - S . suspendue (demande d'action en attente)
  - I . inexistante (avant l'*ouverture* ou initialisation de l'interaction ou après sa fermeture) ;
- une procédure en activité, au maximum, par site.

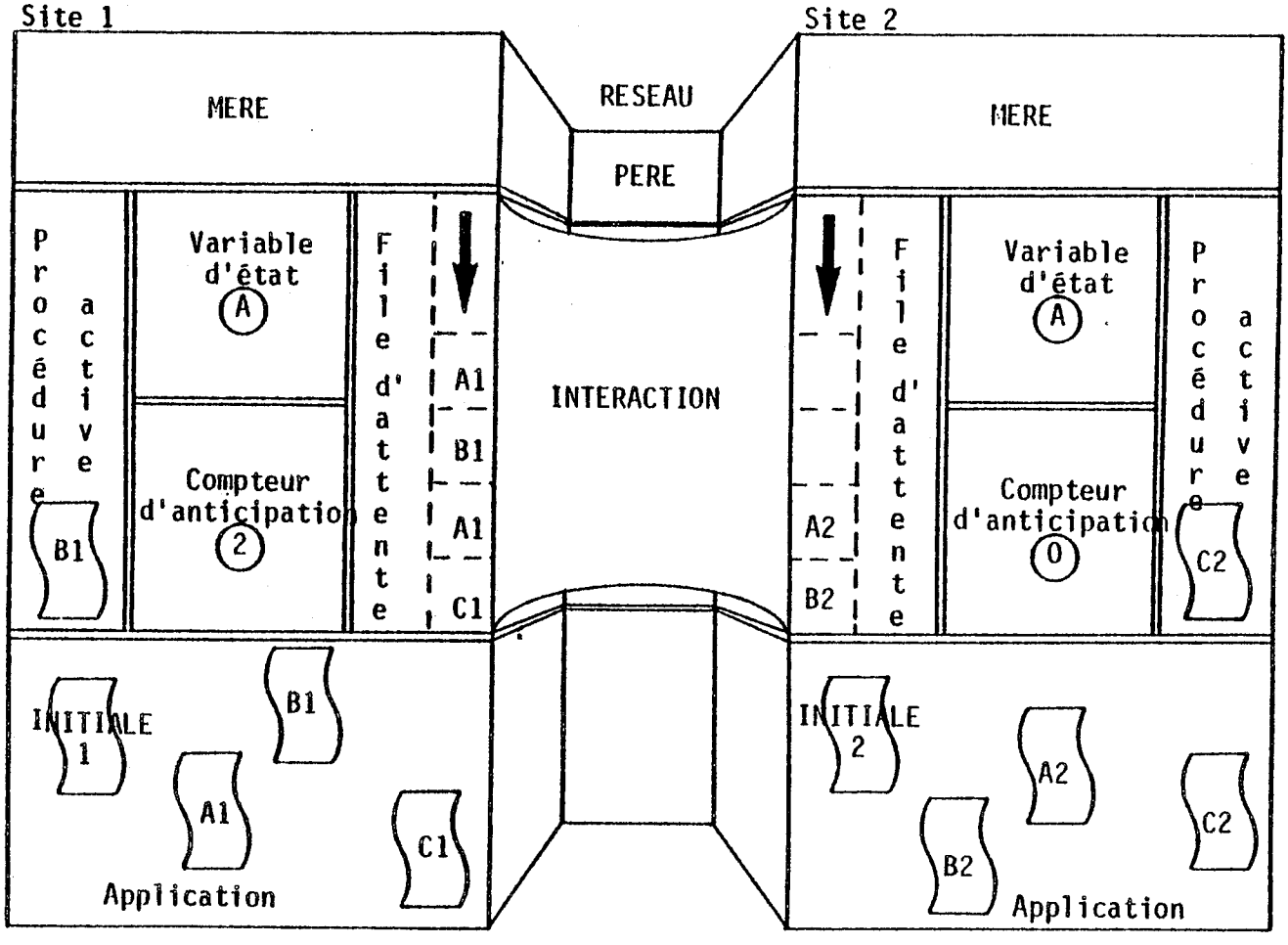


Figure III.1 - Exemple d'une Interaction

La Figure III.1 présente une interaction active I dans les deux sens. Si l'on suppose que la procédure "C2" en activité sur le Site 2 requiert un lancement sur I, d'une procédure du Site 1, cette activation de C2 sera suspendue et la variable d'état de I sur le site 2 passera de "Active" à "Suspendue".

Nous n'avons représenté sur cette figure qu'une seule interaction entre ces deux sites ; il peut en fait en exister plusieurs.

Une règle importante régissant les actions d'une interaction est que celles-ci sont exécutées séquentiellement, dans l'ordre des lancements.

D'autre part, deux actions appartenant à des *interactions différentes* peuvent s'exécuter de manière concurrente. Les pertes de contrôle d'une action ne peuvent se faire que lorsque celle-ci fait appel à MERE, c'est-à-dire lorsqu'elle veut lancer une procédure à distance. L'action est alors *suspendue*.

Une action est *finie* lorsque l'exécution correspondante a atteint le retour de procédure.

SITE 1	SITE 2
INITIALE OITR (S2,ACKITR,I) /* Demande d'ouverture vers S2 de l'Interaction I */ Fin	INITIALE /* Rien à faire dans l'attente d'un appel ⇒ fin pour donner le contrôle au Moniteur */ Fin
ACKITR (Code-retour,I) Si Code-retour = "OK" alors lancer (I,LECTFICH,Nomfich, REPONSE,FINFICH,I) sinon Appel Fin-monit /* Fin d'exécution si l'on ne peut pas ouvrir l'interaction (panne...) */	/* Lorsque MERE a réalisé l'ouverture de I en mettant à jour ses tables sur les deux sites, il donne le contrôle à ACKITR sur S1 */
REPONSE (Buffer, I) Ecrire (Buffer) /* Ecrit l'enregistrement reçu sur l'imprimante du site 1 */ Fin	LECTFICH (Nomfich,REPONSE,FINFICH,I) Tantque ¬ (Fin de fichier) Lire (Nomfich, Buffer) Lancer (REPONSE, Buffer, I) Fin Lancer (FINFICH, I) Fin
FINFICH (I) CITR (I) /* Fermeture de I */ Appel Fin-monit Fin	

Figure III.2 - Exemple d'utilisation du Moniteur d'Exécution Repartie : Impression d'un fichier du Site 2 sur le Site 1

Dans sa première version, PERE ne permet pas le transport et le catalogage d'algorithmes. Il est toutefois aisé de concevoir un type particulier de paramètre à transmettre via PERE à une procédure spéciale de catalogage : le type instruction [ANDRE].

Les catalogues de procédures des diverses machines deviennent alors dynamiques, avec la gestion répartie supplémentaire que cela suppose [NEUBIL].

En fait, tout du moins dans la version actuelle, il n'est pas question de générer un code utilisable par PERE et MERE à partir de la requête de l'utilisateur de la BDR puisqu'il est impossible de transporter le code ainsi généré.

Par contre, on peut transmettre sous forme de paramètres de procédures des éléments d'un graphe représentant l'algorithme à exécuter sur chaque machine, ces graphes étant créés à partir de la requête de l'Utilisateur de la BDR. Les procédures contrôlées par MERE jouent alors le rôle d'un Interpréteur de graphe sur chaque site.

C'est le type de graphe nécessaire que nous allons étudier dans le paragraphe suivant.

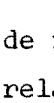
III-2-3 Les Réseaux de Petri Interprétés

III-2-3-1 Les Réseaux de Petri

Il existe un grand nombre de Réseaux de Petri ; notre propos ici n'est évidemment pas de les décrire en détail ; d'autres l'ont très bien fait avant [MOPUSI, TSIBER, PETERS].

Nous voulons simplement en rappeler le principe en précisant qu'ils peuvent être facilement utilisés pour modéliser les calculs parallèles, donc a fortiori le cas qui nous préoccupe.

Nous allons donc les décrire au travers d'un exemple. Un Réseau de Petri peut être représenté sous forme matricielle (matrice d'incidence [MOPUSI]) ou sous forme graphique ; c'est cette dernière présentation que nous allons utiliser [HOLCOM].

Un Réseau de Petri est composé de deux types de noeuds appelés *Places* () et *Transitions* () liés entre eux par deux relations :

- la *relation d'incidence avant* qui lie une place à une transition ;
- la *relation d'incidence arrière* qui lie une transition à une place.

Ces deux relations sont matérialisées par des *Arcs* dans la représentation graphique ; on remarque qu'un Arc ne peut en aucun cas lier deux places ou deux transitions directement.

Ceci nous permet de dessiner un Réseau de Petri quelconque à titre d'exemple (Figure III.3).

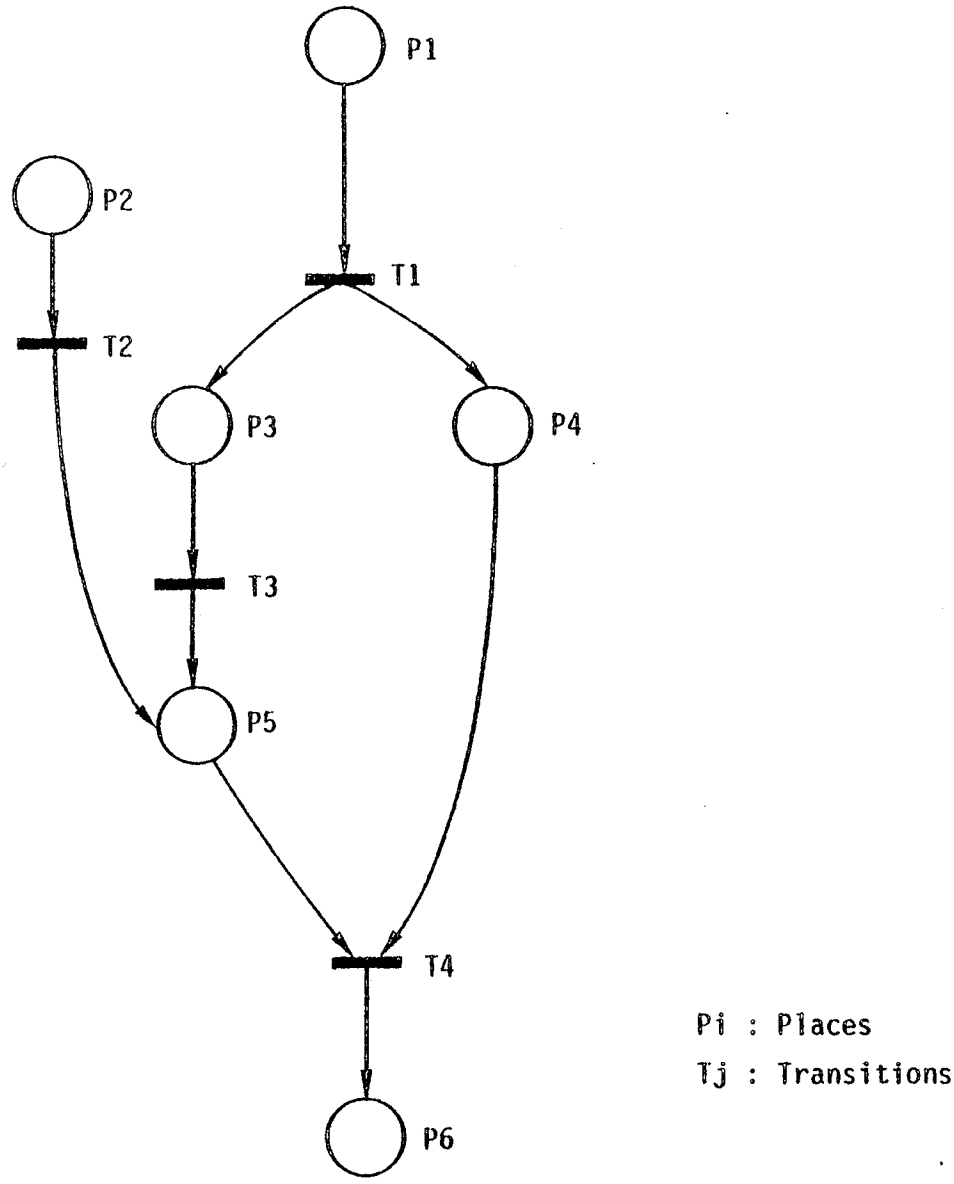


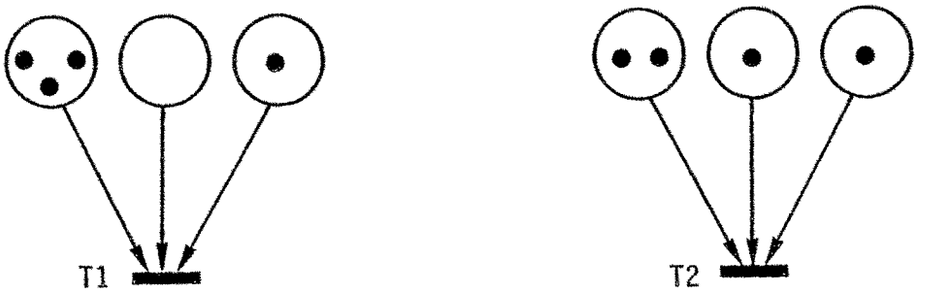
Figure III.3 - Exemple de Réseau de Petri

Nous donnerons une seule définition de Réseau de Petri particulier, le *Graphe Marqué* : dans un Graphe Marqué [MOPUSI, COMMCO] toute place a au plus une transition d'entrée et une transition de sortie.

Le réseau de la Figure III.3 n'est pas un graphe marqué à cause de la phase "P5". En enlevant la partie "P2-T2" et les arcs correspondants, il nous reste un graphe marqué.

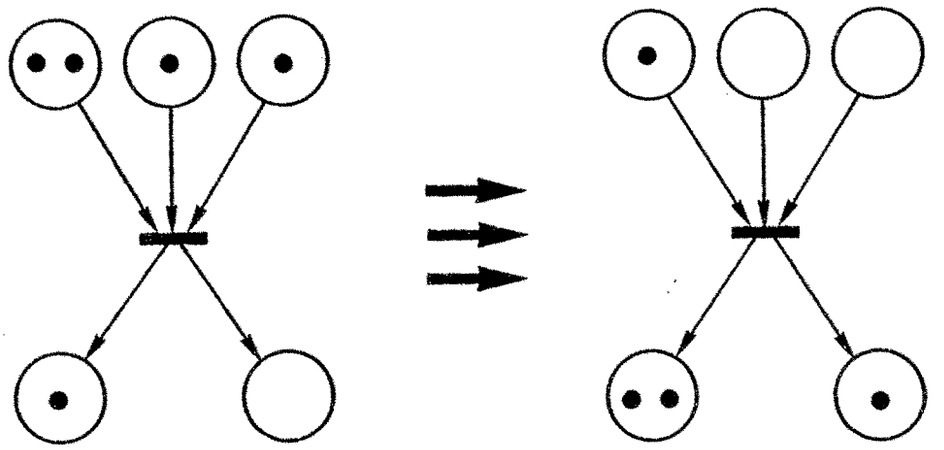
Le principe dynamique des Réseaux de Petri consiste en une évolution de *Jetons* ou *Marques* dans le graphe qui vont de place en place au travers des transitions selon les règles suivantes :

* Une transition "T" est *tirable* si et seulement si toute place d'entré de "T" contient au moins une marque.



Non tirable — Tirable

* La *mise à feu* d'une transition consiste au retrait d'une marque par place d'entrée et au dépôt d'une marque dans chaque place de sortie.



Avant mise à feu — Après mise à feu

La position des marques avant toute mise à feu (ou *marquage initial*) conditionne évidemment tout le cheminement dans le réseau.

Dans l'exemple de la figure III.3 le marquage initial permettant de traverser toutes les places sera {P1,P2} c'est-à-dire une marque dans P1 et une dans P2.

D'un point de vue général, les Réseaux de Petri offrent de multiples variantes ; ils ont toutefois le défaut d'être pauvres en propriétés et trop riches en possibilités.

Le problème est d'en définir une sous-classe suffisante pour modéliser des exécutions réparties et donc suffisamment limitée pour posséder des propriétés intéressantes tout en conservant une bonne clarté d'interprétation.

Nous avons choisi les Réseaux de Petri Interprétés (RDPI) que nous définissons au paragraphe suivant.

III-2-3-2 Définition des RDPI

Ces définitions sont reprises de [MOSISI]. On va considérer un automate composé de deux parties :

- une partie opérative
- une partie commande.

La partie commande donnera des ordres à exécuter à la partie opérative qui rendra des comptes-rendus ou événements (Figure III.4).

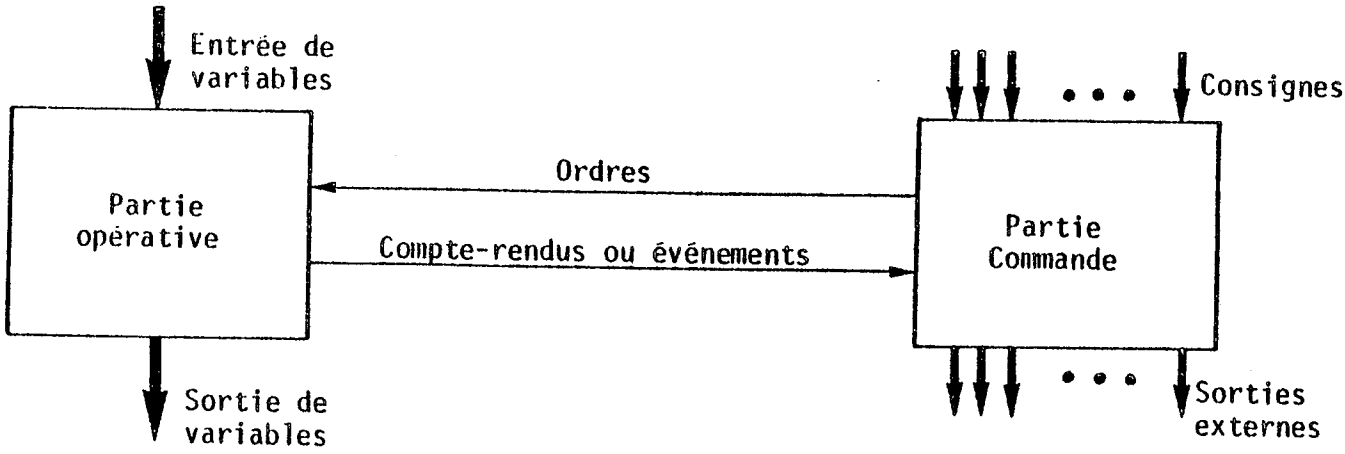


Figure III.4

La partie commande va travailler sur un Réseau de Petri Temporisé-Synchronisé (RDPTS) composé de [MOSISI] :

- un Réseau de Petri tel qu'il a été défini précédemment ;
- un ensemble E d'*événements externes* ;
- une application μ associant à toute transition un événement de $E \cup \{e\}$ où "e" est "l'événement toujours présent" ;
- une suite croissante de réels $\theta = \{\tau_i\}_{i \in IN}$, appelée *base de temps* telle que à toute occurrence d'événements de E correspond un élément de θ ;

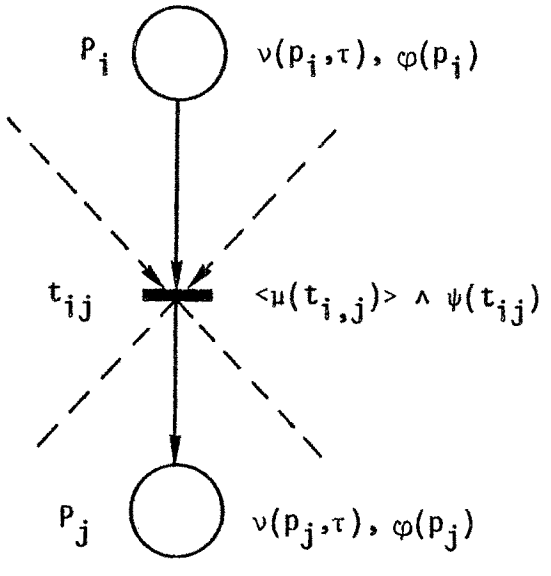
- une application ν de $P \times \theta$ dans θ (avec P correspondant à l'ensemble des places du réseau) telle que $\forall (p,\tau_i) \in P \times \theta$, $\nu(p,\tau_i) \geq \tau_i$, c'est-à-dire :
  la place "p" aura nécessité un temps $\nu(p,\tau_i) - \tau_i$ pour être franchie.

Pour des précisions supplémentaires, consulter [MOSISI].

A ce RDPTS, il faut ajouter trois éléments pour obtenir un RDPI :

- un sous-système opératif (X, OP, C) :
  $X = \{x_1, x_2, \ldots, x_u\}$, ensemble fini de variables prenant leurs valeurs dans les domaines $D_1, D_2, \ldots, D_u$;
  $OP = \{op_1, \ldots, op_v\}$, opérateurs définis comme applications internes de $D_1 \times D_2 \times \ldots \times D_u$;
  $C = \{c_1, c_2, \ldots, c_n\}$, ensemble de conditions (prédicats) sur les variables de X.
- une application $\varphi : P \rightarrow OP$ qui à toute place (P_i) du RDPTS fait correspondre une action $(\varphi(P_i))$.
- une application $\psi : T \rightarrow C$ qui associe une condition $(\psi(t_j))$ à chaque transition (t_j).

Un RDPI sera utilisé comme un RDPTS avec en plus des interactions avec la partie opérative ; ces interactions sont résumées dans le schéma suivant:



- t_{ij} n'est considérée comme réceptive à $\mu(t_{ij})$ que si elle est *validée* et le *prédicat* $\psi(t_{ij})$ *vrai*.

- L'arrivée d'une marque en p_i à l'instant τ active l'opérateur $\varphi(p_i)$; les transformations effectuées par cet opérateur prennent place pendant l'intervalle $]\tau, \nu(p_i,\tau)[$ pendant lequel la marque est mise à l'état indisponible.

En fait, dans l'utilisation que nous ferons par la suite de ces RDPI, nous ne ferons pas intervenir le temps dont il est très difficile de générer un contrôle cohérent à partir d'un graphe de données du type de celui utilisé dans Polyphème.

Nous ne considérerons donc jamais les délais d'exécution :

$$\forall p, \tau, \nu(p,\tau) = \tau$$

ce qui revient à dire que le temps d'exécution des opérateurs est nul.

En reprenant l'exemple du premier chapitre, nous allons donner son application possible aux RDPI en indiquant deux granularités d'exécution (II-2) :

- une opération par relation,
- une opération par n-uplet.

Pour chaque granularité, nous présentons tout d'abord le graphe en Réseau de Petri (Figures III.6 et III.7), puis les applications ϕ et ψ correspondant aux RDPI et définies précédemment.

Nous avons utilisé les conventions suivantes :

- les places : P_i $\quad i \in \mathbb{N}$;
- les transitions : t_j $\quad j \in \mathbb{N}$;
- les relations : R_m $\quad m \in \mathbb{N}$;
- les n-uplets : N_n $\quad n \in \mathbb{N}$, $N_n \in R_n$
- les arcs : ——→ contrôle des R_i ou N_j

  - - - → répercussion de "fin de relation" (Fig. III.7).

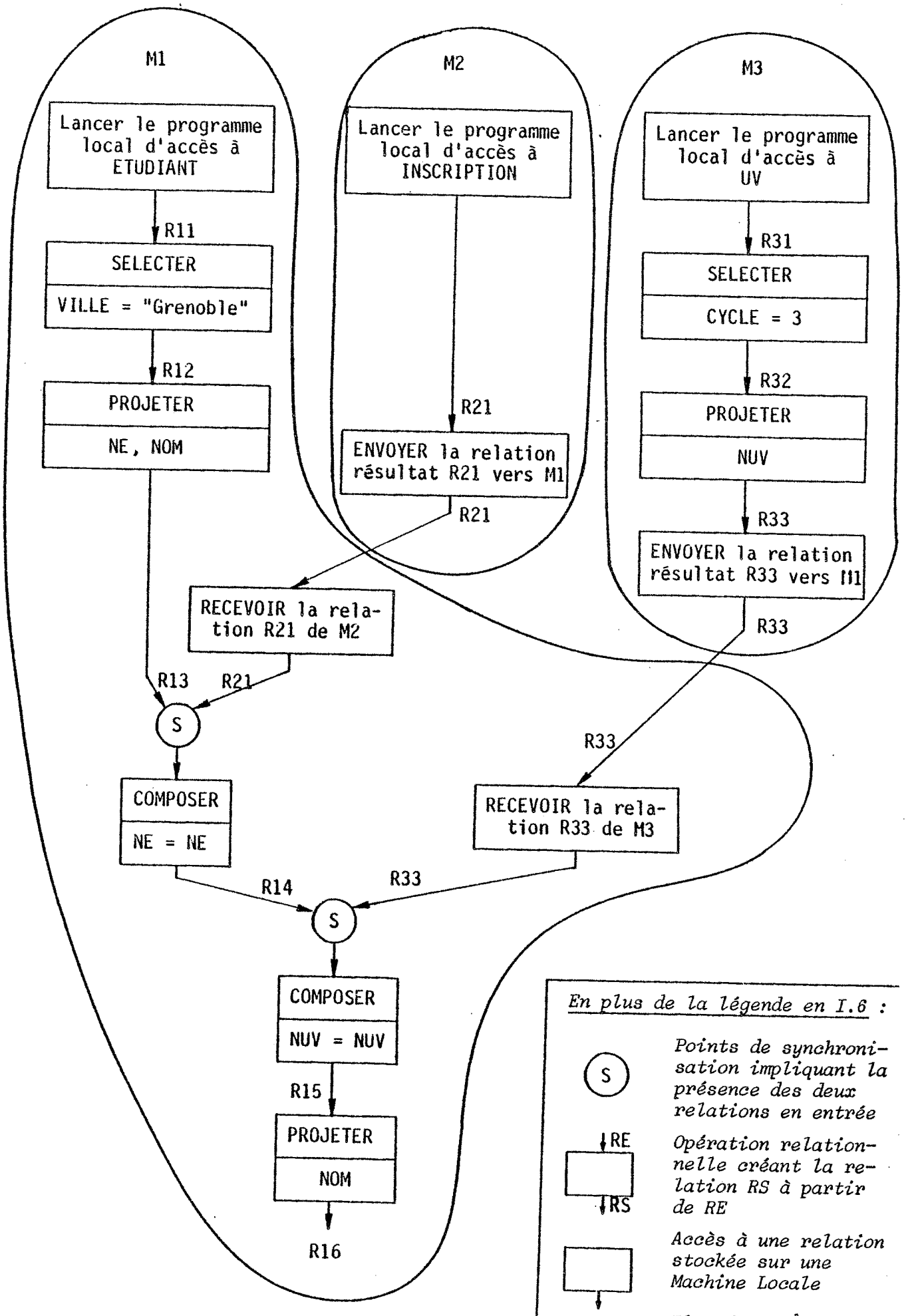


Figure III.5 - Rappel de la figure I.7

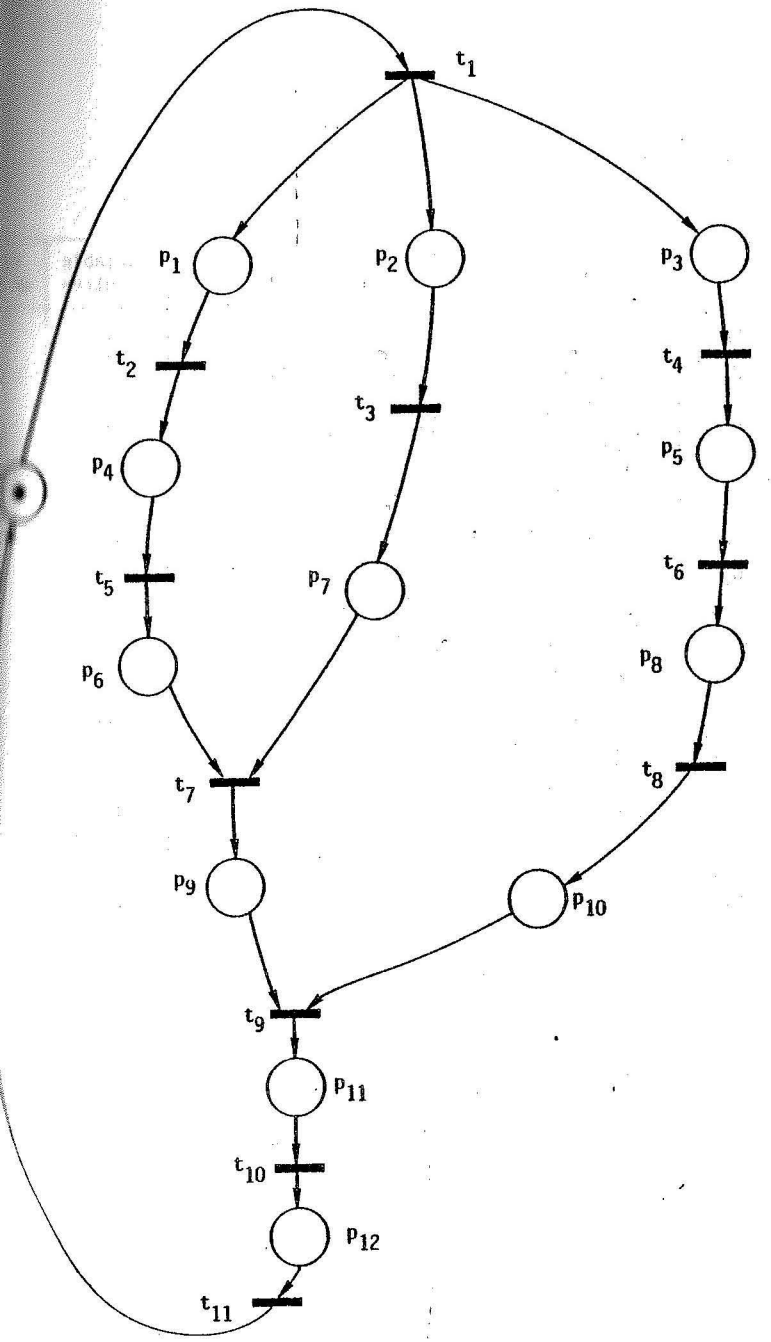


Figure III.6 - RDPI correspondant à la figure III.5 avec une opération par relation

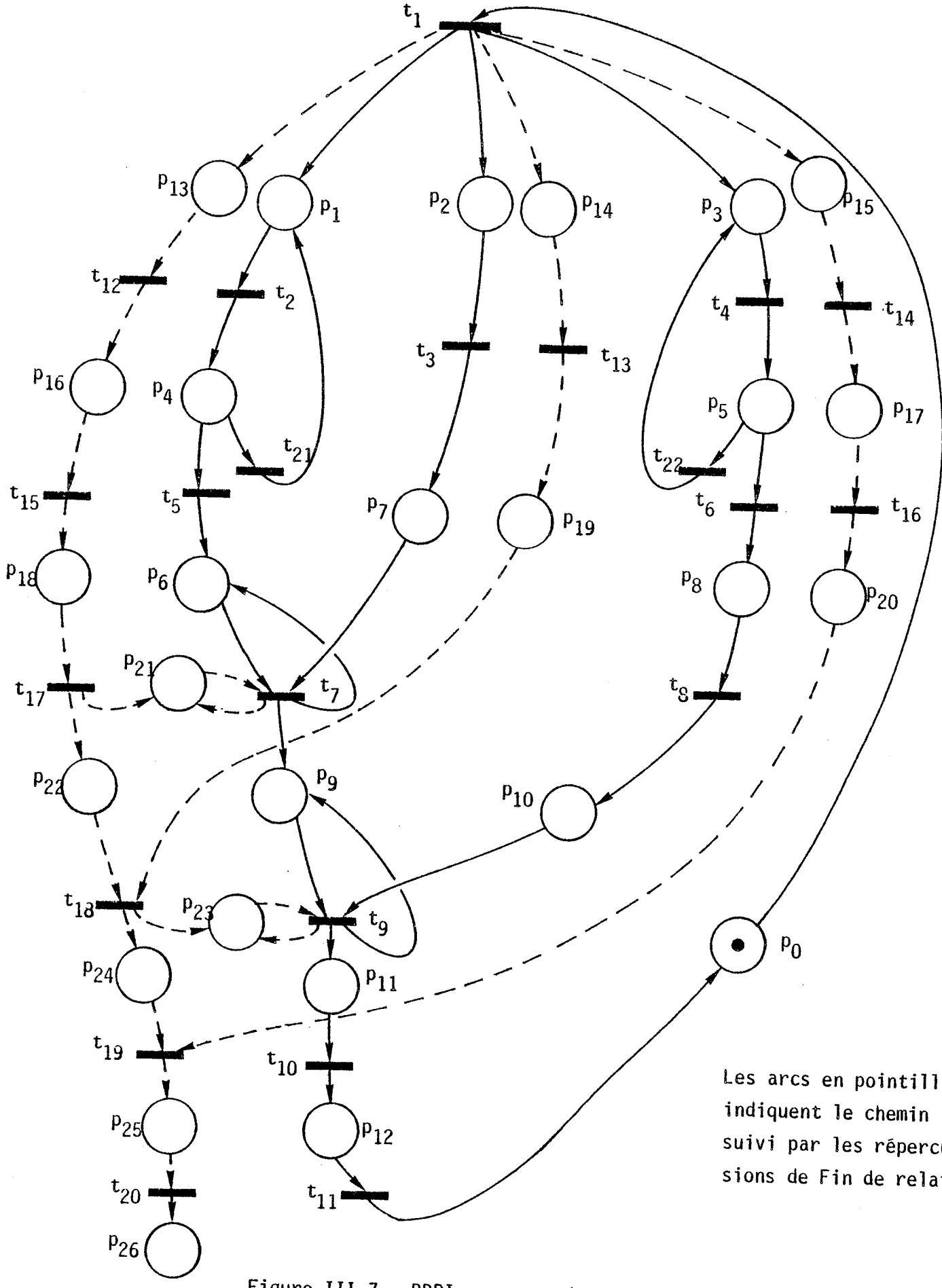
Application ϕ correspondant à l'exemple Figure III.6

Places	Opération	Variable d'entrée	Variable de sortie
P_0	ϕ		
P_1	Lancer le programme d'accès à ETUDIANT sur M1		R11
P_2	Lancer le programme d'accès à INSCRIPTION sur M1		R21
P_3	Lancer le programme d'accès à UV sur M3		R31
P_4	Sélecter VILLE = "Grenoble"	R11	R12
P_5	Sélecter CYCLE = 3	R31	R32
P_6	Projeter NE, NOM	R12	R13
P_7	Transport de M2 vers M1	R21	R21
P_8	Projeter NUV	R32	R33
P_9	Composer NE = NE	R13,R21	R14
P_{10}	Transport de M3 vers M1	R33	R33
P_{11}	Composer NUV = NUV	R14,R33	R15
P_{12}	Projeter NOM	R15	R16

Application ψ correspondant à l'exemple Figure III.6

Transition	Condition
t_1	<e> (toujours vraie)
t_2	<fin d'accès à ETUDIANT>
t_3	<fin d'accès à INSCRIPTION>
t_4	<fin d'accès à UV>
t_5	<fin du SELECT>
t_6	<fin du SELECT>
t_7	<fin du transport de M2 vers M1> ∧ <fin du Project>
t_8	<fin du Project>
t_9	<fin du transport de M3 vers M1> ∧ <fin du Composer>
t_{10}	<fin du Composer>
t_{11}	<fin du Project>

Pour le même exemple de la Figure III.5, le RDPI correspondant à une exécution n-uplet par n-uplet sera le suivant :

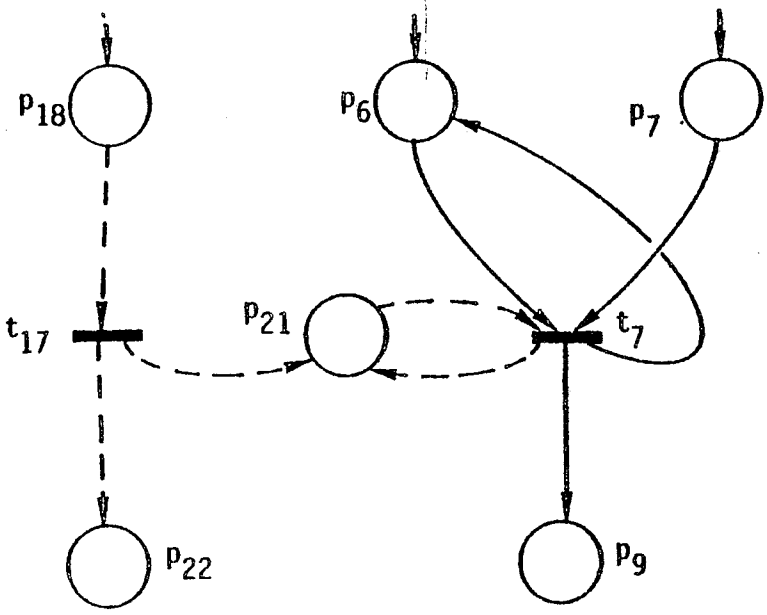


Les arcs en pointillés indiquent le chemin suivi par les répercussions de Fin de relation

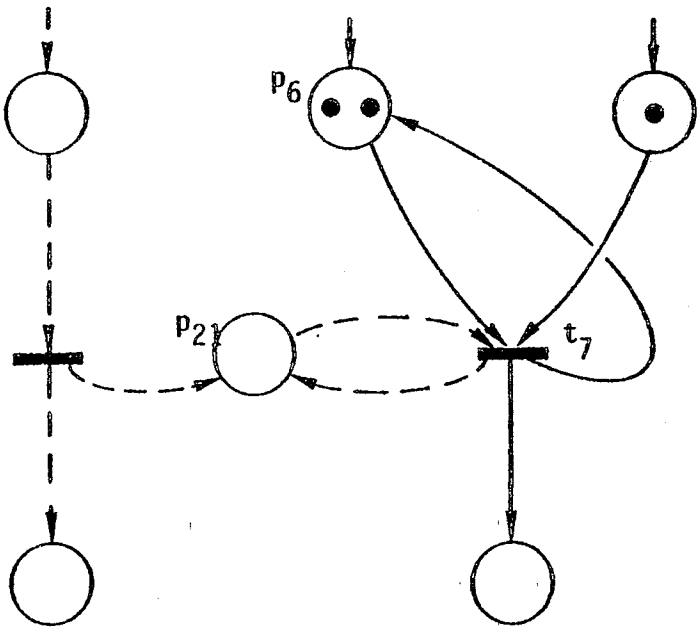
Figure III.7 - RDPI correspondant à la figure III.5 avec une opération par n-uplet

(voir applications ϕ et Ψ en page III.22 et III.23)

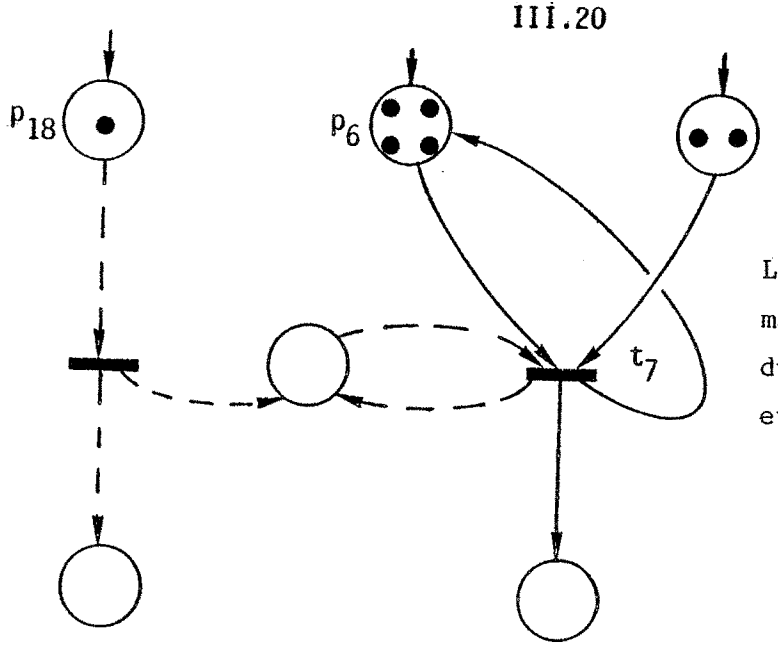
On peut remarquer le principe employé pour exprimer l'attente de la "fin d'une relation" pour réaliser la composition en mode mixte (II-2-3).



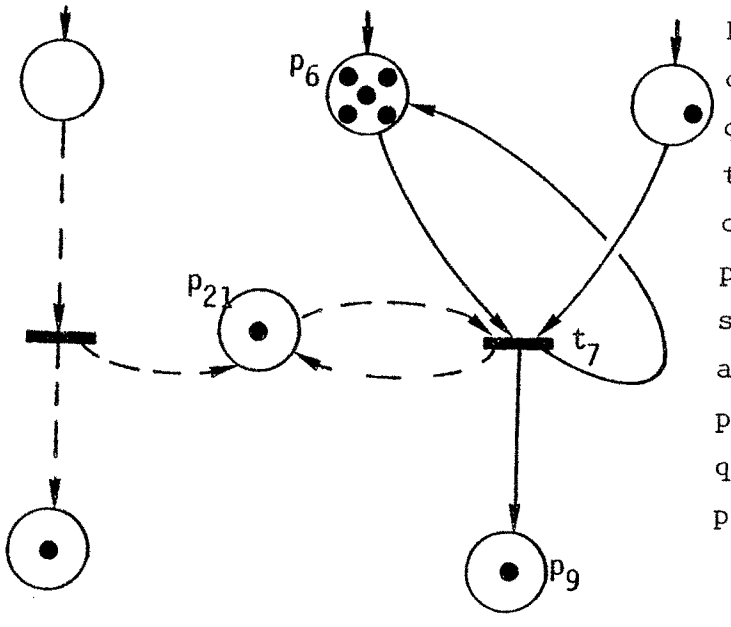
Les schémas suivants donnent un historique de l'exécution de cette partie du graphe dans l'hypothèse où la relation R12 de sortie du "Select (VILLE="Grenoble")" a un cardinal de 5.



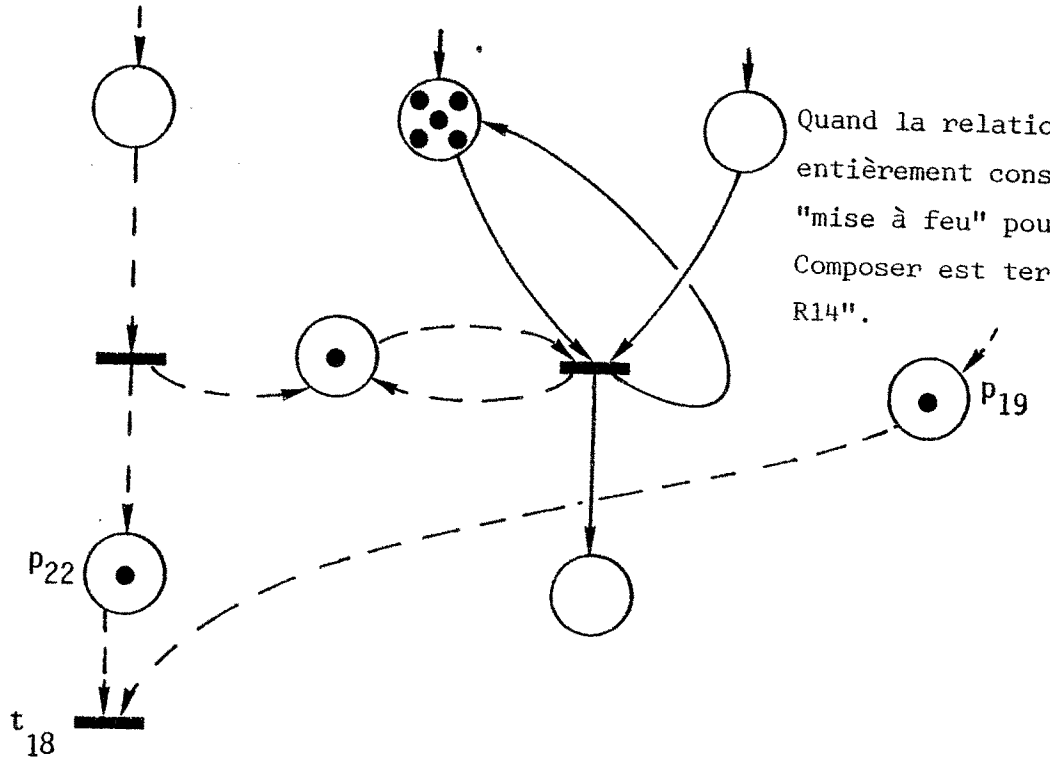
Deux n-uplets "N13" sont disponibles ainsi qu'un n-uplet "N21". Comme il n'y a pas de marque dans P21, la transition t_7 n'est pas tirable.



Le SELECT étant terminé, la marque dans P18 attend la fin du PROJECT alors que quatre "N13" et deux "N21" sont disponibles.



Le PROJECT fini, il est possible de "consommer" des "N21". C'est là qu'intervient l'arc de retour de t_7-p_6 qui garantit la consommation complète de R21 de manière indépendante du cardinal de R13 (on suppose que le Project "p_6" ayant annoncé "fin de relation", il n'est plus réactivé par de nouvelles marques ; sinon il faut rajouter une place fictive).



Quand la relation R21 aura été entièrement consommée, t_{18} sera "mise à feu" pour indiquer que le Composer est terminé : "Fin de R14".

A propos de la figure III.7, il est possible de faire la remarque suivante :

les marques indiquant la disponibilité de n-uplets peuvent être absorbées par paquet au niveau d'une place. Cela pourrait signifier qu'un opérateur travaille sur plusieurs n-uplets simultanément.

En fait, par souci de simplification du schéma, nous n'avons pas fait figurer les places qui garantissent l'unicité de l'élément à traiter par un opérateur. Ce sera à l'interpréteur du RDPI d'en tenir compte.
Dans le cas contraire, on aurait eu pour chaque opérateur deux places supplémentaires sans opération en correspondance (Figure III.8).

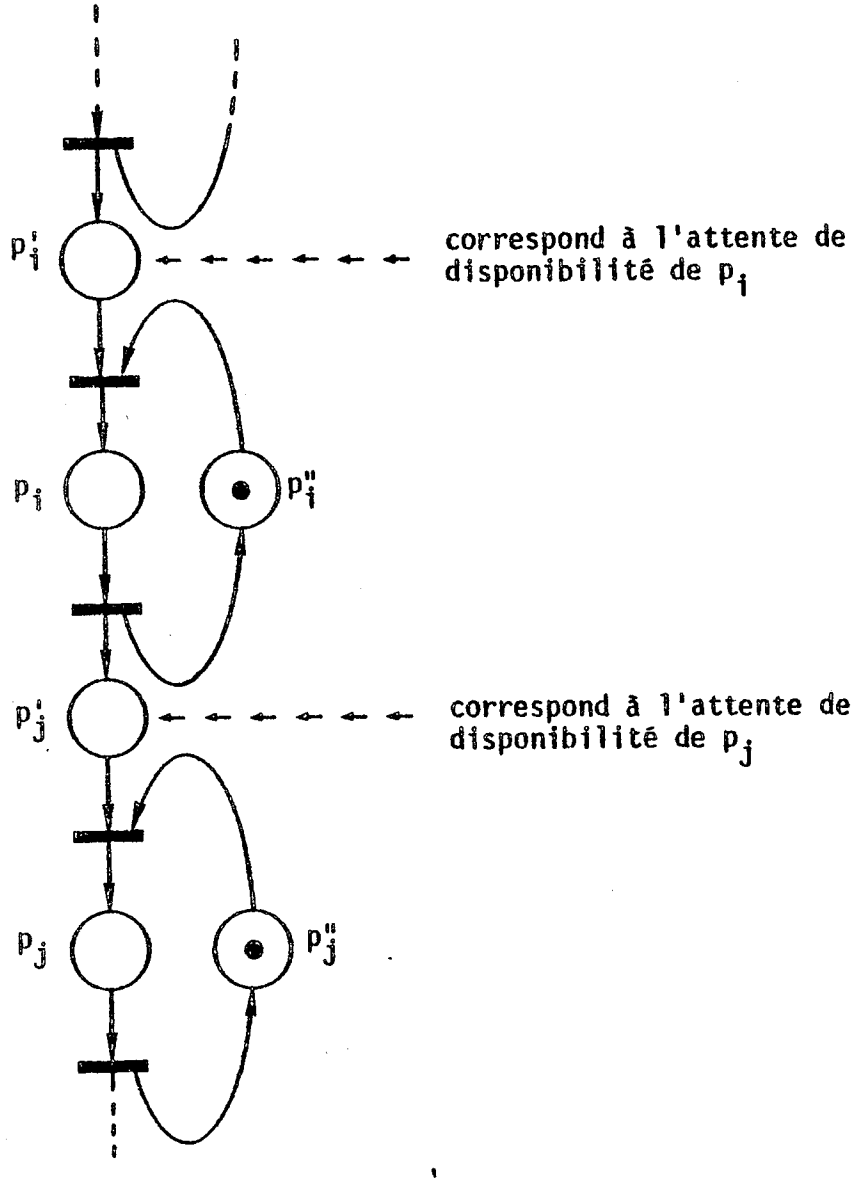


Figure III.8 - Rajout de places sans opérateur pour garantir la non-préemptibilité des opérateurs

Application ϕ correspondant à l'exemple Figure III.7 avec "Ni" désignant un n-uplet de Ri

Places	Opération	Variables d'entrée	Variables de sortie
P_0	ϕ		
P_1	Demander 1 n-uplet de ETUDIANT sur M1		N11
P_2	Demander 1 n-uplet de INSCRIPTION sur M2		N21
P_3	Demander 1 n-uplet de UV sur M3		N31
P_4	Sélecter VILLE = "Grenoble"	N11	N12
P_5	Sélecter CYCLE = 3	N31	N32
P_6	Projeter NE, NOM	N12	N13
P_7	Transport de M2 vers M1	N21	N21
P_8	Projeter NUV	N32	N33
P_9	Composer NE = NE	N13, N21	N14
P_{10}	Transport de M3 vers M1	N33	N33
P_{11}	Composer NUV = NUV	N14, N33	N15
P_{12}	Projeter NOM, stocker dans R16	N15	N16
P_{13}	Attente de FIN - R11		
P_{14}	Attente de FIN - R21		
P_{15}	Attente de FIN - R31		
P_{16}	Attente de FIN - R12		
P_{17}	Attente de FIN - R32		
P_{18}	Attente de FIN - R13		
P_{19}	Attente de FIN du transport R21		
P_{20}	Attente de FIN - R33		
P_{21}	Assure la présence de R13 complète		
P_{22}	Attente de FIN - R14		
P_{23}	Assure la présence de R33 complète		
P_{24}	Attente de FIN - R15		
P_{25}	Attente de FIN - R16		
P_{26}	Annoncer "fin de transaction"		R16

Application ψ correspondant à l'exemple Figure III.7

Transition	Condition
t_1	<e> (toujours vraie)
t_2	<N11 disponible>
t_3	<N21 disponible>
t_4	<N31 disponible>
t_5	<N12 disponible car VILLE (N11) = "Grenoble">
t_6	<N32 disponible car CYCLE (N31) = 3>
t_7	<R13 disponible> ∧ <N21 disponible>
t_8	<N33 disponible>
t_9	<R14 disponible> ∧ <N33 disponible>
t_{10}	<N15 disponible>
t_{11}	<N16 disponible>
t_{12}	<Fin R11>
t_{13}	<Fin R21>
t_{14}	<Fin R31>
t_{15}	<Fin R12>
t_{16}	<Fin R32>
t_{17}	<Fin R13>
t_{18}	<Fin R14> ∧ <Fin du transport de R21>
t_{19}	<Fin R15> ∧ <Fin du transport de R33>
t_{20}	<Fin R16>
t_{21}	<Le n-uplet N11 n'a pas "Grenoble" comme VILLE>
t_{22}	<Le n-uplet N31 n'a pas "3" comme CYCLE>

III-2-3-3 Les RDPI dans un SGBDR

La première qualité que nous avons énoncée pour un outil algorithmique réparti (III-1), c'est la simplicité .... Or, la figure III.7, pour peu qu'on y rajoute des places comme en III.8 ne garantit pas une lisibilité particulière ! C'est qu'en fait les RDPI sont trop généraux pour notre problème.

Nous avons donc proposé quelques restrictions permettant d'adapter les RDPI aux SGBDR. Ces modifications se répercutent évidemment sur le code des interpréteurs. Nous appelons "*Graphe Normalisé avec Emetteurs et Collecteurs*" (GNEC) le type graphique résultant de ces restrictions sur les RDPI.

. Restriction 1 : Tout d'abord nous ne considérerons que des *graphes marqué* (III-2-3-1) dont nous rappelons la définition : "toute place a au plus une transition d'entrée et une transition de sortie" [MOPUSI, COMMCO] (restriction valable pour les GNEC avec une opération par relation).

. Restriction 2 : Les seules transitions acceptées seront :

- soit une place en amont,
  une ou plusieurs places en aval ;

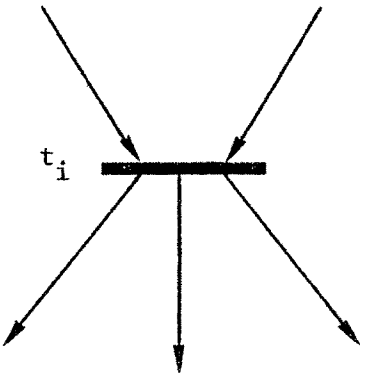
nous parlerons alors d'un "*Emetteur*"

- soit une ou plusieurs places en amont,
  une seule place en aval,

que nous appellerons "*Collecteur*".

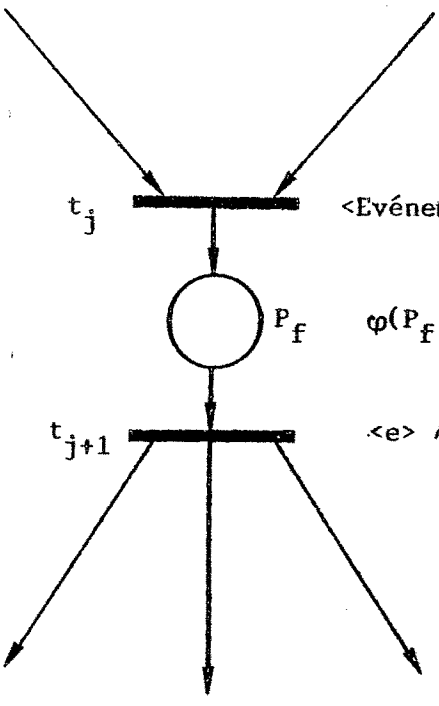
Il est facile de voir qu'au prix d'une place supplémentaire sans opérateu toute transition peut être divisée en Collecteur suivi d'un Emetteur.

Ainsi, pour la transition t_i suivante :



<Evénement> ∧ Condition

devient :



avec <e> = événement toujours présent

où t_j est un collecteur et t_{j+1} un émetteur.

- Restriction 3 : Dans un SGBDR, le graphe de contrôle étant déduit du graphe de données, toutes les conditions testées portent en fait sur la disponibilité ou non des éléments manipulés dans la BDR.
  A partir de cette constatation, nous avons associé à la présence d'une marque en entrée d'une transition la disponibilité de l'élément produit par l'opérateur correspondant à la place délivrant cette marque.

Pour simplifier les écritures, nous avons défini plusieurs types d'Emetteurs et de Collecteurs ; nous parlerons alors d'Emetteurs ou de Collecteurs de type OU, ET, XOU et NON.

Emetteur ET :

ET 

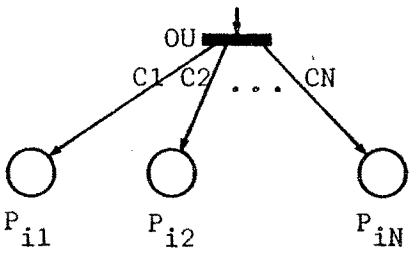
C'est la transition standard qui, lors de sa "mise à feu" génère une marque sur chacun des arcs en sortie.

Collecteur ET :

ET 

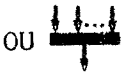
C'est également la conception standard d'une transition : il faut une marque dans chacune des places d'entrée pour qu'ait lieu la "mise à feu".

Emetteur OU :



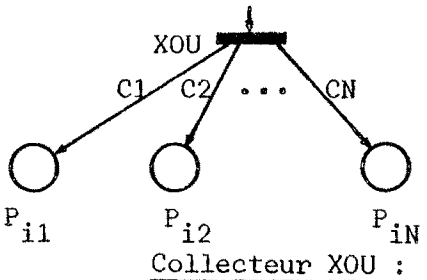
La mise à feu génère une marque sur un ou plusieurs arcs en sortie, en fonction des conditions liées aux arcs ; cela revient à écrire les conditions relatives à l'émetteur E_i sous la forme : (Arc vers P_{i1} $\wedge$ C_1) $\vee$ (Arc vers P_{i2} $\wedge$ C_2) $\vee$ ... $\vee$ (Arc vers P_{iN} $\wedge$ C_N).

Collecteur OU :



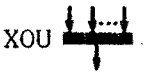
Il suffit d'une ou plusieurs places marquées en entrée pour que la mise à feu puisse avoir lieu.

Emetteur XOU :



En fonction des conditions sur les arcs de sortie, la mise à feu génère une marque sur un seul d'entre eux ; si plusieurs conditions sont vraies, le choix peut être fait de manière aléatoire.

Collecteur XOU :



Il faut une et une seule place marquée en entrée pour que ce collecteur XOU soit tirable.

Emetteur-Collecteur NON :

Cet "émetteur-collecteur NON" n'aura qu'un arc d'entrée et qu'un arc de sortie.
Une marque sera disponible en sortie s'il n'y en a aucune dans la place d'entrée ; réciproquement, dès qu'il y aura une marque dans cette place d'entrée, la marque de sortie sera indisponible.
Cela correspond à une transition sur laquelle l'unique place d'entrée arrive avec un arc inhibiteur [AGEFLY].

Le type de représentation que nous proposons n'exclut pas la présence d'autres conditions sur les variables de la partie opérative au niveau de chaque transition.

Il permet surtout d'exprimer dans le graphisme lui-même les conditions portant sur la disponibilité des données manipulées.

La figure III.9 récapitule ces définitions et donne leur représentation en Réseau de Petri Interprété.

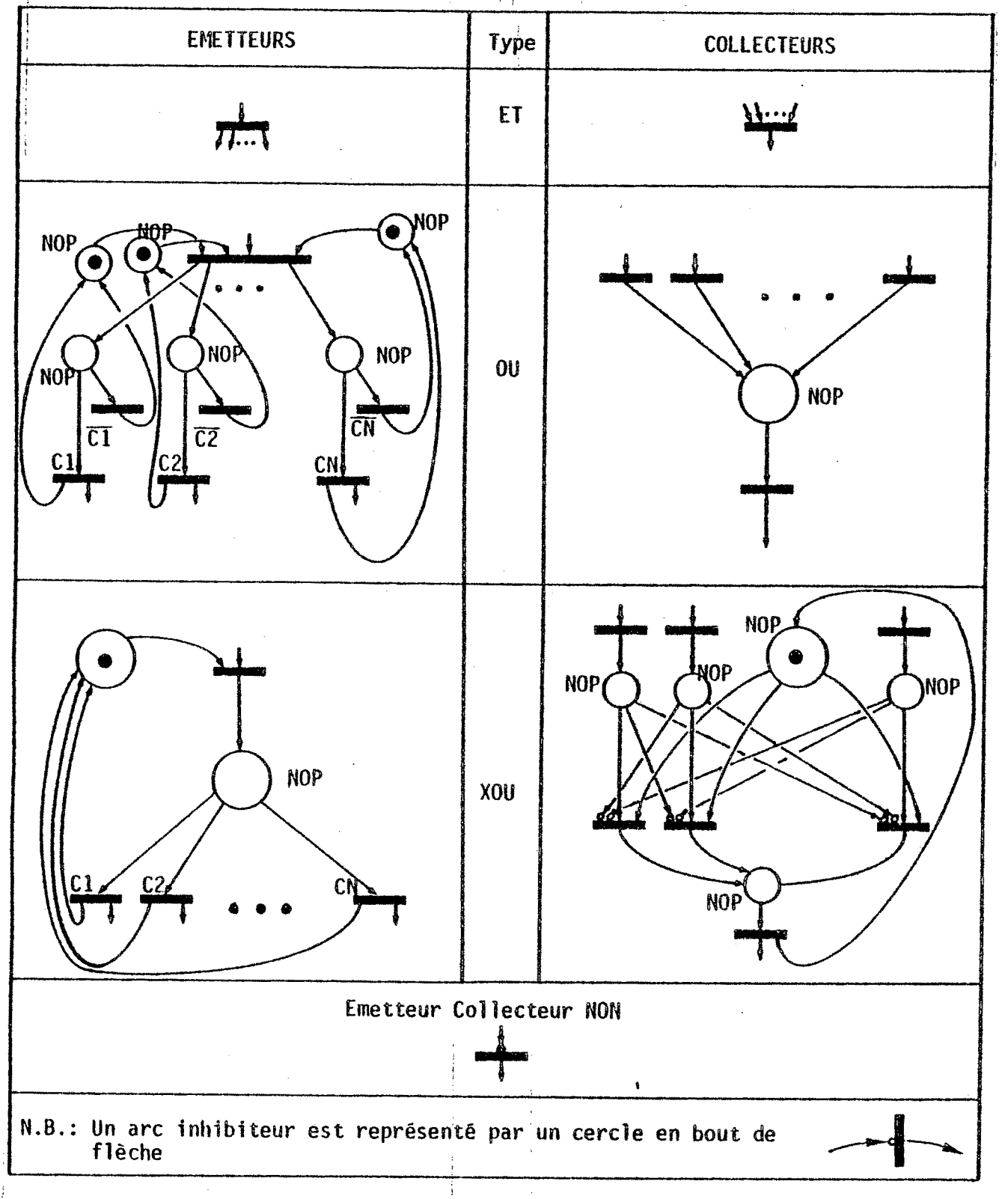


Figure III.9 - RDPI équivalents aux différents types d'Emetteurs et de Collecteurs (avec leur marquage initial)

. Restriction 4 : Nous nous définissons un type de réseau particulier : le graphe normalisé.
Il a la particularité d'avoir un Emetteur de DEBUT en tête, et un Collecteur de FIN pour terminer.
Un réseau d'émetteurs, de collecteurs et de places, tel qu'il a été défini précédemment,sera transformé en graphe normalisé selon l'algorithm suivant :

- toute place dont la relation d'incidence arrière est vide est reliée à l'Emetteur de DEBUT ;
- toute place dont la relation d'incidence avant est vide, est reliée au Collecteur de FIN.

Cette normalisation du réseau implique évidemment une modification du marquage initial.

Ce principe de "graphes normalisés" est à rapprocher des travaux de Valette [VALETT] ; ce dernier, après de nombreuses restrictions sur les réseaux concernés, présente la substitution d'une transition par un réseau "*bienformé*".

Nous avons ajouté cette restriction d'écriture du graphe sous forme normalisée pour deux raisons essentielles :

- faire figurer immédiatement le parallélisme initial d'un réseau donné. C'est ce qu'illustre la figure III.10 où le même graphe est présenté sous les deux formes ;
- permettre le remplacement d'un opérateur quelconque représenté par une place au moyen d'un graphe normalisé complet. Dans le contexte de Polyphème, cela permet de remplacer par exemple l'accès à une relation globale RG par le sous-graphe normalisé combinant les différentes relations pour l'obtention de RG (figure III.10 bis).

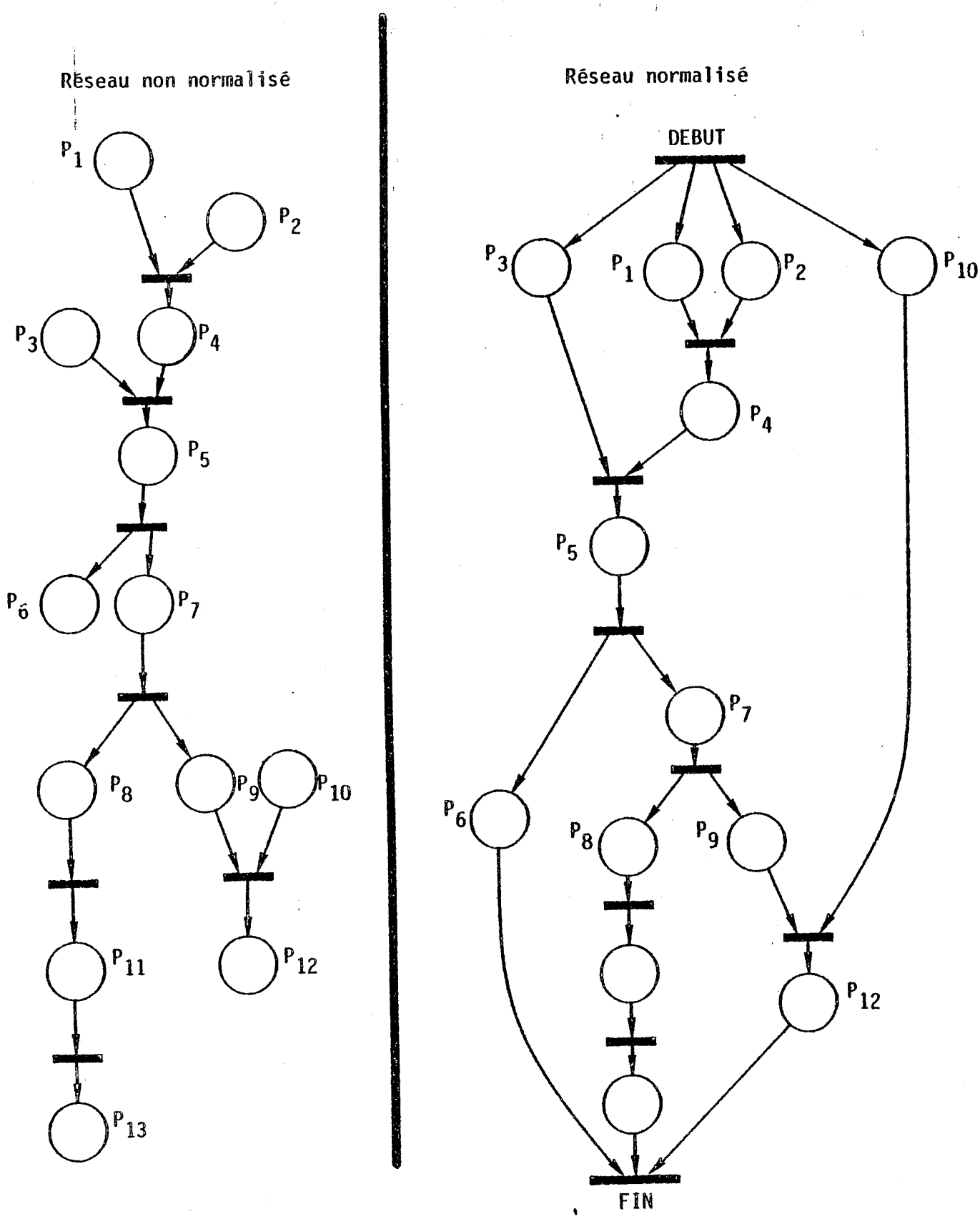


Figure III.10 - Comparaison sur un exemple entre un réseau normalisé ou non

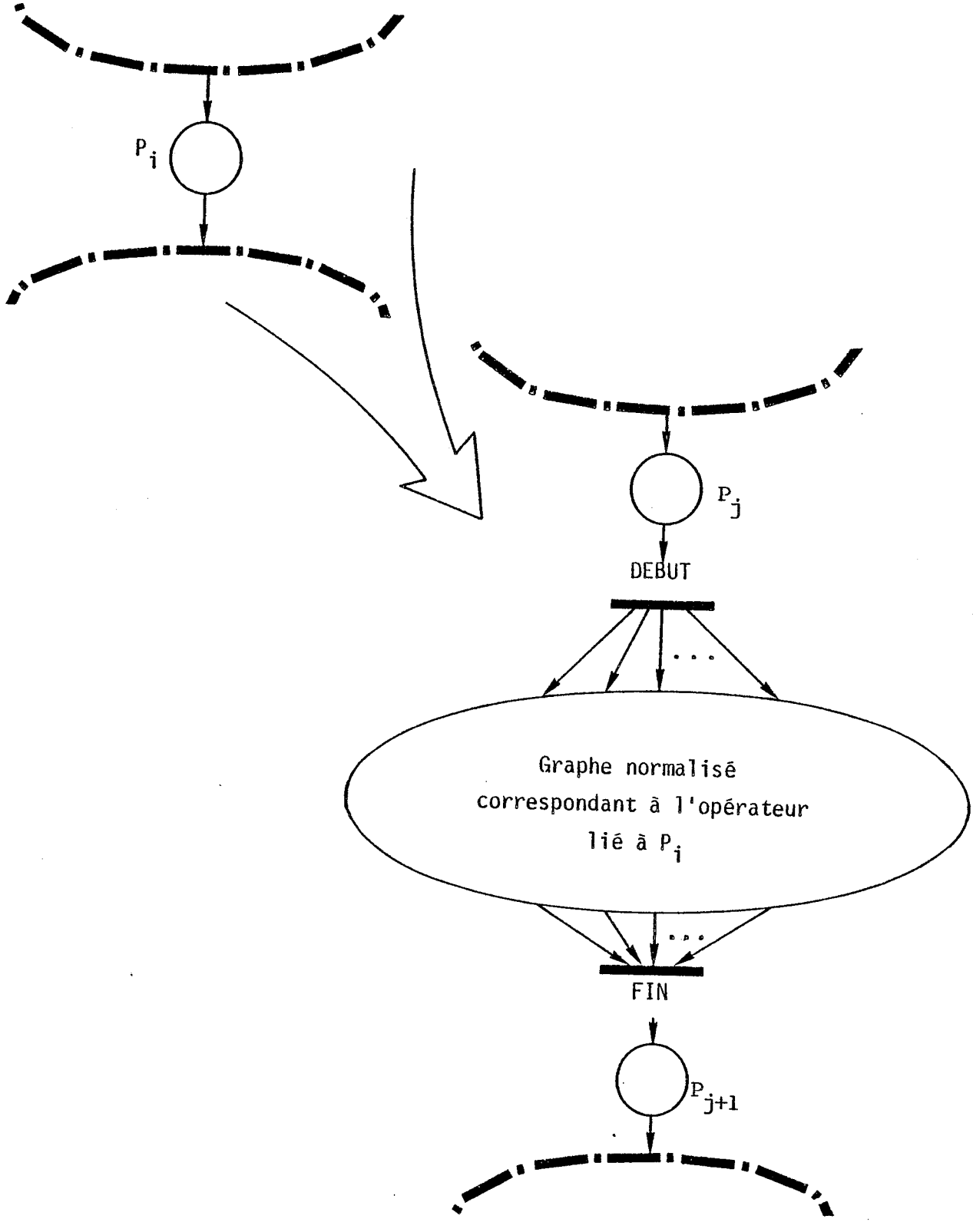


Figure III.10 bis - Remplacement d'une place par un graphe normalisé
($\phi(P_j) = \phi(P_{j+1})$ = {opération neutre}

En résumé, nous avons donc défini les GNEC (Graphes Normalisés avec Emetteurs et Collecteurs) comme un sous-ensemble des Réseaux de Petri Interprétés avec les caractéristiques suivantes :

- graphe marqué (une transition d'entrée et une transition de sortie, au plus, par place)
- transitions type Emetteur 

  ou Collecteur 
- conditions sur les mises à feu des Emetteurs et des Collecteurs (ET, OU, XOU, NON)
- "normalisation" du graphe (un Emetteur de DEBUT et un Collecteur de FIN).

III-2-3-4 Exemple d'utilisation des GNEC

Nous utilisons à nouveau comme exemple celui de la figure III.5.

a) Exécution avec une opération par relation :

Il suffit de reprendre le graphisme de la figure III.6 en précisant les conditions sur les Emetteurs et les Collecteurs (figure III.11).

L'application ϕ d'affectation des opérateurs aux places est la même que celle correspondant à la figure III.6.

Si l'on associe l'arrivée d'une marque à la fin d'opération, l'applicatior ψ donnant les liaisons entre conditions et transitions est entièrement représentée dans le GNEC lui-même.

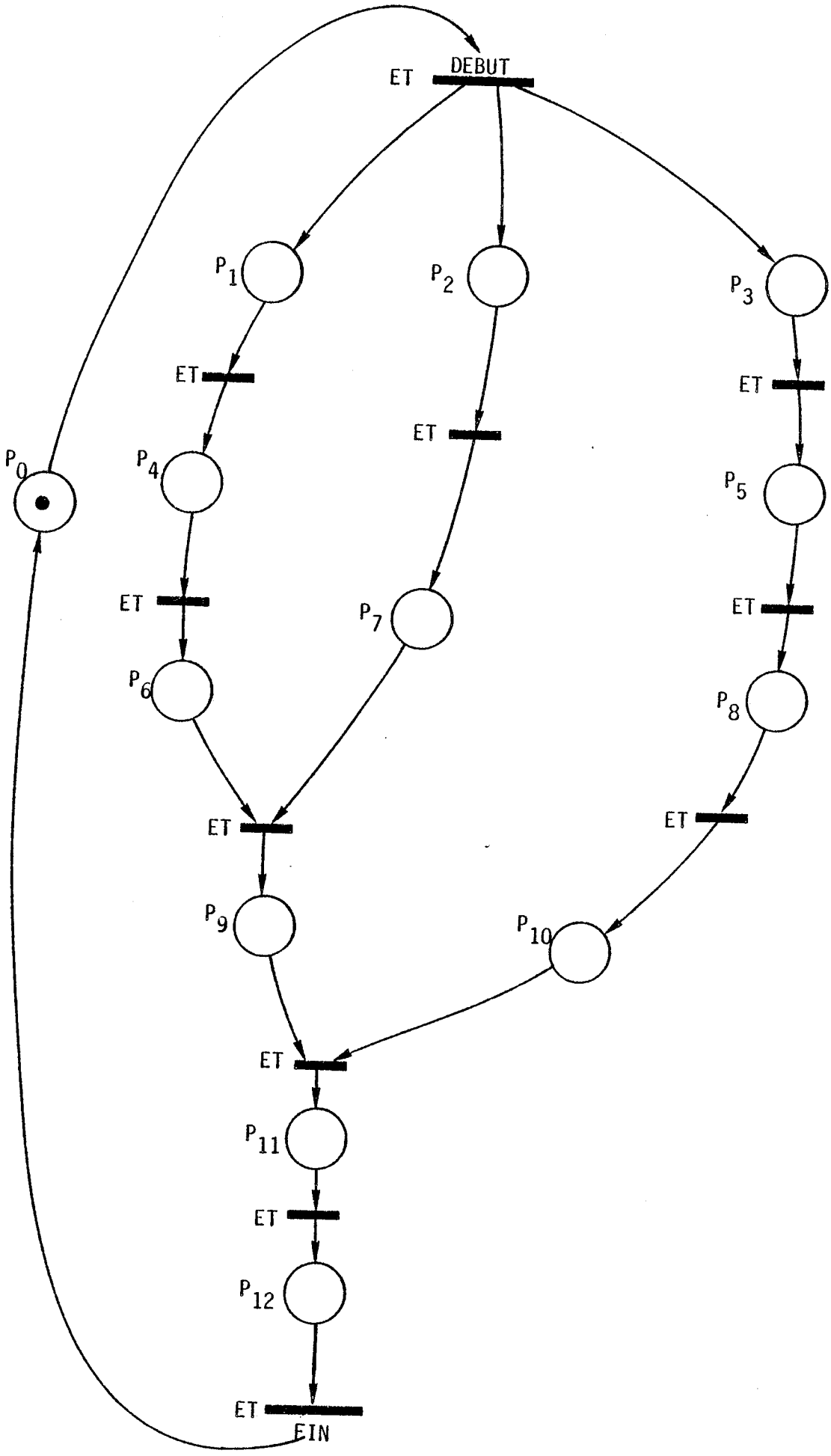


Figure III.11 - Utilisation des GNEC sur l'exemple de la figure III.5 avec une opération par relation

b) Exécution avec une opération par n-uplet :

Dans ce cas complexe, il est souvent préférable de séparer les notions de chemins de contrôle pour les n-uplets et de chemin de répercussion des fins de relation ; il faut alors considérer deux GNEC dont la sémantique d'évolution des marques sera différente :

- dans le GNEC concernant les n-uplets, une marque représentera un n-uplet disponible (figure III.12)
- dans le GNEC lié aux "fins de relation", l'évolution de la marque indiquera la fin des opérations précédentes (figure III.13).

L'interaction entre les deux GNEC se fera par événements et variables de la partie opérative.

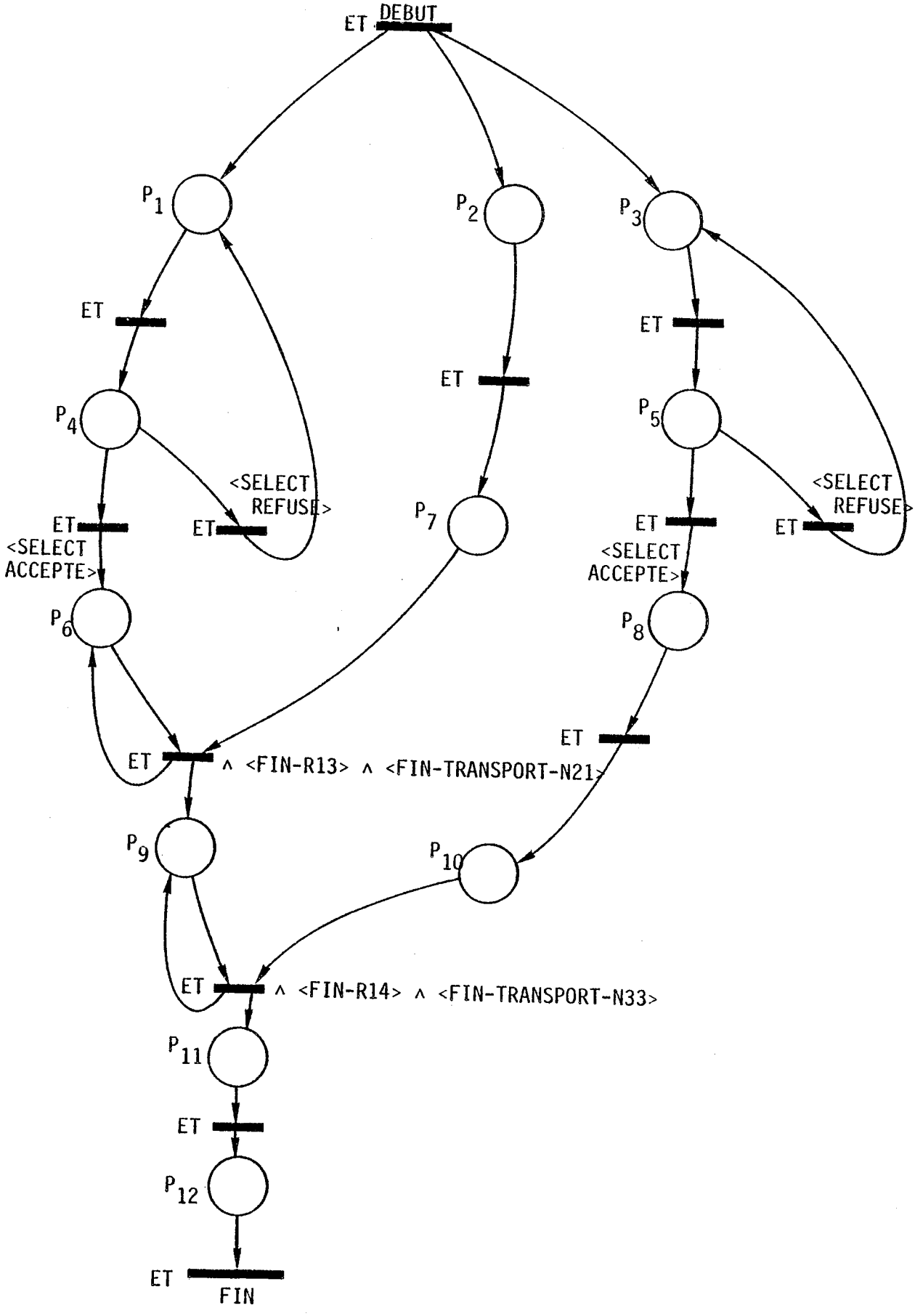


Figure III.12 - GNEC lié aux n-uplets pour une exécution avec une opération par n-uplet de l'exemple fig. III.5
(pour plus de simplicité, nous avons fait figurer les conditions extraites de ψ pour le test sur les variables de la partie opérative)

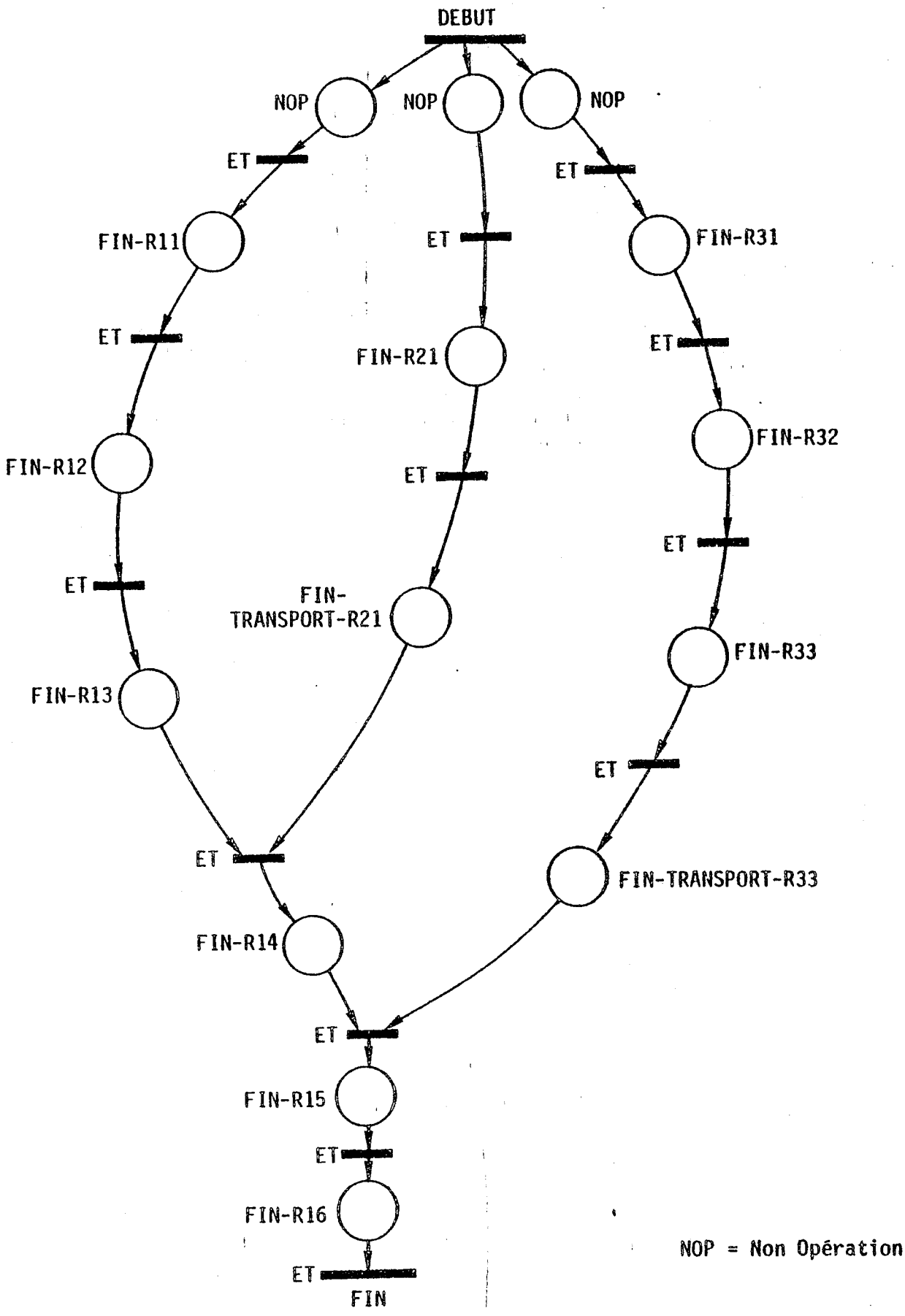


Figure III.13 - GNEC lié aux fins de relations pour une exécution avec une opération par n-uplet de l'exemple fig. III.5. Nous avons fait figurer les booléens qui seront mis à vrai dans la partie opérative en fonction de l'événement testé dans l'Emetteur précédent (non figuré)

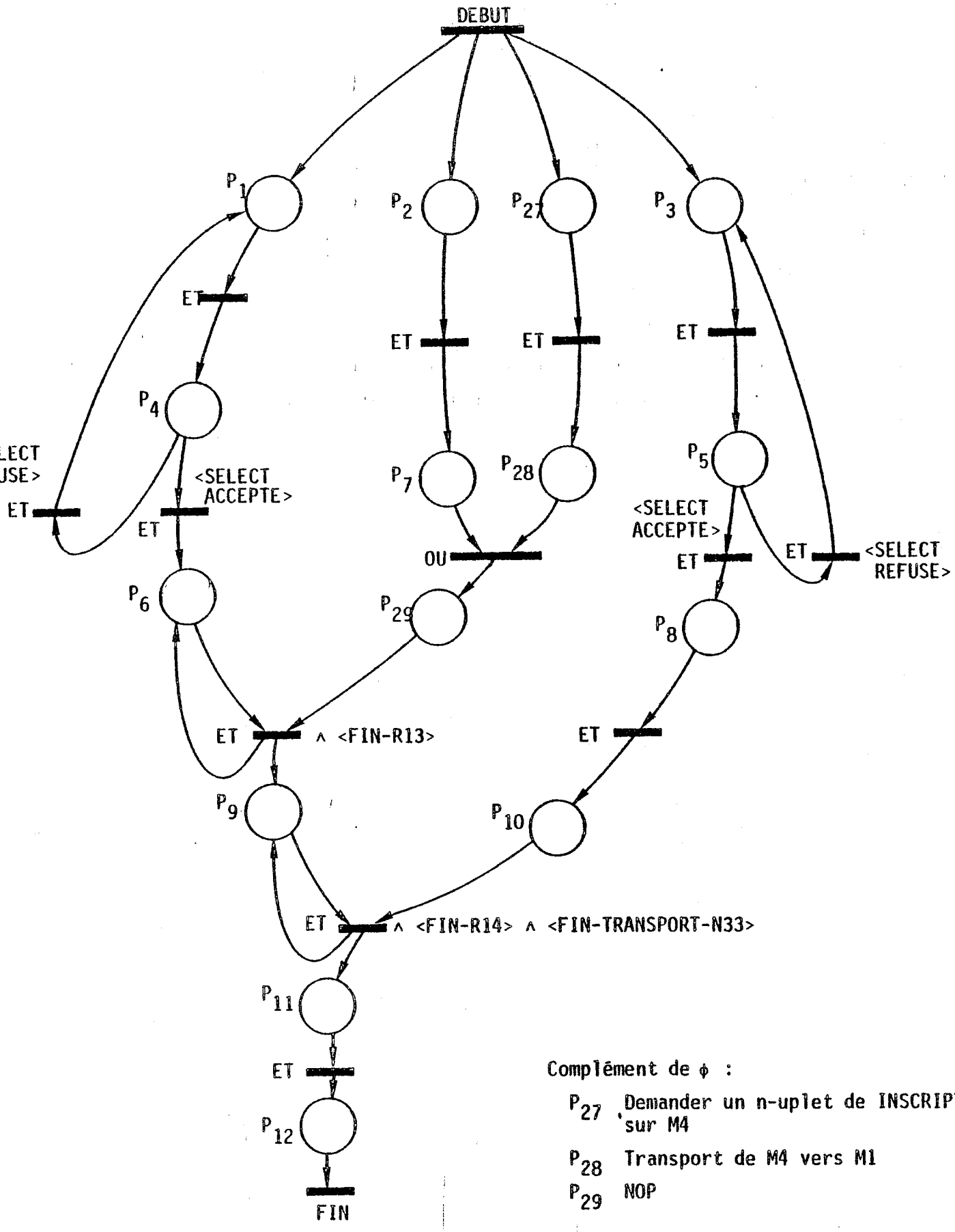
c) Exécution avec une opération par n-uplet et l'emploi de collecteur OU :

Pour illustrer l'emploi d'un collecteur OU, nous allons considérer la relation INSCRIPTION répartie de manière disjointe sur M2 et sur une nouvelle machine locale M4.

Les programmes d'accès à INSCRIPTION sont alors lancés simultanément sur M2 et M4 ; au niveau de M1, il est possible de consommer des n-uplets de cette relation dès qu'ils sont disponibles et quelle qu'en soit leur origine (M2 ou M4).

On obtiendra donc le GNEC de la figure III.14 pour l'évolution des n-uplets.

La fin de la relation INSCRIPTION correspond alors à la fin de transport des éléments venant de M2 et des éléments venant de M4 (voir figure III.15).



Complément de ϕ :

- P_{27} Demander un n-uplet de INSCRIPTION sur M4
- P_{28} Transport de M4 vers M1
- P_{29} NOP

Figure III.14 - GNEC lié aux n-uplets pour une exécution avec une opération par n-uplet de l'exemple figure III.5 en considérant la relation INSCRIPTION répartie sur M2 et M4

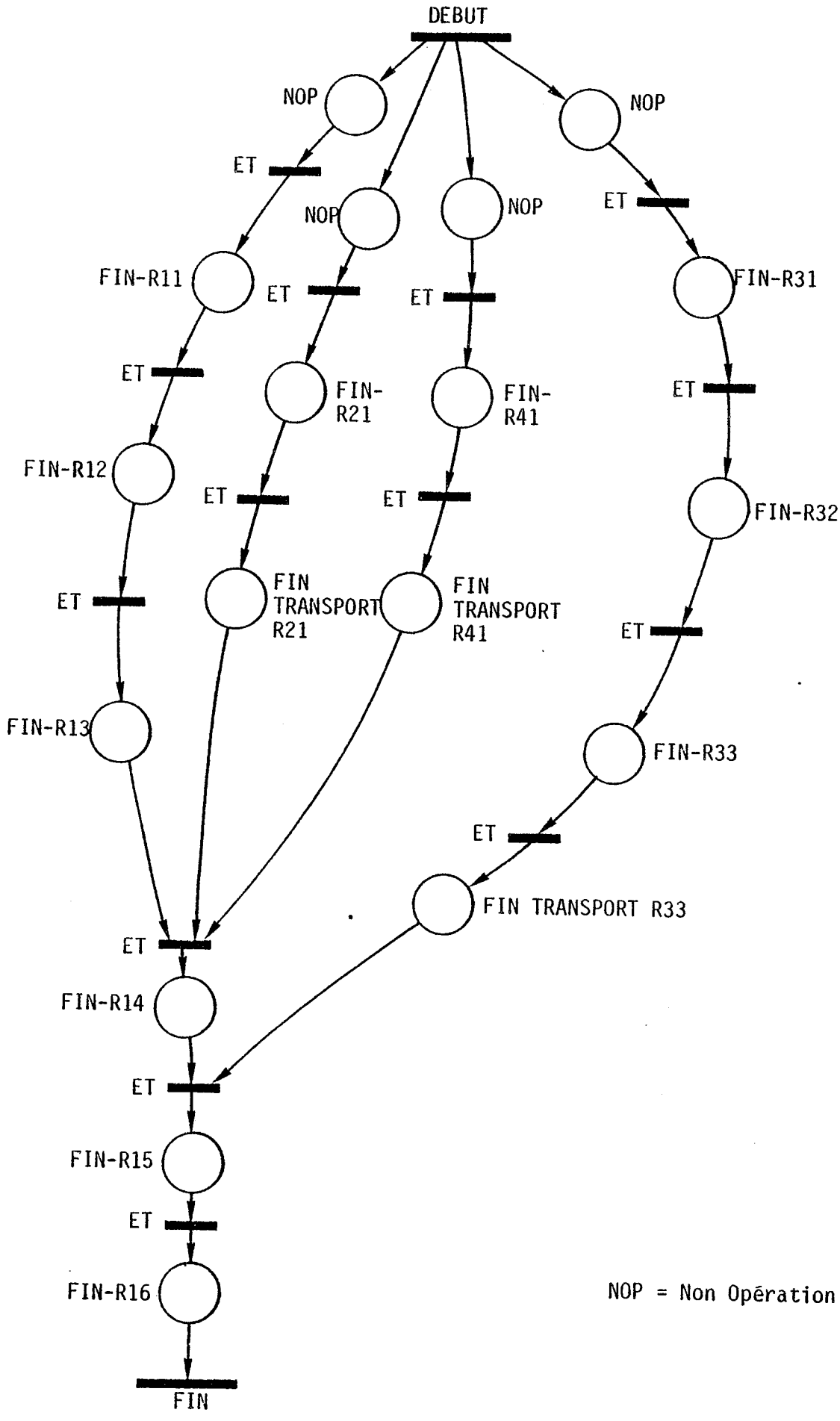


Figure III.15 - Reprise du GNEC lié aux fins de relations de la figure III.13 en considérant la relation INSCRIPTION répartie sur M2 et M4

III-2-3-5 Analyse des GNEC sous l'angle des qualités énumérées au III-1

Ce paragraphe est en fait un développement du tableau présenté en Annexe 1 pour la partie relative aux GNEC.

a) Qualités générales :

. simple :

Une représentation graphique est toujours plus parlante et synthétique qu'un texte de programme pour étudier l'algorithme sous-jacent ; cependant, il ne faut pas néglier la partie décodage du graphe de sa structure interne en machine (tableau) vers une structure effectivement graphique.

Il existe en fait des systèmes graphiques qui permettent de faire afficher sur écran de visualisation des RDPI stockés sous forme de tableaux avec l'évolution du marquage au fur et à mesure de l'interprétation [CAPMOA].

Avec ces systèmes, l'utilisateur peut créer et modifier ses RDPI de manière conversationnelle ; dans notre cas, cet utilisateur sera l'Administrateur de la BDR lors de la création et du test des "Règles globales" (voir chapitre IV).

. modulaire :

Un GNEC est aisément découpable en d'autres GNEC :
tout sous réseau obtenu selon un critère donné (en général le même site d'exécution) peut être transformé en GNEC par le processus de normalisation correspondant à la restriction 4 sur les RDPI (III-2-3-3).

. aisément transportable :

Codables sous forme de tableaux, les GNEC sont facilement transportables puisque assimilables à des données standard ; par exemple aux relations dans un SGBDR relationnel (IV-2).

. sémantiquement denses :

Au moyen de l'application ϕ, il est possible d'associer des opérations prédéfinies sur les sites distants, les interpréteurs se chargeant de les faire exécuter à la demande, ce qui évite de transporter le code correspondant.

. extensible :

Il est toujours possible de rajouter de nouveaux opérateurs à condition de donner aux sites locaux le code permettant de les interpréter ; par exemple, en leur envoyant un GNEC correspondant au nouvel opérateur :
il est en effet possible de remplacer toute place d'un GNEC par un GNEC complet (figure III.11) [VALETT].

. adapté à l'expression du contrôle et du parallélisme :
Les GNEC sont particulièrement adaptés pour exprimer parallélisme et contrôle puisqu'aux possibilités des RDPI dans ce domaine, ils allient une sémantique liée au contrôle de fin de relation dans les SGBDR.

. permettre une pré-analyse des blocages :
Tout comme les RDP sont utilisés pour simuler l'exécution de processus, l'exécution d'un GNEC peut être simulée au niveau global afin de repérer d'éventuels blocages.

Toutefois, dans les hypothèses sur lesquelles nous travaillons, nous avons vu qu'une requête relationnelle globale ne pouvait pas inclure de blocage (II-2-1).

b) Avantages pour l'utilisateur de la BDR :

. transparence :
En permettant un passage aisé du "graphe de donnée" fourni par l'utilisateur de la BDR au schéma d'exécution répartie, les GNEC favorisent la transparence des problèmes de répartition vis-à-vis de cet utilisateur.

c) Avantages pour l'Administrateur de la BDR :

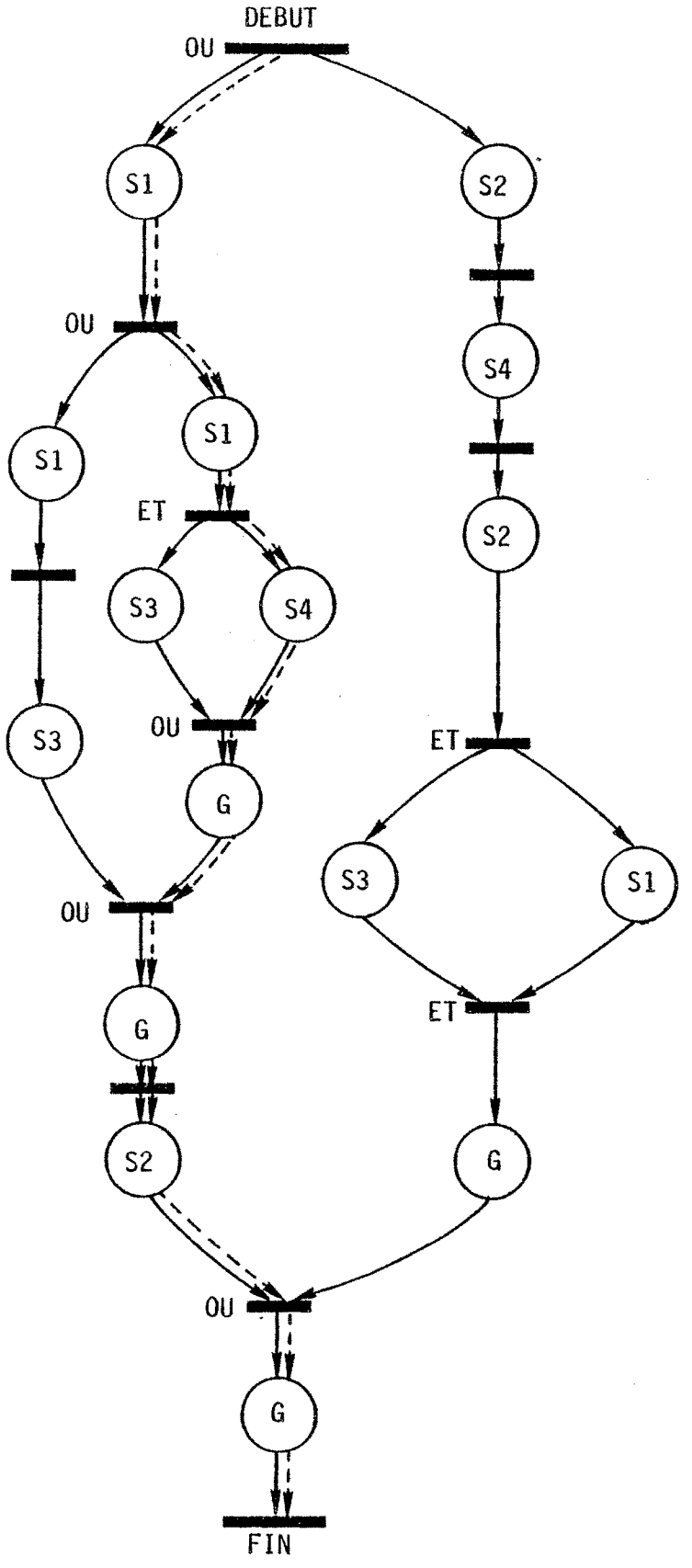
. exprimer les contraintes d'intégrité :
L'Administrateur de la BDR exprime ses contraintes d'intégrité en faisant rajouter des conditions supplémentaires dans l'application ψ, en fonction des différentes relations à atteindre ; d'autres contraintes sont intégrées automatiquement dans le GNEC par substitution d'un opérateur d'accès ou de modification (de relation) par un GNEC filtrant cet accès [STONEB].

. réversibilité à tout état stable de l'exécution :
Une structure graphique comme celle des GNEC permet de bien repérer les modifications déjà exécutées lorsque la granularité d'exécution correspond à un travail relation par relation ; il est possible alors de remonter le GNEC pour revenir à l'état stable précédent [DAVIES].
Par contre, dans une granularité d'exécution "n-uplet par n-uplet", il n'est pas question de retour-arrière dans le graphe vue la complexité due aux différents niveaux d'exécution dans le graphe en cours.

. tolérance aux pannes et optimisation :
La partie de décomposition du graphe de données en GNEC peut exprimer plusieurs moyens d'atteindre la solution demandée en les initialisant sous un émetteur OU.

Lors de l'exécution du GNEC, le choix du chemin dans le graphe se fait après élimination des opérateurs liés aux machines absentes ; l'interpréteur global considère alors les différents chemins possibles à la manière des "AND-type-subgraph" des graphes UCLA-DABG (voir Annexe 2) [MAZARE,BABOES] ; s'il n'existe aucune possibilité d'atteindre le collecteur de FIN avec les machines présentes sur le réseau, la transaction globale est refusée, sans accès réseau inutile ; par contre, si plusieurs chemins sont réalisables, le choix se fait par pondération des opérateurs (coût, temps, volume de transfert ...) et élection d'un optimum (voir figure III.16).

Dans certains cas mêmes où la vitesse prime le coût, une interrogation peut être posée de plusieurs manières en même temps ; de même, une modification est préparée par divers chemins dans le graphe pour être, bien entendu, réalisée de manière unique finalement.



Si : Opérateur devant s'exécuter sur le site "i"

G : Opérateur global

Si la liaison vers le site S3 est coupée, on peut encore répondre à la transaction par le graphe en pointillé.

Par contre, une panne sur S1 ou S2 empêche toute exécution pour la transaction représentée par ce graphe.

Figure III.16 - Exemple de GNEC avec plusieurs chemins possibles pour une même requête

Chapitre IV

Implantation de Graphes Normalisés avec Emetteurs et Collecteurs

Afin d'étudier pratiquement ces Graphes Normalisés avec Emetteurs et Collecteurs (GNEC), nous avons réalisé une Maquette de SGBDR utilisant ce type de graphe de commande.

Cette maquette est construite à partir de deux logiciels pré-existants sur le réseau utilisé :

- le niveau système réparti est offert par PERE-MERE déjà décrit en III-2-2 ;
- les niveaux de stockage relationnels représentés par LAMB [ANDRAD] et les outils de l'Algèbre Relationnelle manipulant des éléments stockés dans LAMB : GLOP [PAIK].

Nous décrivons donc tout d'abord le système LAMB ; nous donnons ensuite le mode de stockage des GNEC dans LAMB en précisant l'algorithme de substitution d'un opérateur par un GNEC complet.

Ceci nous permet ensuite de proposer une architecture pour la maquette de SGBDR avec la solution que nous avons employée pour la décomposition et l'exécution de requêtes globales dans un tel système.

IV-1 LAMB [ANDRAD]

Conçu comme une Base de Données Relationnelle, LAMB (Logiciel d'Accès et de Manipulation de Blocs) est un système composé de primitives de très bas niveau travaillant sur des *Tableaux* (ou Relations) composés de *N-uplets*.

Chaque tableau identifié par un IDT (Identificateur de Tableau) correspond à un *Type* donné de N-uplets. Le type permet de retrouver la description des différents *Champs* des N-uplets pouvant appartenir à ce tableau. Toutes les descriptions de type sont regroupées dans l'un des trois tableaux prédéfinis : le tableau des Types.

Les deux autres tableaux prédéfinis sont :

- le tableau Maître qui contient les descripteurs de tous les tableaux existants et notamment la relation IDT-NOMEX (Nom externe) qui permet à l'utilisateur de LAMB d'identifier son tableau de l'extérieur ;

- le tableau des N-uplets de Contrôle qui décrit en fait les champs des N-uplets.

Un N-uplet est identifié par son IDU (Identificateur de N-uplet) ainsi que par sa clé dont les champs composants sont précisés dans le tableau des Types. Sa position dans le tableau (ou n° de N-uplet) peut aussi être utilisée.

Un des grands intérêts de LAMB implémenté sur un Espace Virtuel statique (celui d'URANUS [NGUY77]) est que les tableaux peuvent être déclarés avec une taille importante.

Les primitives de LAMB, outre la gestion des types et des tableaux, permettent surtout de manipuler les N-uplets comme des éléments d'une relation ; on peut en effet :

- créer un n-uplet,
- tuer un n-uplet,
- accéder à un n-uplet (par n° de n-uplet ou par clé),
- modifier les champs n'appartenant pas à la clé.

Il est aussi possible de parcourir un tableau, de n-uplet en n-uplet, de manière séquentielle.

Il faut ajouter à LAMB le logiciel GLOP [PAIK] qui permet d'activer des opérateurs de l'Algèbre Relationnelle sur les éléments de LAMB :

- Select sur un tableau ;
- Project d'un tableau ;
- Différents types de JOIN entre deux tableaux ;
- Union de deux tableaux ....

IV-2 Stockage des GNEC dans LAMB

IV-2-1 Présentation

Cette implantation va se faire au moyen de trois tableaux différents par GNEC, chaque tableau regroupant un type de noeud :

- le tableau des Emetteurs (EMET)
- le tableau des Collecteurs (COLL)
- le tableau des Opérateurs (OPER).

L'association des trois tableaux correspondant à un même GNEC est faite par un n-uplet d'un tableau général appelé GRAF.

Outre le nom du GNEC et les "Identificateurs de ses Tableaux" (Idt de LAMB), le n-uplet de GRAF comprend également (figure IV.1) :

- un pointeur vers l'émetteur de DEBUT ;
- un pointeur vers le collecteur de FIN ;
- des informations sur le propriétaire de ce graphe et les restrictions d'exécution qui peuvent intervenir (confidentialité).

Les arcs des GNEC sont matérialisés par des pointeurs dits pointeurs AVAL ; ils sont "doublés" par des pointeurs AMONT dont le principal avantage est de permettre un éventuel parcours arrière du graphe (II-2-3-5-c).

Un Emetteur est donc composé de plusieurs pointeurs AVAL et d'un pointeur AMONT ; à l'inverse un Collecteur possède plusieurs pointeurs AMONT pour un pointeur AVAL.

Un noeud Opérateur qui ne doit avoir qu'un arc entrant et qu'un arc sortant possède donc un pointeur AVAL et un pointeur AMONT.

Ces pointeurs sont en fait le "Numéro de n-uplet" de Lamb préfixé d'une lettre "E", "C" ou "O" indiquant à quel type appartient le noeud référencé.

Emetteurs, Collecteurs et Opérateurs contiennent une *Variable d'état* dont la sémantique est étudiée plus loin.

Les applications ϕ et ψ sont incluses directement dans les noeuds correspondants ; dans une première réalisation, nous n'avons pas voulu tenir compte de conditions complexes par émetteur ou collecteur (ψ) ou d'opérations composées (ϕ).

Pour plus de détails sur la structure exacte de ces divers n-uplets, consulter [EUZE78].

Nous allons voir à présent comment ce système permet le remplacement d'un opérateur par un GNEC de manière automatique et rapide.

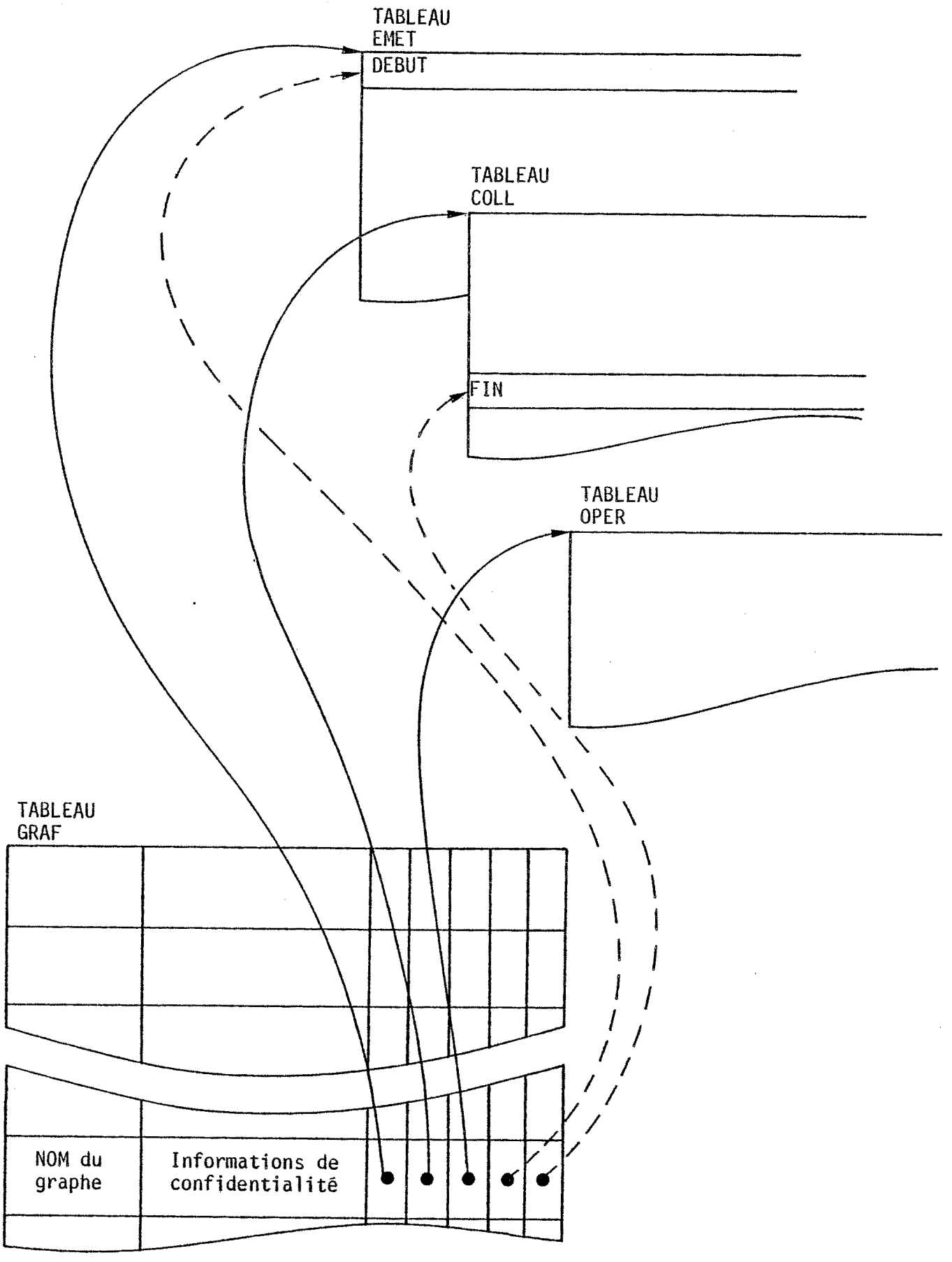
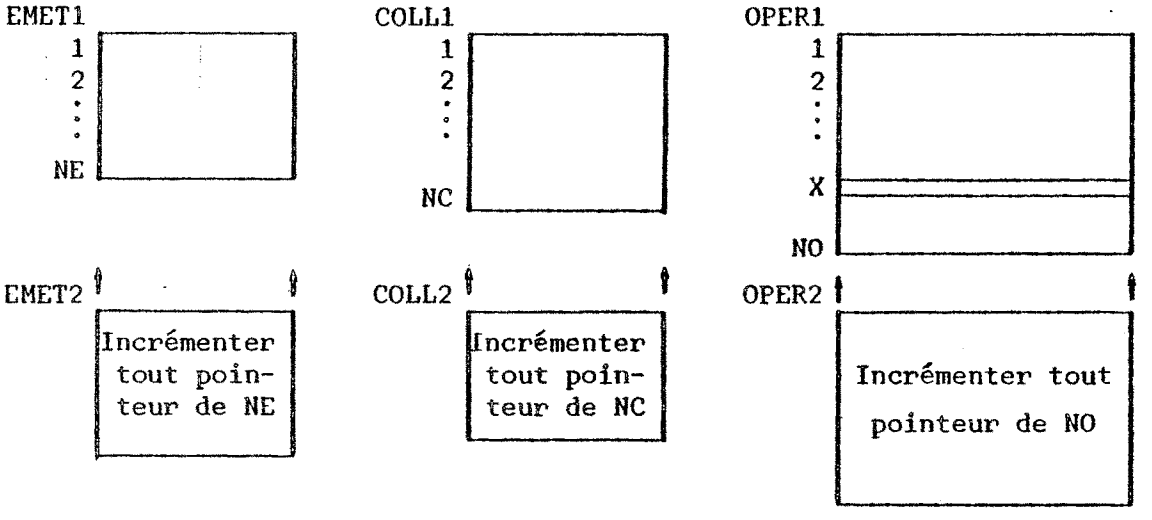


Figure IV.1 - Implantation d'un GNEC dans LAMB

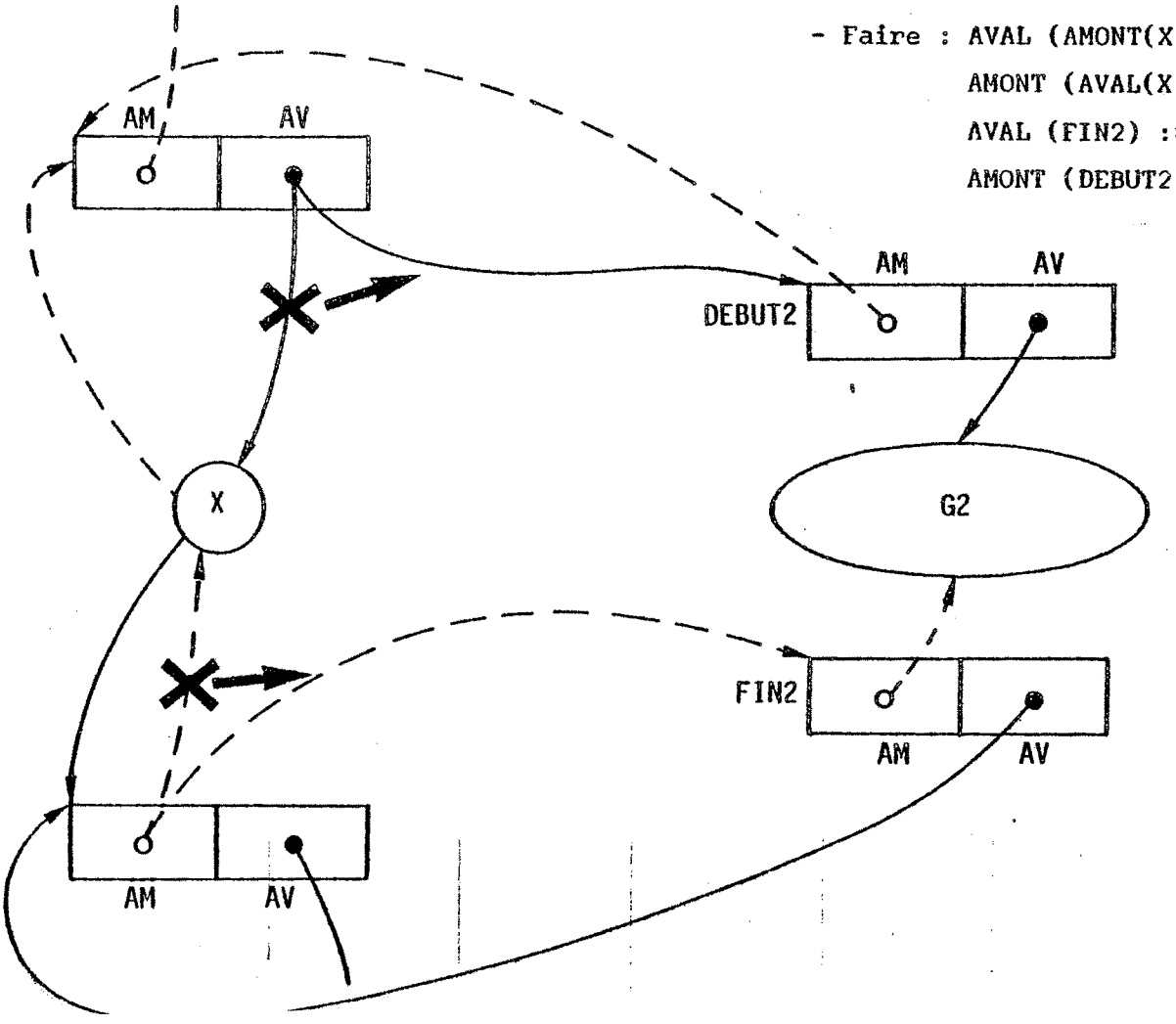
IV-2-2 Substitution d'un opérateur par un GNEC

L'algorithme d'insertion d'un graphe normalisé G2 dans un autre graphe normalisé G1 à la place d'un opérateur X peut s'écrire de la manière suivante :

- Ajouter les tableaux de G2 à la suite des tableaux de même type de G1, en incrémentant toutes les valeurs de G2 des maxima correspondants de G1.

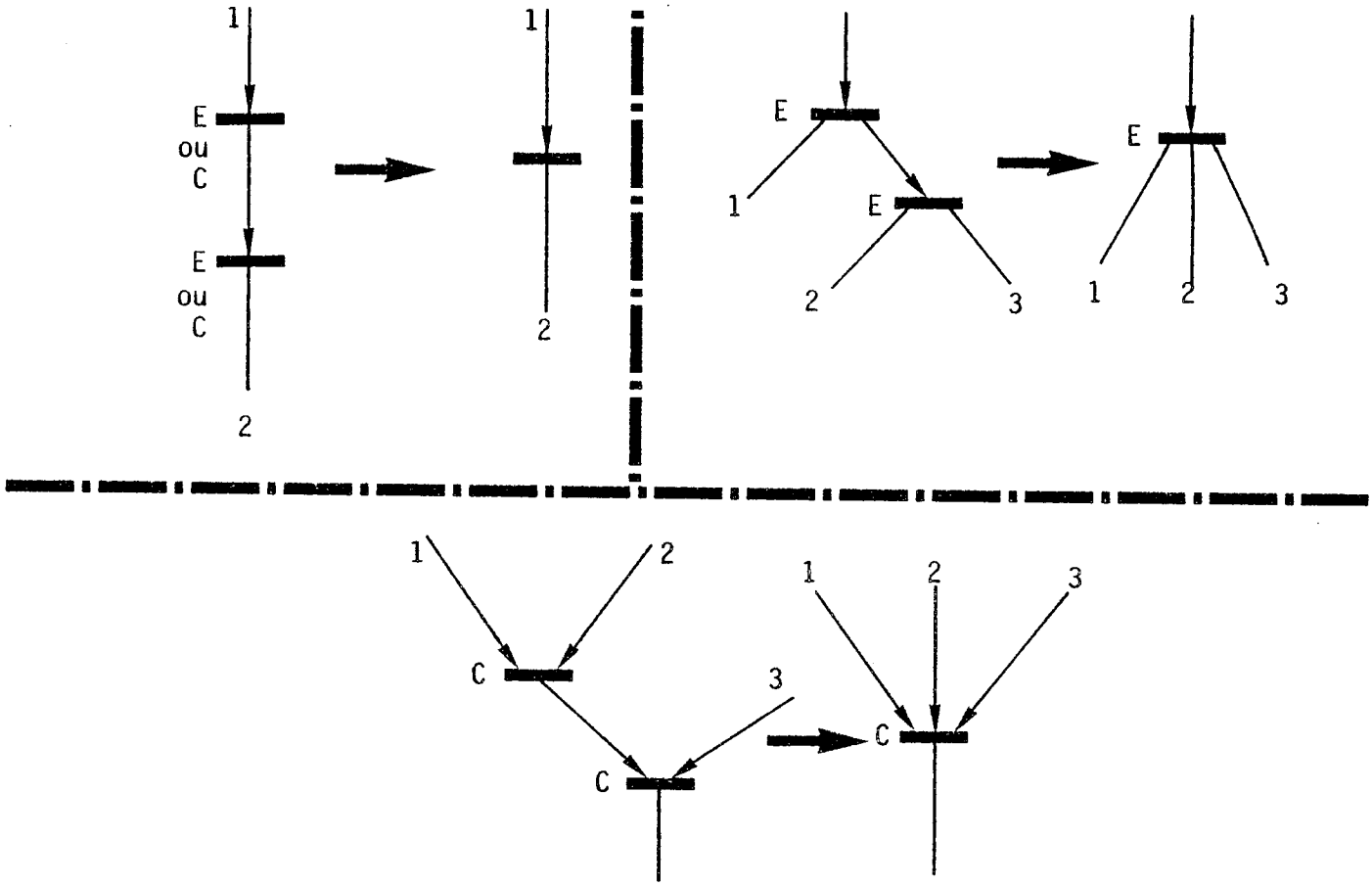


- Faire : AVAL (AMONT(X)) := DEBUT2
  AMONT (AVAL(X)) := FIN2
  AVAL (FIN2) := AVAL(X)
  AMONT (DEBUT2) := AMONT(X)

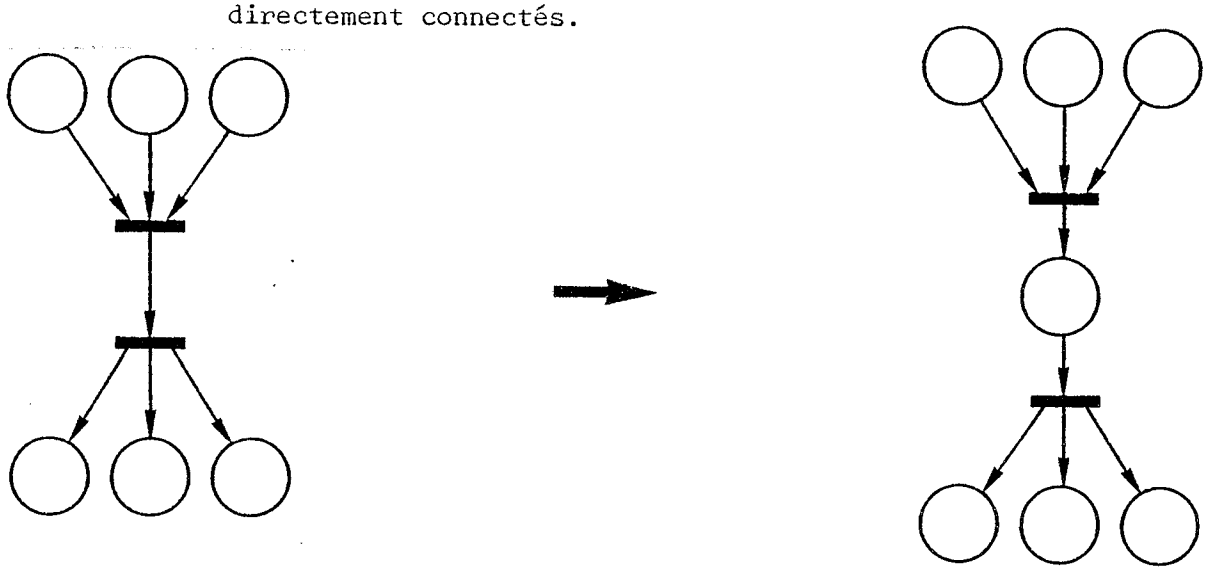


- Supprimer l'opérateur X par positionnement de la variable d'état à valeur nulle (opérateur fictif).

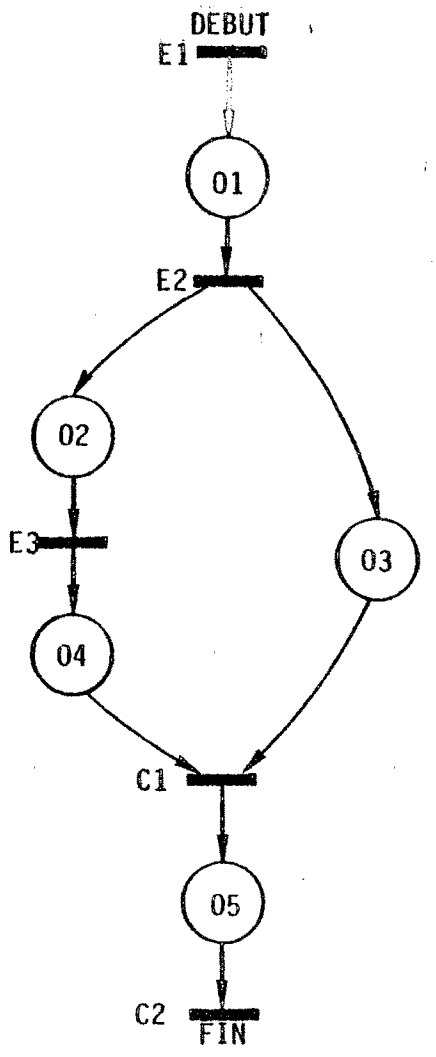
- Supprimer éventuellement des Emetteurs ou des Collecteurs inutiles (faisable aussi quand toutes les insertions sont terminées).



- Rajouter des opérateurs fictifs entre les émetteurs ou collecteurs directement connectés.



La figure IV.2 (abc) donne un exemple d'application de cet algorithme.



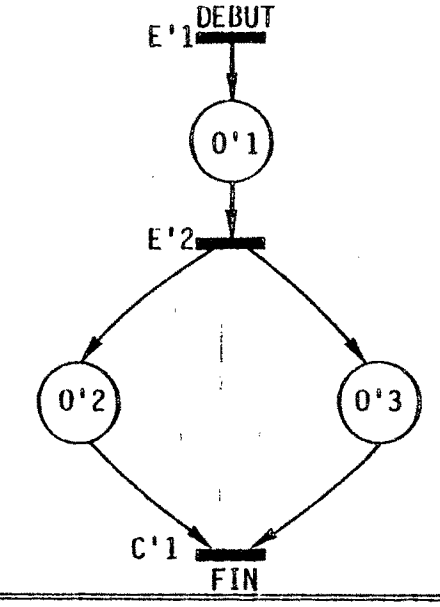
EMET1	AMONT	AVAL1	AVAL2
DEBUT 1		01	
2	01	02	03
3	02	04	

COLL1	AVAL	AMONT1	AMONT2
1	05	04	03
2		05	

FIN

OPER1	AMONT	AVAL
1	E1	E2
2	E2	E3
3	E2	C1
4	E3	C1
5	C1	C2

GNEC G1



EMET2	AMONT	AVAL1	AVAL2
DEBUT 1		0'1	
2	0'1	0'2	0'3

COLL2	AVAL	AMONT1	AMONT2
1		0'2	0'3

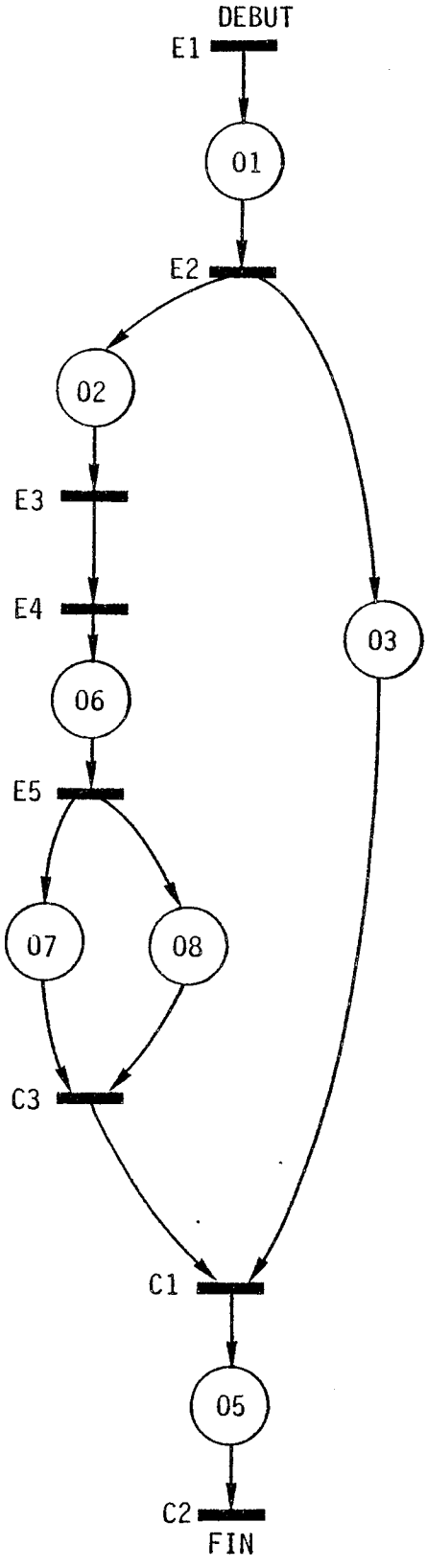
FIN

OPER2	AMONT	AVAL
1	E'1	E'2
2	E'2	C'1
3	E'2	C'1

GNEC G2

Partant du GNEC G1, on remplacera l'opérateur 04 par le GNEC G2.

Figure IV.2.a - Présentation des Graphes G1 et G2

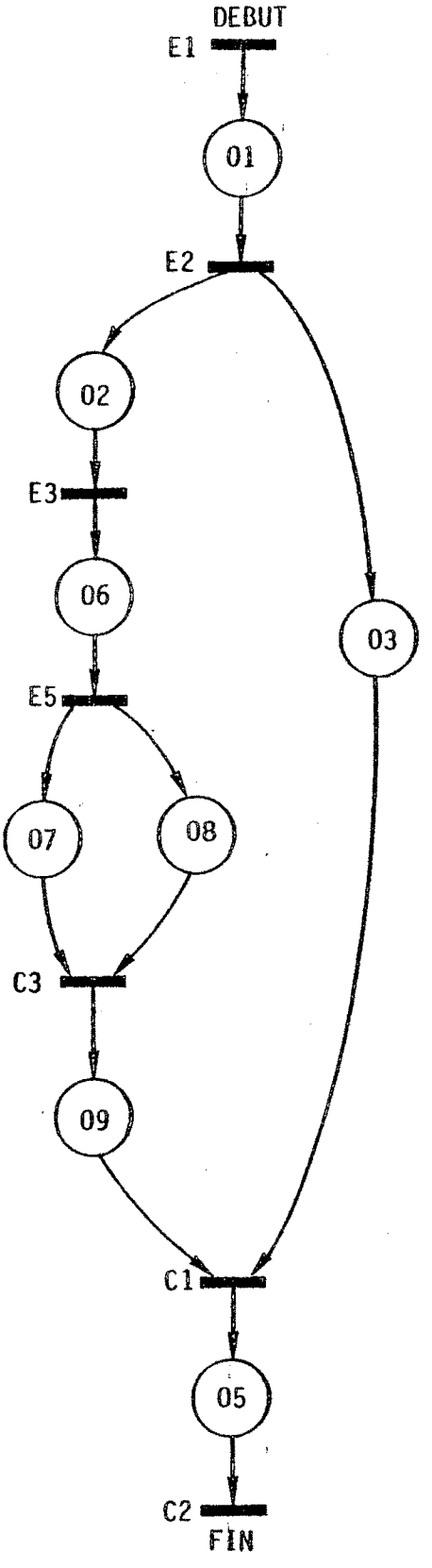


EMET	AMONT	AVAL1	AVAL2
1 DEBUT		01	
2	01	02	03
3	02	E4	
4	E3	06	
5	06	07	08

COLL	AVAL	AMONT1	AMONT2
1	05	C3	03
2 FIN		05	
3	C1	07	08

OPER	AMONT	AVAL
1	E1	E2
2	E2	E3
3	E2	C1
4		
5	C1	C2
6	E3	E5
7	E5	C3
8	E5	C3

Figure IV.2.b - GNEC résultant du remplacement de 04 de G1 par G2



Graphe G1 + G2 après assainissement :

- suppression de E3 ;
- insertion d'un opérateur fictif 09.

Figure IV.2.c - Après assainissement

IV-3 Architecture de la Maquette Polyphème

La maquette Polyphème utilisant les GNEC est principalement basée sur le logiciel LAMB présenté au paragraphe précédent et sur PERE-MERE décrit en III-2-2. On retrouve ces deux logiciels aussi bien au niveau Machine Globale que Machine Locale.

D'un point de vue "critères d'exécution" (Chap. II), les hypothèses prises dans cette maquette sont les suivantes :

- Granularité d'exécution
  Une opération par relation ;
- Localisation pour "algorithmes-commandes-données" (Cas 6)
  Algorithmes implantés
  + Système de commande réparti
  + Données non centralisées ;
- Lieu de traduction du graphe de commandes
  Interprétation répartie.

Nous allons décrire cette maquette en suivant l'évolution d'une requête globale dans les différentes phases qui l'amènent au niveau *exécution répartie* selon la figure II.1.

IV-3-1 Machine Globale

Pour cette maquette, nous nous sommes donné comme forme de Requête Globale de départ le GNEC global correspondant ; cela signifie d'une part que nous n'avons pas voulu aborder les problèmes de traduction de cette requête exprimée en langage de manipulation relationnel vers un graphe de donnée. D'autre part, compte tenu de l'hypothèse faite d'*une opération par relation*, le graphe de donnée correspondant à la requête globale peut être directement assimilé à un graphe de commande.

Ce GNEC global s'adresse donc à des relations de la Vue Globale et il doit être tout d'abord localisé ; les données permettant cette localisation sont mémorisées dans LAMB sous forme de GNEC :

Tout accès ou modification relatifs à une relation globale sont remplacés par le GNEC comportant les actions correspondantes sur les relations des vues locales.

Ces GNEC de localisation sont créés par l'Administrateur de la BDR puisque lui seul connaît les différentes Vues Locales et leur combinaison pour créer la Vue Globale. Il peut disposer à cet effet d'un langage de manipulation spécial prenant en compte les notions de localisation et analysant automatiquement la légalité des GNEC ainsi proposés (voir Annexe 2). On peut voir par exemple qu'une portion de GNEC du type de la figure IV.3 n'a aucun sens.

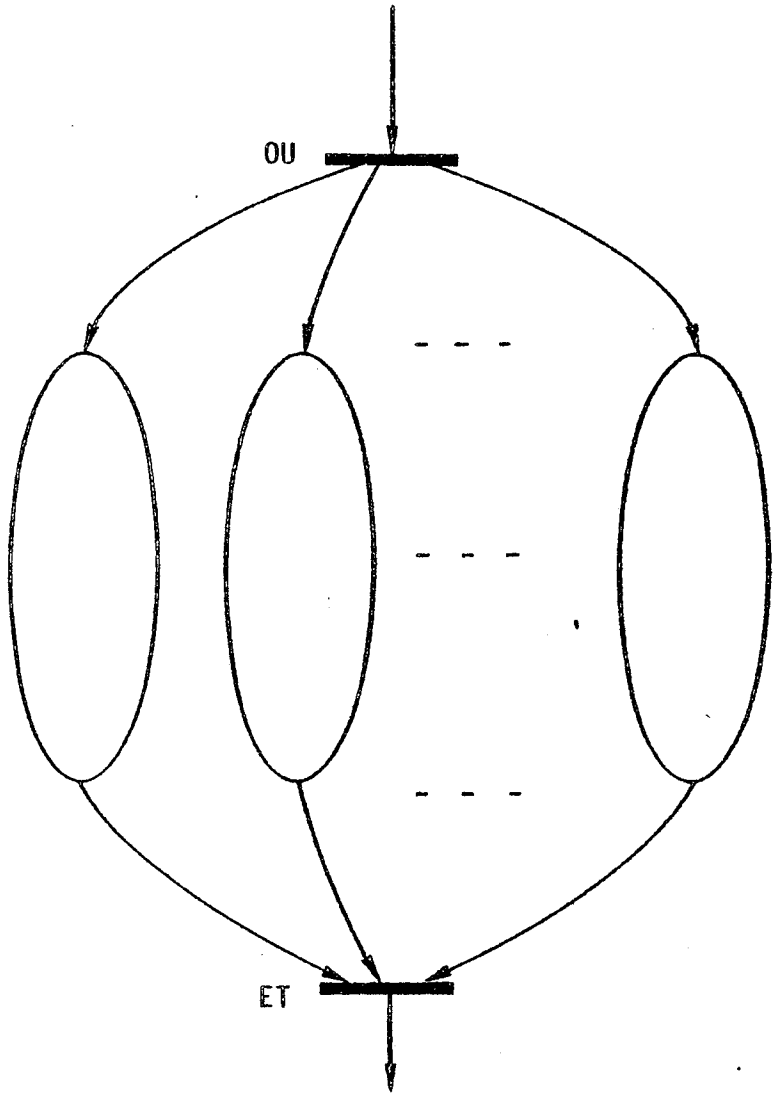


Figure IV.3 - Exemple de portion de GNEC illégale

Les GNEC de localisation ainsi catalogués dans LAMB correspondent en fait à la notion de *Règle Globale* développée dans [ADIBA-CALECA].

Il est facile de voir que l'on peut concevoir des GNEC de localisation standard correspondant aux règles globales implicites immédiatement déductibles du type de répartition de la relation globale concernée.

Par exemple, toutes les relations globales RG_i définies comme l'union sur les "n" sites du réseau de relations locales RL_{ij} ($RG_i = \underset{j=1,n}{\cup} RL_i$) ont la même structure de GNEC de localisation pour l'interrogation et pour les modifications (si les critères de répartition sont identiques).

C'est également lors de cette étape de localisation que doivent être rajoutés les contrôles d'intégrité et de confidentialité préparés par l'Administrateur de la BDR.

Ces diverses juxtapositions de GNEC ayant été effectuées selon le processus indiqué au paragraphe précédent, il est possible d'envisager la phase d'optimisation du graphe par exemple selon les principes proposés par [CALECA].

Une fois cette optimisation effectuée, on est en présence d'un graphe de commande localisé. concernant différentes machines locales de la BDR ; ce GNEC est alors découpé en GNEC mono-machines toujours stockés sous forme de relations dans LAMB.

IV-3-2 Transport des GNEC mono-machines

Ces graphes mono-machines sont alors envoyés vers les machines locales concernées en utilisant les mêmes commandes que celles des relations standard :

ENVOYER une relation

RECEVOIR une relation.

Ces commandes utilisent le système PERE-MERE en permettant la transmission d'un LAMB à l'autre de relations complètes :

une paire ENVOYER-RECEVOIR entre deux machines M1-M2 est implantée sur une interaction M1-M2, ce qui asservit l'ENVOYER aux possibilités du RECEVOIR avec le système d'anticipation décrit en III-2-2.

IV-3-3 Machine locale

Au niveau de cette maquette, nous n'avons pas voulu rentrer dans les détails d'implantation de vues relationnelles sur des bases de données hétérogènes [ADIBA, NGUY77] ; nous avons donc simulé le niveau Machine Locale relationnelle décrit en I-11 grâce à LAMB (stockage, accès et manipulation) et à GLOP [PAIK] (opérateurs relationnels).

Une fois reçu, le GNEC relatif à cette Machine Locale va être analysé par un Interpréteur Local écrit en PL/1 et faisant évoluer une marque dans le graphe selon les modalités indiquées au paragraphe III-2-3-3. C'est à ce niveau qu'interviennent les variables d'état qui indiquent si tel opérateur a ou non été activé ou encore s'il est en attente (pour un ENVOYER ou un RECEVOIR). De même, un collecteur ET peut se trouver en attente de l'arrivée de marques sur les opérateurs en amont ; c'est ce que matérialise la variable d'état en étant mise à la valeur "Attente" dès qu'une marque est apparue dans un des opérateurs en amont. Ce collecteur ne sera mis à feu, bien entendu, que lorsque tous ses opérateurs en entrée sont à l'état "Terminé", ce qui revient à dire que leur marque est disponible en sortie.

Nous ne voulons pas entrer plus avant ici dans les détails d'implantation de cette maquette [EUZE78]. Nous insisterons simplement sur le fait que, compte tenu des hypothèses prises dans cette maquette, les synchronisations entre machines se limitent aux communications de relations par paire ENVOYER-RECEVOIR au moyen de procédures intégrées dans les Interpréteurs et gérées par PERE et MERE.

CONCLUSION

Cette étude nous a permis d'évoluer en fait dans deux directions :

- une analyse des différentes stratégies d'exécution de requête envisageables dans un SGBDR;
- l'élaboration d'un modèle graphique utilisable pour l'exécution répartie de requêtes.

La partie analyse des critères d'exécution (Chapitre II) récapitule des problèmes qui se posent dans toute exécution de requête répartie; il est en effet possible de remplacer le terme "n-uplet" par "enregistrement", "relation" par "fichier", "sélection" par "tri"... suivant le système considéré.
L'avantage du contexte relationnel dans lequel nous nous sommes placés est principalement la rigueur de la terminologie correspondante qui évite les fréquentes ambiguïtés du type "segment-enregistrement-item".
Cette analyse ne peut cependant pas être exhaustive puisque nous avons choisi une architecture simple pour illustrer l'utilisation du modèle graphique.

Des prolongements intéressants de cette étude sont envisageables en étendant progressivement le champ d'investigation :

- Considérer un réseau ayant une architecture dynamique, c'est-à-dire pouvant tomber en panne (!) et avoir des machines locales qui disparaissent puis réapparaissent avec incohérence des objets dupliqués; l'architecture dynamique implique également la possibilité de créer et supprimer des machines locales au gré de l'Administrateur de la BDR.
- Activité simultanée de plusieurs machines globales ayant alors à synchroniser leurs modifications sur les différentes machines locales; dans un tel cas, intervient la gestion et le contrôle des accès concurrents et des interférences sur les bases locales [ADIBCO, GARDAR, ROTGOO].

- Introduction d'utilisateurs locaux "ignorant" l'existence de la BDR et "manipulant" à leur gré la base locale de leur site; cela implique une répercussion dans la BDR de toute modification d'objet dupliqué afin d'assurer la cohérence globale [ROGDBE, SESEWI, WILMS]

L'autre partie de cette étude indique comment utiliser les Réseaux de Petri Interprétés comme support d'exécution répartie dans le contrôle des SGBDR.
Le modèle graphique (GNEC) conçu à partir de ces RDPI possède selon nous les principales propriétés suivantes :

- expression du parallélisme et du contrôle,
- compacité,
- stockage sous forme de tableaux,
- notion de ET, OU, XOU et NON,
- généralité d'utilisation

Nous insisterons surtout sur cette dernière propriété qui indique que tout contexte d'exécution répartie peut tirer profit de l'utilisation des GNEC; les GNEC sont en effet conçus pour gérer et synchroniser des exécutions multi-processeurs; ils peuvent donc s'appliquer par exemple à la gestion de synchronisations dans une machine multi-microprocesseurs du type de celles vers lesquelles tend l'architecture des ordinateurs.

Pour que de telles utilisations des GNEC soient implantées, il est indispensable de concevoir un Langage de Description de ces graphes permettant aux usagers de les introduire facilement en machine et sans passer par le système graphique proposé en III.2.3.5. qui peut s'avérer trop lourd ou trop coûteux.

Dans les tests de l'Interpréteur de GNEC, que nous avons fait, nous avons pu mesurer la nécessité d'un interface qui nous paraît la condition sine qua non de l'utilisation des GNEC.

ANNEXE 1

ANALYSE DES QUALITÉS PRÉSENTÉES PAR LES OUTILS ALGORITHMIQUES RÉPARTIS ÉTUDIÉS DANS CETTE THÈSE

Qualités requises	Outils algorimiques répartis	SIGOR	PERE-MERE	RDPI-GNEC
D'un point de vue général	Simplicité	Oui	Non	Oui
	Modularité	Non	Non	Oui
	Transportabilité	Oui	Non	Oui
	Densité sémantique	Oui	Non	Oui
	Extensibilité	Oui (par appel de procédure)	Oui	Oui
	Expression du parallélisme	Oui	Oui	Oui
	Possibilité de pré-analyse	Non	Non	Oui
Pour l'utilisateur BDR	Transparence	Dépend du volume global de logiciel toléré		Plus facile
Pour l'Administrateur BDR	Expression des contraintes	Complexe si ce sont des contraintes réparties à insérer		Facile
	Réversibilité	Non	Non	Oui
	Tolérance aux pannes-optimisation	Difficile à gérer		Si l'Administrateur de la BDR l'a prévue

ANNEXE 2

GRAPHES DABG

Créés à l'UCLA (University of California - Los Angelès), ces graphes DABG (Directed Acyclic Bilogic Graphs) permettent également une formalisation du calcul parallèle. Nous reprenons les définitions données dans [BABOES] et [MAZARE].

Ils sont composés de noeuds (correspondant aux opérations) et d'arcs (flot de données ou de contrôle). Tout graphe possède un noeud de départ et un noeud de fin et n'a aucun cycle.

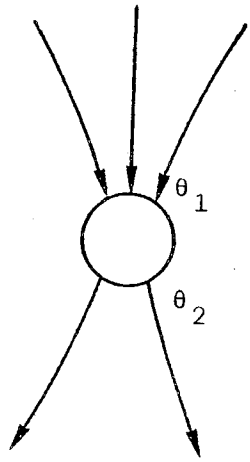
A chaque noeud est associé un couple de symbole donnant la logique (ET, OU exclusif) associée aux entrées-sorties de ce noeud, le premier élément correspondant à l'entrée et le deuxième à la sortie.

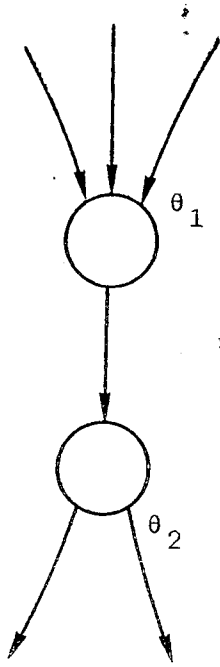
On peut donc trouver les couples suivants :

Entrée	Sortie	Couple	Schéma
OU	OU	(+, +)	
OU	ET	(+, *)	
ET	OU	(*, +)	
ET	ET	(*, *)	

Si l'on reprend le principe de marquage, comme dans les Réseaux de Petri, un noeud (+, +) enverra une marque sur l'une de ses sorties si une marque arrive sur l'une de ses entrées ; par contre, un noeud (*, +) "exigera" une marque sur chacune de ses entrées, et ainsi de suite ....

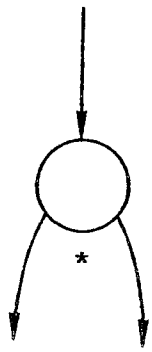
On peut remarquer que tout noeud peut être transformé en deux noeuds de la manière suivante :



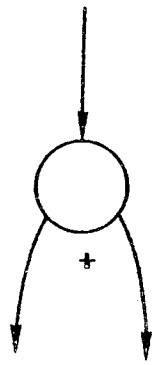


$\forall i, \theta_i \in \{*,+\}$

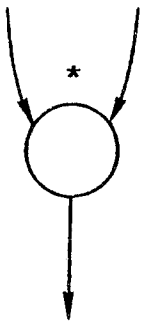
Quelques remarques rapides s'imposent :



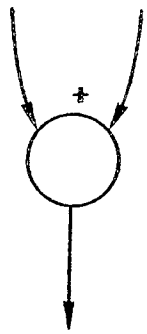
<⇒ "FORK" ou lancement en parallèle [39]



<⇒ Condition (SI ... ALORS ...



<⇒ "JOIN" ou retour de parallélisme [39]



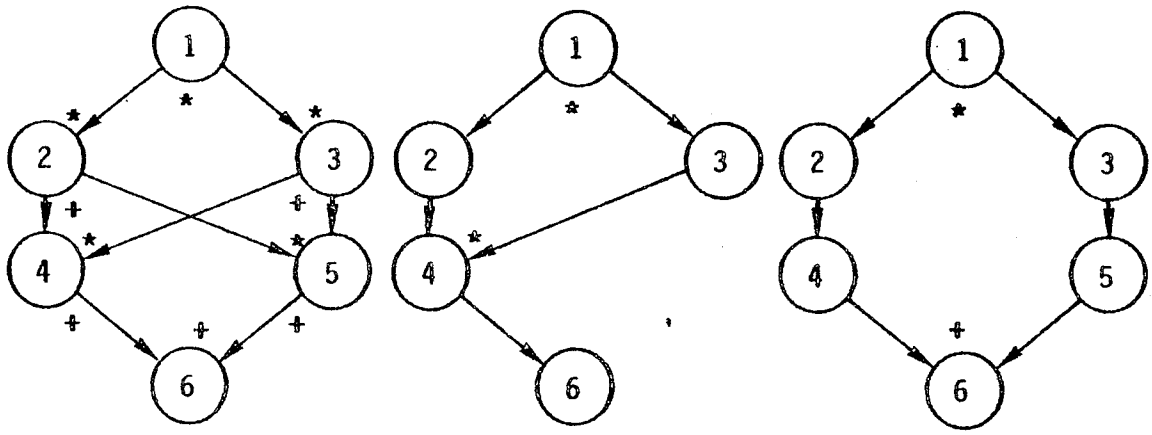
<⇒ Une et une seule des alternatives précédentes arrive (OU exclusif)

Dans de tels outils de simulation, il est important de savoir si le processus modélisé par le graphe a "un sens", c'est-à-dire si, partant du noeud de départ, on peut effectivement atteindre le noeud final, quelle que soit la décision prise dans un noeud de type (,+). Cette notion est appelée *légalité* du graphe ; un graphe G est dit *légal* si dans toute simulation de G on obtient un "*AND-Type-Subgraph*".

Un AND-Type-Subgraph peut être construit de la manière suivante :

- noeud de départ et noeud final en font partie,
- si un noeud (,*) est dans le sous-graphe, tous ses arcs sortants y sont,
- si un noeud (,+) est dans le sous-graphe, un seul arc sortant y est,
- un noeud (*,) est dans le sous-graphe si et seulement si tous ses arcs entrants y sont,
- un noeud (+,) est dans le sous-graphe si et seulement si un et un seul de ses arcs entrants y est,
- lorsque tous les noeuds ont été analysés, on supprime les arcs n'ayant qu'une extrémité dans le sous-graphe.

Nous reprenons ici un exemple donné par [MAZARE].



Directed Acyclic Bilogic Graph DABG

And type subgraph correct

Sous-graphe qui n'est pas un And Type Subgraph

Figure A2.1 - Exemple de DAB-Graphs

BIBLIOGRAPHIE

[ADCAE1] M. ADIBA, J.Y. CALECA, C. EUZET
"A distributed data base system using logical relational machines".
VLDB-78, Berlin.

[ADCAE2] M. ADIBA, J.Y. CALECA, C. EUZET
"Un système général de bases de données réparties comme un réseau de machines relationnelles".
AFCET-78.

[ADIBA] M. ADIBA
"Un modèle relationnel et une architecture pour les systèmes de base de données réparties. Application au Projet POLYPHEME".
Thèse d'Etat, Université de Grenoble, Septembre 1978.

[ADIBCO] M. ADIBA, J.C. CHUPIN, R. DEMOLOMBE, G. GARDARIN, J. LE BIHAN
"Issues in distributed data base management systems : a technical overview".
4th International Conference on VLDB, West Berlin, 13-15 Septembre 1978.

[ADICAL] M. ADIBA, J.Y. CALECA
"Modèle relationnel de données réparties. Problèmes de décomposition".
Conférence sur les modèles relationnels.
Institut de Programmation, Paris, Mars 1978.

[ADIDEL] M. ADIBA, C. DELOBEL
"Les modèles relationnels de Bases de Données".
Ecole IRIA, Avril 1976.

[AGEFLY] T. AGERWALA, M. FLYNN
1973 - First Annual Symp. on Comp. Architecture Fla.
Comments on capabilities, limitations and correctness of Petri Nets.

[ANDRAD] J.M. ANDRADE
"LAMB-V2. Spécifications générales".
NT 26, Projet Polyphème, Grenoble, 1978.

[ANDRE] E. ANDRE
"PERE - Protocole d'exécution répartie".
NT 28, Polyphème, Grenoble.

[BABOES] J.L. BAER, D.P. BOVET, G. ESTRIN
"Legality and other properties of graph models of computations".
Journal of the Association for Computing Machinery.
Vol. 17, n° 3, July 1970.

[BERGLU] R.G. BERGLUND
"Comparing Network Architectures".
Datamation, Février 1978.

[CALECA] J.Y. CALECA
"L'expression et la décomposition de transaction dans un système de bases de données réparties".
Thèse de 3ème Cycle, Université de Grenoble, Septembre 1978.

[CALSTI] J.Y. CALECA, A. STIERS
"Interface relationnel Polyphème, SGBD Socrate".
NT 01, Projet Polyphème, Grenoble 1977.

[CAPMOA] C. CAPLAIN, M. MOALLA
"Spécification d'un algorithme amélioré pour la simulation des RDPI".
Note interne, Mars 1978.

[CHUPIN] J.C. CHUPIN
"Répartition d'applications et de bases de données sur un réseau général d'ordinateurs".
Thèse d'Etat, Université de Grenoble, Octobre 1977.

[CIHOBU] CII-HONEYWELL-BULL
"VCAM sous SIRIS-8. Version C10. Manuel d'utilisation".
1978.

[CODD] E. CODD
"A relational model of data for large shared data banks".
CACM 13, 6, Juin 1970.

[COMMCO] FCOMMONER, A.W. HOLT, S. EVEN, A. PNUELI
"Marked directed graphs".
Journal of Computer and system sciences 5-71.

[DANSER] Nguyen X. DANG, G. SERGEANT
"Etude d'une machine réseau logique : SIGOR, Système d'Interprétation Général Orienté Réseau".
Rapport de Recherche n° 78, ENSIMAG, Juin 1977.

[DATE] C.J. DATE
"An introduction to data base systems".
Addison Wesley.

[DAVIES] C.T. DAVIES Jr.
"A recovery/integrity architecture for data system".
TR 02-525, IBM San José, 19 Mai 1972.

[DEC781] P. DECITRE
"Spécifications de définition de MERE".
NT 31, Projet Polyphème, Grenoble 1978.

[DEC782] P. DECITRE
"Spécifications de réalisation de MERE".
NT , Projet Polyphème, Grenoble 1978.

[ELZI73] M. ELIE, M. ZIMMERMANN
"Vers une approche systématique des protocoles sur un réseau d'ordinateurs. Application au réseau CYCLADES".
Juillet 1973.

[EUZENT] C. EUZET
"Compilation de notions en vue d'une Initiation Réseaux".
NT 2, Polyphème, Grenoble.

[EUZE78] C. EUZET
"Proposition de codification de graphes de décomposition".
NT 27, Projet Polyphème, Grenoble 1978.

[GALY] H. GALY
"Problèmes de stratégies d'ouvertures d'Interactions dans un Moniteur d'Exécution REpartie".
Rapport de DEA, INPG, Grenoble 1978.

[GARDAR] G. GARDARIN
"Résolution des conflits d'accès simultanés à un ensemble d'informations. Application aux BDR".
Thèse d'Etat, Université Pierre & Marie Curie, Paris, Avril 1978

[GOOROT] N. GOODMAN, J.B. ROTHNIE
"A survey of research and development in distributed data base management".
International Conference on Very Large Data Base, Tokyo, Octobre 1977.

[HOLCOM] A.W. HOLT, F. COMMONER
"Events and conditions".
Record of the project MAC.
Conference on concurrent systems and parallel computation.
ACM, New York, 1970.

[LITWIN] W. LITWIN
"Construction d'un SGBDR : Approche SIRIUS, Modèle de référence"
Atelier Bases de Données Réparties, AFCET 1978.

[MAZARE] G. MAZARE
"Etude des modèles théoriques de calcul parallèle".
Rapport interne.

[MOPUSI] M. MOALLA, J. PULOU, J. SIFAKIS
"Réseaux de Petri synchronisés".
Rapport de Recherche N° 80, ENSIMAG, 1977.

[MOSISI] M. MOALLA, J. SIFAKIS, M. SILVA
"A la recherche d'une méthodologie de conception sûre des automatismes logiques basée sur l'utilisation des Réseaux de Petri".
RR N° 138, Laboratoire IMAG, Octobre 1978.

[NEUBIL] E.J. NEUHOLD, H. BILLER
"Distributed data bases on a network of mini-computers".
Journées de BDR, Paris, 18-19 Mars 1977.

[NGUYEN] NGUYEN GIA TOAN
"A unified method for query decomposition and shared information updating in distributed systems".
1st International Conf. on Distributed Computing Systems.
Huntsville (Alabama), October 1979.

[NGUY77] NGUYEN GIA TOAN
"URANUS, Une approche relationnelle à la coopération de bases de données".
Thèse de 3ème Cycle, USMG, Décembre 1977.

[PAIK] In-Sup PAIK
"Définition et réalisation d'un ensemble d'opérateurs relationnels pour une machine Polyphème".
Rapport de DEA, USMG, 1978.

[PETERS] J.L. PETERSON
"Petri nets".
Computing surveys, Vol. 9, n° 3, Septembre 1977.

[POUZIN] L. POUZIN
"Présentation du réseau CYCLADES" aux membres de la Commission Interministérielle de l'Informatique, Novembre 1973.

[ROGOBE] J.B. ROTHNIE, N. GOODMAN, P.A. BERNSTEIN.
"The redundant update methodology of SDD.1. A system for distributed databases (the fully redundant case)".
TR-CCA-77.02, June 15, 1977.

[ROTGOO] J.B. ROTHNIE, N. GOODMAN
"An overview of the preliminary design of SDD-1 : a system for distributed databases".
TR-CCA-77.04, March 31, 1977.

[SESEWI] J. SEGUIN, G. SERGEANT, P. WILMS
"Cohérence et Gestion d'objets dupliqués dans les systèmes distribués".
Rapport de Recherche, Grenoble, 1978.

[STONEB] M. STONEBRAKER
"Implementation of integrity constraints and views by query modification".
ERLM514, University of Berkeley, Californie, Mars 1975.

[TSIBER] D.C. TSICHRITZIS, P.A. BERNSTEIN
"Computer science and applied mathematics".
Appendix II. Computational structures.

[VALETT] R. VALETTE
"Sur la description, l'analyse et la validation des systèmes de commandes parallèles".
Thèse d'Etat, Université Paul Sabatier, Toulouse.

[WILMS] P. WILMS
"Etude d'algorithmes de cohérence d'informations dupliquées et réparties. Formalisation à l'aide de Réseaux de Nutt".
RR N° 160, Laboratoire IMAG, Février 1979.

dernière page de la thèse

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 Avril 1974,

VU le rapport de présentation de Monsieur :

- C. DELOBEL, Professeur à l'Université Scientifique et Médicale de GRENOBLE

Monsieur Christian EUZET

est autorisé à présenter une thèse en soutenance pour l'obtention du titre de DOCTEUR de TROISIEME CYCLE, spécialité "Génie Informatique".

Grenoble, le 13 Septembre 1979

Le Président de l'I.N.P.G.

Ph. TRAYNARD
Président
de l'Institut National Polytechnique