SCORPION : SYSTEME DE MODELISATION ET D'OPTIMISATION

D. DELPUECH	Département Informatique
A. GIREs	Institut de Recherche Economique et de Planification
B. PERE-LAPERNE	Université des Sciences Sociales de Grenoble
M. SOUBIES	B.P. 47 Centre de tri 38040 Grenoble-Cédex FRANCE.

RESUME

Le système SCORPION est un outil informatique qui permet de résoudre des problèmes d'optimisation et de simulation sur des modèles dynamiques représentables sous la forme de graphes orientés.

Un utilisateur définit son problème au moyen d'un programme écrit en langage SCORPION. L'activation (session) comprend trois phases :

La phase de modélisation qui permet :

- de construire le graphe orienté schématisant le modèle,
- d'introduire les données associées au modèle ; ces données sont attachées aux noeuds et arcs du graphe, elles dépendent ou non du temps (modèle dynamique ou statique).
- de générer un ensemble d'équations qui traduisent les relations du modèle (ensemble de contraintes, fonction objectif).
- d'afficher le graphe et les données associées, partiellement ou entièrement.

La phase d'optimisation permet d'effectuer l'optimisation au moyen d'un algorithme mathématique qui donne des valeurs aux inconnues - stratégiques et de flux - telles que la fonction objectif soit optimale sur l'espace défini par les contraintes.

La phase d'analyse permet à l'utilisateur d'exploiter les résultats de l'optimisation dans l'ordre et au niveau d'agrégations qui lui conviennent.

Le système SCORPION présente trois aspects essentiels :

- La possibilité d'enchaîner plusieurs sessions : les résultats, l'ensemble des données, la structure du graphe étant conservée d'une session à l'autre.
- La possibilité de l'utiliser soit en "batch" soit en conversationnel.
- La possibilité d'étendre le langage de référence.

MOTS-CLES : Modélisation, simulation, graphe, optimisation, expressions formelles, extensions syntaxiques et sémantiques, base de données.

LES CONCEPTS DE MODELISATION, D'OPTIMISATION ET D'ANALYSE .

La phase de modélisation a pour but de permettre à l'utilisateur de décrire le modèle de son application. Ceci implique en général :

- définir un domaine - structure et données - sur lequel il veut travailler (domaine économique, d'entreprise, des réseaux, etc...).
- définir une politique de fonctionnement et de développement ou de façon générale une politique qu'il veut appliquer à ce domaine.

Ces deux définitions caractérisent le scénario propre à l'application. Une succession de scénarios consistera en :

- une modification du domaine : ceci se traduisant par un changement de la structure du graphe ou des données,
- une définition de politiques différentes.

Dans le cas où le domaine étudié est représentable sous forme de graphe - c'est-à-dire qu'il se définit comme un ensemble d'opérations (les noeuds du graphe) et de relations entre ces opérations:les transferts (les arcs du graphe)- l'utilisateur dans un premier temps aura à décrire ce graphe, ses différents composants.

Dans un deuxième temps :

- il associera aux opérations et transferts des données : les attributs, qui les caractérisent, et leur affectera une valeur. Les attributs peuvent être définis de façon externe au graphe, c'est-à-dire indépendamment des opérations ou des transferts ;
- il exprimera par les "macro-attributs" les relations qui existent entre :
 - . les inconnues implicites du modèle (les flux) ou les inconnues stratégiques (toute autre inconnue du modèle), et,
 - . les données, les attributs qu'il a défini.

Dans un troisième temps :

A l'aide des macros-attributs, des attributs et des inconnues il traduira la politique qu'il a retenu pour le modèle et l'exprimera par des contraintes et une fonction objectif à optimiser.

I.1. DEFINITION DU GRAPHE.

I.1.1. Les opérations.

Notion d'opération et mode de fonctionnement.

Une "opération" est une entité mettant en jeu un équipement qu'elle utilise selon un ou plusieurs modes de fonctionnement.

Un "mode de fonctionnement" est la manière dont l'équipement associé à une opération donnée est utilisé afin de produire un ou plusieurs "biens" (ou services) en sortie à partir d'un ou plusieurs "biens" admis en entrée.

A un instant donné l'équipement est utilisé selon un ~~seul~~ mode de fonctionnement. Dans le cas où l'opération possède plusieurs modes de fonctionnements ceux-ci seront mis en jeu alternativement durant une période donnée.

Opération monomode.

C'est une opération qui ne possède qu'un seul mode de fonctionnement, celui-ci admettant :

en entrée : zéro, un ou plusieurs biens .

en sortie : zéro, un ou plusieurs biens .

Nous la représenterons (cas général) :

Les relations entre les biens en entrée et ceux en sortie de l'opération sont décrites par l'équation :

$$be_1 + be_2 + \dots + be_n \longrightarrow bs_1 + bs_2 + \dots + bs_m$$

et les relations entre les flux (quantité de biens qui circulent dans les transferts) sont décrites par le système suivant :

$$\text{Flux } (bs_1) = a_{11} \text{ flux } (b_1) + a_{12} \text{ flux } (be_2) + \dots + a_{1n} \text{ flux } (be_n)$$

...

...

$$\text{flux } (bs_m) = a_{m1} \text{ flux } (be_1) + a_{m2} \text{ flux } (be_2) + \dots + a_{mn} \text{ flux } (be_n)$$

Opération multimode .

C'est une opération susceptible d'agir selon plusieurs modes de fonctionnement, chaque mode de fonctionnement étant homogène à une opération monomode.

Nous la représentons (cas général):

Les relations entre les biens sont décrites par un ensemble d'équations (une par mode de fonctionnement) et les relations entre les flux sont décrites par un ensemble de systèmes d'équations.

1.2. Les transferts .

- Notion de transfert .

Un transfert exprime l'existence d'une relation entre les biens "sortant" d'une opération et les biens "entrant" dans une autre. Il traduit la possibilité d'utiliser les biens en sortie de la première par la deuxième opération.

- Transfert élémentaire .

Un transfert élémentaire traduit la possibilité d'utiliser un bien d'un mode de fonctionnement d'une opération par un mode de fonctionnement d'une autre opération.

Ceci s'exprime en caractérisant l'origine et l'extrémité du transfert.

$$b. M_1. O_1 \rightarrow M_1. O_2$$

- Transfert multiple .

Un transfert multiple traduira la possibilité d'utiliser tous les biens issus d'un mode de fonctionnement d'une opération par un autre mode de fonctionnement d'une autre opération. Ceci s'exprime en caractérisant l'origine et l'extrémité du transfert. L'origine et l'extrémité sont :

- . un mode de fonctionnement d'une opération,
- . ou une opération.

Dans ce cas tous les transferts réalisables sont créés.

Dans le cas du schéma précédent où les opérations sont décrites par :

$$\begin{array}{lll}
 \text{pour O1 - M1} & : & b_0 \longrightarrow b_1 + b_2 \\
 & \text{M2} & : & b_0 \longrightarrow b_1 + b_4 \\
 \text{pour O2 - M1} & : & b_1 \longrightarrow b_5 \\
 & \text{M2} & : & b_1 + b_2 \longrightarrow b_6 \\
 & \text{M3} & : & b_3 \longrightarrow b_7
 \end{array}$$

le transfert multiple $O1 \rightarrow O2$ sera équivalent à cinq transferts élémentaires.

I.1.3. Les secteurs.

- Notion de secteur.

Un secteur est un ensemble d'opérations et/ou de transferts élémentaires du graphe complet. Il est caractérisé par un nom. Cet ensemble est en général défini par référence à des caractéristiques communes que possèdent différents éléments. L'utilisateur peut composer un secteur, d'opérations et de transferts élémentaires quelconques.

- Trois types de secteur sont définis implicitement.

- . Le secteur TRANSF : ensemble des transferts élémentaires du graphe.
- . Le secteur OPER : ensemble des opérations du graphe.
- . L'opération est définie comme un secteur composé d'un seul élément:elle-même.

I.2. DEFINITION DES DONNEES .

I.2.1. La période .

La définition de modèles dynamiques pour lesquels les données dépendent du temps exige que l'utilisateur spécifie la période sur laquelle il veut décrire son modèle. La période sera précisée par les deux dates limites de validité du modèle.

I.2.2. Les attributs ,

Un attribut est une caractéristique élémentaire, dépendante ou non du temps, qui peut être associée :

- au mode de fonctionnement d'une opération du graphe,
- a un transfert élémentaire du graphe,
- de façon externe au graphe.

Dans les deux premiers cas ils seront dits localisés, dans le dernier cas non localisés.

L'utilisation de ces attributs apparaît lors de leur déclaration et de leur affectation.

- Déclaration ,

C'est à ce moment que l'utilisateur définit l'ensemble des attributs qu'il utilisera pour le modèle qu'il décrit. Il spécifie s'il s'agit d'attributs qui sont attachés aux éléments du graphe (attribut localisé) ou non (attribut non localisé).

Il précise si ceux-ci peuvent prendre différentes valeurs dans le temps ou non.

- Affectation ,

L'affectation d'un attribut localisé est sa réalisation pour un élément du graphe(mode de fonctionnement d'une opération ou transfert élémentaire) et la définition d'une valeur pour celui-ci.

L'affectation d'un attribut non localisé sera la définition d'une valeur pour celui-ci.

La valeur d'un attribut peut être définie par une expression faisant référence à des attributs déjà définis.

La définition peut être formelle : $A = B * C$, si la valeur de B ou C est modifiée après cette définition la valeur de A sera aussi modifiée,

ou numérique: $A = ? B * C$ prend la valeur $B * C$ lors de la définition, des modifications ultérieures de B et C n'affecteront pas A.

I.2.3. Les inconnues.

Le système permet de prendre en compte deux catégories d'inconnues :

- Les "inconnues implicites" liées à la structure du modèle (graphe).
- Les "inconnues stratégiques" qui comme les attributs peuvent être localisés en des éléments du graphe ou non. Elles représentent toutes les autres inconnues du système.
- "Inconnues implicites":

Les inconnues implicites sont les inconnues associées au flux qui circulent dans le graphe.

- Inconnues ARC : ce sont les inconnues flux mesurant la qualité de bien qui circule dans un transfert élémentaire.
- Inconnues CHEMIN : ce sont les inconnues flux mesurant la quantité qui circule d'un mode de fonctionnement d'une opération initiale à un mode de fonctionnement d'une opération finale.

(Un mode de fonctionnement d'une opération initiale est un mode de fonctionnement qui n'a pas de bien en entrée.

Un mode de fonctionnement d'une opération finale est un mode de fonctionnement qui n'a pas de bien en sortie sauf dans le cas où le graphe possède des cycles où ce sera un mode de fonctionnement d'une opération quelconque).

L'utilisateur dispose de ces deux types d'inconnues implicites.

Dans le premier cas une inconnue est associée à chaque transfert élémentaire du graphe, dans le deuxième cas à chaque chemin du graphe et ceci pour les les années "repère".

La notion "repère" permet d'indiquer les années de la période pour lesquelles une inconnue est définie, (sauf indication explicite d'un sous-ensemble

d'années à considérer comme années repère. Le système considère par défaut toutes les années de la période comme des repères).

Le choix du type d'inconnues dépend de la nature du problème à résoudre. Les inconnues chemin fournissent des éléments d'analyse plus complets mais sont plus nombreuses.

A partir de ces notions élémentaires d'inconnues sont définies les "inconnues agrégées" qui sont également des notions du système.

(Voir l'annexe pour l'ensemble des inconnues agrégées qui peuvent être utilisées).

"Inconnues stratégiques" :

Ces inconnues peuvent être localisées ou non, datées ou pas.

Leur utilisation impose leur déclaration et leur réalisation.

- Déclaration :

La déclaration consiste à indiquer le type de l'inconnue stratégique : localisée ou non et si celle-ci dépend du temps ou non.

- Réalisation :

La réalisation d'une inconnue stratégique localisée consiste en la spécification des éléments auxquels on souhaite associer cette inconnue.

Si l'inconnue stratégique a été déclarée comme dépendant du temps sa réalisation ne sera effective que pour les années repère.

Dans ce cas il y aura pour l'élément considéré autant d'inconnues que d'années repère.

1.2.4. Les macros-attributs .

Un macro-attribut est une notion définie comme une expression quelconque des ATTRIBUTS et/ou des INCONNUES avec la mention d'au moins une inconnue dans l'expression.

Par exemple on définira :

- le coût-de-fonctionnement comme le produit d'un coût-unitaire-de-fonctionnement (attribut) par le flux entrant dans une opération (inconnue implicite) ;
- le coût-d'-investissement comme le produit d'un coût-unitaire-d'-investissement (attribut) par l'accroissement de capacité de l'opération (inconnue stratégique) ;
- le macro-attribut-Z comme égal a : $\text{attribut-a/inc-strat-x} + \text{attribut-b} \cdot \text{LOG (FS-d'opération)}$.

Un macro-attribut peut être localisé ou non et peut dépendre du temps ou non.

Pour l'utiliser il faudra le déclarer puis l'affecter.

Déclaration :

La déclaration consiste à préciser le type de l'attribut.

Affectation : L'affectation consiste :

- dans le cas où le macro-attribut est localisé à spécifier les éléments du graphe (mode de fonctionnement d'opération, transfert élémentaire) où il existe et à donner sa définition par une expression des attributs et des inconnues ;
- dans le cas où il n'est pas localisé à donner sa définition.

I.3. DEFINITION D'UNE POLITIQUE - GENERATION D'EQUATIONS

La structure du domaine, les données ayant été définies l'utilisateur peut désormais traduire :

- les contraintes techniques du domaine global,
- la politique qu'il désire appliquer à celui-ci.

Cette politique s'exprime par un ensemble de contraintes et une fonction objectif à optimiser.

I.3.1. Les contraintes .

Une contrainte est une relation entre deux expressions composées d'attributs, d'inconnues, de macro-attributs, de nombres (avec la mention d'au moins une inconnue ou macro-attribut).

I.3.2. La fonction objectif .

La fonction objectif sera définie comme une expression des attributs, des inconnues, des macro-attributs et de nombres à optimiser.

I.3.3. Génération d'équation .

A partir de la forme externe des contraintes et de la fonction objectif qu'aura défini l'utilisateur, le système générera un ensemble d'équations,

ou les attributs prendront les valeurs affectées lors de leur définition,
ou les inconnues agrégées seront remplacées par les inconnues élémentaires,
ou les macros-attributs auront été remplacés par leur expression.

C'est cet ensemble d'équations, qui peut s'exprimer sous la forme d'un programme d'optimisation,

min (ou max) $g(x_{ie}, x_s)$

sur le domaine définit par

$f_j(x_{ie}, x_s) R_{hj}(x_{ie}, x_s)$ pour $j = 1 \text{ à } n$

(ou x_{ie} représente les inconnues implicites élémentaires.

x_s représente les inconnues stratégiques .

R est un opérateur de relation : $<, =, >, <=, >=$.

n est le nombre de contraintes définies par l'utilisateur),
qui sera résolu par un algorithme d'optimisation.

4. ALGORITHMES - OPTIMISATION .

Le modèle ayant été décrit - par la définition de sa structure des données, d'une politique - l'ensemble des données, des contraintes, de la fonction objectif est transformé en un système d'équations dépendant des inconnues élémentaires qui se présente sous la forme :

$$\left\{ \begin{array}{l} \text{Min } g(x_{ie}, x_s) \\ f_j(x_{ie}, x_s) \leq h_j(x_{ie}, x_s) \text{ pour } j = 1 \text{ à } n. \end{array} \right.$$

Ce problème sera traité par un algorithme d'optimisation.

Dans l'état actuel du système SCORPION, en raison de la meilleure connaissance des algorithmes d'optimisation linéaires qui permettent de traiter un très grand nombre d'inconnues et de contraintes, le problème doit se traduire en terme linéaire c'est-à-dire que les fonctions f_j , h_j et g doivent être linéaires en x_{ie} et x_s .

Dans ces conditions, une solution sera obtenue par un algorithme de simplexe qui fournira comme résultat essentiel, les différentes valeurs des inconnues et les valeurs duales des contraintes.

Toutefois, tous les concepts développés permettent de générer des équations non linéaires des inconnues, une parfaite connaissance et une généralisation des algorithmes non linéaires permettraient alors de résoudre tout problème d'optimisation.

L'aspect ~~génération d'équations~~ essentiel dans le système SCORPION peut également convenir à la résolution de modèle de simulation se définissant à l'aide de fonctions des inconnues, seuls les algorithmes sont à adapter.

I.5. ANALYSE DES RESULTATS .

Bien souvent, l'analyse des résultats est considérablement ralentie par le volume énorme des valeurs numériques à compiler. La possibilité d'utiliser le système de façon conversationnelle pour rechercher les informations est précieuse lors de cette phase.

Cette partie est la plus ouverte et l'extensibilité du système prend ici tout son sens dans la mesure où il est impossible de prévoir à l'avance tous les besoins. L'utilisateur par le moyen de macro-commandes peut créer de nouveaux outils d'analyse qui lui sont propres.

Néanmoins, certains outils d'analyse sont prévus.

L'algorithme d'optimisation fournit toutes les valeurs des inconnues élémentaires l'utilisateur peut obtenir :

- l'ensemble des inconnues agrégées.
- l'évaluation de tous les macros-attributs.
- L'évaluation de toute expression des inconnues, des attributs, des macros-attributs, des valeurs duales, des nombres.

D'autres possibilités telles que l'analyse statistique des résultats ou la sortie de ceux-ci sous forme de tableaux sont envisagées, mais ne sont pas actuellement réalisées.

II - LE LANGAGE LANSKO .

II.1. Présentation du langage .

L'interface entre l'utilisateur et le système SCORPION est assuré par le langage LANSKO qui permet à l'utilisateur :

- l'initialisation et la sauvegarde de ces données en début et en fin de session.
- la modification du langage LANSKO lui-même.
- L'interprétation de commandes par rapport à l'état courant du langage.

Un aspect important réside dans les possibilités de modifications inhérentes au langage. Ces possibilités sont au nombre de deux.

- définition (ou suppression) de macrosyntaxe.
- définition (ou suppression) de macrocommande.

La première fait intervenir des notions de la théorie des langages ; elle sera très rarement mise en oeuvre par un utilisateur mais est plutôt destinée aux personnes chargées d'adapter le système SCORPION à un environnement donné.

La deuxième possibilité offre, au contraire, l'avantage classique de pouvoir définir des ellipses du langage de base.

L'utilisateur peut être amené à définir sa propre syntaxe par équivalences à la syntaxe de base, celle-ci restant inchangée.

Ces deux mécanismes d'extension du langage implique la notion de dialecte. A un instant donné d'une session le dialecte est constitué du langage de base ainsi que des macrosyntaxes et des macrocommandes définies par l'utilisateur ; seules les commandes écrites dans le dialecte courant peuvent être reconnues et interprétées à un moment donné.

D'un point de vue interne au système, la technique retenue pour traiter les extensions du langage de base est celle de la génération d'un code sous forme d'automate à pile d'états-finis. La totalité du langage de base a été définie sous forme de macrosyntaxes. Le langage de base est donc codifié dans la syntaxe propre au générateur d'automate.

I.2. Les expressions.

Les expressions, au sens classique d'expressions arithmétiques jouent un rôle fondamental dans le système SCORPION.

En effet, à part la structure d'un graphe, l'ensemble des données, organisées en base de données, est constitué d'un ensemble de notions (attributs, inconnues, macro-attributs...) et des valeurs numériques ou des expressions formelles de ces notions.

Dans la phase d'optimisation, l'écriture de la fonction objectif et des contraintes consiste à traiter des expressions pour générer des équations ; de même dans la phase d'analyse, il s'agit d'évaluer des expressions pour obtenir des résultats numériques.

D'un point de vue syntaxique, une expression est une combinaison algébrique tout à fait classique d'opérandes (les notions et/ou des nombres) et d'opérateurs.

Une notion peut être définie directement par une valeur numérique ou par une expression numérique :

ex. $a = 5$ ← expressions de type numérique
 $a = 2 * 10$

ou bien par référence à d'autres notions :

ex. $c = ? (a + b) / 2$ ← expression de type numérique
 $d = 2 * (b + f)$ ← expression de type formel

Le traitement d'une expression diffère suivant son type numérique ou formel ; en effet, celle de type numérique est reconnue puis évaluée, seul le résultat étant conservé tandis que celle de type formel est reconnue, puis conservée sous une forme particulière pour être évaluée dans une phase ultérieure.

La reconnaissance d'une expression consiste :

- à l'exprimer sous forme postfixée,
- à générer le pseudo-correspondant.

Dans le cas d'une expression formelle, c'est le pseudo-code qui est conservé et rattaché à la notion qu'il définit.

II.3. Les S et T itérations.

Les S-itérations sont des itérations sur l'espace liées au graphe et les T-itérations sont des itérations liées au temps dans les modèles dynamiques. Les itérations apparaissent comme des préfixes - éventuellement vides - de certaines instructions du langage.

Une S-itération a pour signification la répétition pour un certain nombre d'opérations et/ou de transferts (liste de lieux) regroupés ou non en secteurs d'une même instruction de définition ou d'évaluation.

Ex. $\text{pours } op_1, op_2, \dots, op_n : a = b/c ;$

 préfixe d'une S-itération : instruction de définition.

? $\text{pours } op_1, op_2, \dots, op_n : a ;$

 préfixe d'une S-itération : instruction d'évaluation.
 évaluation.

de même, une T-itération a pour signification la répétition suivant une série de dates d'une instruction de définition ou d'évaluation.

ex : $\text{pour } t = d_1, d_2, \dots, d_m : f = (1 + a) \times (t - d_1) ;$

 préfixe d'une T-itération : instruction de définition.

Une S-itération et une T-itération peuvent être imbriquées :

ex. $\text{pours } op_1, op_2, \dots, op_n :$
 $\text{pour } t = d_1 \text{ à } d_m : f(t) = (1 + a) \times (t - d_1) ;$

Une autre forme de S-itération et de T-itération apparaît lors de l'utilisation des macro-opérateurs (somme, produit...) qui permettent d'évaluer une expression qui porte sur une liste de lieux et/ou sur une série de dates.

ex. $\text{SOM } (op_1, op_2, \dots, op_n : \text{SOM } (t = d_1 \text{ à } d_m : 1/f(t))) ;$

 macro-opérateur, S-somme macro-opérateur, T-somme

III - ARCHITECTURE GENERALE DU SYSTEME .

Le système SCORPION peut être activé suivant deux modes.

Le premier est le mode interprétatif ; c'est celui que connaît l'utilisateur lorsqu'il travaille sur un modèle et ses données à l'aide du dialecte prédéfini.

Les modules suivants sont alors activés : (voir schéma du système).

- le moniteur et le module d'entrée/sortie avec l'utilisateur,
- l'interpréteur d'automate,
- les modules-algorithme d'optimisation, fonction de gestion de la base de données, fonction de gestion des expressions formelles -qui interagissent avec la base de données grâce à la fonction d'entrée/sortie.

Le modèle, les notions entrées et celles calculées se trouvant dans la base de données.

Le deuxième mode de fonctionnement est celui de définition ou de modification du dialecte courant.

Dans ce cas, les modules suivants sont activés : (voir schéma du système)

- le moniteur et le module d'entrées/sorties avec l'utilisateur,
- le générateur d'automate qui est chargé de définir un nouvel automate à pile,
- l'interpréteur d'automate.

Architecture générale du système : schéma

Inconnue agrégée	Désignation dans le système	Expression en fonction de inconnues élémentaires ou inconnues agrégées définies	
Flux "entrant" dans un transfert élémentaire	FE de $\underline{b_1.M1.O1} \rightarrow M2.O2$ désignation du transfert	Si l'on considère des inconnues Arcs - l'inconnue Arc, sinon la somme des inconnues chemin traversant l'Arc.	
Flux "sortant" d'un bien	FS de $\underline{b_1.M1.O1}$ désignation du bien	$= \sum_j \sum_i \text{FE de } \underline{b_1.M1.O1} \rightarrow M1.Oj$	
Flux "sortant" d'un mode de fonctionnement	FS de $\underline{M1.O1}$ désignation du mode de fonctionnement	$= \sum_i \text{FS de } \underline{b_i.M1.O1}$	
Flux "sortant" d'une opération	FS de $\underline{O1}$ désignation de l'opération	$= \sum_i \text{FS de } \underline{M_i.O1}$	

I - ANNEXE Inconnues agrégées définies à partir des inconnues implicites élémentaires.

Inconnue agrégée	Désignation dans le système	Expression en fonction des inconnues élémentaires ou inconnues agrégées définies	
Flux "sortant" d'un transfert élémentaire	FS de $\underline{b_1.M_1.O_1} \rightarrow M_2.O_2$	$= \text{RDT} \star \text{FE de } \underline{B_1.M_1.O_1} \rightarrow M_2.O_2$ <div>$\nearrow$ Rendement du transfert</div>
Flux "entrant" dans un bien	FE de $M_2.O_2$	$= \sum_{j-i} \text{FS de } \underline{b_1.M_1.O_j} \rightarrow M_2.O_2$
Flux "entrant" dans un mode de fonctionnement	FE de $M_2.O_2$	$= \sum_i \text{FE de } \underline{b_i.M_2.O_2}$
Flux "entrant" dans une opération	FE de O_2	$= \sum_i \text{FE de } \underline{M_1.O_2}$

II - ANNEXE - Inconnues agrégées définies à partir des inconnues implicites élémentaires.