EAQUE-LRO : génération de systèmes experts : application à des problèmes d'ordonnancement

Christophe Roche

THESE

présentée à

l' Institut National Polytechnique de Grenoble

par

Christophe ROCHE

EAQUE - LRO
GENERATION DE SYSTEMES EXPERTS
APPLICATION A DES PROBLEMES D'ORDONNANCEMENT

Thèse soutenue le 4 juillet 1984 devant la Commission d'Examen :

Monsieur	L. BOLLIET	:	Pésident
Messieurs	P. JORRAND	}	Examinateurs
	J.P. LAURENT		
	J.C. MANGIN		

A Patricia.

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1982-1983

Président de l'Université : D. BLOCH

Vice-Président : René CARRE
Hervé CHERADAME
Marcel IVANES

PROFESSEURS DES UNIVERSITES :

ANCEAU François	E.N.S.I.M.A.G.
BARRAUD Alain	E.N.S.I.E.G.
BAUDELET Bernard	E.N.S.I.E.G.
BESSON Jean	E.N.S.E.E.G.
BLIMAN Samuel	E.N.S.E.R.G.
BLOCH Daniel	E.N.S.I.E.G.
BOIS Philippe	E.N.S.H.G.
BONNETAIN Lucien	E.N.S.E.E.G.
BONNIER Etienne	E.N.S.E.E.G.
BOUVARD Maurice	E.N.S.H.G.
BRISSONNEAU Pierre	E.N.S.I.E.G.
BUYLE BODIN Maurice	E.N.S.E.R.G.
CAVAIGNAC Jean-François	E.N.S.I.E.G.
CHARTIER Germain	E.N.S.I.E.G.
CHENEVIER Pierre	E.N.S.E.R.G.
CHERADAME Hervé	U.E.R.M.C.P.P.
CHERUY Arlette	E.N.S.I.E.G.
CHIAVERINA Jean	U.E.R.M.C.P.P.
COHEN Joseph	E.N.S.E.R.G.
COUMES André	E.N.S.E.R.G.
DURAND Francis	E.N.S.E.E.G.
DURAND Jean-Louis	E.N.S.I.E.G.
FELICI Noël	E.N.S.I.E.G.
FOULARD Claude	E.N.S.I.E.G.
GENTIL Pierre	E.N.S.E.R.G.
GUERIN Bernard	E.N.S.E.R.G.
GUYOT Pierre	E.N.S.E.E.G.
IVANES Marcel	E.N.S.I.E.G.
JAUSSAUD Pierre	E.N.S.I.E.G.
JOUBERT Jean-Claude	E.N.S.I.E.G.
JOURDAIN Geneviève	E.N.S.I.E.G.
LACOUME Jean-Louis	E.N.S.I.E.G.
LATOMBE Jean-Claude	E.N.S.I.M.A.G.

.../...

LESSIEUR Marcel	E.N.S.H.G.
LESPINARD Georges	E.N.S.H.G.
LONGEQUEUE Jean-Pierre	E.N.S.I.E.G.
MAZARE Guy	E.N.S.I.M.A.G.
MOREAU René	E.N.S.H.G.
MORET Roger	E.N.S.I.E.G.
MOSSIERE Jacques	E.N.S.I.M.A.G.
PARIAUD Jean-Charles	E.N.S.E.E.G.
PAUTHENET René	E.N.S.I.E.G.
PERRET René	E.N.S.I.E.G.
PERRET Robert	E.N.S.I.E.G.
PIAU Jean-Michel	E.N.S.H.G.
POLOUJADOFF Michel	E.N.S.I.E.G.
POUPOT Christian	E.N.S.E.R.G.
RAMEAU Jean-Jacques	E.N.S.E.E.G.
RENAUD Maurice	U.E.R.M.C.P.P.
ROBERT André	U.E.R.M.C.P.P.
ROBERT François	E.N.S.I.M.A.G.
SABONNADIERE Jean-Claude	E.N.S.I.E.G.
SAUCIER Gabrielle	E.N.S.I.M.A.G.
SCHLENKER Claire	E.N.S.I.E.G.
SCHLENKER Michel	E.N.S.I.E.G.
SERMET Pierre	E.N.S.E.R.G.
SILVY Jacques	U.E.R.M.C.P.P.
SOHM Jean-Claude	E.N.S.E.E.G.
SOUQUET Jean-Louis	E.N.S.E.E.G.
VEILLON Gérard	E.N.S.I.M.A.G.
ZADWORNY François	E.N.S.E.R.G.

PROFESSEURS ASSOCIES

BASTIN Georges	E.N.S.H.G.
BERRIL John	E.N.S.H.G.
CARREAU Pierre	E.N.S.H.G.
GANDINI Alessandro	U.E.R.M.C.P.P.
HAYASHI Hirashi	E.N.S.I.E.G.

PROFESSEURS UNIVERSITE DES SCIENCES SOCIALES (Grenoble II)

BOLLIET Louis
Chatelin Françoise

PROFESSEURS E.N.S. Mines de Saint-Etienne

RIEU Jean
SOUSTELLE Michel

CHERCHEURS DU C.N.R.S.

FRUCHART Robert	Directeur de Recherche
VACHAUD Georges	Directeur de Recherche

.../...

ALLIBERT Michel	Maître de Recherche
ANSARA Ibrahim	Maître de Recherche
ARMAND Michel	Maître de Recherche
BINDER Gilbert	
CARRE René	Maître de Recherche
DAVID René	Maître de Recherche
DEPORTES Jacques	
DRIOLE Jean	Maître de Recherche
GIGNOUX Damien	
GIVORD Dominique	
GUELIN Pierre	
HOPFINGER Emil	Maître de Recherche
JOUD Jean-Charles	Maître de Recherche
KAMARINOS Georges	Maître de Recherche
KLEITZ Michel	Maître de Recherche
LANDAU Ioan-Dore	Maître de Recherche
LASJAUNIAS J.C.	
MERMET Jean	Maître de Recherche
MUNIER Jacques	Maître de Recherche
PIAU Monique	
PORTESEIL Jean-Louis	
THOLENCE Jean-Louis	
VERDILLON André	

CHERCHEURS du MINISTERE de la RECHERCHE et de la TECHNOLOGIRE (Directeurs et Maîtres de Recherches, ENS Mines de St. Etienne)

LESBATS Pierre	Directeur de Recherche
BISCONDI Michel	Maître de Recherche
KOBYLANSKI André	Maître de Recherche
LE COZE Jean	Maître de Recherche
LALAUZE René	Maître de Recherche
LANCELOT Francis	Maître de Recherche
THEVENOT François	Maître de Recherche
TRAN MINH Canh	Maître de Recherche

PERSONNALITES HABILITEES à DIRIGER des TRAVAUX de RECHERCHE (Décision du Conseil Scientifique)

ALLIBERT Colette	E.N.S.E.E.G.
BERNARD Claude	E.N.S.E.E.G.
BONNET Rolland	E.N.S.E.E.G.
CAILLET Marcel	E.N.S.E.E.G.
CHATILLON Catherine	E.N.S.E.E.G.
CHATILLON Christian	E.N.S.E.E.G.
COULON Michel	E.N.S.E.E.G.
DIARD Jean-Paul	E.N.S.E.E.G.
EUSTAPOPOULOS Nicolas	E.N.S.E.E.G.
FOSTER Panayotis	E.N.S.E.E.G.

.../...

GALERIE Alain	E.N.S.E.E.G.
HAMMOU Abdelkader	E.N.S.E.E.G.
MALMEJAC Yves	E.N.S.E.E.G. (CENG)
MARTIN GARIN Régina	E.N.S.E.E.G.
NGUYEN TRUONG Bernadette	E.N.S.E.E.G.
RAVAINE Denis	E.N.S.E.E.G.
SAINFORT	E.N.S.E.E.G. (CENG)
SARRAZIN Pierre	E.N.S.E.E.G.
SIMON Jean-Paul	E.N.S.E.E.G.
TOUZAIN Philippe	E.N.S.E.E.G.
URBAIN Georges	E.N.S.E.E.G. (Laboratoire des ultra-réfractaires ODEILLON)
GUILHOT Bernard	E.N.S. Mines Saint Etienne
THOMAS Gérard	E.N.S. Mines Saint Etienne
DRIVER Julien	E.N.S. Mines Saint Etienne
BARIBAUD Michel	E.N.S.E.R.G.
BOREL Joseph	E.N.S.E.R.G.
CHOVET Alain	E.N.S.E.R.G.
CHEHIKIAN Alain	E.N.S.E.R.G.
DOLMAZON Jean-Marc	E.N.S.E.R.G.
HERAULT Jeanny	E.N.S.E.R.G.
MONLLOR Christian	E.N.S.E.R.G.
BORNARD Guy	E.N.S.I.E.G.
DESCHIZEAU Pierre	E.N.S.I.E.G.
GLANGEAUD François	E.N.S.I.E.G.
KOFMAN Walter	E.N.S.I.E.G.
LEJEUNE Gérard	E.N.S.I.E.G.
MAZUER Jean	E.N.S.I.E.G.
PERARD Jacques	E.N.S.I.E.G.
REINISCH Raymond	E.N.S.I.E.G.
ALEMANY Antoine	E.N.S.H.G.
BOIS Daniel	E.N.S.H.G.
DARVE Félix	E.N.S.H.G.
MICHEL Jean-Marie	E.N.S.H.G.
OBLED Charles	E.N.S.H.G.
ROWE Alain	E.N.S.H.G.
VAUCLIN Michel	E.N.S.H.G.
WACK Bernard	E.N.S.H.G.
BERT Didier	E.N.S.I.M.A.G.
CALMET Jacques	E.N.S.I.M.A.G.
COURTIN Jacques	E.N.S.I.M.A.G.
COURTOIS Bernard	E.N.S.I.M.A.G.
DELLA DORA Jean	E.N.S.I.M.A.G.
FONLUPT Jean	E.N.S.I.M.A.G.
SIFAKIS Joseph	E.N.S.I.M.A.G.
CHARUEL Robert	U.E.R.M.C.P.P.
CADET Jean	C.E.N.G.
COEURE Philippe	C.E.N.G. (LETI)

.../...

DELHAYE Jean-Marc	C.E.N.G. (STT)
DUPUY Michel	C.E.N.G. (LETI)
JOUVE Hubert	C.E.N.G. (LETI)
NICOLAU Yvan	C.E.N.G. (LETI)
NIFENECKER Hervé	C.E.N.G.
PERROUD Paul	C.E.N.G.
PEUZIN Jean-Claude	C.E.N.G. (LETI)
TAIEB Maurice	C.E.N.G.
VINCENDON Marc	C.E.N.G.

LABORATOIRES EXTERIEURS

DEMOULIN Eric	C.N.E.T.
DEVINE	C.N.E.T. (R.A.B.)
GERBER Roland	C.N.E.T.
MERCKEL Gérard	C.N.E.T.
PAULEAU Yves	C.N.E.T.
GAUBERT C.	I.N.S.A. Lyon

Je tiens à remercier sincèrement JP.LAURENT, Professeur à l'Université de Savoie et Directeur du Laboratoire d'Informatique Appliquée. Non seulement il m'a fourni, tout au long de cette thèse, son aide et ses encouragements, mais il s'est de plus associé étroitement à mon travail. Je lui suis profondément reconnaissant de sa collaboration amicale et efficace.

Je remercie Monsieur L.BOLLIET, Professeur à l'Université de Grenoble II, qui m'a fait l'honneur de présider le jury de cette thèse.

Je remercie également Monsieur P.JORRAND, Directeur de recherches au CNRS et Responsable du Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle de l'INPG, pour sa participation au jury et pour l'intérêt qu'il a porté à mon travail.

Je remercie enfin Monsieur JC.MANGIN, Professeur à l'Université de Savoie et Directeur du Laboratoire de Génie Civil, de ses nombreux conseils pour l'application d'EAQUE aux problèmes d'ordonnancement, et d'avoir accepté de participer au jury.

.../...

Je tiens également à remercier :

Monsieur B.DENIS, à qui je dois la réalisation de cette thèse dans un contexte industriel,

le Centre de Recherche de CAP SOGETI pour avoir financé la première année de cette thèse,

le service Recherche et Développement de SYMAG pour avoir financé la deuxième année,

enfin, tous ceux qui m'ont aidé par leurs encouragements durant ces deux années. Je pense particulièrement à ma famille, et à mon épouse, Patricia, qui a fait preuve de beaucoup de patience.

EAQUE - LRO

GENERATION DE SYSTEMES EXPERTS

APPLICATION A DES PROBLEMES D'ORDONNANCEMENT

EAQUE - LRO

GENERATION DE SYSTEMES EXPERTS

APPLICATION A DES PROBLEMES D'ORDONNANCEMENT

RESUME : **LRO** (Langage de Représentation d'Objet) offre non seulement les possibilités de définir et d'utiliser des structures de données complexes, mais également un environnement de programmation de type langage orienté objet.
EAQUE est un moteur d'inférence général pour une certaine classe de domaines d'application. Il n'intègre aucune connaissance particulière à un de ces domaines, mais est caractérisé par un ensemble d'options qui leur sont communes. Ces options correspondent à des choix de mode de représentation des connaissances et de structure de contrôle. L'instantiation d'EAQUE à différents domaines d'application génère autant de systèmes experts. Ainsi **ORDF** gère le partage de ressources en nombre limité. **CALINT** simplifie des expressions mathématiques, **EALOG** simule un interpréteur Prolog.
EAQUE est écrit entièrement en LRO-LISP.

MOTS CLES : Intelligence Artificielle, Systèmes Experts, Modes de Représentation des Connaissances, Moteurs d'Inférence Généraux, Ordonnancement.

SOMMAIRE

-* CHAPITRE I *-

INTRODUCTION

-* CHAPITRE I*-

INTRODUCTION

1 - INTRODUCTION
2 - LRO
3 - EAQUE
4 - ORDF
5 - CALINT
6 - EALOG
7 - CONCLUSION

RESUME : Présentation de la thèse et résumés de chaque chapitre.

INTRODUCTION

Le travail que nous présentons a trouvé son origine dans les problèmes d'ordonnancement. Bien qu'ils soient étudiés depuis de nombreuses années, leur complexité est telle qu'ils restent ouverts. Ainsi, l'option que nous avons choisie se veut résolument nouvelle par l'utilisation des techniques de l'Intelligence Artificielle et en particulier de l'approche "Système Expert".

Ce fut également notre volonté de considérer le problème dans son ensemble et non de réaliser une application dédiée à un cas particulier. Cette difficulté supplémentaire offre cependant plusieurs avantages. En particulier cela permet de rendre externes au système les propriétés (parfois procédurales) du domaine étudié, et soulage d'autant le concepteur.

Notre travail a comporté trois phases.

La première avait pour but de nous doter d'outils suffisamment puissants et généraux, d'une part pour définir et manipuler aisément des structures de données complexes telles qu'on les rencontre dans les problèmes d'ordonnancement, d'autre part pour gérer et associer à ces données leurs procédures d'exploitation. C'est la réalisation de **LRO**, Langage de Représentation d'Objet.

La deuxième phase fut l'écriture, à l'aide principalement de LRO et du langage noyau (LISP), de **EAQUE**. EAQUE est un moteur d'inférence général pour une classe de problèmes dont font partie les problèmes d'ordonnancement. LRO est utilisé

pour décrire le problème à résoudre. Le formalisme des règles de production a été choisi pour représenter les connaissances de résolution et de contrôle. Quant au moteur proprement dit, il est de type **EOA** (choix d'état, choix d'objet, choix d'action cf. LAUR84) et utilise un sytème de pondérations heuristiques.

Enfin, la troisième phase est l'application d'EAQUE à des domaines concrets. Par exemple :
- l'ordonnancement des travaux du bâtiment dans le cas où sont connus, pour chaque tâche, sa durée ainsi que le nombre et le type de ressources qu'elle utilise. Nous obtenons ainsi le sytème **ORDF** (Ordonnancement à Ressources et Durées Fixées).
- la simplification de formules mathématiques et le calcul d'intégrales, c'est le système CALINT (CALcul d'INTégrales).
- la simulation d'un interprétcur PROLOG appliqué à une base de connaissances généalogiques.

La figure suivante montre la structure du système ORDF (la structure est identique pour les systèmes CALINT et EALOG), et situe les différentes personnes dans leur fonction, sachant que:

U représente l'Utilisateur final du système ORDF. Il lui soumet différents problèmes qu'il aura exprimés dans un langage quasi-naturel : **LN1.**

E représente l'Expert du domaine d'application. Il fournit au système ORDF les connaissances nécessaires à la résolution d'un problème de ce domaine, et au contrôle de cette résolution. Ces connaissances sont exprimées dans un langage quasi-naturel : **LN2.**
Il aide également l'Ingénieur Cognitif dans sa tâche en précisant les propriétés du domaine étudié pour que celui-ci puisse adapter EAQUE au domaine choisi (voir ci-dessous).

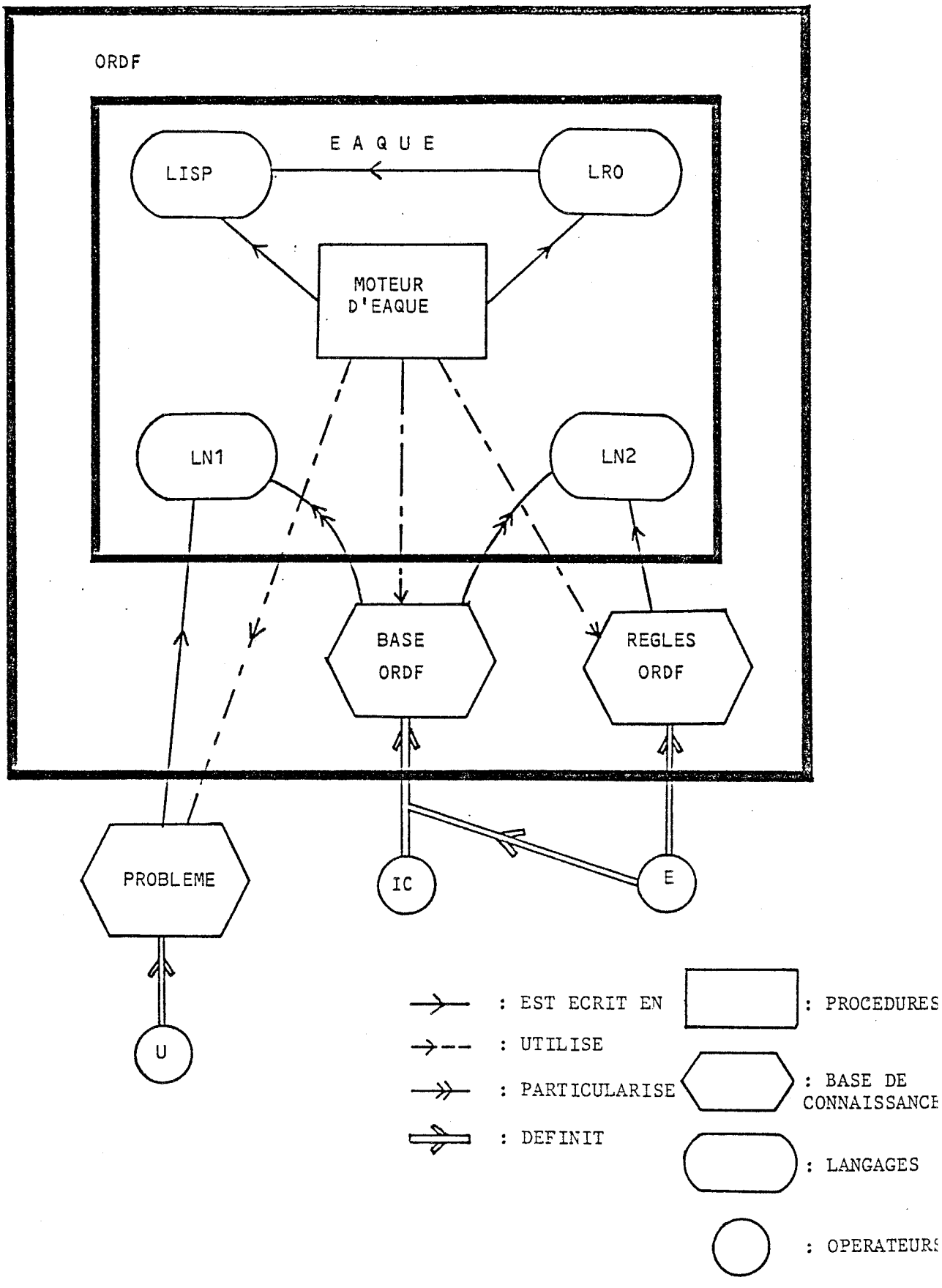
IC représente l'Ingénieur Cognitif. C'est un informaticien rompu aux techniques de l'Intelligence Artificielle et connaissant les caractéristiques de EAQUE. Il fournit à EAQUE l'ensemble des informations nécessaires pour son application au domaine considéré, qui constitue ce que nous appellerons la **BASE ORDF.**

Note :
Certains liens ne sont pas représentés par souci de clarté. Ainsi, il n'est pas indiqué que la description du problème et les règles sont des structures de données de LRO, ou que LN1 et LN2 sont écrits en LISP.

cf figure1

Les chapitres suivants présentent un résumé des trois phases que nous venons d'introduire. Elles seront détaillées dans leur section respective.

FIGURE 1


ORDF
E A Q U E
LISP
LRO
MOTEUR
D'EAQUE
LN1
LN2
BASE
ORDF
REGLES
ORDF
PROBLEME
IC
E
U
: EST ECRIT EN
: UTILISE
: PARTICULARISE
: DEFINIT
: PROCEDURES
: BASE DE
CONNAISSANCE
: LANGAGES
: OPERATEURS

LRO

La représentation des connaissances est un des points clés de la résolution des problèmes. Il existe différents formalismes correspondant aux diverses connaissances que nous manipulons. Après un bref rappel sur les approches déclaratives et procédurales, nous rappellerons les principaux concepts existants et les procédures d'exploitation associées : le formalisme des réseaux sémantiques pour la connaissance descriptive, celui des règles de production pour la connaissance déductive, celui des frames alliant procédural et déclaratif. Nous en viendrons naturellement aux langages associés à ces structures : KRL, FRL et aux langages orientés objet : SMALLTALK.

Initialement, notre but était de développer un formalisme de représentation de connaissances descriptives adapté aux différentes structures de données rencontrées dans les problèmes d'ordonnancement. Les instructions de déclaration et d'exploitation de ce formalisme constituent en fait un langage beaucoup plus général de représentation d'objet, que nous avons appelé **LRO.**

L'élément de représentation élémentaire de la connaissance dans l'environnement LRO est l'**objet.** Il se définit comme un **atome auquel est rattaché un ensemble de propriétés.** Tout objet appartient à une **classe.** La classe permet de regrouper des objets "semblables". Elle est définie par une déclaration de **type** (type d'objet) qui permet de lui associer :

1- un ensemble de propriétés fixe qui sera associé à chaque

objet de cette classe.
2- des valeurs prédéfinies pour certaines propriétés que nous retrouverons par défaut pour tout objet de cette classe s'il n'a pas été précisé d'autres valeurs lors de la déclaration de ces objets.

L'instruction de déclaration d'objet permet d'associer à un objet un nom, la classe à laquelle il appartiendra, et les valeurs des propriétés de cette classe pour cet objet. Notons pour terminer l'aspect déclaratif de l'objet, l'existence de la propriété prédéfinie **EST_UN** qui permet à un objet d'une classe donnée de "pointer" sur un objet d'une classe différente et d'hériter ainsi des propriétés et valeurs de l'objet pointé.
Nous verrons dans la section correspondante le détail des instructions pour ce niveau déclaratif.

LRO permet également de manipuler des procédures. Son principe de base est l'**attachement procédural aux propriétés.**
Ces procédures sont activées sur écriture, par la modification de la valeur de ces propriétés. L'association en une même entité, de données et de leurs procédures d'exploitation, se trouve complétée par l'existence d'envois de **MESSAGES.** Un message est constitué d'un nom, désignant le type de manipulation désirée, d'un destinataire (pouvant être un objet, une classe d'objets, ou tout objet) et de paramètres du message. Le nom du message est une propriété à laquelle est attachée une procédure corespondant à la manipulation souhaitée lors de l'envoi du message. Un objet peut recevoir un message si ce dernier a été précédemment déclaré comme propriété et point d'entrée de la classe de cet objet. Les réponses à un même message pour deux objets d'une même classe diffèrent selon les différences des valeurs des propriétés de ces deux objets. Nous pouvons ainsi associer à une classe le traitement de ses objets et

l'appeler par un envoi de message. Nous verrons lors de la description du moteur d'EAQUE qu'une telle approche a permis de faciliter grandement sa conception, sa programmation et sa mise au point.

EAQUE

L'approche "Système Expert" est actuellement une technique très utilisée dans le domaine de l'Intelligence Artificielle. Elle peut être d'ailleurs considérée comme une méthode particulière de programmation. Nous présenterons l'organisation de tels systèmes et le vocabulaire employé, en insistant particulièrement sur les aspects structure de contrôle.

EAQUE est un moteur d'inférence général pour une certaine classe de problèmes. C'est-à-dire qu'il n'intègre aucune connaissance particulière à un domaine d'application, mais est caractérisé par un ensemble d'options choisies communes à des problèmes rencontrés dans différents domaines. Ces options correspondent à des choix de modes de représentation des connaissances et de structure de contrôle particuliers que nous expliciterons.

LRO est utilisé pour décrire à EAQUE le problème à résoudre. Il est constitué d'un ensemble d'objets (au sens LRO) qui peuvent représenter en particulier des sous-problèmes à résoudre. EAQUE connaît un objet par son nom et non par sa structure. C'est cete dernière qui donnera au sous-problème son sens concret lors d'une application.

Le formalisme des **règles de production** a été choisi pour représenter les connaissances permettant à EAQUE à un moment donné de choisir l'objet (sous-problème courant à résoudre) sur lequel agir. Elles indiquent (en partie droite de la règle) sous quelles conditions (en partie gauche de la règle) tel objet est "intéressant".

Ces règles possèdent un **coefficient d'importance**, le CIR, (Coefficient d'Importance de la Règle). Ces règles expriment donc des **critères de choix d'objets pondérés.**

L'utilisation de ces règles et de leur CIR correspond à un **système de notation des différents objets.** L'objet ayant obtenu la note la plus élevée (dépassant un seuil) est choisi. La fonction de notation et le seuil sont des éléments par défaut de EAQUE redéfinissables par l'Ingénieur Cognitif selon les souhaits de l'expert.
Bien que ce choix d'objet ne correspond pas à un changement d'état réel, il n'y a eu aucune modification faite du contexte, il est considéré comme un point de reprise possible (remise en cause de ce choix) en cas de contradictions. Dans ce but, il lui est associé une autre note, "évaluation du doute", au moment où il a été fait. Cette note est fonction de la note de l'objet retenu, du nombre des autres objets pouvant être choisis et du maximum de leur note. Cette fonction est également redéfinissable dans un contexte d'application particulier.

Le formalisme des **règles de production** a été aussi choisi pour représenter les méta-règles permettant de modifier, selon le contexte, l'importance des règles actions. Ces règles actions sont également représentées sous forme de règles de production. Elles indiquent (en partie droite de la règle) les conditions (en partie gauche de la règle) sous lesquelles il est préférable de faire ou de ne pas faire telles ou telles actions. Ces actions s'appliquent sur l'objet préalablement choisi. Les domaines d'instantiation des variables de la règle sont des valeurs de propriétés des objets. Ces domaines devront être définis lors de l'écriture de la règle puisqu'elles dépendent de l'application et ne peuvent être connues de EAQUE. Ces règles possèdent également un CIR, induisant un **système de notation des actions.** L'action ayant obtenu la note la plus élevée est

choisie. La fonction de notation et le seuil pour les actions sont, soit ceux d'EAQUE (par défaut), soit redéfinissables pour un domaine particulier. A noter qu'à chaque nouvelle action rencontrée lors de la recherche de l'action la plus adéquate, EAQUE génère un objet LRO correspondant, et qu'en ce sens, il peut être qualifié de moteur avec variables.
Ce choix d'action correspond à un changement d'état réel et doit être considéré comme un point de reprise potentiel. C'est pourquoi il lui est associé une note de manière similaire au choix d'objet.

EAQUE possède, pour la saisie des connaissances que nous venons d'énumérer, un squelette d'analyse syntaxique qui sera particularisé lors de son application à un domaine donné.

Le fonctionnement du moteur d'inférence de EAQUE peut se résumer ainsi, sachant que :

1- EAQUE s'"arrête" dès que sa **condition d'arrêt** est vérifiée. La condition d'arrêt est, soit celle d'EAQUE par défaut, à savoir l'inexistence d'objet "actif" (sous-problème à résoudre), soit redéfinissable pour un domaine et un problème particulier.

2- Chaque cycle élémentaire doit vérifier des **objectifs** ou contraintes. Les objectifs sont définis pour un domaine et un problème particulier. Par défaut il n'en existe pas. Le non respect de ces objectifs se traduit par la génération de **contradictions.**

3- Un état ayant généré des contradictions est un état incorrect. Deux solutions sont alors possibles pour les résoudre en fonction des notes asociées aux points de reprise et de l'analyse des objectifs contredits. C'est,

soit une restauration à un état antérieur, soit un "relâchement" des objectifs contredits afin de supprimer les contradictions pour l'état courant.

PROCEDURE EAQUE :

TANT QUE la condition d'arrêt n'est pas vérifiée **ET** que les objectifs sont respectés **REPETER**

! choix d'objet
! choix d'action
! exécuter l'action et propager les conséquences

SI il existe des contradictions
ALORS
! resoudre ces contradictions
! EAQUE

FIN.

La procédure de contrôle est de type EOA : choix d'état, choix d'objet et choix d'action.

LE CHOIX D'ETAT : En principe, l'état sur lequel le système se doit d'agir est le dernier état généré. Le problème du choix d'un autre état ne se pose qu'après rencontre d'une contradiction. Il correspond à un retour arrière sélectif à un état antérieur. L'état à restaurer est celui dont "l'évaluation du doute" a la note maximum.
EAQUE peut être utilisé pour des domaines où les actions sont non permutables et/ou non réversibles puisqu'il contient un mécanisme permettant de restaurer des états antérieurs.

LE CHOIX D'OBJET : Ainsi que nous l'avons vu, le choix d'objet se fait par l'utilisation conjuguée de règles

(critères d'objet) et d'un système de pondérations heuristiques.

LE CHOIX D'ACTION : C'est l'action ayant la note la plus forte parmi les actions applicables, note obtenue par l'application pondérée de règles actions.

EAQUE est écrit entièrement dans l'environnement LRO-LISP.

ORDF

Les problèmes d'ordonnancement sont des problèmes complexes. Il n'existe pas dans certains cas (NP-complets) d'algorithme de résolution en temps et espace raisonnable. C'est pourquoi l'approche heuristique et les techniques de l'Intelligence Artificielle peuvent apporter des solutions intéressantes.

EAQUE est associé à une classe de problèmes dont font partie les problèmes d'ordonnancement. C'est pourquoi nous l'avons appliqué à l'un d'entre eux, celui de l'ordonnancement de travaux (par exemple dans le domaine du bâtiment), dans le cas où il est connu, pour chaque tâche, sa durée ainsi que le nombre et le type de ressources qu'elle utilise. Cette application constituant le système **ORDF**, (Ordonnancement à Ressources et Durées Fixées), est la définition, à l'intention d'EAQUE, de deux ensembles de connaissances. Le premier, déclaratif, est l'ensemble des règles fournies par l'expert, relatives aux choix d'objet et d'action. Le second, que nous avons appelé la **base ORDF**, est l'ensemble des propriétés du domaine énoncées par l'Expert à l'Ingénieur Cognitif, traduites en général sous forme de procédures par celui-ci, et contenant en particulier la définition de ce que sont, dans le domaine, les "objets" et les "actions", et leurs propriétés particulières (utilisées dans l'expression des critères d'objet et d'action).

Il existe deux classes d'objets LRO nécessaires à la description des problèmes soumis à ORDF. Ce sont les classes **tâche-simple** et **ressource.** Les objets de type tâche-simple représentent, dans le domaine du bâtiment, les tâches de l'ouvrage à réaliser pour chaque corps de métier. Ceux de

type ressource décrivent les ressources disponibles et leur quantité.

Citons pour exemple les propriétés associées à la classe tâche-simple. Pred pour les predécesseurs de la tâche, suc pour ses successeurs, durée pour sa durée, res pour les ressources qu'elle utilise, dtot et dtard pour ses dates au plus tôt et au plus tard. Le graphe implicitement défini par les objets de type tâche-simple et les valeurs des propriétés pred et suc traduit les contraintes de conception, contraintes technologiques, de la fabrication de l'ouvrage considéré.

Le calcul des dates au plus tôt et au plus tard, la recherche de circuit, la relation de transitivité, sont des exemples de procédures de la base ORDF.

Les sous problèmes à résoudre d'ORDF sont les ensembles d'objet de type tâche-simple pouvant accéder en même temps à une même ressource dont les quantités sont insuffisantes pour satisfaire plusieurs demandes.

CALINT

En réalisant CALINT, notre but était moins d'obtenir un système de calcul d'intégrales et de simplification d'expressions mathématiques performant, que de montrer qu'EAQUE pouvait être appliqué sans difficulté à d'autres domaines aussi différents que peuvent l'être ORDF et CALINT.

CALINT est principalement destiné à la simplification d'expressions. Il possède pour cela plus de soixante dix règles de réécriture. Cette simplification se décompose en deux phases, la première de développement, la seconde de simplification et de mise en facteur. La description du problème est ici très simple, il n'y a qu'un seul objet LRO décrivant l'expression à simplifier. Par contre la base CALINT, par analogie avec la base ORDF, est relativement riche : somme exacte de deux fractions, détermination de la dérivée d'une expression ...

Exemple:
(2/3)*(4*X-(5/2)*Z+(17/9))-(10/3)*(X+4*Z)+(2/3)*X+(8/5)
se simplifie en (386/135)-15*Z.

EALOG

PROLOG est actuellement un langage très à la mode, c'est pourquoi il nous a semblé utile de montrer que sa structure de contrôle est incluse dans EAQUE :c'est le système EALOG.

EALOG utilise, évidemment, aucune des caractéristiques rattachées aux coefficients d'importance, ni les différents types de règles que permet EAQUE.
Enfin, l'écriture des règles EALOG respecte la syntaxe PROLOG.

Exemples:
Pour la description de la base de faits :
Julien est le fils de Patricia s'écrit :
(FILS JULIEN PATRICIA) -►;
Patricia et Christophe forment un couple s'écrit :
(COUPLE CHRISTOPHE PATRICIA) -►;

Pour les règles de déduction :
La règle suivante : "X est le fils de Y, si X est un homme, et si X est l'enfant de Y" s'écrit :
(FILS ?X ?Y) -►(HOMME ?X) (ENFANT ?X ?Y) ;

CONCLUSION

Elle portera principalement sur l'intérêt de EAQUE et de LRO. Nous verrons la faisabilité, les inconvénients et les avantages d'une approche générale pour la résolution de problèmes.

-* CHAPITRE II *-

LRO

-* CHAPITRE II *-

LRO

RESUME : Il est préférable avant toute réalisation de se doter d'outils logiciels puissants. LRO (Langage de Représentation d'Objet) offre non seulement les possibilités de définir et d'utiliser des structures de données complexes, mais également un environnement de programmation très structuré, de type Langage Orienté Objet.

MOTS CLES : Modes de Représentation des Connaissances, Procédural, Déclaratif, Frames, Règles de production, Langages Orientés Objet, Messages.

INTRODUCTION

Les premières recherches en Intelligence Articielle (années 60) ont principalement porté sur l'élaboration de méthodes générales de résolution de problèmes, appliquées à des domaines bien formalisés, comme la théorie des jeux et le calcul symbolique. Nous en retiendrons principalement les fonctions d'évaluation heuristiques. Citons, pour exemple, la procédure alpha-beta, dont une première formulation a été faite par Newell,Shaw et Simon en 1958.

Quelques années plus tard (années 70), l'étude de problèmes plus "pratiques" a montré qu'il est parfois préférable de disposer d'une base de connaissances riches et structurées, plutôt que d'une procédure de résolution complexe. La représentation des connaissances, et leurs modes d'utilisation, est alors devenu un thème central de recherches en Intelligence Artificielle.

Nous rappellerons les principaux modes de représentation des connaissances utilisés, car ils interviennent tous, d'une manière ou d'une autre, à travers le système LRO-EAQUE.

LA REPRESENTATION DES CONNAISSANCES

Nous utilisons, quotidiennement, des connaissances de nature différente. Cette nature n'est pas liée a cette connaissance en elle-même, mais à l'utilisation qui en sera faite. Ainsi, nous pouvons concevoir des connaissances factuelles, descriptives, décrivant le monde qui nous entoure. "La neige est blanche" en est un exemple. Il existe également des connaissances déductives, permettant d'inférer, par exemple, de nouveaux faits : "S'il existe les conditions météorologiques adéquates, alors il est probable qu'il neige". Nous possédons, de même, des connaissances sur ce que nous savons. Nos connaissances relatives à la neige sont utilisables en hiver.
Il évident, qu'il existe d'autres type de connaissances, par exemple celles qui prennent en compte le déroulement du temps : "hier il a neigé". Mais également des connaissances correspondant à des schémas plus complexes, telles nos réactions face à une situation connue : "prendre un repas dans un restaurant".

Le problème qui se pose, à un informaticien, est le choix de la représentation correspondant à l'utilisation désirée de la connaissance associée; la facilité d'inférer en dépend. Il se présente d'abord, et en général, comme un choix entre une **représentation déclarative** et une **représentation procédurale** de la connaissance. Ces approches n'ont pas à être opposées. Tout formalisme, même s'il intègre les deux, gagne à être vu sous cet aspect. L'approche déclarative est une définition statique des connaissances sans "mode d'emploi associé", cela sous entend un mécanisme chargé d'exploiter ces connaissances qui peuvent être alors

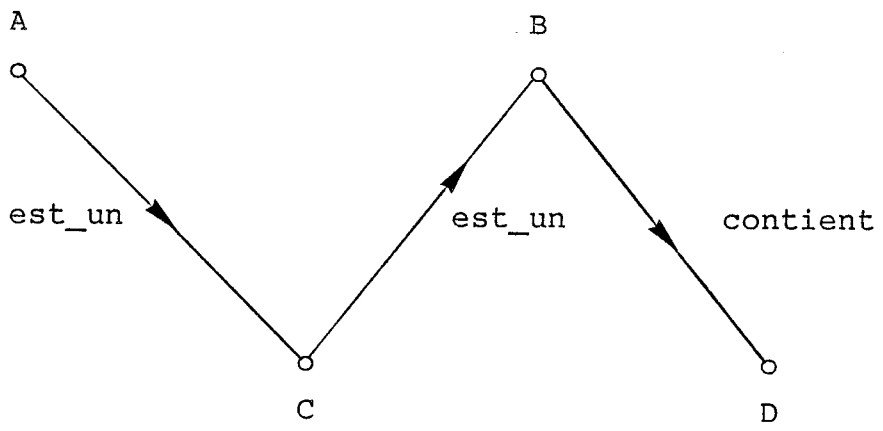
considérées comme indépendantes les unes des autres. L'approche procédurale implique de savoir quand et comment seront utilisées ces connaissances dont les liens sont à expliciter. Elles donnent lieu à l'écriture de programmes.

Les sections suivantes rappellent briévement différents formalismes, à l'exclusion évidemment des procédures, en partant du plus déclaratif, aux formalismes associant déclaratif et procédural.

-1- LES RESEAUX SEMANTIQUES.

Issus de travaux de psychologues sur les modèles de la mémoire humaine (Quillan 1968), ce formalisme est utilisé pour représenter les relations entre concepts. C'est une approche déclararative d'une connaissance descriptive. Il se définit comme un ensemble de noeuds, représentant des objets, des concepts, des individus, des situations, et un ensemble d'arcs orientés traduisant les relations (binaires) entre ces noeuds.

Ce que nous pouvons représenter par le graphe suivant :



Parmi les avantages de ce formalisme, et de ses utilisations possibles afin d'en déduire de nouveaux faits, nous citerons :

- la disponibilité, d'une manière simple, de toutes les informations concernant un noeud. Ainsi, du graphe précédent nous en déduisons que A est aussi un B.
- la notion d'héritage. Il y a distinction entre la notion de classe et d'individus de cette classe. Tout ce qui se rapporte à une classe est vrai pour chacun de ses éléments. Ainsi, si A est un individu et B une classe, A contient aussi D.
- l'utilisation du matching de réseaux, pour la recherche d'un objet.

PROSPECTOR (qui utilise aussi des règles de production) et CASNET, sont parmi les systèmes les plus connus utilisant des réseaux sémantiques.

-2- LES REGLES DE PRODUCTION.

La règle de production est le formalisme utilisé pour la représentation des connaissances déductives dans un système de production. Bien que ces connaissances peuvent être vues comme procédurales, nous retrouvons en elles tous les avantages de l'approche déclarative et particulièrement la modularité.
Les systèmes de production, proposés par POST (1943), sont constitués, schématiquement, de trois parties : un ensemble de règles, une base de faits, et un interpréteur pour contrôler l'activité du système.
Les règles de production sont de la forme : "SI conditions ALORS actions", la base de faits contient les connaissances descriptives, l'interpréteur permet de démontrer un but (un théorème) à partir de la base de faits et de la base de règles.

Le formalisme des règles de production est suffisamment explicite pour ne pas être détaillé. Il est actuellement le plus utilisé pour la représentation des connaissances déductives dans les Sytèmes Experts. PROSPECTOR et MYCIN sont parmi les systèmes les plus connus, utilisant les règles de production. Citons, pour exemple, une règle du cas d'école de MYCIN :

SI 1- le site de la culture est le sang,
et si 2- l'organisme est à gram négatif,
et si 3- l'organisme est de forme bâtonnet,
et si 4- le patient est un hôte à risque,
ALORS il est probable (0.6) que l'organisme est le pseudomonia aeruginosa.

Il existe deux principaux modes d'utilisation des règles :
- la résolution dirigée par les données, ou chaînage avant, ou Modus Ponens : Si les conditions de la règle sont vérifiées dans la base de faits, les actions, décrites dans sa partie droite sont réalisées.
- la résolution dirigée par les buts, ou chaînage arrière : La partie condition d'une règle concluant au but à démontrer, constitue les nouveaux buts à considérer. Dès que ceux-ci sont démontrés, le but initial l'est également.

-3- LES FRAMES.

Il est établi, en psychologie, que face à une situation nouvelle, "voisine" d'expériences antérieures, nous utilisons des connaissances stéréotypées qui en ont été déduites (BARTLETT 1932). Ainsi "aller au restaurant" correspond à un scénario connu. Cette idée est à la base des travaux de MINSKY (1975) sur les frames. Un frame représente un concept générique. Il peut se définir comme un ensemble de propriétés, connaissances descriptives du concept, auxquelles sont attachées, si besoin est, des valeurs par

défaut, le domaine de variables de ces valeurs, ou des procédures (attachements procéduraux) pour déterminer ces valeurs. Ces procédures, rattachées au concept, vont permettre de conduire la résolution du problème. Elles représentent la connaissance procédurale du frame qui allie les deux modes de représentation. Une instance du concept est issue du concept en déterminant pour chaque propriété sa valeur. Les procédures peuvent être activées de différentes façons, pour la détermination de la valeur des propriété auquelles elles sont rattachées, à l'accès de ces valeurs... A noter qu'une propriété d'héritage (EST_UN des réseaux sémantiques), permet de hiérarchiser les frames entre eux.

La première utilisation d'un frame est la génération d'une instance, par la détermination des valeurs des propriétés.

Donnons, pour exemple, un concept générique d'une table.

```
frame TABLE :
      est_un           : MEUBLE
                          /* héritage du concept meuble*/
      nombre_de_pieds  :
                         domaine        : entiers
                         par défaut     : 4
                         si_nécessaire  : compter les pieds
      style            :
                         domaine        : (moderne, Louis-Philippe)
                         par défaut     : moderne
                         si_nécessaire  : SI  pieds courbés Alors
                                            Louis-Philippe
```

Signalons, pour terminer cette partie et introduire la suivante, l'existence de deux langages frames, KRL et FRL, permettant la définition et l'utilisation de frames.

-4- LES LANGAGES ORIENTES OBJET.

Cette quatrième partie peut, par rapport aux trois autres sections précédemment décrites, paraître à première vue hors propos, puisque c'est un langage de programmation. En fait, nous retrouvons dans ce langage beaucoup de caractéristiques communes avec ce que nous venons de voir; citons les concepts d'objet, alliant procédural et déclaratif, de classe d'objets, d'héritage, d'attachement procédural.

Les langages orientés objet constituent certainement, à l'heure actuelle, une des approches de programmation les plus structurées. Ce ne sont pas seulement des langages parmi d'autres, mais une approche différente des systèmes informatiques.

Un système d'informatique "classique" est constitué de deux types d'entités différentes : **les données** et **les procédures** chargées de les exploiter. Dans de tels systèmes, ces entités sont considérées comme indépendantes. Pour les langages orientés objet, il n'existe qu'un type d'entité : **l'objet.** L'objet permet de représenter, d'associer, en une seule entité, les données et leurs procédures d'exploitation. L'utilisation des objets (des données qu'ils représentent) se fait en leur envoyant des **messages.** Un message est constitué d'un nom, désignant le type de manipulation désirée, de l'objet destinataire, et de paramètres du message. En envoyant un message, le programmeur décrit ce qu'il désire, et non commemt l'obtenir. A la différence d'un appel de procédure, un même message peut être interprété de façon différente par différents destinataires. A la réception d'un message le destinataire détermine la procédure, que nous appellerons désormais méthode, à activer. Contrairement aux procédures, ces méthodes, qui ne peuvent être séparées de leurs objets, ne peuvent appeler directement d'autres méthodes.

La notion de classe permet de regrouper des objets de description semblable, chacun de ces objets est une instance de cette classe. Toutes les instances d'une même classe utilise donc les même méthodes (qui sont associées à la classe) en réponse à un même message. Les réponses diffèrent selon les différences des instances.
Notons, pour terminer que les classes sont elles mêmes des objets, et que nous avons, tout comme les réseaux sémantiques et les frames, le concept d'héritage.

Nous verrons, avec LRO, un exemple détaillé d'un tel langage.

LRO

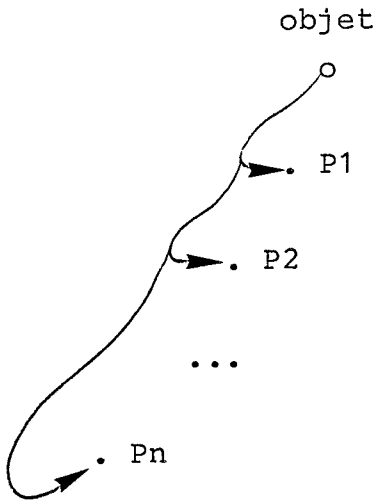
LANGAGE DE REPRESENTATION D'OBJET

Initialement, notre but était de développer un formalisme de représentation de connaissances descriptives adapté aux différentes structures rencontrées dans les problèmes d'ordonnancement. Au fur et à mesure de l'étude de la résolution de ces problèmes, la notion de classe avec des valeurs par défaut, et les attachements procéduraux devenaient nécessaire, nous avons donc associé à ces connaissances purement déclaratives, un aspect procédural. L'approche volontairement générale du problème aidant, la réalisation d'un langage permettant de définir et d'utiliser de telles structures est venue naturellement. Les possibilités qu'offrait alors LRO, et en particulier l'utilisation qui était faite des attachements procéduraux, nous a incité à rajouter, entre autres, les envois de messages; ce dans le but de nous doter pour l'écriture d'EAQUE d'un langage qui nous faciliterait grandement sa réalisation. Ainsi, LRO est utilisé pour décrire le problème à résoudre, et EAQUE est écrit en LRO-LISP.

L'ASPECT DECLARATIF DE LRO

-1- L'OBJET.

L'élément de représentation élémentaire de la connaissance dans l'environnement LRO **est l'objet.** Il se définit comme **un atome auquel est rattaché un ensemble de propriétés.**
Il peut être représenté par la figure suivante :



-2- LA CLASSE.

Tout objet appartient à une classe. La classe permet de regrouper des objets "semblables". Les objets d'une même classe possèdent le même ensemble de propriétés et ne diffèrent que par les valeurs de celles-ci.

-3- L'HERITAGE.

Il existe deux types d'héritages possibles pour un objet :

l'héritage par défaut de valeurs de propriétés, et un héritage de structures d'objets.

-l'héritage par défaut :
La déclaration d'une classe (cf le point suivant) permet de définir l'ensemble des propriétés qui sera associé à chaque objet de cette classe. Associer, lors de cette définition, des valeurs à des propriétés, fait que les objets de cette classe hériteront, par défaut, de ces valeurs, s'il n'a pas été précisé d'autres valeurs pour ces propriétés, lors de la déclaration de ces objets.

-l'héritage de structures d'objets :
L'existence de la propriété prédéfinie EST_UN permet à un objet d'une classe donnée de "pointer" sur un objet d'une classe différente, et d'hériter ainsi de ses propriétés et valeurs. Cett définition de l'héritage est récursive, l'objet récupère également l'héritage, s'il existe, de l'objet "pointé".

-la hiérarchie des classes :
Il n'existe pas de hiérarchie explicite entre les classes. Elle n'est effectivement pas nécessaire, puisqu'elle peut se traduire en termes d'héritage par défaut et d'héritage de structures d'objets. En effet, il suffit de définir pour chaque classe un objet la représentant (ensemble des valeurs par défaut). Associer à une première classe la propriété EST_UN de valeur un objet représentant une deuxième classe, permet de définir la première classe comme une sous classe de la seconde.

-4- LES INSTRUCTIONS SUR LES CLASSES, LES OBJETS, LES PROPRIETES.

-a- déclaration.

* toute classe doit être définie par une déclaration de **type** (type d'objet). Cette déclaration permet d'associer à une classe :
-un nom (le type),
-un ensemble de propriétés fixe qui sera associé à chaque objet de cette classe,
-des valeurs prédéfinies pour certaines propriétés, qu'hériteront, par défaut, tout objet de cette classe s'il n'est pas précisé d'autres valeurs lors de la déclaration de ces objets.

exemple :

```
( TYPE     table     ( EST_UN  meuble )
                     ( PIEDS      4    )
                     STYLE
                     COULEUR              )
```

Cette instruction définit la classe table décrite par les propriétés EST_UN, PIEDS, STYLE et COULEUR. Par défaut toutes les tables ont quatre pieds, et ont toutes les caractéristiques d'un meuble.

Remarque :
Une même propriété peut appartenir à plusieurs classes. Nous en verrons l'intérêt dans la section "aspect procédural" de LRO.

* tout objet doit être défini et appartenir à une classe par une déclaration d'objet. Cette déclaration permet d'associer à un objet :
-un nom,
-la classe à laquelle il appartiendra, et dont il va hériter les propriétés et les valeurs par défaut,
-les valeurs des propriétés de cette classe pour cet objet (valeurs prioritaires, évidemment, sur les valeurs par défaut).

exemple :
(SOMMET table_de_travail table (STYLE directoire)) définit l'objet table_de_travail comme appartenant à la classe table, c'est un meuble qui a quatre pieds (valeurs par défaut héritées de la définition de la classe table), de style directoire, dont la couleur n'est pas précisée.

Remarque :
- l'ordre des propriétés dans la déclaration de l'objet n'a pas d'importance.
- le terme SOMMET, au lieu d'objet, sera justifié dans la section concernant l'implémentation de LRO.

-b- instructions sur les classes.

En dehors de l'instruction de déclaration, nous disposons de plusieurs instructions, dont :
-(PROPS type) : retourne les propriétés définissant le type.
-(SOMMETS type) : retourne la liste des sommets de cette classe au moment considéré.
-RAJOUT : permet de rajouter des propriétés et des valeurs par défaut à une classe, et de là, à tous ses objets.
-KILLT : détruit une classe, et à fortiori ses objets.

-c- instructions sur les objets.

Citons quelques exemples d'instructions portant sur les objets :
-(Nom_de_la_propriété objet) : donne la valeur de la propriété Nom_de_la_propriété pour cet objet.
-(TYPE objet) : retourne le type de l'objet.
-(S= objet1 objet2) : retourne vrai si les deux objets sont identiques, faux sinon.
-(S:= objet1 objet2) : objet1 et objet2 sont de même type, objet1 prend pour valeur objet2.

-(S→ objet1 objet2) : objet1 et objet2 sont de même type, permet le double accès à une même structure.
-(P:= (PROP1 objet1) expr) : la propriété PROP1 de l'objet1 prend pour valeur expr.
Il existe également plusieurs instructions portant sur les listes d'objets.

-c- instructions sur les propriétés.

Il existe deux instructions sur les propriétés, la première permet de connaître tous les objets, quelque soit leur classe, qui utilisent cette propriété (une propriété n'est pas l'exclusivité d'une classe), la deuxième, en anticipant sur la prochaine section, retourne la procédure attachée à cette propriété, si elle existe.

L'ASPECT PROCEDURAL DE LRO

-1- L'ATTACHEMENT PROCEDURAL AUX PROPRIETES.

LRO permet de manipuler des procédures. Son principe de base est **l'attachement procédural aux propriétés.** Ainsi, une même procédure peut être associée à différentes classes par l'intermédiaire d'une même propriété. Ces procédures sont activées sur écriture, par la modification des valeurs des propriétés auxquelles elles sont rattachées, et ce indépendamment de l'objet considéré.
Ainsi, si P1 est une propriété (a apparu au moins une fois dans une déclaration de type), nous lui associons une procédure par l'instruction suivante :
(AT_PROC P1 procédure)
Dès lors, toute instruction (P:= (P1 objet) expr) doit être en fait vue comme (P:= (P1 objet) procédure(expr)). L'affectation à P1 de la valeur expr est en réalité l'affectation à P1 du résultat de la procédure de paramètre(s) expr.

Exemple : Il est associé, à la propriété PIEDS utilisée par la classe table, une procédure permettant de vérifier si le nombre de pieds d'une table est un entier.

-2- LES MESSAGES.

Nous venons de voir que LRO permet d'attacher des procédures à des propriétés, et de là aux classes qui les utilisent. Ainsi, nous pouvons concevoir des procédures chargées

d'exploiter les objets d'une classe. Elles seront associées à cette classe par les attachements procéduraux. Ces procédures s'activeront sur réception du **message** adéquat.

Il existe donc, dans LRO, deux types de propriétés qui peuvent intervenir dans la définition d'une classe :
- les premières sont utilisées pour la représentation des connaissances descriptives. A ces propriétés peuvent être attachées des procédures, de vérification de cohérence par exemple, dont le mode d'activation a été décrit au point -1- précédent.
- les deuxièmes sont utilisées pour représenter les procédures d'exploitation de la classe (ce sont les mêmes procédures pour tous les objets de la classe). Dans ce cas, la propriété prédéfinie **POINTS_ENTREE** a pour valeur l'ensemble des propriétés auxquelles sont attachées ces procédures, elles représentent les **points d'entrée de la classe.**

Le mode d'activation des procédures rattachées aux points d'entrée se fait sur réception d'un message. **Un message est constitué d'un nom, désignant le type de manipulation désirée, d'un destinataire, et de paramètres du message.** Le destinataire peut être un objet, une classe d'objets, ou tout objet. Dans le cas où le destinataire est une classe d'objets, LRO distribue le message sur chaque objet de la classe. Le nom du message est un point d'entrée. Un objet (nous sommes toujours ramené à l'envoi du message à un objet) peut recevoir un message si celui-ci est un point d'entrée de la classe de l'objet, sinon il est ignoré. Les réponses à un même message pour deux objets d'une même classe diffèrent selon les différences des valeurs des propriétés du premier type (descriptives) de ces deux objets

Exemples :
- (►OBJET objet1 (point_entrée2 p1 p2)) est l'envoi

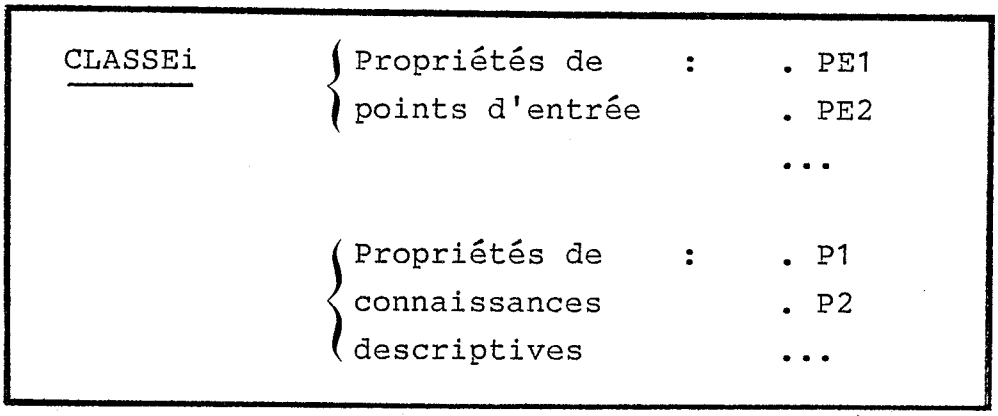
du message point_entrée2 de paramètres p1 et p2 à l'objet objet1

- (►TYPE type1 (point_entrée3 p1)) est l'envoi du message point_entrée3 de paramètre p1 à la classe type1, qui se traduira par l'envoi du même message à tous les objets de cette classe.
- (► (point_entrée4 p2 p4)) est l'envoi du message point_entrée4 de paramètres p2 et p4 à tous les objets, pouvant le recevoir.

L'envoi d'un message à un objet se traduit donc, dans le cas où il est valide, par l'activation de la procédure attachée au point d'entrée correspondant, de paramètres, les paramètres du message.

Note : Pour l'écriture d'une procédure attachée à un point d'entrée, il est utilisé une variable globale, objet, qui désigne l'objet sur lequel portera cette procédure. Par son intermédiaire, nous accédons à toute la structure de l'objet. Elle prendra pour valeur l'objet destinataire final.

Une classe peut se représenter par la figure suivante :



La gestion des messages par LRO est faite en depth-first.

L'envoi d'un message à une classe se traduit par l'envoi du même message à chaque objet de la classe. Les messages qui peuvent être issus de la manipulation d'un objet seront interprétés avant les messages aux objets restants.

Les messages circulent sur un bus "virtuel". Il peut être fermé ou ouvert. Ainsi, nous envoyons un message à une classe dans un but donné. Ce message est distribué sur chaque objet de la classe. Dès que la manipulation d'un objet a sastisfait le but (qui nous avertira par l'envoi du message adéquat), il n'y a aucune raison que les messages restants soient envoyés, le bus est coupé, les messages restants sont perdus.

A noter que LRO traduit un message en modification de propriété (le point d'entrée), afin d'utiliser le même mécanisme décrit au point -1- précédent.

Nous verrons un exemple d'utilisation de ces messages dans l'implémentation d'EAQUE.

-3- LES SEMAPHORES

Il existe, afin de rendre LRO davantage structuré, la propriété prédéfinie **SEMAPHORE.** Sa présence, dans la déclaration d'une classe, interdit toute modification des propriétés de tous les objets de la classe, à l'exception des procédures d'exploitation qui leur sont associées (attachées aux points d'entrée de la classe).

L'IMPLEMENTATION

LRO considère, pour sa gestion des classes et de leurs objets, trois types d'objets : les types, les propriétés et les sommets. Les types sont les objets de la classe prédéfinie TYPE, ils décrivent l'ensemble des classes utilisées (niveau utilisateur). Cette classe est définie par les propriétés suivantes : PROPS, pour l'ensemble des propriétés associé au type (classe); SOMMETS, qui contient l'ensemble des objets de cette classe; et COMMUNS, pour les valeurs par défaut des propriétés. Les propriétés sont les objets de la classe prédéfinie PROPRIETE décrivant l'ensemble des propriétés utilisées (niveau utilisateur). Cette classe est définie par les propriétés suivantes : AT_PROC, de valeur par défaut l'instruction vide, contient la procédure qui lui est associée; SOMMETS, de valeur l'ensemble vide par défaut, contient les objets possédant cette propriété. Enfin, les sommets représentent tous les objets de toutes les classes (niveau utilisateur), mais n'appartiennent à aucune classe prédéfinie (structures différentes).

(Comme (nous (nous en) doutons (LRO est (écrit) en LISP))), cela présente de nombreux avantages, nous citerons en particulier les propriétés associées à un atome, la représentation de la liste de ces propriétés en tant que CDR de l'atome, la présence de la fonction EVAL, etc...

A noter, pour terminer, l'existence d'un garbage collector LRO, qui permet de récupérer, lors de la destruction d'un objet, les listes correspondantes.

-* CHAPITRE III *-

EAQUE

-* CHAPITRE III *-

EAQUE

1 - INTRODUCTION , Systèmes Experts et Structures de Contrôle.
2 - EAQUE.
3 - LE DOMAINE D'EAQUE : OBJETS, ACTIONS, CONDITION D'ARRET, OBJECTIFS.
4 - STRUCTURE GENERALE DU MOTEUR.
5 - CHOIX D'OBJET, CHOIX D'ACTION, EXECUTION.
6 - ETUDE ET RESOLUTION DES CONTRADICTIONS.
7 - INSTANTIATION D'EAQUE A UN DOMAINE D'APPLICATION PARTICULIER.
8 - IMPLEMENTATION.

RESUME : EAQUE est un moteur d'inférence général pour une une certaine classe de domaines d'application. Il n'intègre aucune connaissance particulière à l'un de ces domaines, mais est caractérisé par un ensemble d'options qui leur sont communes. Ces options correspondent à des choix de modes de représentation des connaissances et de structure de contrôle. LRO est utilisé pour décrire le problème à résoudre. Le formalisme des règles de production a été choisi pour représenter les connaissances de résolution et de contrôle. La structure de contrôle est de type EOA. L'instantiation d'EAQUE à différents domaines d'application génère autant de systèmes experts. Enfin, EAQUE est écrit entièrement en LRO-LISP.

MOTS CLES : Moteur d'inférence général, structure EOA, règles de production, fonctions d'évaluation heuristiques, négation d'objectifs, retour arrière sélectif, envois de messages.

INTRODUCTION

Systèmes Experts et Structures de Contrôle

:A: PRELIMINAIRE.
:B: SYSTEMES EXPERTS ET STRUCTURES DE CONTROLE.

:A: PRELIMINAIRE.

L'approche Système Expert est actuellement une technique très utilisée dans le domaine de l'Intelligence Artificielle. Ce succès a plusieurs raisons. In fine, pour les utilisateurs, ce sont les résultats et les performances de ces réalisations, qui sont comparables à celles d'experts humains dans des domaines aussi variés que la Médecine (MYCIN, CASNET), la Géologie (PROSPECTOR), la Chimie (DENDRAL, CONGEN). C'est également le type de problème que nous pouvons espérer résoudre avec cette technique et particulièrement ceux où il n'existe pas d'algorithme de résolution. Mais ce succès est aussi dû à l'approche particulière des problèmes qui est faite et que nous pouvons d'ailleurs considérer comme une méthode de programmation à part entière, programmation déclarative et non plus procédurale (cf chapitre II).

Cette approche, ainsi que nous le verrons au paragraphe suivant, se décompose en deux points. Le premier est l'établissement d'une base de connaissances nécessaire à la résolution d'un type de problème. Cela suppose au préalable le choix d'un formalisme de représentation de ces

connaissances propre au domaine d'application. Le deuxième porte sur le choix d'une structure de contrôle, également propre au domaine d'application, pour gérer ces connaissances.

Nous pensons qu'actuellement la réalisation d'un **MOTEUR D'INFERENCE GENERAL**, c'est-à-dire l'existence d'un formalisme de représentation des connaissances et d'une structure de contrôle universels, valables quelque soient les problèmes est utopique (utilisons-nous le même raisonnement quelque soit le problème ?). Cependant, la réalisation d'un système expert dédié à un domaine particulier, figé dans sa structure de contrôle et ses modes de représentation des connaissances incluant (généralement de manière procédurale) les propriétés du domaine, est devenue davantage un problème de développement ponctuel et de concertation avec les experts qu'un sujet de recherche en Intelligence Artificielle. Il faut noter toutefois que la formalisation d'un domaine peut être un problème très complexe.

Notre approche se veut intermédiaire. **EAQUE** est un moteur d'inférence général non pas pour tous domaines d'application, mais pour une certaine classe d'entre eux. EAQUE n'intègre aucune connaissance particulière à l'un de ces domaines, mais est caractérisé par un ensemble d'options qui leurs sont communes. Ces options correspondent à des choix de modes de représentation des connaissances et de structure de contrôle propres à notre classe de domaines d'application. Nous obtenons entre autres une flexibilité de la structure de contrôle que nous pouvons paramétrer tout en restant dans un mode de raisonnement donné. Ainsi, le retour arrière fait partie du mode de raisonnement, et différentes stratégies peuvent être intégrées selon les domaines.

Il est intéressant de signaler dès maintenant que

l'**INSTANTIATION** (rajout des connaissances nécessaires pour son application à un domaine, par exemple les propriétés de celui-ci) de ce moteur à un domaine particulier nécessite un travail non négligeable de la part de L'Ingénieur Cognitif, travail qui ne peut être fait que par lui. Ainsi pour ORDF et pour CALINT, deux instantiations d'EAQUE, il a fallu (dans chacun des deux cas) ajouter environ 20% au noyau central. De plus, plusieurs instantiations, correspondant à autant d'approches de résolution, sont possible pour un même problème, ce qui complique d'autant la tâche de l'Ingénieur Cognitif et de l'Expert. Nous pouvons raisonnablement en conclure que dans le cas d'un moteur complètement général, la part de travail de l'Ingénieur Cognitif serait telle que la notion même de généralité perdrait tout son sens.

Signalons pour clore cette introduction qu'un tel moteur est un système relativement complexe et que l'utilisation de certaines techniques récentes de l'Intelligence Artificielle a grandement facilité sa réalisation et sa mise au point. Ainsi, la structure de contrôle et les règles de production sont autant d'objets LRO de classes différentes et ce sont des envois de messages qui coordonnent l'ensemble.

:B: SYSTEMES EXPERTS ET STRUCTURES DE CONTROLE.

Il est difficile de donner une définition exacte d'un **SYSTEME EXPERT** tant il existe de différences d'un système à l'autre. Cependant, ils ont un certain nombre de caractéristiques communes.

- Ils s'appliquent particulièrement (mais non exclusivement) à des problèmes non totalement formalisés, pour lesquels on ne dispose pas d'algorithme de résolution (inexistant ou insatisfaisant) comme le diagnostic médical, mais où il existe une expertise humaine, des connaissances subjectives issues de l'expérience.
Ils peuvent être également utilisés pour des problèmes qui sont (en principe) totalement formalisés mais où il faut utiliser une expertise pour réduire l'aspect combinatoire de la résolution. C'est le cas des problèmes d'ordonnancement.

- Une des caractéristiques les plus fondamentales d'une système expert est sans doute la séparation entre les connaissances de résolution et les procédures chargées de les exploiter.
Les connaissances représentant le "savoir faire" de l'expert sont en principe exprimées sous forme déclarative, indépendamment les unes des autres (granules de connaissances). La mise à jour de cet ensemble de connaissances par l'expert (ajout, suppression, modification de nouvelles connaissances) est alors facile à réaliser.

- Il n'y a pas d'enchaînement prédéterminé de ces connaissances comme dans une procédure où tous les cas doivent être prévus à l'avance. Leur utilisation éventuelle dépend de l'état de la base de connaissances à un instant donné (programmation dirigée par les données).

- L'aspect explicatif est important, le système peut justifier la solution trouvée en citant successivement les connaissances utilisées.

- Les Systèmes Experts permettent également de modéliser des raisonnements approximatifs ou incertains, par exemple par l'introduction de coefficients d'importance, d'incertitude ou de vraisemblance associés aux diverses connaissances. Nous verrons au paragraphe III-2- comment peuvent être combinés de tels coefficients.

- Les propriétés du domaine sont généralement intégrées au système (pour des raisons d'efficacité), ce qui rend alors le système expert exclusivement dédié à son domaine d'application.

En Résumé, nous dirons qu'un système expert est composé de deux grandes parties, à savoir **UNE BASE DE CONNAISSANCES** et un **MECANISME D'EXPLOITATION** de cette base (nous trouvons également les termes moteur d'inférence, système cognitif, structure de contrôle).

Note : Une réflexion sur les différentes connaissances que nous manipulons et leur mode de représentation a été faite au chapitre -II- LRO. Nous allons maintenant (sections a et b) présenter une étude sur les mécanismes d'exploitation dûe à JP.LAURENT (LAURE 84). Nous verrons ensuite dans la section c comment cette étude a été reprise par JP.LAURENT et moi-même et comment nous avons affiné certains termes, pour mieux permettre d'expliquer le fonctionnement d'EAQUE.

-a- LA BASE DE CONNAISSANCES.

Une base de connaissances est constituée d'un ensemble de connaissances, les éléments de la base, de nature différente. Cette nature est liée, non pas à cette connaissance en elle-même, mais à l'utilisation qui en sera faite par le mécanisme d'exploitation. En effet, la proposition "le fer est un métal" peut aussi bien s'interpréter comme une connaissance descriptive, factuelle, indiquant que le fer est un métal, que comme une connaissance déductive, qui permet de déduire du fait "x est en fer" le nouveau fait "x est en métal".

Une base de connaissances est composée d'un ensemble d'**objets**, d'un ensemble d'**actions** et d'une **condition d'arrêt.**

-a-1- les objets

Les "objets" sont les éléments de la base de connaissances dont l'utilisation par le mécanisme d'exploitation correspond à une interprétation factuelle de la connaissance correspondante.
Les objets permettent de décrire le domaine d'application et le problème à résoudre.

Exemples : Réseaux Sémantiques, Triplets (objets, attribut, valeur)...

-a-2- les actions

Les "actions" sont les éléments de la base de connaissances dont l'utilisation par le mécanisme d'exploitation correspond à une interprétation déductive de la connaissance correspondante.
Les actions ont pour but principal la modification de la

base de connaissances (par exemple introduction, modification, suppression d'objet), dans le but de transformer l'état courant de la base en un état solution.

Citons pour exemple, la règle de production, qui est, actuellement, le formalisme le plus courant.

-a-3- la condition d'arrêt.

Elle correspond conceptuellement à une procédure, traduisant la condition d'arrêt, lancée après chaque cycle élémentaire du mécanisme d'exploitation, afin de déterminer si le problème est résolu ou non.

-b- LE MECANISME D'EXPLOITATION.

Nous ne parlerons pas ici des procédures d'exploitation associées aux éléments de la base de connaissances qui sont propres à chaque formalisme et que nous avons vues au chapitre II, mais uniquement des différentes structures de contrôle qui permettent de faire évoluer l'état initial de la base de connaissances vers un état solution.

-b-1- états et transitions.

On appelle **TRANSITION** le passage d'un état de la base de connaissances, que l'on appelera désormais état, à un autre état. C'est l'utilisation par le mécanisme d'exploitation d'une action (connaissance permettant de modifier la base de connaissances) appliquée à des éléments de la base de connaissances (objets par exemple). Dans le cas le plus simple où l'action s'applique sur un objet, la transition est caractérisée par l'état où elle s'applique, un couple

(objet , action) pris dans cet état et l'état qu'elle engendre.

Ainsi, dans un système de réécriture pour la simplification d'expressions mathématiques (cf CALINT chapitre V), le choix de l'objet "3Y + 4 + 5Y" et de la règle "K1X? + n'importe quoi + K2X? ⟹ (K1+K2)X? + n'importe quoi" permettra de déclencher une transition dont le résultat est un nouvel état où le premier objet est détruit et remplacé par le nouvel objet "8Y + 4".

-b-2- espace de recherche, cycle de contrôle.

L'espace de recherche est le graphe de tous les états implicitement définis par l'état initial et les successions des transitions possibles. Le but du mécanisme d'exploitation est de développer un sous arbre minimum contenant un état solution (on ne s'intéresse pas à la détermination de toutes les solutions). On appelle **cycle de base** ou **cycle de contrôle,** l'ensemble des opérations nécessaires au mécanisme d'exploitation pour générer un nouvel état. Il existe d'une manière générale cinq étapes.

étape 1 : Détermination de l'ensemble des états activables.
étape 2 : Choix d'état.
étape 3: Détermination de l'ensemble des transitions possibles.
étape 4 : Choix de transition.
étape 5 : Exécution.

Un état activable est un état à partir duquel il est possible d'appliquer une nouvelle transition (sinon on parle d'état "mort"). Pour un état activable donné, l'ensemble des transitions possibles est un ensemble de couples (objet, action).

Selon les domaines d'application, certaines étapes ci-dessus mentionnées peuvent ou non figurer. Les étapes existantes peuvent être implémentées de diverses manières. Ces facteurs déterminent différentes structures de contrôle, et une classification des mécanismes d'exploitation peut en être déduite.

-b-3- le choix d'état.

Au début de chaque cycle de contrôle se pose le problème du choix d'un état parmi les états activables (au départ l'état courant est l'état initial). Dans le cas où l'état courant est le dernier état généré, le problème ne se pose pas et les étapes 1 et 2 n'existent pas. Cependant, ceci n'est pas toujours possible car il est parfois nécessaire de restaurer des états antérieurs pour, par exemple, supprimer des effets incorrects.

La présence des étapes 1 et 2 dépend de la nature des transitions. Il existe des domaines où elles ne sont pas nécessaires. C'est le cas des systèmes monotones où les transitions ne modifient auncune information, ainsi une transition possible à un état le reste à un état suivant. Parcontre, dans le cas où les transitions sont non permutables et/ou non réversibles (c'est en particulier vrai pour les transitions supprimant des informations) ces étapes sont nécessaires.
Dans le cas où les étapes 1 et 2 sont présentes (ce qui permet déjà une première classification des différents moteurs entre eux) il existe de nombreuses possibilités d'effectuer le nouveau choix, en particulier pour le retour arrière : profondeur d'abord, retour arrière sélectif,... Elles dépendent évidemment du domaine d'application.

-b-4- le choix de transition.

Une transition est un couple (objet , action). Choisir une transition c'est donc choisir un couple (objet , action). Il existe plusieurs façons de le faire. Nous pouvons le déterminer globalement, parmi l'ensemble des couples possibles, c'est la **résolution de conflit.** Mais aussi puiqu'une transition se compose de deux éléments, l'objet et l'action, il est possible de décomposer le choix de transition en un choix d'objet et un choix d'action traités séparément. Nous avons alors deux cas de figure possibles : choix d'objet puis choix d'action, ou choix d'action puis choix d'objet.

-b-5- les différents types de moteur d'inférence.

En fonction des remarques des paragraphes précédents, LAURE(84) distingue six types de moteur:

EOA : choix d'Etat, choix d'Objet, choix d'Action.
EAO : choix d'Etat, choix d'Action, choix d'Objet.
EC : choix d'Etat, résolution de conflit.
C : résolution de conflit.
OA : Choix d'Objet, choix d'Action.
AO : Choix d'Action, choix d'Objet.

Exemple :

- EMYCIN est un moteur de type OA. L'objet courant est le dernier objet généré et, en cas d'impasse, il est choisi par un backtrack d'objet de type depth-first.

- OPS est un moteur de type C utilisant pour la résolution de conflit des métarègles.

- GARI est un moteur de type E-C. La résolution de conflit et le choix de la transition à rejeter lors d'un backtrack utilisent des notes associées aux règles de production.

Cette classification pourrait sûrement être complétée avec intérêt en faisant intervenir les différents formalismes de représentation des connaissances au niveau des "objets" et des "actions" (cf chapitre -II- LRO).

-c- REMARQUES.

Le formalisme que nous venons de voir est cependant trop général pour l'étude détaillée d'un moteur comme EAQUE. Nous avons ainsi, JP.Laurent et moi-même, redéfini plus précisément la notion de transition.

Nous prendrons pour exemple le cas suivant :
Soit l'objet barre défini par un ensemble de propriétés et leur valeur. Par exemple, la valeur fer pour la propriété matière associée à l'objet barre.
Soit la règle de production : RP : Si x est en fer alors x est en métal.
Une des actions possibles, dûe à un choix particulier d'utilisation des règles (modus ponens), est : (Rajouter "x est en métal"). Ce qui se traduira en mettant métal comme valeur de la propriété nature de x.
Le choix de cet objet et de cette action correspond à la transition (Rajouter "barre est en métal").

Avec le formalisme vu précédemment la transition est T = (objet , action), soit pour notre exemple T = (barre , RP).

En fait, la **transition** peut se définir plus précisément comme **un triplet (schéma d'action , éléments utiles , éléments modifiés).**

le schéma d'action SA :

C'est l'action au sens formel du terme, détachée de toute autre notion comme les conditions d'activation qui lui sont en principe attachées. Le terme schéma traduit le fait que cette action peut donner lieu à plusieurs instantiations différentes.
Dans le cas de notre exemple le SA est la partie droite de la règle.

les éléments utiles U :

Ce sont toutes les connaissances utilisées pour vérifier les conditions d'activation du SA et pour **l'instancier.**
exemples:
. règles de production (en fait partie gauche),
. éléments descriptifs du domaine et du problème : triplets, réseaux... (intervenant par exemple dans l'évaluation des prémisses d'une règle),
. procédures

les éléments modifiés M :

Ce sont les éléments modifiés (au sens large : modifications, suppressions, ajouts) lors de l'exécution de la transition.

Pour notre exemple:

T = (- Rajouter "nature(x)=métal" ,		SA
- matière(x)=fer		U
si matière(x)= fer alors nature(x)=métal ,		U
- nature(barre)=métal)	(ajout)	M

Un des intérêts de ce formalisme apparait bien dans le cas des règles de production. Il permet de considérer l'action en tant que telle, indépendamment de ses conditions d'activation (cf EAQUE).

Nous adopterons dorénavant le formalisme du triplet (SA , U, M) pour les transitions.

EAQUE

:A: HISTORIQUE
:B: CARACTERISTIQUES
:C: EXEMPLE INFORMEL

:A: HISTORIQUE.

Notre travail a pour origine les problèmes d'ordonnancement. Ce choix s'explique par la place prédominante de ces problèmes dans de nombreux et divers secteurs, et par le fait que bien qu'étudiés depuis de nombreuses années leur compléxité est telle qu'ils restent ouverts. Ainsi, l'option que nous avons choisie se veut résolument nouvelle par l'utilisation des techniques de l'Intelligence Artificielle et en particulier de l'approche Système Expert. Ce fut également notre volonté de considérer le problème, dès le début, dans son ensemble et non de réaliser une application dédiée à un cas particulier. Cette difficulté supplémentaire offre cependant plusieurs avantages. En particulier, cela permet de rendre externes au système les propriétés (souvent procédurales) du domaine étudié, et soulage d'autant le concepteur. Et surtout, l'accent est porté principalement sur les différentes connaissances et le raisonnement utilisés et non plus sur une heuristique particulière.

La présence de structures de données complexes nous a amené à réaliser LRO (cf chapitre II). Quant à la structure de contrôle, le choix s'est porté sur un moteur de type EOA (choix d'Etat, choix d'Objet, choix d'Action), certainement le plus commun des 6 types.
Les caractéristiques de LRO et de la structure EOA nous a incité à réaliser un moteur d'inférence général pour une classe plus large que les problèmes d'ordonnancement.

:B: CARACTERISTIQUES.

Eaque est un moteur d'inférence général pour une classe de domaines d'application définie par les points suivants:

-b-1- la base de connaissances

A la différence d'un système expert, elle vide puiqu'elle découle directement d'un domaine d'application.
EAQUE propose des formalismes de représentation des connaissances que l'on utilise pour l'écriture d'une base de connaissances particulières.
LRO est utilisé pour représenter les objets.
Le formalisme des règles de production est utilisé pour la représentation des actions, des critères de choix d'objet et des critères de choix d'action.

-b-2- le mécanisme d'exploitation.

Il est de type EOA. Il possède un mécanisme de restauration d'état et un système de retour arrière séléctif intégré au système.
Les procédures de choix d'objet, d'action et de l'état de restauration utilisent des fonctions d'évaluation heuristiques associées, pour le choix d'objet et d'action, à l'utilisation de règles de critères de choix (cf ci dessus).

-b-3- la classe de domaines.

EAQUE peut s'appliquer à tout domaine où la résolution correspond à une structure de type EOA (domaines où les actions sont non permutables et/ou non réversibles) et à

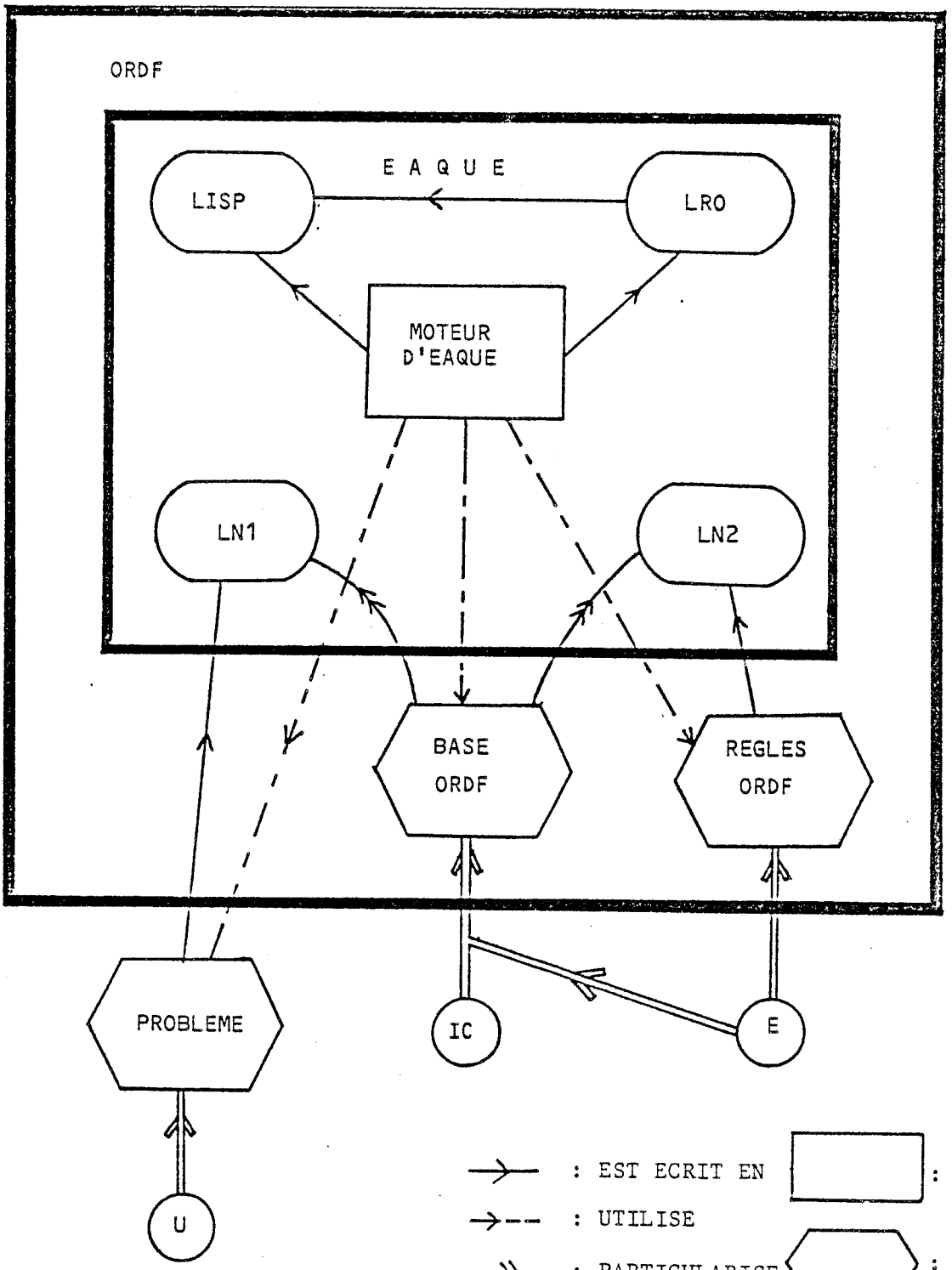
fortiori à une structure de type OA, mais aussi à une structure de type C. Cette dernière structure découle de la façon dont EAQUE modélise les objets et les actions. Ces domaines doivent être tels que leurs connaissances peuvent être représentées par les formalismes d'EAQUE.

-b-4- instantiation d'EAQUE.

Nous appellerons **instantiation** d'EAQUE à un domaine d'application le rajout d'un ensemble de règles, critères de choix d'objet et de choix d'action, et d'un ensemble de procédures et de données descriptives, nécessaires à cette application (cf figure III-1-). A chaque instantiation différente correspond un système expert différent.

FIGURE III-1-

Exemple de l'instantiation ORDF



:C: EXEMPLE INFORMEL.

Définissons, de manière tout à fait informelle, un exemple d'instantiation d'EAQUE afin de pouvoir illustrer la suite de ce chapitre.

Nous considérerons le problème de l'ordonnancement des tâches dans le domaine du bâtiment dans le cas où il est connu, pour chaque tâche, sa durée ainsi que le nombre et le type de ressources qu'elle utilise (cf ORDF chapitre IV).
Par exemple, le coulage d'une dalle est une tâche qui nécessite un matériel de coffrage de dalle, trois hommes et dure deux jours. Pendant deux jours, trois hommes et un coffrage dalle seront indisponibles.
Ces tâches sont partiellement ordonnées entre elles par des contraintes technologiques. Par exemple, on construit les murs avant de les peindre... Il existe des tâches qui ne sont liées par aucune contrainte technologique et qui peuvent être potentiellement réalisées en parallèle : ainsi les différentes piles d'un pont peuvent parfois être construites en même temps. Cependant, si ces tâches utilisent une (ou plusieurs) ressource identique et que la somme de leurs besoins est supérieure au nombre total d'unités disponibles de cette ressource, elles ne pourront rester en parallèle. Nous dirons alors que **ces tâches sont en conflit pour cette ressource** et constituent ce que nous appellerons un **conflit.** Elles devront être ordonnées. Le problème est de savoir comment ordonner ces tâches; de cet ordre dépendra, entre autres choses, la durée du projet.

LE DOMAINE D'EAQUE :

OBJETS, ACTIONS,

CONDITION D'ARRET, OBJECTIFS

:A: LA DESCRIPTION DU PROBLEME.
:B: LES OBJETS.
:C: LES ACTIONS.

:A: LA DESCRIPTION DU PROBLEME A TRAITER.

La description d'un problème est associé à un domaine particulier, la seule contribution que peut apporter EAQUE à ce niveau sont des squelettes d'analyse syntaxique.

Les sommets et les classes de sommets LRO sont le formalisme de représentation des connaissances descriptives du domaine d'application. Nous pouvons ainsi définir des Réseaux Sémantiques et des Frames.

L'Ingénieur Cognitif, avec l'aide de l'expert, définit, dans le cadre de l'instantiation d'EAQUE à un domaine particulier, les classes de sommets LRO nécessaires à la description du problème.

Pour notre exemple (cf III-2-D), il définit deux classes de sommets, les classes TACHE et RESSOURCE. La classe TACHE

possède les propriétés PRED (prédécesseurs), SUC (successeurs), RES (ressources utilisées), DTOT (date au plus tôt), DTARD (date au plus tard). La classe RESSOURCE a les propriétés NBRE (quantité de ressources disponibles), IMPORTANCE (permettant de hiérarchiser les ressources entre elles).

La description par l'Utilisateur d'un problème à résoudre se décompose en trois parties :

-1- la déclaration des sommets.

L'utilisateur déclare les sommets décrivant son problème avec les valeurs des propriétés associées.

En reprenant notre exemple, il déclare un sommet de la classe TACHE par tâche. Par exemple, la tâche Fouille1 a comme prédécesseur Début, comme successeur Fondation1, a un besoin en ressources d'une pelle et de deux hommes, et dure deux jours. Il définit ainsi **le graphe de conception** de l'ouvrage. Il définit également les ressources en associant à chaque ressource un sommet de la classe RESSOURCE.
Il déclarera Fouille1 de la manière suivante:
Fouille1 TACHE PRED = (Début) / RES = ((1 Pelle)(2 Homme)) ;
Il déclarera Pelle de la manière suivante :
Pelle RESSOURCE NBRE = 1 / IMPORTANCE = 3 ;

SYNTAXE

La description d'un problème est un ensemble de déclarations de sommets.

Chaque déclaration doit respecter la syntaxe suivante (syntaxe par défaut d'EAQUE) :

cf FIGURE -III-2-

-2- la condition d'arrêt

La condition d'arrêt est **une conjonction d'expressions** dont la valeur VRAI traduit la fin de la résolution du problème. Elle dépend du domaine et du problème considérés.

Par défaut, la condition d'arrêt est celle d'EAQUE, à savoir l'inexistence d'"objet EAQUE", au sens où nous allons le définir dans le paragraphe :B:.

Pour notre exemple, la condition d'arrêt par défaut d'EAQUE est celle de notre domaine d'application. En effet, nous verrons au chapitre -IV- que les objets représentent des tâches en conflit, et ne plus avoir d'objet signifie ne plus avoir de tâches en conflit, donc que le problème est résolu.

SYNTAXE

La définition de la condition d'arrêt dans le cas où l'on n'utilise pas celle d'EAQUE, doit respecter la syntaxe suivante :

cf FIGURE -III-3-

-3- les objectifs

A la notion de condition d'arrêt nous avons rajouté celle d'**OBJECTIFS.**
Les objectifs permettent de définir des contraintes pour la résolution du problème qui doivent rester vérifiées pour tous les cycles de contrôle. Cela permet d'éviter une dispersion de la résolution et de forcer un retour-arrière, si cela s'avérait nécessaire, le plus tôt possible. Par exemple, une fonction croissante à chaque cycle doit rester inférieure à une borne donnée.

Pour notre exemple, si l'on veut que le projet final se termine avant une certaine date T, c'est à dire que la date au plus tard de la dernière tâche soit inférieure à T, l'objectif est d'avoir, pour chaque cycle, cette date effectivement inférieure à T (ordonner des tâches ne peut qu'augmenter positivement la date au plus tard de la dernière tâche).

Dans le cas où il existe plusieurs objectifs, chaque cycle de contrôle doit les vérifier tous.

A chaque objectif est associé un coefficient de relâchement compris entre 0 et 10. La note 0 indique un objectif obligatoire. Le fait de ne pas le respecter entraînera automatiquement une restauration à un état antérieur. Si les notes sont strictement positives, il est possible de relâcher (ce qui ne veut pas dire supprimer) ces objectifs afin de ne pas faire de backtrack.
Dans le cas d'une fonction dépassant une borne, relâcher l'objectif peut se traduire par la modification de la borne.

Par défaut il n'existe pas d'objectif, c'est à dire pas de contrainte.
SYNTAXE

La définition des objectifs doit respecter la syntaxe suivante:

cf FIGURE -III-4-

LN1 : Langage Naturel 1 :

LN1 est le langage mis à la disposition de l'utilisateur pour décrire son problème.
Un problème est un ensemble de déclarations d'objets LRO. Elles sont de la forme :

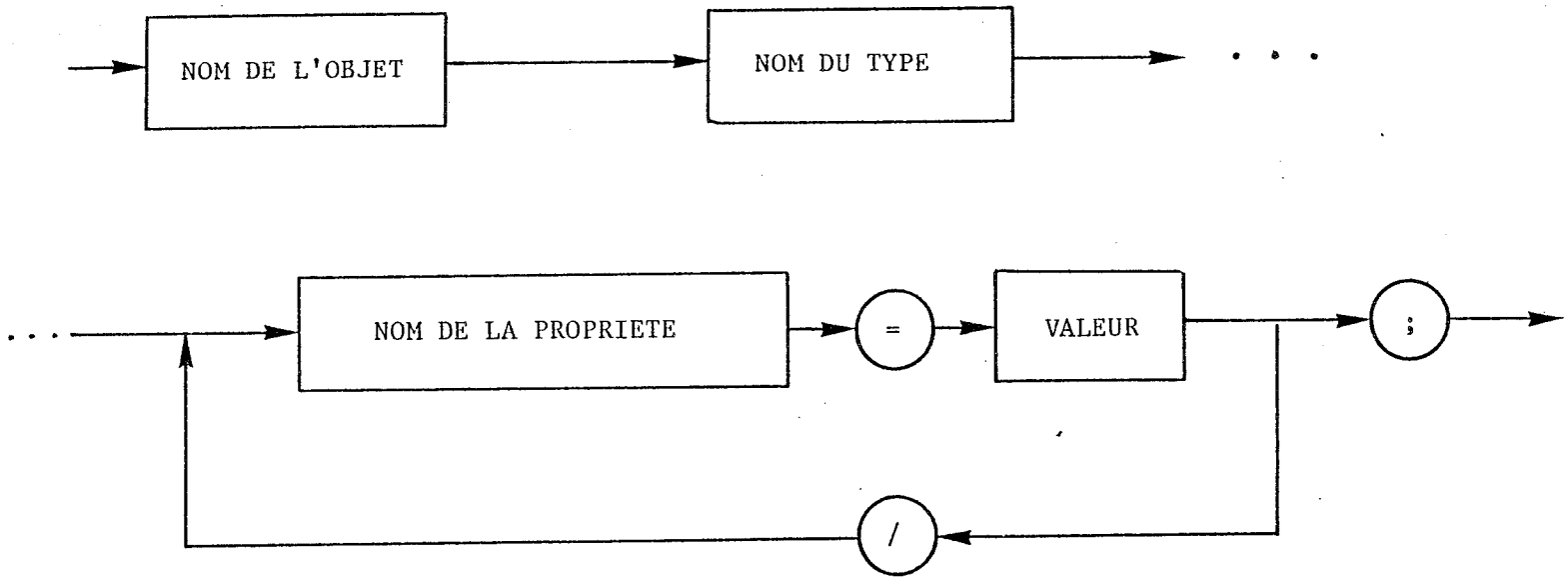
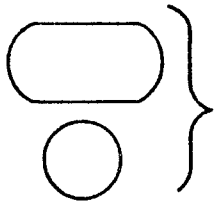


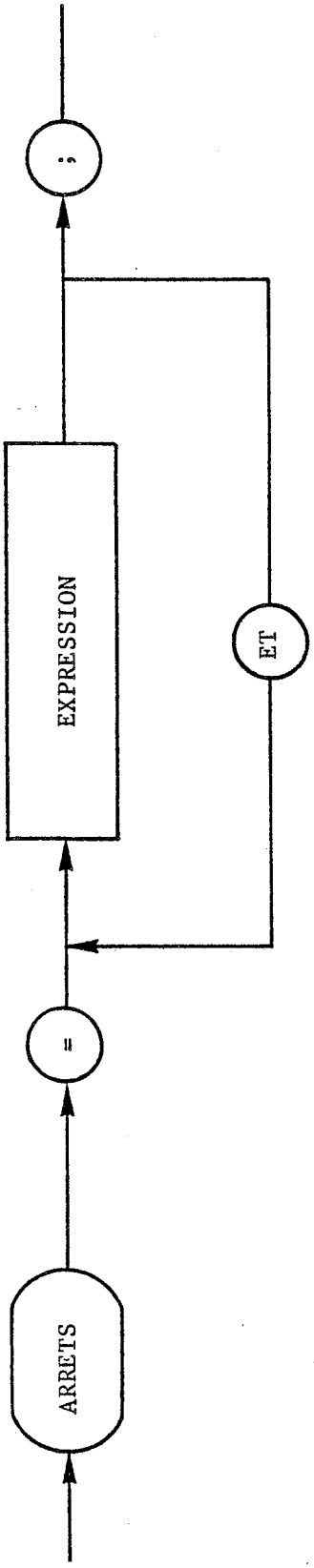
FIGURE -III-2-

Les noms du type et de la propriété dépendent du domaine d'application considéré. Leurs domaines de valeurs seront définis par l'Ingénieur Cognitif.



: mot réservé

FIGURE -III-3-



OBJECTIFS

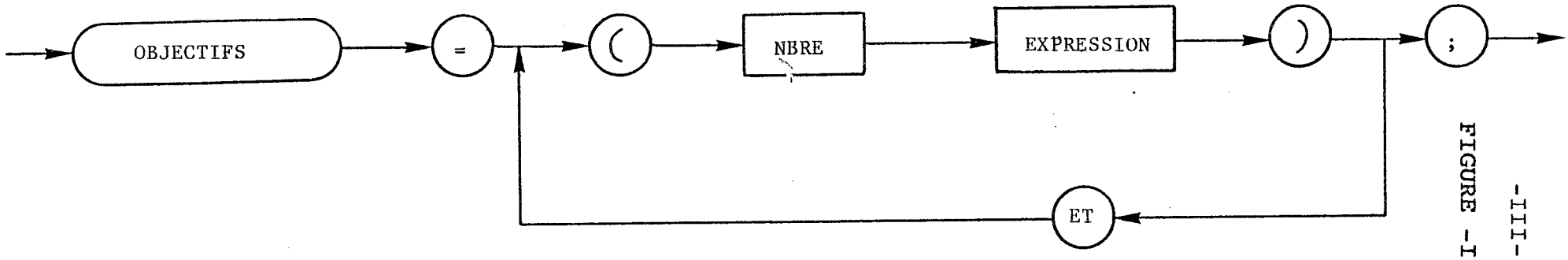


FIGURE -III-4-

NBRE représente le degré de relâchement de l'objectif pris sur 1 échelle de 0 à 10

:B: LES OBJETS.

Note : Le mot objet a été employé dans trois contextes différents, ce qui s'explique par l'indépendance des travaux menés.
Il existe :
- l'objet LRO, représentation élémentaire d'une connaissance pour LRO (cf chapitre -II-).
- l'objet de LAURE(84), représentant une connaissance descriptive.
- l'objet d'EAQUE que nous allons maintenant définir.

Dorénavant l'emploi du mot **objet** sans qualificatif (LRO, LAURE, EAQUE) désignera un **objet EAQUE.**

Signalons dès maintenant, pour une meilleure compréhension de ce chapitre et des suivants, que tous les éléments de la base de connaissances, règles de production et objet (EAQUE) y compris, sont des objets LRO.

Réaliser un moteur général pose le difficile problème de la représentation et de la manipulation d'entités qui ne peuvent être qu'abstraites. Les définitions de l'objet EAQUE et de l'action EAQUE n'ont pu se faire que par "tatonnements" successifs, se modifiant l'un l'autre.

L'objet est pour EAQUE le seul élément de connaissance descriptif qu'il puisse connaître, et il le connaît uniquement par son nom et non par sa structure. C'est cette dernière qui lui donnera un sens concret lors d'une instantiation.

La notion d'objet pour EAQUE est définie, pour une instantiation donnée, par l'Ingénieur Cognitif. Ce peut être un objet LRO ou un ensemble d'objets LRO répondant à

certaines caractéristiques. Les objets ne sont pas forcément déduits directement de la description du problème par l'Ingénieur Cognitif. Cette définition n'est pas neutre en ce qui concerne la stratégie de résolution qui sera appliquée.

Le rôle principal de l'objet est de réunir un ensemble d'informations nécessaires à l'application d'une action portant sur cet objet. Cela correspond particulièrement bien au cas où l'objet représente un sous-problème à résoudre.

Un objet définit en fait les domaines d'instantiation possibles pour les variables des actions EAQUE.
Choisir un objet c'est donc, pour EAQUE définir des ensembles d'instantiation pour ces actions. Sémantiquement, ceci revient à choisir quelle partie de la base de connaissances va être modifiée par la prochaine action.

Dans notre exemple, l'objet n'est pas déduit directement de la description du problème. Chaque objet représente un ensemble de tâches en conflit pour une ressource. Cet ensemble est le seul domaine d'instantiation associé à un objet pour les variables des actions EAQUE qui sont donc des variables de type TACHE.

Remarques :
-1- Il est toujours possible de considérer un seul objet représentant l'ensemble de toutes les instantiations possibles. Le problème du choix de l'objet (cf -III-4-) ne se pose alors pas et nous sommes dans le cas du moteur de type C.
-2- Il y a un problème à résoudre, au sens EAQUE, s'il existe au moins un objet. Nous pouvons toujours, dans le cas où il n'existerait pas de critères de choix d'objet pour le domaine d'application choisi, nous ramener au cas de la remarque précédente.

:C: LES ACTIONS.

La base de connaissances contient, en plus de la description du problème, une première classe de règles. Ces règles qui représenteraient les actions au sens de LAURE(84) et que nous appellerons désormais des **règles actions**, doivent être vues en fait selon le formalisme T = (SA , U , M) exposé dans -III-1:B:c. Le schéma d'action, SA, est la partie droite d'une règle; La partie gauche, qui correspond aux conditions d'activation du schéma d'action, est incluse dans les éléments utilisés, U.
En général un schéma d'action contient des variables et peut s'instancier de plusieurs manières.

NOUS APPELLERONS ACTION EAQUE, QUE NOUS DESIGNERONS DESORMAIS SOUS LE TERME D'ACTION, UN SCHEMA D'ACTION INSTANCIE.

Ainsi, dans notre exemple, "METTRE T1 AVANT T2", où T1 et T2 sont deux tâches en conflit quelconques (T1 et T2 sont des variables), est un schéma d'action; "METTRE fouilles1 AVANT fouille2" est une action (T1 et T2 ont été respectivement instanciées par fouilles1 et fouilles2).

Remarques :
- On pourrait résumer simplement cette distinction fondamentale de la façon suivante : un SA est une procédure, une action est un appel de procédure où les paramètres à passer sont définis.
- La notion d'action pour EAQUE recouvre davantage le concept d'action candidate. Ce qui est somme tout naturel puisque EAQUE est la structure de contrôle d'un système, qu'il s'agisse d'ORDF, de CALINT, d'EALOG ou autre.

DESCRIPTION DES REGLES ACTIONS

-1- BUT

Les règles actions indiquent sous quelles conditions (en partie gauche de la règle) il est préférable de faire ou de ne pas faire telle ou telle action. Il existe donc, dans les règles actions, deux types de règles, les premières pour augmenter l'intérêt d'une action, les deuxièmes pour diminuer cet intérêt. Cela est dû à la fois à la politique de choix de l'action à appliquer (cf le système de notation -III-4- et -III-5-) et parce qu'il existe des domaines où ne pas préférer une action n'est pas forcément équivalent à vouloir préférer la "négation" de cette action, lorsqu'elle existe.

-2- LE CIR

Les règles actions possèdent un **coefficient d'importance**, le **CIR** (Coefficient d'Importance de la Règle). Les CIR permettent de pondérer ces règles actions selon leur importance, ils interviendront bien sûr pour le choix d'action.

En continuant notre exemple, nous pouvons avoir comme règle, la règle suivante :
"Si T1 et T2 sont deux tâches en conflit et si la date au plus tôt de T1 est inférieure à la date au plus tôt de T2 ALORS il est préférable, avec une importance de 5, de mettre T1 avant T2".

-3- première SYNTAXE

Les règles actions sont de la forme :

TYPE nom de la règle **CIR**
SI conditions
ALORS IL EST PREFERABLE DE schéma d'action ;
IL EST PREFERABLE DE NE PAS

Le type est ici règle action. Les CIR varient de 0 à 10. Une règle ayant un CIR de 0 n'aura aucune influence sur toutes les actions implicitement associées à cette règle.
Les conditions et le schéma d'action dépendent évidemment du domaine d'application.

-4- LES VARIABLES

Une régle action peut posséder deux types de variables **les variables actions** et **les variables locales.**

Rappelons qu'EAQUE est de type EOA, le choix d'objet précède donc celui de l'action. L'objet définit les domaines d'instantiations pour les variables des actions EAQUE. L'objet EAQUE est un objet LRO, il en a donc la structure (ensemble de propriétés). Les actions portent sur l'objet choisi et sur ce qui le définit, à savoir ses propriétés.

-a- les variables actions.

Ce sont les variables d'une règle action telles qu'à chacune de leur instance correspond une règle instanciée différente, et donc un schéma d'action instancié différent, une action différente.
Les domaines d'instantiation de ces variables sont définis par l'objet précédemment choisi. Ce sont des propriétés de cet objet. Ces propriétés d'instantiation ne peuvent être

connues à l'avance par EAQUE, elles doivent donc être déclarées pour chaque règle action à l'aide de l'instruction "DOMAINE VARIABLES" qui associe à chaque variable action la propriété d'instantiation correspondante.

Ces variables actions s'écrivent **?nom de variable.**

Dans le cadre de notre exemple, chaque objet a la propriété CONFLIT dont la valeur est un ensemble de sommets en conflit. C'est la propriété d'instantiation.
En reprenant la règle précédemment citée :
"Si T1 et T2 sont deux tâches en conflit et si la date au plus tôt de T1 est inférieure à la date au plus tôt de T2 ALORS il est préférable, avec une importance de 5, de mettre T1 avant T2."
T1 et T2 sont deux variables actions, elles s'écriront ?T1 et ?T2. Nous indiquons à la règle que ?T1 et ?T2 vont s'instancier sur la valeur de CONFLIT, c'est-à-dire que le domaine des variables ?T1 et ?T2 est CONFLIT, en écrivant "DOMAINE VARIABLES :: ?T1 = CONFLIT / ?T2 = CONFLIT"
Notre règle devient :

```
REGLE ACTION RA1 : CIR 5 :
DOMAINE VARIABLES :: ?T1 = CONFLIT /
                     ?T2 = CONFLIT
SI DTOT(?T1) ² DTOT(?T2)
ALORS IL EST PREFERABLE DE METTRE ?T1 AVANT ?T2 ;
```

-b- les variables locales.

Chaque instance de règle s'exécute dans un environnement qui lui est propre. L'utilisateur peut dans ce contexte définir et utiliser deux types de variables prédéfinies, gérées par EAQUE : **les variables locales simples** notées **nom de variable?**, et **les variables locales segments** notées **nom de variable*?.**

Une variable locale simple ne peut prendre comme valeur qu'un seul symbole ou une expression parenthésée, alors qu'une variable locale segment peut prendre comme valeur n'importe quelle expression y compris l'expression vide.

Il existe plusieurs opérations possibles sur ces variables, nous en citerons deux :

- l'unification.

C'est l'unification de deux expressions pouvant contenir des variables simples et aussi, pour l'une d'entre elles seulement, des variables segments. Les listes de substitutions résultantes sont incorporées au contexte, ce qui permettra d'instancier, le cas échéant, toute nouvelle occurence de ces variables.

Note : Les instantiations, à partir d'une liste de substitutions, des variables simples et des variables segments se font de manières différentes. La variable simple est remplacée par la valeur associée, la variable segment, dont la valeur associée est toujours parenthésée, voit son premier niveau de parenthésage supprimé.

Dans l'exemple que nous avons choisi pour illustrer ce chapitre, nous n'avons utilisé à aucun moment les variables locales, c'est pourquoi nous considèrerons exceptionnellement l'exemple de CALINT (cf chapitre -V-).

L'unification avec des variables locales est particulièrement bien adaptée pour la manipulation d'expressions mathématiques.
Ainsi :
Les expressions (X? + Y?) et (a + (b * c)) s'unifient et retournent la liste de substitutions ((X? a)(Y? (b * c)).
les expressions (X*? + Y*?) et (a + (b * c)) s'unifient et

retournent la liste de substitutions ((X*? (a))(Y*? ((b * c))).
Mais les expressions (X? + Y?) et (2 * a + c * b) ne s'unifient pas, alors que les expressions (X*? + Y*?) et (2 * a + c * b) s'unifient et retournent la liste de substitutions ((X*? (2 * a)) (Y*? (c * b))).
Les expressions (X*? c Y*?) et (a * b * c) s'unifient et retournent la liste de substitutions ((X*? (a * b)) (Y*? ())).
Comme nous le montre ce dernier exemple, il est facile de voir si une expression **contient** un caractère ou une expression quelconque. Ainsi une expression E est une somme si elle s'unifie avec l'expression (X*? + Y*?).
Terminons par l'exemple suivant : une expression E est un polynôme du premier degré si elle s'unifie avec l'expression (a? * x? + b?), et si la valeur de x? correspond à une variable au sens mathématique du terme.

- l'affectation.

L'instruction := permet de mettre à jour la valeur d'une variable, ou de la définir avec une valeur initiale au niveau du contexte de l'instance courante de la règle si elle n'a pas déjà été utilisée

-5- SYNTAXE.

Il existe deux syntaxes pour l'écriture d'une même règle action, l'une étant une forme réduite.

cf FIGURES -III-5- ET -III-6-.

FIGURE -III-5-

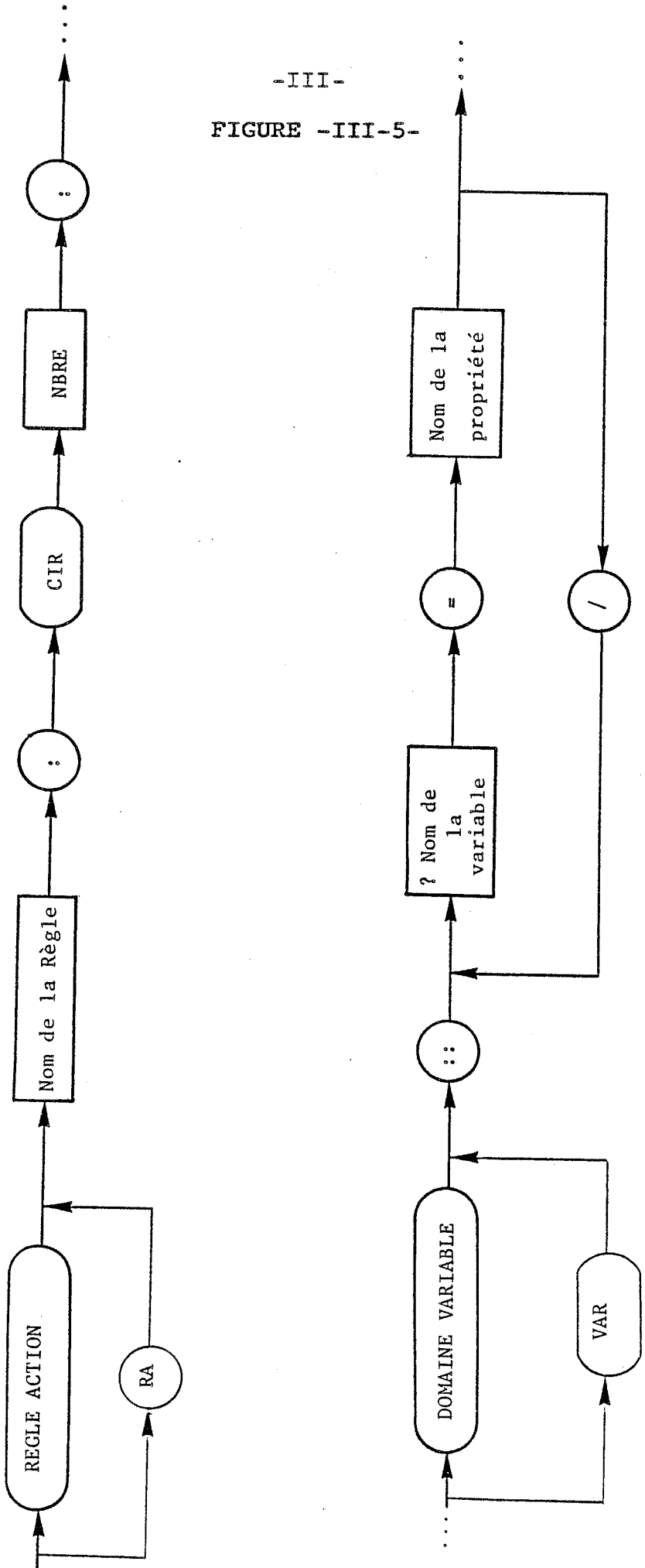
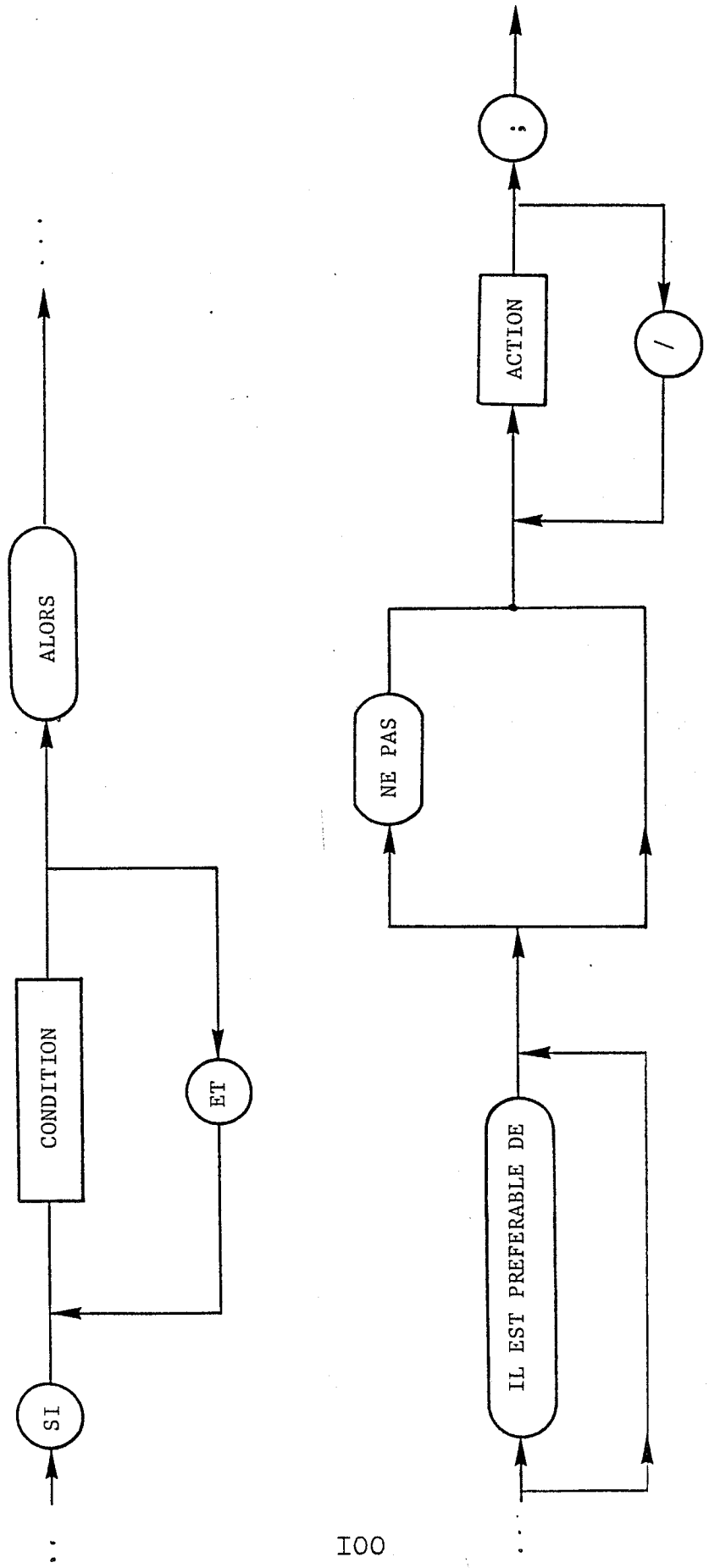

REGLE ACTION
RA
Nom de la Règle
:
CIR
NBRE
:
...
DOMAINE VARIABLE
VAR
::
? Nom de la variable
=
/
Nom de la propriété
...

FIGURE -III-6-



STRUCTURE GENERALE DU MOTEUR

AUTRES CARACTERISTIQUES

:A: MOTEUR.

-1- LES CONTRADICTIONS.

Une **CONTRADICTION** traduit un mauvais fonctionnement du moteur. Les causes peuvent être multiples, objectifs non vérifiés, inexistence d'action applicable à un état donné, conséquences de l'éxécution d'une action,... L'apparition d'une contradiction fait passer le moteur d'une structure OA à une structure EOA.

-2- LA PROCEDURE EAQUE.

Le fonctionnement du moteur EAQUE peut se résumer par la procédure suivante :

PROCEDURE EAQUE :

TANT QUE la condition d'arrêt n'est pas vérifiée **ET**
que les objectifs sont respectés **REPETER**
! choix d'objet
! choix d'action
! exécuter l'action et propager les conséquences

SI il existe des contradictions
ALORS
! résoudre ces contradictions
! EAQUE

FIN.

EAQUE peut être également vu comme un moteur de type OA tant qu'il n'y a pas de contradiction, de type EOA dès qu'il en apparaît une. Nous retrouvons le "focus of attention" au niveau du choix de l'état en prenant comme état courant le dernier état généré.

Remarque : Nous pouvons maintenant revenir sur la condition d'arrêt par défaut d'EAQUE.Pour EAQUE, un état sans contradiction et n'ayant plus d'objet, et donc à fortiori plus d'action applicable, est un état "stable", un état solution.

:B:- AUTRES CARACTERISTIQUES.

Parmi les autres caractéristiques d'EAQUE nous pouvons citer entre autres cinq points :

-1- EAQUE est un moteur avec variables.
-2- EAQUE offre la possibilité d'utiliser l'unification avec variables segments pour l'écriture des règles.
-3- EAQUE contient trois types de règles : les règles actions, les règles de choix d'objet et les règles de choix d'action (de choix de règles actions).
-4- EAQUE possède un niveau méta procédural, avec les fonctions de notation, et un niveau méta déclaratif avec des méta-règles : les règles de choix d'action (de choix de règles action).

-5- l'incertain et l'important.

La modélisation de l'incertain ou de l'important est un des sujets de l'Intelligence Artificielle les plus intéressants, surtout pour un informaticien qui ne se trouve plus ainsi soumis à la "déchirante" logique booléenne.
Le raisonnement de l'expert est basé sur des informations généralement incertaines, incomplètes et parfois incompatibles (si e1 et e2 sont vrais alors e3 est parfois, souvent, presque jamais, vrai). Il s'est donc avéré nécessaire de modéliser l'incertain. La démarche la plus naturelle consiste à attribuer à chaque événement, conseil, action, objet, un coefficient selon sa plausibilité variant entre deux bornes pour le vrai et le faux. Pendant longtemps, la théorie des probabilités est restée la seule approche pour la détermination de ces coefficients et de celui de leur combinaison. Nous en retrouvons une "trace" dans MYCIN pour le calcul de certains coefficients de vraisemblance : cv=c1+c2-c1*c2. Cependant, cette théorie ne permet pas d'avoir des degrés de plausibilité faibles pour

un événement et son contraire : p(a)+(¬a)=1. C'est pourquoi d'autres approches ont été développées soit directement pour des systèmes experts comme MYCIN, ou issues de théories comme celle du crédible et du plausible (Shafer) ou celle des sous ensembles flous (Zadeh).
Prenons un exemple :
Soit la règle d'inférence E1 ET E2 ⟹ E3 de plausibilité p, sachant que les plausibilités de E1 et E2 sont respectivement p1 et p2. La plausibilité de E3, p3, est pour Zadeh le minimum de p1, p2 et p3; alors que pour Shortliffe, Coulon, Kayser c'est une fonction d'affaiblissement de p1, p2 et p (MYCIN : p3=p*min(p1,p2)).
Actuellement la meilleure façon de juger une approche d'une autre est l'expérimentation.

Le système que nous avons choisi est basé non pas sur l'incertain, mais sur l'important. En effet, pour nous le problème est de choisir un objet (une action ou un état) parmi d'autres objets (actions ou états) dont l'existence n'est pas sujette à caution. Etant donné qu'ils sont tous candidats potentiels, l'objet retenu sera celui qui aura obtenu la meilleure note. Il est donc associé à chaque objet une note. Cette note est fonction de sa note courante, initialement nulle, et de la note associée à la règle choix d'objet appliquée à cet objet. Le rôle de la règle est de surnoter ou dénoter (il existe deux types de règles choix d'objet) la note de l'objet selon une fonction par défaut redéfinissable par l'Ingénieur Cognitif.
Pour surnoter:
note(objet) = note(objet) + F(note(objet) , note(règle))
Pour dénoter:
note(objet) = note (objet) - G(note(objet) , note(règle))
La présence d'une note éliminatoire et d'une note de passage permet de restreindre l'ensemble des candidats aux meilleurs éléments susceptibles d'être choisis et d'arrêter dès que possible la notation.

CHOIX D'OBJET

CHOIX D'ACTION

EXECUTION

:A: LE CHOIX D'OBJET.
:B: LE CHOIX D'ACTION.
:C: EXECUTION.

:A: LE CHOIX D'OBJET.

Résumé : Le choix d'objet s'effectue par l'application conjuguée des régles de choix d'objet et de fonctions d'évaluation heuristiques.

-1- BUT, LE CIO.

Au début de chaque cycle de contrôle se pose le problème du choix d'un objet parmi d'autres objets (structure EOA). La détermination de l'ensemble des objets (en particulier initialement) et/ou sa mise à jour à chaque cycle, est à la charge de l'Ingénieur Cognitif. Tous les objets (s'il y en a qu'un seul, le problème ne se pose pas) sont, à priori, autant de candidats possibles. C'est pourquoi **l'objet choisi sera l'objet ayant obtenu la plus forte note.** A cette fin, il est associé à chaque objet une note, un **CIO**, Coefficient d'Importance de l'Objet. Le CIO d'un objet peut être augmenté ou diminué selon son intérêt dans le contexte

courant, à l'aide de **CRITERES DE CHOIX D'OBJET.** Ces critères de choix d'objet sont représentés sous forme de **règles de production** que nous appellerons **REGLES OBJETS.**

Pour notre exemple, nous avons à choisir, à chaque cycle, entre plusieurs conflits, à priori équivalents. Devons-nous commencer par résoudre le conflit (fouilles1 fouilles2 fouilles3) avant le conflit (mur1 mur5 remblai3)? Le choix se fera à l'aide de critères qui indiquent, par exemple, de préférer entre deux conflits le conflit qui contient le plus de tâches.

-2- LES REGLES OBJETS.

Notes :

-1- Il serait possible de considérer les règles objets selon le formalisme exposé dans -III-1-:B:c en prenant une restriction de la base de connaissances, mais l'utilisation qui est faite des règles objets (différente des règles actions) ne nécessite pas un tel formalisme.

-2- Les règles actions et les règles objets ont de nombreux points communs (pour des raisons évidentes d'implémentation) dont une syntaxe équivalente. Mais à la différence des règles actions, EAQUE "connait" la notion d'objet et de partie droite d'une règle objet, car les règles objets font partie de la structure de contrôle puisqu'elles définissent le choix d'objet.

-2-1- but.

Les règles objets indiquent sous quelles conditions (en partie gauche de la règle) il faut surnoter ou dénoter tel ou tel objet. Il existe donc deux types de règles objets, les premières pour augmenter le CIO d'un objet (son intérêt en tant que candidat potentiel), les deuxièmes pour

diminuer ce CIO.

-2-2- le cir.

Les règles objets possèdent un CIR (Coefficient d'Importance de la Règle) qui permettra de "moduler" la surnotation ou la dénotation des objets.

Pour notre exemple, nous pouvons avoir une règle objet telle que :
"SI C1 et C2 sont deux conflits et si C1 a plus de tâches que C2 ALORS, avec une importance de 2, C1 est à préférer à C2, C'est-à-dire augmenter le CIO de C1 et diminuer celui de C2.

-2-3- première syntaxe.

Les règles objets sont de la forme :

TYPE nom de la règle **CIR**
SI conditions
ALORS PG ;

Le type est ici règle objet. Les CIR varient de 0 à 10. Une règle de CIR 0 n'aura, à priori, aucune influence sur les CIO des objets.

-a- les conditions

Les conditions dépendent évidemment de la sémantique des objets, EAQUE ne s'intéresse qu'à leur évaluation booléenne.

-b- la partie gauche (PG)

Les règles objets font partie de la structure de contrôle puisqu'elles définissent le choix d'objet. EAQUE "connait"

donc les différentes parties gauches (PG) qui peuvent apparaître dans une règle objet, ainsi que leur interprétation.
Il existe quatre types de PG différentes qui sont :
- préférer un objet. Ce qui se traduira par une augmentation de son CIO.
- ne pas préférer un objet. Ce qui se traduira par une diminution de son CIO.
- préférer un objet à un autre. Ce qui se traduira par une augmentation du CIO du premier objet et une diminution du CIO du second.
- ne pas préférer un objet à un autre. Ce qui se traduira par une diminution du CIO du premier objet et une augmentation du CIO du second.

Nous pouvons anticiper et dire que la fonction de notation des objets peut inclure des particularités du domaine, ce qui permet de pouvoir différencier la préférence d'un objet à un autre de sa négation, quant à leur influence sur les CIO.

-2-4- les variables

Tout comme les règles actions (cf les règles actions -III-4) nous avons deux types de variables, **les variables objets** qui jouent le rôle qu'avaient les variables actions pour les régles actions, et les variables locales. Cependant, les règles objets faisant partie de la structure de contrôle, EAQUE connait **le domaine d'instantiation des variables objets, c'est l'ensemble des objets existants pour le cycle courant.**

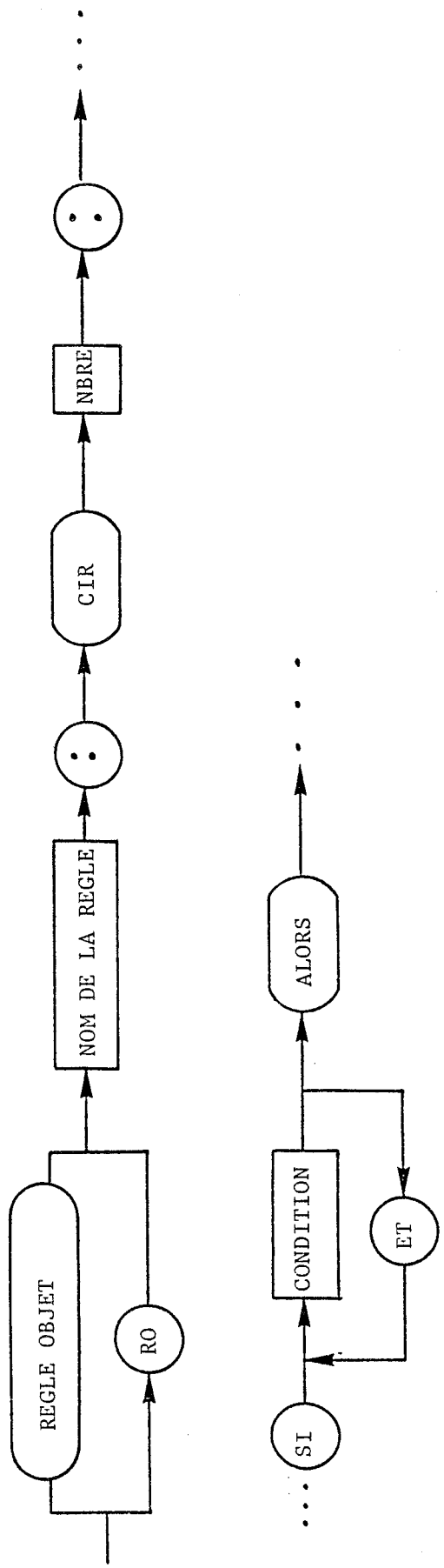
Dans le cadre de notre exemple, une règle possible est :
REGLE OBJET RO1 : CIR 2 :
SI NBRE_DE_SOMMETS ?C1 ³ NBRE_DE_SOMMETS ?C2
ALORS ?C1 EST A PREFERER A ?C2 ;

-2-5- syntaxe

Il existe deux syntaxes pour l'écriture d'une même règle objet, la deuxième étant une forme contractée de la première.

cf FIGURES -III-7- ET -III-8-

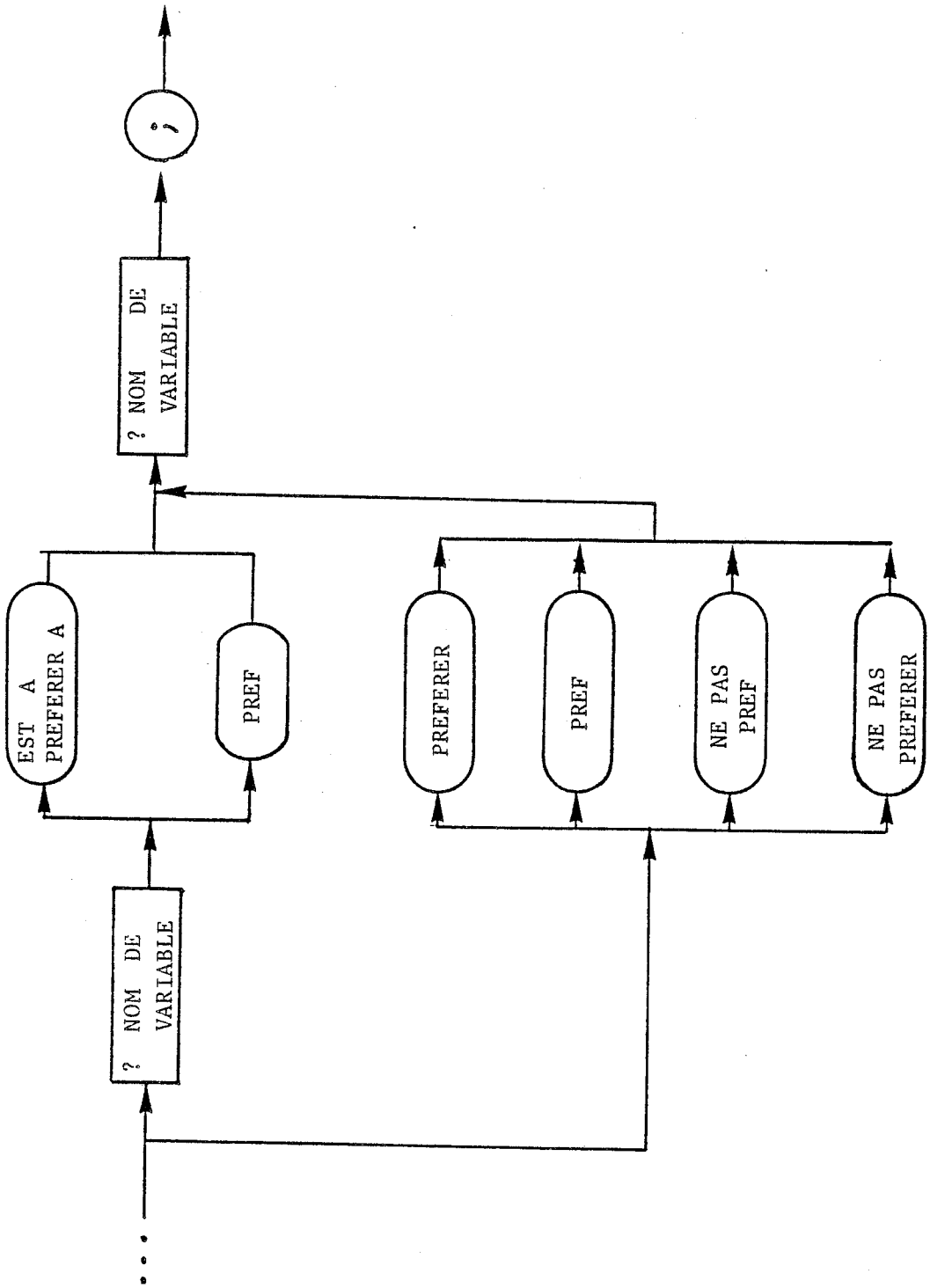
FIGURE -III-7-



LN 2 : Langage Naturel 2.

LN 2 est le langage mis à la disposition de l'utilisateur pour décrire les différents types de règles, et en particulier les règles objets. Les règles objets sont de la forme :

FIGURE -III-8-



NBRE est le Coefficient d'Importance de la Règle (CIR) pris sur 1 échelle de 0 à 10.

-3- LE CHOIX D'OBJET

-3-1- les seuils min et max

L'utilisation des règles objets est la suivante : si les conditions de la régle sont satisfaites, alors modifier, selon les indications de la partie gauche, les CIO des objets instances des variables objets. EAQUE ne s'arrête pas, à priori, sur la première instance de règle objet activable (dont les conditions sont vérifiées), mais va considérer également les autres règles objets. Cependant, il est possible de limiter EAQUE dans sa notation des objets par deux seuils, un CIO minimum et un CIO maximum. Ainsi, dès qu'un objet a son CIO supérieur au CIO maximum, l'évaluation des objets s'arrête, et c'est l'objet choisi. Si un objet a son CIO inférieur au CIO minimum, il est n'est plus considéré comme un candidat possible, EAQUE le rejette. Ces deux CIO ont des valeurs par défaut (10 et -10) en liaison avec les fonctions de notation par défaut. Ils sont évidemment redéfinissables par l'Ingénieur Cognitif. Ainsi, il est tout à fait possible d'imposer à EAQUE de s'arrêter dès l'application de la première règle activable, ou de lui faire appliquer toutes les règles objets.

-3-2- ordre des objets et des règles

Il est évident, d'après ce que nous venons de voir au point précédent, que l'ordre dans lequel les objets et les règles objets vont être considérés a une importance. Afin d'obtenir le plus rapidement possible un objet ayant un CIO supérieur au CIO maximum, les règles objets sont appliquées selon leur CIR décroissant, en essayant, pour une règle donnée toutes ses instances possibles. De même, après chaque cycle de contrôle, les objets sont ordonnés selon leur CIO croissant, afin que les premières instances des variables objets

portent sur les objets de plus fort CIO (un objet ayant un fort CIO est plus susceptible d'être choisi).

-3-3- la procédure du choix d'objet

Au début de chaque cycle, EAQUE détermine l'ensemble des objets. S'il n'existe qu'un seul objet, il est pris par défaut et la procédure s'arrête. Sinon ils sont tous considérés, sur le même plan, comme autant de candidats possibles. Ce qui se traduit par une mise à 0 de tous les CIO (il n'est pas possible pour un moteur général de considérer, pour des objets dont l'existence est antérieure de plusieurs cycles, leur ancien CIO. En effet cela dépend du domaine d'application).

*TANT QUE il n'existe pas d'objet dont le CIO est supérieur au CIO maximum ET qu'il reste des règles à appliquer REPETER
*Prendre la règle de plus fort CIR
*POUR chaque instance vraie de cette règle sur la liste des objets REPETER
*Noter l'objet ou les objets (surnoter, dénoter)
*SI le CIO est supérieur au CIO maximum ALORS *c'est l'objet choisi
*arrêt
*SI le CIO est inférieur au CIO minimum ALORS retirer l'objet de la liste des objets potentiellement candidat
*Enlever la règle de la liste des règles objets
*Ordonner les objets selon leur CIO croissant
*SI il n'existe pas d'objet dont le CIO est supérieur au CIO maximum OU que tous les objets aient été retirés de la liste des objets candidats ALORS l'objet choisi est l'objet de plus fort CIO.

Note sur les instantiations des règles objets :
La recherche d'instances activables de règles (règles instanciées dont les conditions sont vérifiées) est coûteuse en temps. Cependant, il n'est pas possible de garder ici des règles partiellement instanciées d'un cycle à un autre (cf des systèmes comme GARI) car EAQUE peut être appliqué à des domaines non monotones, comme ORDF où, après chaque cycle, un ou plusieurs objets sont détruits. La recherche des instances activables des règles objets correspond à une recherche depth-first d'instances, pour les variables objet, sur les objets pris dans leur ordre décroissant de leur CIO.

-4- LES FONCTIONS DE NOTATION PAR DEFAUT

Les fonctions de notation pour le choix d'objet concernent l'augmentation et la diminution du CIO d'un objet.
- Augmentation du CIO : Elle se fait en ajoutant au CIO de l'objet le CIR de la règle appliquée.
- Diminution du CIO : Elle se fait en retranchant au CIO de l'objet le CIR de la règle appliquée.
Ces fonctions sont redéfinissables par l'Ingénieur Cognitif. Pour cela il a accès directement au CIR de la règle et au CIO de l'objet concernés (passage de paramètres), mais aussi à tout l'environnement (ce qui permet, par exemple, de tenir compte du contenu de la règle appliquée).

-5- EXPLICABILITE

Dans le cas de Systèmes Experts n'appliquant qu'une seule connaissance (Règle de Production par exemple) à la fois, il est facile de tracer la suite des connaissances appliquées pour justifier le résultat. Pour EAQUE qui peut considérer plusieurs connaissances aux effets cumulatifs, cela est moins direct. **EAQUE justifie le choix d'un objet en citant**

le règle, et le nombre de fois qu'elle a été appliquée, qui a le plus influencé ce choix (pour EAQUE, c'est le produit CIR par le nombre d'occurences, le plus grand).

:B: LE CHOIX D'ACTION

Résumé : Le choix d'action s'effectue par l'application conjuguée des règles actions, sur lesquelles ont été préalablement appliquées des méta-règles, et de fonctions d'évaluation heuristiques.

-1- BUT, LE CIA

Rappelons (cf le chapitre -III-3-) qu'une action est un Schéma d'Action instancié (partie droite d'une règle action instanciée).
L'objet étant choisi, étape précédente du cycle de contrôle (structure EOA), se pose le problème de savoir quelle action lui appliquer parmi celles qui sont possibles. Nous verrons à la section 3 de ce même paragraphe, comment elles sont déterminées. Nous avons vu au chapitre -III-3- que les règles actions permettent d'indiquer les actions qu'il est préférable de faire ou de ne pas faire. Cette notion se traduit en associant à chacune d'entre elles, une note, un **CIA**, Coefficient d'Importance de l'Action, que les règles augmenteront ou diminueront selon le cas en fonction de leur CIR. Ainsi, **l'action choisie sera l'action ayant obtenu le plus fort CIA.**
Un objet, ou un contexte, peut particulariser une ou plusieurs règles actions. La prise en compte de cette notion se fait à l'aide de **META-REGLES.** L'application des méta-règles est la première opération de la phase du choix d'action.

N'ayant pas incorporé de méta-règles pour notre exemple, nous citerons le cas d'école de MYCIN.
"Si le site de la culture est stérile, et s'il existe des règles qui mentionnent dans leur prémisses un organisme déjà

rencontré auparavant chez le patient et qui peut être le même dont on recherche l'identité, Alors il est sûr (1.0) qu'aucune d'entre elles ne peut servir."

-2- LES META-REGLES

Notes :

-1- Il serait possible de considérer les méta-règles selon le formalisme exposé dans -III-1-:B:c en prenant une restriction de la base de connaissances, mais l'utilisation qui en est faite ne nécessite pas un tel formalisme.

-2- Les méta-règles ont de nombreux points communs avec les règles objets, dont une syntaxe équivalente, et font également partie de la structure de contrôle d'EAQUE.

-3- Les méta-règles correspondent à un niveau **méta déclaratif**, nous verrons qu'il est également possible d'introduire un niveau méta procédural à travers les fonctions de notation des actions.

-2-1- but.

Les méta-règles indiquent sous quelles conditions (en partie gauche de la règle) telle ou telle règle action est intéressante ou non. Il existe donc deux types de méta-règles, les premières pour augmenter l'intérêt d'une règle action, les deuxièmes pour diminuer cet intérêt. Le CIR des règles actions traduisant l'intérêt de ces règles, les méta-règles vont donc augmenter ou diminuer, selon le cas, les CIR des règles actions.

-2-2le cir.

Les méta-règles possèdent un CIR (Coefficient d'Importance de la Règle) qui permettra de "moduler" la surnotation ou la dénotation des CIR des règles actions.

Citons un exemple informel :
SI l'objet a telle caractéristique et que tel fait apparaisse dans les prémisses de la règle action ALORS, avec une importance de 4, augmenter le CIR de la règle action de 2.

-2-3- première syntaxe.

Les méta-règles sont de la forme :
TYPE nom de la règle **CIR**
SI conditions
ALORS PG ;

Le type est ici méta-règle. Les CIR varient de 0 à 10.

-a- les conditions

Les conditions dépendent évidemment du domaine d'application, EAQUE ne s'intéresse qu'à leur évaluation booléenne.

-b- la partie gauche (PG)

Les méta-règles font partie de la structure de contrôle. EAQUE "connait" donc les différentes parties gauches (PG) qui peuvent apparaître dans une méta-règle, ainsi que leur interprétation.
Il existe deux types de PG différentes qui sont :
- Ajouter au CIR d'une règle action une certaine note.
- Retrancher au CIR d'une règle action une certaine note.

-2-4- les variables

Tout comme les règles objets (cf les règles objet -III-5-:A:) nous avons deux types de variables, **les variables**

règles qui jouent le rôle qu'avaient les variables objets pour les régles objet, et les variables locales. Tout comme les règles objets, les méta-règles font partie de la structure de contrôle, EAQUE connait **le domaine d'instantiation des variables règles, c'est l'ensemble des règles actions existantes.**
Les méta-règles ne s'appliquent que sur une seule règle action à la fois, il n'y a donc qu'une seule variable règle par méta-règle.

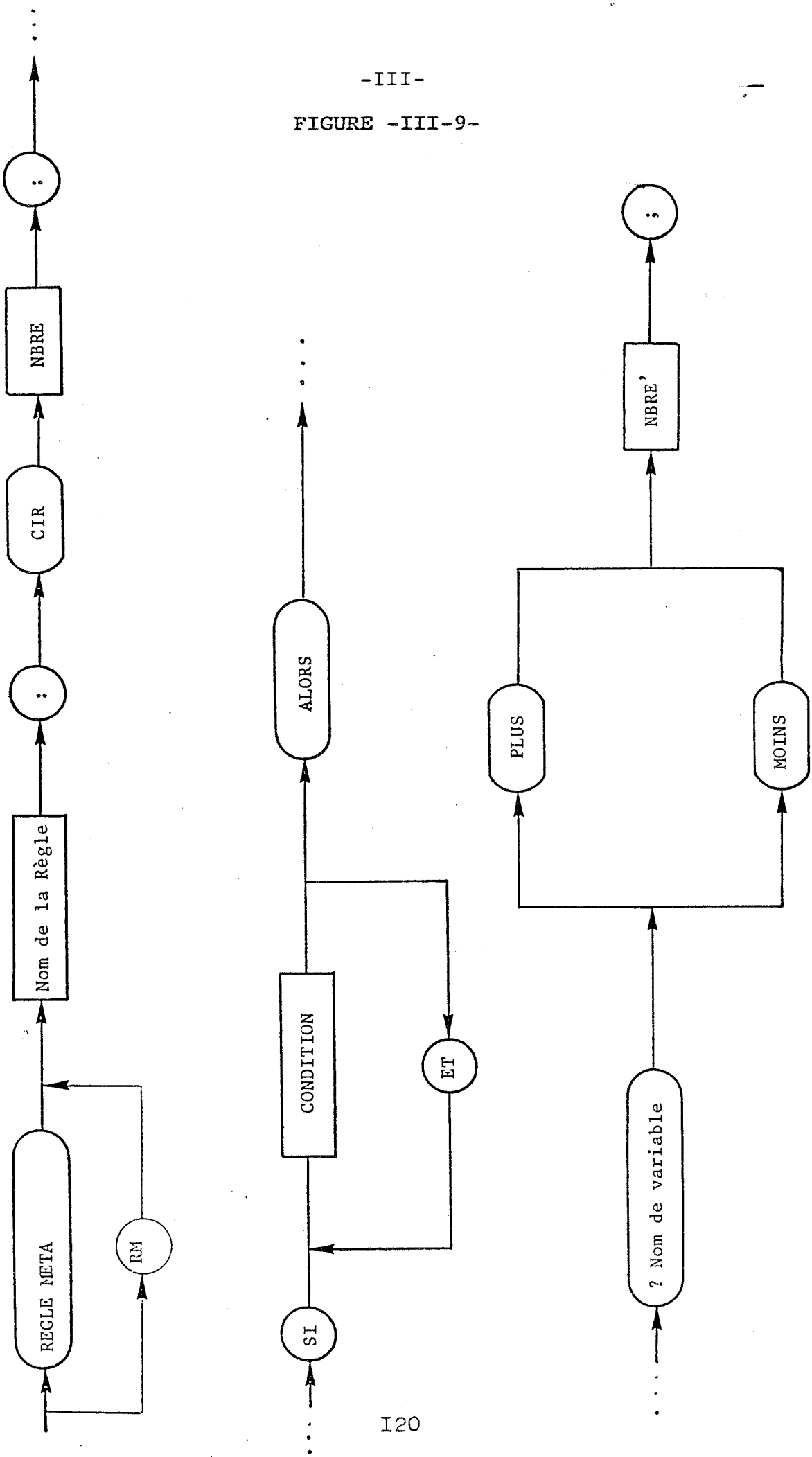
En reprenant notre exemple informel, il s'écrit :
REGLE META RM1 : CIR 4 :
SI COND1(OBJET) ET COND2(?RA)
ALORS ?RA PLUS 2 ;

-2-5- syntaxe

L'écriture des méta-règles doit respecter la syntaxe suivante :

cf FIGURES -III-9-

FIGURE -III-9-


REGLE META
RM
Nom de la Règle
:
CIR
NBRE
:
SI
CONDITION
ET
ALORS
? Nom de variable
PLUS
MOINS
NBRE'
;

-3- LE CHOIX D'ACTION

-3-1- l'application des méta-règles

La première opération est l'application des méta-règles afin de "mettre à jour" les CIR des règles actions. Ce n'est pas une procédure de choix, il n'existe pas de seuil minimum ou maximum comme dans le cas du choix d'objet. De plus, toutes les méta-règles sont appliquées (leur ordre d'application n'a alors pas d'importance) car il y en a généralement peu (par rapport au nombre de règles actions).

L'utilisation des méta-règles est la suivante : si les conditions de la méta-règle sont satisfaites, alors modifier, selon les indications de la partie gauche, le CIR de la règle action (instance de la variable règle de la méta-règle).

Les fonctions de notation par défaut des méta-règles sont :
- pour l'augmentation du CIR : le rajout au CIR de la règle action, de la note à ajouter.
- pour la diminution du CIR : la diminution du CIR de la règle action, de la note à retrancher.

Le CIR de la méta-règle n'intervient pas.
Ces fonctions sont redéfinissables par l'Ingénieur Cognitif. Pour cela, il a accès directement au CIR de la méta-règle, au CIR de l'instance de la variable action et à la note apparaissant en partie droite de la méta-règle (passage de paramètres), mais aussi à tout l'environnement.
Ainsi, nous pouvons totalement exclure, pour un cycle de contrôle donné, des règles actions en mettant leur CIR à 0.

-3-2- la détermination des actions.

EAQUE ne s'arrête pas sur la première instance de règle

action activable pour appliquer l'action correspondante. Sur l'objet choisi, il existe un ensemble d'actions possibles, activer la première trouvée, même si elle est dûe à une certaine politique dans le choix de la règle action à appliquer (par exemple la règle de plus fort CIR), est trop simpliste; il faut tenir compte, dans une certaine mesure, des autres règles actions, et donc des autres actions possibles.

Le choix de l'action à appliquer est, comme le choix de l'objet, basé sur des fonctions d'évaluation heuristiques, sur la recherche de l'élément le plus adéquat. Mais à la différence du choix d'objet où tous les objets possibles sont connus à l'avance, l'ensemble des actions candidates est au départ vide.

Les règles actions permettent non seulement d'augmenter ou de diminuer le CIA d'une action, ainsi que nous l'avons vu au début de ce paragraphe, mais aussi de créer une action si elle n'existait pas déjà.

L'utilisation des règles actions est donc la suivante :

Si une instance de règle est activable (les conditions instanciées sont vérifiées) alors, **il n'y a pas activation de l'action correspondante** (SA instancié), mais :

- si l'action existe déjà dans l'ensemble des actions candidates, alors son CIA sera modifié (augmentation ou diminution) selon les indications de la partie droite (il est préférable de... ou il est préférable de ne pas...).
- si elle n'existe pas, alors il est créé une nouvelle action candidate de CIA initial nul, qui sera modifié selon les indications de la partie droite de la règle.

Remarques :

-1- Nous pouvons ainsi avoir des actions identiques issues de règles actions différentes . Les CIA associés à ces actions permettent d'en tenir compte. Ce qui offre la possibilité de renforcer l'intérêt d'une action dans le cas où les règles favorisent la même action (il est préférable

de...); ou d'atténuer cet intérêt dans le cas où elles se contredisent (il est préférable de... et il est préférable de ne pas...).

-2- Cette approche permet l'introduction de règles actions déconseillant une action, alors que l'utilisation courante des règles de production est l'exécution des actions dès que leurs conditions d'activation sont vraies. Il y a donc une nette séparation entre les actions considérées en tant que telles et les règles de production dont elles sont issues. Le formalisme T = (SA , U , M) traduit bien cette volonté.

exemple :

Soient les règles actions :

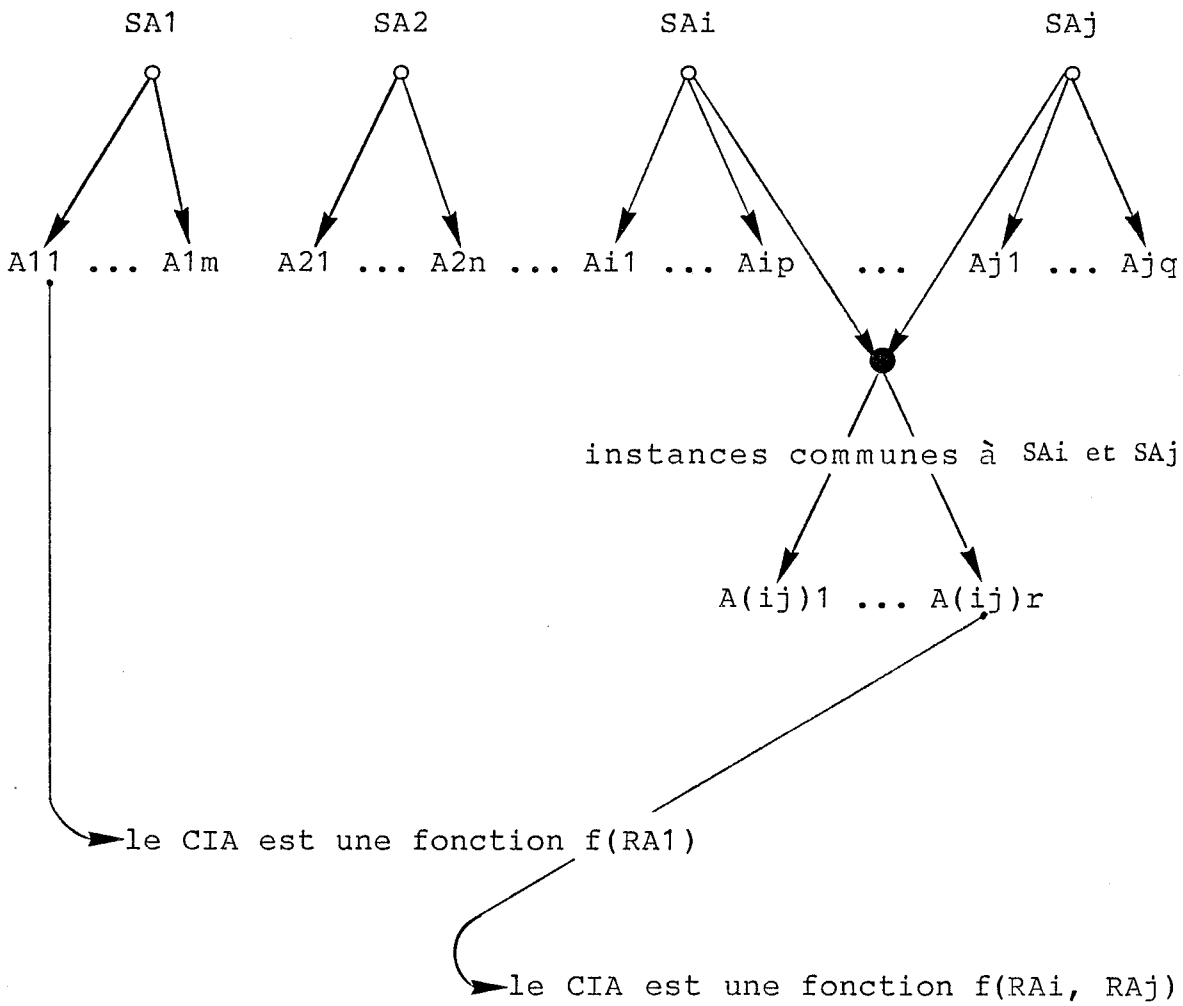
R1 : SI C1 ALORS SA1
R2 : SI C2 ALORS SA2
...
Ri : SI Ci ALORS SAi
...
Rj : SI Cj ALORS SAj
...

L'application de ces règles engendre un ensemble d'actions candidates. Les différentes instances de SA1 sont les actions notées A11...A1m, de même pour SA2,... Les actions issues de plusieurs règles seront notées A(indices des règles)1...A(indices des règles)n.

Ces actions, et leur CIA, peuvent être représentés par la figure suivante :

cf FIGURE -III-10-

FIGURE -III-10-



-3-3- le seuil maximum.

Il est possible de limiter EAQUE dans sa notation des actions par un seuil maximum, un CIA maximum. Ainsi, dès qu'une action a son CIA supérieur au CIA maximum, la recherche de l'action s'arrête et c'est l'action choisie. Nous n'avons pas ici, contrairement au choix d'objet, de CIA minimum puisque cela ne limiterait pas la recherche de la "meilleure" action (n'intervient pas et ne restreint pas les instances des règles actions). Le CIA maximum a une valeur par défaut (10) en liaison avec les fonctions de notation par défaut. Il est évidemment redéfinissable par l'Ingénieur Cognitif. Ainsi, il est tout à fait possible d'imposer à EAQUE de choisir l'action de la première instance de règle activable (fonctionnement courant des systèmes à base de règles de production). Dans ce cas, les règles déconseillant des actions n'ont plus de raison d'être.

-3-4- ordre des règles actions.

Les actions n'ont pas à être ordonnées en fonction de leur CIA, puisqu'ils n'interviennent pas dans la détermination des instances des règles.
Les règles actions sont appliquées selon les CIR décroissants (selon leur importance, et pour obtenir le plus rapidement, s'il y a augmentation du CIA, une action), en essayant pour une règle donnée, toutes ses instances possibles.

-3-5- la procédure de choix d'action.

Au début de chaque cycle, l'ensemble des actions candidates est vide.

*TANT QUE il n'existe pas d'action dont le CIA dépasse le seuil maximum **ET** qu'il reste des règles actions à appliquer **REPETER**
*Prendre la règle action de plus fort CIR
*POUR chaque instance vraie de cette règle sur ses domaines d'instantiations **REPETER**
*SI c'est une nouvelle action **ALORS** créer l'action candidate correspondante, de CIA nul
*noter l'action
*SI le CIA est supérieur au CIA maximum **ALORS** *c'est l'action choisie
*arrêt
*Enlever la règle de la liste des règles actions

***SI** il existe au moins une action **ET** qu'il n'existe pas d'action dont le CIA soit supérieur au CIA maximum **ALORS** l'action choisie est l'action de plus fort CIA.
***SI** il n'existe pas d'action **ALORS** revenir au choix d'objet.

Note sur les instantiations des règles actions
La note est identique à celle que nous avons faite à propos de l'instantiation des règles objets. La recherche d'instances, pour les variables actions, est une recherche depth-first sur leur domaine d'instantiation respectif.

-4- LES FONCTIONS DE NOTATION PAR DEFAUT

Les fonctions de notation pour le choix d'action concernent l'augmentation et la diminution du CIA d'une action.
- l'augmentation du CIA : correspond à une règle conseillant une action, elle se fait en ajoutant au CIA de l'action le CIR de la règle appliquée.

- la diminution du CIA : correspond à une règle action déconseillant une action, elle se fait en retranchant au CIA de l'action le CIR de la règle.

Ces fonctions sont redéfinissables par l'Ingénieur Cognitif. Pour cela, il a accès directement au CIR de la règle et au CIA de l'action concernés (passage de paramètres), ce qui permet d'introduire certaines considérations, comme "on ne prête qu'aux riches" avec une augmentation du CIA d'autant plus grande que le CIA est grand. L'Ingénieur Cognitif a également accès à tout l'environnement, en particulier à la structure de la règle action appliquée et de l'objet modifié, ce qui offre la possibilité d'introduire un niveau **méta procédural.**

-5- EXPLICABILITE

La justification par EAQUE du choix d'action est la même que pour le choix d'objet, à savoir la donnée de la règle action, et de son occurence, ayant le plus influencé ce choix.

:C: EXECUTION

C'est la troisième et dernière étape du cycle de contrôle : l'exécution de l'action. Cette exécution dépend évidemment du domaine d'application; pour EAQUE c'est un appel de procédure. A l'exécution d'une action est associé un ensemble d'informations nécessaires à la restauration à l'état antérieur à l'application de l'action. Ce sont, de la part d'EAQUE, les objets qu'elle détruit et qu'elle crée, de la part de l'Ingénieur Cognitif, lorsque l'action n'a pas d'action "inverse", le programme, écrit dans le langage hôte (LRO-LISP), qu'il faudra exécuter pour revenir à l'état précédent.

ETUDE ET RESOLUTION

DES CONTRADICTIONS

:A: LES CONTRADICTIONS.
:B: LA NOTATION DES CHOIX.
:C: LE RELACHEMENT DES OBJECTIFS.
:D: CHOIX D'ETAT (Restauration, Retour Arrière).

Résumé : Un état ayant généré des contradictions est un état incorrect. Nous verrons quand elles peuvent apparaître, quelles en sont les causes, et comment les résoudre. Ce qui nous amènera, soit à modifier les sources de la contradiction, soit à revenir à un état antérieur.

:A: LES CONTRADICTIONS.

-1- Définition et CAUSES d'une contradiction.

Une contradiction traduit un mauvais fonctionnement du moteur. Il existe cinq causes possibles à une contradiction, nous distinguerons, donc, cinq types de contradictions.

1- type 1 : le non respect des objectifs.

Les objectifs sont des contraintes que doit respecter la base de connaissances à chaque cycle de contrôle (cf -III-

3). Il suffit qu'un seul objectif, quel que soit son coefficient de relâchement, ne soit plus vérifié pour qu'une contradiction soit générée.
Le dépassement d'une date limite à la réalisation d'un projet en bâtiment, est un exemple d'objectif non respecté.

2- type 2 : les conséquences de l'exécution d'une action.

Ces conséquences dépendent, évidemment, du domaine d'application. C'est donc à l'Ingénieur Cognitif de vérifier la cohérence de la base de connaissances, après chaque application de règle; et de générer, si nécessaire, une contradiction (qui se fait simplement en mettant une variable réservée d'EAQUE à vrai).
La création d'un circuit, pour les problèmes d'ordonnancement, est un exemple rentrant dans ce cas.

3- type 3 : il n'existe plus d'action applicable.

C'est le cas où il n'existe pas d'action applicable à l'un, quelconque, des objets d'un état donné. EAQUE ne peut rien faire sur cet état, alors que la condition d'arrêt n'est pas vérifiée, que les objectifs sont respectés, et qu'il reste des objets.

4- type 4 : il n'existe plus d'objet.

C'est le cas où il n'existe plus d'objet, pour l'état considéré, alors que la condition d'arrêt n'est pas vérifiée, et que les objectifs sont respectés.

5- type 5 : il n'existe pas de point de reprise.

Nous verrons qu'une des solutions possibles, face à une contradiction, est de revenir à un état antérieur, où cette contradiction n'existe pas. Il y a de nouveau contradiction

lorsqu'il n'existe pas d'état de restauration.

-2- QUAND?.

La détection et la génération de contradictions se font, au début de chaque cycle de contrôle, afin de vérifier si les objectifs sont respectés; après chaque application d'une action (les conséquences de son exécution); au choix d'objet; au choix d'action; et, le cas échéant, à la recherche de l'état de restauration.

-3- LES SOLUTIONS.

A chaque cause, son remède. Ainsi, pour chaque type de contradictions, nous aurons :

-a- type 5.

C'est une erreur fatale, il n'existe pas de remède à ce type de contradictions. **Pour EAQUE, il n'existe pas de solution raisonnable au problème,** le système s'arrête. Ce qui ne veut pas dire qu'il n'existe pas de solution, mais qu'il n'en n'existe pas de satisfaisante au sens des fonctions d'évaluation définies par l'Ingénieur Cognitif et l'Expert. En effet, nous verrons au chapitre suivant (:B: LA NOTATION DES CHOIX), que les différents états par lesquels passe le système, sont évalués comme candidats possibles à un état de restauration, offrant d'autres choix à l'obtention d'une solution. S'il n'existe aucun état jugé satisfaisant, contradiction de type 5, le système s'arrête.

-b- types 3, 4.

Les types 3 et 4 correspondent au fait qu'EAQUE se trouve

dans un état d'impasse, où il n'y a rien à faire. Il faut relancer la résolution à partir d'un autre état. il y a donc **restauration à un état antérieur,** s'il existe.

-c- type 2.

Tout comme les types 3 et 4, le type 2 correspond à un état d'impasse, il y a également **restauration à un état antérieur,** s'il existe. La différence est que l'Ingénieur Cognitif a, ici, la possibilité de rajouter les connaissances nécessaires à l'interdiction de l'exécution d'une action équivalente, dans un autre état.

-d- type 1.

Il existe deux **solutions possibles** à ce type de contradictions, soit **revenir à un état antérieur,** où les objectifs étaient forcément respectés, soit, si cela est possible, supprimer les causes de la contradiction, c'est le **relâchement des objectifs.**
EAQUE va, dans un premier temps, évaluer le meilleur état de restauration possible (cf chapitre suivant, :B: LA NOTATION DES CHOIX), et lui associer une note. Dans un deuxième temps, EAQUE analyse l'objectif contredit et regarde s'il sait le relacher (cf chapitre suivant, :C: LE RELACHEMENT DES OBJECTIFS); dans l'affirmative, et si le coefficient de relâchement de l'objectif est supérieur à la note du meilleur état de restauration, il y a relâchement de l'objectif, sinon, dans tous les autre cas, il y a retour au meilleur état de restauration. Si plusieurs objectifs sont contredits, on procède de même pour chacun d'entre eux.

:B: LA NOTATION DES CHOIX.

Résumé : Le choix d'objet et le choix d'action, lors de la résolution d'une contradiction, peuvent être remis en cause. Certains choix sont plus susceptibles d'être rejetés que d'autres, nous le traduirons en leur associant une note (de rejet, de doute, fonction des autres choix alors possibles...).

-1- ESPACE DE RECHERCHE.

L'espace de recherche peut se représenter par la figure suivante : cf FIGURE -III-11-.

Le passage d'un état à un autre se fait par une transition, qui a modifié le contexte, nous appellerons ces états, qui sont différents entre eux, des **états réels.**
Une transition se décompose en un choix d'objet et un choix d'action. Or le choix d'objet, tout comme le choix d'action, participe à la limitation de l'espace de recherche. Le choix d'objet définit un état intermédiaire, que nous appellerons un **état virtuel**, car il n'y a pas eu modification de l'état réel.
La figure -III-12- représente l'espace de recherche en tenant compte des états réels et virtuels.

L'état courant est un chemin dans cet espace de recherche, il se définit comme une suite alternée de choix d'objet et de choix d'action :
choix d'objet, choix d'action, choix d'objet, choix d'action...
ou comme une suite alternée, équivalente à la précédente, d'états réels et d'états virtuels.

FIGURE -III-11-

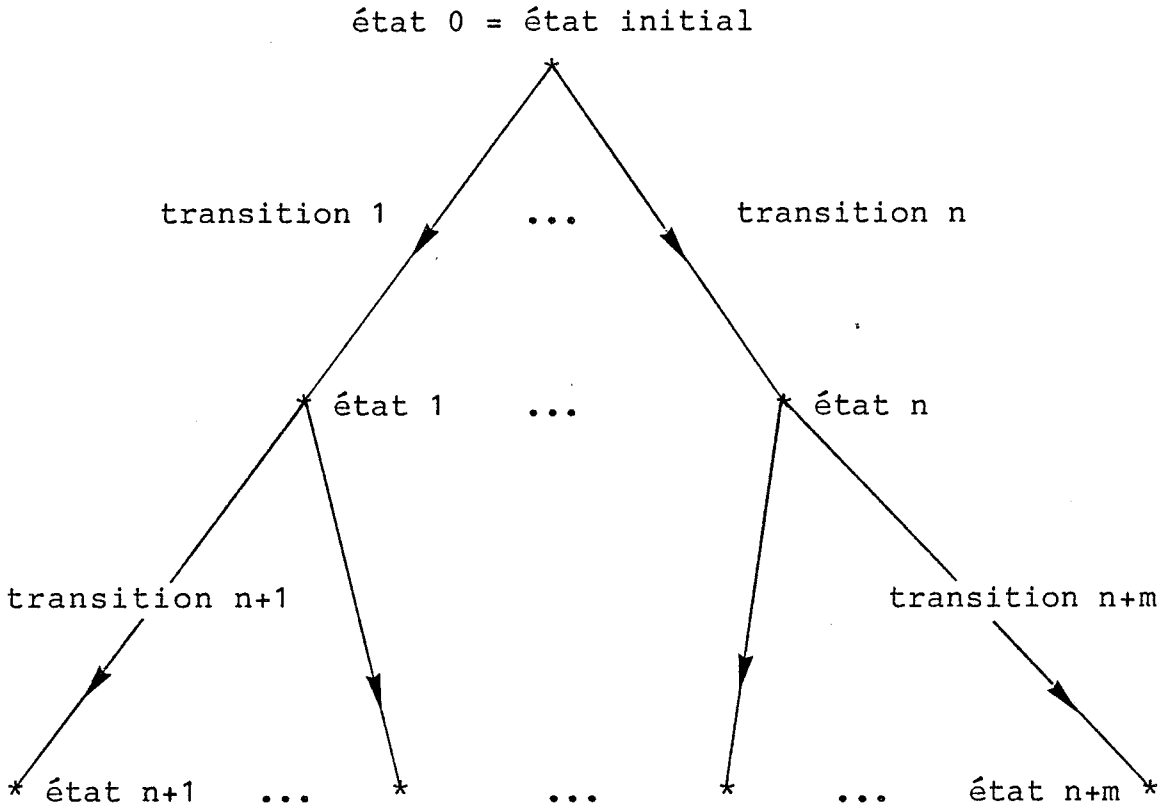
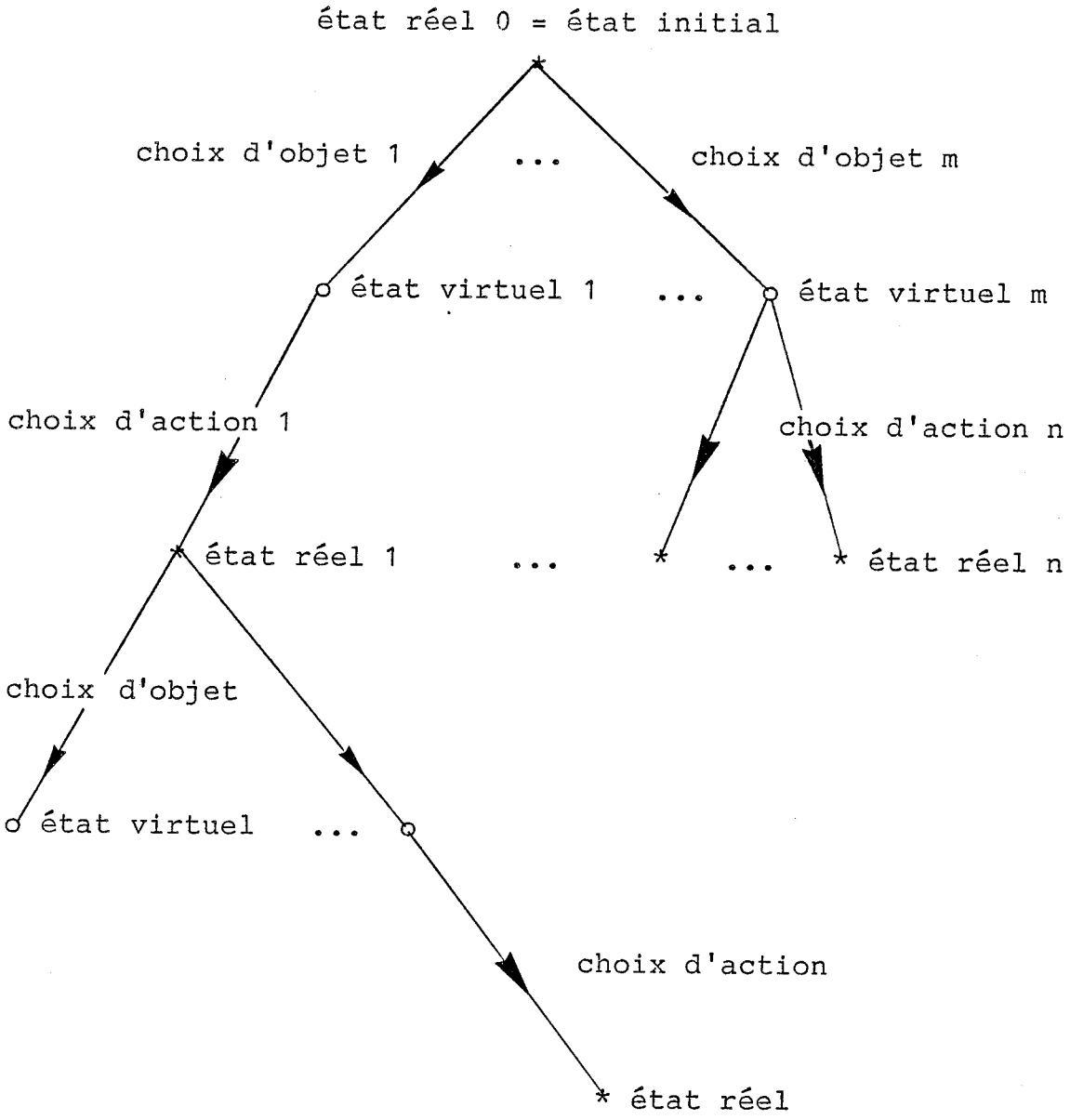


FIGURE -III-12-



-2- LA NOTATION DES CHOIX.

Nous avons vu, au chapitre précédent, qu'il est parfois nécessaire de revenir à un état antérieur, état réel ou virtuel. Il se pose alors le problème du choix d'état. Certains états sont meilleurs, nous nous en doutons, que d'autres. Il est évident qu'entre deux états, par exemple réels, où le premier offrait plusieurs choix d'objet possibles dont plusieurs avec un fort CIO, et le second très pauvre en choix, il est préférable de revenir sur le premier. Tout comme il est préférable, entre deux états "équivalents", de privilégier le plus récent. C'est pourquoi il est associé, comme nous l'avons fait pour les objets et les actions, **une note** à chaque état, qu'il soit virtuel ou réel, représentant sa capacité à être considéré comme point de reprise. Nous dirons également que cette note est associée au choix issu de cet état, choix d'objet pour l'état réel, d'action pour l'état virtuel. Ce qui est somme tout naturel, puisque nous verrons qu'un point de reprise représente en fait, pour EAQUE, un choix à rejeter.

Cette note est affectée à l'état **au moment où a été fait le choix** (d'objet à partir de l'état réel, d'action à partir de l'état virtuel). Il ne peut donc être tenu compte, pour le retour arrière, des informations générées entre la création de cet état et l'apparition d'une contradiction.

Il existe, par défaut, une fonction de notation des états réels ou des choix d'objet, et une fonction de notation des états virtuel ou des choix d'action. Ce sont, pour les deux fonctions, la somme du nombre des autres choix possibles et du maximum de leur coefficient d'importance (CIO pour les objets, CIA pour les actions), diminuée du coefficient d'importance de l'élément choisi; ou 0 si cette note est négative.
Ces fonctions sont redéfinissables par l'Ingénieur

Cognitif. Il a accès, pour cela, directement au coefficient d'importance de l'élément choisi, au nombre des autres choix possibles et du maximum de leur coefficient d'importance (passage de paramètres), mais également à tout l'environnement. Il peut ainsi définir la fonction du depth-first, qui est tout simplement le nombre de choix effectués entre l'état initial et l'état courant (état à noter); privilégier les états les plus récents, les états virtuels ou états réels; distinguer l'intérêt ou le doute du choix d'un état, de la simple notion d'alternative à un choix...

Il faut noter l'existence d'une borne inférieure, 0, indiquant qu'un état ne peut être considéré comme un point de reprise.
La note d'un état est donc forcée à 0 lorsqu'il n'existe qu'un seul objet possible, état réel, ou qu'une seule action possible, état virtuel.

:C: LE RELACHEMENT DES OBJECTIFS.

Il existe un objectif non respecté (génération d'une contradiction de type 1). EAQUE "sait" relacher l'objectif contredit. Son coefficient de relachement est supérieur à toutes les notes associées aux différents états, de l'état initial à l'état courant. Dans ces conditions, EAQUE décide de supprimer la source de la contradiction, c'est-à-dire **relâcher, mais non supprimer, l'objectif contredit**, plutôt que de restaurer un état antérieur.

Relâcher un objectif est un choix, tout comme le choix d'objet ou le choix d'action, mais n'est pas un point de reprise potentiel, puisqu'il n'introduit pas de contrainte supplémentaire, au contraire.

Relâcher un objectif consiste à le remplacer par un nouvel objectif, qui s'éloigne le moins possible du premier, et qui possède le même coefficient de relâchement. EAQUE ne sait que relâcher les objectifs correspondant à des fonctions bornées : expressions inférieures, inférieures ou égales, supérieures, supérieures ou égales, à un nombre. Le relâchement consiste à actualiser la borne par la valeur ayant mis en défaut l'objectif.

En considérant notre exemple informel défini en -III-2-:C:, un objectif possible est d'avoir à tout moment la date au plus tard de la dernière tâche inférieure à une durée limite, par exemple 60 jours. Dans le cas où il n'est pas respecté et que la date au plus tard de la dernière tâche est de 65 jours, le nouvel objectif est d'avoir cette date inférieure ou égale à 65 jours.

Au choix de relâchement d'objectif est associé l'ensemble des informations nécessaires pour un retour arrière, à

savoir l'objectif relaché.

La contradiction est résolue, et le moteur est relancé.

Note : S'il existe plusieurs objectifs non respectés, le relâchement de chacun d'eux s'effectue, de la manière décrite ci dessus, si le minimum des coefficients de relâchement est supérieur à toutes les notes associées aux différents états, et si EAQUE sait relâcher tous ces objectifs.

:D: LE CHOIX D'ETAT (Restauration, Retour Arrière).

Résumé : Pour EAQUE, **le choix d'état correspond à un retour arrière sélectif, suivi par le rejet du choix d'objet ou d'action qui lui a été appliqué.** Il doit également se rappeler, tirer les conséquences et adopter une politique convergente de backtrack, afin d'avoir la possibilité de considérer les choix rejetés comme des choix à part entière.

En fonctionnement "normal", l'état sur lequel le système se doit d'agir est le dernier état généré. Le problème du choix d'un autre état ne se pose qu'après rencontre d'une contradiction, et dans le cas où la solution de restauration à un état antérieur à été choisie.

-1- LE CHOIX D'ETAT.

Nous avons vu au paragraphe précédent :B:, que les états étaient notés selon leur capacité à être considérés comme point de reprise. **L'état à restaurer est le premier état de note maximum.** Le premier signifie le plus récemment appliqué. EAQUE, entre deux états de notes maximum, prendra le plus récent.

Dans le cas où toutes les notes sont nulles, il n'existe aucun point de reprise (génération d'une contradiction de type 5), il n'y a pas de solution au problème, au sens d'EAQUE, et le système s'arrête.

-2- LA RESTAURATION.

L'état à restaurer est séparé de l'état courant par une suite de choix, ou une suite d'états. Revenir à l'état de restauration revient à dépiler cette suite de choix. Si, en

dépilant, on rencontre un choix d'objet, il n'y a aucune information à restaurer, puisqu'il n'y a pas eu de modification du contexte; si c'est un choix de relâchement d'objectif, alors les anciens objectifs sont restaurés; si c'est un choix d'action, les objets détruits par l'exécution de l'action remplacent ceux qu'elle a créés, l'action "symétrique" et le programme de restauration, s'ils existent, sont lancés (l'action symétrique et le programme de restauration, dans le cas où ils sont nécessaires, sont définis par l'Ingénieur Cognitif).

-3- SE RAPPELLER DU REJET.

Sélectionner un état de restauration, c'est sélectionner un choix, d'objet ou d'action, à rejeter, afin d'en effectuer un différent pour diriger la résolution dans une autre direction (il existe au moins un autre choix possible, sinon l'état n'aurait pas été un point de reprise.) C'est pourquoi, nous considérerons, dorénavant, les notes associées aux choix, plutôt qu'aux états.

Errare machina est.
Le rejet d'un choix ne doit pas être définitif, EAQUE doit pouvoir revenir sur cette décision, si nécessaire. Le problème de convergence se pose alors.
Il est associé au choix rejeté une note qui lui sera attribuée s'il est de nouveau pris, nous appellerons une telle note, une **note de reprise.** Cette note doit être différente de celle qu'il avait avant d'être rejeté, sinon il y aura un risque de bouclage. De plus, cette nouvelle note doit être inférieure à la précédente, puisqu'une reprise du choix rejeté, signifie que son rejet n'a pas résolu le problème; il est, en tant que candidat au rejet, moins intéressant que prévu; sa note doit être baissée. De plus, EAQUE ne posséde pas de critère particulier pour la détermination de la note à attribuer au choix rejeté, il

adopte donc une politique générale, qui va pénaliser le moins possible le choix rejeté, tout en assurant, s'il est de nouveau pris, et si le contexte n'a pas changé, un nouveau choix pour la résolution de la contradiction qui ne manquera pas d'arriver. **La note de reprise associée au choix rejeté est la note maximale des choix qui le suivaient** jusqu'à l'état courant, avant l'apparition de la contradiction qui a entrainé son rejet. Une telle politique de résolution est valable quelque soit le domaine d'application.
Si la note maximale des choix, qui suivent celui à rejeter, est nulle. Cela signifie qu'il n'existe pas d'autre choix qui puisse être écarté, si celui qui a été rejeté est de nouveau pris (sous entend une restauration intégrale du contexte). Associer la note de reprise 0 à un choix rejeté, signifie donc, qu'il ne sera jamais repris.
La réutilisation d'un choix rejeté est considéré, par EAQUE, comme un autre choix, à la différence que la note qui lui est attribuée n'est pas déterminée par la fonction de notation des choix, mais est celle qui lui a été associée lors de son rejet (note de reprise). Il pourra, de nouveau, être soumis à un rejet, ce qui se traduira par une note de reprise, associée à ce deuxième rejet, encore plus faible.

La présence de la borne inférieure 0, assure la convergence de la résolution.

Il faut noter que si le choix rejeté est le dernier qui a été effectué (a entrainé la contradiction), alors il ne pourra jamais être de nouveau sélectionné, puisqu'EAQUE n'a pas d'autre alternative, autre que la contradiction. La note 0 est donc associée au rejet de ce choix.

Les choix rejetés, et leur note de reprise, sont associés, évidemment, à l'état sur lequel ils ont été effectués, puisque les nouveaux choix doivent en tenir compte.

Considérons, pour illustration, l'exemple suivant :
Notons (COBJ n) et (CACT m), les couples (choix d'objet et sa note associée) et (choix d'action et sa note associée).
La suite des choix, à l'apparition d'une contradiction, est :
(COBJ1 2)(CACT1 0)(COBJ2 8)(CACT2 1) apparition d'une contradiction.
Le choix rejeté est COBJ2. Il y a restauration à l'état réel le précédant, auquel est associé COBJ2 et sa note de reprise, 1. Ce que nous pouvons représenter par :
(COBJ1 2)(CACT1 0)
! état réel auquel est associé COBJ2 avec sa note de reprise, 1.

-4- LE CHOIX DU NOUVEL ETAT.

Nous considérerons le cas où le choix rejeté est un choix d'objet, il y a donc eu restauration de l'état réel le précédant. Se pose alors le problème d'un nouveau choix d'objet, ou d'un nouvel état virtuel. Le principe est le même pour le choix d'une nouvelle action, ou d'un nouvel état réel, nous ne le détaillerons pas.

PROCEDURE NOUVEAU CHOIX D'OBJET :

-a- Dans le cas où l'état restauré n'a pas déjà engendré de choix d'objet rejeté.
- L'objet, correspondant au choix d'objet rejeté, est enlevé de la liste des objets candidats.
- Il existe forcément, au moins un autre objet possible, par définition de l'état de restauration (note strictement positive).
- S'il existe plus d'un objet :
La recherche du "meilleur" objet se fait de manière

identique au choix d'objet vu au chapitre -III-5- (CHOIX D'OBJET, CHOIX D'ACTION, EXECUTION) : détermination du meilleur CIO par l'application des règles objets.
Si la note associée au choix de ce meilleur objet (cf -III-6-:B: LA NOTATION DES CHOIX) est supérieure à la note de reprise de celui qui vient d'être rejeté, alors c'est le nouveau choix d'objet.
Sinon, le choix rejeté, même avec sa note de reprise qui a diminué son intérêt à être candidat au rejet, est "meilleur" que tous les autres choix d'objet possibles. EAQUE revient sur sa décision, le choix rejeté est de nouveau sélectionné avec comme note, sa note de reprise.
- S'il existe qu'un seul objet, c'est le seul autre choix possible, l'application des règles objets n'a aucune raison d'être pour EAQUE. Il le considère comme le nouveau choix dont la note est la note de reprise du choix d'objet rejeté (puisqu'EAQUE ne l'a pas évalué, il n'a donc, aucune raison à le préférer ou non au choix rejeté).

-b- Dans le cas où cet état a déjà engendré des choix d'objet rejetés.
- Notons N, le maximum des notes de reprise des choix déjà rejetés.
- Si N est plus grand que la note de reprise du choix qu'EAQUE vient de rejeter, alors il y a reprise de l'ancien choix rejeté de note de reprise N. C'est le nouveau choix d'objet, avec comme note N.
- Sinon, nous sommes ramenés au cas -a- précédent, où il est considéré, non pas forcément le dernier choix rejeté, mais le meilleur des choix rejetés (correspondant à N).

INSTANTIATION D'EAQUE

A UN DOMAINE D'APPLICATION PARTICULIER

Résumé : Instancier EAQUE à un domaine, c'est lui rajouter un ensemble de règles "expertes"; et définir, en fonction de ce domaine : ses propriétés, les stratégies pour la structure de contrôle, les classes LRO pour décrire le contexte et le problème, les sémantiques des objets et des actions.

:A: INTRODUCTION

L'instantiation d'EAQUE est le rajout de deux bases de connaissances, afin d'en obtenir un Système Expert dans le domaine d'application choisi.

-1- la première base.

La première base de connaissances, déclarative, est définie par l'expert (représente l'expertise du domaine). Elle est constituée de trois ensembles de règles différents : les règles objets, les méta-règles de choix d'action et les règles actions. Nous appellerons cette base **la base de règles.**
L'expert utilise trois fichiers différents, un par type de règles, qui peuvent être créés, modifiés, mis à jour, par

n'importe quel traitement de texte.
Un programme de précompilation, transparent à l'expert, transforme les règles en une représentation interne utilisable par EAQUE (cf -III-8-).

-2- la deuxième base.

La deuxième est définie conjointement par l'Ingénieur Cognitif et l'Expert. Elle représente une quantité de travail relativement importante qui nécessite de plus, de la part de l'Ingénieur Cognitif, une parfaite connaissance d'EAQUE, et de l'Expert; une approche formelle de son domaine.
Cette base est constituée de deux types d'informations, les premières concernent directement EAQUE, comme la redéfinition des fonctions par défaut, les secondes sont propres au domaine, comme ses propriétés. Cette base sera appelée la **base du domaine.**

la figure -III-13- représente la structure d'EAQUE, la figure -III-14-, son instantiation.

La définition de la base de règles se suffit à elle-même, par contre, la base du domaine nécessite une étude détaillée.

FIGURE -III-13-

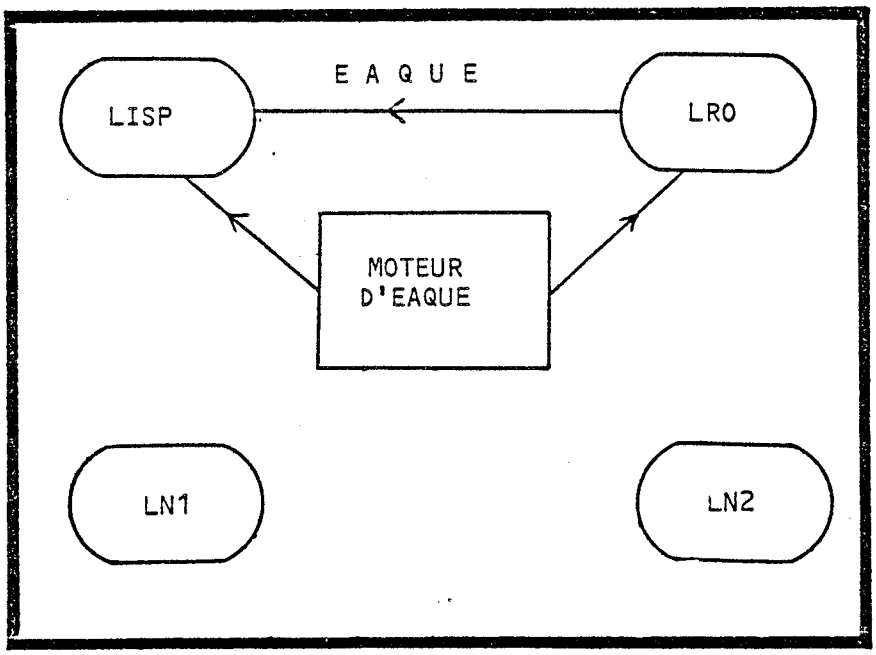
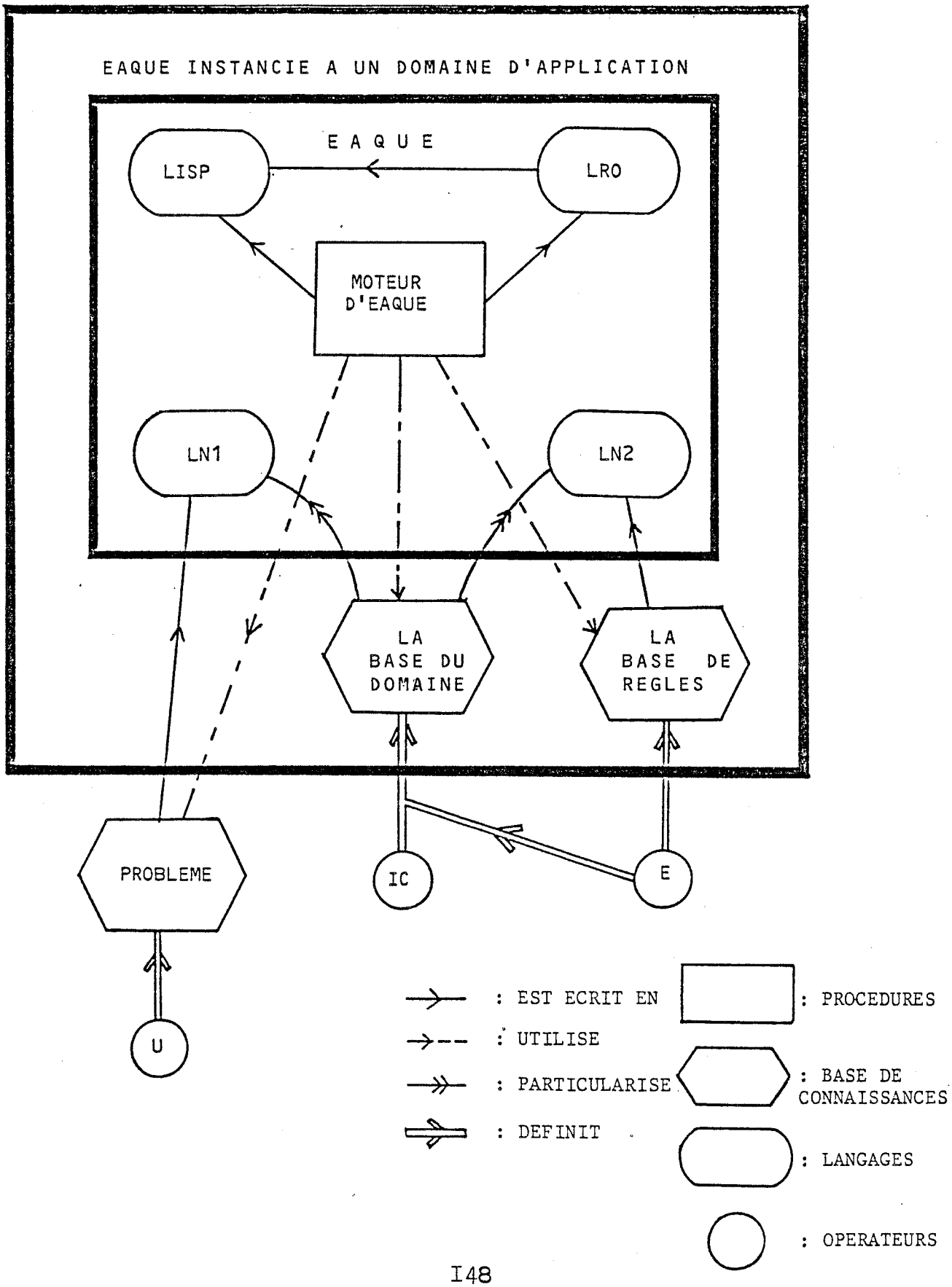


FIGURE -III-14-



:B: LA BASE DU DOMAINE

-1- EAQUE.

Cette section correspond à **la programmation de la structure de contrôle d'EAQUE**, c'est-à-dire à **la définition de stratégies.**

L'Ingénieur Cognitif, si besoin est, redéfinit, en collaboration avec l'Expert, certaines informations à l'intention d'EAQUE. Ce sont les fonctions et les variables possédant des valeurs par défaut. Citons les CIO maximum et minimum, le CIA maximum, les fonctions d'évaluation associées aux règles objets (augmentation, diminution du CIO), aux méta-règles (modification du CIR), aux règles actions (modification du CIA), au choix d'objet, au choix d'action (stratégie du backtrack)...
Pour aider les Ingénieurs Cognitifs dans cette tâche, un fichier utilitaire est mis à leur disposition offrant un ensemble de fonctions prédéfinies. Citons des fonctions de notation des choix correspondant au depth-first, d'autres tenant compte de l'ordre de création des différents états; des fonctions d'analyse syntaxique pour l'écriture des conditions et des actions des règles, des objectifs, de la condition d'arrêt; des fonctions "à personnaliser", de vérification d'objectifs et de condition d'arrêt utilisant des variables...

-2- LE DOMAINE.

Le travail de l'Ingénieur Cognitif, en ce qui concerne le domaine d'application, se divise en cinq points :
- Il définit les classes d'objets LRO qui participe à **la définition du contexte et du problème** (instructions LRO de déclaration de type).

Pour notre exemple ce sont les classes TACHE et RESSOURCE vues au chapitre -III-3-.

-Il définit **la structure des objets** par la déclaration d'une classe LRO réservée.

Cette définition n'est pas neutre, elle sous entend un choix de méthode de résolution.

- Il définit **les actions**, qui vont s'exécuter sur un objet.

Dans l'exemple que nous avons traité tout au long de ce chapitre, c'est l'unique action "METTRE une tâche AVANT une autre tâche".

- Il traduit, procéduralement, **les propriétés du domaine.**

Citons , toujours pour notre exemple, les procédures de détermination des dates au plus tôt et au plus tard, les propriétés liées au notion de graphe et d'ensemble : chemin, fermeture transitive, inclusion, ordre...

- Enfin, il particularise **les squelettes d'analyse syntaxique.**

IMPLEMENTATION

:A: INTRODUCTION.

EAQUE est écrit entièrement en LRO-LISP. Il fait dans sa dernière version trois mille lignes (sans tenir compte des instantiations ORDF, CALINT et EALOG). L'utilisation de techniques issues des langages orientés objet, a grandement aidé l'écriture d'EAQUE. La facilité avec laquelle ont été faites la mise au point et le rajout d'un niveau méta en sont les meilleures preuves. De plus, la structure des Systèmes Experts se prête bien à leur écriture par de tels langages.

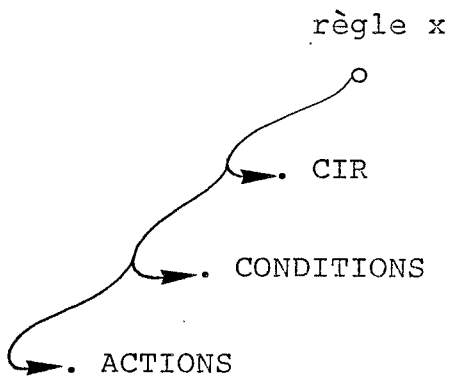
Le moteur d'EAQUE, les règles objets, les méta-règles, les règles actions, les actions, les objets, les choix d'objet, les choix d'action, les relâchement d'objectifs, sont autant d'objets LRO de classes différentes auquelles sont rattachées leurs procédures d'exploitation. Les envois de messages permettent de gérer facilement l'utilisation de ces objets.

:B: LES REGLES.

-1- la représentation des règles

Les règles objets, les méta-règles et les règles actions sont toutes des objets LRO appartenant aux classes REGLE_OBJET, REGLE_META et REGLE_ACTION. Trois propriétés suffisent à les décrire, ce sont : CIR, CONDITIONS et ACTIONS, dont les buts sont évidents. La déclaration de ces classes se fait avec la valeur par défaut 0 pour le CIR.

Nous pouvons représenter ces objets LRO par la figure suivante :



-2- les procédures d'exploitation des règles.

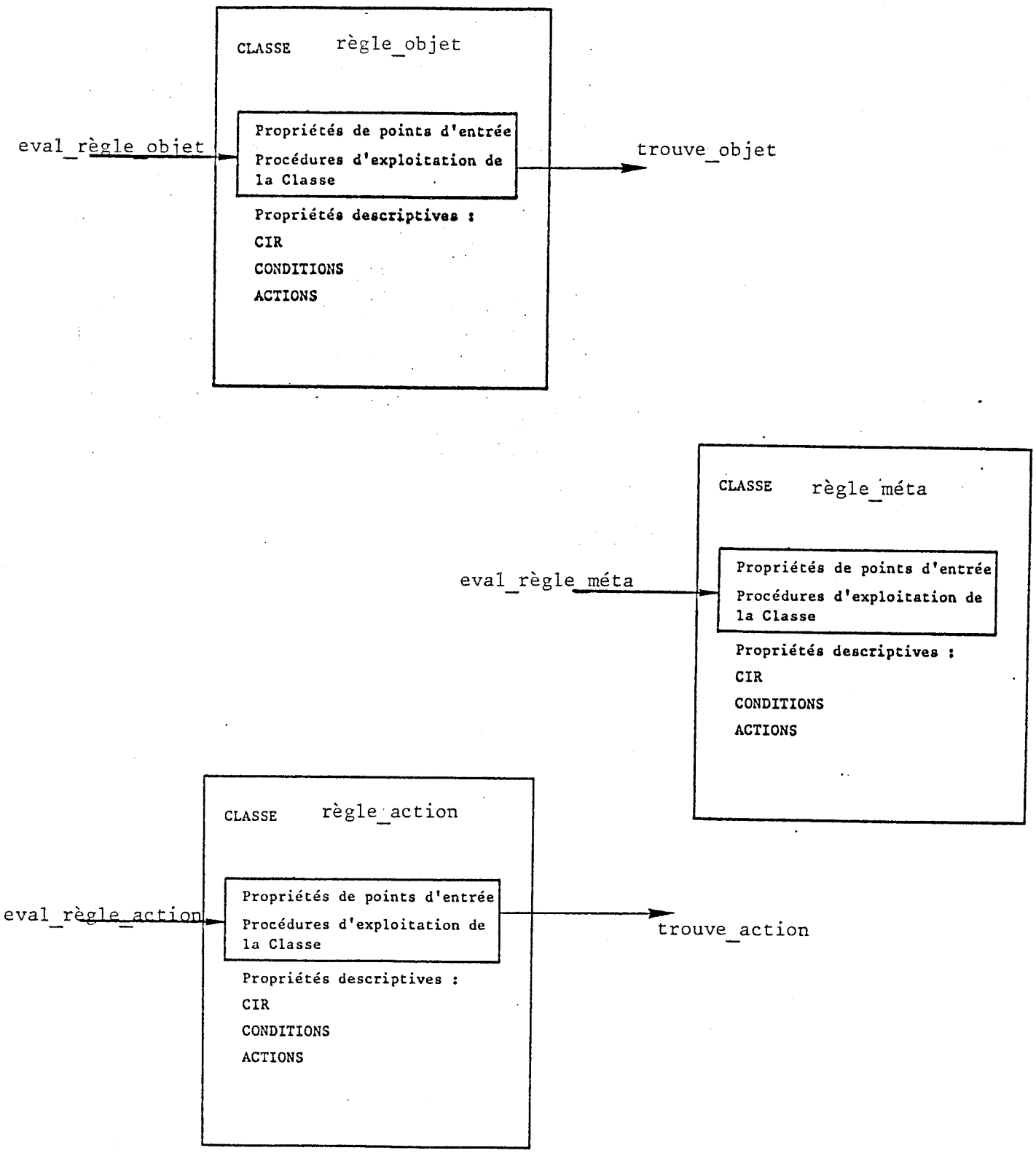
Chacune des trois classes que nous venons de voir, a la propriété SEMAPHORE, ce qui protège en écriture les propriétés de leurs objets LRO.

Ces classes n'ont qu'un point d'entrée, c'est la propriété EVAL_ REGLE_OBJET, pour la classe REGLE_OBJET, auquelle est attachée la procédure d'exploitation d'une règle OBJET; la propriété EVAL_REGLE_META, pour la classe REGLE_META; et la propriété EVAL_REGLE_ACTION, pour la REGLE_ACTION. Cette

procédure peut envoyer un message en direction de l'objet LRO représentant le moteur d'EAQUE. C'est le message TROUVE_OBJET, pour la procédure associée aux règles objets, indiquant qu'il existe un objet dont le CIO dépasse le CIO maximum. C'est le message TROUVE_ACTION, pour la procédure associée aux règles actions, indiquant que le CIA d'une action a dépassé le CIA maximum.

La figure -III-15- donne une visualisation possible des trois classes que nous venons de voir.

FIGURE -III-15-



:C: LE MOTEUR.

Le moteur d'EAQUE est un objet LRO, il est l'unique élément de la classe MOTEUR.
Le moteur n'a à traduire aucune connaissance descriptive, il est donc uniquement un objet virtuel récepteur des messages qui lui seront envoyés.

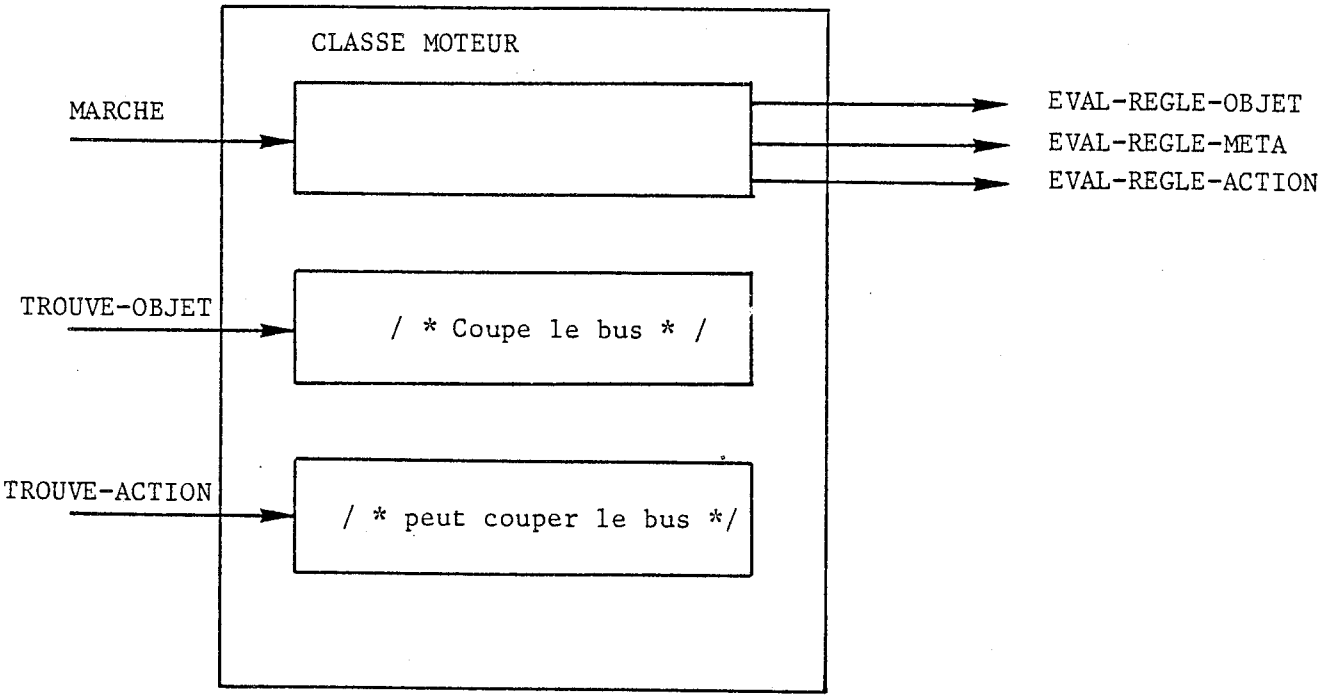
les procédures d'exploitation du moteur.

La classe MOTEUR possède trois points d'entrée, les propriétés MARCHE, TROUVE_OBJET et TROUVE_ACTION, auquelles sont attachées les procédures correspondantes à ces messages.
Sur réception du message MARCHE, le moteur d'EAQUE est activé. La procédure correspondante, qui est la procédure principale d'EAQUE, envoie trois messages, le message EVAL_REGLE_OBJET à la classe REGLE_OBJET, dont la manipulation souhaitée est la notation des CIO des objets; le message EVAL_REGLE_META à la classe REGLE_META, dont la manipulation souhaitée est la mise à jour des CIR; et le message EVAL_REGLE_ACTION à la classe REGLE_ACTION, dont la manipulation souhaitée est la mise à jour des CIA des actions.
Sur réception du message TROUVE_OBJET, la procédure associée coupe le bus des messages, en effet il n'est plus nécessaire que LRO "dispatch" le message EVAL_REGLE_OBJET sur d'autres objets LRO de type REGLE_OBJET, et indique à la procédure principale qu'un objet a un CIO supérieur au CIO maximum.
Il est de même pour la réception du message TROUVE_ACTION, à la différence que la procédure associée ne coupera le bus des messages que si l'action, dont le CIA est maximum, n'est pas une action qui a déjà été rejetée pour l'état considéré.

cf FIGURE -III-16-

Les figures -III-15- ET -III-16 montrent en fait clairement, par les réceptions et les envois de messages, la structure et le "fonctionnement" d'EAQUE.

FIGURE -III-16-



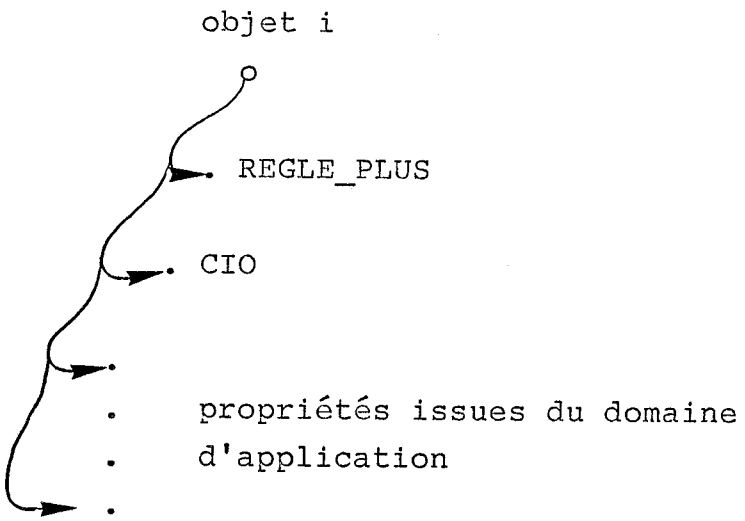
:D: LES OBJETS ET LES ACTIONS.

Les objets et les actions ne représentent que des connaissances descriptives pour les différentes procédures que nous venons de voir. Il n'existe pas de procédure attachée à leur classe respective.

-1- les objets.

Ils appartiennent à la classe OBJET. Cette clase est définie par l'Ingénieur Cognitif, puisque la sémantique de l'objet (traduite par les différentes propriétés de la classe OBJET) dépend du domaine considéré. Il lui est simplement rajouté une propriété supplémentaire, REGLE_PLUS,qui permettra d'indiquer les règles, et leur occurence, qui ont le plus augmenté le CIO de l'objet considéré.

Nous pouvons représenter un objet par la figure suivante :

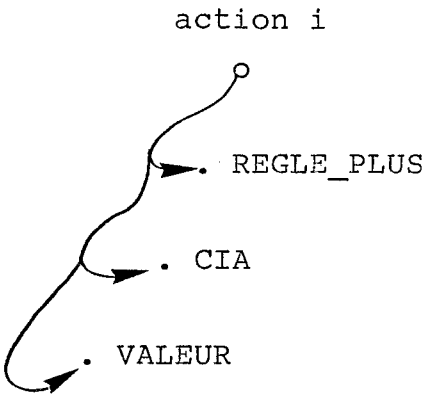


-2- les actions.

Elles appartiennent à la classe ACTION. Trois propriétés

suffisent à les décrire, ce sont : CIA, REGLE_PLUS qui joue le même rôle que pour les objets, et VALEUR qui contient la description de l'action. Ainsi, Chaque fois qu'EAQUE rencontre une nouvelle action, il crée, et lui associe, un nouvel objet LRO de la classe ACTION.

Une action a donc la forme suivante :



:E: TRACE DES MESSAGES

Donnons une suite de messages possible afin de nous faire une idée du fonctionnement d'EAQUE.
Les messages "en gras" sont envoyés par des procédures d'EAQUE, les autres dépendent de LRO.

Le message **(►OBJET moteur (marche))** est, indirectement, envoyé par l'utilisateur.

Pour une cycle de base, nous aurons un ensemble de messages semblable à ce qui suit :

```
(►TYPE règle_méta (eval_règle_méta))
          (→OBJET rm_1 (eval_règle_méta))
          ...
          (→OBJET rm_n (eval_règle_méta))
(►TYPE règle_objet (eval_règle_objet))
          (→OBJET ro_1 (eval_règle_objet))
          ...
          (→OBJET ro_m (eval_règle_objet))
                    (►OBJET moteur (trouve_objet))
                      /* le bus est coupé */
(►TYPE règle_action (eval_règle_action))
          (→OBJET ra_1 (eval_règleèaction))
          ...
          (→OBJET ra_p (eval_règle_action))
                    (►OBJET moteur (trouve_action))
                    /* l'action a déjà été rejetée lors
                       d'un backtrack précédent, le bus
                       n'est pas coupé */
          (→OBJET ra_q (eval_règle_action))
          ...
          (→OBJET ra_s (eval_règle_action))
                    (►OBJET moteur (trouve_action))
```

:F: STRUCTURE D'UN CHOIX D'ACTION

Les choix d'objet, d'action, et de relâchement d'objectifs, sont autant d'objets LRO de classes différentes. Nous donnerons, pour exemple, la structure d'un élément de la classe CHOIX_ACTION.
La classe CHOIX_ACTION est définie par un ensemble de six propriétés, qui sont :
- EST_UN : dont la valeur pointe sur l'objet LRO de la classe ACTION qui représente l'action qui vient d'être exécutée.
- NOTE : La valeur de cette propriété représente la capacité de ce choix d'action à être considéré comme point de reprise. La valeur nulle est la valeur par défaut.
- ACTIONS_REJETEES : décrit les actions rejetées, et leur note de reprise, associées à cet état.
- OBJETS_CREES : contient les objets créés par l'exécution de l'action
- OBJETS_DETRUITS : contient les objets détruits par l'exécution de l'action.
- PROG_RESTAURATION : a pour valeur, si besoin est, le programme de restauration des informations nécessaires au backtrack.

Note : Les trois instantiations d'EAQUE, ORDF, CALINT et EALOG, ont moins pour but la réalisation de systèmes performants pour les problèmes d'ordonnancement ou le calcul d'intégrales, qui ne rentrent pas dans le cadre de ce travail, mais la validation de l'approche EAQUE en tant que système instantiable à des domaines particuliers.
Les performances d'ORDF, malgré le faible temps consacré à la réalisation de la base de connaissances particulière à ce domaine, ont néanmoins été jugées très intéressantes par les spécialistes. On peut très vraisemblablement penser qu'il suffirait, pour les améliorer, d'affiner cette base de connaissances (critères de choix, etc...), sans remettre du tout en cause la structure informatique du système. Ceci fera d'ailleurs l'objet d'une thèse de troisième cycle dans le domaine d'application.

-* CHAPITRE IV *-

ORDF

-* CHAPITRE IV *-

ORDF

RESUME : Les problèmes d'ordonnancement sont des problèmes complexes. Dans certains cas (NP-complets), il n'existe pas d'algorithme de résolution en temps et espace raisonnable. C'est pourquoi l'approche heuristique et les techniques de l'Intelligence Artificielle peuvent apporter des solutions intéressantes. Ces problèmes font partie de la classe de domaines associés à EAQUE, c'est pourquoi nous l'avons instancié à l'un d'entre eux : c'est le système ORDF.

INTRODUCTION

ORDF est destiné aux problèmes de partage de ressources en nombre limité. Il se définit comme le rajout à EAQUE de deux bases de connaissances. La première concerne les propriétés et les connaissances descriptives du domaine. La deuxième est la base de règles expertes.
L'approche déclarative des critères d'ordonnancement, le retour arrière sélectif et la possibilité d'une résolution non chronologique, font de cette instantiation, à notre connaissance, une approche nouvelle des problèmes d'ordonnancement.

LES PROBLEMES D'ORDONNANCEMENT

-1- définition de la classe de problèmes

l'ordonnancement :
- "Etant donné un objectif qu'on se propose d'atteindre et dont la réalisation suppose l'exécution préalable de multiples tâches, soumises à de nombreuses contraintes, établir un ordonnancement, c'est déterminer l'ordre et le calendrier d'exécution des diverses tâches" (KAUFMANN)

le partage de ressources
-"Lorsqu'un certain nombre d'opérations, non ordonnées entre elles par des contraintes de succession, doivent être

réalisées à l'aide d'un même moyen, leur temps d'exécution est nécessairement disjoint" (KAUFFMAN)

ORDF (Ordonnancement à Ressources et Durées Fixées) est destiné aux problèmes de l'ordonnancement des tâches où intervient le partage de ressources en nombre limité, dans le cas où il est connu, pour chacune d'entre elles, sa durée ainsi que le nombre et le type de ressources qu'elle utilise. Nous illustrerons ORDF dans le domaine du bâtiment.

Un projet à réaliser est constitué d'un ensemble de **tâches** à exécuter par chaque corps de métier, les fondations, le coulage de dalles, le séchage de ces dalles en sont des exemples. Ces tâches ont une durée et utilise un certain nombre de resources différentes, la durée et les nombres d'unités de ressources utilisées seront, pour notre problème, considérés comme fixés. Ainsi le coulage d'une dalle est une tâche qui dure deux jours et nécessite un matériel de coffrage de dalle et trois hommes. Les tâches sont **partiellement ordonnées** par des **contraintes d'antériorité** ou de succession traduisant des contraintes technologiques, de conception. Par exemple les fondations sont faites avant la construction des murs. Les tâches et les contraintes d'antériorité définissent donc un graphe partiellement ordonné : **le graphe de conception** de l'ouvrage.

Il existe, nous nous en doutons, des tâches non ordonnés, c'est-à-dire qu'il n'existe pas de chemin de contraintes d'antériorité les liant. Elles peuvent donc être potentiellement réalisées en parallèle : ainsi, les piles d'un pont peuvent parfois être construites en même temps. Cependant, si ces tâches utilisent une, ou plusieurs, ressource commune et que la somme de leurs besoins est supérieure au nombre total d'unités disponibles de cette ressource, elles ne pourront pas rester en parallèle. Nous

dirons que **ces tâches sont en conflit pour cette ressource,** elles constituent **un conflit,** la ressource est dite conflictuelle. Elles devront être ordonnées. Deux solutions sont possibles, la première est de fixer les dates de lancement de chacune de ces tâches afin qu'elles ne soient plus en conflit, nous dirons alors que nous avons **ordonnancé** ces tâches; la seconde solution est de rajouter entre ces tâches des contraintes d'antériorité qui suppriment automatiquement le conflit, nous dirons alors que nous avons **ordonné** ces tâches.

Le problème est de savoir comment ordonnancer ou ordonner les tâches, de cet ordonnancement ou de cet ordre dépendra, entre autres choses, la durée du projet.

Note :
- Une même tâche peut appartenir à plusieurs conflits différents.
-2- Résoudre un conflit peut générer d'autres conflits.

-2- les méthodes de résolution

La résolution exhaustive du problème, par la détermination de toutes les chaînes possibles, et dès lors de l'optimum, n'est pas réaliste au regard de la combinatoire du problème; comme nous le montre la figure suivante où il existe 47832 chaînes différentes !

Nous pouvons distinguer trois types de méthodes :

-1- les cas particuliers

Les techniques employées sont spécifiques à ces cas particuliers. Nous citerons les problèmes à deux machines où il existe un algorithme (algorithme de Johnson) donnant l'optimum pour le critère durée (le problème est NP-complet dès que le nombre de machines est supérieur ou égal à 3).

-2- la programmation mathématique

Ce sont des techniques de programmation linéaire, non linéaire, linéaire mixte et programmation dynamique. La prise en compte du partage de ressources amène à introduire des variables bivalentes rendant ces méthodes souvent infructueuses.

-3- les méthodes heuristiques

C'est actuellement la méthode la plus prometteuse. Le but est la recherche d'une solution, ou d'un ensemble de solutions, proche de l'optimum.

Nous pouvons distinguer deux approches :

- la construction progressive : la recherche d'une solution est faite en construisant pas à pas un ordonnancement en fonction d'une discrétisation du temps et des tâches déjà ordonnancées, et en prenant en compte différents critères.
- les procédures d'exploration (S.E.P branch and bound) : ces procédures sont basées sur l'exploration d'un arbre. Ordonner deux tâches correspond à séparer l'ensemble des solutions possibles en deux sous ensembles, une fonction d'évaluation est appliquée à chacun d'eux afin de guider le choix.

-3- ORDF

ORDF, face à l'ensemble des conflits, procède de la façon suivante :

-1- il choisit le conflit le plus urgent à résoudre (puisque ce choix peut avoir des conséquences importantes) en fonction d'un certain nombre de critères.

-2- le conflit étant choisi, il le résout en ordonnant les tâches du conflit, prises deux par deux. C'est-à-dire en leur rajoutant une contrainte d'antériorité, ce en fonction d'un certain nombre de connaissances propres au domaine.

-3- enfin, ORDF a la possibilité de pouvoir revenir à un état antérieur lorsque l'état courant est "jugé" insatisfaisant.

LA BASE DU DOMAINE

:A: LE DOMAINE
:B: EAQUE

:A: LE DOMAINE.

le travail de l'Ingénieur Cognitif se divise en cinq points :

-1- la définition des classes et des objets LRO pour la description du contexte.

Deux classes sont suffisantes pour décrire le problème à traiter. Ce sont les classes TACHE et RESSOURCE. La classe TACHE représente l'ensemble des tâches à réaliser pour le projet considéré, elle est définie par les propriétés PRED (prédécesseurs), SUC (successeurs), RES (ressources utilisées), DUREE (durée de la tâche) de valeur par défaut 0, DTOT (date au plus tôt), et DTARD (date au plus tard). PRED et SUC permettent de décrire le graphe de conception de l'ouvrage, DTOT et DTARD sont des propriétés directement liées à la résolution du problème. La classe RESSOURCE représente l'ensemble des ressources disponibles, elle est définie par les propriétés NBRE (quantités d'unités de ressources disponibles) et IMPORTANCE. Cette dernière propriété permet de hiérarchiser les ressources entre elles, elle intervient pour la résolution du problème.

-2- la définition des objets.

Elle n'est pas déduite directement de la description du problème. Un objet représente un conflit. C'est un sous problème sur lequel sera focalisée la résolution. La classe OBJET est définie par les propriétés CONF_SOMMETS et CONF_RESSOURCE (hormis la propriété CIO rajoutée par EAQUE). CONF_SOMMETS représente l'ensemble des sommets en conflit pour une même ressource, valeur de CONF_RESSOURCE. Les objets sont déterminés initialement et à chaque fin de cycle, de la façon suivante : Pour toute ressource, et pour chaque ensemble de tâches en parallèle, qui accèdent à la même ressource R, qui ont toutes une intersection non vide de leur intervalle °dtot , dtot+durée§, et que la somme de leur besoin est supérieure au nombre d'unités disponibles de R; il est créé un objet dont la valeur de CONF_SOMMETS est cet ensemble, et dont la valeur de CONF_RESSOURCE est R.

Note : Ce choix de définition pour le conflit implique, lorsque le problème est résolu, une mise à jour des dates au plus tard, qui assurera l'inexistence de conflit pour des dates de lancement comprises entre les dates au plus tôt et au plus tard, à savoir le minimum de la date au plus tard et de la somme, date au plus tôt plus durée.

-3- la définition des actions.

Il existe une seule action qui consiste à mettre une tâche avant (ou après) une autre, elle met à jour les propriétés PRED et SUC de ces deux tâches.

-4- la définition des propriétés du domaine.

Elles sont relativement nombreuses. Nous citerons : le calcul des dates au plus tôt et au plus tard de toutes les tâches (ces dates sont mises à jour après chaque cycle); des fonctions portant sur les ensembles : relation d'ordre, inclusion... ; des fonctions portant sur le graphe (de

conception) : nombre de successeurs et/ou de prédécesseurs total, direct... ; la simulation d'un ordre particulier entre deux tâches sur la date au plus tard de la dernière tâche... Et également l'action inverse, pour le retour arrière, qui restaure les valeurs des propriétés PRED et SUC.

-5- la particularisation des squelettes d'analyse syntaxique

-a- la description du problème

La définition des objets LRO des classes TACHE et RESSOURCE doit respecter la syntaxe définie par l'Ingénieur Cognitif. Ainsi, la valeur de la propriété RES est une liste de couples, où le premier élément est le nombre d'unités de ressources utilisées, et le deuxième, le nom de cette ressource.

-b- les conditions et les actions

Les conditions et les actions sont, évidemment, des fonctions LISP, cependant l'analyse syntaxique définie par l'Ingénieur Cognitif permet l'écriture de fonctions préfixées et infixées à tous les niveaux, avec suppression des parenthéses au premier niveau.

Exemples : ?C1 AVANT ?C2

```
(((NBRE_RES ?S1)*(NBRE_RES ?S1))/(MARGE ?S1))
 >
(((NBRE_RES ?S2)*(NBRE_RES ?S2))/(MARGE ?S2))
```

:B: EAQUE

Cette partie correspond à la programmation de la structure de contrôle d'EAQUE.

ORDF utilise la grande majorité des options proposées par EAQUE.
A chaque cycle de contrôle il peut exister plusieurs objets, ORDF utilise des règles objets pour en choisir un. La fonction de notation des objets, ainsi que les CIO maximum et minimum, sont ceux d'EAQUE par défaut.

A l'heure actuelle ORDF n'utilise pas de méta-règle.
Quant au choix d'action, les deux types de règles actions, conseillant et déconseillant une action, sont utilisés. La fonction de notation des actions est celle d'EAQUE par défaut, et le CIA maximum est mis à 15, ce qui assure, dans la version actuelle d'EAQUE, l'application d'au moins trois règles actions pour qu'une action soit choisie.

Enfin, pour le retour arrière, la fonction de notation des choix est celle d'EAQUE par défaut.

LA BASE DE REGLES

Nous avons actuellement dans ORDF deux types de règles : les règles objets et les règles actions. Nous en verrons des exemples dans l'annexe au paragraphe suivant.

En ce qui concerne les règles actions, nous pouvons distinguer quatre sous ensembles. Le premier prend en compte des critères relatifs à la durée, le deuxième s'attache aux ressources utilisées, le troisième s'intéresse à la structure du graphe, et le dernier est constitué d'une seule règle, la règle par défaut qui ordonne "au hasard" deux tâches de l'objet choisi lorsqu'aucune règle action n'a été activée (son CIR nul assure qu'elle sera appliquée en dernier).

ANNEXES

ORDF

EXEMPLES DE REGLES

POUR LES CRITERES DE CHOIX D'OBJET

-1- Si C1 et C2 sont deux conflits et si l'ensemble des sommets de C1 sont inclus dans C2, alors il est préférable, avec une importance de 5, de commencer par résoudre le conflit dans son ensemble et non localement. C'est à dire choisir C2 plutôt que C1.

```
REGLE OBJET RO1 : CIR 5 :
    SI ?C1 INCLUS ?C2
    ALORS ?C2 EST A PREFERER A ?C1 ;
```

-2- Si C1 et C2 sont deux conflits et si la ressource conflictuelle associée à C1 est plus importante que celle associée à C2, alors il est préférable, avec une importance de 3, de commencer par résoudre le conflit C1 plutôt que le conflit C2.

```
RO RO3 : CIR 3 :
    SI ?C1 RES_PLUS_IMPORTANT ?C2
    ALORS ?C1 PREF ?C2 ;
```

-3- Si C1 et C2 sont deux conflits et si C1 a plus de sommets que C2, alors il est préférable, avec une importance de 1, de commencer par résoudre le conflit le plus "gros". C'est à dire choisir C1 plutôt que C2.

```
RO RO4 : CIR 1 :
    SI ?C1 PLUS_SOMMETS ?C2
    ALORS ?C1 PREF ?C2 ;
```

ORDF

EXEMPLES DE REGLES ACTIONS

A: PRISE EN COMPTE DES DUREES.

-1- Si S1 et S2 sont deux sommets du conflit courant à résoudre, et si la date au plus tôt de S1 est inférieure à la date au plus tôt de S2, alors il est préférable, avec une importance de 5, de mettre S1 avant S2.

```
REGLE ACTION RA13 : CIR 5 :
    DOMAINE VARIABLES :: ?S1 = CONF_SOMMETS /
                         ?S2 = CONF_SOMMETS
    SI (DTOT ?S1) < (DTOT ?S2)
    ALORS IL EST PREFERABLE DE METTRE ?S1 AVANT ?S2 ;
```

-2- Règle "duale": Si S1 et S2 sont deux sommets du conflit courant à résoudre, et si la date au plus tôt de S1 est inférieure à la date au plus tôt de S2, alors il est préférable, avec une importance de 2, de ne pas mettre S2 avant S1.
Il n'y a donc pas forcemment équivalence, du point de vue de l'importance, entre vouloir choisir l'action "mettre S1 avant S2" et vouloir rejeter l'action "mettre S2 avant S1".

```
REGLE ACTION RA14 : CIR 2 :
    DOMAINE VARIABLES :: ?S1 = CONF_SOMMETS /
                         ?S2 = CONF_SOMMETS
    SI (DTOT ?S1) < (DTOT ?S2)
    ALORS IL EST PREFERABLE DE NE PAS METTRE ?S2 AVANT ?S1 ;
```

-3- Si S1 et S2 sont deux sommets du conflit courant à résoudre et si le rapport du produit du nombre d'unités de la ressource conflictuelle utilisées par S1 * sa durée par la marge de S1 est supérieure au même rapport pour S2, alors il est préférable, avec une importance de 7, de mettre en premier le sommet le plus sujet à problème (marge réduite, grande durée, beaucoup de ressources utilisées). C'est à dire de mettre S1 avant S2.

```
REGLE ACTION RA19 : CIR 7 :
     DOMAINE VARIABLES :: ?S1 = CONF_SOMMETS /
                          ?S2 = CONF_SOMMETS
     SI ( ((NBRE_RES ?S1) * (DUREE ?S1)) / (MARGE ?S1) ) >
        ( ((NBRE_RES ?S2) * (DUREE ?S2)) / (MARGE ?S2)) )
     ALORS IL EST PREFERABLE DE METTRE ?S1 AVANT ?S2 ;
```

-4- Si S1 et S2 sont deux sommets du conflit courant à résoudre, et si la marge de S1 est plus petite que la marge de S2, alors il est préférable, avec une importance de 5, de mettre S1 avant S2.

```
REGLE ACTION RA15 : CIR 5 :
     DOMAINE VARIABLES :: ?S1 = CONF_SOMMETS /
                          ?S2 = CONF_SOMMETS
     SI (MARGE ?S1) < (MARGE ?S2)
     ALORS IL EST PREFERABLE DE METTRE ?S1 AVANT ?S2 ;
```

-5-Si S1 et S2 sont deux sommets du conflit courant à résoudre, et si la date au plus tôt de la dernière tâche, dans le cas où l'on mettrait S1 avant S2, est supérieure à la date au plus tôt de la dernière tâche dans le cas où c'est S2 qui serait avant S1, alors il est préférable, avec une importance de 6, de choisr l'action qui va minimiser le temps. C'est à dire mettre S2 avant S1.

```
RA RA44 : CIR 6 :
     VAR :: ?S1 = CONF_SOMMETS /
            ?S2 = CONF_SOMMETS
     SI (SIMULATION ?S1 AVANT ?S2) > (SIMULATION ?S2 AVANT ?S1)
     ALORS METTRE ?S2 AVANT ?S1 ;
```

B: PRISE EN COMPTE DES RESSOURCES.

-5- Si S1 et S2 sont deux sommets du conflit courant à résoudre associé à la ressource R, et si S1 a un besoin en ressource R supérieur à celui de S2, alors il est préférable, avec une importance de 4, de mettre S1 avant S2.

```
RA RA29 : CIR 4 :
    VAR :: ?S1 = CONF_SOMMETS /
           ?S2 = CONF_SOMMETS
    SI (NBRE_RES ?S1) > (NBRE_RES ?S2)
    ALORS METTRE ?S1 AVANT ?S2 ;
```

6- Nous avons également la règle duale avec une importance de 2.

```
RA RA30 : CIR 2 :
    VAR :: ?S1 = CONF_SOMMETS /
           ?S2 = CONF_SOMMETS
    SI (NBRE_RES ?S1) > (NBRE_RES ?S2)
    ALORS NE PAS METTRE ?S2 AVANT ?S1 ;
```

C: PRISE EN COMPTE DE LA STRUCTURE.

1 Si S1 et S2 sont deux sommets du conflit courant à résoudre, et si le nombre de successeurs total de S1 et plus grand que celui de S2, alors il est préférable, avec une importance de 7, de mettre S1 avant S2.

```
RA RA35 : CIR 7 :
    VAR :: ?S1 = CONF_SOMMETS /
           ?S2 = CONF_SOMMETS
    SI (NBRE_SUC_TOTAL ?S1) > (NBRE_SUC_TOTAL ?S2)
    ALORS METTRE ?S1 AVANT ?S2 ;
```

2 Si S1 et S2 sont deux sommets du conflit courant à résoudre, et si l'ensemble des prédécesseurs directs de S1 est inclus dans l'ensemble de tous les prédécesseurs possibles de S2, alors il est préférable, avec une importance de 7, de mettre S1 avant S2.

```
RA RA40 : CIR 7 :
    VAR :: ?S1 = CONF_SOMMETS /
           ?S2 = CONF_SOMMETS
    SI (PRED ?S1) C (PRED_TOTAL ?S2)
    ALORS METTRE ?S1 AVANT ?S2 ;
```

3 Nous avons les règles duales de 1 et 2, et les règles équivalentes pour les successeurs directs et tous les successeurs possibles.

D: PAR DEFAUT.

1 Si pour le conflit courant à résoudre il n'existe aucune action applicable, alors, avec une importance nulle, car cette règle ne doit être appliquée qu'après avoir essayé toutes les autres; prendre deux sommets "au hasard", et mettre l'un avant l'autre.

```
RA  RA50 : CIR 0 :
     VAR :: ?S1 = CONF_SOMMETS /
            ?S2 = CONF_SOMMETS
     SI ?S1 AUCUNE_ACTION ?S2
     ALORS METTRE ?S1 AVANT ?S2 ;
```

ORDF

EXEMPLE DE RESOLUTION D'UN PROBLEME

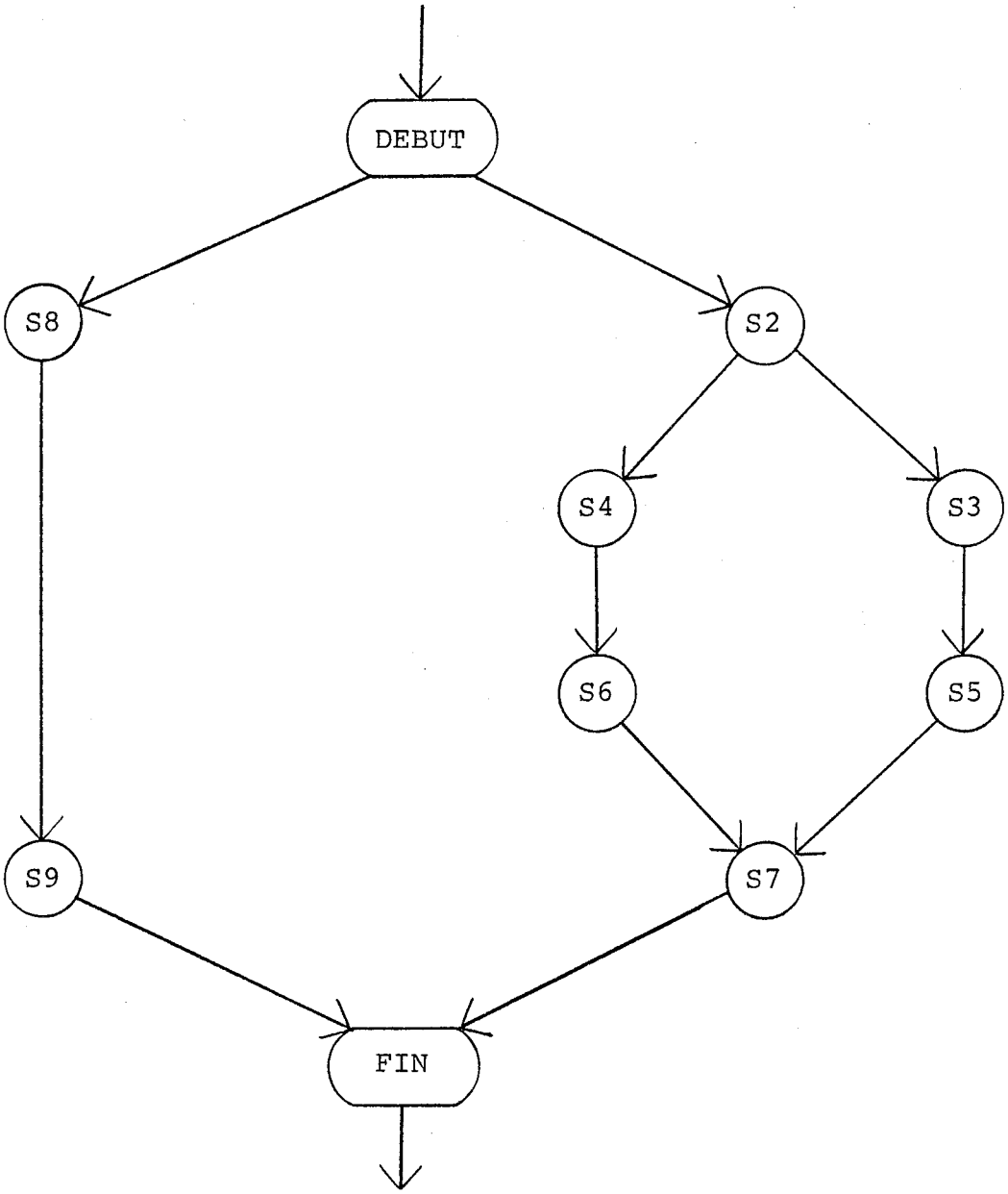
DESCRIPTION DU PROBLEME

%----------%
% EXEMPLE %
%----------%

```
%..... description du probleme .....%

DEBUT    TACHE_SIMPLE      SUC = (SOMMET2 SOMMET8) ;

SOMMET2  TACHE_SIMPLE      PRED = (DEBUT) /
                           SUC = (SOMMET3 SOMMET4) /
                           DUREE = 6 /
                           RES = ((C1 2) (U1 1)) ;

SOMMET3  TACHE_SIMPLE      PRED = (SOMMET2) /
                           SUC = (SOMMET5) /
                           DUREE = 10 /
                           RES = ((C2  2) (U1 1)) ;

SOMMET4  TACHE_SIMPLE      PRED = (SOMMET2) /
                           SUC = (SOMMET6) /
                           DUREE = 9 /
                           RES = ((C3 3) (U1 1)) ;

SOMMET5  TACHE_SIMPLE      PRED = (SOMMET3) /
                           SUC = (SOMMET7) /
                           DUREE = 8 /
                           RES = ((C3 2) (U1 1)) ;

SOMMET6  TACHE_SIMPLE      PRED = (SOMMET4) /
                           SUC = (SOMMET7) /
                           DUREE = 4 /
                           RES = ((C2 1) (U1 1)) ;

SOMMET7  TACHE_SIMPLE      PRED = (SOMMET5 SOMMET6) /
                           SUC = (FIN) /
                           DUREE = 10 /
                           RES = ((C3 2) (U1 1)) ;

SOMMET8  TACHE_SIMPLE      PRED = (DEBUT) /
                           SUC = (SOMMET9) /
                           DUREE = 12 /
                           RES = ((C1 2) (U2 1)) ;

SOMMET9  TACHE_SIMPLE      PRED = (SOMMET8) /
                           SUC = (FIN) /
                           DUREE = 12 /
                           RES = ((C2 3) (U2 1)) ;

FIN  TACHE_SIMPLE          PRED = (SOMMET7 SOMMET9) ;
```

```
%.......... Ressources ..........%

C1   RESSOURCE              NBRE = 2 /
                            IMPORTANCE = 5 ;

C2   RESSOURCE              NBRE = 3 /
                            IMPORTANCE = 6 ;

C3   RESSOURCE              NBRE = 3 /
                            IMPORTANCE = 4 ;

U1   RESSOURCE              NBRE = 1 /
                            IMPORTANCE = 2 ;

U2   RESSOURCE              NBRE = 1 /
                            IMPORTANCE = 1 ;


%.......... Objectifs ..........%

OBJECTIFS = 10 : (DTARD FIN) < 100  ;


FIN!
```

GRAPHE CORRESPONDANT



QUELQUES CYCLES DE RESOLUTION...

Note : Il faut se rappeler que REGLES_APPLIQUEES n'indiquent que les règles qui surnotent l'action (il est préférable de faire ...).

```
"============================================================"

CHOIX_DU_CONFLIT

RESSOURCE_CONFLICTUELLE_:_C2
SOMMETS_:_(SOMMET3 SOMMET6 SOMMET9)
NOTE_OBTENUE_:_11
REGLES_APPLIQUEES_:_((R03 2) (R01 1))

"objet_choisi


"Regle_et_son_occurence_ayant_le_plus_contribue_a_ce_choix:"
R03 2



"============================================================"

CHOIX_DE_L_ACTION

ACTION_CHOISIE_:_(METTRE SOMMET3 AVANT SOMMET6)
NOTE_OBTENUE_:_21
REGLES_APPLIQUEES_:_((RA42 1) (RA40 1) (RA35 1))

"action_choisie


"Regle_et_son_occurence_ayant_le_plus_contribue_a_ce_choix:"
RA42 1



"============================================================"

CHOIX_DU_CONFLIT

RESSOURCE_CONFLICTUELLE_:_C2
SOMMETS_:_(SOMMET3 SOMMET9)
NOTE_OBTENUE_:_9
REGLES_APPLIQUEES_:_((R03 3))

"objet_choisi


"Regle_et_son_occurence_ayant_le_plus_contribue_a_ce_choix:"
R03 3
```

```
"================================================================"

CHOIX_DE_L_ACTION

ACTION_CHOISIE_:_(METTRE SOMMET3 AVANT SOMMET9)
NOTE_OBTENUE_:_20
REGLES_APPLIQUEES_:_((RA44 1) (RA42 1) (RA35 1))

"action_choisie


"Regle_et_son_occurence_ayant_le_plus_contribue_a_ce_choix:"
RA42 1



"================================================================"

CHOIX_DU_CONFLIT

RESSOURCE_CONFLICTUELLE_:_C2
SOMMETS_:_(SOMMET6 SOMMET9)
NOTE_OBTENUE_:_7
REGLES_APPLIQUEES_:_((R03 3))

"objet_choisi


"Regle_et_son_occurence_ayant_le_plus_contribue_a_ce_choix:"
R03 3



"================================================================"

CHOIX_DE_L_ACTION

ACTION_CHOISIE_:_(METTRE SOMMET6 AVANT SOMMET9)
NOTE_OBTENUE_:_18
REGLES_APPLIQUEES_:_((RA15 1) (RA44 1) (RA42 1) (RA35 1))

"action_choisie


"Regle_et_son_occurence_ayant_le_plus_contribue_a_ce_choix:"
RA42 1
```

```
"================================================================"

CHOIX_DU_CONFLIT

RESSOURCE_CONFLICTUELLE_:_C1
SOMMETS_:_(SOMMET2 SOMMET8)
NOTE_OBTENUE_:_7
REGLES_APPLIQUEES_:_((RO2 1) (RO3 2))

"objet_choisi


"Regle_et_son_occurence_ayant_le_plus_contribue_a_ce_choix:"
RO3 2



"================================================================"

CHOIX_DE_L_ACTION

ACTION_CHOISIE_:_(METTRE SOMMET2 AVANT SOMMET8)
NOTE_OBTENUE_:_20
REGLES_APPLIQUEES_:_((RA44 1) (RA42 1) (RA35 1))

"action_choisie '


"Regle_et_son_occurence_ayant_le_plus_contribue_a_ce_choix:"
RA42 1



"================================================================"

CHOIX_DU_CONFLIT

RESSOURCE_CONFLICTUELLE_:_U1
SOMMETS_:_(SOMMET3 SOMMET4)
NOTE_OBTENUE_:_1
REGLES_APPLIQUEES_:_((RO2 1))

"objet_choisi


"Regle_et_son_occurence_ayant_le_plus_contribue_a_ce_choix:"
RO2 1
```

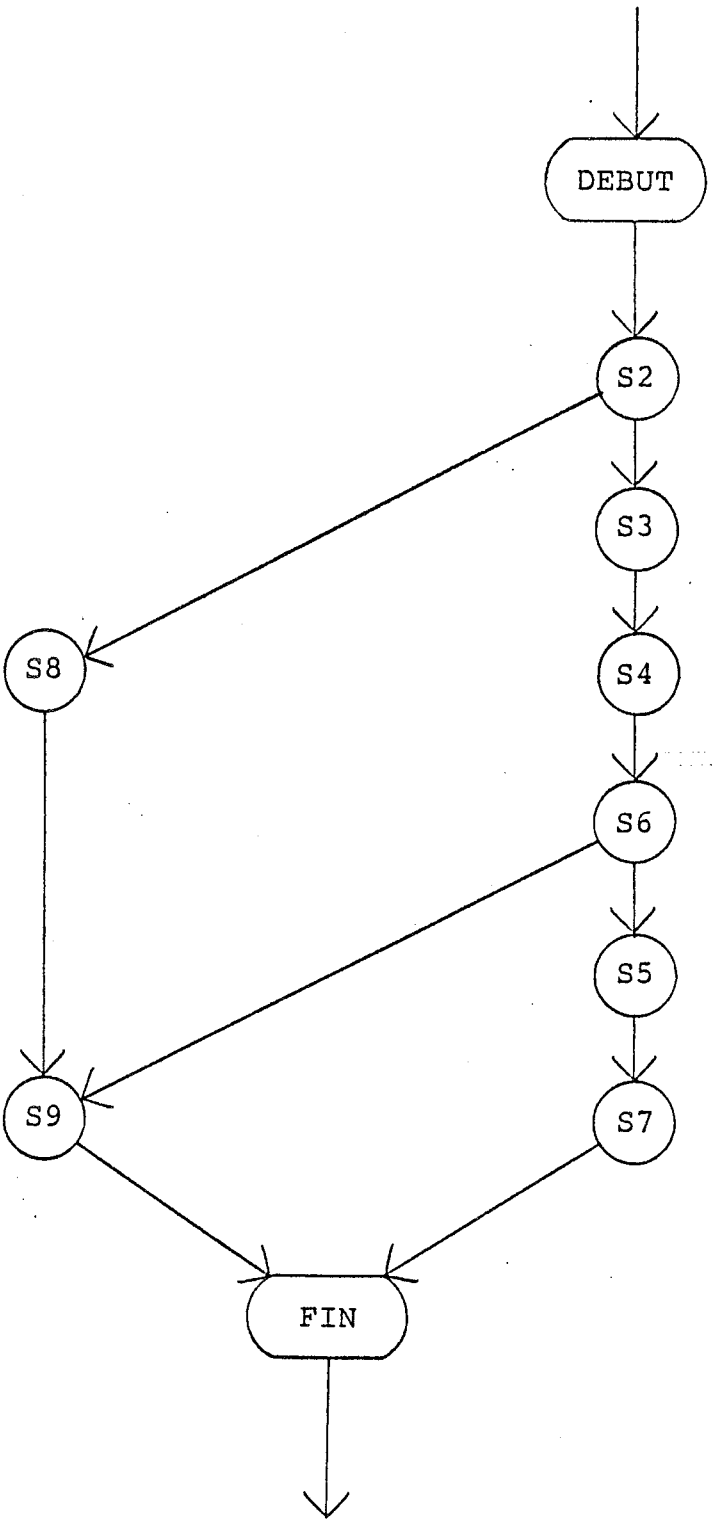
RESULTAT

```
                    "LE_PROBLEME_EST_RESOLU
                    "************************"

"============================================================="

FIN

RES:NIL
DTARD:47
DTOT:47
DUREE:0
SUC:NIL
PRED:(SOMMET7 SOMMET9)

NIL


"============================================================="

SOMMET9

RES:((C2 3) (U2 1))
DTARD:35
DTOT:29
DUREE:12
SUC:(FIN)
PRED:(SOMMET6 SOMMET3 SOMMET8)

NIL


"============================================================="

SOMMET8

RES:((C1 2) (U2 1))
DTARD:23
DTOT:6
DUREE:12
SUC:(SOMMET9)
PRED:(SOMMET2 DEBUT)

NIL
```

```
"============================================================="

SOMMET7

RES:((C3 2) (U1 1))
DTARD:37
DTOT:37
DUREE:10
SUC:(FIN)
PRED:(SOMMET5 SOMMET6)

NIL

"============================================================="

SOMMET6

RES:((C2 1) (U1 1))
DTARD:25
DTOT:25
DUREE:4
SUC:(SOMMET5 SOMMET9 SOMMET7)
PRED:(SOMMET3 SOMMET4)

NIL


"============================================================="

SOMMET5

RES:((C3 2) (U1 1))
DTARD:29
DTOT:29
DUREE:8
SUC:(SOMMET7)
PRED:(SOMMET6 SOMMET4 SOMMET3)

NIL


"============================================================="

SOMMET4

RES:((C3 3) (U1 1))
DTARD:16
DTOT:16
DUREE:9
SUC:(SOMMET5 SOMMET6)
PRED:(SOMMET3 SOMMET2)

NIL
```

```
"============================================================="

SOMMET3

RES:((C2 2) (U1 1))
DTARD:6
DTOT:6
DUREE:10
SUC:(SOMMET4 SOMMET9 SOMMET6 SOMMET5)
PRED:(SOMMET2)

NIL


"============================================================="

SOMMET2

RES:((C1 2) (U1 1))
DTARD:0
DTOT:0
DUREE:6
SUC:(SOMMET8 SOMMET3 SOMMET4)
PRED:(DEBUT)

NIL


"============================================================="

DEBUT

RES:NIL
DTARD:0
DTOT:0
DUREE:0
SUC:(SOMMET2 SOMMET8)
PRED:NIL

NIL
```

GRAPHE CORRESPONDANT



POUR OBTENIR UNE AUTRE SOLUTION...

Pour obtenir une autre solution, le système considère que la solution obtenue n'est pas satisfaisante, génère une contradiction et force ainsi un retour arrière. Dans l'exemple ci dessous, il est décidé de rejeter l'action "mettre S3 avant S6", le conflit restant le même (S3 , S6 , S9), et de prendre la nouvelle action "mettre S3 avant S9".

```
"VOULEZ-VOUS_UNE_AUTRE_SOLUTION_?"
"================================================================="

&PT_REPRISE_TRANS&

PROG_RESTAURATION:NIL
OBJETS_DETRUITS:(RESG0003 RESG0000)
OBJETS_CREES:(TRANSG0009 TRANSG0008)
TRANS_REJETEES:NIL
NOTE:7
EST_UN:TRANSG0007

"choix_de_transition_pris_comme_point_de_backtrack"


"================================================================="

TRANSG0007

REGLE_PLUS:((RA42 1) (RA40 1) (RA35 1))
&DOM_VAR&:NIL
&VAL_TRANS&:(METTRE SOMMET3 AVANT SOMMET6)
&CIT&:21

NIL


"================================================================="

CHOIX_DE_L_ACTION

ACTION_CHOISIE_:_(METTRE SOMMET3 AVANT SOMMET9)
NOTE_OBTENUE_:_13
REGLES_APPLIQUEES_:_((RA44 1) (RA35 1))

"action_choisie"


"Regle_et_son_occurence_ayant_le_plus_contribue_a_ce_choix:"
RA35 1
```

ORDF

EXEMPLE DU PONT

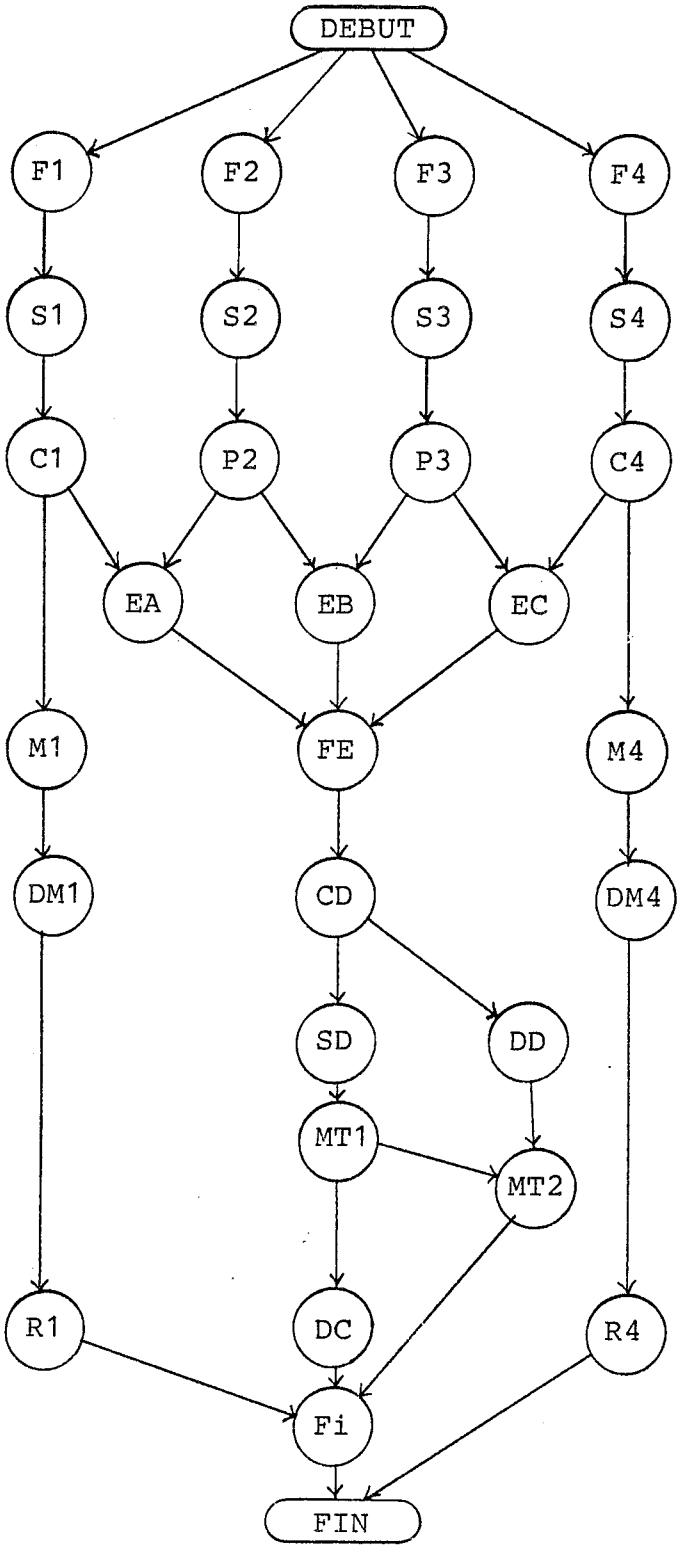
DESCRIPTION DU PROJET PONT

```
%-------------------------------------------------------------------------%
%                          PROBLEME  DU  PONT                             %
%-------------------------------------------------------------------------%


%.....  description des taches  ......%


DEBUT       TACHE_SIMPLE      SUC = (FOUILLE1 FOUILLE2 FOUILLE3 FOUILLE4) ;

FOUILLE1    TACHE_SIMPLE      PRED = (DEBUT) /
                              SUC = (SEMELLE1) /
                              DUREE = 2 /
                              RES = ( (HOMMES 3) (PELLE 1) ) ;

FOUILLE2    TACHE_SIMPLE      PRED = (DEBUT) /
                              SUC = (SEMELLE2) /
                              DUREE = 2 /
                              RES = ( (HOMMES 3) (PELLE 1) ) ;

FOUILLE3    TACHE_SIMPLE      PRED = (DEBUT) /
                              SUC = (SEMELLE3) /
                              DUREE = 2 /
                              RES = ( (HOMMES 3) (PELLE 1) ) ;

FOUILLE4    TACHE_SIMPLE      PRED = (DEBUT) /
                              SUC = (SEMELLE4) /
                              DUREE = 2 /
                              RES = ( (HOMMES 3) (PELLE 1) ) ;

SEMELLE1    TACHE_SIMPLE      PRED = (FOUILLE1) /
                              SUC = (CULEE1) /
                              DUREE = 4 /
                              RES = ( (HOMMES 4) ) ;

SEMELLE2    TACHE_SIMPLE      PRED = (FOUILLE2) /
                              SUC = (PILE2) /
                              DUREE = 3 /
                              RES = ( (HOMMES 4) ) ;

SEMELLE3    TACHE_SIMPLE      PRED = (FOUILLE3) /
                              SUC = (PILE3) /
                              DUREE = 3 /
                              RES = ( (HOMMES 4) ) ;

SEMELLE4    TACHE_SIMPLE      PRED = (FOUILLE4) /
                              SUC = (CULEE4) /
                              DUREE = 3 /
                              RES = ( (HOMMES 4) ) ;

CULEE1      TACHE_SIMPLE      PRED = (SEMELLE1) /
                              SUC = (MUR1 ETAIEMENTA) /
                              DUREE = 8 /
                              RES = ( (HOMMES 7) (COFFRAGE_CULEE 1) ) ;
```

```
CULEE4      TACHE_SIMPLE      PRED = (SEMELLE4) /
                              SUC = (MUR4 ETAIEMENTC) /
                              DUREE = 8 /
                              RES = ( (HOMMES 7) (COFFRAGE_CULEE 1) ) ;

PILE2       TACHE_SIMPLE      PRED = (SEMELLE2) /
                              SUC = (ETAIEMENTA ETAIEMENTB) /
                              DUREE = 5 /
                              RES = ( (HOMMES 5) (COFFRAGE_PILE 1) ) ;

PILE3       TACHE_SIMPLE      PRED = (SEMELLE3) /
                              SUC = (ETAIEMENTB ETAIEMENTC) /
                              DUREE = 5/
                              RES = ( (HOMMES 5) (COFFRAGE_PILE 1) ) ;

MUR1        TACHE_SIMPLE      PRED = (CULEE1) /
                              SUC = (DURCISSEMENT_MUR1) /
                              DUREE = 3 /
                              RES = ( (HOMMES 4) (COFFRAGE_MUR 1) ) ;

MUR4        TACHE_SIMPLE      PRED = (CULEE4) /
                              SUC = (DURCISSEMENT_MUR4) /
                              DUREE = 3 /
                              RES = ( (HOMMES 4) (COFFRAGE_MUR 1) ) ;

DURCISSEMENT_MUR1 TACHE_SIMPLE   PRED = (MUR1) /
                                 SUC = (REMBLAI1) /
                                 DUREE = 8 ;

DURCISSEMENT_MUR4 TACHE_SIMPLE   PRED = (MUR4) /
                                 SUC = (REMBLAI4) /
                                 DUREE = 8 ;

REMBLAI1    TACHE_SIMPLE      PRED = (DURCISSEMENT_MUR1) /
                              SUC = (FIN) /
                              DUREE = 10 /
                              RES = ( (HOMMES 7) (PELLE 1) ) ;

REMBLAI4    TACHE_SIMPLE      PRED = (DURCISSEMENT_MUR4) /
                              SUC = (FIN) /
                              DUREE = 10 /
                              RES = ( (HOMMES 7) (PELLE 1) ) ;

ETAIEMENTA TACHE_SIMPLE       PRED = (CULEE1 PILE2) /
                              SUC = (FERRAILLAGE) /
                              DUREE = 6 /
                              RES = ( (HOMMES 6) ) ;

ETAIEMENTB TACHE_SIMPLE       PRED = (PILE2 PILE3) /
                              SUC = (FERRAILLAGE) /
                              DUREE = 10 /
                              RES = ( (HOMMES 10) ) ;

ETAIEMENTC TACHE_SIMPLE       PRED = (PILE3 CULEE4) /
                              SUC = (FERRAILLAGE) /
                              DUREE = 6 /
                              RES = ( (HOMMES 6) ) ;
```

```
FERRAILLAGE TACHE_SIMPLE     PRED = (ETAIEMENTA ETAIEMENTB ETAIEMENTC) /
                             SUC = (COULAGE_DALLE) /
                             DUREE = 5 /
                             RES = ( (HOMMES 5) ) ;

COULAGE_DALLE TACHE_SIMPLE   PRED = (FERRAILLAGE) /
                             SUC = (SECHAGE_DALLE DURCISSEMENT_DALLE) /
                             DUREE = 1 /
                             RES = ( (HOMMES 10) ) ;

SECHAGE_DALLE TACHE_SIMPLE   PRED = (COULAGE_DALLE) /
                             SUC = (MISE_EN_TENSION1) /
                             DUREE = 4 ;

DURCISSEMENT_DALLE TACHE_SIMPLE   PRED = (COULAGE_DALLE) /
                                  SUC = (MISE_EN_TENSION2) /
                                  DUREE = 8 ;
MISE_EN_TENSION1 TACHE_SIMPLE     PRED = (SECHAGE_DALLE) /
                                  SUC = (MISE_EN_TENSION2 DECOFFRAGE) /
                                  DUREE = 1 /
                                  RES = ( (HOMMES 7) ) ;

MISE_EN_TENSION2 TACHE_SIMPLE     PRED = (DURCISSEMENT_DALLE MISE_EN_TENSION1) /
                                  SUC = (FINITIONS) /
                                  DUREE = 1 /
                                  RES = ( (HOMMES 7) ) ;

DECOFFRAGE TACHE_SIMPLE      PRED = (MISE_EN_TENSION1) /
                             SUC = (FINITIONS) /
                             DUREE = 5 /
                             RES = ( (HOMMES 6) ) ;

FINITIONS  TACHE_SIMPLE      PRED = (MISE_EN_TENSION2 DECOFFRAGE) /
                             SUC = (FIN) /
                             DUREE = 12 /
                             RES = ( (HOMMES 8) ) ;

FIN        TACHE_SIMPLE      PRED = (REMBLAI1 REMBLAI4 FINITIONS ) ;


%..... description des ressources .....%

PELLE              RESSOURCE          NBRE = 1 /
                                      IMPORTANCE = 3 ;

COFFRAGE_CULEE     RESSOURCE          NBRE = 1 /
                                      IMPORTANCE = 2 ;

COFFRAGE_PILE      RESSOURCE          NBRE = 1 /
                                      IMPORTANCE = 2 ;

COFFRAGE_MUR       RESSOURCE          NBRE = 1 /
                                      IMPORTANCE = 2 ;

HOMMES             RESSOURCE          NBRE = 20 /
                                      IMPORTANCE = 1 ;
```

GRAPHE CORRESPONDANT



RESULTAT

```
                    "LE_PROBLEME_EST_RESOLU
                    "************************"

"=============================================================="

FIN

RES:NIL
DTARD:53
DTOT:63
DUREE:0
SUC:NIL
PRED:(REMBLAI1 REMBLAI4 FINITIONS)

NIL


"=============================================================="

FINITIONS

RES:((HOMMES 8))
DTARD:51
DTOT:51
DUREE:12
SUC:(FIN)
PRED:(MISE_EN_TENSION2 DECOFFRAGE)

NIL


"=============================================================="

DECOFFRAGE

RES:((HOMMES 6))
DTARD:46
DTOT:46
DUREE:5
SUC:(FINITIONS)
PRED:(MISE_EN_TENSION1)

NIL
```

```
"================================================================="

MISE_EN_TENSION2

RES:((HOMMES 7))
DTARD:50
DTOT:49
DUREE:1
SUC:(FINITIONS)
PRED:(DURCISSEMENT_DALLE MISE_EN_TENSION1)

NIL

"================================================================="

MISE_EN_TENSION1

RES:((HOMMES 7))
DTARD:45
DTOT:45
DUREE:1
SUC:(MISE_EN_TENSION2 DECOFFRAGE)
PRED:(SECHAGE_DALLE)

NIL


"================================================================="

DURCISSEMENT_DALLE

RES:NIL
DTARD:42
DTOT:41
DUREE:8
SUC:(MISE_EN_TENSION2)
PRED:(COULAGE_DALLE)

NIL


"================================================================="

SECHAGE_DALLE

RES:NIL
DTARD:41
DTOT:41
DUREE:4
SUC:(MISE_EN_TENSION1)
PRED:(COULAGE_DALLE)

NIL
```

```
"================================================================="

COULAGE_DALLE

RES:((HOMMES 10))
DTARD:40
DTOT:40
DUREE:1
SUC:(REMBLAI1 SECHAGE_DALLE DURCISSEMENT_DALLE)
PRED:(FERRAILLAGE)

NIL

"================================================================="

FERRAILLAGE

RES:((HOMMES 5))
DTARD:35
DTOT:35
DUREE:5
SUC:(COULAGE_DALLE)
PRED:(ETAIEMENTA ETAIEMENTB ETAIEMENTC)

NIL

"================================================================="

ETAIEMENTC

RES:((HOMMES 6))
DTARD:29
DTOT:27
DUREE:6
SUC:(FERRAILLAGE)
PRED:(CULEE1 PILE3 CULEE4)

NIL

"================================================================="

ETAIEMENTB

RES:((HOMMES 10))
DTARD:19
DTOT:19
DUREE:10
SUC:(ETAIEMENTA MUR1 FERRAILLAGE)
PRED:(PILE2 PILE3)

NIL
```

```
"=================================================================

ETAIEMENTA

RES:((HOMMES 6))
DTARD:29
DTOT:29
DUREE:6
SUC:(FERRAILLAGE)
PRED:(ETAIEMENTB CULEE1 PILE2)

NIL

"=================================================================

REMBLAI4

RES:((HOMMES 7) (PELLE 1))
DTARD:53
DTOT:51
DUREE:10
SUC:(FIN)
PRED:(REMBLAI1 DURCISSEMENT_MUR4)

NIL


"=================================================================

REMBLAI1

RES:((HOMMES 7) (PELLE 1))
DTARD:43
DTOT:41
DUREE:10
SUC:(REMBLAI4 FIN)
PRED:(COULAGE_DALLE DURCISSEMENT_MUR1)

NIL


"=================================================================

DURCISSEMENT_MUR4

RES:NIL
DTARD:45
DTOT:22
DUREE:8
SUC:(REMBLAI4)
PRED:(MUR4)

NIL
```

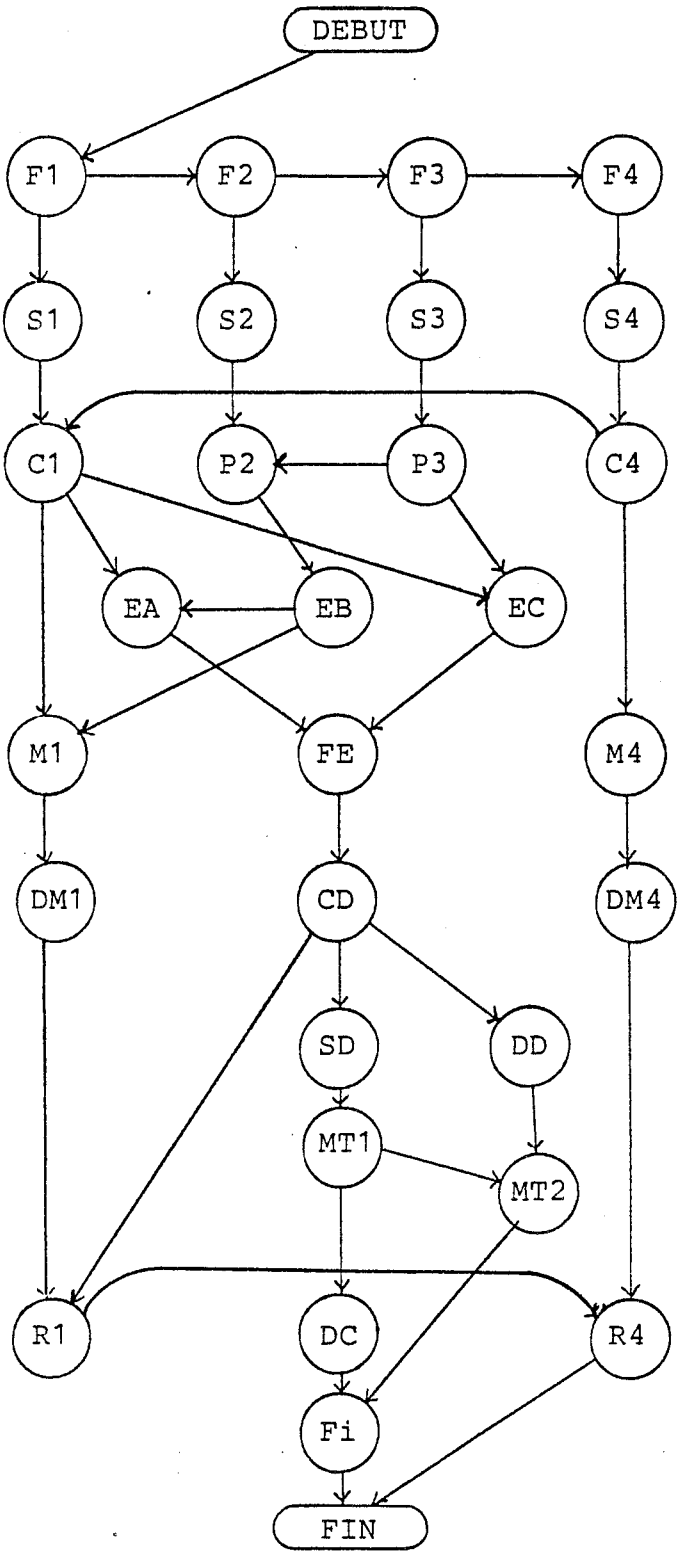
```
DURCISSEMENT_MUR1

RES:NIL
DTARD:35
DTOT:32
DUREE:8
SUC:(REMBLAI1)
PRED:(MUR1)

NIL

"=================================================================

MUR4

RES:((HOMMES 4) (COFFRAGE_MUR 1))
DTARD:42
DTOT:19
DUREE:3
SUC:(DURCISSEMENT_MUR4)
PRED:(CULEE4)

NIL

"=================================================================

MUR1

RES:((HOMMES 4) (COFFRAGE_MUR 1))
DTARD:32
DTOT:29
DUREE:3
SUC:(DURCISSEMENT_MUR1)
PRED:(ETAIEMENTB CULEE1)

NIL

"=================================================================

PILE3

RES:((HOMMES 5) (COFFRAGE_PILE 1))
DTARD:9
DTOT:9
DUREE:5
SUC:(PILE2 ETAIEMENTB ETAIEMENTC)
PRED:(SEMELLE3)

NIL
```

```
"=============================================================

PILE2

RES:((HOMMES 5) (COFFRAGE_PILE 1))
DTARD:14
DTOT:14
DUREE:5
SUC:(ETAIEMENTA ETAIEMENTB)
PRED:(PILE3 SEMELLE2)

NIL

"=============================================================="

CULEE4

RES:((HOMMES 7) (COFFRAGE_CULEE 1))
DTARD:13
DTOT:11
DUREE:8
SUC:(CULEE1 MUR4 ETAIEMENTC)
PRED:(SEMELLE4)

NIL


"=============================================================="

CULEE1

RES:((HOMMES 7) (COFFRAGE_CULEE 1))
DTARD:21
DTOT:19
DUREE:8
SUC:(ETAIEMENTC MUR1 ETAIEMENTA)
PRED:(CULEE4 SEMELLE1)

NIL


"=============================================================="

SEMELLE4

RES:((HOMMES 4))
DTARD:10
DTOT:8
DUREE:3
SUC:(CULEE4)
PRED:(FOUILLE4)

NIL
```

```
"================================================================

SEMELLE3

RES:((HOMMES 4))
DTARD:6
DTOT:6
DUREE:3
SUC:(PILE3)
PRED:(FOUILLE3)

NIL

"================================================================

SEMELLE2

RES:((HOMMES 4))
DTARD:11
DTOT:4
DUREE:3
SUC:(PILE2)
PRED:(FOUILLE2)

NIL

"================================================================

SEMELLE1

RES:((HOMMES 4))
DTARD:17
DTOT:2
DUREE:4
SUC:(CULEE1)
PRED:(FOUILLE1)

NIL

"================================================================

FOUILLE4

RES:((HOMMES 3) (PELLE 1))
DTARD:8
DTOT:6
DUREE:2
SUC:(SEMELLE4)
PRED:(FOUILLE3 FOUILLE2 FOUILLE1 DEBUT)

NIL
```

```
"================================================================="

FOUILLE3

RES:((HOMMES 3) (PELLE 1))
DTARD:4
DTOT:4
DUREE:2
SUC:(FOUILLE4 SEMELLE3)
PRED:(FOUILLE2 FOUILLE1 DEBUT)

NIL

"================================================================="

FOUILLE2

RES:((HOMMES 3) (PELLE 1))
DTARD:2
DTOT:2
DUREE:2
SUC:(FOUILLE3 FOUILLE4 SEMELLE2)
PRED:(FOUILLE1 DEBUT)

NIL


"================================================================="

FOUILLE1

RES:((HOMMES 3) (PELLE 1))
DTARD:0
DTOT:0
DUREE:2
SUC:(FOUILLE2 FOUILLE3 FOUILLE4 SEMELLE1)
PRED:(DEBUT)

NIL


"================================================================="

DEBUT

RES:NIL
DTARD:0
DTOT:0
DUREE:0
SUC:(FOUILLE1 FOUILLE2 FOUILLE3 FOUILLE4)
PRED:NIL

NIL
```

GRAPHE CORRESPONDANT



-* CHAPITRE V *-

CALINT

-* CHAPITRE V *-

CALINT

RESUME : En réalisant CALINT, notre but était moins d'obtenir un système de calcul d'intégrales et de simplification d'expressions mathématiques performant, que de montrer qu'EAQUE peut être appliqué sans difficulté à des domaines aussi différents que peuvent l'être ORDF et CALINT. CALINT est principalement destiné à la simplification d'expressions mathématiques, il possède pour cela plus de 70 règles de réécriture.

INTRODUCTION

Rappelons qu'instancier EAQUE à un domaine d'application revient à lui rajouter deux bases de connaissances. La première concerne le domaine d'application, c'est la base du domaine, la deuxième est la base de règles. Le but principal de CALINT est de mettre en évidence, après ORDF, l'aspect général d'EAQUE. Le calcul d'intégrales est cité davantage pour exemple, par contre la simplification d'expressions mathématiques est plus complète.

CALINT

-1- Le calcul d'intégrales

Une intégrale est définie par une fonction à intégrer, un signe et un coefficient précédent l'intégrale; c'est-à-dire par un triplet (fct , signe , coef) qui représente un objet EAQUE. CALINT détermine, en fait, la primitive de la fonction à intégrer. Pour cela, il utilise deux types de règles actions : les premières permettent de simplifier une intégrale, citons pour exemple : l'intégrale d'une somme est la somme des intégrales, le changement de variables... Les

secondes déterminent les primitives.
L'utilisation que fait CALINT de ces règles actions correspond à l'application de l'action de la première règle applicable considérée selon les CIR décroissants.
Notons que CALINT n'utilise, dans sa version actuelle, aucun critère de choix sur l'intégrale à résoudre (choix d'objet), il prend la première trouvée.

-2- La simplification d'expressions

Une expression à simplifier est un objet EAQUE. Il n'existe à chaque cycle qu'une seule expression à simplifier, le problème du choix d'objet ne se pose alors pas. Cette simplification se fait en deux phases. La première permet de développer l'expression, par exemple : suppression des parenthèses inutiles, développement de produits... La deuxième est une phase de simplification et de factorisation, citons pour exemple : la simplification par 0, la mise en facteur d'expressions identiques...
Ces deux phases correspondent à deux ensembles de règles actions différents. CALINT applique l'action de la première règle applicable considérée selon les CIR décroissants. Ainsi, la première action qu'essaye de faire CALINT, est de déporter toutes les sous expressions négatives en fin d'expression afin de réduire le nombre d'environnements différents pour une sous expression donnée.

A noter :
1- que CALINT effectue des opérations exactes sur les fractions.
2- l'existence de deux règles, l'une permettant de changer de phase dans la simplification (plus de règle applicable correspondant à la première phase), et l'autre indiquant la fin du problème (plus de règle applicable hormis elle même). Ces règles, devant être appliquées après avoir essayé toutes les autres, ont un CIR nul.

Remarque : CALINT est un exemple d'utilisation, dans l'écriture des règles actions, des variables locales simples et segments, ainsi que de l'unification qui leur est associée.

LA BASE DU DOMAINE

:A: LE DOMAINE.
:B: EAQUE.

:A: LE DOMAINE

Le travail de l'Ingénieur Cognitif se divise en cinq points :

-1- La définition des classes et des objets LRO pour la description du contexte

Il existe deux classes LRO, une classe INTEGRALE définie par les propriétés VAL, pour la fonction à intégrer; SIGNE, de valeur par défaut +; COEF, de valeur par défaut 1; et NATURE, de valeur par défaut intégrale. Cette dernière propriété permettra de distinguer les règles actions portant sur les intégrales, des règles actions portant sur les expressions à simplifier. La deuxième est la classe EXPR_A_SIMPLIFIER, définie par les propriétés VAL, pour l'expression à simplifier, et NATURE, de valeur par défaut expression.

-2- la définition des objets

Les objets sont directement issus de la description du problème. La classe des objets est définie par les propriétés VAL, SIGNE de valeur par défaut +, COEF de valeur par défaut 1, et NATURE.

-3- La définition des actions

Il existe trois actions différentes. L'action TRANSFORMATION, pour le calcul d'intégrales, qui permet de remplacer une intégrale par une ou plusieurs autres intégrales (création de nouveaux objets). L'action PRIMITIVE, toujours pour le calcul d'intégrales, permet de remplacer une fonction par sa primitive. Enfin, l'action SIMPLIFICATION, pour la simplification d'expressions mathématiques, qui remplace l'expression par son expression simplifiée.

-4- la définition des propriétés du domaine

Elles sont relativement nombreuses. Nous citerons un ensemble de fonctions booléennes permettant de déterminer, par exemple, si une expression : est linéaire, se réduit à un nombre ou une fraction, est une somme, est la dérivée d'une autre expression... Mais aussi, le développement de produits; la somme, le produit, la simplification de fractions; l'inversion d'une expression; le changement de variables; la dérivation d'une expression...

-5- la particularisation des squelettes d'analyse syntaxique

Les conditions et les actions sont, évidemment, des fonctions LISP, cependant, l'analyse syntaxique qui en est faite permet l'écriture de fonctions préfixées et infixées avec suppression des parenthèses au premier niveau.

Exemple : ENTIER K1?

```
          K3? := (+ K1? k2?)
```

:B: EAQUE

Cette partie correspond à la programmation de la structure de contrôle d'EAQUE.

Dans le cas d'une expression à simplifier, le problème du choix d'objet ne se pose pas, puisqu'il n'en existe qu'un seul. Les règles objets et les CIO n'interviennent pas. Par contre, il peut exister plusieurs objets pour le calcul d'intégrales, mais CALINT ne possède pas, faute de lui en a avoir donné, de règle objet (l'objet choisi est le premier trouvé), les CIO n'interviennent également pas.

Le choix d'action correspond à la première règle activable trouvée, considérée selon les CIR décroissants. Il n'existe donc pas de méta-règle, la fonction de notation des actions est l'affectation au CIA de l'action, du CIR de la règle dont elle est issue, et le CIA maximum est nul.

Quant au backtrack, il n'intervient que dans le cas où la primitive n'est pas trouvée, ou qu'il est demandé une autre solution pour la simplification; il correspond à un depth-first : la fonction de notation des choix est la longueur de la suite des choix jusqu'alors appliqués.

LA BASE DE REGLES

Nous ne donnons ici que la structure générale de la base de règles de CALINT, des exemples seront donnés dans l'annexe au paragraphe suivant.

Il n'existe qu'un seul type de règle : les règles actions.

La base de règles se divise en deux parties distinctes : la première pour l'intégration, la seconde pour la simplification. L'indépendance de ces deux ensembles de règles actions est assurée par l'existence de la condition INTEGRALE_VIDE qui est vérifée s'il n'existe pas, ou plus, de primitive à déterminer. En ce qui concerne la partie de la base destinée à la simplification, la séparation entre les règles correspondant à chacune des deux phases, est assurée par l'existence de deux prédicats : PHASE1 et PHASE2.

Les règles actions permettant de changer de phase et d'indiquer la fin du problème utilisent le prédicat AUCUNE_ACTION indiquant si des règles actions ont été, ou non, activées avant elles. Ainsi, si aucune règle action n'a été jusqu'alors activée et que CALINT est dans la première phase, alors, avec un CIR de 0 qui assure que cette règle ne sera appliquée qu'en dernier, CALINT passe à la phase2.

ANNEXES

CALINT

EXEMPLES DE REGLES ACTIONS

Remarque : Dire qu'une expression E contient les sous expressions E1 et E2 signifie que E est de la forme :
E = EX E1 EY E2 EZ
Où EX, EY et EZ peuvent représenter n'importe quelle expression, y compris l'expression vide.
Une expression est une suite de symboles, et n'est liée à aucune interprétation mathématique.
EX, EY et EZ sont des variables segments que nous noterons par exemple X*?, Y*? et Z*? (nom suivi du suffixe *?).
Ainsi dire qu'une expression E est une somme signifie que E contient au moins une expression réduite au signe +. E est donc de la forme X*? + Y*?.

A: SIMPLIFICATION D'INTEGRALES.

Remarque : une intégrale a une valeur, l'expression à intégrer, un signe et un coefficient précédant l'intégrale. Elle sera représentée par un triplet (val , signe , coeff).

-1- L'intégrale d'une somme est la somme des intégrales.
Si N est la nature de l'expression I, et si I est une expression à intégrer (INT ?N), et si I est une somme (?I UNIF (X*? + Y*?)), alors I se décompose en deux sous expressions à intégrer, chacune ayant un coefficient de 1 et un signe +. Ce sont les intégrales ((X*?) , + , 1) et ((Y*?) , + , 1).

```
RA   RA1 : CIR 10 :
     VAR :: ?I = VAL /
            ?N = NATURE
     SI  INT ?N  ET  ?I UNIF (X*? + Y*?)
     ALORS  TRANSFORMATION  ( (X*?) + 1 )  ( (Y*?) + 1 ) ;
```

-2- Si I est une expression à intégrer, et si I est une fonction linéaire de coefficient K de la forme K * X? (X? est une variable simple, c'est à dire, qu'à la différence des variables segments, elle ne peut correspondre qu'à un seul symbole ou à une expression parenthésée), alors I se transforme en une nouvelle expression à intégrer de valeur X?, de signe + et de coefficent K.

```
RA  RA4 : CIR 8 :
    VAR :: ?I = VAL /
           ?N = NATURE
    SI  INT ?N  ET  ?I UNIF (K? * X?)  ET  ENTIER K?
    ALORS  TRANSFORMATION  ( X? + K? )  ;
```

-3- Si I est une expression à intégrer, et si I est de la forme K * X*?, alors I se transforme en une nouvelle expression à intégrer de valeur X*?, de signe + et de coefficent K. ETOILE (X*?) permet d'éliminer les expressions de la forme K * Y*? + Z*? qui s'unifient bien avec K * X*?.

```
RA  RA160 : CIR 8 :
    VAR :: ?I = VAL /
           ?N = NATURE
    SI  INT ?N  ET  ?I UNIF (K? * X*?)  ET  ENTIER K?  ET  ETOILE (X*?)
    ALORS  TRANSFORMATION  ( (X*?) + K? )  ;
```

-4- Si I est une expression à intégrer, et si I est de la de la forme (AX + B) à la puissance N, alors nous effectuons un changement de variable T = AX + B, et I se transforme en une nouvelle expression à intégrer, de valeur T à la puissance N, de signe + et de coefficient 1/A.

```
RA  RA6 : CIR 6 :
    VAR :: ?I = VAL /
           ?N = NATURE
    SI  INT ?N  ET  ?I UNIF (X? ** N?)  ET  ENTIER N?
       ET  X? UNIF (A? * X1? + B?)  ET  VARIABLE_SIMPLE X1?
       ET  ENTIER A?  ET  ENTIER B?
       ET  T? := (CHANGEMENT_VARIABLE X?)  ET  K? := (/ 1 A?)
    ALORS  TRANSFORMATION  ( (T? ** N?)  +  K?)  ;
```

B: DETERMINATION DES PRIMITIVES

-1- Si I est une expression à intégrer (INT ?N), et si I et une fonction de X (VARIABLE_SIMPLE X?) à la puissance N, N entier (ENTIER N?), alors la primitive de I est X à la puissance T que divise T, avec T=N+1.

```
RA  RA21 : CIR 6 :
    VAR :: ?I = VAL /
           ?N = NATURE
    SI  INT ?N  ET  ?I UNIF ( X? ** N? )  ET  ENTIER N?
        ET  VARIABLE_SIMPLE X?  ET  T? := (+ N? 1)
    ALORS  PRIMITIVE ( X? ** T? / T? )  ;
```

-2- La primitive de F(X) à la puissance N que multiplie F'(X) est F(X) à la puissance N+1 que divise N+1.

```
RA  RA20 : CIR 6 :
    VAR :: ?I = VAL /
           ?N = NATURE
    SI  INT ?N  ET  ?I UNIF (X? ** N? * Y? )
        ET  ENTIER N?  ET  Y? VAR= (DERIVEE X?)  ET  T? := (+ N? 1)
    ALORS  PRIMITIVE ( X? ** T? / T? )  ;
```

-3- La primitive de exponentielle K * Y est exponentielle K * Y que divise K.

```
RA  RA202 : CIR 6 :
    VAR :: ?I = VAL /
           ?N = NATURE
    SI  INT ?N  ET  ?I UNIF ( E ** X? )  ET  X? UNIF (K? * Y?)
        ET ENTIER K?  ET  VARIABLE_SIMPLE Y?
    ALORS  PRIMITIVE ( E ** X? / K? ) ;
```

C: SIMPLIFICATION D'EXPRESSIONS MATHEMATIQUES

La simplification d'expressions mathématiques est de loin la partie la plus importante de CALINT, 70 règles sur 92.
Elle se divise en deux phases, la première de développement la seconde de simplification et de factorisation.
Rappelons que la stratégie du choix d'action est la première règle applicable de plus fort CIR. Les CIR des règles permettent de favoriser certaines stratégies pour la simplification. Ainsi CALINT, après chaque simplification, va déplacer, si elles existent, toutes les sous expressions négatives en fin d'expression.

PHASE1 DE DEVELOPPEMENT

Toutes les règles de cette phase ont une condition indiquant qu'elles ne s'appliquent que pour la phase1.

-1- Suppression des parenthéses inutiles.
Il existe plusieurs règles pour la suppression des parenthèses selon les divers environnements de l'expression parenthésée. Nous en citerons deux.
Si I est une expression à simplifier, et si I contient une expression parenthésée qui n'est pas une fraction, alors les parenthèses sont supprimées dans l'expression I.

```
RA  RA55 : CIR 9 :
    VAR :: ?I = VAL /
           ?N = NATURE
    SI  INTEGRALE_VIDE  ET  PHASE1  ET  RESUL ?N  ET  ?I UNIF ( X*? + (Y*?) )
        ET  NON_FRACTION (Y*?)
    ALORS SIMPLIFICATION ( X*? + Y*? ) ;

RA  RA52 : CIR 9 :
    VAR :: ?I = VAL /
           ?N = NATURE
    SI  INTEGRALE_VIDE  ET  PHASE1  ET  RESUL ?N
        ET  ?I UNIF ( X*? + (Y*?) - Z*? )  ET  NON_FRACTION (Y*?)
    ALORS SIMPLIFICATION ( X*? + Y*? - Z*? ) ;
```

-2- Suppression des parenthèses précédées du signe moins.

Si I est une expression à simplifier, et si I contient une expression parenthésée précédée du signe moins qui n'est pas une fraction, alors inverser l'expression parenthésée, puis supprimer les parenthèses dans l'expression I.

```
RA   RA56 : CIR 9 :
     VAR :: ?I = VAL /
            ?N = NATURE
     SI  INTEGRALE_VIDE  ET  PHASE1  ET  RESUL ?N  ET  ?I UNIF ( X*? - (Y*?) )
        ET  NON_FRACTION (Y*?)  ET  T? := (INVERSER (Y*?))
     ALORS SIMPLIFICATION ( X*? + T? ) ;

RA   RA54 : CIR 9 :
     VAR :: ?I = VAL /
            ?N = NATURE
     SI  INTEGRALE_VIDE  ET  RESUL  ?N  ET  PHASE1
        ET  ?I UNIF ( X*? - (Y*?) - Z*? )  ET  T? := (INVERSER (Y*?))
        ET  NON_FRACTION (Y*?)
     ALORS SIMPLIFICATION ( X*? + T? - Z*? ) ;
```

-3- Développement.

Si I est une expression à simplifier, et si I contient une expression parenthésée précédée d'un coefficient multiplicateur, alors remplacer dans I cette expression par son développement. Le cas de la fraction n'est pas ici à exclure.

```
RA   RA63 : CIR 8 :
     VAR :: ?I = VAL /
            ?N = NATURE
     SI  INTEGRALE_VIDE  ET  PHASE1
        ET  RESUL ?N  ET  ?I UNIF ( X*? + K? * (Y*?) )
        ET  ENTIER K?  ET  T? := (DEVELOPPER (K? * (Y*?)))
     ALORS SIMPLIFICATION ( X*? + T? ) ;

RA   RA64 : CIR 8 :
     VAR :: ?I = VAL /
            ?N = NATURE
     SI  INTEGRALE_VIDE  ET  PHASE1
        ET  RESUL ?N  ET  ?I UNIF ( X*? - K? * (Y*?) )
        ET  ENTIER K?  ET  T? := (DEVELOPPER (K? * (Y*?)))
     ALORS SIMPLIFICATION ( X*? - T? ) ;

RA   RA62 : CIR 8 :
     VAR :: ?I = VAL /
            ?N = NATURE
     SI  INTEGRALE_VIDE  ET  PHASE1  RESUL ?N
        ET  ?I UNIF ( X*? - K? * (Y*?) - Z*? )
        ET  ENTIER K? ET T? := (DEVELOPPER (K? * (Y*?)))
     ALORS SIMPLIFICATION ( X*? - T? - Z*? ) ;
```

PHASE2 DE SIMPLIFICATION ET DE FACTORISATION

Toutes les règles de cette phase ont une condition indiquant qu'elles ne s'appliquent qu'à la phase2.

-1- Règles des signes.

++ donne +, -- donne +, +- ou -+ donne -.

Si I est une expression à simplifier, et si I contient l'expression ++, alors remplacer dans I l'expression ++ par l'expression +.

```
% factorisation %

RA  RA166 : CIR 8 :
    VAR :: ?I = VAL /
           ?N = NATURE
    SI  INTEGRALE_VIDE  ET  RESUL ?N ET PHASE2
       ET  ?I UNIF ( X*? + + Y*?)
    ALORS SIMPLIFICATION  (  X*? + Y*?) ;

RA  RA67 : CIR 8 :
    VAR :: ?I = VAL /
           ?N = NATURE
    SI  INTEGRALE_VIDE  ET  RESUL ?N ET PHASE2
       ET  ?I UNIF ( X*? - - Y*?)
    ALORS SIMPLIFICATION  (  X*? + Y*?) ;

RA  RA68 : CIR 8 :
    VAR :: ?I = VAL /
           ?N = NATURE
    SI  INTEGRALE_VIDE  ET  RESUL ?N ET PHASE2
       ET  ?I UNIF ( X*? - + Y*?)
    ALORS SIMPLIFICATION  (  X*? - Y*?) ;
```

-2- Simplification par 0.

Si I est une expression à simplifier, et si I contient l'expression 0 * X?, alors supprimer cette expression dans I.

(La différence entre 0 * X? et 0 * X*? est importante. En effet si 0 * (a * X + b) est une expression de la forme 0 * X? ou 0 * X*?, par contre 0 * a * X + b est une expression de la forme 0 * X*?, mais non 0 * X?).

```
RA  RA70 : CIR 8 :
    VAR :: ?I = VAL /
           ?N = NATURE
    SI  INTEGRALE_VIDE  ET  RESUL ?N  ET  PHASE2
       ET  ?I UNIF ( 0 * X?)
    ALORS SIMPLIFICATION  (  0  ) ;

RA  RA74 : CIR 8 :
    VAR :: ?I = VAL /
           ?N = NATURE
    SI  INTEGRALE_VIDE  ET  RESUL ?N  ET  PHASE2
       ET  ?I UNIF (X*? + 0 * Y? + Z*?)
    ALORS SIMPLIFICATION  (X*? + Z*?) ;
```

-3- Mise en facteur.

Si I est une expression à simplifier, et si I contient deux expressions identiques K1 * Y? et K2 * Y?, alors faire la somme des coefficients K1 et K2: K3, et remplacer dans I ces deux expressions par K3 * Y?.

```
RA  RA77 : CIR 6 :
    VAR :: ?I = VAL /
           ?N = NATURE
    SI  INTEGRALE_VIDE  ET  RESUL ?N  ET  PHASE2
        ET  ?I UNIF ( K1? * Y? + K2? * Y? )
        ET  ENTIER K1?  ET  ENTIER K2?  ET  K3? := (+ K1? K2?)
    ALORS SIMPLIFICATION  ( K3? * Y? ) ;

RA  RA86 : CIR 6 :
    VAR :: ?I = VAL /
           ?N = NATURE
    SI  INTEGRALE_VIDE  ET  RESUL ?N  ET  PHASE2
        ET  ?I UNIF ( X*? + K1? * Y? + Z*? - K2? * Y? - T*? )
        ET  ENTIER K1?  ET  ENTIER K2?  ET  K3? := (- K1? K2?)
    ALORS SIMPLIFICATION  ( K3? * Y? + X*? + Z*? - T*? ) ;
```

-4- Somme, différence, produit, rapport d'entiers et de fractions.

Si I est une expression à simplifier, et si I contient deux entiers ou deux fractions K1 et K2, alors les remplacer dans I par leur somme, différence,... K3.

```
RA  RA477 : CIR 2 :
    VAR :: ?I = VAL /
           ?N = NATURE
    SI  INTEGRALE_VIDE  ET  RESUL ?N  ET  PHASE2
        ET  ?I UNIF ( K1? + K2?  )
        ET  ENTIER K1?  ET  ENTIER K2?  ET  K3? := (+ K1? K2?)
    ALORS SIMPLIFICATION  ( K3? ) ;

RA  RA486 : CIR 2 :
    VAR :: ?I = VAL /
           ?N = NATURE
    SI  INTEGRALE_VIDE  ET  RESUL ?N  ET  PHASE2
        ET  ?I UNIF ( X*? + K1? + Z*? - K2? - T*? )
        ET  ENTIER K1?  ET  ENTIER K2?  ET  K3? := (- K1? K2?)
    ALORS SIMPLIFICATION  ( K3? + X*? + Z*? - T*? ) ;

RA  RA305 : CIR 2 :
    VAR :: ?I = VAL /
           ?N = NATURE
    SI  INTEGRALE_VIDE  ET  RESUL ?N  ET  PHASE2
        ET  ?I UNIF (Z*? + K1? X*? / K2? - Y*?)  ET  ETOILE (X*?)
        ET  ENTIER K1?  ET  ENTIER K2?  ET  K3? := (/ K1? K2?)
    ALORS SIMPLIFICATION  (Z*? + K3? X*? - Y*?) ;
```

LE CHANGEMENT DE PHASE ET LA FIN DU PROBLEME.

Nous avons deux règles qui permettent de changer de phases et d'indiquer la fin du problème. Elles ont chacune un CIR de 0 puisque le changement de phase et la fin du problème correspondent au fait qu'il n'existe plus de règle applicable hormis elles mêmes, elles doivent donc être appliquées qu'après avoir essayé toutes les autres règles.

```
%... changement de phase et de fin ...%

RA   RA100 : CIR 0 :
     VAR :: ?I = VAL /
            ?N = NATURE
     SI  INTEGRALE_VIDE ET RESUL ?N  ET  AUCUNE_ACTION   ET  PHASE1
     ALORS  ->PHASE2  ;


RA   RA101 : CIR 0 :
     VAR :: ?I = VAL /
            ?N = NATURE
     SI  INTEGRALE_VIDE ET RESUL ?N  ET  AUCUNE_ACTION  ET  PHASE2
     ALORS  FINI  ;



FIN!
```

CALINT

EXEMPLES DE RESOLUTION

EXEMPLE 1

```
%-----------------------------------------------------------%
%                        EXEMPLE1                           %
%-----------------------------------------------------------%


%.... expression a simplifier ...%

 E1  EXPR_A_SIMPLIFIER
 VAL = ((4 * (X + 2 * Y - 5 * Z) - 2 * (Z + 4 * Y) + 5 * X))  ;


%.... condition d'arret ........%

 ARRETS = ->FINI ;


FIN!
```

```
L_EXPR_SIMPLIFIEE_DE_:((4 * (X + 2 * Y - 5 * Z) - 2 * (Z + 4 * Y) + 5 * X))

EST_:((9 * X - 22 * Z))
```

EXEMPLE 2

```
%-------------------------------------------------------%
%                    EXEMPLE2                           %
%-------------------------------------------------------%


%.... expression a integrer ...%

 I1  INTEGRALE
 VAL = ( (4 * X ** 2 - 15 * (SIN X) ** 3 * (COS X)) ) ;


%.... condition d'arret ........%

 ARRETS = ->FINI ;


FIN!
```

```
 LA_PRIMITIVE_DE_:((4 * X ** 2 - 15 * (SIN X) ** 3 * (COS X)))

 EST___________:((+ (4 / 3) * X ** 3 - (15 / 4) * (SIN X) ** 4))_+_CONSTANTE
```

EXEMPLE 3

```
%-----------------------------------------------------%
%                    EXEMPLE3                         %
%-----------------------------------------------------%


%.... expression a integrer ...%

 I1  INTEGRALE
 VAL = ( (E ** (3 * X) + (8 * X + 10) ** 5) )  ;


%.... condition d'arret .......%

 ARRETS = ->FINI ;


FIN!
```

```
LA_PRIMITIVE_DE_:((E ** (3 * X) + (8 * X + 10) ** 5))

EST___________:((+ (1 / 48) * (8 * X + 10) ** 6 + E ** (3 * X) / 3))
_+_CONSTANTE
```

EXEMPLE 4

```
%-------------------------------------------------------%
%                      EXEMPLE4                         %
%-------------------------------------------------------%


%.... expression a simplifier ...%

 E1  EXPR_A_SIMPLIFIER
 VAL = (  ( (2 / 3) * ( 4 * X - (5 / 2) * Z + (17 / 9) )
         - (10 / 3) * ( X + 4 * Z)
         + (2 / 3) * X + (8 / 5)  )  )  ;


%.... condition d'arret ........%

 ARRETS = ->FINI ;


FIN!
```

```
L_EXPR_SIMPLIFIEE_DE_:(((2 / 3) * (4 * X - (5 / 2) * Z + (17 / 9)) - (10 / 3)
* (X + 4 * Z) + (2 / 3) * X + (8 / 5)))

EST_:(((386 / 135) - 15 * Z))
```

QUELQUES CYCLES DE RESOLUTION...

.
.
.

```
"=====================================================
CHOIX_D_OBJET

expression_a_simplifier_:_(((-2 / 3) * X + (34 / 27) + (2 / 3) * X + (8 / 5) -
(40 / 3) * Z - (5 / 3) * Z))
coef_:_1
signe_:_+

"objet_choisi


OBJET_CHOISI_PAR_DEFAUT


"=====================================================
CHOIX_D_ACTION

(SIMPLIFICATION (0 * X + (34 / 27) + (8 / 5) - (40 / 3) * Z - (5 / 3) * Z))

"action_choisie


"Regle_et_son_occurence_ayant_le_plus_contribue_a_ce_choix:"
RA88 1



"=====================================================
CHOIX_D_OBJET

expression_a_simplifier_:_((0 * X + (34 / 27) + (8 / 5) - (40 / 3) * Z - (5 /
3) * Z))
coef_:_1
signe_:_+

"objet_choisi


OBJET_CHOISI_PAR_DEFAUT
```

```
"=====================================================
CHOIX_D_ACTION

(SIMPLIFICATION ((34 / 27) + (8 / 5) - (40 / 3) * Z - (5 / 3) * Z))

"action_choisie


"Regle_et_son_occurence_ayant_le_plus_contribue_a_ce_choix:"
RA120 1



"=====================================================
CHOIX_D_OBJET

expression_a_simplifier_:_(((34 / 27) + (8 / 5) - (40 / 3) * Z - (5 / 3) * Z))
coef_:_1
signe_:_+

"objet_choisi


OBJET_CHOISI_PAR_DEFAUT


"=====================================================
CHOIX_D_ACTION

(SIMPLIFICATION ((34 / 27) + (8 / 5) - 15 * Z))

"action_choisie


"Regle_et_son_occurence_ayant_le_plus_contribue_a_ce_choix:"
RA130 1
```

.
.
.

-* CHAPITRE VI *-

EALOG

-* CHAPITRE VI *-

EALOG

RESUME : La structure de contrôle de PROLOG est incluse dans EAQUE. Pour cela il suffit de programmer EAQUE pour que l'action choisie soit celle de la première règle action activable, et que la fonction de notation des choix corresponde à un backtrack depth-first. Les différents types de règles et les CIR n'interviennent pas.

-1-

INTRODUCTION

PROLOG est actuellement un langage très à la mode, c'est pourquoi il nous a semblé utile de montrer que sa structure de contrôle est incluse dans EAQUE : c'est le système EALOG. Notre but a été uniquement de montrer cette inclusion, ce qui explique la représentation des objets choisie.
Rappelons qu'instancier EAQUE revient à lui rajouter deux bases de connaissances. L'une concerne le domaine d'application, c'est la base du domaine, l'autre est la base de règles.

-2-

PROLOG

PROLOG (PROgrammation LOGique) est un langage de programmation basé sur un sous ensemble de la logique des prédicats du premier ordre, que constituent les clauses de HORN, et sur une règle d'inférence complète, le principe de résolution. Initialement développé dans les années 70 par le Groupe Intelligence Artificielle de A.COLMERAUER pour la compréhension des langues naturelles, il peut être utilisé dans de nombreux domaines comme les bases de données ou le calcul formel.

Rappelons qu'une clause de HORN est une formule bien formée de la forme : A1 et A2 et ... et An ==➤ B, ce qui est logiquement équivalent à la fbf (non A1) ou (non A2) ou ... (non An) ou B.
Quant au principe de résolution, il permet, sur les deux clauses de l'exemple suivant, pour la première : (non A1) ou (non A2) ou B, pour la deuxième : (non B), d'inférer la clause résolvante (non A1) ou (non A2). Ce qui correspond, si B est le théorème à démontrer, à l'application du principe de résolution par réfutation. La démonstration consiste alors, à partir de l'ensemble des clauses, à dériver la clause vide.
En continuant notre exemple, si A1 est une assertion, nous dérivons de (non A1) ou (non A2) et de A1, la clause (non A2). S'il n'existe pas l'assertion A2 (le théorème aurait alors été démontré), mais la clause (non C1) ou (non C2) ou A2, nous dérivons (non C1) ou (non C2), et ainsi de suite...

Nous n'avons pas parlé de variables dans les prédicats, dans ce cas, il suffit de trouver une substitution permettant d'appliquer la résolution.

PROLOG utilise une stratégie depth-first exhaustive (nous ne parlerons pas du cut point, il y a donc dérivation de toutes les solutions possibles) en considérant les clauses dans leur ordre d'écriture, appliquant la première qui peut l'être sur le dernier sous but généré.

Notation : Il existe différentes notations pour l'écriture des règles PROLOG, nous avons adopté celle qui correspond le mieux, à notre avis, à l'idée la plus simple du fonctionnement de l'interpréteur PROLOG. En effet, celui-ci peut être vu comme un système de réécriture en marche avant. La règle "A1 et A2 et ... et An ==➤B" peut se voir comme la réécriture du but B par les sous buts A1, A2, ... An (B est vrai si A1, A2, ... An le sont. Les Ai sont à considérer

dans l'ordre donné).
Ce que nous noterons désormais par :
B --➤A1 A2 ... An ;

Les clauses représentant des assertions, ou hypothèses, n'engendrent pas de sous but (dérive la clause vide), elles s'écriront donc, avec la notation choisie :
C --➤;

Quant au théorème à démontrer, il s'écrit :
--➤B1 B2 ...Bm ;

LA BASE DU DOMAINE

:A: LE DOMAINE
:B: EAQUE

:A: LE DOMAINE.

Le travail de l'Ingénieur Cognitif, en ce qui concerne le domaine d'application, se divise en cinq points.

-1- la définition des classes et des objets LRO pour la description du contexte

Il n'en existe pas.

-2- la définition des objets

L'objet est constitué de la pile des buts et du théorème. Ceci évite à EALOG de gérer les différentes variables que la pile des buts et le théorème peuvent contenir, elles seront gérées automatiquement par EAQUE à travers l'activation des règles actions.

-3- la définition des actions

Il existe une seule action : la résolution. Elle permet, soit de remplacer le dernier but généré par le ou les sous buts donnés par la règle EALOG sélectionnée, soit simplement d'avancer dans la résolution dans le cas d'une assertion.

-4- la définition des propriétés du domaine

Il n'en existe pas, les relations sont traduites sous forme de règles EALOG.

-5- l'analyseur syntaxique

La syntaxe d'écriture des règles actions EALOG est celle des règles PROLOG. EALOG contient donc un analyseur syntaxique de ces règles permettant de les réécrire sous la syntaxe EAQUE. Ainsi la règle EALOG suivante : "A -- B C ;", devient la règle action EAQUE suivante, traduisant son utilisation correspondant à un interpréteur PROLOG : "SI A s'unifie avec le dernier sous but généré (contenu dans la description de l'objet), ALORS remplacer A par B et C ;".

:B: EAQUE

Cette partie correspond à la programmation de la structure de contrôle d'EAQUE afin qu'elle corresponde à la stratégie d'un interpréteur PROLOG.
Il n'existe, à chaque cycle de contrôle, qu'un objet définissant la pile des buts courants et le théorème (pouvant être partiellement instancié). Il n'existe pas de règle objet et tout de qui se rapporte au CIO n'intervient pas. Dans le cas plus optimum de séparation des sous buts, il n'existerait qu'un seul critère de choix, qui est de prendre le sous but le plus récent. La stratégie, pour le choix d'action, est de considérer les règles actions dans leur ordre d'écriture et de prendre la première activable (unification de la partie gauche avec le dernier sous but généré). Les méta-règles, tout comme les CIR et les CIA, n'ont pas de raison d'être. Il n'existe donc pas de méta-règle, et les CIR des règles actions, ainsi que le CIA maximum, sont nuls.
La recherche se faisant en depht-first, la fonction de

notation des choix est la longueur de la suite des choix jusqu'alors appliqués. Cette recherche étant exhaustive, le backtrack est forcé dès que la pile des buts est vide.

LA BASE DE REGLES

Nous avons appliqué EALOG à l'exploitation d'une base de données généalogiques, dont nous verrons quelques exemples à l'annexe. Nous ne parlerons ici que de la structure générale que doit respecter la base de règles pour EALOG, indépendamment d'un domaine d'application.

Il n'existe qu'un type de règles : les règles actions.

La base de règles se divise en quatres groupes distincts et ordonnés. Le premier permet d'exprimer le théorème dans le cas où il ne se réduit pas à un seul prédicat. Le deuxième est la base des faits (des hypothèses, des assertions). Le troisième rassemble les règles de déduction, elles permettent de "réécrire" un sous but par un ou plusieurs autres sous buts, généralement plus simples quant à leur résolution. Enfin, le dernier déterminé non pas par l'Expert, mais par l'Ingénieur Cognitif, correspond à diverses fonctions, comme les fonctions booléennes ou d'impression, auxquelles sont associées des règles EAQUE utilisées de la même manière que les règles PROLOG.

Ces quatre ensembles seront détaillés, sur un exemple, dans l'annexe qui suit.

ANNEXES

EALOG

EXEMPLES DE REGLES ACTIONS

Rappel : Nous avons adopté la syntaxe Prolog pour l'écriture des règles et non celle par défaut d'EAQUE. Mais l'utilisation qui en est faite est la même, puisque l'analyseur syntaxique d'EALOG génère les règles d'EAQUE adéquates.
La règle Prolog "A -➤B C ;" est interprétée de la façon suivante : si A s'unifie avec le dernier sous but (prédicat) généré, alors, après avoir instancié, s'il y a lieu, les différentes variables, remplacer A par les deux nouveaux sous-buts B et C, B étant considéré comme le dernier sous-but généré (A est vrai si B et C le sont).
La règle EAQUE associée est : si le premier but de la pile des buts (dernier sous-but généré) s'unifie avec A, alors il est rajouté, après avoir instancié, s'il y a lieu, les différentes variables, et supprimé le sous-but A, les sous-buts B et C en tête de la pile des buts. Elle s'écrira:
SI (PREM PILE_BUTS) UNIFEVAL A
ALORS RESOLUTION PILE_BUTS THEOREME (B C) ;

Remarque : Les CIR associés aux règles Prolog sont tous nuls et les seuils sont mis à 0. Ainsi EALOG parcourt les règles dans l'ordre dans lequel elles ont été rentrées et prend la première activable.
Ces règles se divisent en quatre groupes distincts et ordonnés.

A: L'EXPRESSION DU THEOREME A DEMONTRER.

Si le théorème à démontrer se réduit à un seul prédicat, alors il est introduit lors de la description du problème. Sinon la première règle de la base de règle Prolog est la règle exprimant les prédicats du théorème.

Ainsi si l'on recherche tous les hommes dont le père habite à Grenoble, la description du problème se résumera à (RECHERCHER X?), et nous ajouterons en tête des règles la règle suivante:
(RECHERCHER X?) -►(HOMME X?) (PERE Y? X?) (HABITE Y? GRENOBLE) ;

B: LA BASE DE FAITS.

C'est un ensemble de règles traduisant des hypothèses. Si le sous-but courant à résoudre s'unifie avec la partie gauche d'une de ces régles, alors il est considéré comme vérifié et n'engendre pas d'autre sous-but (dérive la clause vide).

Ainsi dire que:
- Paul est un homme s'écrira : (HOMME PAUL) -►;
- Marie est une femme s'écrira : (FEMME MARIE) -►;
- Pierre et Sophie forment un couple s'écrira : (COUPLE PIERRE SOPHIE) -►;
- Jean est l'enfant de Marie s'écrira : (ENFANT JEAN MARIE) -►;
- Jean habite Grenoble s'écrira : (HABITE JEAN GRENOBLE) -► ;

C: LES REGLES DE DEDUCTION.

C'est un ensemble de règles permettant de "remplacer" ou de "réécrire" un sous-but par un ou plusieurs autres sous-buts généralement plus simples quant à leur résolution, et dont les valeurs vraies entraînent la valeur vraie pour le sous-but (prédicat) initial.

Ainsi quelque soit x et y, "x est le fils de y" est vrai si "x est un homme" est vrai et si "x est l'enfant de y" est vrai.
Ce qui s'écrira:
(FILS X? Y?) -►(HOMME X?) (ENFANT X? Y?) ;
L'existence d'une deuxième règle permet une autre possibilité dans le cas où la règle précédente n'a pas abouti (il n'existe pas de y dont x est l'enfant).
(FILS X? Y?) -►(HOMME X?) (ENFANT X? Z?) (COUPLE Y? Z?) ;

Autre règle:
Pour tout x et y, x est le grand-père de y s'il existe un z tel que x est le père de z et z le père de y.
Ou alors
Pour tout x et y, x est le grand-père de y s'il existe un z tel que x est le père de z et z la mère de y.
Ce qui s'écrira:
(GRAND_PERE X? Y?) -►(PERE X? Z?) (PERE Z? Y?) ;
(GRAND_PERE X? Y?) -►(PERE X? Z?) (MERE Z? Y?) ;

Autre règle:
Si l'on cherche à savoir si x habite à y et que ce prédicat n'est pas une hypothèse, nous pouvons chercher s'il existe un z qui forme un couple avec x et regarder si z habite à y (dans l'hypothèse où un couple habite au même endroit).
Ce qui s'écrira:
(HABITE X? Y?) -►(COUPLE X? Z?) (HABITE Z? Y?) ;

Remarque : L'ordre d'écriture des règles, et des sous-buts apparaissant à droite du signe -►a une importance primordiale quant à la résolution du problème. Cette remarque est encore plus vraie avec le "cut point".

D: LES FONCTIONS

Certains prédicats sont des fonctions "systèmes" comme inférieur et imprimer. Il leur est associé des régles EAQUE utilisées de la même manière que les règles Prolog, unification avec le dernier sous-but généré, condition supplémentaire correspondant à la fonction considérée et dérivation de la clause vide.

```
%...  FONCTIONS  ...%

RA RA1   : CIR 1 :
         VAR :: ?PB = PILE_BUTS /
                ?T = THEOREME
         SI  (PREM ?PB) UNIFEVAL (INF X? Y?)  ET  X? < Y?  ET  RIEN ?T
         ALORS  RESOLUTION  ?PB ?T NIL ;

RA RA2   : CIR 1 :
         VAR :: ?PB = PILE_BUTS /
                ?T = THEOREME
         SI  (PREM ?PB) UNIFEVAL (IMPRIMER X*?)  ET PRINT X*?  ET  RIEN ?T
         ALORS  RESOLUTION  ?PB ?T NIL ;
```

EALOG

EXEMPLES DE RESOLUTION

les faits et les règles de déduction sont donnés en fin de cette annexe.

EXEMPLE 1

```
%-----------------------------------------------------------%
%                        EXEMPLE1                           %
%-----------------------------------------------------------%


%...   theoreme a demontrer    ...%

T1   THEOREME      VAL = (FILS X? BRIGITTE) ;


%...   condition d arret   ...%

ARRETS = ->FINI ;


FIN!
```

```
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

THEOREME_PROUVE_:_(((FILS DIDIER BRIGITTE)))

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

EXEMPLE 2

Rechercher tous les hommes dont le père habite à Grenoble:
Nous rajoutons comme première règle de la base des règles Prolog:
(RECHERCHER X?) -►(HOMME X?) (PERE Y? X?) (HABITE Y? GRENOBLE) ;

```
%-----------------------------------------------------%
%                    EXEMPLE3                         %
%-----------------------------------------------------%


%...  theoreme a demontrer   ...%

T1  THEOREME     VAL = (RECHERCHER X?) ;


%...  condition d arret  ...%

ARRETS = ->FINI ;


FIN!

 ::::::::::::::::::::::::::::::::::::::::::::::::::::

 THEOREME_PROUVE_:_(((RECHERCHER JEAN)))

 ::::::::::::::::::::::::::::::::::::::::::::::::::::
 ::::::::::::::::::::::::::::::::::::::::::::::::::::

 THEOREME_PROUVE_:_(((RECHERCHER JEAN)))

 ::::::::::::::::::::::::::::::::::::::::::::::::::::
 ::::::::::::::::::::::::::::::::::::::::::::::::::::

 THEOREME_PROUVE_:_(((RECHERCHER MICHEL)))

 ::::::::::::::::::::::::::::::::::::::::::::::::::::
```

QUELQUES CYCLES DE RESOLUTION...

Note : la fonction trace n'ayant pas une grande importance pour EALOG, nous avons gardé la trace par défaut d'EAQUE.

```
"================================================"
&OBJ_ACT&
REGLE_PLUS:NIL
THEOREME:(((FILS X? BRIGITTE)))
PILE_BUTS:(((FILS X? BRIGITTE)))
&CIO&:0
"objet_choisi
"================================================"


OBJET_CHOISI_PAR_DEFAUT


"================================================"
&TRANS_ACT&
REGLE_PLUS:((G0042 1))
&DOM_VAR&:NIL
&VAL_TRANS&:(RESOLUTION ((FILS X? BRIGITTE)) ((FILS X? BRIGITTE)) ((HOMME X?)
(ENFANT X? BRIGITTE)))
&CIT&:1
"action_choisie
"================================================"


"Regle_et_son_occurence_ayant_le_plus_contribue_a_ce_choix:"
G0042 1



"================================================"
&OBJ_ACT&
REGLE_PLUS:NIL
THEOREME:(((FILS X? BRIGITTE)))
PILE_BUTS:(((HOMME X?) (ENFANT X? BRIGITTE)))
&CIO&:0
"objet_choisi
"================================================"


OBJET_CHOISI_PAR_DEFAUT
```

```
"================================================"
&TRANS_ACT&
REGLE_PLUS:((G0004 1))
&DOM_VAR&:NIL
&VAL_TRANS&:(RESOLUTION ((HOMME PAUL) (ENFANT PAUL BRIGITTE)) ((FILS PAUL
BRIGITTE)))
&CIT&:1
"action_choisie
"================================================"


"Regle_et_son_occurence_ayant_le_plus_contribue_a_ce_choix:"
G0004 1


 "================================================"
 &OBJ_ACT&
 REGLE_PLUS:NIL
 THEOREME:(((FILS PAUL BRIGITTE)))
 PILE_BUTS:(((ENFANT PAUL BRIGITTE)))
 &CIO&:0
 "objet_choisi
 "================================================"


 OBJET_CHOISI_PAR_DEFAUT


 "================================================"
 &OBJ_ACT&
 REGLE_PLUS:NIL
 THEOREME:(((FILS PAUL BRIGITTE)))
 PILE_BUTS:(((ENFANT PAUL BRIGITTE)))
 &CIO&:0
 "aucune_action_ne_lui_est_applicable"
 "================================================"
```

```
"================================================"
&PT_REPRISE_TRANS&
PROG_RESTAURATION:NIL
OBJETS_DETRUITS:(TRANSG0091)
OBJETS_CREES:(TRANSG0095)
TRANS_REJETEES:NIL
NOTE:3
EST_UN:TRANSG0094
"choix_de_transition_pris_comme_point_de_backtrack"
"================================================"


"================================================"
TRANSG0094
REGLE_PLUS:((G0004 1))
&DOM_VAR&:NIL
&VAL_TRANS&:(RESOLUTION ((HOMME PAUL) (ENFANT PAUL BRIGITTE)) ((FILS PAUL
BRIGITTE)))
&CIT&:1
NIL
"================================================"


"================================================"
&TRANS_ACT&
REGLE_PLUS:((G0005 1))
&DOM_VAR&:NIL
&VAL_TRANS&:(RESOLUTION ((HOMME PIERRE) (ENFANT PIERRE BRIGITTE)) ((FILS
PIERRE BRIGITTE)))
&CIT&:1
"action_choisie"
"================================================"


"Regle_et_son_occurence_ayant_le_plus_contribue_a_ce_choix:"
G0005 1
```

LA BASE DE FAITS

```
%...     BASE DE FAITS     ...%

(HOMME PAUL) -> ;
(HOMME PIERRE) -> ;
(HOMME JEAN) -> ;
(HOMME FRANCOIS) -> ;
(HOMME CLAUDE) -> ;
(HOMME MICHEL) -> ;
(HOMME ALAIN) -> ;
(HOMME DIDIER) -> ;
(FEMME MARIE) -> ;
(FEMME SOPHIE) -> ;
(FEMME CLAIRE) -> ;
(FEMME ANNE) -> ;
(FEMME SYLVIE) -> ;
(FEMME FRANCOISE) -> ;
(FEMME ANNIE) -> ;
(FEMME BRIGITTE) -> ;
(COUPLE PIERRE SOPHIE) -> ;
(COUPLE PAUL MARIE) -> ;
(COUPLE FRANCOIS ANNE) -> ;
(COUPLE ALAIN BRIGITTE) -> ;
(COUPLE JEAN ANNIE) -> ;
(ENFANT JEAN MARIE) -> ;
(ENFANT FRANCOIS SOPHIE) -> ;
(ENFANT CLAIRE MARIE) -> ;
(ENFANT SYLVIE ANNE) -> ;
(ENFANT ANNE MARIE) -> ;
(ENFANT CLAUDE ANNE) -> ;
(ENFANT ANNIE BRIGITTE) -> ;
(ENFANT MICHEL ANNIE) -> ;
(ENFANT FRANCOISE ANNIE) -> ;
(ENFANT DIDIER BRIGITTE) -> ;

(HABITE ANNE TOULOUSE) -> ;
(HABITE MARIE GRENOBLE) -> ;
(HABITE MICHEL PARIS) -> ;
(HABITE JEAN GRENOBLE) -> ;
(HABITE SOPHIE MONTPELLIER) -> ;
```

LES REGLES DE DEDUCTION

```
%...    REGLES DE DEDUCTION    ...%

(FILS X? Y?) -> (HOMME X?) (ENFANT X? Y?) ;
(FILS X? Y?) -> (HOMME X?) (ENFANT X? Z?) (COUPLE Y? Z?) ;
(FILLE X? Y?) -> (FEMME X?) (ENFANT X? Y?) ;
(FILLE X? Y?) -> (FEMME X?) (ENFANT X? Z?) (COUPLE Y? Z?) ;
(PERE X? Y?) -> (HOMME X?) (ENFANT Y? X?) ;
(PERE X? Y?) -> (HOMME X?) (ENFANT Y? Z?) (COUPLE X? Z?) ;
(MERE X? Y?) -> (FEMME X?) (ENFANT Y? X?) ;
(MERE X? Y?) -> (FEMME X?) (ENFANT Y? Z?) (COUPLE X? Z?) ;
(GRAND_PERE X? Y?) -> (PERE X? Z?) (PERE Z? Y?) ;
(GRAND_PERE X? Y?) -> (PERE X? Z?) (MERE Z? Y?) ;
(GRAND_MERE X? Y?) -> (MERE X? Z?) (MERE Z? Y?) ;
(GRAND_MERE X? Y?) -> (MERE X? Z?) (PERE Z? Y?) ;

(HABITE X? Y?) -> (COUPLE X? Z?) (HABITE Z? Y?) ;
```

-* CHAPITRE VII *-

CONCLUSION

-* CHAPITRE VII *-

CONCLUSION

Le langage LRO et la structure de contrôle EAQUE constituent un outil de développement de Systèmes Experts. Après avoir choisi un domaine d'application particulier, il est possible de définir rapidement les connaissances et les stratégies de résolution les plus adéquates pour le Système Expert correspondant. L'Expert, à l'aide de l'Ingénieur Cognitif, peut expérimenter, en grandeur "réelle", différentes stratégies, différentes structures d'objet... grâce aux systèmes experts qui sont alors générés. Il est toujours possible, pour des raisons d'efficacité, et lorsque tous les paramètres ont été fixés, de réécrire le dernier système généré en un système expert dédié, alors qu'une réalisation directe aurait demandé infiniment plus de temps.

L'idée d'un moteur général valable pour tout Système Expert (moteur universel) parait utopique à de nombreux chercheurs (LAURE 84). Il nous semble effectivement irréaliste d'envisager, à l'heure actuelle du moins, un langage de représentation des connaissances qui soit totalement universel (tout en restant abordable et efficace). De même si l'on ne restreint pas la famille de domaines d'application, il semble impossible , même avec une structure de contrôle très paramétrée de prévoir n'importe quel type de stratégie.
C'est pourquoi nous avons choisi une voie intermédiaire qui refuse la tentation trop ambitieuse du système universel, mais offre néanmoins un caractère intéressant de généralité; comme nous le montrent les instantiations d'EAQUE sur des domaines aussi différents que peuvent l'être les problèmes

d'ordonnancement et le calcul d'intégrales. Nous ne devons, cependant, pas oublier que ces instantiations nécessitent un travail non négligeable de la part de l'Ingénieur Cognitif, il faut en moyenne ajouter 20% au noyau central. Nous pouvons raisonnablement en conclure, que dans le cas d'un moteur universel, ce travail serait tel que la notion même de généralité perdrait tout son sens.

C'est dans cette optique qu'il faut voir le système LRO-EAQUE qui constitue, nous l'espérons, un pas original vers la maîtrise des techniques de génération de Systèmes Experts.

BIBLIOGRAPHIE

Janin S. AIKINS
"Prototypes and production rules : a knowledge representation for computer consultations"
Stanford University
STAN-CS-80-814

Robert BALZER, Lee ERMAN, Philip LONDON, Chuck WILLIAMS
"Hearsay III : a domain-independent framework for expert systems"
Proceedings of the first annual national conference Artificial Intelligence - august 1980

Avron BARR, Edward A. FEIGENBAUM, Paul R. COHEN
"The handbook of Artificial Intelligence"
Volumes 1,2 et 3. Pitman

BIGRE
"Journée d'étude sur les langages orientés objet"
oct. 83

Robert BLANCHE
"Introduction à la logique contemporaine"
Collection U

Alan BUNDY, Bob WELHAM
"Using meta-level inference for selective application of multiple rewrite rule sets in algebraic manipulation "
Artificial Intelligence Vol.16 1981

M. CAYROL, H. FARRENY,H. PRADE
"Vers l'utilisation des sous ensembles flous en Intelligence Artificielle".

LSI - Toulouse

J. CARLIER, P. CHRETIENNE
"Un domaine très ouvert : les problèmes d'ordonnancement"
RAIRO, volume 16 numéro 3 août 82

Jacqueline CHABRIER
"Présentation et utilisation du langage Prolog"
C.R.I Nancy

Eugene CHARNIAK, Christopher K. RIESBECK, Drew V. McDERMOTT
"Artificial Intelligence programming"
Lawrence Erlbanm associates

William J. CLANCEY, Red LETSINGER
"Neomycin : reconfiguring a rule based expert system for application to teaching"
IJCAI 81

Alain COLMERAUER, Henry KANOUI, Michel VAN CANEGHEM
"Prolog, bases théoriques et développements actuels"
TSI Vol 2 n: 4 1983

CNRS GR22
"programmes d'Intelligence Artificielle utilisant une grande quantité de connaissances"
Sept. 79

CNRS GR22
"application de l'Intelligence Artificielle à l'informatique"
Colloque sept. 77

Randall DAVIS
"Expert systems : where are we? and where do we go from here?"

AI magazine spring 82

Randall DAVIS, Bruce BUCHANAN, Edward SHORTLIFFE
"Production rules as a representation for a knowledge-based consultation program"
Artificial Intelligence Vol.8 n:1 feb.77

Randall DAVIS
"Meta-rules : reasoning about control"
MIT A.I Memo No 576

Yannick DESCOTTE
"Représentation et exploitation de connaissances expertes en génération de plans d'actions"
Thèse de troisième cycle - INPG

H. FARRENY
"Un système de maintenance automatique d'inter-relations dans un système de production"
Troisième congrès de reconnaissance des formes et intelligence artificielle - Nancy setp. 81

H. FARRENY
"Un système pour l'expression et la résolution de problèmes orienté vers le contrôle de robots"
Thèse d'état - Université Paul-Sabatier - Toulouse

Robert FAURE, Catherine ROUCAIROL, Pierre TOLLA
"Chemins et flos, ordonnancements"
Gauthier-Villars - programmation

Edward A. FEIGENBAUM
"The art of Artificial Intelligence : themes and case studies of knowledge engineering"
IJCAI 77

Marius FIESCHI
"Sphinx : un système d'aide à la décision en médecine"
Thèse d'état - Faculté de Médecine - Université d'Aix-Marseille

Richard FIKES
"Knowledge representation in automatic planning systems"
SRI TN 119 1976

Peter FRIEDLAND
"Acquisition of procedural knowledge from domain expert"
IJCAI 81

Olivier GASCUEL
"Un système expert dans le domaine médical"
Thèse de troisième cycle - Université Pierre et Marie Curie

Peter HAMMOND
"Logic programming for expert systems"
University of London - TR DOC 82/4

Roger T. HARTLEY
"How expert should an expert system be ?"
IJCAI 81

Werner HORN, Walter BUCHSTALLER, Robert TRAPPL
"Knowledge structure definition for an expert system in primary medical care"
IJCAI 81

Mitsuri ISHIZUKA, K. S. FU, James T. P. YAO
"Inexact inference for rule-based damage assessment of existing structures".
IJCAI 81

A. KAUFMANN

"Méthodes et modèles de la recherche opérationnelle"
Tomes 1 et 2 - Dunod 72

Jean-Claude LATOMBE
"Une application de l'Intelligence Artificielle à la conception assistée par ordinateur (Tropic)"
Thèse d'état - USMG INPG

LAURE 84 : Jean Pierre LAURENT
"La structure de contrôle dans les systèmes experts"
TSI vol 3 no 3 juin 84

JP. LAURENT, M. AYEL, M. SOUTIF
"Un système expert pour définir des campagnes de reconnaissances géotechniques du sol"
Communications congrès RF-IA AFECT 1984

JP. LAURENT
"Les moteurs de système expert : typologie, comparaisons, guide du concepteur"
Journées S.E Avignon 1984

JP. LAURENT
"A program that computs limits using heuristics to evaluate the indeterminate forms"
Artificial Intelligence Vol.4 1973

JL. LAURIERE
"Sur la coloration de certains hypergraphes application aux problèmes d'emploi du temps"
Thèse 3 cycle - Paris VI

M. LOPEZ
"Communication en langue naturelle avec un système d'aide à la conception d'assemblages physiques : un essai d'utilisation des réseaux sémantiques partitionnés"

Thèse de D.I - INPG

JC. MANGIN
"Prise en compte de tâches répétitives dans l'ordonnancement des travaux de batiment"
Thèse d'état - INSA Lyon

Alberto MARTELLI, Ugo MONTANARI
"An efficient unification algorithm"
ACM Vol.4 No 2 april 82

William MARK
"Rule-based inference in large knowledge base"
Proceedings of the 1 annual national conference artificial intelligence" August 80

W. VAN MELLE, A. C. SCOTT, J. S. BENETT, M. PEAIRS
"The emycin manual"
Stanford University - TSAN-CS-81-885

Robert Carter MOORE
"Reasoning from incomplete knowledge in a procedural deduction system"
MIT - AI-TR-347

Nils J. NILSSON
"Principles of Artificial Intelligence"
TIOGA

J.A. ROBINSON
"Logic programming : past, present and future"
New Generation Computing OHMSHA LTD 1983

Christophe ROCHE
"Systeme expert : étude et développement du système Gari"
D.E.A INPG

Earl D. SACERDOTI
"A structure for plans and behavior"
SRI T.N 109

M. SAKAROVITCH
"Programmation linéaire"
"Eléments de la théorie des graphes"
"Optimisation dans les réseaux"
polycopiés ENSIMAG

Mark J. STEFIK
"Planning with constraints"
Stanford University STAN-CS-80-784

Mark STEFIK, Jan AIKINS, Robert BALZER, John BENOIT, Lawrence BIRNBAUM, Frederich HAYES-ROTH, Earl SACERDOTI
"The organisation of expert systems, a tutorial"
Artificial Intelligence No 18 1982

Mark J. STEFIK, Nancy MARTIN
"A review of knowledge based problem solving as a basis for a genetics experiment designing system"
Stanford University STAN-CS-77-596

Mark STEFIK
"Planning and meta-planning (Molgen)"
Stanford University June 80

Austin TATE
"Nonline : a hierarchic non-linear planner"
University of Edinburgh 1976

A. VERE
"Relational production systems"
Artificial Intelligence Vol.8 1977

Patrick H. WINSTON, Richard H. BROWN
"Artificial Intelligence : an MIT perspective"
Volumes 1 et 2 MIT Press

AUTORISATION de SOUTENANCE

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

VU les rapport de présentation de Monsieur J.P LAURENT, Professeur

Monsieur Christophe ROCHE

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

Fait à Grenoble, le 08 juin 1984

Le Président de l'I.N.P.-G

D. BLOCH
Président
de l'Institut National Polytechnique
de Grenoble

P.O. le Vice-Président,

LRO - EAQUE :

GENERATION DE SYSTEMES EXPERTS

APPLICATION A DES PROBLEMES D'ORDONNANCEMENT

C.ROCHE

RESUME : LRO et EAQUE constituent un environnement pour générer des systèmes experts. Un noyau important existe, à la fois pour définir une certaine représentation des connaissances (LRO, langage orienté objet), et pour définir une structure de contrôle appropriée (EAQUE, moteur d'inférence). Pour chaque instantiation du système LRO-EAQUE à une application particulière, les interfaces adéquates sont écrites par l'Ingénieur Cognitif, après dialogue avec les experts du domaine.
EAQUE a été instancié pour des problèmes d'ordonnancement (ORDF), pour la simplification d'expressions mathématiques (CALINT), et pour simuler un interpréteur PROLOG (EALOG).

MOTS CLES : Intelligence Artificielle, Modes de Représentation des Connaissances, Systèmes experts, Moteurs d'inférence Généraux, Ordonnancement.