

REFLEXIONS

REFLEXIONS

ESPRIT P 387

PAR O. CROISSANT

Document émis le 27 Septembre 1985

Diffusion interne seulement.

ELEMENTS DE SPECIFICATIONS

D'UN OUTIL D'INGENIERIE DE LA CONNAISSANCE

ADAPTE AU DIAGNOSTIC COMPLEXE

DANS UN ENVIRONNEMENT EN EVOLUTION

====

L'idée fondamentale de ce papier est le design d'un outil adapté à l'ingénierie de la connaissance pour la classe de problème suivant :

- problème de diagnostic plus complexe que ceux habituellement traités par les outils à base de règle ;
- grande base de connaissance ;
- environnement temps réel avec une constante de temps compatible avec le temps de réponse de l'outil (probablement quelques secondes à quelques minutes).

CARACTERISTIQUES PRINCIPALES

- Prise en compte du temps et raisonnement sur un monde en évolution.
- Facilité de codage de raisonnements sophistiqués habituellement difficiles à coder dans les outils classiques.

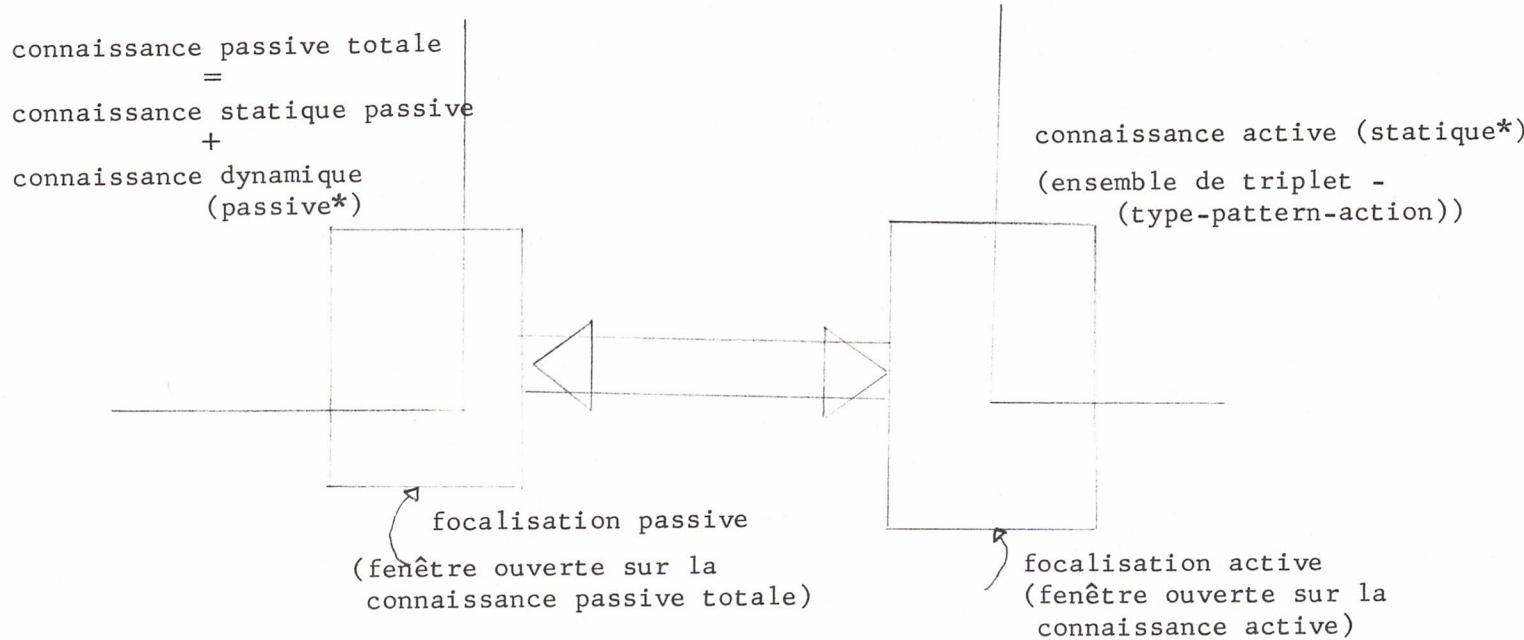
L'outil codera donc de la connaissance : il en existe sous deux statuts : connaissance statique, connaissance dynamique.

Il importe de préciser que la séparation doit être nette et que si l'intersection de ces deux ensembles de types de connaissances n'est pas vide (existence de connaissance statique qui peut être dynamisée), il n'est pas question de permettre à un système de connaissance de changer ses connaissances internes de manière anarchique. Cette rigidité sera garante d'une facilité de mise au point, de maintenance et d'extensibilité. La connaissance dynamique sera le reflet du monde extérieur dont le système aura à faire le diagnostic. Il s'agira d'une connaissance factuelle, structurale, historique et prédictive. Cette connaissance ne pouvant qu'augmenter avec le temps, considérant les limites physiques de l'outil en capacité, celui-ci comprendra un collecteur de connaissance (KC) basé sur de la connaissance, chargé de maintenir un horizon dynamique compatible avec les capacités de l'outil.

La connaissance statique se décompose en deux types de connaissances :

- 1/ la connaissance passive : factuelle et structurale. Celle-ci fournit à l'outil une pré-description du monde ;
- 2/ la connaissance active : celle-ci permet à l'outil de mener le diagnostic en construisant peu à peu le monde dynamique.

Le principe général d'un step de raisonnement est :



La focalisation passive permet aux patterns de la connaissance active de ne chercher à s'instantier que sur un sous-monde du monde passif.

La focalisation active permet à un ensemble de patterns parfaitement délimité d'entrer en concurrence ; celui qui gagne voit son action associée recevoir un signal d'activation.

- * Comme décidé, la connaissance active sera statique et donc la connaissance dynamique sera passive.

Ce step de raisonnement sera caractérisé par les types de connaissance active qui sont en concurrence.

Principalement deux types de connaissance active sont à considérer :

- des descriptions algébriques des relations entre les différentes valeurs paramétrant la connaissance structurale. Ces descriptions permettent de décrire la manière de calculer certaines valeurs quand d'autres sont connues. Il en ressort que ces descriptions s'ordonnent en systèmes de règles à chaînage arrière.
- des blocs de contrôle réalisant du "situation assessment". Ces blocs comprennent deux parties : un pattern qui est instancié si une certaine situation est réalisée, une partie action qui décrit l'ensemble des actions à effectuer si le pattern est instancié. Donc ces blocs se déclenchent suivant un système à chaînage avant ; c'est la seule manière d'obtenir un comportement "event driven" pour les systèmes.

Il faut rajouter à cela deux types de blocs d'actions réalisant des tâches spécifiques et dont le déclenchement est contrôlé par l'outil :

- les blocs appelés "démons" qui sont associés à une valeur ou à un ensemble de valeurs et qui sont activés dès que l'on tente de modifier ou d'accéder à cette valeur ou à une des valeurs de l'ensemble ;
- les blocs dits "collecteurs de connaissance" qui sont activés sur demande de l'outil et permettent de contrôler l'évolution de la connaissance passive dans le temps. Ces blocs entrent en concurrence grâce à un pattern et activent s'il y a lieu un ensemble d'actions associées.

ENTREES - SORTIES :

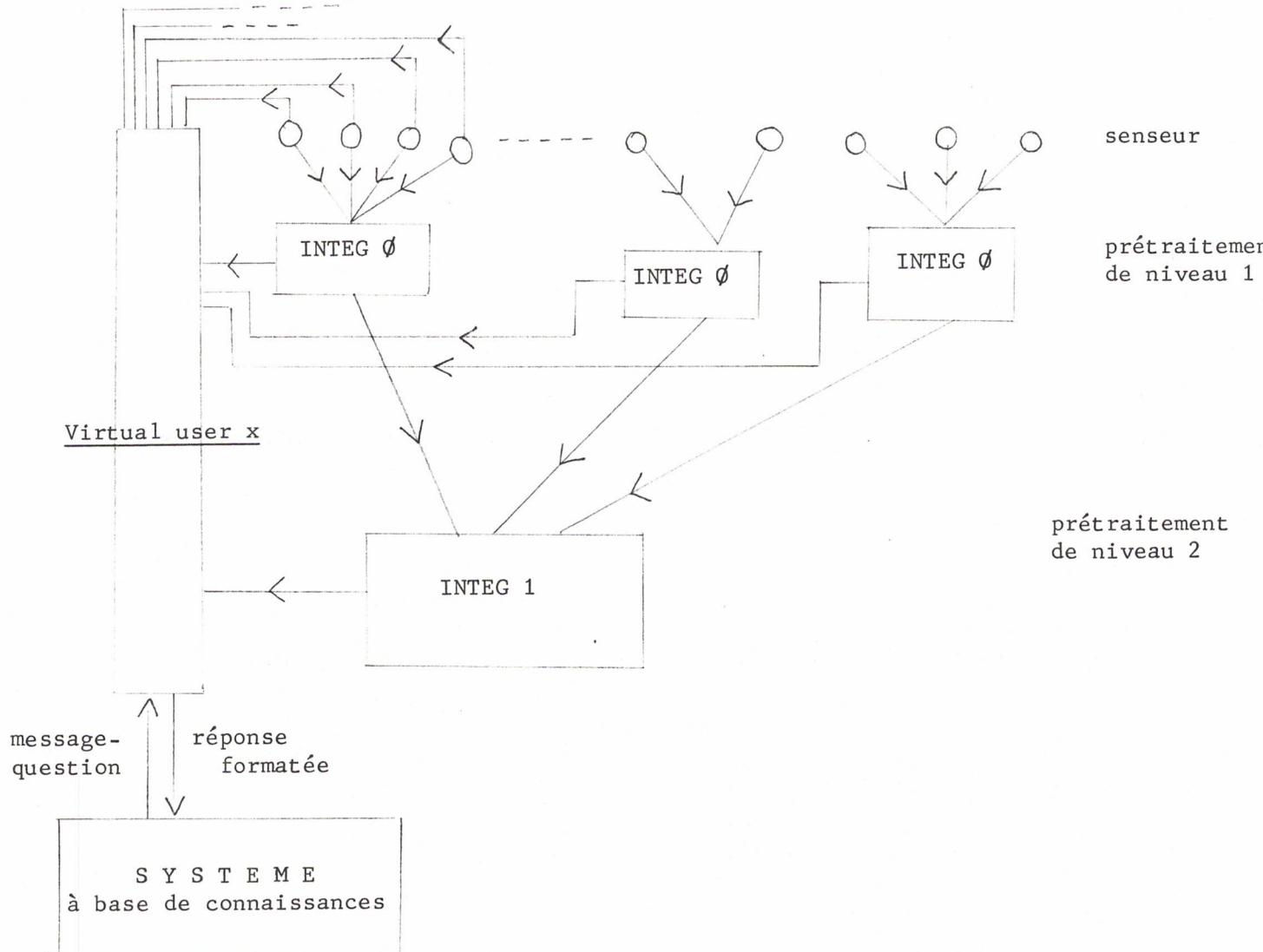
A cause de l'impossibilité pour l'outil d'arrêter le processus de diagnostic en attendant une réponse à une question, un système de message-question est obligatoire : celui-ci

est envoyé à un interface dit "utilisateur virtuel" ; celui-ci gère les questions restées sans réponse et entretient avec l'utilisateur un dialogue en vue d'obtenir des réponses à ces questions.

Dès que l'utilisateur introduit une réponse, celle-ci est formatée par l'utilisateur virtuel et envoyée sur le "bus virtuel d'entrée des informations fournies volontairement" (BVE).

L'utilisateur final est un des utilisateurs virtuels possibles.

Le système procèdera de même avec toutes les questions qu'il est susceptible de poser à tous les interlocuteurs extérieurs. Ceci inclut les interfaces prétraitement sensoriel de tous niveaux. Exemple :



Les réponses formatées ou les informations fournies volontairement constituent des événements extérieurs qui vont devoir être pris en compte par le système. Ceux-ci sont rangés dans une liste d'attente correspondant à leur pouvoir interrupteur.

Il y a deux listes d'attente :

- la liste des événements extérieurs de types compatibles ;
- la liste des événements extérieurs de types micro-compatibles.

Ceci a à voir avec les événements internes dont nous allons parler plus loin, qui peuvent être regardés à deux niveaux :

- un niveau macroscopique correspondant à la notion d'événement global qui comprend plusieurs micro-événements ;
- un niveau microscopique correspondant à la notion de micro-événement. Chaque micro-événement correspond à une exécution d'un bloc de contrôle.

Après chaque micro-événement la liste des événements extérieurs de type micro-compatible est vidée et toutes les modifications associées effectuées.

La notion d'événement micro ou pas a à voir avec l'aspect non monotone de l'outil.

La notion de compatibilité sera examinée lors du chapitre sur la représentation factuelle.

RAISONNEMENT NON MONOTONIQUE

Le raisonnement non monotonique intervient lorsque le système ne possède plus assez de connaissances pour trancher entre deux hypothèses possibles et n'a pas de moyen direct d'obtenir l'invalidation ou la validation de l'une des hypothèses.

Ce peut être le cas lorsque l'on est en présence de deux méthodes : une longue mais générale, l'autre courte mais ne réussissant pas toujours. Dans ce cas là on procède généralement par commencer à essayer les méthodes courtes, puis lorsque celles-ci sont épuisées on se résigne à appliquer la méthode générale. Un autre cas est lorsque différentes possibilités concurrentes existent mais aucune ne paraît plus probable que l'autre et l'outil n'est pas débordé par l'évolution du monde mais on voudrait un

diagnostic aussi rapidement que possible, ou alors certaines hypothèses peuvent ne jamais aboutir à une confirmation ou une infirmation alors que d'autres peuvent aboutir rapidement. L'idée est alors de faire évoluer simultanément plusieurs hypothèses et trancher dès que la possibilité se présentera.

Donc deux traitements de la non-monotonie sont possibles :

- 1/ backtracking, c'est-à-dire privilégier une hypothèse et mettre les autres en réserve au cas où celle-ci échouerait ;
- 2/ traiter toutes les hypothèses parallèlement en attendant que l'évolution comburée du monde extérieur et du raisonnement élimine toutes les hypothèses sauf une.

L'utilisation de ces différentes possibilités peut dépendre du type d'hypothèse et du contexte ; il est donc nécessaire de pouvoir choisir dans le cours du raisonnement le type de méthode qu'il faut employer.

Nous allons représenter l'état en raisonnement qui n'a pas encore fait d'hypothèse par une lettre :

$$\left\{ A \right\}$$

Supposons une hypothèse du premier type, c'est-à-dire telle que soient créées trois hypothèses dont deux sont mises en attente et une troisième est active :

$$\rightarrow \left\{ A_1 / [A_2, A_3] \right\}$$

si l'hypothèse A1 vient à faillir,

$$\rightarrow \left\{ A_2 / [A_3] \right\}$$

si l'hypothèse A2 vient à se confirmer,

$$\rightarrow \left\{ A_2 \right\}$$

Supposons maintenant une hypothèse du second type :

$$\left\{ A \right\} \rightarrow \left\{ A_1 \ A_2 \ A_3 \right\} \text{ les trois processus sont parallèles.}$$

si A1 vient à faillir,

$$\rightarrow \{ A2 \quad A3 \}$$

si A2 vient à se confirmer,

$$\rightarrow \{ A2 \}$$

Supposons un état du raisonnement plus complexe (types 1 et 2 mélangés) :

$$\left\{ \begin{array}{ll} A1 & A2 / [A3 \quad A4] \\ & A5 / [A6 \quad A7] \end{array} \right\}$$

Supposons que A5 vienne à formuler une hypothèse de type 2 :

$$\rightarrow \left\{ \begin{array}{ll} A1 & A2 / [A3 \quad A4] \\ & [A51 \quad A52] / [A6 \quad A7] \end{array} \right\}$$

une évolution possible est :

$$\rightarrow \left\{ \begin{array}{ll} A1 & A2 / [A3 \quad A4] \\ & A51 / [A6 \quad A7] \end{array} \right\}$$

A51 vient à faillir :

$$\rightarrow \left\{ \begin{array}{ll} A1 & A2 / [A3 \quad A4] \\ & A6 / [A7] \end{array} \right\}$$

etc ...

La grammaire de tous les états possibles est donc :

$$\text{état} = \left\{ \text{item } 1 \dots \text{item } n \right\} \quad (n \geq 1)$$

item = processus

$$\text{item} = \text{processus} / [\text{Processus } 1 \dots \text{Processus } k] \quad (k \geq 1)$$

$$\text{item} = [\text{item } 1 \dots \text{item } m] / [[\text{processus } 1 \dots \text{processus } k]] \quad \begin{cases} m \geq 2 \\ k \geq 1 \end{cases}$$

Une hypothèse de type 1 se traduit par une transformation du type :

$$\text{Processus} \longrightarrow \text{Processus} / [[\text{Processus } 1 \dots \text{Processus } k]] \quad (k \geq 1)$$

Une hypothèse de type 2 se traduit par une transformation du type :

$$\left\{ - \text{ processus} - \right\} \rightarrow \left\{ - \text{ processus 1} \dots \text{ processus } n - \right\} (n \geq 1)$$

$$[- \text{ processus} -] \rightarrow [- \text{ processus 1} \dots \text{ processus } n -] (n \geq 1)$$

$$\text{processus} / \boxed{-} \rightarrow [\text{processus 1} \dots \text{ processus } m] / \boxed{-} \quad (m \geq 2)$$

La faillite d'une hypothèse active se traduit par une transformation du type :

$$\left\{ - \text{ processus} \dots \right\} \rightarrow \left\{ - \dots \right\} \quad \neg(\text{vide et} \dots \text{ vide})$$

$$[- \text{ processus} \dots] \rightarrow [- \dots] \quad \neg(\text{vide et} \dots \text{ vide})$$

$$[\text{processus 1} \quad \text{processus 2}] \rightarrow \text{processus 2}$$

$$\text{processus} / \boxed{\text{processus 2}} \rightarrow \text{processus 2}$$

$$\text{processus} / [\text{Processus 1} \dots \text{ Processus } m] \rightarrow \text{processus} / [\text{Processus 2} \dots \text{ Processus } m] \quad (m \geq 2)$$

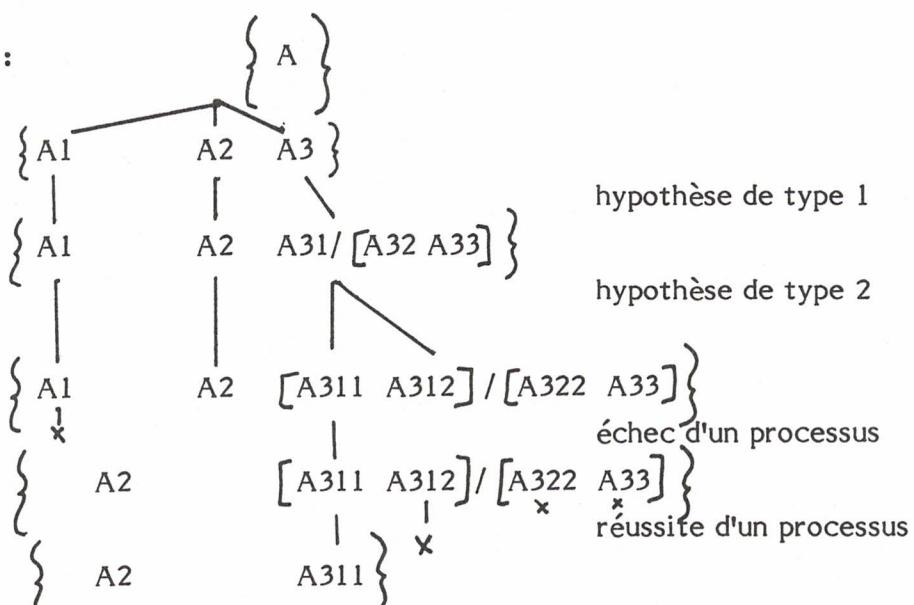
La réussite d'une hypothèse active est plus délicate à traiter car cette réussite traduit la faillite d'un certain nombre d'hypothèses concurrentes. Mais il convient d'établir la notion de niveau de concurrence qui met en lumière la difficulté :

soit le processus : $x = \{ A_1 \ A_2 \ A_3 \}$

Ce niveau d'hypothèse référence donc une hypothèse de type 2 qui a été faite sur un processus ancêtre du processus qui réussit. Tous les processus qui sont fils de ce processus ancêtre et qui sont différents du processus qui réussit doivent être tués.

La grammaire des transformations associées, tout en étant plus complexe, n'en respecte pas moins \mathcal{H} . Elle se compose d'un ensemble de transformations de type "échecs de processus" dépendant de l'historique. Il est mis aussi en évidence ici que le formalisme d'état utilisé est incomplet ; il doit y être adjoint un historique des hypothèses de type I et II. On peut composer une représentation adéquate en juxtaposant les "états" définis précédemment le long d'une ordonnée symbolisant le temps (suite d'événements internes).

Exemple :



Ce processus peut avoir été généré de plusieurs manières possibles :

$$\begin{array}{ccc} \{ A \} & \xrightarrow{\text{Hyp. 1}} & \{ A_1 \ A_2 \ A_3 \} \\ \text{état 1} & & \text{état 0} \end{array} \quad \text{I}$$

ou

$$\begin{array}{ccc} \{ A \} & \xrightarrow{\text{Hyp. 2}} & \{ A_1 \ B \} \xrightarrow{\text{Hyp. 1}} \{ A_1 \ A_2 \ A_3 \} \\ \text{état 2} & & \text{état 1} \quad \text{état 0} \end{array} \quad \text{II}$$

ou

$$\begin{array}{ccc} \{ A \} & \xrightarrow{\text{Hyp. 2}} & \{ C \ A_3 \} \xrightarrow{\text{Hyp. 1}} \{ A_1 \ A_2 \ A_3 \} \\ \text{état 2} & & \text{état 1} \quad \text{état 0} \end{array} \quad \text{III}$$

ou ...

Quand on veut propager la réussite de A_3 , il faut décider quel processus actif tuer ; cela dépend du niveau de l'hypothèse que l'on entend concerner par cette réussite.

cas (I) A3 réussit/_{Hyp. 1} → on tue A1, A2

cas (II) A3 réussit/_{Hyp. 1} → on tue A2
A3 réussit/_{Hyp. 2} → on tue A1, A2

cas (III) A3 réussit/_{Hyp. 1} → on tue A1, A2
A3 réussit/_{Hyp. 2} → on tue A1, A2

REPRESENTATION FACTUELLE ET STRUCTURALE

Les contraintes sur cette représentation factuelle et structurale sont les suivantes :

- description facile d'un dispositif industriel ;
- description facile des notions de contexte, de crise, d'idée de diagnostic permettant de raisonner au niveau Meta ;

- arbre dynamique unique interprétant le parallélisme des processus par une notion de coupe diachrone dans l'arbre dynamique ;
- création de structure à des fins d'hypothèse facilitée par un langage de haut niveau pour effectuer ces créations ;
- notion d'objet induit et implicite permettant de manipuler des concepts comme si l'ensemble des variables dynamiques attachées à ce concept était créé, alors que cela n'est vrai qu'en partie (seules seront créées les variables utiles) : ceci permet de manipuler des objets lourds ou infinis.

Cette connaissance aura deux statuts possibles dans un système opérationnel (outil + base de connaissance) :

- * pré-connaissance de l'environnement : ensemble des structures et faits déjà connus lors du démarrage d'une consultation du système ;
- * connaissance acquise grâce au raisonnement et à la prise en compte des événements extérieurs et des réponses aux questions. Dans ce cas, deux types d'acquisition de connaissances sont envisageables :
 - conclusion pour la valeur d'attributs et de variables dynamiques déjà créés (pas de création de variable dynamique) ;
 - création de variables dynamiques grâce à tout un tas de mécanismes :
 - .. création explicite ou implicite d'ensembles de variables dynamiques à partir d'un canevas ou type d'objet : instanciation ;
 - .. création d'hypothèse de type 1 ou 2 ;
 - .. création explicite ou implicite d'ensembles de variables dynamiques à partir d'un ensemble de meta-canevas ou meta-objet : interprétation d'un statement de création structurelle ;
 - .. création explicite ou implicite d'un type particulier d'objets : objets induits puis en fonction des besoins du raisonnement, création des variables dynamiques associées nécessaires.

NOTION DE META OBJET

Objet de base : FOO, BAR, BAZ

Attribut de base : ATT1 (FOO) ATT2 (BAR) ATT3 (FOO, BAR)
 ATT4 (FOO, BAR, BAZ)

Meta objet TOTO (FOO1 : FOO) [(BAR1 : BAR, BAZ1 : BAZ)

tel que ATT3 (FOO1, BAR1) = Yes

ATT4 (FOO1, BAR1, BAZ1) = Yes]

variables liées : connexion de l'objet avec le monde

Meta objet TITI (TOTO1 : TOTO)

variables libres : structure interne de l'objet

[(FOO1 : FOO) | soit FOO2 = ARG [TOTO1, 1]

et BAR2 = STRUCT [TOTO1, 1]]

et BAZ2 = STRUCT [TOTO1, 2]]

alors ATT4 (FOO1, BAR2, BAZ2) = Yes

et ATT1 (FOO1) = Yes.

Un meta objet peut avoir pour variable des objets ou des meta objets et peut avoir pour définition une structure à base d'objets ou de meta objets mais aucun cercle vicieux de définition n'est toléré ! Il peut être utilisé les statements if, then, else

La notion de meta objet est utilisée pour construire facilement des structures compliquées d'objets et les manipuler *.

- * Cette notion permet d'englober la notion de meta class rencontrée dans small talk en rejetant un particularisme de linstanciation dans un super objet mais permet en plus de manipuler les relations entre objet.

Elle sert aussi à simplifier les patterns des différents éléments de connaissance actifs et à les rendre lisibles ; dans ce cas-là la notion de meta-objet est utilisée comme une contrainte. On parlera alors de meta-contrainte.

Les meta objets peuvent servir à structurer un arbre dynamique ; néanmoins les meta objets de type induit sont plus généraux et forment un ensemble de déclarations de type qui structurent effectivement l'arbre dynamique.

NOTION DE META OBJETS DE TYPE INDUIT (object induit)

Il s'agit d'une déclaration de type ressemblant à une déclaration de meta objets mais où sont tolérés les opérateurs suivants : dans la structure :

Set of [FOO1 : FOO, BAR1 : BAR] (non ordonné)

Seq of [FOO1 : FOO, BAR1 : BAR]

Seq of [(FOO1 : FOO, BAR1 : BAR)]

ainsi qu'une self référence au type que l'on est en train de définir.

Ces "objects" sont en fait des structures qui peuvent être vérifiées ou trouvées dans l'arbre dynamique, mais ne peuvent être instanciées..

OBJETS LOCAUX

Un objet local est un objet qui n'a pas de type générique dans la consultation, mais qui est défini dans le corps d'une procédure ou d'une déclaration : il sert à simplifier le langage et à faciliter la lecture des blocs de contrôle. C'est une notion de meta objet induit local :

Exemple : soit $E = (\text{Set of } x : \text{FOO} \mid \text{att}_1(x) = 6.2 \text{ et } \text{att}_2(x) < 1)$.

Les "statements" HCE , $\text{H}\cancel{\text{C}}\text{E}$, $\text{Y}\cancel{\text{C}}\text{E}$, $\text{Y}\cancel{\text{E}}$ seront alors compilés en l'ensemble de contraintes qui sont nécessaires.

FOCALISATION, COMPATIBILITE ET META OBJETS

En fait les meta objets ont un autre rôle essentiel : ils permettent de délimiter la focalisation dans l'arbre dynamique ; mais aussi et surtout les propriétés de compatibilité et d'incompatibilité avec les informations fournies volontairement par les utilisateurs virtuels sont décrites dans les meta objets de manière standard. Tous les meta objets inclus dans la focalisation sont protégés des modifications intempestives dues aux événements extérieurs. Fondamentalement la création d'hypothèse de type 1 ou 2 n'est rien d'autre qu'une création de meta objets d'un type particulier : les hypothèses, et d'un meta objet de focalisation : l'ensemble des hypothèses.

Néanmoins il est possible de créer une seconde focalisation qui regarde les micro-événements et restreint la focalisation de l'arbre dynamique à la simple hypothèse que l'on est en train de regarder ; cette seconde focalisation (paramétrable dynamiquement) peut inclure des événements de manière exhaustive ou les exclure.

LA FOCALISATION

La base de connaissance est constituée comme un ensemble d'objets dont la place physique dans la base n'est pas significative (déclarativité). Tous ces objets ont un nom mais peuvent être groupés suivant les besoins par des objets particuliers que sont les objets de focalisation.

Un objet de focalisation permet de grouper des objets de base comme des règles de back-chaining ou des blocs de forward chaining, des déclarations génériques de type, mais aussi des éléments de l'arbre dynamique, ...

Ces objets permettent aussi de faire référence à d'autres objets de focalisation. Enfin ils peuvent posséder un pattern qui permet de les mettre en concurrence depuis un autre objet de focalisation ou depuis un block forward de contrôle.

Ces objets de focalisation possèdent aussi une notion de type abstrait, permettant le maniement par blocs et la création de notion de contexte de focalisation.

L'expérience de S1 nous a montré l'adéquation de la représentation de la connaissance sous la forme :

attribut [objet 1, ... objet n] = Valeur,

celle-ci permettant de raisonner facilement sur des objets en relation.

Néanmoins le maniement des ensembles d'objets dynamiques comporte une difficulté : celle des relations entre objets inclus dans les ensembles manipulés et objets extérieurs à ces ensembles. Il est clair que les attributs responsables de ces relations ont leur valeur associée à l'ensemble manipulé, et ce de manière standard. C'est une des raisons pour la notion de variables des meta-objets nous permettant de swapper des meta-objets dans l'arbre dynamique sans conflit de valeurs d'attributs. La focalisation permet la manipulation de deux types d'entité :

- des entités dynamiques : meta-objets ;
- des entités type générique avec connaissance active.

Il est évident qu'il doit exister une certaine cohérence entre la manipulation de ces deux types d'objet. Ceci sera vérifié à chaque manipulation dans l'implémentation d'étude.

La focalisation a trois buts :

- définir clairement les contraintes de compatibilité et micro-compatibilité en protégeant explicitement certaines zones de l'arbre dynamique contre des événements extérieurs d'un certain type (type abstrait). Cette protection est modifiable dynamiquement et gérée automatiquement ;
- permettre l'utilisation de grosses bases de connaissance générique ou de cas particuliers d'objets (meta-objets) en précisant sur quels ensembles de type générique et quels ensembles d'éléments actifs le moteur d'inférence a à régner ;
- permettre explicitement une certaine conduite du raisonnement, la direction de celui-ci étant fixée par la composition en éléments actifs de ce sur quoi le moteur d'inférence règne.

On peut formellement exprimer ces focalisations de la manière suivante :

$$\begin{aligned}
 \text{FE} &= \left\{ M_1 \dots M_i \right\} \\
 \text{FEM} &= \left\{ M'_1 \dots M'_i \right\}
 \end{aligned}
 \quad \left. \begin{array}{l} \text{protection contre les} \\ \text{événements extérieurs} \end{array} \right\}$$

M_j ou M'_j sont des items
item

m_j est un meta objet non instancié et t_j un type abstrait.

$\left. \begin{array}{l} \text{Interdit tout type } [m_1 \dots m_n] \\ \text{interdit } [m_1 \dots m_n] \text{ aux types} \\ \quad [t_1 \dots t_n] \\ \text{autorisé } [m_1 \dots m_n] \text{ uniquement} \\ \quad \text{aux types } [t_1 \dots t_n] \\ \text{item}_1 \text{ et item}_2 \text{ et ... et item}_n \end{array} \right\}$

$$FA = \{ z_1 \dots z_n \} \quad \} \quad \begin{array}{l} \text{base sur laquelle raisonne} \\ \text{le moteur d'inférence} \end{array}$$

z_i est un objet de focalisation.

La focalisation passive est reliée au KBC collecteur de connaissance. Ces deux entités déterminent quelle fenêtre il convient d'ouvrir sur l'arbre dynamique aux éléments actifs référencés par RT. Il est symbolisé par :

$$FP = \{ D_1 \dots D_n \} \quad D_i : \text{désignateur.}$$

Les D_i permettent une classification et une gestion efficace de l'arbre dynamique ; c'est un désignateur d'une partie de l'arbre dynamique ; il s'agit en fait d'objet induit.

La focalisation du système est donc donnée par l'état :

$$F = (FE, FEM, FA, FP).$$

En fait à cause du raisonnement non monotonique, il peut exister plusieurs monde parallèles, chacun d'entre eux possédant sa propre focalisation car celle-ci est dynamique, donc peut dépendre des hypothèses sous-tendant le monde hypothétique.

Considérant donc la représentation introduite dans le chapitre sur la non monotonicité, il convient d'associer à chaque processus actif une focalisation $P_j \rightarrow F_j$.

Un problème survient alors lorsque, à la suite d'une hypothèse de type I, un backtracking est déclenché, c'est-à-dire qu'un autre choix est fait, ce qui invalide toutes les conclusions faites depuis le choix, remet dans les piles tous les événements extérieurs arrivés depuis ce moment et tente de remettre en place une focalisation adaptée à la circonstance ; pour l'ensemble d'événements extérieurs, l'ensemble des modifications imposé par l'extérieur sera fait conformément à la focalisation adoptée. Celle-ci peut être ou la focalisation initiale, ou la focalisation qui a déclenché

le backtracking. Comme il est très difficile de choisir, une solution intermédiaire a été adoptée : le focusing étant un ensemble de liste :

$$F = ((a_1 \dots a_n) \dots (b_1 \dots b_m)).$$

Celui-ci peut être marqué par des pointeurs qui divisent F :

$$F = \left((a_1 \dots a_j \ a_{j+1} \dots a_n) \dots (b_1 \dots b_k \ b_{k+1} \dots b_m) \ j \dots k \right)$$

ou

$$f = \left\{ \underbrace{(a_1 \dots a_j) \dots (b_1 \dots b_k)}_{\begin{array}{l} \text{partie indépendante de} \\ \text{l'hypothèse passée et} \\ \text{dépendant du contexte actuel} \end{array}} \underbrace{(a_{j+1} \dots a_n) \dots (b_{k+1} \dots b_m)}_{\begin{array}{l} \text{partie dépendant de l'hypothèse} \\ \text{et indépendante du contexte actuel} \end{array}} \right\}$$

F_O F_H

On peut écrire que :

$$F = F_O + F_H$$

Au moment de la formulation de l'hypothèse de type I on a :

$$F^1 = F_O^1 + F_H^1$$

Au moment du backtracking on a :

$$F^2 = F_O^2 + F_H^2$$

Juste après le backtracking, la refocalisation est :

$$F^2 = F_O^2 + F_H^1$$

ce qui permet de tenir compte de l'évolution globale du monde avec une redirection de l'hypothèse ; en fait c'est en accord avec le fait que la focalisation doit permettre de représenter et de conduire le raisonnement de manière duale à l'évolution de l'arbre dynamique.

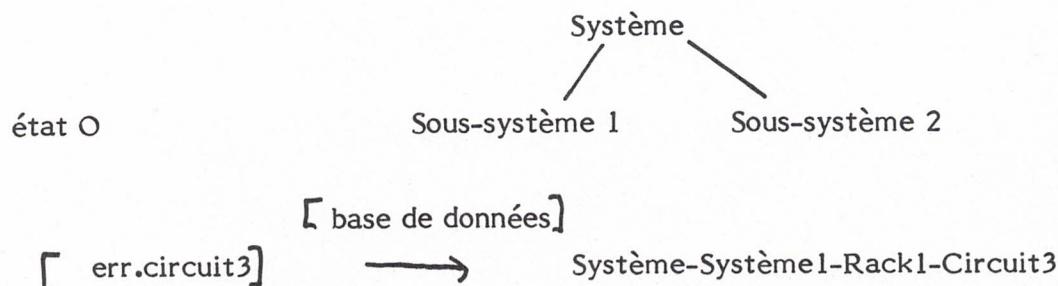
Au début de la consultation seul un certain nombre d'objets sont chargés: ils forment un ensemble stratifié par un certain nombre de hiérarchies. Le chargement d'objets supplémentaires va se faire soit par package prédéterminé, soit par approfondissement particulier de l'arbre. Dans tous les cas à tout instant, il forme un ensemble stratifié pour toutes les hiérarchies. L'intérêt d'utiliser cette dernière procédure est l'économie du nombre de patterns instanciables. L'intérêt du package est de pouvoir intégrer un certain nombre de règles concernant le niveau et l'endroit du circuit examiné.

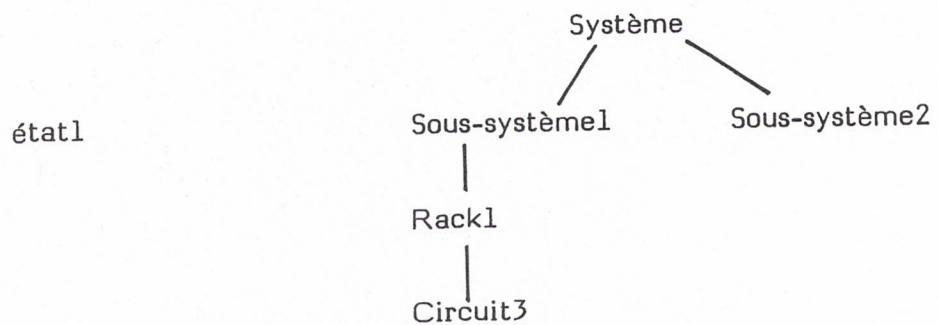
L'approfondissement dynamique a lieu dans deux cas distincts:

- sur réception d'un message concernant un objet non encore instancié,
- lors de la tentative d'instanciation d'un quantifieur universel ou existenciel concernant des objets non présents, mais référencés dans les objets présents.

Le principe de codage de relations entre objets dans toute la base est la relation doublement chaînée: ceci pour assurer la vitesse d'instanciation des patterns maximum, mais aussi pour rendre possible la notion de cut-off.

Approfondissement dynamique automatique
de l'arbre objet sur réception d'un message d'erreur





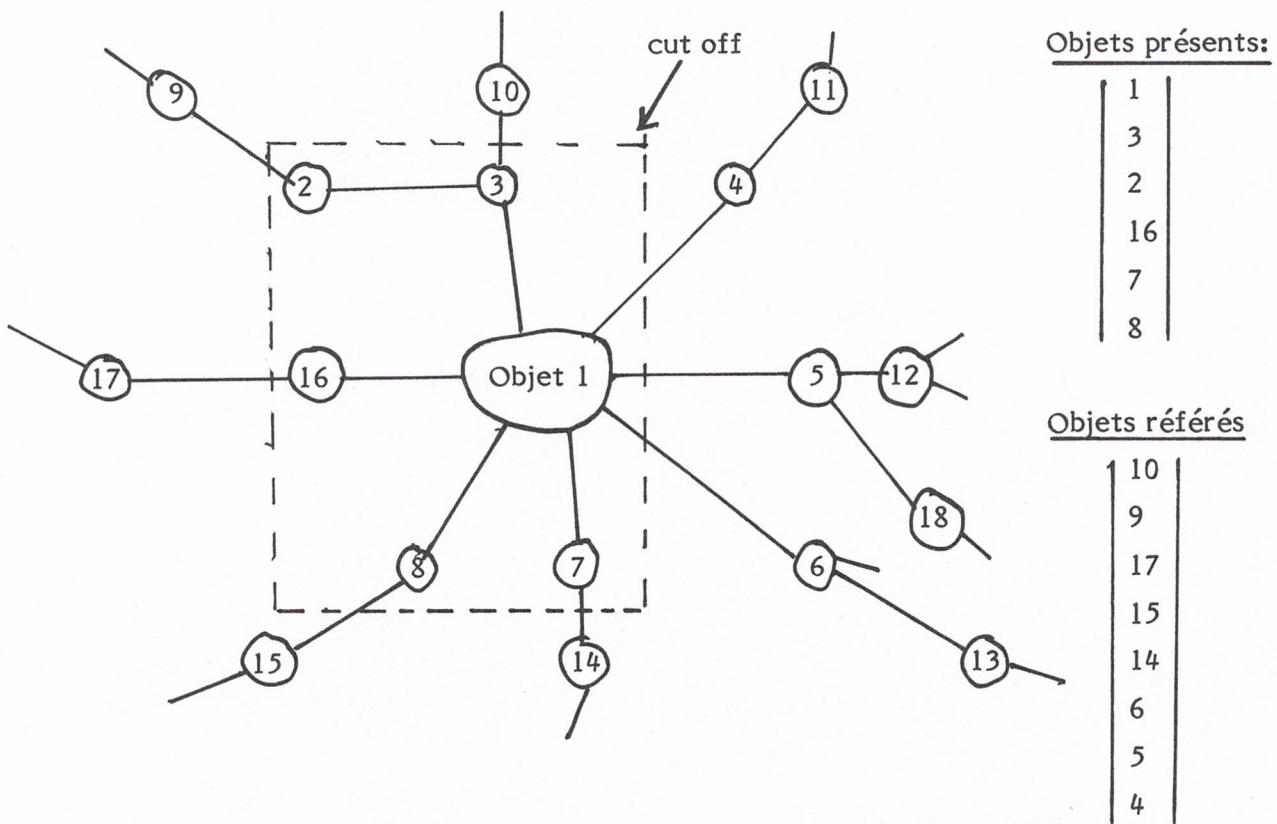
Toute la hiérarchie est chargée afin d'avoir pour un objet toutes les caractéristiques le concernant.

LE CUT-OFF

Formellement dans un réseau d'objets, le cut-off intervient quand on ne peut examiner tous les objets: seule une partie du réseau est donc concernée par la connaissance active.

Cette partie comprend deux sortes d'objets:

- les objets présent dans la focalisation,
- les objets virtuels qui y sont seulement référencés à travers les liens doublement chaînés. Ceux-ci forment le cut-off de la focalisation. Il s'agit d'une limite mouvante qui change chaque fois que de nouveaux objets sont tirés de la base de données pour être instanciés.



REMARQUES SUR L'UTILISATION DU PROCESS

- on fait des hypothèses parallèles lorsque l'on tente de découvrir un diagnostic et qu'on est limité par le feedback de l'installation,
- on tente de classifier des hypothèses et on essaie d'abord les cas les plus probables ou plus rapides dans les autres cas.

Qu'est-ce qu'une panne à diagnostiquer ?

- .. c'est :
 - . un circuit défaillant, ou plusieurs,
 - . une fonctionnalité en dehors de sa zone de fonctionnement normal.

Qu'est-ce qu'un problème dont on recherche la solution ?

- ..c'est :
 - . l'imminence d'une panne qui se rapproche,
 - . un petit problème qui peut se transformer en gros.

- Le diagnostiqueur peut faire deux types d'action :
 - .. tâcher de diagnostiquer le problème,
 - .. tâcher de prendre en compte l'urgence des problèmes.

- Démons temporels :

- .. en fonction du temps associé à un certain nombre d'événements antérieurs $[t(e_i)]_{i \in J}$ et du temps courant, calcul d'une certaine probabilité d'occurrence de l'hypothèse k :

$$f_k(t(e_1) \dots t(e_J), t) \rightarrow 0 \text{ qd } t \rightarrow \infty$$

La seule manière de traiter correctement le temps, c'est d'utiliser le multiprocessing, simulé ou non.

1. Tous les processus attendent un événement du type x , en fonction du contenu de l'événement de type x , telle ou telle hypothèse sera avortée, telle ou telle hypothèse sera confirmée.
2. Tous les processus évoluent selon un examen permanent de l'évolution du monde extérieur, les probabilités f_k évoluant et franchissant certains seuils, certaines hypothèses se trouveront confirmées ou infirmées.
3. Tous les processus contruisent en fonction du monde extérieur un diagramme de fonctionnement qui sélectionne le fonctionnement supposé. En fonction des événements observés, une probabilité peut leur être associée.

Les hypothèses de type 1 , au contraire, permettent de coder de manière automatique plusieurs approches d'un même problème qui sont exclusives les unes des autres, car chacune recommande des actions ou des tests différents (et les exécute ou attend leur résultat).

REMARQUES SUR LE DIAGNOSTIC

- A coder la règle suivante :
 - .. si c'est le delco, alors d'ici deux minutes, tu vas avoir une fuite d'huile,
sinon ce sera autre chose.
- Création de deux process :
 1. C'est le delco → on examine les conséquences d'une telle conviction,
 2. Ce n'est pas le delco → on examine les conséquences d'une telle conviction.

et dans les deux process, un démon temporel associé voit sa probabilité diminuer ou augmenter en fonction des événements extérieurs.

IDEE : On peut associer de manière standard, un démon temporel à tout process créé, et comme les process sont interprétés en forward, des règles permettent de regarder l'évolution de cette probabilité et d'en tenir compte. Dans ce cas là, le raisonnement incertain existe, mais son usage est restreint à l'association (process probabilité), ce qui le rend moins désavantageux par rapport au problème de la difficulté de déterminer précisément des seuils, ainsi les seuils seront associés à des temps caractéristiques, ce qui permettrait un calcul plus aisé ($P = e^{kt}$).

M E R G I N G

Un démon temporel peut décider d'un merging total ou partiel, la procédure a pour paramètre un noeud de l'arbre des hypothèses et mergera tout en dessous de ce noeud, créant un nouveau processus issu du noeud et détruisant tous les process associés à ce noeud par descendance. Le principe est le suivant :

- On procède par le biais de l'arbre et on merge les process les plus jeunes, puis, on remonte en remergeant les process du dessus et ainsi de suite jusqu'au noeud désigné. Le merging de deux process s'effectue de la manière suivante :
 - .. si les objets sont différents, on les additionne;
 - .. si les objets sont les mêmes et les attributs différents, on les additionne;
 - .. si les objets sont les mêmes et les attributs aussi, et que la valeur de l'un d'eux est une liste on additionne à cette liste l'autre;
 - .. si les deux valeurs sont une liste, on les fusionne;
 - .. si aucune des valeurs n'a de liste, on crée une liste et on y insère les deux valeurs.

L'intérêt d'un langage tel que Glisp, est de manipuler des structures internes pour les objets, des structures exotiques en général, mais ces structures doivent être utilisées le moins possible. Néanmoins, il convient de doter l'outil d'accès facile dans ces structures. Pour l'homogénéité de l'outil et son optimisation, il faut pouvoir décrire ces structures de manière modulaire et sous forme d'éléments de connaissance de la K.B. Toutefois, cette partie du système total sera opaque aux procédures d'explication.

L'explication basique est fournie par la mémorisation des événements élémentaires intervenus au cours d'une consultation dans les fichiers. Ceux-ci étant exploités par les why et how.

Il sera possible néanmoins de fabriquer une explication plus sophistiquée faisant intervenir des prises en compte d'éléments supplémentaires rentrés dans la base de connaissances sous forme d'objets d'explication. Ceux-ci permettront d'obtenir une "customized explanation"

ENSEMBLE STRATIFIES

Soit une théorie dans laquelle les théorèmes sont de la forme :

(P , O₁...O_n)

prédicat objet

Définition

- On appelle ensemble stratifié, la donnée $K = (X, \mathcal{M}, t_0, \sqsubset)$

X est un ensemble d'objets, une application de $X \rightarrow \mathbb{N}$

\sqsubset une application de $X - \{t_0\} \rightarrow X$ ($\sqsubset(x) = y$ sera noté $x \sqsubset y$)
qui vérifient les hypothèses suivantes :

- X est fini
- $x \sqsubset y \Rightarrow \mathcal{M}(y) = \mathcal{M}(x) + 1$

LEMME 1

Soit $K = (X, \mathcal{M}, x_0, \sqsubset)$ un ensemble stratifié
 $\forall x \in X, \exists ! (y)_{0 \leq n \leq K} \quad \forall i \quad 0 < i < k-1 \quad Y_i \sqsubset Y_{i+1}$

et
$$\begin{cases} Y_0 = x \\ Y_k = t_0 \end{cases}$$

Démonstration

- Construisons la suite de manière nécessaire :

$$1^{\circ} \quad Y_0 = x$$

2^o Y_i est défini $\Rightarrow Y_{i+1}$ tel que $Y_i \neq Y_{i+1}$, si c'est possible.

Tous les Y_i sont différents car $\mathcal{U}(Y_{i+j}) = \mathcal{U}(Y_i) + j$.

Donc la suite se termine à un Y_k tel qu'il est impossible de trouver $x \in X$ tel que $Y_k \neq x$

$$\text{donc } Y_k = t_0$$

la suite est donc définie de manière unique.

LEMME 2

Soit $\mathfrak{X} = (X, \mathcal{U}, x_0, \mathcal{H})$ est un ensemble stratifié,

$\forall (x, y) \in X^2, \mathcal{U}(x) - \mathcal{U}(y)$ est indépendant de \mathcal{U}

Démonstration

Par application du lemme 0, $\exists x_0, \dots, x_n$ tel que $x_0 = x$ et $x_n = t_0$
 y_0, \dots, y_p tel que $y_0 = y$ et $y_p = t_0$

$$\mathcal{U}(x_n) = \mathcal{U}(x) + n = \mathcal{U}(t_0)$$

$$\mathcal{U}(y_p) = \mathcal{U}(y) + p = \mathcal{U}(t_0)$$

donc par soustraction $\mathcal{U}(x) - \mathcal{U}(y) = p - n$ est indépendant de \mathcal{U}

définition Soit $\mathcal{X} = (X, \mathcal{M}, x_0, \vdash)$ un ensemble stratifié
soit $(x_0 \dots x_n) \in X^n$ tel $x_i \vdash x_{i+1}$ $0 \leq i \leq n-1$

On notera $x_0 \vdash x_n$

Soit $\mathcal{Y} = (Y, \mathcal{M}', x_0, \vdash')$
tel $X \subset Y$, $\mathcal{M}'|_X \equiv \mathcal{M}$, $\vdash'|_X \equiv \vdash$

\mathcal{Y} est plus précis que \mathcal{X} , \mathcal{Y} est une précision de \mathcal{X} , $\mathcal{X} \subset \mathcal{Y}$

LEMME 3

\vdash et $|C$ sont transitives

Démonstration évidente.

Définition

soit P un prédicat sur $\mathcal{X} = (X, \mathcal{M}, x_0, \vdash)$

P est bien formé $\left\{ \forall (O_1 \dots O_n) \in X^n \text{ et } (O'_1 \dots O'_n) \in X^n \right.$

tel $O_i \vdash O'_i$ et $(P \ O_1 \dots O_n)$

alors $(P \ O'_1 \dots O'_n)$ $\left. \right\}$

Théorème fondamental

soit $\mathcal{X} = (X, \mathcal{M}, x_0, \vdash)$ et $\mathcal{Y} = (Y, \mathcal{M}', x_0, \vdash')$
deux ensembles stratifiés tel $\mathcal{X} \subset \mathcal{Y}$

alors P bien formé sur $\mathcal{X} \Rightarrow \exists ! P'$ bien formé sur \mathcal{Y}
tel $P'|_{\mathcal{X}} = P$

Démonstration

Supposons que $Y = X + \{z\}$

dans P' $\exists x \in X$ tel $z \vdash x$

Définissons P' par $(P \setminus \alpha_1 \dots \alpha_i \setminus x \setminus \alpha_{i+1} \dots \alpha_n) \cup (P' \setminus \alpha_1 \dots \alpha_i \cup z \setminus \alpha_{i+1} \dots \alpha_n)$

P' est bien défini et unique.

Revenons à un Y général:

Démontrons le lemme général suivant:

LEMME 4 $X \subset Y$ implique $\exists (z_1 \dots z_n) \in (Y - X)^n$
tel que l'on peut définir une suite $Y_0 \dots Y_n$

tel $X = Y_0 \subset Y_1 \dots \subset Y_n = Y$ et $Y_{i+1} = Y_i + \{z_i\}$

Démonstration par récurrence

1. $X = Y_0$

2. $\left. \begin{array}{l} Y_i \subset Y \\ Y_i \neq Y \end{array} \right\} \Rightarrow \exists z \in Y \text{ et } z \notin Y_i$

par le lemme $\exists p \in \mathbb{N}$ et $x_0 \dots x_p$ tel $x_0 = z$ et $x_p = t_0$

et $x_j \vdash x_{j+1}$ $0 \leq j \leq p-1$

soit x_1 le premier élément $\notin Y_i$ en descendant la suite depuis x_p
cet élément existe car $x_0 \notin Y_i$

donc $x_1 \notin Y_i$ et $x_{i+1} \in Y_i$

on pose $z_i = x_i$ et $Y_{i+1} = Y_i + \{z_i\}$ $Y_i \subset Y_{i+1}$

donc la suite existe (note: elle n'est pas unique). ■

Revenons au théorème fondamental:

Comme $Y - X$ est fini en appliquant le lemme 4, le théorème est démontré par recurrence sur le nombre d'éléments de $Y - X$ ■

L'intérêt des prédictats bien formés est qu'ils représentent des assertions qui restent vraies si on précise les choses, donc à partir d'une certaine précision indépendante de la précision. Ces assertions peuvent être démontrées une bonne fois pour toute dans un monde, où le nombre d'objets évolue, pourvu qu'il évolue en se précisant davantage. Donc, on peut écrire un système de règles permettant de déduire ces prédictats et ce système de règles étant déclenché en backtracking et provoquant la création d'objets qui précisent la marche et permettent de démontrer des assertions. C'est donc un chainage arrière avec forte création dynamique. Les règles de réécriture doivent être telles, qu'à partir de prédictats bien formés, elles n'enchaînent que la déduction d'assertions avec d'autres prédictats bien formés.

- Exemple d'ensembles stratifiés

$$X = (x, y, \underbrace{1, 2, \underbrace{3, 0}}_{n=0})$$
$$\quad \quad \quad n = -1 \quad n = -2 \quad n=0$$

et pour \vdash

:



- Exemple d'un prédictat bien formé

$$(P \alpha \alpha') \equiv (\alpha = \alpha' - 1 \text{ ou } \alpha = \alpha' - 2)$$

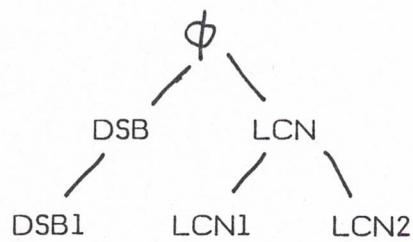
car $(P x y), (P 1 3), (P 2 3)$

ou alors:

$$X = (\emptyset, DSB, LCN, DSB1, LCN1, LCN2)$$

n=0 -1 -1 -2 -2 -2

et pour \vdash



$$(P \ x \ y) \equiv x \text{ influence } y$$
