(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

PARIS

(11) N° de publication : **2 607 284**
(à n'utiliser que pour les commandes de reproduction)

(21) N° d'enregistrement national : **87 11692**

(51) Int Cl⁴ : G 06 F 15/18, 9/00.

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt :** 18 août 1987.

(30) **Priorité :** US, 18 août 1986, n° 897 709.

(43) **Date de la mise à disposition du public de la demande :** BOPI « Brevets » n° 21 du 27 mai 1988.

(60) **Références à d'autres documents nationaux apparentés :**

(71) **Demandeur(s) :** *Société dite : NEURON DATA, INC.* — US.

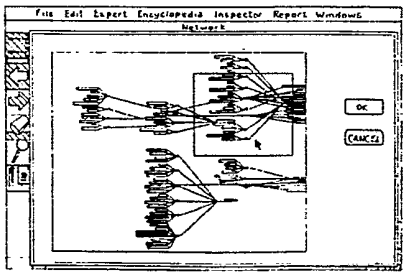
(72) **Inventeur(s) :** Alain T. Rappaport ; Albert M. Gouyet.

(73) **Titulaire(s) :**

(74) **Mandataire(s) :** Rinuy et Santarelli.

(54) Système d'affichage dynamique interactif pour base de connaissances.

(57) L'invention concerne un système de base de connaissances, comprenant un affichage graphique qui permet à l'utilisateur, à tout moment pendant l'utilisation du système, de sélectionner une des règles de connaissances affichées (ou le cas échéant l'unique règle) et d'exécuter une commande qui affiche automatiquement le ou les liens d'une certaine nature par rapport à d'autres règles de connaissances. Ces dernières sont ainsi ajoutées dynamiquement au dessin. La même fonction peut être appliquée de manière récursive dans toute direction jusqu'à épuisement de règles de connaissances à afficher. Le dessin résultant peut être plus large que l'écran d'affichage et un mécanisme de défilement d'écran ou de fenêtre graphique permet à l'utilisateur de se déplacer d'une région à l'autre sur le graphique global. Une série de fonctions permet à l'utilisateur de retirer un lien et sa règle correspondante de l'affichage, de créer un nouveau groupe de règles (un groupe étant défini comme un jeu de règles liées les unes aux autres graphiquement), et de modifier les paramètres de l'affichage en fonction de la nature de ces liens, l'état courant de chaque objet (règles et leurs éléments, liens) et l'état général du système au moment de l'affichage.



FR 2 607 284 - A1

Vente des fascicules à l'IMPRIMERIE NATIONALE, 27, rue de la Convention — 75732 PARIS CEDEX 15

La présente invention concerne la représentation graphique de la connectivité fonctionnelle existant entre les éléments de connaissances dans un système basé sur les connaissances ou le savoir ou le système dit "expert", permettant d'assister l'utilisateur dans la conception et la maintenance des applications utilisant les connaissances ou le savoir d'un expert, et elle concerne plus particulièrement un mécanisme de balayage ou de lecture d'une base de connaissances.

Le système utilisant les bases de connaissances ou système expert offre la possibilité de coder les connaissances ou le savoir de manière déclarative dans un groupe d'éléments facilement modifiables qui sont traités par un programme central dans un domaine le plus général possible. Cette programmation évoluée permet d'implémenter des programmes complexes (diagnostic, planification, conception, etc.). Cependant, les personnes à l'origine de la constitution des connaissances ou du savoir (les experts dans le domaine), doivent décomposer leurs connaissances afin de créer une base dite de connaissances. Cette décomposition ou "décompilation" se réfère au besoin de traduire les connaissances humaines en une collection d'éléments dans un format donné (parfois appelés règles, cadres, schémas, contextes, etc.).

Il existe un nombre illimité de problèmes et de leurs représentations et, par conséquent, desdites règles (éléments de connaissances constituant une tranche ou un palier de connaissances).

Les liens existant entre ces règles constituent un aspect fondamental de la base de connaissances. Chaque règle peut être considérée comme appartenant à l'un des trois jeux suivants :

(1) elle constitue, en partie ou entièrement, un argument d'une autre ou de plusieurs autres règles. En d'autres termes, son activation (définie comme étant son traitement à tout moment par le programme central) induit

l'activation d'autres règles, conduisant à la progression du programme.

(2) Elle comporte elle-même un certain nombre d'arguments, c'est-à-dire d'autres règles dont l'activation déclenchera sa propre activation par le programme central.

(3) Elle est à la fois l'argument (non nécessairement unique) de l'une ou plusieurs autres règles et elle comporte un ou plusieurs arguments qui lui sont propres.

Généralement, le fonctionnement des systèmes utilisant les bases de connaissances conduisent à une séquence de telles activations, orientée par l'information d'entrée et la nature desdites règles. On doit faire la distinction entre les liens possibles et les liens réels. Les liens possibles constituent le jeu de tous les liens qui peuvent être envisagés dans une base de connaissances donnée, tandis que les liens réels sont un sous-jeu du jeu précédemment cité qui correspond aux liens réellement utilisés pour résoudre un problème donné. La présente invention concerne l'un et l'autre des deux types de liens.

Un facteur crucial dans le développement d'une base de connaissances ou d'un jeu de règles de connaissances permettant de résoudre un problème de nature quelconque, réside dans la possibilité de restructurer la carte des liens entre les différentes tranches de connaissances ou règles de manière à obtenir une vision globale de la base de connaissances. En réalité, il est plus facile de se référer à une carte de relations qu'à une liste de règles apparemment non reliées les unes aux autres. L'utilisation des éditeurs de textes pour visualiser les listes de règles nécessite l'écriture manuelle de ces relations.

Dans certaines circonstances, il a été possible de visualiser sur un écran une liste de règles apparentées à une règle particulière. De telles listes peuvent être obtenues, mais ne peuvent être mises en relation davantage, ce qui ne permet pas d'obtenir une vue globale de la base de

connaissances et nécessite un travail manuel supplémentaire. Dans certains cas, ce travail manuel a constitué la pierre d'achoppement, tant sur le plan financier qu'humain, empêchant le développement des applications avec de tels systèmes.

La présente invention concerne une méthodologie graphique permettant d'afficher automatiquement (et ainsi mettre à portée de l'utilisateur) les différents liens existant entre les éléments de connaissances ou règles, indépendamment de la nature du problème à résoudre, de la représentation des connaissances, du format des règles et de l'équipement matériel utilisé.

La présente invention remplace par un procédé automatique le travail manuel de mise en liaison des règles entre elles. Toute règle donnée peut être affichée sur un écran et différents modes de commande permettent à l'utilisateur d'afficher l'un quelconque ou tous les liens existant entre une règle et les autres, de façon récursive si nécessaire. Les liens peuvent être d'ordres de précision variables et sont de types différents.

Les ordres de précision relèvent de la notion selon laquelle on peut relier une règle à une autre sur le graphique résultant en spécifiant simplement cette relation générale. Dans d'autres cas, la représentation graphique du lien peut relier chaque règle par les éléments très précis de ces règles qui constituent cette relation (par exemple, un élément d'une règle peut être une action modifiant la valeur d'un autre élément qui est utilisé dans une autre règle qui, à son tour, sera activée si la première l'est).

Le but de la présente invention est d'apporter un jeu d'outils graphiques permettant d'afficher, de vérifier et de maintenir les relations fonctionnelles existant entre les éléments de connaissances dans un système expert utilisant les bases de connaissances.

Le concept central de l'invention est la possibilité de sélectionner, à tout moment pendant l'utilisation du

système, une partie de l'une des règles affichées (ou de la seule règle affichée) et d'exécuter une commande qui affiche automatiquement le ou les liens d'une certaine nature existant entre la règle choisie et les autres règles. Ces dernières sont ainsi dynamiquement ajoutées au dessin affiché. La même fonction peut être appliquée de façon récursive dans toute direction jusqu'à ce qu'il n'y ait plus d'autres règles de connaissances à afficher. Le dessin résultant peut être de plus grande dimension que l'écran de visualisation et les mécanismes de fenêtre graphique ou de défilement d'écran permettent à l'utilisateur de se déplacer d'une zone du graphique total à une autre. Une série de fonctions permet de retirer un lien et sa ou ses règles correspondantes de l'affichage, afin de créer un nouveau groupe de règles (un groupe étant défini comme un jeu de règles reliées les unes aux autres graphiquement), et de modifier les paramètres de l'affichage en fonction de la nature des liens (voir annexe), l'état actuel de chaque objet (règles et leurs éléments, liens) et l'état global du système au moment de l'affichage.

Le principal obstacle consistant à établir manuellement sur le papier les liens existant entre les règles de connaissances est ainsi considérablement minimisé étant donné que cette tâche est automatisée. La reconstruction automatique permet un balayage ou une lecture dynamique à travers toute la base de connaissances, reconstruisant la structure macroscopique des connaissances nécessaires à l'utilisateur pour établir les relations adéquates au système.

L'impact de l'invention se situe dans la vitesse de développement (construction et élimination des erreurs (déboggage)) d'une base de connaissances donnée et dans la maintenance de cette dernière. De plus, étant donné que de tels dessins peuvent être imprimés, l'invention constitue un outil unique permettant la coopération de plu-

sieurs experts dans la réalisation d'une base de connaissances, ce qui constitue un facteur déterminant dans le développement des bases de connaissances ou sytème expert. En réalité, la visibilité résultante des connaissances réduit considérablement le temps d'apprentissage du système par les premiers utilisateurs et les suivants du fait que le fonctionnement interne est rendu très visible et, par conséquent, accessible pour les perfectionnements et les modifications.

Le mécanisme de balayage dynamique ("DBM"), agit comme un outil puissant pour la phase connue sous le nom de phase d'acquisition des connaissances, qui constitue un point essentiel dans les systèmes intelligents d'aujourd'hui et à venir. Il facilite l'énoncé des connaissances. En conséquence, et de manière plus significative, le mécanisme de balayage dynamique rend l'outil utilisant le système de base de connaissances accessible à des utilisateurs non experts dans le domaine de l'intelligence artificielle.

Par conséquent, un objet de la présente invention est de fournir une interface graphique interactive pour un système de base de connaissances contrôlée par ordinateur.

Un autre objet de la présente invention consiste à offrir dans un système de base de connaissances commandé par un ordinateur, une structure de connaissances visible qui est automatiquement construite par le système.

Un autre objet encore de la présente invention est de fournir un système de connaissances commandé par ordinateur qui permette à l'utilisateur de balayer une base de connaissances pour visualiser et effectuer si nécessaire des modifications aux règles de connaissances et aux liens existant entre elles.

Les objectifs ci-dessus et d'autres encore, les caractéristiques et avantages de la présente invention, ressortiront davantage de la description ci-après de certains modes de réalisation préférés de l'invention conjointement

avec les dessins annexés à la présente demande de manière non limitative.

Sur ces dessins

- la figure 1 représente un diagramme d'un système de base de connaissances mises sur ordinateur selon l'invention,
- la figure 2 est un diagramme de l'architecture générale d'une implémentation selon l'invention,
- les figures 3A à 8 illustrent des affichages sur écran destinés à expliquer le fonctionnement de l'invention,
- la figure 9 est un organigramme illustrant le fonctionnement de l'unité centrale de traitement du système de l'invention pendant une session interactive de travail d'un utilisateur, avec le mécanisme de balayage selon l'invention, et
- les figures 10A à 10J représentent des parties d'affichage sur écran destinées à expliquer le fonctionnement de l'invention.

On a récemment mis au point des systèmes informatiques conçus de manière à exécuter des programmes imitant le raisonnement humain. Ces systèmes reconnaissent des schémas logiques qui sont similaires à ceux reconnus par les humains, et établissent des raisonnements conduisant à des performances proches du travail d'un expert. Pour saisir le raisonnement d'un expert, un procédé connu sous le nom de "modélisation des connaissances" est utilisé. Ce procédé implique d'avoir un module de connaissances, à savoir des structures déclaratives décrivant les connaissances d'un expert (ce module maintient le savoir de l'expert) et des moteurs d'inférence (qui imitent le raisonnement logique de l'expert). Le système selon l'invention utilise les règles "si... alors... faire ..." pour représenter les connaissances d'un expert dans un domaine donné.

Le système selon l'art antérieur faisait appel

à des ordinateurs centraux dits "gros systèmes", tandis que la présente invention est destinée à être mise en application sur un micro-ordinateur ou ordinateur personnel comme représenté sur la figure 1.

En se référant plus particulièrement à la figure 1, une unité centrale de traitement 10 qui peut être constituée par un ordinateur personnel comportant une mémoire interne 12 et une mémoire externe 14 traite des données et autres instructions qui sont entrées interactivement par l'intermédiaire d'un clavier 16 lequel peut intégrer un dispositif 18 du type souris. Divers affichages graphiques illustrant l'utilisation interactive de l'ordinateur par l'utilisateur sont assurés par un moniteur 20. Une imprimante 21 peut également être connectée à l'unité centrale de traitement 10 pour une sortie sur imprimante, en réponse aux instructions données par l'intermédiaire du clavier 16 ou de la souris 18, des éléments choisis des relations existant entre les règles de la base de connaissances, comme expliqué ci-après.

La figure 2 représente plus particulièrement l'architecture générale de l'invention telle que mise en oeuvre. L'unité centrale de traitement 10 comporte un système d'exploitation 22 et un programme d'application expert constitué d'un noyau de système expert 24, d'un compilateur de règles 28, d'un moteur d'inférence 26, d'un dispositif de balayage dynamique 30, et des moyens de réglage 32. L'information provenant de l'utilisateur est fournie au système au moyen des dispositifs périphériques d'interface 16 et 18.

Les figures 3A, 3B et 4 décrivent plus particulièrement le fond du système de connaissances selon l'invention. Les figures 3A, 3B et 4 illustrent des affichages qui apparaissent sur le moniteur 20 pendant l'utilisation du système.

La fonction de l'éditeur de règles du système est illustrée sur la figure 3A. Cette dernière montre les

règles sur lesquelles le système de connaissances est basé et l'utilisateur peut choisir lesquelles des règles sont à utiliser pour exécuter l'opération. La fonction interactive du système est classique en ce que un nombre multiple de fenêtres apparaît simultanément sur l'écran d'affichage du moniteur 20 et l'opérateur manipule une icône 34 apparaissant sur l'écran au moyen de la souris 18. Lorsque l'icône est sur une fonction particulière que l'utilisateur a choisie, l'utilisateur peut actionner un bouton 36 sur la souris 18 ("cliquer") de sorte que l'unité centrale de traitement 10 exécute la fonction affichée. L'utilisateur peut amener la souris 34 sur le recoin 37 et en cliquant sur le bouton 36 peut provoquer le basculement des pages-écran vers l'avant ou vers l'arrière. Les règles sont indexées alphabétiquement par hypothèse. Les "boutons" représentés à droite sur la figure 3B permettent un déplacement rapide sur la liste. La sélection d'une opération basée sur une règle est réalisée en cliquant sur le bouton 36 tandis que le curseur 34 se trouve sur l'une des icônes à la partie inférieure de l'éditeur de règles. Sur l'écran représenté sur la figure 3A, on est en présence des opérations suivantes (de gauche à droite) : (1) créer une nouvelle règle, (2) modifier la règle actuelle, (3) créer une nouvelle règle en recopiant la règle actuelle, (4) supprimer la règle actuelle, et (5) enregistrer la base de connaissances.

Une fois que l'utilisateur a choisi une opération, une nouvelle interface graphique (voir figure 4) apparaît, laquelle constitue l'interface de développement principal. L'utilisateur peut entrer et modifier toutes les règles à partir de cette interface. La seule nécessité est de remplir les conditions, l'hypothèse, et les actions par le biais des menus déroulants représentés sur la figure 4, en utilisant le curseur 34 et le bouton 36 de la souris. L'interface de développement principal de base comprend les fentes ou "cases" IF (SI), THEN (ALORS), AND DO (ET FAIRE), et les

fentes CONTEXT (CONTEXTE). L'utilisateur peut provoquer le déroulement d'un menu approprié en cliquant sur le bouton 36 tandis que le curseur 34 se trouve sur une fente ou case particulière. Le menu se trouvant sur le côté gauche de l'écran montre tous les tests disponibles pour former les conditions de règles suivantes :

Test de conditions

CANCEL (SUPPRIMER)	Ne rien faire
CLEAR (EFFACER)	Effacer la case test
YES and NO (OUI et NON)	Tester les booléens
$>, <, \geq, \leq, =, \neq$	Comparer l'équation numérique à la constante
NAME (NOMMER)	Donner synonyme global
IS and IS $\neq$ (EST et EST $\neq$)	Comparer des données de valeurs multiples à une constante
RESET (MISE A ZERO)	Mettre les données à la valeur "inconnu"
EQUAL and UNEQUAL (EGAL et NON-EGAL)	Comparer deux données à valeurs multiples

Le menu se trouvant sur la partie droite de l'écran représente toutes les actions que l'on peut exécuter :

Opérations

CANCEL (SUPPRIMER)	Ne rien faire
CLEAR (EFFACER)	Effacer la case d'opération
DO (FAIRE)	Calculer l'équation numérique et assigner la valeur à la donnée
LET (LAISSER)	Attribuer une constante à une donnée à valeurs multiples
RESET (MISE A ZERO)	Mettre la donnée à une valeur "inconnu"
SHOW (MONTRER)	Afficher un fichier-texte dans la fenêtre appropriée ou bien dérouler une image pour remplir l'écran
LOADKB (CHARGER BASE DE CONNAISSANCES)	Annexer à la base de connaissances actuelle (courante) une autre base de connaissances

EXECUTE (EXECUTER)	Exécuter une fonction externe
RETRIEVE (EXTRAIRE)	Charger les données provenant d'un fichier externe au format SYLK.

Le menu représenté dans la partie centrale de l'affichage représenté sur la figure 5, qui chevauche les menus précédemment décrits, est destiné à entrer des arguments et des équations. Les fonctions COPIER DONNEES et COPIER HYPOTHESES sont disponibles à partir de ce menu.

Une fois que la règle a été entrée, le bouton "OK" s'assombrit si la règle a été structurée de façon correcte. Si cela n'est pas le cas, en cliquant sur le bouton CHECK (VERIFIER), on provoque l'illumination de la fente ou case annexe. Si un élément d'information a été utilisé de façon incorrecte, mais la règle correctement structurée, un message explicatif apparaîtra après que l'on aura cliqué sur le bouton "OK".

La description ci-dessus est destinée à montrer le contexte de la présente invention. Un problème quelconque peut être présenté de trois manières au système selon la présente invention :

(1) la manière dite "orientée données", dans laquelle seulement quelques données sont connues concernant le problème. L'extraction de ces données initiales peut être automatisée en utilisant une formule d'entrée. L'inférence commencera par un chaînage avant ou déduction.

(2) La manière dite "orientée hypothèses", dans laquelle l'utilisateur désire la confirmation ou l'argumentation sur une ou plusieurs hypothèses. L'inférence commencera alors par un chaînage arrière ou induction.

(3) Une manière mixte combinant la manière "orientée données" et la manière "orientée hypothèses", dans laquelle l'utilisateur connaît un certain nombre de données et désire également vérifier une hypothèse.

Le réseau ou tissu de relations existant entre les divers éléments de l'information connus et les règles

qui les relient sont dénommés "réseau" et cette interrelation est représentée graphiquement sur l'écran du moniteur 20 par une déclaration de données ou de règles et par des lignes tracées les reliant, comme illustré sur la figure 6. Cet écran peut être amené à apparaître par manipulation du curseur sur l'étiquette "Inspector" sur l'écran et par cliquage du bouton 36. Comme expliqué plus en détail ci-après, ce qui constitue une caractéristique principale de l'invention réside dans le fait que le réseau permet à l'utilisateur de balayer une base de connaissances de manière graphique à un degré visuel très élevé et, par conséquent, à voir clairement les relations existant entre les éléments de la base de connaissances. L'outil de balayage permet à la fois une navigation déductive et évocative de la base de connaissances. Un système intelligent basé sur des règles doit raisonner à partir de faits (déductions ou chaînages avant) aussi bien qu'à partir d'objectifs (chaînages arrière ou inductions). La présente invention met en oeuvre ces deux types de raisonnement.

Le procédé d'inférence basé sur des règles de connaissances utilisant la présente invention est amorcé par suggestion d'une hypothèse, la fourniture spontanée d'une donnée, ou une combinaison des deux. Si une donnée quelconque est fournie spontanément, le système peut alors chaîner en avant (déduire), en essayant d'établir autant d'hypothèses que possible. Ce chaînage avant peut être commandé par l'utilisateur en utilisant la fonction "Strategy" (figure 4). Si l'utilisateur ne suggère qu'une hypothèse, le système commence par chaîner en arrière (chaînage arrière ou induction) pour tenter de trouver suffisamment d'informations permettant de vérifier l'hypothèse. Cette information peut conduire le système à d'autres hypothèses à établir, lesquelles peuvent conduire à leur tour à d'autres hypothèses encore, et ainsi de suite.

Afin d'avoir une vue globale de ces interrelations, l'utilisateur positionne le curseur 34 sur la partie "Inspector" de l'écran d'affichage et clique sur le bouton 36 et choisit alors la fonction de vision globale. Il en résulte l'affichage représenté sur la figure 8. Pour obtenir une vue plus détaillée, l'utilisateur positionne le curseur 34 sur l'étiquette "Inspector", clique sur le bouton 36 et il en résulte l'affichage représenté sur la figure 6. En utilisant les icônes d'actionnement représentées sur la gauche de l'écran sur les figures 6 et 10A à 10J, l'utilisateur peut se déplacer sur des ilôts de connaissances de la base de données. Les fonctions appelées par ces icônes autorisent une investigation dans des régions du graphique de connaissances qui ne présentent un intérêt que dans le contexte du moment.

En se référant maintenant à la figure 6, l'icône 38 permet l'affichage d'un ilôt de connaissances unique ou multiple .L'icône 40 affiche les données qui convergent vers l'hypothèse courante. L'icône 42 affiche les hypothèses vers lesquelles cette donnée est orientée. L'icône 44 efface le lien courant. L'icône 46 focalise l'attention du système sur cette donnée ou hypothèse et l'icône 48 vérifie l'opportunité de cette donnée ou hypothèse. Le choix de ces icônes n'affecte que l'affichage et non pas le graphique de connaissances réelles de l'ensemble des règles. A partir de cette fenêtre, toutes les règles peuvent être vues par l'utilisateur et tous les liens existant entre ces règles peuvent être vérifiés. Ceci permet à la dynamique du procédé d'être contrôlée,en constituant ainsi une aide au déboggage (suppression des erreurs) de la base de règles.

En se reportant plus particulièrement à la figure 9, un organigramme des actions réalisées par l'unité centrale de traitement dans la mise en oeuvre de la navigation ou du balayage des connaissances est représenté. En

premier lieu, l'unité centrale de traitement 10 guette un cliquage du bouton 36 à l'étape 50. Il détecte l'endroit où le curseur 34 a été positionné dans la zone graphique et détermine à l'étape 52 si le bouton 36 a été cliqué une nouvelle fois tandis que le curseur 34 a été positionné par-dessus une icône. Si la réponse est oui, l'unité centrale de traitement 10 guette un second cliquage à l'étape 54. Si ce cliquage n'a pas eu lieu, l'unité de traitement centrale 10 poursuit jusqu'à l'étape 56 pour choisir une nouvelle icone pour affichage. Ensuite, l'unité centrale de traitement retourne à l'étape 50 pour observer un second cliquage sur le bouton 36 de la souris.

En supposant qu'à l'étape 54 le bouton 36 de la souris a été cliqué une nouvelle fois, l'unité centrale de traitement poursuit jusqu'à l'étape 58 où il retourne le nombre de l'icône 38-48 sur laquelle le curseur 34 a été positionné. Après un double cliquage du bouton 36, l'unité centrale exécute les fonctions dictées par l'icône et met à jour le réseau à l'étape 62, après quoi l'unité centrale de traitement retourne de nouveau à l'étape 50.

En supposant qu'à l'étape 52, il n'y ait pas eu de cliquage sur les icônes, l'unité centrale de traitement trouvera le descripteur de réseau à l'étape 64 et déterminera ensuite à l'étape 66 si l'opération appelée par ce descripteur est valide. Si elle ne l'est pas, l'unité centrale 10 retourne à l'étape 50. Si la réponse à l'étape 66 est affirmative, l'unité centrale 10 détermine à l'étape 68 si l'affichage doit changer. Si la réponse est non, l'unité centrale de traitement retourne à l'étape 50 et si la réponse est oui, l'unité centrale calcule les nouvelles règles de connaissancesou retire les anciennes à l'étape 70. A l'étape 72, l'unité centrale 10 trouve de nouvelles régions vides dans l'affichage ou en élimine certaines. A l'étape 74, l'unité centrale 10 remplit les zones par des tranches ou règles de connaissances. Le réseau

est alors mis à jour à l'étape 62 et l'unité centrale 10 retourne à l'étape 50.

Les figures 10A-10H illustrent les étapes par lesquelles le réseau de connaissances peut être rempli en utilisant le balayage à la fois dans les modes de chaînage arrière (figure 10B) et chaînage avant (figure 10C). (Les termes "chaînage arrière" et "chaînage avant" (induction et déduction) se réfèrent bien entendu au processus logique et non pas à une direction quelconque de l'affichage).

Les figures 10A, 10B et 10C illustrent la façon dont l'utilisateur a exécuté à la fois des chaînages avant et arrière en positionnant le curseur sur la case "Yes valve-problem" (Oui problème de robinet) sur l'écran d'affichage du moniteur 20 après (par des séquences séparées) avoir choisi les icônes 40 et 42. De manière similaire, les figures 10B et 10C illustrent comment l'utilisateur peut chaîner en avant en positionnant le curseur 34 sur la partie "current task is REF" (la tâche courante est REF) de l'affichage après avoir d'abord choisi l'icône 40.

On trouvera sur les figures 11A-11J et 12A-12C le code source relatif à la caractéristique de balayage de connaissances selon l'invention, écrit en langage de programmation Pascal. Il convient de noter que ledit programme renvoie à certains sous-programmes qui ne sont pas inclus dans l'annexe; cependant ces sous-programmes sont bien connus de l'homme de l'art et leurs fonction et exécution, qui sont listées sur les figures 11A-11J et 12A-12C, sont évidentes à la lecture de leurs titres.

Il va de soi que la présente invention n'est pas limitée aux formes de réalisation décrites et représentées et est susceptible de diverses variantes sans sortir de son cadre.

REVENDICATIONS

1. Système à base de connaissances, comprenant un ordinateur programmé, des moyens de mémorisation (14) pour emmagasiner une base de connaissances, cette base de connaissances comprenant à la fois des données et des règles reliant les données entre elles pour créer des hypothèses, ledit système étant caractérisé par un dispositif d'affichage (20) permettant de visualiser les inter-relations existant entre les données et les hypothèses telles qu'elles sont emmagasinées dans le dispositif de mémoire, un dispositif (16) permettant d'entrer les informations fournies par l'utilisateur et permettant à l'utilisateur du système d'entrer des données, des règles et de modifier ces données ou ces règles couramment (actuellement) emmagasinées dans le dispositif de mémorisation (12), et des moyens permettant l'affichage interactif des inter-relations existant entre les éléments de la base de connaissances, de manière visuelle et interactive avec les modifications réalisées par l'utilisateur.

2. Système à base de connaissances selon la revendication 1, caractérisé en ce que le dispositif pour afficher visuellement de manière interactive les relations existant entre les éléments de la base de connaissances comprend un moyen (16, 36) commandé par l'utilisateur pour choisir la quantité de relations ou liens du réseau de relations devant être affichée simultanément sur le dispositif d'affichage visuel (20).

3. Système à base de connaissances selon la revendication 1 ou 2, caractérisé en ce qu'il comprend de plus un dispositif d'impression (21), le dispositif d'affichage interactif des inter-relations des éléments de la base de connaissances, comprenant un dispositif commandé par l'utilisateur pour déclencher l'impression par le dispositif d'impression d'un diagramme représentant les inter-relations de la base de connaissances choisies par l'utilisateur.

2607284

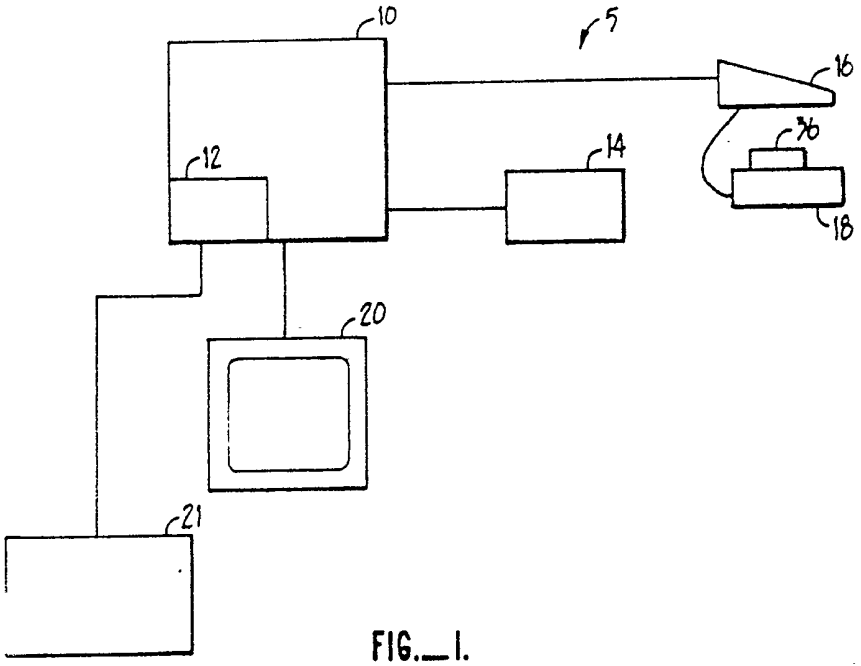


FIG._1.

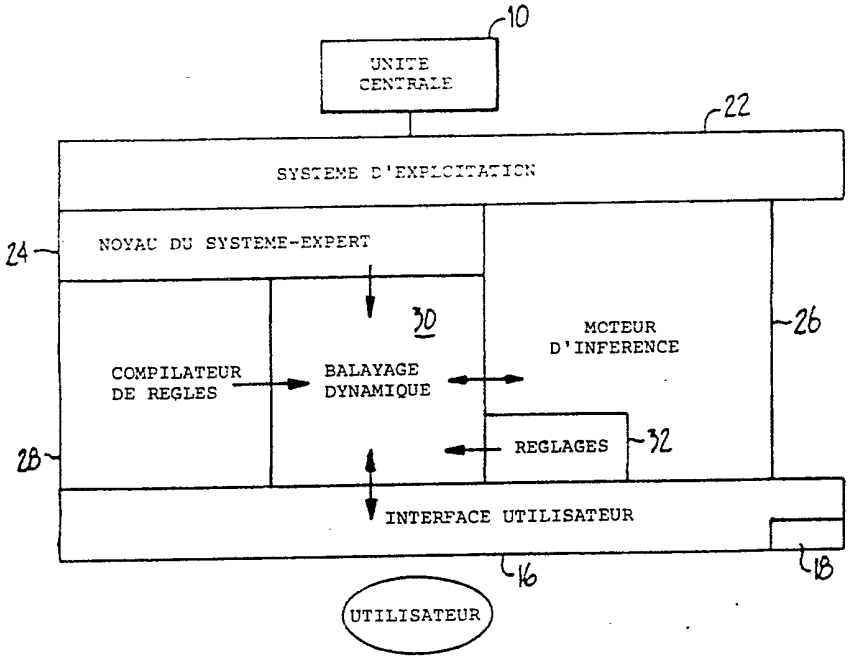


FIG._2.

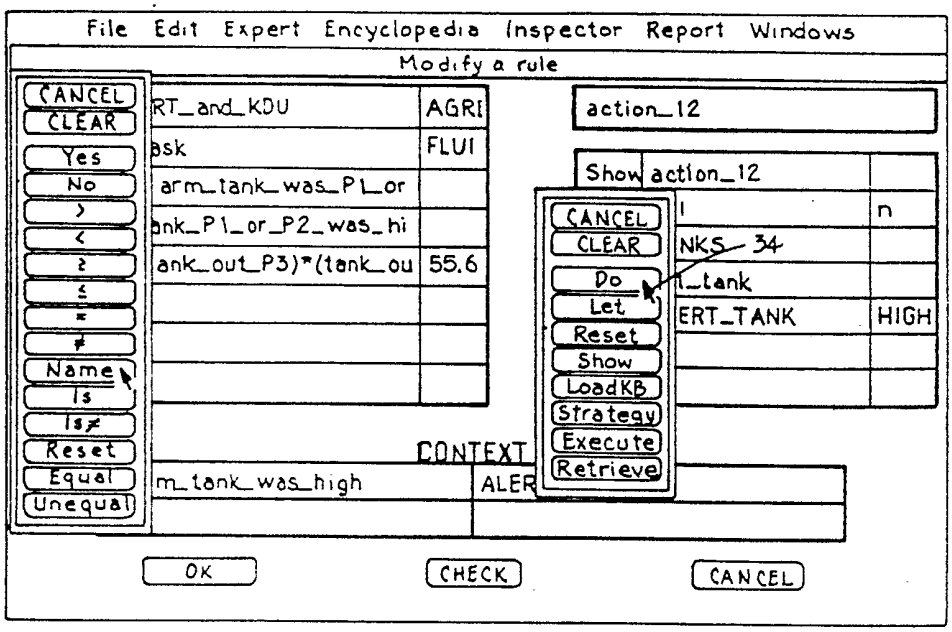


FIG._4.

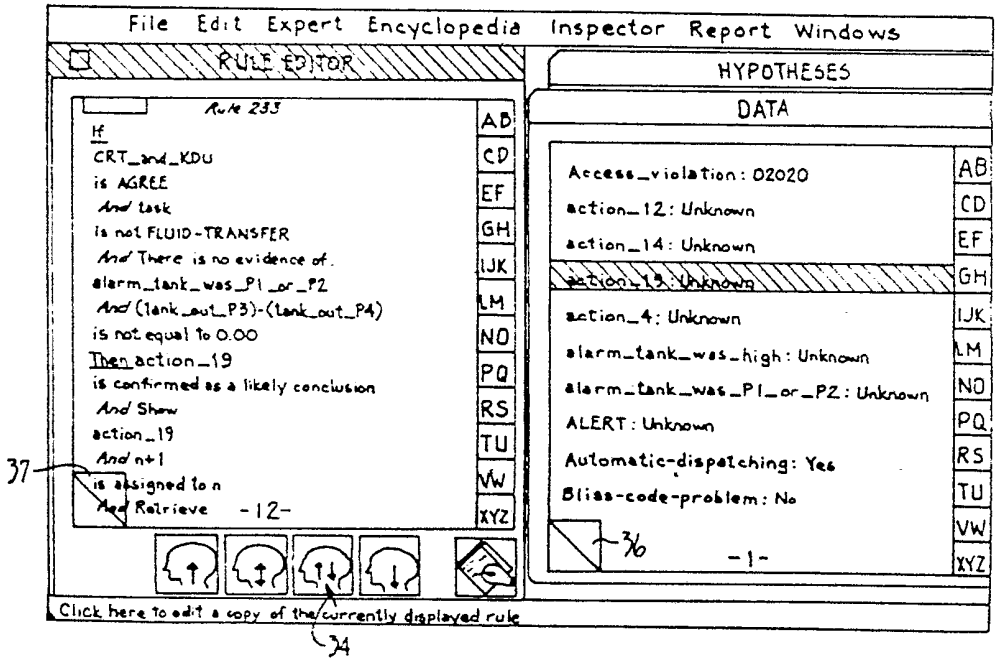


FIG._3A.

FIG._3B.

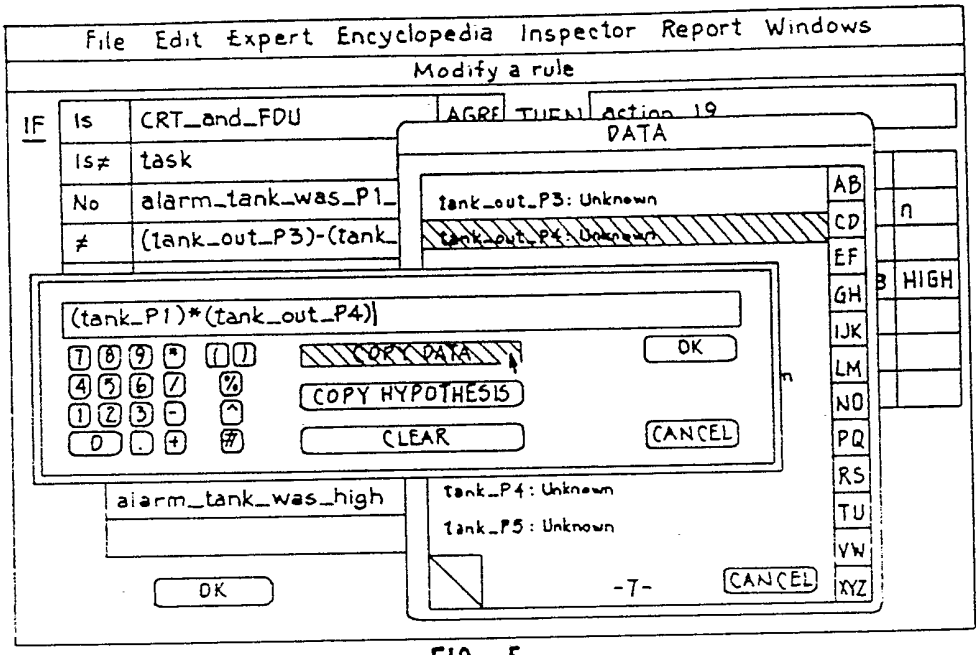

File Edit Expert Encyclopedia Inspector Report Windows
Modify a rule
IF
Is
CRT_and_FDU
Is≠
task
No
alarm_tank_was_P1_
≠
(tank_out_P3)-(tank_
DATA
tank_out_P3: Unknown
tank_out_P4: Unknown
(tank_P1)*(tank_out_P4)
COPY DATA
OK
COPY HYPOTHESIS
CLEAR
CANCEL
alarm_tank_was_high
tank_P4: Unknown
tank_P5: Unknown
OK
-7-
CANCEL
AB
CD
EF
GH
IJK
LM
NO
PQ
RS
TU
VW
XYZ
HIGH

FIG._5.

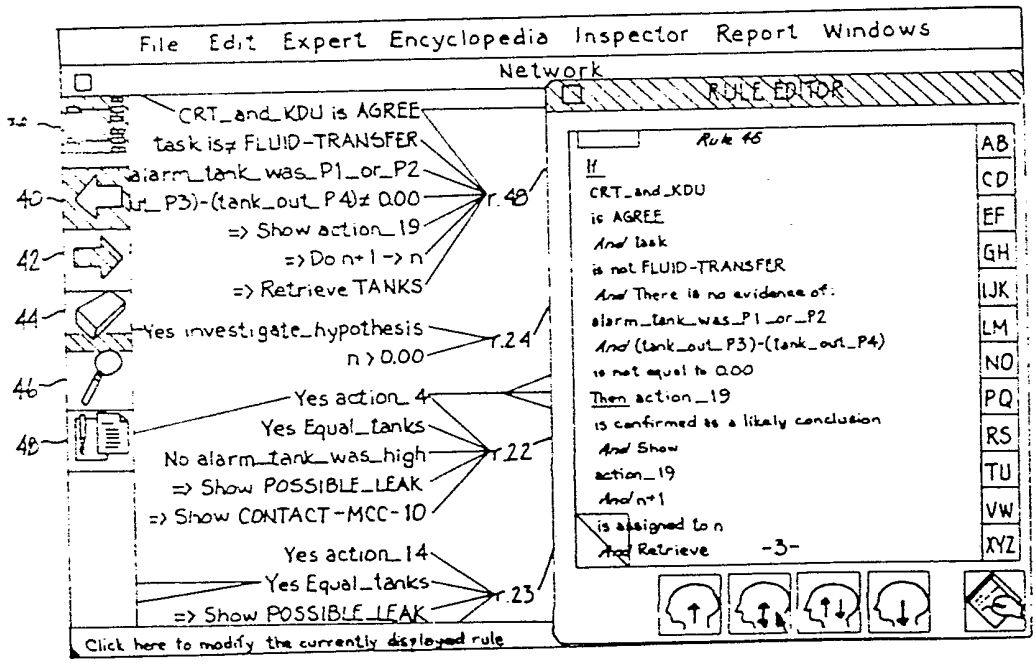

File Edit Expert Encyclopedia Inspector Report Windows
Network
RULE EDITOR
CRT_and_KDU is AGREE
task is≠ FLUID-TRANSFER
alarm_tank_was_P1_or_P2
(tank_out_P3)-(tank_out_P4)≠ 0.00
=> Show action_19
=> Do n+1 -> n
=> Retrieve TANKS
r.48
Yes investigate_hypothesis
n > 0.00
r.24
Yes action_4
Yes Equal_tanks
No alarm_tank_was_high
=> Show POSSIBLE_LEAK
=> Show CONTACT-MCC-10
r.22
Yes action_14
Yes Equal_tanks
=> Show POSSIBLE_LEAK
r.23
Rule 48
If
CRT_and_KDU
is AGREE
And task
is not FLUID-TRANSFER
And There is no evidence of:
alarm_tank_was_P1_or_P2
And (tank_out_P3)-(tank_out_P4)
is not equal to 0.00
Then action_19
is confirmed as a likely conclusion
And Show
action_19
And n+1
is assigned to n
And Retrieve
-3-
Click here to modify the currently displayed rule
AB
CD
EF
GH
IJK
LM
NO
PQ
RS
TU
VW
XYZ

FIG._6.

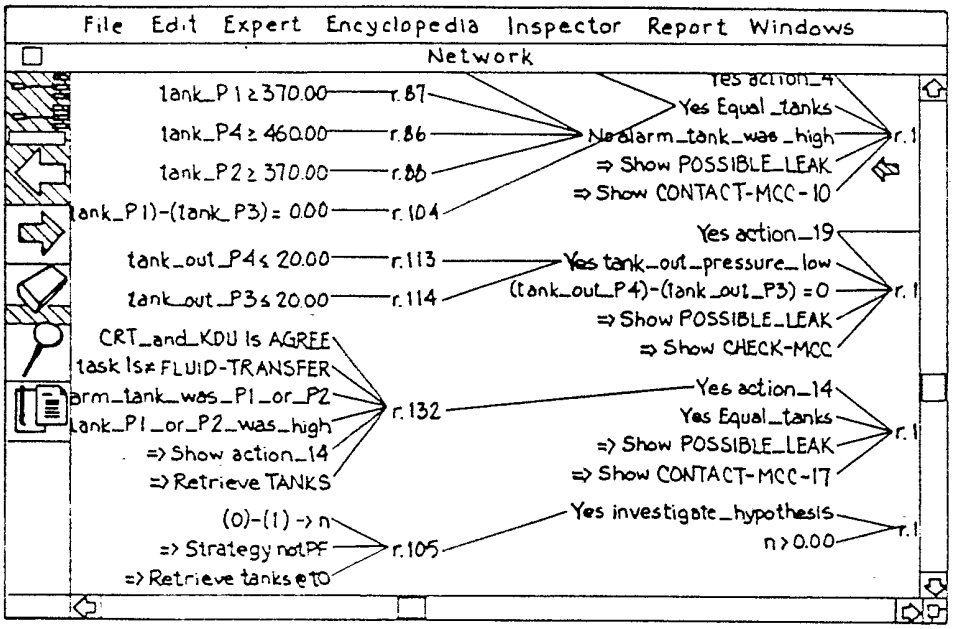

File Edit Expert Encyclopedia Inspector Report Windows
Network
tank_P1 ≥ 370.00
tank_P4 ≥ 460.00
tank_P2 ≥ 370.00
tank_P1)-(tank_P3) = 0.00
r.87
r.86
r.88
r.104
Yes action_4
Yes Equal_tanks
No alarm_tank_was_high
⇒ Show POSSIBLE_LEAK
⇒ Show CONTACT-MCC-10
r.1
tank_out_P4 ≤ 20.00
tank_out_P3 ≤ 20.00
r.113
r.114
Yes tank_out_pressure_low
(tank_out_P4)-(tank_out_P3) = 0
Yes action_19
⇒ Show POSSIBLE_LEAK
⇒ Show CHECK-MCC
r.1
CRT_and_KDU Is AGREE
task Is≠ FLUID-TRANSFER
arm_tank_was_P1_or_P2
tank_P1_or_P2_was_high
⇒ Show action_14
⇒ Retrieve TANKS
r.132
Yes action_14
Yes Equal_tanks
⇒ Show POSSIBLE_LEAK
⇒ Show CONTACT-MCC-17
r.1
(0)-(1) -> n
⇒ Strategy notPF
⇒ Retrieve tanks e TO
r.105
Yes investigate_hypothesis
n > 0.00
r.1

FIG._7.

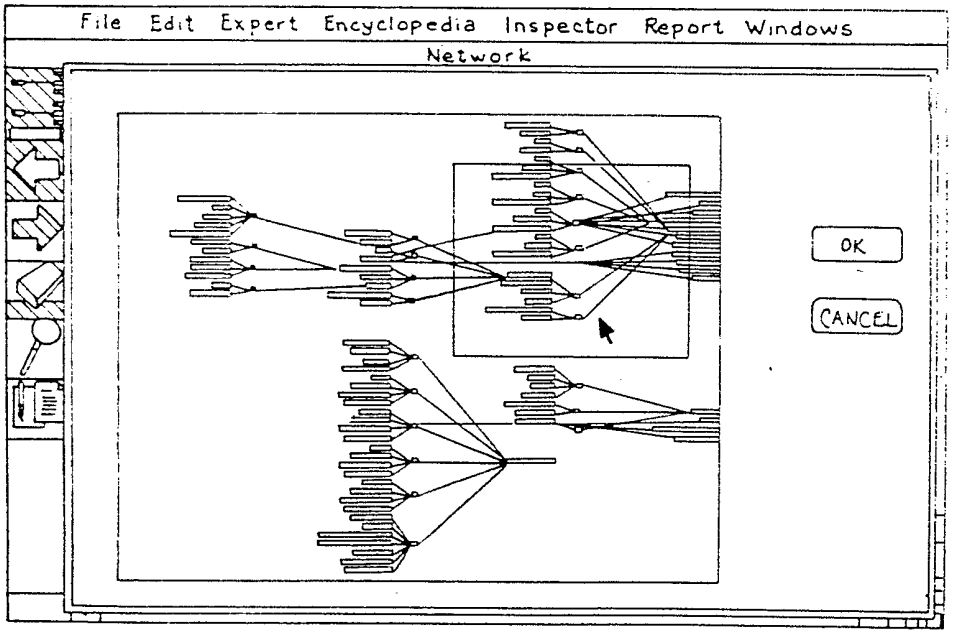

File Edit Expert Encyclopedia Inspector Report Windows
Network
OK
CANCEL

FIG._8.

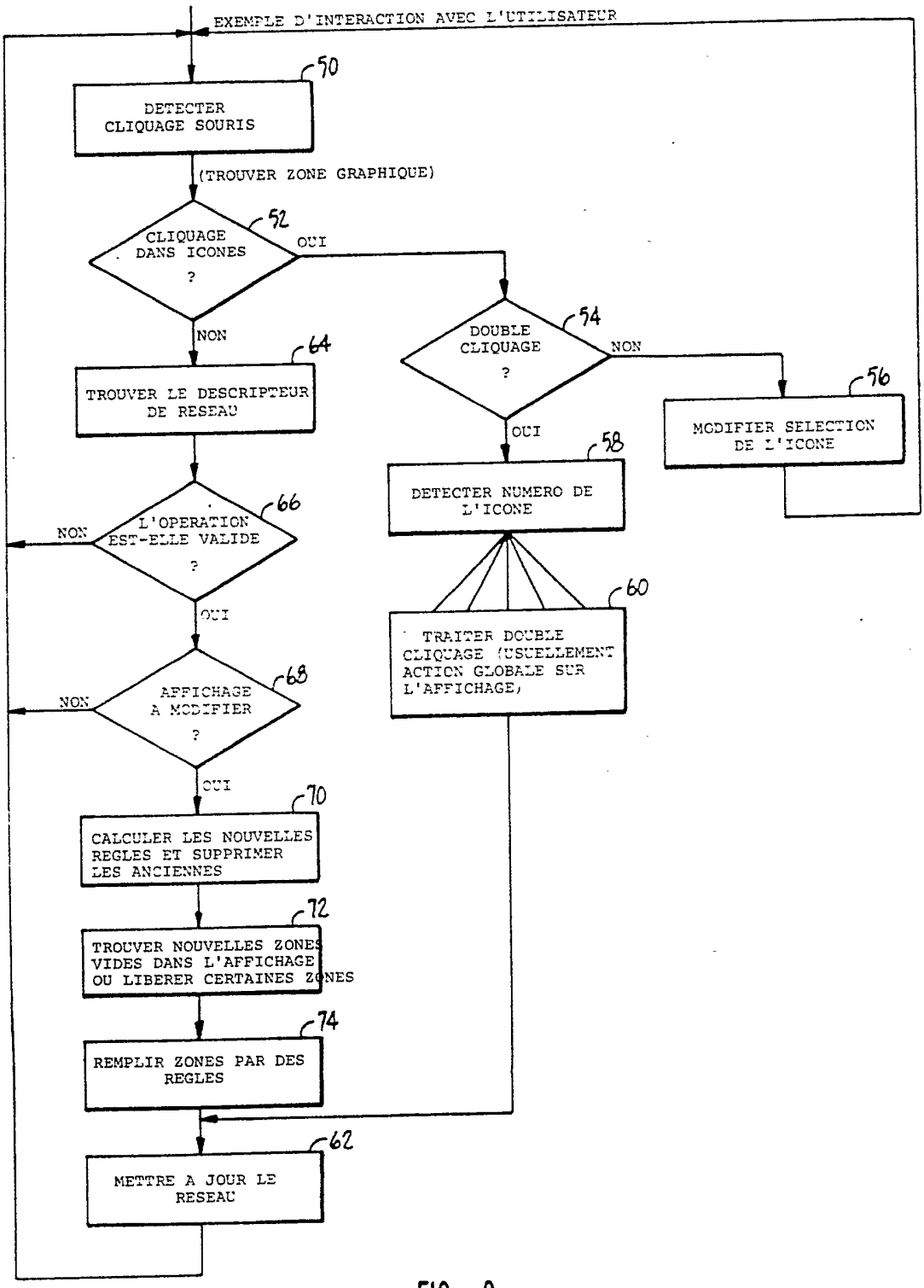

EXEMPLE D'INTERACTION AVEC L'UTILISATEUR
50
DETECTER CLIQUAGE SOURIS
(TROUVER ZONE GRAPHIQUE)
52
CLIQUAGE DANS ICONES ?
OUI
NON
54
DOUBLE CLIQUAGE ?
NON
OUI
56
MODIFIER SELECTION DE L'ICONE
58
DETECTER NUMERO DE L'ICONE
60
TRAITER DOUBLE CLIQUAGE (USUELLEMENT ACTION GLOBALE SUR L'AFFICHAGE)
64
TROUVER LE DESCRIPTEUR DE RESEAU
66
L'OPERATION EST-ELLE VALIDE ?
NON
OUI
68
AFFICHAGE A MODIFIER ?
NON
OUI
70
CALCULER LES NOUVELLES REGLES ET SUPPRIMER LES ANCIENNES
72
TROUVER NOUVELLES ZONES VIDES DANS L'AFFICHAGE OU LIBERER CERTAINES ZONES
74
REMPLIR ZONES PAR DES REGLES
62
METTRE A JOUR LE RESEAU

FIG._9.

FIG._10A.

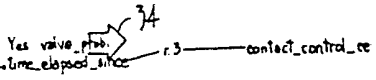


FIG._10B.

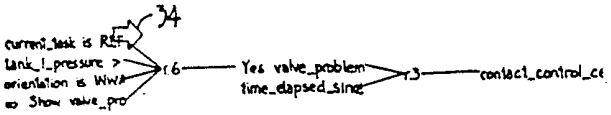


FIG._10C.

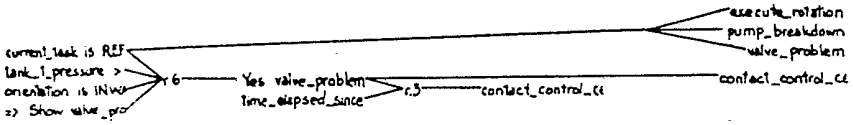


FIG._10D.

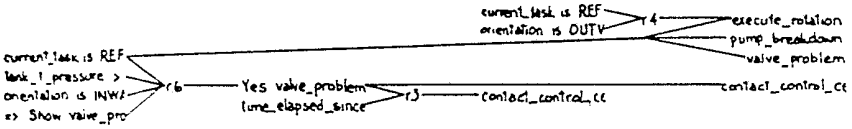


FIG._10E.

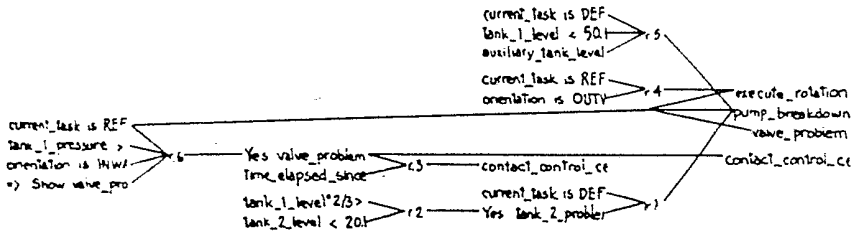


FIG._10F.

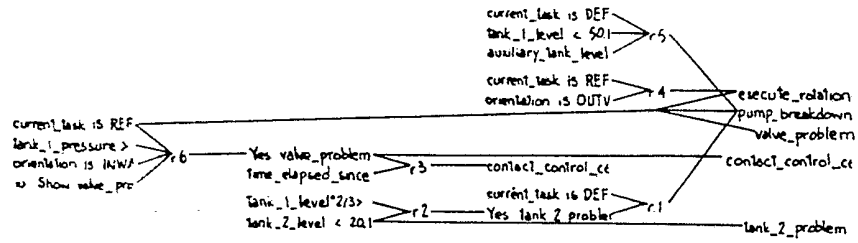


FIG._10G.

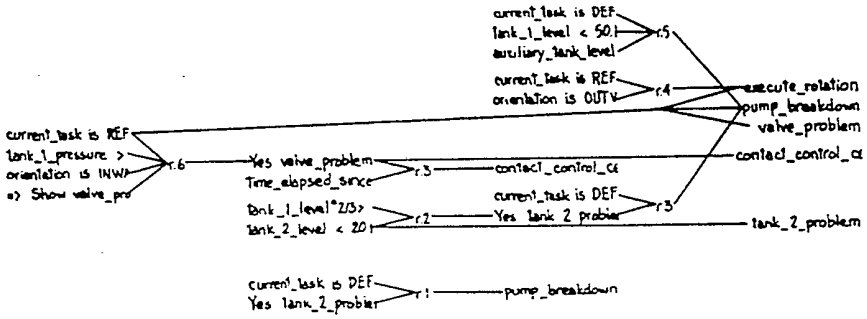


FIG._10H.

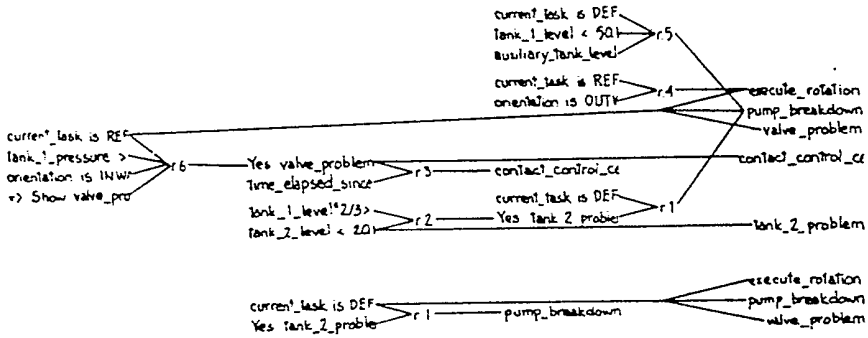


FIG._10I.

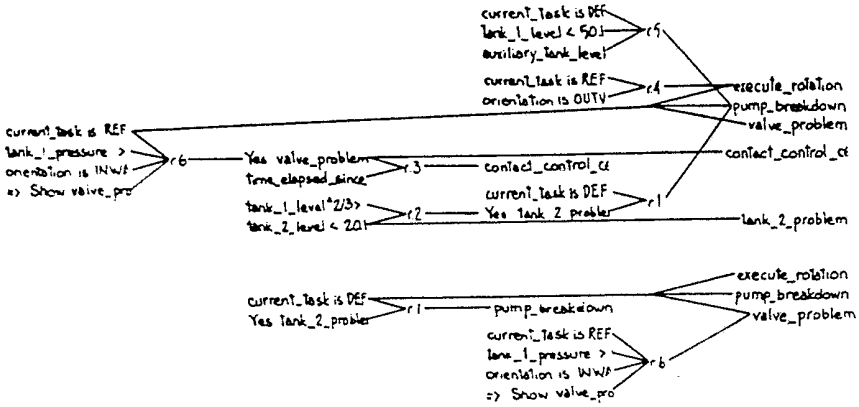
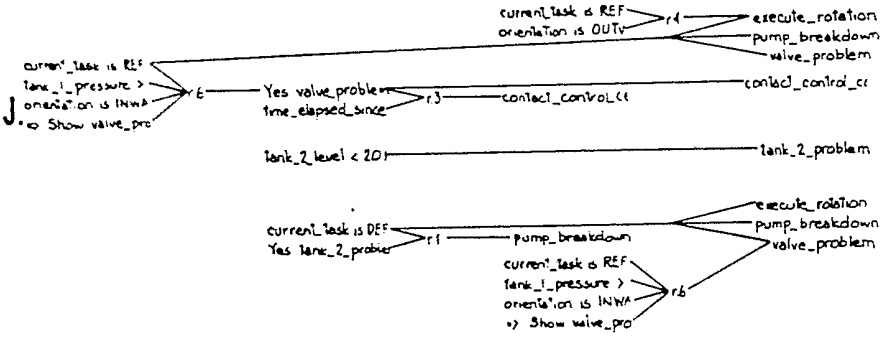


FIG._10J.



```
(* © Copyright NEURON DATA, July 1985 By AMG                              *)

(************************************************************************)
(*                              NetEvent                                *)
(*                                                                      *)
(* All the event handling for the network                               *)
(************************************************************************)
{$S NETPROCESS}

(************************************************************************)
(*                      High Level Routines for NetEvent                *)
(************************************************************************)

(*
 * NetSelChoose.
 *  This routine proccesses a double click in the magnifying glass
 *  icon.
 *)
PROCEDURE NetSelChoose;
VAR
  theDialogPtr: DialogPtr;
  ItemHit     : INTEGER;
BEGIN
  theDialogPtr:= GetNewDialog( NETCHOOSESEL, NIL, POINTER( -1) );
  SetCursor( Arrow );
  REPEAT
    ModalDialog( NIL, ItemHit );
  UNTIL( ItemHit IN [1..3] );
  DisposDialog( theDialogPtr );
  CASE ItemHit OF
  2:
    (* Display a new group of chunks starting with the selected data *)
    NetEvokeInit( AtomHdl(theSelSign), (NetGrpCHosen = NIGRP1) );
  3:
    (* Display a new group of chunks starting with the selected hypothesis *)
    NetAtomInit( AtomHdl(theSelHyp), (NetGrpCHosen = NIGRP1), FALSE );
  END;
END;  { of NetSelChoose }

(*
 * NetDblSelect.
 *  Processes double clicks in the Tools Icons.
 *)
PROCEDURE NetDblSelect;
BEGIN
  IF( (theSelHyp <> NIL) AND (theSelSign <> NIL) ) THEN
    NetSelChoose
  ELSE
    IF( theSelHyp <> NIL ) THEN
      NetAtomInit( AtomHdl(theSelHyp), (NetGrpCHosen = NIGRP1), FALSE )
    ELSE
      IF( theSelSign <> NIL ) THEN
        NetEvokeInit( AtomHdl(theSelSign), (NetGrpCHosen = NIGRP1) );
END;  { of NetDblSelect }

(*
 * FindGZone.
```

FIG. 11A

```
 *  Find in which GZone the click has occured.
 *  returns FALSE if none.
 *  In this examples the Zones are equivalent to fixed columns.
 *)
FUNCTION FindGZone(
             horiz    : INTEGER;
         VAR Col      : INTEGER ): BOOLEAN;
BEGIN
  (* special case for GZone 0 *)
  IF( (horiz < NetTabs[0].HighBound) AND
    (horiz >NetTabs[0].LowBound) ) THEN
    BEGIN
      col:= 0;
      FindGZone:= TRUE;
      EXIT( FindGZone );
    END;
  horiz:= HWinToNet( horiz ); (* change point to net coordinates *)
  col:= 2;
  FindGZone:= FALSE;
  WHILE( (NetTabs[col].LowBound <= horiz) AND ( col <= NETGZone ) ) DO
    col:= col + 1;
  col:= col - 1;
  IF( col = NETGZone + 1) THEN
    EXIT( FindGZone );
  IF( NetTabs[col].HighBound >= horiz ) THEN
    FindGZone:= TRUE; (* the right GZone has been found *)
END;  { of FindGZone }

(*
 *  FindAtom: finds in which atoms the click has occured.
 *  On Entry, pos is the vertical position in the display.
 *      col is the horizontal GZone number.
 *  On Exit, FimdAtom is a descriptor of the network element on which
 *    the user has clicked.
 *)
FUNCTION FindAtom(
            pos    : INTEGER;
            col    : INTEGER ): NetStHdl;
VAR
  RelPos   : INTEGER;
  theHdl   : NetStHdl;
  theGrp   : NetGrpPtr;
BEGIN
  FindAtom:= NetStHdl( NIL );
  theGrp:= FirstNetGrp;
  WHILE( NextNetGrp( theGrp ) <> NIL ) DO
    BEGIN
      theHdl:= FirstNetHdl( theGrp );
      WHILE( theHdl <> NIL ) DO
        BEGIN
          IF( PosInHdl( pos, theHdl, theGrp ) = TRUE ) THEN
            BEGIN
              FindAtom:= theHdl;
              EXIT;
            END;
          theHdl = NextNetHdl( theHdl );
        END;
```

FIG. 11B

```
      theGrp:= NextNetGrp( theGrp );
    END;
END;  { of FindAtom }

(*
 * GetIconNumber.
 *  If the mouse was clicked on one of the tool icons, return its number.
 *  In this example, the icons are in column 0 at the left of the screen.
 *)
FUNCTION GetIconNumber(
            vert  : INTEGER ): INTEGER;
BEGIN
  GetIconNumber:= ( vert - NETGRPHEIGHT ) DIV 33 + 1;
  IF( vert < 32 ) THEN
      GetIconNumber:= 0;
END;  { of GetIconNumber }

(*
 * EventInIcon.
 *  Does the event occur in a tool icon ?.
 *)
PROCEDURE EventInIcon(
              vert  : INTEGER );
VAR
  IconNb  : INTEGER;
  oldTime : LONGINT;
BEGIN
  IconNb:= GetIconNumber( vert );
  IF( (IconNb = NIRULESELECT) OR
    (IconNb = NICLOSE) OR (IconNb = NIRIGHTEXT) ) THEN (*handle double clicks in se
    BEGIN
      oldTime:= NetSeRuTime;
      NetSeRuTime:= TickCount;
      IF( (TickCount - oldTime < GetDblTime) AND
        (NetIconChosen = IconNb) ) THEN (* this should be changed *)
        BEGIN
          CASE NetIconChosen OF
            NIRIGHTEXT:
              (* Get to the overview with the double click in arrow short cut *)
              NetDragWin;
            NICLOSE:
              BEGIN
                (* double click in eraser tool => start with a new drawing *)
                NetHideAll;
                InvalGraf;
              END;
            NIRULESELECT:NetDblSelect;
          END;
        END;
    END;
  IF( IconNb = NIGRP ) THEN
    BEGIN
      NetGrpChosen:= 3 - NetGrpChosen;
      NetGDrawIcon( NetGrpChosen );
    END
  ELSE
    IF( (IconNb <= NETMAXICON) AND (IconNb <> NetIconChosen) ) THEN
```

FOG. 11C

```
      BEGIN
        InvertIcon( NetIconChosen );
        InvertIcon( IconNb );
        NetLoadCursor( IconNb );
        NetIconChosen:= IconNb;
      END;
END;  { of EventInIcon }

(*
 *  OpenRightAll;
 *  Add the right chunks to the display.
 *)
PROCEDURE OpenRightAll(
              theHdl     : NetStHdl;
              FreeDel    : BOOLEAN );
VAR
  oldNumRight : INTEGER;
  theListHdl  : LstHdl;
  theContHdl  : LstHdl;
  myListHdl   : LstHdl;
  AddNumRight : INTEGER;
  i           : INTEGER;
  how         : WAY;
  theStr      : Str255;
  dummy       : BOOLEAN;
BEGIN
  (* different types of action depending on what is clicked on *)
  CASE TypeofNetHdl( theHdl ) OF
    SIGN:
      AtomMissing( theHdl, RIGHT, theListHdl );
    CONDITION:
      BEGIN
        IF( OpofCondNetHdl( theHdl ) IN
          [SHOW,RETRIEVE,STRATEGY,LKB,EXECUTE] ) THEN
    (* some chunks cannot be expanded *)
          EXIT( OpenRightAll );
        AtomMissing( theHdl, RIGHT, theListHdl );
      END;
  END;
  AddNumRight:= ListLength( theListHdl ) + 1;
  IF( AddNumRight = 0 ) THEN
    BEGIN
      DisposHandle( Handle( theListHdl ) );
      EXIT( OpenRightAll );
    END;
  IF( FreeDel = FALSE ) THEN
    NetFreeDelete;
  (* needs update for checking error condition *)
  oldNumRight:= RightNetHdl( theHdl );
  dummy:= NetStChange( theHdl, 0, AddNumRight );
  FOR i:= 1 TO AddNumRight DO
    BEGIN
      GetContext( theListHdl, i-1, theContHdl );
      LstCopy( theListHdl, i-1, myListHdl );
      (* call the expansion routines *)
      MakeBranch( theHdl, theListHdl, i-1, myListHdl, theContHdl,
        0, oldNumRight + i );
```

FIG. 11D

```
    . END;
   (* update the display *)
   NetGrafUpdate;
END;  { of OpenRightAll }

(*
 * ProcRightExt.
 *)
PROCEDURE ProcRightExt(
              theHdl   : NetStHdl );
BEGIN
  IF( TypeofNetHdl( theHdl ) = RULE ) THEN
    Exit( ProcRightExt );
  OpenRightAll( theHdl, FALSE );
END;

(*
 * ProcLeftExt.
 *  Add chunks to the left.
 *)
PROCEDURE ProcLeftExt(
            theHdl   : NetStHdl );
VAR
  i            : INTEGER;
  theListHdl   : LstHdl;
  theContHdl   : LstHdl;
  oldLeftNum   : INTEGER;
  dummy        : BOOLEAN;
BEGIN
  IF( TypeofNetHdl( theHdl ) = RULE ) THEN
    EXIT( ProcLeftExt );
  IF( TypeofNetHdl( theHdl ) = CONDITION ) THEN
    BEGIN
      IF( OpofCondNetHdl( theHdl ) IN
        [SHOW,RETRIEVE,STRATEGY,LKB,EXECUTE] ) THEN
        EXIT( ProcLeftExt );
    END;
  FOR i:= 1 TO LeftNetHdl( theHdl ) DO
    BEGIN
      IF( LeftExtDone( theHdl ) = TRUE ) THEN
        EXIT( ProcLeftExt );
    END;
  GetEvoke( theHdl, theListHdl );
  IF( ListLength( theListHdl ) = -1 ) THEN
    BEGIN
      DisposHandle( Handle(theListHdl) );
      EXIT( ProcLeftExt );
    END;
  NetFreeDelete;
  oldLeftNum:= LeftNetHdl( theHdl );
  dummy:= NetStChange( theHdl, 1, 0 );
  theContHdl:= EmptyLst;
  (* expand the drawing *)
  MakeBranch( theHdl, NIL, -1,theListHdl, theContHdl, oldLeftNum + 1, 0 );
  (* update the display *)
  NetGrafUpdate;
END;  { of ProcLeftExt }
```

FIG. 11E

```
(*
 * LinkExists.
 *  take care of minor problems occuring after settings.
 *)
FUNCTION LinkExists(
            leftHdl   : NetStHdl;
            rightHdl  : NetStHdl ): BOOLEAN;
BEGIN
  LinkExists:= TRUE;
  IF( (theTypeNetHdl( leftHdl ) = NRULETYPE) AND
    (theTypeNetHdl( rigthHdl ) = NCONTEXTTYPE) AND (NetContDisp = FALSE) THEN
    LinkExists:= FALSE;
END;  { of LinkExists }

(*
 * NIsLeftVis.
 *)
FUNCTION NIsLeftVis(
            theHdl  : NetStHdl ): BOOLEAN;
VAR
  extHdl   : NetStHdl;
  i        : INTEGER;
BEGIN
  NIsLeftVis:= TRUE;
  FOR i:= 1 TO LeftNetHdl( theHdl ) DO
    BEGIN
      extHdl:= GetLeftExt( theHdl,i );
      IF( theViewedNetHdl( extHdl ) = TRUE ) THEN
        EXIT( NIsLeftVis );
    END;
  NIsLeftVis:= FALSE;
END;  { of NIsLeftVis }

(*
 * NetWillClose.
 *)
FUNCTION NetWillClose(
            theHdl  : NetStHdl ): BOOLEAN;
VAR
  extHdl   : NetStHdl;
  i        : INTEGER;
BEGIN
  NetWillClose:= TRUE;
  FOR i:= 1 TO RightNetHdl( theHdl ) DO
    BEGIN
      extHdl:= GetRightExt( theHdl, i );
      IF( (isNetHdlEmpty( extHdl ) = FALSE) AND
        (theViewedNetHdl( extHdl ) = TRUE) AND
        (LinkExists( theHdl, extHdl) = TRUE) ) THEN
        EXIT( NetWillClose );
    END;
  IF( NIsLeftVis( theHdl ) = FALSE ) THEN
    EXIT( NetWillClose );
  NetWillClose:= FALSE;
END;  { of NetWillClose }
```

FIG. 11F

```
(*
 * NetKeepView.
 *)
FUNCTION NetKeepView(
              theHdl   : NetStHdl ): BOOLEAN;
VAR
  i : INTEGER;
BEGIN
  NetKeepView:= TRUE;
  FOR i:= 1 TO LeftNetHdl( theHdl ) DO
    IF( IsNetHdlEmpty( GetLeftExt( theHdl, i ) ) = TRUE ) THEN
      EXIT( NetKeepView );
  NetKeepView:= FALSE;
END;  { of NetKeepView }

(*
 * CloseLeaf.
 *)
PROCEDURE CloseLeaf( theHdl: NetStHdl; start: BOOLEAN );
VAR
  i: INTEGER;
  secHdl: NetStHdl;
BEGIN
  IF( IsNetHdlEmpty( theHdl ) = TRUE ) THEN
    EXIT( CloseLeaf );
  FOR i:= 1 TO RightNetHdl( theHdl ) DO
    BEGIN
      secHdl:= GetRightExt( theHdl, i );
      IF( LinkExists( theHdl, secHdl ) = TRUE ) THEN
        CloseLeaf( secHdl, FALSE );
    END;
  IF( (NetkeepView( theHdl ) = FALSE) AND (start = FALSE) ) THEN
    BEGIN
      SettheDeletedNetHdl( theHdl ):= TRUE;
      SettheViewedNetHdl( theHdl ):= FALSE;
      NCloseExec:= TRUE;
    END;
END;  { of CloseLeaf }

(*
 * ProcClose.
 *  Action of the eraser tool on the graphic display.
 *)
PROCEDUREE ProcClose(
              theHdl   : NetStHdl );
VAR
  i         : INTEGER;
  left      : INTEGER;
  theGrp    : NetGrpPtr;
  theFlavor : FLAVOR;
  CloseBool : BOOLEAN;
  Viewed    : BOOLEAN;
BEGIN
  IF( NetWillClose( theHdl ) = TRUE ) THEN
    NetFreeDelete
  ELSE
```

FIG. 11G

```
    EXIT( ProcClose );
  theGrp:= GetGroup( theHdl );
  left:= LeftNetHdl( theHdl );
  theFlavor:= TypeofNetHdl( theHdl );
  CloseLeaf( theHdl, TRUE );
  IF( TheFlavor = RULE ) THEN
    BEGIN
      SettheDeletedNetHdl( theHdl ):= TRUE;
      SettheViewedNetHdl( theHdl ):= FALSE;
      NCLoseExec:= TRUE;
    END
  ELSE
    BEGIN
      IF( NIsLeftVis( theHdl ) = FALSE ) THEN
        BEGIN
          Viewed:= FALSE;
          FOR i:= 1 TO RightNetHdl( theHdl ) DO
           BEGIN
              IF( (IsNetHdlEmpty( GetRightEXt( theHdl, i ) ) = TRUE) AND
                (SettheViewedNetHdl( GetRightExt( theHdl, i ) ) = TRUE) ) THEN
                BEGIN
                  Viewed:= TRUE;
                  LEAVE;
                END;
            END;
          IF( Viewed = FALSE ) THEN
            BEGIN
              SettheDeletedNetHdl( theHdl ):= TRUE;
              SettheViewedNetHdl( theHdl ):= FALSE;
              NCLoseExec:= TRUE;
            END;
        END;
    END;
  IF( NCloseExec = TRUE ) THEN
    BEGIN
      CheckGroup( theGrp );
      NetGrafUpdate;
    END;
END;  { of ProcClose }

(*
 *  NProcRuleSelect.
 *  Action of the magnifying glass icon. Call the restart functions.
 *)
PROCEDURE NProcRuleSelect(
              theHdl   : NetStHdl;
              Erase    : BOOLEAN );
BEGIN
  IF( TypeofNetHdl( theHdl ) = CONDITION ) THEN
    Exit( NProcRuleSelect );
  CASE TypeofNetHdl( theHdl ) OF
    RULE:
      NetHdlAtomInit( theHdl, erase, FALSE );
    SIGN:
      NetSignHdlAtomInit( theHdl, erase, FALSE );
  END;
END;  { of NProcRuleSelect }
```

FIG. 11H

```
(*
 * ProcGetInfo.
 *  Action of the INFO tool on the network.
 *  Get an information file about the chunk theHdl is displaying.
 *)
PROCEDURE ProcGetInfo(
              theHdl   : NetStHdl );
VAR
  i : INTEGER;
  theStr: Str255;
BEGIN
  NetHdlToString( theHdl, theStr );
  DisplayInfoFile( theStr );
END;  { of ProcGetInfo }

(*
 * NetConfirmProcess.
 *  Handles the networks clicks in a dialog fashion.
 *  theHdl region is inverted while the mouse is inside.
 *)
FUNCTION NetConfirmProcess( theHdl: NetStHdl ): BOOLEAN;
VAR
  theRect : Rect;
  GrafRect: Rect;
  thePoint: Point;
  theGrp  : NetGrpPtr;
  dummy   : BOOLEAN;
BEGIN
  theGrp:= GetGroup( theHdl );
  SetTrackMouse( theHdl, theRect );
  GetGrafRect( GrafRect );
  dummy:= SectRect( theRect, GrafRect, theRect );
  NetConfirmProcess:= TrackMouse( theRect );
  SetOrigine( 0, 0 );
END;  { of NetConfirmProcess }


(*
 * ProcessClick.
 *  this is the actual dispatch routine.
 *)
PROCEDURE ProcessClick( theHdl: NetStHdl );
BEGIN
  IF( IsNetHdlEmpty( theHdl ) = NIL ) THEN
    EXIT( ProcessClick );
  IF( NetConfirmProcess( theHdl ) = FALSE ) THEN
    EXIT( ProcessClick );
  Setcursor (watchhdl^^);
  NetGrpCurrent:= GetGroup( theHdl );
  CASE NetIconChosen OF
    NIRIGHTEXT:
      ProcRightExt( theHdl );
    NILEFTEXT:
      ProcLeftExt( theHdl ) ;
    NICLOSE:
        ProcClose( theHdl );  (* now don't erases a group *)
```

FIG. 11I

```
    NIRULESELECT:
      NProcRuleSelect( theHdl, ( NetGrpChosen = NIGRP1 ) );
    NIGETINFO:
      ProcGetInfo( theHdl );
    END; { of case }
  SetCursor (arrow);
END;

(*
 * NetProcessEvent.
 *)
PROCEDURE NetProcessEvent(
            thePoint  : Point );
VAR
  GZone   : INTEGER;
  theHdl  : NetStHdl;
BEGIN
  SetPort( GrafPtr( @NetWindowRec ) );
  (* the two following lines reset global variables *)
  NetGrpCurrent:= NIL;
  (* NoOrgMove;*)
  IF( NProcControl( thePoint ) ) THEN
    EXIT( NetProcessEvent );
  IF FindGZone( thePoint.h, GZone ) THEN
    BEGIN
      IF( GZone = 0 ) THEN
        (* the event occured in the Tools region *)
        EventInIcon( thePoint.v )
      ELSE
        BEGIN
        (* the event was in the display *)
          theHdl:= FindAtom( VWinToNet( thePoint.v ), GZone );
          IF( theHdl <> NIL ) THEN
            ProcessClick( theHdl );
        END;
    END;
  NetGrpCurrent:= NIL;
END;  { of NetProcessEvent }
```

FIG. 11J

Description of non documented routines.
These routines not included with the code do not relate directly to the dynamic browsing mecanism but to the internal representation of the display in memory.

- NetEvokeInit
  First Argument: Starting working memory handle
  Second Argument: Should the display be erased. (Boolean)
This subroutine adds a new group to the display, starting on a given memory handle and displaying all the left linked chunks.

- NetAtomInit
  First Argument: Starting working memory handle
  Second Argument: Should the display be erased. (boolean)
  Third Arguement: should the display be totally deleted from memory.
This subroutine adds a new group to the display, starting on a given memory handle and displaying all the rigth linked chunks.

- HWinToNet.
  Translate an absolute position in the network window into an absolute position in the full display in memory.

- FirstNetGrp.
  Returns a pointer to the first group allocated in the display.

- NextNetGrp
  Returns a pointer to the next group allocated in the display.

- FirstNetHdl.
  Returns a handle to the first chunk allocated in a given group

- PosInHdl.
  Given a position, a chunk handle and the group it belongs to, this routines returns if the position is pointing to the chunk.

- NextNetHdl.
  Given a handle to a chunk, return the next chunk allocated (which will be in the same group).

- NetDragWin
  Displays the overview of the network.

- NetHideAll
  Erases all the chunks from the display. It doesn't delete them from memory (This is done by NetFreeDeleted)

- InvalGraf.
  Forces the update of the display.

- AtomMissing.
  Returns in the third argument a list of all the chunks missing. The second argument indicated if we are requesting the missing chunks to the right or to the left.

- OpOfCondNetHdl
  returns the operator of the chunk displayed. This chunk has to be pointing to a condition (special instance of a chunk)

FIG. 12A

- ListLength
  Returns the length of a list of chunks.

- NetFreeDeleted
  Deletes from memory the invisible chunks.

- RigthNetHdl.
  Returns the number of chunks already displayed to the right of the current chunk.

- GetContext
  Returns a special type of chunks which can be linked to the current chunk.

- LstCopy
  duplicates a list of chunks.

- MakeBranch
  Calls the allocation routine of the display. The arguments indicate the new lists of chunks and how they should be linked to the current chunk.

- NetGrafUpdate.
  Forces an immediate update of the display.

- TypeofNetHdl.
  Returns the type of the chunk displayed.

- LeftExtDone
  Returns if the left chunks are already displayed.

- LeftNetHdl.
  Returns the number of chunks already linked to the left.

- EmptyLst
  returns an empty list of chunks.

- theTypeNetHdl
  Returns the type of the display handle (this is different of the chunk type because the same chunk can be displayed several times in different situations)

- GetLeftExt
  Returns a handle to a left linked chunk.

- GetRigthExt
  Returns a handle to a rigth linked chunk.

- IsNetHdlEmpty.
  Returns if the display handle is pointing to a real chunk. (display handles can be used for graphics operations only)

- theViewedNetHdl
  Returns if the display handle is visible in the display.

- NIsLeftVis
  Returns if the display handle is connected to a visible chunk.

FIG. 12B

- SettheDeletedNetHdl.
  Marks a display handle as deleted
- SettheViewedNetHdl
  Marks a display handle as invisible.
- CheckGroup
  Checks if a group is still visible in the display. If not, this routine deletes it from memory.
- NetHdlAtomInit.
  Calls NetAtomInit with the chunk the display handle is pointing at.
- NetSignHdlAtomInit.
  Calls NetAtomInit with a chunk depending on the chunk the display handle is pointing at.
- DisplayInfoFile
  Brings on the screen some information about hte current chunk.
- SetTrackMouse
  Computes the boundaries the mouse can move inside of, for selecting the current chunk.

FIG. 12C