FONCTIONNEMENT D'UN SYSTÈME À BASE DE RÈGLES[1]

Par Lucien Roy

SOMMAIRE

Introduction **1**
1. Émergence des systèmes à base de connaissances **1**
2. Structure d'un système à base de règles **4**
2.1 Schéma fonctionnel 4
2.2 Fonctions de chaque composante 5
3. Modes d'opération d'un système à base de règles **7**
4. Logique de fonctionnement du moteur d'inférence **10**
4.1 Navigation dans un réseau 11
4.2 Aperçu du fonctionnement d'un moteur d'inférence 13
4.3 Chaînage avant 16
4.4 Chaînage arrière 18
4.5 Variantes des mécanismes d'inférence de base **21**
Chaînage mixte 21
Recherche en largeur ou en profondeur 21
5. Évaluation des mécanismes **22**
5.1 Observations sur le fonctionnement d'un moteur d'inférence 22
5.2 Comparaison et synthèse des méthodes d'inférence 24
6. Optimisation de la performance d'un moteur d'inférence **25**
6.1 Structuration des connaissances dans une base de règles 25
6.2 Priorités d'application des règles 27
6.3 Métarègles 27
6.4 Heuristiques 28
7. Traitement des connaissances incertaines **28**
Conclusion **31**
À retenir **32**

[1] Ce texte est extrait en majeure partie du volume G.Paquette et L. Roy, « Systèmes à base de connaissances, Télé-université et Beauchemin, pp.121-174

Introduction

Après avoir vu différentes formes de représentations des connaissances, nous allons maintenant examiner les outils informatiques susceptibles d'implanter ces connaissances en vue de les exploiter. C'est pourquoi nous passons à l'étude d'un environnement qui supporte une représentation particulière : les systèmes à base de règles.

Nous avons choisi les systèmes à base de règles (SBR) pour trois raisons. Premièrement, ce sont les systèmes à base de connaissances les plus répandus dans les applications. Deuxièmement, les représentations à base de règles mènent à une codification des connaissances en des termes assez simples, les règles si... alors... Il s'agit d'une façon d'exprimer les connaissances qui nous est familière. De plus, les SBR offrent des éditeurs de règles qui facilitent la codification des connaissances d'une application. Finalement, la troisième raison de notre choix est pédagogique. Nous tenons à exploiter deux fonctionnalités originales des SBR, soit la présence de mécanismes de raisonnement transparents et l'existence d'un module d'explication de leur fonctionnement. Cela nous donne accès aux mécanismes d'inférence et à la manière dont des conclusions peuvent être déduites par un programme.

Nous situons d'abord les SBR dans l'évolution générale des divers types de logiciels. La manière d'opérer un SBR sera ensuite abordée par l'étude des interfaces avec leurs usagers. Nous accordons aussi une attention particulière au fonctionnement du module qui fait l'originalité des SBR, le moteur d'inférence. Puis nous présentons des algorithmes d'inférence par chaînage avant et par chaînage arrière· qui sont accompagnés de simulations avec des exemples bases de règles. Enfin, nous procédons par analogie et par raffinement graduel dans l'explication des mécanismes d'inférence. C'est ce qui nous permet de découvrir quelques méthodes d'optimisation pour améliorer leur performance lorsqu'un grand nombre de règles devient nécessaire.

1. Émergence des systèmes à base de connaissances

Les systèmes à base de connaissances forment une famille de logiciels au même titre que les langages de programmation et les systèmes de gestion de bases de données (SGBD). Qu'est-ce qui distingue les SBC des autres familles de logiciels? La genèse des SBC fournit une réponse éclairante à cette question. En effet, l'étude de l'évolution des logiciels de l'informatique nous montre que *données et traitements ont été progressivement dissociés* physiquement, puis logiquement. La figure 1 résume cette évolution.

Revenons d'abord aux *programmes écrits en langage de base* (langage machine ou assembleur), le seul type de langage disponible au début de l'informatique. À cette époque, les programmeurs avaient une très grande marge de manœuvre dans l'utilisation de la mémoire centrale de l'ordinateur. Ils n'étaient pas tenus de définir deux zones distinctes de mémoire, une pour les données et une autre pour les instructions.

La situation change radicalement avec l'apparition de *langages évolués* comme le FORTRAN ou le COBOL, où données et instructions (traitements) occupent des zones séparées de la mémoire. Si la séparation physique est ainsi accomplie, la séparation logique ne l'est pas encore. En effet, les instructions, une fois traduites en langage machine par le compilateur, supposent que les données sont invariables quant à leur emplacement en mémoire et à leur structure. Ce fait est particulièrement gênant pour les zones tampons qui servent aux échanges avec les fichiers. Ainsi, la structure des enregistrements d'un fichier ne peut être changée sans révision et recompilation de tous les programmes y accédant.

Un pas très important est franchi avec *les systèmes de gestion de bases de données* (SGBD). On extrait des programmes la définition des enregistrements pour la placer au début du fichier, dans une sorte d'en-tête (*header*). Les programmes sont alors dotés, sans intervention du programmeur, d'une couche additionnelle de code pour aller chercher, à cet endroit, l'information qui décrit l'organisation des données dans le fichier. Ainsi, les traitements sont codés dans des programmes, alors que les données et leur organisation (la structure) sont entreposées dans une base de données.

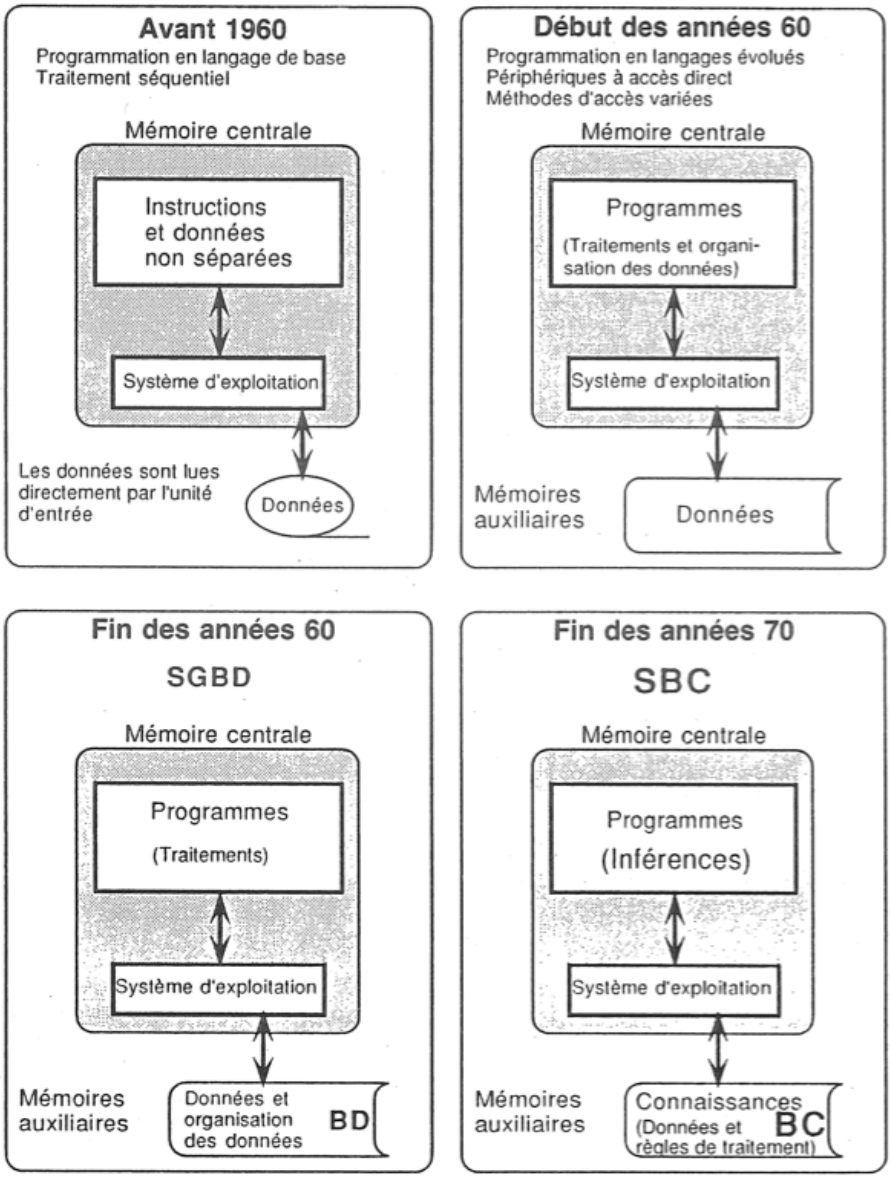


Figure 1 Séparation progressive des données et des programmes.

Avec la mise au point des *systèmes à base de connaissances*, on généralise la notion de données. On parvient à intégrer aux données des structures de contrôle comme des décisions. En effet, on emmagasine les données, leur organisation et les règles de traitement dans une base de connaissances. Les traitements décrits sous forme de règles sont des données pour les mécanismes d'inférence appliqués aux connaissances. La figure 1 illustre cette évolution *d'indépendance des données par rapport aux programmes.* Il en résulte une possibilité intéressante : on peut modifier le contenu d'une base de connaissances et donc, la logique d'un programme sans être obligé de modifier les programmes qui l'exploitent. Le même moteur d'inférence sert à plusieurs bases de connaissances sur des sujets possiblement très différents. Ces possibilités confèrent une souplesse indéniable aux SBC. Cette circonstance fait que le développement d'une application avec un SBC passe d'abord par l'analyse des connaissances à traiter. C'est l'étape cruciale de la représentation des connaissances, objet d'étude d'un texte précédent.

Voyons maintenant comment on peut traduire concrètement la séparation des données et des programmes avec deux outils informatiques distincts. Supposons que l'on désire calculer le montant d'impôt qu'un contribuable doit payer en tenant compte de son état civil, de son revenu brut et d'une table d'imposition basée sur le revenu imposable. Avec un langage procédural évolué comme le Pascal, une solution à notre problème pourrait ressembler à celle-ci :

```
PROCEDURE CALCULER_IMPOT;
BEGIN
READLN (état_civil, revenu_brut);                    (*Lecture des données*)
IF état_civil = "marié" THEN                         (*Calcul de l'exemption*)
        Exemption = 11000                                  (*selon l'état civil*)
    ELSE
        Exemption = 6000;
revenue_imposable : = revenu_brut – exemption;
IF revenue_imposable < = 15000 THEN                  (*Calcul de l'impôt selon*)
        impôt : = revenu_imposable*0.08                    (*le revenu imposable*)
    ELSE
        impôt : = 1200 + (revenu_imposable – 15000) * 0.12)
END;
```

Voici la même solution traduite en règles que l'on pourrait incorporer à une base de règles :

Si paramètre_impôt inconnu
Alors obtenir état civil Et obtenir revenu brut

Si état civil est marié
Alors exemption = 11 000

Si état civil est non marié
Alors exemption = 6 000

Si revenu_imposable inconnu
Alors calculer_revenu_imposable

Si revenu_imposable <= 15 000

Alors impôt = revenu_imposable* 0,08

Si revenu_imposable > 15 000
Alors impôt = 1 200 + (revenu_imposable – 15 000) * 0,12

Bien que ces deux solutions soient équivalentes, elles présentent une différence très importante. Certains éléments du programme Pascal comme les branchements conditionnels *if*.... *then*... sont incorporés à la base de règles et sorties du programme. C'est ainsi que des règles de traitement qui font partie d'un programme deviennent des données pour le moteur d'inférence d'un système à base de règles.

2. Structure d'un système à base de règles

Nous procédons à l'analyse d'un système à base de règles afin d'en identifier les composantes, de montrer comment elles sont agencées et forment un ensemble opérationnel.

2.1 Schéma fonctionnel

Toutes les parties d'un système à base de connaissances doivent travailler en collaboration pour que l'objectif de l'ensemble soit atteint. De ce point de vue, on peut définir la finalité d'un SBC comme étant la possibilité de remplacer un expert d'un domaine bien identifié. Pour y parvenir, un SBC doit supporter des fonctionnalités qui le rendent capable d'acquérir du savoir-faire et de répondre aux besoins de ses usagers. Or, l'expertise se manifeste principalement par l'accumulation de connaissances, la capacité d'utiliser ces connaissances pour résoudre des problèmes et la faculté d'expliquer la manière d'exploiter ces connaissances.

On s'est beaucoup inspiré de ces remarques pour concevoir l'architecture d'un système à base de règles. Ses deux éléments essentiels sont une base de règles et un mécanisme de raisonnement appelé *moteur d'inférence.*

On a conçu un système à base de connaissances comme une machine à qui l'on transmet les connaissances d'un expert sur un sujet donné, en vue de solutionner des problèmes que lui soumettent des usagers. Enfin, sa configuration est déterminée par les conditions d'opération suivantes :

- D'abord, on doit construire une base de règles.
- Ensuite, le SBR dialogue avec l'usager en utilisant la base de connaissances et en gardant une trace des déductions (inférences) effectuées.
- Puis, le moteur d'inférence traite les règles pour effectuer les déductions.

La figure 2 illustre la configuration générale d'un SBR et montre le rôle d'un SBR dans les étapes d'opération d'un système de traitement de l'information. On peut suivre le parcours des connaissances qui sont d'abord analysées et fournies par les experts et les cogniticiens[2] au SBC qui les traite en vue de répondre aux besoins des usagers. On voit que les parties essentielles d'un système à base de connaissances sont :

- les données : une base de connaissances et une base de faits,

[2] *Expert* et *cogniticien* sont des termes génériques. « Expert » désigne une personne qui possède une connaissance approfondie d'un domaine du savoir. « Cogniticien » désigne une personne spécialisée dans des méthodes d'acquisition et de représentation des connaissances permettant de construire une base de connaissances.

- les programmes :le moteur d'inférence (MI) et les deux interfaces pour introduire les connaissances dans la base et fournir à l'usager des déductions du MI ou des explications sur son raisonnement.

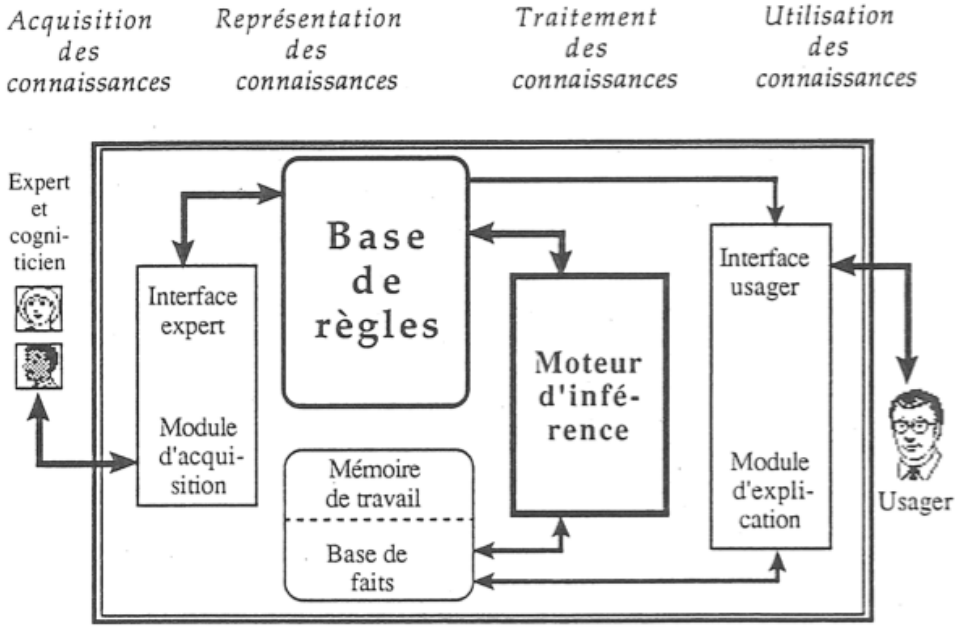


Figure 2 Architecture d'un système à base de règles.

On appelle *coquille* ou système essentiel l'ensemble des programmes formés par le moteur d'inférence et les interfaces usager et expert. On se procure une coquille vide pour construire et exploiter diverses bases de connaissances afin de répondre à des problèmes posés par des usagers. Une telle approche nous permet de centrer notre attention sur la déclaration des connaissances à traiter et sur le dialogue avec l'usager. Cela change complètement l'approche du développement d'applications informatiques. On doit se concentrer sur l'identification, l'acquisition et la représentation des connaissances.

2.2 Fonctions de chaque composante

Les connaissances et les faits

Un système à base de règles traite, d'abord, le savoir d'un domaine applicable à plusieurs problèmes : les connaissances; puis, les données particulières qui caractérisent chaque problème posé au SBR : les faits. Cette distinction présente une analogie avec le modèle humain de traitement de l'information décrit dans le texte 2. En effet, les connaissances sont stockées dans la base de règles, sorte de mémoire à long terme d'un SBR. D'autre part, les faits sont conservés dans la base de faits qui tient lieu de mémoire à court terme.

La base de règles

C'est l'endroit où sont stockées les données d'un SBR. Le contenu de la *base de règles* est une représentation du savoir d'un domaine d'application. Le fonctionnement du SBR est centré sur cette base et la richesse de son contenu en détermine la performance. On doit

créer une base de règles initiales à l'aide de l'interface expert. Le moteur d'inférence en récupère le contenu lors de l'opération selon les inférences qu'il doit effectuer.

La mémoire de travail

C'est une zone de mémoire qui sert de tampon au MI. Il y conserve une trace de ses déductions, des buts intermédiaires qu'il se fixe lors de la recherche d'une solution *et* des règles utilisées. Toutes les déductions effectuées par le moteur d'inférence et les réponses fournies par l'usager sont conservées dans la *base de faits.* Son contenu est géré par le moteur d'inférence et est initialisé de nouveau au début de chaque séance d'interrogation du SBC. L'usager peut initialiser ou interroger le contenu de la mémoire de travail au moyen de l'interface usager. Mais l'essentiel de son contenu constitue la *trace* des raisonnements effectués par le moteur d'inférence. *À* cet effet, la mémoire de travail contient les données suivantes :

- les conclusions déduites à partir de faits considérés vrais par hypothèse,
- les conditions à vérifier pour qu'une conclusion se réalise,
- les règles appliquées ou les messages échangés afin de satisfaire la requête de l'usager.

La base de faits contient les données spécifiques à une séance de travail avec le SBR. Elle occupe une partie de la mémoire de travail ou bien elle est entreposée sur fichier magnétique. On l'appelle aussi base de faits expérimentaux. Les faits sont des connaissances factuelles généralement exprimées sous forme de propositions simples.

Un système à base de connaissances conserve une trace des opérations qu'il effectue en empilant les conclusions, les conditions et les règles impliquées dans son processus d'inférence. La structure de pile permet au SBR de retrouver les déductions effectuées et la façon dont ces déductions furent obtenues. C'est de cette manière qu'un SBC peut expliquer son fonctionnement. Voilà une originalité qu'aucun logiciel classique ne peut offrir.

Le moteur d'inférence (MI)

Il s'agit d'un programme spécifique aux SBR : il fait des déductions en exploitant une base de connaissances et des réponses fournies par l'usager. Sa généralité est telle qu'il peut déduire des faits dans plusieurs contextes, puisqu'il traite des connaissances de plusieurs sujets différents. En effet, le moteur d'inférence peut exploiter plusieurs bases de connaissances, une pour chaque contexte d'application. Le MI est le cœur d'un système à base de connaissances et a trois fonctions :

- Prendre connaissance des requêtes de l'usager entreposées dans la base de faits.
- Répondre aux questions de l'usager : il sélectionne et applique les règles appropriées contenues dans la base de connaissances. C'est ce qui lui permet de déduire des résultats de son raisonnement.
- Générer une trace de ses actions qu'il conserve dans la base de faits.

Les moteurs d'inférence appliquent des règles en suivant des variantes des deux modes de raisonnement suivants :

- en chaînage avant : en partant de faits considérés vrais par hypothèse, il en déduit les conclusions possibles,

- en chaînage arrière : en partant d'une conclusion à démontrer, il découvre les conditions qui doivent être réalisées pour que la conclusion soit vérifiée.

Les interfaces

Les interfaces sont des modules programmés qui tiennent lieu d'unité d'entrée et d'unité de sortie d'un système à base de connaissances. Elles permettent l'interaction avec les deux types d'opérateurs d'un SBR : d'une part, les experts et les cogniticiens qui construisent la base de connaissances et, d'autre part, les usagers qui ont recours aux services du SBR.

L'interface expert

C'est par l'intermédiaire de l'interface expert que l'on peut transmettre à un SBR les connaissances requises à la solution de problèmes dans un domaine d'application. On y parvient en éditant une base de règles. En général, l'interface expert comporte les modules suivants :

- un éditeur de règles pour initialiser ou modifier le contenu de la base,
- un éditeur de dialogue avec l'usager permettant aux experts de prévoir les questions que le SBR devra poser à l'usager pour le guider dans sa recherche de solution,
- un analyseur ou un compilateur de base pour traduire les connaissances des experts en structures de données reconnues par le moteur d'inférence.

En général, l'interface expert sert de lien entre les experts, les cogniticiens et la base de connaissances. Mais elle peut également transposer en règles des données factuelles entreposées dans une base de. données ou dans un chiffrier. C'est une des manifestations de *l'ouverture* des SBR sur d'autres outils informatiques.

L'interface usager

- L'interface usager assure la communication entre l'usager et le SBR. Il s'agit d'un module subordonné au moteur d'inférence. Le module pose des questions à l'usager, au besoin, et recueille les réponses de ce dernier pour les transmettre au moteur d'inférence. La convivialité de cette interface varie selon le prix du système. Actuellement, on essaie de concevoir des interfaces graphiques ou des interfaces qui reconnaîtraient des requêtes en langue naturelle, pour permettre une plus large diffusion des SBR.
- L'interface usager est composée de deux modules : un module de dialogue avec le SBR et un *module d'explication* par lequel le moteur d'inférence explique comment il utilise le contenu de la base de connaissances pour répondre aux questions de l'usager. Il récupère les données empilées par le moteur d'inférence pour restituer la trace du raisonnement effectué.

3. Modes d'opération d'un système à base de règles

Le dialogue entre un usager et un système base de règles peut nous aider à comprendre son fonctionnement. À ce propos, voici un extrait d'une « conversation » avec la coquille *M.I* qui accède à la base de *Conseiller en vins.* Il s'agit d'un petit système expert qui fournit des suggestions de vins pour accompagner un repas. On veut se servir de la base *Conseiller en vins* pour déterminer quel vin serait le plus approprié à un bœuf Strogonoff. Les réponses de l'usager sont précédées du symbole « > » et sont en italique.

Le plat principal contient-il de la viande, du poisson ou de la volaille?
> *viande*
Est-ce qu'il y a du veau dans le repas?
> *non*
Une sauce accompagne-t-elle le repas?
> pourquoi?
Votre réponse va aider à déterminer si la règle suivante s'applique :
Si le principal ingrédient n'est pas du poisson ET
il y a de la sauce ET
la sauce est aux tomates
Alors la couleur du vin recommandée est rouge
Une sauce accompagne-t-elle le repas?
> *oui*
Est-ce que la sauce est épicée, douce, crémeuse ou aux tomates?
> *crémeuse*
Préférez-vous généralement les vins rouges ou blancs?
> *rouge*
Le repas est à saveur délicate, moyenne ou forte?
> *je ne sais pas*
Préférez-vous généralement les vins légers, moyens ou corsés?
> *légers*
Préférez-vous généralement les vins secs, moyens ou sucrés?
> *secs*
Voici mes conclusions :
Barolo (80 %)
Cabernet-Sauvignon (80 %)
Bourgogne (60 %)
> *fin*

Comme toutes les coquilles de SBR, *M.I* peut expliquer son fonctionnement. C'est ainsi qu'à la question *pourquoi?,* il identifie la règle qu'il s'apprête à appliquer. Par ailleurs, il peut continuer son cheminement même lorsque l'usager ignore la réponse à une question comme en témoigne le *je ne sais pas.*

L'exemple précédent montre que l'on peut dialoguer avec un SBR. En général, on peut opérer un SBR selon l'un des deux modes suivants :

- Acquisition des connaissances permettant d'éditer le contenu d'une base de connaissances; l'interface expert est alors utilisée.
- Dialogue avec le SBR qui exploite une base de règles et exécute un mécanisme d'inférence pour déduire des conclusions; l'interface usager se charge de communiquer les conclusions à l'usager et de fournir des explications sur la manière dont s'effectue le raisonnement.

L'acquisition des connaissances est l'étape préalable à l'utilisation d'un SBR. Elle consiste à l'instruire par construction d'une base de règles. Cette étape fait intervenir l'interface expert.

Une fois la base de connaissances constituée, l'usager peut se servir du SBR. C'est l'étape de l'exploitation ou du traitement des connaissances. Concrètement, cette opération prend la forme d'un *dialogue* avec l'usager. Le dialogue SBR-usager s'effectue normalement selon le mode suivant : *le SBR pose une question et l'usager répond.* Le moteur d'inférence se sert des réponses de l'usager et des connaissances dans la base pour arriver à ses fins. Au cours d'une séance de travail, le SBR essaie d'atteindre un *but,* soit en déduisant des conclusions à partir d'hypothèses fournies par l'usager, soit en déterminant des conditions à satisfaire pour que la conclusion fixée par l'usager se réalise. La tâche du moteur d'inférence revient à déduire des faits en affectant des valeurs à des attributs. Pour y parvenir, il applique des règles et il obtient des réponses de l'usager lorsque sa base de connaissances ne lui permet plus de poursuivre son raisonnement.

La possibilité d'opérer en mode explication est exclusive à un système à base de connaissances. Cela permet d'atteindre un double but en travaillant avec un SBR : apprendre les connaissances dont il dispose et observer comment ces connaissances sont mises en pratique. On a déjà mentionné que le moteur d'inférence conserve une trace de ses étapes de raisonnement. Pour ce faire, il gère trois piles dans la base de faits. En effet, le MI empile les réponses de l'usager, les faits déduits et les règles appliquées. Les empilements de la base de faits permettent au module d'explication de fournir à l'usager une réponse à des questions du type :

Pourquoi?	Le module donne les raisons qui ont conduit le SBR à poser telle question à l'usager (justification).
Comment?	Il énumère les arguments (les règles) invoqués pour déduire telle conclusion (justification).
Quoi si?	Il fournit les déductions possibles si telle réponse était fournie par l'usager (anticipation).
Quoi?	Il produit une liste des résultats du raisonnement depuis que le dialogue est amorcé avec l'usager (sommaire).

En somme, le module d'explication d'un SBR peut justifier les résultats déduits par son mécanisme d'inférence, anticiper des conclusions selon des circonstances hypothétiques et produire un sommaire du dialogue avec l'usager. Toutes ces possibilités sont accessibles à n'importe quel moment. Ce sont là des manifestations d'un comportement « intelligent », n'est-ce pas?

Déroulement d'une session de travail

Supposons que nous disposons d'un SBR pouvant accéder à quelques bases de règles préalablement créées, les étapes d'opérations qui permettront d'explorer les principales fonctionnalités du SBR sont celles-ci :

1. Sélection d'une base de règles. – Il s'agit d'indiquer au système quelle base doit être chargée en mémoire.
2. Sélection du mécanisme d'inférence. – La plupart des MI supportent des variantes des deux modes d'inférence de base : le chaînage avant et le chaînage arrière. C'est pourquoi on doit préciser le mode d'inférence à appliquer pour la prochaine étape.

3. Initialisation de la base de faits. – L'usager pose le problème au SBR. Si on a choisi le chaînage avant l'étape précédente, on fournit les faits qui constituent les hypothèses à partir desquelles le MI fera ses déductions. Dans le cas du chaînage arrière, on définit la conclusion à démontrer par le Ml.
4. Enclenchement du moteur d'inférence.
5. Dialogue SBR-usager. – Le moteur d'inférence poursuit son mécanisme de raisonnement. Il peut solliciter l'usager pour qu'il fournisse des réponses aux questions qu'il lui pose. L'usager peut répondre directement aux questions ou demander au SBR de fournir des explications sur le déroulement de la session de travail.

4. Logique de fonctionnement du moteur d'inférence

Nous allons maintenant nous attarder au fonctionnement du module le plus important d'un système à base de règles : le moteur d'inférence. C'est lui qui en constitue la partie active en raison de sa méthode d'exploitation des connaissances. On peut programmer un MI pour qu'il « raisonne », c'est-à-dire qu'il effectue ses déductions en utilisant les règles de différentes manières. Nous verrons les deux modes fondamentaux de fonctionnement d'un MI : en *chaînage avant* et en *chaînage arrière,* chacun de ces modes étant approprié à des contextes particuliers.

Voyons comment un moteur d'inférence peut atteindre un but en appliquant des règles. Pour centrer notre attention sur le mécanisme de déduction, nous partons de la petite base de règles suivante :

Rl	si B et D et E	alors F
R2	si D et G	alors A
R3	si F et C	alors A
R4	si B	alors X
R5	si D	alors E
R6	si A et X	alors H
R7	si C	alors D
R8	si X et C	alors A
R9	si X et B	alors D

Supposons que les faits B et C sont connus. Essayons de démontrer A en appliquant les neufs règles précédentes. La figure 3 montre un cheminement possible.

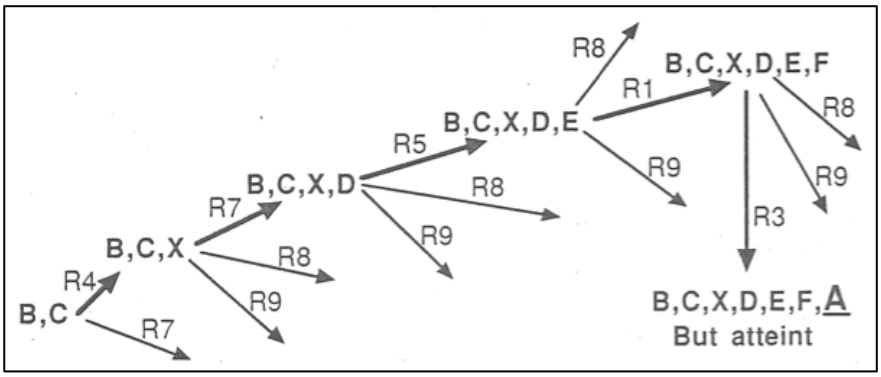


Figure 3 Simulation d'un mode d'inférence.

Les « faits » connus ou déduits sont en caractères gras. Bien que cet exemple ne fasse intervenir que des variables (A, B, C, etc.), on pourrait appliquer notre méthode de déduction à un contexte réel. Il suffirait de remplacer les symboles par des propositions ou des faits comme « La cliente est privilégiée » ou « Le patient est cardiaque » pour en déduire des faits ayant une signification concrète. Nous nous limitons à une forme symbolique pour mettre l'accent sur le *mécanisme* d'inférence.

La procédure d'inférence simulée ici est simple : nous parcourons séquentiellement les règles pour rechercher les symboles connus dans leur partie condition. Quand tous les symboles de la partie condition d'une règle sont déjà connus, nous ajoutons sa conclusion à la liste des faits connus. Nous procédons ainsi jusqu'à ce que le but à atteindre (A) puisse être déduit ou qu'il ne reste plus de règles applicables. Ce mode d'inférence est particulier, on l'appelle chaînage avant. On peut le programmer pour en faire un moteur d'inférence qui pourra déduire des faits à partir de faits déjà connus. C'est donc une forme de raisonnement que l'on peut informatiser.

4.1 Navigation dans un réseau

On peut considérer un moteur d'inférence comme une machine qui recherche des solutions aux problèmes qu'on lui soumet et pour y arriver, il doit parfois suivre un cheminement sinueux. De ce point de vue, la recherche d'une solution se compare à la recherche d'un chemin nous menant d'un point de départ à une destination choisie. Il s'agit d'une recherche d'un parcours dans un réseau par l'application d'une stratégie de navigation. Le choix d'un itinéraire sur un réseau routier ou la détermination du chemin le plus court parmi plusieurs chemins possibles entre deux lieux en sont des exemples.

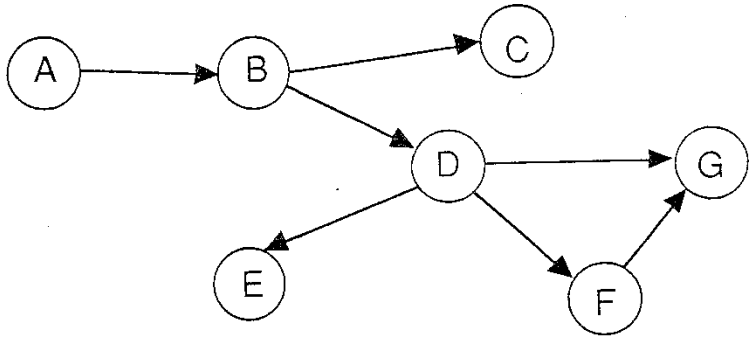


Figure 4 Un petit réseau.

La figure 4 montre que les deux constituants d'un réseau sont les *nœuds* et les *arcs.* Les nœuds correspondent à des lieux ou à des états à atteindre et les arcs sont des chemins, sorte de liens entre les nœuds. Nous n'avons qu'à penser à des réseaux routiers, aériens, de communication ou neuronaux pour imaginer ce que sont les nœuds et les arcs dans chacun de ces contextes.

Une stratégie de navigation dans un réseau repose sur une procédure de *recherche de parcours* entre deux nœuds : le départ et la destination. L'élément essentiel de la stratégie est le suivant : parvenu à un nœud, il nous faut trouver un arc (chemin) qui nous conduira à un nœud successeur pour assurer la poursuite de notre parcours vers la destination. Mais on peut se retrouver dans un cul-de-sac, c'est-à-dire qu'après avoir atteint un nœud, il est possible que l'on ne puisse continuer vers la destination. Dans ce cas, nous devons faire

marche arrière pour essayer d'autres parcours. Par ailleurs, il est toujours possible et instructif de conserver une trace du parcours emprunté et même des essais infructueux en vue d'arriver à une solution. Cela est possible en empilant les nœuds atteints dans le parcours menant du point de départ au point d'arrivée.

Imaginons que la figure 4 représente un réseau routier reliant des villes identifiées par des lettres et que l'on veut atteindre G à partir de A. Il y a plus d'un parcours possible et certains mènent à une impasse. Voici les étapes possibles du trajet A à G :

- A, B, C, cul-de-sac
- retour à B, D, E, cul-de-sac
- retour à D, G ou F, G

Notre stratégie est simple : avancer jusqu'à une impasse et retour en arrière pour trouver un nœud à partir duquel on peut continuer notre cheminement vers le but. À partir d'un nœud, il peut y avoir plusieurs destinations possibles. On doit choisir la direction qui nous semble conduire au but ou qui nous semble 1a plus intéressante. On peut également conserver une trace de notre parcours en notant les nœuds atteints. C'est ce que l'on appelle une description qualitative d'une méthode de navigation systématique dans un réseau. On peut en donner une description plus précise sous forme procédurale. En voici une sous forme d'algorithme :

```
Paramètres de navigation : Réseau, Départ, Destination,
    Pile nœuds
Procédure Parcourir Réseau
        Obtenir Départ et Destination
        Pile_nœuds <- Départ
        TANT QUE (Pile_nœuds non vide et Destination pas atteinte)
           FAIRE
                Identifier les nœuds accessibles
                Si au moins un nœud accessible ALORS
                        Choisir le nœud à atteindre
                        Mémoriser le nœud atteint
                        Empiler (Pile_nœuds <- nœud atteint)
                   SINON
                        Dépiler
```

L'opération « Identifier les nœuds accessibles » permet de reconnaître toutes les directions possibles à partir d'un nœud. Par exemple, à partir du nœud B du réseau de la figure 4, on peut se diriger vers C ou D. Il faut songer à la manière dont on va explorer les directions possibles. C'est précisément la fonction de « Choisir le nœud à atteindre ». On peut décider d'explorer systématiquement toutes les directions possibles ou en choisir une selon un critère décidé à l'avance, comme la distance à franchir entre les nœuds. On peut par exemple décider d'aller toujours vers le nœud le plus rapproché de celui sur lequel on se trouve.

L'opération « Empiler » permet de conserver une trace de notre parcours en construisant une pile des nœuds dans l'ordre où on les a atteints. Ainsi, les nœuds du parcours A à G sont empilés dans « Pile_nœuds ». À la fin du parcours, le contenu de cette pile est [A, B, D et G]

ou [A, B, D, F et G] selon la direction prise au nœud D. Avec « Dépiler », on s'assure que l'on peut faire marche arrière si l'on se retrouve dans une impasse. En effet, « Dépiler » consiste à revenir au nœud précédent en biffant, des directions possibles, le nœud cul-de- sac. Il s'agit d'une stratégie typique de chaînage arrière. Nous allons voir maintenant au tableau 1 que les algorithmes de fonctionnement d'un moteur d'inférence sont analogues à la procédure de navigation dans un réseau.

Tableau 1 – Navigation dans un réseau et fonctionnement d'un moteur d'inférence

Navigation dans un réseau	Fonctionnement d'un système à base de règles
nœud	fait
arc	règle
réseau	base de règles
pile des nœuds	base de faits
départ	hypothèse ou fait à démontrer
destination	conclusion ou fait vérifié
véhicule	moteur d'inférence
Identifier les nœuds accessibles	Sélectionner les règles applicables
Atteindre un nœud	Appliquer une règle
Produire un parcours trouvé	Fournir des explications

4.2 Aperçu du fonctionnement d'un moteur d'inférence

Un moteur d'inférence est un programme qui établit des relations entre les connaissances codifiées dans une base de règles et les faits particuliers soumis par l'usager et entreposés dans la base de faits. Nous allons montrer que le MI s'acquitte de cette tâche en appliquant différents modes de raisonnement. Cependant, tous les mécanismes d'inférence reposent sur le postulat suivant :

On peut représenter les connaissances d'un domaine par une base de règles. Le format général d'une règle est :

Si [partie condition]

Alors [partie conclusion]

où « partie condition » et « partie conclusion » sont des listes d'expressions du type *[attribut] [prédicat] [valeur];* ces expressions sont reliées par les opérateurs *et, ou, non.*

On comprend que de telles expressions décrivent des *faits* en rapport avec un *objet* défini par une liste *d'attributs* qui peuvent prendre différentes *valeurs.* Il s'agit donc d'une représentation qui nous est connue. On veut maintenant une procédure de traitement qui s'applique à cette représentation. Quelle que soit la stratégie adoptée, sa logique générale de fonctionnement est construite sur la séquence des trois opérations suivantes :

- sélection des règles applicables au contenu de la base de faits,
- choix de l'une des règles applicables,
- application d'une règle sélectionnée.

La figure 5 montre le schéma d'exécution du cycle de ces opérations.

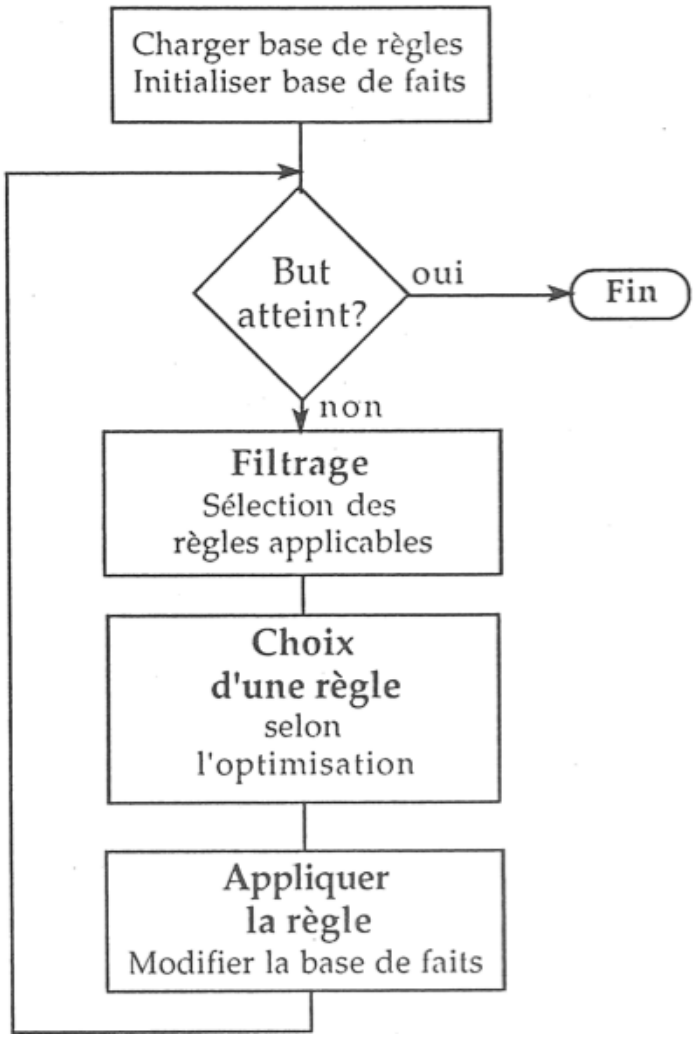


Figure 5 Cycle d'un moteur d'inférence.

La manière dont s'effectuent le filtrage des règles et le choix d'une règle dépend de la logique particulière de fonctionnement du MI. Il y a cependant un dénominateur commun à la plupart des mécanismes d'inférence : l'application de la règle de déduction appelée *modus panens.* Voici un énoncé de cette règle d'inférence logique :

> Soit A et B, deux propositions.
>
> si [A est vraie] et [A alors B] *alors* B est vraie.

À cet effet, *le petit Larousse* définit une *inférence* comme « une opération logique par laquelle on passe d'une vérité à une autre, jugée telle en raison de son lien avec la première ». Pour adapter cette définition au contexte des systèmes à base de règles, on n'a qu'à remplacer le mot « vérité » par le mot « fait ». On verra donc comment un MI peut établir des liens entre des connaissances en appliquant des règles. Pour chacun des mécanismes d'inférence de base, nous produisons un algorithme qui montre comment un MI traite les règles d'une base

de règles, les faits stockés dans la base de faits et les réponses de l'usager, pour solutionner des problèmes auxquels un SBC peut être confronté.

D'autre part, pour illustrer les algorithmes présentés, nous les mettons à l'épreuve dans une situation concrète. Ainsi, nous simulons le fonctionnement d'un MI dans un contexte simplifié, celui du comportement de certains indicateurs économiques. Voici donc une brève description du contexte qui servira à nos *simulations.*

> *Si les taux d'intérêt diminuent, alors les indices boursiers ont tendance à augmenter. Si, au contraire, les taux d'intérêt sont à la hausse, les indices de la bourse auront tendance à diminuer. D'autre part, lorsque le taux de change du dollar est à la baisse, cela crée une tendance à l'augmentation des taux d'intérêt. Inversement, si le taux de change augmente, les taux d'intérêt tendent à diminuer.*

Enfin, la banque centrale peut réagir sur l'économie en réduisant la masse monétaire et en augmentant le taux d'escompte. Cela a pour effet d'augmenter les taux d'intérêt.

Pour exploiter efficacement ces connaissances, on doit identifier les propositions en cause et leur associer une représentation symbolique concise qui fait ressortir les·attributs caractérisant notre exemple.

TI = +	les taux d'intérêt ont tendance à monter
TI = -	les taux d'intérêt ont tendance à diminuer
IB = +	les indices boursiers sont à la hausse
IB = -	les indices boursiers sont à la baisse
$ = +	le taux de change est à la hausse
$ = -	le taux de change est à la baisse
MM= -	la banque centrale réduit la masse monétaire
TE = +	la banque centrale augmente le taux d'escompte

On voit que ces expressions traduisent des faits à l'aide d'un couple attribut-valeur. Ainsi, l'expression TI= + veut dire que l'attribut TI prend la valeur +. D'autre part, il faut comprendre la dépendance mutuelle des attributs pour voir comment la variation d'un attribut comme le taux de change ($) peut affecter l'ensemble du système. À cette fin, on construit un arbre des attributs comme celui illustré à la figure 6. Cet arbre montre que la valeur de l'attribut TI résulte de la valeur de l'attribut $ *ou* de l'effet conjugué des valeurs de TE *et* MM. Enfin, la valeur de TI détermine celle de IB.

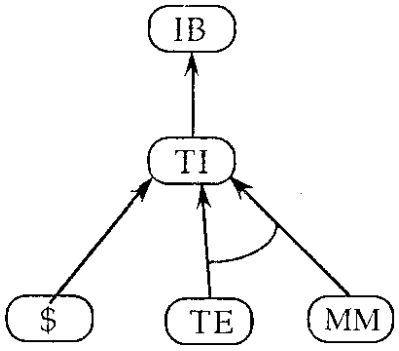


Figure 6 Arbre des attributs.

Cette approche nous permet de reformuler le paragraphe précédent en un ensemble de règles qui constituent une base de connaissances simplifiée sur l'économie. (Voir tableau 2.)

Tableau 2 – Base de règles simplifiée *économie*

Règle	Condition	Action
RI	Si TI = —	Alors 1B = +
R2	Si TI = +	Alors 1B = —
R3	Si $ = —	Alors TI = +
R4	Si $ = +	Alors TI = —
R5	Si MM = — et TE = +	Alors TI = +

4.3 Chaînage avant[3]

Cette stratégie de résolution de problèmes part des faits ou des hypothèses pour en déduire toutes les conséquences possibles. En chainage avant, le moteur d'inférence applique les règles dans le sens du *si* vers le *alors,* c'est-à-dire de la partie condition vers la partie conclusion. C'est généralement de cette façon que les règles sont appliquées lorsque l'on apprend des faits nouveaux et que l'on désire en déduire toutes les conséquences.

Simulation avec la base Économie

Vous apprenez que la banque centrale vient d'augmenter le taux d'escompte et de diminuer la masse monétaire. Vous utilisez un SBC pour déduire les conséquences de ces deux faits (TE = + et MM = -). Voici comment un système à base de règles fonctionnant en chaînage avant pourrait opérer dans une telle situation.

Étape 1

Au début, aucune règle n'est applicable puisque la base de faits est vide. Le système vous interroge au sujet des taux d'intérêt et du taux de change. Vous n'en savez rien. Il vous demande comment varie le taux d'escompte. Vous répondez qu'il augmente. Cela permet au moteur d'inférence d'ajouter le fait TE = + à la base des faits. Le MI examine une après l'autre les règles pour en trouver une dont la partie condition est dans la base de faits. Il n'en trouve aucune, mais la cinquième règle est en partie vérifiée puisque l'une de ces conditions (TE = +) est déjà dans la base de faits. Incapable de déduire lui-même un autre fait, le MI vous demande comment se comporte la masse monétaire. Vous répondez qu'elle diminue et cela permet au MI d'ajouter MM = - à la base de faits.

Étape 2

C'est alors que le processus de déduction par chaînage avant peut être amorcé, puisqu'une règle (R5) est maintenant applicable. Le MI applique la cinquième règle, ce qui l'autorise à ajouter la conclusion de cette règle (TI = +) à la base de faits. Ensuite, il marque la règle RS qui ne peut plus rien nous apprendre dans ce contexte.

[3] Quelques synonymes : déduction, inférence par les données, *forward chaining* et *data driven.*

Étape 3

Il consulte la première règle et constate que sa condition (TI = -) contredit l'un des faits de la base de faits (TI = +). Cela amène le MI à marquer la règle Rl qui ne pourra rien déduire. Par contre, la condition de la règle R2 est satisfaite. En appliquant R2, le MI ajoute le fait IB = - à la base de faits et marque cette règle. Le moteur d'inférence est forcé de s'arrêter, puisqu'il n'y a plus aucune règle applicable.

Au tableau 3, nous avons le contenu de la base de règles et de la base de faits *après* chaque étape du MI.

Tableau 3 Simulation du chaînage avant avec la base de règles *Économie*

Étapes	Règles non marquées	Pile des faits	Règles appliquées
1	R1 : TI = — → IB = + R2 : TI = + → IB = — R3 : $ = — → TI = + R4 : $ = + → TI = — R5 : TE= + et MM = — → TI = +	MM = — TE = +	aucune
2	R1 : TI = — → IB = + R2 : TI = + → IB = — R3 : $ = — → TI = + R4 : $ = + → TI = —	TI = + MM = — TE = +	R5
3	R3 : $ = — → TI = + R4 : $ = + → TI = —	IB = — TI = + MM = — TE = +	R2

Algorithme

Cette méthode consiste à empiler des faits déduits par l'application de règles ou obtenus de l'usager. Le moteur d'inférence cherche dans la base de règles, des règles dont au moins un fait de leur *partie condition* est déjà dans la pile des faits. Appliquer une règle reconnue vraie consiste à *empiler sa conclusion* dans la pile des faits et à marquer cette règle qui ne sera plus consultée puisqu'elle ne pourra plus rien nous apprendre. Une règle est déclarée fausse et également marquée lorsqu'un fait de sa partie condition contredit un fait de la pile des faits. Quand aucune règle n'est applicable, le MI pose une question à l'usager pour obtenir un fait qui lui permettrait d'en déduire de nouveaux. Le procédé continue tant que le dialogue peut se poursuivre avec l'usager et qu'il reste des règles qui ne sont pas marquées.

Les moteurs d'inférence comportent généralement une instruction de type « Fournir des explications » qui peut nous éclairer sur la manière dont sont faites les déductions par le MI.

Revenons à notre simulation avec la base *économie.* À l'étape 1, lorsque le système s'informe de la variation du taux d'escompte (TE), on pourrait répondre *pourquoi.* Ce à quoi le SBR répondrait « Je tente d'appliquer la règle 5: *Si* TE = + et MM= - *alors* TI= +. ». Après l'étape 2,

la commande *quoi* demande au système de produire un sommaire des faits connus. Il n'aurait qu'à fournir le contenu de la pile des faits, c'est-à-dire « TE = +, MM = -, TI= + ». À la fin du dialogue, on pourrait demander *comment* lB = -. La réponse du MI pourrait être « J'ai déduit lB = - en appliquant la règle 2 : *Si* TI= + *alors* lB = - ». On voit donc qu'un SBR peut donner des explications sur son propre fonctionnement à partir du contenu de la mémoire de travail, notamment celui de la pile des faits.

4.4 Chaînage arrière[4]

Cette stratégie de résolution de problèmes part d'une conclusion à démontrer pour identifier les conditions qui doivent être satisfaites. En chaînage arrière, le moteur d'inférence applique les règles dans le sens du *alors* vers le *si,* c'est-à-dire de la partie conclusion vers la partie condition. C'est généralement de cette façon que les règles sont appliquées lorsque l'on veut connaître les conditions requises à la réalisation d'un fait. C'est donc un mode de fonctionnement approprié à la recherche d'informations manquantes à la compréhension d'un phénomène.

Le langage Prolog en est un bel exemple, puisque sa partie essentielle est un moteur d'inférence qui fonctionne en chaînage arrière. Pour parvenir à démontrer une conclusion, les mécanismes de chaînage arrière identifient des buts intermédiaires qu'il faut démontrer pour atteindre un but. Pour y parvenir, ils traitent les règles, empilent les buts à démontrer, les conclusions intermédiaires déduites et les règles appliquées. Ce processus en cascade permet de vérifier toutes les conditions requises à la réalisation d'une conclusion. Cela implique parfois un retour en arrière ou une « remontée » dans la base de règles (*back-tracking*).

Le comportement d'un conducteur qui ne dispose pas d'une carte routière et qui tente par essais et erreurs d'atteindre un lieu est un bon exemple de *retour en arrière.* Dans ce cas, le conducteur doit trouver les lieux qui sont sur un chemin le menant à la destination cherchée. Parvenu à une intersection, il choisira une direction et continuera dans cette voie tant qu'il « sent » qu'il se rapproche du but. Dans le cas contraire, il reviendra en arrière pour trouver un lieu à partir duquel il pourra prendre une autre direction qui le conduira à sa destination. Le déplacement dans un labyrinthe, lorsqu'il est possible de garder une trace de son parcours pour refaire une partie du trajet en sens inverse, en est un autre exemple.

Simulation avec la base Économie

Vous voulez savoir dans quelles conditions vos actions en bourse risquent de perdre de la valeur. Vous utilisez le système à base de règles de l'exemple précédent, en chaînage arrière en partant de la conclusion « les indices boursiers sont à la baisse ». Le MI trouve les conditions à vérifier, c'est-à-dire quels faits doivent se produire pour arriver à la conclusion IB –. Pour faciliter la compréhension du procédé, la figure 7 montre la structure logique arborescente de la base *Économie* qui contient cinq règles exprimant huit faits.

[4] Quelques synonymes : régression, inférence par le but, *backward chaining* et *goal driven.*

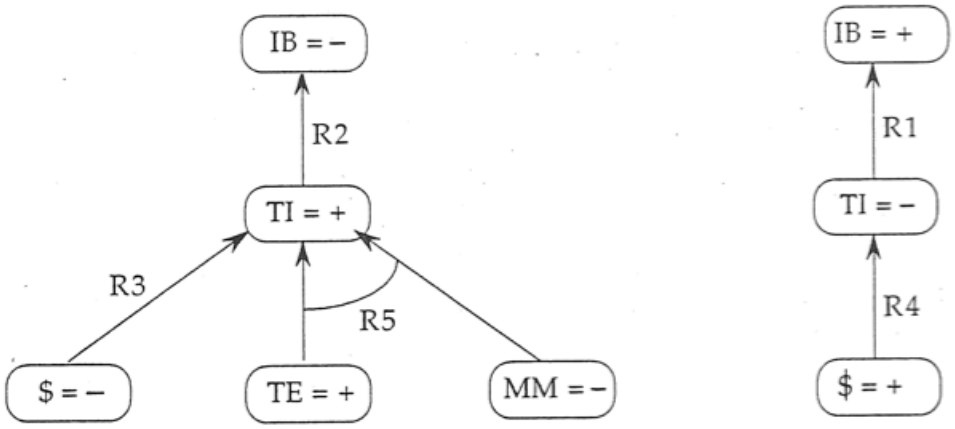


Figure 7 Structure logique de la base *Économie*.

La structure de la base de règles de notre exemple sur l'économie montre que les indices boursiers auront tendance à diminuer dès que les taux d'intérêt seront à la hausse. Ce fait se produit si le taux de change diminue ou encore si la banque centrale augmente le taux d'escompte et diminue la masse monétaire. La figure 3.7 montre que l'on peut arriver à la conclusion IB = - en appliquant la règle 2 après avoir obtenu TI = +. On peut parvenir à cette conclusion intermédiaire par deux chemins différents : en appliquant la règle 3 *ou* la règle 5.

Supposons maintenant que l'on ignore la tendance du taux de change. Par ailleurs, on apprend que la banque centrale vient d'augmenter le taux d'escompte et de diminuer la masse monétaire. Voyons comment un système à base de règles fonctionnant en chaînage arrière pourrait en arriver aux mêmes résultats que nous, à savoir que les indices boursiers auront tendance à diminuer.

Étape 1

Le MI sélectionne les règles 1 et 2, puisqu'elles ont l'attribut IB comme conclusion. Leur attribut condition TI devient le nouveau but à démontrer. Le moteur sélectionne les règles 3, 4 et 5, puisque leur conclu- sion porte justement sur la valeur de l'attribut Tl. La première règle sélectionnée R3 s'appuie sur l'attribut $ dans sa partie condition qui devient à ce moment le nouveau but à atteindre.

Étape 2

Il n'y a aucune règle applicable et le MI pose une question sur la tendance du taux de change ($). Puisque nous n'en savons rien, le MI élimine $ comme conclusion possible et marque les règles 3 et 4, car elles sont devenues inapplicables. Il reste à appliquer la règle 5 qui implique la démonstration à la fois de TE = + et de MM = -. Le système demande comment varie le taux d'escompte. On répond qu'il augmente. Le MI empile donc TE = + dans la base de faits. De la même manière, il vérifie la variation de la masse monétaire. Notre réponse permet au MI d'ajouter MM= - à la pile des faits et d'appliquer la règle 5.

Étape 3

Alors s'effectue la « remontée » pour ajouter TI = + à la pile des faits et l'enlever de la pile des buts, ce qui ramène l'attribut IB comme conclusion à démontrer. Le *retour en arrière* se poursuit. Le MI marque la règle 1 parce que sa condition (TI = -) contredit la conclusion de la règle 5. Enfin, il applique la règle 2 pour conclure à IB =.

Au tableau 4, nous avons le contenu de la base de règles et de la base de faits après chaque étape du MI.

Tableau 4 – Simulation du chaînage arrière avec la base de règles *Économie*

Étapes	Règles non marquées	Pile des faits	Pile des buts	Règles sélectionnées
1	R1 : TI = — → IB = + R2 : TI = + → IB = — R3 : $ = — → TI = + R4 : $ = + → TI = — R5 : TE = + et MM = — → TI = +		$ TI IB	R1 R2 R3 R4 R5
2	R1 : TI = — → IB= + R2 : TI = + → IB = — R5 : TE = + et MM = — → T I = +	MM = — TE = +	TI IB	R5
3		IB = — TI = + MM = — TE = +		R5 R1 R2

Algorithme

Cette méthode consiste à identifier et à empiler des buts intermédiaires qui doivent être vérifiés pour en arriver à un but final. Lorsqu'une conclusion intermédiaire est démontrée, elle est biffée de la pile des buts pour être empilée sur la pile des faits. Comme en chaînage avant, les faits sont déduits par application de règles ou sont obtenus de l'usager. Le moteur d'inférence inspecte la base de règles pour rechercher celles dont au moins un fait de leur *partie conclusion* est déjà dans la pile des buts. Appliquer une règle reconnue vraie consiste à *empiler sa conclusion* dans la pile des faits, à la retirer de la pile des buts et à marquer cette règle qui ne pourra plus rien nous apprendre. Ce sont l'empilement et le dépilement des buts qui permettent au moteur d'inférence d'avancer vers le but final à démontrer ou d'effectuer un retour en arrière (*backtracking*) pour vérifier d'autres buts intermédiaires qui pourraient conduire au but final. Une règle est déclarée fausse et marquée lorsqu'un fait de sa partie condition contredit un fait de la pile des faits. Quand aucune règle n'est applicable, le MI pose une question à l'usager pour obtenir un fait qui lui permettrait d'en déduire de nouveaux. Le procédé continue tant que le dialogue peut se poursuivre avec l'usager et qu'il reste des buts dans la pile des buts.

Pour voir comment opère un tel mécanisme d'inférence, on peut le mettre à l'épreuve dans le contexte d'une enquête, en vue de démontrer qu'un individu est coupable d'un crime. Dans un tel cas, on doit vérifier si le suspect a un alibi, des motifs sérieux de commettre le crime dont on le soupçonne et si des témoignages ou des faits incriminants peuvent confirmer notre hypothèse. Pour y parvenir, on construit d'abord une argumentation qui peut se formuler sous la forme de règles du type « *Si* le suspect a un alibi sérieux *alors* il est innocent >> et « *Si* le suspect était à l'hôpital lors du crime *alors* il a un alibi sérieu ». Ensuite, on interroge des témoins pour tenter de confirmer ou d'infirmer notre argumentation. Il peut arriver que nous soyons incapables de vérifier certaines conditions requises à la confirmation de nos

soupçons. Par ailleurs, il se peut qu'une piste s'avère infructueuse et que nous ayons à remettre en cause certaines hypothèses et à en reformuler de nouvelles, toujours dans le but de démontrer que le suspect est coupable. Il s'agit d'un processus typique de chainage arrière.

4.5 Variantes des mécanismes d'inférence de base

Nous venons de voir deux mécanismes fondamentaux, soit celui en chaînage avant et celui en chaînage arrière. Les algorithmes suggérés donnent une bonne idée du fonctionnement général d'un moteur d'inférence, mais ils ne rendent pas compte de tous les détails. Or, ce sont des « détails» comme la logique d'inférence qui distinguent les systèmes commercialisés à base de règles. Certains MI permettent les chainages avant et arrière, d'autres, un seul des deux. La plupart fonctionnent selon un amalgame et des variantes des deux mécanismes de base. Voici un aperçu de la diversité offerte par les éditeurs de SBC.

Chaînage mixte

Le chaînage mixte combine le chaînage avant et le chaînage arrière. Cependant, à une étape donnée du fonctionnement d'un MI en chaînage mixte, un seul mode d'inférence est appliqué. En général, un MI en chaînage mixte commence son travail en chaînage arrière pour accumuler des informations dans la base de faits. À cette étape, il fait souvent appel à l'usager pour répondre à des questions. Ensuite, il passe en chaînage avant pour déduire toutes les conclusions que lui autorisent les faits connus et empilés dans la base de faits. Le MI applique un critère arbitraire ou parfois un critère défini par le concepteur pour changer de mode d'inférence. Ainsi, une accumulation de cinq faits dans la base de faits peut déclencher le changement du chaînage arrière au chaînage avant.

Recherche en largeur ou en profondeur

La recherche désigne la manière dont la sélection des règles applicables s'effectue. Pour comprendre cette notion, il faut se rappeler qu'une base de règles peut être représentée par un réseau de faits reliés par des règles. Mais il existe un type de réseau particulier, l'arbre, comme celui· de la figure 4. La recherche en largeur ou en profondeur fait référence à la manière de relier les faits en appliquant des règles. En largeur, le moteur d'inférence sélectionne les règles qui permettent de déduire tous les faits appartenant à un même niveau de l'arbre, avant de déduire ceux d'un niveau inférieur. En profondeur, il applique les règles qui déduisent les faits qui « descendent d'une même branche » de l'arbre, avant de déduire ceux des branches latérales. La figure 8 illustre la structure arborescente d'une base de règles fictive. Les termes Fl, F2 et jusqu'à F10 désignent des faits alors que R1, R2 et jusqu'à R9 représentent des règles.

Supposons que l'on fonctionne en chaînage avant et que l'on connaisse les faits F1, F2 et F3. Par une recherche en profondeur, un moteur d'inférence applique dans l'ordre les règles R4, R5, R6, R8 et R9. À la fin de cette démarche, la base de faits contient F1, F2, F3, FS, F6, F7, F9 et F10. D'autre part, une recherche en largeur applique dans l'ordre les règles R3, R4, R5, R6, R7, RS et R9. Nous avions numéroté les règles en parcourant en largeur! Après l'application des règles en largeur, la base de faits contient les dix faits F1 à F10 exprimés dans la base de règles.

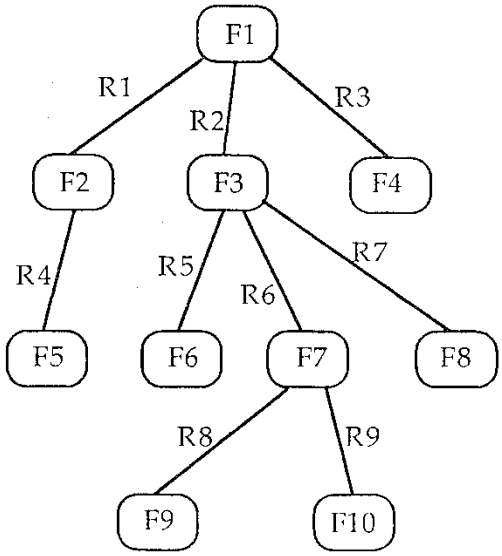


Figure 8 Base de règles parcourue en largeur ou en profondeur.

On constate que les résultats des deux méthodes de recherche diffèrent. En effet, une recherche en largeur déduit tous les faits accessibles à partir des faits connus. Par contre, une recherche en profondeur est plus sélective et plus performante en temps de traitement. Les distinctions entre ces deux variantes de recherche sont décrites plus loin dans la procédure *Choisir la règle à appliquer.*

5. Évaluation des mécanismes

5.1 Observations sur le fonctionnement d'un moteur d'inférence

Le dialogue SBC-usager est intégré au mécanisme d'inférence

La procédure *Recourir à l'usager* est très importante, puisque les réponses obtenues permettent souvent au moteur d'inférence de poursuivre son raisonnement. En effet, la participation de l'usager au fonctionnement du programme constitue un élément de dynamisme qui ajoute à l'intérêt des systèmes à base de connaissances. Pour que cette interaction SBR-usager soit profitable, il faut la concevoir soigneusement. Cela veut dire que les concepteurs d'une base de connaissances devront prévoir les questions que posera éventuellement le MI, lorsqu'il sera momentanément immobilisé dans son processus d'inférence. En règle générale, les SBR permettent d'associer une question à chaque attribut et de définir un choix de valeurs possibles lors de la définition d'une base de règles. Ainsi, lors de la définition de l'attribut nombre de cylindres pour l'objet voiture, on pourra lui associer la question et le choix de réponses suivants : « Vous désirez un moteur de combien de cylindres? [4, 6 ou 8?] » Il est clair que l'on tiendra compte du choix des valeurs d'attributs dans la rédaction des règles pour traiter toutes les situations prévisibles.

Dans nos descriptions d'algorithmes et dans nos simulations, nous avons mentionné la fonction du module d'explication, une possibilité intrinsèque au fonctionnement des moteurs d'inférence découlant de la *gestion d'une base de faits comme une pile.* À n'importe quel moment du dialogue avec le SBR, l'usager peut demander des explications sur la manière dont s'effectue le processus d'inférence des connaissances. Pour l'informer sur ce sujet, le

système n'a qu'à afficher le contenu de la base de faits en dépilant son contenu et ce, sans rompre la logique d'inférence. C'est la fonction de la procédure *Fournir des explications.* De cette manière, l'usager peut savoir quelle règle est appliquée, comment un fait a été déduit, où en est rendu le MI dans ses déductions successives et même quel fait pourrait être déduit si tel autre survenait. Nous ne saurions trop insister sur la valeur pédagogique, voire heuristique, du module d'explication. Il s'agit d'un moyen de mettre l'usager en contact avec la logique de traitement des connaissances d'un sujet.

La base de connaissances évolue lors d'une session de travail

Le contenu de la base de connaissances change lors du dialogue avec l'usager. Au début de la consultation, aucun attribut n'a une valeur, tous les attributs sont « vides ». C'est par les réponses de l'usager et l'application de règles par le MI que des valeurs seront données à des attributs.

D'autre part, le nombre de règles applicables diminue au cours d'une consultation. En effet, le MI marque une règle dès qu'elle est déclarée vraie ou fausse. Une règle ainsi marquée demeure dans la base de connaissances, mais ne sera plus consultée par la suite dans le dialogue avec l'usager. On peut donc voir que plus le dialogue SBC-usager progresse, le processus de recherche des règles dans la base devient plus rapide. Ce fait est très important lorsqu'une base contient un grand nombre de règles, ce qui est le cas des SBC spécialisés dans un sujet vaste.

Le moteur d'inférence est autonome mais il peut solliciter de l'aide

L'usager définit les hypothèses ou détermine le but final d'une consultation. Le MI part de ces faits à démontrer et se fixe lui-même des objectifs intermédiaires à atteindre dans la poursuite de son raisonnement. En chaînage arrière, le MI ramène le problème principal (la démonstration du but final) à des sous-problèmes (la démonstration de buts intermédiaires). Ce fait est caractéristique d'un comportement intelligent. Toutefois, le MI a recours à l'usager pour l'aider à progresser dans sa démarche. Le mécanisme de déduction peut avorter si le MI ne trouve pas de question à poser à l'usager en vue de trouver la valeur d'un attribut. Il est donc essentiel que les concepteurs d'un système à base de règles prévoient un dialogue approprié pour assister le MI dans son fonctionnement.

Les connaissances sont indépendantes du moteur d'inférence

Voilà une réalité qui découle de la séparation très grande entre les données et les programmes dans un système à base de règles. La logique du traitement des règles est intégrée dans le MI et est ainsi complètement dissociée de la base de règles. C'est pourquoi, on peut concevoir des bases de règles sur des sujets différents en ayant toujours recours au même moteur d'inférence. Avec les SBR, la notion d'indépendance des données et des programmes est encore plus accentuée qu'avec les autres catégories de logiciels.

Le MI est générique par rapport aux domaines qu'il peut traiter, puisque sa logique est valable pour n'importe quelle base de connaissances. Il y a toutefois une restriction : toutes les connaissances d'un sujet doivent être codées sous la forme de règles dont le format est :

Si [partie condition]

Alors [partie conclusion]

Étant donné la généralité du moteur d'inférence nous pouvons l'utiliser dans plusieurs contextes. En effet, on peut changer de problème et conserver le même MI, car celui-ci peut être spécialisé dans plusieurs sujets. Pour changer le domaine d'un SBC, il suffit de changer le contenu de sa base de connaissances tout en conservant le même MI.

La base des faits alimente le module d'explication

Le moteur d'inférence gère une seule structure de données dans la base de faits : la pile. Ceci lui permet de reconstituer le déroulement d'une consultation avec l'usager, puisque la base de faits contient une image du dialogue SBC-usager. On dit qu'un SBC peut expliquer et justifier son propre fonctionnement. En effet, le contenu de la base de faits permet au MI :

- d'expliquer pourquoi telle question est posée à l'usager (*pourquoi*)*;*
- d'expliquer comment un fait est déduit (*comment*)*;*
- de faire un sommaire des faits déduits à n'importe quel moment de la consultation (*quoi*)*;*
- d'extrapoler en prédisant quelles déductions seraient possibles si telle réponse était fournie par l'usager (*quoi si*)*.*

5.2 Comparaison et synthèse des méthodes d'inférence

Les mécanismes d'inférence en chaînage avant ou en chaînage arrière se ressemblent beaucoup. En effet, il suffit de constater qu'ils « proviennent » tous deux d'algorithmes généraux présentés aux sections 4.1 et 4.2. D'ailleurs, toutes les variantes d'inférence ont en commun les procédures *Sélectionner les règles applicables, Choisir la règle à appliquer* et *Appliquer une règle.* Ce sont les détails de chacune de ces procédures qui expriment les particularités de chaque mode d'inférence.

En se reportant aux algorithmes présentés aux sections 4.3 et 4.4, on constate que ce sont principalement les procédures *Sélectionner les règles applicables* et *Choisir la règle à appliquer* qui distinguent le mode en chaînage avant de celui en chaînage arrière. En effet, la sélection des règles applicables se fait à partir des conditions en chaînage avant ou à partir des conclusions en chaînage arrière. D'autre part, le choix d'une règle se fait par un mécanisme de contrôle qui peut fonctionner selon l'une des consignes suivantes :

- recherche de toutes les solutions possibles;
- arrêt lorsqu'une première solution est trouvée;
- application des règles en respectant des priorités;
- application des règles en largeur ou en profondeur.

Cette énumération des consignes montre que la logique de la procédure *Choisir la règle à appliquer* est fondamentale : elle détermine la performance du mécanisme d'inférence et les résultats qu'il produira. Nous y reviendrons plus loin.

On a mentionné que chaque mode d'inférence est adapté à un contexte d'opération particulier. Ainsi, le chaînage avant est particulièrement efficace lorsque l'on connaît déjà un ensemble de faits sur un sujet et que l'on désire en déduire les conséquences possibles. Par contre, le chaînage arrière semble approprié lorsque l'on dispose de peu d'informations sur un sujet et que l'on désire connaître les conditions requises à la réalisation d'un événement. Le tableau 5 suggère des critères de comparaison entre les deux modes d'inférence de base.

Tableau 5 – Chaînage avant et chaînage arrière

Particularité	Chaînage avant	Chaînage arrière
Sens de l'inférence	de la condition vers la conclusion	de la conclusion vers la condition
Raisonnement	progression	régression
Situation appropriée	des faits sont déjà connus	peu de faits sont connus
Choix des règles	recherche en sens unique	retour en arrière (*backtracking*)
Contenu de la mémoire de travail	Pile de faits Pile de règles	Pile de faits Pile de règles Pile de buts

6. Optimisation de la performance d'un moteur d'inférence

On est en droit de s'interroger sur la performance d'un MI. Le terme performance ne se mesure pas seulement en temps d'exécution, mais fait également appel à la valeur des solutions obtenues. On pense ici à la capacité d'un programme de déduire et de découvrir des connaissances signifiantes. Cela fait directement référence à la puissance heuristique; ce qui nous amène à considérer la performance d'un SBR tant sur le plan quantitatif que sur le plan qualitatif.

6.1 Structuration des connaissances dans une base de règles

Si l'on veut améliorer le temps d'exécution du mécanisme d'inférence par le MI, on doit d'abord optimiser les procédures *Sélectionner les règles applicables* et *Choisir une règle à appliquer.* La recherche des règles s'effectue dans toute la base. Avec une base volumineuse, la recherche peut alourdir le fonctionnement du MI et peut ralentir le temps de réponse à l'usager. On estime, généralement, que le nombre de règles dans une base croît de façon exponentielle avec le nombre de valeurs d'attributs et de valeurs possibles pour chaque attribut. Ainsi, en supposant que chaque attribut prend en moyenne trois valeurs, une base de règles à cinq attributs pourrait compter jusqu'à 3^5 ou 243 règles alors qu'une base à dix attributs pourrait en contenir 3^{10} c'est-à-dire 59 049 règles. Comme le temps de recherche d'une règle est relié directement au nombre de règles dans une base, on doit rendre le mécanisme de sélection de règles très performant.

Une des stratégies d'optimisation repose sur le fait suivant : il faut regrouper les règles selon un critère qui permet de restreindre la recherche à un sous-ensemble de la base. Or, ce critère existe. En effet, le MI recherche une règle contenant, dans sa partie condition (en chaînage avant) ou dans sa partie conclusion (en chaînage arrière), un certain attribut auquel il tente d'affecter une valeur. La solution s'impose d'elle- même : il suffit de *grouper les règles qui comportent un même attribut* dans la partie condition ou dans la partie conclusion. En appliquant cette stratégie, une base de connaissances est structurée en plusieurs sous-ensembles de règles, chaque sous-ensemble ayant un attribut commun. La recherche d'une règle pourra alors se faire dans un nombre limité de règles. Ainsi, le temps de recherche et le rendement général du MI en seront améliorés.

Notre approche de structuration des attributs, des faits et des règles dans une base de règles s'appuie sur l'hypothèse suivante :

Le moteur d'inférence gère une structure de données de base : la *liste.*

D'abord, rappelons-nous que les piles maintenues par le MI dans la base de faits sont précisément des listes, puisqu'une pile est un cas particulier de la liste. Nous voulons étendre l'usage de la structure de liste à toutes les entités gérées par un MI notamment aux règles et aux attributs dans la base de règles. Commençons par les règles. Une règle est décrite par l'organisation suivante : *Si* [liste de conditions] *Alors* [liste de conclusions]. Au paragraphe précédent, on a suggéré de « chaîner » les règles entre elles. Cela revient donc à maintenir des *listes circulaires de règles.* Lorsqu'une règle est déclarée vraie ou fausse par le Ml, celle-ci sera marquée, c'est-à-dire retirée de la liste des règles à consulter pour ainsi réduire le temps de recherche dans la base. En ce qui concerne les attributs, il suffit d'en faire une autre liste! La fonction d'un MI sera donc d'affecter des valeurs à certains éléments de la liste des attributs. Dès qu'une valeur est affectée à un attribut, celui-ci est exclu de la liste des attributs qui restent à affecter.

Il ne reste qu'à relier les attributs et les règles. En se rappelant que les règles ayant un attribut commun forment une liste, il suffit de relier un attribut à sa liste de règles. Il est donc possible de structurer attributs, faits et règles en les reliant pour former des listes. Par ailleurs, on peut faire une analogie intéressante avec le contexte des systèmes de gestion de base de données (SGBD). En effet, on peut considérer qu'une base de règles contient deux entités : des attributs et des règles. Pour imaginer comment on peut relier ces entités, il suffit de penser qu'un attribut peut se retrouver dans plusieurs règles et qu'une règle peut exprimer l'état de plusieurs attributs, puisque les parties condition et conclusions ont des listes.

Autrement dit, à une règle, on peut associer plusieurs attributs et à un attribut, on peut associer plusieurs règles.

La figure 9 illustre ce point de vue. On peut voir que les attributs Al, A2 et jusqu'à A7 sont reliés en listes dans l'entité Attribut. Il en est de même des règles Rl, R2 et jusqu'à R7 qui forment des classes par regroupements en listes dans l'entité Règle.

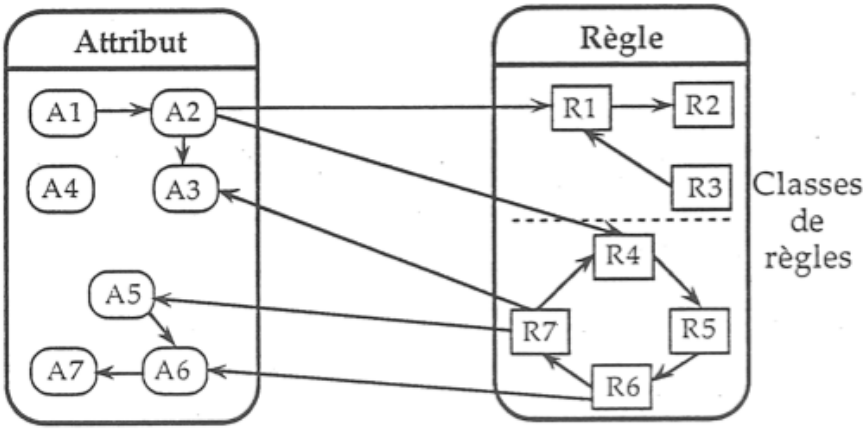


Figure 9 Structures présumées dans une base de règles.

Certaines « coquilles » commerciales de systèmes à base de règles offrent la possibilité d'associer une classe à chaque règle. Il suffit de fournir la valeur de ce champ lors de la création d'une base de règles. On peut associer un attribut particulier à une règle, la classe à laquelle on l'assigne; cela permet de segmenter une base de règles en classes. Or, en définissant des classes, on facilite le travail du MI, puisque l'on restreint la recherche des règles applicables dans une seule classe de règles.

6.2 Priorités d'application des règles

C'est un critère qui indique au moteur d'inférence comment choisir une règle à appliquer parmi plusieurs déjà sélectionnées. Le MI appliquera les règles en commençant par celle dont la priorité est la plus forte. La priorité de chaque règle est fixée par la personne qui définit une base. Pour ce faire, elle doit pondérer la valeur d'une règle selon ses possibilités de déduction par rapport aux autres règles. Ainsi, une règle qui permet de déduire des faits jugés très signifiants dans un contexte donné a une forte priorité. Par contre, une autre règle menant à une conclusion improbable ou de moindre valeur se voit assigner une priorité inférieure. De cette manière, on peut définir une *hiérarchie* dans une base de règles.

Pour définir une priorité de règle, il suffit de fournir une valeur numérique sur une échelle graduée (généralement de 1 à 100). Par exemple, prenons des règles appliquées par un SBR pour conseiller dans le choix de programmes d'études universitaires. Il apparaît normal de connaître d'abord la moyenne cumulative des candidats avant leur note, dans une discipline particulière. On pourra traduire cette intuition en priorité dans les règles de la manière suivante :

Priorité	Règle	
80	Si	moyenne cumulative < 85 %
	Alors	éviter les disciplines contingentées
60	Si	note en français > 90 %
	Alors	envisager des études littéraires

Il existe un critère largement répandu et accepté pour déterminer la priorité d'une règle : à une forte *probabilité* d'occurrence d'un fait dans un contexte donné doit correspondre une haute priorité d'un règle traitant ce fait. Cela veut dire que la priorité d'une règle est directement reliée à la probabilité que sa conclusion survienne et que sa fréquence d'application soit anticipée, en un mot, déterminer son utilité. Comme pour la classe, l'établissement des priorités se fait lors de la création d'une base de règles.

6.3 Métarègles

Les métarègles sont des règles qui décrivent la manière d'appliquer d'autres règles. Dans une base de règles, les métarègles forment une classe particulière. Elles contrôlent la sélection et le choix des autres règles. Il s'agit donc de règles qui sont très souvent appliquées par le MI pour contrôler le processus d'inférence. Les métarègles prennent la ·forme suivante :

	Si	telle condition
	Alors	appliquer telle(s) règle(s)
ou		
	Si	telle condition
	Alors	marquer telle(s) règle(s)
ou		
	Si	telle condition
	Alors	sélectionner telle classe de règle(s)

Voici une métarègle d'une base de règles qui peut alimenter un moteur d'inférence jouant aux échecs :

Si avantage de pièces
Alors sélectionner règles qui mènent à des échanges de pièces

6.4 Heuristiques

Le *petit Robert* définit l'heuristique comme « partie de la science qui a pour objet la découverte des faits ».·Dès les premiers travaux en intelligence artificielle, on a incorporé des heuristiques à des programmes. La caractéristique essentielle d'une procédure heuristique est de se diriger vers un but pour que chaque nouvelle « découverte » nous en rapproche. Cela revient à dire qu'une telle procédure peut évaluer chaque déduction effectuée au cours d'un processus d'inférence et juger si elle contribue à s'approcher ou à s'éloigner de l'objectif. Voilà pourquoi les programmes dits heuristiques sont dotés d'une *fonction d'évaluation.* Comme son nom l'indique, cette fonction évalue *l'état* des connaissances acquises à chaque étape d'inférence et le compare à l'état identifié comme l'objectif du processus.

Une fonction d'évaluation intégrée à la procédure *Choisir la règle à appliquer* fournit au mécanisme d'inférence, des critères d'application de règles. Elle peut indiquer que telle règle est plus pertinente que telle autre dans une situation précise. À titre d'exemple, on peut trouver systématiquement des solutions au jeu du taquin à 8 tuiles en utilisant une fonction d'évaluation comme celle présentée à la figure 2 du texte 1 de ce cours. Si on revient au contexte du jeu d'échecs, on peut imaginer qu'un programme « intelligent » doit disposer d'une procédure d'évaluation de positions pour choisir le meilleur coup parmi plusieurs possibles.

7. Traitement des connaissances incertaines

Jusqu'à maintenant, nous avons implicitement admis que les moteurs d'inférence ont un fonctionnement *déterministe,* c'est-à-dire qu'ils traitent des connaissances absolument certaines. Or très souvent, on ne peut affirmer que la conclusion d'une règle est une conséquence certaine de sa partie condition. En fait, la plupart du temps, un expert s'exprimera en *termes probabilistes* du genre : « Si on vous fait un pontage cardiaque, vous avez 90 % des chances que votre état s'améliore » ou « En entreprenant un forage dans cette région, j'évalue à 50 % la probabilité de trouver du pétrole et à 70 % celle de trouver du gaz naturel. »

Il importe de ne pas affirmer plus que ce que l'on croit raisonnable. Une façon d'appliquer cette règle de sagesse consiste à ajouter à chaque règle un facteur de probabilité ou *facteur de confiance.* Il exprime une certitude plus ou moins grande qu'un phénomène se produira si certaines conditions sont vérifiées. Ce facteur est un nombre entre 0 et 1. Ainsi, un facteur de confiance de 0,7 associé à une condition signifie que nous croyons qu'elle est valable à 70 % et un facteur de 1**,** qu'elle l'est à 100 %. On peut même attribuer un facteur de confiance à une règle. Étant donné que le but des facteurs de confiance est d'évaluer le degré de certitude ou la probabilité des conclusions d'un système, il nous faut des règles de calcul qui permettent de combiner :

- les facteurs de confiance des conditions d'une règle;
- ce résultat avec le facteur de confiance de la règle elle-même pour en déduire le facteur de confiance de la conclusion de la règle;
- entre eux les facteurs de confiance d'une conclusion déductibles de plusieurs règles.

Plusieurs méthodes de calcul des facteurs de confiance sont envisageables. La méthode la plus répandue met en pratique les règles suivantes :

a) Le facteur de confiance de la partie condition d'une règle est le plus petit (*minimum*) de ceux accordés aux conditions (reliées par l'opérateur et) qui la composent.

b) Si plusieurs conditions indépendantes (reliées par l'opérateur ou) conduisent à la même conclusion, le facteur de confiance de celle-ci sera le plus élevé (*maximum*) des facteurs de confiance des conditions.

La figure 10 montre comment on applique les deux premières règles en combinant des facteurs de confiance. Dans cette figure, « cl » désigne une conclusion et « cd » une condition.

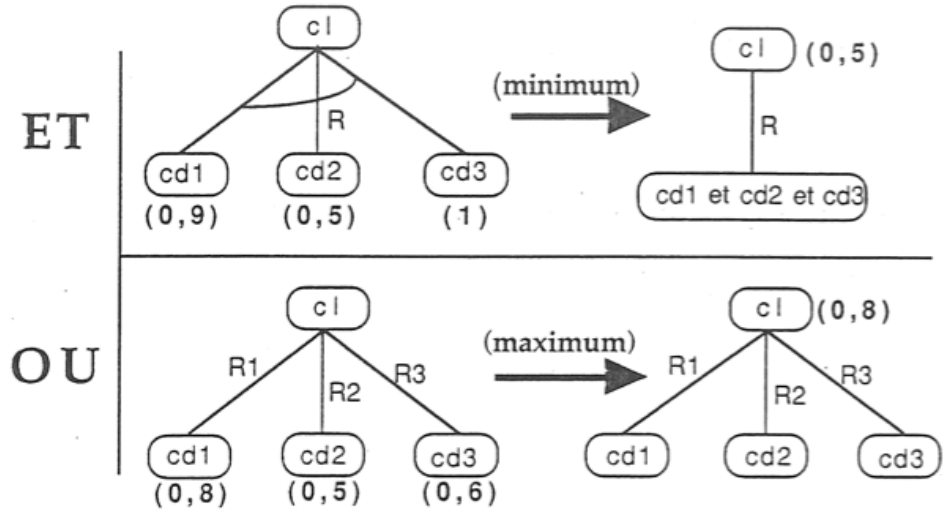
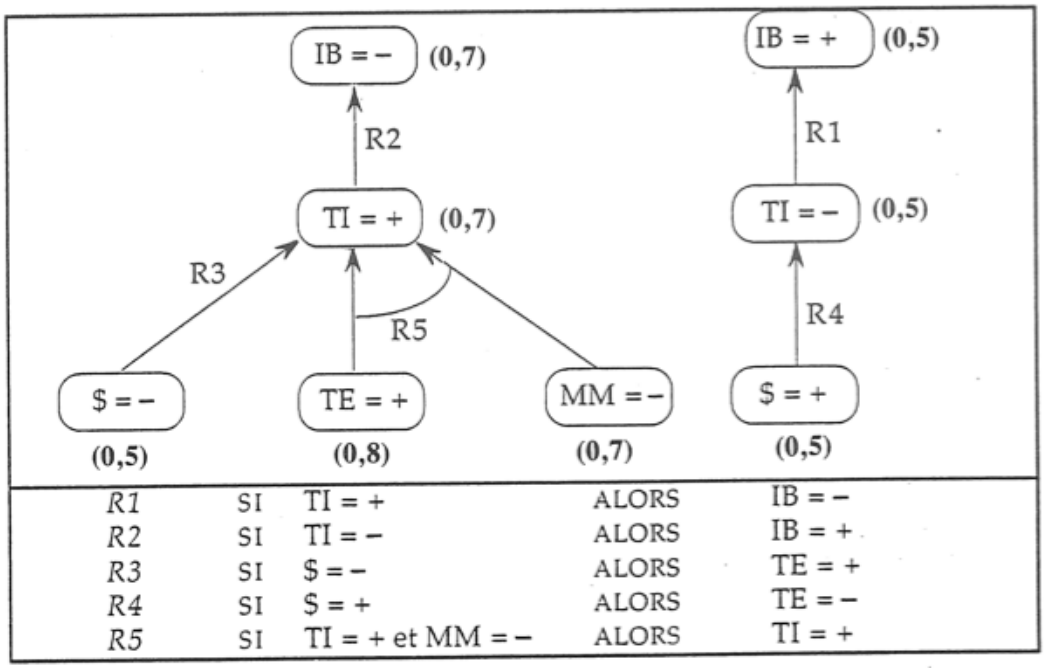


Figure 10 Combinaison des facteurs de confiance.

Appliquons ces critères à la base de règles *Économie* de la section 4. On a interrogé un expert en conjoncture économique pour définir les facteurs de confiance associés à chacun des quatre faits du niveau inférieur de l'arbre représentant cette base. Voyons maintenant, à la figure 11, ce que nous apprendrait cette base de règles en l'interrogeant sur les chances que les indices boursiers s'affaissent (IB = -).

Un moteur d'inférence en chaînage arrière applique d'abord la règle 2, ce qui l'amène à vérifier la tendance des taux d'intérêt (TI). La règle 3 est alors appliquée pour vérifier la tendance du taux de change ($). La condition de la règle 3 nous indique qu'il est certain à 50 % que le taux de change baisse ($ = -). On en conclut qu'il y a 50 % des chances que les taux d'intérêt augmentent (TI = · +). Il y a une autre façon de démontrer ce fait (TI= +). En appliquant la règle 5 dont les conditions TE = + et MM = - ont des facteurs de confiance de 0,8 et de 0,7 respectivement. On prend le minimum de ces deux facteurs, c'est-à-dire 0,7. Dans ce cas, on arrive à la conclusion que TI= + devrait survenir avec une probabilité de 70 %.



R1	SI	TI = +	ALORS	IB = –
R2	SI	TI = –	ALORS	IB = +
R3	SI	$ = –	ALORS	TE = +
R4	SI	$ = +	ALORS	TE = –
R5	SI	TI = + et MM = –	ALORS	TI = +

Figure 11 Les connaissances incertaines de. la base *Économie.*

Le fait TI = + est vérifié de deux façons différentes : par l'appli- cation de la règle 3 ou de la règle 5. On retient le plus élevé des facteurs de confiance de la conclusion déduite : 0,7, puis on applique la règle 2. Cela nous permet de conclure que l'indice boursier va diminuer (lB =) avec un facteur de confiance de 0,7. Autrement dit, le système prévoit que l'indice boursier a 70 % des chances de diminuer.

Il existe beaucoup d'autres types de moteurs d'inférence permet- tant de calculer la vraisemblance d'une conclusion. On peut attribuer, en particulier, un facteur de confiance à une règle. Dans le cas précédent, nous avons fait comme si chaque règle était certaine à 100 %, mais ce n'est pas toujours le cas.

Dans les systèmes où on attribue des facteurs de confiance aux règles et aux faits, le facteur de confiance de la conclusion d'une règle est le *produit* du facteur de confiance de sa partie condition multiplié par celui de la règle elle-même. Ainsi, dans l'exemple précédent, si la règle 1 a un facteur de confiance de 0,8, la conclusion lB = + a un facteur de confiance de 0,8 * 0,5 = 0,4. De cette manière, le facteur de confiance d'une règle peut atténuer celui de sa partie condition.

Conclusion

La réalisation des systèmes à base de connaissances est un pas vers l'objectif fondamental des travaux en intelligence artificielle : la mise au point de machines capables d'un comportement « intelligent ». De ce point de vue, on peut affirmer que les SBC, notamment les systèmes à base de règles, possèdent deux attributs caractéristiques d'un comportement intelligent :

- un mécanisme de raisonnement, le moteur d'inférence,
- une capacité d'explication de leur fonctionnement.

Une autre particularité des SBC est le haut degré de généralité des données qu'ils traitent. Celles-ci sont un amalgame de données factuelles (des faits) comme celles traitées par les systèmes « classiques » de gestion des données, mais également des connaissances déclaratives correspondant aux instructions de branchement dans un programme (des règles) qui décrivent la manière d'utiliser les connaissances factuelles. D'autre part, les systèmes à base de connaissances sont caractérisés par l'indépendance des faits et des règles par rapport aux programmes qui les traitent. Nous avons vu que les systèmes à base de règles sont un des derniers maillons d'une évolution des familles de logiciels marquée par la séparation progressive des données et des programmes.

L'opération d'un système à base de règles prend la forme d'un dialogue système-usager. Le système pose des questions et l'usager répond ou demande des explications sur l'état du processus de déduction. Pour satisfaire les requêtes de l'usager, le moteur d'inférence effectue deux opérations :

- Il établit des liens entre les connaissances dans la base de règles et les données d'un problème dans la base de faits. Par une sélection et une application des règles, il déduit des conclusions ou vérifie des faits.
- Il gère la base de faits comme une pile de données. Cela lui permet de conserver une trace de ses actions et ainsi, d'expliquer son fonctionnement.

Le fonctionnement du moteur d'inférence est contrôlé par la séquence de base suivante : sélection des règles applicables, choix d'une règle à appliquer et application d'une règle. On arrive à deux modes d'inférence distincts en adaptant, selon les besoins, les procédures de sélection et de choix des règles :

- en chaînage avant, lorsque nous connaissons un ensemble de faits desquels on voudrait déduire des conséquences;
- en chaînage arrière, lorsque nous désirons vérifier dans quelles conditions tel fait pourrait survenir.

Il existe des variantes de ces deux modes d'inférence de base qui optimisent le fonctionnement du MI, c'est-à-dire qui le rendent plus performant. La plupart des optimisations reposent sur une structuration des règles pour former des regroupements ou sur la détermination de priorité d'application des règles.

L'étape suivante sera d'apprendre et d'appliquer une méthodologie d'acquisition et d'implantation de connaissances en vue de les exploiter avec un système à base de règles.

À retenir

1. L'émergence des systèmes à base de connaissances (SBC) est caractérisée par la séparation progressive des données et des programmes dans un environnement informatique. On intègre la logique des programmes aux bases de connaissances auxquelles accèdent des programmes de plus haut niveau, les moteurs d'inférence.
2. On opère un SBC par dialogue : le SBC pose des questions et l'usager répond. Les réponses permettent ainsi au SBC d'avancer dans son processus de recherche de solutions du problème soumis par l'usager.
3. Un SBC est constitué d'un mécanisme d'inférence (moteur d'inférence), d'une base de connaissances, d'une base de faits et d'inter- faces avec les experts et les usagers.
4. Un SBC reproduit la démarche d'un expert. Il déduit en exploitant le contenu d'une base de connaissances et en interrogeant l'usager. Il communique ses conclusions à l'usager.
5. Le parcours d'une base de règles par un moteur d'inférence est analogue à celui d'un réseau routier par un << navigateur». Les mécanismes d'inférence basés sur des règles sont construits sur la séquence des procédures suivantes : sélectionner les règles applicables, choisir une règle à appliquer et appliquer une règle.
6. On reconnaît deux modes d'inférence de base : le chainage arrière et le chaînage avant.
7. Le chaînage avant est un mode d'inférence qui permet de déduire des conclusions à partir de faits considérés vrais par hypothèse.
8. Le chaînage arrière est un mode d'inférence qui permet de retrouver les faits requis pour qu'une conclusion soit réalisée.
9. Un SBC conserve en mémoire les opérations qu'il effectue. Cela lui permet de fournir des indications sur son propre fonctionnement. Ainsi, il peut justifier une question, faire connaître le cheminement logique qui l'a conduit à une conclusion et faire un sommaire de ses déductions. Il peut même explorer des éventualités.
10. On peut optimiser les algorithmes d'inférence de base pour les rendre plus performants par divers moyens : classes de règles, métarègles, coefficients de priorités des règles, fonctions heuristiques d'évaluations, facteurs de confiance.