3. SYSTEMES HYBRIDES NEURO-SYMBOLIQUES

Les Systèmes Hybrides Neuro-Symboliques (SHNS) sont des systèmes qui intègrent des approches de l'I.A. symbolique avec des approches de l'I.A. connexionniste. L'intégration de ces deux types d'approches peut être faite de différentes façons. On peut, par exemple, intégrer des connaissances issues de différentes origines et représentations, faire des traitements coopératifs, faire des transferts entre des modules symboliques et des modules connexionnistes, etc. Compte tenu de l'existence de tous ces différents modes d'intégration neuro-symbolique, nous allons présenter dans ce chapitre (voir section 3.2) une classification des SHNS avec des exemples de systèmes connus. Avant de passer à la classification des SHNS, nous allons tout d'abord présenter un bref résumé des raisons que nous amènent à les utiliser et des avantages liées à l'intégration de ces deux approches.

3.1 GENERALITES

Les deux approches, symbolique et connexionniste, présentent des propriétés particulières à chacun. Ces propriétés constituent un ensemble d'avantages et de désavantages liés au choix de l'utilisation de l'une ou de l'autre approche. Le Tableau 3 dresse, d'une manière assez simple et résumé, une liste des comparaisons des propriétés de ces deux approches [LAL 96, BOZ 95, OSO 95, GIA 92, TOW 91].

I.A. Symbolique	I.A. Connexionniste
☺ - L'insertion de connaissances théoriques sur le problème peut être faite d'une façon simple et directe. Il suffit de les expliciter et de les convertir dans le formalisme de représentation de connaissances utilisé.	☹ - Usuellement, on ne peut pas profiter des connaissances théoriques disponibles sur le problème traité. Il faut toujours des exemples pour pouvoir acquérir des connaissances.
☹ - Le traitement est séquentiel, les temps de réponse lors de la consultation du système sont longs.	☺ - Les réseaux sont composés d'une série d'unités de traitement de l'information qui peuvent opérer en parallèle, avec un temps de réponse très rapide.

☺ - L'insertion de connaissances (e.g. règles) peut être faite très rapidement une fois qu'elles ont été déjà traitées par les experts et/ou ingénieur de connaissances.	☹ - Le processus d'apprentissage peut être assez long puisque les poids (connaissances) sont adaptés petit à petit.
☹ - L'apprentissage n'est pas un processus à la base de ce type de systèmes. L'acquisition de connaissances se fait plutôt par explicitation. Par conséquence, on a le *problème du goulot d'étranglement* de l'acquisition de connaissances dans les systèmes experts.	☺ - L'apprentissage et la généralisation de connaissances à partir d'un ensemble d'exemples sont les points forts des approches connexionnistes. L'apprentissage est le processus de base des approches connexionnistes.
☺ - Les approches symboliques permettent d'obtenir des explications sur les réponses données par le système. On peut justifier les réponses basées sur le processus de raisonnement employé par le système et sur les connaissances codés dans sa base. Les connaissances sont codées dans un langage proche du langage naturel, et donc facilement interprétable.	☹ - Les réseaux sont des "boîtes noires", où les connaissances sont codées dans les poids et les interconnexions. On n'a pas accès à une forme compréhensible de ces connaissances, qui puisse être interprété directement par un être humain afin d'expliquer les réponses obtenues.
☹ - Usuellement, pour que le système puisse bien fonctionner, les connaissances théoriques doivent être à la fois correctes et complètes sur le domaine traité. L'approche symbolique n'est pas adaptée aux traitement d'informations approximatives ou incomplètes. L'utilisation de variables numériques pose des problèmes.	☺ - Les réponses du système se dégradent progressivement en présence d'une entrée bruitée. Les réseaux sont très adaptés au traitement d'informations approximatives et incomplètes. Les variables continues ne posent pas de problèmes particuliers par rapport au traitement connexionniste.
☺ - Les connaissances sont représentées par des règles et par des structures de données. Ce sont des connaissances de niveau supérieur.	☺ - Les connaissances codées dans les réseaux représentent bien les relations existantes entres leurs variables d'entrée.
☹ - Le développement d'un système expert est une tache difficile et assez longue.	☹ - La détermination de l'architecture et des paramètres d'un réseau connexionniste peut être une tache difficile et de longue durée.

Tableau 3 - Propriétés de l'approche symbolique et de l'approche connexionniste

On peut constater que l'approche symbolique et l'approche connexionniste sont tout à fait complémentaires. Les SHNS essayent d'exploiter cette complémentarité afin d'améliorer leurs performances et d'avoir des solutions plus robustes. De plus, l'intégration de ces deux approches permet de diminuer, sinon de résoudre complètement, certaines limitations de l'une ou de l'autre approche [MEM 97]. L'intégration neuro-symbolique va aussi nous permettre de nous adresser à un certain nombre de propriétés souhaitées pour les système d'apprentissage automatique, tels que les items décrits dans la section 2.4.2. Enfin, le choix du type d'intégration entre les deux approches va être lui aussi très important, puisqu'il va déterminer les principales propriétés du système hybride.

3.2 TYPES D'INTEGRATION ET CLASSIFICATION DES SHNS

Plusieurs classifications sur les modes d'intégration des systèmes hybrides ont été proposées [LAL 96, HIL 96b, ORS 95, MED 92]. Ces classifications sont plus ou moins semblables. Chaque auteur a essayé de mettre en évidence un aspect par rapport aux autres. Nous avons une interprétation plus simple des différentes classifications proposées. Pour nous, plutôt que d'essayer de créer une seule classification hiérarchique et globale des systèmes hybrides, le plus simple est de faire des classifications en ne considérant qu'une seule propriété à la fois. Nous allons donc reprendre les classifications proposées par d'autres auteurs [LAL 96, ORS 95, HIL 94a, MED 92] et nous allons les diviser en plusieurs groupes considérés séparément les uns des autres. Ainsi, comme pour la classification des différents types de réseaux connexionnistes, nous allons pouvoir classer un système SHNS en considérant séparément chacune de ses propriétés. Nous pensons qu'il est très difficile de classer tous les systèmes dans une structure hiérarchique puisque les systèmes partagent certaines propriétés, en raison de l'hybridation de ses composantes. Nous allons classer les SHNS selon les critères suivants :

- le type d'intégration (approche unifiée, semi-hybride, hybride);
- le degré de couplage (couplage faible, couplage moyen, couplage fort);
- le mode d'intégration (traitement chaîné, sous-traitance, méta-traitement, co-traitement);
- les transferts des connaissances ($S \Rightarrow C$, $S \Leftarrow C$, $S \Leftrightarrow C$);
- le type de représentation des connaissances (localiste, distribuée, combinée);
- le type de codage des connaissances (règles d'ordre 0, règles d'ordre 0^+, règles floues, règles probabilistes, prédicats, arbre de décision, réseaux PMC, réseaux à prototypes, etc);
- le type de méthode de raisonnement (chaînage avant, chaînage arrière, résolution, propagation d'activations - connexionniste);

- le mode d'acquisition de connaissances (apprentissage continu, apprentissage non-continu, apprentissage supervisé, apprentissage non-supervisé);

Les différentes classifications présentées ici doivent beaucoup au travaux développés dans le cadre du projet MIX [HIL 93] et particulièrement aux travaux de B. Orsier, Y. Lallement et M. Hilario.

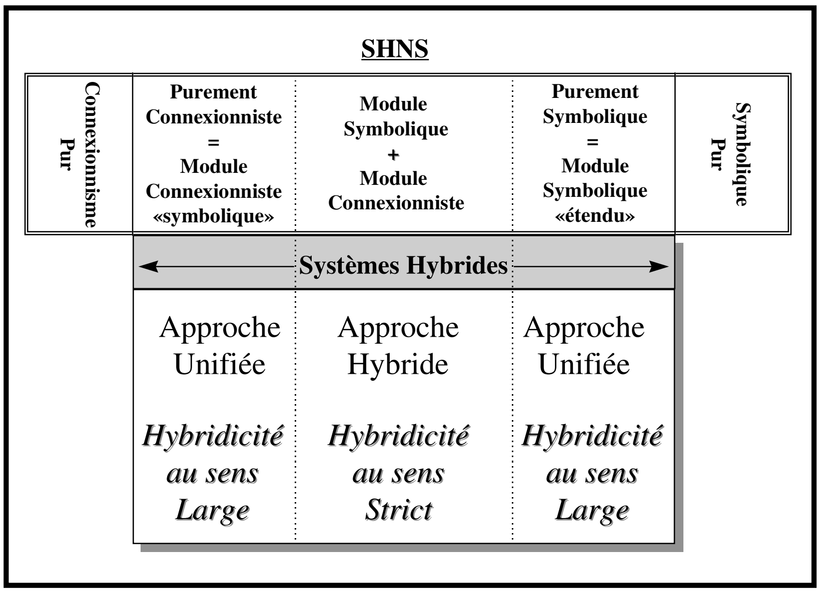


Figure 17 - Intégration Neuro-Symbolique

3.2.1 Types d'Intégration

L'intégration neuro-symbolique peut être classé dans deux grands groupes principaux, selon le "type d'hybridicité" de l'approche employée (voir Figure 17) :

• **Approche Unifiée** : ce type d'approche est aussi connu comme un approche "*pure*" ou comme une *hybridicité au sens large*. L'idée ici est d'intégrer tout dans un seul module, uniquement connexionniste ou uniquement symbolique, les deux types d'approches (symbolique+connexionniste). L'approche unifiée essaye d'integrer les proprietés des systèmes symboliques et des systèmes connexionnistes dans une solution unique qui les unifie. Cela donne l'origine à des *systèmes purement connexionnistes* (connexionnisme éliminatoire, le tout-connexionniste) et à des *systèmes purement symboliques* (le tout-symbolique). Les systèmes purement connexionnistes essayent de réaliser des processus d'inférence symbolique (e.g. résolution, unification de variables) à travers l'utilisation des réseaux de neurones. Ces systèmes sont aussi appellés CSP - *Connectionist Symbolic Processing*. Les systèmes purement symboliques essayent de réaliser des extensions des processus d'inférence symbolique, afin de leur permettre de raisonner sur l'incertain, sur des valeurs continues et approximatives, sur des probabilités, etc.

• **Approche Semi-Hybride** : ce sont des approches où l'on trouve des travaux relatifs aux traductions. Cela regroupe la *compilation* d'une base de règles dans un *réseau (insertion de connaissances*) et l'*explicitation* de règles à partir d'un réseau (*extraction de connaissances*). Certains systèmes sont dits semi-hybrides parce qu'ils ne réalisent qu'un des deux types de transferts de connaissances entre les modules symbolique et connexionniste. Pour nous, à partir du moment où un système offre la possibilité d'insérer et aussi d'extraire des règles, il devient un système hybride complet (*hybridicité au sens strict*), puisque on peut le coupler facilement avec un moteur d'inférence symbolique. Il est vrai qu'on doit aussi considérer le type des règles qui sont insérées et le type des règles extraites du réseau. Cette information est très importante puisqu'elle va nous permettre de savoir si l'on pourra par la suite utiliser facilement les règles dans un module symbolique. Elle nous permet aussi de savoir si l'on va pouvoir garder le même type de "capacité" de représentation de connaissances au niveau de l'insertion de règles ainsi qu'au niveau de l'extraction de règles d'un réseau.

• **Approche Hybride** : ce type d'approche est basé sur une véritable intégration d'au moins deux modules : un module symbolique et un module connexionniste. On peut aussi envisager une intégration à plusieurs modules symboliques et connexionnistes. Donc, dans ce type d'approche on retrouve au moins un système d'inférence symbolique qui travaille en coopération avec au moins un système d'inférence connexionniste. Cettte approche est connue comme l'*hybridicité au sens strict*.

Dans cette thèse, les références que nous faisons aux Systèmes Hybrides Neuro-Symboliques sont faites dans le contexte d'une *hybridicité au sens strict*, sauf si le contraire est indiqué.

3.2.2 Degré de Couplage

Le degré de couplage définit la force d'interaction entre les deux modules. La classification des différents degrés de couplage pourrait être faite selon une échelle continue pour représenter une progression d'une extrémité (pas de couplage) à l'autre (complètement intégrés). Nous allons présenter ici une classification en trois niveaux :

- **Couplage Faible** : dans ce type d'architecture, les divers modules (deux en général) sont reliés par une simple relation d'entrée/sortie, et les communications sont donc unidirectionnelles (*loosely coupled systems*). L'interaction entre les modules est nettement localisée dans l'espace et dans le temps.

- **Couplage Moyen** : dans cette catégorie, les interactions entre les modules sont plus souples, car elles sont bidirectionnelles; il ne s'agit plus simplement de relations d'entrée/sortie, et chacun des modules peut influencer dans une certaine mesure le fonctionnement de l'autre (*couplage étroit* ou *tightly coupled systems*). Par exemple, un module symbolique peut utiliser un réseau de neurones pour traiter le problème, mais en même temps il peut aussi influencer son apprentissage.

- **Couplage Fort** : dans ces systèmes, les modules partagent des structures de données communes, ou au moins l'un des modules a un accès direct à certaines structures de données de l'autre (*fully coupled*). Les connaissances et les données ne sont pas uniquement transférées, elles sont aussi partagées entre les modules qui utilisent des structures internes communes. Tout cela autorise des communications beaucoup plus souples et plus fines.

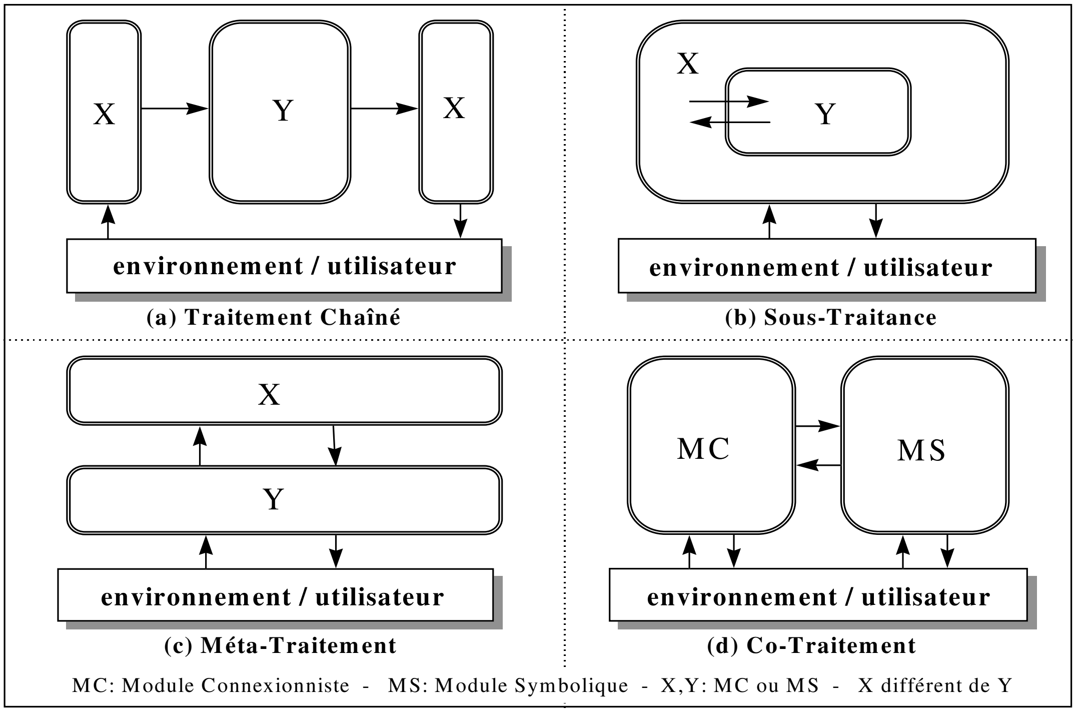


Figure 18 - Modes d'Intégration Neuro-Symbolique

3.2.3 Mode d'Intégration

Les modes d'intégration des systèmes hybrides (au niveau de leur structure) peuvent être classés dans les groupes suivants (voir Figure 18) :

- **Traitement en Chaîne** : le traitement est fait séquentiellement par un module, puis par l'autre (*chainprocessing*). Par exemple, un module connexionniste peut assurer un pré-traitement de données destinées à un module symbolique, ou l'inverse. Ce mode d'intégration est constitué par des interactions du type *pré-traitement* ou *post-traitement*, où les sous-systèmes sont reliés par une simple relation d'entrée/sortie.

- **Sous-Traitance** : le traitement principal est assuré par un des deux modules, qui utilise l'autre comme un prestataire de service pour résoudre des points particuliers (*subprocessing*). Un module principal symbolique peut par exemple appeler un module connexionniste à une certaine étape du traitement. Dans ce type d'interaction, un des modules est entièrement subordonné à l'autre, qui décide quand l'appeler et comment utiliser ses sorties. Le module de sous-traitance n'a pas de couplage direct avec l'environnement ou avec l'utilisateur.

• **Méta-Traitement** : un des deux modules joue le rôle du résolveur de problèmes de base, et l'autre a un rôle de méta-niveau (tel que la surveillance, le contrôle ou l'amélioration des performances) par rapport au premier (*metaprocessing*). Un module symbolique peut par exemple optimiser le fonctionnement d'un réseau connexionniste.

• **Co-Traitement ou Coopération** : les deux modules sont des partenaires égaux dans le processus et coopèrent pour résoudre le problème (*coprocessing*). Un des deux modules pourra par exemple s'intéresser à un aspect particulier du problème et l'autre module au reste du problème, ou encore les deux pourront proposer des solutions qui seront ensuite évaluées par un module tiers. Chacun peut interagir directement avec l'environnement.

Les différents modes d'intégration présupposent l'existence d'un *Module Symbolique* (MS) et d'un *Module Connexionniste* (MC) bien définis. Chaque module peut être facilement discerné de l'autre en raison de ces propriétés particulières.

3.2.4 Transferts de Connaissances

Les transferts de connaissances entre les modules symbolique et connexionniste (voir item types d'intégration - semi-hybride et hybride) peuvent être classés selon la direction des échanges réalisés :

• **Du Symbolique vers le Connexionniste** : les connaissances symboliques sont transférées du module symbolique et intégrés dans le module connexionniste ($S \Rightarrow C$). A notre avis, les méthodes de compilation de règles dans un réseau sont les plus intéressantes. Il existe aussi la possibilité de créer des bases d'apprentissage à partir des connaissances symboliques, mais dans cette approche on n'est pas sûr d'avoir à la fin de l'apprentissage un réseau qui intègre parfaitement toutes les connaissances symboliques (on est obligé de passer par une phase d'apprentissage).

• **Du Connexionniste vers le Symbolique** : les connaissances acquises par apprentissage dans les réseaux connexionnistes peuvent être explicités sous la forme de règles symboliques ($S \Leftarrow C$). L'extraction de règles à partir d'un réseau va permettre leur utilisation dans un module symbolique. Les règles obtenues à partir d'un réseau peuvent être classées dans différents groupes selon le type de codage des connaissances utilisé (voir item ci-dessous - types de codage).

• **Les Transferts Bilatéraux** : les connaissances peuvent être transférées dans les deux sens entre les deux modules symbolique et connexionniste ($S \Leftrightarrow C$). Les systèmes hybrides avec transferts de connaissances bilatéraux incluent normalement des mécanismes de compilation et d'extraction de règles à partir des réseaux. Une fois de plus, il faut se rappeler de l'importance du type de règles qui peuvent être insérées dans les réseaux ainsi que du type de règles qui en sont extraites. Un système hybride avec un bon couplage doit permettre l'insertion et l'extraction de règles avec le même pouvoir de représentation de connaissances.

Il faut préciser qu'ici nous avons classé les systèmes hybrides uniquement par rapport aux transferts de connaissances, tandis que dans les autres groupes de classification présentés nous faisons référence plutôt à des transferts entre les modules de toute sorte d'information (e.g.: paramètres, entrées/sorties, connaissances). Des exemples de méthodes d'insertion et d'extraction de connaissances à partir des réseaux connexionnistes vont être présentés plus en détails dans la section 3.3.

La capacité de faire des transferts de connaissances est une des caractéristiques les plus intéressantes des SHNS, puisqu'elle nous permet d'échanger les connaissances disponibles entre les modules symbolique et connexionniste. La compilation de règles nous permet d'insérer des connaissances initiales dans un réseau, ce qui devrait entraîner une optimisation du processus d'apprentissage. L'extraction de règles nous permet d'expliciter les connaissances d'un réseau qui par conséquent ne sera plus une boîte noire fermée aux yeux de l'utilisateur. Les règles extraites peuvent nous aider à mieux comprendre les réponses données par les réseaux et peuvent nous fournir des informations complémentaires sur le problème traité. Le module symbolique peut aussi profiter des connaissances extraites à partir d'un réseau pour améliorer sa base de connaissances.

Dans la suite de cette thèse, nous avons donnée beaucoup d'importance aux processus de compilation et d'extraction de règles dans les systèmes hybrides, car ces deux processus sont la clef de l'intégration entre les modules symbolique et connexionniste.

3.2.5 Type de Représentation des Connaissances

Les connaissances peuvent être représentées dans les réseaux connexionnistes (approche connexionniste unifiée) de trois façons différentes :

- **Représentation Localiste** : c'est un mode de représentation où chaque noeud du réseau est utilisé pour représenter un concept isolé. Usuellement, il est beaucoup plus facile d'incorporer des connaissances symboliques dans un réseau avec un mode de représentation de connaissances localiste. Chaque unité du réseau va correspondre à un concept. Les réseaux à base de prototypes (e.g. RBF) sont des modèles localistes.

- **Représentation Distribuée** : c'est un mode de représentation où tout un ensemble de noeuds du réseau est utilisé pour représenter chaque concept. La représentation de connaissances est faite d'une façon repartie : un concept est représenté par un ensemble d'unités, chaque unité peut servir à coder plusieurs données. L'apprentissage connexionniste amène plutôt à des représentations distribuées. Tel est le cas des modèles du type PMC utilisant l'apprentissage par rétro-propagation du gradient de l'erreur. L'intégration de connaissances symboliques par une représentation distribuée dans les réseaux connexionnistes se fait plutôt par apprentissage que par compilation.

- **Représentation Combinée** : certains systèmes peuvent intégrer les deux types de représentation dans un seul réseau. Un exemple de réseau avec une représentation combinée est celui des réseaux obtenus par compilation de règles (représentation localiste) et qui après passent par une phase d'apprentissage (représentation distribuée) afin de raffiner cette base de règles. Les systèmes hybrides qui travaillent d'un côté avec une représentation symbolique (localiste) des connaissances et de l'autre côté avec un réseau connexionniste à représentation de connaissances distribuée sont un autre exemple de l'approche combinée.

Nous parlons ici de la classification des types de représentation de connaissances seulement dans les approches connexionnistes unifiées puisque dans les approches symboliques la représentation de connaissances est essentiellement localiste. Dans les approches symboliques, une classification qui peut être intéressante à étudier est celle qui va identifier les types d'interrelations entre les différents concepts (e.g. concepts isolés, concepts avec des relations contextuelles, concepts fortement liés, concepts contradictoires).

3.2.6 Type de Codage des Connaissances

Les connaissances acquises par les systèmes hybrides peuvent être représentées de manière interne de différentes façons. Les principaux types de représentation sont : par des règles d'ordre 0, par des règles d'ordre 0^+, par des règles floues, par des règles probabilistes, par des prédicats de premier ordre, par des arbres de décision, par des réseaux PMC, par des réseaux à prototypes. Nous avons déjà discuté à propos de ces différentes types de codage (représentation) de connaissances dans le Chapitre 2. Dans la suite de ce chapitre, nous allons faire référence principalement aux groupes cités ci-dessus. Une grande partie des systèmes hybrides neuro-symboliques (hybridicité au sens strict) travaillent avec des règles symboliques d'ordre 0 ou d'ordre 0^+.

3.2.7 Type de Méthodes de Raisonnement

Les systèmes hybrides utilisent différentes méthodes d'inférence afin de résoudre les problèmes. Les approches unifiées sont basées sur une seule méthode d'inférence, tandis que les approches hybrides (hybridicité au sens strict) sont basées sur l'intégration de deux modules différents, ce qui normalement implique l'utilisation de deux types différents de raisonnement. Les principaux types de méthodes de raisonnement à considérer sont : le chaînage avant, le chaînage arrière, la résolution, le parcours d'un graphe (ou d'une arbre) et la propagation d'activations (approche connexionniste).

3.2.8 Mode d'Acquisition de Connaissances

Nous allons mettre en évidence certains modes d'acquisition de connaissances qui permettent de mieux discerner les différences existantes entre les systèmes hybrides que nous avons étudiés. Les deux premiers types d'acquisition de connaissances sont l'apprentissage en mode continu (incrémental), considéré par opposition à l'apprentissage en mode non-continu. Les systèmes avec un mode continu d'apprentissage doivent permettre l'ajout de nouvelles connaissances à des connaissances déjà acquises. Certains systèmes permettent seulement un raffinement des connaissances sans pourtant pouvoir acquérir des connaissances vraiment nouvelles sur le problème traité. Les deux autres types de modes d'acquisition de connaissance à considérer sont l'apprentissage supervisé et l'apprentissage non-supervisé. Ceux derniers sont aussi des traits importants pour l'identification des propriétés des SHNS.

3.2.9 Classification des SHNS

Nous allons présenter ici une brève liste de systèmes hybrides neuro-symboliques décrits selon leurs différentes propriétés. Notre idée n'est pas de dresser ici une liste exhaustive de tous les systèmes existants, mais seulement de revoir certains systèmes hybrides connus et de les classer dans les groupes que nous avons décrits. La plupart des systèmes que nous allons présenter ici sont décrits plus en détails dans les ouvrages de base que nous avons consulté sur les systèmes hybrides [LAL 96, HIL 96b, ORS 95, HIL 94a, GIA 92, MED 92].

Les systèmes que nous avons étudié sont les suivantes :

- **SYNHESYS :** le système hybride SYNHESYS (*SYmbolic and Neural Hybrid Expert SYstem Shell*) a été développé au laboratoire LIFIA, par A. Giacometti [GIA 92, ORS 94, ORS 95]. Ce système est composé de deux modules : un module connexionniste basé sur un réseau à prototypes de type ARN2 et un module symbolique d'inférence par chaînage avant et arrière. Le système SYNHESYS permet de consulter les deux modules en parallèle et de faire des transferts de connaissances entre eux dans les deux sens. Nous allons présenter ce système plus en détails dans la section 3.3.1.

- **ProBis :** le système hybride ProBis (*Prototype-Based Indexing System*) a été développé à l'IMAG à Grenoble dans le cadre d'une collaboration entre les laboratoires LEIBNIZ et TIMC, par M. Malek [MAL 95, MAL 96a, MAL 96b]. Ce système réalise une parfaite intégration entre un réseau à prototypes et un module de raisonnement fondé sur des cas. De cette façon, on peut profiter en même temps d'une représentation à base de prototypes (cas généralisés) et d'une représentation à base de cas (cas particuliers). Il permet d'obtenir un réseau à prototypes qui représentent les cas appris. Il permet aussi de garder les cas atypiques et frontières dans la mémoire du système CBR.

- **NESSY3L :** le système hybride NESSY3L (*NEuroSymbolic SYstem with 3 Levels*) a été développé au Lab. LIFIA/LEIBNIZ à Grenoble, par B. Orsier [ORS 95, ORS 97]. Ce système est composé de trois niveaux (niveau symbolique, niveau neuro-symbolique et niveau neuronal), à la différence de la plupart des systèmes hybrides qui n'ont qu'un module symbolique et un autre connexionniste. Avec l'utilisation de plusieurs niveaux, on augmente la simplicité des interfaces et la modularité du système. Ainsi, on peut faire une meilleure division des tâches entre les composantes du système hybride. Le niveau supplémentaire est basé sur le codage de règles dans un 'réseau versatile'. Ce système a été applique à des études d'évitement d'obstacle en robotique.

• **INSS :** le système hybride INSS (*Incremental Neuro-Symbolic System*) a été développé au Lab. LEIBNIZ à Grenoble, par F. Osório [OSO 95, DEC 95, DEC 96, OSO 97, AMY 97]. Ce système a été conçu à partir des idées de base des systèmes SYNHESYS et KBANN. Il est composé par deux modules (symbolique et connexionniste) et permet le transfert des connaissances dans les deux sens. Il met en pratique une méthode d'acquisition constructive de connaissances. Nous allons présenter ce système plus en détails dans le Chapitre 4.

• **KBANN :** les réseaux KBANN (*Knowledge Based Artificial Neural Networks*) ont été développés à l'Université de Wisconsin-Madisson aux Etats-Unis, par G. Towell [TOW 91, TOW 93, SHA 95]. Nous allons présenter ce système plus en détails dans la section 3.3.2.

• **SCANDAL** : Le système SCANDAL (Symbolic-Connectionist Architecture for Neural Design and Training) a été développé au CUI à l'Université de Genève par M. Hilario, A. Rida, B. Orsier et M. Stückelberg [HIL 95c, STU 96, AMY 97]. Ce système utilise une approche similaire à KBANN et INSS, puisqu'il permet aussi l'insertion de règles dans un réseau de type PMC. Le réseau peut être entraîné avec des exemples, et il contient des méthodes d'extraction de règles. La particularité de cette approche est due au fait qu'il a été réalisé à l'aide d'une architecture modulaire composée de différents agents spécialisés. Les agents sont capables par exemple : de construire des arbres de décision et d'obtenir des règles par l'utilisation d'algorithmes comme ceux de C4.5 [QUI 93] ; de construire des bases d'exemples artificiels à partir de règles ; de coder des règles symboliques dans un réseau ; de réaliser l'apprentissage des réseaux avec les algorithmes RPROP [RIE 93], Cascade2 [FAH 95], ou même SCG (Scaled Conjugate Gradient) [MOL 90]. Ainsi, le système SCANDAL offre-t-il plusieurs possibilités d'échange de connaissances entre ses composantes avec l'aide d'un agent superviseur.

• **MACIE :** le système MACIE (*Matrix Controlled Inference Engine*) a été développé par S. Gallant [GAL 88, GAL 93, GAL 95, BOZ 95, NIK 97]. Ce système permet de coder des connaissances symboliques dans un réseau, de les raffiner, puis d'en extraire des règles. Le réseau est composé par des unités qui sont identifiées par les labels des concepts qu'elles représentent. L'existence d'une connexion entre deux unités indique une relation de dépendance entre elles.

Pour pouvoir construire un réseau, les dépendances conceptuelles sont prises en compte afin de créer son architecture. Le réseau peut être complètement interconnecté en absence de connaissances suffisantes sur les dépendances. Après la détermination de la structure initiale du réseau, il est entraîné avec des exemples afin d'adapter ses poids (raffiner les connaissances initiales introduites dans le réseau). Une fois que ce type de réseau ne travaille qu'avec des unités à trois états (+1=Vrai, 0=Inconnu, -1=False), l'extraction de connaissances peut être faite facilement d'une manière presque directe.

- **KBCNN :** les réseaux KBCNN (*Knowledge-Based Conceptual Neural Networks*) ont été développés par L. Fu [FU 90, FU 92, FU 93, FU 94]. Ce type de réseau permet l'insertion d'une base de règles et son raffinement par apprentissage, d'une façon similaire aux réseaux KBANN et MACIE. Le réseau KBCNN utilise un algorithme d'apprentissage basé sur la Rétro-Propagation avec une procédure de groupement d'unités (*hidden unit clustering*). Il prévoit l'ajout d'unités au cas où le réseau reste bloqué pendant le processus d'apprentissage, ainsi que la possibilité d'extraire des règles à partir du réseau.

- **SETHEO :** le système SETHEO [LET 92, LAL 96] est un démonstrateur de théorèmes hybride, composé d'un démonstrateur classique (du type ProLog) et d'un module connexionniste servant à optimiser la recherche de preuves. La preuve consiste à trouver un chemin dans un arbre de recherche. Pour éviter l'exploration systématique de tout l'arbre de recherche, un réseau permet de choisir la branche de l'arbre la plus intéressante. Le module connexionniste sert à définir (guider) la stratégie de recherche du moteur d'inférence. Le module connexionniste est composé par un réseau de type PMC entraîné avec la Rétro-Propagation sur une base d'exemple de preuves de théorèmes.

- **WATTS :** le système WATTS (*Wastewater Treatment System*) a été développé par S. Krovvidy et W. Wee [KRO 92, HIL 96b]. Ce système essaye de réduire le temps de résolution d'un problème à travers l'emploi d'un module CBR couplé à un réseau de Hopfield. Le réseau doit déterminer la séquence de traitements à faire subir à des eaux usées afin de les épurer. Le module CBR sert à donner un état initial au réseau qui va permettre d'améliorer la convergence vers la solution du problème.

- **INNATE-QUALMS :** ce système sert à donner le diagnostic de pannes dans une usine de distillation [BEC 91, LAL 96]. Le système est constitué d'un module connexionniste, INNATE, et d'un module symbolique, QUALMS. Le processus principal est contrôlé par le système expert de diagnostic de pannes. Ce module symbolique appelle un ensemble de réseaux de type PMC afin de obtenir des indications sur les possibles pannes. Une analyse plus fine des réponses des réseaux et de l'état du système permet de confirmer le diagnostic des réseaux ou de proposer une solution alternative.

- **LAM :** le système LAM [MED 94, HIL 96a] est utilisé pour traiter un problème réel, il permet d'aider ses utilisateurs à concevoir divers verres de vitres de gratte-ciel. Le module symbolique (système expert basé sur des règles) aide l'utilisateur dans les tâches de classification des types de verres et du choix du type de verre le plus indiqué étant donné certaines spécifications d'ordre technique et d'ordre pratique. Le système expert appelle un réseau pour réaliser des tâches d'approximation des fonctions. Le réseau est entraîné avec une basé d'exemples de cas pratiques. Il sert à déterminer les paramètres liés aux propriétés sonores et solaires des verres, ce qui était plus difficile à concevoir à travers d'un système basé sur des règles.

- **DDT :** le système DDT (*Device Diagnostic Tool*) a été développé par Gutknecht, Pfeifer et Stolze [GUT 91, LAL 96]. Ce système est destiné à aider au diagnostic de pannes sur un bras de robot. C'est une approche qualifiée de coopérative, car il y a une collaboration entre un système automatique et un système humain. Dans DDT, les deux modules fonctionnent en parallèle pour donner chacun un avis sur le problème à résoudre. Une autre caractéristique intéressante du système DDT est l'amélioration des connaissances avec l'expérience. Les actions et les réponses du système sont stockées dans une base de cas qui est utilisée pour entraîner le réseau lorsque le système n'est pas utilisé.

- **ALVINN :** le système ALVINN (*Autonomous Land Vehicle in a Neural Natwork*) a été développé par D. Pomerleau à l'Université de Carnegie-Mellon - CMU [POM 92, CAU 90]. Une variante de ce système, proposée par Pomerlau [POM 91, HIL 96a], utilise plusieurs réseaux qui ont été entraînés afin de se spécialiser sur certains aspects du contrôle de la conduite autonome (e.g. conduite sur autoroute, conduite sur route secondaire, passage de carrefour, évitement de collision). Le système utilise un module symbolique basé sur des règles pour arbitrer laquelle des réponses des différents sous-modules va être prise en compte. De plus, le module symbolique est aussi responsable pour les tâches de plus haut niveau, tel que la planification de trajectoires.

- **SCRAPS :** le système SCRAPS a été conçu par J. Hendler [HEN 89, LAL 96]. Ce système utilise un raisonnement par analogie. Il est composé par un 'réseau sémantique à propagation de marqueurs', qui est utilisé pour encoder les connaissances dites dures. Il s'utilise d'un module connexionniste de type PMC pour encoder les similarités entre les objets du domaine. Ce réseau associe chacune des ses sorties à un ensemble de traits caractéristiques des objets et ses sorties sont aussi des noeuds du réseau sémantique. De cette façon, le réseau connexionniste va être appelé par le système symbolique afin de mesurer des similarités et ainsi fournir le lien entre l'exemple présenté et les concepts représentés dans le niveau symbolique.

- **RAPTURE :** le système RAPTURE (*Revising Approximate Probabilistic Theories Using Repositories of Examples*) a été développé par J. Mahoney et R. Mooney à l'Université du Texas [MAH 93, MAH 96]. Ce système sert à raffiner des bases de connaissances probabilistes. Il combine des méthodes connexionnistes avec des méthodes symboliques, en utilisant des techniques basées sur les systèmes KBANN et EITHER. La principale caractéristique de ce type de système est l'utilisation des règles et des réseaux avec des indications du degré de certitude (*Certainty Factors*). Le système RAPTURE utilise une version modifiée de la Rétro-Propagation (CFBP) pour raffiner les coefficients de certitude d'une base de règles probabilistes codées dans ce réseau. Il utilise aussi une méthode basée sur le gain d'information (voir la méthode ID3) pour permettre l'ajout de nouvelles règles/unités. Une fois terminé l'apprentissage, les règles peuvent être observées directement dans la structure du réseau.

- **CONSYDERR :** le système CONSYDERR (*CONnectionist SYstem with Dual representation for Evidentiual Robust Reasoning*) a été développé par R. Sun [SUN 91, SUN 95a, SUN 95b, ORS 95]. Ce système est utilisé dans le cadre du raisonnement de sens commun. Pour reproduire cette forme de raisonnement, un système hybride unifié (connexionniste) est réalisé avec deux niveaux : un niveau localiste, qui est une sorte d'implémentation connexionniste d'un système expert ; et un niveau distribué, qui apporte les propriétés purement connexionnistes. Ainsi, le système intègre dans une architecture connexionniste des représentations localistes et distribuées des connaissances, et des processus unifiés de traitement de règles et de similarités.

- **RUBICON :** le système RUBICON a été développé par T. Samad [SAM 92]. Ce système est une implémentation connexionniste d'un système expert basé sur des règles d'ordre 0. C'est un système assez simple qui permet le codage d'une base de règles dans un réseau et son activation. Il possède certaines limitations telles que : le manque d'une méthode pour l'adaptation des poids (ils sont codés en dur) et l'utilisation uniquement de valeurs discrètes (binaires). Une fois que les connaissances sont codés directement sur le réseau, l'interprétation des règles symboliques peut être faite directement sur les unités.

- **RAAM :** les réseaux RAAM (*Recursive Auto-Associative Memory*) ont été crées par J. Pollack [POL 90, ORS 95]. Ce type de réseau est une implémentation d'un approche hybride unifiée. Il permet d'obtenir des représentations de structures réduites de données complexes sur lesquelles on peut faire des inférences. L'idée derrière ce type d'approche est l'étude de la représentation de structures de données récursives et de taille variable (e.g. tableaux, listes, arbres) par des réseaux connexionnistes.

- **BoltzCONS :** les réseaux BoltzCONS ont été développés par D. Touretzky [TOU 89]. Ils permettent de créer et manipuler dynamiquement dans un réseau des structures comme des piles, des arbres ou plus généralement des graphes orientés. Le but ici est le même des RAAM, c.-à-d. étudier les possibilités d'un traitement symbolique par des réseaux connexionnistes.

- **SHRUTI** : les réseaux SHRUTI ont été développés par L. Shastri et V. Ajjanagadde [AJJ 91, SHA 93, HAY 96]. Ce type d'approche permet de créer des réseaux sémantiques connexionnistes dans lesquels sont codés des larges bases de connaissances. Les réseaux SHRUTI permet l'implémentation d'un mécanisme de *'dynamic variable binding'*.

- **EBNN :** l'algorithme EBNN (*Explanation-Based Neural Network Learning*) a été introduit par T. Mitchell et S. Thrun [MIT 93, MIT 97]. Cet algorithme intègre une approche basée sur les réseaux connexionnistes avec une approche EBL. De la même façon que les méthodes EBL, l'algorithme EBNN utilise la théorie sur un domaine pour guider l'apprentissage d'un concept. Cela est fait à travers des explications et des analyses obtenues à partir des exemples d'apprentissage. La méthode d'apprentissage est similaire à la Rétro-Propagation, où l'on utilise un algorithme qui s'appelle TangentProp.

Le Tableau 4 présente la liste des systèmes hybrides neuro-symboliques, ainsi que leurs propriétés particulières.

3.2.10 Discussion sur les SHNS

Après avoir étudié les différents systèmes SHNS et leurs propriétés, nous nous sommes retrouvés face aux questions suivantes :

- Quel est le meilleur type d'approche à utiliser : l'approche unifiée, l'approche semi-hybride ou l'approche hybride ?
- La représentation connexionniste localiste est-elle plus intéressante que la représentation distribué, ou vice versa ?
- Parmi les systèmes étudiés, en existe-t-il un qui regroupe les propriétés les plus importantes ?
- Comment doit-on coupler les modules d'un système hybride ?

Toutes ces questions sont encore très ouvertes à la discussion, puisque il n'y a pas de consensus sur le type 'idéal' de système hybride. Puisque nous n'avons pas les moyens de répondre précisément aux questions présentées ci-dessus, nous avons donc fait un certain nombre de choix vis-à-vis des propriétés que l'on aimerait avoir dans notre système. Les choix qui ont été faits portent principalement sur les aspects présentés dans la section 2.4.2. Nous essayons aussi de mieux exploiter les complémentarités entre les systèmes symboliques et connexionnistes, tel que nous les avons présenté dans le Tableau 3.

Etant donné les aspects considérés, nous avons choisi de ne pas utiliser de méthodes portant sur des approches du type unifiée. Nous nous sommes plus intéressés aux approches hybrides au sens strict, composées par deux modules (symbolique et connexionniste) et offrant la possibilité de faire des transferts dans les deux sens. Ce type d'approche inclut d'une certaine façon les approches semi-hybrides. Nous sommes de l'avis qu'avec une approche hybride au sens strict on doit pouvoir mieux exploiter la complémentarité entre ces modules, et ainsi pouvoir travailler auusi bien à un niveau symbolique qu'à un niveau sub-symbolique. Il est vrai que l'un des points-clefs de la discussion entre approches unifiées et hybrides se situe autour des capacités de représentation de connaissances et d'inférence sur les données. Nous essayons de garder le meilleur de l'approche symbolique ainsi que meilleur de l'approche connexionniste, sans pour autant essayer de les unifier, car cela peut amener à l'imposition de restrictions sur l'une ou l'autre approche.

Plusieurs systèmes sont à la source de nos inspirations : SYNHESYS, KBANN, MACIE, KBCNN, DDT et RAPTURE. Les points communs entre ces systèmes sont la possibilité de faire des transferts de connaissances entre un module connexionniste et un module symbolique, ainsi que le principe de fonctionnement orienté vers une acquisition incrémentale de connaissances. Ces systèmes permettent de réaliser un raffinement d'une théorie sur un domaine (*connectionist theory refinement*).

Nous avons aussi fait un choix sur le type de réseau à utiliser, les réseaux de type PMC. Notre intention étant d'utiliser des modèles assez connus (et reconnus) par la communauté des chercheurs qui travaillent dans le domaine des systèmes connexionnistes, nous avons délibérément évité d'utiliser des réseaux plus particuliers tels que les réseaux à coefficients de certitudes, ou des modèles plus spécifiques tels que les modèles employés dans des systèmes comme CONSYDERR, SHRUTI, RAAM ou EBNN. Notre choix des réseaux PMC peut être justifié par les point suivants :

- Notre but principal était d'arriver à la réalisation d'un système hybride fonctionnel et surtout simple à utiliser. Nos sources d'inspiration sont surtout les SHNS qui possèdent ces deux caractéristiques;

- Les réseaux à coefficients de certitude restent un cas à étudier plus en détails. L'utilisation de ce type de réseau a été critique pendant longtemps [MAH 96] et nous pensons que ce type de formalisme doit être encore mieux analysé avant de proposer des SHNS. Nous avons choisi de ne pas nous lancer sur ce type d'approche qui reste ouvert à l'étude dans des recherches futures;

- Certaines réalisations d'approches connexionnistes unifiées semblent trop se rapprocher des représentations de connaissances au niveau symbolique et pour nous il n'est pas sûr de pouvoir garder toujours les avantages et particularités de l'approche connexionniste (e.g.: la capacité de manipulation de variables numériques et le traitement de données incertaines et approximatives). Une approche comme celle des réseaux CONSYDERR semble être assez intéressante, mais peut-être un peu trop complexe à mettre en place. Cette approche est aussi une des voies de recherche qui sont intéressantes à suivre dans des recherches futures;

- Les réseaux récurrents sont sans aucun doute un domaine à exploiter. Malheureusement, à l'heure actuelle il nous semble que l'utilisation de ce type de réseaux dans des systèmes SHNS n'est pas assez développée et il reste encore beaucoup de problèmes à résoudre. Les réseaux récurrents sont très instables et leur apprentissage est un processus difficile à mettre en place. La plupart des résultats pratiques obtenus avec ce type de réseau concernent des problèmes très simples.

De plus, nous avons été amenés au choix des modèles basés sur les PMC afin d'aller vers une nouvelle voie de recherche dans notre équipe, puisque d'autres recherches sur un système hybride à base de prototypes (SYNHESYS) étaient en cours de développement par d'autres chercheurs de notre groupe. Les résultats obtenus dans ces recherches (qui d'ailleurs ont montré un certain nombre de faiblesses de ce type de SHNS à base de prototypes) nous ont dirigés vers cette nouvelle solution. Une de nos idées était justement d'étudier comment porter les idées de base du système SYNHESYS sur un système hybride basé sur des réseaux PMC.

Un autre point que nous avons considéré est lié à la possibilité de faire une acquisition constructive de connaissances, c.-à-d. que le système doit avoir la capacité de faire évoluer ses connaissances. Dans le cas des réseaux connexionnistes, cela veut dire que le réseau doit être aussi capable de faire évoluer sa structure. Ce type de caractéristique est retrouvé dans les systèmes comme SYNHESYS, KBCNN et RAPTURE. On trouve dans la littérature sur les systèmes hybrides plusieurs références qui indiquent la nécessité de développer des nouvelles recherches sur des méthodes qui puissent permettre de faire évoluer la structure des réseaux connexionnistes employés dans les systèmes hybrides [SHA 95, ORS 95, HIL 94b].

De plus, il est intéressant de préciser que notre système, le système INSS, a été conçu à partir des années 1993/1994, et qu'à cette époque les deux systèmes les plus connus et les plus accessibles à l'étude étaient le système SYNHESYS et le système KBANN. Donc, nous avons été influencés principalement par ces deux systèmes. Nous allons les décrire plus en détails dans les sections suivantes.

Nom du Système	*Type de Intégration*	*Degré de Couplage*	*Mode de Intégration*	*Transfert de Connaissances*	*Type de Représentation*	*Codage des Connaissances*	*Méthode de Raisonnement*	*Méthode de Apprentissage*
SYNHESYS [GIA 92]	Hybride	Forte	Co-Trait.	$S \Rightarrow C : R0^+$ $S \Leftarrow C : R0^+$	Localiste	MS : $R0^+$ MC: Prototype	Avant/Arrière Propagation	Continue Supervisée
ProBis [MAL 96]	Hybride	Moyen	Co-Trait.	$S \Rightarrow C$: Cas $S \Leftarrow C$: Cas	Localiste	MS : CBR MC: Prototype	K-PPV Propagation	Continue Supervisée
NESSY3L [ORS 95]	Hybride	Moyen/Forte(1)	Chaîné/Méta(1)	$S \Rightarrow C: R0^+$ (1)	Combiné	MS: RFloues(1) MC:PMC/Prot(1)	Ch.Avant Propagation	Non-Continue Supervisée
INSS [OSO 95]	Hybride	Moyen	Co-Trait.	$S \Rightarrow C : R0^+$ $S \Leftarrow C: R0^+$ (1)	Combiné	MS : Système Expert (CLIPS) MC : PMC (3)	Avant/Arrière(1) Propagation	Continue Supervisée
KBANN [TOW 91]	Semi-Hybride	Moyen	[pas de MS]	$S \Rightarrow C : R0$ $S \Leftarrow C : R0$ (2)	Localiste	MS : --- MC : PMC	Propagation	Non-Continue Supervisée
SCANDAL [STU 96]	Hybride	Moyen	Co-Trait.	$S \Rightarrow C : R0^+$ (1) $S \Leftarrow C : R0^+$ (1)	Combiné	MS : Règles, arbres de décision, et autres. MC : PMC	Inférence Symbolique et Propagation	Continue (1) Supervisée
MACIE [GAL 88]	Semi-Hybride	Moyen	[pas de MS]	$S \Rightarrow C$: Dépendances $S \Leftarrow C : R0$	Localiste	MS : --- MC : "Pocket Algorithm"	Propagation	Non-Continue Supervisée
KBCNN [FU 93]	Semi-Hybride	Moyen	[pas de MS]	$S \Rightarrow C : R0$ $S \Leftarrow C : R0$	Localiste	MS : --- MC:PMC (3)	Propagation	Continue Supervisée
SETHEO [SUT 90]	Hybride	Faible	Sous-Trait.	$S \Leftarrow C$: Paramètres pour l'optimisation de la preuve	Combiné	MS:Démonstrateur Automatique MC : PMC	Preuve Autom. Propagation	Non-Continue Supervisée
WATTS [KRO 92]	Hybride	Faible	Trait. Chaîné	$S \Rightarrow C$: Cas pré-analysés	Combiné	MS : CBR MC : Réseaux de Hopfield	Propagation	Non-Continue Supervisée
INNATE-QUALMS [BEC 91]	Hybride	Faible	Sous-Trait.	$S \Leftarrow C$: Diagnostic	Combiné	MS : Système Expert MC : PMC	Inférence Symbolique et Propagation	Non-Continue Supervisée
LAM [MED 92]	Hybride	Faible	Sous-Trait.	$S \Rightarrow C$:Cas à traiter $S \Leftarrow C$: Prediction	Combiné	MS : R0 MC : PMC	Ch. Avant Propagation	Non-Continue Supervisée
DDT [GUT 90]	Hybride	Faible	Co-Trait.	Pas d'intéraction	Combiné	MS : Système Expert MC : PMC	Inférence Symbolique et Propagation	Continue (+/-) Supervisée
ALVINN [POM 91]	Hybride	Faible	Méta-Trait.	$S \Rightarrow C$: Problème $S \Leftarrow C$: Réponse	Combiné	MS:Planification et Contrôle MC : PMC	Inférence Symbolique et Propagation	Non-Continue Supervisée
SCRAPS [HEN 89]	Hybride	Fort	Sous-Trait.	Partage de données	Combiné	MS : Réseau Sémantique MC : PMC	Inférence Symbolique et Propagation	Non-Continue Supervisée

RAPTURE [MAH 93]	Hybride	Faible	[pas de MS]	S $\Rightarrow$ C : R0 avec CF	Localiste	MC : CFBP (ajout d'unités basé sur ID3)	Propagation	Continue Supervisée
CONSYDERR [SUN 91]	Unifié	---	[pas de MS]	---	Combiné	MC: Approche Spécifique	Propagation	Continue Supervisée
RUBICON [SAM 92]	Unifié	---	[pas de MS]	---	Combiné	MC: Approche Spécifique	Propagation	Sans-Apprent. (codé directement)
RAAM [POL 90]	Unifié	---	[pas de MS]	---	Distribué	MC : Réseau RAAM	Propagation (unification)	Supervisée
BoltzCONS [TOU 90]	Unifié	---	[pas de MS]	---	Distribué	MC: Approche Spécifique	Propagation (manip. de structures symb. complexes)	Supervisée
SHRUTI [SHA 93]	Unifié	---	[pas de MS]	---	Localiste	MC: Approche Spécifique	Propagation	Codage
EBNN [MIT 93]	Semi-Hybride	Faible	[pas de MS]	---	Distribué	MC: Approche Spécifique (TangentProp)	Propagation (Clauses de Horn représentés par RNAs)	Supervisée

(1) = Améliorations et extensions possibles prevues; (2) = Règles d'ordre 0 et règles du type M_Parmi_N (Subset: M_of_N); (3) = PMC avec Rétro-Prop. modifié
R0 = Règles d'ordre 0; $R0^+$ = Règles d'ordre 0^+; MS = Module Symbolique; MC = Module Connexionniste; CF = Degré de certitude (*Certainty Factor*)

Tableau 4 - Classification de plusieurs SHNS connus

3.3 REALISATIONS DE SYSTEMES HYBRIDES

Nous allons décrire dans cette section les deux systèmes qui ont le plus contribué à la conception du système INSS : le système SYNHESYS et les réseaux KBANN.

3.3.1 Système SYNHESYS

Le système SYNHESYS a été développé au laboratoire LEIBNIZ (ancien lab. LIFIA) par A. Giacometti [GIA 92]. Il est en fait un noyau d'un système expert hybride, qui a été utilisé dans la conception de plusieurs applications, notamment SHADE (aide au diagnostic en électromyographie) et SATAN-hybride (aide à la manoeuvre navale). Ce système a été développé dans le cadre plus général d'une étude psycho-mimétique de l'expertise. Son but était d'étudier et mettre en pratique une approche basée sur la démarche experte humaine afin d'obtenir de meilleurs systèmes experts.

Giacometti défend l'hypothèse suivante : " un expert n'est pas uniquement un sujet qui raisonne à partir de règles ou un sujet qui a mémorisé et est capable de discriminer des milliers de cas particuliers. C'est un être hybride dans le sens où il utilise des connaissances, des modes de pensée et d'apprentissage de natures différentes. Sa compétence ne peut pas être réduite à un mode de pensée et/ou d'apprentissage particulier. Elle résulte en partie de l'interaction entre des modes de pensée et d'apprentissage différents, mais pouvant être complémentaires. ". C'est en se basant sur cette hypothèse que le système SYNHESYS a été conçu.

Nous présentons ici une description très simplifiée du fonctionnement de ce système. Notre but est de donner un aperçu général du système de façon à mettre en valeur ses propriétés principales qui nous ont influencés dans nos recherches et travaux. Le système SYNHESYS se compose d'une architecture à deux modules principaux : le module symbolique et le module connexionniste. Ces deux modules peuvent être utilisés en parallèle dans un mode de coopération ou bien on peut réaliser des transferts de connaissances entre les deux. La Figure 19 montre l'architecture du système, en mettant également en évidence le fait que les deux modules ont exactement les mêmes entrées, provenant d'une situation (ensemble de variables avec leurs attributs respectifs), et qu'ils produisent chacun une décision (réponse du système) qui peut être ensuite analysée par l'expert. Dans les sections suivantes, nous allons d'abord décrire les modules symbolique et connexionniste, pour pouvoir ensuite présenter les méthodes de transferts de connaissances existantes et les types de processus de coopération réalisés entre les modules.

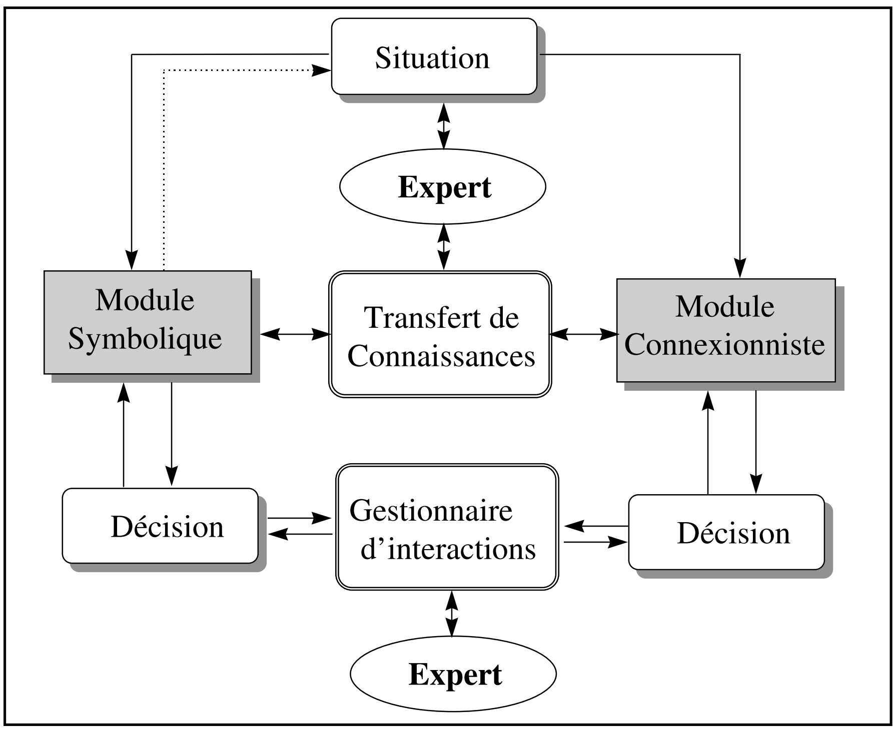


Figure 19 - Architecture du système SYNHESYS

3.3.1.1 Module Symbolique du Système SYNHESYS

Le module symbolique du système SYNHESYS travaille sur des règles propositionnelles d'ordre 0^+. Ces règles représentent les connaissances théoriques du système sur le problème en question. La Figure 20 montre un exemple d'une base de règles utilisé par le système SYNHESYS.

```
BASE_REGLES : exemple
Auteur : Osorio
Date   : 22/01/97

/*---------------------------------------------------------*/

ENTREES
y [0,20]
x [0,30]

ACTIONS
act1 act2 act3

REGLE r1   Si y dans  [0,10] alors y_faible        FINREGLE r1
REGLE r2   Si y dans [10,20] alors y_fort          FINREGLE r2
REGLE r3   Si x dans  [0,10] alors x_faible        FINREGLE r3
REGLE r4   Si x dans [10,20] alors x_moyen         FINREGLE r4
REGLE r5   Si x dans [20,30] alors x_fort          FINREGLE r5
REGLE r6   Si y_faible alors act1                  FINREGLE r6
REGLE r7   Si y dans [14,20] et x_moyen alors act2 FINREGLE r7
REGLE r8   Si y_fort et x_faible alors act3        FINREGLE r8
REGLE r9   Si y_fort et x_fort   alors act3        FINREGLE r9

/*---------------------------------------------------------*/

FIN
```

Figure 20 - Exemple typique d'une base de règles symboliques de SYNHESYS

Le module symbolique possède un moteur d'inférence avec un chaînage avant et aussi un chaînage arrière. Le chaînage avant permet d'établir les décisions (actions/réponses) à partir des situations (cas particuliers) présentées au système. Inversement, le chaînage arrière permet de savoir quelles sont les prémisses nécessaires au déclenchement d'une action spécifique. Le chaînage arrière prend une grande importance par rapport au processus de coopération entre les deux modules. Il permet de valider les réponses obtenues par le module connexionniste (voir section 3.3.1.4).

3.3.1.2 Module Connexionniste du Système SYNHESYS

Le module connexionniste réalisé utilise des réseaux à base de prototypes, appelés ARN2 [GIA 92, ORS 95, MAL 96b]. Ce module permet d'exploiter deux types distincts de réseaux à prototypes : des réseaux à *hyper-sphères* et des réseaux à *hyper-rectangles*. Les réseaux à hyper-sphères sont illustrés par l'exemple de la Figure 11 (voir section 2.3.1.2), pour le cas d'un réseau à deux entrées seulement. La Figure 21 montre un exemple d'un réseau à hyper-rectangles, qui représente lui aussi le cas d'un réseau à deux entrées seulement. L'utilisateur du système peut choisir le type de réseau (un seul à la fois) qui va être employé dans le module connexionniste. Il faut savoir que seuls les réseaux à hyper-rectangles permettent l'extraction de règles symboliques.

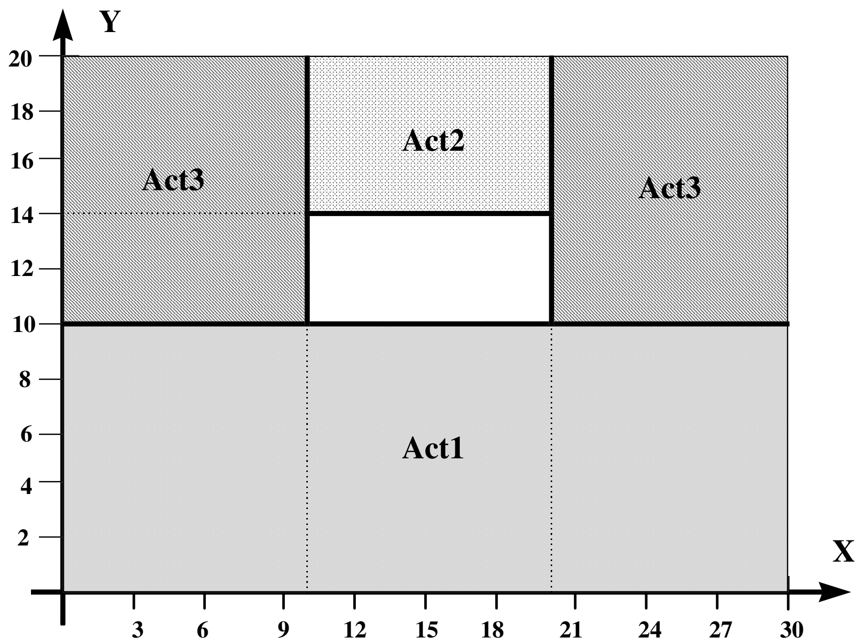


Figure 21 - Prototypes dans un réseaux à hyper-rectangles avec deux entrées

Le MC est composé par un réseau à trois couches : une couche d'entrée, une couche cachée de prototypes et une couche de sortie regroupant les décisions (classes). La dynamique du processus d'activation est déclenchée par une situation qui est présenté à l'entrée du réseau. L'activation de la couche d'entrée va entraîner le calcul des similitudes entre le vecteur d'entrée et les prototypes stockés. Un processus de compétition (WTA - *Winner Take All*) est ainsi déclenché et permet de sélectionner le prototype le plus 'proche' du vecteur d'entrée présenté au réseau. Le prototype gagnant étant associé à une des unités de sortie fait que cette unité est activé à son tour. Dans ce type de réseau, un ou plusieurs prototypes sont associés à une des unités de sortie, et chacune de ces unités représente une décision. Ce modèle est du type incrémental puisqu'on peut ajouter des nouveaux prototypes dans la couche cachée pendant le processus d'apprentissage. La Figure 22 présente un schéma de l'architecture de ce type de réseau.

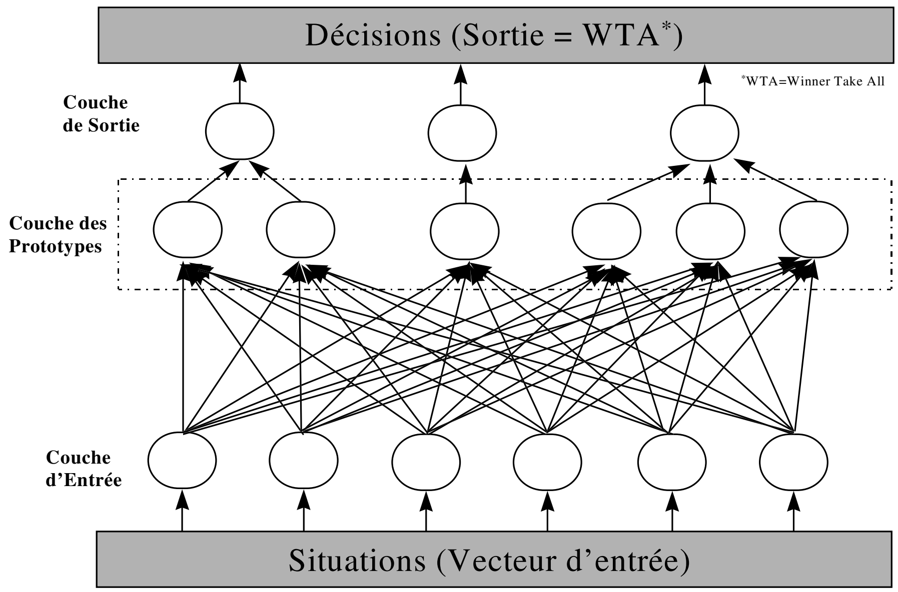


Figure 22 - Architecture multicouches du réseau à prototypes utilisé dans SYNHESYS

L'apprentissage dans un réseau à prototype consiste à réaliser une des tâches suivantes :

- S'il n'y a pas de prototype assez proche de l'exemple présenté, il va falloir créer un nouveau prototype qui va représenter cet exemple, ou il va falloir modifier un des prototype existants afin de l'approcher de l'exemple. Ce cas de figure se produit quand l'exemple se trouve en dehors de la région d'influence de tous les prototypes du réseau, et par conséquence aucune des sorties du réseau n'est activée. En particulier, les réseaux de type ARN2 font la création d'un nouveau prototype dans un cas comme celui-ci;

- Si l'exemple tombe dans la région d'influence d'un prototype qui n'appartient pas à la même classe que lui (la réponse du réseau n'est pas la bonne), il va falloir passer par une phase d'adaptation. Dans ce cas, le prototype activé doit être adapté afin de changer sa région d'influence. L'adaptation peut être obtenue par la réduction de la région d'influence du prototype (on appelle cette approche, qui est la technique employée par les réseaux ARN2, *différentiation*). Une autre possibilité d'adaptation est le déplacement du prototype dans une direction opposé à celle où se trouve l'exemple;

- Si l'exemple tombe dans la région d'influence d'un prototype qui représente la même classe que lui (la réponse du réseau est la bonne), on peut alors réaliser une *accommodation* (déplacement) de la région d'influence du prototype, de façon à mieux prendre en compte l'exemple présenté. L'idée étant de déplacer les prototypes ver le 'centre de gravité' des exemples qu'il représente. Les réseaux ARN2 utilisent ce type d'approche.

La base d'exemples est présentée au réseau jusqu'à ce que le réseau ait assez bien appris le problème, c.-à-d. que l'erreur obtenue dans ses réponses soit arrivée à un niveau satisfaisant. Puisque le réseau est du type incrémental, il est toujours possible d'ajouter des nouvelles unités afin d'améliorer sa performance jusqu'au niveau envisagé. Par contre, il faut faire très attention dans ce type de modèle à bien régler certains paramètres d'apprentissage afin de ne pas ajouter trop d'unités, ce qui implique usuellement une perte de la capacité de généralisation du système. Le cas extrême est celui où le MC va représenter chaque exemple de la base d'apprentissage par un prototype.

Avec ce type méthode, il faut aussi faire attention aux cas contradictoires (deux cas/situations identiques associés à décisions/actions différentes) qui peuvent gêner le processus d'apprentissage. Un autre problème à traiter est l'oubli des cas, puisque tout changement subi par un prototype (différentiation ou accommodation) peut entraîner l'exclusion indésirable (ou même l'inclusion) d'exemples dans le rayon d'influence de celui-ci. La solution dans ce cas est la réalisation de plusieurs présentations de la base d'exemples afin de bien établir les rayons d'influence des prototypes. Ceci étant dit, on s'aperçoit que le module connexionniste de SYNHESYS est très sensible à certains paramètres, tels que : le rayon initial d'influence des prototypes, l'ordre de présentation des exemples, les pas de déplacement et de réduction du rayon d'influence des prototypes, et à la quantité de 'bruit' (cas mal classés) présent dans la base d'apprentissage.

3.3.1.3 Transfert de Connaissances dans le Système SYNHESYS

Le système SYNHESYS met en oeuvre deux types de transferts de connaissances entre les modules symbolique et connexionniste :

• MS ⇒ MC : transfert par la compilation d'un ensemble de règles dans un réseau. Ce type de méthode ne peut être appliqué que dans le cas des réseaux à hyper-rectangles. Il est aussi possible de faire le transfert par génération aléatoire d'un ensemble d'exemples (classés avec l'aide de la base de règles) et de réaliser son apprentissage postérieurement par un réseau à base d'hyper-sphères ou d'hyper-rectangles.

• MC ⇒ MS : transfert par l'extraction de règles d'un réseau. Ce type de méthode de transfert de connaissances n'est disponible que pour les réseaux à base d'hyper-rectangles.

Les algorithmes de compilation et d'extraction de règles ont été présentés de manière formalisée par Giacometti [GIA 92]. Nous allons nous contenter ici d'en décrire ces processus de manière intuitive et très simplifiée. Le fait de nous concentrer sur ces deux types d'algorithmes de transfert de connaissances implique que nous allons décrire les processus qui interviennent exclusivement sur les réseaux à base de hyper-rectangles.

Prenons comme exemples les Figure 20 et Figure 21, qui représentent respectivement une base de règles et les prototypes d'un réseaux à hyper-rectangles. Les règles 'r1' et 'r6' de la Figure 20 peuvent être compilés dans une représentation connexionniste à prototypes, tel que l'on montre par la région indiqué comme 'Act1' dans la Figure 21. Inversement, on peut facilement extraire des règles d'ordre 0^+ à partir des prototypes à hyper-rectangles, comme par exemple, la région indiqué par 'Act2' dans la Figure 21 qui correspond à la règle suivante : *Si* X dans [10,20] et Y dans [14,20] *Alors* Act2. On peut lire cette règle autrement : Si x appartient à l'intervalle [10,20] et y appartient à l'intervalle [14,20] alors l'action Act2 est effectuée, ou bien, on considère que Act2 est actif/vrai. Les exemples à deux variables sont plus simples à interpréter, mais rien ne nous empêche de passer à plus de deux dimensions, avec des prototypes basé sur des hyper-rectangles en N-dimensions. Les processus de compilation et d'extraction marchent aussi bien avec deux variables qu'avec plus.

Dans le cas du transfert de connaissances du module symbolique vers le module connexionniste par génération d'exemples, nous sommes de l'avis que ce type de méthode a quand même un certain nombre d'inconvénients. Premièrement, le fait de créer des exemples aléatoirement ne permet pas de garantir que toutes les règles vont être bien représentés dans la base d'exemples. Deuxièmement, le fait de passer par une phase d'apprentissage à partir des exemples implique un coût computationel additionnel et en plus il ne nous assure pas que le réseau final ait bien appris tous les connaissances codées dans cette base. En conclusion, cette méthode n'est pas capable d'assurer un transfert complet et correct des connaissances entre les deux modules.

3.3.1.4 Coopération entre modules dans le Système SYNHESYS

En plus des méthodes de transfert de connaissances, les modules symbolique et connexionniste peuvent avoir une coopération par interaction à travers du module gestionnaire d'interactions. On peut avoir différents types d'interactions entre les modules selon l'ordre utilisé pour les consulter, la façon d'interpréter ses résultats, et les différentes priorités données aux réponses des deux modules. Il est aussi possible d'avoir des interactions entre l'expert et les modules MS et MC. En particulier, nous nous sommes plus intéressés à l'un de ces modes d'interactions lié au processus de validation des réponses et à la possibilité d'améliorer les connaissances acquises par le système. Nous allons décrire ci-dessous ce mode particulier de fonctionnement de SYNHESYS.

Une fois que les deux modules peuvent être utilisés en parallèle, on peut avoir des réponses contradictoires à la sortie des deux modules. C'est justement à travers la comparaison des réponses des modules MS et MC qu'on va pouvoir mettre en pratique une méthodologie d'analyse des réponses, d'interaction avec l'expert et de raffinement des connaissances acquises par le système. Dans ce mode d'opération, le système SYNHESYS va être considéré comme un *outil d'acquisition constructive et interactive de connaissances*.

L'acquisition interactive de connaissances est faite à travers la coopération entre les différents composantes du système et aussi avec l'aide d'un expert. Ce processus est brièvement décrit ci-dessous :

1. *Activation du MC* : le module connexionniste est activé en premier. Ce module va fournir une décision par sa sortie Dc (Décision Connexionniste). Selon le type de situation proposée à l'entrée du MC, il peut arriver qu'il ne donne aucune réponse. Cela se produit si la situation présentée constitue un cas qui n'est pas suffisamment proche des prototypes connus et ainsi le système ne pourra pas prendre une décision. Dans ce cas, on va être obligé de consulter le MS (on passe à l'étape 4).

2. *Validation de la Dc par le MS* : après l'activation du MC, on va prendre sa sortie Dc et faire l'activation du module symbolique par chaînage arrière. Le chaînage arrière sert à vérifier si la réponse donnée par le MC correspond à une des réponses possibles obtenue à partir des règles du MS. Si l'activation du MS est en accord avec le MC alors la réponse finale du système est communique à l'expert (on passe à l'étape 5). Si l'activation du MS est en contradiction avec le résultat obtenu par le MC alors on doit résoudre le conflit entre les deux modules.

3. *Conflit entre les deux modules* : les réponses du MC et du MS sont passés à l'expert afin qu'il puisse juger laquelle des deux est la réponse correcte. Si la sortie Dc est considérée correcte, alors il va falloir changer la base de règles. Si la sortie Ds est considérée correcte, alors il va falloir adapter le MC (adaptation ou élimination du prototype responsable de la mauvaise réponse). Après cet étape, le système est prêt pour une nouvelle consultation.

4. *MC n'est pas arrivé à une décision* : le MS est activé par chaînage avant et obtient ainsi une réponse Ds (Décision Symbolique). La réponse Ds donnée par le MS doit être validé par l'expert. Le MS est capable de produire une explication de son raisonnement et si l'expert trouve cela correct, il va falloir adapter le MC afin de prendre en compte cet exemple. Dans ce cas, le réseau va être modifié de façon à donner la réponse Ds pour cette situation proposée, ce qui pourra donner l'origine à un nouveau prototype.

5. *MC et MS sont en accord* : si les deux modules ont données des réponses qui sont en accord, alors le système présente à l'expert la solution finale. L'expert va juger cette solution, et

s'il trouve qu'elle est convenable le système reste inaltéré. Si la réponse donnée n'est pas correcte, il va falloir réviser les connaissances des deux modules.

La Figure 23 montre un schéma simplifié du processus d'acquisition interactive de connaissances décrit ci-dessus et qui est réalisé par le système SYNHESYS. Nous trouvons ce mode de coopération entre les deux modules et l'expert très intéressant puisqu'il permet d'améliorer les connaissances symboliques et connexionnistes. Les interactions entre modules permettent de mettre en relation les connaissances théoriques représentées dans le MS avec les connaissances empiriques représentées dans le MC. Cette approche a certains points en commun avec le système DDT [GUT 90], mais le système SYNHESYS va encore plus loin et nous offre la possibilité de faire des transferts de connaissances entre les deux modules.

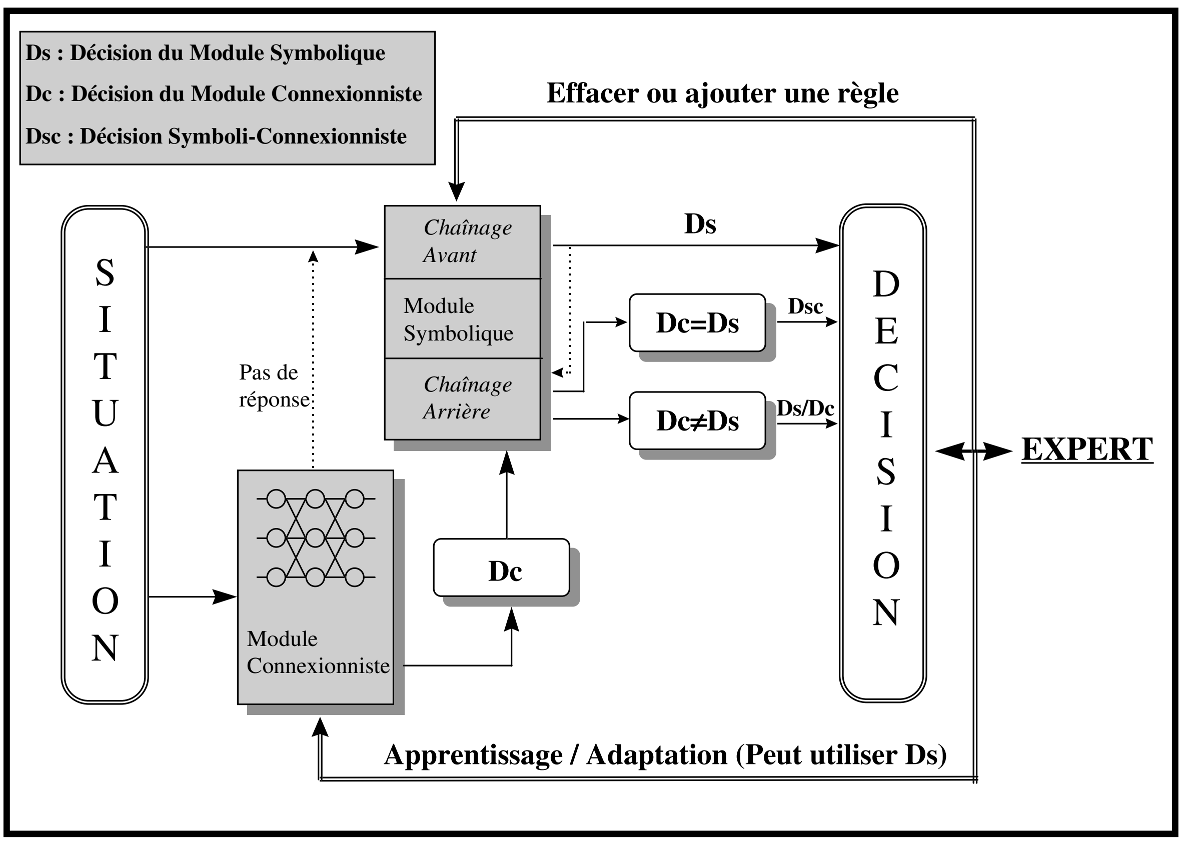


Figure 23 - L'acquisition interactive de connaissances fait par SYNHESYS

3.3.1.5 Discussion sur le modèle de SHNS proposé par SYNHESYS

Le système SYNHESYS est un exemple très réussi d'implémentation d'un noyau de système hybride neuro-symbolique. Ce système exploite bien les méthodes d'intégration entre MS et MC, et permet le transfert de connaissances dans les deux sens entre ses modules. De plus, il montre bien l'importance de l'interaction entre MS, MC et l'expert afin d'améliorer ses connaissances. C'est un système qui possède des caractéristiques que l'on aimerait retrouver dans d'autres systèmes SHNS :

- Le couplage à la fois moyen (avec des transferts dans les deux sens) mais aussi fort (utilisation de liens internes entre les deux modules) [ORS 95, ORS 94]. Le couplage fort permet d'optimiser le processus d'inférence dans le système;

- Les processus d'interaction et de coopération entre ses modules lui donnent un mode d'intégration par co-traitement;

- La possibilité d'augmenter le réseau afin d'acquérir des nouvelles connaissances (modification structurale);

- La possibilité de traiter directement des valeurs continues (variables quantitatives) sans restrictions et sans avoir besoin d'utiliser des techniques de pre-traitement des données (discrétisation). Le système travaille avec des règles d'ordre 0^+ qui peuvent être compilées dans son module connexionniste.

Cependant, le système SYNHESYS présente aussi un certain nombre de limitations. Bien que le module connexionniste avec ses réseaux à prototypes soit un modèle assez intéressant, il possède des propriétés qui peuvent être assez gênantes selon le type de problème traité. La plupart des problèmes (et pas mal d'avantages) du système SYNHESYS sont attribués au choix de son modèle de réseau connexionniste à prototypes, à savoir :

- Les prototypes à hyper-rectangles ne sont pas bien adaptés à la représentation des cas qui constituent des frontières 'diagonales' (non-parallèles aux axes de l'espace d'entrée) ou trop irrégulières [GIA 92] (c'est aussi un problème commun aux arbres de décision[QUI 93]);

- Les connaissances sont représentées par des unités localistes que ne prennent pas en considération des relations existantes *entre* les différentes prototypes. Ce type de représentation connexionniste pose des limitations par rapport à sa capacité générale de représentation de connaissances et complexifie l'apprentissage de règles telles que : Si A=B Alors ActionX ou Si A>B Alors ActionY;

- Les réseaux à hyper-rectangles utilisés par SYNHESYS peuvent présenter des problèmes de non-convergence [ORS 95] et ils manquent d'une preuve plus formelle de leurs propriétés de convergence vers une solution du problème;

- Le processus d'apprentissage passe par des phases de création, accommodation et différentiation des prototypes. Ces processus sont à la fois très sensibles aux paramètres de configuration de l'algorithme d'apprentissage, mais aussi responsables de problèmes comme l'oubli de cas ou la création excessive de prototypes;

- Les réseaux à prototypes sont basés sur une mesure de similarité, qui n'est pas toujours adaptée aux différents problèmes qui sont posés. Le système SYNHESYS utilise un type de réseau et une mesure de similarité qui sont bien adaptés aux variables continues (données quantitatives) mais qui montre des limitations assez fortes par rapport au traitement de variables discrètes (données qualitatives) [ORS 95]. Compte tenu du fait que la majorité des applications des systèmes experts utilisent beaucoup de données qualitatives, le système SYNHESYS a un fort handicap en ce qui concerne le traitement de ce type de problèmes;

- Les réseaux à seulement trois niveaux imposent des restrictions pour la représentation de concepts intermédiaires. Une base de règles symboliques peut en réalité être constituée par plusieurs niveaux intermédiaires. Il faut alors créer des structures internes parfois un peu complexes de connexion entre le module symbolique et le module connexionniste afin de pouvoir préserver les liens existants entre les concepts intermédiaires définis dans une base de règles et les structures connexionnistes de représentation de connaissances;

- Les règles extraites à partir des réseaux sont parfois trop nombreuses. On peut aussi se poser des questions sur *la qualité* et sur *l'utilité* de ces règles après leur extraction, à partir du moment où elles sont fournies à l'expert afin d'être analysées et utilisées dans la pratique.

On trouve une discussion plus détaillée à propos des propriétés et des limitations du système SYNHESYS dans les études publiés par B. Orsier [ORS 95, ORS 94]. Nous n'avons pas l'intention de revenir sur tous ces détails, car notre intérêt était plutôt de mettre en évidence certains facteurs qui nous ont influencés dans le développement de notre système hybride INSS. Dans notre système, on a retenu les points positifs de SYNHESYS et nous les avons combinés avec une autre approche hybride, les réseaux KBANN, afin d'obtenir un SHNS plus performant.

3.3.2 Réseaux KBANN

Les réseaux KBANN (*Knowledge-Based Artificial Neural Networks*) ont été développés dans le MLRG-UWM (*Machine Learning Research Group - University of Wisconsin-Madison*) par G. Towell et J. Shavlik [TOW 91, TOW 93, TOW 94, SHA 94, SHA 95]. Les réseaux KBANN sont une approche semi-hybride qui permet la compilation, le raffinement et l'extraction de règles symboliques. Cette approche permet une intégration de connaissances théoriques et empiriques d'une façon simple et avec des résultats pratiques très bons.

Les réseaux KBANN sont le résultat de l'implémentation de trois algorithmes : le premier algorithme est responsable de la traduction de règles symboliques approximativement correctes vers un RNA de type PMC; le deuxième algorithme fait le raffinement des connaissances du réseau par apprentissage d'une base d'exemples à travers l'utilisation de la Rétro-Propagation; le troisième algorithme va réaliser l'extraction des règles raffinées à partir du réseau qui a été entraîné. De cette façon, ces trois modules permettent de faire des transferts de connaissances entre un module symbolique et le module connexionniste. On peut éventuellement réintroduire les règles raffinées dans le réseau. La Figure 24 présente un schéma des trois modules principaux qui constituent ce type d'approche.

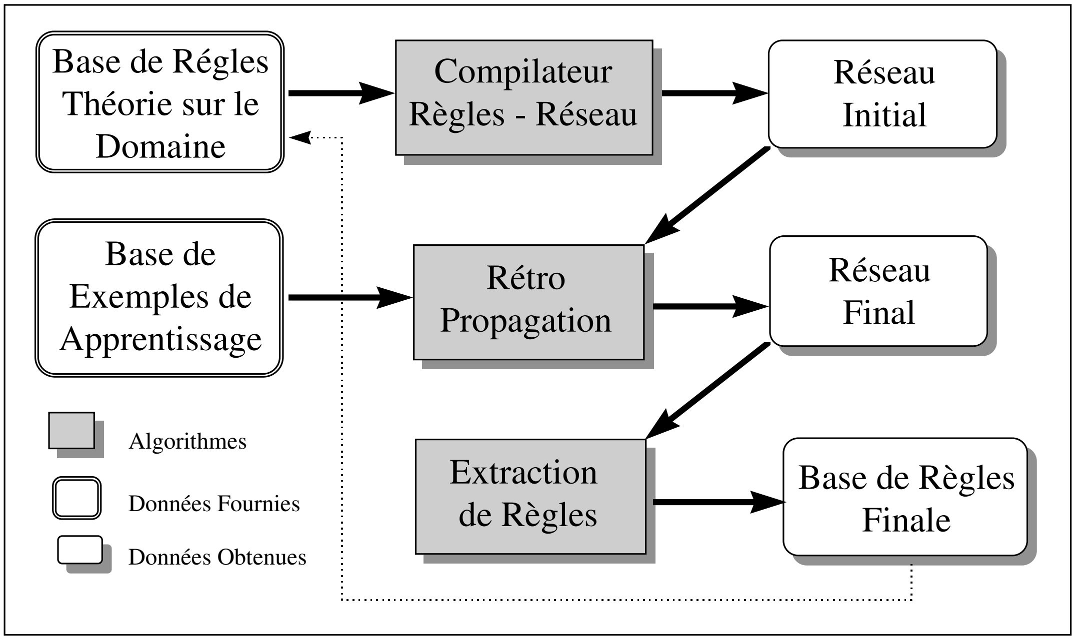


Figure 24 - Schéma de l'intégration neuro-symbolique utilisé dans les réseaux KBANN

Selon Towell [TOW 91], les réseaux KBANN peuvent soit éliminer complètement soit réduire significativement une grande partie des problèmes inhérents aux réseaux de neurones classiques et aux méthodes d'apprentissage empiriques. Il a montré que les réseaux KBANN dépassent nettement les performances des approches symboliques pures utilisés par des systèmes comme EITHER, ID3, COBWEB ou Labyrinth-K, et aussi des approches connexionnistes pures comme c'est le cas des réseaux classiques de type PMC à Rétro-Propagation. Cette approche a été testée sur différentes applications, tel que des problèmes réels du domaine de la biologie moléculaire et dans la modélisation du développement du raisonnement géométrique chez l'enfant. Les résultats obtenus avec ce système sont très encourageants.

Nous allons présenter dans la section suivante la compilation de règles en réseau de neurones, puis la méthode de raffinement de connaissances par apprentissage qui est utilisée dans les réseaux KBANN. Après, nous allons présenter les méthodes d'extraction de règles, pour finir par une discussion sur les propriétés et problèmes des réseaux KBANN.

3.3.2.1 Compilation de Règles dans KBANN

Le transfert de connaissances théoriques (règles symboliques) vers le module connexionniste est réalisé dans les réseaux KBANN par un algorithme de compilation de règles approximativement correctes. Le Tableau 5 montre les relations entre une base de connaissances et les réseaux de type KBANN. La compilation de règles nous permet d'obtenir un réseau qui donne exactement les mêmes réponses que nous pouvons obtenir à partir des règles qui ont servi à le créer. Cette façon de créer les réseaux permet de simplifier beaucoup le processus de spécification et d'apprentissage d'un réseau. L'architecture initiale du réseau est obtenue de façon automatique et nous n'avons plus de soucis par rapport à la détermination du nombre de couches, de neurones et d'interconnexions du réseau. De plus, les règles donnent au réseau des connaissances de départ. Ces connaissances sont codées par des liens qui représentent des relations de dépendance entre les différents attributs d'entrée et les réponses qu'on doit obtenir à la sortie. Cette configuration initiale introduite dans le réseau est probablement plus proche de la configuration finale qu'on veut obtenir qu'une autre configuration aléatoire quelconque des poids. Ainsi, le problème du choix aléatoire des poids initiaux d'un réseau et de ses conséquences (souvent négatives) sur l'apprentissage peut être contrôlé et minimisé. Towell a montré dans ses recherches que ce type d'approche permet de réduire le temps d'apprentissage et le nombre d'exemples nécessaires à un réseau pour bien apprendre à résoudre un problème.

Base de Connaissances	**Réseau de Neurones**
Conclusion Finales	Unités de Sortie
Conclusions intermédiaires	Unités de la couche cachée
Faits (prémisses de base)	Unités d'entrée
Dépendances	Poids des connexions

Tableau 5 - Correspondances entre les bases de connaissances et les RNAs

Les règles utilisées par KBANN sont exprimés par des clauses de Horn avec une notation que nous décrivons dans la Figure 25. Deux restrictions sont imposées sur l'ensemble de règles. Premièrement, les règles doivent exprimer des propositions et ne doivent pas contenir des variables. Deuxièmement, les règles doivent être acycliques, c.-à-d. qu'une règle ne doit pas avoir un conséquent qui intervient dans un de ses antécédents (directement ou indirectement). Donc, les règles utilisées sont des règles d'ordre 0 codées de façon hiérarchique. Les réseaux KBANN travaillent plutôt avec des valeurs discrètes, et les valeurs continues impliquent la réalisation d'un prétraitements des données (une sorte de discrétisation). Les travaux développés par Towell restent imprécis en ce que concerne le traitement de valeurs continues et son application dans la pratique.

<Conséquent> :- <Antécédent> , <Antécédent> , ... <Antécédent> .

<Conséquent> = C'est la partie qui représente la conclusion de la règle.

<Antécédent> = C'est la partie qui représente les prémisses. Un antécédent peut être composé par un **<opérateur>** suivi d'une formule atomique.

<Opérateur> = Il s'applique sur une formule atomique. L'opérateur le plus utilisé est le 'Not', mais les réseaux KBANN acceptent aussi des opérateurs 'Greater-Than', 'Less-Than' et 'N-True'.

Exemple:

Les règles symboliques - Si (A et B) ou (A et Non(D)) Alors C
Si A et N-parmi-M(2,B,C,D,E) Alors F

sont décrites par - C :- A , B.
C :- A , Not D.
F :- A, N-True(2,B,C,D,E).

Figure 25 - Syntaxe et exemple de règle symbolique utilisés dans KBANN

La Figure 25 montre un exemple d'une base de règles compatible avec les réseaux KBANN. Les règles expriment des conjonctions (opérateur 'et logique') entre les termes qui sont séparés par des virgules dans la liste des antécédents d'une règle. Les disjonctions (opérateur 'ou logique') sont représentées par un ensemble composé de plus d'une règle dont les conséquents sont les mêmes. L'opérateur 'Not' permet d'inverser la valeur logique des prémisses, et les opérateurs 'Greater-Than' et 'Less-Than' réalisent certaines opérations de comparaison entre deux variables (valeurs discrètes ordonnées - variables nominales). L'opérateur 'N-True' introduit un concept intéressant du type N-parmi-M (*NofM*), c'est-à-dire que la prémisse est vraie si N des formules atomiques parmi les M listées sont vraies, sinon elle est fausse. Nous allons revenir sur ces opérateurs dans les sections suivantes.

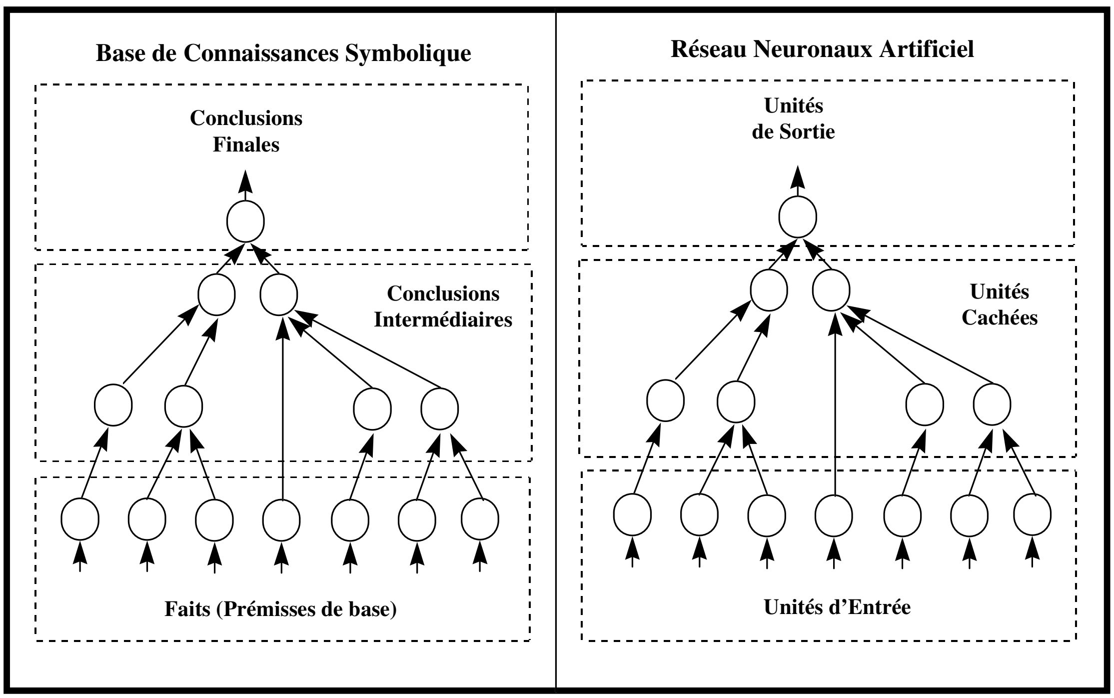


Figure 26 - Correspondance entre la structure d'une base de règles et d'un réseau de neurones

Le processus de compilation de règles dans un réseau est réalisé par un algorithme divisé en quatre étapes principales, à savoir :

1. *Réécriture des disjonctions* : les disjonctions sont réécrites de façon qu'une règle n'ait qu'une seule prémisse (formule atomique, sans négation) comme antécédent. Cette opération est nécessaire pour permettre de réaliser la compilation des règles dans un réseau. La réécriture des

règles ne change pas les connaissances ni le résultat de son application, elle va changer seulement la représentation écrite des règles. Voici un exemple de réécriture des disjonctions :

Avant réécriture	⇒	Après réécriture
A :- B, C, D.	⇒	A :- A'.
		A' :- B, C, D.
A :- D, E, F, G.	⇒	A :- A''.
		A'' :- D, E, F, G.

2. *Construction du réseau* : cette étape permet d'établir les correspondances entre un ensemble de règles et un réseau de neurones. La Figure 26 montre les correspondances entre ces deux types de représentations. Ainsi, la structure du réseau peut être créée à partir des règles symboliques disponibles. Une fois que toutes les unités sont bien disposées dans les couches du réseau selon ces correspondances, on passe à la définition des poids des connexions ainsi que du seuil de chacune des unités (*biais*). Les réseaux KBANN codent les poids et seuils selon le type de règle et les opérateurs qui y sont représentés (voir Figure 27) :

- Les poids des connexions originaires d'une prémisse positive reçoivent des valeurs égales à : **Poids = +W**;
- Les poids des connexions originaires d'une prémisse niée reçoivent des valeurs égales à : **Poids = -W**;
- Les seuils des unités qui représentent les conclusions obtenues à partir d'une conjonction reçoivent des valeurs égales à : **Seuil = (-P + 0.5) * W**;
- Les seuils des unités qui représentent les conclusions obtenues à partir d'une disjonction reçoivent des valeurs égales à : **Seuil = -0.5 * W**;
- La valeur W est une valeur constante spécifiée par l'utilisateur. Plus grand est W, plus grand sera l'écart entre une conclusion positive et une conclusion négative. KBANN utilise une valeur W=4, qui a été déterminé empiriquement et semble donner des bons résultats avec ce type de réseau;

- La valeur P indique le nombre de prémisses positives (sans négation) présentes dans la liste d'antécédents d'une unité qui représente une conclusion.

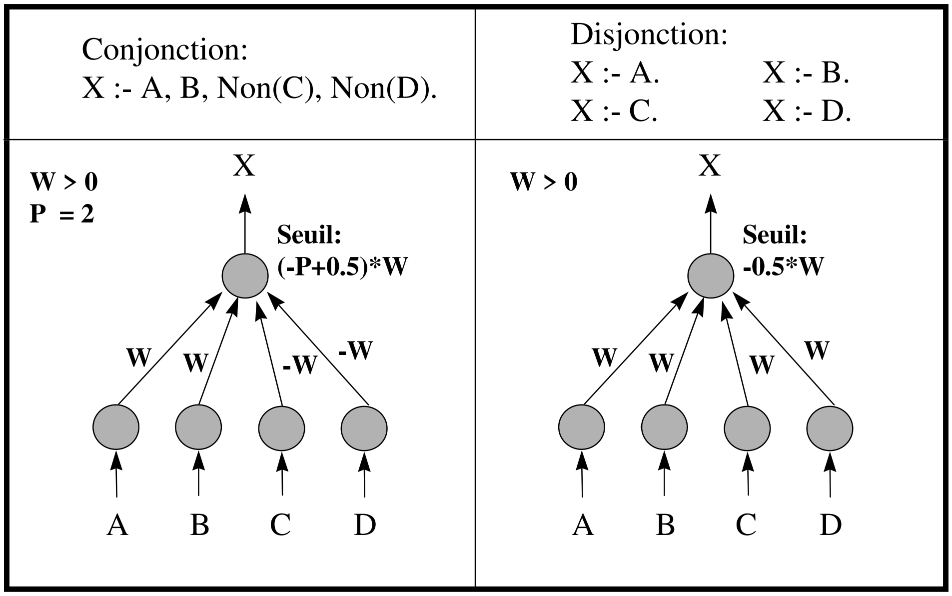


Figure 27 - Obtention des poids et des seuils des unités d'un réseau KBANN

Une preuve mathématique formelle et détaillé de l'exactitude des méthodes de transformation des règles vers des réseaux KBANN est présenté par Towell [TOW 91]. Donc, les réseaux KBANN obtenus par cette méthode codent parfaitement l'ensemble de règles. Des procédures similaires à celle-ci ont été proposées par d'autres chercheurs avant Towell [MCC 43, KLE 56].

3. *Ajout d'unités et de connexions* : une fois que la structure de base du réseau est construite, des unités peuvent y être ajoutés. Cette étape est optionnelle dans la construction des réseaux KBANN. On peut ajouter dans la couche d'entrée des unités qui représentent des attributs d'entrée non référencés dans la base de règles. On peut aussi ajouter des unités dans la couche cachée en cas d'indication explicite de l'expert à propos du placement de ces unités (procédure 'manuelle' d'insertion). Après l'ajout d'unités, on passe à l'étape d'ajout de connexions. Les connexions peuvent être ajoutés de façon à : lier toutes les unités appartenant à deux couches adjacentes; lier tous les unités d'une couche avec les unités des couches supérieures; lier chacune des entrées avec

chacune des unités cachées ou de sortie. Les connexions ajoutées reçoivent des poids et seuils égaux à zéro, et par conséquence, ne changent pas le comportement du réseau par rapport aux règles introduites.

4. Perturbation des poids : l'étape finale de la transformation de règles en réseau est réalisée par une perturbation de tous les poids du réseau. Cette perturbation consiste à additionner une valeur aléatoire très petite à chacun des poids des connexions. La perturbation introduite dans les poids est si faible qu'elle n'a aucun effet sur le comportement des réseaux KBANN avant l'apprentissage. Cependant, cette méthode permet d'éviter les problèmes d'apprentissage résultant d'une symétrie du réseau.

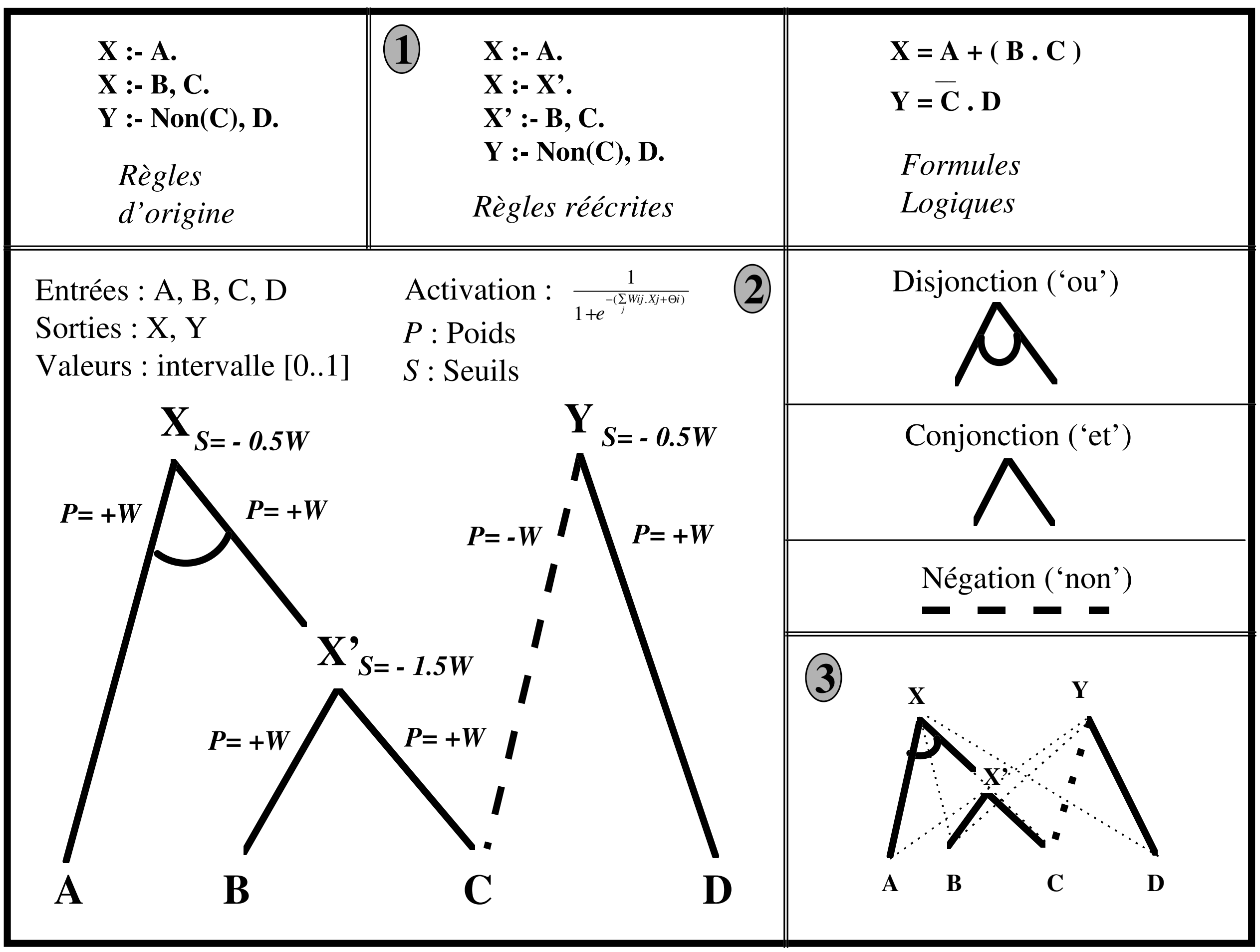


Figure 28 - Processus de transformation de règles dans un réseau KBANN

La Figure 28 montre un schéma du processus de transformation de règles en réseau. Nous précisons que les réseaux KBANN ainsi obtenus sont des réseaux de type PMC, où les valeurs d'entrée de chaque unité sont usuellement comprises dans l'intervalle [0,1] et les sorties de toutes les unités sont obligatoirement comprises dans l'intervalle [0,1]. Les réseaux KBANN utilisent une fonction de transfert du type sigmoïde asymétrique. Dans ces réseaux, une valeur d'entrée ou de sortie égale à 0.0 symbolise une prémisse/conclusion fausse et une valeur égale à 1.0 symbolise une prémisse/conclusion vraie. Les variables avec des attributs manquantes sont codées par des valeurs égales à 0.5. Cette valeur indique ainsi une 'indécision' dans la sortie d'une unité. Les réseaux KBANN n'utilisent pas de variables ayant des valeurs négatives. Les variables avec des attributs quantitatifs (valeurs continues) sont récodées d'une façon discrétisée par des sous-intervalles, où chaque sous intervalle correspond à des valeurs d'activation dans l'intervalle [0..1] (voir Figure 29).

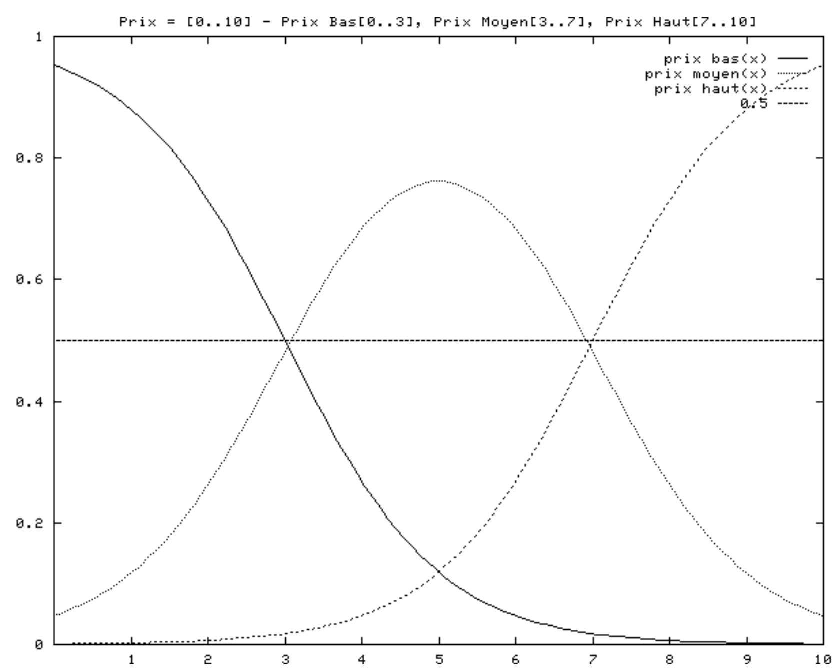


Figure 29 - Exemple de codage d'attributs numériques dans KBANN

3.3.2.2 L'Apprentissage dans les Réseaux KBANN

Les réseaux obtenus à partir des règles symboliques peuvent ensuite être entraînés avec une base d'exemples. Cette procédure permet de raffiner les connaissances introduites précédemment dans le réseau. Ce raffinement se fait en utilisant l'algorithme d'apprentissage de la Rétro-Propagation de l'erreur (voir section 2.3.2)[RUM 96].

Malheureusement, les réseaux KBANN posent des problèmes par rapport à l'algorithme classique de Rétro-Propagation, car ils commencent l'apprentissage avec un comportement fortement influencé par les connaissances compilées sous forme de réseau. Les réseaux compilés donnent des réponses sûres, c.-à-d. très proches de 1.0 (vrai) ou de 0.0 (faux), ce qui va rendre beaucoup plus difficile l'apprentissage. En raison de la formulation mathématique de la Rétro-Propagation, les unités qui ont une sortie proche de zéro ou proche de un ne vont pas avoir une adaptation significative de leurs poids synaptiques indépendamment de l'erreur constatée dans leur sortie. Puisque la dérivée de la fonction d'activation est très faible quand la sortie de l'unité s'approche de 1.0 ou 0.0 (voir Figure 14), cela va bloquer le processus d'apprentissage (voir Section 2.3.2 - Eq.18 et 24). Ce phénomène de paralysie de l'apprentissage est connu comme le problème du *'flat spot'* de la Rétro-Propagation [FAH 88, TOW 91].

Parmi les solutions possibles qui peuvent être utilisées pour résoudre ce problème, Towell a choisi de modifier la fonction d'erreur (erreur quadratique) utilisée par l'algorithme de la Rétro-Propagation. Donc, la seule différence entre la méthode d'apprentissage des réseaux KBANN et la méthode de la Rétro-Propagation se trouve dans la fonction d'erreur utilisé. Dans les réseaux KBANN, cette fonction est la fonction d'erreur d'entropie-croisée (*cross-entropy error function*) [HIN 89]. Nous présentons ci-dessous les différences entre les équations utilisées dans KBANN et dans la Rétro-Propagation.

- Rétro-Propagation - Algorithme classique :

Erreur quadratique :

$$E = \frac{1}{2}\sum_i (Di - Ai)^2$$

Equation utilisée dans le calcul de l'erreur d'une unité (voir Eq. 12, Section 2.3.2) :

$$\frac{\partial E}{\partial Si} = \frac{\partial E}{\partial Ai}\frac{\partial Ai}{\partial Si} = \text{- (Di - Ai).Fa'(Si)} = \delta i$$

- Rétro-Propagation - Algorithme utilisé dans KBANN :

Erreur basée sur la fonction d'entropie-croisée :

$$\text{CE} = -\sum_i [(1 - Di) * Log_2(1 - Ai) + Di * Log_2(Ai)]$$

Equation utilisée dans le calcul de l'erreur d'une unité :

$$\frac{\partial E}{\partial Si} = \frac{\partial E}{\partial Ai}\frac{\partial Ai}{\partial Si} = Log(2) * (\frac{Di}{Ai} - \frac{1 - Di}{1 - Ai}) * Ai * (1 - Ai) = \delta i$$

Dans le cas particulier où Di va assumer seulement deux valeurs (Di=1.0 ou Di=0.0) :

δi = (1 - Ai) pour Di=1.0

δi = -Ai pour Di=0.0

La caractéristique de cette fonction est qu'elle a une dérivée toujours très forte si, pour une sortie particulière, l'écart entre la valeur désirée Di et la valeur réelle Ai est grand. Donc, cela augmente la correction effectuée sur les poids des unités dont les valeurs de sortie sont proches de zéro alors qu'elles devraient être à un, et vice-versa. Ainsi, on augmente la vitesse d'apprentissage et on évite le problème de la paralysie de l'adaptation du réseau. En conclusion, la fonction d'entropie-croisée est plus adaptée aux réseaux KBANN et au type de sortie données après la transformation de règles en réseau. D'après Towell [TOW 91], les résultats pratiques des comparaisons entre les réseaux KBANN avec une fonction d'erreur classique et les réseaux KBANN avec une fonction d'erreur d'entropie-croisée, montrent une supériorité (performance et généralisation) de ce dernier type de fonction d'erreur.

3.3.2.3 Extraction de Règles des Réseaux KBANN

Deux méthodes d'extraction de règles à partir des réseaux ont été développées pour être utilisées avec les réseaux KBANN. La première méthode, qui s'appelle SUBSET, est similaire aux approches proposées par Saito et Nakano [SAI 88], et par Fu [FU 91]. La deuxième méthode a été crée par Towell et Shavlik [TOW 91] et s'appelle algorithme NofM. Tout d'abord, nous allons présenter les propriétés et les caractéristiques qui sont communes aux deux types de méthodes, puis nous allons détailler chacune de ces méthodes dans les section suivantes.

Deux hypothèses sont faites par rapport aux réseaux obtenus par compilation et raffinés par apprentissage :

1. Les unités des réseaux KBANN peuvent être assimilées à des unités à seuil : lorsque leurs valeurs d'entrée (somme pondérée) sont en dessous du seuil qui leur correspond, leur activation est proche de 0.0, sinon elle est proche de 1.0. La fonction d'activation utilisée est toujours la sigmoïde asymétrique, mais selon la spécification de certains paramètres (e.g. la valeur de 'W'), cette fonction peut être assimilée à la fonction seuil [JOD 94a, TOW 93].

2. L'apprentissage ne modifie pas d'une façon significative le rôle qu'une unité avait avant l'apprentissage. Ainsi, nous pouvons garder les identificateurs (étiquettes) des unités qui leur ont été attribués en correspondance aux règles symboliques compilées. Les règles extraites à partir d'un réseau sont plus facilement interprétables, car elles conservent les noms attribués aux prémisses et conclusions des règles symboliques.

Les deux méthodes s'appuient sur l'idée que l'on doit trouver les combinaisons des valeurs d'entrée d'une unité qui font que son activation de sortie soit proche de 1.0. Cette combinaison d'entrées nécessaire à l'activation d'une unité va être exprimé sous la forme d'une règle symbolique. Par conséquence, les méthodes d'extraction cherchent à étudier les relations établies entre l'ensemble des entrées et la valeur du seuil d'activation d'une unité. Une valeur résultant de la somme pondérée des entrées d'une unité supérieure au seuil implique que cette unité soit activée (valeur d'activation proche de 1.0), et par conséquence on va exprimer cette relation entre les entrées et l'activation de la sortie par une règle (e.g. Si entrée1 et entrée2 et non(entrée3) Alors unité-conclusion est active). Une valeur résultante de la somme pondérée des entrées d'une unité inférieur au seuil implique que cette unité ne va pas être activée (valeur d'activation proche de 0.0), et par conséquence on ne va pas exprimer par une règle la relation entre les entrées et la sortie de cette unité.

Puisque les unités des réseaux KBANN ont des entrées toujours très proches de 0.0 ou de 1.0, cela va beaucoup simplifier la recherche des relations valables entre les entrées et la sortie. Nous considérons comme valables les relations dont la sortie de l'unité est activée, c.-à-d. quand on a 'Ai > 0.5'. Ce sont les 'relations valables' qui donnent l'origine aux règles extraites. Dans le cas des réseaux KBANN, l'espace de recherche est réduit à cause des caractéristiques particulières de ces réseaux que l'on vient de décrire. Ainsi, on peut résumer le problème de l'extraction de règles comme suit :

♦ Entrée = 0.0	⇒ Poids = +Wij	⇒ Wij.Xj = 0.0	N'ont pas d'effet sur
♦ Entrée = 0.0 (niée)	⇒ Poids = -Wij	⇒ Wij.Xj = 0.0	la somme pondérée

♦ Entrée = 1.0	⇒ Poids = +Wij	⇒ Wij.Xj= +Wij	Préservent le signe et la
♦ Entrée = 1.0 (niée)	⇒ Poids = -Wij	⇒ Wij.Xj = -Wij	valeur des poids

Sachant que : $Ai = Fa\ (\Sigma_j\ Wij.Xj + \Theta i)$

$Fa(Si) \approx 1.0$ quand $Si > 0.0$

$Fa(Si) \approx 0.0$ quand $Si < 0.0$

Alors on cherche à savoir si : $\Sigma_j\ Wij.Xj + \Theta i > 0.0$

Comme nous savons que $\Theta i < 0.0$ et $Xj \approx 0.0$ ou 1.0, ce qu'il faut trouver ce sont toutes les combinaisons des poids qui dépassent la valeur du seuil : $\Sigma_j Wij > \Theta i$

Finalement, il est important de souligner que cette façon simplifiée de traiter les sorties des activations et de faire la recherche des 'relations valables', a pour résultat l'obtention d'un ensemble de règles qui n'est pas forcement tout à fait égal au réseau qu'il est sensé représenter. Une base de règles obtenue par extraction à partir d'un réseau peut contenir des erreurs d'omission ou de généralisation excessive par rapport à ce réseau. Ces erreurs sont généralement le résultat d'un cumul d'effets des poids faibles qui ne sont pas représentés dans les règles extraites. Un des principaux problèmes de l'extraction de règles est la minimisation de ce type d'erreurs tout en évitant de simplement réécrire le contenu du réseau dans un langage de plus haut niveau.

3.3.2.3.a Algorithme SUBSET d'extraction de règles

L'algorithme SUBSET est une méthode similaire aux approches décrits par Saito et Nakano, ainsi que par L. Fu [SAI 88, FU 91]. Il a été nommé SUBSET car on cherche les sous-ensembles des poids d'entrée d'une unité qui cumulés permettent de depasser son seuil d'activation. Donc, le principe de l'algorithme SUBSET va être de trouver des combinaisons d'unités entrantes (connectés à l'entrée de l'unité en question) qui doivent être activées ou inactivées pour que l'unité correspondante soit activée. Pour chaque combinaison obéissant à cette définition, on crée et on ajoute à la base de règles la règle ayant en prémisses les noms des unités devant être activées et les noms des unités devant être inactivées, et en conclusion, le nom de l'unité étudiée.

On essaie, pour cela, de trouver des sous-ensembles d'unités reliées par des poids positifs dont la somme dépasse le seuil de l'unité étudiée. Pour chaque groupe de poids positifs ainsi exhibés, on regarde parmi les poids négatifs, s'il en existe dont la somme des poids annule les poids précedents plus le seuil. S'il existe de tels poids, alors, les entrées respectives doivent être niées, afin que le groupe de poids positifs extrait active l'unité, sinon, l'activation de ce groupe de poids suffit pour produire l'activation de la sortie de l'unité en question. A cet effet, l'algorithme suivant présenté dans le Tableau 6 a été développé.

Pour chaque sous-ensemble Bp de poids positifs reliés à l'unité Ui

    Si Somme(Bp) + Θi > 0.0

    Alors ajouter Bp à Gp

Pour chaque ensemble P ∈ Gp

    Pour chaque sous-ensemble Bn de poids négatifs reliés à Ui

        Soit Z un nouveau prédicat non utilisé jusqu'à présent

        Si Somme(Bp) + Somme(Bn) + Θi < 0.0

        Alors CréerRègle("Z :- Bn")

    Si Z a des antécédents

    Alors CréerRègle ("Ui :- P, non(Z)")

    Sinon Ui :- P

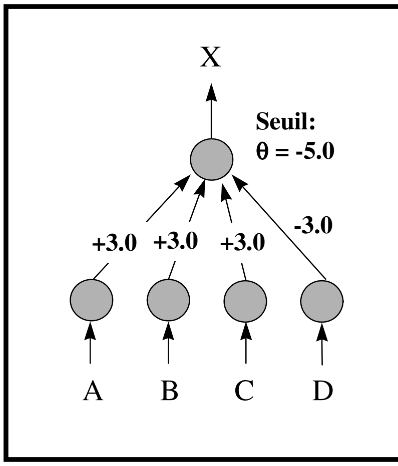
Elimination des règles dupliquées

Ui : Unité 'i' de la couche cachée ou de la couche de sortie

Θi : Seuil d'activation de l'unité 'i'

Tableau 6 - Description de l'algorithme SUBSET

La Figure 30 présente un exemple de l'application de l'algorithme SUBSET à une unité d'un réseau KBANN. Les régles qui y sont représentées, sont le résultat obtenu par l'application de cet algorithme.



X := A, B, C.
X := A, B, non(D).
X := A, C, non(D).
X := B, C, non(D).

<u>Si</u> (A et B et C) ou
(A et B et Non(D)) ou
(A et C et Non(D)) ou
(B et C et Non(D))
<u>Alors</u> X

Figure 30 - Exemple d'application de l'algorithme SUBSET

L'avantage de cette technique est que les règles qui en sont extraites sont compréhensibles et faciles à exploiter. Par contre, la complexité de cet algorithme impose un temps d'exécution qui peut augmenter significativement en fonction du nombre d'entrées de l'unité analysée, car il exécute une recherche de toutes les possibilités de combinaisons des entrées/poids qui mènent à l'activation de la sortie. De plus, cet algoritme doit être appliqué à chaque unité de la couche cachée et de la couche de sortie du réseau. En conséquence, le nombre de règles exhibées augmente très rapidement en fonction du nombre d'entrées et d'unités du réseau, ce qui limite parfois la capacité d'interprétation des règles par un expert et par conséquence son utilisation pratique.

3.3.2.3.b Algorithme NofM d'extraction de règles

Towell, ainsi que les autres chercheurs qui étudient des techniques d'extraction de règles à partir des RNA, ont cherché à perfectionner ces méthodes et à améliorer la qualité des règles obtenues par leurs algorithmes d'extraction. Les critères qu'on utilise pour mesurer la qualité des règles et des algorithmes d'extraction sont, par exemple [AND 95, BOZ 95] :

• L'*exactitude* des règles obtenues : sert à indiquer si les règles sont capables de bien classer les exemples appris et si elles ont une bonne capacité de généralisation;

• La *fidélité* des règles : vérifie si les règles sont capables de bien reproduire le comportement du réseau à partir duquel elles ont été obtenues;

• La *consistance* des règles : sert à déterminer si les résultats obtenus avec la base de règles sont assez proches de ceux obtenus à partir de l'utilisation du réseau, même dans des conditions d'utilisation qui soient différentes de celles rencontrés pendant l'apprentissage;

• L'*intelligibilité* des règles : mesure la difficulté de compréhension des connaissances codées dans la base de règles. Cette mesure est basée principalement sur deux paramètres : le nombre total de règles extraites (moins de règles ⇒ plus d'intelligibilité, plus de règles ⇒ moins d'intelligibilité), et le nombre d'antécédents moyen de chaque règle;

• La *complexité* de l'algorithme d'extraction.

L'algorithme NofM (appelé aussi de MofN) créé par Towell [TOW 91, TOW 93] dans le cadre du système KBANN, a été développé pour pallier les inconvénients de l'algorithme SUBSET et pour essayer d'augmenter la qualité des règles et de l'algorithme d'extraction, selon les critéres que nous venons de décrire. Le principe de fonctionnement de l'algorithme NofM est très proche de celui de l'algorithme SUBSET : on analyse chaque unité de façon séparée en essayant de trouver des relations entre celle-ci et ses antécédents. La première différence se situe au niveau de la forme des règles extraites, qui sont exprimées en fonction d'une primitive de base : NofM. Cette primitive prend en entrée un entier et une base de faits et rend en sortie un booléen. La formule $NofM(i, P_1, P_2, ..., P_n)$ rend vraie (1.0) sa sortie ssi au moins i propositions parmi $P_1, ..., P_n$ sont vraies, sinon elle donne la valeur fausse (0.0). Cette forme a l'avantage d'être plus réduite et nécessite moins de règles pour exprimer les connaissance. Par exemples, les règles extraites de l'exemple présenté dans le paragraphe précédent peuvent s'exprimer par la seule règle : X <- NofM (3, A, B, C, non (D)).

Cet algorithme fonctionne selon le principe suivant, en six étapes :

1. **Regroupement (*clustering*)** : on partitionne, pour chaque unité, l'ensemble des liaisons d'entrée en classes d'équivalence, les liaisons d'une même classe d'équivalence portant des poids dont les valeurs sont proches les unes des autres. La *'proximité'* des valeurs des

poids est établie par un paramètre fourni à l'algorithme d'extraction. KBANN utilise une valeur constante de distance inférieur à 0.25 pour regrouper les valeurs des poids des connexions.

2. **Moyenne (*averaging*)** : ensuite, on fixe la valeur des poids des liaisons d'une même classe à la moyenne des poids de cette classe.

3. **Elagage (*eliminating*)** : on détermine, de façon heuristique, les classes non significatives pour l'unité étudiée, c'est-à-dire les classes dont la valeur d'activation est indifférente pour l'activation de l'unité dont on extrait les règles. Pour cela, on procède sur une base d'exemples: pour chaque exemple, on calcule l'activation du réseau, et on regarde si, en annulant toutes les activations des poids d'une classe particulière, l'activation de l'unité étudiée est modifiée. Si aucun changement notable n'est observé, alors la classe n'est pas significative pour l'exemple courant. Une fois que toute la base a été examinée, on enlève les classes qui ne sont significatives pour aucun exemple de la base.

4. **Optimisation (*optimizing*)** : des modifications sur le seuil de l'unité étudiée peuvent être nécessaires, du fait de l'erreur que peut apporter la partition en classes d'équivalence de ses poids d'entrées, ainsi que la suppression des classes non significatives. Pour cela, on recommence un apprentissage, en gelant tous les poids portés par les liaisons, de manière à ce que seuls les seuils des unités du réseau soient modifiés.

5. **Extraction (*extracting*)** : on exprime ensuite, pour chaque unité du réseau, les relations entre l'unité étudiée et ses antécédents sous la forme d'une inéquation. Cette inéquation est exprimée de la manière suivante :

   - Soit Θ le seuil de l'unité étudiée;
   - Soit C_1, C_2, ..., C_n les identificateurs des liaisons entrantes, portant les poids W_1, W_2, ..., W_n;

- L'inéquation associée à chaque unité est la suivante :

$$W_1*NTrue(C_1)+W_2*NTrue(C_2)+ ... + W_n*NTrue(C_n) + \Theta >0.$$

NTrue est une primitive prenant en entrée une liste de faits, et rendant en sortie un entier qui représente le nombre de faits justes (vrais) parmi ceux de la liste d'entrée.

6. **Simplification (*simplifying*)** : on réécrit enfin les inéquations sous forme de règles de type NofM, en recherchant, dans chaque groupe, le nombre de règles permettant à l'unité responsable de la conclusion d'être activée. Par exemple :

L'unité X possède des liaisons d'entrée à partir des antécédents identifiés par A, B, C, D, réliées par des connexions de poids égaux à 5.1, et des antécédents identifiés par M, N, O, réliées par des connexions de poids égaux à 3.5. L'unité X a un seuil $\Theta = -10.0$. Cette unité est représentée par l'équation suivante :

X :- 5.1 * NTrue (A, B, C, D) + 3.5 * NTrue (M, N, O) > 10.0

Après simplification cet équation va être représentée par les règles suivantes :

X :- NofM(2, A, B, C, D).

X :- NofM(1, A, B, C, D), NofM(2, M, N, O).

X :- M, N, O.

La Figure 31 montre un exemple du processus complet d'extraction de règles par la méthode NofM. Cet algorithme fourni des ensembles de règles assez compacts qui sont usuellement beaucoup plus petits que les ensembles de règles obtenus par l'autre algorithme, SUBSET. Le fait d'exprimer les règles sous une forme plus compacte peut nous permettre d'en réduire le nombre, et ainsi d'en augmenter la compréhension. Par contre, les règles du type NofM sont moins naturelles pour un utilisateur humain, et, même si elles sont plus compactes, elles n'augmentent pas forcément l'intelligibilité de l'ensemble.

En ce qui concerne la complexité des algorithmes d'extraction de KBANN, Towell a déterminé la complexité de l'algorithme SUBSET comme étant de l'ordre exponentielle et la complexité de l'algorithme NofM, dans la pratique, comme étant de l'ordre cubique. En effet, la complexité de NofM est assez difficile à analyser avec précision en raison de la phase d'apprentissage pour l'optimisation des seuils des unités. Towell estime la complexité de l'apprentissage à $O(n^3)$ et les autres étapes de l'algorithmes ont une complexité de O(n), sauf pour l'étape de regroupement des poids qui est de O(n*log(n)). En bref, cet algorithme peut être efficace pour extraire des règles d'unités ayant beaucoup de liaisons, mais semble mal adapté dans le cas où peu de liaisons sont en jeu et la phase d'apprentissage finit par être plus importante.

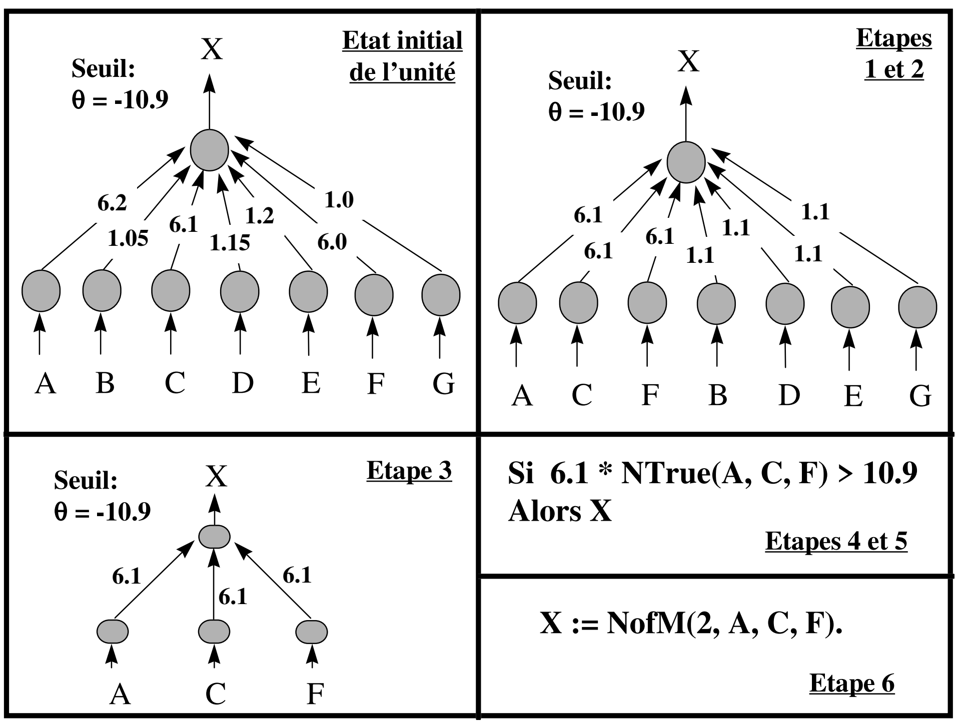


Figure 31 - Exemple d'application de l'algorithme NofM

3.3.2.4 Discussion sur les Réseaux KBANN

Ce système est très intéressant, car il combine de manière très forte une optique symbolique et une optique connexionniste. Cette méthode d'intégration nous permet de tirer pleinement profit de la complémentarité entre ces deux types de modélisation. Par contre, son inconvénient majeur est que, lors de l'apprentissage, il ne modifie pas la structure du réseau, mais seulement les valeurs des poids entre les cellules. Nous allons dresser ici une liste des points forts de KBANN suive d'une autre liste des limitations dont nous pensons qu'elles constituent les points faibles les plus importants de ce type de réseaux.

Les réseaux KBANN présentent des points forts tels que :

- Le transfert bidirectionnel des connaissances entre les modes de représentation connexionniste et symbolique est réalise de manière simple et directe. Les réseaux KBANN nous permettent de profiter en même temps des connaissances théoriques disponibles (règles symboliques) et des connaissances pratiques disponibles (exemples). Ce type de réseau essaie d'utiliser et d'intégrer toutes les connaissances disponibles sur le problème;
- L'insertion de connaissances (règles) dans les réseaux permet d'accélérer significativement le processus d'apprentissage et de résoudre le problème de la spécification d'une topologie initiale et des poids initiaux des réseaux;
- La structure du réseau obtenue par compilation des règles garde une correspondance directe avec la base de règles d'origine. Les concepts intermédiaires y sont préservés;
- Les réseaux KBANN permettent de réaliser un raffinement d'une base de règles approximativement correcte par apprentissage d'exemples;
- L'apprentissage des réseaux KBANN peut être fait à partir d'une base d'exemples plus petite étant donné que le réseau initial peut coder des connaissances a priori sur le problème;
- Les réseaux KBANN sont relativement robustes pour le traitement de problèmes représentés par des bases de règles et d'exemples qui contiennent des données incomplètes ou incorrectes;

- Les méthodes d'extraction de règles utilisées permettent de bien retrouver les connaissances compilées dans le réseau et raffinées par la suite. Ces méthodes profitent du fait que le réseau préserve dans sa structure des liens directs avec la base de règles d'origine;

- Deux méthodes d'extraction de règles à partir des réseaux sont proposées dans KBANN : la méthode SUBSET et la méthode NofM. En particulier, la méthode NofM exploite l'idée de l'extraction d'un type particulier de règles, avec un algorithme plus performant que SUBSET (sous certaines conditions). Ces règles du type NofM ont un plus grand pouvoir d'expression des connaissances et elles permettent de coder les connaissances d'une façon plus compacte.

Cependant, les réseaux KBANN possèdent un certain nombre de points faibles et limitations, à savoir :

- Les règles symboliques sont des propositions d'ordre 0. Les réseaux KBANN ne sont pas très bien adaptés pour traiter des entrées avec des valeurs continues, sauf si celles-ci sont préalablement traitées (e.g. discrétisation des valeurs continues). Les valeurs continues sont pour KBANN des entrées entre 0.0 et 1.0 qui représentent la 'certitude' ou la 'proximité' des valeurs par rapport aux concepts qu'elles représentent (le valeur 0.5 représentant justement l'indécision);

- L'algorithme d'apprentissage représente un des plus gros défauts des réseaux KBANN. Cet algorithme, une version très proche de la Rétro-Propagation du gradient, présente un bon nombre d'inconvénients :

    - ♦ L'apprentissage est fait sur des réseaux statiques. Les changements subis par les réseaux pendant la phase d'apprentissage ne permettent d'exploiter que l'espace des poids, et la structure des réseaux reste toujours inaltérée. Towell propose l'inclusion d'unités dans les réseaux KBANN, mais cela est fait de façon 'manuelle' et avant apprentissage;

    - ♦ L'algorithme de la Rétro-propagation ne marche pas très bien dans les réseaux composés d'un très grand nombre de couches. La méthode de la rétro-propagation de l'erreur vers les couches précédentes ne permet pas d'estimer l'erreur précisément. Ainsi, les ajustements des unités sont de moins en moins précis. Malheureusement, le processus de création de réseaux par compilation de règles obtient la plupart du temps des réseaux avec une quantité assez importante de couches, car elle préserve la hiérarchie présente dans la base de règles symboliques;

- Towell a choisi d'employer un algorithme classique de rétro-propagation et n'a pas pris en compte le fait qu'on dispose maintenant de nombreuses techniques beaucoup plus performantes (e.g. Quick-Prop, R-Prop, Gradient Conjugué). L'algorithme de la Rétro-Propagation présente des inconvénients tels que : la faible vitesse de convergence, la difficulté de réglage des paramètres (α et β), la paralysie de l'apprentissage et l'oubli catastrophique.
- Une analyse et des critiques de la méthode d'apprentissage des réseaux PMC par Rétro-Propagation a été présenté dans la Section 2.3.3.

• Les réseaux KBANN ne marchent pas très bien dans les cas où la base de règles a des problèmes significatifs d'incomplétude ou d'incorrection. Un réseaux KBANN est capable d'ajouter/effacer des antécédents dans les règles, mais il est incapable de créer des nouvelles règles (unités) [OPI 95a]. Les changements nécessaires dans la base de règles ne doivent pas être très importantes pour que l'algorithme d'apprentissage de KBANN puisse fonctionner;

• Les réseaux KBANN doivent préserver les significations d'origine associés aux unités à partir du processus de compilation de règles. L'étiquette associée à une unité n'est valable que si, après l'apprentissage, cette unité conserve son rôle d'origine. Le problème est qu'on ne peut pas être sûr que l'apprentissage ne va pas réaliser un déplacement de la signification des unités (*meaning shift problem*);

• L'extraction de règles dans KBANN implique l'analyse de toutes les unités du réseau. Les méthodes utilisées dans ce type de réseaux ne permettent pas de réaliser une extraction orientée seulement vers les nouvelles connaissances acquises, car l'algorithme d'apprentissage réalise des adaptations sur toutes les unités du réseau. Les règles d'origine qui ont été compilées sous la forme d'un réseau peuvent être modifiées et ainsi les connaissances de départ sont mélangées avec les nouvelles connaissances acquises par apprentissage. Par conséquence, l'algorithme d'extraction doit analyser toutes les unités du réseau afin d'extraire les connaissances qui y sont représentées;

• L'algorithme d'extraction de règles utilisé dans les réseaux KBANN est orienté spécifiquement vers ce type de réseaux, c'est-à-dire qu'il lui faut un réseau avec des sorties 'binaires' (les sorties sont toujours très proches de 0.0 ou de 1.0). Les algorithmes utilisés dans KBANN ne permettent pas de manipuler des valeurs de sortie continues, d'extraire des règles floues, et principalement d'utiliser des valeurs d'entrée 'vraiment' continues;

• Le type de règles symboliques utilisées dans les réseaux KBANN (insertion et extraction) ont une capacité limitée de représentation de connaissances. Nous pensons qu'il faut étudier la possibilité d'utiliser des règles de plus haut niveau pour arriver à une meilleure représentation des connaissances sur certains problèmes (voir Section 4.2.1 et Section 5.3). La capacité de généralisation fournie par un système est aussi limitée par sa capacité de représenter les connaissances.

Nous avons essayé de retenir les points positifs de KBANN, mais aussi de corriger ses points faibles et ses limitations, afin d'obtenir un SHNS plus performant - le système INSS. Pour cela, nous avons choisi de travailler sur un modèle de réseau incrémental avec un apprentissage plus performant, ce qui nous a permis de résoudre une bonne partie des problèmes que nous venons de citer.

3.3.3 Systèmes Connexionnistes de Raffinement de Connaissances

Nous allons discuter brièvement ici les approches connexionnistes de raffinement de connaissances (*connectionist theory refinement*). Les systèmes KBANN et SYNHESYS étant deux exemples typiques de ce type d'approche. En particulier, nous nous sommes intéressés au système KBANN puisqu'il est basé sur des réseaux de type PMC. Ce système présente une forte limitation, car il ne permet pas de faire évoluer la structure de ces réseaux. Par contre, cela ne pose pas de problèmes aux réseaux à prototypes, comme ceux utilisés par SYNHESYS. Cependant, nous trouvons dans la littérature d'autres systèmes connexionnistes de raffinement de connaissances qui sont très proches ou qui sont dérivés de KBANN, dont certains permettent même de faire évoluer la structure du réseau.

Le premier exemple des méthodes dérivées à partir de KBANN sont les algorithmes de construction de réseaux proposés par D. Opitz : TopGen, Regent et Addemup [OPI 95a, OPI 95b, OPI 97]. Ces trois méthodes utilisent les algorithmes de base de KBANN pour la construction d'un réseau initial suivi d'un raffinement des connaissances par apprentissage en utilisant la Rétro-Propagation. La différence entre ces algorithmes et KBANN vient du fait qu'ils permettent de faire évoluer l'architecture des réseaux à travers la génération de plusieurs topologies de réseaux et de l'application d'algorithmes génétiques. Plusieurs réseaux possédants des structures légèrement différentes sont soumis à un processus d'apprentissage avec le même algorithme employé dans KBANN. Ensuite, sont appliquées des techniques de sélection du meilleur réseau (sauf dans le cas de la méthode Addemup, où l'on cherche une architecture modulaire issue d'une combinaison de plusieurs réseaux). Ces trois techniques sont assez intéressantes, mais on peut les classer comme des méthodes du type "force brute", étant donné qu'ils sont très lourds du point de vue computationnel. De plus, les recherches développées aboutissent à des méthodes où l'on ne se pose jamais de questions à propos de la qualité des données d'apprentissage. Les "règles approximativement correctes" compilées dans un réseau sont raffinées à partir d'une base d'apprentissage, mais on n'étudie pas beaucoup les effets résultant de l'utilisation de données "approximativement correctes" dans cette base. Le seul but de ces méthodes est d'obtenir la meilleure généralisation possible à partir des informations disponibles, sans chercher à interpréter les connaissances acquises, comme par exemple, à travers l'identification des règles incorrectes et des nouvelles règles acquises. Les méthodes d'extraction de règles, même s'il n'y a pas de raison apparente qui nous empêche de les utiliser sur les réseaux obtenus par TopGen, Regent ou Addemup, n'ont pas été appliquées sur ces réseaux.

Un autre exemple de réseau connexionniste de raffinement de connaissances, permettant de faire évoluer sa structure, a été développé par J. Fletcher [FLE 93, FLE 94]. Il propose d'utiliser un réseau obtenu par compilation de règles comme une des unités cachées d'un réseau à trois couches (reliée aux unités d'entrées et aux unités de sortie). Les autres unités cachées du réseau sont ajoutées par un algorithme constructif appelé HDE [FLE 94]. Cette approche est intéressante puisqu'on préserve toutes les connaissances introduites par compilation, mais il présente des inconvénients tels que:

- Les réseaux ont une structure restreinte à seulement trois couches;

- Nous ne pouvons pas profiter des connaissances intermédiaires contenues dans les unités obtenues par compilation. Le réseau de départ reste une boîte noire incrustée à côté des unités du réseau constructif;

- Les unités ajoutées dans la couche cachée du réseau sont des unités qui ne semblent pas être compatibles avec les algorithmes d'extraction de règles utilisées dans les réseaux KBANN. La sortie d'une unité n'est pas forcement toujours proche de 0.0 ou de 1.0;

- Les travaux faits par Fletcher sont concentrés surtout sur la construction de réseaux par ajout d'unités. Dans sa thèse, il n'a pas développé d'études plus approfondies sur l'insertion de règles dans un réseau, ni sur la possibilité d'extraire des règles à partir d'un réseau raffiné.

Notre intérêt pour les travaux développés par Fletcher se concentre uniquement sur l'idée de garder le réseau d'origine obtenu par compilation de règles et d'employer un algorithme constructive pour faire évoluer l'architecture du réseau. Les études menées par Opitz [OPI 95a] ont démontré que cette approche est assez limitée par rapport à sa capacité de raffiner et de corriger les connaissances de départ.

Les travaux développés par J. Mahoney dans le cadre de son système RAPTURE [MAH 93, MAH 96], ont abouti à la création d'un système de raffinement de connaissances exprimées par des règles probabilistes (utilise des coefficients de certitude associés aux règles). Le système RAPTURE utilise une méthode de construction de réseaux, basée sur l'algorithme Upstart [FRE 90], qui fait l'ajout d'unités dans la couche cachée connectées directement aux entrées et à une unité de sortie. La méthode employée permet d'obtenir des réseaux assez semblables à ceux obtenus par Fletcher, sauf que dans ce cas Mahoney utilise des réseaux obtenus à partir de règles avec des coefficients de certitude. A notre avis, les principales limitations du système RAPTURE sont :

- Il fait l'apprentissage jusqu'à arriver à 100% de réponses correctes sur la base d'apprentissage, ce qui peut mener à un surapprentissage;

- Il n'est pas capable de traiter des problèmes avec des contradictions (base avec des exemples incorrects);

- Il travaille de la même façon que KBANN par rapport aux entrées continues, et donc il possède les mêmes limitations que KBANN par rapport au traitement de ce type de données;

- Les unités ajoutées dans les réseaux vont constituer une structure toujours composée par seulement trois couches (cela n'est pas le cas des règles compilées sous forme de réseau).

De plus, l'algorithme d'apprentissage employé possède des paramètres difficiles à régler tel l'algorithme classique de la Rétro-Propagation du gradient. Dans sa thèse[MAH 96], Mahoney propose comme futurs travaux possibles, l'étude d'autres techniques connexionnistes pour améliorer les performances de son système. Il indique aussi que l'on pourra interagir avec les experts afin de mieux comprendre les erreurs présentes dans la base de connaissance théorique et étudier les nouvelles règles acquises par le système par apprentissage d'exemples.

Le dernier type de système de raffinement de connaissances que nous allons aborder ici, sont les systèmes basés sur des réseaux récurrents. Dans cette catégorie de systèmes, les approches qui permettent la construction de réseaux à partir de FSAs (*Finite State Automata* - Automates à états finis), ainsi que l'extraction de FSAs à partir des réseaux, font partie d'un des principaux axes de recherche de ce domaine [OML 94, OML 93, OML 96a, FRA 95, FRA 96a, FRA 96b, FRA 92, FRA 91]. Plus récemment, nous trouvons aussi des recherches qui ont étendu ces systèmes aux automates flous (*fuzzy-finite state automata*) [OML 96b]. Cependant, comme nous avons déjà dit antérieurement, nous avons choisi de ne pas travailler sur les réseaux récurrents et par conséquence nous n'avons pas approfondi nos recherches dans cette direction. Toutefois, cette voie de recherche semble être très intéressante. Dans un futur proche, nous comptons pouvoir étudier les possibilités de faire évoluer notre système INSS afin de pouvoir exploiter certaines des techniques utilisées dans cette approche.

3.4 CONCLUSION SUR L'ETUDE DES SHNS

Dans les sections antérieures, nous avons orienté le sujet de discussion de cette thèse vers l'étude des systèmes experts intelligents pour la classification. Notre intérêt s'est porté principalement sur un type particulier de systèmes, les SHNS (Systèmes Hybrides Neuro-Symboliques) avec un mode d'intégration par co-traitement. Dans ce cadre, nous avons présenté dans ce chapitre un aperçu des SHNS, de leur classification et de leurs propriétés. Nous avons aussi mis en valeur les processus de transfert de connaissances entre les modules symbolique et connexionniste, et nous avons fini en présentant d'une façon plus détaillée, les deux systèmes hybrides qui nous ont le plus influencé dans nos recherches : le système SYNHESYS et le système KBANN. Ces deux systèmes présentent des propriétés très intéressantes que nous avons envisagé d'incorporer dans notre système, tout en améliorant leurs points faibles et leurs limitations.

Nous avons essayé de présenter un état de l'art des SHNS dans ce dernier chapitre. Malheureusement, il n'est pas possible d'aborder ici un thème si étendu d'une façon exhaustive et complète. Nous avons choisi de diriger nos études sur certains aspects particuliers des SHNS et de faire une analyse plus complète des systèmes SYNHESYS et KBANN. Ainsi, nous laissons au lecteur la possibilité de consulter d'autres ouvrages plus généraux sur certains thèmes plus vastes non abordés complètement par nous. Le premier exemple, est celui des publications de O. Boz [BOZ 95], de R. Andrews [AND 95] et de L. Decloedt [DEC 96] qui abordent le sujet des transferts de connaissances d'une façon plus complète. En ce qui concerne l'extraction de règles à partir des réseaux, Andrews décrit plusieurs méthodes qui sont classées en deux groupes : *méthodes decompositionnelles* (extraction de règles à partir de l'analyse de chaque unité et de ses composants - poids de connexions et seuil) et *méthodes pédagogiques* (extraction de règles à partir de l'étude du comportement global du réseau sans considérer ses composants). Dans les sections antérieures, nous ne décrivons que des méthodes d'extraction de règles du type decompositionnel. L'autre exemple, est constitué par les études sur l'intégration de systèmes hybrides, qui forment un thème assez complexe et qui est encore un vaste domaine de recherche, tel que l'a bien montré le projet MIX et les chercheurs qui ont participé de ce projet [HIL 94a, ORS 95, LAL 96].