Damien Nouvel
Le 28/07/2002

Compte-rendu de lecture dirigée

Sujet : l'apprentissage de graphes conceptuels

sous la direction de **M. G. Mineau**

Objectifs :

Ce cours vise à initier l'étudiant à certains formalismes de modélisation conceptuelle à l'aide de l'apprentissage automatique, techniques qui peuvent être utilisées dans le but de résumer des textes. Les lectures seront centrées sur la théorie des graphes conceptuels, et complétées par des méthodes d'apprentissage automatique de graphes.

Ces techniques sont étudiées pour l'extraction de connaissances à partir de textes et pour la génération de textes à partir de connaissances. Ces deux techniques, complémentaires et symétriques reposent sur un formalisme commun : celui du langage naturel. Celui-ci peut être représenté sous forme de graphes conceptuels. Le but de cette étude est de déterminer comment il est possible d'apprendre les règles qui régissent ces processus d'abstraction et de synthèse de sémantique.

PLAN :

INTRODUCTION :

La lecture dirigée présentée ici consiste en une étude du formalisme des graphes conceptuels, ses structures en tant que base de connaissances et les algorithmes permettant son implémentation. Ces éléments d'étude sont issus d'une revue de la littérature sur les graphes conceptuels, constituée d'une vingtaine de chapitres et d'articles scientifiques, sous la direction de G. Mineau.

La théorie des graphes conceptuels est inspirée des travaux de C. S. Peirce, philosophe américain du début du vingtième siècle, qui présenta les *graphes existentiels*, une représentation graphique de la logique du premier ordre. Ces travaux, longtemps méconnus, ont été plus particulièrement étudiés lors de l'avènement des nouvelles technologies.
Par ailleurs, de nombreuses représentations des connaissances ont émergé dans les années septante et octante, avec l'aide des sciences cognitives, et ont donné lieu à l'apparition de systèmes à base de connaissances qui utilisent les réseaux sémantiques.
Les travaux de J. F. Sowa ont approfondi la logique des graphes existentiels à la lumière des solutions apportées par les réseaux sémantiques. Un modèle cognitif en est issu, sous le nom de *graphes conceptuels*, donnant lieu à de nombreuses et diverses implémentations.

Les graphes conceptuels sont donc une représentation à la fois computationnelle et interprétable, compromis entre la logique trop souvent abstraite et le langage naturel trop souvent illogique. Ils ont alors permis de construire d'une manière formelle, compréhensible et manipulable des systèmes à base de connaissance possédant des propriétés et des règles très précises.
Plus particulièrement, le typage des données, organisé par une relation d'ordre sous forme de hiérarchie, a permis à la fois de mieux manipuler et de mieux interpréter les graphes conceptuels. Nous accorderons une importance plus particulière aux types tout au long de ce rapport : leur définition, leurs propriétés et la relation d'ordre qui en découle. Nous verrons également comment cette relation d'ordre, établie sur les types, peut également être appliquée de manière très efficace sur les graphes eux-mêmes.

Nous présenterons en premier lieu le formalisme des graphes conceptuels, partant des éléments de base : les concepts et les relations, leur typage et leur quantification, plus quelques variations et extensions du modèle : les contextes, les définitions. En deuxième partie, nous étudierons les opérations sur les graphes conceptuels : les règles de formation canonique et la structure qui en découle directement : le canon. Enfin nous verrons quelques structures de plus haut niveau basées sur les hiérarchies de graphes conceptuels, l'organisation de ces structures, les algorithmes qui l'implémentent et les problématiques liées à la maintenance d'un ensemble de graphes comme base de connaissance.

FORMALISME DES GRAPHES CONCEPTUELS :

Les nœuds des graphes conceptuels :

Concepts et relations :

Les graphes conceptuels sont des graphes, c'est-à-dire un ensemble de nœuds liés entre eux. Chaque graphe va nous permettre de représenter une unité d'information cohérente et complète. Afin de définir le graphe, nous commençons par présenter les briques élémentaires, les nœuds de ces graphes.

Dans le formalisme présenté par [SOWA84], les nœuds constituant les graphes peuvent appartenir à l'un des deux genres suivants : les *concepts* et les *relations*. Comme leurs noms le signifient, les concepts représentent des objets, les relations qualifient les interactions qui peuvent exister entre ces objets dans le cadre d'un graphe.
Pour illustrer ce propos, nous pouvons prendre comme métaphore la réalisation d'un mur de briques : le mur représente le graphe que nous formons, les briques seraient alors les concepts et le ciment serait constitué des relations qui permettent de lier ces concepts entre eux afin de construire le mur.

Les concepts sont donc des « objets mentaux » qui ont une existence reconnue par le cerveau humain et peuvent être définis. Ils ont éventuellement un numéro, une marque qui les différencient les uns des autres. Nous pourrions utiliser comme concept n'importe quel objet qui nous viendrait à l'esprit, car un concept est théoriquement capable de représenter n'importe quel objet dicible. Cette représentation se rapproche de la composante terminologique des systèmes décrits par [MCGREGOR91].
Dans le formalisme des graphes conceptuels, les concepts sont représentés par des carrés qui entourent l'expression scripturale du concept exprimé. Nous pouvons illustrons ceci par le concept **CHAT** et qui peut alors se représenter simplement de la manière suivante :

Les relations sont des interactions entre ces concepts. Elles servent alors à qualifier la structure des propositions impliquant un ensemble de concepts. La plupart du temps, les relations ne sont pas dicibles : leur utilisation nous servira a priori à représenter dans un système ce qui paraît évident pour le cerveau humain. Cette représentation se rapprochera alors plutôt la composante assertionnelle des systèmes décrits par [MCGREGOR91].
Dans le formalisme des graphes conceptuels, les relations sont de la même manière représentées par des cercles qui entourent l'expression scripturale de la relation exprimée. Nous illustrons ceci grâce à la relation **AGENT**, qui exprime qu'un concept (le sujet) est l'agent d'un autre concept (l'action) et qui peut alors se représenter de la manière suivante :

Les concepts et les relations sont deux représentations qui doivent être définis les uns vis-à-vis des autres au cœur d'un système. Leur réunion forme un monde, définit les capacités du système en termes d'expressivité et d'efficacité. La stratégie d'un système basé sur les graphes conceptuels sera directement issue de la manière dont ces objets ont été répartis.
En général, les relations sont d'un nombre très limité pour faciliter les manipulations et uniformiser la structure des graphes. Mais certains systèmes en viennent à former un nombre plus important de relations pour exprimer plus simplement et plus souplement des assertions complexes.

En somme, selon les besoins du système, il peut-être possible d'augmenter le nombre de relations à utiliser. Cependant les concepts sont dans tous les cas les objets élémentaires. Il est d'ailleurs stipulé par [SOWA84] que théoriquement, une relation peut toujours être définie comme un graphe comportant des relations génériques (nommées LNK), qui met en jeu des concepts correspondant à la définition de la relation. Nous reviendrons plus amplement sur ce sujet lors de l'étude des définitions et des contextes.

Les types :

Les concepts et les relations sont alors des représentations issues de l'esprit humain, les premiers correspondant à des termes, les seconds à des liaisons lors de la construction d'assertions à partir de ces termes. Pour manipuler ces représentations, nous adoptons également une organisation particulière permettant de classer et de comparer ces objets entre eux.

Le formalisme présenté par [SOWA84] organise ces données à l'aide de *types* : ceux-ci permettent de regrouper des concepts ou des relations qui ont des propriétés communes sous forme d'ensembles. Ces types correspondent à une classification inspirée de l'esprit humain, à l'image des réseaux sémantiques, utilisables pour un raisonnement proche de celui de l'homme.
Cependant, il faut faire une distinction avec la notion de type définie dans la littérature informatique : en général, les types sont une marque apposée aux données afin qu'un traitement générique puisse être décliné par des codes différents selon le type de chaque information. Par exemple les traitements pour la saisie ou l'affichage d'une donnée seront techniquement différents si la donnée est un entier ou une chaîne de caractère, car ces types ont été codés différemment sur le disque. Ces types sont très utilisés dans les langages actuels, mais pour la plupart d'entre eux, les types se limitent souvent à décrire quelles manipulations correspondent à chaque codage de données. Conséquemment, ils sont assez peu utilisés pour représenter la sémantique d'une donnée : une structure de plus haut niveau est supposée préciser cela, par exemple à l'aide des objets et des interfaces.
En revanche, dans le formalisme des graphes conceptuels, toutes les données sont codées uniformément : ils n'y a plus de considérations techniques qui obligent à différentier les traitements selon le codage utilisé. Alors les types forment ici une classification qui viendra guider le raisonnement sur les données. La classification qui en résulte servira à comparer des graphes, à trouver des analogies, à manipuler les graphes. De plus, les traitements sont ici uniformisés, quel que soit le type de la donnée. Nous décrirons ces traitements plus loin à l'aide d'opérateurs sur les graphes conceptuels.

Pour la représentation des types, [SOWA84] définit un ensemble de types **T** qui donne tous les types disponibles, ainsi qu'une fonction **type()** qui à chaque concept ou relation associe un type $\mathbf{t} \in \mathbf{T}$. Ces types sont donc souvent partagés entre types de concepts $\mathbf{T_c}$ et types de relations $\mathbf{T_r}$. Nous définissons un type comme un label, car c'est finalement ce qui permet le mieux de décrire un concept ou une relation : son expression scripturale. Du point de vue terminologique de [MCGREGOR91], le label pourrait être associé à l'intention d'un type.
Par ailleurs, il est possible d'obtenir l'ensemble des instances d'un type grâce à la dénotation définie par [SOWA84] : pour n'importe quel type $\mathbf{t} \in \mathbf{T}$, l'ensemble des concepts qui appartiennent à ce type est noté $\boldsymbol{\delta}\mathbf{t}$, cette notation correspond alors à l'extension du type.

La quantification :

Les concepts et les relations sont définis existentiellement : à partir du moment où un concept ou une relation est introduite dans un graphe, nous pouvons en déduire que cette donnée existe. Pour [SOWA84], le monde des graphes conceptuels est sémantiquement ouvert : tout ce qui n'est pas défini peut exister. A fortiori, tout ce qui est acquis à travers un graphe ou par un mécanisme d'inférence peut être défini existentiellement sous réserve qu'aucune donnée identique n'existe déjà.
Mais définir les concepts par leur type n'est pas suffisant. En effet, l'utilité d'un type est de formaliser des regroupements des individus, sous forme d'ensembles. Or pour en conserver tout le bénéfice, il nous faut également pouvoir différencier ces individus, afin de savoir si c'est le même individu qui revient dans plusieurs graphes, ou des individus différents d'un même type.

Pour les concepts, [SOWA84] définit un ensemble de *référents* permettant de distinguer les individus d'un même type. Chaque référent représente un individu unique et permet de le distinguer parmi tous les autres concepts. De plus, un *marqueur* peut-être affecté à un référent, qui distingue un concept de manière explicite, en le nommant.

Cette représentation a depuis beaucoup évolué. Par simplification, nous utiliserons les marqueurs comme [SOWA84] utilise les référents. Un marqueur peut être :

- *implicite* : un numéro affecté automatiquement par la machine à un concept lorsqu'il n'est pas différencié explicitement des autres objets du même type, par exemple **#146592** pourrait être un marqueur affecté à un individu qui n'est pas explicitement nommé, ces concepts sont alors qualifiés de *génériques*,
- *explicite* : un nom donné à un concept récurrent ou à un concept que l'on veut pouvoir repérer facilement par exemple **Roméo** peut-être un marqueur pour un concept de type « **CHAT** » que l'on souhaite nommer ainsi explicitement, ces concepts sont nommés *individus*.

Le système maintient alors un ensemble de marqueurs **M**, constitué d'un ensemble de labels implicites ou explicites. En fait, nous pouvons ajouter que chaque marqueur devrait également contenir le type le plus spécifique qui lui a été affecté : cela permettra de connaître quelles changement de types sont possibles pour ce marqueur, nous considérerons pour la suite que cette information est connue et incluse dans le marqueur.
Une fonction **marker()** associe à chaque concept un marqueur $\mathbf{m} \in \mathbf{M}$. Lorsque l'on ne précise pas de marqueur pour un concept, nous considérerons que le système lui affecte automatiquement un marqueur implicite lors de l'acquisition du graphe conceptuel. Les marqueurs implicites correspondant aux marqueurs génériques de [SOWA84], ils peuvent être aussi conçus comme quantificateurs universels. Nous approfondirons ce point de vue lors de l'étude des opérateurs sur les graphes conceptuels, plus particulièrement la restriction.

Le problème des référents ne s'applique qu'aux concepts. En effet, les relations ne font que décrire la structure d'une proposition. Or une relation est totalement dépendante des concepts qu'elle lie. Deux relations seront identiques si et seulement si les concepts qui les entourent sont identiques.
En outre, on ne compare que rarement des relations pour savoir si elles sont identiques : les comparaisons portent la plupart du temps sur les marqueurs des concepts qui forment les propositions, et les types de relations. Et de surcroît, il arrive excessivement rarement que l'on nomme une relation pour la différencier des autres relations du même type : une relation semble par définition générique et ne pas devoir être individualisée explicitement. Il reste tout de même possible de caractériser une relation par les arcs qui la lient à des concepts : cette information pourrait être utilisée comme le marqueur est utilisé pour les concepts. À ce sujet, les graphes conceptuels sont souvent considéré comme ensemble de relations qui lient des paires de concepts : les triplets.
Pour appuyer ceci, nous reprenons la métaphore précitée du mur de briques : autant nous pouvons nommer et utiliser des noms pour chaque brique que nous utilisons, autant il paraît excessivement difficile et inutile de nommer chaque portion de ciment qui lie deux briques entre elles. Nous savons juste qu'il y a un type de ciment entre chaque paire de briques.

Dans le formalisme donné par [SOWA84], le marqueur implicite (ou générique) ne sera pas spécifié, comme son nom l'indique, et ne donnera donc pas lieu à d'autre représentation que celle du type, faite précédemment pour le concept **CHAT**.
Nous représentons un marqueur explicite par ajout du label correspondant au marqueur après le type d'un concept. Par exemple, nous pouvons représenter le concept « Le chat Roméo » comme suit :

CHAT : Roméo

Les ensembles de concepts :

Malgré l'absence de marqueurs universels dans le formalisme des graphes conceptuels, il est tout de même possible de décrire extensionnellement des ensembles de concepts. Ceux-ci sont alors constitués de plusieurs individus d'un même type. Les ensembles ne s'appliquent qu'aux concepts : les relations, comme précisé précédemment, ne sont pas quantifiées.

Pour utiliser les ensembles, nous devons préciser comment le concept de l'ensemble se factorise en l'ensemble des concepts sous-jacents. À cet effet, [SOWA84] définit plusieurs types d'ensembles :

- *collectifs* : on ne considère que la somme des individus,

- *disjonctifs* : on considère l'un des individus sans savoir lequel,
- *distributifs* : on peut considérer chaque individu à la place de l'ensemble,
- *respectifs* : on considère chaque élément vis-à-vis des éléments d'un autre ensemble.

L'ensemble collectif est largement le plus utilisé, nous allons donc en spécifier l'utilisation. Il peut-être noté de manière implicite comme un regroupement de **n** individus d'un même type par l'adjonction du symbole **@n** après le type du concept. Il est également possible de les préciser explicitement en utilisant la notation ensembliste **{}** et en séparant les éléments par des virgules. Par exemple les concepts « quatre animaux » et « les animaux Roméo et Bob » peuvent être représentés comme suit :

Les autres ensembles donnent lieu à de nombreuses controverses : comment factoriser un ensemble selon les éléments qu'il contient, comment traduire l'ensemble en logique des prédicats ? Nous ne les décrirons pas ici par défaut d'une notation uniformément acceptée.

Par ailleurs, un lien peut-être fait avec les problèmes que constituent les contextes, qui permettent de définir un concept comme un graphe. En effet, si nous disposions d'une relation de type **ET**, nous pourrions caractériser un ensemble collectif comme un graphe et ainsi le faire apparaître dans un contexte. En fait, ces aggregats existent déjà d'une certaine manière avec la relation **PART**, qui précise qu'un concept fait partie d'un autre concept qui pourrait alors être du type **ENSEMBLE** ou **AGGREGAT**. Cela se fait également avec la relation **OR**, permettant la description d'un ensemble disjonctif. Nous discuterons donc de la représentation des ensembles ci-après, lors de la description des contextes.

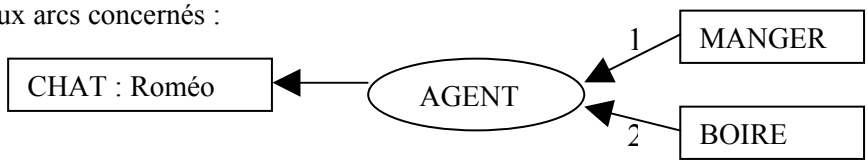
Les arcs :

Les *arcs* forment la structure du graphe conceptuel par liaison des concepts aux relations. Ces arcs donnent tout le sens du graphe en tant qu'assertion. Ils sont, par leur nature, des éléments relationnels du graphe et nous les associerons beaucoup plus facilement aux relations qu'aux concepts. En effet, il est possible de voir un concept exister sans arcs, mais une relation sans arcs n'aurait pas de sens par elle-même. Les arcs forment en quelque sorte l'instanciation de chaque relation, en précisant quels concepts la relation met en jeu.

Il est stipulé par [SOWA84], qu'il n'est pas possible de lier deux concepts entre eux ou deux relations entre elles : chaque arc définit un lien entre une relation et un concept. Ces arcs sont orientés : ils fournissent un ordre de lecture au graphe conceptuel. Cet ordre est issu du sens que l'on donne à la relation dans la proposition, permet de définir des entrées et des sorties pour chaque relation.
Les arcs sont construits à partir des types des relations : chaque type permettra de savoir combien de liens elle supporte en entrée et sortie. Dans la définition donnée par [SOWA84], une relation peut n'avoir aucunes, une ou plusieurs entrées, et aura toujours exactement une sortie. Nous pouvons alors définir les types de relations par le nombre d'arcs qu'elles supportent : monoadique (une sortie), biadique (une entrée, une sortie), triadique (deux entrées, une sortie)… n-adique (n - 1 entrées, une sortie).
Pour illustrer cette utilisation des arcs, nous prenons la relation précitée **AGENT** qui nous permet de définir qu'un concept (la sortie, le sujet) est l'agent d'un autre concept (l'entrée, l'action). Pour cela nous utilisons comme concepts « Roméo mange » : sachant que **Roméo** est l'agent de l'action **MANGER** nous obtenons alors le graphe suivant :

Par ailleurs, lorsqu'une relation dispose de plusieurs entrées, il peut-être utile de numéroter ces entrées : cela donne un ordre qui permet le comparaisons et fournit en outre un sens pour mieux faire la correspondance en langage naturel.

Par exemple, en supposant que la relation **AGENT** soit disponible comme relation triadique, nous pouvons aussi traduire la phrase « Roméo mange et boit », sachant que les actions **MANGER** et **BOIRE** ont toutes deux **Roméo** pour agent, nous pouvons conserver l'ordre d'énumération par ajout de numéros aux arcs concernés :



Cependant, cette propriété des relations d'être connectées à plusieurs concepts comme entrées est très peu utilisée dans les faits : elle augmente la complexité des traitements et fait intervenir des structures dont on ne connaît pas a priori le nombre d'arcs en entrée, ceci pose donc un problème du point de vue de l'uniformisation des modèles computationnels de graphes conceptuels. Ainsi la plupart des systèmes n'utilisent pas cette fonctionnalité et au lieu de cela préfèrent dupliquer une relation si elle dispose de plusieurs entrées. Nous obtiendrons alors une structure beaucoup plus facile à manipuler et à formaliser : les graphes pourront être représentés comme un ensemble de triplets, chaque triplet représentant l'entrée, la relation et la sortie liés. Nous détaillerons ce type de structure plus loin, mais nous conservons cette fonctionnalité pour le moment, afin de conserver l'expressivité du formalisme tel qu'initialement imaginé par [SOWA84].

Les graphes :

Propriétés d'un graphe :

Nous pouvons maintenant utiliser cet ensemble d'éléments : les concepts, les relations et les arcs qui les lient entre eux, afin de former l'unité d'information élémentaire du formalisme : le graphe conceptuel. En effet, comme défini par [SOWA84] théoriquement, chaque graphe doit représenter une unité d'information, la plus élémentaire possible. C'est ici que se retrouvent les composantes terminologiques et assertionnelles du formalisme, comme souhaité par [MCGREGOR91], permettant de représenter une connaissance particulière sous une forme à la fois manipulable (comme assertion) et interprétable (comme ensemble de termes).

Un graphe conceptuel alors un graphe bipartite, orienté, connecté et fini :

- *bipartite* car formé de concepts d'une part et de relations d'autre part,
- *orienté* car tous ses arcs ont un sens,
- *connecté* puisque aucune partie du graphe peut ne pas être connectée au reste du graphe ou le deux parties formeraient simplement deux graphes conceptuels,
- *fini* car il n'est pas possible de représenter un graphe comportant un nombre infini de concepts, de relations ou d'arcs.

Le graphe est donc constitué d'un ensemble de concepts, d'un ensemble de relations et d'un ensemble d'arcs, qui le caractérisent. Différentes structures ont émergé de cette définition du graphe, prenant en compte des aspects de complexité dans la représentation d'un graphe. Les graphes sont alors souvent représentés comme ensemble de triplets : comme précité, chaque triplet représente une relation et la paire de concepts qu'elle lie ; cette représentation sera détaillée ultérieurement. Par ailleurs, les contextes et les définitions vont respectivement remettre en cause les propriétés de connexité et de finitude de graphes.

Le graphe comme proposition logique :

Le graphe conceptuel a une valeur de *proposition* que l'on peut traduire sous forme logique, afin de le manipuler par la suite. Pour cela, [SOWA84] définit la fonction φ qui fait correspondre à chaque graphe conceptuel une formule logique exprimée dans la logique des prédicats.

Pour cela des règles sont précisées. Nous considérons que chaque concept est typé et dispose d'un marqueur comme décrit précédemment, les relations sont typées mais ne disposent pas de marqueurs, les arcs sont orientés et numérotés, les graphes sont bipartites, orientés, finis et connectés. Alors nous pouvons associer :

- à chaque concept de type $\mathbf{t \in T}$ qui dispose d'un marqueur explicite ou implicite $\mathbf{m \in M}$, le préfixe $\exists$ **m** et le prédicat **t(m)**,
- à chaque relation de type $\mathbf{t \in T}$ n-adique liée au concept marqué $\mathbf{m_s \in M}$ en sortie et à l'ensemble ordonné de concepts marqués $\mathbf{(m_1, m_2, \ldots m_{n-1}) \in M^{n-1}}$ en entrées, le prédicat suivant : $\mathbf{t(m_s, m_1, m_2, \ldots m_{n-1})}$.

Il faut finalement réunir tous les préfixes en début de formule, et former une conjonction entre tous les prédicats monoadiques issus des types de concepts et n-adiques issus des relations. Nous obtenons ainsi une formule de la logique des prédicats, une clause.
Par exemple pour notre graphe présenté précédemment illustrant la phrase « Roméo mange et boit », en supposant que les concepts **MANGER** et **BOIRE** ont été marqués implicitement **#1** et **#2** par le système, nous obtiendrions l'équivalent en logique des prédicats par la clause suivante :

$\exists$ **Roméo** $\exists$ **#1** $\exists$ **#2 (CHAT(Roméo)** $\wedge$ **MANGER(#1)** $\wedge$ **BOIRE(#2)** $\wedge$ **AGENT(#1, Roméo)** $\wedge$ **AGENT(#2, Roméo))**

Ce formalisme est donc manipulable sous forme de clauses de la logique des prédicats, afin d'effectuer des mécanismes d'inférences à l'aide des règles classiques de la logique. Il est donc manipulable par un formalisme sous-jacent. De plus, la fonction φ est théoriquement un isomorphisme : à chaque graphe correspond une unique formule logique, à chaque formule logique correspond un unique graphe. De nombreux auteurs, notamment [MCGREGOR91], insistent sur l'importance d'une correspondance directe avec la logique pour les langages de programmation modernes.

Cependant nous remarquons que le formalisme n'inclut pas que la quantification existentielle ($\exists$) et la conjonction ($\wedge$). Si ceci paraît bien s'adapter à des systèmes logiques de type PROLOG, par l'assertion d'un ensemble de conjonctions (i.e. une disjonction de conjonctions, des clauses de Horn), en revanche, la quantification universelle ($\forall$), la disjonction ($\vee$) et la négation ($\neg$) ne sont pas explicitement formulées. Pour pallier à cela, [SOWA84] introduira : les définitions de types pour la quantification universelle, une relation monoadique **NEG** pour la négation, les ensembles disjoints pour la disjonction, mais ces notations restent assez controversées. Par ailleurs ces connecteurs logiques n'apparaissent pas essentiels à la manipulation des graphes et n'ont donc pas donné lieu à un véritable débat et une uniformisation dans la littérature.

La subsomption et les hiérarchies de types :

La subsomption :

Les types vont permettre au système de comparer les concepts entre eux ou les relations entre elles, par suite, il sera possible de construire des hypothèses en se basant sur ces types et d'obtenir ainsi un raisonnement sur les ensembles. Ce raisonnement basé sur les types a été considéré comme une partie essentielle du raisonnement humain, par les analogies qu'il permet.
Pour cela, les graphes conceptuels utilisent une relation de *subsomption* entre types. Cette relation d'ordre partiel permet de comparer les types entre eux selon les généralisations et spécialisations que l'on sait être sémantiquement vraies entre des types d'individus. Comme préconisé par [MCGREGOR91], les systèmes de représentation des connaissances modernes reposent souvent sur cette relation afin d'établir une taxonomie et d'en inférer une classification des termes. De plus, la subsomption est efficace et assez simple à mettre en œuvre dans un système.
Par exemple, sachant que le type **CHAT** est une *spécialisation* du type **ANIMAL** (ou inversement que le type **ANIMAL** est une *généralisation* du type **CHAT**), alors nous pouvons comparer ces deux types en disant que le type **ANIMAL** « subsompt » le type **CHAT**.

La subsomption peut se traduire sous forme logique comme implication entre types. Ainsi si le type $\mathbf{t_1}$ subsompt le type $\mathbf{t_2}$, et qu'un individu appartient au type $\mathbf{t_2}$, alors nous pouvons en inférer logiquement que cet individu appartient également au type $\mathbf{t_1}$, soit : $\mathbf{subompt(t_1, t_2)} \Leftrightarrow \forall \mathbf{x,}\ \mathbf{t_1}(x) \Rightarrow \mathbf{t_2}(x)$. Les relations de subsomption que l'on définit entre les types s'applique extensionnellement aux instances de chaque type, les concepts marqués implicitement ou explicitement. La subsomption représente une condition suffisante décrite par [MCGREGOR91] dans la classification à partir de règles.
On l'interprète également d'un point de vue mathématique comme un ordre partiel sur les ensembles de types. Ainsi nous noterons que le type $\mathbf{t_1}$ subsompt le type $\mathbf{t_2}$ de la manière suivante : $\mathbf{t_2 \leq t_1}$. Par exemple, sachant que le type **CHAT** est un sous-ensemble du type **ANIMAL**, alors nous pouvons comparer ces deux types en spécifiant la relation d'ordre partiel **CHAT ≤ ANIMAL**. En fait l'ordre partiel $\mathbf{t_2 \leq t_1}$ correspond également à la notation ensembliste (extensionnelle) $\mathbf{\delta t_2 \subseteq \delta t_1}$. Nous disons dans ce cas que $\mathbf{t_2}$ est un *sous-type* de $\mathbf{t_1}$, et inversement que $\mathbf{t_1}$ est un *sur-type* de $\mathbf{t_2}$.

Cette relation d'ordre partiel a les propriétés mathématiques suivantes :

- *réflexivité* : $\mathbf{t \leq t}$,
- *transitivité* : si $\mathbf{t_1 \leq t_2}$ et $\mathbf{t_2 \leq t_3}$ alors $\mathbf{t_1 \leq t_3}$,
- *antisymétrie* : si $\mathbf{t_1 \leq t_2}$ et $\mathbf{t_2 \leq t_1}$ alors $\mathbf{t_1 \equiv t_2}$.

Remarquons que nous n'utilisons pas le symbole d'égalité pour l'antisymétrie : deux types qui disposent de labels différents ne peuvent être égaux que s'ils sont des synonymes. Ceci représente un problème du point de vue des redondances, il faut donc éviter de créer des types qui existent déjà. En fait nous utiliserons cette propriété comme un test lors de l'inférence sur les types : nous pourrons ainsi déterminer qu'un type recherché existe déjà, mais cette propriété ne devrait jamais être instanciée par des labels. Par exemple, si l'on sait que **Roméo** est d'un type **t**, tel que **t ≤ CHAT** et **CHAT ≤ t**, il ne faut pas créer un nouveau type, mais affecter à **Roméo** le type **CHAT**.

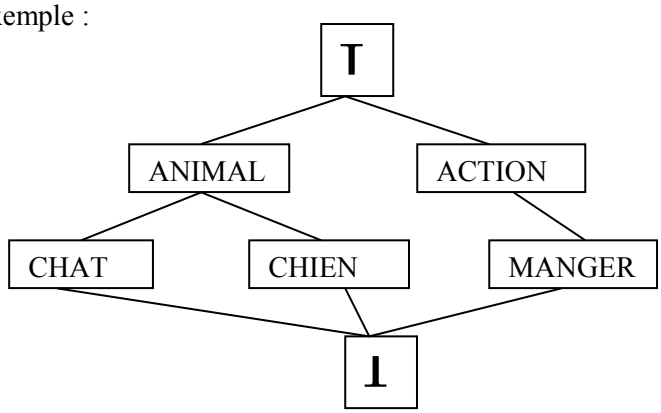
La subsomption permet également d'obtenir une relation d'ordre total sur les types. C'est une relation stricte: $\mathbf{t_1 > t_2}$, qui exclut les propriétés de réflexivité et d'antisymétrie. Celle-ci nous donne de la même manière que précédemment les *sur-types propres* et les *sous-types propres*. Mais comme précisé précédemment, l'instanciation de l'antisymétrie étant exclue, les deux ordres (partiel et total) devraient être équivalents pour les labels de types. Enfin nous pouvons également utiliser le *sur-type minimal* $\mathbf{t_2}$ d'un type $\mathbf{t_1}$ (resp. *sous-type maximal*) comme la borne inférieure des sur-types propres, noté $\mathbf{t_2 \succ\!\!- t_1}$ (resp. borne supérieure des sous-types propres, noté $\mathbf{t_2 -\!\!\prec t_1}$).

Pour utiliser correctement les types, nous allons être très souvent amenés à confronter un marqueur avec le type qui lui est assigné. Pour cela, [SOWA84] définit la relation de conformité, notée **::** qui permet de savoir si un marqueur donné est bien conforme à un type donné. C'est donc une fonction de **T*M** dans **{Vrai, Faux}** à l'aide des propriétés de transitivité et de réflexivité de la relation d'ordre partiel, afin de savoir si le type est conforme au marqueur donné.
Nous voyons ici qu'un marqueur doit effectivement contenir une information de type comme précité : un marqueur **m** est conforme à un type **t** si le type du marqueur **m** est un sous-type du type **t**. Bien que ceci n'ait pas été explicitement indiqué dans la littérature, nous pourrions alors considérer qu'un marqueur est en fait constitué d'un type et d'un label ou d'un numéro. Par ailleurs si un marqueur représente un individu, nous devrions être capable de savoir le type de l'individu à partir de ce marqueur : cela correspond bien à la notion de différenciation d'un individu parmi le groupe des individus du même type.

Hiérarchies des types :

La relation de subsomption, par ses propriétés précitées, permet de classer les types dans une taxonomie. Nous pouvons la caractériser comme une hiérarchie, car l'ordre partiel créé interdit la formation de cycles. Il et donc possible de comparer tous les types entre eux et d'obtenir une réponse en un temps fini.

Par ailleurs, cette hiérarchie se rapproche beaucoup du réseau sémantique dans sa structure. Elle ne contient que les labels de types (qui contiennent la sémantique du système) et se présente sous la forme d'un graphe, par exemple :



Cette hiérarchie sommaire permet de représenter les relations de subsomption et la transitivité efficacement, en un seul diagramme. Nous y voyons aussi apparaître par connectivité du réseau formé la notion de *sur-type commun* et de *sous-type commun* :

- si $\mathbf{t_1 \geq t_2}$ et $\mathbf{t_1 \geq t_3}$ alors $\mathbf{t_1}$ est un sur-type commun de $\mathbf{t_2}$ et $\mathbf{t_3}$,
- si $\mathbf{t_1 \leq t_2}$ et $\mathbf{t_1 \leq t_3}$ alors $\mathbf{t_1}$ est un sous-type commun de $\mathbf{t_2}$ et $\mathbf{t_3}$.

Enfin, comme pour les sur-types et les sous-types, nous pouvons également définir le *sur-type minimal commun* (resp. *sous-type maximal commun*), qui est la borne inférieure des sur-types communs (resp. la borne supérieure des sous-types communs) pour deux types.

Comme représenté dans l'exemple précédent, nous complétons cette hiérarchie d'après [SOWA84] par les types suivants :

- $\top$: le type universel, tel que $\forall \mathbf{t} \in \mathbf{T}, \mathbf{t} \leq \top$,
- $\bot$: le type abstrait, tel que $\forall \mathbf{t} \in \mathbf{T}, \mathbf{t} \geq \bot$.

Ainsi, pour n'importe quelle paire de types, il est toujours possible de trouver un sur-type commun et un sous-type commun. Nous obtenons alors un treillis, un modèle fonctionnel pour l'utilisation des types. Les hiérarchies et treillis, leurs définitions et leurs propriétés sont par ailleurs très bien détaillées par [ELLIS-CALLAGHAN97].

Par ailleurs, il est clairement précisé par [SOWA84] et la littérature sur les graphes conceptuels que les concepts et les relations ne sont pas comparables entre eux : il ne peut exister de mesure qui permette de rapprocher un concept à une relation. Nous obtenons alors deux hiérarchies, respectivement pour les concepts et pour les relations. En général, celle des relations est beaucoup plus importante que celle des concepts afin de favoriser les comparaisons sur ces derniers.
Remarquons également que la subsomption est une relation, mais celle-ci s'exprimant sur les types, elle permet alors de spécifier une règle, ce qu'il n'est pas possible de faire à travers un graphe qui ne comporte pas de marqueurs universels.

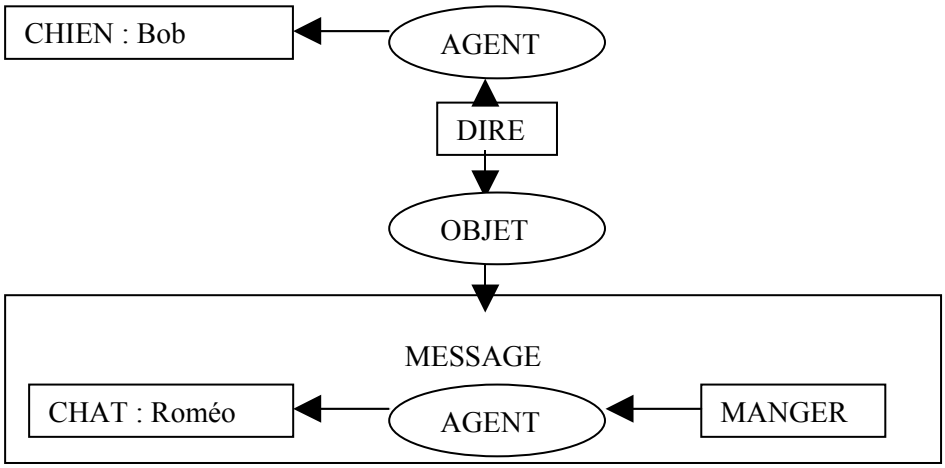
La notion de hiérarchie de types est centrale dans le formalisme des graphes conceptuels. En effet, cela nous permet de connaître le point commun entre deux concepts, par généralisation ou par spécialisation. La plupart des méthodes utilisent ce mécanisme de comparaison entre types afin de comparer les graphes entre eux, et trouver une analogie entre des graphes. Les systèmes reposant sur les graphes conceptuels sont donc très fortement dépendant des hiérarchies constituées pour les concepts et pour les relations : cela déterminera la complexité des recherches, l'expressivité du système, la véracité des hypothèses formées. De plus nous retrouverons plus loin ces hiérarchies pour structurer une base de connaissance exprimée sous forme de graphes conceptuels.

Extension du formalisme, contextes et graphes de définition :

Les contextes :

Les *contextes* sont présentés par [SOWA84] afin de décrire un concept par un graphe : ceux-ci permettent de faire d'une proposition (le graphe) un objet mental (le concept). Cela nous donne un moyen de raisonner avec un plus haut niveau d'abstraction : grâce aux contextes, nous pouvons former des relations entre des graphes. Du point de vue de l'interprétation, ces contextes correspondent à des propositions subordonnées, relatives, infinitives… et du point de vue de la logique, nous pourrions les qualifier comme prédicats de « deuxième ordre », un raisonnement sur les prédicats du premier ordre. Nous pouvons également le rapprocher de la logique floue, qui effectue des assertions de vérité sur d'autres assertions.

Pour la représentation, il suffit de placer un graphe à l'intérieur d'un carré. Ce graphe est typé : on lui assigne un type, mais [SOWA84] précise qu'il ne dispose pas de référent : son référent est en fait l'intérieur du graphe.
Pour illustrer ceci, prenons l'exemple suivant : « Bob dit que Roméo mange », sachant que Bob est du type **CHIEN**. La subordonnée forme le contexte, nous lui donnons alors le type **MESSAGE** et le graphe serait alors :



Ces contextes sont très utilisés, mais de manière éclectique, par les concepteurs de systèmes : quelques exemples sont donnés par [LEISHMAN92] qui les utilise dans les comparaisons entre propositions pour l'abstraction de schémas. Ainsi le schéma global d'une action peut-être représenté par un graphe, mais sans tenir compte des contextes sous-jacents.
L'article de [MINEAU-GERBE] propose de créer des mondes d'assertions par l'utilisation de ces contextes, ce qui permet de regrouper les assertions dans un treillis. Dans ce cas les contextes sont référencés, ce qui en permet une utilisation plus souple. De plus les contextes peuvent contenir plusieurs graphes, qui seront traités selon les représentations d'ensemble de [SOWA84]. Il devient alors possible de représenter des données à l'image du rapport ci-présent : des théories ont été effectuées par des auteurs, qui sont ici exposées, analysées et occasionnellement critiquées. Nous effectuons alors des assertions sur les assertions. Cette approche très novatrice situe les propos selon leur auteur et permet d'effectuer de l'analyse de textes par graphes conceptuels.
Les contextes permettent donc de représenter un plus haut niveau d'abstraction, ils peuvent être également conçus comme des ensembles de graphes, comme [MINEAU-GERBE] le fait. Cependant les problèmes concernant les ensembles ne sont pas résolus pour autant vis-à-vis des éléments qu'ils contiennent.
Il paraît donc possible d'utiliser les contextes pour beaucoup d'applications très diverses, ce qui pose alors le problème de leur uniformisation et de leur intégration dans le formalisme des graphes conceptuels. Mais leur utilité et loin d'être remise en cause, elle parait même de plus en plus pressentie par les concepteurs de systèmes à base de graphes conceptuels.

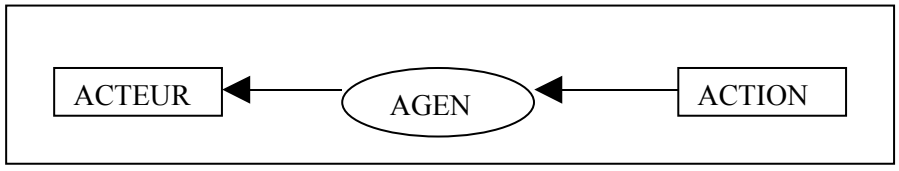
Les graphes de définition :

À défaut d'avoir la possibilité d'exprimer des quantificateurs universels à travers les graphes, il est possible de former des *définitions* sous forme de graphes. La définition est alors supposée caractériser tous les individus d'un type particulier. Nous pouvons la considérer comme une généralisation commune de tous les individus possibles d'un même type.
La définition s'exprime donc sous forme d'un graphe, qui comporte des informations génériques sur le concept que nous voulons définir. D'après [SOWA84], ces définitions sont utiles pour définir des relations : chaque relation peut-être définie comme un graphe conceptuel comportant uniquement des relations de type **LINK**. Inversement, on trouve assez souvent dans la littérature, notamment dans [HARTLEY-COOMBS91], l'utilisation de définition de concepts pour le raisonnement sur les graphes.

Il parait possible de définir un type de concept ou de relation par un autre graphe. Ces définitions sont supposées mettre en jeu des concepts et des relations plus élémentaires, d'un plus bas niveau d'abstraction. Cette règle de définition correspond bien à l'image que l'on en a à travers un dictionnaire. Les définitions sont également supposées être isomorphes : il est toujours possible de remplacer un concept ou une relation par sa définition, et inversement, lorsque l'on reconnaît une définition il est toujours possible de la remplacer par le concept ou la relation qu'elle définit.
On peut également faire un lien ici avec les contextes : la définition représente un ensemble d'individus, et nous avons vu que les contextes pourraient être utilisés afin de représenter des ensembles. Dans un contexte, on représente une instance d'un type, alors que la définition représente toutes les instances d'un type. Il est alors théoriquement possible d'extraire une définition par analogies entre plusieurs contextes, ou inversement de dériver un contexte par instanciation d'une définition.

Un autre type de définition, ici dite *fonctionnelle*, est très utilisée pour décrire une grammaire. Ces règles ne définissent alors pas forcément un type de concept ou de relation, mais leur organisation dans une structure possible. Elles permettent de spécifier des contraintes sur les types, en donnant la place de chaque type dans une structure générique : elles spécifient comment composer des types entre eux lors de la construction de graphes.
Ces définitions sont souvent considérées comme un ensemble de graphes particuliers des systèmes : la base canonique que nous décrirons plus loin. Elles permettent alors de donner une cohérence aux règles de formation des graphes. Elles sont par ailleurs très souvent centrées sur l'utilisation des types de relations : à chacune sont associés un ensemble de types de concepts à travers les arcs. Nombre de systèmes mettent en œuvre ce type de définitions : [MINEAU1], [ELLIS95] et plus particulièrement l'analyseur syntaxique de [NICOLAS] qui exprime ces règles pour former une grammaire du langage naturel.

Nous pouvons illustrer ces deux types de définitions par le type **ACTE**, qui spécifie qu'un concept **ACTEUR** est lié par la relation **AGENT** à un concept **ACTION**. Nous l'exprimons par le graphe suivant :



Remarquons qu'il pourrait être envisageable de représenter une relation de subsomption comme une définition : la subsomption s'applique à tous les individus d'un type et par là correspond bien à l'idée de quantification universelle que donne la définition. Ainsi nous pourrions définir un type $\mathbf{t_1}$ par un autre type $\mathbf{t_2}$, ce qui exprimerait le fait que tous les individus du type $\mathbf{t_1}$ sont également du type $\mathbf{t_2}$.

La définition est donc un outil très utile, qui permet de donner plusieurs degrés d'abstraction à un système, mais reste difficile à manipuler car elle demande une maintenance des valeurs de vérité des définitions à travers les graphes. Elle suppose également des règles très précises pour son expansion et sa contraction, qui n'ont pas été véritablement formalisées.
Nous y voyons un lien très fort avec les contextes et les ensembles. Finalement un contexte peut se concevoir comme une définition dans un cadre localisé, et inversement une définition comme un contexte dans un cadre général.

LES OPÉRATIONS SUR LES GRAPHES CONCEPTUELS :

Définition des opérateurs sur les graphes :

Les règles de formation canoniques :

Nous disposons maintenant d'un formalisme très précis pour la représentation de connaissances sous forme de graphes bipartites, orientés, connectés et finis. Ils ont en plus une correspondance directe en logique des prédicats. Ces propriétés vont alors nous permettre de définir des manipulations, afin de modifier les graphes ou de les composer entre eux. Ces manipulations sont dénommées *opérateurs* sur les graphes conceptuels.

Dans la théorie classique des graphes conceptuels, [SOWA84] présente quatre opérateurs, appelés *règles de formation canonique* :

- *la copie* effectue une copie d'un graphe : elle permet de dupliquer un graphe afin de l'utiliser sans modifier l'original,
- *la simplification* efface une composante d'un graphe, si une relation est dupliquée, alors on peut supprimer l'un des duplicata et tous ses arcs,
- *la restriction* permet de remplacer le type d'un concept du graphe par un de ses sous-types, tant que le marqueur du concept reste conforme à ce sous-type,
- *la jointure* réunit deux graphes à partir d'un concept dans chaque graphe, si ces deux concepts ont les mêmes marqueurs et le même type, alors on copie l'un d'eux dans un nouveau graphe auquel on lie les relations liées aux deux concepts des graphes originaux.

Ces quatre opérateurs sont logiquement consistants dans le formalisme de la logique du premier ordre. Ils sont directement démontrables dans cette logique : dupliquer une clause (copie), supprimer une partie de clause conjonctive (simplification), considérer un concept comme instance d'un sous-type tant que le marqueur y reste conforme (restriction), joindre deux clauses vraies en formant la conjonction (jointure) de ces clauses, sont des règles que la logique des prédicats peut valider formellement.
Nous ajoutons cependant un point dont nous n'avons pas parlé plus haut : d'après [SOWA84], la restriction permet également de remplacer un marqueur générique (ou implicite) par un marqueur explicite (individu) : nous appelons cette règle l'affectation, tout en conservant à l'esprit que c'est normalement une particularité de la simplification :

- *l'affectation* permet de remplacer un marqueur implicite par un marqueur explicite si ce dernier est conforme au type du marqueur implicite.

C'est ici qu'intervient un compromis logique entre la quantification existentielle et la quantification universelle. En effet, si un individu indéterminé d'un type connu assume un rôle dans un graphe, lui affecter un marqueur d'un type conforme revient à dire que n'importe quel autre individu d'un type conforme pourrait également convenir. Cette règle est alors manifestement indémontrable du point de vue de la logique : elle donne de ce fait toute la souplesse et l'originalité du formalisme des graphes conceptuels, qui fonctionne dans un monde sémantiquement ouvert comme l'indique [SOWA84].
Le terme *restriction* correspondra donc à la règle décrite par la littérature. Avec la copie, cette règle donnera aux marqueurs implicites une portée universelle : n'importe quelle assertion générique peut-être copiée et simplifiée autant de fois qu'il y a de marqueurs explicites conformes aux types des concepts génériques.

Ces règles de formation canonique permettent de construire de nouveaux graphes à partir des graphes existant dans le système. Comme nous avons vu, ces règles ne préservent pourtant pas la vérité. Elles permettent uniquement d'imposer des contraintes de sélection aux graphes formés. Ce ne sont alors pas des règles d'inférence. D'après [SOWA84] elles fonctionnent sur le principe de la réfutation, permettant de déterminer qu'un graphe est faux s'il est une dérivation canonique d'un autre graphe qui a été démontré faux.
Il n'y a donc pas de règles qui préservent explicitement la vérité des graphes d'un point de vue logique, au vu de la littérature. Cela n'apparait pas réellement utile pour le raisonnement à base de

graphes conceptuels. Cependant bien peu de systèmes considèrent de stocker des graphes faux afin de pouvoir par la suite vérifier quelles dérivations canoniques sont fausses ! Nous voyons donc ici un accord tacite afin de considérer que les règles de formation canoniques, empêchent par contraintes sélectionnelles de former des non-sens, mais permettent malgré tout d'affirmer par dérivation des graphes que l'on ne sait pas forcément être vrais.
Par ailleurs, ces règles sont des opérateurs de spécialisation. Cela apparaît très clairement pour la restriction et la jointure qui spécifient plus précisément des graphes. D'après [SOWA84], les opérateurs de copie et de simplification ne spécialisent pas vraiment, mais ils ne généralisent pas plus les graphes.

Autres opérateurs sur les graphes :

Les opérateurs précités sont inversibles. Leur application est assez peu courante, ils ne sont donc pas précisés par la littérature étudiée. Néanmoins, nous pouvons les nommer *délétion*, qui supprime la copie d'un graphe existant (copie), *duplicata*, qui double une relation et ses arcs (simplification), *élargissement*, qui permet de donner un sur-type à un concept (restriction), *division*, qui permet de séparer une assertion en deux assertions en dupliquant un concept (jointure). Mais ces opérateurs restent informels, car difficilement utilisables dans le cadre des graphes conceptuels. Il n'y a donc pas vraiment d'opérateurs explicitement conçus pour la généralisation, qui n'est pas l'objectif premier du formalisme des graphes conceptuels.

Il existe de nombreux autres opérateurs sur les graphes conceptuels, qui ne font pas partie des règles de formation canoniques. Ils sont souvent des composées des règles de formation canoniques présentées ci-dessus. Nous en donnons ici quelques uns :

- *la projection* présentée par [SOWA84] recherche dans un graphe une spécialisation d'un autre graphe ; on va parcourir le premier graphe à la recherche d'une dérivation par restriction du second graphe. Cet opérateur permet de retrouver des définitions au sein d'un graphe et sera donc particulièrement utile dans l'application des définitions, ou pour la recherche d'un schéma dans une base de graphes. Malheureusement cet opérateur reste complexe car il faut trouver une analogie entre deux graphes,
- *l'expansion*, proposée par [SOWA84] permet de remplacer un type par un graphe qui représente sa définition ; s'il dispose d'un marqueur, celui-ci est affecté à un concept précisé par la définition,
- *la contraction*, proposée par [SOWA84], l'inverse de l'opérateur précédent, permet de remplacer dans un graphe une partie qui correspond à une définition donnée par le concept qui représente la définition ; le marqueur précisé dans la définition est affecté au concept contracté,
- *la spécialisation*, proposé par [HARTLEY-COOMBS91], elle est composée de la *couverture*, qui permet de trouver des définitions pour des graphes factuels, suivie de la jointure, qui regroupera ses définitions afin de lier ces faits entre eux dans un seul graphe hypothèse,
- *la fragmentation*, également proposé par [HARTLEY-COOMBS91], permet de diviser une hypothèse en d'autres faits que ceux qui ont été spécifiés avant la spécialisation ci-dessus : c'est la composée d'une projection, qui permet de reconnaître le définitions connues, et de la *découverture*, qui divise effectivement l'hypothèse en supprimant des liens,
- *la fusion*, proposée par [ELLIS95] représente la jointure que nous avons décrite plus haut, alors que ce qui est nommé *jointure* par cet auteur est une jointure interne à un seul graphe conceptuel, elle s'intègre dans son formalisme aux règles de formation canonique,

Ces opérateurs sont donc utiles pour des applications spécifiques, et sont souvent conçus pour augmenter les capacités de déduction et d'inférence du formalisme. Contrairement aux opérateurs canoniques, elles ne fonctionnent pas sur la réfutation, et permettent de spécialiser ou de généraliser des graphes selon les besoins.

Le canon et les dérivations canoniques :

Définition du canon :

Les règles de formation canoniques vont donc permettre d'effectuer des dérivations à partir d'un ensemble de graphes ou inversement, de savoir si un graphe est une dérivation d'autres graphes. Les graphes qui ne peuvent être dérivés à partir d'autres graphes forment ce que [SOWA84] appelle la *base canonique*. Comme ces graphes n'ont pu être obtenus par dérivation, leur existence est admise par le système, ces graphes ont pu être obtenus par inférence, avec de opérateurs non canoniques, ou par acquisition de connaissance. Cette base va nous permettre de définir les possibilités du système en terme de dérivations canoniques.
Nous pouvons maintenant introduire la définition selon [SOWA84] du *canon* :

- une hiérarchie de types,
- un ensemble de marqueurs individuels,
- une relation de conformité **::** qui lie les marqueurs aux types,
- un ensemble fini de graphes, la base canonique.

À partir de ce canon, nous pouvons désormais dériver un ensemble de graphes selon les règles de formation canoniques. Le nombre de dérivations possibles est directement dépendant du nombre de types, du nombre de marqueurs individuels et bien entendu du nombre de graphes dans la base canonique. Tous les graphes dérivés de cette manière sont dits *canoniques*.
Le canon comporte donc toutes les composantes nécessaires à un système fonctionnant sur le formalisme des graphes conceptuels. Mais il ne présente que les composantes théoriques : il n'y a pas de structure pour conserver les graphes représentant des faits. Le canon ne comporte pas non plus les règles de formation canonique, ni la relation de subsomption, alors qu'il comporte la relation de conformité et une hiérarchie de types.

Variantes du canon :

Le canon représente donc les informations essentielles pour définir un système basé sur le formalisme des graphes conceptuels. Cette configuration peut varier dans la littérature :

- [CHAMPESME96] nomme le canon *support*, la base est composée de *S-graphes étoiles*, qui ne possèdent qu'une relation, leur donnant une définition fonctionnelle,
- [MINEAU2] présente une hiérarchie formée des labels de types, l'ensemble des marqueurs peut contenir des variables (marqueurs implicites ou génériques) et la base canonique décrit de la même manière que précédemment les types des relations,
- [ELLIS95] précise la présence de graphes atomiques, qu'il n'inclut pas dans le canon, qui est complété par les coatomes : les atomes sont des graphes non instanciés et qui ne comportent pas de connecteurs logiques, les coatomes sont les graphes les plus spécialisés. Les atomes se rapprochent cependant très fortement de la base canonique, et les coatomes à une base de faits.

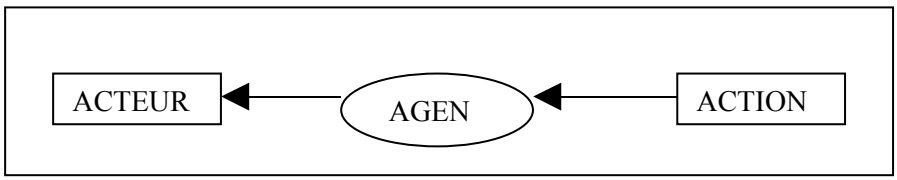
Ainsi la représentation du canon peut varier selon les considérations et les besoins des concepteurs de systèmes. Mais le canon reste théoriquement l'ensemble des éléments de base, qui permettent de définir les données d'un système reposant sur le formalisme des graphes conceptuels.
Remarquons que nous pouvons aisément rapprocher le terme de base canonique à son sens mathématique : dans une matrice, la base canonique est un ensemble limité de vecteurs permettant de décrire n'importe quelle ligne (ou colonne) de la matrice par combinaison linéaire des vecteurs de la base. Mais ils ne peuvent se décrire entre eux si la base est canonique. Le nombre de vecteurs dans la base donne la dimension de l'espace vectoriel créé.
Parallèlement, les opérateurs canoniques sont en quelque sorte les combinaisons linéaires pour les graphes, et chaque graphe de la base canonique donne une dimension supplémentaire au système en lui permettant d'utiliser un graphe de la base pour dériver de nouveaux graphes. En fait nous pourrions qualifier les graphes de la base canoniques de *graphes propres*, à l'image des vecteurs propres dans une matrice.

Diverses considérations pour la base canonique :

La définition et le canon :

L'ensembles des graphes d'un système est une base de connaissances, que l'on considère souvent pour un système expert en deux parties : la base de règles et la base de faits. Comme nous avons vu, la création de faits par spécialisation, grâce aux opérateurs précités, ne pose pas de difficultés, mais un certain nombre de graphes ne peuvent pas être dérivés de cette manière : la base canonique.
Nous allons donc considérer que la base canonique représente pour un système expert la base de règles, qui sont assumées par le système et ne peuvent être inférées. Inversement, les graphes canoniques obtenus par dérivation représentent pour un système expert la base de faits, qui peuvent être expliqués par le canon.

Au vu de la littérature, le graphes de la base canonique concernent en général le contraintes imposées sur les relations, afin qu'elles ne puissent pas lier n'importe quels concepts entre eux. Nous y voyons donc une définition fonctionnelle des relations, qui indique leur utilisation. Et comme nous avons précisé plus haut lors de la définition, les types de concepts impliqués dans la définition d'une relation sont également définis : le type **ACTEUR** est utilisé avec la relation **AGENT**, la définition fonctionnelle du type de concept est donc également contenue dans celle de la relation, de même que pour le concept **ACTION**.
Les graphes de la base canoniques permettent alors de définir fonctionnellement des concepts et des relations selon le rôle qu'ils sont supposés jouer dans les graphes. Nous remarquons qu'il est assez souvent possible d'ajouter à ces définitions fonctionnelles la capacité de définir un type, si l'on en trouve un qui soit approprié. Cependant, comme nous avons dit plus haut, la définition des types n'est pas encore formalisée et il faut prendre de grandes précautions lors de son utilisation. Nous rappelons à titre d'exemple la définition présentée plus haut du type **ACTE** :



Pour la suite, nous utiliserons la base canonique comme un ensemble de définitions fonctionnelles, essentiellement utiles à préciser quels types de concepts peuvent être utilisés avec quels types de relations. La définition de type pour expansion et contraction par utilisation de contextes ne sera pas considérée, mais nous remarquerons qu'il pourrait y avoir ici un rapprochement à effectuer entre la définition, le contexte et les graphes de la base canonique.

Application pratique du canon :

Les graphes de la base canonique peuvent donc être conçus comme des définitions de rôle des concepts et des relations dans les graphes que nous formons. Ils constituent alors une partie essentielle du système du point de l'efficacité et de la validité de la formation de graphes. Ces graphes doivent donc être fondamentalement vrais et cohérents avec les types, afin qu'ils ne donnent lieu qu'à des dérivations qui seront également cohérentes. Pour cela, de nombreux systèmes définissent les types de relations vis-à-vis des types de concepts qu'elles lient.
Ainsi par la base canonique, un lien est effectué entre la hiérarchie des types de concepts et celle des types de relations. Ces hiérarchies sont donc soumises à des contraintes réciproques lors de leur utilisation pour la formation des graphes canoniques. Cela fournit une valeur de vérité aux graphes qui sont directement dérivables de la base canonique. La plupart des graphes de la base canonique seront alors définis manuellement, afin d'éviter toute erreur. Les mécanismes d'inférence pourront également former des graphes, qui s'ils sont validés, peuvent être ajoutés à la base canonique, avec précautions.

LES STRUCTURES DE GRAPHES CONCEPTUELS :

Subsomption entre graphes conceptuels :

Relation de subsomption appliquée aux graphes

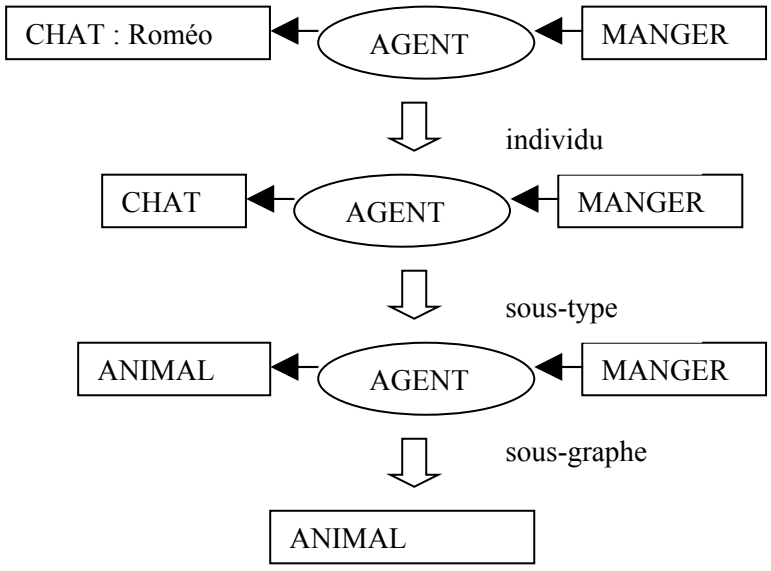
La relation de subsomption que nous avons définie pour les types peut également être mise en oeuvre sur les graphes conceptuels. Nous avons en effet vu que les règles de formation canoniques spécialisent les graphes, et qu'il est possible de créer d'autres opérateurs pour la spécialisation comme pour la généralisation des graphes. Alors nous pouvons déterminer quels paires de graphes entretiennent une relation de subsomption, s'ils sont dérivations l'un de l'autre.
Nous pouvons donc étendre cette même relation aux graphes conceptuels, afin de les comparer et de les classer dans une hiérarchie de graphes conceptuels. La formation de cette structure donne lieu à des gains du point de vue de la complexité des recherches, du stockage des informations, de la compréhension et de la cohérence d'une base de graphes. Il apparaît donc que l'on a beaucoup à gagner lorsque l'on utiliser la même relation sur les graphes que celle que l'on utilisait pour les types de concepts et relations.
La relation de subsomption entre graphes s'écrit de la même manière comme un ordre partiel $\mathbf{g_i \leq g_j}$ (ou un ordre total $\mathbf{g_i < g_j}$). Cette relation conserve les propriétés précitées :

- *réflexivité* : $\mathbf{g \leq g}$,
- *transitivité* : si $\mathbf{g_1 \leq g_2}$ et $\mathbf{g_2 \leq g_3}$ alors $\mathbf{g_1 \leq g_3}$,
- *antisymétrique* : si $\mathbf{g_1 \leq g_2}$ et $\mathbf{g_2 \leq g_1}$ alors $\mathbf{g_1 \equiv g_2}$.

Nous y ajoutons quelques autres propriétés données par [SOWA84] et qui semblent énoncer les propriétés de spécialisation des règles de formation canoniques :

- *sous-graphe* : si $\mathbf{g_1}$ est une partie de $\mathbf{g_2}$ alors $\mathbf{g_2 \leq g_1}$,
- *sous-type* : si $\mathbf{g_1}$ est identique à $\mathbf{g_2}$ excepté que un ou plusieurs types de concepts de $\mathbf{g_1}$ sont des sous-types des types des concepts correspondants dans $\mathbf{g_2}$ alors $\mathbf{g_1 \leq g_2}$,
- *individu* : si $\mathbf{g_1}$ est identique à $\mathbf{g_2}$ excepté que un ou plusieurs marqueurs implicites de concepts dans $\mathbf{g_2}$ sont remplacés par des marqueurs explicites (individuels) dans $\mathbf{g_1}$ alors $\mathbf{g_1 \leq g_2}$,

Il est également précisé qu'un graphe composé d'un concept de type universel $\top$ est une généralisation de tous les autres graphes, mais cette propriété semble découler directement des propriétés précitées et de la hiérarchie de type.
Nous donnons ci-après une illustration de cette relation d'ordre sur le graphe représentant « Le chat Roméo mange », en représentant des graphes de plus en plus généraux.



Nous voyons assez clairement le lien qui est effectué entre la relation d'ordre sur les graphes et les règles de formation canonique : chaque règle donne lieu à une spécialisation donc à un graphe plus spécialisé, plus bas dans la hiérarchie. L'application des opérateurs devrait donc nous permettre de construire une hiérarchie de la même manière que pour les types. Il est cependant beaucoup plus coûteux de développer une hiérarchie sur les graphes, car il faut pour cela démontrer qu'il existe une relation de subsomption entre deux graphes. Nous discuterons à ce sujet plusieurs algorithmes pour la construction d'une hiérarchie de graphes.
Par ailleurs, [SOWA84] montre que si un graphe $\mathbf{g_1}$ est une dérivation canonique de $\mathbf{g_2}$ et que l'on sait que $\mathbf{g_1}$ est un graphe vrai, alors $\mathbf{g_2}$ sera également vrai. En fait, ceci est déduit selon le principe de la réfutation : les règles de formation canoniques sont l'inverse de règles d'inférence. Ainsi si une dérivation de $\mathbf{g_2}$ (hypothèse) nous donne un graphe $\mathbf{g_1}$ (fait) que l'on sait être vrai, alors le graphe $\mathbf{g_2}$ à partir duquel on a dérivé est également vrai. Cependant ce mécanisme n'est pas très utilisé car il y a de trop nombreuses dérivations à essayer à partir d'un graphe général pour trouver un graphe connu, ce n'est pas une propriété véritablement applicable dans la pratique.

Quelques structures issues de la littérature :

Nous allons donc être amenés à raisonner par subsomption sur les graphes. Pour cela des analogies doivent être trouvées entre les graphes. Nous allons présenter ici quelques méthodes analogiques issues de la littérature permettant ce type de raisonnement.

Les travaux de [LEISHMAN92] montrent que les analogies peuvent se baser sur une correspondance de relations, les liens de coréférences entre marqueurs explicites, l'évaluation de contextes, l'abstraction des types de concepts. Ces critères doivent être évalués pour comparer deux analogues, ils donnent éventuellement lieu à une mesure.
Plus de précisions sont données par [SOWA84] et [ELLIS95], qui proposent des méthodes de comparaison très similaires, basées sur les règles de formation canonique afin de décrire la structure hiérarchique des graphes. Cette structure, très classique dans le domaine des graphes conceptuels, a l'avantage d'être formalisée et fonctionnelle. Cependant, elle n'est pas optimale car les graphes sont représentés comme un objet indivisible, alors qu'il est possible de le décrire comme ensemble de triplets. Nous détaillerons une structure de ce type ci-après.

Pour [LEVINSON00], ces comparaisons sont une forme de symétrie. Ce terme est très large et permet la comparaison analogique précitée couplée avec les hiérarchies de graphes. Nous voyons apparaître ici quelques méthodes, dont essentiellement deux sont retenues. La première repose sur la hiérarchie des types de concepts et la description des graphes comme ensemble de noeuds. La seconde méthode, qui est une version plus poussée de la précédente, repose sur une structure de données universelle (UDS, Universal Data Structure), qui enregistre les graphes comme ensemble de triplets, et chaque triplet comme un ensemble de trois nœuds.
Dans le même style, une structure de données par généralisation est présentée par [MINEAU92], celle-ci n'utilise pas des dérivations canoniques, mais décrit toutes le généralisations possibles d'un ensemble d'objets. De même, chaque graphe est représenté comme un ensemble de triplets, et chaque triplet comme un ensemble de trois nœuds. Nous y voyons l'introduction des éléments de généralisations et des listes d'instanciations comme structures de données. Une structure hiérarchique en est déduite, permettant d'optimisation la classification et les recherches dans les graphes. Cette structure sera également présentée plus longuement ci-après.
Une structure universelle de donnée sera également présentée par [MINEAU1]. Cependant le cadre y et bien plus général, l'accent est mis sur la complexité des classifications dans une structure issue d'un ordre partiel, la classification des connaissances est alors décrite selon des paramètre formels et les étapes de classification y sont segmentées en fonctions définie formellement. Ce paradigme se présente plutôt comme un guide pour les concepteurs de systèmes de classification afin d'isoler chaque paramètre qui complexifie un algorithme de classification.

Enfin un autre type de structure est présenté par [ELLIS-CALLAGHAN97] pour la création d'ordres, structure qui s'applique assez directement aux graphes. Les ordres et les treillis sont très précisément

définis par les auteurs, dont de nombreux aspects ont été présentés ci-dessus. Ces ordres sont conçus comme des ensembles d'éléments abstraits ordonnés. Ils donnent par la suite lieu à une possibilité de comparaison entre ordres. Malgré la grande complexité et la difficulté d'interprétation de ce type de structure, cette approche pourrait s'avérer très bénéfique, notamment pour les systèmes multiagents disposant de plusieurs bases de connaissances, plus particulièrement pour l'apprentissage multiagent.

Les algorithmes de classification des graphes :

La classification dans un système de graphes conceptuels est un problème complexe. Bien qu'ayant été formalisé par [SOWA84] sous forme de hiérarchie, d'autres types de structures ont vu le jour, qui prennent en considération des aspects plus pratiques de complexité en temps et en espace de la classification. Les méthodes précitées pour la formation de structures de graphes donnent une organisation de la base de graphes et augmentent l'efficacité de la classification. Cette classification repose sur la recherche de la place de chaque élément au sein de la structure. La plupart du temps, cette recherche est directement issue de la comparaison des graphes deux à deux. En conséquence de quoi les systèmes reposent sur un algorithme analogique, qui effectué de nombreuses fois permettra finalement de déterminer la place de chaque graphe.
Pour cela, nous allons présenter ici essentiellement deux type de structures, qui semblent avoir été efficacement utilisées lors d'implémentation de systèmes de graphes conceptuels : la structure compilée présentée par [ELLIS95] et la structure de généralisation présentée par [MINEAU1].

Compilation de graphes :

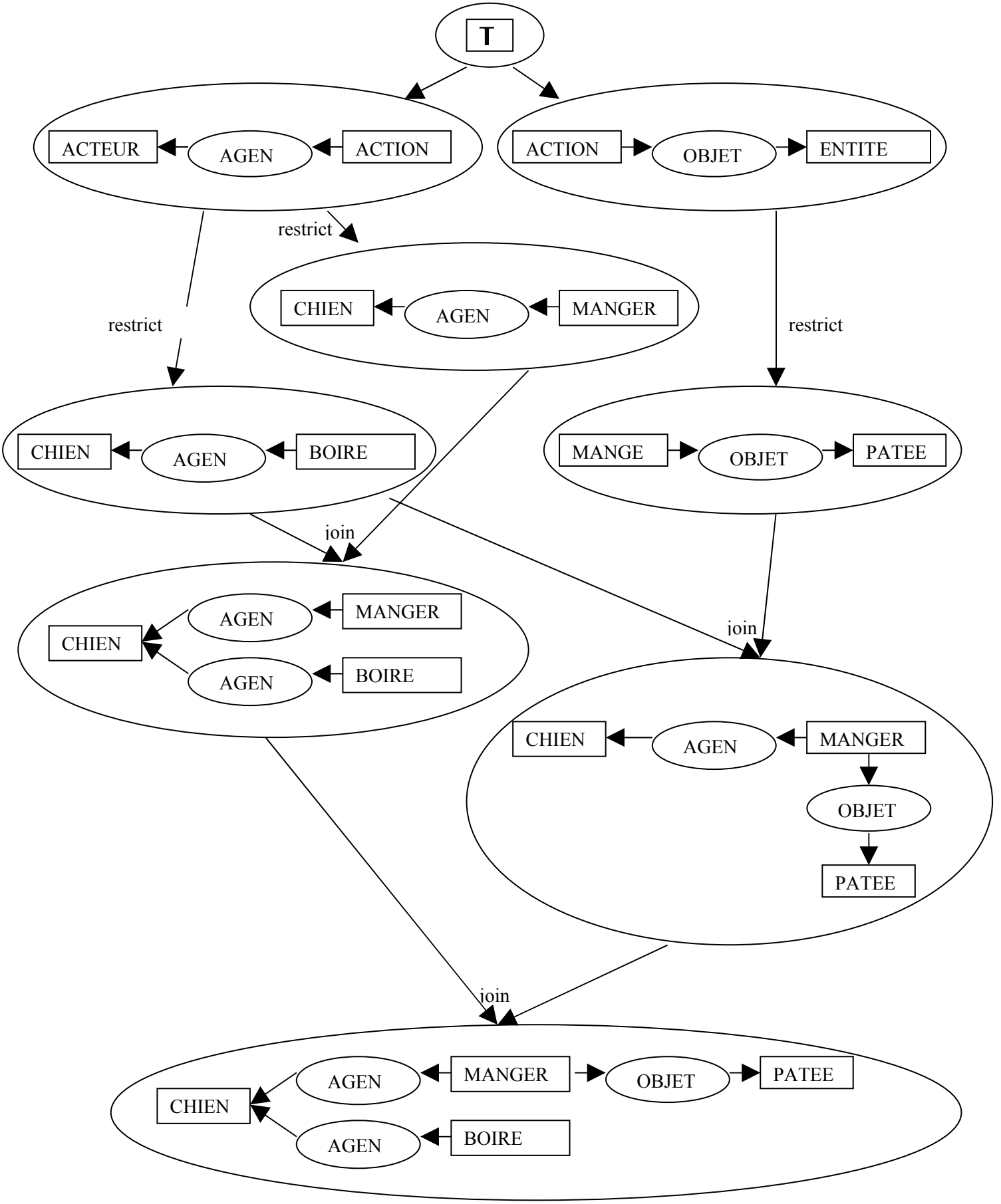
La première structure que nous étudions est fondée sur les hiérarchies de [SOWA84], et a été détaillée et améliorée par [ELLIS95] en une structure *compilée*. C'est donc une structure directement issue de la théorie des graphes conceptuels et des règles de formation canonique. En fait le but est ici de disposer d'un canon, et de décrire une base de connaissances par application des règles de formation canonique aux graphes de la base canonique, sans décrire les graphes eux-mêmes. Les graphes sont compilés car ils sont représentés par une suite d'instructions.

Le canon correspond à la structure définie plus haut. Les règles de formation canonique sont globalement les mêmes, sauf que la jointure est effectuée à l'intérieur d'un graphe et est renommée fusion si elle joint deux graphes. Les opérateurs canoniques sont définis de la manière suivante :

- *copy(**g**)* : copie de graphe **g**,
- *restrict(**g**, **c**, **m**)* : on remplace le marqueur de **c** par **m**,
- *simplify(**g**, $\mathbf{r_1}$, $\mathbf{r_2}$)* : on supprime la relation $\mathbf{r_2}$ si c'est un duplicata de $\mathbf{r_1}$ dans **g**,
- *join(**g**, $\mathbf{c_1}$, $\mathbf{c_2}$)* : on supprime le concept $\mathbf{c_2}$ dans **g** et on lie tous ses arcs à $\mathbf{c_1}$,
- *fuse($\mathbf{g_1}$, $\mathbf{g_2}$, $\mathbf{c_1}$, $\mathbf{c_2}$)* : on supprime le concept $\mathbf{c_2}$ dans $\mathbf{g_2}$ et on lie tous ses arcs à $\mathbf{c_1}$ dans $\mathbf{g_1}$.

Notons que des vérifications doivent être effectuées afin de s'assurer que chaque opération s'applique bien aux paramètres qui lui ont été fournis. Ces vérifications consistent essentiellement à comparer les marqueurs de deux concepts ou de deux relations qui sont supposés être identiques, modulo le type du concept qui doit toujours être conforme à son marqueur.

Alors les paramètres spécifiés pour chaque règle permettent de les considérer comme des fonctions, des procédures qui pour un ensemble de paramètre donnent toujours une même sortie : il est possible d'enregistrer un graphe de manière à pouvoir le reconstituer par la suite. Nous décrivons alors les graphes par une suite de dérivations canoniques. Nous stockons ces graphes dans une hiérarchie, dont les arcs sont des règles permettant d'arriver à chaque nœud de la hiérarchie.
Ci-après se trouve une illustration de ce type de hiérarchie basée sur la compilation des graphes conceptuels comme suite d'instructions. Nous n'y avons utilisé que les règles restrict et join, qui sont les deux règles les plus importantes lors de la construction et de la spécialisation de graphes conceptuels. Ce schéma présente plus particulièrement la structure hiérarchique, les liens issus des dérivations canoniques et les graphes intermédiaires obtenus, mais celle-ci peut, bien entendu, comporter les autres opérateurs précités.


T
ACTEUR
AGEN
ACTION
ACTION
OBJET
ENTITE
restrict
restrict
restrict
CHIEN
AGEN
MANGER
CHIEN
AGEN
BOIRE
MANGE
OBJET
PATEE
join
join
CHIEN
AGEN
MANGER
AGEN
BOIRE
CHIEN
AGEN
MANGER
OBJET
PATEE
join
CHIEN
AGEN
MANGER
OBJET
PATEE
AGEN
BOIRE

Il est possible d'utiliser un algorithme de parcours en profondeur pour rechercher un graphe à travers cette hiérarchie, en partant du graphe universel. Cette recherche fonctionne par spécialisation pour trouver une réponse à une question dans la base de connaissances que constitue la structure.
Nous pouvons également rechercher par généralisation. Dans ce cas on utilise une recherche en largeur, qui permet de chercher dans l'espace des généralisations possibles. Il faut cette fois-ci partir des graphes les plus spécialisés pour savoir si la recherche en est une généralisation.

Mais ces deux types de recherches n'étant pas véritablement efficaces, un troisième type d'algorithme a été développé, pour une recherche dite *topologique*. Celle-ci suppose qu'un graphe ne peut-être une généralisation d'une recherche que si toutes ses généralisations directes sont également des généralisations de la recherche. Ainsi il est possible d'élaguer la recherche en supprimant toutes les spécialisations de graphes qui ne sont pas généralisations de la recherche. C'est donc une recherche en largeur, par ordre ou niveau topologique, mais également contrainte par la condition précitée issue d'informations en profondeur, l'élagage par généralisation.
Nous voyons ici une application directe des règles de réfutation que constituent les règles de formation canonique. En effet, si un graphe ne correspond pas à une recherche, alors aucune de ses dérivations canoniques ne pourra correspondre à cette recherche ; de même la réfutation précise que si un graphe est faux, alors toutes ses dérivations canoniques le sont également.

Graphes généralisés :

Un autre type de structure a été présenté par [MINEAU1] et permet la représentation des graphes conceptuels à l'aide d'un niveau intermédiaire de représentation. Comme précité, les graphes conceptuels peuvent être considérés comme un ensemble de triplets, chaque triplet contenant une relation et les deux concepts qu'elle lie. Ce niveau de représentation va nous permettre d'optimiser la recherche des généralisations et de former une structure élémentaire pour décrire les graphes.

Ainsi les triplets sont étudiés dans la recherche de généralisations communes. On recherche les généralisations communes maximalement spécifiques (MSCG, Maximaly Specific Common Generalizations) parmi un ensemble de graphes. Il faut alors utiliser une structure hiérarchique à travers ces généralisations : on ne détaille dans un nœud que ce qui a été spécialisé par rapport au nœud précédent, ceci ayant l'avantage de gagner de l'espace lors de l'enregistrement des graphes.
Pour généraliser, [MINEAU1] utilise des patrons de généralisation : ils consistent à remplacer une valeur d'un triplet par un caractère générique, noté **?**. Ainsi chaque patron de généralisation pourra contenir une ou plusieurs fois ce caractère, alors que le reste du triplet restera instancié. Chaque patron disposera d'une extension : nous enregistrons cette information dans le patron à l'aide de liste d'instanciation, qui précisent quels éléments remplacent les **?** pour chaque triplet généralisé.
Nous illustrons ceci par trois graphes simples et disposant de généralisations communes :

- $\mathbf{g_1}$: « le chien mange » : **< CHIEN, AGENT, MANGER >**
- $\mathbf{g_2}$ « le chien boit » : **< CHAT, AGENT, BOIRE >**
- $\mathbf{g_3}$ « le chat mange » : **< CHIEN, AGENT, MANGER >**

Quelques généralisations communes sont exprimées à l'aide des **?**. Les listes d'instanciation sont précisées entre guillemets, chacune contenant une liste d'éléments entre parenthèses :

- **< CHIEN, AGENT, ? > { (MANGER), (BOIRE) }**
- **< CHAT, AGENT, ? > { (MANGER) }**
- **< ?, AGENT, ? > { (CHIEN, MANGER), (CHIEN, BOIRE), (CHAT, MANGER) }**

En fait la représentation que nous introduisons ici est formalisée par [MINEAU1] comme une matrice d'intersection : elle permet de spécifier à quel graphe s'applique chaque généralisation par le biais d'une liste d'instanciation. Nous obtenons dans notre cas la matrice d'intersection suivante :

	g_1	g_2	g_3
< CHIEN, AGENT, ? >	x	x	
< CHAT, AGENT, ? >			x
< ?, AGENT, ? >	x	x	x

Pour chaque triplet, nous obtenons huit généralisations possibles, y compris le triplet original et un triplet totalement général. Chaque graphe est ainsi comparé comme ensemble de triplets, dont on donne les listes d'instanciation. Par suite, dans chaque nœud de la hiérarchie se trouveront les informations communes à tous leurs enfants.

La matrice d'intersection va donc donner la place de chaque triplet dans la hiérarchie par rapport aux graphes qu'il généralise. De plus les triplets forment également une hiérarchie entre eux : certains triplets en généralisent d'autres. Or si deux triplets généralisent les mêmes graphes, et que l'un généralise l'autre, autant choisir le plus spécifique d'entre eux, représentant le plus précisément la généralisation considérée. Pour cela nous choisissons à chaque fois les ensembles de triplets qui forment une généralisation commune entre des graphes et de plus qui sont les plus spécifiques à cette généralisation. Cette considération revient à considérer pour chaque ensemble de graphes la généralisation la plus spécifique et permet de réduire considérablement le nombre de généralisations ainsi effectuées.
Cependant il n'y a pas de critère numérique pour créer une généralisation : si les graphes sont assez différents les uns des autres, nous pouvons en déduire qu'il y aura une généralisation pour chaque sous ensemble de graphes, ce qui nous conduira à créer un très grand nombre de généralisations : pour **n** graphes, nous en obtenons $\sum_{i=1..n} \mathbf{C}^{i}_{n}$ dans le pire des cas.

L'algorithme permettant de créer ce type de hiérarchie est constitué de deux parties : la création des nœuds et la création des liens entre ces nœuds, parmi un ensemble de **n** graphes. La création des nœuds consiste à construire les généralisations en tant que graphes, cette étape s'effectue en **O(n.log(n))** opérations. La création de liens entre ces nœuds consiste à lier les graphes à leurs généralisations, cette étape s'effectue en $\mathbf{O(n^2)}$ opérations.
Remarquons que la propriété de subsomption donnée par la hiérarchie de type est également utilisée, les **?** des généralisations communes sont remplacés par le sur-type commun minimal des listes d'instanciation. Ceci donne un sens sémantique plus précis à chaque généralisation et ajoute une étape supplémentaire, de faible coût, aux algorithmes considérés.

Maintenance d'une base de graphes :

Généralités sur les bases de graphes :

Nous avons donc présenté deux structures afin d'enregistrer les graphes de la manière la plus efficace possible par subsomption, comme suite d'instructions ou comme généralisations. Ces méthodes donnent alors deux manières presque équivalentes de transformer un ensemble de graphes en une structure organisée.
Maintenant se pose la question de l'évolution au cours du temps d'une base de ce type. La pérennité d'un système dépend souvent de sa capacité à évoluer au cours du temps. La plupart des systèmes, en plus d'être fonctionnels lors de leur création, doivent permettre la modification du système, pour sa manipulation en écriture et, bien entendu, toujours rester fonctionnels après ces modifications.

Nous allons donc considérer la maintenance d'une base de graphes conceptuels selon deux opérations, ici jugées fondamentales dans la manipulation de ces structures : l'*insertion* et la *délétion*. Par mesure de simplification, nous allons considérer que toute autre opération, qui consisterait à modifier un graphe, peut se traduire en la délétion, suivie de la modification, puis de la réinsertion du graphe dans la base. Pour appuyer ceci, nous remarquons que ces deux opérations sont également fondamentales pour la théorie des bases de données, et que la modification entraîne souvent des problèmes de

consistance, qui peuvent être éventuellement résolues par l'addition de triggers, mais dont la manipulation et la spécification restent incertaines.

Les bases de graphes conceptuels que nous avons examinés plus haut forment une hiérarchie de graphes, par subsomption. Les graphes n'y sont pas enregistrés de la même manière, mais les deux structures sont assez proches dans la mesure où une relation de subsomption peut-être appliquée et donner une logique à l'ensemble de la structure. La transitivité de cette relation nous est d'un grand secours pour notre étude. Nous allons donc considérer une structure générique, sans la préciser, pour pointer quelques problèmes lors de l'insertion ou de la délétion d'un graphe dans la base.

Insertion d'un graphe :

L'insertion d'un graphe dans une base revient à chercher la place du graphe parmi les graphes déjà enregistrés. La relation de subsomption sur les graphes permet donc d'effectuer une recherche, si possible topologique pour tirer parti de la transitivité de cette relation. Il nous faut trouver de quels graphes le graphe à insérer est une généralisation, puis quels graphes en sont une spécialisation. Ainsi pour insérer le graphe, nous pouvons déjà le lier à tous les graphes qui le généralisent, et à tous les graphes qui en sont une spécialisation. Pour illustrer l'insertion d'un graphe dans une base, nous allons examiner deux cas limites.
Il se peut que le graphe considéré ne soit pas une généralisation d'aucun graphe : dans ce cas il faut l'ajouter au canon, il se rapproche alors de la base de définition dont nous avons parlé plus haut. Il peut également exclure d'autres graphes de la base canonique, ses spécialisations. Il permettra alors la dérivation de nombreux autres graphes, ou l'explication de graphes jusqu'ici inexpliqués. Ce genre de graphe peut-être automatiquement obtenu par un mécanisme d'inférence.
Inversement, le graphe peut n'avoir aucunes spécialisations et constituer une feuille de la hiérarchie, dans ce cas il se rapproche de la base de faits. Il peut alors être utilisé pour des inférences mais ne modifiera en rien les capacités du système. Ce genre de graphe peut être automatiquement obtenu par dérivation canonique.

Nous voyons assez bien la relation entre l'inférence et les règles de formation canoniques, qui se complètent l'une l'autre dans un cycle d'insertion de graphes. Les règles d'inférences doivent donc être très précisément élaborées comme l'inverse des règles de formation canonique, afin que le système reste cohérent lors de l'inférence de nouveaux graphes.

Délétion d'un graphe :

La délétion d'un graphe dans une base implique la réorganisation de la base. La relation de subsomption, grâce à sa propriété de transitivité, va nous permettre de lier directement toutes les spécialisations aux généralisations du graphe effacé. Nous allons considérer les deux cas limites comme précédemment.
Si le graphe fait partie de la base canonique, alors il est possible que ses spécialisations soient maintenant inexpliquées. Nous pouvons dans ce cas soit les supprimer, soit intégrer à la base canonique toutes ses spécialisations directes. La deuxième solution, quoique apparemment préférable, peut donner lieu à de nouvelles dérivations plus hasardeuses, puisque la base canonique a été modifiée.
Au contraire, si le graphe n'a pas de spécialisation, il peut remettre en cause de définitions, qui ont éventuellement été inférées à partir de lui. Dans ce cas il peut s'avérer utile de savoir si ses généralisations ont été inférées, et comment. Mais en général il ne devrait pas perturber le système.

La délétion montre une ambiguïté à propos de la réfutation : si un graphe est considéré faux, cela devrait signifier que ses généralisations sont également fausses. Or effacer un graphe ne revient pas à le démentir, mais pourrait remettre en cause ses généralisations. Cependant contrairement à ce que suppose la théorie, nous voyons qu'effacer un graphe général semble beaucoup plus problématique qu'effacer un graphe spécialisé.

CONCLUSIONS :

Nous avons ici présenté les graphes conceptuels comme un formalisme en pleine évolution, qui repose sur un compromis entre la logique des prédicats jugée trop abstraite et le langage naturel jugé pas assez fonctionnel. L'initiateur de ce formalisme, J. F. Sowa, nous a fourni la base théorique d'un modèle fonctionnel et expressif. De nombreux auteurs ont perfectionné cette représentation.

Ce formalisme de représentation des connaissances a donc été présenté dans le rapport ci-présent, ses aspects ont été discutés. Plus particulièrement, nous avons appuyé nos réflexions sur diverses problématiques : l'utilisation des contextes et des ensembles, la spécification des définitions de types et des définitions fonctionnelles, l'application des opérateurs canoniques en tant que règles logiques, l'utilisation d'une hiérarchie comme structure de classification pour différents niveaux d'abstraction.

Nous avons formulés quelques critiques sur ces problématiques, qui paraissent nécessiter une uniformisation de la part de la communauté. Bien entendu, ce rapport n'a pas la prétention d'apporter des solutions à ces problèmes : nous avons discutés ces quelques problématiques que nous jugeons fondamentales pour l'évolution du formalisme.

Les graphes conceptuels sont aujourd'hui dans une phase d'implémentation, grâce à la conception de nombreux outils facilitant la conception de systèmes à base de graphes conceptuels. Il devient possible de développer ce type de systèmes avec un faible coût d'implémentation par rapport à ce qui devait être à chaque fois reconstruit il y a quelques années. Ainsi, il devient de plus en plus aisément concevable de développer des systèmes à base de graphes conceptuels.

L'apparition de formats et de normes, par exemple le format CGIF, l'application CharGer, la librairie java Notio semblent nous mener vers la spécification d'un modèle très fonctionnel. Les quelques problématiques que nous avons précitées paraissent s'intégrer de diverses manières dans ces modèles sans avoir été véritablement uniformisées dans l'application. Nous espérons voir les différentes solutions apportées converger, nous essayerons de contribuer à ce travail, afin qu'un modèle complet puisse en être dérivé, qui favoriserait une plus large exploitation des graphes conceptuels par l'ensemble des acteurs de l'industrie informatique.

RÉFÉRENCES :

[SOWA84] : Sowa, J.F. : Conceptual Structures: Information Processing in Mind and Machine. Morgan Kaufmann (1984). Chap. 3 : Conceptual Graphs.

[MCGREGOR91] : Sowa, J.F. : Principles of Semantic Networks, Explorations in the Representation of Knowledge. Morgan Kaufmann (1991). Chap. 13 : The Evolving Technology of Classification-based Knowledge Representation Systems.

[HARTLEY-COOMBS91] : Sowa, J.F. : Principles of Semantic Networks, Explorations in the Representation of Knowledge. Morgan Kaufmann (1991). Chap. 16 : Reasoning with Graph Operations.

[MINEAU92] : Nagle, T., Nagle, J., Gerholz, L., Eklund, P. : Conceptual Structures. Ellis Horwood Workshops (1992). Chap. 14 : Induction on Conceptual Graphs. Chap.

[LEISHMAN92] : Nagle, T., Nagle, J., Gerholz, L., Eklund, P. : Conceptual Structures. Ellis Horwood Workshops (1992). 17 : An analogical Tool : Partial Correspondences over Conceptual Graphs.

[MINEAU1] : Mineau, G. : A Universal Paradigm for Knowledge Based Structuring Methods. DepartmentofComputer Science, Laval University ().

[LEVINSON00] : Levinson, R. : Symmetry and the Computation of Conceptual Structures. ICCS'00 (2000).

[ELLIS95] : Ellis, G. : Compiling Conceptual Graphs. TKDE 7 (1995).

[ELLIS-CALLAGHAN97] : Ellis, G., Callaghan, S. : Organization of Knowledge Using Order Factors. ICCS'97 (1997).

[JAPPY-NOCK98] : Jappy, P., Nock, R. : PAC Learning Conceptual Graphs. Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (1998).

[CHAMPESME96] : Champesme, M. : Opérateurs de raffinement idéaux pour les graphes conceptuels. Laboratoire d'Informatique de Paris-Nord (1996).

[NICOLAS] : Nicolas, S., Mineau, G., Moulin, B. : Extracting Conceptual Structures from English Texts Using a Lexical Ontology and a Grammatical Parser. Laboratoire d'Intelligence Computationnelle, Laval University ().

[MINEAU-MISSAOUI] : Mineau, G., Missaoui, R. : The Representation of Semantic Constraints in Conceptual Graph Systems. Laboratoire d'Intelligence Computationnelle, Laval University ().

[MINEAU2] : Mineau, G. : The extensional Semantics of the Conceptual Graph Formalism. Laboratoire d'Intelligence Computationnelle, Laval University ().

[MINEAU-GERBE] : Mineau, G., Gerbé, O. : Contexts: A Formal Definition of Worlds of Assertion. Laboratoire d'Intelligence Computationnelle, Laval University ().