Les catégories de connaissances et leur modélisation

François RECHENMANN

INRIA Rhône-Alpes

Alors que les premiers systèmes "experts" reposaient sur l'utilisation de connaissances superficielles et opératoires, représentées essentiellement sous forme de règles, et ne visaient que la résolution de problèmes, les systèmes à base de connaissances actuels exploitent une modélisation à la fois plus fine et plus profonde des connaissances d'un domaine, associées à des connaissances opératoires. Il est de plus maintenant très largement reconnu qu'il ne suffit pas, pour concevoir de tels systèmes, de "recueillir" les connaissances nécessaires, typiquement auprès de ceux – les "experts" – qui ont développé des compétences spécifiques, fruit d'une expérience personnelle. Les connaissances doivent faire l'objet d'une véritable modélisation et le gain de compréhension du domaine obtenu à travers ce processus de modélisation vaut tout autant, et peut-être même plus, que le résultat lui-même, c'est-à-dire la base de connaissances. Dans certaines situations, des bases ne sont ainsi développées que pour expliciter, structurer, organiser et donc mieux partager, au sein d'une certaine communauté, les connaissances d'un domaine. Leurs capacités à participer à la résolution de problèmes sont alors considérées comme secondaires.

Il est clair que ce déplacement des attentes des concepteurs de systèmes à base de connaissances n'est possible que grâce au développement de modèles de connaissances[1] plus puissants et adaptés aux diverses catégories de connaissances à représenter. Une réflexion de plusieurs années, nourrie et confirmée par la conception de grandes bases de connaissances dans un domaine scientifique particulier – la biologie moléculaire[2] – a permis d'identifier quatre grandes catégories de connaissances : descriptives, comportementales, méthodologiques et contextuelles.

Connaissances descriptives

Les connaissances descriptives portent sur les entités du domaine et sur les liens qui les organisent. Les modèles dits à objets sont particulièrement bien adaptés à la représentation de ces connaissances. Le principe de ces modèles est de rassembler, dans une même unité syntaxique de description, la connaissance relative à une entité (l'unité de description est alors appelée *instance*) ou à une collection d'entités (l'unité de description est appelée *classe*). Une classe est décrite par son nom et par une liste d'attributs. Chaque attribut est défini lui-même par un nom et une liste de descripteurs, qui lui associent différentes informations telles que le type de ses valeurs ou des moyens de calculer ces valeurs quand elles sont inconnues dans une instance, par exemple des procédures de calcul (attachement procédural) ou des valeurs par défaut (Figure 1). Une instance n'est composée que d'un nom et de valeurs pour des attributs définis dans sa classe.

---

1 Le terme "modèle" est employé ici avec deux sens différents. Dans l'expression "modèle de connaissances", dérivée de l'expression "modèle de données" (voir les modèles de données hiérarchique, relationnel ou à objets), il fait référence au formalisme employé pour représenter, sous une forme exploitable par un ordinateur, des connaissances. Mais le terme "modèle" est également employé pour désigner le résultat d'un processus de modélisation, ici une base de connaissances exprimées à l'aide d'un modèle au sens de la première acception.

2 F. Rechenmann, "Knowledge bases and computational molecular biology", dans Nicolaas Mars (éd.), *Towards very large knowledge bases*, IOS press, Amsterdam (NL), p. 7-12, 1995, (actes 2nd international conference on building and sharing very large-scale knowledge bases (KBKS), Enschede (NL), 10-13 avril 1995)

Le type d'un attribut peut être une référence à un autre classe, introduisant la notion d'entité complexe : une entité qui est reliée à d'autres à travers les valeurs de ses attributs. Certains modèles traitent de façon spécifique le lien "partie de" qui relie récursivement une entité, dite composite, à ses composants. Il en résulte une structure hiérarchique (méréologie) dans laquelle des valeurs d'attributs sont susceptibles de transiter (héritage "horizontal").

```
{Surface
      sorte-de        = objet ;
      superficie      un réel ;
      sommets         liste-de Points }

{ Panneau
      sorte de        = Surface ;
      triangles       liste de Triangles ;
      superficie      méthode       M2
                                          superficie.triangles ;
      transport       un réel
                      intervalle [0.00 .. 6.00] }

{ Site d'avalanche
      sorte de        = Surface ;
      panneaux        liste de Panneaux ;
      superficie      méthode       M2
                                          superficie.panneaux }
```

Figure 1 - Exemples de descriptions de classes extraites de la base de connaissances du système Elsa d'aide à l'analyse de sites avalancheux[1]. Les classes Panneau et Site d'avalanche sont des spécialisations de la classe Surface, dont elles héritent les attributs superficie et sommets. Dans la classe Site d'avalanche, le type de l'attribut panneaux fait référence à la classe Panneaux. La valeur de l'attribut superficie, si elle est inconnue dans une instance, est susceptible d'être calculée (attachement procédural) par la méthode M2 à partir des superficies des panneaux qui composent le site.

Une capacité intéressante des modèles à objets consiste à pouvoir introduire une description de classe comme variante d'une autre. Dans ce cas, la nouvelle description hérite de la précédente ses attributs et leurs descripteurs et ajoute ses propres éléments de description supplémentaires. L'ensemble des classes est ainsi organisé en une structure hiérarchisée (taxonomie), dans laquelle une classe domine les classes plus spécifiques (Figure 2).

Les modèles à objets autorisent ainsi la description sous une forme hautement structurée de connaissances descriptives. Ils constituent de plus le support de plusieurs mécanismes d'inférence puissants. Ces mécanismes visent d'abord à compléter une instance en déterminant les valeurs d'attributs qui n'ont pas été données lors de sa création. Pour ce faire, l'héritage permet de rechercher, dans les classes et les sur-classes auxquelles appartient l'instance, des valeurs par défaut ou encore de déclencher des attachements procéduraux. Un autre mécanisme fondamental est la classification qui cherche à déterminer, pour une instance donnée, les classes auxquelles elle est susceptible d'appartenir, compte tenu des valeurs de ses attributs. Le principe est de confronter la description de l'entité à la description des classes qui composent la taxonomie. Lors de ces confrontations, il est généralement nécessaire de déterminer des valeurs d'attributs manquantes. La classification permet donc de caractériser une instance en recherchant les classes plus spécifiques auxquelles elle est susceptible d'appartenir et de compléter sa description.

---

1 L. Bisson, *Le raisonnement spatial dans les systèmes à base de connaissances ; application à l'analyse de sites avalancheux*, Thèse de doctorat de l'Université Joseph Fourier, spécialité informatique, Grenoble, 176 p., 1990

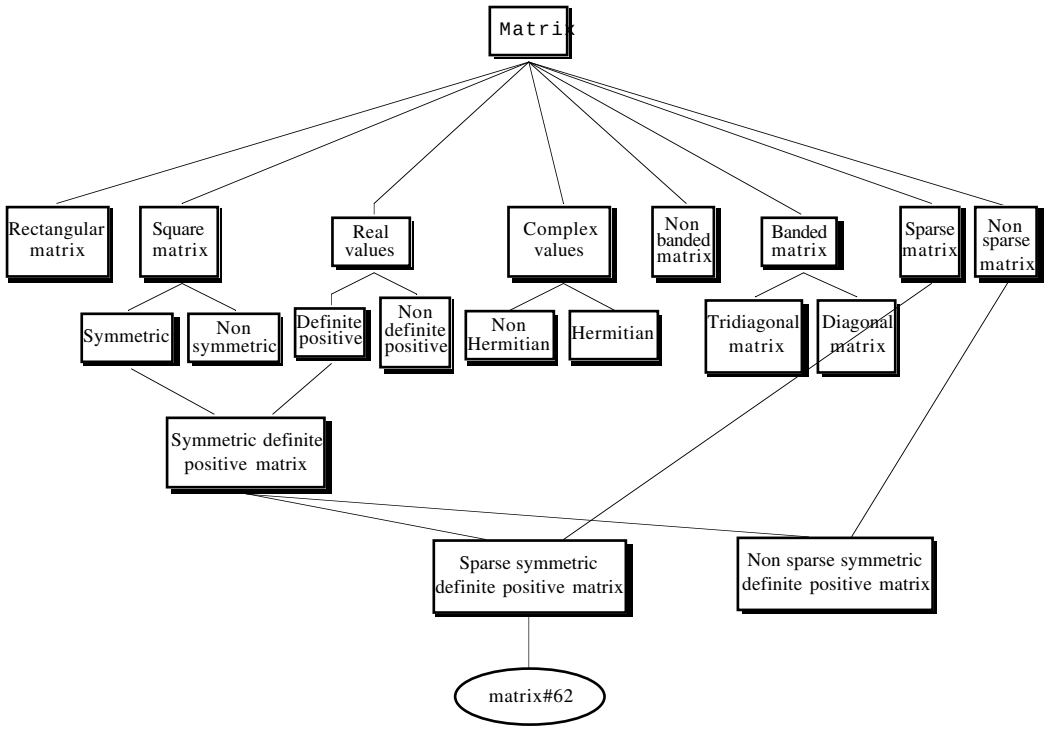

Figure 2 - Exemple de taxonomie. Des classes de matrices de plus en plus spécifiques sont décrites. La spécialisation multiple est utilisée pour créer des classes qui spécialisent simultanément plusieurs sur-classes. matrice#62 est instance de la classe Sparce symmetric definite positive matrix.

Connaissances comportementales

Il est important de souligner la différence fondamentale qui distingue, malgré les similitudes apparentes, les modèles de connaissances à objets, qui sont destinés à décrire des entités du monde réel, des langages de programmation à objets, qui sont destinés à décrire des entités qui résultent déjà d'une étape préalable d'abstraction. Cette distinction apparaît très nettement à propos de la notion de comportement.

Dans un langage de programmation à objets[1], le terme *comportement* désigne l'ensemble des méthodes exécutables qui sont associées à une classe et qui portent sur les entités informatiques instances de cette classe. Par exemple, une fenêtre d'une interface personne-machine possède un comportement déterminé par les méthodes d'ouverture, de déplacement ou de changement de taille. Certes, ces méthodes constituent bien un modèle du comportement des fenêtres, mais cette classe d'objets résulte d'une étape préalable d'abstraction. En l'occurrence, la fenêtre est un artefact.

Dans un modèle de connaissances à objets, le terme *comportement* doit naturellement faire référence au comportement de l'entité réelle, et non pas au comportement de l'entité informatique qui sert de support à la modélisation de l'entité réelle. Ce dernier comportement est d'ailleurs entièrement déterminé par la sémantique du modèle de connaissances. Utiliser un langage de programmation à objets pour modéliser directement des entités du monde réel conduit inéluctablement à mélanger ces deux niveaux de description.

Il existe bien des moyens de représenter le comportement d'une entité du monde réel. Les équations différentielles ordinaires sont ainsi utilisées dans de très nombreux domaines scientifiques. Mais leur écriture suppose une connaissance quantitative des phénomènes.

---

1 L'ouvrage de G. Masini *et al*., "Les langages à objets" (InterEditions, 1989), présente les langages à objets, aussi bien pour la programmation que pour la représentation des connaissances.

De plus, il est très vite difficile de comprendre les liens entre les comportements simulés à l'aide d'un système d'équations et l'expression même de ces équations, et, au delà, avec les hypothèses qui sous-tendent la modélisation. Afin de permettre l'étude de systèmes dynamiques, dont certains éléments ne sont connus que de façon qualitative, et afin de disposer de modèles plus explicables, plusieurs techniques de modélisation et de simulation qualitatives ont été proposées[1].

Leur principe de base est de fournir des outils de description de systèmes dynamiques dans lesquels les opérateurs usuels, tels que l'addition ou la multiplication, sont remplacés par des opérateurs qualitatifs. Par exemple, au lieu d'écrire une équation mathématique qui traduit le fait qu'une variable est proportionnelle au carré d'une autre, on se limitera, par manque de connaissances, à spécifier que les deux variables évoluent dans le même sens. Un modèle est ainsi constitué d'un ensemble de variables qualitatives reliées par de tels opérateurs. Bien entendu, les résultats de la simulation de ces modèles sont eux aussi dégradés. De plus, un même modèle, valeurs de paramètres et de conditions initiales comprises, peut produire plusieurs trajectoires différentes, car, compte tenu des indéterminations issues du caractère qualitatif des opérateurs, un même état peut être suivi de plusieurs états distincts.

Malgré leurs limites, les techniques de modélisation qualitative permettent de rassembler des connaissances incomplètes sur le comportement de systèmes dynamiques. De même que dans un langage de programmation à objets une classe regroupe la description des entités et les méthodes, il serait tout à fait pertinent, dans un modèle de connaissances à objets, de compléter les descriptions des entités en y associant une description de leur comportement, par exemple sous la forme d'équations qualitatives. Il s'agirait alors d'exploiter au mieux les caractéristiques des modèles à objets. Affiner une classe d'entités consisterait, non seulement à affiner la description de ses attributs, mais aussi des équations associées. De même, la relation de composition d'objets pourrait servir de support à un mécanisme de construction de modèles par composition de sous-modèles. Il y a quelques années, le projet Edora[2] étudiait la construction assistée de modèles en utilisant des connaissances sur les entités et les phénomènes impliqués, mais ne tentait pas d'incorporer directement dans les classes les connaissances sur le comportement des entités.

Connaissances méthodologiques

À côté des mécanismes de base – héritage, classification, attachement procédural – qui exploitent les connaissances descriptives, il est nécessaire de mettre en œuvre d'autres connaissances procédurales afin de créer, compléter et caractériser les entités dans le contexte de la résolution d'un problème complexe. Dans de nombreux domaines en effet, de grandes quantités de connaissances procédurales existent sous la forme d'ensembles de procédures, ou sous-programmes, dont l'exploitation est souvent difficile malgré la documentation textuelle qui les accompagne, que ce soit sur papier ou directement en ligne. Il paraît alors naturel de compléter ces connaissances procédurales par des connaissances déclaratives, exploitables par l'ordinateur, sur leur mode d'emploi. Il s'agit d'aider un utilisateur, lors de la résolution d'un problème, à choisir et enchaîner les procédures

---

1 Plusieurs de ces approches sont présentées dans le chapitre 7 intitulé "Raisonnement qualitatif" de l'ouvrage de J.P. Haton *et al*., *Le raisonnement en intelligence artificielle* (InterEditions, 1991). L'ouvrage de B. Kuipers, *Qualitative Reasoning - Modeling and Simulation with Incomplete Knowledge* (MIT Press, 1994) présente en détails l'une d'entre elles.

2 F. Rechenmann et B. Rousseau, "Edora : an object centered knowledge-based approach to dynamic modelling", dans J. Demongeot *et al*. (éd.), *Artificial intelligence and cognitive sciences*, Manchester University Press, 1988.

adéquates, dont l'exécution fournira la solution recherchée. Ces connaissances sur la résolution de problèmes sont appelées connaissances méthodologiques.

Les modèles de tâches ont été conçus pour décrire des démarches qui mènent à la résolution d'une classe de problèmes par décomposition récursive d'un problème particulier en sous-problèmes jusqu'à l'obtention de problèmes élémentaires qui peuvent être résolus directement par l'exécution d'une procédure ou d'un sous-programme adapté. L'exécution d'une tâche résout ainsi le problème auquel elle est associée. Une tâche est décrite en termes de ses entrées et de ses sorties, qui sont des classes d'entités du domaine, et de sa décomposition en tâches plus élémentaires, et ce grâce à des opérateurs tels que la séquence ou le choix. Pratiquement, une tâche est décrite par une classe dans un modèle à objets. Exécuter cette tâche revient alors à en créer une instance, dont seuls les attributs qui correspondent aux entrées sont valués, et à la décomposer récursivement jusqu'à exécuter les tâches élémentaires qui permettront progressivement de déterminer les valeurs des attributs qui sont associés aux sorties de la tâche initiale.

Comme toute classe, une classe qui décrit une tâche est susceptible de posséder des sous-classes qui décrivent des tâches plus spécifiques auxquelles sont associées des décompositions adaptées. On comprend donc que pour résoudre un problème, il y ait intérêt à rechercher la tâche la plus spécifique. C'est donc, ici encore, le mécanisme de classification qui permet, à chaque étape de décomposition d'un problème en sous-problèmes, donc d'une tâche en tâches plus élémentaires, de caractériser chaque sous-problème en recherchant la tâche associée la plus spécifique et de retenir ainsi la décomposition qui lui est attachée et qui se trouve alors être la plus adaptée.

La résolution d'un problème particulier passe donc alternativement par des phases de caractérisation, par classification, du (sous-)problème courant et de décomposition adaptée de ce problème en sous-problèmes plus simples, jusqu'à ce que ces sous-problèmes soient suffisamment élémentaires pour être directement résolus par l'exécution de procédures (Figure 3).

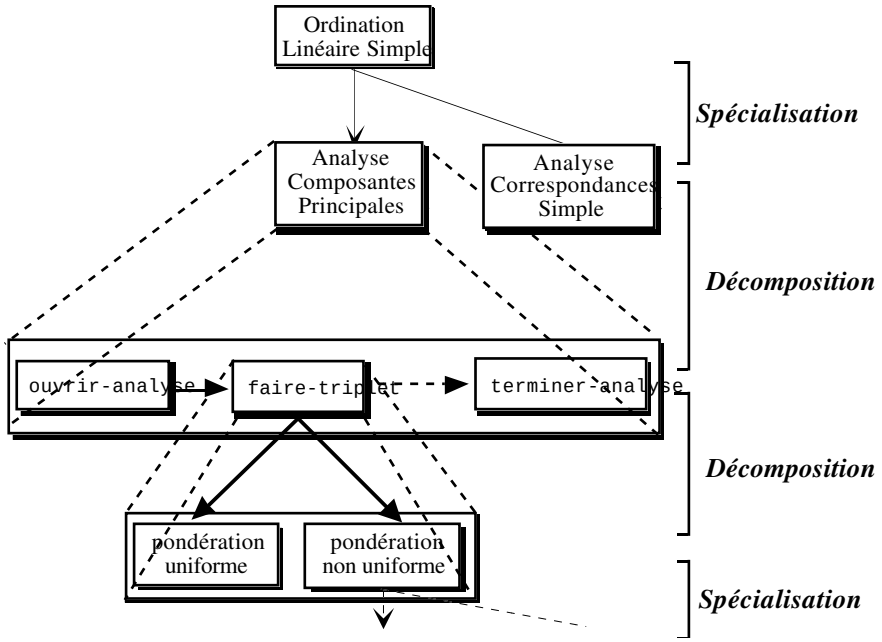

Figure 3 - Alternance des phases de spécialisation et de décomposition pour la résolution de problèmes exploitant un modèle de tâches. Une Ordination linéaire simple doit être effectuée. Selon des caractéristiques des entrées globales à traiter dans la phase de spécialisation, une analyse en composantes principales est choisie. Cette tâche est décomposée selon sa stratégie et ses sous-tâches sont à leur tour exécutées. La classe qui décrit la tâche faire-triplet n'a pas de sous-classes ; elle est directement décomposée en un choix entre pondération uniforme et pondération non-uniforme. Cette dernière est choisie et l'instance créée pour cette tâche est soumise à une nouvelle phase de spécialisation.

Connaissances contextuelles

Comme tout modèle, une base de connaissances résulte d'un long processus itératif, alimenté par des sources de connaissances multiples, et jalonné d'hypothèses et de choix de représentation, dictés par les objectifs de la modélisation. Il est donc à la fois difficile et dangereux de faire utiliser une base de connaissances par des personnes autres que celles qui ont directement participé à sa construction. Difficile, parce que les connaissances formalisées, même énoncées à l'aide d'un langage déclaratif, sont délicates à interpréter. Dangereux, car les objectifs qui ont sous-tendu la conception de la base ne correspondent sans doute pas aux attentes des utilisateurs non concepteurs. Il est donc indispensable de faire figurer, à côté des connaissances de la base proprement dite, des connaissances contextuelles – sur les sources de connaissances (typiquement des articles de revues ou des monographies), les objectifs de la modélisation et les choix de représentation – qui vont aider à prendre connaissance de la base de connaissances. Il est important de souligner que ces connaissances contextuelles ne peuvent être exprimées dans le même modèle que les connaissances de la base, sous peine d'empiler les contextes sans limite. Il s'agit donc d'associer aux éléments de connaissances formelles, typiquement des classes ou des instances dans un modèle à objets, des énoncés en langage naturel.

Une première solution immédiate consiste à associer aux noms de classes, d'attributs et d'instances un texte explicatif ou justificatif. La transformation de ces morceaux de texte en nœuds d'un réseau hypertexte peut être automatisée si les noms de la base sont systématiquement utilisés pour les indexer[1]. Il est alors possible de naviguer dans la base de connaissances (en parcourant les liens entre objets) et dans l'hypertexte, de passer à tout moment de la base à l'hypertexte et inversement (Figure 4).

La limitation majeure de la solution précédente est que la liaison entre la base et son contexte repose exclusivement sur les noms utilisés dans la base. Or, ces noms ne sont pas toujours significatifs. De plus, ils peuvent apparaître dans les textes sous des variantes lexicales, sémantiques ou multilingues. Il est donc indispensable d'établir une interface entre la base de connaissances formelles et les textes. Un lexique est susceptible de jouer ce rôle.

En fait, dans ce contexte, un lexique est amené à jouer simultanément plusieurs rôles complémentaires[2]. Le premier est effectivement de fournir une indirection dans la liaison entre les entités formelles de la base et les textes qui les commentent et les justifient. Mais le lexique constitue également une interface entre la base et l'utilisateur lui-même, donnant à ce dernier la possibilité, en navigant d'un terme à l'autre, de prendre connaissance de la base de connaissances. C'est à travers cette prise de connaissance que des modifications ou des ajouts peuvent être apportés aux connaissances formelles. Enfin, un lexique peut conduire à un choix plus judicieux et systématique des noms de classes et d'attributs.

À coté des connaissances formalisées, il est donc proposé d'énoncer des connaissances terminologiques destinées à les relier à leur contexte d'expression et d'acquisition. En fait, plusieurs méthodologies reconnues d'acquisition des connaissances suggèrent d'expliciter ces connaissances terminologiques avant même de débuter une modélisation formelle des connaissances descriptives, méthodologiques et comportementales.

---

1 S. Grivaud, F. Rechenmann, "Navigation dans les bases de connaissances associant objets et hypertexte", Actes 1es journées *Représentations par objets* (RPO), La Grande Motte, p. 262-280, 22-23 juin 1992

2 F. Lemaire "Knowledge bases, texts and lexicon", dans Nicolaas Mars (éd.), *Towards very large knowledge bases* (actes 2nd international conference on building and sharing very large-scale knowledge bases (KBKS), Enschede (NL), (10-13 avril) 1995) IOS press, Amsterdam (NL), p. 281-288, 1995

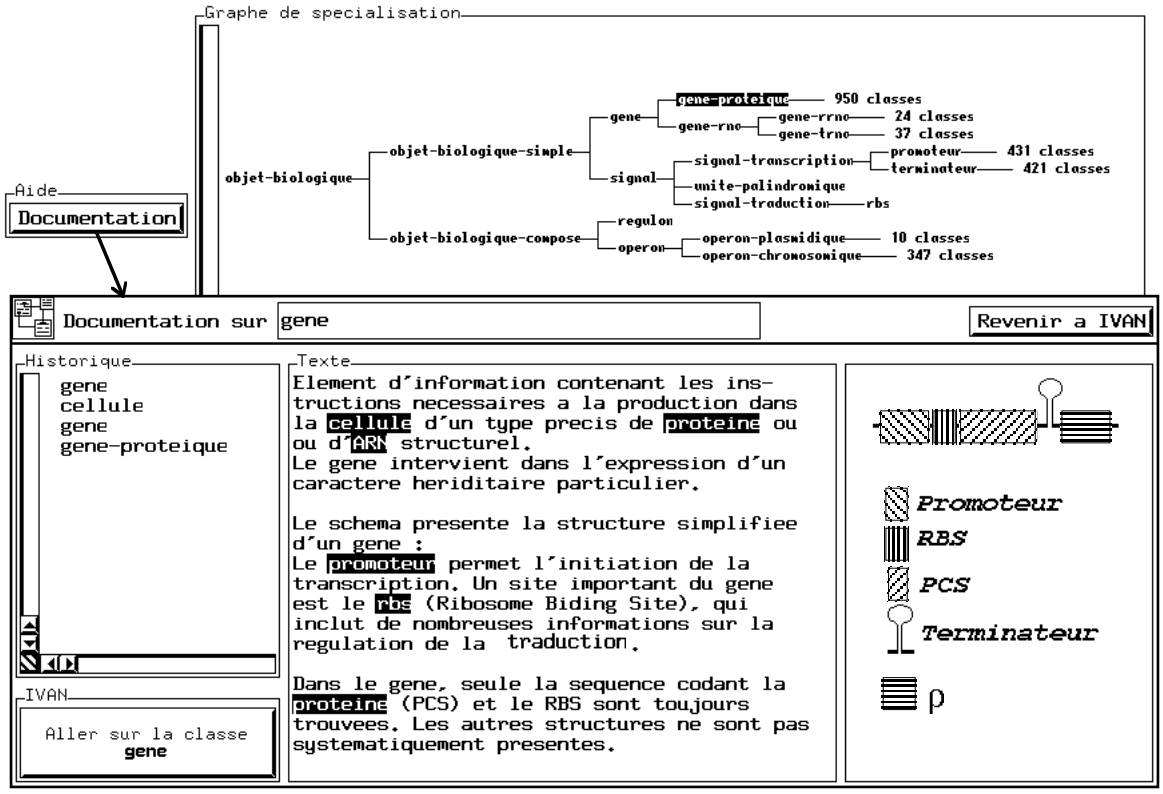

Figure 4 – Navigation dans une base associant des descriptions formelles à un réseau hypertexte. Il s'agit de la base de connaissances ColiGène sur le génome de la bactérie *Escherichia coli*[1]. À la classe gène est associé un nœud de l'hypertexte qui la commente et l'explique. À partir de ce nœud, d'autres nœuds du réseau hypertexte sont accessibles. L'utilisateur peut ainsi naviguer alternativement dans l'hypertexte et dans la base de connaissances.

Conclusion

Les recherches sur la représentation des connaissances s'inscrivaient initialement dans le grand projet de construction d'une intelligence artificielle. Plus tard, le développement des systèmes "experts" a souligné le problème de l'acquisition des connaissances sur lesquelles reposent leurs capacités à résoudre des problèmes bien définis. Rapidement, les concepteurs de ces systèmes se sont rendu compte que le recueil de connaissances était une illusion : les connaissances devaient faire l'objet d'un processus d'élaboration, de modélisation. Cependant, l'objectif principal restait malgré tout la résolution assistée de problèmes. Mais la nécessité d'expliquer des démarches de résolution et de justifier des résultats, qui reposent sur des connaissances par nature non consensuelles, a conduit à adjoindre, aux connaissances nécessaires à la résolution, des connaissances nécessaires à la production d'explications compréhensibles par l'utilisateur de ces systèmes. On peut considérer que la constitution de bases de connaissances aux seules fins de modélisation est un prolongement naturel de cette évolution.

Cette évolution s'est accompagnée de la conception de modèles de connaissances qui viennent maintenant utilement enrichir la panoplie des outils de modélisation, en particulier dans des domaines scientifiques où l'essentiel des connaissances s'exprime sous la forme de descriptions qualitatives, tant structurelles que fonctionnelles.

---

1 G. Perrière, C. Gautier "ColiGene: Object-centered representation for the study of *E. coli* gene expressivity by sequence analysis", *Biochimie*, vol. 75, 5, p. 415-422, 1993