Références bibliographiques

1 Références de collections d'articles

AAAI (1991). Proceedings of the AAAI'91 Workshop on Comparative Analysis of Explanation Planning Architectures, Anaheim, California, juillet 1991.

CS (1989). Proceedings of the 4th Annual Workshop on Conceptual Structures, AAAI Conference, Detroit, Michigan, 1989.

CS (1991). Proceedings of the 6th Annual Workshop on Conceptual Structures, Binghamton, NY, juillet 1991.

CS (1992a). Proceedings of the 7th Annual Workshop on Conceptual Structures, New Mexico, juillet 1992.

CS (1992b). Conceptual Structures: current research and practice, (Editors: Nagle T.E., Nagle J.A., Gerholz L.L. & Eklund P.W.), England, Ellis Horwood Workshops, 1992.

CS (1993). Conceptual Structures: Theory and Implementation, Lecture Notes in Artificial Intelligence, No 754 Springer-Verlag 1993 (Editors: Pfeiffer H.D. and Nagle, T.E.)(Selection of articles presented to the 7th Annual Workshop on Conceptual Graphs, Las Cruces (NM), juillet 1992).

DEXA (1990). Proceedings of the 1st International Conference on Database and EXpert Systems Applications, Vienn (Austria), août, 1990.

DEXA (1991). Proceedings of the 2nd International Conference on Database and EXpert Systems Applications, 1991.

DEXA (1993). Proceedings of the 4th International Conference on Database and EXpert systems Applications (Editors: Varik V., Lazansky J., Wagner R.R.), Prague, septembre 1993.

ECAI (1992). Proceedings of the ECAI Workshop on «Improving the Use of Knowledge-Based Systems with explanations», Vienna, Austria, août 92.

ECHT (1990). Proceedings of the 1st European Conference on Hypertext, (Editors: Rizk A., Streitz N. & André J.), INRIA, Versailles, France, novembre 1990.

ECHT (1992). Proceedings of the 4th ACM Conference on Hypertext, (Editors: Lucarella D., Nanard, J., Nanard M., Paolini P.), ACM Press, Milano, decembre 1992.

EKAW (1992). Proceedings of the 6th European Knowledge Acquisition Workshop ("Current Developments in Knowledge Acquisition"), Berlin/Heidelberg, Springer Verlag (Eds: Wetter T., Althoff K.D., Boose J.H., Gaines B.R., Linster M. & Schmalhofer F.), 1992.

EKAW (1994). Proceedings of the 8th European Knowledge Acquisition Workshop ("A Future for Knowledge Acquisition"), LN AI No 867, Berlin/Heidelberg, Springer Verlag, septembre 1994.

EXPL (1992): *Actes des deuxièmes journées «Explication» du PRC-GDR IA du CNRS*. Sophia-Antipolis, 17-19 juin 1992.

HTX (1989). Proceedings of the 1st ACM Conference on Hypertext, ACM Press, Pittsburgh, novembre 1989.

HTX (1991). Proceedings of the 3rd ACM Conference on Hypertext, ACM Press, San Antonio (Tx), 1991.

HTX (1993). Proceedings of the 5th ACM Conference on Hypertext, ACM Press, Seattle, novembre 1993.

ICCS (1993). Proceedings of the 1st International Conference on Conceptual Structures, (Lecture Notes in Artificial Inteligence, No 699, Springer-Verlag 1993)(Editors:Mineau G.W., Moulin B., Sowa J.F.), Quebec City, Canada, août 1993.

ICCS (1994). Proceedings of the 2nd International Conference on Conceptual Structure, University of Maryland, USA, août 1994.

ICCS (1995). Proceedings of the 3rd International Conference on Conceptual Structures, University of California, Santa Cruz, 14-18 août 1995 (http://www.cs.rmit.edu.au/ICCS95/).

Supplement of ICCS (1995). Supplementary proceeding of the 3rd International Conference on Conceptual Structures, University of California, Santa Cruz, 14-18 août 1995.

ICCS (1996). Proceedings of the 4th International Conference on Conceptual Structures, Sydney, Australia, 19-22 août 1996 (http://www.cs.adelaide.edu.au/~iccs96).

Supplement of ICCS (1996). Supplementary proceeding of the 4th International Conference on Conceptual Structures, Sydney, Australia, 19-22 août 1996

JAC (1993). Actes des 4èmes Journées d'Acquisition des Connaissances (JAC-93), Saint-Raphael, mars-avril 1993.

JGC PRC-GDR IA (1994). Actes de la Journée Graphes Conceptuels du PRC-GDR IA, LIRMM, Montpellier, 25 mars 1994.

KAW (1994). Proceedings of the 8th Banff Knowledge Acquisition for knowledge-based systems Workshop (Editor: Gaines B.R.), University of Calgary, Banff, Alberta, Canada, 30 janvier - 4 février, 1994.

KAW (1995). Proceedings of the 9th Knowledge Acquisition for knowledge-based systems Workshop, (Editors: Gaines B.R., Musen M.), University of Calgary, Banff, Alberta, Canada, 26 février - 3 mars 1995 (http://ksi.cpsc.ucalgary.ca/KAW/KAW95.html).

PEIRCE (1994). Proceedings of the 4th International Workshop on PEIRCE : A Conceptual Graph Workbench (Editors : Ellis G., Levinson R.), University of Maryland, USA, 19 août 1994.
(Ces actes ainsi que la plateforme Peirce sont disponible à ftp://ftp.cs.uq.oz.au/pub/peirce
La page "Peirce Group Home" est disponible à http://www.cs.adelaide.edu.au/~peirce/).

PEIRCE (1995). Proceedings of the 5th International Workshop on PEIRCE, University of California, Santa Cruz, 18 août 1995.

PRC-GDR_IA (1992). Actes des 4èmes journées du PRC-GDR Intelligence Artificielle (Editions Tecknea), Marseille, 19 - 21 octobre 1992.

PRC-GDR_IA (1995). Actes des 5èmes journées du PRC-GDR Intelligence Artificielle (Editions Tecknea), Nancy, 1-3 février 1995.

2 Références d’articles et de rapports

Acker L., Lester J. Souther A. & Porter B. (1991). *Generating Coherent Explanations to answer students's questions*. In Intelligent Tutoring Systems. Burns H., Parlett J.W., Luckhardt Redfield C. (Eds), 1991.

Acker L. & Porter B. (1994). *Extracting viewpoints from knowledge bases*. In Proc. of AAAI'94.

Afrati F. & Koutras C.D. (1990). *A Hypertext Model Supporting Query Mechanisms*. In ECHT (1990).

Aïmeur E. & Ganascia J.-G. (1993). *Elicitation of Taxonomies Based on the Use of Conceptual Graph Operators*. In ICCS (1993).

Albert P. & Vogel C (1989). *K-STATION: un environnement intégré pour le génie cognitif*. In "Génie logiciel et systèmes experts", No 19, juin 1989.

Albert P. & Jacques G. (1993). *Putting CommonKADS at work using Kads-Tool*. In Kennis-technologie’93.

Allen J. (1985). *Maintaining knowledge about temporal intervals.* In Communications of the ACM", Vol. 26, No 11, 1985.

Alpay L. (1996). *Modelling of reasoning strategies, and representation through conceptual graphs: application to accidentology*. INRIA Research Report, N. 2810, 1996.

Anjewierden A., Shadbolt N.R. & Wielinga B.J. (1992). *Supporting Knowledge Acquisition: The ACKnowledge Project*. In "Enhancing the Knowledge Engineering Process -Contributions from ESPRIT", Amsterdam, The Netherlands, Elsevier Science, 1992.

Amann B., Christophides V. & Scholl M. (1993). *HyperPATH/O2: Integrating Hypermedia Systems with Object-Oriented Database Systems.* In DEXA (1993).

Amann B. (1994). *Interrogation d’hypertextes.* Thèse de Doctorat en Informatique, CNAM de Paris, février. 1994.

Anjewierden A. & Wielemaker J. (1992). *Shelley - computer-aided knowledge engineering*. In Knowledge Acquisition (1992) 4.

Alvarez I. (1992). *Explication comparative dans les systèmes experts.* In «Les systèmes experts et leurs applications», Journées Internationales d'Avignon, mai 1991.

Appelt D.E. (1985). *Planning Natural Language Utterances.* In Cambridge University Press, Cambridge, U.K., 1985.

Bachimond B. (1996). *Ontologie et modélisation des connaissances : une approche sémantique.* Communication à la journée "Graphes Conceptuels et Applications", LIRMM, Montpellier, France, 12 février 1996.

Baker M. (1992). *Analysing rhetorical relations in collaborative problem solving dialogues*. In Proc. of the NATO Workshop "Natural Dialogue and Interactive Student Modelling", Varenna (Italy), octobre 1992.

Barrière (1995). *From Machine Readable Dictionaries to a Lexical Knowledge Base of Conceptual Graphs.* In Supplement of ICCS (1995).

Barwise J. & Perry J. (1983). *Situation and Attitudes*. MIT Press, Cambridge, MA.

Barwise J., Gawron J.M., Plotkin G. & Tutiya S, eds. (1991). *Situation Theory and its Applications.* CSLI, Stanford, CA.

Bateman J. (1990). *Upper modeling: Organizing knowledge for natural language processing*. In Proc. of Fifth International Workshop on Natural Language Generation, Pittsburgh, PA, 1990.

Bateman J., Magnini, B. & Rinaldi, F. (1994). *The Generalized Italian,German,English Upper Model*. In Proc. of the ECAI'94 Workshop: 'Comparison of Implemented Ontologies', Amsterdam. (http://www.darmstadt.gmd.de/publish/komet/papers/ecai94.ps).

Bateman J., Henschel R. & Rinaldi F. (1996). *The Generalized Upper Model Hierarchy*. (http://www.darmstadt.gmd.de/publish/komet/gen-um/node9.html).

Beeri C. & Kornatsky Y. (1990). *A Logical Query Language for Hypertext Systems*. In DEXA (1990).

Berard-Dugourd A., Fargues J., Landau M.C. & Rogala J.P. (1989). *Un système d'analyse de texte et de question/ réponse basé sur les graphes conceptuels. Informatique et santé.* Springer-Verlag, 1:1223-233, 1989.

Bernstein M. (1990). *Hypertext and technical writing*. In DEXA (1990).

Biebow B. (1992). *Comparaison entre les graphes conceptuels et un noyau KL-ONE*. In GC_PRC-GDR_IA (1994).

Biezunski M. (1995). Drafts of the "Conventions for the Application of HyTime" (CApH). ftp.techno.com//pub.CApH.docs.

Bois Ph. & Richy H. (1996). *Langage de commande d'édition structurée*. http://www.inria.fr/Rapports/opera/opera.html (section "Langage de commande d'édition structurée").

Bouaud J. (1994). *Un système de production à base de graphes conceptuels ; application dans un système de compréhension de textes*. In GC PRC-GDR IA (1994).

Bouaud J., Bachimond B, Charlet J & Zwiegenbaum P. (1994). *Acquisition and Structuring of an Ontology Within Conceptual Graphs.* In ICCS (1994).

Bouchet C. & Brunet E (1989). *SHELLEY: an Integrated Workbench for KBS Development*. In Avignon'89.

Bourigault D. (1995). *LEXTER, a Terminology Extraction Software for Knowledge Acquisition from Texts*. In KAW (1995).

Bourigault D. & Condamines A. (1995). *Réflexions sur le concept de base de connaissances terminologiques*. In PRC-GDR_IA (1995).

Bournaud I. & Ganascia J.-G. (1995). *Conceptual Clustering of Complex Objects: A Generalization Space based Approach.* In ICCS (1995).

Boy G. (1991). *Computer Integrated Documentation*. NASA Technical Memorandum 103870, septembre 1991.

Boose J.H. & Bradshaw J.M. (1988). Expertise Transfer and Complex Problems: Using AQUINAS as a Knowledge Acquisition Workbench for Knowledge-Based Systems. In "Knowledge Acquisition for Knowledge Based Systems", Vol. 2, Academic Press, 1988.

Borras, Clement P, Despeyroux D., Incerpi T & Janet (1987). CENTAUR: the system. Rapport de recherche INRIA, No 777, décembre 1987.
(Voir aussi "A CENTAUR tutorial" à ftp://ftp.inria.fr/INRIA/publication/publi-ps-gz/RT).

Breuker J., Wielinga B.J., Someren M., Hoog R., Schreiber G., Greef P., Bredeweg B, Wielemaker J. & Billault J. (1987). *Model Driven Knowledge Acquisition: Interpretation Models*. Deliverable A1, Esprit Project 1098 Memo 87, VF project Knowledge Acquisition in formal domains. Breuker (ed).

Breuker J. (1994). *A Suite of Problem Types*. In Breuker J. & van de Velde (1994).

Breuker J. & van de Velde (1994). *CommonKADS Library for Expertise Modelling. Reusable Problem Solving Components.* IOS Press, Amsterdam 1994.

Brown P.J. (1988). *Linking and Searching Within Hypertext*. In Electronic Publishing, vol. 1, No1, 1988.

Brown P.J. (1991). *Using Logical Objects to Control Hypertext Appearance.* In Electronic Publishing, 4 (2), juin 1991.

Burrow A.L. & Eklund P.W. (1995). *Visual Structure Representation Language for Conceptual Structures*. In Supplement of ICCS (1995).

Bürsner S. & Schmidt G. (1995). *Building views on conceptual models for structuring domain knowledge*. In KAW (1995) (http://www.lirmm.fr/~guinaldo/GroupeGC/papers/).

Campbell B. & Goodman J.M. (1988). *HAM: A General Purpose Hypertext Abstract Machine*. In Communications of the ACM 31(7): 856 861, juillet 1988.

Carbonneill B. & Haemmerlé O. (1994a). *ROCK: un système de Question/Réponse fondé sur le formalisme des Graphes Conceptuels*. In Actes du 9ème Congrès Reconnaissance des Formes et Intelligence Artificielle, Paris, janvier 1994.

Carbonneill B. & Haemmerlé O. (1994b). *Standardizing and interfacing relational databases using conceptual graphs.* In ICCS (1994).

Carlson L, & Nirenburg S. (1990). *World modeling for NLP*. Technical Report CMU-CMT-90-121, Center for Machine Translation, Carnegie Mellon University.

Cawsey A. (1991). *Generating Interactive Explanations*. In AAAI (1991).

Chandrasekaran B. (1987). *Towards a Functional Architecture for Intelligence Based on Generic Information Processing Tasks*. In Proc. of the 10th IJCAI, Milan, Italy, 1987.

Champesme M. (1995). *Using Empirical Subsumption to Reduce the Search Space in Learning.* In ICCS (1995).

Chein M. (1992). *Un noyau pour les graphes conceptuels simples*. In GC_PRC-GDR_IA (1992).

Chein M. & Mugnier M.L. (1992a). *Conceptual graphs, fundamental notions*. In Revue d'intelligence Artificielle, 6(4):365-406, 1992.

Chein M. & Mugnier M.L. (1992b). *Specialization: Where do the difficulties occur* ? In Proceedings of the seventh Annual Workshop on Conceptual Structures, Las cruces, New Mexico, 1992.

Chein M. (1994). *Présentation des activités sur les graphes conceptuels simples*. In GC_PRC-GDR_IA (1994).

Chen J.C., Ekberg T.W. & Thompson C. (1990). *Querying an Object-Oriented Hypermedia System.* In "Hypertext: State of the Art" (Editors: R. Mc. Aleese and C. Catherine), Billing & Songs, 1990.

Chevallet J-P (1992). *Un Modèle Logique de Recherche d'Informations appliqué au formalisme des Graphes Conceptuels. Le prototype ELEN et son expérimentation sur un corpus de composants logiciels*. Thèse de Doctorat en Informatique. Université de Grenoble 1 University, France, 1992.

Chevallet J-P (1994). *Utilisation des graphes conceptuels pour des systèmes de recherches d'informations orienté vers la précision des réponses.* In JGC PRC-GDR IA (1994).

Cogis O. & Guinaldo O. (1995). A *Linear Descriptor for Conceptual Graphs and a class for Polynomial Isomorphism Test.* In ICCS (1995).

Cole R.J. & Eklund P.W. (1996). *Application of formal concept analysis to information retrieval using a hierarchically structured thesauris.* In ICCS (1996).

Comparot-Poussier C. (1994). *HYPERBASE : Formalisation et architecture*. Thèse de Doctorat en Informatique. Université Paul Sabatier de Toulouse, France, 31 janvier 1994.

Computing Dictionary (1996). Accessible at http://wombat.doc.ic.ac.uk/

Conklin J. (1987). *Hypertext: An Introduction and Survey*. In "Computer", septembre 1987.

Conklin J. & Begeman (1988). *gIBIS: a Hypertext Tool for Team Design Deliberation.* In ACM Trans. on Information Systems, Vol. 4, No 4, octobre 1988.

Consens M.P. & Mendelzon A.O. (1990). *GraphLog: a Visual Formalism for Real Life Recursion*. In Proc. of the ACM SIGACT-SIGMOD Symp. on Principles of Database Systems, Nashville, Tenessee, 1990.

Consortium Menelas (1994). *Menelas : accès à des compte-rendus d'hospitalisation en langage naturel*. In Actes des 5èmes journées francophones d'informatique médicale, Eds Zweigenbaun P., Genève, 1994.

Corby O. & Dieng R. (1996). *Cokace: A Centaur-based environment for CommonKADS Conceptual Modelling Language*. In Proc. of ECAI'96, 12th European Conference on AI, (Ed. W Wahlster), 1996.

Creasy P. & Moulin B. (1992). *Adding Semantics to Semantic Data Models*. In CS (1992b).

Creasy P (1994). A *Role for Conceptual Graphs in Schema Integration*. In PEIRCE (1994).

Croft W.B. (1987). *Approaches to intelligent information retrieval*. In Information Processing and Management, Vol. 23, No. 4, 1987.

Cyre W.R., Balanchandar S. & Thakar A. (1994). *Knowledge Visualization from Conceptual Structures*. In ICCS (1994).

Davis H., Hall W., Heath I., Hill G. & Wilkins R. (1992). *Towards an Integrated Information Environment with Open Hypermedia Systems.* In ECHT (1992).

Davis R., Shrobe H., Szolovits P. (1993). *What is a Knowledge Representation ?* In AI Magazine, Vol. 17, Spring 1993.

De Young L. (1990). *Linking Considered Harmful*. In ECHT (1990).

Decouchant D., Quint V., Riveil M. & Vatton I. (1993). *Griffon: A Cooperative, Structured, Distributed Document Editor.* Research report No 20, Bull- IMAG, Grenoble, juin 1993.

Decouchant D. (1995). *Structured Cooperative Editing and Group Awareness*. In HCI International'95, 6th International Conference on Human-Computer Interaction (Editors: Anzai Y. and Ogawa K.), Elsevier Science, Yokohama, 9-14 July 1995 (ftp://ftp.imag.fr/pub/OPERA/doc/HCI95_alliance.ps.gz).

DeRose S.J. (1989). *Expanding the notion of links*. In HTX (1989).

Dieng R. (1991). *Relations linking cooperating agents*. In Decentralized Artificial Intelligence, 2, (Eds: Demazeau Y. & Muller J.P.), Elsevier Science Publ., North-Holland, 1991.

Dieng R. (1996). *Comparison of Conceptual Graphs For modelling Knowledge of Multiple Experts*. In proc. of ISMIS'96, Springer-Verlag LNAI 1079, Zakopane, Poland, juin 1996.

Ellis G. (1993a). *The Peirce source code.* 1993.

Ellis G. (1993b). *The Peirce user manual.* 1993.

Ellis G. & Lehmann F. (1994). *Exploiting the Induced Order on Type-labeled Graphs for Fast Knowledge Retrieval.* In ICCS (1994).

Ellis G. (1995). *Managing Complex Objects*. PhD Thesis, Computer Science Department, University of Queensland, 1995.Emerson E.A. & Halpern J.Y. (1985). *Decision Procedures and Expressiveness in Temporal Logic*. In Journal of Computer and System Sciences, Vol. 30, 1985.

Emerson E.A. & Halpern J.Y. (1985). *Decision Procedures and Expressiveness in Temporal Logic*. In Journal of Computer and System Sciences, Vol. 30, 1985.

Esch J.W. (1991). *Visualizing Temporal Intervals Represented as Conceptual Graphs.* In CS (1991).

Esch J.W. (1992). *Graphical Displays and Polymorphism*. In CS (1992b).

Esch J.W. (1994a). *Contexts and Concepts, Abstraction Duals*. In ICCS (1994).

Esch J.W. (1994b). *Contexts, Canons and Coreferent Types*. In ICCS (1994).

Esch J.W. & Levinson R. (1995). *An Implementation Model for Contexts and Negation in Conceptual Graphs.* In ICCS (1995).

Eshelman L. (1988). *MOLE: A Knowledge-acquisition Tool for Cover-and-Differentiate Systems*. In "Automating Knowledge Acquisition for Expert Systems" (Ed: MArcus S.), Boston, Kluwer, 1988.

Fargues J. (1992). *Les référents ensemblistes*. In GC_PRC-GDR IA (1992).

Fall A. (1995). *Spanning Tree Representations of Graphs and Orders in Conceptual Structures.* In ICCS (1995).

Feiner S. (1988). *Seeing the forest for the trees: Hierarchical Display of Hhypertext Structure*. In Proc. of the Conference on Office Information Systems, Palo Alto, Calif., mars 1988.

Feng C., Copeck T., Szpakowicz & Matwin S. (1994). *Semantic Clustering Acquisition of Partial Ontologies From Public Domain Lexical Sources : First Experiments*. Volume 1 of KAW (1994).

Fortune S., Hopcroft J. & Wylie J. (1980). *The directed homeomorphism problem.* In Theoretical Computer Science, 1980.

Francou L. (1993). *Conception d'un éditeur syntaxique pour le language PEPLOM.* CNAM engineer report, Grenoble Center (CUEFA), France, 1993.

Frei H.P. & Steiger D. (1992). *Making Use of Hypertext Links when Retrieving Information*. In ECHT (1992).

Frisse M.E. (1988). *Searching for Information in a Hypertext Medical Handbook.* In Communication of the ACM, Vol. 31, No 7, juillet 1988.

Fuller M., Kent A., Sacks-Davis R., Thom J., Wilkinson R. & Zobel J. (1991). *Querying in a Large Hyperbase*. In DEXA (1991).

Gallagher L., Furuta R. & Stotts P.D. (1990). *Increasing the Power of Hypertext Search with Relational Queries*. In Hypermedia , 2(1):1-14, 1990.

Garcia Ch. (1995). *Construction coopérative d'ontologies dans un cadre de multi-expertise*. Rapport de D.E.A. Informatique, University Montpellier II, France, 28 septembre 1995.

Garcia Ch. (1996a). *Construction coopérative d'ontologies dans un cadre de multi-expertise : ébauche d'un outil*. Actes de JAC'96, Journées acquisition, apprentissage, Sète, 8 mai 1996.

Garcia Ch. (1996b). *Co-operative Building of an Ontologie within Multi-Expertise Framework.* In Proc. of COOP'96, Juan-les-Pins, France, 12-14 juin 1996.

Gardiner D.A., Tjan B.S. & Slagle J.R. (1989). *Extended Conceptual Structures Notation*. In CS (1989).

Garner B.J., Tsui E., Lukose D. & Koh J. (1992). *Extendible Graph Processing in Knowledge Acquisition, Planning and Reasoning.* In CS (1992b).

GC_PRC-GDR_IA (1992). *Formalisation et extensions des graphes conceptuels*. In PRC-GDR_IA (1992).

GC_PRC-GDR_IA (1994). *Graphes conceptuels*. In PRC-GDR_IA (1994).

Genesereth M.R. & Fikes R.E. (1992). *Knowledge Interchange Format, Version 3.0 Reference Manual*. Technical Report Logic-92-1, Computer Science Department, Stanford University.

Gilbert N, Buckland D, Frolich D, Jirotka M & Luff P. (1990). *Providing advice through dialogue*. In ECAI 1990.

Godin R.., Mineau G., Missaoui R. & Mili H. (1995). *Méthodes de classification conceptuelle basées sur les treillis de Galois et applications.* In Revue d'Intelligence Artificielle, Vol. 9, No. 2, 1995.

Goldfarb C.F. (1990). *The SGML Handbook.* Clarendon Press, Oxford, 1990.

Gohsh B.C. & Wuwongse V. (1995). *A Direct Proof Procedure for Definite Conceptual Graph Programs*". In ICCS (1995).

Gomez F. (1995). *Acquiring Knowledge About the Habitat of Animals from Encyclopedic Texts*. In KAW (1995).

Gordon S.E. & Gill R.T. (1992). *Knowledge Acquisition with Question Probes and Conceptual Graph Structures*. In "Questions and Information Systems" (Eds: Lauer T., Peacock E. & Graesser A.C.), Laurence ErlBaum Associates, Publishers, Hilsdale, New Jersey, 1992.

Gruber T.R. (1992). *Towards principles for the design of ontologies used for knowledge sharing*. In Proc. de International Workshop on Formal Ontology, Eds. Gurino, Padova, Italy, 1992.

Gruber T.R. (1993). *A Translation Approach to portable Ontology Specifications*. In Knowledge Acquisition, Vol. 5, 1993.

Gruber T.R. (1994). *The Knowledge Sharing Library on the World Wide Web*. http://www-ksl.stanford.edu/knowledge-sharing/ontologies/README.html.

Guarino N. & Boldrin L. (1993). *Ontological Requirements for Knowledge Sharing.* Paper presented at the IJCAI Workshop for Knowledge Sharing and Information Interchange, Chambery, France, 1996.

Guarino N. & Carrara M. & Giaretta P. (1994). *Formalizing Ontological Commitments.* In Proc. of AAAI'94, Seattle, Washington, 31 juillet - 4 août 1984.

Guha R.V. (1991). *Contexts: A Formalization and some Applications.* In MCCTechnical Report ACT-CYC-423-91, novembre 1991.

Guichard P. (1994). *Conception et réalisation d'une interface graphique pour les graphes conceptuels*. Mémoire d'Ingéniorat CNAM, France, 1994.

Guinan C. & Smeaton A.F. (1992). *Information Retrieval from Hypertext Using Dynamically Planned Tours*. In ECHT (1992).

Gyssens M., Paredaens J. & Van Gucht D. (1990). *A Graph-Oriented Object Model for database End-User Interfaces*. In Proc. of the ACM SIGMOD Conf. on Management of Data, mai 1990.

Haemmerlé O. (1995a). *CoGITo: une plate-forme de développement de logiciels sur les graphes conceptuels*. Thèse de Doctorat en Informatique. Université de Montpellier II, France, 13 janvier 1995. (Résumé à http://www.lirmm.fr/~guinaldo/GroupeGC/papers/).

Haemmerlé O. (1995b). *La plate-formeCoGITo: manuel d'utilisation.* Research report LIRMM 95012, LIRMM, Montpellier, France, Février 1995. (http://www.lirmm.fr/~guinaldo/GroupeGC/papers/).

Haemmerlé O. (1995c). I*mplementation of Multi Agent Systems using Conceptual Graphs for Knowledge and Message representation : the CoGITo Platform*. In Supplement of ICCS (1995). (http://www.lirmm.fr/~guinaldo/GroupeGC/papers/).

Halasz F.G. & Schwartz M. (1990). *The Dexter Hypertext Reference Model*. In Proc. of the NIST Hypertext Standardization Workshop, Gaithersburg, National Institute of Standards and Technology, janvier 1990.

Harmelen F., Balder J. (1992). (*ML)2: A formal language for KADS models of expertise*. In the "Knowledge Acquisition" Journal, mars 1992.

Harmon P., King D. (1985). *Expert Systems: Artificial Intelligence in Business*. Wiley Press Book, John Wiley & Sons Inc., 1985.

Hayes-Roth F., Lenat D.B., Waterman D.A. (1983). *Building Expert Systems*. Reading MA:Addison-Wesley Publishing Company, 1983.

Heaton J.E. & Kocura P. (1993). *Presenting a Peirce Logic base inference engine and theorem prover for Conceptual Graphs*. In ICCS (1993).

Heaton J.E. & Kocura P. (1995). *Conceptual Graphs and Peirce Logic*. In supplement of ICCS (1995).

Heijst G. & Schreiber A.T. (1994). *CUE: Ontology Based Knowledge Acquisition.* In EKAW (1994).

Heijst G., Terspstra P., Wielinga B.J. & Shadbolt N.R. (1992). *Using generalised directive models in knowledge acquisition.* In EKAW (1992).

Heijst G., Schreiber A.T. & Wielinga B.J. (1996). *Using Explicit Ontologies in KBS Development*. In the International Journal of Human-Computer Studies/Knowledge Acquisition, Fall 1996. (ftp://ksi.cpsc.ucalgary.ca/www/IJHCS/VH/VH.ps.Z ou http://ksi.cpsc.ucalgary.ca:80/IJHCS/VH/).

Hopcroft J.E. & Ullman J.D. (1979). *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley Series in Computer Science, 1979.

Hofmann M, Schreiweis U & Langendörfer H. (1990). *An Integrated Approach of Knowledge Acquisition by the Hypertext System CONCORDE*. In ECHT (1990).

Hovy E.H. (1988). *Planning Coherent Multisentential Tests*. In Proceedings of the 26th Annual Meeting of the association of Computational Linguistics, juin 1988.

Huibers T., Ounis I. & Chevallet J.P. (1995). *Axiomatization of a Conceptual Graph Formalism for Information Retrieval in a Situated Framework*. Research report of group MRIM/LGI-IMAG, RAP95-004, 1995 (file://ftp.imag.fr/pub/SIRI/publications/1995/huibers95a.ps.gz").

Hurwitz A. & Rich W. (1993). *GML for Conceptual Graph Networks*. IBM Technical Report No 03.485, mai 1993.

ILOG (1993a). *K-STATION : version 1.1 : Manuel de l'interface-utilisateur, Manuel de référence*. février 1993.

ILOG (1993b). *KADS TOOL : version 1.0 : Manuel de l'interface-utilisateur, Guide méthodologique.* 1993.

ILOG (1993c). *ILOG Views : version 1.2 : Manuel de référence*. 1993.

ISO 8613 (1987). *Information processing - Text and office systems - Office Document Architecture (ODA) and Interchange Format (ODIF)*. International Standard ISO, DIS 8613, 1987.

ISO 8879 (1986). *Information processing - Text and office systems - Standard Generalized Markup Language (SGML)*. International Standard ISO 8879, First edition, 1986.

ISO 10744 (1991). *Hypermedia/Time-based structuring language (HyTime)*. Draft International Standard ISO/IEC, DIS 10744, avril 1991.

ISO 10179 (1991). *Information Technology - Text and office systems - Documents Style and Semantics and Specification Language (DSSSL)*. International Standard ISO, DIS 10179, 1991.

Jonker W., Spee J.W. (1992). *Yet another formalisation of KADS Conceptual Models*. In Proc. of EKAW'92.

Kabbaj A. Frasson C., Kaltenbach M., Djamen J.. *A conceptual and Contextual Object-oriented Logic Programming: the PROLOG++ Language*. In ICCS (1994).

Kamp H. (1981). *Event, discourse representations, and temporal references*. In "Languages", Vol. 64, 1981.

Kavanagh J. (1996). *IKARUS*. In http://www.csi.uottawa.ca/~kavanagh/Ikarus/IkarusInfo.html

Kheirbek A. & Chiaramella Y. (1995). *Integrating Hypermedia and Information Retrieval with Conceptual Graphs*. In HIM'95, Konstanz, Germany, April, 1995 (http://www-lgi.imag.fr/Les.Personnes/Ammar.Kheirbek/).

Knight K. & Luk S. (1994). *Building a Large-Scale Knowledge Base for Machine Translation*. In Proc. of AAAI'94, twelfth national conference on artificial intelligence, juillet 1994.

Kremer R. (1995). *The Design of a Concept Mapping Environment for Knowledge Acquisition.* In KAW (1995).

Kuntzmann-Combelles A. and Vogel C. (1988). *KOD: a support environnement for cognitive acquisition and management*. In Safety and Reliability Symposium, 1998.

Labidi S. (1995). *Ingénierie de la connaissance dans la cadre de projets multi-experts : méthode, techniques et outil*. Thèse de Doctorat en Informatique, Université de Nice - Sophia Antipolis, 1995.

Langevelde I.A., Philipsen A.W. & Treur J. (1992). *Formal spécification of compositional architecture*. In Proc. of ECAI'92, 10th European Conference on Artificial Intelligence (Editor: Neumann), Wiley J. & Sons, Chichester, 1992.

Leane J. (1993). *GRIT: An Implementation of a Graphical User Interface for Conceptual Structures*. Report of B.Sc. (Maths & Comp. Sc.), University of Adelaide, Australia, novembre 1993.

Leclère M. (1995). *Les connaissances du niveau terminologique du modèle des graphes conceptuels : construction et exploitation.* Thèse de Doctorat en Informatique. Université de Montpellier II, France, 21 décembre 1995.

Leclère M. (1996). *C-CHIC : Construction coopérative de hiérarchies de catégories*. In RIA, Revue d'intelligence Artificielle, Numéro spécial Graphes Conceptuels, Vol. 10, 1996.

Lehmann F & Cohn A.G. (1994). *The EGG/YOLK Reliability Hierarchy: Semantic Data Integration Using Sorts With Prototypes*. In Proc. of CIKM'94 (Eds: Adam & Yesha), ACM Press, New York, 1994.

Lenat D.B. & Guha R.V. (1990a). *Building large knowledge-based systems: representation and inference in the Cyc project*. Reading, MA; Sydney; Tokyo:Addison-Wesley, 1990.

Lenat D. & Guha R.V. (1990b). *CYC: towards programs with common sense*. Communications of the ACM, Vol. 33 No 8, août 1990.

Lemaire B. (1992). *Aspects constructifs de la production d'une explication : l'architecture ESMERALDA*. In EXPL (1992).

Lemaire F. (1995). *Knowledge bases, texts and lexicon*. In Proc. of KBKS'95, Enschede, avril 1995.

Lester J.C. & Porter B.W. (1991). *An Architecture for Planning Multi-Paragraph Pedagogical Explanation*. In AAAI (1991).

Levinson R. & Ellis G. (1992). *Multi-level hierarchical retrieval*. In Knowledge Based Systems, 5(3), 1992.

Levinson R. (1994). *UDS: A Universal Data Structure*. In ICCS (1994).

Liddy D.E. & Sung H.M. (1992). *DR_LINK Document Retrieval using Linguistic Knowledge, Project Description*. In Revue ACM: SIGIR Forum Vol 26 no 2, Fall 1992.

LIRMM (1996). *Research activity*. http://www.lirmm.fr/~guinaldo/GroupeGC/RA.html

Long D. & Garigliano R. (1994). *Reasoning by Analogy and Causality: Model and Applications*. Ellis Horwood, 1994.
(ftp:ftp.dur.ac.uk/pub/lolita ou http://www.dur.ac.uk/~dcs3mhs/lolhome.html)

Lucarella D. (1990). *A Model for Hypertext-Based Information Retrieval*. In ECHT (1990).

Lucarella D., Parisotto S. & Zanzi A. (1993). *MORE: Multimedia Object Retrieval Environment*. In HTX (1993).

Lukose D. (1995). *Using Executable Conceptual Structures for Modelling Expertise*. In KAW (1995).

Lukose D., Mineau G., Kremer, Moeller J., Mugnier M & Martin Ph. (1995). *Conceptual Structures for Knowledge Engineering and Knowledge Modelling*. In supplement of ICCS (1995).

Mabrouk M (1992). *Modèle d'hyperdocument basé sur le standard ISO 8613 ODA*. Thèse de Doctorat en Informatique. Institut National des Sciences Appliquées, Lyon, France, juin 1992.

MacGregor R. (1988). *A deductive pattern matcher*. In Proc. of the 7th National Conference on Artificial Intelligence, USA, 1988.

MacGregor R. (1991). *The evolving technology of classification-based knowledge representation systems*. In "Principles of Semantic Networks: Explorations in the Representation of Knowledge" (Ed.: Sowa J.), Morgan Kaufmann, 1991.

Mahesh K. (1996). *A Guided Tour of the Mikrokosmos Ontology*.
http://crl.nmsu.edu/users/mahesh/ontology-guided-tour/ontology-guided-tour.html

Major N. & Reichgelt H. (1990). *ALTO: an Automated Laddering Tool*. In "Current Trends in Knowledge Acquisition", Amsterdam, The Netherlandfs, IOS Press, 1990.

Mann W. & Thompson S. (1988). *Rhetorical Structure Theory : Toward a functional theory of text organization*. In Proc. of Text, 8, 3, 243-281.

Marcus S. & McDermott J. (1989). *SALT: A Knowledge Acquisition Language for Propose-and-Revise Systems*. In Artificial Intelligence, 39(1), 1989.

Marino D.O. (1993). *Raisonnement classificatoire dans une représentation à objets multi-points de vue*. Thèse de Doctorat en Informatique, Grenoble 1 University, France, 1993.

Marman M & Schlageter G. (1992). *Towards a Better Support for Hypermedia Structuring: The HyDESIGN Model*. In ECHT (1992).

Marshall C. & Irish P. (1989). *Guided Tours and On-line Presentations: How authors Make Existing Hypertext Intelligible for Readers*. In HTX (1989).

Marshall C.C, Halasz F.G., Rogers R.A. & Janssen W.C. (1991). *Aquanet, a hypertext tool to hold your knowledge in place*. In HTX (1991).

Martin Ph. (1993a). *Adaptation de KADS pour la construction de Systemes à Base de Connaissances explicatifs*. In JAC (1993). (Available at http://www.inria.fr/acacia/Publications/1993/java93phm.ps.Z).

Martin Ph. (1993b). *A KADS refinement for Explanatory Knowledge Extraction and Modelling*. In Proc. of the 6th Australian Joint Conference on Artificial Intelligence (AI'93), 16-19 November 1993. (http://www.inria.fr/acacia/Publications/1993/ai93phm.ps.Z).

Martin Ph. (1994). *La méthodologie d'acquisition des connaissances KADS et les explications*. INRIA research report No 2179, January 1994.

Martin Ph. (1995a). *Knowledge Acquisition Using Documents, Conceptual Graphs and a Semantically Structured Dictionary.* In KAW (1995). (http://www.inria.fr/acacia/Publications/1995/kaw95phm.ps.Z).

Martin Ph. (1995b). *Links between Electronic Documents and a Knowledge Base of Conceptual Graphs*. In supplement of ICCS (1995). (http://www.inria.fr/acacia/Publications/1995/iccs95phm.ps.Z)

Martin Ph. (1995c). *Using the WordNet Concept Catalog and a Relation Hierarchy for Knowledge Acquisition*. In PEIRCE (1995). (http://www.inria.fr/acacia/Publications/1995/peirce95phm.ps.Z)

Martin Ph. & Alpay L. (1996). *Conceptual Structures and Structured Documents.* In ICCS (1996). (http://www.inria.fr/acacia/Publications/1996/iccs96hm.ps.Z).

Maybury M.T. (1991). *An evaluation of Plan-Based Explanation Architecture*. In AAAI (1991).

McCall R.J., Bennet P.R., D'Oronzo P.S., Ostwald J.L.,, Shipman F.M., & Wallace N.F. (1990). *PHIDIAS: Integrating Graphics into Dynamic Hypertext*. In ECHT (1990).

McKeown K.R., Wish M., Matthews M. (1995). *Tailoring explanations for the user*. In Proceedings of the 9th International Joint Conference on Artificial Intelligence, Los Angeles, CA, août 1985.

McKeown K., Wish M. & Matthews K. (1985). *Tailoring explanations for the user*. In Proc. of the 9th. IJCAI, Vol. 2, 1985.

MHEG (1990). *Coded Representation of Multimedia and Hypermedia Information.* Working Document for the future MHEG Standard "S3", ISO/IEC, 1990.

Michaski R.S., Stepp R.E. & Diday E. (1981). *A Recent Advance in Data Analysis: Clustering Objects into Classes Characterized by Conjunctive Concepts*. In "Progress in Pattern Recognition" (Eds: Kanal L.N.m Rosenfeld A.), North-Holland Publishing Company, 1981.

Miller G.A., Beckwith R., Fellbaum C., Gross D. & Miller K. (1990). *Five Papers on WordNet*. CSL Report 43, Cognitive Science Laboratory, Princetown University, July 1990.
(Ces articles, le système et un prototype sont accessibles à http://www.cogsci.princeton.edu/~wn/ ;
un prototype de "International WordNet" est accessibles à http://nlp01.cs.ul.ie/iwn_main_nlp_demo.html et des détails sur "*EuroWordNet"* sont accessibles à http://www.let.uva.nl/CCL/EuroWordNet.html).

Mineau G. (1990). *Structuration des Bases de Connaissances par Généralisation.* Thèse de Doctorat en Informatique, Université de Montréal, Canada, 1990.

Mineau G. & Allouche M. (1995). *Establishing a Semantic Basis: Toward the Integration of Vocabularies* . In KAW (1995).

Moeller J.-U. (1995). *Operationalisation of KADS Models by using Conceptual Graph Modules.* In KAW (1995).

Moeller J.-U. & Willems M. (1995). *CG-DESIRE: Formal Specification Using Conceptual Graphs.* In KAW (1995).

Moeller J.-U. & Detlev W. (1996). *Editing Conceptual Graphs.* In ICCS (1996).

Montague R. (1974). *Formal Philosophy.* Yale University Press, New Haven, 1994.

Moore J.D., Paris C.L. (1991). *The EES Explanation facility: its tasks and its architecture*. In AAAI (19 91).

Moulin B. (1995a). *Discourse Spaces: A Pragmatic Interpretation of Contexts*. In ICCS (1995).

Moulin B. (1995b). *A Pragmatic Representational Approach of Context and Reference in Discourses*. In ICCS (1995).

Mugnier M.L. & Chein M. (1992). *Polynomial Algorithms for Projection and Matching*. In CS (1992a).

Mugnier M.L. (1994). *On specialization/generalization for conceptual graphs*. In "Journal of Experimental and Theoretical Artificial Intelligence", 1994.

Mugnier M.L. & Chein M. (1996). *Représenter des connaissances et raisonner avec des graphes*. In RIA, Revue d'intelligence Artificielle, Numéro spécial Graphes Conceptuels, Vol. 10, 1996.

Munday C., Sobora F. & Lukose D. (1994). *UNE-CG-KEE: Next Generation Knowledge Engineering Environment.* In Proc. of the 1st Australian Workshop on Conceptual Structures, 1993.
(CGKEE est accessible à turing.une.edu.au /pub/CGKEE).

Musen M.A., Fagan L.M., Combs D.M. & Shortliffe E.H. (1988). *Use of a Domain Model to Drive an Interactive Knowledge Editing Tool.* In "Knowledge-Based Systems", Vol. 2, London. Academic Press, 1989.

Musen M.A. (1989). *Automated Generation of Model-Base Knowledge-Acquisition Tools*. In Research Notes in Artificial Intelligence, London, Pitman, 1989.

Myaeng S.H. (1992). *Conceptual Graphs as a Framework for Text Retrieval*. In CS (1992b).

Myaeng S.H. & Khoo C. (1994). *Linguistic Processing of Text for a Large-Scale Conceptual Information Retrieval System*. In ICCS (1994).

Nanard J. & Nanard M. (1991). *Using Structured Types to incorporate Knowkedge in Hypertext*. In HTX (1991).

Nanard J. & Nanard M. (1993a). *Should anchors be typed too ? An experiment with MacWeb*. In HTX (1993).

Nanard J., Nanard M., Massotte A-M., Djemaa A. Joubert A., Betaille H. & Chauché J. (1993a). *Integrating Knowledge-base Hypertext and Database for Task-oriented Access to Documents*. In DEXA (1993).

Nanard J., Nanard M., Massotte A-M., Djemaa A. Joubert A., Betaille H. & Chauché J. (1993b). *La métaphore du généraliste : Acquisition et utilisation de la connaissance macroscopique sur une base de connaissances techniques.* In JAC (1993).

Nazarenko A. (1992). *Les types de concepts spéciaux*. In GC_PRC-GDR IA (1992).

Nazarenko A. (1994). *Des graphes conceptuels structurés pour représenter la sémantique des phrases complexes*. In JGC PRC-GDR IA (1994).

Neches R. (1994). *The Knowledge Sharing Effort*. http://www-ksl.stanford.edu/knowledge-sharing/papers/kse-overview.html.

Neubert S., Pirlein T. & Schmidt G. (1994). *Top-Down Knowledge Acquisition.* In Proc. of the International Conference on Experts Systems for Development, April 1994.

Newell A. (1982). *The Knowledge Level*. In Artificial Intelligence, Vol. 18, 1982.

Nguyen G.T., Rieu D. & Escamilla J. (1992). *An Object Model for Engineering Design*. In ECOOP'92 (European Conference on Object-Oriented Programming), Utrecht, The Netherlands, juin-juillet 1992.

Nogier J.F. (1991). *Génération automatique de langage et de graphes conceptuels*. Hermès, Paris, 1991.

Normark K. & Orsterbye K. (1996). *Rich Hypertext: a Foundation for Improved Interaction Techniques*. In international Journal of Human Computer Studies, "Incorporating Knowledge Acquisition", Eds: Gaines B.R., Academic Press, 1996.

Nwana H., Paton R., Bench-Capon T & Shave M. (1991). *Facilitating the Development of Knowledge Based Systems: a Critical Review of Acquisition Tools and Techniques*. In AI Communication, Vol. 4(2/3), 1991.

Ogden C.K. & Richards I.A. (1923). *The Meaning of Meaning*. Harbourt, Brace, and World, New York, 8th edition 1946.

OPERA (1996). *Projet OPÉRA - rapport d'activité 1995*. http://www.inria.fr/Rapports/opera/opera.html

OSF (1992). *Open Software Foundation*. OSF/MOTIF Series (5 volumes), 1992.

PEIRCE (1994). Proceedings of the 4th International Workshop on PEIRCE : A Conceptual Graph Workbench (Editors : Ellis G., Levinson R.), University of Maryland, USA, 19 août 1994.
(Ces actes ainsi que la plateforme Peirce sont disponible à ftp://ftp.cs.uq.oz.au/pub/peirce
La page "Peirce Group Home" est disponible à http://www.cs.adelaide.edu.au/~peirce/;).

Pierra G. (1991). *Les bases de la programmation et du génie logiciel*. Dunod Informatique, 1991.

Pfeiffer H.D. & Hartley R.T. (1992). *The Conceptual Programming Environment, CP*. In CS (1992b).

Poole J & Campbell J.A. (1995). *A Novel Algorithm for Matching Conceptual and Related Graphs*. In ICCS (1995).

Puerta A.R., Egar J., Tu S.W. & Musen M.A. (1992). *A Multiple-Method Shell for the Automatic Generation of Knowledge Acquisition Tools*. In "Knowledge Acquisition", Vol. 4, 1992.

Quint V., Nanard M. & André J. (1990). *Towards Document Engineering*. In Electronic Publishing 1990 (Editor: Furuta R.), Cambridge University Press, septembre 1990.

Vercoustre A.-M. (1990). *Structured Editing--Hypertext Approach: Cooperation and Complementarity.* In Electronic Publishing 1990 (Editor: Furuta R.), Cambridge University Press, septembre 1990.

Quint V. & Vatton I. (1992). *Combining Hypertext and Structured Documents in Grif*. In ECHT (1992).

Quint V. & Vatton I. (1994a). *Making Structured Documents Active*. In Electronic Publishing -- Origination, Dissemination and Design, vol. 7, num. 2, juin 1994.

Quint V. & Vatton I. (1994b). *Active Documents as a Paradigm for Human-Computer Interaction*. In Workshop Research issues in the intersection between software engineering and human-computer interaction, Sorrento, Italy, mai 1994.

Quint V. (1995). *Les langages de Grif / The languages of Grif* (translated by E. Munson). Technical report of INRIA-IMAG, 2 rue de Vignate, 38610 Gières - France, 1995.

Quint V., Richy H., Roisin C., Vatton I. (1995). *Grif Manuel utilsateur*. Rapport technique de INRIA-IMAG, 2 rue de Vignate, 38610 Gières - France, 1995.

Quint V. & Vatton I. (1995a). *The Grif Editing Tool Kit*. Technical report of INRIA-IMAG, 2 rue de Vignate, 38610 Gières - France, 1995.

Quint V. & Vatton I. (1995b). *Le mécanisme d'appels externes de Grif*. Technical report of INRIA-IMAG, 2 rue de Vignate, 38610 Gières - France, 1995.

Rademakers Ph. & Vanwelkenhusen J. (1992). Generic Models and Their Support in Modelling Problem Solving Behavior. In "Second Generation Expert Systems" (Eds: David J-M., Krivine J-P., Sommons R.), Springer-Verlag, 1992.

Rastier F., Cavazza M & Abeillé A. (1994). *Sémantique pour l'analyse*. Paris: Masson, France, 1994.

Richy H. (1994). *A hypertext electronic index based on the Grif structured document editor*. In Proc. of Electronic Publishing -- Origination, Dissemination and Design, vol. 7, num. 1, mars 1994.

Ribière M., Dieng R, Blay-Fornarino M. & Pinna-Dery F. (1996). *Link-based Reasoning on Conceptual Graphs*. In Supplement of ICCS (1996).

Rizk A. & Sauter L. (1992). *Multicard: An Open Hypermedia System*. In ECHT (1992).

Roberts D.D. (1973). *The existential Graphs of Charles S. Peirce*. Mouton, The Hague, 1973.

Runkel J.T. & Birmingham W.P. (1994). *Separation of knowledge: a key to reusability*. In KAW (1994).

Sabah G. & Briffault X. (1993). *Caramel: A step towards reflection in natural language understanding systems*. In Proceedings of the IEEE International conference on Tools with Artificial Intelligence, Boston, 1993.

Salvat E. (1993). *Algorithmes de projection pour les graphes conceptuels*. Rapport de D.E.A, Université des sciences et techniques du Languedoc, 1993.

Salvat E., Mugnier M.-L. (1995). *Sound and Complete Forward and Backward Chainings of Graph Rules.* Research report LIRMM I96-004, LIRMM, Montpellier, France, January 1995. (http://www.lirmm.fr/~guinaldo/GroupeGC/papers/).

Sauter L., Ahedo M. & Dagan P. (1992). *Intégration de Hyperpath avec le SGBD orienté-objet Ontos: An Open Hypermedia System*. In ECHT (1992).

Schaar Ph. (1994). *Un environnement de programmation pour le langage graphique Argos.* CNAM engineer report, IMAG, Grenoble, France, mars 1994.

Serres C. (1995). *Modèles et modules pour les systèmes à base de connaissances : l'approche CERISE*. Thèse de Doctorat en Informatique. Université de Paris 6, France, octobre 24, 1995.

Shadbolt N.R. & Wielinga B.J. (1990). *Knowledge- Based Knowledge Acquisition: the Next Generation of Support Tools*. In "Current Trends in Knowledge Acquisition", Amsterdam, The Netherlandfs, IOS Press, 1990.

Schmidt G. (1995). *Building Views on Conceptual Models for Structuring Domain Knowledge*. In KAW (1995).

Schreiber A.T., Wielinga B.J., Akkermans J.M., Van de Velde W. & AnjeWierden A. (1992). *CML: The CommonKADS Conceptual Modelling Language*. In EKAW (1994).

Schuler W. & Smith J.B. (1990). *Author's Argumentation Assistant (AAA): A Hypertext-Based Authoring Tool for Argumentative Texts*. In ECHT (1990).

Schutt H.A. & Streitz N.A. (1990). Hyperbase: a Hypermedia Engine Based on a Relational Database management System. In ECHT(1990).

Skuce D. (1995). *Conventions for Reaching Agreement on Shared Ontologies*. Proc. de KAW'95, Gaines B.R. Eds, Univ. of Calgary, Banff, Canada, février 1995.

Skuce D. & Lethbridge T.C. (1995). *CODE4: A Unified System for managing Conceptual Knowledge*. In International Journal of Human-Computer Studies (1995) 42:413-451, 1995.
(Voir également http://www.csi.uottawa.ca/~doug/CODE4.html)

Skuce D. (1996). *ClearTalk Concepts*. In http://www.csi.uottawa.ca/~kavanagh/Ikarus/Cleartalk.html, April 1996.

Searle J.R. (1985). *Foundations of Illocutionary Logic*. (Editor: Vanderveken D.), New York, Cambridge Univ. Press, 1985.

Sowa J.F. (1984). *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, Reading, MA.

Sowa J.F. - Editor (1991). *Principles of Semantic Networks: Explorations in the Representation of Knowledge*. (Ed.: Sowa J.F.), Morgan Kaufmann, 1991.

Sowa J.F. (1992). *Conceptual Graphs Summary*. In CS (1992b).

Sowa J.F. (1993a). *Relating Diagrams to Logic*. In ICCS (1993).

Sowa J.F. (1993b). *Logical foundations for representing object-oriented systems.* In Journal of Experimental & Theoretical Artificial Intelligence, 5, 1993.

Sowa J.F. (1995a). *Syntax, Semantics, and Pragmatics of Contexts*. In ICCS (1995).

Sowa J.F. (1995b). *Distinctions, Combinations, and Constraints.* Article distribué par Sowa durant la "3rd International Conference on Conceptual Structures (ICCS'95), University of California, Santa Cruz", 14-18 août 1995.

Steels L. (1990). *Components of expertise.* In AI Magazine, 1990.

Steels L. (1993). *The Componential Framework and its Role in Reusability.* In "Second Generation Expert Systems", Berlin Heidelberg, Germany, Springer-Verlag, 1993.

Stotts P. & Furutta R. (1989). *Petri-Net-Based Hypertext: Document Structure with Browsing Semantics*. In ACM Transactions on Information Systems, Vol. 7, No 1, 1989.

Streitz N., Haake J., Hannemann J., Lemke A., Schuler W., Schütt H. & Thüring M. (1992). *Sepia: a Collaborative Hypertext Writing System*. In ECHT (1992).

Sutcliffe R. (1995). Communication personnelle. Une démonstration d'un prototype de WordNet 1.6, le WordNet international est accessible à http://nlp01.cs.ul.ie/iwn_main_nlp_demo.html).

Suthers D. (1993). *An analysis of explanation and its implications for design of explanation planners*. Ph.D. thesis. University of Massachusetts, 1993.

Takeda H., Iino K., Nishida T. (1994). *Agent Communication With Multiples Ontologies*. In Proc. of the W3 Workshop on Heterogeneous Cooperative Knowledge Base, December 15-16, 1994.

Takeda H., Iino K., Nishida T. (1994). *Agent Communication With Multiples Ontologies*. In Proc. of the W3

Tate A. (1994). *A Plan Ontology*. Working document, Artificial Intelligence Applications Institute, 31 octobre 1994.

Tepfenhart W.M. (1992). *Using the Situation Data Model to Construct a Conceptual Basis Set.* In CS (1992b).

Tjan B.S., Gardiner D.A. & Slagle J.R. (1992). *Representing and Reasoning with Set Referents and Numerical Quantifiers*. In CS (1992b).

Toulmin S. (1958). *The Uses of Argument*. Cambridge University Press, 1958.

Tu S.W., Erikson H., Gennari J.H., Shahar Y, & Musen M.A. (1995). *Ontology-Based Configuration of Problem-Solving Methods and Generation of Knowledge Acquisition Tools: The Application of PROTEGÉ-II to Protocol-Based Decision Support*. In Articficial Intelligence in Medecine, 1995.

Ullman J.D. (1988). *Principles of Database and Knowledge Base System*. In Computer Science Press, Vol. 1, 1988.

Utting K. & Yankelovich N. (1989). *Context and Orientation in Hypermedia Networks*. In ACM Transactions on Information Systems, Vol. 7, No1, 1989.

Van Melle W. (1979). *A Domain Independent Production Rule System for Consultation Programs.* In IJCAI'79.

Vilnat A. (1992). *Introduction d'un ordre sur les types de relations*. In GC_PRC-GDR_IA (1992).

Vlissides J. (1990). *Generalized Graphical Object Editing*. Research report CSL-TR-90-427, Computer Systems Laboratory, Stanford University, juin 1990.

Vogel C. (1988). *Génie cognitif.* Masson, 1988.

Vossen P. (1996). *EuroWordNet:Building a multilingual database with wordnets for several European languages*. http://www.let.uva.nl/CCL/EuroWordNet.html.

Wermelinger M. (1995a). *Conceptual Graphs and First-order Logic*. In ICCS (1995).

Wermelinger M. (1995b). *Conceptual Structures Linear Notation: A Proposal for PEIRCE*. In PEIRCE (1995).

Wiederhold G. (1994). *Interoperations, Mediation and Ontologies*. In Proc. of the W3 Workshop on Heterogeneous Cooperative Knowledge Base, December 15-16, 1994.

Wielinga B., Schreiber G. & Breuker J. (1992). *KADS: a modelling approach to knowledge engineering*. In Knowledge Acquisition (1992) 4(1).

Wiil U.K. & Legget J.J. (1992). *Hyperform: Using Extensibility to Developp Dynamic, Open and Distributed Systems.* In ECHT (1992).

Wille R. & Lehmann F. (1994). *Triadic Concept Analysis*. In Proc. of the Int. Conf. on Ordinal and Symbolic Data Analysis in Paris-INRIA, juin 1994.

Willems M. (1995). *Projection and Unification for Conceptual Graphs.* In ICCS (1995).

Wuwongse V. & Manzano M. (1993). *Fuzzy conceptual graphs*. In ICCS (1993).

Yankelovich N., Haan B.J., Meyrowitz N. & Drucker S.M. (1988). *Intermedia: the Concept and the Construction of a Seamless Information Environment*. In "IEEE Computer", Vol. 21, No 1, 1988.

Zellweger P. (1989). *Scripted Documents: A Hypermedia Path Mechanism.* In HTX (1989).

Abréviations

AC : Acquisition de connaissances
BC : Base de Connaissances
ED : Elément de Document
ED GC : Elément de document de type GC (i.e. stockant un GC)
GC : Graphe Conceptuel
GCS : Graphe Conceptuel Simple
RI : Recherche d'informations
SBC : Système à Base de Connaissances

Table des matières

Chapitre 1 Introduction 7
1.1 Acquisition de connaissances et recherche d'informations 7
1.1.1 Informations et connaissances 7
1.1.2 Représenter des connaissances pour la recherche d'information 8
1.1.3 Rechercher des informations pour l'acquisition de connaissances 9
1.2 Objectifs et approche 11
1.2.1 Combinaison de gestion d'informations et de gestion de connaissances 11
1.2.1.1 Choix du langage de représentation des connaissances 13
1.2.2 Une ontologie pour guider l'AC et la RI 13
1.2.3 Plan du rapport 16

PARTIE I : ETAT DE L'ART

Chapitre 2 L'acquisition de connaissances 19
2.1 Introduction 19
2.2 Les grands types d'activités et de connaissances 21
2.2.1 Les grands types de connaissances utilisés en modélisation 23
2.2.1.1 Des ontologies pour les différents types de connaissances 24
2.2.1.2 Les modèles génériques 24
2.2.2 Les activités en acquisition des connaissances 25
2.2.3 Extraction et modélisation ascendantes et descendantes 28
2.2.3.1 L'extraction automatique de connaissances 29
2.2.3.2 Exemples de modèles de tâches génériques pour l'approche descendante 29
2.3 La représentation et l'organisation des connaissances 33
2.3.1 Généricité du langage de représentation de connaissances 33
2.3.2 Encapsulation, vues et modules 34
2.3.2.1 Encapsulation 34
2.3.2.2 Vues, points de vue et modules 35
2.3.2.2.1 Comparaison des vues et des modules 37
2.3.2.2.2 Usages habituels des vues et points de vue 39
2.4 La création et la réutilisation d'ontologies 40
2.4.1 Quelques principes pour construire une ontologie partageable ou réutilisable 42
2.4.1.1 Expliciter les engagements ontologiques 42
2.4.1.2 Créer une ontologie de termes primitifs puis une ontologie formelle 44
2.4.1.2.1 Ontologie de termes primitifs versus ontologie "formelle" 44
2.4.1.2.2 La construction d'une ontologie de termes primitifs 44
2.5 Conclusion 46

Chapitre 3 Structuration, recherche et présentation d'informations 47
3.1 Les différents types de structuration et de recherche 48
3.1.1 Les différents types de recherche 48
3.1.1.1 Recherches séquentielles ou indexées 48
3.1.1.2 Liens statiques ou virtuels, navigation ou requêtes 49
3.1.1.3 Recherches exactes ou approchées 50
3.1.1.3.1 Les critères de précision et de rappel pour la recherche approchée 50
3.1.2 Différentes méthodes de recherche 51
3.1.2.1 L'aide à la navigation 51
3.1.2.1.1 Visualisation partielle de réseaux et/ou d'éléments de documents 51
3.1.2.1.2 Les visites guidées 51
3.1.2.1.3 Restriction de l'espace de recherche 51
3.1.2.2 Les requêtes sur les attributs 52
3.1.2.3 Les requêtes sur la structure d'un réseau (noeuds et liens) 52
3.1.2.3.1 L'interrogation par Datalog 52
3.1.2.3.2 Utilisation d'une logique modale 53
3.1.2.3.3 Le langage graphique GraphLog 53
3.1.2.3.4 Le système MORE 54
3.1.2.3.5 Le modèle GRAM 54
3.1.2.3.6 WebTalk, le langage de script de MacWeb 55
3.1.2.4 La recherche approchée d'éléments via l'indexation de leur contenu 56
3.2 Les systèmes de structuration d'informations 58
3.2.1 Les documents structurés 59
3.2.1.1 Avantages 59
3.2.1.2 Inclusion d'un élément de document 59
3.2.1.3 Les normes documentaires 59
3.2.2 Les systèmes hypertextes 60
3.2.2.1 Les normes hypertextes 60
3.2.2.2 Les modèles typés 61
3.2.2.2.1 MacWeb : une intégration de différentes méthodes d'indexation et de recherche 62
3.2.3 L'éditeur de document structuré Thot 63
3.2.3.1 Un éditeur syntaxique et des langages 63
3.2.3.2 La structuration et la recherche d'informations via l'éditeur Thot 65
3.2.3.3 Une interface fonctionnelle permettant la création de documents actifs 66
3.3 Conclusion 67
3.3.1 Bilan sur les techniques de recherche et de gestion d'information 67
3.3.2 Combinaison de techniques de recherche et de gestion d'information 67
3.3.3 Recherche d'informations et explications 68

Chapitre 4 Les Graphes Conceptuels 71
4.1 Introduction 72
4.2 Le modèle de base des GCs 73
4.2.1 Présentation informelle 73
4.2.2 Présentation formelle 75
4.2.2.1 Support et graphes conceptuels simples 75
4.2.2.2 Morphismes et projections 76
4.2.2.3 Les opérations de spécialisation et de généralisation 76
4.2.2.3.1 Les opérations élémentaires 76
4.2.2.3.2 Des opérations de spécialisation plus complexes 78
4.2.2.4 Lambda-abstractions et définitions de types 79
4.2.2.4.1 Définition de types de concepts 79
4.2.2.4.2 Définition de types de relations 80
4.2.2.4.3 Association de schémas prototypiques à un type de concept 80
4.2.2.4.4 Individus et prototypes 81
4.3 Extensions au modèle de base 82
4.3.1 D'autres types de référents 82
4.3.2 Les référents graphes 83
4.3.2.1 Notion de contextualisation 83
4.3.2.2 Interprétation et gestion des contextes comme des définitions de type 85
4.3.2.3 Interprétation et gestion des contextes comme des modules 85
4.3.2.3.1 Un contexte "module" permettant de définir des types locaux à ce module 85
4.3.2.3.2 Gestion de la négation de contextes avec les "règles d'inférence propositionnelles" 86
4.3.2.3.3 Gestion "orientée objet" des contextes 86
4.3.2.3.4 Gestion des contextes par la projection 87
4.3.2.3.5 Gestion de contexte ad-hoc pour prendre en compte certains types de contextualisation 87
4.3.2.4 Une ontologie pour les types de contextes modules 88
4.3.2.5 L'approche retenue dans CGKAT 90
4.3.2.5.1 Principes 90
4.3.2.5.2 Implémentation des GCs emboîtés dans CGKAT 93
4.3.2.5.3 Extension aux définitions de types 93
4.3.3 Les référents ensemblistes 94
4.3.4 Les types d'ordre supérieur 95
4.3.5 Les acteurs 95
4.4 Applications des GCs 96
4.4.1 Analyse de la langue naturelle 96
4.4.2 Structuration de connaissances 97
4.4.2.1 Comparaison avec KL-ONE 98
4.4.3 Acquisition de connaissances 98
4.5 Environnements de travail pour manipuler des GCs 100
4.5.1 Des interfaces graphiques 101
4.5.1.1 Affichage automatique de graphes 101
4.6 Conclusion 102

PARTIE II : LE TRAVAIL RÉALISÉ

Chapitre 5 Des ontologies pour guider la construction d'une base de connaissances 105
5.1 Introduction 106
5.2 Accès, visualisation et construction de taxinomies 107
5.2.1 Présentation des termes 107
5.2.2 Accès aux types et construction de vues dans une hiérarchie de types 113
5.2.2.1 Construction de vues en utilisant les relations de spécialisation entre les types 113
5.2.2.2 Construction de vues en utilisant les relations de spécialisation entre des méta-informations 113
5.2.2.2.1 Définition actuelle de la spécialisation entre méta-informations 114
5.2.2.2.2 Extension de la définition actuelle de la spécialisation entre méta-informations. 115
5.2.2.3 Construction de vues en utilisant les définitions de types et la projection 115
5.2.2.4 Construction d'une même taxinomie de types par plusieurs utilisateurs 116
5.2.3 Gestion d'autres relations que la relation "sorte-de" 117
5.3 Exploitation de la base générale de connaissances terminologiques WordNet 118
5.3.1 Présentation de WordNet 118
5.3.1.1 Principes de WordNet 118
5.3.1.2 Exploitation de WordNet par des programmes 120
5.3.1.3 Discussion 120
5.3.2 Inclusion dynamique de WordNet dans le treillis des types de concept 120
5.3.2.1 Méthode 120
5.3.2.2 Remarques 121
5.3.3 Intérêt de WordNet pour l'acquisition des connaissances 124
5.4 Une ontologie de haut niveau pour les types de concepts 126
5.4.1 Les distinctions élémentaires 126
5.4.2 Les entités 128
5.4.2.1 Les collections 130
5.4.2.2 Les entités temporelles 131
5.4.2.3 Les entités spatiales 131
5.4.2.4 Les entités abstraites (moyens de représentation, propositions, propriétés, mesures) 132
5.4.2.4.1 Les entités de représentation 132
5.4.2.4.2 Quelques définitions pour les types de modèles préconisés par KADS_I 133
5.4.2.4.3 Les propositions 134
5.4.2.4.4 Les propriétés et les mesures 136
5.4.3 Les situations 138
5.4.3.1 Définitions et remarques sur les processus, les états et les relations 138
5.4.3.2 Les états 139
5.4.3.3 Les processus 140
5.4.4 Les choses jouant des rôles 141
5.4.4.1 Les entités jouant un rôle 142
5.4.4.2 Les situations jouant un rôle 142
5.4.5 Les concepts utilisés par des agents 143
5.4.6 Bilan de la réorganisation des catégories de haut niveau de WordNet 143

5.5 Les tâches de résolution de problèmes et les modèles de tâches génériques 144
5.6 Une ontologie de haut niveau pour les types de relations 149
5.6.1 Intérêt d'une ontologie structurée de types de relations 149
5.6.2 Classifications de types de relations primitives 150
5.6.3 Les relations conceptuelles qui représentent des processus 151
5.6.4 Les relations non binaires 152
5.6.5 Des catégories de haut niveau pour les types de relations 155
5.6.5.1 Les relations à partir d'un concept de type Situation 156
5.6.5.2 Les relations à partir d'un concept de type Proposition 157
5.6.5.3 Les relations ayant une propriété spéciale 159
5.6.5.4 Les relations contextualisantes à partir d'un concept de type Proposition 159
5.6.6 Nécessité de choix supplémentaires 161
5.7 Exploitation de définitions de types pour guider l'extraction et la représentation de connaissances 163
5.7.1 Des listes de relations connectables aux concepts de certains types 163
5.7.1.1 Extraction ascendante de connaissances 165
5.7.1.2 Recherche des définitions de types adéquates 166
5.7.2 Des définitions de types pour guider la modélisation de connaissances 167

Chapitre 6 Des documents structurés pour rechercher et organiser des informations *et* des connaissances 169
6.1 Introduction 170
6.2 Construction d'une base de connaissances via un éditeur de documents structurés 172
6.2.1 Motivations 172
6.2.2 Construction d'un graphe conceptuel via un élément de document 172
6.2.2.1 Principe général d'exploitation d'un modèle structurel par Thot 174
6.2.2.2 Détails sur la création des EDs GC et de leurs composants 175
6.2.2.3 Mise à jour de la base de CoGITo 175
6.2.2.4 Génération des noms de graphes 176
6.2.2.5 Création des définitions de types 176
6.2.2.6 Gestion des inclusions 179
6.2.2.7 Chargement des graphes lors de l'ouverture des documents 179
6.2.3 Association de méta-informations à un graphe conceptuel 179
6.2.4 Construction d'une hiérarchie via un élément de document 181
6.2.5 Utilisation d'un langage de commandes via un élément de document 183
6.2.5.1 Le langage de commandes de CGKAT 185
6.2.6 Comparaison avec d'autres travaux 186

6.3 Indexation d'éléments de documents par des connaissances 187
6.3.1 Indexation des éléments de documents dans le formalisme des GCs 188
6.3.1.1 Différents types d'indexation 188
6.3.1.1.1 Premières restrictions sur les représentations d'EDs 188
6.3.1.2 Un modèle structurel d'extension pour permettre l'indexation par des GCs 189
6.3.1.2.1 Utilité 189
6.3.1.2.2 Contenu 189
6.3.1.3 Ontologie pour la représentation d'éléments de documents 197
6.3.1.3.1 Les différentes notions qui peuvent être représentées 197
6.3.1.3.2 Représentation des "EDs symboles" et des "EDs descriptions" 198
6.3.1.4 Remarques supplémentaires sur la représentation d'EDs 199
6.3.1.4.1 Marques de représentation 199
6.3.1.4.2 Représentation selon un point de vue 199
6.3.1.4.3 Perspectives : génération de représentations d'EDs 200
6.3.1.4.3.1 Génération de représentations d'EDs à partir de représentations de sous-EDs 200
6.3.1.4.3.2 Génération de représentations d'EDs à partir de parties de graphes 200
6.3.1.4.4 Génération d'index synthétisant des représentations d'EDs 201
6.3.2 Recherche d'informations ou de connaissances dans CGKAT 205
6.3.2.1 Navigation 205
6.3.2.1.1 Navigation entre représentations de connaissances 205
6.3.2.2 Requêtes lexicales, structurelles et conceptuelles 206
6.3.2.2.1 Utilisation de sous-chaînes de noms de types dans les GCs requêtes 207
6.3.2.2.1.1 Perspective : exploitation de liens hypertextes entre des mots et des types 207
6.3.2.2.2 Présentation des résultats : GCs et/ou informations indexées par ces GCs 208
6.3.2.2.2.1 Perspectives : présentation des résultats sous une forme hiérarchique 214
6.3.2.2.2.2 Perspectives : génération d'EDs par intégration de commandes d'édition de documents à notre langage de commandes 214
6.3.2.2.3 Présentation de graphes emboîtés 215
6.3.2.2.4 Recherche de généralisations et tests de généralisation 218
6.3.2.2.5 Perspective : recherche de concepts individuels via la recherche de spécialisations 218
6.3.2.2.6 Perspective : recherche de sous-graphes et de séquences 218
6.3.2.2.6.1 Recherches de sous-graphes dans un seul graphe 219
6.3.2.2.6.2 Recherche de sous-graphes dans plusieurs graphes 221
6.3.2.2.7 Génération d'explications 222
6.3.2.2.8 Liens virtuels 225
6.3.3 Comparaison avec d'autres systèmes de recherche d'informations ou d'acquisition de connaissances 226
6.3.3.1 Comparaison avec les systèmes hypertextes 226
6.3.3.2 Comparaison avec les outils d'acquisition des connaissances 229
6.4 Eléments méthodologiques d'utilisation de CGKAT 230
6.4.1 Extraction et modélisation ascendante avec CGKAT 232
6.4.2 Extraction et modélisation descendante avec CGKAT 233
6.4.3 Utilisation de CGKAT en combinaison avec d'autres outils 235
6.4.3.1 Une expérience : échanges de données entre CoKaCe et CGKAT 235

Chapitre 7 Implémentation 237
7.1 Architecture de CGKAT : une approche ouverte 238
7.2 Exploitation de CoGITo et de WordNet 239
7.2.1 L'interface fonctionnelle de CGKAT pour la manipulation d'une base de Graphes Conceptuels 239
7.2.2 Gestion de types 240
7.2.2.1 Définitions de types de concepts ou de types de relations 240
7.2.2.2 Liens entre types de concepts ou entre types de relations 240
7.2.2.3 Recherche de types ou de marqueurs individuels et inclusion dynamique de types de WordNet 241
7.2.2.4 Destruction de types 242
7.2.3 Gestion de graphes 242
7.2.3.1 Indexation 242
7.2.3.2 Graphes emboîtés 242
7.2.3.3 Requêtes conceptuelles 243
7.2.3.4 Jointure maximale 244
7.3 Exploitation de Thot 244
7.3.1 Usage de l'interface fonctionnelle de Thot pour la gestion de structures logiques 245
7.3.2 Quelques choix effectués 247
7.3.2.1 Construction de modèles structurels 247
7.3.2.1.1 Premier exemple : structure d'un ED TypeDefinition 248
7.3.2.1.2 Second exemple : structure d'un ED Relation 249
7.3.2.2 Application des modèles d'interface 250
7.3.2.3 Sauvegarde des documents et des EDs GC dans les documents 250
7.4 Le langage de commandes de CGKAT 251
7.4.1 Construction ou référence à des GCs dans les requêtes 251
7.4.2 Exploitation du shell dans le langage de commandes de CGKAT 251
7.5 Conclusion 252

Chapitre 8 Conclusion 253
8.1 Fonctionnalités et guides 253
8.1.1 Fonctionnalités de CGKAT 254
8.2 Comparaison avec d'autres outils d'AC ou de RI 256
8.2.1 Comparaison par rapport aux systèmes hypertextes 256
8.2.2 Comparaison par rapport aux outils d'AC 258
8.3 Perspectives 260

Chapitre 9 Références bibliographiques 261

Chapitre 10 Abréviations 275

Chapitre 11 Table des matières 277

Chapitre 12 Annexes 285

ANNEXES

Annexe 1 Les menus de CGKAT 287

Annexe 2 Protocoles pour faciliter la construction coopérative et incrémentale d'une hiérarchie de types par plusieurs utilisateurs 295

Annexe 3 Quelques ontologies de haut niveau 301

Annexe 4 Modèles structurels, de présentation et d'interface 317

Annexe 5 Le langage de commande de CGKAT 337

Annexe 6 L'interface fonctionnelle de CGKAT pour la manipulation d'une base de Graphes Conceptuels 339

Annexe 7 L'interface fonctionnelle pour manipuler Thot 343

Annexe 8 Modularisation des fichiers de sauvegarde 345

Annexe 1 Les menus de CGKAT

1 Sommaire

1.1 Menus généraux 287
1.2 Menus d'affichage et de gestion de types 288
1.3 Menus d'affichage et de gestion des marqueurs individuels 292

1.1 Menus généraux

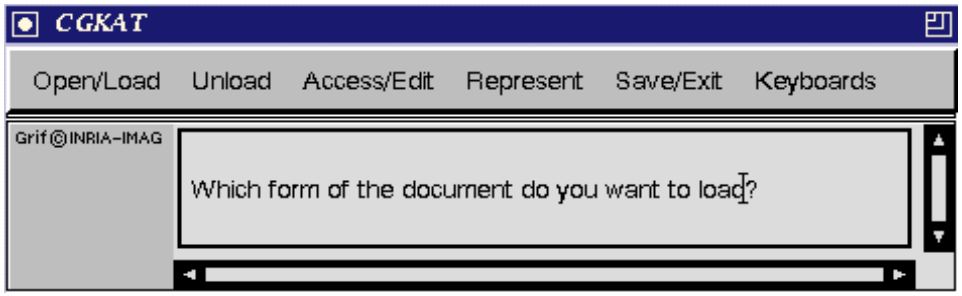


Figure A.1.1: *Le menu d'entrée de CGKAT (la fenêtre centrale permet d'afficher des messages).*

Open the default document
Open document...
Create an Article document
Create a CGReprList document
Create a CG document
Create a document by copying the selected element
Load a BCGCT file
Load a file which contains a CG in linear format
Load supertypes of uncategorized WordNet concept types
Start a linear interpretor to access or update the KB

Figure A.1.2: *Les entrées du menu "Open/Load".*

Unload the CGs in the selected element
Unload CGs according to their nature or their comment
Unload all CGs
Unload all CGs and types

Figure A.1.3: *Les entrées du menu "Unload".*

Edit and use the concept type hierarchy...
Edit and use the relation type hierarchy...
Edit and use the instance hierarchy...
Lexical search in the selected element...

Figure A.1.4: *Les entrées du menu "Access/Edit".*

Represent the selected element with a CG (or a type definition)
Index the selected element with a single concept of type Proposition
Index the selected element with a single concept via the concept type hierarchy menu...
Associate a CG note to the selected element

Figure A.1.5: *Les entrées du menu "Represent".*

Save concepts types and relations type in a BCGCT file
Save the whole environnement in a BCGCT file
Exit

Figure A.1.6: *Les entrées du menu "Save/Exit".*

1.2 Menus d'affichage et de gestion de types

Les figures A.1.9 et A.1.10 ci-après montrent respectivement le menu de gestion des types de concepts et le menu de gestion des types de relations. Voici leurs sous-menus.

Search in the hierarchy
Search in the WordNet ontology
Reset the search filter with the next string

Figure A.1.7: *Les entrées du menu "Search for concept types including the next string" (ces diverses options n'existent pas dans le menu de gestion des types de relations : la recherche ne peut s'effectuer que dans la hiérarchie de types de relations).*

Keep this temporary included WordNet type and its supertypes in the lattice	
Add a new subtype to it	(the new subtype must have been put in the 'String for a concept type' text form)
Add a comment to it	(the comment must have been put in the 'String for a concept type' text form)
Add it as a subtype of an existing concept type	(now select this type and put a space in the 'String for a concept type' text form)
Move it and its subtypes under another supertype	(now select this type and put a space in the 'String for a concept type' text form)
Delete it (its subtypes will belong to its supertype)	
Delete it and its subtypes	
List its subtypes which have many direct supertypes	
Say if it is a subtype of the next concept type	(the type must have been put in the 'String for a concept type' text form)
Save the whole ontology in a BCGCT file	

Figure A.1.8: *Les entrées du menu "Commands on concept types using the below selected concept type" (un sous-menu similaire existe dans le menu de gestion des types de relations).*

List relation types without signature
List relation types with their signatures
List the types of the possible relations from a concept of the selected concept type
List the types of the possible relations from the selected concept

Figure A.1.9: *Les entrées du menu "Other kinds of relation type list" dans le menu de gestion des types de relation (une des entrées nécessite qu'un type soit sélectionné dans le menu de gestion des types de concepts, et une autre nécessite qu'un ED Concept ou ConceptInclusion soit sélectionné dans les documents ouverts).*

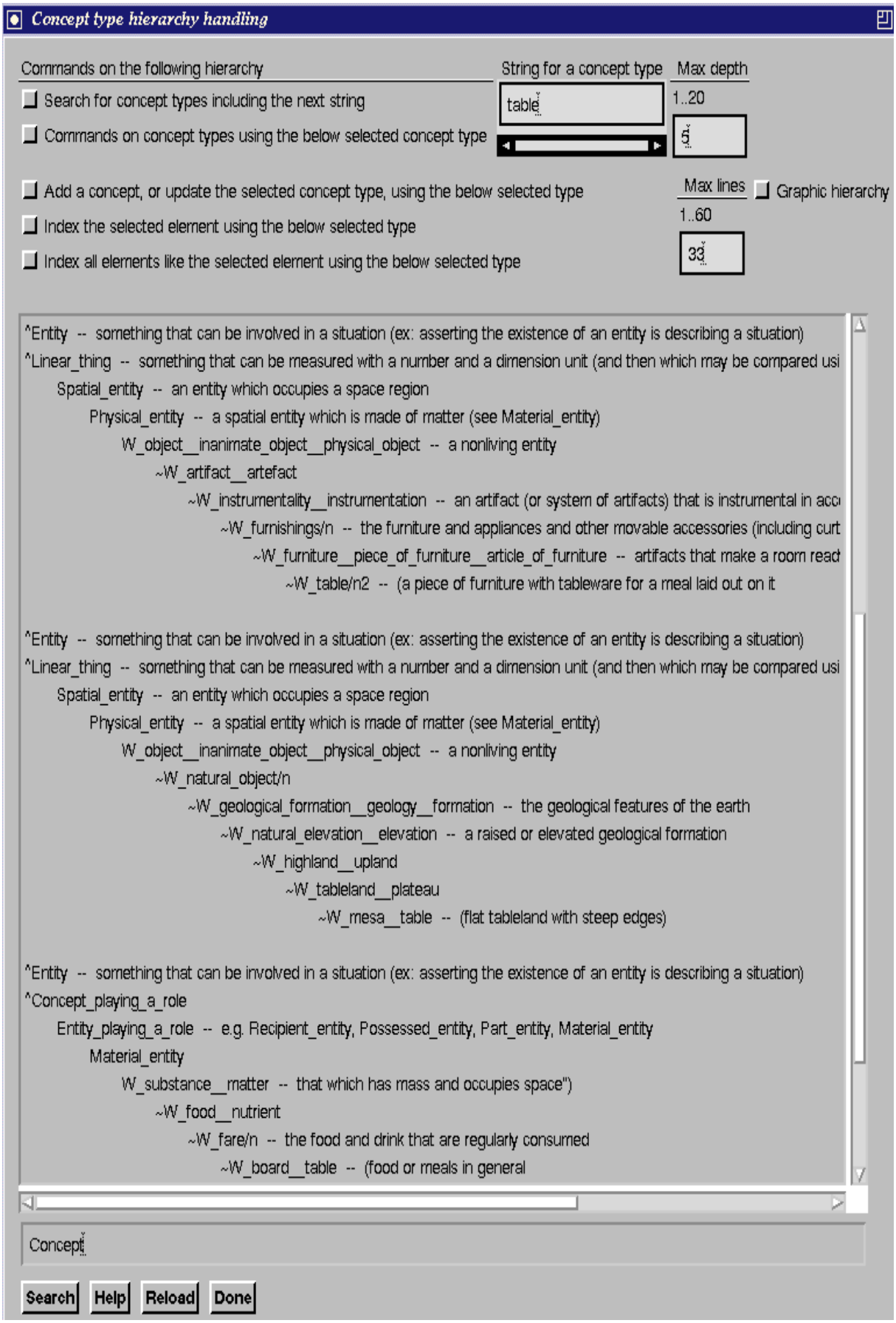


Figure A.1.10: *Le menu de gestion des types de concept montrant des supertypes des types de WordNet relatifs aux sens du mot "table".*

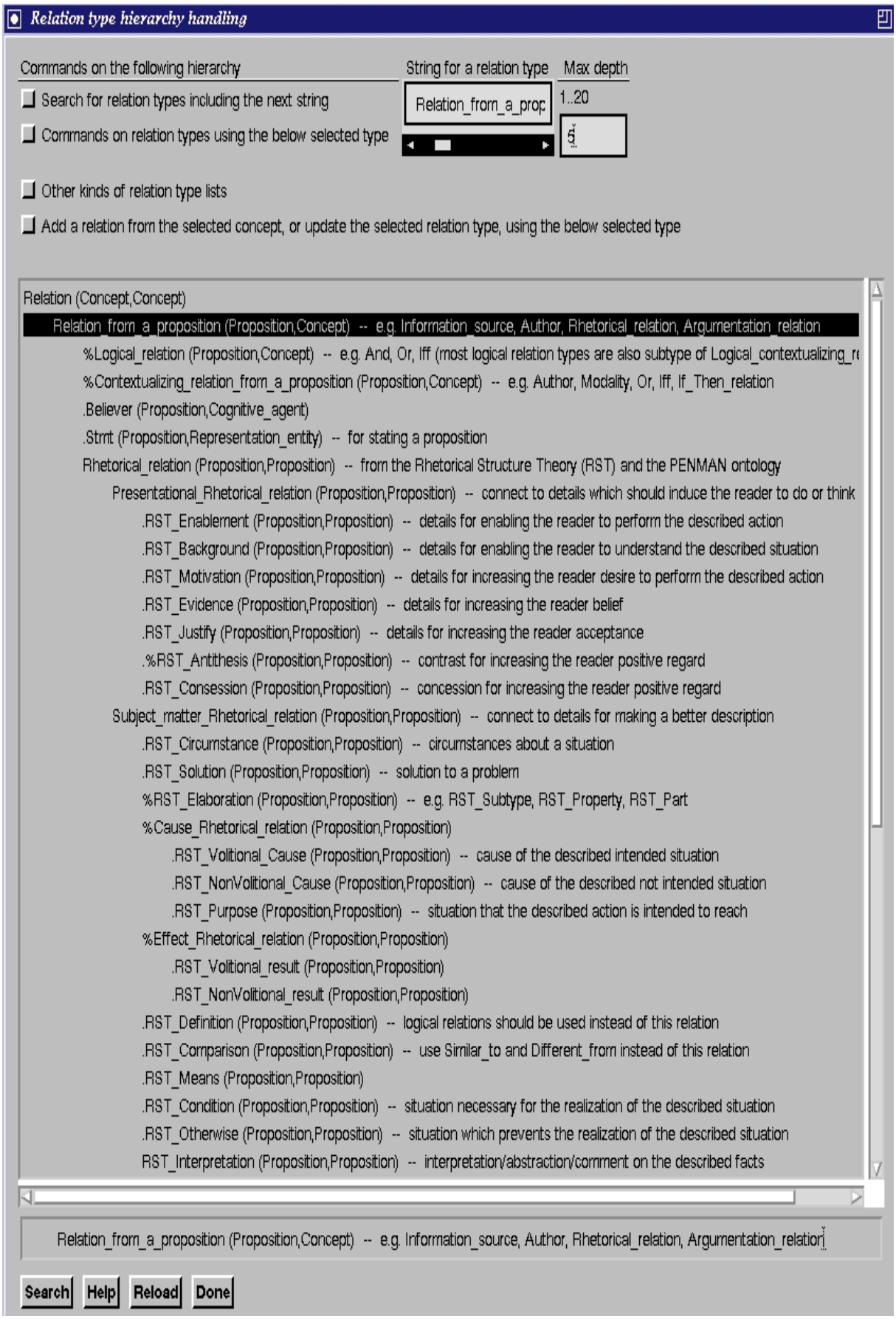


Figure A.1.12: *Le menu de gestion des types de relations montrant le supertype et les premiers sous-types de Relation_from_a_proposition dans l'ontologie proposée par défaut par CGKAT.*

Lorsqu'un type de concept est sélectionné dans le menu de gestion des types de concept,

1) Ses supertypes sont *affichés* (maximum 9) ainsi que ses sous-types (maximum fixé par l'entrée "Max depth" et contrôlé par la présence du mot-clé "e.g." dans les commentaires des types). Cela permet par exemple les recherches par navigation (i.e. la sélection successive de types).
2) Des commandes peuvent être exécutées sur la hiérarchie des types, par rapport au type sélectionné (cf. figure A.1.9).
3) Si de plus un ED est sélectionné dans un document,
   si cet ED est un ED ConceptType, il peut être mis à jour avec la commande "*Add a concept, or update the selected concept type, using the below selected type*",
   sinon si cet ED est un ED Concept, ConceptInclusion ou CGgraph, la même commande permet d'ajouter un nouvel ED Concept au graphe sélectionné ou contenant l'ED sélectionné (ce nouveau concept a pour type celui sélectionné et pour référent un référent générique) ;
   sinon, l'ED sélectionné peut être indexé (représenté) par un concept générique ayant pour type celui sélectionné (plus exactement, afin de permettre de multiples indexations sur l'ED sélectionné, celui-ci est indexé par une liste d'EDs CGRepr dont un ED CGRepr contient le concept générique et certaines méta-informations) ; la commande "*Index the selected element using the below selected type*" peut être utilisée pour cela ;
   la commande "*Index all elements like the selected element using the below selected type*" peut également être utilisée pour indexer de manière identique tous les EDs similaires à l'ED sélectionné (si cet ED est un mot ou un groupe de mot, les EDs similaires sont les autres occurences de ce mot ou de ce groupe de mots dans le document, sinon les EDs similaires sont les EDs de même type dans le document).
   Notre système d'indexation a été détaillé dans la section 6.3.1.2.

Lorsqu'un type de relation est sélectionné dans le menu de gestion des types de relation,

1) Ses supertypes sont affichés (maximum 9) ainsi que ses sous-types (maximum fixé par l'entrée "Max depth" et contrôlé par la présence du mot-clé "e.g." dans les commentaires des types). Cela permet par exemple les recherches par navigation (i.e. la sélection successives de types). différents types d'affichage peuvent être choisis via la commande "*Other kinds of relation type lists*" (cf. figure A.1.10) ;
2) Des commandes peuvent être exécutées sur la hiérarchie des types par rapport au type sélectionné.
3) Si de plus un ED est sélectionné dans un document,
   si cet ED est un ED RelationType, il peut être mis à jour avec la commande "*Add a relation from the selected concept, or update the selected relation type, using the below selected type*",
   sinon si cet ED est un ED Concept ou ConceptInclusion, la même commande permet de relier à ce concept une nouvelle relation de type celui sélectionné ; la nouvelle relation est également connectée à un nouveau concept dont le référent est générique et dont le type est déterminé à partir de la signature de la relation (la relation est supposée binaire).

1.3 Menu d’affichage et de gestion des marqueurs individuels

Le menu de gestion de marqueurs individuels a été construit suivant une philosophie similaire à celle des menus de gestion des types de concepts ou de relation. Ainsi, si CoGITo était étendu pour permettre de manipuler des types d’ordre supérieur à 1, il serait aisé d’étendre le code actuel de CGKAT pour permettre d’afficher et de gérer la hiérarchie des types/marqueurs individuels (hiérarchie suivant la relation de conformité). Par abus de langage et pour des raisons de simplicité nous employons dans ce menu le terme "d’instance" au lieu de référent individuel.

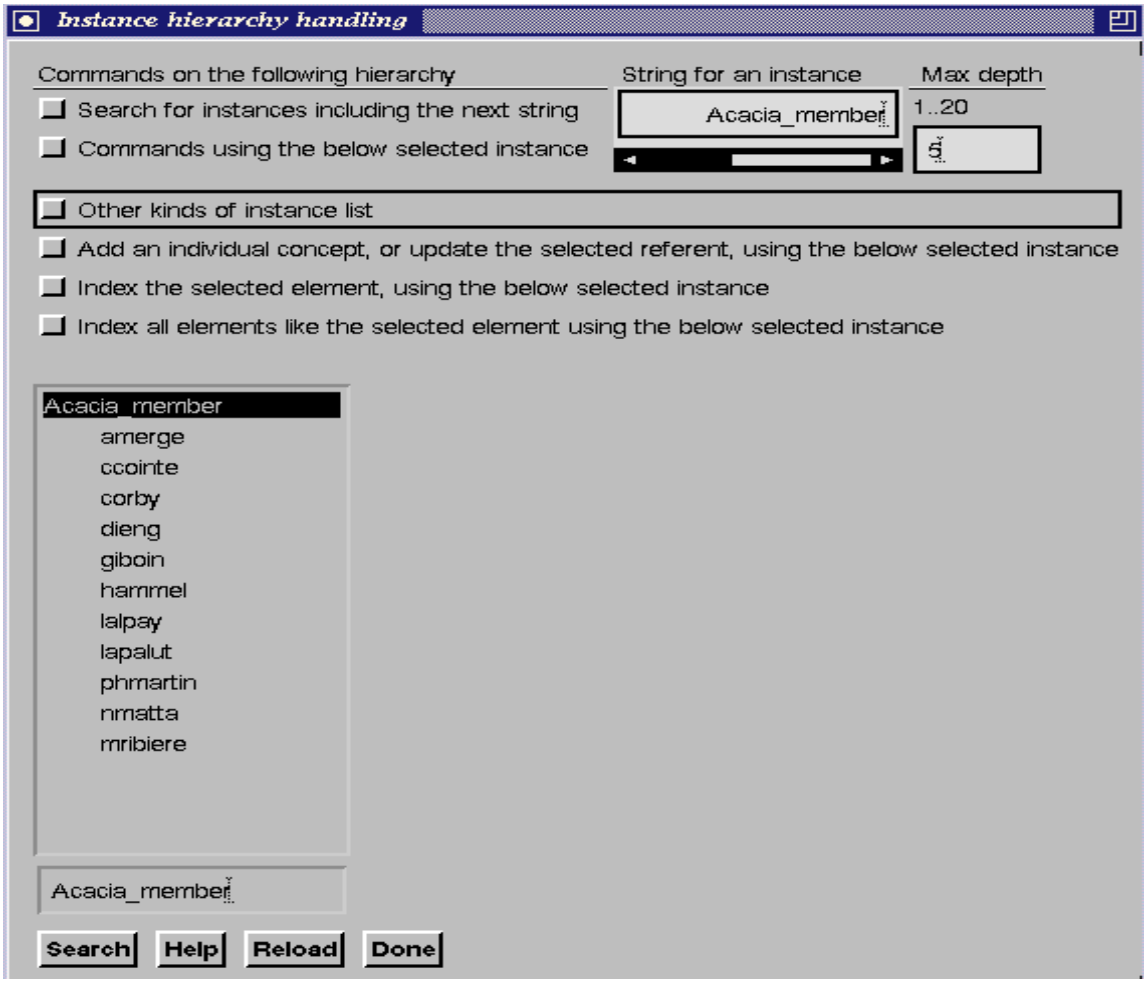


Figure A.1.13: *Le menu de gestion des marqueurs individuels montrant ceux conformes au type de concept "Acacia_member" (membre de l’équipe Acacia de l’INRIA Sophia Antipolis).*

Lorsqu’un type est sélectionné dans ce menu, les marqueurs individuels qui lui sont conformes sont affichés. Le type peut être choisi via le sous-menu suivant.

List the instances of the concept type selected in the 'Concept type hierarchy handling' menu
List the instances of the selected concept or concept type in a document

Figure A.1.14: *Les entrées du sous-menu "Other kinds of instance list".*

Un marqueur individuel peut également être recherché avec une sous-chaine (comme les types dans les menus de gestion des types).

Lorsqu'un marqueur individuel est sélectionné dans ce menu,
1) des commandes peuvent être exécutées par rapport au marqueur sélectionné ;
2) si de plus un ED est sélectionné dans un document,
si cet ED est un ED Individual, il peut être mis à jour avec la commande "*Add an individual concept, or update the selected referent, using the below selected instance*",
sinon si cet ED est un ED Concept, ConceptInclusion ou CGgraph, la même commande permet d'ajouter un nouvel ED Concept au graphe sélectionné ou contenant l'ED sélectionné (ce nouveau concept a pour référent le marqueur sélectionné et pour type le type auquel est conforme ce marqueur) ;
sinon, l'ED sélectionné peut être indexé (représenté) par un concept indivividuel ayant pour référent le marqueur sélectionné et pour type le type auquel est conforme ce marqueur (plus exactement, l'ED sélectionné est indexé par une liste d'EDs CGRepr dont un ED CGRepr contient le concept individuel et certaines méta-informations) ; la commande "*Index the selected element using the below selected instance*" peut être utilisée pour cela ;
la commande "*Index all elements like the selected element using the below selected instance*" peut également être utilisée pour indexer de manière identique tous les EDs similaires à l'ED sélectionné.

Annexe 2 Protocoles pour faciliter la construction coopérative et incrémentale d'une hiérarchie de types par plusieurs utilisateurs

Il semble plus rapide pour des cogniticiens travaillant à représenter une même expertise, de construire de manière coopérative et incrémentale une même hiérarchie de types plutôt que de travailler chacun de leur côté, "en aveugle" puis de fondre leurs hiérarchies en une seule. En effet, dans le premier cas, chaque cogniticien peut réutiliser les types d'autres utilisateurs et positionner chacun de ses types, dès leur création et de manière directe ou indirecte, par rapport aux types d'autres utilisateurs. Il complète donc ces types des autres utilisateurs, ou les spécialise ou encore les généralise. L'introduction d'une incohérence dans la hiérarchie des types peut ainsi être détectée et corrigée au plus tôt, par le système (e.g. la création d'un cycle ou d'un sous-type commun à des types exclusifs) ou par un cogniticien (e.g. un cogniticien averti des spécialisations ou généralisations effectuées par d'autres cogniticiens sur des types qu'il a créés, peut remarquer une mauvaise interprétation du sens qu'il accordait à ces types, et modifier la hiérarchie pour corriger ces erreurs).
Un type représente des objets ayant certaines caractéristiques. Si deux cogniticiens se rendent compte qu'ils n'accordent pas exactement les mêmes cararactéristiques aux objets représentés par un type, ces cogniticiens font en réalité référence à deux notions distinctes et deux types différents doivent être créés pour représenter ces notions (ces deux types peuvent alors avoir des noms utilisateurs identiques, i.e. des alias identiques, mais auront des noms système différents). Il est intéressant que ces deux types soit interclassés avec différents liens, e.g. un lien de spécialisation ou un lien d'exclusion.

Pour faciliter la construction coopérative et incrémentale d'une hiérarchie de types par plusieurs utilisateurs, en *minimisant* les conflits et la nécessité pour les utilisateurs de se rencontrer fréquemment afin de négocier le choix des types, de leurs noms, de leurs définitions et de leur organisation, nous proposons le protocole suivant :

- Un utilisateur peut *mettre des alia*s sur les noms des types créés par d'autres utilisateurs et visualiser et manipuler ces types sous ces alias (ce qui permet un certain confort dans la réutilisation des types et évite ou réduit les conflits de noms de type entre utilisateurs). Par défaut, pour un utilisateur, un type créé par un autre utilisateurs doit pouvoir être visualisé et manipulé avec le nom donné par son créateur concaténé avec le nom identificateur de ce créateur[1].

- Un utilisateur peut poser des commentaires privés (i.e. visibles de lui-seul) sur des types créés par d'autres utilisateurs.

- Un utilisateur peut *ne visualiser que certains types* de la hiérarchie, e.g. les types créés par certains utilisateurs, les sous-types de certains types. Nous utilisons ici le terme "certains" à chaque fois qu'un sous-ensemble de choses est spécifié par un utilisateur, manuellement ou via une requête. Le système de filtrage peut être similaire à celui que nous avons implémenté dans les menus de gestion de taxinomies de CGKAT (cf. section 5.2.2.1). Les types pourraient également être colorés de manière différente suivant les utilisateurs qui les ont créés.

1. Il s'agit d'une visualisation en mode "utilisateur" (cf section 5.2.1). Les noms "système" sont inconnus des utilisateurs. Un utilisateur ne peut donner le même nom ou alias à deux types différents.

- Un utilisateur peut *ajouter des liens entre des types*[1] *ou des définitions à des types*[2] (si le système ne détecte pas d'incohérence ou s'il peut la réparer par le système comme nous le verrons plus loin). *Cet ajout est possible **même si** ces types ont été créés par d'autres utilisateurs.*
  Un utilisateur peut *détruire des liens entre types ou modifier des définitions **à condition** qu'il en soit le créateur.* Les points suivants sont destinés à permettre/faciliter la gestion de ces ajouts, modifications et destructions lorsque la hiérarchie est construite/partagée par de multiples utilisateurs).
  Ainsi, un utilisateur peut ajouter des informations sur un type qu'il n'a pas créé, à condition que le système ne détecte pas d'incohérence entre ces nouvelles informations et celles déjà existantes sur ce type (quels que soient les utilisateurs les ayant créées). Mais il ne peut détruire ou modifier des informations sur ce type s'il ne les a pas créées.
  Lorsqu'un lien redondant est ajouté, e.g. un lien de spécialisation entre un type et un de ses sous-types, ce lien n'est pas enregistré (même si les types reliés ont été créés par différents utilisateurs). Si l'ajout d'un lien entraîne la redondance d'un autre lien, ce dernier est supprimé (même si les types reliés ont été créés par différents utilisateurs) ;

- Un utilisateur peut demander au système à être averti de certains types de modifications, *effectuées par d'autres utilisateurs sur certains types*. Par modification de type, nous désignons tout ajout/destruction de lien sur un type et tout ajout/destruction/modification de définition d'un type. Il semble intéressant que le créateur d'un type modifié soit par défaut averti par le système de cette modification à moins qu'il n'ait exprimé le désir contraire. De même, le créateur d'un sous-type (direct ou indirect) d'un type modifié peut par défaut être averti si ce créateur n'est pas l'auteur de la modification et s'il ne s'est pas opposé à de tels avertissements.

- Un utilisateur peut demander à être *enregistré comme co-créateur de certains liens et de certains types et de leurs supertypes*, afin d'exprimer son accord et son intérêt pour ces types, leurs définitions et leur organisation *au moment de cet enregistrement* (il exprime ainsi qu'il aurait pu créer lui-même cette partie de la hiérarchie). Il semble intéressant que le co-créateur d'un type modifié soit par défaut averti de cette modification à moins qu'il n'ait exprimé le désir contraire. Un utilisateur doit pouvoir sélectionner un sous-ensemble de types sur le fait qu'ils ont été co-créés par certains utilisateurs, par exemple pour signaler qu'il désire ou non visualiser ces types.
  Un utilisateur qui met un alias sur un type d'un autre utilisateur, ou l'utilise dans un GC, devient automatiquement co-créateur de ce type. De même, un utilisateur qui ajoute un lien entre des types devient co-créateur de ces types, ainsi que de leurs supertypes au moment de cet ajout, à condition qu'il ne soit pas déjà créateur de ces types et de ces supertypes ;

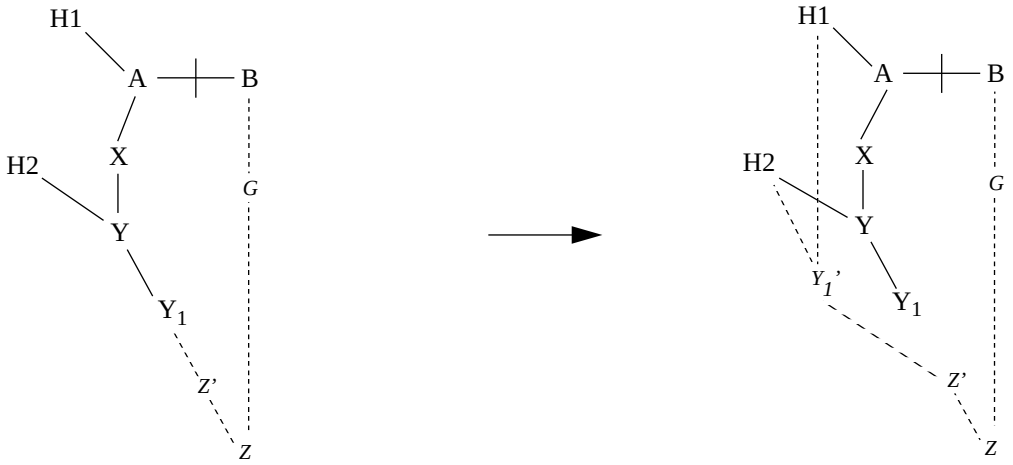
- Un utilisateur ne peut ajouter un lien entre des types, ni ajouter ou modifier une définition d'un type *si* cela entraîne une incohérence détectée par le système (e.g. un cycle de liens de spécialisation ou un sous-type commun à des types exclusifs) *sauf* lorsqu'il a créé ces types et que l'incohérence peut être levée par le système. Les deux cas suivants illustre des levées d'incohérence possibles par le système.
  Une levée d'incohérence doit permettre à un utilisateur Ux d'ajouter de modifier des types qu'il a créés sans être géné par les *spécialisations* qui ont été faites de ses types par d'autres utilisateurs (ce sont ces spécialisations qui créent l'incohérence) : le système doit modifier (le moins possible) les liens ou les types des autres utilisateurs de manière à lever l'incohérence. En effet, il n'est pas envisageable que Ux attende que les autres utilisateurs corrige la manière dont ils ont spécialisés

1. Par exemple un lien de spécialisation ou un lien d'exclusion. L'***ajout d'un sous-type*** à un type peut se faire avec une définition de type ou une commande, et soit *créer un lien de spécialisation* entre un type existant et un nouveau type, soit *créer un lien de spécialisation* entre deux types déjà existant. De même pour l'ajout d'un supertype.

2. Définitions par conditions nécessaires et/ou suffisantes, ou par conditions typiques.

ses types. Par ailleurs, la correction effectuée par le système peut être partiellement ou entièrement ou partiellement acceptée par les autres utilisateurs, i.e. peu ou pas modifiée par ces utilisateurs.

1) Soit A, B, X et Y des types ayant été créés par un utilisateur Ux (X peut être confondu avec A). Soit Z un type non co-créé par Ux et sous-type de X et de Y. Si
1) A et B sont exclusifs et que Ux ajoute un lien de spécialisation entre X et Y qui rend Z sous-type[1] de A et B (Z est alors sous-type de Y, X, A et B), ou bien
2) Z est sous-type de Y, X, A et B, et que Ux ajoute un lien d'exclusion entre A et B,
l'incohérence engendrée peut être corrigée par le système de la manière indiquée ci-dessous. Le principe retenu est tout d'abord de "clôner" un des supertypes de Z créé par Ux, c'est à dire de le remplacer par un type "clône" ayant les mêmes supertypes que le type clôné sauf ceux qui entraînent l'incohérence. La figure ci-dessous illustre la méthode (décrite après la figure).



Dans cet exemple, Y1, Y, X, A, B ont été créés par Ux, et Z, Z', G par l'utilisateur Uz,
S_{AB_Z} = { Z, Z', Y1, Y, X, A, B } et $S_{Z'}$ = { H1, H2 }
(H1 et H2 sont des supertypes de Y_1 qui n'entraînent pas d'incohérence).

Le système a créé Y_1' au nom de Uz
et remplacé le lien entre Z' et Y_1
par le lien entre Z' et Y_1'
(une option dans la méthode décrite ci-dessous serait de demander à Ux (ou Uz) s'il préfère que B soit clôné au lieu de Y_1).

Soit S_{AB_Z} l'ensemble des types A, B, Z et des types à la fois supertypes de Z et sous-types de A ou de B. Soit Uz le créateur de Z, soit Z' le plus grand supertype qui appartient à S_{AB_Z} et qui est sous-type direct d'un type Y_1 appartenant à S_{AB_Z} et créé par Ux (si plusieurs types peuvent jouer le rôle de Y_1, choisir n'importe lequel ou éventuellement[2] demander à Ux). Soit $S_{Z'}$ l'ensemble des supertypes de Z' et n'appartenant pas à S_{AB_Z}.

a) Le système crée un type atomique Y_1' en spécialisant les types de $S_{Z'}$, i.e. en créant un sous-type commun aux types les plus spécialisés de $S_{Z'}$. Cela ne devrait pas créer d'incohérence détectable par le système puisqu'il n'y en avait pas avant l'ajout du lien entre X et Y.

b) Le système détruit le lien de spécialisation entre Y_1 et Z' (ce qui supprime l'incohérence) et pose un lien de spécialisation entre Y_1' et Z'. Si la spécialisation de Y1 avait été effectuée via une définition pour le type Z', Y_1 est remplacé par Y_1' dans le concept porteur du paramètre formel de la définition[3]. Le type Y_1', son lien avec Z' et ses liens avec ses supertypes directs, sont créés par le

1. Lorsque nous ne précisons pas "sous-type direct", un sous-type peut être direct ou indirect.

2. Ux peut ne pas avoir envie de s'occuper des spécialisations faites sur ses types par d'autres utilisateurs. Quant à Uz, il pourrait mettre un certain temps à répondre si le système lui posait la question et pendant ce temps, la hiérarchie de types serait incohérente ce qui pourrait géner Ux dans son travail.

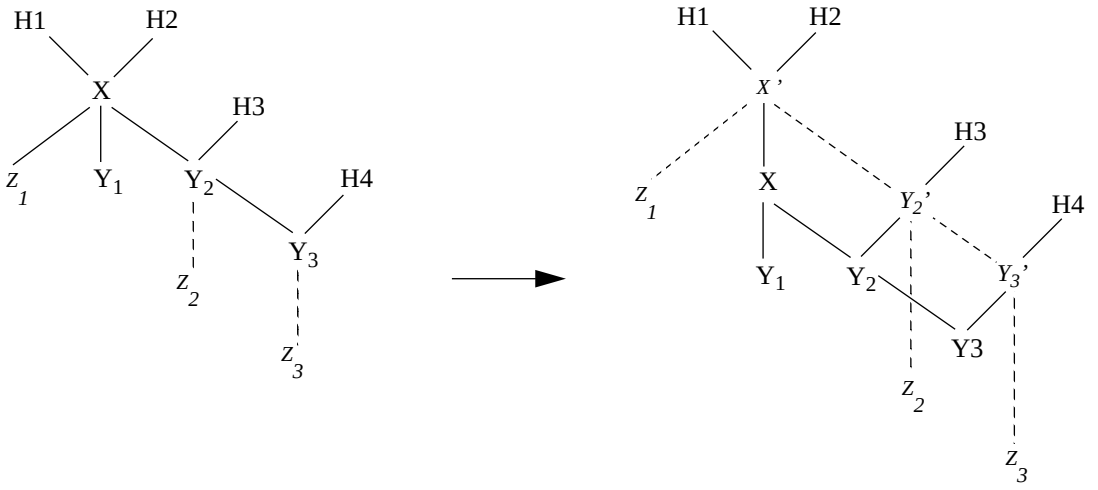
3. Voir section 4.2.2.4 à la page 77. Nous supposons qu'il s'agit d'une définition d'un type de concept et qu'il n'y a donc qu'un seul concept porteur du paramètre formel.

système tout comme si Uz les avait créés lui-même. Uz n'est plus co-créateur de Y_1. Les co-créateurs de Z deviennent co-créateurs de Y_1' et ne le sont plus de Y_1. Pour le créateur et pour chaque co-créateur, le nom sous lequel Y_1' lui est visible peut par défaut être construit avec celui sous lequel Y_1 lui était visible mais avec en plus un préfixe tel que "clône_de_".

c) Le système envoie un message à Uz pour l'avertir de cette destruction et ces créations de remplacement.

Ainsi, Z hérite des mêmes propriétés qu'avant la destruction du lien entre Y_1 et Z' sauf de celles qui entraînaient l'incohérence détectée. L'opération effectuée par le système est donc une relaxation de contraintes pour lever l'incohérence. Grâce à cela, un utilisateur peut s'il le désire poser des liens entre des types qu'il a créés sans se soucier de la façon dont d'autres utilisateurs ont spécialisé ces types (ces spécialisations peuvent d'ailleurs être invisibles aux yeux de cet utilisateur : la détection d'incohérence prend tous les types en compte, visibles ou non).
Si la hiérarchie de types avait une structure de treillis, après une telle opération, elle peut avoir perdue cette structure. Notons cependant que les types et les liens engendrés par de telles relaxations de contraintes peuvent être temporaires dans le sens où leurs "créateurs" (ceux au nom de qui le système a créé ces types), une fois avertis de ces créations, vont probablement les remplacer par des types ou des liens plus adéquats.

2) Lorsque l'ajout ou la modification d'une définition[1] à un type X par un utilisateur Ux ayant créé ce type, entraîne une incohérence et que celle-ci est due (entre autres) à l'existence de sous-types de X créés par d'autres utilisateurs que Ux (soit Ez l'ensemble de ces sous-types), cette incohérence peut être corrigée par le système de la manière indiquée ci-dessous.
Le principe retenu est comme précédemment une relaxation de contrainte via le clônage d'un type : ici les types clônés sont le type X et éventuellement certains de ses sous-types (le clônage est différent de précédemment).
La figure ci-dessous illustre la méthode (décrite à la page suivante).



$Ez = \{ Z_1, Z_2, Z_3 \}$
Ux vient d'ajouter une définition à X. Cette nouvelle définition entraîne une incohérence avec les (définitions des) types Z_1, Z_2 et Z_3, lesquels n'ont pas été créés par Ux.

Le système a généré les types X', Y_2' et Y_3' respectivement équivalents à X, Y_2 et Y_3 avant l'ajout de la nouvelle définition à X.

1. Le type X existe déjà et une définition par conditions nécessaires et/ou suffisantes ou typiques lui est ajouté. Cette opération ne crée pas un sous-type à X, elle le définit de manière plus précise. Voici deux exemples de définitions qui peuvent être ajoutées aux type Cat :
1) NC for Cat (x) are [Pet : *x]. //dans cette définition, [Pet:*x] est le concept porteur du paramètre formel *x
2) TC for Cat (x) are [Cat:*x]->(Attr)->[Lazy]. //dans cette définition, [Cat:*x] est le concept porteur du
//paramètre formel *x

a) Le système insère entre X et ses supertypes directs, le type X' qui a les mêmes définitions que X avait avant l'ajout ou la modification de la définition. Il transfère à X' les sous-types directs de X appartenant à Ez (s'il y en a).

b) Appelons "clôner un type Y sous le type C" l'opération consistant tout d'abord à insérer entre Y et ses supertypes directs qui ne sont pas sous-types de X, le type Y' (si Y est atomique, Y' est atomique, sinon Y' a les mêmes définitions que Y en remplaçant dans ces définitions le type du concept porteur de la variable par C) puis à sous-typer C par Y'.
Appelons "clône de Y" le résultat du clônage d'un sous-type de X sous un sous-type de X'.
Le système effectue les opérations suivantes :

```
pour chaque sous-type Y de X et
                appartenant à Ez ou ayant un sous-type appartenant à Ez,
    si Y n'appartient pas à Ez,
        si Y est un sous-type direct de X, clôner Y sous X',
        sinon clôner Y sous le clône de son supertype direct sous-type de X
    sinon (Y appartient à Ez),
        si Y est un sous-type direct de X
            remplacer le lien de spécialisation entre X et Y
            par un lien de spécialisation entre X' et Y
        sinon
            soit G le supertype direct de Y qui est sous-type de X, soit G' le clône de G,
            remplacer le lien de spécialisation entre G et Y
            par un lien de spécialisation entre G' et Y.
            Si  la spécialisation de G avait été effectuée via une définition pour le type Y,
            Y1 est remplacé par Y1' dans le concept porteur du paramètre formel de
            cette définition.
```

Les opérations de clônage et de remplacement de liens ci-dessus sont, tout comme dans le cas précédent, des opérations de relaxation de contraintes destinées à supprimer des incohérences. Les "créateurs" des types et les liens ainsi créés sont les créateurs des types de Ez. Si un clône est nécessaire pour lever les incohérences créées par plusieurs types de Ez, l'un seulement des créateurs de ces types devient créateur des clônes, les autres créateurs deviennent co-créateurs de ces clônes. Les créateurs des types de Ez ne sont plus co-créateurs du type X et des types clônés. Les noms sous lesquels ces créateurs voient le type X' et les clônes peuvent par défaut être construits en utilisant respectivement les noms sous lesquels il voyaient X et les types qui ont été clônés mais avec en plus un préfixe tel que "clône_de_".

c) Le système envoie un message aux créateurs de chacun des types de Ez afin de les avertir des liens détruits et des types et liens créés en leurs noms.

Ainsi, un utilisateur peut ajouter des définitions à des types qu'il a créés sans se soucier de la façon dont d'autres utilisateurs ont spécialisé ces types. Ceci peut cependant entraîner la création de nombreux types artificiels. Il est donc préférable qu'un utilisateur prenne en compte la façon dont ses types ont été spécialisés par d'autres utilisateurs avant de les modifier. Ainsi par exemple, plutôt que d'ajouter des définitions à ses types, il peut être préférable d'ajouter des sous-types définis à ces types.

- *Un utilisateur Ux qui,* étant donné une modification d'un de ses types par un autre utilisateur Uy, *comprend que Uy a mal interprété le sens qu'il accordait à ce type, peut aider Uy* (et la construction coopérative de la hiérarchie de types) *en ajoutant à son tour plus d'informations à ses types de manière à expliciter l'incohérence et la rendre détectable par le système*. Le système va alors modifier certains liens créés par cet autre utilisateur de manière à réparer cette incohérence puis va avertir cet utilisateur afin qu'il ré-étudie l'organisation de ses types. Ainsi,
1) l'utilisateur Ux ne modifie pas directement les types de Uy mais précise ses types,
2) des modifications minimales pour lever l'incohérence sont effectuées et automatiquement,
3) Uy peut ré-étudier *quand il le souhaite* la manière dont ses types peuvent réutiliser les types de Ux (ce dernier n'est *pas bloqué* en attendant les corrections de Uy).
Notons que pour expliciter l'incohérence et ainsi engendrer des corrections, Uy peut poser un lien d'exclusion entre un de ses types et l'un de ceux de Uy. S'il utilise cette facilité, il doit préalablement étudier les types de Uy et leur organisation, de manière à s'assurer que Uy a réellement

mal interprété le sens de certains de ses types (il peut en effet s'agir d'un simple problème de nommage de type ; dans ce cas, Ux peut mettre des alias sur des types de Uy de manière à les visualiser sous des noms qu'il juge plus adéquats).

- Lorsqu'un utilisateur Ux détruit un lien qu'il a créé entre deux types X et Y et que ce lien a des co-créateurs, ce lien continue à exister mais ne soit plus visible par Ux.
  Si par contre, ce lien n'a pas de co-créateur, il est détruit ; dans ce cas, si ce lien était un lien de spécialisation et que Y n'avait pas d'autre supertype, Y est détruit ainsi que ses éventuels sous-types (ceux-ci ont forcément été créés pas Ux et n'ont pas de co-créateur). Toutefois, si Y a des sous-types, le système doit demander à Ux une confirmation de la destruction de Y et de ses sous-types.

Annexe 3 Quelques ontologies de haut niveau

3 Sommaire

3.1 Les types de plus haut niveau proposés par CGKAT 302
3.1.1 Les types de concepts de plus haut niveau proposés par CGKAT 302
3.1.2 Les types de relations de plus haut niveau proposés par CGKAT 303
3.2 Les catégories de plus haut niveau de WordNet 1.5 304
3.3 Les types de concepts de haut niveau de Sowa (1992) 307
3.4 Les types de concepts de haut niveau de Sowa (1995b) 307
3.5 Les types de concept de haut niveau de Tepfenhart (1992) 308
3.6 Les catégories de plus haut niveau de Skuce (1995) 309
3.7 Les types de concepts de Bateman & al. (1996) 310
3.8 Les types de concepts de haut niveau de Mikrokosmos 311
3.9 Les catégories de plus haut niveau de CYC 312
3.10 Les types de relations de haut niveau de Bateman & al. (1996) 313
3.11 Les types de relations de haut niveau de Mikrokosmos 314
3.12 Les types de relations de Myaeng & Koo (1994) 315
3.13 Les types de relations casuelles selon Vilnat (1992) 315

Nous présentons ci-après quelques figures montrant l'organisation des catégories de plus haut niveau dans diverses taxinomies, en commençant par celles données par CGKAT pour les types de concept et les types de relations.

Nous détaillons ainsi les catégories de haut niveau de WordNet qui sont incluses et (ré)organisées par l'ontologie de CGKAT pour les types de concepts, c'est à dire les catégories de haut niveau de WordNet pour les ensembles de synonymes formés avec des noms.

Nous présenterons ensuite les typologies de Sowa (1992), Sowa (1995b) et Tepfenhart (1992) dont nous nous sommes le plus inspirés.

Puis, à titre de comparaison, nous montrerons les catégories de plus haut niveau des ontologies de Skuce (1995), de Bateman (1990), de Mikrokosmos (Mahesh, 1996), de CYC (Lenat & Guha, 1990a) et du Generalized Upper Model[1] (Bateman & al., 1994) (Bateman & al, 1996).

Le Generalized Upper Model contient également des catégories que nous avons fournies en tant que types de relation. Nous montrerons également d'autres figures concernant des classifications de types de relations : celles de Mikrokosmos (Mahesh, 1996), de Myaeng & Koo (1994) et de Vilnat (1992).

1. Le "Generalized Upper Model" est une évolution du "PENMAN upper Model" (Bateman, 1990).

3.1 Les types de plus haut niveau proposés par CGKAT

3.1.1 Les types de concepts de plus haut niveau proposés par CGKAT

Figure A.3.1: *Les types de concepts de plus haut niveau dans l'ontologie proposée par défaut par CGKAT.*

3.1.2 Les types de relations de plus haut niveau proposés par CGKAT

```
Relation (Concept,Concept)
    Attributive_relation (Concept,Concept)  --  e.g. Chrc, Attr, Manner, Name, Role, Possession, Accompaniment
    Component_relation (Concept,Concept)  --  e.g. Part, Main_Part, Element, Subset, Subtask, FirstSubTask
    Constraint_or_measure_relation (Concept,Concept)  --  e.g. mathematical, spatial and temporal relations
    Relation_from_a_situation (Situation,Concept)  --  i.e. only from a state or a process
        %Constraint_or_measure_relation_from_a_situation (Concept,Concept)  --  i.e. only from a state or a process
            .Descr (Situation,Proposition)  --  for describing a situation
            %Temporal_relation_from_a_situation (Situation,Concept)
                Situation_temporal_location (Situation,Temporal_entity)  --  e.g. Point_in_time, Duration, T_Since, T_Until
                Temporal_relation_between_situations (Situation,Situation)  --  e.g. T_Succ, Task_Succ, Terminaison, Consequence
            %Spatial_relation_from_a_process (Process,Spatial_entity)  --  e.g. Source, Destination, Path
        Relation_from_a_process (Process,Concept)
            .%Purpose (Process,Situation)  --  a purpose of a process is also a reason to do this process
            %Recipient (Process,Goal_directed_agent)  --  e.g. Beneficiary
            .%Experiencer (Process,Goal_directed_agent)
            %Result (Process,Concept)
                .O (Process,Concept)  --  output only object
                %IO (Process,Concept)  --  input-output object, e.g. Object_to_modify, Object_to_mute, State
            Object (Process,Concept)  --  this case relation is also usually called Patient or Theme
                I (Process,Concept)  --  input only object (for example for evaluation or comparison processes)
                    .Material (Process,Concept)
                    .%Parameter (Process,Concept)
                    .SI (Process,Concept)  --  static input (e.g. for KADS tasks)
                    .DI (Process,Concept)  --  dynamic input (e.g. for KADS tasks)
                %IO (Process,Concept)  --  input-output object, e.g. Object_to_modify, Object_to_mute, State
            .%Initiator (Process,W_causal_agent__cause__causal_agency)
            .%Agent (Process,Goal_directed_agent)
            .Instrument (Process,Entity)
            %Manner (Process,Measure)  --  process attribute, e.g. Quickly
            .Method (Process,Process)
            %SubProcess (Process,Process)  --  e.g. SubTask, FirstSubTask, LastSubTask
            %Spatial_relation_from_a_process (Process,Spatial_entity)  --  e.g. Source, Destination, Path
    Relation_to_a_situation (Concept,Situation)  --  i.e. only to a state or a process, e.g. Relation_to_a_process
    Relation_from_a_proposition (Proposition,Concept)  --  e.g. Information_source, Author, Rhetorical_relation, Argumentation_relation
    Relation_refering_to_a_process (Concept,Concept)  --  e.g. Summarize, Change_location, Relation_between_agents
    Relation_with_a_special_property (Concept,Concept)  --  e.g. Transitive_relation, Symmetric_relation, Contextualizing_relation
    Relation_used_by_an_agent (Concept,Concept)  --  for accessing the relation types accepted/used by a (group of) agent(s)
        .Relation_created_by_an_agent (Concept,Concept)  --  may be useful when many knowledge engineers work on the same ontology
```

Figure A.3.2: *Les types de relations de plus haut niveau dans l'ontologie proposée par défaut par CGKAT.*

3.2 Les catégories de plus haut niveau de WordNet 1.5

entity -- (something having concrete existence; living or nonliving)
=> life form, organism, being, living thing -- (any living entity)
=> cell -- (the basic structural and functional unit of all organisms; ...)
=> causal agent, cause, causal agency -- (any entity that causes events to happen)
=> object, inanimate object, physical object -- (a nonliving entity)
=> thing -- (an entity that is not named specifically; "I couldn't tell what the thing was")
=> whole, whole thing, unit -- (an assemblage of parts that is regarded as a single entity; ...)
=> subject, content, depicted object -- (something selected by an artist for graphic representation)
=> unit, building block -- (a single undivided natural entity occurring in the composition of something else)
=> part, piece -- (a portion of a natural object)
=> necessity, essential, requirement, requisite, necessary, need -- (anything indispensible that is needed; ...)
=> inessential -- (anything that is not essential)
=> variable -- (something that is likely to vary; something that is subject to variation; ...)
=> anticipation -- (some early entity whose type or style anticipates a later one; ...)

location -- (a point or extent in space)
=> home -- (the country or state or city where you live; ...)
=> earth -- (the abode of mortals (as contrasted with heaven or hell))
=> imaginary place -- (a place said to exist in religious or fictional writings)
=> line -- (a spatial location defined by a real or imaginary unidimensional extent)
=> point -- (the precise location of something; "she walked to a point where she could survey the whole street")
=> region, part -- (the extended spatial location of something; "the farming regions of France"; ...)
=> region -- (a large indefinite location on the surface of the earth)
=> whereabouts -- (the general location where something is; ...)
=> pass, mountain pass, notch -- (the location in mountains that is lower than the surrounding peaks)

abstraction -- (a concept formed by extracting common features from examples)
=> time -- (the continuum of experience in which events pass from the future through the present to the past)
=> space -- (the unlimited 3-dimensional expanse in which everything is located)
=> attribute -- (an abstraction belonging to or characteristic of an entity)
=> measure, quantity, amount, quantum -- (how much there is of anything)
=> set -- (an abstract collection of numbers or symbols; "the set of prime numbers is infinite")

group, grouping -- (any number of entities (members) considered as a unit)
=> arrangement -- (an orderly grouping (of things or persons))
=> kingdom -- (a basic group of natural objects)
=> biological group -- (a group of plants or animals)
=> world, human race, humanity, humankind, mankind, man -- (all of the inhabitants of the earth; ...)
=> people -- (any group of persons (men or women or children) collectively; ...)
=> social group -- (people sharing some social relation)
=> collection, aggregation, accumulation, assemblage -- (several things grouped together)
=> edition -- (all of the identical copies of something offered to the public at the same time; ...)
=> ethnic group, ethnos -- (people of the same race or nationality who share a distinctive culture)
=> race -- (people who are believed to belong to the same genetic stock)
=> subgroup -- (a distinct and often subordinate group within a group)
=> sainthood -- (saints collectively)
=> citizenry, people -- (the body of citizens of a state or country; "the Spanish people")
=> population -- (a group of organisms of the same species populating a given area; ...)
=> multitude, masses, mass, hoi polloi, people -- (the common people generally; ...)
=> system -- (a group of interrelated elements comprising a unified whole)

Figure A.3.3: *Les catégories de haut niveau de WordNet relatives à des "entités" (selon nous).*

state -- (the way something is with respect to its main attributes; ...)
=> skillfulness -- (the state of being skillful)
=> cognitive state, state of mind -- (the state of a person's cognitive processes)
=> medium -- (a state that is intermediate between extremes; a middle position; "a happy medium")
=> condition -- (a mode of being or form of existence of a person or things: "the human condition")
=> condition, status -- (a condition or state at a particular time: "a condition (or state) of disrepair"; ...)
=> situation, state of affairs -- (the general state of things; the combination of circumstances at a given time;...)
=> relationship -- (a state of connectedness between people (especially an emotional connection); ...)
=> relationship -- (a state involving mutual dealings between people or parties or countries)
=> utopia -- (ideally perfect state esp in its social and political and moral aspects)
=> dystopia -- (state in which the condition of life is extremely bad as from deprivation or oppression or terror)
=> nature, wild, natural state, state of nature -- (a wild primitive state untounched by civilization; ...)
=> isomerism -- (the state of being an isomer; ...)
=> degree, level, stage, point -- (a specific identifiable position in a continuum or series or esp in a process; ...)
=> office, power -- ((of a government or government official) holding an office means being in power; ...)
=> status, position -- (the relative position or standing of things or esp persons in a society, ...)
=> being, beingness, existence -- (the state or fact of existing: "a point of view gradually coming into being";...)
=> nonbeing
=> employment -- (the state of being employed or having a job)
=> unemployment -- (the state of being unemployed or not having a job: ...)
=> order -- (established customary state esp. of society; "order ruled in the streets"; "law and order")
=> disorder -- (a disturbance of the peace or of public order)
=> hostility, enmity, antagonism -- (a state of deep-seated ill-will)
=> illumination -- (the degree of visibility of your environment)
=> emotional state, spirit -- (the state of a person's emotions (especially with regard to pleasure or dejection);...
=> freedom -- (the condition of being free; the power to act or speak without externally imposed restraints)
=> representation, delegacy, agency -- (the state of serving as an official and authorized delegate or agent)
=> dependence, dependance, dependency -- (lack of independence or self-sufficiency)
=> motion -- (a state of change; "they were in a state of steady motion")
=> motionlessness, stillness
=> dead letter, non-issue -- (the state of something that has outlived its relevance)
=> action, activity, activeness -- (the state of being active; "his sphere of action"; "volcanic activity")
=> inaction, inactivity, inactiveness -- (the state of being inactive)
=> temporary state
=> imminence, imminency, impendence, impendency, forthcomingness -- (... liable to happen soon)
=> readiness, preparedness -- (the state of being ready or prepared for use or action esp military action:...)
=> union -- (the state of being united)
=> maturity, matureness -- (state of being mature; full development)
=> immaturity, immatureness -- (not having reached maturity)
=> grace, state of grace -- (a state of sanctification by God)
=> omniscience -- (the state of being omniscient; having infinite knowledge)
=> omnipotence -- (the state of being omnipotent; having unlimited power)
=> perfection, flawlessness, ne plus ultra
=> integrity, unity, wholeness -- (an unreduced or unbroken completeness or totality)
=> imperfection, imperfectness -- (the state or an instance of being imperfect)
=> receivership -- (the state of property that is in the hands of a receiver; "the business is in receivership")
=> state of matter -- (gases and liquids and solids are the three traditional states of matter)

Figure A.3.4: *Les catégories de haut niveau de WordNet relatives à des "états" (selon nous).*

act, human action, human activity -- (something that people do or cause to happen)
=> action -- (something done (usually as opposed to something said); ...)
=> nonaccomplishment, nonachievement -- (an act that does not achieve its intended goal)
=> rejection -- (the act of rejecting something; "his proposals were met with rejection")
=> activity -- (any specifi c activity or pursuit; "they avoided all recreational activity")
=> judgment, judgement, assessment -- (the act of judging or assessing; ...)
=> production -- (the act of producing something)
=> stay -- (continuing or remaining in a place; "they had a nice stay in Paris")
=> residency, residence, abidance -- (the act of dwelling in a place)
=> inactivity -- (being inactive)
=> hindrance, interference -- (the act of hindering or obstructing or impeding)
=> stop, stoppage -- (the act of stopping something; "the third baseman made some remarkable stops")
=> group action -- (action taken by a group of people)
=> communication, communicating -- (the activity of communicating)
=> speech act -- (the use of speech to perform some act)

event -- (something that happens at a given place and time)
=> might-have-been -- (an event that could have occurred but never did)
=> nonevent -- (an anticipated event that turns out to be far less signifi cant than was expected)
=> happening, occurrence, natural event -- (an event that happens)
=> social event -- (an event characteristic of persons forming groups)
=> miracle -- (a marvellous event manifesting a supernatural act of God)
=> Fall -- (the lapse of mankind into sinfulness because of the sin of Adam and Eve; ...)

phenomenon -- (any state or process known through the senses rather than by intuition or reasoning)
=> natural phenomenon, nature -- (all non-artifi cial phenomena)
=> levitation -- (the phenomenon of a person or thing rising into the air by apparently supernatural means)
=> metempsychosis, rebirth -- (after death the soul begins a new cycle of existence in another body)

Figure A.3.5: *Les catégories de haut niveau de WordNet relatives à des "processus" (selon nous).*

3.3 Les types de concepts de haut niveau de Sowa (1992)

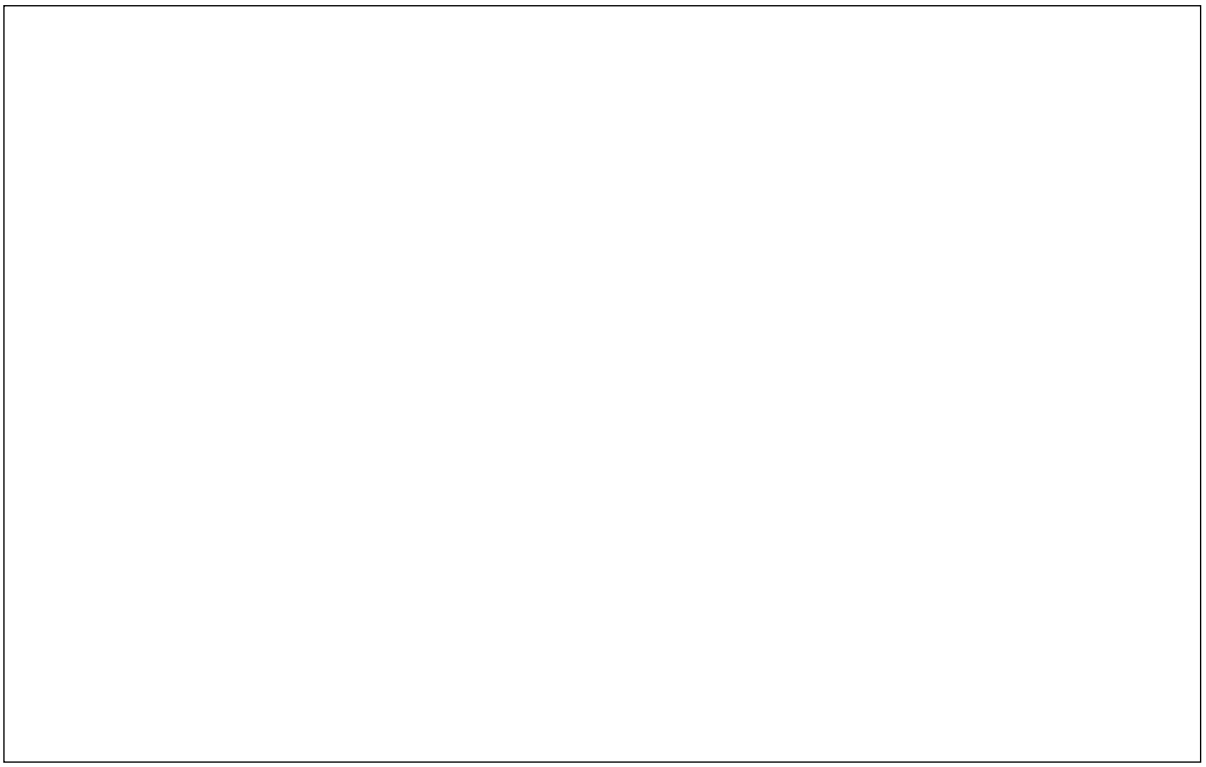


Figure A.3.6: *Types de concepts de haut niveau de Sowa (1992).*

3.4 Les types de concepts de haut niveau de Sowa (1995b)

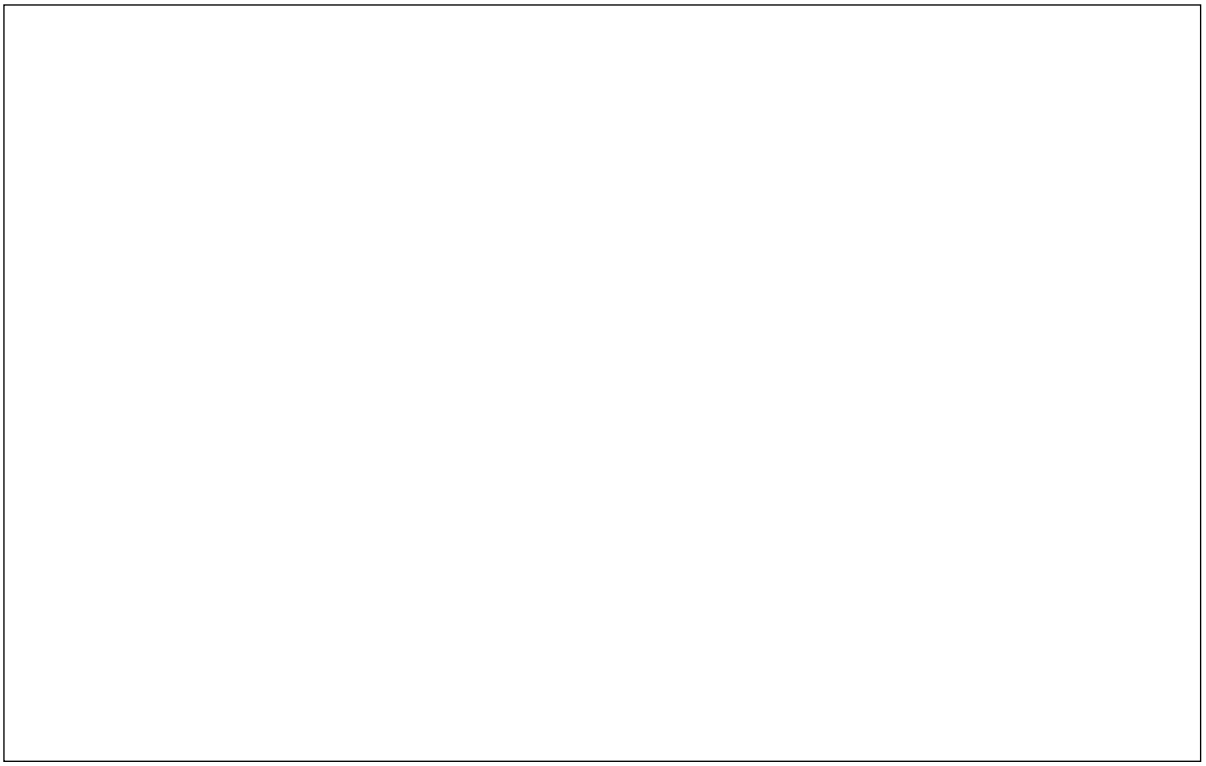


Figure A.3.7: *Types de concepts de haut niveau de Sowa (1995b).*

3.5 Les types de concept de haut niveau de Tepfenhart (1992)

Concept -- *the supertype for all concept types*

Entity -- *location, object, material, or property*
Location -- *a place and/or a time which is occupied by an object*
Place -- *a region of space (a point in space or a set of point in space)*
Time -- *a period of time (a point in time or a set of point in time)*
Object -- *thing which has a manifestation in the real world*
Abstract_object -- *thing which has an individual conceptual manifestation in the real world*
Extended_abstract_object -- *its boundaries are well defined, e.g. Military_service*
Localised_abstract_object -- *its boundaries are ill-defined e.g. Currency, Time_zone*
Concrete_object -- *thing which has an individual physical manifestation in the real world*
Substantial_object -- *its boundaries are well defined, e.g. a table, a car, an animal*
Insubstantial_object -- *its boundaries are ill- defined, e.g. a cloud*
Material -- *a constituent substance of an object*
Property -- *attribute, characteristic, feature, or capability of an object or material*
Attribute -- *directly measurable physical property of an object or material*
Characteristic -- *physical aspect of an object or material*
Feature -- *limited surface aspect of an object*
Capability -- *ability of an object to perform actions*

Scene -- *state of the world described in terms of the entities and the relationship that hold among them at a particular point in time*
State_of_affairs -- *description of the world at an instant in time*
Locus_of_affairs -- *scene which serves as token for a number of concepts and relationships which which hold among them without change throughout a situation*
State_of_affairs -- *description of the world at an instant in time*

Situation -- *a denotation for a set of scenes and the actions which connect them*
Event -- *a situation which consists of an initial scene, a final scene and an action which links the two*
Activity -- *a situation with more than two scenes*

Value -- *a measure of comparison or a label; it can be numeric (e.g. 4) or symbolic (e.g. Longer)*

Figure A.3.8: *Les premiers sous-types de Concept selon Tepfenhart (1992).*

Notons que notre notion d'entité inclut les notions d'entité et de valeur définies par Tepfenhart, et que notre notion de situation inclut les notions de scène et de situation de Tepfenhart. Il n'aurait pas été possible de classer les catégories de haut niveau de WordNet suivant les distinctions qui ont conduit Tepfenhart aux notions de scènes, de situations (ensemble de scènes), d'objets aux frontières bien définies ou mal définies. Nous n'avons pas ajouté ces distinctions à notre ontologie car elles ne nous semblent pas usuelles ou fondamentales dans la recherche ou la représentation des connaissances. Par ailleurs, à propos de la notion d'espace, il nous a semblé important d'introduire une catégorie pour les "entités spatiales", ce qui inclut les régions de l'espace mais aussi les objets qui occupe ces régions. Ainsi par exemple, nos relations spatiales, que nous avons signées sur le type Spatial_entity, peuvent s'appliquer aussi bien à des objets *occupant* des régions de l'espace, qu'à ces régions elles-même.

3.6 Les catégories de plus haut niveau de Skuce (1995)

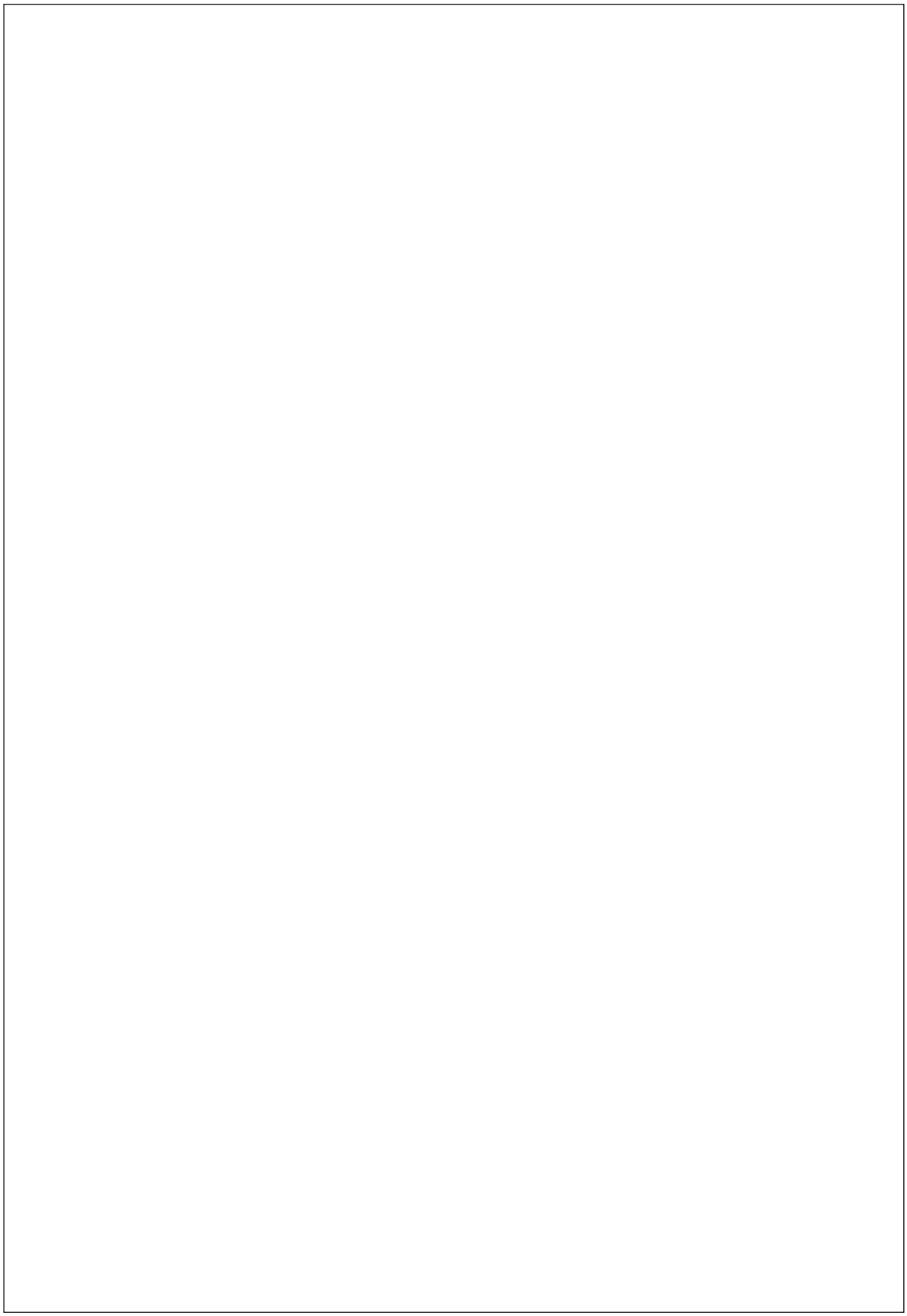


Figure A.3.9: *Les catégories de plus haut niveau de Skuce (1995).*

3.7 Les types de concepts de Bateman & al. (1996)

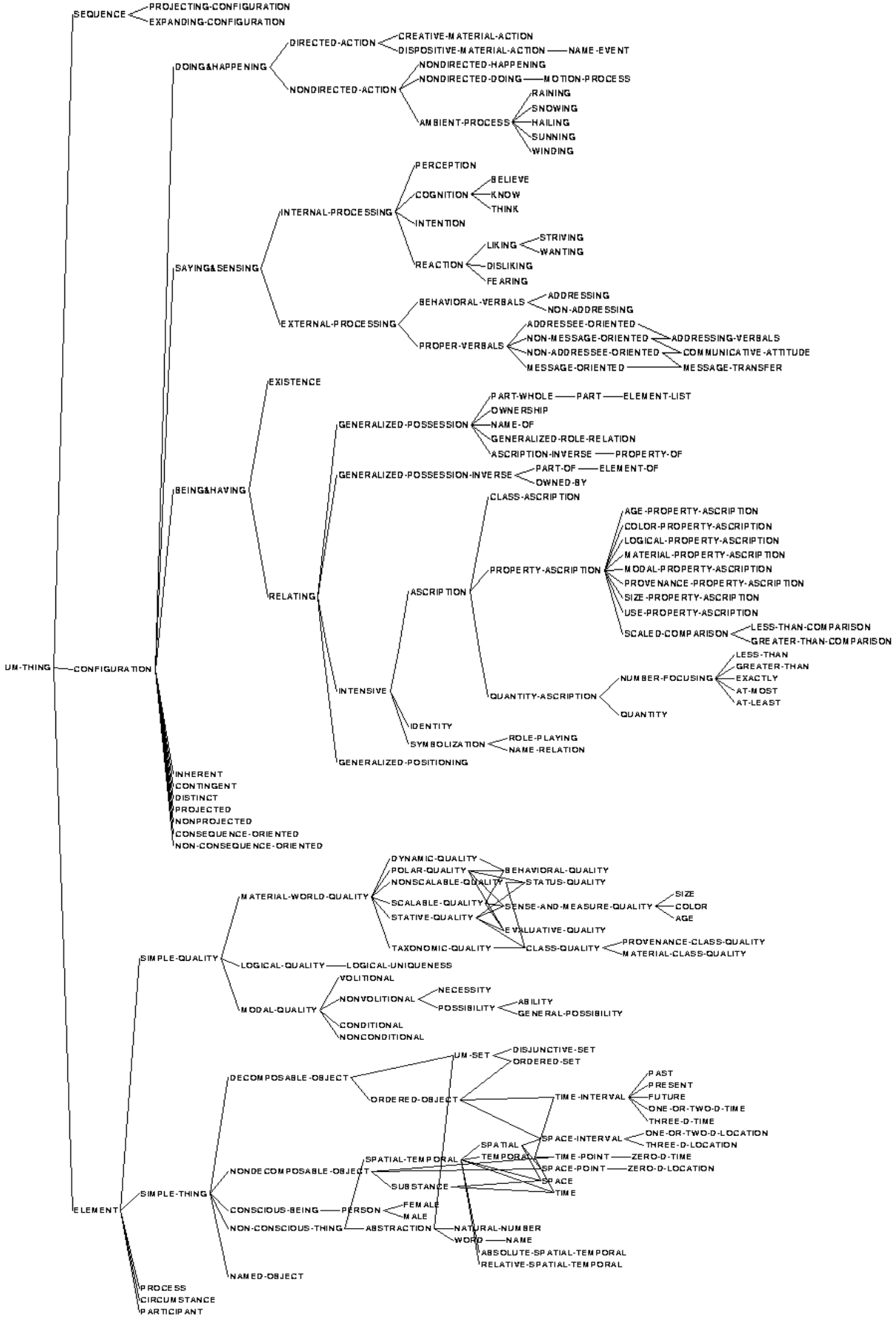


Figure A.3.10: *Types de concepts de haut niveau de Bateman & al. (1996).*

3.8 Les types de concepts de haut niveau de Mikrokosmos

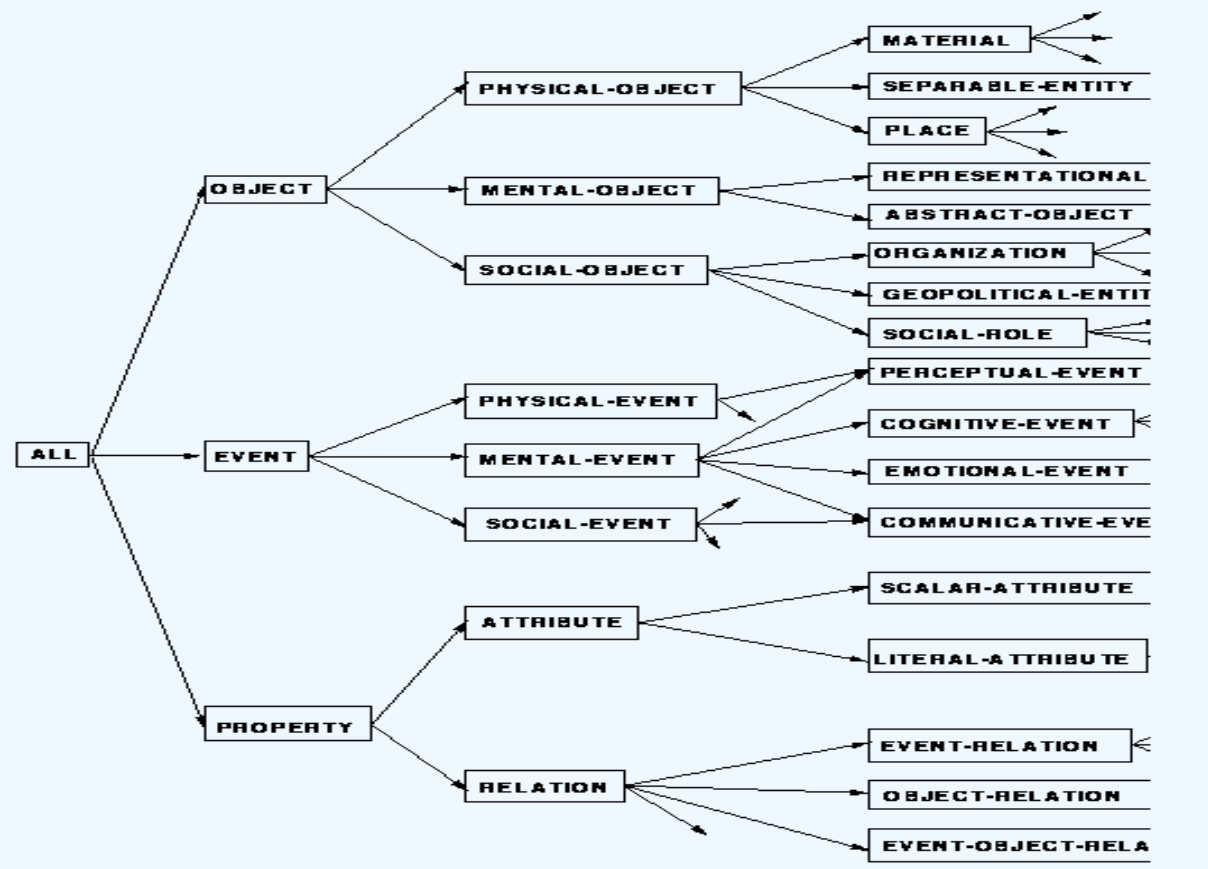


Figure A.3.11: *Types de concepts de haut niveau de Mikrokosmos (Mahesh, 1996).*

3.9 Les catégories de plus haut niveau de CYC

Figure A.3.12: *Les catégories de plus haut niveau de CYC (Lenat & Guha, 1990a)*

3.10 Les types de relations de haut niveau de Bateman & al. (1996)

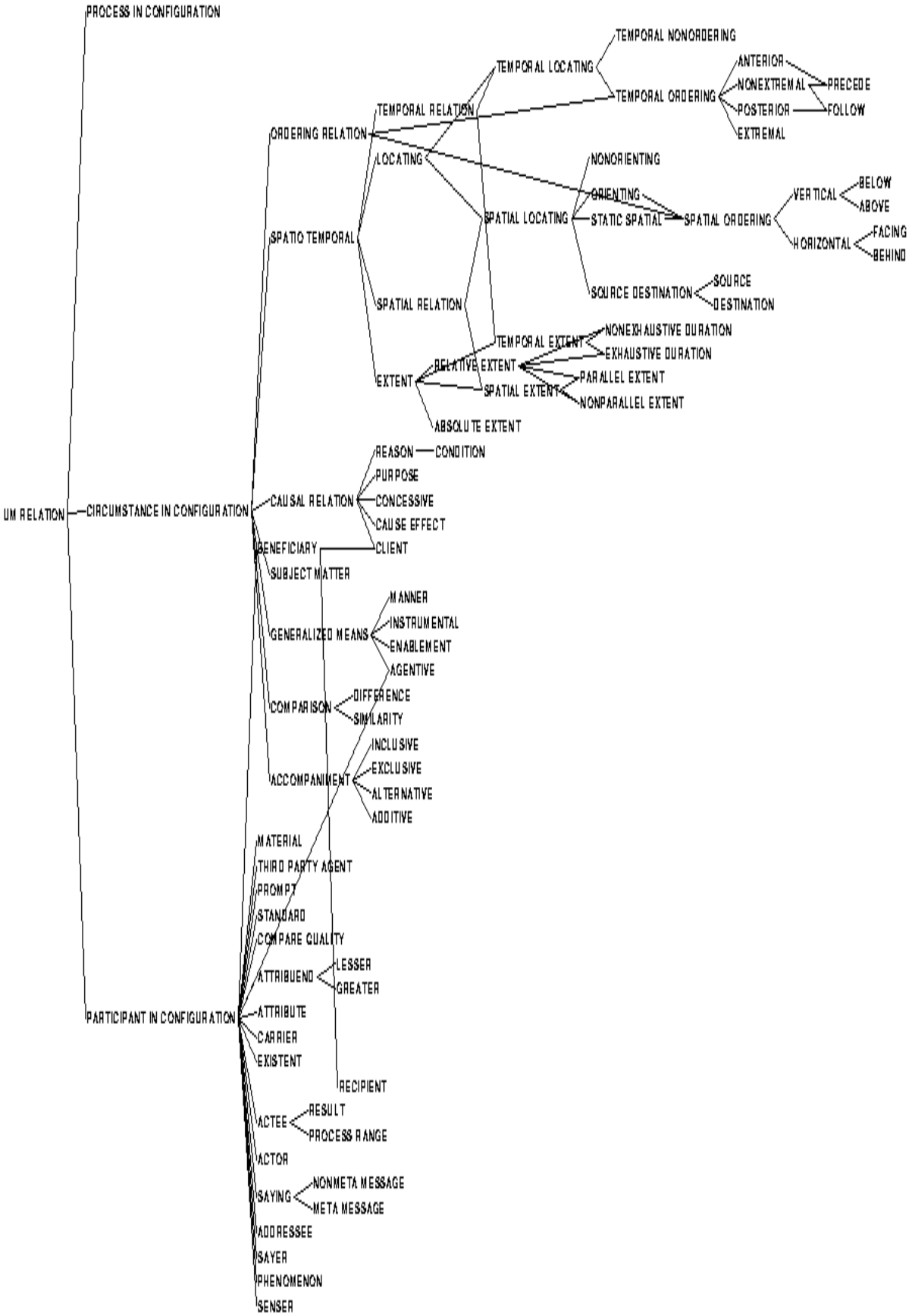


Figure A.3.13: *Types de relations de haut niveau de Bateman & al. (1996).*

3.11 Les types de relations de haut niveau de Mikrokosmos

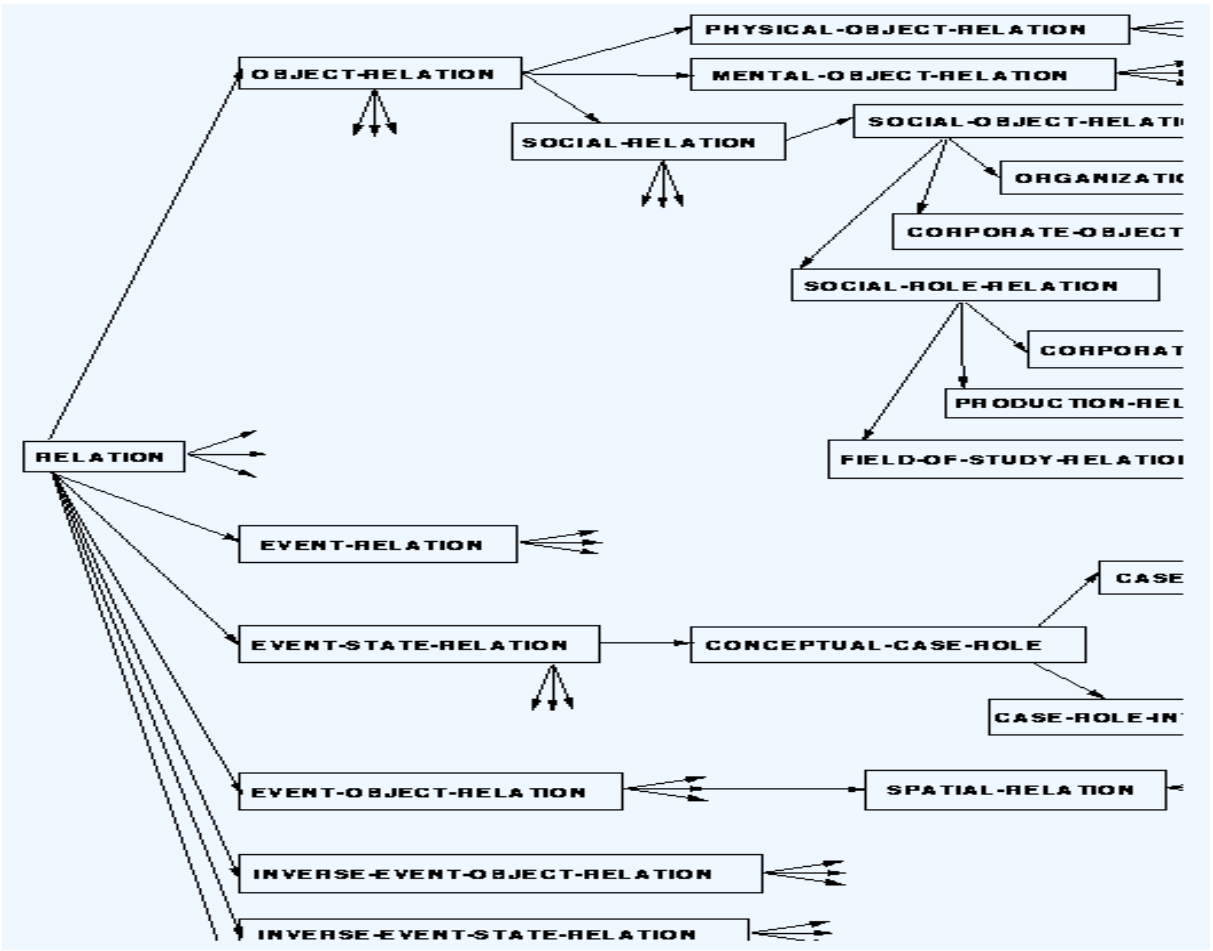


Figure A.3.14: *Types de relations de haut niveau de Mikrokosmos (Mahesh, 1996).*

3.12 Les types de relations de Myaeng & Koo (1994)

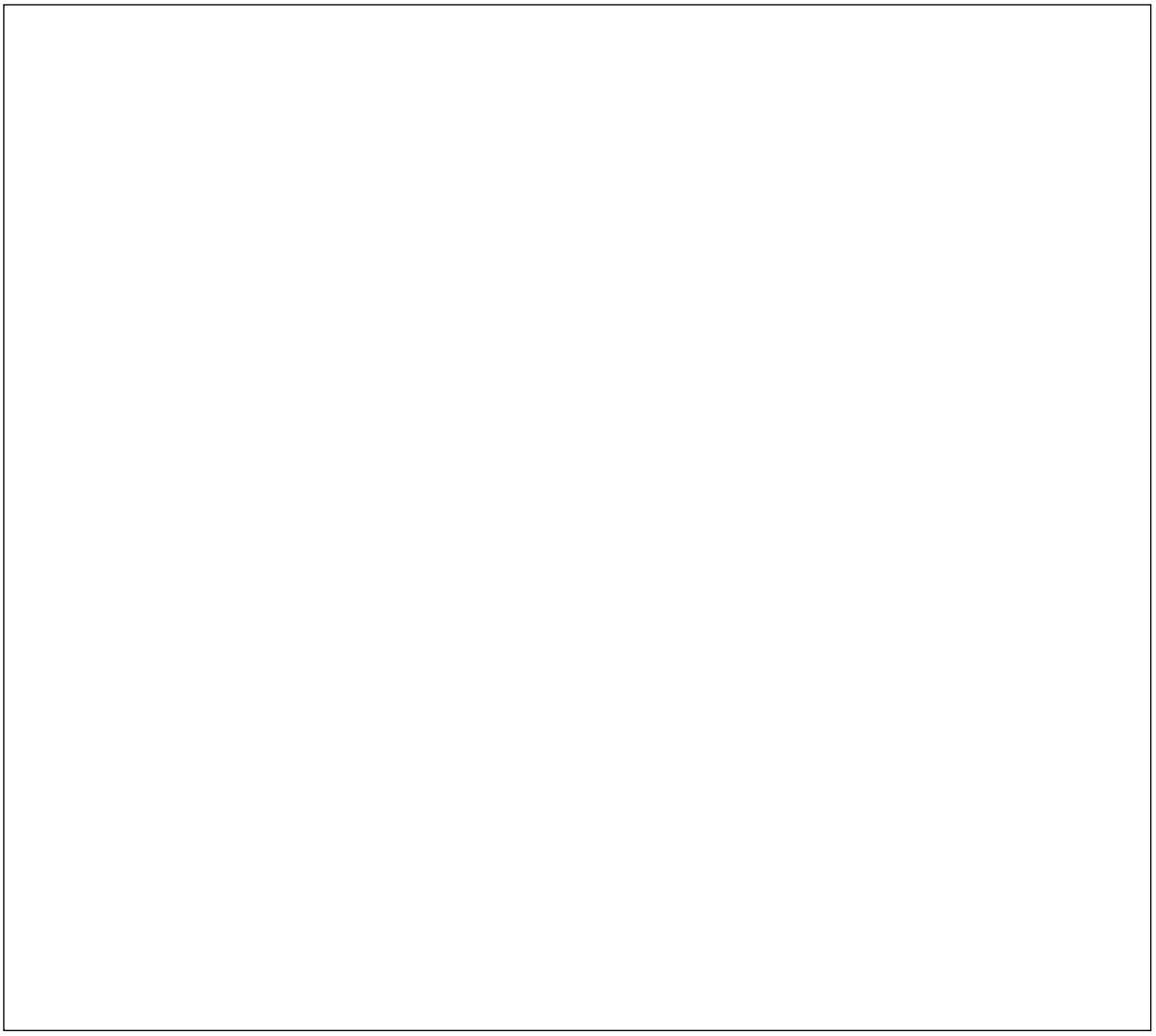


Figure A.3.15: *Les types de relations utilisés dans DR-LINK (Myaeng & Koo, 1994).*

3.13 Les types de relations casuelles selon Vilnat (1992)

Cas -- *le supertype des relations casuelles*
- Agent -- *Initiateur, Causateur et Experiencer*
- Instrumental
  - Instrument -- *Outil, Méthode*
  - Adjuvant
  - Matière -- *Ingrédient, Composant*
- Objet -- *Thème, Objet, Résultat, But*
- Situatif
  - Locatif -- *Lieu, Chemin, Origine, Destination, Distance*
  - Temps -- *Instant, Début, Fin, Durée, Intervalle*
- Destinataire -- *Bénéficiaire, Patient, Source, Possesseur*
- Descriptif
  - Manière
  - Mesure
  - Attribut -- *Etat, Qualité*

Figure A.3.16: *Les types de relations casuelles selon Vilnat (1992)*

Annexe 4 Modèles structurels, de présentation et d'interface

4 Sommaire

4.1 Modèles pour un Graphe Conceptuel **318**
4.1.1 Modèle structurel 318
4.1.2 Modèle de présentation 320
4.1.3 Modèle d'interface 325
4.2 Modèles pour un GC et certaines méta-informations **327**
4.2.1 Modèle structurel 327
4.2.2 Modèle de présentation 327
4.2.3 Modèle d'interface 328
4.3 Modèles pour l'indexation d'éléments de documents **329**
4.3.1 Modèle structurel d'extension 329
4.3.2 Modèle de présentation 330
4.3.3 Modèle d'interface par défaut 332
4.4 Modèles pour une hiérarchie **333**
4.4.1 Modèle structurel 333
4.4.2 Modèle de présentation 333
4.4.3 Modèle d'interface 335
4.5 Modèles pour utiliser un langage de commandes **336**
4.5.1 Modèle structurel 336
4.5.2 Modèle de présentation 336
4.5.3 Modèle d'interface 336

4.1 Modèles pour un Graphe Conceptuel

4.1.1 Modèle structurel

Pour des raisons de clarté, trois parties du schéma suivant sont présentées séparément. Elles sont référées ci-dessous par les commentaires {*1}, {*2} et {*3}.

```
STRUCTURE CG;   DEFPRES CGP;   { Structural model for a (DE of type) CG;  default presentation: CGP }
STRUCT
   CG = CASE OF   { here a CG is either a CGgraph, a SingleConcept or a TypeDefinition }
            CGgraph (ATTR  CGname = TEXT;  Comment = TEXT;  To_Element = REFERENCE(ANY);
                           ChangeIntoDefinition = Yes, No;
                           ConceptFrame = Without, Rectangular, RectangularShaded,
                                          Rounded, RoundedShaded;
                           RelationFrame = WithoutFrame, Rounded, RoundedShaded;
                           OrigRelLink1_Pos = Center1_, Right, Left1_, Top, Bottom;
                           DestRelLink1_Pos = Center_1, Right, Left_1, Top, Bottom;
                           OrigRelLink2_Pos = Center2_, Right, Left2_, Top, Bottom;
                           DestRelLink2_Pos = Center_2, Right, Left_2, Top, Bottom;
                           ArrowOnLink1 = ok, no;  ArrowOnLink2 = yes, not )
                  = ConceptOrRel_List with ConceptFrame?=Rectangular, RelationFrame?=WithoutFrame,
                                           OrigRelLink1_Pos?=Center1_, DestRelLink1_Pos?=Left_1,
                                           OrigRelLink2_Pos?=Left2_,   DestRelLink2_Pos?=Left_2,
                                           ArrowOnLink1 ?= no,  ArrowOnLink2 ?= yes;
            SingleConcept  = CASE OF Concept; ConceptInclusion;  END;
            TypeDefinition (ATTR  !DefKind = NSC_for_ConceptType, NC_for_ConceptType,
                                             SC_for_ConceptType, TC_for_ConceptType,
                                             NSC_for_RelationType, Unspecified_DefKind,
                                             No_more_a_definition;
                                  !Completed = not_yet, yes;  { '!' means that the attribute is mandatory }
                                   ... { same attributes as for CGgraph here}  )
                  = BEGIN  DefinedType = TEXT;  LambdaVar = TEXT;
                           DefBody = CGgraph;
                    END  with  DefKind ?=TC_for_ConceptType,  Completed ?=not_yet;
        END;                { with --> default values for attributes }
   ConceptOrRel_List = LIST OF (ConceptOrRelation);

   ConceptOrRelation = CASE OF
      Concept (ATTR  IDinCG=INTEGER;  ... { same attributes as for CGgraph here }  )
            = BEGIN  ConceptType = TEXT;
                     Referent = CASE OF  Variable=TEXT;  Individual=TEXT;  END;
                   ? CGReferent (ATTR  ... { same attributes as for CGgraph } )     { ? --> optional }
                           = CASE OF  CGgraph; TypeDefinition;  END;
                   ? DescriptionReferent = UNIT;  { -> texts, images and other ED; not copied in CoGITo }
              END;
      ConceptInclusion (ATTR  IDinCG; Comment; To_Element; ChangeIntoDefinition; ConceptFrame )
            = BEGIN  TheIncludedConcept = Concept;  END;
      Relation (ATTR  IDinCG;  Comment; To_Element;   !OrigNode=REFERENCE(SingleConcept);
                                                      !DestNode=REFERENCE(SingleConcept);  ... {*1})
            = BEGIN
                 LinkToOrig (ATTR OrigRelLink1_Pos; DestRelLink1_Pos; ArrowOnLink1; ... {*2} )
                       = BEGIN   GRAPHICS;   ? LinkToOrig_Mark = TEXT;   END;
                 ? OtherLinkToOrig = LIST OF (LinkToOrig);
                   RelationType (ATTR RelationFrame) = TEXT;
                   LinkToDest (ATTR OrigRelLink2_Pos; DestRelLink2_Pos; ArrowOnLink2; ... {*3} )
                       = BEGIN   GRAPHICS;   ? LinkToDest_Mark = TEXT;   END;
                 ? OtherLinkToDest = LIST OF (LinkToDest);
              END;
      RelationAndConcept = BEGIN  Concept; Relation;  END;  { gadget for building CGs faster }
                         END;

UNITS
   IncludedCG = CGgraph;
```

```
EXCEPT
   CG : NoMove,NoResize;          CGgraph : NoMove,NoResize;              CGReferent : NoMove,NoResize;
   Relation : NoSelect;           ConceptType : NoMove,NoResize;          Referent : NoMove,NoResize;
   Individual : NoMove,NoResize;  Variable : NoMove,NoResize;             DefinedType : NoMove,NoResize;
   LambdaVar : NoMove,NoResize;            TheIncludedConcept: NoSelect,NoMove,NoResize;
   TEXTE : NoSelect,NoMove,NoResize;       To_Element : ActiveRef;        DefKindCopy : Invisible;
END
```

Voici les parties omises {*1}, {*2} et [*3}.

```
{*1}:  //for non binary relations, other reference attributes than OrigNode and DestNode are necessary
   OrigNode2=REFERENCE(SingleConcept);   OrigNode3=REFERENCE(SingleConcept);
   OrigNode4=REFERENCE(SingleConcept);   OrigNode5=REFERENCE(SingleConcept);
   DestNode2=REFERENCE(SingleConcept);   DestNode3=REFERENCE(SingleConcept);
   DestNode4=REFERENCE(SingleConcept);   DestNode5=REFERENCE(SingleConcept);


{*2}:  //By default, the first link of a relation goes from the center of the first concept box
         //to the left of the relation box. The following attributes on this link allow the user
         // to specify other origin and destination points on theses boxes for this link
   OrigLink_Center=REFERENCE(ConceptOrRelType);  OrigLink_Right=REFERENCE(ConceptOrRelType);
   OrigLink_Left=REFERENCE(ConceptOrRelType);      OrigLink_Top=REFERENCE(ConceptOrRelType);
   OrigLink_Bottom=REFERENCE(ConceptOrRelType);
   DestLink_Center=REFERENCE(ConceptOrRelType);    DestLink_Right =REFERENCE(ConceptOrRelType);
   DestLink_Left=REFERENCE(ConceptOrRelType);     DestLink_Top=REFERENCE(ConceptOrRelType);
   DestLink_Bottom=REFERENCE(ConceptOrRelType);


{*3}:  //By default, the second link of a relation goes from the left of the relation box
         //to the center of the second concept box. The following attributes on this link allow the user
         //to specify other origin and destination points on theses boxes for this link
   OrigLink_Center;   OrigLink_Right;   OrigLink_Left;    OrigLink_Top;    OrigLink_Bottom;
   DestLink_Center;  DestLink_Right;   DestLink_Left;    DestLink_Top;   DestLink_Bottom;

//If another link is added by the user between a concept box and a relation box, it goes from the center of the
//concept box to the center of the relation box, or conversely. Non binary relations may be graphically built in
//this way but CGKAT does not yet build these relations in CoGITo (-> the linear format must be used in order //
to build non binary relations in CoGITo).
```

4.1.2 Modèle de présentation

```
{ This schema describes the Presentation of a Conceptual Graph
        Philippe Martin - INRIA Sophia Antipolis Projet ACACIA - January 1995
}
PRESENTATION CG;
VIEWS           CGView;
COUNTERS        CptPageCorps:  Rank of Page(Texte_integral) Init FirstPageNumber;
VAR             VarPageCorps:  Text '';   Value (CptPageCorps, Arabic);
DEFAULT
   BEGIN
        HorizRef: * . Bottom;   VertRef: * . Left; Width: Enclosed . Width;   Height: Enclosed . Height;
        HorizPos: Left = Enclosing . Left;   VertPos: Top = Enclosing . Top;
        Visibility: Enclosing =;    Depth: Enclosing =;
        Font: Enclosing =;   Style: Enclosing =;   Size: Enclosing =;    LineSpacing: Enclosing =;
        Justify: No;   Adjust: Enclosing =;    Indent: Enclosing =;
        UnderLine:  Enclosing =;  Thickness: Enclosing =; LineWeight: Enclosing =; FillPattern: Enclosing =;
        Foreground: Enclosing =; BackGround: Enclosing =;
   END;

BOXES

   NumPagePaireCorps: BEGIN  Content: VarPageCorps;
        Fillpattern : nopattern; Font: times; Style: Roman;  Size : 11 pt;
        VertPos: Top = Previous SAUT_PAGE . Bottom + 0.3 cm;
        HorizPos: Left = Previous SAUT_PAGE . Left;  Height:   1.6 cm;
        END;
  PageCorps: BEGIN Width: 14 cm; Height: 25 cm;
        VertPos: Top = Enclosing . Top + 2 cm;   HorizPos: Left = Enclosing . Left + 3.5 cm;
        CreateAfter (NumPagePaireCorps);
        END;

  SepCopyBox: BEGIN  Depth: Creator - 1;
        HorizPos: Left = Previous . Right + 0.2; { [ : ] }    VertPos: Top = Creator . Top;      Width: 0.3;
        Content: Text ':';
        END;
  ReferentCopyBox: BEGIN  Depth: Creator - 1;
        HorizPos: Left = Previous SepCopyBox . Right + 0.2;
        VertPos: Top = Creator . Top;
        Copy(Referent);
        END;
  SepTypeReferent: BEGIN  Content: Text ':';
        HorizPos: Left = Previous ConceptType . Right + 0.2; { [ : ] }      Width: 0.3;
        END;


  RectangularFrame: BEGIN  Content: Graphics 'R';  Depth: Creator + 1;
        Indent: 0;  BackGround: Creator =;
        FillPattern: backgroundcolor; { -> filled; with "Creator =" not sure }
        HorizPos: Left = Creator . Left - 0.3;     VertPos: Top = Creator . Top - 0.2;
        Width: Creator . Width + 0.6;      Height: Creator . Height + 0.4;
        END;
  InclRectangularFrame: BEGIN  Content: Graphics 'R';  Depth: Creator + 1;
        Indent: 0;  BackGround: Creator =;
        FillPattern: backgroundcolor; { -> filled; with "Creator =" not sure }
        HorizPos: Left = Creator . Left;       VertPos: Top = Creator . Top - 0.1;
        Width: Creator . Width;      Height: Creator . Height + 0.2;
        END;
  RefRectangularFrame: BEGINContent: Graphics 'R';  Depth: Creator + 1;
        Indent: 0;  BackGround: Creator =; FillPattern: backgroundcolor;
        LineStyle: Dashed; {Dotted;} { ForeGround: Black; doesn't work }
        HorizPos: Left = Creator . Left - 0.3;       VertPos: Top = Creator . Top - 0.2;
        Width: Right = Previous ReferentCopyBox . Right + 0.4;       Height: Creator . Height + 0.4;
        END;
```

```
RectangleWithShade: BEGIN  Content: Graphics 'R';  Depth: Creator + 1;
        Indent: 0;  BackGround: Creator =;  FillPattern: backgroundcolor;     LineStyle: Solid;
        HorizPos: Left = Creator . Left - 0.3;       VertPos: Top = Creator . Top - 0.2;
        Width:  Creator . Width + 0.6;      Height: Creator . Height + 0.4;
        END;
InclRectangleWithShade: BEGIN  Content: Graphics 'R';  Depth: Creator + 1;
        Indent: 0;  BackGround: Creator =;  FillPattern: backgroundcolor;     LineStyle: Solid;
        HorizPos: Left = Creator . Left - 0.6;       VertPos: Top = Creator . Top - 0.2;
        Width:  Creator . Width + 0.9;     Height: Creator . Height + 0.4;
        END;
RefRectWithShade: BEGIN  Content: Graphics 'R';  Depth: Creator + 1;
        Indent: 0;  BackGround: Creator =;  FillPattern: backgroundcolor;
        LineStyle: Dashed; { LineStyle: Solid;}
        HorizPos: Left = Creator . Left - 0.3;    VertPos: Top = Creator . Top - 0.2;
        Width:  Right = Previous ReferentCopyBox . Right + 0.4;    Height: Creator . Height + 0.4;
        END;


RectangularShade: BEGIN  Content: Graphics 'R';  Depth: Creator + 2;
        Background: Black; Foreground: Black; FillPattern: backgroundcolor;  LineStyle: Solid;
        HorizPos: Left = Creator . Left - 0.1;   VertPos: Top = Creator . Top;
        Width:  Creator . Width + 0.6;  Height: Creator . Height + 0.4;
        END;
InclRectangularShade: BEGIN  Content: Graphics 'R';  Depth: Creator + 2;
        Background: Black; Foreground: Black; FillPattern: backgroundcolor;   LineStyle: Solid;
        HorizPos: Left = Creator . Left - 0.4;  VertPos: Top = Creator . Top;
        Width:  Creator . Width + 0.9;   Height: Creator . Height + 0.4;
        END;
RefRectangularShade: BEGIN  Content: Graphics 'R';  Depth: Creator + 2;
        Background: Black; Foreground: Black; FillPattern: backgroundcolor;
        HorizPos: Left = Creator . Left - 0.1;
        LineStyle: Solid;
        VertPos: Top = Creator . Top;
        Width:  Right = Previous ReferentCopyBox . Right + 0.4;
        Height: Creator . Height + 0.4;
        END;

RoundedFrame: BEGIN  Content: Graphics 'C';  Depth: Enclosing + 1;
        Indent: 0;  BackGround: Creator =;  FillPattern: backgroundcolor;
        HorizPos: Left = Creator . Left - 0.3; VertPos: Top = Creator . Top - 0.2;
        Width:  Creator . Width + 0.6;  Height: Creator . Height + 0.4;
        END;
RefRoundedFrame: BEGIN  Content: Graphics 'C';  Depth: Enclosing + 1;
        LineStyle: Dashed;  BackGround: Creator =;  FillPattern: Creator =;  Indent: 0;
        HorizPos: Left = Creator . Left - 0.3;  VertPos: Top = Creator . Top - 0.2;
        Width:  Right = Previous ReferentCopyBox . Right + 0.4;  Height: Creator . Height + 0.4;
        END;

RoundedWithShade: BEGIN  Content: Graphics 'C';  Depth: Creator + 1;
        LineStyle: Solid;  BackGround: Creator =; FillPattern: backgroundcolor;  Indent: 0;
        HorizPos: Left = Creator . Left - 0.3;  VertPos: Top = Creator . Top - 0.2;
        Width:  Creator . Width + 0.6;   Height: Creator . Height + 0.4;
        END;
RefRoundedWithShade:  BEGIN  Content: Graphics 'C';  Depth: Creator + 1;
        Indent: 0;  LineStyle: Dashed;  { LineStyle: Solid;}
        BackGround: Creator =;   FillPattern: backgroundcolor;
        HorizPos: Left = Creator . Left - 0.3;  VertPos: Top = Creator . Top - 0.2;
        Width:  Right = Previous ReferentCopyBox . Right + 0.4;   Height: Creator . Height + 0.4;
        END;
RoundedShade: BEGIN  Content: Graphics 'C';  Depth: Creator + 2;
        Background: Black;  Foreground: Black;  FillPattern: backgroundcolor;  LineStyle: Solid;
        HorizPos: Left = Creator . Left - 0.1;   VertPos: Top = Creator . Top;
        Width: Creator . Width + 0.6;   Height: Creator . Height + 0.4;
        END;
```

```
 RefRoundedShade: BEGIN  Content: Graphics 'C';  Depth: Creator + 2;
            Background: Black;  Foreground: Black;  FillPattern: backgroundcolor;  LineStyle: Solid;
            HorizPos: Left = Creator . Left - 0.1;   VertPos: Top = Creator . Top;
            Width:  Right = Previous ReferentCopyBox . Right + 0.4;    Height: Creator . Height + 0.4;
            END;


 NSC_RTbox : BEGIN  Content: Text 'Necessary and sufficient conditions for ';
            HorizPos: Left = Enclosing . Left;   VertPos: Top = Previous . Bottom;
            END;
 NSC_CTbox : BEGIN  Content: Text 'Necessary and sufficient conditions for ';
            HorizPos: Left = Enclosing . Left;   VertPos: Top = Previous . Bottom;
            END;
 SC_CTbox :  BEGIN  Content: Text 'Sufficient conditions for ';
            HorizPos: Left = Enclosing . Left;   VertPos: Top = Previous . Bottom;
            END;
 NC_CTbox :  BEGIN  Content: Text 'Necessary conditions for ';
            HorizPos: Left = Enclosing . Left;   VertPos: Top = Previous . Bottom;
            END;
 TC_CTbox :  BEGIN  Content: Text 'Typical conditions for ';
            HorizPos: Left = Enclosing . Left;   VertPos: Top = Previous . Bottom;
            END;
 U_CTbox  :  BEGIN  Content: Text 'Unspecified conditions for ';
            HorizPos: Left = Enclosing . Left;  VertPos: Top = Previous . Bottom;
            END;
 ND_CTbox :  BEGIN  Content: Text 'No more a definition : ';
            HorizPos: Left = Enclosing . Left;   VertPos: Top = Previous . Bottom;
            END;

 LeftParBox: BEGIN  Content: Text '(';
            HorizPos: Left = Previous . Right +0.2;  VertPos: Top = Previous . Top;
            END;
 RightParBox: BEGIN  Content: Text ') are ';
            HorizPos: Left = Previous . Right +0.2;  VertPos: Top = Previous . Top;
            END;


RULES

 CG: BEGIN  Indent: 0;  Adjust: VMiddle; Gather: Yes;  Visibility: 7;
            PageBreak: No; LineBreak: No; LineWeight: 1 pt; LineStyle: Solid;
            HorizPos: Left = Enclosing . Left;  VertPos: Top = Previous . Bottom + 1;
            END;
 CGgraph: BEGIN  Indent: 0;  Adjust: VMiddle; Gather: Yes;  Visibility: 7;
            PageBreak: No; LineBreak: No; LineWeight: 1 pt; LineStyle: Solid;
            HorizPos: Left = Enclosing . Left;   VertPos: Top = Previous . Bottom + 0.6;
            Page(PageCorps);
            END;
 ConceptInclusion:   Visibility: 7;
 TheIncludedConcept: BEGIN Visibility: 7;  HorizPos : Left = Enclosing . Left + 4;  END;
 SingleConcept: BEGIN   Visibility: 7;   VertPos : Top = Previous . Top + 0.1 {cm UserSpecified}; END;

 Concept: BEGIN    Visibility: 7;
            VertPos : Top = Enclosing . Top + 0.1 {cm UserSpecified};
            HorizPos : Left = Enclosing . Left + 0.5 {cm UserSpecified};
            END;
 Referent: BEGIN
            HorizPos : Left = Previous . Right + 0.1; { [ : ]  UserSpecified->loop}
            CreateBefore(SepTypeReferent);
            END;
 CGReferent: BEGIN Depth: Enclosing - 2;
            VertPos : Top = Previous . Bottom + 0.2;   HorizPos: Left= Previous . Left;
            END;
 DescriptionReferent: BEGIN  Depth: Enclosing - 2;
```

```
            VertPos : Top = Previous . Bottom + 0.2;  HorizPos: Left= Previous . Left;
            END;
 RelationType: BEGIN    Visibility: 10;
            VertPos: Top = Enclosing . HMiddle;  HorizPos: Left = Enclosing . VMiddle;
            END;

 LinkToOrig:  Depth : Enclosing + 2;
 LinkToDest:  Depth : Enclosing + 2;
 OtherLinkToOrig: Depth : Enclosing + 2;
 OtherLinkToDest: Depth : Enclosing + 2;

LinkToOrig_Mark: BEGIN   Visibility: 10; line; Adjust: VMiddle;
            HorizPos : VMiddle = Previous . VMiddle;
            VertPos  : HMiddle = Previous . HMiddle;
            END;
 LinkToDest_Mark: BEGIN   Visibility: 10; line; Adjust: VMiddle;
            HorizPos : VMiddle = Previous . VMiddle;
            VertPos  : HMiddle = Previous . HMiddle;
            END;
 GRAPHIQUE: BEGIN    Visibility: 10;
            Height: Enclosing . Height;Width: Enclosing . Width;
            HorizPos : Left = Enclosing . Left;   VertPos : Top = Enclosing . Top;
            END;

 TypeDefinition: BEGIN Indent: 0;  Adjust: VMiddle; Gather: Yes; Visibility: 7;
            PageBreak: No; LineBreak: No; LineWeight: 1 pt; LineStyle: Solid;
            HorizPos: Left = Enclosing . Left;  VertPos: Top = Previous . Bottom + 1;
            Page(PageCorps);
            END;
 DefinedType: BEGIN  Visibility: 7;
            VertPos : Top = Previous . Top;   HorizPos: Left = Previous AnyBox . Right + 0.1;
            END;
 LambdaVar : BEGIN  Visibility: 7;   CreateBefore(LeftParBox);
            VertPos : Top = Previous . Top;  HorizPos: Left = Previous . Right + 0.1;
            CreateAfter(RightParBox);
            END;
 DefBody : BEGIN  Depth: Enclosing - 2;
            VertPos : Top = Previous . Bottom + 0.2;      HorizPos: Left= Enclosing . Left  + 1;
            END;


ATTRIBUTES

  OrigNode :   BEGIN  VertPos : Top  = refered . HMiddle;          HorizPos: Left = refered . VMiddle;  END;
  DestNode   : BEGIN  Height : Bottom = refered . HMiddle;        Width  : Right  = refered . VMiddle;  END;

  OrigLink_Center : BEGIN  VertPos : Top = refered . HMiddle;     HorizPos : Left = refered . VMiddle;  END;
  DestLink_Center : BEGIN  Height : Bottom = refered . HMiddle; Width  : Right  = refered . VMiddle;  END;

  OrigLink_Top    : BEGIN  VertPos : Top = refered . Top;            HorizPos : Left  = refered . VMiddle;   END;
  DestLink_Top   : BEGIN  Height : Bottom = refered . Top;        Width : Right = refered . VMiddle;    END;

  OrigLink_Bottom : BEGIN  VertPos : Top = refered . Bottom;       HorizPos : Left = refered . VMiddle;    END;
  DestLink_Bottom : BEGIN  Height : Bottom = refered . Bottom;  Width  : Right  = refered . VMiddle;  END;

  OrigLink_Right  : BEGIN  VertPos : Top = refered . HMiddle;    HorizPos : Left  = refered . Right;  END;
  DestLink_Right  : BEGIN  Height : Bottom = refered . HMiddle; Width : Right = refered . Right;     END;

  OrigLink_Left   : BEGIN  VertPos : Top = refered . HMiddle;     HorizPos : Left = refered . Left;    END;
  DestLink_Left   : BEGIN  Height : Bottom = refered . HMiddle;  Width  : Right  = refered . Left;    END;


  ConceptFrame(Concept) = Rectangular:
            CreateAfter(RectangularFrame);
```

```
  ConceptFrame(Concept) = RectangularShaded: BEGIN
          CreateAfter(RectangleWithShade); CreateAfter(RectangularShade); END;
  ConceptFrame(Concept) = Rounded:
          CreateAfter(RoundedFrame);
  ConceptFrame(Concept) = RoundedShaded:   BEGIN
          CreateAfter(RoundedWithShade); CreateAfter(RoundedShade);  END;

  ConceptFrame(ConceptInclusion) = Rectangular:
          CreateAfter(InclRectangularFrame);
  ConceptFrame(ConceptInclusion) = RectangularShaded: BEGIN
          CreateAfter(InclRectangleWithShade);CreateAfter(InclRectangularShade);END;
  ConceptFrame(ConceptInclusion) = Rounded:
          CreateAfter(RoundedFrame);
  ConceptFrame(ConceptInclusion) = RoundedShaded:   BEGIN
          CreateAfter(RoundedWithShade); CreateAfter(RoundedShade); END;

  ConceptFrame(ConceptReference) = Rectangular:
          CreateAfter(RefRectangularFrame);
  ConceptFrame(ConceptReference) = RectangularShaded: BEGIN
          CreateAfter(RefRectWithShade);CreateAfter(RefRectangularShade);END;
  ConceptFrame(ConceptReference) = Rounded:
          CreateAfter(RefRoundedFrame);
  ConceptFrame(ConceptReference) = RoundedShaded:   BEGIN
          CreateAfter(RefRoundedWithShade); CreateAfter(RefRoundedShade); END;

  RelationFrame(RelationType) = Rounded:
          CreateAfter(RoundedFrame);
  RelationFrame(RelationType) = RoundedShaded:   BEGIN
          CreateAfter(RoundedWithShade); CreateAfter(RoundedShade); END;


  DefKindCopy = NSC: BEGIN CreateBefore(NSC_CTbox);
                  HorizPos: Left= Previous AnyBox . Right + 0.1; END;
  DefKindCopy = NC : BEGIN CreateBefore(NC_CTbox);
                  HorizPos: Left= Previous AnyBox . Right + 0.1; END;
  DefKindCopy = SC : BEGIN CreateBefore(SC_CTbox);
                  HorizPos: Left= Previous AnyBox . Right + 0.1; END;
  DefKindCopy = TC : BEGIN CreateBefore(TC_CTbox);
                  HorizPos: Left= Previous AnyBox . Right + 0.1; END;
  DefKindCopy = RD : BEGIN CreateBefore(NSC_RTbox);
                  HorizPos: Left= Previous AnyBox . Right + 0.1; END;
  DefKindCopy = UD : BEGIN CreateBefore(U_CTbox);
                  HorizPos: Left= Previous AnyBox . Right + 0.1; END;
  DefKindCopy = ND : BEGIN CreateBefore(ND_CTbox);
                  HorizPos: Left= Previous AnyBox . Right + 0.1; END;

END
```

4.1.3 Modèle d'interface

Un modèle d'interface permet de spécifier quelles fonctions C doivent être appelées lorsqu'un utilisateur effectue certains événements (e.g. sélection, création, destruction, sauvegarde, chargement) sur des EDs de certains types. Voici donc ci-dessous les événements traités par CGKAT pour des EDs composant un ED GC. Chaque fonction appellée met à jour la base de CoGITo compte-tenu de l'action qui vient d'être réalisée.

```
APPLICATION CG;

ELEMENTS  { events on DEs }

  TEXTE:   BEGIN  StdElemSelect.Post: TextSelectPost;
           END;
  ConceptType:BEGIN StdElemSelect.Post: ConceptTypeSelPost;
                StdElemDelete.Pre : TypeDelPre;
                StdElemTextModify.Pre: TypeTextModPre;
           END;
  Referent:   BEGIN StdElemSelect.Post: ReferentSelPost;
                StdElemDelete.Pre : ReferentDelPre;
           END;
  Variable:   BEGIN StdElemNew.Pre    : VariableNewPre;
                StdElemTextModify.Post: VarTextModPost;
           END;
  Individual: BEGIN StdElemNew.Pre    : IndividualNewPre;
                StdElemDelete.Pre : IndividualDelPre;
                StdElemTextModify.Pre: TypeTextModPre;
           END;

  Concept:    BEGIN StdElemNew.Post   : ConceptNewPost;
                StdElemSelect.Post: ConceptSelectPost;
                StdElemDelete.Pre : ConceptDelPre;
                StdElemPaste.Post : ConceptPastePost;
              { StdElemRead.Post  : ConceptReadPost; {  !UseBCGCT }
           END;
  ConceptInclusion:
           BEGIN StdElemNew.Post   : ConceptInclNewPost;
                StdElemDelete.Pre : ConceptDelPre;
                StdElemPaste.Post : ConceptIncPastePost; { to do }
           END;
  ConceptReference:
           BEGIN { StdElemNew.Post   : ConceptRefNewPost; }
                StdElemDelete.Pre : ConceptDelPre;
           END;

  Relation:   BEGIN StdElemNew.Post   : RelationNewPost;
                StdElemDelete.Pre : RelationDelPre;
                StdElemPaste.Post : RelationPastePost;
           END;
 RelationType:BEGIN StdElemDelete.Pre : RelationDelPre;
                StdElemTextModify.Pre: TypeTextModPre;
           END;

  LinkToOrig:      StdElemDelete.Pre : RelationDelPre;
  LinkToDest:      StdElemDelete.Pre : RelationDelPre;
  GRAPHIQUE:  StdElemDelete.Pre : RelationDelPre;

  RelationAndConcept:
           BEGIN StdElemNew.Pre    : RelConceptNewPre;
                 StdElemDelete.Pre : RelConceptDelPre;
           END;
  ConceptOrRelation:StdElemMenu.Pre   : ElementMenuPre;
```

```
  CGgraph:    BEGIN StdElemNew.Pre    : CGgraphNewPre;
              { StdElemRead.Pre   : CGgraphReadPre; { !UseBCGCT }
                StdElemRead.Post  : CGgraphReadPost;
                StdElemSave.Post  : CGgraphSavePost;
                StdElemPaste.Post : CGgraphPastePost;
                StdElemDelete.Pre : CGgraphDelPre;
            END;
  DefBody:    BEGIN StdElemNew.Pre    : CGgraphNewPre;
              { StdElemRead.Pre   : CGgraphReadPre; { !UseBCGCT }
                StdElemRead.Post  : CGgraphReadPost;
                StdElemSave.Post  : CGgraphSavePost;
                StdElemPaste.Post : CGgraphPastePost;
                StdElemDelete.Pre : CGgraphDelPre;
            END;
  TypeDefinition:
            BEGIN StdElemNew.Pre    : TypeDefNewPre;
                StdElemNew.Post   : TypeDefNewPost;
                StdElemDelete.Pre : CGgraphDelPre;
            END;


ATTRIBUTES  { events on DE attributes }

  Comment:   BEGIN  StdAttrModify.Post: CommentModPost;
                END;
  ArrowOnLink1:  BEGIN  StdAttrModify.Post: ArrowOnLink1ModPost;
                    StdAttrCreate.Post: ArrowOnLink1ModPost;
                END;
  OrigRelLink1_Pos: BEGIN  StdAttrModify.Post: OrigLink1PosModPost;
                    StdAttrCreate.Post: OrigLink1PosModPost;
                END;
  DestRelLink1_Pos: BEGIN  StdAttrModify.Post: DestLink1PosModPost;
                    StdAttrCreate.Post: DestLink1PosModPost;
                END;
  ArrowOnLink2:  BEGIN  StdAttrModify.Post: ArrowOnLink2ModPost;
                    StdAttrCreate.Post: ArrowOnLink2ModPost;
                END;
  OrigRelLink2_Pos: BEGIN  StdAttrModify.Post: OrigLink2PosModPost;
                    StdAttrCreate.Post: OrigLink2PosModPost;
                END;
  DestRelLink2_Pos: BEGIN  StdAttrModify.Post: DestLink2PosModPost;
                    StdAttrCreate.Post: DestLink2PosModPost;
                END;
  ChangeIntoDefinition:BEGIN  StdAttrModify.Post: ChangeIntoDefModPost;
                    StdAttrCreate.Post: ChangeIntoDefModPost;
                END;
  DefKind:        BEGIN  StdAttrModify.Post: DefKindModPost;
                END;
  Completed:       BEGIN  StdAttrModify.Post: CompletedModPost;
                END;

END
```

4.2 Modèles pour un GC et certaines méta-informations

4.2.1 Modèle structurel

```
STRUCTURE CGRepr;   DEFPRES CGReprP;    { Structural model for a (DE of type) CGRepr }
STRUCT
   CGRepr = BEGIN
                  User = TEXT;  { Author of the CG }              Viewpoint = TEXT;
                  Documentation = BEGIN
                                      CreationDate  = TEXT;     SourceAuthor = TEXT;
                                      ContextOfUse = TEXT;      Comment = TEXT;
                                  END;
                ? GeneratedCG = CG;  { representation proposed by CGKAT to the user }
                  TheRepr = CG;  { the representation chosen by the user }
               END;
EXCEPT   { exceptions to standard edition rules for DEs }
   TheRepr : NoSelect,NoMove,NoResize;   { this DE won't be selected, moved and resized --> instead }
END;                                     {     its structural parent will be selected, moved or resized }
```

4.2.2 Modèle de présentation

```
PRESENTATION CGRepr;   {presentation associated to DEs which have a representation in a cg base }
VIEWS   MainView;

BOXES

 SepBox: BEGIN   Content: Text '---------------------- ';
   HorizPos: Left = Enclosing . Left;       VertPos:  Top = Previous . Bottom + 1;
  END;
 UserBox: BEGIN    Content: Text 'User: ';  Visibility: 5;
   HorizPos: Left =  Enclosing . Left;      VertPos:  Top  = Previous . Bottom;
  END;
 ViewBox: BEGIN    Content: Text 'Viewpoint: ';
   HorizPos: Left =  Enclosing . Left;      VertPos:  Top  = Previous . Bottom;
  END;

 CreationDateBox: BEGIN    Content: Text 'Creation date: ';
   HorizPos: Left =  Enclosing . Left;      VertPos:  Top = Enclosing . Top;
  END;
 SourceBox: BEGIN    Content: Text 'Source (e.g. expert): ';
   HorizPos: Left =  Enclosing . Left;      VertPos:  Top  = Previous . Bottom;
  END;
 ContextOfUseBox: BEGIN  Content: Text 'Context of use (e.g. by the expert): ';
   HorizPos: Left =  Enclosing . Left;      VertPos:  Top  = Previous . Bottom;
  END;
 CommentBox: BEGIN   Content: Text 'Comment: ';
   HorizPos: Left =  Enclosing . Left;      VertPos:  Top  = Previous . Bottom;
  END;
 GenerReprBox: BEGIN   Content: Text 'Synthesis of subparts: ';
   HorizPos: Left =  Enclosing . Left;      VertPos:  Top  = Previous . Bottom + 0.6;
  END;


RULES
  CGRepr: BEGIN  Visibility: 9;   Font: helvetica;   Size : 6;     Indent: 0;    LineSpacing: 1.1;
     HorizPos: Left = Enclosing . Left;          VertPos:  Top  = Previous . Bottom + 2;
     END;
  User: BEGIN  Visibility: 5;  {CreateBefore(SepBox);}  CreateBefore(UserBox);
      HorizPos: Left= Previous UserBox . Right;      VertPos:  Top = Previous UserBox . Top;
      END;
```

```
  View: BEGIN  Visibility: 9;  CreateBefore(ViewBox);
      HorizPos: Left = Previous . Right;          VertPos:  Top  = Previous . Top;
      END;
  Documentation: BEGIN  Visibility: 5;
      HorizPos: Left = Enclosing . Left + 1;     VertPos:  Top  = Previous . Bottom;
      END;
  CreationDate: BEGIN  CreateBefore(CreationDateBox);
      HorizPos: Left = Previous . Right;          VertPos:  Top  = Previous . Top;
     END;
  Source: BEGIN  CreateBefore(SourceBox);
       HorizPos: Left = Previous . Right;         VertPos:  Top  = Previous . Top;
       END;
  ContextOfUse: BEGIN  CreateBefore(ContextOfUseBox);
       HorizPos: Left = Previous . Right;         VertPos:  Top  = Previous . Top;
       END;
  Comment: BEGIN  CreateBefore(CommentBox);
       HorizPos: Left = Previous . Right;         VertPos:  Top  = Previous . Top;
       END;
  GeneratedCG:  BEGIN  Visibility: 5;  CreateBefore(GenerReprBox);
      HorizPos: Left = Enclosing . Left;          VertPos:  Top  = Previous  . Bottom + 1;
      END;
   TheRepr: BEGIN  Visibility: 9;  Indent: 0; { Gather: Yes; }
      HorizPos: Left = Enclosing . Left;          VertPos:  Top  = Previous . Bottom + 1;
      CreateAfter(SepBox);
      END;
END
```

4.2.3 Modèle d’interface

```
APPLICATION CGRepr;
ELEMENTS
  CGRepr: BEGIN
              StdElemNew.Pre : CGReprNewPre;
              StdElemNew.Post: CGReprNewPost;
             END;
  View:      StdElemTextModify.Post: ViewTextModPost;
  User:      StdElemTextModify.Post: UserTextModPost;
END
```

4.3 Modèles pour l'indexation d'éléments de documents

4.3.1 Modèle structurel d'extension

```
STRUCTURE  EXTENSION CGExtRepr;     DEFPRES CGExtReprP;     { Structural extension model }

ATTR   { These attributes may be created and associated to any DE }
   To_CGReprOnElem = REFERENCE(CGRepr);   { any DE -->(Representation)--> CGRepr }
   To_CGNoteOnElem = REFERENCE(CGRepr);   { any DE -->(Annotation)--> CGRepr }

   To_CGReprsOnElem = REFERENCE(CGReprsOnElem);
      { any DE  -->(Representations)-->  CGReprsOnElem=LIST  OF (CGReprOnElem=CGRepr)  }
   To_CGcReprsOnElem = REFERENCE(CGcReprsOnElem);
      { any DE  -->(Representations)-->  CGReprsOnElem=LIST  OF (CGcReprOnElem=CGRepr)  }
   To_CGNotesOnElem = REFERENCE(CGNotesOnElem);

   To_Descriptor= REFERENCE(ANY);  {any DE  --> Descriptor including main information of a CGRepr}
   To_Anything = REFERENCE(ANY); {any DE --> any DE}

   SessionColored = REFERENCE(ANY);  { A way for CGKAT to handle various colorations for indexed
                                       DEs when a user access (read --> know) their indexations }
   ForegroundColor  = black,grey,white,yellow,green,cyan,blue,violet,purple,magenta,red,orange;
   BackgroundColor = black,grey,white,yellow,green,cyan,blue,violet,purple,magenta,red,orange;
                        { --> colors of DEs may be specifed by users or programs using attributes }

STRUCT   Repres_Mark = PAIR;   { A pair of representation marks is useful for isolating consecutive DEs
          in a document, or a substring in a textual element; then, since the above attributes may be
          associated to such a pair of marks, consecutive DEs or words may be indexed }
EXTENS   ROOT + (Repres_Mark);              { --> representation marks may be inserted around any DE }
          CG - (Repres_Mark);    CGRepr - (Repres_Mark);      { --> except around a CG and a CGRepr }

ASSOC   { --> kinds of DEs which are not in the main tree of a document but which may be associated to
           the DEs of the main tree in a document --> for each kind, a new tree root DE may be created in
           the document for collecting the DEs of this kind (the tree root DE is a list)  }
   CGElem = CG; { --> a tree root DE named CGElems (= LIST OF (CGElem)) may be created  }
   CGReprsOnElem  = CGReprList;   { list of CGRepr where the CG is a CGgraph or a type definition }
   CGcReprsOnElem = CGReprList; { list of CGRepr where the CG is a single concept }
   CGNotesOnElem = CGReprList;   { list of CGRepr for noting any DE, not for representing it }

UNITS   aCG = CG;  { --> where units are allowed, a CG may be built }

EXCEPT    { active reference attributes allow users to do hypertext navigation: when a DE has such an
             attribute, a double clic on this DE moves the focus to the refered DE }
   To_CGReprOnElem : ActiveRef;      To_CGNoteOnElem  : ActiveRef;
   To_CGReprsOnElem : ActiveRef;    To_CGNotesOnElem : ActiveRef;
   To_CGcReprsOnElem : ActiveRef;  To_Descriptor : ActiveRef;   To_Anything : ActiveRef;
END
```

4.3.2 Modèle de présentation

```
PRESENTATION CGExtRepr;
VIEWS   MainView;
PRINT   MainView, CGReprsOnElems, CGNotesOnElems;
COUNTERS
  CptPageCorps:  Rank of Page(Texte_integral) Init FirstPageNumber;
VAR
  VarPageCorps:  Text '-';  { Value (CptPageCorps, Arabic); }
BOXES
 NumPagePaireCorps: BEGIN  Content: VarPageCorps;
            Fillpattern : nopattern; Font: times; Style: Roman; Size : 11 pt;
            VertPos: Top = Previous SAUT_PAGE . Bottom + 0.3 cm;
            HorizPos: Left = Previous SAUT_PAGE . Left;  Height:   1.6 cm;
            END;
 PageCorps: BEGIN Width: 14 cm; Height: 25 cm;
            VertPos: Top = Enclosing . Top + 2 cm;
            HorizPos: Left = Enclosing . Left + 3.5 cm;
            CreateAfter (NumPagePaireCorps);
            END;

 WhiteBox: BEGIN   Content: Text ' ';
            HorizPos: Left = Previous . Left;   VertPos: Top = Previous . Bottom;
            END;
 CGNotesOnElemBox: BEGIN     Size : 8;  Content: Text 'CGs notes on the document element';
            VertPos: Top = Previous . Bottom + 50;    HorizPos: Left = Enclosing . Left + 5;
            END;
 CGElemsBox: BEGIN    Size : 8;  Content: Text 'CG elements for the document';   END;

 CGReprsOnElemBox: BEGIN     Size : 8;     Content: Text 'Representations of the document element';
            VertPos: Top = Previous . Bottom + 50;    HorizPos: Left = Enclosing . Left + 5;
            END;
 CGReprsOnElemBox2: BEGIN   Visibility: 5;
            Content: Text '(double click on any of these representations to retrieve the element they represent)';
            HorizPos: Left = Enclosing . Left + 2;    VertPos: Top = Previous . Bottom + 0.5;
            END;
 CGReprsOnElemBox3: BEGIN  Visibility: 5;
            Content: Text '(apply "Search -> Reference to..." on a concept to retrieve the CGs that use it)';
            HorizPos: Left = Enclosing . Left + 2;   VertPos: Top = Previous . Bottom;
            END;
 CGReprsOnElemBox4: BEGIN  Visibility: 5;
            Content: Text '(the dashed concepts or relations refer to identical concepts or connections between
concepts)';
            HorizPos: Left = Enclosing . Left + 2;   VertPos: Top = Previous . Bottom;
            END;

RULES
 CGReprsOnElems: BEGIN  Visibility: 9;  Font: helvetica;    Size : 6;
   Indent: 0;    Justify: Yes;   Hyphenate: Yes;     LineSpacing: 1.1;
   Width: Enclosing.Width - 1;   HorizPos:Left = Enclosing.Left + 0.5;     Page(PageCorps);
   END;
 CGReprsOnElem:  BEGIN   Visibility: 9;     CreateBefore(CGReprsOnElemBox);
  {CreateBefore(CGReprsOnElemBox2);     CreateBefore(CGReprsOnElemBox3);
   CreateBefore(CGReprsOnElemBox4);}    CreateBefore(WhiteBox);
   Width: Enclosing . Width;     HorizPos: Left = Enclosing . Left;    VertPos: Top = Previous . Bottom;
   END;

 CGcReprsOnElems: BEGIN  Visibility: 9;  Font: helvetica;    Size : 6;
   Indent: 0;    Justify: Yes;   Hyphenate: Yes;     LineSpacing: 1.1;
   Width: Enclosing.Width - 1;   HorizPos:Left = Enclosing.Left + 0.5;     Page(PageCorps);
   END;
 CGcReprsOnElem:  BEGIN  Visibility: 9;  CreateBefore(CGReprsOnElemBox);
  {CreateBefore(CGReprsOnElemBox2);     CreateBefore(CGReprsOnElemBox3);
   CreateBefore(CGReprsOnElemBox4);}    CreateBefore(WhiteBox);
```

```
    Width: Enclosing . Width;     HorizPos: Left = Enclosing . Left;    VertPos: Top = Previous . Bottom;
    END;

  CGNotesOnElems: BEGIN  Visibility: 9;  Font: helvetica;    Size : 6;
    Indent: 0;    Justify: Yes;   Hyphenate: Yes;     LineSpacing: 1.1;
    Width: Enclosing.Width - 1;   HorizPos:Left = Enclosing.Left + 0.5;     Page(PageCorps);
    END;
  CGNotesOnElem:  BEGIN  Visibility: 9;   CreateBefore(CGNotesOnElemBox);
    Width: Enclosing . Width;     HorizPos: Left = Enclosing . Left;     VertPos: Top = Previous . Bottom;
    END;

  CGElems: BEGIN  Visibility: 9;   CreateBefore(CGElemsBox);
    Font: helvetica;    Size : 6;    Indent: 0;    Justify: Yes;   Hyphenate: Yes;     LineSpacing: 1.1;
    Width: Enclosing.Width - 1;   HorizPos:Left = Enclosing.Left + 0.5;
    END;
  CGElem:  BEGIN  Visibility: 9;
    Width: Enclosing . Width;   HorizPos: Left = Enclosing . Left;    VertPos: Top = Previous . Bottom + 1;
    END;

  First Repres_Mark: BEGIN   Content: Text '\253';     Foreground: Red;     UnderLine: Underlined;
    VertPos: Top = Previous NOT First Repres_Mark . Bottom;
    HorizPos: Left = Previous NOT Second Repres_Mark . Left + 0.8;
    END;
  Second Repres_Mark: BEGIN  Content: Text '\273';     Foreground: Red;     UnderLine: Underlined;
    VertPos : Top = Previous NOT Second Repres_Mark . Bottom;
    HorizPos : Left = Previous NOT First Repres_Mark . Left + 0.8;
    END;

  TEXTE: Depth: Enclosing - 1;


ATTRIBUTES

    To_CGReprOnElem  : UnderLine: Underlined;
    To_CGNoteOnElem  : UnderLine: Underlined;
    To_CGReprsOnElem : UnderLine: Underlined;
    To_CGcReprsOnElem: UnderLine: Underlined;
    To_CGNotesOnElem : UnderLine: Underlined;
    To_Descriptor    : UnderLine: Underlined;
    To_Anything      : UnderLine: Underlined;

    Elem_Depth      : Depth     : Elem_Depth;
    Elem_Visibility : Visibility: Elem_Visibility;

    DisplayHelp(CGReprsOnElem) = 1: BEGIN CreateAfter(WhiteBox);  CreateAfter(CGReprsOnElemBox2);
                                          CreateAfter(CGReprsOnElemBox3);
                                          CreateAfter(CGReprsOnElemBox4);
                                    END;

    ForegroundColor = white      : Foreground : White;
    ForegroundColor = grey       : Foreground : Grey;
    ForegroundColor = black      : Foreground : Black;
    ForegroundColor = red        : Foreground : Red;
    ForegroundColor = yellow     : Foreground : Yellow;
    ForegroundColor = green      : Foreground : Green;
    ForegroundColor = cyan       : Foreground : Cyan;
    ForegroundColor = blue       : Foreground : Blue;
    ForegroundColor = violet     : Foreground : Violet;
    ForegroundColor = purple     : Foreground : Purple;
    ForegroundColor = magenta : Foreground : Magenta;
    ForegroundColor = orange     : Foreground : Orange;
    BackgroundColor = white      :  BEGIN Background: White;   FillPattern: backgroundcolor; END;
    BackgroundColor = grey       :  BEGIN Background: Grey;    FillPattern: backgroundcolor; END;
    BackgroundColor = black      :  BEGIN Background: Black;   FillPattern: backgroundcolor; END;
```

```
  BackgroundColor = red        :  BEGIN Background: Red;     FillPattern: backgroundcolor; END;
  BackgroundColor = yellow     :  BEGIN Background: Yellow;  FillPattern: backgroundcolor; END;
  BackgroundColor = green      :  BEGIN Background: Green;   FillPattern: backgroundcolor; END;
  BackgroundColor = cyan       :  BEGIN Background: Cyan;    FillPattern: backgroundcolor; END;
  BackgroundColor = blue       :  BEGIN Background: Blue;    FillPattern: backgroundcolor; END;
  BackgroundColor = violet     :  BEGIN Background: Violet;  FillPattern: backgroundcolor; END;
  BackgroundColor = purple     :  BEGIN Background: Purple;  FillPattern: backgroundcolor; END;
  BackgroundColor = magenta : BEGIN Background: Magenta; FillPattern: backgroundcolor; END;
  BackgroundColor = orange     :  BEGIN Background: Orange;  FillPattern: backgroundcolor; END;
END
```

4.3.3 Modèle d'interface par défaut

Le modèle suivant spécifie les fonctions C qui doivent être appelées lorsqu'un utilisateur effectue certains événements sur des EDs, quels que soient leurs types, si aucun autre schéma d'interface ne spécifie des fonctions à appeler pour ces événements sur des EDs de ces types.

```
APPLICATION EDITOR;
DEFAULT
  BEGIN
   StdElemSelect.Post       : ElemSelectPost;       { LDoc, -> ConceptInclToComplet }
   StdElemExtendSelect.Post: ExtendSelectPost;    { LDoc }
   StdElemActivate.Post    : ElemActivatePost;     { -> handling of dynamic links }

   StdElemNew.Pre           : ElemNewPre;           { UnloadCGsInElem }
   StdElemNew.Post          : ElemNewPost;          { cgReprListET }

   StdDocCreate.Post        : DocCreatePost;
   StdDocOpen.Post        : DocOpenPost;
   StdDocSave.Pre           : DocSavePre;
   StdDocSave.Post        : DocSavePost;
   StdDocClose.Pre        : DocClosePre;
   StdDocClose.Post       : DocClosePost;
  END;
END
```

4.4 Modèles pour une hiérarchie

Ces modèles sont dérivés des modèles Arbre2.S et Arbre2P.P de la bibiliothèque de Thot.

4.4.1 Modèle structurel

```
STRUCTURE Hierarchy;   DEFPRES HierarchyP;    { Structural model for a (DE of type) Hierarchy }
STRUCT
   Hierarchy (ATTR !Presentation_form = Vertical, Horizontal, Indented_List;
                    NodeFrame = Without, Rectangular, RectangularShaded, Rounded, RoundedShaded;
                    AboutNodeWidth = Fixed,Variable )
      = Sub_Hierarchy with Presentation_form?=Horizontal, AboutNodeWidth?=Variable, NodeFrame?=Without;

   Sub_Hierarchy (ATTR  NodeFrame)
      = BEGIN  Node (ATTR RefClone=REFERENCE(Node)) = LIST OF (UNIT);  {strings and images are units}
            ? Descendants = LIST OF (Sub_Hierarchy);              {any ED type may be declared as a unit}
        END;

EXCEPT   { exceptions to standard edition rules for DEs }
   Node : NoMove;          Descendants: NoMove,NoResize;
END;
```

4.4.2 Modèle de présentation

```
PRESENTATION Hierarchy;

DEFAULT
  BEGIN
  HorizRef: * . Bottom;   VertRef: * . Left;   Width: Enclosed . Width;   Height: Enclosed . Height;
  HorizPos: Left = Enclosing . Left;   VertPos: Top = Enclosing . Top;
  Visibility: Enclosing =;   Font: Enclosing =;   Style: Enclosing =;   Size: Enclosing =;
  LineSpacing: Enclosing =;   Justify: No;   Adjust: Enclosing =;   Indent: Enclosing =;   Depth: Enclosing =;
  UnderLine: Enclosing =;   Thickness: Enclosing =;   Foreground: Enclosing =;   BackGround: Enclosing =;
  FillPattern: Enclosing =;   LineWeight: Enclosing =;
  END;

BOXES
  LienVert: BEGIN   Content: Graphics '/';
            VertPos: Bottom = Next Sub_Hierarchy . Top + 0.1;   HorizPos: Left = Next Sub_Hierarchy . VMiddle;
            Height: Top = Enclosing . Top;    Width: Right = Enclosing . VMiddle;
            END;
  LienHoriz:  BEGIN   Content: Graphics '/';
            VertPos: Bottom = Enclosing . HMiddle;     HorizPos: Left = Enclosing . Left;
            Height: Top = Next Sub_Hierarchy . HMiddle;    Width: Right = Next Sub_Hierarchy . Left;
            END;
  TraitVertical:  BEGIN   Content: Graphics 'v';
            Visibility: Enclosing -1 min 5;  Width: 0.6;  Height: Previous Descendants . Height;
            VertPos: Top = Previous Descendants . Top;  HorizPos: Right = Previous Descendants . Left;
            END;

  BoiteRectangle:  BEGIN  Content: Graphics 'R';
            HorizPos: Left = Previous Node . Left - 0.3;  VertPos: Top = Previous Node . Top - 0.3;
            Width: Previous Node . Width + 0.6;   Height: Previous Node . Height + 0.6;
            Depth: Enclosing + 1;  Indent: 0;  BackGround: Creator =;  FillPattern: Creator =;
            END;
  RectangleAvecOmbre:  BEGIN  Content: Graphics 'R';
            HorizPos: Left = Previous Node . Left - 0.3;  VertPos: Top = Previous Node . Top - 0.3;
            Indent: 0;  Width: Previous Node . Width + 0.6;  Height: Previous Node . Height + 0.6;
            Depth: Creator + 1;  LineStyle: Solid;  BackGround: Creator =;  FillPattern: backgroundcolor;
            END;
```

```
  OmbreRectangle:  BEGIN  Content: Graphics 'R';
            HorizPos: Left = Previous Node . Left - 0.1;  VertPos: Top = Previous Node . Top - 0.1;
            Width: Previous Node . Width + 0.6;  Height: Previous Node . Height + 0.6;
            Depth: Creator + 3; Background: Black;  Foreground: Black;
            FillPattern: backgroundcolor;  LineStyle: Solid;
            END;

  BoiteArrondie:  BEGIN  Content: Graphics 'C';
            HorizPos: Left = Previous Node . Left - 0.3;  VertPos: Top = Previous Node . Top - 0.3;
            Width: Previous Node . Width + 0.6;  Height: Previous Node . Height + 0.6;
            Depth: Enclosing + 1;  Indent: 0;  BackGround: Creator =;  FillPattern: Creator =;
            END;
  ArrondiAvecOmbre:  BEGIN  Content: Graphics 'C';
            HorizPos: Left = Previous Node . Left - 0.3;   VertPos: Top = Previous Node . Top - 0.3;
            Width: Previous Node . Width + 0.6;  Height: Previous Node . Height + 0.6;
            Depth: Creator + 1;  LineStyle: Solid;  BackGround: Creator =;  FillPattern: backgroundcolor;
            Indent: 0;
            END;
  OmbreArrondi:  BEGIN  Content: Graphics 'C';
            HorizPos: Left = Previous Node . Left - 0.1;   VertPos: Top = Previous Node . Top - 0.1;
            Width: Previous Node . Width + 0.6;   Height: Previous Node . Height + 0.6;
            Depth: Creator + 3;  LineStyle: Solid;
            Background: Black;  Foreground: Black;   FillPattern: backgroundcolor;
            END;

  BoiteEllipse:  BEGIN  Content: Graphics 'c';
            HorizPos: Left = Previous Node . Left - 0.3;   VertPos: Top = Previous Node . Top - 0.3;
            Width: Previous Node . Width + 0.6;  Height: Previous Node . Height + 0.6;
            Depth: Enclosing + 1;  Indent: 0;  BackGround: Creator =;  FillPattern: Creator =;
            END;
  EllipseAvecOmbre:  BEGIN  Content: Graphics 'c';
            HorizPos: Left = Previous Node . Left - 0.3;  VertPos: Top = Previous Node . Top - 0.3;
            Width: Previous Node . Width + 0.6;  Height: Previous Node . Height + 0.6;
            Depth: Creator + 1;  LineStyle: Solid;  BackGround: Creator =;  FillPattern: backgroundcolor;
            Indent: 0;
            END;
  OmbreEllipse:  BEGIN  Content: Graphics 'c';
            HorizPos: Left = Previous Node . Left - 0.1;    VertPos: Top = Previous Node . Top - 0.1;
            Width: Previous Node . Width + 0.6;  Height: Previous Node . Height + 0.6;
            Depth: Creator + 3;  LineStyle: Solid;
            Background: Black;  Foreground: Black;  FillPattern: backgroundcolor;
            END;


RULES

  Hierarchy:  BEGIN
            HorizPos: Left = Enclosing . Left;  VertPos: Top = Previous . Bottom + 1;
            Adjust: VMiddle;  Indent: 0;  Gather: Yes;  PageBreak: No;
            LineBreak: No;  LineWeight: 1 pt;  LineStyle: Solid;
            END;
  Sub_Hierarchy:  BEGIN  END;
  Node:  BEGIN
            HorizPos: nil;
            VertPos: Top = Enclosing . Top + 0.3;
            END;
  Descendants: BEGIN
            VertPos: Top = Previous . Bottom; HorizPos: VMiddle = Node . VMiddle;
            Visibility: Enclosing -1 min 5;
            END;
```

```
ATTRIBUTES

  Disposition =  Liste_indent\351e:  BEGIN
            Width: Enclosing . Width;  Adjust: Left;  Indent: -3;
            END;
  Disposition(Sub_Hierarchy) = Arbre_vertical:  BEGIN
            HorizPos: Left = Previous Sub_Hierarchy . Right + 1;  VertPos: Top = Enclosing . Top + 1.1;
            CreateBefore(LienVert);
            END;
  Disposition(Sub_Hierarchy) = Arbre_horizontal:  BEGIN
            HorizPos: Left = Enclosing . Left + 2;   VertPos: Top = Previous Sub_Hierarchy . Bottom + 0.5;
            CreateBefore(LienHoriz);
            END;
  Disposition(Sub_Hierarchy) = Liste_indent\351e:  BEGIN
            HorizPos: Left = Enclosing . Left;  Width: Enclosing . Width;  VertPos: Top = Previous . Bottom;
            END;
  Disposition(Node) = Arbre_horizontal:  BEGIN
            HorizPos: Left = Enclosing . Left + 0.3;  VertPos : nil;
            END;
  Disposition(Node) = Liste_indent\351e:  BEGIN
            Width: Enclosing . Width - 0.6;
            HorizPos: Left = Enclosing . Left + 0.3;  VertPos: Top = Previous . Bottom;
            Line;  Adjust: Left; Indent: -3;
            END;
  Disposition(Descendants) = Arbre_horizontal:  BEGIN
            VertPos: HMiddle = Node . HMiddle;  HorizPos : Left = Previous . Right;
            END;
  Disposition(Descendants) = Liste_indent\351e:  BEGIN
            CreateAfter(TraitVertical);  Width: Enclosing . Width - 1;
            VertPos: Top = Previous . Bottom;  HorizPos: Left = Enclosing . Left + 1;
            END;

  RefClone : BEGIN  VertPos : Bottom = refered . Bottom;  HorizPos: Right= refered . Right;
            END;

  Largeur_noeuds(Node) = Fixe:  BEGIN  Width: 5;  Line;
            END;
  Largeur_noeuds(Node) = Variable:  Width: Enclosed . Width;

  Cadre_noeuds(Node) = Rectangle:  CreateAfter(BoiteRectangle);
  Cadre_noeuds(Node) = Rectangle_ombr\351:  BEGIN
            CreateAfter(RectangleAvecOmbre);  CreateAfter(OmbreRectangle);
            END;
  Cadre_noeuds(Node) = Arrondi:  CreateAfter(BoiteArrondie);
  Cadre_noeuds(Node) = Arrondi_ombr\351:  BEGIN
            CreateAfter(ArrondiAvecOmbre);  CreateAfter(OmbreArrondi);
            END;
  Cadre_noeuds(Node) = Ellipse:  CreateAfter(BoiteEllipse);
  Cadre_noeuds(Node) = Ellipse_ombr\351e:  BEGIN
            CreateAfter(EllipseAvecOmbre);  CreateAfter(OmbreEllipse);
            END;

END
```

4.4.3 Modèle d'interface

Nous n'avons pas débuté la répercussion dans CoGITo de la construction avec Thot de hiérarchies de types ou de GCs. Nous n'avons donc pas écrit de modèle d'interface pour les hiérarchies.

4.5 Modèles pour utiliser un langage de commandes

4.5.1 Modèle structurel

```
STRUCTURE CGRequests;   DEFPRES CGRequestsP;    { Structural model for a (DE of type) CGRequests }
STRUCT
   CGRequests = LIST OF (RequestAnswer);

   RequestAnswer  = BEGIN  Request = UnitList;
                           Answers = LIST OF (Answer=UnitList);
                    END;

   UnitList = LIST OF (AnyElement=UNIT);

UNITS    { Declaration of the kinds of EDs which may be units
           (in addition to strings, images, graphics elements and symbols) }
   StructuredElement = NATURE;      { Any kind of structured DE declared as reusable in documents }
END;
```

4.5.2 Modèle de présentation

```
PRESENTATION CGRequests;
VIEWS   MainView;
BOXES
  RequestBox:  BEGIN  Content: Text '> ';  HorizPos: Left = Enclosing . Left;   END;

RULES
  CGRequests: BEGIN  Width : Enclosing . Width; { -> cursor visibility }   END;

  RequestAnswer: BEGIN Width : Enclosing . Width; { -> no truncature }
           HorizPos: Left = Enclosing . Left;        VertPos : Top = Previous . Bottom + 1;
           END;

  Request:  BEGIN CreateBefore(RequestBox);
           Line; Indent: 0;      Width : Enclosing . Width - 6;  { -> CR + LF instead of CR }
           HorizPos: Left = Previous RequestBox . Right;      VertPos : Top  = Previous RequestBox . Top;
           END;
  Answers:  BEGIN Width : Enclosing . Width;
           HorizPos : Left = Enclosing . Left;     VertPos : Top = Previous . Bottom + 0.5;
           END;
  Answer:  BEGIN { Line; }  Indent: 0;     Width : Enclosing . Width - 3;
           HorizPos : Left = Enclosing . Left;     VertPos : Top = Previous . Bottom + 0.5;
           END;
  UnitList: BEGIN   Line;   Width : Enclosing . Width;    VertPos : Top = Previous . Bottom + 0.3;
           END;
  AnyElement:BEGIN Linebreak:Yes;   { VertPos : Top = Previous . Bottom + 0.3; }
           END;
  GRAPHIQUE: BEGIN  Width: 2;  Height: 1; END;
  SYMBOLE:    BEGIN  Width: 1;  Height: 1; END;
END
```

4.5.3 Modèle d'interface

```
APPLICATION CGRequests;
ELEMENTS
  Request:  BEGIN      StdElemSelect.Post: RequestSelectPost;
                       StdElemRead.Post:   RequestReadPost;   {when a document is loaded }
            END;
END
```

Annexe 5 Le langage de commande de CGKAT

Nous ne donnons bien sûr ici que les commandes définies dans CGKAT. Toutes les commandes accessibles depuis le shell sont également accessibles depuis CGKAT à condition de les préfixer par 'sh'. Inversement, les commandes de CGKAT sont accessibles depuis le shell en les préfixant par 'ck'. Voici la liste de ces commandes telles qu'elle est fournie en réponse à la commande "help".

```
    Load commands:
> load BCGCT_file_name1 [cpp option] [BCGCT_file_name2 .. BCGCT_file_nameN]
                //load a BCGCT file
> loadLCG CG_name Linear_CG_file_name
                //load a file the content of which is a CG in linear format; a CG name must begin by a '#'
> openDoc GrifDoc_name1 [GrifDoc_name2 .. GrifDoc_nameN]
                //open a Thot document (-> load in the CG base the CGs defined in this document)

    Search commands:
> spec T1       //search subtypes or instances of the type T1 (depth:3; some supertypes are also given)
> spec cg1      //search CGs which are specializations of CG1
> gene T1       //search supertypes of types (or types of instances) which includes the term T1
> type of I1    //search types of instances (begining by '#' or not)
> gene CG1      //search CGs which are generalizations of CG1
> def of T1     //give the list of lambdas used for defining T1
    Note 1: 'spec' and 'gene' may respectively be abbreviated by '?' and '^'
    Note 2: the following commands control the output of requests with CGs (see the user manual):
                'meta','no meta','context [CGs]','no context [CGs]',
                'use linear','use CGs','use Repr','use Annot','use Assoc','use Repr&CGs',
                'use Repr&Annot','use Annot&CGs','use Repr&Annot&CGs'
    Note 3: the search may be done on normal CGs and/or on type definitions (lambda-abstractions):
                'on lambdas' (or 'on def'),'on CGs','on all'
    Note 4: in commands here and below, CGs may be refered by their names (beginning by '#') or
                built in linear or graphic format

    Test commands:
> ? CG1 < CG2       //test if CG1 specializes CG2
> ? T1 < T2         //test if T1 is a subtype of T2

    Assertion commands:
> CG1               //(re)assert CG1 (and displays ist content)
> ! CG1             //assert CG1
> ! CT1             //declare the concept type CT1
> ! I1 : CT1        //declare the instance I1
> ! RT1 (CT1,..,CTn)  //declare the relation type RT1
> T1 < T2 [T3 .. Tn]  //subtype T2 ..Tn by T1
> T1 > T2 [T3 .. Tn]  //subtype T1 by T2 .. Tn
> CG1 < CG2         //assert CG1 and CG2 but at present does nothing more since no structure
                    //      is implemented for storing a specialization relation on CGs
```

```
> NSC for T1 (x) are CG1          //give a definition of the concept type T1
                                  //      with necessary & sufficient conditions
> NSC for T1 (x1,..,xn) are CG1   //give a definition of the relation type T1
                                  //      with necessary and sufficient conditions
> NC for T1 (x) are CG1           //associate necessary conditions to the concept type T1
> SC for T1 (x) are CG1           //associate sufficient conditions to the concept type T1
> TC for T1 (x) are CG1           //associate typical conditions to the concept type T1
> name CG1 new_name               //(re)assert CG1 and renames it (without checking if other CGs it
> copy CG1 CG2_name               //(re)assert CG1 and copies it under another name
```

CG generation commands:

```
> ijoin on c1 c2 CG1 [CG_result_name]                  //internal join on a CG
> join on c1 c2 CG1 CG2 [CG_result_name]               //external join of two CGs
> isojoin on c1 c2 CG1 CG2 [CG_result_name]            //isojoin on two CGs ; the format for
                                                       //      concepts is:   type[:referent]
> maxjoin [-n CG_result_name] CG1 CG2 [CG3 .. CGn]     //the maximal isojoin between CG1 .. CGn
                                                       //      according to the number of joined
                                                       //      concepts and relations
```

Deletion commands (cf. also the menu 'Unload'):

```
> delCT CT1 [CT2 .. CTn]          //delete concept types if no CG use them
> delRT RT1 [RT2 .. RTn]          //delete relation types if no CG use them
> delI I1 [I2 .. In]              //delete individual marks if no CG use them
> delCG CG1 [CG2 .. CGn]          //for each CG (or lambda if the CG is not found),
                                  //      delete it (and the lambda it may be associated to)
> delLambda L1 [L2 .. Ln]         //delete lambdas and their associated CGs
                                  //      (the defined type becomes atomic but is not deleted)
> delCGs                          //delete all CGs
> closeDoc GrifDoc_name1 [GrifDoc_name2 .. GrifDoc_nameN]
```

Save commands :

```
> save env BCGCT_file_name        //save the environment
> save support BCGCT_file_name    //save the support
> save CG [BCGCT_file_name]       //assert and save a CG (or a lambda as in 'del CG1')
                                  //      in ".G"+$CG_name if no file name is given
> saveLCG CG Linear_CG_file_name  //assert and save a CG in linear format
                                  //      without meta-information
```

Trace commands: 'listCGs', 'trace', 'no trace'

Other commands:

```
> script Script_file_name         //execute the commands in the text file
                                  //      (each command must be ended by a '.' and a return)
> : any comment here              //':' do not execute its arguments (it may be useful in scripts)
> sh command                      //execute a command via the shell
> linearForm CG1 [CG2 .. CGn]     //give the linear form of CGs (useful in shell scripts)
> display CG1 [CG2 .. CGn]        //display CGs in linear format or with EDs (useful in shell scripts)
> set Variable [Value]            //set environnement variables ('setenv' is the same command)
```

Annexe 6 L'interface fonctionnelle de CGKAT pour la manipulation d'une base de Graphes Conceptuels

Les fonctions suivantes permettent de créer et manipuler les classes d'objets de CoGITo sans avoir à connaitre (déclarer) ces classes d'objets : tous les paramètres sont des chaînes de caractères ou des valeurs numériques. Le retour des fonctions est soit une chaîne de caractères contenant un message d'erreur (vide si la fonction s'est déroulée sans erreur), soit un entier pour les fonctions de test ou de comptage de nombre d'éléments dans une liste. Ces fonctions manipulent des classes d'objets C++ mais sont déclarées comme des fonctions C de manière à pouvoir être appelées par des programmes C. Elles sont réparties dans trois fichiers : cgServer.cc, conceptServer.cc et relationServer.cc. Voici les déclarations relatives à ces fonctions.

```
/***************************** cgServer.h ***********************************
 CGKAT functions for handling CoGITo CGs
 ****************************************************************************/
#define initP      /* used before a function parameter for indicating that it is initialized by this function */
#define varP       /* used before a function parameter for indicating that it is an in-out parameter */
#define DEFAULT_ZONE 1     /* By default, CGs are stored in the first CG zone */

/*------------------ Create the environnement (the base) and the support (the ontology) ----------*/
char* SgNewEnvironnement (int nbZonesCGs, int nbZonesLambdas);
char* SgNewSupport (char *supportName, int maxCTypes, int maxRelTypes,
                    int maxIndividuals, char *RelationTypeName);
int     SgSupport (); /* test if a support is loaded */

/*------------------ Load and save types and CGs ------------------------------------------------*/
char* SgLoadBCGCTFile (char *fileName);   /* Load types and CGs */
char* SgSaveCurrentSupportIn (char *fileName);
char* SgSaveCurrentEnvIn (char *fileName);
char* SgSaveCG (char *cgName, char *fileName);
char* SgPutLinearFormOfCGIn (char *cgName, char *fileName, int withMeta);
char* SgAddLinearFormOfCGIn (char *cgName, char *fileName,  char *sBefore, char *sAfter);
char* SgPutLambdaLinearFormIn (char *typeName, char *defKind, char *fileName, int withMeta);

/*------------------ Print CGs and type definitions ---------------------------------------------*/
char* SgPrintCG (char *cgName);
char* SgPrintLambdasOfType (char *typeName, char *defKind, char *lambdaName, int withMeta);
char* SgPrintAllCGs (char *specFileName);
        /*------------------ Set options about how displaying CGs (e.g. request results) --------*/
        char* SgWithTrace ();               char* SgWithoutTrace ();
        char* SgWithMetaInfo ();            char* SgWithoutMetaInfo ();
        char* SgWithContextCGs ();          char* SgWithoutContextCGs ();
```

```
/*----------------- Test on CGs ------------------------------------------------------------*/
int     SgIsaCG (char *cgName);
int     SgIsaEmptyCG (char *cgName);
int     SgIsaSpecialization (char *g1Name, char *g2Name); /* G1 < G2 ? */

/*----------------- Search on CGs ----------------------------------------------------------*/
char* SgSpecializationsOf (char *cgName, char *specFileName,
                           int withLinearCGs, int withEmbeddingCG, int doGeneralization);
int     SgNbSpecializationsOf (char *cgName, int onAll);
        /*------------ Set options about where CGs must be searched with requests ------------*/
        char* SgOnLambdas ();       char* SgNotOnLambdas ();    char* SgOnAllCGs ();

/*----------------- Add CGs ----------------------------------------------------------------*/
char* SgAddCG (char *cgName, char *cgNature, char *cgSet, char *comm);
char* SgAddCGfromLinearCGFile (char *cgName,char *cgNature, char*cgSet, char*comm,
                               char *fi leName);
char* SgAddCGfromLinearCG (char*linearCG, char*cgNature, char*cgSet, char *cgComm,
                           int keepFile, initP char *cgName);
char* SgCopyCG (char *cgName1, char *cgName2);

/*----------------- Generate CGs -----------------------------------------------------------*/
char* SgJoinCGsOn (char *ct1, char *ref1, char *ct2, char *ref2,
                   char *cgName1, char *cgName2, varP char *cgName3);
char* SgIsoJoinCGsOn (char *ct1,char *ref1, char *ct2, char *ref2,
                      char *cgName1, char *cgName2,
                      varP char *cgName3, initP int *nbSommets);
char* SgMaxJoinCGs  (char *cgName1, char *cgName2,
                     varP char *cgName3, initP int *nbSommets);

/*----------------- Delete CGs -------------------------------------------------------------*/
char* SgDelLambda (char *lambdaName);
char* SgDelLambdaWithoutCG (char *cgName);
char* SgDelCG (char *cgName);
char* SgDelAllCGs (char *nature, char *slot, char *value);

/*----------------- About comments of CGs --------------------------------------------------*/
char* SgGetCommentOfCG (char *cgName, char *slot, initP char *value);
char* SgDelCommentOfCG (char *cgName, char *slot);
char* SgModifyCommentOfCG (char *cgName, char *slot, char *value);
char* SgGetElseModCommentOfCG (char *cgName, char *slot, varP char *value);

/*----------------- About names, natures and indexing sets of CGs --------------------------*/
char* SgModifyNameOfCG (char *cgName, char *cgNewName);
char* SgModifyNatureOfCG (char *cgName, char *nature);
char* SgModifySetOfCG (char *cgName, char *set);
```

```
/*************************** conceptServer.h *******************************
 CGKAT functions for handling concepts in CoGITo CGs and concept types in the CoGITo support
 ***************************************************************************/

/*----------------- Test on concept types and individual markers ---------------------------------*/
int     SgIsaInstance (char *instName);
int     SgIsaInstanceOf (char *instName, char *ctn2);
int     SgIsaConceptType (char *ctName);
int     SgIsaCSubType (char *ctn1, char *ctn2);
int     SgIsaSubType (char *ctn1, char *ctn2);

/*----------------- Get concept types and individual markers (in the support) --------------------*/
char* SgListConceptTypesWithManyParents (char *cTypeName);
int     SgFirstSuperType (char *ctn, initP char *father);
int     SgInstTypeName (char *instName, initP char *father);

/*-----------------Add, modify and delete concept types and individual markers -----------------*/
char* SgAddInstance (char *instName, char *typeName);
char* SgAddCommentToConceptType (char *cTypeName, char *comment);
char* SgDeclareConceptType (char *cTypeName);
char* SgDefi neType (char *typeName, char *defKind, char *var, char *cgName);
char* SgKeepWordNetTypeInLattice (char *cTypeName);
char* SgLoadSupertypesOfUnsortedWNCTypes ();
char* SgAddConceptType (char *cTypeName, char *cFatherTypeName);
char* SgModInstanceName (char *instName, char *newName);
char* SgModInstanceType (char *instName, char *cTypeName);
char* SgDelInstance (char *instName);          char* SgSafeDelInstance (char *instName);
char* SgRecursDelRelationInstance (char *instName);
char* SgDelConceptType (char *cTypeName);   char* SgSafeDelConceptType (char *cTypeName);
char* SgRecursDelConceptType (char *cTypeName);

/*-----------------Add, modify and delete concepts in CGs ---------------------------------------*/
char* SgAddConceptToCG (char *cgName, char *cgSet, char *comm,
                        char *cTypeName, char *referent, initP int *conceptIdent);
char* SgSetTypeOfConceptInCG (char *cgName, int conceptIdent, char *type);
char* SgSetReferentOfConceptInCG (char *cgName, int conceptId, char *referent);
char* SgDelConceptInCG (char *cgName, int conceptIdent);

/*-----------------Build an indented list for displaying different concept type hierarchies --------*/
        /*--- for a given concept type ---*/
char* SgCTypeHierarchy (int descendantDepth, char *fi lter, char *cTypeName,
                        initP int *focusIndx, initP int *nbTypes,
                        initP char **pHierarchy, initP int *size);
        /*--- for WordNet concept types corresponding to the meanings of a term ---*/
char* SgWNCTypesHierarchiesFor (char *term, char *fi lter,
                        initP int*nbTypes, initP char **pHierarchy, initP int*size);
        /*--- for concept types the names of which include a given term ---*/
char* SgCTypesHierarchiesFor (char *term, char *fi lter, initP int *nbTypes, /* for
                        initP char **pHierarchy, initP int *size);
```

```
/*-----------------Build an indented list for displaying different type/instance hierarchies --------*/
char* SgInstHierarchyForType (int descendantDepth, char *cTypeName,
                              initP int *focusIndx, initP int *nbTypes,
                              initP char **pHierarchy, initP int *size);
char* SgInstHierarchiesFor (char *term, initP int *nbTypes,
                            initP char **pHierarchy, initP int *size);

/*----------------- About a list of concept types expressing to the meanings of some given terms --------*/
char* SgAddToTermList (char *term, int withWN, initP int *pNbTerms,
                       initP char **pTermList, initP int *size);
char* SgDelInTermList (char *term, int withWN, initP int *pNbTerms,
                       initP char **pTermList, initP int *size);



/************************ relationServer.h *********************************
 CGKAT functions for handling relations in CoGITo CGs and relation types in the CoGITo support
 ***************************************************************************/

/*----------------- Test on relation types ------------------------------------------------------*/
int     SgIsaRelationType (char *rTn1);
int     SgIsaRSubType (char *rTn1, char *rTn2);

/*----------------- Get relation types (in the support) -----------------------------------------*/
char* SgListRelationTypesWithManyParents (char *rTypeName);
int     SgRelFirstSuperType (char *rTn, initP char *father);

/*-----------------Add, modify and delete relation types ---------------------------------------*/
char* SgAddCommentToRelationType (char *rTypeName, char *comment);
char* SgDeclareRelationType (char *rTypeName, char *sig);
char* SgAddRelationType (char *rTypeName, char *rFatherTypeName);
char* SgRecursDelRelationType (char *rTypeName);
char* SgDelRelationType (char *rTypeName);          char* SgSafeDelRelationType (char *rTypeName);
char* SgDelRelationInCG (char *cgName, int relationIdent);

/*-----------------Add, modify and delete relations in CGs -------------------------------------*/
char* SgAddRelationToCG (char *cgName, char *cgSet, char *rTypeName,
                         int sourceConceptIdent, int destConceptIdent,
                         char *elemIdent, initP  int *relIdent);
char* SgSetTypeOfRelationInCG (char *cgName, int relationIdent, char *typeName,
                               initP int *newRelIdent);

/*-----------------Build an indented list for displaying different relation type hierarchies --------*/
        /*--- for a given relation type ---*/
char  *SgRTypeHierarchy (int descendantDepth, char *rTypeName,
                         char *forConceptType, int withSignature,
                         initP int *focusIndx, initP int *nbTypes,
                         initP char **pRelHierarchy, initP int * size);
        /*--- for relation types the names of which include a given term ---*/
char* SgRTypesHierarchiesFor (char *term,  initP int *nbTypes,
                              initP char **pHierarchy, initP int *size);
```

Annexe 7 Une interface fonctionnelle pour manipuler les structures logiques des documents Thot

Voici l'interface fonctionnelle que nous avons écrite pour utiliser plus facilement l'interface fonctionnelle de Thot.

```
/****************************** grif.h ************************************
 Basic functions over the Grif Editing Toolkit : complement of $GRIFDIR/api
 ************************************************************************/
#include "content.h"
#include "attribute.h"
#include "reference.h"
#include "view.h"
#include "presentation.h"
#include "interface.h"
#include "selection.h"
#include "ecfaction.h"
#include "ecf.h"
#include "message.h"
#include "ToolMessages.h"
#define STOP      True
#define GO_ON False
typedef enum {NoColor,Black,Grey,White,
                    Yellow,Green,Cyan,Blue,Violet,Purple,Magenta,Red,Orange} myColor;
#define MaxlBufElem 5000   /* for a text element */

/* ---------------------- About element types ---------------------------------------------------- */
/* DE type creation */          ElementType mkET (SSchema s,   int ident);
/* DE type init.       */          void                initET (ElementType *et, SSchema s,  int ident);

/* DE type access    */          char* GtGetElemTypeName (Element elem);
/* DE type test        */          int       GtIsText (Element elem);

/* ---------------------- About elements ---------------------------------------------------------- */
            /* DE creation */
Element  GtMkSonTo (Element e, Document doc, SSchema extSchema, int ident);
Element  GtMkTextSonTo (Element e, Document doc, SSchema extSchema, char *s);
Element  GtMkInclSonTo (Element e, Document doc, Element elemToInclude, Document d);
Element  GtMkNewSonTo (Element e, Document doc, SSchema extSchema, int ident);
Element  GtMkSibTo (Element e, Document doc, SSchema extSchema, int ident);
Element  GtMkTextSibTo (Element e, Document doc, SSchema extSchema, char *s);
Element  GtMkInclSibTo (Element e, Document doc, Element elemToInclude, Document d);

            /* DE access */
Element  GtGetSibling (Element elem);
Element  GtGetTypedAncestors (Element elem, ElementType et1, ElementType et2);
```

```
Element   GtGetTyped3Ancestors (Element elem, ElementType et1, ElementType et2, ElementType et3);
void      GtGetElemReferedByAttr (Attribute attr, Document doc, initP Element *target,
                                  initP char *targetDocName, initP Document *targetDoc);
void      GtGetElemReferedBy (Element elem, initP Element *target,
                              initP char *targetDocName, initP Document *targetDoc);
Element  GtGetMkAssociatedRoot (Document doc, SSchema extSchema, char *name, int id);

          /* DE content access */
char*     GtGetStaticStringForTextContent (Element elem);
int       GtGetSelectedText (initP char *buffSel, const int maxlBuffSel);
void      GtSearchTermsFromElement (char *terms, int nbterms, int withWordNet,
                                    varP Element *pElemCour, varP int *pFirstChar, initP int *pLastChar);

          /* DE content test */
int       isSep (char c);
int       isaWord (char *t, int lt);        int manyWords (char *t, int lt);
int       noPunctInWords (Element elem);


/* ---------------------- About attribute types ------------------------------------------------- */
/* attribute type creation */        Attribute Type mkAT (SSchema s,  int ident);
/* attribute type init.      */      void        initAT (AttributeType *at,  SSchema s,  int ident);

/* ---------------------- About attributes ------------------------------------------------------- */
          /* attribute creation */
Attribute GtMkAttributeTo (Element e, Document doc, SSchema extSchema, int ident);
Attribute GtGetMkAttributeTo (Element e, Document doc, SSchema sch, int ident);
          /* attribute (creation &) initialization */
Attribute GtSetMkAttrValueTo (Element e, SSchema sch, int ident, int value);
Attribute GtSetMkTextAttrValueTo (Element e, SSchema sch, int ident, char *s);
          /* attribute value access */
Attribute GtGetTextAttrValue (Element e, SSchema sch, int ident, initP char *s, initP int *pLng);
Attribute GtGetAttrValue (Element e, SSchema sch, int ident, initP int *pValue);

/* ---------------------- About schemas ---------------------------------------------------------- */
SSchema GtGetMkSchemaExtension (char *name, Document doc);

/* ---------------------- About marks ------------------------------------------------------------ */
void      GtSplitTextIn (Element *pFirstSelElem, Element *pLastSelElem,
                         int fi rstChar, int lastChar, int lastElemFirstChar, int lastElemLastChar,
                         int lBuff, initP char *buff, initP int *nbWords);
Element AddFirstMarkBefore (Element fi rstSelElem, SSchema sch, int numMark);
Element AddSecondMarkAfter (Element lastSelElem, SSchema sch, int numMark);
Element IfMarkGiveLastMark (Element *pElem);
```

Annexe 8 Modularisation des fichiers de sauvegarde

5 Sommaire

5.1 L'ontologie par défaut de CGKAT 346
5.2 Exemple de fichier de sauvegarde pour une application 375

Par "fichiers de sauvegarde", nous désignons ici des fichiers scripts ou au format BCGCT, et dont le chargement via le langage de commande entraîne la construction de types et/ou des GCs dans CoGITo. Par "modularisation", nous désignons ici la possibilité d'indiquer à l'intérieur de tels fichiers que suivant le contexte, certains types ou GCs doivent ou non être construits (chargés) ou que d'autres fichiers doivent être chargés préalablement.

Dans un fichier script, les instructions du shell peuvent être utilisées pour conditionner l'appel de certaines commandes et donc la constructions de types ou de GCs ou le chargement d'autres fichiers.

Par contre, le format BCGCT n'inclut pas d'instruction de test. Pour permettre la modularisation d'un fichier au format BCGCT, nous avons modifié CoGITo sur deux points : 1) le préprocesseur cpp est désormais appelé sur un fichier BCGCT avant que son contenu soit interprété (ce fichier peut ainsi contenir des macros de test et d'inclusion de fichiers tout comme dans un programme C), et 2) les fichiers au format BCGCT peuvent maintenant être découpés en blocs "anonymes" et non uniquement "nommés" (un bloc "nommé" spécifie le nom du support dans lequel il doit être chargé, tandis qu'un bloc "anonyme" peut être chargé dans n'importe quel support).

Nous présentons dans la section suivante le contenu d'un fichier au format BCGCT stockant et modularisant en différents blocs l'ontologie par défaut de CGKAT. Le premier bloc déclare les deux types de plus haut niveau, "Concept" et "Relation". Les deux blocs suivants déclarent et organisent respectivement des types de concept haut niveau (dont ceux de WordNet) et des types de relations élémentaires, de manière à faciliter la modélisation/représentation et la recherche de connaissances. Ces deux blocs spécialisent les types Concept et Relation et donc incluent le premier bloc si celui-ci n'a pas déjà été chargé. Les blocs suivants spécialisent certains types de concepts de concepts déclarés dans le second bloc. Ces blocs sont respectivement relatifs aux collections, aux structures abstraites de données, aux tâches primitives de KADS, aux tâches basiques (génériques) de KADS, aux tâches d'acquisition de connaissances à réaliser dans KADS et aux modèles à construire dans KADS_I.

Ces divers blocs étant accompagnés d'instructions de chargement conditionnel, le chargement sélectif des divers blocs de ce fichier est possible. La réutilisation du contenu de ce fichier peut bien sûr également s'effectuer simplement par copie et modification.

Le fichier présenté dans la deuxième section montre comment les types fournis par CGKAT, dont ceux de WordNet peuvent être exploités pour organiser les types d'une application. CGKAT offre une commande permettant de charger les supertypes des types de WordNet inclus dans le treillis (les supertypes non déjà inclus). Dans ce fichier, cette fonctionnalité est exploitée : lorsqu'un type de WordNet est utilisée, ses supertypes ne sont pas déclarés. En contrepartie, la commande que nous venons de mentionner devra être lancée aprés le chargement du fichier.

8.1 L’ontologie par défaut de CGKAT

```
// ============================== KB.tops ================================
// Subject: default ontology provided by CGKAT for concept types
//          and relation types (about 200 concept types and 200 relation types)
//         It includes the main top-level concept types of the WordNet ontology
//         (90.000 types) then this ontology is accessible by browsing in the
//         CGKAT 'Concept type hierarchy handling' menu, and if it is accessed
//         with a lexical search, the supertypes of the retrieved types will be
//         well categorised.
//
// Author: phmartin@sophia.inria.fr (->abbreviation used below: 'PM')
//
// Notes:1) A ';' in a comment is interpreted like a comment field ending.
//       2) 'e.g.' in a comment of a type limits the display of its subtypes.
//       3) The WordNet top-level types for verbs, adjectives and adverbs
//          are not included here (types for adjectives and adverbs cannot be
//          structured, and the WordNet types for verbs are poorly structured,
//          so too many top-level types should have been included; furthermore,
//          some types for nouns also refer to actions, thus it doesn't seems
//          necessary to include (and use) WordNet types for verbs). Then, if
//          these non-included types are retrieved via a lexical search, they
//          will be placed under 'Concept' and displayed at the end of the
//          ontology (this is also the case for 'W_abstraction' the subtypes
//          of which have been dispatched in this ontology).
//       4) The concept types used in the relation types signatures are:
//          Concept, Collection, Situation, Process, Problem_Solving_Task,
//          Goal_directed_causal_entity, Cognitive_agent,
//          Temporal_entity, Spatial_entity, Measure,
//          Proposition, Issue,Position,Argument,Fact,
//          Representation_entity, Linear_thing, Number, Integer,
//          Property, Modality, Truthness, Measure
//
//       5) The subtypes of 'W_time/n' cannot be accessed by asking for the
//          subtypes of 'W_time/n' (i.e. by selecting it) but they can be
//          retrieved using a lexical search on "time". It is probably a bug
//          of WordNet: it seems that some types do not correctly refer to
//          their subtypes or their supertypes.
//
// =======================================================================
```

```
#ifndef CGKAT_ontology
#define CGKAT_ontology

//The following ontologies are declared below. For not loading an ontology,
// get the comments off in one of the following line (but if there is an error
// message, the line number may be wrong) or comment the whole ontology
// with the C comment kind (/* ...*/).
//#define PM_Concept_ontology
//#define       RH_Collection_ontology
//#define       PM_Structured_ADT_ontology
//#define       KADS_Primitive_task_ontology
//#define       KADS_PST_ontology
//#define       KADS_Real_life_PST_ontology
//#define       KADS1_Models_ontology
//#define PM_Relation_ontology

Begin  //Allocation of memory for types and instances and declaration of
        //'Concept' and 'Relation', the two upper first order types.
        //Actually, 'Concept' need not to be declared, it is hard-coded in CoGITo.
        //This is also a synthesis of how things can be declared in BCGCT format

  Support: CGKATontology(2000,500,500); //default number of concept types, relation types and individuals
                                        //allocated for the user

    Lattice:
            ConceptTypes:       //declaration of concept types
//Concept{Comment:@:the supertype of all first-order concept types (also called T, Thing or Entity in other
ontologies);};    //already declared in CGKAT
            EndConceptTypes;

            SymRelOnTypes:      //declaration of links between concept types   (useless here)
            EndRelOnTypes;

            Order:              //ordering of concept types   (useless here)
            EndOrder;
    EndLattice;


    TRelSet:
            RelationTypes:      // declaration of relation types
 Relation{Signature:2,Concept,Concept};
//Here the type 'Relation' subsumes only binary relation types. The relation signatures must be exactly respected
//in CGs. So "[Cat]->(Relation)->[Mat]" cannot be written using "Relation{Signature:1,Concept}" nor
//"Relation{Signature:3,Concept,Concept,Concept}".
//However, we found only three usual relation types which are not binary: Not (Negation),
//S_Between (Between for spatial entities) and T_Between (Between for temporal entities).
//Truthness{Signature:2,Proposition,Truthness} may be used instead of Neg{Signature:1,Proposition}, and
//a composition of S_Before, S_After, T_Before and T_After, may be used instead of S_Between and T_Between.
            EndRelationTypes;

            SymRelOnTypes:      // declaration of links between relation types   (useless here)
            EndRelOnTypes;

            Order:              // ordering of relation types   (useless here)
            EndOrder;
    EndTRelSet;

  EndSupport;
End
#endif CGKAT_ontology
```

//---------- **The top-level concept types** ----------
#ifndef PM_Concept_ontology
#define PM_Concept_ontology

Begin //A concept is either of type 'Entity' or 'Situation'.
//Any CG describes a real or imaginary situation, by asserting the existence of one or several entities or //situations, and by asserting some relations between them. Only a situation can "occur" in time and //space. A relation like 'Point_in_time' cannot be connected to a concept of type 'Entity' but it can be //connected to a concept of type 'Situation' which describes the existence of the entity at a certain //point of time.
//The subtypes of 'Concept_playing_a_role" refer to a special role played by some entities or situations.

Support:ANON+; // anonymous support -> it can be added to any other support
// if there is no detected incoherencies

Lattice:
ConceptTypes: //'**Concept**' must have been declared

Entity{Comment: ^:Any; @:something that can be involved in a situation (ex: asserting the existence of an entity is describing a situation);};

Collection{Comment: ^:Any; @:a group or structure of abstract/concrete entities or situations, e.g. Ordered_Collection;};
W_group__grouping{Comment:@:any number of entities considered as a unit (e.g.);};
W_set/n2{Comment:@:an abstract collection of numbers or symbols, e.g. "the set of prime numbers is infinite";};

Temporal_entity{Comment: ^:Any; @:point or interval in time e.g. Time_period, Point_in_time,W_time/n;};
W_time/n{Comment:@:the continuum of experience in which events pass from the future through the present to the past;};
Point_in_time{Comment:@:point in time of a Situation;};
Time_period{Comment:@:duration of a Situation;};

Spatial_entity{Comment:^:Any;@:an entity which occupies a space region;};
W_space/n{Comment: ^:Any; @:the unlimited 3 dimensional expanse in which everything is located;};
W_location/n{Comment: ^:Any; @:a point or extent in space;};
Physical_entity{Comment: ^:Any; @:a spatial entity which is made of matter (see Material_entity);};
W_object__inanimate_object__physical_object{Comment:@:a nonliving entity;};
W_life_form__organism__being__living_thing{Comment:@:any living entity;};
W_cell/n{Comment:@:the basic structural and functional unit of all organisms;};
Imaginary_spatial_entity{Comment: ^:Any; @:e.g. Cartoon_Character;};
Cartoon_Character;

Spatial_entity_with_a_special_property{Comment: ^:Any; @:e.g. Smooth_entity, Lumpy_entity;};
Single_unit__Monad{Comment:@:anything considered as a single unit;};
Smooth_entity{Comment:@:entity perceived without discontinuities;};
Smooth_single_unit;
Continuum;
Homogeneous_entity;
Variable_entity;
Lumpy_entity{Comment:@:entity perceived with discontinuities;};
Lumpy_single_unit;
Composite_entity;
Organism{Comment:@:its parts cannot be strictly defined/separated;};
Assembly{Comment:@:its parts are discrete and can be separated;};

Abstract_entity{Comment: ^:Any; @:information or representation of an information (this entity is not spatial nor temporal);};

Representation_entity{Comment: ^:Any; @:Lexical_data (e.g. Integer, CG), Non_lexical_data (e.g. Image);};
W_representation/n1{Comment:@:a visual or tangible rendering of someone or something;};
W_communication/n{Comment:@:something that is communicated between people or groups;};
Lexical_data{Comment: ^:Any;};
Atomic_ADT{Comment:@:atomic Abstract Data Types;};
Linear_ADT{Comment:@:e.g. Number, Character;};
Character;
Number;
Integer;
Real;
Structured_ADT{Comment:@:structured Abstract Data Type, e.g. List, CG;};
Non_lexical_data{Comment: ^:Any; @:e.g. a figure, an image, an audio record;};
Representation_of_a_process{Comment: ^:Any;};
Script{Comment:@:information that specifies all the possible executions of some processes, e.g. a computer program, a musical score;};
Kinetic_script{Comment:@:script on motions;};
Procedure{Comment:@:e.g. a computer program, a schedule, a musical score;};
History{Comment:@:information that describes the execution of a process;};
Document_element{Comment: ^:Any; @:it may be lexical or not and it may represent an entity or a situation;};

Proposition{Comment: ^:Any; @:information (description of situation), e.g. Description, Argument, Hypothesis, Belief;};
W_message__content__subject_matter__substance{Comment:@:what a communication that is about something is about;};
Description;
Narration{Comment:@:report, story, biography, etc.;};
Exposition{Comment:@:operational or locational plan, etc.;};
Description_with_a_KADS_Inference_structure;
Proposition_for_argumentation;
Argument{Comment:@:deduction, persuasion, etc.;};
Issue{Comment:@:cf the argumentation relations;};
Position{Comment:@:a claim;};
Fact;
Abstraction{Comment:@:a concept which abstracts some data;};
Belief;
KADS_Role;
Hypothesis{Comment:@:Situation imagined as a being a possible cause of something;};
Observable{Comment:@:in the sense given by KADS;};
Observable_behaviour{Comment:@:in the sense given by KADS;};
Finding{Comment:@:in the sense given by KADS;};
Observation;
Complaint{Comment:@:in the sense given by KADS;};
Norm{Comment:@:in the sense given by KADS;};
Difference{Comment:@:in the sense given by KADS;};
Discrepancy_class{Comment:@:in the sense given by KADS;};
Diagnosis_result{Comment:@:in the sense given by KADS;};
Parameter{Comment:@:in the sense given by KADS;};
System_model{Comment:@:in the sense given by KADS;};
Historical_data{Comment:@:in the sense given by KADS;};
Contextualizing_proposition{Comment:@:e.g. Hypothesis, Belief, Position;};

Property{Comment: ^:Any; @:a dimension of an object, e.g. Mass, Volume, Form, Color, Speedness, Intelligence;};
W_property__attribute__dimension{Comment:@:a property or an attribute or a dimension;};
W_attribute/n{Comment:@:an abstraction belonging to or characteristic of an entity;};
Property_of_a_proposition{Comment: ^:Any; @:e.g. Modality;};
Modality{Comment:@:ex. of values: Always, Necessary, Possible, Never;};
Truthness{Comment:@:ex. of values: True, False, Unknown_truthness;};

Measure{Comment: ^:Any; @:a measure of a property of an object, e.g. W_measure__quantity__amount__quantum;};
W_measure__quantity__amount__quantum{Comment:@:how much there is of anything;};

Entity_playing_a_role{Comment: ^:Any; @:e.g. Recipient_entity, Possessed_entity, Part_entity, Material_entity;};
W_causal_agent__cause__causal_agency{Comment:@:any entity that causes events to happen;};
Goal_directed_causal_entity{Comment:@:Problem solver or interactional agent;};
Cognitive_agent{Comment:@:for example an animal or an AI-agent;};
Conscious_agent{Comment:@:for example a person;};
W_person__individual__someone__mortal__human__soul{Comment:@:a human being;};
AnyUser{Comment:@:the supertype for all users;};
Acacia_member{Comment:@:a member of the group ACACIA at the INRIA sophia Antipolis, France;};
W_consumer/n1{Comment:@:a person who purchases or uses goods or services;};
W_expert/n1{Comment:@:a person who performs skillfully;};
W_engineer__applied_scientist__technologist{Comment:@:a person who uses scientific knowledge to solve practical problems;};
Knowledge_engineer;
Non_conscious_cognitive_agent{Comment:@:e.g. AI_Agent;};
Perhaps_goal_directed_causal_entity{Comment:@:e.g. supernatural forces;};
Without_goal_causal_entity{Comment:@:non conscious Entity and not an AI_Agent;};
W_necessity__essential__requirement__requisite__need{Comment:@:anything indispensible that is needed;};
W_inessential/n{Comment:@:anything that is not essential;};
Recipient_entity;
W_recipient__receiver{Comment:@:a person who gets something;};
Possessed_entity;
W_possession/n{Comment:@:anything owned or possessed;};
Material_entity;
W_substance__matter{Comment:@:that which has mass and occupies space");};
Part_entity;
W_part__portion{Comment:@:something determined in relation to something that includes it;};
W_part__piece{Comment:@:a portion of a natural object;};
W_unit__building_block{Comment:@:a single undivided natural entity occurring in the composition of something else;};
Whole_entity;
W_whole__whole_thing__unit{Comment:@:an assemblage of parts that is regarded as a single entity;};
W_subject__content__depicted_object{Comment:@:something selected by an artist for graphic representation;};
W_variable/n{Comment:@:something that is likely to vary, something that is subject to variation;};
W_anticipation/n{Comment:@:some early entity whose type or style anticipates a later one;};
W_thing/n1{Comment:@:an entity that is not named specifically;};

Situation{Comment: ^:Any; @:something that "occurs", or that is imagined, in a region of time and space (it can be described by propositions);};

State{Comment:^:Any; @:situation that is not changing and that does not make a change during a given period of time;};
W_state/n{Comment:@:the way something is with respect to its main attributes;};
W_psychological_feature/n{Comment:@:a feature of a mental life, e.g. a feeling, a viewpoint;};
W_cognition__knowledge{Comment:@:the psychological result of perception and learning and reasoning;};
W_content__cognitive_content__mental_object{Comment:@:the sum or range of what has been perceived, discovered, or learned;};
AnyDomain{Comment:@:the supertype for all domains;};
W_knowledge_domain__knowledge_base{Comment:@:the content of a particular domain or field of knowledge;};
W_discipline__subject__subject_area__subject_field__field__field_of_study__study__branch_of_knowledge{Comment:@:a branch of knowledge;};
W_attitude__mental_attitude{Comment:@:a complex mental orientation involving beliefs and feelings;};
W_orientation/n{Comment:@:an integrated set of attitudes and beliefs;};
AnyView{Comment:@:the supertype for all viewpoints;};
W_position__view__perspective{Comment:@:a way of regarding situations or topics etc.;};
W_relation/n{Comment:@:a situation belonging to or characteristic of two entities or parts together;};

Process{Comment:@:situation that makes a change during some period of time;};
Event{Comment: ^:Any; @:a process that makes a change during a period of time considered as very short, e.g. W_event/n;};
W_event/n{Comment: ^:Any; @:something that happens at a given place and time;};
W_act__human_action__human_activity{Comment: ^:Any; @:something that people do or cause to happen, e.g. Think;};
Problem_solving_process{Comment: ^:Any; @:a cognitive activity made by an agent for solving a problem;};
Problem_Solving_Task{Comment: ^:Any; @:PST (a process such as the ones modelled in knowledge acquisition), e.g. a diagnostic;};
Real_life_PST{Comment: ^:Any; @:a problem solving task which is composed of more basic problem solving task;};
Knowledge_engineering{Comment: ^:Any; @:techniques for making a knowledge based system;};
Problem_solving_method{Comment: ^:Any; @:a way of executing a problem solving task or of decomposing it into subtasks;};
Process_playing_a_role{Comment: ^:Any; @:Method (e.g. W_method/n), SubProcess (e.g. SubTask), etc.;};
Method{Comment: ^:Any; @:a way of doing something;};
W_method/n{Comment:@:a way of doing something, esp. a systematic one; implies an orderly logical arrangement (usually in steps);};
//Problem_solving_method;
SubProcess{Comment:@:a process which is component of another process;};
SubTask{Comment:@:a task which is component of another task;};
Process_contextualizing_its_object{Comment: ^:Any; @:e.g. Think; Believe;};
W_phenomenon/n{Comment: ^:Any; @:any state or process known through the senses rather than by intuition or reasoning;};

Situation_playing_a_role{Comment: ^:Any; @:e.g. Process_playing_a_role, W_result_outcome;};
//Process_playing_a_role
W_consequence__effect__outcome__result__upshot{Comment:@:a phenomenon that follows and is caused by some previous phenomenon;};
W_result__outcome{Comment:@:the situation that exists when something ends;};

Concept_with_a_special_property{Comment: ^:Any; @:e.g. Linear_thing;};
Linear_thing{Comment: ^:Any; @:something that can be measured with a number and a dimension unit (and then which may be compared using relations like LinearLessThan);};
//Temporal_entity;
//Spatial_entity;
//Linear_ADT;

Concept_playing_a_role{Comment:^:Any;};
//Entity_playing_a_role{Comment: ^:Any; @:e.g. Recipient_entity, Possessed_entity, Part_entity, Material_entity;};
//Situation_playing_a_role{Comment: ^:Any; @:e.g. Process_playing_a_role, W_result_outcome;};
Concept_being_a_mediation{Comment:^:Any; @:anything which serves as a mediating circumstance for relating other things;};

Concept_used_by_an_agent{Comment: ^:Any; @:a concept used by an agent or a group of agents (expert(s), knowledge engineer(s), etc.);};
Concept_used_by_a_knowledge_engineer{Comment: ^:Any; @:e.g. Concept_used_by_a_KADS_knowledge_engineer;};
Concept_used_by_a_KADS_knowledge_engineer;
Concept_created_by_an_agent{Comment: ^:Any; @:may be useful when many knowledge engineers work on the same ontology;};

EndConceptTypes;

SymRelOnTypes: // declaration of links between concept types

Entity Excl Situation;
Temporal_entity Excl Abstract_entity;
Spatial_entity Excl Abstract_entity;
Physical_entity Excl Imaginary_spatial_entity;
Representation_entity Excl Proposition;
Lexical_data Excl Non_lexical_data;
Atomic_ADT Excl Structured_ADT;
Representation_entity Excl Property;
Representation_entity Excl Measure; Property Excl Measure;
Proposition Excl Property; Proposition Excl Measure;
Goal_directed_causal_entity Excl Without_goal_causal_entity;

Process Excl State;

EndRelOnTypes;

Order: // ordering of concept types

//**Entity**<Concept; //by default

Collection<Entity;
W_group__grouping<Collection;
W_set/n2<Collection;

Temporal_entity<Entity;
W_time/n<Temporal_entity;
Point_in_time<Temporal_entity;
Time_period<Temporal_entity;

Spatial_entity<Entity;
W_space/n<Spatial_entity;
W_location/n<Spatial_entity;
Physical_entity<Spatial_entity;
W_object__inanimate_object__physical_object<Physical_entity;
W_life_form__organism__being__living_thing<Physical_entity;
W_cell/n<Physical_entity;
Imaginary_spatial_entity<Spatial_entity;
Cartoon_Character<Imaginary_spatial_entity;
Spatial_entity_with_a_special_property<Spatial_entity;
Single_unit__Monad<Spatial_entity_with_a_special_property;
Smooth_entity<Spatial_entity_with_a_special_property;
Smooth_single_unit<Smooth_entity;
Continuum<Smooth_entity;
Homogeneous_entity<Continuum;
Variable_entity<Continuum;
Lumpy_entity<Spatial_entity_with_a_special_property;
Lumpy_single_unit<Lumpy_entity;
Composite_entity<Lumpy_entity;
Organism<Composite_entity;
Assembly<Composite_entity;

Abstract_entity<Entity;
Representation_entity<Abstract_entity;
W_representation/n1<Representation_entity;
W_communication/n<Representation_entity;
Lexical_data<Representation_entity;
Atomic_ADT<Lexical_data;
Linear_ADT<Atomic_ADT;
Character<Linear_ADT;
Number<Linear_ADT;
Integer<Number;
Real<Number;
Structured_ADT<Lexical_data;
Structured_ADT<Collection;
Non_lexical_data<Representation_entity;
Representation_of_a_process<Representation_entity;
Script<Representation_of_a_process;
Kinetic_script<Script;
Procedure<Script;
History<Representation_of_a_process;
Document_element<Representation_entity;

Proposition<Abstract_entity;
W_message__content__subject_matter__substance<Proposition;
Description<Proposition;
Narration<Description;
Exposition<Description;
Description_with_a_KADS_Inference_structure<Description; Description_with-_a_KAD-_Inference_structure<Concept_used_by_a_KADS_know-ledge_engineer;
Proposition_for_argumentation<Proposition;
Argument<Proposition_for_argumentation;
Issue<Proposition_for_argumentation;
Position<Proposition_for_argumentation;
Fact<Proposition_for_argumentation;
Abstraction<Proposition;
Belief<Proposition;
KADS_Role<Proposition; KADS_Role<Concept_used_by_a_KADS_knowledge_engineer;
Hypothesis<KADS_Role;
Observable<KADS_Role;
Observable_behaviour<Observable;
Finding<KADS_Role;
Observation<Finding;

Complaint<KADS_Role;
Norm<KADS_Role;
Difference<KADS_Role;
Discrepancy_class<KADS_Role;
Diagnosis_result<KADS_Role;
Parameter<KADS_Role;
System_model<KADS_Role;
Historical_data<KADS_Role;
Contextualizing_proposition<Proposition;
Hypothesis<Contextualizing_proposition;
Belief<Contextualizing_proposition;
Position<Contextualizing_proposition;

Property<Abstract_entity;
W_property__attribute__dimension<Property;
W_attribute/n<Property;
Property_of_a_proposition<Property;
Modality<Property_of_a_proposition;
Truthness<Property_of_a_proposition;

Measure<Abstract_entity;
W_measure__quantity__amount__quantum<Measure;

Entity_playing_a_role<Entity;
W_causal_agent__cause__causal_agency<Entity_playing_a_role;
Goal_directed_causal_entity<W_causal_agent__cause__causal_agency;
Cognitive_agent<Goal_directed_causal_entity;
Conscious_agent<Cognitive_agent;
W_person__individual__someone__mortal__human__soul<Conscious_agent;
AnyUser<W_person__individual__someone__mortal__human__soul;
Acacia_member<AnyUser;
W_consumer/n1<AnyUser;
W_expert/n1<W_person__individual__someone__mortal__human__soul;
W_engineer__applied_scientist__technologist<W_person__individual__someone__mortal__human__soul;
Knowledge_engineer<W_engineer__applied_scientist__technologist;
Acacia_member<Knowledge_engineer;
Non_conscious_cognitive_agent<Cognitive_agent;
Perhaps_goal_directed_causal_entity<W_causal_agent__cause__causal_agency;
Without_goal_causal_entity<W_causal_agent__cause__causal_agency;
W_necessity__essential__requirement__requisite__need<Entity_playing_a_role;
W_inessential/n<Entity_playing_a_role;
Recipient_entity<Entity_playing_a_role;
W_recipient__receiver<Recipient_entity;
Possessed_entity<Entity_playing_a_role;
W_possession/n<Possessed_entity;
Material_entity<Entity_playing_a_role;
W_substance__matter<Material_entity;
Part_entity<Entity_playing_a_role;
W_part__portion<Part_entity;
W_part__piece<Part_entity;
W_unit__building_block<Part_entity;
Whole_entity<Entity_playing_a_role;
W_whole__whole_thing__unit<Whole_entity;
W_subject__content__depicted_object<Entity_playing_a_role;
W_variable/n<Entity_playing_a_role;
W_anticipation/n<Entity_playing_a_role;
W_thing/n1<Entity_playing_a_role;

```
//Situation<Concept;   //by default

    State<Situation;
        W_state/n<State;
        W_psychological_feature/n<State;
            W_cognition__knowledge<W_psychological_feature/n;
                W_content__cognitive_content__mental_object<W_cognition__knowledge;
                    AnyDomain<W_content__cognitive_content__mental_object;
                        W_knowledge_domain__knowledge_base<AnyDomain;
                            W_discipline__subject__subject_area__subject_field-
__field__field_of_study__study__branch_of_knowledge<W_knowledge_domain__knowledge_base;
                W_attitude__mental_attitude<W_cognition__knowledge;
                    W_orientation/n<W_attitude__mental_attitude;
                        AnyView<W_orientation/n;
                            W_position__view__perspective<AnyView;
        W_relation/n<State;

    Process<Situation;
        Event<Process;
            W_event/n<Event;
        W_act__human_action__human_activity<Process;
        Problem_solving_process<Process;
            Problem_Solving_Task<Problem_solving_process;
                Real_life_PST<Problem_Solving_Task;
                    Knowledge_engineering<Real_life_PST;
            Problem_solving_method<Problem_solving_process;
        Process_playing_a_role<Process;
            Method<Process_playing_a_role;
                W_method/n<Method;
                Problem_solving_method<Method;
            SubProcess<Process_playing_a_role;
                SubTask<SubProcess;
            Process_contextualizing_its_object<Process_playing_a_role;
        W_phenomenon/n<Situation;

    Situation_playing_a_role<Situation;
        W_consequence__effect__outcome__result__upshot<Situation_playing_a_role;
        W_result__outcome<Situation_playing_a_role;


//Concept_with_a_special_property<Concept;   //by default
    Linear_thing<Concept_with_a_special_property;
        Temporal_entity<Linear_thing;
        Spatial_entity<Linear_thing;
        Linear_ADT<Linear_thing;


//Concept_playing_a_role<Concept;   //by default
    Entity_playing_a_role<Concept_playing_a_role;
    Situation_playing_a_role<Concept_playing_a_role;
    Concept_being_a_mediation<Concept_playing_a_role;


//Concept_used_by_an_agent<Concept;   //by default
    Concept_used_by_a_knowledge_engineer<Concept_used_by_an_agent;
        Concept_used_by_a_KADS_knowledge_engineer<Concept_used_by_a_knowledge_engineer;
    Concept_created_by_an_agent<Concept_used_by_an_agent;

        EndOrder;

    EndLattice;
```

```
    Conf:
 phmartin,Acacia_member;
 True,Truthness;     False,Truthness;       Unknown_thruthness,Truthness;
    EndConf;

    EndSupport;
End
#endif PM_Concept_ontology
```

```
//------------------------------------------ The top-level relation types ------------------------------------------
#ifndef PM_Relation_ontology
#define PM_Relation_ontology

Begin    //specializes the relation type 'Relation'
 Support:ANON+;
    TRelSet:

            RelationTypes:
//Relation{Signature:2,Concept,Concept};   must have been declared

    Attributive_relation{Signature:2,Concept,Concept}{@:e.g. Chrc, Attr, Manner, Name, Role, Possession,
Accompaniment;};
            Chrc{Signature:2,Concept,Property}{@:Characteristic;};
            Attr{Signature:2,Concept,Measure}{@:Attribute;};
                    Manner{Signature:2,Process,Measure}{@:process attribute, e.g. Quickly};
                            Quickly{Signature:2,Process,Measure};
                  Physical_characteristic{Signature:2,Concept,Property};
            Name{Signature:2,Concept,Representation_entity};
            Role{Signature:2,Concept,Concept};
            Possession{Signature:2,Concept,Concept};
            Accompaniment{Signature:2,Concept,Concept}{@:ex:      [Leave]->(Agent)->[Man:*x]->(Accompa-
niment)->[Woman:*y];};
                    DistributiveAccm{Signature:2,Concept,Concept}{@:each;};
                    CollectiveAccm{Signature:2,Concept,Concept}{@:all together;};
                    DisjunctiveAccm{Signature:2,Concept,Concept}{@:exclusive;};
                    RespectiveAccm{Signature:2,Concept,Concept}{@:in same order;};
                    CumulativeAccm{Signature:2,Concept,Concept}{@:as a whole;};


    Component_relation{Signature:2,Concept,Concept}{@:e.g. Part, Main_Part, Element, Subset, Subtask,
FirstSubTask;};
            Part{Signature:2,Concept,Concept};
                    Element{Signature:2,Collection,Concept};
                    SubSet{Signature:2,Collection,Collection};
                    SubSituation{Signature:2,Situation,Situation};
                            SubProcess{Signature:2,Process,Process}{@:e.g.          SubTask,          FirstSubTask,
LastSubTask;};
                                    SubTask{Signature:2,Problem_Solving_Task,Problem_Solving_Task};
                                    FirstSubTask{Signature:2,Problem_Solving_Task,Problem_Solving_Task};
                                    LastSubTask{Signature:2,Problem_Solving_Task,Problem_Solving_Task};
                    Main_Part{Signature:2,Concept,Concept};
            Parts{Signature:2,Concept,Collection};
```

Constraint_or_measure_relation{Signature:2,Concept,Concept}{@:e.g. mathematical, spatial and temporal relations;};

Measure{Signature:2,Property,Measure};

Similar_to{Signature:2,Concept,Concept}{@:or Closeness;};
Equivalence_relation{Signature:2,Concept,Concept}{@:reflexive, symmetric and transitive relation;};
Equal{Signature:2,Linear_thing,Linear_thing}{@:or Conforms_to;};
Close_to{Signature:2,Concept,Concept};
Analogy_with{Signature:2,Concept,Concept};

Different_from{Signature:2,Concept,Concept};
Opposite{Signature:2,Concept,Concept}{@:e.g. Complement;};

Ordering_relation{Signature:2,Concept,Concept}{@:in its common acceptation (see below for the mathematical sense);};
GreaterThan{Signature:2,Concept,Concept}{@:e.g. BetterThan, LinearGreaterThan(subtype of Total_ordering_relation);};
BetterThan{Signature:2,Concept,Concept};
GreaterThanOrEqual{Signature:2,Concept,Concept}{@:e.g. BetterThan, LinearGreaterThanOrEqual;};
LessThan{Signature:2,Concept,Concept}{@:e.g. LinearLessThan;};
LessThanOrEqual{Signature:2,Concept,Concept}{@:e.g. LinearLessThanOrEqual;};
Partial_ordering_relation{Signature:2,Concept,Concept}{@:reflexive, antisymmetric and transitive relation;};
Total_ordering_relation{Signature:2,Concept,Concept}{@:partial ordering relation where x=<y or y=<x for every pair x and y;};
LinearGreaterThan{Signature:2,Linear_thing,Linear_thing};
LinearGreaterThanOrEqual{Signature:2,Linear_thing,Linear_thing};
LinearLessThan{Signature:2,Linear_thing,Linear_thing};
LinearLessThanOrEqual{Signature:2,Linear_thing,Linear_thing};

Relation_from_a_collection{Signature:2,Collection,Concept}{@:e.g. Average, Count, Subset, Intersects_with, Union;};
Average{Signature:2,Collection,Number};
Count{Signature:2,Collection,Integer}{@:or Quantity;};
Subset{Signature:2,Collection,Collection}{@:or Inclusion;};
Intersects_with{Signature:2,Collection,Collection};
Meets{Signature:2,Collection,Collection};
Has_no_intersection_with{Signature:2,Collection,Collection}{@:or Exclusion;};
Complement{Signature:2,Collection,Collection};
//Set_operation_result{Signature:3,Collection,Collection,Collection}{@:e.g. Intersection, Union, Difference;};
//Intersection{Signature:3,Collection,Collection,Collection};
//Union{Signature:3,Collection,Collection,Collection};
//Difference{Signature:3,Collection,Collection,Collection};

Logical_relation{Signature:2,Proposition,Concept}{@:e.g. And, Or, Iff (most logical relation types are also subtype of Logical_contextualizing_relation);};
And{Signature:2,Proposition,Proposition};
Logical_contextualizing_relation{Signature:2,Proposition,Concept};
Truthness{Signature:2,Proposition,Truthness};
Or{Signature:2,Proposition,Proposition};
Xor{Signature:2,Proposition,Proposition};
NecessaryCondition{Signature:2,Proposition,Proposition}{@:logical implication, deduction, generalization;};
SufficientCondition{Signature:2,Proposition,Proposition};
Iff{Signature:2,Proposition,Proposition}{@:or LogicalEquivalence;};

SemanticRelation{Signature:2,Concept,Concept};
Generalization{Signature:2,Concept,Concept}{@:e.g. logical implication (NecessaryCondition), Classification, Induction;};
Classification{Signature:2,Concept,Concept};
Identification{Signature:2,Concept,Concept}{@:(instance->class);};
Abstraction{Signature:2,Concept,Concept}{@:(class->superclass);};
Induction{Signature:2,Concept,Concept}{@:(instances->class);};
Specialization{Signature:2,Concept,Concept}{@:e.g. SufficientCondition, Instanciation, RST_Elaboration;};
Instanciation{Signature:2,Concept,Concept}{@:(class->instance);};
Concept_refinement{Signature:2,Concept,Concept}{@:(class->subclass);};
Causation_relation{Signature:2,Concept,Concept}{@:e.g. Iff, Causation, Initiator, Cause_Rhetorical_relation;};
Effect_relation{Signature:2,Concept,Concept}{@:e.g. Consequence, Effect_Rhetorical_relation, Perceiver;};

Temporal_relation{Signature:2,Concept,Concept};
Temporal_relation_between_temporal_entities{Signature:2,Temporal_entity,Temporal_entity}{@: e.g. T_After, T_Between;};
T_Near{Signature:2,Temporal_entity,Temporal_entity}{@:between two temporal entities;};
T_Before{Signature:2,Temporal_entity,Temporal_entity}{@:between two temporal entities;};
T_After{Signature:2,Temporal_entity,Temporal_entity}{@:between two temporal entities;};
Temporal_relation_from_a_situation{Signature:2,Situation,Concept};//e.g. Duration, T_Until, T_Succ, Terminaison
Situation_temporal_location{Signature:2,Situation,Temporal_entity}{@:e.g. Point_in_time, Duration, T_Since, T_Until;};
Point_in_time{Signature:2,Situation,Temporal_entity}{@:between a situation and a time region;};
Duration{Signature:2,Situation,Time_period}{@:between a situation and a time period;};
T_Since{Signature:2,Situation,Temporal_entity}{@:between a situation and a time region;};
T_Until{Signature:2,Situation,Temporal_entity}{@:between a situation and a time region;};
Temporal_relation_between_situations{Signature:2,Situation,Situation}{@: e.g. T_Succ, Task_Succ, Terminaison, Consequence;};
T_Pred{Signature:2,Situation,Situation}{@:between two situations;};
Condition{Signature:2,Situation,Situation};
Causation{Signature:2,Situation,Situation};
Task_Pred{Signature:2,Problem_Solving_Task,Problem_Solving_Task};
T_Succ{Signature:2,Situation,Situation}{@:between two situations;};
Terminaison{Signature:2,Situation,Situation};
Consequence{Signature:2,Situation,Situation}{@:reverse of Causation;};
Task_Succ{Signature:2,Problem_Solving_Task,Problem_Solving_Task};
Statement_time{Signature:2,Proposition,Temporal_entity}{@:when the proposition was stated (asserted);};

Spatial_relation{Signature:2,Concept,Spatial_entity};
Spatial_relation_between_spatial_entities{Signature:2,Spatial_entity,Spatial_entity}{@:e.g. On, Above, S_In, S_After, S_Interior;};
On{Signature:2,Spatial_entity,Spatial_entity};
Above{Signature:2,Spatial_entity,Spatial_entity};
S_In{Signature:2,Spatial_entity,Spatial_entity};
S_Near{Signature:2,Spatial_entity,Spatial_entity};
S_Interior{Signature:2,Spatial_entity,Spatial_entity}{@:or Content;};
S_Exterior{Signature:2,Spatial_entity,Spatial_entity};
S_Before{Signature:2,Spatial_entity,Spatial_entity};
S_After{Signature:2,Spatial_entity,Spatial_entity};

Constraint_or_measure_relation_from_a_situation{Signature:2,Concept,Concept}{@:i.e. only from a state or a process;};
Descr{Signature:2,Situation,Proposition}{@:for describing a situation;};
// Temporal_relation_from_a_situation{Signature:2,Situation,Concept}{@:e.g. Duration, T_Succ, Consequence, Terminaison;};
// Spatial_relation_from_a_process{Signature:2,Process,Spatial_entity}{@:e.g. Source, Destination, Path;};

Relation_from_a_situation{Signature:2,Situation,Concept}{@:i.e. only from a state or a process;};

//***Constraint_or_measure_relation_from_a_situation***{Signature:2,Concept,Concept}{@:i.e. only from a state or a process;};

Relation_from_a_process{Signature:2,Process,Concept};
Purpose{Signature:2,Process,Situation}{@:a purpose of a process is also a reason to do this process;};
Recipient{Signature:2,Process,Goal_directed_causal_entity}{@:e.g. Beneficiary;};
Beneficiary{Signature:2,Process,Goal_directed_causal_entity};
Experiencer{Signature:2,Process,Goal_directed_causal_entity};
Result{Signature:2,Process,Concept};
O{Signature:2,Process,Concept}{@:output only object;};
Object{Signature:2,Process,Concept}{@:this case relation is also usually called Patient or Theme;};
I{Signature:2,Process,Concept}{@:input only object (for example for evaluation or comparison processes);};
Material{Signature:2,Process,Concept};
Parameter{Signature:2,Process,Concept};
SI{Signature:2,Process,Concept}{@:static input (e.g. for KADS tasks);};
DI{Signature:2,Process,Concept}{@:dynamic input (e.g. for KADS tasks);};
IO{Signature:2,Process,Concept}{@:input-output object, e.g. Object_to_modify, Object_to_mute, State;};
Object_to_modify{Signature:2,Process,Concept}{@:a property only is modified, e.g. the location;};
Object_to_mute{Signature:2,Process,Concept}{@:the object is transformed, e.g. with [Cut], [Destroy], [Mix];};
State{Signature:2,Process,Concept}{@:e.g. [Lay]->(State)->[Cup] "the cup is laid";};
Initiator{Signature:2,Process,W_causal_agent__cause__causal_agency};
Agent{Signature:2,Process,Goal_directed_causal_entity};
Instrument{Signature:2,Process,Entity};
Method{Signature:2,Process,Process};
Spatial_relation_from_a_process{Signature:2,Process,Spatial_entity}{@:e.g. Source, Destination, Path;};
Source{Signature:2,Process,Spatial_entity};
Destination{Signature:2,Process,Spatial_entity};
Path{Signature:2,Process,Spatial_entity};

Relation_to_a_situation{Signature:2,Concept,Situation}{@:i.e. only to a state or a process, e.g. Relation_to_a_process;};

Relation_to_a_process{Signature:2,Concept,Process}{@:e.g. SI-, the reverse of SI;};
I-{Signature:2,Concept,Process}{@:reverse of I (input);};
Material-{Signature:2,Concept,Process}{@:reverse of Material;};
Parameter-{Signature:2,Concept,Process}{@:reverse of Parameter;};
SI-{Signature:2,Concept,Process}{@:reverse of SI (static input);};
DI-{Signature:2,Concept,Process}{@:reverse of SI (static input);};

Relation_from_a_proposition{Signature:2,Proposition,Concept}{@:e.g. Information_source, Author, Rhetorical_relation, Argumentation_relation;};

Contextualizing_relation_from_a_proposition{Signature:2,Proposition,Concept}{@:e.g. Author, Modality, Or, Iff, If_Then_relation;};
Information_source{Signature:2,Proposition,Concept};
Author{Signature:2,Proposition,Cognitive_agent};
Modality{Signature:2,Proposition,Modality};
// Logical_contextualizing_relation{Signature:2,Proposition,Concept};
If_Then_relation{Signature:2,Proposition,Proposition}{@:for representing rules (not normal CGs), premisse -> conclusion;};

Believer{Signature:2,Proposition,Cognitive_agent};
Stmt{Signature:2,Proposition,Representation_entity}{@:for stating a proposition;};

Rhetorical_relation{Signature:2,Proposition,Proposition}{@:from the Rhetorical Structure Theory (RST) and the PENMAN ontology;};

Presentational_Rhetorical_relation{Signature:2,Proposition,Proposition}{@:connect to details which should induce the reader to do or think something;};
RST_Enablement{Signature:2,Proposition,Proposition}{@:details for enabling the reader to perform the described action;};
RST_Background{Signature:2,Proposition,Proposition}{@:details for enabling the reader to understand the described situation;};
RST_Motivation{Signature:2,Proposition,Proposition}{@:details for increasing the reader desire to perform the described action;};
RST_Evidence{Signature:2,Proposition,Proposition}{@:details for increasing the reader belief;};
RST_Justify{Signature:2,Proposition,Proposition}{@:details for increasing the reader acceptance;};
RST_Antithesis{Signature:2,Proposition,Proposition}{@:contrast for increasing the reader positive regard;};
RST_Consession{Signature:2,Proposition,Proposition}{@:concession for increasing the reader positive regard;};

Subject_matter_Rhetorical_relation{Signature:2,Proposition,Proposition}{@:connect to details for making a better description;};
RST_Circumstance{Signature:2,Proposition,Proposition}{@:circumstances about a situation;};
RST_Solution{Signature:2,Proposition,Proposition}{@:solution to a problem;};
RST_Elaboration{Signature:2,Proposition,Proposition}{@:e.g. RST_Subtype, RST_Property, RST_Part;};
RST_Subtype{Signature:2,Proposition,Proposition};
RST_Instance{Signature:2,Proposition,Proposition};
RST_Illustration{Signature:2,Proposition,Proposition};
RST_Attributive_relation{Signature:2,Proposition,Proposition};
RST_Property{Signature:2,Proposition,Proposition};
RST_Attribute{Signature:2,Proposition,Proposition};
RST_Possession{Signature:2,Proposition,Proposition};
RST_Part{Signature:2,Proposition,Proposition};
RST_Subtask{Signature:2,Proposition,Proposition};
RST_Specialization{Signature:2,Proposition,Proposition}{@:use if none of the previous relations can apply;};
Cause_Rhetorical_relation{Signature:2,Proposition,Proposition};
RST_Volitional_Cause{Signature:2,Proposition,Proposition}{@:cause of the described intended situation;};
RST_NonVolitional_Cause{Signature:2,Proposition,Proposition}{@:cause of the described not intended situation;};
RST_Purpose{Signature:2,Proposition,Proposition}{@:situation that the described action is intended to reach;};
Effect_Rhetorical_relation{Signature:2,Proposition,Proposition};
RST_Volitional_result{Signature:2,Proposition,Proposition};
RST_NonVolitional_result{Signature:2,Proposition,Proposition};
//Logical_relation

RST_Definition{Signature:2,Proposition,Proposition}{@:logical relations should be used instead of this relation;};
RST_Comparison{Signature:2,Proposition,Proposition}{@:use Similar_to and Different_from instead of this relation;};
RST_Means{Signature:2,Proposition,Proposition};
RST_Condition{Signature:2,Proposition,Proposition}{@:situation necessary for the realization of the described situation;};
RST_Otherwise{Signature:2,Proposition,Proposition}{@:situation which prevents the realization of the described situation;};
RST_Interpretation{Signature:2,Proposition,Proposition}{@:interpretation/abstraction/ comment on the described facts;};
RST_Evaluation{Signature:2,Proposition,Proposition}{@:positive interpretation/abstraction/comment on facts;};
RST_Restatement{Signature:2,Proposition,Proposition};
RST_Summary{Signature:2,Proposition,Proposition};
RST_Theme{Signature:2,Proposition,Proposition};
RST_Contrast{Signature:2,Proposition,Proposition}{@:comparability and differences of two situations;};

Symmetric_Rhetorical_relation{Signature:2,Proposition,Proposition}{@:e.g. RST_Restatement;};

Argumentation_relation{Signature:2,Proposition,Proposition}{@:from the AAA system (Schuler&Smith, 1990);};
Serves{Signature:2,Issue,Issue}{@:source issue serves dest. issue;};
Replacement{Signature:2,Issue,Issue}{@:dest. issue is a replacement of source issue;};
Suggestion{Signature:2,Proposition,Issue}{@:dest. issue is a suggested by the source proposition;};
Answer{Signature:2,Issue,Position}{@:dest. position is an answer to the source issue;};
Objection{Signature:2,Position,Argument}{@:dest. argument is an objection to the source position;};
Confirmation{Signature:2,Position,Argument}{@:dest. argument is a support of the source position;};
Contribution{Signature:2,Proposition,Proposition};
Reference{Signature:2,Proposition,Fact}{@:dest. fact is a reference of the source proposition;};
Contradiction{Signature:2,Proposition,Proposition};

Relation_refering_to_a_process{Signature:2,Concept,Concept}{@:e.g. Summarize, Change_location;};
Summarize{Signature:2,Concept,Concept};
Change_location{Signature:2,Concept,Concept};

Relation_with_a_special_property{Signature:2,Concept,Concept}{@:e.g. Transitive_relation, Symmetric_relation, Contextualizing_relation;};
Reflexive_relation{Signature:2,Concept,Concept}{@:for all x, xRx, e.g. R="as old as";};
Irreflexive_relation{Signature:2,Concept,Concept}{@:for all x, not(xRx), e.g. R="is the mother of";};
Symmetric_relation{Signature:2,Concept,Concept}{@:xRy implies yRx, e.g. R="is the spouse of";};
Asymmetric_relation{Signature:2,Concept,Concept}{@:(implicit supertype in this ontology) xRy implies not(yRx), e.g. R="is the husband of";};
Antisymmetric_relation{Signature:2,Concept,Concept}{@:xRy and yRx implies y=x, e.g. R="was present at birth of";};
Transitive_relation{Signature:2,Concept,Concept}{@:xRy and yRz implies yRz, e.g. R="is an ancestor of";};
Contextualizing_relation{Signature:2,Concept,Concept}{@:relation wich spefies when the content of the connected concept is true:};
// Contextualizing_relation_from_a_proposition{Signature:2,Proposition,Concept};
Contextualizing_relation_from_a_situation{Signature:2,Situation,Concept};

Relation_used_by_an_agent{Signature:2,Concept,Concept}{@:for accessing the relation types accepted/ used by a (group of) agent(s);};
Relation_created_by_an_agent{Signature:2,Concept,Concept}{@:may be useful when many knowledge engineers work on the same ontology;};

EndRelationTypes;

SymRelOnTypes: // declaration of links between relation types
// In this ontology, exclusive links could be set between nearly all relation/ types except when they specialize //each other or when one type is subtype of 'Relation_created_by_an_agent'. Here are some special examples.
Reflexive_relation Excl Irreflexive_relation;
Symmetric_relation Excl Asymmetric_relation;
SI Excl SI-;
EndRelOnTypes;

Order: // ordering of relation types

Attributive_relation<Relation;
Chrc<Attributive_relation;
Attr<Chrc;
Name<Attr;
Manner<Attr;
Quickly<Manner;
Physical_characteristic<Chrc;
Role<Attributive_relation;
Possession<Attributive_relation;
Accompaniment<Attributive_relation;
DistributiveAccm<Accompaniment;
CollectiveAccm<Accompaniment;
DisjunctiveAccm<Accompaniment;
RespectiveAccm<Accompaniment;
CumulativeAccm<Accompaniment;

Component_relation<Relation;
Element<Component_relation;
Part<Component_relation; //Part<Partial_ordering_relation;
SubSet<Part;
SubSituation<Part;
SubProcess<SubSituation;
SubTask<SubProcess;
FirstSubTask<SubTask;
LastSubTask<SubTask;
Main_Part<Part;
Parts<Component_relation;

Constraint_or_measure_relation<Relation;

Measure<Constraint_or_measure_relation;

Similar_to<Constraint_or_measure_relation; Similar_to<Symmetric_relation;
Equivalence_relation<Similar_to;
Equivalence_relation<Transitive_relation; Equivalence_relation<Reflexive_relation;
Equal<Equivalence_relation; Equal<Total_ordering_relation;
Close_to<Similar_to;
Analogy_with<Close_to;

Different_from<Constraint_or_measure_relation; Different_from<Symmetric_relation;
Opposite<Different_from;

Ordering_relation<Constraint_or_measure_relation;
GreaterThan<Ordering_relation;
BetterThan<GreaterThan;
GreaterThanOrEqual<Ordering_relation;
LessThan<Ordering_relation;
LessThanOrEqual<Ordering_relation;
Partial_ordering_relation<Ordering_relation; Partial_ordering_relation<Reflexive_relation;
Partial_ordering_relation<Antisymmetric_relation; Partial_ordering_relation<Transitive_relation;
Part<Partial_ordering_relation;
Total_ordering_relation<Partial_ordering_relation;
LinearGreaterThanOrEqual<Total_ordering_relation;
LinearGreaterThanOrEqual<GreaterThanOrEqual;
LinearGreaterThan<Total_ordering_relation; LinearGreaterThan<GreaterThan;
LinearLessThanOrEqual<Total_ordering_relation;
LinearLessThanOrEqual<LessThanOrEqual;
LinearLessThan<Total_ordering_relation; LinearLessThan<LessThan;

Relation_from_a_collection<Constraint_or_measure_relation;
Average<Relation_from_a_collection;
Count<Relation_from_a_collection;
Subset<Relation_from_a_collection;
Intersects_with<Relation_from_a_collection; Intersects_with<Symmetric_relation;
Meets<Intersects_with;
Has_no_intersection_with<Relation_from_a_collection;
Has_no_intersection_with<Symmetric_relation;
Complement<Has_no_intersection_with; Complement<Opposite;
//Set_operation_result<Relation_from_a_collection;
//Set_operation_result<Relation_refering_to_a_process;
//Intersection<Set_operation_result;
//Union<Set_operation_result;
//Difference<Set_operation_result;

Logical_relation<Constraint_or_measure_relation;
And<Logical_relation;
Logical_contextualizing_relation<Logical_relation;
Truthness<Logical_contextualizing_relation;
Or<Logical_contextualizing_relation;
Xor<Logical_contextualizing_relation;
NecessaryCondition<Logical_contextualizing_relation; NecessaryCondition<Generalization;
NecessaryCondition<Causation_relation;
Iff<NecessaryCondition;
SufficientCondition<Logical_contextualizing_relation; SufficientCondition<Specialization;
SufficientCondition<Causation_relation;
Iff<SufficientCondition;

SemanticRelation<Constraint_or_measure_relation;
Generalization<SemanticRelation;
Classification<Generalization;
Identification<Classification;
Abstraction<Classification;
Induction<Generalization;
Specialization<SemanticRelation;
Instanciation<Specialization;
Concept_refinement<Specialization;
Causation_relation<SemanticRelation;
Effect_relation<SemanticRelation;

Temporal_relation<Constraint_or_measure_relation;
Temporal_relation_between_temporal_entities<Temporal_relation;
T_Near<Temporal_relation_between_temporal_entities;
T_Near<Close_to;
T_Before<Temporal_relation_between_temporal_entities;
T_Before<LinearLessThan;
T_After<Temporal_relation_between_temporal_entities;
T_After<LinearGreaterThan;
Temporal_relation_from_a_situation<Temporal_relation;
Situation_temporal_location<Temporal_relation_from_a_situation;
Point_in_time<Situation_temporal_location;
Duration<Situation_temporal_location;
T_Since<Situation_temporal_location;
T_Until<Situation_temporal_location;
Temporal_relation_between_situations<Temporal_relation_from_a_situation;
T_Pred<Temporal_relation_between_situations;
T_Pred<LinearGreaterThan;
T_Pred<Causation_relation;
Condition<T_Pred;
Causation<T_Pred;
Task_Pred<T_Pred;
T_Succ<Temporal_relation_between_situations;
T_Succ<LinearLessThan;
T_Succ<Effect_relation;
Terminaison<T_Succ;
Consequence<T_Succ;
Task_Succ<T_Succ;
Statement_time<Temporal_relation;

Spatial_relation<Constraint_or_measure_relation;
Spatial_relation_between_spatial_entities<Spatial_relation;
On<Spatial_relation_between_spatial_entities;
Above<Spatial_relation_between_spatial_entities;
S_In<Spatial_relation_between_spatial_entities;
S_Near<Spatial_relation_between_spatial_entities;
S_Near<Close_to;
S_Interior<Spatial_relation_between_spatial_entities;
S_Exterior<Spatial_relation_between_spatial_entities;
S_Before<Spatial_relation_between_spatial_entities;
S_Before<LinearLessThan;
S_After<Spatial_relation_between_spatial_entities;
S_After<LinearGreaterThan;

Constraint_or_measure_relation_from_a_situation<Constraint_or_measure_relation;
Descr<Constraint_or_measure_relation_from_a_situation;
Temporal_relation_from_a_situation<Constraint_or_measure_relation_from_a_situation;

Relation_from_a_situation<Relation;

Constraint_or_measure_relation_from_a_situation<Relation_from_a_situation;

Relation_from_a_process<Relation_from_a_situation;
Purpose<Relation_from_a_process; Purpose<Causation_relation;
Recipient<Relation_from_a_process; Recipient<Effect_relation;
Beneficiary<Recipient;
Experiencer<Relation_from_a_process; Experiencer<Effect_relation;
Result<Relation_from_a_process; Result<Effect_relation;
O<Result;
IO<Result;
Object<Relation_from_a_process;
I<Object;
Material<I;
Parameter<I; Parameter<Causation_relation;
SI<I;
DI<I;
IO<Object;
Object_to_modify<IO;
Object_to_mute<IO;
State<IO;
Initiator<Relation_from_a_process; Initiator<Causation_relation;
Agent<Relation_from_a_process; Agent<Causation_relation;
Instrument<Relation_from_a_process;
Manner<Relation_from_a_process;
Method<Relation_from_a_process;
SubProcess<Relation_from_a_process;
Spatial_relation_from_a_process<Relation_from_a_process;
Spatial_relation_from_a_process<Constraint_or_measure_relation_from_a_situation;
Spatial_relation_from_a_process<Spatial_relation;
Source<Spatial_relation_from_a_process;
Destination<Spatial_relation_from_a_process;
Path<Spatial_relation_from_a_process;

Relation_to_a_situation<Relation;

Relation_to_a_process<Relation_to_a_situation;
I- <Relation_to_a_process;
Material- < I-;
Parameter- <I-;
SI- <I-;
DI- <I-;

Relation_from_a_proposition<Relation;

Logical_relation<Relation_from_a_proposition;

Contextualizing_relation_from_a_proposition<Relation_from_a_proposition;
Information_source<Contextualizing_relation_from_a_proposition;
Author<Information_source;
Modality<Contextualizing_relation_from_a_proposition;
Logical_contextualizing_relation<Contextualizing_relation_from_a_proposition;
If_Then_relation<Contextualizing_relation_from_a_proposition;

Believer<Relation_from_a_proposition;
Stmt<Relation_from_a_proposition;

Rhetorical_relation<Relation_from_a_proposition;

Presentational_Rhetorical_relation<Rhetorical_relation;
RST_Enablement<Presentational_Rhetorical_relation;
RST_Background<Presentational_Rhetorical_relation;
RST_Motivation<Presentational_Rhetorical_relation;
RST_Evidence<Presentational_Rhetorical_relation;
RST_Justify<Presentational_Rhetorical_relation;
RST_Antithesis<Presentational_Rhetorical_relation;
RST_Consession<Presentational_Rhetorical_relation;

Subject_matter_Rhetorical_relation<Rhetorical_relation;
RST_Circumstance<Subject_matter_Rhetorical_relation;
RST_Solution<Subject_matter_Rhetorical_relation;
RST_Elaboration<Subject_matter_Rhetorical_relation;
RST_Elaboration<Specialization;
RST_Subtype<RST_Elaboration;
RST_Instance<RST_Elaboration;
RST_Illustration<RST_Elaboration;
RST_Attributive_relation<RST_Elaboration;
RST_Property<RST_Attributive_relation;
RST_Attribute<RST_Attributive_relation;
RST_Possession<RST_Attributive_relation;
RST_Part<RST_Elaboration;
RST_Subtask<RST_Elaboration;
RST_Specialization<RST_Elaboration;
Cause_Rhetorical_relation<Subject_matter_Rhetorical_relation;
Cause_Rhetorical_relation<Causation_relation;
RST_Volitional_Cause<Cause_Rhetorical_relation;
RST_NonVolitional_Cause<Cause_Rhetorical_relation;
RST_Purpose<Cause_Rhetorical_relation;
Effect_Rhetorical_relation<Subject_matter_Rhetorical_relation;
Effect_Rhetorical_relation<Effect_relation;
RST_Volitional_result<Effect_Rhetorical_relation;
RST_NonVolitional_result<Effect_Rhetorical_relation;
//Logical_relation<Subject_matter_Rhetorical_relation;
RST_Definition<Subject_matter_Rhetorical_relation;
RST_Comparison<Subject_matter_Rhetorical_relation;
RST_Means<Subject_matter_Rhetorical_relation;
RST_Condition<Subject_matter_Rhetorical_relation;
RST_Otherwise<Subject_matter_Rhetorical_relation;
RST_Interpretation<Subject_matter_Rhetorical_relation;
RST_Evaluation<RST_Interpretation;
RST_Restatement<Subject_matter_Rhetorical_relation;
RST_Summary<Subject_matter_Rhetorical_relation;
RST_Theme<Subject_matter_Rhetorical_relation;
RST_Contrast<Subject_matter_Rhetorical_relation;

Symmetric_Rhetorical_relation<Rhetorical_relation;
Symmetric_Rhetorical_relation<Symmetric_relation;
RST_Restatement<Symmetric_Rhetorical_relation;
RST_Contrast<Symmetric_Rhetorical_relation;
RST_Antithesis<RST_Contrast;

Argumentation_relation<Relation_from_a_proposition;
Serves<Argumentation_relation;
Replacement<Argumentation_relation;
Suggestion<Argumentation_relation;
Answer<Argumentation_relation;
Objection<Argumentation_relation;
Confirmation<Argumentation_relation;
Contribution<Argumentation_relation;
Reference<Argumentation_relation;
Contradiction<Argumentation_relation;

Relation_refering_to_a_process<Relation;
Summarize<Relation_refering_to_a_process;
Change_location<Relation_refering_to_a_process;
Change_location<Object_to_modify;

Relation_with_a_special_property<Relation;
Reflexive_relation<Relation_with_a_special_property;
Irreflexive_relation<Relation_with_a_special_property;
Symmetric_relation<Relation_with_a_special_property;
Asymmetric_relation<Relation_with_a_special_property;
Antisymmetric_relation<Relation_with_a_special_property;
Transitive_relation<Relation_with_a_special_property;
Contextualizing_relation<Relation_with_a_special_property;
Contextualizing_relation_from_a_proposition<Contextualizing_relation;
Contextualizing_relation_from_a_situation<Contextualizing_relation;
Temporal_relation_from_a_situation<Contextualizing_relation_from_a_situation;
Spatial_relation_from_a_process<Contextualizing_relation_from_a_situation;

Relation_used_by_an_agent<Relation;
Relation_created_by_an_agent<Relation_used_by_an_agent;

EndOrder;
EndTRelSet;
EndSupport;
End
#endif PM_Relation_ontology

//---------- **Some collection types** ----------
#ifndef RH_Collection_ontology
#define RH_Collection_ontology

Begin //specializes the type 'Collection' with Roger&Hartley's distinctions on collections
Support:ANON+;
Lattice:
ConceptTypes: // declaration of concept types

//**Collection**{Comment:@:a group or structure of abstract/concrete entities or situations, which may have special characteristics, e.g. Ordered_collection;}; //it must have been declared
Finite_collection;
Infinite_collection;
Open_collection{Comment:@:separable, potentially infinite collection of individuals ;};
Set__Unordered_open_collection{Comment:@:without duplicate elements;};
Graph__Network{Comment:@:set of nodes and arcs;};
List__Ordered_open_collection{Comment:@:without duplicate elements;};
Bag{Comment:@:e.g. collection with possible duplicates;};
Sequence{Comment:@:ordered collection;};
Closed_collection{Comment:@:or Atomic_collection;};
Group__Unordered_closed_collection;
Tuple__Ordered_closed_collection;
Singleton;
Conjunctive_collection{Comment:@:or collective (conjoined individuals);};
Disjunctive_collection{Comment:@:or distributive (alternative individuals);};

```
        Empty_set{Comment:@:its subtypes come from (Pfeiffer&hartley, 1992);};
            Empty_tuple;
                Singleton_tuple;
                        //List__Ordered_open_collection;
            Disjunctive_set;
                Disjunctive_group;
                    //Singleton_group;
                Singleton_set;
                    Singleton_group;
                        Conjunctive_group;
                            //Tuple__Ordered_closed_collection;
                    Conjunctive_set;
                        //List__Ordered_open_collection;
                            //Tuple__Ordered_closed_collection;

        EndConceptTypes;


        SymRelOnTypes:  // declaration of links between concept types
  Finite_collection Excl Infinite_collection;
  Open_collection Excl Closed_collection;
  Conjunctive_collection Excl Disjunctive_collection;
        EndRelOnTypes;


        Order:  // ordering of concept types
    Finite_collection<Collection;
    Infinite_collection<Collection;
    Open_collection<Collection;
        Set__Unordered_open_collection<Open_collection;
            Graph__Network<Set__Unordered_open_collection;
        List__Ordered_open_collection<Open_collection;
        Bag<Open_collection;
            Sequence<Bag;
    Closed_collection<Collection;
        Group__Unordered_closed_collection<Closed_collection;
        Tuple__Ordered_closed_collection<Closed_collection;
    Singleton<Collection;
    Conjunctive_collection<Collection;
    Disjunctive_collection<Collection;
    Empty_set<Collection;
        Empty_tuple<Empty_set;
            Singleton_tuple<Empty_tuple;
                List__Ordered_open_collection<Singleton_tuple;
        Disjunctive_set<Empty_set;
            Disjunctive_group<Disjunctive_set;
                Singleton_group<Disjunctive_group;
            Singleton_set<Disjunctive_set;
                Singleton_group<Singleton_set;
                    Conjunctive_group<Singleton_group;
                            Tuple__Ordered_closed_collection<Singleton_group;
                Conjunctive_set<Singleton_set;
                    List__Ordered_open_collection<Conjunctive_set;
                        Tuple__Ordered_closed_collection<List__Ordered_open_collection;

         EndOrder;
   EndLattice;

  EndSupport;
End
#endif RH_Collection_ontology
```

```
//---------------------------------------------- Some structured abstract data types ----------------------------------------
#ifndef PM_Structured_ADT_ontology
#define PM_Structured_ADT_ontology

Begin     //specializes the type 'Structured_ADT' (discrete representation entity)
  Support:ANON+;
    Lattice:
          ConceptTypes: // declaration of concept types
//Structured_ADT{Comment:@:Abstract Data Types for collections, e.g. List, CG;};
//this type must have already been declared

    Record__Structure{Comment:@:ADT of an unordered closed collection;};
    Ordered_open_collection_ADT;
          Stack{Comment:@:(may be implemented with a List_ADT or an Array);};
          Keyed_collection_ADT{Comment:@:(idem) e.g. a dictionnary ADT;};
    Ordered_closed_collection_ADT;
          Array{Comment:@:ADT of an ordered closed collection;};
          Queue{Comment:@:(may be implemented with a List_ADT or an Array);};
    Graph_ADT;
          CG{Comment:@:conceptual graph;};
    Model_ADT;
          EndConceptTypes;


          SymRelOnTypes:  // declaration of links between concept types
//Exclusion links between collection types apply on their subtypes and then on structured abstract data types
          EndRelOnTypes;


          Order:  // ordering of concept types
    Record__Structure<Structured_ADT;
    Record__Structure<Tuple__Ordered_closed_collection;
    Ordered_open_collection_ADT<Structured_ADT;
    Ordered_open_collection_ADT<List__Ordered_open_collection;
          List_ADT<Ordered_open_collection_ADT;
          Stack<Ordered_open_collection_ADT;
          Keyed_collection_ADT<Ordered_open_collection_ADT;
    Ordered_closed_collection_ADT<Structured_ADT;
    Ordered_closed_collection_ADT<Tuple__Ordered_closed_collection;
          Array<Ordered_closed_collection_ADT;
          Queue<Ordered_closed_collection_ADT;
    Graph_ADT<Structured_ADT;
    Graph_ADT<Graph__Network;
          CG<Graph_ADT;
    Model_ADT<Structured_ADT;
          EndOrder;

    EndLattice;
  EndSupport;
End
#endif PM_Structured_ADT_ontology
```

```
//---------------------------------------------- types for KADS primitive tasks ------------------------------
#ifndef KADS_Primitive_task_ontology
#define KADS_Primitive_task_ontology

Begin     //specializes the type 'Problem_Solving_Task'
  Support:ANON+;
    Lattice:

          ConceptTypes: // declaration of concept types
//Problem_Solving_Task{Comment:@:PST (in the sense it is used in the Knowledge Acquisition litterature),
e.g. KADS1_basic_PST;};

    KADS_Primitive_Task{Comment:@: e.g. Generalize, Specify, Compare, Modify;};
          Collect;
          Generalize{Comment:@:find a superclass/superset for one or several instances/facts or classes;};
                Classify{Comment:@:(the class hierarchy is given);};
                      Abstract_class{Comment:@:class->superclass;};
                      Identify{Comment:@:instance->class;};
                Cluster{Comment:@:(here classes are learnt from examples);};
          Specialize{Comment:@:find a subclass/subset or an instance/fact;};
                Refine{Comment:@:class->subclass;};
                Instanciate{Comment:@:class->subclass/instance;};
                Select{Comment:@:set of instance->subset of instance;};
          Compare{Comment:@:in the sense given by KADS;};
                Compare_Value;
                Match{Comment:@:compare structures;};
          Modify{Comment:@:in the sense given by KADS;};
                Assign{Comment:@:a value to an attribute of something;};
                Evaluate{Comment:@:produces a concept regarding the structure of something;};
                Transform{Comment:@:e.g. sort, restructure;};
                Assemble;
                Decompose;
          Determine_truth_or_relevance{Comment:@:e.g. Establish (Cover, Verify);};
                Establish;
                Cover;
                Verify;
          Explore_option{Comment:@:e.g. Make_decision (Select), Propose_solution (Generate);};
                Make_decision;
                Propose_solution;
                Generate;
          Reduce_working_set{Comment:@:e.g. Anticipate, Prefer, Rule_out;};
                Anticipate;
                Prefer;
                Rule_out;
          EndConceptTypes;


          Order:  // ordering of concept types
    KADS_Primitive_Task<Problem_Solving_Task;
    KADS_Primitive_Task<Concept_used_by_a_KADS_knowledge_engineer;
          Collect<KADS_Primitive_Task;
          Generalize<KADS_Primitive_Task;
                Classify<Generalize;
                      Abstract_class<Classify;
                      Identify<Classify;
                Cluster<Generalize;
          Specialize<KADS_Primitive_Task;
                Refine<Specialize;
                Instanciate<Specialize;
                Select<Specialize;
          Compare<KADS_Primitive_Task;
                Compare_Value<Compare;
                Match<Compare;
```

```
        Modify<KADS_Primitive_Task;
            Assign<Modify;
            Evaluate<Modify;
            Transform<Modify;
            Assemble<Modify;
            Decompose<Modify;
        Determine_truth_or_relevance<KADS_Primitive_Task;
            Establish<Determine_truth_or_relevance;
                Cover<Establish;
                Verify<Establish;
        Explore_option<KADS_Primitive_Task;
            Make_decision<Explore_option;
                Select<Make_decision;
            Propose_solution<Explore_option;
                Generate<Propose_solution;
        Reduce_working_set<KADS_Primitive_Task;
            Anticipate<Reduce_working_set;
            Prefer<Reduce_working_set;
            Rule_out<Reduce_working_set;
        EndOrder;
    EndLattice;
  EndSupport;
End
#endif KADS_Primitive_task_ontology
```

```
//------------------------------ types for KADS basic (or generic) problem solving tasks ----------------------
#ifndef KADS_PST_ontology
#define KADS_PST_ontology

Begin    //specializes the type 'Problem_Solving_Task'
  Support:ANON+;
    Lattice:
        ConceptTypes: // declaration of concept types
// Problem_Solving_Task{Comment:@:PST (in the sense it is used in the Knowledge Acquisition litterature),
e.g. KADS1_basic_PST;};

    KADS_basic_PST{Comment:@:a basic (or generic) task which may be a component of a real life PST;};
        KADS_System_Analysis_PST;
            KADS_Classification_PST;
                KADS_Assessment_PST{Comment:@:e.g. KADS1_Assessment_PST;};
                    KADS1_Assessment_PST;
                KADS_Monitoring_PST{Comment:@:e.g. KADS1_Monitoring_PST;};
                    KADS1_Monitoring_PST;
                KADS_Diagnosis_PST{Comment:@:e.g. KADS1_Systematic_diagnosis_PST;};
                    KADS1_Systematic_diagnosis_PST;
            KADS_Prediction_PST{Comment:@:e.g. KADS1_Prediction_PST;};
                KADS1_Prediction_PST;
        KADS_System_Modification_PST;
            KADS_Repair_PST{Comment:@:e.g. KADS1_Repair_PST;};
                KADS1_Repair_PST;
            KADS_Remedy_PST{Comment:@:e.g. KADS1_Remedy_PST;};
                KADS1_Remedy_PST;
```

```
        KADS_System_Synthesis_PST;
            KADS_Assignment_PST;
                KADS1_Configuration_PST{Comment:@:(unknown structure, given elements);};
                KADS1_Scheduling_PST{Comment:@:(unknown structure, given elements);};
            KADS_Design_PST;
                KADS1_Refinement_Design_PST{Comment:@:(given structure, unknown elements);};
                KADS1_Transformational_Design_PST{Comment:@:(given structure, given
elements);};
                KADS1_Open_ended_Design_PST{Comment:@:(unknown structure, unknown
elements);};
            KADS_Planning_PST;
                KADS1_Planning_PST{Comment:@:(unknown structure, unknown elements);};
            KADS_Reconstruction_PST;
            KADS_Modelling_PST;

        EndConceptTypes;


        Order:  // ordering of concept types
    KADS_basic_PST<Problem_Solving_Task;
     KADS_basic_PST<Concept_used_by_a_KADS_knowledge_engineer;
        KADS_System_Analysis_PST<KADS_basic_PST;
            KADS_Classification_PST<KADS_System_Analysis_PST;
                KADS_Assessment_PST<KADS_Classification_PST;
                    KADS1_Assessment_PST<KADS_Assessment_PST;
                KADS_Monitoring_PST<KADS_Classification_PST;
                    KADS1_Monitoring_PST<KADS_Monitoring_PST;
                KADS_Diagnosis_PST<KADS_Classification_PST;
                    KADS1_Systematic_diagnosis_PST<KADS_Diagnosis_PST;
            KADS_Prediction_PST<KADS_System_Analysis_PST;
                KADS1_Prediction_PST<KADS_Prediction_PST;
        KADS_System_Modification_PST<KADS_basic_PST;
            KADS_Repair_PST<KADS_System_Modification_PST;
                KADS1_Repair_PST<KADS_Repair_PST;
            KADS_Remedy_PST<KADS_System_Modification_PST;
                KADS1_Remedy_PST<KADS_Remedy_PST;
        KADS_System_Synthesis_PST<KADS_basic_PST;
            KADS_Assignment_PST<KADS_System_Synthesis_PST;
                KADS1_Configuration_PST<KADS_Assignment_PST;
                KADS1_Scheduling_PST<KADS_Assignment_PST;
            KADS_Design_PST<KADS_System_Synthesis_PST;
                KADS1_Refinement_Design_PST<KADS_Design_PST;
                KADS1_Transformational_Design_PST<KADS_Design_PST;
                KADS1_Open_ended_Design_PST<KADS_Design_PST;
            KADS_Planning_PST<KADS_System_Synthesis_PST;
                KADS1_Planning_PST<KADS_Planning_PST;
            KADS_Reconstruction_PST<KADS_System_Synthesis_PST;
            KADS_Modelling_PST<KADS_System_Synthesis_PST;
        EndOrder;

    EndLattice;
  EndSupport;
End
#endif KADS_PST_ontology
```

```
//------------------------------------ KADS knowledge engineering task types ---------------------------
#ifndef KADS_Real_life_PST_ontology
#define KADS_Real_life_PST_ontology
Begin     //specializes the type Real_life_PST
  Support:ANON+;
    Lattice:

          ConceptTypes:
//Knowledge_engineering{Comment:^:Any; @:a problem solving task which is composed of more basic
problem solving task;};
    Knowledge_engineering_with_KADS_{Comment:@:combination and refinement of basic PSTs for
modelling a real life task;};
          Environment_analysis_with_KADS;     //->KADS_Organization_Model
          Problem_analysis_with_KADS;          //->KADS_Application_Model
          Task_analysis_with_KADS;             //->KADS_Task_Model
          Function_analysis_with_KADS;         //->KADS_Conceptual_Model
          Implementation_analysis_with_KADS; //->KADS_Design_Model
          EndConceptTypes;

          Order:
    Knowledge_engineering_with_KADS<Knowledge_engineering;
    Knowledge_engineering_with_KADS<Concept_used_by_a_KADS_knowledge_engineer;
          Environment_analysis_with_KADS<Knowledge_engineering_with_KADS;
          Problem_analysis_with_KADS<Knowledge_engineering_with_KADS;
          Task_analysis_with_KADS<Knowledge_engineering_with_KADS;
          Function_analysis_with_KADS<Knowledge_engineering_with_KADS;
          Implementation_analysis_with_KADS<Knowledge_engineering_with_KADS;
          EndOrder;
    EndLattice;
 EndSupport;
End
#endif KADS_Real_life_PST_ontology
```

```
//--------------------------------------- types for the  models to build in KADS_I -------------------------------
#ifndef KADS1_Models_ontology
#define KADS1_Models_ontology

Begin     //specializes the types 'Model_ADT'
  Support:ANON+;
    Lattice:
          ConceptTypes: // declaration of concept types
//Model_ADT;  //this type must have been declared
    KADS1_Model;
          KADS1_Organisational_Model;
          KADS1_Application_Model;
          KADS1_Task_Model;
          KADS1_Conceptual_Model;
          KADS1_Design_Model;
          KADS1_Model_of_Expertise;
                KADS1_Model_of_Problem_Solving_Expertise;
                KADS1_Model_of_Cooperation_Expertise;
                KADS1_Model_of_Communication_Expertise;
          KADS1_Domain_Layer;
          KADS1_Inference_Layer;
          KADS1_Task_Layer;
          KADS1_Strategy_Layer;
          EndConceptTypes;
```

```
        Order:  // ordering of concept types
    KADS1_Model<Model_ADT;
    KADS1_Model<Concept_used_by_a_KADS_knowledge_engineer;
        KADS1_Organisational_Model<KADS1_Model;
        KADS1_Application_Model<KADS1_Model;
        KADS1_Task_Model<KADS1_Model;
        KADS1_Conceptual_Model<KADS1_Model;
        KADS1_Design_Model<KADS1_Model;
        KADS1_Model_of_Expertise<KADS1_Model;
            KADS1_Model_of_Problem_Solving_Expertise<KADS1_Model_of_Expertise;
            KADS1_Model_of_Cooperation_Expertise<KADS1_Model_of_Expertise;
            KADS1_Model_of_Communication_Expertise<KADS1_Model_of_Expertise;
        KADS1_Domain_Layer<KADS1_Model;
        KADS1_Inference_Layer<KADS1_Model;
        KADS1_Task_Layer<KADS1_Model;
        KADS1_Strategy_Layer<KADS1_Model;
        EndOrder;
    EndLattice;
  EndSupport;
End
#endif KADS1_Models_ontology
```

8.2 Exemple de fichier de sauvegarde pour une application

Le fichier suivant déclare et organise des types importants pour l'analyse d'accident de la route. Le but de ce fichier n'est pas de recenser tous les principaux types relatifs à cette expertise mais simplement d'illustrer la manière dont une expertise peut être modélisée et un fichier au format BCGCT construit, en exploitant les types de WordNet. Une modularisation en divers blocs serait nécessaire si plus de types étaient recensés.

Ce fichier, nommé "KB.accid", contient une macro d'inclusion du fichier contenant l'ontologie par défaut de CGKAT ("KB.tops"). Aussi, si cette ontologie a déjà été chargée, la commande suivante doit être utilisée pour charger KB.accid : load -DCGKAT_ontology KB.accid
En effet, CGKAT ne sait pas quels blocs de fichiers de sauvegarde ont déjà été chargés (car afin de pouvoir compléter dans n'importe quelle ontologie, les blocs sont anonymes et non nommés). Cette information est gérée via des macros cpp. Pour que CGKAT connaisse les blocs déjà chargés, et donc gère lui-même ces informations, il faudrait étendre le format BCGCT pour permettre de déclarer les noms des blocs, et étendre CoGITo pour stocker ces informations.

```
// ============================== accid.tops ====================================
// Subject: some concept types used in road accident analysis
// Author: phmartin@sophia.inria.fr
// Called with : load KB.accid    or   load -DPM_Concept_ontology KB.accid
// Note: display problems with 'Vehicle/n' and 'Vehicule/n1' (WordNet bug ?)
// ==============================================================================

#ifndef PM_Road_accident_analysis_ontology
#define PM_Road_accident_analysis_ontology

#ifndef PM_Concept_ontology
#include "KB.tops"
#endif
Begin
  Support:ANON+;
    Lattice:
        ConceptTypes: // declaration of concept types


 Accidentology{Comment:@:the accident analysis discipline;};
    Road_accident_analysis_domain{Comment:@:road accidentology;};


 Concept_used_by_an_expert_in_road_accident_analysis;
//Below, the types of concepts used in road accident analysis are declared
//      and categorized as subtypes of the adequate WordNet types (or
//      WordNet concept types are directly used if they have many subtypes
//      interesting for the application).
//   They could be categorized as subtypes of
//   Concept_used_by_an_expert_in_road_accident_analysis
//   but it's boring to do and to use, then this information is put in the
//   type comments with attribute/value pairs that CGKAT can use for filtering
//   the display of the concept type hierarchy. A value may be a concept type
//   name, or an individual name, or any string (without ';' inside). In the
//   two first cases, the Subtype relation and the Instance relation are
//   exploited by CGKAT for filtering.


 INRETS_expert{Comment:@:an expert working for the INRETS;};
```

//**Physical_entity** (driver, vehicle, road infrastructure, their subparts and the physical entities used by the driver)

W_driver/n{Comment:@:the operator of a motor vehicle; domain:Road_accident_analysis_domain; user:phmartin;};
W_vehicle/n{Comment:@:a conveyance that transports people or objects;}; //(the other sense of 'vehicle' in WordNet is "communication media")
Vehicle{Comment: domain:Road_accident_analysis_domain; user:phmartin;};
Vehicle_involved_in_the_accident;
W_motor_vehicle__automotive_vehicle{Comment:@:a self-propelled wheeled vehicle that does not run on rails;};
W_car__auto__automobile__machine__motorcar;

W_infrastructure/n1{Comment:@:the basic framework or features of a system or organization;}; //(a man-made object)
Road_infrastructure{Comment: domain:Road_accident_analysis_domain; user:phmartin;};

Vehicle_part{Comment: domain:Accidentology; user:phmartin;};
//WordNet offers a lot of types for the subparts of a car,
//but CGKAT does not yet exploit the WordNet SubPart relation.
W_brake/n{Comment:@:used to slow or stop a vehicle; domain:Road_accident_analysis_domain; user:phmartin;};
W_tire__tyre{Comment:@:covering for a wheel; domain:Road_accident_analysis_domain; user:phmartin;};
W_live_axle__driving_axle{Comment:@:the axle of a self-propelled vehicle that provides the driving power; domain:Road_accident_analysis_domain; user:phmartin;};
W_suspension__suspension_system{Comment:@:a system of springs or shock absorbers connecting the wheels and axles to the chasis of a wheeled vehicle; domain:Road_accident_analysis_domain; user:phmartin;};

Infrastructure_part;
W_path/n{Comment:@:a way especially designed for a particular use; domain:Accidentology; user:phmartin;};
W_road__route{Comment:@:an open way (generally public) for travel or transportation; domain:Accidentology; user:phmartin;};

Accident_factor_Entity{Comment: domain:Accidentology; user:phmartin;};
W_drug/n{Comment:@:something that is used as a medicine or narcotic; domain:Accidentology; user:phmartin;};

//**Proposition** (vehiculated by a message, a signal)
W_error__mistake;
Erroneous_message{Comment: domain:Accidentology; user:phmartin;};
W_hint__clue{Comment:@:a slight indication;};
Clue{Comment: domain:Accidentology; user:phmartin;};
Information_on_an_accident; // < Observable;
Important_transcient_information_on_an_accident;

//**Property_or_attribute**
W_erroneousness__error{Comment:@:inadvertent incorrectness;};
Erroneousness{Comment: domain:Accidentology; user:phmartin;};
W_speed__swiftness__velocity{Comment:@:the rate at which something happens;};
Speed_Property{Comment: domain:Road_accident_analysis_domain; user:phmartin;};
W_visibility__visibleness{Comment:@:quality or fact or degree of being visible;};
Visibility_Property{Comment: domain:Road_accident_analysis_domain; user:phmartin;};

//**State**
W_conspicuousness__visibility;
Visibility_State{Comment: domain:Road_accident_analysis_domain; user:phmartin;};
W_speed__velocity{Comment:@:distance travelled per unit time;};
Speed_State{Comment: domain:Road_accident_analysis_domain; user:phmartin;};
W_wear/n{Comment:@:impairment resulting from long use;};
Wear_State{Comment: domain:Road_accident_analysis_domain; user:phmartin;};
//Psychological feature
W_error__erroneous_belief;
Erroneous_belief{Comment: domain:Accidentology; user:phmartin;};

```
//Process
    W_behavior__behaviour{Comment:@:(psychology) the aggregate of the responses or reactions or
movements made by an organism in any situation;};
        Behavior{Comment: domain:Accidentology; user:phmartin;};
    W_mistake__error__fault{Comment:@:a wrong action attributable to bad judgment or ignorance or
inattention;};
        Doing_an_error{Comment: domain:Accidentology; user:phmartin;};
    W_risk__peril__danger{Comment:@:a venture undertaken without regard to possible loss or injury;};
        Taking_risk{Comment: domain:Accidentology; user:phmartin;};
    W_speed__speeding__hurrying__hastening{Comment:@:changing location rapidly;};
        Speeding{Comment: domain:Road_accident_analysis_domain; user:phmartin;};
    W_test_trial_run;

  //Event
    W_accident/n1{Comment:@:a misfortune;};
        Accident_Event{Comment: domain:Accidentology; user:phmartin;};
    W_factor__component{Comment:@:anything that contributes causally to a result;};
        Accident_factor_Event{Comment: domain:Accidentology; user:phmartin;};
    W_hit__hitting__striking{Comment:@:a act of hitting one thing with another;};
        W_collision__crash__smash{Comment:@:the act of colliding with something;
domain:Road_accident_analysis_domain; user:phmartin;};
    W_distortion__deformation;

  //Task
    Task_of_road_accident_analysis{Comment:@:any task of road_accident_analysis;};
        W_diagnosis__diagnosing{Comment:@:identifying the nature or cause of some phenomenon;};
            Road_accident_analysis_task;
        W_gather__gathering{Comment:@:the act of gathering something;};
            Gathering_of_information_on_an_accident;
                Getting_a_general_view_on_an_accident;
                Gathering_important_transcient_information_on_an_accident;
                Quick_technical_collection_of_information_on_an_accident;
                Look_for_defaults;
                //the instances of the above subtypes may be related by subtask relations
        W_reconstruction/n3{Comment:@:an interpretation formed by piecing together bits of evidence;};
            Accident_reconstruction;

     EndConceptTypes;



      SymRelOnTypes:  // declaration of links between concept types
//to be done
      EndRelOnTypes;



     Order:  // ordering of concept types

INRETS_expert<W_expert/n1;


 //WordNet concept types need not to be categorized, the function
 //"Load supertypes of uncategorized WordNet concept types" can do it (this
 //function is accessible in the menu "Open/Load" of CGKAT).

    Accidentology<
W_discipline__subject__subject_area__subject_field__field__field_of_study__study__branch_of_knowledge;
        Road_accident_analysis_domain<Accidentology;

    Concept_used_by_an_expert_in_road_accident_analysis<Concept_used_by_an_agent;
```

```
//Physical_entity
    Vehicle<W_vehicle/n;
        Vehicle_involved_in_the_accident<Vehicle;
        W_motor_vehicle__automotive_vehicle<Vehicle;
            W_car__auto__automobile__machine__motorcar<W_motor_vehicle__automotive_vehicle;
    Road_infrastructure<W_infrastructure/n1;
    Vehicle_part<W_part__piece;
        //We do not subtype Vehicle_part (the type for the vehicle parts will
        //be accessible when CGKAT will handle the part relation between types)
    Accident_factor_Entity<Without_goal_causal_entity;

//Proposition
    Erroneous_message<W_error__mistake;
    Clue<W_hint__clue;
    Information_on_an_accident<Observable;
        Important_transcient_information_on_an_accident<Information_on_an_accident;

//Property_or_attribute
    Visibility_Property<W_visibility__visibleness;
    Erroneousness<W_erroneousness__error;
    Speed_Property<W_speed__swiftness__velocity;

//State
    Visibility_State<W_conspicuousness__visibility;
    Speed_State<W_speed__velocity;      Wear_State<W_wear/n;
    Erroneous_belief<W_error__erroneous_belief;

//Process
    Behavior<W_behavior__behaviour;
    Doing_an_error<W_mistake__error__fault;
    Taking_risk<W_risk__peril__danger;
    Speeding<W_speed__speeding__hurrying__hastening;
    W_test_trial_run<W_act__human_action__human_activity;     //for allowing
                    //the load of IntervJL.PIV before categorizing W_test_trial_run

  //Event
    Accident_Event<W_accident/n1;
    Accident_factor_Event<W_factor__component;
    W_collision__crash__smash<W_hit__hitting__striking;
    W_collision__crash__smash<Event;     //for loading IntervJL.PIV before
    W_distortion__deformation<Event;     //categorizing these types

  //Task
    Task_of_road_accident_analysis<Real_life_PST;
        Road_accident_analysis_task<W_diagnosis__diagnosing;
        Road_accident_analysis_task<Task_of_road_accident_analysis;
        Gathering_of_information_on_an_accident<W_gather__gathering;
        Gathering_of_information_on_an_accident<Task_of_road_accident_analysis;
            Getting_a_general_view_on_an_accident<Gathering_of_information_on_an_accident;
            Gathering_important_transcient_information_on_an_accident<
Gathering_of_information_on_an_accident;
            Quick_technical_collection_of_information_on_an_accident<
Gathering_of_information_on_an_accident;
            Look_for_defaults<Gathering_of_information_on_an_accident;
        Accident_reconstruction<W_reconstruction/n3;
        Accident_reconstruction<Task_of_road_accident_analysis;

        EndOrder;

    EndLattice;

  EndSupport;
End
#endif PM_Road_accident_analysis_ontology
```

Résumé. Des tâches courantes lors de la réalisation d'un système à base de connaissances, sont la recherche et la représentation d'informations contenues dans des documents (e.g. des retranscriptions d'interviews d'experts), la création et la manipulation de documents (e.g. documentation technique), la recherche et la manipulation de connaissances dans une base de connaissances (e.g. pour les valider).
Afin de faciliter l'exécution de telles tâches par un cogniticien, nous avons créé un outil logiciel permettant l'utilisation combinée :
a) des techniques avancées de structuration et de gestion de documents offertes par l'éditeur de documents structurés et hypertextes Thot, et
b) de techniques avancées de représentation et d'organisation de connaissances permises par le formalisme des Graphes Conceptuels.
Des représentations de connaissances peuvent ainsi :
a) être stockées, recherchées et gérées dans des documents via l'éditeur Thot, et
b) être exploitées pour permettre la recherche des informations qu'elles indexent dans des documents. De telles recherches peuvent s'effectuer par navigation ou par requête et permettre la génération de documents qui sont des vues sur des parties de documents ou de la base sélectionnées sur des critères conceptuels.
De plus, afin de guider et faciliter le travail du cogniticien dans la représentation et la recherche de connaissances et d'informations, nous avons constitué une ontologie comprenant :
a) des types de relations élémentaires usuelles (relations rhétoriques, méréologiques, spatiales, temporelles, mathématiques, etc.) ;
b) des types de concepts généraux que nous avons spécialisés par les 90.000 types de concepts de la base générale de connaissances terminologique WordNet.
Nous montrons comment l'exploitation de cette ontologie par des cogniticiens permet d'améliorer la cohérence, l'extensibilité et la réutilisabilité de leurs représentations de connaissances.
Cette thèse ne traite pas du problème de l'extraction automatique de connaissances à partir du documents.

Abstract. Some usual tasks in designing a knowledge-based system are document information retrieval/representation (e.g. expert interview retranscriptions), document creation/manipulation (e.g. technical documentation) and knowledge retrieval/handling (e.g. for validating the knowledge base).
In order to help the knowledge engineer to perform these tasks, we have built a knowledge acquisition tool which combines:
a) the advanced document structuring/handling techniques proposed by the structured document editor Thot, and
b) the advanced knowledge representation/organisation techniques enabled through the Conceptual Graph formalism.
Thus, these knowledge representations can be stored, retrieved and handled with the editor Thot and our tool can exploit them for allowing the retrieval of document parts which are indexed by these representations. The user may retrieve knowledge representations or document parts by navigation or conceptual requests. The results of these requests are generated virtual documents (or "views") collecting parts of documents or parts of the knowledge base which are selected on conceptual criteria.
Furthermore, in order to help the knowledge engineer in knowledge representation/retrieval, we have built an ontology which consists of:
a) usual basic relation types (e.g. rhetorical, merelogical, spatial, temporal and mathematical relations);
b) top-level concept types that we have specialized by the 90,000 concept types of the terminological knowledge base WordNet.
We show how the exploitation of such an ontology by knowledge engineers tends to improve the coherence, the extendibility and the reusability of their knowledge representations.
This thesis does not tackle the problem of automatic knowledge extraction from documents.