Description Logic in Prac : A CLASSIC Application

Deborah L. McGuinness
Lori Alperin Resnick
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974
{dlm,resnick}@research. att.com

Charles Isbell
MIT AI Lab
545 Tech Square
Cambrige MA 02139
isbell@ai.mit.edu

Description logic-based configuration applications have been used within AT&T since 1990 to process over two and a half billion dollars worth of orders. While this family of applications[4] has widely acknowledged importance, it is difficult to use for pedagogical purposes since the typical product configured is a highly interconnected, complicated technical piece of equipment like the DACS IV-2000.[1] We have developed a smaller-scale configuration application that has analogous reasoning processes but a more approachable domain—that of building home theater systems. This application provides a platform for explaining how Description Logic-based Systems (DLSs) work, in our case the CLASSIC knowledge representation system[l], and how they can support industrial applications like configuration.

CLASSIC[2] is an object-centered representation and reasoning tool with a formal foundation in description logic. CLASSIC and many DLSs are particularly well suited for applications in areas like configuration that must

1. encode rich class and object descriptions;
2. provide active inference (such as automatic classification of classes and objects into a generalization hierarchy, rule firing and maintenance, inheritance, propagation, etc.);
3. explain the reasoning process;
4. handle an incomplete and incrementally evolving knowledge base; and
5. handle errors in a way that keeps the knowledge base consistent, but also provides useful information to the user.

We will provide some examples in our domain that illustrate each of these areas.

Class and object descriptions: As in any application, we need a domain ontology in which to work. Our home theater application contains a knowledge base including a concept taxonomy and instance descriptions. The knowledge base was created by working with an expert in the domain. The database of instance information was also hand-compiled for this small application but in other applications where we work with changing instance information, we have written automatic translation routines that periodically access databases and then update our knowledge base. The terminological knowledge base contains definitional information concerning classes as well as rules. We worked directly with an expert to obtain these rules, but in our larger applications of this sort [4], system builders begin with preexisting rule specifications and use a rule translator to generate CLASSIC rules. Rules in this application fall into two classes: both hard and fast electrical rules (for example, a receiver must have an A/B switch in order to support secondary main speakers), and "rules of thumb" (for example, home theater systems do not have more than one TV or two VCRs). All products configured by the knowledge base must abide by the hard and fast electrical rules, and products configured following our "guidance" also follow the rules of thumb.

Active inference: The home theater application uses CLASSIC to provide active inference after our interface has guided the user through a few simple questions. We assume that people want to build audio only, home theater only, or combination audio/video systems and that they already have a price range in mind. Thus, we ask which type of system they want, and what quality they are willing to pay for. With these two inputs, the application uses CLASSIC to ask follow up questions as appropriate and to produce a complete (abstract) description of a consistent product. For example, if the user chooses a high-quality combination system, then CLASSIC deduces that the target system must have an amplifier, preamplifier, tuner, main, surround, and center speakers, a subwoofer, VCR, and TV, and presents this information graphically. CLASSIC calculates the deductive closure of the information provided, which usually implies properties of the system as a whole as well as properties of all the individual components. The user can view the completed information on any component just by clicking on the icon. For example, if she clicks on the TV, she sees, among other things, that the list price must be at least $1000.

Explanation: CLASSIC can justify all of its beliefs.[2]. In the example above, if the user asks how

[1] DACS IV-2000 is a digital cross-connect system that processes digitized signals for some US standard transmission rates.

[2] CLASSIC is freely available for academic purposes, and commercially available for other purposes. It has been distributed to over 80 universities and is in use in many internal projects within AT&T.

the TV acquired its price restriction, she learns that a rule fired which says that high-quality systems must have high-quality components, which for TVs enforces a minimum price of $1000. The explanation facility can also answer other questions such as why one object does or does not "subsume" (is or is not more general than) another object, why a rule fired on an object, or why an error occurred.

Incomplete and evolving knowledge bases: CLASSIC allows refinement of (and changes to) a system specification. For example, a user could add new components (e.g., a turntable), chosen from a panel of icons. She might also "instantiate" a component description by choosing a particular make and model. The interface will only generate choices that appear to be consistent with the information that CLASSIC has derived about the component (by using the specification of the component as a query to the database of individuals). The user may also delete a requirement on the system, in which case any deductions that were made as a result of this requirement are removed from the specification. When the user has finished refining the system to her satisfaction, she can ask the application to complete the specification for her. The application will then choose consistent makes and models for all the components she has left unspecified. She can then view a parts list, after which she might want to ship the order off to the factory.

If the user is not familiar with different types of stereo equipment, she may wish to trust our expert, and build a system starting with one of the example systems, where all the components are known to work well together. She can then refine this system according to her needs, requesting alternative makes and models to the ones chosen, and adding and removing components.

Errors: Although CLASSIC and the application minimize the places where a user can make an error, errors can still occur the application does not ask CLASSIC to precalculate all possible consequences of a given choice. The user could make a choice which would cause a rule to fire, which would then cause a propagation of some inconsistent information. For example, suppose the user wants to build a system, starting with a few components she already owns, including a small TV. She may later add a price range for the whole system, and based on this information, CLASSIC classifies her system as a high-quality system. All the high-quality system rules then fire, including one which requires the system's TV to have at least a 27-inch diagonal. This information gets propagated onto the user's small TV, which causes an error, CLASSIC does not allow the knowledge base to be in an inconsistent state, so it will roll back the knowledge base to the previous consistent state, meanwhile saving copies of all the individuals that led to the error, in their inconsistent states. If the user asks CLASSIC for an explanation of the error, CLASSIC can access the inconsistent state information to generate an explanation.

We feel that description logic technology is particularly well matched to this style of configuration problem for the following reasons: First, the application is fairly logical (not heuristic) so we would either have to implement the logic in a programming language or start with a tool like CLASSIC that incorporates a formal logic. Second, this domain is naturally hierarchical and rule information is appropriate at many different levels of the taxonomy. DLSs support hierarchical rules instead of using a more traditional, flat rule-based approach. This may simplify knowledge engineering and maintenance[3]. CLASSIC rules can be simpler because they only need to contain content appropriate to a certain level of concept in the hierarchy, and they do not need to contain any control information. Finally, the application naturally incorporates many different types of inference; a few of which include: inheritance, propagation, bounds constraints, and rules. These can be encoded directly in DLSs instead of needing to be paraphrased into rules. Possibly more importantly, explanations of the reasoning process may be in terms of the naturally occurring inferences.

This home theater system is a simple example of a family of applications where a description logic-based platform is used to implement standard configuration tasks and provide the basis for additional functionalities. The deployed applications built on this design have provided many advantages including decreased order processing intervals (facilitating hypothetical configuration evaluations, which were previously infeasible), reductions in personnel required to maintain product information, accurate and up-to-date pricing for sales quotes, elimination of duplication in databases, and identification of incompatible knowledge.

Acknowledgements

We wish to thank the entire CLASSIC group, particularly Peter Patel-Schneider and Ron Brachman, for their insightful comments on this work.

References

[1] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick, and A. Borgida. Living with CLASSIC: When and How to Use a KL-ONE-Like Language. In *Principles of Semantic Networks: Explorations in the representation of knowledge,* J. Sowa, editor, Morgan-Kaufmann, pp. 401-456, 1991.

[2] D. L. McGuinness and A. Borgida. Explaining Subsumption in Description Logics. In *Proc. IJCAJ,* Montreal, August 1995.

[3] J. R. Wright, D. L. McGuinness, C. Foster, and G. T. Vesonder. Conceptual Modeling using Knowledge Representation: Configurator Applications. In *Proc. Artificial Intelligence in Distributed Information Networks, IJCAI-95,* Montreal, 1995.

[4] Wright, J. R., Weixelbaum, E. S-, Brown, K., Vesonder, G. T., Palmer, S. R., Berman, J. I., Moore, H. H.(A knowledge-based configurator that supports sales, engineering, and manufacturing at AT&T Network Systems. In *Proceedings of the Innovative Applications of Artificial Intelligence Conference,* pp.183-193, 1993.

Revealing Collection Structure through Information Access Interfaces

Marti A. Hearst Jan O. Pedersen

Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
(415) 812-4742
{hearst,pedersen }@parc.xerox.com

Information Access research at Xerox PARC focuses on amplifying the users' cognitive abilities, rather than trying to completely automate them. This framework emphasizes the participation of the user in a cycle of query formulation, presentation of results, followed by query reformulation, and so on. This framework is intended to help the user iteratively refine a vaguely understood information need. Since the focus is on query repair, the information presented is typically not document descriptions, but rather intermediate information that indicates relationships between the query and the retrieved documents. We have developed information access tools intended to supply some of this functionality, and describe two of these here.

As an illustration, suppose a user is interested in medical diagnosis software. Assume that initially the user has available a large, unfamiliar information source. In our example, this source is the 2.2 Gigabyte TIPSTER text collection [Harman, 1993]. Because the collection is unfamiliar, the user will be unsure whether it contains relevant information, and if so, how to access it.

To address this situation, we have developed a browsing method, called *Scatter/Gather* [Cutting et a/., 1992; 1993], that allows a user to rapidly assess the general contents of a very large collection by scanning through a dynamic, hierarchical representation that is motivated by a table-of-contents metaphor. Initially the system automatically *scatters,* or clusters, the collection into a small number of document groups, and presents short summaries of the groups to the user. These summaries consist of two types of information: topical titles (titles of documents close to the cluster centroid) and typical terms (terms of importance in the cluster). Based on these summaries, the user selects one or more of the groups for further study. The selected groups are *gathered)* or unioned, together to form a subcollection. The system then applies clustering again to scatter the new subcollection into a small number of document groups, which are again presented to the user. With each successive iteration the groups become smaller, and therefore more detailed. The user may, at any time, switch to a more focused search method. Figure 1 shows a portion of the top level clusters on the TIPSTER collection.

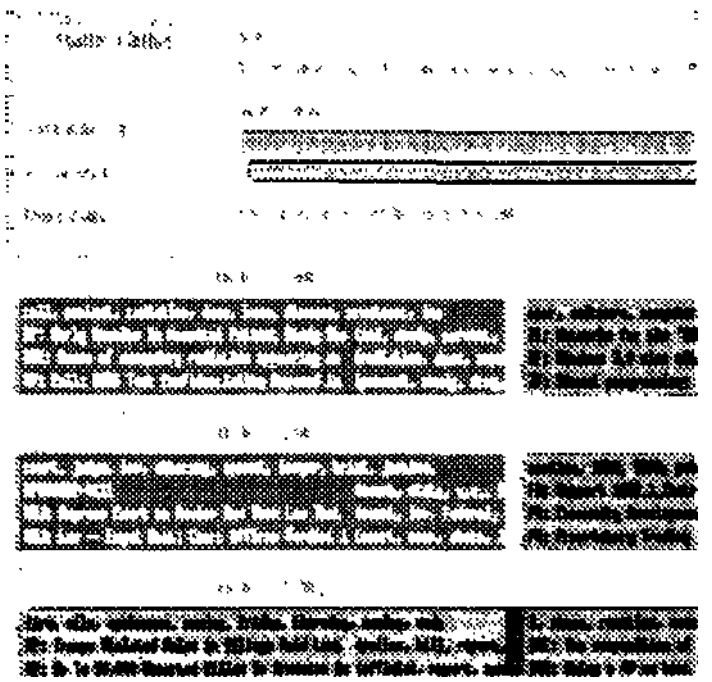

Figure 1: A portion of a top-level view of the Scatter/Gather algorithm over the TIPSTER corpus.

By browsing the collection in this manner, the user obtains an idea about the technical contents of the corpus, and can choose whether or not to further explore here or try another text collection. From the titles and terms retrieved, it becomes apparent that the collection contains commercially oriented discussions of technology, rather than predominantly academic ones. From this overview information, the user can conclude that this is indeed a promising collection for the user's information need.

Once a premising collection has been identified, the user can issue a search. In a typical information retrieval system, documents satifying the query are returned and are rank-ordered according to some function of the number of hits for each term [Salton, 1988]. But this kind of ranking is opaque to the user; it is not clear how well each term is represented in the retrieved documents.

To address these issues, the *TileBars* interface [Hearst, 1995] allows the user to make informed decisions

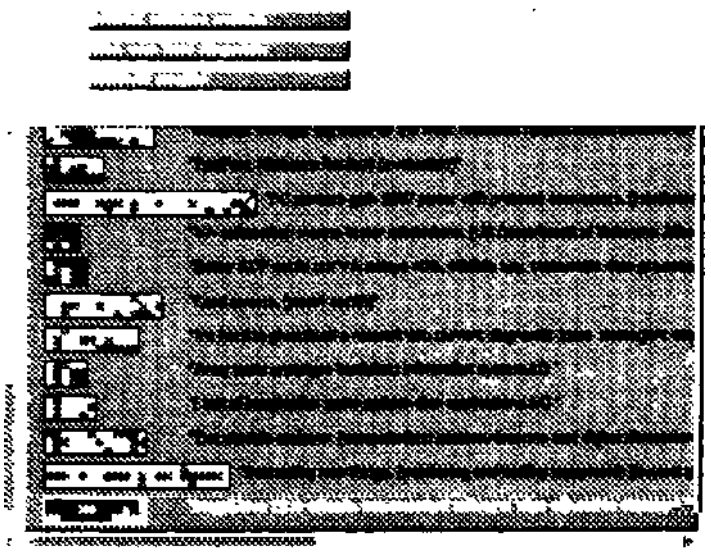

Figure 2: The TileBar Display on a query about automated systems for query diagnosis.

about which documents and which passages of those documents to view, based on the distributional behavior of the query terms in the documents. The goal is to simultaneously and compactly indicate (i) the relative length of the document, (ii) the frequency of the term sets in the document, and (iii) the distribution of the term sets with respect to the document and to each other. Each document is partitioned in advance into a set of multi-paragraph subtopical segments using an algorithm called *TextTiling* [Hearst, 1994].

Figure 2 shows an example run on a query about automated systems for medical diagnosis, run over the ZIFF portion of the TIPSTER collection. Each large rectangle indicates a document, and each square within the document represents a coherent text segment. The darker the segment, the more frequent the term (white indicates 0, black indicates 8 or more hits, the frequencies of all the terms within a term set are added together). The top row of each rectangle correspond to the hits for Term Set 1, the middle row to hits of Term Set 2, and the bottom row to hits of Term Set 3. The first column of each rectangle corresponds to the first segment of the document, the second column to the second segment, and so on.

The TileBars representation allows the user to sort the retrieved documents according to which aspects of the query are most important. For example, in the figure the query is formulated as: (patient OR *medicine* OR *medical)* AND (test OR *scan* OR *cure* OR *diagnosis)* AND *(software* OR *program).* This formulation allows the interface to indicate the role played by each conceptual part of the query: the medical terms, the diagnosis terms, and the software terms. In Figure 2, the user has indicated that the diagnosis aspect of the query must be strongly present in the retrieved documents, by setting the minimum term distribution percentage to 30% for the second termset. The document whose title begins *"VA automation means faster admissions"* is quite likely to be relevant to the query, and has all three term sets well-distributed throughout. By contrast, the document whose title begins *"It's hard to ghosibust a network ..."* is about computer-aided diagnosis, but has only a passing reference to *medical* diagnosis, as can be seen by the graphical representation. If the user decides that medical terms should be better represented, the constraint on this term set can be adjusted accordingly.

Note that a system that simply ranks the documents does not make these kinds of distinctions available to the user. The graphical representation allows the users to rapidly assess the structure of the retrieved documents with respect to the query, to better aid their decisions about which documents to view, or how to refine the query.

References

[Cutting *et ai,* 1992] Douglass R. Cutting, Jan O. Pedersen, David Karger, and John W. Tukey. Scatter/Gather: A cluster-based approach to browsing targe document collections. In *Proceedings of the 15th Annual International ACM/SIGIR Conference,* pages 318-329, Copenhagen, Denmark, 1992.

[Cutting *et al.,* 1993] Douglass R. Cutting, David Karger, and Jan Pedersen. Constant interaction-time Scatter/Gather browsing of very large document collections. In *Proceedings of the 16ih Annual International ACM/SIGIR Conference,* pages 126135, Pittsburgh, PA, 1993.

[Harman, 1993] Donna Harman. Overview of the first Text REtrieval Conference. In *Proceedings of the 16th Annual International ACM/SIGIR Conference,* pages 36-48, Pittsburgh, PA, 1993.

[Hearst, 1994] Marti A. Hearst. Multi-paragraph segmentation of expository text. In *Proceedings of the 32nd Meeting of the Association for Computational Linguistics,* June 1994.

[Hearst, 1995] Marti A. Hearst. Tilebars: Visualization of term distribution information in full text information access. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems,* Denver, CO, May 1995. ACM.

[Salton, 1988] Gerard Salton. *Automatic text processing : the transformation, analysis, and retrieval of information by computer.* Addison-Wesley, Reading, MA, 1988.