

Description Logic in Practice: 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

Cambridge 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[1], and how they can support industrial applications like configuration.

CLASSIC² 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.

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

²CLASSIC 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 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

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. IJCAI*, 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.