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
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[5] 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.

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; however, in other applications that work with changing instance information, automatic translation routines periodically access databases and then update the knowledge base[3]. 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[5], 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. If CLASSIC supported defaults, the rules associated with guided stereo systems would have been defaults.

Active inference: The home theater application uses CLASSIC to provide active inference after the 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 usually 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, imagine that the user chooses a combination system, but does not specify a particular level of quality. CLASSIC deduces that the target system must have main speakers, a VCR, and a TV. Since all stereo systems should have some kind of preamplifier and amplifier, and the choice of an unspecified combination system does not imply anything about them, the user is asked to decide whether she wants seperate amplifier components or a receiver (which includes an amplifier, preamplifier and a tuner). Because there is no pricing information, CLASSIC is unable to infer more components. The information that CLASSIC is given and has deduced

[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 85 universities and is in use in many internal projects within AT&T.

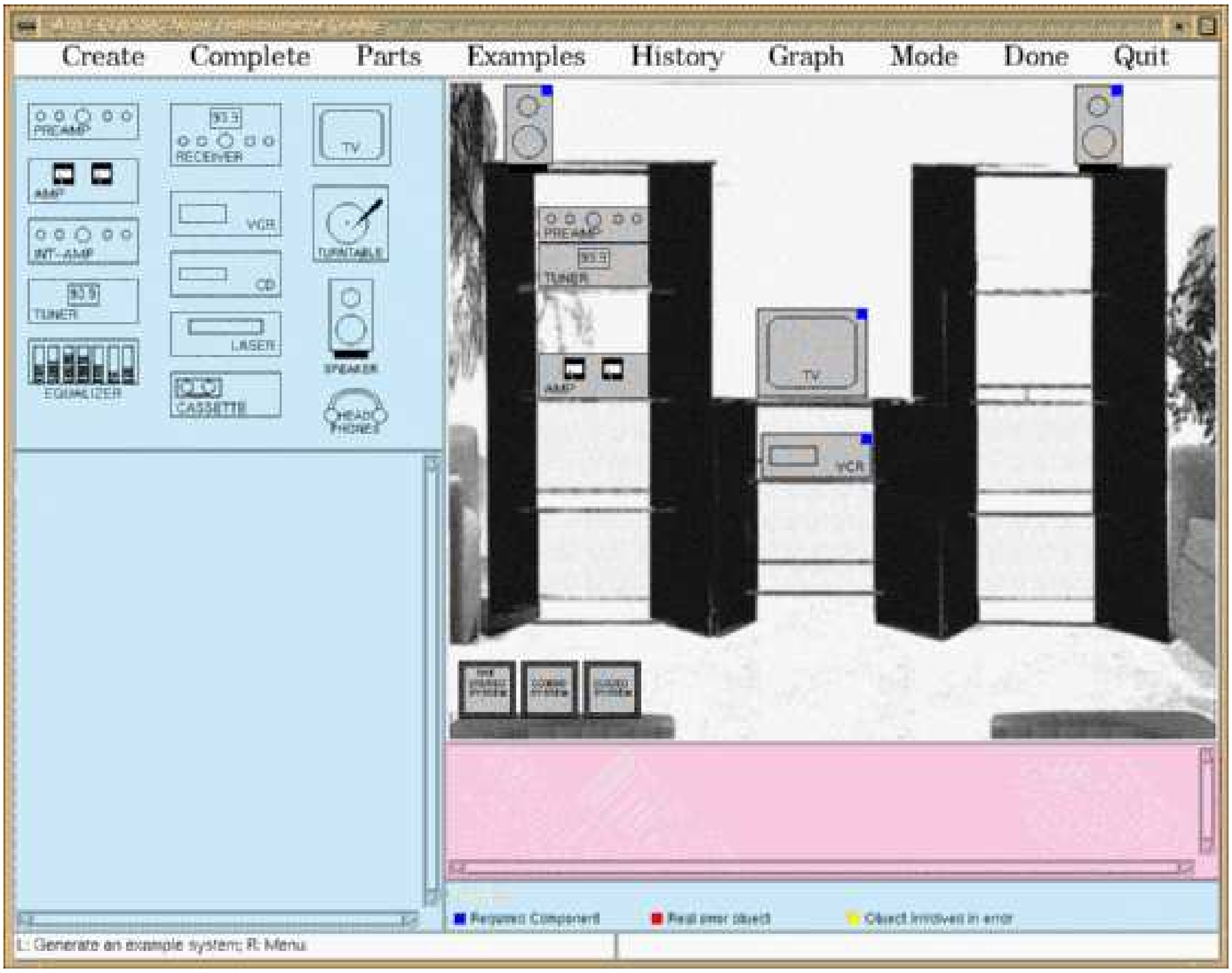

Figure 1: **The CLASSIC Home Entertainment Configuration System.** The window has four major areas. The first (upper left) is a components pane, with icons representing all the kinds of components that CLASSIC knows about. Clicking on one of these icons adds a component of that type to the stereo system under construction. The second area (upper right) is the living room pane, where all the components that have been added to the stereo system (either directly by the user or by CLASSIC's inferences) are presented graphically. There are also icons representing the stereo system itself as well as important concepts of which it is an instance (e.g. **High Quality System**). Clicking an icon allows the user to add, remove or display information about the object that icon represents. An icon with a small square in its upper right-hand corner is required. The third area (lower left) is the display pane. Information about components, pricing information and explanations are displayed here. The final area (lower right) is the error pane. Icons representing objects that have caused errors are displayed here. Clicking on one of these icons allows the user to display or explain information about that object.

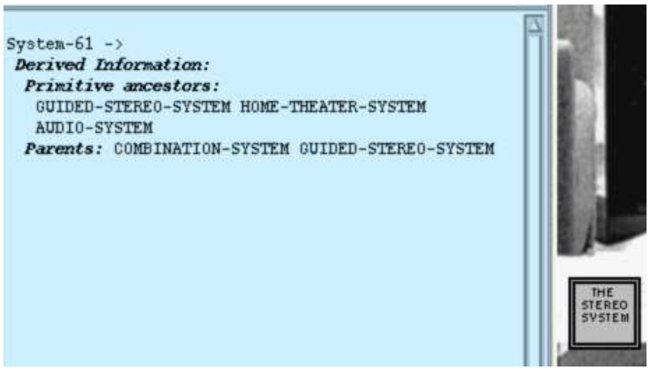

Figure 2: **Selected information about the stereo system being created.** Thus far, all we really know about the system is that it is a combination audio and home theater system and that it is following our expert's rules of thumb.

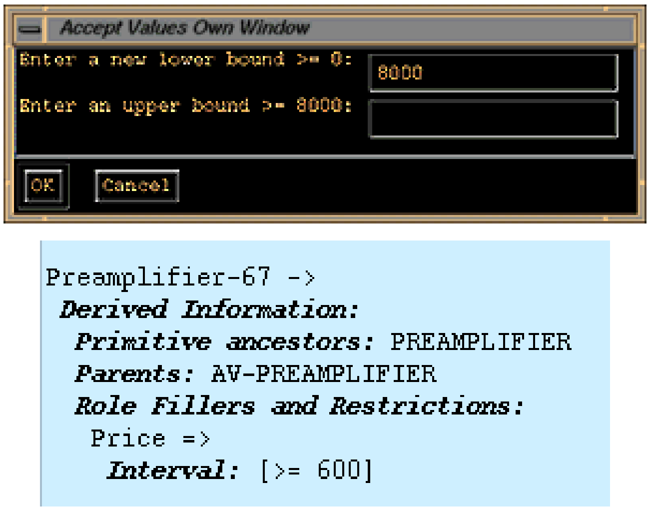

Figure 3: **Information propagation.** After adding a price restriction to the stereo system, CLASSIC deduces that it is a high–quality system. Since it is also guided by our expert's rules of thumb, this propagates pricing information to all of the components, including the preamplifier.

is presented graphically (see Figure 1).

CLASSIC calculates the deductive closure of the information provided. The user can view the completed information on any component just by clicking on its icon. The user can also view properties of the whole system by clicking on a special stereo system icon. Because she has chosen an unspecified stereo system, if she clicks on it, she will not see much that she does not already know: this is a guided combination system (see Figure 2)[3].

This changes, however, once she adds some information. For example, deciding that she is willing to spend at least $8000 on her dream system not only makes the salesperson happy, but allows CLASSIC to deduce that she wants a high–quality combination system. CLASSIC then infers that her system must have in addition to its main speakers, a pair of surround speakers, a center speaker and a subwoofer. As is often the case, providing information about the system as a whole implies properties of individual components. Clicking on the icon for the preamplifier, for example, reveals that, among other things, it must have a list price of at least $600 (see Figure 3).

Explanation: CLASSIC can justify all of its beliefs[2]. Not only can the user view any piece of information, she can also have any deduction explained. In our example, if the user asks how the preamplifier acquired its price restriction, she learns that a rule fired that says that high–quality systems must have high–quality components, which for preamplifiers enforces a minimum price of $600 (see Figure 4). 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. Inferences can also have templates associated with them that may be used to present explanations in terms that are acceptable to the user.

Incomplete and evolving knowledge bases: As was the case with specifying a minimum price for the stereo system, CLASSIC allows continual refinement of (and changes to) its knowledge base. In addition to specifying price information, a user might also "instantiate" a component description by choosing a particular make and model. Since the system must be consistent, the interface will only generate choices that appear to satisfy all constraints that CLASSIC has derived about the component (by using the specification of the component as a query to the database of individuals).

It is worth stressing that adding information to the system or to any of its components can have many implications. For example, choosing a particular tuner often allows CLASSIC to instantiate the preamplifier (see Figure 5).

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 that would then cause a propagation of some inconsistent information. In our example, the user already has a TV and, as it turns out, systems under our guidance may only have at most one TV. So, attempting to add another one generates 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 (see Figure 6).

Additional functionality: In addition to instanti-

[3] CLASSIC actually infers far more about the object; however, CLASSIC is able to prune "uninteresting" display information. For example, in this application components have information associated with them that describe how they should be displayed. Displaying these implementation details would only be confusing.

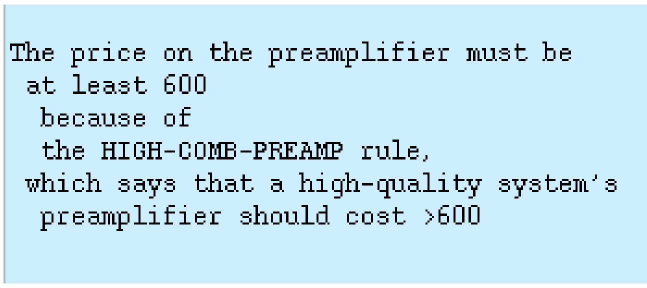

Figure 4: **Explaining new information.** CLASSIC is capable of explaining all of its inferences. Above, the preamplifier's price restriction is the result of a particular rule about high-quality stereo systems.

ating components and adding new components, the user may also delete a requirement on the system. In this case, any deductions that were made as a result of this requirement are removed from the specification. 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. 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 (see Figure 7).

Discussion: We feel that description logic-based technology is particularly well matched to this style of configuration problem for several 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[4]. 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

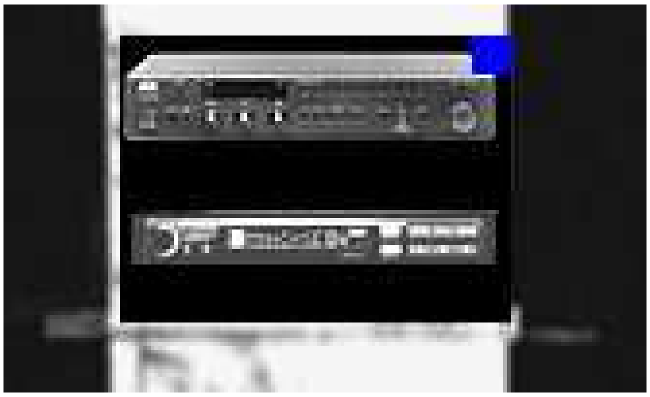

Figure 5: **More information propagation.** Choosing a particular tuner causes CLASSIC to deduce a matching preamplifier. Notice that instantiated components are distinguished from non-instantiated ones in that instantiated components have "real" pictures associated with them while uninstanitated components only have icons.

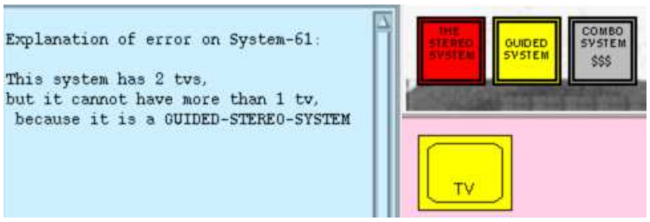

Figure 6: **Errors.** Adding another TV, while having a certain appeal, is not allowed by the system because guided stereo systems can have at most one TV. The offending object is placed in the error pane where it is available to be inspected and explained.

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. We also wish to thank the PROSE/QUESTAR team, particularly Jon Wright, Harry Moore, Jay Berman, Charlie Foster, and Pat Saleh, for continuing feedback on applications needs.

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] R. J. Brachman, P. G. Selfridge, L. G. Terveen, B. Altman, A. Borgida, F. Halper, T. Kirk, A. Lazar, D. L. McGuinness, and L. A. Resnick. Integrated Support for Data Archaelogy. in *International Journal of Intelligent and Cooperative Information Systems*, 2(2), 1993, pp. 159–185.

[4] 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.

[5] 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.

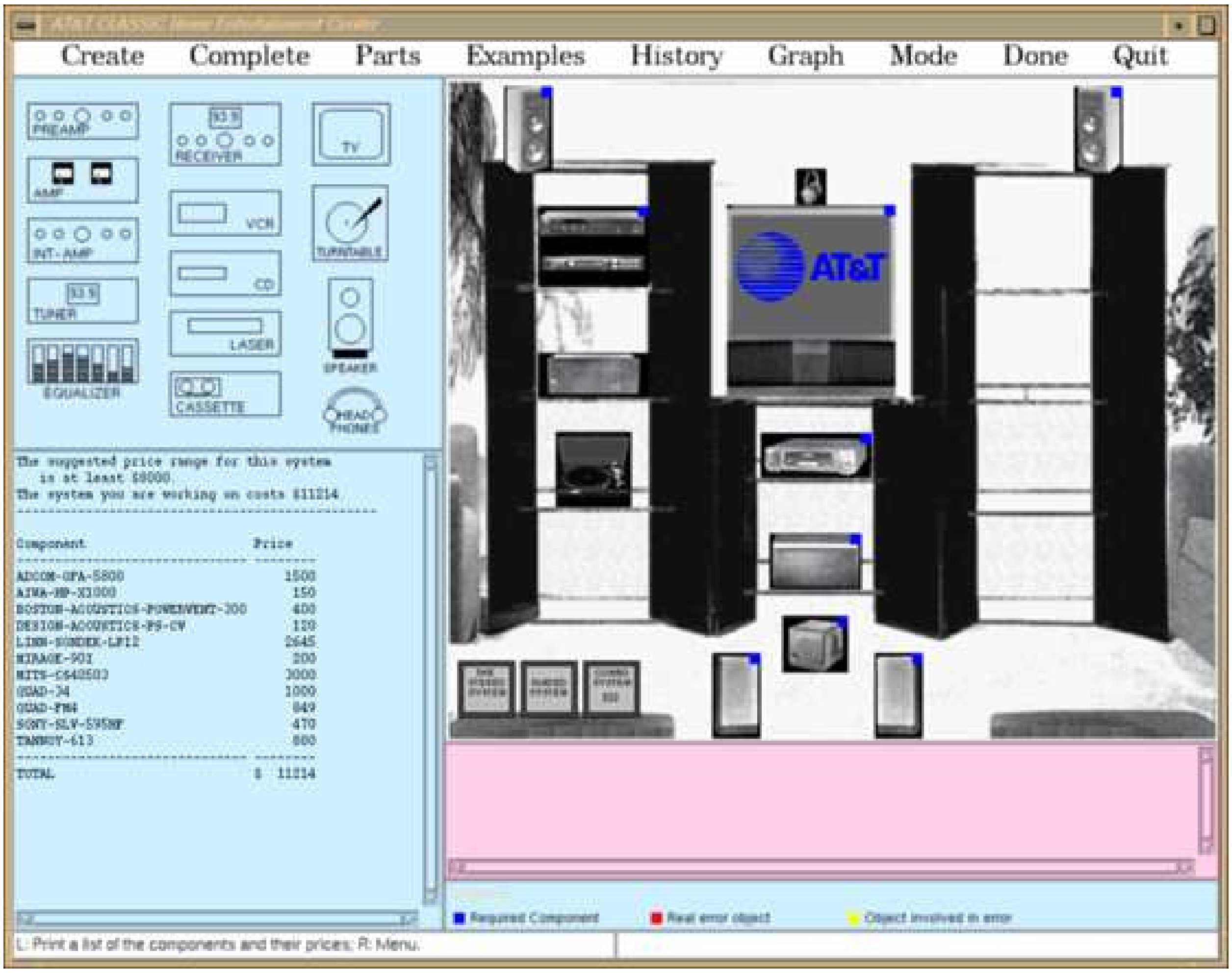

Figure 7: **A completed system.** CLASSIC guarantees that the system above is wholly consistent and will work together.