Lessons from decompiling an embodied cognitive model

Audrey Girouard[1], Noah W. Smith[1], Frank E. Ritter[2]
audrey.girouard@tufts.edu, noah.smith@tufts.edu, frank.ritter@psu.edu
[1]Tufts University (United States)
Department of Computer Science
[2]The Pennsylvania State University (United States)
College of Information Sciences and Technology
Applied Cognitive Science Lab

Abstract
Cognitive models and intelligent agents are becoming more complex and pervasive. It is time again to consider high-level behavior representation languages and development environments that make it easier to create, share, and reuse cognitive models. One of these languages is Herbal, a high-level behavior representation language. Users represent knowledge in Protégé, an ontology editor. Herbal compiles this knowledge into cognitive models in Soar, a rule-based cognitive architecture. Herbal includes the ability to automatically link the resulting models to dTank, a simple, distributed tank game co-developed with Herbal.

To understand the theoretical implications of the process of compiling cognitive models from high-level descriptions more clearly, we generated by hand the Herbal high-level description of a well-written, medium-sized (50 rule) Soar model that plays dTank. This process is a type of decompilation process, of going from low- to high-level language, that yields lessons for both the compiler and the process of modeling. Many of the constructs in the model were supported by Herbal, particularly elaborations and simple and regular actions. In some cases, Herbal's representation prevents the user from generating incomplete, incorrect or atheoretical code—we saw hand written code that cannot be generated by Herbal because it is incorrect or overgeneral. This process also highlights problems with Herbal. Certain types of theoretically sound hand-written rules do not yet possess an exact translation to the high-level language (mostly knowledge about which action to chose, and links across known representations). We have several suggestions for constructs to be added.

This process of decompilation illustrates how users are creating models and could do so more easily and less error prone with more appropriate languages, in addition to helping develop Herbal, and, if automated, this decompilation process done by hand could lead to a decompilation feature in Herbal to help explain raw Soar code.

299 words
04-14-06