

Extracting the Ontological Structure of OpenCyc for Reuse and Portability of Cognitive Models

Bradley J. Best

Nathan Gerhart

Adaptive Cognitive Systems

1942 Broadway Street

Boulder, CO 80302

303-413-3472

{ bjb, ngerhart } @adcogsys.com

Christian Lebiere

Carnegie Mellon University

5000 Forbes Ave

Pittsburgh, PA 15213

cl@cmu.edu

Keywords:

OpenCyc, Ontology, Model Portability

ABSTRACT: *Large scale general-purpose knowledge ontologies, such as OpenCyc, have been suggested as a means of increasing the portability and reuse of cognitive models through a mapping onto domain-independent language. Previous efforts have revealed that this mapping process is difficult to perform due to several factors including the difficulty of understanding the underlying structure of the ontology and mismatches in representation between the target cognitive modeling architecture and the source ontology. We present a method of extracting, pruning, and visualizing the structure of OpenCyc localized around a given set of related terms and explore a set of examples targeted at the representational assumptions of the ACT-R cognitive architecture. Furthermore, we discuss the implications of both a quick-and-easy mapping method and a more robust methodology. The work described, though in its early stages, provides assistance in both rapid understanding of the OpenCyc structure and the process of mapping domain-dependent terms to a general ontology.*

1. Introduction

A central issue in developing a general-purpose layer between simulation environments and cognitive architectures is the representation to be used and its implications for further architectural processing. To attain generality with respect to the simulation environment, commitment to a common, general representation framework is necessary. An additional advantage of this approach is that it should foster on the cognitive architecture side much greater reuse of models than is currently the case. Even for closely related situations, models are usually not reused but instead re-engineered completely to accommodate a different environment. One potential source for such a representational commitment are general ontologies, such as Cyc, that have attracted much investment in recent decades. However, ontologies are fundamentally logic-based formalisms that might not be consistent with the representational, computational, architectural and behavioral commitments made by existing cognitive architectures.

To avoid having the ontological tail wag the architectural dog, it is essential to design a mapping from ontology to representation that is consistent with architectural practice and that leverages the key mechanisms of the target architectures. Ball, Rogers, and Gluck (2004) suggested that the creation of such a layer – the integration of cognitive architectures with general ontologies such as OpenCyc – might provide a remedy to some of the issues involved in cognitive modeling, but they did not go as far as actually implementing such a layer. Best and Lebiere (2009) described a series of issues in integrating intelligent agents into virtual environments and a corresponding set of solutions, some realized, some proposed, that related directly increasing the range and portability of cognitive models, and similarly proposed the integration of large-scale general knowledge ontologies, and OpenCyc in particular, as a means for addressing this issue. This paper describes the current state of our continuing research on this topic, including a functioning implementation of a mapping layer that will be explained in the context of multiple examples. For

specificity's sake, we will focus on the mapping to the ACT-R cognitive architecture (Anderson & Lebiere, 1998; Anderson et al. 2004), but our approach is general enough to apply to related architectures, especially production systems and other symbolic architectures featuring structured representations.

2. OpenCyc

OpenCyc is the open-source version of the Cyc general knowledge base, a large-scale ontology containing both broad general knowledge (e.g., facts relating objects like chairs to their purpose as seating furniture) and specific facts tied to domains (e.g., facts relating specific Army terrain mapping types to the cover and concealment they provide). OpenCyc, created by Cycorp, is written in a proprietary Lisp-like language and includes JAVA and ASCII APIs, as well as a command-line and web-based interface. For more details about Cyc/OpenCyc, see Matuszek et al (2006) and the OpenCyc homepage (www.OpenCyc.org).

3. Extracting Information from OpenCyc

Ontologies are primarily constituted of three types of information: 1) basic terms and their types including hierarchical organization, 2) relations between these terms, and 3) inference rules applying to these terms and relations. Mapping terms and types into ACT-R chunks and their types is reasonably straightforward, but the issue of multiple inheritance across types is much more complex because cognitive architectures typically do not support this mechanism, often limiting themselves, as in the case of ACT-R, to the simpler single inheritance mechanism, for reasons both practical such as efficiency of implementation and theoretical such as cognitive plausibility (e.g., limits on the size of a unit of representation). Basic options to address this issue within the context of the ACT-R architecture include:

- Leveraging the simpler, single inheritance architectural mechanism and treating multiple inheritance in a separate way
- Leveraging other architectural mechanisms such as subsymbolic partial matching and activation spreading mechanisms
- Representing the terms and their types explicitly and requiring that the architecture perform type inferences in an interpretive rather than automatic way

These approaches are potentially complementary, but their implications for processing are fundamental. For instance, the simpler, more explicit and modular representation schemes also impose the most demanding processing requirements upon the architecture. Our research approach is to be strongly guided by behavioral and neural knowledge of representation to derive a robust and effective compromise between these options.

Relations between terms are potentially straightforward to represent but inferences are not. Like terms, there is a natural trade-off between the complexity of the representation and the efficiency of the architectural processes that can apply to it. One possibility is to focus on purely representational issues and consider knowledge-based inferences to be beyond the scope of an interface between environments and architecture. That is often the approach taken in modeling where knowledge and control are tightly intertwined and optimized to the task at hand, but the generality of the representation commitment in this case imposes additional constraints on the necessity to be able to reason upon the knowledge in order to compensate for the lack of hardwired control.

The approach we have taken has 3 main steps, 1) determining an appropriate mapping, 2) pruning an extracted hierarchy, and 3) visualizing the results, each of which are described in detail below. All examples use domain-specific terms from the dTank virtual environment (Morgan et al 2005).

3.1 Determining Appropriate Term Mapping

For any domain, the first (and potentially the most difficult) step is to determine an appropriate mapping from domain-specific terms to the general OpenCyc vocabulary. In section 4, we discuss the implications of two ends of the mapping spectrum: a simple lookup vs. an in-depth exploration of the OpenCyc structure and implied meaning.

Cycorp provides a web-based browser (the KB browser) for exploring and manipulating OpenCyc. Using the KB browser, one can find close matches based on English "pretty strings" (e.g., a search for "tank" returns links to the OpenCyc constants Tank-Vehicle and LiquidStorageTank). Stopping at this result is what we refer to as the simple lookup. Note that the simple lookup mapping procedure uses the domain-specific name as the most important (i.e., the only) criteria.

To perform a more accurate search, one would use the simple lookup as a starting point and dig for more specific constants. It is important to mention here that the full meaning of an OpenCyc term is best understood as a combination of 1) the name, 2) the related (more general/specific) terms, and 3) the "comment" tag associated with the term. For a walk-through of the general search procedure, see the tutorial (Cycorp 2002).

However, a question remains: what feature of the search term is most important? Is it the visual representation of the term in the environment? Is it the name of the term in the environment? Or is it the behavior of the term in the environment? The speed and accuracy of mapping terms onto OpenCyc are impacted by the choice of the most important feature. For example, consider the terrain feature "Woods" from the dTank environment. When interacting with dTank, there is a terrain object that

appears to be made of pine trees and is the same size as the tank. It is named "Woods" by the dTank authors. When an agent is touching the "Woods" object, several things happen: 1) the agent can only travel at a fraction of their maximum speed, 2) projectiles are less likely to hit and damage the agent, 3) the amount of the map that the agent can see is restricted, and 4) the agent is less likely to be visible to other agents (the amount depends on the terrain the other agents occupy).

All four of these features define the "Woods" object in dTank, but it is highly unlikely that we can find a term in OpenCyc that matches all of these features exactly. Therefore, we have to choose the level of accuracy that is sufficient for our purposes. As an illustration, however, we present the process of determining several different possible terms, in increasing accuracy.

If we choose the name, "Woods", as most important, a simple lookup returns `WoodedArea`. The comment associated with `WoodedArea` is "A specialization of `GeographicalRegion`. Each `WoodedArea` is a place with a lot of trees." If we choose the visual representation of "Woods" as most important, a deeper search starting from `WoodedArea` returns `ConiferForest-C4` as a candidate mapping term. OpenCyc describes `ConiferForest-C4` as "A specialization of `ConiferForest`. Each `ConiferForest-C4` is a `GeographicalRegion` that is 75-100% covered with coniferous trees." The good news from this search is that `ConiferForest-C4` is a specialization of `WoodedArea`, so all attributes that apply to `WoodedArea` also apply to `ConiferForest-C4`.

Ultimately, it appears that the most reasonable term in OpenCyc for "Woods" is "`ConiferForest-C3`" (a less dense version of `ConiferForest-C4`). It matches "Woods" on a semantic and visual level. Additionally, `ConiferForest-C3` generalizes to `CanopyClosure-Dense`, `ConcealmentFromAerialDetection-Good`, and `CoverFromDirectFire-Good` (descriptions which closely match the cover and concealment properties of "Woods"). The effect of slowing agents is not quite covered, but the proportion of slowing (50%-75%) is at least similar to the density of the trees. Despite a rather exhaustive search of OpenCyc, our term is still not quite perfect.

The simple lookup mapping for "Woods" is "`WoodedArea`", while the in-depth exploration mapping is "`ConiferForest-C3`". There was substantial work in determining the **single best** OpenCyc term for "Woods"; for a discussion of whether or not it was worth it, see section 4.

3.2 Pruning the Hierarchy

We have created software written in Common Lisp that communicates with OpenCyc through an ASCII API. Once a collection of domain-specific terms have been mapped to OpenCyc terms, we can extract the hierarchical structure from OpenCyc. This structure is a multiple-

inheritance tree with a root at "Thing" (the most general OpenCyc term) and leaves for each of the supplied terms.

Once the web of terms has been extracted from OpenCyc, some amount of pruning can be done; the level of pruning (or possibly expansion) depends highly on the intended use of the web. For instance, a web pruned from the root down to the most specific parent term (Lowest Common Genl or LCG) is a useful way to get an overall sense of the complexity and structure of OpenCyc. Pruning to just the key terms (terms that contain more than one child term) results in significant pruning and is probably the best, most compact way to visualize the relationships of the terms to each other. The resulting web can also be pruned to a single-inheritance tree. The single-inheritance tree may be the most useful for mapping to ACT-R since it matches the single-inheritance mechanism in ACT-R. Visualizations of each method of pruning are shown in the next section, "3.3 Visualizing the Hierarchy".

Our current pruning methods involve selecting nodes and roots for pruning based on the count of leaves reachable from each node. Roots which have child nodes with the same count are removed as a method of automatically finding the LCG. Nodes which have no increased count compared to child nodes are removed as a method of simplifying the branches of the hierarchy. When creating the single-inheritance tree, parents with lower counts are retained; the object is to get the deepest, skinniest tree possible which would correspond to the richest discrimination tree in representation space.

Because the pruning and visualization of the OpenCyc structure is quick and automated, we recommend exploring all versions of pruning and use the resulting visualization to determine the structure that is most useful for the desired task.

3.3 Visualizing the Hierarchy

We have come to the realization that understanding terms and their relationships is nearly as hard a problem as determining a relationship in the first place. Thus, we have invested considerable effort in developing methods for quickly visualizing any mapping of ontology to cognitive architecture.

Our software incorporates the open-source graph visualization software, GraphViz (www.graphviz.org). We translate the OpenCyc structure into a GraphViz-compatible representation of nodes and edges; GraphViz automatically handles the layout and visualization of the structure.

The following figures are representations of different pruning methods applied to the same structure; the OpenCyc structure connecting all of the terrain objects from dTank. For all figures, the green boxes represent the initial list of terms that was used to generate the structure (the user-determined OpenCyc terms). The ellipses

represent the top of the object hierarchy (LCG), and white boxes represent intermediate terms that were extracted due to their connection to both the LCG and the atomic terms. Note that we do not include the completely unpruned structure up to “Thing” as the image is only readable when poster-sized. Indeed, the unpruned structure up to the LCG (GeographicalThing) is barely readable.

Figure 1 is the extracted web of all seven dTank terrain concepts up to their LCG: GeographicalThing. The labels are intentionally unreadable; the point of including the figure is to illustrate the scope and complexity of the hierarchy up to the LCG. Figure 2 is the same structure, but pruned to only the key terms. Only the terms that

directly decompose into more than one term are considered key terms. Notice that the entire left half of Figure 1 is pruned down to GeographicalRegion, OutdoorLocation, and CoverFromDirectFire-Good in Figure 2. Also note that Figure 2 provides just as much information as Figure 1 about the similarities between terms.

Figure 3 is the web from Figure 1 pruned to just single-inheritance. Figure 4 is a fully-pruned version of Figure 3, where only key terms are included. Note that there is very little difference between the two fully-pruned figures; Figure 2 has only one more term than Figure 4, which is CoverFromDirectFire-Good.

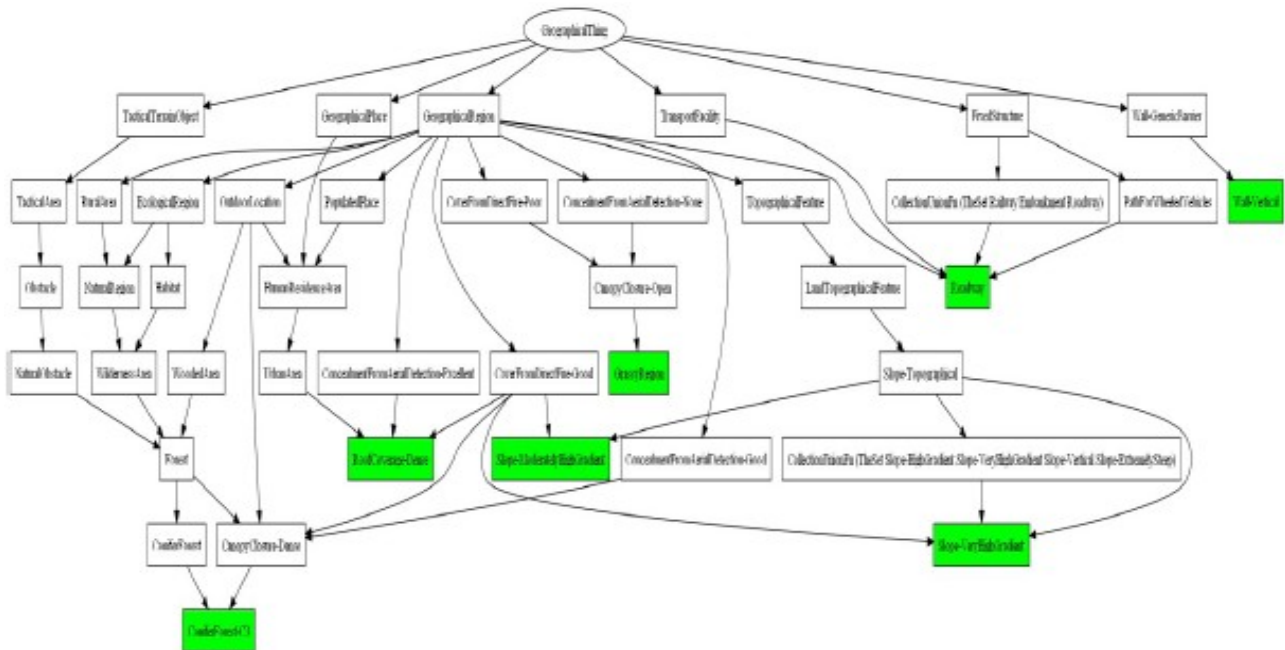


Figure 1: Full Hierarchy of dTank Terrain Terms up to the Lowest Common Genl. Node labels are deliberately unreadable; the same structure (parsed) is presented clearly in the following figures.

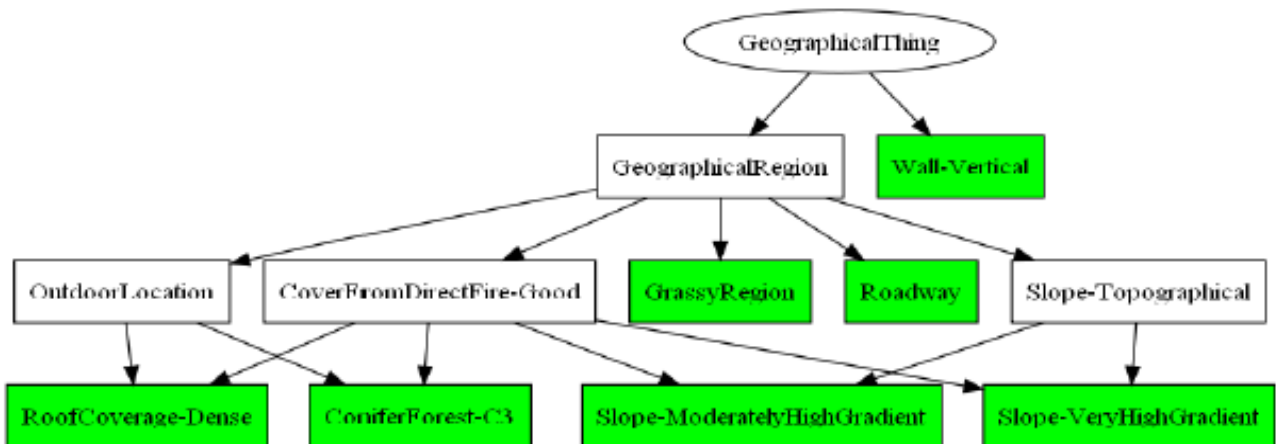


Figure 2: Pruned Hierarchy of dTank Terrain Terms up to the Lowest Common Genl

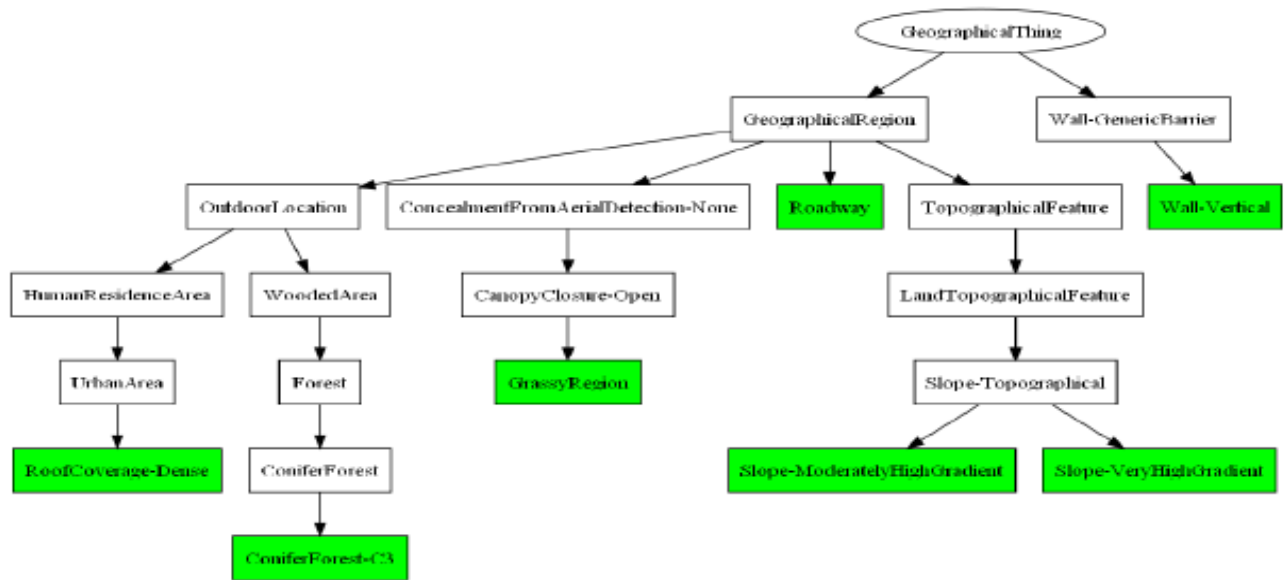


Figure 3: Single-Inheritance Hierarchy of dTank Terrain Terms up to the Lowest Common Genl

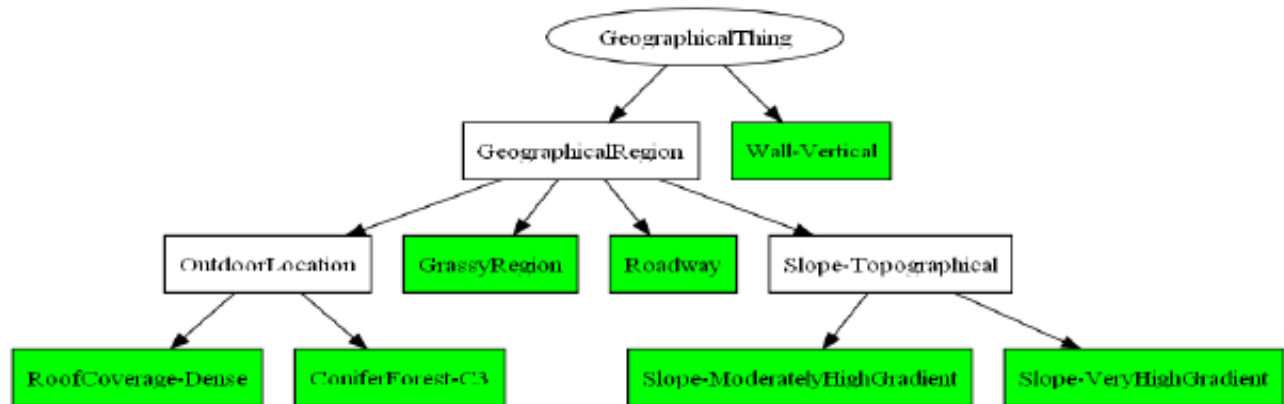


Figure 4: Single-Inheritance Pruned Hierarchy of dTank Terrain Terms up to the Lowest Common Genl

In a single inheritance hierarchy derived from Figure 4, this term would be represented as a relation, CoverFromDirectFire, between various object types and their value, Good. This same relation with different objects types, e.g. GrassyRegion, and values, e.g. Poor, could also be used to represent related terms such as CoverFromDirectFire-Poor. This approach born out of the necessity of leveraging a cognitive architecture with a limited single-inheritance mechanism thus has a number of advantages. First, it makes explicit the semantic relation between apparently unrelated terms CoverFromDirectFire-Good and CoverFromDirectFire-Poor (and CoverFromDirectFire-Excellent, etc.) and thus provides a unification of those terms. Second, it also introduces a distinction between terms in the hierarchy that correspond to fundamental distinctions (e.g., a human-built structure vs. a natural feature), and thus are mapped to the type hierarchy in the cognitive architecture, and those that correspond to superficial, potentially changing features (e.g., a forest provides good cover from fire unless it is sprayed with defoliant) that are mapped to relations binding objects to properties and their values.

However, as previously mentioned, the needs of the user should determine which of the four representations is most useful.

4. Considerations for the Mapping Process

We have chosen to pursue a limited static mapping of terms to the cognitive architecture, largely for performance reasons. Ontologies are logic-based formalisms that often make unreasonable runtime demands upon the systems operating upon them (e.g. rule-based inference). However, embedded agents in real-time environments (as is the case of most of our target environments) are under severe time constraints to produce effective behavior. Moreover, cognitive architectures impose additional constraints upon the space of acceptable processing mechanisms, ruling out some (e.g., logical inference) in favor of others (e.g., subsymbolic mechanisms of activation spreading and matching, adaptive learning processes, etc). These considerations have been extremely important in providing a set of constraints for designing a feasible interaction between OpenCyc and a cognitive architecture.

In the previous section, we presented two vastly different methods for determining a translation from domain-specific terms and their OpenCyc counterparts. All previous figures showed the structure obtained by the in-depth exploration method. The quick lookup mapping of the same terms created a similar, but not identical structure (structure not shown). However, it is unclear as to whether the differences between the two structures present any problems to the main goal, which is model reuse and portability.

It would seem that the time saved in the mapping process (on the order of 15 minutes **per term**) presents a strong case for using the simple lookup procedure. The terms obtained from this procedure are not quite accurate, however, when it comes to describing the behavior of the terms in the specific domain. In fact, one runs the risk of creating a mapping that is still domain-specific, despite the use of generic vocabulary. If the attributes related to the terms are idiosyncratic, then the term cannot be reused in a different domain. The time required to rectify the situation is likely orders of magnitude less than the time needed to create a new model from scratch. Ultimately, the proposed abstraction to domain-independent vocabulary could present a substantial step towards model reuse and portability.

5. Discussion

The choice of whether to perform the representational mapping between OpenCyc and a cognitive architecture statically or dynamically has significant implications. While dynamic access to the ontology and knowledge base is more general, static mapping requires less meta-cognitive management on the part of the architecture and is easier to manage. However, given the size of ontologies such as OC, it would impose significant capacity commitments upon the architecture. The solution we have employed here is a combination of a static mapping of key representational terms with dynamic access to additional knowledge (e.g. inference) as needed. A full static mapping is not, as of this date, feasible within the ACT-R cognitive architecture, but this is a practical limitation rather than a theoretical one and may be overcome as the architecture is applied to larger-scale problems and domain-specific models are integrated into increasingly complex assemblies converging to the knowledge of a human individual (or collective).

Another area where the current paper has been somewhat silent is that of inter-agent communication. The ontological approach taken here might be used to provide a solution not only to the acquisition of information from, and the expression of actions upon, the environment but also to the communication between entities operating in that environment. For instance, plans of action might be expressed using the same terms with appropriate augmentations, potentially allowing even agents developed using different formalisms to communicate

with each other. This use would correspond to the more recent purpose of ontologies, which is to facilitate and integrate communication across electronic media.

6. References

- Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review* 111, (4). 1036-1060.
- Anderson, J. R., & Lebiere, C. (1998). *The Atomic Components of Thought*. Mahwah, NJ: Lawrence Erlbaum Associates.
- Ball, J., Rodgers, S., & Gluck, K. (2004). Integrating ACT-R and Cyc in a large-scale model of language comprehension for use in intelligent agents. In *Papers from the AAI Workshop*, Technical Report WS-04-07, pp. 19-25. Menlo Park, CA: AAI Press.
- Best, B. J. & Lebiere, C. (2006). Cognitive agents interacting in real and virtual worlds. In R. Sun (ed.), *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*. Cambridge University Press; New York, NY, 186-218.
- Cycorp (2002). *OpenCyc Tutorial : An Introductory Walk Through Ontological Engineering*. Retrieved from http://www.cyc.com/cycdoc/walkthroughs/oeintro_cats_frames_long.html.
- Euzenat, J. & Shvaiko, P. (2007). *Ontology Matching*. Springer.
- Langley, P. & Choi, D. (2006). A unified cognitive architecture for physical agents. *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. Boston: AAI Press.
- Lenat, Douglas. Hal's Legacy: 2001's Computer as *Dream and Reality*. "From 2001 to 2001: Common Sense and the Mind of Hal"
- Matuszek, C., Cabral, J., Witbrock, M., and DeOliveira, J (2006). An introduction to the syntax and content of Cyc. In *AAAI Spring Symposium*.
- Morgan, G. P., Ritter, F. E., Stevenson, W. E., Schenck, I. N., & Cohen, M. A. (2005). dTank: An environment for architectural comparisons of competitive agents. In *Proceedings of the 14th Conference on Behavior Representation in Modeling and Simulation*, 133-140. 105-BRIMS-044. Orlando, FL.
- Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard Univ. Press.

Author Biographies

BRAD BEST is a Principal Scientist at Adaptive Cognitive Systems, LLC, in Boulder, CO., where he focuses on cognitive modeling of adaptive behavior in complex environments, especially those that have significant spatial and temporal aspects. His current research interests include integrating perception with decision making in robotic and virtual agents and the development of methods for analyzing, understanding and visualizing model behavior in these environments.

NATHAN GERHART is a Research Programmer at Adaptive Cognitive Systems, LLC, in Boulder, CO. He is the resident subject matter expert for performance in virtual environments and is currently writing software to assist with the exploration and optimization of parameterized models as well as the OpenCyc interaction software discussed in this paper.

CHRISTIAN LEBIERE is a research faculty in the Psychology Department at Carnegie Mellon University. His main research interest is computational cognitive architectures and their applications to psychology, artificial intelligence, human-computer interaction, decision-making, intelligent agents, robotics and neuromorphic engineering.