

Knowledge Re-use in Simulated Entities

Harry S. Delugach Ph.D.
University of Alabama -
Huntsville
Computer Science Department
Technology Hall N-300 Univ. of
Alabama-Huntsville
Huntsville, AL 35899
256-890-6614
delugach@cs.uah.edu

David J. Skipper Ph.D.
Bevilacqua Research Corporation
P.O. Box 14207
Huntsville, AL 35815
256-882-6229
davids@brc2.com

Lisa Cox
University of Alabama -
Huntsville
Computer Science Department
Technology Hall N-300 Univ. of
Alabama-Huntsville
Huntsville, AL 35899
256-890-6614
lcox@cs.uah.edu

Joseph McEniry
Bevilacqua Research Corporation
P.O. Box 14207
Huntsville, AL 35815
256-882-6229
joem@brc2.com

This work was done in support of the Threat Systems Management Office, Redstone Arsenal, Alabama

Keywords:

Knowledge Acquisition, Knowledge Collaboration, Behavior Modeling.

ABSTRACT: *It is a common tenet of simulation development that the entities that are being simulated must behave in a manner that is consistent with the manner of behavior exhibited by the actual entity. For non-sentient entities, which are subject to fundamental physical laws and constraints, their behavior in simulations is reasonably well understood. These behaving entities can frequently be used in multiple simulations, which offsets the cost of developing the physics based components. The behavior of sentient entities is not as clear-cut. While there are behavioral components that are reasoned, there are also aspects that are not reasoned, at least at a conscious level. Despite seemingly clear doctrine or training, human behavior continues to be a complex mix of emotional responses, reflexes and reasoning, all of which are influenced by numerous dependencies on the other entities and the surrounding conditions in the simulation. Consequently, good, realistic behavior models are typically expensive to build due to the emotional and knowledge acquisition efforts involved, which is then followed by the requisite knowledge modeling effort. Such models also tend to lack reusability in multiple simulations, due to their tailoring for specific environments. Simulation developers then face a dilemma: either expend resources or accept lower fidelity. This paper describes techniques being used to overcome this limitation by establishing knowledge re-use and collaboration methods. The approach consists of four parts: 1) Simplified knowledge acquisition, 2) Modeling emotions and knowledge in a standard formats; 3) Developing knowledge repositories that operate with the standard formats; 4) Modeling dependencies in standard formats. An example of the entity operation from knowledge based in a repository is presented, followed by a more general demonstration of simplified knowledge acquisition and knowledge repositories.*

1. Introduction

1.1 Overview

Obtaining human-like behavior from simulated entities is a complex problem. Humans exhibit a mix of inter-related rational and non-rational processes that culminate in a visible set of actions. Since these actions are visible, they are commonly called the *behavior* of an entity. In practice these actions can be

viewed as the external behavior which was brought about by the internal processes or behaviors. Ultimately, a realistic human-like entity must model the internal behaviors to give rise to the observable behaviors in a natural manner, so that a rich, human like interaction can take place in a simulated system. By addressing behavior modeling solely through procedural descriptions, the entity actions will be stiff and scripted, giving a rather stupid appearance.

Unfortunately, developing a detailed view of human-like behavior is not an easy process. This calls for developing detailed conceptual models of not only the logical components of behavior, but also requires developing models of the non-rational, or emotional components of behaviors. By mixing them into a single processing entity, we can then understand the interactions of the composite entity model. There has been a great deal of work in both conceptual and emotional based modeling. However, the acquisition cost for the supporting data and the subsequent integration of these two components into a simulation can be quite high. Even when integrated, there can exist subtle interactions and dependencies, which flavor the exhibited behavior, that do not have clear properties for a model developer.

Bevilacqua Research Corporation and the University of Alabama at Huntsville have developed an ongoing collaboration for extending the behavior of simulated entities. The goals of the collaboration are first, reduce the cost of using advanced realistic human like entities, and second, enhancing the realism by developing suitable modeling techniques for rational and non-rational components along with the subtle dependencies needed for future simulated entities. The focus of this paper is collaborative research efforts on cost reductions through knowledge re-use, with added considerations for explicit model interactions. The paper is divided into sections dealing with several aspects of model standardization, simplified acquisition and re-use through standard repositories.

1.2 Conceptual Graphs

A core technology being used in our work is the knowledge representation of conceptual graphs. We have spent several years in exploring this representation, using extensions and additions as required. Conceptual graphs are a general, robust, logic-based semantic network representation that have been extended to capture procedural knowledge and non-monotonic knowledge (i.e. knowledge that changes over time). For additional information on conceptual graphs, see [1] and [2], as well as various conference proceedings.

Conceptual graphs have a number of advantages for this type of work that have been mentioned previously at this conference [3]. They support general knowledge representation at all levels, interaction with external entities and databases, logical inference, natural language processing, and dynamic (non-monotonic) representations. Another important advantage is their standardized interchange format, which is described in the next section.

1.2 Conceptual Graph Interchange Format (CGIF)

Conceptual graphs are a representation of some aspect of knowledge. That representation is intended to be used by some end user, whether a human or a computer. When the representation is intended to be suitable for easy computer use, the form should be compact, easily parsed and supported by national standards. The conceptual graph representation designed for computer use is the Conceptual Graph Interchange Format (CGIF). It is currently draft proposed American National Standards (dpANS) Standard. The intent is complete the formal standardization within the year [4]. CGIF is fundamentally a text based representation of the conceptual graph, with sufficient flexibility to allow embedding use specific information within the general structure. Since it is a general representation, it says nothing about processing. The knowledge represented by the CGIF based graph may be interpreted and processed by any knowledge processor that is provided with a CGIF import facility. Similarly, any system with a CGIF export facility can create representations that can be processed by entirely different systems. In the long term, translators for other systems, such as CYC would also be possible. If these translators were readily based on the Open Source model, access to previous knowledge engineering projects would be quite feasible. Additionally, since the interchange format is based on traditional American Standard Character Interchange (ASCII) format, it is readily straightforward to design and construct repositories of knowledge. The practical aspects of implementing CGIF are discussed in the later sections.

2. Knowledge Acquisition

Recent years have reflected a distinct shift in thinking when it comes to developing systems that exhibit intelligent behavior. The information used by the system is no longer called *data*, but instead is called *knowledge*. This is not just a matter of terminology: it is becoming increasingly apparent that real intelligence comes from a richer description of an application domain than just recording procedural instructions. Intelligence leads to novel behaviors, dynamic adjustment of goals, re-organization of existing knowledge, and so on.

A key feature of any effective knowledge-based system is its ability to acquire knowledge that is needed. Very simply, if one cannot acquire knowledge, one won't be able to use it. Most present knowledge acquisition techniques are ad-hoc, informal, subject to human bias, error-prone and time-consuming. We have been pursuing an approach that greatly reduces these negative effects.

As we described earlier [3], our approach gets its knowledge using repertory grids to guide a natural language acquisition process. Repertory grids have the advantage of being firmly rooted in human psychological principles. They also have the advantage of being relatively unbiased, in the sense that the person providing the knowledge also provides the structure and terminology of the knowledge acquisition process.

3. Standardization

In this section, we discuss the advantages of standardization in knowledge representations, and describe our experience in implementing a knowledge-based system intended to meet a standard.

3.1 Background

When thinking in terms of future intelligent entities, it is expected that deeply intelligent entities will likely require extremely large knowledge bases (graph sets in excess of 1 million nodes) to provide the requisite realistic behaviors. Clearly, it is not judicious to focus only on knowledge acquisition improvements. If the systems are not fully capable of learning by themselves, then large scale intelligent systems developers will need cooperative knowledge environments, as well as the ability to store and reuse knowledge sets from a variety of sources. A standard knowledge representation should have these advantages:

- Supports re-use of knowledge bases among applications, encouraging a “plug and play” attitude in the use of domain knowledge bases.
- Leverages the large cost of acquiring human behavioral knowledge extending its use beyond a single project
- Encourages additional support tools, such as query engines, explanation systems, and natural language systems that we cannot afford to custom-build for medium-sized projects.

Equally clearly, human behavior is driven by more than rationale thought processes. Developers must have knowledge of the non-rational components. Due to a lack of standard formats for representing non-rational components, only those non-rational components that can be represented by conceptual graphs will be considered here.

3.2 CGIF Implementation

Conceptual Graphs offer several implementation advantages. First, and foremost, among these

advantages is the emerging standardization, the Conceptual Graph Interchange Format (CGIF). Other important implementation considerations are hierarchical structure for abstraction levels and partitioning of knowledge, potentially over multiple processors. Next are actors, which provide clear points for external interaction as well as hybrid system simplification. Finally, there is the graphical basis, for which numerous well understood algorithms exist to permit manipulation and processing over single and multiple processors for large scale systems.

The draft proposed Conceptual Graph Standard defines a knowledge base as a single conceptual graph. However, knowledge bases are then described as consisting of four related contexts, so that the ontology and conceptual structures can be completely defined. According to the standard the elements of a knowledge base are the Type Hierarchy, the Relation Hierarchy, the Outermost Context, and the Catalog of Individuals. While there are no specific directions as to the exact implementation, all of these components are expected to be described using CGIF, so that the knowledge bases could be exchanged and shared, regardless of the underlying implementation or knowledge processor in use. However, while remaining within the standard, assumptions and additions were built in to handle the practicalities of implementing and working with knowledge.

BRC has implemented this representation of knowledge as a collection of files containing CGIF definitions of the knowledge base itself and its four contexts. The basic format of these files has been adapted from the standard, with additional organizational restraints adopted as necessary. These include the use of context keywords and “C” style preprocessing of the incoming text stream. This permits breaking large knowledge sets into manageable components. The following is the six line, CGIF compatible, text stream, to describe a knowledge base:

```
[KnowledgeID : #include "TextFileName1"]  
[TypeHierarchy : #include "TextFileName2"]  
[RelationHierarchy : #include "TextFileName3"]  
[OutermostContext : #include "TextFileName4"]  
[CatalogOfIndividuals : #include "TextFileName5"]  
[CatalogOfActors : #include "TextFileName6"]
```

Four of these keywords are recognizable from the standard. The other two-keyword contexts handle the bookkeeping identification and tracing of the knowledge and the description of actors necessary to utilize the knowledge in a real system. Since these are not specified in the dpANS CGIF, the rationale for these implementation-driven additions is as follows.

First, in the authors' experience with large rule based systems, it was found that traceability and verification of a knowledge system was easier if the key descriptors and identifiers were embedded within the knowledge itself. Second, actors seem essential for implementing dynamic systems, but they occupy a strange position in CGs. In one sense they are simply a kind of relation, yet when implementing an actual actor, a specific actor instance is needed, not an actor type (e.g. not just a database actor, but the specific database to be used and the database actor itself). Thus actors should have a referent field to explicitly identify which instance of an actor they happen to be, in this graph. So they also share properties of concepts. While referent fields have not, but possibly should be, added to the actors for implementation, a catalog of actors implemented in the CG is needed and was used, just like was done with concepts. Similar arguments can be made for a CatalogOfDemons based on the work of Delugach [5] [6]

To continue explaining the knowledge base components above, the "TextFileName_" tells a preprocessor where to find the detailed CGIF descriptions, which define the context. Other internal standards were also developed to facilitate developing real systems. For example, BRC has decided to provide the definition of all concepts within a conceptual graph in the first section of the CGIF representation, akin to providing variable definitions at the beginning of a "C" procedure. Another approach for coordinating users would be to add an optional preprocessor usage appendix to the CGIF standard.

Parsing breaks into two tasks, which are the opposing faces of the same coin. The first task is the reading of CGIF and then, the data object construction of the associated, semantically equivalent internal representation of CGs. The opposing task is the generation of CGIF from internal CG data objects. The internal formats depend on the exact implementation at each developer's site. Therefore only a few comments are in order for the actual parser.

BRC's CGIF Parser has been implemented in conventional manner using the Flex lexical analyzer & Bison grammar matcher. All lexical symbols and grammatical structure definitions are based upon the Extended Backus-Naur Form (EBNF) given in the dpANS CG Standard [4]. Bison tries, ultimately, to match all inputs to a single structure, and in the BRC CGIF Parser, this is a CG. This parser is integrated into BRC's class definition of a CG and into the CORE Tool Kit.

Conceptual Graphs can contain concepts with *designators*. These are nested CGs and a realistic

CGIF parser must accommodate the nested construction of CGs. One problematic aspect of the standard is that all concepts and types must be defined before they are referenced; the knowledge base outline above accomplishes that.

The ability to generate and parse CGIF standard representations is fundamental to using the standard for re-use and knowledge interchange. As more applications conform to the standard, there will be more opportunities for this interoperability.

4. Dependencies

We have found the notion of dependency to be an important aspect in integrating rational and non-rational behaviors. For example, a helicopter pilot under attack may exceed the maximum recommended speed in which to perform a particular maneuver, due to increased tension or a desire to reduce his vehicle's exposure to enemy action. A dependency is therefore established between the rational maneuver and the non-rational tension.

An important feature of dependency analysis is that there are many different kinds of dependency. We have been able to establish some dimensions of dependency and are in the process of developing ways to attach numeric weights or measures to some of them. This will allow a simulation agent to be fine-tuned in choosing its next action(s) to more accurately reflect what a real person or unit would do.

Based upon the values of the attributes, we can then establish a hierarchy of dependency types [7]. Our initial (less than orthogonal) attributes are shown with general definitions in Table 1.

Table 1: Dependency attributes

Attribute	Definition
Need	The need on which the dependency is based. Is usually represented as a list of required capabilities.
Criticality	A measure of the importance of this dependency to the success of the "needing" entity.
Strength	A measure of the frequency of the need/criticality – how often does the need/importance influence operation?
Impact	Possible repercussions of failure at this dependency.
Sensitivity	How vulnerable is this dependency to compromise?
Stability	A measure of the dependency's vulnerability to compromise over time.

This approach to dependency will allow much richer descriptions of simulated entities. We can also model

some non-rational aspects such as fear or cognitive overload by attaching numbers based on functions or tables, rather than have dependency attributes that are strictly linear.

5. Repositories

The success of a knowledge-based system will depend in large part on how "smart" it is. Human activity reflects the accumulation of knowledge over thousands of years, as disseminated through oral and written communication. To be "smart," a knowledge-based system clearly needs lots of knowledge. We have shown how a standard representation is a practical way to store the knowledge. Now we will suggest how large bodies of knowledge (multiple knowledge bases) are to be designed and maintained.

5.1 Background

With the presence of knowledge representation standards, effective storage and retrieval of knowledge is possible between systems. Further, a standard enhances the stability of the representation in time, thus amortizing the effort needed to create a repository over a longer time frame. (For example, if the standard is extended, transforming existing knowledge in the standard is usually accommodated with ease.) The usefulness over time and usefulness across systems are both valuable advantages, but having tools for knowledge discovery in these knowledge bases can offer an additional reason for implementing repositories. Finally, for intelligent entities, standard repositories provide a means for ensuring use of the correct most current knowledge for an object.

5.2 Approach

Use of the CGIF format simplifies some parts of the knowledge repository's design, since CGIF is stored in standard text format. The initial requirements were:

- Store, search, retrieve, and edit knowledge sets,
- Access locally or over a network
- Support for analysis tools
- Support for simulation runtime knowledge delivery

These are readily accomplished with an off-the-shelf database manager. The first version was implemented at BRC using the open source MySQL database manager.

5.2 Future Issues

The field of knowledge discovery is an open area for future research on standard repositories. To support this there are two key areas of automation that have to be addressed in these repositories. The first is the

ability to merge knowledge sets to form larger knowledge sets. The glib response is that this is merely a join of the graphs. In actuality, there is the potential for semantic discontinuities in this that may require extensive human intervention or extensive knowledge processing in the merge process itself. The second problem that must be addressed is the problem of subset extraction. Entities likely will not require all of the knowledge present, in the knowledge sets, either for domain or runtime reasons. The issues of abstraction and sub-graph definition again may require substantial human intervention. When human intervention is required, the utility of the repository to a running simulation is greatly reduced. Fully automating these may require substantial knowledge about the join split process and the interaction with semantics of the knowledge.

6. Summary

We have described in this paper the main components of a knowledge-based system to deal with realistic (i.e., large) bodies of knowledge for simulating intelligent entities.

7. References

- [1] J. F. Sowa, *Conceptual Structures: Information Processing in Mind and Machine*. Reading, Mass.: Addison-Wesley, 1984.
- [2] J. F. Sowa, *Knowledge Representation: Logical, Philosophical, and Computational Foundations*: Brooks/Cole, 2000.
- [3] H. S. Delugach and D. S. Skipper, "Knowledge Techniques for Advanced Conceptual Modeling," in *9th Conf. On Computer Generated Forces and Behavior Representation*. Orlando, FL, 2000.
- [4] J. F. Sowa, "Conceptual Graph Standard," <http://www.bestweb.net/%7Esowa/cg/cgstand.htm>, 2001.
- [5] H. S. Delugach, "Dynamic Assertion and Retraction of Conceptual Graphs," presented at Proc. Sixth Annual Wkshop on Conceptual Graphs, Binghamton, New York, 1991.
- [6] H. S. Delugach, "Specifying Multiple-Viewed Software Requirements With Conceptual Graphs," *Jour. Systems and Software*, vol. 19, pp. 207-224, 1992.
- [7] N. Guarino, "A Formal Ontology of Properties," in *12th Intl. Conf. on Knowledge Engineering and Knowledge Management*, R. Dieng, Ed. Juan-les-Pins, France: Springer Verlag, 2000.

Author Biographies

HARRY DELUGACH is an associate professor in the Computer Science Department at the University of Alabama in Huntsville. He has a B.A. in Chemistry from Carleton College in Northfield, Minn., an M.S. in computer science from the University of Tennessee, and a Ph.D. in computer science from the University of Virginia. He is a member of the ACM, IEEE Computer Society, as well as a past conference chair and current editorial board member for the International Conference on Conceptual Graphs.

DAVID SKIPPER is a software developer with BRC. He has a B.S. degree in Physics from the University of Texas at Arlington and a Ph.D. in Physics from the University of Missouri at Rolla. He is a member of the APS, IEEE, ACM, and AAI. He is also a part time

instructor in both the Computer Engineering and Computer Science Departments at UAH.

LISA COX is a Graduate Research Assistant at UAH completing the Ph.D. program in Computer Science. She received her bachelor's degree in Engineering Physics from Murray State University (Kentucky) in 1984 and her master's degree in CS from UAH in 1990. She has worked in the defense industry since 1984 and is currently Technical Director of LCDC, a software consulting business.

Joseph McEniry is a software developer for BRC. He is also completing his bachelors degree in Computer Science at UAH.