

Implementing Knowledge Interchange for Simulated Entities

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

Joseph McEniry
Bevilacqua Research Corporation
P.O. Box 14207
Huntsville, AL 35815
256-882-6229 ext. 104
joem@brc2.com

David Bjorne
Bevilacqua Research Corporation
P.O. Box 14207
Huntsville, AL 35815
256-882-6229 ext. 103
davidb@brc2.com

Keywords:

Knowledge Acquisition, Knowledge Collaboration, Behavior Modeling.

ABSTRACT: *This paper describes the techniques, which are being developed and used by Bevilacqua Research Corporation (BRC), to address the cooperative development and reuse of knowledge for intelligent simulations. The work is based on the use of the draft proposed American National Standards (dpANS) Conceptual Graph standard that defines a Conceptual Graph Interchange Format (CGIF). This standard, while originated for Conceptual Graphs, offers the flexibility and robustness to describe knowledge. The knowledge description can then be utilized in any application that can understand the dpANS standard. The paper summarizes the state of the current standardization efforts, then discusses parsing CGIF, and concludes by discussing knowledge applications and some implications of the standardization of knowledge representations.*

1. Introduction

The demand for greater realism in simulations forces developers to upgrade the behavior of simulated entities. Consequently, the field of intelligent entities is heavily populated with techniques aimed at providing intelligent behavior in simulated entities. While there are difficult issues associated with using knowledge to produce an efficient, intelligent agent, the more fundamental problem is obtaining knowledge in the first place. Indeed, the volume of knowledge required can become a cost limiting factor unless significant advances are made in knowledge acquisition. Previous work by Bevilacqua Research Corporation (BRC) and University of Alabama Huntsville (UAH) [1] describes an approach developed in recognition of the knowledge acquisition issue. The first component was the development of the Cognitive Reasoning Engine (CORE) Tool set [2] for graphically modeling knowledge. The second component is the ongoing development of Tracked Repertory Grid [3,4,5] Tool set to provide semi-automated knowledge acquisition. The third component is developing methods for collaboration and knowledge re-use. This paper describes these ongoing efforts based on emerging knowledge standards and the application of the standards to develop knowledge servers to supply knowledge to multiple applications.

This paper is organized as follows. The next two sections describe the standard that is the basis for knowledge sharing and the current status and issues

with that standard. The remaining sections deal with the implementation of knowledge as defined by the standard and the implementation of a knowledge server to share knowledge with multiple applications.

2. Draft Proposed Conceptual Graph Standard

Conceptual Graphs (CGs) are based on the work of John Sowa [6], who in turn based his work on the graphical logic of Charles Peirce [7]. With a foundation in logic representation, CGs also have first-order and predicate logic operations defined allowing for inference and theorem proving procedures.

Visually, a Conceptual Graph (CG) mimics the knowledge representation ability of diagrams used in discussions using whiteboards, slides, and napkins. These drawings are often text snippets (typically enclosed in squares or ovals) and lines (possibly with a label) connecting one snippet to another. Experts often use these visual aids to quickly and effectively communicate complicated, technical details during brainstorming sessions. In CGs, text snippets are called Concepts, and line connections are called Conceptual Relations. A Concept may also contain another CG to provide contextual or nested information.

2.1 Standard Overview

The current version of the Conceptual Graph Standard is a draft proposed American National Standard

(dpANS) [8]. As such, it has a few sections that are still “under construction” and some sections that will be modified to eliminate inconsistencies or to expand certain features. At present, the standard defines the following:

- Terminology related to CGs
- Formats in which CGs may appear
- Core abstract syntax of CGs
- CGIF as a transfer format
- Extended syntax and mappings into CGIF
- Logical foundations and operations

2.2 Open Issues

While most of those items are well defined for the current version of the standard, there are still refinements being discussed on the CG mailing list, cg@cs.uah.edu. While discussions cover some esoteric matters, the focus is mainly on the CGIF grammar definition, because it provides an information system transfer format that permits CG examples to be transferred electronically. This provides a means for continuation of discussions of the other, highly abstracted, issues.

CGIF is, at present, still under refinement because it has some borderline ambiguous grammatical constructs and some implied/unmentioned lexical categories. There are several topics being addressed to allow for cleaner and easier parsing:

Constituency of Label usage among node types can enforce consistency in grammar structure construction.

Comprehensive designations of begin symbols that expand the lexicon categories can remove ambiguity of tokens even with a single look-ahead parser.

A single comment field with unique begin and end markers can make the lexing of comments simplistic, but association schemes still need to be strictly defined.

The CGIF grammar definition makes use of some lexical symbols that are subsets of various defined symbols, and for clarity these should be defined separately.

Other important issues being addressed or needing additional attention are:

- Use of Encoded Literals
- Enumeration of Defined Quantifiers

- Role of Actors
- Procedure for Projection Operator
- Translation of Lambda Expressions

3. Conceptual Graph Interchange Format (CGIF)

Intelligent systems in the past have generally been based on rule base systems for knowledge representation and knowledge utilization. Initially, another draft proposed standard was started for computer exchange of knowledge, and it was called Knowledge Interchange Format (KIF). KIF was designed to exchange rules for rule based systems. Due to the greater expressive power of the Conceptual Graph approach and the fact that propositions can be represented as concepts, BRC determined that KIF was inadequate for planned applications and decided to utilize the CGIF format, rather than the KIF format, for exchanging knowledge.

3.1 CGIF Overview

CGIF is defined specifically for complete and effective transfer of CGs between information systems. It uses an American National Standards Institute (ANSI) character set, so that virtually any computer system that can interpret ANSI text could parse CGIF. The CGIF definition is based on the CG abstract syntax definition, so CGIF is at most as complete as the CG definition is. Because of the requirement for effective transfer in a digital form, CGIF is designed for single-pass parsing and it is not particularly suitable for human reading. For a complete definition of CGIF, see section 6 of [8] available at:

<http://www.bestweb.net/~sowa/cg/cgdpansw.htm>

A brief description of just a few core components is provided here. In CGIF, a CG can contain one or more Concept, Relation, and other. A Concept is enclosed in brackets and may contain a typelabel, a set of co-referent links, a referent, and a comment. A Relation and its connections are enclosed in parenthesis, and it requires a type label and may contain a comment. The following example of a CG in CGIF is straight out of the CG Draft Standard:

```
[Rock *x1] [Place *x2] [Person *x3] (Betw ?x1 ?x2 ?x3)[Hard ?x4] (Attr ?x2 ?x4)
```

This example’s English equivalent is: “There exist a particular rock, a particular place, a particular person, and a particular hardness such that the person is between the rock and the place, which has the attribute

of being that hard.” More naturally, this would be: “A person is between a rock and a hard place.”(Figure 1)

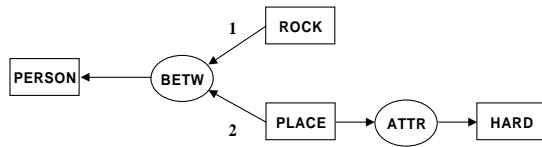


Figure 1 Between Example

3.2 CGIF Parsing

BRC has implemented a CGIF Parser using Flex & Bison versions ported for Win32. Flex is a lexical analyzer designed to pass successfully matched tokens read in from an input stream onto Bison, which is a rule based grammar matcher. Bison matches tokens to simple grammatical structures, and these are combined with other structures and more tokens to form more complicated grammatical structures. All lexical symbol and grammatical structure definitions are described using Extended Backus-Naur Form (EBNF) which is given in [8]. With each match, user provided code, if any is to be executed. Bison tries to ultimately match all input to a single structure. In the BRC CGIF Parser, this is a CG. When CGIF structures are matched, the parser makes extensive use of C++ methods to construct BRC’s CG internal format.

Flex allows the use of states to support different token matching schemes among a finite set of contexts. An example of a special condition, where symbols need to be interpreted differently, is within a comment. State management is complicated by the fact that some grammatical structures can be encapsulated in different contexts, such as a Concept residing directly in a CG or in a Relation. In BRC’s parser, complication is overcome through the use of a state stack to keep a memory of the states so far. The handling of the stack must take into consideration that there are times when one symbol ends two states as in the case of the close bracket for a Concept with a Referent.

Since CGs may contain Concepts with Designators, which are CGs, a CGIF parser must allow for this nested construction of CGs. BRC’s parser handles this through the use of a lexer special sub-state of the Referent state, a stack of CGs in the grammar matcher, and careful timing of node addition to the top CG on the aforementioned stack. When a beginning symbol for a Concept, Relation, or Actor is met within the Referent of a Concept this indicates a CG is contained within that Concept. This changes the Referent state from the “Referent without CG started” sub-state to the “Referent with CG started” sub-state and a new CG is dynamically created and pushed on the stack of CGs. A

CG exists on the stack until it is completely matched, and nested CGs can be added, completed, and removed while its owning CG is residing on the stack. Nodes are added to the top CG, when all grammatical structures that define them are matched. This ensures that any contained (nested) CGs of a Concept will have all of their nodes added to them before the owning Concept is added to its CG.

The most complicated aspect of implementing a CGIF parser is determining how Co-referent Sets should be parsed and used. Co-referents establish an identity connection for Concepts across nesting boundaries, although they are also used to indicate a single Concept is connected to multiple, co-nested Relations. BRC has implemented them so the only time when a Concept is NOT created for a bound label is in the case of a bound label immediately contained in a Relation that is co-nested with pre-existing Concept that is a part of that co-referent set. In BRC’s CGIF Parser, Co-referents are maintained with a collection of Co-referent Sets and an associated collection of Label Identifiers. A Co-referent Set is created for each Defining Label, which is assumed to be a Dominant Concept of the Co-referent Set, and the Set is added to the collection of Co-referent Sets of the CG that contains the Dominant Concepts. This occurs when the CG is matched as a grammatical structure, and a check is made against all existing Co-referent Sets for the CG’s unique identification in the CG ID field of the Concept that is the first Dominant Concept in that set.

3.3 Open Issues

Because the CG Standard is still in draft form, any CGIF Parser is also a draft version. BRC is currently working towards the completion of the standard. Until the standard is complete, BRC's parser will be a functional, yet “in progress”, version. One restriction of BRC's CGIF Parser (done to simplify the handling of Defined and Bound Labels and Co-referent Sets) that is not inherent in the CGIF definition is the requirement of each Bound Label to have an associated Defined Label predefined. This will generally not require any special CGIF stream construction because the Define Label will normally come before one of its Bound Label. One approach to ensure Define Label predefinition is to begin every CG by listing of each of its Concepts with an associated Defining Label.

BRC is in the process of developing test cases to determine the rate of memory consumption by the parser. Bison has its own limits on the number of constructs that are put on its parser stack before the final match, but the Bison parser stack does compress many simpler structures as they combined to form the

more complex structures. These structures end up being references to the dynamically created Conceptual Structures, and the number of these items will, of course, be restricted by available memory. This will become a very important issue as even a small number of groups begin collaborating on the acquisition of larger knowledge domains.

4. Knowledge Bases

4.1 Definition

The draft proposed Conceptual Graph Standard defines a knowledge base as a single conceptual graph. However, knowledge bases are also 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:

1. **Type Hierarchy**, which describe the relational linking (lattice) of types within the knowledge base,
2. **Relation Hierarchy**, which describe the relational linking (lattice) of relations within the knowledge base,
3. **Outermost Context**, which is a top-level conceptual graph describing the subject of this knowledge base. Descriptors of concepts within the outermost context can define other conceptual graphs, providing a multi-tiered level of knowledge.
4. **Catalog of Individuals**, which contains a disconnected series of contexts defining specific concepts defined throughout the knowledge base

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.

4.2 Implementation

During the initial development of CORE, the emphasis was on the simple graphical creation of conceptual graphs. At that time, the system understood both Sowa's Linear Form and an internal BRC graph format. While the capability existed to combine graphs, limited attention was paid towards how these multiple graphs actually related to each other or the

difference in a graph and a knowledge base. As the standard has evolved, a conscious decision was made to evolve representations to the CGIF standard so as to gain interchangeable, shareable knowledge bases. As work has progressed, CORE is evolving to better handle the representation of knowledge as described in the conceptual graph standard. However, while remaining within the standard, assumptions and additions were built in to handle the practicalities of 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. 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 [6] and demons [9, 10] necessary to utilize the knowledge in a real system. The "TextFileName_" tells the preprocessor where to find the detailed CGIF descriptions, which define that context. Other internal standards were also developed to facilitate 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. BRC encourages other developers to exchange implementation ideas and specific keywords to ensure that parsers understand the desired use without explicit user intervention.

CORE has been adapted to provide graphical and textual editors for the creation of each piece of a knowledge base. Validity checks for each separate part of the knowledge base are accounted for in the individual components. The four pieces may be created separately, and joined at a later time using a simple dialog. Once created, a knowledge base can be reloaded and modified as necessary, then shared with other users.

5. Knowledge Servers

Emergence of viable standards for knowledge bases opens the real possibility of sharing/re-using knowledge and collaborative development of knowledge. However, like software re-use, this will not occur without an investment in the technology and techniques to “make it happen.” The obvious viewpoint is that a knowledge server is a database of knowledge bases. From this the basic requirements are straightforward. The database must contain the knowledge base elements described in the previous section for each knowledge base and maintain the ability to export and import the knowledge bases using CGIF. The knowledge server must have the means to insert, modify, and remove knowledge bases. It must have the ability to find knowledge based on a variety of keys, including subject, authors and keywords. It must also contain knowledge management information to provide categorization and tracing of knowledge. Next, it must contain the capability to seek other servers and to query other knowledge servers for knowledge bases. Finally, a knowledge server should have the basic tools to test and analyze the knowledge bases contained within the knowledge server for consistency and validity. Finally, tools to learn new knowledge from the residence knowledge bases are needed.

BRC has taken a first step and begun development of an initial knowledge server to provide behavioral knowledge for intelligent entities on demand. The basic design is shown in Figure 2.

6. Summary and Future Directions

BRC has adopted a standard for exchanging knowledge with other applications and users. Based on that, BRC has implemented a parser to read the standard format into BRC tools. BRC and UAH also plan to extend the use of this standard to other areas

BRC maintains a goal of developing highly intelligent entities and agents. This has given specific directions to many efforts at BRC. The developments frequently focus on:

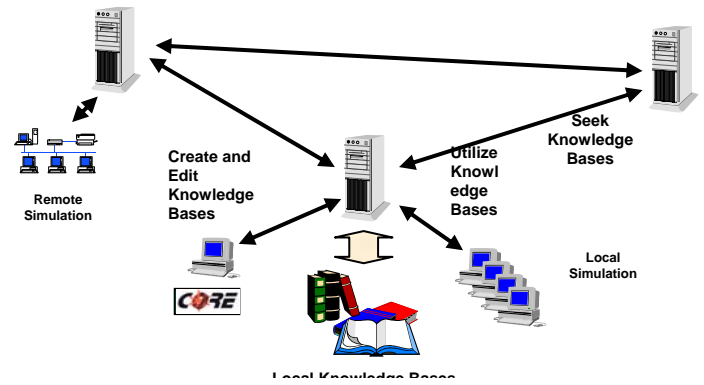


Figure 2 Knowledge Servers

- Organization and utilization of very large knowledge bases,
- Acquisition of very large knowledge bases

Future developments at BRC will focus on combining of individual knowledge bases into larger and extracting smaller knowledge bases. Additional work will focus on meta-knowledge bases. BRC is also working on automated acquisition and validation tools based on the use of the knowledge descriptions in [8]. Finally, cooperative development and sharing through development of knowledge servers is an ongoing collaborative effort between BRC and UAH.

7. References

1. Delugach, H.S. and D.J. Skipper Knowledge Techniques for Advanced Conceptual Modeling in CGF 2000, Ninth Conference on Computer Generated Forces and Behavioral Representation. 2000. Orlando, Florida.
2. BRC, Cognitive Reasoning Engine (CORE) Toolkit Users Manual. 2000, Bevilacqua Research Corporation, BRC-UM-001/2000: Huntsville, AL.
3. Wolf, R.P. and H.S. Delugach, Knowledge Acquisition via Tracked Repertory Grids. 1996, Computer Science Dept., Univ. Alabama in Huntsville.
4. Wolf, R.P. and H.S. Delugach. Knowledge Acquisition via the Integration of Repertory Grids and Conceptual Graphs. in ICCS '96, 4th Intl. Conf. on Conceptual Structures. 1996. University of New South Wales, Sydney, Australia.
5. Delugach, H.S., Repertory Grid Graphs: A Hybrid of Conceptual Graphs and Repertory Grids, in Proc. 11th Knowledge Acquisition Workshop (KAW-98), B. Gaines and M. Musen, Editors. 1998, University of Calgary: Banff, Alberta, Canada.
6. Sowa, J.F., Information Processing in Mind and Machine. 1984, Reading, MA: Addison-Wesley Publ.

7. Peirce, C.S.: Collected Papers of Charles S. Peirce, Hartshorne, Weiss, and Burks (eds.), Harvard University Press, Cambridge, MA, 1932
8. Conceptual Graph Standard Information Technology (IT) - Conceptual Graphs draft proposed American National Standard (dpANS) NCITS.T2/98-003.
9. Raban, R. and H.S. Delugach, Animating Conceptual Graphs, in Conceptual Structures: Fulfilling Peirce's Dream, D. Lukose, et al., Editors. 1997, Springer-Verlag. p. 431-445.
10. Delugach, H.S. Dynamic Assertion and Retraction of Conceptual Graphs. in Proc. Sixth Annual Workshop on Conceptual Graphs. 1991. Binghamton, New York: SUNY Binghamton, New York.

8. Author Biographies

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 AAAI. He is also a part time instructor in both the Computer Engineering and Computer Science Departments at UAH.

Joseph McEniry is a software developer with BRC. He is completing a B.S. degree in Computer Science at the University of Alabama in Huntsville.

David Bjorne is a software developer with BRC. He has a B.S. degree in Electrical Engineering from the University of Alabama in Huntsville.