

# Conceptual Graph Based Intelligent Decision Aids

H. Barbara Sorensen Ph.D.<sup>\*</sup>

*U.S. Air Force Research Laboratory, Mesa, Arizona, 85212*

Debra C. Evans Ph.D.<sup>†</sup>

*Cognitive and Behavioral Systems, Baden, Pennsylvania, 15005*

Harry S. Delugach Ph.D.<sup>‡</sup>

*University of Alabama - Huntsville, Huntsville, Alabama, 35899*

*and*

David J. Skipper Ph.D.<sup>§</sup>

*Bevilacqua Research Corporation, Huntsville, Alabama, 35802*

**This paper describes techniques that are being developed jointly by Cognitive and Behavioral Systems (CABS), Bevilacqua Research Corporation (BRC) and the University of Alabama, Huntsville (UAH) to address the knowledge engineering problems of today's decision aids. These techniques are based on the original Conceptual Graph work of Sowa. Conceptual Graphs provide distinct advantages in semantic clarity, perceptual flexibility, and extensive reasoning capabilities. These developing techniques are applied to distinct models through three components. The techniques are first applied to the design of highly detailed situational knowledge models. Second, these techniques are applied to detailed operator knowledge models. The final component is tailored to perceptual and reasoning interactions of the operator models and the situational models that can lead to adaptive intelligent decision aids. This paper will describe the methods and the tools used by CABS, BRC and UAH to develop decision aid systems.**

## I. Introduction

THE development of decision aids had a tremendous leap in capability and availability when digital computer based decision aids evolved. This placed basic decision assistance within the average person's grasp, leading to a wider usage. The emergence of smaller, more powerful low cost computers opens the door to significant improvements in the intelligence possessed by the decision support system. This, in turn, offers the potential for easier use and better understanding between the human and the machine, that can provide a synergistic problem solving capability more in the lines of groups of cooperating human experts, rather than a more limited "human with a spreadsheet" level of interaction. Of course, this assumes that both human and machine bring sufficient intelligence and perception to understand and cooperate with their opposite. While the human side usually provides a sufficient intelligence level for this type of interaction, the computer side intelligence is a continuing difficult problem. In this paper, we examine, from our perspective, some considerations when trying to develop intelligent decision aids. Because we are focusing on this problem as one of cognitive intelligence, we will not pursue the important and complex issues of the actual interfaces with the human, such as speech, hearing, touch and vision.

This paper is organized in the following manner. In the next section we begin by examining Conceptual Graphs of Sowa<sup>1</sup> as a knowledge representation method that underlies all of our work with intelligent systems. This is

---

<sup>\*</sup> Senior Research Scientist, US Air Force Research Laboratory, 6030 S. Kent St., Mesa, AZ 85212, Member

<sup>†</sup> Principal, 127 Lovi Road, Baden, PA 15005

<sup>‡</sup> Associate Professor Computer Science Department, Technology Hall N 300, Univ. of Alabama in Huntsville, Huntsville, AL 35899

<sup>§</sup> Director Information Systems, Bevilacqua Research Corp., 1900 Golf Road, Ste F, Huntsville, AL 35802

followed by a brief discussion on decision aids and our perception of their shortcomings. We then illustrate the application of Conceptual Graphs to decision aids.

## II. Background

### A. Conceptual Graphs

Conceptual graphs are the culmination of several directions in cognitive science, semantic networks, natural language processing and formal logic. These topics were synthesized into conceptual graphs by John Sowa in his original book. Conceptual graphs are intended as a general knowledge representation to help solve problems in natural language understanding, human decision-making, behavioral modeling, and many other important areas of interest.

We consider a conceptual graph to be depicted on a flat surface called the *sheet of assertion*. If several things appear on the sheet of assertion, then all of them are in effect at the same time. In a simple world, a single sheet of assertion may suffice to represent all the knowledge needed to solve a problem; however, in general there may be multiple sheets, which together form a *knowledge base*. The term *graph* can have several meanings, depending on how it is used. Strictly speaking, a graph is any collection of conceptual graph elements that are treated as one thing. Often the term graph will mean a set of linked or nested elements, though unconnected elements still may belong together for some purpose. We now consider those elements that make up a conceptual graph.

A *concept* is a basic building block for a conceptual graph. A concept is any instance of a thing. Consider it to be a basic unit of thought. A concept can be an entity, such as a person, a characteristic of an entity such as height, an idea such as justice, a composite of several related concepts (called a context) or something of an unknown type such as *thing*. A concept may have several parts associated with it, such as a **type** and a **referent**. A concept is represented as a labeled rectangle in a conceptual graph. The referent field, when present, is separated from the type field by a single ‘:’ Examples (Figure 1), where color has been added here for emphasis, are:



Figure 1

Each concept has two main components: a *type* and a *referent*. The type tells what kind of thing the concept is, whereas the referent identifies which thing the concept represents. A *type* is denoted by a *type label*, which indicates what kind of thing a concept is. Types are arranged in a hierarchy. We can generalize among types. For example, something of type **Dog** may also be of type **Animal** and **Pet**. There are several different ways in which the thing can be identified in the referent. The thing can be identified by name **Person: Daniel**, or by some identifying code in a computer system **Picture: #4827**, or by a symbol for purposes of reasoning **Point: \*a**, or by a reference to an actual thing (e.g., a link to a picture or a tax identification number).

A *relation* denotes some relationship between two or more concepts. It is depicted as a labeled circle, oval or rounded rectangle. Directed lines, which connect the relation with the concept, depict the direction of a relationship. Thus the person Daniel owns the chair (Figure 2).



Figure 2

Sowa lists a pre-defined set of relations typically used in typical conceptual graph systems. A conceptual graph system would expect that both a relation’s name and the associated concept types are specified in a usable predefined list. An *actor* is a functional transformation relation between concepts, shown as a diamond. The inputs to the function are the referents of linked concepts. The outputs determine by the functional transformation determine the referent value(s) of other linked concepts. In its simplest form, an actor represents a functional relationship, where the identity of related concepts is constrained by some other concepts. Another, more powerful use of actors is as a “hook” or window to the part of the outside world being represented by the conceptual graph. Actors then provide a means of changing our knowledge base, keeping it current with what is actually going on in the real world. Thus our conceptual model of the world changes as the world changes.

It is often the case that a system will have more than one statement to make about a particular concept. For example, we may want to say: “The glass is on the table” and “The same glass is full of milk”. For convenience and

sometimes for ease of modeling, the statements can be given separately, as long as there is a way to show that the apparent two glasses are, in fact, the same glass. Conceptual graphs accomplish this with a construct called a *co-referent link* or *line of identity* that links two or more concepts that refer to the same thing.

Since two or more separate concepts in a system may refer to the same individual thing, there may be a set of concepts that all share the same identity or are linked to the same identity. The set of such linked concepts forms what is called a *co-referent set*. A conceptual graph system may have many such co-referent sets, each of which refers to a distinct individual thing.

Types in a conceptual graphs system are arranged in a *type hierarchy* (similar statements can be made for relations) or partial ordering. Just as in most type- or class-oriented approaches, sub-types inherit characteristics from their super-types, forming an “is-a” or “is-a-kind-of” specialization relationships among them as shown in the following Figure 3.

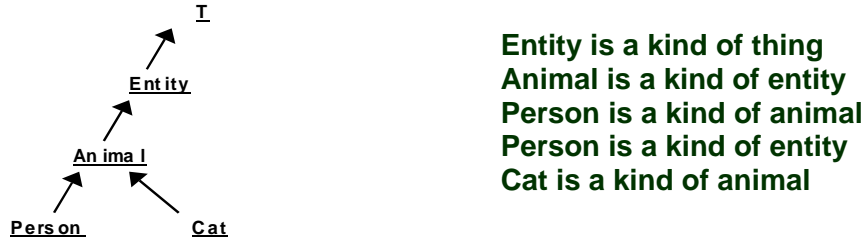


Figure 3

We are then able to reason with or make assumptions about super and subtypes based on knowledge about concepts within the hierarchy. This leads to replacement of concepts in a conceptual graph with acceptable super or sub-types as a part of graph reasoning.

One important idea that makes conceptual graphs especially suitable for large-scale systems is the idea of a graph *context*. With contexts we can show knowledge about knowledge, rather than just knowledge itself through graphs. For example, the figure below shows the knowledge that “it is false that a pig flies”. The context, shown inside an enclosed rectangle, represents the graph that “a pig flies”; by negating the entire context, we have a relationship about the entire statement, not just the **Pig** concept or the **Fly** concept shown in Figure 4.

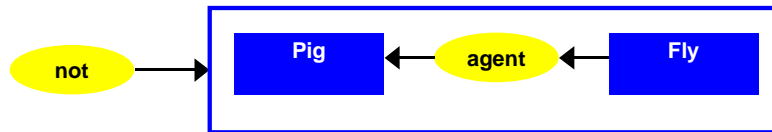


Figure 4

Once some knowledge has been encoded into conceptual graphs, we can use the formal representation to do searches, reasoning and inference on the knowledge. The biggest advantages of conceptual graphs is that these representations can be used to answer questions whose form is not yet known, and that they can be used to develop inferences of new knowledge without having to know the form of the new knowledge.

## B. General Suitability

In the previous section we discussed the basic components and capabilities of conceptual graph systems. While the following sections discuss the general nature of decision aids, we can intuitively examine conceptual graphs for those properties that may be useful in human computer decision systems. We shall accomplish this by loosely defining the meaning of an intelligent system.

We shall require that a system that is labeled as intelligent shall be capable of representing, storing and using diverse, complex sets of knowledge. The system uses knowledge by identifying and retrieving knowledge based on need and context in both a short-term reactive, or perceptual sense and a longer term reasoned usage. Using the knowledge also implies managing the knowledge to adjust, aggregate or specialize the knowledge itself to a task at hand.

Given this emphasis on knowledge representation and knowledge usage, as well as the cost of acquiring knowledge, a simple means of exchanging and reusing knowledge, while not a direct part of intelligence, is a highly

desirable capability. Further, when other computers are involved, the exchange channels should not require translation into human speech or gestures, rather it should be a fast direct “computer mind-to-computer mind” form. Given these high level requirements, what about Conceptual Graphs? First and foremost, conceptual graphs provide a means of representing, storing and exchanging knowledge about a given domain. In addition, Conceptual Graphs provide a basis for well-established formal reasoning about that same knowledge. Closely related to the formal reasoning comment, the graphical nature of the knowledge representation also supports both perceptual recognition through a diversity of graph matching methods as well as a similar variety of queries through graph matching.

Finally, recognizing that intelligence requires managing and growing knowledge, we can assert that there are extensive methods for knowledge mining, adaptation and learning with Conceptual Graphs. Having noted that intelligent systems could be constructed from Conceptual Graphs, we now focus on decision aids.

### III. Decision Aids Essentials

#### A. Types of Decision Aids

What is a decision aid? In general, a decision aid can be said to be any tool that makes a decision task more manageable. While typically we refer to an individual’s decision, collaborative decision aids are those tools that make a group’s decision task more manageable. Throughout history, people have used many things to help them with decision support—from cuneiform ticks on clay tablets for counting, lists of good and bad points of different decision possibilities, maps for way finding, spread sheets to aid in presenting quantitative data, notes, decision tables or trees, to advice from others, *et cetera*, depending on the task, time and availability of support. With the advent of electronic-based technologies, these methods for supporting decisions have been updated to use the new presentation methods.

##### 1. Active or Passive

Decision aids, or decision support systems as they are often called, can be classified in several ways. For instance, they may be sorted by the scope of the task’s decision process as handled by the aid versus the scope handled by the human. Other classifications may be the degree of memory support or informational organization for the task performer; for example, lists, data envisioning tools, and maps are of this nature. In contrast to these “passive” aids there are directive or “active”. Examples are decision trees, expert systems, and human expert consultants, which provide direction with regard to the decision.

##### 2. Tool Source

There are other classifications such as “tool source”. Aids may be separated into groups by tool development responsibility. Many people develop, over time, their own personal decision tools to aid them in job performance. An example would be constructing a graph to represent information contained within a spreadsheet, or making notes or lists to guide them. On the other hand, people, other than the user, develop many tools for use by task performers. This group of tools would include such tools as displays of complex, multi-faceted information, expert systems, and decision trees.

##### 3. Knowledge Source

Another dimension, on which decision support tools may be arranged, is with regard to the gathering of information to be used by the tool, or “knowledge source”. The tool user might be the one responsible for the gathering and input of information; this is the case with databases, notes, and spreadsheets. With other classes of aids, the tool might encompass a set of choice points and directions for data gathering based on previous input, but the user must still furnish the information to be processed. In the most automated situations, the tool might share or have total responsibility for data gathering, in addition to presenting it and, possibly, guiding the decision making.

##### 4. User Types

Finally, one may identify decision aids by their target users or audiences. A decision support system that is to be used by a novice task performer is likely to have different characteristics than one used by someone more experienced with performing the task. A novice may require directive decision support for both routine and non-routine tasks, whereas a more task-experienced person requires less support during routine task performance than does a novice. However, a more skilled performer may still require support for non-routine tasks or those in which information needed to make the required decisions is degraded because of lack of availability or reliability.

Although there are many different tools that can be used to help us in decision making, the tool type focus here, and the most pertinent to aeronautics and astronautics, are those that can:

- Guide users in the performance of critical or non-routine tasks;
- Supply methods for organizing and presenting large amounts of data (possibly with aiding data-gathering tasks);

- And/or supporting decision making in situations in which information is degraded in some fashion.

Decision aids that reflect these characteristics are frequently automated.

Notice that computer help systems are not included in the list of types of decision aids. The reason for this is that help systems, as they are currently designed, do not aid decision-making, they explain the tool usage. They are used for accessing information that may then be used for decisions, in the same way that one would access a dictionary for information. Additionally, one should not consider training to be a decision aid. It can, and should, serve the process of teaching decision-making skills and the use of decision aids, but in itself, training is not a decision support system.

## **B. Design Examples**

The number of intelligent decision support systems or aids is expanding rapidly. In almost every domain, there can be found at least one research group developing such systems. A recent web search resulted in more than twenty pages of web sites containing the phrase “decision support systems.” There were more than twenty other pages that included the phrase “decision aids.” It is not within the scope of this paper to present an exhaustive discussion of these products. However, two different, but representative, systems have been selected for presentation in the following paragraphs.

The first system is the Integrated Weather Effects Decision Aid (IWEDA). The Army Research Laboratory (ARL) has recently developed IWEDA to aid in tactical decision-making by commanders with regard to weapons deployment (Sauter, et al.<sup>2</sup>). The function of IWEDA is to transform collected, raw weather data, which has been provided by models or analyses, into graphically presented and analyzed weather information appropriate to the current tactical situation. Thus the commander can have weather intelligence for both friendly and threat systems over an extended period of time.

IWEDA, as a decision aid, is representative of systems that organize and display information in a way that is conducive to better decision making. However, IWEDA does not guide the commander in the use of that information.

An example of a directive aiding can be seen in the Agent Enhanced Decision Guide Environment (AEDGE™), a product currently under development which will be used to assist decision making by submarine commanders (Hicks, et al.<sup>3</sup>). AEDGE™ will support real-world monitoring and data gathering, enhanced information organization, and decision option analysis and presentation in the face of degraded or time-constrained information. In other words, based on available information within a tactical situation, the system will indicate “situationally” appropriate options and their outcome risks to the commander for his/her consideration.

## **C. Shortfalls of Previous Approaches**

There are seemingly hundreds of so-called decision aids that have been developed or are currently under development, so why do we need to rethink the design and construction methods now? The reason is that these aids are very specific to their tasks and to their audiences, and thus are limited in their applicability or breadth of capabilities. We might say they are lacking in flexibility to handle more than one narrow job. This compels us to build more of these inflexible decision aids to cover all of our important topics.

What can be done about the lack of flexibility? A simple answer is found in basic elements of an intelligent decision aid, the breadth of the knowledge content, the cognitive skills, and the supporting computations. Greater flexibility requires more of everything.

As for the content, we need cost effective improvements in developing the content and the representations, or models of the content. Creating a decision aid requires extensive resources. Information for design must be gathered from both knowledgeable sources and members of the expected audience(s). This information must then be codified as cognitive or computational elements. The systems are then tested and fine-tuned. Rarely is any part reusable for other decision aids. Rarely are any of these development tasks automated. The key first step here is to compact knowledge representations that support reuse of that knowledge, which means standard knowledge-interchange patterns.

Once the basic content is developed, another obstacle is apparent. That is the cognitive architecture strategies that are used for modeling and utilizing knowledge within the tools. Many complex decision aids incorporate the use of some type of rule-based structure for representing task structures and informational requirements. In our experience, these focus on logic more than versatile representations. The resulting structures can be difficult to build, hard to modify, expand exponentially as paths are added, and do not easily translate into a presentation of information that is readily understood by developer or end-user. Architectures other than rule-based ones have been used to develop complex decision aids. These include neural networks, fuzzy logic, and genetic algorithms, for

example. However, similar basic criticisms, as applied to rule-based approaches, may be directed at these other architectures.

This only leaves computation as an issue. In this case, algorithmic processes dominate and selection of the best algorithm for a specified computation is usually a process of selecting from existing algorithms. Other than the content implications of that problem, space does not permit us to give this area significant consideration in this paper.

Having identified decision aid flexibility as a problem and consequently having associated that with content, cognition, and computation, we look for ways to improve the three areas by recalling the previously listed strengths of Conceptual Graphs. We examine Conceptual Graph based decision aids in the next section.

#### IV. Conceptual Graph Decision Aids

In the previous section, we discussed the general problems with decision aids. We concluded that a major problem was that decision aids lack breadth or flexibility in part due to the difficulty in obtaining and using sufficient content with a sufficient cognitive skill. In this section we look at employing conceptual graphs to overcome these difficulties.

##### A. Prototypical Decision Aid

Before continuing, we identify a prototype for a decision aid as the basis for further discussion in this paper. The components of particular interest are the internal world model, the perceptual components, the thinker and the knowledge. Figure 5 illustrates the main parts of a typical knowledge-based decision aid system.

**Real World** – This is the environment in which the decision-maker and the system are operating. It is important to note that the real world is not directly accessible by the system: we can only model portions of the real world that matter to us based on the limits of the sensors we have to perceive the real world.

**User** – The “human-in-the-loop” who has responsibility for making decisions and is expecting the system to provide assistance, either in the form of “do-this-now” statements, a prioritized set of possible decisions, etc. While this is often seen as an external item, we sometimes develop a user model for a Computer observer to track the behavior of the user much as one friend learns the idiosyncrasies of another friend.

**Computer World Model** – the domain specific knowledge model consisting of ontology, rules and observations about the particular domain of interest (e.g., aircraft navigation). We assume this model will change often to reflect changes in the real world situation of interest. Based on the knowledge modeling strengths of Conceptual Graphs coupled with the dynamic behavior of actors within Conceptual Graphs, we use Conceptual Graphs to construct our world models.

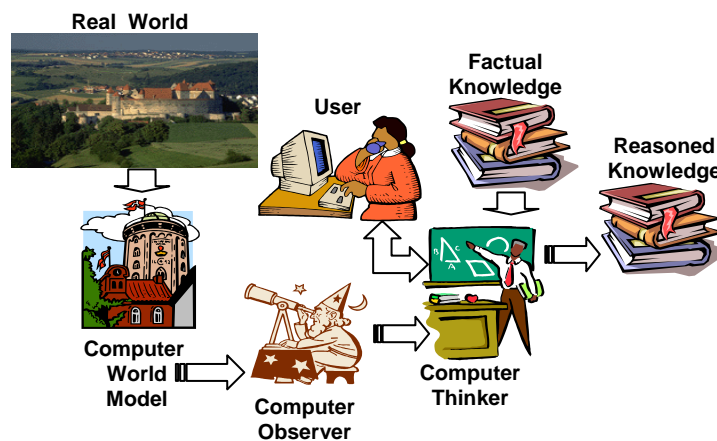


Figure 5

**Computer Observer** – This is an automated system that looks for changes in the computer world model at regular intervals. Since we cannot examine the real world directly, the observer assumes the world model is correct at any point in time, and notes differences at a later time, possibly indicating that a decision is necessary. This is a key feature in our system, in that the system itself can recognize when a decision needs to be made, rather than relying

on the user to know what has changed. This reflexive part was earlier identified with graph matching in Conceptual Graphs.

**Factual Knowledge** – This is the “common sense” ontology, rules, and observations that the system knows (e.g., time, space, physics, users). Together with the computer world model, this forms our model of the universe. Acquiring and maintaining this model is a major undertaking, even for seemingly small systems, because the amount and richness of knowledge needed to make decisions can be quite large. We assume this model will remain relatively stable. Again because of the flexibility and the interchange capabilities, we use Conceptual Graphs here as well. And we can make a similar comment about the next element as well.

**Reasoned Knowledge** – Is the knowledge based on the inferences that the system can make based on its facts and observations of the world or the user. In theory, this component should contain all that the system finds out that is needed for the current decision. This forms the basis for adaptation and learning.

**Computer Thinker** – Contains the inference engine or “brain” that can figure things out based on the factual knowledge. It contains access methods to the factual knowledge and pre-defined rules of inference that govern whether correct inferences are made. For example, if events have a time stamp associated with them, it is possible that a previous event caused a later event, but the reverse would clearly be incorrect. It serves as the reasoning center, where all the relevant facts, inferences, and world model characteristics are processed, analyzed and verified. It also serves as the thought behind interactions with the user. If we believe that humans use different logic “modes” to solve different types of problems, then the multiple reasoning capabilities of Conceptual Graphs make them a choice as the basis for the Computer Thinker.

## B. Knowledge Elements

One of the most difficult parts of any knowledge-based solution is: “Where does its underlying knowledge come from and how does one acquire and model it?” Essentially knowledge can come from three sources; importing from known knowledge sets, from human guided acquisition, and through machine learning. While all of these are possible, the cost of the knowledge increases as we step through this sequence. We have found conceptual graphs to be an effective knowledge modeling technique even when graphs are constructed by hand. Tools for editing and validating graphs greatly speed up the process. Further, once acquired, conceptual graphs offer knowledge reuse and interchange options through currently emerging conceptual graph standards.

Before acquiring knowledge for such a system, we first define an organization to the knowledge that we will loosely call an *ontology*: a collection of terms and definitions, with constraints that must be met among them and the knowledge graphs we will create. This is similar to any modeling effort, in that interested parties must establish a common vocabulary (or else manage multiple vocabularies) and define its concepts. In conceptual graph terms, we establish the concepts and relations that we will be using the process the knowledge, and then construct sets of graphs over time to support our reasoning.

Multiple knowledge-based systems can be integrated into such a system even if they happen to use differing ontologies to start with. As long as the ontologies have well-defined concept types and constraints, it is possible (with manual assistance) to establish correspondences between them, and thereby reason across multiple sets of knowledge. For a truly affordable, flexible and powerful system, multiple knowledge sources will likely be necessary, and automated ontology mapping will be of great assistance.

In order to support practical decision-making, decision aid systems require more input than a mere description of some situation. They require also knowledge about the domain in which the decision will be made. Builders of knowledge-based systems are recognizing that they need knowledge about underlying assumptions and general principles in order to support decision-making that will rival a human’s in accuracy. It is important to stress that this underlying knowledge is usually much greater than the situation descriptions themselves.

One important component of domain knowledge is the presence of an *ontology* – a set of terms and relationships that hold within the domain. New ontological tools are being developed, among them the Web Ontology Language (OWL<sup>4</sup>), a language for developing ontologies for the Semantic Web. Most of these ontologies rely on a list of terms with definitions, while OWL also provides subtyping and supertyping capabilities, along with some relationships between terms. Conceptual graphs go beyond these ontology mechanisms by providing reasoning capabilities among the terms, as well as constraints and rules that hold among the terms. In essence, conceptual graphs model the concepts of the ontology, whereas OWL and its derivatives only model terms.

Another advantage of conceptual graphs is that they are an integral part of an ongoing effort to standardize logical formalisms. This effort, called the Common Logic Project, is an approved International Standards Organization (ISO) project<sup>5</sup> at the Working Draft stage and will result in a standard syntax and semantics for conceptual graphs, as well as some other logic formalisms. The syntax standard is called Conceptual Graph Interchange Format (CGIF) and will map directly to the semantics of the Common Logic standard. This will allow

conceptual graph tools to exchange their knowledge and ontologies between systems, resulting in increased re-use and interoperability.

### C. Design Example

To illustrate how conceptual graphs represent knowledge that can be used for decision-making, we will show a simple example. In this example, an aircraft (in this case, a helicopter) is in a particular location and we want to get to another location. The “decision” is therefore how to get from the starting point to the destination.

The following graph shows a typical situation. Our own aircraft is located with a particular set of coordinates, altitude 45 feet, heading 180 at 70 knots (we’re in a hurry). There are other features: a hangar and a water tower, (Figure 6). In a production-level system, the places and coordinates would of course be specified.

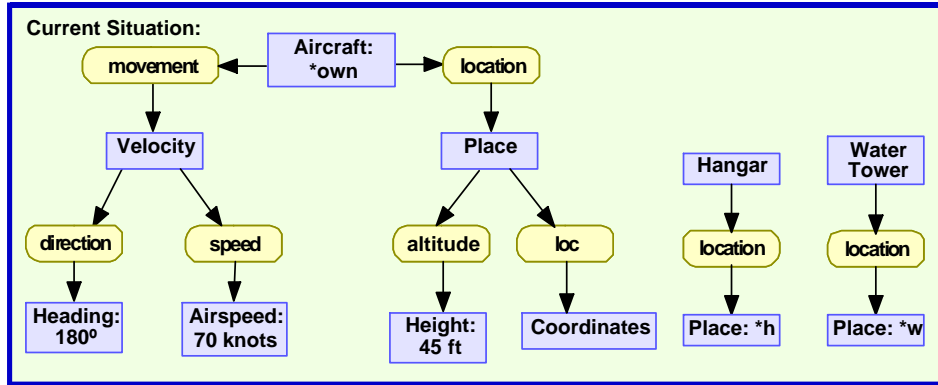


Figure 6

Our goal is to be at a particular destination, the hangar, traveling at a different heading (Figure 7).

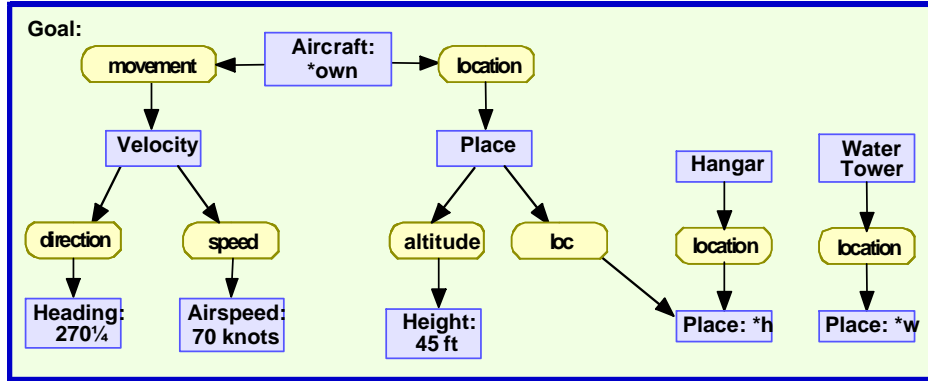


Figure 7

In order to reach the goal situation, we first compare the graphs and see where they differ. In this case there are two differences: the heading and the location of the aircraft. To transform the current situation into the goal situation, we have to search our knowledge base for information about places and in particular, places that change. One kind of graph in our knowledge base might be the following (Figure 8).

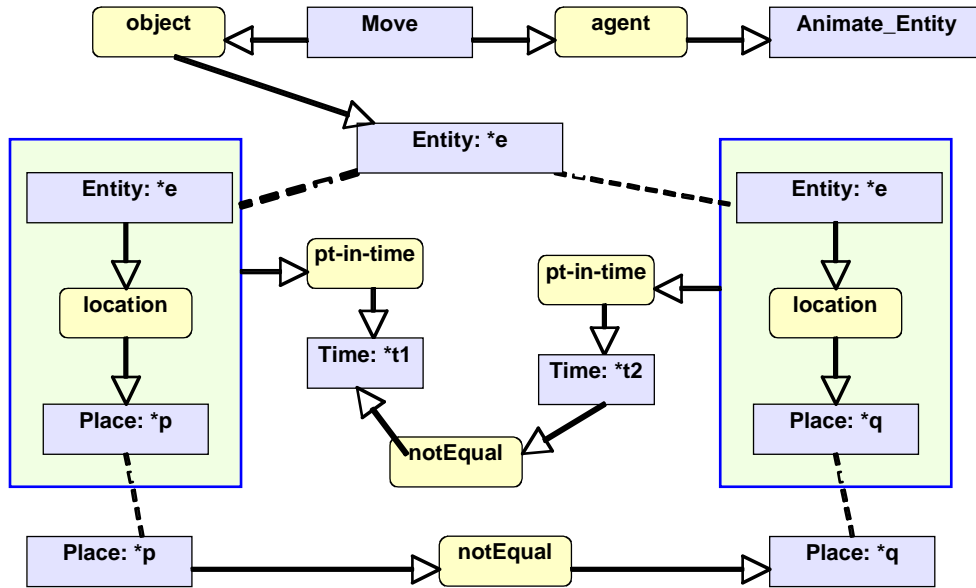


Figure 8

This graph expresses the “common-sense” notion that if there is an entity that is in two different places at two different points in time, then there exists an animate entity that is the agent of a “move” action on the entity, i.e. something must actively move the entity. The decision support system therefore can infer that some animate entity must perform a move on the aircraft to get it from one place to another.

There are some important points to understand with respect to this kind of example. In order to simplify the graphs for this paper, some assumptions have been left out of the graphs. Of course, if an automated system doesn’t know these assumptions, the inference described will not be possible. A more detailed conceptual graph would explicitly contain these assumptions. Some of the assumptions made are:

- The hangar place and the aircraft’s original coordinates are different.
- The aircraft is an entity.
- It can be known by the system that two places are either the same or different.
- The system can keep track of time and situations that hold at various points in time.

The strengths of conceptual graphs lie in the fact that the relationship between the two locations does not constitute a “rule” – it merely expresses a relationship between moving, places and time. There are additional inferences that could be made – for example, according to the previous graph, the entity being moved and the animate entity doing the moving could be the same entity; i.e., an entity can move itself. This is quite permissible given the graph above.

## V. Conclusion

In this paper we have presented the idea that Conceptual Graphs provide a more viable approach to developing intelligent systems due to the representational flexibility, the reasoning support and the current efforts in standardization. At present several intelligent software developments are in process by the authors. Based on the understandings identified in this paper, we finish by commenting on the architectural approach in the Conceptual Graph systems we are using and developing.

**Maintainability** – We have found that an underlying simple graph system is essential to later tailoring for specific applications. We have also found that whenever complexity of usage or reasoning is needed, domain knowledge should be modeled as a conceptual graph to permit reasoning on that usage.

**Large scale** – We might suspect based on the human brain model and our experience that deep intelligence will require large numbers of graph nodes, whether in one large graph or in many smaller graphs. Therefore scalability of the graphs and the ability to partition over multiple computers has become an important consideration. This in turn leads to synchronization concerns over a large knowledge base spread over a network.

**Reasoning** – We find multiple reasoning methods are available in graphs from basic perceptions to graph unification. What we are still studying is what mode or combination of modes leads to faster better answers in large-scale systems.

**Reuse** – At present knowledge acquisition and modeling is so expensive that building intelligent (large-scale) systems is cost prohibitive without cost sharing. That cost sharing can be accomplished by interchange standards. We anticipate that the current ISO efforts will yield usable results soon that can be brought to bear on this need.

**Actors** – While rule based systems grew out of logic modeling, conceptual graphs grew out of knowledge representation, and is often seen as a static unchanging model of a set of knowledge. We have found that the actor (and the Demon (Delugach<sup>6</sup>)) bring dynamic behavior into the graph and thus for real systems represent a required element. Consequently all of the reasoning techniques are implemented as actors or demons so that the appropriate reasoning type can be embedded within the appropriate graph, including self-modifying graphs.

## References

<sup>1</sup>Sowa, John F. *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, 1984.

<sup>2</sup>Sauter, D., “An Interactive Information and Processing System to Assist the Military with Command and Control Decision Making,” *Proceedings of the 16<sup>th</sup> Conference on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology*. American Meteorological Society, Boston, MA, 2000, pp.279-282.

<sup>3</sup>Hicks, J. D, Stoyen, A. D., and Zhu, Q., “Intelligent Agent-Based Software Architecture for Combat Performance under Overwhelming Information Inflow and Uncertainty,” *Proceedings of the Seventh IEEE International Conference on Complex Computer Systems*, IEEE Computer Society Conference Publishing Services, Los Almitos, CA, 2001, pp.200-211.

<sup>4</sup>Web Ontology Language, W3C Consortium Standard, <http://www.w3.org/2004/OWL/>

<sup>5</sup>Common Logic Project, WD 24707, available from ISO JTC1/SC32 Secretariat, <http://www.jtc1sc32.org/>.

<sup>6</sup>Delugach, H.S., Specifying Multiple-Viewed Software Requirements With Conceptual Graphs. *Jour. Systems and Software*. Vol. 19, 1992, pp. 207-224.