# A Knowledge-Sharing Strategy

**David Goldstein** and **Albert Esterline**

North Carolina A&T State University
Greensboro, North Carolina
goldstn @ncat.edu, esterlin@ncat.edu

## Introduction

Sophisticated cooperative reasoning among agents requires that the agents be able to share models of their reasoning processes. Developing intelligent systems cost-effectively necessitates that components be reused. In order to facilitate sharing and reusing knowledge among distributed (knowledge-based) applications, we have been developing a canonical representation for acquiring and transmitting semantics. This representation will include both a scheme for representing semantics and a hierarchy of concepts used to describe predicates.

We have been developing and formalizing a hierarchy describing knowledge to facilitate (1) the specification of knowledge and assumptions employed by a system, (2) transferring knowledge among agents (applications) in a system and (3) for resolving conflicts among these agents (Figure 1). We are utilizing derivatives of object diagrams [Rumbaugh91], semantic networks [CQ69], and conceptual dependencies [SR74] to describe the fundamental concepts which underlay various algorithms and knowledge representations. By formally describing higher level concepts via these fundamental concepts, we intend to reason about the semantics of and translate knowledge among applications employing different knowledge representations. This hierarchy of concepts will be validated by performing analysis on a complex domain (probably plastics manufacturing) to ensure that the formalisms used to describe elements in the hierarchy are sufficient.

## Background

Attempts at sharing knowledge among applications have largely been based upon various knowledge acquisition methods. Such methods include KADS [WSB92], KL-ONE [BS, 85] and Conceptual Graphs [Sowa84]. The implicit understanding is that experimenters using such methods would use tools supporting the methods. Examples of such tools are the Shelley integrated workbench [AWR92] and the PROTEGE system for constructing knowledge editors [Musen89]. Our approach here is closest to Conceptual Graphs, in that the motivation is from a linguistic basis.

We are using derivatives of object diagrams, conceptual dependencies, and semantic networks. We are employing conceptual dependencies because of their success in natural language processing; natural language processing intuitively seems a more difficult problem then describing the exchange of knowledge, and so should suffice. We are employing a derivative of object diagrams for (1) inheritance and (2) abstraction. Object diagrams facilitate organizing the concepts contained in a specific application.

## Approach

The initial thrust of our project has been to analyze a domain (manufacturing) simply to provide a mechanism for refining the problem. We are also analyzing the standards and taxonomies/data dictionaries developed in relevant efforts. The analysis is to provide a basis for creating a taxonomy of knowledge. This hierarchy will support the object-oriented models (in a partial reference architecture) by defining a lattice of semantic concepts. The most "atomic" elements of the lattice will be the primitives (probably Product Data Exchange Specification measurement units, basic transformation operators, and conceptual-dependency concepts) behind any system reflecting this work. Lower level concepts will be used to define higher level concepts; we will use equations to define mathematically or symbolically an object and its attributes, such as acceleration, in terms of other objects, such as velocity.

We have modified object diagrams to facilitate pointers to (1) our semantic network/conceptual dependency information and (2) functions to perform the actual translation of concepts. For example, entities in Figure 2 would include pointers to numerous functions, the simplest being those performing translation, rotation and scaling, to accomplish this conversion. More complex functions are required when translation must be performed across dimensions, for example, three-dimensional CSG representations to two-dimensional planes.

The modified object diagrams are being used to create higher level concepts in the hierarchy, such as materials; physical items; configurations; parts; rules about parts; and mechanisms for resolving conflicting part requirements, manufacturing operations and deductions (Figure 3). We are attempting to build dual versions of each model as we proceed. The first version is hierarchy defined to minimize the semantic assumptions made by the lowest level elements in the hierarchy (which in turn define higher level elements); by minimizing the number of primitives this should facilitate provable correctness and an exhaustive reasoning capability. The other version of the models contains functional programmed "hacks" to expedite real-time processing.

We have also been exploring derivatives of conceptual dependencies to represent the concepts of the hierarchy from a linguistic standpoint. This, in turn, provides a basis for defining the intension of concepts (as opposed to the extension of concepts taken by many approaches using only predicate calculus derivatives in defining crucial concepts, such as the predicates used in knowledge.) Hence, the object models contain pointers to derivatives of conceptual dependencies (Figure 4) providing possible contexts for the objects in various applications. We are currently addressing shortcomings in semantic dependencies to accommodate inheritance and a better representation of constraints.

A promising direction we are currently exploring uses "feature terms" as formal explications of conceptual dependencies. Feature description languages [Smolka92] subsume unification grammars used in computational linguistics and terminological logic (such as KL-ONE) used in knowledge representation. Feature terms include the $\psi$-terms of the logic programming language LIFE [Ait-Kaci94, AP93]. [Smolka92] shows that feature description languages can be viewed as a type of language facilitating of first-order predicate logic with equality and develops a Tarski-style semantics for them. Ait-Kaci [APG93] develops a formal system of order-sorted feature (OSF) terms that logically models flexible record objects with recursive class definitions that accommodate multiple inheritance. Sort definitions are viewed as axioms forming an OSF theory. OSF theory unification, then, is a process of normalizing OSF terms that enforces (possibly inherited) structural constraints imposed on sorts by their definitions.

## Status

We have implemented the modified object diagrams and conceptual dependencies in a "toy" system to crystallize ideas. The system considers transferring information between a CAD system and design critiques in the assembly (manufacturing) domain. The system has been implemented in Prolog and is being ported to LIFE (to support inheritance). Our methods are designed to be extensible, such that an engineer in almost any domain could extend the hierarchy for his application (given enough time).

We are still building the actual hierarchy of concepts. The hierarchy will certainly incorporate concepts from manufacturing standards such as PDES and various ISO standards. The formalizing mechanisms and hierarchy will be refined, verified and validated through further domain analysis and experimentation during later steps of this endeavor. Our first genuine application will probably result from our work with a major, U.S. manufacturer.

We will use the hierarchy to develop efficient algorithms to facilitate transferring information among agents. Such algorithms would enable agents to transmit their underlying semantics (such as assumptions) and the characteristics of their procedural knowledge employed to other agents (i.e., to have agents "teach" other agents). Similarly, agents could share semantic knowledge to develop models of each other's thought processes. We will analyze the hierarchy and knowledge protocol to determine their (1) algebraic and computational properties, (2) descriptive limitations and (3) underlying requirements (such as synchronization required.)

The algorithms and hierarchies will be realized as software as resources permit. The "logically sound" models will be implemented in LIFE [Ait-Kaci94]. LIFE incorporates aspects of logic, functional, and object-oriented programming. Our "hacked" version will be developed via "wrappers" to a cooperative reasoning framework (such as the Distributed Artificial Intelligence Toolkit [Goldstein94]). We would also like to consider the requirements of a (graphical or language-based) knowledge-sharing mechanism. Finally, we intend to realizing our work by at least beginning the development of an application (probably a concurrent engineering testbed).

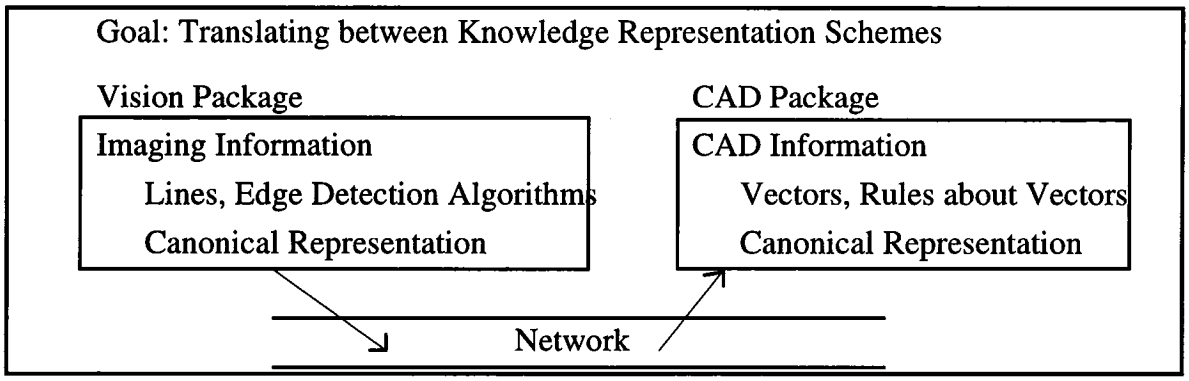


Figure 1: Transferring Knowledge among Applications
A canonical representation can be used to translate to and from an application's specific knowledge representation. We intend to develop a hierarchical canonical representation with computable elements at the lowest level of the hierarchy to preserve the semantics of knowledge translated to and from the canonical representation.

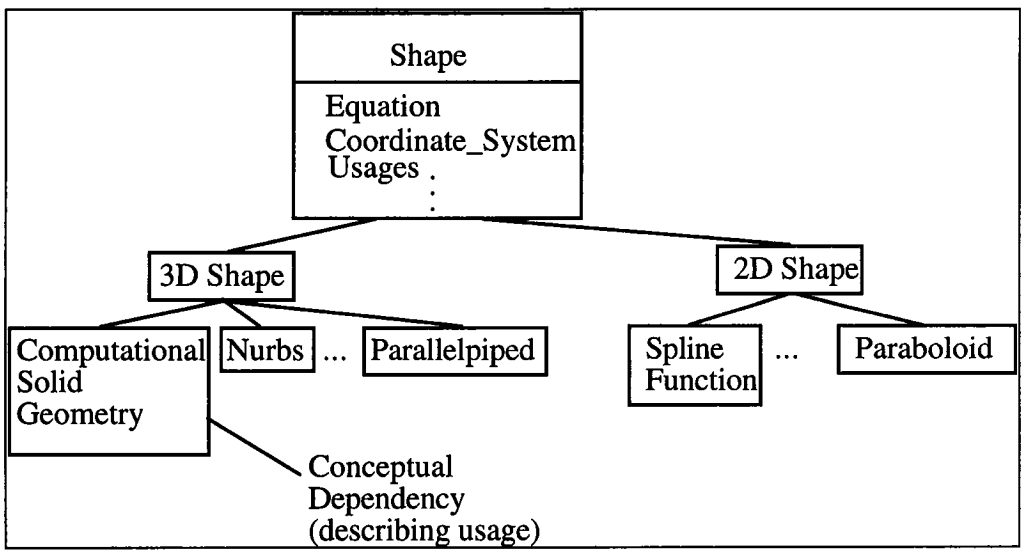


Figure 2: Modified Object Diagrams

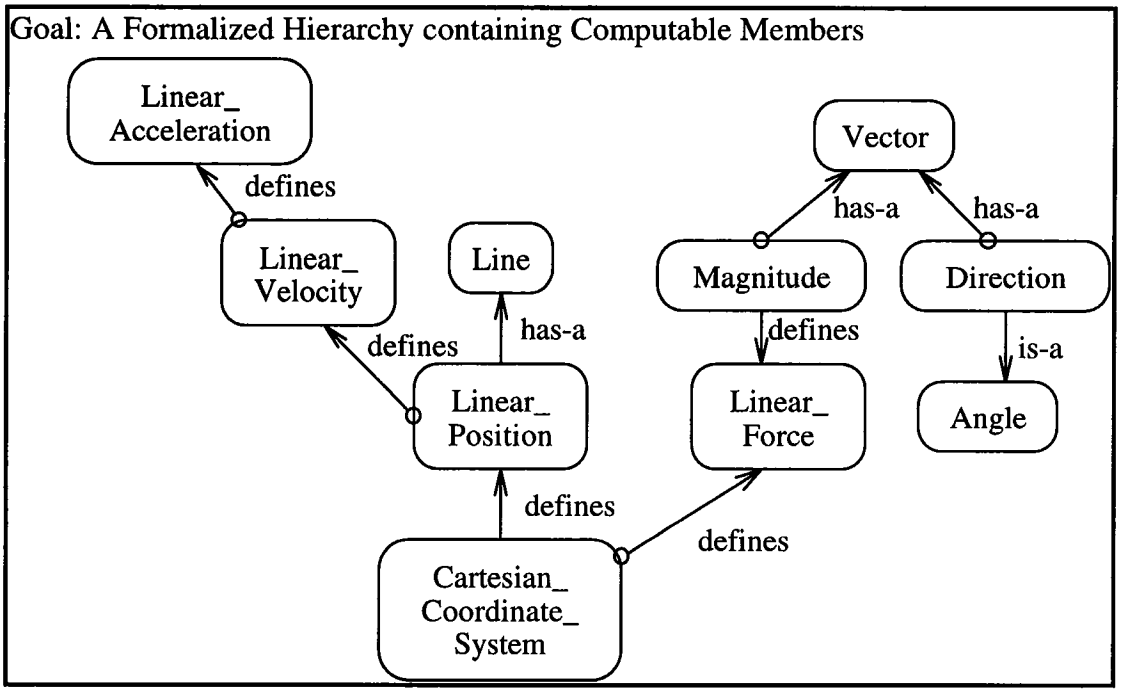


Figure 3: Partial Lattice of Informational Concepts
A trivial lattice of informational concepts. The actual lattice developed will incorporate attributes, methods and definitions (e.g., equations) describing concepts. The lattice will also be split into separate, distinct hierarchies to reflect various dependencies.

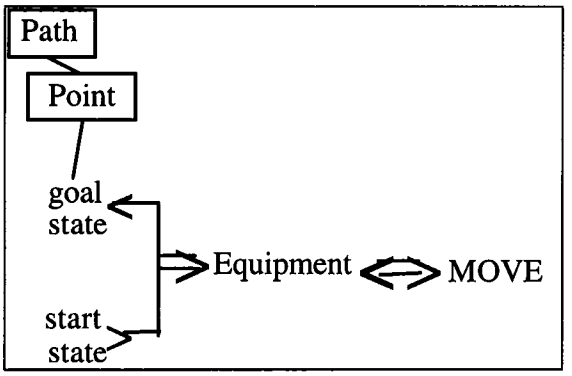


Figure 4: Conceptual Dependency reflecting Usage of an Object

**References**


[Ait-Kaci94] Ait-Kaci, H. *et al.* (March, 1994) . The Wild LIFE Handbook, DEC Paris Research Laboratory.

[AP93] Ait-Kaci, H. & Podelski, A. (1993) . "Towards a Meaning of Life," J. of Logic Programming, vol. 16, pp. 195--234.

[APG93] Ait-Kaci, H. & Podelski, A. & Goldstein, S.C. (May, 1993) . "Order-Sorted Feature Theory Unification," DEC Paris Research Laboratory, Research report 32.

[AWT92] Anjewierden, A. & Wielemaker, J. & Toussaint, C. (1992) . "Shelley - Computer-Aided Knowledge Engineering," Knowledge Acquisition, vol. 4, pp. 109--125.

[BS85]Brachman, R.J. & Schmolze, J.G.(1985) . "An Overview of the KL-ONE Knowledge Representation System," Cognitive Science, vol. 9 , pp. 171--126.

[CQ69] Collins, A. & Quinllian, M. (1969) . Retrieval time from semantic memory, Journal of Verbal Learning & Verbal Behavior, 8:240--247.

[Rumbaugh91] J. Rumbaugh *et al.* (1991) . Object-Oriented Modeling and Design. Prentice-Hall, Englewood Cliffs, NJ.

[Goldstein 94] Goldstein, D. (January, 1994) . "The Distributed Artificial Intelligence Toolkit", AI Expert, Miller-Freeman Publishing, pp. 30--34.

[Musen89] M. Musen, M. (1989) . Automated Generation of Model-Based Knowledge-Acquisition Tools, Morgan Kaufmann, San Mateo, CA.

[Smolka92] Smolka, G. (1992) . "Feature Constraint Logics for Unification Grammars," J. of Logic Programming, vol. 12, pp. 51--87.

[Sowa84] Sowa, J. (1984) . Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley, Reading, MA.

[SR74] Schank, R. & Reiger, C. (1974) . "Inference and the computer understanding of natural language", Artificial Intelligence 5(4):373--412.

[WSB92] Wielinga, B.J. & Schreiber, A. Th. & Breuker, J.A. (1992) . "KADS: A Modelling Approach to Knowledge Engineering," Knowledge Acquisition, vol. 4, pp. 5--53.