Semantics of Conceptual Graphs

John F. Sowa
IBM Systems Research Institute
205 East 42nd Street
New York, NY 10017

ABSTRACT: Conceptual graphs are both a language for representing knowledge and patterns for constructing models. They form models in the AI sense of structures that approximate some actual or possible system in the real world. They also form models in the logical sense of structures for which some set of axioms are true. When combined with recent developments in nonstandard logic and semantics, conceptual graphs can form a bridge between heuristic techniques of AI and formal techniques of model theory.

1. Surface Models

Semantic networks are often used in AI for representing meaning. But as Woods (1975) and McDermott (1976) observed, the semantic networks themselves have no well-defined semantics. Standard predicate calculus does have a precisely defined, model theoretic semantics; it is adequate for describing mathematical theories with a closed set of axioms. But the real world is messy, incompletely explored, and full of unexpected surprises. Furthermore, the infinite sets commonly used in logic are intractable both for computers and for the human brain.

To develop a more realistic semantics, Hintikka (1973) proposed surface models as incomplete, but extendible, finite constructions:

> Usually, models are thought of as being given through a specification of a number of properties and relations defined on the domain. If the domain is infinite, this specification (as well as many operations with such entities) may require non-trivial set-theoretical assumptions. The process is thus often non-finitistic. It is doubtful whether we can realistically expect such structures to be somehow actually involved in our understanding of a sentence or in our contemplation of its meaning, notwithstanding the fact that this meaning is too often thought of as being determined by the class of possible worlds in which the sentence in question is true. It seems to me much likelier that what is involved in one's actual understanding of a sentence S is a mental anticipation of what can happen in one's step-by-step investigation of a world in which S is true. (p. 129)

The first stage of constructing a surface model begins with the entities occurring in a sentence or story. During the construction, new facts may be asserted that block certain extensions or facilitate others. A standard model is the limit of a surface model that has been extended infinitely deep, but such infinite processes are not a normal part of understanding.

This paper adapts Hintikka's surface models to the formalism of conceptual graphs (Sowa 1976, 1978). Conceptual graphs serve two purposes: like other forms of semantic networks, they can be used as a canonical representation of meaning in natural language; but they can also be used as building blocks for constructing abstract structures that serve as models in the model-theoretic sense.

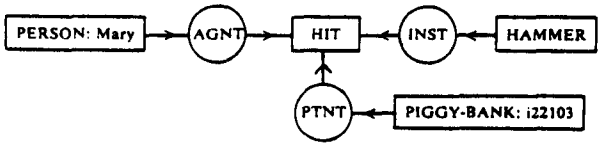
- Understanding a sentence begins with a translation of that sentence into a conceptual graph.
- During the translation, that graph may be joined to frame-like (Minsky 1975) or script-like (Schank & Abelson 1977) graphs that help resolve ambiguities and incorporate background information.
- The resulting graph is a nucleus for constructing models of possible worlds in which the sentence is true.
- Laws of the world behave like demons or triggers that monitor the models and block illegal extensions.
- If a surface model could be extended infinitely deep, the result would be a complete standard model.

This approach leads to an infinite sequence of algorithms ranging from plausible inference to exact deduction; they are analogous to the varying levels of search in game playing programs. Level 0 would simply translate a sentence into a conceptual graph, but do no inference. Level 1 would do frame-like plausible inferences in joining other background graphs. Level 2 would check constraints by testing the model against the laws. Level 3 would join more background graphs. Level 4 would check further constraints, and so on. If the constraints at level $n+1$ are violated, the system would have to backtrack and undo joins at level n. If at some level, all possible extensions are blocked by violations of the laws, then that means the original sentence (or story) was inconsistent with the laws. If the surface model is infinitely extendible, then the original sentence or story was consistent.

Exact inference techniques may let the surface models grow indefinitely; but for many applications, they are as impractical as letting a chess playing program search the entire game tree. Plausible inferences with varying degrees of confidence are possible by stopping the surface models at different levels of extension. For story understanding, the initial surface model would be derived completely from the input story. For consistency checks in updating a data base, the initial model would be derived by joining new information to the pre-existing data base. For question-answering, a query graph would be joined to the data base; the depth of search permitted in extending the join would determine the limits of complexity of the questions that are answerable. As a result of this theory, algorithms for plausible and exact inference can be compared within the same framework; it is then possible to make informed trade-offs of speed vs. consistency in data base updates or speed vs. completeness in question answering.

2. Conceptual Graphs

The following conceptual graph shows the concepts and relationships in the sentence "Mary hit the piggy bank with a hammer." The boxes are concepts and the circles are conceptual relations. Inside each box or circle is a type label that designates the type of concept or relation. The conceptual relations labeled AGNT, INST, and PTNT represent the linguistic cases agent, instrument, and patient of case grammar.



Conceptual graphs are a kind of semantic network. See Findler (1979) for surveys of a variety of such networks that have been used in AI. The diagram above illustrates some features of the conceptual graph notation:

- Some concepts are *generic*. They have only a *type label* inside the box, e.g. HIT or HAMMER.
- Other concepts are *individual*. They have a colon after the type label, followed by a *name* (Mary) or a unique identifier called an *individual marker* (i22103).

To keep the diagram from looking overly busy, the hierarchy of types and subtypes is not drawn explicitly, but is determined by a separate partial ordering of type labels. The type labels are used by the formation rules to enforce selection constraints and to support the inheritance of properties from a supertype to a subtype.

For convenience, the diagram could be linearized by using square brackets for concepts and parentheses for conceptual relations:

[PERSON:Mary]→(AGNT)→[HIT:c1]←(INST)←[HAMMER]
[HIT:c1]←(PTNT)←[PIGGY-BANK:i22103]

Linearizing the diagram requires a *coreference index*, c1, on the generic concept HIT. The index shows that the two occurrences designate the same act of hitting. If HIT had been an individual concept, its name or individual marker would be sufficient to indicate the same act.

Besides the features illustrated in the diagram, the theory of conceptual graphs includes the following:

- For any particular domain of discourse, a specially designated set of conceptual graphs called the *canon*,
- Four *canonical formation rules* for deriving new *canonical graphs* from any given canon,
- A method for defining new concept types: some canonical graph is specified as the *differentia* and a concept in that graph is designated the *genus* of the new type,
- A method for defining new types of conceptual relations: some canonical graph is specified as the *relator* and one or more concepts in that graph are specified as *parameters*,
- A method for defining composite entities as structures having other entities as *parts*,
- Optional quantifiers on generic concepts,
- Scope of quantifiers specified either by embedding them inside type definitions or by linking them with functional dependency arcs,
- Procedural attachments associated with the functional dependency arcs,
- Control marks that determine when attached procedures should be invoked.

These features have been described in the earlier papers; for completeness, the appendix recapitulates the axioms and definitions that are explicitly used in this paper.

Heidorn's (1972, 1975) Natural Language Processor (NLP) is being used to implement the theory of conceptual graphs. The NLP system processes two kinds of Augmented Phrase Structure rules: *decoding rules* parse language inputs and create graphs that represent their meaning, and *encoding rules* scan the graphs to generate language output. Since the NLP structures are very similar to conceptual graphs, much of the implementation amounts to identifying some feature or combination of features in NLP for each construct in conceptual graphs. Constructs that would be difficult or inefficient to implement directly in NLP rules can be supported by LISP functions. The inference algorithms in this paper, however, have not yet been implemented.

3. Logical Connectives

Canonical formation rules enforce the selection constraints in linguistics: they do not guarantee that all derived graphs are true, but they rule out semantic anomalies. In terms of graph grammars, the canonical formation rules are context-free. This section defines logical operations that are context-sensitive. They enforce tighter constraints on graph derivations, but they require more complex pattern matching. Formation rules and logical operations are complementary mechanisms for building models of possible worlds and checking their consistency.

Sowa (1976) discussed two ways of handling logical operators in conceptual graphs: the *abstract approach*, which treats them as functions of truth values, and the *direct approach*, which treats implications, conjunctions, disjunctions, and negations as operations for building, splitting, and discarding conceptual graphs. That paper, however, merely mentioned the approach; this paper develops a notation adapted from Gentzen's sequents (1934), but with an interpretation based on Belnap's conditional assertions (1973) and with computational techniques similar to Hendrix's partitioned semantic networks (1975, 1979). Deliyanni and Kowalski (1979) used a similar notation for logic in semantic networks, but with the arrows reversed.

Definition: A *sequent* is a collection of conceptual graphs divided into two sets, called the *conditions* $u_1,...,u_n$ and the *assertions* $v_1,...,v_m$. It is written $u_1,...,u_n \rightarrow v_1,...,v_m$. Several special cases are distinguished:

- A *simple assertion* has no conditions and only one assertion: $\rightarrow v$.
- A *disjunction* has no conditions and two or more assertions: $\rightarrow v_1,...,v_m$.
- A *simple denial* has only one condition and no assertions: $u \rightarrow$.
- A *compound denial* has two or more conditions and no assertions: $u_1,...,u_n \rightarrow$.
- A *conditional assertion* has one or more conditions and one or more assertions: $u_1,...,u_n \rightarrow v_1,...,v_m$.
- An *empty clause* has no conditions or assertions: $\rightarrow$.
- A *Horn clause* has at most one assertion; i.e. it is either an empty clause, a denial, a simple assertion, or a conditional assertion of the form $u_1,...,u_n \rightarrow v$.

For any concept a in an assertion v_i, there may be a concept b in a condition u_j that is declared to be *coreferent* with a.

Informally, a sequent states that if all of the conditions are true, then at least one of the assertions must be true. A sequent with no conditions is an unconditional assertion; if there

are two or more assertions, it states that one must be true, but it doesn't say which. Multiple assertions are necessary for generality, but in deductions, they may cause a model to split into models of multiple alternative worlds. A sequent with no assertions denies that the combination of conditions can ever occur. The empty clause is an unconditional denial; it is self-contradictory. Horn clauses are special cases for which deductions are simplified: they have no disjunctions that cause models of the world to split into multiple alternatives.

Definition: Let C be a collection of canonical graphs, and let s be the sequent $u_1,...,u_n \rightarrow v_1,...,v_m$.

- If every condition graph is covered by some graph in C, then the conditions are said to be *satisfied.*
- If some condition graph is not covered by any graph in C, then the sequent s is said to be *inapplicable* to C.

If $n=0$ (there are no conditions), then the conditions are trivially satisfied.

A sequent is like a conditional assertion in Belnap's sense: When its conditions are not satisfied, it asserts nothing. But when they are satisfied, the assertions must be added to the current context. The next axiom states how they are added.

Axiom: Let C be a collection of canonical graphs, and let s be the sequent $u_1,...,u_n \rightarrow v_1,...,v_m$. If the conditions of s are satisfied by C, then s may be *applied* to C as follows:

- If $m=0$ (a denial or the empty clause), the collection C is said to be *blocked.*
- If $m=1$ (a Horn clause), a copy of each graph u_i is joined to some graph in C by a covering join. Then the assertion v is added to the resulting collection C'.
- If $m \geq 2$, a copy of each graph u_i is joined to some graph in C by a covering join. Then all graphs in the resulting collection C' are copied to make m disjoint collections identical to C'. Finally, for each j from 1 to m, the assertion v_j is added to the j-th copy of C'.

After an assertion v is added to one of the collections C', each concept in v that was declared to be coreferent with some concept b in one of the conditions u_i is joined to that concept to which b was joined.

When a collection of graphs is inconsistent with a sequent, they are blocked by it. If the sequent represents a fundamental law about the world, then the collection represents an impossible situation. When there is only one assertion in an applicable sequent, the collection is extended. But when there are two or more assertions, the collection splits into as many successors as there are assertions; this splitting is typical of algorithms for dealing with disjunctions. The rules for applying sequents are based on Beth's semantic tableaux (1955), but the computational techniques are similar to typical AI methods of production rules, demons, triggers, and monitors.

Deliyanni and Kowalski (1979) relate their algorithms for logic in semantic networks to the resolution principle. This relationship is natural because a sequent whose conditions and assertions are all atoms is equivalent to the standard clause form for resolution. But since the sequents defined in this paper may be arbitrary conceptual graphs, they can package a much larger amount of information in each graph than the low level atoms of ordinary resolution. As a result, many fewer steps may be needed to answer a question or do plausible inferences.

4. Laws, Facts, and Possible Worlds

Infinite families of possible worlds are computationally intractable, but Dunn (1973) showed that they are not needed for the semantics of modal logic. He considered each possible world w to be characterized by two sets of propositions: laws L and facts F. Every law is also a fact, but some facts are merely contingently true and are not considered laws. A proposition p is necessarily true in w if it follows from the laws of w, and it is possible in w if it is consistent with the laws of w. Dunn proved that semantics in terms of laws and facts is equivalent to the possible worlds semantics.

Dunn's approach to modal logic can be combined with Hintikka's surface models and AI methods for handling defaults. Instead of dealing with an infinite set of possible worlds, the system can construct finite, but extendible surface models. The basis for the surface models is a canon that contains the blueprints for assembling models and a set of laws that must be true for each model. The laws impose obligatory constraints on the models, and the canon contains common background information that serves as a heuristic for extending the models.

An initial surface model would start as a canonical graph or collection of graphs that represent a given set of facts in a sentence or story. Consider the story,

> Mary hit the piggy bank with a hammer. She wanted to go to the movies with Janet, but she wouldn't get her allowance until Thursday. And today was only Tuesday.

The first sentence would be translated to a conceptual graph like the one in Section 2. Each of the following sentences would be translated into other conceptual graphs and joined to the original graph. But the story as stated is not understandable without a lot of background information: piggy banks normally contain money; piggy banks are usually made of pottery that is easily broken; going to the movies requires money; an allowance is money; and Tuesday precedes Thursday.

Charniak (1972) handled such stories with demons that encapsulate knowledge: demons normally lie dormant, but when their associated patterns occur in a story, they wake up and apply their piece of knowledge to the process of understanding. Similar techniques are embodied in production systems, languages like PLANNER (Hewitt 1972), and knowledge representation systems like KRL (Bobrow & Winograd 1977). But the trouble with demons is that they are unconstrained: anything can happen when a demon wakes up, no theorems are possible about what a collection of demons can or cannot do, and there is no way of relating plausible reasoning with demons to any of the techniques of standard or nonstandard logic.

With conceptual graphs, the computational overhead is about the same as with related AI techniques, but the advantage is that the methods can be analyzed by the vast body of techniques that have been developed in logic. The graph for "Mary hit the piggy-bank with a hammer" is a nucleus around which an infinite number of possible worlds can be built. Two individuals, Mary and PIGGY-BANK:i22103, are fixed, but the particular act of hitting, the hammer Mary used, and all other circumstances are undetermined. As the story continues, some other individuals may be named, graphs from the canon may be joined to add default information, and laws of the world in

the form of sequents may be triggered (like demons) to enforce constraints. The next definition introduces the notion of a *world basis* that provides the building material (a canon) and the laws (sequents) for such a family of possible worlds.

Definition: A *world basis* has three components: a canon C, a finite set of sequents L called *laws*, and one or more finite collections of canonical graphs $\{C_1,...,C_n\}$ called *contexts*. No context C_i may be blocked by any law in L.

A world basis is a collection of nuclei from which complete possible worlds may evolve. The contexts are like Hintikka's surface models: they are finite, but extendible. The graphs in the canon provide default or plausible information that can be joined to extend the contexts, and the laws are constraints on the kinds of extensions that are possible.

When a law is violated, it blocks a context as a candidate for a possible world. A default, however, is optional; if contradicted, a default must be undone, and the context restored to the state before the default was applied. In the sample story, the next sentence might continue: "The piggy bank was made of bronze, and when Mary hit it, a genie appeared and gave her two tickets to *Animal House*." This continuation violates all the default assumptions; it would be unreasonable to assume it in advance, but once given, it forces the system to back up to a context before the defaults were applied and join the new information to it. Several practical issues arise: how much backtracking is necessary, how is the world basis used to develop possible worlds, and what criteria are used to decide when to stop the (possibly infinite) extensions. The next section suggests an answer.

5. Game Theoretic Semantics

The distinction between optional defaults and obligatory laws is reminiscent of the AND-OR trees that often arise in AI, especially in game playing programs. In fact, Hintikka (1973, 1974) proposed a game theoretic semantics for testing the truth of a formula in terms of a model and for elaborating a surface model in which that formula is true. Hintikka's approach can be adapted to elaborating a world basis in much the same way that a chess playing program explores the game tree:

- Each context represents a position in the game.
- The canon defines possible moves by the current player.
- Conditional assertions are moves by the opponent.
- Denials are checkmating moves by the opponent.
- A given context is consistent with the laws if there exists a strategy for avoiding checkmate.

By following this suggestion, one can adapt the techniques developed for game playing programs to other kinds of reasoning in AI.

Definition: A *game* over a world basis W is defined by the following rules:

- There are two participants named *Player* and *Opponent*.
- For each context in W, Player has the first move.
- Player moves in context C either by joining two graphs in C or by selecting any graph in the canon of W that is joinable to some graph u in C and joining it maximally to u. If no joins are possible, Player passes. Then Opponent has the right to move in context C.
- Opponent moves by checking whether any denials in W are satisfied by C. If so, context C is blocked and is deleted from W. If no denials are satisfied, Opponent may apply any other sequent that is satisfied in C. If no sequent is satisfied, Opponent passes. Then Player has the right to move in context C.
- If no contexts are left in W, Player loses.
- If both Player and Opponent pass in succession, Player wins.

Player wins this game by building a complete model that is consistent with the laws and with the initial information in the problem. But like playing a perfect game of chess, the cost of elaborating a complete model is prohibitive. Yet a computer can play chess as well as most people do by using heuristics to choose moves and terminating the search after a few levels. To develop systematic heuristics for choosing which graphs to join, Sowa (1976) stated rules similar to Wilks' preference semantics (1975).

The amount of computation required to play this game might be compared to chess: a typical middle game in chess has about 30 or 40 moves on each side, and chess playing programs can consistently beat beginners by searching only 3 levels deep; they can play good games by searching 5 levels. The number of moves in a world basis depends on the number of graphs in the canon, the number of laws in L, and the number of graphs in each context. But for many common applications, 30 or 40 moves is a reasonable estimate at any given level, and useful inferences are possible with just a shallow search. The scripts applied by Schank and Abelson (1977), for example, correspond to a game with only one level of look-ahead; a game with two levels would provide the plausible information of scripts together with a round of consistency checks to eliminate obvious blunders.

By deciding how far to search the game tree, one can derive algorithms for plausible inference with varying levels of confidence. Rigorous deduction similar to model elimination (Loveland 1972) can be performed by starting with laws and a context that correspond to the negation of what is to be proved and showing that Opponent has a winning strategy. By similar transformations, methods of plausible and exact inference can be related as variations on a general method of reasoning.

6. Appendix: Summary of the Formalism

This section summarizes axioms, definitions, and theorems about conceptual graphs that are used in this paper. For a more complete discussion and for other features of the theory that are not used here, see the earlier articles by Sowa (1976, 1978).

Definition 1: A *conceptual graph* is a finite, connected, bipartite graph with nodes of the first kind called *concepts* and nodes of the second kind called *conceptual relations*.

Definition 2: Every conceptual relation has one or more *arcs*, each of which must be attached to a concept. If the relation has n arcs, it is said to be *n-adic*, and its arcs are labeled 1, 2, ..., n.

The most common conceptual relations are dyadic (2-adic), but the definition mechanisms can create ones with any number of arcs. Although the formal definition says that the arcs are numbered, for dyadic relations, arc 1 is drawn as an arrow pointing towards the circle, and arc 2 as an arrow pointing away from the circle.

Axiom 1: There is a set T of *type labels* and a function *type*, which maps concepts and conceptual relations into T.

- If *type(a)=type(b)*, then *a* and *b* are said to be of the *same type*.
- Type labels are partially ordered: if *type(a)≤type(b)*, then *a* is said to be a *subtype* of *b*.
- Type labels of concepts and conceptual relations are disjoint, noncomparable subsets of T: if *a* is a concept and *r* is a conceptual relation, then *a* and *r* may never be of the same type, nor may one be a subtype of the other.

Axiom 2: There is a set I={i1, i2, i3, ...} whose elements are called *individual markers*. The function *referent* applies to concepts:

- If *a* is a concept, then *referent(a)* is either an individual marker in I or the symbol @, which may be read *any*.
- When *referent(a)* ∈ I, then *a* is said to be an *individual concept*.
- When *referent(a)=@*, then *a* is said to be a *generic concept*.

In diagrams, the referent is written after the type label, separated by a colon. A concept of a particular cat could be written as [CAT:i4133]. A generic concept, which would refer to any cat, could be written [CAT:@] or simply [CAT]. In data base systems, individual markers correspond to the surrogates (Codd 1979), which serve as unique internal identifiers for external entities. The symbol @ is Codd's notation for null or unknown values in a data base. Externally printable or speakable *names* are related to the internal surrogates by the next axiom.

Axiom 3: There is a dyadic conceptual relation with type label NAME. If a relation of type NAME occurs in a conceptual graph, then the concept attached to arc 1 must be a subtype of WORD, and the concept attached to arc 2 must be a subtype of ENTITY. If the second concept is individual, then the first concept is called a *name* of that individual.

The following graph states that the word "Mary" is the name of a particular person: ["Mary"]→(NAME)→[PERSON:i3074]. If there is only one person named Mary in the context, the graph could be abbreviated to just [PERSON:Mary].

Axiom 4: The *conformity relation* :: relates type labels in T to individual markers in I. If *t*∈T, *i*∈I, and *t::i*, then *i* is said to *conform to t*.

- If *t≤s* and *t::i*, then *s::i*.
- For any type *t*, *t::@*.
- For any concept *c*, *type(c)::referent(c)*.

The conformity relation says that the individual for which the marker *i* is a surrogate is of type *t*. In previous papers, the terms *permissible* or *applicable* were used instead of *conforms to*, but the present term and the symbol :: have been adopted from ALGOL-68. Suppose the individual marker i273 is a surrogate for a beagle named Snoopy. Then BEAGLE::i273 is true. By extension, one may also write the name instead of the marker, as BEAGLE::Snoopy. By axiom 4, Snoopy also conforms to all supertypes of BEAGLE, such as DOG::Snoopy, ANIMAL::Snoopy, or ENTITY::Snoopy.

Definition 3: A *star graph* is a conceptual graph consisting of a single conceptual relation and the concepts attached to each of its arcs. (Two or more arcs of the conceptual relation may be attached to the same concept.)

Definition 4: Two concepts *a* and *b* are said to be *joinable* if both of the following properties are true:

- They are of the same type: *type(a)=type(b)*.
- Either *referent(a)=referent(b)*, *referent(a)=@*, or *referent(b)=@*.

Two star graphs with conceptual relations *r* and *s* are said to be *joinable* if *r* and *s* have the same number of arcs, *type(r)=type(s)*, and for each *i*, the concept attached to arc *i* of *r* is joinable to the concept attached to arc *i* of *s*.

Not all combinations of concepts and conceptual relations are meaningful. Yet to say that some graphs are meaningful and others are not is begging the question, because the purpose of conceptual graphs is to form the basis of a theory of meaning. To avoid prejudging the issue, the term *canonical* is used for those graphs derivable from a designated set called the *canon*. For any given domain of discourse, a canon is defined that rules out anomalous combinations.

Definition 5: A *canon* has three components:

- A partially ordered set T of type labels.
- A set I of individual markers with a conformity relation ::.
- A finite set of conceptual graphs with type of concepts and conceptual relations in T and with referents either @ or markers in I.

The number of possible canonical graphs may be infinite, but the canon contains a finite number from which all the others can be derived. With an appropriate canon, many undesirable graphs are ruled out as noncanonical, but the canonical graphs are not necessarily true. To ensure that only true graphs are derived from true graphs, the laws discussed in Section 4 eliminate inconsistent combinations.

Axiom 5: A conceptual graph is called *canonical* either if it is in the canon or if it is derivable from canonical graphs by one of the following *canonical formation rules*. Let *u* and *v* be canonical graphs (*u* and *v* may be the same graph).

- Copy: An exact copy of *u* is canonical.
- Restrict: Let *a* be a concept in *u*, and let *t* be a type label where *t≤type(a)* and *t::referent(a)*. Then the graph obtained by changing the type label of *a* to *t* and leaving *referent(a)* unchanged is canonical.
- Join on a concept: Let *a* be a concept in *u*, and *b* a concept in *v*. If *a* and *b* are joinable, then the graph derived by the following steps is canonical: First delete *b* from *v*; then attach to *a* all arcs of conceptual relations that had been attached to *b*. If *referent(a)* ∈ I, then *referent(a)* is unchanged; otherwise, *referent(a)* is replaced by *referent(b)*.
- Join on a star: Let *r* be a conceptual relation in *u*, and *s* a conceptual relation in *v*. If the star graphs of *r* and *s* are joinable, then the graph derived by the following steps is canonical: First delete *s* and its arcs from *v*; then for each *i*, join the concept attached to arc *i* of *r* to the concept that had been attached to arc *i* of *s*.

Restriction replaces a type label in a graph by the label of a subtype; this rule lets subtypes inherit the structures that apply to more general types. Join on a concept combines graphs that have concepts of the same type: one graph is overlaid on the other so that two concepts of the same type merge into a single concept; as a result, all the arcs that had been connected to either concept are connected to the single merged concept. Join on a star merges a conceptual relation and all of its attached concepts in a single operation.

Definition 6: Let *v* be a conceptual graph, let *v'* be a subgraph of *v* in which every conceptual relation has exactly the same arcs as in *v*, and let *u* be a copy of *v'* in which zero or more concepts may be restricted to subtypes. Then *u* is called a *projection* of *v*, and *v'* is called a *projective origin* of *u* in *v*.

The main purpose of projections is to define the rule of join on a common projection, which is a generalization of the rules for joining on a concept or a star.

Definition 7: If a conceptual graph *u* is a projection of both *v* and *w*, it is called a *common projection* of *v* and *w*.

Theorem 1: If *u* is a common projection of canonical graphs *v* and *w*, then *v* and *w* may be *joined on the common projection u* to form a canonical graph by the following steps:

- Let *v'* be a projective origin of *u* in *v*, and let *w'* be a projective origin of *u* in *w*.
- Restrict each concept of *v'* and *w'* to the type label of the corresponding concept in *u*.
- Join each concept of *v'* to the corresponding concept of *w'*.
- Join each star graph of *v'* to the corresponding star of *w'*.

The concepts and conceptual relations in the resulting graph consist of those in $v-v'$, $w-w'$, and a copy of u.

Definition 8: If v and w are joined on a common projection u, then all concepts and conceptual relations in the projective origin of u in v and the projective origin of u in w are said to be *covered by the join*. In particular, if the projective origin of u in v includes all of v, then the entire graph v is covered by the join, and the join is called a *covering join* of v by w.

Definition 9: Let v and w be joined on a common projection u. The join is called *extendible* if there exist some concepts a in v and b in w with the following properties:

- The concepts a and b were joined to each other.
- a is attached to a conceptual relation r that was not covered by the join.
- b is attached to a conceptual relation s that was not covered by the join.
- The star graphs of r and s are joinable.

If a join is not extendible, it is called *maximal*.

The definition of maximal join given here is simpler than the one given in Sowa (1976), but it has the same result. Maximal joins have the effect of Wilks' preference rules (1975) in forcing a maximum connectivity of the graphs. Covering joins are used in Section 3 in the rules for applying sequents.

Theorem 2: Every covering join is maximal.

Sowa (1976) continued with further material on quantifiers and procedural attachments, and Sowa (1978) continued with mechanisms for defining new types of concepts, conceptual relations, and composite entities that have other entities as parts. Note that the terms *sort*, *subsort*, and *well-formed* in Sowa (1976) have now been replaced by the terms *type*, *subtype*, and *canonical*.

7. Acknowledgment

I would like to thank Charles Bontempo, Jon Handel, and George Heidorn for helpful comments on earlier versions of this paper.

8. References

Belnap, Nuel D., Jr. (1973) "Restricted Quantification and Conditional Assertion," in Leblanc (1973) pp. 48-75.

Beth, E. W. (1955) "Semantic Entailment and Formal Derivability," reprinted in J. Hintikka, ed., *The Philosophy of Mathematics*, Oxford University Press, 1969, pp. 9-41.

Bobrow, D. G., & T. Winograd (1977) "An Overview of KRL-0, a Knowledge Representation Language," *Cognitive Science*, vol. 1, pp. 3-46.

Charniak, Eugene (1972) *Towards a Model of Children's Story Comprehension*, AI Memo No. 266, MIT Project MAC, Cambridge, Mass.

Codd, E. F. (1979) "Extending the Data Base Relational Model to Capture More Meaning," to appear in *Transactions on Database Systems*.

Deliyanni, Amaryllis, & Robert A. Kowalski (1979) "Logic and Semantic Networks," *Communications of the ACM*, vol. 22, no. 3, pp. 184-192.

Dunn, J. Michael (1973) "A Truth Value Semantics for Modal Logic," in Leblanc (1973) pp. 87-100.

Findler, Nicholas V., ed. (1979) *Associative Networks*, Academic Press, New York.

Gentzen, Gerhard (1934) "Investigations into Logical Deduction," reprinted in M. E. Szabo, ed., *The Collected Papers of Gerhard Gentzen*, North-Holland, Amsterdam, 1969, pp. 68-131.

Heidorn, George E. (1972) *Natural Language Inputs to a Simulation Programming System*, Technical Report NPS-55HD72101A, Naval Postgraduate School, Monterey.

Heidorn, George E. (1975) "Augmented Phrase Structure Grammar," in R. Schank & B. L. Nash-Webber, eds., *Theoretical Issues in Natural Language Processing*, pp. 1-5.

Hendrix, Gary G. (1975) "Expanding the Utility of Semantic Networks through Partitioning," in *Proc. of the Fourth IJCAI*, Tbilisi, Georgia, USSR, pp. 115-121.

Hendrix, Gary G. (1979) "Encoding Knowledge in Partitioned Networks," in Findler (1979) pp. 51-92.

Hewitt, Carl (1972) *Description and Theoretical Analysis (Using Schemata) of PLANNER*, AI Memo No. 251, MIT Project MAC, Cambridge, Mass.

Hintikka, Jaakko (1973) "Surface Semantics: Definition and its Motivation," in Leblanc (1973) pp. 128-147.

Hintikka, Jaakko (1974) "Quantifiers vs. Quantification Theory," *Linguistic Inquiry*, vol. 5, no. 2, pp. 153-177.

Hintikka, Jaakko, & Esa Saarinen (1975) "Semantical Games and the Bach-Peters Paradox," *Theoretical Linguistics*, vol. 2, pp. 1-20.

Leblanc, Hughes, ed. (1973) *Truth, Syntax, and Modality*, North-Holland Publishing Co., Amsterdam.

Loveland, D. W. (1972) "A Unifying View of Some Linear Herbrand Procedures," *Journal of the ACM*, vol. 19, no. 2, pp. 366-384.

McDermott, Drew V. (1976) "Artificial Intelligence Meets Natural Stupidity," *SIGART Newsletter*, No. 57, pp. 4-9.

Minsky, Marvin (1975) "A Framework for Representing Knowledge," in Winston, P. H., ed., *The Psychology of Computer Vision*, McGraw-Hill, New York, pp. 211-280.

Schank, Roger, & Robert Abelson (1977) *Scripts, Plans, Goals and Understanding*, Lawrence Erlbaum Associates, Hillsdale, N. J.

Sowa, John F. (1976) "Conceptual Graphs for a Data Base Interface," *IBM Journal of Research & Development*, vol. 20, pp. 336-357.

Sowa, John F. (1978) "Definitional Mechanisms for Conceptual Graphs," presented at the International Workshop on Graph Grammars, Bad Honnef, Germany, Nov. 1978.

Wilks, Yorick (1975) "Preference Semantics," in E. L. Keenan, ed., *Formal Semantics of Natural Language*, Cambridge University Press, pp. 329-348.

Woods, William A. (1975) "What's in a Link: Foundations for Semantic Networks," in D. G. Bobrow & A. Collins, eds., *Representation and Understanding*, Academic Press, New York.