Logical Structures in the Lexicon

John F. Sowa
IBM Systems Research Institute
500 Columbus Avenue
Thornwood, NY 10594

email: sowa@watson.ibm.com

Abstract

The lexical entry for a word must contain all the information needed to construct a semantic representation for sentences that contain the word. Because of that requirement, the formats for lexical representations must be as detailed as the semantic forms. Simple representations, such as features and frames, are adequate for resolving many syntactic ambiguities. But since those notations cannot represent all of logic, they are incapable of supporting all the function needed for semantics. Richer semantic-based approaches have been developed in both the model-theoretic tradition and the more computational AI tradition. Although superficially in conflict, these two traditions have a great deal in common at a deeper level. Both of them have developed semantic structures that are capable of representing a wide range of linguistic phenomena. This paper compares these approaches and evaluates their adequacy for various kinds of semantic information that must be stored in the lexicon. It presents conceptual graphs as a synthesis of the logicist and AI representations designed to support the requirements of both.

1 Semantics from the Point of View of the Lexicon

To understand a semantic theory, start by looking at what goes into the lexicon. In one of the early semantic theories in the Chomskyan tradition, Katz and Fodor (1963) did in fact start with the lexicon. More recent theories, however, almost treat the lexicon as an afterthought. Yet the essence of the theory is still in the lexicon: every element of the semantic representation of a sentence ultimately derives from something in the lexicon. That principle is just as true for Richard Montague's highly formalized grammar as for Roger Schank's "scruffy" conceptual dependencies, scripts, and MOPs.

Context and background knowledge are also important, since most sentences cannot be understood in isolation. This fact contradicts Frege's principle of compositionality, which says that the meaning of a sentence is derived from the meanings of the words it contains. Yet context can also be stated in words and sentences. Even when nonlinguistic surroundings are necessary for understanding a sentence, every significant feature could be encoded in a sentence; people normally encode such information when they tell a story. More general encyclopedic knowledge is contextual information that was learned at an earlier time; it too could be stated in sentences. An extended Fregean principle should therefore say that the meaning of a sentence must be derivable from the meanings of the words in the sentence together with the meanings of the words in the sentences that describe the relevant context and background knowledge.

Besides the meanings of words, grammar and logic are necessary to combine those meanings into a complete semantic representation. But there are competing theories about how much grammar and logic is necessary, how much is expressed in the lexicon, and how much is expressed in the linguistic system outside the lexicon. Lexically based theories suggest that the grammar rules should be simple and that most of the syntactic complexity should be encoded in the lexicon. One might even go further and say that most of the syntactic complexity isn't syntactic at all. It is the result of interactions among the logical structures of the underlying concepts. In his work on semantically based syntax, Dixon (1991) maintained that syntactic irregularities and idiosyncrasies are not accidental. Instead, he showed that many of them can be predicted from the semantics of the words. Such theories imply that a system would only need a simple grammar to represent a language if it had sufficiently rich semantic structures.

A rich theory of semantics in the lexicon must also explain how the semantics got into the lexicon. A child could learn an initial stock of meanings by associating prelinguistic structures with words. But even those prelinguistic structures are shaped, polished, and refined by long usage in the context of sentences. They are combined with the structures learned from other words, and they are molded into patterns that are traditional in the language and culture. More complex, abstract, and sophisticated concepts are either learned exclusively through language or through experiences that are highly colored and shaped by language. For these reasons, the meaning representations in the lexicon should be derivable from the semantic representations for sentences. As a working hypothesis, the two should be identical: the same knowledge representation language should be used for representing meanings in the lexicon and for representing the semantics of sentences and extended discourse structures.

This paper explores the implications of that hypothesis. Section 2 reviews several different lexical representations and their implications. Section 3 compares the underlying assumptions of the model-theoretic and AI traditions and shows that they are computationally more compatible than their metaphysics would suggest. Section 4 illustrates the use of conceptual graphs for representing the semantic content of lexical entries. Section 5 shows how such representations can be used to handle aspects of language that require both logic and background knowledge.

2 Review of Lexical Representations

Features are one of the oldest and simplest semantic representations. In his *Universal Characteristic*, Leibniz (1679) assigned prime numbers to semantic primitives and represented compound concepts by products of the primes. If RATIONAL were represented by 2 and ANIMAL by 3, then their product 6 would represent RATIONAL ANIMAL or HUMAN. That representation generates a lattice: concept A is a supertype of B if the number for A divides the number for B; the minimal common supertype of A and B corresponds to their greatest common divisor; and the maximal common subtype of A and B corresponds to their least common multiple. Leibniz tried to use his system to mechanize Aristotle's syllogisms, but a feature-based representation is too limited. It cannot distinguish different quantifiers or show how primitive concepts are related to one another in compounds. Modern notations use bit strings instead of products of primes, but their logical power is just as limited as Leibniz's system of 1679.

In their feature-based system, Katz and Fodor (1963) factored the meaning of a word into a string of features and an undigested lump called a *distinguisher*. Following is their representation for one sense of *bachelor*:

bachelor → *noun* → *(Animal)* → *(Male)* → *(Young)*
→ [*fur seal when without a mate during the breeding time*].

In this definition, *noun* is the syntactic marker; the features *(Animal)*, *(Male)*, and *(Young)* are the semantic markers that contain the theoretically significant information; and the phrase in brackets is the unanalyzed distinguisher. Shortly after it appeared, the Katz-Fodor theory was subjected to devastating criticisms. Although no one today uses the theory in its original form, those criticisms are worth mentioning because many of the more modern approaches suffer from the same limitations:

- The sharp distinction between semantic features and the distinguisher is so fundamental to the theory that it should have an enormous impact on the structures of language and the normal use of language. Yet there is no linguistic evidence from syntax or cooccurrence patterns to indicate that it has any effect whatever.
- The distinguisher is made up of words, each of which has its own meaning. A complete semantic theory should explain how the meanings of the words in the distinguisher contribute to the meaning of the whole. But such an analysis would imply a deeper representation that underlies both the features and the distinguisher.
- The Katz-Fodor theory treats different word senses as if they were purely accidental groupings that have no relationship to one another. Yet the four senses of *bachelor* do have something in common. They all represent an immature or transitional stage that leads to some further goal: a student who has completed an academic step on the way to becoming a master or doctor; a young knight who is still an apprentice to another; a seal on its way to full maturity as a patriarch of the herd; or an unmarried man who has not yet started to form his own family. The feature-distinguisher theory does not show the commonality; it cannot explain how these meanings developed from a common root or why they remain associated with the same word form.
- If the features had no deeper structure, there would be nothing to constrain their possible combinations. Yet certain combinations, such as (*Abstract*) + (*Color*) or (*Action*) + (*Weight*), never occur. More structure is needed in the theory to explain why such combinations are impossible.
- Finally, many features cannot be named with a single word. Certain Mexican dialects, for example, make a distinction between a *difunto*, a deceased person who was married at the time of death, and an *angelito*, a deceased person who was not married at the time of death (El Guindi 1986). A feature such as *(Married-at-the-time-of-death)* is so blatantly nonprimitive that it cries out for a theory that represents deeper structures.

These criticisms do not imply that features are useless. But they indicate that features are derived from some deeper, more fundamental representation.

Despite their limitations, features are attractive because they are easy to program, and they generate convenient computational structures, such as Leibniz's lattice. Yet a lattice generated by features permits too many impossible combinations. A well-structured definitional mechanism is necessary to generate lattices without those undesirable nodes.

The prototype for all definitions is Aristotle's method of genus plus differentiae: a new term is defined by specifying its genus or supertype plus the differentiae that distinguish it from other subtypes of the same genus. The result of such definitions is a tree. The tree becomes a lattice when the differentiae are specified in a system of logic that can determine when one definition is implied by another. Such a lattice, variously known as a type, taxonomic, or subsumption hierarchy, forms the basis for many systems of frames and semantic networks. These lattices have several advantages over feature lattices: only the desired nodes are ever defined; most, if not all, of the impossible combinations can be proved to be contradictory in the underlying logic; and the logic can specify relationships that a simple concatenation of features can never represent.

A serious computational problem arises with definitional systems that are based on logic. If the logic is rich enough to express anything in first-order predicate calculus, the proof procedures can become intractable. To simplify the computations, many frame systems adopt a restricted logic, usually without negations or disjunctions. Yet those restricted logics cannot express all the definitions used in language. The definition of *penniless*, for example, requires a negation to express "not having a penny." The word *bachelor* in Katz and Fodor's example requires temporal logic to express "without a mate during the breeding time." In general, every logical quantifier, Boolean operator, and modal operator occurs in dictionary definitions. Although tractability is an important feature in a computational system, a logic that is suitable for natural language semantics must be able to represent anything that people might say. A problem solver or reasoner might have to simplify the statements in order to improve efficiency, but those simplifications are likely to be highly domain-dependent. A general language handler must represent every statement in as rich a form as it was originally expressed.

In several articles written shortly before his death, Richard Montague (1974) began the tradition of applying model-theoretic techniques to natural language semantics. He started with Carnap's intuition (1947) that the intension of a sentence is a function from possible worlds to truth values. He combined that notion with Frege's principle of compositionality to develop a systematic way of deriving the function that represents the meaning of a sentence. The intension of each word in each lexical category would correspond to a functional form of some sort. The intension of the noun *unicorn*, for example, would be a function that applies to entities in the world and generates the value **true** for each unicorn and **false** for each nonunicorn. Other lexical categories would be represented by λ-expressions that would combine with the functional forms that happened to occur to their right or left. As an example, Montague's lexical entry for the word *be* is a function that checks whether the predicate is true of the subject. The idea is straightforward, but his implementation uses a rather heavy notation with many subtle distinctions:

$$\lambda\mathscr{P}\ \lambda x\ (\mathscr{P}(y) \text{ where } \mathrm{ext}(x) = \mathrm{ext}(y)).$$

This λ-expression defines the intension of *be* as a function with two arguments: $\mathscr{P}$ is the intension of the phrase that follows the word *be*, and x is the intension of the subject of *be*. The body of the expression applies the function $\mathscr{P}$ to some y whose extension is equal to the extension of x. Montague's lexicon consists of such constructions that apply functions of functions to generate other functions of functions of functions.[1]

[1] To improve readability, this example modifies Montague's notation by adding the keyword "where" and using the function ext(x) for extension instead of Montague's cryptic marks.

Besides constructing functions, Montague used them to solve certain logical puzzles. One of them is Barbara Partee's example: *The temperature is ninety, and it is rising. Therefore, ninety is rising.* To avoid the conclusion that a constant like ninety could change, Montague drew some subtle distinctions. He treated *temperature* as an "extraordinary noun" that denoted "an individual concept, not an individual". He also gave special treatment to *rise*, which "unlike most verbs, depends for its applicability on the full behavior of individual concepts, not just on their extensions." As a result, he claimed that *The temperature is ninety* asserted the equality of extensions, but that *The temperature is rising* applied the verb *rise* to the intension. Consequently, the conclusion that ninety is rising would be blocked, since *rise* would not be applied to the extension. To linguists, Montague's distinction between words whose semantics depend on intensions and those whose semantics depend on extensions seemed like an *ad hoc* contrivance; he never gave any linguistic evidence to support it. To psychologists, the complex manipulations required for processing the λ-expressions seemed unlikely to have any psychological reality. And to programmers, the infinities of possible worlds seemed computationally intractable. Yet for all its unnaturalness, Montague's system was an impressive achievement: it showed that formal methods of logic could be applied to natural languages, that they could define the semantics of a significant subset of English, and that they could represent logical aspects of natural language with the depth and precision usually attained only in artificial systems of logic.

At the opposite extreme from Montague's logical rigor are Roger Schank's informal diagrams and quasi-psychological theories that were never tested in controlled psychological experiments. Yet they led his students to build impressive demos that exhibited interesting language behavior. As an example, the Integrated Partial Parser (Schank, Lebowitz, & Birnbaum 1980) represents a fairly mature stage of Schank's theories. IPP would analyze newspaper stories about international terrorism, search for words that represent concepts in that domain, and apply scripts that relate those concepts to one another. In one example, IPP processed the sentence, *About 20 persons occupied the office of Amnesty International seeking better jail conditions for three alleged terrorists.* To interpret that sentence, it used the following dictionary entry for the word *occupied*:

```
(WORD-DEF OCCUPIED
  INTEREST 5
  TYPE     EB
  SUBCLASS SEB
  TEMPLATE (SCRIPT  $DEMONSTRATE
            ACTOR   NIL
            OBJECT  NIL
            DEMANDS NIL
            METHOD  (SCENE    $OCCUPY
                     ACTOR    NIL
                     LOCATION NIL))
  FILL     (((ACTOR)        (TOP-OF *ACTOR-STACK*))
            ((METHOD ACTOR) (TOP-OF *ACTOR-STACK*)))
  REQS     (FIND-DEMON-OBJECT
            FIND-OCCUPY-LOC
            RECOGNIZE-DEMANDS))
```

This entry says that *occupied* has interest level 5 (on a scale from 0 to 10) and it is an event builder (EB) of subclass scene event builder (SEB). The template is a script of type

$DEMONSTRATE with an unknown actor, object, and demands. As its method, the demonstration has a scene of type $OCCUPY with an unknown actor and location. At the end of the entry are fill and request slots that give procedural hints for finding the actor, object, location, and demands. In analyzing the sample sentence, IPP identified the 20 persons as the actors, the office as the location, and the better jail conditions as the demand.

The fill and request slots implement the Schankian "expectations." A fill slot is filled with something previously found in the sentence, and a request slot waits for something still to come. They serve the same purpose as Montague's left and right cancellation rules for categorial grammar. The act of filling the slots corresponds to the λ rules for expanding a function that is applied to a list of arguments. Their differences in style are more significant than their differences in computational mechanisms:

- Schank's antiformalist stance is irrelevant, since anything that can be programmed on a digital computer could be formalized. One Prolog programmer, in fact, showed that most of the slot filling in Schank's parsers and script handlers could be done directly by Prolog's unification algorithm. Techniques such as unification and graph grammars could be used to formalize Schank's systems while making major improvements in clarity, robustness, and generality.
- Montague's appearance of rigor results from his use of Greek letters and logical symbols. Yet some constructions, such as his solution to Partee's puzzle, are contrivances that programmers would call "hacks" (if they ever took the trouble to work their way through his notation).
- Schank and Montague had different attitudes about what aspects of language were most important. Schank believed that the ability to represent and use world knowledge is the essence of language understanding, and Montague believed that the ability to handle scope of quantifiers and modalities was the most significant. They were both right in believing that their favorite aspects were important, but they were both wrong in ignoring the others.

Schank and Montague represented different aspects of language with different methodologies, but they are complementary rather than conflicting. Yorick Wilks (1991) observed that Montague's lexical entries are most complex for words like *the*, for which Schank's entries are trivial. Conversely, Schank's entries are richest for content words, which Montague simply put in one of his categories, while ignoring their connotations. Both logic and background knowledge are important, and the lexicon must include both kinds of information.

3 Metaphysical Baggage and Observable Results

Linguistic theories are usually packaged in metaphysical terms that go far beyond the available evidence. Chomsky's metaphysics may be summarized in a single sentence from *Syntactic Structures*: "Grammar is best formulated as a self-contained study independent of semantics." For Montague, the title and opening sentence of "English as a Formal Language" express his point of view: "I reject the contention that an important theoretical difference exists between formal and natural languages." Schank's outlook is summarized in the following sentence from *Conceptual Information Processing*: "Conceptual Dependency Theory was always intended to be a theory of how humans process natural

language that was explicit enough to allow for programming it on a computer." These characteristic sentences provide a key to understanding their authors' motivation. Yet their actual achievements are easier to understand when the metaphysics is ignored. Look at what they do, not at what they say.

In their basic attitudes and metaphysics, Schank and Montague are irreconcilable. Montague is the epitome of the kind of logician that Schank has always denounced as misguided or at least irrelevant. Montague stated every detail of his theory in a precise formalism, while Schank made sweeping generalizations and left the detailed programming to his students. For Montague, the meaning of a sentence is a function from possible worlds to truth values; for Schank, it is a diagram that represents human conceptualizations. On the surface, their only point of agreement is their implacable opposition to Chomsky and "the developments emanating from the Massachusetts Institute of Technology" (Montague 1970). Yet in their reaction against Chomsky, both Montague and Schank evolved positions that are remarkably similar, although their terminology hides the resemblance. What Chomsky called a noun, Schank called a picture producer, and Montague called a function from entities to truth values. But those terms are irrelevant to anything that they ever did: Schank never produced a single picture or even stated a plausible hypothesis about how one might be produced from his diagrams; Montague never applied any of his functions to the real world, let alone the infinity of possible worlds he so freely assumed. In neutral terms, what Montague and Schank did could be described in a way that makes the logicist and AI points of view nearly indistinguishable:

1. Semantics, not syntax, is the key to understanding sentence structure. The traditional grammatical categories are surface manifestations of the fundamental semantic categories.
2. Associated with each word is a characteristic semantic structure that determines how it combines with other words in a sentence.
3. The grammar of a language can be reduced to relatively simple rules that show what categories of words may occur on the right or the left of a given word (the cancellation rules of categorial grammar or the Schankian expectations). The variety of sentence patterns is not the result of a complex grammar, but of the complex interactions between a simple grammar and the underlying semantic structures.
4. The meaning of a sentence is derived by combining the semantic structures for each of the words it contains. The combining operations are primarily semantic, although they are guided by word order and inflections.
5. The truth of a sentence in a possible world is computed by evaluating its meaning representation in terms of a model of that world. Although Schank never used logical terms like *denotation*, his question-answering systems embodied effective procedures for computing denotations, while Montague's infinities were computationally intractable.

Terms like *picture producer* or *function from entities to truth values* engender heated arguments, but they have no effect on the application of the theory to language or its implementation in a computer program. Without the metaphysical baggage, both theories incorporate a semantic-based approach that is widely accepted in AI and computational linguistics.

At the level of data structures and operations, there are significant differences between Montague and Schank. Montague's representations were λ-expressions, which have the associated operations of functional application, λ-expansion, and λ-contraction. His metaphysics gave him a rigorous methodology for assigning each word to one of his categories of functions (even though he never actually applied those functions to the real world or any possible world). And his concerns about logic led him to a careful treatment of quantifiers, modalities, and their scope. Schank's representations are graphs on paper and LISP structures of various kinds in his students' programs. The permissible operations include any manipulations of those structures that could be performed in LISP. Schank's lack of a precise formalism gave his students the freedom and flexibility to invent novel solutions to problems that Montague's followers never attempted to address, such as the use of world knowledge in language understanding. Yet that lack of formalism led to *ad hoc* accretions in the programs that made them unmaintainable. Many of Schank's students found it easier to start from scratch and write a new parser than to modify one that was written by an earlier generation of students. Montague and Schank have complementary strengths: rigor vs. flexibility; logical precision vs. open-ended access to background knowledge; exhaustive analysis of a tiny fragment of English vs. a broad-brush sketch of a wide range of language use.

Montague and Schank represent two extremes on the semantic-based spectrum, which is broad enough to encompass most AI work on language. Since the extremes are more complementary than conflicting, it is possible to formulate approaches that combine the strengths of both: a precise formalism, the expressive power of intensional logic, and the ability to use background knowledge in language understanding. To allow greater flexibility, some of Montague's rigid constraints must be relaxed: his requirement of a strict one-to-one mapping between syntactic rules and semantic rules; his use of λ-expressions as the meaning representation; and his inability to handle ellipses, metaphor, metonymy, anaphora, and anything requiring background knowledge. With a well-designed formalism, these constraints could be relaxed while still allowing formal definitions of the permissible operations.

4 Lexical Representations in Conceptual Graphs

Without a formal system of logic, the issues that a lexical theory must address can only be discussed at an anecdotal level. A formal system can clarify the issues, make them precise, and lead the discussion to a deeper level of detail. This paper uses the theory of conceptual graphs: a system of logic with a graph notation designed for a direct mapping to and from the semantic structures of natural language. They have a graph structure based on the semantic networks of AI and C. S. Peirce's existential graphs, which form a complete system of logic. They are as formal and expressive as Montague's intensional logic, but they permit a broader range of operations on the formalism. The theory has been presented in book form (Sowa 1984) and in a recent summary (Sowa 1991); see those sources for more detail. An earlier paper on lexical issues with conceptual graphs (Sowa 1988) did not discuss logic explicitly. This section will give some examples to illustrate the kind of logical structures that are needed in the lexicon.

A basic feature of conceptual graph theory is the use of *canonical graphs* to represent the expected roles associated with each concept type. Canonical graphs are similar to the case frames used in many systems, but they are richer in the kinds of structures and logical

operators they support. As an example, Figure 1 shows a canonical graph that represents the lexical pattern associated with the verb *support*, It shows that every instance of the concept type SUPPORT has four expected participants: an animate agent, some entity as patient, some entity as instrument, and a purpose, which is represented by a nested context. That context, which might represent something at a different time and place from the outer context, shows that the entity is in some state.

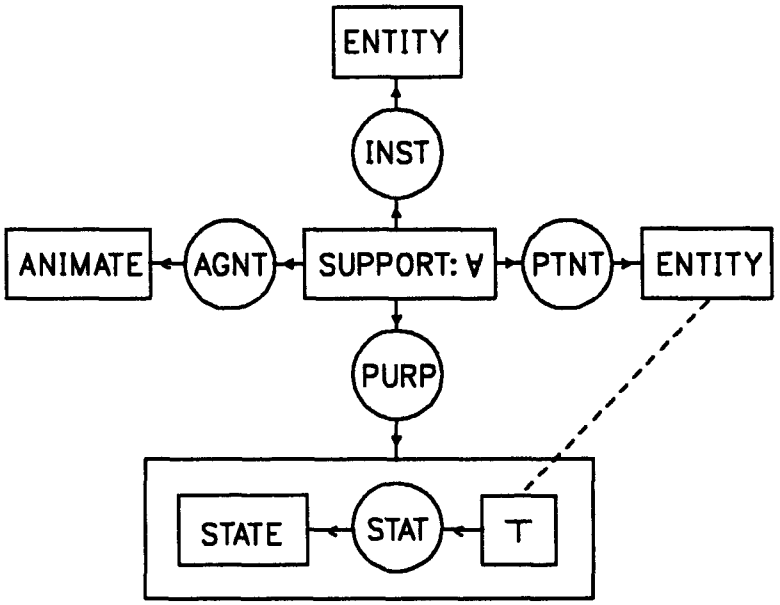

Figure 1. Canonical graph for the lexical type SUPPORT

Whereas case frames merely show the thematic roles for a verb and the expected concept types that can fill those roles, conceptual graphs can grow arbitrarily large: they can show long-range dependencies far removed from the central concept; and they may contain nested contexts that show situations at different times and in different modalities. The dotted line in Figure 1 is a *coreference link* that crosses context boundaries; it shows that the entity that is the patient of SUPPORT is coreferent with the thing in the nested context.

Conceptual graphs are a complete system of logic with their own model-theoretic semantics, but there is also a *formula operator* ϕ that maps conceptual graphs into predicate calculus. Following is the result of applying ϕ to Figure 1:

```
(∀x)(∃y)(∃z)(∃u)(∃v)(support(x) ⊃
   (animate(y) ∧ entity(z) ∧ entity(u) ∧ situation(v) ∧
      agnt(x,y) ∧ ptnt(x,z) ∧ inst(x,u) ∧ purp(x,v) ∧
         description(v, (∃w)(state(w) ∧ stat(z,w))))).
```

This formula and the graph in Figure 1 express exactly the same information with identical ontological presuppositions. The first three lines of the formula represent the standard

information that is typical of case frames. The fourth line, however, represents structures typical of situation semantics. It says that the situation v is described by a formula, which says that there exists a state w and that the entity z is in state w; i.e. the purpose of supporting is to maintain an entity in some state. The predicate description(s,p) means that a situation s is described by a proposition p. This is one of the predicates that result from the context-creating boxes in conceptual graphs, which are necessary for representing a variety of structures in language.

Besides being more readable, the graph contains structures that are not supported in predicate calculus, such as the context box that encloses the purpose of the supporting. That box itself represents a concept to which relations can be attached to express purpose, causality, time sequence, and other intersentential connectives. The description predicate in the formula requires a version of higher order logic with nested propositions and quantification over the situations described by those propositions. Although the description predicate allows some contexts to be translated into predicate calculus by ϕ, there are other features associated with contexts that cannot be translated. One such feature is the *indexical referent* #, which is used to represent unresolved anaphoric references, tenses, and other context-dependent phenomena. The formula operator ϕ is undefined for graphs containing # until those references have been resolved.

As another example, Figure 2 shows the canonical graphs for the concepts EASY and EAGER, which are used for the adjectives *easy* and *eager* as well as the adverbs *easily* and *eagerly*.

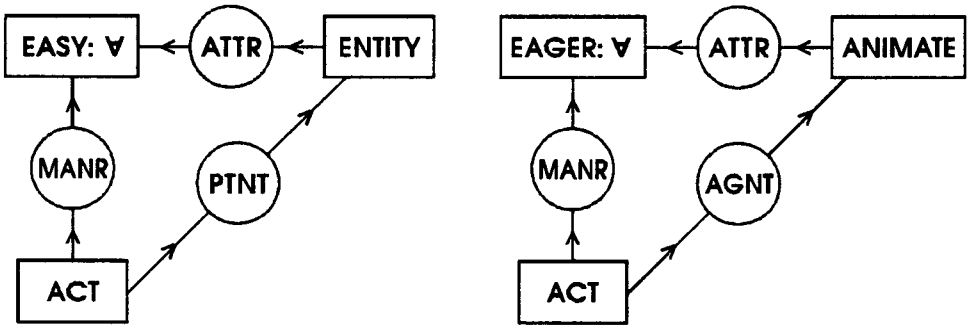

Figure 2. Canonical graphs for EASY and EAGER

The first graph says that every instance of EASY is an attribute of some ENTITY, and it is also the manner of some ACT. That ENTITY also happens to be the patient of the same ACT. The graph for EAGER has the same shape as the graph for EASY, but EAGER is an attribute of some ANIMATE being that is the agent of some ACT. These graphs illustrate the point that many syntactic features result from deeper logical properties. In this case, the AGNT and PTNT relations have different preferences for expression as subject or object. As a result, the canonical graphs permit sentences of the following form:

John easily does the homework.
John eagerly does the homework.

The homework is easy for John to do.

But they rule out the following sentences:

* *The homework is eager for John to do.*
* *John is easy to do the homework.*

Sowa and Way (1986) showed how such graphs could be used in a semantic interpreter. The grammar would only require general rules for adjectives and adverbs; no features would be required to distinguish special properties of *easy* and *eager* or their adverbial forms. Instead, the correct options would be selected and the incorrect ones would be blocked by the graph unification operations (the maximal joins of conceptual graphs).

Besides the display form for conceptual graphs (Figures 1 and 2), there is also a more compact linear form. Following are the linear representations for the graphs in Figure 2:

```
[EASY: ∀]-
   (ATTR)←[ENTITY]←(PTNT)←[ACT: *x]
   (MANR)←[*x].

[EAGER: ∀]-
   (ATTR)←[ANIMATE]←(AGNT)←[ACT: *x]
   (MANR)←[*x].
```

In the linear notation, concepts are enclosed in square brackets, and conceptual relations are enclosed in parentheses. The hyphens show that additional relations attached to a node are continued on separate lines. The variable *x indicates that the concept [*x] represents the same node as the concept [ACT: *x]. Converting from the box and circle notation to the linear notation causes cycles be broken, and variables like *x are needed to show cross-references. The point at which the cycle is broken is purely arbitrary; different linearizations represent exactly the same graph.

The examples cited in Section 2 — *penniless, difunto,* and *angelito* — can also be defined in conceptual graphs. The definition of PENNILESS, for example, requires a negation:

```
PENNILESS = (λx) [STATE: *x]←(STAT)←[PERSON: *y]
   ¬[ [*y]→(POSS)→[PENNY]].
```

This definition says that PENNILESS is a state x of a person y, where it is false that y has possession (POSS) of a penny. Systems that do not allow negations in definitions may make the reasoning process faster, but they make it impossible to define many kinds of concepts. The definition for the concept type DIFUNTO would require a relation WHEN linking two situations:

```
DIFUNTO = (λx) [PERSON: *x]→(STAT)→[DEAD]
   [SITUATION: [*x]→(STAT)→[MARRIED]]→(WHEN)→[SITUATION: [*x]←(PTNT)←[DIE]].
```

By this definition, a difunto is a person x in state dead, where x was in a situation of being married when a situation occurred in which x died. The definition of angelito is similar, but with a negation inside the first situation. In reasoning by inheritance, both DIFUNTO and ANGELITO could be treated as simple subtypes of DEAD-PERSON, and the details inside the definition could be ignored. But to determine whether a partic-

ular individual was a difunto or an angelito, the details could be recovered by expanding the λ-expression.

New types of conceptual relations can also be defined by λ-abstraction. The relation WHEN in the previous examples could be defined by the following graph:

```
WHEN = (λx,y)
   [SITUATION: *x]→(PTIM)→[TIME]←(PTIM)←[SITUATION: *y].
```

In this definition, WHEN has two formal parameters x and y; each of them refers to a situation that occurs at the same point in time (PTIM). Any occurrence of the relation WHEN in a graph could be replaced by the sequence of concepts and relations between [*x] and [*y]. The graph in the definition of DIFUNTO could be expanded to the following:

```
[SITUATION: [*x]→(STAT)→[MARRIED]]→(PTIM)→[TIME]-
   (PTIM)←[SITUATION: [*x]←(PTNT)←[DIE]].
```

This graph says that x is in the state married at some point in time, which is the same time that x dies. Since the graph is too long to fit on one line, the hyphen shows that the relation attached to [TIME] is continued on the next line.

In Section 2, one criticism of Katz and Fodor's theory was its inability to show the commonality among different senses of the same word. Some linguists, such as Ruhl (1989), even maintain a principle of *monosemy*: each word has a single very abstract meaning, and the multiple senses that appear in dictionary definitions are the result of applying the word in different domains. For Katz and Fodor's example of *bachelor*, there is a such a unifying meaning: "an animate being x preparing for a situation in which x is in a mature state." That definition could also be expressed in a conceptual graph:

```
BACHELOR = (λx) [ANIMATE: *x]←(AGNT)←[PREPARE]-
   (PURP)→[SITUATION: [*x]→(STAT)→[STATE]→(ATTR)→[MATURE]].
```

This central meaning for the concept type BACHELOR explains why Pope John Paul II, who is an unmarried man, would not be called a bachelor: he has already achieved the ultimate stage in his profession, which has celibacy as a precondition.

Partee's puzzle about the temperature may be represented in conceptual graphs by distinguishing temperature as a state from its measure as a number. The sentence *The temperature is 90* would therefore be treated as an abbreviation for the sentence *The measure of the temperature is 90.* Such abbreviations are common in ordinary language, and words such as *measure* and *amount* are evidence for a distinction that is familiar to most speakers. The following graph shows that the temperature is in the state of RISE and its current measure happens to be 90°F.

```
[TEMP-MEASURE: 90F]←(MEAS)←[TEMPERATURE: #]→(STAT)→[RISE].
```

In this graph, the temperature is in a state of rising. Since its measure of 90°F is not directly attached to [RISE], the value of 90 will not change. Instead, the temperature at a later time will have a different measure. Unlike Montague's subtle distinctions, this solution to Partee's puzzle is based on concepts derived from the words and phrases used by people who talk about temperature.

As an example of a complex sentence containing nested contexts and quantification, Figure 3 shows the graph for a sentence that defines *Prix Goncourt* as "an institution

comprising a panel of judges who each year award a prize of money to an author who published an outstanding literary work."

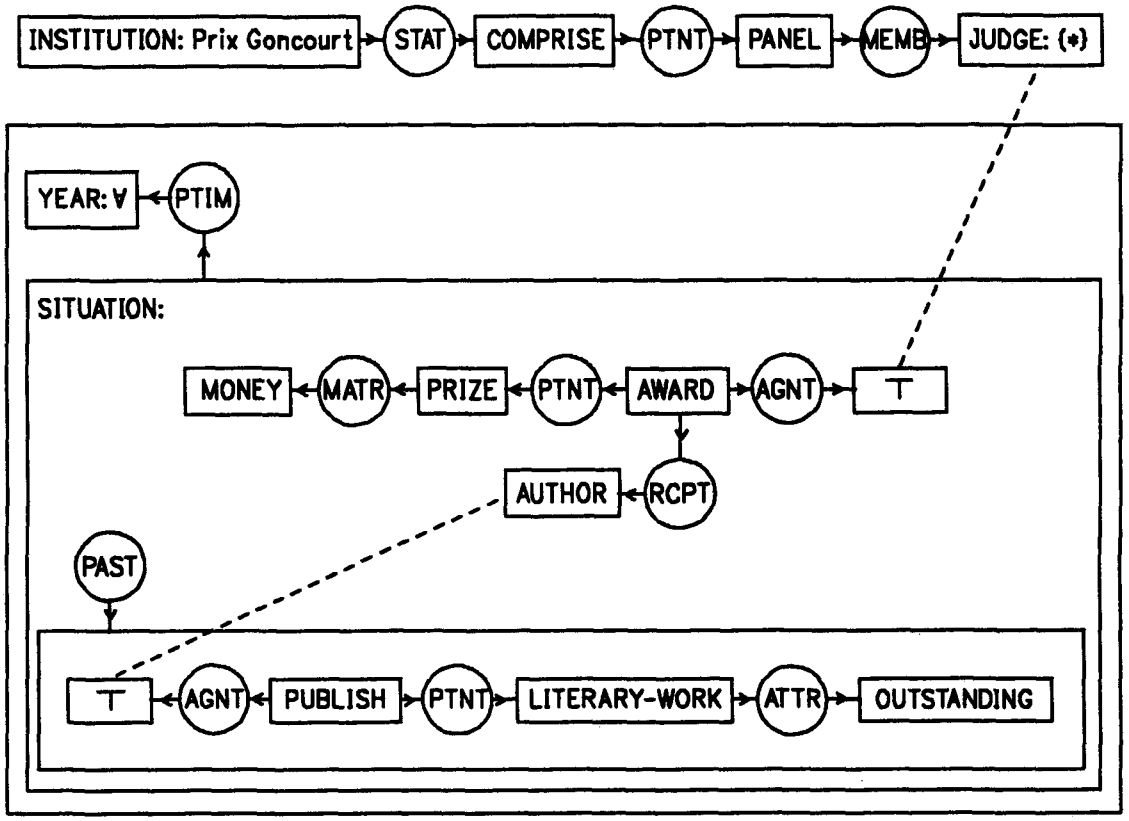

Figure 3. Conceptual graph that defines the Prix Goncourt

In Figure 3, the quantifier ∀ permits the concept [YEAR] to be instantiated with different years, in each of which a separate author is awarded a separate instance of the prize. The relation (PAST) shows the past tense of *published*; that relation is not a primitive, since it may be expanded according to the following definition:

PAST = (λx)[SITUATION: *x]→(PTIM)→[TIME]←(SUCC)←[TIME: #].

This defines (PAST) as a monadic relation with a formal parameter x. It applies to a situation whose point in time (PTIM) is a successor to some contextually defined time. The marker # indicates a reference to be resolved to the point in time of the containing context, in this case the year of the award — i.e. the publishing occurred before the awarding.

5 Operations on Knowledge in the Lexicon

The purpose of a rich semantic representation is to support a rich set of operations on the representation. Of the various reasoning systems that have been implemented for conceptual graphs, some are theorem provers that are based on the logical structure of the graphs; others emphasize heuristic techniques based on semantic distance measures; and others combine logic and heuristics to speed up the proofs. Fargues et al. (1986) implemented a Prolog-like theorem prover with conceptual graphs as the replacement for the usual predicates. It incorporated several advances over standard Prolog: arbitrarily large graphs as the unit of inference instead of single predicates; semantic unification that derives maximal common supertypes when joining graphs; and the ability to do λ-expansion and contraction of types. Garner and Tsui (1988) implemented an inference engine that used heuristics and semantic distance measures to guide the proofs. They demonstrated that it could handle the kinds of scripts processed by the Schankian systems, but with a more robust, formally defined representation. Hartley and Coombs (1991) showed how conceptual graphs could be used in abductive reasoning for generating models that satisfied given constraints.

Interpreting nonliteral language is an application where background knowledge is essential. Way (1991) presented a book-length study of metaphor using conceptual graphs. According to her hypothesis, the purpose of a good metaphor is to refine the concept hierarchy by creating a new type. As a result of interpreting a metaphor, a word is generalized to a concept type that includes the original meaning plus a more abstract meaning that can be transferred to a new domain. Her approach takes advantage of the operations for joining and projecting graphs and the λ-expressions for defining new concept types.

Metonymy is another kind of nonliteral language that requires detailed semantic structures in the lexicon and detailed operators for processing them. As an example of metonymy, consider the sentence *The White House announced the budget.* Syntactically, *White House* is the subject of *announce*, but semantic constraints rule out the building as a possible agent of the concept ANNOUNCE. Therefore, a semantic interpreter might construct a graph with "subj" as a syntactic annotation on the relation, but with the type of relation unspecified:

```
[BUILDING: White House]←(; subj)←[ANNOUNCE]→(PTNT)→[BUDGET: #].
```

After constructing this graph in the parsing stage, the semantic interpreter must determine the unknown relation type and insert it in front of the semicolon. It could search for background knowledge about the White House, discovering that people work there who make announcements. From the graphs that state that knowledge, it could abstract the following λ-expression to define a possible relation between an act and a building:

```
(λx,y) [ACT: *x]→(AGNT)→[PERSON]→(LOC)→[BUILDING: *y].
```

This definition relates an act x whose agent is a person located in a building y. The entire λ-expression could then be inserted just before the semicolon of the undefined relation:

```
[BUILDING: White House]→(
   (λx,y) [ACT: *x]→(AGNT)→[PERSON]→(LOC)→[BUILDING: *y];
      subj)←[ANNOUNCE]→(PTNT)→[BUDGET: #].
```

When the λ-expression is expanded, the concept marked by x (the first parameter) is joined to the concept with the arrow pointing towards the relation; and the concept

marked by y (the second parameter) is joined to the concept with the arrow pointing away from the relation. The syntactic annotation "subj" must be dropped, since no single relation in the expansion exactly corresponds to the original subject of the sentence.

```
[BUILDING: White House]←(LOC)←[PERSON]←(AGNT)←[ANNOUNCE]-
   (PTNT)→[BUDGET: #].
```

This graph represents the expanded sentence *A person at the White House announced the budget*. The option of omitting the conceptual relation in cases of metonymy or ellipsis allows certain decisions to be deferred until additional information is obtained from the context or from background knowledge.

As another example of metonymy, consider the problem of multiple meanings of the term *Prix Goncourt*, discussed by Kayser (1988). He found seven metonyms for that term: a literary prize, the money awarded as the prize, the person who received the prize, the panel that awards the prize, the book that won the prize, the time that the prize was won, or the institution that grants a new instance of the prize each year. Following are his sample sentences and their English translations:

- Prize: *Le PG a été attribué à X.* [The PG was awarded to X.]
- Money: *X a versé son PG à la Croix Rouge.* [X turned over his PG to the Red Cross.]
- Person: *Le PG a été félicité par le Président.* [The PG was congratulated by the President.]
- Panel: *Le PG a admis un nouveau juré.* [The PG admitted a new judge.]
- Book: *Peux-tu aller m'acheter le PG à la librairie X?* [Could you go buy the PG for me at bookstore X?]
- Time: *Depuis son PG, il est devenu arrogant.* [Since his PG, he has become arrogant.]
- Institution: *Le PG pervertit la vie littéraire.* [The PG perverts the literary life.]

Each of these metonyms could be interpreted by the method of constructing a λ-expression for the unknown relation. The graph in Figure 3 provides the basic background knowledge for constructing those relations.

With Figure 3 as background knowledge, each metonym of *Prix Goncourt* can be determined by finding a suitable path through the graph and mapping it into a λ-expression that defines the unknown relation. The verbs in the input sentences impose selectional constraints that determine the direction the path may take. When the constraints imposed by the verb are not strong enough, additional background knowledge derived from other words in the input sentence may be needed; that knowledge could be represented in other conceptual graphs that would also be joined to the input graph. Following is a sketch of how such a system could interpret each metonym:

- Prize: The verb *attribué* in the input maps to the concept [AWARD]. The corresponding concept in the background graph is linked to [PRIZE] by the relation (PTNT). A maximal join of the input graph to the background graph starting with the two concepts of type AWARD would automatically associate PG with the node [PRIZE].
- Money: X could present either the prize itself or the money of the prize to the Red Cross. Background knowledge that people give money to charitable organizations

would lead to a preference for [MONEY]. That knowledge could be represented in a separate conceptual graph triggered by the term *Red Cross*.

- Person: The verb *félicité* maps to [CONGRATULATE], with its selectional constraints for PERSON or a subtype such as AUTHOR. Judges are also persons, but the node [JUDGE: {*}] is marked as plural by the symbol {*} and is therefore unlikely to be indicated as *the* Prix Goncourt. Information about salience should also be marked on the graph: [AUTHOR], [PRIZE], and [LITERARY-WORK] are more salient and hence more likely to be selected.
- Panel: The verb *admis* maps to the concept [ADMIT], which would select an agent of type PERSON or a collection of persons, such as a panel. But that constraint would permit either [AUTHOR], [JUDGE], or [PANEL] as the agent. The remainder of the sentence *un nouveau juré* introduces the concept [JUDGE], which would unify with the set of judges linked by the member relation to [PANEL]. The preference rule for increased connectivity would select the concept [PANEL], especially since one sense of ADMIT would include the admission of a member to a set.
- Book: The verb *acheter* maps to [BUY], which would prefer a nonhuman physical entity as patient. Buying a prize is possible, but that might suggest bribing the panel. The background knowledge that bookstores sell books would give a strong preference for BOOK, which would unify with [LITERARY-WORK] (although this is another example of metonymy, since *book* could refer to the literary work or to a physical object in which the work is printed).
- Time: The preposition *depuis* requires a point in time as its object. The concept [YEAR: $\forall$] indicates an entire series of possible times. The possessive pronoun *son*, coreferent with *il*, indicates a particular person, which would most likely select the node [AUTHOR], which would occur in one instance of a year, which would then be the correct time.
- Institution: The verb *pervertit* maps to [PERVERT], which could have a human as agent or almost anything as instrument, either of which might occur in subject position. But the present tense of the verb suggests a continuing influence; therefore, the subject must be something outside the scope of the quantifier on [YEAR: $\forall$]. Since [AUTHOR], [PRIZE], and [MONEY] are all inside that scope, there would be a separate instance of them for each year. A continuing perversion could only be exerted by something outside that scope, such as the institution or the panel; when either is permissible, salience might prefer the node [INSTITUTION].

Once a concept node has been selected by one of these mechanisms, the correct metonym could then be defined by a λ-abstraction over the graph with that node marked as the formal parameter. As these examples illustrate, the process of interpretation is complex: it requires a great deal of domain-dependent knowledge; and it must be sensitive to many syntactic and semantic features, including verb tenses, definite and indefinite articles, and quantifier scopes. Yet the kind of analysis required, although complex, is within the realm of what is computable — but only if the background knowledge and lexical entries are encoded in a suitably rich knowledge representation language.

6 Towards a Synthesis of the Logicist and AI Traditions

Linguists who work in the tradition of Noam Chomsky are fond of saying that semantic theory is not as well developed as syntax. That may be true of their work, but it is not true of the model-theoretic tradition that follows from the work of Richard Montague. Nor is it true of the computational work in the AI tradition. These two approaches, which are often considered diametrically opposed, have complementary strengths and weaknesses. With a suitable knowledge representation, it is possible to have the best of both: a formal system of logic that can accommodate background knowledge as used in AI systems. Features and frames are too weak to serve as a complete system of logic. Graphs are potentially much more powerful, but they must be formalized in order to support all logical operators. The inventor of the modern linear notation for predicate calculus was C. S. Pierce, who later abandoned the linear form in favor of his *existential graphs*, which he called "the logic of the future." Remarkably, Peirce's graphs have a context structure that is isomorphic to Kamp's Discourse Representation Structures (1981). As a synthesis of Peirce's graphs with the AI work on semantic networks, conceptual graphs benefitted from a stroke of serendipity: their contexts can directly support Kamp's rules for resolving anaphora, even though that was not one of their original design criteria. A great deal of research is undoubtedly necessary to support all the semantic structures of language, but these felicitous convergences give hope that such a synthesis of the logicist and AI traditions is proceeding in the right direction.

References

Dixon, R. M. W. (1991) *A New Approach to English Grammar on Semantic Principles*, Oxford University Press, New York.

Fargues, Jean, Marie Claude Landau, Anne Dugourd, & Laurent Catach, (1986) "Conceptual graphs for semantics and knowledge processing," *IBM Journal of Research and Development* **30:1**, 70-79.

Garner, B.J., & Tsui, E. (1988) "General purpose inference engine for canonical graph models," *Knowledge Based Systems* **1:5**, pp. 266-278.

Hartley, Roger T., & Michael J. Coombs (1991) "Reasoning with graph operations," in J. F. Sowa, ed., *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, Morgan Kaufmann Publishers, San Mateo, CA, pp. 487-505.

Kamp, Hans (1981) "Events, discourse representations, and temporal references," *Langages* **64**, 39-64.

Katz, Jerrold J., & Jerry A. Fodor (1963) "The structure of a semantic theory," *Language* **39**, 170-210. Reprinted in J. A. Fodor & J. J. Katz, eds. (1964) *The Structure of Language*, Prentice-Hall, Englewood Cliffs, NJ, pp. 479-518.

Kayser, Daniel (1988) "What kind of thing is a concept?" *Computational Intelligence* **4:2**, 158-165.

Montague, Richard (1970) "English as a formal language," reprinted in Montague (1974), pp. 188-221.

Montague, Richard (1974) *Formal Philosophy*, Yale University Press, New Haven.

Ruhl, Charles (1989) *On Monosemy: A Study in Linguistic Semantics*, State University of New York Press, Albany.

Schank, Roger C., ed. (1975) *Conceptual Information Processing*, North-Holland Publishing Co., Amsterdam.

Schank, Roger C., Michael Lebowitz, & Lawrence Birnbaum (1980) "An integrated understander," *American Journal of Computational Linguistics* **6**, 13-30.

Sowa, John F. (1976) "Conceptual graphs for a database interface," *IBM Journal of Research and Development* **20:4**, pp. 336-357.

Sowa, John F. (1984) *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, Reading, MA.

Sowa, John F. (1988) "Using a lexicon of canonical graphs in a semantic interpreter," in M. Evens, ed., *Relational Models of the Lexicon*, Cambridge University Press, pp. 73-97.

Sowa, John F. (1991) "Towards the expressive power of natural language," in J. F. Sowa, ed., *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, Morgan Kaufmann Publishers, San Mateo, CA, pp. 157-189.

Way, Eileen C. (1991) *Dynamic Type Hierarchies*, Kluwer Academic Publishers.

Wilks, Yorick A. (1991) Personal communication.