FLUSH: A Flexible Lexicon Design

David J. Besemer and Paul S. Jacobs
Artificial Intelligence Branch
GE Corporate Research and Development
Schenectady, NY 12301 USA

Abstract

Approaches to natural language processing that use a phrasal lexicon have the advantage of easily handling linguistic constructions that might otherwise be extragrammatical. However, current phrasal lexicons are often too rigid: their phrasal entries fail to cover the more flexible constructions. FLUSH, for Flexible Lexicon Utilizing Specialized and Hierarchical knowledge, is a knowledge-based lexicon design that allows broad phrasal coverage.

I. Introduction

Natural language processing systems must use a broad range of lexical knowledge to account for the syntactic use and meaning of words and constructs. The problem of understanding is compounded by the fact that language is full of *nonproductive* constructs—expressions whose meaning is not fully determined by examining their parts. To handle these constructs, some systems use a phrasal lexicon [Becker, 1975, Wilensky and Arens, 1980b, Jacobs, 1985b, Steinacker and Buchberger, 1983, Dyer and Zernik, 1986], a dictionary designed to make the representation of these specialized constructs easier.

The problem that phrasal lexicons have is that they are too rigid: the phrasal knowledge is entered in a way that makes it difficult to represent the many forms some expressions may take without treating each form as a distinct "phrase". For example, expressions such as "send a message", "give a hug", "working directory", and "pick up" may be handled as specialized phrases, but this overlooks similar expressions such as "give a message", "get a kiss", "working area", and "take up". Specialized constructs must be recognized, but much of their meaning as well as their flexible linguistic behavior may come from a more general level.

A solution to this problem of rigidity is to have a hierarchy of linguistic constructions, with the most specialized phrases grouped in categories with other phrases that behave similarly. The idea of a linguistic hierarchy is not novel, having roots in both linguistics [Lockwood, 1972, Halliday, 1978] and Artificial Intelligence [Sondheimer *et al.*, 1984]. Incorporating phrasal knowledge into such a hierarchy was suggested in some AI work [Wilensky and Arens, 1980a], but the actual implementation of a hierarchical phrasal lexicon requires substantial extensions to the phrasal representation of such work.

The Flexible Lexicon Utilizing Specific and Hierarchical knowledge (FLUSH) is one component in a suite of natural language processing tools being developed at the GE Research and Development Center to facilitate rapid assimilation of natural language processing technology to a wide variety of domains. FLUSH has characteristics of both traditional and phrasal lexicons, and the phrasal portion is partitioned into four classes of phrasal entries:

- word sequences
- lexical relations
- linguistic relations
- linguistic/conceptual relations

FLUSH's mechanisms for dealing with these four classes of specialized phrases make use of both general and specific knowledge to support extensibility.

FLUSH is the lexical component of a system called TRUMP (TRansportable Understanding Mechanism Package) [Jacobs, 1986b], used for language analysis in multiple domains. This paper will describe the phrasal knowledge base of FLUSH and its use in TRUMP.

II. Compound Lexical Knowledge in FLUSH

Because the knowledge embodied in single word lexemes is not enough to account for nonproductive expressions, FLUSH contains phrasal entries called *compound lexemes*. This section first illustrates how each of the four classes of compound lexemes is represented in FLUSH and then describes the algorithm for accessing the compound lexemes. So that the reader is better equipped to understand the figures in the rest of this paper, the next paragraph briefly introduces the knowledge representation scheme that is employed by FLUSH.

Knowledge representation in FLUSH is uses Ace [Jacobs and Rau, 1984, Jacobs, 1985a], a hierarchical knowledge representation framework based on structured inheritance. Most of Ace's basic elements can be found in other knowledge representation schemes (e.g., *isa* links, slots, and inheritance)[Bobrow and Winograd, 1977, Brachman and Schmolze, 1985, Wilensky, 1986], but Ace has the

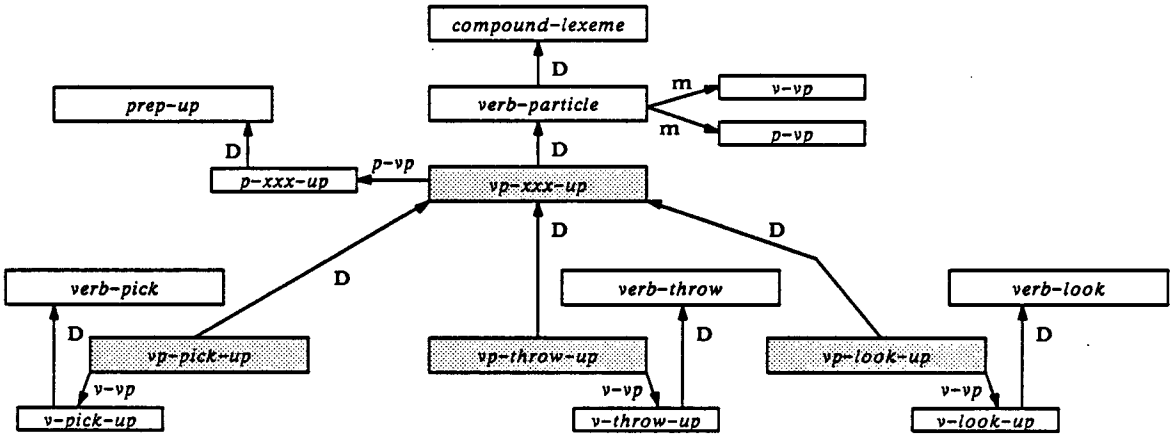

Figure 1: The compound lexeme *verb-particle-xxx-up*

unique ability to represent referential and metaphorical mappings among categories (see descriptions of *ref* and *view* below). The primitive semantic connections in an Ace hierarchy include the following:

dominate — defines an *isa* link between two categories. This relation is labeled with a "D" in the figures.
`(dominate action running)` means that *running* is an *action*—i.e., *action* dominates *running*.

manifest — defines a constituent of a category. Unless a *role-play* applies (see below), this relation is labeled "m" in the figures.
`(manifest action actor)` means that an *action* has an *actor* associated with it. This is analogous to a slot in other knowledge representations.

role-play — establishes a relationship between a constituent (slot) of a dominating category and a constituent (slot) of a dominated category. In the figures, this relation is labeled with the appropriate role name for the constituent.

```
(dominate action running
    (role-play actor runner))
```

means that in *running*, the role of *actor* (inherited from *action*) is played by the *runner*.

ref — defines a mapping between an entity in the *linguistic hierarchy* and an entity in the *conceptual hierarchy*. This relation is labeled "*ref*" in the figures.
`(ref lex-run running)` means that when the lexical category *lex-run* is invoked, the concept of *running* should be invoked as well. This is the main channel through which semantic interpretation is accomplished.

view — defines a metaphorical mapping between two categories in the conceptual hierarchy.

```
(view transfer-event action
    (role-play source actor))
```

means that in certain cases, an *action* can be metaphorically viewed as a *transfer-event*, with the *actor* viewed as the *source* of the transfer.

This brief introduction to Ace will help the reader understand the descriptions of the representation and access of compound lexemes that are presented in the next two subsections.

A. Compound Lexemes

1. Word Sequences

Word sequences are phrases such as "by and large" and "let alone" that must be treated as compound words because there is little hope in trying to determine their meaning by examining their components. Internally, these word sequences may or may not be grammatical (e.g., "kick the bucket" is internally grammatical, but "by and large" is not).

Because type of compound lexeme is very specific, a separate category exists for each word sequence under the general category of *word-sequence*. Lexical constraints are placed on the different constituents of the *word-sequence* relation by dominating them by the appropriate simple lexeme. This is one method that can be used to establish constraints on compound lexemes, and it is used throughout the compound lexeme hierarchy.

2. Lexical Relations

Lexical relations include compound lexical entities such as "pick up" and "sell out" that can appear in a variety of surface forms, but have some general relationship among their simple lexeme constituents. Compound lexemes such as verb-particles ("pick up"), verb-prepositions ("take to"), and helper-verbs ("get going") all fall into the category of *lexical relations*. In contrast to the individual subcategories of word sequences, there are many entries that fall underneath each individual subcategory of lexical relations. Most of the entries under these subcategories, however, share constituents with other entries, which makes generalizations possible. For example, Figure 1 shows how all *verb-particles* that have *up* as the par-

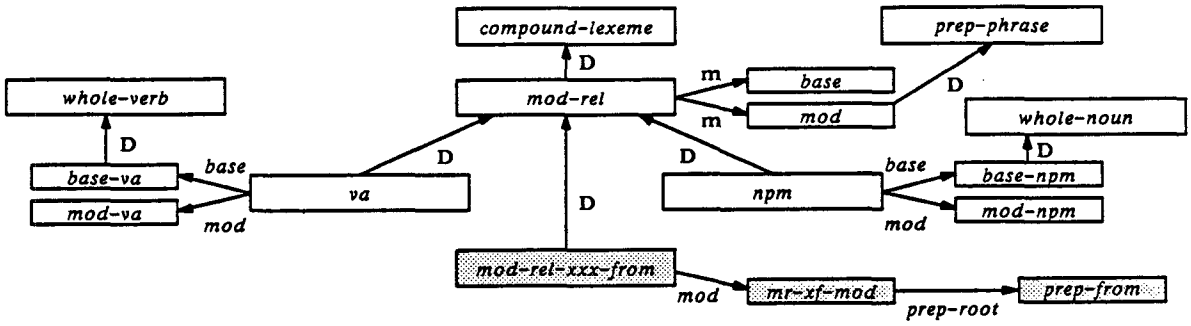

Figure 2: The modifying-relation compound-lexeme hierarchy.

ticle (e.g., "pick up", "throw up", "look up") are represented.

This generalization in representing seemingly specific phrases is what makes FLUSH extensible. If a new verb-particle with *up* as the particle is added to the system (e.g., "hang up"), it inherits everything except the verb from the structure above it—that is, the general properties of *verb-particle* relations are inherited (such as the transposition of the particle with the object "it"), and the specific properties of *verb-particles* having the preposition "up" (the constraint on the preposition itself, and possibly some default semantics for the particle) are inherited.

3. Linguistic Relations

Linguistic relations are invoked according to constraints on their constituents, where the constituents may be simple lexemes, compound lexemes, or syntactic structures. An example occurs in the sentence "John was sold a book by Mary" where the object of the preposition is the main actor of the event described by the verb. This condition occurs only when the whole verb is in the passive form (constraint 1) and the preposition in the modifying prepositional phrase is *by* (constraint 2).

Linguistic relations are difficult to represent for two reasons: their constituents are not always simple lexemes and usually there are additional constraints on each constituent. It has been found, however, that a great deal of generality can be extracted from most of the linguistic relations to make accessing them easier. The best example of a linguistic relation is the class of the modifying prepositional phrases. In some instances, prepositional phrases modify noun phrases and verb phrases in almost the same way (e.g., "The man *on the hill* is a skier" and "We had a picnic *on the hill*"). In other cases prepositional phrases modify noun phrases and verb phrases in completely different ways (e.g., "The man *by the car* is my father." and "The boy was hit *by the car*."). FLUSH is able to represent both types of linguistic relation by having more than one level of generic representation. Figure 2 shows the general modifying relation (*mod-rel*) at the first level below *compound-lexeme*. Prepositional phrases that are homogeneous across noun phrases and verb phrases are represented underneath this category. Below *mod-rel* in Figure 2 are the verb-adjunct (*va*) and noun-post-modifier (*npm*) categories, which make up the second level of generic representation. Prepositional phrases that modify verb phrases and noun phrases differently are represented underneath these categories.

As an example, in Figure 2 the *mod-rel* category has the more specific modifying relation *mod-rel-xxx-from* underneath it, which is a modifying relation where the preposition in the modifier is *prep-from*. Example uses of this prepositional phrase are found in the sentences: "The man arrived from New York" and "The woman from Boston is my aunt".

4. Linguistic/Conceptual Relations

These are expressions that cannot be easily handled as exclusively linguistic constructs, such as "giving permission", "getting permission", and "having permission". These expressions can be represented as an *abstract possession concept where the possessed is "noun-permission"*, thus combining a class of concepts with a lexical category.

These compound lexemes have the unique characteristic of allowing linguistic relations to have explicit conceptual constraints. In the phrase "give a hug" there is an abstract relationship between the concept of *giving* and the simple lexeme *noun-hug* that implies the concept of *hugging*. Figure 3 shows the representation of this linguistic/conceptual relation. This kind of compound lexeme is invoked by the semantic interpreter, rather than by the parser, during a process called *concretion*—making concepts more concrete. The scope of this paper does not permit a discussion of concretion, but refer to [Jacobs, 1986b] for more information.

The descriptions in this section illustrate how FLUSH is able to represent a wide range of lexical phenomena in a hierarchical and uniform manner. The four classes of compound lexemes that are described encompass many of the usually problematic expressions in natural language, yet they are represented in a way that supports extension and adaptation. The next section describes how these representations are accessed by FLUSH.

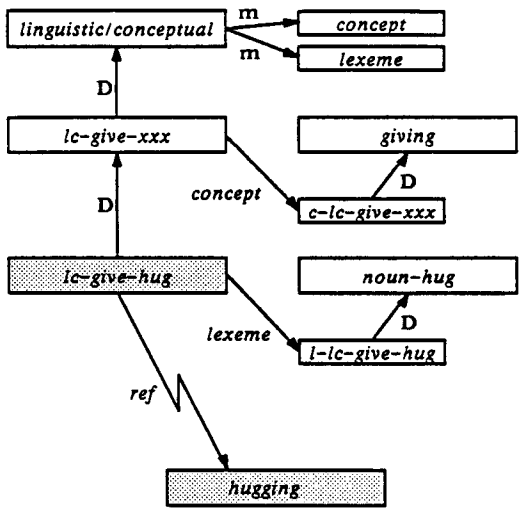

Figure 3: The linguistic/conceptual relation *lcr-give-hug*.

B. Access

Although the compound lexeme representations illustrated in the previous section differ, FLUSH is able to employ a fairly flexible algorithm for accessing them. When the parser encounters a relation that may constitute a compound lexeme, it passes the name of the relation and the constituents that fill the appropriate roles to FLUSH. If FLUSH finds a compound lexeme that satisfies the constraints, it passes the lexeme back to the parser.

For example, if TRUMP is working on the sentence "John picked up the book", it encounters a possible verb-particle relationship between the verb "picked" and the preposition "up". When this relationship is apparent to the parser, FLUSH is called with the *verb-part* relation with the constituents of *pt-verb-pick* as the verb and *prep-up* as the particle:

```
(find-compound verb-part
               (v-verb-part pt-verb-pick)
               (p-verb-part prep-up))
```

In this example, the compound lexeme *verb-part-pick-up* is found by FLUSH and is returned to the parser. If instead the sentence is "John meditated up the hill", the parser takes the same action, but no compound lexeme is found by FLUSH because "meditated up" has no special meaning.

FLUSH uses a two step procedure to locate specific compound lexemes. First, entries below the given relation in the hierarchy are checked to see if any of them satisfy the given constraints. If a compound lexeme exists, it is usually found during this step. There are some cases, however, in which the desired compound lexeme exists as a subcategory of an ancestor of the given relation. This situation was seen in the description of the modifying relation (*mod-rel*), verb-adjunct (*va*), and noun-post-modifier (*npm*) in the previous section (see Figure 2). In this case, a second step in the search process looks at the sibling categories. This process continues until either the top of the *compound-lexeme* hierarchy is reached (which happens immediately for most relations) or until a suitable compound lexeme is found.

The process of finding a compound lexeme below the given relation is a matching problem. In response to the example call to `find-compound` above, the lexicon proceeds to look at the defined categories underneath *verb-part*, which include *verb-part-xxx-up, verb-part-xxx-out, verb-part-xxx-off*, etc., to see which one(s) satisfies the constraints. *verb-part-xxx-up* is found as a possibility, resulting in the same function being called recursively with the remaining constraints to find an appropriate category below it:

```
(find-compound verb-part-xxx-up
               (v-verb-part pt-verb-pick))
```

This process is repeated until one of two conditions occurs: either the given constraints are exhausted, in which case a category that satisfies all of them has been found; or there are no more categories to search but there are still constraints left, in which case no match has been found and it may be appropriate to search the ancestors' subcategories. In this example, the *verb-part-pick-up* category is found and returned on the second recursion, therefore, there is no need to search the hierarchy at a higher level.

If instead the parser is working in the sentence "The man arrived from New York", it encounters a possible verb-adjunct (*va*) relation between the verb "arrived" and the prepositional phrase "from New York". The lexicon is called with the *va* relation, but the first step in the search process (i.e., looking below the given relation) does not yield a compound lexeme because *mod-rel-xxx-from* is defined in terms of the *mod-rel* relation rather than in terms of the *va* relation (see Figure 2). So even though the relation that the parser encounters in the pattern is a verb-adjunct relation, the lexicon is flexible enough that it can apply more general knowledge to the retrieval problem.

The meanings of compound lexemes are represented and accessed using a reference pointer that links the linguistic category to a conceptual structure. Some of the conceptual reference pointers for compound lexemes are more complicated than simple lexical access because often there are several components that need to be mapped, but they are still defined in terms of the *ref* association [Jacobs, 1986a]. The example form below defines a reference from the compound lexeme *mod-rel-xxx-from* to the *transfer-event* concept:

```
(ref transfer-event <-> mod-rel-xxx-from
     (source <-> m-mod-rel-xxx-from))
```

This reference establishes that the modifying relation *mod-rel-xxx-from* should invoke the *transfer-event* concept, and the modifier part of *mod-rel-xxx-from*, namely *m-mod-rel-xxx-from*, should fill the role of *source* in this *transfer-event*. In the sentence "The man arrived from New York",

the prepositional phrase "from New York" invokes *mod-rel-xxx-from*. In turn, the *transfer-event* concept is invoked with "New York" as the source of the transfer.

The explanations above illustrate that FLUSH is capable of representing and accessing most of the different types of lexical knowledge that natural language processing systems need to have. They also show how FLUSH can do most of it in a general manner, making extensions fairly straightforward. FLUSH is equipped also with a mechanism for automatic acquisition of new lexemes, described in [Besemer, 1986]. The discussion that follows concentrates on the application of the hierarchical lexicon to semantic interpretation in TRUMP.

III. Semantic Interpretation using FLUSH

Section II. described the organization of the FLUSH lexicon, distinguishing several classes of lexical knowledge and showing the use of a hierarchical knowledge representation in representing examples of each class. One goal of this hierarchical organization is parsimony: because categories of compound lexemes inherit their constraints from more general categories, the number of linguistic constraints encoded explicitly can be reduced. A second function of the hierarchical representation, perhaps more important, is to facilitate the interpretation of the meaning of a compound lexeme.

Semantic interpretation is facilitated by each of the classes of compound lexemes discussed in section II.. The simple example of word sequences allows the semantic interpreter to set aside the meanings of the individual words to interpret phrases such as "by and large" and "kick the bucket" correctly. Lexical relations, such as "pick up" and "working directory", permit the association of specialized meanings as well as the contribution of certain flexible lexical classes to the meaning of a phrase. For example, the phrase "branch manager" is interpreted using knowledge that it belongs to a lexical category common with "lab manager" and "program manager". Linguistic relations such as *mod-rel-xxx-from* permit general lexical knowledge to apply to the filling of conceptual roles. Linguistic/conceptual relations such as *lcr-give-hug* permit the specialized interpretation of expressions such as "give a hug" in a broad range of surface forms.

The following examples illustrate the operation of the TRUMP semantic interpreter and its use of the FLUSH lexicon.

Example 1:

Send the *laser printer characteristics* to the *branch manager*.

Processing the above sentence stimulates a steady flow of information between TRUMP'S parser and semantic interpreter and the FLUSH lexical access mechanism. The lexical analyzer recognizes "laser", "printer" and "characteristics" as nouns, but the search for compound lexical entries is activated only as the parser recognizes that the nouns form a compound. The specific entry for "laser printer" in the FLUSH lexicon, returned using the compound access method described in the previous section, provides two important pieces of information to TRUMP: First, it gives the semantic interpreter the correct meaning of the phrase, permitting TRUMP to forbear consideration of interpretations such as "a printer that prints lasers". Second, it enables the parser to favor the grouping [[laser printer] characteristics] over [laser [printer characteristics]] and thus come up with a viable meaning for the entire phrase.

The handling of the relationship between "characteristics" and "laser printer" makes use of the middle-level category *cn-xxx-characteristic*, much like the *verb-particle-xxx-up* category described in section II. The *cn-xxx-characteristic* category, representing compound nominals whose second noun is "characteristic", is associated with its meaning via a REF link in the following way:

```
(ref characteristic <-> cn-xxx-characteristic
     (manifester <-> 1n-cn-xxx-characteristic))
```

The above association, in which *1n-cn-xxx-characteristic* denotes the first noun of a particular nominal compound, suggests the interpretation "characteristics of the laser printer". The treatment of this association as a middle-level node in the hierarchical lexicon, rather than as an independent lexical entry, has two features: First, it is often overridden by a more specific entry, as in "performance characteristics". Second, it may cooperate with more specific lexical or conceptual information. For example, the conceptual role *manifester* is a general one that, when applied to a more specific category, can lead to a specific interpretation without requiring a separate conceptual entry. This would happen with "laser printer performance characteristics".

The phrase "branch manager", like "laser printer characteristics", is interpreted using an intermediate entry *cn-xxx-manager*. While FLUSH has the capability, like PHRAN [Wilensky and Arens, 1980b], to constrain this category with the semantic constraint that the first noun must describe a bureaucratic unit, it is at present left to the semantic interpreter to determine whether the preceding noun can play such an organizational role.

Example 2:

Cancel the transmission *to the printer*.

In this example, the lexical access mechanism must determine that "to the printer" invoked the *mod-rel-xxx-to* linguistic relation, which can be attached either to the verb "cancel" or the nominal "transmission". The semantic interpreter then finds the following association:

```
(ref transfer-event <-> mod-rel-xxx-to
```

```
(destination <-> m_mod-rel-xxx-to))
```

The REF association above indicates that the object of the preposition "to" is related to the *destination* role of some generalized transfer event. Since "cancel" describes no such event, but "transmission" does, TRUMP correctly interprets "printer" as being the destination of the transmission. This allows the semantic interpreter to handle this example much in the same way as it would handle "Transmit the job *to the printer*", because the *mod-rel* relation class includes both postnominal modifiers and adverbial prepositional phrases. As in the previous example, the semantic interpreter can make use of the interaction between this general interpretation rule and more specific knowledge; for example, "the sale of the the book *to Mary*" invokes the same *mod-rel-xxx-to* relation, but the role of Mary is determined to be *customer* because that role is the conceptual specialization of the *destination* of a transfer. The process of correctly determining a conceptual role using linguistic relations is described in [Jacobs, 1987].

Example 3:

How many arguments does the command *take*?

There are two major differences between this example and the previous two: First, the lexicon is driven by information passed from TRUMP'S semantic interpreter, not only from the parser. In the previous example, the parser recognizes a potential relationship between a verb or nominal and a prepositional phrase. In this case, the semantic interpreter must determine if the *conceptual* relationship between the concept of *taking* and the term "arguments" invokes any special lexical knowledge. Second, the interpretation of "take arguments" is not a specialization of an abstract concept such as *transfer-event*, but rather is a result of a metaphorical *view* mapping from this concept to the concept of *command-execution*.

The interpretation of this sentence thus proceeds as follows: At the completion of the syntactic parse, the semantic interpreter produces an instantiation of the concept *taking* with the object *arguments*. The lexical access system of FLUSH, using the same discrimination process that determines a specialized linguistic relation, identifies *lcr-transfer-arguments* as a linguistic/conceptual relation invoked by the concept of a transfer with the lexical term "argument" attached to the conceptual *object* role. The same linguistic/conceptual relation is invoked by "giving arguments" or "getting arguments". The semantic interpreter continues by determining the metaphorical mapping between the *transfer-event* concept and the *command-execution* concept, a mapping that derives from the same conceptual relationships as other similar metaphors such as "The recipe takes three cups of sugar." In this way the amount of specialized information used for "take arguments" is kept to a minimum; effectively, FLUSH in this case is merely recognizing a linguistic/conceptual trigger for a general metaphor.

This section has described the application of the FLUSH lexicon to the process of semantic interpretation in the TRUMP system. The examples illustrate some characteristics of the flexible lexicon design that differ from other phrasal systems: (1) There are a broad range of categories to which specialized information may be associated. The treatment of "branch manager" and "transmission to" illustrates the use of compound lexical knowledge at a more abstract level than other programs such as PHRAN. (2) The hierarchical lexicon reduces the number of phrasal entries that would be required in a more rigid system. Expressions such as "take arguments" and "get arguments" share a common entry. (3) The quantity of information in each phrasal entry is minimized. Linguistic constraints are often inherited from general categories, and the amount of semantic information required for a specialized entry is controlled by the method of determining an appropriate conceptual role. The "take arguments" expression thus does not require explicit representation of the relationships between linguistic and conceptual roles.

IV. Conclusion

FLUSH is a flexible lexicon designed to represent linguistic constructs for natural language processing in an extensible manner. The hierarchical organization of FLUSH, along with the provision for a number of types of phrasal constructs, makes it easy to use knowledge at various levels in the lexical hierarchy. This design has the advantage of handling specialized linguistic constructs without being too rigid to deal with the range of forms in which these constructs may appear, and facilitates the addition of new constructs to the lexicon. FLUSH permits the correct semantic interpretation of a broad range of expressions without excessive knowledge at the level of specific phrases.

References

[Becker, 1975] J. Becker. The phrasal lexicon. In *Theoretical Issues in Natural Language Processing*, Cambridge, Massachusetts, 1975.

[Besemer, 1986] D. Besemer. *FLUSH: Beyond the Phrasal Lexicon.* Technical Report 086CRD181, General Electric Corporate Research and Development, 1986.

[Bobrow and Winograd, 1977] D. Bobrow and T. Winograd. An overview of KRL, a knowledge representation language. *Cognitive Science*, 1(1), 1977.

[Brachman and Schmolze, 1985] R. Brachman and J. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2), 1985.

[Dyer and Zernik, 1986] M. Dyer and U. Zernik. Encoding and acquiring meanings for figurative phrases. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, New York, 1986.

[Halliday, 1978] M. A. K. Halliday. *Language as Social Semiotic.* University Park Press, Baltimore, Maryland, 1978.

[Jacobs, 1985a] P. Jacobs. *A Knowledge-Based Approach to Language Production.* PhD thesis, University of California, Berkeley, 1985. Computer Science Division Report UCB/CSD86/254.

[Jacobs, 1985b] P. Jacobs. PHRED: a generator for natural language interfaces. *Computational Linguistics,* 11(4), 1985.

[Jacobs, 1986a] P. Jacobs. Knowledge structures for natural language generation. In *Proceedings of the Eleventh International Conference on Computational Linguistics,* Bonn, Germany, 1986.

[Jacobs, 1986b] P. Jacobs. Language analysis in not-so-limited domains. In *Proceedings of the Fall Joint Computer Conference,* Dallas, Texas, 1986.

[Jacobs, 1987] P. Jacobs. A knowledge framework for natural language analysis. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence,* Milan, Italy, 1987.

[Jacobs and Rau, 1984] P. Jacobs and L. Rau. Ace: associating language with meaning. In *Proceedings of the Sixth European Conference on Artificial Intelligence,* Pisa, Italy, 1984.

[Lockwood, 1972] D. Lockwood. *Introduction to Stratificational Linguistics.* Harcourt, Brace, and Jovanovich, New York, 1972.

[Sondheimer *et al.*, 1984] N. Sondheimer, R. Weischedel, and R. Bobrow. Semantic interpretation using KL-ONE. In *Proceedings of the Tenth International Conference on Computational Linguistics,* Palo Alto, 1984.

[Steinacker and Buchberger, 1983] I. Steinacker and E. Buchberger. Relating syntax and semantics: the syntactico-semantic lexicon of the system VIE-LANG. In *Proceedings of the First European Meeting of the ACL,* Pisa, Italy, 1983.

[Wilensky, 1986] R. Wilensky. Knowledge representation - a critique and a proposal. In J. Kolodner and C. Riesbeck, editors, *Experience, Memory, and Reasoning,* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1986.

[Wilensky and Arens, 1980a] R. Wilensky and Y. Arens. *PHRAN-A Knowledge-based Approach to Natural Language Analysis.* Electronics Research Laboratory Memorandum UCB/ERL M80/34, University of California, Berkeley, 1980.

[Wilensky and Arens, 1980b] R. Wilensky and Y. Arens. PHRAN-a knowledge-based natural language understander. In *Proceedings of the 18th Annual Meeting of the Association for Computational Linguistics,* Philadelphia, 1980.