In *Mind & Machines* (4) 59-73. A special issue dedicated to the semantic representation used by running natural language understanding systems, edited by Stuart Shapiro

'KRISP'
a representation for the semantic interpretation of texts

David D. McDonald[1]
November 1992

Abstract

KRISP is a representation system and set of interpretation protocols that is used in the Sparser natural language understanding system to embody the meaning of texts and their pragmatic context. It is based on a denotational notion of semantic interpretation, where the phrases of a text are directly projected onto a largely pre-existing set of individuals and categories in a model, rather than first going through a level of symbolic representation such as a logical form. Semantic interpretation in Sparser is fully compositional, with each rule of syntactic form incorporating either a direct reference to the corresponding semantic object or an instruction for its access or construction in terms of the denotations of the rule's immediate constituents.

KRISP defines a small set of semantic object types, grounded in the lambda calculus, and supports the principle of uniqueness, whereby there is only one representation of every distinct entity rather than several sets of descriptions requiring unification. It supplies first class objects to represent partially-saturated relationships, such as seen in conjunction reductions and even in simple terms like *foreign* (e.g. is it relative to Japan or the U.S.?), which facilitates discourse history searches for the missing information. KRISP supports online subtyping of categories when new descriptions are seen, and separates the manifestation of new combinations of categories in a given individual from the reification of such combinations as distinct domain types.

KRISP is being used to develop a core set of concepts for such things as names, amounts, time, and modality, which are part of a few larger models for domains including Who's News and joint ventures. It is targeted at the task of information extraction, emphasizing the need to relate the entities mentioned in new texts to a large set of pre-defined entities and those read about in earlier articles or earlier in the same article.

1. Introduction

To understand a text, a person must find a relationship between the elements of the text and her model of the entities in the world. The process of finding this relationship engages a person's knowledge of their language, their semantic conception of the world, and their pragmatic appreciation of the situation they are in. All of this, including much of their syntactic knowledge of language, is mediated by the representational system that they employ; indeed, it is arguable that it is a model couched in this representation that is the actual target of the understanding process. Given this assumption that the proper domain of interpretation for texts by people is a mental model rather than the real world, we have to assume that the relationship of this model of the world in the speaker's mind to the actual world must be mediated by other sorts of processes. One might study these by research on real, situated robots, but probably not just with research on the semantics of natural language.

To study the possible nature of this representational system using the tools of artificial intelligence, we presuppose that the actual system that people use can be emulated by a computational artifact—the subject of this paper. Then, in order to provide a context in which an implemented language understanding process can function, we construct a (small) world model and populate it with a set of individuals, categories, and relationships appropriate to the subject matter of the texts that are to serve as the input.

We are not, for present purposes, going to go further and incorporate into this architecture an artificial speaker/hearer who would appreciate the consequences of what is understood and then act accordingly, and we will not even consider here any of the processes for generating new texts from this model (summaries, explanations, etc.), though the representational system and world model are specifically designed for facilitate text generation as summarized below.[2] We will assume instead that the properties of the representation we are proposing and its utility can be sufficiently demonstrated by considering only the process of recovering the projection of a text onto the speaker's world model, and the limited, text-based reasoning procedures this entails. In any event, this is the only portion of the cognitive architecture that we have implemented at the time this paper is written.

KRISP (pronounced "crisp") is the result of a long-standing body of work on efficient representations for language generation. One of the goals of this work is that the representation should facilitate a bi-directional statement of the form-meaning relationship, where the same rules in the same notation can be used both for parsing and generation (perhaps with the mediation of a compiler so that the runtime structure of the rules is efficient); see McDonald 1993. The first substantial development and implementation of KRISP is as the target semantic representation of the language comprehension system, 'Sparser' (McDonald 1992a). Sparser[3] is a deterministic, bottom-up chart parser that employs a number of different parsing techniques, including finite-state recognizers, context-free and context-sensitive phrase structure rules, and heuristic search based on patterns of closed class words and the semantic type of isolated constituents. ("KRISP" is an acronym for "Knowledge Representation In Sparser", which, if prosaic, does reflect the fact that the two systems were designed to operate with tight integration.)

KRISP falls into the intellectual tradition of the 'KL-One' family of languages for knowledge representation (see, e.g., Brachman & Schmolze 1985, MacGregor 1991). Like them, it is based on the notion of structured conceptual objects, with a strict discipline on how information may be inherited through a taxonomic lattice. It differs from them in de-emphasizing term-classification as the basis of what concepts there are and how their relationships are distributed, favoring instead techniques that allow concepts to be built up compositionally from the phrases found in actual texts. It also puts more emphasis on the representation of particular individuals and partially saturated relations, along with a {{new}} conception of how 'role' relationships should be construed—{{one that permits variations in perspective to be factored out from a common body of information}} and gives the frame–slot–value relation its own type as a first-class entity.

This paper will move from motivations to definitions to examples. The first goal is to explain why one should be developing a yet another new representation after now more than three decades of work in the field. To do this, we first set the scene by appreciating what any representation must be like in order to support the efficient semantic interpretation of texts, and then move to the principles that underlie KRISP's design, gleaned from our experience in trying to adapt older representational formalisms to the task of natural language generation.

Following this introduction we will lay out KRISP's object types and the rationales behind them (§2.1). We then get to the heart of the matter by looking at the 'protocols' for KRISP's use in semantic interpretation—its choice of denotations for the various classes of syntactic constructions and their composition (§3). These sections will include several examples of semantic problems in actual texts and how the machinery that KRISP supplies simplifies their solution. We will close with a summary of where this ongoing research stands.

1.1 Fitting the representation to the task

Natural language understanding is a very particular task; its representational system should be equally particular if the task is to be done efficiently. Representations developed for other purposes: robotic motion planning, temporal reasoning, even plan recognition, are unlikely to provide the needed 'notational efficacy' as defined by Woods (1987). Woods' desideratum that the form and primitive operations of a representational system be designed to suit the processes that will use them is often overlooked in representation research since much of this research focuses instead on just the expressive power of systems and their logical properties in facilitating general reasoning.

Language understanding, however, is not general reasoning, especially since what we are ultimately studying is the psychological process engaged in by people. Rather it is an encapsulated, fast, mandatory, and unavoidable automatic process that is in the same class, psychologically, as human vision or walking (as contrasted with 'central' processes such as playing chess, deciding who to vote for, or teaching a course); see, e.g., Fodor 1983, Marslen-Wilson & Tyler 1987.

Given this specificity, we should look for a representation whose formal properties as a computational system are a close fit to the requirements of the process of understanding a text—of going from the presentation of a text's form as a time-sequence of sounds or orthographic characters to the recovery or construction of mental objects representing the individuals, categories and relations it conveys. We thus expect the representation to be sensitive to the way the information in a text is aggregated into chunks, the patterns in which it is structured by syntactic relations, and the time-course over which it becomes available.

Sensitivity to the way language conveys information carries over into the assumptions one makes about how speakers conceptualize their knowledge. Some conceptions can sustain an online mapping from text to semantic entities, supplying the counterpart of each word or syntactic phrase as soon as the parser forms it; others have to wait until a number of phrases or even entire sentences have accumulated before expressions in their notation and with their primitives can legitimately be produced.

The possibility of principled accounts of how people conceptualize their knowledge—model their world—is largely neglected in studies of representation, especially in artificial intelligence. This is not surprising since if one is building a system that reasons in isolation there is no obvious external rationale for why one set of primitives and relations should be selected over an alternative. By contrast, systems in robotics or vision have more principled models because the real-world physics or geometry of the problem provides the needed external rationales.

By the same token, when we look at accounts of the conceptualizations underlying people's use of natural language, the strongest and most principled work that we see is tied strongly to the

structure of language—something concrete and relatively accessible to our study, while the mental entities that constitute meaning in the mind are not (e.g. Talmy 1987; Jackendoff 1983).

To arrive at a principled account one needs to begin with some broad assumptions, for instance whether there is to be only a small number of primitives (emphasizing common sets of inferences) or an arbitrarily large number (emphasizing their unique qualities and leaving shared inferences to a different mechanism). Such fundamental choices then have to fit within a governing philosophy whose purpose is to establish how differences in possible conceptualizations are to be adjudicated. This in turn has to be coupled with a set of protocols that spell out the particulars of how the concepts are to actually be mapped to texts—semantic interpretation, given all of the various phrasal types and grammatical relations that the language supports. If the classification of these linguistic types is well suited to the task and its processing algorithms, then the choice of conceptualizations—the deployment of the representational system—has a good chance of being both coherent and effective.

1.2 Principles from Language Generation

Natural language generation systems are usually designed as interface components. As such, the generator[4] is linked to some knowledge based system—the *speaker* it is producing texts for. This speaker maintains a *model* of its world couched in some representation. The model is a body of information—what the speaker knows, and it is organized into *units* of various types as dictated by the representational system. These units are passed to or selected by the generator and form the basis of the texts it produces.

Some of the units represent individuals. For example a model of executives changing jobs such as that required for thinking about a news source like the Wall Street Journal's "Who's News" column will initially include certain well-known people and companies, a set of particular titles, a calendar of particular times, etc. and it will learn about a great many other individuals in the course of reading articles from the column.

Other units will represent the concepts or, as I would prefer to say, the *categories* that define the speaker's very notions of 'people', 'companies', 'times', etc. and provide the basis by which individuals—the members of the classes defined by these categories—are recognized and manipulated. The model will include categories that represent generalizing abstractions over the concrete kinds and relations directly exhibited in the language. These capture common patterns among units, such as the notion of an agent or the fact that organizations of all sorts define a set of roles taken on by the people in them, and that these roles are given conventional names ('vice-president', 'student') and participate in a common set of syntactic constructions.

The relationship between such units and the generator's knowledge of the kinds of texts it can produce is usually established by a set of mapping tables that act as links to the linguistic resources appropriate to their realization: specific words, subcategorization frames, subsequent reference criteria, etc., possibly involving several intermediate levels of representation. The generator's task is to utilize these tables to take a set of units selected from the speaker's domain model—the speaker's representation of the information it wants to be communicated—and produce a coherently organized, cohesive and grammatical text that realizes that information.

Historically, the difficulty has come in the fact that in practically every generation project, the representational system that the speaker used was developed for a purpose other than generation. Experience has shown that in the preponderance of cases these representations had an ontologyl and set of conceptualizations that were at odds with what generators require to work efficiently. It is probably not an accident that the generators that have historically produced the most natural and fluent texts were also the ones that developed their own systems of knowledge representation.

As a consequence, generator designers have had to go to great lengths to reformulate the units of the speaker's model into something that (1) was a closer fit to the way information is deployed in a text, and (2) provided more options for reorganizing how the information was

distributed so that it could be accommodated to the differing contexts encountered in fluent discourse. There would be no problem if every unit presented to the generator for realization was isomorphic in structure to the text it was intended to produce; but that is rarely the case. Even in the instances where there is one plausible text with an isomorphic structure, most representations do not have the flexibility to reconfigure their units so that a direct realization is also possible for the variations on that text that will be dictated by varying discourse contexts.

Rather than catalog a set of deficits with particular systems and representations, we will present the conclusions of our experience as a set of principles—desiderata for a representational system that is to support efficient generation.

1. **The speaker's knowledge—its model—should be set up as an interconnected set of relatively small units.**

By making the amount of information contained within the primitive units small, seldom larger than a single word (or fixed-phrase) or a simple one-argument relation, we maximize the possibilities for rearranging the assemblage of units selected for an utterance in order to best fit the discourse context. In different contexts what the audience already knows will vary, and consequently what should be omitted from the utterance as redundant or should be given special salience because it is unusual must vary as well. To give a simple example,[5] consider this text, excerpted from an newswire story on a joint venture.

> "*The new firm will be 50 percent owned by Takarabune, 45 percent by Merciries and the rest by a Japanese advertisement concern, Cybac Co., the company said.*"

We all know, as speakers of English, that "45 percent" refers to a '45% ownership of the new firm', yet that information is not given explicitly but must be inferred from context, just as a generator must appreciate the redundancy and the availability of that reduced construction in producing the text in the first place. If the units are large (e.g. the equivalent of simple clauses) the generator will have to either be satisfied with disfluent texts or construct its own decompositions of what the speaker supplies.

2. **The structure and content of every unit of information is immediately accessible.**

This is an admonition for the use of typed structured objects as the basis of the representation, rather than expressions whose content and inter-relationships can only be determined by scanning them and remembering what was found (such as the typical representation of a logical form). It continues the theme of facilitating decomposition and rearrangement of an utterance's speaker-internal representation because it makes it possible to retain all of the options about how a unit will be realized while its position in the text is still being determined.

The phrase "45 percent" above should be the realization of a (complex) unit meaning 'a 45 percent portion of the ownership of Taiwan Takarabune Confectionery Ltd.' (the "new firm"), and not just a simple percentage. By retaining the full information all the way until the unit's context has been established, the generator has the option of giving it other realizations such as "45% of the firm" or "Merciries has a 45% stake". (Similarly a parser, when confronted with any of those phrasings, should interpret all of them as the same full unit.)

3. **There is a first-class object type corresponding to every class of syntactic category in the language.**

The notion of a first class object comes from the study of programming languages where it refers to those objects that can be returned as the value of functions. Here the focus is on what elements of a representational system can be factored out of the context they are part of and still retain their identity. In the usual notation for the predicate calculus, the letters representing variables are not first class, since they gain their meaning only by being in construction with a particularly placed quantifier and their instances in a formula's terms. In KL-One, roles are not first class since there are no instances of the relationships that roles define apart from their links to particular concepts.

A key concern here is the representation of phrases that present only partial information. Such phrases are ubiquitous, ranging from conjunction reductions to single words: the word *overseas,* for example, has a different denotation when the article is from a Japanese news agency versus an American one. The generator needs to know, as it considers various decompositions of the information it has selected, whether the units it factors out can be realized on their own or not. By having first class objects to represent relationships that are only partially saturated and linking them to the linguistic constructions for partial information, a generator can state the constraints on possible decompositions easily.

When this principle is considered from the perspective of parsing, it is the problem of being confronted with partial information and needing find a denotation for the phrase. If the representation provides a first class object to represent the state of knowing only part of what a phrase refers to, then the manipulation of that object to recover the rest of the information from context or through defaults is markedly simpler to carry out.

4. All domain entities have a unique representation in the model.

This is the *uniqueness principle* that plays a central role in the design of the SNePS representational system (e.g. Maida & Shapiro 1982). Its implications are wide-spread. Consider, for example, the paragraphs below, excerpted from a Wall Street Journal "Who's News" article from February 14, 1991.

> "*After almost nine tumultuous years, George L. Ball resigned yesterday as chairman and chief executive officer of Prudential-Bache Securities Inc., the nation's fourth-largest securities firm.*
>
> *Mr. Ball's departure signaled that Prudential Insurance Co. of America, the brokerage firm's parent, had run out of patience with Mr. Ball's quest to turn the unit into a Wall Street powerhouse. Amid mounting losses, Prudential had steadfastly backed the 52-year-old chairman in one failed foray after another.*"

This text has several subsequent references in it: "*Mr. Ball's departure*", "*the brokerage firm*", "*Mr. Ball* ", "*the unit* " "*Prudential* ", and "*the 52-year-old chairman*". A generator must know when it is realizing a second instance of an entity it has already mentioned, and by far the easiest way to do this is to notice that the representational entities for the two instances are the identical object, rather than attempting to match descriptive expressions.

From the parsing standpoint, this principle implies first of all that the output of the understanding process should be a set of objects rather than expressions describing them. We need this independently of other considerations since many of the entities being referred to are likely to already be known—Prudential Insurance for example. A descriptive output would miss the fact that the entity referenced in the text was one it already knew unless it went through an additional reconciliation process. The timing of this process might be of little concern when parsing isolated sentences, but extended texts must be understood incrementally (Mellish 19XX). For this reason Sparser recovers the denotations of phases online as they are parsed.

The uniqueness principle requires a parser to make the denotation of a subsequent reference be the very same representational entity that it looked up or constructed for the initial reference. This will lead to including a set of cross-indexes among entities and their relations so that, for example, the parser can look up what 'chairmen' it has already seen in this article, and finding only one conclude that the information '52 years old' is to be understood as providing additional information about Mr. Ball rather than being an index to select between various chairmen of different ages.

One other point should be made here that applies only to parsing. In generation, the process starts with complete information about what object types there are in the model and how they are interrelated. A parser on the other hand will have only the basic domain types corresponding to its core vocabulary. It will need to be able to construct new domain types compositionally as it sees new descriptions and especially new compound definite references, allowing it to construct

a type for, e.g., *Japanese companies* on the fly rather than needing to wait for human intervention.

To summarize this, the reason, practicalities aside, why we have developed a new representation, KRISP, rather than taken one off the shelf, is that the older representations we are familiar with are deficient on one or more counts given this set of principles. We want to use a representation for the target of parsing that is the same as we would use as the source of generation—otherwise it is impossible to develop a bi-directional treatment of linguistic rules, which is an important long-term goal. In particular, we see weaknesses in the available representations for the interpretation of phrases that express only partial information, and an insufficient number of first-class objects, especially for concept-role-value relationships. This deficit makes it difficult to preserve the uniqueness principle when dealing with reciprocal relationships where the differences in expression appear to follow from differences in perspective rather than in the information conveyed.

2. KRISP's representational system

As a representational system, KRISP embodies a theory of how information is structured in a natural language text. This is a hypothesis about the kinds of entities that texts denote and the semantic relationships by which the linguistic structure of a text maps to a model of the information it contains.[6] There are, of course, two different levels involved: what we might call the 'domain' level, where we are representing both the particular instances and the general kinds of people, events, moments, amounts, etc. in profusion that a text can be talking about; and the 'epistemological' level (following Brachman 1979), where we spell out the kinds of representational objects by which we are to capture the information a text supplies at the domain level.

KRISP defines a set of object types at the epistemological level along with a set of protocols that govern how the syntactic phrases of a text are to be interpreted in terms of them. This provides a framework to govern how models of the world at the domain level are to be designed, while leaving open the substantive issues of what distinctions in kind or subtleties of relationship are implied by the particular usages one finds in texts. In the next section we will introduce these object types, and look at their corresponding structures in the lambda calculus. Their rules of formation will be taken up in §3 since they are the rules of semantic interpretation of texts and most objects are introduced into the model and gain further properties through the interpretation of texts rather than by, say, the instantiation of prototype concepts.

2.1 Ontology

At its lowest level, a model in KRISP consists of a set of objects—the 'units' of information that the model contains—that are linked together by pointers. The units are first class objects; the pointers are not. Units are structured, with named fields containing the pointers from them to other units. Formally, pointers may be viewed as access functions taking a unit as their single argument and returning the unit or list of units to which they point.

The fields that a unit has are determined by its type. KRISP defines four primary types of units in its epistemological level: individual, binding, variable, and category, as well as two types of units that are blends of the primary types: derived categories and partially saturated relations. Everythis in a KRISP model is of one (and only one) of these types.

An ***individual*** is roughly comparable to Montague's 'e' type, KL-One's 'nexuses', or Classic's 'individuals'.[7] Individuals are used to represent particular things in the world, concrete or abstract, rather than kinds or generics. They are the typical denotations of the maximal-projections of the major syntactic categories, i.e. noun phrases and most clauses. An individual

has a *domain type* consisting of one or more categories (e.g. transition event, person, retirement, title, unicorn), and in accordance with its domain type it can enter into bindings with (stand in relation to) other individuals ('George Ball is the one who retired', 'the positions (he) retired from were chairman and CEO').

Relationships between individuals (and other types of entities) are represented by units of type ***binding***. Bindings are three-tuples consisting of an individual, a variable, and another individual. Bindings are sanctioned by the variables defined by categories. We will talk in terms of 'binding an individual to a variable'

For example 'George Ball is the one who retired' would be denoted by a binding involving (1) the individual that denoted this instance of the category retirement (i.e. it would also include the position he left, the time, etc.; see §3.1 below), (2) a variable defined by that category, say with the name 'agent', and (3) the individual denoting Mr. Ball.

These tuples may be thought of as directed, labeled links going from the first individual via the variable (the label) to the second. Broadly speaking, bindings are the frame–slot–value structures of standard frame systems, with the difference that in KRISP we are always dealing with individuals rather than kinds. In KL-One the closest analogy would be an individual concept, one of its roles (the analog of the variable) and a second individual concept, the slot's value.

The key difference from these alternative representations is that in KRISP a binding is a first-class object with an existence independent of and separable from the individuals that comprise it. This was not the case in KL-One (except at its seldom-used 'meta-level'), and is rarely the case in the implementation of frame systems; for them these tuples are only implicit in the structures defined by concepts or frames. By contrast, while KRISP's bindings point to and are pointed to by the units that comprise them, they are distinct units and provide the denotations for a number of linguistic constructions, including most copular clauses and the grammatical relationships that tie phrasal heads to their complements and adjuncts.

The relationships into which an individual can enter, and with them the variables that define the domain types of the bindings that represent those relationships, are determined by the ***categories*** that define the individual's type. An individual can and usually will have more than one category in its domain type. A category corresponds to a predicate; the airity of the predicate establishing the number of ***variables*** associated with it. Variables are local to the category that defines them, in the sense that if we were to notate them using names ('x', 'agent', 'members') we could substitute other names without changing their meaning (This is 'alpha reduction' in the lambda calculus). Categories represent kinds. They are the denotations of, among other things, most phrasal heads, i.e. common nouns and verbs.

Part of a variable's definition is its value restriction, a category or categories stipulating the types of individual that can be "bound" to that variable. Categories are organized into a taxonomic lattice on the basis of what variables they define and the restrictions on them. The notion of subsumption is applicable to this lattice, in that an individual whose type includes a relatively low category, say 'retire', will also satisfy all of the categories that 'retire' is a specialization of such as, say, 'agentive-event'. Any variable that is defined by 'agentive-event' or the intermediary categories ('transition', 'job-change', and 'leave-position') can be bound by the individual, and the variables may have their value restrictions specialized when their binding categories are specialized, such as shifting the restriction from a general (higher) category like 'agent' to a specific (lower) category such as 'person'.

All categories are primitive, which is to say that their position in the lattice does not provide necessary or sufficient criteria for their satisfaction. Indeed, the taxonomic lattice does almost no work in KRISP since the function of categories is to supply the denotations of certain classes of words.[8] Any category one might propose to define that does not have a corresponding lexical item or suitable phrase or grammatical relationship, or that could not be formed out of such by abstracting their complements would be highly suspect.

One class of exceptions to the relevance of the taxonomy is the formation of an analytic category through the composition of categories denoted by linguistic primitives (e.g. words) with generic morphological or grammatical elements such as plural, past-tense, modals, negation, etc. Additionally, many descriptive N-bar phrases (see prior footnote) want to be treated as categories since they usually do not introduce individuals in a text (unlike names or clauses) but rather categorize existing individuals or add attributive properties. The denotations of such phrases are usually partially saturated individuals rather than categories however.

Since any verb or noun could in principle undergo most of these compositions, potentially resulting in a drastic multiplication of the number of categories in the model, a mechanism is provided whereby the 'composite category' can be represented implicitly by having a list of categories in an individual's type field.Thus the individual representing a phrase like *17 vice presidents* would have two categories making up its type: 'title' and 'collection'. An intriguing question for further research is whether there is some semi-automatic way to determine which of these compositions should be reified as literal categories, possibly by looking at the pattern of terms used in definite references.

2.2 Correspondences in the lambda calculus

One type remains to be introduced, the lambda-form; but before doing that it will be best to describe the relationship of KRISP's types to their foundational computational reference system—the lambda calculus, which we will do by way of an example.

Consider the formula

```
advisor/ee(Chomsky, Ross)
```

It predicates of two logical constants, Chomsky and Ross, that they stand in the relation of thesis-advisor–to-student (which happens to be a true fact in this world for Noam Chomsky and Haj Ross). This formula corresponds to an individual in KRISP, one whose type is the category 'advisor/ee', and which has two bindings, one tieing it to an individual of type person with name *Noam Chomsky* , and another tieing it to a second person individual with the name *Haj Ross*.

The predicate 'advisor/ee' can be seen as a function of two arguments, which we can construct in the lambda calculus by naming variables corresponding to the parameter positions those arguments will take on. It is convenient to select as the names of these variables the name of type (satisfying category) they are restricted to.

```
λ professor . λ student . advisor/ee(professor, student)
```

The corresponding KRISP category has two variables in just this sense, and has the same semantics as this function—it forms an individual (the original formula) when both of its variables are bound to individuals of the appropriate type.

The formula is intended as the representation of texts like these and others:

"*Chomsky was Ross's advisor*"
"*Ross was Chomsky's advisee*"
"*Ross was Chomsky's student*"

Leaving for later the question of the past tense and the varying vocabulary in those texts, we now ask what is the denotation of the phrase *Ross's advisor*? In KRISP it is an object of type ***lambda-form***, i.e. an object corresponding to the partially reduced lambda calculus formula shown below in two different formats. In the first, the reduction has gone through; in the second, the reduction has been set up but we have yet not actually done the substitution of the constant for the variable. Note that the function application indicated by the parenthesized expression in the second instance of the formula demonstrates the semantics of a KRISP binding—bindings are the assignment of an individual to a particular variable in a specific formula.

```
 professor . advisor/ee(professor, Ross)

 professor .
     (  student.advisor/ee(professor, student)
        Ross )
```

A lambda-form is a unit that is part way between a category and an individual. In a lambda form some but not all of a category's variables are bound, and a record is kept of what variables remain open. In the course of a parse there is often a stage where the denotation of a phrase in isolation (*Ross's advisor*) is a lambda-form. Then the larger constituent of which that phrase is a part (e.g. *Ross's advisor was Chomsky*) adds information that provides the binding of the open variable(s), yielding an individual now that the relation ('advisor/ee') is fully saturated.

2.3 Notation

The definition of categories and the rules of formation that take categories into individuals are intimately caught up with the definition of a grammar for some topic, and will be discussed in the next section. Here we will simply look at the units from the example just given, showing their fields and the pointers that link them together. In the process we will illustrate how KRISP's representation captures the fact that all of the example texts encode the same information.

All objects, by warrant of their status as representational units, have fields for the mundane information that makes a representational system practical to use in a large, commercially-oriented program. There is a link to a symbol that acts as the unit's name, and one to a Lisp plist for record-keeping information such as when the unit was created and what file its definition is in. Beyond that the fields are specific to the different types.

A *category* has one germane field, its link to its variables. Below is the standard printed representation of a category. Following the Lisp conventions for structured objects, we do not actually show the links, leaving them implicit in the positional notation.

```
#<category  advisor/ee
   :variables ( #<variable advisor/ee.professor
                   :category #<advisor/ee>
                   :v/r  #<category professor>>
                #<variable advisor/ee.student
                   :category #<advisor/ee>
                   :v/r  #<category student> > ) >
```

A unit, as an object in a model, is printed within angle brackets prefixed with a sharp sign ("#<…>"). Using the sharp signs as a guide, we see that the expression above involves five units. Just following the initial bracket is the name of the units' type; when it is obvious from context what type of unit we are dealing with this may be omitted; sometimes only the units' symbolic name is used: `#<advisor/ee>`. The pointer from the category to its two variables is indicated by the field named `:variables` (this is the access function), immediately followed by the object pointed to, here a list of the two variables.

A *variable* has two fields: one holds a pointer to the category that defines it; the other a pointer to its value restriction (`:v/r`) the category (or list of categories, interpreted as an 'or') that restrict what it can be bound to. The symbolic name of a variable is the name of its category appended to the designated name of the variable, separated by a dot.

A *binding* has three fields: one points to the individual whose type includes the category that owns the variable; the second points to that variable, and the third to the individual bound to it.

```
#<binding
   :body  #<individual advisor/ee-1>
   :variable  #<variable advisor/ee.professor>
   :value  #<individual person-1> >
```

There is more convenient print form for bindings, shown below. The individual sanctioning the binding is given by its name (a generated symbol based on its first category and a number) separated from the variable by a dot; and then an equal sign and the individual being bound.

```
#<advisor/ee-1.professor = #<Chomsky>>

#<advisor/ee-1.student = #<Ross>>
```

An *individual* has three fields. The first, `:type`, points to the list of the categories that collectively define the individual's domain-type. The second, `:binds`, points to all of the binding where the individual is the one that owns the variable. The third, `:bound-in`, is the inverse case: it points to all of the bindings where the individual is the one that is bound. The examples show other short-cut notations such as the use of the value of a name variable to stand for the individual with the name (`#<Chomsky>`), and the use of italics and double quotes to indicate that the value is a 'word' one of the a system primitive types in Sparser.

```
#<individual  advisor/ee-1
   :type  ( #<category advisor/ee> )
   :binds
      ( #<advisor/ee-1.professor = #<Chomsky>>
        #<advisor/ee-1.student = #<Ross>>  )>

#<individual  person-1
   :type  ( #<category person>
            #<□ employment/position #<title professor>> )
   :binds
       ( #<person.name "Noam Chomsky"> )
   :bound-in
       ( #<advisor/ee-1.professor = #<Chomsky>> )>

#<individual  person-2
   :type  ( #<person>
            #<□ employment/position #<title student>> )
   :binds
       ( #<person.name "Haj Ross"> )
   :bound-in
       ( #<advisor/ee-1.student = #<Ross>> )>
```

It is crucial here to again point out that every object in KRISP is unique, occurring only once in the model ('the principle of uniqueness', see §1.2 above). Thus the two bindings in the binds field of advisor-1 and their repeated print-forms in the bound-in fields person-1 and person-2 are the identical objects. Similarly, the individual #<Ross> within the advisor/ee binding of person-2 and person-2 itself are the same object—a recursive reference made less obvious by the use of the two different printing styles.

The idea that this relation between Ross and Chomsky is the same fact regardless of the perspective from which it is seen is captured structurally in KRISP by having the individuals representing the two people point to the identical binding objects. This is impossible to state structurally in a KL-One -like representational system because roles (their equivalent of KRISP's variables) are not first class objects in these systems. Rather the 'advisor-of' role, let

us say, from the KL-One individual concept for Chomsky to the one for Ross is a different object from the, say, 'advised-by' role from Ross to Chomsky. We view this capacity to extend the uniqueness principle to frame–slot–value relationships to be a significant advance in knowledge representation design.

A *lambda-form* has four fields: the three fields of an individual and an additional field pointing to which of its variables have yet to be bound. There are two lambda-forms shown above in a highly abbreviated notation. Here they have taken on the role of categories in defining part of the types of the two person individuals. Lambda-forms can do this because they share with categories the essential property of defining variables, just as they share with individuals the property of participating in bindings.

Note that in the lambda calculus we can create quite elaborate functions by taking a formula that involves many predicates and function applications and then abstracting out a single deeply embedded variable. So in this case we have taken a compound predicate: 'an employment relationship between a person and a object of (domain) type 'position' indicating a particular title and organization', and have construed it as a function of one variable, 'title', and then bound this variable to the individual `#<title professor>`, leaving the other variables ('person', 'organization') open. Such a complex manipulation is warranted because English sanctions constructions like *Professor Chomsky* or *former President Carter*, indicating thereby that even though we know that Chomsky must be a professor at some particular institution, we are allowed to omit that information and still have a semantically sensible text.

To recap this section, we have presented KRISP's five types of representational objects, and illustrated how they are interrelated in a model via the pointers in their particular fields. We have shown how these types have a foundation in the lambda-calculus; they amount to a reification of different formula and function types. Categories correspond to predicates and define a set of local variables; individuals are fully saturated predications; bindings reify the application of a variable to a value; and lambda-forms correspond to partially saturated, possibly compound formulas.

We have alluded to the protocols that map these types to linguistic entities. A name denotes an individual, as does a phrase where all the required arguments to a given relational lexical head have been supplied. Lambda-forms are the denotations of syntactically complete phrases that are conceptually incomplete (unsaturated). Bindings are the denotations of most copular clauses. In the next section we will take up the details of how these denotations are established in the course of parsing a text.

The third aspect of a representational system, the framework that disciplines the work of a person developing a world model for some domain of discourse, has begun to raise its head here in the idea that we should model the relationship between a person and their title as involving the intermediate categories of 'employment' and 'position'. The rationale at work is that the representation of related texts should reflect their shared information through the use of the identical individuals, bindings, and lambda-forms. The vocabulary these texts use is being taken seriously as evidence for what categories we should postulate in order to make this structure-sharing possible.

Thus, to anticipate the next example, the fact that we can say such things as *Mr. Ball used to work for Prudential-Bache*, or *the positions of chairman and CEO at Pru-Bache used to be held by the same person*, is being taken as evidence that we should cluster the information in terms of the relationships of employment and positions, rather than, say, use a single category lumping all the variables together in a single relationship. By providing these additional categories we can capture the commonalities between these statements and the original by literally using of the identical units.

We are using units that contain smaller amounts of information than they might in principle, and in so doing are allowing the full 'retirement' relation to be factored in different ways,

grouping some of the units while excluding others, all the time structurally capturing the fact that they are indeed just different aspects of the identical information.

3. Integrating semantic interpretation and syntactic parsing

Having laid out the target representation for semantic interpretation, our next step is see how the language understanding system, Sparser, maps a text into its terms. In doing this, our main concern will be the general guidelines or 'protocols' that govern how the syntactic and lexical forms of natural language are to be interpreted in terms of KRISP objects. At the same time, however, something much more arbitrary, our choices of how to conceptualize particular kinds of information, must inevitably come into the discussion. Fortunately this is a separable aspect of the research, since one can disagree about whether, for example, there is an employment relationship in the chain that links a person to their title, while remaining comfortable with the representational system in which that relationship, or some alternative, is couched.

We should begin with a prècis of how Sparser works. Overall, Sparser is a one-pass parser that takes a stream of characters as input and produces a stream of transactions on its model of the world as its "output". A 'stream of transactions' is a somewhat unusual way of talking about what a parser produces, but it is fitting. Sparser scans entire news articles at a time rather than single sentences, and it makes crucial use of the interpretations of early phrases in the interpretation of later ones, especially for subsequent references to the same individuals and to fill in information that has been omitted as redundant or inferable from something said earlier. If Sparser were not doing semantic interpretation online as each word is scanned and each syntactic phrase completed, such operations would be either impossible or markedly less efficient.

There are roughly four kinds of transactions between Sparser and its world model represented in KRISP. The most common is noting a new instance of reference to an object already present in the model. All content words and many context-specializing syntactic constructions are already linked to their model-level counterparts as part of their definition in the grammar; indeed, one is not permitted to write a rule for Sparser (i.e. a context free or context-sensitive phrase structure rewriting-rule) without specifying its interpretation given the interpretations of the constituents it composes. The other standard transactions are the introduction of new individuals into the model, the addition of new bindings to established individuals, and the refinement of the model's set of categories through specializations.

The parsing process begins with the buffered stream of characters, potentially taken directly from a newswire. The stream is tokenized, recognizing known words and categorizing unknown words in terms of their morphological and orthographic properties. These words are then entered into a chart, where a set of parsing processes are applied, continuously producing a single set of semantically interpreted 'edges' (non-terminal parse nodes), the extensions to the model, and a discourse history as the text is processed. We can see this at work in a simple example by looking at a snapshot of the processing of the earlier example: *Ross was Chomsky's student* .

Sparser uses a semantic grammar. This means that its primary set[9] of non-terminal labels are drawn from a domain-specific vocabulary, just as the conceptualizations of its model are. Thus for our snapshot, the proper name *Ross* is spanned with an edge labeled 'person'; *was* is spanned with the label 'be'; and *Chomsky's student* is spanned with the label 'student'. Note that this three constituent sentential form is exactly the same as if we were parsing the sentence *Ross was a student* —the difference between the two is reflected only in the interpretations of their complements: the word *student* denotes a particular title, an individual; while *Chomsky's student* denotes a lambda-form as discussed above, effectively raising the type of its head to a level in the category hierarchy where it can stand in relation to a 'person' (more specifically a person with the title 'professor'; cf. *Robert Reich is one of President Clinton's economic advisors*).

Completion of the verb phrase *was Chomsky's student* would have resulted in a comparable type-raising if it hadn't already been done. This is because the VP must denote a predicate, an object with an open variable to which we can bind the subject. The composition of the subject NP *Ross* results in that binding, and ipsofacto the saturation of the advisor/ee relation that the complement NP introduced into the model, resulting in a normal individual: the individual-1 shown earlier in §2.2.

Let us review what was just described. We assumed that the individuals Haj Ross and Noam Chomsky were already represented in the model, and could be recognized from instances of their last names. This illustrates one protocol:

```
"prefer existing objects over new ones"
```

Similarly, the already known title individual denoted by *student* was referenced as the meaning of that word, this direct association to the appropriate object in the model being part of what it means for the system to have a word as part of its known vocabulary (along with representing its form class, noun, and its rules for syntactic combination). This illustrates another protocol:

```
"every sense of a known word is reflected in a direct link from the
 word to the corresponding model object"
```

The phrase, *Chomsky's student*, gets its interpretation in either of two ways. Either we already understand that combination and we describe the needed type-raising lambda-form in the statement of this rule for Sparser's semantic grammar (i.e. the rule `student -> person+possessive student`). Alternatively we deduce what the combination must denote by searching through the value-restriction statements of the categories in the taxonomic lattice to find the closest category with a variable restricted to be a 'student' and where there is also another variable restricted to be a 'person' or some specialization of a person. Plainly it is easier to already know what to do, and this is indeed one of the reasons for employing a semantic grammar in Sparser, namely to make it possible to encode quite specific semantic relationships by employing a set of semantic category labels that reflect the categories of the domain model.

At the verb phrase level, notice that there was no interpretation for the verb *was*. It did not refer to a category; in particular it did not refer to the hypothetical two-place relation 'isa' that was so often used in early versions of semantic network representations. Instead the word was treated as a literal, with the meaning of the VP coming for the most part from the identity of its complement. (Sparser's labeling conventions give this VP the label 'be-student', reflecting the passing through of the meaning-determining information.) The same is true for the other closed-class, grammatical function words: preposition, determiners, modals, etc., illustrating another protocol:

```
"function words and other grammar-specific lexical elements (e.g.
 past tense) do not have denotations per se, rather they frame
 grammar rules that modulate or pass-through the denotations of their
 syntactic arguments."
```

Strictly speaking this is not entirely true of *was* since that word does contribute the information that the relation held in the past relative to the time of speech. In some formula-oriented treatments of meaning this is captured by treating 'past' as an operator. In KRISP the analog is to add a binding—adding information to the VP's denotation without changing its type.

Similarly, other kinds of optional adjuncts such as location or more specific temporal information ("… *while he was at MIT*") are also treated by adding bindings to the individual representing the basic relationship. The set of variables sanctioning (making sense of) such adjuncts will be bound relatively high in the taxonomic hierarchy, in this case at the abstract category 'event', reflecting the verb-like nature of 'advisor/ee' as a relationship situated in time rather than, say, a spatial relationship situated in space.

Finally, notice that the interpretation of this sentence went directly from a linguistic surface analysis to the denotations of the phrases in the semantic model—there was no intervening level of analysis where the text was reformulated as, e.g., a logical expression that would only later receive a model-theoretic interpretation once it was complete. There are several reasons for this lack of a notion of 'logical form' in Sparser. The most important is the need for an online interpretation, where the referents of early phrases are available to aid in the interpretation of later ones. Another is that the construction of an intermediate descriptive formula seems only to muddy the waters, obscuring the real task of identifying the entities that the text refers to—especially considering that a great many of these entities will have existed well before the text was ever seen, and that the point of a semantic interpretation is to link the text to these entities, rather than to construct redundant descriptions of them. In a computational setting, a logical form may make sense for the task of question answering, where the formula can be readily recast as an expression in a data access language like SQL, and the model, such as it is, consists of an external data base. It makes markedly less sense, however, when the task is extracting information from extended texts over time, where a coherent history of the events that the system reads about requires a carefully structured model.

4. The Discourse-history and Cross-indexing of individuals

Suppose the system already knew that Ross was Chomsky's student, which is to say that there was an individual in the model for that particular advisor/ee relationship even before that text was parsed. In this case, by uniqueness principle, we want the semantic interpretation of the text to pick out the already existing individual, rather than create a new one. To do this we draw on another aspect of the KRISP representational system: its techniques for cross-indexing the objects in the model and its integration into the discourse history Sparser maintains as it is parsing a text.

We will look at these techniques in the context of a more interesting example, the paragraph given earlier about the retirement of Mr. George Ball from Prudential-Bache. Sparser's grammar for the Who's News domain can handle the main clause of its first sentence (repeated below), as well as the definite references in the second paragraph; the rest of the second paragraph is off the topic of business executives changing their jobs and is ignored.

> " *… George L. Ball resigned yesterday as chairman and chief executive officer of Prudential-Bache Securities Inc., the nation's fourth-largest securities firm.*
>
> *Mr. Ball's departure … Prudential Insurance Co. of America, the brokerage firm's parent … Mr. Ball's … the unit … Prudential … the 52-year-old chairman …*"

Sparser maintains a *discourse history*, where it records what objects (KRISP units) have been identified in the course of parsing the text. The record is added to incrementally as each phrase, with the object that denotes it, is completed. The history records every object, and indexes them by their position in the text: Mr. Ball thus has four instances in the history; Prudential-Bache has three; Prudential Insurance has two; the retirement event has two; and the title 'chairman' has two.

The discourse history is also indexed by the objects' domain-types, so that one can ask, e.g., was there a previous instance of the title 'chairman' and what context did it appear in. Presently the position indexes are reaped paragraph by paragraph once the parse has moved two paragraphs ahead, and we are experimenting with specific strategies for longer distance records, such as noting in this case that Mr. Ball is the thematic person.

As a result of parsing the first sentence, a complete retirement event, an individual, will be recorded in the discourse history. Briefly summarizing the course of that parse, it started semantically with the verb *resigned*, which denotes a category. This category reflects one particular sense of the word *resign*, which we indicate informally by giving it the name 'resign-from-a-position', and formally through the restrictions on its variables (its actor, the subject,

must be a person), and through the parsing rules written for it by which we indicate that it is intransitive, contrasting it with another sense of the verb that is fairly frequent in business texts: to resign debit.

The subject, Mr. Ball, is the first to compose with the verb, yielding in this case a lambda-form, since the resign-from-a-position relation is incomplete (unsaturated) unless a position, a title and its company, is also included. The needed position is the very next constituent, which in this grammar gets the label 'as-title'. Once this adjunct is added to the simple clause we get an individual as the denotation of the whole sentence.

When a new individual is created, it is not only indexed under its primary category, but also under a selected set of categories higher up along its specialization chain in the taxonomic hierarchy, according to annotations included with these abstract categories as part of their definition. (These are abstract in the sense that they do not have direct lexical realizations, but serve instead as the locus for inferences that are common to the whole set of lexicalizable categories that they are generalizations of.) For the present example these will include the categories 'leave-position' (generalizing over *quit*, *leave*<company>, etc.), 'job-event' (which adds *elect*, *appoint*, etc.), and finally 'event'.

To satisfy the uniqueness principle, we must never create a new object where there is already an object with the same properties already in the model. (By 'properties' we mean here the object's type, its categories, value-restrictions, and its bindings, which is about all there is to objects in KRISP.) This means that before creating an object, we should first look in the model for one that already satisfies the semantic interpretation rule of the phrase we are completing, given that phrase's daughter constituents. Object identity is thus defined recursively, with the formation of composite objects in the model mirroring step by step the formation of composite linguistic phrases in the text.

We have now introduced enough machinery to look at the semantic interpretation of the subsequent reference to the first paragraph's retirement event, the phrase *Mr. Ball's departure* at the beginning of the second paragraph. The head of that phrase, and consequently the basis of its interpretation, is the nominalized form of the verb *depart*. Like 'retire-from-a-position', this word includes the category 'leave-position' among its secondary indexes in this domain. Had the phrase been a clause based on the verb form, we would wait to look-for or create an individual until we had seen its direct object since one can leave many things besides jobs. However here we have a definite noun phrase—a subsequent reference to something that appeared earlier in the text. The information about what position Mr. Ball departed from has been omitted as redundant in this context, and we are left to reinstate that information by following out the one concrete anchor that we do have, the reference to Mr. Ball.

After the first sentence, we got the representation below for Mr. Ball. Note that we have changed the presentation of the bindings in this individual's bound-in field from the way they were shown earlier so as to emphasize the way they will be viewed here as indexes: We have left out the mention of the person being bound (it is, after all, always the very individual holding the binding), and we have presented more of the individual doing the binding, which is in this case the first instance of the retirement relationship. This is not a change in the definition of the objects, only in the perspective in the way Sparser' semantic interpretation processes view them.

```
#<individual  person-1
   :type  ( #<person> )
   :binds
       ( #<person.name = #<name "Ball, George L., Mr.">> )
   :bound-in
       ( #<employment/past  chairman & CEO, Pru-Bache>
         #<retire/leave-position/now  chairman & CEO …> )>
```

The lookup in the model runs roughly as follows. If the object denoted by *Mr. Ball's departure* is already in the model then there will be a binding tieing that individual to the individual representing Mr. Ball. We know that this individual will be the same one in both

instances, since semantic interpretation proceeds bottom up and left to right in step with the parsing, meaning that *Mr. Ball* will receive its interpretation before either *departure* or the whole phrase does.[10] Thus we look at the person individual denoted by this instance of *Mr. Ball* (it will be the same object as the earlier instance, we just retrieve it via a different index) and consider the bindings in its bound-in field.

We look at the list of categories of the individuals binding the person—their domain-type, and if one of these types satisfies the type denoted by *departure* then we have found our pre-existing individual. This check is a matter of looking for a shared secondary category index, in this case 'leave-position', and the same time anchor, 'now'. The second binding in person-1's bound-in slot has these properties, and so in accordance with the uniqueness principle we return the individual making that binding, which is the one denoted by the original full sentence.

5. Concluding remarks

KRISP is a practical system that has been deployed with the Sparser language understanding system over the course of the last year, following on a considerably larger body of work trying to identify just why other representational system were continually proving awkward to use as the source for the generation of fluent texts.

As a practical system in a working parser, it has undergone a goodly amount of revision as new problems in text were tackled. The lambda-form type, for example, is a recent addition, developed initially to solve the problem of how to represent phrases like *17 vice presidents* that are sometimes descriptions (making them nominally categories), and sometimes names (making them nominally individuals). Once a new type is introduced, it widens the design space available for solving semantic problems in texts, and so can lead to cleaner treatments of long-standing problems, such as the denotation of phrases that denote logically incomplete information.

Whenever there has been a question about what direction the design should take, the choice of analysis has always been make by falling back on how the information would look in the lambda calculus. This reinforces the basic distinction between predicates and their arguments, or between saturated relations and the functions that can be abstracted from them. KRISP is essentially just an object-oriented repackaging of the lambda calculus, with the insight that the binding of a variable to a value should be given a first class representation.

Another prime-directive is the principle of uniqueness. That, coupled with the goal of an online semantic interpretation during the course of a parse, led to the development of KRISP's system of cross-indexes between objects via bindings and the discourse history. Every representational system that is more than a paper notation will have comparable indexes just so that its users can access the objects they define. It is only in KRISP, so far as we are aware, that the indexing system is given substantial tasks such as finding the denotations of subsequent references, and consequently is brought under the scrutiny of the linguistic system, which provides a principled basis for deciding what is a good design or a bad design.

The evolution of KRISP will continue as improved treatments and alternative analyses are developed. Significant evolution in the design is likely soon as we actually hook up a generator and develop mapping tables for KRISP objects. We will start with the requirement that the mapping be bi-directional, and explore the hypothesis that 'anything a speaker can think (read 'formulate in its representation') it can say'. This is likely to stress many of the conceptualizations developed thus far only for parsing; but it can ultimately yield a sounder, more efficient design as we use the need to determine why the generator should produce one text rather than a variant with most of the same information to arrive at what primitives we should adopt and how we should structure their representation in the speaker's mental model.

6. References

Brachman, Ronald J. (1979) "On the Epistemological Status of Semantic Networks", in Findler (ed.) *Associative Networks*, Academic Press, New York.

_____ & James G. Schmolze (1985) "An Overview of the KL-ONE Knowledge Representation System", *Cognitive Science* 9, 171-216.

_____, Deborah McGuinness, Peter Patel-Schneider, Lori Resnick & Alexander Borgida (1991) "Living with Classic: When and How to Use a KL-ONE-like Language", in Sowa 1991, pp.401-456.

Fodor, Jerry (1983) *The Modularity of Mind*, MIT Press, Cambridge.

Jackendoff, Ray (1983) *Semantics and Cognition*, MIT Press, Cambridge, Massachusetts.

Mac Gregor, Robert (1991) "The Evolving Technology of Classification-based Knowledge Representation Systems, in Sowa 1991, pp.385-400.

Maida, Anthony & Stuart Shapiro (1982) "Intensional Concepts in Propositional Semantic Networks", *Cognitive Science* 6, 291-330.

Marslen-Wilson, William & Lorraine Komisarjevsky Tyler (1987) "Against Modularity", in Garfield, Jay (ed.) *Modularity in Knowledge Representation and Natural-Language Understanding*, MIT Press, pgs. 37-62.

McDonald, David D. (1993) "Reversible NLP by Deriving the Grammars from the Knowledge Base", ///////, in press.

_____ (1992a) "An Efficient Chart-based Algorithm for Partial-Parsing of Unrestricted Texts", proceedings of the 3d Conference on Applied Natural Language Processing (**ACL**), Trento, Italy, April 1992, pp. 193-200.

_____ (1992b) "Type-Driven Suppression of Redundancy in the Generation of Inference-Rich Reports", proceedings of the 6th Intl. Workshop for Natural Language Generation, Trento, Italy, April 1992, Springer-Verlag Lecture Notes in Artificial Intelligence #587, pp. 73-88.

Montague, Richard (1970) "The Proper Treatment of Quantification in Ordinary English", reprinted in Thomason (ed.) *Formal Philosophy: Selected Papers of Richard Montague*, Yale University Press, 1974.

Sowa, John (ed.) (1991) *Principles of Semantic Networks*, Morgan Kaufman, San Mateo, California.

Talmy, Leonard (1987) "The Relation of Grammar to Cognition", in Rudzka-Ostyn (ed.) *Topics in Cognitive Linguistics*, John Benjamins.

Woods, William (1987) "Knowledge Representation: What's important about it?", in Cercone & McCalla (eds) *The Knowledge Frontier*, Springer-Verlag, New York, pp. 44-79.

---

[1] Author's address: 14 Brantwood Road, Arlington MA 02174-8004.
Internet: MCDONALD@CS.BRANDEIS.EDU.

[2] For a discussion of one generation problem that we have treated with KRISP: how to control the realization of redundant information in an extended text, see McDonald 1992b.

[3] "Sparser" stands for "sparse parser", reflecting the fact that the system is designed to function robustly even when it has rules for only a small portion of a text. The portions within its grammar receive a thorough, largely conventional analysis; the other portions are treated heuristically or ignored entirely. This capability is a necessary property of any language understanding system that is applied to real texts taken directly from newswires without intervention or prefiltering.

[4] By "generator" I mean to refer to a system that handles all of the processes involved in the production of a text given a speaker's model of her world and her intentions. This is more than just a 'surface realization component' that starts with a logical form. It includes the processes that give an extended text its organization and coherence, the processes or mappings that find lexical and syntactic realizations for the conceptual objects in the speaker's model, and

(depending on the design) processes that examine the speaker's model to determine what information should be selected for inclusion in the utterance.

[5] This example, and the others that will be given in this section, is an example of the kinds of texts Sparser can handle with its present grammars.

[6] The standard notion of a model is a two-tuple consisting of a set of entities and an interpretation function. The entities supply the denotations of expressions in the language being interpreted, and the interpretation function defines the mapping from expressions to their denotations. We are proposing just such a model here, with the difference that rather than attempt to formally define the interpretation function we informally provide a set of protocols as its standin. To do otherwise would be presumptuous at this point, since after all the language we are mapping from is unrestricted English, and to be able to formalize its mapping function would be to have solved the natural language problem!

[7] See respectively Montague 1970; Brachman & Schmolze 1985, and Brachman et al. 1991.

[8] More generally, there can be a category for any sub-maximal projection of a lexical item. The unit corresponding to *London-based investment bank* would be a perfectly good category. A maximal projection such as the NP *the London-based investment bank* or *London-based investment banks* may or may not actually receive a denotation directly depending on their context in a text, but would, when parsed, invoke procedures to either search the discourse history for an individual that satisfied that description, or to create an individual of type 'investment bank' with a binding indicating its location.

[9] There is also a conventional set of syntactic labels accompanying the semantic labels on constituents. These labels, e.g., verb, modal, NP, clause, are used in default rules that are checked if no semantically-labeled rule applied to a given pair of adjacent constituents. These default rules of 'syntactic form' are presently used extensively within the verb-group but not at the clause level.

[10] This second instance of Mr. Ball is taken as denoting the same individual as the first by a comparable lookup process where the name object, broken down into its individual name elements, is itself reciprocally related to the person individual though bindings in the name elements' bound-in fields. The identity is established on the basis of sharing the same last name. Note that this works less well for the two companies, since they share the element *Prudential*; that judgement is more heuristic.