In Iwanska & Shapiro (eds.) Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language, MIT Press, 2000, pp. 77-110. Lightly edited 8/07.

# Issues in the Representation of Real Texts: The Design of KRISP

David D. McDonald[1]

May 1998

## Abstract

KRISP[2] 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 contexts. 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. It defines a small set of semantic object types, grounded in the lambda calculus, supports the principle of uniqueness, and supplies first class objects to represent partially-saturated relationships. KRISP is essentially an object-oriented repackaging of a typed lambda calculus, with the insight that the binding of a variable to a value should be given a first class representation.

As a representational system, KRISP embodies a theory of how information is structured in a natural language text. This is a hypothesis about the nature of the objects that texts denote and the principles of semantic interpretation that map them to a model comprised of such objects. Its fundamental motivations stem from investigations over the course of the last decades to try and establish just why it was that other representational systems were always turning out to be awkward when used as the source for the generation of fluent prose.

---

1 Author's address for correspondence: 14 Brantwood Road, davidmcdonald@alum.mit.edu. Other publications available via http://alum.mit.edu/www/davidmcdonald/.

2 "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.)

1. Introduction ..... 2
1.1 Representation ..... 3
1.2 Why a new representation ..... 3
1.3 Areas addressed ..... 4
2. Problems for Representations ..... 5
2.1 Displaced and non-standard constituents ..... 6
2.2 The source of the problems ..... 7
2.3 Glass ceiling ..... 8
2.4 Partially saturated relations ..... 9
2.5 Dynamically formulated functional types ..... 10
3. KRISP and SPARSER ..... 12
3.1 Having a model ..... 13
4. KRISP's set of Types ..... 16
4.1 'Units' – the sub-representational level ..... 16
4.2 Primary epistemological types ..... 17
4.2.1 Individuals ..... 17
4.2.2 Categories ..... 18
4.2.3 Variables ..... 19
4.2.4 Bindings ..... 20
4.3 Compound types ..... 21
5. Object-Creating Forms ..... 24
6. An Extended Example ..... 27
6.1 Simple semantic interpretation ..... 28
6.2 Semantic composition via interior open variables ..... 32
7. Principles ..... 35
8. Concluding remarks ..... 38
9. References ..... 39

## 1. Introduction

When a person uses language, there is a relationship between what they hear or say and things[3] in the world. The relationship is not direct; it is mediated by complex mental states and structures—the 'stuff' that is the mental target of the processes by which we make sense of what we hear or

[3] The most important of these 'things' are the people to whom the person is talking and the community that encompasses them. These are the truly interesting relationships, but unfortunately they are also the most difficult to capture given today's methodology and research.

read, and the source on which we draw as we speak or write. It encompasses a person's knowledge of language and his capacity to communicate given the available words and syntax. It includes a person's semantic conception of the world at large, and his pragmatic appreciation of the particular situation he is in at the time.

## 1.1 Representation

In the cognitive sciences, we typically think of this stuff as consisting of a representation that is acted upon by the processes that are set in motion when we use language. It provides a live model of the person's linguistic capacity, of their conception of the world, of their situation. What this representation consists of in the human mind we do not know. Like the other higher cognitive abilities that people have, it is not accessible to easy experimental manipulation and must be approached indirectly. That the 'stuff' even divides neatly into a representation and a set of manipulating processes is only a working hypothesis. All we know for sure is that the knowledge of the language–reality relationship must have some sort of long-term embodiment in the mind and that the nearest thing to that in those entities whose (limited) cognitive abilities we do understand—computers—is a representation.

The most familiar kind of representation for a non-technical reader is a map. As a representation, it indicates to its intended user (a person trying to find out where she is) what is in her world (streets, towns, points of the compass), their properties, and the relationships among them. It has a form (a large sheet of paper with carefully drawn colored lines, shadings, and marks). And it is used in a particular way (trace the lines representing streets with your eyes, read the names indicating towns, etc.). Notably, it also has to be brought into existence (engraved at the print shop), and it has a designer—the person who decides what it should contain, what form it should take, and what level of detail it should include.

## 1.2 Why a new representation

The thrust of this paper is to present a new representation of the literal information conveyed by a natural language utterance (which we will refer to from here on as a 'text' regardless of whether it is spoken or written). The reader should be suspicious of this. Representational systems (or 'representation languages') have been studied in artificial intelligence since the very beginnings of the field forty years ago. Logical systems for formal reasoning date back at least to the work of Frege and the meta-mathematicians that followed him more than a century ago and under some construals

back to the ancient Greeks. Reasoning and representation has become a sophisticated field in AI with a rich body of established practice and a dozen or more well characterized systems or methodologies that could be used 'right off the shelf' or easily implemented anew. Why then, given all the options already in existence, should we try to develop yet another one?

If we leave aside the practical issues of access or commercial licensing, the only reasonable answer to this question is that the existing systems are inadequate. A new set of problems has arisen, problems that none of the existing systems were designed to address. This is not to say that they could not be applied to these tasks, and many of them have been, rather that our goal is a well-engineered fitness to the task. Efficiency and explanatory constraints should emerge directly from the structure of the representation rather than stipulated as an afterthought or brought about through careful programming.

## 1.3 Areas addressed

There are three areas where we see problems that must be addressed in research on representations for modeling and manipulating the information in natural language texts.

- Comprehension of real-world texts
- Real-time ('online') processing speed
- Bi-directional representation

We are long-past the era of working with made-up examples or simple questions. Today we are studying the comprehension of real texts: texts written by professional writers to be read by people with a serious interest in knowing what they have to say.

Similarly, the system should be fast. Formally it should have online time-complexity (i.e. using a limited, constant amount of processing per token), and it should be real-time in the practical sense that it should, for example, be able to read news stories as fast as they appear on the web. In this, the representation should be a plus, not a burden.

The final desideratum, one that is at least as crucial in determining the form of the representation's design as the need for speed, is that the representation of the information that has been comprehended should be the very one from which it later produces ('generates') summaries, makes restatements from different perspectives, or translates into other languages.

This is the property of a representation being 'reversible' or 'bi-directional', and it tends to be neglected except for those relatively few places

that do research on generation systems. The benefits of having a reversible representation—in the parsimony of stating facts and the simple engineering efficiency—are well known (see, e.g., the collection of papers in Strzalkowski 1994), but the examples have been few.

It must also be said at the onset that there is also much that the representation described in this paper does not do. Its focus, as we shall see, is on what could be called 'practical' problems. Its capabilities are weak to non-existent when it comes to many of the basic issues in representation that concern classical logicians or mathematical linguists.

For example, except for very specialized technical problems in parsing, It has no deductive component. It has no easy way to express a general fact like 'all men are mortal'. And certainly no attempt has been made to handle the problems that vex linguistic philosophers such as those inherent in the notorious 'donkey sentences' ("*If Pedro owns a donkey, he will ride it to town tomorrow*"; see Schubert & Pelletier 1989 for an enlightening discussion).

Part of the reason for these lacunae is simply a matter of time and attention: less than one man-year has gone into the development of KRISP to date. But the more substantial reason is that the question of how to even approach issues of deduction and generalization, when applied to the full subtleties of real texts (consider this sentence), remains very much unanswered within the computational linguistics and AI communities. Because I decided long ago that it was no longer going to advance the field to work on small or artificial problems, the complexity of these issues in their real context makes them something to put on the shelf for another day.

## 2. Problems for Representations

Before proceeding to describe the new representation—KRISP—it is important to provide some concrete examples of the problems that prompted its development. In this section, we will look at examples in language comprehension and (informally) at the kinds of representational devices KRISP uses to deal with them.

First some background. The initial work on KRISP was done in the early 1990s and it was first deployed as part of the author's work with colleagues at Brandeis University in DARPA's Tipster program (DARPA 1993). Tipster involved (in part) the analysis of a large corpus (1,000 articles) of text in English in the domain of 'joint ventures'—the investment by several companies in some new firm or enterprise that they own

together. The corpus was taken from newspapers and newswires presenting straight news (rather than commentary) aimed directly at the business community.

Tipster-1 was an ambitious attempt to develop a serious, exploitable capability to do a very particular and limited kind of language comprehension that goes under the rubric of "information extraction". The managers of the Tipster program appreciated that it was virtually impossible even just to define the range of knowledge that goes into understanding all of the information presented in these texts (as the human businessman would do), and so they defined a specific, limited set of information that systems were to find in the text and then 'extract' it and put into the appropriate fields of a relational database in a well defined format.

## 2.1 Displaced and non-standard constituents

To see two of the representation problems that KRISP addresses, consider the sentence below; the second sentence in a short article from the Tipster joint ventures corpus.

> *"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."*

Here the information to be extracted is the relative portion of the new joint venture that each of the three participating companies owns. The fact that Cybac is an advertising company, for example, is completely irrelevant to the task. The system doing the comprehension is free to ignore it and needs to know nothing at all about advertising or other types of business activity.

Even with these simplifications, the task remains a daunting one because those sentence poses difficult problems of syntactic and semantic analysis. It is virtually certain that no system developed in the 1970s or 1980s (when most of the fundamental work on language comprehension by machines was done) could have handled it, and few can do so today. If that were not enough, this particular pattern of how 'joint capitalization' is expressed is only one of about a dozen, and is not at all rare.

In this instance the sentence poses two problems. The first is the displaced constituent "*50 percent*"—'displaced' because it does not appear in its canonical position, which for this constituent would be in a partitive construction such as"*50 percent of the new firm*". If the focus of the author of that article had been on the parent companies rather than the new venture

then it might have been phrased *"Takarabune will own 50 percent of the new company, Merciries 45 percent, and …"*. A generation system might want to do that as well.

The second problem is the 'non-standard constituents' in the later part of the sentence. As speakers of English, we all know that the phrase "*45 percent*" refers to a '45% ownership of the new firm' yet that information is not given explicitly but must be inferred from context. The question is how to do that inference in a general and reliable way.

### 2.2 The source of the problems

The standard means of identifying and populating a domain relationship during the semantic interpretation of a parse is to take the syntactic head of the phrase as the basis of the type of object or relation to be instantiated, e.g. "*firm*" or "*owned*", and then to interpret the syntactic arguments to the head as values for the relation's arguments or the object's properties, e.g. "*new*" or "*by Takarabune*". What makes non-standard constituents a problem in this approach is that a 'constituent' like "*45 percent by Merciries*" does not permit this kind of analysis. If this phrase is indeed a constituent then its head would be '45 percent', and there is no sensible way that a company can directly provide an argument to a percentage.

Similarly, the constituent "*50 percent*" is displaced in the sense that it is not in construction with (roughly speaking, adjacent to) the syntactic argument or head that it would need in order to denote a normal relation. In the era of transformational grammar we would have said it had moved; today we would be more circumspect. The problem for the parser is to identify which of the other phrases in the text is its head and to then arrange for the two to be passed on to semantic analysis as though they had appeared in the canonical partitive construction.

We can summarize these two problems by saying that they arise from phrasal patterns that violate the presumptions of traditional, syntax-driven compositional semantic analysis, presumptions that have been in place in linguistic semantics since (at least) the work of Montague (1970, or Partee 1984) or in artificial intelligence since Winograd (1972) and Woods (1973).

In the artificial intelligence version of this tradition (the only option with real parsers and generators), the stuff that is the semantic representation of the information in a text consists of terms representing individuals (e.g. the four companies in the example) and predicates representing states, properties, and relations (e.g. 'own', 'be a certain percentage of', 'new'). Each word or fixed phrase typically has a direct projection onto a term or a

predicate, and the syntax of the text, as determined by the grammar that the parser is using, defines how they are composed into larger and larger forms by applying the terms as arguments to the predicates in order to create formulas. In the more recent treatments, the terms and predicates may be typed, structured objects and the information each object represents may actually be distributed across the elements of complex taxonomic or meronymic lattices, but the essence is still the same: a syntactic grammar of adjacency relationships between constituents drives the process of semantic composition and determines the content and format of the final representation of the information in the text.

Given these sorts of representational elements with which to work, the only option for dealing with displaced constituents is either to greatly complicate the machinery of the parser (by introducing transformations that will move the constituent to the 'correct' place) or to add constructions to the grammar that capture the newly observed pattern directly. The same is true of non-standard constituents, where the problems are so significant that the approach is often to move to a syntactic formalism that is specifically attuned to the problem, such as categorial grammar (Steedman 1987, 1996).

## 2.3 Glass ceiling

The problem with this approach, which the experience with the Tipster corpus makes plain, is that real texts exhibit a strikingly large variety of phrasal patterns, far more than would have been considered a 'large' set in the past when developing grammars for restricted, short-utterance tasks like question answering. I believe that we cannot expect to succeed if what we do is attempt to catalog all the individual syntactic constructions and give each of them its own compositional, rule-by-rule interpretation. The enormous number of constructions and the variations imposed on them by the peculiarities of the semantics of different words will swamp any such effort.

We may be seeing this already in what has become known among the participants in the MUC ("Message Understanding Conferences") and Tipster community as the 'glass ceiling' that limits further progress in information extraction. As of when this was written, for reasons unknown, no one in that community has had a combined precision/recall score in their tests on blind texts higher than 60% (which also includes a non-trivial number of mistakes) despite over six years of trying. (See Sundheim 1995, particularly Table 4, pg. 27.). Systems instead climb steadily to about that

point as they mature, but then stay there, so far indefinitely. (Inter-subject agreement among people in the same task is above 95%.)

If the experience of the Brandeis group is any judge, the source of the glass ceiling lies directly in the central problem of real texts: that there are a great many ways of saying the same thing. Taken as a problem in grammar, identifying and coding all these patterns is a Herculean task. In the tests, groups make mistakes or miss instances simply because they are confronted with constituent patterns that their systems have never seen before.

As the reader probably already expects, I see the answer to the glass ceiling to be to move ever more of the work out of syntax and into in semantics. Keep the grammar and the style of the parser's syntactic analysis relatively simple and move the complexity into the semantics by developing more powerful compositional mechanisms and more expressive and flexible representations. KRISP is offered as a first step in this direction.

### 2.4 Partially saturated relations

One of the principles underlying the design of KRISP is that the semantic projection of any grammatically well-formed phrase in a text should be a first-class object. In KRISP the phrase "*50 percent*" projects to a first class object that is ontologically a partially saturated relation: a function that carries within it information about what kind of thing it combines with and the role that it plays when the composition is finally formed. (The details of this notion are elaborated in §4.3.)

The concept 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 cannot be factored out—are not first class—since they gain their meaning only by being in construction with a particularly placed quantifier and appearing at certain positions in a formula. For example, imagine that we have the following formula in the predicate calculus as the representation of "*Takarabune owns 50 percent of the firm*":

```
Exists(t) Exists(f)
   name(t, "Takarabune") AND known(f) AND company(f)
     AND jointly-owns(t, f, 50)
```

Here the letters t and f have no meaning on their own; the t could just as well be the t in "*jointly*" for all we know about it outside of that formula. Only by combining these letters with existential quantifiers do we establish their semantic role as variables, and even then only for this particular formula. The same is true of the number 50 in that formula, which takes on its relevant meaning only by being the third argument in a formula with the predicate 'jointly-owns'.

In KRISP, we can take information that we know is incomplete ('50 percent of what?') and represent it as a thing that we can pass around and manipulate independently of the larger context. The problem of a displaced constituent then becomes a matter of allowing the parser to carry the un-composable partially saturated relations up the headline (the progressive composition of the denotation of "*owned*" with its regular arguments) until a binding can be found for it with a suitable type. The movement of the constituent back to its canonical location is thus performed at the semantic level rather than the syntactic.

### 2.5 Dynamically formulated functional types

Using KRISP, the problem of non-standard syntactic constituents such as the combination in this example of a quantifier ("*45 percent*") and a prepositional phrase ("*by Merciries*") is also solved at a semantic level. Linguistically, what we are presented with is the elision of the other constituents of what would have been a full clause had it appeared first in this conjoined sentence rather than second (e.g. "*Merciries owns 45 percent of the new firm*"). We can get similarly reduced sets of non-composing constituents in the other syntactic contexts that permit what was once called reduction under identity, such as in answer to a question.

Leaving aside the question of how a parser would determine that the correct syntactic scope of "*45 percent by Merciries*" is the whole first clause ("*The new firm will be 50 percent owned by Takarabune*"), the question for the semantic interpretation process is how do we know what was left out from the reduced conjuncts and how do we restore it so that each conjunct will semantically denote an object that carries the same amount of information as the denotation of the original clause. The procedure has to be a dynamic one—formulated on the fly as the analysis is done—since even a cursory examination of a real corpus will show that we

cannot do this with any sort of fixed interpretation schema. There are too many possibilities for what can appear as the reduced, non-standard constituents for any such treatment to be possible.

The general formulation of the treatment here is to treat the phrases of the non-standard constituent as arguments to a function. Applying the arguments yields a complete (fully saturated) relation (i.e. a formula); the question is how to construct the function. With KRISP as the basis of the representation, we can break down any multi-term relation into a set of individual variable bindings. From this perspective the first, full clause of the sentence is a set of three bindings: one for 'the new firm', i.e. the joint venture playing the role of the thing that is partially owned; one for 'Takarabune', which is one of the parent companies playing the role of the (partial) owner; and one for '50 percent', the measurement playing the role of the amount of the partial ownership.

We form the function dynamically by examining the domain-level types of the constituents in the reduced conjuncts (in this case a measurement and one of the parent companies) and subtracting the equivalent terms from the object denoted by the full clause. This creates a first-class, partially-saturated relation that is underspecified in two of its variables, which can then be applied to the two uncomposed terms in the reduced clauses to form full relations.

Semantic interpretation through the application of complex functions is standard notion in the semantics of categorial grammars. What we are doing here is moving the construction of the function from domain of syntactic parsing to that of semantic composition. In a syntactically driven approach, we would have needed to have a rule that combined a measurement quantifier and agentive prepositional phrase because we would need a syntactic structure to categorize the roles of the two constituents before we could proceed with a semantic analysis. This has the obvious, possibly insurmountable problem of needing to enumerate and formulate all of the possible combinations of reduced constituents (equivalently, what can be elided from the first 'full' clause of the conjunction). Using KRISP, all we need at the syntactic level is a segmentation that delimits the scope of the conjuncts, and the substantive work is done at the semantic level. This would not be possible if we had not expanded the number of kinds of semantic stuff that we allow in our interpretations and the kinds of operations we allowed to happen in the course of formulating an interpretation, which is the thrust of the project to develop KRISP.

## 3. KRISP and SPARSER

The concepts behind KRISP are the result of a long-standing body of work on efficient representations for language generation. If we start from information represented in KRISP, we should get a generation process that is maximally efficient in the sense of McDonald, Meteer, and Pustejovsky (1987)—one where the representation of the speaker's situation and its mapping to linguistic resources is organized in a finely modularized, readily recombined network of elements from which information can be selected and given a linguistic formulation in a way that is easily tuned to the requirements of the focus, perspective, and degree of detail that happen to hold at the time.

At this time, the use of KRISP for generation remains largely just a goal. The substantial development and implementation of KRISP has been instead as the target semantic representation of the language comprehension system SPARSER (McDonald 1992). SPARSER is a fully implemented, mature system for natural language comprehension that was developed during the early 1990s as a commercial information extraction system targeted at the rapid analysis of single subject, information-dense, typically short, business news articles such as one finds on newswire services and in the columns of newspapers such as the Wall Street Journal or the Financial Times (the primary sources for the Tipster corpus). It has since been applied to a variety of corpora and language-analysis tasks. Substantial grammars have been developed for the articles in the Wall Street Journal's "Who's News" column (the source of the texts used in MUC-6 (DARPA 1995)) and for press releases reporting companies quarterly earnings. It has also been applied to the problem of acquiring the concepts of a new domain and their linguistic realizations from large corpora, in particular a set of reference manuals for the Macintosh operating system.

SPARSER produces a full, in-depth, linguistically principled analysis of those parts of the text that it can understand, deliberately ignoring the rest. Its analyses will omit much of the information that (we assume) a person would understand; but in those parts for which it does have a grammar and semantic model, its understanding of the literal content of a text is just as detailed as what a person would arrive at. In this sense it is not a 'partial' parser as that term is understood in the parsing literature (see, e.g., Bunt and Tomita 1996) but rather a 'sparse' parser.

SPARSER is among the very fastest information extraction systems in the literature (compare Appelt et al. 1995). Its speed varies with the percentage of the text that receives an analysis. The article below (Wall

Street Journal, November 2d, 1989 from the Data Collection Initiative's corpus), for example, is analyzed in every detail and is processed at the rate of 180 words per second (50Mhz Apple Quadra 900 using Macintosh Common Lisp version 2.1); it takes longer to print the words of the text on the screen than it does to process it.

> *"Economist Newspaper Ltd. (London) --- Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29. Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group."*

By contrast, only about a half of this next text is analyzed (just the elements that are underlined), and it runs at 240 w/sec. (It is excerpted from a Wall Street Journal "Who's News" article from February 14, 1991). A text consisting of just a sequence of uninterpreted nonsense words ("aaa") runs at 900 w/sec.

> *"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."*

The output of SPARSER is a stream of event triggers and a domain-level semantic model populated with the individuals, properties, and relations that were detected in the text, all represented using KRISP. The semantic grammar that drives SPARSER's analysis is instantiated as a side-effect of executing KRISP object and category creating expressions. Known objects in the model are looked up when the text refers to them by using automatically created mappings included with the rules of the grammar. New KRISP individuals and (notably) composite categories are created as needed to represent new information found during an analysis. All this will be described in later sections.

### 3.1 Having a model

The relationship between the text and the representation of its content in SPARSER/KRISP is direct and immediate. It is a denotational relationship of just the same sort as the one between expressions in a

mathematical logic (a kind of language) and the model that provides the basis for the truth values of those expressions by providing an interpretation for their terms. In SPARSER's use of KRISP, there is no intermediate language of 'logical form' into which the text is first reformulated; no mediating intensional logic. The words in the text and the syntactic objects built from them have an extensional mapping to corresponding objects in SPARSER's internal model of the world.

It is worth taking a moment to make sure that it is clear what we are talking about ontologically when we say 'model'. There is a tendency, when logicians look at a representation developed in AI, for them to see things of a syntactic sort not unlike their own parentheses and letters; that is, things that require an interpretation before their denotations can be established. (Searle (1992) is a good example of this sort of argument. For exceptionally well articulated counter-arguments see Smith (1996).) Models for most logicians are abstract things about which they prove properties; not things that they actually construct. Furthermore, the stuff that populates a logician's model consists of individuals of very simple types such as pure numbers or tuples. The menagerie of types and structures in an AI system is alien to that way of thinking, if only because its complexity makes it virtually impossible to prove theorems about it. Similar considerations lead many within AI to eschew such complex, 'scruffy' treatments in favor of the first order predicate calculus.

To understand what a model is like in KRISP it will be helpful to review the usual steps taken by an AI system that can answer questions. What sort of stuff does it manipulate along the way? A question-answering system starts with an expression written in a language, say an English sentence. As it parses the expression and does its semantic interpretation, it typically constructs another linguistic (symbolic) expression, for example a query written in a database access language like SQL. That expression is then passed to the database and is itself parsed and interpreted. The result at that point is something different. We stop making more expressions and start to execute actions against stuff of a different kind: tuples in a database. Expressions were used, offline, to create those tuples and have them added to the database, but once there and active they became the stuff of a model, not linguistic objects.

Consider another case whose fuzzy edges may make things clearer. In Winograd's classic SHRDLU system (1972), the output of the semantic interpretation phase was also expressions: small programs written on the fly in the programming language MicroPlanner. Suppose, contrary to fact, that

the entire range of what SHRDLU could understand had been mapped out and schematized, and that rather than build its MicroPlanner programs from scratch it would just select the appropriate program already compiled down to PDP-10 assembly language, set its parameters, and execute it. Did it still make an expression requiring interpretation? I think the answer to that question is yes. But again, the blocks world against which those programs would have run was comprised of stuff of a different kind. It was a circumscribed universe of objects consisting of the children's blocks on SHRDLU's table and the spatial relationships among the blocks and with the table.

Let us consider what kind of stuff they consisted of. There actually was a (short) time when one of the experimental pick and place robots at the MIT AI Lab was deployed to manipulate physical blocks under instructions from a reimplementation of SHRDLU. In this case the blocks world was part of our ordinary external reality, which is decidedly not an expression requiring interpretation. In the original implementation there was also the manipulation of non-linguistic things, namely the display of a simulated robot hand and table with blocks on a computer screen. Ray tracings and excited phosphors are not linguistic stuff, but by the same token they are not, per se, stuff that is isomorphic to the information content of the sentences SHRDLU understood its users to be talking about when they typed in questions for it to answer or commands for it to execute. That connection lay only in the mind of the human viewer of the display.

What then is the ontological status of the stuff inside SHRDLU that it made reference to when it had understood a sentence and was proceeding to set up the directives to drive the display or move the robot arm? It used Lisp symbols such as ':B6' to stand for particular blocks, and it used Lisp lists to stand for the relations among them, '(on :B6 :B3)', all of which were things constructed out of the memory locations and bit patterns of the computer it was running on, and accessible by name or pointer from the Lisp data heap in which they resided.[4] This, I claim, is the stuff of a model in the same sense as a logician would use that term; as are the KRISP units

[4] Note that when we write '(on :B6 :B3)' we are writing an expression in a grammar of parentheses, letters, and whitespace that a Lisp program can interpret as directives to identify certain systematically corresponding internal data structures of an ontologically different sort than we typed in. Do not confuse the expression (a means to an end) with the thing it denotes. In a significant sense the data structure is really manipulable only to the running computer program in which it resides (i.e. SHRDLU). Any possibility for us, as people, to observe it will require the construction of some external expression or visual configuration such as a print out in a symbolic debugger or a hexadecimal core dump.

that comprise the content of SPARSER's understanding of the texts it analyzes.

## 4. KRISP's set of Types

As a representational system, KRISP embodies a theory of how information is structured in a natural language text. This is a hypothesis about the nature of the objects that texts denote and the principles of semantic interpretation that map them to a model comprised of such objects. In KRISP the objects lie on two different levels, one realized in terms of the other.

As the interpretation of a text we have objects that lie at 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. Entities at the domain level are in turn constructed out of stuff that resides at the 'epistemological' level (following Brachman 1979). At this level lie the fundamental kinds of representational objects—the paper and ink from which the map is assembled. Generalizations about the nature of the mapping from linguistic (text-level) entities to domain level objects are stated here.

We introduce KRISP's epistemological level below. This will include giving part of the rules of formation for domain-level units by discussing how each type of unit is typically realized in English. A full example of a semantic-interpretation will follow in §5.

### 4.1 'Units' – the sub-representational level

In a functioning system, every representational level is itself realized (implemented, made operational) by objects and processes at a still lower level. At the lowest level of any semantic interest in KRISP (below it lie the entities of the programming language it is runs in, Lisp in this case), a model consists of a set (technically a heap) of typed objects—the 'units' of information that the model contains. Units are the fundamental chunks from which the information in a text is presumed to be composed. Units virtually always refer to other units, but there is a notion of minimality to them. Nothing smaller than a unit carries any information content.

Units are linked together by pointers. The units are first class objects; the pointers are not. Units are structured, with named fields containing pointers from them to other units. The set of fields that a unit has are determined by its type. Every unit is unique; that is, there will only ever be

one unit in the model with a given type and given set of units that it points to in its fields. This uniqueness is maintained by the machinery that brings long-term units into existence and manages the semantic interpretation process.

While the notion of a unit having a name is not really sensible (names are themselves entities that need a representation in the model), there still has to be some way to differentiate one unit from the next in an exposition like this (or while debugging a program), and the simplest convention to adopt is to use the word to which the unit would normally correspond as the basis of its name. When necessary, we will use a bracket notation ("#<…>") to indicate that we are dealing with model-level objects rather than the words or expressions that realize them in a text.

### 4.2 Primary epistemological types

KRISP defines four primary types of units in its epistemological level: 'individual', 'category', 'variable' and 'binding'. It also defines two types of units that are blends of the primary types: 'derived categories' and 'partially saturated individuals'. Every object (unit) in a KRISP model has one (and only one) of these types.

#### 4.2.1 Individuals

For concreteness, the discussion here will make use of this sentence, which we have seen earlier, and we will start with the notion of an individual, the most ordinary of the six types of units in KRISP.

> *"George L. Ball resigned yesterday as chairman and chief executive officer of Prudential-Bache Securities Inc."*

There are 15 individuals in this text as analyzed by SPARSER's semantic grammar for Who's News articles. We have one person, one 'resign' event, one day, two titles, one company, two names (one each for the person and the company) and within those names six 'name elements' and one 'incorporation term'. (For a discussion of how proper names are analyzed see McDonald 1996.)

An individual is roughly comparable to Montague's 'e' type, KL-One's 'nexuses', or Classic's 'individuals' (see respectively Montague 1970, Brachman & Schmolze 1985, and Brachman et al. 1991). Individuals are used to represent particular things in the world, concrete or abstract. In terms of semantic interpretation, they are the denotations of maximal-projections of the major syntactic categories, e.g. proper names (designating conventional individuals), referential noun phrases (things

anchored in space), and most clauses (things anchored in time; see Talmy 1987).

As a unit, an individual has three fields: one for its type and two for the relationships in which it participates. The nature of these relationships will be taken up below as part of the description of bindings. To define what can appear in (is pointed to by) the type field of an individual we must now look at the notion of a category.

### 4.2.2 Categories

Categories represent kinds. They are the denotations of most phrasal heads, e.g. common nouns and verbs. The type field of an individual contains at least one but possibly several category units that collectively define its domain-level type (e.g. transition event, person, being in a state of retirement, title, unicorn). The relationships into which an individual can enter are determined by the categories listed in its type. When the type field contains more than one category, one of them is indicated as primary. It will be an immutable, defining aspect of the individual—'person' rather 'programmer'—and defines most of the individual's properties. The primary category is central to an individual's semantic identity as the denotation of a natural language word or phrase, hence 'resign-1' rather than, say, 'job-event-1'. By convention, the primary type is listed first when there are several categories.[5]

Categories are predicates. They define relational types with individuals as the corresponding tokens. As a predicate, a category has a specific arity. In a conventional AI frame system we would say that it has a certain set of 'slots'. In mathematical logic we would say that a formula based on that predicate would have a certain number of variables. In the lambda calculus—the best reference model for KRISP's epistemological level—we would say that a category is a function open in a certain number of variables. In KRISP we also use the notion of variables and make them first class objects. The number and identity of the variables associated with a category, along with its position in the lattice (below), are what constitute a category's identity and information content; qua unit, they are a category's two fields.

---

[5] For the purpose of this paper, we will designate newly created individuals using a gensym based on their primary category: #<resign-1>, #<person-3>. If the individual is part of Sparser's background knowledge as the denotation of some proper name or the equivalent, then we will just use that English phrase: #<chairman>. If the type of a unit is not clear from context it is included: #<individual 2/13/91>. If the sortal distinction (the domain type) is important and the epistemological type is clear from context, then the individual's primary category is used: #<name-of-a-person "Ball, George L.">.

### 4.2.3 Variables

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 (alpha reduction in the lambda calculus). Every variable has a value restriction: a category or list of alternative categories that stipulate the kinds of individuals it can be bound to. The two fields on a unit of type variable are the category to which it belongs and its value restriction.

In the example at the start of this section, the verb "*resign*", which is the head of its clause, denotes a category. This category is part of SPARSER's a priori knowledge about the domain of Who's News, and was entered by hand into the model using a category-defining expression as we will see in the next section. It is given the internal name 'resign-position' to distinguish it from the other sense of the word that appears frequently in business text, as in "*to resign debt*". Resign-position has two variables, one for the person resigning and another for the position[6] they resigned from.

Categories are organized into a subsumption lattice on the basis of the variables they define and the restrictions on them. The variables are also linked into this lattice so that the correspondence between the variables of a category and those of its super-category(s) is well-founded.[7] An individual whose domain type includes a relatively specific category, like 'resign-position', will also satisfy—read: may bind the variables of—all of the categories that resign-position is a specialization of, e.g., 'leave-position', 'job-change', 'agentive-event', 'transition-event', and 'event'.

Subsumption, however, is not a well-developed aspect of KRISP because it was designed as a classifier where a set of descriptions would be given to it and it would find the node in the lattice at which they should be placed. Rather KRISP is a source of denotations that are established compositionally through the actions of a parser. Any new, derived categories that a text may refer to are always the daughters (subcategories)

---

6 Here a 'position' is category that combines a position proper, as designated by some title or titles, and the company at which the position is held: *chairman and chief executive officer of Prudential-Bache Securities Inc*. Making choices about how information is to be grouped into units is the interesting and substantial work of defining a domain model. In this case it was motivated by the fact that the position and company are invariably realized together in a single constituent.

7 This means that the set of pointers connecting categories/variables in this lattice are effectively organized into 'cables', a notion that was originally articulated by Ron Brachman and Brian Smith during the 1970s as part of a never-completed declarative calculus to be call 'Mantiq' (Smith, *pers com*. 8/07).

of the elements from which they are built and lexically anchored categories are effectively natural kinds. Derived categories are kept unique through the channeling provided by the variable binding lattice (see below).

### 4.2.4 Bindings

Relationships and their components in KRISP are represented by units of type binding. Bindings are three-tuples consisting of a unit (typically an individual), a variable, and another unit. They represent facts of the form 'unit-1 stands in the relation variable-1 to unit-2'. Broadly speaking, bindings are the frame–slot–value structures of standard frame systems, with the difference that binding are first-class objects and exist independently of the units that comprise them. Bindings are sanctioned by the variables defined by the participating individuals' categories. All of the relationships in which an individual participates and all of its attributes other than its categorizations *per se* are represented by the bindings it is part of.

Bindings provide the denotations for a number of linguistic constructions including most copular clauses and the individual grammatical relationships that tie phrasal heads to their complements and adjuncts (subject, temporal adjunct, etc.). In language comprehension, most semantic relations correspond to the syntactic relationship of a head to one of its arguments, with the set of relationships (bindings) accruing incrementally from the head outwards in the course of a parse. The head will denote a category, and when all of its required arguments have been determined (those required logically, not just syntactically), the result will be a fully saturated relation, i.e. an individual.

To have the best understand the notion of a binding we should consider what a category like resign-position would look like in the lambda calculus, where it is a function based on a predicate with two arguments. The reference text here is our ongoing example: "*George L. Ball resigned yesterday as chairman and chief executive officer of Prudential-Bache Securities Inc*".

```
λ position . λ person .
    resign-position(person, position)
```

If we ignore the question of sense disambiguation for the moment, in our current example the verb "*resign*" will, given SPARSER's grammar, compose first with the adverb "*yesterday*", binding the time variable that resign-position inherits from the event category, and the resulting verb group will then compose with the position constituent "*chairman and chief*

*executive officer of Prudential-Bache Securities Inc*." and bind resign-position's position variable. In the lambda calculus, the resulting expression—the denotation of the verb phrase—would look like this:

```
λ person .
  ( ( λ position .
        ( λ time .
              resign-position/event
                  (person, position, time) )
        #<date 2/13/91> )
    #<position Chairman & CEO, Prudential-Bache
               Securities> )
```

This expression shows that we have applied a individual to the variable 'position', in preparation for doing a beta reduction; and that we have done the same for the inherited variable 'time', applying an individual of type date, which is one of the subcategories of the type 'moment-in-time' that is the value restriction on the generic time variable in SPARSER's model of the world.

In KRISP, unlike the lambda calculus, those argument applications are represented by first class objects: bindings. The equivalent of that aspect of the lambda form above are these two bindings, shown relative to the (partially saturated) individual that is the equivalent of the formula's body, designated 'resign-position 16'.[8]

```
bindings 105
  :variable #<position>
  :value #<position 1>
  :body #<resign-position 16>

binding 102
  :variable #<time>
  :value #<date 2/13/91>
  :body #<resign-position 16>
```

## 4.3 Compound types

What are we to say about the epistemological status of the denotation for this verb phrase in this example ("*resigned yesterday as chairman and*

[8] Numbers such as this '16' are taken from the actual indicies that were used in a run of the example text done when this article was being written. They have no intrinsic significance and just serve to differentiate one unit of a given type from another. They derive from the units' positions in the resource heap of the Lisp implementation.

*chief executive officer of Prudential-Bache Securities Inc*.")? If we accept that it is a unit of information, rather than just something ephemeral in the state of the semantic interpretation of the full sentence, then we have to decide what the unit's type should be. Given the options so far, this unit has some of the properties of a category in that it defines a predicate that is being open in the variable 'person'. It also has some of the properties of an individual in that it is a particular (if incomplete) thing in the world, being an event anchored to a particular day and involving a particular position, which is sufficient information to make it unique under most circumstances.

To accommodate such objects in a disciplined way, KRISP has the type **partially saturated individual**. Like individuals, such units have a domain type and participate in bindings. Like categories, they define a set of open variables and have positions in the taxonomic lattice. As units they have the fields of an individual plus the field 'open-in' which lists the variables that have yet to be bound. It is suggestive, though not technically accurate, to say that a partially saturated individual that finally gets all of its open variables bound in the course of an interpretation changes its type at that moment and becomes an individual. (Actually a new object is created. The others are retained during at least the life of the analysis as candidate derived categories. See below.) Here is the printed form of the denotation of this verb phrase; the bindings are printed in a reduced format since the identity of their body is understood.

```
resign-position 16
  :type ( resign-position past )
  :bindings ( #<time = #<date 2/13/91>>
              #<position = #<position 1>> )
  :open-in ( #<person> )
```

Earlier in this section, single nouns, verbs, and most fixed phrases were described as denoting categories. This is not strictly accurate as it depends on precisely what we call a word. The denotation of the verb is different in "*he resigned*" versus "*he intends to resign*". The first picks out an event that occurred, the second one an event that may or may not occur—more of a type than a token. Moreover, while we don't know from just its spelling-form what would satisfy an instance of the infinitive form of "*resign*" in the world, we do know something about an instance of "*resigned*", namely that it categorizes an event that happened in the past. To deal in a principled way with such differences, KRISP includes units of type **derived category** in its epistemological level.

Simple, primitive categories (natural kinds) are the denotations of uninflected content words. They are primitive in the technical sense that their position in the lattice does not provide necessary or sufficient criteria for their satisfaction. Derived categories are analytic and are the denotations of compositions of linguistic primitives. Most reflect the addition of morphological or grammatical elements such as plural, past-tense, modals, negation, etc. Many descriptive N-bar level phrases also want to be treated as derived categories since they do not introduce individuals in a text but rather categorize existing individuals or add attributive properties. Something as elaborate as "*London-based investment bank*" can be reified as a derived category.

In practice there is a fine line between a derived category and a partially saturated individual. Phrases that we want to treat as categories because of their function as predicates and lack of saturation will often incorporate individuals (e.g. the city of London). For predefined units, our criteria has been to work from the prototypical denotational patterns, i.e. that head nouns and verbs denote categories and that full NPs and clauses denote individuals. When argument and adjunct phrases are composed onto a head line, if they are more attribute like, rather than more classifying, then to that extent we are more likely to declare the denotation of the new phrase to be a partially saturated individual rather than a derived category.

Thus the word "*month*" is taken to denote a category, even though it has particular attributes such as being part of a year and being composed of between 28 and 31 days. "*December*" denotes an individual since it is realized as a proper name and saturates the logically (though not syntactically) apparent variables of months, i.e. the name, the position within the year, and the specific number of days.

Recall the denotation of the verb phrase in this example ("*[George L. Ball] resigned yesterday as chairman and chief executive officer of Prudential-Bache Securities Inc*"). It is treated here as a partially saturated individual, but would it be useful to think of it as a kind of thing? What about "*retire from IBM*"? Is that a sufficiently frequent event that a reasoning system would want to give it the status of a category? These are questions of knowledge engineering and design. My criterion when such units are being derived automatically is that any phrase that is syntactically analyzed at less than a maximal X-bar level is treated as a derived category if it occurs more than once, and otherwise as a partially saturated relation. As we will see, the mechanics of these two kinds of unit are sufficiently similar that the shift from one to the other is not large.

This section has introduced KRISP's set of epistemological types and illustrated some of their correspondences to particular kinds of words or phrases. However, the denotation of a phrase in isolation explains only part of the process of semantic interpretation. In the next sections we will look at interactions between words and other constituent types, illustrating the kinds of protocols at work as phrases are composed and the denotations of the resulting edges are computed rule by rule in step with the syntactic processes.

## 5. Object-Creating Forms

Let us move now to looking at KRISP'S equivalent of a data base tuple-creating SQL expression and see some examples of the kind of things a person does when populating a KRISP-based model with objects by hand. Here, for example, is the expression that creates the category 'resign-position'.

```
(define-category  resign-position
  :specializes  leave-position
  :instantiates  job-event
  :binds ((person . person)
          (position . position))
  :realization
     (:tree-family intransitive
      :mapping ((s . job-event)
                (vg . :self)
                (np/subject . person)
                (agent . person))
      :main-verb "resign"
      :additional-rules
        ((:pp-adjunct
           (s (s as-title)
              :head left-referent
              :binds (position  right-referent))))))
```

This is an expression in a domain-level language implemented in Lisp. When it is executed it returns a unit of type category with the indicated properties. If such a unit already exists it is accessed and returned rather than another unit being created. This particular category is fit into the lattice as a subcategory of leave-position. It gets an annotation to the effect that for purposes of subsequent reference to individuals of this type in the text they should be classified in terms of a higher category in the lattice,

job-event and instances of it should be recorded as such in SPARSER's discourse history (i.e. leaving and entering a position are considered things of the same kind for that purpose). Two variables are defined, named 'person' and 'position', and it is indicated that they are restricted to be bound only to individuals whose types include categories that happen to have those same names.

The other field of the expression, realization[9], takes us to a aspect of KRISP that we have not yet touched on—one that provides the basis for KRISP as a reversible representation that can be used equally well as the source of of the production of texts or the target of their comprehension. This field specifies how information belonging to this category (individuals of that type) is expressed in English. Furthermore, the schematization of the information in that field is such that it can be used either by either the parsing machinery in SPARSER or the surface structure creating machinery in Mumble-86, the companion system to SPARSER in the bi-directional system I have been developing (see McDonald 1998).

The key to its reversibility is the fact that this specification is based on the use of grammatical constructs that are available in both systems: the tree families of a Tree Adjoining Grammar (:TAG") (Joshi 1985, Abeille 1988). Differences in notation aside, the nature of the tree families within the Mumble generator is virtually the same as in a standard TAG and thus need not be illustrated here. Their nature inside SPARSER is another matter, since there they are reconstructed to suit the needs of an efficient parser. (Full details are given in McDonald (1993).) Here is the expression that creates the 'exploded tree family' for trees headed by intransitive verbs like *retire*.

```
(define-exploded-tree-family  intransitive
  :binding-parameters ( agent )
  :labels ( s vg np/subject )
  :cases
     ((:subject (s (np/subject vg)
                   :head right-edge
                   :binds (agent left-edge)))))
```

This expression creates a mapping template that is applied to the information in the definition of 'resign-position' to create a set of phrase

---

9 'Realization', a term adopted from Systemic Grammar, means here the mapping of a body of information in one form, e.g. values in a system network or in a knowledge base, into another form with different epistemological properties, possible adding or dropping ancillary information along the way according to the differences in the representations used.

structure rules in SPARSER's semantic grammar, listed here in abbreviated form.

```
#<PSR1243  job-event ->  job-event as-title>
#<PSR1347  job-event ->  person resign-position>
#<PSR1343  resign-position ->  "resign">
#<PSR1344  resign-position ->  "resigns">
#<PSR1345  resign-position ->  "resigned">
#<PSR1346  resign-position ->  "resigning">
```

Let us look at the expression that would have created the second of these rules had we written it by hand. This will give us a sense of how KRISP's facilities are used in SPARSER's semantic interpretation process.

```
(define-cfr job-event (person resign-position)
  :form s
  :referent (:head right-edge
             :binds (person left-edge)))
```

As its base, SPARSER is a chart parser that creates a set of edges (parse nodes) bottom up in one pass from left to right through the text. All of its edges are binary, and are formed through the operation of a relatively complex control structure that examines selected adjacent pairs of edges, one to the 'left' and one to the 'right'. When a match is found (as in "*George L. Ball resigned*", which is parsed as the pair of edges semantically labeled 'person' and 'resign-position'), a new edge is formed; it is given the appropriate semantic label ('job-event') and syntactic label ('s'); and its denotation (referent) is constructed. In this case the rule dictates that the referent is to be formed by taking the unit already established as the referent of the head constituent (resign-position, the edge to the right) and adding a binding that assigns the referent of the left-edge (person) to the person variable of the resign-position category. The result is this partially saturated individual:

```
resign-position 13
   :type (resign-position past)
   :binds  #<person = #<Ball, George l>>
   :open-in (position)
```

These examples have provided a short look at the machinery that handles the construction of new units. This is only part of the picture however. Any realistic language comprehension system starts with a significant amount of knowledge already built in. To facilitate this, KRISP includes a language for creating individuals and bindings as well as

categories and variables. Here, for example, is an expression that creates a individual to represent a particular country, with the side-effect of also adding a rule to the grammar that has a direct pointer to this unit so that the referent of the chart edge can be supplied without needing to do any lookup or object construction. In this analysis, the category 'country' has only one logically required variable, 'name', restricted to an object of type name; the name object (an individual) is automatically created as part of the evaluation of this expression. (The fact stated here is of course no longer accurate; however the work in which this individual was used predates the change in geo-political status.)

```
(define-individual country
  :name "Hong Kong" )
```

Creating a partially saturated individual is a bit more elaborate. We have to specify the category of the eventual individual and the variable that this unit (a "psi" for short) is to bind. Here is the definition-schema for the class of psi that are percentages.

```
(define-dependent-term percentage
  :specializes measurement
  :binds ((value . number))
  :governing-relation (amount measurement)
  :realization
       (:tree-family quantity+kind
        :mapping ((np . self)
                  (np-head . ("percent" "%"))
                  (modifier . number)
                  (quantity . value)
                  (result-type . self))
```

## 6. An Extended Example

We can now walk through a relatively complex example where the use of the special facilities of KRISP provides a significant benefit. It will also let us review the basic correspondences between the different types of KRISP units and particular linguistic constructs. We will use the following sentence, which is taken from a short article from the Kyoto News Service about a joint venture by the Yoshinoya D & C Co. to set up another of its noodle restaurant chains in Hong Kong.

> *"Hong Kong will be Yoshinoya's third overseas market following 11 outlets in Taiwan and 44 restaurants in the United States."*

To make things more realistic, we will imagine that SPARSER does not have any knowledge about the last three head nouns: *market*, *outlets*, and *restaurants*. No language comprehension system today should ever expect to have all the vocabulary it will encounter in its target corpus, and must have facilities for making some sense out of unknown words, especially when they occur in a well-understood context as they do here. It will be as though the text was this:

> "*Hong Kong will be Yoshinoya's third overseas xxx following 11 yyys in Taiwan and 44 zzzs in the United States*."

## 6.1 Simple semantic interpretation

To SPARSER's lexicalized semantic grammar, the example text looks like the following sequence of preterminal categories:

```
country + "will" + be + country-possessive + ordinal
+   location + unknown + "following" + number +
unknown/plural + "in" + country + "and" + number +
unknown/plural + "in" + "the" + country
```

The words which have a predefined interpretation in KRISP, such as we saw above for the example of "*Hong Kong*", appear in the chart as edges labeled with the category that they have in KRISP. Function words appear as themselves. Words that are outside of SPARSER's (initial) vocabulary are given the label 'unknown', and have annotations that indicate their morphological or orthographic properties (glossed here as part of the name).

Starting at the beginning of the text with 'country', the parsing process proceeds incrementally to the right, where the adjacency of the modal "*will*" indicates that the verb group has started and that Hong Kong must, correspondingly, be the sentence's subject. "*Will*" denotes the category 'future-time' in our analysis. It combines with any verb or verb group to its right, forming a new edge over those two constituents with the same label as its right neighbor, and imposing the constraint that the denotation of the result include future-time in its domain-type. Here is the expression in SPARSER that creates this rule.

```
(define-form-rule ("will" verb)
  :form verb-group
  :referent (:head right-edge
             :subtype future-time))
```

This is a rule that mixes syntactic and lexical labels (or in other instances, semantic labels). It specifies that the word "*will*" may combine with an edge of any semantic category to its right so long as that edge has the syntactic ('form') label of 'verb'. The result is a new edge with the same label as the semantic label on edge indicated as the head. The 'subtype' directive adds a category to the type field of the unit denoted by the head edge. Note that if this unit is a category (as it will in this instance since the edge to the right is semantically labeled 'be', subsuming differences in the tense of the verb 'to be') and if it denotes a category that is underspecified (in the sense of Hobbs et. al 1993), then what we are doing is creating or looking up a derived category. Otherwise we would be adding the category to the type field of an individual or partially saturated individual.

Proceeding onward, a low-level scan will delimit the next constituent after 'be', identifying it as a noun phrase since it definitively starts with the possessive company edge, "*Yoshinoya's*", and ends just before the subordinate conjunction "*following*". That is enough to infer that the word "*market*" must be a noun; and since nouns denote categories, we can construct a placeholder category to act as its denotation, giving it the same name as the word. Of course we know nothing about this new category's properties, but we can (1) link all subsequent instances of this word to this same category, and (2) accrue some of its properties by seeing how it is modified by the other words it is in construction with. The chart thus gets an edge over the word "*market*' with 'market' as its semantic label, 'head-noun' as its syntactic label, and the new category as its referent.

The preceding three constituents within the noun phrase ("*Yoshinoya's third overseas* ___") can be combined with 'market' using default rules of form such as the one shown earlier for "*will*". The interesting question is what that combination denotes. Overall the phrase is a description of Hong Kong, the same sort of relationship as if we had said "*Hong Kong is a former British colony*". Just what form this relationship takes in KRISP depends on the nature of the description. Saying that *Hong Kong is an island* is definitively assigning it to a category, which would corresponding to adding the category 'island' to the individual representing Hong Kong, similarly for 'former British Colony'. These ascriptions indicate properties

of the individual that dictate how it should behave as an object subject to inferences (e.g. its residents are probably bilingual in English), and as such should be part of its type field as in any object-oriented system.

On the other hand, what is going on if we read just that "*Hong Kong is third*" (or 'overseas' or 'Yoshinoya's')? These are obviously incomplete statements, and they are placing Hong Kong in some sort of relationship where the type of relation is determined by the nature of the missing element ('member of a set', 'geographically separated by an ocean', 'owned by'). In KRISP this corresponds to binding the individual representing Hong Kong to some variable.

As terms in SPARSER's grammar and model, the meanings of the three known constituents ('third", 'overseas', 'Yoshinoya's') have representations in KRISP. They are partially saturated individuals, and as such they specify the category (domain type) of the individual they would be part of, the variable(s) in it that they bind, and their value(s), and the variables in that category that remain open, notably including the value restrictions on those variables. Using this machinery, SPARSER is able to construct a denotation for the phrase as a whole that is sufficiently 'loose' that all its implicit relationships with the rest of the text can be instantiated using relatively simple compositional mechanisms. We will focus here on what happens with the interpretation of "*third*". The other two constituents are handled in the same way.

The ordinal third identifies a position within a sequence and indicates that the thing that it modifies holds that position. Here is its definition, along with the definitions for sequences and for the property of being a member of a sequence. Note that 'ordinal' is a schema that defines the denotations of the whole class of ordinals. The actual instances are defined by substituting a specific word for 'self'. Obviously only a few are predefined, just those where the number involved has a name that cannot be evaluated compositionally (i.e. twelfth, billionth, etc., but not twenty second). The others are defined on the fly as needed. As they have so many variations, the realizations of the categories are omitted to save space.

```
(define-dependent-term ordinal
  :binds ((value . number))
  :governing-relation (member-of-a-sequence position)
  :realization (:adjective . self))

(define-category member-of-a-sequence
  :binds ((position . ordinal)
          (sequence . sequence)
          (item . anything)))

(define-category sequence
  :specializes collection
  :binds ((members . collection)
          (count . number)
          (type . category)))
```

At present, the rules that specify the grammatical properties of partially saturated individuals like the dependent term "*third*" are written by hand since I have yet to devise a comfortable notation by which to derive them as a side-effect of the evaluation of their defining forms as is done with categories. This is not much of a burden, since under a TAG analysis all compositional adjectives and adverbs are auxiliary trees. To the parser, these correspond to simple Chomsky adjunctions and so are naturally captured as form rules:

```
(def-form-rule  ( ordinal (n-bar np-head) )
   :form  n-bar ;; This is the syntactic label on the resulting edge.
                ;; Its semantic label is a copy of the label on the head.
   :referent (:head right-edge
              :subtype member-of-a-sequence
              :bind (position left-edge))
```

This rule says that whatever else the unit to the right of the ordinal may be, it is also a member of a sequence, and that with respect to that sequence it is the third item. (In the first line of the expression, the parentheses within the righthand side of the rule are a shorthand for defining two rules.)

Note that the effect of this rule has been to attribute a property to the unknown word "*market*", namely that it denotes something that can be a member of a sequence. The composition of the other two premodifiers make similar attributions, and the final result is this unit.

```
market-1
  :type (#<market>
         #<owned-by-a-company>
         #<member-of-a-sequence>
         #<has-a-location>)
  :bindings (#<relative-location = #<overseas>>
             #<position-in-sequence = #<third>>
             #<owning-company = #<Yoshinoya D & C>>)
  :open-in (#<variable sequence>
            #<variable reference-location>)
```

Moving on to the rest of the sentence, a similar process will create denotations for "*11 outlets in Taiwan*" and "*44 restaurants in the United States*", attributing to the stand-in categories for the two head nouns (which you may recall we have supposed to be unknown words for the purpose of this example) the categories 'collection' and 'located-in-a-country'. Note that since neither the numbers or the locations are dependent terms, the denotations for these two phrases will be ordinary (saturated) individuals rather than partially saturated individuals. Conjoined together, these two units form a collection, and when composed with "*following*" that collection is specialized to be a sequence.

## 6.2 Semantic composition via interior open variables

At this point in our exposition we have laid out enough of the representational machinery that KRISP provides to illustrate how it may enable us to break through the glass ceiling of the standard framework for semantic interpretation that we sketched in §2.3. Within that framework, the construction of denotations is done by rule-by-rule (syntax-driven) composition. Working with KRISP we are also syntax-driven, but rather than restricting ourselves to operations over patterns of atomic labels, we are working with structured objects and can allow elements of those objects to have interactions of their own. This moves much of the work of forming an interpretation into the semantics, and relieves the syntax of the need to account for every last possible pattern of constituents. In particular, we allow for functionally driven compositions from the interior of constituents according to the dictates of any unbound variables they may include.

Consider the interaction here of the noun phrase and the gerundive adjunct that follows it (" *Yoshinoya's third overseas market following 11 outlets in Taiwan and 44 restaurants in the United States*"). We permit them to combine syntactically through a default form rule to the effect that any postnominal qualifier can Chomsky adjoin to the NP proceeding it. As

always, we would expect the combination of any modifier and head to involve binding a variable (if that cannot happen then we have a displaced constituent). Default rules, however, cannot in themselves define bindings, since the head unit that would supply the category that contributes the variables to be bound is by definition anonymous.

To address these cases, we have loosened the compositional process so that we can look for the possibility of bindings that are sanctioned by any open variables in the denotations of one or the other constituent, using as our criteria just the domain types specified in the variables' value restrictions. In this instance, the noun phrase is open in a sequence, and the adjunct is just such a sequence. If we gloss the unit denoted by the noun phrase as market-1, which includes the relevant binding b-1 as shown in detail below, and gloss the sequence denoted by the adjunct as Taiwan&U.S.-1, then the composition of the noun phrase and the adjunct results in the binding designated b-2:

```
binding b-1
  :variable #<position>
  :value #<individual ordinal third>
  :body #<market-1>   ; qua its category, 'member of a sequence'

binding b-2
  :variable #<sequence>
  :value #<Taiwan&U.S.-1>
  :body #<market-1>
```

Now let us look at what can be done with the last grammatical relationship in this sentence with a denotation yet to be accounted for, the combination of Hong Kong with its predicate nominal (market-1) via the underspecified relation 'be'. Again, we would not like to burden the grammar with enumerating every possible category that might be used to describe a city (though some of them, such as population, location, or governance are obvious candidates if one was going to develop a domain model for cities), and we certainly would not want to presume that we have to anticipate that a city can be 'a place where a company can market its products' in order to have an adequate information extraction grammar for joint ventures. Anything that we can get through a general rule operating opportunistically through the objects in the semantic model is much to be preferred.

In this case, we can do two things. First we can apply a general rule that given an individual as the denotation of the subject and a derived

category as the predicate (the type field of market-1), the verb 'be' imposes a substitution relationship between the two. We can then look for what compositions can be made given the type constraints involved (read: additions to #<Hong Kong>'s type or to bindings it can be part of). In addition, we can 'distribute' the categories of the predicate and see if there are any rules already in the grammar that specify how they compose.

Following the first procedure, we see that #<market> is just a category that we constructed on the fly, so all that can be done there is to add it to #<Hong Kong> as another category in its type field. #<Owned-by-a-company> is predefined, but in SPARSER's grammar the only thing that companies can own is other companies so the value restriction precludes any compositions there. Composition with #<overseas> (by way of a rule that combines cities and locations, cf. "*Boston is in Massachusetts*") yields a partially saturated individual since we don't know what place Hong Kong is being taken to be overseas with respect to. (From the larger context we, as educated people, know it is Japan, since Yoshinoya is a Japanese company and the East and South China Seas lie between the two locations; but it is asking a lot of a system designer to attempt to formalize that inference today in any general way.)

In the case of #<category member-of-a-sequence> we can do a lot. Part of the (presently hand-written) grammar of sequences is the rule:

```
(define-form-rule ( np is-ordinal )
  :form s
  :referent (:head right-edge
             :binds (item left-edge)))
```

The predicate (market-1) already has values for the other two variables of the member-of-a-sequence relation, and the application of this rule binds the third, yielding the individual:

```
member-of-a-sequence 1
  :type ( member-of-a-sequence )
  :bindings (#< position = #<third> >
             #< sequence = #<Taiwan&U.S.-1> >
             #< item = #<Hong Kong> > ))
```

This ends our extended example. To summarize the rules of semantic interpretation we have used, the most basic was a direct link between a known word or phrase ("*Hong Kong*", "*third*") and a KRISP unit that was

brought into existence when the model was initially populated.[10] The second kind of rule was specified as part of the syntactic rules for combining adjacent, semantically labeled constituents, typically as head and argument. These interpretation rules add bindings or subtypes to the interpretation of the head, interpretations that are then passed up the headline of the syntactic analysis for further compositions as the syntactic analysis extends.

The final kind of rule is more heuristic and has yet to be properly evaluated through testing on a large corpus. This is to apply general rules of phrasal composition that use the syntactic labels on the constituents and to then look for combinations among the constituents' referents opportunistically on the basis of the semantic types of the constituents and the variables that are open in any partially saturated individuals, the most unusual of the epistemological types supported in KRISP.

## 7. Principles

KRISP is fundamentally a semantic network. It is part of the intellectual tradition that began with Quillian (1968) and continues with the 'KL-One' family of languages for knowledge representation (see, e.g., Brachman & Schmolze 1985, MacGregor 1991). Like them, it permits the representation of of actual individuals and the complex categories that describe them, and entwines both in a net of structured links that enable a reasoning system to efficiently draw conclusions about their properties by following this already built roadmap.

KRISP makes several specific improvements on its predecessors, improvements designed to expedite the only kind of reasoning to which it has so far been applied: the dynamic construction of a model of the information in the text. We can summarize these improvements in terms of three general principles or desiderata that should govern the development of any modern representation. Adhering to these principles facilitates any attempts to achieve the criteria set out at the beginning of this paper: application to real texts, rapid, efficient processing, and reversibility. This is the first:

---

[10] In these cases, the model has more of the trappings of a knowledge base than the interpretation of a text, but this is a necessary state of affairs, since for an information extraction system to have any hope of comprehending real texts it must bring a lot to the table in terms of known individuals and facts before the process even begins.

**The representation should consist of an inter-connected set of relatively small units.**

The smaller the unit, the less often it will have to be torn down and reassembled when we need to express the same information from a different perspective. Consider the example from the last section. It is redundant to have categories for both 'sequence' and 'member of a sequence' since the one relation can be logically derived from the other. Nevertheless, English provides phrasings for both, which means that having independent semantic projections for both will make for direct, quick interpretations. Given the interconnection of KRISP's units and a simple set of copying rules it is virtually no effort to flesh-out one form given the other.[11]

More to the point, suppose we had only implemented 'member of a sequence' but latter wanted a companion language generation system to say something like "*Yoshinoya has three markets overseas: Taiwan, the United States, and Hong Kong*". To construct that final conjunction we would have to unpack the three member-of-a-sequence relationships and create a new unit (at least in the state of the generator), whereas if both forms are maintained we get the conjunction as an already packaged unit. Another consideration relevant to this example brings us to the second principle:

**There should be a first-class object type in the representation for every class of syntactic category in the language.**

One of the ways of looking at the notion of a first class object is that it provides a representational 'term'—something that another term can stand in relation to. Designing the model to supply both a unit for the group and a unit for each company's membership in the group allows those facts to be reasoned about separately (e.g. for making summaries about the breadth of international markets with the group; summarizing with other investments by Yoshinoya with the membership relation; etc.).

Making the classic 'concept–slot–value' of KL-One a first class object (a binding) simplifies many different activities that in other kinds of representations are problematic or even impossible. For example we can read the relation out (traverse the network) from either direction. If Yoshinoya 'has a market in' the U.S., then we can alternatively start from the unit for the U.S. and ask what foreign companies have markets there. A

---

[11] These copying rules are effectively meaning postulates. They have been only sporadically implemented and are ad hoc at the time this is written.

generator can formulate the text "*The U.S. is the second overseas market for Yoshinoya*" starting with the very same units that were formed when the text with the alternative focus was parsed.

The rule of thumb has been to package no more information into a single unit (binding, category) than corresponds to a single word or a simple one-argument relation. This maximizes the possibilities for rearranging sets of units to best fit the discourse context given what is redundant or salient at the moment. The other side of this coin, however, is to ensure that when we have a particular body of information with an ongoing identity over time or space—something that entails more than one unit to represent it—we must ensure that it retains this identity regardless of how its component units come to be rearranged in the course of linguistic processing. This takes us to our third principle:

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

This is the 'uniqueness principle' that was first articulated in the context of the SNePS representational system (e.g. Maida & Shapiro 1982). Some of its consequences are obvious and could be glossed as the requirement to have a 'canonical form' for the comprehension system's output. The individual we get from analyzing "*The new firm will be 50 percent owned by Takarabune*" should also be returned for "*Takarabune will own 50 percent of the shares in the new firm*" or even for "*Takarabune's share of the firm*" should that phrase appear later in the same text—the very same object, not just the same description ('eq' rather than 'equal' in Lisp).

Adherence to this principle has important consequence for the efficiency of the generation process. A generator must always appreciate when it is realizing a second instance of an entity it has already mentioned so that it can deploy the appropriate pronoun or reduced form. By far the easiest way to do this is to notice that the representational entities for the two instances are the identical unit, rather than attempting to match descriptive expressions. Comparing two graphs for equality (which is what matching descriptions comes down to) is a procedure whose time complexity includes the size of the description as one of its factors. It is far better to take up that factor just once when the initial text for the individual is being indexed as it is parsed (at which time it constitutes just a slight increment on the N plus log N factor that is already at play within the parsing process) than to have to experience it time and time again.

A more subtle consequence of this principle comes in an area we have not even touched on in this paper: the machinery that is needed to ensure that the uniqueness principle is maintained as a matter of intrinsic design rather than deliberate programming. Briefly, the standard taxonomic lattice that organizes the categories and their variables according to sub- and super-classes is extended 'below' the usual level of detail to incorporate a lattice of all of the combinations of bound and unbound variables that could appear in partially saturated individuals. This provides an indexing structure against which to lay out all of the individuals in the model: those that have been predefined and those that are created during the process of comprehending a text, saturated and unsaturated. During a parse, the categories and bindings of each individual are built up gradually unit by unit and indexed to the corresponding nodes of the 'partial saturation' lattice. This incremental indexing process shadows the syntactic construction of a parse as argument and adjunct constituents are added to a head.

The lattice is so constructed that bindings made in different orders will all lead to the same node for any given combination of variables. Thus a simple collection of the variables and their values node by node suffices to provide a conduit to a common, single representation of any individual or partial individual that the system has ever been informed about.

## 8. Concluding remarks

The design for KRISP has been developed incrementally during the last several years and is still evolving. Its fundamental motivations stem from investigations over the course of the last decades to try and establish just why it was that other representational systems were always turning out to be awkward when used as the source for the generation of fluent prose.

As a practical system in a working parser, KRISP has undergone a goodly amount of revision as new problems in text were tackled. The partially-saturated-individual 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 the new type was introduced, it widened the design space available for solving semantic problems in these texts, and lead to cleaner treatments of long-standing problems.

Whenever there has been a question about what direction the design should take, the choice of analysis has always been made 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 a typed lambda calculus, with the insight that the binding of a variable to a value should be given a first class representation.

The evolution of KRISP will continue as improved treatments and alternative analyses are developed. The mechanics of a hook up between SPARSER and my generation systems has just recently begun, and this is likely to prompt many adjustments in the design, small and large, as issues in the combined efficiency of the two very different processes accessing a single representation are considered and solutions found.

One of the problems for generation systems has always been how to assemble a description of how constructions are used in different genres of text—descriptions that could then be used to produce texts that are true to one or another style as required. It is likely that a combination of the indexing structures developed to implement the uniqueness principle and an annotation of the kinds of syntactic constructions used when the component units of an individual are assembled and indexed could lead to just such a 'style-recording' mechanism as a simple side-effect of just having the parser read the text. Should this prove to be true, it will be a significant validation of the design of KRISP as a whole.

## 9. References


Abeille, Anne (1988) "A French Tree Adjoining Grammar", technical report, Dept. of Computer & Information Science, University of Pennsylvania.

Appelt, Doug; Jerry Hobbs, John Bear, David Israel, Megumi Kameyama, Andy Keller, David Martin, Karen Myers, Mabry Tyson (1995) "SRI International FASTUS System MUC-6 Test Results and Analysis" in DARPA 1995, 237-248.

DARPA (1993) the Proceedings of the Tipster Text Program (Phase I), September 1993, available from Morgan Kaufmann Publishers, San Francisco.

DARPA (1995) the Proceedings of the Sixth Message Understanding Conference (MUC-6), available from Morgan Kaufmann Publishers, San Francisco.

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

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

Bunt, Harry & Masaru Tomita (eds.) (1996) *Recent Advances in Parsing Technology*, Kluwer Academic Publishers, Dordrecht.

Hobbs, Jerry, M. Stickel, Paul Martin, D. Edwards (1993) "Interpretation as Abduction" *Artificial Intelligence* 63: 341-385.

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

Joshi, Aravind K. (1985) How much context-sensitivity is required to provide reasonable structural descriptions: tree adjoining grammars. in Dowty et al. (eds.) *Natural Language Processing*, Cambridge University Press.

MacGregor, 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**, pp. 291-330

McDonald, David D. (1993), "Reversible NLP by Deriving the Grammars from the Knowledge Base", in Strzalkowski, ed., *Reversible Grammar in Natural Language Processing*, Kluwer Academic, in press

_____ (1992) "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.

_____ (1996) "Internal and External Evidence in the Identification and Semantic Categorization of Proper names", in Boguraev & Pustejovsky (eds.) *Corpus Processing for Lexical Acquisition*, MIT Press, Cambridge, pp. 21-39.

_____ (1998) "Controlled Realization of Complex Objects by Reversing the Output of a Parser", in the Proceedings of the 9th International Conference on Natural Language Generation, St. Cathrines, Ontario, Canada.

_____ (1988) Marie Meteer & James Pustejovsky "Factors Contributing to Efficiency in Natural Language Generation", in Kempen (ed.) *Natural Language Generation: Recent Advances in Artificial Intelligence, Psychology, and Linguistics*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1987, pp. 159-181.

Mellish, C. S. (1985) *Computer Interpretation of Natural Language Descriptions*, Ellis Horwood, Chichester, England

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.

Partee, Barbara H. (1984) "Compositionality", in Landman & Veltman (eds.) *Varieties of Formal Semantics*, Foris, Dordrecht, 281-312.

Quillian, M. R. (1968) "Semantic Memory", in Minsky (ed.) *Semantic Information Processing*, MIT Press, Cambridge.

Schubert, Lenhart and Francis J. Pelletier (1989) "Generically Speaking, or, Using Discourse Representation Theory to Interpret Generics", in Chierchia, Partee & Turner (eds.) *Properties, Types and Meaning, vol. II: Semantic Issues*, Kluwer Academic, Dordrecht,

Searle, John (1992) *The Rediscovery of the Mind*, MIT Press, Cambridge.

Smith, Brian C. (1996) *On the Origin of Objects*, MIT Press, Cambridge.

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

Steedman, Mark (1987 "Combinatory Grammars and Parsitic Gaps" *Natural Language & Linguistic Theory* 5, 403-439.

_____ (1996) *Surface Structure and Interpretation*, MIT Press, Cambridge.

Strzalkowski, Tomek (ed.) (1994) *Reversible Grammar in Natural Language Processing*, Kluwer Academic Publishers, Dordrecht.

Sundheim, Beth (1995) "Overview of Results of the MUC-6 Evaluation", in DARPA 1995, 13-32.

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

Winograd, Terry (1972) *Understanding Natural Language*, Academic Press, New York.

Woods, William A. (1973) "An Experimental Parsing System for Transition Network Grammars", in Rustin (ed.) *Natural Language Processing*, Algorithmics Press, New York, pp. 111-154