A LINGUISTIC APPROACH TO THE PROBLEM OF SLOT SEMANTICS

H. Van Dyke Parunak
Industrial Technology Institute
Ann Arbor, MI

ABSTRACT

Most frame-based knowledge representation (KR) systems have two strange features. First, the concepts represented by the nodes are nouns rather than verbs. Verbal ideas tend to appear mostly in describing roles or slots. Thus the systems are *asymmetric.* Second, and more seriously, the slot names on frames are arbitrary and not defined in the system. Usually no metasystem is given to account for them. Thus the systems are *not closed.*

Both these features can be avoided by structures inspired by case-based linguistic theories. The basic ideas are that an ontology consists of separate, parallel lattices of verbal and nominal concepts, and that the slots of concepts in each lattice are defined by reference to the concepts in the other lattice. Slots of verbal concepts are derived from cases, and restricted by nominal concepts. Slots of nominal concepts include *conducts* (verbal concepts) and derivatives of the slots of verbal concepts.

Our objective in this paper is not to define a new KR language, but to use input from the study of natural cognition (case grammar) to refine technology for artificial cognition.

TERMINOLOGY AND NOTATION

Concepts are predicates over instances (Hayes 1979), and are named with a prefixed "C-". Variables over concepts are lower-case letters near 'n' (a mnemonic for "intension"), while variables over instances ("extensions") are lower-case letters near 'x'. $x \epsilon m$ means "x is an instance of concept m." This paper is limited to the problem of defining concepts, and does not discuss how to make assertions about them (Woods 1975).

The links between concepts fall into two general categories, depending on whether or not they indicate subsumption (Brachman 1983). One concept C-1 *subsumes* another C-2 just when $\forall x.(C\text{-}2\ x) \rightarrow (C\text{-}1\ x)$. A non-subsuming link from one concept to another is a *slot.*

A slot is a two-place predicate over instances, and links two concepts, its *parent* (the concept to which it belongs) and its *restriction.* For the application of a slot to two instances to be true, it is necessary (but not sufficient) for the first argument to be an instance of the slot's parent, and the second to be an instance of its restriction. Slot names begin with "S-". To identify a slot's parent in cases of ambiguity, postfix the name of the parent concept, omitting the prefixed "C-". Thus the slot recording a lathe's workpiece is S-Workpiece.Lathe.

A functional notation (cf. Vilain 1985) describes manipulations of concepts and slots. For example, **CMeet** combines two concepts to form a concept that meets the restrictions of them both. (CMeet C-1 C-2) has the semantics $\lambda x.(C\text{-}1\ x)\ \&\ (C\text{-}2\ x)$. For example, C-Son = (CMeet C-Child C-Male). **CRestrict** builds a new concept from an old one by restricting the eligible slot-fillers for some slot of the older concept. (CRestrict C-1 S-1 C-2) has the semantics $\lambda x.(C\text{-}1\ x)\&\forall y.(S\text{-}1\ x\ y) \rightarrow (C\text{-}2\ y)$. If C-Machine has a slot S-Status, C-AvailableMachine = (CRestrict C-Machine S-Status C-Idle) describes exactly those machines that are idle. **CMeet** and **CRestrict**, though not a complete set, suffice to illustrate the ideas in this paper. See Vilain 1985 for a fuller set.

THE PROBLEM OF LACK OF CLOSURE

The main problem addressed by this paper is that the semantics of common frame formalisms lack closure both within individual frames and between frames.

Intra-Frame Closure

In

```
C-Lathe
  S-Workpiece
  S-SerialNumber
  S-Location
  S-Size
```

the various slots stand in different relations to C-Lathe. The filler of S-Workpiece changes as the lathe removes chips of metal. The filler of S-Location can change, too, but as a result of reorganizing the factory, not of the normal operation of the lathe. S-Location and S-Size are attributes of the lathe, yet they differ in their semantics from one another, and from S-SerialNumber.

The use of natural language names to identify slots is seductive, since it suggests that the structure has more meaning than the system can actually access. Formally, S-Workpiece has no more meaning than S-G10032. In common net formalisms, slot semantics are defined only implicitly through the inference code, in direct violation of the KR agenda of separating domain knowledge from processing strategy.

Inter-Frame Closure

A factory knowledge base will describe C-Workpiece as well as S-Workpiece, but the only connection available between these in many models is in pseudo-English slot names. S-Workpiece.Lathe should be more rigorously defined as the C-Workpiece associated with C-Lathe.

KRYPTON's role restrictions (Brachman *et al.* 1985) and KODIAK's MANIFEST operation (Wilensky 1984) address the inter-frame slot problem. For example, (MANIFEST C-Machine C-Tool) produces the concept "Tool-Of-Machine," so that C-

Machine gains a S-Tool slot pointing to the associated C-Tool. But the intra-frame closure problem remains. The same operation that produces *Tool-of-Machine = (MANIFEST C-Machine C-Tool)* also produces *Size-of-Machine = (MANIFEST C-Machine C-Size)* and *Action-of-Machine = (MANIFEST C-Machine C-Action).* Yet the relationship between C-Machine and its dependent concept in each of these cases is very different from the others.

TWO LINGUISTIC CONCEPTS

The linguistic concepts that suggest a solution to the closure problem are the distinction of nouns and verbs, and the grammatical theory of case.

The first concept simply observes that every known human language distinguishes nouns from verbs (Longacre 1976). Furthermore, natural languages universally distinguish three kinds of predication: statives (represented in English by adjectives and the verb "to be"), processes ("to become"), and actions ("to do," "to happen"). Any system that emulates human intelligence should mirror these distinctions.

The second concept, of verbal cases, is a sort of typing scheme for the arguments (nouns) of predicates (verbs) (Bruce 1975). Typical verbal cases include Agent, Object, Dative, Locative (From, To, At), Temporal, Material, Range, and Instrument. In the sentence "John hit the ball to Bill yesterday", "John" is the Agent, "the ball" is the Object, "Bill" is the Dative, and "yesterday" is Temporal. A restricted set of such cases (about 20 in most systems) suffices to define all the roles that nouns play toward verbs in a natural language. As a form of "slot" on low-level verbal concepts, cases were an inspiration for frames (Minsky 1974). This paper uses them to define slots for all concepts.

DEALING WITH ASYMMETRY

Early net systems treat nominal and verbal concepts symmetrically, with separate subsumption lattices for each (Quillian 1968; Hays 1973; Szolovits, Hawkinson, and Martin 1977). Modern net and hybrid formalisms are overbearingly noun-oriented in their representation of concepts. The ultimate root of modern subsumption lattices is typically "thing," forcing events to be nominalized if they are to be represented at all. Nodes representing events are typically named as gerunds, betraying their nominalization. KL-ONE and its descendents view concepts as playing "roles" instead of filling slots, further emphasizing that these concepts are construed as things.

Maintaining separate subsumption lattices for concepts derived from nouns and verbs respectively is consistent with the universal human distinction between such concepts. More pragmatically, it opens the way to define slot semantics within the system, by defining the slots in each lattice in terms of concepts in the other. Specifically, we let C-SummumGenus subsume C-Thing and C-Predication. C-Predication in turn subsumes C-Be for stative concepts, C-Become for process concepts, and C-Do for action concepts.

DEALING WITH LACK OF CLOSURE

Closing the definition of slots within a frame system requires addition of a third component, the *slot-maker*, to concepts and slots. In this section we discuss the general concept, then illustrate three kinds of slot-makers that we have found useful.

The Slot-Maker

A slot-maker maps two concepts to a slot, as does the MANIFEST operation in KODIAK (Wilensky 1984). The argument concepts are the parent and restriction of the slot, respectively. Some slot-makers require a third argument, defined below.

Having a slot-maker provides inter-frame closure. Having a set of slot-makers allows each to induce a distinct semantics on the slot created from its arguments, and thus provides intra-frame closure. So far, three kinds of slot-maker appear useful: one to define case slots, one to define conduct slots, and one to define component slots. In these descriptions, C-N(ominal) denotes some subset of C-Thing, and C-V(erbal) denotes some subset of C-Predication.

1. *(M-Case <case> C-V C-N)* defines slots of verbal concepts from nominal concepts, on the basis of linguistic cases. *C-V* is the parent of the slot, and *C-N* is its restriction. M-Case can be viewed as a set of slot-makers, one for each case. For example, (M-Case Agent C-Cut C-Thing) produces the slot that represents the agent of a cutting action as a thing. Each case has a distinctive semantics that it conveys to slots it defines.

2. *(M-Conduct C-N C-V <Slot of C-V>)* defines slots of nominal concepts from verbal concepts. *C-N* is the parent of the slot, and *C-V* is its restriction. For example, (M-Conduct C-Machine C-Cut S-Agent) produces the slot that describes a machine's cutting action. The third argument of M-Conduct tells what slot of the verbal concept the nominal concept occupies in performing the conduct. A nominal concept may have several conduct slots. Since stative predications of color and size are verbal concepts, (M-Conduct C-Machine C-Color S-Object) is the appropriate way to generate a slot to describe the color of a machine.

3. *(M-Part <aggregate concept> <component concept>)* describes a concept in terms of its component parts. The aggregate concept is the parent, and the component concept is the restriction. Nominal concepts can have nominal components, and verbal concepts can have verbal components.

Slot-makers close the universe of slots and concepts, but constitute a new component with respect to which the system remains open. The closure problem has moved up a level, not disappeared. Still, many problem domains need no more than M-Conduct, M-Part, and a dozen or so M-Case slot-makers. Few domains can be satisfied with this few

primitive slots. Also, the slot-maker concept lets systems reason explicitly about the relation of slots to concepts, something that traditional architectures do not allow.

Case Slots

Case slots offer a natural mechanism for recording the interaction between the nominal and verbal semilattices. The M-Case slot-maker defines them in the first place:

```
S-Agent.Do =  (M-Case Agent C-Do C-Thing)
S-Instrument.Do = (M-Case Instrument C-Do C-Thing)
S-Object.Do = (M-Case Object C-Do C-Thing)
```

Beginning with these slots of C-Do, C-Cut is defined as a subclass of C-Do whose agent is a machine, whose instrument is a tool, and whose object is a part:

```
C-Cut     = (CMeet
              (CRestrict C-Do S-Agent C-Machine)
              (CMeet
                (CRestrict C-Do S-Instrument C-Tool)
                (CRestrict C-Do S-Object C-Part)))
```

Conduct Slots

Just as case slots of a verbal concept have nominal restrictions, conduct slots of a nominal concept have verbal restrictions. For instance, the slot on C-Lathe that describes its intended action is

```
S-Action.Lathe = (M-Conduct C-Lathe C-Cut S-Agent)
```

That is, a lathe's action is the cutting of which it is the agent.

Since S-Action.Lathe represents a verbal concept, it has its own case slots, which can relate C-Lathe to other nominal concepts. A slot on a nominal concept C-1 that refers to an instance of another nominal concept can be derived from a case slot of some conduct of C-1. Nominal concepts can be related to one another only by way of some predication (or by another slot-maker, such as M-Part).

For example, the lathe's workpiece slot S-Workpiece.Lathe is defined in terms of the S-Object case slot in S-Action.Lathe.

```
S-Workpiece.Lathe =
  λ x y. ∃ z.
    (S-Action.Lathe x z) & (S-Object.Cut z y)
```

This definition captures the semantics that a lathe's workpiece (**y**) is the object of some cutting action (**z**) performed by the lathe (**x**), thus achieving intra-frame closure. In general, if S-1 is a conduct slot of nominal concept C-N derived from verbal concept C-V, and S-2 is a case slot of C-V, a slot S-3 of C-N with a restriction in the nominal lattice can be defined

```
S-3 = λ x y. ∃ z. (S-1.N x z) & (S-2.V z y)
```

The conduct slot generated for C-Lathe (S-Action) is derived from the frame for C-Cut.

For each slot in C-Cut, C-Lathe now has an associated slot. The slots in C-Cut, in turn, are generated from the slots of C-Do by restriction relative to concepts in the nominal semilattice. As the system grows, slot definitions tend to "zig-zag" back and forth between the nominal and verbal semilattices. Among other mechanisms for specialization, verbal concepts specialize by taking more restricted nominal concepts as case slots, while nominal concepts specialize by taking more restricted verbal concepts as conduct slots.

Examples of Component Slots

Component slots describe an entity as a component of a larger aggregate. Such aggregation can be either temporal or spatial. Typically, verbal concepts aggregate temporally to form more complex verbal concepts, and nominal concepts aggregate spatially to form more complex spatial concepts. Examples of languages that can be used to describe aggregates include Allen 1983 for temporal structures and Eastman 1973 for spatial ones.

As an example, consider a verbal aggregate, C-Deliver. Intuitively, a delivery takes place when someone receives something from one party and later gives it to another party. With C-Deliver defined in this way,

```
S-Reception.Deliver = (M-Part C-Deliver C-Receive)
```

That is, the slot S-Reception of C-Deliver takes an instance of the concept C-Receive that is a part of the concept C-Deliver.

This construction assumes the formalization of C-Deliver. It is produced by an *aggregation function* that maps component concepts into a composite concept. Given this aggregation function (call it AF-Deliver),

```
C-Deliver = (AF-Deliver C-Receive C-Give)
```

Then the construction

```
S-Reception.Deliver = (M-Part C-Deliver C-Receive)
```

simply names the reception that is already built into the delivery.

The specific aggregation function in question is:

```
AF-Deliver =
λ m n.
 λ x. ∃ r ∈ m, g ∈ n, y, z, w, v.

   (S-Agent r v) & (S-Dative g w) & (v <> w) &          ;0
   (S-Agent x y) & (S-Dative r y) & (S-Agent g y) &     ;1
   (S-Object x z) & (S-Object r z) & (S-Object g z) &   ;2
   (S-Dative x w) &                                     ;3
   (Before r g)                                         ;4
```

The outer lambda binds `m` and `n` to the arguments of AF-Deliver, which are intended to be concepts of receiving and of giving, respectively. When these are bound, the value of AF-Deliver is the inner lambda, which is a predicate on **x**. The existentially quantified

variables in this inner lambda represent an instance of receiving and giving (**r** and **g**, respectively), and four others used in clauses 1-4 to relate the concepts of the aggregate. Clause 0 insures that the source and destination of the delivery are not the same. Clause 1 constrains the agent of C-Deliver to be the same as the dative (recipient) of the component action of reception **r** and also the agent of the component action of giving **g**. Clause 2 identifies the object delivered with that both received and given by the deliverer. Clause 3 identifies the recipient of the delivery with the recipient of the component giving action. Clause 4 constrains the reception component of the delivery to take place before the giving component.

There are countably infinitely many aggregation functions, each typically useful for defining only one or a small family of composite concepts. AF-Deliver, for example, is only defined when its arguments are verbal concepts with case slots for datives, objects, and agents. The aggregation function makes the composition of an aggregate concept from its elements explicit, so that the M-Part slot-maker can name these components in relation to the aggregate.

CONCLUSION

Ontologies with separate subsumption lattices for nominal and verbal concepts permit the closure of slot semantics. A generalization of the linguistic notion of case permits definition of the slots in each branch of the ontology by reference to concepts in the other branch. Slots of verbal concepts are derived from linguistic cases, restricted by nominal concepts. Slots of nominal concepts include conduct slots restricted by verbal concepts; derivatives of the case slots of the verbal concepts defining a conduct slot, restricted by nominal concepts; and components of aggregate concepts.

The programs of the Industrial Technology Institute are partially supported by the W.K. Kellogg Foundation.

REFERENCES

Brachman, R.J. (1983). "What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks." *IEEE Computer* **16**, 30-36.

Brachman, R.J.; Fikes, R.E.; & Levesque, H.J. (1985). "KRYPTON: A Functional Approach to Knowledge Representation." In Brachman and Levesque, eds., *Readings in Knowledge Representation* (Los Altos: Morgan Kaufmann), 411-430.

Bruce, B. (1975). "Case Systems for Natural Language." *Artificial Intelligence* **6**, 327-360.

Hayes, P.J. (1979). "The Logic of Frames." in D. Metzing, ed., *Frame Conceptions and Text Understanding*, Berlin: Walter de Gruyter, 46-61.

Hays, D.G. (1973). "Types of Processes on Cognitive Networks." *International Conference on Computational Linguistics*, Pisa, Italy, 523-532.

Longacre, R.E. (1976). *An Anatomy of Speech Notions*. Lisse: Peter de Ridder.

Minsky, M. (1974). "A Framework for Representing Knowledge." MIT AI Memo No.

306.

Quillian, M.R. (1968). "Semantic Memory." in M. Minsky, *Semantic Information Processing*, Cambridge: MIT Press, 227-270.

Szolovits, P.; Hawkinson, L.B.; & Martin, W.A. (1977). "An Overview of OWL." MIT/LCS/TM-86, Cambridge, MA.

Vilain, M. (1985). "The Restricted Language Architecture of a Hybrid Representation System." *IJCAI-85* **9**, 547-551.

Wilensky, R. (1984). "Knowledge Representation--A Critique and A Proposal." *Proceedings of the First Annual Workshop on Theoretical Issues in Conceptual Information Processing*, Atlanta.

Woods, W.A. (1975). "What's in a Link: Foundations for Semantic Networks," in Bobrow, D.G. and A. Collins, *Representation and Understanding: Studies in Cognitive Science* (New York: Academic Press) 35-82.