Case Grammar:
A Linguistic Tool for Engineering Agent-Based Systems

Van Parunak
Center for Electronic Commerce
Industrial Technology Institute
van@iti.org, http://www.iti.org/~van

Abstract

Case grammar is a model widely used by field linguists for studying actual human languages. It is an extension to the semantic domain of the syntactic "case endings" used by many languages, such as German and ancient Greek and Latin, to mark the role of nouns in sentences (for example, as subject or object). Studies of a wide range of human languages suggest that all human languages share a common set of such semantic cases, though they may mark them in different ways. Thus cases represent a fundamental characteristic of human cognition that can be of great value in engineering systems that must interact with humans.

This paper surveys three agent-based systems in which ITI has used case grammar as an engineering tool at three different levels. It is drawn largely from the full papers referenced in the introduction below. These examples show that case grammar provides an integrating framework for constructing the internal knowledge representation system of a single agent, for identifying the agents that are active in a domain, and for defining an abstract space within which the behaviors of agents can be described.

Section 1 provides a simple introduction to case theory, and explains why it is relevant to engineering. Sections 2, 3, and 4 describe three systems in whose design case grammar has been an important tool.

1. Overview of Case Grammar

Case relations [Cook 1979; Bruce 1975] describe the relation between a verb and the other components (typically nouns) of a single proposition. In this section we outline the big ideas of case theory, then present a typical list of cases, and explain why a linguistic theory such as case grammar is important for engineering.

1.1. The Big Ideas of Case Grammar

The basic idea of case theory is that each verb has a set of named slots that can be filled by other items, typically nouns. Each slot describes the semantic role of its filler with respect to the verb. Verbs can be characterized largely by the slots that they support and selection restrictions on what can fill them.

The strong similarity between case frames as a general KR device in AI is not coincidental. Historically, Minsky's original idea of frames was inspired by case theory [Minsky 1981]. [Parunak 1989] develops a general-purpose frame language that makes full use of case theory.

The cases of case theory are deep or semantic cases. Unlike surface level, syntactic cases such as "subject" or "direct object," they do not change under grammatical transformation of the verb (for example, from active to passive), as illustrated in Table 1. "Boy" and "ball" change surface-level roles when the verb changes from active to passive voice, but their deep-level roles remain unchanged.

Table 1: Surface vs. Deep Cases

	Surface		Deep	
	boy	ball	boy	ball
The boy hit the ball.	subject	direct object	CAgent	Object
The ball was hit by the boy.	prep. object	subject	CAgent	Object

1.2. A Sample List of Cases

The following list of cases, strongly influenced by [Cook 1979], is illustrative. As demonstrated in Section 3, the set of cases adopted for a system will depend on the domain in which the system operates [Fillmore 1977], and constitutes a major design parameter for any real application.

Object.--Object is the most semantically neutral slot, identifying the entity whose position, existence, or state is under discussion. Every verb, even statives

such as "to be red" and process verbs such as "to become red," has an Object, the entity that (in these examples) is or becomes red, respectively, and Object is the only case that is present with every verb. In the case of action verbs, Object is sometimes called Patient, indicating that it undergoes the action of the verb.

CAgent.--The role of the CAgent is to cause the action of the verb. The standard term is "Agent," but the notion is narrower than the DAI use of "Agent," which may be acted upon and still be considered an agent. To avoid confusion, we designate the case-grammar role as Causal Agent or CAgent.

CAgents are characteristic of action verbs, in which an agent brings some process about. Such verbs always have both a CAgent and an Object ("The robot [CAgent] painted-red (verb) the ball [Object]"). In some cases, the Object and the CAgent are coreferential, and may be merged in the surface structure ("He ran" = "He [CAgent] moved (verb) himself [Object]").

Experiencer.--The Experiencer is an entity whose mental or emotional state is affected by the action of the verb. "Dave [CAgent] showed (verb) the DAI group [Experiencer] his program [Object]."

Beneficiary.--The Beneficiary is an entity that possesses an object or participates with a CAgent in a transfer of an object. "Robby [CAgent] brought (verb) Phil [Beneficiary] a drink [Object]."

Location.--The Location slot indexes the action of the verb positionally. "Opus [CAgent] put (verb) the bottles [Object] in the dumpster [Location]." In computerized applications, it is often filled by a reference to a coordinate system rather than an explicit object: "Opus [CAgent, Object] moved (verb) to coordinates (35,24) [Location]."

Time.--The Time slot indexes the action of the verb in time. "Odysseus [CAgent] will clean (verb) the lab [Object] this week [Time]." Like the Location slot, it is often filled by reference to a coordinate system: "Tom [CAgent] handed in (verb) his thesis [Object] at 4:59 PM [Time]."

Instrument.--The instrument is an entity that is used by the CAgent to bring about the action of the verb. "The robot [CAgent] sprayed (verb) paint [Object] on the car [Location] with Nozzle 34R [Instrument]."

Comitative.--The Comitative slot is filled by another entity (typically an agent) that is associated with the CAgent during the action of the verb, and provides a reasonable mechanism for modeling teamwork and cooperation. In a sufficiently general behavior, teamwork is more efficiently modeled as set-valued CAgent: "{Opus and Odysseus} [CAgent] cleaned (verb) the lab [Object]." At a more detailed level, the team members will be doing different things, and the Comitative slot is an efficient way to link their complementary behaviors: "Opus [CAgent] cleaned (verb) trash [Object] with Odysseus [Comitative]"; "Odysseus [CAgent] swept (verb) the floor [Object] with Opus [Comitative]." This information could be recovered by tracing up from detailed behaviors to recover more general behaviors with set-valued CAgents, but the Comitative slot permits caching this information to improve efficiency.

1.3. The Engineering Relevance of Case Grammar

The universal nature of cases suggests that they represent a fundamental characteristic of human thought, one that transcends cultural and linguistic differences. Such a fundamental characteristic is of vital importance in engineering artificial systems, for two reasons.

1. Human intelligence constitutes an existence proof that structures based on this pattern can indeed support rational thought and interaction. For thousands of years, human languages have been an important tool in describing and reasoning about the behavior of intelligent agents. Representational techniques that are used consistently across all known languages may well be as useful in support of artificial systems as they are in support of natural ones. In constructing artificial system with such properties, we can profitably copy nature.

2. Artificial systems usually interact with humans. People need to understand them at some level in order to create them, use them, and maintain them. A system whose own semantic structure follows the same structures as human thought should be easier for people to understand and thus to design, construct, use, and maintain.

2. Cases and Knowledge Representation[1]

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

[1] This material is extracted from [Parunak 1989].

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 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.

2.1. 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'. $x \in m$ 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 $\forall x.(C\_1\ x) \rightarrow (C\_2\ x)$. A non-subsuming link from one concept to another is a *slot*.

A slot is a two-place predicate over objects, 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 $\lambda x.(C\_1\ x)\ \&\ (C\_2\ x)$. 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 $\lambda x.(C\_1\ x)\ \&\forall y.(S\_1\ x\ y) \rightarrow (C\_2\ y)$. If C_Machine has a slot S_Status, `C_AvailableMachine = (CRestrict C_Machine S_Status C_Idle)` describes exactly those machines that are idle. These two, though not a complete set, suffice to illustrate the ideas in this paper. See Vilain 1985 for a fuller set.

2.2. 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.

2.2.1. 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 operations 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 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.

2.2.2. 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_Action)`. Yet the relationship between C_Machine and its dependent concept in each of these cases is very different from the others.

2.3. 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). 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.

2.4. Dealing with Asymmetry

Early net systems treat nominal and verbal concepts symmetrically, with separate subsumption lattices for each (Quinlan 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.

2.5. 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.

2.5.1. 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 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. 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. M_Case can be viewed as set of slot-makers, one for each case. For example, `(M_Case CAgent 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. 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 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.

2.5.2. 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)))
```

2.5.3. 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 is the object of the cutting action performed by the lathe, 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 nominal semilattice. As the system grows slot definitions tend "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.

2.6. Cases and Knowledge Representation

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.

3. Cases and System Design[2]

Existing research on agent-based factory control and scheduling takes two major approaches. Some researchers focus attention on the workstations as

[2] This material is extracted from [Parunak, Baker, & Clark, 1995].

agents, through which parts move passively. Others make agents out of parts as well as workstations. In selecting agents for a recent scheduling architecture, we wanted to begin with the broadest possible set of candidates. While some entities may prove unnecessary, it's easier to cast the net broadly and leave some as stubs than to build an architecture into which omitted entities cannot easily be added later. Case grammar is a way to ensure complete coverage.

Instead of the case names used by linguists (which are too general for our purposes, and in the case of "Agent" might lead to unfortunate confusion), we define a set that is meaningful for manufacturing. We can describe what happens in a factory in a series of sentences of the form, "Joe oversaw Unit Process 12 on Part 25 from Acme Supplies, using Mill 32, Cutter 86, and Part Program 19, producing Part 26 for US Army Order 22." The essential components of such a sentence, and the case labels by which we shall refer to them, are:

Unit Process ("Unit Process 12").--This is the fundamental "verb" of manufacturing, the unit process (the level below which manufacturing ceases to be a discrete activity). It is useful for our purposes to abstract the Unit Process away from all of the Resources that are used to perform it, since much of scheduling consists of identifying alternative Resources that can be used to perform what we would like to consider the same underlying Unit Process. The Manufacturing Studies Board of the National Academy of Sciences distinguishes five basic types of unit processes: MasS_Change, Phase- Change, Structure-Change, Deformation, and Consolidation [NASMSB 1995]. An instantiation of a Unit Process in space and time (e.g., the specific instance of heat treating performed in Oven 18 at 14:32 3 May 1993) is an Operation.

Resource ("Mill 32," "Cutter 86," "Part Program 19").--The linguist's Instruments; the "tools" that are needed to perform an Operation (an instance of the Unit Process), including machines, material handling devices, energy, tooling, fixtures, gauges, part programs, and documentation. Resources are characterized by such things as maintenance schedules, availability, and cost of use. Those aspects of the machine operator that are required to complete a unit process are best modeled as a Resource as well. Other aspects of humans are covered in the Manager category.

Manager ("Joe").--This is the human responsible for the Operation. In an automated factory, it becomes the plant manager. Automated representatives watch for things like chaos, performance metrics, energy consumption, cash flow, ...;

Part ("Part 25," "Part 26").--The inputs (Materials) and outputs (Products) for a Unit Process. There may be more than one input (in assembly) or output (in disassembly, or sawing up bar stock, or injection molding for a tree of parts). Between Unit Processes, Parts are in the custody of material handling operations such as transport and storage mechanisms. Theoretically, Material Handling may be considered just another Unit Process that changes the age and location of a part. However, these changes do not alter the part functionally, and the part number does not change across a material handling operation (as it does across other Unit Processes). Thus we make Parts responsible for their own material handling, and permit them to acquire necessary Resources just as Unit Processes do in order to move from one Unit Process to another. This generalization of a Part (which at any moment may or may not actually contain a part) may be called a Buffer.

Customer ("US Army Order 22").--The linguistic Beneficiary; the one who benefits from the execution of the work. The Customer represents a single purchase decision, or order. This cohesion is necessary in order to let the system handle each unit in a purchase separately (for processing simplicity) and yet be able to identify different parts that will all be made or not made based on the same purchase decision (so that we can take advantage of economies of scale).

Supplier ("Acme Supplies").--Another variety of Beneficiary, this time the one from whom the input material is purchased.

To complete the design, these categories are reviewed against overall system requirements. This review leads to the merging of some categories and the refinement of others. The case analysis does not provide a finished system design, but does give a starting point that lends itself to discussion among the developers and has proven to be very robust in terms of covering the issues that need to be addressed.

4. Cases and Agent Behavior[3]

This section describes the application of case theory to analyzing the behavior of individual agents and their interactions with one another.

[3] This material is extracted from [Parunak 1992].

4.1. The Problem

[Durfee and Montgomery 1990, 1991] (hereafter D&M) generalize organizations, plans, and schedules into the notion of "behavior." Behaviors can be situated in a "behavior space" indexed on the WH-words (What, When, Where, Who, Why, and How), and interactions of behaviors (such as conflicts) can be detected by searching for relationships (such as overlaps) in this space. For example, if two agents have behaviors that require them to be in the same place at the same time, those behaviors conflict.

Searching for all possible interactions among behaviors is in general combinatorially prohibitive. For time and space, lack of intersection between larger intervals guarantees lack of intersection between lower ones. This hierarchical heuristic permits drastic pruning.

D&M suggest that other dimensions of behavior space might also be viewed as hierarchies, permitting similar pruning, but the extension is not straightforward. Furthermore, there are ambiguities and omissions in the use of English WH- words as dimensions of behavior space. This section uses case grammar to provides a cognitive justification for, and refinement of, the description of behavior space by WH- words. The full paper from which this is material is drawn [Parunak1992] uses another linguistic concept (inter-propositional relations) to generalize the mechanisms they use for testing overlap in space and time so that they can be applied along other dimensions of behavior space.

The English WH- words are surface markers for two rather different sets of linguistic relations, one within and the other between predications. We will develop the notion that a behavior is a predication, whose inner structure is revealed by relations of the first sort, and whose connection to other behaviors can be described by relations of the second sort.

Roughly speaking, relations in the first set (case relations) exist between the verb of a predication and its nouns. For instance, the predication "Opus collected bottles" contains one verb ("collected") and two nouns ("Opus" and "bottles"). Each of these nouns stands in a different relation to the verb. "Opus" names the agent who causes the action of the verb to occur, while "bottles" identifies the entities on which the action takes place. This extract from [Parunak 1992] develops the notion of case relations and shows how they permit a refinement of the notion of a behavior in general and some of the WH- concepts in particular.

If the relations in the first set join nouns to a verb in a single predication, relations in the second set (discourse relations) join multiple predications together. The sentence "Opus cleaned the lab by gathering bottles" consists of two predications: "Opus cleaned the lab" and "Opus gathered bottles." It further asserts that the second predication describes the *means* by which the first becomes true. [Parunak 1992] also develops the notion of discourse relations and shows how they refine the WH- concepts that are not subsumed under case theory.

This section and [Parunak 1992] do not pretend to be thorough expositions of case and discourse grammar, respectively. The sets of cases and discourse relations that they present are neither complete nor unique partitions of the spaces that they cover. My objective is to illustrate the usefulness of these two aspects of linguistic theory for representing behaviors in a DAI system. The references should be consulted for more complete expositions.

4.2. Cases and D&M's Behavior Space

While D&M do not base their work explicitly on case theory, they have intuitively recognized a number of the distinctions implied by linguistic cases. The following list shows where they fit in, and also illustrates how use of the case formalism can fill in gaps that they have inadvertently left.

Object.--D&M do not distinguish the Object slot, but merge it into the verb in *What*-fillers such as "containerS_collected" and "bottleS_recycled" (where "containers" and "bottles" are Objects to their verbs). This clumping rests on a sound intuition: a verb by itself is not a proposition, and every proposition requires at least an Object. For the kinds of verbs that form reasonable behaviors, other things besides an Object are needed, too. Theoretically, it is useful to tease out the Object along with other cases, recalling that the definition of a behavior requires specification of its case fillers as well as its verb.

CAgent.-- D&M use *Who* to represent the CAgent role.

Experiencer.--The Experiencer is not covered in D&M, though it is important for modeling behaviors of communicating agents.

Beneficiary.-- D&M do not model this role.

Location.--The Location slot corresponds to the *Where* dimension in D&M.

Time.--The Time slot corresponds to D&M's *When.*

Instrument.-- D&M do not provide for an Instrument explicitly, though it represents an alternative interpretation of *How*.

Comitative.--D&M do not model this slot.

4.3. Slots and Fillers

With a set of slots in hand, we can represent any proposition as a tuple <verb, CAgent = ..., Object = ..., ...>. Cases not listed in the tuple have the value NIL.

Slots can be filled in three ways. Most commonly, slots are filled with entities that populate the universe (such as agents or inanimate things) or coordinates over the universe (such as space and time). Object can embed another complete proposition in the case of experiential verbs ("Tom [CAgent] told (verb) Ed [Experiencer] {that his dissertation was finished} [Object = proposition]"). Third, a slot can be unified with a slot filler in another proposition (not necessarily in the same slot), as in English relative clauses. "Program" is CAgent in "The program [CAgent] ran (verb) the robot [Object]," but when this proposition is embedded in another for slot unification, the CAgent of the embedded proposition becomes the Object of the embedding one: "Piotr [CAgent] showed (verb) Ed [Experiencer] the program [Object] that ran the robot." Another instructive example is "Tom [CAgent] worked on (verb) MICE [Object] until (the time that) Ed told him it was ready [Time]," where the Time slot in the main proposition is filled by unification with the filler of the implicit Time slot in the embedded proposition. (The second and third approaches establish a relation between two propositions and thus go beyond the simple "noun-verb" relations that characterize pure case theory. They are discussed further in [Parunak 1992].)

4.4. From Case Frames to Behaviors

A behavior is a proposition: that is, a case frame with its slots filled. (A case frame with some slots unfilled, or filled by variables, is a behavior schema.) This section develops this definition in more detail, taking into account the three classes of proposition recognized by linguists (stative, process, and action), and shows how the definition of a behavior as proposition unifies the concepts of organization, plan, and schedule, while illuminating some of our intuitions about distinctions among them by considering different ways in which the Time slot may be filled.

4.4.1. Classes of Propositions

The preceding discussion of the Object and CAgent slots hinted that not every verb can take every case frame. There are three basic verb types, depending on what cases are obligatory and what transforms are possible.

Stative verbs take no CAgent slot, and cannot be progressive or imperative. They describe some static situation: "The box [Object] is turning red (verb)," "The robot [Beneficiary] receives (verb) the box [Object]." The prototypical English process verb is "Become."

Action verbs require both CAgent and Object, and can be progressive and imperative. "The boy [CAgent] colorS_red (verb) the box [Object]." "The programmer [CAgent] gives (verb) the box [Object] to the robot [Beneficiary]." Action verbs can often be paraphrased as causative of a Process proposition, and the CAgent is the causer: "The boy [CAgent] causeS_to-become-red (verb) the box [Object]." The prototypical English action verb is "Do."

The question of whether a proposition of any class can define a behavior, or whether behaviors must be behaviors of some CAgent and therefore be Action propositions, awaits experimentation by means of implementation. If one permits stative and process as well as action behaviors, then the entity whose behavior is being discussed will be sometimes the Object, sometimes the CAgent, of the proposition, weakening the useful distinction that the case perspective provides. Certainly, a system needs to be able to reason about states (represented by stative propositions) and events (represented by process propositions) as well as actions; it's not clear at this point whether it is cleaner to follow the universal lead of human language and distinguish these from one another, or to lump them together into a generalized notion of behavior.

4.4.2. Organizations, Plans, and Schedules

The model of a behavior as a proposition supports and illuminates D&M's claim that the notion of Behavior subsumes that of Organization, Plan, and Schedule. All three of the latter are sets of related behaviors ("collective behavior"). The next few paragraphs show that many common intuitions about the distinctive character of Organization vs. Plan vs. Schedule can be related to differences in the relationships of the Time slot-fillers in the behaviors in a set. The intent here is not to offer more rigorous

definitions of these concepts, but to show how they are subsumed by the notion of Behavior.

A set of behaviors whose Time fillers are all concurrent corresponds to the common notion of an *Organization*. If the behaviors differ in their verbs, they represent a functional organization. If they have the same verbs but differ in their Object fillers, they represent a product organization. If the dominant difference is in Beneficiary slots, they are organized by market sectors. The concurrent Time fillers guarantee that these behaviors are going on together. (In 1950 IBM had a typewriter division but no personal computer division; in 1990 it had the latter but not the former; it is not useful to describe IBM in either 1950 or 1990 as an organization with a typewriter division and a personal computer division.) Figure 1 shows the relation of the behaviors that make up an Organization like this, where '=' indicates temporal concurrence.

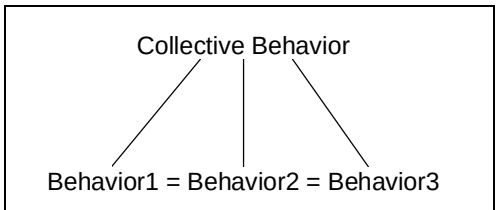

Figure 1: Concurrent Behaviors

A set of behaviors whose Time fillers are specified relative to one another in such a way that their completion times form a partial order corresponds to the common notion of a Plan. Figure 2 shows such a Plan, using '<' to indicate temporal precedence.

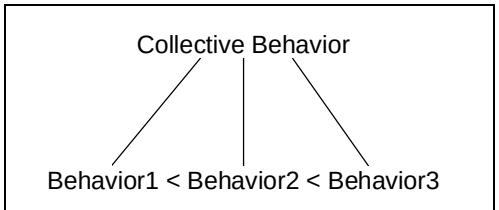

Figure 2: Ordered Behaviors

We often think of a *Schedule* is a set of behaviors whose Time fillers are bound directly to the time coordinate system defined for the universe, rather than (as in a Plan) to its other component behaviors. If '@' represents the time index of a behavior, Figure 3 might represent a Schedule

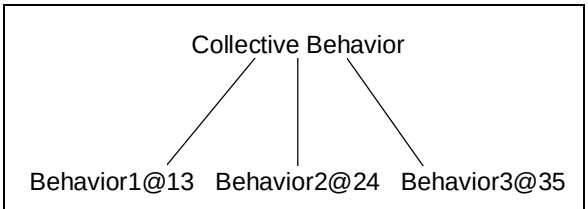

Figure 3: Scheduled Behaviors

In this example, the temporal interrelations between component behaviors are now implicit in the time indices, no longer explicit as in the Plan. As a result, if one behavior slips its schedule, a conflict may arise with other behaviors that depend on the first being complete. Thus the difference in the filler of the Time slot is directly responsible for one of the fundamental challenges in scheduling.

4.5. Cases and Agent Behavior

The notion of behavior space introduced by D&M is an important tool for reasoning about interactions of agents. The sketch of this space that they develop using the informal device of English WH- words can be made more rigorous and complete by exploiting linguistic case relations to expound the structure of a behavior. [Parunak 1992] also illustrates how another cognitive construct, propositional relations, can further strengthen the analysis and design of agent behaviors.

Summary

Case grammar is an important tool whose value for engineering agent-based systems has been demonstrated at three levels.

Knowledge Representation.--For some tasks, agents need to maintain and manipulate a symbolic model of their world, using (for instance) a first-order logic database or a semantic net. The categories in such system are often defined by introspection on the part of the researcher, and it not clear how the overall structure can be grounded or closed. A set of linguistic cases can be applied to four pro-verbs ("do", "be", and "become," which correspond to three categories of predication universally manifested in the world's languages) to recursively define nominal concepts, more complex verbal concepts, and more complex slots that can be used to modify those concepts. This approach thus applies case grammar to the internal architecture of a single agent.

Identifying Agents.--One of the first tasks in applying agents to a domain is to identify the kinds of agents that are necessary. Case grammar can be used to derive a candidate set of such agents directly from prose descriptions of the domain. Not only the nouns, but also the verbs in such statements can (with appropriate discipline) serve as candidate agents, which are further refined on the basis of requirements identified for the system as a whole. This approach provides a symmetry of analysis and a degree of

adaptability in the resulting architecture that is often lacking in other methods.

Modeling Behavior.--Agents often need to reason about behavior (both their own and that of other agents). The organization of a group of agents, the plans that they prepare, and schedules that govern their activities can all be considered specializations of the notion of "behavior," and can be used to define a "behavior space" within which agents can be situated. Case grammar can be used to formalize the structure of this space and ensure that it is complete enough to describe any behavior that can be talked about.

In each of these cases, case grammar not only enables us to copy nature's successful engineering, but also to build systems that will interface cleanly with human users, because they use commensurate cognitive constructs.

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

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

Cook, W.A., 1979. *Case Grammar: Development of the Matrix Model*. Washington: Georgetown University.

Durfee & Montgomery, 1990. "A Hierarchical Protocol for Coordinating Multiagent Behaviors." *Proceedings of the 1990 AAAI*, 86,93.

Fillmore, C.J., 1977. "The Case for Case Reopened." *Studies in Syntax and Semantics* 8, 59-81.

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.

Minsky, M., 1981. "A Framework for Representing Knowledge." In J. Haugeland, ed., *Mind Design*. Cambridge: MIT Press, 95-128.

NASMSB, 1995. *Unit Manufacturing Processes: Issues and Opportunities in Research.* Washington, DC: National Academy Press.

Parunak, 1989. "A Linguistic Approach to the Problem of Slot Semantics." *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, 797-804.

Parunak, 1992. "How to Describe Behavior Space." *Working Papers of the 11th International Workshop in Distributed Artificial Intelligence*, 303-316.

Parunak, Baker, & Clark, 1995. "Proposals for the AARIA Agent Architecture." Internal working paper.

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

Szolovits, P.; Hawkinson, L.B.; and 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." ICJAI-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.