SPECIAL SECTION

THE ROLE OF FRAME-BASED REPRESENTATION IN REASONING

A frame-based representation facility contributes to a knowledge system's ability to reason and can assist the system designer in determining strategies for controlling the system's reasoning.

RICHARD FIKES and TOM KEHLER

A fundamental observation arising from work in artificial intelligence (AI) has been that expertise in a task domain requires substantial knowledge about that domain. The effective representation of domain knowledge is therefore generally considered to be the keystone to the success of AI programs [15] (see Figure 1). Domain knowledge typically has many forms, including descriptive definitions of domain-specific terms (e.g., "power plant," "pump," "flow," "pressure"), descriptions of individual domain objects and their relationships to each other (e.g., "P1 is a pump whose pressure is 230 psi"), and criteria for making decisions (e.g., "If the feedwater pump pressure exceeds 400 psi, then close the pump's input value"). Because of this emphasis on representation and domain knowledge, systems that use AI techniques to achieve expertise are often referred to as *knowledge-based systems*, or simply as *knowledge systems*.

In order for a knowledge system to use domain-specific knowledge, it must have a language for representing that knowledge. The basic criteria for a knowledge representation language are the following:

- *Expressive power*—Can experts communicate their knowledge effectively to the system?
- *Understandability*—Can experts understand what the system knows?
- *Accessibility*—Can the system use the information it has been given?

Experience has made it increasingly clear that none of the major knowledge representation languages is by itself able to satisfy all of these criteria. Early attempts at building intelligent systems used the first-order predicate calculus as their representation language (e.g., [10]). The predicate calculus was appealing because of its very general expressive power and well-defined semantics. However, because the language constructs are very fine grained and do not provide adequate facilities for defining more complex constructs, domain experts have difficulty using the predicate calculus or understanding knowledge expressed in it. Also, the generality of the predicate calculus has been a significant barrier to the development of effective deduction facilities for using knowledge expressed in it.

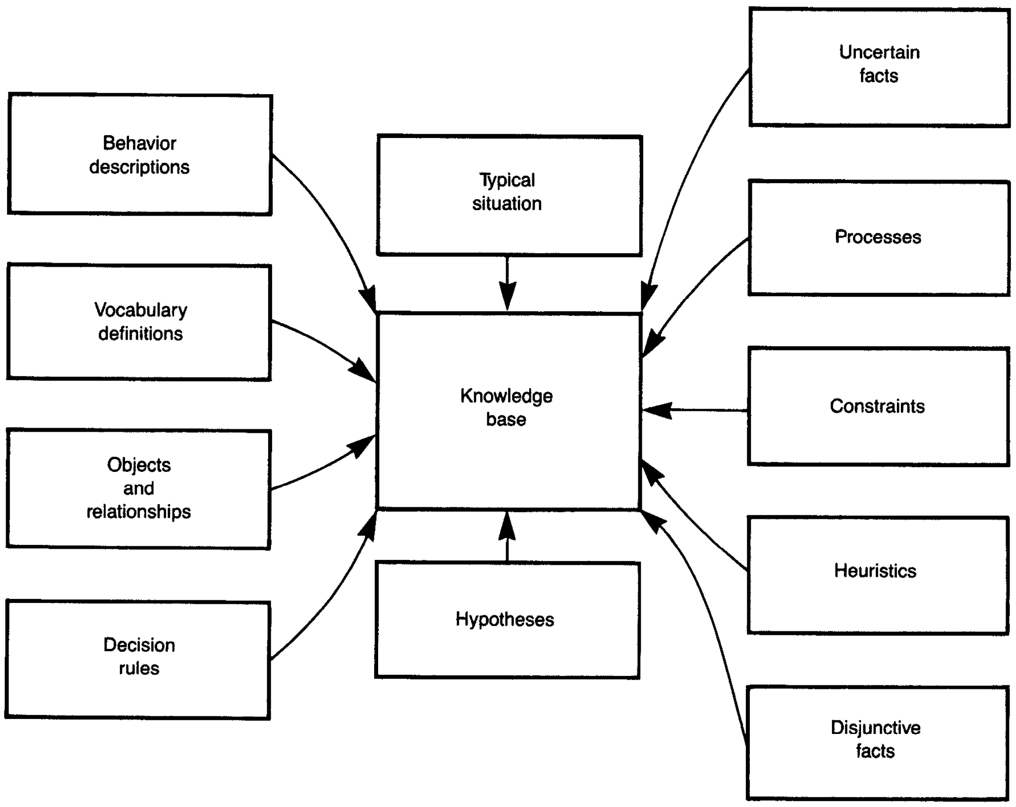
These difficulties helped motivate the development of "semantic networks" (e.g., [11]), and various "object-oriented" representation languages based on *frames* (e.g., [2, 4]). Frame languages provide the knowledge-base builder with an easy means of describing the types of domain objects that the system must model. The description of an object type can contain a prototype description of individual objects of that type; these prototypes can be used to create a default description of an object when its type becomes known in the model.

A frame provides a structured representation of an object or a class of objects. For example, one frame might represent an automobile, and another a whole class of automobiles (see Figure 2). Constructs are available in a frame language for organizing frames that represent classes into taxonomies. These constructs allow a knowledge-base designer to describe each class as a *specialization* (subclass) of other more generic classes. Thus, automobiles can be described as vehicles plus a set of properties that distinguish autos from other kinds of vehicles.

The advantages of frame languages are considerable: They capture the way experts typically think about much of their knowledge, provide a concise structural representation of useful relations, and support a concise definition-by-specialization technique that is easy for most domain experts to use. In addition, special-purpose deduction algorithms have been developed that exploit the structural characteristics of frames to rapidly perform a set of inferences commonly needed in knowledge-system applications.

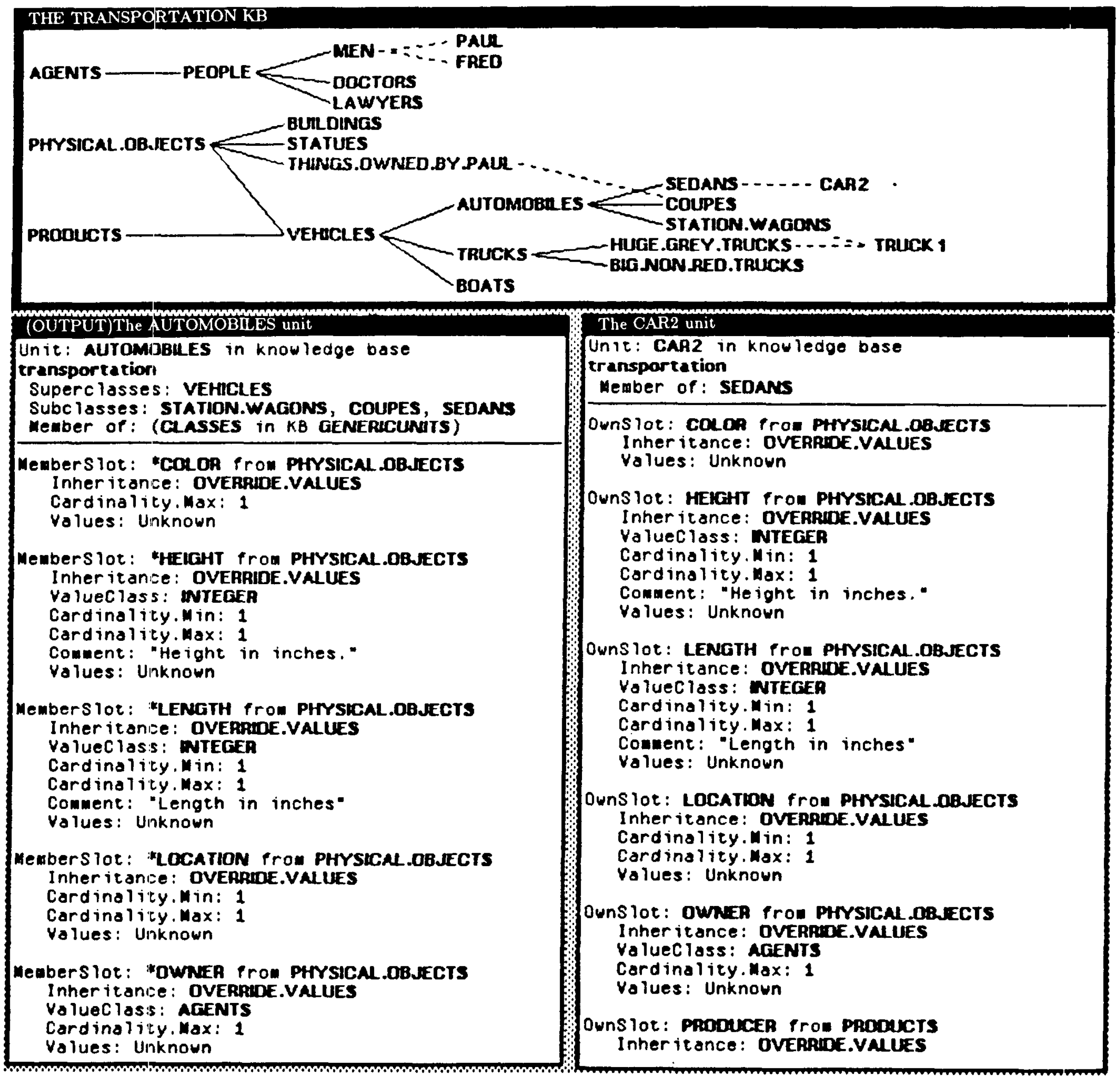
In addition to encoding and storing beliefs about a problem domain, a representation facility typically performs a set of inferences that extends the explicitly held set of beliefs to a larger, virtual set of beliefs. Thus, the representation facility participates in the system's reasoning activities by providing these "automatic" inferences as part of each assertion and retrieval operation. Frame languages are particularly powerful in this regard because the taxonomic relationships among frames enable descriptive information to be shared among multiple frames (via *inheritance*) and because the internal structure of frames enables semantic integrity constraints to be automatically maintained.

One of the basic tenets of knowledge-system technology is that domain knowledge can be more effectively used by a system and more easily understood by a system's users if it is represented in declarative rather than procedural form. Frame systems, however, provide no direct facilities for declaratively describing how the knowledge stored in frames is to be used. Traditionally, the only way of associating domain-dependent behavior with frames has been by attaching to them in various ways procedures written in the underlying programming language (e.g., LISP) (as in, for example, KL-ONE [4] and KRL [2]). Additional facilities are needed in such systems for declaratively describing domain-dependent inference rules, analysis decision rules, actions that can be taken in the domain by



Expertise in a task domain usually draws on many different kinds of knowledge about that domain. The representation and reasoning facilities in AI systems must be able to integrate different kinds of knowledge into a coherent knowledge base that can effectively support the system's activities.

FIGURE 1. The Kinds of Knowledge That Can Go into a Knowledge Base



```
Unit: AUTOMOBILES in knowledge base
transportation
 Superclasses: VEHICLES
 Subclasses: STATION.WAGONS, COUPES, SEDANS
 Member of: (CLASSES in KB GENERICUNITS)

MemberSlot: *COLOR from PHYSICAL.OBJECTS
   Inheritance: OVERRIDE.VALUES
   Cardinality.Max: 1
   Values: Unknown

MemberSlot: *HEIGHT from PHYSICAL.OBJECTS
   Inheritance: OVERRIDE.VALUES
   ValueClass: INTEGER
   Cardinality.Min: 1
   Cardinality.Max: 1
   Comment: "Height in inches."
   Values: Unknown

MemberSlot: *LENGTH from PHYSICAL.OBJECTS
   Inheritance: OVERRIDE.VALUES
   ValueClass: INTEGER
   Cardinality.Min: 1
   Cardinality.Max: 1
   Comment: "Length in inches"
   Values: Unknown

MemberSlot: *LOCATION from PHYSICAL.OBJECTS
   Inheritance: OVERRIDE.VALUES
   Cardinality.Min: 1
   Cardinality.Max: 1
   Values: Unknown

MemberSlot: *OWNER from PHYSICAL.OBJECTS
   Inheritance: OVERRIDE.VALUES
   ValueClass: AGENTS
   Cardinality.Max: 1
   Values: Unknown
```

```
Unit: CAR2 in knowledge base
transportation
 Member of: SEDANS

OwnSlot: COLOR from PHYSICAL.OBJECTS
   Inheritance: OVERRIDE.VALUES
   Values: Unknown

OwnSlot: HEIGHT from PHYSICAL.OBJECTS
   Inheritance: OVERRIDE.VALUES
   ValueClass: INTEGER
   Cardinality.Min: 1
   Cardinality.Max: 1
   Comment: "Height in inches."
   Values: Unknown

OwnSlot: LENGTH from PHYSICAL.OBJECTS
   Inheritance: OVERRIDE.VALUES
   ValueClass: INTEGER
   Cardinality.Min: 1
   Cardinality.Max: 1
   Comment: "Length in inches"
   Values: Unknown

OwnSlot: LOCATION from PHYSICAL.OBJECTS
   Inheritance: OVERRIDE.VALUES
   Cardinality.Min: 1
   Cardinality.Max: 1
   Values: Unknown

OwnSlot: OWNER from PHYSICAL.OBJECTS
   Inheritance: OVERRIDE.VALUES
   ValueClass: AGENTS
   Cardinality.Max: 1
   Values: Unknown

OwnSlot: PRODUCER from PRODUCTS
   Inheritance: OVERRIDE.VALUES
```

Frames provide structured representations of objects or classes of objects. The AUTOMOBILES frame shown here (lower left) represents the class of all automobiles, and the CAR2 frame (lower right) represents a specific automobile that is a member of that class. Frames allow classes to be described as specializations of other more generic classes and for those descriptions to be organized into taxonomies. Thus, automobiles can be described as vehicles plus a set of properties that distinguish autos from other kinds of vehicles. The transportation taxonomy shown here (top) uses solid lines to represent class–subclass relationships and dashed lines to represent class–member relationships. For example, VEHICLES is a subclass of both PHYSICAL.OBJECTS and PRODUCTS, and TRUCK1 is a member of both HUGE.GREY.TRUCKS and THINGS.OWNED.BY.PAUL.

FIGURE 2. A Frame Taxonomy

various agents, simulations of object behavior, etc.

The most popular and effective representational form for declarative descriptions of domain-dependent behavioral knowledge in knowledge systems has been pattern/action decision rules, called *production rules* (e.g., [6, 7]). Production rules are, in effect, a subset of the predicate calculus with an added prescriptive component indicating how the information in the rules is to be used during reasoning. Production rules can be easily understood by domain experts and have sufficient expressive power to represent a useful range of domain-dependent inference rules and behavior specifications. By themselves, however, production rules do not provide an effective representation facility for most knowledge-system applications. In particular, their expressive power is inadequate for defining terms and for describing domain objects and static relationships among objects.

The major inadequacies of production rules are in areas that are effectively handled by frames. A great deal of success, in fact, has been achieved by integrating frame and production rule languages to form hybrid representation facilities that combine the advantages of both component representation techniques (e.g., LOOPS® [18], KEE® (Knowledge Engineering Environment®) [12], and CENTAUR [1]). These systems have shown how a frame language can serve as a powerful foundation for a rule language. The frames provide a rich structural language for describing the objects referred to in the rules and a supporting layer of generic deductive capability about those objects that does not need to be explicitly dealt with in the rules. Frame taxonomies can also be used to partition, index, and organize a system's production rules. This capability makes it easier for the domain expert to construct and understand rules, and for the system designer to control when and for what purpose particular collections of rules are used by the system.

Although a primary motivation for Minsky's introduction of frames [4] was to semantically direct the reasoning of scene-analysis systems, most of the subsequent work on frame-based systems (e.g., KRL [2], UNITS [17], and KL-ONE [4]) has focused on structural representation issues rather than on the control of reasoning. The information stored in frames has often been treated as the "database" of the knowledge system, whereas the control of reasoning has been left to other parts of the system. This focus on structural representation issues has helped to elucidate the semantics of the common frame constructs and to demonstrate their usefulness for organizing and storing knowledge (e.g., [5]). Little attention, however, has been paid to whether and how those constructs can be useful for controlling reasoning.

Recent experience with frame-based representation facilities in complex application domains has shown that frames can play an important role throughout the system, including in the control of reasoning components. For example, the structural features of frame languages have proved to be very useful for organizing and controlling the behavior of large collections of production rules. These uses of frames are our central theme in this article. We elaborate the various ways in which a frame-based representation facility participates in a knowledge system's reasoning functionality and can assist the system designer in determining strategies for controlling a system's reasoning.

LOOPS is a trademark of Xerox Corporation.
KEE and Knowledge Engineering Environment are trademarks of IntelliCorp.

COMPONENTS OF A FRAME-BASED REPRESENTATION FACILITY

In this section we summarize the basic components of a typical frame-based representation facility in order to indicate the salient features of frame systems and to provide a context for the discussions in subsequent sections. The facility described is a component of the KEE system [12]. In order to highlight the role that a frame-based representation facility plays in the reasoning of a knowledge system, our description explicitly distinguishes between the semantic interpretation of frame language constructs (e.g., that a `MemberOf` link between frames M and C denotes the proposition that the object represented by M is a member of the class represented by C), and the reasoning services that are typically provided by a frame-based representation facility (e.g., that when a `MemberOf` link is created between frames M and C, the default description in C of members of the class represented by C is added to M).

Structural Features

Taxonomy Descriptions. The frame-based representation language included in the KEE system provides typical frame language constructs for describing individuals and classes of individuals in an application domain (see Figure 3). Each individual or class is represented by a frame.[1] Frames can be organized into taxonomies using two constructs that represent relationships between frames: *member links*, representing class membership, and *subclass links*, representing class containment or specialization. These links provide two standard interpretations of the meaning of "is–a" links, as in "A truck *is a* vehicle" and "`TRUCK1` *is a* truck." (See [3] for a discussion of the variety of interpretations of "is–a" in frame systems.)

Frames can incorporate sets of attribute descriptions called *slots*. A distinguishing characteristic of frame-based languages is that a frame representing a class can contain prototype descriptions of members of the class as well as descriptions of the class as a whole. In the KEE system, prototype descriptions are distinguished from other descriptive information by the use of two kinds of slots, *own slots* and *member slots*. Own slots can occur in any frame and are used to describe attributes of the object or class represented by the frame. Member

[1] Some systems use other terms for what we are calling frames. For example, frames are called *units* in the KEE system and *concepts* in KL-ONE. We use the single generic term "frame" in all cases here for consistency.

```
----------------------------------------------------------

Frame: TRUCKS in knowledge base TRANSPORTATION
   Superclasses: VEHICLES
   Subclasses: BIG.NON.RED.TRUCKS, HUGE.GREY.TRUCKS
   MemberOf: CLASSES.OF.PHYSICAL.OBJECTS

----------------------------------------------------------
        .
        .
        .

MemberSlot: HEIGHT from PHYSICAL.OBJECTS
   ValueClass: INTEGER
   Cardinality.Min: 1
   Cardinality.Max: 1
   Units: INCHES
   Comment:  "Height in inches."
   Values: Unknown

MemberSlot: LENGTH from PHYSICAL.OBJECTS
   ValueClass: NUMBER
   Cardinality.Min: 1
   Cardinality.Max: 1
   Units: METERS
   Comment:  "Length in meters"
   Values: Unknown
        .
        .
        .

OwnSlot: LONGEST from CLASS.OF.PHYSICAL.OBJECTS
   ValueClass: TRUCKS
   Cardinality.Min: 1
   Cardinality.Max: 1
   Comment:  "The longest known truck"
   Values: Unknown

OwnSlot: TALLEST from CLASS.OF.PHYSICAL.OBJECTS
   ValueClass: TRUCKS
   Cardinality.Min: 1
   Cardinality.Max: 1
   Comment:  "The tallest known truck"
   Values: Unknown
        .
        .
        .

----------------------------------------------------------
```

This frame describes class `TRUCKS` as a subclass of class `VEHICLES` and as a member of class `CLASSES.OF.PHYSICAL.OBJECTS`. Member slots in the `TRUCKS` frame like `LENGTH` and `HEIGHT` provide a prototype description of each class member. Own slots like `LONGEST` and `TALLEST` describe attributes of the class as a whole.

FIGURE 3. The `TRUCKS` Frame

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

fastcase® Smarter legal research.