

NOTICE WARNING CONCERNING COPYRIGHT RESTRICTIONS:
The copyright law of the United States (title 17, U.S. Code) governs the making of photocopies or other reproductions of copyrighted material. Any copying of this document without permission of its author may be prohibited by law.

Describe - An Explanation Facility For An Object-Based Expert System

by

Robert L. Joseph
Carnegie-Mellon University
Pittsburgh, PA 15217

J. Robert Endsor

Alex Dickinson

Richard L. Blumenthal
AT&T Bell Laboratories
Holmdel, New Jersey 07744

Abstract

An important property of an expert system is the ability to explain its actions to its users and developers. This paper discusses the structure and implementation of an explanation system that is used to describe the actions an associated object-based expert system. The construction of an expert system as a collection of objects has significant consequences on the design of an explanation system. These issues and our solution are described.

This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 4976, monitored by the Air Force Avionics Laboratory Under Contract F33615-84-K-1520.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

Table of Contents

1. INTRODUCTION	1
2. OVERVIEW OF DESCRIBE	2
2.1. Rule Entry	2
2.1.1. User Viewpoints	2
2.1.2. Classification	3
2.2. Data Base Justification	3
2.3. User Interactions - Dynamic Views	3
3. INITIALIZATION OF DESCRIBE	3
4. RULE DEFINITION	4
5. RULE EXECUTION	5
6. DATA JUSTIFICATION	5
7. THE USER INTERFACE	6
8. SUMMARY	7

Describe - An Explanation Facility For An Object-Based Expert System

by

Robert L. Joseph

Carnegie-Mellon University
Pittsburgh, PA 15217

J. Robert Endsor

Alex Dickinson

Richard L. Blumenthal

AT&T Bell Laboratories
Holmdel, New Jersey 07744

1. INTRODUCTION

An important property of an expert system is its ability to describe itself. The modern expert system should be able to explain what it is doing, why it is pursuing a course of action, and why it has reached a conclusion. Explanation facilities are an important aid to the construction, refinement, and understanding of a large expert system. An expert system is usually built incrementally, and its knowledge base often begins as a small, incomplete set of rules. In this early stage an explanation facility displays views of the knowledge base and activities of the rule-based computation, thus indicating the needed developments. As the system matures through a series of refinements the explanation aids in the development and monitoring of changes. Finally explaining the behavior and response of a mature system increases its credibility and supports user-confidence.

Object-based expert systems are becoming common. An object is a language construct that encapsulates a set of data and its associated operations. An object-based program is constructed as a collection of objects that communicate with each other via messages. The expert systems that have been built using this methodology (e.g. Ha¹, Ni², Te³) distribute rules among several objects, called knowledge sources. These knowledge sources cooperate to solve problems in the system's domain.

Describe, written in Zetalisp⁴, is an explanation facility for object-based expert systems. Rules can be defined throughout the knowledge sources. The user can then display the structure and behavior

of the the expert system, present various views of the system's rules and show what the system has done in the past. Describe even provides the ability to record and retrieve justifications attached to the expert system's data entries. Finally the user interacts with the system through an *inspection type* window interface. The Describe facility provides an expert system with a dynamic and effective designing, explanation and debugging tool.

The next section presents an overview of the facility. Initialization of Describe is explained in Section 3. Section 4 presents the rule entry mechanism of the system, and Section 5 describes rule invocation. The data justification are discussed next in Section 6 and in Section 7 an explanation of the user interfaces is given.

2. OVERVIEW OF DESCRIBE

Describe executes in the Zetalisp environment on the Symbolics Lisp Machine. Figure 1 illustrates the structure of the system and its environment. The expert system itself interacts with Describe through the rule definition and invocation mechanisms, and the data justification mechanism. Using the *defrule* macro, the expert system declares its rules and records them in the Describe rule data base. These rules are then invoked as functions. A trace of the rule execution that results in a conclusion or data entry may be recorded and retrieved with the Describe data justification mechanism. The Describe user interface provides an "inspection" window that allows the user to see the system rules and activity.

2.1. Rule Entry

Rules are entered into an expert system by use of the *defrule* macro. Each rule belongs to a class of rules, and is declared within some object. Different rules are of interest to different users, so each rule has an associated set of user viewpoints.

2.1.1. User Viewpoints

Like the XPLAIN⁵ system, Describe presents explanation from the perspective of the person interested in the information. For instance, the information presented in a casual user's viewpoint would likely be much less detailed than that presented in a programmer's viewpoint.

A set of user's viewpoint is associated with each rule as it is defined, and during interaction with Describe, the user has a viewpoint. Only those rules that are associated with this viewpoint are used in descriptions to the user. Describe has operations that allow the user to see the set of all viewpoints, to add viewpoints, and to change his viewpoint.

2.1.2. Classification

The rules of the expert system are defined in objects. This encapsulation generates a useful structure for the system's rules. The rules are also grouped into user defined classes, yielding another structure, one which is presumably orthogonal to the first. For example, a plumonary diagnosis object and a prescription object might each contain pneumonia rules and bronchitis rules. Describe has operation to display the rules of each class, as well as the rules of each object containing rules.

2.2. Data Base Justification

As an expert system reaches conclusions or makes entries into some data base, it might wish to record justifications for these actions. Describe provides an operation that offers an explanation for the current action. Describe also provides operations that parse this execution trace. These could then be used to later justify a conclusion or data base entry. The justification functions are accessed by function calls in the user's program.

2.3. User Interactions - Dynamic Views

The Describe user interface provides the user with several views of an expert system as it is running. The user may see all the rules that have fired, all the rules the have taken some action, all the true rules , and all the false rules. Thus the Describe user interface maybe used as a debugging tool during system development.

3. INITIALIZATION OF DESCRIBE

The function *DES-initialize* creates the Describe system and is called exactly once before the expert system is executed. *DES-initialize* is called with two arguments. The first argument sets up some defaults for user convience. This argument is a list and each item in the list is a pair. The first element of each pair is the name of a class of rules, and the second element is a list of viewpoints associated with that rule class. Later, when defining a rule of some class, the default viewpoints associated with that rule will be the viewpoint of the second element.

The second argument is a list of the viewpoints that are to be used in the system.

DES-initialize returns a value that allows the user to access the just-initialize instance of the Describe data base. This value must be preserved if the user is going to interact with the system other than to define rules. In particular the justification functions and Describe user interface require this value.

A call to *DES-initialize* has the form:

```
(DES-initialize '((class1 (class-viewpoints-list)) ....)
                '(global-viewpoint-list))
```

In line example:

```
;;; The only rule-class is identify-rules; the only viewpoint is horse.
;;; animal-des is the variable that can be used to access describe data base.
```

```
(setq animal-des (DES-initialize '((identify-rules)) '(horse)))
```

An execution of a given expert system is termed a session. Between successive sessions the expert system should reset the Describe system with the message `:reset`. This message resets the trace information to an initial state, while preserving the rule data base (*DES-initialize* destroys both trace information and the rule base).

4. RULE DEFINITION

The rules of the expert system are defined using the *defrule* macro, whose structure is:

```
(defrule (rule-name flavor-name) (argument)
        (class-name viewpoint-list)
        IF if-clause
        THEN then-clause
        ELSE else-clause-list
        "documentation")
```

where *viewpoint-list*, *else-clause-list*, and *documentation string* are optional parameters.

The name of the rule is that given by the first parameter. The rule is local to the flavor named by the second parameter. (Actually the rule is implemented as a *defun-method*). *Arg-list* is a list of arguments to be passed to the rule whenever it is invoked. This rule is a member of the rule class. If the *viewpoint-list* is specified, it overrides the class default. The body of the rule is specified by an *if-then* or *if-then-else* expression. The *documentation* is optional, unevaluated text.

Defrule returns the method that must be called to access the rule.

The series of calls to *defrule* that defines the system's rules should only occur after a call to *DES-initialize*. The code segment below illustrates rule definition.

```

;;; A method to carry out the rule definitions.
;;; Notice the scoping. The rules will only be able to access globals and
;;; instance variables of the flavor named in their definition. Violation
;;; of this will not be noticed until run-time as this is when the defrule
;;; macro within a macro is expanded.
(defmethod (animal-farm :define-rules) ()

  ;; Rule id1 in the animal-farm flavor is of rule-class identify-rules,
  ;; and has the viewpoint student.
  (defrule (id1 animal-farm) () (identify-rules '(student)
    IF (send data-base :recall '(animal has hair))
    THEN ((send data-base :remember '(animal is mammal))))

  ;; Rule id2 in the animal-farm flavor is also of rule-class identify-rules,
  ;; and may be seen by all viewpoints.
  (defrule (id2 animal-farm) () (identify-rules)
    IF (send data-base :recall '(animal gives milk))
    THEN ((send data-base :remember '(animal is mammal)))
    "this is a definitive rule for mammals")

  .
  .
  )

```

5. RULE EXECUTION

A rule is local to the flavor the is specified in its definition (not necessarily the flavor the definition is carried out in). The rule is executed by the form:

```
(rule-name args ....)
```

A rule returns the value of its evaluated predicate (if-clause).

6. DATA JUSTIFICATION

There are several functions provided for data justification. The first function, justify, is used to record the system activities that led to a conclusion or data entry. The remaining functions aid in the examination of the justify function reentries.

(justify description-system) - This function returns a list of all the rules that have fired during the session. (The symbol description-system represents the value returned by DES-initialize). Each entry on the list has the following format:

```
(flavor-name class name viewpoint-list parameter-value-list
  if-clause-evaluation)
```

This function's value indicates the reason that the current system state has been reached, and presumably justifies present conclusions or data entries.

(justify-name entry) - Returns the name from an entry passed to it.

(justify-flavor entry) - Returns the flavor of an entry.

(justify-viewpoint entry) - Returns the viewpoint of an entry.

(justify-parm-val entry) - Returns the value that was passed as a parameter to the rule for the specific entry.

(justify-rule-eval entry) - Returns what the rule evaluated to for a particular entry.

(justify-class entry) - Returns what the rule's class is for a particular entry.

7. THE USER INTERFACE

The describe user interface is a window-based tool. The interface is part of the system menu and is accessed by typing Select-D. It may also be invoked by the form.

(describer description-system)

The user types the name of the symbol bound to the description system (i.e., the value returned by *DES-initialize*) in the user program. Describe then has access to the rule base of the expert system, allowing dynamic tracing and rule status examination.

A typical usage of the Describe user interface is shown in figure 2. The user's expert system is running in the lower window (a lisp listner) and the Describe descriptions occupy the upper windows. As rules execute they print out in the scrolling trace window. The other windows service menu requests made by the user. Services available include:

1. Display all the rules in the current viewpoint.
2. Display all the unevaluated rules in the current viewpoint.
3. Display all the false rules in the current viewpoint.
4. Display all the true rules in the current viewpoint.
5. Display all the rules in the current viewpoint that have been activated.
6. Display an overview of the system.
7. Display further menus that enable the display of rules in particular classes and flavors.

Note that each call to *DES-initialize* creates a new rule base, so the function describer will need to be called again with the new value.

8. SUMMARY

Describe is an explanation facility for object-based expert system executing in the Zetalisp environment. This facility provides a rule-entry mechanism. It also displays the rule structure and execution trace of the associated expert system. Finally, Describe can be used to record and retrieve justification attached to the expert system's data entries. As of the writing of this article the system Describe is successfully being used in several expert systems at Bell Laboratories.

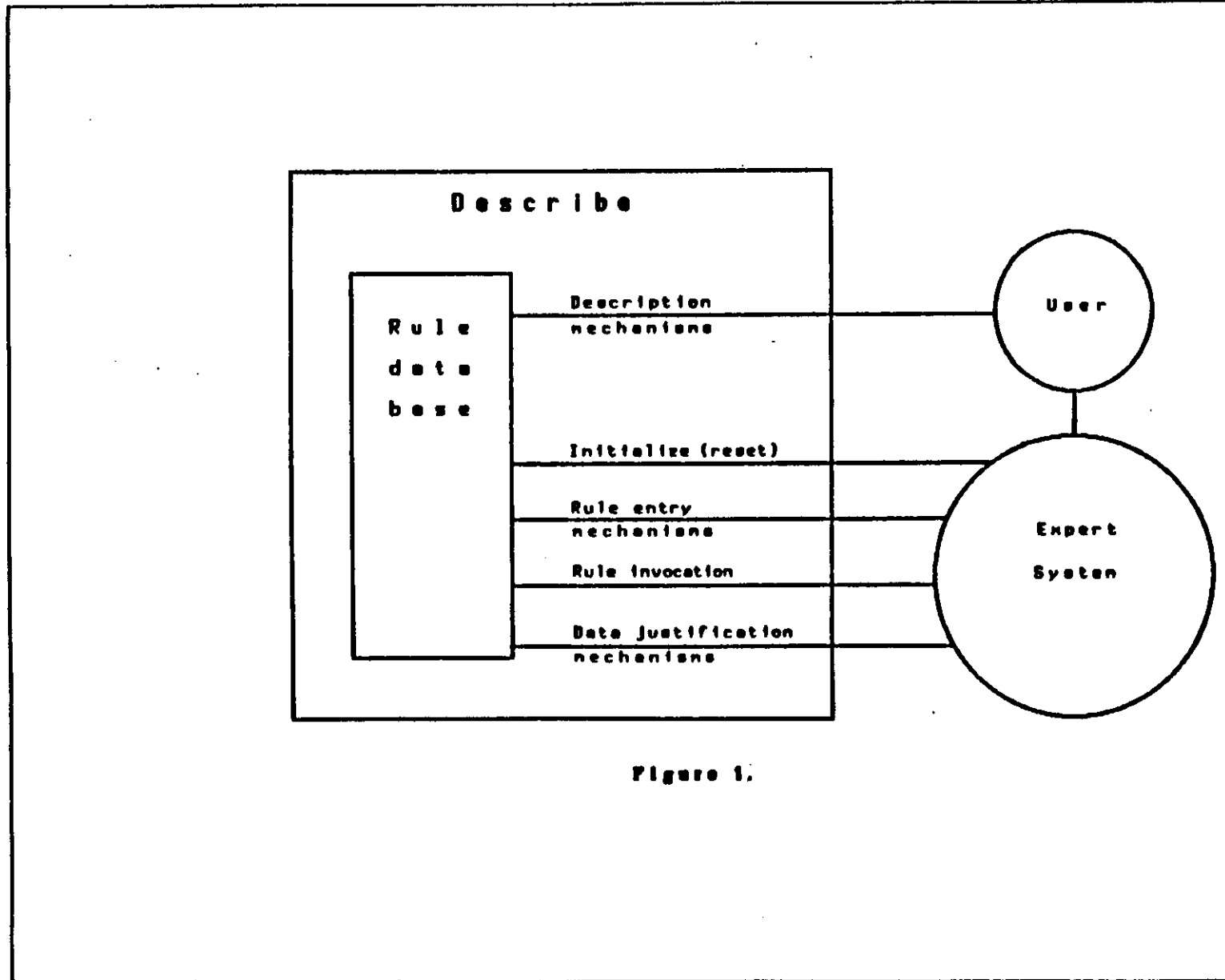



Figure 1.

animal-dea	
Interactor [#DESCRIPTION-SYSTEM 572441]	
<p style="text-align: right;"><i>More above</i></p> <p>Trace information [ALL] (ANIMAL-FARM ID2) has parameters NIL * NIL viewpoint (:ALL) IF (SEND DATA-BASE :RECALL (QUOTE (ANIMAL GIVES MILK))) (ANIMAL GIVES MILK) THEN ((SEND DATA-BASE :REMEMBER (QUOTE (ANIMAL IS MAMMAL))))</p> <p style="text-align: right;"><i>More below</i></p>	
<p style="text-align: center;"><i>top of items</i></p> <p>True IF Clauses [ALL] (ANIMAL-FARM ID2) * (ANIMAL GIVES MILK) viewpoint (:ALL) (ANIMAL-FARM ID1) * (ANIMAL HAS HAIR) viewpoint (:ALL) (ANIMAL-FARM ID2) * (ANIMAL GIVES MILK) viewpoint (:ALL) (ANIMAL-FARM ID1) * (ANIMAL HAS HAIR) viewpoint (:ALL)</p> <p style="text-align: right;"><i>More below</i></p>	<p>Trace</p> 
<p style="text-align: right;"><i>More above</i></p> <p>Current Rules [ALL] Rule: (ANIMAL-FARM ID15) Class: IDENTIFY-RULES Arguments: NIL Viewpoint: (ALL) IF (AND (SEND DATA-BASE :RECALL (QUOTE (ANIMAL IS BIRD))) (SEND DATA-BASE :RECALL (QUOTE (ANIMAL FLYS WELL)))) THEN ((SEND DATA-BASE :REMEMBER (QUOTE (ANIMAL IS ALBATROSS)))) ELSE NIL Explanation NIL</p> <p style="text-align: right;"><i>More below</i></p>	<p>Refresh</p> <ul style="list-style-type: none"> Set Viewpoint Print Rules Unevaluated Rules See Rule Menu False Rules True Rules Action Rules System Overview Exit
<p>()</p> <p>The facts are: ((ANIMAL HAS NOSE) (ANIMAL IS MAMMAL) (ANIMAL GIVES MILK) (ANIMAL HAS HAIR))</p> <p>Enter new facts (end with nil) : (animal has stripes)</p> <p>()</p> <p>The facts are: ((ANIMAL HAS STRIPES) (ANIMAL HAS NOSE) (ANIMAL IS MAMMAL) (ANIMAL GIVES MILK) (ANIMAL HAS HAIR))</p> <p>Enter new facts (end with nil) : []</p>	
Lisp Listener 2	

10/05/84 19:55:06 Alex

FLOYD:

Screen Dumping

Figure 2

References

1. Hayes-Roth, F., "The HEARSAY-II speech-understanding system: intergrating knowledge to solve uncertainty," *Computing Surveys* 12 (2), 1980, pp. 213 - 253.
2. Nii, H. P. and Feigenbaum, E. A., *Rule-based Understanding of Signls in pattern Directed Inference Systems*, D.A. Waterman and R. Hayes-Roth (eds) Academic Press, 1978.
3. Terry Allan, "The Crystals Project: Hierarchical Control of Production Systems," Tech. report, Stanford University Computer Science Department, May 1984.
4. Weinreb, D. and D. Moon, *Lisp Machine Manual*, 1981.
5. Swartout, William R., *XPLAIN: a System for Creating and Explaining Expert Consulting Programs*, Addison-Wesley, 1983, pp. 285 - 325.