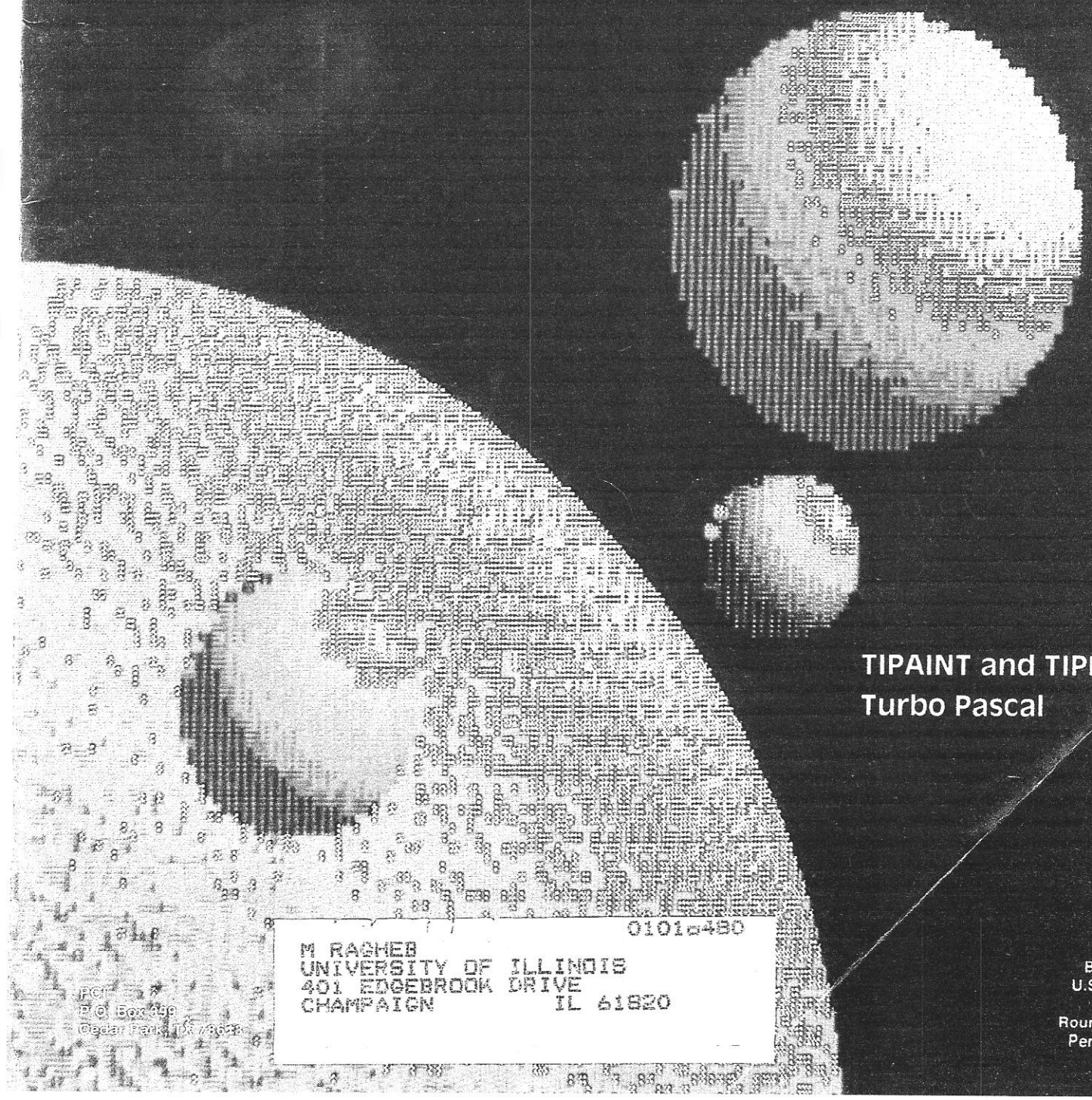
TI PROFESSIONAL COMPUTING

VOL. 3, NO. 6
June 1986
$3.50

A PCI Publication for Texas Instruments Computer Users

Expanding Vistas in Graphics



TIPAINT and TIPDRAW
Turbo Pascal

PCI
P.O. Box 49
Cedar Park, TX 78613

Bulk Rate
U.S. Postage
PAID
Round Rock, TX
Permit No. 45

Building Expert Systems in Prolog on the Explorer Machine

By Dan Rehfeldt and
Magdi Ragheb

The use of the Prolog Artificial Intelligence (AI) Language for building Production-Rule-Analysis or Expert Systems on the Texas Instrument's Explorer Symbolic Programming Machine is demonstrated. The same problem structure introduced by Winston and Horn in the LISP Language [1] is followed, and an enhanced version in Prolog of the problem as exposed by Yazdani [2] is presented.

Prolog, which stands for Programming in Logic is becoming the predominant AI language in Europe and Japan. It was initially developed in 1972 by A. Colmerauer and P. Roussel at the University of Marseilles, France [3]. Prolog is a simple programming language for logical programming, and as used in this work, its Explorer machine implementation provides an interface to LISP, thus generating a Knowledge Engineering Tool drawing on the strengths of both languages. Prolog enjoys an international popularity, and the publicized Japanese Fifth-Generation project has adopted Prolog as the fundamental language for the supercomputers they plan to build in the near future [4].

Like LISP [1], Prolog [3] is designed for symbolic rather than simple numerical computations. Computation in Prolog is controlled logical deductions. One states the facts known about a given situation and Prolog proceeds to tell whether or not any specific conclusion can be deduced from those facts. In the Knowledge Engineering Terminology, its control structure is logical inference. Although Prolog may be the best current implementation of logic programming, it cannot handle all the deductions that are theoretically possible in predicate calculus.

In Prolog, a programmer does not specify how the computer is to perform its assigned tasks, but just gives a description of the task as a sequence of constraints to be satisfied. This distinguishes it from the LISP programming language: in LISP one must specify the "how" of a given computation, whereas in Prolog one only specifies the "what" of a computation. In the latter case the machine is assigned the task of determining how to carry out the "how" part, which frees the programmer from worrying about the details of the algorithms which perform a given task. This is achieved at the expense of more computational resources, but those are becoming increasingly cheaper. This approach depends on assigning the different tasks of "how" and "what" to two different groups of researchers. As discussed by J. deKleer this idea may "Enable the Japanese to leap-frog over American software methodology" [5]. The combination of Prolog's theoretical elegance and its declarative interpretation will probably result in its steadily increasing acceptance worldwide. In the following, its use in building Expert Systems is demonstrated using the Animal Identification Problem [1,2], and its implementation on the Explorer machine is described. See figures 1-8, pages 13-26.

PROBLEM DESCRIPTION

For representation, we construct a logical model of the problem at hand using the technique of Fault-Tree Analysis from the field of System Analysis following the methodology described in Ref. 6. We consider a Knowledge-Base consisting of seven animals whose characteristics can be described in terms of the OR, AND and NOT logical operations:

Albatross = Bird * Fly * Fly-Well
Penguin = Bird * $\overline{\text{Fly}}$ * Swim * Black-and-white-color
Ostrich = Bird * $\overline{\text{Fly}}$ * Long-Neck * Long-Legs * Black-and-white-color
Giraffe = Ungulate * Long-Neck * Long-Legs * Dark-Spots (1)
Zebra = Ungulate * Black-Stripes
Tiger = Mammal * Carnivore * Yellow-color * Black-stripes
Cheetah = Mammal * Carnivore * Yellow-color * Dark-Spots

where (*) denotes the AND operation and (-) denotes the NOT operation.

These logical relationships are shown in Figure 1. In this figure, we can notice that the parameters at the bottom of the trees in single-framed rectangles are basic facts that need to be obtained interactively from the program user, whereas those in double-framed rectangles are intermediate deductions that need to be further described. They can be logically described as follows:

Carnivore = Eats-meat + (Pointed-teeth * Claws * Forward-eyes)
Ungulate = (Hoofs * Mammal) + (Mammal * Chews-Cud) (2)
Mammal = Hair + Gives-milk
Bird = Feathers + (Fly * Lay-eggs)

where (+) denotes the OR operation.

These relationships are shown in

continued on page 18

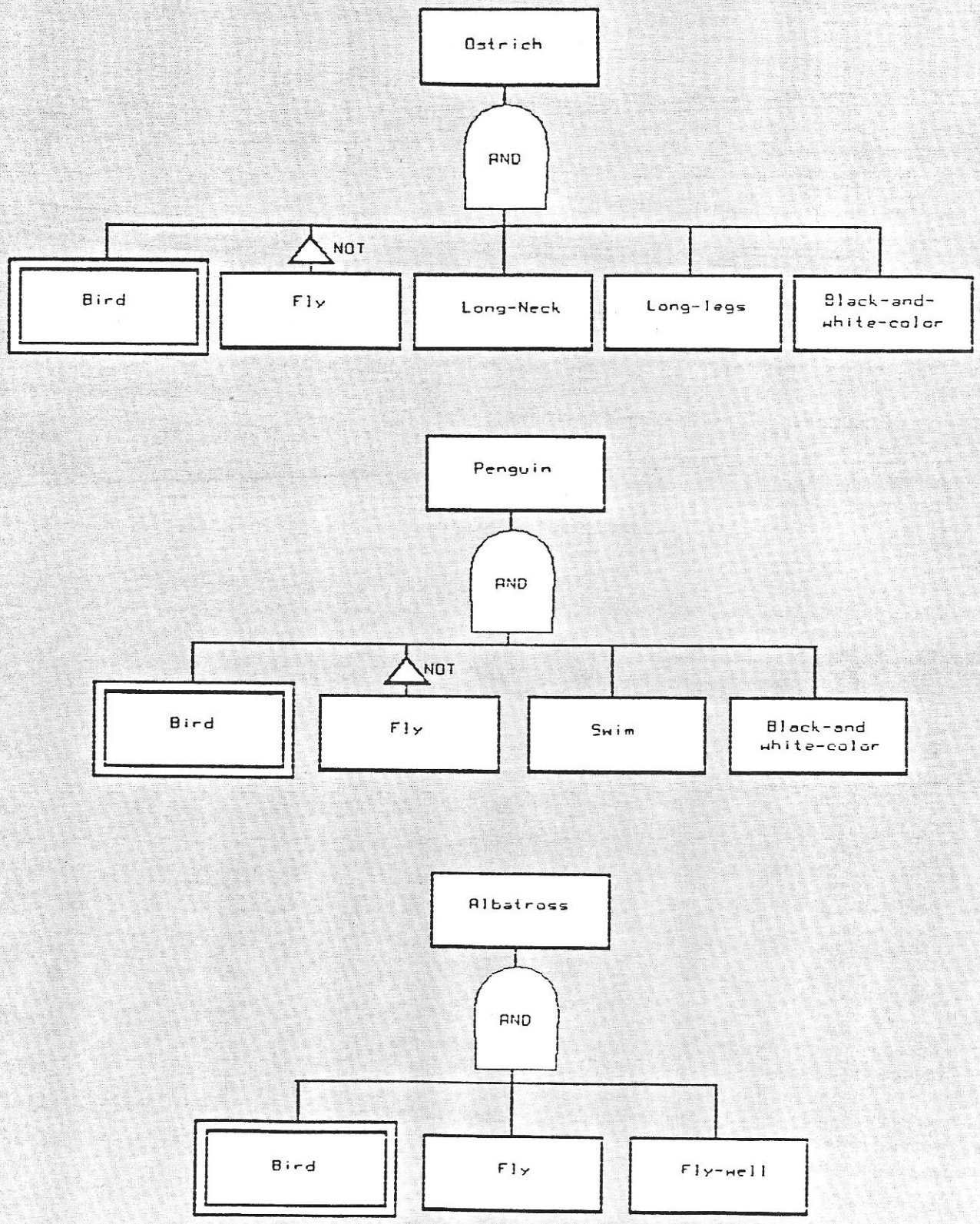


Fig. 1 Logical relationships describing animals to be identified.

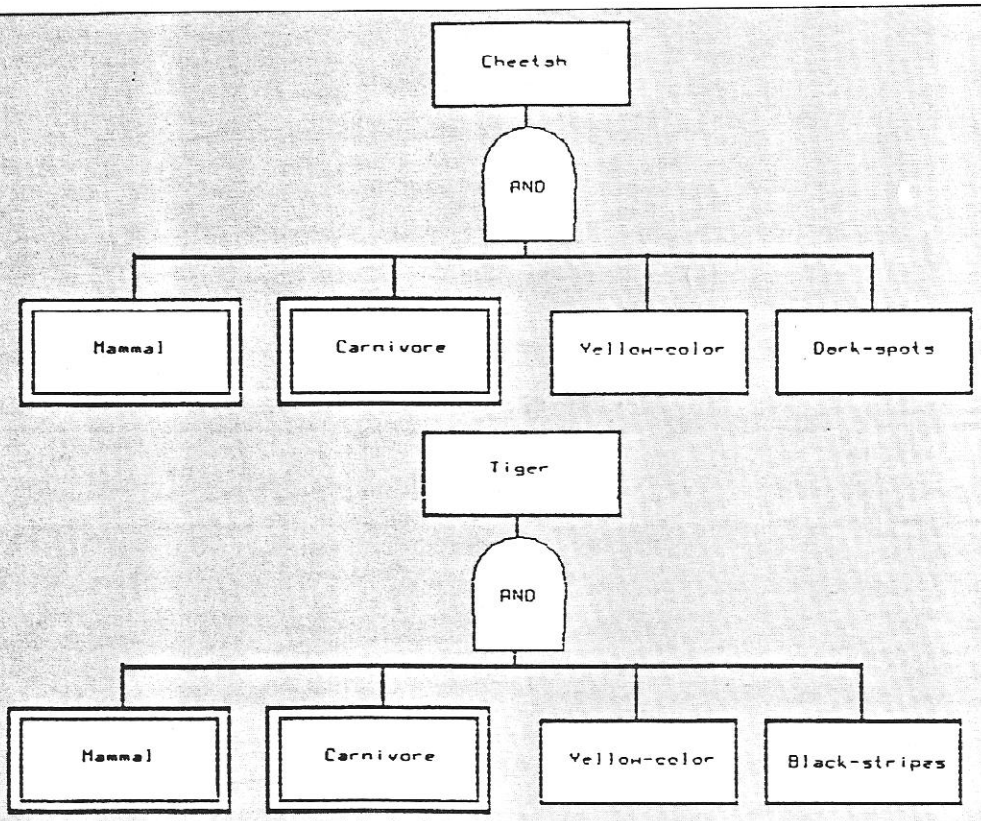


Fig.I (continued) Logical Relationships describing animals to be identified.

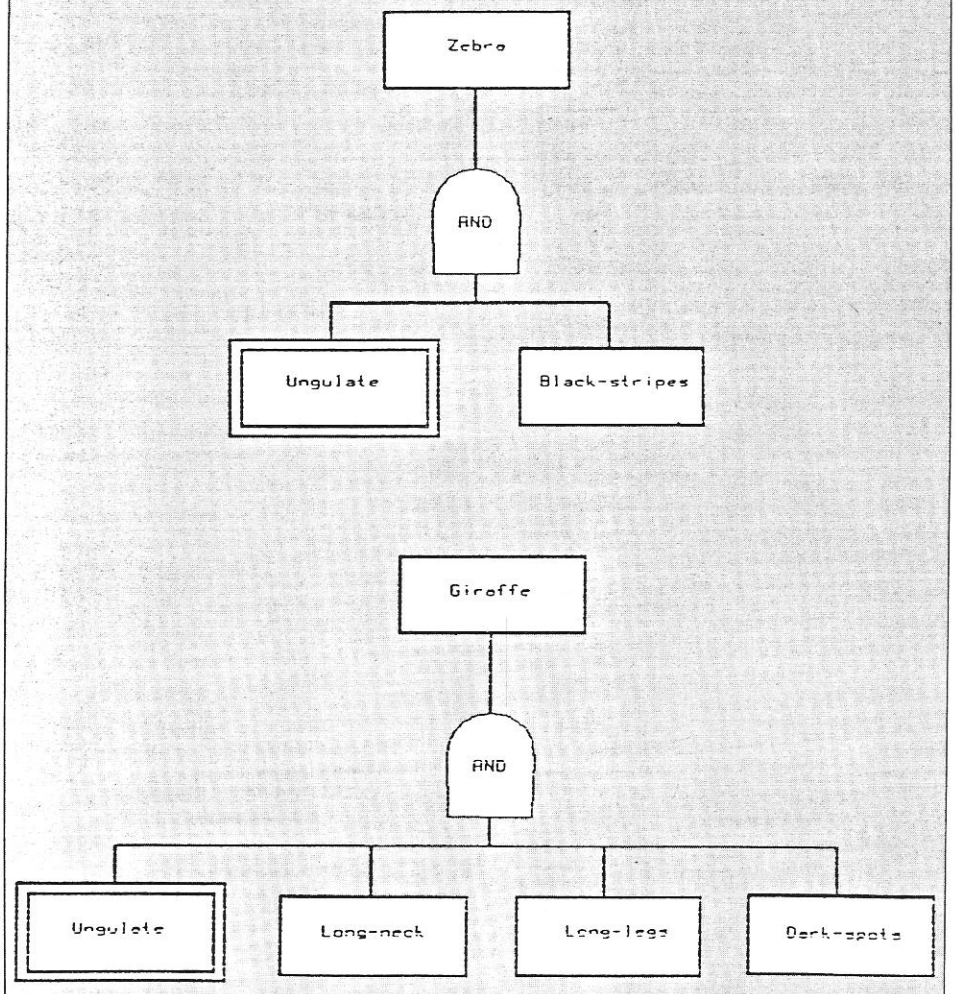
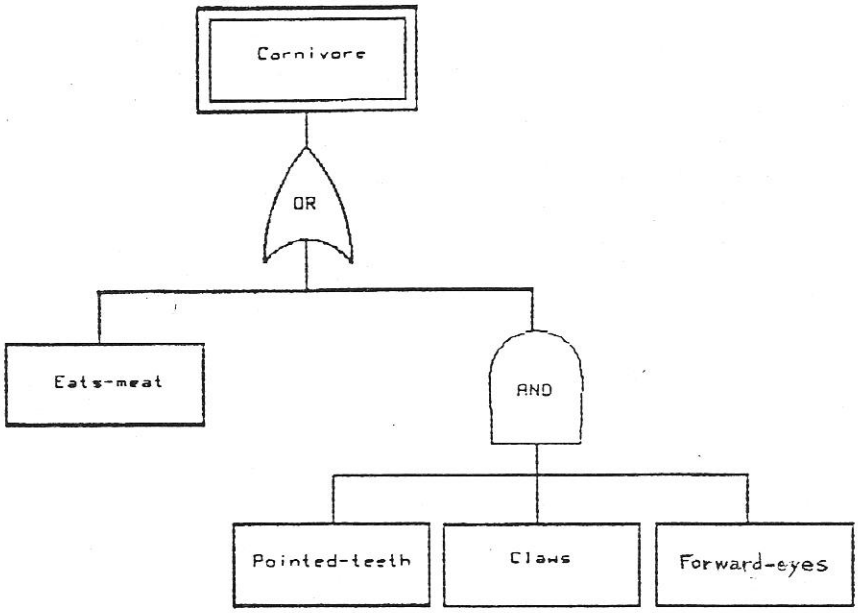


Fig. 1 (Continued) Logical relationships describing animals to be identified.

Information Requested

Inform the TI Professional Computing Community of your users group, newsletter and/or BBS. Send complete information to: Editor, TI Professional Computing, 12416 Hymeadow Drive, Austin, TX 78750.

Samples for inclusion in Users Group Notes are welcome.



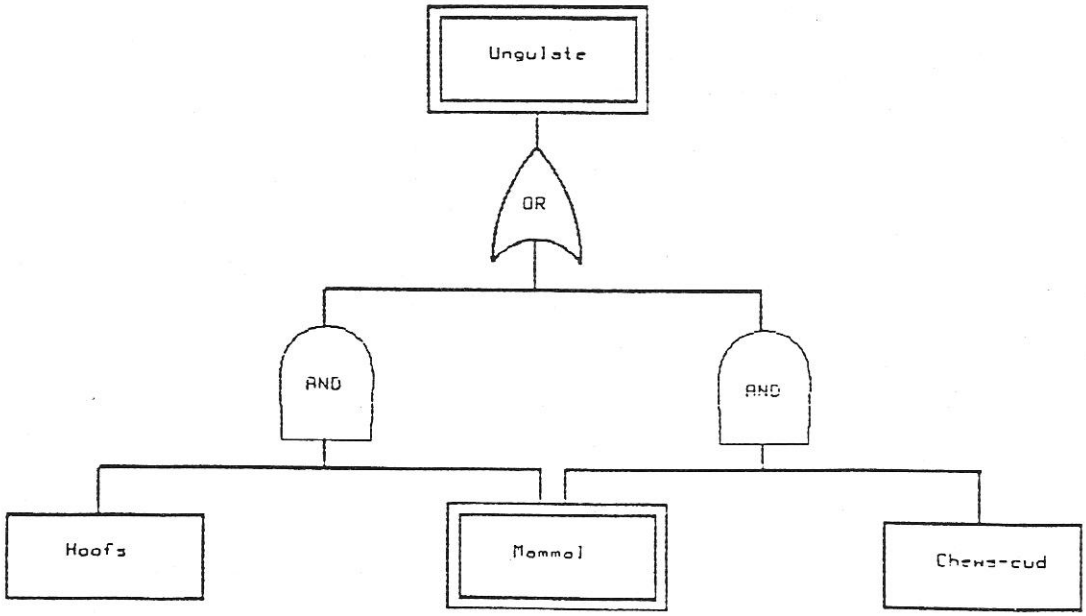


Fig. 2 Logical Descriptions of Intermediate Deductions.

Figure 2 continued on page 20

Figure 2 using the AND and OR logical gates. The identification problem them becomes:

Animal = Albatross + Penguin + Ostrich + Giraffe + Zebra + Tiger + Cheetah (3)

Substitution from Eqns. 1 and 2 into Eqn. 3 yields the logical expression for the goal or top-event of a logical tree:

Animal = ((Feathers + (Fly * Lay-eggs)) * Fly * Fly-well) + ((Feathers + (Fly * Lay-eggs)) * $\overline{\text{Fly}}$ * Swim * Black-and-white-color) + ((Feathers + (Fly * Lay-eggs)) * $\overline{\text{Fly}}$ * Long-Neck * Long-Legs * Black-and-white-color) + (((Hoofs * (Hair + Gives-milk)) + ((Hair + Gives-milk) * Chews-Cud)) * Long-Neck * Long-Legs * Dark Spots) + (((Hoofs * (Hair + Gives-milk)) + ((Hair + Gives-milk) * Chews-Cud)) * Black-Stripes) + ((Hair + Gives-milk) * (Eats-meal + (Pointed-teeth * Claws * Forward-eyes)) * Yellow-color * Black-stripes) + ((Hair + Gives-milk) * (Eats-meat + (Pointed-teeth * Claws * Forward-eyes) * Yellow-color * Dark-spots) (4)

This representation is shown in the form of a Goal Tree in Figure 3. This Goal Tree can then be translated into a Knowledge-Base and coupled to an Inference-Engine forming a Production-Rule system, as shown in Figure 4, in the form of a Prolog program.

ENHANCED PROLOG PRODUCTION-RULE SYSTEM

The prolog program is a direct translation of the logical gates representing the problem in Eqn. 4 and Figure 3. A typical AND gate from Figure 1 is written as:

```
animal-is (tiger):-
    identified-as (mammal),
    identified-as (carnivore),
    positive (characterized-by, yellow-color),
    positive (characterized-by, black-stripes).
```

The (:-) symbol can be considered as corresponding to (if) and the comma (,) for the (AND) logical operation.

A typical OR gate from Figure 4, is written as:

```
identified-as (mammal):-
    positive(characterized-by, hair).
identified-as (mammal):-
    positive(able-to, give-milk).
```

Enhancements for applications on the Explorer machine over Yazdani's version [2] are here discussed. The main enhancement to the program is accomplished by adding the statements:

```
abolish(pos,2),
abolish(neg,2).
```

which resets the memory to the original knowledge-base or long-term memory and prepares it for a new interactive inquiry by the user. The predicates (pos) and (neg) are created by the (asserta) predicate and added as new facts to the short-term memory part of the knowledge-base. A modification in the logic and the knowledge base, keeps the program from asking if the animal flies well, when it has been told that the animal doesn't even fly at all, in the case of the albatross.

The statement "asserta(X:-!)" gives a syntax error for version 1.0 of Prolog on the Explorer, so it was changed to "asserta(X)". The statement "add(X):-run:-asserta(X) also gave a syntax error on the Explorer, so it was changed to "add(X):-asserta(X). run:-. . ." Since the cut operator was deleted from the assertion, it had to be added to the positive (X) and negative (X) predicates elsewhere in the modified program. Also the names under which the assertions were stored was changed to keep the program from going into an infinite loop when "not(P)" is called. The predicate "not(P)" is a good example of the classical use of the cut (!) operator:

```
not(P):-call (P),!,fail.
not(P).
```

The first attempt at not(P) begins to succeed if P exists. The cut operator is then used to keep Prolog from backtracking to try to satisfy "not" by some other means. If call(P) succeeds, the first call to not(P) fails, and since the cut operator is used before it fails, Prolog cannot attempt the second not(P). If, however, call(P) fails, the cut operator is not encountered, and prolog is free to attempt as many other ways as it deems necessary to satisfy not(P), namely the second call to not(P) which succeeds by its sheer simplicity.

Through the use of the (pos) and (neg) predicates, the program "remembers" any positive or negative answers about its previous inquiries, which avoids asking the same question twice. It is also capable of indentifying cases of incomplete information, where the short-term memory is cleared, and the user is directed to start a new input by initiating the search using the "start." predicate.

TYPICAL CASE RESULTS

On the Explorer machine, the program was written and edited using the Z macs editor and stored as a file with the identifier (.pl). Figure 5 shows how to call the program, by first initializing (prolog) from the Lisp listener, then using consult or reconsult to call the program file. The program is initialized by entering "start." followed by a period, as required by Prolog, and then pressing the "return" key. Figures 6, 7, and 8 shows typical case results for the identification of a bird (penguin), a mammal (giraffe), and a carnivore

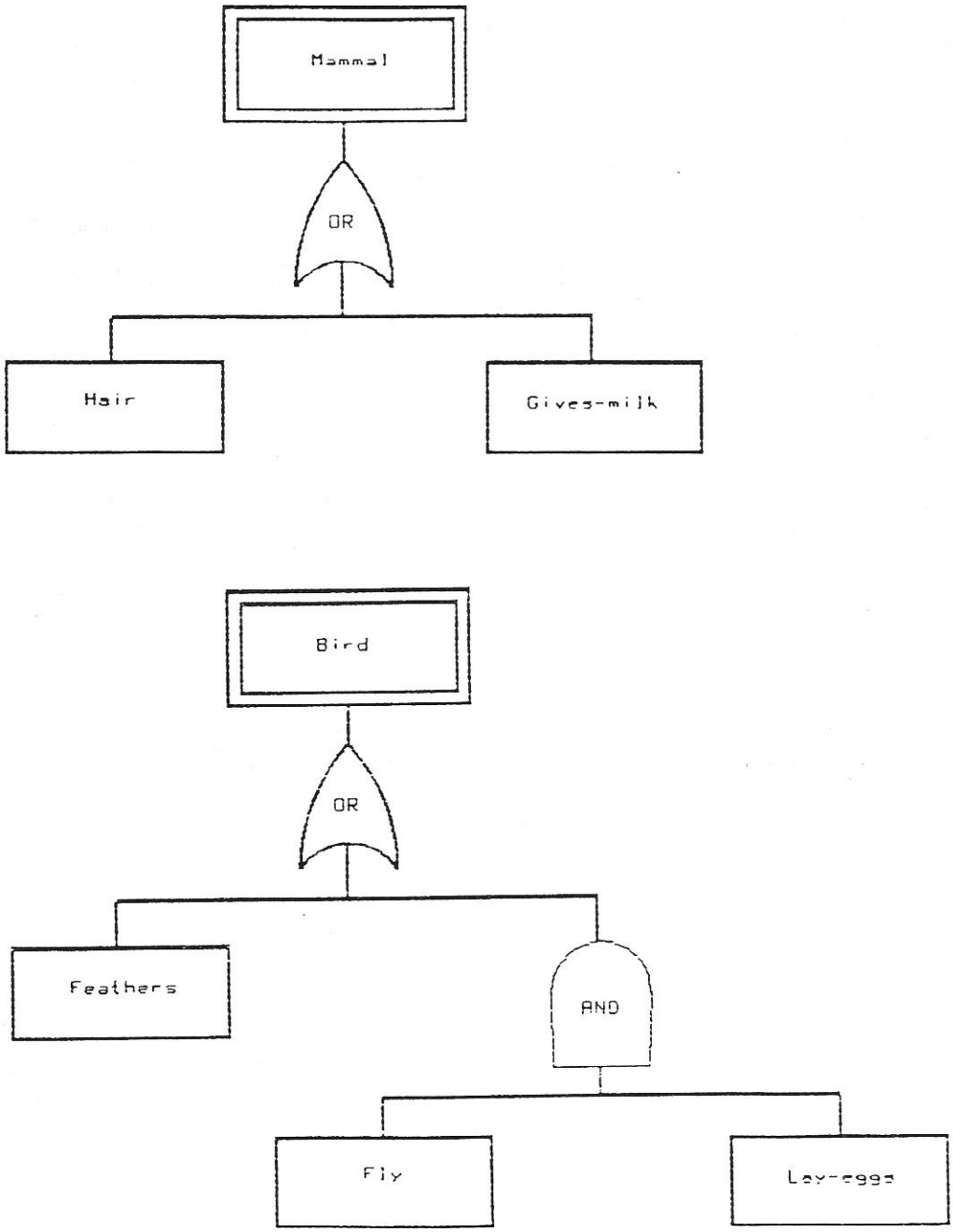


Fig. 2 (Continued) Logical Descriptions of Intermediate Deductions.

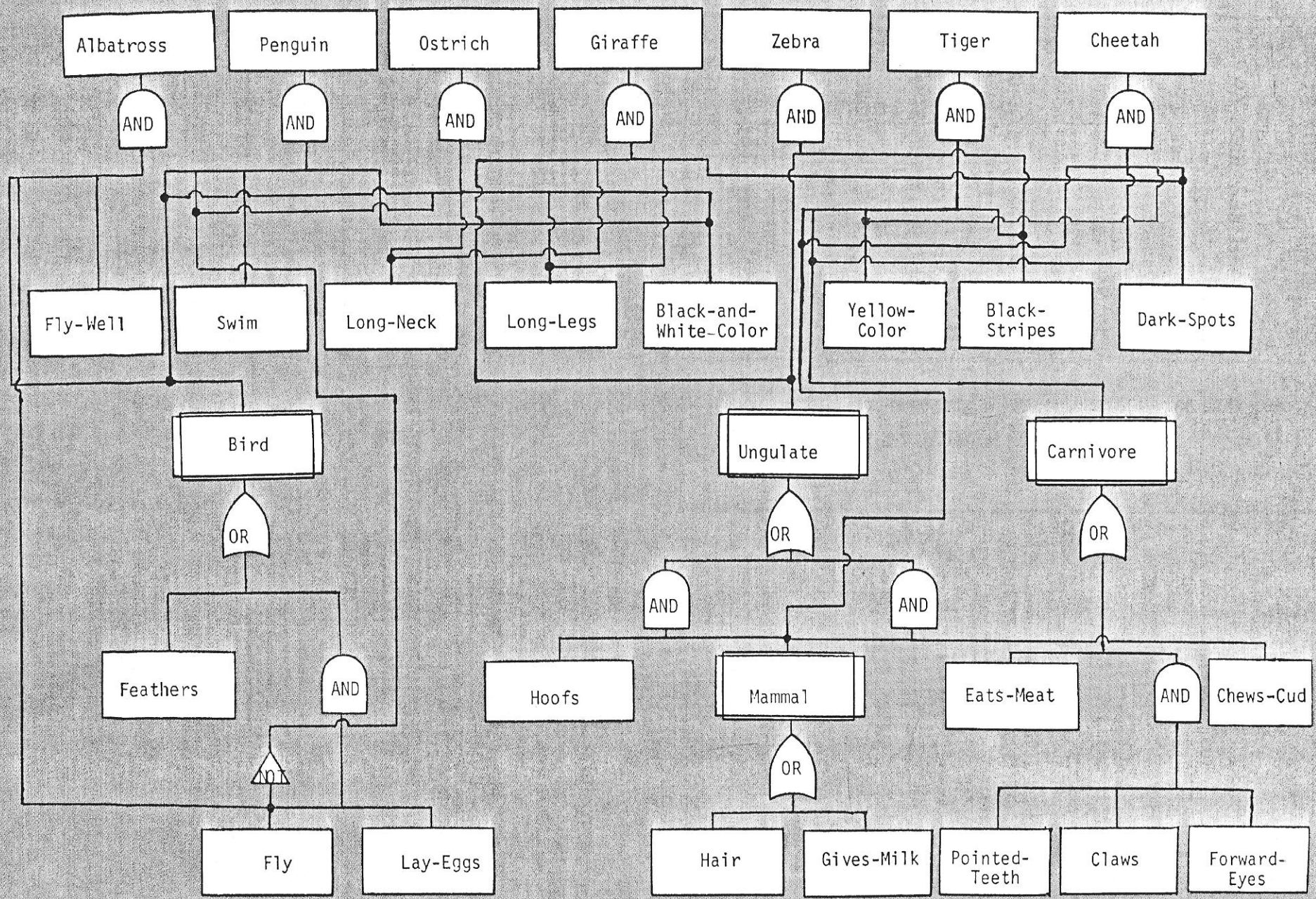


Fig. 3 Goal Tree representation of identification problem.

```
/*****************************************************************************/
/*                                                                           */
/*          ANIMAL IDENTIFICATION IN PROLOG FOR THE EXPLORER LISP MACHINE    */
/*                                                                           */
/*                          Magdi Ragheb and Dan Rehfeldt                    */
/*                                                                           */
/*                              Version 2.1, Feb. 1986                       */
/*                                                                           */
/*                  University of Illinois at Urbana-Champaign               */
/*                                                                           */
/*****************************************************************************/
/*                              KNOWLEDGE BASE                               */
/*                            Long-term memory                               */
/*****************************************************************************/

animal-is(albatross):-
        identified-as(bird),
        positive(able-to,fly),
        positive(able-to,fly-well).

animal-is(penguin):-
        identified-as(bird),
        negative(able-to,fly),
        positive(able-to,swim),
        positive(characterized-by,black-and-white-color).

animal-is(ostrich):-
        identified-as(bird),
        negative(able-to,fly),
        positive(characterized-by,long-neck),
        positive(characterized-by,long-legs),
        positive(characterized-by,black-and-white-color).

animal-is(giraffe):-
        identified-as(ungulate),
        positive(characterized-by,long-neck),
        positive(characterized-by,long-legs),
        positive(characterized-by,dark-spots).

animal-is(zebra):-
        identified-as(ungulate),
        positive(characterized-by,black-stripes).

animal-is(tiger):-
        identified-as(mammal),
        identified-as(carnivore),
        positive(characterized-by,yellow-color),
        positive(characterized-by,black-stripes).

animal-is(cheetah):-
        identified-as(mammal),
        identified-as(carnivore),
        positive(characterized-by,yellow-color),
        positive(characterized-by,dark-spots).

identified-as(ungulate):-
        identified-as(mammal),
        positive(characterized-by,hoofs).
identified-as(ungulate):-
        identified-as(mammal),
```

Fig. 4 Prolog-Explorer Version of the Animal Identification Problem.

```
        positive(able-to,chew-cud).

identified-as(mammal):-
        positive(characterized-by,hair).
identified-as(mammal):-
        positive(able-to,give-milk).

identified-as(bird):-
        positive(characterized-by,feathers).
identified-as(bird):-
        positive(able-to,fly),
        positive(able-to,lay-eggs).

identified-as(carnivore):-
        positive(able-to,eat-meat).
identified-as(carnivore):-
        positive(characterized-by,pointed-teeth),
        positive(characterized-by,claws),
        positive(characterized-by,forward-eyes).

/*****************************************************************************/
/*                              INFERENCE-ENGINE                             */
/*****************************************************************************/

positive(X,Y):-
        pos(X,Y),!.
positive(X,Y):-
        not(neg(X,Y)),nl,
        inquire(X,Y,Reply),
        check-yes(Reply).

negative(X,Y):-
        neg(X,Y),!.
negative(X,Y):-
        not(pos(X,Y)),nl,
        inquire(X,Y,Reply),
        check-no(Reply).

check-yes(yes).
check-no(no).

inquire(X,Y,Reply):-
        write( ' is the animal that you want me to identify ').
        write(X),
        write(' ').
        write(Y),
        write(' ? ').nl,nl,
        read(Reply),
        add-new-fact-to-knowledge-base(Reply,X,Y).

/**Adding newly-acquired facts to short-term memory**/

add-new-fact-to-knowledge-base(yes,X,Y):-
        asserta(pos(X,Y)).
add-new-fact-to-knowledge-base(no,X,Y):-
        asserta(neg(X,Y)).

not(P):-call(P),!,fail.
not(P).
```

Fig. 4 (Continued) Prolog-Explorer Version of the Animal Identification Problem.

```
start:-
        write('                    ANIMAL IDENTIFICATION PROBLEM IN PROLOG       ').nl.
        write('                            Version 2.1 Feb. 1986                 ').nl.
        write('                        Magdi Ragheb and Dan Rehfeldt             ').nl.
        write('                    University of Illinois at Urbana-Champaign    ').nl.
        write(' The knowledge I possess allows me to identify the following seven animals:
        nl,nl.
        write('                                    Albatross').nl.
        write('                                    Penguin').nl.
        write('                                    Ostrich').nl.
        write('                                    Giraffe').nl.
        write('                                    Zebra').nl.
        write('                                    Tiger').nl.
        write('                                    Cheetah').nl,nl.
        write('Please answer my questions with "yes." or "no.", without forgetting the').
             write('period, then press the return key. Let us start.').nl,nl.
        animal-is(Y).
        write('I can deduce from the information that you gave me that the animal you ').
        write('have on your mind is the "').
        write(Y).
        write('".').nl,nl,nl.
        write('Let us try another case by typing "start.", without forgetting the period.
        nl.
        write('then pressing the "return" key. Please go ahead.').nl.

        /**Clearing newly-acquired facts from short-term memory**/

        abolish(pos,2).
        abolish(neg,2).
start:-
        write('The information that you gave me is insufficient for identification of ').
        write('the animal on your mind. Try me another time by entering "start".').nl.
        write('followed by a period, and then press "return" on the keyboard. ').nl,nl.

        /**Clearing newly-acquired facts from short-term memory**/

        abolish(neg,2).
        abolish(pos,2).
```

Fig. 4 (Continued) Prolog-Explorer Version of the Animal Identification Problem.

```
> (prolog)

Prolog requires Zetalisp to be on. I am turning it on for you.
When you exit normally from Prolog, you will be returned to Common Lisp.

Explorer Prolog Version 1.0

| ?- reconsult('pl:raghab;identification.pl').

              Reconsulted pl:raghab;identification.pl  6K Characters

| Yes
| ?- start.
                 ANIMAL IDENTIFICATION PROBLEM IN PROLOG
                          Version 2.1 Feb. 1986
                       Magdi Ragheb and Dan Rehfeldt
                 University of Illinois at Urbana-Champaign

 The knowledge I possess allows me to identify the following seven animals:

                                Albatross
                                Penguin
                                Ostrich
                                Giraffe
                                Zebra
                                Tiger
                                Cheetah

Please answer my questions with 'yes.' or 'no.', without forgetting the
period, then press the return key. Let us start.

 Is the animal that you want me to identify characterized-by feathers ?

|:

Lisp Listener 1
```

```
Choose Variable Value
Menu Tools
Help

Done
Abort

Basic Menu

Clear Screen
Start Input Over
Forward
Backward
Forward Word
Backward Word
Beginning of Line
End of Line
Rubout Last Char
Delete Character
Retrieve Input Typed Las
Retrieve Last Kill
```

Fig. 5 Program Calling from the Lisp Listener on the Explorer Machine.

```
start.
                 ANIMAL IDENTIFICATION PROBLEM IN PROLOG
                          Version 2.1 Feb. 1986
                       Magdi Ragheb and Dan Rehfeldt
                 University of Illinois at Urbana-Champaign

 The knowledge I possess allows me to identify the following seven animals:

                                Albatross
                                Penguin
                                Ostrich
                                Giraffe
                                Zebra
                                Tiger
                                Cheetah

Please answer my questions with 'yes.' or 'no.', without forgetting the
period, then press the return key. Let us start.

 Is the animal that you want me to identify characterized-by feathers ?

|:yes.

 Is the animal that you want me to identify able-to fly ?

|:no.

 Is the animal that you want me to identify able-to swim ?

|:yes.

 Is the animal that you want me to identify characterized-by black-and-white-color ?

|:yes.
I can deduce from the information that you gave me that the animal you
have on your mind is the 'penguin'.

Let us try another case by typing 'start.', without forgetting the period,
then pressing the 'return' key. Please go ahead.

| Yes
| ?-

Lisp Listener 1
```

```
Choose Variable Value
Menu Tools
Help

Done
Abort

Basic Menu

Clear Screen
Start Input Over
Forward
Backward
Forward Word
Backward Word
Beginning of Line
End of Line
Rubout Last Char
Delete Character
Retrieve Input Typed Las
Retrieve Last Kill
```

Fig. 6 Typical Case for Identification of a Bird.

```
Please answer my questions with 'yes.' or 'no.', without forgetting the
period, then press the return key. Let us start.

 Is the animal that you want me to identify characterized-by feathers ?

|:no.
 Is the animal that you want me to identify able-to fly ?

|:no.
 Is the animal that you want me to identify characterized-by hair ?

|:yes.
 Is the animal that you want me to identify characterized-by hoofs ?

|:yes.
 Is the animal that you want me to identify characterized-by long-neck ?

|:yes.
 Is the animal that you want me to identify characterized-by long-legs ?

|:yes.
 Is the animal that you want me to identify characterized-by dark-spots ?

|:yes.
I can deduce from the information that you gave me that the animal you
have on your mind is the 'giraffe'.

Let us try another case by typing 'start.', without forgetting the period,
then pressing the 'return' key. Please go ahead.

| Yes
| ?-

Lisp Listener 1
```

```
Choose Variable Value
Menu Tools
Help

Done
Abort

Basic Menu

Clear Screen
Start Input Over
Forward
Backward
Forward Word
Backward Word
Beginning of Line
End of Line
Rubout Last Char
Delete Character
Retrieve Input Typed Las
Retrieve Last Kill
```

Fig. 7 Typical Case for Identification of a Mammal.

```
|:no.
 Is the animal that you want me to identify able-to fly ?

|:no.
 Is the animal that you want me to identify characterized-by hair ?

|:yes.
 Is the animal that you want me to identify characterized-by hoofs ?

|:no.
 Is the animal that you want me to identify able-to give-milk ?

|:yes.
 Is the animal that you want me to identify able-to chew-cud ?

|:no.
 Is the animal that you want me to identify able-to eat-meat ?

|:yes.
 Is the animal that you want me to identify characterized-by yellow-color ?

|:yes.
 Is the animal that you want me to identify characterized-by black-stripes ?

|:yes.
I can deduce from the information that you gave me that the animal you
have on your mind is the 'tiger'.

Let us try another case by typing 'start.', without forgetting the period,
then pressing the 'return' key. Please go ahead.

| Yes
| ?-

Lisp Listener 1
```

```
Choose Variable Value
Menu Tools
Help

Done
Abort

Basic Menu

Clear Screen
Start Input Over
Forward
Backward
Forward Word
Backward Word
Beginning of Line
End of Line
Rubout Last Char
Delete Character
Retrieve Input Typed Las
Retrieve Last Kill
```

Fig. 8 Typical Case for Identification of a Carnivore.

(tiger), respectively. In all cases, an answer of "yes." or "no.", followed by a period and pressing of the "return" key is needed for a successful interactive session.

CONCLUSIONS

We demonstrated the implementation of the animal identification problem using Prolog on the Explorer Lisp machine. This is a typical application of Prolog for building Production-Rule Analysis or Expert Systems. Prolog offers here a simple way of constructing a knowledge-base coupled to an inference-engine which possesses a high degree of simplicity. The animal identification problem is generic, and can be easily applied to the identification problems arising in a variety of fields [6].

Acknowledgements

The manuscript preparation by Mrs. Catherine Pritchard is much appreciated. This work was partially supported by funds from Texas Instruments Corp., The Illinois Power Company, and the Illinois Department of Nuclear Safety.

References

1 P.H. Winston and B.K.P. Horn, "Lisp," Second Edition, Addison-Wesley Publishing Company, Reading, Massachusetts, (1984).

2 M. Yazdani, "Knowledge Engineering in Prolog," in: Expert Systems, Principles and Case Studies, R. Forsyth, Ed., Chapman and Hall Computing, London, New York, (1984).

3 J.R. Ennals, "Beginning Micro-Prolog," Harper and Row Publishers, New York, Second Edition (1984).

4 P. Harmon and D. King, "Expert Systems, Artificial Intelligence in Business," John Wiley and Sons, Inc., New York (1985).

5 J. deKleer, "Book Review: E.A. Feigenbaum and P. McCorduck, The Fifth Generation," Artificial Intelligence: An International Journal, 22, p. 225 (1984).

6 M. Ragheb and D. Gvillo, "Development of Model-Based Fault-Identification Systems on Microcomputers," Proc. Conf. on "Applications of Artificial Intelligence III," SPIE Proceedings Vol. 635, SPIE's 1986 Technical Symposium on Optics and Optoelectronics, March 31-April 4, Orlando, Florida (1984).