Report 82-33
Stanford -- KSL

Scientific DataLink

Welcome to the MRS TUTOR!!!
Jan Clayton,
Nov 1982

card 1 of 1

Stanford Heuristic Programming Project
Memo HPP-82-33

November, 1982

WELCOME to the MRS TUTOR!!!

by

Jan Clayton

This tutor is designed to introduce you to the syntax and basic database accessing functions of MRS. This document is a transcript of an interaction with the MRS tutor.

Computer Science Department
Stanford University
Stanford, California 94305

Funding for this work was provided by ONR Contract N00014-81-K-0004.

Table of Contents

1. What is a Representation Language?

Representation languages provide a way to store and retrieve facts from a computer. Since English is a grammatically and textually ambiguous language, representation systems use a more formal language to describe the world. The way in which the words or symbols of a language are put together to form phrases and sentences is termed the "syntax" or "grammar" of the language. The next several sections of this tutorial will introduce you to the syntax of the MRS language.

1.1 The Syntax of MRS

MRS is a predicate calculus-like language. It is based upon combinations of "symbols" into "terms" and "propositions". Facts are stated as N-tuples or lists which describe a relationship between individuals. The statement

```
(NEIGHBOR BERTRAM ARTHUR)
```

consists of three symbols. Neighbor represents a relationship between the second and third symbols which represent objects (in this case people) in the world. The proposition above states that Bertram and Arthur are neighbors.

1.2 Types of Symbols

In general, symbols within MRS expressions can represent a number of things, including objects, functions, relations, actions, etc. More specifically, there are three general categories of symbols.

1. Object symbols name specific objects, concepts or individuals. Examples: STANFORD, KENNEDY, ELEPHANTS, BLUE (the color), JEALOUSY

2. Function symbols represent functions on the objects in the world. Examples: PRESIDENT-OF, HEIGHT-OF, +, COLOR-OF

3. Relation symbols represent relations between objects in the world. Examples: NEIGHBOR, >, TALLER-THAN, =

1.3 Terms

Sometimes we need to talk about an individual or object that doesn't have a name (or for whose name we don't know). For example, Fido's tail does not have a particular name or symbol associated with it, but we may need to refer to it. To solve this problem, symbols can be combined into more complex expressions to form functional "terms". Examples:

```
(TAIL-OF FIDO)
(PRESIDENT-OF STANFORD)
(+ 2 2)
(HEIGHT-OF (PRESIDENT-OF STANFORD))
```

Expressions of this type along with all symbols are called terms. More formally, any N-ary function symbol associated with N terms (Function term1 ... termN) is a term.

It is important to note that even though a functional term like (MOTHER-OF FRED) may be associated with an individual in the world named Sally, the database will not automatically know that fact.

1.4 Atomic Propositions

In general, a fact is stated within MRS in the form of a proposition which consists of any N-ary relation symbol and N terms, (relation term1 ... termN).

MRS allows the use of functional terms in stating facts about the world.

```
(= (MOTHER-OF FRED) SALLY)
(> (* 2 3) (+ 2 3))
(= (NAME (DOG-OF BEATRICE)) NOODLES)
```

1.5 Logical Propositions

Quite often facts cannot be stated as a direct fact like (NEIGHBOR BERTRAM ARTHUR). For example, you may want to say that Sally is not a neighbor of Bertram, or that if George fails the test he will be sad. Complex propositions such as negations, disjunctions, conjunctions and contingencies can be expressed as non-atomic propositions using the logical symbols NOT, OR, AND and IF respectively.

```
(NOT (NEIGHBOR SALLY GEORGE))
(IF (FAILED GEORGE TEST)
    (SAD GEORGE))
(OR (NEIGHBOR PALOALTO MENLOPARK)
    (NEIGHBOR SANFRANCISCO LOSANGELES))
```

1.6 Exercises

For the following problems, indicate which are Terms (T), Propositions (P), or Illegal (I) constructions according to the rules of MRS syntax.

BLOCK3
Term, Proposition or Illegal?

The answer expected was: T

All symbols are terms by definition.

OR
Term, Proposition or Illegal?

The answer expected was: T

All symbols are terms by definition.

(FRIEND-OF ZELDA)
Term, Proposition or Illegal?

The answer expected was: T

Although this has the same form as a proposition, it is a term because it represents an object (person, place or thing) and not a fact.

(SISTER-OF (NEIGHBOR-OF (BROTHER-OF ALFRED)))
Term, Proposition or Illegal?

The answer expected was: T

This expression is a term, not a proposition. It refers to a specific person or thing. And each argument also refers to a specific thing.

(INFRONTOF (BLOCK-NEXT-TO BLOCK2) (BLOCK-SUPPORTED-BY BLOCK2))
Term, Proposition or Illegal?

The answer expected was: P

This expression states a fact about the relationship between two things which are represented as terms. Facts about objects are always propositions.

(LOGICALOPERATOR NOT)
Term, Proposition or Illegal?

The answer expected was: P

Although Not is a reserved word in MRS, this does not mean that you can't make statements about it.

(LARGE BLOCK1)
Term, Proposition or Illegal?

The answer expected was: P

This is the simplest form of a proposition, a relation followed by symbols. It is not a term because it states a fact.

(CUBE (PYRAMID (BLOCK-SUPPORTED-BY BLOCK5)))
Term, Proposition or Illegal?

The answer expected was: I

All relations take terms as arguments. In this case, both arguments are propositions.

(NOT (BLOCK-SUPPORTED-BY BLOCK4))
Term, Proposition or Illegal?

The answer expected was: I

Logical operators must have propositional arguments. In this problem the arguments are terms.

(OR BLOCK2 BLOCK6)
Term, Proposition or Illegal?

The answer expected was: I

This is an illegal statement because logical operators must have propositions (facts) as their arguments. Symbols are terms, not propositions.

(NOT (= (COPY-OF BLOCK4) BLOCK2))
Term, Proposition or Illegal?

The answer expected was: P

This is a legal statement. Because it states a fact it is a proposition. "=" is a relation that takes two arguments with must be terms, and NOT must modify a proposition.

(CARPENTER LINDA (OR JOE BILL))
Term, Proposition or Illegal?

The answer expected was: I

There are two errors here. First of all relations have arguments which are terms. The second argument of this propositions starts out like it is going to be a logical proposition. Second, all logical operators must have propositional arguments. In this case the arguments are symbols.

```
(IF (SMALL BLOCK4))
Term, Proposition or Illegal?
```

The answer expected was: I

The Logical operator IF can only have two propositional arguments.

```
(AND (INFRONTOF (COPY-OF BLOCK5) BLOCK3) (CUBE (BLOCK-BENEATH BLOCK1)))
Term, Proposition or Illegal?
```

The answer expected was: P

This is a correct form of a proposition. AND can take any number of propositional arguments.

```
(NOT (BEHIND (NEIGHBOR-OF BLOCK2) BLOCK6) (CUBE BLOCK6))
Term, Proposition or Illegal?
```

The answer expected was: I

The Logical operator NOT can only have one propositional argument.

1.7 Quantified Propositions

Up to now we have only used symbols that are constants. Quite often, however, one needs to refer to arbitrary or unknown individuals. For example you might want to say "Somebody killed George's aunt" or "elephants are grey" To do this you need to use universally or existentially quantified symbols

```
(EXIST P (KILLED (AUNT-OF GEORGE) P))
(ALL E (IF (ELEPHANT E) (= (COLOR-OF E) GREY)))
```

The proposition (All x1 ... xN (prop x1 ... xN)) states that the proposition (prop x1 ... xN) is true for all possible values of x1 ... xN. The proposition (Exists x1 ... xN (prop x1 ... xN)) states that there is some x1 ... xN for which (prop x1 ... xN) is true. The All and Exist quantifiers can also be nested within each other or can be used with non-atomic propositions. For example, you can state "for all numbers there is another number that is greater in value" with the proposition

```
(ALL X (EXIST Y (> Y X)))
```

Each quantifier can also take multiple variables as arguments.

```
(ALL H R (IF (AND (HORSE H) (RABBIT R))
             (CAN-OUTRUN H R)))
```

Be careful of your use of nested quantifiers. A change in their position in respect to each other can change the meaning of the proposition drastically. For example the two propositions

```
(ALL X (EXISTS Y (LOVES X Y)))
(EXIST Y (ALL X (LOVES X Y)))
```

mean "Every person loves someone" and "There is a person that is loved by everyone" respectively.

1.8 Free Variables in Propositions

You may use a shorthand for specifying variables. The prefix character $ denotes a universally quantified variable when entering facts. To illustrate this shorthand notation the following example shows equivalent statements.

```
(ALL E (IF (ELEPHANT E) (GREY E)))

(IF (ELEPHANT $E) (GREY $E))
```

1.9 Exercises

```
    For the following problems, indicate which are Propositions (P),
    or Illegal (I) constructions according to the rules of MRS syntax.

(EXIST X (BEHIND X BLOCK3))
Proposition or Illegal? (P or I)

    The answer expected was: P

    This is the proper form of a proposition with a quantifier.  It's last
    argument states a fact.


(EXIST JOE (BROTHER-OF (PARENT-OF SALLY)) GEORGE)
Proposition or Illegal? (P or I)

    The answer expected was: I
```

Quantifiers (ALL and EXIST) must have one or more variables to quantify, as well as a proposition. All the arguments in the example above are terms, and although symbols can double as variables, complex or functional terms cannot. Finally, the last argument must be a proposition.

```
(ALL X (BLUE X) (PYRAMID BLOCK3))
Proposition or Illegal? (P or I)
```

The answer expected was: I

A quantifier can only quantify one proposition.

```
(ALL FROGS (INSTACK (BLOCK-BENEATH BLOCK6)
                    (BLOCK-BENEATH BLOCK2)
                    (BLOCK-SUPPORTED-BY BLOCK4)))
Proposition or Illegal? (P or I)
```

The answer expected was: P

The variables that a quantifier quantifies do not have to be found in the following proposition. (But it makes more sense if they do!)

```
(ALL BLOCK1 BLOCK3)
Proposition or Illegal? (P or I)
```

The answer expected was: I

The last argument to a quantifier must be a proposition. In this case the last argument is a term, not a proposition.

```
(EXIST X5 Y (OR (RED Y) (AND (CUBE X5)
                             (INSTACK BLOCK6 BLOCK5 BLOCK1)
                             (INFRONTOF BLOCK1 BLOCK2))))
Proposition or Illegal? (P or I)
```

The answer expected was: P

If takes 2 propositional arguments. Both of these arguments are legal propositions..

```
(NOT (OR (BEHIND BLOCK3 BLOCK1)
         (ONTOPOF BLOCK1 BLOCK4)
         (SMALL BLOCK4)))
Proposition or Illegal? (P or I)
```

The answer expected was: P

This is a legal logical proposition.

(ALL (TALLERTHAN SALLY (FRIEND-OF VIRGINIA)))
Proposition or Illegal? (P or I)

The answer expected was: I

A quantifier must have at least one variable to quantify.

(ALL)
Proposition or Illegal? (P or I)

The answer expected was: I

A quantifier must have at least one variable and one proposition as arguments.

(ALL X3 Z (TALLERTHAN (FRIEND-OF X3) (PARENT-OF Z)))
Proposition or Illegal? (P or I)

The answer expected was: P

As many variables as needed can be quantified with one quantifier.

(EXIST Y (SISTER-OF Y))
Proposition or Illegal? (P or I)

The answer expected was: I

At first glance, this statement looks legal within the constraints of the MRS syntax. However, under normal interpretations the last argument refers to a thing, and because of that, it is a term, not a proposition.

(INSTACK (ALL HORSES) BLOCK1)
Proposition or Illegal? (P or I)

The answer expected was: I

Although quantifiers do not have to be on the outermost level of a proposition, they must quantify a proposition. (All Horses) is the illegal part of this statement.

2. Entering and Accessing Facts

Now that we know how to describe facts, it is necessary to know how to tell MRS that you want to enter or remove a fact from the system. MRS provides the users with several subroutines to access and modify the data base.

2.1 The $STASH Command

A Simple way of entering facts into a data base is with the $STASH command. ($STASH <p>) enters the proposition <p> directly into the current data base. (Below and in subsequent examples MRS is doing the actual evaluation of the typed expressions.)

```
USER: ($STASH '(NEIGHBOR BEATRICE BERTRAM))
MRS:  (NEIGHBOR BEATRICE BERTRAM)

USER: ($STASH '(NEIGHBOR BEATRICE SALLY))
MRS:  (NEIGHBOR BEATRICE SALLY)
```

2.2 $LOOKUP

($LOOKUP <P>) checks to see whether the proposition <P> has been entered in to the current data base. $LOOKUP makes no inferences on other propositions in the database, but "looks up" <p> to see if it has been entered. If <P> contains any free variables, the valued returned from $LOOKUP will contain a list of variable bindings that satisfy <P>. If <P> is free of variables and <P> is in the data base, the result will be ((T . T)).

The following facts are in the data base:

```
        (NEIGHBOR BEATRICE BERTRAM)
        (NEIGHBOR BEATRICE SALLY)

USER: ($LOOKUP '(NEIGHBOR BEATRICE BERTRAM))
MRS:  ((T . T))

USER: ($LOOKUP '(NEIGHBOR BEATRICE $X))
MRS:  (($X . SALLY) (T . T))
```

The command $LOOKUPS is an extension of $LOOKUP. Like $LOOKUP, it looks to see whether <p> is in the data base. The major difference between the two is when free variables are used, $LOOKUPS returns ALL the bindings for which <P> is true.

```
USER: ($LOOKUPS '(NEIGHBOR BEATRICE BERTRAM))
```

```
MRS:  (((T . T)))

USER: ($LOOKUPS '(NEIGHBOR BEATRICE $X))
MRS:  ((($X . SALLY) (T . T)) (($X . BERTRAM) (T . T)))

USER: ($LOOKUPS '(NEIGHBOR $Y BERTRAM))
MRS:  ((($Y . BEATRICE) (T . T)))
```

2.3 Removing Facts with $UNSTASH

Individual propositions in the data base can be removed with the $UNSTASH command.

The following facts are in the data base:

```
        (NEIGHBOR BEATRICE BERTRAM)
        (NEIGHBOR BEATRICE SALLY)

USER: ($UNSTASH '(NEIGHBOR BEATRICE BERTRAM))
MRS:  (NEIGHBOR BEATRICE BERTRAM)

USER: ($LOOKUP '(NEIGHBOR BEATRICE BERTRAM))
MRS:  NIL

USER: ($LOOKUPS '(NEIGHBOR BEATRICE $X))
MRS:  ((($X . SALLY) (T . T)))
```

2.4 $ASSERT

Usually a user will not only want to have facts stored in the database, but will also want to enable the system to do a certain amount of reasoning about those facts at a later time. ($ASSERT <p>) enters a proposition and performs all appropriate forward inference on that fact. In the initial start-up system of MRS $STASH and $ASSERT are equivalent.

```
USER: ($ASSERT '(APPLE FRED))
MRS:  (APPLE FRED)

USER: ($ASSERT '(IF (APPLE $X) (RED $X)))
MRS:  (IF (APPLE $X) (RED $X))
```

2.5 $TRUEP

($TRUEP <p>) attempts to infer whether the proposition <p> is true. <p> does not need to be in the database verbatim for $TRUEP to return ((T . T)) as was the case for $LOOKUP. Instead it has the ability to "infer" from other propositions that <p> is true. If existential variables are used in <p>, then $TRUEP will return a list of bindings for $X that satisfy <p>.

The following facts are in the database:

```
          (APPLE FRED)
          (IF (APPLE $X) (RED $X))

USER: ($TRUEP '(APPLE FRED))
MRS:  ((T . T))

USER: ($LOOKUP '(APPLE FRED))
MRS:  ((T . T))

USER: ($TRUEP '(RED FRED))
MRS:  ((T . T))

USER: ($LOOKUP '(RED FRED))
MRS:  NIL

USER: ($TRUEP '(RED $X))
MRS:  (($X . FRED) (T . T))
```

From the example you can see that $TRUEP is a much more powerful command than $LOOKUP. In addition, there is a extension to $TRUEP called $TRUEPS which returns all bindings for an existential variable in <p> that satisfy <p>.

```
USER: ($ASSERT '(RED BARNEY))
MRS:  (RED BARNEY)

USER: ($TRUEPS '(RED BARNEY))
MRS:  (((T . T)))

USER: ($TRUEPS '(RED $W))
MRS:  ((($W . BARNEY) (T . T)) (($W . FRED) (T . T)))
```

2.6 $UNASSERT

$ASSERT also has a partner command, ($UNASSERT <p>), which not only removes <p> from the data base, but also undoes any forward inferences that might have been made when the proposition was originally entered.

Facts in the data base are:

```
          (APPLE FRED)
          (IF (APPLE $X) (RED $X))
          (RED BARNEY)

USER: ($UNASSERT '(IF (APPLE $X) (RED $X)))
MRS:  (IF (APPLE $X) (RED $X))

USER: ($TRUEP '(RED FRED))
```

```
MRS:  NIL

USER: ($TRUEPS '(RED $WHO))
MRS:  ((($WHO . BARNEY) (T . T)))
```

2.7 Exercises

In the problems below there are a set of propositions that are in the data base. For the command, indicate, with either Y or N, whether the given response is what MRS will return.

```
(IF (FROG $X) (GREEN $X))
         (FROG FRED)
  User:  ($LOOKUP '(GREEN FRED))
  MRS:   ((T . T))
The right response? (Y or N)
```

The answer expected was: N

$LOOKUP only looks for a fact that is in the data base. No deduction (backward chaining) is performed.

```
(If (Frog $X) (Green $X))
         (Frog Fred)
  User:  ($Truep '(Green Fred))
  MRS:   ((T . T))
The right response? (Y or N)
```

The answer expected was: Y

$TRUEP backward chains through the data base to deduce the fact if it is not in the data base.

```
(If (Bird $X) (Blue $X))
         (If (Canary $X) (Bird $X))
         (Canary Tweety)
         (Canary Twiddle)
  User:  ($Truep (Blue $Y))
  MRS:   ((($Y . Tweety) (T . T)) (($Y . Twiddle) (T . T)))
The right response? (Y or N)
```

The answer expected was: N

If one asks for $TRUEP to return a binding for a variable, it will only return the first binding it finds in the database that satisfies the proposition.

```
(If (Bird $X) (Blue $X))
         (Bird Tweety)
```

```
          (Bird Twiddle)
  User:   ($Trueps '(Blue $Y))
  MRS:    ((($Y . Tweety) (T . T)) (($Y . Twiddle) (T . T)))
The right response? (Y or N)
```

The answer expected was: Y

The only difference between $TRUEP and $TRUEPS is that $TRUEPS will return all bindings for variables.

3. Using Inference

In past units, we have refered to inference a number of times without explaining what is meant by the word. Formally, inference refers to the act of deriving a conclusion in logic by either induction or deduction.

In MRS, conclusions are derived from facts and rules that are already in the database. There are a number of techniques that can be used to infer new facts; the two presented here are backward and forward chaining since those are the two most heavily used inference techniques in rule-based systems.

It is important to state that inference techniques are independent of any particular representation or methods of retrieving facts. Both of the chaining techniques mentioned above are used with many different types of representations: frames, semantic nets, and production systems to name a few.

3.1 $TRUEP and Backward Chaining

As mentioned previously, if a fact is not found directly in the database, $TRUEP will attempt to infer the fact from others that are in the database. The type of inference used is backward chaining (goal-directed reasoning).

```
(IF (ELEPHANT $X) (GREY $X))
(ELEPHANT CLYDE)
```

If these two assertions are in the database, and we ask "Is Clyde grey?" [($TRUEP '(GREY CLYDE))], MRS will first look for the fact (GREY CLYDE). When its search is unsuccessful, MRS will look for implications that might be able to conclude that Clyde is grey (ALL X (IF (...) (GREY X)). In practice, MRS tries to match the right-hand side of all implications with (GREY CLYDE).

In this particular instance, MRS finds the implication (IF (ELEPHANT $X) (GREY $X)) and then tries to prove that the left-hand side of the implication (ELEPHANT CLYDE) is true. Since that assertion is in the database, Truep returns ((T . T)) which is the truth value.

Backward chaining does not necessarily stop after examining one level of implications. If (ELEPHANT CLYDE) had not been in the database, MRS would have looked for implications that might be able to prove that Clyde is an elephant.

Below is a more extensive example of backward chaining, where the queries that $TRUEP makes in trying to prove a fact are traced. The following assertions are assumed in the database.

```
        (IF (AND (PLANT $P) (PURPLE $P)) (POISONOUS $P))
        (IF (MUSHROOM $M) (PLANT $M))
        (IF (TREE $T) (PLANT $T))
        (IF (VIOLET $X) (PURPLE $X))
        (VIOLET PHIL)
        (MUSHROOM PHIL)
USER: ($TRUEP '(POISONOUS PHIL))
MRS:
(1 ENTER TRUEP ((POISONOUS PHIL)))
  (2 ENTER TRUEP ((AND (PLANT PHIL) (PURPLE PHIL))))
    (3 ENTER TRUEP ((TREE PHIL)))
    (3 EXIT TRUEP NIL)
    (3 ENTER TRUEP ((MUSHROOM PHIL)))
    (3 EXIT TRUEP ((T . T)))
    (3 ENTER TRUEP ((PURPLE PHIL)))
      (4 ENTER TRUEP ((VIOLET PHIL)))
      (4 EXIT TRUEP ((T . T)))
    (3 EXIT TRUEP ((T . T)))
  (2 EXIT TRUEP ((T . T)))
(1 EXIT TRUEP ((T . T))) ((T . T))
```

3.2 $ASSERT and Forward Chaining

Forward chaining is another type of reasoning that MRS can perform. As the name implies, forward chaining reasons in the opposite of backward chaining. Instead of being "goal-directed", forward chaining is "data-directed", that is initiated and driven by the addition of new facts.

If a new fact is asserted (when forward chaining is turned on), MRS will automatically try to prove as many facts as possible from this new assertion. The forward chaining mechanism attempts to match the "left side" of each implication with the new fact, and if successful will assert the proposition of the right side.

Let's say that the following facts are in the data base:

```
(IF (ELEPHANT $X) (GREY $X))
(IF (ELEPHANT $X) (PACHYDERM $X))
```

If we then assert (ELEPHANT CLYDE), MRS will automatically assert both (GREY CLYDE) and (PACHYDERM CLYDE).

Note: If (IF (ELEPHANT $X) (VEGETARIAN $X)) is added later, the fact (VEGETARIAN CLYDE) will not automatically be asserted! In the case of asserting a new implication, MRS will not try to forward chain on the left side of the new implication, just on the entire implication. This is not an inherit insufficiency of forward chaining, but it is a reasoning limitation within the current MRS implementation.

Note also that forward chaining is not turned on in the initial MRS system. To do so, give the MRS command ($ASSERT '(MYTOASSERT $X FS-ASSERT)).

The following trace of MRS forward chaining uses the assertions below.

```
            (IF (AND (PLANT $P) (PURPLE $P)) (POISONOUS $P))
            (IF (TOADSTOOL $X) (PLANT $X))
            (PURPLE PHIL)
            (PURPLE FRED)

USER: ($ASSERT '(TOADSTOOL PHIL))
MRS:
(1 ENTER ASSERT ((TOADSTOOL PHIL)))
  (2 ENTER ASSERT ((PLANT PHIL)))
    (3 ENTER ASSERT ((POISONOUS PHIL)))
    (3 EXIT ASSERT |(POISONOUS PHIL)|)
  (2 EXIT ASSERT |(PLANT PHIL)|)
(1 EXIT ASSERT |(TOADSTOOL PHIL)|) (TOADSTOOL PHIL)
```

3.3 Forward vs. Backward Chaining

From the previous sections on Forward and Backward chaining, it is hard to see any advantage of one over the other. From the examples they appear to be almost exactly the same. What are the differences?

The most obvious difference is that forward chaining occurs when you are building the database. There is no need to query the database to get the inference mechanism to work. On the otherhand, backward chaining occurs only when a query takes place.

Although the above difference may not influence the choice between the use of the two reasoning schemes, the following should. The major difference between the forward and backward chaining has to do with the shape of the database that they search best. If a database has many initial facts with implications that determine very few goal states (fan-in), it is best to use forward chaining. However, if your database can have a large number of possible conclusions with a small set of initial facts, backward chaining is the inference method of choice. Why? Because in each case it is best to limit the amount of time infering facts that will never be used.

The following example should shed some light on the subject.

```
(IF (ELEPHANT $X) (PACHYDERM $X))
(IF (ELEPHANT $X) (BIG $X))
(IF (BIG $X) (HEAVY $X))
(IF (ELEPHANT $X) (STRONG $X))
```

If we assert the fact (ELEPHANT CLYDE) with the above assertions in the database, a forward chainer would immediately assert four additional facts, (PACHYDERM CLYDE), (BIG CLYDE), (HEAVY CLYDE) and (STRONG CLYDE). If (ELEPHANT EDWARD) is asserted, four additional facts will be put in the database. There are many possible conclusions for each new fact in this this database (it has the characteristic of fan-out).

A backward chainer in this situation would not asserted any new facts when (ELEPHANT CLYDE) is added, but would create new facts only if a query is made. To find out (STRONG CLYDE) there is no need to know (HEAVY CLYDE), etc., and the backward chainer will not try to prove these facts.

It is evident from this example that backward chaining is best to use with a database like the one above.

The similarities between the two mechanisms are also important. Both forward and backward chaining use the same rule of inference, Modus Ponens. This method of inference is also entirely separate from the method of search used to find facts in the database.

Caution! There are possible problems that can arise using chaining. One can construct a database that will reason in loops. MRS, at this time, is unable to stop circular reasoning, so beware of situations like the following.

For forward chaining:

```
1)  (IF (AND (INTEGER $X) (INTEGER $Y)) (INTEGER (+ $X $Y)))
2)  (IF (P A) (P B))
    (IF (P B) (P C))
    (IF (P C) (P A))
```

For backward chaining:

```
1)  (IF (AND (INTEGER $X) (INTEGER (+ $Y $X))) (INTEGER $Y))
2)  (IF (P A) (P B))
    (IF (P B) (P C))
    (IF (P C) (P A))
```

All analogous situations will cause MRS to loop indefinitely.

3.4 Interacting with MRS

Using the relations (Frog x), (Hates a b), (Giraffe x), (Herbivore x), (Vegetarian x) meaning "x is a Frog", "a hates b", etc., enter the following statements as propositions into MRS.

```
Fred is a frog.
MRS statement: ($STASH '(FROG.FRED))
(FROG FRED)

Sally is a frog.
MRS statement: ($STASH '(FROG SALLY))
(FROG SALLY)

Sally hates Fred.
MRS statement: ($STASH '(HATES SALLY FRED))
(HATES SALLY FRED)

Fred hates everything.
MRS statement: ($STASH '(HATES FRED $X))
(HATES FRED $X)

Frogs do exist.
MRS statement: ($STASH '(FROG ?X))
(FROG ?X)

Is Sally a frog?
MRS statement: ($LOOKUP '(FROG SALLY))
((T . T))

Is George a frog?
MRS statement: ($LOOKUP '(FROG GEORGE))
NIL

Does George hate Sally?
MRS statement: ($LOOKUP '(HATES GEORGE SALLY))
NIL

Who hates Fred?
MRS statement: ($TRUEP '(HATES $X FRED))
(($X . FRED) (T . T))

Who are all the frogs?
MRS statement: ($TRUEPS '(FROG $X))
((($X . ?X) (T . T)) (($X . SALLY) (T . T)) (($X . FRED) (T . T)))

George is a giraffe.
MRS statement: ($STASH '(GIRAFFE GEORGE))
(GIRAFFE GEORGE)
```

```
All Giraffes are herbivores.
MRS statement: ($STASH '(IF (GIRAFFE $X)(HERVIVORE $X)))
(IF (GIRAFFE $X) (HERVIVORE $X))

All herbivores are vegetarians.
MRS statement: ($STASH '(IF (HERBIVORE $X) (VEGETARIAN $X))))
(IF (HERBIVORE $X) (VEGETARIAN $X))

Is George a herbivore?
MRS statement: ($TRUEP '(HERBIVORE GEORGE))
((T . T))

Gertrude is a herbivore.
MRS statement: ($STASH '(HERBIVORE GERTRUDE))
(HERBIVORE GERTRUDE)

Who is a vegetarian?
MRS statement: ($TRUEPS '(VEGETARIAN $X))
((($X . GERTRUDE) (T . T)) (($X . GEORGE) (T . T)))
```