Report 82-27
Stanford -- KSL

Scientific DataLink

An Overview of MRS for AI Experts.
Michael R. Genesereth,
Jan 1984

card 1 of 1

Stanford Heuristic Programming Project
Memo HPP-82-27

First Version October 1982
Current Version January 1984

An Overview of MRS for AI Experts

by

Michael R. Genesereth

Department of Computer Science
School of Humanities and Sciences
Stanford University

MRS is a knowledge representation system intended for use by AI researchers in building expert systems. It offers a diverse repertory of commands for asserting and retrieving information, with various representations (e.g. property lists, propositional representation), various inference techniques (e.g. backward chaining, forward chaining, and resolution, all with optional caching and truth maintenance), and various search strategies (e.g. depth-first, breadth-first, and best-first search). The initial system includes a vocabulary of concepts and facts about logic, sets, mappings, arithmetic, and procedures.

What differentiates MRS from many other knowledge representation systems is its ability to observe, reason about, and control its own activity. In MRS the system is treated as a domain in its own right. One can write sentences about subroutines and other sentences and allow the system to reason with them, just as it reasons about geology or medicine. In practice, MRS uses this "meta-level" information in deciding how to satisfy its users' goals. Thus, one can switch representations or inference methods by changing these sentences, and one can easily implement a variety of different control schemes.

This document is an overview of the more important features of MRS. Note that it is not a manual or a users' guide, and it is not intended to be comprehensive, only suggestive. For fuller details, the reader should consult the MRS Dictionary.

BLANK PAGE

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                    Data Base
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(STASH '(FATHER ARTHUR BEATRICE))
P268

(STASH '(FATHER ARTHUR BERTRAM))
P269

(STASH '(FATHER BERTRAM CORNISH))
P270

(STASH '(MOTHER BEATRICE CAREY))
P271

(LOOKUP '(FATHER $X BERTRAM))
(($X . ARTHUR) (T . T))

(LOOKUP '(FATHER ARTHUR $Y))
(($Y . BERTRAM) (T . T))

(LOOKUPS '(FATHER ARTHUR $Y))
((($Y . BERTRAM) (T . T)) (($Y . BEATRICE) (T . T)))

(LOOKUPS '(FATHER $X $Y))
((($Y . CORNISH) ($X . BERTRAM) (T . T))
 (($Y . BERTRAM) ($X . ARTHUR) (T . T))
 (($Y . BEATRICE) ($X . ARTHUR) (T . T)))

(PRFACTS 'BERTRAM)
P269: (FATHER ARTHUR BERTRAM)
P270: (FATHER BERTRAM CORNISH)
DONE

(PRFACTS 'FATHER)
P268: (FATHER ARTHUR BEATRICE)
P269: (FATHER ARTHUR BERTRAM)
P270: (FATHER BERTRAM CORNISH)
DONE
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                               Inference
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(STASH '(IF (FATHER $X $Y) (PARENT $X $Y)))
P272

(STASH '(IF (MOTHER $X $Y) (PARENT $X $Y)))
P273

(STASH '(IF (AND (PARENT $Y $Z) (FATHER $X $Y)) (GFATHER $X $Z)))
P274

(LOOKUP '(GFATHER ARTHUR CORNISH))
NIL

(TRUEP '(GFATHER ARTHUR CORNISH))
(($1 . BERTRAM) (T . T))

(TRUEP '(GFATHER $X CORNISH))
(($X . ARTHUR) ($2 . BERTRAM) (T . T))

(TRUEPS '(GFATHER ARTHUR $Y))
((($Y . CAREY) ($3 . BEATRICE) (T . T))
 (($Y . CORNISH) ($3 . BERTRAM) (T . T)))

(TRUEP '(BAGOF $Y (GFATHER ARTHUR $Y) $S))
(($S CORNISH CAREY) (T . T))

(STASH '(IF (AND (BAGOF $Y (GFATHER $X $Y) $S) (LEN $S $N))
            (NUMGCHILDREN $X $N)))
P275

(STASH '(LEN NIL 0))
P276

(STASH '(IF (AND (LEN $L $M) (+ $M 1 $N))
            (LEN ($X . $L) $N)))
P277

(TRUEP '(NUMGCHILDREN ARTHUR $N))
(($N . 2) ($6 CORNISH CAREY) (T . T))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                             Justifications
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(SETQ JUSTIFY T)
T

(TRUEP '(GFATHER ARTHUR CORNISH))
(($12 . BERTRAM) (T . T))

(WHY '(GFATHER ARTHUR CORNISH))
P282: (GFATHER ARTHUR CORNISH) by BC
    P269: (FATHER ARTHUR BERTRAM)
    P279: (PARENT BERTRAM CORNISH)
    P274: (IF (AND (PARENT $Y $Z) (FATHER $X $Y)) (GFATHER $X $Z))
DONE

(WHY '(PARENT BERTRAM CORNISH))
P279: (PARENT BERTRAM CORNISH) by BC
    P270: (FATHER BERTRAM CORNISH)
    P272: (IF (FATHER $X $Y) (PARENT $X $Y))
DONE

(WHY '(FATHER ARTHUR BERTRAM))
P269: (FATHER ARTHUR BERTRAM) by BC
DONE

(WHERE '(FATHER ARTHUR BERTRAM))
P282: (GFATHER ARTHUR CORNISH) by BC
    P269: (FATHER ARTHUR BERTRAM)
    P279: (PARENT BERTRAM CORNISH)
    P274: (IF (AND (PARENT $Y $Z) (FATHER $X $Y)) (GFATHER $X $Z))
DONE

(SETQ JUSTIFY NIL)
NIL

(EMPTY 'JUSTIFY)
DONE
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                              Task Tracing
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(TRACETASK '&X)
DONE

(TRUEP '(PARENT ARTHUR BERTRAM))
      Executing BCDISP on
        ((PARENT ARTHUR BERTRAM))
        ((T . T))
        NIL
        NIL

      Executing BCDISP on
        ((MOTHER ARTHUR BERTRAM))
        ((T . T))
        (P273)
        (((PARENT ARTHUR BERTRAM)) NIL)

      Executing BCDISP on
        ((FATHER ARTHUR BERTRAM))
        ((T . T))
        (P272)
        (((PARENT ARTHUR BERTRAM)) NIL)

      Executing BCDISP on
        NIL
        ((T . T))
        (P272)
        (((PARENT ARTHUR BERTRAM)) NIL)

      Executing BCDISP on
        NIL
        ((T . T))
        NIL
        NIL

      Executing SUCCEED on
        ((T . T))
((T . T))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                    Caching
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(SETQ CACHE 'CACHE)
CACHE

(TRUEP '(PARENT ARTHUR BERTRAM))
      Executing BCDISP on
        ((PARENT ARTHUR BERTRAM))
        ((T . T))
        NIL
        NIL

      Executing BCDISP on
        ((MOTHER ARTHUR BERTRAM))
        ((T . T))
        (P273)
        (((PARENT ARTHUR BERTRAM)) NIL)

      Executing BCDISP on
        ((FATHER ARTHUR BERTRAM))
        ((T . T))
        (P272)
        (((PARENT ARTHUR BERTRAM)) NIL)

      Executing BCDISP on
        NIL
        ((T . T))
        NIL
        (((PARENT ARTHUR BERTRAM)) NIL)

      Executing BCDISP on
        NIL
        ((T . T))
        NIL
        NIL

      Executing SUCCEED on
        ((T . T))
((T . T))

(TRUEP '(PARENT ARTHUR BERTRAM))
      Executing BCDISP on
        ((PARENT ARTHUR BERTRAM))
        ((T . T))
        NIL
        NIL

      Executing BCDISP on
        NIL
        ((T . T))
        NIL
        NIL

      Executing SUCCEED on
        ((T . T))
((T . T))
```

```
(SETQ CACHE NIL)
NIL

(EMPTY 'CACHE)
DONE

(UNTRACETASK)
(&X)


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                 Demons
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(STASH '(IF (AND (BROTHER $X $Y) (FATHER $Y $Z))
       (RUNNABLE (ASSERT (UNCLE $X $Z)))))
P284

(STASH '(TOASSERT (BROTHER &A &B) FC))
P285

(ASSERT '(BROTHER ANHEUSER ARTHUR))
DONE

(LOOKUP '(UNCLE $X $Y))
(($Y . BERTRAM) ($X . ANHEUSER) (T . T))

(STASH '(TOASSERT (UNCLE &A &B) FC))
P289

(STASH '(IF (UNCLE $X $Y)
            (RUNNABLE (PRINT ($X  is the uncle of  $Y)))))
P290

(ASSERT '(BROTHER ARMAND ARTHUR))
(ARMAND | is the uncle of | BEATRICE)
(ARMAND | is the uncle of | BERTRAM)
DONE

(STASH '(TOASSERT (NOT &A) FC))
P294

(STASH '(IF (NOT $P)
            (RUNNABLE (UNSTASH $P))))
P295

(ASSERT '(NOT (FATHER ARTHUR BERTRAM)))
DONE

(LOOKUP '(FATHER ARTHUR BERTRAM))
NIL
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                              Task Ordering
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(STASH '(FATHER ANDOVER BIDDIE))
P297

(STASH '(MOTHER ALLISON BIDDIE))
P298

(TRUEP '(PARENT $X BIDDIE))
(($X . ALLISON) (T . T))

(SETQ PREFERRED T)
T

(STASH '(PREFERRED (BCDISP ((FATHER &X &Y)) . &L)
                   (BCDISP ((MOTHER &U &V)) . &M)))
P299

(TRUEP '(PARENT $X BIDDIE))
(($X . ANDOVER) (T . T))

(STASH '(IF (AND (LENGTH $K $M) (LENGTH $L $N) (< $M $N))
            (PREFERRED (BCDISP $K . $KL) (BCDISP $L . $LL))))
P300
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        Change of Inference Procedure
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(STASH '(TOASSERT &P CNF-ASSERT))
P304

(STASH '(TOTRUEP &P RESOLUTION))
P305

(ASSERT '(IF (PARENT $X $Y) (ANC $X $Y)))
DONE

(ASSERT '(IF (ANC $X $Y) (REL $X $Y)))
DONE

(TRUEP '(IF (PARENT $X $Y) (REL $X $Y)))
(($ (REL $X $Y)) ($ (ANC $X $Y)) (T . T))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                              Change of Representation
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(ASSERT '(REPN (HEIGHT &X &Y) PL))
P301

(STASH '(HEIGHT ARTHUR 71))
71

(GET 'ARTHUR 'HEIGHT)
71

(LOOKUP '(HEIGHT ARTHUR $N))
(($N . 71) (T . T))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                              Procedural Attachment
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(STASH '(TOSTASH (AGE &X &N) STASH-AGE))
P312

(DEFUN STASH-AGE (P)
  (STORE (AGE (CADDR P)) (CONS (CADR P) (AGE (CADDR P)))))
STASH-AGE

(ARRAY AGE T 100)
AGE

(STASH '(AGE KATIE 23))
(KATIE)

(AGE 23)
(KATIE)
```