```
   SSSSSSSS  VV       VV  WW       WW  MM       MM    CCCCCCCC  HH       HH              MM       MM    AAAAAA      SSSSSSSS
   SSSSSSSS  VV       VV  WW       WW  MM       MM    CCCCCCCC  HH       HH              MM       MM    AAAAAA      SSSSSSSS
SS           VV       VV  WW       WW  MMMM   MMMM  CC          HH       HH              MMMM   MMMM   AA        AA  SS
SS           VV       VV  WW       WW  MMMM   MMMM  CC          HH       HH              MMMM   MMMM   AA        AA  SS
SS           VV       VV  WW       WW  MM MM MM MM  CC          HH       HH              MM  MM  MM   AA        AA  SS
SS           VV       VV  WW       WW  MM  MMM MM  CC          HH       HH              MM  MM  MM   AA        AA  SS
  SSSSSS     VV       VV  WW       WW  MM      MM  CC          HHHHHHHHHHH              MM      MM   AA        AA    SSSSSS
  SSSSSS     VV       VV  WW       WW  MM      MM  CC          HHHHHHHHHHH              MM      MM   AA        AA    SSSSSS
        SS   VV       VV  WW  WW   WW  MM      MM  CC          HH       HH              MM      MM   AAAAAAAAAA            SS
        SS   VV       VV  WW  WW   WW  MM      MM  CC          HH       HH              MM      MM   AAAAAAAAAA            SS
        SS     VV   VV     WWWW    WWWW  MM      MM  CC          HH       HH      ....    MM      MM   AA        AA          SS
        SS     VV   VV     WWWW    WWWW  MM      MM  CC          HH       HH      ....    MM      MM   AA        AA          SS
SSSSSSSS         VV        WW       WW  MM      MM    CCCCCCCC  HH       HH      ....    MM      MM   AA        AA  SSSSSSSS
SSSSSSSS         VV        WW       WW  MM      MM    CCCCCCCC  HH       HH      ....    MM      MM   AA        AA  SSSSSSSS
```

```
44    44      000000      000000                  44    44      000000    5555555555
44    44      000000      000000                  44    44      000000    5555555555
44    44    00      00  00      00                44    44    00      00  55
44    44    00      00  00      00                44    44    00      00  55
44    44    00    0000  00    0000                44    44    00    0000  555555
44    44    00    0000  00    0000                44    44    00    0000  555555
4444444444  00  00  00  00  00  00                4444444444  00  00  00        55
4444444444  00  00  00  00  00  00                4444444444  00  00  00        55
      44    0000    00  0000    00                      44    0000    00          55
      44    0000    00  0000    00                      44    0000    00          55
      44    00      00  00      00      ,,,,            44    00      00  55      55
      44    00      00  00      00      ,,,,            44    00      00  55      55
      44      000000      000000          ,,                44      000000    555555
      44      000000      000000          ,,                44      000000    555555
```

```
BBBB  U   U N   N DDDD  Y   Y        AAA
B   B U   U NN  N D   D  Y Y        A   A
B   B U   U N N N D   D   Y         A   A
BBBB  U   U N  NN D   D   Y         AAAAA
B   B U   U N   N D   D   Y         A   A
BBBB  UUUUU N   N DDDD    Y         A   A
```

```
LPTSPL VERSION 6(344)  RUNNING ON LPT500
 *START* USER BUNDY A [400,405] JOB  SVWMCH SEQ. 672 DATE 20-APR-77 10:49:21 MONITOR 6.02A/RP04/DN87 *START*
REQUEST CREATED: 20-APR-77  10:50:08
FILE: DSKA2:SVWMCH.MAS[400,405] CREATED: 20-APR-77 10:47:00 <155>  PRINTED: 20-APR-77 10:49:28
QUEUE SWITCHES:  /PRINT:ARROW /FILE:ASCII /COPIES:1 /SPACING:1 /LIMIT:74 /FORMS:NORMAL
```

%%%%%SVWMCH . CMD @10:47 20-APR-1977 <055> (20)

```
SVWMCH.CMD
USVW.OLD
XTRACT.OLD
MOTION.OLD
SCHEMA.OLD
FUNCC.OLD
CNVERT.OLD
II
OLDXRT
```

\\\\\

Old SVW Mechs poogram

```
%%%%%USVW    .  OLD    @14:17 19-APR-1977 <055> (964)


/*USVW*/
/*UTILITIES FOR SVW PROLOG*/
/*BUNDY JAN 1976*/

/*OPERATOR DECLARATIONS*/

:- OP(200,FY,"-").        /*UNARY MINUS*/

:- OP(300,XFY,":").          /*EXPONENTIATION*/

:- OP(410,XFY,".").          /*LIST CONS*/

:- OP(490,XFY,"%").          /*BINARY MINUS*/

:- OP(700,XFX,">").     /*GREATER THAN*/

:- OP(700,XFX,">=").       /*GREATER THAN OR EQUAL TO*/

:- OP(850,XFY,"&").          /*CONJUNCTION*/

:- OP(950,XFY,"#").          /*DISJUNCTION*/

/* *, +, /, = ARE ALREADY DECLARED*/


/*TRACING*/

PR(←MES) :- WRITE(←MES),NEWLINE.

PPR(TRUE).

PPR(←E&←EC) :- TRACE(←E,2), PPR(←EC).

TRACE(←MES,←N) :- TFLAG(←M), ←M<←N, !.

TRACE(←MES,←N) :- WRITE(←MES),NEWLINE.

TFLAG(3) .

TLIM(←N) :-  ASSERTD(TFLAG(←N)).

/*CONVENTIONS  0. NO PRINTING
               1. BARE MINIMUM
               2. NORMAL PROTOCOL (DEFAULT)
               3. GENERAL DEGUGGING
             > 3. SPECIAL DEGUGGING*/
```

```
/*SETS*/

MEMBER(←X,←X,←YS) :- !.

MEMBER(←X,←Y,←YS) :- MEMBER(←X,←YS).

SUBSET(NIL,←YS) .

SUBSET(←X.←XS,←YS) :- MEMBER(←X,←YS), SUBSET(←XS,←YS).

UNION(NIL,←YS,←YS) .

UNION(←X,←XS,←YS,←ZS) :- MEMBER(←X,←YS), !, UNION(←XS,←YS,←ZS).

UNION(←X,←XS,←YS,←X.←ZS) :- UNION(←XS,←YS,←ZS).

SUBTRACT(NIL,←YS,NIL) .

SUBTRACT(←X.←XS,←YS,←ZS) :- MEMBER(←X,←YS), !, SUBTRACT(←XS,←YS,←ZS).

SUBTRACT(←X.←XS,←YS,←X.←ZS) :- SUBTRACT(←XS,←YS,←ZS).

SETEQ(←SET1,←SET2) :- SUBSET(←SET1), SUBSET(←SET2).


/*WORDS AND NUMBERS*/

INTEGER(←N) :- NATNUM(←N), !.

INTEGER(←N)  :- ←N =.. ",".←M, NATNUM(←M).

NATNUM(←N) :- ←N =.. ←M.NIL, CHECKLIST(DIGIT,←M).

BOUND(←X) :- THNOT(VAR(←X)).
NONVAR(←X) :- THNOT(VAR(←X)).

WORD(←X) :- ←X =.. (←H.←T).NIL, LETTER(←H).

CONCAT(←W1,←W2,←W3) :- ←W1 =.. ←LS1.NIL, ←W2 =.. ←LS2.NIL,
    APPEND(←LS1,←LS2,←LS3), ←W3 =.. ←LS3.NIL.



/*MISC*/

/*IF THEN ELSE*/
COND(←P,←Q,←R) :- ←P, !, ←Q.
COND(←P,←Q,←R) :- ←R.

/*PREDICATE APPLICATION*/
APPLY(←P,←ARGS1) :-
  ←P  =.. ←LS.←ARGS2, APPEND(←ARGS1,←ARGS2,←ARGS),
  ←Q =.. ←LS.←ARGS, !, ←Q.
/*X AND Y ARE DIFFERENT SYMBOLS*/
DIFF(←X,←X) :- !, FAIL.

DIFF(←X,←Y) .
```

```
/*GENERATES NEW SYMBOL Q WITH PREFIX PREF*/
GENSYM(←PREF,←Q) :- VAR(←Q), CURRENT(←PREF,←N),
    DENY(COUNT(←PREF,←N)),+(←N,1,←N1),ASSERT(COUNT(←PREF,←N1)),
    CONCAT(←PREF,←N1,←Q), !.

CURRENT(←PREF,←N) :- COUNT(←PREF,←N), !.
CURRENT(←PREF,0).


/*CONDITIONAL GENSYM*/
CGENSYM(←PREF,←X) :- GENSYM(←PREF,←X), !.
CGENSYM(←PREF,←X).

/*FIND ALL XS OBEYING P*/
FOUND(NIL).

FINDALL(←P,←DUM) :-
  APPLY(←P,←X,NIL), FOUND(←XL),
  DENY(FOUND(←XL)),ASSERT(FOUND(←X.←XL)),FAIL.

FINDALL(←P,←XL) :- FOUND(←XL),
  DENY(FOUND(←XL)), ASSERT(FOUND(NIL)), !.


/*FUNNY CALLS*/

/*JUST CALL*/
CALL(←L) :- ←L.

/*GARBAGE COLLECT CALL*/
GCC(←L) :- ONCEANDFAIL(←L).
GCC(←L) :- RETRIEVE(←L).

ONCEANDFAIL(←L) :- ←L, ASSERT(RETRIEVE(←L)), !, FAIL.


/*LIT ASSUMED FALSE SINCE NOT PROVABLE*/
THNOT(←LIT) :- ←LIT, !, FAIL.
THNOT(←LIT).


NLC(←L) :- SUBGOAL←OF(←L), !, FAIL.   /*NON LOOP CALL*/

NLC(←L) :- ←L.


CC(←L) :- FPC(←L), !.   /*CREATIVE CALL*/
CC(←L) :- CCFLAG(ON), DECLARE(←L), !.

CCFLAG(ON).


/*L CALLED REPEATEDLY*/
AGAIN(←L) :- ←L, PR(←L), FAIL.

AGAIN(←L) :- PR(THATS-ALL-FOLKS).


/*FUNNY ASSERTS*/
```

```
/*ASSERTION REMOVABLE ON BACKUP*/
POSTULATE(←L) :- ASSERT(←L).
POSTULATE(←L) :- DENY(←L),FAIL.

PASSERT(←ASS) :- ASSERT(←ASS), TRACE(←ASS,6).   /*PRINT ASSERT*/

/*PRINT ASSERT AT BOTTOM*/
PASSERTC(←ASS) :- ASSERTC(←ASS), TRACE(←ASS,6).

CASSERTC(←L) :- ←L,!.   /*CONDITIONAL ASSERT*/

CASSERTC(←L) :- ASSERTC(←L),!.


TRUE .



\\\\\
```

```
%%%%%XTRACT  .  OLD     @16:30 19-APR-1977 <055> (2477)



/* XTRACT*/
/*MECHANICS EQUATION EXTRACTION ROUTINES*/
/*GATHERED BY ALAN BUNDY 8/9/76*/



/*ABSENT FRIENDS*/

CONVERT(←X,←X).
SIMPLIFY(←X,←X).
SIMSOLVE(←E,←X,←E).

/*MARPLES ALGORITHM*/

/*BASIS CASE*/
GETEQNS(NIL,←GS,←US,TRUE,NIL).

/*GET EQUATIONS WITH NO INTERMEDIATES*/
GETEQNS(←X,←XS,←GS,←US,←E&←ES,←X,←XS1)
    :- MAKESL(←X,←SL), CHOOSEEQN(←SL,←E,←U,←US,OFF),
    TRACE(EQUATION-←U-FORMED,6),
    WORDSIN(←E,←VS), MEMBER(←X,←VS),
    UNION(←X,←XS,←GS,←YS), SUBSET(←VS,←YS),
    TRACE(←E-SOLVES-FOR-←X,5),
    GETEQNS(←XS,←X,←GS,←U,←US,←ES,←XS1).

/*GET EQUATIONS WITH INTERMEDIATES*/
GETEQNS(←X,←XS,←GS,←US,←E&←ES,←X,←XS1)
    :- MAKESL(←X,←SL), CHOOSEEQN(←SL,←E,←U,←US,ON),
    TRACE(EQUATION-←U-FORMED,6),
    WORDSIN(←E,←VS), MEMBER(←X,←VS),
    UNION(←X,←XS,←GS,←YS), SUBTRACT(←VS,←YS,←ZS), DIFF(←ZS,NIL),
    APPEND(←XS,←ZS,←WS),
    TRACE(←E-SOLVES-FOR-←X-BUT-INTRODUCES-←ZS,5),
    GETEQNS(←WS,←X,←GS,←U,←US,←ES,←XS1).

/*ASSUME X IS ELIMINABLE*/
GETEQNS(←X,←XS,←GS,←US,←ES,←X,←XS1)
    :- TRACE(I-ASSUME-←X-IS-ELIMINABLE,2),
    GETEQNS(←XS,←X,←GS,←US,←ES,←XS1).


PASS(←N,←S,←US) :- OLDEQN(←E,←N,←S), !, FAIL.

PASS(←N,←S,←US) :- UNUSED(←N,←S,←US), !.


UNUSED(←NAME,←SIT,←US) :- MEMBER(←NAME-←SIT,←US), !, FAIL.

UNUSED(CONSTACCEL-←N1,←SIT,←US) :- MEMBER((CONSTACCEL-←N2)-←SIT,←US),
    MEMBER((CONSTACCEL-←N3)-←SIT,←US), DIFF(←N2,←N3), !, FAIL.
```

```
UNUSED(RESOLVE,(←PART,←PER,←DIR1),←US) :-
    MEMBER(RESOLVE-(←PART-←PER-←DIR2),←US),
    MEMBER(RESOLVE-(←PART-←PER-←DIR3),←US),
    DIFF(←DIR2,←DIR3), !, FAIL.

UNUSED(RELVEL,←PSET1-←TIME,←US) :-
    MEMBER(RELVEL-(←PSET2-←TIME),←US), SETEQ(←PSET1,←PSET2), !, FAIL.

UNUSED(RELACCEL,←PSET1-←TIME,←US) :-
    MEMBER(RELACCEL-(←PSET2-←TIME),←US),
    SETEQ(←PSET1,←PSET2), !, FAIL.

UNUSED(←NAME,←SIT,←US).

CHOOSEEQN(←C1,←REST,←E,←C1-←S,←US,←SW) :-
    MAKEEQN(←E,←C1-←S,←US,←SW).

CHOOSEEQN(←C1,←REST,←E,←U,←US,←SW) :-
    CHOOSEEQN(←REST,←E,←U,←US,←SW).

MAKESL(←X,←SL) :- EQNSLIST(←EQNS), TYPE(←X,←XTYPE),
    SUBLIST(TEST(←XTYPE),←EQNS,←SL),
    TRACE(SHORTLIST-IS-←SL,7).

TEST(←EQN,←XTYPE) :- RELATES(←EQN,←VTYPES),MEMBER(←XTYPE,←VTYPES).

MAKEEQN(←E,←N-←S,←US,←SW) :- OLDEQN(←E,←N,←S), UNUSED(←N,←S,←US).

MAKEEQN(←E,←N-←S,←US,←SW) :-
    ASSERTD(CCFLAG(←SW)),
    ISEQN(←E,←N-←S,←US), CASSERTC(OLDEQN(←E,←N,←S)).




/*THE EQUATIONS*/



EQNSLIST([RESOLVE,STRACCEL,RELVEL,RELACCEL,
TIMESUM,DISTSUM,(CONSTACCEL-1),(CONSTACCEL-2),CONSTVEL,
(CONSVENERGY-1),(CONSVENERGY-2),AVERVEL]).

RELATES(RESOLVE,[FORCE,ACCEL,MASS]) :- !.
RELATES(STRACCEL,[ACCEL]) :- !.
RELATES(RELVEL,[VEL]) :- !.
RELATES(RELACCEL,[ACCEL]) :- !.
RELATES(TIMESUM,[DURATION]) :- !.
RELATES(DISTSUM,[DISTANCE]) :- !.
RELATES(CONSTACCEL-1,[ACCEL,VEL,DURATION]) :- !.
RELATES(CONSTACCEL-2,[ACCEL,DISTANCE,VEL,DURATION]) :- !.
RELATES(CONSTVEL,[VEL,DISTANCE,DURATION]) :- !.
```

```
RELATES(CONSVENERGY-1,[VEL,DISTANCE]) :- !.
RELATES(CONSVENERGY-2,[VEL,DISTANCE]) :- !.
RELATES(AVERVEL,[VEL,DISTANCE,DURATION]) :- !.

ISEQN(←FORCESUM=←M*←A*←COSTERM,RESOLVE-(←PART-←PER-←DIR1),←US)
    :- PERIOD(←PER), PARTICLE(←PART),
    FIXDIR(←DIR1),
    PASS(RESOLVE,(←PART-←PER-←DIR1),←US),
    CC(MASS(←PART,←M,←PER)),
    CC(ACCEL(←PART,←A,←DIR2,←PER)),
    COSFACT(←DIR1,←DIR2,←COSTERM),
    SUMFORCES(←PART,←DIR1,←PER,←FORCESUM).


ISEQN(2*←A0=←A1-←A2,STRACCEL-(←SYS-←TIME),←US) :-
    PULLSYS(←SYS,←PULL,←STR,←P1,←P2,←TIME),
    PASS(STRACCEL,(←SYS-←TIME),←US),
    CC(ACCEL(←PULL,←A0,270,←TIME)),
    CC(ACCEL(←P1,←A1,270,←TIME)),
    CC(ACCEL(←P2,←A2,90,←TIME)).


ISEQN(←V13=←V123,RELVEL-([←P1,←P2,←P3]-←TIME),←US) :-
    PARTICLE(←P1),PARTICLE(←P2),DIFF(←P1,←P2),
    PARTICLE(←P3),DIFF(←P1,←P3),DIFF(←P2,←P3),PERIOD(←TIME),
    PASS(RELVEL,[←P1,←P2,←P3]-←TIME,←US),
    CC(RELVEL(←P1,←P2,←V12,←DIR12,←TIME)),
    CC(RELVEL(←P2,←P3,←V23,←DIR23,←TIME)),
    CC(RELVEL(←P1,←P3,←V13,←DIR13,←TIME)),
    VECADD(←V12,←DIR12,←V23,←DIR23,←V123,←DIR13).


ISEQN(←A13=←A123,RELACCEL-([←P1,←P2,←P3]-←TIME),←US) :-
    PARTICLE(←P1),PARTICLE(←P2),DIFF(←P1,←P2),
    PARTICLE(←P3),DIFF(←P1,←P3),DIFF(←P2,←P3),PERIOD(←TIME),
    PASS(RELACCEL,[←P1,←P2,←P3]-←TIME,←US),
    CC(RELACCEL(←P1,←P2,←A12,←DIR12,←TIME)),
    CC(RELACCEL(←P2,←P3,←A23,←DIR23,←TIME)),
    CC(RELACCEL(←P1,←P3,←A13,←DIR13,←TIME)),
    VECADD(←A12,←DIR12,←A23,←DIR23,←A123,←DIR13).



ISEQN(←T=←SUM,TIMESUM-←P,←US) :-
    PARTITION(←P,←PS), PASS(TIMESUM,←P,←US),
    CC(DURATION(←P,←T)), SUMDURS(←PS,←SUM).

ISEQN(←D=←SUM,DISTSUM-(←P-←OBJ),←US) :- PARTITION(←P,←PL),
    CC(DISTANCE(←OBJ,←D,←P)), PASS(DISTSUM,←P-←OBJ,←US) ,
    SUMDIST(←OBJ,←PL,←SUM).

ISEQN(←V=←U+(←A*←T),(CONSTACCEL-1)-(←P-←OBJ),←US)
    :- PARTICLE(←OBJ), PERIOD(←P),
    ACCEL(←OBJ,←A,←DIR,←P), PASS(CONSTACCEL-1,(←P-←OBJ),←US),
    ISQINVAR(←A), DIFF(←A,ZERO),
    CC(DURATION(←P,←T)), INITVEL(←OBJ,←U,←DIR,←P),
    FINVEL(←OBJ,←V,←DIR,←P).


ISEQN(←D=(←U*←T)+((1/2)*←A*(←T:2)),(CONSTACCEL-2)-(←P-←OBJ),←US)
```

```
    :- PARTICLE(←OBJ), PERIOD(←P),
    ACCEL(←OBJ,←A,←DIR,←P), PASS(CONSTACCEL-2,(←P-←OBJ),←US),
    ISQINVAR(←A),
    DIFF(←A,ZERO), CC(DURATION(←P,←T)),
    INITVEL(←OBJ,←U,←DIR,←P), CC(DISTANCE(←OBJ,←D,←P)).

ISEQN(←V=←U+←A*←T=←S,CONSTVEL-(←P-←OBJ),←US)
    :- PARTICLE(←OBJ), PERIOD(←P),
    VEL(←OBJ,←V,←DIR,←P), PASS(CONSTVEL,(←P-←OBJ),←US),
    ISQINVAR(←V), DIFF(←V,ZERO),
    CC(DURATION(←P,←T)), CC(DISTANCE(←OBJ,←S,←P)).

ISEQN((1/2)*←V↑2-(1/2)*←U↑2=G*←H,(CONSVENERGY-1)-(←P-←O),←US)
    :- PARTICLE(←O), PERIOD(←P),
    PASS(CONSVENERGY-1,←P-←O,←US),
    DBC(MOTION(←O,←PATH,←START,←SIDE,←P)),
    FREE(←PATH,←P,←O),
    DROP(←PATH,←START,←H),
    FINVEL(←O,←V,←DIR1,←P), INITVEL(←O,←U,←DIR2,←P).

ISEQN((1/2)*←V↑2-(1/2)*←U↑2=G*←H,(CONSVENERGY-2)-(←P-←O),←US)
    :- DBC(MOTION(←O,←PATH,←START,←SIDE,←P)),
    PASS(CONSVENERGY-2,←P-←O,←US),
    FREE(←PATH,←P,←O),
    TYPICAL←DROP(←PATH,←START,←H),
    CC(VEL(←O,←V,←DIR1,←P)),
    INITVEL(←O,←U,←DIR2,←P).

ISEQN(←AV=←D/←T,AVERVEL-(←P-←OBJ),←US)
    :- PARTICLE(←OBJ), PERIOD(←P),
    AVERVEL(←OBJ,←AV,←DIR,←P), PASS(AVERVEL,←P-←OBJ,←US),
    CC(DISTANCE(←OBJ,←D,←P)), CC(DURATION(←P,←T)).




/*SYMBOLS*/

WORDSIN(←E,NIL) :- ISCONST(←E), !.

WORDSIN(←E,NIL) :- ISQCONST(←E), !.

WORDSIN(←E,←E,NIL) :- WORD(←E), !.

WORDSIN(←E,←VS) :-  ←E =..←S,←ARGS , SWORDSIN(←ARGS,←VS).

SWORDSIN(NIL,NIL).

SWORDSIN(←ARG,←ARGS,←VS)
    :- WORDSIN(←ARG,←V1), SWORDSIN(←ARGS,←V2), UNION(←V1,←V2,←VS).

ISQCONST(ZERO) :- !.

ISQCONST(←Q) :- MEASURE(←Q,←M), ISCONST(←M).

ISCONST(←V) :- ISNUM(←V), !.
```

```
ISCONST(←V) :- CONST(←V), !.

ISconST(←V) :-  ←V =..←S.NIL , !, FAIL.

ISCONST(←V) :-  ←V =..←S.←ARGS , !, LISCONST(←ARGS).

LISCONST(NIL).

LISCONST(←V.←VL) :- ISCONST(←V), !, LISCONST(←VL).

ISSYM(←E) :-  ←E =..←S.NIL .

COMP(←E) :-  ←E =..←S.NIL , !, FAIL.

COMP(←E).

ISNUM(←E) :- INTEGER(←E).




/*INFERENCE*/

GIVENS(←GL) :- FINDALL(GIVEN,←GL).

GIVEN(G).

SOUGHTS(←SL) :- FINDALL(SOUGHT,←SL).

SUMDURS(NIL,0).

SUMDURS(←P.←PS,←T+←SUM) :- CC(DURATION(←P,←T)), !, SUMDURS(←PS,←SUM).


SUMDIST(←OBJ,NIL,0).

SUMDIST(←OBJ,←P.←PL,←D+←SUM) :- CC(DISTANCE(←OBJ,←D,←P)),
    !, SUMDIST(←OBJ,←PL,←SUM).


COSFACT(←DIR,←DIR,1) :- !.
COSFACT(←DIR1,←DIR2,COS(←ANS)) :-
  INTEGER(←DIR1), INTEGER(←DIR2),
  ←ANS IS ←DIR1-←DIR2, !.
COSFACT(←DIR1,←DIR2,COS(←DIR1-←DIR2)).

SUMFORCES(←PART,←DIR1,←PER,←FORCESUM) :-
    FORCE(←PART,←FORCE,←DIR2,←PER),
    COSFACT(←DIR1,←DIR2,←COSTERM),
    FORCESUM(←SOFAR), DENY(FORCESUM(←SOFAR)),
    ASSERT(FORCESUM((←FORCE*←COSTERM)+←SOFAR)), FAIL.

SUMFORCES(←PART,←DIR1,←PER,←FORCESUM) :-
    FORCESUM(←FORCESUM), DENY(FORCESUM(←FORCESUM)),
    ASSERT(FORCESUM(0)).
```

```
FORCESUM(0).

INITVEL(←OBJ,←U,←DIR,←P) :-
    CC(INITIAL(←P,←BEGIN)), CC(VEL(←OBJ,←U,←DIR,←BEGIN)).


FINVEL(←OBJ,←V,←DIR,←P) :-
    CC(FINAL(←P,←END)), CC(VEL(←OBJ,←V,←DIR,←END)).

VECADD(←MAG1,←DIR,←MAG2,←DIR,←MAG1%←MAG2,←DIR).

ISQINVAR(ZERO) :- !.

ISQINVAR(←Q) :- MEASURE(←Q,←M), ISINVAR(←M), !.

ISINVAR(←V) :- INVAR(←V), !.

ISINVAR(←V) :- ISNUM(←V), !.

ISINVAR(←V) :- ISSYM(←V), !, FAIL.

ISINVAR(←V) :-  ←V =..←S.←ARGS , !, LISINVAR(←ARGS).

LISINVAR(NIL) :- !.

LISINVAR(←V.←VL) :- ISINVAR(←V), LISINVAR(←VL), !.

FIXDIR(←DIR) :- NONVAR(←DIR), !.
FIXDIR(270).
FIXDIR(0).


ACCEL(←X,ZERO,←DIR,←TIME) :- FIXED←CONTACT(←X,EARTH,←TIME).

ACCEL(←PART,ZERO,←DIR,←PER) :-
    VEL(←PART,←V,←DIR,←PER), ISQINVAR(←V),PERIOD(←PER), !.

ACCEL(←PART,←A,←DIR,←TIME) :- NLC(RELACCEL(←PART,EARTH,←A,←DIR,←TIME)).

RELACCEL(←PART,EARTH,←A,←DIR,←TIME) :- NLC(ACCEL(←PART,←A,←DIR,←TIME)).

/*GRAVITATIONAL FORCE*/
FORCE(←PART,←M*G,270,←TIME) :-
    PARTICLE(←PART), CC(MASS(←PART,←M,←TIME)).

/*TENSION IN STRING*/
FORCE(←PART,←T,←DIR,←TIME) :-
    PARTICLE(←PART), FIXED←CONTACT(←PART,←STR,←TIME),
    STRING(←STR), CC(DIRECTION(←STR,←DIR,←TIME)),
    CC(TENSION(←STR,←T,←TIME)).



STRING(LEFT(←STR)) :- STRING(←STR).

STRING(RIGHT(←STR)) :- STRING(←STR).


INITIAL(←P,←M) :-
```

```
  PERIOD(←P), PARTITION(←E,←P,←REST), INITIAL(←E,←M), !.

INITIAL(←P,←M) :-
  PERIOD(←P), PARTITION(←E,←PL),
  NEXTTO(←P1,←P,←PL), NLC(FINAL(←P1,←M)), !.

FINAL(←P,←M) :-
  PERIOD(←P), PARTITION(←E,←PLIST), LAST(←P,←PLIST),
  FINAL(←E,←M), !.

FINAL(←P,←M) :-
  PERIOD(←P), PARTITION(←E,←PLIST),  NEXTTO(←P,←P1,←PLIST),
  NLC(INITIAL(←P1,←M)), !.

DROP(←PATH,←START,-(←D*TAN(←ANG))) :-
     CONCAVITY(←PATH,STLINE),
     INCLINE(←PATH,←ANG,←START), GROUND(←PATH,←D).

DROP(←PATH,←START,←HSUM) :-
     PARTITION(←PATH,←PL), SUMDROPS(←PL,←START,←HSUM).

SUMDROPS(NIL,←START,Ø).

SUMDROPS(←P.←PL,←START,←H+←SUM) :-
     DROP(←P,←START,←H), !, FARENDO(←P,←START,←FINISH),
     SUMDROPS(←PL,←FINISH,←SUM).


FREE(←PATH,←PER,←O) :- PARTICLE(←O), PERIOD(←PER),
    PATH(←PATH), PROBTYPE(ROLLER-COASTER).

VEL(←PART,←VEL,←DIR,←MOM) :-
  INITIAL(←PER,←MOM), VEL(←PART,←VEL,←DIR,←PER), ISQINVAR(←VEL), !.

VEL(←PART,←VEL,←DIR,←MOM) :-
  FINAL(←PER,←MOM), VEL(←PART,←VEL,←DIR,←PER), ISQINVAR(←VEL), !.

TIME(←T) :- PERIOD(←T), !.

TIME(←T) :- MOMENT(←T), !.

MEASURE(ZERO,Ø) :- !.

MEASURE(G,32) :- !.

CCMEASURE(←O,←M) :- ASSERTO(CCFLAG(ON)), CC(MEASURE(←Q,←M)).


UNIT(ZERO,←U) :- !.

UNIT(G,FT/SECS:2) :- !.

UNIT(←Q,←U) :-
  STANDSYS(←SYS), QUANTITY(←Q,←D,←T), STANDUNIT(←D,←U,←SYS).


QUANTITY(←Q,1,S) :- ISA(←Q,COFRI), !.

QUANTITY(←Q,1,S) :- ISA(←Q,ANG), !.
```

```
QUANTITY(←Q,T,S) :- ISA(←Q,DURATION), !.
QUANTITY(←Q,L,S) :- ISA(←Q,DISTANCE), !.
QUANTITY(←Q,M,S) :- ISA(←Q,MASS), !.
QUANTITY(←Q,M*(L/(T:2)),V) :- ISA(←Q,FORCE), !.
QUANTITY(←Q,M*(L/(T:2)),S) :- ISA(←Q,TENSION), !.
QUANTITY(←Q,M*(L/(T:2)),V) :- ISA(←Q,REACTION), !.
QUANTITY(←Q,M*(L/T(T:2)),V) :- ISA(←Q,FRICTION), !.
QUANTITY(←Q,L/T,V) :- ISA(←Q,VEL), !.
QUANTITY(←Q,L/T:2,V) :- ISA(←Q,ACCEL), !.
QUANTITY(←Q,L/T:2,V) :- ISA(←Q,RELACCEL), !.
QUANTITY(←Q,L/T,V) :- ISA(←Q,RELVEL), !.
QUANTITY(←Q,1,S) :- ISA(←Q,DIRECTION), !.

TYPE(←Q,←QTYPE) :- QUANTITY(←Q,←DIM,←VS), NAME(←QTYPE,←DIM).

NAME(MASS,M).
NAME(DISTANCE,L).
NAME(DURATION,T).
NAME(VEL,L/T).
NAME(ACCEL,L/(T:2)).
NAME(FORCE,M*(L/(T:2))).

ISVECT(←Q) :- QUANTITY(←Q,←D,V).
ISSCAL(←Q) :- QUANTITY(←Q,←D,S).



/*DECLARATIONS*/

DECLARE(←L) :- ←L =., ←LS,←ARGS,
    DECLARECHECKS(←ARGS), !, ←PROP =,, ←LS,NIL,
    MESS(←PROP,←ARGS), ASSERT(←L).


DECLARECHECKS(←ARGS):-
  SEPERATE(←ARGS,←PARGS,←FARGS,←NUM),
  CHECKLIST(BOUND,←PARGS).
```

```
MESS(←PROP,[←Q]) :-
    MAKE(←PROP,←Q),
    TRACE(LET-←Q-BE-A-←PROP,2).

MESS(←PROP,[←OBJ,←Q]) :-
    MAKE(←PROP,←Q),
    TRACE(LET-←Q-BE-THE-←PROP-OF-←OBJ,2).

MESS(←PROP,[←OBJ,←Q,←TIME]) :-
    MAKE(←PROP,←Q),
    TRACE(LET-←Q-BE-THE-←PROP-OF-←OBJ-IN-←TIME,2).

MESS(←PROP,[←OBJ,←Q,←DIR,←TIME]) :-
    MAKE(←PROP,←Q), MAKE(DIRECTION,←DIR),
    TRACE(LET-←Q-BE-THE-←PROP-OF-←OBJ,2),
    TRACE(IN-DIRECTION-←DIR-AND-TIME-←TIME,2).

MESS(←PROP,[←OBJ1,←OBJ2,←Q,←DIR,←TIME]) :-
    MAKE(←PROP,←Q), MAKE(DIRECTION,←DIR),
    TRACE(LET-←Q-BE-THE-←PROP-OF-←OBJ1-RELATIVE-TO-←OBJ2,2),
    TRACE(IN-DIRECTION-←DIR-AND-TIME-←TIME,2).


MAKE(←PROP,←Q) :- GENSYM(←PROP,←Q), !, ASSERT(ISA(←Q,←PROP)).

MAKE(←PROP,←Q).



\\\\\
```

```
%%%%MOTION  .  OLD    @16:55 19-APR-1977 <055> (1754)



/*MOTION INFERENCE RULES*/
/*ALAN BUNDY 29/12/76*/


/*GENERAL QUESTION ANSWERING ROUTINE*/
QA(←QUAL,←QUAN) :-
  ←QUAL, FINDALL(PROVISO,←CONDLIST),
  TRACE(←QUAL-OK-PROVIDED-←CONDLIST,2),
  SOLVEALL(←CONDLIST,←SOLNS),
  SUBST(←SOLNS,←CONDLIST,←NCONDLIST),
  TRACE(NEW-CONDITIONS-ARE-←NCONDLIST,2),
  APPLY(←QUAN,←NCONDLIST.NIL).


/*MAKING CONDITIONS*/
/*SEE IF ITS TRUE*/
CONDITION(←L) :- ←L,!.

/*SEE IF ITS FALSE*/
CONDITION(←L) :- NOT(←L), !, FAIL.

/*OTHERWISE NOTE IT FOR LATER*/
CONDITION(←L) :- POSTULATE(PROVISO(←L)),
  TRACE(STORING-PROVISO-←L,6).


/*NEGATION*/
NOT(X>Y) :- Y>=X.
NOT(X>=Y) :- Y>X.


/*MOTION CHECK*/
MOTION(←PART,←PATH,←START,←SIDE,←PER) :-
  INPLACE(←PART,←PATH,←START,←SIDE,←BEGIN),
  CUE(TIMESYS(←PER,←BEGIN,←END)),
  TRACE(CHECKING-MOTION-OF-←PART-ON-←PATH-FROM-←START,7),
  TRACE(ON-←SIDE-DURING-←PER,7),
  MOTION1(←PART,←PATH,←START,←SIDE,←PER),
  CUE(MOTSYS(←PART,←PATH,←START,←SIDE,←PER)).


/*GENERAL MOTION ON PATH*/   /*LETS THROUGH TOO MANY POSIBILITIES*/
MOTION1(←PART,←PATH,←START,←SIDE,←PER) :-
  INITIAL(←PER,←BEGIN),
  GETSTARTED(←PART,←PATH,←START,←SIDE,←BEGIN),
  NOSTOPPING(←PART,←PATH,←START,←SIDE,←PER),
  NOTAKEOFF(←PART,←PATH,←START,←SIDE,←PER),
  NOFALLOFF(←PART,←PATH,←START,←SIDE,←PER).


/*BREAK INTO SUBPATHS*/
```

```
MOTION1(←PART,←PATH,←START,←SIDE,←PER) :-
  ARRANGEPATH(←PATH,←START,←NPATHLIST),
  MAKEPERIODS(←NPATHLIST,←PER,←PERLIST),
  MULTIMOTION(←PART,←NPATHLIST,←START,←SIDE,←PERLIST),

ARRANGEPATH(←PATH,←START,←NPATHLIST) :-
  PARTITION(←PATH,←PATHLIST),
  END(←PATH,←START,←END),
  CONDREV(←END,←PATHLIST,←NPATHLIST),

MAKEPERIODS(←NPATHLIST,←PER,←PERLIST) :-
  MAPLIST(MAKEONE,←NPATHLIST,←PERLIST),
  ASSERT(PARTITION(←PER,←PERLIST)),
  TRACE(LET-←PER-BE-DIVIDED-INTO-←PERLIST,2),

/*DEAL WITH EACH SUBPATH*/
MULTIMOTION(←PART,NIL,←START,←SIDE,NIL),

MULTIMOTION(←PART,←PATH,←PATHL,←START,←SIDE,←PER,←PERL) :-
  MOTION(←PART,←PATH,←START,←SIDE,←PER),
  FAREND(←PATH,←START,←FINISH),
  MULTIMOTION(←PART,←PATHL,←FINISH,←SIDE,←PERL),

/*MAKE ONE PERIOD*/
MAKEONE(←PATH,←PER) :- DECLARE(PERIOD(←PER)),

GETSTARTED(←PART,←PATH,←START,←SIDE,←BEGIN) :-
  VEL(←PART,ZERO,←DIR,←BEGIN), !,
  COND(INCLINE(←PATH,ZERO,←START),
    NUDGE(←PART,ZERO,←BEGIN),TOP(←PATH,←START)),

/*HEADED IN RIGHT DIRECTION*/
GETSTARTED(←PART,←PATH,←START,←SIDE,←BEGIN) :-
  INCLINE(←PATH,←ANG,←START),
  VEL(←PART,←V,←ANG,←BEGIN),
  CONDITION(←V>ZERO),


/*DOWNHILL RUN*/
NOSTOPPING(←PART,←PATH,←START,←SIDE,←PER) :-
  END(←PATH,←START,←END), OPPOSITE(←END,←OEND),
  SLOPE(←PATH,←HEND), DIFF(←OEND,←HEND), !.

/*MAKES IT TO THE TOP*/
NOSTOPPING(←PART,←PATH,←START,←SIDE,←PER) :-
 FINVEL(←PART,←V,←DIR,←PER), CONDITION(REAL(←V)),

/*BELOW PATH*/
NOTAKEOFF(←PART,←PATH,←START,←SIDE,←PER) :-
  BELOW(←PATH,←START,←SIDE), !.

/*SLOPE DOES NOT DROP AWAY*/
NOTAKEOFF(←PART,←PATH,←START,←SIDE,←PER) :-
  CONCAVITY(←PATH,←CONC), DIFF(←CONC,RIGHT), !.

/*INSUFFICIENT VEL TO TAKE OFF*/
NOTAKEOFF(←PART,←PATH,←START,←SIDE,←PER) :-
  CONCAVITY(←PATH,RIGHT),
  NORMAL(←PATH,←DIR),
  CC(REACTION(←PATH,←PART,←N,←DIR,←PER)),
```

```
  CONDITION(←N>=ZERO).


/*SUPPORTED*/
NOFALLOFF(←PART,←PATH,←START,←SIDE,←PER) :-
  ABOVE(←PATH,←START,←SIDE), !.

/*VERTICAL FALL*/
NOFALLOFF(←PART,←PATH,←START,←SIDE,←PER) :-
  SLOPE(←PATH,←START), CONCAVITY(←PATH,STLINE),
  INCLINE(←PATH,270,←START), !.

/*STICKS ON*/
NOFALLOFF(←PART,←PATH,←START,←SIDE,←PER) :-
  CONCAVITY(←PATH,RIGHT),
  NORMAL(←PATH,←DIR1),
  CC(VEL(←PART,←V,←DIR2,←PER)),
  RADIUS←OF←CURVATURE(←PATH,←R),
  CONDITION((←V:2)*SIN(←DIR1)>=←R*G), !.

/*FREEFALL*/
NOFALLOFF(←PART,←PATH,←START,←SIDE,←PER) :-
  CONCAVITY(←PATH,←CONC), DIFF(←CONC,RIGHT),
  ASSERT(FALLS←OFF(←PART,←PATH,←START,←SIDE,←PER)),
  !, FAIL.

/*DESCRIBING PATHS*/

/*PATH WITH MONOTONIC SLOPE*/
MONOPATH(←PATH) :- SLOPE(←PATH,LEFT).
MONOPATH(←PATH) :- SLOPE(←PATH,RIGHT).
MONOPATH(←PATH) :- SLOPE(←PATH,HOR).

/*POINT1 AND POINT2 ARE OPPOSITE ENDS OF PATH*/
FAREND(←PATH,←POINT1,←POINT2) :-
  END(←PATH,←POINT1,←END1), OPPOSITE(←END1,←END2),
  END(←PATH,←POINT2,←END2).

/*UPPER SIDE OF PATH*/
ABOVE(←PATH,←START,←SIDE) :-
  MONOPATH(←PATH), END(←PATH,←START,←SIDE).

/*LOWER SIDE OF PATH*/
BELOW(←PATH,←START,←SIDE1) :-
  MONOPATH(←PATH),
  END(←PATH,←START,←SIDE2), OPPOSITE(←SIDE1,←SIDE2).

/*START IS THE TOP OF PATH*/
TOP(←PATH,←START) :- END(←PATH,←START,←END), SLOPE(←PATH,←END).

/*PARTICLE IS IN PLACE AT START OF PATH*/
INPLACE(←PART,←PATH1,←FINISH,←SIDE,←END) :-
  DBC(MOTION(←PART,←PATH2,←START,←SIDE,←PER)),
  FAREND(←PATH2,←START,←FINISH),
  FINAL(←PER,←END), !.

INPLACE(←PART,←PATH,←START,←SIDE2,←TIME) :-
  DBC(AT(←PART,←START,←TIME)),
  SIDE(←PART,←START,←SIDE1,←TIME),
  END(←PATH,←START,←END),
```

```
  CONDVAL(←END,←SIDE1,←SIDE2).


/*CONSECUTIVE PATHS HAVE THE SAME INCLINATION
(ADD SMOOTHNESS CONDITION) */
INCLINE(←PH2,←ANG,←PT) :-
  PARTITION(←PHØ,←PHL),
  NEXTTO(←PH1,←PH2,←PHL),
  NLC(INCLINE(←PH1,←ANG,←PT)).

/* YOU CAN'T BE IN TWO PLACES AT ONCE */
AT(←M,←P1,←T) :-
  CHECKLIST(BOUND,[←M,←P1,←T]), DBC(AT(←M,←P2,←T)),
  DIFF(←P1,←P2), !, FAIL.


/*YOU GET TO YOUR DESTINATION BY TRAVELING THERE*/
AT(←PART,←PLACE2,←MOM2) :-
  FAREND(←PATH,←PLACE2,←PLACE1),
  INPLACE(←PART,←PATH,←PLACE1,←SIDE,←MOM1),
  CUE(TIMESYS(←PER,←MOM1,←MOM2)),
  NLC(MOTION(←PART,←PATH,←PLACE1,←SIDE,←PER)).

/*CONDITIONAL REVERSE*/
CONDREV(LEFT,←LIST,←LIST).

CONDREV(RIGHT,←LIST,←RLIST) :- REV(←LIST,←RLIST).



/*CONDITIONAL VALUE*/
CONDVAL(LEFT,←SIDE,←SIDE).
CONDVAL(RIGHT,←SIDE1,←SIDE2) :- OPPOSITE(←SIDE1,←SIDE2).

/*PARITY CHANGER*/
OPPOSITE(LEFT,RIGHT).
OPPOSITE(RIGHT,LEFT).

/*CALL MARPLES TO UNPACK PROVISOS*/

SOLVEALL(←CONDLIST,←ANS) :-
  SETUP(←CONDLIST,←ES,←XS1), CRUNCH(←ES,←XS1,←ANS).

SETUP(←CONDLIST,←ES,←XS1) :- SWORDSIN(←CONDLIST,←VARS),
    GIVENS(←GS), SUBTRACT(←VARS,←GS,←XS),
    TRACE(ATTEMPTING-TO-SOLVE-FOR-←XS-IN-TERMS-OF-←GS,5),
    GETEQNS(←XS,←GS,NIL,←ES,←XS1),
    TRACE(EQUATIONS-EXTRACTED,2), PPR(←ES).

CRUNCH(←ES,←XS1,←ANS) :-
    CONVERT(←ES,←ES1), TRACE(CONVERTED-TO,2), PPR(←ES1),
    SIMPLIFY(←ES1,←ES2), TRACE(SIMPLIFIED-TO,2), PPR(←ES2),
    MAPLIST(CCMEASURE,←XS1,←XS2), TRACE(UNKNOWNS-ARE-←XS2,2),
    SIMSOLVE(←ES2,←XS2,←ANS), TRACE(ANSWER-IS,2), PPR(←ANS).

/*APOLOGIES FOR ABSENCE*/
CRUNCH(←ES,←XS1,←ES) :- TRACE(CRUNCH-NOT-LOADED,2).

SUBST(←SOLNS,←CL,←CL) :- TRACE(SUBST-NOT-LOADED,2).
```

```
EVAL(←CONDS) :- TRACE(EVAL-NOT-LOADED,2).

MIN(←Q,←MV,←CONDS) :- TRACE(MIN-NOT-LOADED,2).


/*DETAILED INFERENCES FOR PROVISOS*/

/*RADIUS OF CURVATURE OF CIRCLE SEGMENT*/
RADIUS←OF←CURVATURE(←PATH,←R) :-
  PARTOF(←PATH,←CIRCLE), CIRCLE(←CIRCLE),
  CC(RADIUS(←CIRCLE,←R)).

/*PART-WHOLE RELATIONSHIP*/
PARTOF(←WHOLE,←WHOLE).
PARTOF(←PART,←WHOLE) :- BITOF(←PART,←WHOLE).


BITOF(←PART,←WHOLE) :- PARTITION(←WHOLE,←PARTS),
  MEMBER(←PART,←PARTS).

/*VERTICAL DROP OF CIRCLE SEGMENT*/
DROP(←PATH,←START,←R*(SIN(←DIR1)-SIN(←DIR2))) :-
  PARTOF(←PATH,←CIRCLE), CIRCLE(←CIRCLE),
  FAREND(←PATH,←START,←FINISH),
  ANGLE(←START,←DIR1,←CIRCLE), ANGLE(←FINISH,←DIR2,←CIRCLE),
  RADIUS(←CIRCLE,←R).

/*TYPICAL DROP WITHIN CIRCLE SEGMENT*/
TYPICAL←DROP(←PATH,←START,←R*(SIN(←DIR1)-SIN(←DIR2))) :-
  PARTOF(←PATH,←CIRCLE), CIRCLE(←CIRCLE),
  CC(ANGLE(←START,←DIR1,←CIRCLE)),
  NORMAL(←PATH,←DIR2),
  RADIUS(←CIRCLE,←R).

/*NORMAL TO PATH*/

/*ALWAYS USE NORMAL OF LARGEST PATH*/
NORMAL(←PATH,←DIR) :-
  BITOF(←PATH,←SUPERPATH), NORMAL(←SUPERPATH,←DIR).

/*NORMAL IS ANGLE AT TYPICAL POINT*/
NORMAL(←PATH,←DIR) :-
  CC(TYPICAL←POINT(←PATH,←X)),
  CC(ANGLE(←X,←DIR,←PATH)).


/*AN OBJECT IS IN CONTACT WITH THE PATH IT MOVES ON*/
CONTACT(←PART,←PATH,←PER) :-
  DBC(MOTION(←PART,←PATH,←START,←SIDE,←PER)).

/*PATHS ARE FIXED IN THE ROLLER-COASTER WORLD*/
FIXED←CONTACT(←PATH,EARTH,←PER) :-
  PATH(←PATH), PERIOD(←PER),
  PROBTYPE(ROLLER-COASTER).

/*PATHS ARE SMOOTH IN THE ROLLER-COASTER WORLD*/
COEFF(←PATH,ZERO) :-
  PATH(←PATH), PROBTYPE(ROLLER-COASTER).
```

```
/*THE MASS OF A PARTICLE IS CONSTANT*/
MASS(←PART,←M,←PER1) :-
  PARTICLE(←PART), PERIOD(←PER2),
  DIFF(←PER1,←PER2), NLC(MASS(←PART,←M,←PER2)).

/*ACCEL TOWARDS THE CENTRE OF A CURVE*/
ACCEL(←PART,-(←V:2)/←R,←DIR2,←TIME) :-
  DBC(MOTION(←PART,←PATH,←START,←SIDE,←TIME)),
  CC(VEL(←PART,←V,←DIR1,←TIME)),
  RADIUS←OF←CURVATURE(←PATH,←R),
  CC(NORMAL(←PATH,←DIR2)).

/*REACTION*/
FORCE(←PART,←N,←DIR,←TIME) :-
  CONTACT(←PART,←OBJ,←TIME),
  CC(REACTION(←OBJ,←PART,←N,←DIR,←TIME)).

/*FRICTION*/
FORCE(←PART,←F,←DIR,←TIME) :-
  CONTACT(←PART,←OBJ,←TIME),
  THNOT(COEFF(←OBJ,ZERO)),
  CC(FRICTION(←OBJ,←PART,←F,←DIR,←TIME)).



\\\\\
```

```
%%%%%SCHEMA  .  OLD    @16:56 19-APR-1977 <055> (170)


/*SCHEMATA (AND CUES)*/
/*ALAN BUNDY 28/2/77 */


/*CUE IN SCHEMA*/
CUE(←KEY) :- SCHEMA(←KEY,←DECL,←ASS,←DEF),
  TRACE(PULLING-IN-SCHEMA-←KEY,6),NEWLINE,
  TRACE(DECLARATIONS,7),CHECKLIST(CALL,←DECL),NEWLINE,
  TRACE(ASSERTIONS,7), CHECKLIST(PASSERT,←ASS),NEWLINE,
  TRACE(DEFAULTS,7), CHECKLIST(PASSERTC,←DEF),NEWLINE, !.

SCHEMA(TIMESYS(←PER,←MOM1,←MOM2),
  [DECLARE(PERIOD(←PER)),CC(INITIAL(←PER,←MOM1)),
    CC(FINAL(←PER,←MOM2))],
  [MOMENT(←MOM1), MOMENT(←MOM2)],
  [] ).

SCHEMA(MOTSYS(←PART,←PATH,←START,←SIDE1,←PER),
  [CC(FAREND(←PATH,←FINISH,←START)), CC(FINAL(←PER,←END)),
   CC(INCLINE(←PATH,←ANG,←FINISH)), CC(VEL(←PART,←V,←ANG,←END))],
  [DBC(MOTION(←PART,←PATH,←START,←SIDE1,←PER)),
   DBC(AT(←PART,←FINISH,←END)) ],
  [(←V>ZERO :- TOP(←PATH,←START), PROBTYPE(ROLLER-COASTER))] ).


CALL(←L) :- ←L.



\\\\\
```

```
%%%%%FUNCC    .  OLD    @16:56 19-APR-1977 <055> (180)


/*FUNCC*/
/*FUNCTION CALL PREDICATE CALL*/
/*ALAN BUNDY FEB 1977*/



/*DECIDE TYPE OF CALL*/
FPC(←L) :- TRACE(CALLING-←L,7),FAIL.
FPC(←L) :- CHECKARGS(←L), !, FC(←L).
FPC(←L) :- ←L.

/*FUNCTION CALL*/
FC(←L) :- SILLY(←L), !, FAIL.
FC(←L) :- ←L, !.

/*PREDICATE ARGS ALL BOUND*/
CHECKARGS(←L) :- ←L=.. ←PROP.←ARGS, SEPERATE(←ARGS,←PARGS,←FARGS,←NUM),
  CHECKLIST(BOUND,←PARGS).

/*IS CALL SILLY?*/
SILLY(←L) :- ←L=.. ←PROP.←ARGS, SEPERATE(←ARGS,←PARGS,←FARGS,←NUM),
  CHECKLIST(BOUND,←FARGS), SEPERATE(←NARGS,←PARGS,←NFARGS,←NUM),
  ←NL =.. ←PROP.←NARGS, DBC(←NL), DIFF(←FARGS,←NARGS).

/*SEPERATE FUNCTION AND PREDICATE ARGS*/
SEPERATE([←Q],[],[←Q],1).
SEPERATE([←OBJ,←Q],[←OBJ],[←Q],2).
SEPERATE([←OBJ,←Q,←TIME],[←OBJ,←TIME],[←Q],3).
SEPERATE([←OBJ,←Q,←DIR,←TIME],[←OBJ,←TIME],[←Q,←DIR],4).
SEPERATE([←OBJ1,←OBJ2,←Q,←DIR,←TIME],[←OBJ1,←OBJ2,←TIME],[←Q,←DIR],5).



\\\\\
```

```
%%%%%CNVERT  .  OLD    @10:34 20-APR-1977 <055> (818)


/*CNVERT*/
/*MECHANICS UNIT CONVERSION ROUTINES*/
/*GATHERED TOGETHER BY ALAN BUNDY ON 8/9/76*/



/*UNIT CONVERSION*/


CONVERT(←ES,←ES1) :- STANDSYS(←SYS),
  ASSERTD(CCFLAG(ON)), ECONV(←ES,←ES1).

SCONV(NIL,NIL).

SCONV(←E.←ES,←E1.←ES1) :- ECONV(←E,←E1), SCONV(←ES,←ES1).

ECONV(TRUE,TRUE) :- !.

ECONV(←E,←E) :- ISCONST(←E), !.

ECONV(←E,←E) :- WORD(←E), QUANTITY(←E,1,ND).

ECONV(←E,←E1) :- WORD(←E) , !, VCONV(←E,←E1).

ECONV(←E,←E1) :-  ←E =..←S.←ARGS , SCONV(←ARGS,←ARGS1) ,
     ←E1 =..←S.←ARGS1 .

VCONV(←E,←E1) :- VCONV1(←E,←E1),!.

VCONV(←E,←E1) :- VCONV2(←E,←E1),!,
    CASSERTC(VCONV1(←E,←E1)),TRACE(←E-GOES-TO-←E1,4).

VCONV2(←V,←M) :- UNIT(←V,←U) , STANDARD(←U,←U),
!, CC(MEASURE(←V,←M)).

VCONV2(←V,←M2) :- UNIT(←V,←U)
, STANDARD(←U,←U1) , CC(MEASURE(←V,←M)),
    MCONV(←M,←U,←M1,←U1), SIMPLIFY(←M1,←M2).

STANDARD(←U,←U1) :- STANDSYS(←SYS), DIMENSIONS(←U,←D),
    STANDUNIT(←D,←U1,←SYS).


DIMENSIONS(LBS,M) :- !.

DIMENSIONS(TONS,M) :- !.


DIMENSIONS(HRS,T) :- !.

DIMENSIONS(MINS,T) :- !.
```

```
DIMENSIONS(SECS,T) :- !.

DIMENSIONS(MLS,L) :- !.

DIMENSIONS(YDS,L) :- !.

DIMENSIONS(FT,L) :- !.

DIMENSIONS(INS,L) :- !.

DIMENSIONS(←U,←U) :- ISCONST(←U), !.

DIMENSIONS(←U,←D) :-  ←U =..←S.←ARGS  ,
    LDIMENSIONS(←ARGS,←DARGS),  ←D =..←S.←DARGS .


LDIMENSIONS(NIL,NIL).

LDIMENSIONS(←U.←UL,←D.←DL) :- DIMENSIONS(←U,←D) ,
    LDIMENSIONS(←UL,←DL).

MCONV(←M,←U,←C.←M,←U1) :- FACTOR(←C,←U1,←U).

INA(2240,LBS,TONS).

INA(60,MINS,HRS).

INA(60,SECS,MINS).

INA(1760,YDS,MLS).

INA(3,FT,YDS).

INA(12,INS,FT).

FACTOR(1,←U,←U) :- ISUNIT(←U), !.

FACTOR(←U,←U,←U) :- ISCONST(←U), !.  /*FOR SQUARES ETC*/

FACTOR(←C,←U1,←U) :- INA(←C,←U1,←U), !.

FACTOR(←C,←U1,←U) :- COMP(←U1),
     ←U1 =..←.←ARGS1 ,  ←U =..←S.←ARGS ,
    SFACTOR(←CS,←ARGS1,←ARGS),  ←C =..←S.←CS , !.

FACTOR(1/←C,←U1,←U) :- BEFORE(←U,←U1), FACTOR(←C,←U,←U1), !.

FACTOR(←C1.←C2,←U2,←U) :- INA(←C1,←U1,←U),
    FACTOR(←C2,←U2,←U1), !.

SFACTOR(NIL,NIL,NIL) :- !.

SFACTOR(←C.←CS,←U1.←US1,←U.←US) :- FACTOR(←C,←U1,←U),
    SFACTOR(←CS,←US1,←US).

BEFORE(←U,←U1) :- DIMENSIONS(←U,←D), POSN(←D,←U,←P),
    POSN(←D,←U1,←P1), ←P<←P1, !.

POSN(M,LBS,1).
```

```
POSN(M,TONS,2).

POSN(T,SECS,1).

POSN(T,MINS,2).

POSN(T,HRS,3).

POSN(L,INS,1).

POSN(L,FT,2).

POSN(L,YDS,3).

POSN(L,MLS,4).

ISUNIT(←U) :- POSN(←D,←U,←N).




/*FIXING UNIT SYSTEM*/

STANDSYS(←SYS) :-
  NOLOOP ,SOUGHTS(←SL), CONVLIST(UNIT,←SL,←UL1),
  SWORDSIN(←UL1,←PUL1), FIX(T.L.M.NIL,←PUL1,←REM1,←SYS),
  GIVENS(←GL) ,CONVLIST(UNIT,←GL,←UL2),
  SWORDSIN(←UL2,←PUL2), FIX(←REM1,←UL2,←REM2,←SYS),
  FIXREST(←REM2,←SYS),DECLARE(STANDSYS(←SYS)),
  SYSFOUND.

NOLOOP :- SYSSOUGHT, !, FAIL.
NOLOOP :- ASSERT(SYSSOUGHT).

SYSFOUND :- DENY(SYSSOUGHT).

FIX(NIL,←UL,NIL,←SYS) .

FIX(←D1.←DL1,←UL,←DL2,←SYS) :-
  SUBLIST(DIMENSIONS(←D1),←UL,←CANDL), DIFF(←CANDL,NIL),
   !, VOTE(←CANDL,←WIN),
  GENSYM(SYS,←SYS), DECLARE(STANDUNIT(←D1,←WIN,←SYS)),
  FIX(←DL1,←UL,←DL2,←SYS).

FIX(←D1.←DL1,←UL,←D1,←DL2,←SYS) :-
  FIX(←DL1,←UL,←DL2,←SYS).

VOTE(←CANDL,←WIN) :-
  LISTTOSET(←CANDL,←CANDS),
  MAPLIST(SCOREEACH(←CANDL),←CANDS,←PAIRSL),
  FAVOURITE(←PAIRSL,←WIN,←SCORE).

SCOREEACH(←U,←U.0,NIL).
SCOREEACH(←U,←U.←N1,←U.←REST) :-
  !, SCOREEACH(←U,←U,←N,←REST) ,+(←N,1,←N1).
SCOREEACH(←U,←U.←N,←HD,←TL) :- SCOREEACH(←U,←U.←N,←TL).
```

```
FAVOURITE((←U,←S),NIL,←U,←S) :- !.
FAVOURITE((←U1,←S1),←PL,←U2,←S2) :-
  FAVOURITE(←PL,←U2,←S2), ←S1<←S2 , !.
FAVOURITE((←U,←S),←PL,←U,←S).

FIXREST(NIL,←SYS).

FIXREST(T,L,M,NIL,←SYS) :- !, DEFAULT(←SYS).

FIXREST(←D,←DL,←SYS) :-
  DEFAULT(←SYS2) ,STANDUNIT(←D,←U,←SYS2),
  DECLARE(STANDUNIT(←D,←U,←SYS)),FIXREST(←DL,←SYS).

DEFAULT(FLS).

STANDUNIT(T,SECS,FLS) :- !.
STANDUNIT(L,FT,FLS) :- !.
STANDUNIT(M,LBS,FLS) :- !.

STANDUNIT(←U,←U,←SYS) :- ISCONST(←U) ,!.

STANDUNIT(←D,←U,←SYS) :- ISSYM(←D), !,FAIL.

STANDUNIT(←D,←U,←SYS) :- ←D=..(←S.←DARGS) ,
  MAPLIST(STANDUNIT(←SYS),←DARGS,←UARGS), ←U=..(←S.←UARGS) .
```

\\\\\

%%%%I I . @11:44 7-JAN-1977 <Ø57> (4)

:- I I.

\\\\\

```
%%%%%OLDXRT  .          @13:39 2-MAR-1977 <055> (287)


/*FRAMES AND GO, OLD BITS OF XTRACT*/

/* FRAMES */


FRAME1(PULLSYS(←SYS,←PULL,←STR,←P1,←P2,←TIME),
    [ PROBTYPE(PULLEY),
      CONTACT(←P1,LEFT(←STR),←TIME),
      CONTACT(←P2,RIGHT(←STR),←TIME),
      (DIRECTION(LEFT(←STR),90,←TIME) :- !),
      (DIRECTION(RIGHT(←STR),90,←TIME) :- !),
      STRING(←STR),
      PARTICLE(←PULL) ]).


FRAME2(PULLSYS(←SYS,←PULL,←STR,←P1,←P2,←TIME),
      [(TENSION(LEFT(←STR),←T,←TIME) :- TENSION(←STR,←T,←TIME),
        COEFF(←PULL,ZERO),ELASTIC(←STR,ZERO)),
       (TENSION(RIGHT(←STR),←T,←TIME) :- TENSION(←STR,←T,←TIME),
        COEFF(←PULL,ZERO),ELASTIC(←STR,ZERO))    ]).


FRAME3(PULLSYS(←SYS,←PULL,←STR,←P1,←P2,←TIME),
      [FIXED←CONTACT(←PULL,EARTH,←TIME),
        COEFF(←PULL,ZERO),
       ELASTIC(←STR,ZERO),
       MASS(←PULL,ZERO,←TIME),
       MASS(←STR,ZERO,←TIME)  ]).

FRAME(←CUE,←L,←DEFAULTS) :-
    FRAME1(←CUE,←L1),FRAME2(←CUE,←L2),APPEND(←L1,←L2,←L),
    FRAME3(←CUE,←DEFAULTS).



/* MARPLES ALGORITHM*/


GO :- SOUGHTS(←XS), GIVENS(←GS),
    TRACE(ATTEMPTING-TO-SOLVE-FOR-←XS-IN-TERMS-OF-←GS,5),
    GETEQNS(←XS,←GS,NIL,←ES,←XS1),
    TRACE(EQUATIONS-EXTRACTED,2), PPR(←ES) ,
    CONVERT(←ES,←ES1), TRACE(CONVERTED-TO,2), PPR(←ES1),
    SIMPLIFY(←ES1,←ES2), TRACE(SIMPLIFIED-TO,2), PPR(←ES2),
    MAPLIST(CDMEASURE,←XS1,←XS2), TRACE(UNKNOWNS-ARE-←XS2,2),
    SIMSOLVE(←ES2,←XS2,←ANS), TRACE(ANSWER-IS,2), PPR(←ANS).



\\\\\
```