SRI International

Working Notes on PARADISE Chess Patterns

Technical Note 509

August 15, 1991

By:

David E. Wilkins
Senior Computer Scientist

Artificial Intelligence Center
Computer and Information Sciences Division

Abstract

This report contains the patterns used in PARADISE, a knowledge-based chess program written at Stanford University in the late 1970s. This report contains primarily data and is intended for people writing pattern-based game-playing programs who wish to know the details of the patterns in PARADISE.

1 Introduction

PARADISE (PAttern Recognition Applied to DIrecting SEarch) was written in the late 1970s to find the best move in tactically sharp middle game positions from the games of chess masters [5, 6, 7, 8]. It's goal was to build an expert knowledge base and to reason with it to discover plans and verify them with a small tree search. Like human masters, the program had a large number of stored "patterns", and analyzing a position involved matching these patterns to suggest plans for attack or defense. This analysis was verified and possibly corrected by a small search of the game tree (tens of positions).

Work on PARADISE ended in 1978, but work on chess programs has continued to the present. However, most of this work has been on performance-oriented, search-based programs. In the decade after PARADISE, there was very little work on knowledge-based programs. Recently, there has been a rekindling of interest in pattern-based chess programs [1, 2, 3], particularly for addressing issues in machine learning. Many researchers involved in this area have requested access to PARADISE's database of chess patterns. This report provides this access. This introduction is followed by a listing of all the patterns used in PARADISE. Following that are a set of working notes made during the development of these patterns. These working notes are unedited, not grammatically correct, and not guaranteed correct. They are included because some researchers have found them useful.

To interpret the patterns, the reader should be familiar with the literature on PARADISE [5, 6, 8]. When the working notes refer to chess positions by number, these are positions from the book *Win at Chess* [4]. This data is presented in the hope that it will assist the efforts of current and future researchers.

References

[1] Levinson, R., A self-learning, pattern-oriented chess program. *International Computer Chess Association Journal*, 12(4):207–215, December 1989.

[2] Levinson, R. and Snyder, R., Adaptive Pattern oriented chess. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 1991.

[3] Marsland, T.A. and Schaeffer, J., editors. *Computers, Chess and Cognition.* Springer-Verlag, 1990.

[4] Reinfeld, .F, *Win At Chess*, Dover Books, 1958.

[5] Wilkins, D.E., "Using patterns and plans to solve problems and control search", AIM-329, Computer Science Department, Stanford University, 1979.

[6] Wilkins, D.E., "Using Knowledge to Control Tree Searching", *Artificial Intelligence 18*, 1982, pp. 1-51.

[7] Wilkins, D.E., "Using Patterns and Plans in Chess", *Artificial Intelligence 14*, 1980, pp. 165-203.

[8] Wilkins, D.E., "Using Chess Knowledge to Reduce Search", Chapter 10 in *Chess Skill in Man and Machine*, Peter Frey editor, Springer-Verlag, 1983.

APPENDIX A

Patterns used by PARADISE

```
PRIMITIVE
DIR
((P1 SQ))

PRIMITIVE
THRU
((LP1 P1 SQ))

PRIMITIVE
ETHRU
((LP1 DP1 SQ))

PRIMITIVE
OTHRU
((LP1 P1 SQ))

PRIMITIVE
DSC
((LP1 P1 SQ))

PRIMITIVE
OBJ
((LP1 DP1 SQ))

PRIMITIVE
LEGALMOVE
((P1 SQ))


PRIMITIVE
PINRAY
((LP1 DP1 DP2)
(OR     (PATTERN ETHRU LP1 DP1 (LOC DP2 BD))
        (PATTERN OBJ LP1 DP1 (LOC DP2 BD))))

PRIMITIVE
PIN
((LP1 DP1 DP2)
(PATTERN PINRAY LP1 DP1 DP2)
(FUNCTION SAFEP LP1 (LOC DP2 BD))
(NEVER  (FUNCTION EXCHANGE LP1 (LOC DP2 BD)))
(NEVER (EXISTS (P1)
        (PATTERN LEGALMOVE P1 (LOC DP2 BD))
        (GREAT (VALUE DP2) (VALUE P1)))))

PRIMITIVE
ONLYDIR
((P1 SQ)
(PATTERN DIR P1 SQ)
(NEVER (EXISTS (P2) (PATTERN DIR P2 SQ) (FUNCTION NEQ P2 P1))))

PRIMITIVE
ONLYPRO
((P1 SQ)
(PATTERN ONLYDIR P1 SQ)
(NEVER (EXISTS (P2) (PATTERN OTHRU P2 P1 SQ))))
```

```
PRIMITIVE
DSCTHR
((P1 P2 DP1 SQ)
(PATTERN DSC P2 P1 (LOC DP1 BD))
(FUNCTION SAFEP P2 (LOC DP1 BD))
(NEVER (FUNCTION EXCHANGE P2 (LOC DP1 BD)))
(PATTERN LEGALMOVE P1 SQ)
(NEVER (FUNCTION LINE (LOC P2 BD) SQ (LOC DP1 BD)))
(IMPLIES NIL ((PATTERN DIR DP1 SQ))
 (NEVER (PATTERN ONLYPRO P1 SQ))
 (NEVER (FUNCTION OVERPRO (COLOR P1) SQ) (FUNCTION NEQ 0 (TYPE DP1)))
 (NEVER (FUNCTION SAFEP DP1 SQ))))

PRIMITIVE
MOBIL
((P1 SQ)
(PATTERN LEGALMOVE P1 SQ)
(OR (FUNCTION SAFEP P1 SQ)
 (EXISTS (P2 DP1)  (PATTERN DSCTHR P1 P2 DP1 SQ)
  (ASGREAT (VALUE DP1) (VALUE P1))
  (IMPLIES (DP2) ((OR   (PATTERN THRU DP2 P1 (LOC P2 BD))
                        (PATTERN DIR DP2 (LOC P2 BD))))
    (OR (GREAT (EXCHVAL P1 SQ) (EXCHVAL DP2 (LOC P2 BD)))
        (EXISTS (DP3) (FUNCTION CANATTACK P1 SQ (LOC DP3 BD))
          (FUNCTION SAFEP P1 (LOC DP3 BD))
          (NEVER (PATTERN DIR DP3 (LOC P2 BD)))
          (NEVER (PATTERN THRU DP3 P1 (LOC P2 BD)))
          (GREAT (PLUS (VALUE (OCCUPANT SQ BD)) (EXCHVAL P1 (LOC DP3 BD)))
                 (EXCHVAL DP2 (LOC P2 BD)))))))
 (EXISTS (LP1 DP1) (PATTERN DSC LP1 P1 SQ) (PATTERN ONLYPRO DP1 SQ))))

PRIMITIVE
MOBSTAY
((P1 SQ)
(PATTERN MOBIL P1 SQ)
(IMPLIES NIL ((FUNCTION SAFEP P1 SQ))
        (NEVER (FUNCTION EXCHANGE P1 SQ))))

PRIMITIVE
CHECKMOVE
((P1 SQ)
(PATTERN LEGALMOVE P1 SQ)
(EXISTS (DK) (OR  (FUNCTION CANATTACK P1 SQ (LOC DK BD))
        (EXISTS (LP1) (PATTERN DSC LP1 P1 (LOC DK BD))
                (NEVER (FUNCTION LINE (LOC LP1 BD) SQ (LOC DK BD)))))))

PRIMITIVE
MATE
((P1 SQ)
(PATTERN CHECKMOVE P1 SQ)
(NEVER (EXISTS (SQ1 DK) (FUNCTIONE NEWFLIGHT SQ1 P1 SQ DK (LOC DK BD)))))
```

```
PRIMITIVE
PROTECTS
((P1 P2 SQ1 DP1)
(OR
((PATTERN LEGALMOVE DP1 (LOC P2 BD))
        (PATTERN ONLYPRO P1 (LOC P2 BD))
        (FUNCTION EQUAL SQ1 (LOC P2 BD)))
(EXISTS (DK) (PATTERN DIR DK (LOC P2 BD))
        (PATTERN ONLYPRO P1 (LOC P2 BD))
        (FUNCTION EQUAL DP1 DK) (FUNCTION EQUAL SQ1 (LOC P2 BD)))
(EXISTS (K) (PATTERN MATE DP1 SQ1)
        (PATTERN ONLYPRO P1 SQ1)
        (NEVER (EXISTS (SQ3) (FUNCTION LINE SQ1 SQ3 (LOC K BD))
                (FUNCTION OVERPRO (COLOR DP1) SQ3)
                (NEVER (PATTERN MOBIL P1 SQ3))))
        (FUNCTION EQUAL K P2))
(EXISTS (SQ2 K) (PATTERN MATE DP1 SQ2)
        (NEVER (PATTERN DIR P1 SQ2))
        (UNIQUE (FUNCTION LINE SQ2 SQ1 (LOC K BD))
                (FUNCTION OVERPRO (COLOR DP1) SQ1))
        (PATTERN ONLYPRO P1 SQ1)
        (PATTERN MOBIL P1 SQ1)
        (FUNCTION EQUAL P2 K))))

PRIMITIVE
ENPRIS
((P1 DP1)
(PATTERN MOBSTAY P1 (LOC DP1 BD)))
```

```
THREAT
MATENOW
((P2 SQ1 DK)
(PATTERN MATE P2 SQ1)
(PATTERN MOBSTAY P2 SQ1)
(IMPLIES NIL ((NEVER (FUNCTION CANATTACK P2 SQ1 (LOC DK BD))))
  (NEVER (EXISTS (LP1 DP1) (PATTERN DSC LP1 P2 (LOC DK BD))
                (PATTERN LEGALMOVE DP1 (LOC LP1 BD))
                (FUNCTION NEQ DP1 (OCCUPANT SQ1 BD)))))
(NEVER (EXISTS (DSP1) (PATTERN DIR DSP1 SQ1)))
(IMPLIES NIL ((PATTERN DIR DK SQ1)) (NEVER (PATTERN ONLYPRO P2 SQ1)))
(NEVER (EXISTS (DSP1 SQ2 OP1) (FUNCTION LINE SQ1 SQ2 (LOC DK BD))
  (OR   ((PATTERN MOBIL OSP1 SQ2)  (FUNCTION NEQ DSP1 (OCCUPANT SQ1 BD))
          (NEVER (IMPLIES (DP2) ((OR (PATTERN OIR DP2 SQ2)
            (PATTERN THRU OP2 P2 SQ2)
            (EXISTS (DP3) (PATTERN OTHRU OP2 DP3 SQ2))))
           (OR (FUNCTION EQUAL OP2 DSP1) (FUNCTION EQUAL DP2 (OCCUPANT SQ1 BD))))))
        ((PATTERN DIR DSP1 SQ2) (PATTERN PIN P2 DSP1 DP1)
          (FUNCTION NEQ DSP1 (OCCUPANT SQ1 BD))))))
(NEVER (EXISTS (P3 SQ) (PATTERN DIR DK SQ)
        (NEVER (FUNCTION OCCBYCOL SQ (COLOR OK)))
        (UNIQUE (PATTERN OIR P3 SQ) (FUNCTION NEQ P3 P2))
        (FUNCTION LINE (LOC P3 BD) SQ1 SQ)
        (NEVER (EXISTS (LP1) (PATTERN THRU LP1 P2 SQ)))
        (NEVER (FUNCTION CANATTACK P2 SQ1 SQ))))
(PLANNEW ((P2 SQ1)) (LIKELY 0) (THREAT 1200))
(RETURN))

THREAT
MATENOW1
((P2 SQ1 DK)
(PATTERN MATE P2 SQ1)
(IMPLIES NIL ((NFVER (FUNCTION CANATTACK P2 SQ1 (LOC DK BD))))
  (NEVER (EXISTS (LP1 DP1) (PATTERN DSC LP1 P2 (LOC DK BD))
                (PATTERN ENPRIS DP1 LP1)
                (FUNCTION NEQ DP1 (OCCUPANT SQ1 BD)))))
(PATTERN MOBSTAY P2 SQ1)
(PLANNEW ((P2 SQ1)) (LIKELY 0) (THREAT (PLUS 5 (WIN OK)))))

THREAT
MATENOW2
((P1 SQ DSP1 OK)
(PATTERN MATENOW1 P1 SQ DK)
(PATTERN ONLYPRO DSP1 SQ)
(ACTIONNEW SAFER (P1 SQ) (LIKELY 0) (THREAT (PLUS 10 (WIN DK)))
 (PLAN (DSP1 NIL) (CHKMOVE P1 SQ))
 (DECOYCONO (IMPLIES (SQ3)
         ((PATTERN DIR OK SQ3) (NEVER (FUNCTION OCCBYCOL SQ3 (COLOR DK))))
        (OR (NEVER (PATTERN ONLYPRO P2 SQ3)) (FUNCTION CANATTACK P1 SQ SQ3))))))
```

```
THREAT
TRAPPED
((DMP1)
(NOT (EXISTS (SQ) (PATTERN MOBIL DMP1 SQ)))
(NEVER (EXISTS (P1) (PATTERN ENPRIS P1 DMP1)))
(ACTIONNEW ATTACK ((COMPCOLOR DMP1) (LOC DMP1 BD)) (THREAT (WIN DMP1))
  (COND (NEVER (EXISTS (SQ2) (PATTERN DIR DMP1 SQ2)
        (NEVER (FUNCTION OCCBYCOL SQ2 (COLOR DMP1)))
        (PATTERN ONLYPRO P1 SQ2)
        (NEVER (FUNCTION CANATTACK P1 SQ SQ2))
        (NEVER (FUNCTION LINE SQ (LOC DMP1 BD) SQ2))))))

THREAT
EXCHTRAP
((DMP1 OP1 SQ1 P1 P2)
(PATTERN TRAPPED OMP1)
(PATTERN LEGALMOVE P1 (LOC DMP1 BD))
(NEVER (PATTERN MOBIL P1 (LOC DMP1 BD)))
(PATTERN ENPRIS DP1 P1) (PATTERN ETHRU DP1 P1 (LOC DMP1 BD))
(PATTERN ONLYPRO DP1 (LOC P1 BD))
(PATTERN LEGALMOVE P2 SQ1)
(FUNCTION CANATTACK P2 SQ1 (LOC P1 BD))
(FUNCTION CANATTACK P1 (LOC P1 BD) (LOC DMP1 BD))
(OR     (PATTERN MOBSTAY P2 SQ1)
        (IMPLIES (OP2) ((PATTERN DIR DP2 SQ1))
 (OR ((FUNCTION EQUAL DP2 DP1) (GREAT (VALUE DMP1) (VALUE P2)))
     ((PATTERN ONLYPRO DP2 (LOC DP1 BD)) (GREAT (VALUE DP1) (VALUE P2))))))
(GREAT (PLUS (VALUE DP1) (EXCHVAL P2 (LOC DMP1 BD)))  (VALUE P1))
(PLANNEW ((P2 SQ1) (((DP1 (LOC P1 BD)) (P2 (LOC P1 BD)) NIL (KILL P2 DMP1))
  ((ANYBUT DP1) (P1 (LOC DP1 BD)) (NIL (LOC DP1 BD)) (SAFEMOVE P2 (LOC P1 BD)))
  ((DP1 NIL) (KILL P1 DMP1))))
 (LIKELY 1) (THREAT (FORKT (EXCH P2 DMP1)
        (PLUS (EXCH P2 DMP1) (LESS (WIN DP1) (WIN P1)))))))

THREAT
DSCATT
((P1 P2 DP1 OP2 SQ)
(PATTERN DSCTHR P2 P1 OP1 SQ)
(NEVER (PATTERN TRAPPED DP1))
(FUNCTION CANATTACK P2 SQ (LOC DP2 BD))
(FUNCTION NEQ DP1 DP2)
(FUNCTION SAFEP P2 (LOC DP2 BD))
(NEVER (PATTERN DIR P2 (LOC DP2 BD)))
(NEVER (FUNCTION EXCHANGE P2 (LOC DP2 BD)))
(PIECESAFE P1 (EXCHVAL P2 (LOC DP2 BD)))
(IMPLIES NIL ((OR (FUNCTION SAMEVAL OP1 P2) (GREAT (VALUE P2) (VALUE DP1))))
        (NEVER (PATTERN DIR DP1 SQ)))
(PLANNEW ((P2 SQ) (((ANYBUT DP2) (KILL P2 DP2)) ((ANYBUT DP1) (KILL P1 DP1))))
  (THREAT (PLUS (FORKT (EXCH P2 DP2) (EXCH P1 OP1)) (EXCHVAL P2 SQ)))
  (LIKELY 0)))
```

```
THREAT
CANTRAP
((P1 SQ DMP1)
(PATTERN LEGALMOVE P1 SQ)
(FUNCTION CANATTACK P1 SQ (LOC DMP1 BD))
(FUNCTION NEQ (TYPE DMP1) 0)
(FUNCTION SAFEP P1 (LOC DMP1 BD))
(NEVER (PATTERN TRAPPED DMP1))
(NEVER (EXISTS (P2) (PATTERN ENPRIS P2 DMP1)))
(NEVER (PATTERN DIR P1 (LOC DMP1 BD)))
(IMPLIES (SQ2) ((PATTERN MOBIL DMP1 SQ2))
 (OR    (FUNCTION LINE (LOC DMP1 BD) SQ SQ2)
        (EXISTS (P2) (PATTERN THRU P2 P1 SQ2)
          (OR   (NEVER (FUNCTION OVERPRO (COLOR P2) SQ2))
                (GREAT (VALUE DMP1) (VALUE P2))))
        ((OR    (NEVER (FUNCTION OVERPRO (COLOR P1) SQ2))
                (GREAT (VALUE DMP1) (VALUE P1)))
         (OR    (FUNCTION CANATTACK P1 SQ SQ2)
                (FUNCTION LINE SQ (LOC DMP1 BD) SQ2)))
        (EXISTS (P2 P3 SQ3) (PATTERN DSC P3 P2 SQ2)
                (FUNCTION NEQ P3 P1)
                (PATTERN ONLYPRO DMP1 SQ2)
                (PATTERN CHECKMOVE P2 SQ3)
                (GREAT (VALUE DMP1) (PLUS (EXCHVAL DMP1 SQ2) (VALUE P2))))))
(NEVER (EXISTS (SQ1) (PATTERN ONLYPRO P1 SQ1)
        (NEVER (FUNCTION CANATTACK P1 SQ SQ1))
        (PATTERN LEGALMOVE DMP1 SQ1)))
(NEVER (EXISTS (SQ1) (PATTERN OBJ DMP1 P1 SQ1)
        (NEVER (FUNCTION OCCBYCOL SQ1 (COLOR DMP1)))
        (FUNCTION SAFEP DMP1 SQ1)))
(ACTIONNEW MOVE1 (P1 SQ) (THREAT (EXCHVAL P1 (LOC DMP1 BD)))
  (LIKELY 0) (SAFECDND (FUNCTION NEQ DP1 DMP1))
  (PLAN (SAFEMOVE P1 SQ) NIL (KILL P1 DMP1))))


THREAT
DSCATTM
((P1 P2 DP1 DK SQ P3 SQ1)
(PATTERN DSCTHR P2 P1 DP1 SQ)
(NEVER (PATTERN TRAPPED DP1))
(PATTERN MATE P3 SQ1) (FUNCTION NEQ P3 P2)
(FUNCTION CANATTACK P2 SQ SQ1)
(NEVER (FUNCTION LINE (LOC P3 BD) SQ SQ1))
(OR     (PATTERN ONLYDIR DK SQ1)
        ((PATTERN DIR DP1 SQ1) (NEVFR (FUNCTION OVERPRO (COLOR P1) SQ1))
          (NEVER (EXISTS (SQ3) (PATTERN MOBSTAY DP1 SQ3)
                (FUNCTION CANATTACK DP1 SQ3 SQ1)
                (NEVER (FUNCTION CANATTACK P2 SQ SQ3))))))
(IMPLIES (SQ2)  ((PATTERN DIR DK SQ2) (NEVER (FUNCTION OCCBYCOL SQ2 (COLOR DK))))
  (OR (NEVER (PATTERN ONLYPRO P2 SQ2)) (FUNCTION CANATTACK P3 SQ1 SQ2)
        (FUNCTION CANATTACK P2 SQ SQ2)))
(IMPLIES NIL ((OR (FUNCTION SAMEVAL DP1 P2) (GREAT (VALUE P2) (VALUE DP1))))
        (NEVER (PATTERN DIR DP1 SQ)))
(PLANNEW ((P2 SQ) ((NIL (MATEMOVE P3 SQ1)) (NIL (KILL P1 DP1))))
  (THREAT (PLUS (EXCH P1 DP1) (EXCHVAL P2 SQ)))  (LIKELY 0)))
```

```
THREAT
ENPRISE
((P1 DP1)
(PATTERN ENPRIS P1 DP1)
(PLANNEW ((P1 (LOC DP1 BD))) (THREAT (EXCH P1 DP1)) (LIKELY 0)
  (PLAN1 (KILL P1 DP1))))

THREAT
EP2
((P1 DP1 P2 DP2)
(PATTERN ENPRIS P1 DP1)
(OR     (GREAT (VALUE DP1) (EXCHVAL P1 (LOC DP1 BD)))
        (PATTERN CHECKMOVE P1 (LOC DP1 BD)))
(PATTERN ENPRIS P2 DP2)
(FUNCTION NEQ P1 P2)
(FUNCTION NEQ DP1 DP2)
(IMPLIES (DP3)
 ((OR (PATTERN DIR DP3 (LOC DP1 BD)) (PATTERN ETHRU DP3 P1 (LOC DP1 BD))))
  (FUNCTION NEQ DP3 DP2)
  (NEVER (EXISTS (K) (FUNCTION CANATTACK DP3 (LOC DP1 BD) (LOC K BD))))
  (NEVER (FUNCTION CANATTACK DP3 (LOC DP1 BD) (LOC DP2 BD)))))

THREAT
EP2P
((P1 DP1 P2 DP2)
(PATTERN EP2 P1 DP1 P2 DP2)
(GREAT (VALUE DP1) (EXCHVAL P1 (LOC DP1 BD)))
(PLANNEW ((P1 (LOC DP1 BD)) (NIL (LOC DP1 BD)) (KILL P2 DP2))
 (THREAT (FORKT (WIN DP1) (PLUS (EXCH P1 DP1) (EXCH P2 DP2))))
 (LIKELY 0)))

THREAT
EP2C
((P1 DP1 P2 DP2 DK)
(PATTERN EP2 P1 DP1 P2 DP2)
(PATTERN CHECKMOVE P1 (LOC DP1 BD))
(NEVER (EXISTS (SQ1) (PATTERN MOBIL DP2 SQ1)
  (NEVER (PATTERN ONLYPRO DP2 SQ1)) (FUNCTION LINE (LOC DP1 BD) SQ1 (LOC DK BD))))
(IMPLIES NIL ((ASGREAT (VALUE P2) (VALUE DP2)))
 (NEVER (EXISTS (SQ1) (PATTERN DIR DK SQ1)
        (FUNCTION CANATTACK DK SQ1 (LOC DP2 BD))
        (NEVER (FUNCTION CANATTACK P1 (LOC DP1 BD) SQ1))
        (NEVER (FUNCTION OCCBYCOL SQ1 (COLOR DK)))
        (OR (PATTERN MOBSTAY DK SQ1) (PATTERN ONLYPRO P1 SQ1)))))
(PLANNEW ((P1 (LOC DP1 BD)) NIL (KILL P2 DP2))
 (THREAT (PLUS (EXCH P1 DP1) (EXCH P2 DP2)))
 (LIKELY 0)))

THREAT
QUEENP
((PN1 SQ)
(PATTERN MOBSTAY PN1 SQ)
(FUNCTION R8 PN1 SQ)
(PLANNEW ((PN1 SQ)) (THREAT 120) (LIKELY 0)))
```

```
THREAT
FORKSQ
((SP1 SQ DMP1 DMP2)
(FUNCTIONE FRKSQ SP1 SQ DMP1 DMP2)
(NEVER (PATTERN DIR SP1 (LOC DMP1 BD)))
(NEVER (PATTERN DIR SP1 (LOC DMP2 BD))))

THREAT
FORKSQ1
((SP1 SQ DMP1 DMP2)
(PATTERN FORKSQ SP1 SQ DMP1 DMP2)
(IMPLIES NIL ((PATTERN DIR DMP1 SQ) (NEVER (PATTERN MOBSTAY SP1 SQ)))
 (GREAT (VALUE DMP1) (VALUE SP1)))
(IMPLIES NIL ((PATTERN DIR DMP2 SQ) (NEVER (PATTERN MOBSTAY SP1 SQ)))
 (GREAT (VALUE DMP2) (VALUE SP1)))
(ACTIONNEW MOVE (SP1 SQ) (THREAT (FORK SP1 DMP1 DMP2)) (LIKELY 0)
 (COND  (NEVER  (FUNCTION CANATTACK SP1 SQ2 (LOC DMP1 BD)))
        (NEVER  (FUNCTION CANATTACK SP1 SQ2 (LOC DMP2 BD))))
 (SAFECOND (FUNCTION NEQ DP1 DMP1) (FUNCTION NEQ DP1 DMP2))
 (PLAN (SAFEMOVE SP1 SQ)
   (((ANYBUT DMP1) (KILL SP1 DMP1)) ((ANYBUT DMP2) (KILL SP1 DMP2))))))

THREAT
FORKSQW
((SP1 SQ DMP1 DMP2)
(PATTERN FORKSQ1 SP1 SQ DMP1 DMP2)
(PATTERN LEGALMOVE SP1 SQ)
(EXISTS (DP1)
 (OR    (PATTERN ONLYDIR DP1 (LOC DMP1 BD))
        (PATTERN ONLYDIR DP1 (LOC DMP2 BD)))
 (OR    (FUNCTION EQUAL DP1 (OCCUPANT SQ BD))
        (FUNCTION LINE (LOC DP1 BD) SQ (LOC DMP1 BD))
        (FUNCTION LINE (LOC DP1 BD) SQ (LOC DMP2 BD))))
(ACTIONNEW MOVE1 (SP1 SQ) (THREAT (FORKT (WIN DMP1) (WIN DMP2)))
 (LIKELY 0) (SAFECOND (FUNCTION NEQ DP1 DMP1) (FUNCTION NEQ DP1 DMP2))
 (PLAN (SAFEMOVE SP1 SQ)
   (((ANYBUT DMP1) (KILL SP1 DMP1)) ((ANYBUT DMP2) (KILL SP1 DMP2))))))
```

```
THREAT
FORKSAC
((SP1 SQ DMP1 DMP2 DP1 P2)
(OR     (PATTERN FORKSQ SP1 SQ DMP1 DMP2)
        (PATTERN FORKSQ SP1 SQ DMP2 DMP1))
(OR     (PATTERN MOBSTAY SP1 SQ)
        ((PATTERN LEGALMOVE SP1 SQ)
         (PATTERN DIR DMP1 SQ)
         (OR (NEVER (FUNCTION OVERPRO (COLOR SP1) SQ))
             ((FUNCTION PAWN SP1) (EXISTS (LP1) (PATTERN DSC LP1 SP1 SQ))))))
(PATTERN ONLYPRO DP1 (LOC DMP1 BD))
(FUNCTION NEQ DP1 DMP2)
(IMPLIES NIL ((PATTERN DIR DMP2 SQ)) (GREAT (VALUE DMP2) (VALUE SP1)))
(PATTERN LEGALMOVE P2 (LOC DMP1 BD))
(FUNCTION NEQ P2 SP1)
(GREAT (PLUS (VALUE DP1) (VALUE DMP1)) (VALUE P2))
(GREAT (PLUS (VALUE DMP2) (VALUE DMP1)) (VALUE P2))
(IMPLIES NIL ((FUNCTION CANATTACK DP1 (LOC DMP1 BD) SQ))
 (GREAT (PLUS (VALUE DMP1) (VALUE OP1)) (PLUS (VALUE P2) (VALUE SP1))))
(PLANNEW ((P2 (LOC DMP1 BD)) (DP1 (LOC DMP1 BD)) (SAFEMOVE SP1 SQ)
  (((ANYBUT DP1) (KILL SP1 DP1)) ((ANYBUT DMP2) (KILL SP1 DMP2))))
 (LIKELY 0)
 (THREAT (LESS (PLUS (WIN (OCCUPANT SQ BD))
        (PLUS (WIN DMP1) (FORKT (WIN DP1) (EXCH SP1 DMP2))))
   (WIN P2)))))

THREAT
PINATT
((LP1 SQ DMP1 DMP2)
(FUNCTIONE PINAT LP1 SQ DMP1 DMP2)
(NEVER (PATTERN LEGALMOVE LP1 (LOC DMP1 BD))
 (FUNCTION LINE (LOC LP1 BD) (LOC DMP1 BD) (LOC DMP2 BD)))
(EXISTS (SQ2) (PATTERN MOBIL DMP1 SQ2))
(ACTIONNEW MOVE1 (LP1 SQ) (THREAT (FORK LP1 DMP1 DMP2)) (LIKELY 0)
 (SAFECOND (FUNCTION NEQ DP1 DMP1) (FUNCTION NEQ DP1 DMP2))
 (PLAN (SAFEMOVE LP1 SQ)
   (((DMP1 NIL) (KILL LP1 DMP2)) ((ANYBUT DMP1) (KILL LP1 DMP1))))))

THREAT
PINATTW
((LP1 SQ DMP1 DMP2)
(PATTERN PINATT LP1 SQ DMP1 DMP2)
(PATTERN ONLYPRO DMP1 DMP2)
(NEVER (EXISTS (SQ1) (PATTERN MOBSTAY DMP1 SQ1)
 (FUNCTION CANATTACK DMP1 SQ1 (LOC DMP2 BD))
 (NEVER (FUNCTION LINE SQ SQ1 (LOC DMP2 BD)))
 (NEVER (FUNCTION CANATTACK LP1 SQ SQ1))
 (NEVER (EXISTS (LP2) (PATTERN THRU LP2 LP1 SQ1)))))
(ACTIONNEW MOVE1 (LP1 SQ) (THREAT (FORKT (EXCH LP1 DMP1) (WIN DMP2))) (LIKELY 0)
 (SAFECOND (FUNCTION NEQ DP1 DMP1) (FUNCTION NEQ DP1 DMP2))
 (PLAN (SAFEMOVE LP1 SQ)
   (((DMP1 NIL) (KILL LP1 DMP2)) ((ANYBUT DMP1) (KILL LP1 DMP1))))))
```

```
THREAT
POSDECOY
((P1 SQ DSP2 SQ1 DK)
(PATTERN ONLYDIR DK SQ)  (PATTERN LEGALMOVE P1 SQ)
(NEVER (PATTERN MOBIL P1 SQ))
(OR
(EXISTS (P3 SQ2) (PATTERN DIR DK SQ2)
 (OR    (PATTERN THRU P3 P1 SQ2)
        ((PATTERN DIR P3 SQ2) (FUNCTION NEQ P1 P3))
        (EXISTS (P4) (PATTERN DSC P3 P4 (LOC P1 BD))
         (FUNCTION LINE (LOC P4 BD) (LOC P1 BD) SQ2))))
(IMPLIES (SQ2) ((FUNCTION CANATTACK DK SQ SQ2))
  (OR   (FUNCTION OCCBYCOL SQ2 (COLOR DK))
        (FUNCTION LINE SQ2 SQ (LOC DK BD))
        (FUNCTION EQUAL SQ2 (LOC DK BD))
        (EXISTS (P3) (OR ((PATTERN DIR P3 SQ2) (FUNCTION NEQ P3 P1))
                         (PATTERN THRU P3 P1 SQ2))))))
(OR
(EXISTS (LP1) (PATTERN DSC LP1 DSP2 SQ)
        (PATTERN LEGALMOVE DSP2 SQ1)
        (NEVER (FUNCTION LINE (LOC LP1 BD) SQ1 SQ)))
((OR (PATTERN LEGALMOVE DSP2 SQ1)
     (EXISTS (MP1) (PATTERN DIR MP1 (LOC P1 BD))
        (FUNCTION EQUAL MP1 DSP2) (FUNCTION EQUAL SQ1 (LOC P1 BD))))
  (FUNCTION NEQ DSP2 P1)
  (OR   (FUNCTION CANATTACK DSP2 SQ1 SQ)
        ((FUNCTION CANATTACK DSP2 SQ1 (LOC P1 BD))
          (FUNCTION LINE SQ1 (LOC P1 BD) SQ)
          (EXISTS (LP1) (FUNCTION EQUAL LP1 DSP2)))
        ((FUNCTION CANATTACK DSP2 SQ1 (LOC DK BD))
          (FUNCTION LINE SQ1 (LOC DK BD) SQ)
          (EXISTS (LP1) (FUNCTION EQUAL LP1 DSP2))))
  (OR   (FUNCTION SAFEP DSP2 SQ1)
        (PATTERN MOBIL DSP2 SQ1)
        ((PATTERN DIR (OCCUPANT SQ BD) SQ1)
          (NEVER (FUNCTION OVERPRO (COLOR DSP2) SQ1))
          (IMPLIES NIL ((FUNCTION CANATTACK DK SQ SQ1))
                 (NEVER (PATTERN ONLYPRO DSP2 SQ1))))
        ((PATTERN DIR DK SQ1)
          (NEVER (FUNCTION CANATTACK DK SQ SQ1))
          (NEVER (FUNCTION OVERPRO (COLOR DSP2) SQ1))))
  (IMPLIES NIL ((NEVER (EXISTS (P3)
    (OR (PATTERN DIR P3 SQ1) (PATTERN THRU P3 P1 SQ1) (PATTERN THRU P3 DSP2 SQ1))
    (FUNCTION NEQ P3 P1) (FUNCTION NEQ P3 DSP2))))
   (NEVER (FUNCTION CANATTACK DK SQ SQ1))))
(EXISTS (LP1) (PATTERN DSC LP1 DSP2 (LOC P1 BD))
        (FUNCTION LINE (LOC DSP2 BD) (LOC P1 BD) SQ)
        (PATTERN LEGALMOVE DSP2 SQ1)
        (NEVER (FUNCTION LINE (LOC LP1 BD) SQ1 SQ))))
(IMPLIES NIL ((PATTERN CHECKMOVE DSP2 SQ1))
        (NEVER (FUNCTION LINE SQ1 SQ (LOC DK BD))))
```

```
THREAT
DECOYK1
((P1 SQ P2 SQ1 DK)
(PATTERN POSDECOY P1 SQ P2 SQ1 DK)
(IMPLIES (SQ3) ((FUNCTIONE NEWFLIGHT SQ3 P2 SQ1 DK SQ))
  (NEVER (EXISTS (SQ2) (FUNCTIONE NEWFLIGHT SQ2 P2 SQ1 DK SQ3))))
(PLANNEW ((P1 SQ) (DK SQ) (CHKMOVE P2 SQ1))
 (THREAT (PLUS (WIN DK) (EXCHVAL P1 SQ)))  (LIKELY 0) (HINT 1)))

THREAT
DK-MOBIL
((P1 SQ DK SP1 SQ2)
(EXISTS (P2 SQ1) (PATTERN POSDECOY P1 SQ P2 SQ1 DK))
(NEVER (EXISTS (P2 SQ1) (PATTERN DECOYK1 P1 SQ P2 SQ1 DK)))
(OR     (PATTERN LEGALMOVE SP1 SQ2)
        ((PATTERN DIR SP1 (LOC P1 BD))
         (FUNCTION EQUAL SQ2 (LOC P1 BD))))
(FUNCTION NEQ P1 SP1)
(OR     (FUNCTION SAFEP SP1 SQ2)
        (PATTERN ONLYDIR (OCCUPANT SQ BD) SQ2)
        (PATTERN ONLYPRO DK SQ2)))

THREAT
DECOYK2
((P1 SQ P2 SQ1 DK)
(PATTERN POSDECOY P1 SQ P2 SQ1 DK)
(NEVER (EXISTS (AP1 SQ3) (PATTERN DECOYK1 P1 SQ AP1 SQ3 DK)))
(IMPLIES (SQ3) ((FUNCTIONE NEWFLIGHT SQ3 P2 SQ1 DK SQ))
 (OR (EXISTS (SP1 SQ2) (PATTERN DK-MOBIL P1 SQ DK SP1 SQ2)
        (IMPLIES NIL ((FUNCTION EQUAL SP1 P2)) (FUNCTION EQUAL SQ2 SQ1))
        (FUNCTIONE CA3GONE SP1 SQ2 SQ3 SQ (LOC P1 BD) (LOC DK BD)))
     (EXISTS (SP1 SQ2) (PATTERN DK-MOBIL P1 SQ DK SP1 SQ2)
        (NEVER (FUNCTION PAWN SP1))
        (IMPLIES NIL ((FUNCTION EQUAL SP1 P2)) (FUNCTION EQUAL SQ2 SQ1))
        (EXISTS (SQ4)  (FUNCTION IN-VUE SP1 SQ4 SQ3)
         (OR (FUNCTION SAFEP SP1 SQ4)
             (EXISTS (DP1) (UNIQUE (PATTERN DIR DP1 SQ4))
                (OR (FUNCTION EQUAL DP1 DK) (FUNCTION EQUAL DP1 (OCCUPANT SQ BD)))))
         (FUNCTIONE CA3GONE SP1 SQ2 SQ4 SQ (LOC P1 BD) (LOC DK BD))
         (FUNCTIONE CA3GONE SP1 SQ4 SQ3 SQ (LOC P1 BD) (LOC DK BD))))))
(PLANNEW ((P1 SQ) (DK SQ) (CHKMOVE P2 SQ1) (DK NIL) (ATTACKP DK))
 (LIKELY 1) (THREAT (PLUS (WIN DK) (EXCHVAL P1 SQ))) (HINT 1)))
```

```
THREAT
MATESQ
((P1 SQ DK)
(FUNCTIONE CANATACK P1 SQ (LOC OK BD))
(NEVER (PATTERN MATENOW1 P1 SQ DK))
(IMPLIES (SQ2) ((FUNCTIONE NEWFLIGHT SQ2 P1 SQ DK (LOC DK BD)))
 (OR
  (EXISTS (SQ1) (FUNCTION LINE SQ1 (LOC DK 8D) SQ2)
    (FUNCTION CANATTACK P1 SQ SQ1) (FUNCTION CANATTACK P1 SQ1 (LOC DK BD))
    (NEVER (EXISTS (SQ3) (FUNCTIONE NEWFLIGHT SQ3 P1 SQ1 DK SQ2))))
  ((NEVER (EXISTS (SQ3) (FUNCTIONE NEWFLIGHT SQ3 P1 SQ DK SQ2)))
   (EXISTS (P3 SQ1) (FUNCTION CANATTACK DK SQ2 SQ1)
    (OR (PATTERN THRU P3 P1 SQ1)
        ((PATTERN DIR P3 SQ1) (FUNCTION NEQ P1 P3))
        (EXISTS (DP1) (FUNCTION EQUAL DP1 (OCCUPANT SQ BD))
          (OR   ((PATTERN THRU P3 DP1 SQ1) (FUNCTION NEQ P1 P3))
                ((PATTERN THRU P3 P1 SQ) (FUNCTION LINE (LOC P3 BD) SQ SQ1))))
        (EXISTS (DP1) (PATTERN THRU P3 DP1 SQ1) (FUNCTION NEQ P3 P1)
                (EXISTS (P2 SQ4) (PATTERN ONLYPRO DP1 SQ4)
                  (PATTERN LEGALMOVE P2 SQ4)
                  (FUNCTION NEQ P2 P1) (FUNCTION NEQ P2 P3)
                  (EXISTS (SQ3) (FUNCTION CANATTACK P2 SQ4 SQ3)
                        (FUNCTION CANATTACK DK SQ2 SQ3)))))))))

THREAT
MATE2
((P1 SQ SQ1 DK)
(PATTERN MATESQ P1 SQ DK)
(PATTERN MOBSTAY P1 SQ)
(IMPLIES (DP1) ((FUNCTION EQUAL DP1 (OCCUPANT SQ 8D)))
        (NEVER (FUNCTION SAFEP DP1 SQ)))
(NEVER (PATTERN MATE P1 SQ))
(FUNCTION CANMOVE P1 SQ SQ1)
(IMPLIES NIL ((FUNCTION EQUAL SQ1 (LOC DK BD)))
  (EXISTS (P2) (PATTERN DSC P2 P1 SQ1)))
(IMPLIES (SQ2) ((FUNCTIONE NEWFLIGHT SQ2 P1 SQ DK (LOC DK BD)))
 (OR    (FUNCTION CANATTACK P1 SQ1 SQ2)
        ((FUNCTION CANATTACK P1 SQ1 (LOC DK BD))
         (FUNCTION LINE SQ1 (LOC DK BD) SQ2))))
(PLANNEW ((P1 SQ) (DK NIL) (MATEMOVE P1 SQ1))
  (THREAT (WIN DK)) (LIKELY 0)))

THREAT
MATEND2D
((P2 SQ DK)
(PATTERN MATESQ P2 SQ DK)
(NEVER (EXISTS (SQ1) (PATTERN MATE2 P2 SQ SQ1 DK)))
(FUNCTION OCCBYCOL SQ (COLOR P2))
(PATTERN DSC P2 (OCCUPANT SQ BD) (LOC OK BD))
(ACTIONNEW ATTACK ((COLOR P2) (LOC DK BD)) (THREAT (WIN DK))
 (COND  (FUNCTION EQUAL P1 (DCCUPANT SQ BD)))
 (SAFECOND (FUNCTION NEQ DP1 DK))))
```

```
THREAT
MATENO2
((P1 SQ DK)
(PATTERN MATESQ P1 SQ DK)
(NEVER (EXISTS (SQ1) (PATTERN MATE2 P1 SQ SQ1 DK)))
(NEVER (PATTERN MATENO2D P1 SQ DK))
(ACTIONNEW MOVE (P1 SQ) (THREAT (WIN DK)) (LIKELY 0) (PLAN (CHKMOVE P1 SQ))
 (CONO  (NEVER  (FUNCTIDN CANATTACK P1 SQ2 (LOC DK BD))))
 (SAFECOND (FUNCTION NEQ DP1 DK))))

THREAT
MATE2U
((P1 SQ SQ2 DK)
(PATTERN MATE2 P1 SQ SQ2 DK)
(NEVER (FUNCTION SAFEP P1 SQ2))
(ACTIONNEW SAFE (P1 SQ2) (THREAT (WIN DK)) (LIKELY 0)
  (PLAN NIL (CHKMOVE P1 SQ) (DK NIL) (MATEMOVE P1 SQ2))
  (DECOYCOND (NEVER (FUNCTION CANATTACK DP1 SQ1 SQ2)))
  (SAFECOND (FUNCTION NEQ DP1 DK))))

THREAT
TRAPMATE
((P1 SQ1 P2 SQ2 DK)
(NEVER (PATTERN TRAPPED DK))
(PATTERN CHECKMOVE P2 SQ2)
(PATTERN MOBSTAY P2 SQ2)
(NEVER (PATTERN MATE P2 SQ2))
(NEVER (EXISTS (DP1) (PATTERN ENPRISE DP1 P2)))
(PATTERN MOBSTAY P1 SQ1)
(FUNCTION NEQ P1 P2)
(FUNCTION NEQ SQ1 SQ2)
(IMPLIES (SQ3) ((FUNCTIONE NEWFLIGHT SQ3 P2 SQ2 DK (LOC DK BD)))
        (FUNCTION CANATTACK P1 SQ1 SQ3)
        (NEVER (EXISTS (DP1 SQ) (FUNCTION LINE SQ1 SQ SQ3)
          (PATTERN MOBIL DP1 SQ) (NEVER (PATTERN ONLYPRO DP1 SQ)))))
(OR     (NEVER (PATTERN MATESQ P2 SQ2 DK))
        (FUNCTION OCCBYCOL SQ1 (COLOR DK))))

THREAT
TM1
((P1 SQ1 P2 SQ2 DK)
(PATTERN TRAPMATE P1 SQ1 P2 SQ2 DK)
(OR     (FUNCTION CANATTACK P1 SQ1 (LOC DK BD))
        ((NEVER (EXISTS (SQ) (PATTERN DIR DK SQ)
          (NEVER (FUNCTION OCCBYCOL SQ (COLOR DK)))
          (PATTERN ONLYPRO P1 SQ) (NEVER (FUNCTION CANATTACK P1 SQ1 SQ))))
         (IMPLIES NIL ((PATTERN DIR P1 SQ2)) (FUNCTION CANATTACK P1 SQ1 SQ2))))
(PLANNEW ((P1 SQ1) NIL (MATEMOVE P2 SQ2))
        (THREAT (WIN DK)) (LIKELY 0)))

THREAT
TM2
((P1 SQ1 P2 SQ2 DK)
(PATTERN TRAPMATE P1 SQ1 P2 SQ2 DK)
(NEVER (PATTERN TM1 P1 SQ1 P2 SQ2 DK))
(PLANNEW ((P1 SQ1) NIL (MATEMOVE P2 SQ2))
        (THREAT (WIN DK)) (LIKELY 1)))
```

```
THREAT
TWOCHECK
((P1 P2 DK SQ1 SQ2)
(PATTERN CHECKMOVE P1 SQ1)
(PATTERN MOBSTAY P1 SQ1)
(NEVER (PATTERN MATE P1 SQ1))
(PATTERN LEGALMOVE P2 SQ2)
(FUNCTION NEQ SQ1 SQ2) (FUNCTION NEQ P1 P2)
(NEVER (FUNCTION LINE (LOC P2 BD) SQ1 SQ2))
(IMPLIES (SQ) ((FUNCTIONE NEWFLIGHT SQ P1 SQ1 DK (LOC DK BD)))
(FUNCTION CANATTACK P2 SQ2 SQ)
(OR     ((OR    (PATTERN MOBSTAY P2 SQ2)
                (EXISTS (DLP1) (PATTERN ONLYPRO DLP1 SQ2)
                        (FUNCTION LINE SQ2 SQ (LOC DLP1 BD))))
         (IMPLIES NIL ((NEVER (EXISTS (P3)
                (OR (PATTERN DIR P3 SQ2) (PATTERN THRU P3 P2 SQ2)))))
          (OR (FUNCTION CANATTACK P1 SQ1 SQ2)
              (NEVER (FUNCTION CANATTACK DK SQ SQ2)))))
        ((PATTERN ONLYPRO DK SQ2)
         (OR (FUNCTION CANATTACK P1 SQ1 SQ2)
             (NEVER (FUNCTION CANATTACK DK SQ SQ2))))))

THREAT
TWOC1
((P1 P2 DK SQ1 SQ2)
(PATTERN TWOCHECK P1 P2 DK SQ1 SQ2)
(IMPLIES (SQ) ((FUNCTIONE NEWFLIGHT SQ P1 SQ1 DK (LOC DK BD)))
  (IMPLIES (SQ3) ((FUNCTIONE NEWFLIGHT SQ3 P1 SQ1 DK SQ))
        (OR (FUNCTION CANATTACK P2 SQ2 SQ3)
            ((FUNCTION LINE SQ2 (LOC DK BD) SQ3)
             (FUNCTION CANATTACK P2 SQ2 (LOC DK BD))
             (NEVER (FUNCTION PAWN P2))))))
(PLANNEW ((P1 SQ1) (DK NIL) (MATEMOVE P2 SQ2))
 (THREAT (PLUS (EXCHVAL P1 SQ1) (WIN DK))) (LIKELY 0)))

THREAT
THREEC
((P1 P2 DK SQ1 SQ2)
(PATTERN TWOCHECK P1 P2 DK SQ1 SQ2)
(NEVER (PATTERN TWOC1 P1 P2 DK SQ1 SQ2))
(IMPLIES (SQ) ((FUNCTIONE NEWFLIGHT SQ P1 SQ1 DK (LOC DK BD)))
 (IMPLIES (SQ3) ((FUNCTIONE NEWFLIGHT SQ3 P1 SQ1 DK SQ))
  (OR   (FUNCTION CANATTACK P2 SQ2 SQ3)
        (EXISTS (P3 SQ4) (PATTERN MOBSTAY P3 SQ4)
          (FUNCTION CANATTACK P3 SQ4 SQ)
          (FUNCTION NEQ P3 P1) (FUNCTION NEQ P3 P2)
          (FUNCTION NEQ SQ4 SQ1) (FUNCTION NEQ SQ4 SQ2)
          (NEVER (FUNCTION LINE (LOC P3 BD) SQ1 SQ4))
          (NEVER (FUNCTION LINE (LOC P3 BD) SQ2 SQ4))))))
(PLANNEW ((P1 SQ1) (DK NIL) (CHKMOVE P2 SQ2) (DK NIL) (ATTACKP DK))
 (THREAT (PLUS (EXCHVAL P1 SQ1) (WIN DK))) (LIKELY 0)))
```

```
THREAT
DMATESQ
((P1 SQ SQ1 P3 SQ2 DP2)
(EXISTS (DK) (PATTERN MATESQ P1 SQ DK))
(NEVER (PATTERN DIR P1 SQ))
(NEVER (EXISTS (P2) (PATTERN THRU P1 P2 SQ)))
(PATTERN LEGALMOVE P1 SQ1)
(FUNCTION CANATTACK P1 SQ1 (LOC P3 BD))
(FUNCTION NEQ P1 P3)
(FUNCTION LINE SQ1 (LOC P3 BD) SQ)
(NEVER (FUNCTION LINE (LOC P1 BD) SQ1 SQ))
(OR     (NEVER (PATTERN DIR P3 SQ1))
        (FUNCTION OVERPRO (COMPCOLOR P3) SQ1))
(PATTERN LEGALMOVE P3 SQ2)
(NEVER (PATTERN MOBSTAY P3 SQ2))
(OR     ((PATTERN MATE P3 SQ2) (EXISTS (DK) (FUNCTION EQUAL DK DP2)))
        (FUNCTION EQUAL DP2 (OCCUPANT SQ2 BD)))
(NEVER (FUNCTION LINE SQ1 SQ2 SQ))
(NEVER (EXISTS (DP1) (PATTERN THRU DP1 P3 SQ1) (FUNCTION NEQ DP1 DP2)
        (ASGREAT (VALUE P1) (VALUE DP1)) (NEVER (PATTERN ONLYPRO DP1 SQ2))))
(OR     (PATTERN MOBSTAY P1 SQ1)
        ((NEVER (FUNCTION OVERPRO (COLOR P1) SQ1))
         (NEVER (EXISTS (DP1) (PATTERN THRU DP1 P3 SQ1) (FUNCTION NEQ DP1 DP2)))
         (PATTERN DIR DP2 SQ1))))

THREAT
DMWIN1
((P1 SQ SQ1 P3 SQ2 DP2 DK)
(PATTERN DMATESQ P1 SQ SQ1 P3 SQ2 DP2)
(FUNCTION LINE SQ1 SQ (LOC DK BD))
(FUNCTION CANATTACK P1 (LOC P3 BD) (LOC DK BD))
(PLANNEW  ((P3 SQ2) NIL (CHKMOVE P1 SQ1)) (LIKELY 0)
  (THREAT (FORKT (PLUS (EXCH P3 DP2) (WIN DK)) (WIN DP2))))
(RETURN))

THREAT
DMWIN2
((P1 SQ SQ1 P3 SQ2 DP2 DK DP1)
(PATTERN DMATESQ P1 SQ SQ1 P3 SQ2 DP2)
(FUNCTION LINE SQ1 SQ (LOC DK BD))
(PATTERN PROTECTS DP1 DP2 SQ2 P3)
(FUNCTION LINE SQ1 (LOC DP1 BD) (LOC DK BD))
(FUNCTION CANATTACK P1 (LOC P3 BD) (LOC DP1 BD))
(FUNCTION CANATTACK P1 (LOC DP1 BD) (LOC DK BD))
(PLANNEW  ((P3 SQ2) (NIL SQ2) (SAFEMOVE P1 SQ1)) (LIKELY 0)
  (THREAT (FORKT (PLUS (EXCH P3 DP2) (WIN DK)) (WIN DP2))))
(RETURN))

THREAT
DMSAFE
((P1 SQ SQ1 P3 SQ2 DP2 DK)
(PATTERN DMATESQ P1 SQ SQ1 P3 SQ2 DP2)
(FUNCTION SAFEP P1 SQ)
(OR     (FUNCTION CANATTACK P1 (LOC P3 BD) SQ)
        (EXISTS (DP1) (PATTERN PROTECTS DP1 DP2 SQ2 P3)
                (FUNCTION LINE SQ1 (LOC DP1 BD) SQ)
                (FUNCTION CANATTACK P1 (LOC P3 BD) (LOC DP1 BD))
                (FUNCTION CANATTACK P1 (LOC DP1 BD) SQ)))
(PLANNEW  ((P3 SQ2) (NIL SQ2) (SAFEMOVE P1 SQ1)) (LIKELY 1)
  (THREAT (PLUS (WIN DK) (EXCHVAL P3 SQ2)))))
```

```
THREAT
CHKFORCE
((P1 DP2 P3 SQ DK)
(PATTERN ENPRIS P1 DP2)
(PATTERN CHECKMOVE P3 SQ)
(NEVER (PATTERN MATE P3 SQ))
(IMPLIES NIL ((FUNCTION LINE (LOC P1 BD) SQ (LOC DP2 BD)))
        (FUNCTION EQUAL P1 P3))
(IMPLIES NIL ((FUNCTION EQUAL P1 P3))
        (FUNCTION LINE (LOC P1 BD) SQ (LOC DP2 BD)))
(IMPLIES NIL ((PATTERN DIR P3 (LOC DP2 BD)))
        (OR (FUNCTION CANATTACK P3 SQ (LOC DP2 BD))
            (FUNCTION OVERPRO (COLOR DP2) (LOC DP2 BD))))
(IMPLIES (SQ2) ((FUNCTIONE NEWFLIGHT SQ2 P3 SQ DK (LOC DK BD)))
        (FUNCTION CANATTACK P1 (LOC DP2 BD) SQ2))
(IMPLIES NIL ((PATTERN DIR DP2 SQ))
        (NEVER (FUNCTION ASVAL P3 DP2)))
(ACTIONNEW MOVE1 (P3 SQ) (THREAT (EXCH P1 DP2)) (LIKELY 0)
  (SAFECOND (FUNCTION NEQ DP1 DP2))
  (SAFECOND1 (FUNCTION NEQ P2 P1))
  (PLAN (CHKMOVE P3 SQ) (DK NIL) (KILL P1 DP2))))

THREAT
FORKPK
((SP1 SQ DPN1 DK)
(PATTERN CHECKMOVE SP1 SQ)
(PATTERN MOBSTAY SP1 SQ)
(FUNCTION CANATTACK SP1 SQ (LOC DPN1 BD))
(FUNCTION SAFEP SP1 (LOC DPN1 BD))
(NEVER (PATTERN DIR SP1 (LOC DPN1 BD)))
(NEVER (EXISTS (SQ2) (PATTERN DIR DK SQ2)
        (FUNCTION CANATTACK DK SQ2 (LOC DPN1 BD))
        (NEVER (FUNCTION CANATTACK SP1 SQ SQ2))
        (OR (PATTERN LEGALMOVE DK SQ2) (PATTERN ONLYPRO SP1 SQ2))))
(PLANNEW ((SP1 SQ)) (LIKELY 0)
 (THREAT (PLUS (EXCHVAL SP1 SQ) (EXCHVAL SP1 (LOC DPN1 BD))))))

THREAT
DECOYPRO
((P1 SQ2 DSP1 DP1)
(PATTERN PROTECTS DP1 DSP1 SQ2 P1)
(NEVER (PATTERN MOBSTAY P1 SQ2))
(ACTIONNEW DECOY (P1 SQ2 DP1)
  (THREAT (WIN DSP1)) (LIKELY 0)
  (PLAN (ANYBUT DSP1) (SAFEMOVE P1 SQ2))))
```

```
THREAT
ATTPRO
((P1 P2 SQ1 SQ2 DSP1 DP1)
(PATTERN PROTECTS DP1 DSP1 SQ2 P1)
(NEVER (PATTERN MOBSTAY P1 SQ2))
(PATTERN LEGALMOVE P2 SQ1)
(FUNCTION CANATTACK P2 SQ1 (LOC DP1 BD))
(FUNCTION SAFEP P2 (LOC DP1 BD))
(FUNCTION NEQ P2 P1)
(NEVER (FUNCTION LINE (LOC P1 BD) SQ1 SQ2))
(OR     (PATTERN MOBSTAY P2 SQ1)
        ((PATTERN LEGALMOVE P2 SQ1) (PATTERN ONLYOIR DP1 SQ1)
          (GREAT (VALUE DSP1) (VALUE P2)))))

THREAT
ATTPRO1
((P1 P2 SQ1 SQ2 DSP1 DP1)
(PATTERN ATTPRO P1 P2 SQ1 SQ2 DSP1 DP1)
(OR  (PATTERN CHECKMOVE P2 SQ1)
   (NEVER (EXISTS (SQ) (PATTERN LEGALMOVE DP1 SQ)
        (FUNCTION CANATTACK DP1 SQ SQ2)
        (OR (PATTERN MOBSTAY DP1 SQ)
            ((PATTERN ONLYPRO P2 SQ) (NEVER (FUNCTION CANATTACK P2 SQ1 SQ)))))))
(PLANNEW ((P2 SQ1) (((DP1 NIL) (KILL P1 DSP1)) ((ANYBUT DP1) (KILL P2 DP1))))
  (THREAT (FORKT (WIN DSP1) (EXCH P2 DP1))) (LIKELY 0)))

THREAT
ATTPRO2
((P1 P2 SQ SQ1 SQ2 SQ3 DSP1 DP1)
(PATTERN ATTPRO P1 P2 SQ1 SQ2 DSP1 DP1)
(UNIQUE (PATTERN LEGALMOVE DP1 SQ)
 (FUNCTION CANATTACK DP1 SQ SQ2)
 (OR (PATTERN MOBSTAY DP1 SQ)
     ((PATTERN ONLYPRO P2 SQ) (NEVER (FUNCTION CANATTACK P2 SQ1 SQ)))))
(FUNCTION CANMOVE P2 SQ1 SQ3)
(FUNCTION CANATTACK P2 SQ3 SQ)
(FUNCTION CANATTACK P2 SQ3 SQ2)
(NEVER (FUNCTION LINE (LOC P1 BD) SQ3 SQ2))
(OR     (FUNCTION SAFEP P2 SQ3)
        ((PATTERN ONLYPRO DP1 SQ3) (NEVER (FUNCTION CANATTACK DP1 SQ SQ3))))
(PLANNEW ((P2 SQ1) (((DP1 SQ) (SAFEMOVE P2 SQ3)
        (((ANYBUT OSP1) (KILL P1 DSP1)) ((ANYBUT DSP1) (KILL P2 DSP1))
                ((ANYBUT DP1) (KILL P2 DP1))))
   ((ANYBUT DP1) (KILL P2 OP1))
   (NIL (KILL P1 DSP1))))
 (THREAT (FORKT (WIN DSP1) (EXCH P2 DP1))) (LIKELY 0)))
```

```
THREAT
SACDSC
((P1 P2 DMP1 DP2)
(PATTERN LEGALMOVE P1 (LOC DP2 BD))
(PATTERN DSC P2 P1 (LOC DMP1 BD))
(PATTERN DIR DP2 (LOC DMP1 BD))
(NEVER (FUNCTION LINE (LOC P2 BD) (LOC DP2 BD) (LOC DMP1 BD)))
(ASGREAT (PLUS (VALUE DMP1) (VALUE DP2)) (VALUE P1),
(PLANNEW ((P1 (LOC DP2 BD)) NIL (KILL P2 DMP1))
        (THREAT (PLUS (WIN DMP1) (EXCH P1 DP2))) (LIKELY 0)))

THREAT
SACPIN
((P1 LP2 SQ DP1 DMP1 DK)
(PATTERN LEGALMOVE P1 (LOC DP1 BD))
(PATTERN ONLYDIR DK (LOC DP1 BD))
(DR     (PATTERN MOBSTAY LP2 SQ)
        ((PATTERN DSC LP2 P1 SQ)  (FUNCTION SAFEP LP2 SQ)))
(FUNCTION NEQ P1 LP2)
(FUNCTION LINE SQ (LOC DMP1 BD) (LOC DP1 BD))
(FUNCTION CANATTACK LP2 (LOC DMP1 BD) (LOC DP1 BD))
(FUNCTION CANATTACK LP2 SQ (LOC DMP1 BD))
(ASGREAT (PLUS (VALUE DMP1) (VALUE DP1)) (VALUE P1))
(PLANNEW ((P1 (LOC DP1 BD)) (DK (LOC DP1 BD)) (SAFEMOVE LP2 SQ))
        (THREAT (PLUS (EXCH P1 DP1) (EXCH LP2 DMP1))) (LIKELY 0)))

THREAT
SACDECSUP
((P1 P2 DP1 DMP1 DP2)
(PATTERN LEGALMOVE P1 (LOC DP1 BD))
(PATTERN DSC P2 P1 (LOC DMP1 BD))
(PATTERN ONLYDIR DP2 (LOC DP1 BD))
(PATTERN DIR DP2 (LOC DMP1 BD))
(NEVER (FUNCTION CANATTACK DP2 (LOC DP1 BD) (LOC DMP1 BD)))
(ASGREAT (PLUS (VALUE DMP1) (VALUE DP1)) (VALUE P1))
(PLANNEW ((P1 (LOC DP1 BD)) (DP2 (LOC DP1 BD)) (KILL P2 DMP1))
        (THREAT (PLUS (WIN DMP1) (EXCH P1 DP1))) (LIKELY 1)))
```

```
THREAT
DSCDECOY
((P1 P2 SQ1 DP1 DP2 DP3 DP4)
(PATTERN LEGALMOVE P1 (LOC DP1 BD))
(NEVER (PATTERN ENPRIS P1 DP1))
(NEVER (FUNCTION OVERPRO (COLOR P1) (LOC DP1 BD)))
(UNIQUE (PATTERN DIR DP4 (LOC DP1 BD))
  (OR   (FUNCTION SAFENB DP4 (LOC DP1 BD))
        (FUNCTION SAFEP DP1 (LOC DP1 BD))))
(PATTERN THRU P2 P1 SQ1)
(NEVER (PATTERN DIR P2 (LOC DP1 BD)))
(NEVER (PATTERN OTHRU P2 P1 (LOC DP1 BD)))
(FUNCTION CANATTACK P2 SQ1 (LOC DP2 BD))
(FUNCTION CANATTACK P2 SQ1 (LOC DP3 BD))
(FUNCTION NEQ DP2 DP3) (FUNCTION NEQ DP1 DP2)
(FUNCTION NEQ DP1 DP3) (FUNCTION NEQ DP4 DP3) (FUNCTION NEQ DP4 DP2)
(FUNCTION SAFEP P2 (LOC DP3 BD))
(OR     (PATTERN DIR DP2 (LOC DP1 BD))
        (PATTERN OTHRU DP2 DP4 (LOC DP1 BD)))
(IMPLIES NIL ((PATTERN DIR DP2 SQ1))
        (NEVER (FUNCTION CANATTACK DP2 SQ1 (LOC DP1 BD))))
(IMPLIES (DP5) ((PATTERN DIR DP5 SQ1))
  (OR   (FUNCTION EQUAL DP5 DP4)
        (FUNCTION EQUAL DP5 DP2)
        (FUNCTION EQUAL DP5 DP1)))
(GREAT (VALUE DP3) (EXCHLOSS P1 (LOC DP1 BD)))
(GREAT (PLUS (VALUE DP1) (VALUE DP4)) (PLUS (VALUE P1) (VALUE P2)))
(PLANNEW ((P1 (LOC DP1 BD)) (DP4 (LOC DP1 BD)) (P2 SQ1))
(LIKELY 0)
(THREAT (FORKT (WIN DP1) (FORKT (PLUS (WIN DP1) (LESS (WIN DP3) (WIN P1)))
        (LESS (PLUS (WIN DP4) (WIN DP1)) (PLUS (WIN P1) (WIN P2)))))))
```

```
ATTACK
DSCNT
((LP1 P2 SQ SQ3)
(PATTERN DSC LP1 P2 SQ3)
(GREAT (VALUE P2) THREAT)
(PATTERN MOBIL P2 SQ)
(PLANNEW ((P2 SQ)) (THREAT (EXCH LP1 (OCCUPANT SQ3 BD))) (LIKELY 0)))

ATTACK
DSCA
((LP1 P2 SQ SQ3 DP1)
(FUNCTION EQUAL DP1 (OCCUPANT SQ3 BD))
(PATTERN DSCTHR P2 LP1 DP1 SQ)
(ASGREAT THREAT (VALUE P2)))

ATTACK
DSC1A
((LP1 P2 SQ SQ3 DP1)
(PATTERN DSCA LP1 P2 SQ SQ3 DP1)
(EXISTS (DP2) (FUNCTION EQUAL (LOC DP2 BD) SQ)
(PLANNEW ((P2 SQ)) (THREAT (PLUS (EXCH P2 DP2) (EXCH LP1 DP1))) (LIKELY 0))))

ATTACK
DSC2A
((LP1 P2 SQ SQ3 DP1)
(PATTERN DSCA LP1 P2 SQ SQ3 DP1)
(EXISTS (DMP1) (FUNCTION CANATTACK P2 SQ (LOC DMP1 BD))
                (FUNCTION SAFEP P2 (LOC DMP1 BD))
                (NEVER (FUNCTION EXCHANGE P2 (LOC DMP1 BD))))
(PLANNEW ((P2 SQ)) (THREAT (PLUS 5 (EXCH LP1 DP1))) (LIKELY 0)))

ATTACK
DSC3A
((LP1 P2 SQ SQ3 DP1)
(PATTERN DSCA LP1 P2 SQ SQ3 DP1)
(EXISTS (DK) (FUNCTION CANATTACK P2 SQ (LOC DK BD)))
(PLANNEW ((P2 SQ)) (THREAT (PLUS 10 (EXCH LP1 DP1))) (LIKELY 0)))

ATTACK
DSCBKI
((LP1 P2 SQ SQ3 DP1)
(FUNCTION EQUAL DP1 (OCCUPANT SQ3 BD))
(PATTERN DSCTHR P2 LP1 DP1 SQ)
(ASGREAT THREAT (VALUE P2))
(EXISTS (DLP1 SQ2)
  (PATTERN LEGALMOVE DLP1 SQ2)
  (FUNCTION LINE (LOC LP1 BD) SQ2 SQ3)
  (FUNCTION LINE (LOC DLP1 BD) SQ SQ2))
(PLANNEW ((P2 SQ)) (THREAT (PLUS 3 (EXCH LP1 DP1))) (LIKELY 0)))
```

```
ATTACK
DSCSAFE
((LP1 P2 SQ SQ3 DP1)
(FUNCTION EQUAL DP1 (OCCUPANT SQ3 BO))
(PATTERN DSC LP1 P2 SQ3)
(ASGREAT THREAT (VALUE P2))
(NEVER (PATTERN DSC2A LP1 P2 SQ SQ3 DP1))
(NEVER (PATTERN DSCBKI LP1 P2 SQ SQ3 DP1))
(PATTERN MOBIL P2 SQ)
(PLANNEW ((P2 SQ)) (THREAT (EXCH LP1 DP1)) (LIKELY 0)))

ATTACK
DSCUSAFE
((LP1 P2 SQ SQ3 DP1)
(FUNCTION EQUAL DP1 (OCCUPANT SQ3 BD))
(PATTERN DSCTHR P2 LP1 DP1 SQ)
(ASGREAT THREAT (VALUE P2))
(NEVER (PATTERN DSC2A LP1 P2 SQ SQ3 DP1))
(NEVER (PATTERN DSCBKI LP1 P2 SQ SQ3 DP1))
(NOT (EXISTS (SQ2) (PATTERN MOBIL P2 SQ2)))
(PLANNEW ((P2 SQ)) (THREAT (PLUS (EXCHVAL P2 SQ) (EXCH LP1 DP1))) (LIKELY 0)))
```

```
ATTACK
DIRECT
((P1 SQ SQ3)
(FUNCTIONE CANATACK P1 SQ SQ3)
(NEVER (PATTERN DIR P1 SQ3))
(IMPLIES (P2) ((PATTERN DSC P1 P2 SQ3))
  (NEVER (FUNCTION LINE (LOC P1 BD) SQ SQ3)))
(IMPLIES NIL
  ((FUNCTION NEQ 0 (OCCUPANT SQ3 BD))
   (FUNCTION OVERPRO (COLOR P1) SQ3))
 (GREAT (VALUE (OCCUPANT SQ3 BD)) (VALUE P1))))

ATTACK
DIRECT1
((P1 SQ SQ3)
(PATTERN DIRECT P1 SQ SQ3)
(PATTERN MOBSTAY P1 SQ)
(CHECKCOND COND)
(ACTION MOVE (P1 SQ) (THREAT (EXCHVAL P1 SQ3)) (COND)
  (PLAN (SAFEMOVE P1 SQ)) (LIKELY 0)))

ATTACK
DIRECT2
((P1 SQ SQ3)
(PATTERN DIRECT P1 SQ SQ3)
(NEVER (EXISTS (SQ1) (PATTERN DIRECT P1 SQ1 SQ3) (PATTERN MOBSTAY P1 SQ1)))
(IMPLIES NIL ((PATTERN LEGALMOVE P1 SQ))
        (CHECKCOND COND))
(ACTION MOVE (P1 SQ) (THREAT (EXCHVAL P1 SQ3)) (LIKELY 0)
        (PLAN (SAFEMOVE P1 SQ))
        (COND (NEVER (FUNCTION CANATTACK P1 SQ2 SQ3)))))

ATTACK
DIRECTDWIN
((P1 SQ SQ3 P2 SQ1)
(PATTERN DIRECT P1 SQ SQ3)
(NEVER (EXISTS (SQ2) (PATTERN DIRECT P1 SQ2 SQ3) (PATTERN MOBSTAY P1 SQ2)))
(PATTERN THRU P1 P2 SQ)
(PATTERN DIRECT P2 SQ1 SQ3)
(PATTERN LEGALMOVE P2 SQ1)
(NEVER (FUNCTION LINE (LOC P1 BD) SQ1 SQ))
(GREAT (EXCHVAL P1 SQ3) (EXCHLOSS P2 SQ1))
(OR     (FUNCTION SAFEP P1 SQ)
        (PATTERN DIR (OCCUPANT SQ1 BD) SQ)
        (EXISTS (DP1) (PATTERN DNLYPRO DP1 SQ1) (PATTERN DIR DP1 SQ)
          (NEVER (FUNCTION CANATTACK DP1 SQ1 SQ))))
(PLANNEW ((P2 SQ1) NIL (SAFEMOVE P1 SQ))
  (THREAT (PLUS (EXCHVAL P1 SQ1) (EXCHVAL P1 SQ3))) (LIKELY 0)))

ATTACK
DECBLKR
((LP1 SQ3 DP1)
(PATTERN THRU LP1 DP1 SQ3)
(ACTION DECOY (LP1 SQ3 DP1) (THREAT (EXCHVAL LP1 SQ3)) (LIKELY 0)))
```

```
ATTACK
BLKER
((LP1 SQ DP1 SQ3)
(PATTERN LEGALMOVE LP1 SQ)
(FUNCTION LINE SQ (LOC DP1 BD) SQ3)
(FUNCTION CANATTACK LP1 SQ (LOC DP1 BD))
(FUNCTION CANATTACK LP1 (LOC DP1 BD) SQ3))

ATTACK
DECBLKP
((LP1 SQ DPN1 SQ3)
(PATTERN BLKER LP1 SQ DPN1 SQ3)
(OR     (PATTERN MOBSTAY LP1 SQ)
        ((PATTERN DIR DPN1 SQ) (NEVER (FUNCTION OVERPRO (COLOR LP1) SQ))))
(ACTION DECOY (LP1 SQ DPN1) (THREAT (PLUS (WIN (OCCUPANT SQ BD)) (EXCHVAL LP1 SQ3)))
  (LIKELY 0) (PLAN (DPN1 NIL) (SAFEMOVE LP1 SQ))
  (DECOYCOND (NEVER (FUNCTION LINE SQ SQ1 SQ3)))))

ATTACK
DBLK
((LP1 SQ DP1 SQ3 DP2 P1)
(PATTERN BLKER LP1 SQ DP1 SQ3)
(PATTERN ONLYDIR DP1 (LOC DP2 BD))
(PATTERN LEGALMOVE P1 (LOC DP2 BD))  (FUNCTION NEQ P1 LP1)
(GREAT (EXCHVAL LP1 SQ3) (EXCHLOSS P1 (LOC DP2 BD))))

ATTACK
DBLK1
((LP1 SQ DP1 SQ3 DP2 P1 P3)
(PATTERN DBLK LP1 SQ DP1 SQ3 DP2 P1)
(PATTERN THRU P3 DP2 SQ3)
(FUNCTION NEQ P3 P1) (FUNCTION NEQ P3 LP1)
(GREAT (EXCHVAL P3 SQ3) (EXCHLOSS P1 (LOC DP2 BD)))
(PLANNEW ((P1 (LOC DP2 BD)) (DP1 (LOC DP2 BD)) (SAFEMOVE LP1 SQ))
  (LIKELY 0)
 (THREAT (PLUS (EXCH P1 DP2) (PLUS (WIN (OCCUPANT SQ BD)) (EXCHVAL LP1 SQ3))))))

ATTACK
DBLK2
((LP1 SQ DP1 SQ3 DP2 P1 P3 SQ2)
(PATTERN DBLK LP1 SQ DP1 SQ3 DP2 P1)
(NEVER (PATTERN DBLK1 LP1 SQ DP1 SQ3 DP2 P1 P3))
(OR     (PATTERN MOBIL P3 SQ2)
        ((PATTERN DIR P3 (LOC P1 BD)) (FUNCTION SAFEP P3 (LOC P1 BD))
          (FUNCTION EQUAL SQ2 (LOC P1 BD))))
(FUNCTION NEQ P3 P1) (FUNCTION NEQ P3 LP1)
(FUNCTION CANATTACK P3 SQ2 (LOC DP2 BD))
(FUNCTION CANATTACK P3 (LOC DP2 BD) SQ3)
(GREAT (EXCHVAL P3 SQ3) (EXCHLOSS P1 (LOC DP2 BD)))
(PLANNEW ((P1 (LOC DP2 BD)) (DP1 (LOC DP2 BD)) (SAFEMOVE LP1 SQ)
           (DP1 NIL) (SAFEMOVE P3 SQ2))
  (LIKELY 0)
 (THREAT (PLUS (EXCH P1 DP2) (PLUS (WIN (OCCUPANT SQ BD)) (EXCHVAL LP1 SQ3))))))
```

```
MOVE
BACK1
((P1 SQ2 SQ3)
(NEVER (PATTERN DIR P1 SQ3))
(NEVER (PATTERN LEGALMOVE P1 SQ3))
(NEVER (FUNCTION OVERPRO (COLOR P1) SQ3))
(PATTERN LEGALMOVE P1 SQ2)
(FUNCTION CANMOVE P1 SQ2 SQ3)
(CHECKCOND COND))

MOVE2
M2COND
((P1 SQ3)
(NEVER (EXISTS (SQ2) (PATTERN BACK1 P1 SQ2 SQ3)))
(RETURN))

MOVE2
M2WIN
((P1 SQ2 SQ3 DK)
(PATTERN BACK1 P1 SQ2 SQ3)
(FUNCTION CANATTACK P1 SQ3 (LOC DK BD))
(PATTERN MOBSTAY P1 SQ2)
(FUNCTION SAFEP P1 SQ3)
(CHECKCOND COND)
(NEVER (EXISTS (SQ1) (FUNCTIONE NEWFLIGHT SQ1 P1 SQ3 DK (LOC DK BD))))
(IMPLIES (DP1 SQ1) ((PATTERN MOBSTAY DP1 SQ1)
  (OR (FUNCTION LINE SQ2 SQ1 SQ3) (FUNCTION LINE SQ3 SQ1 (LOC DK BD))))
        (PATTERN ONLYPRO DP1 SQ1))
(IMPLIES (P2) ((EXISTS (DP1) (PATTERN DIR DP1 SQ3)) (PATTERN ONLYDIR P2 SQ3))
  (NEVER (EXISTS (DP1) (PATTERN MOBIL DP1 (LOC P2 BD)))))
(PLAN ((P1 SQ2) NIL (GOAL1 PLAN))
 (THREAT (PLUS (GOAL1 THREAT) (EXCHVAL P1 SQ2))) (PLAN)))

MOVE2
M2SAFE
((P1 SQ2 SQ3)
(PATTERN BACK1 P1 SQ2 SQ3)
(FUNCTION SAFEP P1 SQ3)
(CHECKCOND COND)
(ACTION MOVE (P1 SQ2) (THREAT (PLUS (GOAL1 THREAT) (EXCHVAL P1 SQ3)))
 (LIKELY (ADD1 (GOAL1 LIKELY)))
 (DECOYCOND (NEVER (FUNCTION CANATTACK DP1 SQ1 SQ3)))
 (PLAN (SAFEMOVE P1 SQ2) NIL (GOAL1 PLAN))))

MOVE2
M2DSAFE
((P1 SQ2 SQ3 P3 DP1)
(PATTERN BACK1 P1 SQ2 SQ3)
(NEVER (FUNCTION SAFEP P1 SQ3))
(PATTERN ONLYPRO DP1 SQ3)
(PATTERN DSC P3 P1 SQ3)
(GREAT (VALUE DP1) (VALUE P1))
(CHECKCOND COND)
(ACTION MOVE (P1 SQ2) (LIKELY (ADD1 (GOAL1 LIKELY)))
 (DECOYCOND (NEVER (FUNCTION CANATTACK DP1 SQ1 SQ3)))
 (PLAN (SAFEMOVE P1 SQ2) NIL (GOAL1 PLAN))))
```

```
MOVE2
M2UNSAFE
((P1 SQ2 SQ3)
(PATTERN BACK1 P1 SQ2 SQ3)
(NEVER (PATTERN M2SAFE P1 SQ2 SQ3))
(CHECKCOND COND)
(ACTION MOVE (P1 SQ2) (LIKELY (ADD1 (GOAL1 LIKELY)))
 (PRECOND (SAFE P1 SQ3))
 (DECOYCOND (NEVER (FUNCTION CANATTACK DP1 SQ1 SQ3)))
 (PLAN (SAFEMOVE P1 SQ2) NIL (GOAL1 PLAN))))

MOVE2
M2UWIN
((P1 P2 DP1 SQ2 SQ3)
(PATTERN BACK1 P1 SQ2 SQ3)
(NEVER (PATTERN M2SAFE P1 SQ2 SQ3))
(CHECKCOND COND)
(PATTERN ONLYPRO DP1 SQ3)
(PATTERN LEGALMOVE P2 (LOC DP1 BD))
(FUNCTION NEQ P1 P2)
(OR     (PATTERN MOBIL P1 SQ2)
        ((NEVER (FUNCTION OVERPRO (COLOR P1) SQ2))
         (PATTERN DIR DP1 SQ2)))
(GREAT TOPTHREAT (EXCHLOSS P2 (LOC DP1 BD)))
(PLAN ((P2 (LOC DP1 BD)) NIL (SAFEMOVE P1 SQ2) NIL (GOAL1 PLAN))
 (LIKELY (ADD1 (GOAL1 LIKELY))) (PLAN)
 (THREAT (PLUS (GOAL1 THREAT) (EXCH P2 DP1)))))

MOVE2
CLEAR
((P1 P2 SQ2 SQ3)
(PATTERN DIR P1 SQ3)
(FUNCTION EQUAL P2 (OCCUPANT SQ3 BD))
(NEVER (FUNCTION PAWN P1))
(FUNCTION SAFEP P1 SQ3)
(PATTERN MOBIL P2 SQ2)
(PIECESAFE P1 (EXCHVAL P2 SQ2))
(ACTION MOVE (P2 SQ2) (LIKELY (ADD1 (GOAL1 LIKELY)))
 (DECOYCOND (NEVER (FUNCTION CANATTACK DP1 SQ1 SQ3)))
 (PLAN (SAFEMOVE P1 SQ2) NIL (GOAL1 PLAN))))
```

```
MOVE1
CLEAR1
((P1 SQ3 P2 SQ)
(PATTERN DIR P1 SQ3)
(FUNCTION EQUAL P2 (OCCUPANT SQ3 BD))
(OR     (FUNCTION SAFEP P1 SQ3)
        ((FUNCTION SAFEP P2 SQ3) (ASGREAT (VALUE P2) (VALUE P1))))
(IMPLIES NIL ((PATTERN DIR P2 (LOC P1 BD)))
        (GREAT (VALUE P2) (VALUE P1)))
(OR     (PATTERN MOBSTAY P2 SQ)
        ((PATTERN MOBIL P2 SQ)
         (EXISTS (DP1) (FUNCTION EQUAL DP1 (OCCUPANT SQ BD))
                (ASGREAT (VALUE DP1) (VALUE P2)))))
(NEVER (FUNCTION LINE (LOC P1 BD) SQ SQ3))
(ACTION FORCE (P2 SQ P1) (PLAN NIL (GOAL1 PLAN)) (COND)(PLAN1)))

MOVE1
SACLINE
((P1 SQ3 P2 DP1 DP2)
(FUNCTION SAFEP P1 SQ3)
(PATTERN OTHRU P1 P2 (LOC DP1 BD))
(FUNCTION LINE (LOC P1 BD) (LOC DP1 BD) SQ3)
(FUNCTION CANATTACK P1 (LOC DP1 BD) SQ3)
(PATTERN ONLYDIR DP2 (LOC DP1 BD))
(GREAT TOPTHREAT (EXCHLOSS P2 (LOC DP1 BD)))
(PLAN ((P2 (LOC DP1 BD)) (DP2 (LOC DP1 BD)) (KILL P1 DP2) NIL (GOAL1 PLAN))
  (LIKELY (ADD1 (GOAL1 LIKELY))) (THREAT (PLUS (EXCH P2 DP1) (GOAL1 THREAT)))
  (PLAN) (PLAN1)))

MOVE1
M1THRUDK
((P1 LP1 SQ SQ3 DK)
(FUNCTION EQUAL P1 LP1)
(FUNCTION SAFEP P1 SQ3)
(PATTERN CHECKMOVE P1 SQ)
(PATTERN MOBSTAY P1 SQ)
(FUNCTION LINE SQ (LOC DK BD) SQ3)
(FUNCTION CANATTACK P1 (LOC DK BD) SQ3)
(PLAN ((P1 SQ) (DK NIL) (GOAL1 PLAN))
  (THREAT (PLUS (EXCHVAL P1 SQ) (GOAL1 THREAT))) (PLAN)))
```

```
MOVE1
M1COND
((P1 SQ3) (NEVER
 (OR    (PATTERN LEGALMOVE P1 SQ3)
                (EXISTS (AP1) (PATTERN THRU P1 AP1 SQ3))))
(RETURN))

MOVE1
M1DSCAT
((P1 P2 SQ1 SQ3 DMP1)
(PATTERN DSC P1 P2 SQ3)
(FUNCTION SAFEP P1 SQ3)
(PATTERN LEGALMOVE P2 SQ1)
(FUNCTION CANATTACK P2 SQ1 (LOC DMP1 BD))
(FUNCTION SAFEP P2 (LOC DMP1 BD))
(GREAT THREAT (EXCHLOSS P2 SQ1))
(IMPLIES (DP1) ((FUNCTION EQUAL SQ3 (LOC DP1 BD))
                (OR (FUNCTION SAMEVAL DP1 P2) (GREAT (VALUE P2) (VALUE DP1))))
        (NEVER (PATTERN DIR DP1 SQ1)))
(PIECESAFE P1 (EXCHVAL P2 (LOC DMP1 BD)))
(PLAN ((P2 SQ1) ( (NIL (GOAL1 PLAN))
                  (NIL (KILL P2 DMP1))))
  (THREAT (FORKT (EXCH P2 DMP1) (PLUS (EXCHVAL P2 SQ1) (GOAL1 THREAT))))
  (PLAN) ))

MOVE1
M1DSCCAP
((P1 P2 SQ3 DMP1)
(PATTERN DSC P1 P2 SQ3)
(FUNCTION SAFEP P1 SQ3)
(PATTERN LEGALMOVE P2 (LOC DMP1 BD))
(GREAT THREAT (EXCHLOSS P2 (LOC DMP1 BD)))
(IMPLIES (DP1) ((FUNCTION EQUAL SQ3 (LOC DP1 BD)))
        (NEVER (PATTERN DIR DP1 (LOC DMP1 BD))))
(PIECESAFE P1 (EXCHVAL P2 (LOC DMP1 BD)))
(PLAN ((P2 (LOC DMP1 BD)) NIL (GOAL1 PLAN))
        (THREAT (EXCH P2 DMP1) (GOAL1 THREAT)) (PLAN))
(RETURN))

MOVE1
M1THRU
((P1 SQ3 DP1)
(PATTERN THRU P1 DP1 SQ3)
(NEVER (EXISTS (DP2) (PATTERN DIR DP2 (LOC DP1 BD))))
(PLAN ((P1 (LOC DP1 BD)) NIL (GOAL1 PLAN)) (PLAN) (COND)
         (THREAT (PLUS (WIN DP1) (PLUS (GOAL1 THREAT) (EXCHVAL P1 SQ3)))))
(RETURN))
```

```
MOVE1
M1DECBLK
((P1 SQ3 DP1)
(PATTERN THRU P1 DP1 SQ3)
(OR     ((FUNCTION SAFEP P1 SQ3) (NEVER (FUNCTION EXCHANGE P1 SQ3)))
        ((PATTERN DIR DP1 SQ3) (NEVER (FUNCTION OVERPRO (COLOR P1) SQ3))))
(CHECKCOND SAFECOND)
(ACTION DECOY (P1 SQ3 DP1)
 (PLAN (DP1 NIL) (GOAL1 PLAN)))
(RETURN))

MOVE1
M2DECBLK
((P1 SQ3 DP1)
(PATTERN THRU P1 DP1 SQ3)
(NEVER (PATTERN M1DECBLK P1 SQ3 DP1))
(CHECKCOND SAFECOND)
(ACTION DECOY (P1 SQ3 DP1)     (PLAN (DP1 NIL) (GOAL1 PLAN))
 (DECOYCOND (EXISTS (DP2) (FUNCTION EQUAL SQ1 (LOC DP2 BD))
          (PATTERN DIR DP2 SQ3) (NEVER (FUNCTION CANATTACK DP1 SQ1 SQ3)))))
(RETURN))

MOVE1
QUIT
((P1 SQ3)
(NEVER (PATTERN LEGALMOVE P1 SQ3))
(RETURN))

MOVE1
M1
((P1 SQ3)
(PATTERN MOBSTAY P1 SQ3)
(IMPLIES (DP1) ((FUNCTION EQUAL DP1 (OCCUPANT SQ3 BD)))
        (NEVER (FUNCTION SAFEP DP1 SQ3)))
(REMATRS T ((PRECOND NIL)))
(PLAN ((P1 SQ3) (GOAL1C PLAN)) (PLAN) (COND)
         (THREAT (PLUS (GOAL1 THREAT) (EXCHVAL P1 SQ3)))
        (PLAN1 (SAFEMOVE P1 SQ3)))
(RETURN))

MOVE1
M1PREWIN
((P1 SQ3 P2 DP1 SQ2)
(PATTERN MOBSTAY P1 SQ3)
(NEVER (GOAL (PRECOND NIL)))
(FOREACHGOAL
(OR (GOAL1 (PRECOND (SAFE P2 SQ2)))
    (GOAL1 (PRECOND (SAFER P2 SQ2))))
(UNIQUE (PATTERN DIR DP1 SQ2))
(FUNCTION EQUAL SQ3 (LOC DP1 BD))
(PLAN ((P1 SQ3) (GOAL1C PLAN)) (PLAN) (COND)
        (THREAT (PLUS (GOAL1 THREAT) (EXCH P1 DP1)))
        (PLAN1 (KILL P1 DP1))))
(RETURN))
```

```
MOVE1
M1PREWN2
((P1 SQ3 LP1 DP1 DP2 SQ2)
(PATTERN MOBSTAY P1 SQ3)
(NEVER (GOAL (PRECOND NIL)))
(FOREACHGOAL
(OR (GOAL1 (PRECOND (SAFE P1 SQ2)))
    (GOAL1 (PRECOND (SAFER P1 SQ2))))
(UNIQUE (PATTERN DIR DP1 SQ2))
(GREAT (VALUE DP1) (VALUE P1))
(PATTERN THRU LP1 DP2 SQ2)
(FUNCTION EQUAL SQ3 (LOC DP2 BD))
(PLAN ((P1 SQ3) (GOAL1C PLAN)) (PLAN)
 (THREAT (PLUS (EXCH P1 DP2) (GOAL1 THREAT)))
 (PLAN1 (KILL P1 DP2)) ))
(RETURN))

MOVE1
M1PRE
((P1 SQ3)
(PATTERN MOBSTAY P1 SQ3)
(NEVER (GOAL (PRECOND NIL)))
(FOREACHGOAL
(ACTION (GOAL1 PRECOND) (PLAN NIL (GOAL1 PLAN))
        (THREAT (PLUS (GOAL1 THREAT) (EXCHVAL P1 SQ3)))
        (PRECOND)(COND) ))
(RETURN))

MOVE1
M1OTHRU
((P1 SQ3 P2 ATR1)
(OR     (PATTERN OTHRU P2 P1 SQ3)
        ((PATTERN DIR P1 SQ3) (PATTERN DIR P2 SQ3) (FUNCTION NEQ P1 P2)))
(OR     (CONCEPT MOVE (P2 SQ3) NIL ((ATR1 THREAT)))
        (CONCEPT MOVE1 (P2 SQ3) NIL ((ATR1 THREAT))))
(NEVER (FUNCTION OVERPRO (COLOR P2) SQ3))
(PLAN ((P1 SQ3) (NIL SQ3) (SAFEMOVE P2 SQ3))  (PLAN)
 (THREAT (FORKT (GOAL1 THREAT) (PLUS (EXCHVAL P1 SQ3) ATR1)))))

MOVE1
M1UNSAFE
((P1 SQ3)
(NEVER (PATTERN MOBIL P1 SQ3))
(NEVER (FUNCTION EXCHANGE P1 SQ3))
(REMATRS T ((PRECOND NIL)))
(ACTION SAFE (P1 SQ3) (PLAN NIL (GOAL1 PLAN)) (COND))
(RETURN))

MOVE1
M1USAFE
((P1 SQ3)
(OR     (FUNCTION EXCHANGE P1 SQ3)
        (PATTERN MOBSTAY P1 SQ3))
(REMATRS T ((PRECOND NIL)))
(ACTION SAFER (P1 SQ3) (PLAN NIL (GOAL1 PLAN)) (COND)))
```

```
SAFE1
COND
S1COND
((P1 SQ)
(OR     ((PATTERN LEGALMOVE P1 SQ)
         (NEVER (FUNCTION OVERPRO (COLOR P1) SQ)))
        ((NEVER (PATTERN LEGALMOVE P1 SQ))
         (IMPLIES NIL ((FUNCTION OVERPRO (COLOR P1) SQ))
          (EXISTS (P2) (PATTERN DIR P2 SQ)))))
(NEVER (EXISTS (DP1) (PATTERN DIR DP1 SQ)
        (FUNCTION ASVAL P1 OP1) (FUNCTION NPIN DP1))))

SAFE1
SUPI
((P1 P2 SQ1 SQ)
(PATTERN MOBSTAY P2 SQ1)
(CHECKCOND SAFECOND1)
(FUNCTION CANATTACK P2 SQ1 SQ)
(FUNCTION NEQ P1 P2)
(NEVER (FUNCTION LINE (LOC P1 BD) SQ1 SQ))
(NEVER (PATTERN DIR P2 SQ))
(PIECESAFE P1 (EXCHVAL P2 SQ1))
(NEVER (EXISTS (DP1)    (PATTERN DIR DP1 SQ)
        (GREAT (VALUE P1) (VALUE DP1))  (GREAT (VALUE P2) (VALUE DP1))))
(ACTION FORCE (P2 SQ1 P1)))

SAFE1
SUPID
((P1 P3 P2 SQ SQ1)
(OR     (PATTERN DSC P3 P2 SQ)
        ((PATTERN THRU P3 P2 (LOC P1 BD))
         (FUNCTION LINE (LOC P3 BD) (LOC P1 BD) SQ)
         (PATTERN LEGALMOVE P1 SQ) (NEVER (PATTERN OTHRU P2 P1 SQ))))
(FUNCTION NEQ P1 P2)
(CHECKCOND SAFECOND1)
(PATTERN LEGALMOVE P2 SQ1)
(NEVER (FUNCTION LINE (LOC P3 BD) SQ1 SQ))
(OR     (PIECESAFE P1 (EXCHVAL P2 SQ1))
        (EXISTS (DMP1) (FUNCTION CANATTACK P2 SQ1 (LOC DMP1 BD))
         (PIECESAFE P1 (EXCHVAL P2 (LOC DMP1 BD)))))
(IMPLIES NIL ((NEVER (PATTERN MOBIL P2 SQ1)))
        (GOAL (LIKELY 0))  (GREAT THREAT (VALUE P2)))
(ACTION FORCE (P2 SQ1 P1)))

SAFE1
DECBLK
((P1 SQ DP1 LP1)
(PATTERN LEGALMOVE P1 SQ)
(PATTERN THRU LP1 DP1 SQ)
(CHECKCOND SAFECOND)
(ACTION DECOY (P1 SQ DP1)
 (DECOYCOND (NEVER (FUNCTION LINE (LOC LP1 BD) SQ1 SQ))
        (NEVER (FUNCTION EQUAL P2 LP1)))))
```

```
SAFE1
SUPCHK
((P1 LP1 SQ1 SQ DK)
(PATTERN CHECKMOVE LP1 SQ1)
(FUNCTION NEQ P1 LP1)
(FUNCTION LINE SQ1 (LOC DK BD) SQ)
(PATTERN MOBSTAY LP1 SQ1)
(NEVER (FUNCTION LINE (LOC P1 BD) SQ1 SQ))
(NEVER (PATTERN DIR LP1 SQ))
(PLAN ((LP1 SQ1) (GOAL PLAN))
  (THREAT (PLUS (GOAL1 THREAT) (EXCHVAL LP1 SQ1))) (PLAN)))

SAFER
BLOCKP
((P1 SQ DP1 P2 SQ1)
(PATTERN LEGALMOVE P1 SQ)
(PATTERN DIR DP1 SQ)
(CHECKCOND SAFECOND)
(FUNCTION LINE SQ SQ1 (LOC DP1 BD))
(NEVER (FUNCTION SAFENB DP1 SQ1))
(PATTERN LEGALMOVE P2 SQ1)
(CHECKCOND SAFECOND1)
(FUNCTION NEQ P1 P2)
(IMPLIES NIL ((PATTERN DIR P2 SQ)) (FUNCTION CANATTACK P2 SQ1 SQ))
(GREAT THREAT (EXCHLOSS P2 SQ1)))

SAFER
BLOCK1
((P1 SQ DP1 P2 SQ1)
(PATTERN BLOCKP P1 SQ DP1 P2 SQ1)
(NEVER (FUNCTION OVERPRO (COLOR P1) SQ))
(ACTION FORCE (P2 SQ1 P1)))

SAFER
BLOCK2
((P1 SQ OP1 P2 SQ1)
(PATTERN BLOCKP P1 SQ DP1 P2 SQ1)
(NEVER (PATTERN BLOCK1 P1 SQ DP1 P2 SQ1))
(NEVER (FUNCTION OVERPRO2 (COLOR P1) SQ))
(OR     ((PATTERN MOBIL P2 SQ1)
         (FUNCTION CANATTACK P2 SQ1 SQ))
        (IMPLIES (DP2) ((PATTERN DIR DP2 SQ1) (FUNCTION SAFENB DP2 SQ1))
         (PATTERN DIR DP2 SQ)
         (NEVER (FUNCTION CANATTACK DP2 SQ1 SQ))))
(ACTION FORCE (P2 SQ1 P1)))

SAFER
SRCOND
((P1 SQ)
(FUNCTION OVERPRO (COLOR P1) SQ)
(IMPLIES NIL ((NEVER (PATTERN LEGALMOVE P1 SQ)))
  (NEVER (EXISTS (P2) (PATTERN DIR P2 SQ))))
(RETURN))
```

```
SAFER
CAPOPP
((P1 OP1 SQ)
(OR     (PATTERN DIR DP1 SQ)
        (EXISTS (AP1 AP2 K DK) (PATTERN THRU DP1 AP1 SQ)
          (PATTERN DIR AP1 SQ)
          (NEVER (PATTERN PIN AP2 AP1 DK))
          (NEVER (PATTERN PIN AP2 AP1 K))))
(IMPLIES (LP1 DMP1) ((PATTERN PIN LP1 DP1 DMP1))
        (FUNCTION ASVAL DP1 DMP1))
(CHECKCOND SAFECOND)
(ACTION CAPTURE (DP1 P1)))

SAFER
PINOPP
((P1 SQ DSP1 DK LP1 SQ1)
(PATTERN DIR DSP1 SQ)
(PATTERN MOBSTAY LP1 SQ1)
(NEVER (PATTERN DIR LP1 SQ))
(NEVER (PATTERN PIN LP1 DSP1 DK))
(FUNCTION LINE SQ1 (LOC DSP1 BD) (LOC DK BD))
(FUNCTION CANATTACK LP1 SQ1 (LOC DSP1 BD))
(FUNCTION CANATTACK LP1 (LOC DSP1 BD) (LOC DK BD)))

SAFER
POWIN
((P1 SQ DSP1 DK LP1 SQ1)
(PATTERN PINOPP P1 SQ DSP1 DK LP1 SQ1)
(PATTERN MATE P1 SQ)
(EXISTS (DSP2) (UNIQUE (PATTERN DIR DSP2 SQ)))
(PLAN ((LP1 SQ1) (GOAL1 PLAN))
 (THREAT (PLUS (GOAL1 THREAT) (EXCHVAL LP1 SQ1))) (PLAN)))

SAFER
PO1
((P1 SQ DSP1 DK LP1 SQ1)
(PATTERN PINOPP P1 SQ DSP1 DK LP1 SQ1)
(NEVER (PATTERN POWIN P1 SQ DSP1 DK LP1 SQ1))
(ACTION FORCE (LP1 SQ1 P1)))
```

```
SAFER
DECOYSUP
((P1 SQ DP1)
(PATTERN LEGALMOVE P1 SQ)
(NEVER (FUNCTION OVERPRO (COLOR P1) SQ))
(PATTERN DIR DP1 SQ)
(NEVER  (EXISTS (DP2) (PATTERN DIR DP2 SQ) (FUNCTION NEQ DP2 DP1)
                (FUNCTION ASVAL P1 DP2) (FUNCTION NPIN DP2)))
(CHECKCOND SAFECOND))

SAFER
DS1
((P1 SQ DP1)
(PATTERN DECOYSUP P1 SQ DP1)
(IMPLIES NIL ((PATTERN ONLYPRO P1 SQ))
 (NEVER (EXISTS (DP2) (PATTERN DIR DP2 SQ) (FUNCTION NEQ DP2 DP1)
                (FUNCTION NPIN DP2))))
(ACTION DECOY (P1 SQ DP1)))

SAFER
DS2
((P1 SQ DP1)
(PATTERN DECOYSUP P1 SQ DP1)
(NEVER (PATTERN DS1 P1 SQ DP1))
(ACTION DECOY (P1 SQ DP1)  (DECOYCOND (PATTERN MOBIL P2 SQ1))))

SAFER
WINEX1
((P1 SQ P2 DP1)
(PATTERN MOBIL P2 SQ)
(CHECKCOND SAFECOND1)
(FUNCTION NEQ P1 P2)
(FUNCTION EQUAL SQ (LOC DP1 BD))
(PIECESAFE P1 (EXCHVAL P2 SQ))
(PLAN ((P2 SQ) (NIL SQ) (GOAL1C PLAN))
 (PLAN) (THREAT (PLUS (GOAL1 THREAT) (EXCH P2 DP1)))))

SAFER
WINEX2
((P1 SQ P2)
(PATTERN MOBIL P2 SQ)
(FUNCTION NEQ P1 P2)
(CHECKCOND SAFECOND1)
(EXISTS (DP1)   (NEVER (PATTERN WINEX1 P1 SQ P2 DP1))
                (PATTERN DIR DP1 SQ)
                (FUNCTION CANATTACK P2 SQ (LOC DP1 BD))
                (FUNCTION SAFEP P2 (LOC DP1 BD))
                (PIECESAFE P1 (EXCHVAL P2 (LOC DP1 BD))))
(PLAN ((P2 SQ) (NIL SQ) (GOAL1C PLAN))   (PLAN)))
```

```
SAFER
DECOYSAC
((P1 SQ P2 DP1 ATR1)
(PATTERN LEGALMOVE P1 SQ)
(OR (CONCEPT SAFE (P2 SQ) NIL ((ATR1 PLAN)))
    (CONCEPT SAFER (P2 SQ) NIL ((ATR1 PLAN))))
(FUNCTION NEQ P1 P2)
(GREAT THREAT (VALUE P2))
(PATTERN LEGALMOVE P2 SQ)
(PATTERN ONLYPRO DP1 SQ)
(OR (NEVER (EXISTS (DP2) (PATTERN ENPRIS DP2 P1)))
    (PATTERN CHECKMOVE P2 SQ))
(PLAN ((P2 SQ) (((NIL SQ) (P1 SQ) (REPLACE (SAFEMOVE P1 SQ) (GOAL PLAN)))
        ((REPLACE (SAFEMOVE P2 SQ) ATR1))))
 (THREAT (PLUS (GOAL1 THREAT) (EXCHVAL P2 SQ))) (PLAN)))

SAFER
MAKEFLY
((P1 SQ P2 SQ1 DP1)
(PATTERN DIR DP1 SQ)
(CHECKCOND SAFECOND)
(NEVER (PATTERN MOBIL P2 (LOC DP1 BD)) (FUNCTION NEQ P1 P2))
(PATTERN MOBSTAY P2 SQ1)
(CHECKCOND SAFECOND1)
(FUNCTION CANATTACK P2 SQ1 (LOC DP1 BD))
(NEVER (PATTERN DIR P2 (LOC DP1 BD)))
(IMPLIES NIL ((FUNCTION EQUAL P2 P1))
        (FUNCTION CANATTACK P1 SQ1 SQ)
        (IMPLIES (DP1) ((FUNCTION EQUAL DP1 (OCCUPANT SQ BD)))
                (NEVER (FUNCTION SAFEP DP1 SQ))))
(FUNCTION SAFEP P2 (LOC DP1 BD))
(PIECESAFE P1 (EXCHVAL P2 (LOC DP1 BD)))
(IMPLIES (SQ2) ((OR (PATTERN MOBIL DP1 SQ2)
        ((PATTERN LEGALMOVE DP1 SQ2) (PATTERN ONLYPRO P2 SQ2))))
 (OR    (FUNCTION CANATTACK P2 SQ1 SQ2)
        (EXISTS (P3) (PATTERN THRU P3 P2 SQ2))
        (NEVER (FUNCTION CANATTACK DP1 SQ2 SQ)))))

SAFER
MAKEFLY1
((P1 SQ P2 SQ1 DP1)
(PATTERN MAKEFLY P1 SQ P2 SQ1 DP1)
(NEVER (FUNCTION LINE (LOC P1 BD) SQ1 SQ))
(NEVER (EXISTS (DP2) (PATTERN DIR DP2 SQ)
        (FUNCTION NEQ DP2 DP1)
        (GREAT (VALUE P1) (VALUE DP2))
        (NEVER (EXISTS (P3 DP3) (PATTERN PIN P3 DP2 DP3) (FUNCTION NEQ P3 P2)))))
(PLAN ((P2 SQ1) (GOAL1 PLAN)) (PLAN)
   (THREAT (PLUS (EXCHVAL P2 SQ1) (FORKT (GOAL1 THREAT) (EXCH P2 DP1))))))

SAFER
MAKEFLY2
((P1 SQ P2 SQ1 DP1)
(PATTERN MAKEFLY P1 SQ P2 SQ1 DP1)
(NEVER (PATTERN MAKEFLY1 P1 SQ P2 SQ1 DP1))
(PLAN ((P2 SQ1) (GOAL1 PLAN)) (PLAN) (LIKELY (ADD1 (GOAL1 LIKELY)))
   (THREAT (PLUS (EXCHVAL P2 SQ1) (FORKT (GOAL1 THREAT) (EXCH P2 DP1))))))
```

```
DECOY
MOVDEC
((P1 SQ P2 SQ1 DP1)
(PATTERN DIR DP1 SQ1)
(NEVER (FUNCTION LINE (LOC P1 BD) SQ1 SQ))
(OR     (NEVER (FUNCTION OVERPRO (COLOR P1) SQ1))
        (PATTERN ONLYDIR OP1 SQ1))
(NEVER (FUNCTION CANATTACK OP1 SQ1 SQ))
(NEVER (FUNCTION LINE SQ1 (LOC DP1 BD) SQ))
(PATTERN LEGALMOVE P2 SQ1)
(FUNCTION NEQ P1 P2)
(OR     (NEVER (PATTERN DIR P2 SQ))
        ((PATTERN ONLYPRO DP1 SQ) (FUNCTION NEQ SQ SQ1))
        (PATTERN THRU P1 DP1 SQ))
(GREAT TOPTHREAT (DIF (VALUE P2) (VALUE (OCCUPANT SQ1 BD))))
(NEVER (EXISTS (P3 DP2 DK) (PATTERN PIN P3 DP1 DP2)
  (OR   (FUNCTION EQUAL DK DP2)
        ((ASGREAT (VALUE DP2) (VALUE P2)) (ASGREAT (VALUE DP2) TOPTHREAT)))))
(CHECKCOND DECOYCOND)
(ACTION FORCE (P2 SQ1 P1) (DECOY DP1) (PLAN (DP1 SQ1) (GOAL1C PLAN))))

DECOY
SPLIT
((P1 SQ P2 SQ1 DSP1)
(PATTERN MOVDEC P1 SQ P2 SQ1 DSP1)
(FUNCTION CANATTACK P2 SQ1 (LOC DSP1 BD))
(NEVER (EXISTS (P3) (PATTERN MOBIL P3 (LOC DSP1 BD))))
(NEVER (EXISTS (DMP1) (FUNCTION CANATTACK P2 SQ1 (LOC DMP1 BD))
        (FUNCTION NEQ DMP1 DSP1) (FUNCTION SAFEP P2 (LOC DMP1 BD))
        (NEVER (PATTERN DIR P2 (LOC DMP1 BD)))))
(FUNCTION SAFEP P2 (LOC DSP1 BD))
(PIECESAFE P1 (EXCHVAL P2 (LOC DSP1 BD))))


DECOY
SPLITD
((P1 SQ P2 SQ1 DP1)
(PATTERN SPLIT P1 SQ P2 SQ1 DP1)
(PATTERN DIR DP1 SQ)
(IMPLIES (SQ2) ((OR (PATTERN MOBIL DP1 SQ2)
        ((PATTERN LEGALMOVE DP1 SQ2) (PATTERN ONLYPRO P2 SQ2))))
 (OR    (FUNCTION CANATTACK P2 SQ1 SQ2)
        (EXISTS (P3) (PATTERN THRU P3 P2 SQ2))
        (NEVER (FUNCTION CANATTACK DP1 SQ2 SQ))))
(PLAN ((P2 SQ1) (DP1 NIL) (GOAL1C PLAN)) (PLAN)
   (THREAT (PLUS (EXCHVAL P2 SQ1) (FORKT (GOAL1 THREAT) (EXCH P2 DP1))))))

DECOY
SPLITB
((P1 SQ P2 SQ1 DP1)
(PATTERN SPLIT P1 SQ P2 SQ1 DP1)
(NEVER (PATTERN DIR DP1 SQ))
(EXISTS (P3) (PATTERN OBJ P3 DP1 SQ)
(IMPLIES (SQ2) ((OR (PATTERN MOBIL DP1 SQ2)
        ((PATTERN LEGALMOVE DP1 SQ2) (PATTERN ONLYPRO P2 SQ2))))
 (OR    (FUNCTION CANATTACK P2 SQ1 SQ2)
        (EXISTS (P3) (PATTERN THRU P3 P2 SQ2))
        (NEVER (FUNCTION LINE (LOC P3 BD) SQ2 SQ)))))
(PLAN ((P2 SQ1) (DP1 NIL) (GOAL1C PLAN)) (PLAN)
   (THREAT (PLUS (EXCHVAL P2 SQ1) (FORKT (GOAL1 THREAT) (EXCH P2 DP1))))))
```

```
FORCE
COND
FCOND
((P1 SQ) (GREAT THREAT (EXCHLOSS P1 SQ)))

FORCE
FCHECK
((P1 SQ)
(PATTERN CHECKMOVE P1 SQ)
(PLAN ((P1 SQ) (GOAL PLAN)) (THREAT (PLUS (EXCHVAL P1 SQ) (GOAL1 THREAT)))
 (PLAN) (DECOY))
(RETURN))

FORCE
FMATE
((P1 SQ P2 SQ1 DK)
(PATTERN MATE P2 SQ1)
(PATTERN ONLYDIR DK SQ1)
(OR     (FUNCTION CANATTACK P1 SQ SQ1)
        ((FUNCTION CANATTACK P1 SQ (LOC P2 BD))
         (FUNCTION LINE SQ (LOC P2 BD) SQ1) (EXISTS (LP1) (FUNCTION EQUAL LP1 P1))))
(NEVER (FUNCTION LINE (LOC P2 BD) SQ SQ1))
(IMPLIES (SQ2)  ((PATTERN DIR DK SQ2) (NEVER (FUNCTION OCCBYCOL SQ2 (COLOR DK))))
  (OR (NEVER (PATTERN ONLYPRO P1 SQ2)) (FUNCTION CANATTACK P1 SQ SQ2)))
(PLAN ((P1 SQ) (GOAL PLAN)) (THREAT (PLUS (EXCHVAL P1 SQ) (GOAL1 THREAT)))
 (PLAN) (DECOY))
(RETURN))

FORCE
UNLIKE
((P1 SQ P2)
(GOAL (DECOY NIL))
(PATTERN MOBIL P1 SQ)
(PIECESAFE P2 THREAT)
(PLAN ((P1 SQ) (GOAL PLAN)) (PLAN) (LIKELY (ADD1 (GOAL1 LIKELY)))))

FORCE
FTHREAT
((P1 SQ DMP1 P2)
(FUNCTION CANATTACK P1 SQ (LOC DMP1 BD))
(NEVER (PATTERN DIR P1 (LOC DMP1 BD)))
(FUNCTION SAFEP P1 (LOC OMP1 BD))
(IMPLIES (AP1) ((PATTERN THRU P1 AP1 (LOC DMP1 BD)) (PATTERN DIR AP1 (LOC DMP1 BD)))
 (OR    (GREAT (VALUE DMP1) (VALUE P1))
        (EXISTS (P3) (PATTERN DIR P3 (LOC DMP1 BD)) (GREAT (VALUE P3) (VALUE P1)))))
(IMPLIES NIL ((PATTERN DIR DMP1 SQ))
  (OR   (PATTERN MOBIL P1 SQ)
        (NEVER (FUNCTION SAFENB DMP1 SQ))
        (GOAL (DECOY DMP1))))
(PIECESAFE P2 (PLUS (EXCHVAL P1 (LOC DMP1 BD)) (EXCHVAL P1 SQ))))

FORCE
CAPONLY
((P1 SQ DP1 P2)
(FUNCTION EQUAL DP1 (OCCUPANT SQ BD))
(OR     (PIECESAFE P2 (EXCHVAL P1 SQ))
        (PATTERN LEGALMOVE DP1 (LOC P2 BD)))
(PLAN ((P1 SQ) (GOAL PLAN))  (DECOY)
  (THREAT (FORKT (PLUS (EXCHVAL P1 SQ) (GOAL1 THREAT)) (WIN DP1))) (PLAN)))
```

```
FORCE
DECOK1
((P1 SQ P2)
(EXISTS (DMP1) (PATTERN FTHREAT P1 SQ DMP1 P2))
(REMATRS T ((DECOY NIL)))
(SET ATR1 (DMP1) (((PATTERN FTHREAT P1 SQ DMP1 P2)) (EXCHVAL P1 (LOC DMP1 BD))))
(PLAN ((P1 SQ) (GOAL1 PLAN))  (DECOY)
  (THREAT (FORKT (PLUS (EXCHVAL P1 SQ) (GOAL1 THREAT)) ATR1))
  (PLAN)))

FORCE
DECOK2
((P1 SQ P2 DMP1)
(REMATRS NIL ((DECOY NIL)))
(UNIQUE (PATTERN FTHREAT P1 SQ DMP1 P2))
(PLAN ((P1 SQ) (GOAL1 PLAN))  (DECOY) (LIKELY (ADD1 (GOAL1 LIKELY)))
  (THREAT (FORKT (PLUS (EXCHVAL P1 SQ) (GOAL1 THREAT)) (EXCHVAL P1 (LOC DMP1 BD))))
  (PLAN)))

FORCE
DECOK3
((P1 SQ P2)
(REMATRS NIL ((DECOY NIL)))
(EXISTS (DMP1) (NEVER (PATTERN DECOK2 P1 SQ P2 DMP1))
        (PATTERN FTHREAT P1 SQ DMP1 P2))
(SET ATR1 (DMP1) (((PATTERN FTHREAT P1 SQ DMP1 P2))  (EXCHVAL P1 (LOC DMP1 BD))))
(PLAN ((P1 SQ) (GOAL1 PLAN))  (DECOY)
  (THREAT (FORKT (PLUS (EXCHVAL P1 SQ) (GOAL1 THREAT)) ATR1))
  (PLAN)))
```

```
SAFEMOVE
YES
((P1 SQ)
(PATTERN MOBSTAY P1 SQ)
(PLAN ((P1 SQ)) (PLAN1 (SAFEMOVE P1 SQ)) (THREAT (EXCHVAL P1 SQ) (GOAL1 THREAT)))
(RETURN))

SAFEMOVE
NO
((P1 SQ)
(PATTERN LEGALMOVE P1 SQ)
(REMOVETHR P1 SQ)
(ACTION MOVE1 (P1 SQ)))

KILL
YESK
((P1 DP1)
(FUNCTION NEQ 0 (LOC DP1 BD))
(PATTERN MOBSTAY P1 (LOC DP1 BD))
(PLAN ((P1 (LOC DP1 BD))) (PLAN1 (KILL P1 DP1))
 (THREAT (EXCHVAL P1 (LOC DP1 BD)) (GOAL1 THREAT)))
(RETURN))

KILL
NOK
((P1 DP1)
(FUNCTION NEQ 0 (LOC DP1 BD))
(PATTERN LEGALMOVE P1 (LOC DP1 BD))
(FUNCTION EXCHANGE P1 (LOC DP1 BD))
(ACTION SAFER (P1 (LOC DP1 BD)) (PLAN NIL (GOAL1 PLAN))))

KILL
NOK1
((P1 DP1)
(FUNCTION NEQ 0 (LOC DP1 BD))
(PATTERN LEGALMOVE P1 (LOC DP1 BD))
(NEVER (PATTERN NOK P1 DP1))
(ACTION SAFE (P1 (LOC DP1 BD)) (PLAN NIL (GOAL1 PLAN))))

CAPTURE
EASY
((P1 P2 DP1)
(PATTERN MOBIL P2 (LOC DP1 BD))
(FUNCTION NEQ P1 P2)
(PIECESAFE P1 (EXCHVAL P2 (LOC DP1 BD)))
(PLAN ((P2 (LOC DP1 BD)) (GOAL PLAN))
 (THREAT (PLUS (GOAL1 THREAT) (EXCH P2 DP1)))
  (PLAN1 (KILL P2 DP1))  (PLAN)))
```

```
ATTACKP
ATCKP
((P1)
(ACTION ATTACK ((COMPCOLOR P1) (LOC P1 BD))))

CHKMOVE
YESCK
((P1 SQ)
(PATTERN MOBSTAY P1 SQ)
(PATTERN CHECKMOVE P1 SQ)
(PLAN ((P1 SQ)) (PLAN1 (CHKMOVE P1 SQ)) (THREAT (EXCHVAL P1 SQ) (GOAL1 THREAT)))
(RETURN))

CHKMOVE
NOCK
((P1 SQ DK)
(PATTERN CHECKMOVE P1 SQ)
(REMOVETHR P1 SQ)
(ACTION MOVE1 (P1 SQ) (SAFECOND (FUNCTION NEQ DP1 DK))
  (DECOYCOND (FUNCTION NEQ DP1 DK))))

MATEMOVE
YESMATE
((P1 SQ)
(PATTERN MOBSTAY P1 SQ)
(PATTERN MATE P1 SQ)
(PLAN ((P1 SQ)) (PLAN1 (MATEMOVE P1 SQ)) (THREAT (EXCHVAL P1 SQ) (GOAL1 THREAT)))
(RETURN))

MATEMOVE
NOMATE
((P1 SQ DK)
(PATTERN MATE P1 SQ)
(REMOVETHR P1 SQ)
(ACTION MOVE1 (P1 SQ) (SAFECOND (FUNCTION NEQ DP1 DK))
  (DECOYCOND (FUNCTION NEQ DP1 DK))))
```

```
ENPRI
NIL
ENPRISET

ENPRI
NIL
EP2

ENPRI
NIL
EP2P

ENPRI
NIL
EP2C

ENPRI
NIL
MATENOW1

ENP
NIL
MATENOW

ENP
NIL
ENPRISET

ENP
NIL
EP2

ENP
NIL
EP2P

ENP
NIL
EP2C

ENP
NIL
QUEENP
```

```
TERM
INCHECK
((P1 DK)
(PATTERN ENPRIS P1 DK)
(PLANNEW ((P1 (LOC DK BD))) (LIKELY 0) (THREAT (WIN DK)))
(RETURN))

TERM
NIL
MATENOW

TERM
NIL
MATENOW1
TERM
NIL
EP2
TERM
NIL
EP2P
TERM
NIL
EP2C

TERM
NIL
QUEENP

TERM
MOBTERM
((P1 SQ)
(PATTERN MOBSTAY P1 SQ)
(NEVER (EXISTS (DP1) (PATTERN ENPRIS P1 DP1)    (FUNCTION EQUAL SQ (LOC DP1 BD))
  (EXISTS (DP2) (PATTERN DIR OP2 SQ)
   (OR  (FUNCTION SAFENB DP2 SQ)
        ((ASGREAT (VALUE P1) (VALUE DP2))
         (IMPLIES (LP1 DP3) ((PATTERN PIN LP1 DP2 DP3)
                (FUNCTION NEQ LP1 P1) (FUNCTION NEQ DP3 DP1))
          (GREAT (DIF (VALUE P1) (VALUE DP2)) (EXCHVAL LP1 (LOC DP3 BD)))))))))

TERM
FORK1
((SP1 SQ DMP1 DMP2)
(FUNCTIONE FRKSQ SP1 SQ DMP1 DMP2)
(PATTERN MOBTERM SP1 SQ)
(NEVER (PATTERN DIR SP1 (LOC DMP1 BD)))
(NEVER (PATTERN DIR SP1 (LOC DMP2 BD))))

TERM
NOFORK
((SP1 SQ DMP1 DMP2 MP1)
(OR     (PATTERN FORK1 SP1 SQ DMP1 DMP2)
        (PATTERN FORK1 SP1 SQ DMP2 DMP1))
(PATTERN LEGALMOVE DMP1 (LOC MP1 BD))
(FUNCTION NEQ MP1 SP1)
(DR     (ASGREAT (EXCHVAL OMP1 (LDC MP1 BD)) (EXCHVAL SP1 (LOC DMP2 BD)))
        ((PATTERN ONLYPRO SP1 (LOC MP1 BD))
        (NEVER (FUNCTION CANATTACK SP1 SQ (LOC MP1 BD)))
        (ASGREAT (VALUE MP1) (EXCHVAL SP1 (LOC DMP2 BD))))))
```

```
TERM
FORKWIN
((SP1 SQ DMP1 DMP2)
(PATTERN FORK1 SP1 SQ DMP1 DMP2)
(NEVER (EXISTS (MP1) (OR (PATTERN NOFORK SP1 SQ DMP1 DMP2 MP1)
                         (PATTERN NOFORK SP1 SQ DMP2 DMP1 MP1))))
(OR     (PATTERN ONLYDIR (OCCUPANT SQ BD) (LOC DMP1 BD))
        (PATTERN ONLYDIR (OCCUPANT SQ BD) (LOC DMP2 BD)))
(PLANNEW ((SP1 SQ)) (THREAT (PLUS (FORKT (WIN DMP1) (WIN DMP2)) (EXCHVAL SP1 SQ)))
        (LIKELY 0)))

TERM
FORKREG
((SP1 SQ DMP1 DMP2)
(PATTERN FORK1 SP1 SQ DMP1 DMP2)
(NEVER (PATTERN FORKWIN SP1 SQ DMP1 DMP2))
(NEVER (EXISTS (MP1) (OR (PATTERN NOFORK SP1 SQ DMP1 DMP2 MP1)
                         (PATTERN NOFORK SP1 SQ DMP2 DMP1 MP1))))
(IMPLIES NIL ((ASGREAT (VALUE SP1) (VALUE DMP2)))
 (NEVER (EXISTS (SQ1) (PATTERN DIR DMP1 SQ1)
        (FUNCTION CANATTACK DMP1 SQ1 (LOC DMP2 BD))
        (NEVER (FUNCTION CANATTACK SP1 SQ SQ1))
        (NEVER (FUNCTION OCCBYCOL SQ1 (COLOR DMP1)))
        (OR (PATTERN MOBSTAY DMP1 SQ1) (PATTERN ONLYPRO SP1 SQ1)))))
(IMPLIES NIL ((ASGREAT (VALUE SP1) (VALUE DMP1)))
 (NEVER (EXISTS (SQ1) (PATTERN DIR DMP2 SQ1)
        (FUNCTION CANATTACK DMP2 SQ1 (LOC DMP1 BD))
        (NEVER (FUNCTION CANATTACK SP1 SQ SQ1))
        (NEVER (FUNCTION OCCBYCOL SQ1 (COLOR DMP2)))
        (OR (PATTERN MOBSTAY DMP2 SQ1) (PATTERN ONLYPRO SP1 SQ1)))))
(PLANNEW ((SP1 SQ)) (THREAT (PLUS (FORK SP1 DMP1 DMP2) (EXCHVAL SP1 SQ)))
        (LIKELY 0)))

TERM
FORKSAC1
((SP1 SQ DMP1 DMP2 DK P2)
(OR     (PATTERN FORK1 SP1 SQ DMP1 DMP2)
        (PATTERN FORK1 SP1 SQ DMP2 DMP1))
(NEVER (EXISTS (MP1) (PATTERN NOFORK SP1 SQ DMP2 DMP1 MP1)))
(PATTERN ONLYPRO DK (LOC DMP1 BD)) (FUNCTION NEQ DK DMP2)
(IMPLIES NIL ((PATTERN DIR DMP2 SQ)) (GREAT (VALUE DMP2) (VALUE SP1)))
(PATTERN LEGALMOVE P2 (LOC DMP1 BD)) (FUNCTION NEQ P2 SP1)
(GREAT (PLUS (VALUE DMP2) (VALUE DMP1)) (VALUE P2))
(IMPLIES NIL ((FUNCTION CANATTACK DK (LOC DMP1 BD) SQ))
 (GREAT (PLUS (VALUE DMP1) (VALUE DK)) (PLUS (VALUE P2) (VALUE SP1))))
(PLANNEW ((P2 (LOC DMP1 BD)) (DK (LOC DMP1 BD)) (SAFEMOVE SP1 SQ)
  (((ANYBUT DK) (KILL SP1 DK)) ((ANYBUT DMP2) (KILL SP1 DMP2))))
 (LIKELY 0)   (THREAT (LESS (PLUS (WIN (OCCUPANT SQ BD))
        (PLUS (WIN DMP1) (FORKT (WIN DK) (EXCH SP1 DMP2)))) (WIN P2)))))

TERM
ENPRISET
((P1 DP1)
(PATTERN ENPRIS P1 DP1)
(PLANNEW ((P1 (LOC DP1 BD))) (THREAT (EXCH P1 DP1)) (LIKELY 0)))
```

```
TERM
DSC1
((P1 P2 DP1 DP2 SQ)
(PATTERN DSCTHR P2 P1 DP1 SQ)
(PATTERN MOBSTAY P2 SQ)
(FUNCTION CANATTACK P2 SQ (LOC DP2 BD))
(FUNCTION NEQ DP2 DP1)
(FUNCTION SAFEP :2 (LOC OP2 BD))
(NEVER (PATTERN DIR P2 (LOC DP2 BD)))
(PLANNEW ((P2 SQ) ((NIL (KILL P2 DP2)) (NIL (KILL P1 DP1))))
        (THREAT (PLUS (EXCHVAL P2 SQ) (BOTH P1 DP1 P2 DP2))) (LIKELY 0)))

TERM
PIN1
((LP1 SQ DMP1 DMP2)
(FUNCTIONE PINAT LP1 SQ DMP1 DMP2)
(PATTERN MOBTERM LP1 SQ)
(NEVER (PATTERN DIR LP1 (LOC DMP1 BD)))
(IMPLIES NIL ((ASGREAT (VALUE LP1) (VALUE DMP1)))
 (NEVER (EXISTS (SQ1) (PATTERN DIR DMP2 SQ1)
        (FUNCTION CANATTACK DMP2 SQ1 (LOC DMP1 BD))
        (NEVER (FUNCTION CANATTACK LP1 SQ SQ1))
        (NEVER (FUNCTION OCCBYCOL SQ1 (COLOR DMP2)))
        (OR (PATTERN MOBSTAY DMP2 SQ1) (PATTERN ONLYPRO LP1 SQ1)))))
(PLANNEW ((LP1 SQ) (((DMP1 NIL) (KILL LP1 DMP2)) (NIL (KILL LP1 DMP1))))
 (THREAT (PLUS (FORK LP1 DMP1 DMP2) (EXCHVAL LP1 SQ))) (LIKELY 0)))

TERM
NIL
FORKPK

TERM
NIL
TRAPMATE

TERM
THR1
((P1 SQ1 P2 SQ2 DK)
(PATTERN TRAPMATE P1 SQ1 P2 SQ2 DK)
(NEVER (PATTERN MATE1 P1 SQ1 DK))
(FUNCTION CANATTACK P1 SQ1 (LOC DK BD))
(ACTIONNEW NONTERM (P1 SQ1) (THREAT (WIN DK))))

TERM
NIL
TWOCHECK

TERM
TC2
((P1 P2 DK SQ1 SQ2)
(PATTERN TWOCHECK P1 P2 DK SQ1 SQ2)
(NEVER (PATTERN MATE1 P1 SQ1 OK))
(IMPLIES (SQ) ((FUNCTIONE NEWFLIGHT SQ P1 SQ1 OK (LOC DK BD)))
  (IMPLIES (SQ3) ((FUNCTIONE NEWFLIGHT SQ3 P1 SQ1 DK SQ))
        (FUNCTION CANATTACK P2 SQ2 SQ3)))
(ACTIONNEW NONTERM (P1 SQ1) (THREAT (WIN OK))))
```

```
DEFSAFE
BLOCK
((P1 DP1 SQ SQ1)
(GOAL (NOBLOCK NIL))
(FUNCTION LINE (LOC P1 BD) SQ1 SQ)
(PATTERN MOBIL DP1 SQ1)
(ACTION DEFTHREAT (DP1 SQ1)))

DEFSAFE
BLOCKSAC
((P1 DP1 SQ SQ1)
(GOAL (NOBLOCK NIL))
(FUNCTION LINE (LOC P1 BD) SQ1 SQ)
(PATTERN LEGALMOVE DP1 SQ1)
(NEVER (PATTERN MOBIL DP1 SQ1))
(GREAT SAVE (EXCHLOSS DP1 SQ1))
(OR     (NEVER (PATTERN ONLYPRO DP1 SQ1))
        (EXISTS (DK) (FUNCTION EQUAL SQ (LOC DK BD))
          (PATTERN DIR DK (LOC DP1 BD))
          (IMPLIES (P2) ((PATTERN DIR P2 (LOC DP1 BD)))
           (PATTERN ONLYPRO P1 (LOC DP1 BD))
           (NEVER (FUNCTION CANATTACK P1 SQ1 (LOC DP1 BD)))))
        (EXISTS (DK LP1 SQ2) (FUNCTION EQUAL SQ (LOC DK BD))
          (PATTERN DIR DK SQ2) (NEVER (FUNCTION OCCBYCOL SQ2 (COLOR DK)))
          (NEVER (FUNCTION CANATTACK P1 SQ1 SQ2))
          (PATTERN ONLYDIR LP1 SQ2) (FUNCTION LINE (LOC LP1 BD) SQ1 SQ2)))
(ACTION DEFTHREAT (DP1 SQ1)))

DEFSAFE
CAPTUR
((P2 DP1 SQ P1)
(OR     (EXISTS (AP1) (PATTERN THRU P2 AP1 SQ) (PATTERN DIR AP1 SQ))
        (PATTERN DIR P2 SQ)
        (PATTERN ONLYPRO P1 (LOC P2 BD)))
(OR     (PATTERN MOBIL DP1 (LOC P2 BD))
        ((PATTERN LEGALMOVE DP1 (LOC P2 BD))
         (GREAT SAVE (EXCHLOSS DP1 (LOC P2 BD)))))
(ACTION DEFTHREAT (DP1 (LOC P2 BD))))

DEFSAFE
RUNP
((DP1 P1 SQ)
(OR     (PATTERN MOBSTAY P1 SQ)
        (EXISTS (DK) (FUNCTION EQUAL SQ (LOC DK BD)) (PATTERN DIR P1 SQ)))
(FUNCTION EQUAL SQ (LOC DP1 BD))
(ACTION RUN (DP1 P1 (LOC P1 BD))))

DEFSAFE
QUIT1
((P1 SQ DK)
(FUNCTION EQUAL SQ (LOC DK BD))
(RETURN))

DEFSAFE
MOBILOK
((DP1 SQ1 P1)
(OR     (PATTERN MOBIL DP1 SQ1)
((PATTERN LEGALMOVE DP1 SQ1)
 (PATTERN ONLYDIR P1 SQ1)
 (EXISTS (P2) (PATTERN ONLYPRO P1 (LOC P2 BD))
        (NEVER (FUNCTION CANATTACK P1 SQ1 (LOC P2 BD)))
        (ASGREAT (VALUE P2) (VALUE DP1))))))
```

```
DEFSAFE
COUNTER
((P1 SQ DP1 P2 DP2 SQ1)
(GOAL (NOBLOCK NIL))
(PATTERN MOBSTAY P1 (LOC DP1 BD))
(FUNCTION EQUAL SQ (LOC DP1 BD))
(EXISTS (SQ2) (PATTERN LEGALMOVE DP1 SQ2) (NEVER (PATTERN MOBIL DP1 SQ2))
 (UNIQUE (PATTERN DIR P2 SQ2) (FUNCTION NEQ P1 P2)))
(FUNCTION NEQ P1 P2)
(OR     (PATTERN MOBILOK DP2 SQ1 P1)
        ((PATTERN LEGALMOVE DP2 SQ1)
         (GREAT (EXCHVAL P1 (LOC DP1 BD)) (EXCHLOSS DP2 SQ1))
         (IMPLIES (P3) ((PATTERN DIR P3 SQ1))
           (OR (FUNCTION EQUAL P3 P2) (FUNCTION EQUAL P3 P1)))))
(FUNCTION CANATTACK DP2 SQ1 (LOC P2 BD))
(ASGREAT (EXCHVAL DP2 (LOC P2 BD)) (EXCHVAL P1 SQ))
(ACTION DEFTHREAT (DP2 SQ1)))

DEFSAFE
BLOCKSUP
((P2 DP1 SQ SQ1)
(OR     (EXISTS (DP1) (PATTERN ETHRU P2 DP1 SQ))
        (EXISTS (P3) (PATTERN OTHRU P2 P3 SQ)))
(FUNCTION LINE (LOC P2 BD) SQ1 SQ)
(PATTERN MOBIL DP1 SQ1)
(ACTION DEFTHREAT (DP1 SQ1) (LIKELY (ADD1 (GOAL1 LIKELY)))))

DEFSAFE
PROTECT
((DP1 SQ1 P1 SQ DK)
(NEVER (EXISTS (DMP1)
        (FUNCTION EQUAL DMP1 (OCCUPANT SQ BD))
        (GREAT (VALUE DMP1) (VALUE P1))))
(PATTERN MOBILOK DP1 SQ1 P1)
(FUNCTION NEQ SQ (LOC DP1 BD))
(NEVER (PATTERN DIR DP1 SQ))
(OR     (FUNCTION CANATTACK DP1 SQ1 SQ)
        (EXISTS (DLP1) (FUNCTION EQUAL DP1 DLP1)
          (FUNCTION CANATTACK DP1 SQ1 (LOC P1 BD))
          (FUNCTION LINE SQ1 (LOC P1 BD) SQ)))
(ACTION DEFTHREAT (DP1 SQ1)))

DEFSAFE
DISCOV
((DP1 DP2 SQ SQ1)
(PATTERN DSC DP1 DP2 SQ)
(PATTERN MOBIL DP2 SQ1)
(NEVER (FUNCTION LINE (LOC DP1 BD) SQ1 SQ))
(ACTION DEFTHREAT (DP2 SQ1)))
```

```
DEFSAFE
PINDEF
((P1 SQ DMP1 DMP2)
(GOAL (NOBLOCK NIL))
(PATTERN MOBSTAY P1 SQ)
(FUNCTION CANATTACK P1 SQ (LOC DMP1 BD))
(FUNCTION SAFEP P1 (LOC DMP1 BD))
(FUNCTION CANATTACK P1 (LOC DMP1 BD) (LOC DMP2 BD))
(FUNCTION LINE SQ (LOC DMP1 BD) (LOC DMP2 BD))
(FUNCTION SAFEP P1 (LOC DMP2 BD))
(ACTION RUN (DMP1 P1 SQ))
(ACTION RUN (DMP2 P1 (LOC DMP1 BD))))

DEFSAFE
FORKDEF
((P1 SQ DMP1)
(GOAL (NOBLOCK NIL))
(PATTERN MOBSTAY P1 SQ)
(FUNCTION CANATTACK P1 SQ (LOC DMP1 BD))
(FUNCTION SAFEP P1 (LOC DMP1 BD))
(NEVER (PATTERN DIR DMP1 SQ) (ASGREAT (VALUE P1) (VALUE DMP1)))
(EXISTS (DMP2)
        (FUNCTION CANATTACK P1 SQ (LOC DMP2 BD))
        (FUNCTION NEQ DMP1 DMP2)
        (FUNCTION SAFEP P1 (LOC DMP2 BD))
        (NEVER (PATTERN DIR DMP2 SQ) (ASGREAT (VALUE P1) (VALUE DMP2))))
(ACTION RUN (DMP1 P1 SQ)))

DEFSAFE
NORECAP
((P1 SQ DMP1)
(GOAL (NOBLOCK NIL))
(PATTERN DIR P1 SQ)
(FUNCTION OCCBYCOL SQ (COLOR P1))
(FUNCTION CANATTACK P1 SQ (LOC DMP1 BD))
(FUNCTION SAFEP P1 (LOC DMP1 BD))
(NEVER (PATTERN DIR DMP1 SQ) (ASGREAT (VALUE P1) (VALUE DMP1)))
(ACTION RUN (DMP1 P1 SQ)))

DEFSAFE
BLKCHK
((P1 SQ DK DP1 SQ1)
(OR (PATTERN CHECKMOVE P1 SQ) (NEVER (GOAL (NOBLOCK NIL))))
(PATTERN MOBILOK DP1 SQ1 P1)
(FUNCTION LINE SQ SQ1 (LOC DK BD)))

DEFSAFE
BC1
((P1 SQ DK DP1 SQ1 P2)
(PATTERN BLKCHK P1 SQ DK DP1 SQ1)
(PATTERN ENPRIS P2 DP1)
(ACTION DEFTHREAT (DP1 SQ1) (SAVE (PLUS 3 (GOAL1 SAVE)))))

DEFSAFE
BC2
((P1 SQ DK DP1 SQ1 P2)
(PATTERN BLKCHK P1 SQ DK DP1 SQ1)
(NEVER (PATTERN BC1 P1 SQ DK DP1 SQ1 P2))
(ACTION DEFTHREAT (DP1 SQ1)))
```

```
DEFSAFE
RUNCHK
((P1 SQ DK SQ1)
(OR (PATTERN CHECKMOVE P1 SQ) (NEVER (GOAL (NOBLOCK NIL))))
(PATTERN LEGALMOVE DK SQ1)
(NEVER (FUNCTION CANATTACK P1 SQ SQ1))
(NEVER (FUNCTION LINE SQ (LOC DK BD) SQ1))
(ACTION DEFTHREAT (DK SQ1)))

DEFSAFE
NPINCHK
((P1 SQ DK SQ1 DSP1 LP1)
(OR (PATTERN CHECKMOVE P1 SQ) (NEVER (GOAL (NOBLOCK NIL))))
(PATTERN MOBSTAY P1 SQ)
(PATTERN DIR DSP1 SQ)
(PATTERN PIN LP1 DSP1 DK)
(PATTERN LEGALMOVE DK SQ1)
(NEVER (FUNCTION LINE (LOC OSP1 BD) (LOC DK BD) SQ1))
(ACTION DEFTHREAT (DK SQ1)))

DEFSAFE
FLYCHK
((P1 SQ DK SQ1 DP1 SQ2)
(OR (PATTERN CHECKMDVE P1 SQ) (NEVER (GOAL (NOBLOCK NIL))))
(PATTERN DIR DK SQ1)
(FUNCTION EQUAL DP1 (OCCUPANT SQ1 BD))
(NEVER (FUNCTION CANATTACK P1 SQ SQ1))
(OR     (PATTERN MOBILOK OP1 SQ2 P1)
        ((IMPLIES (SQ3) ((PATTERN MOBILOK DP1 SQ3 P1))
           (FUNCTION CANATTACK DK SQ1 SQ3)
           (OR  (FUNCTION SAFEP DK SQ3)
                ((PATTERN ONLYPRO P1 SQ3) (NEVER (FUNCTION CANATTACK P1 SQ SQ3)))))
         (PATTERN LEGALMOVE DP1 SQ2)
         (GREAT SAVE (VALUE DP1))))
(IMPLIES (P2)
 ((OR   ((PATTERN DIR P2 SQ1) (FUNCTION NEQ P2 P1))
        (PATTERN DSC P2 P1 SQ1)
        (PATTERN DTHRU P2 P1 SQ1)))
 (OR    (FUNCTION LINE (LOC P2 BD) SQ2 SQ1)
        (FUNCTION EQUAL SQ2 (LOC P2 BD))))
(ACTION DEFTHREAT (DP1 SQ2)))

DEFSAFE
DESPOOCHK
((DP1 MP1)
(GOAL (NOBLOCK NIL))
(PATTERN CHECKMOVE DP1 (LOC MP1 BD))
(NEVER (PATTERN MOBIL DP1 (LOC MP1 BD)))
(EXISTS (P1) (PATTERN MOBSTAY P1 (LOC DP1 BD)))
(GREAT SAVE (EXCHLOSS DP1 (LOC MP1 BD)))
(ACTION DEFTHREAT (DP1 (LOC MP1 BD))))



DEFTHREAT
NOTGOOD
```

```
((DP1 SQ1)
(NEVER (CHECKCOND COND))
(RETURN))
```

```
DEFTHREAT
LOSE
((DP1 SQ1)
(NEVER (PATTERN CHECKMOVE DP1 SQ1))
(EXISTS (P1 DP2) (PATTERN ENPRIS P1 DP2)
        (FUNCTION NEQ DP1 DP2)
        (FUNCTION NEQ SQ1 (LOC P1 BD))
        (FUNCTION CANATTACK P1 (LOC DP2 BD) SQ1)
        (NEVER (FUNCTION CANATTACK DP1 SQ1 (LOC DP2 BD)))
        (NEVER (FUNCTION LINE (LOC P1 BD) SQ1 (LOC DP2 BD)))
        (NEVER (PATTERN DIR P1 SQ1)))
(PLAN ((DP1 SQ1) NIL (GOAL PLAN)) (PLAN)
  (THREAT (PLUS (GOAL1 THREAT)  (LESS (EXCHVAL DP1 SQ1) 1))))
(RETURN))

DEFTHREAT
PRTCT
((DP1 SQ1)
(EXISTS (P2 DP2) (PATTERN ENPRIS P2 DP2)
        (FUNCTION NEQ DP1 DP2)
        (NEVER (PATTERN DIR DP1 (LOC DP2 BD)))
        (FUNCTION CANATTACK OP1 SQ1 (LOC DP2 BD))
        (IMPLIES (P1) ((PATTERN ENPRIS P1 DP2) (FUNCTION NEQ P1 P2)
          (OR   (NEVER (EXISTS (DP3) (PATTERN DIR DP3 (LOC DP2 BD))))
                (EXISTS (DP3) (PATTERN ONLYDIR DP3 (LOC DP2 BD))
                  (FUNCTION LINE (LOC OP3 BD) SQ1 (LOC DP2 BD)))))
         (GREAT (PLUS (VALUE P1) (VALUE P2)) (PLUS (VALUE DP2) (VALUE DP1)))))
(PLAN ((OP1 SQ1) NIL (GOAL PLAN)) (PLAN)
 (THREAT (PLUS (GOAL1 THREAT)  (EXCHVAL DP1 SQ1)))  (SAVE (PLUS 2 (GOAL1 SAVE)))))

DEFTHREAT
DEFCAP
((DP1 SQ1)
(FUNCTION OCCBYCOL SQ1 (COMPCOLOR DP1))
(GREAT (EXCHVAL DP1 SQ1) -1)
(PLAN ((DP1 SQ1) NIL (GOAL PLAN)) (PLAN)
  (THREAT (PLUS (GOAL1 THREAT)  (PLUS 5 (EXCHVAL DP1 SQ1)))))
(RETURN))

OEFTHREAT
OEFCK
((DP1 SQ1)
(PATTERN CHECKMOVE DP1 SQ1)
(PLAN ((DP1 SQ1) NIL (GOAL PLAN)) (PLAN)
  (THREAT (PLUS (GOAL1 THREAT)  (PLUS 3 (EXCHVAL DP1 SQ1)))))
(RETURN))
```

```
DEFTHREAT
DEFATT
((DP1 SQ1 P2)
(FUNCTION CANATTACK DP1 SQ1 (LOC P2 BD))
(FUNCTION SAFEP DP1 (LOC P2 BD))
(IMPLIES (DP2) ((PATTERN ENPRIS P2 DP2) (FUNCTION NEQ DP1 DP2))
        (FUNCTION CANATTACK DP1 SQ1 (LOC DP2 BD))
        (IMPLIE‿ (P1) ((PATTERN ENPRIS P1 DP2) (FUNCTION NEQ P1 P2))
          (GREAT (VALUE P2) (VALUE DP1))))
(PLAN ((DP1 SQ1) NIL (GOAL PLAN)) (PLAN)
  (THREAT (PLUS (GOAL1 THREAT)  (PLUS 2 (EXCHVAL DP1 SQ1)))))
(RETURN))

DEFTHREAT
ELSE
((DP1 SQ1)
(PLAN ((DP1 SQ1) NIL (GOAL PLAN)) (PLAN)
  (THREAT (PLUS (GOAL1 THREAT)  (EXCHVAL DP1 SQ1)))))
```

```
RUN
MOBOK
((DP1 SQ P1 SQ1)
(PATTERN MOBIL DP1 SQ)
(OR     (NEVER  (FUNCTION CANATTACK P1 SQ1 SQ))
        ((PATTERN ENPRIS P1 OP1) (PATTERN ONLYPRO P1 SQ)))
(NEVER  (FUNCTION LINE SQ1 (LOC DP1 BD) SQ))
(NEVER  (FUNCTION LINE (LOC P1 BD) (LOC DP1 BD) SQ))
(ACTION DEFTHREAT (DP1 SQ)))

RUN
CAP
((DP1 P2 P1 SQ1)
(PATTERN MOBIL DP1 (LOC P2 BD))
(NEVER (PATTERN MOBOK DP1 (LOC P2 BD) P1 SQ1))
(ACTION DEFTHREAT (DP1 (LOC P2 BD))))

RUN
DEFAULT1
((OP1 SQ)
(FUNCTIONE DIRMOSTMOB DP1 SQ)
(PLAN ((DP1 SQ)) (SAVE (PLUS 1 (GOAL1 SAVE)))))

RUN
KRUN
((DP1 SQ DK)
(FUNCTION EQUAL DP1 DK)
(PATTERN LEGALMOVE DP1 SQ)
(PLAN ((DP1 SQ))))

RUN
DESPER
((DP1 P1 P2)
(PATTERN ENPRIS P1 DP1)
(ASGREAT (EXCHVAL P1 (LOC DP1 BD)) (VALUE OP1))
(NEVER (EXISTS (SQ) (PATTERN MOBIL DP1 SQ)))
(PATTERN LEGALMOVE DP1 (LOC P2 BD))
(NEVER (EXISTS (P3) (PATTERN LEGALMOVE DP1 (LOC P3 BD))
  (GREAT (VALUE P3) (VALUE P2))))
(PLAN ((DP1 (LOC P2 BD))) (THREAT (EXCH DP1 P2))))


SAVE
SAVEMATE
((DP1 SQ K P1 DP2)
(GREAT THREAT (VALUE K))
(PATTERN MOBSTAY P1 (LOC DP2 BD))
(EXISTS (SQ1) (PATTERN DIR K SQ1)
 (NEVER (FUNCTION OCCBYCOL SQ1 (COLOR K)))
 (PATTERN ONLYDIR DP2 SQ1)
 (NEVER (FUNCTION CANATTACK DP1 SQ SQ1))
 (NEVER (EXISTS (DP2) (PATTERN THRU DP2 DP1 SQ1))))
(FINALPLAN ((P1 (LOC DP2 BD))) (LIKELY 0) (THREAT (EXCH P1 DP2))))
```

```
DEFENSE
NIL
NOMOVE

DEFENSE
HELPCAP
((P1 SQ DP1 P2 DP2)
(FIRSTMOVE P1 SQ)
(FUNCTION EQUAL SQ (LOC DP1 BD))
(GREAT (VALUE DP1) (EXCHVAL P1 SQ))
(OR      (PATTERN ENPRIS P2 DP2)
        ((PATTERN DIR P2 (LOC DP2 BD)) (FUNCTION SAFEP P2 (LOC DP2 BD))
          (NEVER (FUNCTION EXCHANGE P2 (LOC DP2 BD)))))
(FUNCTION NEQ P1 P2)
(FUNCTION NEQ DP1 DP2)
(IMPLIES (DP3)
 ((OR (PATTERN DIR DP3 SQ) (PATTERN ETHRU DP3 P1 SQ)))
  (FUNCTION NEQ DP3 DP2)
  (NEVER (EXISTS (K) (FUNCTION CANATTACK DP3 SQ (LOC K BD))))
  (OR (NEVER (FUNCTION CANATTACK DP3 SQ (LOC DP2 BD)))
      (GREAT (VALUE DP2) (VALUE P2))))
(FINALPLAN GOAL
 (THREAT (FORKT (WIN DP1) (PLUS (EXCH P1 DP1) (EXCH P2 DP2)))))
(RETURN))

DEFENSE
ELSE1
((P1 SQ)
(FIRSTMOVE P1 SQ)
(FINALPLAN GOAL))
```

```
DEFEP
NIL
NOMOVE

DEFEP
NIL
HELPCAP

DEFEP
ALLOWFRK
((P1 SQ DP1 SQ1 P2)
(FIRSTMOVE P1 SQ)
(OR     (PATTERN MOBSTAY DP1 SQ1)
        ((PATTERN LEGALMOVE DP1 SQ1) (PATTERN ONLYPRO P1 SQ1)
         (NEVER (FUNCTION CANATTACK P1 SQ SQ1))))
(FUNCTION NEQ DP1 (OCCUPANT SQ BD))
(IMPLIES NIL  ((FUNCTION EQUAL SQ1 (LOC P1 BD)))
        (PATTERN ENPRIS DP1 P1) (NEVER (PATTERN ONLYPRO DP1 SQ1)))
(FUNCTION CANATTACK DP1 SQ1 SQ)
(GREAT (VALUE P1) (VALUE DP1))
(FUNCTION CANATTACK DP1 SQ1 (LOC P2 BD))
(FUNCTION SAFEP DP1 (LOC P2 BD))
(NEVER (PATTERN DIR P2 SQ1) (ASGREAT (VALUE DP1) (VALUE P2)))
(NEVER (PATTERN LEGALMOVE DP1 (LOC P2 BD)))
(FINALPLAN GOAL
 (LOSS (PLUS (GOAL1 LOSS) (FORKT (LESS (WIN P1) (WIN DP1))
        (EXCH DP1 P2)))))
(RETURN))

DEFEP
NIL
ELSE1
```

```
DEFOFF
NIL
BLOCK

DEFOFF
NIL
CAPTUR

DEFOFF
RUNPOFF
((DP1 P1 SQ SQ1)
(FUNCTION EQUAL SQ (LOC DP1 BD))
(PATTERN MOBSTAY P1 (LOC DP1 BD))
(PATTERN MOBIL DP1 SQ1)
(NEVER (FUNCTION LINE SQ1 (LOC DP1 BD) (LOC P1 BD)))
(NEVER (FUNCTION CANATTACK P1 SQ SQ1))
(NEVER (EXISTS (DP2 P2) (PATTERN ONLYPRO DP1 (LOC DP2 BD))
        (NEVER (FUNCTION CANATTACK DP1 SQ1 (LOC DP2 BD)))
        (PATTERN LEGALMOVE P2 (LOC DP2 BD))))
(ACTION DEFTHREAT (DP1 SQ1)))

DEFOFF
NIL
QUIT1

DEFOFF
NIL
PROTECT

DEFOFF
NIL
RUNCHK

DEFOFF
NIL
BLKCHK

DEFOFF
NIL
BC1

DEFOFF
NIL
BC2

EPDESPO
EPDESP
((P1 DP1)
(PATTERN ENPRIS P1 DP1)
(NEVER (EXISTS (DP2 P2) (PATTERN PIN DP2 P1 P2)))
(NEVER (EXISTS (DP2 P2 SQ) (PATTERN PROTECTS P1 P2 SQ DP2)))
(FINALPLAN ((P1 (LOC DP1 BD))) (THREAT (EXCH P1 DP1)) (LIKELY 0)))

EPDESPO
MCHK
((P1 SQ)
(PATTERN CHECKMOVE P1 SQ)
(PATTERN MOBIL P1 SQ)
(FUNCTION OCCBYCOL SQ (COMPCOLOR P1))
(FINALPLAN ((P1 SQ)) (SAVE 10) (LIKELY 0)))
```

```
PLAN1
NOMOVE
((P1 SQ)
(NEVER (FIRSTMOVE P1 SQ))
(FINALPLAN GOAL)  (RETURN))

PLAN1
CHECK
((P1 SQ DK)
(FIRSTMOVE P1 SQ)
(PATTERN CHECKMOVE P1 SQ)
(FINALPLAN GOAL (PLAN1))
(RETURN))

PLAN1
FMATE1
((P1 SQ P2 SQ1 DK)
(FIRSTMOVE P1 SQ)
(PATTERN MATE P2 SQ1)
(NEVER (FUNCTION LINE (LOC P2 BD) SQ SQ1))
(IMPLIES (SQ2)  ((PATTERN DIR DK SQ2) (NEVER (FUNCTION OCCBYCOL SQ2 (COLOR DK))))
  (OR (NEVER (PATTERN ONLYPRO P1 SQ2)) (FUNCTION CANATTACK P2 SQ1 SQ2)
        (FUNCTION CANATTACK P1 SQ SQ2))))

PLAN1
FMATE2
((P1 SQ P2 SQ1 DK)
(PATTERN FMATE1 P1 SQ P2 SQ1 DK)
(PATTERN ONLYDIR DK SQ1)
(IMPLIES (DP1) ((PATTERN ENPRIS DP1 P2))  (FUNCTION EQUAL DP1 (OCCUPANT SQ BD)))
(FINALPLAN GOAL (PLAN1))
(RETURN))
```

```
PLAN1
GIVE1
((P1 SQ P2 SQ1 DP1)
(FIRSTMOVE P1 SQ)
(OR (PATTERN PIN DP1 P1 P2)
    ((PATTERN ENPRIS OP1 P2) (FUNCTION NEQ P1 P2)
     (FUNCTION NEQ OP1 (OCCUPANT SQ BO))
     (OR (NEVER (OR (PATTERN DIR (OCCUPANT SQ BD) (LOC P2 BO))
                    (PATTERN OTHRU (OCCUPANT SQ 8D) DP1 (LOC P2 BO))))
         ((NEVER (EXISTS (P3) (PATTERN DIR P3 (LOC P2 BO))))
          (NEVER (FUNCTION LINE SQ (LOC OP1 BD) (LOC P2 BD))
                 (FUNCTION CANATTACK P1 SQ (LOC OP1 BO)))
          (NEVER (FUNCTION CANATTACK P1 SQ (LOC P2 BD)))))
     (IMPLIES NIL ((ASGREAT (VALUE OP1) (VALUE P2)))
        (NEVER (FUNCTION CANATTACK P1 SQ (LOC P2 BO))))
     (NEVER     (FUNCTION CANATTACK P1 SQ (LOC P2 BD))
                (PATTERN ONLYPRO DP1 (LOC P2 BO))
        (EXISTS (DMP1 OMP2) (FUNCTION CANATTACK P1 (LOC P2 BO) (LOC OMP1 BD))
                (FUNCTION CANATTACK P1 (LOC P2 BD) (LOC DMP2 BO))
                (FUNCTION NEQ DMP1 DMP2)
                (ASGREAT (EXCHVAL P1 (LOC OMP1 BD)) (EXCHVAL P1 (LOC DMP2 BO)))
                (ASGREAT (PLUS (VALUE OP1) (PLUS (VALUE (OCCUPANT SQ BO))
                        (EXCHVAL P1 (LOC DMP2 BD)))) (VALUE P2))))))
(FUNCTION EQUAL SQ1 (LOC P2 BO))
(NEVER (FUNCTION LINE SQ1 SQ (LOC DP1 BD)))
(NEVER (FUNCTION EQUAL SQ (LOC OP1 BD)))
(NEVER (EXISTS (DP2) (FUNCTION EQUAL DP2 (OCCUPANT SQ BD))
                (PATTERN PROTECTS DP1 DP2 SQ P1)
                (ASGREAT (VALUE DP2) (EXCHVAL DP1 SQ1))))
(NEVER (EXISTS (DP2)
 (OR ((FUNCTION CANATTACK P1 SQ (LOC OP2 BD)) (FUNCTION NEQ DP2 DP1))
     (FUNCTION EQUAL DP2 (OCCUPANT SQ BD))
     ((FUNCTION CANATTACK P1 SQ (LOC OP1 BD))
        (FUNCTION LINE SQ (LOC DP1 BD) (LOC DP2 BD))
        (FUNCTION CANATTACK P1 (LOC DP1 BD) (LOC OP2 BD))))
  (ASGREAT (EXCHVAL P1 (LOC DP2 BD)) (EXCHVAL DP1 SQ1)))))
```

```
PLAN1
GIVE2
((P1 SQ P2 SQ1 DP1)
(FIRSTMOVE P1 SQ)
(PATTERN PROTECTS P1 P2 SQ1 DP1)
(NEVER (EXISTS (K) (PATTERN OIR DP1 (LOC K BD))
        (FUNCTION LINE (LOC DP1 BD) SQ1 (LOC K BD))))
(NEVER (FUNCTION CANATTACK P1 SQ (LOC P2 BD)))
(IMPLIES (LP1) ((FUNCTION EQUAL P1 LP1))
        (NEVER (FUNCTION LINE SQ (LOC P1 BD) SQ1)))
(NEVER (FUNCTION LINE SQ1 SQ (LOC DP1 BD)))
(NEVER (FUNCTION EQUAL SQ (LOC DP1 BD)))
(NEVER (EXISTS (DP2) (FUNCTION EQUAL DP2 (OCCUPANT SQ BD))
        (PATTERN PROTECTS DP1 DP2 SQ P1)
        (ASGREAT (VALUE DP2) (VALUE P2))))
(NEVER (EXISTS (DP2)
 (OR ((FUNCTION CANATTACK P1 SQ (LOC DP2 BD)) (FUNCTION NEQ DP2 DP1))
     ((FUNCTION CANATTACK P1 SQ (LOC DP1 BD))
        (FUNCTION LINE SQ (LOC DP1 BD) (LOC DP2 BD))
        (FUNCTION CANATTACK P1 (LOC DP1 BD) (LOC DP2 BD))))
  (ASGREAT (EXCHVAL P1 (LOC DP2 BD)) (VALUE P2))
  (IMPLIES NIL ((FUNCTION CANATTACK DP1 SQ1 (LOC DP2 BD))
        (ASGREAT (VALUE DP2) (PLUS (VALUE P2) (VALUE P1)))))))

PLAN1
GIVEUP
((P1 SQ P2 SQ1 DP1)
(OR (PATTERN GIVE1 P1 SQ P2 SQ1 DP1)
    (PATTERN GIVE2 P1 SQ P2 SQ1 DP1))
(IMPLIES (P3 SQ2 DK) ((PATTERN FMATE1 P1 SQ P3 SQ2 DK) (PATTERN DIR DP1 SQ2))
  (OR   (FUNCTION CANATTACK DP1 SQ1 SQ2)
        (FUNCTION OVERPRO (COLOR P1) SQ2))))

PLAN1
PLANOK
((P1 SQ)
(FIRSTMOVE P1 SQ)
(NEVER (EXISTS (P2 DP1 SQ1) (PATTERN GIVEUP P1 SQ P2 SQ1 DP1)))
(FINALPLAN GOAL (PLAN1))
(RETURN))
```

```
PLAN1
THREATOK
((P1 SQ)
(FIRSTMOVE P1 SQ)
(SET ATR2 (DP1) (((FUNCTION CANATTACK P1 SQ (LOC DP1 BD))
        (FUNCTION SAFEP P1 (LOC DP1 BD))
        (NEVER (EXISTS (P2 SQ1) (PATTERN GIVEUP P1 SQ P2 SQ1 DP1))))
    (EXCH P1 DP1))
 (((FUNCTION CANATTACK P1 SQ (LOC DP1 BD))
    (PATTERN ONLYPRO (OCCUPANT SQ BD) (LOC DP1 BD))
    (NEVER (EXISTS (P2 SQ1) (PATTERN GIVEUP P1 SQ P2 SQ1 DP1))))
   (WIN DP1))
 (((FUNCTION EQUAL DP1 (OCCUPANT SQ BD)))
    (LESS (WIN DP1) (EXCHVAL P1 SQ))))
(IMPLIES (P2 DP1 SQ1) ((PATTERN GIVEUP P1 SQ P2 SQ1 DP1))
 (OR
  ((PATTERN ONLYPRO P1 SQ1) (NEVER (FUNCTION CANATTACK P1 SQ SQ1))
   (OR  (GREAT THREAT (DIF (VALUE P2) ATR2))
        (GREAT SAVE (DIF (VALUE P2) ATR2))))
  ((OR (FUNCTION CANATTACK P1 SQ SQ1) (NEVER (PATTERN ONLYPRO P1 SQ1)))
   (OR  (GREAT THREAT (DIF (EXCHVAL DP1 (LOC P2 BD)) ATR2))
        (GREAT (VALUE (OCCUPANT SQ BD)) (DIF (EXCHVAL DP1 (LOC P2 BD)) ATR2))
        (GREAT SAVE (DIF (EXCHVAL DP1 (LOC P2 BD)) ATR2))))))
(SET ATR1 (P2 DP1 SQ1)
 (((PATTERN GIVEUP P1 SQ P2 SQ1 DP1)) (EXCH DP1 P2))
 (((PATTERN GIVEUP P1 SQ P2 SQ1 DP1) (PATTERN ONLYPRO P1 SQ1)
        (NEVER (FUNCTION CANATTACK P1 SQ SQ1)))
 (WIN P2)))
(FINALPLAN GOAL  (PLAN1)  (LOSS (LESS ATR1 ATR2))))
```

```
PLAN1
CAPFIX
((P1 SQ P3 DP1)
(EXISTS (MP1 SQ1) (PATTERN GIVEUP P1 SQ MP1 SQ1 DP1))
(PATTERN MOBIL P3 (LOC DP1 BD))
(FUNCTION NEQ P1 P3)
(FUNCTION NOTINGOAL P3)
(PIECESAFE P1 (EXCHVAL P3 (LOC DP1 BD)))
(GREAT TOPTHREAT (PLUS 5 (EXCHVAL P3 (LOC DP1 BD))))
(SET ATR1 (P2 DP2 SQ2)
 (((PATTERN GIVEUP P1 SQ P2 SQ2 DP2)
        (NEVER (PATTERN CHECKMOVE P3 (LOC DP1 BD)))
        (FUNCTION NEQ DP1 DP2)  (FUNCTION NEQ P2 P3))
 (EXCH DP2 P2)))
(FINALPLAN (FIX GOAL ((P3 (LOC DP1 BD)) NIL)) (LOSS ATR1) (PLAN1)
  (THREAT (FORKT (GOAL1 THREAT) (WIN DP1)))))

PLAN1
CAPDECOY
((P1 SQ P3 DP1 DP2 DMP1 SQ1)
(EXISTS (MP1 SQ2) (PATTERN GIVEUP P1 SQ MP1 SQ2 DP1))
(OR     (GOAL1 (THREAT (FORK P1 DP1 DP2)))
        (GOAL1 (THREAT (FORK P1 DP2 DP1))))
(PATTERN PROTECTS DMP1 DP1 SQ1 P3)
(FUNCTION NEQ P1 P3)
(FUNCTION NEQ DMP1 DP2)
(PIECESAFE P1 (VALUE DMP1))
(PIECESAFE P1 (VALUE DP2))
(GREAT (VALUE DMP1) (EXCHLOSS P3 (LOC DP1 BD)))
(GREAT (VALUE DP2) (EXCHLOSS P3 (LOC DP1 BD)))
(SET ATR1 (P2 DP3 SQ2)
 (((PATTERN GIVEUP P1 SQ P2 SQ2 DP3)
        (FUNCTION NEQ DP1 DP3)
        (FUNCTION NEQ DP3 DMP1)
        (FUNCTION NEQ P2 P3))
 (EXCH DP2 P2)))
(FINALPLAN ((P3 (LOC DP1 BD)) (DMP1 (LOC DP1 BD)) (P1 SQ)
   (((ANYBUT DMP1) (KILL P1 DMP1)) ((ANYBUT DP2) (KILL P1 DP2))))
  (PLAN1) (LOSS ATR1)  (THREAT (PLUS (FORK P1 DP2 DMP1) (EXCH P3 DP1))))
(RETURN))

PLAN1
REJECT
((P1 SQ P2 SQ1 DP3)
(NEVER (PATTERN THREATOK P1 SQ))
(NEVER (PATTERN CAPFIX P1 SQ P2 DP3))
(PATTERN GIVEUP P1 SQ P2 SQ1 DP3)
(NEVER (EXISTS (P3 SQ2 DP2) (PATTERN GIVEUP P1 SQ P3 SQ2 DP2)
        (OR (FUNCTION NEQ DP2 DP3) (FUNCTION NEQ SQ1 SQ2))
        (GREAT (EXCHVAL DP2 (LOC P3 BD)) (EXCHVAL DP3 (LOC P2 BD)))))
(REMATRS T ((LIKELY 0)))
(NEVER (GOAL1 (PLAN1 NIL)))
(ACTIONNEW DEFOFF (DP3 SQ1) (LIKELY 1) (SAVE 0)
  (COND (NEVER (FUNCTION EQUAL DP1 P1)))
  (THREAT (GOAL THREAT)) (PLAN (GOALP PLAN))))

NIL
```

APPENDIX B

Working Notes on Patterns

most things straightforward. PIN tries to assure that something will be lost if DP1 moved.

DSCTHR
denotes uncovered moves that threaten discovered attacks. Knows that discovered attack is valid and that uncovering move cannot be trivially captured by piece being attacing in dsc attack.

MOBIL: means a square can be landed on without loss (of piece that moved). These 2 were here but giving up functions must be reasoned about separately so they have been removed.
  1) don't let pinned piece move if it loses. Handling of pins not exact. Comparing piece being captured to pin object to decide if moving pinned piece is ok could be tempered with comparing the expected loss or gain on both squares although this would not be exact either.
  2) does not ok move which gives up only protection of a piece that is attacked unless the move wins at least that much.
MOBIL includes obvious discovered moves. Otherwise unsafe move is approved if it has dscthr square to move to and the piece being threatened is as valuable as the piece being sacrificed and the opponent cannot exchange the attacking piece with less loss than we would suffer from sacrificed piece. Here we give sacrificed piece credit for things it attacks but don't count attacking pieces which can capture our attacking piece. (Perhaps should put other requirements on piece we attack, such as: don't allow it if it can move to a non-overprotected sq from which it protects the dsc attack object (if such move not blocked by P1 to SQ)).
(use of attack: pos 27 after RxR, QxR, R-F1)

MOBSTAY: means sqare can be landed on and opponent must lose to capture the piece.
  This is not true when an exchange of two pieces will keep material equal.
  If you want to take care of that case, add this to mobstay:
```
(IMPLIES (DP1) ((FUNCTION EQUAL DP1 (OCCUPANT SQ BD)))
        (NEVER (FUNCTION SAFEP DP1 SQ)))
```
  That not added now cuz two piece exchs leave opportunities open (I think: no documented cases of needing it).
  Not having that in has not hurt at all except occassionally when we are trying to mate and don't want mating piece exchanged. In that case we check above line in mating production.

PROTECTS
Recognizes captures and mate moves.
DP1 threatens to win P2 by landing on SQ1 (P2's loc except when mating).
Only P1 prevents it. For capture, requires only P1 bears dir on sq1.
Actually should recognize situation when more than one piece bears on square but all are needed -- should be easy (not overpro) but this seems to be working fine so leave it. (ok if some piece bear othru p1 to sq1)

For mate want to know 1) protects matesq and no safe interpose, or 2) no protection of matesq and this only safe interpose.
Interposing trip is hard cuz want to know that square safe to land on even after opponent has made checking move. Thus must be overprotected now which implies we can move there. As an approximation say P1 necessary for interposition if it can do MOBIL move to interposing square. It is not necessary if there are at least two more pieces bearing on the square but no way to test this so let's approximate.

THREAT EP DSC FORK

this takes COL as an argument and looks for threats for that side.
Assumes nothing from GOAL in atrs.
It posits PLANs, MOVE goals, and ATTACK goals.

nb: NOT DIR P1 SQ3 does not instantiate unless it is true for some instantiations, and then only instans first var.

ENPRISE: uses MOBSTAY to avoid suggestting exchanges for their own sake.

EP2,P,C: gives credit for 2 eps when first doesn't affect second and/or first checks. Needed primarily for epeval. No known place where ep2c used, but ep2p works in 78 (i think - somewhere).

DSCATT: the threat of capturing a piece by uncovering a dsc attack at the same time will be recognized as en prise since MOBIL handles that. So here recognize fork make by moving uncovering piece to attack another.

FORKSQ: FUNCTIONE instantiates thing in reasonable time. COND atr given so that move goal will not generate stupid attempts at two move forks. Safecond prevents decoying of fork objects. FORKSQ win recognizes special situation and gives it optimistic threat. (Possibly very over optimistic, see comments on TERM regarding need conds in threat) Special situation only recognized for 1 move forks since specialness will probably go away in two moves and don't want to generate too much overoptimistic bullshit.

FORKSQW: recognizing winning situation when forking move captures only piece protecting fork object. Since exchval is still 0 or less special prods were put in to give more optimistic threat for this situation . (used somewhere in tree of 78)

FORKSAC: takes DMP1 with P2 to get DP1 to recapture, then forks DP1 and DMP2 with SP1. Make sure that what we threaten to win is worth giving up P2 for.

CANTRAP: checks if we canattack piece and take away his escapes at same time. Checks for can attack move on piece not already attacked and not trapped that we can safely capture. To assure no escape, checks that for all mobil moves of that piece that either 1) moving thru line we're going to attack along, 2) sq attacked by someone else thru piece we're moving, or 3) attacking piece can attack it from new square. An attack here also requires that square should not be overpro against us unless opponent piece more valuable. Lastly, check that no obvious escapes uncovered: not give up onlypro of sq he can move to, and no have him bearing thru out attacking piece to safe squares (OBJ used since P1 not bear dir on DMP1). Threat should be exchval on dmp1 since he should be able to just sit there and exchange. This may let in too many attacks in cases where P1 is more valuable than DMP1, perhaps should require that not be true. (does 42)

THREAT PIN DECOY

PINATT: obvious
  Since this happens a lot and is not very threatening, only MOVE1 attacks are
  tried. To do 2 move attacks, just change action to MOVE and put this in atrl:
 (COND (NEVER (FUNCTION LINE SQ2 (LOC DMP1 BD) (LOC DMP2 BD))))

DECOYK (old)
~Tries sacrificial moves to squares which only K protects when 1)
~move either checks king, captures piece, or guards one king escape square;
~and 2)it is possible to attack the sacrificial square by some means on the
~next move. (If K does not protect, then move should be safe.)
SACHECK
~This one like DECOYK except allows sacrifice when one piece other than K
~protects (whether K does or not), as long as condition 2) is replaced by:
~there is piece that can recapture with check safely.
~(perhaps requiring safe check is too restrictive; could also allow safely
~guarding check squares)
~~Has been removed since made bad plans and no good ones:

DECOYK (new)
The old one made too many ridiculous sacs, so use these:
POSDECOY
matches if only DK protects SQ and P1 can move there unsafely and P2 can
attack SQ from SQ1 on next move. Well:
First exists P3 SQ2 is just to make sure some other piece than the sacrificed one
is in the general vicinity. (prevents stupid giveaways)
Or has 3 parts for testing attack: first is discovered attack, second is
direct attack, and third is discovered attack that also goes thru sacrificed
piece also. (Direct covers case when would be thru sacrificed piece.)
In discovered ones could require that uncovering move must not be attackable by
king from SQ unless some offen piece besides P1 or P2 bears on it.
(Leave out for now since no errors from this.)
Works: pos 4, pos 9 at ply 5, pos 78 for QxN.

DECOYK1
Says to go ahead with sac if DK on SQ and P2 on SQ1 looks like a matesq
(ie one or fewer moves for king). Previously only checked if some piece
other than P2 was bearing in the area since didn't have power then to
test this matesq kind of stuff. The likely is 1 from previous experience.
Even though the sac forces a reply, the plan almost never worked since
it usually just threw a piece away. Perhaps 0 should be retried when
using this matesq test since plan much more likely too work, although the
matesq calculation assumes P1 is still around when actually it's been
sacrificed.

THREAT decoyk2

DK-MOBIL
DECOYK2
This invented while doing pos 49 and 55. Basically lots of invisible mating nets available after perturbing effects of a posdecoy. (They work in likely 1 situations). Initial idea to cover these types of positions is do it if we can attack king every place he can move to after second check. But as 55 shows, a two move attack can be enough in likely 1 situations, so allow that to. 49 Shows that attacking move need not be safe if the is safe move to make it safe in likely 1 situations (this is not checked since it gets very hairy to determine safety and hairy to know what attacks what since 2 pieces gone, instead we call squares safe if DK is only pro which matches for all case ala 49 which need safe move).

```
;Anyway, DK-MOBIL checks for all moves which may be considered safe after
;the sacrifice.  These will be used to get attacks on flight squares.
;This is expensive but hopefully
;there will usually be 0 posdecoys and never more than 1 with same P1 and SQ.
;may be many P2 and SQ1s so we won't handle those.  Special case of P2 moving from
;his new square should perhaps be handled later.

;onlypro of DK not very accurate.  Alternative is to also require one of these 3:
;-sq2 must be invue of P2 from SQ1 (this requires P2 and SQ1 to be around
;        and a test for P3 being P2, invue used to avoid checking possibility of
;        it bearing thru pl and/or piece on sq.
;-DK must not attack sq2 from sq3 (the square he flies to).  This one requires
;        the flight square which is a pain but we must do this one to avoid
;        losing obvious winners.
;-some P4 that canattack sq2 safely (thus making it safe as in 49).
```

Thought about getting plan that included this third attack on the flight sq, but that means producing plan immediately before we know if all flight squares are covered (e.g.in implies of DECOYK2) or else writing other prods to do special trip when there is UNIQUE flight square. This is however very expensive for what we get so it was forgotten.

Since DECOYK2 very expensive, never do it is DECOYK1 matches for any second move. For each flight square, find an SP1 to attack it in one or two moves both of which are safe according to DK-MOBIL. If using second checking piece to get flsq then it must move to SQ1 as its first move.
If it canattack then OK it else perhaps it attacks thru pl and/or sq which have now been emptied (should be LP to do this but MP used since want knights in 2 move attacks but must eliminate pawns).
For 2 move attack get second square to move to from first (once again check for thrus on pl and/or sq) and if found see if it is safe by dk-mobil.

NOTE that discovered attacks are not recognized.

MATESQ

MATESQ requires that if K has flight squares he must not have any flight squares from those. (ie all moves must trap him)
Previous matesq stopped there (didn't care if we could attack flight square).
MATESQ now also requires that there must be second enemy piece around king (DELETED: either in current locaton or)
on flight square location. Being around implies any kind of control: direct, thru mating piece, or thru piece on mating square.
But, recognizes special
case of king being trapped on back row (or any line): here can be flight squares after king move but if P1 can move onto line that king's moving along and remove all flight squares in that manner then ok it.
Idea here is we want mates than only have one thing wrong: if we can mate outright then don't care if another piece around, but if can't then require it or else may need one fix to make square safe and one move to bring another piece up and another to get him safely to attacking sq.
Special case attempts to recognize when checking piece could also deliver the mate after the flight sq so then only need to make it safe.

If MATESQ matches, want to move P1 to SQ but check for special case.
MATE2 recognizes when matesq matched cuz king could move to plan where he's trapped and P1 can move mobstay to SQ and from there can move to SQ1 from which he attacks all flight squares of king.
In this case, suggest plan of moving P1 to SQ and then to SQ1.
(If sq1 unsafe plan may fail at ply 3)
Alternatives:
-don't require mobstay
  (rejected since too many things to fix
  for plan to work then, as is there is still move goal for these);
-could let second move be by other than P1, ie have a P2 and
use (or P1=p2∧canmove p2 sq sq1 P1≠P2∧legalmove p2 sq1) where p2 on sq1
covers all flight sqs after P1 to sq
  (rejected since may be suggested elsewhere, eg. likely that p2→sq1 matesq already;
  --note that P1 to SQ still suggested only would do this to do P2 first,
  may still want to do this if arguments in MATE2U below hold:
  if do it, remember that p2 is only really effective after P1 to SQ so may just
  want to prepare P2 first (eg make safe))

MATEN02
Does MOVE goal for all matesqs that didn't fit special case. (Not doing special case ones in addition makes for less clutter).

MATE2U
Notes that sq1 not safe and first tries to make it safe before moving P1 to sq.
NB: plan of moving P1 to Sq has already been suggested.
This done cuz opponent has limited way to answer safe making moves since mate threatened and process of checking first may change things so that current safe plans may not work after check. (Written for pos 76 after BxN, PxB)
Just uses SAFE since cond in SAFE1 should skip it when necessary.

MATESQ new, unused ATTPRO

Wrote new way that would check for attacks on flight sqare:
(seems to add 1.5 to 2 cpu seconds per analysis, too expensive for good it does)
MATESQ etc: Use FUNTIONE to get P1 and SQ instantiated in ccheck.
  Require P1 to attack all flight squares then we win, else
  it must attack all flight squares from new square (ie open move does not
  lead to more openings). If there is some way to attack this open move then
  suggest check with likely0, else likely 1. Very rough calculation done
  for whether or not we can attack since we don't mind trying things here
  even if we really cannot attack.

ATTPRO 0;1,2,3
Written for 16. DSP1 is on SQ2. P1 wants to capture him. DP1 prevents.
P2 to SQ1 makes DP1 split. 1 Does it if DP1 cannot split and save DSP1.
2 does it if DP1 can only split to one good place and P2 can then fork
DP1 and DSP1. 3 not written yet. Should do it if third piece (not p1,p2)
can attack DP1 on his split spot. Attack must be threatening (exchloss on SQ
should be less than value of DP1), must not block p1's line to sq2, and must
be mobstay or have DP1 as only pro and not ca dp1 from split spot.

THREAT ALL ELSE

TRAPMATE
This suggests moves when one move can cover flight path and next move can check king. Needed in pos 90 after Nxg7, Kxg7, B-e5, Q-c3, Rxf6, R-h8, Rxh6, K-f8 in order to suggest bishop move.

TWOCHECK
Like TRAPMATE but suggests moves in other order, ie check first. It also requires the ocvering move cover first flight move and any flight moves that can be made from there. Only works when there is unique first flight move tho this could be changed (use IMPLIES) to recognize cover move that covers all flight squares and all flight sqs from those flight sqs. Does not do it if check move is a matesq.
(Works on pos 82) (TC1 used since TERM KS uses TWOCHECK)

CHKFORCE
Approves move if it checks and forces king to square where an ep piece can then be taken with check. Works for R-q7 in 78.

DECOYPRO
Matches when DP1 protects DSP1, (use DSP so don't get protections of mating squares which are handled in matesq) Decoy DP1 to make capture possible.

FORKPK
FORKSQ does not check forks of pawns. Many reasons: it tries unsafe ones and there are huge number of forks with pawns. Also huge number makes looking for all possible fork squares very expensive. FORKPK is a prod that's better than nothing: it's inexpensive and finds obvious forks of pawn and king. Useful in end game quiescence searches.

It checks for obvious replies: it king can move to sQ2 and protect pawn.
It's just an approximation: maybe out attack kills pawn even if k protects!
Not done in regular forks because here we know: which piece is more valuable, easy to calculate if SQ2 will be safe for a king after our move, approximation would not be reasonable in general case.

LAST 3 PRODS
These are pretty wierd. They suggest nifty sacrificial attacks on pieces which is good, but they don't care if sacrifice is more than piece attacked and they don't care if piece is overprotected. They are in here because in pos90 these attacks lead to mating net somewhere in future since the sacrifices also weaken king pawn structure--a fact not noted here.
Could require, at least in first and third, that DP2 be near DK.

THREAT DMATESQ stuff

DMATESQ
P1 can mate on SQ (though may not be safe) but can't move there. P1 can move to SQ1 from where he bears DSC onto SQ thru P3.
(Actually may be more intervening pieces, later productions check if they are decoyed by p3 moving). P3 has legal capture or mate move which uncovers.
(Don't allow mobstay ones since they will be suggested anyway!)
P1 should be able to safely move to sql after P3 moves.
This checked by checking current safety and seeing if moving will subtract or add any supports for us or opponent. (76: Q-N4)

Now want to make plans based on whether SQ is safe or not.
If DP1 protects SQ2 from P3 then have case where DP1 decoyed.
Opponent can accept loss from P3 to SQ2 so forkt this with mating threat.
If SQ safe and p1 canattack sq drom p3's loc then ok it,
else see if DP1's decoy will make canattack true.
Special case where safe not requried: if decoy of DP1 allows check thru SQ then it's likely 0. (eg in 76, decoy pawn from in front of king).

DMWIN1,2
Here recognize special winning case (a la 76). These 2 could be combined with OR and exists on DP1 as in DMSAFE, but want plan to be nil in 1 and (nil sq2) in 2.
Can't use exchval p1 sql accurately.
RETURN implies these must be last but would be good idea to have SKIP command implemented anyway.

DMSAFE
here we threaten to win K except for loss of P3 on SQ2. This is likely 1 since we wait a move to get P1 to SQ. Therefore assume defense will not let P3 stay so no forkt (if offense leaves P3 then threat may be less but it will be likely 0). Can't use EXCHVAL P1 SQ accurately since P3 may be making it safe. No position known that this works for but may be one someday.

MATESQD unused

This was first attempt at DMATE that was rejected.

If we can move piece to threat mate such that we can uncover it next time
then do it with likely 1. If first move not safe then it's a little nebulous
since we don't know if there is safe or forcing uncovering move.
Therefore require moves which make the move safe to move the piece
which the move will be discovering thru. This is a bit weird -- should
probably skip MUSTMOVECOND entirely and just let likely 1 throw out weird
plans. As is, SAFE does not always check cond if its capturing or forcing
so it's not always enforced anyway.
(works on 76)

```
THREAT
DMATESQ
((P1 SQ DK SQ1 P3)
(PATTERN MATESQ P1 SQ DK)
(FUNCTION SAFEP P1 SQ) ;can't require this on 76 but without it deep shit
(NEVER (PATTERN DIR P1 SQ))
(NEVER (EXISTS (P2) (PATTERN THRU P1 P2 SQ)))
(PATTERN LEGALMOVE P1 SQ1)
(FUNCTION CANATTACK P1 SQ1 (LOC P3 BD))
(FUNCTION LINE SQ1 (LOC P3 BD) SQ)
(FUNCTION CANATTACK P1 (LOC P3 BD) SQ))

THREAT
DMSAFE
((P1 SQ DK SQ1 P3)
(PATTERN DMATESQ P1 SQ DK SQ1 P3)
(PATTERN MOBSTAY P1 SQ1)
(PLANNEW  ((P1 SQ1) NIL (ATTACK (COLOR P1) SQ)) (LIKELY 1)
  (THREAT (PLUS (WIN DK) (EXCHVAL P1 SQ1)))))

THREAT
DMUNSAFE
((P1 SQ DK SQ1 P3)
(PATTERN DMATESQ P1 SQ DK SQ1 P3)
(NEVER (PATTERN MOBSTAY P1 SQ1))
(ACTIONNEW MOVE1 (P1 SQ1) (LIKELY 1)
  (THREAT (WIN DK))  (PLAN (SAFEMOVE P1 SQ))
  (MUSTMOVECOND (FUNCTION EQUAL P2 P3))))
```

ATTACK

expects COL and SQ3 and tries to attack SQ3 for that side.
Posits PLANs and MOVE goals. Expects threat attribute, but ignores all others (in particular plans, conds, and likelys).
After using threat atr it will ignore it and pass on as threat the exchange of the attacking piece for the piece on SQ3. Note that this gives a zero threat if SQ3 not occupied-- for now we only attack pieces so this should be ok. (To attack sqs nothing else needs be changed although some dsc attacks won't be done.)
Attacking a square is different than getting a piece to a square.
not recognized: 2 move attacks where first move sets up disc attack and second uncovers it.

FIRST 5 DSC PRODS: handle case where can make discovered attack on SQ3. If threat not worth uncovering piece then only safe moves suggested. Else, only moves which attack a defensive major piece (that can be captured with gain), capture an enemy piece, or block an interposable piece are suggested. If there are no such moves, then all safe moves are suggested and if no safe moves then all legal moves suggested.

DIRECT: get candidates for direct attack. Does not allow pieces already attacking, nor more valuable pieces attacking an overprotected piece.

DIRECT1: this posits MOVE goals for MOBSTAY moves. No COND is given here since MOVE KS does not try a 2 move solution when a MOBSTAY 1 move solution is present. COND checked here not in DIRECT to avoid checking it in some cases. (Prevents untrapping of piece on SQ3)

DIRECT2: posits MOVES goals that would accomplish a direct attack on SQ3. If a piece can attack SQ3 with a MOBSTAY move, then no others are suggested to shorten solution. COND keeps MOVE KS from accomplishing goal by first move of a 2 move solution. COND only checked when legalmove since its expensive and probably won't throw out non-legal move anyway.

DBLK2: works on ply 3 of pos96. All other dblks just added for completeness, don't know if they work for any pos.

DECBLKR obvious and necessary

BLKER finds situation where LP1 could CA SQ3 from SQ except for DP1 blocking.

DECBLKP if its a pawn then decoy it. Not good for non-pawns since after decoy he just moves back to interpose again.

DBLK can do for pieces if we decoy DP1 to some DP2 which is blocking another attacking line to SQ3. This finds DP2 and some P1 to attack him.

DBLK1 if P3 has pinray thru Dp2 to sq3 then just ok decoy.

DBLK2 else if we can make pinning move after sac is taken then do it

MOVE

this binds P1 and SQ3 and attempts to move P1 to SQ3 so that the opponent would have to lose something to capture P1 there.
Expects COND attribute for pattern that must match intermediate square of 2 move attack by P1, and PRECOND attr that must be done as goal before plan atr used.
Expects threat, likely attributes, and plan attribute with plan that begins with opponent's move after P1→SQ3.
This will put in (safemove p1 sq3) when other things done first.
The threat is what is threatened after P1→sq3, thus this ks adds any threat realized by the actual move p1→sq3. In general this means adding exchval when the move is mobil (if not mobil might be subtracting from threat value without meaning to).

Problem is CHKFORCE or so may want p1 to sq3 to be a checkmove so have plan begin with move of P1 to SQ3 and this KS then add in front of that or takes cdr when it wants to move innediately.

Has MOVE2 KS to set up MOVE goals for first move of 2 move attacks, and MOVE1 KS to do first move of all attacks.

CHECKCOND checked in prodcutions besides backl, since same SQ2 may get thru on a second atrl and then get approved for first atrl that cond should have rejected.

This KS is harder to read than others. It is a necessary capsulization of knowledge, otherwise every threat in THREAT and some ATTACKs need all these productions to be repeated for themselves. However, concentrating them here means that much information from the original production must be given in the atr lists for use here. Thus a lot of hairy kludge type things are here to process this information.

MOVE2 MOVE3
This sets up patterns for reasonable 2 move sequences for getting P1 to SQ3.
If SQ3 is overprotected then nothing is reasonable.
Also, is square occupied by friend then no 2 move sequence considered.
Using LEGALMOVE prevents use of M3 KS for finding 3 move attacks.
(when it is allowed there are very many attacks)
(also allowing it means putting LEGALMOVE into each prod of 2 move attacks)
Any conditions on the goal must be verified true for the intermediate square.

```
MOVE3
M3
((P1 SQ2 SQ3)
(PATTERN BACK1 P1 SQ2 SQ3)
(NOT (PATTERN DIR P1 SQ2))
(FUNCTION SAFENB P1 SQ3)
(CONCEPT MOVE (P1 SQ2) NIL NIL)
(ACTION MOVE (P1 SQ2) (THREAT (GOAL THREAT))
 (LIKELY (ADD1 (GOAL1 LIKELY)))
 (PLAN NIL (SAFEMOVE P1 SQ3) (GOAL PLAN))
 (REASON (M3 P1 SQ2 SQ3) (GOAL REASON))))
```
matches when P1 needs 3 or more moves to reach SQ3 via SQ2 and needs at least
2 moves to get there via any route. This is only considered threat when
SQ3 is safe for P1 and when there is already a threat from SQ2.

M2SAFE: sets up MOVE goal for first move of 2 move attack when second move
  seems safe for P1 to stay on SQ3.
M2UNSAFE: when second move not safe, then
  set up goal for first move with PRECOND of making second safe.
CLEAR: if SQ3 occupied by friend and safe for P1 and friend has MOBIL move,
  then suggest move goal for MOBIL move with plan of putting P1 on SQ3.
  Now this adds 1 to likely but should not do this if mobil move has a
  threat, eg. is a capture or has own move goal.

The following was removed because: since SQ3 is not overprotected, then the
first move of the two move attack will likely make SQ3 safe.
Thus do not care about exchange (hopefully).
(NEVER (FUNCTION EXCHANGE P1 SQ3))
That line was removed from both M2SAFE and M2UNSAFE.
(This occurrred while doing pos90 after NxNP, KxP, and Q-B4)
(Not sure what pos caused M2USAFE in first place but should try others.)
MOVE2
M2USAFE: when opponent can exchange P1 on SQ3 (whether safe or not)
  then allow 2 move attack with SAFER precondition which will try to
  prevent exchange. PRECOND value must be recognizable goal
```
((P1 SQ2 SQ3)
(PATTERN BACK1 P1 SQ2 SQ3)
(FUNCTION EXCHANGE P1 SQ3)
(ACTION MOVE (P1 SQ2) (THREAT (GOAL THREAT))
 (LIKELY (ADD1 (GOAL1 LIKELY)))
 (PRECOND (SAFER P1 SQ3))
 (PLAN NIL (SAFEMOVE P1 SQ3) (GOAL PLAN))))
```

~nothing done about unpinning P1 so it can move.

Perhaps should recognize when first move of 2 makes threat (ie has other
move or movel or planl goal for it). But combination gives likely 0 with
forkt of threats which means we're subsumed by ones which will be likely 0
plans anyway. But what we want to see threat of this move in case where
it may not be recognized anyway. For example, we don't recognize just

an attack of a piece we can capture or an attack of a piece which protects
something as a threat (since they can escape). But if we ca do this with
first move then perhaps we should have likely 0 with a forkt threat!

```
DECOYSUP (old)
REMOVED FOR DECOY KS WHICH DOES IT BETTER (SAFE1 THING BELOW DONE)
this attempts to make a SQ safe by decoying enemy pieces.  It uses MOBIL
moves that safely attack a major enemy piece that hopefully can only by
answered by moving a piece which protects the square to be made safe.
There could be a UNIQUE around DIR DP1 SQ to assure this, but since
the move is MOBIL there aren't going to be many ways to stop it and the
UNIQUE misses the case when you'd be forcing him to start the exchange will
an unfavorable piece (eg Q).
DMP1 and DP1 are assigned in parallel.  Avoids checking one for all instantiations
of other.  SAFECOND left til end since it may changes atrls in goal and the
effect of this would be lost in the parallel section.
Note that DP1 can be DMP1.  Also second part of OR (THRU) should
really be in SAFE1 and not SAFER but I didn't want to use the prod twice.
((P1 SQ P2 SQ1 DP1 DMP1)
(PATTERN LEGALMOVE P1 SQ)
(PATTERN MOBIL P2 SQ1)
(FUNCTION NEQ P1 P2)
(FUNCTION NEQ SQ SQ1)
(NEVER (FUNCTION LINE (LOC P1 BD) SQ1 SQ))
(NEVER (FUNCTION OVERPRO (COLOR P1) SQ1))
(PARALLEL

((PATTERN DIR DP1 SQ1)
(OR     (PATTERN DIR DP1 SQ)
        (EXISTS (LP1) (PATTERN THRU LP1 DP1 SQ)
                (NEVER (FUNCTION LINE (LOC LP1 BD) SQ1 SQ))))
(NEVER (FUNCTION CANATTACK DP1 SQ1 SQ))
(NEVER  (EXISTS (DP2)
        (PATTERN DIR DP2 SQ)
        (FUNCTION NEQ DP2 DP1)
        (FUNCTION ASVAL P1 DP2)))
(NEVER (EXISTS (P3 DP2) (PATTERN PIN P3 DP1 DP2) (FUNCTION ASVAL DP2 P2))))

((FUNCTION CANATTACK P2 SQ1 (LOC DMP1 BD))
(NEVER (PATTERN DIR P2 (LOC DMP1 BD)))
(FUNCTION SAFEP P2 (LOC DMP1 BD))
(PIECESAFE P1 (PLUS (EXCHVAL P2 (LOC DMP1 BD)) (EXCHVAL P2 SQ1)))))

(CHECKCOND SAFECOND)
(PLAN ((P2 SQ1) (GOAL PLAN))
  (THREAT (FORKT (EXCH P2 DMP1) (PLUS (GOAL THREAT) (EXCHVAL P2 SQ1))))
  (PLAN)))
```

MOVE1

This KS combines many atrls of MOVE goals into single plans or SAFE and SAFER goals. Some of these many atrls may have preconds.
Handles threatening discovering moves and will also decoy an enemy piece blocking the line of the move.

```
M1DECBLK: if DP1 blocks move then set up goal of decoying him if SQ3 looks
 safe or if DP1 bears on it (since decoying may make it safe then).
 DECOYCOND prevents decoy from taking place on sq that would also block.
M1:if P1 can stay on SQ3 then select atrls with no precond
 and suggest plan with least likely atr and combo of plans in goal.
 (Check for no 2 move exchange -- if there is one M1USAFE will make safer goal)
M1DPIN: if P1 can move to but not stay on SQ3 and by so doing can
 uncover a pin of the piece which protects SQ3 (and is less valuable than P1)
 then suggest plan of moving P1 to SQ3 followed by taking the pin object
 if the pinned piece moves, else continuing plan.
M1EX: if P1 safe on SQ3 but can be exchanged then SAFER goal to stop exchange.
M1PREWIN: if P1 can stay on SQ3 and all have preconds and moving P1 to SQ3
  solves a precond then suggest PLAN for it and eliminate atrl with solved
  precond (foreachgoal does it).
  Usually P1 will equal P2 (ie solve precond for self.)
  Makes plan for case when move captures only piece protecting PRECOND square.
M1PREWN2: like above but suggests plan when move captures a piece and this
  allows both moving piece and a friendly thru piece to bear on the precond
  square which must be protected only once and by a more valueable piece
  than the one being moved.
M1PRE: if P1 can stay on SQ3 and all have preconds  then make goals of preconds.
M1UNSAFE: if not exchangable and not safe then set up
  SAFE goal for those with no precond (forget precond ones).
M1USAFE: if exchangable (whether safe or not) set
  up SAFER goal to prevent exchange for those with no preconds.
  (if mobstays get here then tere is 2 move exchange so ok it even if no exchange)
(above 2 do not check overpro since this done in safe and safer has prod that works
for some overpro ones -- could check overpro2 if desried)

This prod makes system work for pos3: .
M1DPIN
((P1 SQ3 LP1 DP1 DP2)
(PATTERN LEGALMOVE P1 SQ3)
(NEVER (PATTERN MOBSTAY P1 SQ3))
(REMATRS T ((PRECOND NIL)))
(UNIQUE (PATTERN DIR DP1 SQ3)
        (FUNCTION ASVAL P1 DP1))
(PATTERN THRU LP1 P1 (LOC DP1 BD))
(NEVER (FUNCTION LINE (LOC LP1 BD) SQ3 (LOC DP1 BD)))
(FUNCTION CANMOVE LP1 (LOC DP1 BD) (LOC DP2 BD))
(FUNCTION SAFEP LP1 (LOC DP2 BD))
(NEVER (FUNCTION EXCHANGE LP1 (LOC DP2 BD)))
(FUNCTION MOREVAL DP2 P1)
(PLAN ((P1 SQ3) (((DP1 NIL) (LP1 (LOC DP2 BD))) ((GOAL PLAN))))
        (LIKELY (GOAL1 LIKELY)) (THREAT (GOAL THREAT))))

(do return after it and put before M1SAFE,UNSAFE)
```

SAFE

Takes Pl and SQ as args. Tries to make it safe for Pl to land on SQ. Expects threat, likely attributes, and plan attribute with plan that begins with opponent's move after any safe making moves. Plan usually has (safemove pl sq) in it.

Threat is for what will happen after Pl→sq. Here we add any threat made in making it safe. Perhaps should also add any threat made by capturing the piece on sq (if one exists) but this is very hard since we don't know how adding supports, etc. will affect occup. Since it's not safe now we'll just assume that we've got back to neutral by helping, it some cases may actually be winning piece on sq so perhaps we might add some to threat for this. (Especially applicable on prods where we've forced one of their pieces away.)

SAFE1 KS tries any thing to make move SAFE and SAFER KS only tries things that will prevent exchanges (they also make it safe.)
Thus, SAFE1 should contain only things that will not help when we are in an exchange situation. All other things sould be in SAFER.
Basically SAFE1 has prods which add one more support for offense which does not help if they can exchange anyway.
It is assumed that SQ is not overprotected--else this is all worthless.
(But overpro only works when Pl bears on SQ, if not and overpro then require some piece on Pl's side to bear dir on sq).

SAFE1

SUPI
this supports SQ if support will help. If supporting move captures a piece then better threat reflects this.

SUPCHK
like SUPI, supports SQ except does it by attacking thru enemy king.
It does it even if opponent has lower value piece attacking sq on the theory that a check is a perturbing thing that may change things.

DSC PRODS
(Pl must be here so deducolor can decide how to instan vars)
Done differently here than in ATTACK KS. There we know that actually uncovering attack is winning thing so never increase likely. Here we may just be uncovering support so we don't know anything is threatened. Suggest uncovering moves which capture or attack pieces even if not mobil (if threat good enuf and likely 8) and keep same likely. If none of those then suggest any mobil ones but increase likely. (Perhaps should prefer ones that block interposing line?)

DECBLK
If we have a support that is being blocked, set up goal of decoying blocker.
Only do if Pl to SQ is legalmove since such involved things as decoying don't work well a few moves ahead. DECOYCOND makes sure decoy does not still block.

SAFER

SRCOND
Does all sorts of horribles if tries it for any square so make sure not overpro first: but his only good test when P1 bears dir on sq, if not and it's overpro then require someone on P1's side bear dir on sq. This no longer tested as condition since decoysupl does number that wipes out two enemy pieces which may work on overpro squares so try it first with overpro2 test and then kill ones that are overpro.

CAPTURE
Takes P1 and DP1 as args. Attempts to capture DP1 with a MOBIL move but does not use P1 to do it.

WINEX1
This starts an exchange on SQ when we can safely do i.

UNPIN1
this tries to unpin piece protecting Sq to be made safe; not used and no UNPIN KS.

DECOYSAC
This says if there are two pieces that can both do damage on SQ and only one opponent piece attacking SQ then sacrifice one piece on SQ to get second piece there if threat warrants it. Ideally would like to know if first piece is necessary for threat of second piece (eg guards king flight squares). Think about how to do it.

MAKEFLY
attack and drive away opponent's piece that protects sq
ok to use piece we're tring to make safe if we can still move to Sq and if he can't exchange us, even with two move exchange
Make it likely 1 if def still has lesser piece which can recapture P1 (else in pos98 will play r-ql to make q6 safe for wb). Only time would want this likely 8 is when we intend to make 2 pieces fly and more valuable one must be done first (this is unlikely situation).

CAPBLOCK
Meant to set goal of capturing dpl when it is true
that some friendly piece would bear on sq (the goal) except for this dpl
and an indefinite number of his own pieces in the way.

```
(       (FUNCTION INVUE LP1 SQ)
        (NEVER (PATTERN DIR LP1 SQ))
        (NEVER (EXISTS (AP1) (PATTERN THRU LP1 AP1 SQ)))
        (UNIQUE (FUNCTION LINE (LOC LP1 BD) (LOC DP1 BD) SQ)
                (NEVER (FUNCTION INVUE DP1 SQ)))))
(NEVER (FUNCTION EQ P1 LP1))
(ACTION CAPTURE (DP1 P1) (THREAT (GOAL THREAT)) (PLAN (GOAL PLAN))
 (LIKELY (GOAL1 LIKELY))))
```

SAFER

DECOYSUP
New prod tries so decoy enemy pieces bearing on sq, but only when P1 can move there (see SAFE1). Also it's not tried if there is another piece which can still exchange on SQ.
perhaps should try decoys even if overpro since can keep decoying pieces til none left -- e.g. M39 in Pitrat's paper. Or at least try any mobil decoys all the time, and maybe any decoys when not overpro2.

Actually if P1 is onlypro then don't want to decoy when opponent has second piece unless decoy is a mobil move. Done with DS1,2. Else sac queen to remove support when he has Q,R and K all bearing on SQ and we're trying to get B on SQ with no other supports.

BLOCKP
;old decoysup1 says decoy one piece into path of another (GONE NOW)

BLOCK2
idea is to gain two protections thru blocking of DP2's line to SQ.
blocking done on SQ1. either blocking piece must be safe and ca sq from sq1
or all safe recaptures must not ca Sq from sq1 and must be pieces that bore on SQ.

DECOYSAC
works on 1. Last or wants to do piecesafe on p1 and threat from p2's safe goal,
but use this approximation (better and more expensive would be to look for pieces attacked by p2's move whenever p1 en prise.

DECOY KS (DP1 P1 SQ)
This tries to decoy DP1 to help make P1 to SQ more plausible (safer or possible).
Expects threat, likely attributes, and plan attribute with plan that begins
with opponent's move after any safe making moves. Plan usually has
(safemove p1 sq) in it. DECOYCOND may put requirements on SQ1 (where we'll move
to decoy dp1) and on DP1. (Usually: SQ1 not on certain line, and DP1 can't attack
certain sq from SQ1).

NOW DONE IN FORCE
Threat is for what will happen after P1→sq. Here we add any threat made
in making it safe. Perhaps should also add any threat made by capturing the
piece on sq (if one exists) but this is very hard since we don't know how
adding supports, etc. will affect occup. Since it's not safe now we'll
just assume that we've got back to neutral by helping, it some cases may
actually be winning piece on sq so perhaps we might add some to threat for this.
(Especially applicable on prods where we've forced one of their pieces away.)

Examples: N-N5 decoys N to help Q protect D7 in pos78.
        QxP decoys B to help R mate in pos68.
        Q-f8 decoys R to help R mate in pos27.

MOVDEC
This pattern finds a move P2 to SQ1 that should decoy DP1 if it threatens
something. Of course, don't decoy with P1 or block his line to SQ. In
fact, there may be other pieces than DP1 which can capture on SQ1 but the
OVERPRO clause throws out obvious losers. We could require DP1 as the
only DP bearing on SQ1 but that would miss the case when DP1 is the
optimal piece to deploy to SQ1. (hope OVERPRO throws out losers)
Don't decoy DP1 to place where
he can attack SQ (must check if canattack thru current location).
Require that we threaten more than we lose by leaving P2 hanging (since
we don't plan to recapture).
Also require that P2 not be protecting SQ or we'll be defeating the purpose.
If DP1 is pinned to king or piece more valuable than P2 and threat then
don't try decoy. (he can't or won't take it)

This now says either never dir p2 sq or ethru p1 dp1 sq.
If never dir, then cannot decoy to object square (i.e. SQ cannnot be SQ1).
The only time we want SQ=SQ1 is when P1 bears ethru piece we're decoying--
then decoying DP1 to SQ is great (if you move to SQ and he doesn't take decoy
then you're very awkward). To decoy to SQ in non-ETHRU cases: WINEX should
do it for a safe move, and DECOYSAC should do it for non safe move.

WANT TO USE SPECIFIC RESPONSE (DP1 SQ1) SO THAT DON'T STRING INFINITE BUNCH
OF DECOYS TOGETHER IN SEARCH WHEN HE JUST AVOIDS EACH ONE -- NECESSARY BECAUSE
CURSEVAL DOES NOT TRY BETTER HIGHER WHEN PLANFROMTOP IS TRUE.

SPLIT
Recognizes case when only threat of P2 to SQ1 is against DP1 itself.
(Use "only" and DSP1 to avoid redoing what FORCE will do.)
Misses case when decoying pawn when one other piece is attacked.
This is threat only if DP1 cannot flee to useful place.

SPLITD
When DP1 bears dir on SQ, approves split when he no longer does.

SPLITB
When DP1 blocks P3 from SQ, approves split when he no longer does.

FORCE

(p1 sq p2) p1 moves to sq as first move in plan and prepares a move of p2.
FORCE expects plan after p1→sq. Expects threat which doesn't take into account
move of P1 to SQ. Expects DECOY atr which is name of piece. This used to
ok threat against that piece and to require some threat before approving.
Uses p2 to see if piecesafe is ok for it.

OK checks as is, just add on plan and threat. If not decoy then ok as is if
mobil with 1 added likely (get good threat here since forkt later may lessen threat).
Then find all threats made by P1 sq. If none then ok it if its a capture.
If some then ok it and make list for all of them -- threat also takes
into account any possible capture.

FTHREAT figures we got a threat when we attack someone that we don't already
and we can safely capture him. If he can take us back, then he must
be piece we're decoying or we must be mobil. P2 must be piecesafe.

DECTHREAT → FTHREAT
This finds DMP1s that are advantageously attacked by P2 to SQ1. Don't count
one that can take us right back (unless they are piece to be decoyed) since this
is obvious loser. (except they are more valuable than P2 and can be recaptured:
eg in 52 get Q for R and N).

In the following two prods would want plan to specify recapture instead of
NIL for defense. This would involve taking cdr of plan in atr1 which is fine
but bigger problem is when plan is RA list, then must return list of things
each with cdr removed so this no done.

CAPONLY
If no attacks but it's a capture then approve it.

DECOK
Now we know there is a DMP1 threatened. We use SET1 so only one atr1 made for
all DMP1s found.

Problem in 78 on N-D6, cuz whenever we want to make a square safe, we just attack
his queen in such a way that it can take us then this is oked as decoy.
Such a plan should only be used if queen cannot flee to square from where it
can maintain protection (makefly). New prods added to DECOY to do this.

Now require that there be two pieces attacked when doing decoy.
DECOK1 does all non-decoys together to avoid ineffieciency.
For threat, really want only second best of attacked pieces since he can flee
with best one but used this way to keep optimistic since best one might protect
second best one or perform some other function we don't know about. (Perhaps
post Can-Make-FLee concept for this move to use if we have decoy or so).
Important to use exchval, with exch then in tree plan keeps thinking it can
win that best one when it has already fled.

Here is a case of possible non-optimistic threat evaluation:
If threat is (exchval p1 sq3) or (exch p1 dp3) for some DP3 or SQ3, then
if we decoy a piece that was bearing on SQ3 away so that he no longer
does then we will do better than this exchval shows.
Not sure how to correct this.

DEFSAFE

Takes P1 and SQ as arguments. Suggests moves by other side to stop P1 from moving to SQ or to blunt the effect of the move.
Expects SAVE atr which it passes on, else PLAN1 eliminates possible best moves (eg sacrifices to stop mate).
Also expects LIKELY and THREAT and PLAN and COND which are all passed on.
When suggesting MOBIL moves, a DEFTHREAT goal is posited so any threat a move might make will be considered in ordering.

BLOCK
suggests MOBIL moves which block the move

BLOCKSAC
if no mobil blocks then suggests non mobil ones if they lose less than SAVE. Right now only captures and blocks are done without mobils since they are strongest counters to stopping something.

CAPTUR
tries to capture pieces which bear DIR, ETHRU, or OTHRU on SQ and pieces which have P1 as their only support. Uses MOBIL moves and legalmoves where capturing piece minus captured piece must not exceed SAVE of goal.

RUNP
if SQ occupied any P1 is MOBSTAY on it, then set up goal of RUN for piece on SQ.

QUIT
stop here if king is in check

BLOCKSUP
suggests MOBIL moves which block P2 from SQ when P2 bears ETHRU or OTHRU on SQ

PROTECT
suggests MOBIL moves which add another support for SQ unless king is on it

DISCOV
suggests any MOBIL move which uncovers an attack on SQ.

PINDEF, FORKDEF
recognize if P1 will fork or pin from SQ and sets up goals to move the pieces which will be pinned or forked. Technically, should have condition that they will move off line of threat--(could be implemented by having COND atr and passing that atr to DIRMOSTMOB in RUN KS)
In forkdef just need one run goal since both halves of fork will match as DMP1.

RUNCHK
if K is threatened then start moving him early looking for an escape route

DEFTHREAT TERM DEFOFF

Takes DP1 and SQ1 as arguments. Assumes this is MOBIL move to be made.
Checks COND which may put constaints on DP1.
Creates plan by putting DP1 to SQ1 on front of plan in goal.
Expects SAVE and LIEKLY atrs which it passes on.
Gives threat of 1 or exch value when move captures, threat of 1 when it threatens a piece, else nothing. (adds to existing threat)

LOSE
This helps reject moves that make his ep caps even more threatening.

TERM

This finds threats for quiescence evaluation. Only finds ones that are sure to work. There is a crock on the fork one since we can't have conds in attribute specification.
If uniquedir true for dmp1 then want (win dmp1) instead of exch.
Same for DMP2. To be accurate need four prods for four combinations of uniquedir and must keep other three from firing when anyone does.
To avoid this, just make both be WIN if one is (optimistic).
Good place for optimism since we are capturing and threatening at same time.
Does not set up PLAN1 attr since epeval does own defense (thus needs own ep prod).

Exch must be used to get optim evaluation in enprise case.

MOBTERM keeps from being overoptimistic when thinks it can do a trip from SQ since its mobstay when actually it can only exchange itself for piece on sq. Could use in threat also but don't need to be so accurate there.

FORKREG tries to check that one piece cannot adequately defend other. (e.g. in 78 after Ng5 NXN Rd7 QXR QXQ many forks of white queen with black king and b (or n) where king can protect b after wq leaves rank 7.

TWOCHECK and TRAPMATE borrowed from THREAT to recognize patterns where there might be a mate around but it's not sure enough to try the move.

DSC1 captures on uncovering move found by enprise.

DEFOFF

Just like DEFSAFE except only uses some of its prods (leaves out very defensive ones, keeps more offensive ones) and replaces RUN with prod that's more particualar about what it runs.

RUN DEFENSE DEFEP

This KS is used to move a piece when it's in danger where it is. Essentially a defensive KS. It assumes the goal has a SAVE attribute which tells how much will be saved by moving the piece to a safe square. (Cannot use threat since threat should give an expected board value, which will still be same. But must have SAVE so we will try move even if it gives up something (PLAN1) as long as what it gives up is less than SAVE.
It's arguments are DP1 P1 and SQ1. It suggests moves for DP1 so that P1 will not attack it from SQ1.

A move that captures a piece is given a positive threat. Moves that attack or protect a vulnerable piece and the move with the most mobil moves on the next ply are given a threat of 1, all other MOBIL moves are suggested with threat of zero. If a king must run then all legalmoves are suggested since we don't care how much we lose saving the king.

DEFENSE

This used to help better differentiate defensive suggestions. HELPCAP improves threat of any capture when offense has a piece hanging elsewhere.
(e.g. after r-d6 in 78, taking R not bad since wq is ep)
Cannot use enpris pattern since don't match when king in check (kept in as or to get discovered en prises).

ALLOWFRK increases loss when moving to place where simple fork can be made.
Really only want to do this in epeval so make DEFEP KS for that.
(This is worth it since it's expensive prod.)
Helps get right king move in G0022.FIE from 78. It does not recognize forks where attack line goes thru moved piece -- this would be easy addition if needed.

PLAN1

checks for giving up functions.
If nothing given up, or if checking then ok it.
If we are threatening or saving more than we're giving up,
then decrease threat by what we're giving up:
This done be keeping separate LOSS atr and using THREATLOSS to eval threat
by using WIN instead of EXCH (since def cannot exch if he's doing another threat).
Important that each prod matches only once for all giveups, else atrls are
grossly large. Note that matching GIVEUP implies that FIRSTMOVE already matched.
SET used to form loss atr so prod only matches once.
We test to see which give ups will still be so.

GIVEUP
there are 4 giveups: pin, ep, protects, mates

```
        For ep must check that we're not capturing necessary support to
        make the ep capture. Thus, disallow when capturing a supporting piece.
        This is not entirely accurate but errors in optimistic way. Actually
        want to know if support necessary for the ep (something occup does not
        tell us currently) but this would then have to be checked for case when
        P1 bears on ep square from SQ. Thus just do this optim approximation.
        BETTER approximation now: if nobody protects ep piece and we haven't
         captured only support then allow it anyway (but not if capturing piece
         bears on ep piece from sq since then shift of two supports will
         mean exchval is way off) Does following:
        BR---WR-------BR  Want BxR to giveup enprise rook.
             \     /
              BB WQ
        This was wrongly analyzed as needed to see that RxR save rook (ie have
        SAVE KS to add on SAVE for such goodies --problem is that save values
        would overpower good forcing moves which don't give opponent chance to
        recapture his threat.
there are 4 refutes: block line, capture tpiece, cap piece which tpiece protects
        so that captured piece better than losing object,
        threaten piece (not tpiece) so that capturing it better than losing object
All refutes apply to all giveups. However, in case of last two refutes,
        value of object will be whole value for protects but only exchval
        for ep,pin (either for mate)
```

Also mate needs SQ1 to threaten, rest don't but use it as P2's loc.
Thus DP1 threatens to win P2 by landing on SQ1 (P2's loc except when mating).
To accomodate difference between value and exchval, one pat matches ep,pin and
their 4 diff conds, another matches protect,mate and their 4 conds (mate here
since needs some of same tests as protects).
Will combine 2 pats into one -- first two refutes could be here but they aren't
long and they eliminate matches where they are.
Actually this SQ1 shit should go into PROTECTS and have it recognize protection
of matesqs also!!
This is easy -- protects only used in PLAN1 capdecoy and in refuttes and giveups.

!!!Actually, when protects is for ep capture, it does not protect the value of the
ep piece, but only threatens to get the value of the piece that will capture --
so should use minimum of ep piece and capturing piece as threat.!!!!
To do this need a COND in threatok so it's been left out.

Fixes proposed are not checked for giving things up since that involves
infinte regress.

This removed since has nothing to do with P1 and SQ. If needed should be in
threat where you suggest moving pieces that are ep, etc.
CHKFIX
((P1 SQ P2 SQ1 DK)

```
(EXISTS (DP1) (PATTERN GIVEUP P1 SQ P2 DP1))
(PATTERN CHECKMOVE P2 SQ1 DK)
(PATTERN MOBIL P2 SQ1)
(FINALPLAN (FIX GOAL ((P2 SQ1) NIL))
 (SAFECOND) (THREAT (PLUS (GOAL1 THREAT) (EXCHVAL P2 SQ1)))))
```

To make threat optimistic we must realize that if he plays move to get loss then we can take another piece with P1. This is subtracted from LOSS. Note that we can't capture piece that he makes loss threat with-- to be accurate must match subtracted loss with other loss (done by having giveup concepts where loss expressed as atr and the SET must access concept to form list) but instead just don't allow capture of any piece that has a giveup threat.
This done in THREATOK only since capfix and capdecoy are weird anyway.

CHECK
Oks without loss if check -- could check for giveups having dpl as dk but then must only use these giveups when forming loss

THREATOK
Does trip (per above) of getting threats pl makes and adding them into loss, checking that loss not to big (changes obvious giveups to wins.

REJECT
This sets up goal to fix defense's refutation if plan rejected.

CAPFIX
Uses forkt of threats which is not optimistic, but using plus would be a likely 1 plan and as a likely 0 produces poor plans with huge threat values.
 (eg in 78 after N-N5, K-e7 it plays QxP(g6) with plus and likely 0)

SAFEMOVE

This takes move as argument. If its safe then suggests plan of moving it, else sets up move goal to try and accomplish it.

KILL

Takes 2 pieces as arguments. If first can capture the second then suggests this, otherwise forget it. When doing forks and so on, this allows plan to specify a capture without the system solving the goal of bringing it about when the piece to be captured has moved.

```
DESPERADO
ATTROYAL
((SP1 SQ DR1)
(PATTERN MOBIL SP1 SQ)
(FUNCTION CANATTACK SP1 SQ (LOC DR1 BD))
(NEVER (PATTERN DIR SP1 (LOC DR1 BD)))
(PLAN ((SP1 SQ)) (THREAT (WIN DR1)) (LIKELY 0)))
```

This KS removed in favor of TERM KS. This production forgottern as unmotivated.

most things straightforward. PIN tries to assure that something will be lost if DP1 moved.

DSCTHR
denotes uncovered moves that threaten discovered attacks. Knows that discovered attack is valid and that uncovering move cannot be trivially captured by piece being attacing in dsc attack.

MOBIL: means a square can be landed on without loss (of piece that moved). These 2 were here but giving up functions must be reasoned about separately so they have been removed.

1) don't let pinned piece move if it loses. Handling of pins not exact. Comparing piece being captured to pin object to decide if moving pinned piece is ok could be tempered with comparing the expected loss or gain on both squares although this would not be exact either.
2) does not ok move which gives up only protection of a piece that is attacked unless the move wins at least that much.

MOBIL includes obvious discovered moves. Otherwise unsafe move is approved if it has dscthr square to move to and the piece being threatened is as valuable as the piece being sacrificed and the opponent cannot exchange the attacking piece with less loss than we would suffer from sacrificed piece. Here we give sacrificed piece credit for things it attacks but don't count attacking pieces which can capture our attacking piece. (Perhaps should put other requirements on piece we attack, such as: don't allow it if it can move to a non-overprotected sq from which it protects the dsc attack object (if such move not blocked by P1 to SQ)).
(use of attack: pos 27 after RxR, QxR, R-F1)

MOBSTAY: means sqare can be landed on and opponent must lose to capture the piece.
This is not true when an exchange of two pieces will keep material equal.
If you want to take care of that case, add this to mobstay:

```
(IMPLIES (DP1) ((FUNCTION EQUAL DP1 (OCCUPANT SQ BD)))
        (NEVER (FUNCTION SAFEP DP1 SQ)))
```

That not added now cuz two piece exchs leave opportunities open (I think: no documented cases of needing it).
Not having that in has not hurt at all except occassionally when we are trying to mate and don't want mating piece exchanged. In that case we check above line in mating production.

PROTECTS
Recognizes captures and mate moves.
DP1 threatens to win P2 by landing on SQ1 (P2's loc except when mating).
Only P1 prevents it. For capture, requires only P1 bears dir on sq1.
Actually should recognize situation when more than one piece bears on square but all are needed -- should be easy (not overpro) but this seems to be working fine so leave it. (ok if some piece bear othru p1 to sq1)

For mate want to know 1) protects matesq and no safe interpose, or 2) no protection of matesq and this only safe interpose.
Interposing trip is hard cuz want to know that square safe to land on even after opponent has made checking move. Thus must be overprotected now which implies we can move there. As an approximation say P1 necessary for interposition if it can do MOBIL move to interposing square. It is not necessary if there are at least two more pieces bearing on the square but no way to test this so let's approximate.

THREAT EP DSC FORK

this takes COL as an argument and looks for threats for that side.
Assumes nothing from GOAL in atrs.
It posits PLANs, MOVE goals, and ATTACK goals.

nb: NOT DIR P1 SQ3 does not instantiate unless it is true for some instantiations, and then only instans first var.

ENPRISE: uses MOBSTAY to avoid suggestting exchanges for their own sake.

EP2,P,C: gives credit for 2 eps when first doesn't affect second and/or first checks. Needed primarily for epeval. No known place where ep2c used, but ep2p works in 78 (i think - somewhere).

DSCATT: the threat of capturing a piece by uncovering a dsc attack at the same time will be recognized as en prise since MOBIL handles that. So here recognize fork make by moving uncovering piece to attack another.

FORKSQ: FUNCTIONE instantiates thing in reasonable time. COND atr given so that move goal will not generate stupid attempts at two move forks. Safecond prevents decoying of fork objects. FORKSQ win recognizes special situation and gives it optimistic threat. (Possibly very over optimistic, see comments on TERM regarding need conds in threat) Special situation only recognized for 1 move forks since specialness will probably go away in two moves and don't want to generate too much overoptimistic bullshit.

FORKSQW: recognizing winning situation when forking move captures only piece protecting fork object. Since exchval is still 0 or less special prods were put in to give more optimistic threat for this situation . (used somewhere in tree of 78)

FORKSAC: takes DMP1 with P2 to get DP1 to recapture, then forks DP1 and DMP2 with SP1. Make sure that what we threaten to win is worth giving up P2 for.

CANTRAP: checks if we canattack piece and take away his escapes at same time. Checks for can attack move on piece not already attacked and not trapped that we can safely capture. To assure no escape, checks that for all mobil moves of that piece that either 1) moving thru line we're going to attack along, 2) sq attacked by someone else thru piece we're moving, or 3) attacking piece can attack it from new square. An attack here also requires that square should not be overpro against us unless opponent piece more valuable. Lastly, check that no obvious escapes uncovered: not give up onlypro of sq he can move to, and no have him bearing thru out attacking piece to safe squares (OBJ used since P1 not bear dir on DMP1). Threat should be exchval on dmpl since he should be able to just sit there and exchange. This may let in too many attacks in cases where P1 is more valuable than DMP1, perhaps should require that not be true. (does 42)

THREAT PIN DECOY

PINATT: obvious
  Since this happens a lot and is not very threatening, only MOVE1 attacks are
  tried. To do 2 move attacks, just change action to MOVE and put this in atrl:
```
 (COND (NEVER (FUNCTION LINE SQ2 (LOC DMP1 BD) (LOC DMP2 BD))))
```

DECOYK (old)
~Tries sacrificial moves to squares which only K protects when 1)
~move either checks king, captures piece, or guards one king escape square;
~and 2)it is possible to attack the sacrificial square by some means on the
~next move. (If K does not protect, then move should be safe.)
SACHECK
~This one like DECOYK except allows sacrifice when one piece other than K
~protects (whether K does or not), as long as condition 2) is replaced by:
~there is piece that can recapture with check safely.
~(perhaps requiring safe check is too restrictive; could also allow safely
~guarding check squares)
~~Has been removed since made bad plans and no good ones:

DECOYK (new)
The old one made too many ridiculous sacs, so use these:
POSDECOY
matches if only DK protects SQ and P1 can move there unsafely and P2 can
attack SQ from SQ1 on next move. Well:
First exists P3 SQ2 is just to make sure some other piece than the sacrificed one
is in the general vicinity. (prevents stupid giveaways)
Or has 3 parts for testing attack: first is discovered attack, second is
direct attack, and third is discovered attack that also goes thru sacrificed
piece also. (Direct covers case when would be thru sacrificed piece.)
In discovered ones could require that uncovering move must not be attackable by
king from SQ unless some offen piece besides P1 or P2 bears on it.
(Leave out for now since no errors from this.)
Works: pos 4, pos 9 at ply 5, pos 78 for QxN.

DECOYK1
Says to go ahead with sac if DK on SQ and P2 on SQ1 looks like a matesq
(ie one or fewer moves for king). Previously only checked if some piece
other than P2 was bearing in the area since didn't have power then to
test this matesq kind of stuff. The likely is 1 from previous experience.
Even though the sac forces a reply, the plan almost never worked since
it usually just threw a piece away. Perhaps 0 should be retried when
using this matesq test since plan much more likely too work, although the
matesq calculation assumes P1 is still around when actually it's been
sacrificed.

THREAT decoyk2

DK-MOBIL
DECOYK2
This invented while doing pos 49 and 55. Basically lots of invisible mating nets available after perturbing effects of a posdecoy. (They work in likely 1 situations). Initial idea to cover these types of positions is do it if we can attack king every place he can move to after second check. But as 55 shows, a two move attack can be enough in likely 1 situations, so allow that to. 49 Shows that attacking move need not be safe if the is safe move to make it safe in likely 1 situations (this is not checked since it gets very hairy to determine safety and hairy to know what attacks what since 2 pieces gone, instead we call squares safe if DK is only pro which matches for all case ala 49 which need safe move).

```
;Anyway, DK-MOBIL checks for all moves which may be considered safe after
;the sacrifice.  These will be used to get attacks on flight squares.
;This is expensive but hopefully
;there will usually be 0 posdecoys and never more than 1 with same P1 and SQ.
;may be many P2 and SQ1s so we won't handle those.  Special case of P2 moving from
;his new square should perhaps be handled later.

;onlypro of DK not very accurate.  Alternative is to also require one of these 3:
;-sq2 must be invue of P2 from SQ1 (this requires P2 and SQ1 to be around
;       and a test for P3 being P2, invue used to avoid checking possibility of
;       it bearing thru p1 and/or piece on sq.
;-DK must not attack sq2 from sq3 (the square he flies to).  This one requires
;       the flight square which is a pain but we must do this one to avoid
;       losing obvious winners.
;-some P4 that canattack sq2 safely (thus making it safe as in 49).
```

Thought about getting plan that included this third attack on the flight sq, but that means producing plan immediately before we know if all flight squares are covered (e.g.in implies of DECOYK2) or else writing other prods to do special trip when there is UNIQUE flight square. This is however very expensive for what we get so it was forgotten.

Since DECOYK2 very expensive, never do it is DECOYK1 matches for any second move.
For each flight square, find an SP1 to attack it in one or two moves both of which are safe according to DK-MOBIL. If using second checking piece to get flsq then it must move to SQ1 as its first move.
If it canattack then OK it else perhaps it attacks thru p1 and/or sq which have now been emptied (should be LP to do this but MP used since want knights in 2 move attacks but must eliminate pawns).
For 2 move attack get second square to move to from first (once again check for thrus on p1 and/or sq) and if found see if it is safe by dk-mobil.

NOTE that discovered attacks are not recognized.

MATESQ

MATESQ requires that if K has flight squares he must not have any flight squares from those. (ie all moves must trap him)
Previous matesq stopped there (didn't care if we could attack flight square).
MATESQ now also requires that there must be second enemy piece around king
(DELETED: either in current locaton or)
on flight square location. Being around implies any kind of control: direct, thru mating piece, or thru piece on mating square.
But, recognizes special
case of king being trapped on back row (or any line): here can be flight squares after king move but if P1 can move onto line that king's moving along and remove all flight squares in that manner then ok it.
Idea here is we want mates than only have one thing wrong: if we can mate outright then don't care if another piece around, but if can't then require it or else may need one fix to make square safe and one move to bring another piece up and another to get him safely to attacking sq.
Special case attempts to recognize when checking piece could also deliver the mate after the flight sq so then only need to make it safe.

If MATESQ matches, want to move P1 to SQ but check for special case.
MATE2 recognizes when matesq matched cuz king could move to plan where he's trapped and P1 can move mobstay to SQ and from there can move to SQ1 from which he attacks all flight squares of king.
In this case, suggest plan of moving P1 to SQ and then to SQ1.
(If sq1 unsafe plan may fail at ply 3)
Alternatives:
-don't require mobstay
  (rejected since too many things to fix
  for plan to work then, as is there is still move goal for these);
-could let second move be by other than P1, ie have a P2 and
use (or P1=p2∧canmove p2 sq sq1 P1≠P2∧legalmove p2 sq1) where p2 on sq1
covers all flight sqs after P1 to sq
  (rejected since may be suggested elsewhere, eg. likely that p2→sq1 matesq already;
  --note that P1 to SQ still suggested only would do this to do P2 first,
  may still want to do this if arguments in MATE2U below hold:
  if do it, remember that p2 is only really effective after P1 to SQ so may just
  want to prepare P2 first (eg make safe))

MATENO2
Does MOVE goal for all matesqs that didn't fit special case. (Not doing special case ones in addition makes for less clutter).

MATE2U
Notes that sq1 not safe and first tries to make it safe before moving P1 to sq.
NB: plan of moving P1 to Sq has already been suggested.
This done cuz opponent has limited way to answer safe making moves since mate threatened and process of checking first may change things so that current safe plans may not work after check. (Written for pos 76 after BxN, PxB)
Just uses SAFE since cond in SAFE1 should skip it when necessary.

MATESQ new, unused   ATTPRO

Wrote new way that would check for attacks on flight sqare:
(seems to add 1.5 to 2 cpu seconds per analysis, too expensive for good it does)
MATESQ etc: Use FUNTIONE to get P1 and SQ instantiated in ccheck.
  Require P1 to attack all flight squares then we win, else
  it must attack all flight squares from new square (ie open move does not
  lead to more openings). If there is some way to attack this open move then
  suggest check with likely0, else likely 1.. Very rough calculation done
  for whether or not we can attack since we don't mind trying things here
  even if we really cannot attack.

ATTPRO 0;1,2,3
Written for 16. DSP1 is on SQ2. P1 wants to capture him. DP1 prevents.
P2 to SQ1 makes DP1 split. 1 Does it if DP1 cannot split and save DSP1.
2 does it if DP1 can only split to one good place and P2 can then fork
DP1 and DSP1. 3 not written yet. Should do it if third piece (not p1,p2)
can attack DP1 on his split spot. Attack must be threatening (exchloss on SQ
should be less than value of DP1), must not block p1's line to sq2, and must
be mobstay or have DP1 as only pro and not ca dp1 from split spot.

THREAT ALL ELSE

TRAPMATE
This suggests moves when one move can cover flight path and next move can check king. Needed in pos 90 after Nxg7, Kxg7, B-e5, Q-c3, Rxf6, R-h8, Rxh6, K-f8 in order to suggest bishop move.

TWOCHECK
Like TRAPMATE but suggests moves in other order, ie check first. It also requires the ocvering move cover first flight move and any flight moves that can be made from there. Only works when there is unique first flight move tho this could be changed (use IMPLIES) to recognize cover move that covers all flight squares and all flight sqs from those flight sqs. Does not do it if check move is a matesq.
(Works on pos 82) (TC1 used since TERM KS uses TWOCHECK)

CHKFORCE
Approves move if it checks and forces king to square where an ep piece can then be taken with check. Works for R-q7 in 78.

DECOYPRO
Matches when DP1 protects DSP1. (use DSP so don't get protections of mating squares which are handled in matesq) Decoy DP1 to make capture possible.

FORKPK
FORKSQ does not check forks of pawns. Many reasons: it tries unsafe ones and there are huge number of forks with pawns. Also huge number makes looking for all possible fork squares very expensive. FORKPK is a prod that's better than nothing: it's inexpensive and finds obvious forks of pawn and king. Useful in end game quiescence searches.

It checks for obvious replies: it king can move to sQ2 and protect pawn. It's just an approximation: maybe out attack kills pawn even if k protects! Not done in regular forks because here we know: which piece is more valuable, easy to calculate if SQ2 will be safe for a king after our move, approximation would not be reasonable in general case.

LAST 3 PRODS
These are pretty wierd. They suggest nifty sacrificial attacks on pieces which is good, but they don't care if sacrifice is more than piece attacked and they don't care if piece is overprotected. They are in here because in pos90 these attacks lead to mating net somewhere in future since the sacrifices also weaken king pawn structure--a fact not noted here.
Could require, at least in first and third, that DP2 be near DK.

THREAT DMATESQ stuff

DMATESQ
P1 can mate on SQ (though may not be safe) but can't move there. P1 can move to SQ1 from where he bears DSC onto SQ thru P3.
(Actually may be more intervening pieces, later productions check if they are decoyed by p3 moving). P3 has legal capture or mate move which uncovers.
(Don't allow mobstay ones since they will be suggested anyway!)
P1 should be able to safely move to sq1 after P3 moves.
This checked by checking current safety and seeing if moving will subtract or add any supports for us or opponent. (76: Q-N4)

Now want to make plans based on whether SQ is safe or not.
If DP1 protects SQ2 from P3 then have case where DP1 decoyed.
Opponent can accept loss from P3 to SQ2 so forkt this with mating threat.
If SQ safe and p1 canattack sq drom p3's loc then ok it,
else see if DP1's decoy will make canattack true.
Special case where safe not requried: if decoy of DP1 allows check thru SQ then it's likely 0. (eg in 76, decoy pawn from in front of king).

DMWIN1,2
Here recognize special winning case (a la 76). These 2 could be combined with OR and exists on DP1 as in DMSAFE, but want plan to be nil in 1 and (nil sq2) in 2.
Can't use exchval p1 sq1 accurately.
RETURN implies these must be last but would be good idea to have SKIP command implemented anyway.

DMSAFE
here we threaten to win K except for loss of P3 on SQ2. This is likely 1 since we wait a move to get P1 to SQ. Therefore assume defense will not let P3 stay so no forkt (if offense leaves P3 then threat may be less but it will be likely 0). Can't use EXCHVAL P1 SQ accurately since P3 may be making it safe. No position known that this works for but may be one someday.

MATESQD unused

This was first attempt at DMATE that was rejected.

If we can move piece to threat mate such that we can uncover it next time
then do it with likely 1. If first move not safe then it's a little nebulous
since we don't know if there is safe or forcing uncovering move.
Therefore require moves which make the move safe to move the piece
which the move will be discovering thru. This is a bit weird -- should
probably skip MUSTMOVECOND entirely and just let likely 1 throw out weird
plans. As is, SAFE does not always check cond if its capturing or forcing
so it's not always enforced anyway.
(works on 76)

```
THREAT
DMATESQ
((P1 SQ DK SQ1 P3)
(PATTERN MATESQ P1 SQ DK)
(FUNCTION SAFEP P1 SQ) ;can't require this on 76 but without it deep shit
(NEVER (PATTERN DIR P1 SQ))
(NEVER (EXISTS (P2) (PATTERN THRU P1 P2 SQ)))
(PATTERN LEGALMOVE P1 SQ1)
(FUNCTION CANATTACK P1 SQ1 (LOC P3 BD))
(FUNCTION LINE SQ1 (LOC P3 BD) SQ)
(FUNCTION CANATTACK P1 (LOC P3 BD) SQ))

THREAT
DMSAFE
((P1 SQ DK SQ1 P3)
(PATTERN DMATESQ P1 SQ DK SQ1 P3)
(PATTERN MOBSTAY P1 SQ1)
(PLANNEW  ((P1 SQ1) NIL (ATTACK (COLOR P1) SQ)) (LIKELY 1)
  (THREAT (PLUS (WIN DK) (EXCHVAL P1 SQ1)))))

THREAT
DMUNSAFE
((P1 SQ DK SQ1 P3)
(PATTERN DMATESQ P1 SQ DK SQ1 P3)
(NEVER (PATTERN MOBSTAY P1 SQ1))
(ACTIONNEW MOVE1 (P1 SQ1) (LIKELY 1)
  (THREAT (WIN DK))  (PLAN (SAFEMOVE P1 SQ))
  (MUSTMOVECOND (FUNCTION EQUAL P2 P3))))
```

ATTACK

expects COL and SQ3 and tries to attack SQ3 for that side.
Posits PLANs and MOVE goals. Expects threat attribute, but ignores all others (in particular plans, conds, and likelys).
After using threat atr it will ignore it and pass on as threat the exchange of the attacking piece for the piece on SQ3. Note that this gives a zero threat if SQ3 not occupied-- for now we only attack pieces so this should be ok. (To attack sqs nothing else needs be changed although some dsc attacks won't be done.)
Attacking a square is different than getting a piece to a square.
not recognized: 2 move attacks where first move sets up disc attack and second uncovers it.

FIRST 5 DSC PRODS: handle case where can make discovered attack on SQ3. If threat not worth uncovering piece then only safe moves suggested. Else, only moves which attack a defensive major piece (that can be captured with gain), capture an enemy piece, or block an interposable piece are suggested. If there are no such moves, then all safe moves are suggested and if no safe moves then all legal moves suggested.

DIRECT: get candidates for direct attack. Does not allow pieces already attacking, nor more valuable pieces attacking an overprotected piece.

DIRECT1: this posits MOVE goals for MOBSTAY moves. No COND is given here since MOVE KS does not try a 2 move solution when a MOBSTAY 1 move solution is present. COND checked here not in DIRECT to avoid checking it in some cases. (Prevents untrapping of piece on SQ3)

DIRECT2: posits MOVES goals that would accomplish a direct attack on SQ3. If a piece can attack SQ3 with a MOBSTAY move, then no others are suggested to shorten solution. COND keeps MOVE KS from accomplishing goal by first move of a 2 move solution. COND only checked when legalmove since its expensive and probably won't throw out non-legal move anyway.

DBLK2: works on ply 3 of pos96. All other dblks just added for completeness, don't know if they work for any pos.

DECBLKR obvious and necessary

BLKER finds situation where LP1 could CA SQ3 from SQ except for DP1 blocking.

DECBLKP if its a pawn then decoy it. Not good for non-pawns since after decoy he just moves back to interpose again.

DBLK can do for pieces if we decoy DP1 to some DP2 which is blocking another attacking line to SQ3. This finds DP2 and some P1 to attack him.

DBLK1 if P3 has pinray thru Dp2 to sq3 then just ok decoy.

DBLK2 else if we can make pinning move after sac is taken then do it

MOVE

this binds P1 and SQ3 and attempts to move P1 to SQ3 so that the opponent would have to lose something to capture P1 there.
Expects COND attribute for pattern that must match intermediate square of 2 move attack by P1, and PRECOND attr that must be done as goal before plan atr used.
Expects threat, likely attributes, and plan attribute with plan that begins with opponent's move after P1→SQ3.
This will put in (safemove p1 sq3) when other things done first.
The threat is what is threatened after P1→sq3, thus this ks adds any threat realized by the actual move p1→sq3. In general this means adding exchval when the move is mobil (if not mobil might be subtracting from threat value without meaning to).

Problem is CHKFORCE or so may want p1 to sq3 to be a checkmove so have plan begin with move of P1 to SQ3 and this KS then add in front of that or takes cdr when it wants to move innediately.

Has MOVE2 KS to set up MOVE goals for first move of 2 move attacks, and MOVE1 KS to do first move of all attacks.

CHECKCOND checked in prodcutions besides backl, since same SQ2 may get thru on a second atrl and then get approved for first atrl that cond should have rejected.

This KS is harder to read than others. It is a necessary capsulization of knowledge, otherwise every threat in THREAT and some ATTACKs need all these productions to be repeated for themselves. However, concentrating them here means that much information from the original production must be given in the atr lists for use here. Thus a lot of hairy kludge type things are here to process this information.

MOVE2 MOVE3
This sets up patterns for reasonable 2 move sequences for getting P1 to SQ3.
If SQ3 is overprotected then nothing is reasonable.
Also, is square occupied by friend then no 2 move sequence considered.
Using LEGALMOVE prevents use of M3 KS for finding 3 move attacks.
(when it is allowed there are very many attacks)
(also allowing it means putting LEGALMOVE into each prod of 2 move attacks)
Any conditions on the goal must be verified true for the intermediate square.

```
MOVE3
M3
((P1 SQ2 SQ3)
(PATTERN BACK1 P1 SQ2 SQ3)
(NOT (PATTERN DIR P1 SQ2))
(FUNCTION SAFENB P1 SQ3)
(CONCEPT MOVE (P1 SQ2) NIL NIL)
(ACTION MOVE (P1 SQ2) (THREAT (GOAL THREAT))
 (LIKELY (ADD1 (GOAL1 LIKELY)))
 (PLAN NIL (SAFEMOVE P1 SQ3) (GOAL PLAN))
 (REASON (M3 P1 SQ2 SQ3) (GOAL REASON))))
```
matches when P1 needs 3 or more moves to reach SQ3 via SQ2 and needs at least
2 moves to get there via any route. This is only considered threat when
SQ3 is safe for P1 and when there is already a threat from SQ2.

M2SAFE: sets up MOVE goal for first move of 2 move attack when second move
  seems safe for P1 to stay on SQ3.
M2UNSAFE: when second move not safe, then
  set up goal for first move with PRECOND of making second safe.
CLEAR: if SQ3 occupied by friend and safe for P1 and friend has MOBIL move,
  then suggest move goal for MOBIL move with plan of putting P1 on SQ3.
  Now this adds 1 to likely but should not do this if mobil move has a
  threat, eg. is a capture or has own move goal.

The following was removed because: since SQ3 is not overprotected, then the
first move of the two move attack will likely make SQ3 safe.
Thus do not care about exchange (hopefully).
(NEVER (FUNCTION EXCHANGE P1 SQ3))
That line was removed from both M2SAFE and M2UNSAFE.
(This occurrred while doing pos90 after NxNP, KxP, and Q-B4)
(Not sure what pos caused M2USAFE in first place but should try others.)
MOVE2
M2USAFE: when opponent can exchange P1 on SQ3 (whether safe or not)
  then allow 2 move attack with SAFER precondition which will try to
  prevent exchange. PRECOND value must be recognizable goal
```
((P1 SQ2 SQ3)
(PATTERN BACK1 P1 SQ2 SQ3)
(FUNCTION EXCHANGE P1 SQ3)
(ACTION MOVE (P1 SQ2) (THREAT (GOAL THREAT))
 (LIKELY (ADD1 (GOAL1 LIKELY)))
 (PRECOND (SAFER P1 SQ3))
 (PLAN NIL (SAFEMOVE P1 SQ3) (GOAL PLAN))))
```

~nothing done about unpinning P1 so it can move.

Perhaps should recognize when first move of 2 makes threat (ie has other
move or move1 or plan1 goal for it). But combination gives likely 0 with
forkt of threats which means we're subsumed by ones which will be likely 0
plans anyway. But what we want to see threat of this move in case where
it may not be recognized anyway. For example, we don't recognize just

an attack of a piece we can capture or an attack of a piece which protects something as a threat (since they can escape). But if we ca do this with first move then perhaps we should have likely 0 with a forkt threat!

DECOYSUP (old)
REMOVED FOR DECOY KS WHICH DOES IT BETTER (SAFE1 THING BELOW DONE)
this attempts to make a SQ safe by decoying enemy pieces. It uses MOBIL
moves that safely attack a major enemy piece that hopefully can only by
answered by moving a piece which protects the square to be made safe.
There could be a UNIQUE around DIR DP1 SQ to assure this, but since
the move is MOBIL there aren't going to be many ways to stop it and the
UNIQUE misses the case when you'd be forcing him to start the exchange will
an unfavorable piece (eg Q).
DMP1 and DP1 are assigned in parallel. Avoids checking one for all instantiations
of other. SAFECOND left til end since it may changes atrls in goal and the
effect of this would be lost in the parallel section.
Note that DP1 can be DMP1. Also second part of OR (THRU) should
really be in SAFE1 and not SAFER but I didn't want to use the prod twice.

```
((P1 SQ P2 SQ1 DP1 DMP1)
(PATTERN LEGALMOVE P1 SQ)
(PATTERN MOBIL P2 SQ1)
(FUNCTION NEQ P1 P2)
(FUNCTION NEQ SQ SQ1)
(NEVER (FUNCTION LINE (LOC P1 BD) SQ1 SQ))
(NEVER (FUNCTION OVERPRO (COLOR P1) SQ1))
(PARALLEL

((PATTERN DIR DP1 SQ1)
(OR      (PATTERN DIR DP1 SQ)
         (EXISTS (LP1) (PATTERN THRU LP1 DP1 SQ)
                 (NEVER (FUNCTION LINE (LOC LP1 BD) SQ1 SQ))))
(NEVER (FUNCTION CANATTACK DP1 SQ1 SQ))
(NEVER   (EXISTS (DP2)
         (PATTERN DIR DP2 SQ)
         (FUNCTION NEQ DP2 DP1)
         (FUNCTION ASVAL P1 DP2)))
(NEVER (EXISTS (P3 DP2) (PATTERN PIN P3 DP1 DP2) (FUNCTION ASVAL DP2 P2))))

((FUNCTION CANATTACK P2 SQ1 (LOC DMP1 BD))
(NEVER (PATTERN DIR P2 (LOC DMP1 BD)))
(FUNCTION SAFEP P2 (LOC DMP1 BD))
(PIECESAFE P1 (PLUS (EXCHVAL P2 (LOC DMP1 BD)) (EXCHVAL P2 SQ1)))))

(CHECKCOND SAFECOND)
(PLAN ((P2 SQ1) (GOAL PLAN))
  (THREAT (FORKT (EXCH P2 DMP1) (PLUS (GOAL THREAT) (EXCHVAL P2 SQ1))))
  (PLAN)))
```

MOVE1

This KS combines many atrls of MOVE goals into single plans or SAFE and SAFER goals. Some of these many atrls may have preconds.
Handles threatening discovering moves and will also decoy an enemy piece blocking the line of the move.

```
M1DECBLK: if DP1 blocks move then set up goal of decoying him if SQ3 looks
 safe or if DP1 bears on it (since decoying may make it safe then).
 DECOYCOND prevents decoy from taking place on sq that would also block.
M1:if P1 can stay on SQ3 then select atrls with no precond
 and suggest plan with least likely atr and combo of plans in goal.
 (Check for no 2 move exchange -- if there is one M1USAFE will make safer goal)
M1DPIN: if P1 can move to but not stay on SQ3 and by so doing can
 uncover a pin of the piece which protects SQ3 (and is less valuable than P1)
 then suggest plan of moving P1 to SQ3 followed by taking the pin object
 if the pinned piece moves, else continuing plan.
M1EX: if P1 safe on SQ3 but can be exchanged then SAFER goal to stop exchange.
M1PREWIN: if P1 can stay on SQ3 and all have preconds and moving P1 to SQ3
  solves a precond then suggest PLAN for it and eliminate atrl with solved
  precond (foreachgoal does it).
  Usually P1 will equal P2 (ie solve precond for self.)
  Makes plan for case when move captures only piece protecting PRECOND square.
M1PREWN2: like above but suggests plan when move captures a piece and this
  allows both moving piece and a friendly thru piece to bear on the precond
  square which must be protected only once and by a more valueable piece
  than the one being moved.
M1PRE: if P1 can stay on SQ3 and all have preconds  then make goals of preconds.
M1UNSAFE: if not exchangable and not safe then set up
  SAFE goal for those with no precond (forget precond ones).
M1USAFE: if exchangable (whether safe or not) set
  up SAFER goal to prevent exchange for those with no preconds.
  (if mobstays get here then tere is 2 move exchange so ok it even if no exchange)
(above 2 do not check overpro since this done in safe and safer has prod that works
for some overpro ones -- could check overpro2 if desried)
```

This prod makes system work for pos3:

```
M1DPIN
((P1 SQ3 LP1 DP1 DP2)
(PATTERN LEGALMOVE P1 SQ3)
(NEVER (PATTERN MOBSTAY P1 SQ3))
(REMATRS T ((PRECOND NIL)))
(UNIQUE (PATTERN DIR DP1 SQ3)
        (FUNCTION ASVAL P1 DP1))
(PATTERN THRU LP1 P1 (LOC DP1 BD))
(NEVER (FUNCTION LINE (LOC LP1 BD) SQ3 (LOC DP1 BD)))
(FUNCTION CANMOVE LP1 (LOC DP1 BD) (LOC DP2 BD))
(FUNCTION SAFEP LP1 (LOC DP2 BD))
(NEVER (FUNCTION EXCHANGE LP1 (LOC DP2 BD)))
(FUNCTION MOREVAL DP2 P1)
(PLAN ((P1 SQ3) (((DP1 NIL) (LP1 (LOC DP2 BD))) ((GOAL PLAN))))
        (LIKELY (GOAL1 LIKELY)) (THREAT (GOAL THREAT))))
```

(do return after it and put before M1SAFE,UNSAFE)

SAFE

Takes Pl and SQ as args. Tries to make it safe for Pl to land on SQ. Expects threat, likely attributes, and plan attribute with plan that begins with opponent's move after any safe making moves. Plan usually has (safemove pl sq) in it.

Threat is for what will happen after Pl→sq. Here we add any threat made in making it safe. Perhaps should also add any threat made by capturing the piece on sq (if one exists) but this is very hard since we don't know how adding supports, etc. will affect occup. Since it's not safe now we'll just assume that we've got back to neutral by helping, it some cases may actually be winning piece on sq so perhaps we might add some to threat for this. (Especially applicable on prods where we've forced one of their pieces away.)

SAFE1 KS tries any thing to make move SAFE and SAFER KS only tries things that will prevent exchanges (they also make it safe.)
Thus, SAFE1 should contain only things that will not help when we are in an exchange situation. All other things sould be in SAFER.
Basically SAFE1 has prods which add one more support for offense which does not help if they can exchange anyway.
It is assumed that SQ is not overprotected--else this is all worthless.
(But overpro only works when Pl bears on SQ, if not and overpro then require some piece on Pl's side to bear dir on sq).

SAFE1

SUPI
this supports SQ if support will help. If supporting move captures a piece then better threat reflects this.

SUPCHK
like SUPI, supports SQ except does it by attacking thru enemy king.
It does it even if opponent has lower value piece attacking sq on the theory that a check is a perturbing thing that may change things.

DSC PRODS
(Pl must be here so deducolor can decide how to instan vars)
Done differently here than in ATTACK KS. There we know that actually uncovering attack is winning thing so never increase likely. Here we may just be uncovering support so we don't know anything is threatened. Suggest uncovering moves which capture or attack pieces even if not mobil (if threat good enuf and likely 0) and keep same likely. If none of those then suggest any mobil ones but increase likely. (Perhaps should prefer ones that block interposing line?)

DECBLK
If we have a support that is being blocked, set up goal of decoying blocker.
Only do if Pl to SQ is legalmove since such involved things as decoying don't work well a few moves ahead. DECOYCOND makes sure decoy does not still block.

SAFER

SRCOND
Does all sorts of horribles if tries it for any square so make sure not overpro first: but his only good test when P1 bears dir on sq, if not and it's overpro then require someone on P1's side bear dir on sq. This no longer tested as condition since decoysupl does number that wipes out two enemy pieces which may work on overpro squares so try it first with overpro2 test and then kill ones that are overpro.

CAPTURE
Takes P1 and DP1 as args. Attempts to capture DP1 with a MOBIL move but does not use P1 to do it.

WINEX1
This starts an exchange on SQ when we can safely do i.

UNPIN1
this tries to unpin piece protecting Sq to be made safe; not used and no UNPIN KS.

DECOYSAC
This says if there are two pieces that can both do damage on SQ and only one opponent piece attacking SQ then sacrifice one piece on SQ to get second piece there if threat warrants it. Ideally would like to know if first piece is necessary for threat of second piece (eg guards king flight squares). Think about how to do it.

MAKEFLY
attack and drive away opponent's piece that protects sq
ok to use piece we're tring to make safe if we can still move to Sq and if he can't exchange us, even with two move exchange
Make it likely 1 if def still has lesser piece which can recapture P1 (else in pos90 will play r-q1 to make q6 safe for wb). Only time would want this likely 0 is when we intend to make 2 pieces fly and more valuable one must be done first (this is unlikely situation).

CAPBLOCK
Meant to set goal of capturing dp1 when it is true
that some friendly piece would bear on sq (the goal) except for this dp1
and an indefinite number of his own pieces in the way.

```
(       (FUNCTION INVUE LP1 SQ)
        (NEVER (PATTERN DIR LP1 SQ))
        (NEVER (EXISTS (AP1) (PATTERN THRU LP1 AP1 SQ)))
        (UNIQUE (FUNCTION LINE (LOC LP1 BD) (LOC DP1 BD) SQ)
                (NEVER (FUNCTION INVUE DP1 SQ)))))
(NEVER (FUNCTION EQ P1 LP1))
(ACTION CAPTURE (DP1 P1) (THREAT (GOAL THREAT)) (PLAN (GOAL PLAN))
 (LIKELY (GOAL1 LIKELY))))
```

SAFER

DECOYSUP
New prod tries so decoy enemy pieces bearing on sq, but only when Pl can move there (see SAFE1). Also it's not tried if there is another piece which can still exchange on SQ.
perhaps should try decoys even if overpro since can keep decoying pieces til none left -- e.g. M39 in Pitrat's paper. Or at least try any mobil decoys all the time, and maybe any decoys when not overpro2.

Actually if Pl is onlypro then don't want to decoy when opponent has second piece unless decoy is a mobil move. Done with DS1,2. Else sac queen to remove support when he has Q,R and K all bearing on SQ and we're trying to get B on SQ with no other supports.

BLOCKP
;old decoysupl says decoy one piece into path of another (GONE NOW)

BLOCK2
idea is to gain two protections thru blocking of DP2's line to SQ.
blocking done on SQ1. either blocking piece must be safe and ca sq from sql or all safe recaptures must not ca Sq from sql and must be pieces that bore on SQ.

DECOYSAC
works on 1. Last or wants to do piecesafe on pl and threat from p2's safe goal, but use this approximation (better and more expensive would be to look for pieces attacked by p2's move whenever pl en prise.

DECOY KS (DP1 P1 SQ)
This tries to decoy DP1 to help make P1 to SQ more plausible (safer or possible). Expects threat, likely attributes, and plan attribute with plan that begins with opponent's move after any safe making moves. Plan usually has (safemove p1 sq) in it. DECOYCOND may put requirements on SQ1 (where we'll move to decoy dp1) and on DP1. (Usually: SQ1 not on certain line, and DP1 can't attack certain sq from SQ1).

NOW DONE IN FORCE
Threat is for what will happen after P1→sq. Here we add any threat made in making it safe. Perhaps should also add any threat made by capturing the piece on sq (if one exists) but this is very hard since we don't know how adding supports, etc. will affect occup. Since it's not safe now we'll just assume that we've got back to neutral by helping, it some cases may actually be winning piece on sq so perhaps we might add some to threat for this. (Especially applicable on prods where we've forced one of their pieces away.)

Examples: N-N5 decoys N to help Q protect D7 in pos78.
QxP decoys B to help R mate in pos68.
Q-f8 decoys R to help R mate in pos27.

MOVDEC
This pattern finds a move P2 to SQ1 that should decoy DP1 if it threatens something. Of course, don't decoy with P1 or block his line to SQ. In fact, there may be other pieces than DP1 which can capture on SQ1 but the OVERPRO clause throws out obvious losers. We could require DP1 as the only DP bearing on SQ1 but that would miss the case when DP1 is the optimal piece to deploy to SQ1. (hope OVERPRO throws out losers)
Don't decoy DP1 to place where
he can attack SQ (must check if canattack thru current location).
Require that we threaten more than we lose by leaving P2 hanging (since we don't plan to recapture).
Also require that P2 not be protecting SQ or we'll be defeating the purpose.
If DP1 is pinned to king or piece more valuable than P2 and threat then don't try decoy. (he can't or won't take it)

This now says either never dir p2 sq or ethru p1 dp1 sq.
If never dir, then cannot decoy to object square (i.e. SQ cannnot be SQ1).
The only time we want SQ=SQ1 is when P1 bears ethru piece we're decoying-- then decoying DP1 to SQ is great (if you move to SQ and he doesn't take decoy then you're very awkward). To decoy to SQ in non-ETHRU cases: WINEX should do it for a safe move, and DECOYSAC should do it for non safe move.

WANT TO USE SPECIFIC RESPONSE (DP1 SQ1) SO THAT DON'T STRING INFINITE BUNCH OF DECOYS TOGETHER IN SEARCH WHEN HE JUST AVOIDS EACH ONE -- NECESSARY BECAUSE CURSEVAL DOES NOT TRY BETTER HIGHER WHEN PLANFROMTOP IS TRUE.

SPLIT
Recognizes case when only threat of P2 to SQ1 is against DP1 itself.
(Use "only" and DSP1 to avoid redoing what FORCE will do.)
Misses case when decoying pawn when one other piece is attacked.
This is threat only if DP1 cannot flee to useful place.

SPLITD
When DP1 bears dir on SQ, approves split when he no longer does.

SPLITB
When DP1 blocks P3 from SQ, approves split when he no longer does.

FORCE

(pl sq p2) pl moves to sq as first move in plan and prepares a move of p2.
FORCE expects plan after pl→sq. Expects threat which doesn't take into account move of Pl to SQ. Expects DECOY atr which is name of piece. This used to ok threat against that piece and to require some threat before approving. Uses p2 to see if piecesafe is ok for it.

OK checks as is, just add on plan and threat. If not decoy then ok as is if mobil with 1 added likely (get good threat here since forkt later may lessen threat). Then find all threats made by Pl sq. If none then ok it if its a capture. If some then ok it and make list for all of them -- threat also takes into account any possible capture.

FTHREAT figures we got a threat when we attack someone that we don't already and we can safely capture him. If he can take us back, then he must be piece we're decoying or we must be mobil. P2 must be piecesafe.

DECTHREAT → FTHREAT
This finds DMPls that are advantageously attacked by P2 to SQ1. Don't count one that can take us right back (unless they are piece to be decoyed) since this is obvious loser. (except they are more valuable than P2 and can be recaptured: eg in 52 get Q for R and N).

In the following two prods would want plan to specify recapture instead of NIL for defense. This would involve taking cdr of plan in atrl which is fine but bigger problem is when plan is RA list, then must return list of things each with cdr removed so this no done.

CAPONLY
If no attacks but it's a capture then approve it.

DECOK
Now we know there is a DMPl threatened. We use SETl so only one atrl made for all DMPls found.

Problem in 78 on N-D6, cuz whenever we want to make a square safe, we just attack his queen in such a way that it can take us then this is oked as decoy. Such a plan should only be used if queen cannot flee to square from where it can maintain protection (makefly). New prods added to DECOY to do this.

Now require that there be two pieces attacked when doing decoy.
DECOKl does all non-decoys together to avoid ineffieciency.
For threat, really want only second best of attacked pieces since he can flee with best one but used this way to keep optimistic since best one might protect second best one or perform some other function we don't know about. (Perhaps post Can-Make-FLee concept for this move to use if we have decoy or so).
Important to use exchval, with exch then in tree plan keeps thinking it can win that best one when it has already fled.

Here is a case of possible non-optimistic threat evaluation:
If threat is (exchval pl sq3) or (exch pl dp3) for some DP3 or SQ3, then if we decoy a piece that was bearing on SQ3 away so that he no longer does then we will do better than this exchval shows.
Not sure how to correct this.

DEFSAFE

Takes P1 and SQ as arguments. Suggests moves by other side to stop P1 from moving to SQ or to blunt the effect of the move.
Expects SAVE atr which it passes on, else PLAN1 eliminates possible best moves (eg sacrifices to stop mate).
Also expects LIKELY and THREAT and PLAN and COND which are all passed on.
When suggesting MOBIL moves, a DEFTHREAT goal is posited so any threat a move might make will be considered in ordering.

BLOCK
suggests MOBIL moves which block the move

BLOCKSAC
if no mobil blocks then suggests non mobil ones if they lose less than SAVE.
Right now only captures and blocks are done without mobils since they are strongest counters to stopping something.

CAPTUR
tries to capture pieces which bear DIR, ETHRU, or OTHRU on SQ and pieces which have P1 as their only support. Uses MOBIL moves and legalmoves where capturing piece minus captured piece must not exceed SAVE of goal.

RUNP
if SQ occupied any P1 is MOBSTAY on it, then set up goal of RUN for piece on SQ.

QUIT
stop here if king is in check

BLOCKSUP
suggests MOBIL moves which block P2 from SQ when P2 bears ETHRU or OTHRU on SQ

PROTECT
suggests MOBIL moves which add another support for SQ unless king is on it

DISCOV
suggests any MOBIL move which uncovers an attack on SQ.

PINDEF, FORKDEF
recognize if P1 will fork or pin from SQ and sets up goals to move the pieces which will be pinned or forked. Technically, should have condition that they will move off line of threat--(could be implemented by having COND atr and passing that atr to DIRMOSTMOB in RUN KS)
In forkdef just need one run goal since both halves of fork will match as DMP1.

RUNCHK
if K is threatened then start moving him early looking for an escape route

DEFTHREAT TERM DEFOFF

Takes DP1 and SQ1 as arguments. Assumes this is MOBIL move to be made.
Checks COND which may put constaints on DP1.
Creates plan by putting DP1 to SQ1 on front of plan in goal.
Expects SAVE and LIEKLY atrs which it passes on.
Gives threat of 1 or exch value when move captures, threat of 1 when it threatens a piece, else nothing. (adds to existing threat)

LOSE
This helps reject moves that make his ep caps even more threatening.

TERM

This finds threats for quiescence evaluation. Only finds ones that are sure to work. There is a crock on the fork one since we can't have conds in attribute specification.
If uniquedir true for dmp1 then want (win dmp1) instead of exch.
Same for DMP2. To be accurate need four prods for four combinations of uniquedir and must keep other three from firing when anyone does.
To avoid this, just make both be WIN if one is (optimistic).
Good place for optimism since we are capturing and threatening at same time.
Does not set up PLAN1 attr since epeval does own defense (thus needs own ep prod).

Exch must be used to get optim evaluation in enprise case.

MOBTERM keeps from being overoptimistic when thinks it can do a trip from SQ since its mobstay when actually it can only exchange itself for piece on sq. Could use in threat also but don't need to be so accurate there.

FORKREG tries to check that one piece cannot adequately defend other. (e.g. in 78 after Ng5 NXN Rd7 QXR QXQ many forks of white queen with black king and b (or n) where king can protect b after wq leaves rank 7.

TWOCHECK and TRAPMATE borrowed from THREAT to recognize patterns where there might be a mate around but it's not sure enough to try the move.

DSC1 captures on uncovering move found by enprise.

DEFOFF

Just like DEFSAFE except only uses some of its prods (leaves out very defensive ones, keeps more offensive ones) and replaces RUN with prod that's more particualar about what it runs.

RUN DEFENSE DEFEP

This KS is used to move a piece when it's in danger where it is. Essentially a defensive KS. It assumes the goal has a SAVE attribute which tells how much will be saved by moving the piece to a safe square. (Cannot use threat since threat should give an expected board value, which will still be same. But must have SAVE so we will try move even if it gives up something (PLAN1) as long as what it gives up is less than SAVE.
It's arguments are DP1 P1 and SQ1. It suggests moves for DP1 so that P1 will not attack it from SQ1.

A move that captures a piece is given a positive threat. Moves that attack or protect a vulnerable piece and the move with the most mobil moves on the next ply are given a threat of 1, all other MOBIL moves are suggested with threat of zero. If a king must run then all legalmoves are suggested since we don't care how much we lose saving the king.

DEFENSE

This used to help better differentiate defensive suggestions. HELPCAP improves threat of any capture when offense has a piece hanging elsewhere.
(e.g. after r-d6 in 78, taking R not bad since wq is ep)
Cannot use enpris pattern since don't match when king in check (kept in as or to get discovered en prises).

ALLOWFRK increases loss when moving to place where simple fork can be made.
Really only want to do this in epeval so make DEFEP KS for that.
(This is worth it since it's expensive prod.)
Helps get right king move in G0822.FIE from 78. It does not recognize forks where attack line goes thru moved piece -- this would be easy addition if needed.

```
PLAN1

checks for giving up functions.
If nothing given up, or if checking then ok it.
If we are threatening or saving more than we're giving up,
then decrease threat by what we're giving up:
This done be keeping separate LOSS atr and using THREATLOSS to eval threat
by using WIN instead of EXCH (since def cannot exch if he's doing another threat).
Important that each prod matches only once for all giveups, else atrls are
grossly large.  Note that matching GIVEUP implies that FIRSTMOVE already matched.
SET used to form loss atr so prod only matches once.
We test to see which give ups will still be so.

GIVEUP
there are 4 giveups: pin, ep, protects, mates
        For ep must check that we're not capturing necessary support to
        make the ep capture.  Thus, disallow when capturing a supporting piece.
        This is not entirely accurate but errors in optimistic way.  Actually
        want to know if support necessary for the ep (something occup does not
        tell us currently) but this would then have to be checked for case when
        P1 bears on ep square from SQ.  Thus just do this optim approximation.
        BETTER approximation now: if nobody protects ep piece and we haven't
         captured only support then allow it anyway (but not if capturing piece
         bears on ep piece from sq since then shift of two supports will
         mean exchval is way off)  Does following:
        BR---WR-------BR  Want BxR to giveup enprise rook.
             \     /
              BB WQ
        This was wrongly analyzed as needed to see that RxR save rook (ie have
        SAVE KS to add on SAVE for such goodies --problem is that save values
        would overpower good forcing moves which don't give opponent chance to
        recapture his threat.
there are 4 refutes: block line, capture tpiece, cap piece which tpiece protects
        so that captured piece better than losing object,
        threaten piece (not tpiece) so that capturing it better than losing object
All refutes apply to all giveups.  However, in case of last two refutes,
        value of object will be whole value for protects but only exchval
        for ep,pin (either for mate)
Also mate needs SQ1 to threaten, rest don't but use it as P2's loc.
Thus DP1 threatens to win P2 by landing on SQ1 (P2's loc except when mating).
To accomodate difference between value and exchval, one pat matches ep,pin and
 their 4 diff conds, another matches protect,mate and their 4 conds (mate here
 since needs some of same tests as protects).
Will combine 2 pats into one -- first two refutes could be here but they aren't
 long and they eliminate matches where they are.
Actually this SQ1 shit should go into PROTECTS and have it recognize protection
 of matesqs also!!
This is easy -- protects only used in PLAN1  capdecoy and in refuttes and giveups.

!!!Actually, when protects is for ep capture, it does not protect the value of the
ep piece, but only threatens to get the value of the piece that will capture --
so should use minimum of ep piece and capturing piece as threat.!!!!
To do this need a COND in threatok so it's been left out.

Fixes proposed are not checked for giving things up since that involves
infinte regress.

This removed since has nothing to do with P1 and SQ.  If needed should be in
threat where you suggest moving pieces that are ep, etc.
CHKFIX
((P1 SQ P2 SQ1 DK)
(EXISTS (DP1) (PATTERN GIVEUP P1 SQ P2 DP1))
(PATTERN CHECKMOVE P2 SQ1 DK)
(PATTERN MOBIL P2 SQ1)
(FINALPLAN (FIX GOAL ((P2 SQ1) NIL))
 (SAFECOND) (THREAT (PLUS (GOAL1 THREAT) (EXCHVAL P2 SQ1)))))
```

To make threat optimistic we must realize that if he plays move to get loss then we can take another piece with P1. This is subtracted from LOSS. Note that we can't capture piece that he makes loss threat with-- to be accurate must match subtracted loss with other loss (done by having giveup concepts where loss expressed as atr and the SET must access concept to form list) but instead just don't allow capture of any piece that has a giveup threat.
This done in THREATOK only since capfix and capdecoy are weird anyway.

CHECK
Oks without loss if check -- could check for giveups having dp1 as dk but then must only use these giveups when forming loss

THREATOK
Does trip (per above) of getting threats p1 makes and adding them into loss, checking that loss not to big (changes obvious giveups to wins.

REJECT
This sets up goal to fix defense's refutation if plan rejected.

CAPFIX
Uses forkt of threats which is not optimistic, but using plus would be a likely 1 plan and as a likely 0 produces poor plans with huge threat values.
 (eg in 78 after N-N5, K-e7 it plays QxP(g6) with plus and likely 0)

SAFEMOVE

This takes move as argument. If its safe then suggests plan of moving it, else sets up move goal to try and accomplish it.

KILL

Takes 2 pieces as arguments. If first can capture the second then suggests this, otherwise forget it. When doing forks and so on, this allows plan to specify a capture without the system solving the goal of bringing it about when the piece to be captured has moved.

```
DESPERADO
ATTROYAL
((SP1 SQ DR1)
(PATTERN MOBIL SP1 SQ)
(FUNCTION CANATTACK SP1 SQ (LOC DR1 BD))
(NEVER (PATTERN DIR SP1 (LOC DR1 BD)))
(PLAN ((SP1 SQ)) (THREAT (WIN DR1)) (LIKELY 0)))
```

This KS removed in favor of TERM KS. This production forgottern as unmotivated.