Procedure Backtrack(DATA)

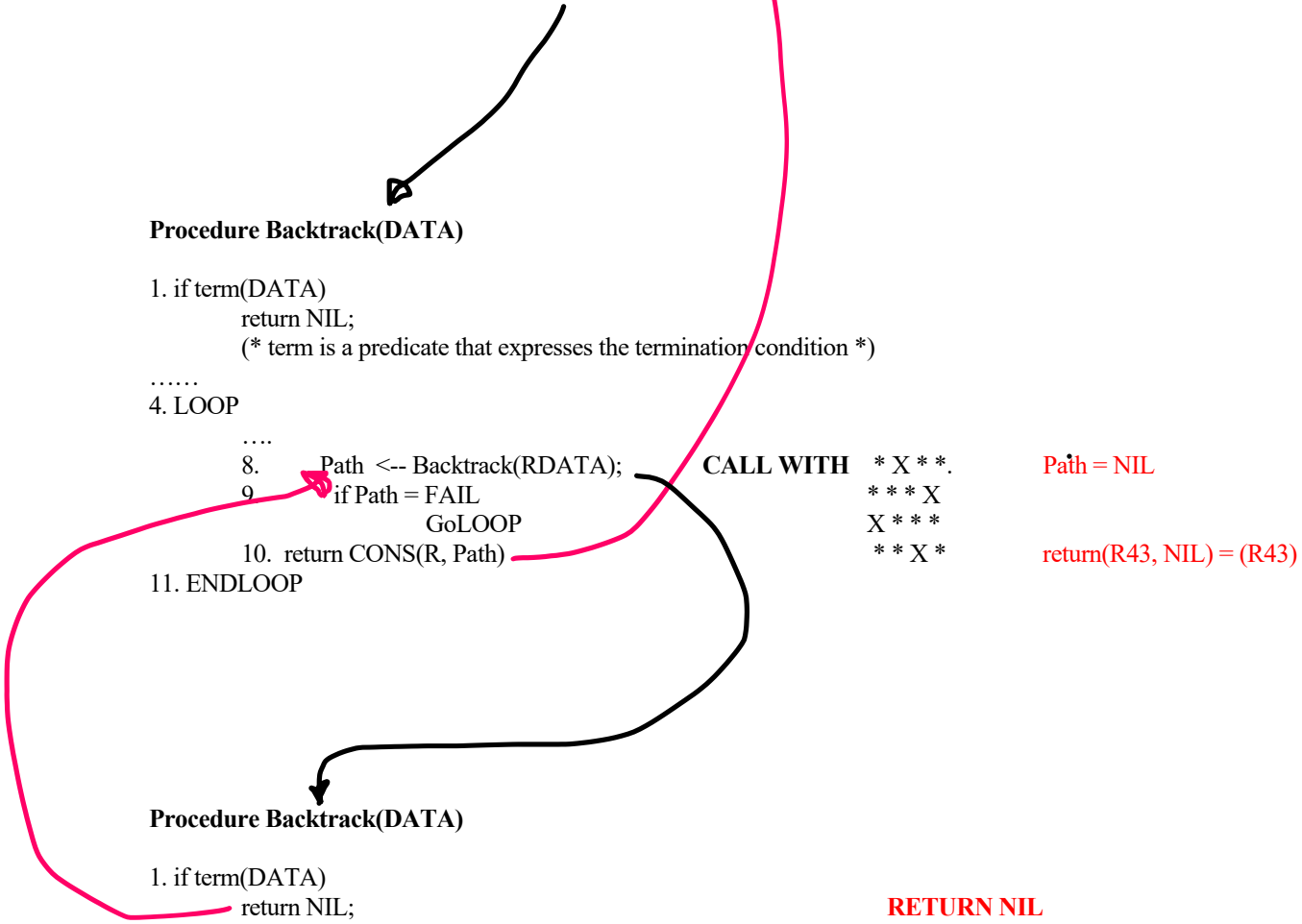
```
1. if term(DATA)
        return NIL;
        (* term is a predicate that expresses the termination condition *)
……
4. LOOP
        ….
        8.      Path  <-- Backtrack(RDATA);     CALL WITH  * X * *   Path = (R24,R31,R43)
        9.       if Path = FAIL                              * * * *
                        GoLOOP                               * * * *
        10.  return CONS(R, Path)                            * * * * returnCONS(R12, (R24,R31,R43)) =
11. ENDLOOP                                                 (R12, R23, R31, R43)
```

Procedure Backtrack(DATA)

```
1. if term(DATA)
        return NIL;
        (* term is a predicate that expresses the termination condition *)
……
4. LOOP
        ….
        8.      Path  <-- Backtrack(RDATA);     CALL WITH * X * *.      Path = (R31,R43)
        9.       if Path = FAIL                             * *  * X
                        GoLOOP                              * *  * *
        10.  return CONS(R, Path)                           * *  * *   returnCONS
                                                                (R24, (R31,R43)) = (R24,R31,R43)
11. ENDLOOP
```

Procedure Backtrack(DATA)

```
1. if term(DATA)
        return NIL;
        (* term is a predicate that expresses the termination condition *)
……
4. LOOP
        ….
        8.      Path  <-- Backtrack(RDATA);     CALL WITH  * X * *              Path = (R43)
        9.       if Path = FAIL                             * *  * X
                        GoLOOP                              X*  * *
        10.  return CONS(R, Path)                           * * * *     returnCONS(R31(R43) = (R31,R43)
11. ENDLOOP
```



Procedure Backtrack(DATA)

1. if term(DATA)
   return NIL;
   (* term is a predicate that expresses the termination condition *)
……
4. LOOP
   ….
   8. Path <-- Backtrack(RDATA); **CALL WITH** * X * *. Path = NIL
   9. if Path = FAIL * * * X
      GoLOOP X * * *
   10. return CONS(R, Path) * * X * return(R43, NIL) = (R43)
11. ENDLOOP

Procedure Backtrack(DATA)

1. if term(DATA)
   return NIL; **RETURN NIL**
   (* term is a predicate that expresses the termination condition *)
……
4. LOOP
   ….
   8. Path <-- Backtrack(RDATA);
   9. if Path = FAIL
      GoLOOP
   10. return CONS(R, Path)
11. ENDLOOP

PLEASE NOTE

Black lines indicate a recursive call with the current DATA (a board configuration).

Red lines indicate a return from a copy of the procedure being currently executed (I have skipped some steps of the algorithm, namely backtracking steps and checking for deadends.

Path in red font indicates the current path upon each return (also in red)