SETL Newsletter no. 185

Leonard Vanek

March 28, 1977

USING OUTPUT FROM THE SETL COPY OPTIMIZER

The copy optimization phase of the SETL optimizer transmits the results of its analysis via "copy state" descriptors attached to each argument of the quadruples of intermediate code. These descriptors are accessed via the COPYFLAG map, which maps variable occurences to one of five values indicating how the code generator is to emit code for the quadruples. These five copy states are:

NOCOPY - no copying or share-bit manipulation is done

PRECOPY - copy before a destructive use

POSTCOPY - copy after a value transmission operation

SETSHRBIT - turn on the share-bit after a value transmission operation

TESTSHRBIT - if the share-bit is on, copy before a destructive use

For the first and second arguments of a quadruple, there exist opcodes which allow any of these states to be realized. For the third argument - which can never be used destructively - PRECOPY and TESTSHRBIT will never apply.

If for some argument (occurence) A, COPYFLAG(A) = POSTCOPY then after the execution of the instruction containing A, A must have an isolated value. That is, A must be a new (unshared) copy of its value. The same is true when COPYFLAG(A) = PRECOPY, however in this case there is an added requirement resulting from the fact that A is being used destructively. The copy operation must be carried out before the instruction is executed.

If COPYFLAG(A) = SETSHRBIT, the copy optimizer has concluded that A may ultimately need to be copied but the decision to copy can be postponed until run-time by using the share-bit mechanism [NL164]. In this case (which is the conditional form of POSTCOPY) after the instruction is executed the share-bit of A is turned on; otherwise, the share-bit retains its original value.

If COPYFLAG(A) = TESTSHRBIT it is possible that A may be used destructively in safety, but only a test of A's share-bit will confirm or deny this. Hence, if the share-bit is set the value of A will be copied before A is modified. This is the conditional form of the PRECOPY condition.

Finally, if COPYFLAG(A) = NOCOPY, A can be used directly with there being no need to copy it or make use of its share-bit.

The following examples illustrate how the code generator will use the COPYFLAG.

Examples -

```
                    A = B with C
```

1. Copyflag(B) = PRECOPY   copyflag(C) = POSTCOPY

```
T = B
B = copy(B)
A = T with C
C = copy(C)
```

2. Copyflag(B) = TESTSHRBIT   copyflag(C) = SETSHRBIT

```
T = B
if sharebit(B) then B = copy(B)
A = T with C
setsharebit(C)
```

3. Copyflag(B) = NOCOPY   copyflag(C) = NOCOPY

```
A = B with C
```

No other copy states are possible for this opcode.

```
                    F(X) = Y
```

```
4.  Copyflag(F) = PRECOPY   copyflag(X) = POSTCOPY
                            copyflag(Y) = SETSHRBIT

                    F = copy(F)
                    F(X) = Y
                    X = copy(X)
                    setsharebit(Y)
```

Note that there is no need to use a temporary T when the PRECOPied argument is the first (output) argument. This is because for all such cases, the operation is a sinister assignment, which kills the old value of its first argument.

```
5.  Copyflag(F) = TESTSHRBIT
    Copyflag(X) = copyflag(Y) = NOCOPY

                    if sharebit(F) then F = copy(F)
                    F(X) = Y
```

```
6.  A = B       copyflag(A) = copyflag(B) = POSTCOPY

                    A = B
                    A = copy(A)
                    B = copy(B)
```

```
7.   A = arb B          copyflag(A) = POSTCOPY

                          A = arb B

                          A = copy(A)
```

For this operation B is neither modified nor transmitted, so its copyflag will always be NOCOPY.

The following fragment of a SETL program summarizes the operation of a code generator which uses the copyflag to minimize copying operations. Regardless of the semantics of the opcode of the quadruple being processed, the quadruple will be written as "A = B op C".

```
case copyflag(A) of

(PRECOPY):      emit(A = copy(A));

(TESTSHRBIT):   emit(if sharebit(A) then A = copy(A));

else            noop;

end case;


case copyflag(B) of

(PRECOPY):      emit(T = B);
                emit(B = copy(B));
                emit(A = T op C);

(TESTSHRBIT):   emit(T = B);
                emit(if sharebit(B) then B = copy(B));
                emit(A = T op C);

else            emit(A = B op C);

end case;
```

```
case copyflag(A) of
(POSTCOPY):     emit(A = copy(A));
(SETSHRBIT):    emit(setsharebit(A));
else            noop;
end case;

case copyflag(B) of
(POSTCOPY):     emit(B = copy(B));
(SETSHRBIT):    emit(setsharebit(B));
else            noop;
end case;

case copyflag(C) of
(POSTCOPY):     emit(C = copy(C));
(SETSHRBIT):    emit(setsharebit(C));
else            noop;
end case;
```

```
PAGE   LINE  COL  CARD#

   6    35     3    198  MACRO DEFINITIONS         .DM;.EF
                                              77  (  1  '['
                                              93  )  1  ']'
                                             195  C  0   12
                                             196  D  1  'MARCH 29, 1977'
                                             211  L  0   35
                                             215  P  2    6
                                             227  T  1  '11:02'
                                        4036  BYTES REMAIN IN MACLIB.
```