Common Sub-Expression Elimination

Common Sub-expression Elimination (CSE) will be performed over a basic block. The first task is to build a Directed Acyclic Graph (dag) for the 4-tuples in a basic block. A dag is a directed graph that has no cycles. For example, suppose we had the following 4-tuples:

1. $\text{I$1} + a 5$
2. $\text{I$2} + \text{I$1} c$
3. $d = \text{I$2}$
4. $\text{I$3} + c 15$
5. $b = \text{I$3}$

The dag for the above 4-tuples would be

Click [here](#) for the dag.

**Note:**

The leaves of the dag are labeled with either a variable or a constant. Interior nodes are labeled by an operator and annotated by the intermediate result name and/or variable.
dag Construction

1. for each 4-tuple in the basic block do

   1.1 create a node for each operand if one does not exist.

   1.2 if the 4-tuple form is (l$n, operator, operand_1, operand_2)
       then
       1.2.1 if there is a node labeled operator, whose left child is operand_1
                and whose right child is operand_2
                then
                1.2.1.1 annotate the operator node with l$n
                else
                1.2.1.2 create a new node whose label is operator
                1.2.1.3 annotate the new node with l$n

   1.3 if the 4-tuple form is (l$n, operator, operand_1, -)
       then
       1.3.1 if there is a node labeled operator, whose only child is operand_1
               then
               1.3.1.1 annotate the operator node with l$n
               else
               1.3.1.2 create a new node whose label is operator
               1.3.1.3 annotate the new node with l$n

   1.4 if the 4-tuple form is (var, =, operand_1, -)
       then
       1.4.1 append var to the node that operand_1 annotates

Consider the following sequence of 4-tuples

|   | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  | 13  | 14  | 15  | 16  | 17  | 18  | 19  | 20  | 21  | 22  | 23  | 24  | 25   |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1 | l$1 | +   | a   | b   | 14  | l$10| +   | l$9 | c   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 2 | l$2 | +   | l$1 | c   | 15  | d   | =   | l$10|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 3 | c   | =   | l$2 |     | 16  | L15 | Label|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 4 | l$3 | +   | a   | b   | 17  | l$11| +   | a   | b   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 5 | b   | =   | l$3 |     | 18  | l$12| +   | l$11| c   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 6 | l$4 | +   | a   | b   | 19  | c   | =   | l$12|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 7 | d   | =   | l$4 |     | 20  | l$13| +   | a   | b   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 8 | l$5 | +   | a   | b   | 21  | l$14| +   | l$13| c   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 9 | l$6 | +   | l$5 | c   | 22  | d   | =   | l$14|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 10| c   | =   | l$6 |     | 23  | l$16| +   | a   | b   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 11| B$7 | <   | a   | b   | 24  | l$17| +   | l$16| c   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 12| L8  | JPC | B$7 |     | 25  | a   | =   | l$17|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 13| l$9 | +   | a   | b   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
The 4-tuples and dag for the first basic block are

1. $I^1 + a b$
2. $I^2 + I^1 c$
3. $c = I^2$
4. $I^3 + a b$
5. $b = I^3$
6. $I^4 + a b$
7. $d = I^4$
8. $I^5 + a b$
9. $I^6 + I^5 c$
10. $c = I^6$
11. $B^7 < a b$

Click [here](#) for the dag.

The 4-tuples and dag for the second basic block are

13. $I^9 + a b$
14. $I^{10} + I^9 c$
15. $d = I^{10}$

Click [here](#) for the dag.
The 4-tuples and dag for the third basic block are

17. $l$s11 + a b
18. $l$s12 + $l$s11 c
19. c = $l$s12
20. $l$s13 + a b
21. $l$s14 + $l$s13 c
22. d = $l$s14
23. $l$s16 + a b
24. $l$s17 + $l$s16 c
25. a = $l$s17

Click [here](#) for the dag.
4-tuple Generation from a dag

Notice that in a dag that represents a basic block that is terminated with a conditional jump there will be at least two nodes that are terminating nodes in terms of dependencies. The sub-dag that has a terminating node that is directly used in the conditional jump should have its code generated last in the sequence of generating code for the sub-dags within a dag. The basic idea for generating 4-tuples from a dag is that for each sub-dag we will generate code starting at the leaves and working to the node that has no additional dependencies.

4-tuple Code Generation Algorithm

1. For each sub-dag do
   1.1 while there are unmarked interior nodes within the sub-dag do
      1.1.1 Starting at the deepest level of the chosen sub-dag write the 4-tuple for the interior node
      1.1.2 mark the interior node

Consider the DAG below

Resulting 4-tuples

Initially all nodes are unmarked, and we have 2 sub-dags. Choosing the sub-dag whose terminating node is not directly used in the conditional jump, we have the dag
We mark the nodes and generate the following code

1.

2.

3.

4.

Click [here](#) for the 4-tuples.
The second sub-dag that we have is

Since the node whose label is $I_1, I_3, b$ is marked, we do not generate any code for that node. Hence, the only 4-tuple that we have from this sub-dag is

1. $B$\$7 < a b$

The combined 4-tuples for the first dag are

1. $b + a b$
2. $c + c b$
3. $d + b a$
4. $c + d c$
5. $B$\$7 < a b$

or we have recognized and eliminated the common sub-expression $a + b$. Also note that the resulting 4-tuples generated from a dag may yield a better register allocation since the operands are evaluated closer to where they are used than before.
The dag is

\[
\begin{align*}
\text{a} & \quad \text{5} \\
+ & \quad 1S1 \\
\downarrow & \quad 1S2, d \\
+ & \quad 1S3, b
\end{align*}
\]

The dag for the first basic block is

\[
\begin{align*}
\text{a} & \quad \text{b} & \quad \text{c} \\
+ & \quad + & \quad + \\
1S1, 1S3, b & \quad 1S4, d, 1S5 & \quad 1S2, c \\
< & \quad + & \quad + \\
B\$7 & \quad 1S6, c
\end{align*}
\]
The dag for the second basic block is

The dag for the third basic block is

Return
The following 4-tuples are generated from the above dag:

1. \( b + a b \)
2. \( c + b c \)
3. \( d + a b \)
4. \( c + d c \)

Notice that we have really combined two 4-tuples into one, namely we have combined

\[
\begin{align*}
I$1 & + a b \\
b & = I$1
\end{align*}
\]

into

\[
\begin{align*}
b & + a b
\end{align*}
\]