

198:515 Programming Languages and Compilers I

Problem Set 6 Sample Solution

Problem 1

Modify the functions `mk-closure` and `apply-cl` in the statically scoped, call-by-value interpreter “ValueStatic.ss” in order to implement a dynamically scoped, call-by-value interpreter.

ANSWER: Look at file `~uli/cs515/scheme/ValueDynamic.ss` on the graduate cluster (paul.rutgers.edu).

Problem 2

Write Scheme functions to implement TINY’s symbol table (`look-up`, `extend`, `extend*`) using a data representation instead of the functional representation given in class.

```
(define extend
  (lambda (env x v)
    (cons '(,x ,v) env)))

(define empty-env '())

(define lookup
  (lambda (env y)
    (cond
      ((null? env) (error "unbounded identifier"))
      ((eq? (caar env) y) (cadar env))
      (else (lookup (cdr env) y)))))

(define extend* ;; same as in functional representation
  (lambda (env xs vs)
    (if (null? xs)
        env
        (extend* (extend env (car xs) (car vs)) (cdr xs) (cdr vs)))))
```

Alternative: A general strategy is to use Scheme’s notion of association lists. ”An association list is a proper list whose elements are key-value pairs of the form (key . value).” [Dybvig, The Scheme Programming Language, 2nd Edition] If you build this list in `'extend'`, then you can use Scheme’s built-in `'assq'` to easily implement `'lookup'`.

Problem 3

A variable x is *live* at the **end (exit)** of a basic block b , i.e., $x \in LV(b)$, iff

- there is a path in the control flow graph from basic block b to basic block b' such that x is not redefined on the path (definition-free path with respect to x), and
- x is used (read) in basic block b' , and
- if b' contains a definition (write) of x , then x has to be used inside b' before it is redefined (upward exposed use).

1. Give the “generic” data flow equation for a basic block b , i.e., define $LV(b)$.

Just to remind you, here is the “generic” data flow equation for the Reaching Definitions (RD) problem:

$$RD(b) = \bigcup_{x \in pred(b)} (\text{GEN}(x) \cup (\text{RD}(x) - \text{KILL}(x)))$$

Answer:

$$LV(b) = \bigcup_{x \in succ(b)} (\text{GEN}(x) \cup (LV(x) - \text{KILL}(x)))$$

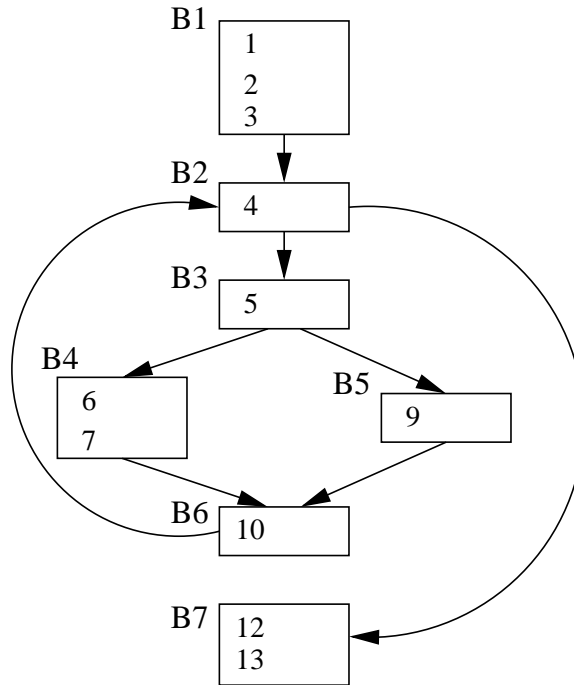
2. Show the control flow graph for the following program:

```

1      A = ...
2      B = ...
3      C = ...
4  L1:  if ... goto L4
5      if ... goto L2
6      B = A
7      A = A + 1
8      goto L3
9  L2:  B = C
10 L3:  C = A + B
11     goto L1
12 L4:  print A
13     halt

```

Answer:



3. What is the universe of data flow facts (data flow lattice) for LV in our example program? Can the elements in the universe be represented as bit vectors? Explain.

Answer:

Subsets of all variables used in the program. In our example, this would be all possible subsets of set $\{A, B, C\}$. The question to answer for each basic block b is: *Is there a possibility that variable X is used before it is redefined after basic block b is executed?* Clearly, the lattice can be represented by bitvectors of length 3. Each vector position is associated with one program variable. If the bit is on ($= 1$), it means that the corresponding variable is life on exit. If the bit is off ($= 0$), it means that the variable is not life on exit.

4. Give the GEN and KILL sets for each basic block. `print A` is considered a use (read) of variable A . Basic blocks without references to variables have empty GEN and KILL sets, i.e., $GEN = \emptyset$ and $KILL = \emptyset$. Note: A data flow fact can be killed and generated in the same basic block.

Answer:

basic block	GEN	KILL
1	\emptyset	{A, B, C}
2	\emptyset	\emptyset
3	\emptyset	\emptyset
4	{A}	{A, B}
5	{C}	{B}
6	{A, B}	{C}
7	{A}	\emptyset

How would you initialize $LV(b)$ for each basic block before running the iterative algorithm to compute the maximal fixed point solution?

Answer:

Initialize of $LV(b)$ to \emptyset .

Give the final solution of the LV problem for each basic block.

Answer:

basic block b	$LV(b)$
1	{A, C}
2	{A, C}
3	{A, C}
4	{A, B}
5	{A, B}
6	{A, C}
7	\emptyset

5. Describe an optimization transformation that uses the LV information. Could your optimization be applied to our example program? If yes, how would the transformation change the program?

Answer:

$LV(b)$ can be used for dead code elimination. If the basic block b does contain a downward exposed assignment to a variable X , and X is not in $LV(b)$, then the assignment to X can be eliminated (assuming that there is no side effect in evaluating the RHS of the assignment to variable X). Therefore, the assignment to variable B in basic block $B1$ (statement 2) can be eliminated.