SETL Newsletter # 138B

J.T. Schwartz
January 6, 1976

Updating the Lower Bound of a Set of Integers in Set-Theoretic Strength Reduction.

Suppose that a is a tuple. Then the set

(1) $$s = \{m,\ 1 \le m \le \#\, a \mid a(m) \in B\}$$

is continuous against small changes

(2) $$a(n) = x$$

in a, to update (1) after a is changed by (2), one executes

(3) $$s = s - \{n\} + \textbf{if } x \in B \textbf{ then } \{n\} \textbf{ else } \underline{n\ell}.$$

In optimizing a program implicitly involving the set s (e.g., one in which the existential quantifier $1 \le \exists m \le \#\, n \mid a(m) \in B$ appears) but not explicitly making any use of set (as distinct from tuple) operations, one may not wish to introduce any set operations; thus, one may wish to avoid explicit use of the set s, even in simple update operations like (3). In dealing with such cases, it may be advantageous to keep the minimum member M of the set (1), or rather some reasonable lower bound for M, available as a kind of coarse estimate for the set s. The update operation for M that corresponds to (3) is

(3') $$M = \textbf{if } x \in B \ \underline{\textbf{and}}\ n < M \textbf{ then } n \textbf{ else } M.$$

This remark is easily extended to apply to sets of integers defined by conditions more general than that of (1). A lower bound for the minimum of the set

(1a) $$\{m,\ 1 \le m \le \#\, a \mid a(m) > b\}$$

is updated after (2) by executing

(3a) $\quad M = \text{if } x > b \text{ \underline{and} } n < M \text{ then } n \text{ else } M.$

If g(m) is a function with an inverse h(m), then a lower bound M for the minimum of

(1b) $\quad \{m,\ 1 \leq m \leq \#\, a \mid a(g(m)) > b\}$

is updated after (2) by executing

(3b) $\quad m = \text{if } x > b \text{ \underline{and} } h(n) < M \text{ \underline{and} } h(n) \geq 1 \text{ then } h(n) \text{ else } M.$

Suppose that the sets (1), (1a), (1b) do not appear explicitly in the program P being optimized, but need to be considered only because of their implicit use in existential quantifiers, e.g., in $1 \leq \exists m \leq \#\, a \mid a(g(m)) > b$. Then we can proceed as follows:

(i) Introduce a lower bound M_E corresponding to each such existential E initialising all the quantities thereby introduced to 1;

(ii) Modify the existential E to begin its search at the lower bound M_E (since by definition the range $1 \leq m < M_E$ contains no m satisfying the condition of the existential E). For example, this means that we modify

$$1 \leq \exists m \leq \#\, a \mid a(m) < b$$

to become

$$M \leq \exists m \leq \#\, a \mid a(m) < b.$$

(iii) Execute M = m, where m is the bound variable of the existential E, immediately after E is evaluated, and provided that E evaluates to <u>true</u> (since the evaluation of E will have established a new, and generally better, lower bound for the set of m satisfying the existential condition.)

(iv) Update M_E in the manner indicated by (3'), (3a), (3b), as appropriate, immediately following any assignment (2) to a.

The optimization defined by (i-iv) can be applied if the only changes to a are assignments (2), and if the form of the condition clause of the existential E is such as to allow an efficient update operation similar to (3'), (3a), or (3b) to be defined. Note that when a is modified in some manner more radical than (2) it may be necessary to re-initialise M_E to 1.

As noted in NL 138, multiple occurences of a in the condition clause of an existential E can be handled by updating for each occurence separately. For example, a lower bound M for the set

(4) $\{m, 1 \leq m < \# a \mid a(m) > a(m+1)\}$

can be updated after the assignment (2) by executing the two statements

(5)
```
M = if x > a(n+1) and n < M then     n else M;
M = if a (n-1) > x and 1 ≤ (n-1)< M then n-1 else M;
```

In the bubble-sort example

```
(while 1 ≤ ∃n < # a | a(n) > a(n+1))
        t = a(n);
        a(n) = a(n+1);
        a(n+1) = t;
end while;
```

application of the general method that has been sketched, followed by the use of various obvious identities and inequalities, can yield the optimized form

```
M = 1
(while  M ≤  ∃n ≤ # a | a(n) > a(n+1))
        t = a(n);
        a(n) = a(n+1);
        M = if n-1 ≥ 1 then n-1 else 1;
        a(n+1) = t;
end while;
```