My Programming Habits

Matt Wette

1 Habits

This chapter explains some of my programming habits. I hope it helps to relieve the head-scratching.

1.1 Modifying Compound Data Types

In general I do not use `set-car!` or `set-cdr!`. I do use `vector-set!` and `hashq-set!`

1.2 A-Lists Versus Hash Tables

In general I prefer a-lists over hash tables in Scheme because a-lists are Scheme-like. To update an entry in an alist I will just paste the new entry on the front. For example,

```
(let ((al '((foo . 1) (bar . 2) (baz . 3))))
  ...
  (acons 'bar 99 al))
```

If the values of the a-list are lists and I want to add something to the list I just use cons, as in the following:

```
(let ((al '((foo 1) (bar 2) (baz 3))))
  ...
  (acons 'bar (cons 99 (assq-ref al 'bar)) al)
```

This modification costs just two cons cells.

1.3 Iteration

For iteration I usually use *named-let* and often in concert with *cond*. The order of variable declarations in my named-let are the result variable, followed by iteration variables in order of slowest to fastest modification. In the cond I usually evaluate in the order fastest to slowest modification. Consider the following C code fragment:

```
res = 0;
for (i = 0; i < ni, i++) {
  res += 100*i;
  for (j = 0; j < nj, j++) {
    res += 10*j;
    for (k = 0; k < nk, k++) {
      res += k;
    }
  }
}
```

In Scheme, I would express this as

```
(let iter ((res 0) (i 0) (j 0) (k 0))
  (cond
   ((< k nk) (iter (+ res k) i j (1+ k)))
   ((< j nj) (iter (+ res (* 10 j) i (1+ j) 0))
   ((< i ni) (iter (+ res (* 100 i) (1+ i) 0 0))
   (else res)))
```

1.4 The Free Documentation License

The Free Documentation License is included in the Guile Reference Manual. It is included with the NYACC source as COPYING.DOC.