

Exercise 4.16.

In this exercise we implement the method just described for interpreting internal definitions. We assume that the evaluator supports `let` (see exercise 4.6).

- Change `lookup-variable-value` (section 4.1.3) to signal an error if the value it finds is the symbol `*unassigned*`.
- Write a procedure `scan-out-defines` that takes a procedure body and returns an equivalent one that has no internal definitions, by making the transformation described above.
- Install `scan-out-defines` in the interpreter, either in `make-procedure` or in `procedure-body` (see section 4.1.3). Which place is better? Why?


Answer.

- To fix `lookup-variable-value` for this purpose, we simply add a predicate to test the corresponding value when the desired variable is reached.

```
(define (lookup-variable-value var env)
  (define (env-loop env)
    (define (scan vars vals)
      (cond ((null? vars)
              (env-loop (enclosing-environment env)))
            ((eq? var (car vars))
              (if (eq? (car vals) '*unassigned*)
                  (error "Variable not yet assigned -- LOOKUP-VARIABLE-VALUE" var)
                  (car vals)))
            (else (scan (cdr vars) (cdr vals)))))
    (if (eq? env the-empty-environment)
        (error "Unbound variable" var)
        (let ((frame (first-frame env)))
          (scan (frame-variables frame)
                (frame-values frame)))))
    (env-loop env))
```

- Observe that after transformation, the procedure body consists of 4 components: a tag `'let`, a list of initializations, a list of assignments and a list of applications. Since the procedure body is a list of expressions, we can successively filter it to obtain the lists of variables, values and applications. We then assemble the lists of initializations and assignments by exploiting two internal procedures `list-of-inits` and `list-of-assignments`.

```
(define (scan-out-defines body)
  (let ((definitions (filter definition? body))
        (applications (filter (not definition?) body)))
    (let ((vars (filter definition-variable definitions))
          (vals (filter definition-value definitions)))
      (define (list-of-inits vars)
        (if (null? vars)
            '()
            (cons (list (car vars) '*unassigned*)
                  (list-of-inits (cdr vars)))))
      (define (list-of-assignments vars vals)
        (if (null? vars)
            '()
            (cons (list 'set! (car vars) (car vals))
                  (list-of-assignments (cdr vars) (cdr vals)))))
      (cons 'let
            (cons (list-of-inits vars)
                  (list-of-assignments vars vals)
                  applications)))))
```

*. Creative Commons  2014, Lawrence X. Amlord (颜世敏, aka 颜序).
Email address: informlarry@gmail.com

```
(append (list-of-assignment vars vals)
         applications))))))
```

c. Personally, I believe that installing `scan-out-defines` in the `make-procedure`—the constructor for procedures is a better choice, for only in this way the transformation is done once. Otherwise, the interpreter will have to perform this transformation whenever a compound procedure is applied, which will undoubtedly reduce the efficiency of our metacircular evaluator.