

#### Exercise 4.18.

Consider an alternative strategy for scanning out definitions that translates the example in the text to

```
(lambda <vars>
  (let ((u '*unassigned*)
        (v '*unassigned*))
    (let ((a <e1>)
          (b <e2>))
      (set! u a)
      (set! v b))
    <e3>))
```

Here `a` and `b` are meant to represent new variable names, created by the interpreter, that do not appear in the user's program. Consider the `solve` procedure from section 3.5.4:

```
(define (solve f y0 dt)
  (define y (integral (delay dy) y0 dt))
  (define dy (stream-map f y))
  y)
```


Will this procedure work if internal definitions are scanned out as shown in this exercise? What if they are scanned out as shown in the text? Explain.

#### Answer.

The `solve` procedure won't work if internal definitions are scanned out as shown in this exercise, for it enforces the restriction that the defined variables' value can be evaluated without using any of the variables' values. But the evaluation of `dy` of the inner `let` in `solve` after transformation requires the value of another defined variable `y`, this violates the rule.

However, if the internal definitions are scanned out as shown in the text, the `solve` procedure will again work. Because, by the strategy for scanning out internal definitions in the text, evaluation of `y` and `dy` only occurs in the body of the `solve` procedure been transformed.

---

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