SICP sections 3.5.4 - 3.5.5

November 23, 2007 at 16:06 Tags SICP

The code for this section is in Common Lisp.

Here's the code for `integral` and `solve`. Note the usage of `labels` to translate Scheme's recursive variable definitions to CL:

```
(defun integral (delayed-integrand initial-value dt)
  (labels (
      (int ()
        (cons-stream
          initial-value
          (let ((integrand (force delayed-integrand)))
            (add-streams
              (scale-stream integrand dt) (int))))))
    (int)))

(defun solve (f y0 dt)
  (labels (
      (y () (integral (delay (dy)) y0 dt))
      (dy () (stream-map f (y))))

    (y)))
```

Exercise 3.77

```
(defun integral (delayed-integrand initial-value dt)
  (cons-stream
    initial-value
    (let ((integrand (force delayed-integrand)))
      (if (stream-null? integrand)
        the-empty-stream
        (integral
          (delay (stream-cdr integrand))
          (+  (* dt (stream-car integrand))
              initial-value)
          dt)))))
```

Exercise 3.78

```
(defun solve-2nd (a b y0 dy0 dt)
  (labels (
      (y () (integral (delay (dy)) y0 dt))
      (dy () (integral (delay (ddy)) dy0 dt))
      (ddy () (add-streams
                (scale-stream (dy) a)
                (scale-stream (y) b))))
    (y)))
```

Exercise 3.79

```
(defun solve-2nd (f y0 dy0 dt)
  (labels (
      (y () (integral (delay (dy)) y0 dt))
      (dy () (integral (delay (ddy)) dy0 dt))
      (ddy () (stream-map f (dy) (y))))
    (y)))
```

Exercise 3.80

```
(defun RLC (R L C dt)
  (labels (
      (rlc-model (vc0 il0)
        (labels (
            (il ()
              (integral (delay (dil)) il0 dt))
            (vc ()
              (integral (delay (dvc)) vc0 dt))
            (dil ()
              (add-streams
                (scale-stream (vc) (/ 1 L))
                (scale-stream (il) (- (/ R L)))))
            (dvc ()
              (scale-stream (il) (/ -1 C))))
          (cons (vc) (il)))))
    #'rlc-model))
```

For comments, please send me an email.

Back to top