Structure and Interpretation of Computer Programs
Second Edition
Sample Problem Set

Streams And Series

Streams

Part 1: Tutorial exercises

Prepare the following exercises for oral presentation in tutorial:

Tutorial exercise 1: Describe the streams produced by the following definitions. Assume that `integers` is the stream of non-negative integers (starting from 1):

```
(define A (cons-stream 1 (scale-stream 2 A)))

(define (mul-streams a b)
  (cons-stream
   (* (stream-car a) (stream-car b))
   (mul-streams (stream-cdr a)
                (stream-cdr b))))

(define B (cons-stream 1 (mul-stream B integers)))
```

Tutorial exercise 2 Given a stream `s` the following procedure returns the stream of all pairs of elements from `s`:

```
(define (stream-pairs s)
  (if (stream-null? s)
      the-empty-stream
      (stream-append
       (stream-map
        (lambda (sn) (list (stream-car s) sn))
        (stream-cdr s))
       (stream-pairs (stream-cdr s)))))

(define (stream-append s1 s2)
  (if (stream-null? s1)
      s2
      (cons-stream (stream-car s1)
                   (stream-append (stream-cdr s1) s2))))
```

(a) Suppose that `integers` is the (finite) stream 1, 2, 3, 4, 5. What is `(stream-pairs s)`? (b) Give the clearest explanation that you can of how `stream-pairs` works. (c) Suppose that `s` is the stream of positive integers. What are the first few elements of `(stream-pairs s)`? Can you suggest a modification of `stream-pairs` that would be more appropriate in dealing with infinite streams?

Part 2: Laboratory—Using streams to represent power series

We described in lecture a few weeks ago how to represent polynomials as lists of terms. In a similar way, we can work with *power series*, such as

$$\begin{aligned}
e^x &= 1 + x + \frac{x^2}{2} + \frac{x^3}{3 \cdot 2} + \frac{x^4}{4 \cdot 3 \cdot 2} + \cdots \\
\cos x &= 1 - \frac{x^2}{2} + \frac{x^4}{4 \cdot 3 \cdot 2} - \cdots \\
\sin x &= x - \frac{x^3}{3 \cdot 2} + \frac{x^5}{5 \cdot 4 \cdot 3 \cdot 2} - \cdots
\end{aligned}$$

represented as streams of infinitely many terms. That is, the power series

$$a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots$$

will be represented as the infinite stream whose elements are $a_0, a_1, a_2, a_3, \ldots$.[1]

Why would we want such a method? Well, let's separate the idea of a series representation from the idea of evaluating a function. For example, suppose we let $f(x) = \sin x$. We can separate the idea of evaluating f, e.g., $f(0) = 0, f(.1) = 0.0998334$, from the means we use to compute the value of f. This is where the series representation is used, as a way of storing information sufficient to determine values of the function. In particular, by substituting a value for x into the series, and computing more and more terms in the sum, we get better and better estimates of the value of the function for that argument. This is shown in the table, where $\sin \frac{1}{10}$ is considered.

Coefficient	x^n	term	sum	value
0	1	0	0	0
1	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	.1
0	$\frac{1}{100}$	0	$\frac{1}{10}$	.1
$-\frac{1}{6}$	$\frac{1}{1000}$	$-\frac{1}{6000}$	$\frac{599}{6000}$	.099833333333
0	$\frac{1}{10000}$	0	$\frac{599}{6000}$	.099833333333
$\frac{1}{120}$	$\frac{1}{100000}$	$\frac{1}{12000000}$	$\frac{1198001}{12000000}$	.09983341666

The first column shows the terms from the series representation for sine. This is the infinite series with which we will be dealing. The second column shows values for the associated powers of $\frac{1}{10}$. The third column is the product of the first two, and represents the next term in the series evaluation. The fourth column represents the sum of the terms to that point, and the last column is the decimal approximation to the sum.

[1] In this representation, all streams are infinite: a finite polynomial will be represented as a stream with an infinite number of trailing zeroes.

With this representation of functions as streams of coefficients, series operations such as addition and scaling (multiplying by a constant) are identical to the basic stream operations. We provide series operations, though, in order to implement a complete power series data abstraction:

```
(define (add-streams s1 s2)
  (cond ((stream-null? s1) s2)
        ((stream-null? s2) s1)
        (else
         (cons-stream (+ (stream-car s1) (stream-car s2))
                      (add-streams (stream-cdr s1)
                                   (stream-cdr s2))))))

(define (scale-stream c stream)
  (stream-map (lambda (x) (* x c)) stream))

(define add-series add-streams)

(define scale-series scale-stream)

(define (negate-series s)
  (scale-series -1 s))

(define (subtract-series s1 s2)
  (add-series s1 (negate-series s2)))
```

You can use the following procedure to examine the series you will generate in this problem set:

```
(define (show-series s nterms)
  (if (= nterms 0)
      'done
      (begin (write-line (stream-car s))
             (show-series (stream-cdr s) (- nterms 1)))))
```

You can also examine an individual coefficient (of x^n) in a series using `series-coeff`:

```
(define (series-coeff s n)
  (stream-ref s n))
```

We also provide two ways to construct series. `Coeffs->series` takes an list of initial coefficients and pads it with zeroes to produce a power series. For example, `(coeff->series '(1 3 4))` produces the power series $1 + 3x + 4x^2 + 0x^3 + 0x^4 + \ldots$.

```
(define (coeffs->series list-of-coeffs)
  (define zeros (cons-stream 0 zeros))
  (define (iter list)
    (if (null? list)
        zeros
        (cons-stream (car list)
                     (iter (cdr list)))))
  (iter list-of-coeffs))
```

`Proc->series` takes as argument a procedure p of one numeric argument and returns the series

$$p(0) + p(1)x + p(2)x^2 + p(3)x^3 + \cdots$$

The definition requires the stream `non-neg-integers` to be the stream of non-negative integers: $0, 1, 2, 3, \ldots$.

```
(define (proc->series proc)
  (stream-map proc non-neg-integers))
```

Note: Loading the code for this problem set will change Scheme's basic arithmetic operations `+`, `-`, `*`, and `/` so that they will work with rational numbers. For instance, `(/ 3 4)` will produce 3/4 rather than .75. You'll find this useful in doing the exercises below.

Lab exercise 1: Load the code for problem set 9. To get some initial practice with streams, write and turn in definitions for each of the following:

- `ones`: the infinite stream of 1's.
- `non-neg-integers`: the stream of integers, $1, 2, 3, 4, \ldots$
- `alt-ones`: the stream $1, -1, 1, -1, \ldots$
- *zeros*: the infinite stream of 0's. Do this using `alt-ones`.

Now, show how to define the series:

$$\begin{aligned} S_1 &= 1 + x + x^2 + x^3 + \cdots \\ S_2 &= 1 + 2x + 3x^2 + 4x^3 + \cdots \end{aligned}$$

Turn in your definitions and a couple of coefficient printouts to demonstrate that they work.

Lab exercise 2: Multiplying two series is a lot like multiplying two multi-digit numbers, but starting with the left-most digit, instead of the right-most.

For example:

```
    11111
 x  12321
---------

11111
 22222
  33333
   22222
    11111
---------
136898631
```

Now imagine that there can be an infinite number of digits, i.e., each of these is a (possibly infinite) series. (Remember that because each "digit" is in fact a term in the series, it can become arbitrarily large, without carrying, as in ordinary multiplication.)

Using this idea, complete the definition of the following procedure, which multiplies two series:

```
(define (mul-series s1 s2)
  (cons-stream ⟨ E1 ⟩
               (add-series ⟨ E2 ⟩
                           ⟨ E3 ⟩)))
```

To test your procedure, demonstrate that the product of S_1 (from exercise 1) and S_1 is S_2. What is the coefficient of x^{10} in the product of S_2 and S_2? Turn in your definition of `mul-series`. (Optional: Give a general formula for the coefficient of x^n in the product of S_2 and S_2.)

Inverting a power series

Let S be a power series whose constant term is 1. We'll call such a power series a "unit power series." Suppose we want to find the *inverse* of S, namely, the power series X such that $S \cdot X = 1$. To see how to do this, write $S = 1 + S_R$ where S_R is the rest of S after the constant term. Then we want to solve the equation $S \cdot X = 1$ for S and we can do this as follows:

$$\begin{aligned} S \cdot X &= 1 \\ (1 + S_R) \cdot X &= 1 \\ X + S_R \cdot X &= 1 \\ X &= 1 - S_R \cdot X \end{aligned}$$

In other words, X is the power series whose constant term is 1 and whose rest is given by the negative of S_R times X.

Lab exercise 3: Use this idea to write a procedure `invert-unit-series` that computes $1/S$ for a unit power series S. To test your procedure, invert the series S_1 (from exercise 1) and show that you get the series $1 - x$. (Convince yourself that this is the correct answer.) Turn in a listing of your procedure. This is a very short procedure, but it is very clever. In fact, to someone looking at it for the first time, it may seem that it can't work—that it must go into an infinite loop. Write a few sentences of explanation explaining why the procedure does in fact work, and does not go into a loop.

Lab exercise 4: Use your answer from exercise 3 to produce a procedure `div-series` that divides two power series. `Div-series` should work for any two series, provided that the denominator series begins with a non-zero constant term. (If the denominator has a zero constant term, then `div-series` should signal an error.) Turn in a listing of your procedure along with three or four well-chosen test cases (and demonstrate why the answers given by your division are indeed the correct answers).

Lab exercise 5: Now suppose that we want to integrate a series representation. By this, we mean that we want to perform symbolic integration, thus, for example, given a series

$$a_0 + a_1x + a_2x^2 + a_3x^3 + \cdots$$

we want to return the integral of the series (except for the constant term)

$$a_0x + \frac{1}{2}a_1x^2 + \frac{1}{3}a_2x^3 + \frac{1}{4}a_3x^4 + \cdots$$

Define a procedure `integrate-series-tail` that will do this. Note that all you need to do is transform the series

$$a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad \cdots$$

into the series

$$a_0 \quad \frac{a_1}{2} \quad \frac{a_2}{3} \quad \frac{a_3}{4} \quad \frac{a_4}{5} \quad \frac{a_5}{6} \quad \cdots$$

Note that this means that the procedure generates the coefficients of a series starting with the first order coefficient, not that the zeroth order coefficient is 0.

Turn in a listing of your procedure and demonstrate that it works by computing `integrate-series-tail` of the series S_2 from exercise 1.

Lab exercise 6: Demonstrate that you can generate the series for e^x as

```
(define exp-series
  (cons-stream 1 (integrate-series-tail exp-series)))
```

Explain the reasoning behind this definition. Show how to generate the series for sine and cosine, in a similar way, as a pair of mutually recursive definitions. It may help to recall that the integral

$$\int \sin x = -\cos x$$

and that the integral

$$\int \cos x = \sin x$$

Lab exercise 7: Louis Reasoner is unhappy with the idea of using `integrate-series-tail` separately. "After all," he says, "if we know what the constant term of the integral is supposed to be, we should just be able to incorporate that into a procedure." Louis consequently writes the following procedure, using `integrate-series-tail`:

```
(define (integrate-series series constant-term)
  (cons-stream constant-term (integrate-series-tail series)))
```

He would prefer to define the exponential series as

```
(define exp-series
  (integrate-series exp-series 1))
```

Write a two or three sentence clear explanation of why this won't work, while the definition in exercise 6 does work.

Lab exercise 8: Write a procedure that produces the derivative of a power series. Turn in a definition of your procedure and some examples demonstrating that it works.

Lab exercise 9: Generate the power series for tangent, and secant. List the first ten or so coefficients of each series. Demonstrate that the derivative of the tangent is the square of the secant.

Lab exercise 10: We can also generate power series for inverse trigonometric functions. For example:

$$\tan^{-1}(x) = \int_0^x \frac{dz}{1+z^2}$$

Use this equation, plus methods that you have already created, to generate a power series for arctan. Note that $1+z^2$ can be viewed as a finite series. Turn in your definition, and a printout of the first few coefficients of the series.

Lab exercise 11: One very useful feature of a power series representation for a function is that one can use the initial terms in the series to get approximations to the function. For example, suppose we have

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots$$

We have represented this by the series of coefficients:

$$\{a_0 \quad a_1 \quad a_2 \quad a_3 \quad \ldots\}$$

Now suppose that we want to approximate the value of the function f at some point x_0. We could successively improve this approximation[2] by considering the following

$$f(x_0) \approx a_0 \tag{1}$$
$$f(x_0) \approx a_0 + a_1 x_0 \tag{2}$$
$$f(x_0) \approx a_0 + a_1 x_0 + a_2 x_0^2 \tag{3}$$
$$\tag{4}$$

Notice that each of these expressions (1), (2), (3) could also be captured in a stream representation, with first term a_0, second term $a_0 + a_1 x_0$ and so on.

Implement this idea by defining a procedure `approximate` which takes as arguments a value x_0 and a series representation of a function f, and which returns a stream of successive approximations to the value of the function at that point $f(x_0)$. Turn in a listing of your code, as well as examples

[2]actually there are some technical issues about whether the argument is in the radius of convergence of the series, but we'll ignore that issue here

of using it to approximate some functions. Note that to be very careful, this is not a series representation but a stream one, so you may want to think carefully about which representations to use.

Optional additional exercises

he *Bernoulli numbers* B_n are defined by the coefficients in the following power series:[3]

$$\frac{x}{e^x - 1} = B_0 + B_1 x + \frac{B_2 x^2}{2!} + \frac{B_3 x^3}{3!} + \cdots = \sum_{k \geq 0} \frac{B_k x^k}{k!}$$

Write a procedure that takes n as an argument and produces the nth Bernoulli number. (Check: All the odd Bernoulli numbers are zero except for $B_1 = -1/2$.) Generate a table of the first 10 (non-zero) Bernoulli numbers. Note: Since $e^x - 1$ has a zero constant term, you can't just use `div-series` to divide x by this directly.

It turns out that the coefficient of x^{2n-1} in the series expansion of $\tan x$ is given by

$$\frac{(-1)^{n-1} 2^{2n} (2^{2n} - 1) B_{2n}}{(2n)!}$$

Use your series expansion of tangent to verify this for a few of values of n.

[3] The Bernoulli numbers arise in a wide variety of applications involving power series and approximations. They were introduced by Jacob Bernoulli in 1713.