Announcements
Efficient Sequence Processing
Sequence Operations

Map, filter, and reduce express sequence manipulation using compact expressions

Example: Sum all primes in an interval from \( a \) (inclusive) to \( b \) (exclusive)

```python
def sum_primes(a, b):
    total = 0
    x = a
    while x < b:
        if is_prime(x):
            total = total + x
        x = x + 1
    return total
```

```
def sum_primes(a, b):
    return sum(filter(is_prime, range(a, b)))
```

(sum) \( \Theta(1) \)

(filter) \( \Theta(1) \)

(range iterator) next: 1 end: 6

Space: \( \Theta(1) \)
Streams
Streams are Lazy Scheme Lists

A stream is a list, but the rest of the list is computed only when needed:

- \(\text{car } (\text{cons } 1 2)\) \(\rightarrow\) 1
- \(\text{car } (\text{cons-stream } 1 2)\) \(\rightarrow\) 1
- \(\text{cdr } (\text{cons } 1 2)\) \(\rightarrow\) 2
- \(\text{cdr-stream } (\text{cons-stream } 1 2)\) \(\rightarrow\) 2
- \(\text{cons } 1 (\text{cons } 2 \text{ nil})\)
- \(\text{cons-stream } 1 (\text{cons-stream } 2 \text{ nil})\)

Errors only occur when expressions are evaluated:

- \(\text{cons } 1 (/ 1 0)\) \(\rightarrow\) ERROR
- \(\text{cons-stream } 1 (/ 1 0)\) \(\rightarrow\) (1 . #[delayed])
- \(\text{car } (\text{cons } 1 (/ 1 0))\) \(\rightarrow\) ERROR
- \(\text{car } (\text{cons-stream } 1 (/ 1 0))\) \(\rightarrow\) 1
- \(\text{cdr } (\text{cons } 1 (/ 1 0))\) \(\rightarrow\) ERROR
- \(\text{cdr-stream } (\text{cons-stream } 1 (/ 1 0))\) \(\rightarrow\) ERROR
- (Demo)
Stream Ranges are Implicit

A stream can give on-demand access to each element in order

(define (range-stream a b)
  (if (>= a b)
      nil
      (cons-stream a (range-stream (+ a 1) b)))))

(define lots (range-stream 1 10000000000000000000))

(scm> (car lots)
 1
(scm> (car (cdr-stream lots))
 2
(scm> (car (cdr-stream (cdr-stream lots)))
 3
Infinite Streams
Integer Stream

An integer stream is a stream of consecutive integers

The rest of the stream is not yet computed when the stream is created

```
(define (int-stream start)
  (cons-stream start (int-stream (+ start 1))))
```

(Demo)
Stream Processing

(Demo)
Recursively Defined Streams

The rest of a constant stream is the constant stream

\[
(def \ ones \ (\text{cons-stream} \ 1 \ \text{ones}))
\]

Combine two streams by separating each into car and cdr

\[
(def \ (\text{add-streams} \ s \ t) \\
\quad \ (\text{cons-stream} \ (+ \ (\text{car} \ s) \ (\text{car} \ t)) \\
\quad \quad \quad \ (\text{add-streams} \ (\text{cdr-stream} \ s) \\
\quad \quad \quad \quad \ (\text{cdr-stream} \ t))))
\]

\[
(def \ \text{ints} \ (\text{cons-stream} \ 1 \ (\text{add-streams} \ \text{ones} \ \text{ints})))
\]
Example: Repeats

\[
\text{(define } a \text{ (cons-stream 1 (cons-stream 2 (cons-stream 3 a)))))}
\]

\[
\text{(define } f \text{ s) (cons-stream (car s)}
\text{ (cons-stream (car s)}
\text{ (f (cdr-stream s))))})
\]

\[
\text{(define } g \text{ s) (cons-stream (car s)}
\text{ (f (g (cdr-stream s))))})
\]

What's (prefix a 8)? \[() 1 \ 2 \ 3 \ 1 \ 2 \ 3 \ 1 \ 2 \]

What's (prefix (f a) 8)? \[() 1 \ 1 \ 2 \ 2 \ 3 \ 3 \ 1 \ 1 \]

What's (prefix (g a) 8)? \[() 1 \ 2 \ 2 \ 3 \ 3 \ 3 \ 3 \ 1 \]
Higher-Order Stream Functions
Higher-Order Functions on Streams

Implementations are identical, but change cons to cons-stream and change cdr to cdr-stream

\[
\begin{align*}
\text{(define (map-stream f s)} & \text{)} \\
& \text{(if (null? s)} \\
& \text{ nil} \\
& \text{(cons-stream (f (car s))} \\
& \text{ (map-stream f} \\
& \text{ (cdr-stream s)))))) \\
\end{align*}
\]

\[
\begin{align*}
\text{(define (filter-stream f s)} & \text{)} \\
& \text{(if (null? s)} \\
& \text{ nil} \\
& \text{(if (f (car s))} \\
& \text{ (cons-stream (car s)} \\
& \text{ (filter-stream (cdr-stream s)))} \\
& \text{ (filter-stream (cdr-stream s))))) \\
\end{align*}
\]

\[
\begin{align*}
\text{(define (reduce-stream f start)} & \text{)} \\
& \text{(if (null? s)} \\
& \text{ start} \\
& \text{(reduce-stream f} \\
& \text{ (cdr-stream s)} \\
& \text{ (f start (car s))))) \\
\end{align*}
\]
A Stream of Primes

For any prime $k$, any larger prime must not be divisible by $k$.

The stream of integers not divisible by any $k \leq n$ is:

- The stream of integers not divisible by any $k < n$
- Filtered to remove any element divisible by $n$

This recurrence is called the Sieve of Eratosthenes

$2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13$