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
```

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

Streams are Lazy Scheme Lists

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

```scheme
(scm> (car lots))

(scm> (car (cdr-stream lots)))
```

Errors only occur when expressions are evaluated:

```scheme
(scm> (cons 1 (/ 1 0))) -> ERROR
```

Stream Ranges are Implicit

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

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

```
(define lots (range-stream 1 10000000000000000000))
```

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

\[
\text{(define (int-stream start)} \left(\text{cons-stream start (int-stream (+ start 1))}\right))
\]

Stream Processing

Recursively Defined Streams

The rest of a constant stream is the constant stream

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

Combine two streams by separating each into car and cdr

\[
\text{(define (add-streams s t)}} \left(\text{cons-stream (+ (car s) (car t)) (add-streams (cdr-stream s) (cdr-stream t))}\right))
\]

\[
\text{(define ints (cons-stream 1 (add-streams ones ints)))}
\]

Higher-Order Stream Functions

Higher-Order Functions on Streams

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

\[
\text{(define (map-stream f s)}} \left(\text{if (null? s) nil (cons-stream (f (car s)) (map-stream f (cdr-stream s)))}\right))
\]

\[
\text{(define (filter-stream f s)}} \left(\text{if (null? s) nil (if (f (car s)) (cons-stream (car s) (filter-stream f (cdr-stream s))) (filter-stream f (cdr-stream s)))}\right))
\]

\[
\text{(define (reduce-stream f) start)} \left(\text{if (null? s) start (reduce-stream f (cdr-stream s) (f start (car s)))}\right))
\]

A Stream of Primes

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

\[
2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
\]