Stream & SQL

SICP22 TAs

Stream

- A queue to be processed one by one.
- Evaluation strategy: eager & lazy
    - The latter allows infinite list.
    - List & generator in python.
- “Stream calculus”:
    - Eagerly calculate the first element, lazily calculated the rest.

SQL

- Just declare what you want.
    - Which means, clarify what you want before coding!

Lab10p1: fix the bug

We don't want to filter all the stream at once. We do filtering one by one, leaving the rest of stream untouched. LAZY EVAL.

```
(define (filter-stream f s)
  (if (null? s) nil
      (let ((rest (filter-stream f (cdr-stream s))))
        (if (f (car s))
            (cons-stream (car s) rest)
            rest))))
```

Lab10p1: fix the bug

```
(define (filter-stream f s)
   (if (null? s) nil
        (if (f (car s))
            (cons-stream (car s) (filter-stream f (cdr-
stream s)))
            (filter-stream f (cdr-stream s)))))
```

We found one element!

Leaving the rest of stream untouched, with cons-stream special form.

Lab10p2: slice

```
Not difficult, see example answer.
```

Lab10p3: combine-stream

```
Not difficult, see example answer.
```

Lab10p3.1: factorial

```
(1 2 3 4 5 …)
  (1 2 3 4 …)
    (1 2 3 …)
      (1 2 …)
(1 2 6 24 …)
```

```
Combine a new `positive` stream (`naturals 1`) with `*` after calculating
every one element of the stream.
```

```
See example answer.
```

Lab10p3.2: fib

```
(0 1 1 2 3 5 8 …)
    (0 1 1 2 3 5 8 …)
  (~~0~~ 1 1 2 3 5 8 …)

fib(i) = fib(i-1) + fib(i-2)
```

Recursively defined stream.

See example answer.

Lab10p3.3: exp

```
Let i be the ith element in stream.
exp(x)[0] = 1
exp(x)[i] = exp(x)[i-1] + x^i/fact(i)

Expose i with `combine-with naturals` for x^i.
Get fact(i) with `combine-with factoricals`, the stream has implicit i
already.

See example answer.
```

Lab10p4: non-decreasing

```
Just elaborate on “how to calculate the first element”.

See example answer.
```

Lab10p5: my-stream

Use “thunk” for lazy evaluation.

- Pack the calculation into a lambda.
- For example, pack expression e with lambda() e.

See example answer.

Lab10p5: primes

After we sieve out an element, we enlarge the sieve by stacking another filter function for the rest of the stream.

See example answer.

Hw10&lab11

- See example answer ☺