61A Lecture 38

Friday, December 6

Announcements

- Homework 11 due Wednesday 12/10 @ 11:59pm
  - All you have to do is vote on your favorite recursive art
- Homework 12 due Wednesday 12/10 @ 11:59pm
  - All you have to do is complete the final online survey
  - Grading and last remarks
- 30 hours of review sessions next week! Pick and choose topics.
- The final exam is on Thursday 12/18 @ 3pm emphasizing:
  - Functions and environments
  - Sequences, trees, and recursion
  - Mutation and object-oriented programming
  - Iterators, generators, and streams
  - Scheme and SQL

Implementing Recursive Tables

Reminder: a Select Class

The SQL parser creates an instance of the `Select` class for each `select` statement

```
>>> class Select:
        """select [columns] from [tables] where [condition]."""
        def __init__(self, columns, tables, condition):
            self.columns = columns
            self.tables = tables
            self.condition = condition
            self.make_row = create_make_row(self.columns)
        def execute(self, env):
            """Join, filter, and map rows from tables to columns."""
            from_rows = join(self.tables, env)
            filtered_rows = filter(self.filter_fn, from_rows)
            return map(self.make_row, filtered_rows)
        def filter_fn(self, row):
            if self.condition:
                return eval(self.condition, row)
            else:
                return True
```

Simplified version of http://composingprograms.com/examples/sql/sql_exec.py

Recursive Tables

In SQL's limited form of recursion, a table can be constructed by iterating through rows

HW 10: When dogs are stacked on top of one another, the total height of the stack is the sum of the heights of the dogs; *list dogs in increasing order of height within a stack*

```
with
  stacks(names, n, total, tallest) as (
    select name, 1, height, height from dogs union
    select names || ',' || name, n+1, total+height, height
      from stacks, dogs
      where n < 4 and tallest < height
  )
select names, total from stacks where ...;
```

(Demo)

names	n	total	tallest
abraham	1	26	26
abraham,barack	2	78	52
abraham,clinton	2	73	47
abraham,clinton,barack	3	125	52
barack	1	52	52
clinton	1	47	47
clinton,barack	1	99	52

Logic Programming

Logic Programming Languages

Logic programming languages (such as Prolog) are more powerful declarative languages

In SQL:

- All rows in a table have the same number of columns, restricted to primitive values
- **No mutual recursion:** two or more tables cannot be defined in terms of each other
- **No tree recursion:** the table being defined can only appear once in a from clause

In Prolog:

- Output can contain structured data (sequences, trees, etc.)
- Any kind of recursion is allowed
- Scaling challenges
- Programs may not terminate, even if the output is finite

Addition

What if we could write a tree-recursive `select` statement?

Select ll sum expressions that evaluate to a number less than 3

```
with
  ints(n) as (
    select 1 union select 2 union select 3
  ),
  sums(exp, value) as (
    select n, n from ints union
    select '(' || a.exp || '+' || b.exp || ')', a.value + b.value
      from sums as a, sums as b where a.value + b.value <= 3
  )
select ...;
```

(Demo)

exp	value
1	1
2	2
3	2
(1+1)	2
(1+2)	3
(2+1)	3
((1+1)+1)	3
(1+(1+1))	3

Life

Thanks for being amazing!

Please stay for the HKN survey