Laziness

Streams and demand-driven computation

15-150, April 16, 2020
Frank Pfenning

Learning Goals

- How to represent infinite data structures
- How to encapsulate computation
- Understanding demand-driven (lazy) computation
- Examples
  - Infinite sequences (Fibonacci numbers, primes)
  - Input/output (keystrokes, mouse events, web servers)
  - Video or audio streams

Some Cautions

- Not quite the built-in I/O streams of SML
- Not quite Haskell's laziness (but: see Tuesday's lecture!)
- Today, we build and use a simple and universal stream library

Let’s Program!

Sieve of Eratosthenes

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
(2)	3	X	5	X	7	X	9	X	11	X	13	X	15	X	17
2	(3)		5	X	7		X		11	X	13		X		17
2	3		(5)		7			X	11		13		X		17
2	3		5		(7)				11		13	X			17
2	3		5		7				11		13				17
2	3		5		7				(11)		13				17

Summary

- How to represent infinite data structures
- How to encapsulate computation
  - Closures as suspended computation
- Understanding demand-driven (lazy) computation
- Examples
  - Infinite sequences (Fibonacci numbers, primes)