Complexity in Practice

EECS 214

October 30, 2015

ASL expressions

Addition:

```
(+ 4 5)                    ; 9
```

ASL expressions

Addition:

```
(+ 4 5)                          ; 9
```

Nesting:

```
(+ (* 3 5) (/ 2 3))              ; 10
```

ASL expressions

Addition:

```
(+ 4 5)                          ; 9
```

Nesting:

```
(+ (* 3 5) (/ 2 3))              ; 10
```

Parens matter! Error because 3 and 4 aren't functions:

```
(+ (3) (4))                      ; ERROR!
```

ASL expressions

Addition:

```
(+ 4 5)                          ; 9
```

Nesting:

```
(+ (* 3 5) (/ 2 3))              ; 10
```

Parens matter! Error because 3 and 4 aren't functions:

```
(+ (3) (4))                      ; ERROR!
```

Variables!

```
(define n (+ 2 3))
(* n n)                          ; 25
```

Functions!

```
; f->c : Number -> Number
; converts a temperature from °F to °C
(define (f->c degrees-f)
  (* 5/9 (- degrees-f 32)))
```

Functions (and tests)!

```
; f->c : Number -> Number
; converts a temperature from °F to °C
(define (f->c degrees-f)
  (* 5/9 (- degrees-f 32)))

(check-expect (f->c 212)    100)
(check-expect (f->c 68)      20)
(check-expect (f->c -40)    -40)
```

ASL lists in C++

```
struct cons {
    cons(Value f, cons* r) : first(f), rest(r)
    { }

    const Value first;
    const cons* const rest;
};

bool cons?(const cons* lst)
{ return lst != nullptr; }

bool empty?(const cons* lst)
{ return lst == nullptr; }
```

ASL lists in ASL

The empty list:

```
' ( )
```

ASL lists in ASL

The empty list:

```
'()
```

A one element list:

```
(cons 3 '())
```

ASL lists in ASL

The empty list:

```
'()
```

A one element list:

```
(cons 3 '())
```

A three element list:

```
(cons 1 (cons 2 (cons 3 '())))
```

ASL lists in ASL

The empty list:

```
'()
```

A one element list:

```
(cons 3 '())
```

A three element list:

```
(cons 1 (cons 2 (cons 3 '())))
```

Sugar for the same:

```
'(1 2 3)
```

ASL lists in ASL

The empty list:

```
'()
```

A one element list:

```
(cons 3 '())
```

A three element list:

```
(cons 1 (cons 2 (cons 3 '())))
```

Sugar for the same:

```
'(1 2 3)
```

ASL lists operations

```
(check-expect (first '(1 2 3))    1)

(check-expect (rest '(1 2 3))     '(2 3))

(check-expect (cons? '(1 2 3))    #true)
(check-expect (cons? '())         #false)

(check-expect (empty? '(1 2 3))   #false)
(check-expect (empty? '())        #true)
```

List length in C++

```
size_t length(cons* lst)
{
    size_t result = 0;

    while (lst != nullptr) {
      ++result;
      lst = lst->rest;
    }

    return result;
}
```

Recursive list length in C++

```
size_t length(cons* lst)
{
    if (lst == nullptr)
        return 0;
    else
        return 1 + length(lst->rest);
}
```

List length in ASL

```
; len : [List-of X] -> Nat
(define (len lst)
  (cond
    [(empty? lst) 0]
    [(cons? lst)  (+ 1 (len (rest lst)))]))

(check-expect (len '())          0)
(check-expect (len '(5))         1)
(check-expect (len '(5 5))       2)
(check-expect (len '(5 5 5 5))   4)
```

– To DrRacket! –

Big-O equalities

- $\mathcal{O}(f(n) + c) = \mathcal{O}(f(n))$

Big-O equalities

- $\mathcal{O}(f(n) + c) = \mathcal{O}(f(n))$

  IOW: $f(n) + c \in \mathcal{O}(f(n))$

Big-O equalities

- $\mathcal{O}(f(n) + c) = \mathcal{O}(f(n))$

  IOW: $f(n) + c \in \mathcal{O}(f(n))$
- $\mathcal{O}(cf(n)) = \mathcal{O}(f(n))$

Big-O equalities

- $\mathcal{O}(f(n) + c) = \mathcal{O}(f(n))$

  IOW: $f(n) + c \in \mathcal{O}(f(n))$
- $\mathcal{O}(cf(n)) = \mathcal{O}(f(n))$
- $\mathcal{O}(\log_k f(n)) = \mathcal{O}(\log_j f(n))$

Big-O equalities

- $\mathcal{O}(f(n) + c) = \mathcal{O}(f(n))$

  IOW: $f(n) + c \in \mathcal{O}(f(n))$
- $\mathcal{O}(cf(n)) = \mathcal{O}(f(n))$
- $\mathcal{O}(\log_k f(n)) = \mathcal{O}(\log_j f(n))$
- $\mathcal{O}(f(n) + g(n)) = \mathcal{O}(f(n))$ when $g \lll f$

Big-O inequalities

If $j < k$ then:

$1 \lll \log n$	constants are less than logs…
$\lll n^j$	are less than polynomials…
$\lll n^k$	are less than polynomials of higher degree…
$\lll n^k \log n$	are less than polynomial-log…
$\lll j^n$	are less than exponentials…
$\lll j^k$	are less than exponentials with greater exponents.