A Short C and OCaml Rant

COS 326

Speaker: Andrew Appel

Princeton University

Last Time: Java Pair Rant

Java has a paucity of types

- There is no type to describe just the pairs
- There is no type to describe just the triples
- There is no type to describe the pairs of pairs
- There is no type …

OCaml has many more types

- use option when things may be null
- do not use option when things are not null
- OCaml types describe data structures more precisely
  - programmers have fewer cases to worry about
  - entire classes of errors just go away
  - type checking and pattern analysis help prevent programmers from ever forgetting about a case

Summary of Java Pair Rant

Java has a paucity of types

- There is no type to describe j t the pairs
- There is n type to describe t
- There is no describ
- There is no t

OCam

- use c
- d

-

- type ch d p analys help prevent programmers from ever fo g about a case

SCORE: OCAML 1, JAVA 0

C, C++ Rant

Java has a paucity of types

- but at least when you forget something,

it ***throws an exception*** instead of ***silently going off the trolley***!

If you forget to check for null pointer in a C program,

- no type-check error at compile time
- no exception at run time
- it might crash right away (that would be best), or
- it might permit a buffer-overrun (or similar) vulnerability
- so the hackers pwn you!

Summary of C, C++ rant

Java has a paucity of types

- but at least when you forg omethi

it ***throws a ption*** ins g off trolley!

If you

- no ty

SCORE:

OCAML 1, JAVA 0, C -1

- it m), or
- it simila ulnerability
- so the hac n y

MORE THOUGHTS ON LISTS

The (Single) List Programming Paradigm

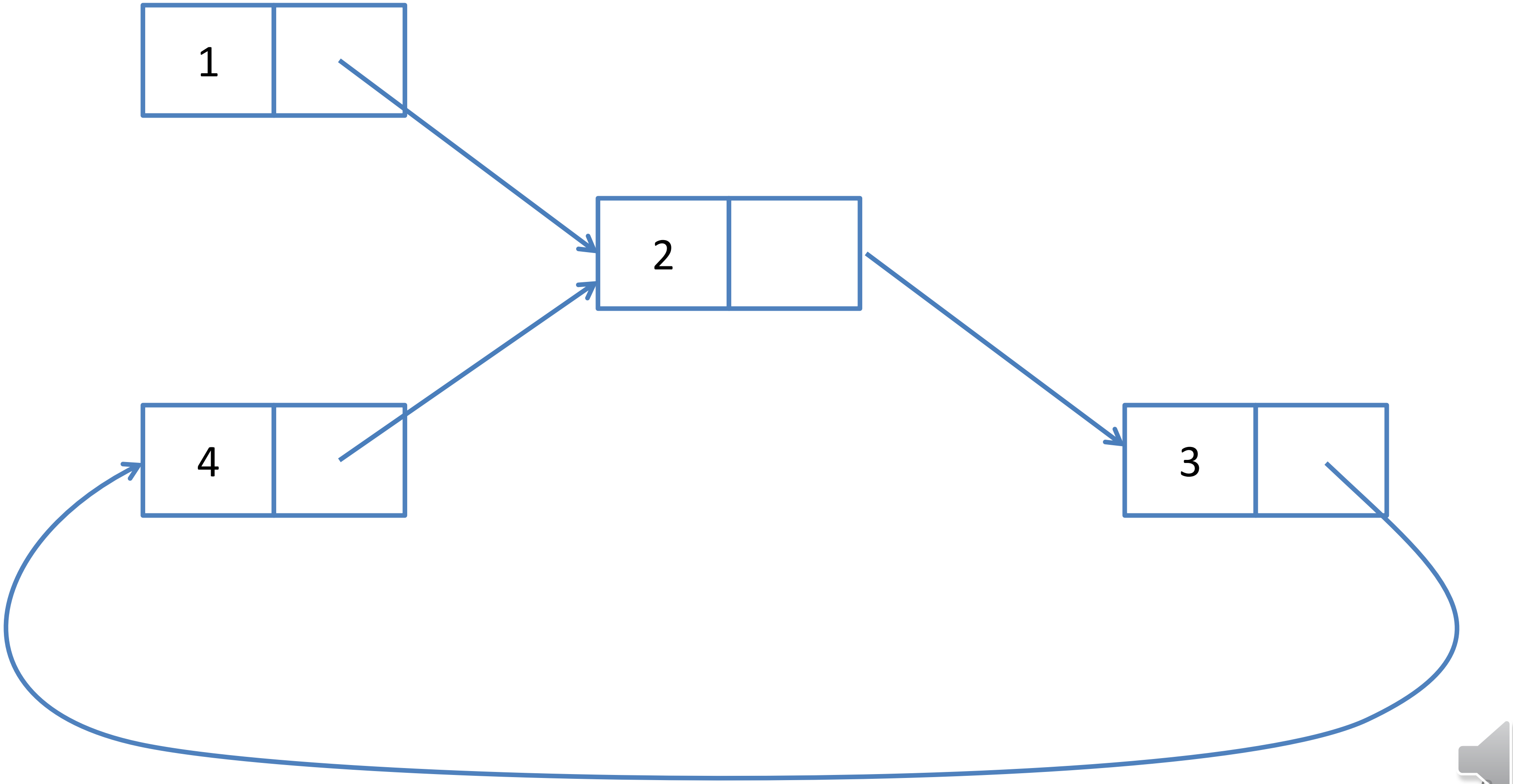
- Recall that a list is either:
  - [ ]   (the empty list)
  - v :: vs   (a value v followed by a *previously constructed list* vs)

- Some examples:

```
let l0 = [];;                     (* length is 0 *)
let l1 = 1::l0;;                  (* length is 1 *)
let l2 = 2::l1;;                  (* length is 2 *)
let l3 = 3::l2;;                  (* length is 3 *)
…
```

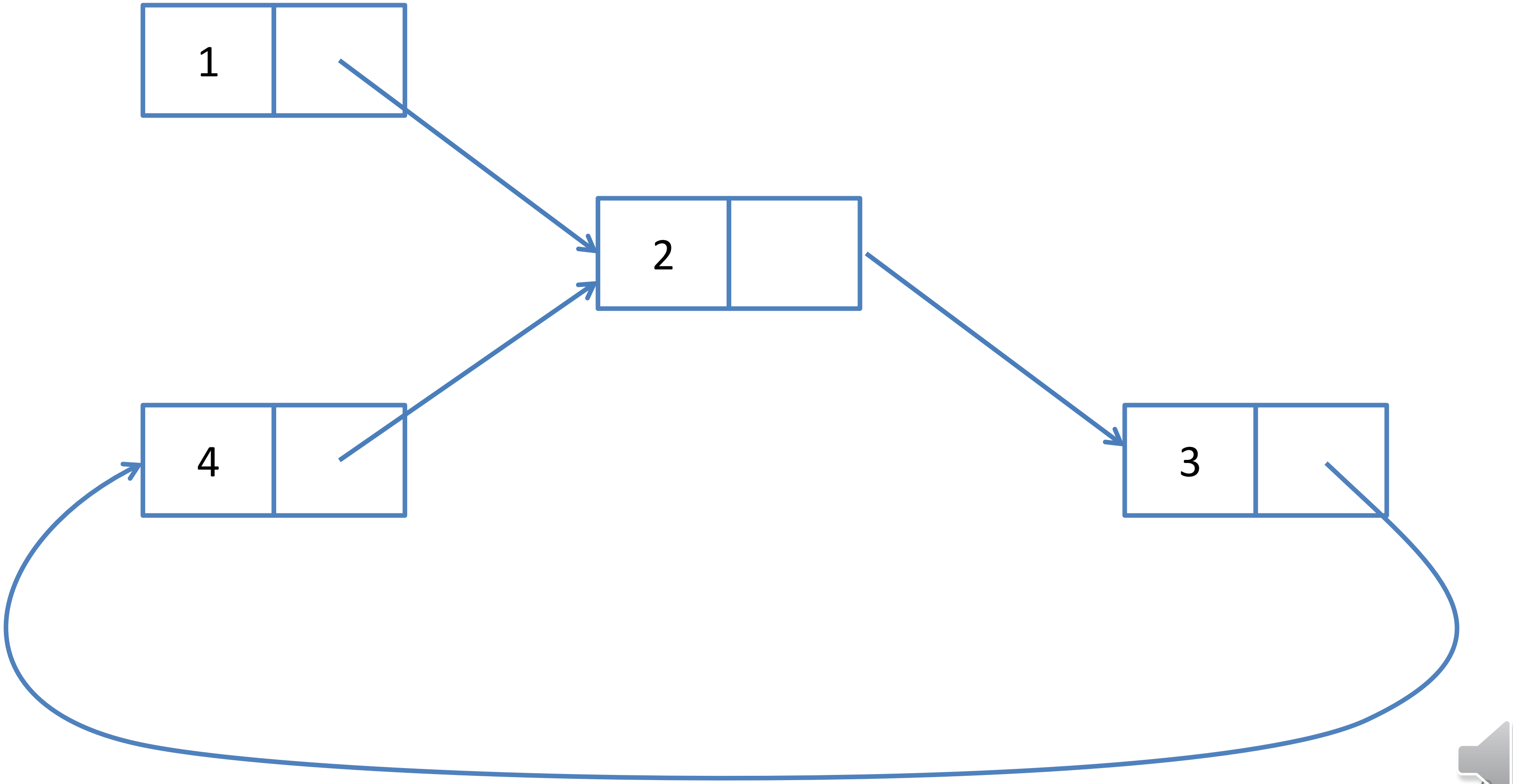
Consider This Picture

- Consider the following picture. How long is the linked structure?
- Can we build a value with type int list to represent it?



Consider This Picture

- How long is it? Infinitely long?
- Can we build a value with type int list to represent it? No!
  - all values with type int list have finite length



The List Type

- Is it a good thing that the type list does not contain any infinitely long lists? Yes!

- A terminating list-processing scheme:

```
let rec f (xs : int list) : int =
  match xs with
    [] -> … do something not recursive …
  | hd::tail -> …  f tail …
```

terminates because f only called recursively on smaller lists

A Loopy Program

```
let rec loop (xs : int list) : int =
  match xs with
    [] -> 0
  | hd::tail -> hd + loop (0::tail)
```

Does this program terminate?

A Loopy Program

```
let rec loop (xs : int list) : int =
  match xs with
    [] -> []
  | hd::tail -> hd + loop (0::tail)
```

Does this program terminate? No! Why not? We call loop recursively on (0::tail). This list is the same size as the original list -- not smaller.

Take-home Message

ML has a *strong type system*

- ML *types say a lot* about the set of values that inhabit them

In this case, the tail of the list is *always* shorter than the whole list

This makes it easy to write functions that terminate; *it would be harder if you had to consider more cases*, such as the case that the tail of a list might loop back on itself. *Moreover OCaml hits you over the head to tell you what the only 2 cases are!*

Note: Just because the list type excludes cyclic structures does not mean that an ML program can't build a cyclic data structure if it wants to. *ML is better than other languages* because it gives you *control* over the values you want to program with, via types!

Rant #2: Imperative lists

- One week from today, ask yourself: Which is easier:
  - Programming with immutable lists in ML?
  - Programming with pointers and mutable s in C/Java
  - I guarantee you are going ML
    - there a ny mor r in a
    - so many

SCORE: OCAML 2, JAVA 0

C: why bother?

Do not believe his lies.

```
let rec xs : int list = 0::xs
```

SCORE: OCAML 1.8, JAVA 0

C: why bother?