PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

CSE 341
Section 4

Higher-Order Functions and Closures
Winter 2019

Learning Objectives

- The “Value Restriction”
- Higher-Order Functions (QC, ~35 min)
  - Understand higher order functions and their expressiveness
  - Become familiar with anonymous functions
- Currying and partial application (~5 min)

Type Inference

How does type inference work?

- A good answer is outside the scope of this class.
- For weird enough cases, this is a topic of active research.

Type Inference

How does type inference work **in SML**?

- Still mostly outside the scope of this class
- We'll talk about it on Monday
- Today, we'll go over an SML-specific quirk

The Value Restriction

Let’s hop into Emacs

Key Concepts Review

- Higher-order functions
  - Pass functions around like any data
  - Closures: functions *capture* references to their environment
  - Lexical scoping: variables are captured at time of creation
- Higher-order function idioms:
  - foldl, map, etc.
- Polymorphic functions
  - Functions that are *generic* over the type of arguments

Higher-order functions

Functions are no different from any program data.

An extremely powerful feature! The “defining feature” of functional programming.*

* debatable

Higher-order functions

QuickCheck time! (~5 minutes, ungraded)

Speak with a friend if you like

Higher-order functions

What is the type of `fold`?

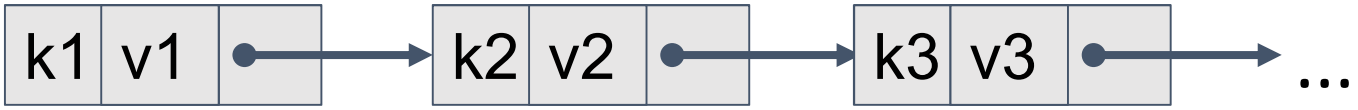
In what order does `fold` process its elements?

Is there the *one true type* for a `fold` function? Why/Why not?

Higher-Order Functions

Let’s look at an association list representation of a map and some operations (Emacs)

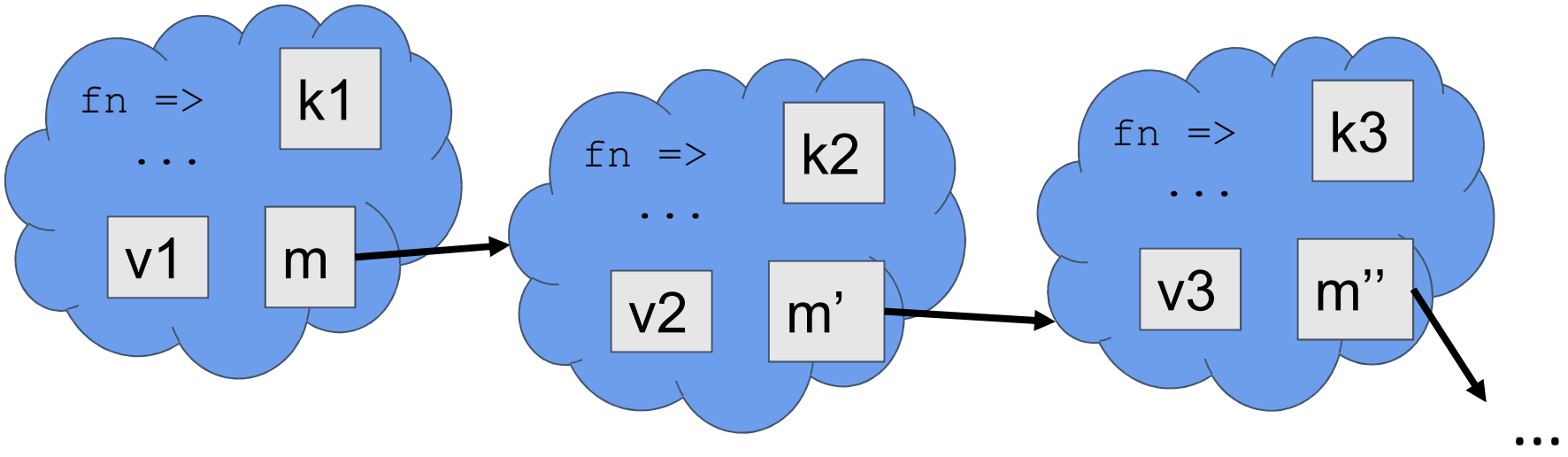
Association Lists



Closure-Based Representation

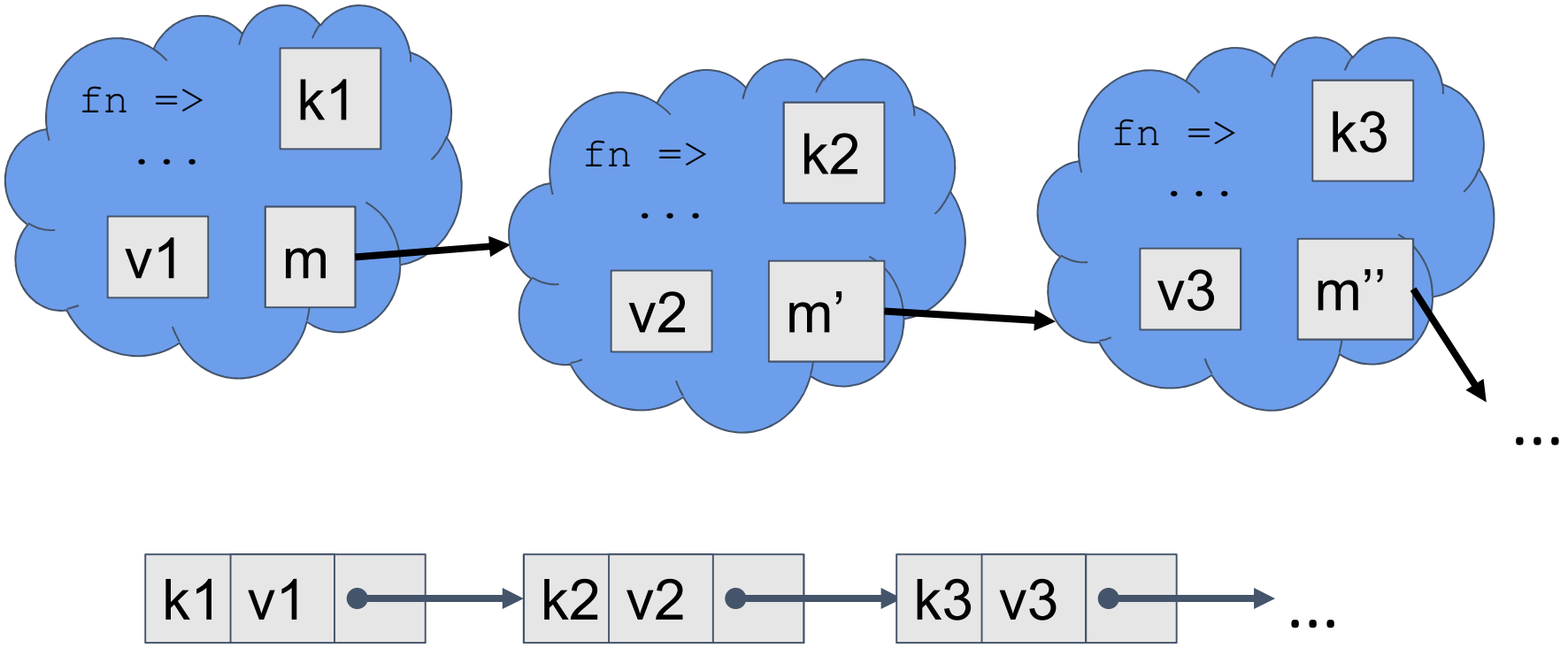
- The function (map!) returned by add captures:
  - the inserted key (k)
  - the inserted value (v)
  - the original map (m)

Closure-Based Representation



Does this look familiar?

Closure-Based Representation



Benefits of this representation

- Remove is O(1)
- Map is O(1) (kinda!)
  - Only ends up transforming values accessed later (emacs)
  - Although the result can be more expensive computationally (why?)