CS450
Structure of Higher Level Languages
Lecture 30: Dynamic dispatching
Tiago Cogumbreiro
Today we will learn...

- Dynamic dispatching
- Manual dynamic-dispatching
- Type-directed dynamic dispatching
- Type-directed dynamic dispatching with
Dynamic dispatch
(aka operator overload)

Motivation
The problem: how to unify syntax?

Three different possibilities of the same pattern

State monad  Error monad  List monad
Can we do better?

Can we avoid copy-pasting our macro?
Let us study two solutions

1. Make the macro parametric
2. Use dynamic dispatch (aka operator overload)
Option 1: parametric notation

(manual dynamic dispatch)
Option 1: parametric notation

- Add a level of indirection
- Lookup a structure that holds bind and pure
- Add notation on top of that structure
The struct Monad

Redefine macro
Example 1
Example 2
Option 2:
Type-directed dynamic dispatching
Type-directed bind

Limitations

- The types of values need to be consistent
- Idea: wrap values with structs
- Use a single function to perform dynamic dispatching

Implementation
Type-directed effectful operations

An effectful operation is a function that takes a state and returns an effect. Racket has no way of being able to identify that, so we need to wrap functions with a struct to mark them as effectful operations.
Type-directed effectful operation

Re-implementing the stack-machine operations. Notice that the do-notation calls \overline{\text{\ldots}}, which in turn calls \overline{\text{\ldots}}.
Type-directed optional result

Optional values
Limitations

1. No way to implement.
2. If we need to add a new type, we will need to change
Can we do better?

Racket = implicit+automatic dynamic dispatching
Defining a dynamic-dispatch function

1. We use to declare a function that is dispatched dynamic according to the type
   *Think declaring an abstract function.*

2. We inline each version of each type inside the structure
   *Think giving a concrete implementation of an abstract function.*