CSE 341 AB: Section 6

Josh Pollock
Office Hours: Tuesdays 3:00pm - 4:00pm
Agenda

1. Dynamic typing under the hood.

2. Memoization
   a. A review of promises
   b. Lexical scope and mutation
   c. Mutable pairs and lists
   d. Association lists
   e. Putting it all together

3. Streams with (immutable) state
How Are Dynamically-Typed Languages Implemented?

Data is in **binary** at runtime.

In a statically typed language, we have guarantees that our functions will get the right type of data.

In a dynamically typed language, we have no such guarantees. We need to keep type information around at runtime. If the types are wrong, we throw a contract violation.
<table>
<thead>
<tr>
<th>type tag</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td>500325123</td>
</tr>
<tr>
<td></td>
<td>43453254</td>
</tr>
<tr>
<td>STRING</td>
<td>500325123</td>
</tr>
<tr>
<td></td>
<td>43453254</td>
</tr>
</tbody>
</table>
To the code!
Sophisticated Streams

A good way to think about a stream is a computation that produces some output and a new computation.

Sometimes it’s convenient for this computation to remember some state.

We can use immutability to simulate state by passing immutable values around.

Example: Stream of natural numbers.

- We need to keep some state in the computation so it knows what number to output.
- Let’s keep track of the number we want to output!