Environments with functions: first try

let add = fun (x, y) -> x + y
let three = add (1, 2)
Environments with functions: first try

let add = fun (x, y) -> x + y
let three = add (1, 2)

<table>
<thead>
<tr>
<th>Var</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>fun (x, y) -&gt; x + y</td>
</tr>
</tbody>
</table>
Environments with functions: first try

let add = fun (x, y) -> x + y
let three = add (1, 2)
Environments with functions: first try

```ocaml
let add = fun (x, y) -> x + y
let three = add (1, 2)
```

<table>
<thead>
<tr>
<th>Var</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>fun (x, y) -&gt; x + y</td>
</tr>
<tr>
<td>three</td>
<td>3</td>
</tr>
</tbody>
</table>
Environments with functions: first try

let add = fun x -> fun y -> x + y
let add1 = add 1
let three = add1 2

<table>
<thead>
<tr>
<th>Var</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>fun x -&gt; fun y -&gt; x + y</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Environments with functions: first try

let add = fun x -> fun y -> x + y
let add1 = add 1
let three = add1 2
Environments with functions: first try

let add = fun x -> fun y -> x + y
let add1 = add 1
let three = add1 2
Environments with functions: first try

let \( x = 1 \) in
let \( f \ y = x + y \) in
let \( x = 2 \) in
\( f \ 2 \)
Environments with functions: first try

let \( x = 1 \) in
let \( f \ y = x + y \) in
let \( x = 2 \) in
\( f \ 2 \)

<table>
<thead>
<tr>
<th>Var</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>1</td>
</tr>
</tbody>
</table>
Environments with functions: first try

let x = 1 in
let f y = x + y in
let x = 2 in
f 2

<table>
<thead>
<tr>
<th>Var</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>f</td>
<td>fun y -&gt; x + y</td>
</tr>
</tbody>
</table>
Environments with functions: first try

let x = 1 in
let f y = x + y in
let x = 2 in
f 2

<table>
<thead>
<tr>
<th>Var</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>f</td>
<td>fun y -&gt; x + y</td>
</tr>
<tr>
<td>x</td>
<td>2</td>
</tr>
</tbody>
</table>
Environments with functions: first try

```plaintext
let x = 1 in
let f y = x + y in
let x = 2 in
f 2
```

<table>
<thead>
<tr>
<th>Var</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>f</td>
<td>fun y -&gt; x + y</td>
</tr>
<tr>
<td>x</td>
<td>2</td>
</tr>
<tr>
<td>y</td>
<td>2</td>
</tr>
</tbody>
</table>

x should still be 1 in f!
Second try: use *closures*

• Closure: function code + environment
• This will be the value of a function

• in ML:
  ```ml
  type value = ...
  | VClos of var * expr * env
  and env = (var * value option ref) list
  ```
With closures

let x = 1 in
let f y = x + y in
let x = 2 in
f 2
With closures

let x = 1 in
let f y = x + y in
let x = 2 in
f 2

<table>
<thead>
<tr>
<th>Var</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
</tr>
</tbody>
</table>
With closures

let x = 1 in
let f y = x + y in
let x = 2 in
f 2
With closures

let x = 1 in
let f y = x + y in
let x = 2 in
f 2

<table>
<thead>
<tr>
<th>Var</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>f</td>
<td>(fun y -&gt; x + y,</td>
</tr>
<tr>
<td></td>
<td>)</td>
</tr>
<tr>
<td>x</td>
<td>2</td>
</tr>
</tbody>
</table>
With closures

let x = 1 in
let f y = x + y in
let x = 2 in
f 2

Call the function with the environment from the closure (+ arguments)
Interpreting with closures

• Interpreting a function: fun x -> e
  • Return a closure with variable x, expression e, current environment

• Interpreting an application e1 e2
  • Interpret e1 to closure (x, e, env)
  • Interpret e2 to arg value v
  • Add x -> v to env, interpret e with this env