Implementing Type Checking
walk the AST

• For each syntactic structure that can have a type, add a type field to the AST

• Perform a post-order walk of the AST to assign types to each node in the AST

• Base cases:
  • Variables: get types from symbol table
  • Literals: get types from node type
walk the AST

• For each syntactic structure that can have a type, add a type field to the AST
• Perform a post-order walk of the AST to assign types to each node in the AST

• Inductive cases:
  • Expressions: compare types of component sub-expressions
  • Assignment: compare LHS and RHS

Is: $x = (a + b) + c$ well typed?

<table>
<thead>
<tr>
<th>Is: $x$ well typed?</th>
<th>Is: $a + b$ well typed?</th>
<th>Is: $c$ well typed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>int</td>
<td>int</td>
</tr>
</tbody>
</table>

Is: $a$ well typed?

<table>
<thead>
<tr>
<th>Is: $b$ well typed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
</tr>
</tbody>
</table>
walk the AST

• For each syntactic structure that can have a type, add a type field to the AST
• Perform a post-order walk of the AST to assign types to each node in the AST
• Inductive cases:
  • Expressions: compare types of component sub-expressions
  • Assignment: compare LHS and RHS
  • Conditionals: compare LHS and RHS

Is: 
\[(x < y)\]
well typed?

Is: 
x
well typed?

int

Is: 
y
well typed?

int
walk the AST

• For each syntactic structure that can have a type, add a type field to the AST
• Perform a post-order walk of the AST to assign types to each node in the AST

• Inductive cases:
  • Function calls: compare types of sub-expressions to argument types in symbol table
  • Note: type assigned to function call should be the return type of the function!
walk the AST

• For each syntactic structure that can have a type, add a type field to the AST
• Perform a post-order walk of the AST to assign types to each node in the AST

• Inductive cases:
  • Function calls: compare types of sub-expressions to argument types in symbol table
  • Return statements: compare return expression type to return type of function in symbol table

Is: return (x + y) well typed?

Is: foo well typed?

Is: x + y well typed?

Is: x well typed?

Is: y well typed?

int x int -> int

int

int

int
walk the AST

• For each syntactic structure that can have a type, add a type field to the AST
• Perform a post-order walk of the AST to assign types to each node in the AST
• If any node cannot be typed, return an error!
next: optimization