Thursday 10/10/2013: Type analysis

Lecture topics:
I Types in Languages
II Types in Tiger
III Types in Compilers

I Types in Languages

What is a type?

<table>
<thead>
<tr>
<th>Point of view</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructive</td>
<td>built-in or composite</td>
<td>type A = ( \text{int, string} )</td>
</tr>
<tr>
<td>Denotational</td>
<td>set of values</td>
<td>( A[1,2] = &quot;x&quot;, A[2,2] = &quot;y&quot; )</td>
</tr>
<tr>
<td>Abstraction</td>
<td>set of operations</td>
<td></td>
</tr>
</tbody>
</table>

What are types good for?
- finding errors
- distinguishing operations

Type constructor = language feature that allows the programmer to define a new composite type \( \neq \text{OOP constructor} \)

Type system =
- built-in types
- type constructors
- \( \text{type} = \text{entity association} \)
- type relations
- type correctness rules
- expression typing rules

Type equivalence:
Type relation defined as follows:
\( T_1 \equiv T_2 \) iff program can use value of type \( T_1 \) whose \( T_2 \) is expected and vice versa

Some Tiger types:
- type \( A = \text{array of int} \)
- type \( B = \text{array of int} \)
- type \( C = \exists p : A, q : \text{string} \)
- type \( D = \exists p : A, q : \text{string} \)
- type \( E = A \)

Structural equivalence =
\( T_1 \equiv T_2 \) iff
they are the same primitive type OR
composed from equivalent types using
same type constructor

Example: \( A \equiv B \equiv E, C \equiv D \)

Name equivalence =
\( T_1 \equiv T_2 \) if
they come from the same type
constructor application OR
they are aliases

Example: \( A \equiv B, A \equiv E, C \equiv E \)

II Types in Tiger

See tiger-spec.pdf on class webpage

Built-in types: int, string, void, nil

Type constructors: array, record

Type \( = \text{entity association} \):
- fieldDec, funDec, varDec, forExp

Type relations: name equivalence;
nil "matches" any record type

Type correctness rules:
- for example, infixExpr with \( = \) or \( <> \)
- operand types must match

Expression typing rules:
- for example, infixExpr with \( = \) or \( <> \)
- result type is int

III Types in Compilers

Diagram:

- AST
- Type analyzer
- Symbol table
- Error messages
Translation scheme

\[
\begin{align*}
S & \rightarrow B \mid D \\
B & \rightarrow \varepsilon \\
L & \rightarrow \varepsilon \\
L & \rightarrow SL \mid E \\
D & \rightarrow \text{id} := E; \\
E & \rightarrow \text{intLit} \\
E & \rightarrow \text{stringLit} \\
E & \rightarrow \text{id} \\
E & \rightarrow E_1 = E_2
\end{align*}
\]

push(B.scope)

pop(B.scope)

D.sym.type = E.type
E.type = id
E.type = string
E.sym = lookup(id)
E.type = E.sym.type
if E.type matches E.type
then E.type = int
else error

Announcements

10/17 guest lecture on bottom-up parsing
10/24 midterm exam
after 10/24 compiler back-end
revised project and homework due dates