CS 131 Compilers: Discussion 8: Scope & Type Inference and Unification

杨易为 季杨彪 尤存翰
{yangyw,jiyb,youch}@shanghaitech.edu.cn

2021 年 4 月 23 日

1 Scope

Whenever a scope is introduced, the compiler gives it a name and puts it in a structure (a tree) that makes it easy to determine the position of that scope in relation to other scopes, and it is marked as being the current scope. When a variable is declared, its assigned to the current scope.

Four kinds of scopes and their life cycle in python:

1. local (local scope), that is, the temporary variables defined in the function, when the function ends, the life cycle of the variables ends.
2. enclosed (closure, the local scope of the nested parent function), that is, the local variables of the outer function of the closure, the end of the outer function, the end of the life cycle of the variable.
3. global (global variables), that is, variables defined at the module level, the module is destroyed, the life cycle of the variable will end.
4. bulit-in (built-in function) is a variable built in python interpreter, virtual machine. The order of searching variables is: (1) -> (2) -> (3) -> (4)
   ] a = 10 def func(): a = 20 print(a) func() print(a) """ Output: 20 10 """

```
a = 10
def func():
  a = 20
    global a  # IndentationError: unindent does not match any outer indentation
  print a
func()
print a
```

2 Mid Term preparation

2.1 Regular Expressions

(a) Write a regex that matches binary strings divisible by 8.
(b) Provide aregulargrammar for the regex from part (a).

2.2 Finite State Automata

(a) Write the corresponding NFA for the regular expression (0|1)?(10|01)+;
(b) Convert the NFA from part (a) into a DFA.

2.3 Grammar Rewriting and LL(k) Parsing

$$
\begin{aligned}
S &: E \dashv \\
E &: E + E \\
&\mid E * E \\
&\mid \text{ID}
\end{aligned}
$$

(a) Show that the grammar is ambiguous with two different leftmost derivations of the string $a+b*c$;
(b) Rewrite this grammar so that it preserves the standard order of operations, is LL(1),and is unambiguous. Draw the resulting tree for the string $a + b * c$;
(c) Write down the equivalent unambiguous grammar that enforces both left associativity and correct precedence. Why can' t this be achieved with an LL(1) grammar?

2.4 Earley' s Algorithm

Consider the following grammar:

$$
\begin{aligned}
P &: E \dashv \\
E &: ID \\
&\mid \lambda ID \,.\, E \\
&\mid E\ E
\end{aligned}
$$

1. Use it to parse the following expression with Earley's algorithm: λ`ID` . `ID` `ID` $\dashv$

	λ	`ID`	.	`ID`	`ID`
	0	1	2	3	4	5
a						
b						
c						
d						
e						
f						
g						
h						
i						
j						

2. Is the expression in the language? Is the grammar ambiguous?