

Script generated by TTT

Title: Petter: Compiler Construction (25.06.2020)
- 43: String Hashing

Date: Thu Jun 25 10:13:02 CEST 2020

Duration: 13:55 min

Pages: 8

Resolving Identifiers

Observation: each identifier in the AST must be translated into a memory access

Problem: for each identifier, find out what memory needs to be accessed by providing **rapid** access to its **declaration**

37/67

Resolving Identifiers

Observation: each identifier in the AST must be translated into a memory access

Problem: for each identifier, find out what memory needs to be accessed by providing **rapid** access to its **declaration**

Ideas:

- 1 **rapid** access: replace every identifier by a **unique integer**
→ integers as keys: comparisons of integers is faster

37/67

Resolving Identifiers

Observation: each identifier in the AST must be translated into a memory access

Problem: for each identifier, find out what memory needs to be accessed by providing **rapid** access to its **declaration**

Ideas:

- 1 **rapid** access: replace every identifier by a **unique integer**
→ integers as keys: comparisons of integers is faster
- 2 link each usage of a variable to the **declaration** of that variable
→ for languages without explicit declarations, create declarations when a variable is first encountered

37/67

Rapid Access: Replace Strings with Integers

Idea for Algorithm:

Input: a sequence of strings
 Output: ① sequence of numbers
 ② table that allows to retrieve the string that corresponds to a number
 Apply this algorithm on each identifier during *scanning*.

Implementation approach:

- count the number of new-found identifiers in `int count`
- maintain a *hashtable* $S : \text{String} \rightarrow \text{int}$ to remember numbers for known identifiers

We thus define the function:

```
int indexOfIdentifier(String w) {
    if (S(w) ≡ undefined) {
        S = S ⊕ {w ↦ count};
        return count++;
    } else return S(w);
}
```

38/67

Example: Replacing Strings with Integers

Input:

Peter Piper picked a peck of pickled peppers

If Peter Piper picked a peck of pickled peppers

wheres the peck of pickled peppers Peter Piper picked

Output:

40/67

Implementation: Hashtables for Strings

- allocate an array M of sufficient size m
- choose a *hash function* $H : \text{String} \rightarrow [0, m - 1]$ with:
 - $H(w)$ is cheap to compute
 - H distributes the occurring words equally over $[0, m - 1]$

Possible generic choices for sequence types ($\vec{x} = \langle x_0, \dots, x_{r-1} \rangle$):

$$H_0(\vec{x}) = (x_0 + x_{r-1}) \% m$$

$$H_1(\vec{x}) = (\sum_{i=0}^{r-1} x_i \cdot p^i) \% m$$

$$= (x_0 + p \cdot (x_1 + p \cdot (\dots + p \cdot x_{r-1} \dots))) \% m$$

for some prime number p (e.g. 31)

- The hash value of w may not be unique!
 - Append (w, i) to a linked list located at $M[H(w)]$
 - Finding the index for w , we compare w with all x for which $H(w) = H(x)$
- access on average:
 - insert: $\mathcal{O}(1)$
 - lookup: $\mathcal{O}(1)$

39/67

Example: Replacing Strings with Integers

Input: 0 1 2 3 4 5 6 7

Peter Piper picked a peck of pickled peppers

If Peter Piper picked a peck of pickled peppers

wheres the peck of pickled peppers Peter Piper picked

Output: 1 2 3 4 5 6 7 8 9

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6			
										7	9	10	4	5	6	7	0	1	2

40/67

Example: Replacing Strings with Integers

Input:

Peter Piper picked a peck of pickled peppers

If Peter Piper picked a peck of pickled peppers

wheres the peck of pickled peppers Peter Piper picked

Output:

0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6
7 9 10 4 5 6 7 0 1 2

and

0	Peter	6	pickled
1	Piper	7	peppers
2	picked	8	If
3	a	9	wheres
4	peck	10	the
5	of		

