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

Ideas:

rapid access: replace every identifier by a unique integer

→ integers as keys: comparisons of integers is faster

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:

- 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
 - ightarrow for languages without explicit declarations, create declarations when a variable is first encountered

37/67

37/67

Rapid Access: Replace Strings with Integers

Idea for Algorithm:

Input: a sequence of strings

Output: sequence of numbers

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
- ullet maintain a *hashtable* $S: \mathbf{String} o \mathbf{int}$ to remember numbers for known identifiers

We thus define the function:

```
\begin{array}{ll} \textbf{int} & \textbf{indexForIdentifier}\{\textbf{String } w\} \\ & \textbf{if} & S(w) \equiv \textbf{undefined} \\ & S = S \oplus \{w \mapsto \texttt{count}\}; \\ & \textbf{return } \texttt{count++}; \\ & \} & \textbf{else } & \textbf{return } & S(w); \\ & \} \end{array}
```

Example: Replacing Strings with Integers

Input:

Peter Piper picked a peck of pickled peppers										
If Peter Pip		er picked		а	pec	k of	pi	ckled	peppers	
wheres	the	peck	of	pick	led	peppe	ers	Peter	Piper	picked

Output:

Implementation: Hashtables for Strings

- allocate an array M of sufficient size m
- ② choose a hash function $H: \mathbf{String} \to [0, m-1]$ with:
 - \bullet H(w) is cheap to compute
 - H distributes the occurring words equally over [0, m-1]

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

$$H_{0}(\vec{x}) = (x_{0} + x_{r-1}) \% m$$

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

$$= (x_{0} + p) \cdot (x_{1} + p) \cdot (\dots + p) \cdot (x_{r-1} \cdot \dots)) \% m$$
for some prime number p (e.g. 31)

39/67

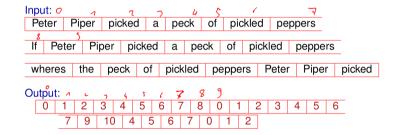
40/67

- X The hash value of w may not be unique!
- \rightarrow 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)
```

38/67

Example: Replacing Strings with Integers



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 Hashtable with m = 7 and H_0 : 0 Peter 0 If 8 the 10 pickled Piper peppers picked pickled 6 peck 4 picked 2 8 lf 3 a of 5 wheres 9 peppers 7 9 wheres 4 peck 10 the 5 Piper 1 Peter 0 a 3 5 of

40/67