Script **generated by TTT**

Title: Petter: Compiler Construction (18.06.2020) - 40: L-Attributed Grammars

Date: Thu Jun 18 13:28:08 CEST 2020

Duration: 10:11 min

Pages: 9

Demand-Driven Evaluation

Observations

- each node must contain a pointer to its parent
- *only required* attributes are evaluated
- the evaluation sequence depends – in general – on the actual syntax tree
- the algorithm must track which attributes it has already evaluated
- the algorithm may visit nodes more often than necessary

⇝ the algorithm is not local

Demand-Driven Evaluation

Observations

- each node must contain a pointer to its parent
- *only required* attributes are evaluated
- the evaluation sequence depends – in general – on the actual syntax tree
- the algorithm must track which attributes it has already evaluated
- the algorithm may visit nodes more often than necessary

⇝ the algorithm is not local

in principle:

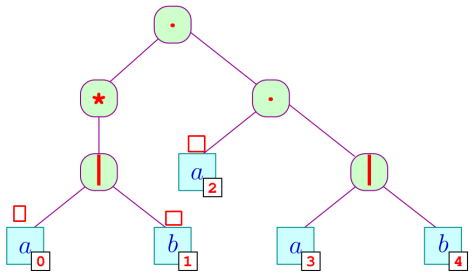
- evaluation strategy is dynamic: difficult to debug
- usually all attributes in all nodes are required

⇝ computation of all attributes is often cheaper

⇝ perform evaluation in *passes*

Implementing State

Problem: In many cases some sort of state is required.
Example: numbering the leafs of a syntax tree



Example: Implementing Numbering of Leafs

Idea:

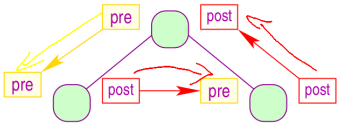
- use helper attributes pre and post
- in pre we pass the value for the first leaf down (inherited attribute)
- in post we pass the value of the last leaf up (synthesized attribute)

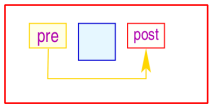
root: $\text{pre}[0] := 0$, $\text{pre}[1] := \text{pre}[0]$, $\text{post}[0] := \text{post}[1]$

node: $\text{pre}[1] := \text{pre}[0]$, $\text{pre}[2] := \text{post}[1]$, $\text{post}[0] := \text{post}[2]$

leaf: $\text{post}[0] := \text{pre}[0] + 1$

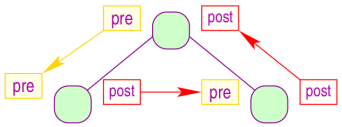
L-Attributation





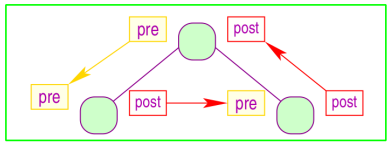
- the attribute system is apparently strongly acyclic

L-Attributation



- the attribute system is apparently strongly acyclic
- each node computes
  - the inherited attributes before descending into a child node (corresponding to a pre-order traversal)
  - the synthesized attributes after returning from a child node (corresponding to post-order traversal)

L-Attributation



- the attribute system is apparently strongly acyclic
- each node computes
  - the inherited attributes before descending into a child node (corresponding to a pre-order traversal)
  - the synthesized attributes after returning from a child node (corresponding to post-order traversal)

Definition L-Attributed Grammars

An attribute system is L-attributed, if for all productions $S \to S_1 \dots S_n$ every inherited attribute of S_j where $1 \leq j \leq n$ only depends on

1. the attributes of $S_1, S_2, \dots S_{j-1}$ and
2. the inherited attributes of S.

L-Attributation

Background:

- the attributes of an L-attributed grammar can be evaluated during parsing
- important if no syntax tree is required or if error messages should be emitted while parsing
- example: pocket calculator

L-Attributation

Background:

- the attributes of an L-attributed grammar can be evaluated during parsing
- important if no syntax tree is required or if error messages should be emitted while parsing
- example: pocket calculator

L-attributed grammars have a fixed evaluation strategy:
a single *depth-first* traversal

- in general: partition all attributes into $\mathcal{A} = A_1 \cup \ldots \cup A_n$ such that for all attributes in A_i the attribute system is L-attributed
- perform a depth-first traversal for each attribute set A_i

$\rightsquigarrow$ craft attribute system in a way that they can be partitioned into few L-attributed sets