Script **generated by TTT**

Title: Petter: Compiler Construction (18.06.2020) - 39: Demand Driven Evaluation

Date: Thu Jun 18 11:46:01 CEST 2020

Duration: 06:47 min

Pages: 9

From Dependencies to Evaluation Strategies

Possible strategies:

1. let the *user* define the evaluation order

From Dependencies to Evaluation Strategies

Possible strategies:

1. let the *user* define the evaluation order
2. *automatic* strategy based on the dependencies

From Dependencies to Evaluation Strategies

Possible strategies:

1. let the *user* define the evaluation order
2. *automatic* strategy based on the dependencies
3. consider a *fixed* strategy and only allow an attribute system that can be evaluated using this strategy

Linear Order from Dependency Partial Order

Possible *automatic* strategies:

Linear Order from Dependency Partial Order

Possible *automatic* strategies:

1. demand-driven evaluation
   - start with the evaluation of any required attribute
   - if the equation for this attribute relies on as-of-yet unevaluated attributes, evaluate these recursively
2. evaluation in passes
   for each pass, pre-compute a global strategy to visit the *nodes* together with a local strategy for evaluation *within each node* type

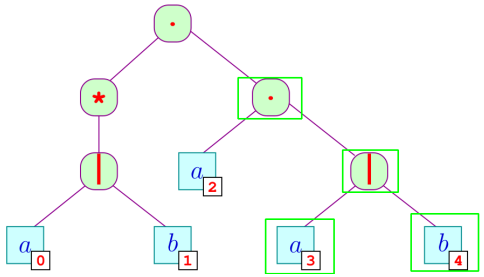
   ⇝ *minimize* the number of *visits* to each node

Example: Demand-Driven Evaluation

Compute next at leaves a_2, a_3 and b_4 in the expression $(a|b)^* a (a|b)$:

$|$: $\mathsf{next}[1] := \mathsf{next}[0]$
$\mathsf{next}[2] := \mathsf{next}[0]$

$\cdot$: $\mathsf{next}[1] := \mathsf{first}[2] \cup (\mathsf{empty}[2] \,?\, \mathsf{next}[0] : \emptyset)$
$\mathsf{next}[2] := \mathsf{next}[0]$

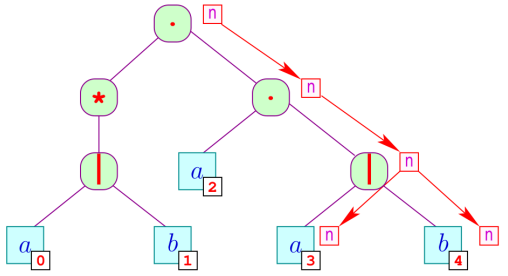


Example: Demand-Driven Evaluation

Compute next at leaves a_2, a_3 and b_4 in the expression $(a|b)^* a (a|b)$:

$|$: $\mathsf{next}[1] := \mathsf{next}[0]$
$\mathsf{next}[2] := \mathsf{next}[0]$

$\cdot$: $\mathsf{next}[1] := \mathsf{first}[2] \cup (\mathsf{empty}[2] \,?\, \mathsf{next}[0] : \emptyset)$
$\mathsf{next}[2] := \mathsf{next}[0]$

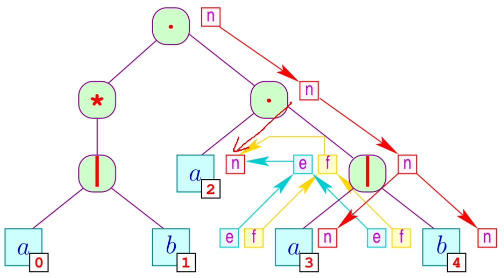


Example: Demand-Driven Evaluation

Compute next at leaves a_2, a_3 and b_4 in the expression $(a|b)^* a(a|b)$:

$\boxed{|}$: next[1] := next[0]
next[2] := next[0]

$\boxed{\cdot}$: next[1] := first[2] ∪ (empty[2] ? next[0] : ∅)
next[2] := next[0]



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