Attribute Dependencies

– Wilhelm/Maurer: Compiler Design, Chapter 9 –

Reinhard Wilhelm
Universität des Saarlandes
wilhelm@cs.uni-sb.de

Attribute Dependencies

Attribute dependencies

- relate attribute occurrences (instances),
- describe which attribute occurrences (instances) depend on which other occurrences (instances),
- constrain the order of attribute evaluation,
- are input to evaluator generators.

Types of Dependencies

Local dependencies between attribute occurrences in a production according to a semantic rule,

Individual dependency graph of attribute instances of a tree obtained by pasting together local dependency graphs of productions (instances)

Global dependencies between attributes of a non-terminal induced by individual dependency graphs.

- An individual dependency graph may contain a cycle. Attribute instances on this cycle can not be evaluated.
- AG is **noncircular** if none of its individual dependency graphs contains a cycle.

Theorem

AG is well–formed iff it is noncircular.

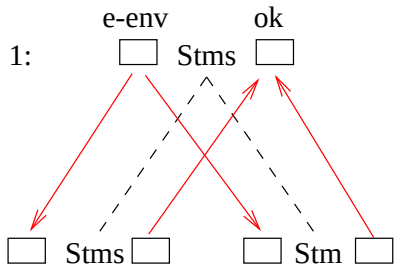
Local Dependencies

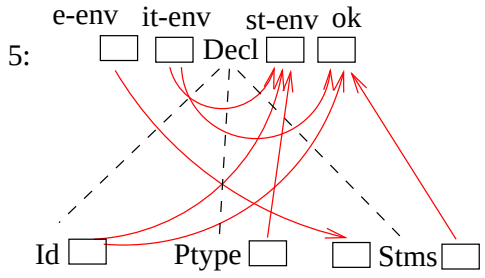
- **production local dependency relation** $Dp(p) \subseteq O(p) \times O(p)$:

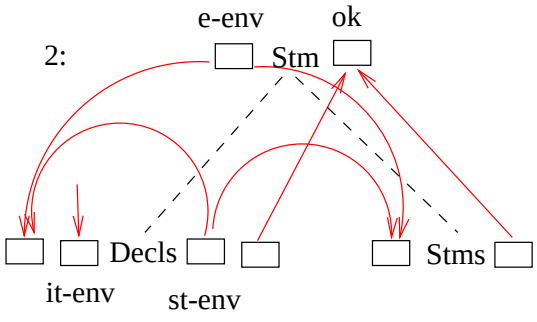
$$b_j \; Dp(p) \; a_i \qquad \text{iff} \qquad a_i = f_{p,a,i}(\ldots, b_j, \ldots)$$

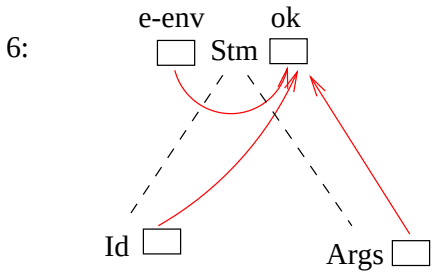
- Attribute occurrence a_i at X_i depends on b_j at X_j iff b_j is argument in the semantic rule of a_i.
- Representation of this relation by its directed graph, the **production local dependency graph**, also denoted by $Dp(p)$.

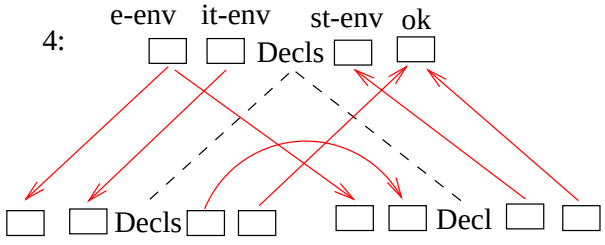
Local Dependencies in the Scopes-AG



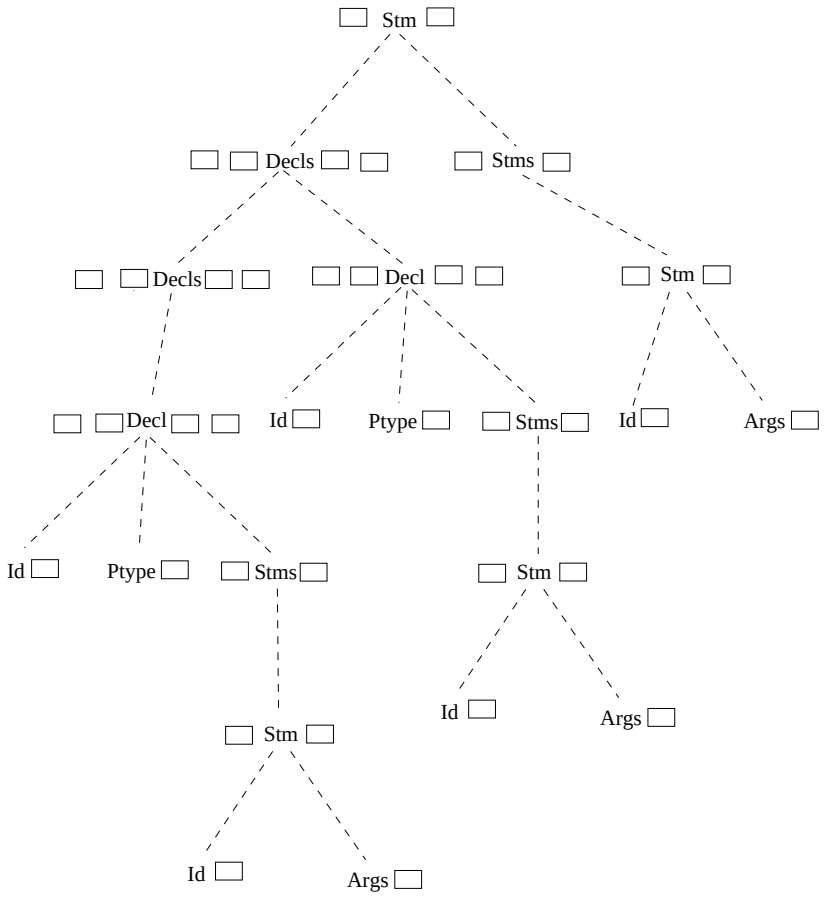




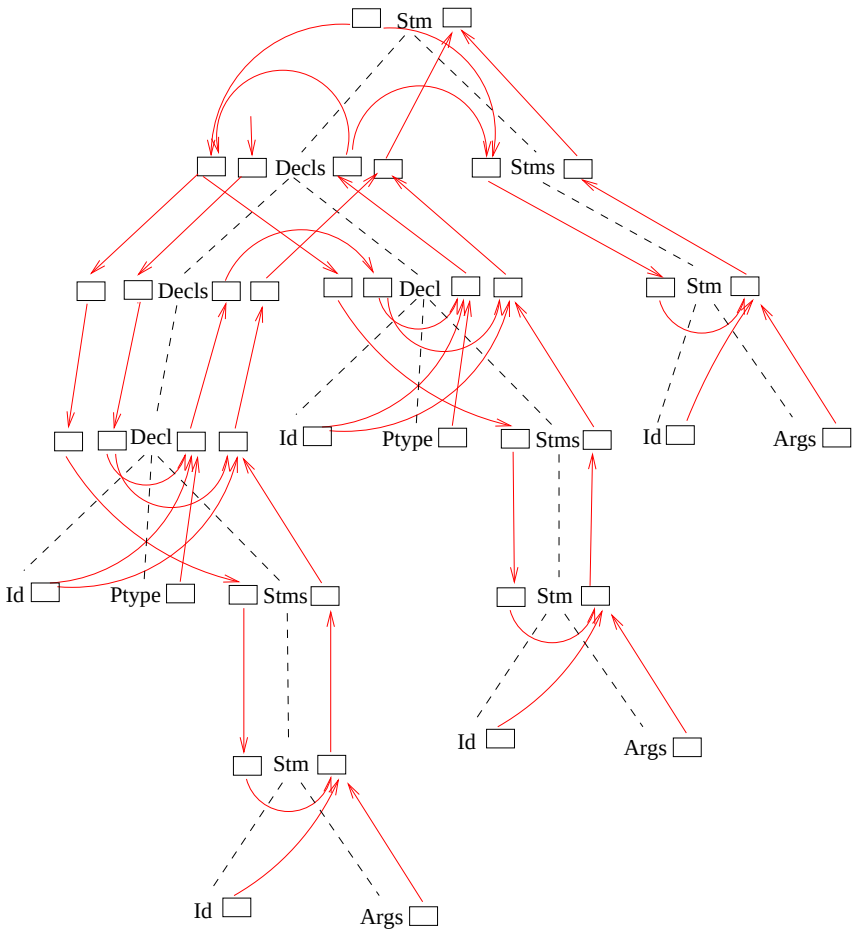




Individual Dependency Graph



Individual Dependency Graphs



A First Attribute Evaluator

Principle:

1. Topological sorting of the individual dependency graph of a tree.
2. Attribute evaluation then done in the resulting order.

Topological sorting

- takes a partial order (an acyclic graph),
- produces a total order compatible with the partial order,
- i.e., resulting total order, an **evaluation order**.

Topological sorting

- Keeps a set of **candidates** to be inserted next into the total order,
  initialized with the minimal elements of the order,
- In each step
  - Selects a candidate and inserts it into the total order,
  - Removes it from the set of candidates,
  - Removes it from the partial order,
  - Makes all elements only depending on this candidate to candidates,
- Until the set of candidates is empty.
- Partial order nonempty $\Rightarrow$ graph acyclic.

Can serve as a *dynamic* test for well formedness.