Attribute Dependencies

Wilhelm/Seidl/Hack: Compiler Design, Volume 2, Chapter 4 – Reinhard Wilhelm
Universität des Saarlandes
wilhelm@cs.uni-saarland.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 cannot 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**
  \[ Dp(p) \subseteq O(p) \times O(p) : \]
  \[ b_j \ Dp(p) \ a_i \ \iff \ a_i = f_{p,a,i}(\ldots, b_j, \ldots) \]

- 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

1: \( e\)-env \( \rightarrow \) ok
   \( \text{Stms} \) \( \rightarrow \) ok
2: \( e\)-env \( \rightarrow \) ok
   \( \text{Stm} \) \( \rightarrow \) ok
3: \( \text{Stms} \) \( \rightarrow \) ok
4: \( e\)-env \( \rightarrow \) ok
   \( \text{Decls} \) \( \rightarrow \) ok
5: \( e\)-env \( \rightarrow \) it-env \( \rightarrow \) st-env \( \rightarrow \) ok
   \( \text{Decl} \) \( \rightarrow \) ok
6: \( e\)-env \( \rightarrow \) ok
   \( \text{Stm} \) \( \rightarrow \) ok
7: \( \text{Id} \) \( \rightarrow \) ok
   \( \text{Ptype} \) \( \rightarrow \) ok
8: \( \text{Stms} \) \( \rightarrow \) ok
   \( \text{Args} \) \( \rightarrow \) ok
Attribute Dependencies

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 \textit{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.
Example Evaluation
Properties of this Evaluator

- Evaluation order determined at evaluation time, i.e. compile time; therefore this evaluator is called the dynamic evaluator,
- Additional effort for the determination of the evaluation order at evaluation time,
- "Data driven" strategy, i.e. the availability of its arguments triggers the evaluation of an instance,
- Evaluates all instances in a tree,
- Evaluates each instance exactly once.
Alternatives

- Evaluation order may be fixed before evaluation time, e.g. by a fixed evaluation “plan” for each production,
- “Demand driven” strategy
  - Starts with a demand of some maximal elements in the partial order,
  - Demand for evaluation is passed to arguments,
  - Computed values are passed back.
- Properties of the demand driven strategy:
  - Allows the selective evaluation of a subset of “interesting” attribute instances,
  - Only instances needed for the evaluation of these attribute instances are evaluated,
  - May evaluate instances several times, depending on the implementation, i.e. on whether computed values are stored.
Attribute Dependencies

Issues

- Separation into
  - **Strategy phase**: Evaluation order is determined,
  - **Evaluation phase**: Evaluation proper of the attribute instances directed by this evaluation strategy.

- Goal: Preparation of the strategy phase at generation time, i.e., evaluation orders, evaluation plans, etc. are precomputed from the AG; may include a *static* test for well formedness,

- Complexity of
  - **Generation**: Runtime in terms of AG size,
  - **Evaluation**: Size of evaluator, time optimality of evaluation.

- AG subclasses, hierarchy: Expressivity, Generation algorithms, Complexity.
Lower Characteristic Graphs

Given $t$, tree with root label $X$

- “Projecting” the dependencies in $Dt(t)$ onto the attributes of $X$ yields the lower characteristic graph of $X$ induced by $t$, $Dt↑_t(X)$.

- $Dt↑_t(X)$ contains an edge from $a \in Inh(X)$ to $b \in Syn(X)$ iff there exists a path from the instance of $a$ at the root to the instance of $b$ at the root in $Dt(t)$. 

\[
\begin{array}{ccccccccc}
    a & b & c & X & d & e \\
    \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
    a & b & c & X & d & e \\
\end{array}
\]
Example: Lower Characteristic Graphs

Lower characteristic graphs induced by the previous individual dependency graph:

\[
\begin{align*}
\text{e-env} & \quad \text{ok} \\
\begin{array}{c}
\begin{array}{c}
\text{Stms} \\
\end{array}
\end{array}
\end{align*}
\]
Upper Characteristic Graphs

$n$ inner node in $t$ labeled $X$, regards the upper tree fragment of $t$ at $n$, $t \backslash n$,

- “Projecting” the dependencies in $Dt(t \backslash n)$ onto the attributes of $X$ yields the upper characteristic graph of $X$ induced by $t$, $Dt_{\downarrow t,n}(X)$.

- $Dt_{\downarrow t,n}(X)$ contains an edge from $a \in \text{Syn}(X)$ to $b \in \text{Inh}(X)$ iff there exists a path from the instance of $a$ at $n$ to the instance of $b$ at $n$ in $Dt(t \backslash n)$. 
Example: Upper Characteristic Graphs

Upper characteristic graphs induced by the previous individual dependency graph:
Strategic Information in Characteristic Graphs

- At the root of a subtree means:
  - `it-env` evaluated $\Rightarrow$ `st-env` can be evaluated
  - `e-env` not evaluated $\Rightarrow$ `ok` cannot be evaluated during a downward visit.

- At the root of a subtree means:
  - `st-env` unevaluated $\Rightarrow$ `e-env` cannot be evaluated during an upward visit;
Induced Global Dependencies

The induction of characteristic graphs:

1. Local dependency graphs, $D_p(p)$: Relation on attribute occurrences of $p$
   Type conversion + Pasting

2. Individual dependency graph, $D_t(t)$: Relation on attribute instances in $t$
   Transitive closure and restriction

3. Relation on attribute instances of node $n$ with $\text{sym}(n) = X$
   Type conversion

4. Lower characteristic graph $D_t↑_t(X) \subseteq \text{Inh}(X) \times \text{Syn}(X)$:
   Relation on attributes of $X$ or

5. Upper characteristic graph $D_t↓_{t,n}(X) \subseteq \text{Syn}(X) \times \text{Inh}(X)$:
   Relation on attributes of $X$. 
Computation of Global Dependency Graphs

- So far, the characteristic graph induced by one tree (fragment).
- Nonterminal $X$ has
  - a set, $D_t^\uparrow(X)$, of lower characteristic graphs and
  - a set, $D_t^\downarrow(X)$, of upper characteristic graphs.
- These sets are computed at generation time by GFA.
- Only non–terminals can contribute,
  i.e., for $p : X_0 \to X_1 \ldots X_{n_p}$ this means $X_i \in V_N$ for all $1 \leq i \leq n_p$.
- Watch out for “typing problems”!
Formalization of “Pasting”

$R_0, R_1, \ldots, R_{np}$ relations on the sets $Attr(X_0), Attr(X_1), \ldots, Attr(X_{np})$, resp.

The pasting operation $Dp(p)[\cdot]$ has functionality $Attr(X_0)^2 \times Attr(X_1)^2 \times \ldots \times Attr(X_{np})^2 \rightarrow O(p) \times O(p)$.

$Dp(p)[R_0, R_1, \ldots, R_{np}]$ is the following relation on $O(p)$:

$$Dp(p) \cup R_0^0 \cup R_1^1 \cup \ldots \cup R_{np}^{np},$$

where $b_i R_i^i a_i$ iff $b R_i a$.

The relations on the attributes of $X_0, X_1, \ldots, X_{np}$ are regarded as relations on attribute occurrences and unioned.

We write $Dp(p)[\emptyset, R_1, \ldots, R_{np}]$ as $Dp(p)[R_1, \ldots, R_{np}]$. 
Formalization of Upward “Projection”

Upward projection $R^\uparrow(p)[\cdot]$ has functionality:
$Attr(X_1)^2 \times \ldots \times Attr(X_{np})^2 \rightarrow Inh(X_0) \times Syn(X_0)$.
$R^\uparrow(p)[R_1, \ldots, R_n]$ is the following relation:

$$b \ R^\uparrow(p)[R_1, \ldots, R_n] \ a \text{ iff } b_0 \ Dp(p)[R_1, \ldots, R_n]^+ \ a_0.$$
Formalization of Downward “Projection”

Downward projection \( R_{\downarrow i}(p)[\cdot] \) has functionality:
\[
\text{Attr}(X_0)^2 \times \text{Attr}(X_1)^2 \times \ldots \times \text{Attr}(X_{np})^2 \to \text{Syn}(X_i) \times \text{Inh}(X_i)
\]

\( R_{\downarrow i}(p)[R_0, R_1, \ldots, R_{np}] \) is defined by
\[
b \ R_{\downarrow i}(p)[R_0, R_1, \ldots, R_{np}] \ a \iff b_i \ \text{Dp}(p)[R_0, R_1, \ldots, R_i-1, \emptyset, R_{i+1}, \ldots, R_{np}]^+ \ a_i
\]
Let $\mathcal{R}_1, \ldots, \mathcal{R}_{np}$ be sets of relations on $Attr(X_1), \ldots, Attr(X_{np})$, resp.

$$R_{\uparrow i}(p)[\mathcal{R}_1, \ldots, \mathcal{R}_{np}] = \{R_{\uparrow i}(p)[R_1, \ldots, R_{np}] \mid R_i \in \mathcal{R}_i, (1 \leq i \leq np)\}$$

$$R_{\downarrow i}(p)[\mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_{np}] = \{R_{\downarrow i}(p)[R_0, R_1, \ldots, R_{np}] \mid R_j \in \mathcal{R}_j (0 \leq j \leq np)\}$$

for all $i$ in $(1 \leq i \leq np)$. 

GFA: Lower Characteristic Graphs

**Evaluation time:**
How to compute $Dt^t(X_0)$ for a tree $t$ with root label $X_0$ and
$prodrhoblock{\varepsilon}_1 = p : X_0 \to X_1 \ldots X_{np}$?

Let the relations $Dt^t_{/1}(X_1), \ldots, Dt^t_{/np}(X_{np})$ be already computed.

![Diagram of a tree with nodes $X_0, X_1, X_i, X_{np}$]  

Compute $Dt^t(X_0)$ locally as

$$Dt^t(X_0) = R^p[Dt^t_{/1}(X_1), \ldots, Dt^t_{/np}(X_{np})]$$
GFA: Lower Characteristic Graphs cont’d

This suggests for the \textit{generation time}:
\[ Dt^\uparrow(X) = \bigcup_{p : p[0] = X} R^\uparrow(p)[Dt^\uparrow(p[1]), \ldots, Dt^\uparrow(p[n_p])]. \]

Least fixpoint is the set of the sets of lower characteristic graphs.
GFA–Problem Lower Characteristic Graphs

One step in the fixpoint iteration for production \( p \):

1. Paste all combinations of lower characteristic graphs onto \( D_p(p) \),
2. Project the resulting graphs onto the attributes of \( X_0 \),
3. Form the union all the resulting sets for \( X_0 \).

<table>
<thead>
<tr>
<th>bottom up-GFA-Problem lower characteristic graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>lattices ( {D(X) = \mathcal{P}(\mathcal{P}(\text{Inh}(X) \times \text{Syn}(X)))} ) ( X \in V_N )</td>
</tr>
<tr>
<td>part. order ( \subseteq ) (subset inclusion on sets of relations)</td>
</tr>
<tr>
<td>bottom ( \emptyset ) (empty set of relations)</td>
</tr>
<tr>
<td>transf. fct. ( {Lc_p : D(p[1]) \times \ldots \times D(p[n_p]) \to D(p[0]) \mid Lc_p(\mathcal{R}<em>1, \ldots, \mathcal{R}</em>{n_p}) = R\uparrow(p)[\mathcal{R}<em>1, \ldots, \mathcal{R}</em>{n_p}] } ) ( p \in \mathcal{P} )</td>
</tr>
<tr>
<td>comb. fct. ( \cup ) (union on sets of relations)</td>
</tr>
</tbody>
</table>
A Static Non–circularity Test

- A lower char. graph represents all dependencies in the trees inducing it.
- Pasting all combinations of lower char. graphs onto local dep. graphs produces a cyclic graph if AG is circular. Hence:
  - AG is noncircular iff all graphs in $Dp(p)[Dt^\uparrow(X_1), \ldots, Dt^\uparrow(X_{n_p})]$ for all productions $p$ are noncyclic.
  - $|\bigcup_X Dt^\uparrow(X)|$ exponential in $|Attr|$.
- The non–circularity test is exponential.
GFA: Upper Characteristic Graphs

**Compile time:**

Regard $p$ applied at node $n$ in $t$.

Already computed

$$Dt_{t,n}(X_0)$$ and $$Dt_{t/n1}(X_1), \ldots, Dt_{t/nn_p}(X_{n_p}).$$

Compute $$Dt_{t,n_i}(X_i) \quad (1 \leq i \leq n_p)$$ using the operation $$R{\downarrow}_i(p)[\ldots].$$
GFA: Upper Characteristic Graphs cont’d

This suggests for generation time:

\[ Dt \downarrow(S) = \{\emptyset\} \]

\[ Dt \downarrow(X) = \bigcup_{p[i]=X} R \downarrow_i(p)[Dt \downarrow(p[0]), Dt \uparrow(p[1]), \ldots, Dt \uparrow(p[n_p])] \]

Least fixpoint is the set of the sets of upper characteristic graphs.
GFA–Problem Upper Characteristic Graphs

The sets of lower characteristic graphs are assumed to be computed before.

They are constant parts of the functions $Uc_{p,i}$. 
Characteristic graphs are:

**Exact:**  For each characteristic graph there is at least one tree (fragment), whose individual dependency graph induces it,

**Costly:**  There may be exponentially many of them.
Approximative Attribute Dependencies

What is the “strategic” interpretation of edges in (lower) characteristic graphs?

Evaluator visits subtree at $n$ with $b, c$ evaluated. Through this visit, it can

- evaluate $d$ and $e$,
- not evaluate $f$. 
What does “approximation” mean? deleting edges? adding edges?
Deleting the edge from $a$ to $f$:

- Evaluator assumes, $f$ can be evaluated when value of $b$ is known.
- Makes a fruitless visit to the subtree at $n$.
- Inefficient strategy!
Adding edges from $a$ to $d$ and $e$:

- Evaluator would not visit the subtree at $n$ with evaluated $b$ and $c$ and unevaluated $a$,
- Evaluator would only visit the subtree, when also the value of $a$ is known.
- Visits may be delayed.
Resumee:

- Reduced dependency graphs may cause fruitless visits,
- Augmented dependency graphs may delay visits,
- Added edges may introduce cycles (cause an infinite delay).
I/O–Graphs

- Are an upper bound on the lower dependencies,
- There may be I/O–graphs with no corresponding tree,
- There is one graph per nonterminal.

### bottom up-GFA-problem I/O-graphs

<table>
<thead>
<tr>
<th>lattices</th>
<th>( { D(X) = \mathcal{P}(\text{Inh}(X) \times \text{Syn}(X)) } ) for ( X \in V_N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>part. order</td>
<td>( \subseteq ) (subset inclusion on relations)</td>
</tr>
<tr>
<td>bottom</td>
<td>( \emptyset )</td>
</tr>
</tbody>
</table>
| transf. fct.      | \( \{ \text{lo} : D(p[1]) \times \ldots \times D(p[n_p]) \to D(p[0]) \mid \)
|                   | \( \text{lo}(g_1, \ldots, g_{n_p}) = R^{\uparrow}(p)[g_1, \ldots, g_{n_p}] \} \) for \( p \in \mathcal{P} \) |
| comb. fct.        | \( \cup \) (union on relations)                                                  |

Yields the system of equations:

\[
\text{IO}(X) = \bigcup_{p : p[0] = X} R^{\uparrow}(p)[\text{IO}(p[1]), \ldots, \text{IO}(p[n_p])]
\]

AG is absolutely noncircular if for all productions \( p \) the graph \( D_p(p)[\text{IO}(p[1]), \ldots, \text{IO}(p[n_p]) \] is acyclic.
A Noncircular, but not Absolutely Noncircular AG

Its only two trees have no cyclic dependencies.

For computing $IO(X)$ $Dp(2)$ and $Dp(3)$ are unioned and inserted in $Dp(1)$ producing a cycle.