Grammar Flow Analysis

– Wilhelm/Maurer: Compiler Design, Chapter 8 –

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

25. Oktober 2011
Generators for compiler components require information about the language. This information is collected on the specification of the language, 

- the context-free grammar describing its syntax,
- the attribute grammar describing its static semantics.

**Grammar flow analysis** is a static analysis of grammars computing such information.
## Notation

<table>
<thead>
<tr>
<th>Generic names</th>
<th>for</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A, B, C, X, Y, Z$</td>
<td>Non-terminal symbols</td>
</tr>
<tr>
<td>$a, b, c, \ldots$</td>
<td>Terminal symbols</td>
</tr>
<tr>
<td>$u, v, w, x, y, z$</td>
<td>Terminal strings</td>
</tr>
<tr>
<td>$\alpha, \beta, \gamma, \varphi, \psi$</td>
<td>Strings over $V_N \cup V_T$</td>
</tr>
<tr>
<td>$p, p', p_1, p_2, \ldots$</td>
<td>Productions</td>
</tr>
</tbody>
</table>

- **Standard notation for production**
  
  $p = (X_0 \rightarrow u_0X_1u_1 \ldots X_nu_n)$

  $n_p$ – **Arity** of $p$

- $(p, i)$ – Position $i$ in production $p$ ($0 \leq i \leq n_p$)

- $p[i]$ stands for $X_i$, ($0 \leq i \leq n_p$).

- $X$ **occurs** at position $i$ – $p[i] = X$
Reachability and Productivity

Non-terminal $A$ is

reachable \iff there exist $\varphi_1, \varphi_2 \in V_T \cup V_N$ such that
\[ S \xrightarrow{*} \varphi_1 A \varphi_2 \]

productive \ iff there exists $w \in V_T^*$, $A \xrightarrow{*} w$

These definitions are useless for tests; they involve quantifications over infinite sets.
A two level Definition

1. A nonterminal is reachable through its occurrence $(p, i)$ with $i > 0$ iff $p[0]$ is reachable,
2. A nonterminal is reachable iff it is reachable through at least one of its occurrences,
3. $S'$ is reachable.

1. A nonterminal $A$ is productive through production $p$ iff $A = p[0]$ and all nonterminals on the right side are productive.
2. A nonterminal is productive iff it is productive through at least one of its alternatives.

- Reachability and productivity for a grammar given by a (recursive) system of equations.
- Least solution wanted to eliminate as many useless nonterminals as possible.
Typical Two Level Simultaneous Recursion

Productivity:
1. property of left side nonterminal depends on the properties of the right side nonterminals,
2. combination of the information from the different alternatives for a nonterminal.

Reachability:
1. property of occurrences of nonterminals on the right side depends on the property of the left side nonterminal,
2. combination of the information from the different occurrences for a nonterminal,
3. the initial property.

In the specification
1. given by transfer functions
2. given by combination functions
Schema for the Computation

- **Grammar Flow Analysis (GFA)** computes a property function $I : V_N \rightarrow D$ where $D$ is some domain of information for nonterminals, mostly properties of sets of trees,

- Productivity computed by a **bottom-up Grammar Flow Analysis (bottom-up GFA)**

- Reachability computed by a **top-down Grammar Flow Analysis (top-down GFA)**
Trees, Subtrees, Tree Fragments

$S$

Parse tree

Subtree for $X$

upper tree fragment for $X$

$X$ reachable: Set of upper tree fragments for $X$ not empty,

$X$ productive: Set of subtrees for $X$ not empty.
Bottom-up GFA

Given a cfg $G$.

A **bottom-up GFA-problem** for $G$ and a property function $I$:

**D**: a domain $D^\uparrow$,

**T**: transfer functions $F_p^\uparrow: D^\uparrow^{n_p} \rightarrow D^\uparrow$ for each $p \in P$,

**C**: a combination function $\nabla^\uparrow: 2^{D^\uparrow} \rightarrow D^\uparrow$.

This defines a system of equations for $G$ and $I$:

$$I(X) = \nabla^\uparrow \{ F_p^\uparrow (I(p[1]), \ldots, I(p[n_p])) \mid p[0] = X \} \quad \forall X \in V_N$$

$E \rightarrow ET$ \quad $\text{Pr}(E), \text{Pr}(T)$
Top-down GFA

Given a cfg $G$.
A top down – GFA-problem for $G$ and a property function $I$:

**D:** a domain $D\downarrow$;

**T:** $n_p$ transfer functions $F_{p,i}\downarrow: D\downarrow \rightarrow D\downarrow$, $1 \leq i \leq n_p$, for each production $p \in P$,

**C:** a combination function $\nabla\downarrow: 2^{D\downarrow} \rightarrow D\downarrow$,

**S:** a value $l_0$ for $S$ under the function $I$.

A top-down GFA-problem defines a system of equations for $G$ and $I$:

$$
I(S) = l_0
$$

$$
l(p,i) = F_{p,i}\downarrow (l(p[0])) \text{ for all } p \in P, 1 \leq i \leq n_p
$$

$$
l(X) = \nabla\downarrow \{l(p,i) \mid p[i] = X\}, \text{ for all } X \in V_N - \{S\}
$$
Recursive System of Equations

Systems like (I↑) and (I↓) are in general recursive. Questions: Do they have

- solutions?
- unique solutions?
They do have solutions if
- the domain
  - is partially ordered by some relation \( \sqsubseteq \),
  - has a uniquely defined smallest element, \( \bot \),
  - has a least upper bound, \( d_1 \sqcup d_2 \), for each two elements \( d_1, d_2 \)
  - and has only finitely ascending chains,

and

- the transfer and the combination functions are monotonic.

Our domains are finite, all functions are monotonic.
Fixpoint Iteration

- Solutions are fixpoints of a function $I : [V_N \rightarrow D] \rightarrow [V_N \rightarrow D]$.
- Computed iteratively starting with $\bot$, the function which maps all nonterminals to $\bot$.
- Apply transfer functions and combination functions until nothing changes.

We always compute least fixpoints.
Productivity Revisited

\[
D \uparrow \{false \sqsubseteq true\} \quad \text{true for productive}
\]

\[
F_p \uparrow \land \quad (true \text{ for } n_p = 0)
\]

\[
\Delta \uparrow \lor \quad (false \text{ for nonterminals without productions})
\]

Domain: \( D \uparrow \) satisfies the conditions,

transfer functions: conjunctions are monotonic,

combination function: disjunction is monotonic.

Resulting system of equations:

\[
Pr(X) = \lor\{\land_{i=1}^{n_p} Pr(p[i]) | p[0] = X\} \quad \text{for all } X \in V_N
\]

(Pr)
Example: Productivity

Given the following grammar:

\[ G = (\{S', S, X, Y, Z\}, \{a, b\}, \{(S' \rightarrow S), (S \rightarrow aX), (X \rightarrow bS \mid aYbY), (Y \rightarrow ba \mid aZ), (Z \rightarrow aZX)\}, S') \]

Resulting system of equations:

\[
\begin{align*}
Pr(S) &= Pr(X) \\
Pr(X) &= Pr(S) \lor Pr(Y) \\
Pr(Y) &= true \lor Pr(Z) = true \\
Pr(Z) &= Pr(Z) \land Pr(X)
\end{align*}
\]

Fixpoint iteration

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>Fixpoint</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

True | False
Reachability Revisited

\[ D \downarrow \quad \text{false} \subseteq \{ \text{true} \} \quad \text{true for reachable} \]

\[ F_{p,i} \downarrow \quad \text{id} \quad \text{identity mapping} \]

\[ \nabla \downarrow \quad \vee \quad \text{Boolean Or (false, if there is no occ. of the nonterminal)} \]

\[ l_0 \quad \text{true} \]

Domain: \( D \downarrow \) satisfies the conditions,

transfer functions: identity is monotonic,

combination function: disjunction is monotonic.

Resulting system of equations for reachability:

\[
\begin{align*}
Re(S) &= \text{true} \\
Re(X) &= \bigvee \{ Re(p[0]) \mid p[i] = X, 1 \leq i \leq n_p \} \forall X \neq S
\end{align*}
\]

\((Re)\)
Example: Reachability

Given the grammar \( G = (\{S, U, V, X, Y, Z\}, \{a, b, c, d\}, \) \)

The equations:

\[
\begin{align*}
S & \rightarrow Y \\
Y & \rightarrow YZ \mid Ya \mid b \\
U & \rightarrow V \\
X & \rightarrow c \\
V & \rightarrow Vd \mid d \\
Z & \rightarrow ZX
\end{align*}
\]

Fixpoint iteration:

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>U</th>
<th>V</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
Re(S) &= true \\
Re(U) &= false \\
Re(V) &= Re(U) \lor Re(V) \\
Re(X) &= Re(Z) \\
Re(Y) &= Re(S) \lor Re(Y) \\
Re(Z) &= Re(Y) \lor Re(Z)
\end{align*}
\]
First and Follow Sets

Parser generators need precomputed information about sets of

- prefixes of words for nonterminals (words that can begin words for non-terminals)
- followers of nonterminals (words which can follow a nonterminal).

Strategic use: **Removing non-determinism from expand moves of the** $P_G$

These sets can be computed by GFA.
Another Grammar for Arithmetic Expressions

Left-factored grammar $G_2$, i.e. left recursion removed.

$$
S \rightarrow E \\
E \rightarrow TE' \\
E' \rightarrow +E|\epsilon \\
T \rightarrow FT' \\
T' \rightarrow *T|\epsilon \\
F \rightarrow id|(E)
$$

$G_2$ defines the same language as $G_0$ and $G_1$. 
The $\text{FIRST}_1$ Sets

- A production $N \rightarrow \alpha$ is applicable for symbols that “begin” $\alpha$
- Example: Arithmetic Expressions, Grammar $G_2$
  - The production $F \rightarrow id$ is applied when the current symbol is $id$
  - The production $F \rightarrow (E)$ is applied when the current symbol is $(
  - The production $T \rightarrow F$ is applied when the current symbol is $id$ or $(
- Formal definition:

$$\text{FIRST}_1(\alpha) = \{ 1 : w \mid \alpha \Rightarrow^* w, w \in V_T^* \}$$
The \textit{FOLLOW}_1 Sets

- A production $N \rightarrow \epsilon$ is applicable for symbols that “can follow” $N$ in some derivation.
- Example: Arithmetic Expressions, Grammar $G_2$
  - The production $E' \rightarrow \epsilon$ is applied for symbols $\# \text{ and } )$
  - The production $T' \rightarrow \epsilon$ is applied for symbols $\#, ) \text{ and } +$
- Formal definition:

\[
FOLLOW_1(N) = \{a \in V_T | \exists \alpha, \gamma : S \Rightarrow^* \alpha Na \gamma\}
\]
Definitions

Let \( k \geq 1 \)

**k-prefix** of a word \( w = a_1 \ldots a_n \)

\[ k : w = \begin{cases} a_1 \ldots a_n & \text{if } n \leq k \\ a_1 \ldots a_k & \text{otherwise} \end{cases} \]

**k-concatenation**

\( \oplus_k : V^* \times V^* \rightarrow V^{\leq k} \), defined by \( u \oplus_k v = k : uv \)

extended to languages

\( k : L = \{ k : w \mid w \in L \} \)

\( L_1 \oplus_k L_2 = \{ x \oplus_k y \mid x \in L_1, y \in L_2 \} \).

\( V^{\leq k} = \bigcup_{i=1}^{k} V^i \) set of words of length at most \( k \) ...

\( V^{\leq k}_{T\#} = V^{\leq k}_T \cup V^{k-1}_T \{\#\} \) ... possibly terminated by \#. 
**FIRST**\(_k\) and **FOLLOW**\(_k\)

\[ \text{FIRST}_k : (V_N \cup V_T)^* \rightarrow 2^{V_T} \leq_k \] where

\[ \text{FIRST}_k(\alpha) = \{ k : u \mid \alpha \Rightarrow^* u \} \]

set of \(k\)-prefixes of terminal words for \(\alpha\).

\[ \text{FOLLOW}_k : V_N \rightarrow 2^{V_T \#} \leq_k \]

where

\[ \text{FOLLOW}_k(X) = \{ w \mid S \Rightarrow^* \beta X \gamma \text{ and } w \in \text{FIRST}_k(\gamma) \} \]

set of \(k\)-prefixes of terminal words that may immediately follow \(X\).
GFA-Problem $FIRST_k$

\[
\text{bottom up-GFA-problem } FIRST_k
\]

\[
\begin{align*}
L &\quad (2^{V_T^k}, \subseteq, \emptyset, \cup) \\
T &\quad Fir_p(d_1, \ldots, d_{n_p}) = \{u_0\} \oplus_k d_1 \oplus_k \{u_1\} \oplus_k d_2 \oplus_k \ldots \oplus_k d_{n_p} \oplus_k \{u_{n_p}\}, \\
&\quad \quad \text{if } p = (X_0 \rightarrow u_0 X_1 u_1 X_2 \ldots X_{n_p} u_{n_p}); \\
&\quad Fir_p = k : u \text{ for a terminal production } X \rightarrow u \\
C &\quad \cup \\
\end{align*}
\]

The recursive system of equations for $FIRST_k$ is

\[
Fi_k(X) = \bigcup_{\{p | p[0] = X\}} Fir_p(Fi_k(p[1]), \ldots, Fi_k(p[n_p])) \quad \forall X \in V_N
\]

\[(Fi_k)\]
**FIRST**$_k$ Example

The bottom up-GFA-problem **FIRST**$_1$ for grammar $G_2$ with the productions:

0 : $S$ $\rightarrow$ $E$  
3 : $E'$ $\rightarrow$ $+E$  
6 : $T'$ $\rightarrow$ $^*T$

1 : $E$ $\rightarrow$ $TE'$  
4 : $T$ $\rightarrow$ $FT'$  
7 : $F$ $\rightarrow$ $(E)$

2 : $E'$ $\rightarrow$ $\varepsilon$  
5 : $T'$ $\rightarrow$ $\varepsilon$  
8 : $F$ $\rightarrow$ $\text{id}$

$G_2$ defines the same language as $G_0$ und $G_1$.

The transfer functions for productions 0 – 8 are:

$Fir_0(d) = d$

$Fir_1(d_1, d_2) = Fir_4(d_1, d_2) = d_1 \oplus_1 d_2$

$Fir_2 = Fir_5 = \{\varepsilon\}$

$Fir_3(d) = \{+\}$

$Fir_6(d) = \{*\}$

$Fir_7(d) = \{(\}\}$

$Fir_8 = \{\text{id}\}$
Iteration

Iterative computation of the $FIRST_1$ sets:

<table>
<thead>
<tr>
<th></th>
<th>$S$</th>
<th>$E$</th>
<th>$E'$</th>
<th>$T$</th>
<th>$T'$</th>
<th>$F$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>
GFA-Problem $FOLLOW_k$

---

**top down-GFA-problem $FOLLOW_k$**

$L = (2^\tau_{\#}, \subseteq, \emptyset, \cup)$

$T F_{ol_{p,i}}(d) = \{u_i\} \oplus_k F_{i_k}(X_{i+1}) \oplus_k \{u_{i+1}\} \oplus_k \ldots \oplus_k F_{i_k}(X_{n_p}) \oplus_k \{u_{n_p}\} \oplus_k d$

if $p = (X_0 \rightarrow u_0 X_1 u_1 X_2 \ldots X_{n_p} u_{n_p})$;

$C \cup$

$S = \{\#\}$

The resulting system of equations for $FOLLOW_k$ is

$F_{ol_{k}}(X) = \bigcup_{\{p|p[i] = X, 1 \leq i \leq n_p\}} F_{ol_{p,i}}(F_{ol_{k}}(p[0])) \forall X \in V_N - \{S\}$

$F_{ol_{k}}(S) = \{\#\}$

$(F_{ol_{k}})$
FOLLOW\(_k\) Example

Regard grammar \(G_2\). The transfer functions are:

\[
\begin{align*}
Fol_{0,1}(d) &= d \\
Fol_{1,1}(d) &= Fi_1(E') \oplus_1 d = \{+, \varepsilon\} \oplus_1 d, \\
Fol_{1,2}(d) &= d \\
Fol_{3,1}(d) &= d \\
Fol_{4,1}(d) &= Fi_1(T') \oplus_1 d = \{\ast, \varepsilon\} \oplus_1 d, \\
Fol_{4,2}(d) &= d \\
Fol_{6,1}(d) &= d \\
Fol_{7,1}(d) &= \{\}\}
\end{align*}
\]

Iterative computation of the FOLLOW\(_1\) sets:

<table>
<thead>
<tr>
<th></th>
<th>(S)</th>
<th>(E)</th>
<th>(E')</th>
<th>(T)</th>
<th>(T')</th>
<th>(F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>{#}</td>
<td>(\emptyset)</td>
<td>(\emptyset)</td>
<td>(\emptyset)</td>
<td>(\emptyset)</td>
<td>(\emptyset)</td>
<td>(\emptyset)</td>
</tr>
</tbody>
</table>