Grammar Flow Analysis

- Wilhelm/Maurer: Compiler Design, Chapter 8 –

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

## 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
  \[ p = (X_0 \rightarrow u_0X_1u_1 \ldots X_{n_p}u_{n_p}) \]
  \[ n_p - \text{Arity of } p \]
- $(p, i) -$ 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 $S \Rightarrow^* \varphi_1 A \varphi_2$

productive: iff there exists $w \in V_T^*$, $A \Rightarrow^* w$

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

1. A non-terminal is **reachable through its occurrence** \((p, i)\) iff \(p[0]\) is reachable,
2. A non-terminal is **reachable** iff it is reachable through at least one of its occurrences,
3. \(S'\) is reachable.

1. A non-terminal \(A\) is **productive through production** \(p\) iff \(A = p[0]\) and all non-terminals \(p[i](1 \leq i \leq n_p)\) are productive.
2. A non-terminal 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 non-terminals as possible.
Typical Two Level Simultaneous Recursion

Productivity:
1. dependence of property of left side non-terminal on right side non-terminals,
2. combination of the information from the different alternatives for a non-terminal.

Reachability:
1. dependence of property of occurrences of non-terminals on the right side on the property of the left side non-terminal,
2. combination of the information from the different occurrences for a non-terminal,
3. the initial property.

In the specification

1. given by transfer functions
2. given by combination functions
Grammar Flow Analysis

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 non-terminals, 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

$S$

$X$

Parse tree

$S$

$X$

Subtree for $X$

$S$

$X$

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 $D↑$,
- **T**: transfer functions $F_p↑: D↑^{n_p} → D↑$ for each $p ∈ P$,
- **C**: a combination function $∇↑: 2^{D↑} → D↑$.

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

\[
I(X) = ∇↑\{F_p↑ (I(p[1]), \ldots, I(p[n_p])) | p[0] = X\} \quad ∀X ∈ V_N
\]

\[ (I↑) \]
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 $I_0$ for $S$ under the function $I$.

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

\[
\begin{align*}
I(S) & = I_0 \\
I(p, i) & = F_{p, i\downarrow}(I(p[0])) \quad \text{for all } p \in P, \ 1 \leq i \leq n_p \\
I(X) & = \nabla\downarrow \{I(p, i) \mid p[i] = X\}, \quad \text{for all } X \in V_N - \{S\}
\end{align*}
\]
Recursive System of Equations

Systems like \((I\uparrow)\) and \((I\downarrow)\) 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 non-terminals to $\bot$.  
- Apply transfer functions and combination functions until nothing changes.
Productivity Revisited

\[ D^\uparrow \{ \text{false} \sqsubseteq \text{true} \} \quad \text{true for productive} \]
\[ F_p^\uparrow \land \quad (\text{true for } n_p = 0) \]
\[ \nabla^\uparrow \lor \quad (\text{false for non-terminals without productions}) \]

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

**transfer functions:** conjunctions are monotonic,

**combination function:** disjunction is monotonic.

Resulting system of equations:

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

\((Pr)\)
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:

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

<table>
<thead>
<tr>
<th>Fixpoint iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
</tr>
<tr>
<td>false</td>
</tr>
</tbody>
</table>
Reachability Revisited

\[ D \downarrow \quad false \sqsubseteq \{true\} \quad true \text{ for reachable} \]
\[ F_{p,i} \downarrow \quad id \quad \text{identity mapping} \]
\[ \nabla \downarrow \quad \lor \quad \text{Boolean Or (false, if there is no occ. of the non-terminal)} \]

\[ l_0 \quad true \]

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

Transfer functions: identity is monotonic,

Combination function: disjunction is monotonic.

Resulting system of equations for reachability:

\[ Re(S) = true \]
\[ Re(X) = \lor \{ Re(p[0]) \mid p[i] = X, 1 \leq i \leq n_p \} \forall X \neq S \]  \((Re)\)
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>
</tbody>
</table>
First and Follow Sets

Parser generators need precomputed information about sets of

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

Strategic use: Removing non-determinism from expand moves of the $P_G$
These sets can be computed by GFA.
The $FIRST_1$ 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:

$$FIRST_1(\alpha) = \{ a \in V_T | \exists \gamma : \alpha \Rightarrow^* a\gamma \}$$
Another Grammar for Arithmetic Expressions

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

$$S \rightarrow E$$
$$E \rightarrow TE'$$  \hspace{1cm} (E \text{ generates } T \text{ with a continuation } E')
$$E' \rightarrow +E|\epsilon$$  \hspace{1cm} (E' \text{ generates possibly empty sequence of } +T\text{s})
$$T \rightarrow FT'$$  \hspace{1cm} (T \text{ generates } F \text{ with a continuation } T')
$$T' \rightarrow *T|\epsilon$$  \hspace{1cm} (T' \text{ generates possibly empty sequence of } *F\text{s})
$$F \rightarrow \text{id}|(E)$$

$G_2$ defines the same language as $G_0 \text{ und } G_1$. 
The $FOLLOW_1$ 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 $#$ and $)$
  - The production $T' \rightarrow \epsilon$ is applied for symbols $#$, $)$ and $+$
- Formal definition:

$$FOLLOW_1(N) = \{ a \in V_T | \exists \alpha, \gamma : S \xrightarrow{*} \alpha Na\gamma \}$$
Definitions

Let \( k \geq 1 \)

\( k \)-prefix of a word \( w = a_1 \ldots a_n \)
\[
k : w = \left\{ \begin{array}{ll}
a_1 \ldots a_n & \text{if } n \leq k \\
a_1 \ldots a_k & \text{otherwise}
\end{array} \right.
\]

\( k \)-concatenation

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

extended to languages

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

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

\[
FIRST_k : (V_N \cup V_T)^* \rightarrow 2^V_T \quad \text{where}
\]

\[
FIRST_k(\alpha) = \{ k : u \mid \alpha \xrightarrow{*} u \}
\]

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

\[
FOLLOW_k : V_N \rightarrow 2^{V_T \#} \quad \text{where}
\]

\[
FOLLOW_k(X) = \{ w \mid S \xrightarrow{*} \beta X \gamma \text{ and } w \in FIRST_k(\gamma) \}
\]

set of \( k \)-prefixes of terminal words that may immediately follow \( X \).
GFA-Problem $\text{FIRST}_k$

---

**bottom up-GFA-problem $\text{FIRST}_k$**

**L** $(2^{V^<k}, \subseteq, \emptyset, \cup)$

**T** $\text{Fir}_p(d_1, \ldots, d_{n_p}) = \{u_0\} \oplus_k d_1 \oplus_k \{u_1\} \oplus_k d_2 \oplus_k \ldots \oplus_k d_{n_p} \oplus_k \{u_{n_p}\}$, if $p = (X_0 \to u_0 X_1 u_1 X_2 \ldots X_{n_p} u_{n_p})$;

$\text{Fir}_p = k : u$ for a terminal production $X \to u$

**C** $\cup$

The recursive system of equations for $\text{FIRST}_k$ is

$$\text{Fi}_k(X) = \bigcup \{ \text{Fir}_p(\text{Fi}_k(p[1]), \ldots, \text{Fi}_k(p[n_p])) \mid \{p \mid p[0] = X\} \forall X \in V_N$$

($\text{Fi}_k$)
**FIRST$_k$ Example**

The bottom up-GFA-problem $FIRST_1$ for grammar $G_2$ with the productions:

$$
0 : \quad S \quad \rightarrow \quad E \\
3 : \quad E' \quad \rightarrow \quad +E \\
6 : \quad T' \quad \rightarrow \quad *T \\
1 : \quad E \quad \rightarrow \quad TE' \\
4 : \quad T \quad \rightarrow \quad FT' \\
7 : \quad F \quad \rightarrow \quad (E) \\
2 : \quad E' \quad \rightarrow \quad \varepsilon \\
5 : \quad T' \quad \rightarrow \quad \varepsilon \\
8 : \quad F \quad \rightarrow \quad \text{id}
$$

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

The transfer functions for productions 0 – 8 are:

- $Fir_0(d) = d$
- $Fir_1(d_1, d_2) = Fir_4(d_1, d_2) = d_1 \oplus_1 d_2$
- $Fir_2 = Fir_5 = \{\varepsilon\}$
- $Fir_3(d) = \{+\}$
- $Fir_6(d) = \{\ast\}$
- $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 \( \textit{FOLLOW}_k \)

---

top down-GFA-problem \( \textit{FOLLOW}_k \)

\[ \begin{align*}
\mathbf{L} & \quad (2^{\mathcal{V}_T}, \subseteq, \emptyset, \cup) \\
\mathbf{T} & \quad \text{Fol}_{p,i}(d) = \{u_i\} \oplus_k F_i(X_{i+1}) \oplus_k \{u_{i+1}\} \oplus_k \ldots \oplus_k F_i(X_{n_p}) \oplus_k \{u_{n_p}\} \oplus_k d \\
& \quad \text{if } p = (X_0 \rightarrow u_0 X_1 u_1 X_2 \ldots X_{n_p} u_{n_p}); \\
\mathbf{C} & \quad \cup \\
\mathbf{S} & \quad \{\#\}
\end{align*} \]

The resulting system of equations for \( \textit{FOLLOW}_k \) is

\[ \begin{align*}
\text{Fo}_k(X) &= \bigcup_{\{p|p[i] = x, 1 \leq i \leq n_p\}} \text{Fol}_{p,i}(\text{Fo}_k(p[0])) \quad \forall X \in V_N - \{S\} \\
\text{Fo}_k(S) &= \{\#\} \\
(Fo_k)
\end{align*} \]
**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>$S$</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>
</tr>
</tbody>
</table>