Grammar Flow Analysis

– Wilhelm/Maurer: Compiler Design, Chapter 8 –

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

2. November 2009

Notation

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

- Standard notation for production
  $p = (X_0 \longrightarrow u_0 X_1 u_1 \ldots X_{n_p} u_{n_p})$
  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 $S \stackrel{*}{\Longrightarrow} \varphi_1 A \varphi_2$

productive: iff there exists $w \in V_T^*$, $A \stackrel{*}{\Longrightarrow} w$

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.

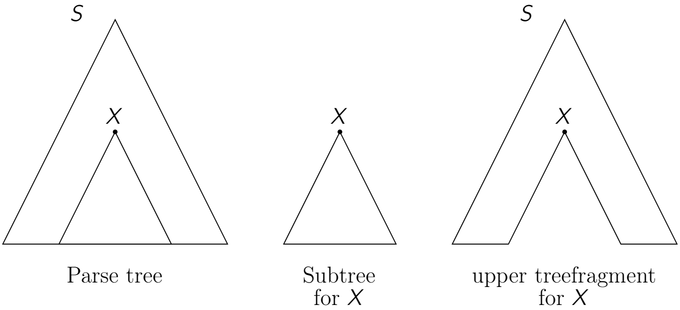
In the specification

1. given by **transfer functions**
2. given by **combination functions**

Schema for the Computation

- **Grammar Flow Analysis (GFA)** computes a property function $I : V_N \longrightarrow D$
  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



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** $F_p\uparrow\colon D\uparrow^{n_p} \to D\uparrow$ for each $p \in P$,

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

This defines a system of equations for G and I:

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

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** $F_{p,i}\downarrow\colon D\downarrow \rightarrow D\downarrow$, $1 \leq i \leq n_p$, for each production $p \in P$,
- C: a **combination function** $\nabla\downarrow\colon 2^{D\downarrow} \rightarrow D\downarrow$,
- 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

$$\begin{array}{lcl} I(S) & = & I_0 \\ I(p,i) & = & F_{p,i}\downarrow\,(I(p[0]))\ \text{for all}\ p \in P,\ 1 \leq i \leq n_p \\ I(X) & = & \nabla\downarrow\,\{I(p,i) \mid p[i] = X\},\ \text{for all}\ X \in V_N - \{S\} \end{array} \qquad (I\downarrow)$$

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
  $\mathbf{I} : [V_N \to D] \to [V_N \to D]$.
- Computed iteratively starting with $\perp\!\!\perp$, the function which maps all non-terminals to $\perp$.
- Apply transfer functions and combination functions until nothing changes.

We always compute least fixpoints.

Productivity Revisited

$D\uparrow$	$\{false \sqsubseteq true\}$	*true* for productive
$F_p\uparrow$	$\bigwedge$	(*true* for $n_p = 0$)
$\nabla\uparrow$	$\bigvee$	(*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:

$$\boxed{Pr(X) = \bigvee\{\bigwedge_{i=1}^{n_p} Pr(p[i]) \mid p[0] = X\} \text{ for all } X \in V_N} \qquad (Pr)$$

Example: Productivity

Given the following grammar:

$$G = (\{S', S, X, Y, Z\}, \{a, b\}, \left\{ \begin{array}{lcl} S' & \rightarrow & S \\ S & \rightarrow & aX \\ X & \rightarrow & bS \mid aYbY \\ Y & \rightarrow & ba \mid aZ \\ Z & \rightarrow & aZX \end{array} \right\}, S')$$

Resulting system of equations:

$$\begin{array}{lcl} Pr(S) & = & Pr(X) \\ Pr(X) & = & Pr(S) \vee Pr(Y) \\ Pr(Y) & = & true \vee Pr(Z) = true \\ Pr(Z) & = & Pr(Z) \wedge Pr(X) \end{array}$$

Fixpoint iteration

S	X	Y	Z
false	*false*	*false*	*false*

Reachability Revisited

$D\downarrow$	$false \sqsubseteq \{true\}$	*true* for reachable
$F_{p,i}\downarrow$	id	identity mapping
$\nabla\downarrow$	$\bigvee$	Boolean Or (*false*, if there is no occ. of the non-terminal)
I_0	$true$	

Domain: $D\downarrow$ satisfies the conditions,

transfer functions: identity is monotonic,

combination function: disjunction is monotonic.

Resulting system of equations for reachability:

$$\begin{array}{l} Re(S) = true \\ Re(X) = \bigvee\{Re(p[0]) \mid p[i] = X,\ 1 \leq i \leq n_p\}\ \forall X \neq S \end{array} \qquad (Re)$$

Example: Reachability

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

$$\left\{\begin{array}{lcl} S & \rightarrow & Y \\ Y & \rightarrow & YZ \mid Ya \mid b \\ U & \rightarrow & V \\ X & \rightarrow & c \\ V & \rightarrow & Vd \mid d \\ Z & \rightarrow & ZX \end{array}\right\}, S)$$

The equations:

$$\begin{array}{lcl} Re(S) & = & true \\ Re(U) & = & false \\ Re(V) & = & Re(U) \vee Re(V) \\ Re(X) & = & Re(Z) \\ Re(Y) & = & Re(S) \vee Re(Y) \\ Re(Z) & = & Re(Y) \vee Re(Z) \end{array}$$

Fixpoint iteration:

S	U	V	X	Y	Z
true	*false*	*false*	*false*	*false*	*false*

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.

Another Grammar for Arithmetic Expressions

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

$S \rightarrow E$	
$E \rightarrow TE'$	E generates T with a continuation E'
$E' \rightarrow +E \mid \epsilon$	E' generates possibly empty sequence of $+T$s
$T \rightarrow FT'$	T generates F with a continuation T'
$T' \rightarrow *T \mid \epsilon$	T' generates possibly empty sequence of $*F$s
$F \rightarrow \mathbf{id} \mid (E)$	

G_2 defines the same language as G_0 und G_1.

The $FIRST_1$ Sets

- A production $N \rightarrow \alpha$ is applicable for symbols that "begin" α
- 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:

$$FIRST_1(\alpha) = \{a \in V_T | \exists \gamma : \alpha \stackrel{*}{\Longrightarrow} a\gamma\}$$

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:

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

Definitions

Let $k \geq 1$

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

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

k-**concatenation**

$\oplus_k : V^* \times V^* \to V^{\leq k}$, defined by $u \oplus_k v = k : uv$

extended to languages

$k : L = \{k : w \mid w \in L\}$

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

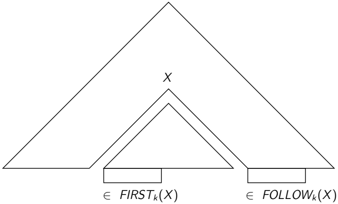
$V^{\leq k} = \bigcup_{i=1}^{k} V^i$ set of words of length at most k ...

$V_{T\#}^{\leq k} = V_T^{\leq k} \cup V_T^{k-1}\{\#\}$ ... possibly terminated by $\#$.

$FIRST_k$ and $FOLLOW_k$

$FIRST_k : (V_N \cup V_T)^* \rightarrow 2^{V_T^{\leq k}}$ where
$FIRST_k(\alpha) = \{k : u \mid \alpha \stackrel{*}{\Longrightarrow} u\}$

set of k–prefixes of terminal words for α .



$FOLLOW_k : V_N \rightarrow 2^{V_{T\#}^{\leq k}}$ where
$FOLLOW_k(X) = \{w \mid S \stackrel{*}{\Longrightarrow} \beta X \gamma \text{ and } w \in FIRST_k(\gamma)\}$

set of k–prefixes of terminal words that may immediately follow X.

GFA-Problem $FIRST_k$

bottom up-GFA-problem $FIRST_k$
L $(2^{V_T^{\leq k}}, \subseteq, \emptyset, \cup)$
T $Fir_p(d_1, \ldots, d_{n_p}) = \{u_0\} \oplus_k d_1 \oplus_k \{u_1\} \oplus_k d_2 \oplus_k \ldots \oplus_k d_{n_p} \oplus_k \{u_{n_p}\}$, if $p = (X_0 \longrightarrow u_0 X_1 u_1 X_2 \ldots X_{n_p} u_{n_p})$; $Fir_p = k : u$ for a terminal production $X \rightarrow u$
C $\cup$

The recursive system of equations for $FIRST_k$ is

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

(Fi_k)

$FIRST_k$ Example

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

$$
\begin{array}{llllllllllll}
0: & S & \rightarrow & E & 3: & E' & \rightarrow & +E & 6: & T' & \rightarrow & *T \\
1: & E & \rightarrow & TE' & 4: & T & \rightarrow & FT' & 7: & F & \rightarrow & (E) \\
2: & E' & \rightarrow & \varepsilon & 5: & T' & \rightarrow & \varepsilon & 8: & F & \rightarrow & \textbf{id}
\end{array}
$$

G_2 defines the same language as G_0 und G_1.
The transfer functions for productions $0 - 8$ are:

$$
\begin{array}{l|l}
Fir_0(d) = d & Fir_3(d) = \{+\} \\
Fir_1(d_1, d_2) = Fir_4(d_1, d_2) = d_1 \oplus_1 d_2 & Fir_6(d) = \{*\} \\
Fir_2 = Fir_5 = \{\varepsilon\} & Fir_7(d) = \{(\} \\
 & Fir_8 = \{\textbf{id}\}
\end{array}
$$

Iteration

Iterative computation of the $FIRST_1$ sets:

S	E	E'	T	T'	F
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

GFA-Problem $FOLLOW_k$

top down-GFA-problem $FOLLOW_k$
L $(2^{V_{T\#}^{\leq k}}, \subseteq, \emptyset, \cup)$
T $Fol_{p,i}(d) = \{u_i\} \oplus_k Fi_k(X_{i+1}) \oplus_k \{u_{i+1}\} \oplus_k \ldots \oplus_k Fi_k(X_{n_p}) \oplus_k \{u_{n_p}\} \oplus_k d$ if $p = (X_0 \longrightarrow u_0 X_1 u_1 X_2 \ldots X_{n_p} u_{n_p})$;
C $\cup$
S $\{\#\}$

The resulting system of equations for $FOLLOW_k$ is

$$Fo_k(X) = \bigcup_{\{p \mid p[i] = X, 1 \leq i \leq n_p\}} Fol_{p,i}(Fo_k(p[0])) \ \forall X \in V_N - \{S\}$$

$$Fo_k(S) = \{\#\}$$

(Fo_k)

$FOLLOW_k$ Example

Regard grammar G_2. The transfer functions are:

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

Iterative computation of the $FOLLOW_1$ sets:

S	E	E'	T	T'	F
$\{\#\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$