Syntax Analysis

Recursive Equations over Grammars

– Wilhelm/Seidl/Hack: Compiler Design, Syntactic and Semantic Analysis–

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

29. Oktober 2013

Properties of a Grammar

Sometimes need to determine properties of (constituents of) a grammar:

- whether the grammar has useless symbols,
- what can start a word for a nonterminal,
- what can follow after a nonterminal.

Properties are expressed as recursive systems of equations.

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.
- We need equivalent definitions that allow (efficient) computation.
- Eliminate non-reachable and non-productive nonterminals from the grammar,
- does not change the described language.

Two-Level Definitions

1. A non-terminal Y is reachable through its occurrence in $X \rightarrow \varphi_1 Y \varphi_2$ iff X is reachable,
2. A non-terminal is reachable iff it is reachable through at least one of its occurrences,
3. S' is reachable.

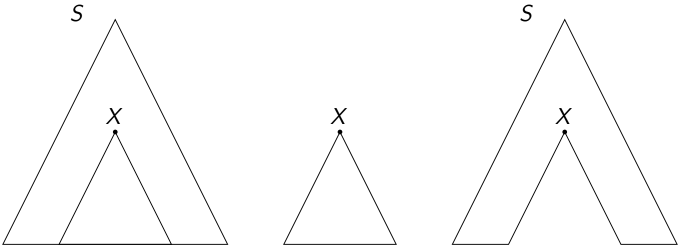
$$\begin{aligned} Re(S') &= true \\ Re(X) &= \bigvee_{Y \rightarrow \varphi_1 X \varphi_2} Re(Y) \quad \forall X \neq S' \end{aligned}$$

1. A non-terminal X is productive through production $X \rightarrow \varphi$ iff all non-terminals occurring in φ are productive.
2. A non-terminal is productive iff it is productive through at least one of its alternatives.

$$Pr(X) = \bigvee_{X \rightarrow \alpha} \bigwedge \{Pr(Y) \mid Y \in V_N \text{ occurs in } \alpha\} \quad \text{for all } X \in V_N$$

- These definitions translate reachability and productivity for a given grammar into (recursive) systems of equations.
- System describes a function $\mathbf{I} : [V_N \to \mathbb{B}] \to [V_N \to \mathbb{B}]$ with $false \sqsubseteq true$
- Iteration starting with smallest element,
  - $Re(S') = true$, $Re(X) = false, \forall X \neq S'$
  - $Pr(X) = false, \forall X \in V_N$
- Least solution wanted to eliminate as many useless non-terminals as possible.

Trees, Subtrees, Tree Fragments



Parse tree

Subtree for X

upper treefragment for X

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

X productive: Set of subtrees for X not empty.

Recursive System of Equations

Questions: Do these recursive systems of equations have

- solutions?
- unique solutions?

They do have solutions if

- the property domain D
  - is partially ordered by some relation $\sqsubseteq$,
  - has a uniquely defined smallest element, $\perp$,
  - has a least upper bound, $d_1 \sqcup d_2$, for each two elements d_1, d_2

  and
- the functions occurring in the equations are monotonic.

Our domains are finite, all functions are monotonic.

Fixed Point Iteration

- Solutions are fixed points 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 $\perp$.
- Evaluate equations until nothing changes.
- Iteration is guaranteed if D has only finitely ascending chains,

We always compute least fixed points.

Example: Productivity

Given the following grammar:

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

Resulting system of equations:

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

Fixed-point iteration

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

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:

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

Fixed-point 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 that can follow a non-terminal).

Use: Removing non-determinism from expand moves of the P_G

Another Grammar for Arithmetic Expressions

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

$$
\begin{array}{lcll}
S & \rightarrow & E & \\
E & \rightarrow & TE' & E \text{ generates } T \text{ with a continuation } E' \\
E' & \rightarrow & +E|\epsilon & E' \text{ generates possibly empty sequence of } +T\text{s} \\
T & \rightarrow & FT' & T \text{ generates } F \text{ with a continuation } T' \\
T' & \rightarrow & *T|\epsilon & T' \text{ generates possibly empty sequence of } *F\text{s} \\
F & \rightarrow & \mathbf{id}|(E) &
\end{array}
$$

G_2 defines the same language as G_0 and G_1.

The $FIRST_1$ Sets

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

$$
\begin{aligned}
S &\rightarrow E \\
E &\rightarrow TE' \\
E' &\rightarrow +E|\epsilon \\
T &\rightarrow FT' \\
T' &\rightarrow *T|\epsilon \\
F &\rightarrow \mathbf{id}|(E)
\end{aligned}
$$

- Example: Arithmetic Expressions, Grammar G_2
  - production $F \rightarrow id$ is applied when current symbol is **id**
  - production $F \rightarrow (E)$ is applied when current symbol is **(**
  - production $T \rightarrow F$ is applied when current symbol is **id** or **(**
- Formal definition:

$$FIRST_1(\alpha) = \{1 : w \mid \alpha \stackrel{*}{\Longrightarrow} w, w \in V_T^*\}$$

The $FOLLOW_1$ Sets

$$
\begin{aligned}
S &\rightarrow E \\
E &\rightarrow TE' \\
E' &\rightarrow +E|\epsilon \\
T &\rightarrow FT' \\
T' &\rightarrow *T|\epsilon \\
F &\rightarrow \textbf{id}|(E)
\end{aligned}
$$

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 $w = a_1 \dots a_n$

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

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 ...

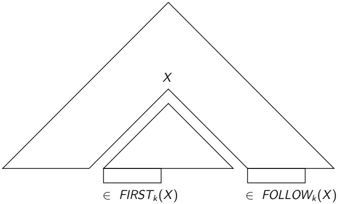
$V_{T\#}^{\leq k} = V_T^{\leq k} \cup V_T^{k-1}\{\#\}$ ... possibly terminated by $\#$.

Properties

Let $k \geq 1$, and $L_1, L_2, L_3 \subseteq V^{\leq k}$.

$$
\begin{array}{ll}
(a) & L_1 \oplus_k (L_2 \oplus_k L_3) = (L_1 \oplus_k L_2) \oplus_k L_3 \\
(b) & L_1 \oplus_k \{\varepsilon\} = \{\varepsilon\} \oplus_k L_1 = k : L_1 \\
(c) & L_1 \oplus_k L_2 = \emptyset \quad \text{iff} \quad L_1 = \emptyset \vee L_2 = \emptyset \\
(d) & \varepsilon \in L_1 \oplus_k L_2 \quad \text{iff} \quad \varepsilon \in L_1 \wedge \varepsilon \in L_2 \\
(e) & k : (L_1 L_2) = k : L_1 \oplus_k k : L_2
\end{array}
$$

$FIRST_k$ and $FOLLOW_k$



$$FIRST_k : (V_N \cup V_T)^* \to 2^{V_T^{\le k}} \text{ where}$$
$$FIRST_k(\alpha) = \{k : u \mid \alpha \xRightarrow{*} u\}$$

set of k–prefixes of terminal words for α

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

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

$FIRST_k$

Theorem

$FIRST_k(Z_1, Z_2, \ldots, Z_n) =$
$FIRST_k(Z_1) \oplus_k FIRST_k(Z_2) \oplus_k \ldots \oplus_k FIRST_k(Z_n)$

The recursive system of equations for $FIRST_k$ is

$$\begin{array}{l} FIRST_k(X) = \bigcup_{\{X \to \alpha\}} FIRST_k(\alpha) \quad \forall X \in V_N \\ FIRST_k(a) = \{a\} \quad \forall a \in V_T \end{array} \qquad (Fi_k)$$

$FIRST_1$ Example

Grammar G_2 below defines the same language as G_0 and G_1.

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

The equations $FIRST_1$ for grammar G_2:

Grammar G_2 below defines the same language as G_0 and G_1

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

The equations $FIRST_1$ for grammar G_2:

$$
\begin{array}{lcl}
FIRST_1(S) & = & FIRST_1(E) \\
FIRST_1(E) & = & FIRST_1(T) \oplus_1 FIRST_1(E') \\
FIRST_1(E') & = & \{\varepsilon\} \cup \{+\} \oplus_1 FIRST_1(E) \\
FIRST_1(T) & = & FIRST_1(F) \oplus_1 FIRST_1(T') \\
FIRST_1(T') & = & \{\varepsilon\} \cup \{*\} \oplus_1 FIRST_1(T) \\
FIRST_1(F) & = & \{\mathrm{Id}\} \cup \{(\} \oplus_1 FIRST_1(E) \oplus_1 \{)\}
\end{array}
$$

Iteration

Iterative computation of the $FIRST_1$ sets:

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

$FOLLOW_k$

The system of equations for $FOLLOW_k$ is

$$FOLLOW_k(X) = \bigcup_{\{Y \to \varphi_1 X \varphi_2\}} FIRST_k(\varphi_2) \oplus_k FOLLOW_k(Y) \ \forall X \in V_N -$$

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

(Fo_k)

$FOLLOW_k$ Example

Regard grammar G_2. The system of equations is:

$$
\begin{array}{lcl}
FOLLOW_1(S) & = & \{\#\} \\
FOLLOW_1(E) & = & FOLLOW_1(S) \cup FOLLOW_1(E') \cup \{)\} \oplus_1 FOLLOW_1(F) \\
FOLLOW_1(E') & = & FOLLOW_1(E) \\
FOLLOW_1(T) & = & \{\varepsilon, +\} \oplus_1 FOLLOW_1(E) \cup FOLLOW_1(T') \\
FOLLOW_1(T') & = & FOLLOW_1(T) \\
FOLLOW_1(F) & = & \{\varepsilon, *\} \oplus_1 FOLLOW_1(T)
\end{array}
$$

Iterative computation of the $FOLLOW_1$ sets:

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