Syntax Analysis

Recursive Equations over Grammars


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 $S \xrightarrow{*} \varphi_1 A \varphi_2$

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

- These definitions are useless for tests; they involve quantifications over infinite sets.
- 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.

$Re(S') = true$

$Re(X) = \bigvee Y \rightarrow \varphi_1 X \varphi_2 Re(Y) \quad \forall X \neq S'$

1. A non-terminal $X$ is productive through production $X \rightarrow \varphi$ iff all non-terminals occurring in $\varphi$ are productive.

2. A non-terminal is productive iff it is productive through at least one of its alternatives.

$Pr(X) = \bigvee X \rightarrow \alpha \bigwedge \{Pr(Y) \mid Y \in V_N \text{ occurs in } \alpha\}$ 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 \( I : [V_N \rightarrow \mathbb{B}] \rightarrow [V_N \rightarrow \mathbb{B}] \) with \( \text{false} \sqsubseteq \text{true} \).

Iteration starting with smallest element,

- \( \text{Re}(S') = \text{true}, \text{Re}(X) = \text{false}, \forall X \neq S' \)
- \( \text{Pr}(X) = \text{false}, \forall X \in V_N \)

Least solution wanted to eliminate as many useless non-terminals as possible.
Trees, Subtrees, Tree Fragments

- **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, $\bot$,
  ▷ 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 $I : [V_N \rightarrow D] \rightarrow [V_N \rightarrow D]$.
- Computed iteratively starting with $\bot$, the function which maps all non-terminals to $\bot$.
- 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:

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

Resulting system of equations:

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

Fixed-point iteration

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{Pr}(S) \text{=} \text{Pr}(X)</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>
Example: Reachability

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

The equations:

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

Fixed-point iteration:

<table>
<thead>
<tr>
<th>S</th>
<th>U</th>
<th>V</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<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 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.

\[
\begin{align*}
S & \rightarrow E \\
E & \rightarrow TE' \quad E \text{ generates } T \text{ with a continuation } E' \\
E' & \rightarrow +E | \epsilon \quad E' \text{ generates possibly empty sequence of } +T \text{s} \\
T & \rightarrow FT' \quad T \text{ generates } F \text{ with a continuation } T' \\
T' & \rightarrow *T | \epsilon \quad T' \text{ generates possibly empty sequence of } *F \text{s} \\
F & \rightarrow \text{id} | (E)
\end{align*}
\]

$G_2$ 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” $\alpha$

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

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

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

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

extended to languages

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

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

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

$\mathcal{V}^{\leq k}_T \# = \mathcal{V}^{\leq k}_T \cup \mathcal{V}^{k-1}_T \{ \# \}$ . . . possibly terminated by $\#$.
Let $k \geq 1$, and $L_1, L_2, L_3 \subseteq V^{\leq k}$.

\begin{align*}
(a) \quad & L_1 \oplus_k (L_2 \oplus_k L_3) = (L_1 \oplus_k L_2) \oplus_k L_3 \\
(b) \quad & L_1 \oplus_k \{\varepsilon\} = \{\varepsilon\} \oplus_k L_1 = k : L_1 \\
(c) \quad & L_1 \oplus_k L_2 = \emptyset \quad \text{iff} \quad L_1 = \emptyset \vee L_2 = \emptyset \\
(d) \quad & \varepsilon \in L_1 \oplus_k L_2 \quad \text{iff} \quad \varepsilon \in L_1 \wedge \varepsilon \in L_2 \\
(e) \quad & k : (L_1 L_2) = k : L_1 \oplus_k k : L_2
\end{align*}
**FIRST**_k_ and **FOLLOW**_k_  

**FIRST**_k_ : \((V_N \cup V_T)^* \rightarrow 2^{V_T^{\leq k}}\) where  
**FIRST**_k_(\(\alpha\)) = \{ \(k : u \mid \alpha \Rightarrow^* u\) \}  
set of \(k\)-prefixes of terminal words for \(\alpha\)  

**FOLLOW**_k_ : \(V_N \rightarrow 2^{V_T^{\leq k} \#}\) where  
**FOLLOW**_k_(\(X\)) = \{ \(w \mid S \Rightarrow^* \beta X \gamma \) and \(w \in \text{FIRST}_k(\gamma)\) \}  
set of \(k\)-prefixes of terminal words that may immediately follow \(X\)
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
\[ FIRST_k(X) = \bigcup_{\{X \to \alpha\}} FIRST_k(\alpha) \quad \forall X \in V_N \]
\[ FIRST_k(a) = \{a\} \quad \forall a \in V_T \quad \text{(Fi}_k) \]
Syntax Analysis

FIRST₁ Example

Grammar $G₂$ below defines the same language as $G₀$ and $G₁$.

\[
\begin{align*}
0: & \quad S \rightarrow E \\
1: & \quad E \rightarrow TE' \\
2: & \quad E' \rightarrow \varepsilon \\
3: & \quad E' \rightarrow +E \\
4: & \quad T \rightarrow FT' \\
5: & \quad T' \rightarrow \varepsilon \\
6: & \quad T' \rightarrow *T \\
7: & \quad F \rightarrow (E) \\
8: & \quad F \rightarrow id
\end{align*}
\]

The equations $FIRST₁$ for grammar $G₂$: 
Grammar $G_2$ below defines the same language as $G_0$ and $G_1$

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

The equations $\text{FIRST}_1$ for grammar $G_2$:

\[
\begin{align*}
\text{FIRST}_1(S) &= \text{FIRST}_1(E) \\
\text{FIRST}_1(E) &= \text{FIRST}_1(T) \oplus_1 \text{FIRST}_1(E') \\
\text{FIRST}_1(E') &= \{\epsilon\} \cup \{+\} \oplus_1 \text{FIRST}_1(E) \\
\text{FIRST}_1(T) &= \text{FIRST}_1(F) \oplus_1 \text{FIRST}_1(T') \\
\text{FIRST}_1(T') &= \{\epsilon\} \cup \{*\} \oplus_1 \text{FIRST}_1(T) \\
\text{FIRST}_1(F) &= \{id\} \cup \{() \oplus_1 \text{FIRST}_1(E) \oplus_1 \{}\}
\end{align*}
\]
## 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>
The system of equations for $\text{FOLLOW}_k$ is

$$\text{FOLLOW}_k(X) = \bigcup \{ \text{FIRST}_k(\varphi_2) \oplus_k \text{FOLLOW}_k(Y) \mid Y \rightarrow \varphi_1X\varphi_2 \} \quad \forall X \in V_N - \text{FOLLOW}_k(S) = \{ \# \}$$

($F_{0k}$)
FOLLOW$_k$ Example

Regard grammar $G_2$. The system of equations is:

\[
\begin{align*}
\text{FOLLOW}_1(S) &= \{\#\} \\
\text{FOLLOW}_1(E) &= \text{FOLLOW}_1(S) \cup \text{FOLLOW}_1(E') \cup \{)\} \oplus_1 \text{FOLLOW}_1(F) \\
\text{FOLLOW}_1(E') &= \text{FOLLOW}_1(E) \\
\text{FOLLOW}_1(T) &= \{\varepsilon, +\} \oplus_1 \text{FOLLOW}_1(E) \cup \text{FOLLOW}_1(T') \\
\text{FOLLOW}_1(T') &= \text{FOLLOW}_1(T) \\
\text{FOLLOW}_1(F) &= \{\varepsilon, *\} \oplus_1 \text{FOLLOW}_1(T)
\end{align*}
\]

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>