Compiler Construction

Lecture 4: Lexical Analysis III (First-Longest-Match Analysis)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH Aachen University

noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/cc10/

Winter semester 2010/11

Outline

The Extended Matching Problem I

Definition

Let $n \geq 1$ and $\alpha_1, \ldots, \alpha_n \in RE_\Omega$ with $\varepsilon \notin [\![\alpha_i]\!] \neq \emptyset$ for every $i \in [n]$ $(= \{1, \ldots, n\})$. Let $\Sigma := \{T_1, \ldots, T_n\}$ be an alphabet of corresponding tokens and $w \in \Omega^+$. If $w_1, \ldots, w_k \in \Omega^+$ such that

- $w = w_1 \ldots w_k$ and
- for every $j \in [k]$ there exists $i_j \in [n]$ such that $w_j \in [\![\alpha_{i_j}]\!]$,

then

- $(w_1, \ldots, w_k)$ is called a decomposition and
- $(T_{i_1}, \ldots, T_{i_k})$ is called an analysis

of w w.r.t. $\alpha_1, \ldots, \alpha_n$.

Problem (Extended matching problem)

Given $\alpha_1, \ldots, \alpha_n \in RE_\Omega$ and $w \in \Omega^+$, decide whether there exists a decomposition of w w.r.t. $\alpha_1, \ldots, \alpha_n$ and determine a corresponding analysis.

The Extended Matching Problem II

Observation: neither the decomposition nor the analysis are uniquely determined

Example

1. $\alpha = a^+$, $w = aa$
   $\implies$ two decompositions (aa) and (a, a) with unique analysis each
2. $\alpha_1 = a \mid b$, $\alpha_2 = a \mid c$, $w = a$
   $\implies$ unique decomposition (a) but two analyses (T_1) and (T_2)

Goal: make both unique $\implies$ deterministic scanning

Outline

Ensuring Uniqueness

Two principles:

1. Principle of the longest match ("maximal munch tokenization")
   - for uniqueness of decomposition
   - make lexemes as long as possible
   - motivated by applications: e.g., every (non-empty) prefix of an identifier is also an identifier
2. Principle of the first match
   - for uniqueness of analysis
   - choose first matching regular expression (in the given order)
   - therefore: arrange keywords before identifiers (if keywords protected)

Principle of the Longest Match

Definition 4.1 (Longest-match decomposition)

A decomposition $(w_1, \ldots, w_k)$ of $w \in \Omega^+$ w.r.t. $\alpha_1, \ldots, \alpha_n \in RE_\Omega$ is called a longest-match decomposition (LM decomposition) if, for every $i \in [k]$, $x \in \Omega^+$, and $y \in \Omega^*$,

$$w = w_1 \ldots w_i x y \implies \text{there is no } j \in [n] \text{ such that } w_i x \in [\![\alpha_j]\!]$$

Corollary 4.2

Given w and $\alpha_1, \ldots, \alpha_n$,

- *at most one LM decomposition of w exists (clear by definition) and*
- *it is possible that w has a decomposition but no LM decomposition (see following example).*

Example 4.3

$w = aab$, $\alpha_1 = a^+$, $\alpha_2 = ab$
$\implies$ (a, ab) is a decomposition but no LM decomposition exists

Principle of the First Match

Problem: a (unique) LM decomposition can have several associated analyses (since $[\![\alpha_i]\!] \cap [\![\alpha_j]\!] \neq \emptyset$ with $i \neq j$ is possible; cf. keyword/identifier problem)

Definition 4.4 (First-longest-match analysis)

Let $(w_1, \ldots, w_k)$ be the LM decomposition of $w \in \Omega^+$ w.r.t. $\alpha_1, \ldots, \alpha_n \in RE_\Omega$. Its first-longest-match analysis (FLM analysis) $(T_{i_1}, \ldots, T_{i_k})$ is determined by

$$i_j := \min\{l \in [n] \mid w_j \in [\![\alpha_l]\!]\} \text{ for every } j \in [k].$$

Corollary 4.5

Given w and $\alpha_1, \ldots, \alpha_n$, there is at most one FLM analysis of w. It exists iff the LM decomposition of w exists.

Outline

Algorithm 4.6 (FLM analysis—overview)

Input: *expressions* $\alpha_1, \ldots, \alpha_n \in RE_\Omega$, *tokens* $\{T_1, \ldots, T_n\}$, *input word* $w \in \Omega^+$

Procedure:

1. *for every* $i \in [n]$, *construct* $\mathfrak{A}_i \in DFA_\Omega$ *such that* $L(\mathfrak{A}_i) = [\![\alpha_i]\!]$ *(see DFA method; Alg. **??**)*
2. *construct the product automaton* $\mathfrak{A} \in DFA_\Omega$ *such that* $L(\mathfrak{A}) = \bigcup_{i=1}^{n} [\![\alpha_i]\!]$
3. *partition the set of final states of* $\mathfrak{A}$ *to follow the first-match principle*
4. *extend the resulting DFA to a backtracking DFA which implements the longest-match principle, and let it run on* w

Output: *FLM analysis of* w *(if existing)*

(2) The Product Automaton

Definition 4.7 (Product automaton)

Let $\mathfrak{A}_i = \langle Q_i, \Omega, \delta_i, q_0^{(i)}, F_i \rangle \in DFA_\Omega$ for every $i \in [n]$. The product automaton $\mathfrak{A} = \langle Q, \Omega, \delta, q_0, F \rangle \in DFA_\Omega$ is defined by

- $Q := Q_1 \times \ldots \times Q_n$
- $q_0 := (q_0^{(1)}, \ldots, q_0^{(n)})$
- $\delta((q^{(1)}, \ldots, q^{(n)}), a) := (\delta_1(q^{(1)}, a), \ldots, \delta_n(q^{(n)}, a))$
- $(q^{(1)}, \ldots, q^{(n)}) \in F$ iff there ex. $i \in [n]$ such that $q^{(i)} \in F_i$

Lemma 4.8

The above construction yields $L(\mathfrak{A}) = \bigcup_{i=1}^{n} L(\mathfrak{A}_i)$ $(= \bigcup_{i=1}^{n} [\![\alpha_i]\!])$.

Remark: similar construction for intersection $(F := F_1 \times \ldots \times F_n)$

(3) Partitioning the Final States

Definition 4.9 (Partition of final states)

Let $\mathfrak{A} = \langle Q, \Omega, \delta, q_0, F \rangle \in DFA_\Omega$ be the product automaton as constructed before. Its set of final states is partitioned into $F = \biguplus_{i=1}^{n} F^{(i)}$ by the requirement

$(q^{(1)}, \ldots, q^{(n)}) \in F^{(i)}$ iff $q^{(i)} \in F_i$ and $\forall j \in [i-1] : q^{(j)} \notin F_j$

(or: $F^{(i)} := (Q_1 \setminus F_1) \times \ldots \times (Q_{i-1} \setminus F_{i-1}) \times F_i \times Q_{i+1} \times \ldots \times Q_n$)

Corollary 4.10

The above construction yields ($w \in \Omega^+$, $i \in [n]$):

$$\hat{\delta}(q_0, w) \in F^{(i)} \text{ iff } w \in [\![\alpha_i]\!] \text{ and } w \notin \bigcup_{j=1}^{i-1} [\![\alpha_j]\!].$$

Definition 4.11 (Productive states)

Given $\mathfrak{A}$ as above, a state $q \in Q$ is called productive if there exists $w \in \Omega^*$ such that $\hat{\delta}(q, w) \in F$. The set of productive states of $\mathfrak{A}$ is denoted by P (and thus $F \subseteq P$).

(4) The Backtracking DFA I

Goal: extend $\mathfrak{A}$ to the backtracking DFA $\mathfrak{B}$ with output by equipping the input tape with two pointers: a backtracking head for marking the last encountered match, and a lookahead for determining the longest match.

A configuration of $\mathfrak{B}$ has three components (remember: $\Sigma := \{T_1, \ldots, T_n\}$ denotes the set of tokens):

1. a mode $m \in \{N\} \uplus \Sigma$:
   - $m = N$ ("normal"): look for first match (no final state reached yet)
   - $m = T \in \Sigma$: token T has been recognized, looking for possible longer match
2. an input tape $vqw \in \Omega^* \cdot Q \cdot \Omega^*$:
   - v: lookahead part of input ($v \neq \varepsilon \implies m \in \Sigma$)
   - q: current state of $\mathfrak{A}$
   - w: remaining input
3. an output tape $W \in \Sigma^* \cdot \{\varepsilon, \mathsf{lexerr}\}$:
   - Σ^*: sequence of tokens recognized so far
   - lexerr: a lexical error has occurred (i.e., a non-productive state was entered or the suffix of the input is not a valid lexeme)

(4) The Backtracking DFA II

Definition 4.12 (Backtracking DFA)

- The set of configurations of $\mathfrak{B}$ is given by
$$(\{N\} \uplus \Sigma) \times \Omega^* \cdot Q \cdot \Omega^* \times \Sigma^* \cdot \{\varepsilon, \mathsf{lexerr}\}$$
- The initial configuration for an input word $w \in \Omega^+$ is $(N, q_0 w, \varepsilon)$.
- The transitions of $\mathfrak{B}$ are defined as follows (where $q' := \delta(q, a)$):
  - normal mode: look for a match
$$(N, qaw, W) \vdash \begin{cases} (T_i, q'w, W) & \text{if } q' \in F^{(i)} \\ (N, q'w, W) & \text{if } q' \in P \setminus F \\ \textbf{output: } W \cdot \mathsf{lexerr} & \text{if } q' \notin P \end{cases}$$
  - backtrack mode: look for longest match
$$(T, vqaw, W) \vdash \begin{cases} (T_i, q'w, W) & \text{if } q' \in F^{(i)} \\ (T, vaq'w, W) & \text{if } q' \in P \setminus F \\ (N, q_0 vaw, WT) & \text{if } q' \notin P \end{cases}$$
  - end of input
$$\begin{aligned} (T, q, W) &\vdash \textbf{output: } WT && \text{if } q \in F \\ (N, q, W) &\vdash \textbf{output: } W \cdot \mathsf{lexerr} && \text{if } q \in P \setminus F \\ (T, vaq, W) &\vdash (N, q_0 va, WT) && \text{if } q \in P \setminus F \end{aligned}$$

(4) The Backtracking DFA III

Lemma 4.13

Given the backtracking DFA $\mathfrak{B}$ as before and $w \in \Omega^+$,

$$(N, q_0 w, \varepsilon) \quad \vdash^* \quad \begin{cases} W \in \Sigma^* & \textit{iff } W \textit{ is the FLM analysis of } w \\ W \cdot \textsf{lexerr} & \textit{iff no FLM analysis of } w \textit{ exists} \end{cases}$$

Example 4.14

$\alpha = (ab)^+$, $w = abaa$ (on the board)

Remarks:

- Time complexity: $\mathcal{O}(|w|^2)$ in worst case

Example 4.15

$\alpha_1 = a$, $\alpha_2 = a^*b$, $w = a^m$ requires $\mathcal{O}(m^2)$

- Improvement by tabular method (similar to Knuth-Morris-Pratt Algorithm for pattern matching in strings)

  Literature: Th. Reps: *"Maximal-Munch" Tokenization in Linear Time*, ACM TOPLAS 20(2), 1998, 259–273