第四讲　自顶向下语法分析

2016-09-28

1. 基本思想

对于给定语言的文法和一个单词符号串（终结符串），一般的自顶向下分析过程是：从文法开始符号进行推导，每一步推导都获得文法的一个句型，直到产生一个句子，恰好是所期望的单词符号串。每一步推导是对当前句型中剩余的某个非终结符进行展开，即使用以该非终结符为左部的某个产生式的右部替换该非终结符。如果不存在这样的推导，则表明该单词符号串存在语法错误。

例如，给定如下文法 $G[S]$：

$$S \to AB$$
$$A \to aA \mid \varepsilon$$
$$B \to b \mid bB$$

针对单词符号串 $aaab$ 的一个自顶向下分析过程为：

S // 使用产生式 $S \to AB$
$\Rightarrow AB$ // 使用产生式 $A \to aA$
$\Rightarrow aAB$ // 使用产生式 $B \to b$
$\Rightarrow aAb$ // 使用产生式 $A \to aA$
$\Rightarrow aaAb$ // 使用产生式 $A \to aA$
$\Rightarrow aaaAb$ // 使用产生式 $A \to \varepsilon$
$\Rightarrow aaab$

2. 带回溯的自顶向下分析

一般的自顶向下分析过程存在两类非确定性。其一是在每一步推导中，选择对哪一个非终结符进行展开。其二是如果选定的非终结符是多个产生式的左部，那么应该选择使用哪一个产生式。这种非确定性导致推导过程需要不断地进行试探和回溯。例如，对于文法 $G[S]$ 单词符号串 $aaab$，以下是一个试探的推导过程：

S // 使用产生式 $S \to AB$
$\Rightarrow AB$ // 使用产生式 $A \to aA$
$\Rightarrow aAB$ // 使用产生式 $B \to bB$
$\Rightarrow aAbB$ // 使用产生式 $A \to aA$
$\Rightarrow aaAbB$ // 使用产生式 $A \to aA$
$\Rightarrow aaaAbB$ // 使用产生式 $A \to \varepsilon$
$\Rightarrow aaabB$ // 必须回溯

回溯会带来很高的复杂度，这在编译程序的设计中是不现实的。试想一个实际中需要编

译的程序单位可能会包含多少个单词符号，就会得出这样的结论。那么在自顶向下分析过程中，如何避免回溯呢？解决办法就是消除上述两类非确定性。

很容易想到，如果我们只允许最左推导或最右推导，那么就可以避开上述的第一类非确定性。由于通常都是从左到右读入单词符号串，所以我们可以规定在自顶向下分析过程只使用最左推导。例如，对于文法 $G[S]$ 单词符号串 $aaab$，以下试探推导过程的每一步总是对最左边的非终结符进行展开：

S // 使用产生式 $S \to AB$
$\Rightarrow AB$ // 使用产生式 $A \to aA$
$\Rightarrow aAB$ // 使用产生式 $A \to aA$
$\Rightarrow aaAB$ // 使用产生式 $A \to \varepsilon$
$\Rightarrow aaB$ // 必须回溯

这个试探推导过程虽然出现了必须回溯的情形，但我们能够确信这是由于第二类非确定性造成的。也就是说，在前面的推导步骤中选错了产生式。

3. 自顶向下预测分析

如果在自顶向下分析过程中，我们能够同时消除两类非确定性，即能够保证非终结符选择和产生式选择都是确定的，那么这样的分析过程就是**确定的自顶向下分析**。

为了解决消除第二类非确定性的问题，通常采取的策略是向前查看确定数目的单词符号，然后确定应该选择哪一个产生式进行最左推导。这种方法称为**自顶向下预测分析**。成功分析的结果是一个唯一的最左推导。

我们先来看一个可以进行自顶向下预测分析的例子。

对于本节前面所讨论的文法 $G[S]$，以及任意的单词符号序列 $a^n b^m$（$n \geq 0, m > 0$），总存在如下的最左推导：

S // 使用产生式 $S \to AB$
$\Rightarrow AB$ // 使用产生式 $A \to aA$
$\Rightarrow aAB$ // 使用产生式 $A \to aA$
……
$\Rightarrow a^n AB$ // 使用产生式 $A \to \varepsilon$
$\Rightarrow a^n B$ // 使用产生式 $B \to bB$
$\Rightarrow a^n bB$ // 使用产生式 $B \to bB$
……
$\Rightarrow a^n b^{m-1} B$ // 使用产生式 $B \to b$
$\Rightarrow a^n b^m$

不难看出，对于给定的 $a^n b^m$ 来说，这是唯一的最左推导。然而，问题是当最左的非终结符为 A 时，我们如何在产生式 $A \to aA$ 和 $A \to \varepsilon$ 之间做出选择；同样，当最左的非终结符为 B 时，我们如何在产生式 $B \to bB$ 和 $B \to b$ 之间做出选择。

我们来分析一下向前查看确定数目的单词符号进行预测分析的可能性。首先，在需要展开的最左非终结符为 A 时，我们通过向前查看 1 个单词符号，就可以在产生式 $A \to aA$ 和 $A \to \varepsilon$ 之间作出选择。当查看到下一个单词符号为 a 时，选择 $A \to aA$；而当查看到下一

个单词符号为 b 时，则选择 $A \to \varepsilon$。然而，在需要展开的最左非终结符为 B 时，当查看到下一个单词符号为 b 时，并不足以在 $B \to bB$ 和 $B \to b$ 之间做出选择。我们将向前查看单词符号的数目变为 2。通过分析，当后两个单词符号为 bb 时，我们应该选择 $B \to bB$；而当后两个单词符号中只有一个 b 时（后跟一个单词符号序列的结束符号），应该选择 $B \to b$。

由于 $L(G[S]) = \{ a^n b^m \mid n \geq 0, m > 0 \}$，所以我们可以得出结论：只要向前查看 2 个单词符号，就可预测分析 $L(G[S])$ 中的所有句子。

我们先来看一个不能进行自顶向下预测分析的例子。设有如下文法 $G'[S]$：

$$S \to Sa$$
$$S \to b$$

对于 $L(G[S])$ 中 的任意的单词符号序列 ba^n （$n \geq 0$），总存在如下的最左推导：

S // 使用产生式 $S \to Sa$
$\Rightarrow Sa$ // 使用产生式 $S \to Sa$
$\Rightarrow Saa$ // 使用产生式 $S \to Sa$
……
$\Rightarrow Sa^n$ // 使用产生式 $S \to b$
$\Rightarrow ba^n$

然而，在第 1 步推导时，在向前查看到第 2 个单词符号为 a 时，我们才会选择产生式 $S \to Sa$；在第 2 步推导时，在向前查看到第 3 个单词符号为 a 时，我们才会选择产生式 $S \to Sa$；这样，在第 n 步推导时，我们就需要向前查看 $n+1$ 个单词。这样，无论向前查看单词符号的数确定为多少，都无法满足对 $L(G)$ 中所有句子进行预测分析的需求。

可见，不是所有文法都是可以成功实施自顶向下预测分析的。在实践中，通常我们可以对文法的设计进行适当限制，以满足预测分析的要求。比如 我们将在下一节讨论的 LL(1) 文法是一种满足这种要求的文法，并且足以满足多数程序设计语言的文法描述需求。

4. LL（1） 分析

LL(1) 分析是应用较普遍的一种自顶向下预测分析方法。

LL(1) 中的第一个 “L” 代表从左（Left）向右扫描单词符号，第二个 “L” 代表产生的是最左（Leftmost）推导，“1” 代表向前查看（lookahead）一个单词符号。

LL(1) 分析方法仅适用于 LL(1) 文法。为给出 LL(1) 文法的定义，我们首先介绍两个重要概念：First 集合和 Follow 集合。这些概念在介绍自底向上分析方法时也要用到。

4.1 First 集合和 Follow 集合

我们先来看 **First 集合**的定义：

设上下文无关文法 $G = (V_N, V_T, P, S)$。对 $\alpha \in (V_N \cup V_T)^*$，

$$\text{First}(\alpha) = \{ a \mid \alpha \Rightarrow^* a\beta, a \in V_T, \beta \in (V_N \cup V_T)^* , \text{或者 } \alpha \Rightarrow^* \varepsilon \text{ 时 } a = \varepsilon \}$$

直观来理解，一个句型 α 若可以推导出另一个以终结符 a 开头的句型，那么 a 属于

First（α）；若 α 可以推导出 ε，那么 ε 属于 First（α）。

由于任何推导都可以对应一个最左推导（可以参考2.2.5的结果直接得出该结论），所以 First 集合的定义也可以基于最左推导给出：

$$\text{First}(\alpha) = \{ a \mid \alpha \Rightarrow_{lm}^* a\beta, a \in V_T, \beta \in (V_N \cup V_T)^* \text{，或者 } \alpha \Rightarrow_{lm}^* \varepsilon \text{ 时 } a = \varepsilon\}$$

在实际应用中，对于给定文法 $G = (V_N, V_T, P, S)$，我们通常会关心下列集合 X_G 中句型的 First 集合：

$$X_G = V_N \cup V_T \cup \{\varepsilon\} \cup \{v \mid A \rightarrow u \in P, \text{且 } v \text{ 是 } u \text{ 的后缀}\}$$

对所有 $x \in X_G$，可以通过如下过程计算 First（x）：

- 初始时，对 $x \in V_T \cup \{\varepsilon\}$，置 First ($x$) = {$x$}；对其它 x，置 First (x) = Φ ；
- 重复如下步骤，直到所有 First 集合没有变化为止：

  (1) 对于 $y_1y_2...y_k \in \{v \mid A \rightarrow u \in P, \text{且 } v \text{ 是 } u \text{ 的后缀}\}$，其中 $k \geq 1$，$y_j \in V_N \cup V_T$（$1 \leq j \leq k$），若 $\forall j{:}1 \leq j \leq i\text{-}1\ (\varepsilon \in \text{First}(y_j)) \wedge \varepsilon \notin \text{First}(y_i)$，其中 $1 \leq i \leq k$ ，则令

  $$\text{First}(y_1y_2...y_k) = \text{First}(y_1) \cup \text{First}(y_2) \cup ... \cup \text{First}(y_i) - \{\varepsilon\}$$

  否则，若 $\forall j{:}1 \leq j \leq k\ (\varepsilon \in \text{First}(y_j))$，则令

  $$\text{First}(y_1y_2...y_k) = \text{First}(y_1) \cup \text{First}(y_2) \cup ... \cup \text{First}(y_k)$$

  (2) 若有 $A \rightarrow y_1y_2...y_k \in P$， 则置 First ($A$) = First($A$) $\cup$ First ($y_1y_2...y_k$) 。

例 1 设上下文无关文法 $G = (V_N, V_T, P, S)$，其中 P 为

$S \rightarrow AB$
$A \rightarrow Da \mid \varepsilon$
$B \rightarrow cC$
$C \rightarrow aADC \mid \varepsilon$
$D \rightarrow b \mid \varepsilon$

试计算下列集合 X_G 中句型的 First 集合：

$$X_G = V_N \cup V_T \cup \{\varepsilon\} \cup \{v \mid A \rightarrow u \in P, \text{且 } v \text{ 是 } u \text{ 的后缀}\}$$

解 集合 X_G 为：

$$X_G = \{\varepsilon, S, A, B, C, D, a, b, c, AB, Da, cC, aADC, ADC, DC\}$$

初始时，置 First (a) = {a}，First (b) = {b}，First (c) = {c}，First (ε) = {ε}，而对其它的 $x \in X_G$，置 First (x) = Φ 。

第一轮应用以上三个步骤之后，X_G 中句型的 First 集合更新情况为：

First (cC) = {c}
First ($aADC$) = {a}
First (A) = {ε }
First (B) = {c}
First (C) = {ε, a}