ソフトウェア基礎論　16－2
プログラミング言語　4－2
（1986. 2. 7）

L-attributed LL(1) grammars are LR-attributed

Ikuo Nakata and Masataka Sassa

University of Tsukuba

1. Introduction

Attribute grammars are an extension of context-free grammars which unify syntax and semantics of programming languages. This paper concerns classes of attribute grammars for which attributes can be evaluated in a single pass during parsing without making a syntax tree. They are becoming attractive due to their efficiency and practicality and the fact that most modern programming languages are now designed around the easier one-pass processing techniques.

A class of attribute grammars called L-attributed grammars which can be easily combined with LL(1)-parsers, or recursive descent parsers, is well known. However, another class of attribute grammars called LR-attributed grammars [4, 6] has not been well known because of the difficulty of their definition and the restriction posed on their inherited attributes. LR-attributed grammars can be combined with LR-parsers which are more powerful than LL-parsers, but has been considered less powerful as tools for semantic analysis.

In this paper we will show that LR-attributed LR(1)-grammars are more powerful than L-attributed LL(1)-grammars by proving that L-attributed LL(1)-grammars are LR-attributed. In proving the theorem we use several lemmas which can also be used to prove the well known theorem that an LL(1)-grammar is an LR(1)-grammar [1].

2. Notation

In the following, upper-case letters such as A, B, C, ... are used as nonterminals; lower-case letters such as a, b, c, ... as terminals; X, Y, ... as grammar symbols (either nonterminals or terminals); α, β, γ, ... as strings of grammar symbols.

An attribute a of symbol X is represented by X.a. The set of inherited and synthesized attributes of symbol X are represented by AI(X) and AS(X), respectively.

Definition 1 For the production $X_0 \rightarrow X_1 \ldots X_n$ (where X_0 is a nonterminal), $AI(X_0)$ and $AS(X_j)$ (j=1, ... ,n) are called ***input attribute occurrences***.

As in much of the literature, we assume the following.

Assumption 1 Only input attribute occurrences appear in the right side of a semantic rule of a given grammar (often called ***Bochmann normal form***).

L-attributed grammars are defined as follows under Assumption 1.

Definition 2 An attribute grammar is *L-attributed* iff for any production $X_0 \to X_1 \dots X_n$ the following conditions hold:

(1) The attribute occurrences in $AI(X_k)$ $(1 \le k \le n)$ depend only on the values of attribute occurrences in

$$AI(X_0) \cup \cup_{i=1}^{k-1} AS(X_i).$$

(2) The attribute occurrences in $AS(X_0)$ depend only on the values of attribute occurrences in

$$AI(X_0) \cup \cup_{i=1}^{n} AS(X_i).$$

For a given grammar, LR states can be constructed as usual. We assume that the start symbol of the grammar is Z and the grammar is augmented by the production "$Z' \to Z$".

An *LR(1) item* (*LR item* or *item* for short) of a grammar G is $[A \to \alpha \cdot \beta, a]$ where "$A \to \alpha\beta$" is a production of G, and a is a terminal. Define a relation $\downarrow$ on items by

$$i \downarrow j \text{ iff } \exists B\text{: } i=[A \to \alpha \cdot B\beta, a] \text{ and } j=[B \to \cdot \gamma, b]$$

$$\text{where } b \in First(\beta a)=\{c \mid \beta a \Rightarrow^* c\delta\}.$$

The closure set of an item set is given by the reflexive transitive closure $\downarrow^*$. For an item set R

$$Closure(R)=\{j \mid \exists i \in R\text{: } i \downarrow^* j\}.$$

An LR automaton is given by putting

$$S_0=Closure(\{[Z' \to \cdot Z, \$]\})$$

$$Next(S,X)=Closure(Succ(S,X))$$

where

$$Succ(S,X)=\{[A \to \alpha X \cdot \beta, a] \mid \exists [A \to \alpha \cdot X\beta, a] \in S\}$$

These item sets such as S_0 or Next(S,X) are called states of the LR automaton or LR states. The kernel and the nonkernel of a state is defined as

$$Kernel(S_0)=\{[Z' \to \cdot Z, \$]\}$$

$$Kernel(Next(S,X))=Succ(S,X)$$

$$Nonkernel(S)=S - Kernel(S)$$

As in [5, 6], we subdivide an LR state into (LR-) partial states according to lookahead terminals.

Definition 3 The partial state of an LR state S with lookahead a, PS(S,a), is defined as

$$PS(S,a)=\{i \mid i=[A \to \alpha \cdot \beta, b] \in S;\ First(\beta b) \ni a\}$$

Next, we define a set of inherited attributes, IN(PS), which should be evaluated at partial state PS as follows:

Definition 4

$$IN(PS)=\{B.b \mid \exists [A \to \alpha \cdot B\beta, a] \in PS\text{: } B.b \in AI(B)\}$$

3. LR-attributed grammars

An LR-attributed grammar is defined as follows:

Definition 5 A grammar G is said to be *LR-attributed* iff

(1) G is L-attributed

(2) For any partial state PS of the LR automaton for G, and for any inherited attribute $A.a \in IN(PS)$, the evaluation rule of A.a can be uniquely determined.

Note: More concrete definition of LR-attribute grammars than the above one is given in [6]. However, the above definition is sufficient for the present purpose.

Our theorem can be stated as follows:

Theorem 1 Attribute grammars which are LL(1) and L-attributed are LR-attributed.

To prove the theorem, we notify that for any production $A \to \alpha B \beta$, the semantic rule for any $B.b \in AI(B)$ is unique (by the definition of attribute grammars). Therefore, it suffices if the following can be proved for any partial state of an L-attributed LL(1) grammar:

For any partial state PS and for any nonterminal B, there exists at most one item of the form

$[A \to \alpha \cdot B\beta,\ a]$

in PS.

Each LR state is constructed by repeating the operations Succ and Closure. We will prove the above property along the sequence of these operations after proving the following lemmas.

Lemma 1 If $PS(S,a) \ni i$ then there exists $j \in Kernel(S)$ such that $j \downarrow^* i$ and $j \in PS(S,a)$

Proof: For any $i \in S$ there exists $j \in Kernel(S)$ such that $j \downarrow^* i$ from the construction method of LR states. Let $j = [A \to \alpha \cdot B\beta,\ b]$. If $j = i$, the lemma holds trivially. If $j \neq i$ then $j \downarrow^+ i$. Therefore, i can be written as $[C \to \cdot \gamma,\ d]$ and $First(B\beta b) \supset First(\gamma d)$. $PS(S,a) \ni i$ means $First(\gamma d) \ni a$, therefore, $First(B\beta b) \ni a$, thus $j \in PS(S,a)$. □

Lemma 2 If, G is an LL(1) grammar, and for an LR state S of G, $PS(S,a) \cap Kernel(S)$ has at most one element for any terminal a, then

(1) if there are two items i_1, i_2 in a partial state PS of S, then $i_1 \downarrow^* i_2$ or $i_2 \downarrow^* i_1$

(2) there is at most one item in PS of the form $[C \to \gamma \cdot X\delta,\ b]$ (for some C, γ, δ, ignoring the difference of b) for any partial state PS of S and any grammar symbol X

(3) for any grammar symbol X, if $i_1, i_2 \in S$, $i_1 = [C_1 \to \gamma_1 \cdot X\delta_1,\ b_1]$, $i_2 = [C_2 \to \gamma_2 \cdot X\delta_2,\ b_2]$ and $i_1 \neq i_2$ (i.e. $C_1 \neq C_2$ or $\gamma_1 \neq \gamma_2$ or $\delta_1 \neq \delta_2$) then X is a nonterminal, X generates only ε and $First(\delta_1 b_1) \cap First(\delta_2 b_2) = \emptyset$.

Proof: Let us take a partial state $PS(S,a_0)$ of S. From lemma 1, there exists an item i such that $i \in Kernel(S) \cap PS(S,a_0)$. This i is the unique element of $Kernel(S) \cap PS(S,a_0)$ from the given condition. Therefore, for any $j \in PS(S,a_0)$ $i \downarrow^* j$ holds. Let $i = [A \to \alpha \cdot B\beta,\ a]$.

We will prove (1) first. Let $i_1, i_2 \in PS(S,a_0)$. If one of them is equal to i then (1) holds. Therefore, we can assume that $i \neq i_1$, $i \neq i_2$ and $i_1, i_2 \in Nonkernel(S)$. Let $i_1 = [C_1 \to \cdot \delta_1,\ b_1]$, $i_2 = [C_2 \to \cdot \delta_2,$

$b_2]$ ($C_1 \neq C_2$ or $\delta_1 \neq \delta_2$). From lemma 1 $i\downarrow^* i_1$ and $i\downarrow^* i_2$ hold. If neither $i_1\downarrow^* i_2$ nor $i_2\downarrow^* i_1$ holds then there exist two derivation sequences

$$B \Rightarrow \cdots \Rightarrow C_1 \ldots \Rightarrow \delta_1 \ldots$$

$$B \Rightarrow \cdots \Rightarrow C_2 \ldots \Rightarrow \delta_2 \ldots$$

and none of them is a subsequence of the other. Therefore, there exist two productions and two derivation sequences

$$C_0 \to \eta_1,\ C_0 \to \eta_2\ (\eta_1 \neq \eta_2)$$

$$B \Rightarrow^* C_0 \ldots \Rightarrow \eta_i \ldots \Rightarrow^* C_i \ldots \Rightarrow \delta_i \ldots\quad (i=1,2;\ C_0 \text{ may be equal to } C_1 \text{ and } C_2).$$

These sequences can be imbedded in sequences

$$Z \Rightarrow^* \alpha_0 A a \ldots \Rightarrow \alpha_1 B \beta a \ldots \Rightarrow^* \alpha_1 C_0 \ldots \Rightarrow \alpha_1 \eta_i \ldots \Rightarrow^* \alpha_1 C_i b_i \ldots \Rightarrow \alpha_1 \delta_i b_i \ldots\quad (\alpha_1 = \alpha_0 \alpha).$$

This means C_0 can be expanded to either η_1 or η_2 during the top down parsing by lookahead symbol $a_0 \in First(\delta_i b_i)$. This contradicts the LL(1) condition [1]. This completes the proof of (1).

We will prove (2) next. Let $j=[C \to \gamma \cdot X\delta,\ b] \in PS(S,a_0)$. (i) Let $X=B$. If there exist two such j's, at least one of them is in Nonkernel(S) since $Kernel(S) \cap PS(S,a_0)$ has only one element. Thus we assume that $j \notin Kernel(S)$, therefore, $\gamma=\varepsilon$. It means there exists a derivation

$$B \Rightarrow^* C \ldots \Rightarrow B\delta \ldots$$

because $i\downarrow^* j$. This means the existence of a left recursion which contradicts the assumption that the given grammar is an LL(1) grammar [1]. (ii) Let $X \neq B$. If there exist two such j's, they should be in Nonkernel(S), namely

$$j_1=[C_1 \to \cdot X\delta_1,\ b_1],\ j_2=[C_2 \to \cdot X\delta_2,\ b_2] \in PS(S,a_0),\ C_1 \neq C_2 \text{ or } \delta_1 \neq \delta_2$$

then we can assume $j_1\downarrow^* j_2$ by (1). Therefore, there exists a derivation

$$C_1 \Rightarrow X\delta_1 \Rightarrow^* C_2 \ldots \Rightarrow X\delta_2 \ldots$$

which shows a left recursion. This contradicts the LL(1) condition.

Now, we can prove (3). If there exist $i_1, i_2 \in S$ such that $i_j=[C_j \to \gamma_j \cdot X\delta_j,\ b_j]$ and $i_1 \neq i_2$ then they do not belong to the same PS by (2). If there exists a terminal $a \in First(X)$ then both i_1 and i_2 belong to $PS(S,a)$. This contradicts the above. Therefore, X generates only ε, and X is a nonterminal. Since i_1 and i_2 do not belong to the same PS $First(X\delta_1 b_1) \cap First(X\delta_2 b_2)=\emptyset$. Therefore, $First(\delta_1 b_1) \cap First(\delta_2 b_2)=\emptyset$ because $First(\delta_i b_i)=First(X\delta_i b_i)$. □

Lemma 3 If G is an LL(1) grammar, for any partial state PS of any LR state S, $PS \cap Kernel(S)$ has only one element

Proof: The initial state $S_0=Closure(\{[Z' \to \cdot Z,\ \$]\})$ has only one item in its kernel, therefore, it satisfies the condition of lemma 3. Assume that an LR state S satisfies the condition. Then we can show that $S'=Closure(Succ(S,X))$ also satisfies the condition for any X as follows. If S' has only one item in its kernel, S' satisfies the condition. If there exist two items j_1, j_2 in Kernel(S'), then S has two items i_1, i_2 such that $i_1=[C_1 \to \gamma_1 \cdot X\delta_1,\ b_1]$, $i_2=[C_2 \to \gamma_2 \cdot X\delta_2,\ b_2]$, $i_1 \neq i_2$ and $j_1=[C_1 \to \gamma_1 X \cdot \delta_1,\ b_1]$,

$j_2=[C_2 \to \gamma_2 X \cdot \delta_2, b_2]$. Since the condition of lemma 2 holds for S, $First(\delta_1 b_1) \cap First(\delta_2 b_2) = \emptyset$ by (3) of lemma 2. This means that every item in the kernel of S' belongs to different PS. □

Proof of theorem 1: By lemma 3 any LR state S of G satisfies the condition of lemma 2. Therefore, by (2) of lemma 2, for any partial state PS and for any nonterminal B, there exists at most one item of the form

$$[A \to \alpha \cdot B\beta, a]$$

in PS. The proof is thus complete. □

By using the above lemmas we can also prove the well known theorem as follows.

Theorem 2 LL(1)-grammars are LR(1)-grammars

Proof: The theorem can be proved by showing that there is no conflict in any reduce state. Let G be an LL(1)-grammar, S be an LR state of G and PS(S,a) be a partial state of S. Let $i=[A \to \alpha \cdot, a]$ be an item in PS(S,a). This means that S is a reduce state with lookahead symbol a and there is no item j such that $i \downarrow^* j$. If $i \in Kernel(S)$ then $PS(S,a)=\{i\}$ by lemma 3 which means there is only one item i with lookahead a, therefore, there is no conflict. If $i \in Nonkernel(S)$ then for any $j \in PS(S,a)$ $j \downarrow^* i$ hold by (1) of lemma 2. Therefore, j is the form of $[B \to \beta \cdot C\gamma, b]$ where C is a nonterminal which means that j causes no reduce or shift action, hence no conflict. This completes the proof. □

Note: There are not much differences between the proofs of both theorems. In the form $[A \to \alpha \cdot B\beta, a]$ in the proof of theorem 1, if $[A \to \alpha \cdot B\beta, a] \in PS(S,a_0)$ and B shrinks to ε the form may be $[A \to \alpha \cdot a_0 \ldots,]$ or $[A \to \alpha \cdot, a_0]$. These forms correspond to the ones in the proof of theorem 2. This fact seems to correspond to the fact that an evaluator of an L-attributed LL(1) grammar can be simulated by that of an S-attributed LR(1) grammar because an LL(1) grammar augmented by ***marker nonterminals*** is an LR(1) grammar [2].

4. Concluding remarks

We have proved that L-attributed LL(1) grammars are LR-attributed. It is worth notice that this has only become possible by defining LR-attributed grammars using LR partial states [6] instead of LR states as in the original definition [4].

We have developed a compiler generator called Rie based on a subclass of LR-attributed grammars [3]. Translators and compilers are being built using Rie. The result of this paper ensures wider applicability of such a class of attribute grammars based on LR grammars.

References

[1] Aho, A. V. and Ullman, J. D., *The Theory of Parsing, Translation and Compiling*, Vol. I: *Parsing*, Vol. II: *Compiling*, Prentice-Hall, 1972, 1973.

[2] Aho, A. V., Sethi, R. and Ullman, J. D., *Compilers, Principles, Techniques, and Tools*, Addison-Wesley, 1985

[3] Ishizuka, H. and Sassa, M., A compiler generator based on an attribute grammar, *Proc. 26th Programming Symposium of IPS Japan*, Hakone, 69-80(1985), (in Japanese).

[4] Jones, N. D. and Madsen, M., Attribute-influenced LR parsing, *Lecture Notes in Comp. Sci.* **94**, 393-407(1980).

[5] Purdom, P. and Brown, C. A., Semantic routines and LR(k) parsers, *Acta Inf.* **14**, 299-315(1980).

[6] Sassa, M., Ishizuka, H. and Nakata, I., A contribution to LR-attributed grammars, *J. Inf. Process.* **8**, 196-206(1985).