Information Processing Letters 35 (1990) 111–114
North-Holland

20 July 1990

A NOTE ON AMBIGUITY IN CONTEXT-FREE GRAMMARS

Tom ALTMAN

Department of Computer Science, University of Kentucky, Lexington, KY 40506, USA

George LOGOTHETIS

AT&T Bell Laboratories, Summit, NJ 07901, USA

Communicated by W.M. Turski
Received 15 December 1989
Revised 19 March 1990

Keywords: Formal languages, theory of computation, ambiguity

1. Introduction

We show that ambiguity in context-free grammars can be decomposed into *union-ambiguity* and *concatenation-ambiguity*. We also show that the traditional proof of the undecidability of ambiguity in context-free grammars addresses only one part of ambiguity, namely the union-ambiguity. We extend this result by showing that the concatenation-ambiguity is also undecidable.

Context-free grammars (CFGs) and the languages they generate, the context-free languages (CFLs), are useful in defining the syntax of programming languages, formalizing the notion of parsing, and numerous other string-processing applications. Chomsky [2,3,4] originated the CFG formalism. Ambiguity in CFGs was first studied by Floyd [6], Cantor [1], and Greibach [7].

Traditionally, the definition of context-free languages from CFGs is expressed in terms of derivations or, equivalently, in terms of parse trees. Let G be a CFG $\langle T, N, P, Z\rangle$, where T and N are the sets of terminals and nonterminals, respectively, P is the set of productions, and $Z \in N$ is the start symbol. A grammar symbol X is a terminal or a nonterminal. The productions in P have the form $A \to \alpha$, where A is a nonterminal and α is either the empty string ϵ or a string $X_1 \ldots X_n$ of grammar symbols. Each nonterminal A defines a language $L(A)$ over the alphabet T: a terminal string x is a sentence of $L(A)$ iff there exists a parse tree with root A and frontier x. By definition, the language $L(G)$ is the language $L(Z)$. A grammar is called *ambiguous* iff there is a sentence of $L(G)$ that has more than one parse tree, and *unambiguous*, otherwise.

2. An algebraic view of ambiguity

We now proceed to describe ambiguity in CFGs in algebraic terms, namely in terms of the standard algebraic operations of $\cup$ (union) and $\circ$ (concatenation) on languages. Let L_1 and L_2 be languages over a common alphabet Σ.

- $L_1 \cup L_2 = \{ x \in \Sigma^* \mid x \in L_1 \text{ or } x \in L_2 \}$.
- $L_1 \circ L_2 = \{ x_1 x_2 \in \Sigma^* \mid x_1 \in L_1 \text{ and } x_2 \in L_2 \}$.

We start by defining the notions of ambiguous union and ambiguous concatenation.

Definition 2.1. The operation $L_1 \cup L_2$ is *ambiguous* if $L_1 \cap L_2 \neq \emptyset$.

Definition 2.2. The operation $L_1 \circ L_2$ is *ambiguous* if there exist $x_1, y_1 \in L_1$, $x_2, y_2 \in L_2$, $x_1 \neq y_1$ and $x_2 \neq y_2$, such that $x_1 x_2 = y_1 y_2$.

Property 2.3. *Both union-ambiguity and concatenation-ambiguity are monotonic: Let L_1, L_2, M_1, and M_2 be languages that satisfy $L_1 \subseteq M_1$ and $L_2 \subseteq M_2$. If $L_1 \cup L_2$ is ambiguous, then $M_1 \cup M_2$ is ambiguous. Furthermore, if $L_1 \circ L_2$ is ambiguous, then $M_1 \circ M_2$ is ambiguous.*

An algebraic formula built in terms of these operations on languages will be called a *language expression*. Definitions 2.1 and 2.2 describe the ambiguity of single language operations; they can also be extended to general language expressions.

Definition 2.4. A language expression is called *ambiguous* if one or more of the operations into which it is decomposed is ambiguous.

Property 2.5. *It can be shown that a language expression of the form $L_1 \cup L_2 \cup \cdots \cup L_n$ is ambiguous iff $L_i \cap L_j \neq \emptyset$ for some $1 \leqslant i < j \leqslant n$.*

Property 2.6. *It can be shown that a language expression of the form $L_1 \circ L_2 \circ \cdots \circ L_n$ is ambiguous iff there exist strings x_i, $y_i \in L_i$, $i = 1, \ldots, n$, such that $x_1 x_2 \ldots x_n = y_1 y_2 \ldots y_n$ and $x_i \neq y_i$ for some $1 \leqslant i \leqslant n$.*

The definition of context-free languages by a CFG can be expressed as the solution to a system of algebraic equations. The unknowns in these equations are the languages of the various nonterminals in the grammar. The equations contain language expressions that are formed from the right-hand sides of the productions in the grammar. More specifically, we first group all productions $A \to \alpha_1$, $A \to \alpha_2, \ldots, A \to \alpha_n$ of a nonterminal A into the form $A \to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n$. This "composite" production for A can then be transformed into the equation $L(A) = L(\alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n)$ that "defines" the language $L(A)$, based on the following simple transformations.

- $L(\alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n) \equiv L(\alpha_1) \cup L(\alpha_2) \cup \cdots \cup L(\alpha_n)$.
- $L(X_1 X_2 \ldots X_n) \equiv L(X_1) \circ L(X_2) \circ \cdots \circ L(X_n)$.
- For $t \in T$, $L(t) \equiv \{t\}$.
- $L(\epsilon) \equiv \{\epsilon\}$.

Therefore, to each nonterminal A in a CFG corresponds an equation $L(A) = le$ defining $L(A)$. The language expression le, the *defining expression* of $L(A)$, has the form of unions of concatenations of languages. Collectively, the definitions of the nonterminal languages form a complete system of language equations that determine a unique minimal solution for each of these languages. For a comprehensive treatment of this algebraic view of context-free languages, the reader is referred to [5].

Definition 2.7. The language $L(A)$ of a nonterminal A in a CFG is *ambiguously defined* if the language expression in the equation defining $L(A)$ is ambiguous. A CFG is *algebraically ambiguous* if the language of one or more of its nonterminals is ambiguously defined.

Theorem 2.8. *A CFG without useless nonterminal symbols is algebraically ambiguous iff it is ambiguous (in the traditional sense).*

Proof. ($\Leftarrow$) Let us assume that a grammar is ambiguous in the traditional sense. Consider two distinct parse trees, T and T', with the same root label A and the same frontier x. There are two possibilities:

(a) The top children nodes in the two parse trees are different. This means that the top level decompositions in the two trees use two different productions, $A \to \alpha_1$ and $A \to \alpha_2$, respectively. This implies that $L(\alpha_1) \cup L(\alpha_2)$ in the equation defining $L(A)$ is ambiguous, since $x \in L(\alpha_1) \cap L(\alpha_2)$.

(b) The top children nodes in the two parse trees are the same, meaning that the top level decompositions in both trees are based on the same production, say $A \to X_1 \ldots X_n$. Each of the two trees is decomposed into n subtrees, $T_1, \ldots, T_n$ and $T_1', \ldots, T_n'$, respectively. Let $x_1, \ldots, x_n$ be the frontiers of $T_1, \ldots, T_n$ and $x_1', \ldots, x_n'$ be the frontiers of $T_1', \ldots, T_n'$, respectively. Clearly, we have $x_1 \ldots x_n = x_1' \ldots x_n' = x$. Two possibilities arise:

(1) $x_i \neq x_i'$ for some $1 \leqslant i \leqslant n$. In this case, the language expression $L(X_1) \circ \cdots \circ L(X_n)$ in the equation defining $L(A)$ is ambiguous.

(2) $x_i = x_i'$ for all $1 \leqslant i \leqslant n$. In this case, for each $1 \leqslant i \leqslant n$, T_i and T_i' have the same root, X_i, and the same frontier, x_i. Since $T \neq T'$, some T_i must be different from T_i', and we can then apply the same approach to each of the tree pairs. This process will eventually terminate under case (a) or case (b1), since each subtree is shorter than its parent tree.

This shows that traditional ambiguity implies algebraic ambiguity.

($\Rightarrow$) Let us assume that the definition of some $L(A)$ has a union-ambiguity. This means that there are two productions $A \to \alpha_1$ and $A \to \alpha_2$ of A such that $L(\alpha_1) \cap L(\alpha_2) \neq \emptyset$. It follows that for every $x \in L(\alpha_1) \cap L(\alpha_2)$ there are two distinct parse trees with root A and frontier x, differing at the top-level decomposition because one tree uses production $A \to \alpha_1$ and the other $A \to \alpha_2$.

Now, let us assume that the definition of some $L(A)$ has a concatenation-ambiguity. This means that there is a production $A \to X_1 X_2 \ldots X_n$ such that $L(X_1) \circ L(X_2) \circ \cdots \circ L(X_n)$ is ambiguous. This, in turn, implies that there are strings x_i, $y_i \in L(X_i)$, $i = 1, \ldots, n$, such that $x_1 x_2 \ldots x_n = y_1 y_2 \ldots y_n = x$ and $x_i \neq y_i$ for some $1 \leqslant i \leqslant n$. Therefore, there are two distinct parse trees with the same root A, the same frontier x, and the same top-level decomposition via the production $A \to X_1 \ldots X_n$, differing in the subtrees with root X_i that have different frontiers, namely x_i and y_i, respectively. □

3. Undecidability of concatenation-ambiguity

The problem of detecting ambiguity in CFGs can be shown to be undecidable by reducing Post's correspondence problem (PCP) to it. The proof, reproduced below, is basically the one found in [8]. An instance PCP(A, B) of Post's correspondence problem consists of two lists of strings over a common alphabet Σ, $A = w_1, w_2, \ldots, w_k$ and $B = u_1, u_2, \ldots, u_k$. The instance PCP($A$, B) has a solution iff there exists a sequence $i_1, i_2, \ldots, i_n$ of integers such that $n \geqslant 1$, $1 \leqslant i_j \leqslant k$ for $j = 1, \ldots, n$, and

$$w_{i_1} w_{i_2} \ldots w_{i_n} = u_{i_1} u_{i_2} \ldots u_{i_n}.$$

Let $A = w_1, w_2, \ldots, w_k$ and $B = u_1, u_2, \ldots, u_k$ be the lists of strings over Σ in an instance of a PCP(A, B). Also, let $a_1, a_2, \ldots, a_k$ be k distinct symbols not in Σ, and consider the following CFG corresponding to the PCP(A, B).

$$S \to W \mid U,$$
$$W \to w_i W a_i \mid w_i a_i,\ i = 1, \ldots, k,$$
$$U \to u_i U a_i \mid u_i a_i,\ i = 1, \ldots, k.$$

Clearly, the PCP(A, B) has a solution iff this grammar is ambiguous. See [8] for details.

Now, since $a_1, a_2, \ldots, a_k$ are distinct symbols not in Σ, this grammar can be ambiguous if and only if the union $L(W) \cup L(U)$ implied in the definition of S is ambiguous. In other words, the ability to decide whether the union of two context-free languages is ambiguous is all that is needed to solve PCP. The above proof leaves open the decidability question for the problem of detecting ambiguity of concatenation of context-free languages. As we show next, this problem is undecidable too.

Theorem 3.1. *The problem of union-ambiguity detection is reducible to the problem of concatenation-ambiguity detection.*

Proof. Let L and M be two languages over a common alphabet Σ. Let us suppose that we want to determine whether or not $L \cup M$ is ambiguous, i.e., whether or not $L \cap M \neq \emptyset$. Let \$ be a symbol not in Σ, and consider the languages $L' = L \circ \{\$\} \cup \{\epsilon\}$ and $M' = M \circ \{\$\} \cup \{\epsilon\}$ over the alphabet $\Sigma \cup \{\$\}$. We will show that $L \cup M$ is ambiguous iff $L' \circ M'$ is ambiguous.

($\Rightarrow$) Let's assume that $L \cup M$ is ambiguous, and let $x \in L \cap M$. Consider $y_1 = x\$ \in L'$, $z_1 = \epsilon \in M'$ and $y_2 = \epsilon \in L'$, $z_2 = x\$ \in M'$. Clearly, $y_1 z_1 = y_2 z_2 = x\$$, and $y_1 \neq y_2$, $z_1 \neq z_2$. Therefore, $L' \circ M'$ is ambiguous.

($\Leftarrow$) Let us assume that $L' \circ M'$ is ambiguous, and consider $y_1, y_2 \in L'$, $y_1 \neq y_2$, and $z_1, z_2 \in M'$, $z_1 \neq z_2$, such that $y_1 z_1 = y_2 z_2$. Since $y_1 \neq y_2$, w.l.o.g. let y_1 be the longer of the two strings. Since $y_1 \neq \epsilon$, y_1 must have the form $y\$$, where $y \in L$. Since y_2 must be a prefix of y, and y cannot contain \$, y_2 can only be ϵ. Therefore, we have

$y\$z_1 = z_2$. Since $z_2 \neq \epsilon$, z_2 must have the form $z\$$, where $z \in M$. Therefore, we have $y\$z_1 = z\$$. Since z cannot contain \$, z_1 must be ϵ. Therefore, we have $y\$ = z\$$, which implies $y = z$, which implies that $L \cap M \neq \emptyset$, i.e., that $L \cup M$ is ambiguous. □

Theorem 3.2. *The problem of concatenation-ambiguity detection for context-free languages is undecidable.*

Proof. If L and M are context-free languages, the languages $L' = L \circ \{\$\} \cup \{\epsilon\}$ and $M' = M \circ \{\$\} \cup \{\epsilon\}$ used in the proof of Theorem 3.1 are both context-free. It follows that for context-free languages the union-ambiguity detection is reducible to the concatenation-ambiguity detection. □

References

[1] D.C. Cantor, On the ambiguity problem of Backus systems, *J. ACM* **9** (4) (1962) 477–479.

[2] N. Chomsky, Three models for the description of language, *IEEE Trans. Inform. Theory* **2** (3) (1956) 113–124.

[3] N. Chomsky, On certain formal properties of grammars, *Inform. and Control* **2** (2) (1959) 137–167.

[4] N. Chomsky, Formal properties of grammars, in: *Handbook of Math. Psych. Vol. 2* (Wiley, New York, 1963).

[5] J.H. Conway, *Regular Algebra and Finite Machines* (Chapman & Hall, London, 1971).

[6] R.W. Floyd, On ambiguity in phrase structure languages, *Comm. ACM* **5** (10) (1962) 526–534.

[7] S.A. Greibach, The undecidability of the ambiguity problem for minimal linear grammars, *Inform. and Control* **6** (2) (1963) 119–125.

[8] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation* (Addison-Wesley, Reading, MA, 1979).