TEORÍA DE AUTÓMATAS Y LENGUAJES FORMALES

Francisco Vico

departamento
Lenguajes y
Ciencias de la Computación

área de conocimiento
Ciencias de la Computación e
Inteligencia Artificial

ETSI Informática
Universidad de Málaga

fjvico@uma.es
geb.uma.es/fjv

10 de diciembre de 2015

Notation

$\mathbb{N}$	the set of nonnegative integers (or natural numbers), i.e., {0, 1, 2, … } (U+2115)
$\mathbb{P}$	the set of positive integers (U+2119)
$\mathbb{R}$	the set of real numbers (U+211D)
$\mathbb{Z}$	the set of integers (U+2124)
$\varnothing$	the empty set (U+2205)
$\subseteq$	the (infix) subset relation between sets (U+2286)
$\subset$	the (infix) proper subset relation between sets (U+2282)
$\cup$	the infix union operation on sets (U+222A)
$\cap$	the infix intersection operation on sets (U+2229)
$\sim$	the prefix complementation operation on sets (U+007E)
$-$	the infix set difference operation on sets (U+2212)
$\times$	the infix cartesian product of sets (U+00D7)
A^n	the postfix n-fold cartesian product of A, i.e. A× … ×A (n times)
2^A	the powerset of A

Context-free languages

From a practical point of view, for each grammar $G = (\Sigma, V, S, P)$ representing some language, the following two problems are important:

1. Membership problem: Given a string over Σ, does it belong to $L(G)$?
2. Parsing problem: Given a string in $L(G)$, how can it be derived from S?

The importance of the membership problem is quite obvious—given an English sentence or computer program, we wish to know if it is grammatically correct or has the right format. Solving the membership problem for context-free grammars is an integral step in the lexical analysis of computer programs, namely the stage of decomposing each statement into tokens, prior to fully parsing the program. For this reason, the membership problem is also often referred to as lexical analysis (cf. [Drobot, 1989]). Parsing is important because a derivation usually brings out the "meaning" of the string. For example, in the case of a Pascal program, a derivation of the program in the Pascal grammar tells the compiler how the program should be executed. The following theorem qualifies the decidability of the membership problem for the four classes of grammars in the Chomsky hierarchy. Proofs of the first assertion can be found in [Chomsky, 1963, Harrison, 1978, Hopcroft and Ullman, 1979], while the second assertion is treated below.

Theorem The membership problem for Type-0 grammars is undecidable in general, but it is decidable given any context-sensitive grammar. For context-free grammars the problem is decidable in polynomial time, and for regular grammars, linear time. Since context-free grammars play a very important role in describing computer programming languages, we discuss the membership and parsing problems for context-free grammars in more detail in this and the next section. First, let us look at another example of a context-free grammar. For convenience, let us abbreviate a set of productions

$$A \to \alpha_1, \ldots, A \to \alpha_n$$

with the same left-hand side nonterminal as

$$A \to \alpha_1 \mid \ldots \mid \alpha_n .$$

Definition 6.4. Each derivation $X_0 \Rightarrow_G^* w$ a so-called derivation tree (or parse tree).

Definition 6.8. A context-free grammar G is ambiguous if there is a string x ∈ L(G) that has two distinct derivation trees. Otherwise G is unambiguous.

Unambiguity is a very desirable property because it promises a unique interpretation of each sentence in the language.

Example Consider a grammar for all valid arithmetic expressions that are composed of unsigned positive integers and symbols +, ∗, (,). For convenience, let us use the symbol n to denote any unsigned positive integer—it is treated as a terminal. This grammar has the productions

$$S \rightarrow T{+}S \mid S{+}T \mid T$$
$$T \rightarrow F{*}T \mid T{*}F \mid F$$
$$F \rightarrow n \mid (S)$$

Two possible different derivation trees for the expression 1 + 2 ∗ 3 + 4 are shown in the figure. Thus the grammar is ambiguous. The left tree means that the first addition should be done before the second addition, while the right tree says the opposite.

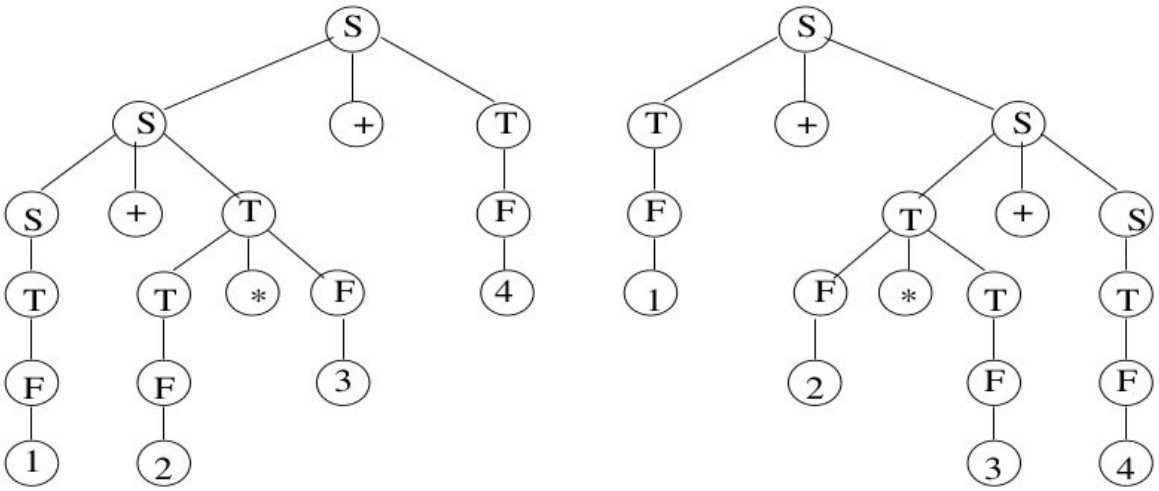

Figure Different derivation trees for the expression 1 + 2 ∗ 3 + 4.

Although in the above example different derivations/interpretations of any expression always result in the same value because the operations addition and multiplication are associative, there are situations where the difference in the derivation can affect the final outcome. Actually, the above grammar can be made unambiguous by removing the redundant productions $S \rightarrow T{+}S$ and $T \rightarrow F{*}T$.

This corresponds to the convention that a sequence of consecutive additions or multiplications is always evaluated from left to right. Deleting the two productions does not change the language of strings generated by the grammar, but it does fix unique interpretations of those strings.

It is worth noting that there are context-free languages that cannot be generated by any unambiguous context-free grammar. Such languages are said to be inherently ambiguous. An example taken from [Hopcroft and Ullman, 1979] (where this fact is proved) is

$$\{0^m1^m2^n3^n | m, n > 0\} \cup \{0^m1^n2^n3^m | m, n > 0\}.$$

The reason is that every context-free grammar G must yield two parse trees for some strings of the form $x = 0^n1^n2^n3^n$, where one tree intuitively expresses that x is a member of the first set of the union, and the other tree expresses that x is in the second set.

Definition 6.6. A derivation from a sentential form to another is said to be leftmost (or rightmost) if at each step the leftmost (or rightmost, respectively) nonterminal is replaced.

Definition 6.22. A CF-grammar is in Chomsky's normal form if its productions are all of the form:

$$X \to YZ \quad \text{or} \quad X \to a$$

Definition 6.24. A CF-grammar is in Greibach's normal form if its productions are of the form:

$$X \to aw$$

Definition 6.28. A pushdown automaton (PDA) is a septuple $M = (Q, \Sigma, \Gamma, S, Z, \delta, A)$ where:

- $Q = \{q_1, \ldots, q_m\}$ is a finite set of states, the elements if which are called states.
- Σ is the input alphabet, the alphabet of the language.
- Γ is the finite stack alphabet, i.e., the set of symbols appearing in the stack.
- $S \subseteq Q$ is the set of initial states.
- $Z \subseteq \Gamma$ is the set of bottom symbols of the stack.
- δ is the transition function which maps each triple (q_i, a, X), where q_i is a state, a is an input symbol or Λ and X is a stack symbol, to exactly one finite set $T = \delta(q_i, a, X)$ (possibly empty) of pairs (q, α) where q is a state and α is a word over the stack alphabet; cf. the transition function of a Λ-NFA.
- $A \subseteq Q$ is the set of terminal states.

Definition 6.30. In order to define the way a PDA handles its memory structure, we introduce the triples (q_i, x, α) where qi is a state, x is the unread part (suffix) of the input word and α is the contents of the stack, given as a word with the "topmost" symbol at left. These triples are called configurations of M.

Definition 6.41. A configuration chain of M accepting the word w (and ending with an empty stack) then corresponds to a leftmost derivation of w by the CF-grammar:

$$G = (\Delta \cup \{X_0\}, \Sigma, X_0, P)$$

where the productions P are given above. Conversely, a leftmost derivation of the word w by G corresponds to a chain of configurations of M accepting w.

Definition 6.47. For any PDA the language recognized by it is a CF-language. Moreover, it may be assumed that the PDA then has only three states, an initial state, an intermediate state and a terminal state, and only one bottom symbol.

Definition 6.48. If a CF-language L can be generated by a grammar in Chomsky's normal form having p nonterminals, $z \in L$ and $|z| \geq 2^{p+1}$, then z may be written in the form $z = uvwxy$ where $|vwx| \leq 2^{p+1}$, $vx \neq \Lambda$, $w \neq \Lambda$, and the words $uv^n wx^n y$ are all in L.

Definitions

ambiguous context-free grammar: a context-free grammar in which some derivable terminal strings have two distinct derivation trees.

Chomsky normal form: a form of context-free grammar in which every rule has the form $A \to BC$ or $A \to a$, where A, B, C are nonterminals and a is a terminal.

context-free grammar: a grammar whose rules have the form $A \to \beta$, where A is a nonterminal and β is a string of nonterminals and terminals.

context-free language: a language that can be described by some context-free grammar.

context-sensitive grammar: a grammar whose rules have the form $\alpha \to \beta$, where α, β are strings of nonterminals and terminals, and $|\alpha| \leq |\beta|$.

context-sensitive language: a language that can be described by some context-sensitive grammar.

derivation or parsing: a sequence of applications of rules of a grammar that transforms the start symbol into a given terminal string or sentential form.

derivation tree or parse tree: a rooted, ordered tree that describes a particular derivation of a string with respect to some context-free grammar.

(formal) language: a set of strings over some fixed alphabet.

(formal) grammar: a description of some language, typically consisting of a set of terminals, a set of nonterminals, a distinguished nonterminal called the start symbol, and a set of rules (or productions) of the form $\alpha \to \beta$, which determine which substrings α of a sentential form can be replaced by some other string β.

leftmost (or rightmost) derivation: a derivation in which at each step, the leftmost (respectively, rightmost) nonterminal is rewritten.

membership problem (or lexical analysis): the problem or process of deciding whether a given string is generated by a given grammar.

parsing problem: the problem of reconstructing a derivation of a given input string in a given grammar.

regular expression: a description of some language using the operators union, concatenation, and Kleene closure.

regular language: a language that can be described by some regular expression, or equivalently, by some right-linear/regular grammar.

right-linear or regular grammar: a grammar whose rules have the form $A \rightarrow cB$, $A \rightarrow c$, or $A \rightarrow \varepsilon$, where A, B are nonterminals, c is a terminal, and ε is the empty string.

sentential form: a string of terminals and nonterminals obtained at some step of a derivation in a grammar.

Bibliography

(Links are provided to either publicly available PDF files, or to books in the University of Malaga's library stock, where available.)

[Angluin, 1980] Angluin, D. 1980. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*. 21:46-62.

[Chomsky, 1956] Chomsky, N. 1956. Three models for the description of language. *IRE Trans. on Information Theory*. 2(2):113-124.

[Chomsky, 1963] Chomsky, N. 1963. Formal properties of grammars. In *Handbook of Mathematical Psychology* Vol. 2, 323-418. John Wiley and Sons, New York.

[Chomsky and Miller, 1958] Chomsky, N. and Miller, G. 1958. Finite-state languages. *Information and Control*. 1:91-112.

[Drobot, 1989] Drobot, V. 1989. *Formal Languages and Automata Theory*. Computer Science Press, Rockville, MD.

[Floyd and Beigel, 1994] Floyd, R.W. and Beigel, R. 1994. *The Language of Machines: an Introduction to Computability and Formal Languages*. Computer Science Press, New York.

[Gurari, 1989] Gurari, E. 1989. *An Introduction to the Theory of Computation*. Computer Science Press, Rockville, MD.

[Harel, 1992] Harel, D. 1992. *Algorithmics: The Spirit of Computing*. Addison-Wesley, Reading, MA.

[Harrison, 1978] Harrison, M. 1978. *Introduction to Formal Language Theory*. Addison-Wesley, Reading, MA.

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

[Jiang et al, 2002] Jiang, T., Li, M., Ravikumar, B. and Regan, K.W. 2002. *Formal grammars and languages*. Retrieved on December 7, 2015, from http://www.cs.ucr.edu/~jiang/cs215/tao-new.pdf.

[Jiang et al., 1995] Jiang, T., Salomaa, A., Salomaa, K., and Yu, S. 1995. Decision problems for patterns. *Journal of Computer and System Sciences*. 50(1):53-63.

[Kleene, 1956] Kleene, S. 1956. Representation of events in nerve nets and finite automata. In *Automata Studies*, 3-41. Princeton University Press, NJ.

[Kleene, 1952] Kleene, S. 1952. *Introduction to Metamathematics*. Walters-Noordhoff & North-Holland.

[Lind and Marcus, 1995] Lind, D. and Marcus, 1995 *Symbolic Dynamics*, Academic Press.

[Normann, 2010] Normann, D. 2010. *Introduction to Computability Theory*. Retrieved on December 7, 2015, from http://www.mn.uio.no/math/tjenester/kunnskap/kompendier/comptheory.pdf.

[Post, 1943] Post, E. 1943. Formal reductions of the general combinatorial decision problems. *Amer. J. Math*. 65:197-215.

[Ramos and Morales, 2011] Ramos, G., Morales, R. (2011) *Teoría de Autómatas y Lenguajes Formales*. Edición de Autor.

[Ruohonen, 2009] Ruohonen, K. (2009). *Formal languages*. Retrieved on December 7, 2015, from http://math.tut.fi/~ruohonen/FL.pdf.

[Salomaa, 1966] Salomaa, A. 1966. Two complete axiom systems for the algebra of regular events. *J. ACM.* 13(1):158-169.

[Searls, 1993] Searls, D. 1993. The computational linguistics of biological sequences. In *Artificial Intelligence and Molecular Biology*. L. Hunter (ed.), MIT Press, 1993, pp. 47-120.

[Wood, 1987] Wood, D. 1987. *Theory of Computation*. Harper and Row.