A Toolkit for Generating Sentences from Context-Free Grammars

Zhiwu Xu*, Lixiao Zheng* and Haiming Chen*
*State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences,
Beijing, China
Email: {zhiwu, zhenglx, chm}@ios.ac.cn

***Abstract*—Producing sentences from a grammar, according to various criteria, is required in many applications. It is also a basic building block for grammar engineering. This paper presents a toolkit for context-free grammars, which mainly consists of several algorithms for sentence generation or enumeration and for coverage analysis for context-free grammars. The toolkit deals with general context-free grammars. Besides providing implementations of algorithms, the toolkit also provides a simple graphical user interface, through which the user can use the toolkit directly. The toolkit is implemented in Java and is available at http://lcs.ios.ac.cn/~zhiwu/toolkit.php. In the paper, the overview of the toolkit and the description of the GUI are presented, and experimental results and preliminary applications of the toolkit are also contained.**

***Keywords*-context-free grammars; sentence generation; sentence enumeration; coverage criteria; toolkit**

I. INTRODUCTION

Grammars, especially context-free grammars, are fundamental structures in computer science. Producing sentences from a grammar, according to various coverage criteria or some other constraints, is required in many applications, such as parser/compiler testing, natural language processing, grammar validation, test case generation, bioinformatics, etc. Recently, grammar engineering has been recognized as an emerging field of software engineering due to the fact of lacking solid engineering methods and techniques for grammars on one hand, and due to the importance of grammars in computer science and in other areas on the other hand [1]. Techniques for generating sentences form a basic building block for grammar engineering [2].

Thus a toolkit package for generating sentences is useful. However, presently only very limited abilities to generate sentences are provided in a few grammar tools [3], [4]. On the other hand, there are research issues on sentence generation or enumeration algorithms that need to explore. Based on the research on the algorithms by the authors and others, this paper presents a toolkit for context-free grammars, which mainly consists of several algorithms for sentence generation or enumeration and for coverage analysis for context-free grammars.

This work was supported by the National Natural Science Foundation of China under Grant Nos. 60573013, 60721061.

The commonly used algorithm for sentence generation from context-free grammars is the one proposed by Purdom [5], which produces a small set of sentences satisfying rule coverage. We found out through analysis and experiments that in most cases, Purdom's algorithm produces too few sentences and the sentences differ too much in length, i.e., some of them are much longer and more complex in structure than others. These sentences are inadequate in many practical applications. To avoid this, an extended algorithm [6] is proposed and implemented which generates more and simpler sentences. A more precise criteria is context-dependent rule coverage [2]. However, as noted in [7], implementation and computation of this metric is more involved. A sentence generation algorithm fulfilling context-dependent rule coverage based on Purdom's algorithm is proposed and implemented [8], which, to the best of the authors' knowledge, is the first algorithm for context-dependent rule coverage. An extended algorithm, similar to the idea of the above improvement of Purdom's algorithm, is also proposed and implemented in the toolkit.

The toolkit contains algorithms [9], [10] for sentence and parse tree enumeration for context-free grammars, which are the first linear algorithms and answer the open problem [11] about the existence of such algorithms. The algorithms also lead to several methods of sentence generation for context-free grammars, generally called user-controlled methods in the paper, such as random generation, bounded range enumeration, and structure-sensitive generation of sentences.

The toolkit deals with general context-free grammars, which have no restrictions on grammars, and is implemented in Java as Java classes. Besides providing implementations of algorithms, it also provides a simple graphical user interface, through which one can use the toolkit directly.

The algorithms implemented in the toolkit have been tested through experiments. The toolkit has also been used in some preliminary applications, which are briefly introduced in the paper.

The paper is organized as follows. In the next section we review the basic concepts and notations of context-free grammars. Section III gives an overview of the toolkit. Section IV describes the graphical user interface and presents some examples. Section V gives some applications that use our toolkit. Section VI discusses related work. Section VII

contains a conclusion.

II. Context-Free Grammar

A *context-free grammar* is a tuple $G = \langle N, T, P, S\rangle$, where N is a finite set of *nonterminals*, T is the alphabet or a finite set of *terminals* ($N \cap T = \emptyset$), $S \in N$ is the *start symbol* and $P \in N \times (N \cup T)^*$ is a finite set of *rules*. If $(X, u) \in P$, we write $X \to u$, and call X the left-hand side and u the right-hand side. If there are several rules with the same left-hand side X, we write $X \to u_1|\cdots|u_n$.

Given $\alpha, \beta \in (N \cup T)^*$, we write $\alpha \Rightarrow \beta$ if there exist $\gamma, \delta, \eta \in (N \cup T)^*$ and $X \in N$ such that $\alpha = \gamma X \delta$, $\beta = \gamma\eta\delta$, and $X \to \eta \in P$. We write $\alpha \overset{*}{\Rightarrow} \beta$ if there exist $\alpha_0, \ldots, \alpha_n (n \geq 0)$ such that $\alpha = \alpha_0 \Rightarrow \ldots \Rightarrow \alpha_n = \beta$.

If $S \overset{*}{\Rightarrow} \alpha \in T^*$, we say α is a *sentence* of G. The context-free language defined by G, denoted by $\mathcal{L}(G)$, is composed of all sentences of G.

Let $N = \{X_1, \ldots, X_m\}$. Let $\mathcal{L}_i$ be the language of $G_i = \langle N, T, P, X_i\rangle$, $i = 1, \ldots, m$. If there is a derivation $S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} \alpha u \beta \overset{*}{\Rightarrow} w$, where $X \in N$, $\alpha, \beta \in (N \cup T)^*$ and $u, w \in T^*$, we say u is a sub-sentence of w. Moreover, if $S \overset{p}{\Rightarrow} u_0 X_1 u_1 \ldots X_r u_r \overset{*}{\Rightarrow} u_0 v_1 u_1 \ldots v_r u_r (= w)$, where r is the number of nonterminals at the right-hand side of rule p and $v_i \in T^*$, we say v_i is the i-th (from left to right) direct sub-sentence of w.

There are four different kinds of sentence sets constructed in Dong's algorithm [9], [10]:

- $heap_n^i$, containing the sentences in $\mathcal{L}_i$ whose height[1] is no more than $n + 1$,
- $hierarchy_n^i$, containing the sentences of height $n + 1$ in $\mathcal{L}_i$,
- $cluster_n^p$, containing the sentences of height $n+1$ using rule p as the first one in the derivation, and
- $cube_n^{p,t}$, containing the sentences in $cluster_n^p$ whose t-th (from right to left) direct sub-sentence is from the $(n-1)$-th hierarchy.

Then the volume data are defined as the numbers of sentences contained in each set.

III. The Toolkit: an Overview

In this section, we give an overview of our toolkit. The toolkit contains the following functionalities and algorithms:

- Manipulations of context-free grammars
  - Editing of context-free grammars
  - Checking the usefulness and reachability of nonterminals
- Sentence generation with coverage criteria
  - Purdom's algorithm [5]
  - An extension of Purdom's algorithm [6]
  - CDRC-P algorithm [8]
  - An extension of CDRC-P algorithm
- Sentence enumeration
  - Dong's algorithms [9], [10]
  - Sentence generation methods based on sentence enumeration
- Sentence analysis
  - Earley's algorithm [12]
  - Coverage analysis algorithms
- Graphical user interface (GUI)
  - Sentence generation panel
  - Sentence enumeration panel
  - Sentence analysis panel

The toolkit provides an editor for the user to input and edit a context-free grammar. The editor can check whether the grammar is *well-defined*, that is, the grammar has no useless nor unreachable nonterminals. The toolkit focuses on sentence generation, sentence enumeration and coverage analysis, which constitute the main functionalities of the toolkit. The toolkit also contains a simple graphical user interface, which consists of three panels corresponding to the three main functionalities and through which the user can use the toolkit directly. The toolkit is available at http://lcs.ios.ac.cn/~zhiwu/toolkit.php.

The sentence generation algorithms provided in our toolkit are based on grammar coverage criteria, including rule coverage [5] and context-dependent rule coverage [2]. Purdom's algorithm [5] generates a small sentence set satisfying rule coverage, while an extension of Purdom's algorithm [6] builds upon Purdom's algorithm with a length control mechanism. For context-dependent rule coverage, similarly there are two algorithms, namely CDRC-P algorithm [8], based on Purdom's algorithm, and an extension of CDRC-P algorithm, extending CDRC-P algorithm with a length control mechanism.

Dong's algorithm partitions the sentences into (in)finite sets, then sorts each sentence set according to some order, called *hierarchical lexicographic order* [9], [10]. Based on Dong's algorithm, several generation methods of sentences, called user-controlled methods, are implemented.

The toolkit is implemented in Java as Java classes. It can be used as a Java library. By adding it into the classpath, the user can invoke the algorithms contained in the toolkit in his own programs. The javadoc API specification is available online too.

To illustrate the toolkit, we use the following grammar as the sample grammar throughout the paper. We use angle brackets to distinguish nonterminals from terminals.

```
<S> -> <E>
<E> -> <E> + <T> | <T>
<T> -> <T> * <F> | <F>
<F> -> id | ( <E> )
```

[1] In the paper, the height of a parse tree is defined as the maximum path length which is the number of nonterminals in a path traversed from the root to the leaf node.

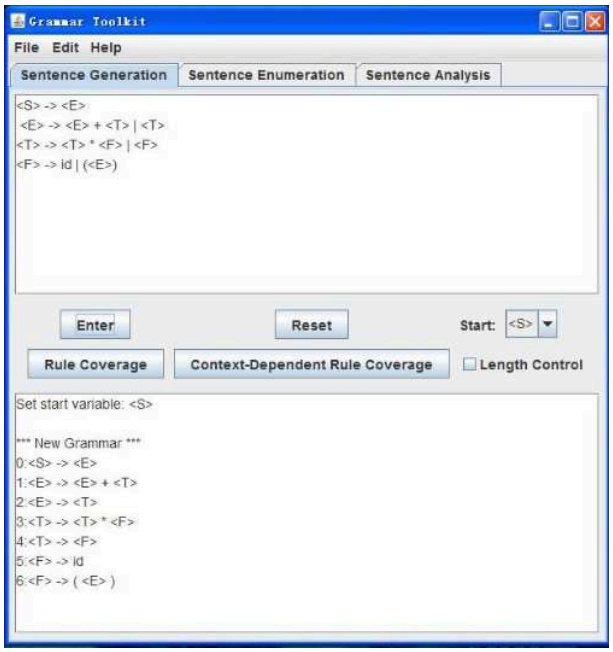

Figure 1. Editing of the sample grammar

IV. Graphical User Interface

The interface consists of three panels, which correspond to the three main functionalities: sentence generation, sentence enumeration and sentence analysis. Each of these panels is implemented independently as a Java class, so that each of them can be used independently. A general view of the interface is presented in Figure 1.

A. Editor and Console

Each panel contains a grammar editor which enables the user to edit, load or save a grammar and to check whether a grammar is well-defined, and an output console which outputs the outcomes and extra information like sentence length, running time and so on.

The grammar rules are written as the following form: $<X>->\alpha_1|\ldots|\alpha_n$, where nonterminals are bracketed by angle brackets to be distinguished from terminals. Once the grammar rules are input, the editor will check whether the grammar is well-defined. Figure 1 shows an example of editing the sample grammar. By default, the start symbol corresponds to the left-hand side of the first rule. The user can also indicate the start symbol through the "Start" dropdown box.

B. Sentence Generation Panel

The sentence generation panel allows the user to generate a sentence set that satisfies rule coverage or context-dependent rule coverage. A length control mechanism can as well be configured on in the sentence generation process by marking the "Length Control" check box. After generation, the number of generated sentences, the information of sentences' lengths, and the total time for generating sentences are also counted. For instance, we generate sentence sets from the sample grammar by invoking each algorithm. The results are summarized in Table I, where "Purdom" denotes Purdom's algorithm, "Purdom-E" an extension of Purdom's algorithm, "CDRC-P" CDRC-P algorithm and "CDRC-P-E" an extension of CDRC-P algorithm.

Algorithm	No. of sentences	Avg len. of sentences	Min. of lengths	Max. of lengths
Purdom	1	19	19	19
Purdom-E	2	12	10	14
CDRC-P	2	23.5	5	42
CDRC-P-E	7	15.6	5	19

Table I
SENTENCES GENERATED BY EACH ALGORITHM.

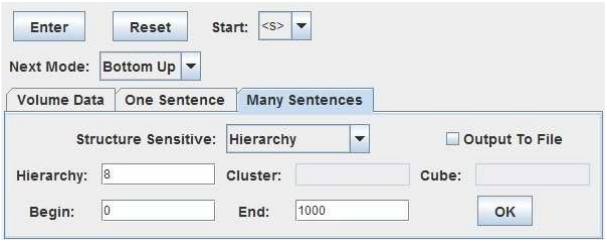

Figure 2. Sentence Enumeration Panel

C. Sentence Enumeration Panel

The sentence enumeration panel consists of three tabs. Figure 2 shows a general view of the panel.

The first tab enables the user to query the volume data among a bounded hierarchical range by giving the "From" hierarchy index and the "To" hierarchy index. It also allows the user to choose which kinds of volume data to query by marking the corresponding check boxes.

The second tab allows the generation of the previous or next sentence for a given sentence and the conversion between sentences and natural numbers. There are two modes, bottom-up and top-down, for computing the next or previous sentence of a given sentence. Through the "Next Mode" dropdown box, the user can choose either he likes. According to hierarchical lexicographic order [9], [10], there exists a one-to-one correspondence between natural numbers and sentences. Therefore, given a sentence, we can compute its sequence number and vice versa.

The third tab allows the user to enumerate a bounded range of sentences by giving the sequence numbers of the first and last sentences. The range can be restricted in a hierarchy, cluster or cube, by giving the corresponding indices in which case only the sentences in the intersection of the range and the restriction are enumerated. After enumeration, the total time for enumerating sentences, the actual number of enumerated sentences and the average time for enumerating

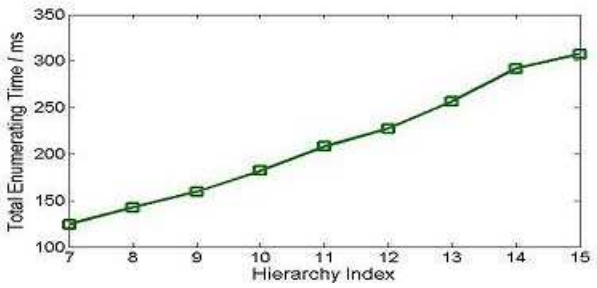

Figure 3. The total time of enumerating sentences on the sample grammar

a sentence are also counted. The hierarchies, clusters and cubes are used for expert users. For normal users, we will consider how to give the restriction by giving some rules or structures in the future.

When enumerating a bounded range of sentences, the console does not output all the sentences. Additionally, the toolkit allows the user to export the enumerated sentences into a file if he sets the "File" checkbox on. The output file contains the corresponding sequence number and parse tree for each enumerated sentence, as well as the statistic information mentioned above.

Using the toolkit, we conducted some experiments for Dong's algorithm. One example is on the sample grammar. We enumerated the first 1000 sentences of each hierarchy from the 7-th hierarchy to the 15-th hierarchy (the numbers were selected arbitrarily). We chose the top-down mode to generate the next sentence. The experiments were performed on a PC of P4 3.2G CPU and 2G RAM. These experimental results are given in Figure 3, which shows that the real computing time for generating the next sentence of height n is $O(n)$.

D. Sentence Analysis Panel

Through the sentence analysis panel, the user can parse a sentence or analyze the coverage of a sentence set, including rule coverage and context-dependent rule coverage. For example, given a grammar and a sentence, if the sentence belongs to the grammar, all possible parses of the sentence are extracted. This is useful for checking sentence ambiguity.

V. Applications

The toolkit has been used in several research projects, namely SAQ [13], grammar testing methods [14], and testing of LFC.

SAQ supports construction and validation of specifications consisting of context-free grammars and operations upon context-free languages. Initially SAQ supports grammar validation but quite weak [13]. We enhanced it by adding several sentence generation and enumeration algorithms contained in this toolkit. In the experiments, several errors in a number of grammars in SAQ bases were found. It is observed that the bounded range enumeration is very useful for this kind of grammar validation.

The second project is an extension of the research of the grammar validation of SAQ, which aims at systematic methods and techniques for testing context-free grammars. A systematic framework for testing context-free grammars is implemented using the algorithms in the toolkit [14]. From experiments some errors were found in the C and Java grammars that are from comp.compilers FTP at ftp://ftp.iecc.com/pub/file/, see [14] for detail.

LFC is a language based on recursive functions on context-free languages [15], [16], [17]. It uses context-free grammars to specify data types, and supports pattern matching definition of functions, where patterns represent structures of the grammars. LFC was designed to support formal specification acquisition and was used in SAQ. Test data for LFC programs can be derived directly by using the sentence generation algorithms provided by this toolkit. We have conducted preliminary experiments to investigate the usefulness of the generated test data in terms of fault detection capability. In particular, we took CDRC-P algorithm as test data generator and conducted mutation testing [18] on five LFC programs, which are taken from SAQ. We applied five mutation operators [19] to each program and compared the result of each mutant with the original result to check whether a mutant is killed by the test data. Through experiments and analysis, we found that the ratio of the killed mutants to the total mutants is above 30% for all the five programs, which reflects the relatively good capability for fault detection of the test data generated by the CDRC-P algorithm in this toolkit.

Furthermore, there are situations where sentences that satisfy some properties are required. In these cases we can enumerate sentences in a bounded range and select valid sentences w.r.t. the properties from them.

VI. Related Work

Bilska et al. [20] presented a collection of tools for formal languages and automata theory. The tools include JFLAP for creating and simulating finite automata, pushdown automata and Turing machines, PumpLemma for proving specific languages are not regular, and some others. Møler [21] implemented a grammar tool which contains a parser and an ambiguity analyzer for context-free grammars. The parser parses a sentence without giving parses. This tool is used mainly for analyzing grammar ambiguity. Almeida et al. [22] presented an interactive graphical environment CGM for the manipulation of context-free languages. It allows the editing of context-free grammars, the conversion to Chomsky normal form, sentence parsing and the construction of parse trees. The above tools do not provide sentence generation, sentence enumeration, or coverage analysis as our toolkit.

Programming Languages Laboratory at University of Calgary provided an online tool `Context Free Grammar`

`Checker` [3] to check the basic properties of context free grammars, including the usefulness and reachability of the nonterminals, the first sets, the follow sets and so on. After checking, it generates no more than 20 sentences, which are the first 20 ones ordered by the sentence length (token number). Compared with ours, its sentence generation is too simple.

Another related work is the sentence generation tool `Forson` [4], implemented by Alfonso Tarantini. `Forson` takes a Bison grammar file as input and provides random sentence generation, following Grow's algorithm (may not terminate), or sentence generation with rule coverage, following Purdom's algorithm, for the grammar defined in the input file. It only covers a small part of the functionalities of our toolkit. Moreover, `Forson` is a batch program, which means that it does not provide a graphical user interface.

VII. Conclusion

The paper presents a toolkit for generating sentences from context-free grammars. The toolkit supports sentence generation with coverage criteria, sentence enumeration and sentence analysis. It also contains a simple graphical user interface, through which the user can use the toolkit directly. The toolkit provides richer functionalities for sentence generation, enumeration and analysis than existing tools, thus is expected to have more applications.

In the future, we plan to study the unresolved problems mentioned in the paper and enrich the toolkit by adding more functionalities, such as a graphical parse tree editor.

Acknowledgment

The authors thank Yunmei Dong for his suggestions on developing the toolkit and on implementing, improving and experimenting with the sentence enumeration algorithm and its GUI.

References

[1] P. Klint, R. Lämmel, and C. Verhoef, "Towards an engineering discipline for grammarware," *ACM Transaction on Software Engineering and Methodology*, vol. 14, no. 3, pp. 331–380, 2005.

[2] R. Lämmel, "Grammar testing," in *Proceedings of Fundamental Approaches to Software Engineering (FASE'01)*, vol. 2029, 2001, pp. 201–216.

[3] U. o. C. Programming Languages Laboratory, "Context free grammar checker," http://smlweb.cpsc.ucalgary.ca/.

[4] A. Tarantini, "Forson," http://sourceforge.net/projects/forson/.

[5] P. Purdom, "A sentence generator for testing parsers," *BIT*, vol. 12, no. 3, pp. 366–375, 1972.

[6] L. Zheng and D. Wu, "A sentence generation algorithm for testing grammars," in *Proceedings of the 33rd Annual International Computer Software and Applications Conference (COMPSAC'09)*, vol. 1, 2009, pp. 130–135.

[7] T. L. Alves and J. Visser, "A case study in grammar engineering," in *Proceedings of the 1st International Conference on Software Language Engineering (SLE'08)*, vol. 5452, 2008, pp. 285–304.

[8] Y. Shen and H. Chen, "Sentence generation based on context-dependent rule coverage," *Computer Engineering and Applications*, vol. 41, no. 17, pp. 96–100, 2005, in Chinese.

[9] Y. Dong, "Counting and hierarchical lexicographic enumeration of CFL sentences," *Science in China (Series E)*, vol. 36, pp. 1375–1413, 2006, in Chinese.

[10] Y. Dong, "Linear algorithm for lexicographic enumeration of CFG parse trees," *Science in China (Series F - Information Science)*, vol. 52, no. 7, pp. 1177–1202, 2009.

[11] P. Dömösi, "Unusual algorithms for lexicographical enumeration," *Acta Cybernet*, vol. 14, no. 3, pp. 461–468, 2000.

[12] J. Earley, "An efficient context-free parsing algorithm," *Communications of the ACM*, vol. 13, no. 2, pp. 94–102, 1970.

[13] Y. Dong, K. Li, H. Chen, Y. Hu, R. Zhang, and R. Tang, "Design and implementation of the formal specification acquisition system saq," in *Proceedings of the Conference on Software: Theory and Practice, IFIP 16th World Computer Congress 2000*, 2000, pp. 201–211.

[14] L. Zheng and H. Chen, "A systematic framework for grammar testing," in *Proceedings of the 8th International Conference on Computer and Information Science (ICIS'09)*, 2009, pp. 1013–1019.

[15] Y. Dong, "Recursive functions of context free languages (I)-the definitions of CFPRF and CFRF," *Science in China (Series F)*, vol. 45, no. 1, pp. 25–39, 2002.

[16] Y. Dong, "Recursive functions of context free languages (II)-validity of CFPRF and CFRF definitions," *Science in China (Series F)*, vol. 45, no. 2, pp. 1–21, 2002.

[17] H. Chen and Y. Dong, "Towards practical computable functions on context-free languages," in *Proceedings of the 4th Annual Conference on Theory and Applications of Models of Computation(TAMC'06)*, vol. 3959, 2006, pp. 555–565.

[18] F. G. S. Richard A. DeMillo, Richard J. Lipton, "Hints on test data selection: Help for the practicing programmer," *IEEE Computer*, vol. 11, pp. 34–41, 1978.

[19] A. J. Offutt, A. Lee, G. Rothermel, R. H. Untch, and C. Zapf, "An experimental determination of sufficient mutant operators," *ACM Transactions on Software Engineering Methodology*, vol. 5, pp. 99–118, 1996.

[20] A. O. Bilska, K. H. Leider, M. Procopiuc, O. Procopiuc, S. H. Rodger, J. R. Salemme, and E. Tsang, "A collection of tools for making automata theory and formal languages come alive," in *ACM SIGCSE Bulletin*, 1997, pp. 15–19.

[21] A. Møler, "dk.brics.grammar," http://www.brics.dk/grammar/, 2008.

[22] A. Almeida, J. Alves, N. Moreira, and R. Reis, "CGM: A context-free grammar manipulator," in *Proceedings of Compiler, Related Technologies and Applications (CoRTA'08)*, 2008.