First Steps Towards Automatic Question Generation and Assessment of LL(1) Grammars

Ricardo Conejo(✉), José del Campo-Ávila, and Beatriz Barros

Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, Campus de Teatinos, 29071 Málaga, Spain
{conejo,jcampo,bbarros}@uma.es

Abstract. Automatic question generation and the assessment of procedural knowledge is still a challenging research topic. This article focuses on case of it, the LL(1) grammar design. This is a well known technique for construct a top-down parser. There are many tools that given a context-free grammar can construct the LL(1) tables, but they are not designed for assessment. This article describes an application that covers all the tasks needed to automatize the assessment process.

Keywords: Automatic assessment · Question generation · Procedural knowledge · Adaptive feedback · Top-down parsing

1 Introduction

Compiler construction is a compulsory subject in almost all computer science degrees. Here the student learn different algorithms, tools and methods necessary to understand how a compiler for a programming language is constructed [1]. A core part of compiler construction is the design of the language grammar and the construction of the parser. The LL(1) technique allows to construct efficient top-down parsers based on theoretical grounds, but it requires some conditions to be met for the grammar design. Let's introduce some concepts that are used in this article:

A context-free grammar (CFG) is defined as $G(N, T, P, S)$, where N is a set of *non-terminal* symbols, T is a set of *terminal* symbols, P is a set of *production rules* (of the form $A \rightarrow \gamma$, where A is called the *antecedent* and $\gamma \in (N \cup T)^*$ is called the *consequent*), and S is the *axiom*. The languages that can be generated by a CFG are called context-free languages (CFL).

LL(1) grammars are a subset of context-free grammar (CFG) that accomplish the LL(1) condition. There is a well-known algorithm to efficiently determine if a context-free grammar is LL(1) and construct its parsing table. It is based on the construction of the functions $FIRST$, $FOLLOW$ and the directive symbols of each production rule DS [1]. LL(1) languages are those context-free language that can be generated by an LL(1) grammar.

M. M. Rodrigo et al. (Eds.): AIED 2022, LNCS 13356, pp. 271–275, 2022.
https://doi.org/10.1007/978-3-031-11647-6_50

This article describes the implementation of a computer-based assessment application, constructed as a plugin of the SIETTE assessment system (see Sect. 2) that is able to automatically generate a random CFG and evaluate the student answers.

2 System Architecture

In order to assess the student knowledge and skills needed to design and implement LL(1) parsers for a given language, we have implemented a plug-in extension of the SIETTE assessment environment. Using this plug-in and some of the standard features of SIETTE, we are able to automatically generate different types of questions.

2.1 The SIETTE Assessment System

SIETTE [2] is a general-purpose automatic assessment environment that supports the generation of different question based on JSP templates, different types of questions and student answer interfaces; automated recognition of students' open answers based on regular expression patterns; and a flexible support of any other assessment requirement based on the construction of a plug-in extension. SIETTE implements the Classical Test Theory (CTT), Item Response Theory (IRT), Computer Adaptive Testing (CAT), and it provides built-in statistical and psychometric tools to analyze students, tests and questions results.

The student answer is given to SIETTE in a plain or structured text format. SIETTE recognizes whether the answer is correct using a pattern matching process. Patterns are provided by the teacher and the matcher algorithm is implemented as a plug-in. There are some default matcher plugins that are already implemented in SIETTE. On of them is the *SIETTE regular expression* that allows to recognize the student answer based on a regular expression pattern provided by the teacher or, in this case, automatically generated.

2.2 Automatic Generation of Context-Free and LL(1) Grammars

One of the first challenges of this project is to define a way to generate small context-free languages that can be used to pose questions to students. The alphabet of these languages (*terminal symbols*) is restricted to lowercase letter in order to be easy to write it in text format. *non-terminal symbols* are written using uppercase letters. The *axiom* of the grammar is always the *non-terminal symbol* that appears on the left hand side of the first rule.

Small context-free grammars can be generated just by setting the antecedent and a random length string that combines terminal and non-terminal symbols. This strategy requires validating the generated grammar and repeating the process until a correct grammar is obtained.

On the other hand, a well defined context-free grammar can be generated based on composition of "building block" grammars. The building blocks are

tiny context-free grammars with just two or three production rules. Some of them are listed below:

$$\begin{aligned} A &\rightarrow Aa \\ A &\rightarrow a \end{aligned} \tag{1}$$

$$\begin{aligned} A &\rightarrow aAb \\ A &\rightarrow ab \end{aligned} \tag{2}$$

The plug-in defines some building block grammars, but they can be easily extended as needed. Using this building blocks we apply a composition rule just by replacing a *terminal* symbol with a *non-terminal* symbol of another building block. For instance, combining Block 1 and Block 2 in this order will generate the following grammar:

$$\begin{aligned} A &\rightarrow AB \\ A &\rightarrow B \\ B &\rightarrow aBb \\ B &\rightarrow ab \end{aligned}$$

On the other hand, combining Block 2 and Block 1 grammars can give one of these four grammars:

$$\begin{array}{cccc} A \rightarrow BAb & A \rightarrow BAb & A \rightarrow aAB & A \rightarrow aAB \\ A \rightarrow Bb & A \rightarrow Bb & A \rightarrow aB & A \rightarrow aB \\ B \rightarrow Ba & B \rightarrow Bb & B \rightarrow Ba & B \rightarrow Bb \\ B \rightarrow a & B \rightarrow b & B \rightarrow a & B \rightarrow b \end{array}$$

Note that there are four possible ways to combine Block 2 and Block 1 grammars, because we have two alternative options: (1) In Block 2, there are two terminal symbols, so we have two options to replace a *terminal* with a *non-terminal* of the second grammar; (2) we have to choose if the *terminal* symbols of the resulting grammar are the same or not. Nevertheless, without loss of generality, we can always assume that *terminals* are different, and at the end of the generation process, two or more symbols can be merged as a single *terminal*. That is, in the last example, options 2 and 4 can be obtained from options 1 and 3 just by considering that *terminal* a and *terminal* b are the same. This process is delayed until we finish the combination process.

Thus, a context-free grammar can be randomly generated by selecting the building blocks to combine, the number of combinations to apply (or alternatively the number of production rules in the final grammar) and the final number of *terminal* symbols (which will randomly merge two symbols until the desired number of *terminal* symbols is met).

Finally, a validation and refinement process is triggered to eliminate unused rules or symbols, and/or duplicate rules, to guarantee that the context-free grammar is correct and that the FIRST and FOLLOW sets can be effectively calculated.

2.3 Automatic Construction of LL(1) Parsers

Given a context-free grammar it is always possible to compute FIRST and FOLLOW functions and obtain the directive symbols of each production rule [1]. The result of these functions are a set of symbols. Determining if a context-free grammar accomplishes the LL(1) condition depends on these sets.

The system requires a student response by asking to type the symbols in the set. The student can shuffle the order of symbols in the set, but the pattern will recognize the answer anyway. Figure 1 presents a composed question where a common grammar has been generated, and some questions about FIRST and FOLLOW sets are posed. Each question is evaluated independently.

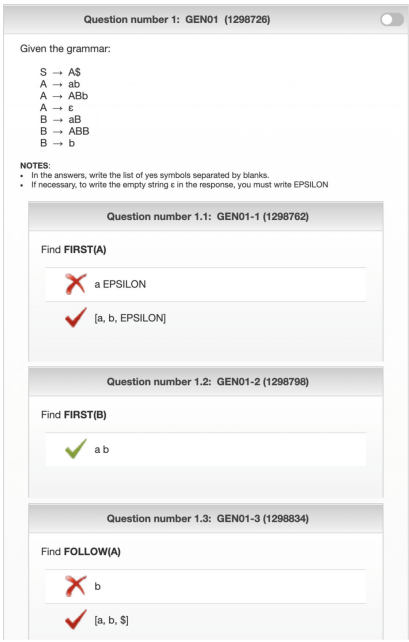

Fig. 1. A composed SIETTE question about FIRST and FOLLOW sets

3 Conclusion

The application described in this article provides a way for the student to enhance the practice of design of the LL(1) context-free grammars. Although the generation of an LL(1) grammar and the recognition of grammar equivalence are unsolvable issues in the general case, a heuristic approach can provide a practical solution for assessment purposes.

The application has been designed and used for formative and summative assessment. It includes automatic recognition of student answers and personalized feedback. The application is embedded in the SIETTE system, which provides additional features that can be used, such us adaptive question selection, scoring procedure selection, access control, etc. Question difficulty can be controlled by means of the number of building block grammar combinations, but it can also be obtained empirically through SIETTE question calibration and learning analytic tools.

We do not claim that the system itself is responsible for the increase in the student scores, but the data obtained from the students that have used the system shows that it helps them to practise and be aware of their progress.

Links to on-line assessments

FIRST and FOLLOW: https://www.siette.org/siette?idtest=631978
LL(1) analysis: https://www.siette.org/siette?idtest=633742
LL(1) table construction: https://www.siette.org/siette?idtest=525382
LL(1) grammar design: https://www.siette.org/siette?idtest=633700

References

1. Aho, A.V., Lam, M.S., Sethi, R., Ullman, J.D.: Compilers: Principles, Techniques, and Tools, 2nd edn. Addison-Wesley Longman Publishing Co., Inc., Boston (2006)
2. Conejo, R., Guzmán, E., Trella, M.: The SIETTE automatic assessment environment. Int. J. Artif. Intell. Educ. **26**(1), 270–292 (2015). https://doi.org/10.1007/s40593-015-0078-4