Syntax Error Recovery via Context-GFree Languge Reachability

Semyon Grigorev
St. Petersburg State University
198504, Universitetsky
prospekt 28
Peterhof, St. Petersburg,
Russia
rsdpisuy@gmail.com

ABSTRACT

Syntax error recovery is an important functionality of such development tools like IDEs and language services. For practical reason it is necessary to find best recovery in terms of minimization of the original string trnsformations. One of the formal view on this problem is a language editing distance problem: it is necessary to find string in the given language such that editing distance betwitn this string and the given string is a minimal. We show that another formalization can be constucted upon another classical problem which names a context-free language path querying (CFPQ). To find the best recovery, it is necessary to find path with minimal weight in the special graph such that labels along this path form a string in the given context-free language. Practical parsing algorithm, such as generalized LL or LR can be used for CFPQ. Moreover, GLL-based algorithm for CFPQ can create an SPPF which make it usefull for parsing. That meens algorithms for CFPQ can be used for parsing with syntax error recovery. Thus we show how to bridge the gap detween syntax error recovery and CFPQ.

Keywords

Syntax analysis, parsing, error recovery, CFL reachability, Generalized parsing, GLL

1. INTRODUCTION

Modern tools for development (IDEs, language services, etc) provide much more information "on the fly"—during code typing. The problem is that it is necessary to build parsing tree to get such information. But code not contimiously correct. To be able to provide all possible information even for parlially incorrect code syntax error recovery mechanism is requred.

General goal of error recovery is to find syntactically correc string which a close to the given string as mach as possible, and provide parsing result of this string. In this terms error recovery problem is a language editing distance problem: for the given language and the given string it is necessary to find string which is in language such that editing distance between this atring and the given is minimal.

On the other hand, one can treet input as a linear graph where token is an edge. Similar view, for exmaple, was proposed by Jonstone at Parsing@SLE !!! on multivariant tokenization parsing. After thet one can precompute all possible editing steps statically and add appropriate edges into input graph: epsilon edges for delition, parallel edges with tokens for repacement, and loop edges for insertion. Additional edges should be weighed alike in classical editing problems. After that the problem is to find shortest path from start vertex to final such that string along this path is in the given language.

Williams on cliques — CFL distance to clique. Static graph. Can we do it in dynamic?

GLL [2] for CFPQ with structural representation of result [1].

1. bridging the gap between ....
2. We propose the way to utilize CFPQ algorithms
3. We evaluate ...

2. ERROR RECOVERY TO CFL REACHCBILITY

Editing distance: to find string which is in the given lznguage, such that editing distance is minimal.

2.1 Preliminaries

Token is a triple (aka edge).

2.2 Example

We start with example. Suppose, the given language is a simple arithmetic language which is specified by grammar G_1 :

$$s \rightarrow \text{'('}\ s\ \text{')'} \mid NUM \mid s\ \text{'+'}\ NUM.$$

The string to parse is ". Firstly we convert this string into linear graph and fix start and final vertices.

After that we add edges for all possible atomic editing steps (replacement, deletion, insertion).

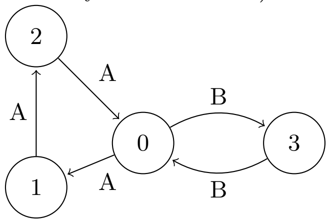
Now we can solve CFL reachcbility problem and get structural representation of result, or derivation tree for the string which are correst and closest to the input string.

ISBN 978-1-4503-2138-9.

DOI: 10.1145/1235

2.3 Algorithm

Now we explain how it should works in general case. Additional edges with error markers goes forward and with all tokens, goes in the its start vertex (as a result we have loops, but in they can be limited).



It is not necessary to create this graph explicetely. We just can specify generation function which should generate appropriate set of tokens for the given position.

Number of edges may be optimized by filtering with FIRST(k), it can reduce false attempts.

One can customize weight function and maximal length of insertions.

GLL operates woth queue of descriptors. To optimize order of descriptors processing we introduce priority queue for descriptors. How to choose priority function? — ordered tuples!

In the general case we can get SPPF rather single trrr, so it may be necessary to choose the best tree from SPPF after parsing finish.

Priority is a number of additional edges (not from the original input) in processed prefix. Suffix length.

3. EVALUATION

We implement proposed algortihm as a modification of solution which is proposed in [**?**].

The goal of evaluation is to compare performance of GLL and GLL with error recovery.

Simple arithmetic expressions.

Grammar.

Input generation. Without tokenezation.

Timing. Original GLL. GLL with error recovery.

Cases: without errors, error in the end of file, in the middle, abd in the start, several errors.

4. DUSCUSSION AND CONCLUSION

We propose a way to reduce error recovery problem to a special case of context-free reachability problem. This way we demonstrate deep interconnection betwee these two problems.

We provide the algorithm and its implementation witin evaluation on some preliminary data. We show that !!!!

Future research includes both theoretical and paractical parts. In the practical way we should provide high-quolity implementation of proposed algorithm, and evaluate it on well-known datasets. For example, on data from the BlackBox project [**?**]. Also we should compare our results with other similar solutions such as [**?**] and [**?**].

In theory we should find interconnections between CFL reachability, CFPQ, CFL editing distance end error recovery in order to unify methods and share ideas and solutions.

5. REFERENCES

[1] S. Grigorev and A. Ragozina. Context-free path querying with structural representation of result. In *Proceedings of the 13th Central & Eastern European Software Engineering Conference in Russia*, CEE-SECR '17, pages 10:1–10:7, New York, NY, USA, 2017. ACM.

[2] E. Scott and A. Johnstone. Gll parsing. *Electronic Notes in Theoretical Computer Science*, 253(7):177–189, 2010.