On Development of Static Analysis Tools for String-Embedded Languages

Marat Khabibullin
St. Petersburg Academic University
194021, Khlopina Str 8/3
St. Petersburg, Russia
maratx387@gmail.com

Andrei Ivanov
St. Petersburg State University
198504, Universitetsky prospekt 28
Peterhof, St. Petersburg, Russia
ivanovandrew2004@gmail.com

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

ABSTRACT

Some programs can produce string expressions with embedded code in other programming languages while running. This embedded code should be syntactically correct as it is typically executed by some subsystem. A program in Java language that builds and sends SQL queries to the database it works with can be considered as an example. In such scenarios, languages like SQL are called string-embedded and ones like Java – host languages.

In spite of the fact such an approach of programs building is being replaced by alternative ones, for example by ORM and LINQ, string-embedding is still used in practice. Development and reengineering of the programs with string-embedded languages is complicated because the IDE and similar tools process the code embedded in strings as host language string literals and cannot provide the functionality to work with this code. To facilitate the development process, string-embedded code highlighting, completion, navigation and static errors checking would be useful. For the purposes of reengineering, embedded code metrics computation would be helpful.

Currently existing tools to string-embedded languages support only operate with one host language and a fixed set of string-embedded ones. Their functionality is often limited. Moreover, it is almost impossible or requires a substantial amount of work to add a support for both new host and string-embedded language. Attempts to extend their functionality often result in the same problem.

In this paper we present the platform which can be used for relatively fast and easy building of endpoint tools that provide a support for different string-embedded languages inside different host languages. The tools built for T-SQL and arithmetic expressions language embedding in C# are demonstrated as the examples of how the platform can be used.

ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

Keywords

String-embedded language, integrated development environment, IDE, approximation, control flow graph, CFG

1. INTRODUCTION

When writing certain kinds of programs on some programming language one frequently needs to construct a code on some other language. The program written on Java, C# or PHP constructing SQL queries and sending them to the database can be considered as an example. In such a program SQL queries can be constructed as string literals, both by writing the entire query inside the literal or by forming final query dynamically, i.e. by combining parts using string operations (concatenation, replace) and common language constructions (loops, conditional expressions). It is important to note that in the later case the parts are no need to be correct SQL expressions. Other examples of the approach in question include forming JavaScript code inside Java when writing web-applications, building of dynamic SQL queries using Dynamic-SQL, xml-files generation, etc. The language that manipulates strings containing code is called the host language. The language which code is written inside string literals is called string-embedded language.

It is useful to have the ability to perform static analysis on string-embedded languages. On the one hand, such analysis would make it possible to support string-embedded languages in IDEs by making syntax highlighting, static errors checking and other functions, previously only available for host languages, become available for string-embedded ones right inside string literals. On the other hand, programs writing approach under consideration is more and more frequently replaced by more advanced approaches. For example, speaking about embedded SQL such approaches include ORM (object-relational mapping) and LINQ (language integrated query). In spite of this fact, there exist many programs where string-embedded languages are used. These programs are still in use so they need a support and maintenance. In particular reengineering can be performed for them.

Static analysis can be useful during the reengineering process at least in two ways. The first one is extracting some information about string-embedded code. Examples include estimating embedded program's structural complexity using such metrics as cyclomatic complexity [**?**], or, in the case of embedded SQL, estimating tables usage frequency in queries to check the possibility of restructuring the database the

embedded program works with. The second way is automated transformation of the embedded code to move from the string embedding approach to alternative ones (for example, to move from string-embedded SQL to LINQ). It is important to note that the possibility of such transformations in general case is questionable even in theory. However, in particular cases such transformations can be performed automatedly.

The problem of string-embedded languages static analysis poses a number of challenges. Firstly, not all string expressions in a program contain embedded code and analyzer must be able to differentiate one expressions from another. Secondly, as it was mentioned before, string expressions can be formed dynamically. Thus, we need to construct them according to the operations they are formed with before we can start the analysis itself. Thirdly, the program in the host language in general case may produce the set of strings with embedded code so the analyzer must be able to build this set and represent it in a way convenient for the following analysis. Finally, the lexical and syntax analysis algorithms that can work not only with a single string but with the set of strings are needed.

Despite the utility of string-embedded languages static analysis, existing tools are mostly intended to support embedded code in IDEs and can hardly be used for reengineering problems solving. Moreover, the majority of the tools work only with specific host and string-embedded languages.

In this paper we present the platform for string-embedded languages support, which is a part of the YaccConstructor [**?**] project. The YaccConstructor is devoted to the experiments in the field of static analysis. The platform under discussion is intended as a basis for endpoint tools, making the process of its creation relatively easy and fast. Endpoint tools can be created for different host and string-embedded languages. The platform is designed to be extensible, so it is able to add different functions based on string-embedded language static analysis (from the syntax highlighting to metrics computation). This paper describes the platform and shows how to create endpoint tools.