EN 600.425 DECLARATIVE METHODS - TERM PROJECT PROPOSAL

Guoye Zhang, Qiang Zhang

March 6, 2017

1 Introduction

For the term project, we have two choices in mind and we have not decided to narrow down to either of them.

2 Toward Compiled Little Languages

Most current little languages are represented in a textual format, ranging from the most prominent languages such as HTML/CSS and SQL to particularly domain-specific ones like CNF for SAT solvers. Despite the indubitable human-readability such textual little languages afford, correct processing of them, on the other hand, can be a major chanllenge for programmers. Every now and then, programmers attempt to use regular expression to parse HTML which is not a regular language; countless security vulnerabilities arise from lack of escaping; even large CDN provider Cloudflare recently leaked a large amount of customer information due to unclosed HTML tag. It is time to reconsider using plain text within communications between machines.

We hope to build a compiler toolkit for little languages which provide fast and secure manipulation of data and semantics, as well as translation between binary and textual representations.

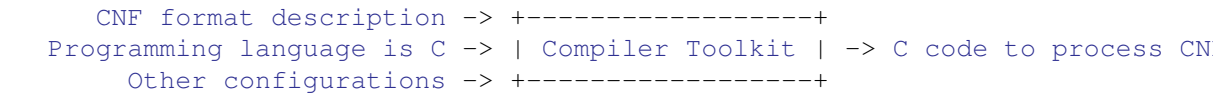
Our idea is inspired by two popular projects — LLVM and Protobuf. LLVM is a collection of modular and reusable compiler toolchain for general purpose languages. It has many front-ends, optimizers, and back-ends, and it pioneers the concept of intermediate representation (IR). Instead of machine code, our compiler produces something similar to IR that represents data and semantics. Since we are targeting little languages, we intend that back-end be provided by the user.

Protobuf is an extensible mechanism for serializing structured data. It is language and platform neutral, which is achieved by generating code in any supported programming languages and inserting them into programs. However, Protobuf does not allow anything more complex than the combination of basic data types. We hope to use a similar language-independent approach, but break some of Protobuf's current limitations. Beside data, we are building support for semantics into the serialized format.

Workflow of using our compiler toolkit:

1. Write the description for your little language
2. Choose a supported programming language to write your program
3. Use our toolkit to generate code to process your little language
4. Add the generated code to your program

An example:



We intend to exploit the full power of binary representation to provide not only efficient encoding of little languages but also simpler interfaces for programmers to work with them, in hope of enabling more complex operations like comparison and optimization.

3 Generalized GUI for Describing Declarative Languages

Inventing and validating a declarative language is hard. A large percentage of them don't even have any documentations. It is error-prone to write code for printing and parsing declarative languages without formal specifications.

We are hoping to build a graphical user interface for describing declarative languages. We use GUI to create and visualize the rules for the language, then automatically detect conflict and ambiguities within rules.

Using those rules, we can process and visualize any expression in this language. We can even produce parser in supported programming languages.