Automating Code Magnet Generation

Julia Dana

Contents

1 Introduction

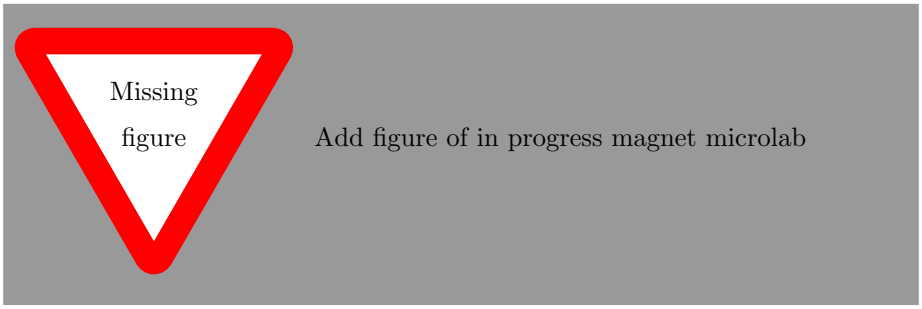
1.1 The Overview: Easier Creation of Code Magnet Microlabs

The purpose of this project is to assist in the creation of code magnet microlab assignments for WAGS by creating magnets from a completed solution file. This is accomplished in a manner that supports multiple programming languages, and allows additional languages to be added with minimal configuration. Additional tools that support this idea of easier creation of assignments are also included, such as an automated interaction with the WAGS website to create assignments. Also, this project defines new formats for the specification of magnets, including object and JSON.

Improve this sentence, implement XML

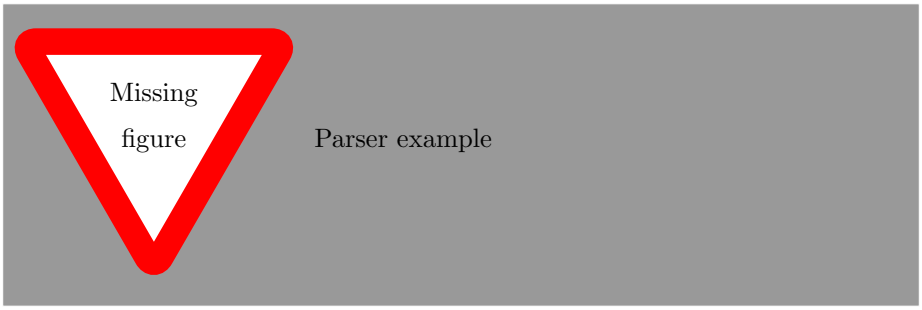
1.2 The Context: What is WAGS?

WAGS (Web Automated Grading System) is an ongoing project of Appalachian State University. It is an online tool for microlabs. Microlabs are short, 5-10 minutes hands-on activities that are intended to be done as a part of a regular (i.e. not lab) class session to reinforce the concepts that are currently being covered. There are multiple types of microlabs provided by WAGS, but the one that this project is interested in is code magnet microlabs. These are microlabs where the student is given pieces of code (code magnets) and has to choose the correct ones and order them correctly to "write" a solution to the microlab.

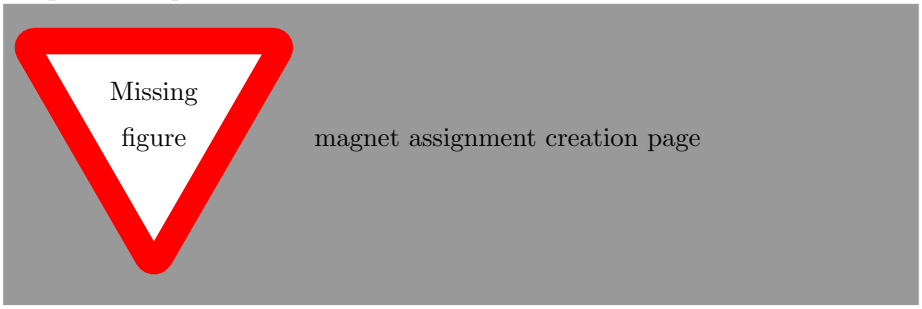


1.3 The Problem: Brittle Input

The problem is that creating these magnet microlab assignments on the WAGS website is a somewhat painful process. The current parser creates a magnet per line, and this can result in having to format the input file so that is no longer the same as a solution file, and is indeed no longer valid for its language.



If your input cannot be handled by this brittle parser. WAGS does provide a manual input for magnets. However, using this manual input requires magnets to either be entered one at a time to the magnet creation wizard, or for the user to directly type the final magnet (including HTML escape sequences) to the parsed output section.



1.4 The Solution: Parsing by Grammar

The solution to this is to use a more robust parser that is based on the grammar of a language, rather than simply splitting on newlines. This project will be using the ANTLR4 parser generator[2] and grammars for common languages[1].

Discuss grammars

Describe ANTLR4

2 Development results and future extensions

2.1 What It Does: Internal Functions

2.1.1 Parses by Grammar

2.1.1.1 Java

2.1.1.2 Python

2.1.2 Improved Represention of Magnets

2.1.3 Alternative Magnets

2.2 What It Does: External Functions

2.2.1 Interaction With WAGS Website

2.2.2 GUI Tool

2.3 What It Could Do: Future Extensions

2.3.1 Supporting New Languages

2.3.2 Automatic Generation of Alternative Magnets

3 User Guide

3.1 Quick Start - CLI Tool

3.2 Quick Start - GUI Tool (tent.)

3.3 Explanation of Output

3.4 Automatic Upload to WAGS Website

3.5 Creating Alternative Magnets

3.6 Controlling the Output (tent.)

4 Developer Notes

4.1 The Environment: Languages and Libraries Used

4.2 The Design

4.3 The Configuration: Specifying Magnet Sections

4.4 The Expansion: Adding a New Language

4.4.1 New Grammar Specification

A new G4 file can be added to the system. However, whitespace is require to go on channel 1, or your magnets will not have any whitespace, and you might

get things like “publicclassMyClass”.

Final export of bibliography from Mendeley and link it

Make sure citations and bibliography are styled correctly

References

[1] antlr/grammars-v4.

[2] Terence Parr. *The Definitive ANTLR 4 Reference*. The Pragmatic Programmers, 2012.