CS 3860 Report

Siraphob (Ben) Phipathananunth

May 1, 2022

1 Introduction

This semester I carried out research in computer science under the supervision of Dr. Daniel Balasubramanian. The project focused on improving the developer experience for the Formula[5] language by creating a syntax highlighter and integrating with editors such as Emacs and VS Code. By achieving this, the work would make it easier for developers to write and read Formula code, since the syntax highlighting would allow them to distinguish semantic elements at a glance, and features such as auto-completion and linting hints in their editor would let them navigate and work with Formula code more easily.

Traditionally, adding IDE-like features for a language to an editor was a labor-intensive process. For many popular languages such as Java, only a handful of editors had support for advanced navigation and code manipulation systems. The system would have to parse the source code, extract semantic elements such as function and variable declarations and usage before finally reporting back to the user. In addition, the tools would be integrated specifically for that editor, making it difficult to reuse in other settings.

2 Our contributions

To make progress on these tasks more effectively, we used Tree-sitter[1], a parser generator tool that generates a GLR parser from a description of the language grammar. The generated parser is *incremental*, meaning that when an edit to the file is made, subsequent parses of the whole program are fast, thus it can run on every keystroke in an editor setting. It also has *error recovery*, so a useful syntax tree can be generated even in the presence of syntax errors. Furthermore, the parser has no external dependencies and is written in C, and the tree can be processed in many different languages through FFI bindings.

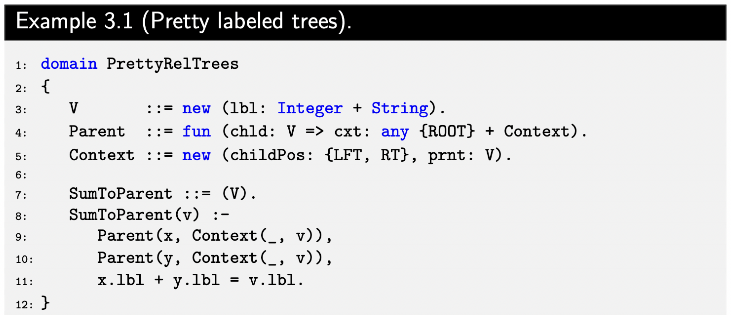
From February 8–11 I worked on a Promela grammar for Tree-sitter[10], the language used by the Spin model checker.[3] This was done to gain familiarity with the tool while providing useful output at the same time. From February 17–March 30 I worked on a Formula grammar for Tree-sitter.[6] The status of Formula grammar is complete and has been comprehensively tested on every Formula file in the compiler test folder. The grammar closely follows the ANTLR grammar given in the source code for Formula.[4] All the work done was released under a permissive MIT license.

This work also served as the foundation on which to build other tools. Alongside the focus on Formula, I wrote two blog posts[8][7] and created a semantic linter for the Nix language.[2][9] I also worked with Stephen Johnson from Vanderbilt's ISIS institute who helped with finding issues in the grammar and discussed improvements to the grammar that would make error recovery more robust. Johnson also created a plugin for VS Code that uses the semantic information returned by a WebAssembly build of the grammar to provide on-hover hints and semantic highlighting.

Some illustrative examples are shown below. In Figure 1 we see a typical example of how Formula code would be presented with very sparse highlighting. In Figure 2 we see the syntax highlighting generated by Tree-sitter using *queries* on the parse tree (queries are used to assist in tagging and filtering for syntax trees of interest.) Record identifiers are red and italicized, type names are green, enums are gold, and so on. The colors can be adapted to various color schemes appropriate for context in which the code is to be displayed. Finally, in Figure 3 we see integration of the work into an existing editor, Emacs.[11]

3 Future work

There are multiple possible directions for future work. The grammar could be modified to make the resulting trees less noisy by hiding nodes. Editor integration could be improved to take into account common indentation patterns in Formula. On the user's side, tools such as auto-formatters, linters and context-sensitive rewriters could be created to help standardize the format and style of Formula code. A re-implementation of the Formula grammar also brought to light inherent design choices in the grammar that can hinder error recovery, such as the use of the character `.` in marking the end of sentences which can lead to ambiguity when it is used next to a dotted identifier such as `foo.bar`. With this knowledge one can interactively and incrementally explore variations of the Formula grammar with more robust properties.

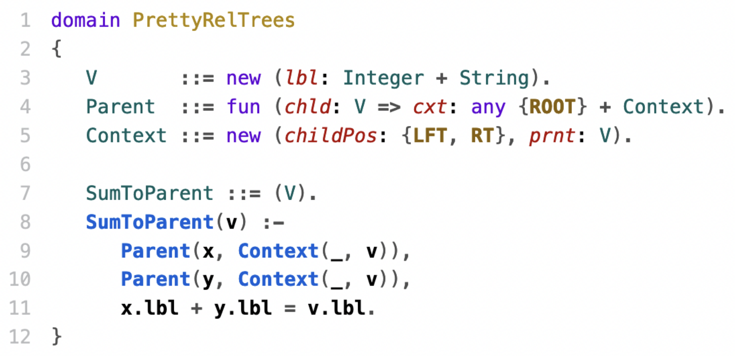


Example 3.1 (Pretty labeled trees).

```
domain PrettyRelTrees
{
   V       ::= new (lbl: Integer + String).
   Parent  ::= fun (chld: V => cxt: any {ROOT} + Context).
   Context ::= new (childPos: {LFT, RT}, prnt: V).

   SumToParent ::= (V).
   SumToParent(v) :-
      Parent(x, Context(_, v)),
      Parent(y, Context(_, v)),
      x.lbl + y.lbl = v.lbl.
}
```

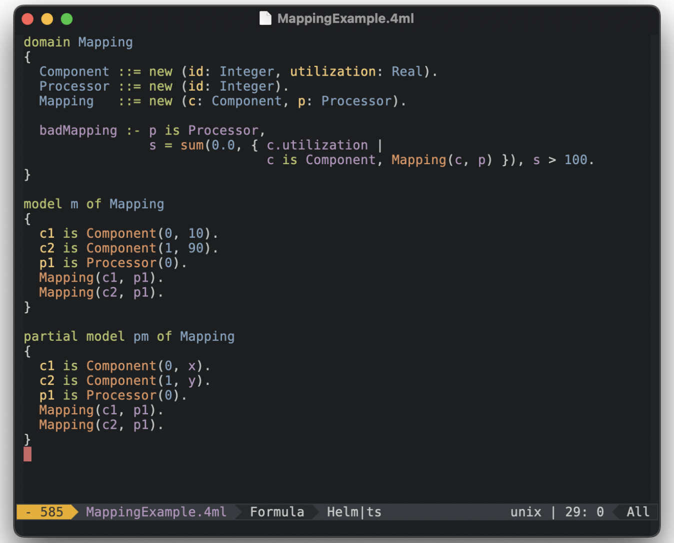
Figure 1: Code block example from the Formula manual.



```
domain PrettyRelTrees
{
   V       ::= new (lbl: Integer + String).
   Parent  ::= fun (chld: V => cxt: any {ROOT} + Context).
   Context ::= new (childPos: {LFT, RT}, prnt: V).

   SumToParent ::= (V).
   SumToParent(v) :-
      Parent(x, Context(_, v)),
      Parent(y, Context(_, v)),
      x.lbl + y.lbl = v.lbl.
}
```

Figure 2: The same code with our syntax highlighting applied.



MappingExample.4ml

```
domain Mapping
{
  Component ::= new (id: Integer, utilization: Real).
  Processor ::= new (id: Integer).
  Mapping   ::= new (c: Component, p: Processor).

  badMapping :- p is Processor,
                s = sum(0.0, { c.utilization |
                               c is Component, Mapping(c, p) }), s > 100.
}

model m of Mapping
{
  c1 is Component(0, 10).
  c2 is Component(1, 90).
  p1 is Processor(0).
  Mapping(c1, p1).
  Mapping(c2, p1).
}

partial model pm of Mapping
{
  c1 is Component(0, x).
  c2 is Component(1, y).
  p1 is Processor(0).
  Mapping(c1, p1).
  Mapping(c2, p1).
}
```

```
- 585  MappingExample.4ml  Formula  Helm|ts                unix | 29: 0  All
```

Figure 3: Integration into the Emacs text editor.

References

[1] Max Brunsfeld. *Tree-sitter webpage*. URL: `https://tree-sitter.github.io/tree-sitter/` (visited on 04/28/2022).

[2] Eelco Dolstra. *The purely functional software deployment model*. Utrecht University, 2006.

[3] Gerard J. Holzmann. *The SPIN Model Checker - primer and reference manual*. Addison-Wesley, 2004, pp. I–XII, 1–596. ISBN: 978-0-321-22862-8.

[4] Ethan K. Jackson. *ANTLR grammar for Formula*. URL: `https://github.com/VUISIS/formula-dotnet/blob/a651e31b7c4ac096fec705f771509bb6462a8d8a/Src/Core/API/Parser/FormulaParser.g4` (visited on 04/28/2022).

[5] Ethan K. Jackson and Wolfram Schulte. "FORMULA 2.0: A Language for Formal Specifications". In: *Unifying Theories of Programming and Formal Engineering Methods - International Training School on Software Engineering, Held at ICTAC 2013, Shanghai, China, August 26-30, 2013, Advanced Lectures*. Ed. by Zhiming Liu, Jim Woodcock, and Huibiao Zhu. Vol. 8050. Lecture Notes in Computer Science. Springer, 2013, pp. 156–206. DOI: `10.1007/978-3-642-39721-9_4`. URL: `https://doi.org/10.1007/978-3-642-39721-9/_4`.

[6] Siraphob (Ben) Phipathananunth. *Formula grammar for Tree-sitter*. URL: `https://github.com/siraben/tree-sitter-formula` (visited on 04/28/2022).

[7] Siraphob (Ben) Phipathananunth. *How to write a linter using tree-sitter in an hour*. Mar. 22, 2022. URL: `https://siraben.dev/2022/03/22/tree-sitter-linter.html` (visited on 04/28/2022).

[8] Siraphob (Ben) Phipathananunth. *How to write a tree-sitter grammar in an afternoon*. Mar. 1, 2022. URL: `https://siraben.dev/2022/03/01/tree-sitter.html` (visited on 04/28/2022).

[9] Siraphob (Ben) Phipathananunth. *Nix-lint: a semantic linter for Nix using tree-sitter*. URL: `https://github.com/siraben/nix-lint` (visited on 04/28/2022).

[10] Siraphob (Ben) Phipathananunth. *Promela grammar for Tree-sitter*. URL: https://github.com/siraben/tree-sitter-promela (visited on 04/28/2022).

[11] Siraphob (Ben) Phipathananunth. *siraben's dotfiles*. URL: https://github.com/siraben/dotfiles/blob/59032d5c383eb9064b2a430313d0490e53678cb3/emacs/.emacs.d/modules/siraben-packages.el#L287 (visited on 04/28/2022).