coala

Ocaml Support In Coala

October 2022

—

Yashashwee Chakrabarty : 2022MCS2057
Dhananjay Kumar Jha : 2022MCS2059
Vatsal Agarwal : 2022MCS2056

Overview

COALA:

Coala is a free and open-source language independent analysis toolkit, written in Python. The primary goal of Coala is to make it easier for developers to create rules which a project's code should conform to. Coala emphasizes reusability and pluggability of analysis routines. Coala comes with support for plenty of languages out of the box. Following is an exhaustive list of languages coala supports according to their documentation:

Languages coala provides algorithms for		
C	Latex	SQL
C++	Lua	Stylus
C#	Markdown	Swift
CMake	Matlab/Octave	TypeScript
CoffeeScript	Natural Language (English)	Verilog
CSS	Perl	VHDL
Dart	PHP	Vimscript
Fortran	Python 2	XML
Go	Python 3	YAML
Haskell	R	
HTML	reStructured Text	
Java	Ruby	
JavaScript	Scala	
JSP	SCSS	
Julia	sh & bash scripts	

We use this as a proof that the feature(s) that we shall implement here are not already available. You can find this documentation here: (https://github.com/coala/coala-bears#languages-supported)

BEARS:

Bears are interdependent functional units in the coala ecosystem that perform a certain analysis task on your code. Our main proposal would be building a new bear to support Ocaml. Bears can be language dependent as well as language independent.

OCAML:

OCaml is a strictly evaluated functional language with some imperative features. It is strongly and statically typed, but instead of using manually written type annotations, it infers types of expressions using Hindley-Milner algorithm. It makes type annotations unnecessary in most cases, but can be a major source of confusion for beginners. It is this confusion that we aim to reduce.

Goals

1. Add Ocaml to the core list of languages in Coala.
2. Add native helper bears to automatically add comments and recommendations based on the following ocaml “best practices” presented in the following doc (https://wiki.xenproject.org/wiki/OCaml_Best_Practices_for_Developers).

3. Add a few efficiency recommendation bears, such as suggesting when to use standard module functions in place of ones typed by the user.
4. Implement a Global Bear which scans the whole code, to identify which dependencies it has and generates the requirement files which will help user to deal with the dependencies before running the program.

Specifications

Since we are adding a new language to Coala. We would need to make a few changes to the core parsing library to understand and tokenize Ocaml code. From there we would need to create the different bears that would analyze a piece of ocaml code to add comments or syntax recommendations for the code.

Contingency

I. In Case Work isn't enough

Would fix a bug in the core library. Tentatively one of these:

(https://github.com/coala/coala/issues/3912)

II. In case work is too much

Reduction in the number of bears (including the global bear for the dependencies) would be the only option. Although the addition of Ocaml as a core language would still be the top priority.

Note: Contingencies are suggested only for extreme cases where all assumptions made to create this document have failed. For the most part we believe this should be enough.