March 26, 2018

Personal Details

Name : Utkarsh Sharma
Address : 1 Saint Michael's Road, #16-02 One Saint Michael's, Singapore 328006
Email ID : f20160600@goa.bits-pilani.ac.in
Contact Number : +91 6383999427
Github : https://github.com/Utkarsh1308
LinkedIn : https://www.linkedin.com/in/utkarsh-sharma-375159178/
University Information : BITS Pilani KK Birla Goa Campus
Degree : Msc Biological Science + BTech Chemical Engineering
Graduation Year : 2021

Developing Environment : Ubuntu, Windows

Project Summary

Coala helps detect and fix errors in code with the help of bears. Coala converts these fixes provided by different bears to its own Diff class instance which helps the user choose the appropriate action by reviewing the fix and the display results. But not every issue in the code can have just a single correct solution. The fix can and should also be modifiable by the preferences of the user. My Google Summer of code project basically aims on achieving the following things-

1. Offer multiple diffs, i.e. multiple different fixes to the same issue. Giving the user a choice to pick the most desirable one.
2. Produce interactive diffs that enable the bears to incorporate input from the user into the diffs offered by using a templating mechanism.

Related Issues of Interest:

1. Result: We want better diffs
2. Allow bears to pass their own applicable actions
3. KeywordBear: Add support for suggesting substitutions
4. ApplyPatchAction / patch conflict: Ask user whether to place patch with git-like conflict marker into the affected file
5. Add `next_action` behavior in Result Action

Related PRs of Interest:

1. Result.py: Add "actions field"
2. Ignore Action: Remove action after comment added
3. Add Result.actions and ResultAction.next_action fields

Benefits to the community:

Coala is a popular open source programming language analysis tool which helps companies and individuals set the rules in which they want their code to conform to and automatically detect and fix errors. Any keen student or professional interested in minimizing their workload and work more efficiently can clone coala from github or install it through pip and thus get to know about the organisation.

How did I get to know about COALA ?

I started coding in python when I was in my first year. I loved the idea to code applications and run them on my own laptop ! Github provides you a platform for recognizing your work. This year, during my first semester, I got to know coala through the GSOC program. I got hyper excited when I got to know that coala was coded in python. So back in August I started contributing to coala and got really hyped up by the code which used so many new technologies which I hadn't covered even after my one year experience in the field. I got to know about yaml and how it's used to

automate gitignore files, I also helped improve the coala projects site where the gsoc projects are listed by adding support for multiple status labels and helped rewriting filters to use dicts instead of lists. I learnt a lot more about python and web development contributing to the coala codebase and plan to continue doing so.

My Contributions -

Issues Generated -

Typo in commit_based_perftest.md
Fix mistake in Writing_Good_Commits.rst

Merged Pull Requests -

coala_main.py: Change variable name and value
Add .gitignore to coala/cEPs
Flags: Support for multiple status labels
FilterHelper.py: Rewrite filters
Writing_Good_Commits.rst: Fix Mistake
coala_color_svg: Add viewbox attribute
SASS.py: Add language definition for SASS

Deliverables :

Bears can pass their own applicable actions :

Although this is a part of a different GSOC Project, solving this would in turn help improve diff handling.

To achieve this functionality, we need to add a field inside Result which is a list of ResultAction instances

```
Result.actions = XYZAction(...)
```

And the bear would generate a result with the possible actions. It's possible to set even multiple ones

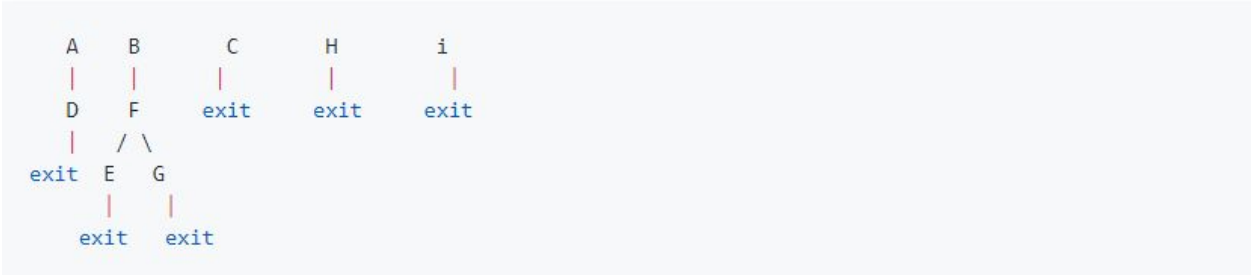
```
my_result = Result('Something went wrong')
my_result.actions += ApplyPatchAction(...)
my_result.actions += EmailSomeone(...)
```

You can also add multiple patches which would solve the choice of multiple diffs for an issue

```
my_result.actions += ApplyPatchAction(...) # <-- patch variant 1
my_result.actions += ApplyPatchAction(...) # <-- patch variant 2
```

The issue involves allowing the bears to add custom actions apart from those provided initially by coala (`ApplyPatchAction`, `PrintMoreInfoAction`, `ShowPatchAction`, `IgnoreResultAction`)

Instead of collecting actions in a list, we can also allow actions to be cascaded or combined based on logical if-else conditions, so the bear can define an ActionSequence as



Where A, B, C, D.... are Result.action instances

Each action returns a value, that would help arrange the execution of actions in a decision tree like structure. It would still be possible to add/concatenate different action sequences.

```
Result.ActionSeq =  CoalaDefaultSeq(...)
Result.ActionSeq += XYZSeq(...)
```

Add next_action behavior in Result Action:

The above approach would allow very flexible behavior suited to the situation of the user. The approach would involve implementing a field into ResultAction called **next_action.** It's a list of actions the user can take after choosing an action. It would involve 3 cases

1. Empty List: action done, do nothing more
2. One element inside list: execute next_action immediately
3. More than one element inside list: provide a choice to the user which action to execute next

Currently, in the ConsoleInteraction module, in the print_result function, the user is prompted to choose any one of the applicable actions from the pre-defined CLI_ACTIONS list that contains (OpenEditorAction, ApplyPatchAction, PrintMoreInfoAction, ShowPatchAction, etc.)

Simply adding new actions to them should work.

cli_actions = CLI_ACTIONS + result.actions

Choice of Multiple Diffs for an issue

Some bears might have multiple solutions for the same problem. Say LanguageToolBear detects a spelling error and suggests multiple fixes. Currently we have to choose one of them automatically, It would be nice to offer all of them to the user in this case.

This can be solved with adding an optional actions field in Result.py which contains a collection of ResultAction instances associated with the Result. This will be followed by writing the tests in ResultTest.py

A sample UI would look something like this

```
|    | [INFO] Some problem in code
|    |     | + Most suitable patch (having highest confidence_score) displayed.
|    | *0: Do nothing
|    |  1: Open file(s)
|    |  2: Apply patch
|    |  3: Add ignore comment
|    |  4: Show patch 2 (default name for the action, but can be customized)
|    |  5: Show patch 3 (This option changes into “More Patches” option,
|    |                   if there are more that 3 suggested patches)
|    | Enter number (Ctrl-D to exit):
```

Interactive Diffs

Sometimes you need a value from the user to generate a patch. Say you have a bear that detects badly named variables, it would be easy to generate a diff with that bear *if* you know what the new variable name should be which you don't want to determine automatically.

A solution would be to let bears pass a function which generates a diff out of some user input, i.e. ask him "what would be a good variable name for that?" and then rename it accordingly.

```
|   2| user_name = 'utkarsh1308'
|   3| login(user_name)
|    | [INFO] ConstantsBear:
|    | This variable is a constant and should be capitalized.
|    | The following actions are applicable to this result:
|    |  0: Apply no further actions.
|    |  1: Open the affected file(s) in an editor.
|    |  2: Apply the patch ('capitalize').
|    |  3: Apply the interactive patch ('rename').
|    |  4: Show the patch ('capitalize').
|    |  5: Show the interactive patch ('rename').
|    | Please enter the number of the action you want to execute.
```

On choosing option 3 or 5 , coala will ask you to input some values which would be used to generate a diff with that info like this

```
Please type the value for 'variable_name':
```

and the user types a name that is going to be inserted into the diff wherever a placeholder is defined there.

For Interactive patches the diff should have placeholders using Python str.format method. Then the bear can use those and declare variables and we don't even need to distinguish between an interactive and a simple patch.

During the community bonding period I would like to discuss with my mentors the design to implement this feature. I think this can be implemented fairly easy once the work on next_action is completed using the approach above.

The final task at hand - Combining both Features

By the end of this project I aim to have at least a few bears that makes use of this new diff behavior.

I also aim to have at least one bear which would use the combined functionalities of both an interactive patch and multiple patches

A sample UI for NamingBear

```
|    | [INFO] Unconventional identifier naming style detected.
|    |     | - Max_studentLimit = 5
|    |     | + {max_student_limit} = 5
|    | *0: Do nothing
|    |  1: Open file(s)
|    |  2: Apply patch (changes Max_studentLimit to max_student_limit, has highest
                        confidence due to user\'s snake-case preference in coafile)
|    |  3: Apply interactive patch (to let the user manually change the variable name)
|    |  4: Show CONSTANTS patch (python specific, will propose to use MAX_STUDENT_LIMIT)
|    |  5: Do nothing
|    | Enter number (Ctrl-D to exit):
```

Timeline of Work -

Time Frame	Start Date	End Date	Task	Priority
Community Bonding Period	May 6, 2019	May 27, 2019	During this period I would merge a cEP for my project. I would like to discuss with my mentors the various issues I might face forehand and also come up with a design for generic bears that can offer multiple patches and support interactive diffs, thus kickstarting my GSOC journey (I have my end sems during this time but I will make sure to be in contact with my mentors throughout the period)	High
	May 28. 2019	June 3, 2019	I will start my journey by adding an actions field to `Result.py` and writing the tests required. I'll also add functionality for `next_action` behavior based on actions taken by the user and the `ResultAction.next_action` field.	Medium
	June 4, 2019	June 9, 2019	Replace cli_actions with result.actions field.The checks for applicability will be replaced with	Medium

			result.get_filtered_actions() method. Also allow functionality to filter native and custom actions and specify if multiple instances of same action using a variable. Also start working on bears giving choice of multiple diffs.	
	June 10, 2019	June 15, 2019	Refactor tests to test if the actions are displayed or executed. Also refactor action behavior depending on different criterias.	High
	June 16,2019	June 23, 2019	Add the functionality for all bears to show multiple diffs for an issue using the actions field implemented earlier.. Also update the docs on https://api.coala.io/ Also start thinking of a design for a bear which uses multiple patches for a single problem.	Very High
	June 24,2019	June 28, 2019	I will finish up the workload for Phase 1 evaluation and fully document the api and finish up any left work before submitting the Phase 1 Evaluations.	Very High
	June 29, 2019	July 7, 2019	I will start writing the	Medium

			multiple patches bear and write the tests and console interaction along with it. I will also finish up implementing the next_action behavior so that the bears can pass their own applicable actions.	
	July 8, 2019	July 11, 2019	Start working on bears using interactive diffs where user can specify preferred fix (eg. variable) and implementation of templated patches	Very High
	July 12, 2019	July 21, 2019	Finish up writing the multiple diff bear and almost finish up with the interactive diff bear. Document their api in https://api.coala.io/ and finish up writing tests for the bears and testing code	Very High
	July 22, 2019	July 26, 2019	I will try to write articles based on my project and even get rid of as many bugs in the code. Finishing up any leftover work before submitting the Phase 2 evaluations.	Low
	July 27, 2019	July 31, 2019	Testing and adding of combined functionality of multiple patches with interactive behavior for bears.	Low

	August 1, 2019	August 11, 2019	Define an output format for the external bears that the linters can adapt to if I have time left. Finishing up the implementation for templated patches.	Low
	August 12, 2019	August 18, 2019	Thorough testing and exception handling to be done in this period. Also finish implementing the new diff behavior in at least 1 bear.	High
	August 19, 2019	August 26, 2019	Preparing the final documentation. Testing the code to fix any minor bugs. Submit the final mentor evaluation.	Very High

How much time will I be able to contribute to my project -

My timezone is GMT + 5:30 and I will be able to devote as much as 8-9 hours daily and will try my best to meet the workload of 50-60 hours a week. I will be writing weekly reports and will always be active during my summer vacations. I will be solely focused on the project and will try my best to complete the full project as per my schedule.

Outreach -

Organisations like COALA motivate us developers to continue to increase our skill set and not stay stale with the knowledge we already have. I would personally like to mentor upcoming students in other organisations and would love to continue my work with COALA :)

About Me -

I am an python developer and have worked on different projects since my first year to improve my skills in python. I have knowledge in different domains in python from web scraping to machine learning. I also have experience in web development. I aim to be a full stack web developer by the time I graduate.

I am also a core part of a startup named TabOverSpace. I have done most of the work on the main site which required knowledge of both python and django along with web development and database management. The code is on github. I also took part in the Hacktoberfest competition and successfully completed it.

I believe I have the understood the working of the codebase having worked on it since August and I think I have the necessary skills to complete my project hence I am looking forward to it. I have researched this project thoroughly for the past month looking at all the issues and PRs filed in related to it daily and studied each and every discussion in the comments to be the best candidate for this project. I have also tried to contact my mentors on gitter and discussed new ideas and designs on the main chat. Unfortunately I couldn't talk with my mentors before the project started.

I think being familiar with the codebase and the task at hand and also being a part of coala for 9 months and counting would give me an upper hand in completing my project.

Patience is a quality which I have and I am quite proud to boast it ! I always regret the fact about not pursuing a CS degree as I myself am pursuing a dual degree in Biological Science and Chemical engineering, but I have been able to keep my interests intact throughout this journey ! I love programming and enhancing my skills :)