HSoC 2017 Application
CodeWorld - Error Parsing

About Me

Name and Contact Info

Name: Pranjal Tale
University: Indian Institute of Technology, Roorkee
Major: Applied Mathematics
Email: pranjaltale16@gmail.com
Github: github.com/pranjaltale16
Personal Website: pranjaltale16.github.io
Time Zone: IST (UTC +5:30)
Resume: Link

Personal Background

- I'm Pranjal Tale, a Sophomore at Indian Institute of Technology Roorkee, India majoring in Applied Mathematics.
- I got introduced to programming 4 years ago in my junior year at high school and it has sparked my interest since then. I started with C++ and developed small games, meanwhile, learning the fundamentals of HTML/CSS, SQL and XML.
- At college, I joined Information Management Group (IMG), IIT Roorkee - a student group which is responsible for the development and maintenance of internet and intranet applications in the institute which are hosted on a central platform called Channel I, including the Website of IIT Roorkee. It was here at IMG that my hobby developed into a passion.
- I now have a strong knowledge of web development frameworks, the MVC and MTV architectures. I've used Node.js, AngularJS, Django, Python quite extensively. I have also gained experience in coding in Haskell while playing around with the code of Codeworld. While some of these may not be relevant to CodeWorld, it has helped me gain an intuitive and superior understanding of structured web-development. I work on Elementary OS based on Ubuntu 16.04 with Vim as my primary text editor.

Programming Experience

- Information Management Group (IMG)
  At IMG, we strive to lead in the invention, development and maintenance of applications for IIT Roorkee like Placement portal, Notice Board, Study Portal (Lectures and Tutorials) and many more applications. And to glue all these intranet applications together is a platform named Channel I, which is developed in its entirety from scratch by us and is developed on Django.
  All these apps with a plethora of features make it very difficult to manage and maintain our code. Thus, we use various tools for it: Gitlab to manage code privately, Sentry to catch and report exceptions(configured with git to solve them via pull requests) and many more.
- Projects
  Since our work is mainly on intranet, I can't share their links directly. If asked, I can show you a demo via teamviewer.
  - **Nobel:**
    Tech Stack: Nodejs, Express, Socket.io, Redis, Semantic-UI
    Nobel is an internal Chat Room for the members of IMG built on Node.js, Socket.IO and Semantic-UI. This was developed to allow communications in case of an urgent matter. It keeps track of the members present in the lab at any instant of time and allows us to interact with them from outside the lab.
  - **IMG Website:**
    Tech Stack: Django, MySQL, apache2, Materialize, jQuery
    It is a fully functional website for IMG which, when launched, will allow all the members to login, maintain their profiles, write and share blogs both privately and publicly along with and admin panel to manage information on the website.
  - **Open Source and Hackathons:**
    Tech Stack: PHP, python, Django, SymPy, Microsoft-Cloud
    I actively contribute to open source organisations and have participated in various hackathons. I have implemented complete login mechanisms on both Django and PHP and hosted them on my own apache server. I have regularly been contributing to SymPy(Symbolic manipulation library written in python).
    I have also successfully completed Microsoft's Code.Fun.Do where we have to make an app from scratch in 24 hours. I made an app on Food Waste Management for which I used Microsoft Azure cloud services and backend was built on Django.

Contributions to open source

My contributions as listed here have helped me gain experience in understanding the flow of any pre-written code at a rapid pace and enabled me to edit/add new code and features with ease:

Cadasta

1. Add user profile images #110 **(Closed)**. Here is the Pull Request.

Sympy

1. Correct the diophantine function in Sympy module.(**Merged**). Here is the Pull Request.

Other Contributions

1. Registered Django-admin in Amplio(a feedback submission and discussion application, **Merged**). Here is the Pull Request.

The Project

The Problem and Motivation

Project Description

It is desirable to make children and beginners more comfortable with Haskell. Hence, I would like to work on the project titled as 'Improvements to parsing, compiling, and errors'. Currently there is a system in place for regex-based replacement of error messages, but uses are mostly limited to removing the worst parts of errors. This project would be to expand efforts and make Haskell more friendly to children and beginners. Primarily I will focus on (numbers in brackets indicate the projected number of weeks it would take me to implement them):

1. Improving Regex-model to show error messages more clearly. (2)
2. Code preprocessing to show common errors. (2)
3. Adding a test suite model for verifying or testing the regex-model.(3)
4. Enforcing certain strict rules like addition of parenthesis and improving UI.(1)
5. Adding analytics module on the working grounds of error log.(3)
6. Modifying the current suggestion box in a more detailed manner.(1)

Ideas, Resources and Implementation

- ### Improving Regex-model to show error messages more clearly.

  There is a set of regex defined in `addToMessage` function which finds a match sent by runMessageHandler and displays the error. In this part of the project my work will be to improve the regex handler in order to show errors in more clear manner. Also, I will categorise the regex

  Ex1: If I write my code as

  ```
  circl(2)
  ```

  it gives me output as

  ```
  Line 1, Column 1: error:
  Parse error: naked expression at top level
  ```

  So, this can be altered as for case 1

  ```
  Line 1, Column 1: error:
  Parse error: naked expression at top level
  main missing: program should have main associated to call
       circle(2)
  Perhaps you meant circle(2) to draw a circle of radius 2
  ```

  Ex2: if I write my code as

  ```
  main = drawingOf(r)
  r = rectangle(3,3,3)
  ```

  it gives me output as

  ```
  Couldn’t match the expected type (No, No) with actual type
       (No, No, No)
  In the parameters of rectangle, namely(3,3,3)
  ```

  So, this can be altered as for case 1

  ```
  Couldn’t match the expected type (No, No)
       with actual type (No, No, No)
  In the parameters of rectangle, namely(3,3,3)
  function rectangle accepts two parameters three given
  ```

- **Code preprocessing to show common errors.**

  Linting is the process of checking the source code for Programmatic as well as Stylistic errors. This is most helpful in identifying some common and uncommon mistakes that are made during coding. So for this problem we have to write a linter for Haskell incorporating CodeWorld API. Mainly linters are based on the concepts of Regex and there are many linters available for Haskell along with source code. Here are certain links (writing-a-linter, haskell-linter, haskell-src-ext) that I am going to follow to write a linter for codeworld.

  Main modifications are to be done in the web module and in form of javascript functions. On each keypress, a javascript function will be called which will map the code to certain regex based grammar and return the corresponding output.

  If there is something wrong in the code, it will get underlined and possible suggestions (or hints) will be shown on hover.

- **Adding a test suite model for testing the regex-model.**

  There is no test suite in order to check for the error outputs. This part of the project calls for addition of a test suite in order to test the current regex based generated error codes.

  In this test suite I am going to add a javascript based test model in which, sample code will be parsed through the codeworld source code and then the generated error after parsing from regex will be matched with the expected output.

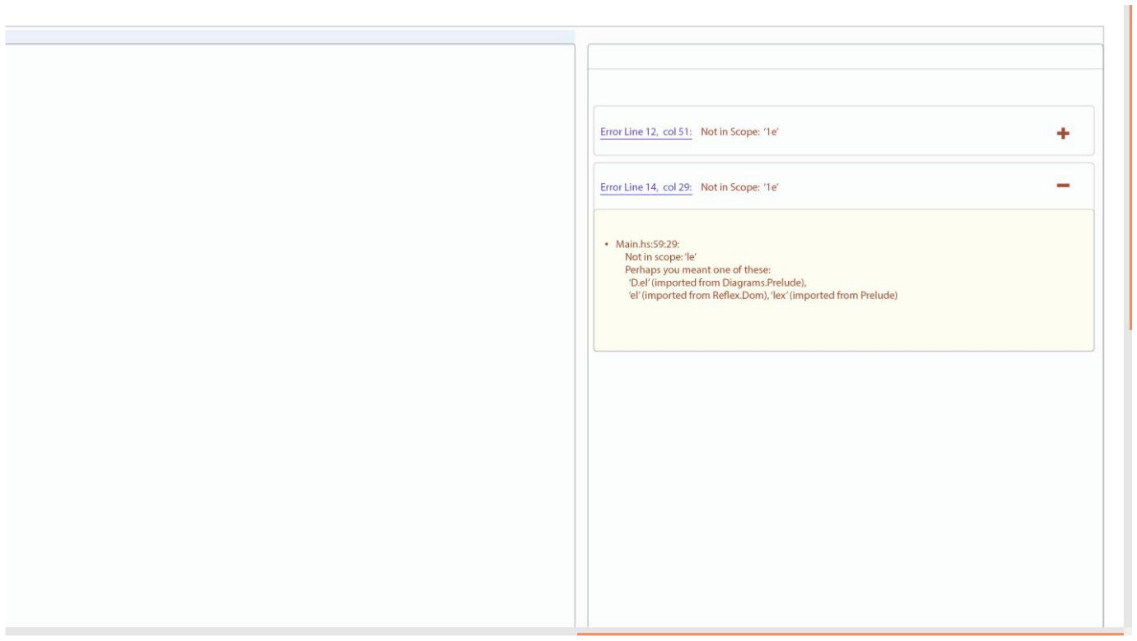
  The project will begin by adding a base structure for the same. Also, we have a predefined function in javascript module which sends codes to Haskell compiler and processes errors. The main work will be of adding a base directory structure making easier for addition of any other test suite in future. Then I'll complete it by writing the sample test cases for the all the regex cases. Also there are several js based frameworks available like mocha and qunit to follow.

- **Enforcing certain strict rules like addition of parenthesis and improving UI.**

  This can be achieved by adding grammar in the linter. If there are more than one parameters that are passed then addition of parenthesis around both of them.

Adding indentation: once the user enters for the new line it should directly indent the next line on the basis of core written in the previous line(not always), similar to the tree structure.

Here is the improved UI to show error messages inspired from HSnippet.



Error Line 12, col 51: Not in Scope: '1e' +

Error Line 14, col 29: Not in Scope: '1e' −

```
• Main.hs:59:29:
    Not in scope: 'le'
    Perhaps you meant one of these:
     'D.el' (imported from Diagrams.Prelude),
     'el' (imported from Reflex.Dom), 'lex' (imported from Prelude)
```

- **Modifying the current suggestion box in a more detailed manner.**
  This will call for showing more detailed error messages. This can be achieved by sending more clear error messages or exceptions along with the error messages from codeworld-server Build.hs.
  Main task in this will be to look around for how the compilation is done by haskell and add exceptions and more detailed error messages there.
  For ex: If there is no `main` in the code we can send an exception like 'main missing' etc.

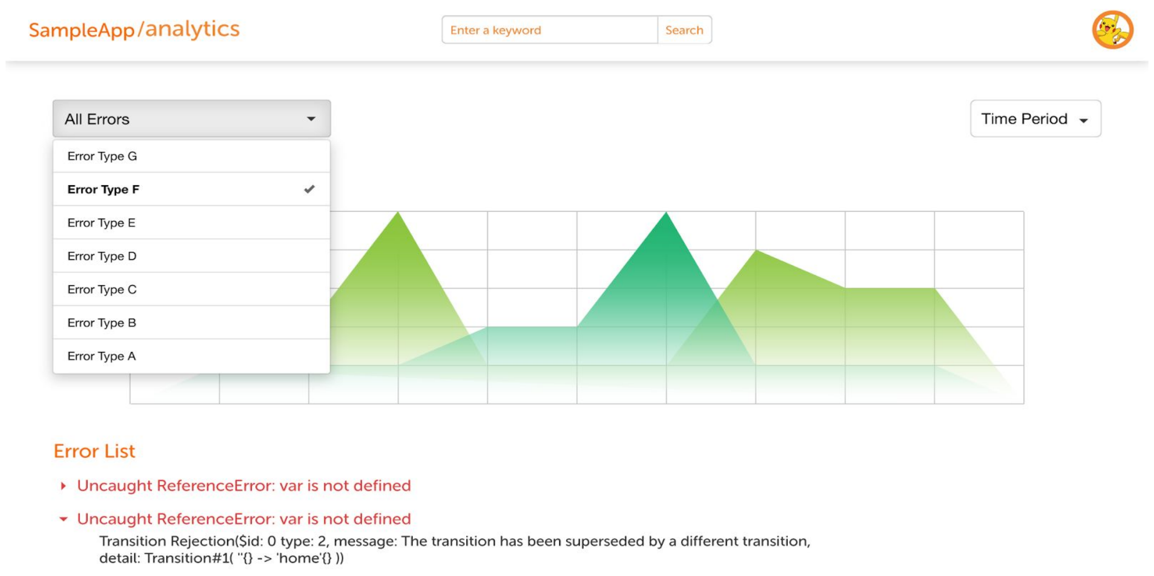
- **Adding analytics module on the grounds of error log.**
  Analytics is the discovery, interpretation, and communication of meaningful patterns in data. Especially valuable in areas rich with recorded information, analytics relies on the simultaneous application of statistics, computer programming and operations research to quantify performance. There are several articles which explain how we can write our

own analytic module or how analytics works. The main motive of this analytics module will be to improve CodeWorld and its error handling.
In this module we will attach a TypeError with every error message which will be classified by regular expression patterns, and add patterns until you've caught most of them.
I will save the error messages in JSON format along with the TypeError associated with it.
Ex: here is the structure for saving the data

```
{
   "error": [
      {
      "Category":"category1",
      "program-hash":"hash",
       "duration": "time-zone.now",
       "message": "regex-based-error",
       "raw-error": "raw-error",
       "user": "optional",
      },
   ]
}
```

We can think of a different structure also to save error messages. Categories will be allotted on the basis of regex. Then the next task would be to process this data and provide a view so that it can be easily viewed and analysed and hence providing a web page to analyse the result. There we can have filters to search for error logs in order to analyse the results in a better way.
Also results from this will help us in improving the regex patterns and providing extra information with error messages.
Only certain people will have access of this page and data(on the basis of clientId). In the end with the help of TypeError we can add more details to our error messages, like after calling the addToMessage the parsed error and raw error will be passed to a function which will save it and assign keywords to it and on the basis of that we can give more information.

Here is the sample UI



Error List

- Uncaught ReferenceError: var is not defined
- Uncaught ReferenceError: var is not defined

```
Transition Rejection($id: 0 type: 2, message: The transition has been superseded by a different transition,
detail: Transition#1( ''{} -> 'home'{} ))
```

Proposed Timeline

Community Bonding Period (22nd May - 13th June).

I will utilise this time to get more familiarized with the current code structure, read the documentation and get to know my mentor(s) and other fellow community members so that we can work together with an amplified efficiency. Some of the important things to study up on will be to learn more about **errors and exceptions in Haskell**, the current flow of code, regex model along with planning and strategizing about the changes that would need to be done with the mentor. There are also other things to look upon and discuss with the mentor during this period (especially for test Suite Model and functions in Build.hs of codeworld-server).

Week 1 (June 13th - June 20th)

In this week I'll concentrate on exhausting as many unseen errors as possible and coming up with concrete regex patterns for them at the same time, devising error messages that are more beginner and children friendly.

Week 2 (June 21st - June 27th)

This week will be dedicated to updating the regex pattern and corresponding error messages devised in Week 1 in the `addToMessage` function in `codeworld_shared.js`. This will be followed by a rigorous and thorough testing session.

First Pull Request

Week 3 (June 28th - July 3rd)

I'll finalize and formulate the grammar rules that will be included in regex based patterns to be mapped for the **linter** after thorough discussion and brainstorming sessions with the mentor.

Week 4 (July 4th - July 10th)

Once the grammar rules are finalized, I'll code the javascript function that will be called upon each keystroke and will map aforementioned rules to the linter. Again before the pull request, there will be a thorough testing session.

Second Pull Request

Week 5 (July 11th - July 18th)

I'll start working on the test suite model which will take me 3 weeks to complete. In this week, I'll set up the basic architecture of suite along with one basic test case to get it up and running.

Weeks 6-7 (July 18th - July 31st)

These two weeks I'll write rigorous test files for all the error messages and test it properly so that I'll be ready for my third pull request.

Mid Term Evaluation

Third Pull Request

Week 8 (August 1st - August 7th)

In week 8, I'll update the linter to incorporate the enforcing of certain important rules that are a must for writing a program like the completion of parentheses etc. A testing session will precede the fourth pull request. Also I will make changes in the UI part of codeworld as shown above.

Fourth Pull Request

Week 9 (August 8th - August 14st)

I'll discuss and decide on the various categories based on the regex groups in the errors in this week and start implementing the basic architecture for the same.

Week 10 (August 15th - August 21st)

This week will be dedicated to writing the classification model which would then classify the errors as one of our predefined categories. This may take more than one week so it is possible that a part of next week is taken up in this.

Week 11 (August 22nd - August 28th)

In this week I'll develop a method which would enable us to view the classified errors in an easy and informative way. After this, there'll be a thorough testing session again before a pull request is made.

Fifth Pull Request

Week 12 (August 29nd - -   )

This week will be to catch up in case anything is left due to some unforeseen circumstances. If we are well on track and everything is completed, then I'll use this week to enhance the suggestion box to make it more intuitive.

Final Pull Request

Final Evaluation

Notes

I was busy reading the references mentioned above in this proposal for the past 8-10 days, and preparing the proposal. So, I wasn't able to devote time to the bugs and issues and hence not a lot of work is done by me in recent times. Now that I've completed the proposal, I'll be concentrating mainly on bug fixes and solving issues. This will help me in getting a better understanding about CodeWorld and building a solid foundation for the same.

Also, after completing any subtask or before making any pull request I will update all the three(build.sh, run.sh, install.sh) as per the requirements.

I am going to use Trello for task management and keeping the track of all the records.

I am going to use Pre Community Bonding period in order to resolve bugs, fix issues and learning more about Haskell. Making myself more familiar with the codebase and during Community Bonding period I will primarily focus on digging into the codebase and discussing, strategizing and planning things with the mentor.

I have no other commitments for the summers and therefore I can devote 45-50 hours per week. I tried my level best to present maximum things and up to the best of my knowledge.

Even after the summer of Haskell, I will keep on contributing to CodeWorld and will try to improve the FunBlocks module.