Language Evolution Report of CoPro

- I. Rajasekhar Reddy, CS20BTECH11020

Conic Problems (CoPro), as the name suggests, was started with the aim to build a compiler for a domain specific language that could help in solving tough mathematical problems in the area of conics.

We first started with working on comments in our language

Commenting:

We have used '#' to comment a line in the code. The whole line gets commented if it starts with '#'

After which, we planned to work on declarations,

Declarations:

The format in which we decided to declare is similar to C, "type-specifier identifier : initialization". Type-specifier can be specifically named as point/ line/ pairs of lines/ parabola/ ellipse/.... Or overall it can be named as a conic.

Coming to the identifier we can use any combination of alphabets or "_" or even numbers, other than the main keywords. An identifier should also not start with a number. After the identifier is written we can assign the values using colon(":").

For initializations all the other variables are declared as in C-Language (int, double, string). We have no use of char or char* so we have removed that. Conics are initialized by their equation coefficients (6 values separated by commas)

a, b, 2h, 2g, 2f, c from the general form of conic (ax^2 + by^2 + 2hxy + 2gx + 2fy + c).

Now we started to work on function definitions.

Functions:

The format of function definition is

Function_name: arguments -> return_type

    compound_statement

Function_name can be anything as an identifier and followed by arguments separated by commas and the compound_statement. Initially we planned to begin and end the compound statement with single quotes(‘) and we have changed it to “<<” and “>>”.

Also many inbuilt functions are written to be easier for the user like finding the tangents, normals at a point on the curve, intersection of lines. These can be accessed directly without the usage of any headers.

Now we went for printing and scanning statements

OUT/IN statements :

Input can be taken by just writing “input : identifier”. To take multiple inputs we have to write this input followed by an identifier multiple times in the new line.

Output is also similar, it too requires each item to be printed to be written on multiple lines.

After this, we started doing conditional statements

Conditional Statements:

while/for : We have used only one that is loop. The syntax is

```
Loop (condition) :
    compound_statement
```

if/elseif/else : Syntax is similar to loop

```
If (condition):
    compound_statement
elseif (condition):
    compound_statement
else:
    compound_statement
```

Frac variable : To make much easier for the user to define the slopes or any variables which are in the fraction form we have introduced another variable frac where user can define a frac variable in the form of p/q

Also next to this we decided to remove the end of line. At first we had many conflicts with writing grammar and then at last we worked and fixed it.

This is the manner in which our language has evolved over time due to various reasons.