Before we talk about the data cleaning dsl, let me introduce background about data cleaning. Data cleaning is a perivious step of data analysis. Basically it will help users turn a messy datasource into more clean one, in order to do next work. It contains different tasks, for example, fill in missing values, Identify outliners and smooth out noisy data, Correct inconsistent data, Resolve redundancy caused by data integration.

There are many tools can help data scientists do data cleaning job. These tools were developed from different aspect to face different requirements. For example, R and Python are programming languages, it also has special libraries to help users in this part. SPSS, Google refine, excel are famours tools in visual data analysic. SQL is widely used in database system, it also has high performance in data cleaning. Some programmer not satisfy with these tools, they can use other programming languages like Java, C# to do data cleaning with their own scripts. And in here, we use Haskell to create a domain specific language for data cleaning, it has these features: 1.Shallow Embedded Domain Specific Language2.Command line programming3.Functional programming language4.Logical reasonable operators5.Can be used in real work environment

In this dsl, we support 5 basic operators, search, delete, duplicate, update, join. Which can satisify the basic requirement for data cleaning.

Before I explain these 5 operators, here is an example:

suppose we have this dataset, and it contains many mass informations, and repetitive rows.

If we want to change a value, and delete mess information in here, what should we do? I think at leat we can use 4 operators in here. search, delete, duplicate and update.

For this input spreadsheet, we define type class for them. For one row, it equals to datalist, the spreadsheet, it equals to list of list. Every content in the cell is string type. After we have these basic types, we can create our operators.

Let me introduce first operator, search operator. The search operator give a element as key word, an input dataset, then output the search result. The result is a dataset which just contains key word row. For example, we have a spreadsheet, called input2.csv, and the content is show in the left. then we apply search operator with keyword 2 to this spreadsheet, then we can get the datalist contains “2”. The result shown in the right part.

The second operator is delete operator. We design this operator have two sub usages: delete row or delete column. First I will explain delete column. delete row is similar with it. The main idea in this function design is move the column out of the dataset. So we set a helper function. moveEleout. This function takes the column number of the element “a”, and a datalist [a], then output the final result as a pair of (a, [a]). Then we can pick second part of the pair then pattern matching to get the result dataset. Here is the result to show, we have another dataset, then we want to delete the final column contains mess code. So we apply the column number as 3, then specify the dataset we operate, and we can get a new dataset as result.

The third operator is duplicate operator. In GHCI, we have “nub” can duplicate same elements from a list, except empty element. So if the dataset not contains empty element, we can use nub. So in the example, we can duplicate [1,2,3], not [1,2,3,””]. However it is normal that a spreadsheet contains empty cells or rows. If we want to duplicate this dataset, we need a new function. So we design dupForAll this function which fit our requirement. It need follow 2 steps:  Step 1. check if two datalists can be duplicate, step 2 duplicate the whole dataset. For step 1, we have a helper function called canBeDup, this is used for checking if the input datalist satisfy with the requirement. If it is true, then we can apply a dataset to dupForAll to get the result.

The forth operator is update value operator. In here we have two ways to update value. 1. change value by target value, we call “change by value”, 2. change value by column and row number. we call the function “change value by position”

We have two helper functions in here to assist change value by position

The first one is moveEleIn, it is similar with moveEleOut. The second one is changeHelperByPos. the case show in the slide can clear explain how it works. It takes an element, and replace it into the specific position.

Let me use an example to show change value by position function. In the input2.csv, we want to change A1’s value from 1 to “a”, so our position in here is (1,1), and the new value is a. change value by search is similar ideas, It takes a Dset, an original string, and a target string as input, and output a Dset that change the original string to the target string.

above 4 operators can be used in single or multiple datasource

so we back to the example we see before, we can delete the last column, duplicate 1,7,8 rows, and update values by using search and update operators.

the fifth operator join is used in multiple datasource environment. For multiple datasource, this is an datacleaning plan example which list operation steps for users

1. confirm the key column is unique---duplicate for key

2. join data sets by key value

3. join these data set one by one

4. use basic data cleaning operators

For the first step, we create a function called dupForKey, this will keep the key column unique. Then second step we create function called joinForKey. This function takes two dataset with key column number, then join them together. With first step, we can keep the keyvalue unique, so we can use this function safely.

The rest of the steps we can follow what we did for single datasource.

we use haskell library Text.csv to parse the input csv files, and transform to dataset.

users have two ways to use our DSL for their data cleaning task. They can choose write line by line in the command console, or they can program the whole DSL into one file, then run the file. Here is an example in the slide.

Every tools contain different limitations, like SQL, it just run on database platform, and hardly to generate visual analysis. Excel vba need users have programming skill, same as python, R. In our DSL, we just have basic operators, and we didn’t create analytic functions. data cleaning is a very deep area, it can be combine with data mining, so machine learning method can be applied in here. Another limitation is user need to install haskell library Text.csv, this is not in the haskell prelude library. Another limitation is when the input file is very large, the performance is not very well.

Our future work will focus on using statistic method to create analysis functions, and improve the performance when input a large dataset.