Programming Language Concepts Report

Syntax

algorithms will follow a simple template :
from [...] **select** […] **where** […] **orderby** [...]

from : selects the tables we work with (Q will correspond to a file called q.csv) :

from P and Q select P.all, Q.all where empty orderby empty

select : returns and prints the tables coming after : from A select A.all where empty orderby empty

where : will only order the output if the rows respect the condition laid after : from P and Q select P.all where P.0 == Q.0 orderby empty

orderby : will order the output in the specified order (including 'empty') using the first column : from A select A.all where empty orderby alphabetical

from E and X **select** E.all **where** E.0 == X.0 **orderby** alphabetical

e.csv

Thomas	Kathy	Ron
Berny	Josh	
Kate	Austin	
Alice	Bob	John

x.csv

Thomas	Ferdinand	
Berny	Ron	Harry
Austin	Steven	
Alice	Randy	Paul

→

output

Alice	Bob	John
Berny	Josh	
Thomas	Kathy	Ron

functions can be applied to the tables after **select**, i.e :

from E and X **select** E.all **conjunction** X.all where empty orderby empty

I.csv

Thomas		Ron
Larry	Josh	

K.csv

Thomas	Thomas	Bob
Larry		

L.csv

Thomas	Todd	Todd
Larry	Josh	Amy

M.csv

Josh	Amy	Chris
Todd	Todd	Jonhson

[...] select **I**.all **conjunction** **K**.all [...]

from a1,a2 ; b1,b2 to a1,a2,b1,b2

Thomas		Ron	Thomas	Thomas	Bob
Thomas		Ron	Larry		
Larry	Josh		Thomas	Thomas	Bob
Larry	Josh		Larry		

[...] select **L**.all **composition** **M**.all [...]

from a1,q1,q2 ; q1,q2,a3 to a1,a3

Larry	Chris
Thomas	Jonhson

[...] select **I**.all **leftJoin** **K**.all **on** 0 [...]

[...] select **I**.all **rightJoin** **K**.all **on** 0 [...]

fromSQL, 'on 0' means we are matching from the left-most column, we can also use 'on last' for right-most matching

Larry	Josh	
Thomas	Thomas	Ron

Thomas	Thomas	Bob
Larry	Josh	

[...] select **K**.all **matchPairs** **L**.all [...]

from a1,a1,a2 ; b1,b2,b2 to a2,b2

Bob	Thomas

Evaluation function

The evaluation function follows the syntax order :

i.e. **from** E and X **select** E.all **where** E.0 == X.0 **orderby** alphabetical

Firstly the **from** part. It opens and reads every file mentioned inside the **from** statement. It then returns an array of pairs containing the name of the file and its content as follows [(name,content)]. When the evaluation reads the file it removes the last line of the file if it is empty, and removes all the spaces before and after the value contained inside the cell. This array is then passed on to the select part of the evaluation function.

Secondly, **select**. It takes the array of files and their content as parameters. It evaluates operations such as 'merge', 'comma', 'conjunction', and does so recursively using a leftmost innermost strategy. Once it reaches a table value coming from a file it retrieves the desired file content and works it's way up through the different operations. Once the evaluation is done it passes the table and the array of files to the where part.

Thirdly, **where**. To begin with, the **where** part evaluates the boolean value and returns a function which takes a row of each file's table, and of the evaluated table, then returns a boolean value. If it returns true the where part keeps the row if it returns false it returns blank and moves on to the next row. If the value is empty the where function just returns the table given by the select.

Finally, **orderby**, we take the table produced by the where part and applies the sorting function specified by the program. If the value is alphabetical it returns the table given by the **where** part sorted lexicographically. If it sees empty it just returns the table given by the **where**.

Main language features

Any whitespace is ignored

Comments can be added, this is done by writing "--" first, then writing the comment. "--" has to be written on each line that you want to comment, anything following "--" will be ignored, e.g.:

We have followed a SQL like language, using From Select Where and Order by as a format to write our answers, we have included other SQL keywords like leftJoin, merge and etc, to handle what operations are done on the tables.

We also have some basic tokens like boolean, int, var. Also, some additions like "all" which is used like the following: A.all this is parsed as FromTable Table "A" all, this will select all columns from the table, this is better than writing out every column from the table like: A.1 A.2 A.3 A.4 etc…

We have a format for tables columns and rows: A.1.2 is parsed into Table "A", Column "1" and Row "2", this is grouped together and called a Cell. Columns can be selected by using the following format: A.1 which is parsed into Table "A", Column "1". Otherwise, Tables can be selected by just writing A which is parsed into Table "A".

Informative error messages

We have used the "posn" wrapper, so that we can identify where the error has occurred

If the language was not identified an error will be displayed: "Unknown Parse Error".

If a token was not identified, another error with the line and column number of the unidentified token will be shown

Type checker

We have type checking for each of the expressions defined in the grammar

We have 6 type tokens to determine types these are: Bool, Int, String, Ordering, Empty, Cell