Programming Language Concepts

Programming Language Design Coursework

EventCQ

User Manual

1. Introduction

Event Conjunctive Query (EventCQ) is a programming language with the expressive power of conjunctive queries. Conjunctive queries are the simplest form of queries that can be expressed over a database therefore EventCQ would be very useful for processing such queries over data stored in comma separated values files.

2. Syntax

The syntax of EventCQ is simple and concise while still allowing the full expressive power of conjunctive queries. A consequence of this simplicity is the fact that an EventCQ program can be a minimum of 2 lines and a maximum of 4, excluding any comments.

The language is an interpreter for the recursive grammar for the calculus of conjunctive queries:

$$\Phi ::= \Phi \wedge \Phi \mid x_i = x_j \mid R(\vec{x}) \mid \exists x.\Phi$$

2.1 Dataset loading and output

As mentioned earlier, each program must consist of at least 2 lines, the first one being the file names and their respective columns while the last line being the order of printing. These lines are mandatory for a program to be defined.

The first line states the names of the files for processing which does not require the ".csv" file extension.

Each file name should be followed by variable names of the columns contained in ".csv" files. Every column must be given a variable name, and these are contained in brackets and separated from each other by space.

The final line contains all the names of columns not eliminated by an existential quantifier (explained in 2.2) and the desired order of printing.

```
1   A ( x1 x2 ) B ( x3 x4 )
2   x1 x3 x2 x4
```

First line of a program defines files A.csv and B.csv and their columns.

Last line of a program defines the order of printing output.

A.csv	B.csv	Expected output
		1,3,2,4
1,2	3,4	1,3,2,4
1,2	3,4	1,3,2,4
		1,3,2,4

If any of the files read are empty, then the output will be empty as well. The same file can be read multiple times, and the column names can differ for each different reading.

2.2 Defining equality and existential quantification conditions

Between the mandatory lines of each problem, additional conditions can be defined, namely list of equalities and/or list of variables bound under the scope of an existential quantifier.

The beginning and end of an equality list is marked by an equality sign "=".

Each equality is separated from other equalities by a space and each element of an equality pair is separated from the other element with an equality sign.

```
1   P ( x1 ) Q ( x2 )
2   = x1=x2 =
3   x1 x2
```

Line 2 defines an equality pair between the columns x1 and x2.

P.csv	Q.csv	Expected output
1	1	1,1
2	2	2,2
3	2	2,2
4	Sofa	

```
2   = x1=x2 x3=x4 =
```

An example of defining multiple equality pairs.

Similarly to equality, the beginning and end of an existential quantification list is marked by a minus sign "-".

The variables in that list specify which variables are removed from the list of variables that must be printed. Variables inside that list are also separated from each other by space.

```
1   R ( x1 z )
2   - z -
3   x1
```

Line 2 defines an existential quantifier "z".

R.csv	Expected output
Michelangelo,Buonarotti	Antonello
Leonardo,Da Vinci	Leonardo
Antonello,Da Messina	Michelangelo

```
2    - z1 z2 -
```

An example of multiple variables bound by an existential quantifier.

Note: It doesn't matter if a list of variables bound under the scope of an existential quantifier appears after or before a list of equalities and vice versa.

```
1    A (x1 z) B ( z x2 )
2    - z -
3    = x1=x2 =
4    x1 x2
```

is the same as

```
1    A (x1 z) B ( z x2 )
2    = x1=x2 =
3    - z -
4    x1 x2
```

If there are 2 or more columns with the same name, you will only consider the rows from the Carthusian product of the files that have the elements corresponding to the columns named previously equal.

If there are 2 or more columns with the same name, the program will act as if they are all in an equality relation with each other and are all but one under the scope of an existential quantifier.

```
1    A ( x1 x2 ) B ( x2 x3 )
2    x1 x2 x3
```

Last column of A and first column of B have the same variable name "x2".

A.csv	B.csv	Expected output
1,3 1,2	3,1 3,2 2,2	1,2,2 1,3,1 1,3,2

3. Additional Features

3.1 Comments

EventCQ allows for user comments which are marked by two minuses "--", inspired by Haskell. Like most programming languages, they can be situated where desired.

3.2 Error message

Where there is a syntax error a simple error message will be displayed:

```
myinterpreter.exe: lexical error
```

Appendix A: Problems from coursework specification in EventCQ

Problem 1:

```
A ( x1 x2 ) B ( x3 x4 )
x1 x3 x2 x4
```

Problem 2:

```
A ( x1 x2 ) B ( x2 x3 )
x1 x2 x3
```

Problem 3:

```
P ( x1 ) Q ( x2 )
= x1=x2 =
x1 x2
```

Problem 4:

```
R ( x1 z )
- z -
x1
```

Problem 5:

```
A ( x1 z ) B ( z x2 )
- z -
x1 x2
```

Problem 6:

```
R (x1) S(z)
- z -
x1
```

Problem 7:

```
R (x1 z1) R(z1 z2) R(z2 x2)
- z1 z2 -
x1 x2
```

Problem 8:

```
R (x1 x2) R (x2 x3) R (x3 x4) R (x4 x5)
= X1=X5 =
x1
```

Problem 9:

```
A (x1 z1) B(z1 z2) C(z2 x2)
- z1 z2 -
x1 x2
```

Problem 10:

```
S ( x1 x2 x3 ) S( z1 z1 z2 ) S ( z1 z2 z2 )
- z1 z2 -
x1 x2 x3
```