

Introduction

You will write a program that computes the BCNF decomposition of a pair (relation, functional dependencies). You must first implement the classes and methods described below:

1. **Relation** class: A relation is a set of attributes and can be implemented easily using an array `A` of integers with values of 0 or 1, where `A[i]=1` means that the attribute with ASCII (UNICODE) code `i` is in the relation.
 - `Relation(String in_r)`
 - `String toString()`
 - `boolean equals(Relation r2)`
 - `boolean contains(char c)`
 - `boolean subset(Relation r2)`
 - `Relation powerSetFirst()`
 - `Relation powerSetNext()`
 - `Relation union(Relation r2)`
 - `Relation intersect(Relation r2)`
2. **Fd** class: A functional dependency is an object containing two relations (sets). One on the left hand side and one on the right hand side.
 - `Fd(Relation in_lhs,Relation in_rhs)`
 - `String toString()`
 - `boolean BCNFviolation(Relation s)`
check if this functional dependency is a BCNF violation with respect to the given set of attributes of relation `s`
 - `Relation getLHS()`
 - `Relation getRHS()`
3. **FdList** class: A list of functional dependencies.
 - `FDList()`
 - `String toString()`
 - `void insert(Fd f)`
 - `Fd getFirst()`
 - `Fd getNext()`
 - `Relation closure(Relation r)`
computes the closure with respect to a list of functional dependencies

4. `RelList` class: A list of relations.

- `RelList()`
- `String toString()`
- `void insert(Relation r)`
- `Relation getFirst()`
- `Relation getNext()`

Activities

1. Write a program, using an object oriented programming language, that takes as input a file with a relation (set of attributes R) and a set of functional dependencies and outputs all non-trivial functional dependencies that can be obtained from the ones in the file. Notice that you will need to compute the closure of all possible subsets, and all non-trivial functional dependencies must be such that $X \rightarrow X^+ - X, \forall X \subseteq R$.

The first line of the file must be the attribute list of the relation separated by spaces. Assume that the only possible attributes are the 26 letters from **A** through **Z**.

Sample file:

```
A B C D E
A B -> C
C -> D
D -> B E
```

2. In order to determine if a given functional dependency $X \rightarrow Y$ is a BCNF violation with respect to a set of attributes S , notice that:

- X must not be a superkey, i.e. XY must not contain all attributes in S which can be checked with the condition

$$S \not\subseteq XY$$

- $X \rightarrow Y$ must be a valid functional dependency with respect to S , which can be checked by making sure that the following conditions are true:
 - $X \subseteq S$
 - $Y \cap S \neq \emptyset$

Modify your program from question 1 so that it outputs those functional dependencies which are BCNF violations.

3. Implement the algorithm given in class to compute the BCNF decomposition of a given relation and a set of functional dependencies. Your program must output, in parenthesis, the attributes of each relation in BCNF. Even though the algorithm given in class is recursive, you can use a stack of relations instead.

What to turn in

- The source code of all your classes including your driver BCNF decomposition program.
- The output produced by your program on the (sample) file given above.
- The output produced by your program on another example of your choice.