Finding All Keys

Part 1 of the Algorithm

RemoveRedundant(S,F)

Input: Attributes S and FDs F

Output: Minimal T such that $T \subseteq S$ and $S \subseteq T^+_F$

- T:=S
- Foreach $A \in S$ do
  - If $A \in (T\text{-}A)^+_F$ then T:=T-A
- Return T

What does this remind you of?

Part 2 of the Algorithm

All Keys(R,F)

Input: Schema R and FDs F

Output: All keys of R with respect to F

- *Keys*:= {RemoveRedundant(R,F)}
- Foreach $K \in$ *Keys* do
  - Foreach $X \to A \in F$ for which $A \in K$ do
    - $S := K\text{-}\{A\} \cup X$
    - If S does not contain any $J \in$ *Keys* then
      - S' := RemoveRedundant(S,F)
      - Add S' to Keys
- Return *Keys*

Example

- R = ABCD
- F = {AB → C, C → DA, BD → C, AD → B}
- Find all the keys of R

Correctness

- **Claim:** Every K added to Keys is a key of R
- **Proof:** By induction. Let K_i be the i-th key added to Keys
  - Base Case: i=1. Then, K_1 is obviously a key by the definition of remove redundant.
  - Induction Step: Assume for j<i. Let K_j for some j<i and $X \to A \in F$ be such that K_i returned from RemoveRedundant($K_j\text{-}A \cup X$,F). By the induction hypothesis, K_j is a key. It immediately follows that $K_j\text{-}A \cup X$ is a superkey, and RemoveRedundant($K_j\text{-}A \cup X$,F) is a key.

Correctness

- **Claim:** Every key K is eventually added to Keys
- **Proof:**
  - First, observe that at least one key will be added to Keys.
  - Now, suppose that there is some key K' that is not in Keys.
  - We will show that the algorithm will find some additional key to add to Keys.
  - Let K'' be a maximal subset of R containing K', but not containing any key in Keys.
  - Since K'' does not contain any key in Keys and Keys is not empty, there is some attribute in R that is not in K''.

Correctness (cont)

- Since K''+ contains all attributes, there must be some functional dependency $X \rightarrow A$ such that $X \subseteq K''$, but $A \notin K''$.
- By the choice of K'', we have that $K'' \cup A$ contains some key K in Keys.
- During its iteration over K and $X \rightarrow A$, the set $K\text{-}A \cup X$ will be computed.
- Note that $X \subseteq K''$ and $K\text{-}A \subseteq K''$, and therefore $K\text{-}A \cup X \subseteq K''$.
- Since K'' does not contain any key in Keys, $K\text{-}A \cup X$ also does not and a new key will be generated by the algorithm.

Runtime

- Runs in polynomial time in the size of the input and output, i.e., in the size of R,F,Keys.
- For each K in Keys and each FD in F we:
  - iterate over Keys (to check containment)
  - call RemoveRedundant