Normal Forms

- First Normal Form (1NF): Every attribute must be atomic. Attributes containing several pieces of information are not allowed. An example of a non-atomic attribute is an attribute holding several phone numbers like "(555)555-5555; (666)666-6666, (777) 777-7777". We will assume that every table that we design is already in 1NF. It is not possible to check for 1NF based on functional depencencies.

- Second Normal Form (2NF): $\forall\, X \to Y \ \in \ F^+$
  if Y is non-prime $\Rightarrow$ $X \not\subset K$, where K is a (candidate) key.

- Third Normal Form (3NF): $\forall\, X \to Y \ \in \ F^+$
  if Y is non-prime $\Rightarrow$ X must be a superkey (notice that a (candidate) key is also a superkey).

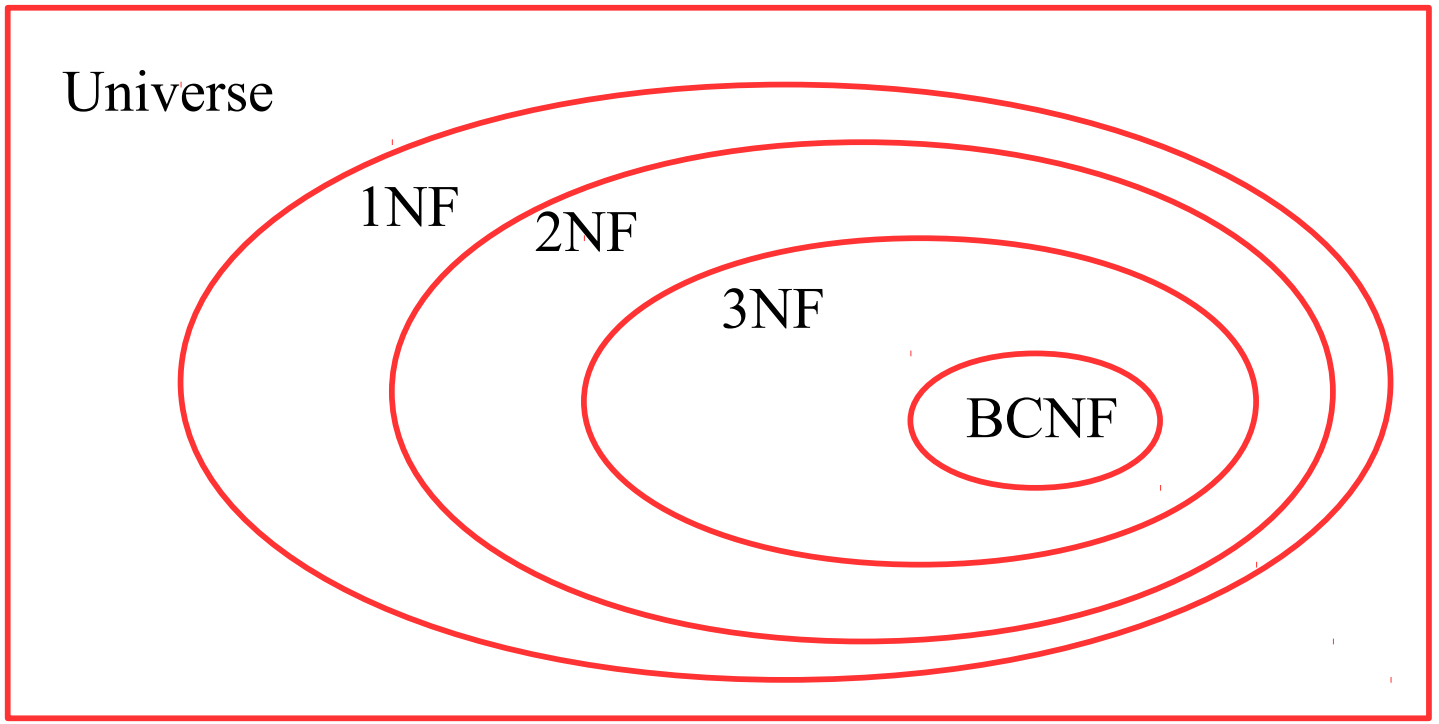
- Boyce-Codd Normal Form (BCNF): $\forall\, X \to Y \ \in \ F^+$
  X must be a superkey.

In the cases of 2NF, 3NF, and BCNF, we can show that a particular (R, F) is not in one of those normal forms by finding a violation, i.e. a functional dependency that does not satisfy the definition. For example, a 2NF violation is a functional dependency where its RHS is non-prime but the LHS is a proper subset of some key. Similarly, a 3NF violation is a functional dependency where the RHS is non-prime but the LHS is not a superkey. A BCNF violation is a functional dependency where the LHS is not a superkey.

Normal Form Hierarchy

Properties:

- BCNF $\subseteq$ 3NF: If the LHS of every functional dependency is a superkey, then it will be a superkey even in the case where de RHS is not prime.
- 3NF $\subseteq$ 2NF: If the LHS of every functional dependency is a superkey whenever the RHS is non-prime, then the LHS cannot be a proper subset of some key, or else the key would not have been minimal.



How to determine the strongest normal form

Given a pair (R,F), relation and set of functional dependencies.

We start trying to find violations for 2NF.
If no violations found for 2NF, we try 3NF next.
If no violations found for 3NF, we try BCNF
If no violations are found for BCNF, then it is in BCNF

Three properties that might help in special cases:

- If every key is a singleton (consists of only one attribute), then it cannot have a non-empty proper subset, therefore (R,F) is in 2NF.
- If all attributes are prime, then there are no violations to 3NF, therefore, (R,F) is in 3NF.
- If there are no functional dependencies, then no violations will be found, and (R,F) is in BCNF.

Example 1 (Normal Forms):

Determine the strongest normal form that the following (R,F) satisfies:

$$R(ABC)\ F = \{A \to B,\ B \to C\}$$

First we find the keys, as described before: A

Since the key is a singleton, it is in 2NF.
Notice that the functional dependency $B \to C$ has a non-prime attribute on the RHS and the LHS is not a superkey. Therefore $B \to C$ is a 3NF violation.

(R, F) is in 2NF, but is not in 3NF.

Example 2 (Normal Forms):

Determine the strongest normal form that the following (R,F) satisfies:

$R(ABCDE)\ F = \{ABD \to C,\ BC \to D,\ CD \to E\}$

We find first all keys as described before:

None or Left	Both	Right
AB	CD	E

$$\frac{AB^+}{AB} \qquad \frac{ABC^+}{ABCDE} \qquad \frac{ABD^+}{ABDCE}$$

So we have two keys: ABC and ABD.

A, B, C, and D are prime attributes, and E is non-prime.
If we compute F^+, we will obtain $BC \to E$, where E is non-prime and BC is a proper subset of ABC, which is a (candidate) key. Therefore $BC \to E$ is a 2NF violation.

(R, F) is not in 2NF.

Example 3 (Normal Forms):

Determine the strongest normal form that the following (R,F) satisfies:

$R(ABC)$ $F = \{AB \to C,\ C \to A\}$

We find first all keys as described before:

None or Left	Both	Right
B	AC	

$$\frac{B^+}{B} \qquad \frac{BA^+}{BAC} \qquad \frac{BC^+}{BCA}$$

So we have two keys: AB and BC.

All attributes are prime, therefore (R, F) is in 3NF. We will now try to find a BCNF violation

Notice that the LHS of the functional dependency $C \to A$ is not a superkey, therefore it is a BCNF violation.

(R, F) is in 3NF but not in BCNF.

BCNF decomposition algorithm:

The following algorithm returns a set of pairs of relations and functional dependencies that would apply for those relations.

$$Q = \{(R_1, F_1),\ (R_2, F_2),\ \ldots, (R_n, F_n)\}$$

decomposeBCNF(R,F)

$Q = \emptyset$

find a BCNF violation $X \rightarrow W$

$Y = X^+ - X$

Assume that $R(XYZ)$

$R_1 = R_1(XY)$

$R_2 = R_2(XZ)$

$F_1 = $ project F onto XY

$F_1 = $ project F onto XZ

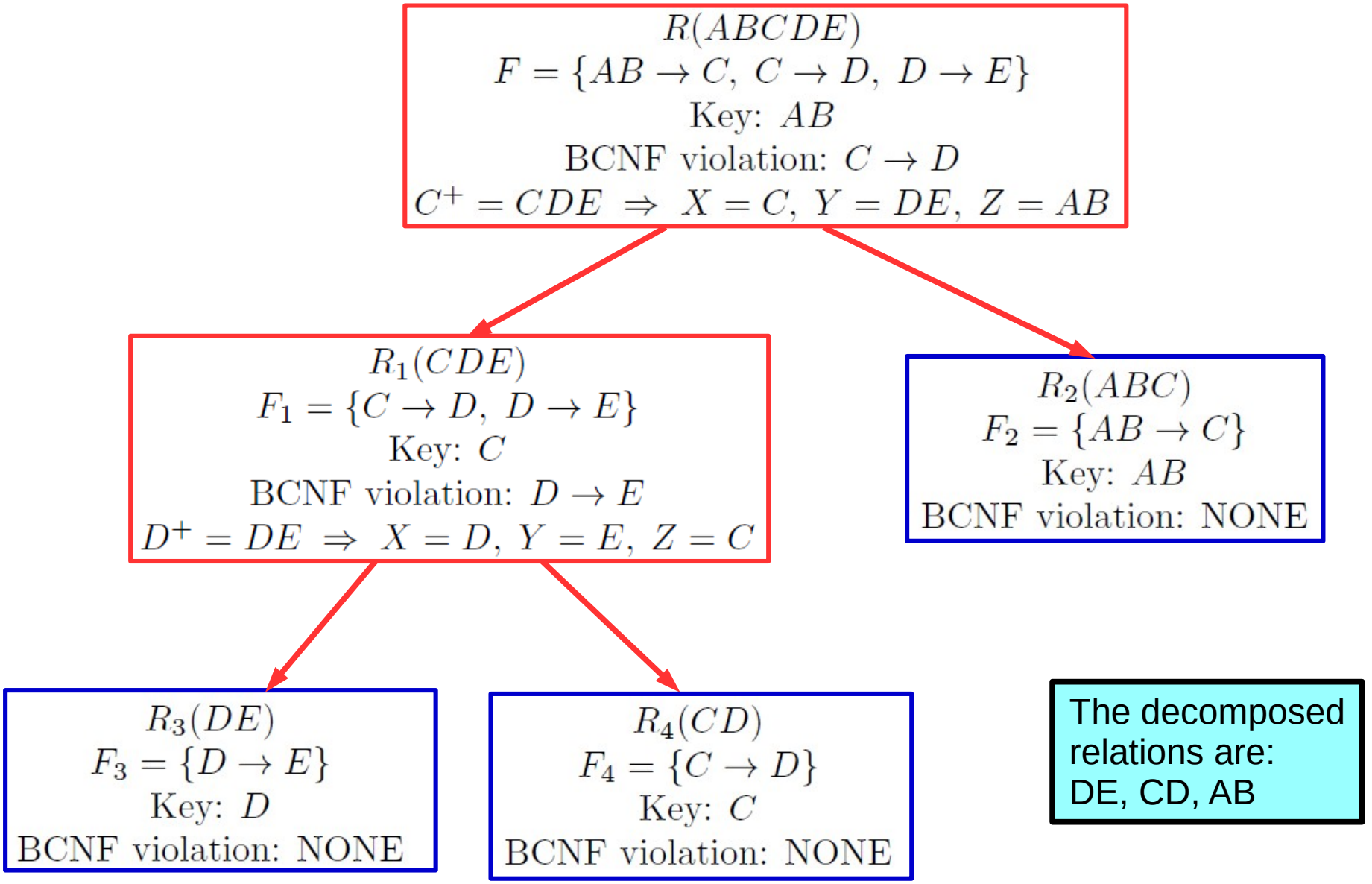
$Q = Q \cup \{\text{decomposeBCNF}(R_1, F_1), \text{decomposeBCNF}(R_2, F_2), \}$

return Q

The algorithm must be recursive because once a table has been decomposed, it might still not be in BCNF. The algorithm computes a tree of tables where the leaves represent the final BCNF decomposition.

Example 1 (BCNF decomposition)

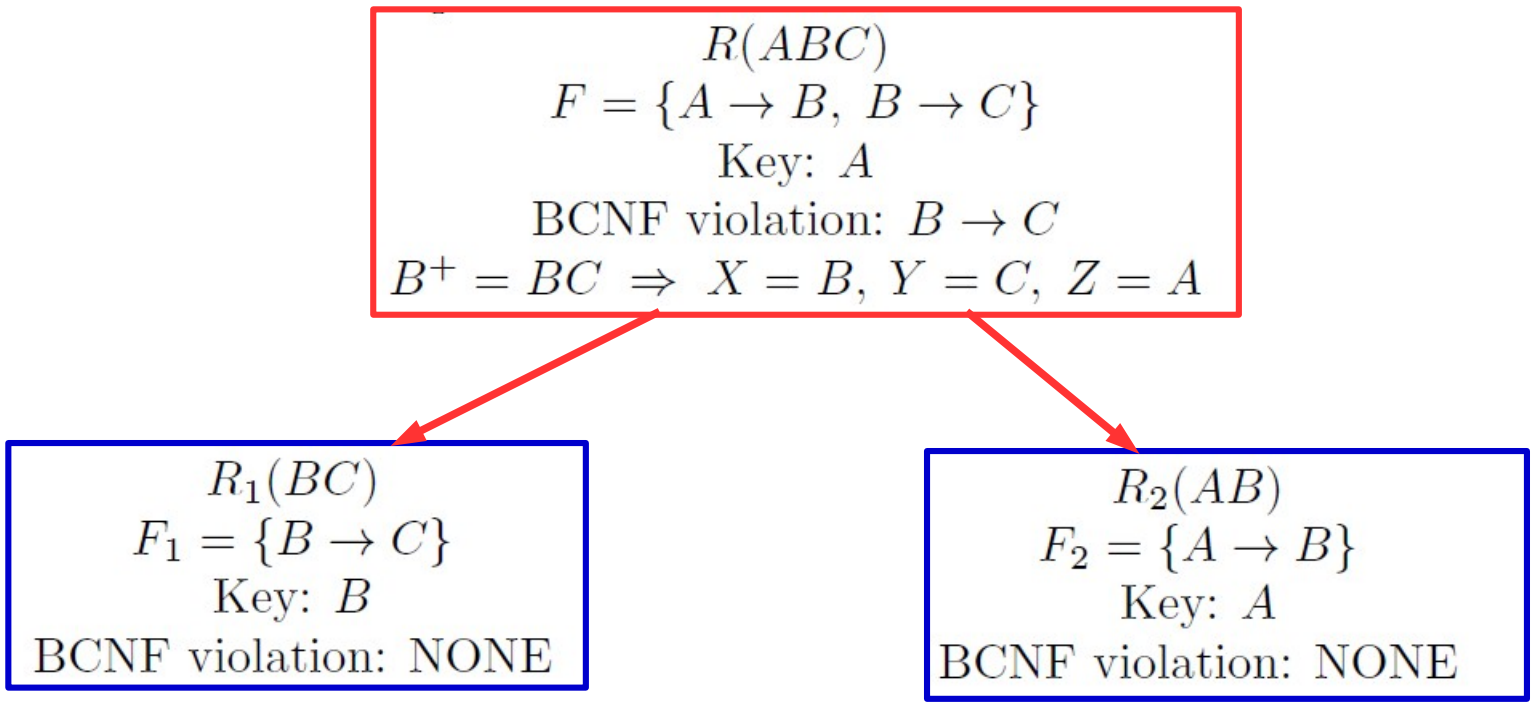
Given $R(ABCDE)$, $F = \{AB \to C,\ C \to D,\ D \to E\}$
Decompose R into tables in BCNF.



The decomposed relations are:
DE, CD, AB

Example 2 (BCNF decomposition)

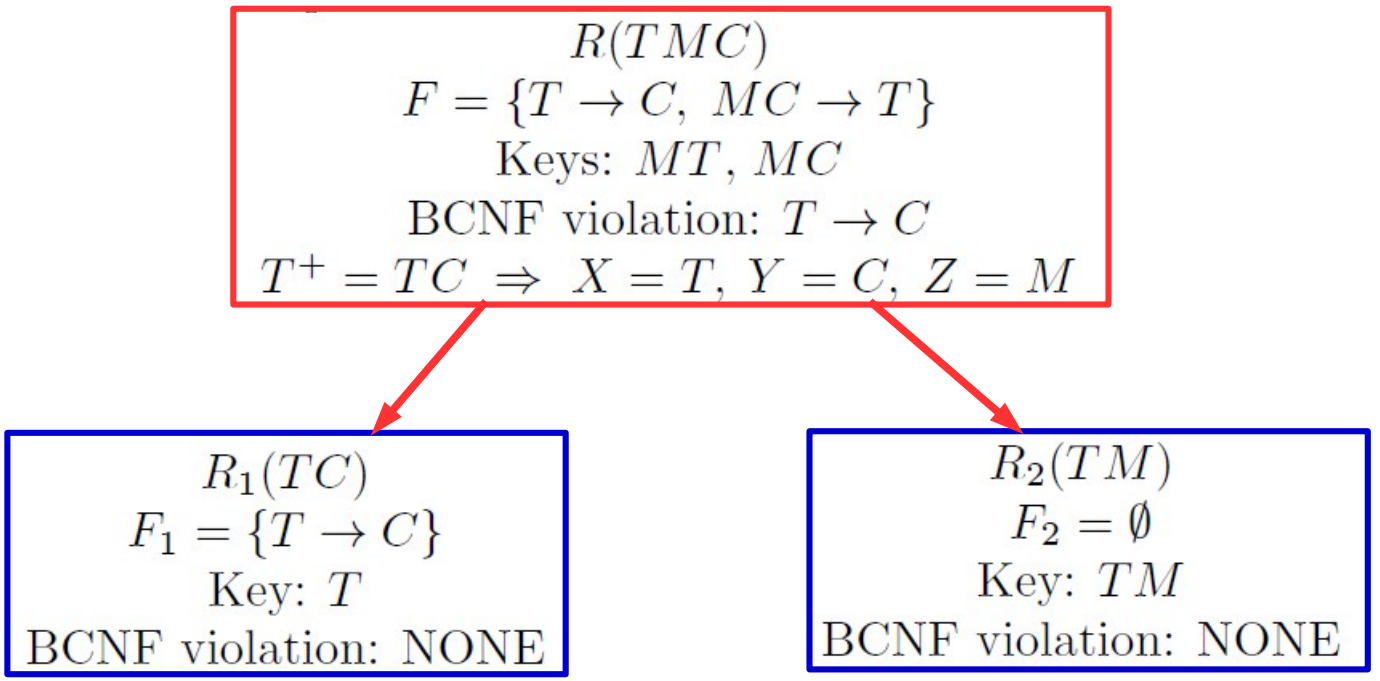
Given $R(ABC)$, $F = \{A \to B,\ B \to C\}$
Decompose R into tables in BCNF.



The decomposed relations are:
BC, AB

Example 3 (BCNF decomposition)

Given $R(TMC)$, $F = \{T \to C,\ MC \to T\}$
Decompose R into tables in BCNF.



The decomposed relations are:
TC, TM

Lost Functional Dependencies: Notice that it is not possible from the projected functional dependencies of the decomposed tables, to obtain the original set of functional dependencies.

$$F_1 \cup F_2 = \{T \to C\}$$

The functional dependency $MC \to T$ has disappeared!

Goals of decomposition

- El**imination of anomalies:**
  BCNF eliminates insertion, deletion and update anomalies
  3NF does not eliminate them but reduces them

- **Lossless decomposition:**
  As long as the decomposition is made according to Fuctional Dependencies it is always going to be lossless.
  BCNF decomposition is lossless, an algorithm to perform 3NF decomposition should also be based on functional dependencies.

- **Functional Dependency Preservation:**
  As we saw from the previous example, BCNF decomposition does not necessarily preser functional dependencies.
  3NF decomposition does preserve functional dependencies, so we need to figure out a way to decompose a table into tables in 3NF

Minimal Cover

In order to decompose a table based on functional dependency preservation we will use a minimal set of functional dependencies from which all FDs in F can be obtained.

Definition:
A minimal cover M of (R,F) is a minimal set of F^+ that satisfies:

$$M^+ = F^+$$

Computing a Minimal Cover:

1. Rewrite all functional dependencies so that the RHS are singletons.
2. Remove those attributes from non-singleton LHS that can be derived from the other ones. (Example: $AB \rightarrow C$, and $B \subseteq A^+$, then B is not needed in the functional dependency since it can de derived from A, so we get $A \rightarrow C$
3. Remove redundant functional dependencies $X \rightarrow A$. Compute X^+ without using $X \rightarrow A$ (notation: $X^+_{\{X \rightarrow A\}}$), if $A \in X^+_{\{X \rightarrow A\}}$, then there is no need for $X \rightarrow A$.
4. Use the union rule on LHS of all functional dependencies.

Example (Minimal Cover)

Compute the minimal cover of $R(ABCDE)$,
$A \to D,\ BC \to AD,\ C \to B,\ E \to A,\ E \to D$

1. $A \to D,\ BC \to A,\ BC \to D,\ C \to B,\ E \to A,\ E \to D$
2. We look only at those where the LHS is not a singleton:
   - BC: $B^+ = B,\ \ C^+ = CBDA$
     Therefore B is not needed since it can be derived from C.
     We only keep $C \to A,\ C \to D$
3. In the following table, the second column represents $X^+_{\{X \to A\}}$ for each functional dependency $X \to Y$.

$A \to D$	A	keep
$C \to A$	CDB	keep
$C \to D$	$CABD$	eliminate
$C \to B$	CD	keep
$E \to A$	ED	keep
$E \to D$	EAD	eliminate

4. The minimal cover is: $A \to D,\ C \to AB,\ E \to A$

3NF Decomposition Algorithm

We are given R and a minimal set of functional dependencies F_{min}. The algorithm outputs a set Q of tables in 3NF.

$Q = \emptyset$
for each $X \rightarrow Y \in F_{min}$
    if $XY \not\subseteq T \ \ \forall T \in Q$
        $Q = Q \cup XY$
if no relation $T \in Q$ contains a candidate key
    Let K be a candidate key
    $Q = Q \cup K$
/* At the end of this algorithm it is possible to still
   have schemas in Q that are contained in another, so
   we will add one final step:*/
Remove from Q any schema T that is contained in another one.

Example 1 (3NF Decomposition)

Decompose $R(ABCDE)$ into tables in 3NF,
$A \to D,\ BC \to AD,\ C \to B,\ E \to A,\ E \to D$

- For this example we already computed a minimal cover:
- $F_{min} = \{A \to D,\ C \to AB,\ E \to A\}$.
- For each functional dependency in F_{min} we create a schema:
  $Q = \{AD,\ ABC,\ AE\}$
  Making sure that no schema is contained in another one.
- Since the only key is CE, and it is not contained in any schema in Q, we add CE to Q.

Final tables in 3NF: $Q = \{AD, ABC, AE, CE\}$