Some shortcomings of Bernstein's algorithm

- **Shortcoming 1.** Bernstein's algorithm does not guarantee reconstructibility (or losslessness).

Example 3. Given R (Course#, Preq#, Cname, Cdesc) with

$\mathbb{F}$ = {Course#, Preq# $\rightarrow$ Cname
Course# $\rightarrow$ Cname, Cdesc}

Step 1 $\mathbb{G}$ = {Course# $\rightarrow$ Cname, Cdesc}

Step 2 $\mathbb{H} = \mathbb{G}$

:

Step 6 R_1 (Course#, Cname, Cdesc)

Note: We lose information about Preq#.

- **Q:** How to resolve this problem?

In fact we have Course# $\twoheadrightarrow$ Preq#

(**Note**. It is a multi-valued dependency, to be discussed later. Bernstein's algorithm does not handle MVDs).

We need another relation:

R_2 (Course#, Preq#)

❖ **Shortcoming 2.** Bernstein's algorithm does not find **all** the keys.

Example 4. Given R (A, B, C, D)
with $\mathbb{F} = \{ AB \rightarrow CD, C \rightarrow B \}$
Apply the algorithm, we will get

R_1 (A, B, C, D)
R_2 (C, B)

- In fact, {A, C} is also a key of R_1.
  This is called an implicit key.

❖ **Note:** R_1 is not in BCNF.

❖ **Note:** To find all the keys of a relation is NP-complete.

Q: What is the meaning of NP-complete? A term from complexity theory.

❖ **Shortcoming 3.** Bernstein's algorithm does not remove all the superfluous attributes (i.e. redundant attributes).

Example 5. Given $\mathbb{F}$ = { AD $\rightarrow$ B, B $\rightarrow$ C, C $\rightarrow$ D,AB $\rightarrow$ E, AC $\rightarrow$ F }

Step 1 $\mathbb{G} = \mathbb{F}$

Step 2 $\mathbb{H} = \mathbb{G} = \mathbb{F}$

:

Step 6 R_1 (A, B, C, D, E, F)

R_2 (B, C)

R_3 (C, D)

❖ **Note:** C is superfluous in R1, but R1 is in 3NF. However, D is not superfluous. Remove C from R_1 and get

R'_1 (A, B, D, E, F)

Note: Ling & Tompa & Kameda method removes **all** superfluous attributes.

❖ **Shortcoming 4.** The set of relations produced by the algorithm depends on the non-redundant covering found.

Example 6. Given $\mathbb{F}$ = {AD → B, B → C, C → D, AB → **E**, AC→ **F**, AD → **F**, AC → **E**}

Case 1 If $\mathbb{H}$ = {AD → B, B → C, C → D,AB → E, AC → F}

Then the set of relation is

R_1 (A , B , C , D , E , F)

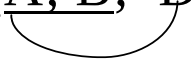
R_2 (B, C)

R_3 (C, D)

Case 2 If $\mathbb{H}$ = {AD → B, B → C, C → D, AB → E, AD → F }

Then the set of relations is

R'_1 (A, B, D, E, F)



R_2 (B, C)

R_3 (C, D)

Case 3 If $\mathbb{H} = \{AD \rightarrow B, B \rightarrow C, C \rightarrow D, AC \rightarrow F, AC \rightarrow E\}$

Then we have

R''_1 (<u>A, C,</u> D, B, E, F)

R_2 (<u>B,</u> C)

R_3 (<u>C,</u> D

- **Note** that AB is a key but it is not found by the algorithm.

Case 4 If $\mathbb{H} = \{AD \rightarrow B, B \rightarrow C, C \rightarrow D, AC \rightarrow E, AD \rightarrow F\}$

Then we have

R'''_1 (<u>A, C,</u> D, B, E, F)

R_2 (<u>B,</u> C)

R_3 (<u>C,</u> D)

- **Note** that AB is a key but it is not found by the algorithm.

Note that Case 2 gives the best solution. What is the meaning?

❖ **Shortcoming 5.** A BCNF relation set may contain **superfluous attributes**, i.e. redundant attributes which can be removed.

Example: Given a set of relations

R_1 (Model#, Serial#, **Price**, Color)
R_2 (Model#, Name)
R_3 (Serial#, Year)
R_4 (Name, Year, Price)

❖ **Note:** All relations are in BCNF, but R_1 contains a superfluous attribute **Price**, i.e. Price can be removed from R_1 without losing any information. How to prove it?

❖ **Note:** 3NF and BCNF are defined for **individual relations** but not the **whole relational schema**.

Ref: Ling, Tompa, & Kameda method takes the **whole relational schema** into consideration and removes superfluous attributes.

Note. Some relations generated by Step 6 may have more than one key. We need to choose their preliminary key. Why and how to choose?

- **Q:** Any impact on other relations after choosing primary key for some relation which has more than one key?

E.g. A database schema generated by Bernstein's Algorithm has the below relations:

Student (NRIC, S#, Name, DOB)
Course (C#, Title, Desc)
Take (NRIC, C#, Grade)

Note that Student relation has two keys, i.e. NRIC and S#. We choose S# as its preliminary key, and we also need to change NRIC in Take relation to S# and the relation Take becomes

Take (S#, C#, Grade)

Q: Why?