> *result* := $\{R\}$;
> *done* := false;
> **while** (**not** *done*) **do**
>   **if** (there is a schema R_i in *result* that is not in BCNF)
>   **then begin**
>   let $\alpha \rightarrow \beta$ be a nontrivial functional dependency that holds
>   on R_i such that α^+ does not contain R_i and $\alpha \cap \beta = \emptyset$;
>   *result* := $(result - R_i) \cup (R_i - \beta) \cup (\alpha, \beta)$;
>   **end**
>   **else** *done* := true;

Figure 7.11 BCNF decomposition algorithm.

Unfortunately, the latter procedure does not work when a relation schema is decomposed. That is, it *does not* suffice to use F when we test a relation schema R_i, in a decomposition of R, for violation of BCNF. For example, consider relation schema (A, B, C, D, E), with functional dependencies F containing $A \rightarrow B$ and $BC \rightarrow D$. Suppose this were decomposed into (A, B) and (A, C, D, E). Now, neither of the dependencies in F contains only attributes from (A, C, D, E), so we might be misled into thinking that it is in BCNF. In fact, there is a dependency $AC \rightarrow D$ in F^+ (which can be inferred using the pseudotransitivity rule from the two dependencies in F) that shows that (A, C, D, E) is not in BCNF. Thus, we may need a dependency that is in F^+, but is not in F, to show that a decomposed relation is not in BCNF.

An alternative BCNF test is sometimes easier than computing every dependency in F^+. To check if a relation schema R_i in a decomposition of R is in BCNF, we apply this test:

- For every subset α of attributes in R_i, check that α^+ (the attribute closure of α under F) either includes no attribute of $R_i - \alpha$, or includes all attributes of R_i.

If the condition is violated by some set of attributes α in R_i, consider the following functional dependency, which can be shown to be present in F^+:

$$\alpha \rightarrow (\alpha^+ - \alpha) \cap R_i.$$

This dependency shows that R_i violates BCNF.

7.5.1.2 BCNF Decomposition Algorithm

We are now able to state a general method to decompose a relation schema so as to satisfy BCNF. Figure 7.11 shows an algorithm for this task. If R is not in BCNF, we can decompose R into a collection of BCNF schemas $R_1, R_2, \ldots, R_n$ by the algorithm.