The set of restrictions $F_1, F_2, \ldots, F_n$ is the set of dependencies that can be checked efficiently. We now must ask whether testing only the restrictions is sufficient. Let $F' = F_1 \cup F_2 \cup \cdots \cup F_n$. F' is a set of functional dependencies on schema R, but, in general, $F' \neq F$. However, even if $F' \neq F$, it may be that $F'^+ = F^+$. If the latter is true, then every dependency in F is logically implied by F', and, if we verify that F' is satisfied, we have verified that F is satisfied. We say that a decomposition having the property $F'^+ = F^+$ is a **dependency-preserving decomposition**.

Figure 7.10 shows an algorithm for testing dependency preservation. The input is a set $D = \{R_1, R_2, \ldots, R_n\}$ of decomposed relation schemas, and a set F of functional dependencies. This algorithm is expensive since it requires computation of F^+. Instead of applying the algorithm of Figure 7.10, we consider two alternatives.

First, note that if each member of F can be tested on one of the relations of the decomposition, then the decomposition is dependency preserving. This is an easy way to show dependency preservation; however, it does not always work. There are cases where, even though the decomposition is dependency preserving, there is a dependency in F that cannot be tested in any one relation in the decomposition. Thus, this alternative test can be used only as a sufficient condition that is easy to check; if it fails we cannot conclude that the decomposition is not dependency preserving; instead we will have to apply the general test.

We now give a second alternative test for dependency preservation that avoids computing F^+. We explain the intuition behind the test after presenting the test. The test applies the following procedure to each $\alpha \rightarrow \beta$ in F.

result = α
repeat
    for each R_i in the decomposition
        $t = (result \cap R_i)^+ \cap R_i$
        $result = result \cup t$
until (*result* does not change)

The attribute closure here is under the set of functional dependencies F. If *result* contains all attributes in β, then the functional dependency $\alpha \rightarrow \beta$ is preserved. The decomposition is dependency preserving if and only if the procedure shows that all the dependencies in F are preserved.

The two key ideas behind the preceding test are as follows:

- The first idea is to test each functional dependency $\alpha \rightarrow \beta$ in F to see if it is preserved in F' (where F' is as defined in Figure 7.10). To do so, we compute the closure of α under F'; the dependency is preserved exactly when the closure includes β. The decomposition is dependency preserving if (and only if) all the dependencies in F are found to be preserved.