Chapter 9

Summarizing Itemsets

The search space for frequent itemsets is usually very large and, as discussed in Chapter 8, grows exponentially with the number of items, and we discussed several strategies for minimizing the various costs while mining itemsets. Although effective in several cases, these strategies do not reduce the computational complexity of the problem, since a low support may result in an intractable number of frequent itemsets. An alternate approach, studied in in this chapter, is to determine condensed representations of the frequent itemsets, that is, representations that summarize the characteristics of the set of itemsets partially or totally. A condensed representation aims to remove all redundant information in the frequent itemsets and the representation may be much smaller than the original frequent itemsets. The use of condensed representations not only reduces the computational and storage demands, but also makes it easier to analyze the correlations found.

There are two main issues in summarizing itemsets using such representation. The first is the representation's ability to recover the frequent itemsets and the second is the algorithm that generates such representations.

In this chapter we discuss three of these representations: closed, maximal, and non-derivable itemsets.

9.1 Maximal and Closed Frequent Itemsets

Given a database over the tids $\mathcal{T}$, and items $\mathcal{I}$, let $\mathcal{F}$ denote the set of all frequent itemsets, i.e.,

$$\mathcal{F} = \{X \mid X \subseteq \mathcal{I} \text{ and } sup(X) \geq minsup\} \tag{9.1}$$

Example 9.1: Consider the dataset given in Figure 9.1a. Using any of the algorithms discussed in Chapter 8 and $minsup = 3$, we obtain the frequent itemsets shown in Figure 9.1b. Notice that there are 19 frequent itemsets out of the 31

1	ABDE
2	BCE
3	ABDE
4	ABCE
5	ABCDE
6	BCD

(a) Transaction Database

sup	itemsets
6	B
5	E, BE
4	A, C, D, AB, AE, BC, BD, ABE
3	AD, CE, DE, ABD, ADE, BCE, BDE, ABDE

(b) Frequent Itemsets ($minsup = 3$)

Figure 9.1: An Example Database

possible ones (not including the empty itemset), and we want to summarize them, as discussed next.

Maximal Frequent Itemsets A frequent itemset $X \in \mathcal{F}$ is called *maximal* if it has no frequent supersets. Let $\mathcal{M}$ be the set of all maximal frequent itemsets, which is given as

$$\mathcal{M} = \{X \mid X \in \mathcal{F} \text{ and } \nexists Y \supset X, \text{ such that } Y \in \mathcal{F}\} \tag{9.2}$$

Example 9.2: In the example given in Figure 9.1 there are just two maximal itemsets, ABDE and BCE. Notice how a frequent itemset must be a subset of one of the maximal itemsets.

Closed Frequent Itemsets Recall from Chapter 8 the function $\mathbf{t} : 2^{\mathcal{I}} \to 2^{\mathcal{T}}$ that maps itemsets to tidsets (8.2), and the function $\mathbf{i} : 2^{\mathcal{T}} \to 2^{\mathcal{I}}$ that maps tidsets to itemsets (8.1), defined as follows:

$$\mathbf{t}(X) = \{t \mid X \subseteq \mathbf{i}(t)\} \tag{9.3}$$

$$\mathbf{i}(T) = \{x \in \mathcal{I} \mid \forall t \in T,\ t \text{ contains } x\} \tag{9.4}$$

Define by $\mathbf{c} : 2^{\mathcal{I}} \to 2^{\mathcal{I}}$ the *closure operator*, defined as

$$\mathbf{c}(X) = \mathbf{i} \circ \mathbf{t}(X) = \mathbf{i}(\mathbf{t}(X)) \tag{9.5}$$

The closure operator $\mathbf{c}$ maps itemsets to itemsets, and it is:

1. *Extensive*: $X \subseteq \mathbf{c}(X)$
2. *Monotonic*: If $X_i \subseteq X_j$, then $\mathbf{c}(X_i) \subseteq \mathbf{c}(X_j)$

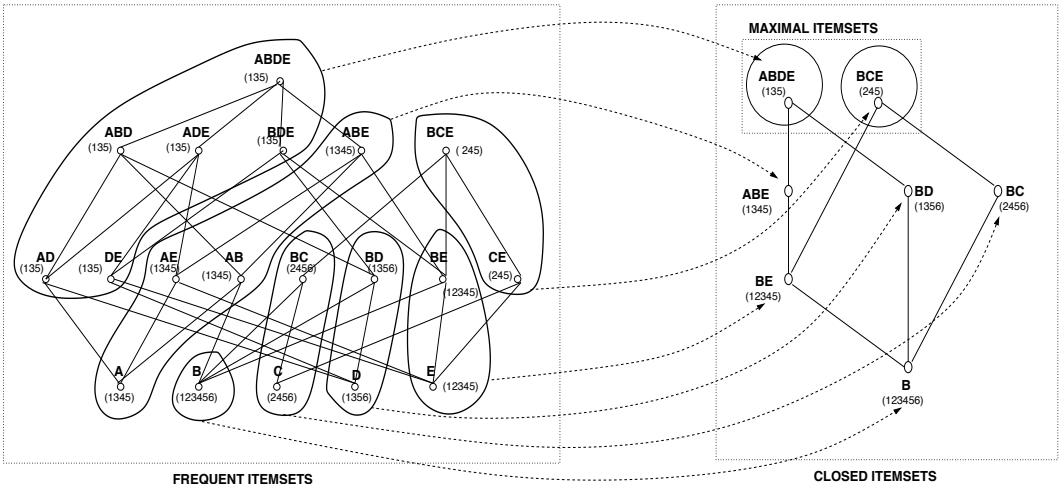

Figure 9.2: Frequent, Closed Frequent and Maximal Frequent Itemsets

3. *Idempotent*: $\mathbf{c}(\mathbf{c}(X)) = \mathbf{c}(X)$

An itemset X is called *closed* if $\mathbf{c}(X) = X$, i.e., if X is a fixed-point of the closure operator $\mathbf{c}$. On the other hand if $X \neq \mathbf{c}(X)$, then X is not closed, but the set $\mathbf{c}(X)$ is called its closure. From the properties of the closure operator, both X and $\mathbf{c}(X)$ have the same tidset. It follows that a frequent set $X \in \mathcal{F}$ is closed if it has no frequent superset *with the same frequency*, since by definition, it is the largest itemset that appears in the tidset $\mathbf{t}(X)$. Let $\mathcal{C}$ be the set of all closed frequent itemsets, which can be written as

$$\mathcal{C} = \{X \mid X \in \mathcal{F} \text{ and } \nexists Y \supset X \text{ with } sup(X) = sup(Y)\} \tag{9.6}$$

Example 9.3: The closed (as well as maximal) itemsets of the database given in Figure 9.1a with $minsup = 3$ are given in Figure 9.2. We can see, for instance, that the itemsets AD, DE, ABD, ADE, BDE, and $ABDE$, occur in the same three transactions ($\{1, 3, 5\}$), and thus the largest itemset among these, namely $ABDE$, is the closed itemset. Using the closure operator yields the same result. For example, $\mathbf{c}(AD) = \mathbf{i}(\mathbf{t}(AD)) = \mathbf{i}(135) = ABDE$, which indicates that the closure of AD is $ABDE$.

The following relationship holds between the set all, closed, and maximal frequent itemsets: $\mathcal{M} \subseteq \mathcal{C} \subseteq \mathcal{F}$, which is illustrated in Figure 9.2. We can see clearly the *equivalence classes* of itemsets that have the same tidsets; the largest itemset in each class is a closed itemset. We can also see that the maximal itemsets are a subset of the closed itemsets.

9.1.1 Mining Maximal Frequent Itemsets

Mining maximal itemsets requires additional steps beyond simply determining the frequent itemsets. Assuming that the set of maximal frequent itemsets is initially empty, i.e., $\mathcal{M} = \emptyset$, each time we generate a new frequent itemset X, we have to perform the following maximality checks:

1. **Subset Check:** $\nexists Y \in \mathcal{M}$, such that $X \subset Y$. If such a Y exists, then clearly X is not maximal. Otherwise, we add X to $\mathcal{M}$, as a potentially maximal itemset.

2. **Superset Check:** $\nexists Y \in \mathcal{M}$, such that $Y \subset X$. If such a Y exists, then Y cannot be maximal, and we have to remove it from $\mathcal{M}$.

These two maximality checks take $O(|\mathcal{M}|)$ time, which can get expensive, especially as $\mathcal{M}$ grows, thus for efficiency reasons it is crucial to minimize the number of times these checks are performed.

Any of the frequent itemset mining methods we considered in Chapter 8 can be extended to mine maximal frequent itemsets by adding the maximality checking steps. Here we consider the GenMax method, that is based on the tidset intersection approach. We shall see that it never inserts a non-maximal itemset into $\mathcal{M}$. It thus eliminates the superset checks and requires only subset checks to determine maximality.

Algorithm 9.1 shows the pseudo-code for GenMax. The initial call takes as input the set of frequent items along with their tidsets, $\langle i, \mathbf{t}(i)\rangle$, and the initially empty set of maximal itemsets, $\mathcal{M}$. Given a set of IT-pairs $\langle X, \mathbf{t}(X)\rangle$, the recursive GenMax method works as follows. In lines 1-3, we check if the entire current branch can be pruned by checking if the union of all the itemsets, $Y = \bigcup X_i$, is already subsumed by (or contained in) some maximal pattern $Z \in \mathcal{M}$. If so, no maximal itemset can be generated from the current branch, and it is pruned. On the other hand, if the branch is not pruned, we intersect each IT-pair $\langle X_i, \mathbf{t}(X_i)\rangle$ with all the other IT-pairs $\langle X_j, \mathbf{t}(X_j)\rangle$ to generate new candidates N_{ij}, which are added to the IT-pair set P_i (lines 5-10). If P_i is not empty, a recursive call to GenMax is made in a DFS manner to find other potentially frequent extensions of X_i (line 11-12). On the other hand, if P_i is empty, it means that X_i cannot be extended, and it is potentially maximal. In this case, we add X_i to the set $\mathcal{M}$, provided that X_i is not contained in any previously added maximal set $Z \in \mathcal{M}$ (lines 14-15). Note also that because of this check for maximality before inserting any itemset into $\mathcal{M}$, we never have to remove any itemsets from it. In other words, all itemsets in $\mathcal{M}$ are guaranteed to be maximal. Upon termination of GenMax, the set $\mathcal{M}$ contains the final set of all maximal frequent itemsets. The GenMax approach also includes a number of other optimizations to reduce the maximality checks, and to improve the support computations. We omit them here to retain clarity.

Algorithm 9.1: Algorithm GenMax

```
// Initial Call:
    GenMax({⟨i, t(i)⟩ : i ∈ I, sup(i) ≥ minsup}, minsup, M = ∅)
GenMax (P = ⟨X_i, t(X_i)⟩, minsup, M):
1  Y = ⋃ X_i
2  if ∃Z ∈ M, Y ⊆ Z then
3      return // Prune entire branch
4  foreach ⟨X_i, t(X_i)⟩ ∈ P do
5      P_i ← ∅
6      foreach ⟨X_j, t(X_j)⟩ ∈ P, with j > i do
7          N_ij = X_i ∪ X_j
8          t(N_ij) = t(X_i) ∩ t(X_j)
9          if sup(N_ij) ≥ minsup then
10             P_i ← P_i ∪ {⟨N_ij, t(N_ij)⟩}
11     if P_i ≠ ∅ then
12         GenMax(P_i, minsup, M)
13     else
14         if ∄Z ∈ M, X_i ⊆ Z then
15             M = M ∪ X_i    // Add X_i to maximal set
```

Example 9.4: Figure 9.3 shows the execution of GenMax on the example database given in Figure 9.1a using $minsup = 3$. The root of the tree represents the initial call of GenMax with all IT-pairs consisting of single items and their tidsets. As we proceed in a depth-first manner, we reach the left-most leaf $ABDE$, which cannot be extended further. Since $\mathcal{M}$ is also empty, $ABDE$ is clearly maximal and this we make $\mathcal{M} = \{ABDE\}$. When we try to process the ADE branch we find that it is subsumed by $ABDE$, and we thus prune it. Likewise AE gets pruned. At this stage, all maximal itemsets starting with A have been found, and we proceed with the B branch. The left-most B branch, namely BCE cannot be extended further. Since BCE is not a subset of any maximal itemset in $\mathcal{M}$, we insert it as a maximal itemset, so that $\mathcal{M} = \{ABDE, BCE\}$. It is easy to see that from this point on, all other branches are subsumed by one of the maximal itemsets, and are thus pruned.

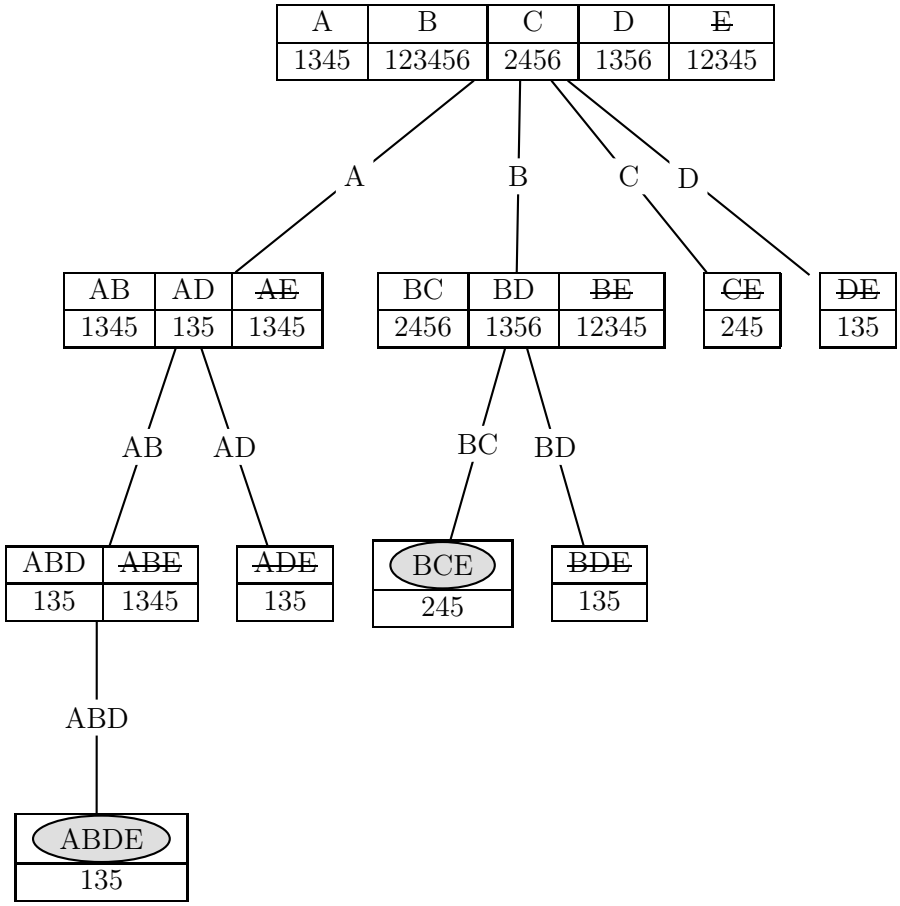

Figure 9.3: Mining Maximal Frequent Itemsets. Maximal itemsets are shown as shaded ovals, whereas pruned branches are shown with the strike-through. Infrequent itemsets are not shown.

9.1.2 Mining Closed Frequent Itemsets

Mining closed frequent itemsets requires that we perform closure checks, i.e., $X = \mathbf{c}(X)$. Direct closure checking can be very expensive, since we would have to verify that X is the largest itemset common to all the tids in $\mathbf{t}(X)$, i.e., $X = \bigcap_{t \in \mathbf{t}(X)} \mathbf{i}(t)$. Instead, we will describe a vertical tidset intersection based method called CHARM that performs more efficient closure checking. Given a collection of IT-pairs $\{\langle X_i, \mathbf{t}(X_I)\rangle\}$, the following three properties hold:

1. If $\mathbf{t}(X_i) = \mathbf{t}(X_j)$, then $\mathbf{c}(X_i) = \mathbf{c}(X_j) = \mathbf{c}(X_i \cup X_j)$

   In other words, we can replace every occurrence of X_i with $X_i \cup X_j$ and prune the branch under X_j, since its closure is identical to the closure of $X_i \cup X_j$.

2. If $\mathbf{t}(X_i) \subset \mathbf{t}(X_j)$, then $\mathbf{c}(X_i) \neq \mathbf{c}(X_j)$ but $\mathbf{c}(X_i) = \mathbf{c}(X_i \cup X_j)$

   This property means that we can replace every occurrence of X_i with $X_i \cup X_j$, but we cannot prune X_j since it generates a different closure. Note that if $\mathbf{t}(X_i) \supset \mathbf{t}(X_j)$ then we simply interchange the role of X_i and X_j.

3. If $\mathbf{t}(X_i) \neq \mathbf{t}(X_j)$, then $\mathbf{c}(X_i) \neq \mathbf{c}(X_j) \neq \mathbf{c}(X_i \cup X_j)$

   This property means that we cannot remove either X_i or X_j, since each of them generates a different closure.

Algorithm 9.2: Algorithm CHARM

```
// Initial Call:
    CHARM({⟨i, t(i)⟩ : i ∈ I, sup(i) ≥ minsup}, minsup, C = ∅)
CHARM (P = ⟨X_i, t(X_i)⟩, minsup, C):
1  Sort P in increasing order of support (i.e., by increasing |t(X_i)|)
2  foreach ⟨X_i, t(X_i)⟩ ∈ P do
3      P_i ← ∅
4      foreach ⟨X_j, t(X_j)⟩ ∈ P, with j > i do
5          N_ij = X_i ∪ X_j
6          t(N_ij) = t(X_i) ∩ t(X_j)
7          if sup(N_ij) ≥ minsup then
8              if t(X_i) = t(X_j) then
                   // Property 1
9                  Replace X_i with N_ij in P and P_i
10                 Remove ⟨X_j, t(X_j)⟩ from P
11             else if t(X_i) ⊂ t(X_j) then
                   // Property 2
12                 Replace X_i with N_ij in P and P_i
13             else
                   // Property 3:  t(X_i) ≠ t(X_j)
14                 P_i ← P_i ∪ {⟨N_ij, t(N_ij)⟩}
15     if P_i ≠ ∅ then
16         CHARM(P_i, minsup, C)
17     if ∄Z ∈ C, X_i ⊆ Z and t(X_i) = t(Z) then
18         C = C ∪ X_i    // Add X_i to closed set
```

Algorithm 9.2 presents the pseudo-code for CHARM. Just like the other tidset intersection based methods, it initially takes as input the set of all frequent single items, along with their tidsets. Initially $\mathcal{C}$, the set of all closed itemsets, is empty.

Given any IT-pair set $P = \{\langle X_i, \mathbf{t}(X_i)\rangle\}$, the method first sorts them by increasing order of support ($|\mathbf{t}(X_i)|$) in line 1. For each itemset X_i we try to extend it with all other items X_j in the sorted order, and we apply the above three properties to prune branches where possible. First we make sure that $N_{ij} = X_i \cup X_j$ is frequent (lines 5-7). If yes, then we check properties 1 and 2 in lines 8-12. Note that whenever we replace X_i with $N_{ij} = X_i \cup X_j$, we make sure to do so in the current set P, as well as the new set P_i. Only when property 3 holds, do we add the new extension N_{ij} to the set P_i (lines 13-14). If the set P_i is not empty, then we make a recursive call to the method (lines 15-16). Finally, if X_i is not a subset of any closed set Z with the same support, we can safely add it to the set of closed itemsets, $\mathcal{C}$ (lines 17-18).

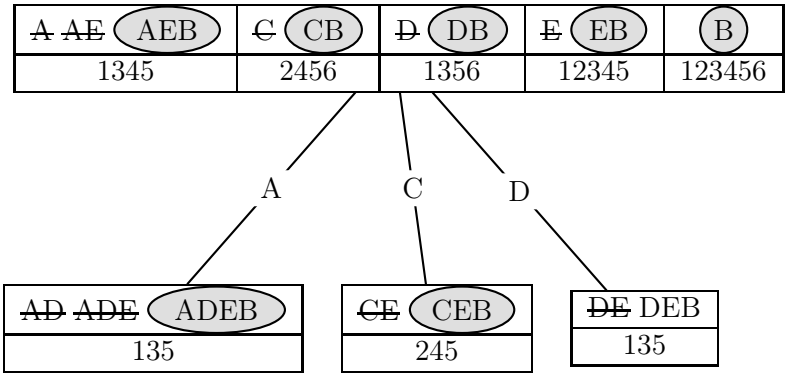

Figure 9.4: Mining Closed Frequent Itemsets. Closed itemsets are shown as shaded ovals. Strike-through represents itemsets X_i replaced by $X_i \cup X_j$ during execution of the algorithm. Infrequent itemsets are not shown.

Example 9.5: Let us consider how CHARM mines the closed itemsets from the example database in Figure 9.1a, using $minsup = 3$. Figure 9.4 shows the sequence of steps. The initial set of IT-pairs, after support based sorting, is shown at the root of the search tree. The sorted order is A, C, D, E, and B. First A is combined with C, but AC is not frequent. AD is frequent and since $\mathbf{t}(A) \neq \mathbf{t}(D)$, we add $\langle AD, 135\rangle$ to the set P_A (Property 3). When we combine A with E, property 2 applies, and we simply replace all occurrences of A in P and P_A with AE, which is illustrated with the strike-through. Likewise, since $\mathbf{t}(A) \subset \mathbf{t}(B)$ all current occurrences of A (actually AE) are replaced by AEB. The set P_A thus contains only one itemset $\{\langle ADEB, 135\rangle\}$. When CHARM is invoked with P_A as the IT-pair, it jumps straight to line 17, and adds $ADEB$ to $\mathcal{C}$, as a newly found closed itemset. When the call returns, we check whether AEB can be added as a closed itemset. AEB is a subset of $ADEB$, but it does not have the same support, thus AEB is also added to $\mathcal{C}$. At this point all closed itemsets containing A have been found.

Next we process the C branch. CD is infrequent and thus pruned. CE is frequent and it is added to P_C as a new extension (via property 3). Since $\mathbf{t}(C) \subset \mathbf{t}(B)$, all occurrences of C are replaced by CB, and $P_C = \{\langle CEB, 245\rangle\}$. CBE and CB are both found to be closed. The computation proceeds in this manner until all closed frequent itemsets are found. Note that when we get to DEB and perform the closure check, we find that it is a subset of the closed $ADEB$ and also has the same support, thus DEB is not closed.

9.2 Non-Derivable Itemsets

An itemset is said to be *non-derivable* if its support cannot be derived from some combination of the support of its subsets. The set of all frequent non-derivable itemsets ($NDI \subseteq \mathcal{F}$) are a summary or condensed representation of the set of all frequent itemsets. Furthermore, NDI is lossless with respect to support, that is, the exact support of all other frequent itemsets can be deduced from it.

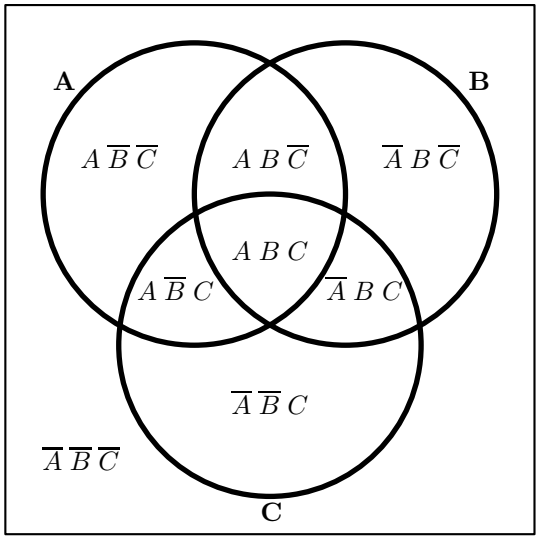

Figure 9.5: Tid Partition Induced by Three Items

Let $\mathcal{T}$ be a set of tids, and let $\mathcal{I}$ be a set of items. Let A, B, and C be any three items in $\mathcal{I}$. These three sets induce a partition on the space of all tids, as illustrated in the Venn Diagram of Figure 9.5. For example, the region labeled $A\,\overline{B}\,\overline{C}$ represents those tids that contain A but neither B nor C. In general, any set of items X, with size $|X| = k$, induces a partition with 2^k distinct regions.

Notice that each region is in fact a *generalized itemset*, which is defined as an itemset that may contain either an item or its negation. A generalized itemset can

be represented as $X\overline{Y}$, where X consists of regular items and Y consists of negated items. For example for $A\overline{BC}$, we have $X = A$ and $Y = BC$. Define the support of a generalized itemset $X\overline{Y}$ as the number of transactions that contain all items in X, but no items in Y, given as

$$sup(X\overline{Y}) = |\{t \in \mathcal{T} \mid X \subseteq \mathbf{i}(t) \text{ and } Y \cap \mathbf{i}(t) = \emptyset\}| \tag{9.7}$$

Consider how we can compute the support of $A\overline{BC}$ in Figure 9.5. We start with all the tids for A, and remove the tids contained in regions AB and AC. However, we realize that this removes the tids in ABC twice, so we need to add those back. In other words, the support of $A\overline{BC}$ is given as

$$sup(A\overline{BC}) = sup(A) - sup(AB) - sup(AC) + sup(ABC) \tag{9.8}$$

Notice that the support of $A\overline{BC}$ is given by some combination of the supports of the (regular) supersets of A that do not have any negated items. In fact the supersets are obtained by combining A with any subset of the set of negated items $\{B, C\}$. Another way to say this is that if we let $I = \{A\} \cup \{B, C\} = ABC$, then the support of $A\overline{BC}$ is given as a combination of itemsets in the set $\{J \mid A \subseteq J \subseteq ABC\}$.

Inclusion-Exclusion Principle

Let $X\overline{Y}$ be a generalized itemset, and let $I = X \cup Y$. Then the support of $X\overline{Y}$ can be expressed as some combination of the supports of supersets $J \supseteq X$, such that $J \subseteq I$. The precise formula is given by the well-known *inclusion-exclusion principle* in combinatorics:

$$sup(X\overline{Y}) = \sum_{X \subseteq J \subseteq I} (-1)^{|J \setminus X|} sup(J) \tag{9.9}$$

where $|J \setminus X| = J - X$. Let us verify this for $X\overline{Y} = A\overline{BC}$

$$\begin{aligned} sup(A\overline{BC}) &= (-1)^0\ sup(A) + && J = A, |J \setminus X| = 0 \\ &\quad (-1)^1\ sup(AB) + && J = AB, |J \setminus X| = 1 \\ &\quad (-1)^1\ sup(AC) + && J = AC, |J \setminus X| = 1 \\ &\quad (-1)^2\ sup(ABC) && J = ABC, |J \setminus X| = 2 \\ &= sup(A) - sup(AB) - sup(AC) + sup(ABC) \end{aligned} \tag{9.10}$$

As another example, the support of $X\overline{Y} = \overline{ABC}$, with $X = \emptyset$ and $Y = ABC$. Figure 9.6 shows the cardinality of $J \setminus X$ and the corresponding signs to use in the inclusion-exclusion formula, which yields the following combination:

$$\begin{aligned} sup(\overline{ABC}) &= sup(\emptyset) \\ &\quad - sup(A) - sup(B) - sup(C) \\ &\quad + sup(AB) + sup(AC) + sup(BC) \\ &\quad - sup(ABC) \end{aligned} \tag{9.11}$$

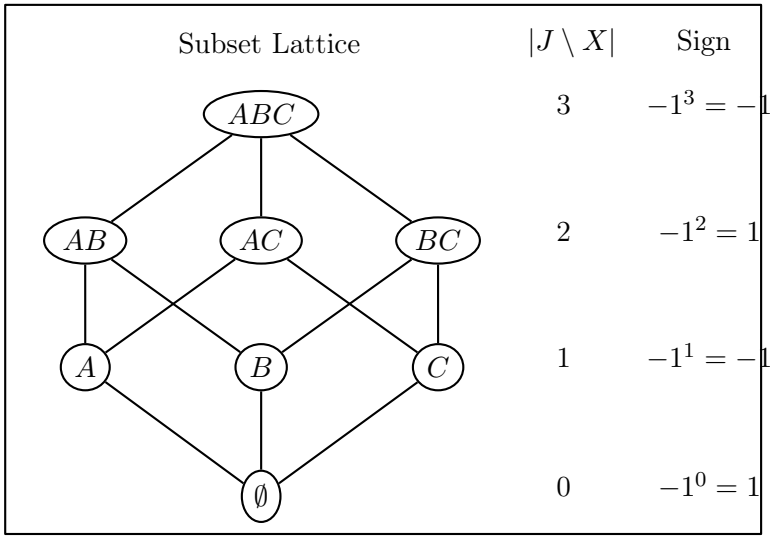

Figure 9.6: Subset Lattice and Signs for Different Levels

Support Bounds for an Itemset

Notice that the inclusion-exclusion formula for the support of $X\overline{Y}$, always starts at some itemset X and has terms for all subsets between X and $I = X \cup Y$. In other words, each of the eight regions in Figure 9.5 has a term for $sup(ABC)$. In general each of the $2^{|I|}$ regions in the partition induced by items in I has a term for I. Also note that the support of any generalized itemset must be at least zero (negative support would be a meaningless concept), i.e.,

$$sup(X\overline{Y}) \geq 0 \tag{9.12}$$

For example, setting $sup(A\overline{BC}) \geq 0$, gives us

$$\begin{aligned} sup(A\overline{BC}) &= sup(A) - sup(AB) - sup(AC) + sup(ABC) \geq 0, \text{ which implies} \\ sup(ABC) &\geq -sup(A) + sup(AB) + sup(AC) \end{aligned}$$

Notice that this rule gives a lower bound on the support of ABC.

As another example, setting $sup(\overline{ABC}) \geq 0$, yields

$$\begin{aligned} sup(\overline{ABC}) &= sup(\emptyset) - sup(A) - sup(B) - sup(C) + sup(AB) + sup(AC) + sup(BC) - \\ &\quad sup(ABC) \geq 0 \\ \text{or } sup(ABC) &\leq sup(\emptyset) - sup(A) - sup(B) - sup(C) + sup(AB) + sup(AC) + sup(BC) \end{aligned}$$

Notice that this rule gives an upper bound on the support of ABC.

In fact, from each of the regions in Figure 9.5, we get one rule, and out of the eight possible rules, exactly four give upper bounds and the other four give lower

bounds on the support of ABC, as shown below:

$$
\begin{array}{lll}
sup(ABC) & \geq 0 & \text{when } X = ABC \\
& \leq sup(AB) & \text{when } X = AB \\
& \leq sup(AC) & \text{when } X = AC \\
& \leq sup(BC) & \text{when } X = BC \\
& \geq sup(AB) + sup(AC) - sup(A) & \text{when } X = A \\
& \geq sup(AB) + sup(BC) - sup(B) & \text{when } X = B \\
& \geq sup(AC) + sup(BC) - sup(C) & \text{when } X = C \\
& \leq sup(AB) + sup(BC) + sup(AC) - & \\
& sup(A) - sup(B) - sup(C) + sup(\emptyset) & \text{when } X = \emptyset
\end{array}
$$

The above derivation rules are schematically summarized in Figure 9.7.

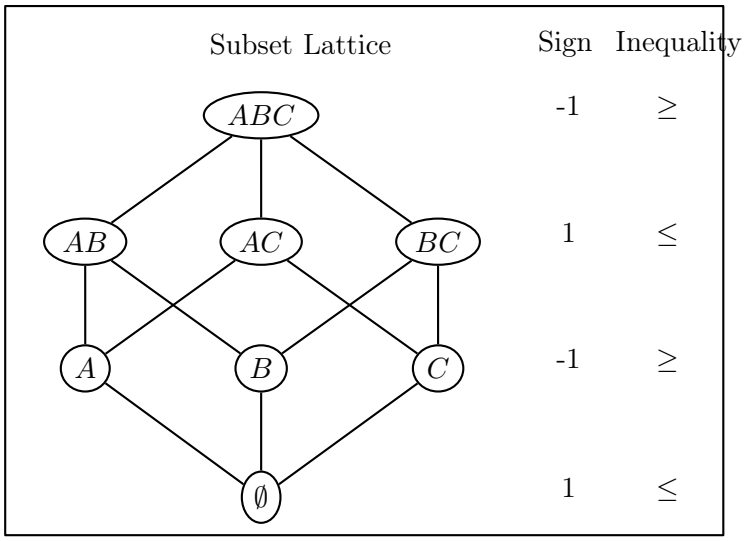

Figure 9.7: Derivation Rule Summary

In the general case, for any itemset of interest I, we can derive the rules for the upper and lower bound on its support, by using each set $X \subseteq I$ the starting point, and after rearranging the terms in the inclusion-exclusion formula (9.9), we get the following two general rules:

$$\textbf{Upper Bounds } (|I \setminus X| \text{ is odd}): \; sup(I) \leq \sum_{X \subseteq J \subseteq I} (-1)^{(|I \setminus J|+1)} sup(J) \quad (9.13)$$

$$\textbf{Lower Bounds } (|I \setminus X| \text{ is even}): \; sup(I) \geq \sum_{X \subseteq J \subseteq I} (-1)^{(|I \setminus J|+1)} sup(J) \quad (9.14)$$

Note that the only difference in the two equations is the inequality, which depends on the starting point X for the rule derivation.

Non-Derivable Itemsets

Since we have several derivation rules for the support lower and upper bounds for any itemset I, we can determine the *Least Upper Bound*, denoted $lub(I)$, among all the upper bounds, and the *Greatest Lower Bound*, denoted $glb(I)$, among all the lower bounds for $sup(I)$ from the different rules. It follows that $sup(I) \in [glb(I), lub(I)]$. In other words, the actual support of I must clearly be at least $glb(I)$, and at most $lub(I)$. If the $glb(I)$ and $lub(I)$ are identical, then $sup(I)$ can be derived exactly using the supports of its subsets, and I is called **derivable**. On the other hand, if $glb(I) \neq lub(I)$ then we cannot exactly determine the support of I from its subsets (we only know the range), and in this case I is called **non-derivable**. Thus the set of all **Non-Derivable Itemsets (NDI)** is given as follows:

$$NDI = \{I \in \mathcal{F} \mid glb(I) \neq lub(I)\} \tag{9.15}$$

where $\mathcal{F}$ is the set of all frequent itemsets (for a given $minsup$ threshold).

Example 9.6: Consider the database shown in Figure 9.1a, and the corresponding set of frequent itemsets $\mathcal{F}$ shown in Figure **??**. Here are a few examples of how to derive the support bounds for a given itemset. For example for $I = A$, we have

$$\begin{aligned} sup(A) &\leq sup(\emptyset) = 6 \\ &\geq 0 \end{aligned}$$

Thus $sup(A) \in [0, 6]$ and A is non-derivable.

For $I = AB$, we have

$$\begin{aligned} sup(AB) &\leq sup(A) = 4 \\ &\leq sup(B) = 6 \\ &\geq 0 \\ &\geq sup(A) + sup(B) - sup(\emptyset) = 4 + 6 - 6 = 4 \end{aligned}$$

Thus $sup(AB) \in [4, 4]$, or $sup(AB) = 4$, which means that AB is derivable.

As another example for $I = ABDE$, we get the following upper bounds:

$$\begin{aligned} sup(ABDE) &\leq sup(ABD) = 3 \\ &\leq sup(ABE) = 4 \\ &\leq sup(ADE) = 3 \\ &\leq sup(BDE) = 3 \\ sup(ABDE) &\leq sup(A) - sup(AB) - sup(AD) - sup(AE) + \\ &\quad sup(ABD) + sup(ABE) + sup(ADE) \\ &= 4 - (4 + 3 + 4) + (3 + 4 + 3) = 3 \end{aligned}$$

From these upper bounds, we know that $sup(ABDE) \leq 3$. Let's consider the lower bound rule derived from AB, we get:

$$\begin{aligned} sup(ABDE) &\geq sup(ABD) + sup(ABE) - sup(AB) \\ &= 3 + 4 - 4 = 3 \end{aligned}$$

At this point we know that $sup(ABDE) \geq 3$, so without processing any further rules, we can immediately conclude that $sup(ABDE) \in [3, 3]$, which means that $ABDE$ is derivable.

For the example database in Figure 9.1a, the set of non-derivable itemsets (with the support bounds) is given as:

$$NDI = \{A[0,6], B[0,6], C[0,6], D[0,6], E[0,6], AD[2,4], AE[3,4], CE[3,4], DE[3,4]\}$$

We leave it as an exercise to develop an efficient algorithm to mine all the non-derivable frequent itemsets.

9.3 Annotated References

9.4 Exercises and Projects

1. Consider the vertical database shown, and then answer the questions below.

A	B	C	D	E
1	2	1	1	2
3	3	2	6	3
5	4	3		4
6	5	5		5
	6	6		

   (a) Using intersections of tidlists, find all frequent itemsets with minimum support of 3 (show the tidlist for each itemset).

   (b) List all closed frequent sets.

   (c) List all maximal frequent and minimal infrequent sets.

2. Consider the database shown below:

   Let D denote the full database, which consists of 10 transactions. Let S denote a random sample from D, with 5 transactions, consisting of the following Tids $\{2, 3, 4, 7, 9\}$. Let minimum support be 40%. Answer the following questions: