Association rule mining

we will now introduce a new kind of rule that can be learned in a wholly unsupervised manner and is prominent in data mining applications.

Suppose we observed eight customers who each bought one or more of apples, beer, crisps and nappies:

Transaction	*Items*
1	nappies
2	beer, crisps
3	apples, nappies
4	beer, crisps, nappies
5	apples
6	apples, beer, crisps, nappies
7	apples, crisps
8	crisps

Each *transaction* in this table involves a set of *items*; conversely, for each item we can list the transactions in which it was involved:

transactions 1, 3, 4 and 6 for nappies, transactions 3, 5, 6 and 7 for apples, and so on.

We can also do this for sets of items: e.g., beer and crisps were bought together in transactions 2, 4 and 6; we say that item set {beer, crisps} *covers* transaction set {2, 4, 6}.

There are 16 of such item sets (including the empty set, which covers all transactions); using the subset relation between transaction sets as partial order, they forma lattice (Figure 6.17).

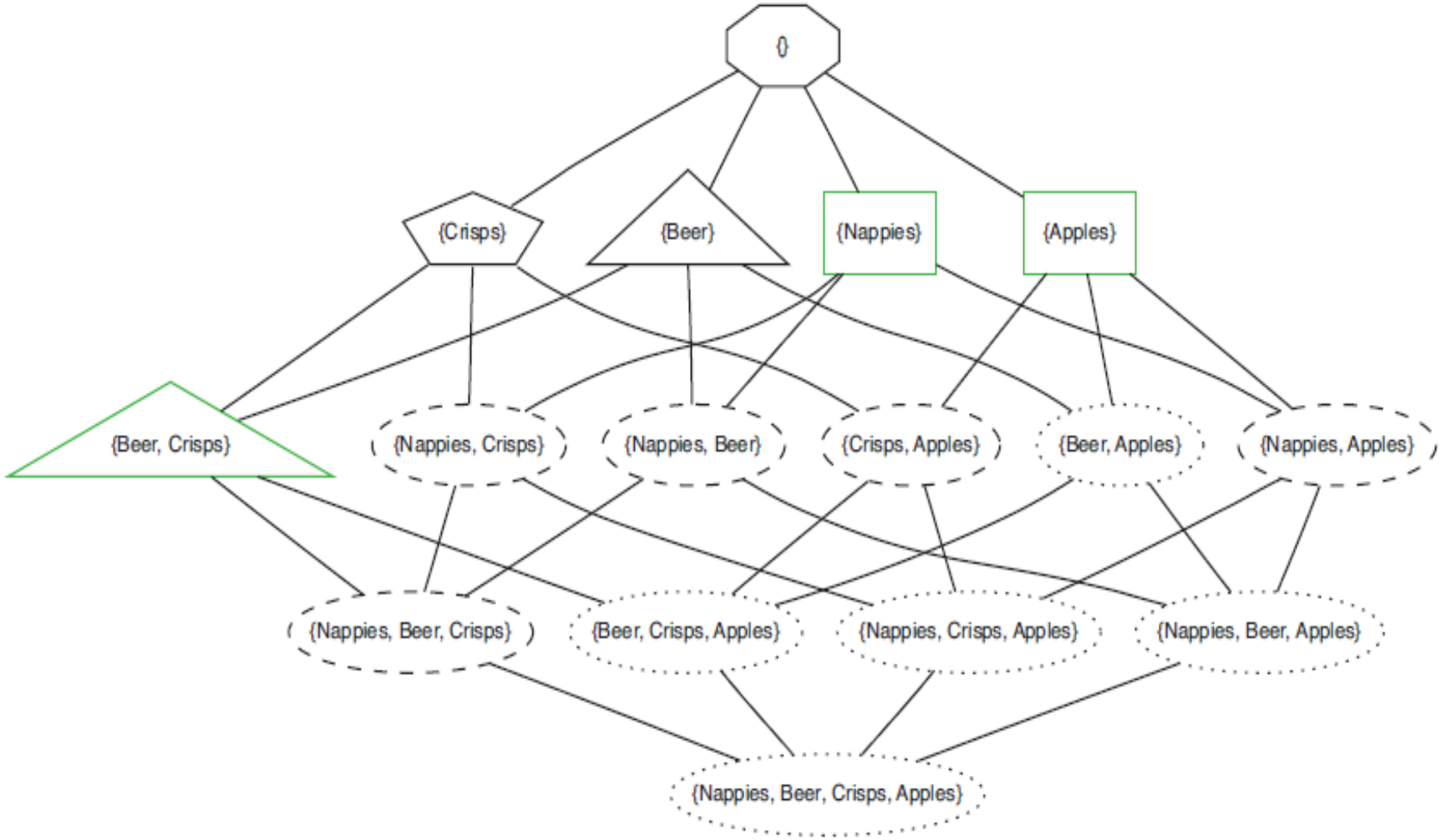

Figure 6.17. An item set lattice. Item sets in dotted ovals cover a single transaction; in dashed ovals, two transactions; in triangles, three transactions; and in polygons with n sides, n transactions. The maximal item sets with support 3 or more are indicated in green.

Let us call the number of transactions covered by an item set *I* its *support*, denoted Supp(*I*) (sometimes called frequency).

We are interested in *frequent item sets*, which exceed a given support threshold *f*0. Support is *monotonic*: when moving down a path in the item set lattice it can never increase.

This means that the set of frequent item sets is *convex* and is fully determined by its lower boundary of largest item sets: in the example these maximal1 frequent item sets are, for *f*0 = 3: {apples}, {beer, crisps} and {nappies}.So, at least three transactions involved apples; at least three involved nappies; at least three involved both beer and crisps; and any other combination of items was bought less often.

Because of the monotonicity property of item set support, frequent item sets can be found by a simple enumerative breadth-first or level-wise search algorithm (Algorithm 6.6). The algorithm maintains a priority queue, initially holding only the empty itemset which covers all transactions. Taking the next candidate item set *I* off the priority queue, it generates all its possible extensions

Algorithm 6.6: FrequentItems(D, f_0) – find all maximal item sets exceeding a given support threshold.

```
   Input  : data D ⊆ 𝒳; support threshold f0.
   Output : set of maximal frequent item sets M.
1  M ← ∅;
2  initialise priority queue Q to contain the empty item set;
3  while Q is not empty do
4      I ← next item set deleted from front of Q;
5      max ← true ;                    // flag to indicate whether I is maximal
6      for each possible extension I′ of I do
7          if Supp(I′) ≥ f0 then
8              max ← false ;           // frequent extension found, so I is not maximal
9              add I′ to back of Q;
10         end
11     end
12     if max = true then M ← M ∪ {I};
13 end
14 return M
```

We can speed up calculations by restricting attention to *closed item sets*. A closed item set contains all items that are involved in every transaction it covers. For example, {beer, crisps} covers transactions 2, 4 and 6; the only items involved in each of those transactions are beer and crisps, and so the item set is closed.

However, {beer} is not closed, as it covers the same transactions, hence its closure is {beer, crisps}. If two item sets that are connected in the lattice have the same coverage, the smaller item set cannot be closed. The lattice of closed item sets is shown in Figure 6.18.

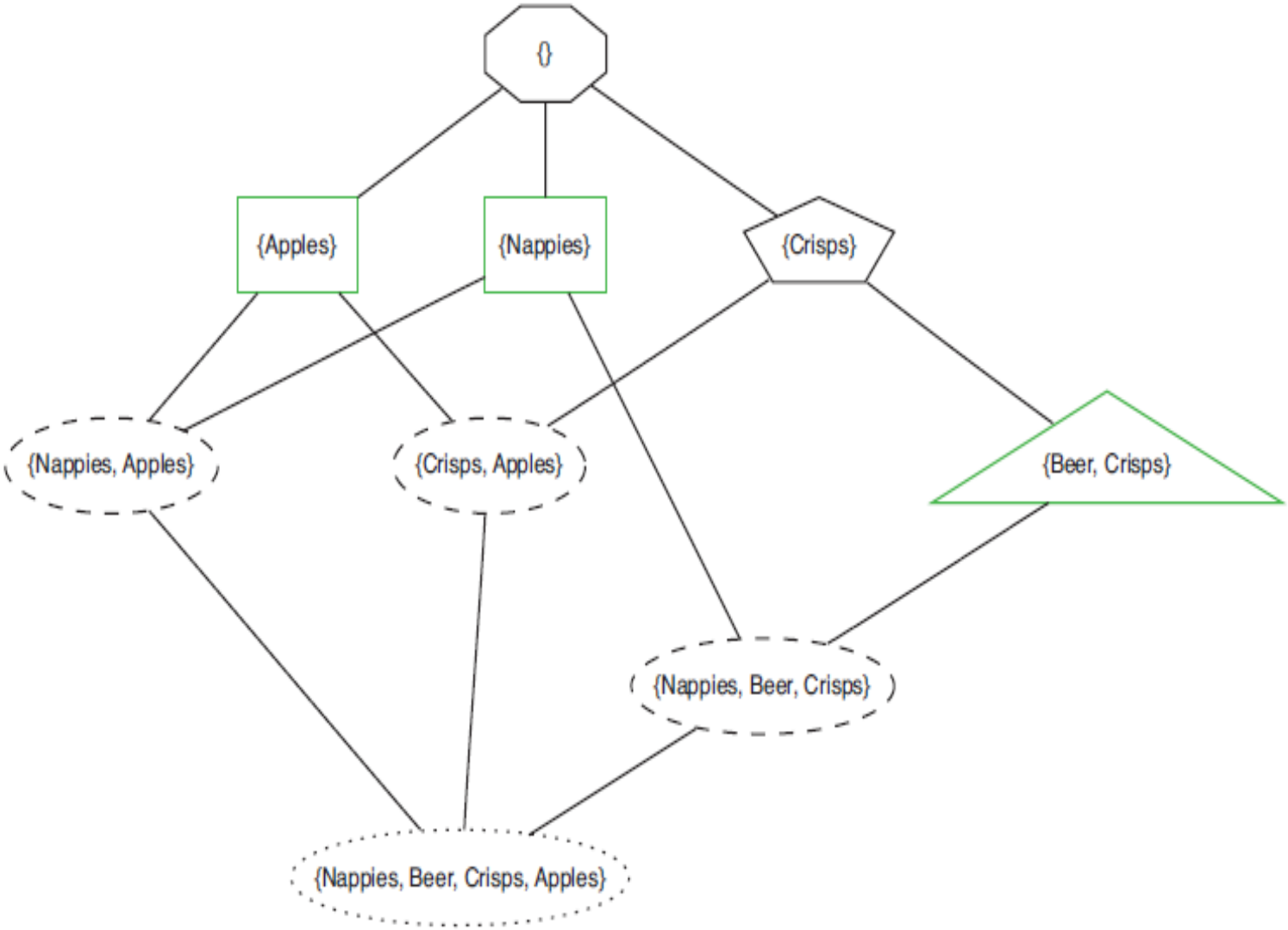

Figure 6.18. Closed item set lattice corresponding to the item sets in Figure 6.17. This lattice has the property that no two adjacent item sets have the same coverage.

So what is the point of these frequent item sets? The answer is that we will use them to build *association rules*, which are rules of the form ·**if** *B* **then** *H*· where both body *B* and head *H* are item sets that frequently appear in transactions together.

Pick any edge in Figure 6.17, say the edge between {beer} and {nappies,beer}. We know that the support of the former is 3 and of the latter, 2: that is, three transactions involve beer and two of those involve nappies as well. We say that the *confidence* of the association rule ·**if** beer **then** nappies· is 2/3. Likewise, the edge between {nappies} and {nappies,beer} demonstrates that the confidence of the rule ·**if** nappies **then** beer· is 2/4.

There are also rules with confidence 1, such as ·**if** beer **then** crisps·; and rules with empty bodies, such as ·**if** true **then** crisps·, which has confidence 5/8

But we only want to construct association rules that involve frequent items. The rule ·**if** beer ∧ apples **then** crisps· has confidence 1, but there is only one transaction involving all three and so this rule is not strongly supported by the data.

So we first use Algorithm 6.6 to mine for frequent item sets; we then select bodies *B* and heads *H* from the frequent sets *m*, discarding rules whose confidence is below a given confidence threshold. Algorithm 6.7 gives the basic algorithm. Notice that we are free to discard some of the items in the maximal frequent sets

Algorithm 6.7: AssociationRules(D, f_0, c_0) – find all association rules exceeding given support and confidence thresholds.

Input : data $D \subseteq \mathscr{X}$; support threshold f_0; confidence threshold c_0.
Output : set of association rules R.

```
1 R ← ∅;
2 M ← FrequentItems(D, f0) ;                    // FrequentItems: see Algorithm 6.6
3 for each m ∈ M do
4 |   for each H ⊆ m and B ⊆ m such that H ∩ B = ∅ do
5 |   |   if Supp(B ∪ H)/Supp(B) ≥ c0 then R ← R ∪ {·if B then H·}
6 |   end
7 end
8 return R
```

A run of the algorithm with support threshold 3 and confidence threshold 0.6 gives the following association rules:

·**if** beer **then** crisps· support 3, confidence 3/3

·**if** crisps **then** beer· support 3, confidence 3/5

Association rule mining often includes a *post-processing* stage in which superfluous rules are filtered out, e.g., special cases which don't have higher confidence than the general case. One quantity that is often used in post-processing is *lift*, defined as

$$\text{Lift}(\cdot\textbf{if } B \textbf{ then } H\cdot) = \frac{n \cdot \text{Supp}(B \cup H)}{\text{Supp}(B) \cdot \text{Supp}(H)}$$

We can also apply frequent item set analysis to our dolphin data set, if we treat each literal Feature = Value as an item, keeping in mind that different values of the same feature are mutually exclusive.

The item set lattice is(Figure 6.19).

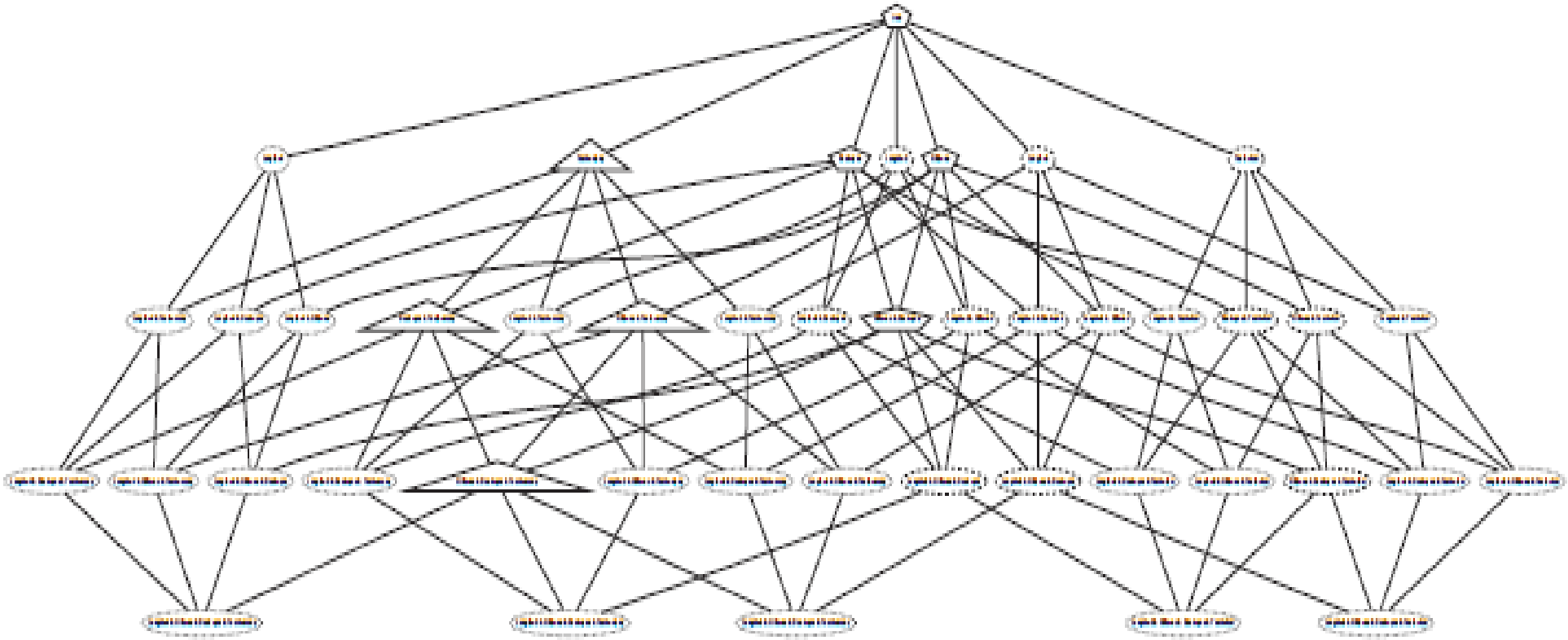

Figure 6.19. The item set lattice corresponding to the positive examples of the dolphin example in Example 4.4 on p.115. Each 'item' is a literal Feature = Value; each feature can occur at most once in an item set. The resulting structure is exactly the same as what was called the hypothesis space in Chapter 4.

The reduction to closed concepts/item sets is shown in Figure 6.20. We can see that, for instance, the rule

if Gills = no ∧ Beak = yes **then** Teeth = many

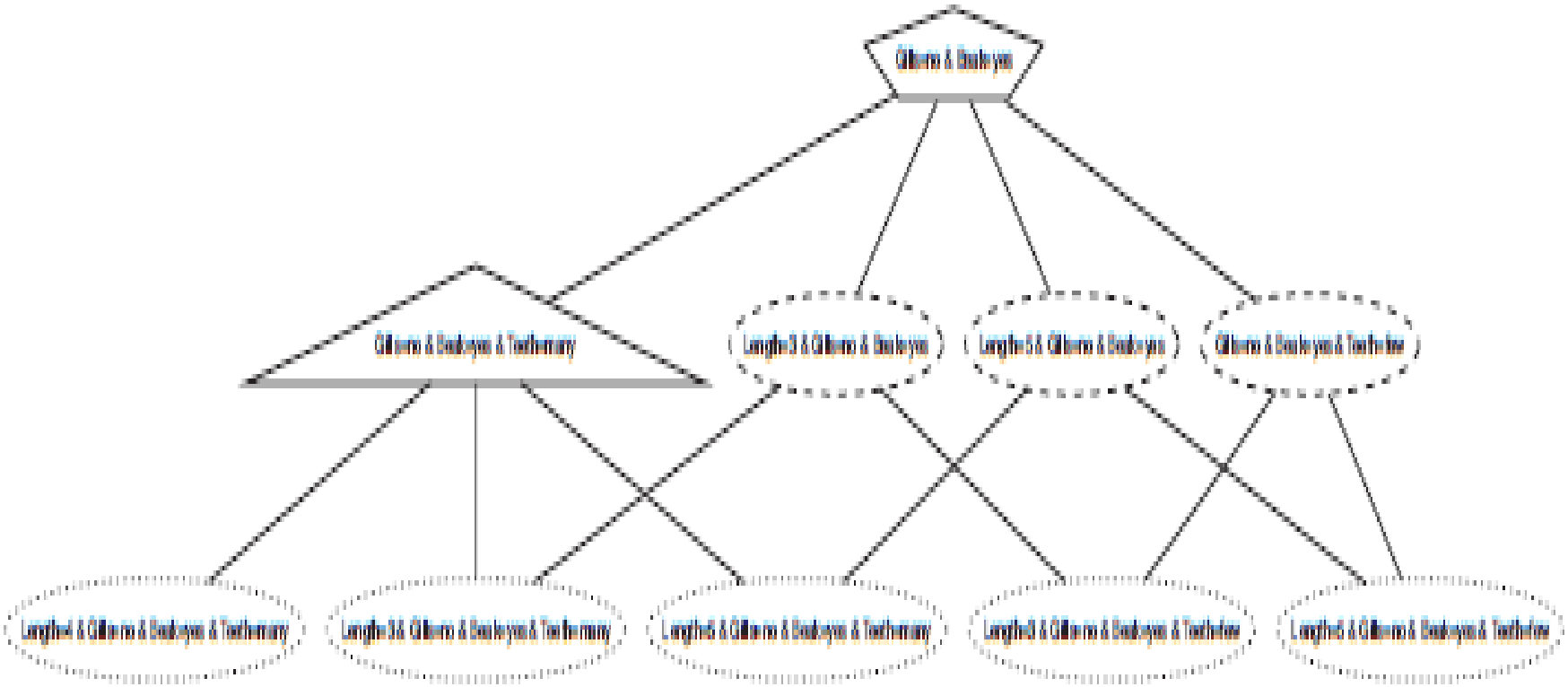

Figure 6.20. Closed item set lattice corresponding to the item sets in Figure 6.19.