頻出パターンマイニング

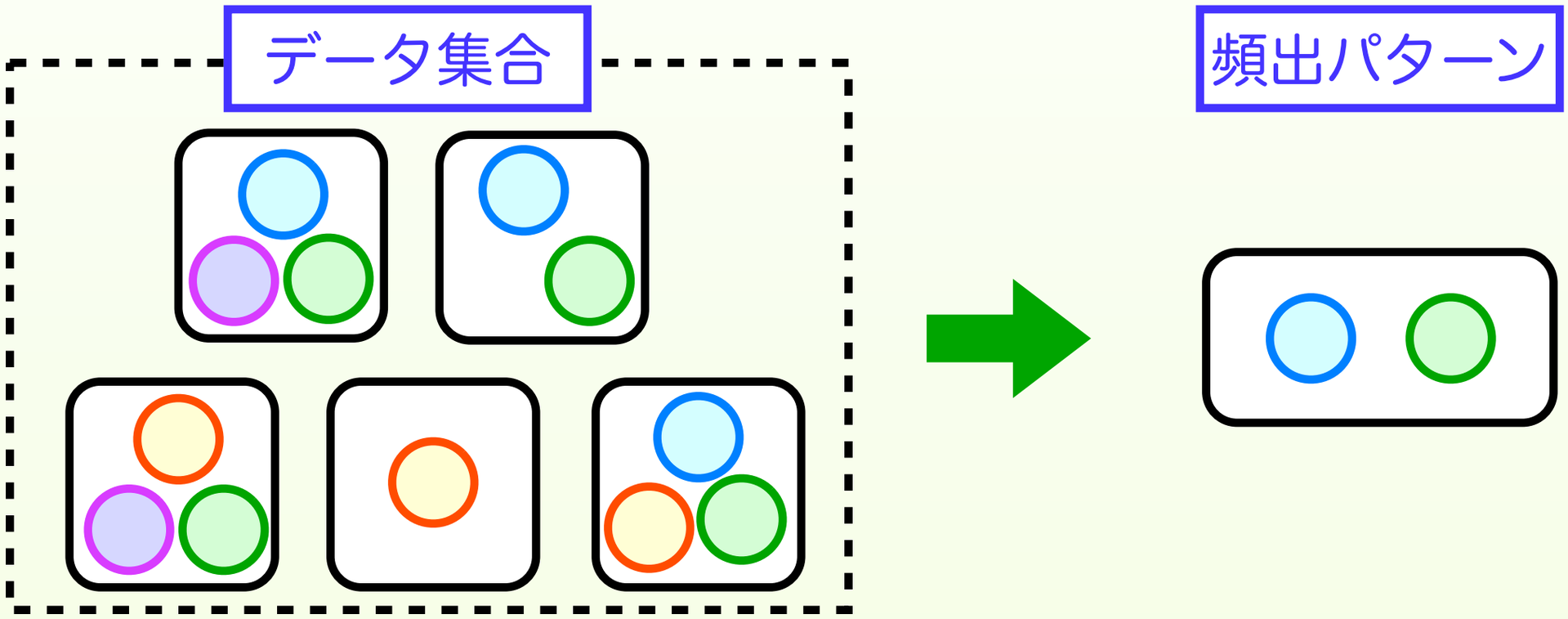
Frequent Pattern Mining

神嶌 敏弘

http://www.kamishima.net/

頻出パターンマイニング

頻出パターンマイニング ある制約を満たすパターンで，
データベース中に高頻度に存在するものを全て列挙する



データ集合の要素 アイテム集合，系列データ，時系列，木，グラフ…

相関ルール

Association Rule

相関ルール

相関ルール Association Rule

$X \Rightarrow Y$

X：前提部 (antecedent)
Y：結論部 (consequent)

X と Y は互いに同じものを含まないアイテムの集合
(アイテム：商品などの「もの」)

X という条件が満たされる場合には，同時に Y という条件が満たされる場合も頻繁に生じる

例：{ 牛乳, パン } ⇒ { 卵 }

牛乳とパン（Xに相当）を同時に買う人は，高い頻度で卵 (Yに相当) を買う

相関ルール

条件Xがトランザクションiを満たす

⇕

Xがトランザクションiの部分集合

例： 条件 X = {牛乳, ハンバーガー}

T_1 = {牛乳, サンドウィッチ, ハンバーガー}

T_2 = {パスタセット, 牛乳}

条件 X ⊆ 取引1 → トランザクション1を満たす

条件 X ⊈ 取引2 → トランザクション2を満たさない

バスケットデータ分析

XとYが共に頻繁に満たされる相関ルールを抽出

相関ルール

利用例 X と Y を組み合わせてセット商品作る

{ 緑茶, ツナおにぎり } ⇒ { タラコおにぎり }
{ 緑茶, ツナおにぎり } ⇒ { コンブおにぎり }

このような相関ルールが共に見つかったとする

緑茶, ツナ, タラコ, コンブおにぎり

このようなセットメニューを作り，単体で買うより少し価格を下げておくと，顧客あたりの購入単価の向上に役立つだろう

相関ルール

利用例

- X と Y を近くに並べて同時購入を促す

  {インスタントラーメン} ⇒ {チャーシュー}

インスタントラーメン売り場にチャーシューを並べると同時購入が増えるだろう

- Xの販売数が増加する場合にYの在庫を増やしておく

  {牛乳} ⇒ {パン}

牛乳の特売をすると，牛乳の販売数が増えると，パンの販売数も増えると予測される
よって，パンの仕入れ数を増やしておく

相関ルール

バスケット データ Basket Data

一度のトランザクション(ひとまとめの取引)ごとに，同時に購入された商品の一覧をまとめたデータ

例： T_1 = {牛乳, サンドウィッチ, ハンバーガー}
T_2 = {焼肉弁当, ポテトサラダ}
⋮
T_N = {パスタセット, 牛乳}

トランザクションT_1は，今日の最初の客との取引.
その客の買い物かご(バスケット)には牛乳，サンドウィッチ，ハンバーガーが入っていた

相関ルール

支持度 support(X)

全トランザクション数に対する条件Xを
満たすトランザクションの割合

例：条件 X＝{a, b} の場合

○ T1 = {a, b, c}	T2 = {a, d}
T3 = {b, d, e}	○ T4 = {a, b, e}
○ T5 = {a, b, c}	T6 = {d, e}

全トランザクション数 = 6
条件を満たすトランザクション数 (○印) = 3

$$\text{support}(X) = \frac{\text{条件Xを満たすトランザクション数}}{\text{全トランザクション数}} = \frac{3}{6} = 0.5$$

相関ルール

確信度 confidence(X,Y)

条件Xを満たすトランザクション数に対する，条件XとYの両方を満たすトランザクション数の割合

$$\text{confidence}(X,Y) = \frac{\text{条件XとYを共に満たすトランザクション数}}{\text{条件Xを満たすトランザクション数}}$$

条件XとYを共に満たす ⇔ XとYが共にトランザクションの部分集合

$$X \cup Y \subseteq T_i$$

↓

$$\text{confidence}(X,Y) = \frac{\text{support}(X \cup Y)}{\text{support}(X)}$$

Apriori

Apriori

Aprioriアルゴリズム

- 大規模なバスケットデータから，幅優先型の探索で相関ルールを列挙する

R.Agrawal and R.Srikant "Fast Algorithms for Mining Association Rules", VLDB 1994

入力 バスケットデータのデータベース
最小支持度 minsup
最小確信度 minconf

出力 バスケットデータのデータベースから次の条件を満たす相関ルール X⇒Y を全て見つけ出す

support(X∪Y) ≧ minsup　　confidence(X,Y) ≧ minconf

Apriori

直接的な解法: 作ることが可能な相関ルールを，全て作って，その支持度と確信度が条件を満たすかを検査

しかし!⬇

- アイテムが10種類の場合でも，それらを組み合わせて作ることのできる相関ルールは 57,002 種と多い
- アイテム数が増えると，さらに膨大になる

⬇

無理!

⬇

支持度と確信度の特徴を使って効率よく探索

Apriori

Step 1. 頻出アイテム集合の探索

- 相関ルール X⇒Y の支持度と確信度の計算には

support(X∪Y) と support(X) の値が必要

- 条件XとYを共に満たすトランザクションは必ず条件Xを満たすので

support(X∪Y) ≦ support(X)

↓

support(X) ≧ minsup を満たすアイテム集合だけで作れる相関ルールだけを考えればよい

このアイテム集合 X を頻出アイテム集合 (frequent itemset)

Step 2. 相関ルールの探索

頻出アイテム集合から相関ルールを生成

Apriori：頻出アイテム集合の抽出

目的

与えられたバスケットデータについて
support(X) ≧ minsup
を満たす全てのアイテム集合 X を抽出

表記

- これら頻出アイテム集合全体の集合を F で表す
- F のうち，ちょうど k個 のアイテムだけを含むアイテム集合で構成されるもの k-頻出アイテム集合 F_k

例: support({a,b})≧minsup とする
{a, b} は2個のアイテムを含むので 2-頻出アイテム集合 F_2 の要素

Apriori：頻出アイテム集合の抽出

効率よく頻出アイテム集合を見つける

F_1～F_k の頻出アイテム集合が分かっているとき
F_{k+1} を効率良く見つける

- X_k はk個のアイテムを含むアイテム集合
- X_{k+1} は，X_k に一つアイテムを加えたアイテム集合

X_k は X_{k+1} の部分集合
$X_{k+1} \supset X_k$

→ $support(X_k) \geqq support(X_{k+1})$

↓

$support(X_k) < minsup$ → $support(X_{k+1}) < minsup$

X_{k+1}からアイテムを一つ除いてできるアイテム集合が
F_k の要素でないなら，X_{k+1} は頻出ではない

※補足：代数的には上半束のdownward closure propertyという

Apriori：頻出アイテム集合の抽出

X_{k+1} からアイテムを一つ除いてできるアイテム集合が F_k の要素でないなら，X_{k+1} は頻出ではない

↓

頻出アイテム集合になりうるアイテム集合

k+1-候補集合　C_{k+1}：X_{k+1}から一つアイテムを除いた集合が全てF_kに含まれる

例：　F_k＝{a,b} {b,c} {a,c} {a,d} の場合

- {a,b,c} は一つ要素を除いてできる {a,b} {b,c} {a,c} が全てF_kの要素なので，C_{k+1}に含める
- {a,b,d} は {b,d}　がF_kの要素ではないので，C_{k+1}に含めない

C_{k+1} 中のアイテム集合だけについて頻出かを検証

Apriori：頻出アイテム集合の抽出

- k = 1
- データベースを数え上げて F_1 を生成
  - ループ先頭
  - F_k から C_{k+1} を生成
    - C_{k+1} 中の集合が実際に頻出かどうかを，データベースを数え上げて F_{k+1} を生成
    - F_{k+1} が空ならばループを終了
    - k = k + 1; ループ先頭に戻る
- 出力：$F_1, F_2 \cdots F_{k+1}$

Apriori：相関ルールの抽出

目的

前のステップで求めた頻出アイテム集合の集合 F から，次の条件を満たす相関ルールを全て抽出

相関ルール：$X \Rightarrow Y$

$X \cup Y \in F$

$$\text{confidence}(X,Y) = \frac{\text{support}(X \cup Y)}{\text{support}(X)} \geqq \text{minconf}$$

XuY がFの要素なら，XやYもFの要素であることに注意

Apriori：相関ルールの抽出

$X \cup Y = X' \cup Y'$ かつ $X \supset X'$ である二つの相関ルール
$X \Rightarrow Y$ と $X' \Rightarrow Y'$ の確信度はそれぞれ

$$\text{confidence}(X,Y) = \frac{\text{support}(X \cup Y)}{\text{support}(X)} \qquad \text{confidence}(X',Y') = \frac{\text{support}(X' \cup Y')}{\text{support}(X')}$$

$X \cup Y = X' \cup Y' \rightarrow \text{support}(X \cup Y) = \text{support}(X' \cup Y')$

$X \supset X' \rightarrow \text{support}(X) \leqq \text{support}(X')$

$\text{confidence}(X,Y) \geqq \text{confidence}(X',Y')$

例：$\{a,b\} \Rightarrow \{c\}$ と $\{a\} \Rightarrow \{b,c\}$ なら
$\text{confidence}(\{a,b\},\{c\}) \geqq \text{confidence}(\{a\},\{b,c\})$

この関係を使って効率よく相関ルールを見つける

Apriori：相関ルールの抽出

要素数2個以上の F 中のアイテム集合 G から
X∪Y=G を満たす相関ルール X⇒Y を全て抽出

Y がk個のアイテムを含む相関ルールが全て抽出済みのときに
Y'がk+1個のアイテムを含む相関ルールを見つける

↓

confidence(X',Y') ≧ minconf であるためには
Y' から X' にどのアイテムを1個だけ戻して作れる全ての相関ルール
X⇒Y で confidence(X,Y) ≧ minconf が成立する必要

例：{a,b,c}⇒{d} と {a,b,d}⇒{c} について，どちらか一方の確信度が minconf 未満なら，相関ルール {a,b}⇒{c,d} の確信度 confidence({a,b},{b,c})≧minconf とはなりえない

Apriori：相関ルールの抽出

- F_2 中の頻出アイテム集合 {A,B} については，{A}⇒{B} と {B}⇒{A} の相関ルールの確信度を検査
- k≧3 の F_k 中の各頻出アイテム集合について
  - j=1
  - {k-1個}⇒{1個} 形式の相関ルールの確信度を検査
  - ループ開始
  - {k-{j+1}個}⇒{j+1個} 形式の相関ルールの候補を，{k-j個}⇒{j個} 形式のルールから求める
  - 実際に確信度が minconf 以上かどうかを検証
  - 新たなルールが見つからない場合は終了
  - j = j +1 としてループを続行

FP-growth

Frequent Pattern-growth

FP-growth

FP-growthアルゴリズム

- 深さ優先型の探索で頻出アイテム集合を列挙する
- trie に似たFP-tree方法でデータを格納する

J.Han, J.Pei, and Y.Yin “Mining Frequent Patterns without Candidate Generation” SIGMOD 2000

入力 バスケットデータのDB
最小支持度 minsup

出力 次の条件を満たす全て頻出アイテム集合列挙

support(X) ≧ minsup

Apriorアルゴリズムとの比較

- Apriori は余分な候補集合を多く生成する ➡候補集合の抑制
- FP-tree をメモリ上に保持するので，メモリの使用量は多い (頻出集合の数に比例)

FP-tree

FP-tree

基本方針

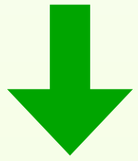
- あるアイテム a を一つだけ含む集合 {a} が頻出でないときは，a を含むどのアイテム集合も頻出にはならない
  - a を含むアイテム集合は候補から除外
- 頻出かどうかは，アイテム集合の頻度のみで決まる
  - 各アイテムがどのトランザクションに含まれていたかは忘れて，頻度のみを保持する

FP-tree

- Aprioriと同様に，DBを一度走査して，アイテムを一つだけ含むアイテム集合 (すなわちApriori F_1)
- アイテムを頻度の降順でソートして，頻度と共に格納

バスケットDBの例，minsup = 2/9 として頻出集合を求める

T_1	a, b, e	T_4	a, b, d	T_7	a, c
T_2	b, d, f	T_5	a, c, g	T_8	a, b, c, e
T_3	b, c	T_6	b, c	T_9	a, b, c



アイテムの頻度

L = 〈 {b:7} {a:6} {c:6} {d:2} {e:2} {f:1} {g:1} 〉

頻出ではないので除外

FP-tree

追加するトランザクション

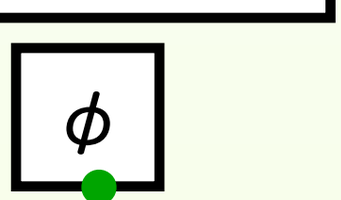
$T_1 = \{a,b,e\}$

Lの順序＝アイテム頻度でソートする

$T_1 = \{b,a,e\}$

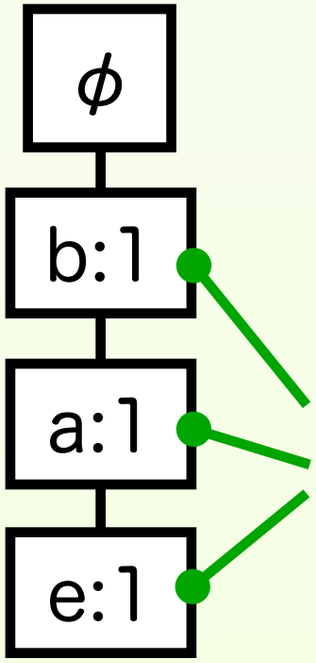
L	
アイテム	頻度
b	7
a	6
c	6
d	2
e	2

初期状態のFP-tree



ルートのみの空の状態

1個加えた状態

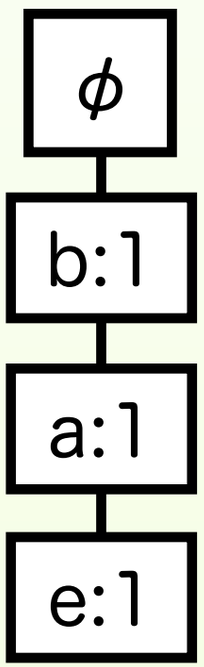


どのアイテムもまだ一回ずつ

FP-tree

L	
アイテム	頻度
b	7
a	6
c	6
d	2
e	2

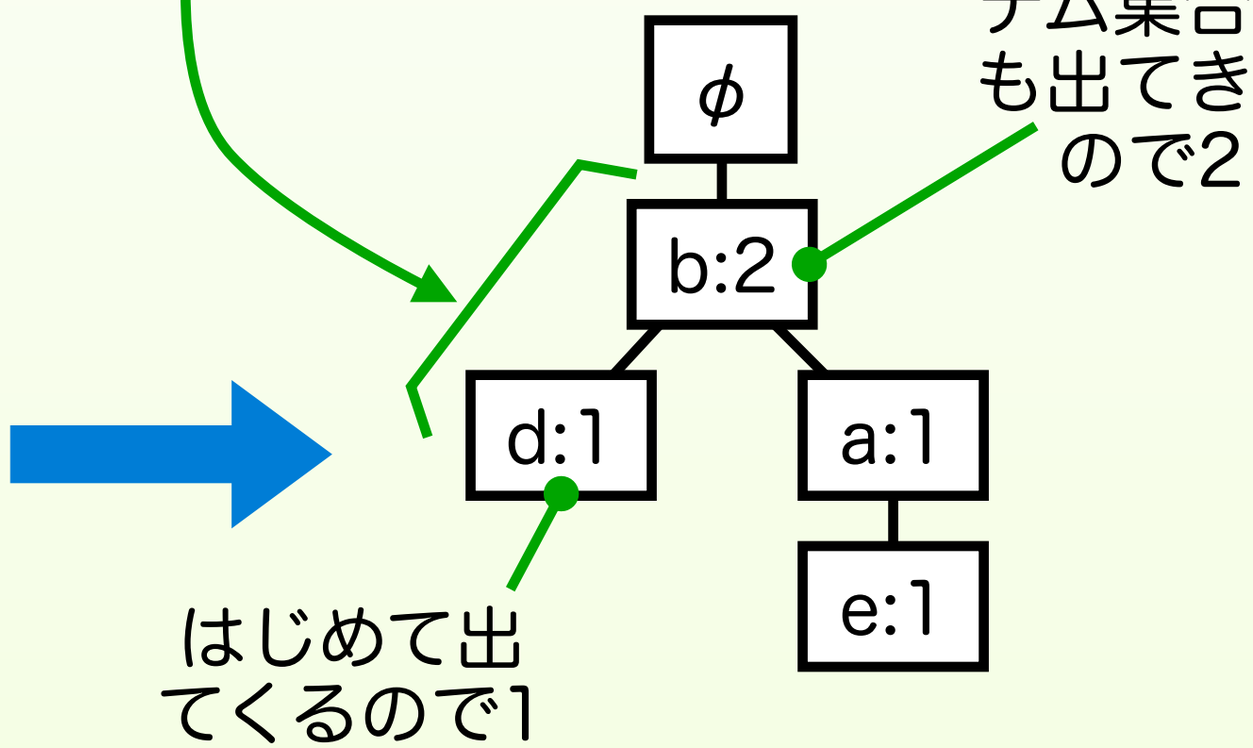
1個加えた状態



ソート済みのトランザクション

T_2 = {b,d}

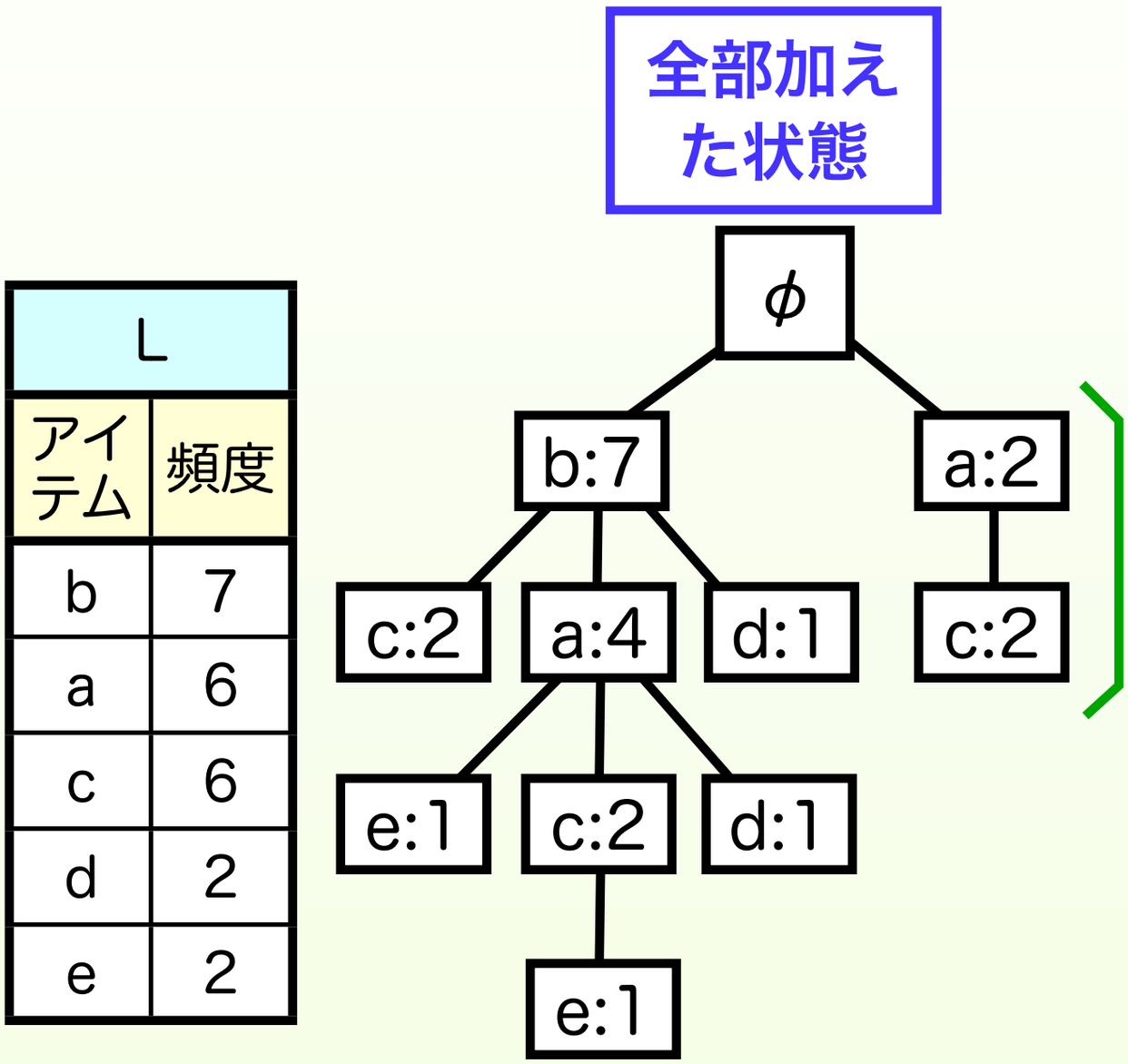
2個加えた状態



1個めのアイテム集合にも出てきたので2

はじめて出てくるので1

FP-tree

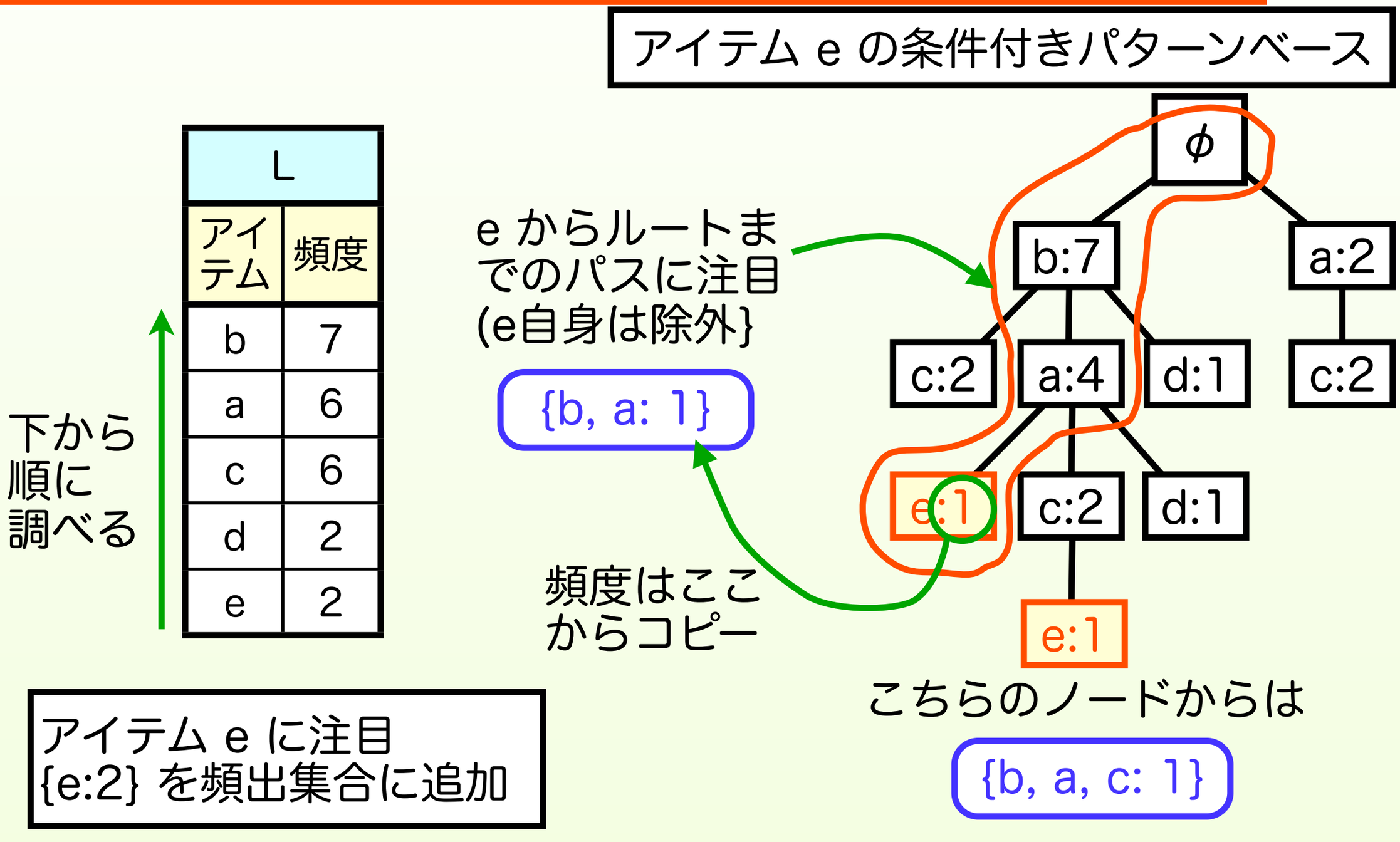


アイテム b を含まない
$T_5 = T_7 = \{a,c\}$
はルートノードから別の枝になる

node-link：同じアイテムのノードを連続して参照するためののリンク

FP-growth

条件付きパターンベース (Conditional Pattern Base; CPB)



アイテム e に注目
{e:2} を頻出集合に追加

FP-growth

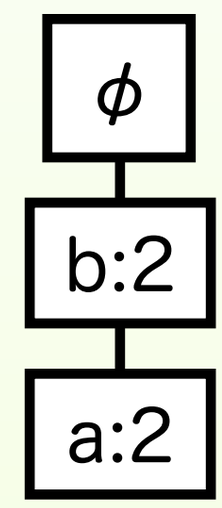
CPBから条件付きFP-treeを生成：treeに枝分かれがない場合

アイテム e のCPB：{b, a: 1} {b, a, c: 1}

アイテム e の
条件付きFP-tree

L	
アイテム	頻度
b	2
a	2
c	1

頻出ではないので c は除外



条件付き FP-tree に枝別れがない

- tree中の全てのアイテムの組み合わせを選び出す：{a} {b} {b, a}
- 頻度は一番小さなものに設定
  {b: 2} {a: 2} {b, a: 2}
- これはアイテム e の条件付きFP-treeなので，e を加えたものを頻出集合に追加
  {b, e: 2} {a, e: 2} {b, a, e: 2}

補足：例えば {ϕ}-{b:3}-{a:2}-{c:1} というFP-treeの場合
{b: 3} {a: 2} {c: 1} {b, a: 2} {b, c: 1} {b, a, c: 1} になる

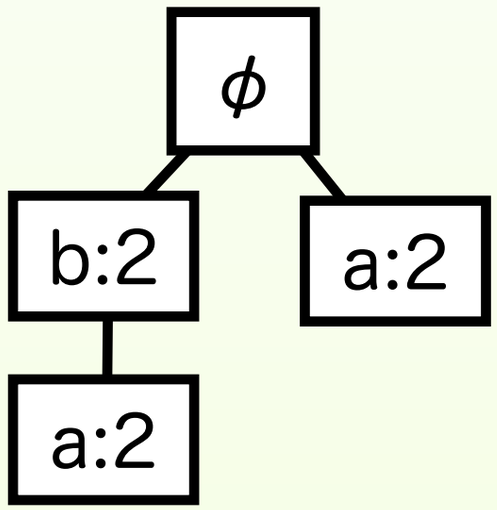
FP-growth

CPBから条件付きFP-treeを生成：treeに枝分かれがある場合

アイテム c のCPB：{b, a: 2} {b: 2} {a: 2}

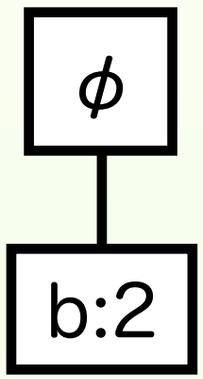
アイテム c の
条件付きFP-tree

L	
アイテム	頻度
b	4
a	4



**条件付き FP-tree に
枝別れがある
再帰的にFP-growthを適用**

さらに a で条件
付けした FP-tree

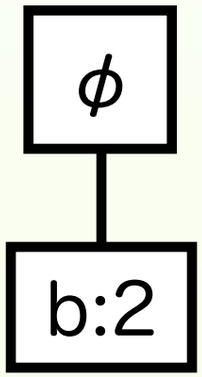


さらに b で条件
付けした FP-tree

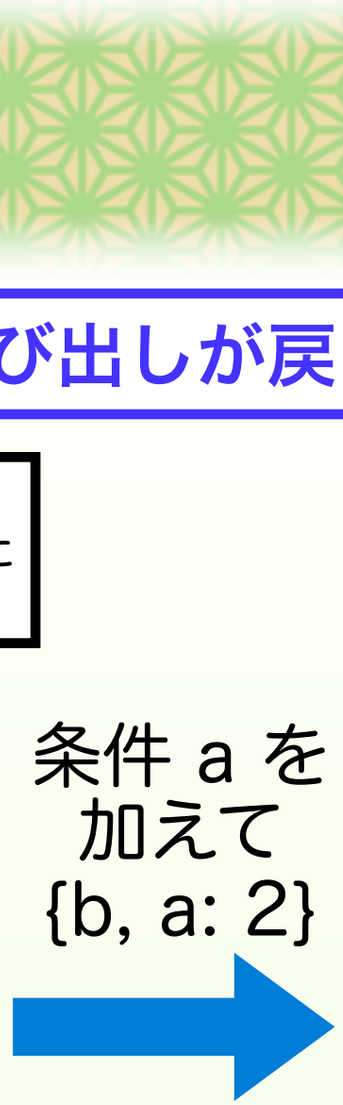
FP-growth

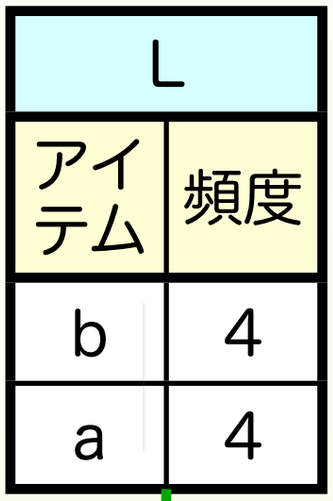
再帰呼び出しが戻る様子

さらに a で
条件付けした
FP-tree



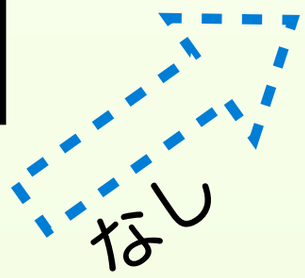
条件 a を
加えて
{b, a: 2}



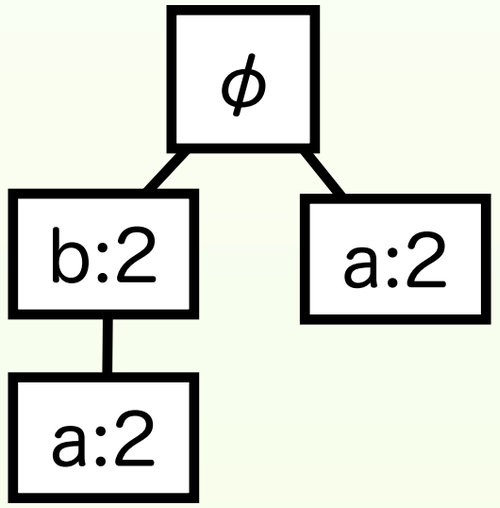


リストからは
{b:4} {a:4}

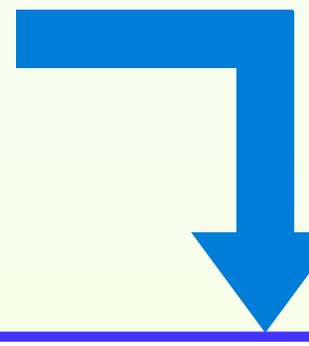
さらに b で
条件付けした
FP-tree



アイテム c の
条件付きFP-tree



再帰を1段戻ると
{b:4} {a:4} {b,a:2}
が得られた



条件付けしたアイテム c を加えた
{b, c: 4} {a, c: 4}
{b, a, c: 2}
を頻出集合に追加

FP-growth

リスト L の下のアイテム i から順に処理をする

1.アイテム i のみを含む集合 {i} は頻出集合に追加
2.アイテム i の条件付きパターンベース (CPB) を生成
3.この CPB に対する 条件付きFP-tree を生成

a)もし枝なしのFP-treeだったら，それに含まれるアイテムの組み合わせにアイテム i を加えたものを頻出に追加

b)もし枝ありのFP-treeだったら，そのFP-treeに再帰的にFP-growthを適用し，返されたアイテム集合にアイテム i を加えたものを頻出集合に追加

系列データ

Sequence Data

系列データ

バスケットデータ　同時に購入したアイテムの集合

T_1 = {牛乳, サンドウィッチ, ハンバーガー}
T_2 = {パスタセット, 牛乳}

系列データ　ある人が購入したアイテム集合の系列

1月10日	1月12日	・・・	1月30日
牛乳, サンドウィッチ, ハンバーガー	幕の内弁当, 緑茶		牛乳, サンドウィッチ，サラダ

表記

- 同時に購入したアイテムを括弧でまとめる
- ただし，同時に購入したものが1個だけなら括弧は省略

例：〈c a (bd) a〉
　　初回は c，2回目 a，3回目 bとd 4回目 a を購入

系列データ

バスケットデータの系列の包含関係

系列の順序を保ったままアイテム集合の間に包含関係がある場合

S2はS1に含まれる

S_1 ＝〈 b　(a c)　e　(d f)　〉

S_2 ＝〈　　(a c)　　d　〉

- 複数のアイテム集合をまたいで包含関係は成立しない
  例：〈(a b)〉は〈a b〉には含まれない
- 順序関係が保たれていれば，間に幾つアイテム集合があってもよい
  例：〈a b c d e f g h〉に〈a h〉は含まれる

系列パターンマイニング

系列パターンマイニング

minconf以上の割合でDB中のデータに含まれるような系列データ**（＝頻出系列パターン）**を全て列挙する

例：S_1＝〈a b c〉S_2＝〈b (a c)〉S_3＝〈a c〉というDBがあり，
minconf=2/3 の場合
〈a c〉は S_1 と S_3 に含まれるため頻出系列パターンだが，
〈a b〉はS_1にのみ含まれるだけなので頻出系列パターンではない

利用例

- 〈 (焼肉タレ 牛肉) 豆腐〉という系列パターンが見つかったとする
  焼肉タレと牛肉のバーゲンの後では，豆腐の仕入れを増やしておく
- 〈デジタル一眼レフ 望遠レンズ〉という系列パターンがあるとき
  一眼レフを購入した顧客に，次回には望遠レンズを薦める

PrefixSpan

PREFIX-projected Sequential PAtterN mining

PrefixSpan

PrefixSpanアルゴリズム

- 深さ優先型の探索で頻出系列パターンを列挙する
- 発見済みのパターンにマッチした残りをpostfixと呼び，そのpostfixだけを保持する射影データベースを使うことで効率化を図る

J.Pei, J.Han, B. Mortazavi-Asl, H.Pinto, Q.Chen, U.Dayal, M.-C.Hsu "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Groush" ICDE 2001

入力 系列データのDB
最小支持度 minsup

出力 次の条件を満たす全て頻出系列パターン P を列挙
support(P) ≧ minsup

※ 参考：系列パターンマイニングの幅優先型探索アルゴリズムとしてAprioriAllや，その改良版 GSP（Generalized Sequential Pattern）がある

PrefixSpan

アイテム {a～g} に対する系列DBの例，minsup = 2/4 とする

S_1	〈 a (abc) (ac) d (cf) 〉	S_3	〈 (ef) (ab) (df) c b 〉
S_2	〈 (ad) c (bc) (ae) 〉	S_4	〈 e g (af) c b c 〉

※ アイテムには辞書順を決めておく．ここでは a から g の順とする

postfix 系列 S の，系列 T に対する postfix は，S 中で T が最初に含まれていた部分までを除いた，S の残りの部分

例：S＝S_1＝〈a (abc) (ac) d (cf)〉の場合

- T=〈a〉なら〈 **a** (abc) (ac) d (cf)〉にマッチして，postfixは
  〈 (abc) (ac) d (cf)〉
- T=〈a a〉なら〈 **a** (**a**bc) (ac) d (cf)〉にマッチして，postfixは
  〈 (_bc) (ac) d (cf)〉 （“_”は，Tの最後の要素にマッチした残り）
- T=〈a b〉なら〈 **a** (a**b**c) (ac) d (cf)〉にマッチして，postfixは
  〈 (_c) (ac) d (cf)〉 （集合abcでbより辞書順で前のaは残さない）

PrefixSpan

射影DB (projected database)

DB中の全系列について，　与えられた系列の postfix を求めたもの

S_1	〈 a (abc) (ac) d (cf) 〉	S_3	〈 (ef) (ab) (df) c b 〉
S_2	〈 (ad) c (bc) (ae) 〉	S_4	〈 e g (af) c b c 〉

例：

- 〈a〉-射影DBは
  〈(abc) (ac) d (cf)〉　〈(_d) c (bc) (ae)〉　〈(_b) (df) c b〉　〈(_f) c b c〉
- 〈e〉-射影DBは
  〈(_f) (ab) (df) c b〉　〈(af) c b c〉
  ※ S_2に対して〈e〉をマッチさせると，postfix は空になるので含めない
- 〈a b〉-射影DBは
  〈(_c) (ac) d (cf)〉　〈(_c) (ae)〉　〈c〉
  〈a〉-射影DBより必ず小さくなる➡少ないメモリで保持できる

PrefixSpan

PrefixSpanアルゴリズム

S_1	〈 a (abc) (ac) d (cf) 〉	S_3	〈 (ef) (ab) (df) c b 〉
S_2	〈 (ad) c (bc) (ae) 〉	S_4	〈 e g (af) c b c 〉

1 アイテム一つだけの系列の頻度を調べる

〈a〉: 4, 〈b〉: 4, 〈c〉: 4, 〈d〉: 3, 〈e〉: 3, 〈f〉: 3

ただし，頻度が2未満のアイテム g は除外する

2 辞書順で最初の系列〈a〉に注目．〈a〉-射影DBを計算

〈a〉-射影DB〈(abc) (ac) d (cf)〉〈(_d) c (bc) (ae)〉〈(_b) (df) c b〉〈(_f) c b c〉

3 可能な拡張パターン全てが頻出がどうかを調べる

- アイテム集合 a に，辞書順にアイテムを加えて拡張する
  〈(ab)〉〈(ac)〉〈(ad)〉…〈(af)〉（〈(aa)〉は除外する）
- 系列〈a〉に，アイテム集合が一つだけの系列を加えて拡張する
  〈a a〉〈a b〉〈a c〉…〈a f〉（〈a a〉も含まれる）

PrefixSpan

S_1	〈 a (abc) (ac) d (cf) 〉	S_3	〈 (ef) (ab) (df) c b 〉
S_2	〈 (ad) c (bc) (ae) 〉	S_4	〈 e g (af) c b c 〉

〈a〉-射影DB〈(abc) (ac) d (cf)〉〈(_d) c (bc) (ae)〉〈(_b) (df) c b〉〈(_f) c b c〉

- 〈(ab)〉について調べるときは，〈a〉-射影DBで，“_” を含むアイテム集合に b が含まれるか，または (ab) を含むアイテム集合があるかを調べればよい.
  ここでは，S_1 と S_3 の〈a〉-射影DBにマッチする.

- 〈a b〉について調べるときは，“_” を含まないアイテム集合に，b を含むものがあるかを調べればよい.
  ここでは，S_1，S_2，S_3，S_4 の〈a〉-射影DBにマッチする.

PrefixSpan

S_1	〈 a (abc) (ac) d (cf) 〉	S_3	〈 (ef) (ab) (df) c b 〉
S_2	〈 (ad) c (bc) (ae) 〉	S_4	〈 e g (af) c b c 〉

〈a〉-射影DB〈(abc) (ac) d (cf)〉〈(_d) c (bc) (ae)〉〈(_b) (df) c b〉〈(_f) c b c〉

最終的に，〈a〉を，一つ拡張してできる系列で頻出になるものは…

〈a a〉: 2，〈a b〉: 4，〈(ab)〉: 2，〈a c〉: 4，〈a d〉: 2，〈a f〉: 4

4 再帰的な実行

- 〈a〉を拡張してできる頻出パターンについて順番に，射影DBを求め，さらに一つづつ拡張していく
- 系列パターン〈a〉について，さらに拡張しても頻出パターンが見つからなかったら，残りの系列パターン〈b〉～〈f〉についても処理

※ 拡張するごとに，頻出パターンにはなりにくくなり，それを格納するのに必要なメモリも減るため効率が良い

PrefixSpan

実行には次のサブルーチンを PrefixSpan(〈〉, l, DB) で呼び出す

サブルーチン：PrefixSpan(α, l,　DB)
α：系列，l：αの長さ，DB データベース
1.DB を走査して，次のいずれかを満たす頻出アイテム b を見つける
　a)b はαの最後のアイテム集合に加えることが可能
　b)〈b〉を，αの末尾に連結可能
2.各頻出アイテム b について，b を使ってαを拡張したパターンα'を生成し，α'を出力
3.各α'について，α'-射影DB を生成し，PrefixSpan(α', l+1，α'-射影DB) を呼び出す

その他の改良

- **bi-level projection**：効率化のため，二つずつ拡張する
- **pseudo-projection**：射影DBには重複部分を複製しないようにしてメモリに格納

その他の頻出パターンマイニング

アイテム集合

頻出飽和アイテム集合 (frequent closed item set)

大規模DBから，頻出アイテムを列挙すると膨大な数の集合が見つかる

↓

人間はチェックできず，分析結果を活用できない

support(X)=support(Y) かつ Y⊂X のとき X の方が役立つ

飽和アイテム集合：このようなXの中で一番大きな集合

バスケットデータ

1,2,5,6,7,9
2,3,4,5
1,2,7,8,9
1,7,9
2,3,5,7,9
2,7,9

minconf = 3/6 の場合

{1} {1,7} {1,9} **{1,7,9}**
{2}
{7} {9} **{7,9}**
{2,7} {2,9} **{2,7,9}**

青字が頻出飽和アイテム集合

※ 系列データなど頻出パターンマイニングでも飽和集合は利用される

系列データ

単一系列データの頻出パターン

Webのログデータは，人ごとに分かれておらず，一つに繋がっている
その中で繰り返し生じるパターンを見つけたい

無理に分割すると…

重複して数えられたり…

パターンが分断されたり…

分割しないと… **頻出**の定義に困る…

- **Winepi**：平滑窓（上記A）で区切り，重複して数えないようにして，全平滑窓数に対する，マッチした窓数がしきい値以上
- **Minepi**：パターンがマッチしている系列上の最小の領域（極小発生；minimal occurence）の全領域に対する比がしきい値以上

H.Mannila, H.Toivonen, and A.I.Verkamo “Discovery of Frequent Episodes in Event Sequences” Data Mining and Knowledge Discovery (1997)

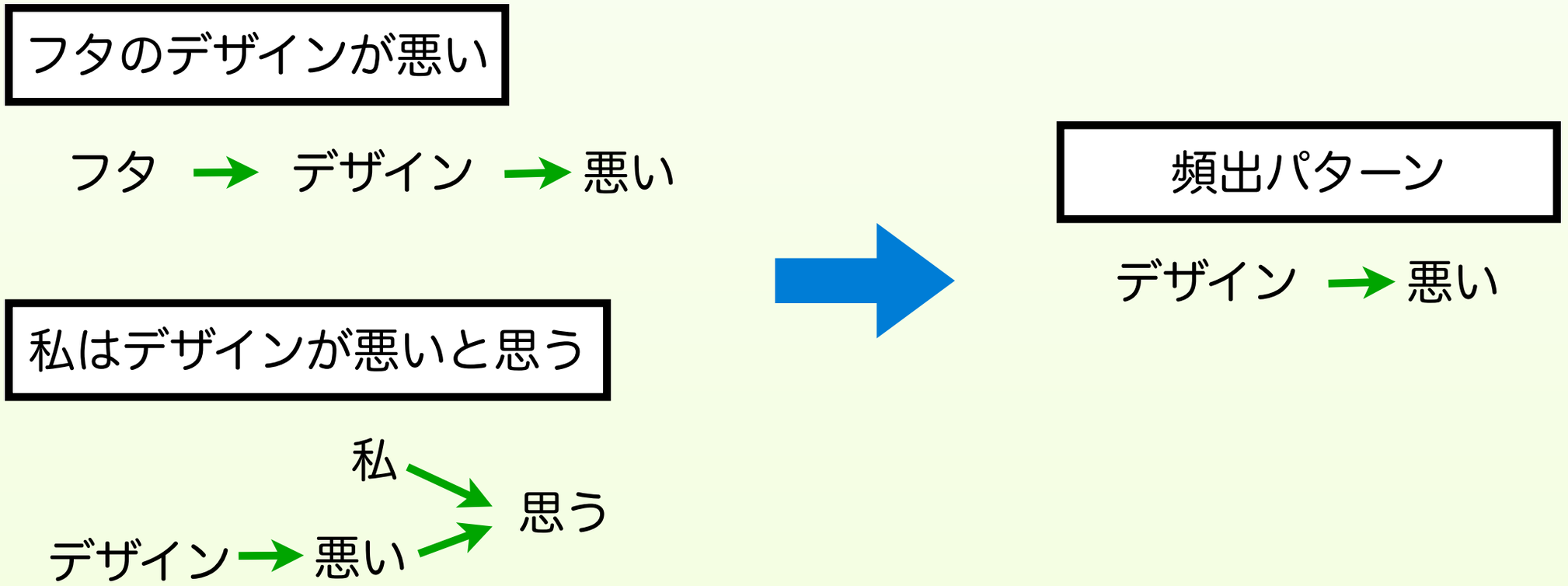
※ この論文で扱う系列はPrefixSpanが扱うデータとは若干異なり，タイムスタンプをもつ

木

- 木で表現されたデータからの頻出パターンマイニング
- 兄弟ノード間に順序がある順序木と，そうでない非順序木がある

利用例

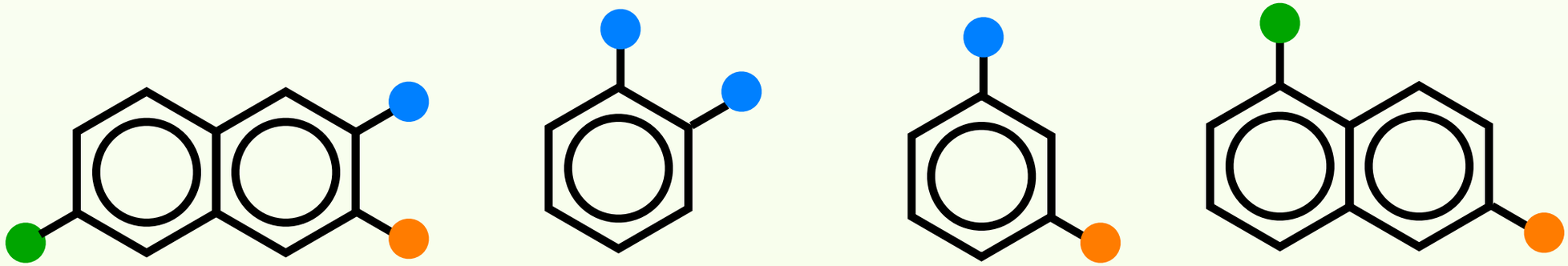
- XMLで表されたデータは木構造をもつ
- 自然言語文を係り受け解析した結果の木を分析する



グラフ

グラフマイニング：グラフで表現されたデータからのマイニング

例：回路，化合物，タンパク構造，生物学的ネットワーク，社会ネットワーク，Web，ワークフロー，XMLデータ



化合物のグラフ：結合が辺，分子や基がノード
このグラフの中から，頻出する部分グラフを抽出
➡ 特定の性質・薬効をもつパターンに相当

- グラフは同一性の判定が難しいので特殊な索引付け技術が必要
- カーネルを使う方法もある

まとめ

- **頻出パターンマイニング**
  ある制約を満たすパターンで，データベース中に高頻度に存在するものを全て列挙する
- **頻出アイテム集合**：DB中で高頻度で含まれるアイテム集合．Apriori と FP-growthアルゴリズムを紹介
- **相関ルール**：X が起きるときには Y も起きやすい．頻出アイテム集合から生成する．
- **系列データ**：アイテム集合の系列．PrefixSpanアルゴリズムを紹介
- **その他の頻出パターンマイニング**：飽和集合，系列データ，木，グラフなど

補足資料

Apriori具体例

例：おにぎり専門店「Oms-B」

アイテム　5種類

A. タラコ　B. ツナ　　C. コンブ　D. サケ　　E. ウメ

トランザクション

- 総トランザクション数 N=5

相関ルールの条件

- 最小支持度　minsup = 0.4
- 最小確信度　minconf = 0.7

例：おにぎり専門店「Oms-B」

	a タラコ	b ツナ	c コンブ	d サケ	e ウメ
T1		●	●	●	
T2	●	●			
T3	●			●	●
T4		●	●	●	
T5		●		●	

各行が一つのトランザクションを表す，購入したアイテムに●印

頻出アイテム集合の抽出

1-頻出アイテム集合

総トランザクション数 N=5
minsup = 0.4

	a タラコ	b ツナ	c コンブ	d サケ	e ウメ
T_1		●	●	●	
T_2	●	●			
T_3	●			●	●
T_4		●	●	●	
T_5		●		●	

アイテムを1種類含む全ての集合を検証

- {a} を満たすトランザクションはT_2とT_3の2個
  - → support({a}) = 2/N = 0.4 ≧ minsup
  - → {a} は頻出
- {e} を満たすトランザクションはT_3の1個
  - → support({e}) = 1/N = 0.2 < minsup
  - → {e} は頻出ではない

以下，同様に考えると F_1={a} {b} {c} {d}

頻出アイテム集合の抽出

2-頻出アイテム集合

総トランザクション数 N=5
minsup = 0.4
F1={a} {b} {c} {d}

	a タラコ	b ツナ	c コンブ	d サケ	e ウメ
T_1		●	●	●	
T_2	●	●			
T_3	●			●	●
T_4		●	●	●	
T_5		●		●	

F_1から候補アイテム集合C_2を生成

- {a,b} は {a} と {b} がF_1の要素なので，C_2 の要素
- {a,e}は{e}がF_1の要素ではないので，C_2の要素ではない

以下，同様に C_2={a,b} {a,c} {a,d} {b,c} {b,d} {c,d}

- {a,b}を満たすトランザクションはT_2の1個
  support({a,b}) = 1/N = 0.2 < minsup ➡ {a,b}は頻出ではない
- {b,c}を満たすトランザクションはT_1とT_4の2個
  support({b,c}) = 2/N = 0.4 ≧ minsup ➡ {b,c}は頻出

以下，同様に F_2={b,c} {b,d} {c,d}

頻出アイテム集合の抽出

3-頻出アイテム集合

総トランザクション数 N=5
minsup = 0.4
F_2= {b,c} {b,d} {c,d}

	a タラコ	b ツナ	c コンブ	d サケ	e ウメ
T_1		●	●	●	
T_2	●	●			
T_3	●			●	●
T_4		●	●	●	
T_5		●		●	

F_2から候補アイテム集合C_3を生成

- {a,b,c} は {a,b}と{a,c}がF_2の要素ではないので，C_3の要素ではない
- {b,c,d} は {b,c} {b,d} {c,d} がF_2の要素なので，C_3の要素

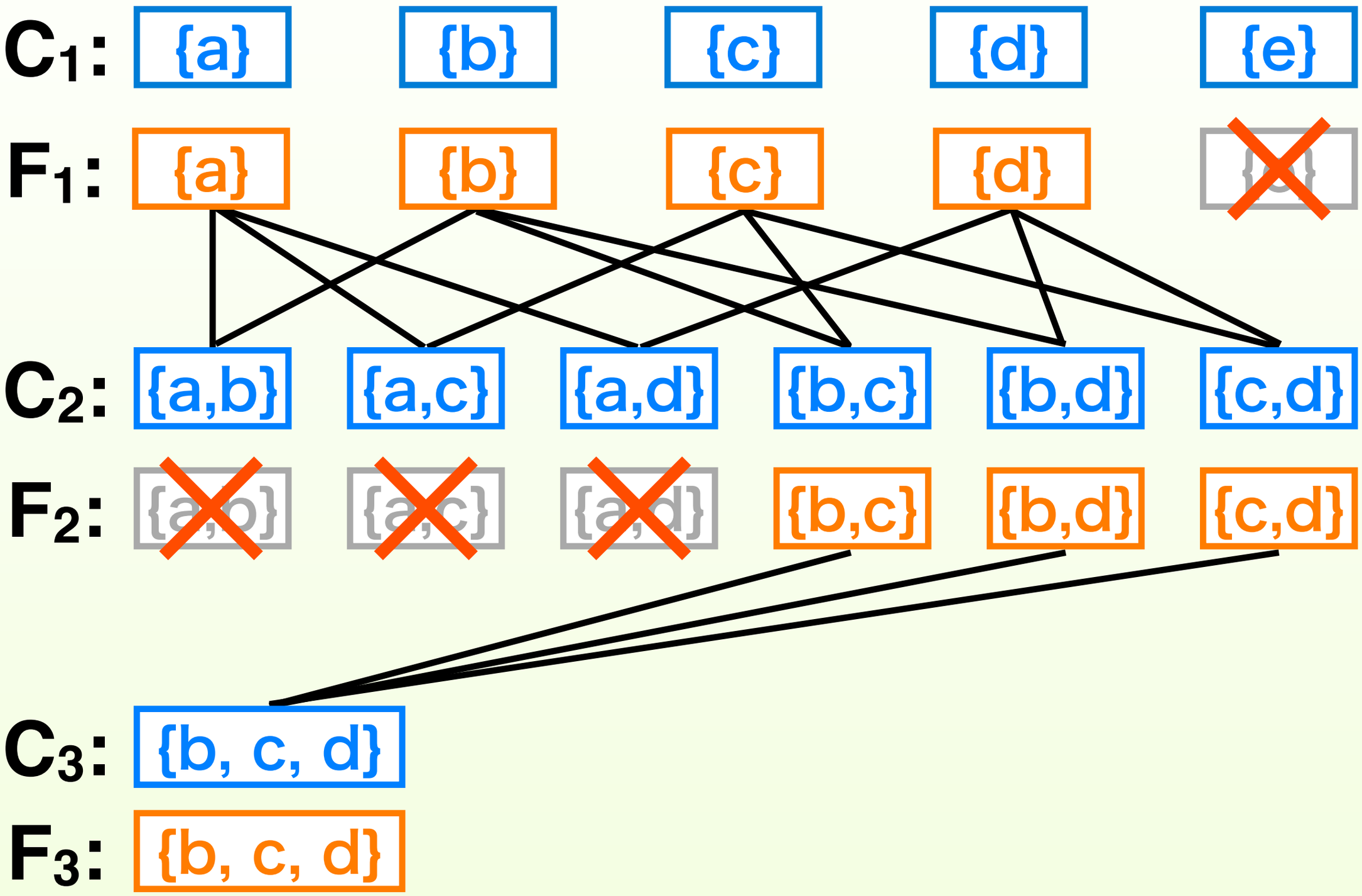
以下，同様に C_3={b,c,d}

- {b,c,d}を満たすトランザクションはT_1とT_4の2個
  support({b,c,d}) = 2/N = 0.4 ≧ minsup ➡ {b,c,d}は頻出

よって F_3={b,c,d}

候補集合 C_4 は空になるのでここで終了

頻出アイテム集合の抽出



相関ルールの抽出

N=5, minsup=0.4, minconf=0.7
F_1={a} {b} {c} {d}
F_2= {b,c} {b,d} {c,d}
F_3={b,c,d}

	a タラコ	b ツナ	c コンブ	d サケ	e ウメ
T_1		●	●	●	
T_2	●	●			
T_3	●			●	●
T_4		●	●	●	
T_5		●		●	

F_2から相関ルールを抽出

{b,c} からは {b}⇒{c} と {c}⇒{b} の二つの相関ルールが作れる
ここで support({b,c})=2/N, support({b})=4/N, support({c})=2/N

- {b}⇒{c} の確信度は
  confidence({b},{c}) = support({b,c})/support({b}) = 2/4 < minconf
  ➡ {b}⇒{c} を廃棄
- {c}⇒{b} の確信度は
  confidence({c},{b}) = support({b,c})/support({c}) = 2/2 ≧ minconf
  ➡ {c}⇒{b} を抽出

F_2から抽出される相関ルールは

{c}⇒{b}, {b}⇒{d}, {d}⇒{b}, {c}⇒{d}

相関ルールの抽出

N=5, minsup=0.4, minconf=0.7
F_1={a} {b} {c} {d}
F_2= {b,c} {b,d} {c,d}
F_3={b,c,d}

	a タラコ	b ツナ	c コンブ	d サケ	e ウメ
T_1		●	●	●	
T_2	●	●			
T_3	●			●	●
T_4		●	●	●	
T_5		●		●	

{b,c,d}から相関ルールを抽出 (Yが1個のアイテムを含む場合)

{b,c,d}, {b,c}, {b,d}, {c,d} の支持度はそれぞれ 2/N, 2/N, 3/N, 2/N

- {b,c}⇒{d} の確信度は
  confidence({b,c},{d}) = support({b,c,d})/support({b,c}) = 2/2
  確信度はminconf以上 ➡ {b,c}⇒{d} を抽出
- {b,d}⇒{c} の確信度は
  confidence({b,d},{c}) = support({b,c,d})/support({b,d}) = 2/3
  確信度はminconf未満 ➡ {b,d}⇒{c} を廃棄
- {c,d}⇒{b} の確信度は
  confidence({c,d},{b}) = support({b,c,d})/support({c,d}) = 2/2
  確信度はminconf以上 ➡ {c,d}⇒{b} を抽出

相関ルールの抽出

{b,c,d}から相関ルールを抽出 (Yが2個のアイテムを含む場合)

ここまでで抽出されているルールは {c,d}⇒{b} と {b,c}⇒{d}

{b,d}⇒{c} の確信度がminconf 未満なので，{b}⇒{c,d} と {d}⇒{b,c} の確信度はminconf以上にはならない

➡ {b}⇒{c,d} と {d}⇒{b,c} を考慮しなくてよい

- {c}⇒{b,d} の確信度は
  confidence({c},{b,d}) = support({b,c,d})/support({c}) = 2/2
  確信度はminconf以上 ➡ {c}⇒{b,d} を抽出

F_3から抽出される相関ルールは {c,d}⇒{b}, {b,c}⇒{d}, {c}⇒{b,d}

まとめると，抽出される相関ルールは

{c}⇒{b}, {b}⇒{d}, {d}⇒{b}, {c}⇒{d}
{c,d}⇒{b}, {b,c}⇒{d}, {c}⇒{b,d}