Computational Learning Theory
Frequent Item Set Mining

Akihiro Yamamoto 山本 章博
http://www.iip.ist.i.kyoto-u.ac.jp/member/akihiro/
akihiro@i.kyoto-u.ac.jp

Contents

- Bit Vectors
- Item Set Mining
- The A Priori Algorithm
- Depth-First Search
- Formal Concept Analysis
- Closed Patterns

LEARNING FROM BIT VECTORS

Data Structure : Bit Vector

- An *n*-dimension bit vector is just a sequence composed of *n* bits where a bit is from {0, 1}.

Example: (0, 0, 1, 1, 0, 0, 1, 1)

  - In the last part of this course, we assume the length of sequences in data set should be fixed and equal to *n*.
  - Sometimes each dimension of vectors is indicated with a specific name called an attribute.

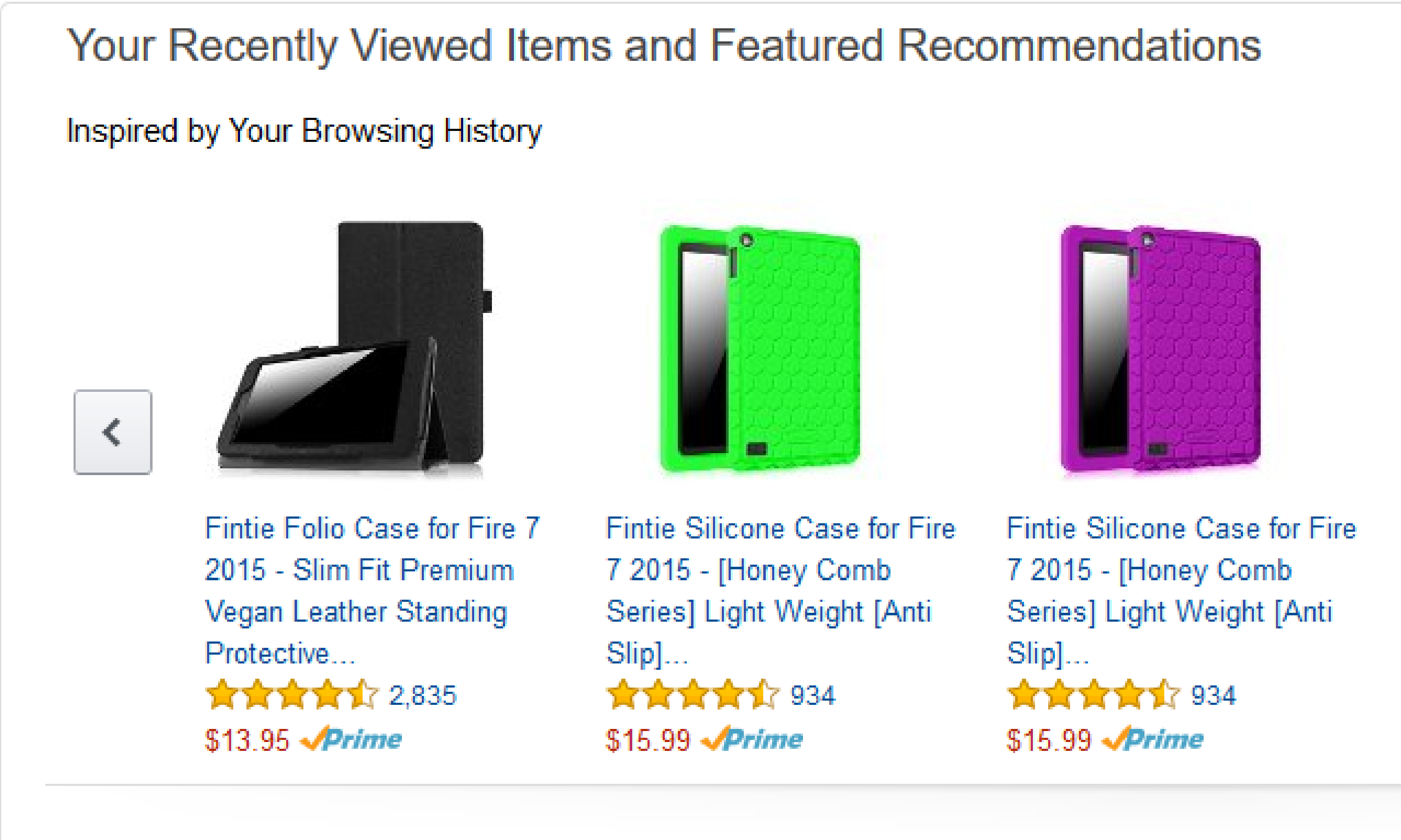
ID	A	B	C	D	E	F
1	1	0	1	1	0	0
2	0	1	1	0	1	0
3	1	1	1	0	1	0
4	1	1	0	0	1	1

ITEM SET MINING

What is item set mining?

- Originally from market basket analysis or affinity analysis
  - Market basket analysis might tell a retailer that customers often purchase shampoo and conditioner together. [Wikipedia]
- Discovering co-occurrence relationships among activities performed by (or recorded about) specific individuals or groups [Wikipedia]

For Reccomendations



A Simple Example

- Set of all items: $X = \{A, B, C, D, E, F\}$

Transaction ID	Item Sets
…	
3256	{A, C, D}
3257	{B, C, E}
3258	{A, B, C, E}
3259	{A, B, E, F}
…	….

- "Items A and C might be bought together."

Bit-vector Representation

- Every transaction can be represented as a bit-vector of n dimension, where $n = |X|$.

ID	A	B	C	D	E	F
…						
3256	1	0	1	1	0	0
3257	0	1	1	0	1	0
3258	1	1	1	0	1	0
3259	1	1	0	0	1	1
…						

Bag of Words

- Let $X = \{A_1, A_2, \ldots, A_k)$ be a finite set of words.
- For a sentence s, we define $T(s) = (x_1, x_2, \ldots, x_k)$ where

$$x_i = 1 \text{ if word } A_i \text{ appears in } s$$
$$= 0 \text{ o.w.}$$

for $i = 1, 2, \ldots, n$

Example

W =(arithmetic, book, compute, paper, suppose, square, symbol, write)

s_1: Computing is normally done by writing certain symbols on paper.

s_2: We may suppose this paper is divided into squares like a child's arithmetic book.

$$T(s_1) = (0, 0, 1, 1, 0, 0, 1, 1)$$
$$T(s_2) = (1, 1, 0, 1, 1, 1, 0, 0)$$

Mathematical Definitions

- Assuming a finite set of all items as attributes

$$X = \{A_1, A_2, \ldots, A_n\}$$

- A transaction is a pair $t = (i, T)$ of an identifier $i \in \mathbf{N}$ and a finite set of items $T \in X$
- A transaction database D is a finite set of transactions in which no pair of transactions have a same identifier, that is,

$$t = (i, T) \in D \text{ and } s = (j, S) \in D \text{ imply } i \neq j.$$

- A pattern is a finite set of items.
  - Transactions are for training data patterns are rules.

Mathematical Definitions (2)

- For a pattern P and a transaction $t = (i, T)$, we say t satisfies P (or P matches t) iff $P \subset T$.
- Let $D(P) = \{ t \mid P \text{ matches } t \}$.
- The support of P in a transaction database D is defined as supp(P) = $| D(P) | / | D |$.
  - The support is also called the relative frequency.

Definition of Learning Task

- Assuming a set of items X
- For a given transaction database D and a minimal support (threshold) σ s.t. $0 \leq \sigma \leq 1$, enumerate all patterns P s.t. supp(P) $\geq \sigma$.

A Very Simple Example

ID	A	B	C	D	E	F
1	1	0	1	1	0	0
2	0	1	1	0	1	0
3	1	1	1	0	1	0
4	1	1	0	0	1	1

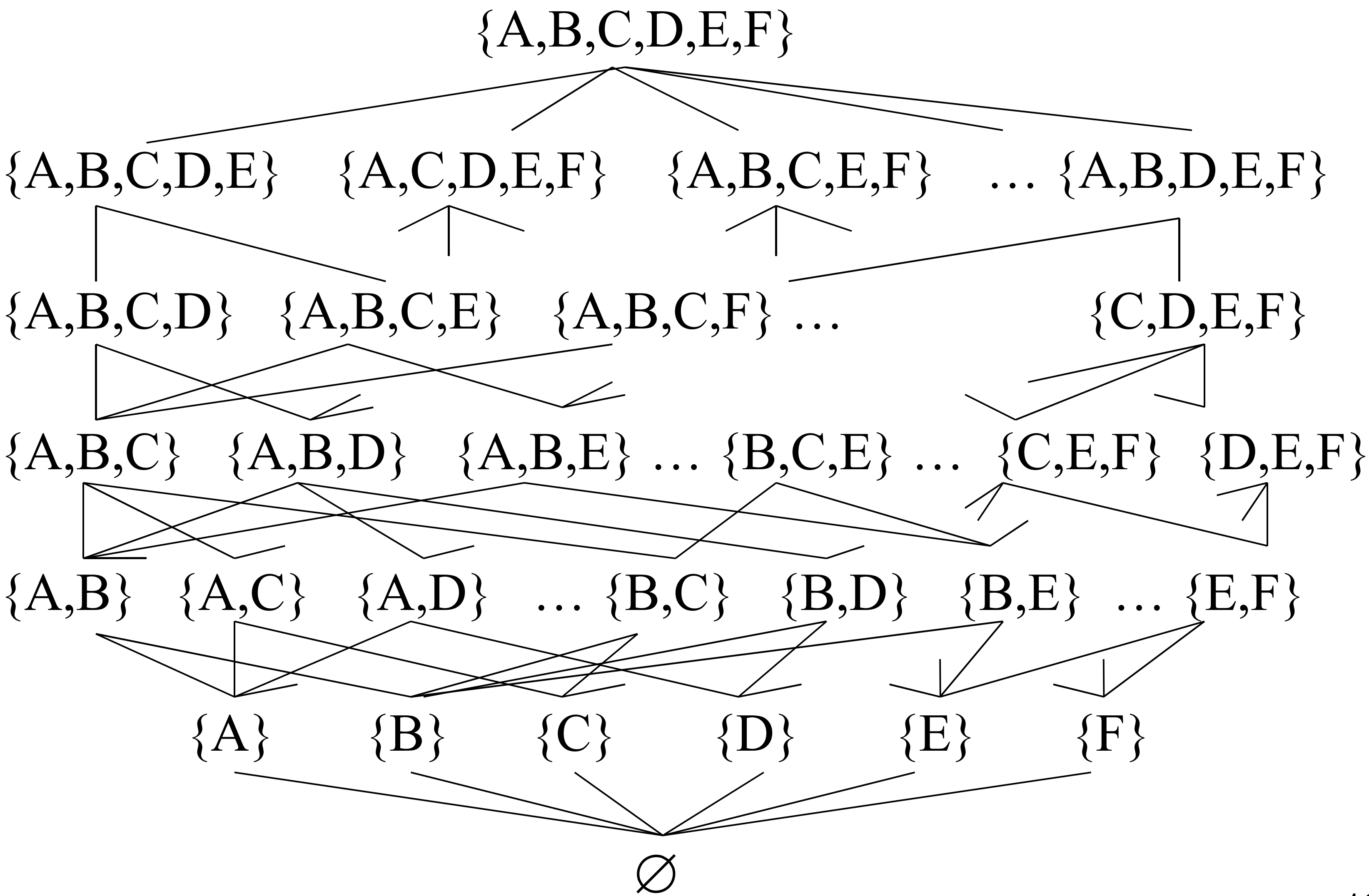
supp({A})= supp({B})= supp({C})= supp({E})= 0.75,

supp({D})= supp({F})= 0.25

supp({A, B})=supp({A, C})=0.5, supp({A, D})=0.25,…

THE A-PRIORI ALGORITHM

Hasse Diagram of Patterns



Monotonicity of the Support

Lemma For two patterns P and Q,

$$P \subseteq Q \Rightarrow \text{supp}(P) \geq \text{supp}(Q)$$

ID	A	B	C	D	E	F
1	1	0	1	1	0	0
2	0	1	1	0	1	0
3	1	1	1	0	1	0
4	1	1	0	0	1	1

supp({A})=0.75 ≥ supp({A , B})=0.25

supp({B})=0.5≥ supp({A , B})=0.25

supp({A})= 0.75 ≥ supp({A , C})=0.5

A Priori Algorithm [Agrawal et al. 93]

1. Let $k = 1$.
2. Let $C_1 = \{\{A\} \mid A \in X\}$.
3. Let $L_k = \{P \in C_k \mid \mathrm{supp}(P) \geq \sigma\}$.
4. If $L_k = \varnothing$ then halt, otherwise

   Let $C_{k+1} = \{\, P \cup Q \mid P \in L_k, Q \in L_k, |P \cup Q| = k+1,$ but $P \cup Q$ does not subsumes any $R \in C_i - L_i\,(i \leq k) \}$.

   Increment k.

   Repeat Step 4.

An Example of Run(1)

ID	A	B	C	D	E	F
1	1	0	1	1	0	0
2	0	1	1	0	1	0
3	1	1	1	0	1	0
4	1	1	0	0	1	1

$\sigma = 0.5$

$C_1 = \{\{A\}, \{B\}, \ldots, \{F\}\}$

$L_1 = \{\{A\},\{B\},\{C\},\{E\}\}$

$C_2 = \{\{A, B\}, \{A, C\}, \{A, E\}, \{B, C\}, \{B, E\}, \{C, E\}\}$

$L_2 = \{\{A, B\},\{A, C\}, \{A, E\}, \{B, C\}, \{B, E\}, \{C, E\}\}$

$C_3 = \{\{A,B,C\}, \{A,B,E\} \{B,C,E\}\}$

$L_3 = \{\{A,B,E\},\{B,C,E\}\}$

An Example of Run (2)

ID	A	B	C	D	E	F
1	1	0	1	1	0	0
2	0	1	1	0	1	0
3	1	1	1	0	1	0
4	0	1	0	0	1	1

$\sigma = 0.5$

C_1 = {{A}, {B}, …, {F}}

L_1 = {{A},{B},{C},{E}}

C_2 = {~~{A, B},~~ {A, C},
~~{A, E},~~ {B, C},
{B, E}, {C, E}}

L_2 = {{A, C}, {B,C},
{B, E}, {C, E}}

C_3 = {~~{A,B,C},~~ ~~{A,B,E}~~
{B,C,E}}

L_3 = {B,C,E}

An Example of Run(3)

ID	A	B	C	D	E	F
1	1	1	0	1	0	0
2	0	1	1	0	1	0
3	1	1	1	0	1	0
4	0	1	0	0	1	1

$\sigma = 0.5$

$C_1 = \{\{A\}, \{B\}, \ldots, \{F\}\}$

$L_1 = \{\{A\},\{B\},\{C\},\{E\}\}$

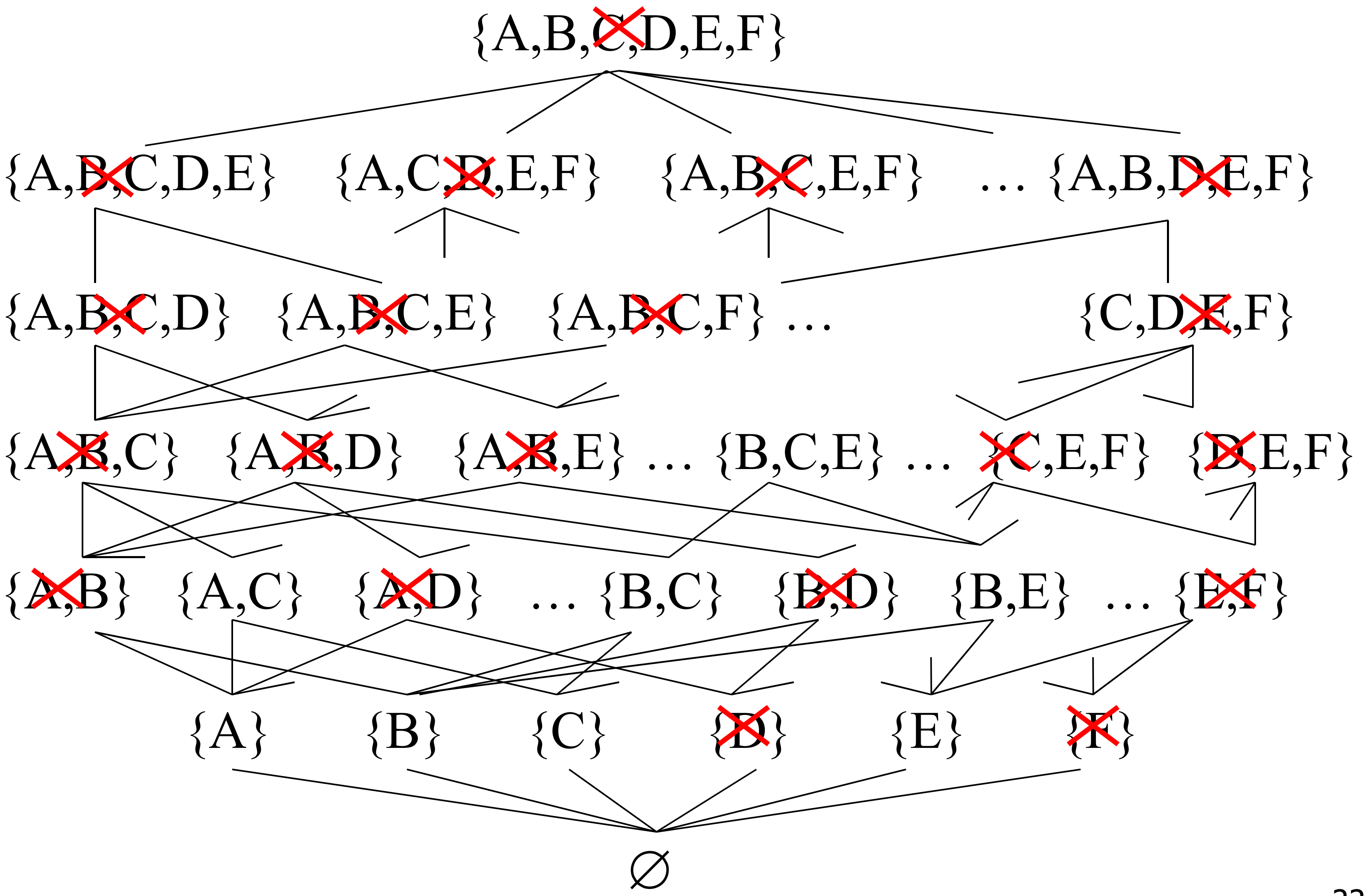
$C_2 = \{\{A, B\},$ ~~$\{A, C\},$~~ ~~$\{A, E\},$~~ $\{B, C\}, \{B, E\}, \{C, E\}\}$

$L_2 = \{\{A, B\}, \{B, C\}, \{B, E\}, \{C, E\},\}$

$C_3 = \{$~~$\{A, B, C\}$~~$,$ ~~$\{A, B, E\}$~~$,$ ~~$\{A, C, E\}$~~$, \{B, C, E\}\}$

$L_3 = \{\{B, C, E\}\}$

Hasse Diagram of Patterns



DEPTH-FIRST SEARCH

A Propri Algorithm [Agrawal et al. 93]

1. Let $k = 1$.
2. Let $C_1 = \{\{A\} \mid A \in X\}$.
3. Let $L_k = \{P \in C_k \mid \text{supp}(P) \geq \sigma\}$.
4. If $L_k = \varnothing$ then halt, otherwise

   Let $C_{k+1} = \{ P \cup Q \mid P \in L_k, Q \in L_k, |P \cup Q| = k+1,$ but $P \cup Q$ does not subsumes any $R \in C_i - L_i \ (i \leq k)$ $\}$.

   Increment k.

   Repeat Step 4.

Depth-First Search Algorithm(1)

- Assuming a total ordering for the set X of items.

  Example : A > B > C > D > E > F

- Regarding (Implementing) every pattern $P \in L_k$ as a list of items in which items are ordered in the descending order.

For two lists $P = [P', A_i] \in L_k$ and $Q = [P', A_j] \in L_k$ of descending order, the list $[P', A_i, A_j]$ does not subsume any $R \in C_i - L_i \, (i \leq k)$.

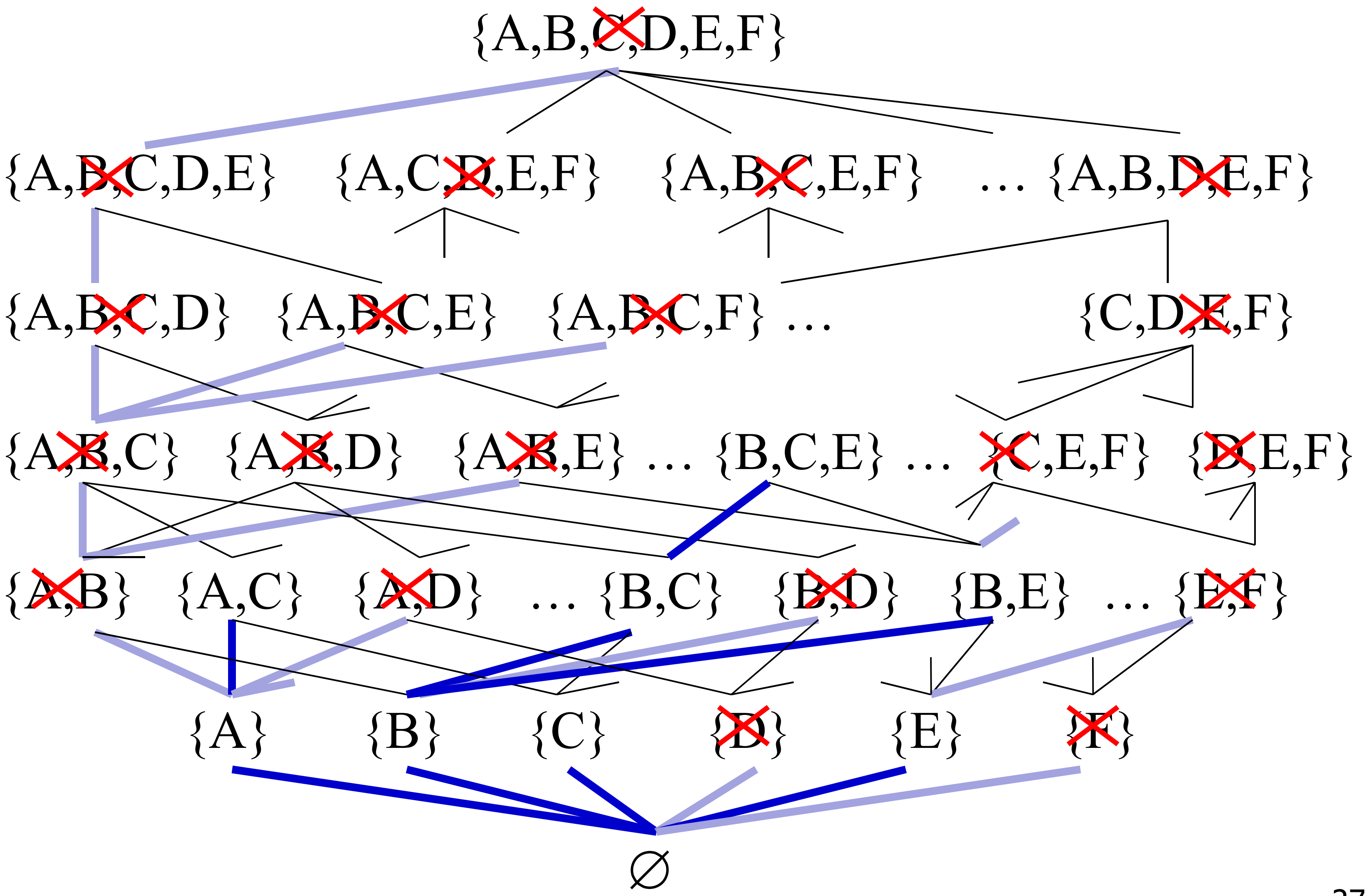
Depth-First Search Algorithm(2)

- A more simplified method is to let

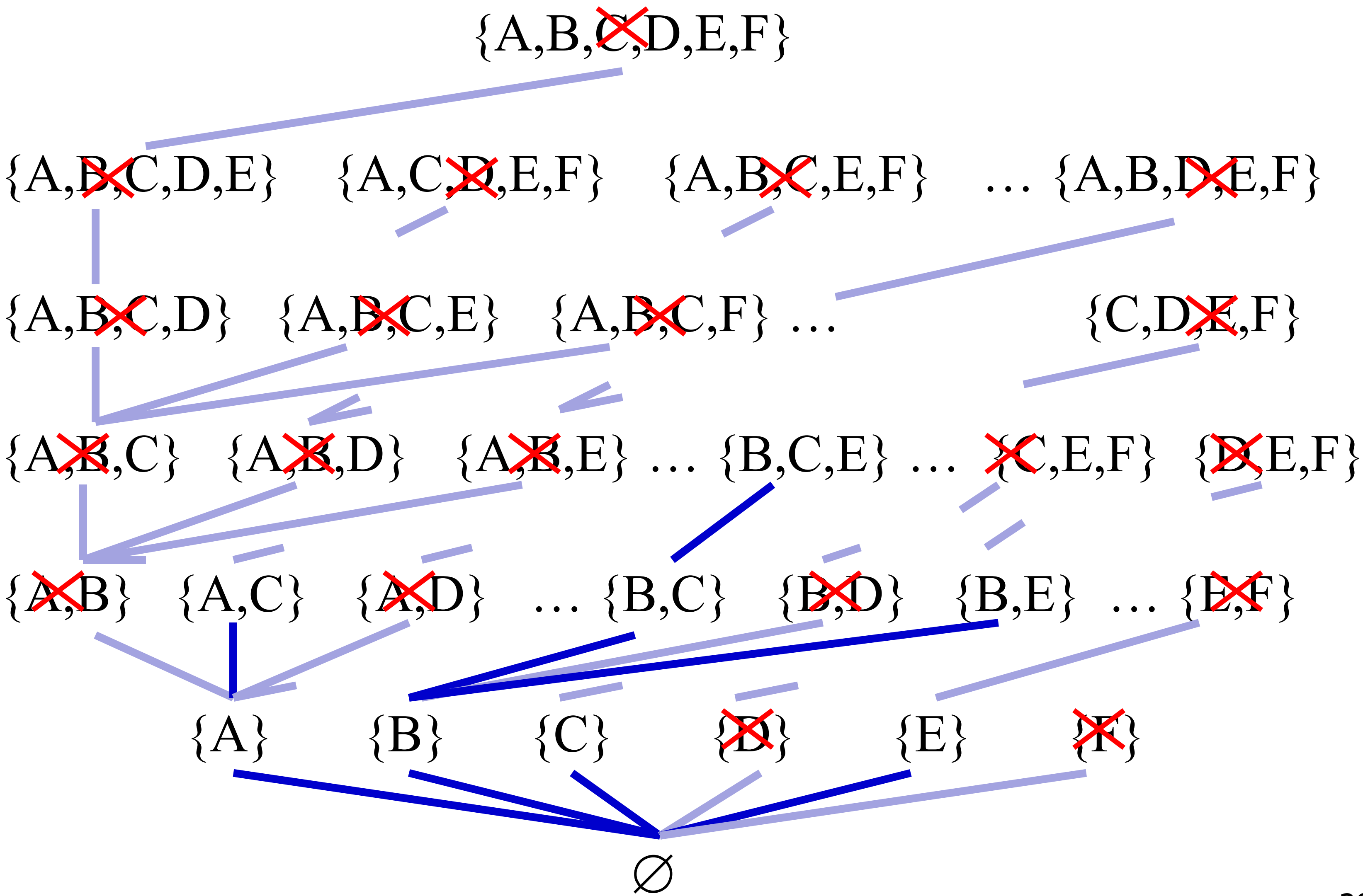
  $C_{k+1} = \{\ [P, A_i\ , A_j] \mid [P\ , A_i] \in L_k \text{ and } A_i > A_j\ \}$

- Instead of this version of C_{k+1} as it is, we can design a depth-first search algorothm.

Depth-First Search in the Diagram



Depth-First Search in the Diagram

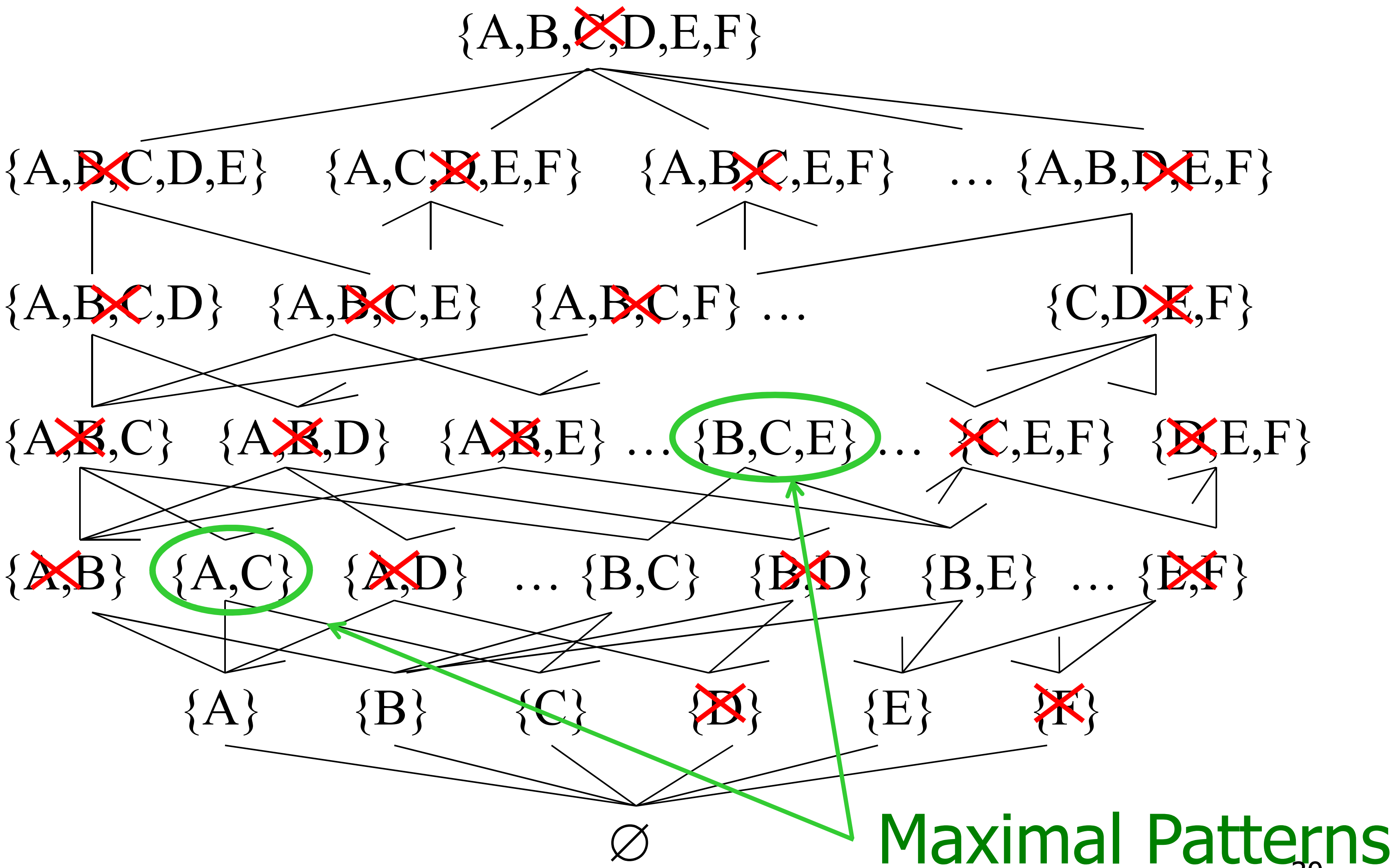


Maximal Patterns

- A pattern P is maximal as the answer of the task if
  $\mathrm{supp}(P) \geq \sigma$ and
  no pattern Q s.t. $Q \supset P$ satisfies $\mathrm{supp}(Q) \geq \sigma$.

- From the monotonicity of the support function, every subset S of a maximal pattern P satisfies
  $\mathrm{supp}(S) \geq \sigma$.
  - We may enumerate only maximal patterns.

Maximal Patterns in the Hasse Diagram



FP-TREES

What is an FP-Tree?

- We regard (implement) every transaction as a list of items in the descending order defined by the support of each item.
  - When a minimal support σ is given, we can neglect all items A such that supp(A) < σ.
- We regard (implement) a transaction database D as a prefix tree $T'(D)$.
- An FP-tree $T(D)$ is obtained by giving links among nodes whose labels are same.

Example of FP-tree(1)

ID	Item Set
1	{A, B, D}
2	{B, C, E}
3	{A, B, C, E}
4	{B, E, F}

ID	A	B	C	D	E	F
1	1	1	0	1	0	0
2	0	1	1	0	1	0
3	1	1	1	0	1	0
4	0	1	0	0	1	1

- Constructing the table of the supports

  $\sigma = 0.5$

B	4	
E	3	
A	2	
C	2	
~~D~~	~~1~~	
~~E~~	~~1~~	

Example of FP-tree(2)

- Represent every transaction as a list of the descending order.

B	4	
E	3	
A	2	
C	2	

ID	Item Set
1	{A, B, D}
2	{B, C, E}
3	{A, B, C, E}
4	{B, E, F}

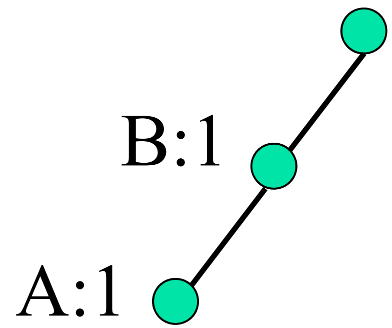
ID	Item List
1	[B, A~~, D~~]
2	[B, E, C]
3	[B, E, A, C]
4	[B, E~~, F~~]

Example of FP-tree(3)

ID	Item List
1	[B, A]
2	[B, E, C]
3	[B, E, A, C]
4	[B, E]

$\sigma = 0.5$

B	4	
E	3	
A	2	
C	2	

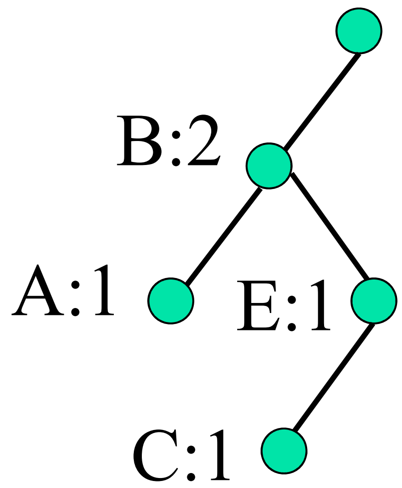


Example of FP-tree(4)

ID	Item List
1	[B, A]
2	[B, E, C]
3	[B, E, A, C]
4	[B, E]

$\sigma = 0.5$

B	4	
E	3	
A	2	
C	2	

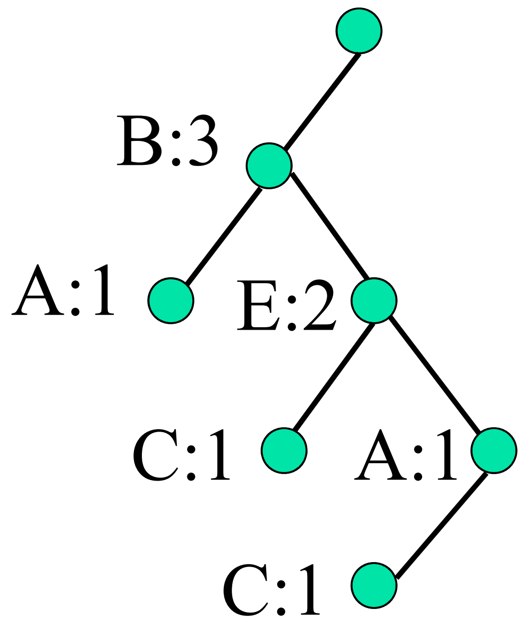


Example of FP-tree(5)

ID	Item List
1	[B, A]
2	[B, E, C]
3	[B, E, A, C]
4	[B, E]

$\sigma = 0.5$

B	4	
E	3	
A	2	
C	2	

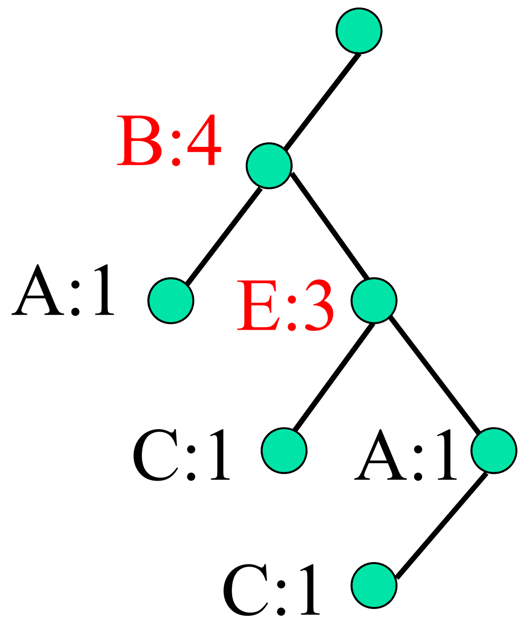


Example of FP-tree(6)

ID	Item List
1	[B, A]
2	[B, E, C]
3	[B, E, A, C]
4	[B, E]

$\sigma = 0.5$

B	4	
E	3	
A	2	
C	2	

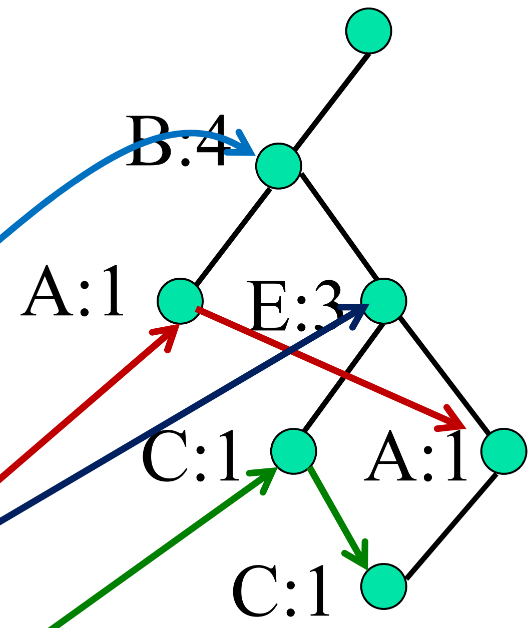


Example of FP-tree(7)

ID	Item List
1	[B, A]
2	[B, E, C]
3	[B, E, A, C]
4	[B, E]

$\sigma = 0.5$

B	4	
E	3	
A	2	
C	2	



The FP-Growth Algorithm[Han et al. 00]

- Given a minimal support σ
- Let L be the list of items $[A_1, A_2 \ldots, A_m]$ satisfying $\text{supp}(A_k) \geq \sigma$ in the ascending order of the support.

FP-growth(T, L)

1. If T consists of one path p, enumerate all patterns at least one A_i s.t. $\text{supp}_N(A_i) \geq \sigma$ and all items in L.
2. For $k = 1,2, \ldots, n$, repeat the following:

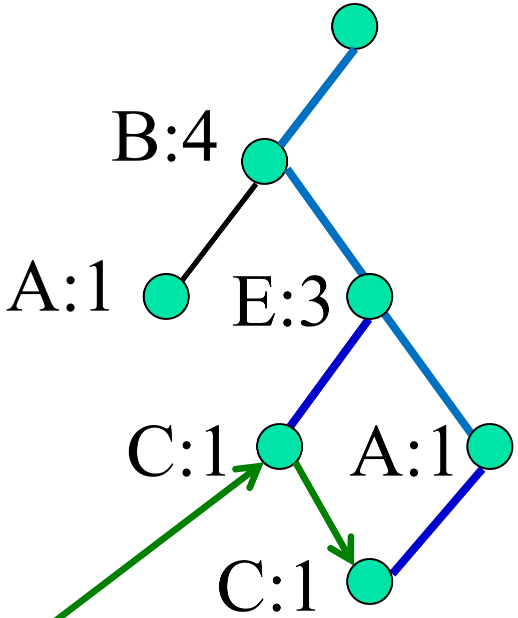
- Construct the conditional transaction database D' and FP-tree $T(D')$ by gathering items from the root of T and the parent of A_k, and execute FP-growth(T', $[A_k, L]$).

Example Run of FP-Growth(1-1)

ID	Item List
1	[B, A]
2	[B, E, C]
3	[B, E, A, C]
4	[B, E]

$\sigma = 0.5$

B	4	
E	3	
A	2	
C	2	



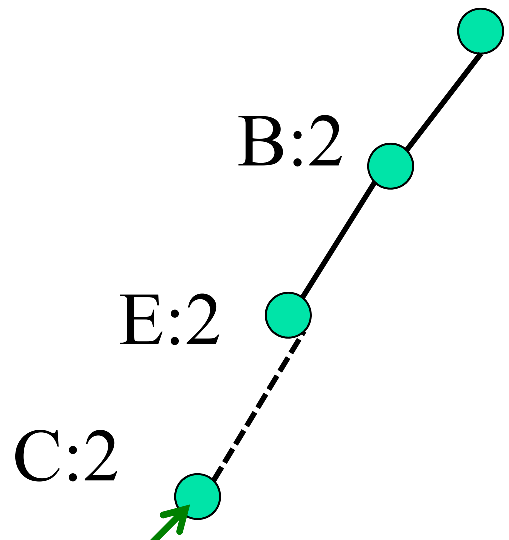
Conditional Data

Item List	supp
[B, E, C]	1
[B, E, A, C]	1

Example Run of FP-Growth(1-2)

ID	Item List
1	[B, A]
2	[B, E, C]
3	[B, E, A, C]
4	[B, E]

$\sigma = 0.5$



B	2	
E	2	
C	2	

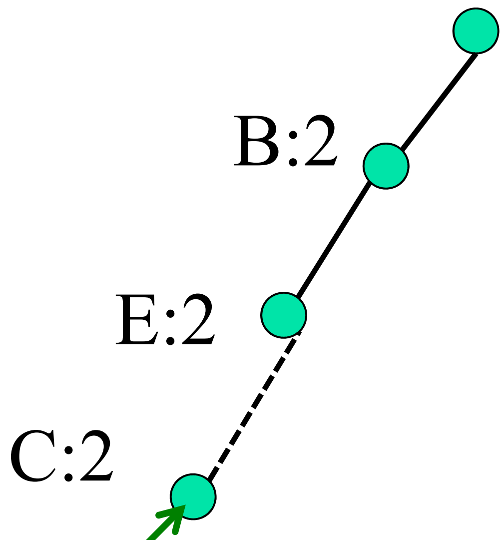
Conditional Data

Item List	supp
[B, E, C]	1
[B, E, A, C]	1

Example Run of FP-Growth(1-3)

ID	Item List
1	[B, A]
2	[B, E, C]
3	[B, E, A, C]
4	[B, E]

$\sigma = 0.5$



B	2	
E	2	
C	2	

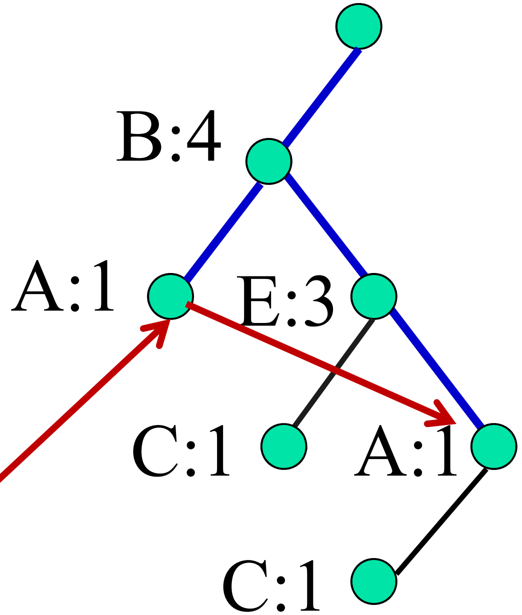
supp({ B, E, C}) =0.5
supp({ B, C }) =0.5
supp({ E, C }) =0.5
supp({ C }) =0.5

Example Run of FP-Growth(1-4)

ID	Item List
1	[B, A]
2	[B, E, C]
3	[B, E, A, C]
4	[B, E]

$\sigma = 0.5$

B	4	
E	3	
A	2	
C	2	



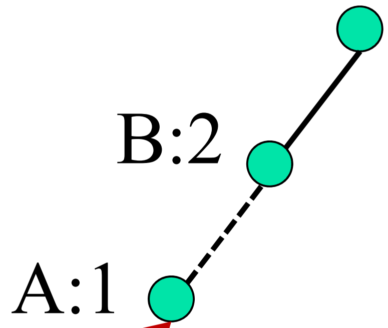
Conditional Data

Item List	supp
[B, A]	1
[B, E, A]	1

Example Run of FP-Growth(1-5)

ID	Item List
1	[B, A]
2	[B, E, C]
3	[B, E, A, C]
4	[B, E]

$\sigma = 0.5$



B	2	
A	2	

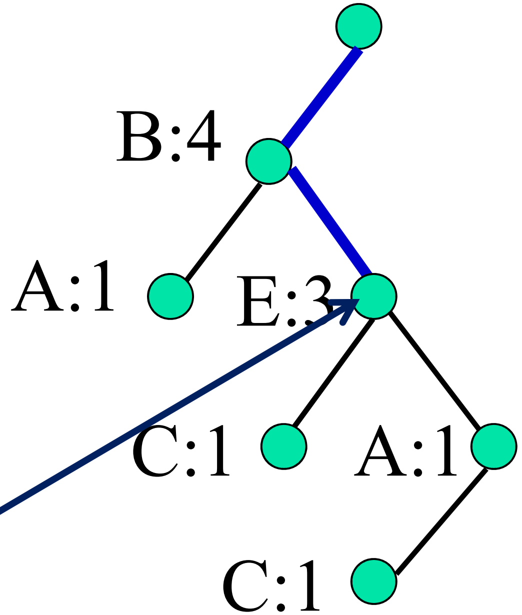
supp({B, A})=0.5
supp({A })=0.5

Example Run of FP-Growth(1-6)

ID	A	B	C	D	E	F
1	1	1	0	1	0	0
2	0	1	1	0	1	0
3	1	1	1	0	1	0
4	0	1	0	0	1	1

$\sigma = 0.5$

B	4	
E	3	
A	2	
C	2	



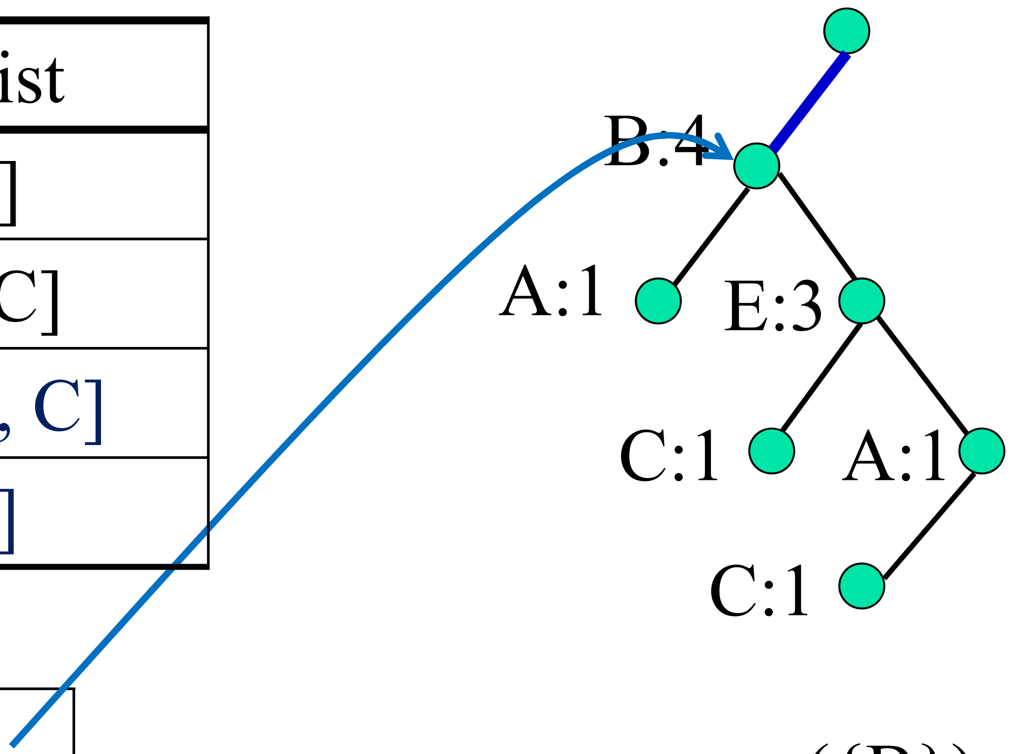
supp({ B, E})=0.75

Example Run of FP-Growth(1-7)

ID	Item List
1	[B, A]
2	[B, E, C]
3	[B, E, A, C]
4	[B, E]

$\sigma = 0.5$

B	4	
E	3	
A	2	
C	2	



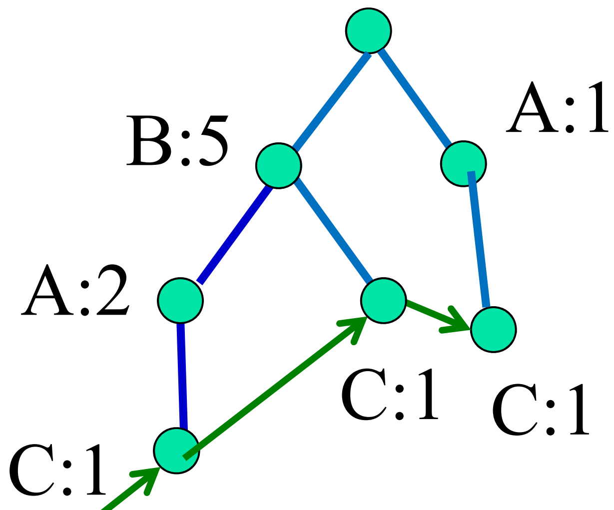
supp({B})=1

Example Run of FP-Growth(2-1)

ID	Item List
1	[B,A]
2	[B, C, D]
3	[B, A, C]
4	[B,A]
5	[A, C]
6	[B, E]

$\sigma = 0.3$

B	5	
A	4	
C	3	



Conditional Data

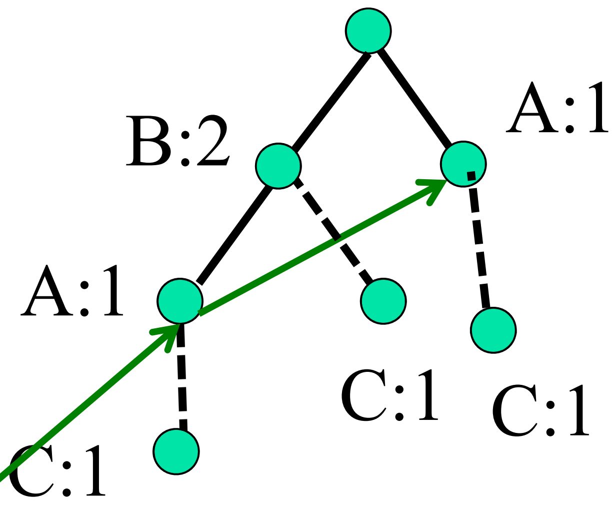
Item List	supp
[B, A, C]	1
[B, C]	1
[A, C]	1

Example Run of FP-Growth(2-2)

ID	Item List
1	[B,A]
2	[B, C, D]
3	[B, A, C]
4	[B,A]
5	[A, C]
6	[B, E]

$\sigma = 0.3$

B	2	
A	2	



Conditional Data

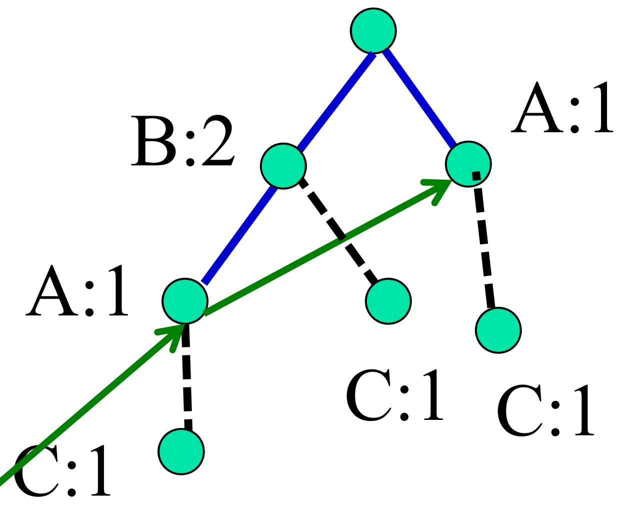
Item List	supp
[B, A, C]	1
[B, C]	1
[A, C]	1

Example Run of FP-Growth(2-3)

ID	Item List
1	[B,A]
2	[B, C, D]
3	[B, A, C]
4	[B,A]
5	[A, C]
6	[B, E]

$\sigma = 0.3$

B	2	
A	2	



Conditional Data

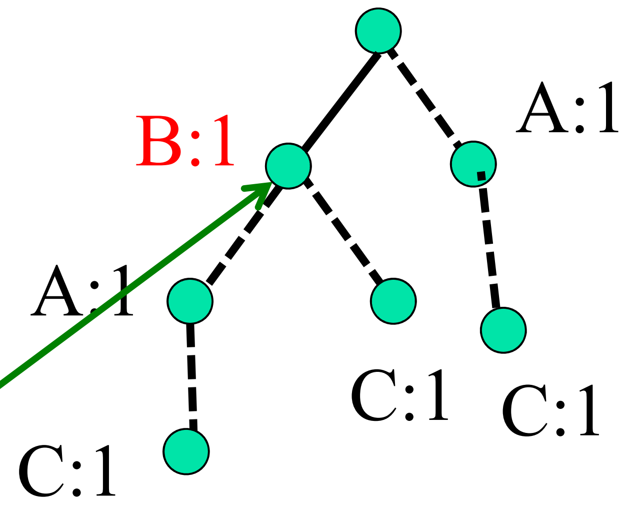
Item List	supp
[B, A, C]	1
[A, C]	1

Example Run of FP-Growth(2-4)

ID	Item List
1	[B,A]
2	[B, C, D]
3	[B, A, C]
4	[B,A]
5	[A, C]
6	[B, E]

$\sigma = 0.3$

B	1	



supp{A, C}=0.333…

Conditional Data

Item List	supp
[B, A, C]	1
[A, C]	1

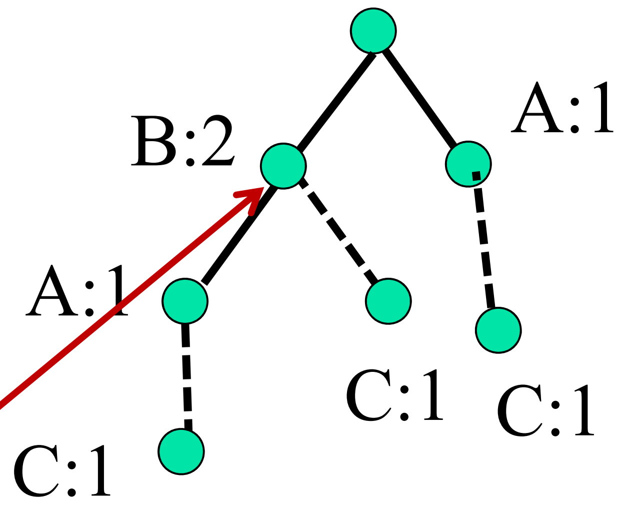
Example Run of FP-Growth(2-5)

ID	Item List
1	[B,A]
2	[B, C, D]
3	[B, A, C]
4	[B,A]
5	[A, C]
6	[B, E]

$\sigma = 0.3$

B	2	
A	2	

supp{B, C}=0.333…
supp{C}=0.333…



Conditional Data

Item List	supp
[B, A, C]	1
[B, C]	1
[A, C]	1

FORMAL CONCEPT ANALYSIS

A Simple Example

- Set of all items: $X = \{A, B, C, D, E, F\}$

Transaction ID	Item Sets
…	
3256	{A, C, D}
3257	{B, C, E}
3258	{A, B, C, E}
3259	{A, B, E, F}
…	….

- "Items A and C might be bought together."

Bit-vector Representation

- Every transaction can be represented as a bit-vector of n dimension, where $n = |X|$.

ID	A	B	C	D	E	F
…						
3256	1	0	1	1	0	0
3257	0	1	1	0	1	0
3258	1	1	1	0	1	0
3259	1	1	0	0	1	1
…						

Context Table Representation

- Instead of "1", we use ●.

ID	A	B	C	D	E	F
…						
3256	●		●	●		
3257		●	●		●	
3258	●	●	●		●	
3259	●	●			●	●
…						

Formal Concepts

- A formal concept is a maximal rectangular filled with ●, without considering the ordering of law and column.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}	m_{11}	m_{12}
g_1	●	●	●	●	●	●	●	●	●	●		
g_2	●	●	●	●			●	●				
g_3	●	●	●		●	●					●	●
g_4	●	●		●	●	●					●	●

	m_1	m_2	m_3	m_4	m_7	m_8	m_5	m_6	m_9	m_{10}	m_{11}	m_{12}
g_1	●	●	●	●	●	●	●	●	●	●		
g_2	●	●	●	●	●	●						
g_3	●	●	●				●	●			●	●
g_4	●	●		●			●	●			●	●

Intuitive Explanation

In the context of item set mining, a formal concept is a pair of a set *A* of transaction and a set *B* of items such that

- every transaction in *A* contains all items in *B*,
- every items in *B* is contained by all transactions in *A*,
- for every item *i* which is not in *B*, at least one transaction in *A* does not contain *i*, and
- for every transaction *t* which is not in *A*, at least one item is not contained by *t*.

Mathematical Definition

- A formal context $K=(G, M, I)$ consists of two sets G (objects, *Gegenstand*) and M (attributes, *Merkmal*) and a binary relation $I \subseteq G \times M$.
- We define two functions $f: 2^G \to 2^M$ and $h: 2^M \to 2^G$

$$f(A) = \{ m \in M \mid (g, m) \in I \text{ for all } g \in A \}$$

$$h(B) = \{ g \in G \mid (g, m) \in I \text{ for all } m \in B \}$$

  - The pair (f, h) is called a Galois connection between 2^G and 2^M.
- A formal concept of K is a pair $C=(A, B)$ with $A \subseteq G$ and $B \subseteq M$ such that $f(A)=B$ and $h(B) = A$, i.e.

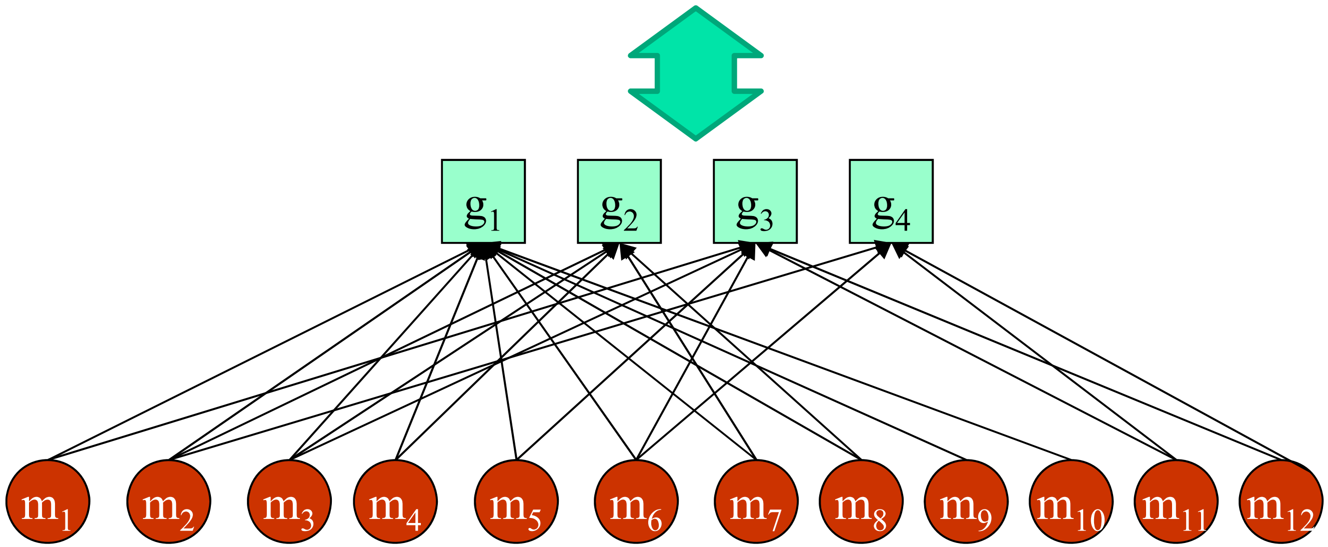
$$h(f(A))=A \text{ and } f(h(B))=B.$$

  - A is called the extent of C and B is called the intent of C.

Bipartite Graph Representation

- Every context table can be represented as a bipartite graph.
- Every formal concept is a represented as a bipartite clique.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}	m_{11}	m_{12}
g_1	●	●	●	●	●	●	●	●	●	●		
g_2	●	●	●	●			●	●				
g_3	●	●	●		●	●					●	●
g_4	●	●		●	●	●					●	●



Some Propositions

For a context $K=(G, M, I)$, $A, A_1, A_2 \subseteq G$ and $B, B_1, B_2 \subseteq M$,

- $A_1 \subseteq A_2 \Rightarrow f(A_2) \subseteq f(A_1)$
- $B_1 \subseteq B_2 \Rightarrow h(B_2) \subseteq h(B_1)$
- $A \subseteq h(f(A))$
- $B \subseteq f(h(B))$
- $A \subseteq h(B) \Leftrightarrow B \subseteq f(A) \Leftrightarrow A \times B \subseteq I$

- $h(f(A_1 \cup A_2)) = h(f((h(f(A_1)) \cup h(f(A_2))))$
- $f(h(B_1 \cup B_2)) = f(h((f(h(B_1)) \cup f(h(B_2))))$
- $A_1 \subseteq h(f(A_2) \Rightarrow h(f(A_1)=h(f(A_2))$ and $h(f(A_1 \cup A)=h(f(A_2 \cup A))$
- $B_1 \subseteq f(h(B_2) \Rightarrow f(h(B_1)=f(h(B_2))$ and $f(h(B_1 \cup B)=f(h(B_2 \cup B))$

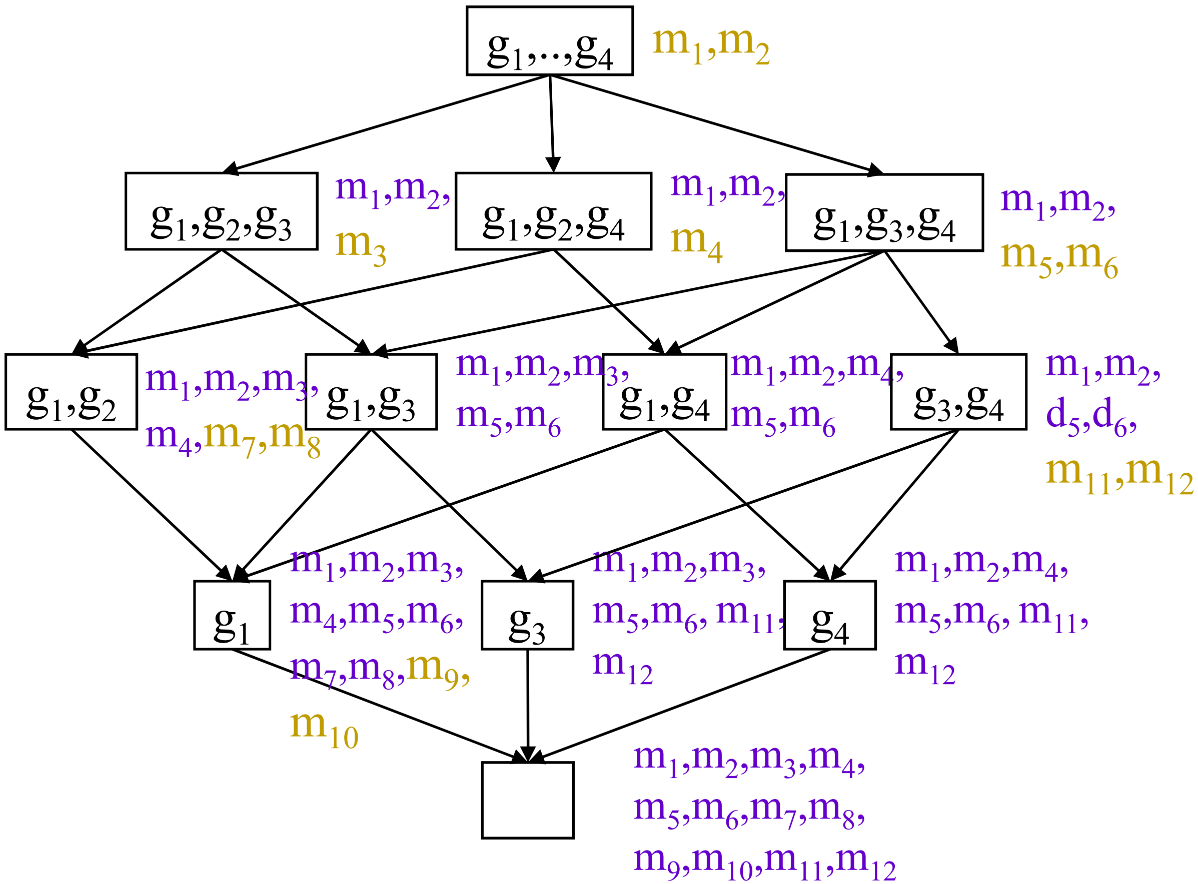
Some Propositions

For formal concepts C_1=(A_1, B_1) and C_2=(A_2, B_2),

$$A_1 \subseteq A_2 \Leftrightarrow B_2 \subseteq B_1$$

Hasse Diagram of FCs

- We can draw another Hasse diagram with all of the formal concepts.



CLOSED PATTERNS

Closed Item Sets [Pasquier et al.]

- For a transaction data, we let *G* is the set of all transaction id and *M* is the set of all items.
- An pattern *B* is closed iff $B = f(h(B))$, i.e, $(h(B), B)$ is a formal concept.

1	*a*		*c*	*d*	
2		*b*	*c*		*e*
3	*a*	*b*	*c*		*e*
4		*b*			*e*
5	*a*	*b*	*c*		*e*

$\sigma = 0.5$

Frequent closed pattern: *c*, *ac*, *be*, *bce*

Frequent but not closed pattern: *a*, *bc*, …

- For a transaction data, we let *G* is the set of all transaction ids and *M* is the set of all items.

Lemmas

Lemma For a context $K=(G, M, I)$, $A \subseteq G$ and $B \subseteq M$

- $h(f(A)) = \cap_{g \in G} \{f(\{g\}) \mid A \subseteq f(\{g\})\}$
- $f(h(B)) = \cap_{m \in M} \{h(\{m\}) \mid B \subseteq f(\{m\})\}$

Corollary For closed patterns B_2, if $B_2 \subseteq B_1$ and $B_2 \neq B_1$, then supp(B_2) > supp(B_1).
Corollary For two closed patterns B_1 and B_2, if $B_2 \subseteq B_1$ and $B_2 \neq B_1$, then supp(B_2) > supp(B_1).

Lemma [Pasquier et al.] Every pattern B_1 of supp(B_1) = σ can be derived from some closed pattern B_2 of supp(B_2)= σ.

Proposition

Proposition Every maximally frequent closed pattern is a frequent closed pattern.

An Example of Run(1)

ID	A	B	C	D	E	F
1	1	0	1	1	0	0
2	0	1	1	0	1	0
3	1	1	1	0	1	0
4	1	1	0	0	1	1

$\sigma = 0.5$

$C_1 = \{\{A\}, \{B\}, \ldots, \{F\}\}$

$L_1 = \{\{A\},\{B\},\{C\},\{E\}\}$

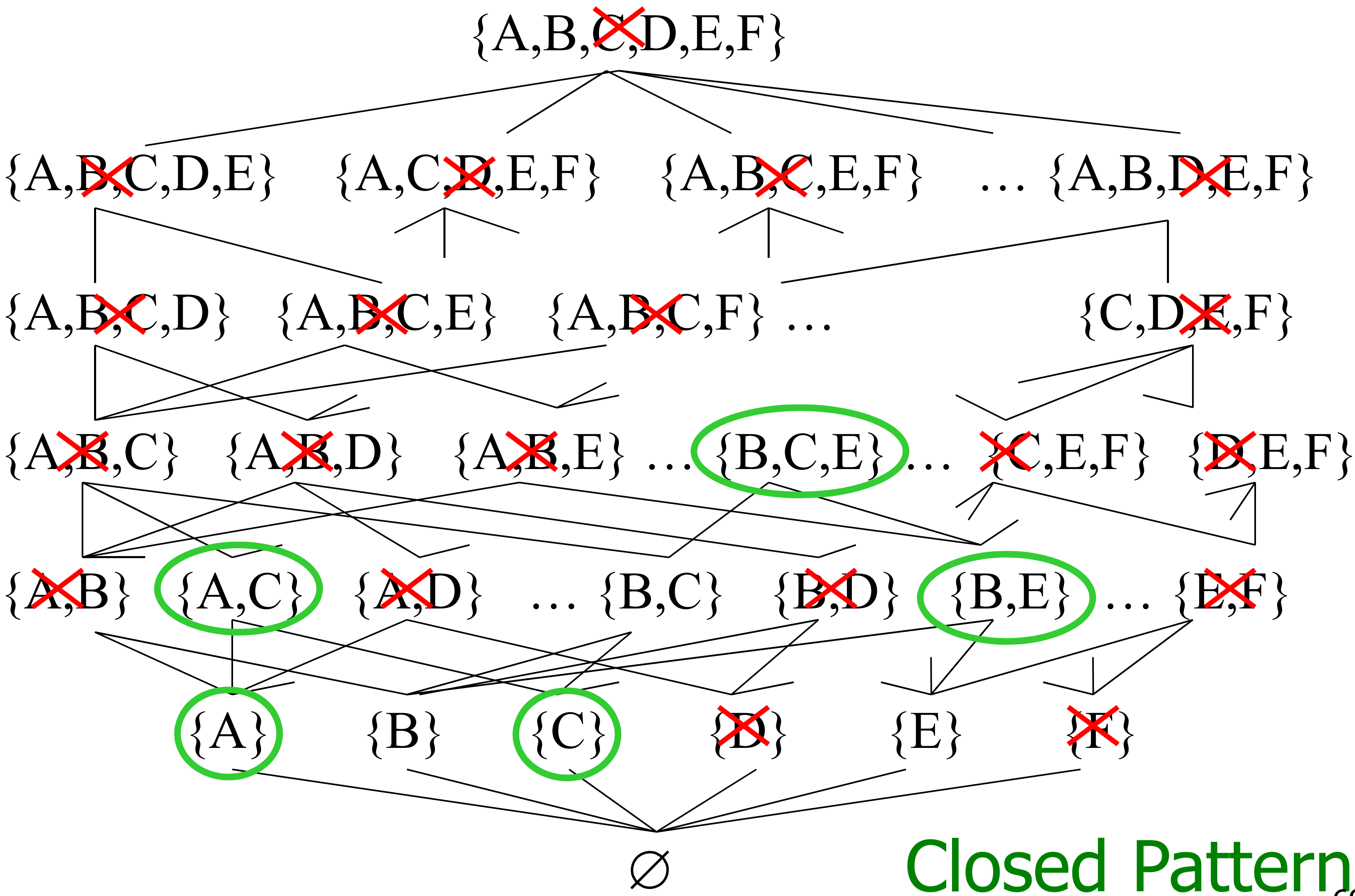
$C_2 = \{\{A, B\}, \{A, C\}, \{A, E\}, \{B, C\}, \{B, E\}, \{C, E\}\}$

$L_2 = \{\{A, B\},\{A, C\}, \{A, E\}, \{B, C\}, \{B, E\}, \{C, E\}\}$

$C_3 = \{\{A,B,C\}, \{A,B,E\} \{B,C,E\}\}$

$L_3 = \{\{A,B,E\},\{B,C,E\}\}$

Maximal Patterns in the Hasse Diagram



Closed Patterns

Frequent Closed ItemSets

$\sigma = 0.5$

ID	A	B	C	D	E	F
1	●		●	●		
2		●	●		●	
3	●	●	●		●	
4	●	●			●	●

ID	A	B	C	D	E	F
1	●		●	●		
2		●	●		●	
3	●	●	●		●	
4	●	●			●	●

ID	A	B	C	D	E	F
1	●		●	●		
2		●	●		●	
3	●	●	●		●	
4	●	●			●	●

Frequent Closed ItemSets

$\sigma = 0.25$

ID	A	B	C	D	E	F
1	●		●	●		
2		●	●		●	
3	●	●	●		●	
4	●	●			●	●

ID	A	B	C	D	E	F
1	●		●	●		
2		●	●		●	
3	●	●	●		●	
4	●	●			●	●

ID	A	B	C	D	E	F
1	●		●	●		
2		●	●		●	
3	●	●	●		●	
4	●	●			●	●

ID	A	B	C	D	E	F
1	●		●	●		
2		●	●		●	
3	●	●	●		●	
4	●	●			●	●

Available Algorithm

Takeaki Uno and Tatsuya Asai, Hiroaki Arimura and Yuzo Uchida LCM: An Efficient Algorithm for Enumerating Frequent Closed Item, IEEE ICDM'04 Workshop FIMI'03