# BIDE: Efficient Mining of Frequent Closed Sequences

Jianyong Wang and Jiawei Han
University of Illinois at Urbana-Champaign

*To appear in ICDE 2004*

Presented by: Yi-Hung Wu
Date: 2004/3/1

# Where will data mining research go?

**Data**

Frequent Itemsets,
Association Rules,
Sequential Patterns,
Clusters, Outliers…

**Knowledge**

Text Classification,
Web Usage Prediction,
Feature Selection,
Anomaly Detection, …

**Action**

Microarray Classification,
“In Vivo” Spam Filtering,…

**Profit**

Constraint-based

Applied Data Mining

Action-oriented

Text Mining

Web Mining

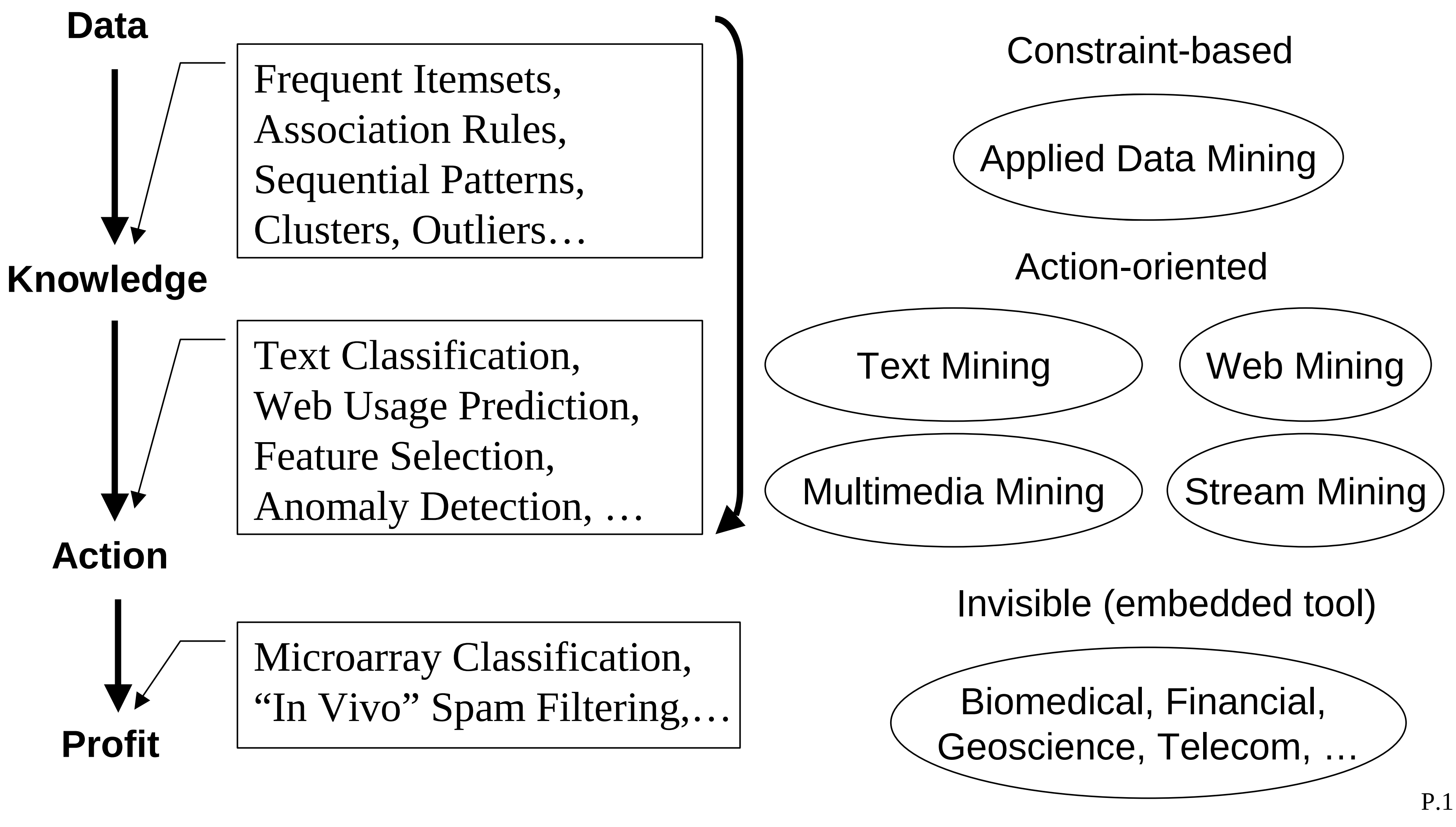
Multimedia Mining

Stream Mining

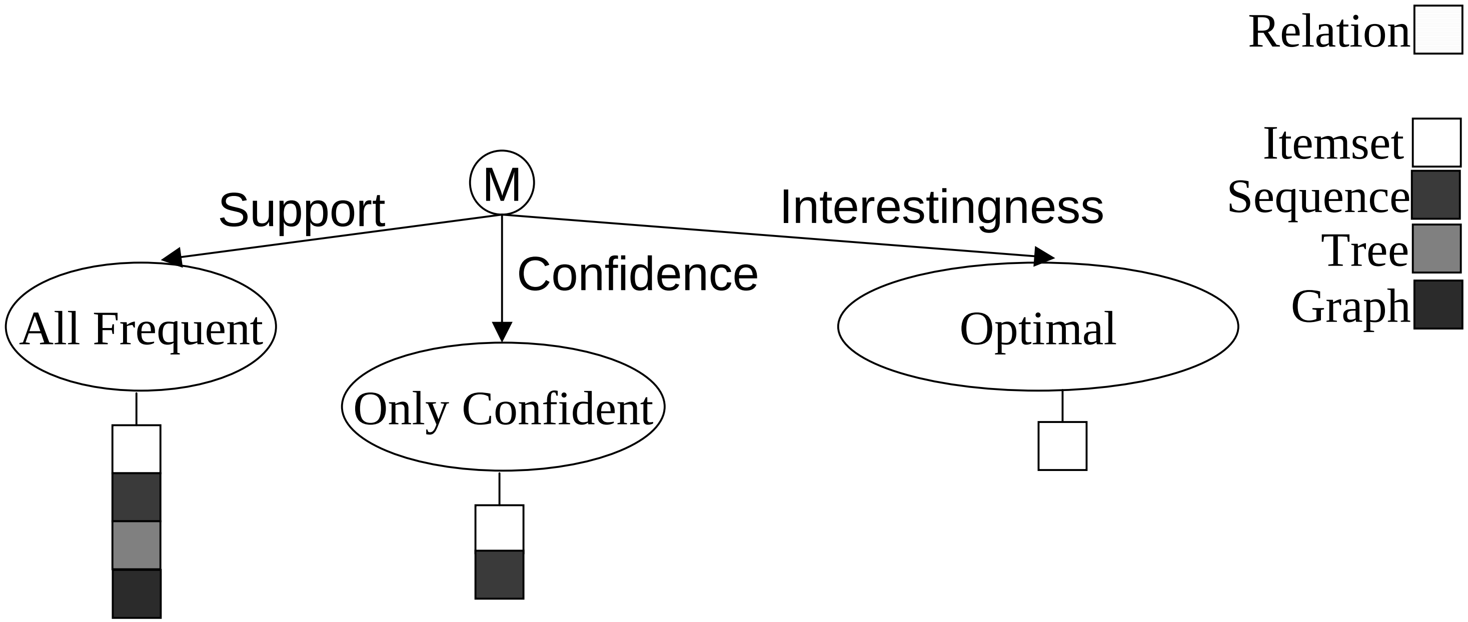
Invisible (embedded tool)

Biomedical, Financial,
Geoscience, Telecom, …

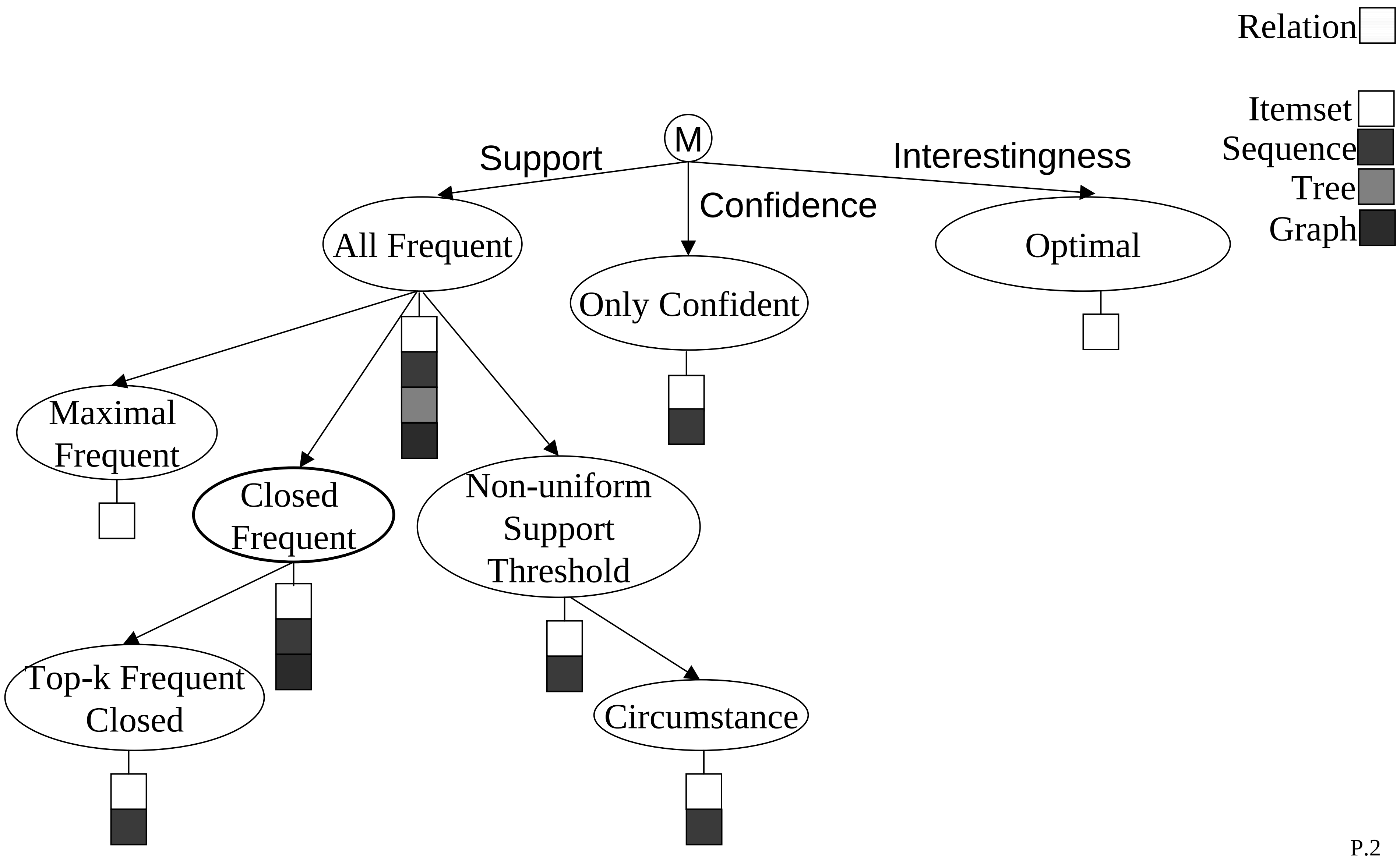
# Where will data mining research go?



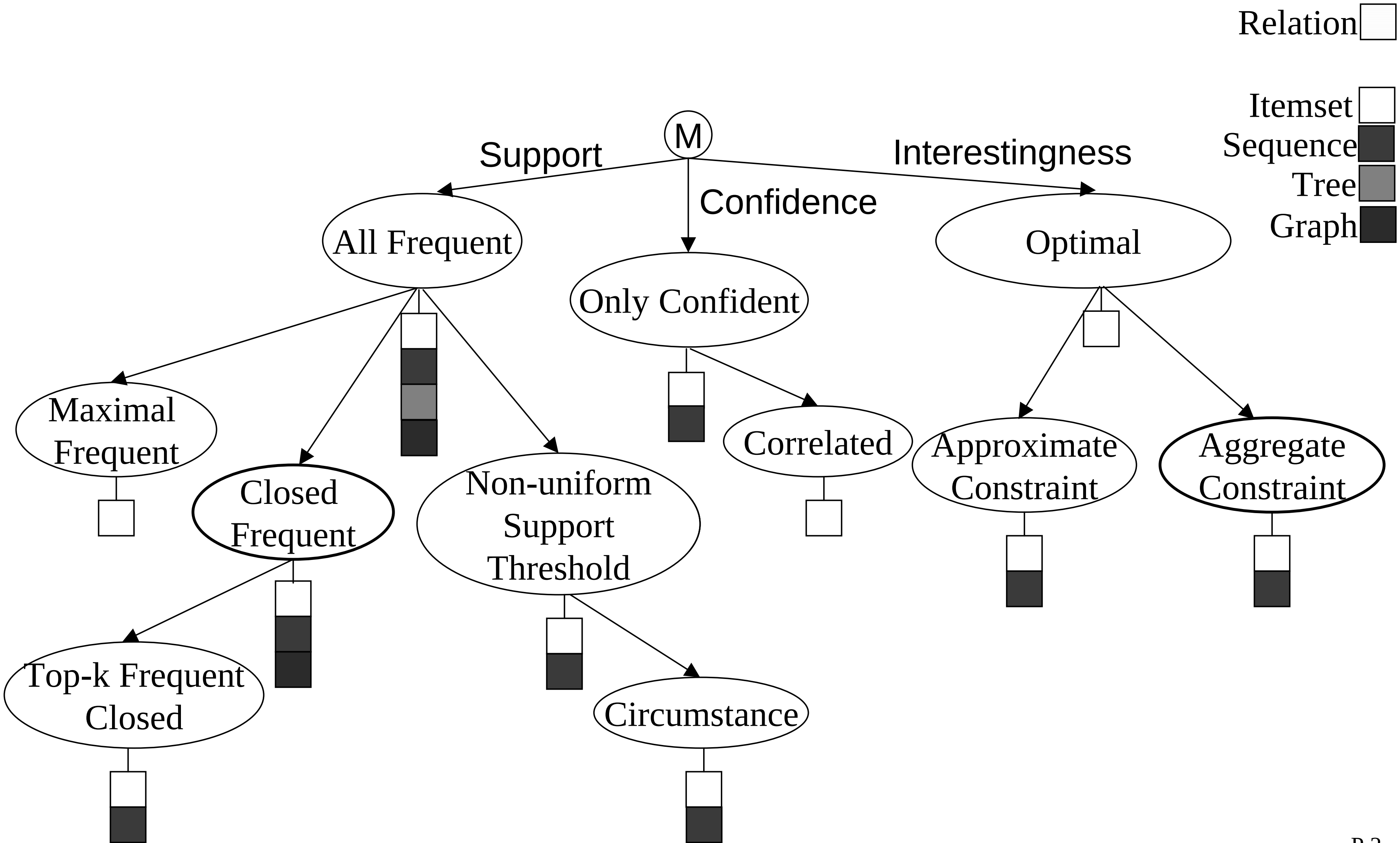
# Which patterns are interesting (applicable)?



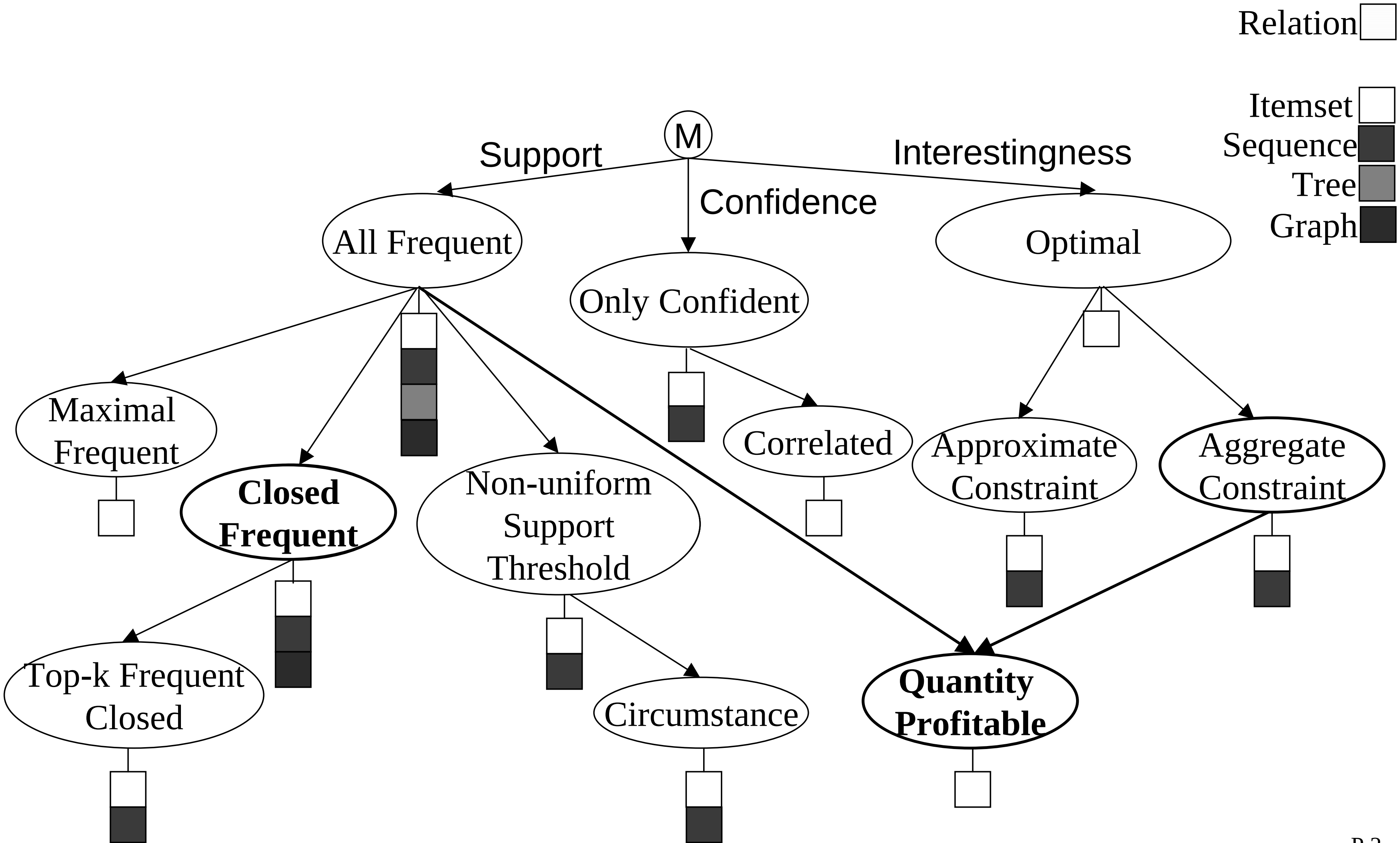
# Which patterns are interesting (applicable)?



# Which patterns are interesting (applicable)?



# Which patterns are interesting (applicable)?



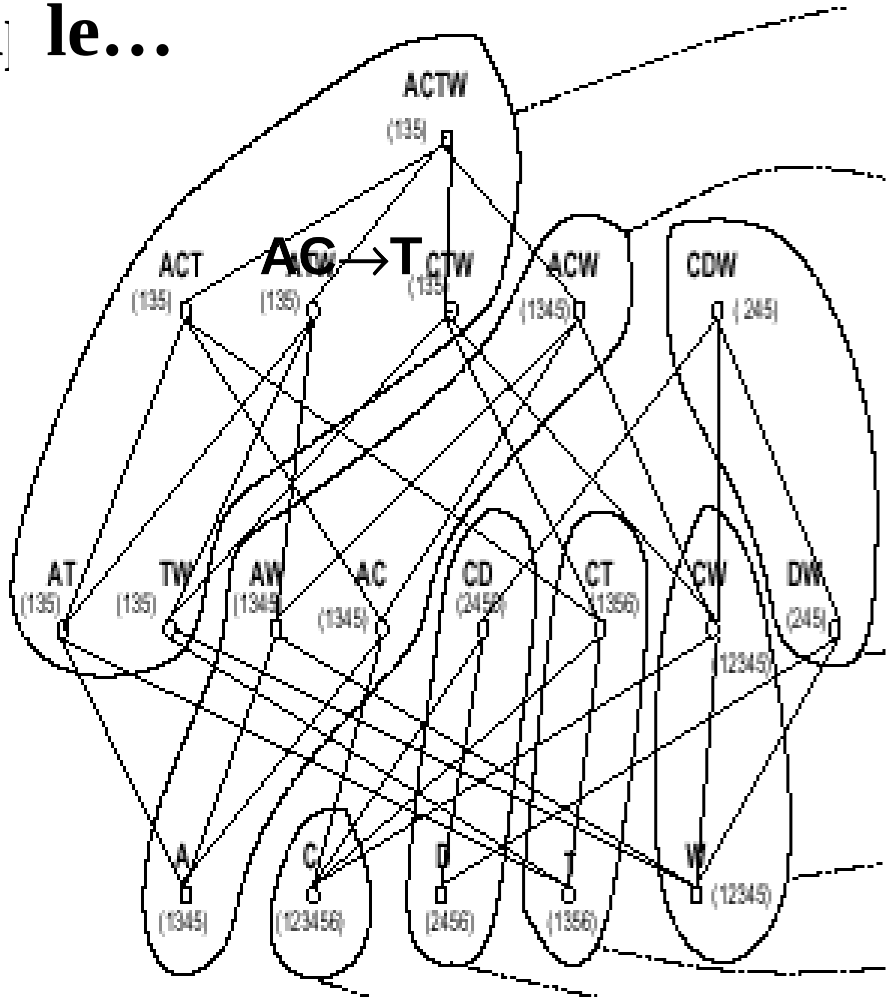
# What is “Closed Frequent”?

- **Take itemset as an example…**
  - F ⊇ Closed F ⊇ Max F

| Transcation | Items |
|---|---|
| 1 | A C T W |
| 2 | C D W |
| 3 | A C T W |
| 4 | A C D W |
| 5 | A C D T W |
| 6 | C D T |

MINIMUM SUPPORT = 50%

| Support | Itemsets |
|---|---|
| 100% (6) | C |
| 83% (5) | W, CW |
| 67% (4) | A, D, T, AC, AW<br>CD, CT, ACW |
| 50% (3) | AT, DW, TW, ACT, ATW<br>CDW, CTW, ACTW |



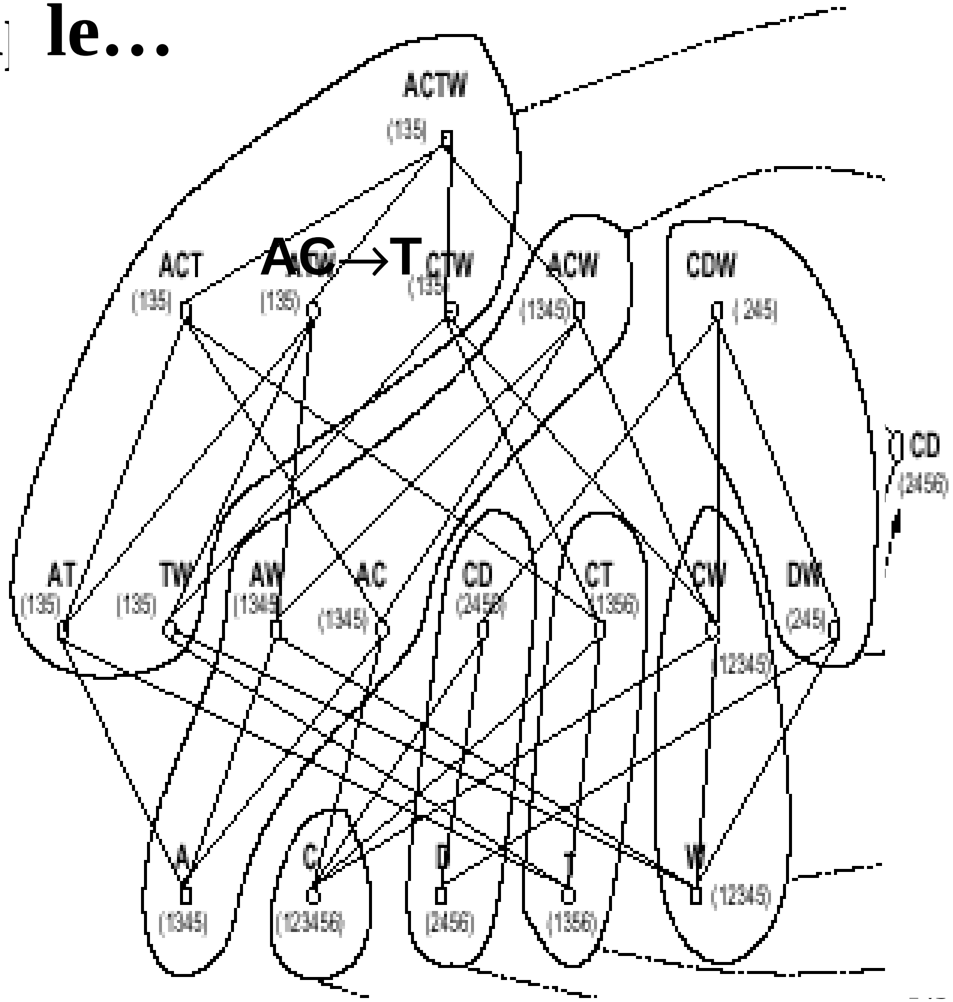
# What is “Closed Frequent”?

- **Take itemset as an example…**
  - F ⊇ Closed F ⊇ Max F

| Transcation | Items |
|---|---|
| 1 | A C T W |
| 2 | C D W |
| 3 | A C T W |
| 4 | A C D W |
| 5 | A C D T W |
| 6 | C D T |

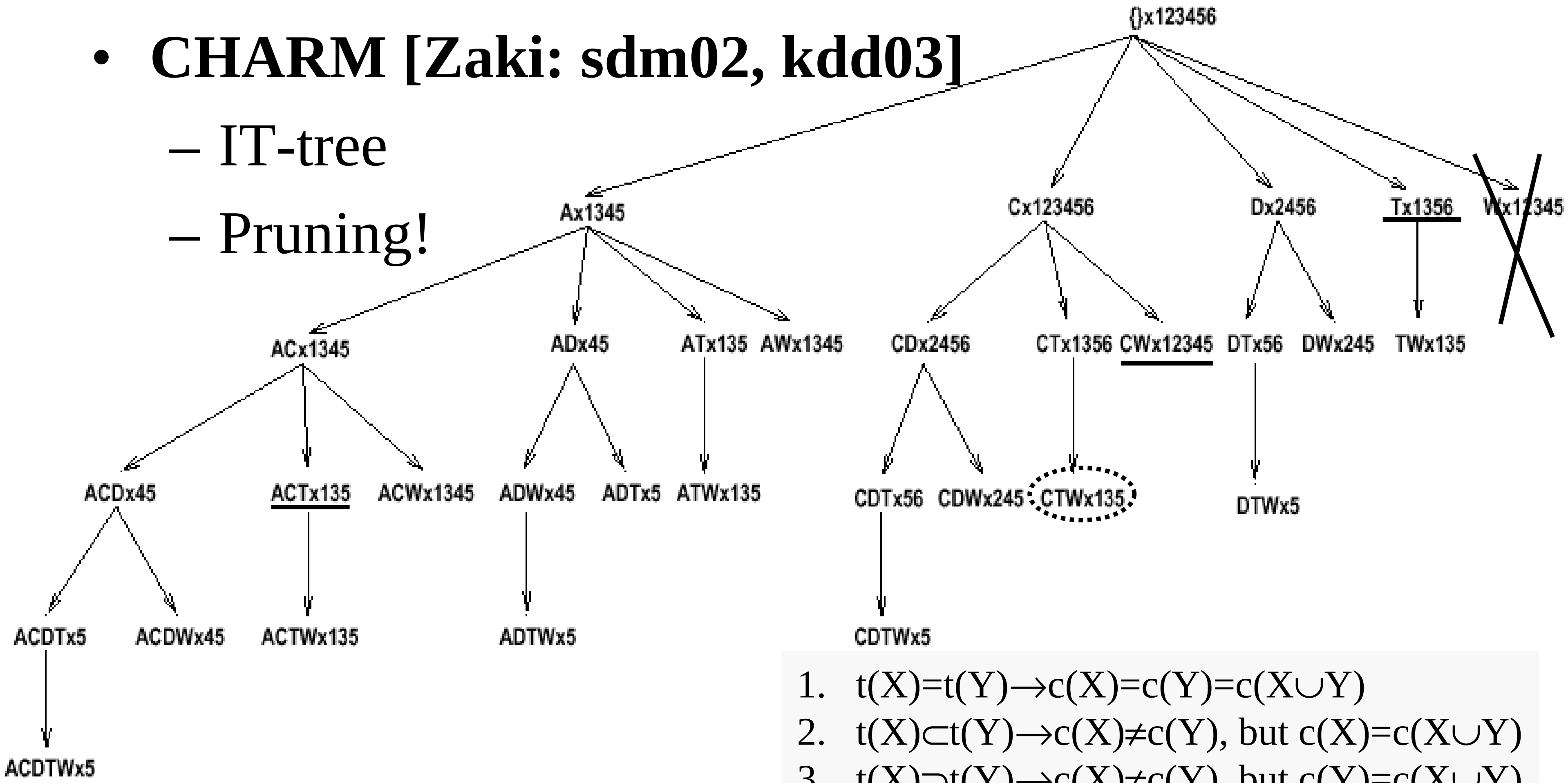
MINIMUM SUPPORT = 50%

| Support | Itemsets |
|---|---|
| 100% (6) | C |
| 83% (5) | W, CW |
| 67% (4) | A, D, T, AC, AW<br>CD, CT, ACW |
| 50% (3) | AT, DW, TW, ACT, ATW<br>CDW, CTW, ACTW |



# How to mine closed frequent itemsets?

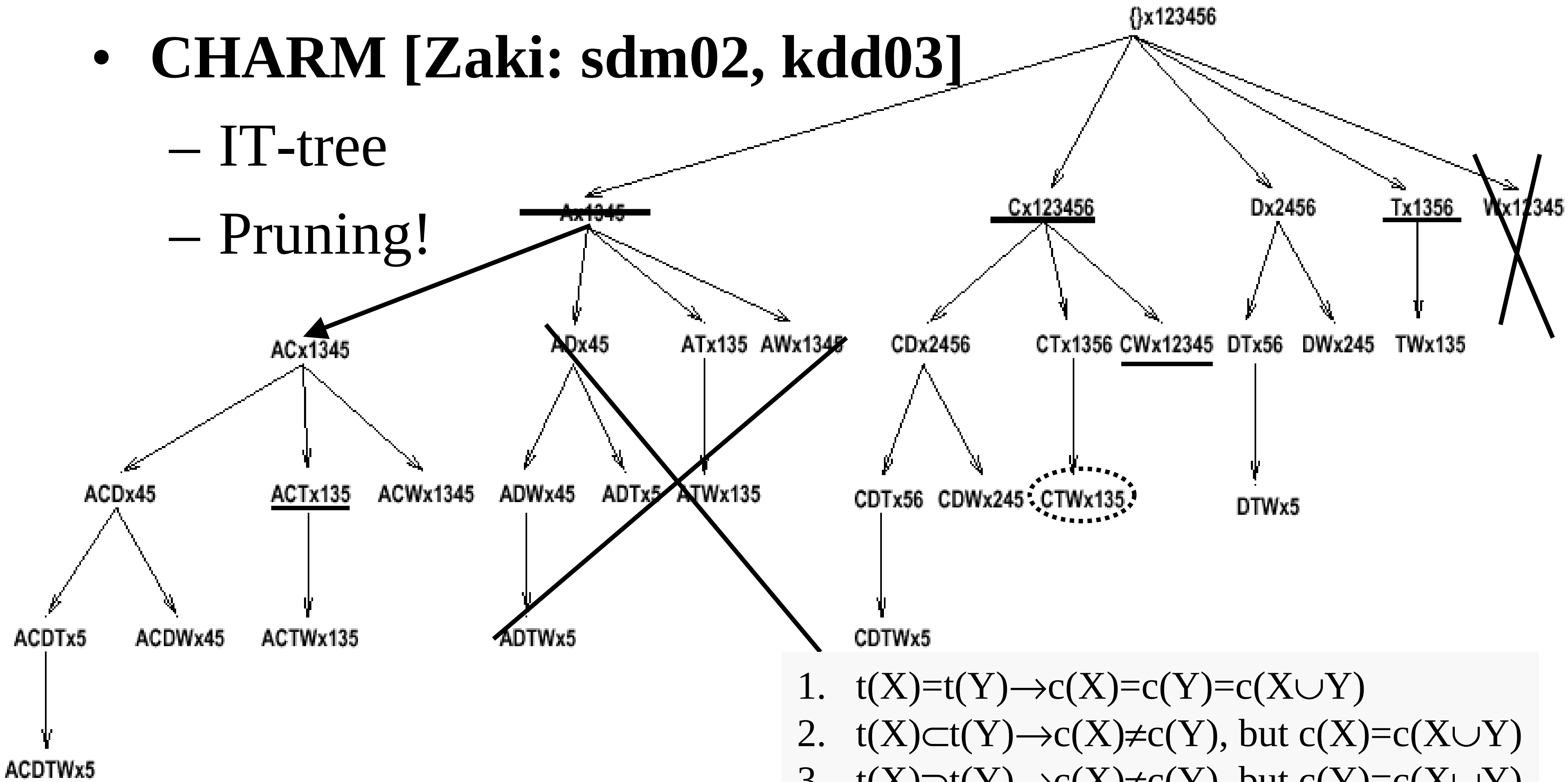
- **CHARM [Zaki: sdm02, kdd03]**
  - IT-tree
  - Pruning!




1. $t(X)=t(Y) \rightarrow c(X)=c(Y)=c(X \cup Y)$
2. $t(X) \subset t(Y) \rightarrow c(X) \neq c(Y)$, but $c(X)=c(X \cup Y)$
3. $t(X) \supset t(Y) \rightarrow c(X) \neq c(Y)$, but $c(Y)=c(X \cup Y)$
4. Otherwise $\rightarrow c(X) \neq c(Y) \neq c(X \cup Y)$

# How to mine closed frequent itemsets?

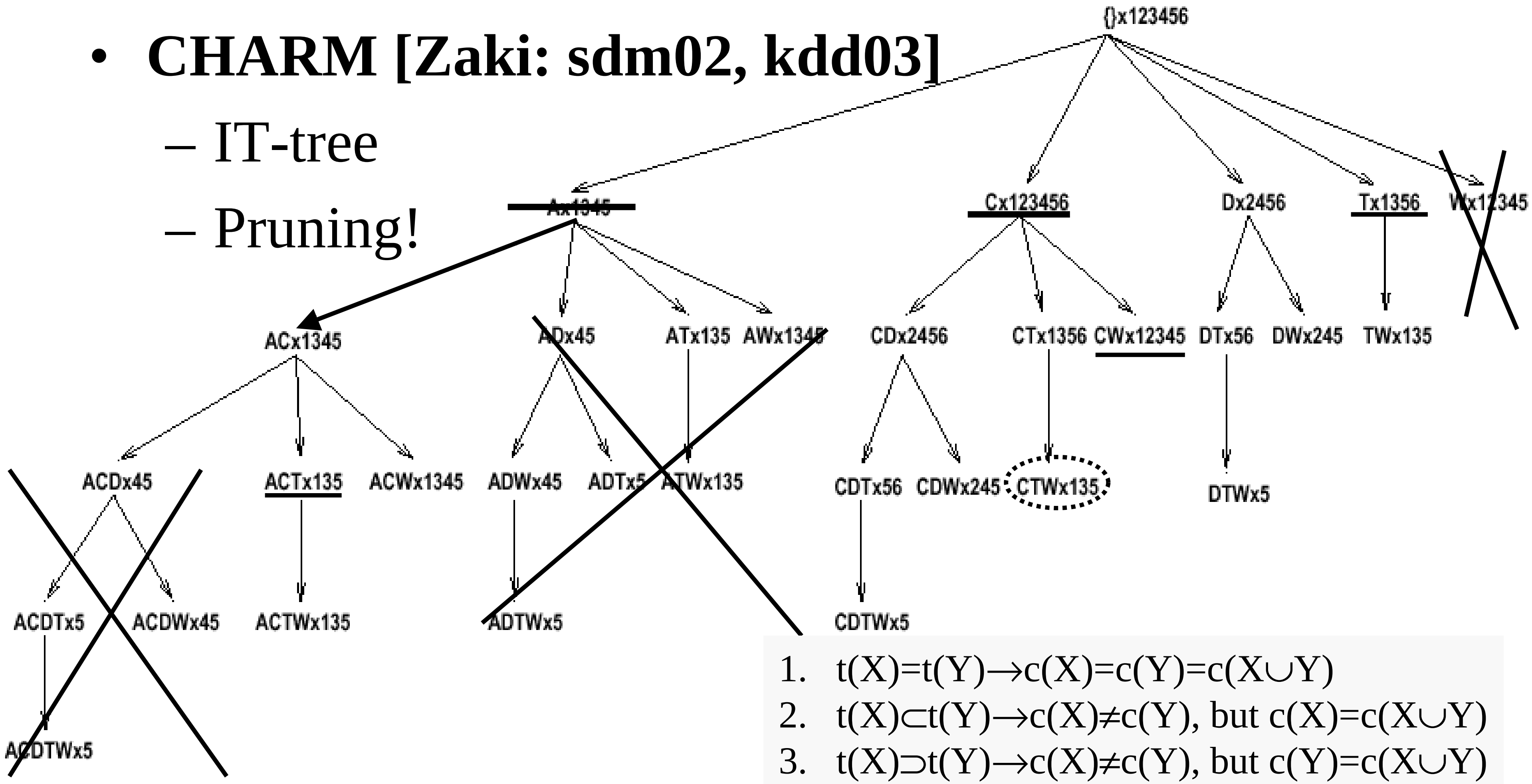
- **CHARM [Zaki: sdm02, kdd03]**
  - IT-tree
  - Pruning!




1. $t(X)=t(Y) \rightarrow c(X)=c(Y)=c(X \cup Y)$
2. $t(X) \subset t(Y) \rightarrow c(X) \neq c(Y)$, but $c(X)=c(X \cup Y)$
3. $t(X) \supset t(Y) \rightarrow c(X) \neq c(Y)$, but $c(Y)=c(X \cup Y)$
4. Otherwise $\rightarrow c(X) \neq c(Y) \neq c(X \cup Y)$

# How to mine closed frequent itemsets?

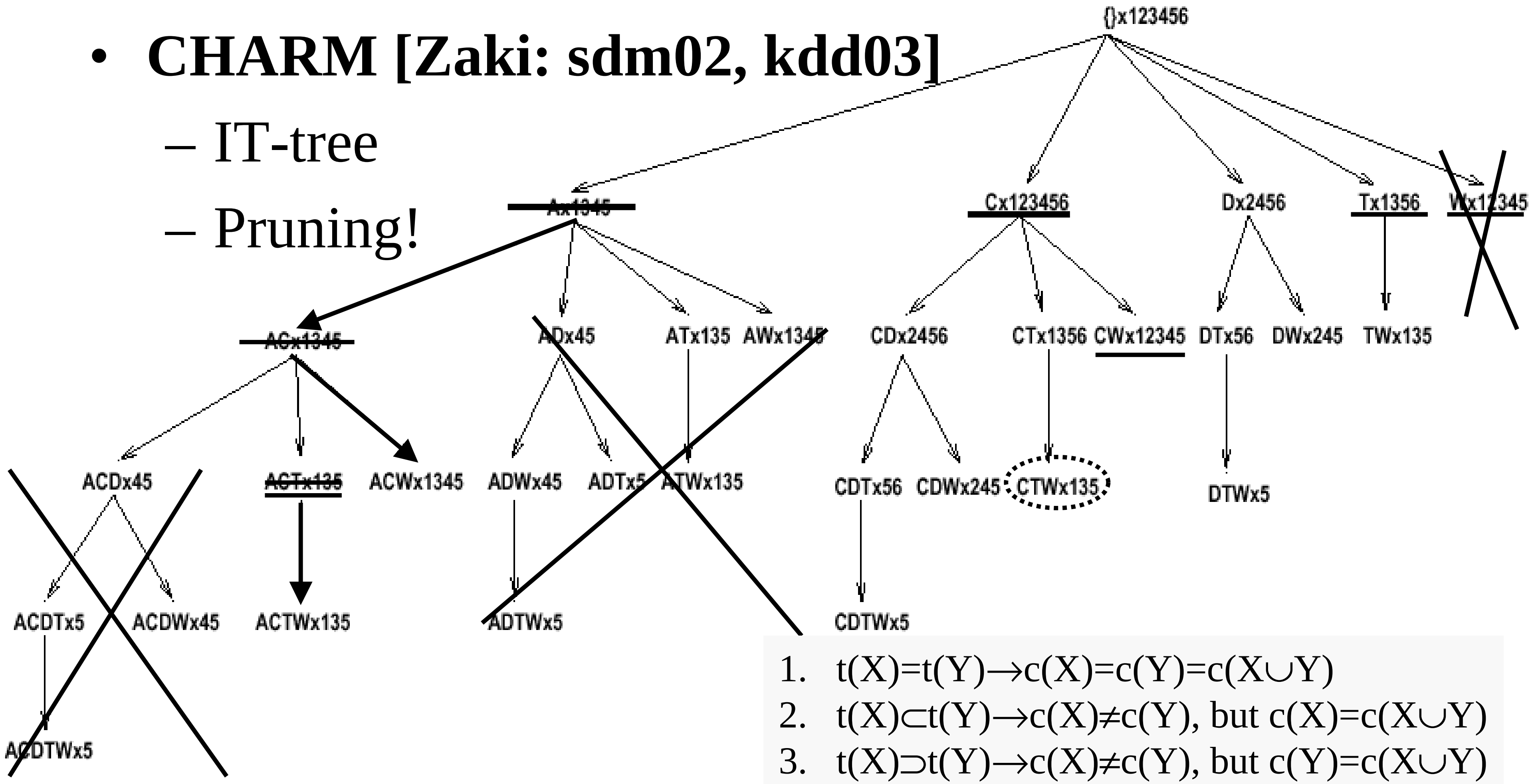
- **CHARM [Zaki: sdm02, kdd03]**
  - IT-tree
  - Pruning!




1. $t(X)=t(Y) \rightarrow c(X)=c(Y)=c(X \cup Y)$
2. $t(X) \subset t(Y) \rightarrow c(X) \neq c(Y)$, but $c(X)=c(X \cup Y)$
3. $t(X) \supset t(Y) \rightarrow c(X) \neq c(Y)$, but $c(Y)=c(X \cup Y)$
4. Otherwise $\rightarrow c(X) \neq c(Y) \neq c(X \cup Y)$

# How to mine closed frequent itemsets?

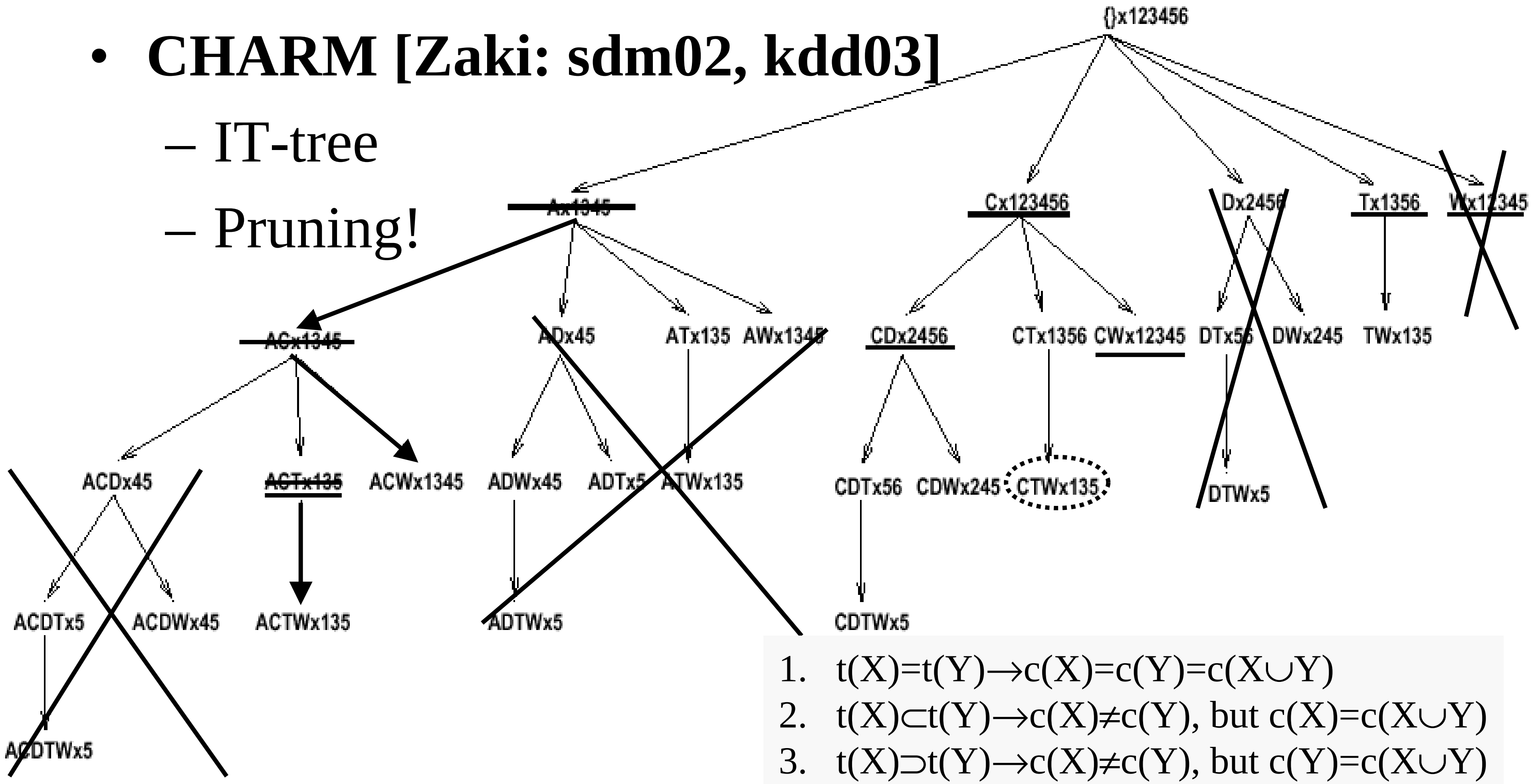
- **CHARM [Zaki: sdm02, kdd03]**
  - IT-tree
  - Pruning!




1. t(X)=t(Y)→c(X)=c(Y)=c(X∪Y)
2. t(X)⊂t(Y)→c(X)≠c(Y), but c(X)=c(X∪Y)
3. t(X)⊃t(Y)→c(X)≠c(Y), but c(Y)=c(X∪Y)
4. Otherwise→c(X)≠c(Y)≠c(X∪Y)

# How to mine closed frequent itemsets?

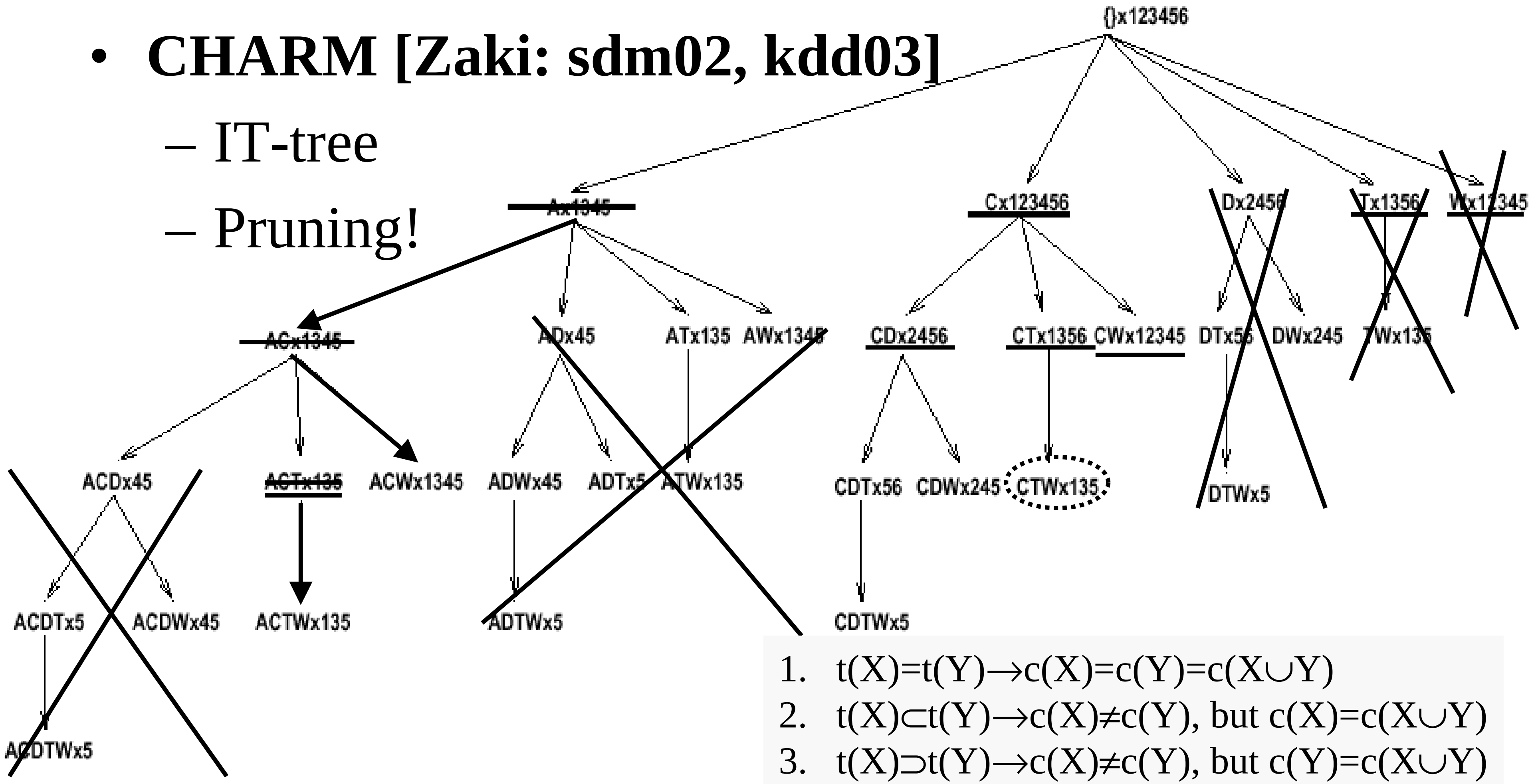
- **CHARM [Zaki: sdm02, kdd03]**
  - IT-tree
  - Pruning!




1. t(X)=t(Y)→c(X)=c(Y)=c(X∪Y)
2. t(X)⊂t(Y)→c(X)≠c(Y), but c(X)=c(X∪Y)
3. t(X)⊃t(Y)→c(X)≠c(Y), but c(Y)=c(X∪Y)
4. Otherwise→c(X)≠c(Y)≠c(X∪Y)

# How to mine closed frequent itemsets?

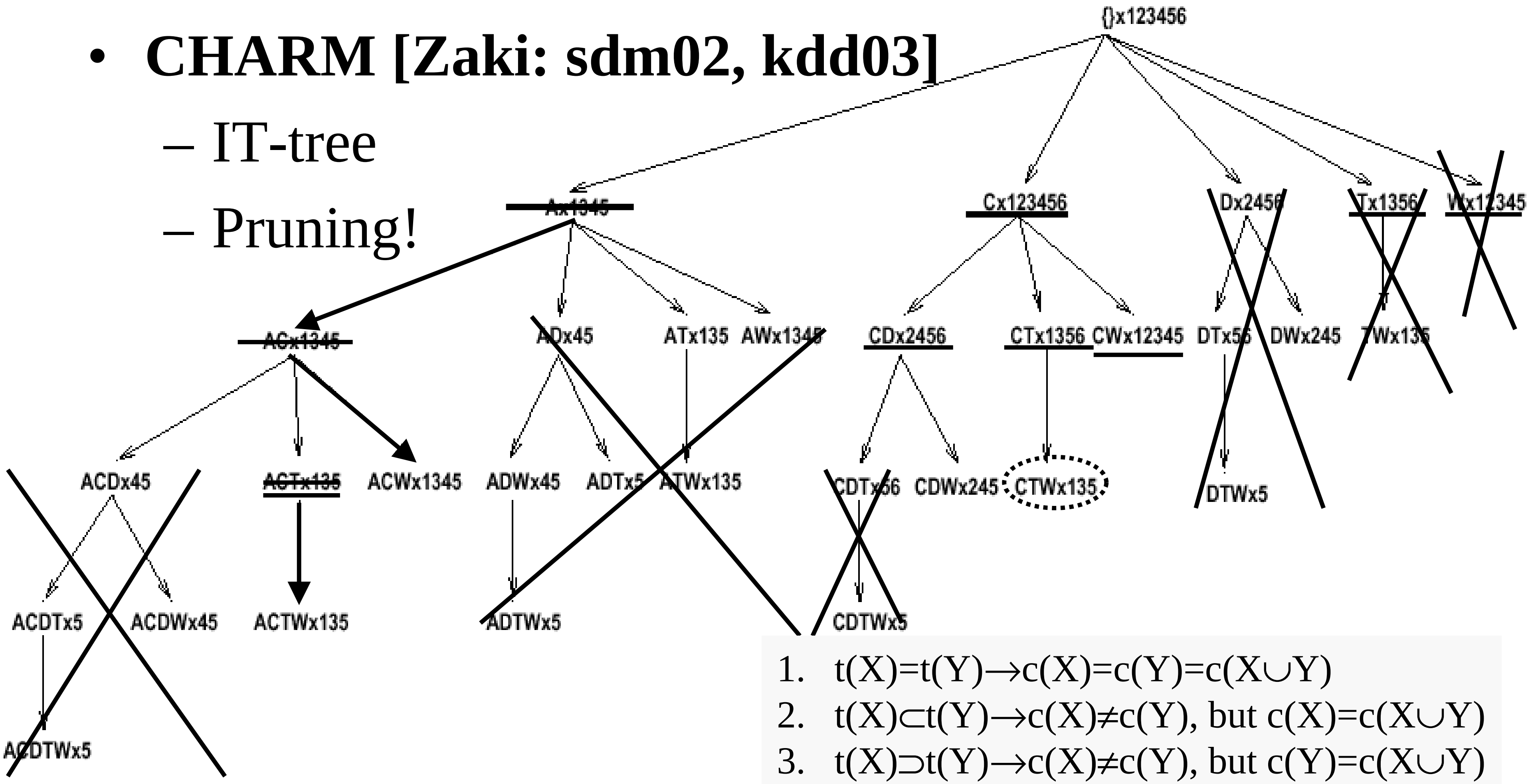
- **CHARM [Zaki: sdm02, kdd03]**
  - IT-tree
  - Pruning!




1. $t(X)=t(Y) \rightarrow c(X)=c(Y)=c(X \cup Y)$
2. $t(X) \subset t(Y) \rightarrow c(X) \neq c(Y)$, but $c(X)=c(X \cup Y)$
3. $t(X) \supset t(Y) \rightarrow c(X) \neq c(Y)$, but $c(Y)=c(X \cup Y)$
4. Otherwise $\rightarrow c(X) \neq c(Y) \neq c(X \cup Y)$

# How to mine closed frequent itemsets?

- **CHARM [Zaki: sdm02, kdd03]**
  - IT-tree
  - Pruning!




1. $t(X)=t(Y) \rightarrow c(X)=c(Y)=c(X \cup Y)$
2. $t(X) \subset t(Y) \rightarrow c(X) \neq c(Y)$, but $c(X)=c(X \cup Y)$
3. $t(X) \supset t(Y) \rightarrow c(X) \neq c(Y)$, but $c(Y)=c(X \cup Y)$
4. Otherwise $\rightarrow c(X) \neq c(Y) \neq c(X \cup Y)$

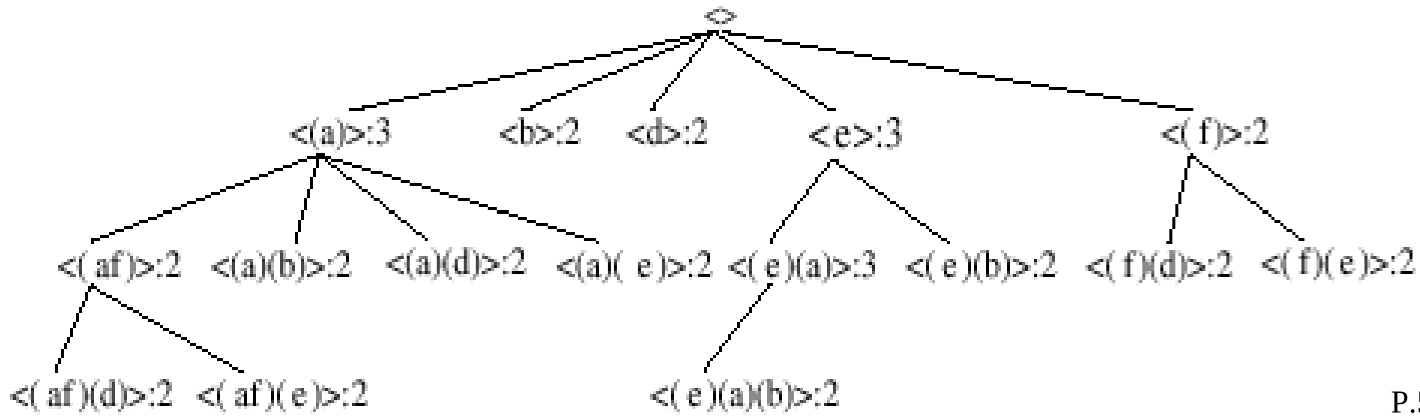
# What is a Closed Frequent Sequence?

- **CloSpan [Yang&Han: sdm03]**
  - Lexicographic sequence tree
  - Pruning!

| Seq ID. | Sequence |
|---|---|
| 0 | $\langle (af)(d)(e)(a) \rangle$ |
| 1 | $\langle (e)(a)(b) \rangle$ |
| 2 | $\langle (e)(abf)(bde) \rangle$ |

*Given two sequences,* $s \sqsubseteq s'$ *and also* $\mathcal{I}(D_s) = \mathcal{I}(D_{s'})$,
*then* $\forall \gamma$, $support(s \diamond \gamma) = support(s' \diamond \gamma)$.

- Early Termination by Equivalence



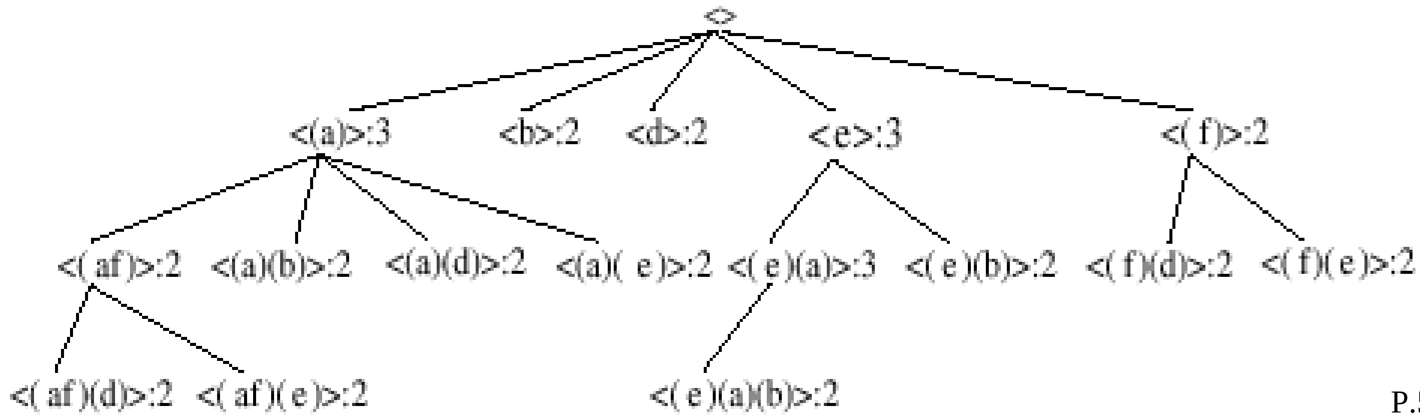
# What is a Closed Frequent Sequence?

- **CloSpan [Yang&Han: sdm03]**
  - Lexicographic sequence tree
  - Pruning!

| Seq ID. | Sequence |
|---|---|
| 0 | $\langle (af)(d)\underline{(e)(a)} \rangle$ |
| 1 | $\langle \underline{(e)(a)}(b) \rangle$ |
| 2 | $\langle \underline{(e)(a}bf)(bde) \rangle$ |

*Given two sequences,* $s \sqsubseteq s'$ *and also* $\mathcal{I}(D_s) = \mathcal{I}(D_{s'})$,
*then* $\forall \gamma$, $support(s \diamond \gamma) = support(s' \diamond \gamma)$.

- Early Termination by Equivalence



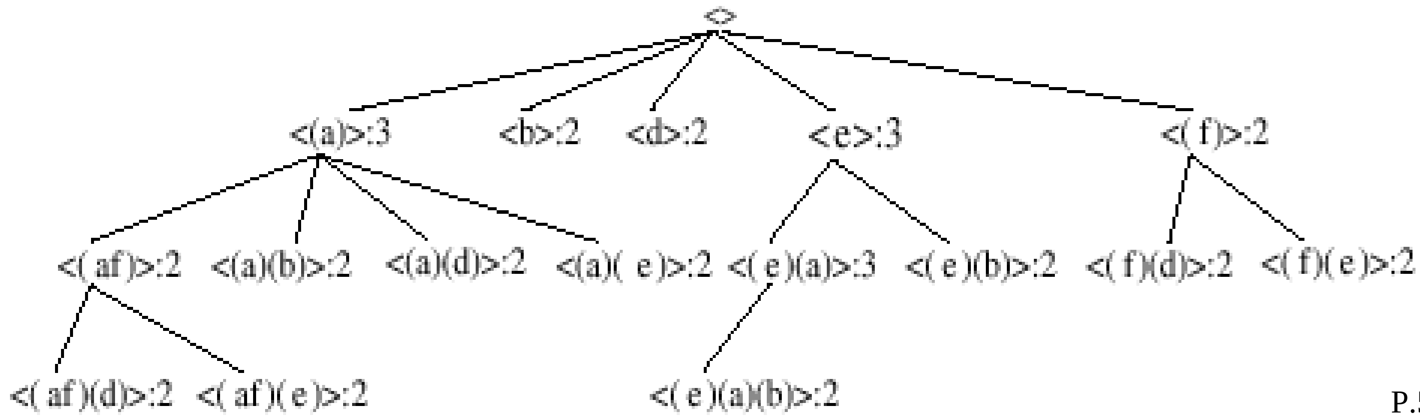
# What is a Closed Frequent Sequence?

- **CloSpan [Yang&Han: sdm03]**
  - Lexicographic sequence tree
  - Pruning!

| Seq ID. | Sequence |
|---|---|
| 0 | $\langle(\underline{af})(d)(e)(a)\rangle$ |
| 1 | $\langle(e)(a)(b)\rangle$ |
| 2 | $\langle(e)(\underline{abf})(bde)\rangle$ |

*Given two sequences,* $s \sqsubseteq s'$ *and also* $\mathcal{I}(D_s) = \mathcal{I}(D_{s'})$, *then* $\forall\gamma, support(s \diamond \gamma) = support(s' \diamond \gamma)$.

- Early Termination by Equivalence



# What is a Closed Frequent Sequence?

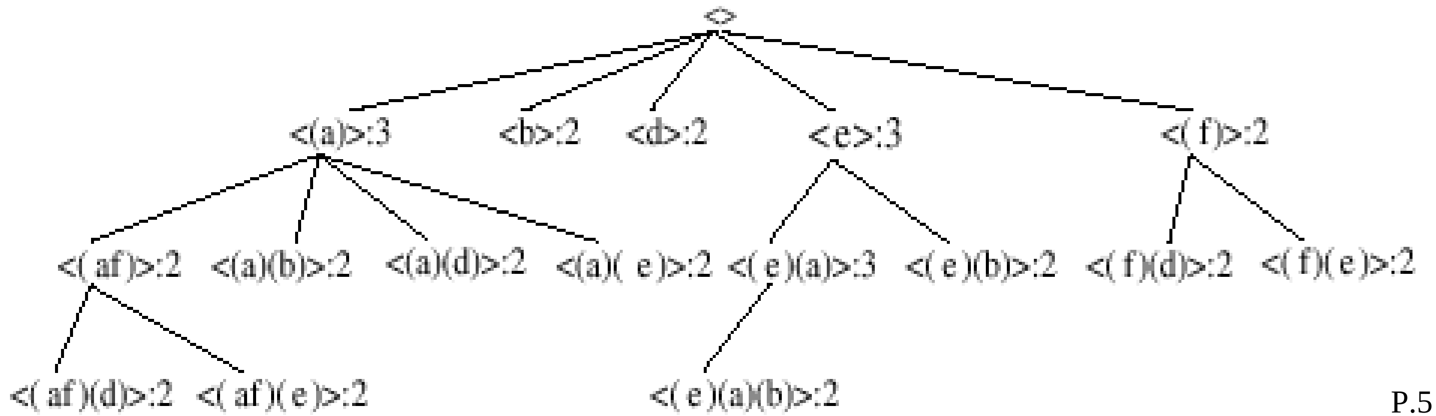
- **CloSpan [Yang&Han: sdm03]**
  - Lexicographic sequence tree
  - Pruning!

| Seq ID. | Sequence |
|---|---|
| 0 | $\langle (af)(d)(e)(a) \rangle$ |
| 1 | $\langle (e)(a)(b) \rangle$ |
| 2 | $\langle (e)(abf)(bde) \rangle$ |

*Given two sequences,* $s \sqsubseteq s'$ *and also* $\mathcal{I}(D_s) = \mathcal{I}(D_{s'})$,
*then* $\forall \gamma, support(s \diamond \gamma) = support(s' \diamond \gamma)$.

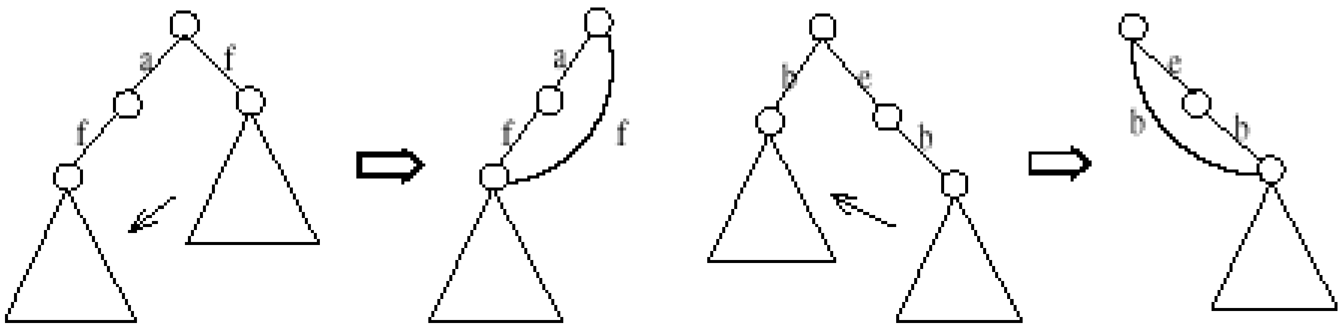
- Early Termination by Equivalence

$D_s = D_{s'} \Leftrightarrow \mathcal{I}(I$



# How to mine closed frequent sequences?

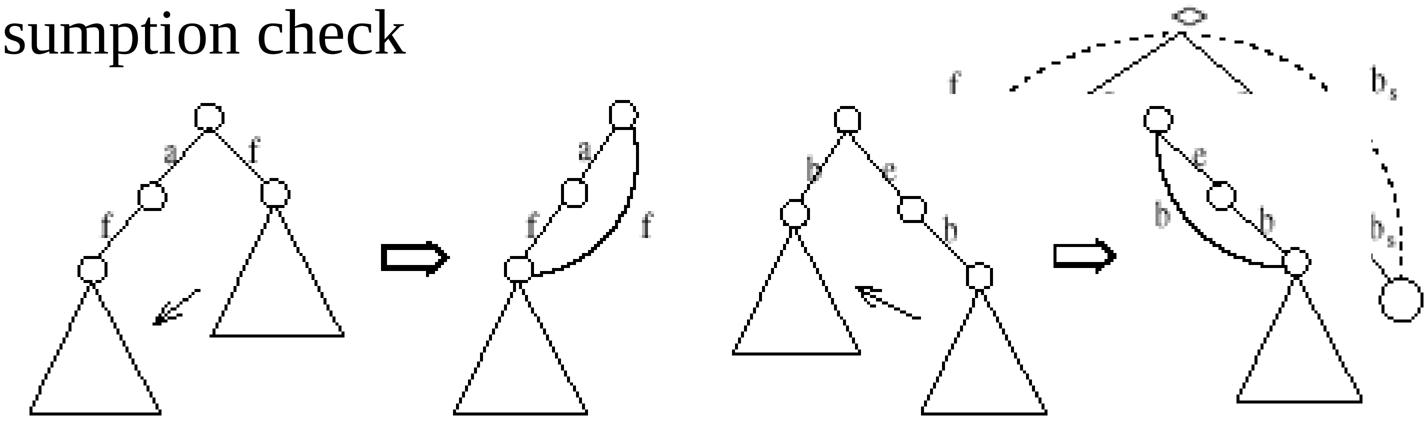
- **Stage 1: Generate candidate sequences**
  - PrefixSpan + Pruning! $\Rightarrow$ Prefix sequence lattice
- **Stage 2: Eliminate non-close sequences**
  - Hashing: size, s-id sum
    - Support equality
    - Subsumption check




(a) backward sub-pattern (b) backward super-pattern

# How to mine closed frequent sequences?

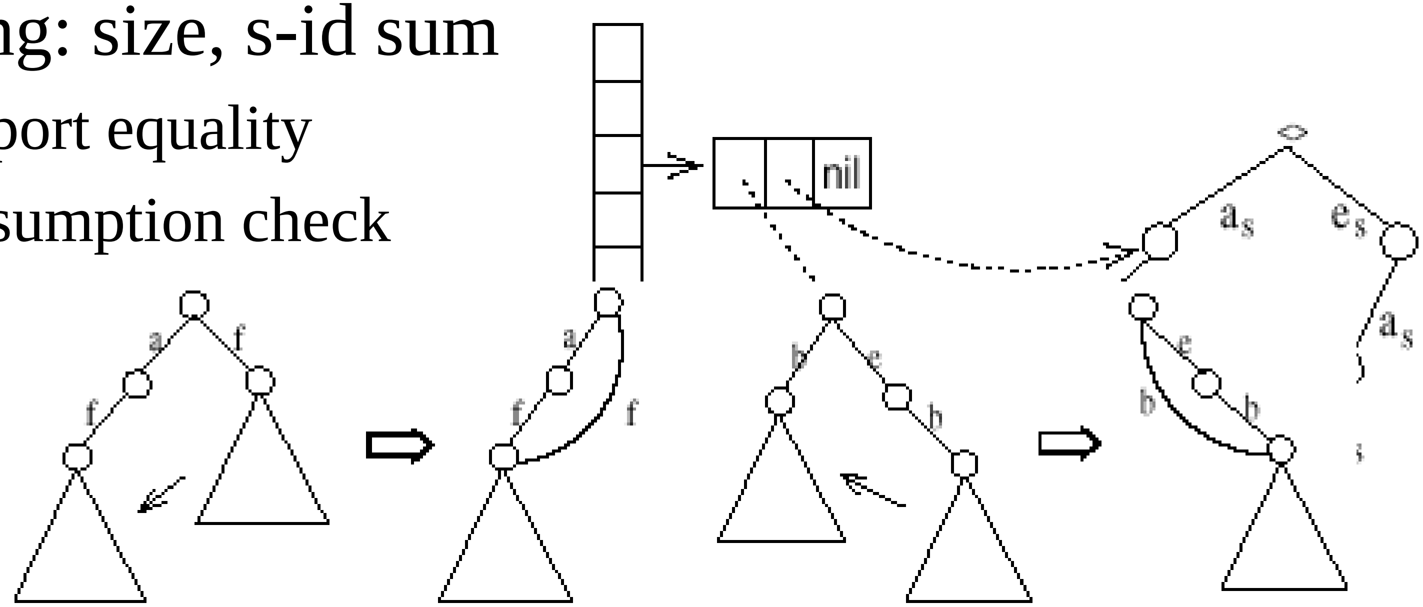
- **Stage 1: Generate candidate sequences**
  - PrefixSpan + Pruning! ⇒ Prefix sequence lattice
- **Stage 2: Eliminate non-close sequences**
  - Hashing: size, s-id sum
    - Support equality
    - Subsumption check



(a) backward sub-pattern (b) backward super-pattern

# How to mine closed frequent sequences?

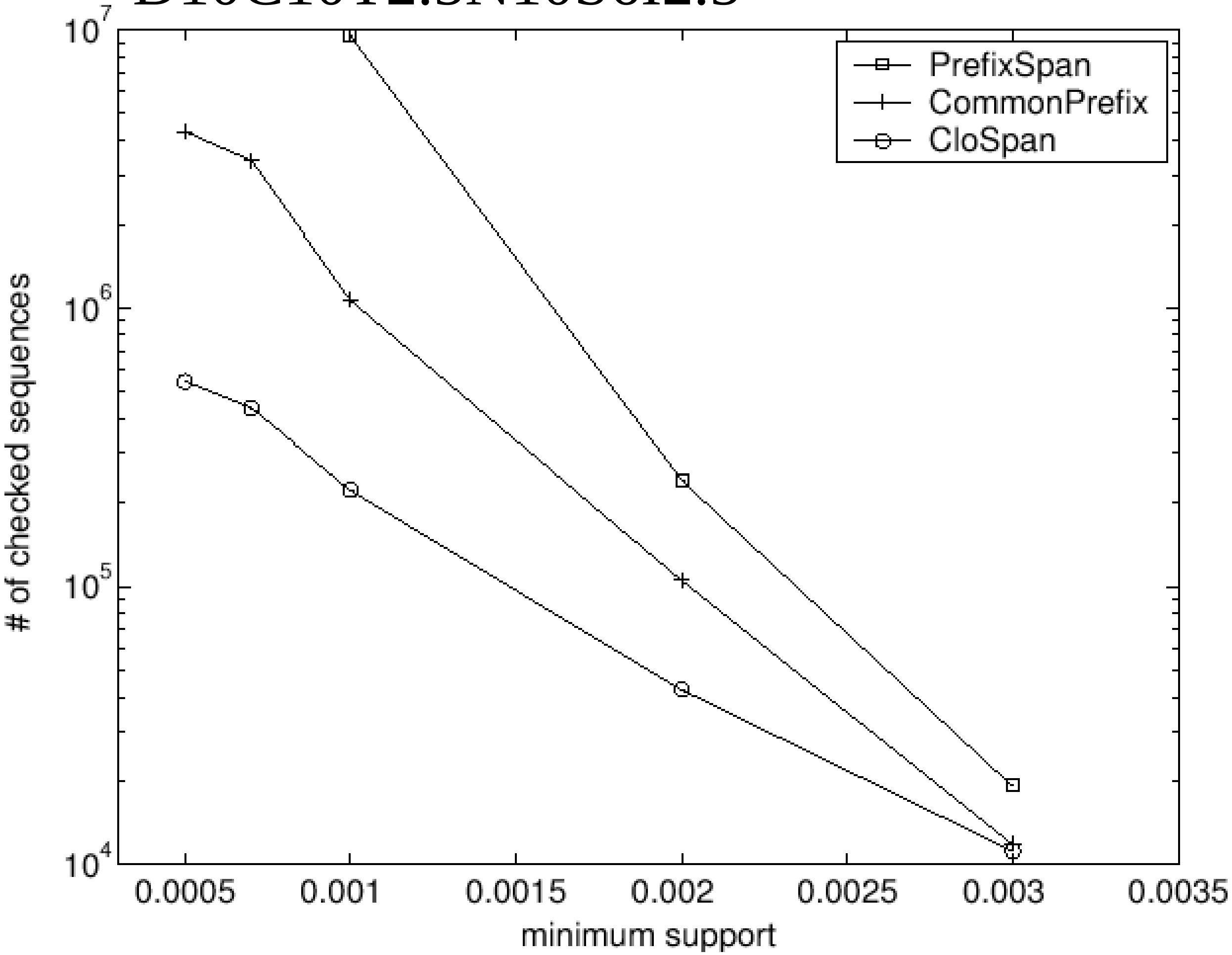
- **Stage 1: Generate candidate sequences**
  - PrefixSpan + Pruning! ⇒ Prefix sequence lattice
- **Stage 2: Eliminate non-close sequences**
  - Hashing: size, s-id sum
    - Support equality
    - Subsumption check




(a) backward sub-pattern (b) backward super-pattern

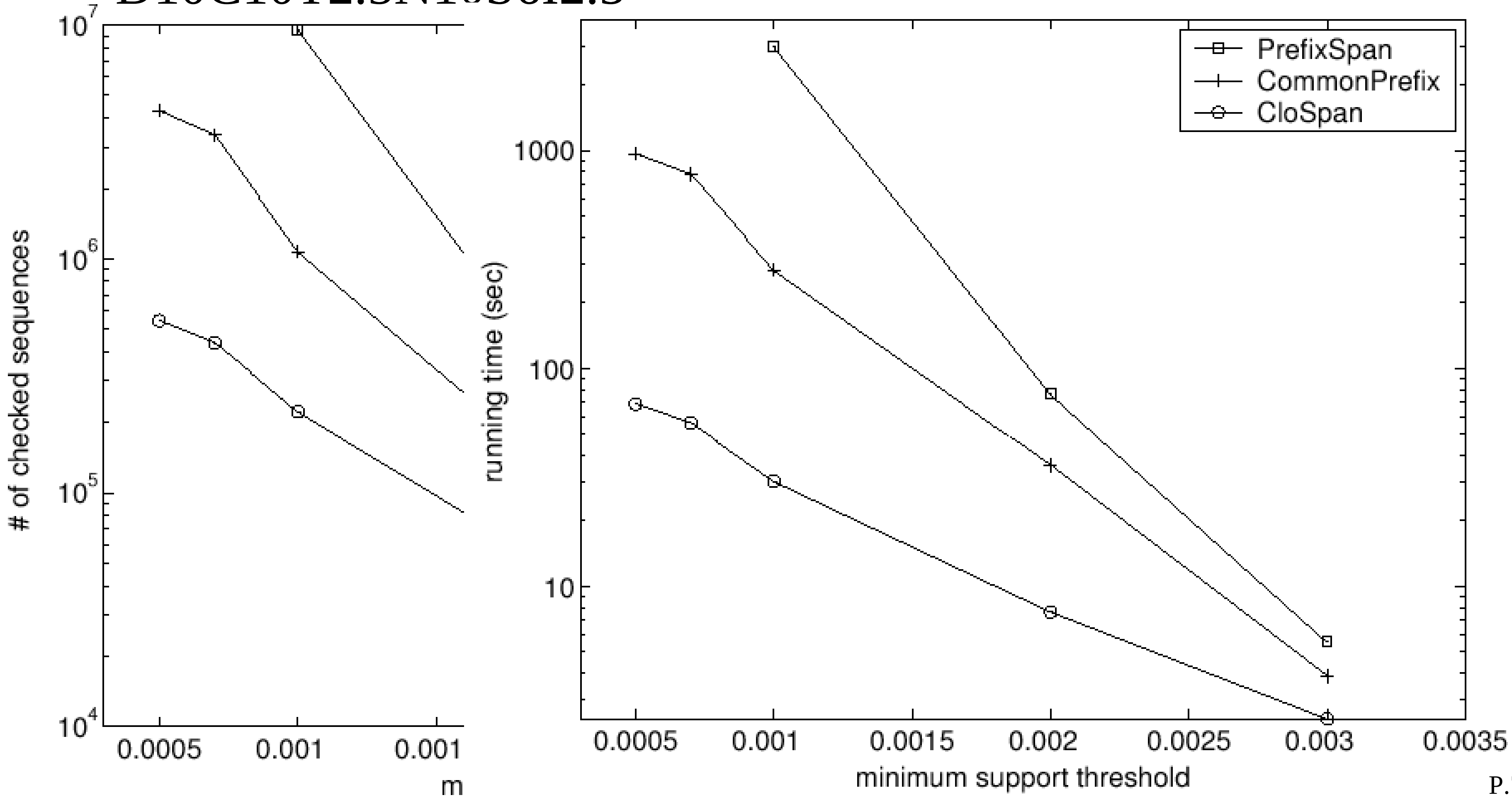
# How well does CloSpan perform?

- D10C10T2.5N10S6I2.5



# How well does CloSpan perform?

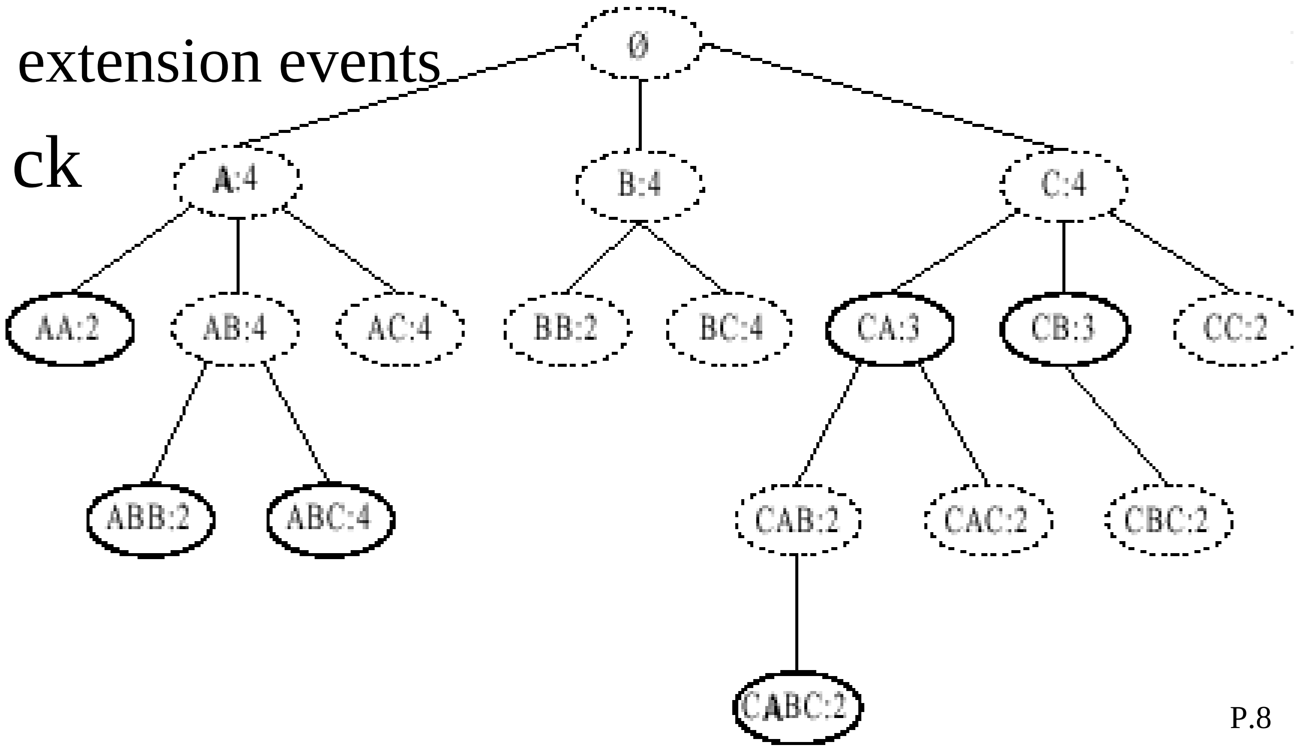
- D10C10T2.5N10S6I2.5



# Can we mine closed frequent sequences without candidate maintenance?

| Sequence identifier | Sequence |
|---|---|
| 1 | C A A B C |
| 2 | A B C B |
| 3 | C A B C |
| 4 | A B B C A |

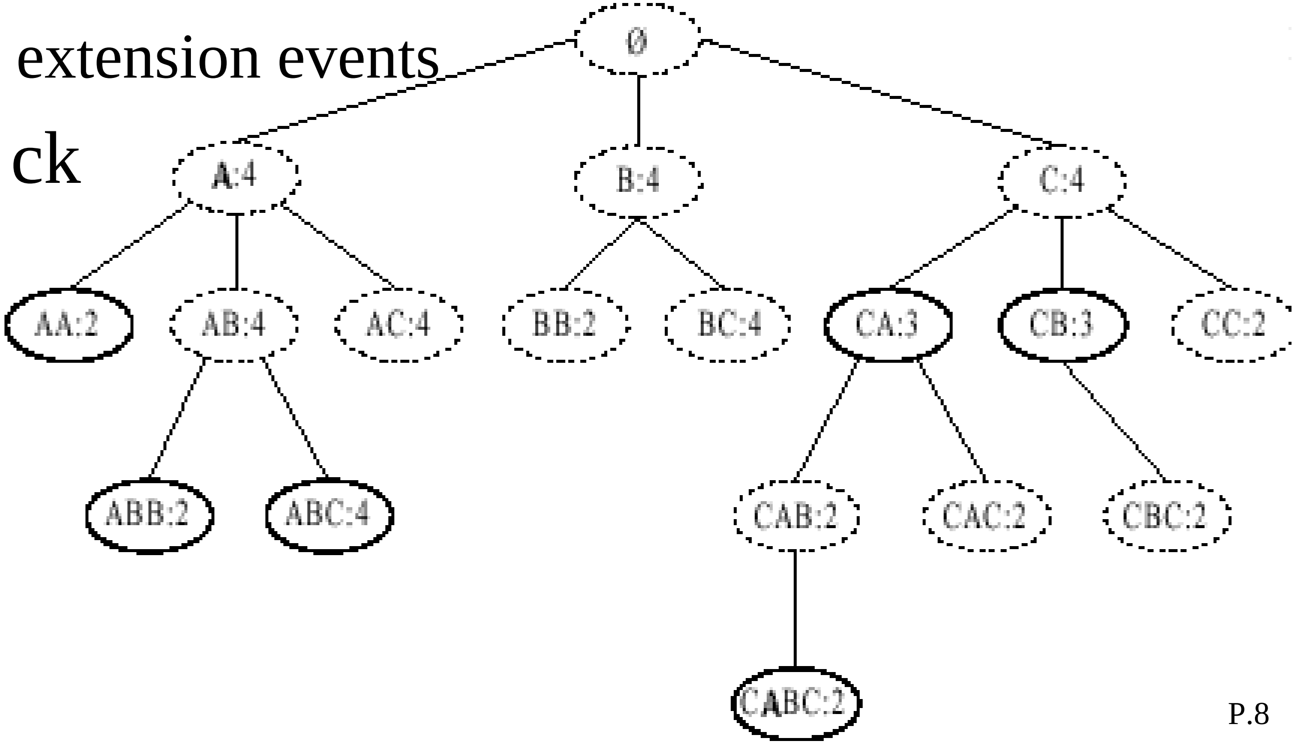
- **BIDE**
  - BI-Directional Extension
    - Forward extension events
    - Backward extension events
  - Closure check
    - No FE
    - No BE
  - Pruning!
    - BackScan



# Can we mine closed frequent sequences without candidate maintenance?

- **BIDE**
  - BI-Directional Extension
    - Forward extension events
    - Backward extension events
  - Closure check
    - No FE
    - No BE
  - Pruning!
    - BackScan

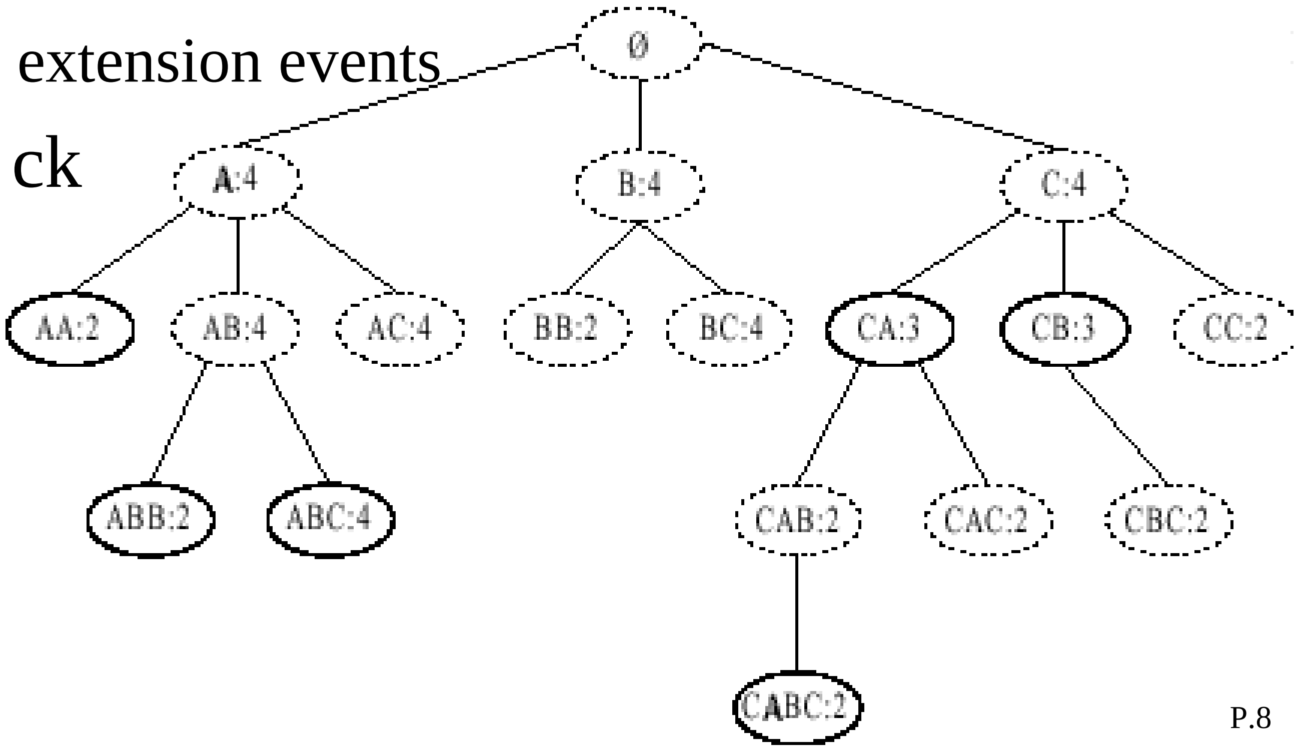
| Sequence identifier | Sequence |
|---|---|
| 1 | C A A B C |
| 2 | A B C B |
| 3 | C A B C |
| 4 | A B B C A |



# Can we mine closed frequent sequences without candidate maintenance?

- **BIDE**
  - BI-Directional Extension
    - Forward extension events
    - Backward extension events
  - Closure check
    - No FE
    - No BE
  - Pruning!
    - BackScan

| Sequence identifier | Sequence |
|---|---|
| 1 | C A A B C |
| 2 | A B C B |
| 3 | C A B C |
| 4 | A B B C A |



# Where to find forward/backward extensions?

$S_1$: $MP_{11}$, $MP_{12}$, $MP_{13}$
$S_2$: $MP_{21}$, $MP_{22}$, $MP_{23}$
…
$S_n$: $MP_{n1}$, $MP_{n2}$, $MP_{n3}$

# Where to find forward/backward extensions?

- **FE={locally frequent items with full supports}**

$S_1$: $MP_{11}$, $MP_{12}$, $MP_{13}$
$S_2$: $MP_{21}$, $MP_{22}$, $MP_{23}$
…
$S_n$: $MP_{n1}$, $MP_{n2}$, $MP_{n3}$

# Where to find forward/backward extensions?

- **FE={locally frequent items with full supports}**
- **For prefix <u>ABC</u>, given $C_1A_1A_2BC_2DA_3C_3E$**
  - Last instance = $C_1A_1A_2BC_2DA_3C_3$
  - $LL_i$: the i-th last-in-last appearance
    - $LL_1 = A_2$, $LL_2 = B$, $LL_3 = C_3$
  - $MP_i$: the i-th maximum period
    - $MP_1 = C_1A_1$, $MP_2 = A_2$, $MP_3 = C_2DA_3$

$S_1$: $MP_{11}$, $MP_{12}$, $MP_{13}$
$S_2$: $MP_{21}$, $MP_{22}$, $MP_{23}$
…
$S_n$: $MP_{n1}$, $MP_{n2}$, $MP_{n3}$

# Where to find forward/backward extensions?

- **FE={locally frequent items with full supports}**
- **For prefix <u>ABC</u>, given $C_1A_1A_2BC_2DA_3C_3E$**
  - Last instance = $C_1A_1A_2BC_2DA_3C_3$
  - $LL_i$: the i-th last-in-last appearance
    - $LL_1 = A_2$, $LL_2 = B$, $LL_3 = C_3$
  - $MP_i$: the i-th maximum period
    - $MP_1 = C_1A_1$, $MP_2 = A_2$, $MP_3 = C_2DA_3$
- **BE={items appearing in each of $MP_i$, $\exists i$)}**
  - Scan backward each of $MP_i$, $\forall i \Rightarrow$ ScanSkip

$S_1$: $MP_{11}$, $MP_{12}$, $MP_{13}$
$S_2$: $MP_{21}$, $MP_{22}$, $MP_{23}$
…
$S_n$: $MP_{n1}$, $MP_{n2}$, $MP_{n3}$

# How does BIDE improve the mining efficiency?

# How does BIDE improve the mining efficiency?

- **BackScan: <u>ABC</u>, $C_1A_1A_2BC_2DA_3C_3E$**
  - $LF_i$: the i-th last-in-first appearance
    - $LF_1 = A_2$, $LF_2 = B$, $LF_3 = C_2$
  - $SMP_i$: the i-th semi-maximum period
    - $SMP_1 = C_1A_1$, $SMP_2 = A_2$, $SMP_3 = \varnothing$

# How does BIDE improve the mining efficiency?

- **BackScan: <u>ABC</u>, $C_1A_1A_2BC_2DA_3C_3E$**
  - $LF_i$: the i-th last-in-first appearance
    - $LF_1 = A_2$, $LF_2 = B$, $LF_3 = C_2$
  - $SMP_i$: the i-th semi-maximum period
    - $SMP_1 = C_1A_1$, $SMP_2 = A_2$, $SMP_3 = \varnothing$
- **$\exists$e $\exists$i, e appears in each of $SMP_i$**
  - Stop projection!

# How does BIDE improve the mining efficiency?

**BIDE (*SDB*, *min_sup*, *FCS*)**

**Input**: an input sequence database *SDB*, a minimum support threshold *min_sup*

**Output**: the complete set of frequent closed sequences, *FCS*

```
FCS = Ø ;
F1=frequent 1-sequences(SDB, min_sup);
for (each 1-sequence f1 in F1) do
    SDB^f1 = pseudo projected database (SDB);
for (each f1 in F1) do
    if (!BackScan(f1, SDB^f1))
        BEI=backward extension check (f1, SDB^f1);
        call bide(SDB^f1, f1, min_sup, BEI, FCS);
return FCS;
```

# How does BIDE improve the mining efficiency?

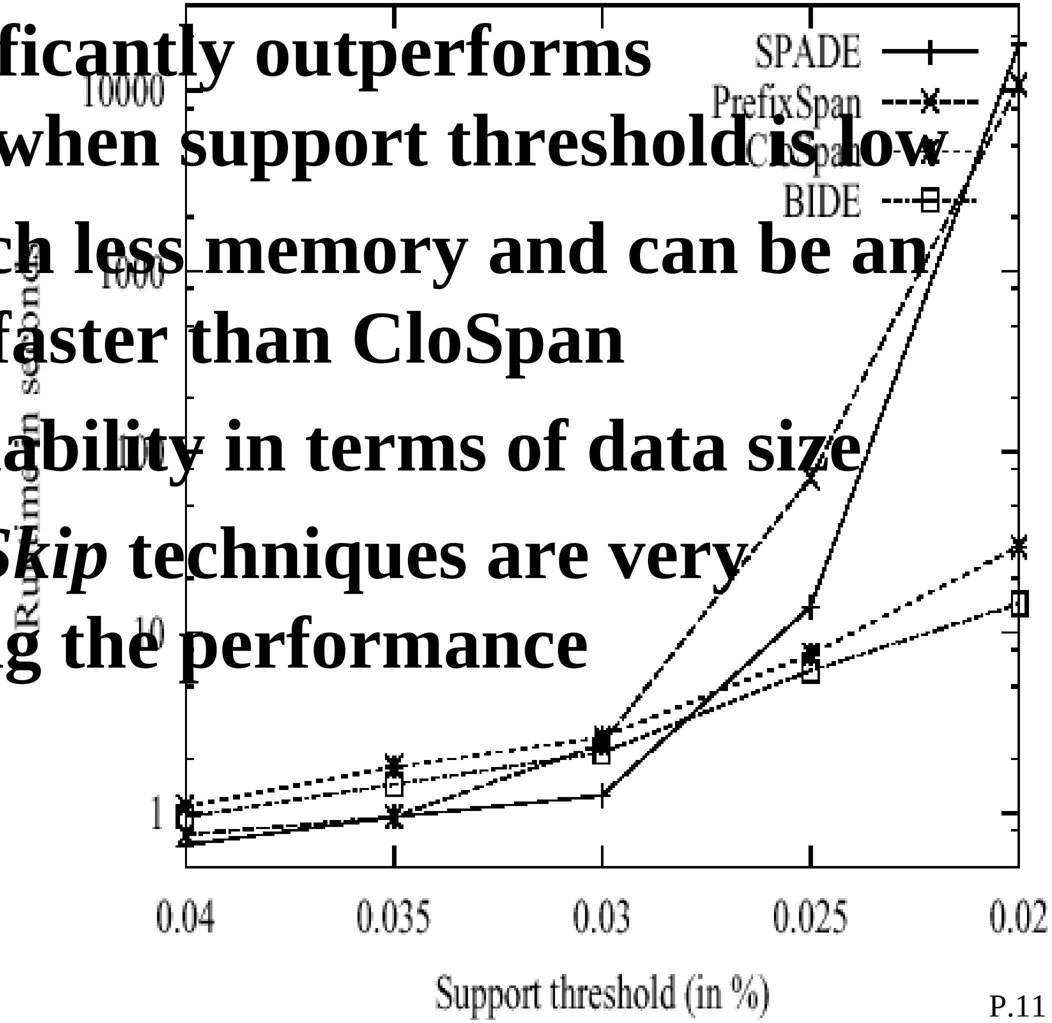
```
bide (Sp_SDB, Sp, min_sup, BEI, FCS)
Input: a projected sequence database Sp_SDB, a prefix sequence Sp,
       a minimum support threshold min_sup, and the number of backward
       extension items BEI
Output: the current set of frequent closed sequences, FCS
LFI = locally frequent items (Sp_SDB);
FEI = |{z in LFI | z.sup = sup^SDB (Sp)}|;
if ((BEI+FEI)==0)
    FCS=FCS ∪ {Sp};
for (each i in LFI) do
    Sp^i = <Sp,i>;
    SDB^Spi = pseudo projected database (Sp_SDB, Sp^i);
for (each i in LFI) do
    if (!BackScan(Sp^i, SDB^Spi))
        BEI=backward extension check (Sp^i, SDB^Spi);
        call bide(SDB^Spi, Sp^i, min_sup, BEI, FCS);
```

# Does BIDE perform much better?

- **BIDE/CloSpan significantly outperforms PrefixSpan/SPADE when support threshold is low**
- **BIDE consumes much less memory and can be an order of magnitude faster than CloSpan**
- **BIDE has linear scalability in terms of data size**
- ***BackScan* and *ScanSkip* techniques are very effective in enhancing the performance**

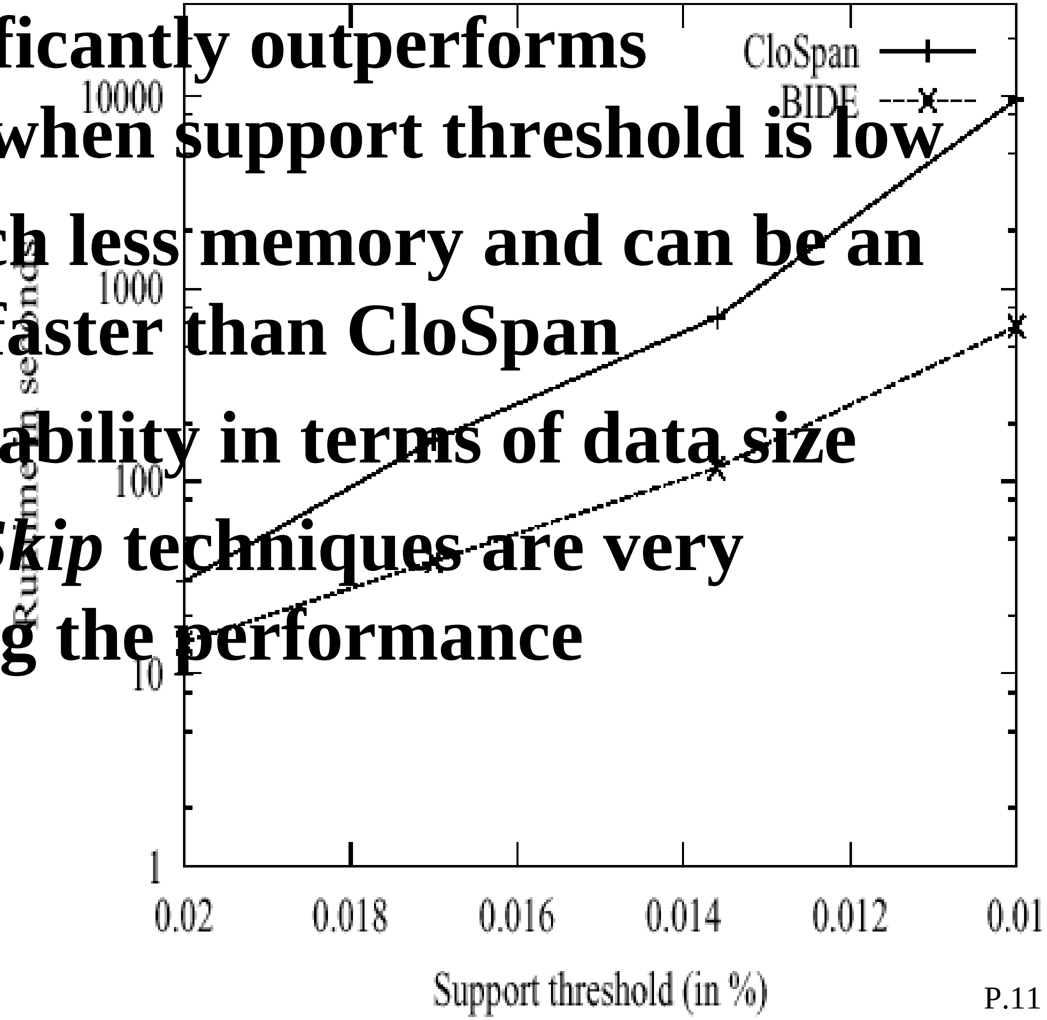
# Does BIDE perform much better?

- **BIDE/CloSpan significantly outperforms PrefixSpan/SPADE when support threshold is low**
- **BIDE consumes much less memory and can be an order of magnitude faster than CloSpan**
- **BIDE has linear scalability in terms of data size**
- ***BackScan* and *ScanSkip* techniques are very effective in enhancing the performance**



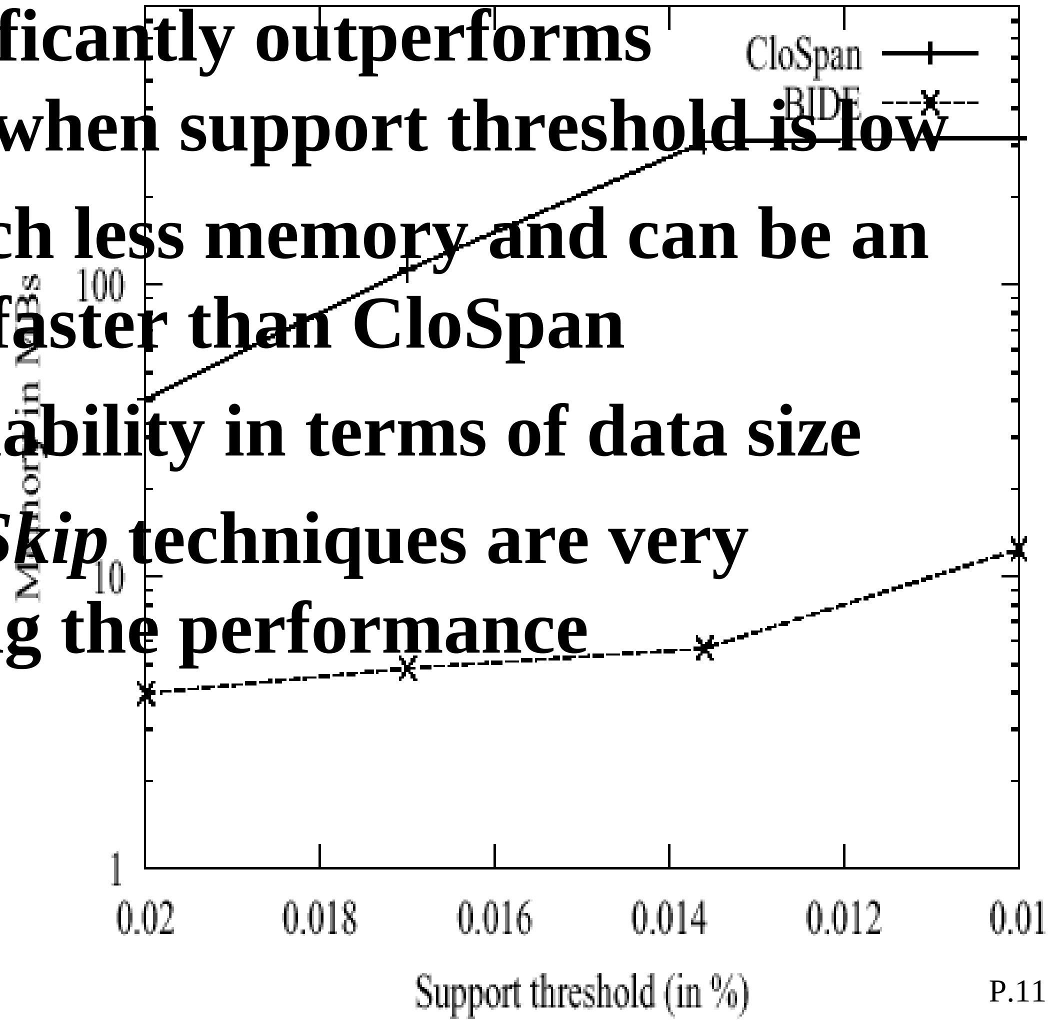
# Does BIDE perform much better?

- BIDE/CloSpan significantly outperforms PrefixSpan/SPADE when support threshold is low
- BIDE consumes much less memory and can be an order of magnitude faster than CloSpan
- BIDE has linear scalability in terms of data size
- *BackScan* and *ScanSkip* techniques are very effective in enhancing the performance



# Does BIDE perform much better?

- **BIDE/CloSpan significantly outperforms PrefixSpan/SPADE when support threshold is low**
- **BIDE consumes much less memory and can be an order of magnitude faster than CloSpan**
- **BIDE has linear scalability in terms of data size**
- ***BackScan* and *ScanSkip* techniques are very effective in enhancing the performance**



# Conclusion Remarks

- **Closed Frequent has the same expressive power as All Frequent, but provides more compact results and likely better efficiency.**
- **Integrated optimization techniques for *database projection*, *search space pruning*, and *pattern-closure checking* are required.**
- **Move *candidate-maintenance-and-test* paradigm to a new paradigm without candidate maintenance**

# Any Question?

| Seq ID. | Sequence |
|---|---|
| 0 | $\langle(af)(d)(e)(a)\rangle$ |
| 1 | $\langle(e)(a)(b)\rangle$ |
| 2 | $\langle(e)(abf)(bde)\rangle$ |
| 3 | <(e)(b)(a)(bd)> |

# My Questions…

| Seq ID. | Sequence |
|---|---|
| 0 | $\langle (af)(d)(e)(a) \rangle$ |
| 1 | $\langle (e)(a)(b) \rangle$ |
| 2 | $\langle (e)(abf)(bde) \rangle$ |
| 3 | <(e)(b)(a)(bd)> |

# My Questions…

- **CloSpan**
  - $<(e)(b)>$ vs. $<(e)(a)(b)>$
  - $D_{<(w)(y)(z)>} = D_{<(x)(y)(z)>} = D_{<(y)(z)>}$
  - $D_{<(w)(x)(y)>} = D_{<(x)(y)>} = D_{<(z)(x)(y)>}$

| Seq ID. | Sequence |
|---|---|
| 0 | $\langle (af)(d)(e)(a) \rangle$ |
| 1 | $\langle (e)(a)(b) \rangle$ |
| 2 | $\langle (e)(abf)(bde) \rangle$ |
| 3 | $<(e)(b)(a)(bd)>$ |

# My Questions…

| Seq ID. | Sequence |
|---|---|
| 0 | $\langle (af)(d)(e)(a) \rangle$ |
| 1 | $\langle (e)(a)(b) \rangle$ |
| 2 | $\langle (e)(abf)(bde) \rangle$ |
| 3 | <(e)(b)(a)(bd)> |

- **CloSpan**
  - <(e)(b)> vs. <(e)(a)(b)>
  - $D_{<(w)(y)(z)>} = D_{<(x)(y)(z)>} = D_{<(y)(z)>}$
  - $D_{<(w)(x)(y)>} = D_{<(x)(y)>} = D_{<(z)(x)(y)>}$
- **BIDE**
  - *How to efficiently compute or maintain $MP_i$/$SMP_i$?*
  - *Does it easily adapt BIDE to sequences of itemsets?*

# My Questions…

- **CloSpan**
  - <(e)(b)> vs. <(e)(a)(b)>
  - $D_{<(w)(y)(z)>} = D_{<(x)(y)(z)>} = D_{<(y)(z)>}$
  - $D_{<(w)(x)(y)>} = D_{<(x)(y)>} = D_{<(z)(x)(y)>}$

| Seq ID. | Sequence |
|---|---|
| 0 | $\langle (af)(d)(e)(a) \rangle$ |
| 1 | $\langle (e)(a)(b) \rangle$ |
| 2 | $\langle (e)(abf)(bde) \rangle$ |
| 3 | <(e)(b)(a)(bd)> |

- **BIDE**
  - *How to efficiently compute or maintain $MP_i$/$SMP_i$?*
  - *Does it easily adapt BIDE to sequences of itemsets?*
- **What is the difference between Closed Frequent Sequences and Non-trivial Repeating Patterns?**