Dynamic Re-ordering in Mining Top-k Productive Discriminative Patterns

Yoshitaka Kameya* and Ken'ya Ito
Meijo University

Outline

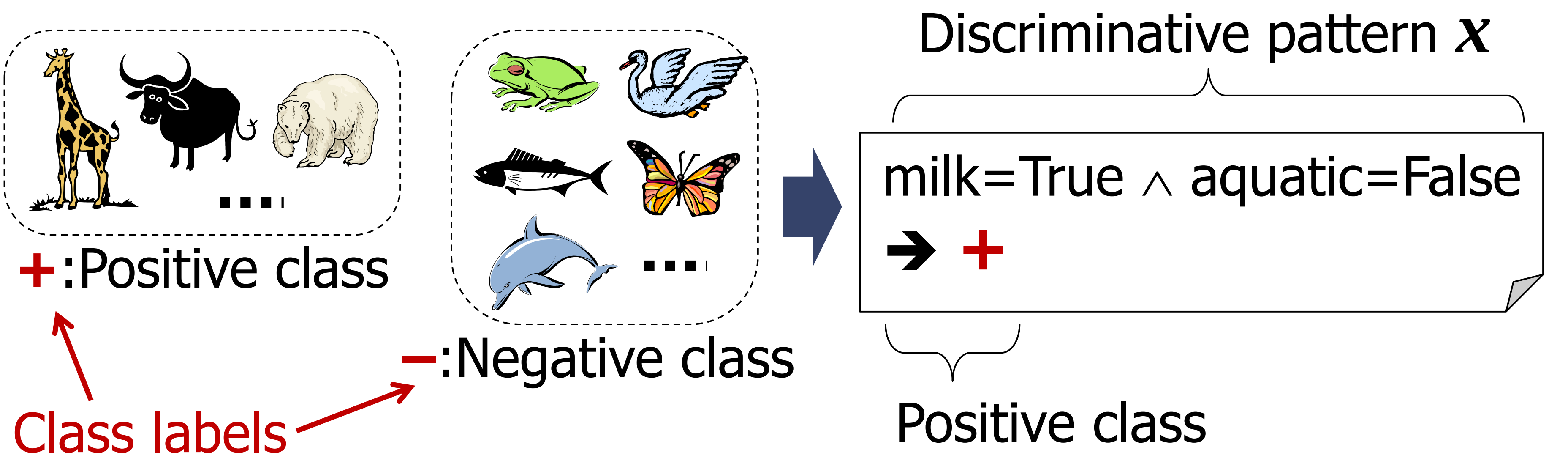
- Background
- Dynamic re-ordering in mining top-k productive discriminative patterns
- Experiments
- Related work and Conclusion

Outline

- Background
- Dynamic re-ordering in mining top-k productive discriminative patterns
- Experiments
- Related work and Conclusion

Background: Discriminative Patterns (1)

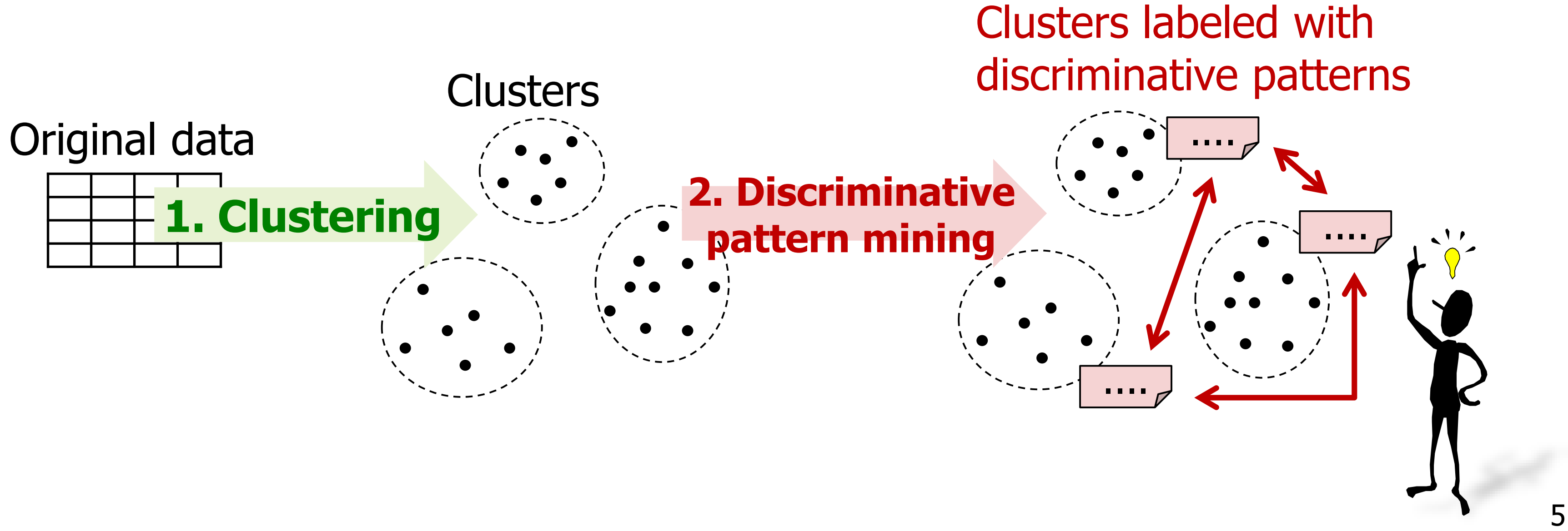
- Discriminative patterns:
  - Show differences between two groups (classes)
  - Used for:
    - Characterizing the positive class
    - Building more precise classifiers



Background: Discriminative Patterns (2)

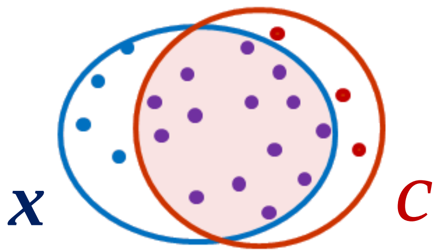
- Discriminative patterns tend to be more meaningful than frequent patterns (thanks to class labels)
- Are class labels always available?
  - Comparing groups is a standard (and promising) starting point in data analysis
  - Clustering can find groups (classes) !

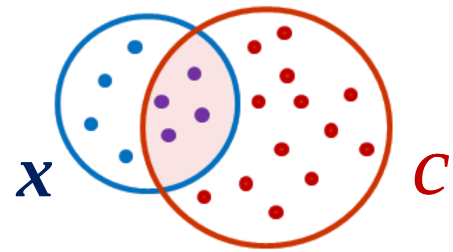
    → **Cluster labeling**



Background: Discriminative Patterns (3)

- Quality score: Measures the overlap between pattern x and positive class c



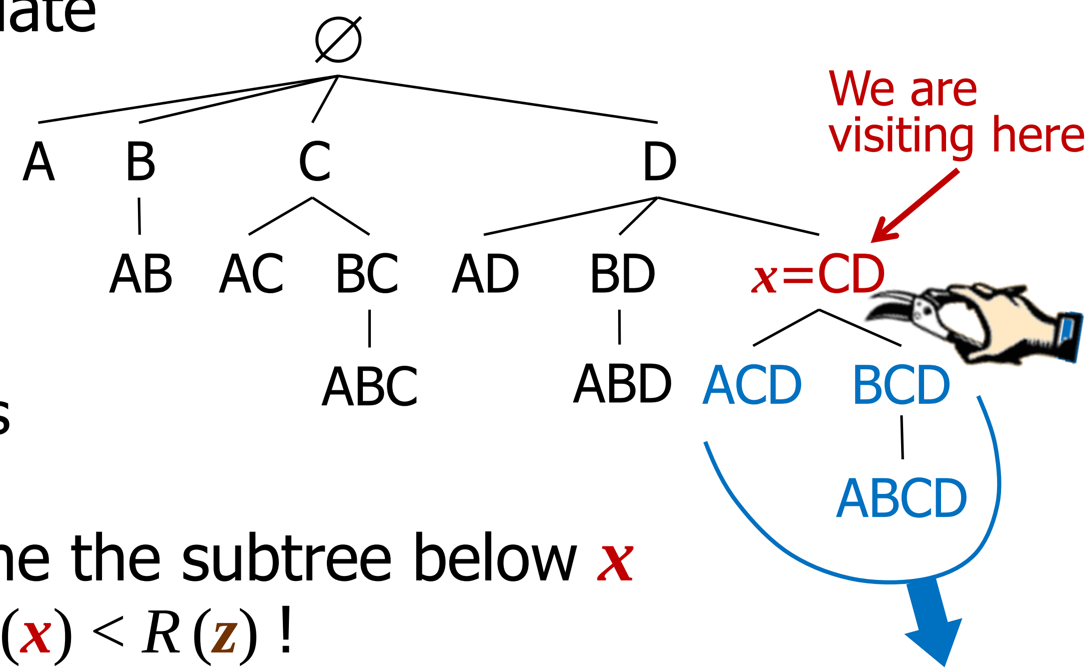


- Most of popular quality scores are *not* anti-monotonic:
  - Confidence, Lift
  - Support difference, Weighted relative accuracy, Leverage
  - F-score, Dice, Jaccard
  - ...

➔ Branch & bound pruning is often used
[Morishita+ 00][Zimmermann+ 09][Nijssen+ 09]

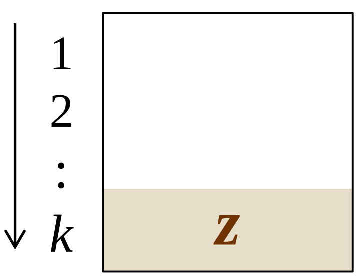
Background: B&B Pruning for Top-k Patterns

- Suppose: we are visiting a pattern $\boldsymbol{x}$ in a depth-first search
- We compute the upper bound $U(\boldsymbol{x})$ of its quality $R(\boldsymbol{x})$
  ($U(\boldsymbol{x})$ = an *optimistic* estimate of qualities of $\boldsymbol{x}$'s extensions)
- We prune the subtree below $\boldsymbol{x}$ if $U(\boldsymbol{x}) < R(\boldsymbol{z})$,
  where $\boldsymbol{z}$ is the k-th candidate



Candidate list
for *tentative* top-k patterns

Descending w.r.t. quality

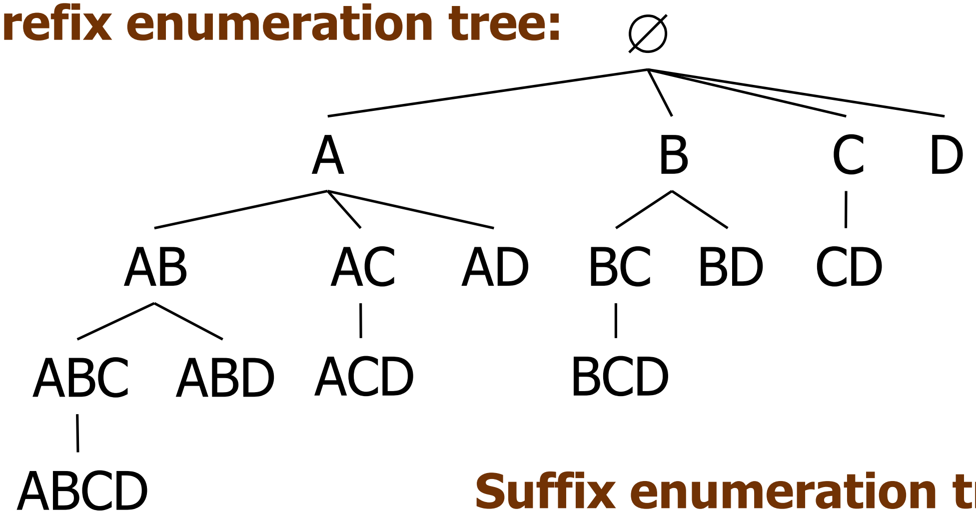


Prune the subtree below $\boldsymbol{x}$
if $U(\boldsymbol{x}) < R(\boldsymbol{z})$!

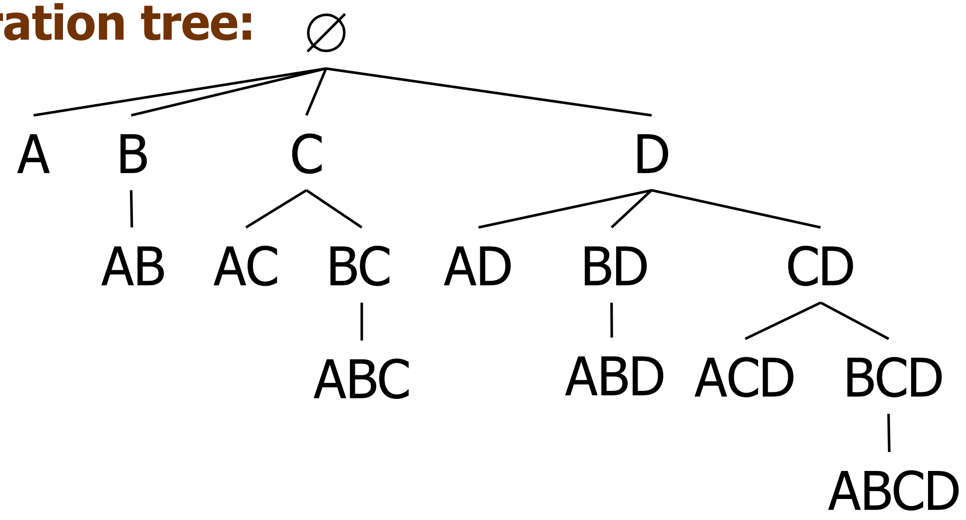
Optimistic estimate:
$U(\boldsymbol{x})$

Background: Suffix Enumeration Trees (1)

Prefix enumeration tree:



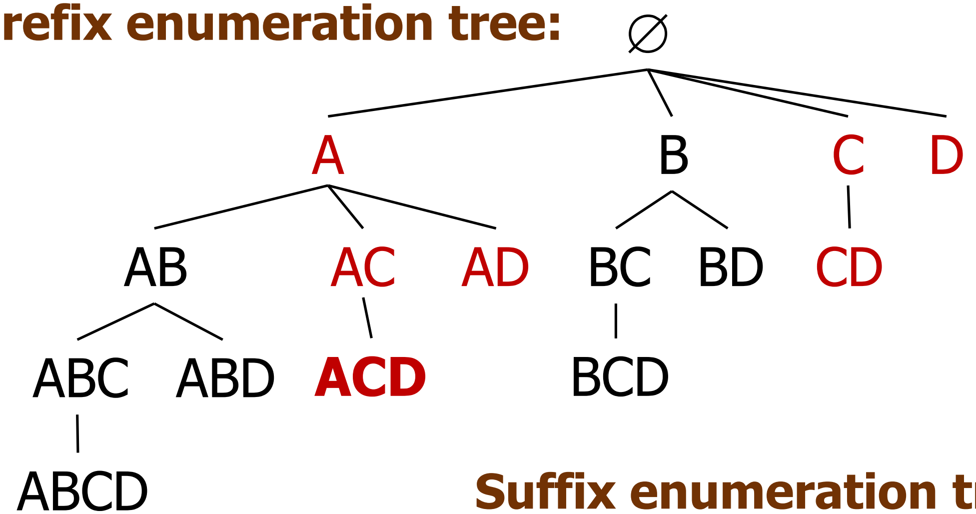
Suffix enumeration tree:



Background: Suffix Enumeration Trees (1)

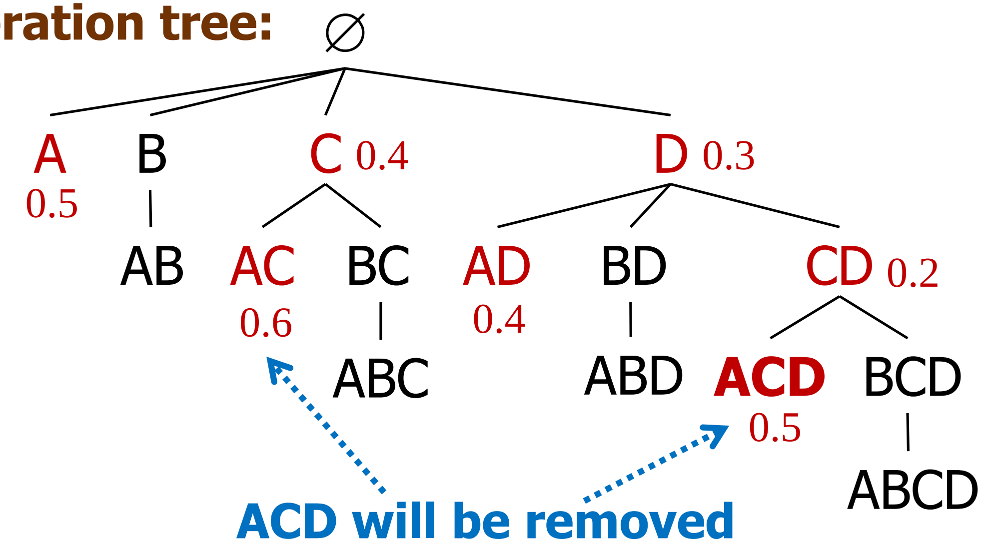
- Beneficial for checking the productivity constraint in a depth-first search

Prefix enumeration tree:



Productivity constraint:
Every pattern must *not* be of less quality than its sub-pattern

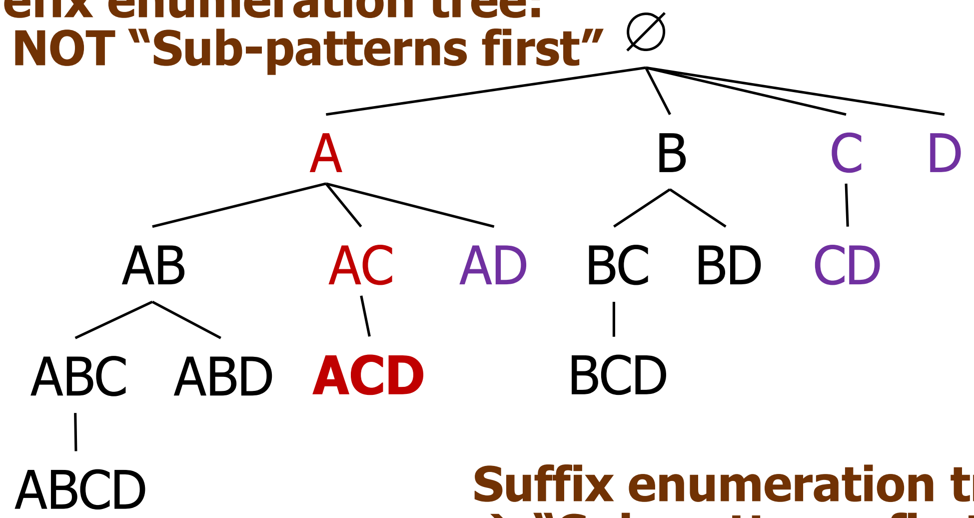
Suffix enumeration tree:



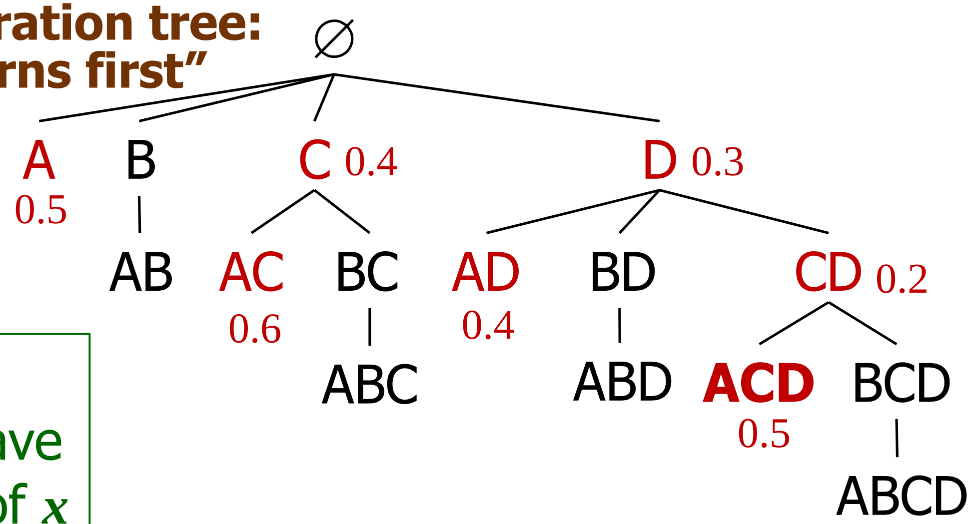
Background: Suffix Enumeration Trees (1)

- Beneficial for checking the productivity constraint in a depth-first search

Prefix enumeration tree:
→ NOT "Sub-patterns first"



Suffix enumeration tree:
→ "Sub-patterns first"



"Sub-patterns first" property:
When visiting a pattern x, we have already visited all sub-patterns of x

Background: Suffix Enumeration Trees (2)

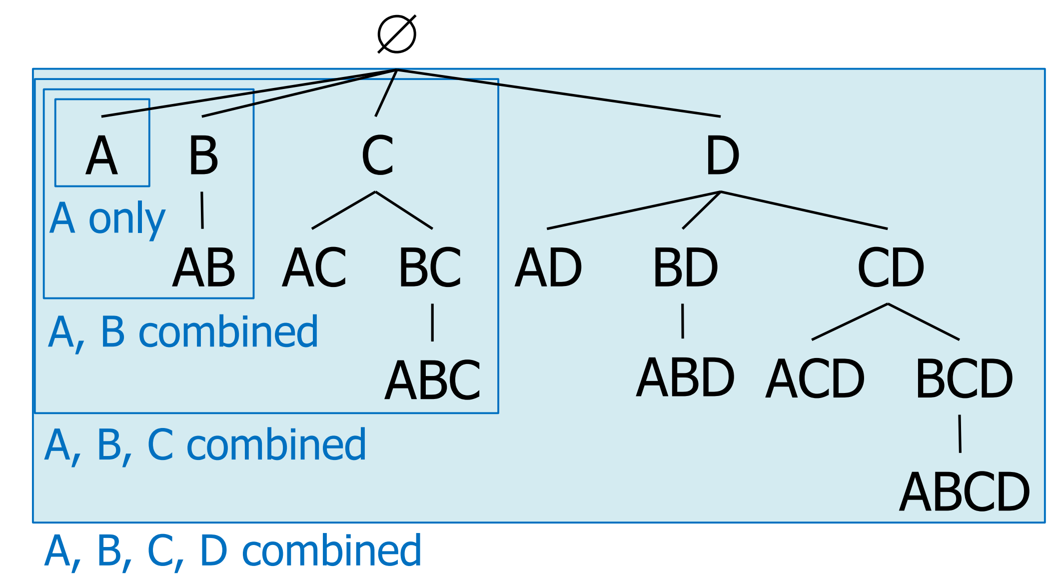
- *Also* beneficial for effective B&B pruning

Suppose: A = the highest quality item,
B = the 2^{nd} highest quality item,
C = the 3^{rd} highest quality item,
…

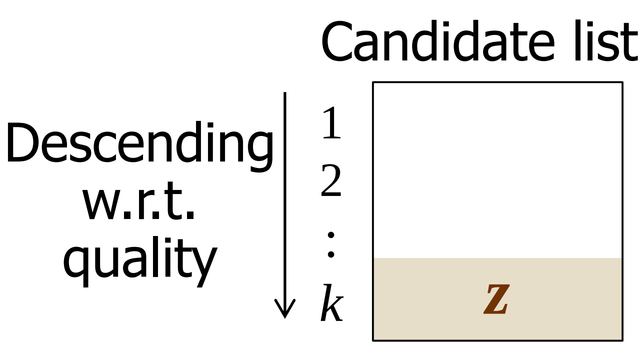
➔ Items of higher quality are combined earlier

➔ Patterns of higher quality *would* be visited earlier

Suffix enumeration tree:



B&B pruning would be more aggressive!



We prune the subtree below x if $U(x) < R(z)$

➔ Threshold in B&B pruning is higher if z has a higher quality

Outline

✓ Background

- Dynamic re-ordering in mining top-k productive discriminative patterns
  - Basic idea
  - Justification
- Experiments
- Related work and Conclusion

Outline

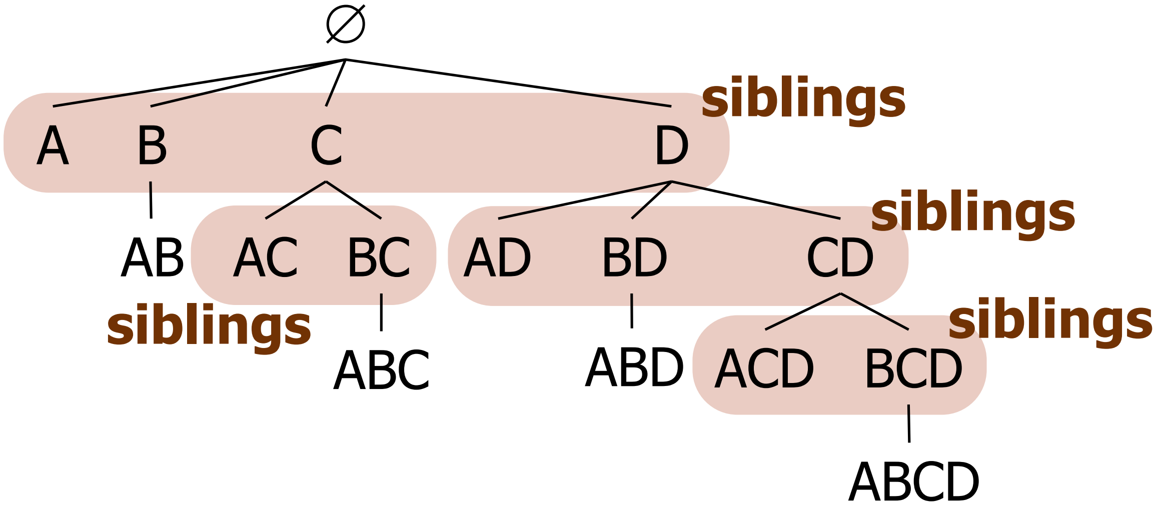
✓ Background

- Dynamic re-ordering in mining top-k productive discriminative patterns
  - Basic idea
  - Justification
- Experiments
- Related work and Conclusion

Our proposal: Basic idea (1)

- **Basic idea**:
  Re-order sibling patterns *dynamically* according to their qualities

  ➔ Patterns of higher quality will be visited *yet earlier*
  ➔ B&B pruning will be *yet more* aggressive



Our proposal: Basic idea (2)

- **Example:**
  - 10 transactions
  - Quality is measured by F-score

Dataset

	Class	Transaction
Positive	+	{A, B}
	+	{A, C, E}
	+	{A, D}
	+	{B, C, E}
	+	{B, D}
Negative	–	{A, B, C}
	–	{B, E}
	–	{C, D}
	–	{C, D, E}
	–	{E}

Our proposal: Basic idea (4)

- **Example:**
  - 10 transactions
  - Quality is measured by F-score

Recall of {A} = 3 / 5 = 0.6

Precision of {A} = 3 / 4 = 0.75

F-score of {A} =
2 * 0.6 * 0.75 / (0.6 + 0.75) = 0.67

Dataset

	Class	Transaction
Positive	+	{A, B}
	+	{A, C, E}
	+	{A, D}
	+	{B, C, E}
	+	{B, D}
Negative	–	{A, B, C}
	–	{B, E}
	–	{C, D}
	–	{C, D, E}
	–	{E}

  - Similarly, we have:
    - F-score of {A} = 0.67
    - F-score of {B} = 0.6
    - F-score of {C} = 0.4
    - F-score of {D} = 0.44
    - F-score of {E} = 0.4

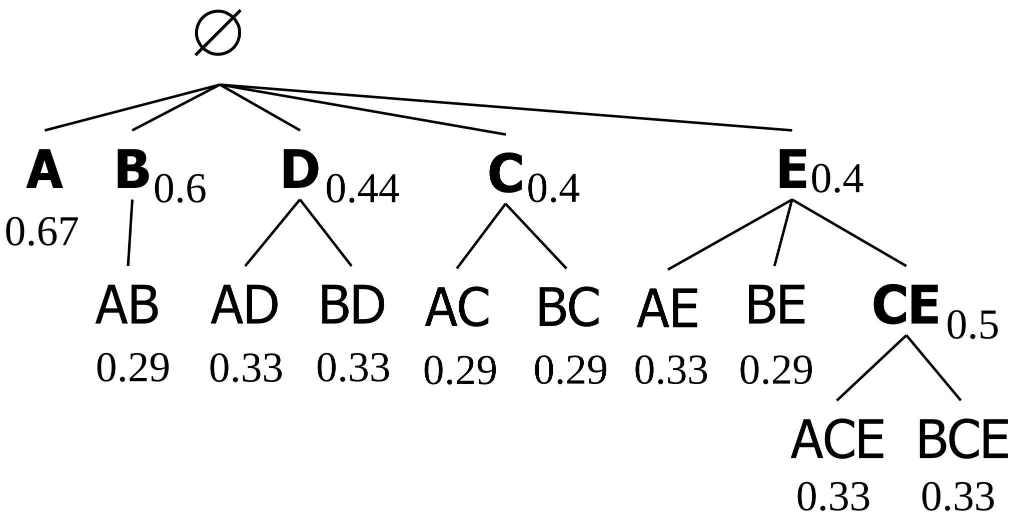
Static ordering among patterns:

A < B < D < C < E

Our proposal: Basic idea (4)

- **Example:**
  - 10 transactions
  - Quality is measured by F-score

Suffix enumeration tree under static ordering A < B < D < C < E:



Dataset

	Class	Transaction
Positive	+	{A, B}
	+	{A, **C**, **E**}
	+	{A, D}
	+	{B, **C**, **E**}
	+	{B, D}
Negative	–	{A, B, **C**}
	–	{B, **E**}
	–	{**C**, D}
	–	{**C**, D, **E**}
	–	{**E**}

(Note)
Patterns that do not appear in the dataset are hidden

"Sub-patterns first" property holds and we have productive patterns {A}, {B}, {C, E}, {D}, {C}, {E}

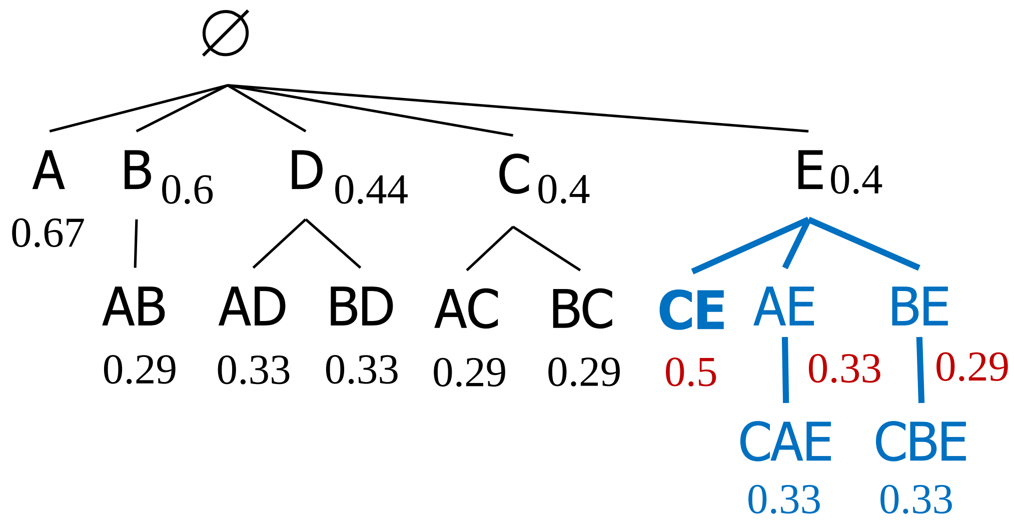
Our proposal: Basic idea (4)

- **Example:**
  - 10 transactions
  - Quality is measured by F-score

Dataset

	Class	Transaction
Positive	+	{A, B}
	+	{A, C, E}
	+	{A, D}
	+	{B, C, E}
	+	{B, D}
Negative	–	{A, B, C}
	–	{B, E}
	–	{C, D}
	–	{C, D, E}
	–	{E}

Suffix enumeration tree with dynamic re-ordering:



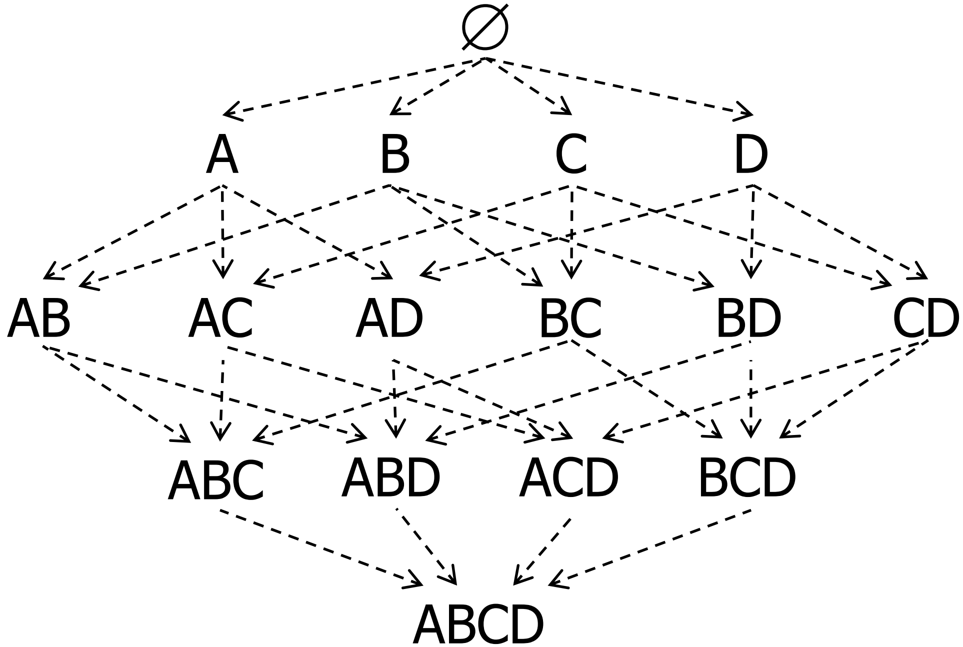
{C, E} comes earlier than before and it is interesting to see the "sub-patterns first" property *still* holds ➔ **Why?**

Outline

- ✓ Background
- Dynamic re-ordering in mining top-k productive discriminative patterns
  - ✓ Basic idea
  - – Justification
- Experiments
- Related work and Conclusion

Our proposal: Justification (1)

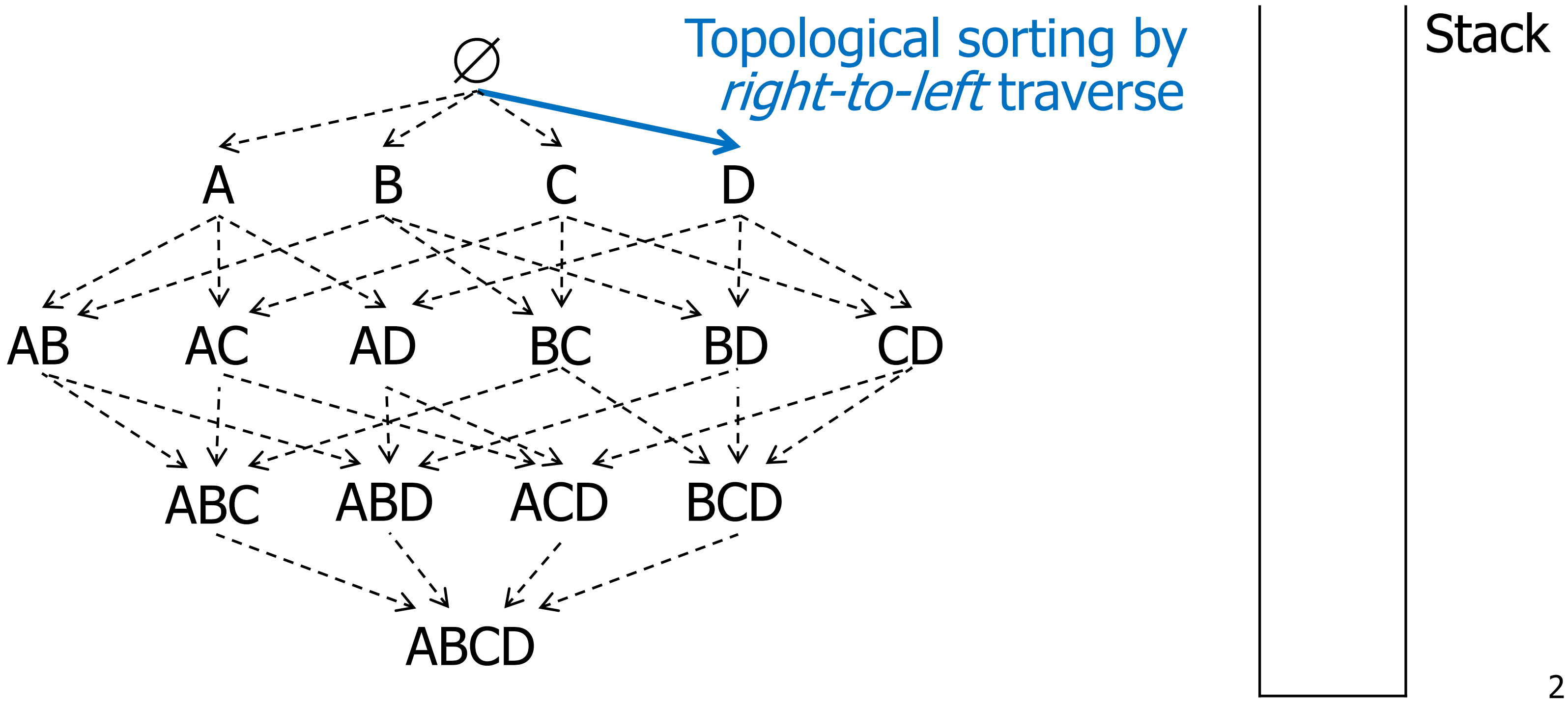
- "Sub-patterns first" property is assured *even with* dynamic re-ordering
- **Key observation:**
  Visiting order of a search =
  $\exists$topological order over a Hasse diagram
  $\Rightarrow$ The search is "sub-patterns first"



Our proposal: Justification (2)

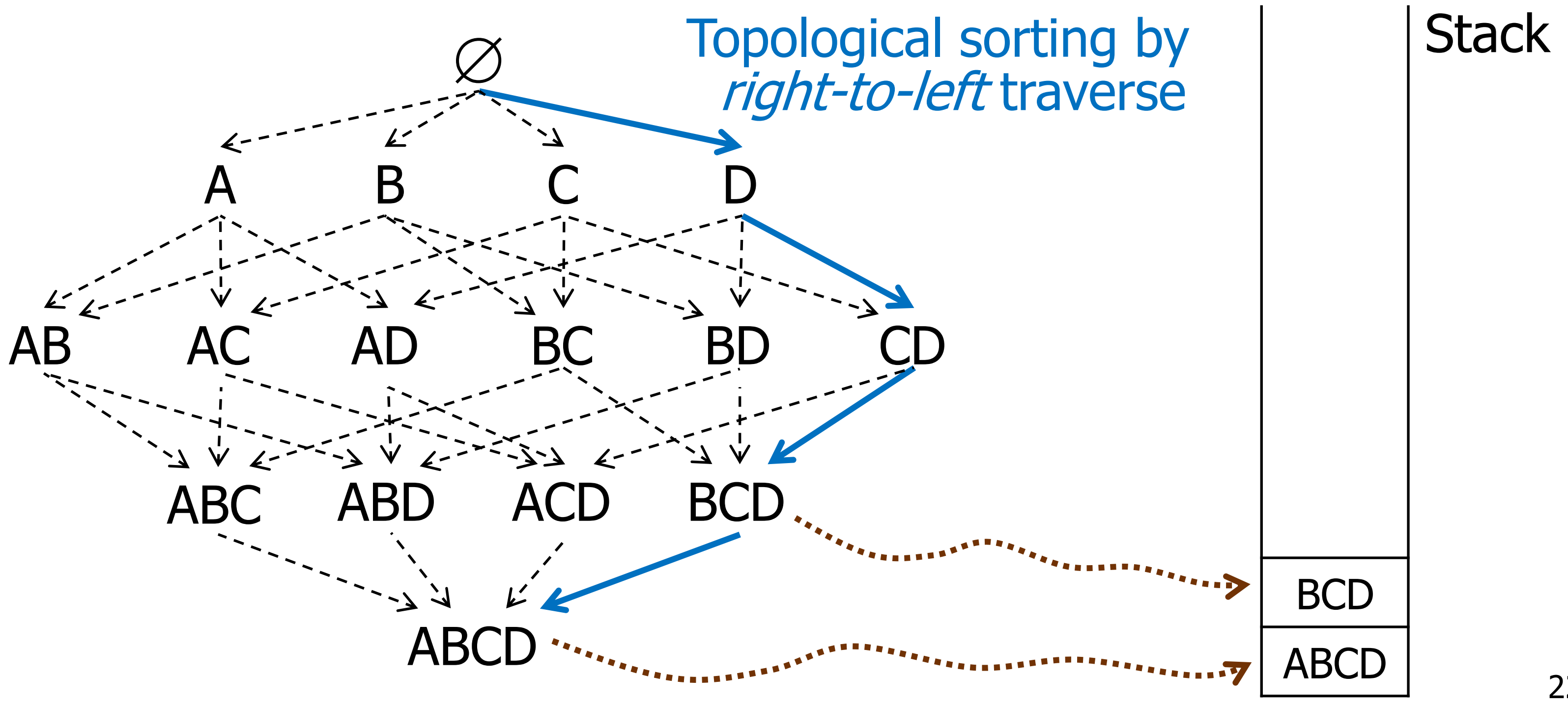
- "Sub-patterns first" property is assured *even with* dynamic re-ordering

- **Key observation:**
  Visiting order of a search =
  ∃topological order over a Hasse diagram
  ⇒ The search is "sub-patterns first"



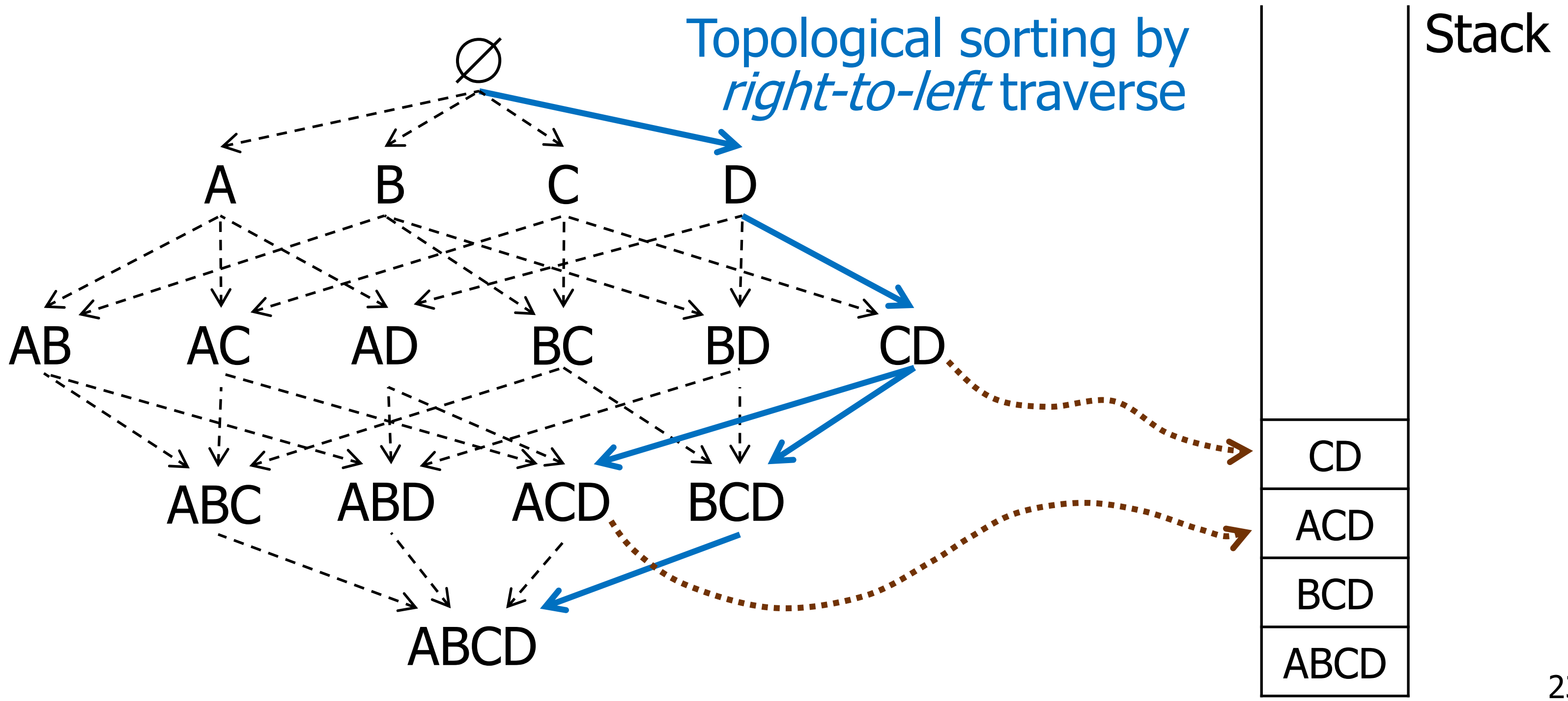
Our proposal: Justification (2)

- "Sub-patterns first" property is assured *even with* dynamic re-ordering
- **Key observation:**
  Visiting order of a search =
  ∃topological order over a Hasse diagram
  $\Rightarrow$ The search is "sub-patterns first"



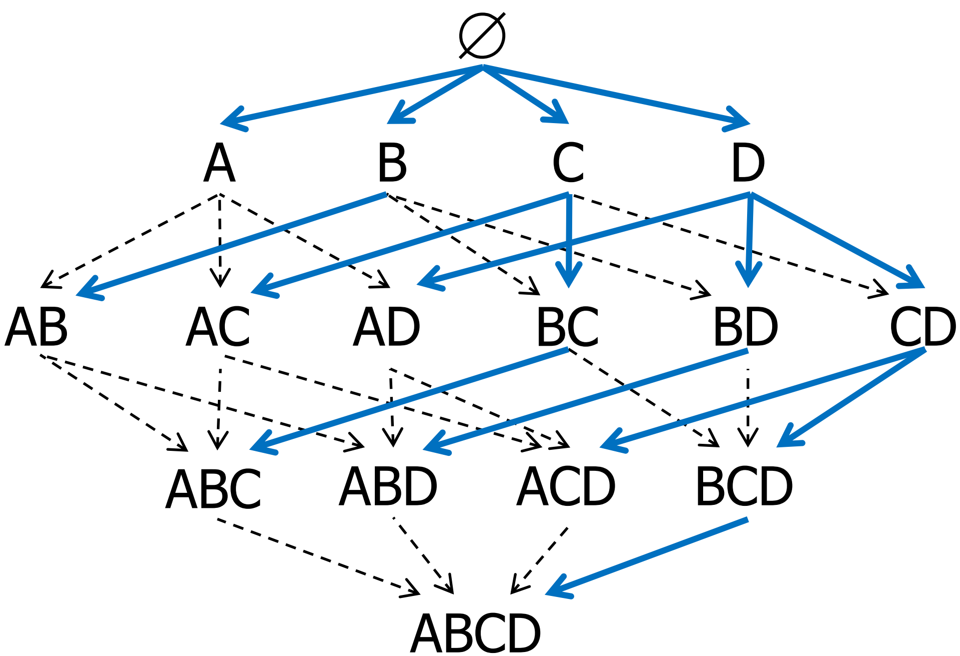
Our proposal: Justification (2)

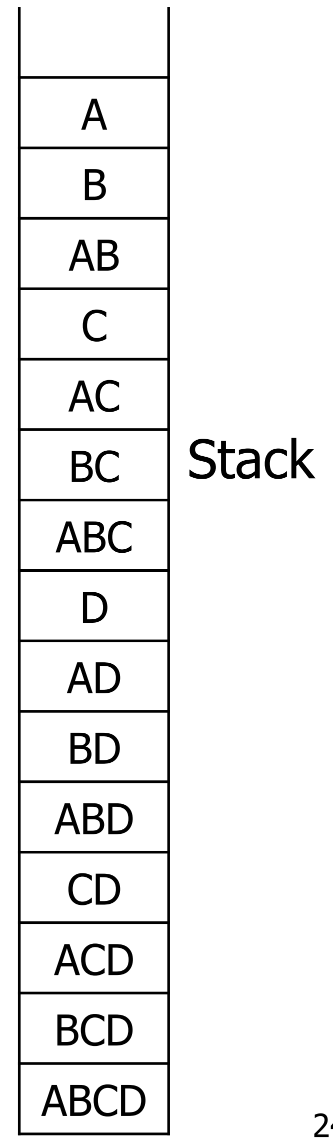
- "Sub-patterns first" property is assured *even with* dynamic re-ordering
- **Key observation:**
  Visiting order of a search =
  ∃topological order over a Hasse diagram
  ⇒ The search is "sub-patterns first"



Our proposal: Justification (2)

- "Sub-patterns first" property is assured *even with* dynamic re-ordering
- **Key observation:**
  Visiting order of a search =
  $\exists$topological order over a Hasse diagram
  $\Rightarrow$ The search is "sub-patterns first"



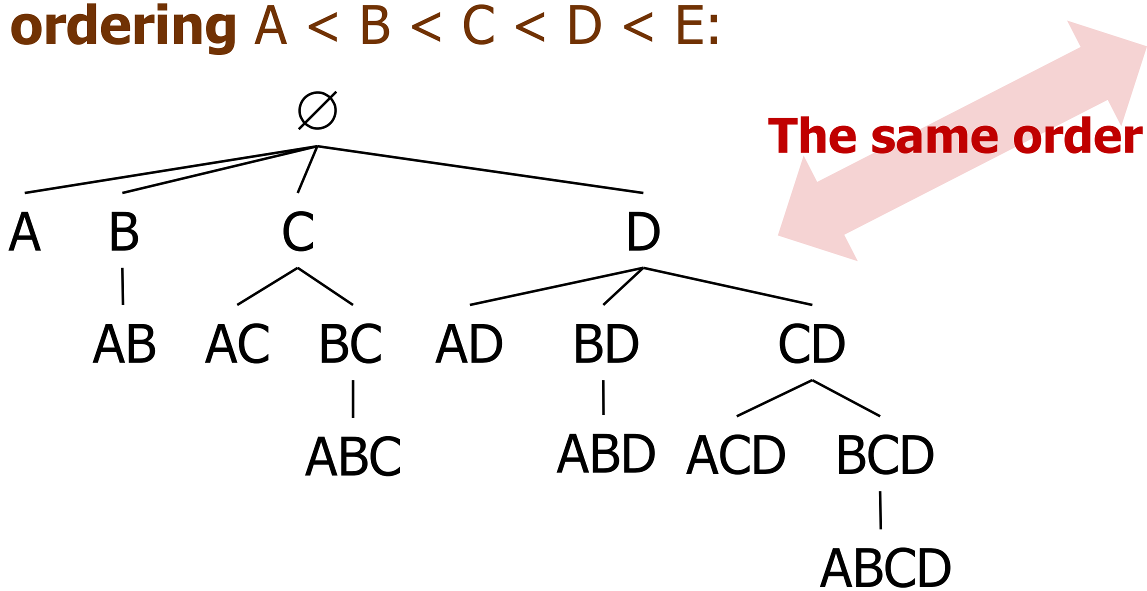


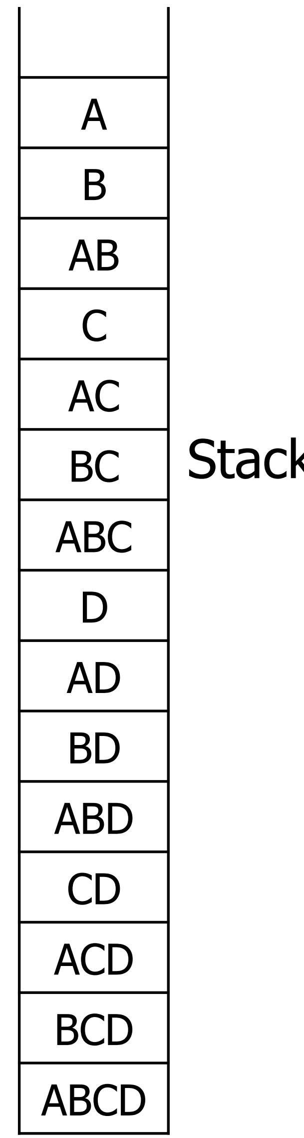
Our proposal: Justification (2)

- "Sub-patterns first" property is assured *even with* dynamic re-ordering

- **Key observation:**
  Visiting order of a search =
  $\exists$topological order over a Hasse diagram
  $\Rightarrow$ The search is "sub-patterns first"

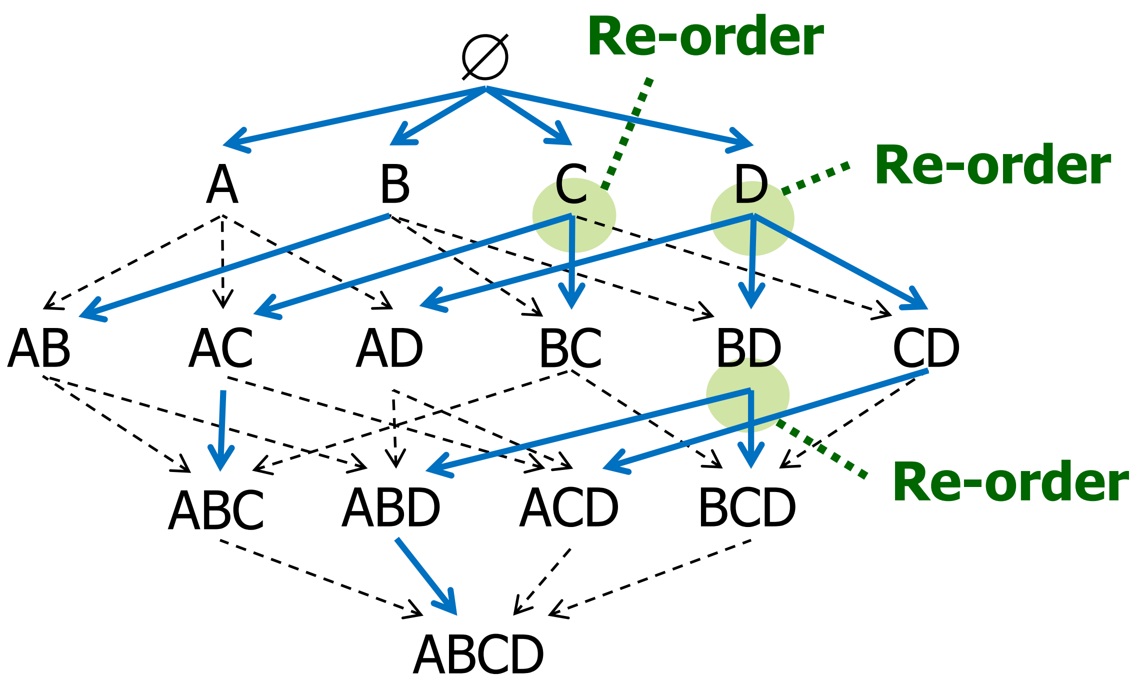
Suffix enumeration tree
with a static ordering A < B < C < D < E:

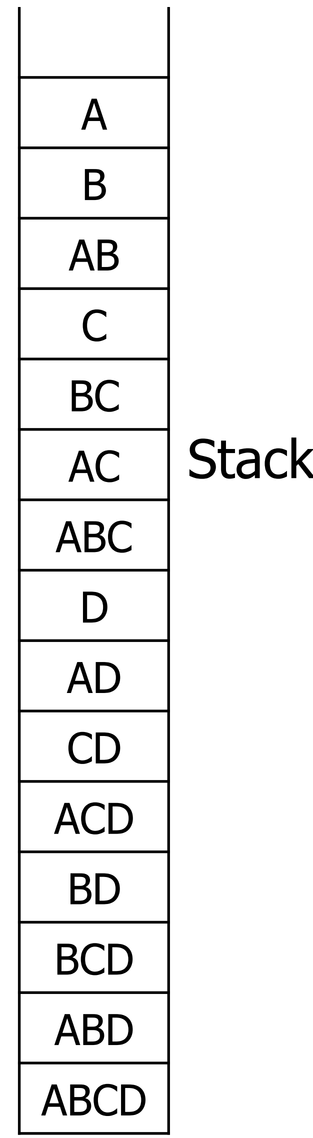




Our proposal: Justification (3)

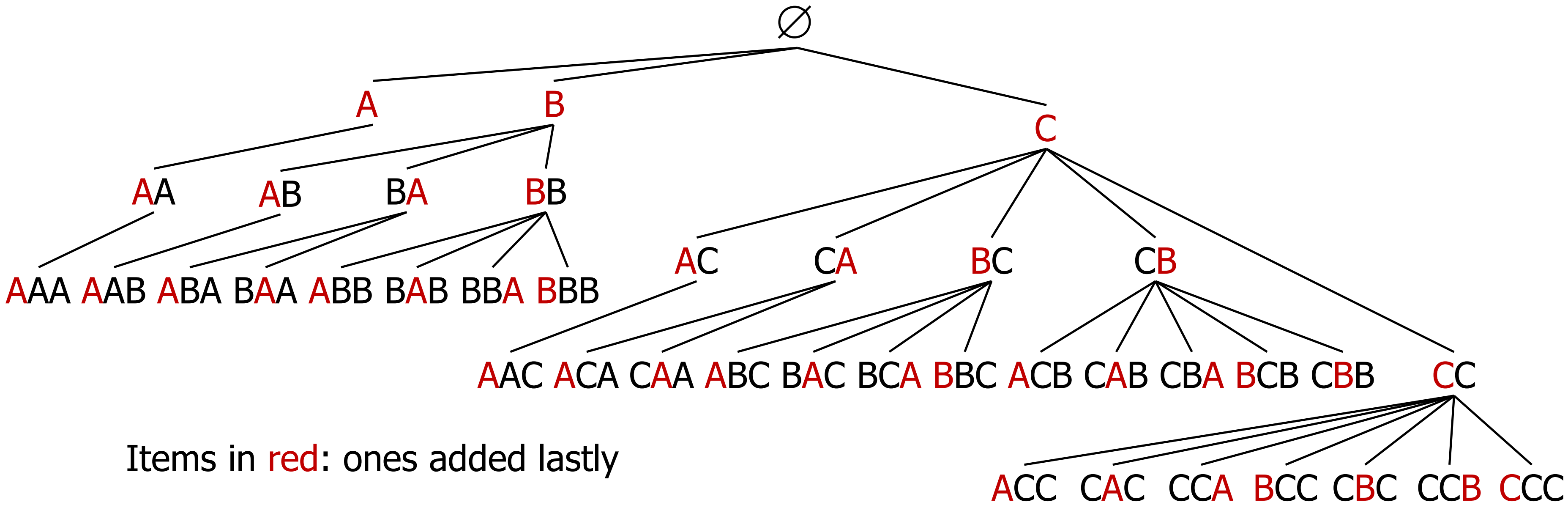
- "Sub-patterns first" property is assured *even with* dynamic re-ordering
- We can always consider a topological sorting that simulates our dynamic re-ordering





Our proposal: Justification (4)

- Topological sorting over a Hasse diagram also help us justify a "sub-patterns first" enumeration tree for sequence patterns:



To build this enumeration tree, we extend $\boldsymbol{x}$ whose lastly added item is u as follows:

- Insert items u or x such that $x < u$ in the ascending order w.r.t. $<$
- When inserting x, insert it everywhere outside/between the items in $\boldsymbol{x}$
- When inserting u, insert it on the left side of the lastly added u

- SPADE-like algorithm using a vertical layout can work with this tree, though max-gap constraint does not hold monotonically

Outline

- ✓ Background
- ✓ Dynamic re-ordering in mining top-k productive discriminative patterns
  - ✓ Basic idea
  - ✓ Justification
- Experiments
- Related work and Conclusion

Experiments: Settings

- Target: 16 datasets preprocessed by the CP4IM project:

Dataset	#Trans.	#Items
anneal	812	93
audiology	216	148
australian-credit	653	125
german-credit	1,000	112
heart-cleveland	296	95
hepatitis	137	68
hypothyroid	3,247	88
kr-vs-kp	3,196	73

Dataset	#Trans.	Items
lymph	148	68
mushroom	8,124	110
primary-tumor	336	31
soybean	630	50
splice-1	3,190	287
tic-tac-toe	958	28
vote	435	48
zoo-1	101	36

- We compare 3 variants of FP-growth with:
  - Static ordering based on quality (**Static**)
  - Static random ordering (**Random**)
  - Dynamic re-ordering (**Dynamic**; the proposed method)

Experiments: Results (1)

- Number k of output patterns = 1 (**lightweight** cases)

Dataset	Entire # of visited patterns			
	Static	Dynamic	Random	Reduction ratio
anneal	2.4E+5	2.4E+5	2.5E+5	0.0
audiology	N/A	N/A	N/A	N/A
australian-credit	5.1E+3	5.1E+3	1.1E+4	0.0
german-credit	3.4E+2	3.4E+2	3.6E+2	0.0
heart-cleveland	5.7E+3	5.7E+3	7.1E+3	0.0
hepatitis	8.0E+1	8.0E+1	9.0E+1	0.0
hypothyroid	1.2E+3	1.2E+3	2.5E+3	0.0
kr-vs-kp	2.0E+5	2.0E+5	2.6E+5	0.0
lymph	1.1E+4	1.1E+4	1.2E[illegible]	[illegible]
mushroom	1.2E+2	1.2E+2	1.4E[illegible]	[illegible]
primary-tumor	8.8E+2	8.8E+2	1.1E[illegible]	[illegible]
soybean	4.3E+3	4.3E+3	5.1E[illegible]	[illegible]
splice-1	2.5E+2	2.5E+2	2.5E[illegible]	[illegible]
tic-tac-toe	2.7E+1	2.7E+1	2.8E[illegible]	[illegible]
vote	4.8E+1	4.8E+1	5.1E+1	0.0
zoo-1	5.4E+1	5.4E+1	7.2E+1	0.0

(Note) "f E + i" indicates "$f \times 10^i$"

Reduction ratio
= (⟨Static⟩ − ⟨Dynamic⟩) / ⟨Static⟩

- **Dynamic** shows *no* performance improvement from **Static**
- **Static** and **Dynamic** work slightly better than **Random**

Experiments: Results (2)

- Number k of output patterns = 1 (**lightweight** cases)

Dataset	Running time (sec)			
	Static	Dynamic	Random	Reduction ratio
anneal	1.11	1.30	1.15	−0.17
audiology	N/A	N/A	N/A	N/A
australian-credit	0.49	0.64	0.64	−0.29
german-credit	0.40	0.40	0.44	0.01
heart-cleveland	0.45	0.45	0.61	−0.01
hepatitis	0.06	0.07	0.08	−0.07
hypothyroid	0.73	0.76	0.77	−0.03
kr-vs-kp	0.86	1.52	1.71	−0.76
lymph	0.44	0.48	0.44	−0.08
mushroom	0.21	0.21	0.44	0.01
primary-tumor	0.09	0.10	0.11	−0.13
soybean	0.21	0.23	0.24	−0.09
splice-1	0.65	0.65	0.66	0.00
tic-tac-toe	0.05	[illegible]	[illegible]	[illegible]
vote	0.05	[illegible]	[illegible]	[illegible]
zoo-1	0.03	[illegible]	[illegible]	[illegible]

Reduction ratio
= $(\langle\text{Static}\rangle - \langle\text{Dynamic}\rangle) / \langle\text{Static}\rangle$

Dynamic is slightly slower than **Static** due to some overhead by re-ordering (though it seems ignorable in practice)

Experiments: Results (3)

- Number k of output patterns = 50 (**burdensome** cases)

Dataset	Entire # of visited patterns			
	Static	Dynamic	Random	Reduction ratio
anneal	9.0E+5	7.6E+5	7.5E+6	0.16
audiology	N/A	N/A	N/A	N/A
australian-credit	1.7E+5	1.4E+5	1.1E+7	0.17
german-credit	2.3E+6	1.1E+6	3.2E+5	0.51
heart-cleveland	3.2E+4	2.7E+4	4.5E+6	0.16
hepatitis	3.1E+7	1.4E+7	7.7E+6	0.54
hypothyroid	N/A	N/A	N/A	N/A
kr-vs-kp	4.3E+5	4.3E+5	9.8E+5	0.00
lymph	2.1E+4	1.9E+4	4.4E+4	0.06
mushroom	2.0E+4	1.7E+4	1.0E+4	0.16
primary-tumor	3.8E+4	2.4E+4	2.4E+4	0.37
soybean	1.4E+4	1.4E+4	1.6E+4	0.00
splice-1	1.5E+3	1.5E+3	1.0E+4	0.01
tic-tac-toe	2.0E+3	1.4E+3	1.3E+3	0.30
vote	1.6E+5	8.0E+4	4.6E+4	0.49
zoo-1	2.7E+3	2.6E+3	2.1E+3	0.01

Reduction ratio
= (⟨Static⟩ – ⟨Dynamic⟩) / ⟨Static⟩

Dynamic outperforms **Random** in some cases

Experiments: Results (3)

- Number k of output patterns = 50 (**burdensome** cases)

Dataset	Entire # of visited patterns			
	Static	Dynamic	Random	Reduction ratio
anneal	9.0E+5	7.6E+5	7.5E+6	0.16
audiology	N/A	N/A	N/A	N/A
australian-credit	1.7E+5	1.4E+5	1.1E+7	0.17
german-credit	2.3E+6	1.1E+6	3.2E+5	0.51
heart-cleveland	3.2E+4	2.7E+4	4.5E+6	0.16
hepatitis	3.1E+7	1.4E+7	7.7E+6	0.54
hypothyroid	N/A	N/A	N/A	N/A
kr-vs-kp	4.3E+5	4.3E+5	9.8E+5	0.00
lymph	2.1E+4	1.9E+4	4.4E+4	[illegible]
mushroom	2.0E+4	1.7E+4	1.0E+4	[illegible]
primary-tumor	3.8E+4	2.4E+4	2.4E+4	[illegible]
soybean	1.4E+4	1.4E+4	1.6E+4	[illegible]
splice-1	1.5E+3	1.5E+3	1.0E+4	0.01
tic-tac-toe	2.0E+3	1.4E+3	1.3E+3	0.30
vote	1.6E+5	8.0E+4	4.6E+4	0.49
zoo-1	2.7E+3	2.6E+3	2.1E+3	0.01

Reduction ratio
= (⟨Static⟩ − ⟨Dynamic⟩) / ⟨Static⟩

Dynamic alleviates the bad influence of the initial order

Experiments: Results (4)

- Number k of output patterns = 50 (**burdensome** cases)

Dataset	Running time (sec)			
	Static	Dynamic	Random	Reduction ratio
anneal	2.69	2.93	45.76	–0.17
audiology	N/A	N/A	N/A	N/A
australian-credit	0.89	0.83	44.12	0.06
german-credit	20.16	5.15	6.42	0.74
heart-cleveland	0.70	0.70	17.39	0.01
hepatitis	117.56	42.75	20.52	0.64
hypothyroid	N/A	N/A	N/A	N/A
kr-vs-kp	2.07	2.21	8.29	–0.06
lymph	0.51	0.52	1.01	–0.03
mushroom	1.02	0.93	1.40	0.09
primary-tumor	0.96	0.70	0.74	0.27
soybean	0.44	0.47	0.46	–0.05
splice-1	1.21	1.33	1.69	–0.10
tic-tac-toe	0.18	0.19	0.17	–0.06
vote	1.61	1.45	0.88	0.10
zoo-1	0.17	0.19	0.18	–0.09

Reduction ratio
= (⟨Static⟩ – ⟨Dynamic⟩) / ⟨Static⟩

Dynamic shows
a stable performance

Experiments: Results (5)

- We also recorded the number of visited patterns until *true* top-k pattern lastly found has been visited
(= the **effective** number of visited patterns)

Dataset	Entire # of visited patterns			Effective # of visited patterns		
	Static	Dynamic	Random	Static	Dynamic	Random
anneal	9.0E+5	7.6E+5	7.5E+6	8.9E+5	7.5E+5	7.1E+6
audiology	N/A	N/A	N/A	N/A	N/A	N/A
australian-credit	1.7E+5	1.4E+5	1.1E+7	1.4E+4	6.6E+3	1.0E+7
german-credit	2.3E+6	1.1E+6	3.2E+5	2.3E+6	1.1E+6	3.2E+5
heart-cleveland	3.2E+4	2.7E+4	4.5E+6	1.8E+3	8.8E+2	4.5E+6
hepatitis	3.1E+7	1.4E+7	7.7E+6	3.1E+7	1.4E+7	7.7E+6
hypothyroid	N/A	N/A	N/A	N/A	N/A	N/A
kr-vs-kp	4.3E+5	4.3E+5	9.8E+5	1.8E+3	1.7E+3	8.1E+5
lymph	2.1E+4	1.9E+4	4.4E+4	3.3E+3	2.6E+3	3.8E+4
mushroom	2.0E+4	1.7E+4	1.0E+4	2.0E+4	1.7E+4	1.0E+4
primary-tumor	3.8E+[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
soybean	1.4E+[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
splice-1	1.5E+[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
tic-tac-toe	2.0E+[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
vote	1.6E+5	8.0E+4	4.6E+4	1.6E+5	7.9E+4	4.0E+4
zoo-1	2.7E+3	2.6E+3	2.1E+3	2.2E+3	2.2E+3	1.9E+3

Dynamic works as a better *anytime* algorithm than others for some datasets

Outline

✓ Background
✓ Dynamic re-ordering in mining top-k productive discriminative patterns
  ✓ Basic idea
  ✓ Justification
✓ Experiments
- Related work and Conclusion

Related work and Conclusion

- “Sub-patterns first” property was firstly introduced in selecting frequent minimal generators [Li+ 06]
- Dynamic re-ordering itself has been introduced in:
  - OPUS [Webb 95]
  - SD-Map* [Atzmueller+ 09]
- This work’s originality:
  productivity constraint + dynamic re-ordering
  - Formally justified using the notion of topological sorting over a Hasse diagram
  - Empirically supported by experiments

Thank you for your attention!

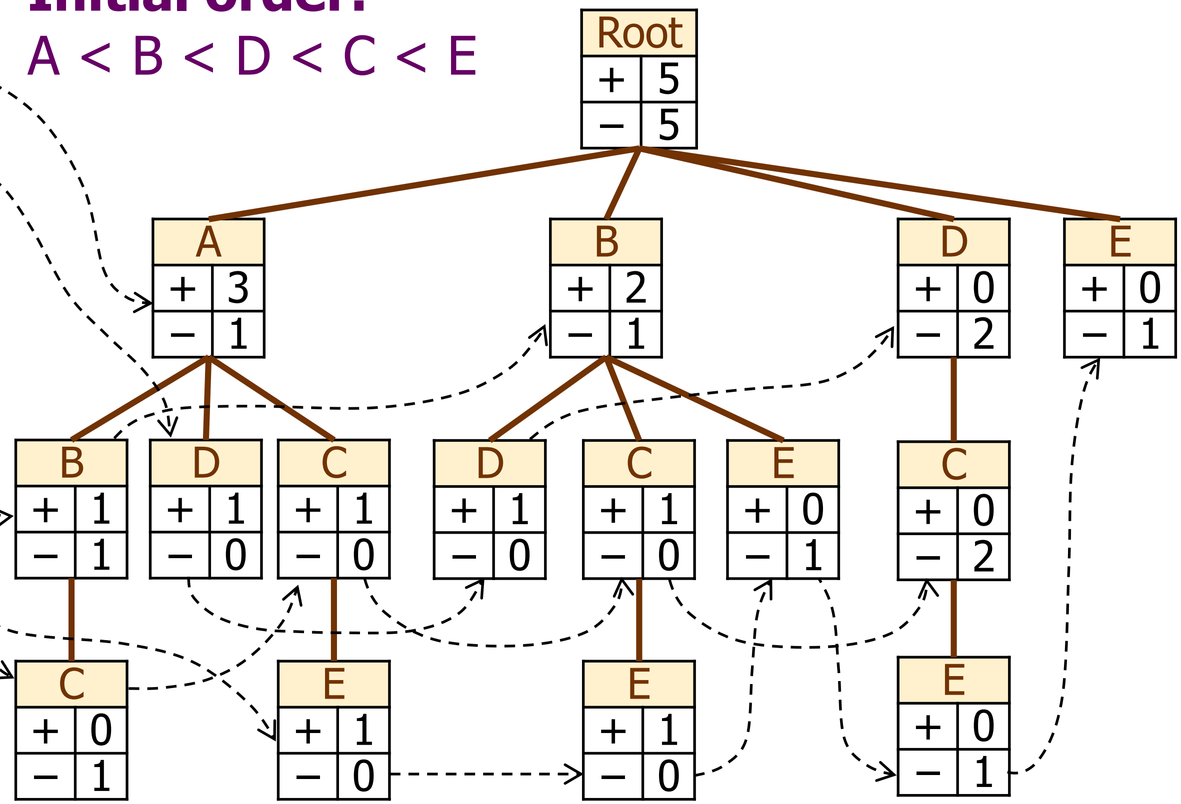
Implementation (1)

- We re-order the items in the header table and conditional transactions while building a FP-tree

Header Table			
Item	+	−	F-score
A	3	1	0.67
B	3	2	0.60
D	2	2	0.44
C	2	3	0.40
E	2	3	0.40

Class	Trans.
+	{A, B}
+	{A, C, E}
+	{A, D}
+	{B, C, E}
+	{B, D}
−	{A, B, C}
−	{B, E}
−	{C, D}
−	{C, D, E}
−	{E}

Initial order:
A < B < D < C < E

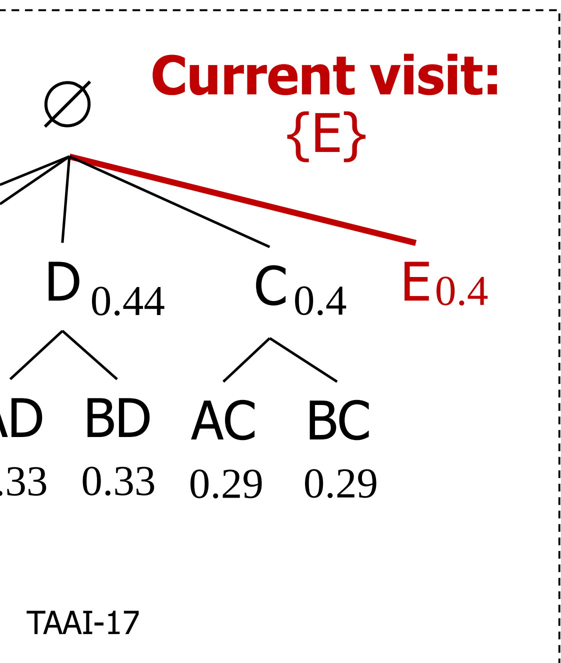


Initial FP-tree

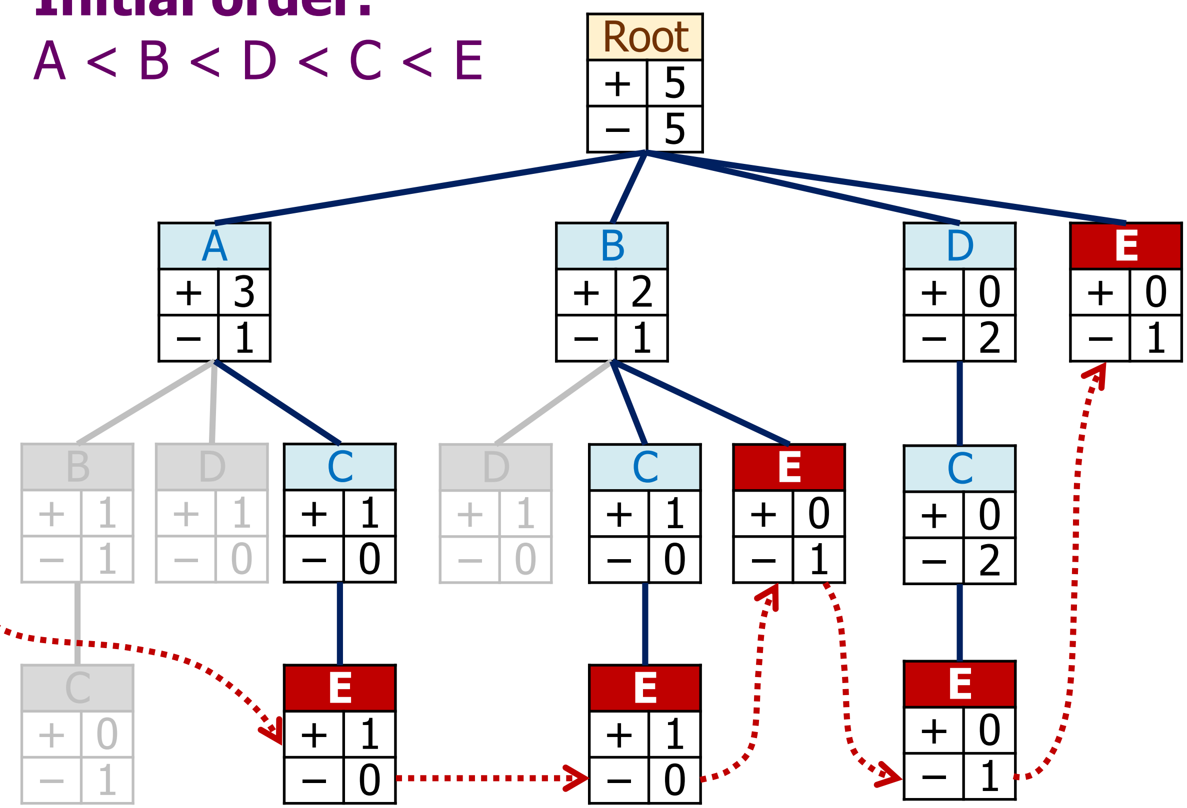
Implementation (2)

- We re-order the items in the header table and conditional transactions while building a FP-tree (cont'd)

Header Table			
Item	+	–	F-score
A	3	1	0.67
B	3	2	0.60
D	2	2	0.44
C	2	3	0.40
E	2	3	0.40



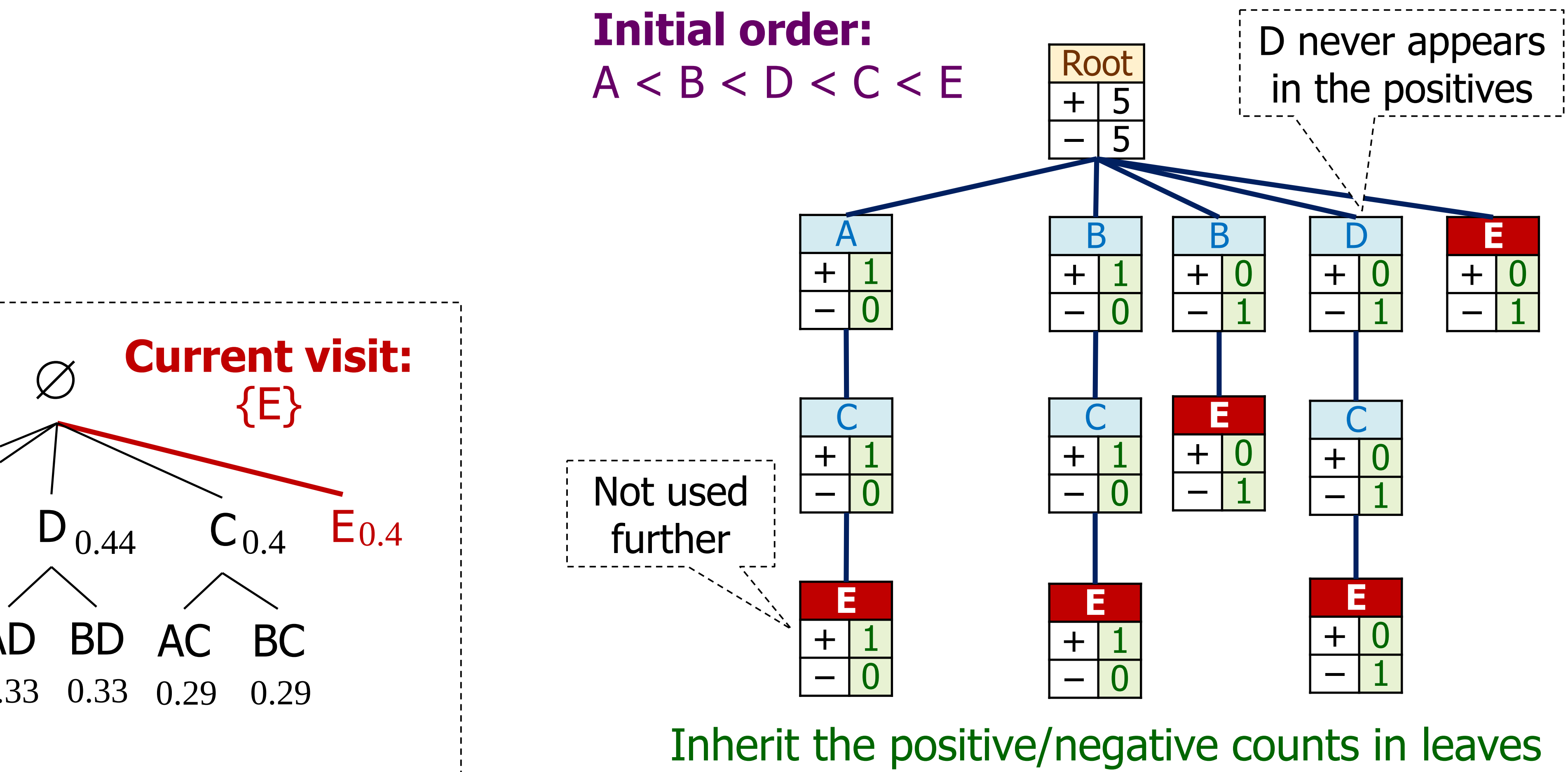
Initial order:
A < B < D < C < E



Initial FP-tree

Implementation (3)

- We re-order the items in the header table and conditional transactions while building a FP-tree (cont'd)



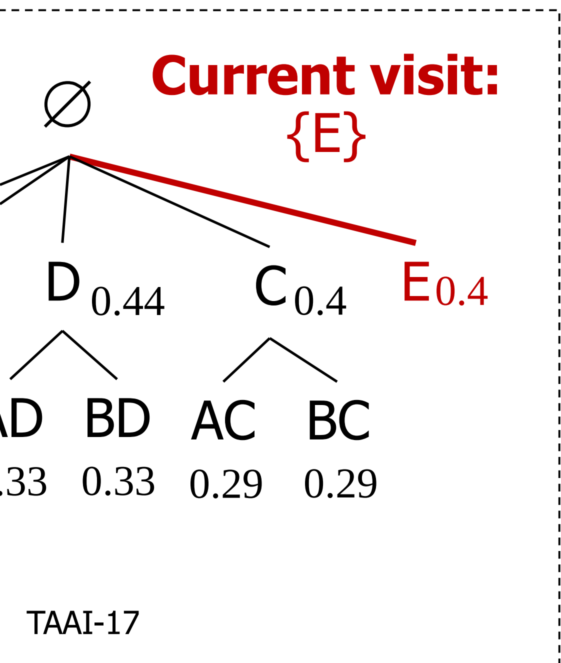
Implementation (4)

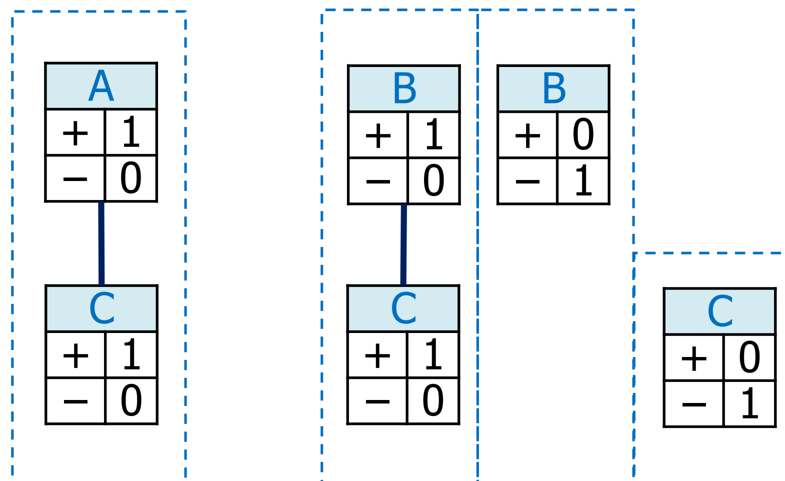
- We re-order the items in the header table and conditional transactions while building a FP-tree (cont'd)

Header Table			
Item	+	–	F-score
A	1	0	0.67
B	1	1	0.60
C	2	1	0.40

Initial order:

A < B < D < C < E





Conditional transactions

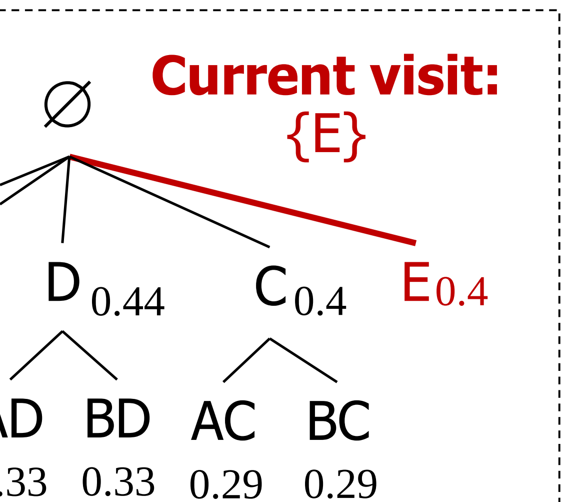
Implementation (5)

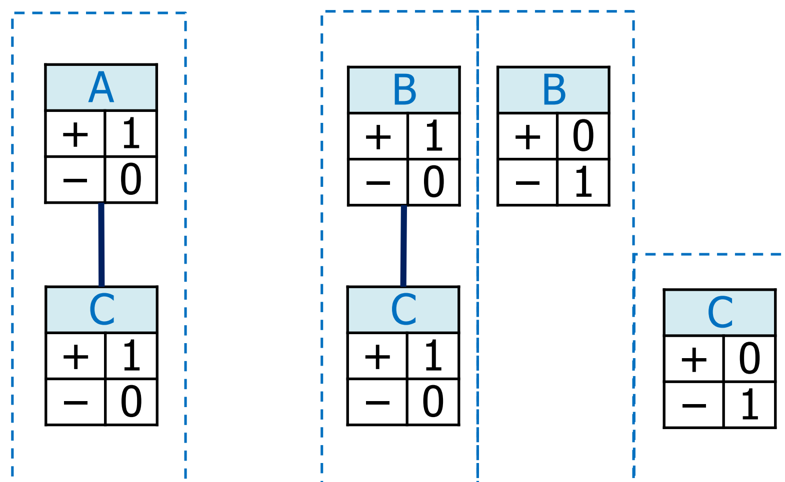
- We re-order the items in the header table and conditional transactions while building a FP-tree (cont'd)

Header Table			
Item	+	−	F-score
A	1	0	0.33
B	1	1	0.29
C	2	1	0.50

Conditional order on {E}:
C < A < B

Compute F-scores





Conditional transactions

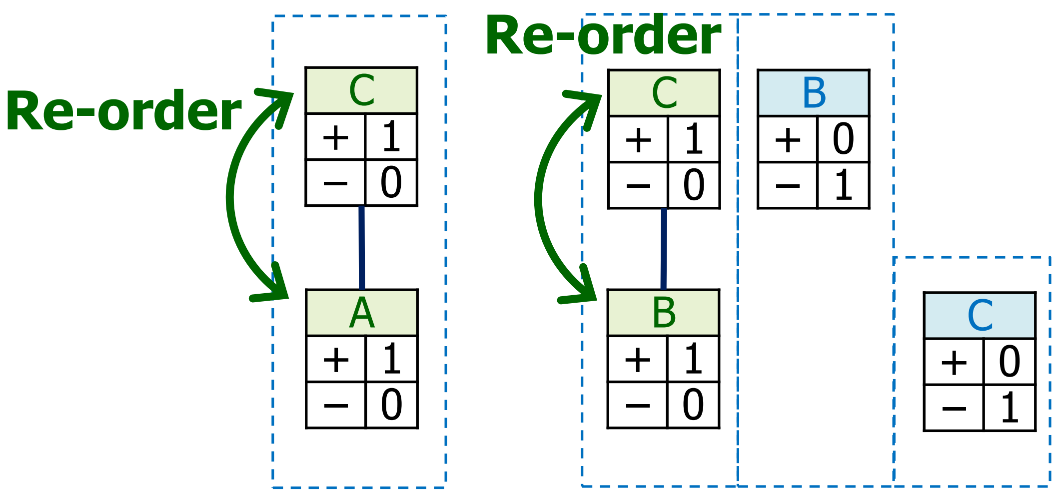
Implementation (6)

- We re-order the items in the header table and conditional transactions while building a FP-tree (cont'd)

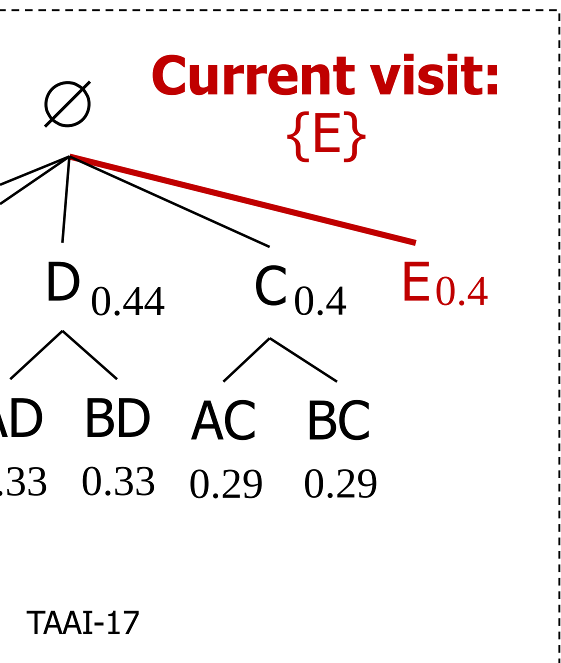
Header Table			
Item	+	−	F-score
C	2	1	0.50
A	1	0	0.33
B	1	1	0.29

Re-order

Conditional order on {E}:
C < A < B



Conditional transactions

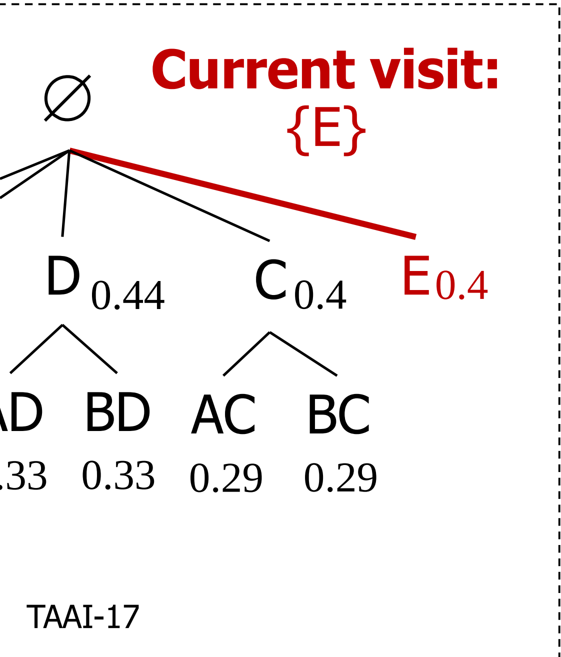


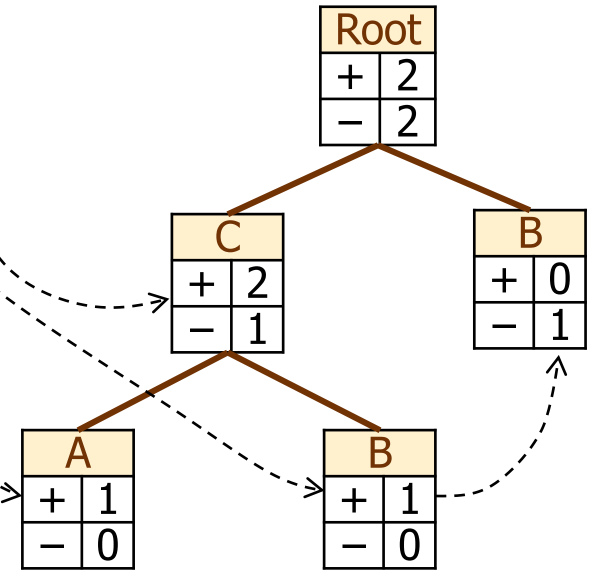
Implementation (7)

- We re-order the items in the header table and conditional transactions while building a FP-tree (cont'd)

Header Table			
Item	+	–	F-score
C	2	1	0.50
A	1	0	0.33
B	1	1	0.29

Conditional order on {E}:
C < A < B





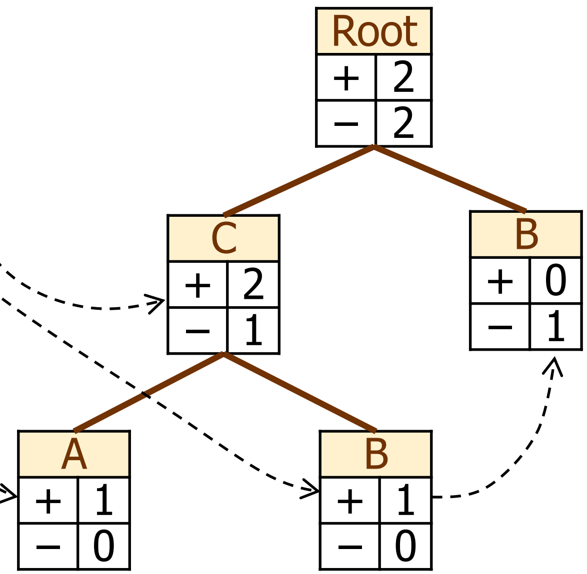
New FP-tree

Implementation (8)

- We re-order the items in the header table and conditional transactions while building a FP-tree (cont'd)

Header Table			
Item	+	−	F-score
C	2	1	0.50
A	1	0	0.33
B	1	1	0.29

Conditional order on {E}**:**
C < A < B



New FP-tree