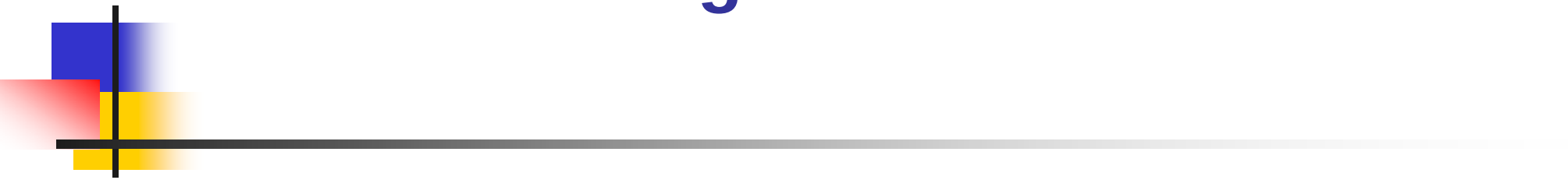
Frequentness-Transition Queries for Distinctive Pattern Mining from Time-Segmented Databases



Shin-ichi Minato
Hokkaido University
Sapporo, Japan

Takeaki Uno
National Inst. of Informatics
Tokyo, Japan

Background

LCM

Fast algorithm for frequent itemset mining.

ZDDs

Compact data structure for manipulating large-scale data.

LCM over ZDDs

[PAKDD2008]

Generates compact ZDDs of large-scale frequent itemsets ready for various post-processing.

Application to sequential databases.

Distinctive pattern mining from time-segmented DBs

This work

Related work

- Mining frequent sequential patterns.
  - Called “sequence mining” or “episode mining.”
    → many previous studies. [Parida2000, Pisanti2003, Wang2004]
- Mining combinatorial patterns with sequential property.
  - **Emerging pattern mining** [Dong1999]
    to find patterns frequent in DB1 but not in DB2.
    → compares two DBs, or two periods of one sequential DB.
  - **Transitional pattern mining** [Wan2007]
    to find patterns with a significant frequency change between two periods of a sequential DB.
    → also finds a time point of the two periods for each pattern.

Differential mining methods are useful but hard to compute.

“Apriori property” no longer holds. → cannot prune search space.

Related work

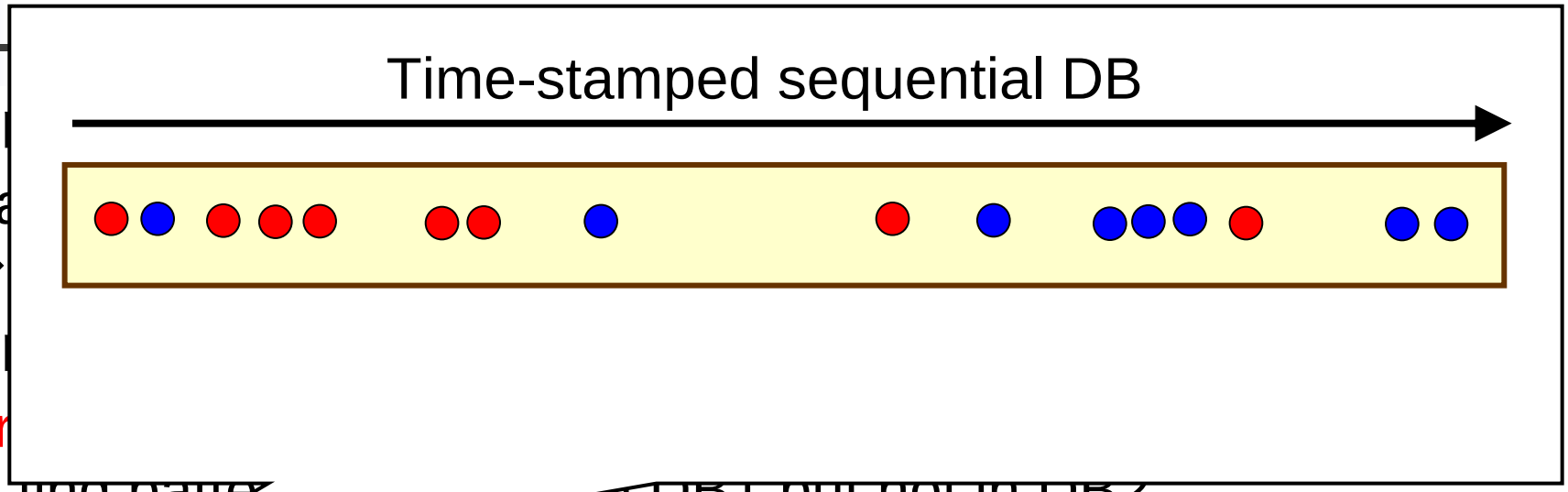
- Mining frequent sequential patterns.
  - Called “sequence mining” or “episode mining.”
    → many previous studies. [Parida2000, Pisanti2003, Wang2004]
- Mining combinatorial patterns with sequential property.
  - **Emerging pattern mining** [Dong1999]
    to find patterns frequent in DB1 but not in DB2.
    → compares two DBs, or two periods of one sequential DB.
  - **Transitional pattern mining** [Wan2007]
    to find patterns with a significant frequency change between two periods of a sequential DB.
    → also finds a time point of the two periods for each pattern.

Differential mining methods are useful but hard to compute.

“Apriori property” no longer holds. → cannot prune search space.

Related work

- Mini
  - Ca
    →
- Mini
  - **Er**
    to find patte
    → compares two DBs, or two periods of one sequential DB.
  - **Transitional pattern mining** [Wan2007]
    to find patterns with a significant frequency change between two periods of a sequential DB.
    → also finds a time point of the two periods for each pattern.

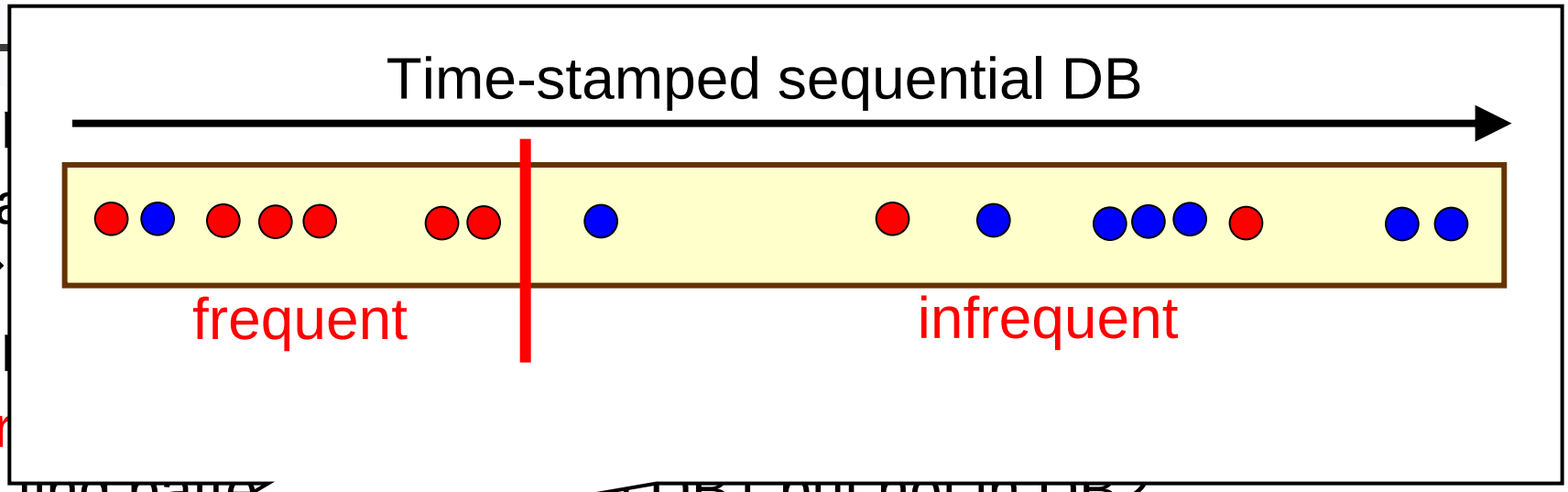


Differential mining methods are useful but hard to compute.

"Apriori property" no longer holds. → cannot prune search space.

Related work

- Mini
  - Ca
    →
- Mini
  - **Er**
    to find patte
    → compares two DBs, or two periods of one sequential DB.
  - **Transitional pattern mining** [Wan2007]
    to find patterns with a significant frequency change between two periods of a sequential DB.
    → also finds a time point of the two periods for each pattern.

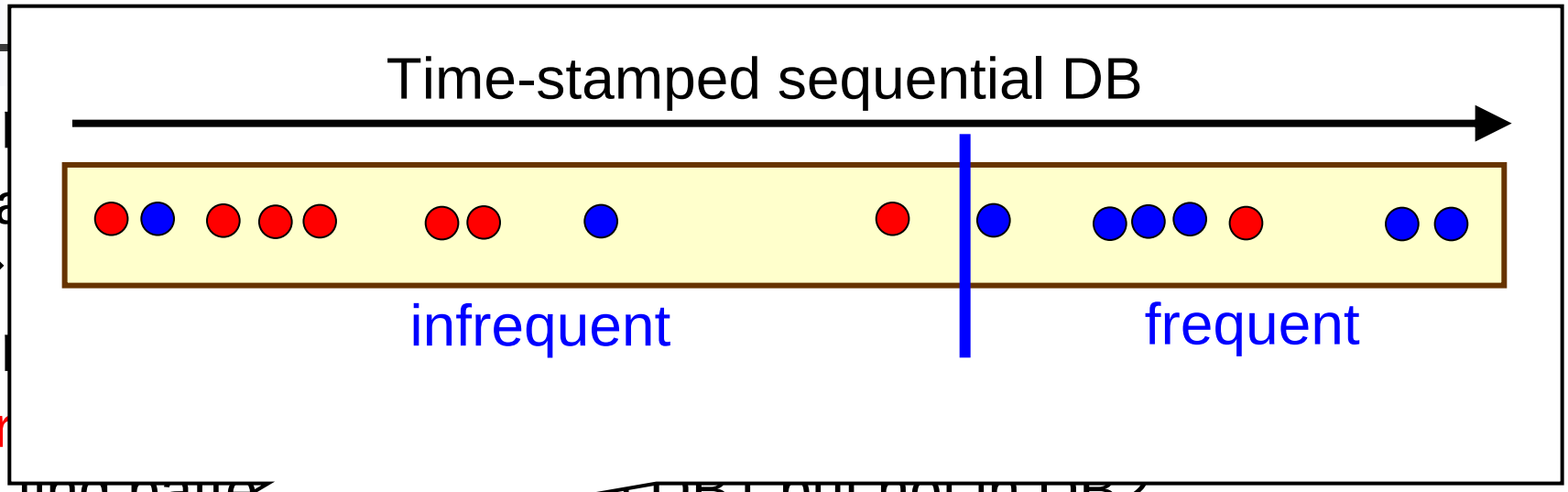


Differential mining methods are useful but hard to compute.

"Apriori property" no longer holds. → cannot prune search space.

Related work

- Mini
  - Ca
    →
- Mini
  - **Er**
    to find patte
    → compares two DBs, or two periods of one sequential DB.
  - **Transitional pattern mining** [Wan2007]
    to find patterns with a significant frequency change between two periods of a sequential DB.
    → also finds a time point of the two periods for each pattern.



Differential mining methods are useful but hard to compute.

"Apriori property" no longer holds. → cannot prune search space.

Related work

- Mining frequent sequential patterns.
  - Called “sequence mining” or “episode mining.”
    → many previous studies. [Parida2000, Pisanti2003, Wang2004]
- Mining combinatorial patterns with sequential property.
  - **Emerging pattern mining** [Dong1999]
    to find patterns frequent in DB1 but not in DB2.
    → compares two DBs, or two periods of one sequential DB.
  - **Transitional pattern mining** [Wan2007]
    to find patterns with a significant frequency change between two periods of a sequential DB.
    → also finds a time point of the two periods for each pattern.

Differential mining methods are useful but hard to compute.

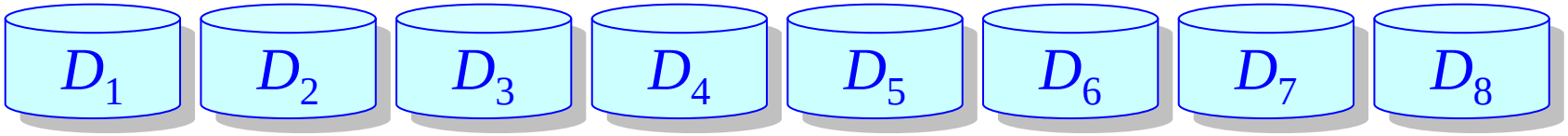
“Apriori property” no longer holds. → cannot prune search space.

Our ideas

- Digitizing time frame of sequential DBs.
  - We consider ***time-segmented DBs***. $(D_1, D_2, \ldots, D_T)$
- Quantization of pattern frequency.
  - We consider ***frequentness sequence*** for each pattern.
    "H"("L"): High (Low) frequency. i.e. "LLLLHHLLLHHHH…"
- Using ZDDs for processing very large-scale patterns.
  - Benefit of digitization and quantization of the model.
- Using regular expression for query language.
  - ***Frequentness-transition query:***
    L*HHHL* means any sequence as "LL…LHHHLL…L"
  - Good trade-off between expressive power and complexity.

Distinctive pattern mining from time-segmented DBs

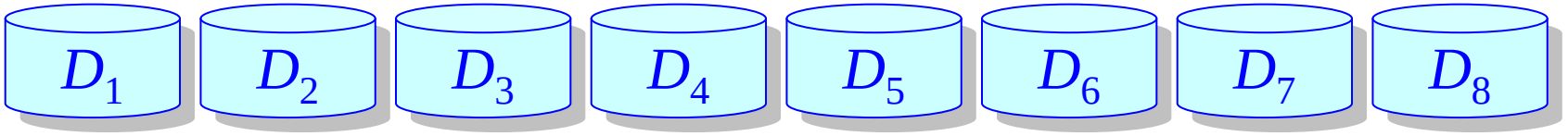
(Daily transaction databases)



Itemset	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8
$\{a\ b\ c\}$	---	---	---	---	Freq.	Freq.	Freq.	Freq.
$\{b\ c\ e\ g\}$	---	---	Freq.	Freq.	Freq.	---	---	---
$\{d\ e\}$	---	---	---	---	---	Freq.	Freq.	Freq.
$\{a\}$	Freq.	Freq.	---	Freq.	Freq.	Freq.	Freq.	Freq.

Distinctive pattern mining from time-segmented DBs

(Daily transaction databases)



Itemset	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8
$\{a\ b\ c\}$	---	---	---	---	Freq.	Freq.	Freq.	Freq.
$\{b\ c\ e\ g\}$	---	---	Freq.	Freq.	Freq.	---	---	---
$\{d\ e\}$	---	---	---	---	---	Freq.	Freq.	Freq.
$\{a\}$	Freq.	Freq.	---	Freq.	Freq.	Freq.	Freq.	Freq.

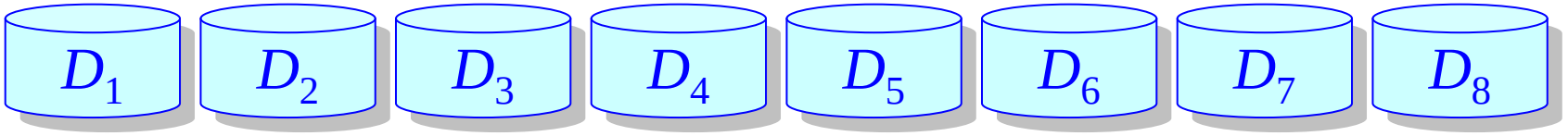
Frequentness-transition query: (Frequent: H, Infrequent: L)

LL*HH*

(We don't mind when it becomes frequent.)

Distinctive pattern mining from time-segmented DBs

(Daily transaction databases)



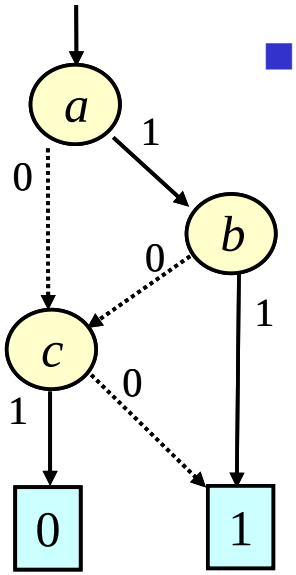
Itemset	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8
$\{a\ b\ c\}$	---	---	---	---	Freq.	Freq.	Freq.	Freq.
$\{b\ c\ e\ g\}$	---	---	Freq.	Freq.	Freq.	---	---	---
$\{d\ e\}$	---	---	---	---	---	Freq.	Freq.	Freq.
$\{a\}$	Freq.	Freq.	---	Freq.	Freq.	Freq.	Freq.	Freq.

Frequentness-transition query: (Frequent: H, Infrequent: L)

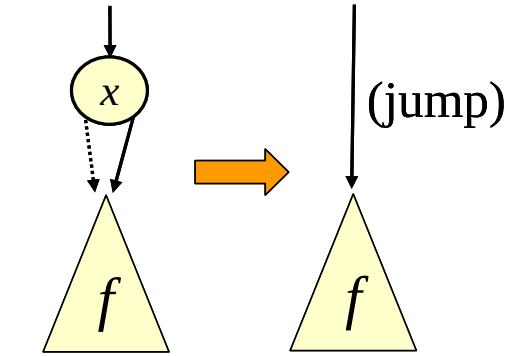
L*HHHL*

(We don't mind when it becomes frequent.)

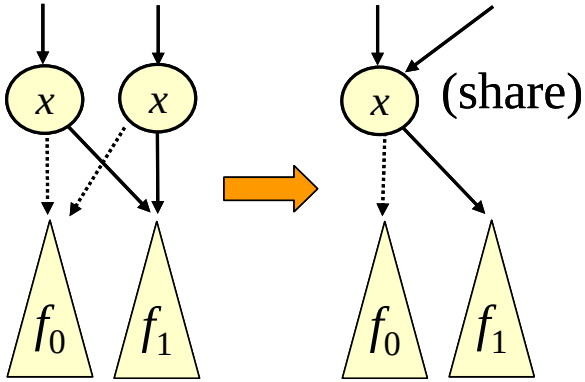
BDD (Binary Decision Diagram) [Bryant86]



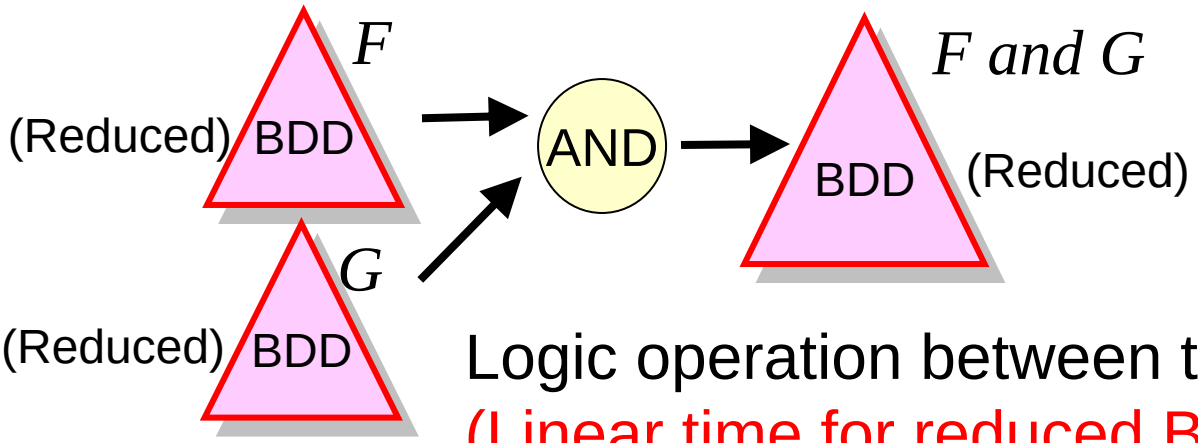
- Graph representation of Boolean function data.
  - Compact & Canonical form for Boolean functions under a fixed variable ordering.



Node elimination rule



Node sharing rule



Logic operation between two reduced BDDs.
(Linear time for reduced BDD size.)

Boolean function and combinatorial itemset

a	b	c	F	
0	0	0	0	
1	0	0	0	
0	1	0	0	
1	1	0	1	→ ab
0	0	1	1	→ c
1	0	1	1	→ ac
0	1	1	0	
1	1	1	0	

Boolean function:

$F = (a\ b\ {\sim}c)\ \mathrm{V}\ ({\sim}b\ c)$

Combinatorial itemset:

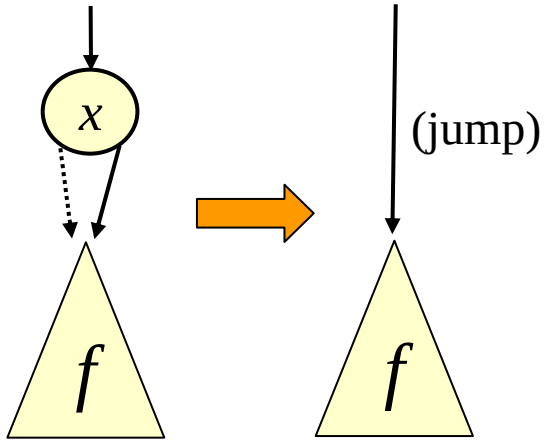
$F = \{ab, ac, c\}$

(customer's choice)

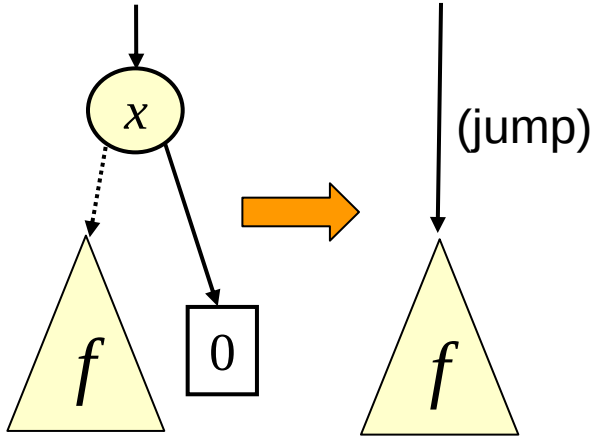
- Operations of combinatorial itemsets can be done by BDD-based logic operations.
  - Union of sets → logical OR
  - Intersection of sets → logical AND
  - Complement set → logical NOT

Zero-suppressed BDD (ZDD) [Minato93]

- A variant of BDDs for **combinatorial itemets.**
- Uses a new reduction rule different from ordinary BDDs.
  - Eliminate all nodes whose “1-edge” directly points to 0-terminal.
  - Share equivalent nodes as well as ordinary BDDs.
- If an item x does not appear in any itemset, the ZDD node of x is automatically eliminated.
  - When average appearance ratio of each item is 1%, ZDDs are more compact than ordinary BDDs, up to 100 times.



Ordinary BDD reduction



Zero-suppressed reduction

Frequent itemset mining

- Basic and well-known problem in database analysis.

Record ID	Tuple
1	*a b c*
2	*a b*
3	*a b c*
4	*b c*
5	*a b*
6	*a b c*
7	*c*
8	*a b c*
9	*a b c*
10	*a b*
11	*b c*

Frequency threshold = 10 → $\{ b \}$

Frequency threshold = 8 → $\{ ab, a, b, c \}$

Frequency threshold = 7 → $\{ ab, bc, a, b, c \}$

Frequency threshold = 5 → $\{abc, ab, bc, ac, a, b, c \}$

Frequency threshold = 1 → $\{abc, ab, bc, ac, a, b, c \}$

LCM over ZDDs [Minato et al. 2008]

- ***LCM:*** *[Uno2003]*
  Output-linear time algorithm of frequent itemset mining.
- ***ZDD:*** *[Minato1993]*
  A compact graph-based representation
  for very large-scale sets of combinations.

Record ID	Tuple
1	*a b c*
2	*a b*
3	*a b c*
4	*b c*
5	*a b*
6	*a b c*
7	*c*
8	*a b c*
9	*a b c*
10	*a b*
11	*b c*

LCM over ZDDs

Freq. thres. $\alpha = 7$

$\{ ab, bc, a, b, c \}$

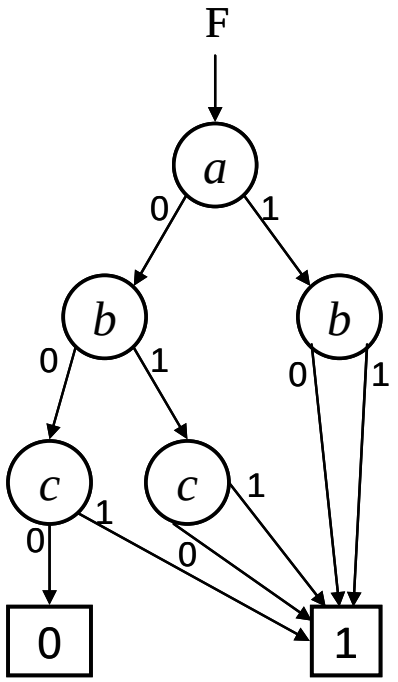


Table 2. Co… CM ov… prev…

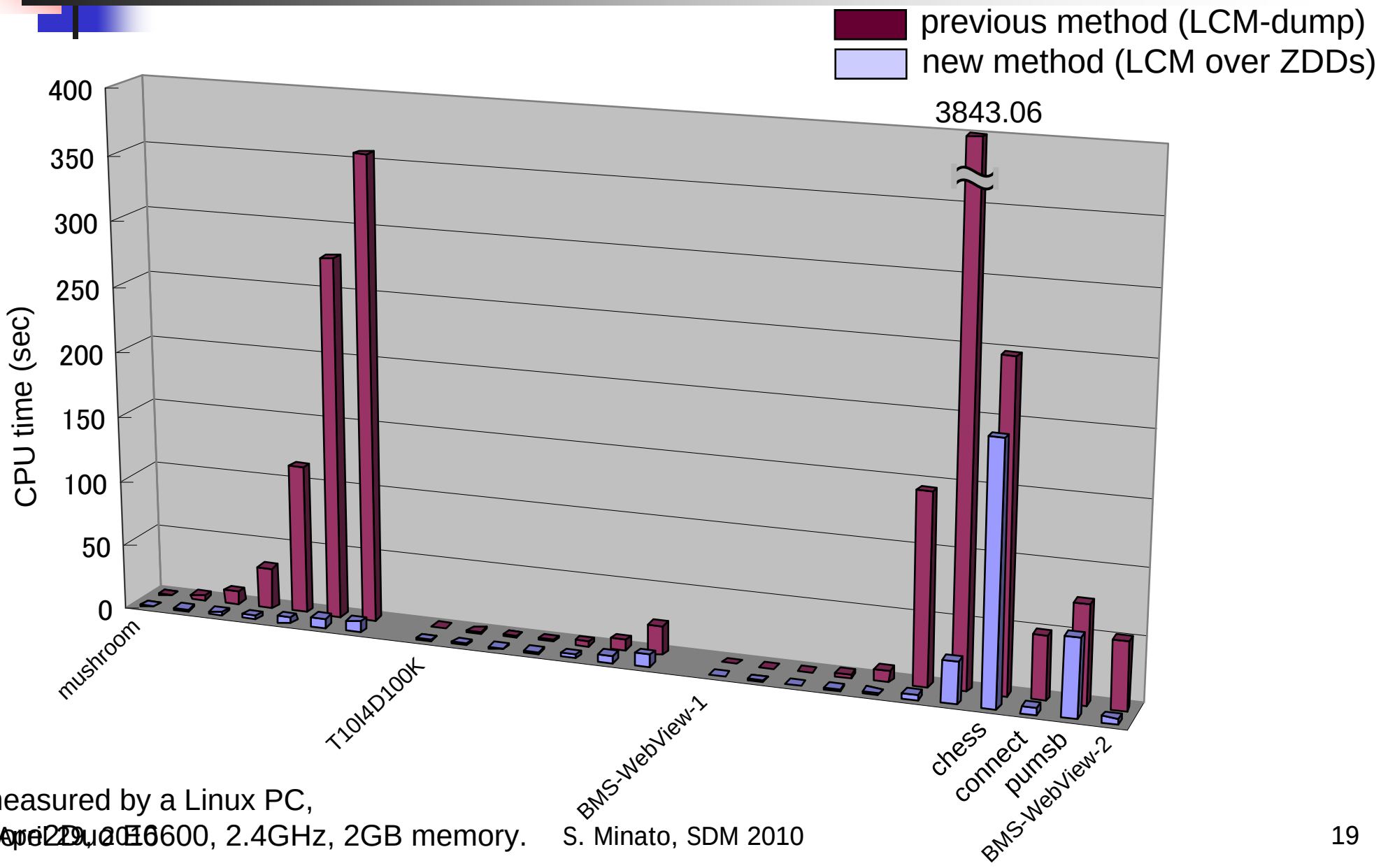
solutions

LCM over ZDDs

Original LCM

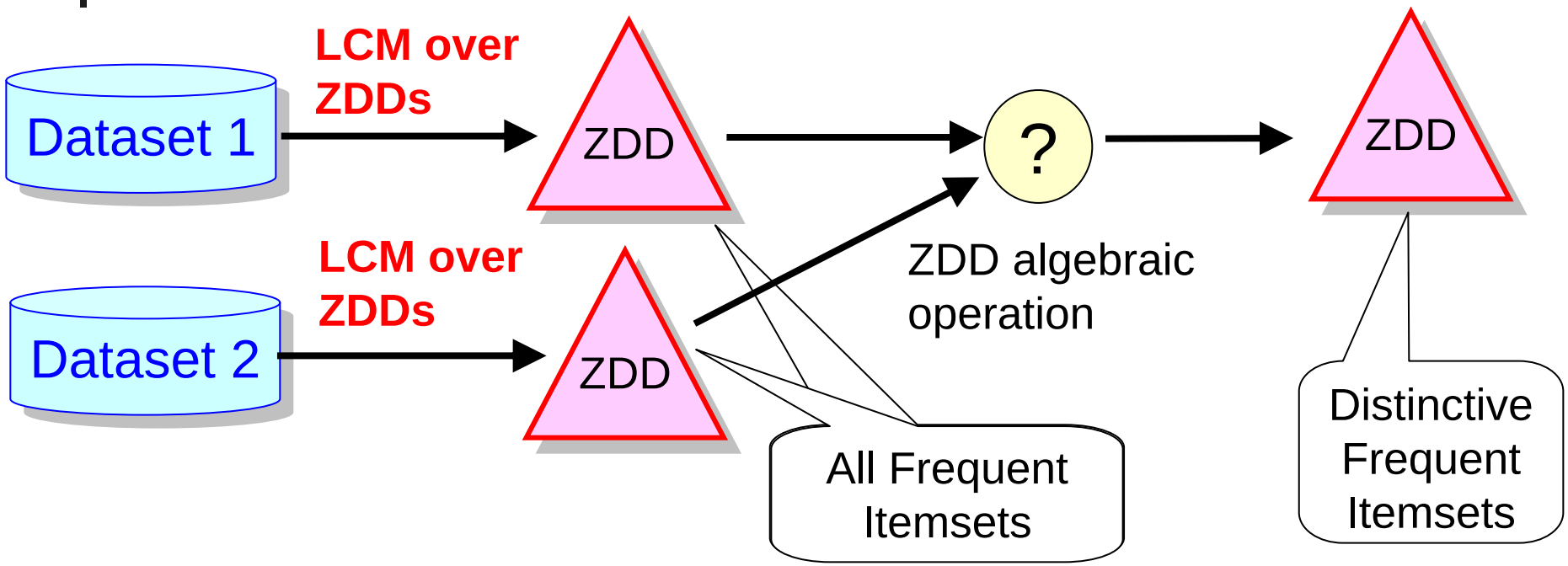
Dataset name: min. support	# itemsets	CMov… |ZBDD|	CMov… Time(s)	…t Time(s)	LC… Time(s)	…-growth Time(s)
mushroom: 1,000	123,287	760	0.50	0.49	0.64	1.78
500	1,442,504	2,254	1.32	1.30	3.29	3.49
300	5,259,786	4,412	2.25	2.22	9.96	5.11
200	18,094,822	6,383	3.21	3.13	31.63	6.24
100	66,076,586	11,584	5.06	4.87	114.21	6.72
70	153,336,056	14,307	7.16	7.08	277.15	6.97
50	198,169,866	17,830	8.17	7.86	357.27	6.39
T10I4D100K: 100	27,533	8,482	0.85	0.85	0.86	209.82
50	53,386	16,872	0.97	0.92	0.98	242.31
20	129,876	58,413	1.13	1.08	1.20	290.78
10	411,366	173,422	1.55	1.36	1.64	332.22
5	1,923,260	628,491	2.86	2.08	3.54	370.54
3	6,169,854	1,576,184	5.20	3.15	8.14	386.72
2	19,561,715	3,270,977	9.68	5.09	22.66	384.60
BMS-WebView-1: 50	8,192	3,415	0.11	0.11	0.12	29.46
40	48,544	10,755	0.18	0.18	0.22	48.54
36	461,522	28,964	0.49	0.42	0.98	67.16
35	1,177,608	38,164	0.80	0.69	2.24	73.64
34	4,849,466	49,377	1.30	1.07	8.58	83.36
33	69,417,074	59,119	3.53	3.13	144.98	91.62
32	1,531,980,298	71,574	31.90	29.73	3,843.06	92.47
chess: 1,000	29,442,849	53,338	197.58	197.10	248.18	1,500.78
connect: 40,000	23,981,184	3,067	5.42	5.40	49.21	212.84
pumsb: 32,000	7,733,322	5,443	60.65	60.42	75.29	4,189.09
BMS-WebView-2: 5	26,946,004	353,091	4.84	3.62	51.28	118.01

Performance of LCM over ZDDs



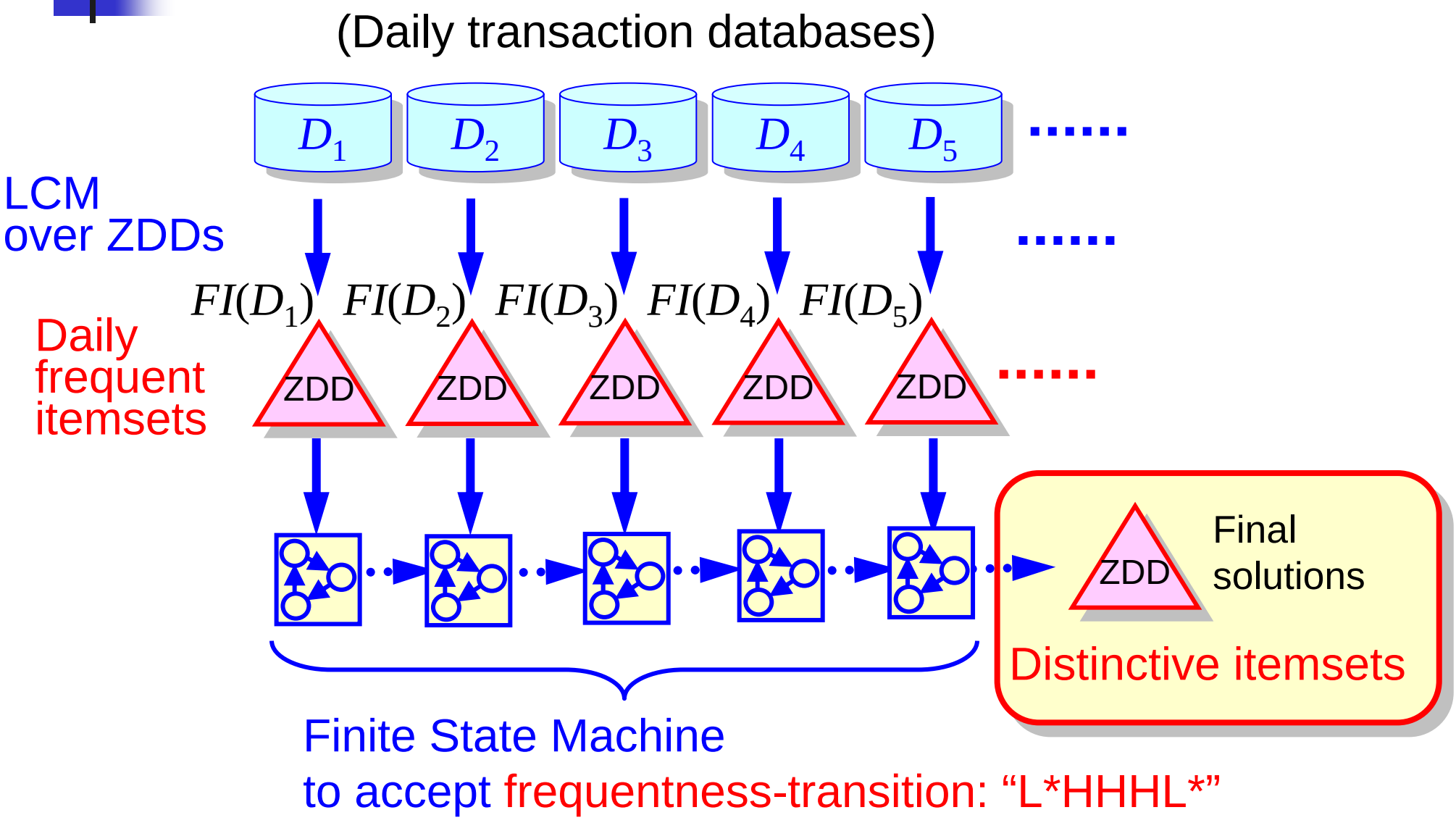
measured by a Linux PC,
Core2Duo E6600, 2.4GHz, 2GB memory.

Post Processing after LCM over ZDDs

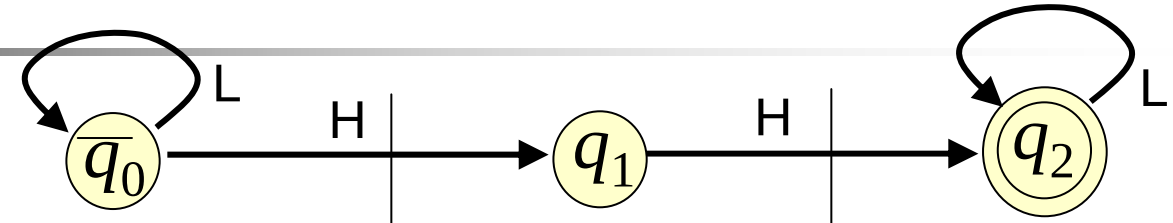


- We can extract distinctive itemsets by comparing frequent itemsets for multiple sets of databases.
  - Various ZDD algebraic operations can be used for the comparison of the huge number of frequent itemsets.

Distinctive itemset mining for time-segmented DBs



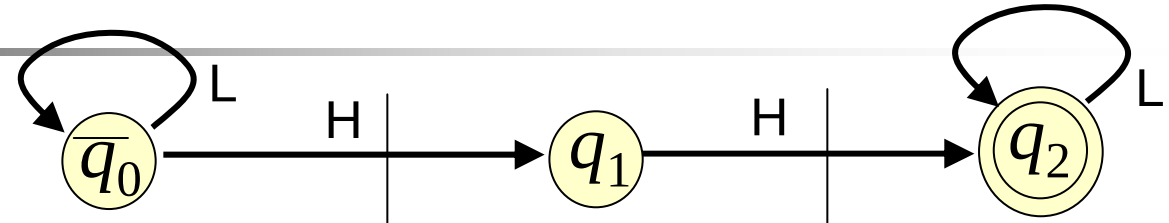
Behavior of FSM for accepting L*HHL*



$FI(D_i)$	{ }	{ }	{ }
{ λ , a, c, d, ac, ad, cd, acd }			
{ λ , a, b, ab }			
{ λ , a, b, c, d ab, bc, bd, cd, bcd }			
{ λ , b, c, d, bc, bd, cd, bcd }			
{ λ , a, b, c, d, e, bc, cd }			

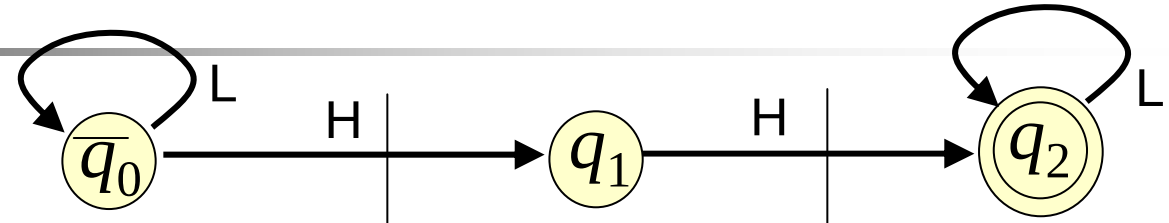
Final result

Behavior of FSM for accepting L*HHL*



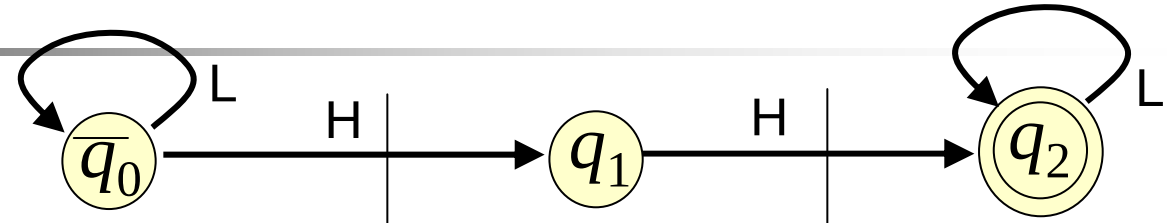
$FI(D_i)$	{ }	{ }	{ }
——			
{ λ , a, c, d, ac, ad, cd, acd }	{ λ , a, c, d, ac, ad, cd, acd }	{ λ , a, c, d, ac, ad, cd, acd }	{ }
{ λ , a, b, ab }			
{ λ , a, b, c, d ab, bc, bd, cd, bcd }			
{ λ , b, c, d, bc, bd, cd, bcd }			
{ λ , a, b, c, d, e, bc, cd }			

Behavior of FSM for accepting L*HHL*



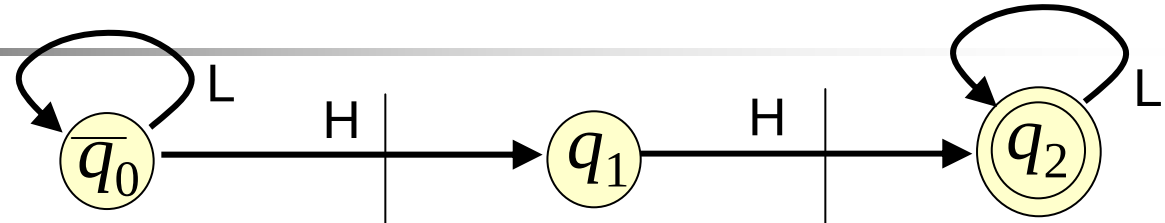
$FI(D_i)$	{ }	{ }	{ }
{ λ , a, c, d, ac, ad, cd, acd }	{ λ , a, c, d, ac, ad, cd, acd }	{ λ , a, c, d, ac, ad, cd, acd }	{ }
{ λ , a, b, ab }	{ λ , a, b, c, d, ab, ac, ad, cd, acd }	{b, ab}	{λ , a}
{ λ , a, b, c, d ab, bc, bd, cd, bcd }			
{ λ , b, c, d, bc, bd, cd, bcd }			
{ λ , a, b, c, d, e, bc, cd }			

Behavior of FSM for accepting L*HHL*



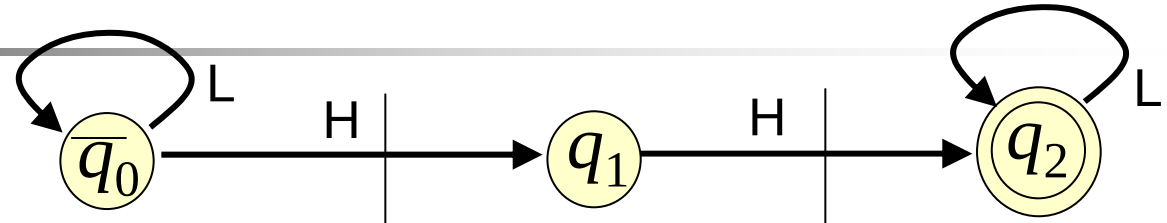
$FI(D_i)$	$\{\ \}$	$\{\ \}$	$\{\ \}$
$\{\lambda, a, c, d, ac, ad, cd, acd\}$	$\{\lambda, a, c, d, ac, ad, cd, acd\}$	$\{\lambda, a, c, d, ac, ad, cd, acd\}$	$\{\ \}$
$\{\lambda, a, b, ab\}$	$\{\lambda, a, b, c, d, ab, ac, ad, cd, acd\}$	$\{b, ab\}$	$\{\lambda, a\}$
$\{\lambda, a, b, c, d, ab, bc, bd, cd, bcd\}$	$\{\lambda, a, b, c, d, ab, ac, ad, bc, bd, cd, acd, bcd\}$	$\{bc, bd, bcd\}$	$\{b, ab\}$
$\{\lambda, b, c, d, bc, bd, cd, bcd\}$			
$\{\lambda, a, b, c, d, e, bc, cd\}$			

Behavior of FSM for accepting L*HHL*



$FI(D_i)$	$\{\ \}$	$\{\ \}$	$\{\ \}$
$\{\lambda, a, c, d, ac, ad, cd, acd\}$	$\{\lambda, a, c, d, ac, ad, cd, acd\}$	$\{\lambda, a, c, d, ac, ad, cd, acd\}$	$\{\ \}$
$\{\lambda, a, b, ab\}$	$\{\lambda, a, b, c, d, ab, ac, ad, cd, acd\}$	$\{b, ab\}$	$\{\lambda, a\}$
$\{\lambda, a, b, c, d, ab, bc, bd, cd, bcd\}$	$\{\lambda, a, b, c, d, ab, ac, ad, bc, bd, cd, acd, bcd\}$	$\{bc, bd, bcd\}$	$\{b, ab\}$
$\{\lambda, b, c, d, bc, bd, cd, bcd\}$	$\{\lambda, a, b, c, d, ab, ac, ad, bc, bd, cd, acd, bcd\}$	$\{\lambda\}$	$\{b, ab, bc, bd, bcd\}$
$\{\lambda, a, b, c, d, e, bc, cd\}$			

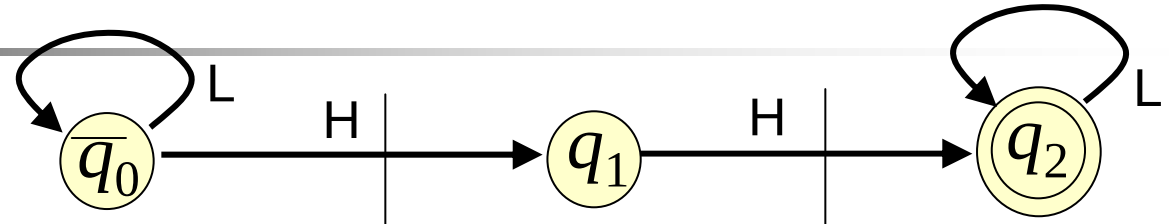
Behavior of FSM for accepting L*HHL*



$FI(D_i)$	$\{\ \}$	$\{\ \}$	$\{\ \}$
$\{\lambda, a, c, d, ac, ad, cd, acd\}$	$\{\lambda, a, c, d, ac, ad, cd, acd\}$	$\{\lambda, a, c, d, ac, ad, cd, acd\}$	$\{\ \}$
$\{\lambda, a, b, ab\}$	$\{\lambda, a, b, c, d, ab, ac, ad, cd, acd\}$	$\{b, ab\}$	$\{\lambda, a\}$
$\{\lambda, a, b, c, d, ab, bc, bd, cd, bcd\}$	$\{\lambda, a, b, c, d, ab, ac, ad, bc, bd, cd, acd, bcd\}$	$\{bc, bd, bcd\}$	$\{b, ab\}$
$\{\lambda, b, c, d, bc, bd, cd, bcd\}$	$\{\lambda, a, b, c, d, ab, ac, ad, bc, bd, cd, acd, bcd\}$	$\{\lambda\}$	$\{b, ab, bc, bd, bcd\}$
$\{\lambda, a, b, c, d, e, bc, cd\}$	$\{\lambda, a, b, c, d, e, ab, ac, ad, bc, bd, cd, acd, bcd\}$	$\{e\}$	$\{ab, bd, bcd\}$

Final result

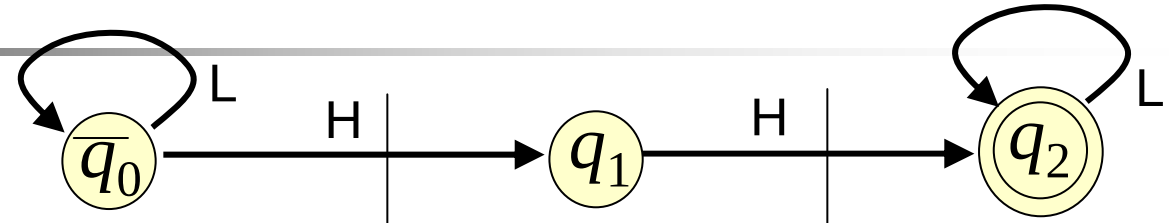
Behavior of FSM for accepting L*HHL*



	$FI(D_i)$	q_0	q_1	q_2
	——	{ }	{ }	{ }
L	{ λ , a, c, d, ac, ad, cd, acd }	{ λ , a, c, d, ac, ad, cd, acd }	{ λ , a, c, d, ac, ad, cd, acd }	{ }
H	{ λ , a, b, ab }	{ λ , a, b, c, d, ab, ac, ad, cd, acd }	{b, ab}	{λ , a}
H	{ λ , a, b, c, d ab, bc, bd, cd, bcd }	{ λ , a, b, c, d, ab, ac, ad, bc, bd, cd, acd, bcd }	{bc, bd, bcd}	{b, ab}
L	{ λ , b, c, d, bc, bd, cd, bcd }	{ λ , a, b, c, d, ab, ac, ad, bc, bd, cd, acd, bcd }	{λ}	{ b, ab, bc, bd, bcd }
L	{ λ , a, b, c, d, e, bc, cd }	{ λ , a, b, c, d, e, ab, ac, ad, bc, bd, cd, acd, bcd }	{e}	{ab, bd, bcd}

Final result

Behavior of FSM for accepting L*HHL*

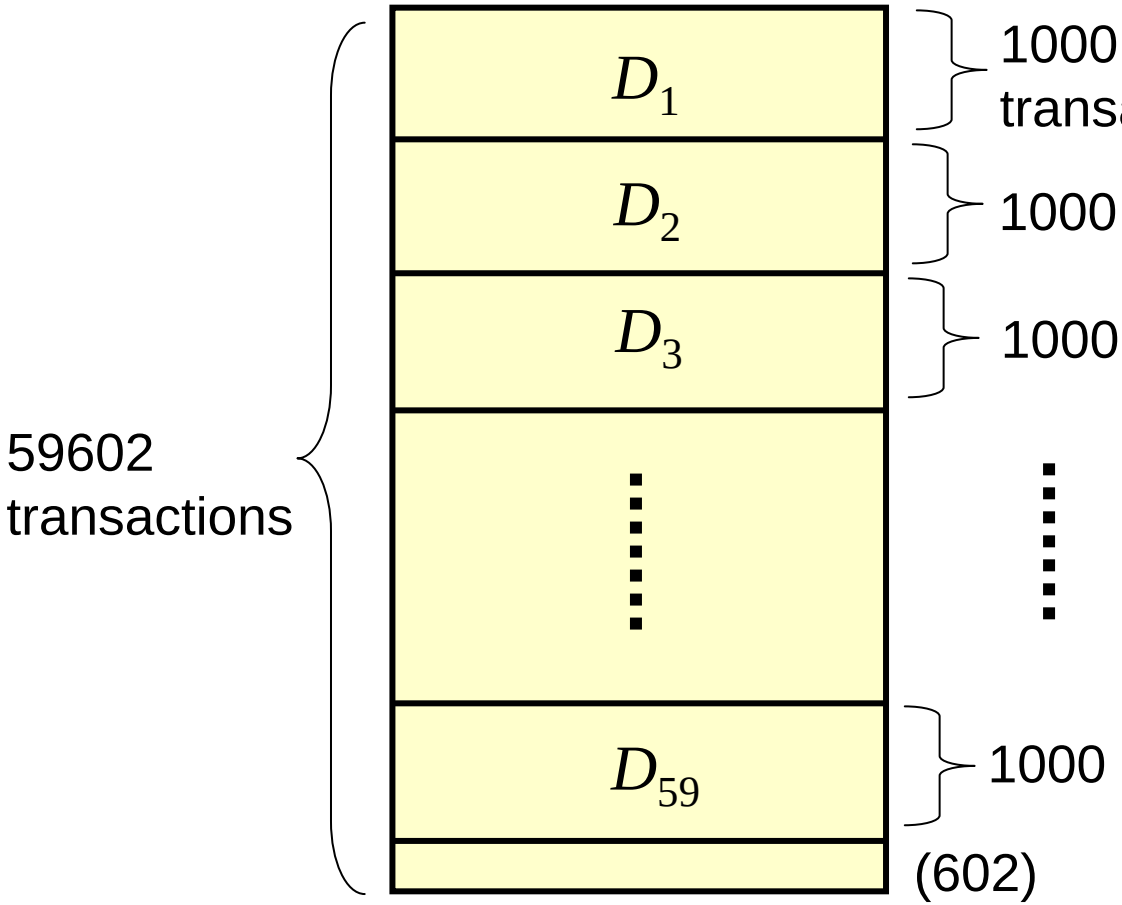


	$FI(D_i)$	q_0	q_1	q_2
	——	{ }	{ }	{ }
L	{ λ , a, c, d, ac, ad, cd, acd }	{ λ , a, c, d, ac, ad, cd, acd }	{ λ , a, c, d, ac, ad, cd, acd }	{ }
L	{ λ , a, b, ab }	{ λ , a, b, c, d, ab, ac, ad, cd, acd }	{b, ab}	{λ , a}
H	{ λ , a, b, c, d ab, bc, bd, cd, bcd }	{ λ , a, b, c, d, ab, ac, ad, bc, bd, cd, acd, bcd }	{bc, bd, bcd}	{b, ab}
H	{ λ , b, c, d, bc, bd, cd, bcd }	{ λ , a, b, c, d, ab, ac, ad, bc, bd, cd, acd, bcd }	{λ}	{ b, ab, bc, bd, bcd }
L	{ λ , a, b, c, d, e, bc, cd }	{ λ , a, b, c, d, e, ab, ac, ad, bc, bd, cd, acd, bcd }	{e}	{ab, bd, bcd}

Final result

Dataset used in our experiment

Dataset: BMS-WebView-1 (divided into 59 segments)



- Access log of an online shopping site.
- Each transaction: a set of web pages viewed by one customer.
- Original dataset has no information on sequential order.
- We artificially generate a time-segmented databases.

Performance Evaluation

Table 3: Experimental results for benchmark data

Dataset (#Segment)	ρ (%)	Freq. trans. query X	#Itemset $FI(\mathcal{D})$	#Itemset solutions	Time (sec)
BMS-WebView-1 (59 segments)	0.5	L*HHHL*	$7.21 \cdot 10^{16}$	37	0.40
	0.4	L*HHHL*	$7.14 \cdot 10^{44}$	852	4.51
	0.3	L*HHHL*	$1.18 \cdot 10^{46}$	$3.57 \cdot 10^{44}$	42.00
	0.5	HH*LL*HH*	$7.21 \cdot 10^{16}$	7	0.40
	0.4	HH*LL*HH*	$7.14 \cdot 10^{44}$	6	4.41
	0.3	HH*LL*HH*	$1.18 \cdot 10^{46}$	19	42.90
BMS-WebView-2 (77 segments)	0.4	L*HHHL*	666,654	300	1.75
	0.3	L*HHHL*	9,236,264	1,493	2.69
	0.2	L*HHHL*	$1.44 \cdot 10^{17}$	38,895	7.04

(2.4GHz Core2Duo PC, 2 GB mem., SuSE 10, GNU C++)

852 distinctive itemsets are extracted from **713,623,846,352,979,940,529,143,133,451,627,984,798,184,096 (= 7.14 x 10^{44})** of frequent itemsets **in 4.51 sec**.

Confirmation of the experimental result

Dataset: BMS-WebView-1

L*HHHL*

α = 0.4 %

852 distinctive itemsets are extracted.

{18631, 18643} is frequent only in: D_{50}, D_{51}, D_{52}

{46293, 46285, 46281} is frequent only in: D_6, D_7, D_8

- We discovered local events only seen in this time segments. (very difficult to find by other methods.)
  → A kind of knowledge discovery.

Conclusion

- We presented a new mining problem: ***"Frequentness-transitional pattern mining."***
  - Regular expressions for specifying transitional queries such as L*HHHL*, HH*LL*HH*, etc.
  - We may use ternary quantization as: L*MMHHHMML*.
- In this problem, closed/maximal itemset mining techniques are not effective.
  - Frequentness-transitional patterns are **not always closed/maximal.** (→ we have to check all frequent itemsets.)
  - ZDD-based compression is effective for manipulating large-scale itemsets by set operations.
- Our new mining method for time-segmented databases is very powerful and will have various applications to real-life problems.