Specialised vs Declarative Data Mining

Software Testing Applications

Nadjib Lazaar, CNRS, University of Montpellier

Join works with: M. Maamar, Y. Lebbah, S. Loudni, C. Bessiere, et. al.

SIMULA, Oslo, 11 oct. 2018

DATA MINING

DATA MINING

- **Data Mining (DM)** or Knowledge Discovery in Databases (KDD) revolves around the investigation and creation of knowledge, processes, algorithms, and the mechanisms for **retrieving potential knowledge** from **data collections**.

DATA MINING

- **Data Mining (DM)** or Knowledge Discovery in Databases (KDD) revolves around the investigation and creation of knowledge, processes, algorithms, and the mechanisms for **retrieving potential knowledge** from **data collections**.

Mining on:

- Itemsets (Finding itemsets from a collection of transactions)

DATA MINING

- **Data Mining (DM)** or Knowledge Discovery in Databases (KDD) revolves around the investigation and creation of knowledge, processes, algorithms, and the mechanisms for **retrieving potential knowledge** from **data collections**.

Mining on:

- Itemsets (Finding itemsets from a collection of transactions)
- Sequences (Finding subsequences from collection of sequences)

DATA MINING

- **Data Mining (DM)** or Knowledge Discovery in Databases (KDD) revolves around the investigation and creation of knowledge, processes, algorithms, and the mechanisms for **retrieving potential knowledge** from **data collections**.

Mining on:

- Itemsets (Finding itemsets from a collection of transactions)
- Sequences (Finding subsequences from collection of sequences)
- Graphs (Finding subgraphs from collection of graphs)

DATA MINING

- **Data Mining (DM)** or Knowledge Discovery in Databases (KDD) revolves around the investigation and creation of knowledge, processes, algorithms, and the mechanisms for **retrieving potential knowledge** from **data collections**.

Mining on:

- Itemsets (Finding itemsets from a collection of transactions)
- Sequences (Finding subsequences from collection of sequences)
- Graphs (Finding subgraphs from collection of graphs)
- Tree, Geometric structures…

DATA MINING APPLICATIONS

DATA MINING APPLICATIONS

- Market Basket Analysis [Agrawal93]

DATA MINING APPLICATIONS

- Market Basket Analysis [Agrawal93]
- Future Healthcare
  - Great potential to improve health systems [Obenshain04]

DATA MINING APPLICATIONS

- Market Basket Analysis [Agrawal93]
- Future Healthcare
  - Great potential to improve health systems [Obenshain04]
- Education
  - Knowledge from data educational environments [Scheuer12]

DATA MINING APPLICATIONS

- Market Basket Analysis [Agrawal93]
- Future Healthcare
  - Great potential to improve health systems [Obenshain04]
- Education
  - Knowledge from data educational environments [Scheuer12]
- Fraud and Intrusion detection [Wang10] [Lee98]

DATA MINING APPLICATIONS

- Market Basket Analysis [Agrawal93]
- Future Healthcare
  - Great potential to improve health systems [Obenshain04]
- Education
  - Knowledge from data educational environments [Scheuer12]
- Fraud and Intrusion detection [Wang10] [Lee98]
- Lie detection and Criminal Investigation [Chen04]

DATA MINING APPLICATIONS

- Market Basket Analysis [Agrawal93]
- Future Healthcare
  - Great potential to improve health systems [Obenshain04]
- Education
  - Knowledge from data educational environments [Scheuer12]
- Fraud and Intrusion detection [Wang10] [Lee98]
- Lie detection and Criminal Investigation [Chen04]
- Bio Informatics [Hoffman97]

DATA MINING APPLICATIONS

- Market Basket Analysis [Agrawal93]
- Future Healthcare
  - Great potential to improve health systems [Obenshain04]
- Education
  - Knowledge from data educational environments [Scheuer12]
- Fraud and Intrusion detection [Wang10] [Lee98]
- Lie detection and Criminal Investigation [Chen04]
- Bio Informatics [Hoffman97]
- …

DATA MINING APPLICATIONS

Inputs	Mining process	Outputs

DATA MINING APPLICATIONS

Inputs	Mining process	Outputs
- shopping basket	Marketing	- Custumers behavior - Frequent products

DATA MINING APPLICATIONS

Inputs	Mining process	Outputs
- shopping basket	Marketing	- Custumers behavior - Frequent products
- DNA sequencing - Classes of genes	Bio-Informatics	- Protein structure prediction - Cancer classification

DATA MINING APPLICATIONS

Inputs	Mining process	Outputs
- shopping basket	Marketing	- Custumers behavior - Frequent products
- DNA sequencing - Classes of genes	Bio-Informatics	- Protein structure prediction - Cancer classification
Aurora Project: - Execution traces - Flow diagram - Source code	Software Engineering	- Program comprehension - Fault localization/prediction

FREQUENT ITEMSET MINING

[Agrawal et al, 93]

FREQUENT ITEMSET MINING

[Agrawal et al, 93]

- Aims at finding regularities in datasets (e.g., shopping behavior of customers)

FREQUENT ITEMSET MINING

[Agrawal et al, 93]

- Aims at finding regularities in datasets (e.g., shopping behavior of customers)

In market basket analysis:

- Find sets of products that are frequently bought together

FREQUENT ITEMSET MINING

[Agrawal et al, 93]

- Aims at finding regularities in datasets (e.g., shopping behavior of customers)

In market basket analysis:

- Find sets of products that are frequently bought together

Often found patterns are expressed as association rules, for example:

- **If a customer buys bread and wine, then she/he will probably also buy cheese.**

FREQUENT ITEMSET MINING (PROBLEM)

FREQUENT ITEMSET MINING (PROBLEM)

- Aims at finding regularities in datasets (e.g., shopping behavior of customers)

FREQUENT ITEMSET MINING (PROBLEM)

- Aims at finding regularities in datasets (e.g., shopping behavior of customers)
- **Given:**
  - A set of items $I = \{i_1, \ldots, i_n\}$
  - A set of transactions overs the items $T = \{t_1, \ldots, t_m\}$
  - A minimum support θ

FREQUENT ITEMSET MINING (PROBLEM)

- Aims at finding regularities in datasets (e.g., shopping behavior of customers)
- **Given:**
  - A set of items $I = \{i_1, \ldots, i_n\}$
  - A set of transactions overs the items $T = \{t_1, \ldots, t_m\}$
  - A minimum support θ
- **The need:**
  - The set of itemset P s.t.:

$$freq(P) \geq \theta$$

STANDARD ITEMSET MINING

STANDARD ITEMSET MINING

	A	B	C	D	E	F	G	H
t1:		B	C		E	F	G	H
t2:	A			D			G	
t3:	A		C	D				H
t4:	A				E	F		
t5:		B			E	F		
t6:		B			E	F	G	

STANDARD ITEMSET MINING

t1:		B	C		E	F	G	H
t2:	A			D			G	
t3:	A		C	D				H
t4:	A				E	F		
t5:		B			E	F		
t6:		B			E	F	G	

STANDARD ITEMSET MINING

t1:		B	C		E	F	G	H
t2:	A			D			G	
t3:	A		C	D				H
t4:	A				E	F		
t5:		B			E	F		
t6:		B			E	F	G	

$$cover(BEF) = \{t_1, t_5, t_6\}$$

STANDARD ITEMSET MINING

t1:		B	C		E	F	G	H
t2:	A			D			G	
t3:	A		C	D				H
t4:	A				E	F		
t5:		B			E	F		
t6:		B			E	F	G	

$$cover(BEF) = \{t_1, t_5, t_6\}$$

$$freq(BEF) = 50\,\%$$

STANDARD ITEMSET MINING

- **Brute force enumeration is infeasible**
  - 128 items 10^{68} itemsets (atoms in the universe)

t1:		B	C		E	F	G	H
t2:	A			D			G	
t3:	A		C	D				H
t4:	A				E	F		
t5:		B			E	F		
t6:		B			E	F	G	

$$cover(BEF) = \{t_1, t_5, t_6\}$$

$$freq(BEF) = 50\,\%$$

STANDARD ITEMSET MINING

- **Brute force enumeration is infeasible**
  - 128 items 10^{68} itemsets (atoms in the universe)
- **Several specialised algorithms have been developed:**

Apriori, Eclat, FP-Growth, LCM…

t1:		B	C		E	F	G	H
t2:	A			D			G	
t3:	A		C	D				H
t4:	A				E	F		
t5:		B			E	F		
t6:		B			E	F	G	

$$cover(BEF) = \{t_1, t_5, t_6\}$$

$$freq(BEF) = 50\,\%$$

STANDARD ITEMSET MINING

- **Brute force enumeration is infeasible**
  - 128 items 10^{68} itemsets (atoms in the universe)
- **Several specialised algorithms have been developed:**

Apriori, Eclat, FP-Growth, LCM…

- **Dealing with basic user's constraints:**

Frequency, Condensed representations (closedness, maximality,…), Size…

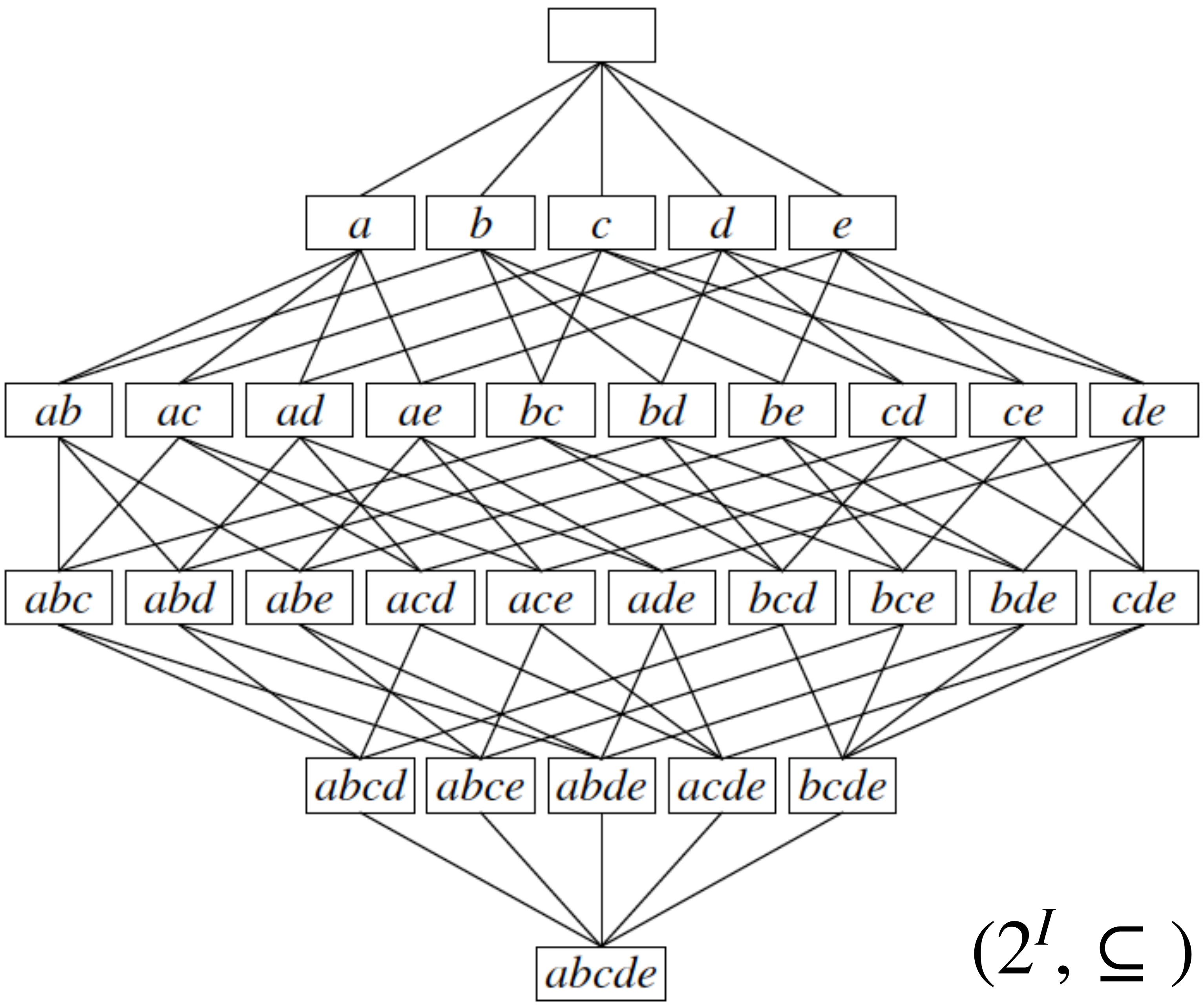
t1:		B	C		E	F	G	H
t2:	A			D			G	
t3:	A		C	D				H
t4:	A				E	F		
t5:		B			E	F		
t6:		B			E	F	G	

$$cover(BEF) = \{t_1, t_5, t_6\}$$

$$freq(BEF) = 50\,\%$$

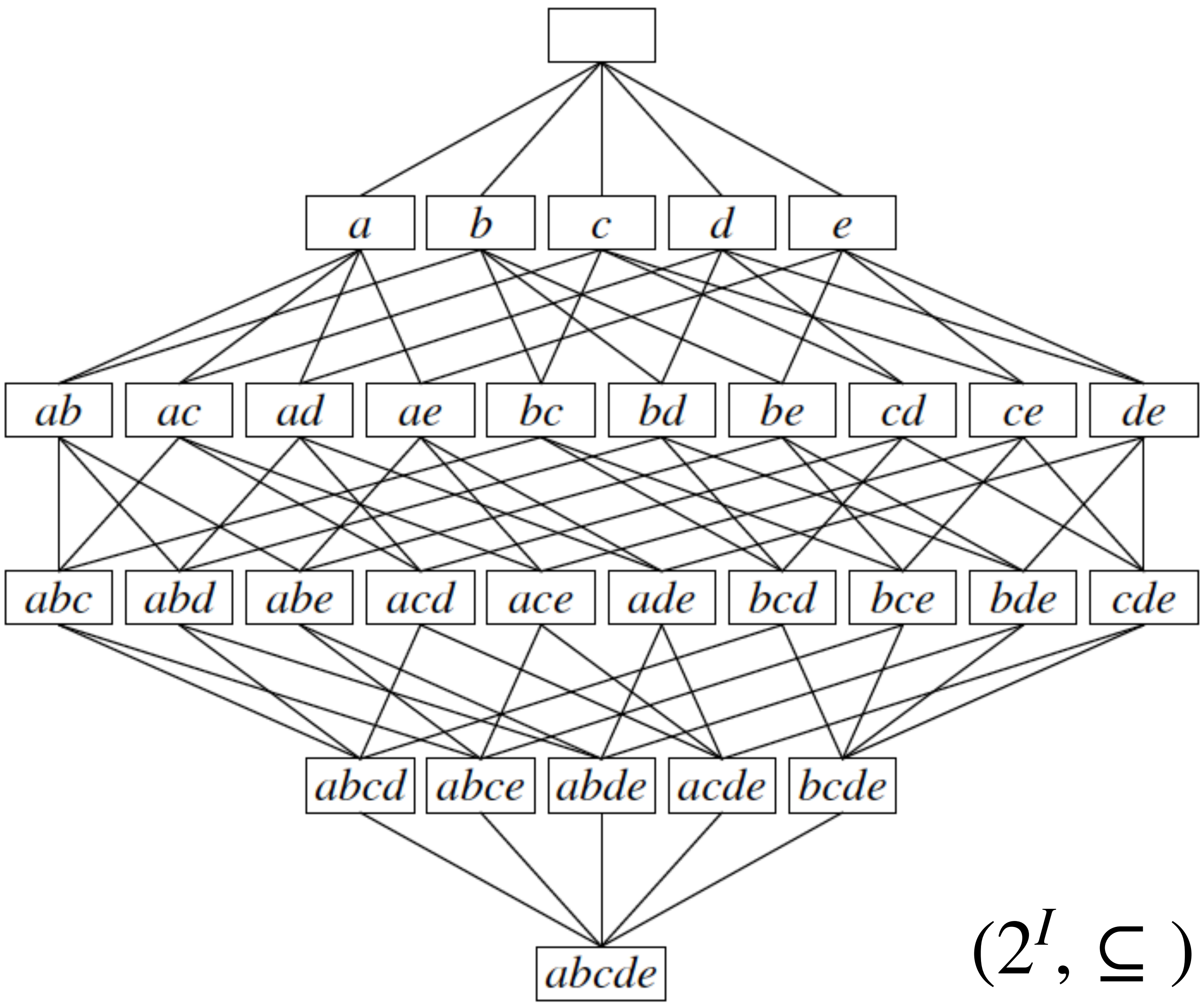
EXAMPLE

EXAMPLE



$$(2^I, \subseteq)$$

EXAMPLE



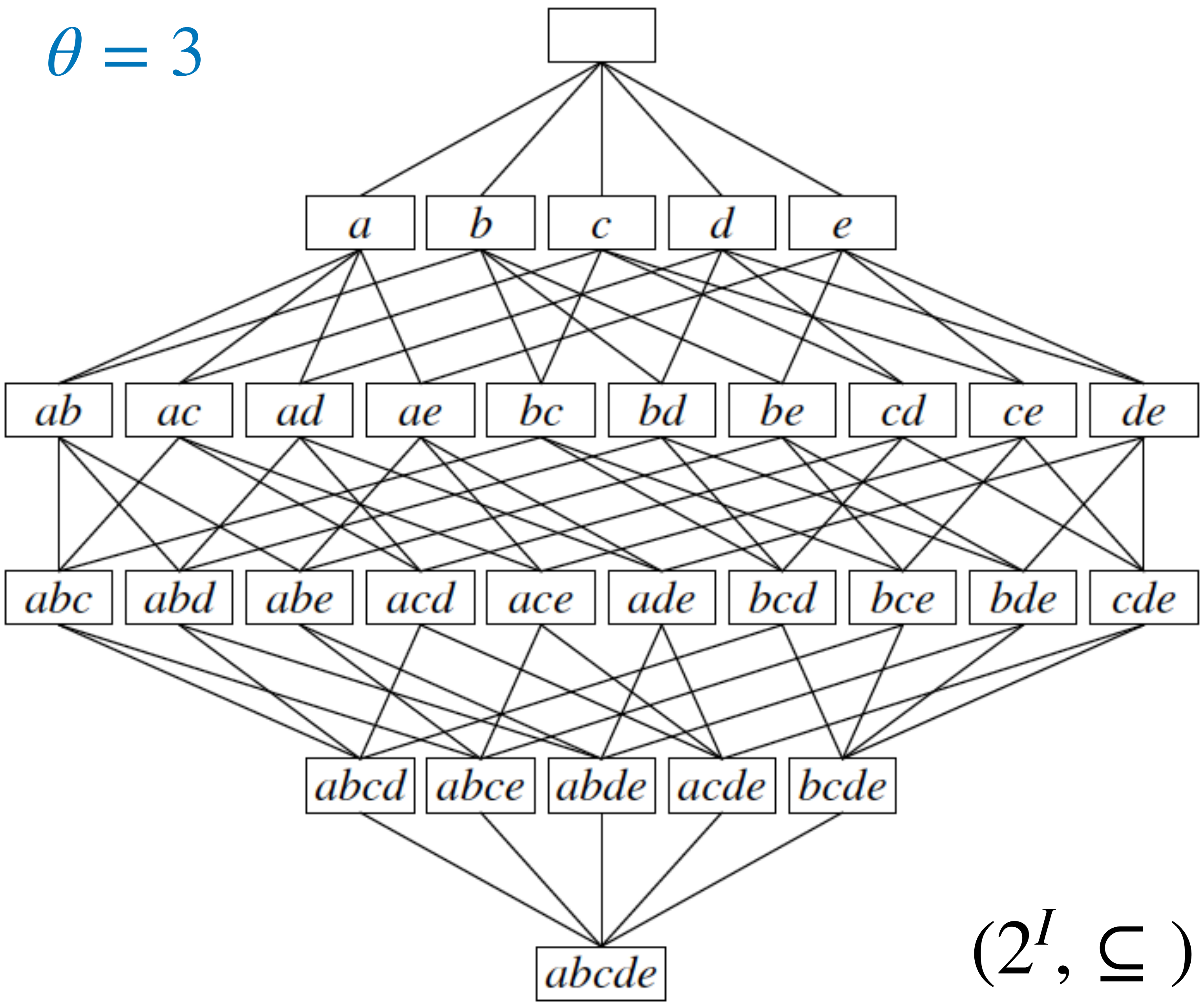
$(2^I, \subseteq)$

D

	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

EXAMPLE

$\theta = 3$



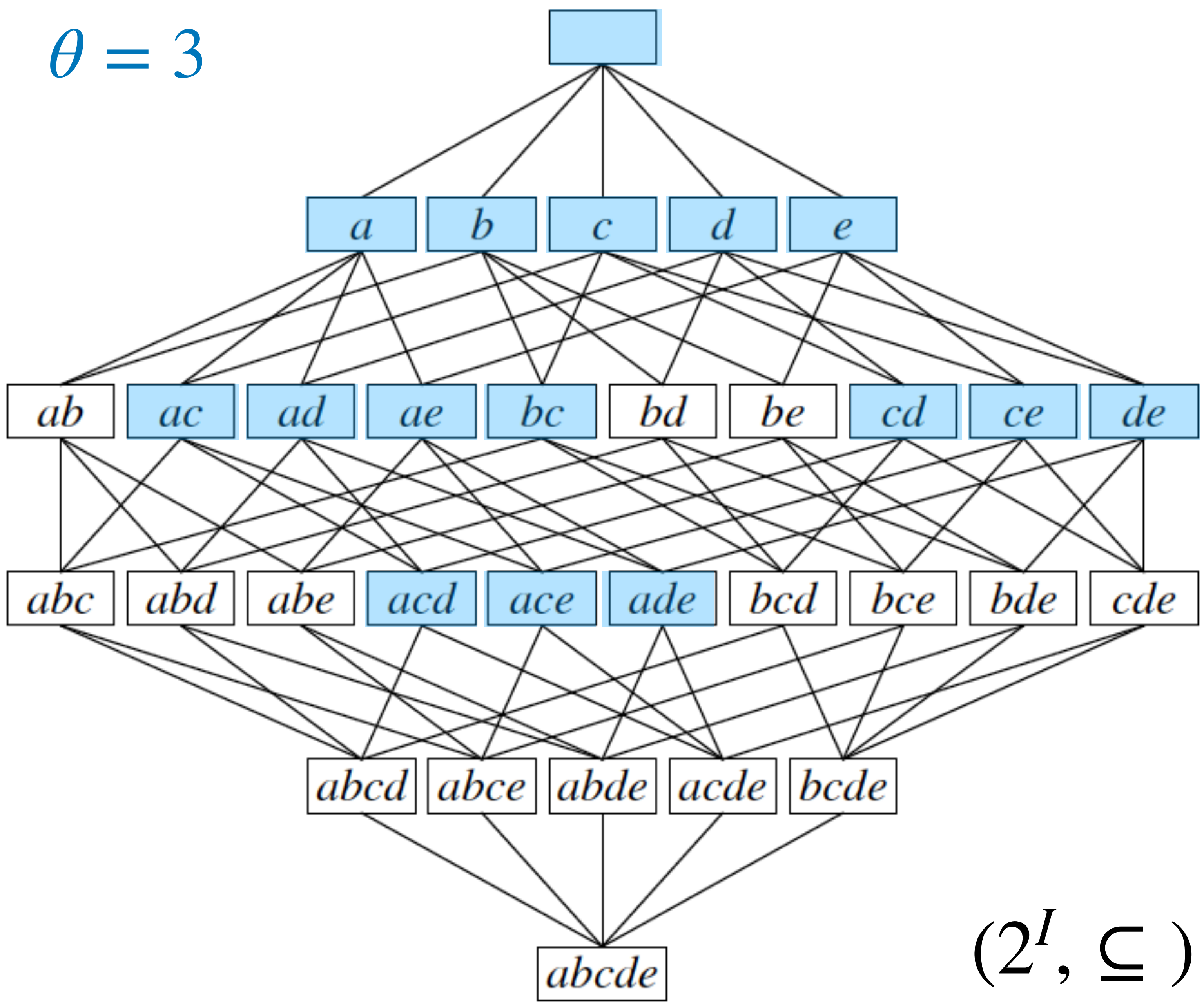
$(2^I, \subseteq)$

D

	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

EXAMPLE

$\theta = 3$



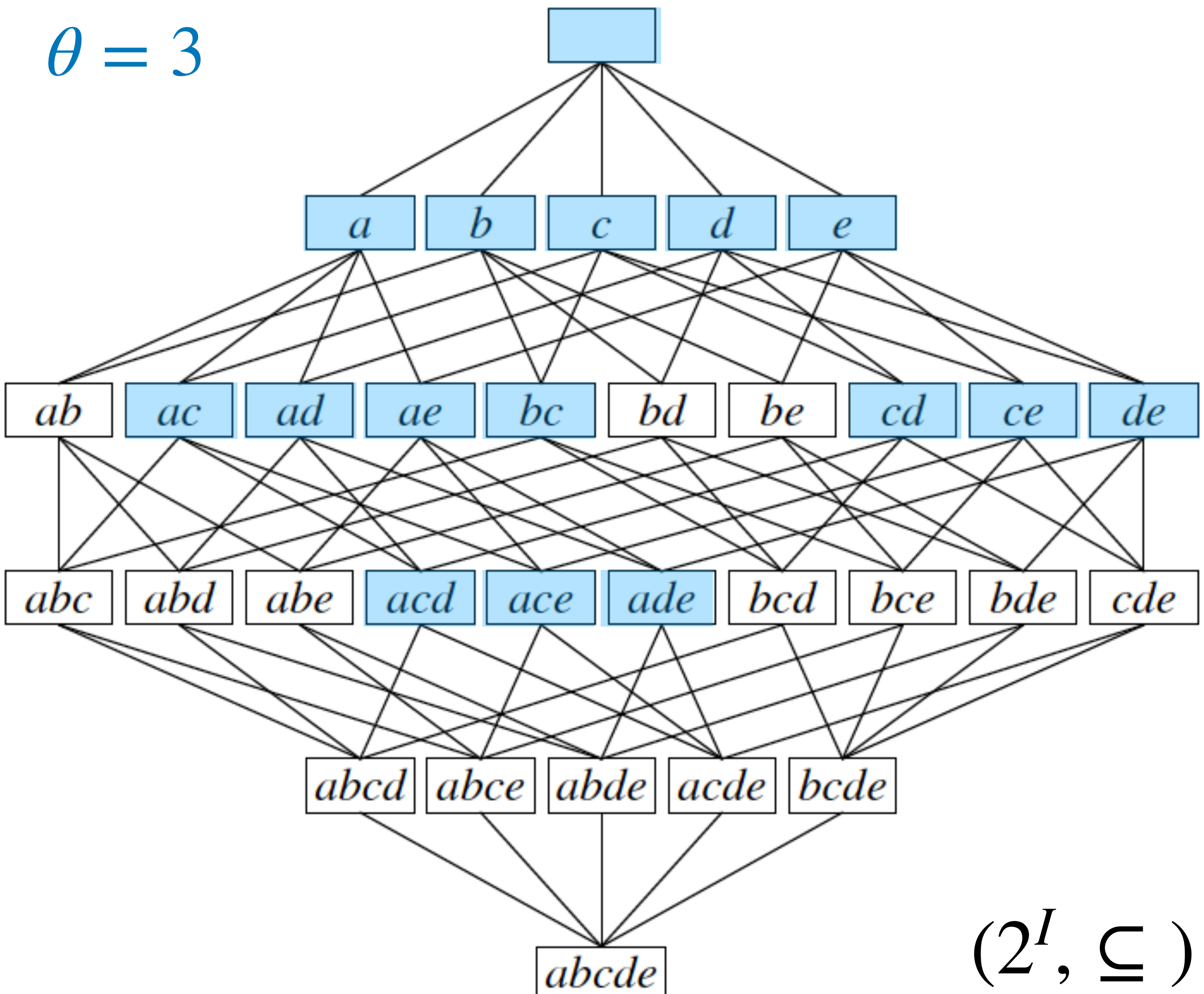
$(2^I, \subseteq)$

D

	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

EXAMPLE

$\theta = 3$



$(2^I, \subseteq)$

D

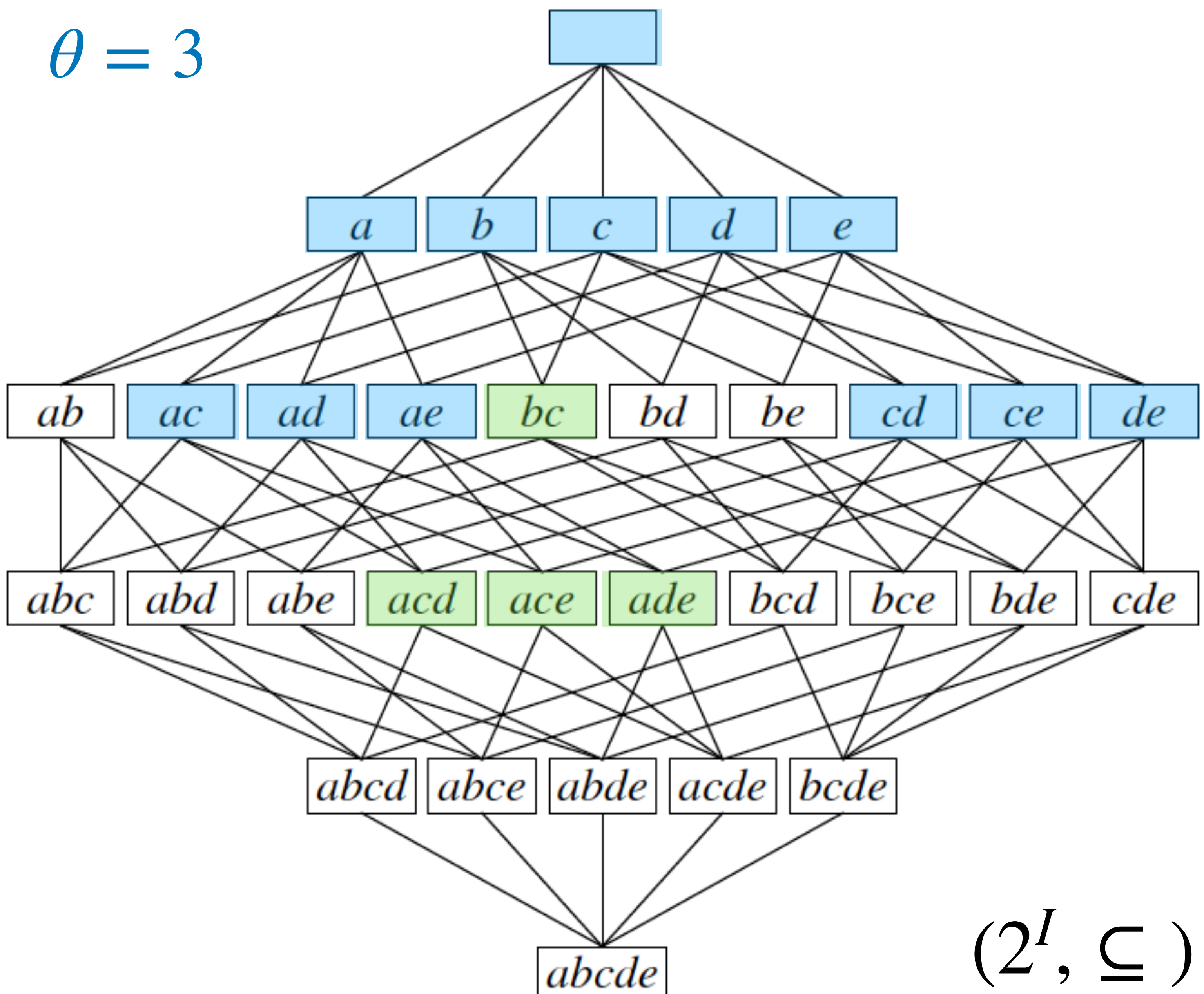
	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

Maximal

$$M_\theta = \{P \in \mathcal{I} \mid freq(P) \geq \theta \wedge \forall P' \supset P : freq(P') < \theta\}$$

EXAMPLE

$\theta = 3$



$(2^I, \subseteq)$

D

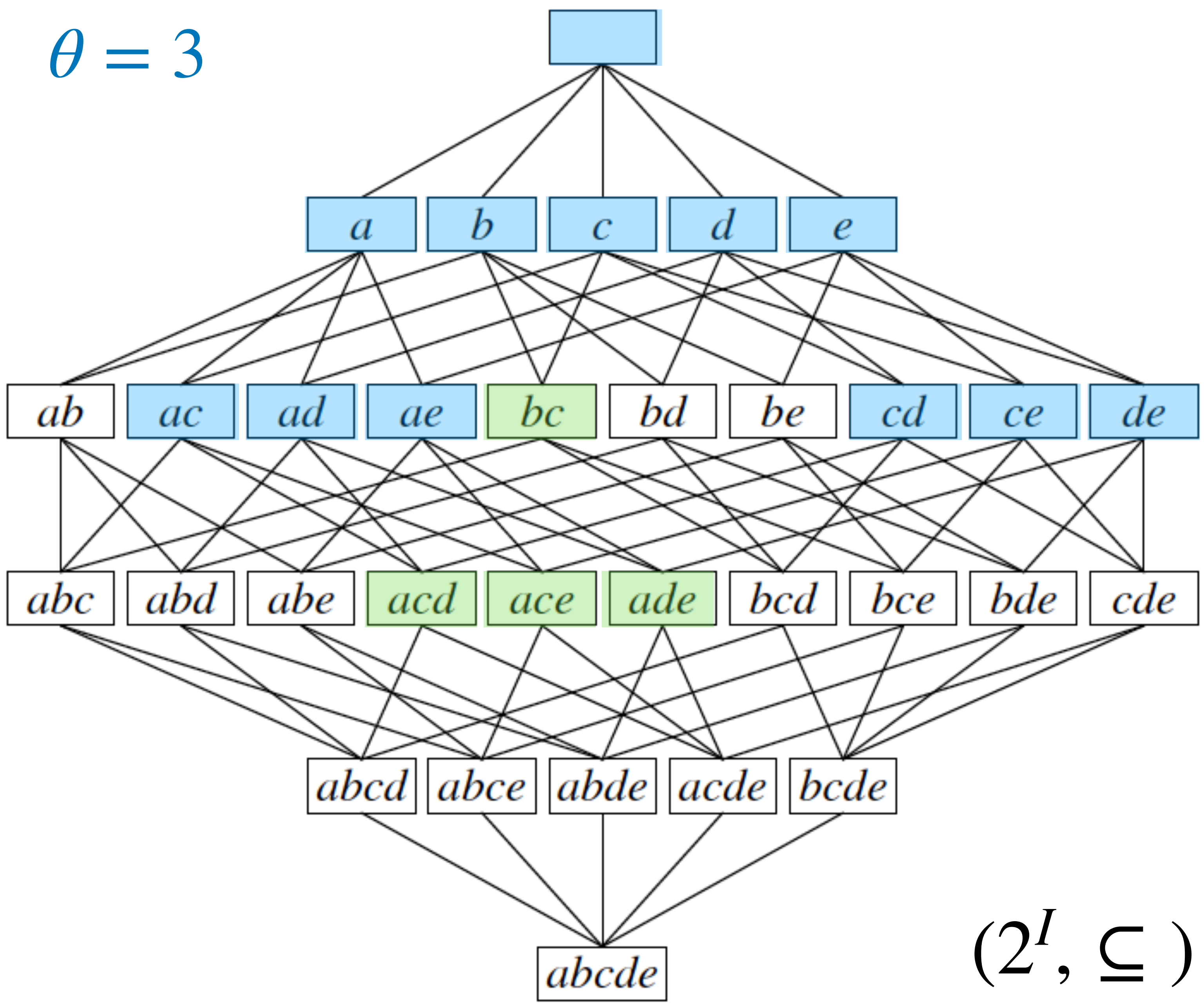
	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

Maximal

$$M_\theta = \{P \in \mathcal{I} \mid freq(P) \geq \theta \wedge \forall P' \supset P : freq(P') < \theta\}$$

EXAMPLE

$\theta = 3$



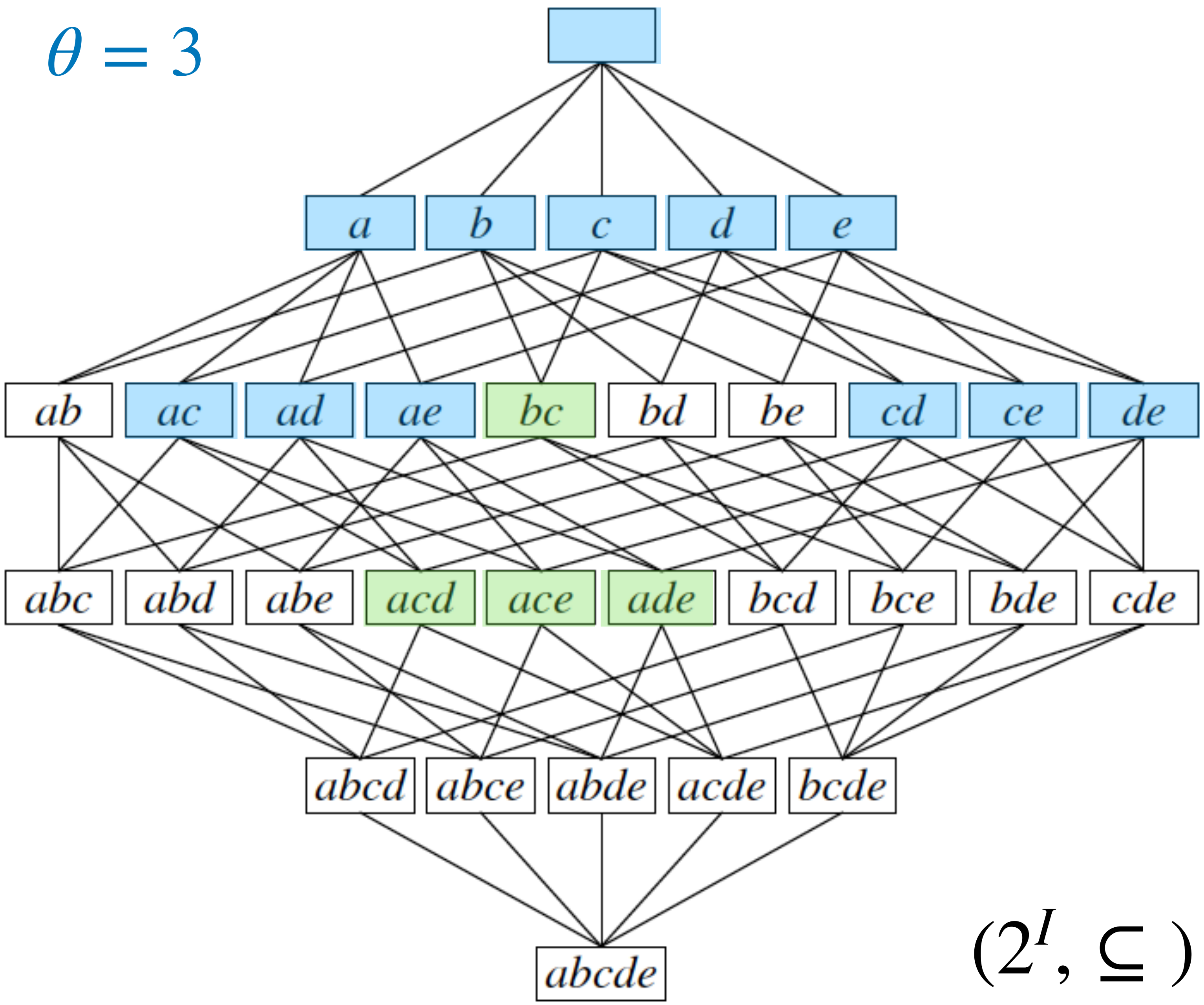
$(2^I, \subseteq)$

D

	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

EXAMPLE

$\theta = 3$



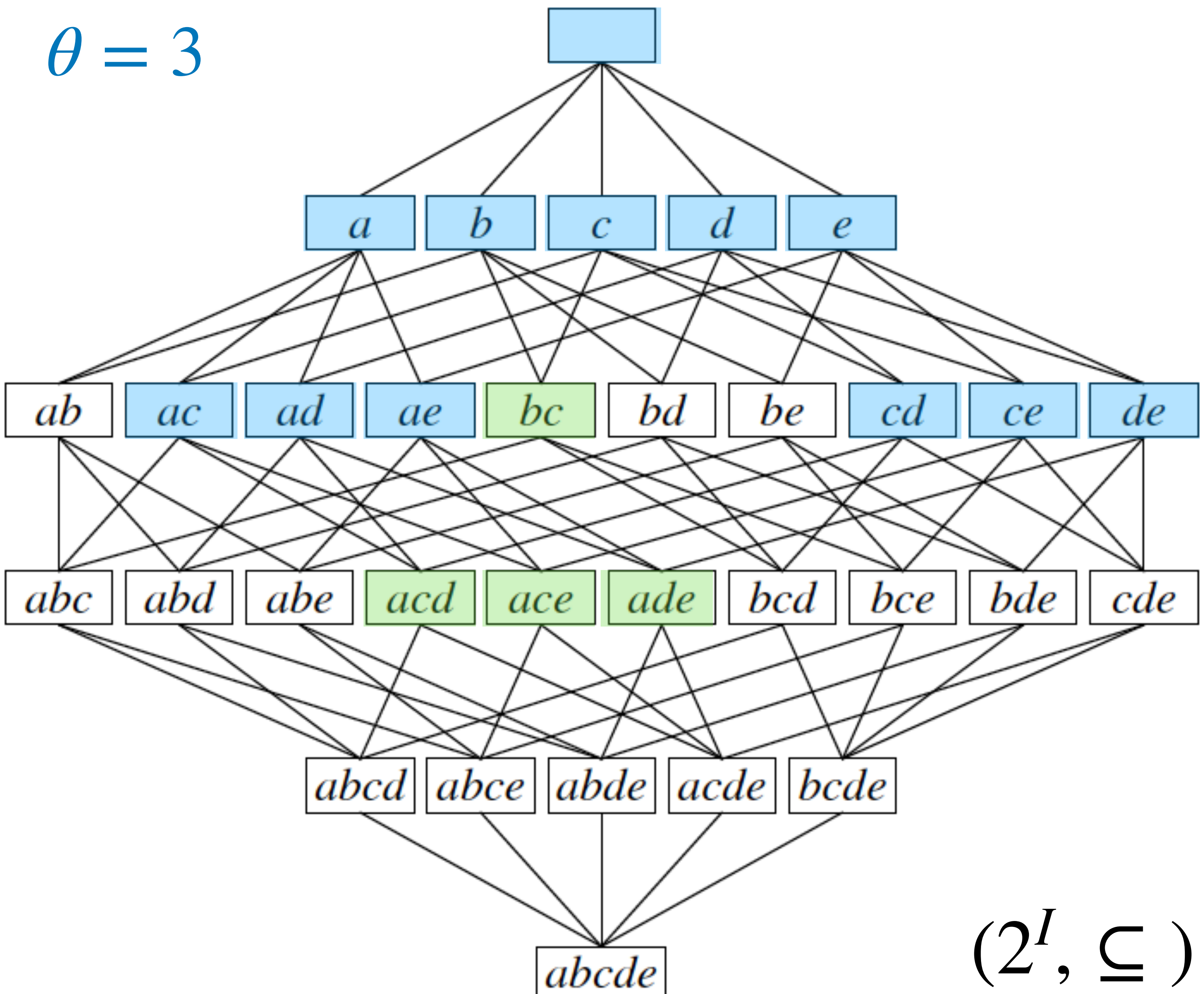
$(2^I, \subseteq)$

D

	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

EXAMPLE

$\theta = 3$



$(2^I, \subseteq)$

D

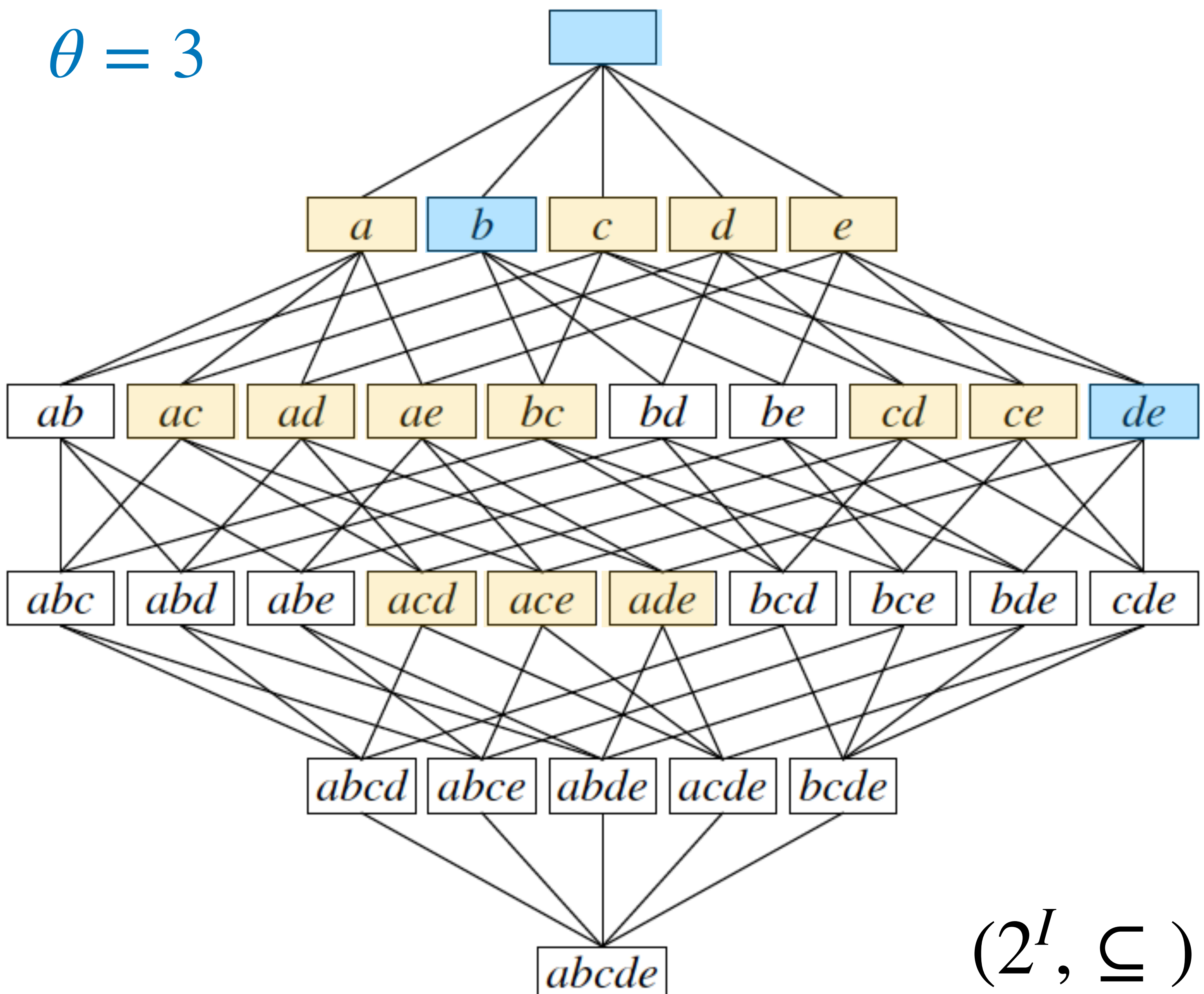
	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

Closedness

$$M_\theta = \{P \in \mathcal{I} |\ freq(P) \geq \theta \wedge \forall P' \supset P : freq(P') < \theta\}$$

EXAMPLE

$\theta = 3$



$(2^I, \subseteq)$

D

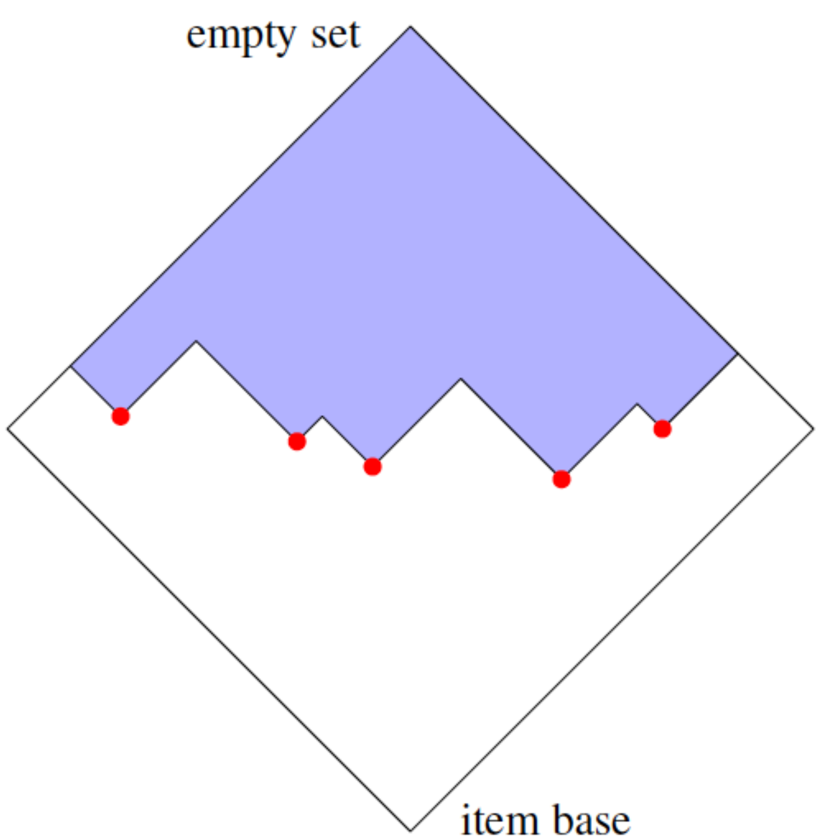
	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

Closedness

$$M_\theta = \{P \in \mathcal{I} \mid \ freq(P) \geq \theta \wedge \forall P' \supset P : freq(P') < \theta\}$$

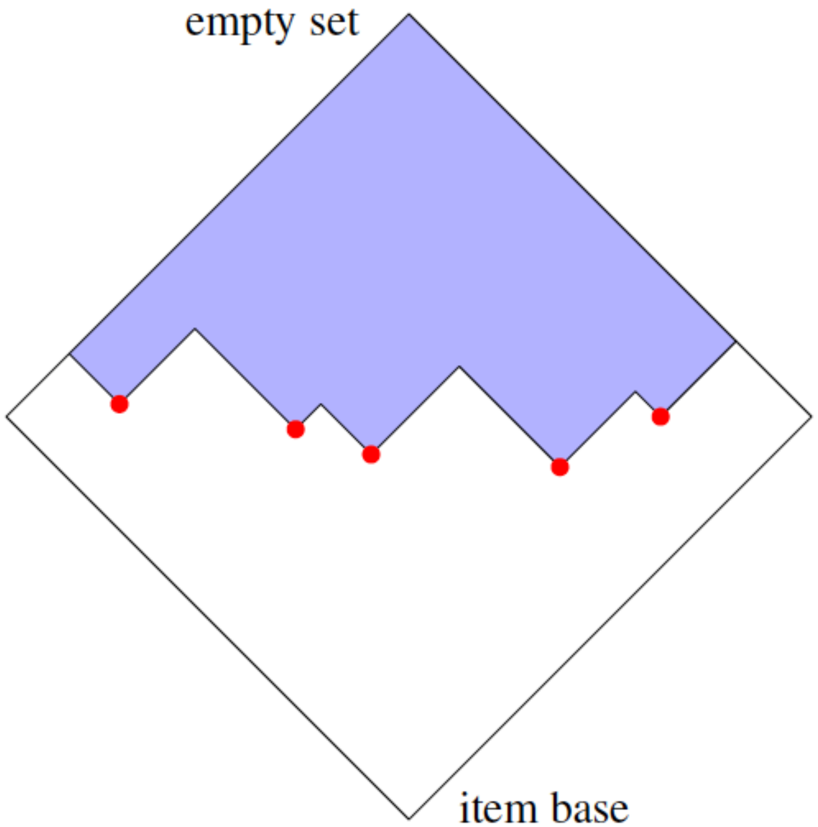
CONDENSED REPRESENTATION

CONDENSED REPRESENTATION

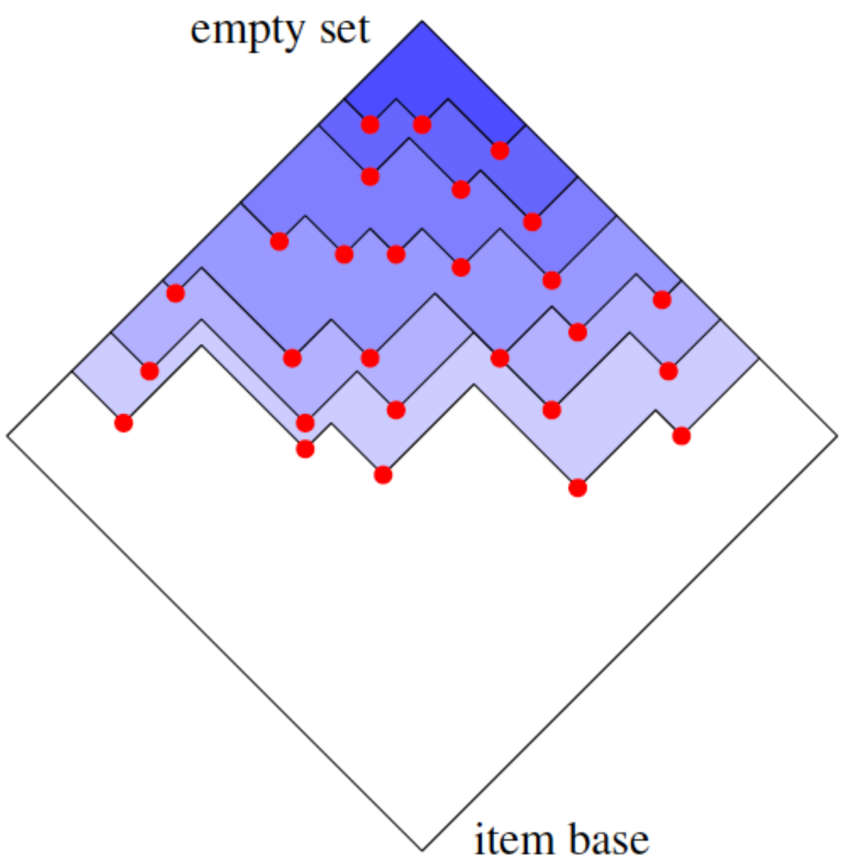


maximal (frequent) item sets

CONDENSED REPRESENTATION

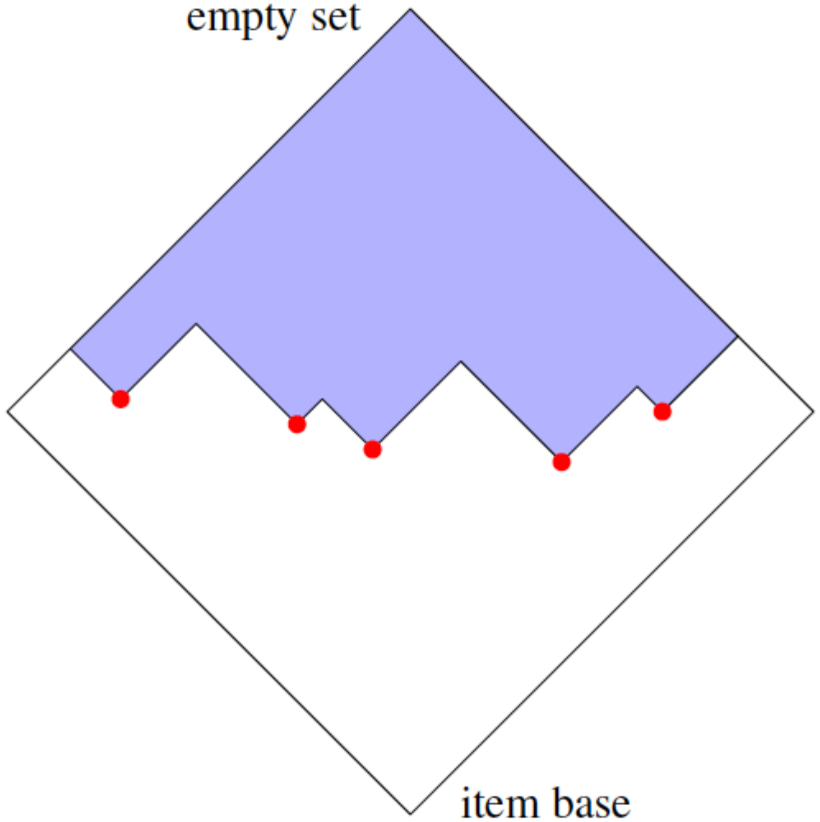


maximal (frequent) item sets

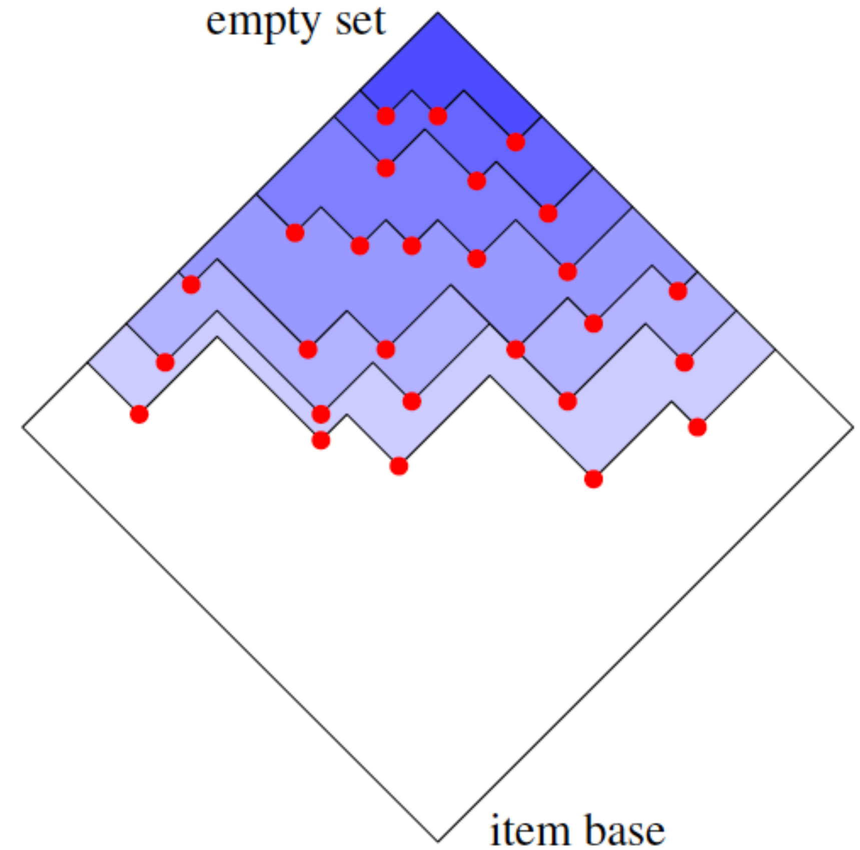


closed (frequent) item sets

CONDENSED REPRESENTATION



maximal (frequent) item sets



closed (frequent) item sets

Dataset	#Frequent	#Closed	#Maximal
Zoo-1	151 807	3 292	230
Mushroom	155 734	3 287	453
Lymph	9 967 402	46 802	5 191
Hepatitis	27 . 10^7	1 827 264	189 205

SPECIALIZED VS DECLARATIVE DATA MINING

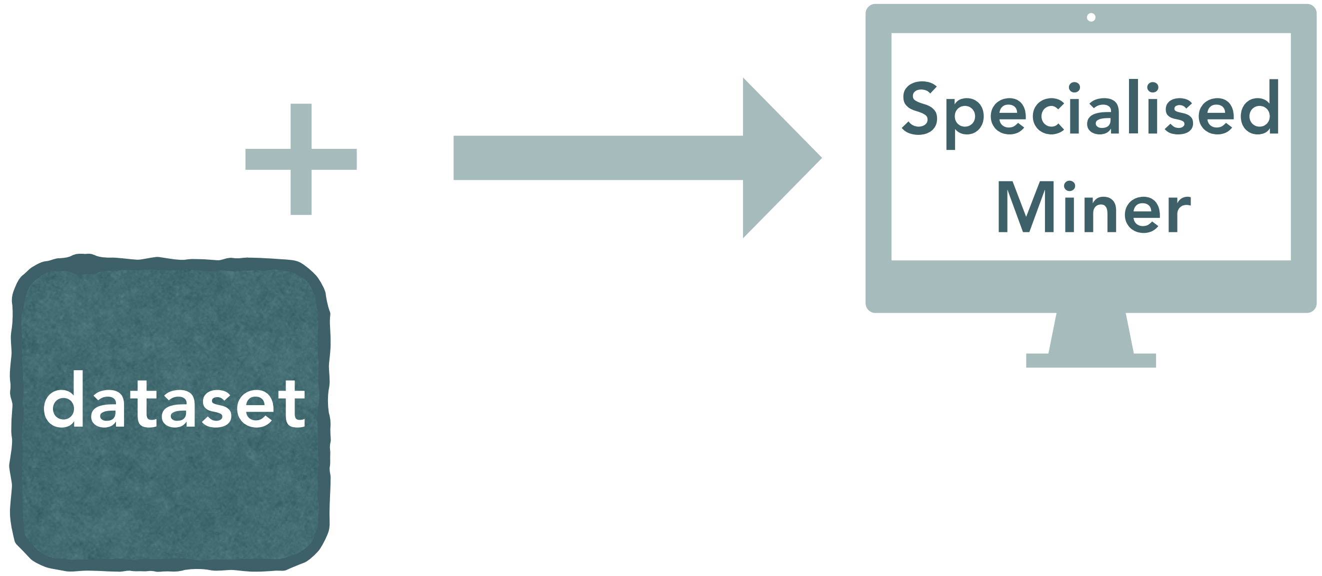
SPECIALIZED VS DECLARATIVE DATA MINING

SPECIALIZED VS DECLARATIVE DATA MINING

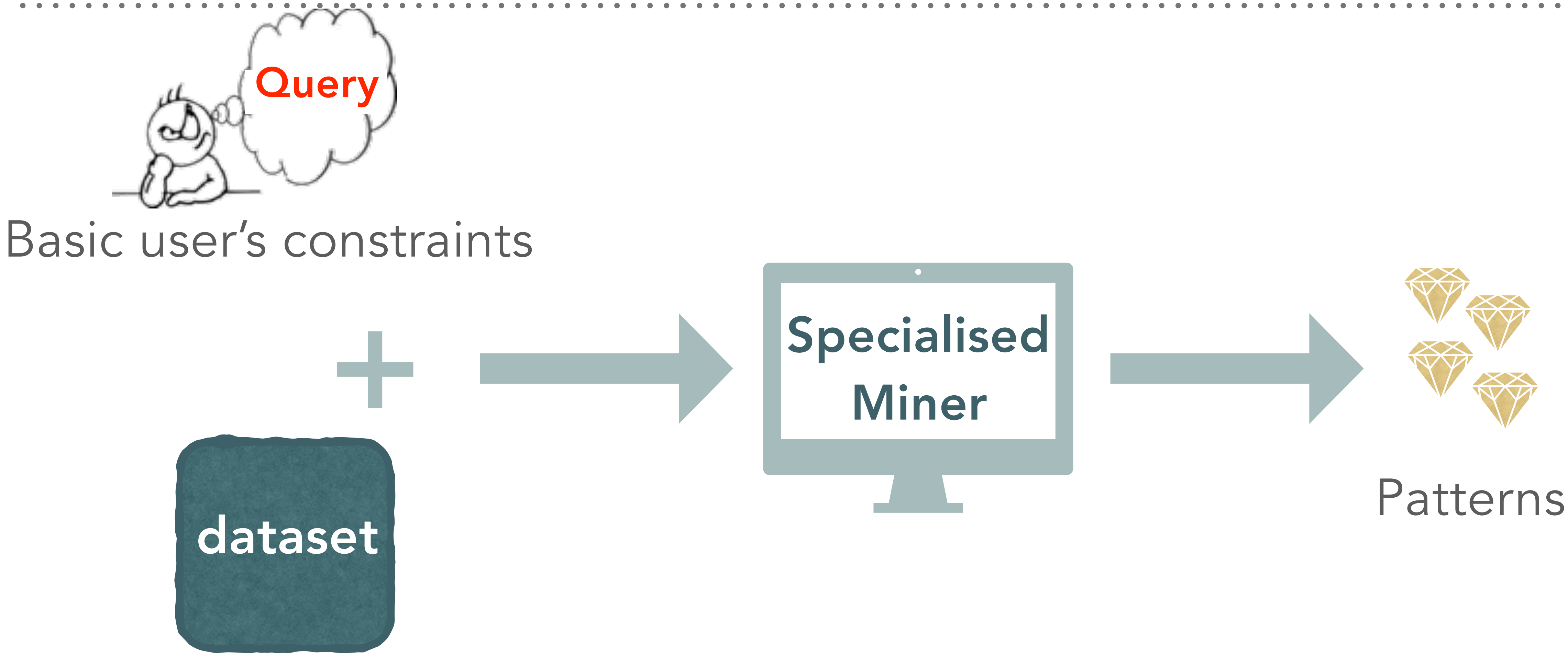
Basic user's constraints

SPECIALIZED VS DECLARATIVE DATA MINING

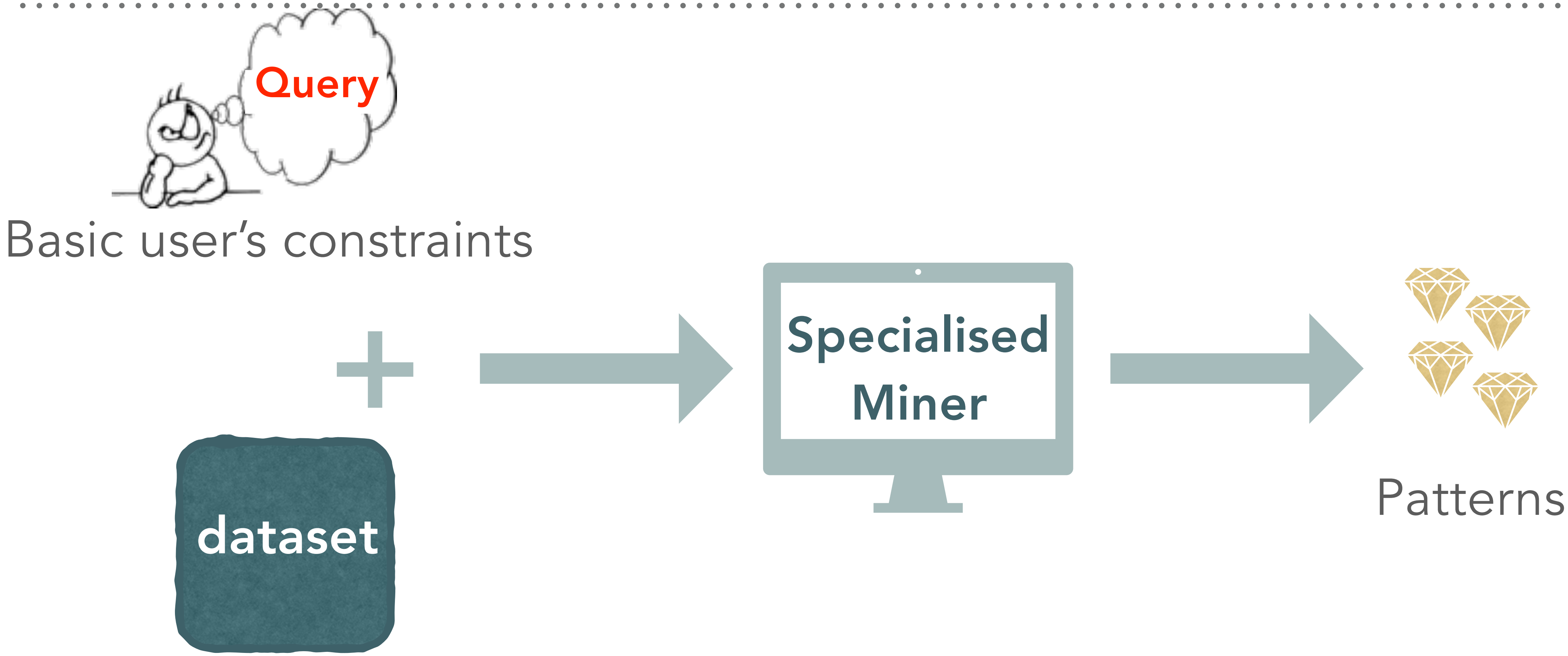
Basic user's constraints



SPECIALIZED VS DECLARATIVE DATA MINING

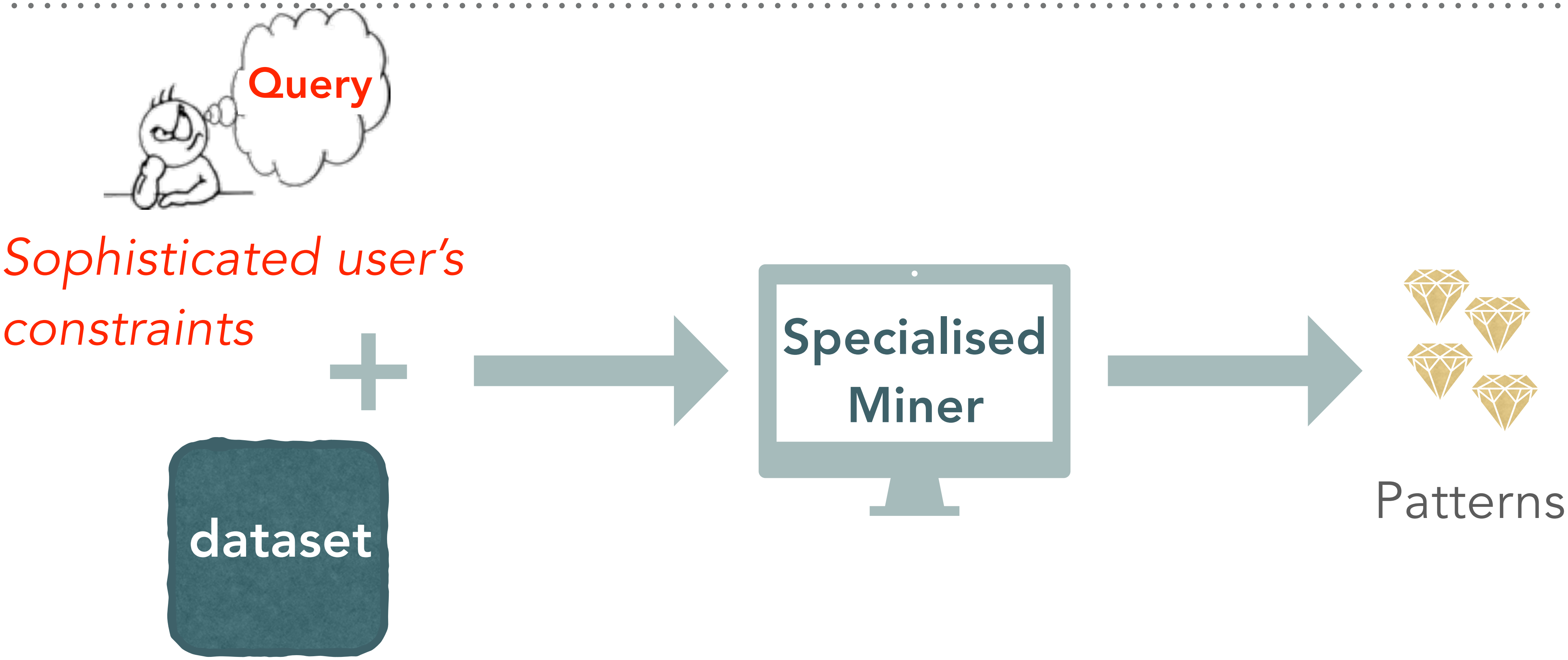


SPECIALIZED VS DECLARATIVE DATA MINING



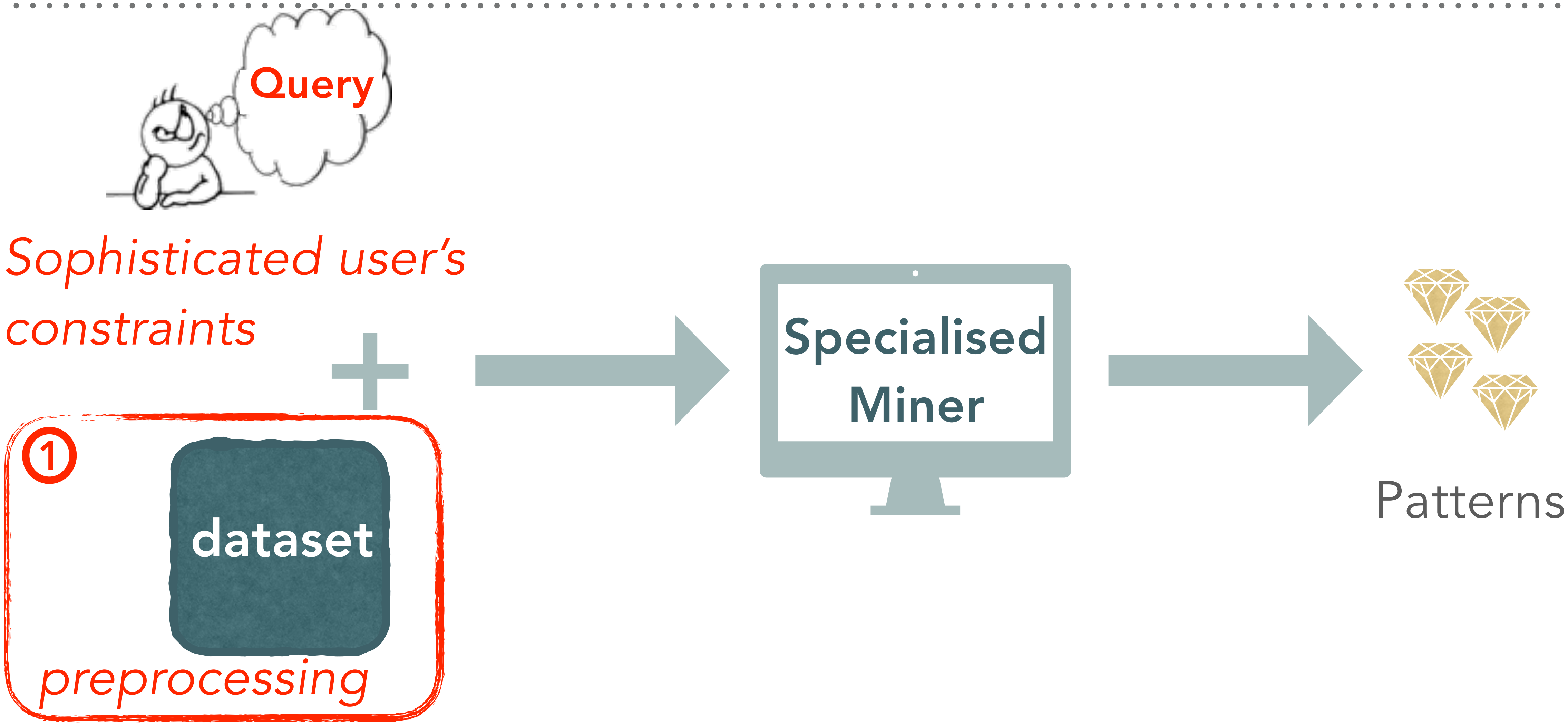
Limitations: Dealing with sophisticated user's constraints **[Wojciechowski and Zakrzewicz, 02]**

SPECIALIZED VS DECLARATIVE DATA MINING



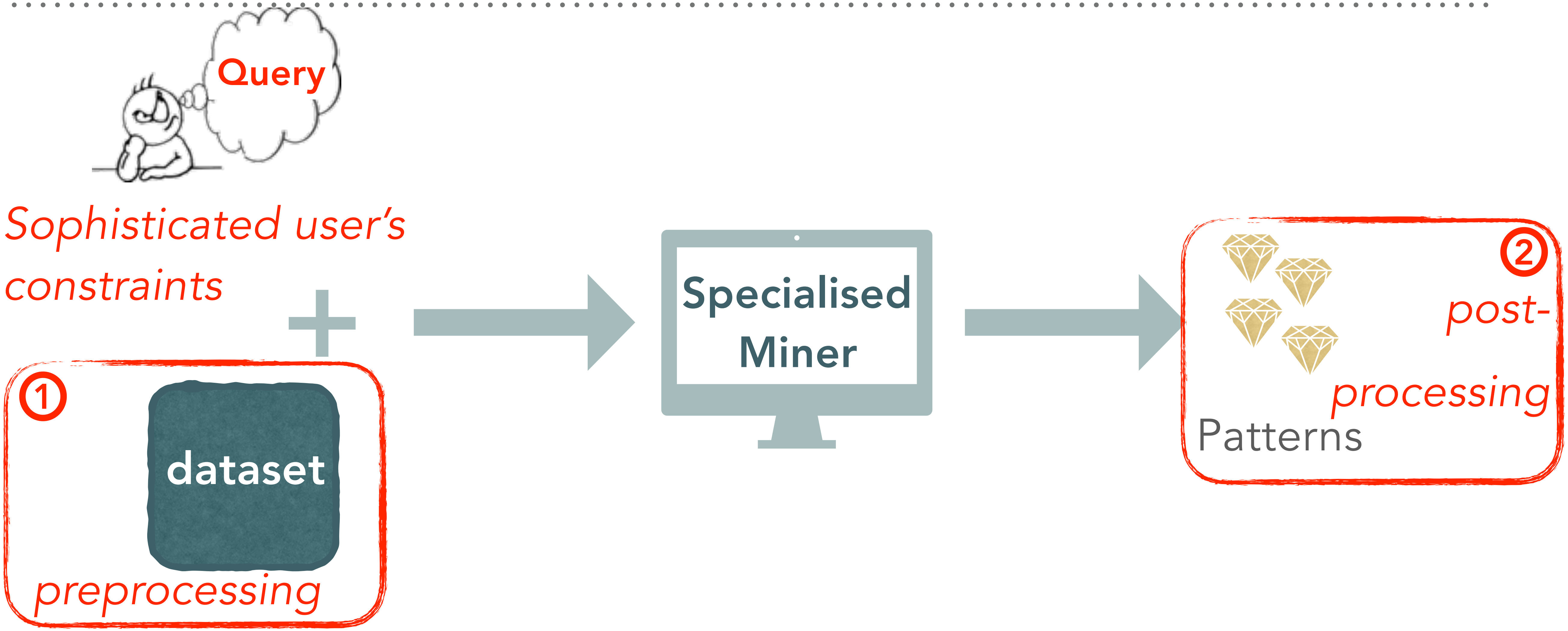
Limitations: Dealing with sophisticated user's constraints **[Wojciechowski and Zakrzewicz, 02]**

SPECIALIZED VS DECLARATIVE DATA MINING



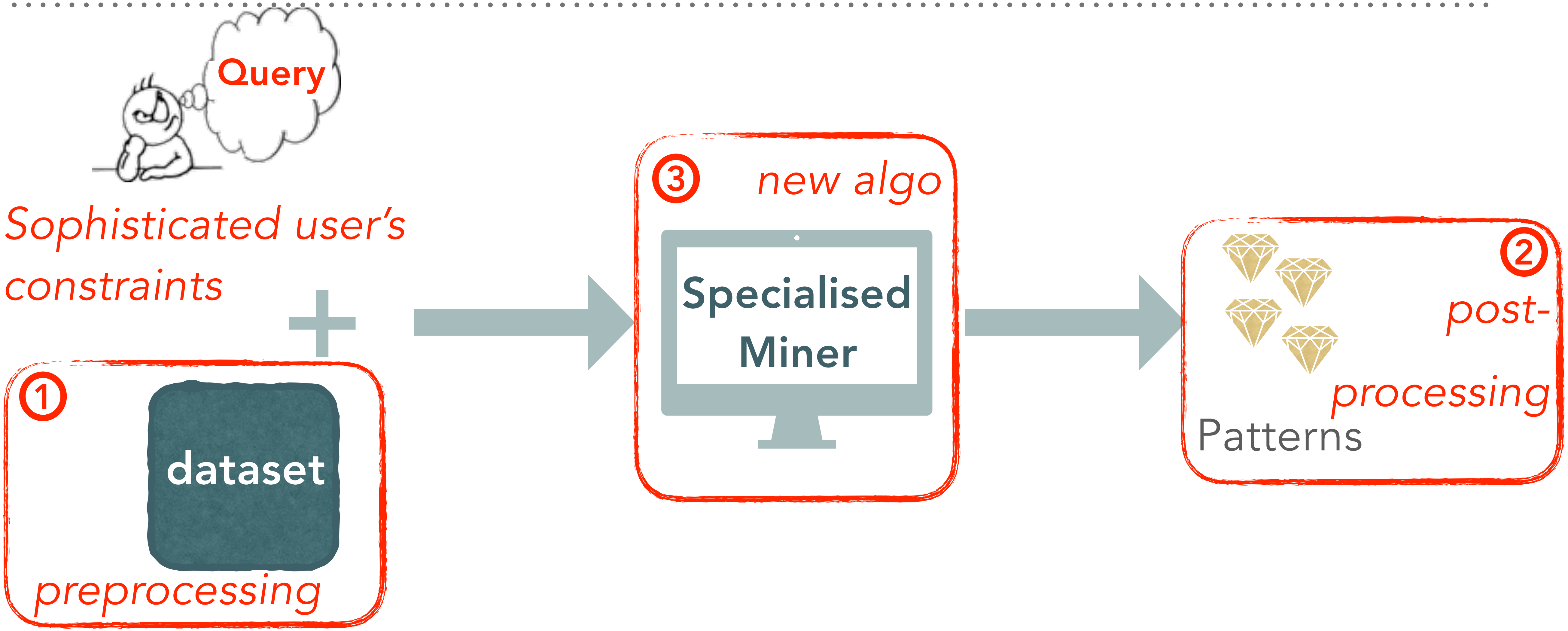
Limitations: Dealing with sophisticated user's constraints **[Wojciechowski and Zakrzewicz, 02]**

SPECIALIZED VS DECLARATIVE DATA MINING



Limitations: Dealing with sophisticated user's constraints **[Wojciechowski and Zakrzewicz, 02]**

SPECIALIZED VS DECLARATIVE DATA MINING



Limitations: Dealing with sophisticated user's constraints **[Wojciechowski and Zakrzewicz, 02]**

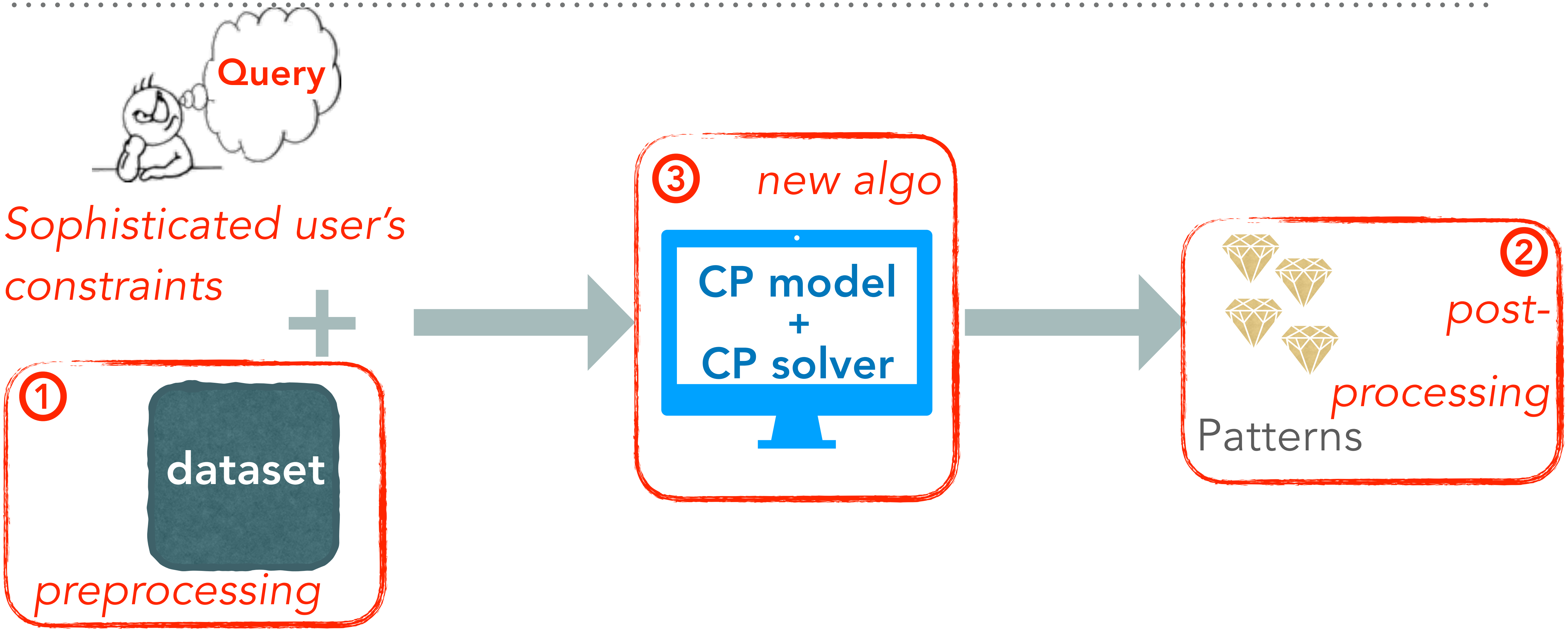
SPECIALIZED VS DECLARATIVE DATA MINING

Limitations: Dealing with sophisticated user's constraints **[Wojciechowski and Zakrzewicz, 02]**

Need: *Declarative way to deal with more complex queries*

➤ ***Declarative data Mining***

SPECIALIZED VS DECLARATIVE DATA MINING

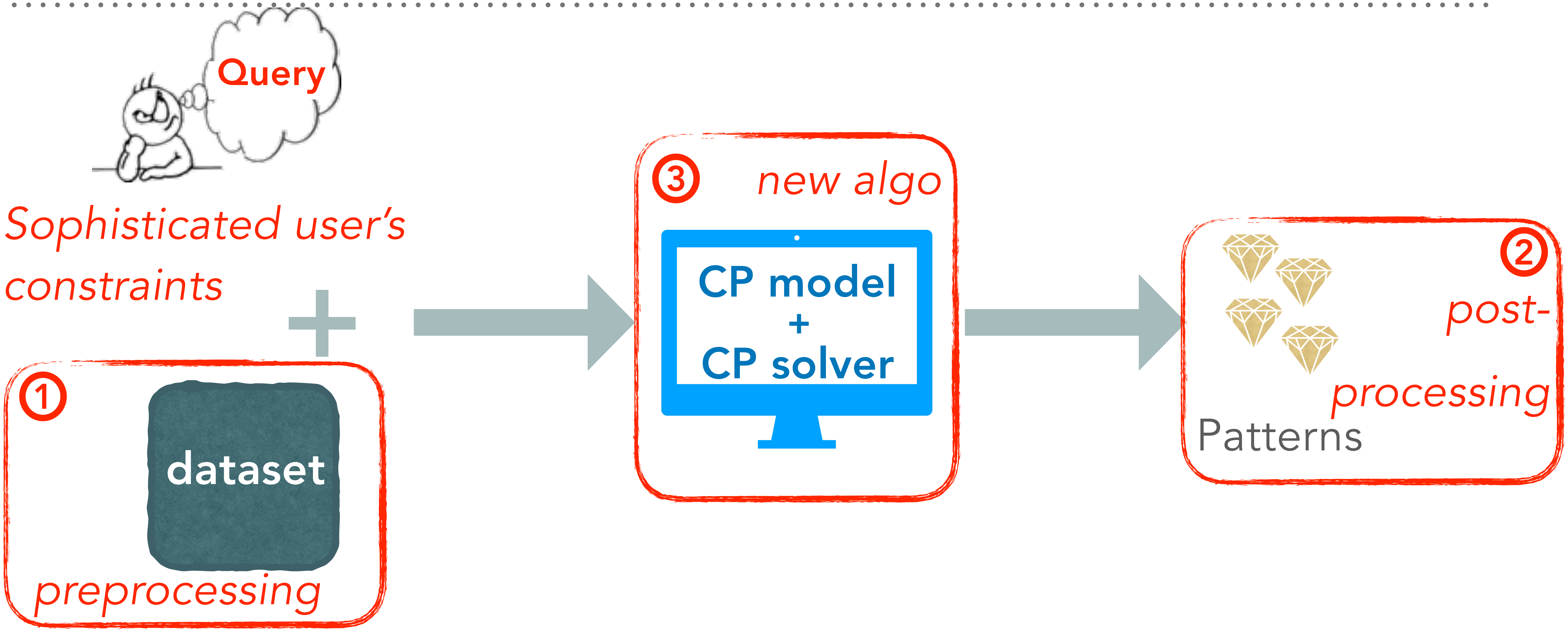


Limitations: Dealing with sophisticated user's constraints **[Wojciechowski and Zakrzewicz, 02]**

Need: *Declarative way to deal with more complex queries*

➤ ***Declarative data Mining***

SPECIALIZED VS DECLARATIVE DATA MINING

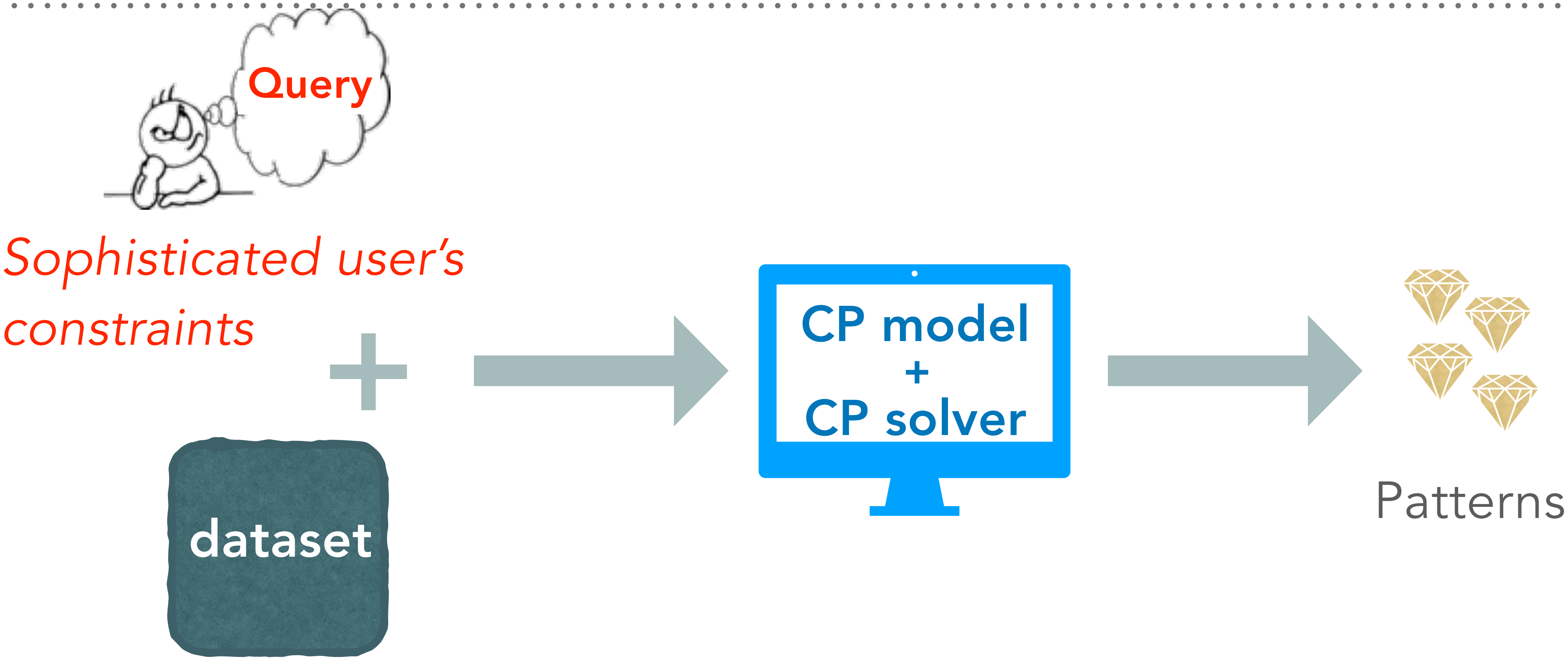


Limitations: Dealing with sophisticated user's constraints **[Wojciechowski and Zakrzewicz, 02]**

Need: *Declarative way to deal with more complex queries*

➤ ***Declarative data Mining***

SPECIALIZED VS DECLARATIVE DATA MINING



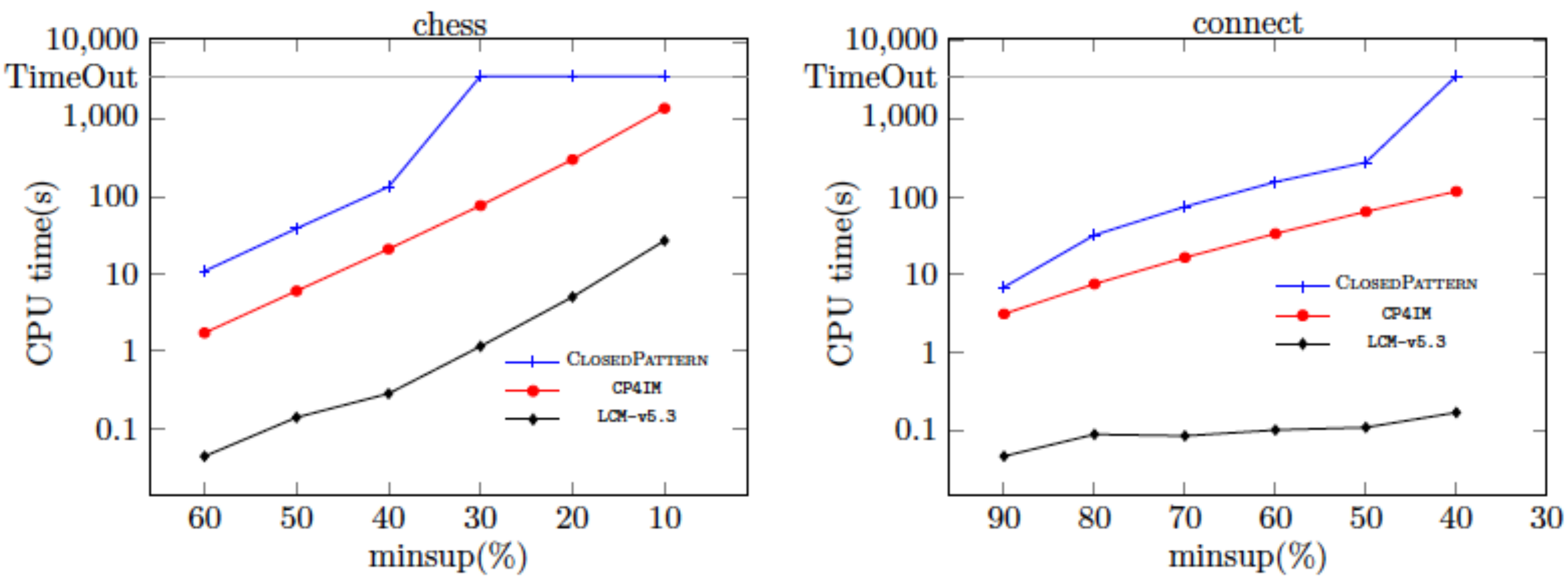
Limitations: Dealing with sophisticated user's constraints **[Wojciechowski and Zakrzewicz, 02]**

Need: *Declarative way to deal with more complex queries*

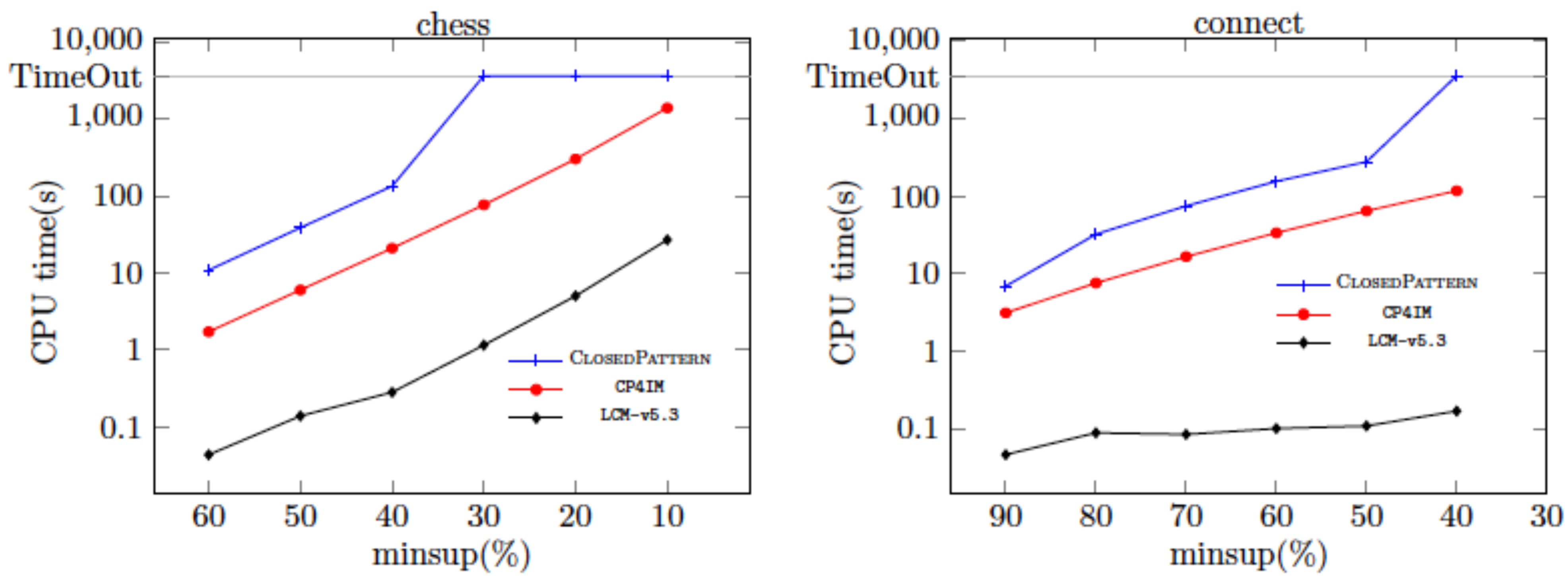
➤ ***Declarative data Mining***

SPECIALISED VS DECLARATIVE DATA MINING

SPECIALISED VS DECLARATIVE DATA MINING

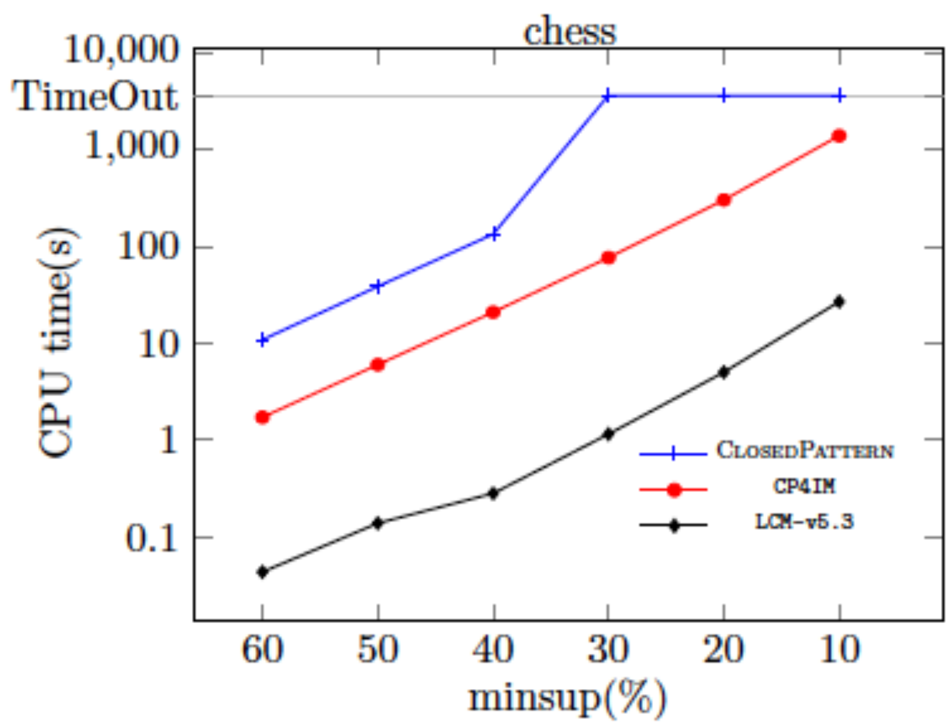


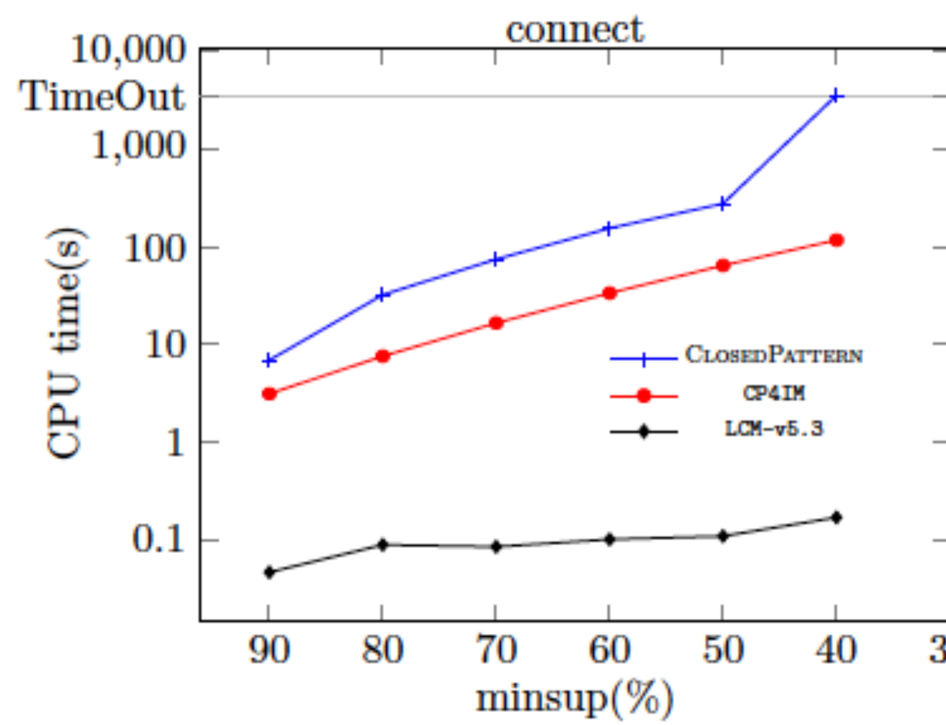
SPECIALISED VS DECLARATIVE DATA MINING



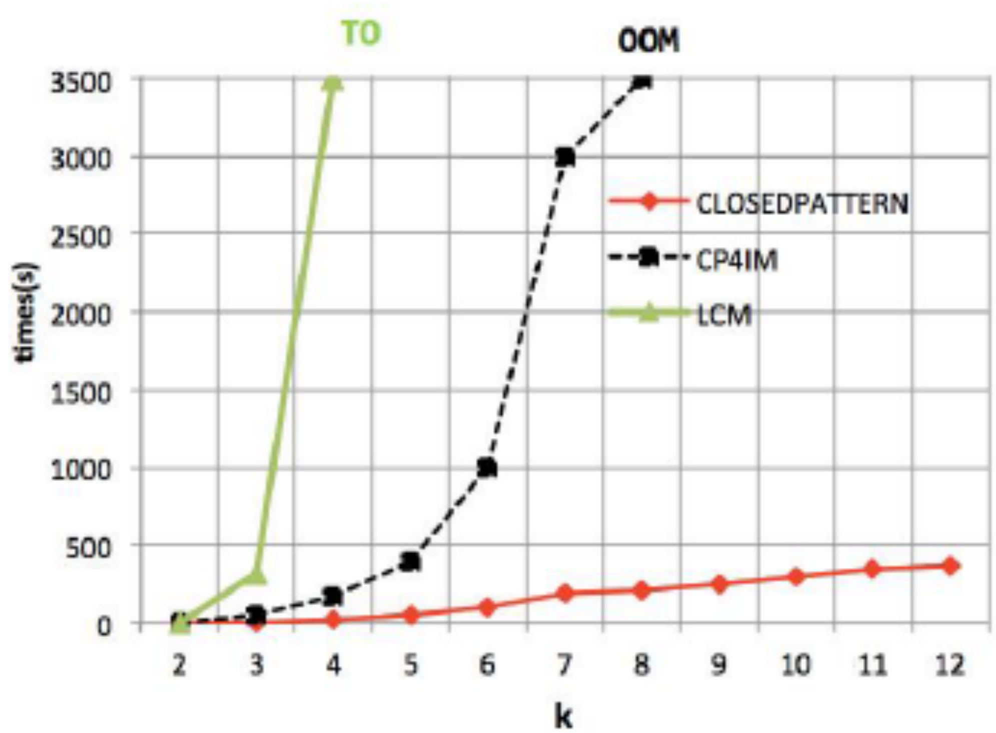
Specialised
is the winner!

SPECIALISED VS DECLARATIVE DATA MINING

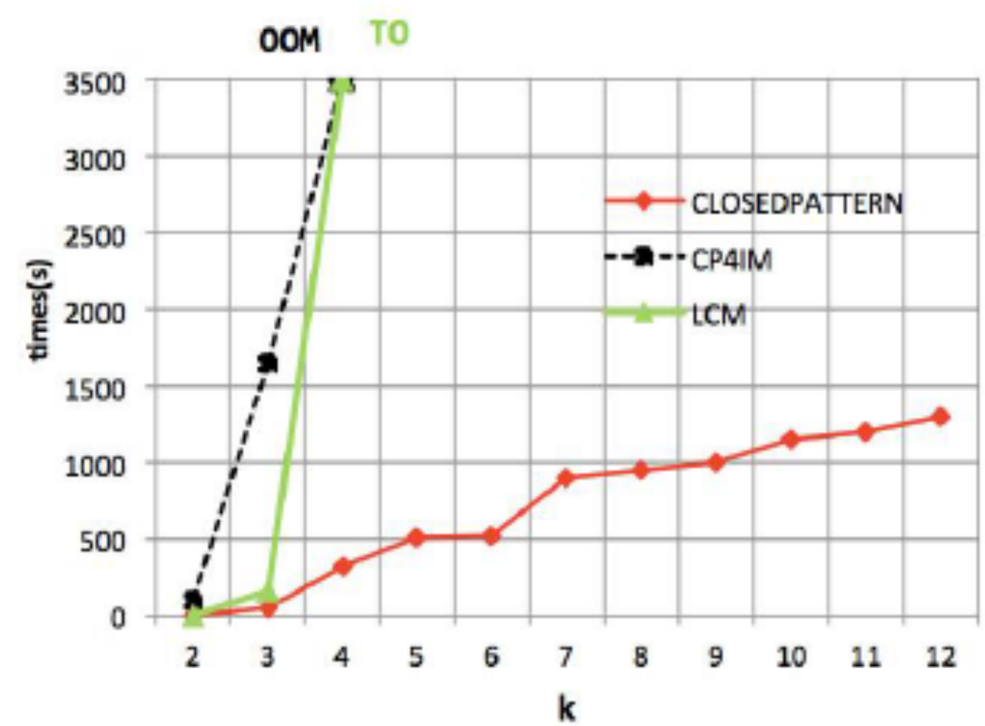




Specialised is the winner!

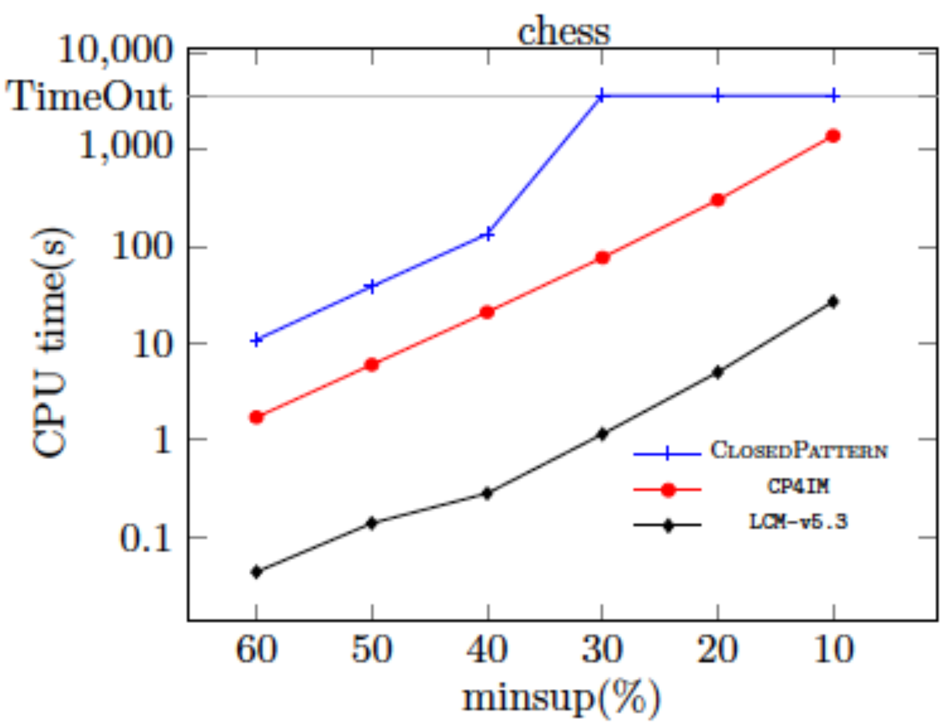


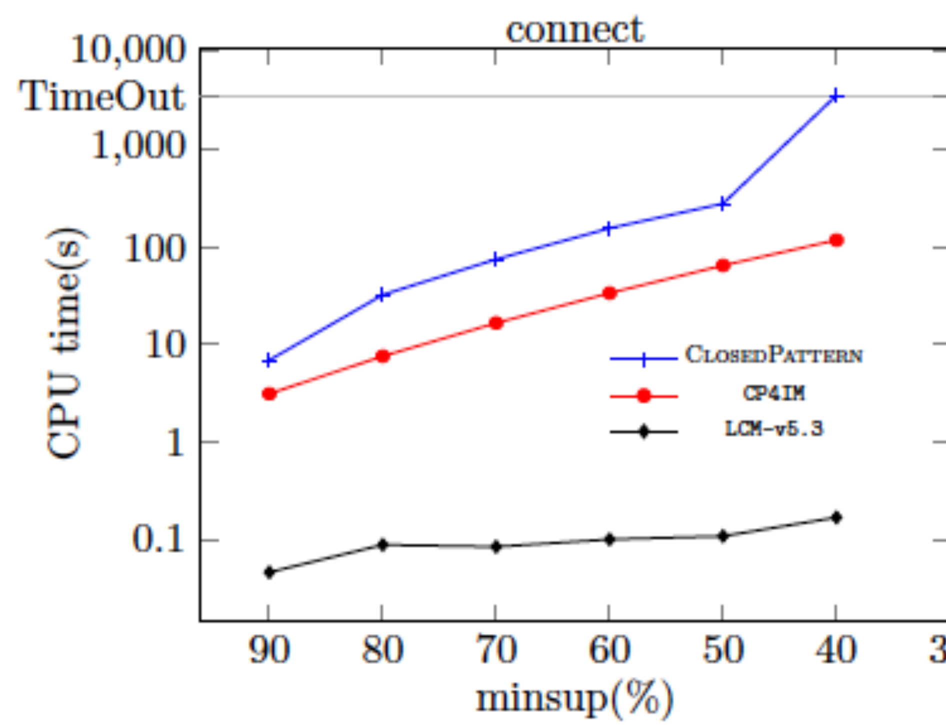
chess (θ = 80%, *lb* = 2, *ub* = 10)



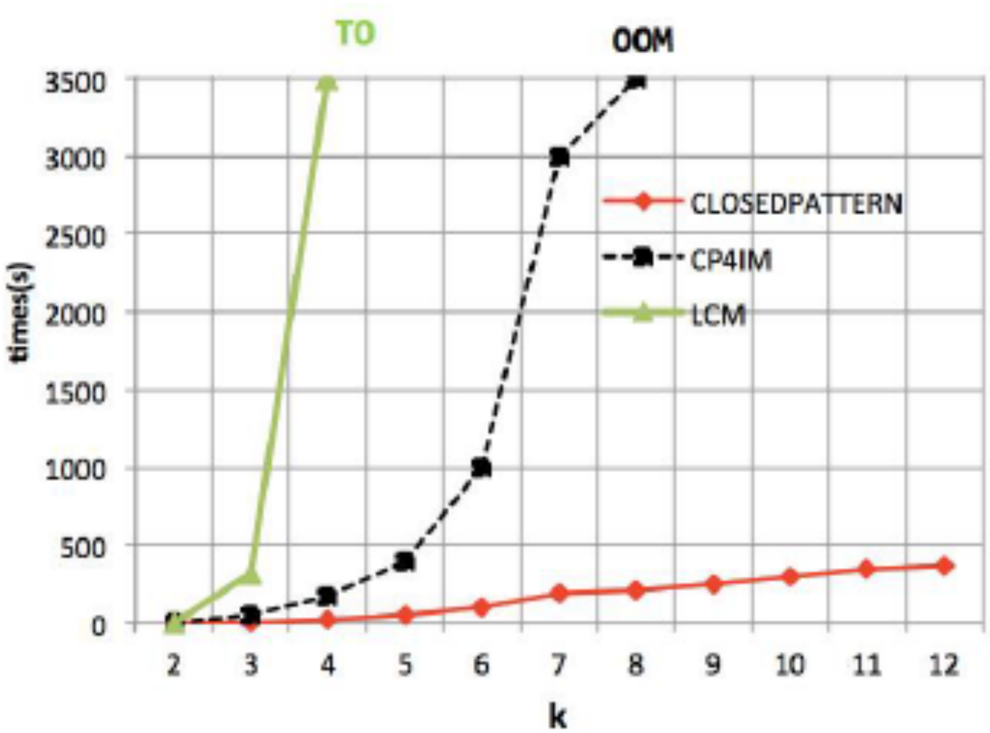
connect (θ = 90%, *lb* = 2, *ub* = 10)

SPECIALISED VS DECLARATIVE DATA MINING

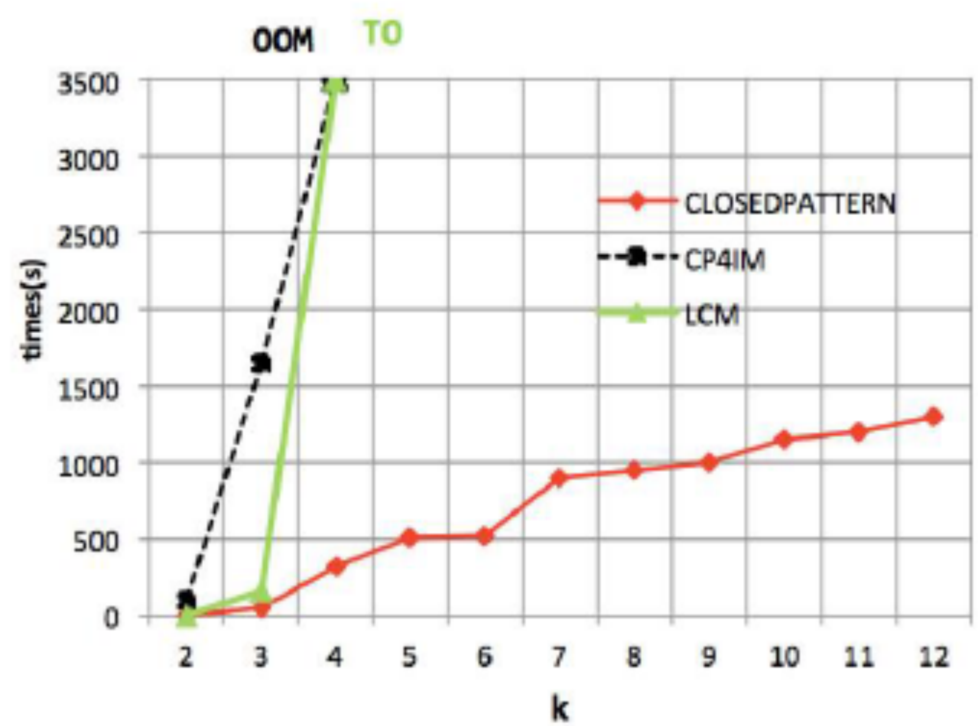




Specialised
is the winner!



chess (θ = 80%, *lb* = 2, *ub* = 10)



connect (θ = 90%, *lb* = 2, *ub* = 10)

Declarative
is the winner!

SPECIALISED VS DECLARATIVE DATA MINING

SPECIALISED VS DECLARATIVE DATA MINING

Preprocessing + Specialised step vs Declarative

Instances	$\#\mathcal{I}_i$	$\#\mathcal{T}_i$	(lb_I, ub_I)	(lb_T, ub_T)	$\#D$	#FCIs	PP-LCM	CP-ItemSet
Zoo_70_6	6	10	(2,3)	(2,3)	5,775	8	39.69	**1.75**
Zoo_50_11	6	10	(3,4)	(3,4)	11,550	9	88.66	**3.36**
Zoo_85_5	6	10	(2,6)	(2,10)	57,741	8	521.89	**31.86**
Primary_82_5	3	12	(2,3)	(2,10)	16,280	8	199.58	**36.13**
Vote_70_6	6	29	(2,3)	(2,3)	142,100	2	TO	**118.67**
Vote_72_5	8	29	(2,3)	(2,3)	341,040	2	TO	**201.79**
Mushroom_80_5	17	12	(2,2)	(2,2)	8,976	10	446.42	**102.68**
Mushroom_82_5	17	12	(2,2)	(3,3)	29,920	7	TO	**455.19**
Chess_90_16	5	34	(2,3)	(2,2)	11,220	3	286.42	**87.22**

TO: **timeout**

SPECIALISED VS DECLARATIVE DATA MINING

SPECIALISED VS DECLARATIVE DATA MINING

Specialised + postprocessing vs Declarative

Instances	*ub*	*lb*	ECLAT-Z-PP	SAT	CP	#TOT
Zoo_5	2	11	479.26	3.92	**0.36**	27
Zoo_5	1	9	491.48	0.17	**0.06**	12
Vote_5	4	8	37.69	282.25	**0.66**	13
Vote_5	1	2	38.49	1.41	**0.05**	23
Anneal_80	2	13	1567.48	1.14	**0.26**	76
Anneal_80	1	12	1622.19	0.53	**0.15**	73
Chess_60	2	9	280.60	2.17	**0.20**	20
Chess_60	1	8	284.22	1.07	**0.08**	24
Mushroom_10	1	11	249.00	47.52	**0.07**	14
Connect_90	1	11	61.80	30.41	**0.26**	12
T10_0.02	1	11	84.47	TO	**5.44**	0
T40_0.1	1	11	TO	TO	**8.33**	39
Pumsb_80	1	12	741.49	OOM	**0.34**	32

TO: timeout OOM: out-of-memory

CONCLUSIONS (PART I)

CONCLUSIONS (PART I)

- Specialised methods are suitable for:
  - Enumerating Patterns
  - Taking into account classic constraints (simple queries)

CONCLUSIONS (PART I)

- Specialised methods are suitable for:
  - Enumerating Patterns
  - Taking into account classic constraints (simple queries)
- Declarative methods are suitable for:
  - Taking into account user's constraints (complex queries)
  - Iterative data mining process

CONCLUSIONS (PART I)

- Specialised methods are suitable for:
  - Enumerating Patterns
  - Taking into account classic constraints (simple queries)
- Declarative methods are suitable for:
  - Taking into account user's constraints (complex queries)
  - Iterative data mining process

Time left?

FAULT LOCALISATION

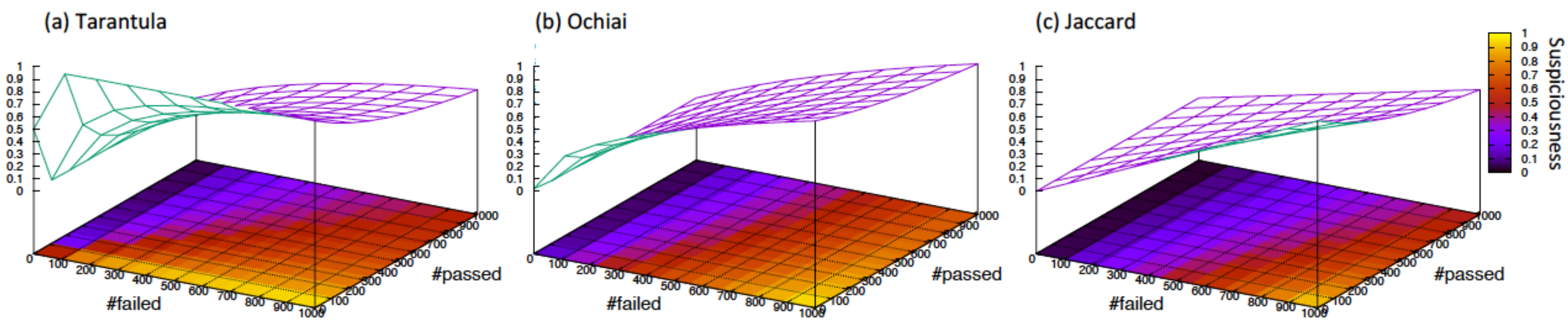
FAULT LOCALISATION

- The need: identify a subset of statements that are susceptible to explain a fault in a program
  - Precision <=> Efficiency

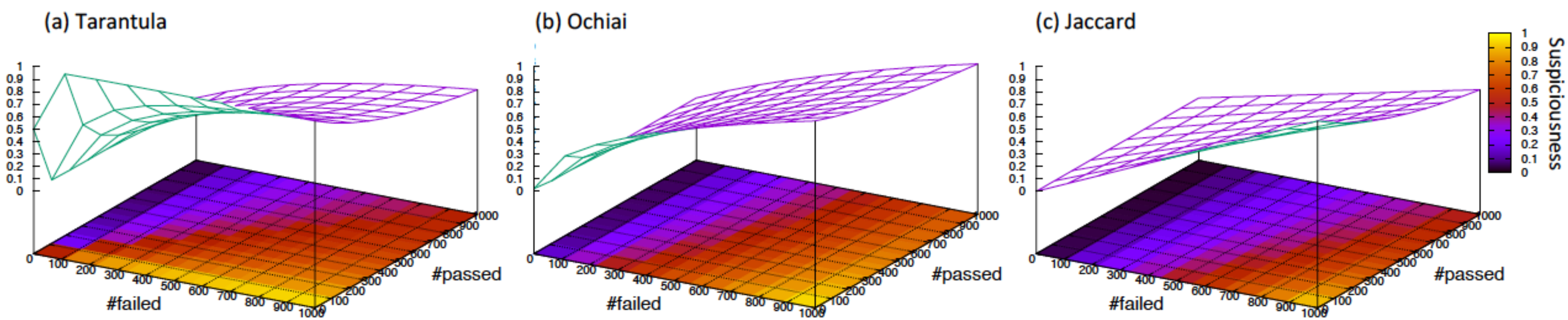
FAULT LOCALISATION

- The need: identify a subset of statements that are susceptible to explain a fault in a program
  - Precision <=> Efficiency
- Spectrum-based approaches: (ranking metrics - suspiciousness score)
  - Tarantula [Jones and Harrold 05]
  - Ochiai [Abreu et al. 07]
  - Jaccard [Abreu et al. 07]
  - …

FAULT LOCALISATION (MOTIVATIONS)

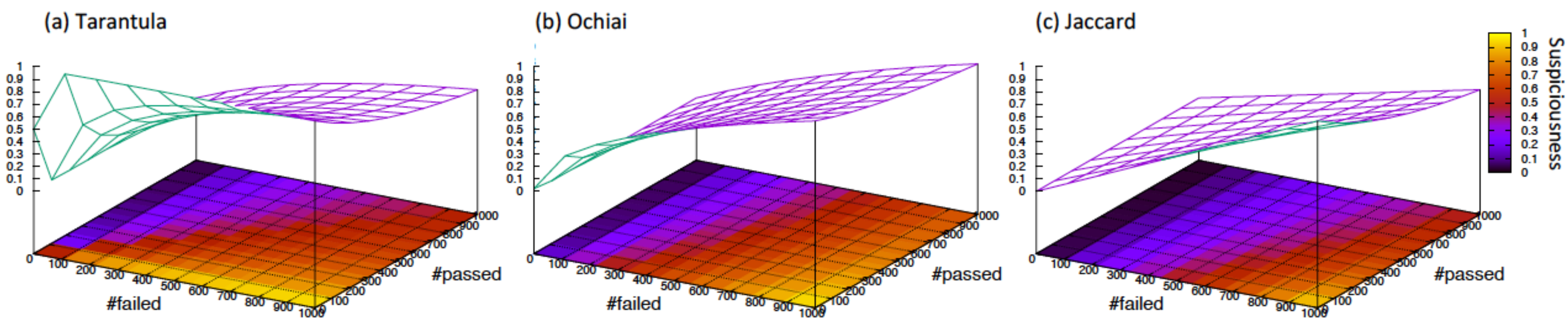


FAULT LOCALISATION (MOTIVATIONS)



- **Pros:** Quick localisation

FAULT LOCALISATION (MOTIVATIONS)



- **Pros:** Quick localisation
- **Cons:** independent evaluation of each statement at the expense of accuracy

FAULT LOCALISATION (MOTIVATIONS)

FAULT LOCALISATION (MOTIVATIONS)

Program : Character counter	Test cases							
	tc_1	tc_2	tc_3	tc_4	tc_5	tc_6	tc_7	tc_8
`function count (char *s) {`								
`int let, dig, other, i = 0;`								
`char c;`								
e_1: `while (c = s[i++]) {`	1	1	1	1	1	1	1	1
e_2: `if('A'<=c && 'Z'>=c)`	1	1	1	1	1	1	0	1
e_3: `let += 2; //- fault -`	1	1	1	1	1	1	0	0
e_4: `else if ( 'a'<=c && 'z'>=c )`	1	1	1	1	1	0	0	1
e_5: `let += 1;`	1	1	0	0	1	0	0	0
e_6: `else if ( '0'<=c && '9'>=c )`	1	1	1	1	0	0	0	1
e_7: `dig += 1;`	0	1	0	1	0	0	0	0
e_8: `else if (isprint (c))`	1	0	1	0	0	0	0	1
e_9: `other += 1;`	1	0	1	0	0	0	0	1
e_{10}: `printf("%d %d %d\n", let, dig, other);}`	1	1	1	1	1	1	1	1
Passing/Failing	F	F	F	F	F	F	P	P

FAULT LOCALISATION (MOTIVATIONS)

Program : Character counter	Test cases							
	tc_1	tc_2	tc_3	tc_4	tc_5	tc_6	tc_7	tc_8
`function count (char *s) {` `int let, dig, other, i = 0;` `char c;`								
e_1: `while (c = s[i++]) {`	1	1	1	1	1	1	1	1
e_2: `if('A'<=c && 'Z'>=c)`	1	1	1	1	1	1	0	1
e_3: `let += 2; //- fault -`	1	1	1	1	1	1	0	0
e_4: `else if ( 'a'<=c && 'z'>=c )`	1	1	1	1	1	0	0	1
e_5: `let += 1;`	1	1	0	0	1	0	0	0
e_6: `else if ( '0'<=c && '9'>=c )`	1	1	1	1	0	0	0	1
e_7: `dig += 1;`	0	1	0	1	0	0	0	0
e_8: `else if (isprint (c))`	1	0	1	0	0	0	0	1
e_9: `other += 1;`	1	0	1	0	0	0	0	1
e_{10}: `printf("%d %d %d\n", let, dig, other);}`	1	1	1	1	1	1	1	1
Passing/Failing	F	F	F	F	F	F	P	P

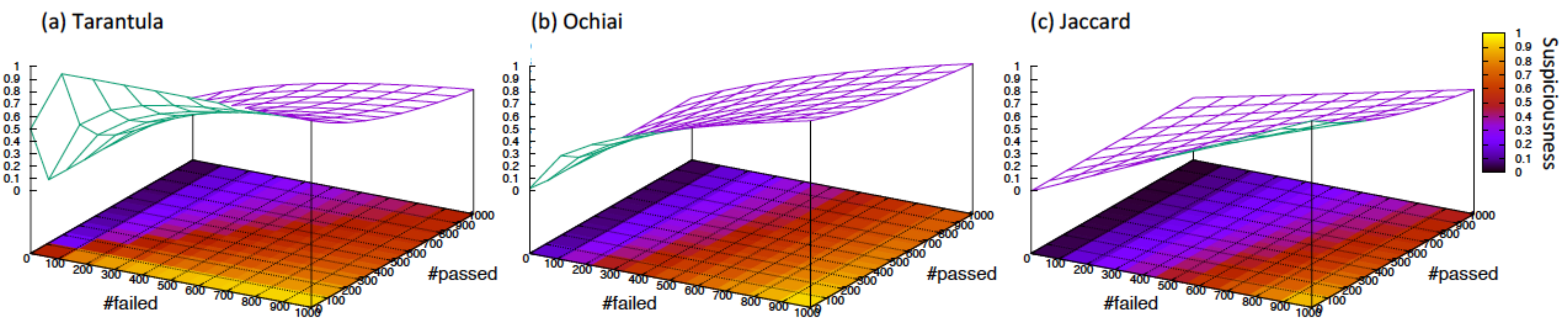
FAULT LOCALISATION (MOTIVATIONS)

Program : Character counter	Test cases tc_1	tc_2	tc_3	tc_4	tc_5	tc_6	tc_7	tc_8

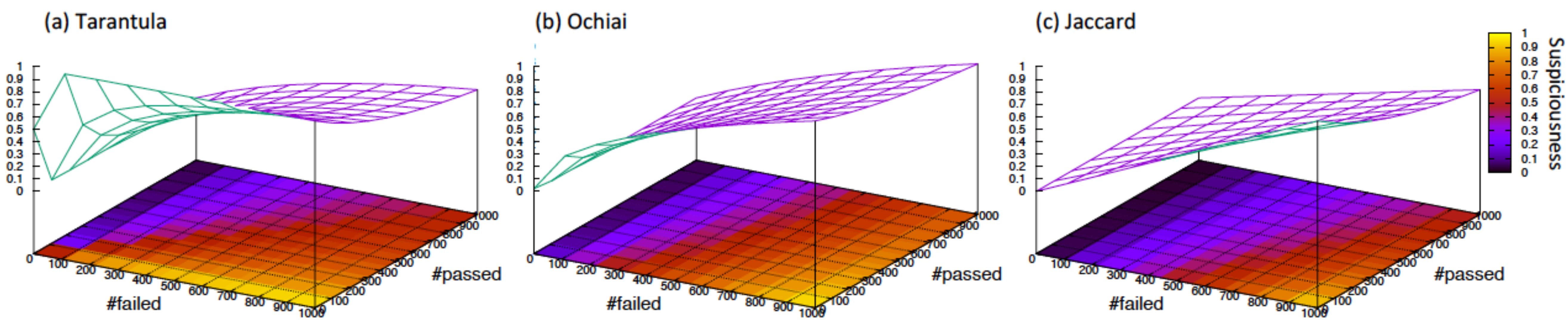
`function count (char *s) {`
 `int let, dig, other, i = 0;`
 `char c;`
 e_1: `while (c = s[i++]) {`
	1	1	1	1	1	1	1	1

e_2: `if('A'<=c && 'Z'>=c)`	1	1	1	1	1	1	0	1
e_3: `let += 2; //- fault -`	1	1	1	1	1	1	0	0
e_4: `else if ( 'a'<=c && 'z'>=c )`	1	1	1	1	1	0	0	1
e_5: `let += 1;`	1	1	0	0	1	0	0	0
e_6: `else if ( '0'<=c && '9'>=c )`	1	1	1	1	0	0	0	1
e_7: `dig += 1;`	0	1	0	1	0	0	0	0
e_8: `else if (isprint (c))`	1	0	1	0	0	0	0	1
e_9: `other += 1;`	1	0	1	0	0	0	0	1
e_{10}: `printf("%d %d %d\n", let, dig, other);}`	1	1	1	1	1	1	1	1
Passing/Failing	F	F	F	F	F	F	P	P

FAULT LOCALISATION (MOTIVATIONS)

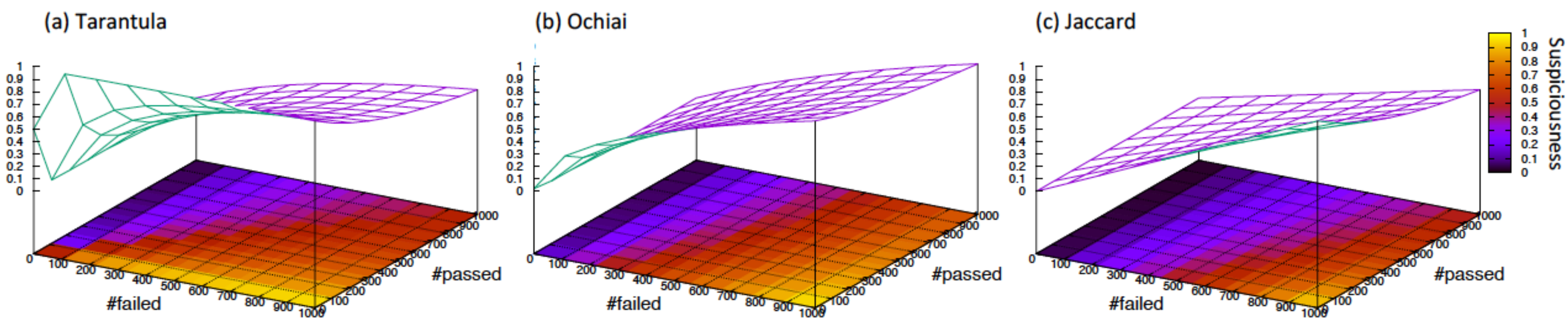


FAULT LOCALISATION (MOTIVATIONS)



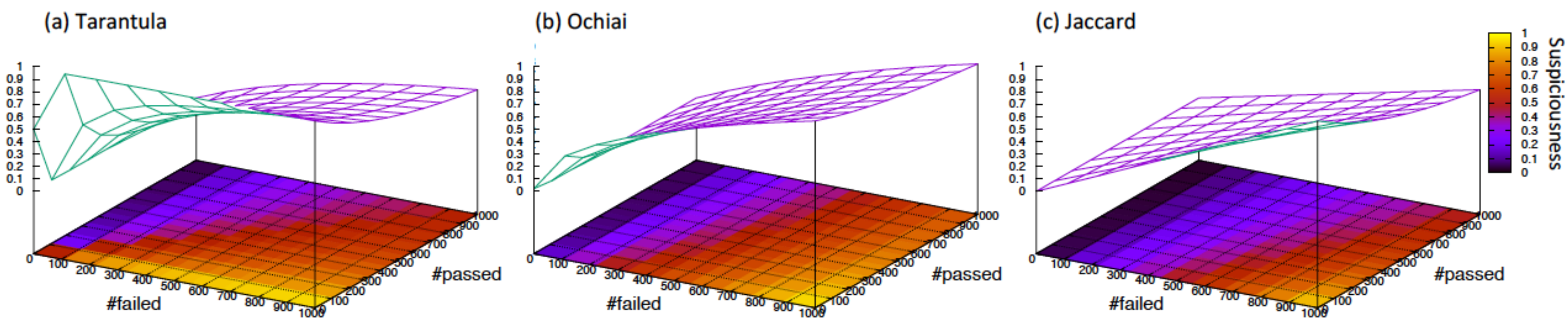
- **Pros:** Quick localisation

FAULT LOCALISATION (MOTIVATIONS)



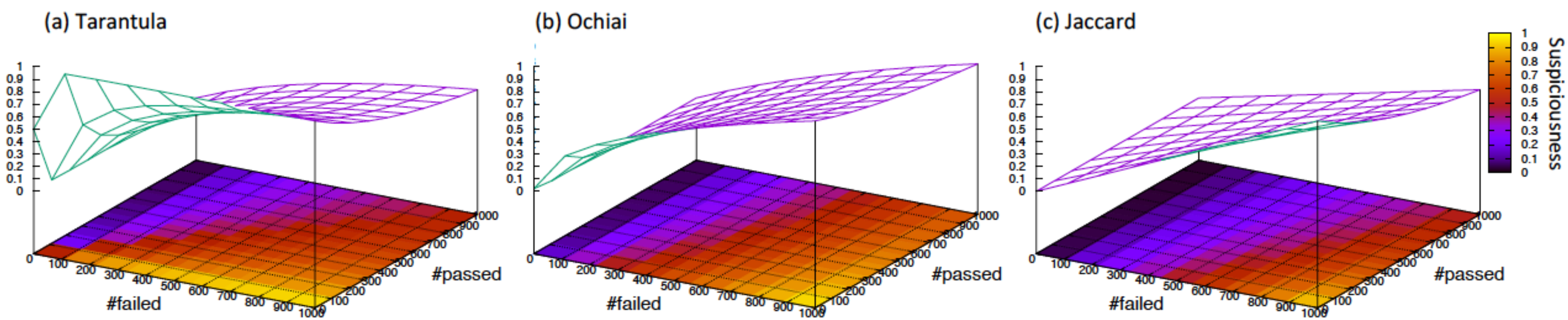
- **Pros:** Quick localisation
- **Cons:** independent evaluation of each statement at the expense of accuracy

FAULT LOCALISATION (MOTIVATIONS)



- **Pros:** Quick localisation
- **Cons:** independent evaluation of each statement at the expense of accuracy
- Need: more finer-grained localisation, taking into account user's constraints

FAULT LOCALISATION (MOTIVATIONS)



- **Pros:** Quick localisation
- **Cons:** independent evaluation of each statement at the expense of accuracy
- Need: more finer-grained localisation, taking into account user's constraints
- How: Use of Declarative Data Mining

FAULT LOCALISATION (MOTIVATIONS)

	Test cases							
Program : Character counter	tc_1	tc_2	tc_3	tc_4	tc_5	tc_6	tc_7	tc_8

`function count (char *s) {`
 `int let, dig, other, i = 0;`
 `char c;`
 e_1: `while (c = s[i++]) {`
	1	1	1	1	1	1	1	1

e_2: `if('A'<=c && 'Z'>=c)`	1	1	1	1	1	1	0	1
e_3: `let += 2; //- fault -`	1	1	1	1	1	1	0	0
e_4: `else if ( 'a'<=c && 'z'>=c )`	1	1	1	1	1	0	0	1
e_5: `let += 1;`	1	1	0	0	1	0	0	0
e_6: `else if ( '0'<=c && '9'>=c )`	1	1	1	1	0	0	0	1
e_7: `dig += 1;`	0	1	0	1	0	0	0	0
e_8: `else if (isprint (c))`	1	0	1	0	0	0	0	1
e_9: `other += 1;`	1	0	1	0	0	0	0	1
e_{10}: `printf("%d %d %d\n", let, dig, other);}`	1	1	1	1	1	1	1	1
Passing/Failing	F	F	F	F	F	F	P	P

FAULT LOCALISATION (MOTIVATIONS)

	Test cases							
Program : Character counter	tc_1	tc_2	tc_3	tc_4	tc_5	tc_6	tc_7	tc_8
`function count (char *s) {`								
`int let, dig, other, i = 0;`								
`char c;`								
e_1: `while (c = s[i++]) {`	1	1	1	1	1	1	1	1
e_2: `if('A'<=c && 'Z'>=c)`	1	1	1	1	1	1	0	1
e_3: `let += 2; //- fault -`	1	1	[illegible]	[illegible]	[illegible]	[illegible]	0	0
e_4: `else if ( 'a'<=c && 'z'>=c )`	1	1	[illegible]	[illegible]	[illegible]	[illegible]	0	1
e_5: `let += 1;`	1	1	[illegible]	[illegible]	[illegible]	[illegible]	0	0
e_6: `else if ( '0'<=c && '9'>=c )`	1	1	[illegible]	[illegible]	[illegible]	[illegible]	0	1
e_7: `dig += 1;`	0	1	0	1	0	0	0	0
e_8: `else if (isprint (c))`	1	0	1	0	0	0	0	1
e_9: `other += 1;`	1	0	1	0	0	0	0	1
e_{10}: `printf("%d %d %d\n", let, dig, other);}`	1	1	1	1	1	1	1	1
Passing/Failing	F	F	F	F	F	F	P	P

Fault localisation

=

Mining Task

PATTERN SUSPICIOUSNESS DEGREE (PSD)

PATTERN SUSPICIOUSNESS DEGREE (PSD)

- PSD function. Given a pattern P of a program:

$$\boxed{PSD(P) = freq^{-}(P) + \frac{|FAIL| - freq^{+}(P)}{|PASS| + 1}}$$

PATTERN SUSPICIOUSNESS DEGREE (PSD)

- PSD function. Given a pattern P of a program:

$$PSD(P) = freq^{-}(P) + \frac{|FAIL| - freq^{+}(P)}{|PASS|+1}$$

- PSD-dominance relation. Given two patterns P_i and P_j

$$P_i \rhd_{PSD} P_j \Leftrightarrow PSD(P_i) > PSD(P_j)$$

PATTERN SUSPICIOUSNESS DEGREE (PSD)

- PSD function. Given a pattern P of a program:

$$\boxed{PSD(P) = freq^{-}(P) + \frac{|FAIL| - freq^{+}(P)}{|PASS| + 1}}$$

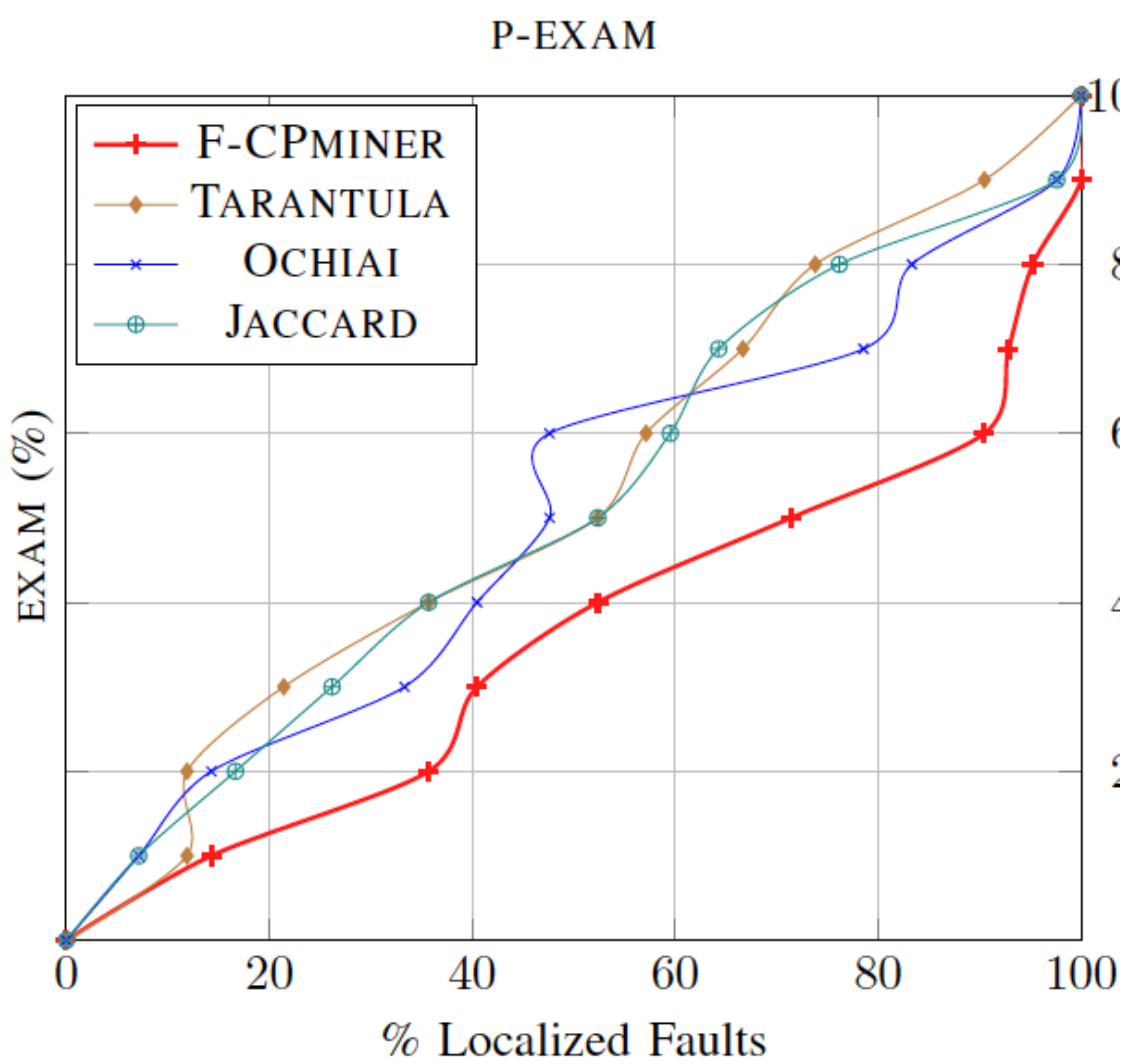
- PSD-dominance relation. Given two patterns P_i and P_j

$$\boxed{P_i \rhd_{PSD} P_j \Leftrightarrow PSD(P_i) > PSD(P_j)}$$

- Top-k suspicious patterns.

$$\boxed{\text{top-k} = \{P | \; \nexists P_1, \ldots, P_k : \forall 1 \leq j \leq k, \; P_j \rhd_{PSD} P\}}$$

FCP-MINER TOOL (SOME RESULTS)



CONCLUSIONS (PART II)

CONCLUSIONS (PART II)

- Software Testing/Program comprehension tasks can be tackled using Data Mining
  - Trace analysis
  - Test suites mining
  - Source code mining
  - …

CONCLUSIONS (PART II)

- Software Testing/Program comprehension tasks can be tackled using Data Mining
  - Trace analysis
  - Test suites mining
  - Source code mining
  - …
- Think about using Declarative methods in Software Testing

CONCLUSIONS (PART II)

- Software Testing/Program comprehension tasks can be tackled using Data Mining
  - Trace analysis
  - Test suites mining
  - Source code mining
  - …
- Think about using Declarative methods in Software Testing