Declarative Modeling for Machine Learning and Data Mining

Text

Lab for Declarative Languages and Artificial Intelligence

Joint work with especially
Tias Guns and Siegfried Nijssen

KATHOLIEKE UNIVERSITEIT LEUVEN

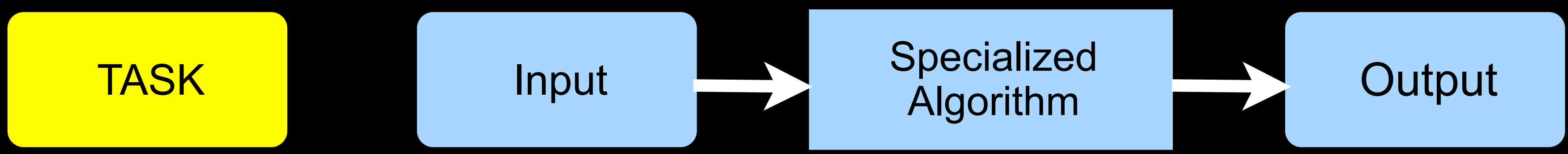
WARNING

CLAIMS MADE
ABOUT FCA
MAY BE NAIVE

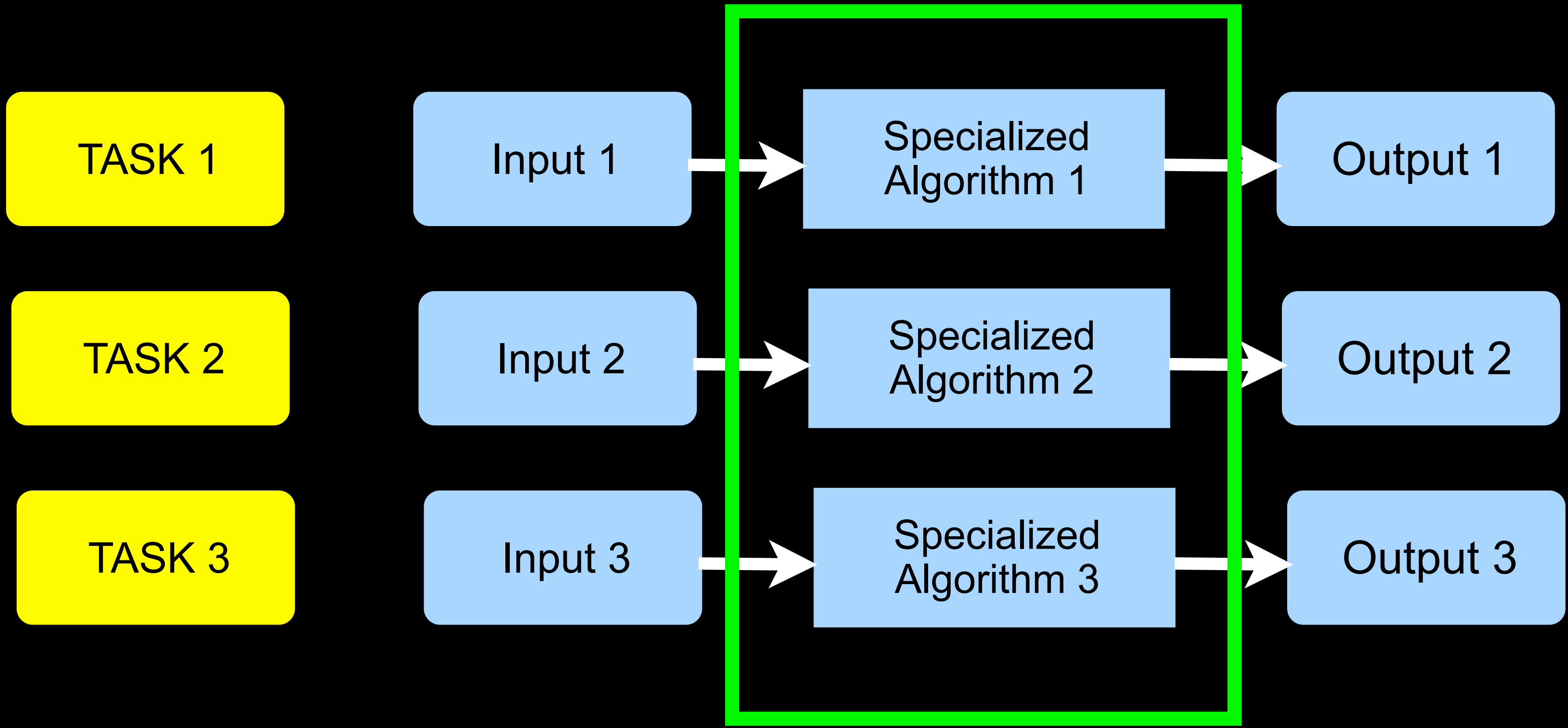
Developing software that relies on ML and DM

Is VERY HARD

1. Formalize learning / mining task
2. Design algorithm / technique to use
3. Implement the algorithm
4. Use the software



Developing software that relies on ML and DM



state HOW to solve problem

waste of resources

Developing software that relies on ML and DM

Is VERY HARD

- it requires a deep understanding of the underlying algorithms and procedures (i.e., be a ML/DM expert)
- existing algorithms/code are very specific and limited
- are hard to extend or generalize ML/DM
- there is only little re-use of existing code

Might be true for FCA as well ?

Long standing open questions

Can we design programming languages containing machine learning primitives?

Can a new generation of computer programming languages directly support writing programs that learn?

Why not design a new computer programming language that supports writing programs in which some subroutines are hand-coded while others are specified as "to be learned." Such a programming language could allow the programmer to declare the inputs and outputs of each "to be learned" subroutine, then select a learning algorithm from the primitives provided by the programming language.

Tom Mitchell, *The Discipline of Machine Learning*, 2006

Questions remain open

Though some relevant work on

- probabilistic & adaptive programming languages
- inductive query languages for data mining [Imielinski and Mannila, 95; EU cInQ and IQ projects]
- inductive logic programming and statistical relational learning
- Learning based Java [Roth et al. 10]
- kLog [Frasconi et al.]

We are still far away from programming languages that support machine learning or data mining

Our Vision

Declarative Modeling is KEY to answer the question

- Specify WHAT the problem IS
- Often as a constraint satisfaction or optimization problem

Instead of Procedural approaches

- Specify HOW the problem should be SOLVED
- Specify programs

Why declarative modeling ?

DECLARATIVE

- few lines of code
- easy to understand, maintain, change
- one theory for multiple tasks
- can be used with multiple "solvers", e.g., exact and approximate
- formal verification possible

PROCEDURAL

- 1000s of lines of code
- hard to understand, maintain or change
- one program for each task
- solver is built in the program

Plan for this talk

Declarative Modeling has never been systematically applied to ML/DM.

- Yet all the necessary ingredients are available to do this.
- So we are starting to develop this.

I will introduce some useful principles

- declarative languages and their connection to constraint programming / solver technology

Illustrations on

Might apply to FCA as well ?

- constraint-based item set mining
- probabilistic modeling (very fast)

Three observations

Observation 1

Machine learning and data mining are essentially constraint satisfaction and optimization problems

Data Mining

Given

- a database containing instances or transactions D

  the set of instances

- a hypothesis space or pattern language L
- a selection predicate, query or set of constraints Q

Find $Th(Q,L,D) = \{ h \in L \mid Q(h,D) = true \}$

Hello. Sign in to get personalized recommendations. New customer? Start here.

Want it by Dec. 24?

9 days left to order
using FREE Super Saver Shipping
Restrictions Apply

Your Amazon.com | Today's Deals | Gifts & Wish Lists | Gift Cards | Your Account | Help

Books

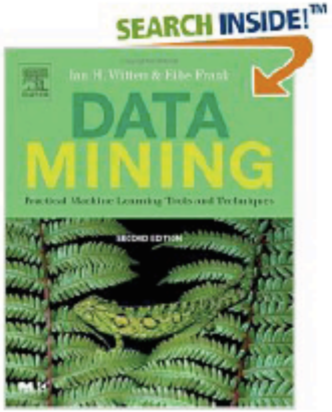
Books | Advanced Search | Browse Subjects | Hot New Releases | Bestsellers | The New York Times® Best Sellers | Libros En Español | Bargain Books | Textbooks

Prime

To get this item by **Wednesday**, Dec 12 order within 36hr 11min.

Amazon Prime gives you unlimited Free Shipping › learn more

Upgrade to FREE Two-Day Shipping on this item with Amazon Prime



Share your own customer images

Search inside this book

Are You an Author or Publisher?
Find out how to publish your own Kindle Books

Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems) (Paperback)

by Ian H. Witten (Author), Eibe Frank (Author) "Human in vitro fertilization involves collecting several eggs from a woman's ovaries, which, after fertilization with partner or donor sperm, produce several embryos..." (more)

Key Phrases: informational loss function, category utility formula, generic object editor, Naive Bayes, Name Function, Peter Peggy (more...)

(18 customer reviews)

List Price: $62.95

Price: $62.95 & this item ships for **FREE with Super Saver Shipping.** Details

Quantity: 1

Add to Shopping Cart

or

Sign in to turn on 1-Click ordering.

More Buying Choices

58 used & new from $36.00

Have one to sell? Sell yours here

Add to Wish List

Add to Shopping List

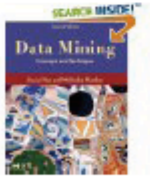
Add to Wedding Registry

Add to Baby Registry

Tell a friend

Customers Who Bought This Item Also Bought

Page 1 of 10



Data Mining, Second Edition, Second Editi... by Jiawei Han (24) $64.95

Introduction to Data Mining, (First Edition) by Pang-Ning Tan (7) $91.20

The Elements of Statistical Learning by T. Hastie (24) $76.30

Data Preparation for Data Mining (The Morg... by Dorian Pyle (11) $65.95

Pattern Recognition and Machine Learning (... by Christopher M. Bishop (24) $61.80

Any Category Algorithms Business & Investing Computer Mathematics

Computer Science Human Vision & Language Systems Internet

Natural Language Processing Professional Science Software Statistics

Theory of Computing

Itemset mining

Given

- a set of items I
- a transaction $t \subseteq I$.   *So,* $X = 2^I$
- D *is* a set of transactions.
- $L = X = 2^I$
- a frequency threshold c, *with* $freq(h,D) = |\{ d \mid d \in D, h \subseteq d \}|$

Find $Th(Q,L,D) = \{ h \in L \mid freq(h,D) > c \}$

Machine learning

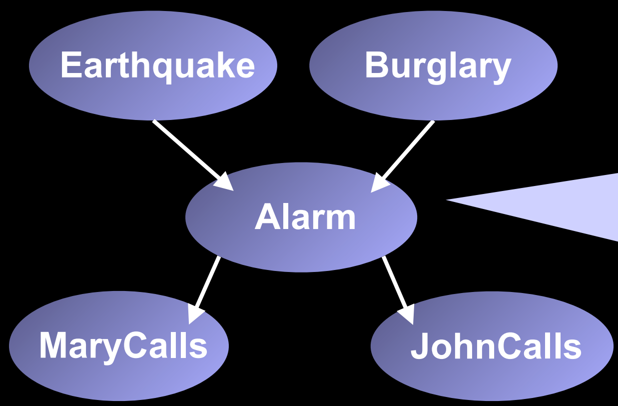
Given

- an unknown target function $f: X \to Y$
- a hypothesis space L containing functions $X \to Y$
- a dataset of examples $E = \{ (x, f(x)) \mid x \in X \}$
- a loss function *loss*$(h,E) \to \mathbb{R}$

Find $h \in L$ that minimizes *loss*(h,E)

supervised

Bayesian Networks



E	*B*	*P(A | B,E)*	
e	b	0.9	0.1
e	$\bar{b}$	0.2	0.8
$\bar{e}$	b	0.9	0.1
$\bar{e}$	$\bar{b}$	0.01	0.99

A graphical model encodes conditional independencies

P(E,B,A,J,M) = P(E).P(B).P(A|E,B).P(J|A).P(M|A)

P(e,not b, a, j, not m) =

P(e). P(not b). P(a|e, not b).P(j|a).P(not m|a)

Using the Joint Prob. Distribution,

any query P(Q |evidence) can be answered

Possible Dataset

B	E	A	J	M
true	true	?	true	false
?	true	?	?	false
...	...	...	...	...
true	false	?	false	true

Learning Probabilistic Models

Given

- an unknown target function P: X → Y   Y=[0,1]
- a hypothesis space L containing functions X → Y (graphical models)
- a dataset of examples E = { (x, _) | x ∈ X }   generative
- a loss function *loss*(h,E) → $\mathbb{R}$

$$\prod_{e \in E} P(e|h)$$

Find h ∈ L that minimizes *loss*(h,E)

maximize likelihood

generative

Observation 2

There has been an enormous progress in solver technology for basic constraint satisfaction and optimization problems

- SAT, ASP, CSP, Constraint programming, maxSAT, weighted model counting, ...
- Many problems are reduced to these basic problems ... and solved efficiently

What about ML/ DM ?

What about FCA ?

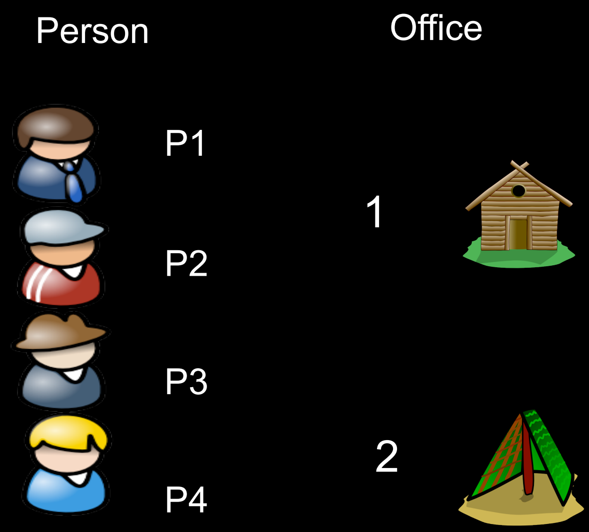
Constraint Satisfaction

Given

- a set of variables V
- the domain D(x) of all variables x in V
- a set of constraints C on values these variables can take

Find an assignment of values to variables in V that satisfies all constraints in C

Constraint Satisfaction



Variables:

P1,P2,P3,P4
with
domain {1 , 2 }

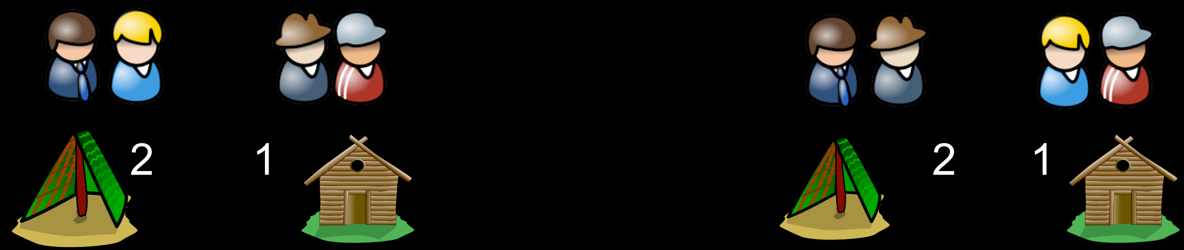
Constraints:

P1 != P2

P3 != P4

P1 != 1

Solutions:



Constraint Programming

Two key ideas

- propagation of constraints, e.g., from

  D(P1) = {1} and D(P2) = {1,2,3} and P1 != P2 infer that 1 ∉ D(P2) and simplify D(P2) = {2,3}

  propagator: if D(x) = {d} and x!=y then delete d from D(y)

- if you cannot propagate, instantiate (or divide) and recurse, e.g.,

  call with D(P2)={2} and with D(P2)={3}

  P2=2 P2=3

Search

Algorithm 1 Constraint-Search(D)

D :=propagate(D)
if D is a false domain **then**
  return
end if
if $\exists x \in \mathcal{V} : |D(x)| > 1$ **then**
  $x := \arg\min_{x \in \mathcal{V}, D(x) > 1} f(x)$
  for all $d \in D(x)$ **do**
    Constraint-Search($D \cup \{x \mapsto \{d\}\}$)
  end for
else
  Output solution
end if

D(P1) = {1,2}
D(P2) = {1,2}
D(P3) = {1,2}
D(P4) = {1,2}

P1 != P2

P3 != P4

P1 != 1

Search

Algorithm 1 Constraint-Search(D)

D :=propagate(D)
if D is a false domain **then**
    return
end if
if $\exists x \in \mathcal{V} : |D(x)| > 1$ **then**
    $x := \arg\min_{x \in \mathcal{V}, D(x) > 1} f(x)$
    for all $d \in D(x)$ **do**
        Constraint-Search($D \cup \{x \mapsto \{d\}\}$)
    end for
else
    Output solution
end if

D(P1) = {1,2}
D(P2) = {1,2}
D(P3) = {1,2}
D(P4) = {1,2}

P1 != P2

P3 != P4

P1 != 1

Search

Algorithm 1 Constraint-Search(D)

D :=propagate(D)
if D is a false domain **then**
    return
end if
if $\exists x \in \mathcal{V} : |D(x)| > 1$ **then**
    $x := \arg\min_{x \in \mathcal{V}, D(x) > 1} f(x)$
    for all $d \in D(x)$ **do**
        Constraint-Search($D \cup \{x \mapsto \{d\}\}$)
    end for
else
    Output solution
end if

D(P1) = {   2}
D(P2) = {1,2}
D(P3) = {1,2}
D(P4) = {1,2}

P1 != P2

P3 != P4

P1 != 1

Search

Algorithm 1 Constraint-Search(D)

```
D :=propagate(D)
if D  is a false domain then
    return
end if
if ∃x ∈ V : |D(x)| > 1 then
    x := arg min_{x∈V, D(x)>1} f(x)
    for all d ∈ D(x) do
        Constraint-Search(D ∪ {x ↦ {d}})
    end for
else
    Output solution
end if
```

D(P1) = {   2}
D(P2) = {1,2}
D(P3) = {1,2}
D(P4) = {1,2}

P1 != P2

P3 != P4

P1 != 1

Search

Algorithm 1 Constraint-Search(D)

D :=propagate(D)
if D is a false domain **then**
    return
end if
if $\exists x \in \mathcal{V} : |D(x)| > 1$ **then**
    $x := \arg\min_{x \in \mathcal{V}, D(x) > 1} f(x)$
    for all $d \in D(x)$ **do**
        Constraint-Search($D \cup \{x \mapsto \{d\}\}$)
    end for
else
    Output solution
end if

D(P1) = {   2}
D(P2) = {1   }
D(P3) = {1,2}
D(P4) = {1,2}

P1 != P2

P3 != P4

P1 != 1

choose P3 = 1

Search

Algorithm 1 Constraint-Search(D)

D :=propagate(D)
if D is a false domain **then**
    return
end if
if $\exists x \in \mathcal{V} : |D(x)| > 1$ **then**
    $x := \arg\min_{x \in \mathcal{V}, D(x) > 1} f(x)$
    for all $d \in D(x)$ **do**
        Constraint-Search($D \cup \{x \mapsto \{d\}\}$)
    end for
else
    Output solution
end if

D(P1) = { 2}
D(P2) = {1 }
D(P3) = {1 }
D(P4) = {1,2}

P1 != P2

P3 != P4

P1 != 1

Search

Algorithm 1 Constraint-Search(D)

D :=propagate(D)
if D is a false domain **then**
    return
end if
if $\exists x \in \mathcal{V} : |D(x)| > 1$ **then**
    $x := \arg\min_{x \in \mathcal{V}, D(x)>1} f(x)$
    for all $d \in D(x)$ **do**
        Constraint-Search($D \cup \{x \mapsto \{d\}\}$)
    end for
else
    Output solution
end if

D(P1) = { 2}
D(P2) = {1 }
D(P3) = {1 }
D(P4) = { 2}

P1 != P2

P3 != P4

P1 != 1

& backtrack

Search

Algorithm 1 Constraint-Search(D)

D :=propagate(D)
if D is a false domain **then**
    return
end if
if $\exists x \in \mathcal{V} : |D(x)| > 1$ **then**
    $x := \arg\min_{x \in \mathcal{V}, D(x) > 1} f(x)$
    for all $d \in D(x)$ **do**
        Constraint-Search($D \cup \{x \mapsto \{d\}\}$)
    end for
else
    Output solution
end if

D(P1) = {   2}
D(P2) = {1   }
D(P3) = {1,2}
D(P4) = {1,2}

P1 != P2

P3 != P4

P1 != 1

choose P3 = 2

Search

Algorithm 1 Constraint-Search(D)

D :=propagate(D)
if D is a false domain **then**
  return
end if
if $\exists x \in \mathcal{V} : |D(x)| > 1$ **then**
  $x := \arg\min_{x \in \mathcal{V}, D(x) > 1} f(x)$
  for all $d \in D(x)$ **do**
    Constraint-Search($D \cup \{x \mapsto \{d\}\}$)
  end for
else
  Output solution
end if

D(P1) = {   2}
D(P2) = {1   }
D(P3) = {   2}
D(P4) = {1,2}

P1 != P2

P3 != P4

P1 != 1

Search

Algorithm 1 Constraint-Search(D)

D :=propagate(D)
if D is a false domain **then**
    return
end if
if $\exists x \in \mathcal{V} : |D(x)| > 1$ **then**
    $x := \arg\min_{x \in \mathcal{V}, D(x) > 1} f(x)$
    for all $d \in D(x)$ **do**
        Constraint-Search($D \cup \{x \mapsto \{d\}\}$)
    end for
else
    Output solution
end if

D(P1) = { 2}
D(P2) = {1 }
D(P3) = { 2}
D(P4) = {1 }

P1 != P2

P3 != P4

P1 != 1

Constraint Programming

There is a lot more to say

- about propagators -- how to modify domains
- about choosing the next variable to instantiate
- about types of constraints and domains used
- about implementations ...
- about modeling languages ...

Observation 3

Solver technology facilitates the development of high-level declarative modeling languages

- specify the WHAT -- not the HOW
- systems processing constraints should find a solution satisfying the model

Examples include

- ZINC, Essence, constraint programming, ...

Very flexible approach ... not just in constraint programming ... convex optimisation

Main Claim

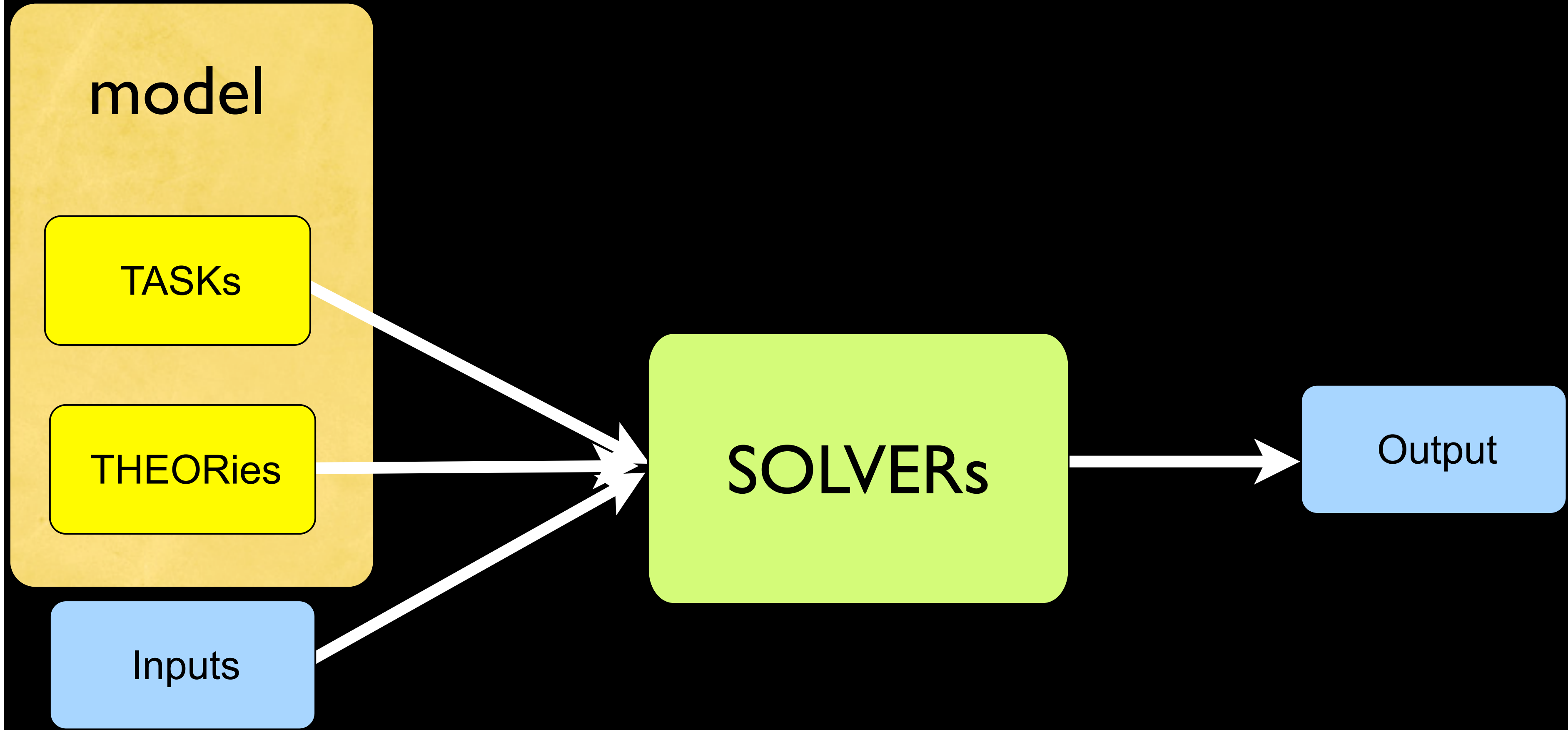
We can obtain programming languages for ML / DM by applying the same principles as constraint programming

Essentially three languages

- Modeling -- specify the problem -- the what
- Solver -- translation of the problem -- the how
- Programming -- in which everything is embedded

Translation is essential step !

How does it work



Only state WHAT the problem is

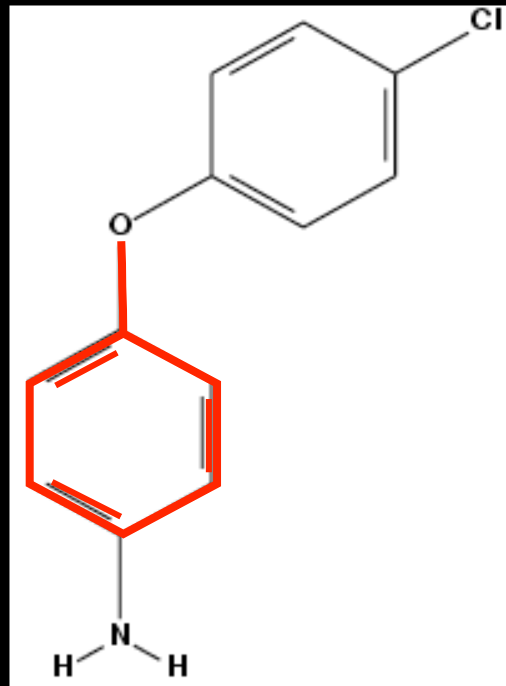
Data = Input

Pattern mining

Another example

Assume

- analysing a dataset
- e.g. molecules
- looking for patterns of interest
- patterns are subgraphs



Itemset Mining

Many interesting problems ... data mining as constraint satisfaction

- which patterns are frequent ?

frequent pattern mining

- which patterns are frequent in the active and infrequent in the inactive compounds ? and do not contain any halogens ? or benzene rings ?
- which patterns are significant w.r.t. classes ?

correlated pattern mining

- all patterns ? k-best patterns ?
- which pattern *set* is the *best* concept-description for the actives ? for the inactives ?

pattern set mining

still no general system that can do all of this

Pattern mining

- Traditional pattern mining:
  $Th(\mathcal{L}, Q, \mathcal{D}) = \{p \in \mathcal{L} | Q(p, \mathcal{D}) = true\}$
- Correlated pattern mining with function $\phi(p, \mathcal{D})$, (χ^2),
  $Th(\mathcal{L}, Q, \mathcal{D}) = \arg_{p \in \mathcal{L}} \max_k \phi(p, \mathcal{D})$
- Pattern set mining
  $Th(\mathcal{L}, \mathcal{Q}, \mathcal{D}) = \{P \subseteq \mathcal{L} | \mathcal{Q}(P, \mathcal{D}) = true\}$

Queries/Predicates *Q* employ *constraints*
such as frequency, generality, closedness, ...

Constraint-Based Mining

Numerous constraints have been used

Numerous systems have been developed

And yet,

- new constraints often require new implementations
- very hard to combine different constraints

There is not yet a modeling language for CBM

Again an analogy with FCA ?

Constraint Programming

Exists since about 20 ? years

A general and generic methodology for dealing with constraints across different domains

Efficient, extendable general-purpose systems exist, and key principles have been identified

Surprisingly CP has not been used for data mining ?

CP systems often more elegant, more flexible and more efficient than special purpose systems

I will argue that this is also true for Data Mining !

Yields a programming/modeling language for CBM

Results in Itemset mining

Use Constraint Programming for

1) Local Pattern Mining (using itemsets)

2) Correlated Pattern Mining (top-k)

3) Mining Patterns Sets (submitted)

[KDD 08, KDD 09, ECML/PKDD 10, AAAI 10, AIJ 11, IEEE TKDE 11]

Results by Guns, Nijssen and De Raedt

Provides evidence for main claims !

Itemset mining

Let's try to apply CP for item-set mining,

the simplest form of data mining

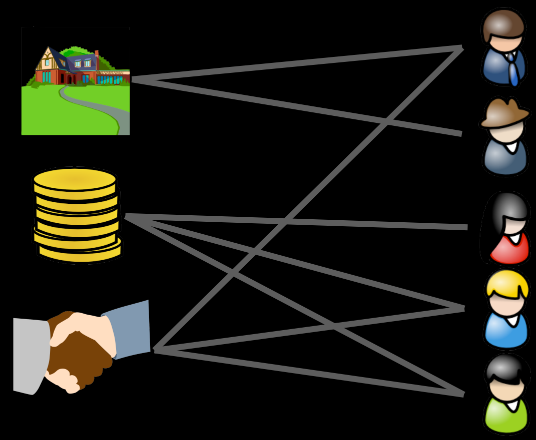
$$Th(\mathcal{L}, Q, \mathcal{D}) = \{p \in \mathcal{L} | Q(p, \mathcal{D}) = true\}$$

- $\mathcal{L} = 2^{\mathcal{I}}$, i.e., itemsets
- $\mathcal{D} \subset \mathcal{L}$, i.e., transactions
- $Q(p, \mathcal{D}) = true$ if $freq(p, \mathcal{D}) \geq t$

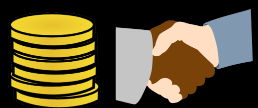
Data Set

Items

Transactions



$D_{ti} = 0 \text{ or } 1$

frequency  =2

Frequent Item Set Mining in MiningZinc

```
int: NrI; int: NrT; int: Freq;
array[1..NrT] of set of int: D;

var set of 1..NrI: Itemset;
var set of 1..NrT: Trans;

constraint card(Trans) >= Freq;

constraint forall (t in ub(Trans)) (
   t in Trans ↔ Itemset subset D[t] )

solve satisfy;
```

Math-like notation

User defined constraints

Efficient solving

Possible to efficiently translate this using the techniques to follow for a wide range of constraints

Specifying the WHAT -- how to translate ?

Closed Freq. Itemset Mining

```
int: NrI; int: NrT; int: Freq;
array[1..NrT] of set of int: D;

var set of 1..NrI: Itemset;
var set of 1..NrT: Trans;

constraint card(Trans) >= Freq;

constraint Trans = cover(Itemset, D);
constraint Itemset = cover_inv(Trans, D);

solve satisfy;
```

- Closure constraints:

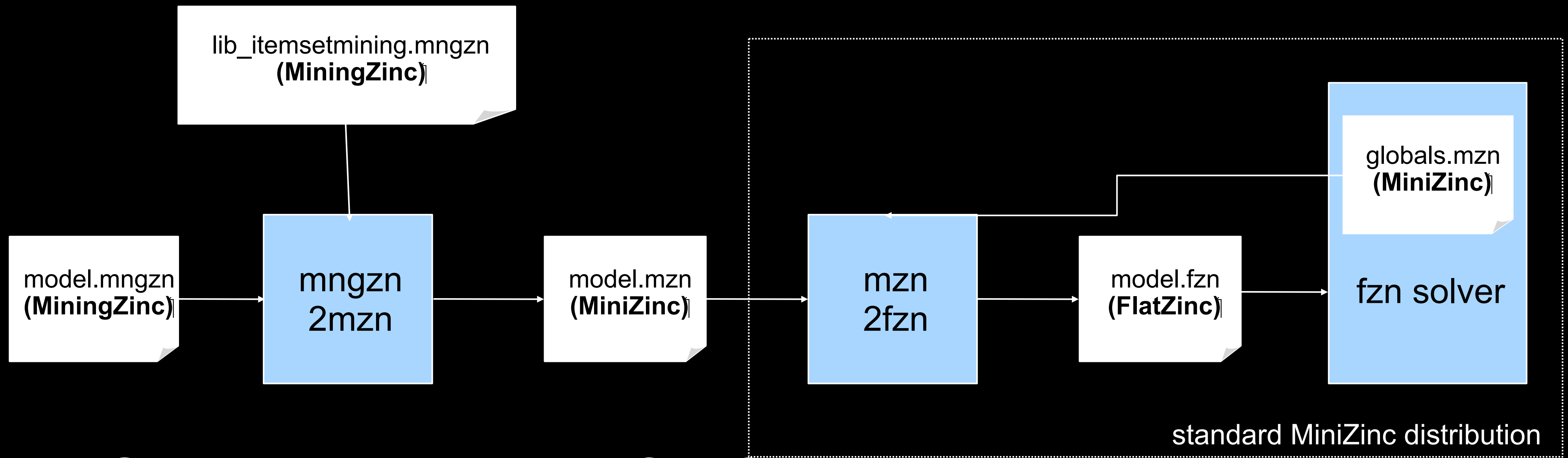
```
function var set of int: cover(Itemset, D) = let {
        var set of int: Trans,
        constraint forall (t in ub(Trans)) (
  t in Trans ↔ Itemset subset D[t] )
} in Trans;

function var set of int: cover_inv(Trans, D) = let {
        var set of int: Itemset,
        constraint forall (i in ub(Itemset)) (
  i in Itemset ↔ Trans subset D[i] )
} in Itemset;
```

MiningZinc

```
int: NrI; int: NrT; int: Freq;
array[1..NrT] of set of int: D;

var set of 1..NrI: Itemset;
var set of 1..NrT: Trans;

constraint card(Trans) >= Freq;

constraint forall (t in ub(Trans)) (
   t in Trans ↔ Itemset subset
D[t] )

solve satisfy;
```

```
function var set of int: frequency(Itemset, D) = …
function var set of int: cover(Itemset, D) = …
```

- Math-like notation
- User-defined constraints
- Efficient solving



- Solver independent: CP, SAT, MIP, spec. solvers, ...

The Model in Essence'

Algorithm 1 The basic fim_cp model in Essence'

```
given NrT, NrI : int
given TDB : matrix indexed by [int(1..NrT),int(1..NrI)] of bool
given Freq : int
find Items : matrix indexed by [int(1..NrI)] of bool
find Trans : matrix indexed by [int(1..NrT)] of bool
such that
$ encode TDB: every Trans its complement has no supported Items
forall t: int(1..NrT).
  Trans[t] <=> ((sum i: int(1..NrI). !TDB[t,i]*Items[i]) <= 0),
$ frequency: every Item is supported by sufficently many Trans
forall i: int(1..NrI).
  Items[i] => ((sum t: int(1..NrT). TDB[t,i]*Trans[t]) >= Freq)
```

Solver language
Translated model

$$\forall t : T_t = 1 \Leftrightarrow \sum_i I_i(1 - D_{ti}) = 0$$

$$\sum_t T_t \geq minsup \quad \text{iff} \quad \forall i : I_i = 1 \Rightarrow \sum_t T_t D_{ti} \geq minsup$$

We use Gecode !

Encoding a Data Set

Vectors as itemsets *I_i = 0 or 1*   0   1   1

and transactionsets *T_t = 0 or 1*   0   0   0   1   1

Goal find all itemsets *(I,T) such that*

- *I is frequent & I covers exactly T's transactions*
- frequency(I,D) > Freq AND T = covers(Itemset,D)

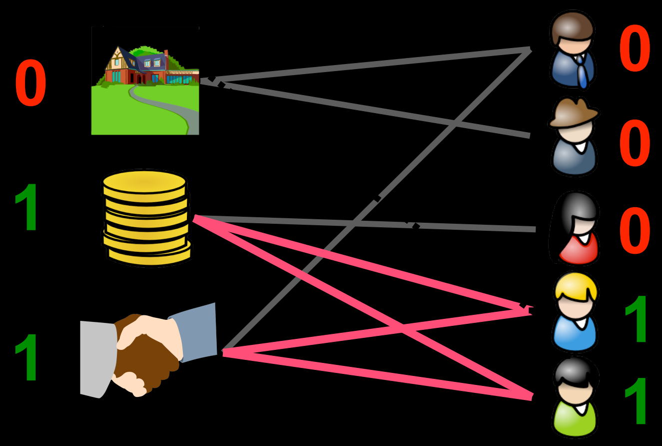
Encoding a Data Set

frequent

$$\sum_t T_t \geq minsup$$

exact coverage=

T is extension of *I*



$$T_t = 1 \Leftrightarrow \sum_i I_i(1 - D_{ti}) = 0$$ reified constraint

$$\text{for all } i : I_i = 0 \text{ or } (I_i = 1 \text{ and } (1 - D_{ti}) = 0)$$
$$\text{for all } i : I_i = 0 \text{ or } (I_i = 1 \text{ and } D_{ti} = 1)$$

where $D_{ti=1}$ if transaction t contains item i

Reified Frequency

	i1 0/1	i2 0/1	i3 0/1	i4 0/1
t1 0/1	1	0	1	1
t2 0/1	1	1	0	1
t3 0/1	0	0	1	1

$$\forall i : I_i = 1 \Rightarrow \sum_t T_t D_{ti} \geq minsup$$

IF i1=1 THEN t1+t2 ≥ freq

Exact Coverage

	i1 0/1	i2 0/1	i3 0/1	i4 0/1
t1 0/1	1	0	1	1
t2 0/1	1	1	0	1
t3 0/1	0	0	1	1

$$\forall t : T_t = 1 \Leftrightarrow \sum_i I_i(1 - D_{ti}) = 0$$

IF t1=1 THEN i2=0

One Propagator

Reified constraints of the form $C \Leftrightarrow x$.

- decompose into $C \Rightarrow x$ and $C \Leftarrow x$
- for $C \Rightarrow x$ do:
  - IF $0 \in D(x)$ and C THEN delete 0 from $D(x)$
  - IF $D(x) = 0$ THEN apply propagators for $\neg C$
- for $C \Leftarrow x$ do:
  - IF $1 \in D(x)$ and $\neg C$ THEN delete 1 from $D(x)$
  - IF $D(x) = 1$ THEN apply propagators for C

Another Propagator

Summation constraint: $\sum_{x \in V} w_x x \geq \theta$
with variables V and real-valued weights w_x

Define $x^{max} = \max_{d \in D(x)} d$ and $x^{min} = \min_{d \in D(x)} d$
$V^+ = \{x \in V | w_x \geq 0\}$ and $V^- = \{x \in V | w_x < 0\}$.

Then
$\sum_{x \in V^-} w_x x^{min} + \sum_{x \in V^+} w_x x^{max} \geq \theta$
must be satisfied

Another Propagator

IF $\sum_{x \in V^-} w_x x^{min} + \sum_{x \in V^+} w_x x^{max} \geq \theta$

  IF $\sum_{x \in V^-} w_x x^{min} + \sum_{x \in V^+ \setminus \{x'\}} w_x x^{max} < \theta$

  THEN $D(x') = \{1\}$

  ENDIF

ELSE $D(x') = \emptyset$

ENDIF

$$x_1 + x_2 + x_3 \geq 2,$$
$$D(x_1) = \{1\}, D(x_2) = \{0,1\}, D(x_3) = \{0,1\};$$

One of x_2 and x_3 must have the value 1, but if

$$x_1 + x_2 + x_3 \geq 3,$$
$$D(x_1) = \{1\}, D(x_2) = \{0,1\}, D(x_3) = \{0,1\};$$

the propagator determines that $D(x_2) = D(x_3) = \{1\}$.

Exact Coverage

	i1 0/1	i2 0/1	i3 0/1	i4 0/1
t1 0/1	1	0	1	1
t2 0/1	1	1	0	1
t3 0/1	0	0	1	1

$$\forall t : T_t = 1 \Leftrightarrow \sum_i I_i(1 - D_{ti}) = 0$$

IF t1=1 THEN i2=0

Reified Frequency

	i1 0/1	i2 0/1	i3 0/1	i4 0/1
t1 0/1	1	0	1	1
t2 0/1	1	1	0	1
t3 0/1	0	0	1	1

$$\forall i : I_i = 1 \Rightarrow \sum_t T_t D_{ti} \geq minsup$$

IF i1=1 THEN t1+t2 ≥ freq

Example

propagate i2 freq

	i1 0/1	i2 0/1	i3 0/1	i4 0/1
t1 0/1	1	0	1	1
t2 0/1	1	1	0	1
t3 0/1	0	0	1	1

$$\forall t : T_t = 1 \Leftrightarrow \sum_i I_i(1 - D_{ti}) = 0$$

$$\forall i : I_i = 1 \Rightarrow \sum_t T_t D_{ti} \geq minsup$$

Example

propagate t1 coverage

	i1 0/1	i2 0	i3 0/1	i4 0/1
t1 0/1	1	0	1	1
t2 0/1	1	1	0	1
t3 0/1	0	0	1	1

$$\forall t : T_t = 1 \Leftrightarrow \sum_i I_i(1 - D_{ti}) = 0$$

$$\forall i : I_i = 1 \Rightarrow \sum_t T_t D_{ti} \geq minsup$$

Example

		i1	i2	i3	i4
		1	0	0/1	0/1
t1	1	1	0	1	1
t2	0/1	1	1	0	1
t3	0/1	0	0	1	1

branch i1 =1

$$\forall t : T_t = 1 \Leftrightarrow \sum_i I_i(1 - D_{ti}) = 0$$

$$\forall i : I_i = 1 \Rightarrow \sum_t T_t D_{ti} \geq minsup$$

Example

		i1	i2	i3	i4
		1	0	0/1	0/1
t1	1	1	0	1	1
t2	0/1	1	1	0	1
t3	0	0	0	1	1

propagate t3 coverage

$$\forall t : T_t = 1 \Leftrightarrow \sum_i I_i(1 - D_{ti}) = 0$$

$$\forall i : I_i = 1 \Rightarrow \sum_t T_t D_{ti} \geq minsup$$

Example

propagate i3 freq

		i1	i2	i3	i4
		1	0	0	0/1
t1	1	1	0	1	1
t2	0/1	1	1	0	1
t3	0	0	0	1	1

$$\forall t : T_t = 1 \Leftrightarrow \sum_i I_i(1 - D_{ti}) = 0$$

$$\forall i : I_i = 1 \Rightarrow \sum_t T_t D_{ti} \geq minsup$$

Example

propagate t2 coverage

		i1	i2	i3	i4
		1	0	0	0/1
t1	1	1	0	1	1
t2	1	1	1	0	1
t3	0	0	0	1	1

$$\forall t : T_t = 1 \Leftrightarrow \sum_i I_i(1 - D_{ti}) = 0$$

$$\forall i : I_i = 1 \Rightarrow \sum_t T_t D_{ti} \geq minsup$$

Example

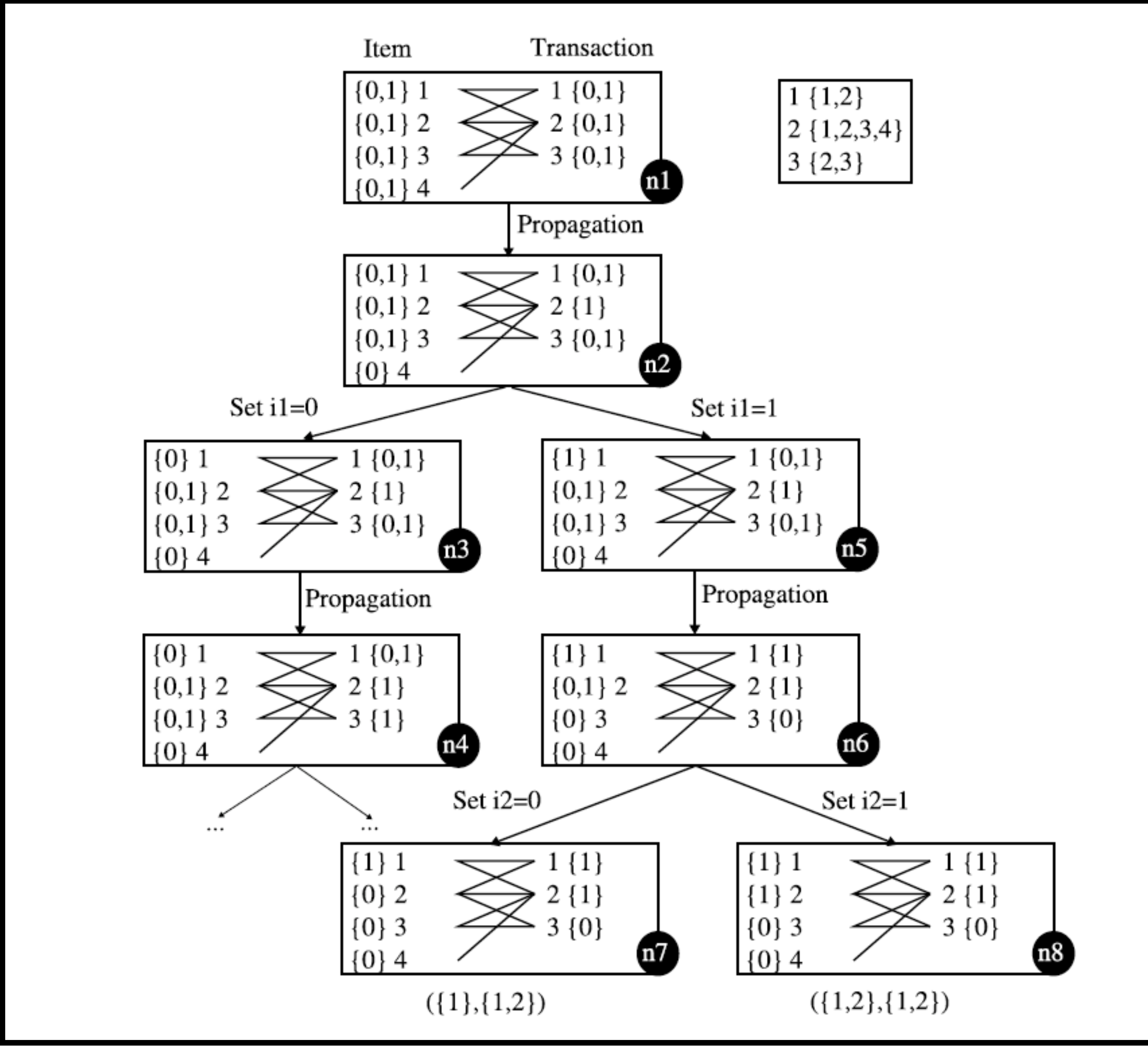
propagate i4 freq

		i1	i2	i3	i4
		1	0	0	1
t1	1	1	0	1	1
t2	1	1	1	0	1
t3	0	0	0	1	1

$$\forall t : T_t = 1 \Leftrightarrow \sum_i I_i(1 - D_{ti}) = 0$$

$$\forall i : I_i = 1 \Rightarrow \sum_t T_t D_{ti} \geq minsup$$

Search Tree



Item
Transaction
{0,1} 1
{0,1} 2
{0,1} 3
{0,1} 4
1 {0,1}
2 {0,1}
3 {0,1}
n1
1 {1,2}
2 {1,2,3,4}
3 {2,3}
Propagation
{0,1} 1
{0,1} 2
{0,1} 3
{0} 4
1 {0,1}
2 {1}
3 {0,1}
n2
Set i1=0
Set i1=1
{0} 1
{0,1} 2
{0,1} 3
{0} 4
1 {0,1}
2 {1}
3 {0,1}
n3
{1} 1
{0,1} 2
{0,1} 3
{0} 4
1 {0,1}
2 {1}
3 {0,1}
n5
Propagation
Propagation
{0} 1
{0,1} 2
{0,1} 3
{0} 4
1 {0,1}
2 {1}
3 {1}
n4
{1} 1
{0,1} 2
{0} 3
{0} 4
1 {1}
2 {1}
3 {0}
n6
...
...
Set i2=0
Set i2=1
{1} 1
{0} 2
{0} 3
{0} 4
1 {1}
2 {1}
3 {0}
n7
{1} 1
{1} 2
{0} 3
{0} 4
1 {1}
2 {1}
3 {0}
n8
({1},{1,2})
({1,2},{1,2})

Further Constraints

monotonic and anti-monotonic

emerging patterns (use two datasets)

(delta)-closed sets and (delta)-free sets

correlated patterns (e.g. significant patters)

maximal sets

convertible constraints (e.g. min average cost item)

as well as numerous combinations possible

Exact Coverage (always needed)

$$T_t = 1 \Leftrightarrow \sum_i I_i(1 - D_{ti}) = 0$$

Frequent Itemsets

Easy to change !

$$I_i = 1 \Rightarrow \sum_t D_{ti} T_t \geq minsup$$

Maximal Itemsets (supersets are not frequent)

$$I_i = 1 \boxed{\Leftrightarrow} \sum_t D_{ti} T_t \geq minsup$$

Closed Itemsets (supersets have strictly lower frequency

$$\mathrm{I}_i = 1 \Leftrightarrow \textstyle\sum_t T_t(1 - D_{ti}) = 0$$ + Frequency

delta Closed Itemsets

$$\mathrm{I}_i = 1 \Leftrightarrow \textstyle\sum_t T_t(1 - \boxed{\delta} - D_{ti}) \boxed{\leq} 0$$ + Frequency

Other Systems

	LCM [15]	MAFIA [6]	ExAMiner [4]	DualMiner [5]	CP
Constraints on data					
Minimum frequency	X	X	X	X	X
Maximum frequency				X	X
Emerging patterns					X
Condensed Representations					
Maximal	X	X		X	X
Closed	X	X			X
$\delta-$Closed					X
Constraints on syntax					
Max/Min total cost			X	X	X
Minimum average cost			X		X
Max/Min size	X	X	X	X	X

Table 1: Comparison of Itemset Miners

most flexible system today CP 4 IM - downloadable

Experiments

	#Trans.	#Items	Density	#Patterns 1%
German Credit	1000	77	0.28	29 088 485
Letter	20000	74	0.33	1 037 221 530
Segment	2310	74	0.51	(time out)

Compared to
LCM
Mafia
Patternist

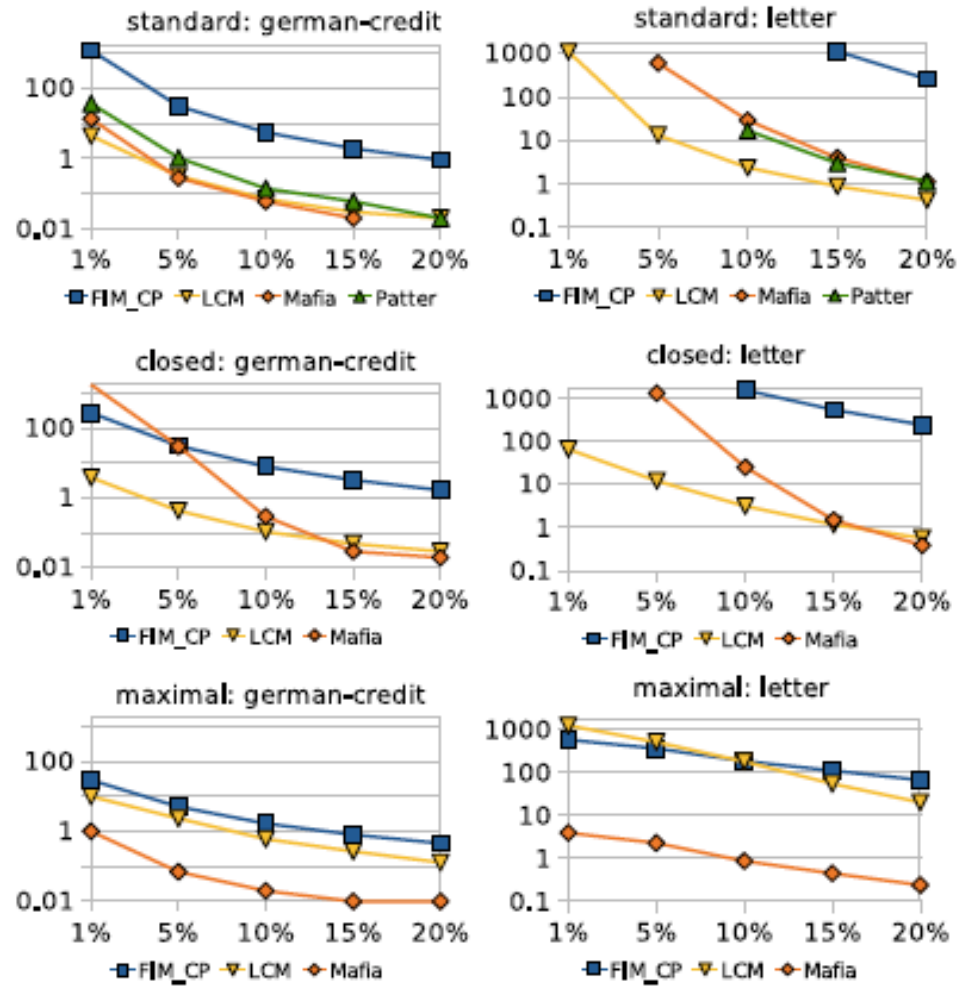


Figure 3: Runtimes of itemset miners on standard problems for different values of minimum support

Experiments

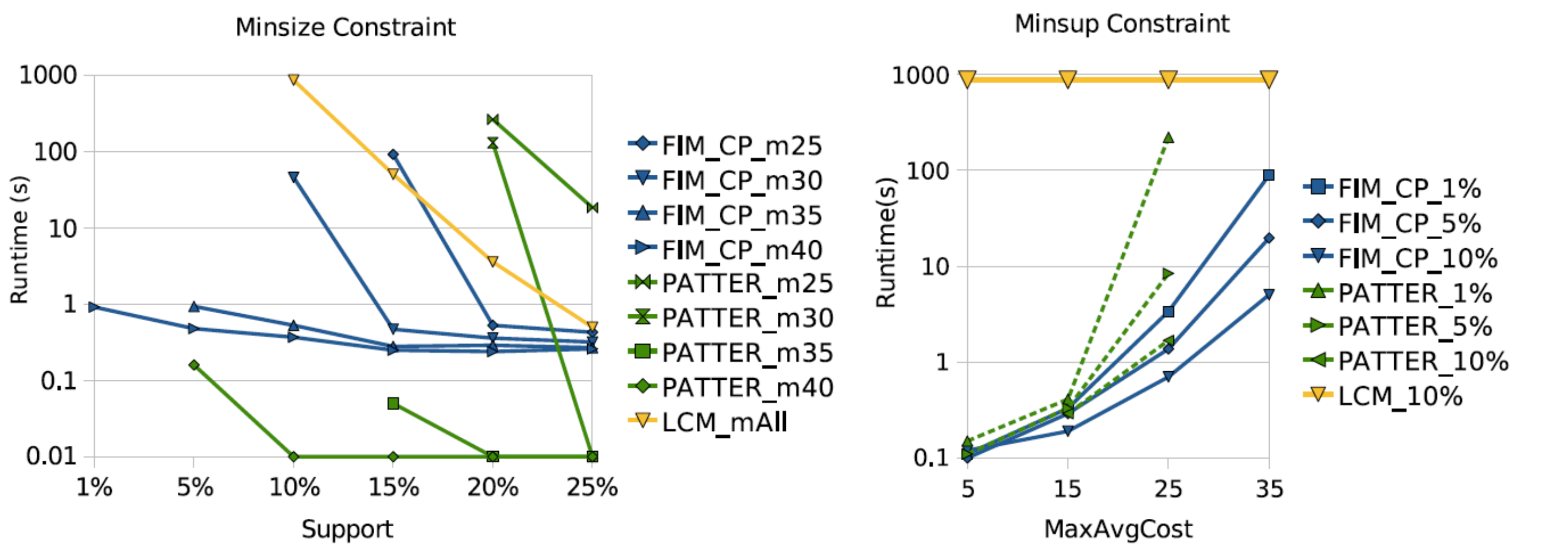


Figure 4: Runtimes of itemset miners on Segment data under constraints

For highly constrained problems, already competitive

CP for Itemset Mining

CP already competitive when having strong constraints

CP can easily handle new constraints and new combinations of constraints

General purpose.

Proof of principle as how to translate high-level model into solver language

Challenges

In Constraint Programming, different solvers optimized for different domains (reals, discrete domains, ...)

In Data Mining, different pattern types and data

- graphs, trees, sequences with CP ?

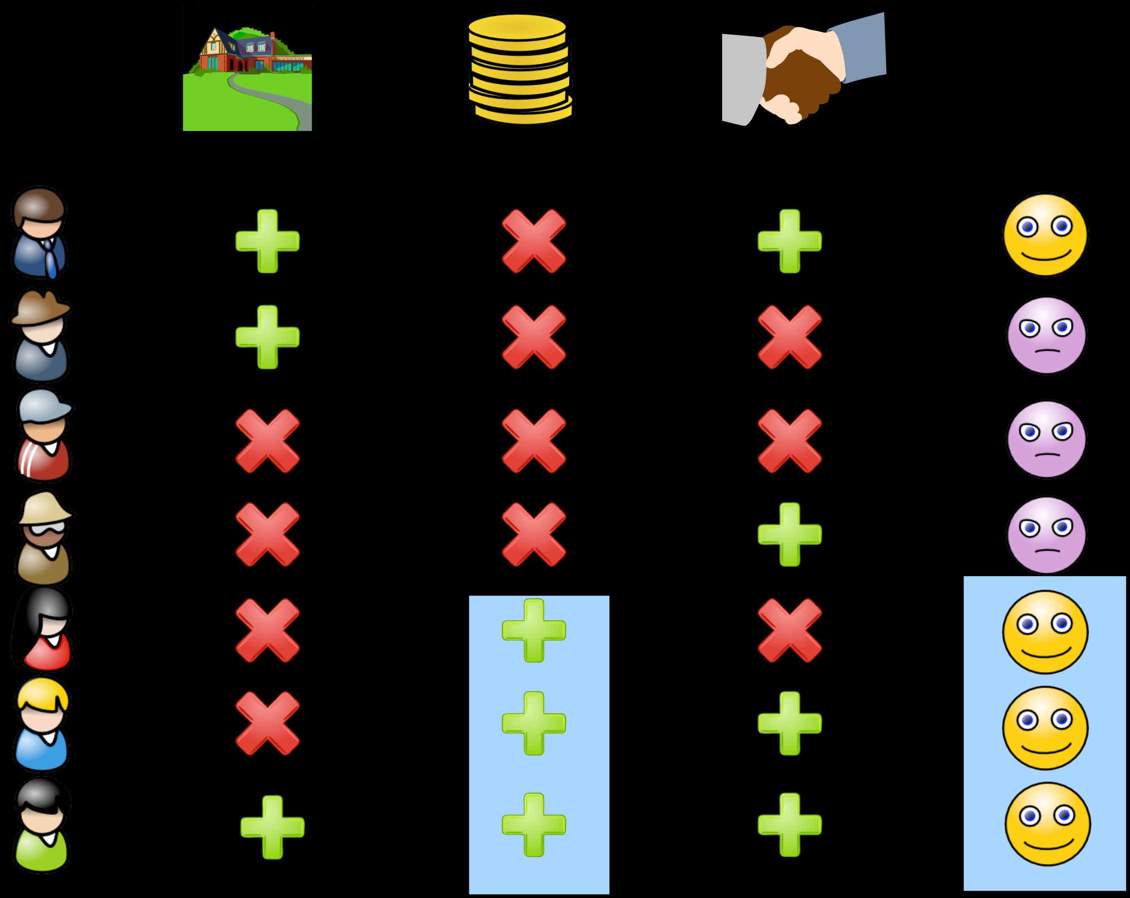
Large numbers of reified constraints unusual for CP

CP for Correlated Pattern Mining

Top-k Correlated Pattern Mining

- $\mathcal{D}$ now consists of two datasets, say P and N
- a correlation function $\phi(p, \mathcal{D})$, e.g., χ^2
- $Th(\mathcal{L}, Q, \mathcal{D}) = \arg_{p \in \mathcal{L}} \max_k \phi(p, \mathcal{D})$

Correlated Itemset Mining



cov	**3**	**0**	3
not	**1**	**3**	4
	4	3	

Correlated/Discriminative Itemset Mining

```
int: NrI; int: NrT; int: Freq;
array[1..NrT] of set of int: D;
set of int: pos; set of int: neg;

var set of 1..NrI: Itemset;
var set of 1..NrT: Trans;

constraint Trans = cover(Itemset, D);
constraint Itemset = cover_inv(Trans, D);

solve maximize
        card(Trans intersect pos) – card(Trans intersect neg);
```

accuracy

Alternative opt. functions, for example:

```
solve maximize chi2(Trans, pos, neg);
```

with:

```
function float: chi2(Trans, pos, neg) = ...
```

Function should not be decomposed;
automatically derive a bound?

Specifying the WHAT -- how to translate ?

Correlation function

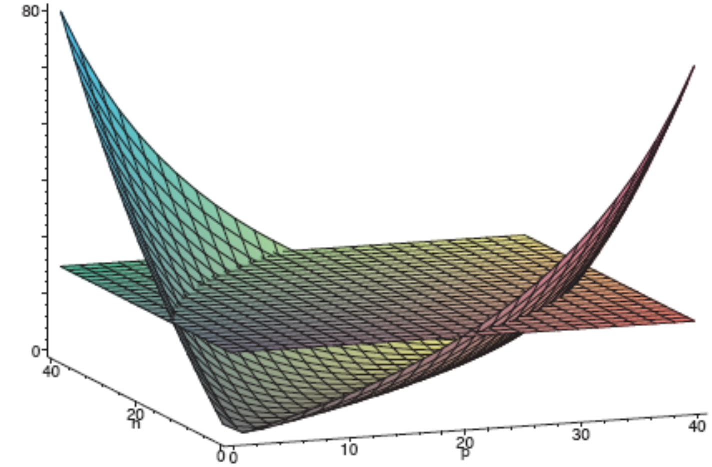
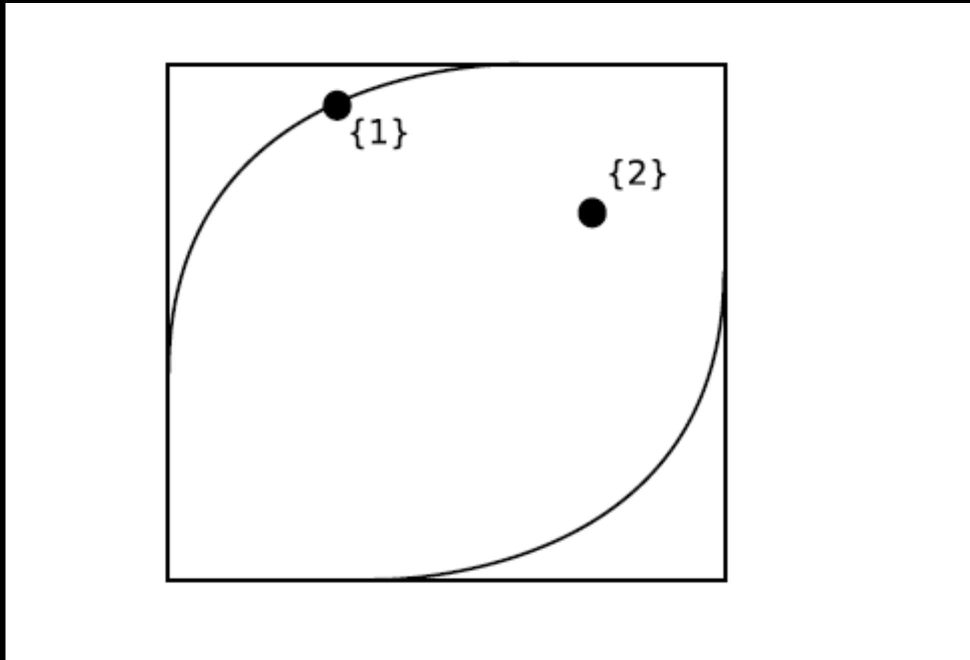


Figure 1: A plot of the χ^2 scoring function, and a threshold on χ^2.



Projection on PN-space
Nijssen KDID

1-support bound

Han et al. 08

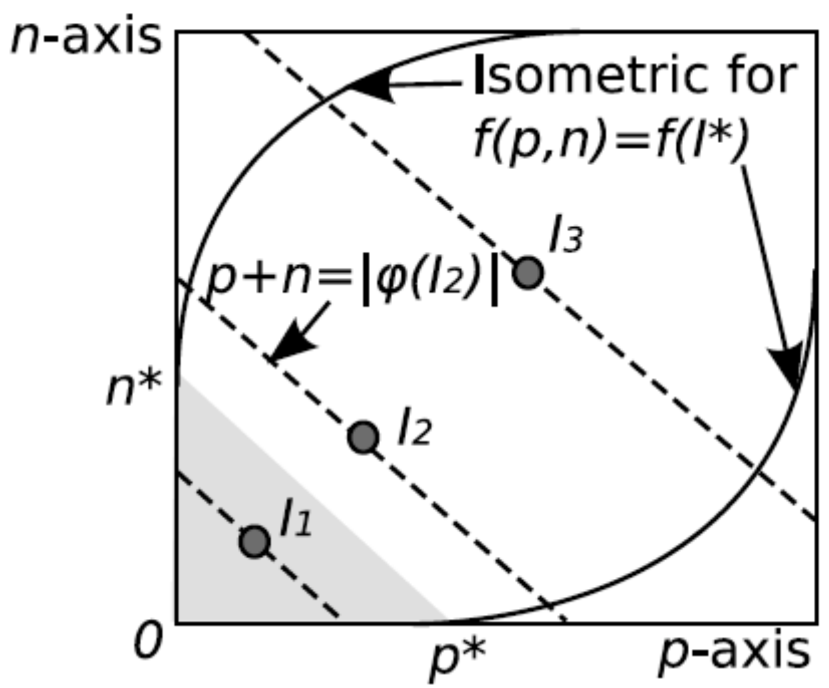


Figure 2: The 1-support bound in PN-space.

2-support bound

Morishita & Sese 98

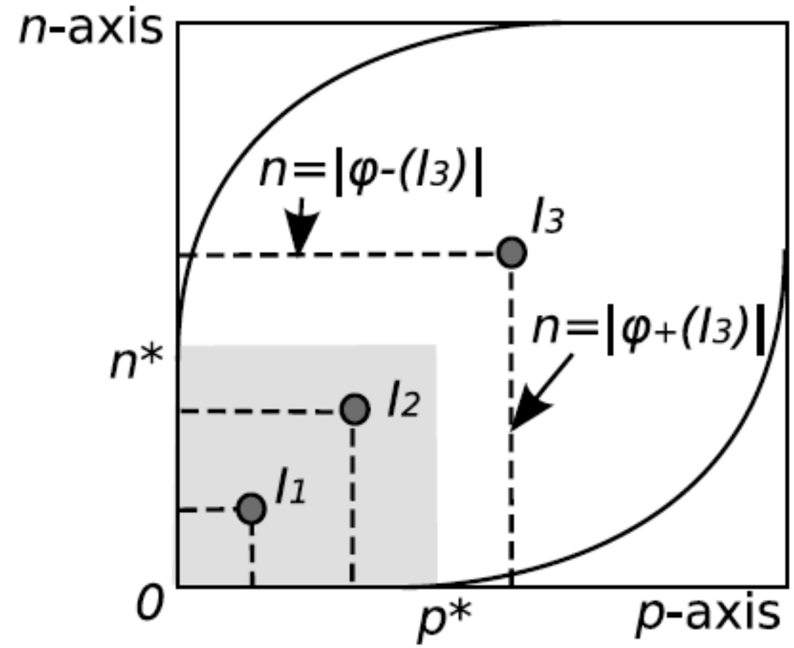


Figure 3: The 2-support bound in PN-space.

4-support bound

Nijssen et al. KDD 09 AIJ 11

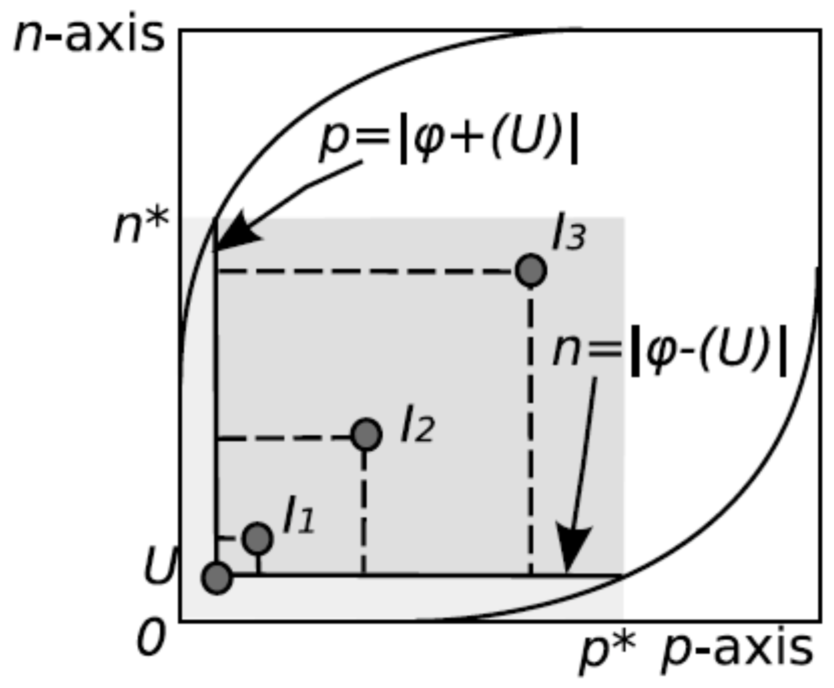
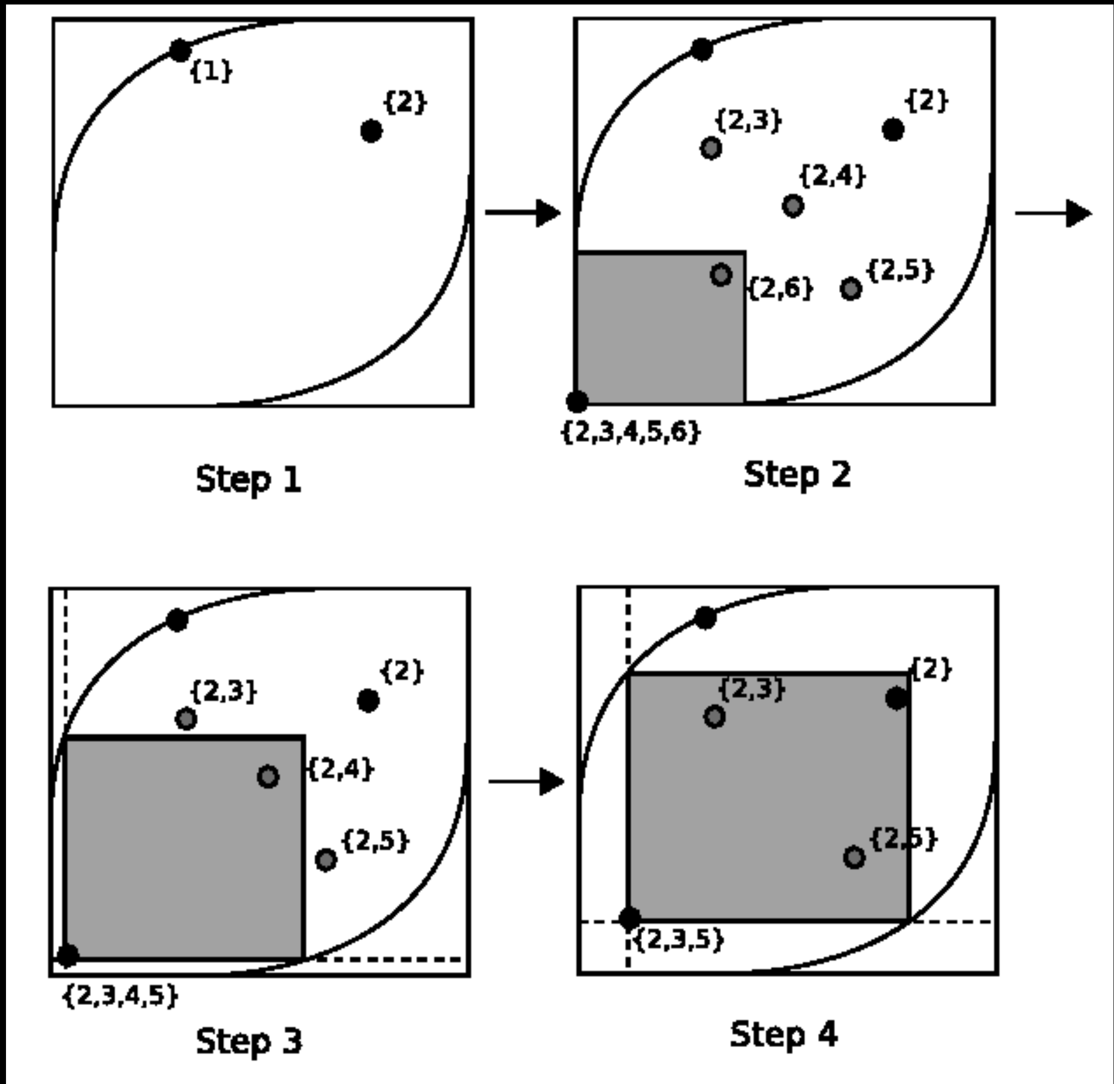


Figure 4: The 4-support bound in PN-space.

Illustration



Experiments

Name	Density	4-supp.	2-supp.	1-supp.
anneal	0.45	0.22	24.09	72.71
australian-credit	0.41	0.30	0.63	17.52
breast-wisconsin	0.5	0.28	13.66	228.08
diabetes	0.5	2.45	128.04	>
german-credit	0.34	2.39	66.79	>
heart-cleveland	0.47	0.19	2.15	29.58
hypothyroid	0.49	0.71	10.91	>
ionosphere	0.5	1.44	>	>
kr-vs-kp	0.49	0.92	46.20	713.35
letter	0.5	52.66	>	>
mushroom	0.18	14.11	13.48	27.31
pendigits	0.5	3.68	>	>
primary-tumor	0.48	0.03	0.13	0.85
segment	0.5	1.45	>	>
soybean	0.32	0.05	0.07	0.38
splice-1	0.21	30.41	31.11	35.02
vehicle	0.5	0.85	>	>
yeast	0.49	5.67	781.63	>

900s
timeout

Constraint Programming

It works (extremely well)

- written another propagator
- whenever a pattern satisfying the constraint is found update the threshold

Pattern Set Mining

Pattern Sets

Most data miners are not directly interested in all solutions or the top-k solutions to a pattern mining task, but typically post-process

Patterns are then used as features in classifiers or clusterers

So, why not apply constraint based mining to pattern sets directly ? [Zimmermann *PhD. 2009*] [Guns et al, IEEE TKDE]

Pattern Sets

Consider a set of itemsets

$$\{\{a,b,c\},\{b,d,e\},\{c,e,f\}\}$$

Can be interpreted as DNF expression

$$(a \wedge b \wedge c) \vee (b \wedge d \wedge e) \vee (c \wedge e \wedge f)$$

Useful for concept-learning and clustering

from local to global pattern mining

Pattern Sets

Can we apply Constraint-Based Mining to Pattern Set Mining ? $Th(\mathcal{L}, \mathcal{Q}, \mathcal{D}) = \{P \subseteq \mathcal{L} | \mathcal{Q}(P, \mathcal{D}) = true\}$

What are meaningful constraints ?

- local constraints on $I \in P$ such as $freq(I, \mathcal{D}) \geq minsup$
- constraints on all pairs of patterns $I_1, I_2 \in P$, e.g. $|covers(I_1, \mathcal{D}) \cap covers(I_2, \mathcal{D})| \leq t$
- global constraints $freq(P, \mathcal{D}) \geq t'$
- correlation, top-k, ...

Properties

Many properties of local pattern mining carry over, though sometimes in a subtle way, e.g.

$$(a \wedge b \wedge c) \vee (b \wedge d \wedge e)$$
$$\text{is more specific than}$$
$$(a \wedge b \wedge c) \vee (b \wedge d \wedge e) \vee (c \wedge e \wedge f)$$

Thus

$$freq((a \wedge b \wedge c) \vee (b \wedge d \wedge e)) \leq$$
$$freq((a \wedge b \wedge c) \vee (b \wedge d \wedge e) \vee (c \wedge e \wedge f))$$

Thus, anti-monotonicity reversed

One Step
Pattern Set Mining

Recent work : mine directly for

$$Th(\mathcal{L}, \mathcal{Q}, \mathcal{D}) = \{P \subseteq \mathcal{L} | \mathcal{Q}(P, \mathcal{D}) = true\}$$

where |P| =k => k-pattern set mining

using CP
clustering,
concept-learning
redescription mining
tiling

k-Pattern Sets

Key idea:

- fix the number of considered patterns in the set to k
- replace (T,I) by ($T, I_1, \ldots, I_k$) and specify constraints as before, ensure also that one does not obtain permutations of patterns ...
- add optimization criterion ... to find best k-pattern set

Pattern *Set* Mining

```
int: NrI; int: NrT;        int K;
array[1..NrT] of set of int: TDB;
set of int: pos; set of int: neg;

% pattern set
array[1..K] of var set of 1..NrI: Items;
constraint lexleq(Items);  % remove symmetries

% every pattern is closed 'on the positives'
constraint let { TDBp = [TDB[t] | t in pos] } in
      forall (d in 1..K) (
            Items[d] = cover_inv(cover(Items[d], TDBp), TDBp)

% accuracy of pattern set
solve maximize
      let { Trans = union(d in 1..K) (cover(Items[d], TDB)) } in
      card(Trans intersect pos) - card(Trans intersect neg);
```

Generality

Can model instantiations of:

- Concept learning (k-term DNF learning)
- Conceptual clustering
- k-Tiling
- Redescription mining
- ...

k-Pattern Set Mining

Key points:

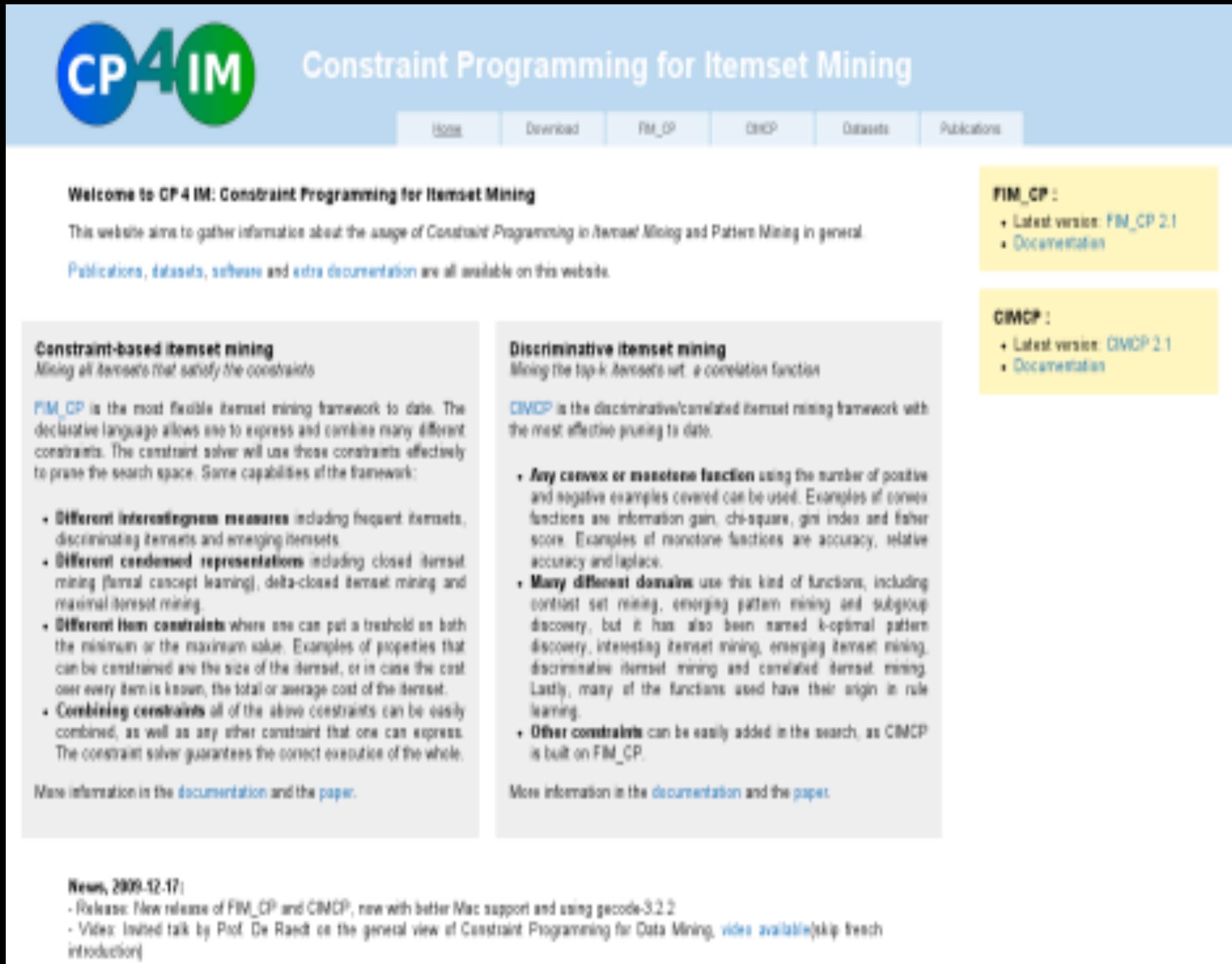
- A general modeling language for such tasks
- One-step exhaustive mining using CP
- Lessons about the interaction between local and global constraints

Conclusions Pattern Mining

Constraint programming --

- largely unexplored in data mining/machine learning though directly applicable
- using constraint programming principles results in a declarative modeling language for ML/DM
- using constraint programming solvers results in good performance
- several interesting open questions and new perspective

http://dtai.cs.kuleuven.be/CP4IM



Several open questions

What range of tasks can we model ?

Which modeling primitives do we need?

Do we need to adapt the solvers ? approximate solvers ?

Which translations to use ?

How to incorporate optimization ?

Zinc is only one framework ? What about others ?

Constraint satisfaction + Constrained Optimization

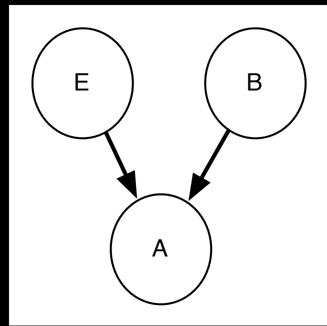
Other forms of ML/DM

Same principles should apply to

- probabilistic models and statistical relational learning
- other forms of machine learning
  - power of kernel and SVM methods comes from convex optimization (but at solver level)

Bayesian network learning

```
type state=record(boolean:A,E,B);
int NrEx;
array[1..NrEx] of state: Data;
var probdistribution for state: p;
constraint p(A,E,B) = p(E) * p(B) * p(A | E,B);

solve maximize likelihood(p,Data);
```




```
function var probability: likelihood(p,Data)= let {
    ...
} ;
```

Probabilistic Programming

Integrate probabilistic models into programming languages

Strongly tied to Statistical Relational Learning

Several such languages exist ... the alphabet soup

- Church, Prism, IBAL, Blog, ProbLog, kLog, CLP(BN), Figaro, ...
- integrated in programming languages such as Scheme, Prolog, Ocaml, Scala

Alarms

```
0.01:: earthquake.

0.02:: burglary.

alarm :- burglary.

alarm :- earthquake.

calls(X) :-

    neighbor(X), alarm, pcall(X).

0.7::pcall(X).

neighbor(john). neighbor(mary). neighbor(an).
```

Random variables

- earthquake.
- burglary.
- pcall(john).
- pcall(an).
- pcall(mary).

ProbLog IJCAI 07, TPLP 11a, TPLPb

Alarms

0.01:: earthquake.

0.02:: burglary.

alarm :- burglary.

alarm :- earthquake.

calls(X) :-

    neighbor(X), alarm, pcall(X)..

0.7::pcall(X).

neighbor(john). neighbor(mary). neighbor(an).

Random variables

- earthquake.
- burglary.
- pcall(john).
- pcall(an).
- pcall(mary).

Assume

- earthquake.
- pcall(john).

implies

calls(john).

http://www.cs.kuleuven.be/~dtai/problog/

ProbLog

Home | Download | Installation | Tutorial | Datasets | Publications | Contact

Welcome to the ProbLog Web Site

ProbLog is a probabilistic Prolog, a probabilistic logic programming language. It is available for download and it is well documented. It has been integrated in YAP-Prolog.
This website contains all the information to install and run ProbLog. Datasets are available for download in preprocessed form, to allow for comparison with different techniques. The publication section contains the scientific papers that were published on this matter.

Mailing List

There is a mailing list where updates are announced and where you can discuss with other users and developers of ProbLog.

- Subscribing, unsubscribing and browsing the archive can be done here
- If you are subscribed, you can send a message to all people on the list by writing to PROBLOG@LISTSERV.CC.KULEUVEN.AC.BE

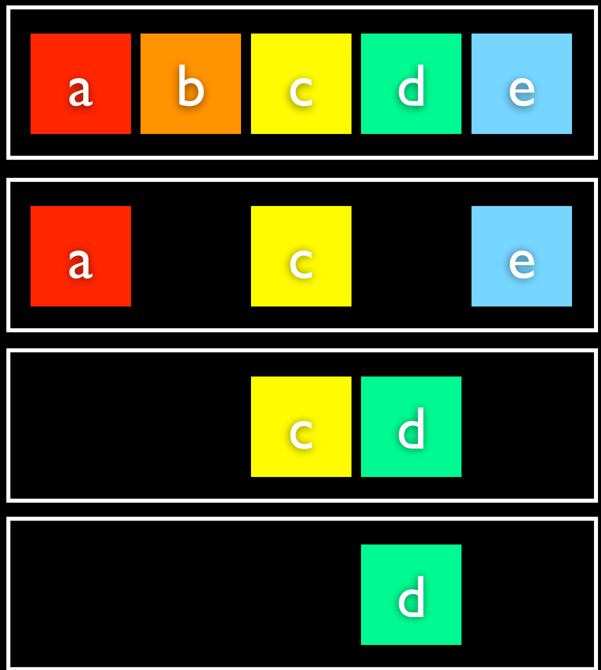
Links
DTAI Group
K.U. Leuven

Other SRL Systems

- Profile Toolbox
- PRISM
- AILog 2
- BLOG
- IBAL
- Alchemy

Valid XHTML
Valid CSS
Last modified: Tue Nov 10 12:17:34 CET 2009

Distribution over possible Worlds



$$p_a.p_b.p_c.p_d.p_e$$

$$p_a.(1-p_b).p_c.(1-p_d).p_e$$

$$(1-p_a).(1-p_b).p_c.p_d.(1-p_e)$$

$$(1-p_a).(1-p_b).(1-p_c).p_d.(1-p_e)$$

...

Semantics
Prob(Q) and Pr(Q|E)

a	b	c	d	e	true
a		c		e	false
		c	d		false
			d		false

...

positive :-
a,b,c
positive :-
b,c,d
positive :-
b,d,e.
?-P(positive).
?-P(positive|e).

$$P_s(q|T) = \sum_{L \subseteq L_T, BK \cup L \models q\theta} P(L|T)$$

Learning

As in Graphical Models

Learn parameters from partial datacases

- true : alarm, calls(john), earthquake
- false : burglary
- unknown: pcall(), calls(mary), calls(an).

Probabilistic Programming

Various inference strategies exist to answer queries

- exact, approximate, ...
- some can be tied in to graphical model "solvers" (packages by e.g. Darwiche)

Various learning strategies

- similar situation
- few solvers that deal with learning ...

The programming part

In an integrated programming language, learning is just constraint satisfaction and optimization

- in ProbLog and kLog -- just a query
- in CP -- just a call to a solver

Results / output can be used afterwards ...

Inputs / can also be "programmed"

Compositionality principle -- outputs of learning / mining can be used further on, also as inputs for further learning tasks.

Conclusions

Declarative modeling languages for ML / DM can provide an answer to Mitchell's question.

We can realize this by applying the principles of constraint programming and knowledge representation

Essentially three components

- Modeling -- specify the problem -- the what
- Solver -- translation of the problem -- the how
- Programming -- in which everything is embedded
- with Translations -- an essential step !

Conclusions

All the necessary ingredients are available to realize declarative modeling languages for ML/DM

- machine learning & data mining
- declarative modeling, constraint programming and knowledge representation
- programming language technology

So we are going to do it

What about FCA ?

Questions ?