Declarative Modeling for Machine Learning and Data Mining

Lab for Declarative Languages and Artificial Intelligence

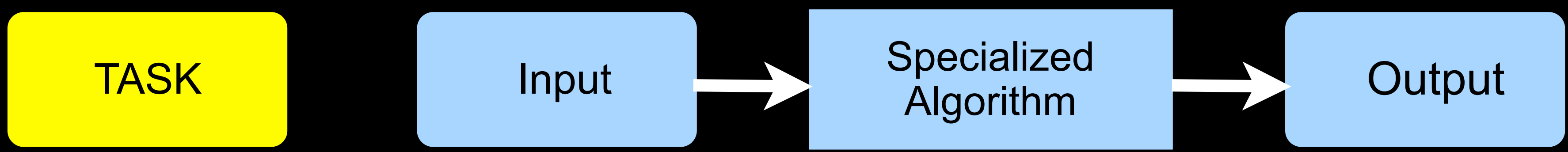
Joint work with especially
Tias Guns and Siegfried Nijssen and Paolo Frasconi
and the EU FET ICON project

KU LEUVEN

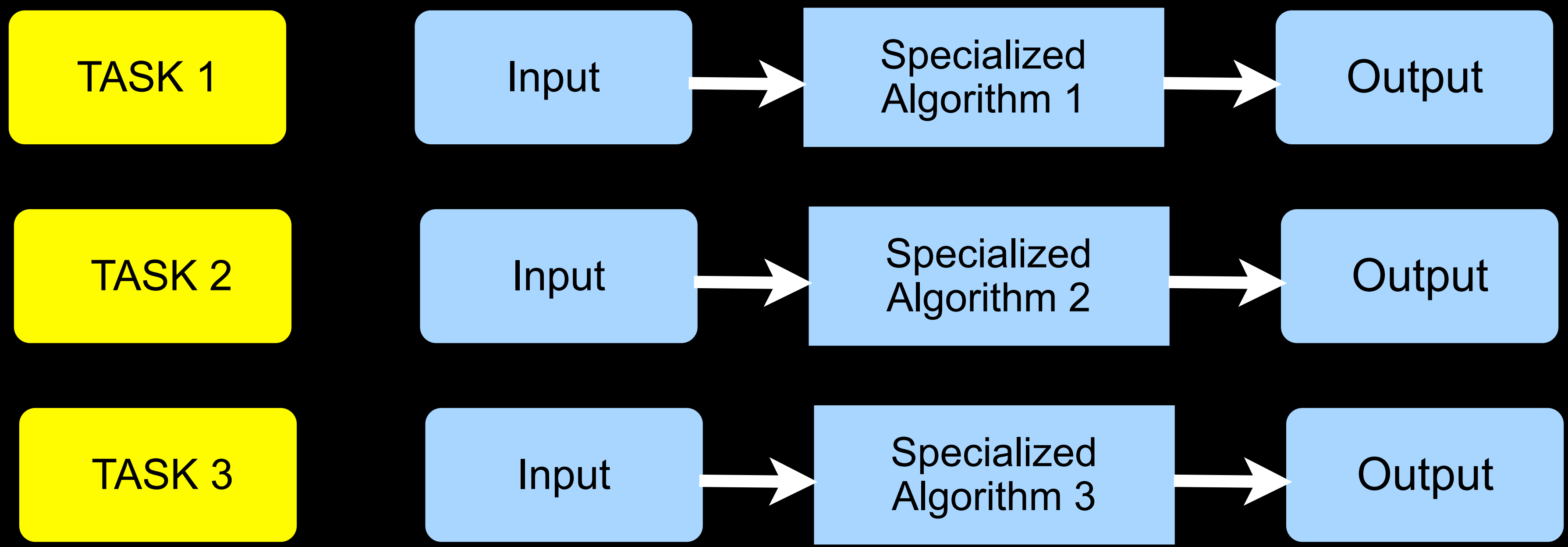
Our work today ...

We typically ...

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



And do it again ...

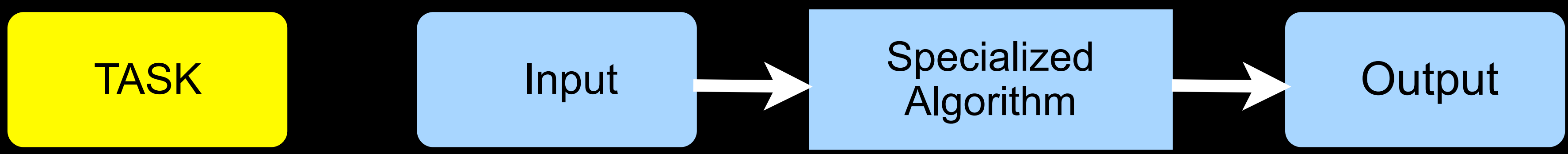


Our work today ...

We typically ...

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

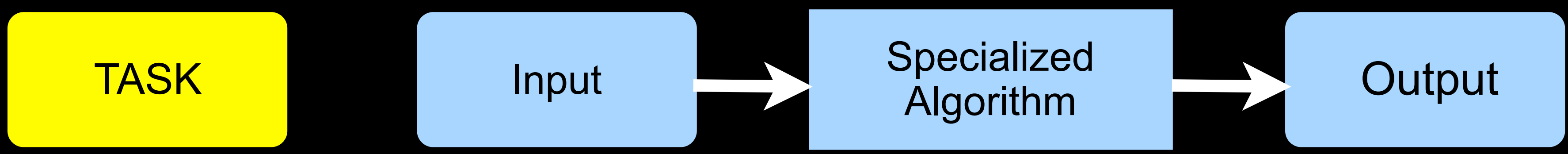
hard



The Challenge

Cannot we simplify this ... ?

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



Key Point

The key point I want to make is that POTENTIALLY we can by adopting a Declarative Modeling paradigm

first steps have been taken ...

e.g. use of Convex Optimisation

Overview Talk

- The Challenge:
  - from Programming to Modeling for ML/DM
- The what, why and how of Declarative Modeling (and Constraint Programming)
- How does this relate to ML/DM ?
- Evidence: a case study in pattern mining
- Perspective / Discussion

The What, Why and How
of
Declarative Modeling

What is declarative modeling ?

Model

```
array [1..9, 1..9] of var 1..9: sq;
predicate row_diff(int: r) =
    all_different (c in 1..9) (sq[r, c]);
predicate col_diff(int: c) =
    all_different (r in 1..9) (sq[r, c]);
predicate subgrid_diff(int: r, int: c) =
    all_different (i, j in 0..2) (sq[r + i, c + j]);

constraint forall (r in 1..9) (row_diff(r));
constraint forall (c in 1..9) (col_diff(c));
constraint forall (r, c in {1, 4, 7}) (subgrid_diff(r, c))

solve satisfy;
```

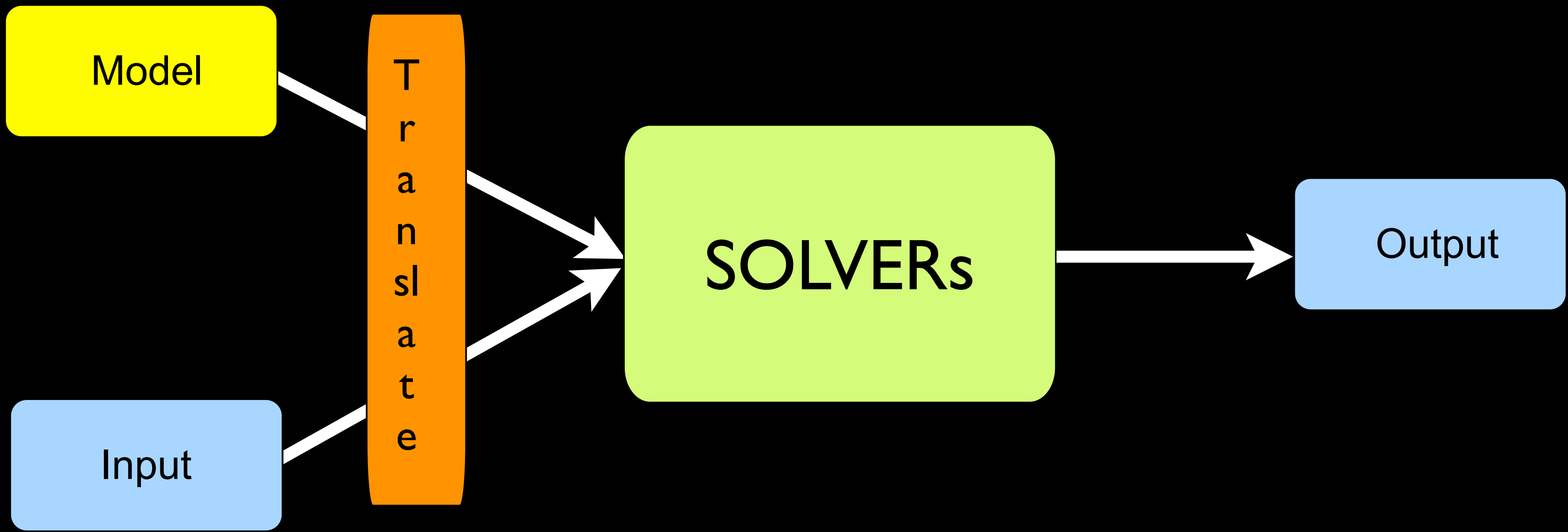
Inputs

1								6
		6		2		7		
7	8	9	4	5		1		3
			8		7			4
				3				
	9				4	2		1
3	1	2	9	7			4	
	4			1	2		7	8
9		8						

Zinc family of languages

How does it work ?

MODEL specifies task = constraints
+ optimization criterion



state WHAT the problem is

Data = Input

different SOLVERS possible

Why declarative modeling ?

DECLARATIVE

- few lines of code
- easy to understand, maintain, change
- 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
- solver is built in the program

Here - CONSTRAINT PROGRAMMING
Also -- ANSWER SET PROGRAMMING

Constraint Programming

CSP

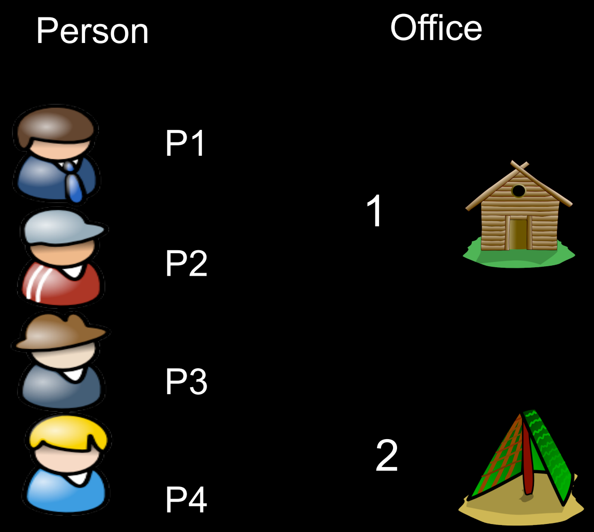
Given

- 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

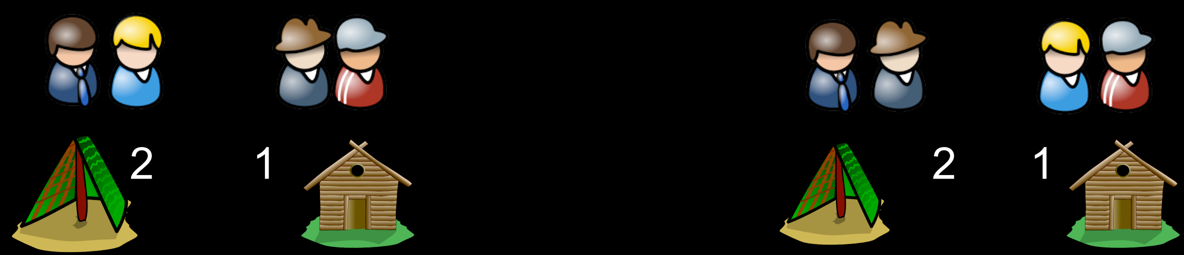
Zinc [Garcia de la Banda *et al.*CP 06]

Constraint Satisfaction




```
var P1, P2, P3, P4: {1,2};

constraint P1 != P2;
constraint P3 != P4;
constraint P1 !=1;

solve satisfy;
```

Solutions



Solvers for CP

Two key ideas

- propagation of constraints, e.g., from

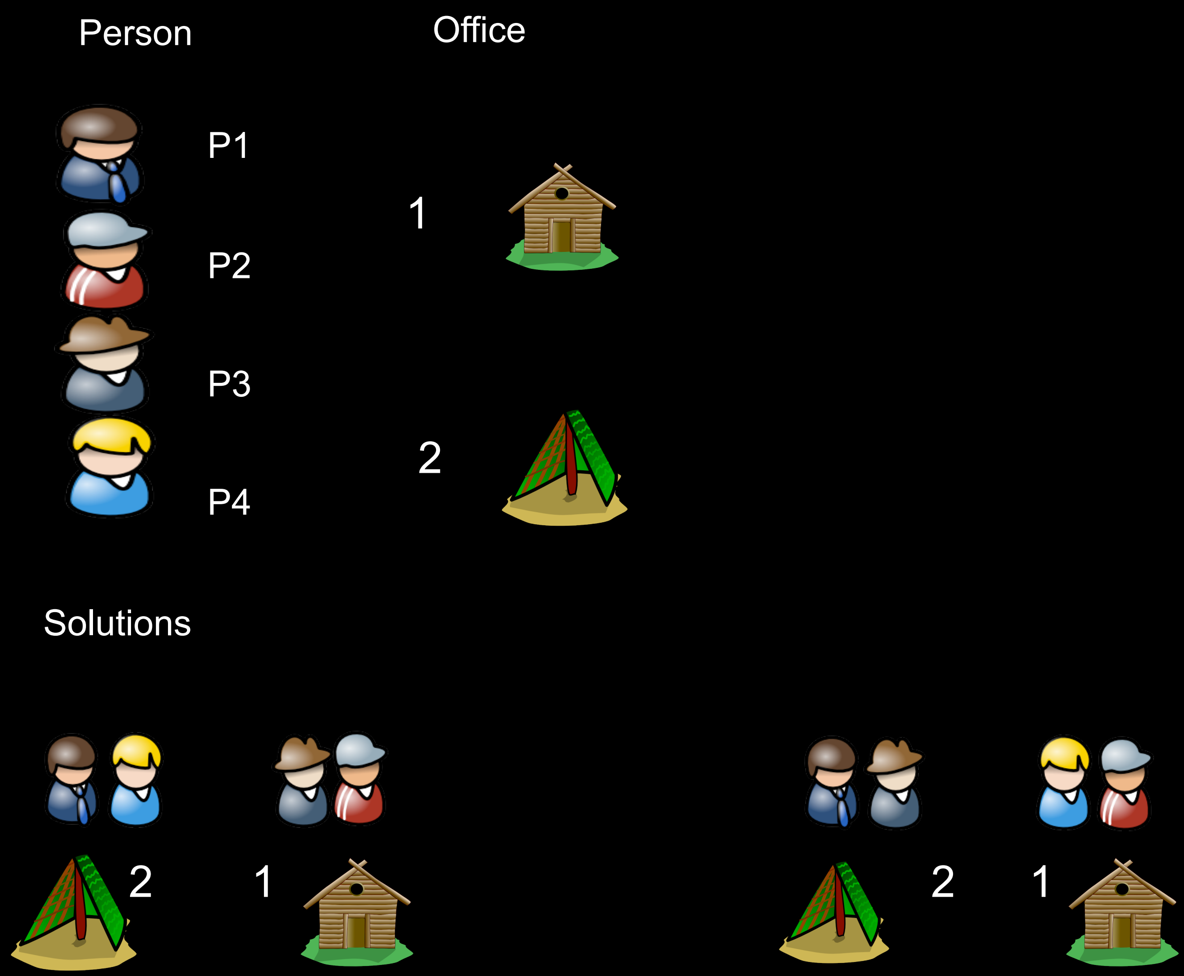
  D(P1) = {1} and D(P2) = {1,2,3} and P1 != P2 infer that 1 ∉ D(P2) and simplify D(P2) = {2,3}

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

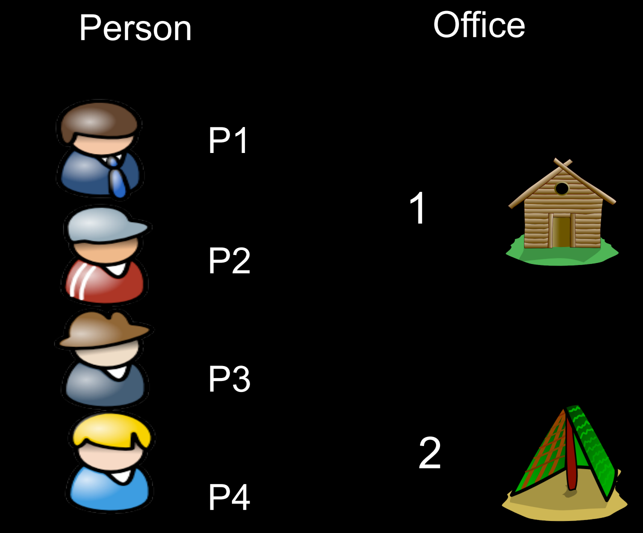
- if you cannot propagate, instantiate (or divide) and recurse, e.g.,

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

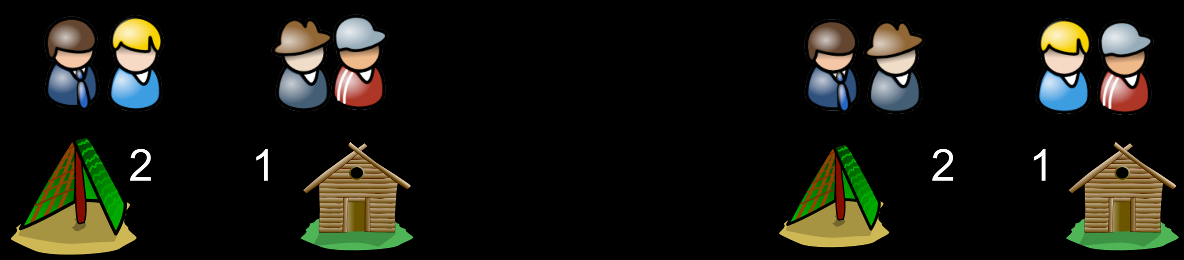
  P2=2 P2=3


Person
Office
P1
P2
P3
P4
1
2
Solutions
2
1
2
1

Search



Solutions



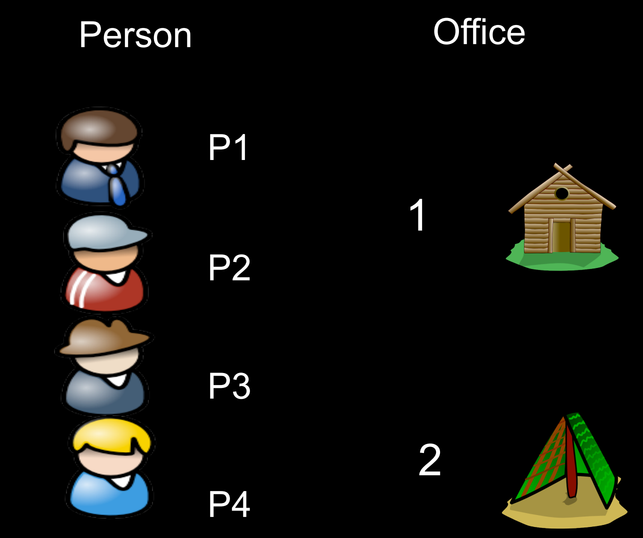
D(P1) = {1,2}
D(P2) = {1,2}
D(P3) = {1,2}
D(P4) = {1,2}

P1 != P2

P3 != P4

P1 != 1

Search



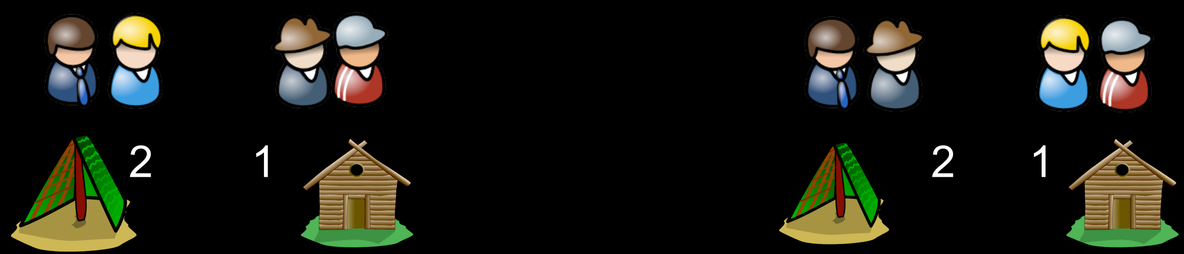
D(P1) = {1,2}
D(P2) = {1,2}
D(P3) = {1,2}
D(P4) = {1,2}

P1 != P2

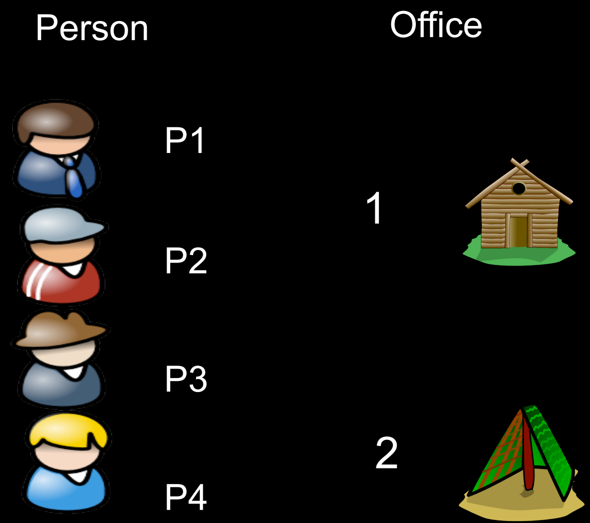
P3 != P4

P1 != 1

Solutions



Search



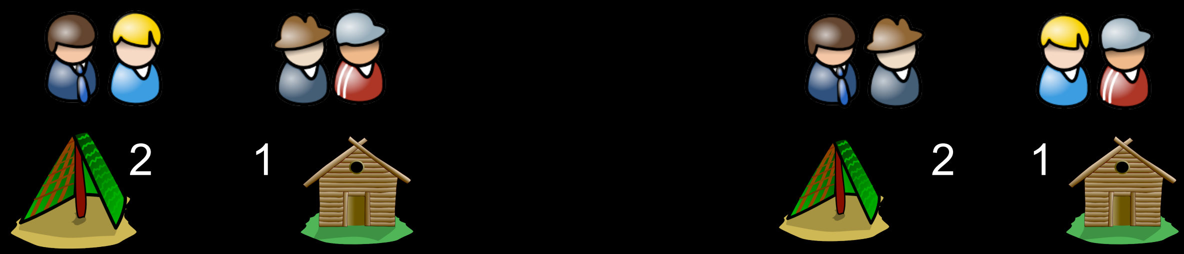
D(P1) = {   2}
D(P2) = {1,2}
D(P3) = {1,2}
D(P4) = {1,2}

P1 != P2

P3 != P4

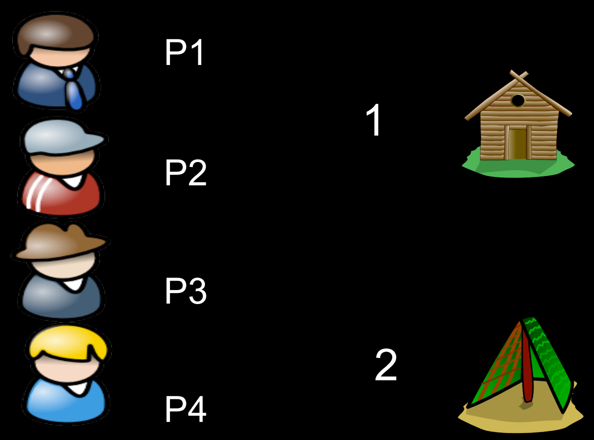
P1 != 1

Solutions

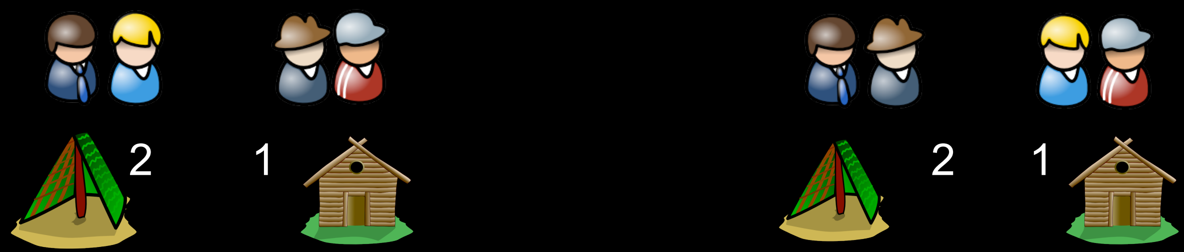


Search

Person　　Office



Solutions



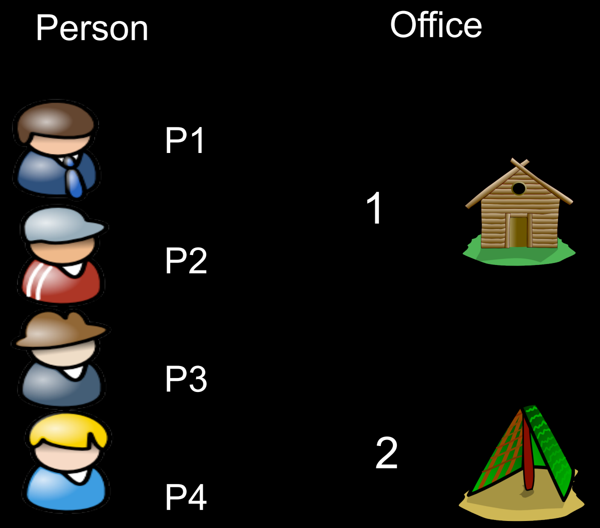
D(P1) = {   2}
D(P2) = {1,2}
D(P3) = {1,2}
D(P4) = {1,2}

P1 != P2

P3 != P4

P1 != 1

Search



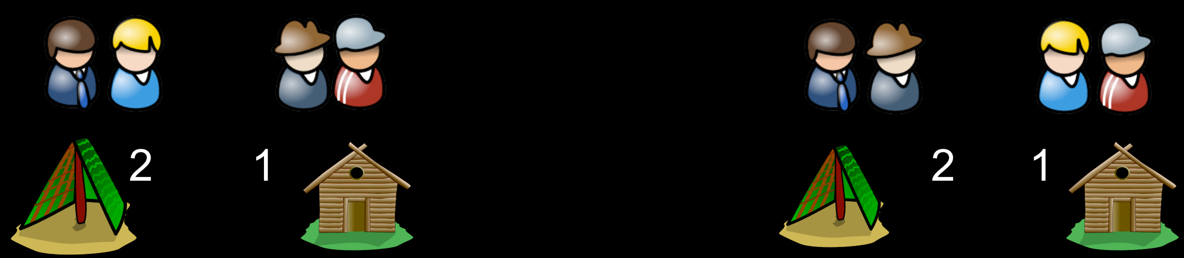
D(P1) = {   2}
D(P2) = {1   }
D(P3) = {1,2}
D(P4) = {1,2}

P1 != P2

P3 != P4

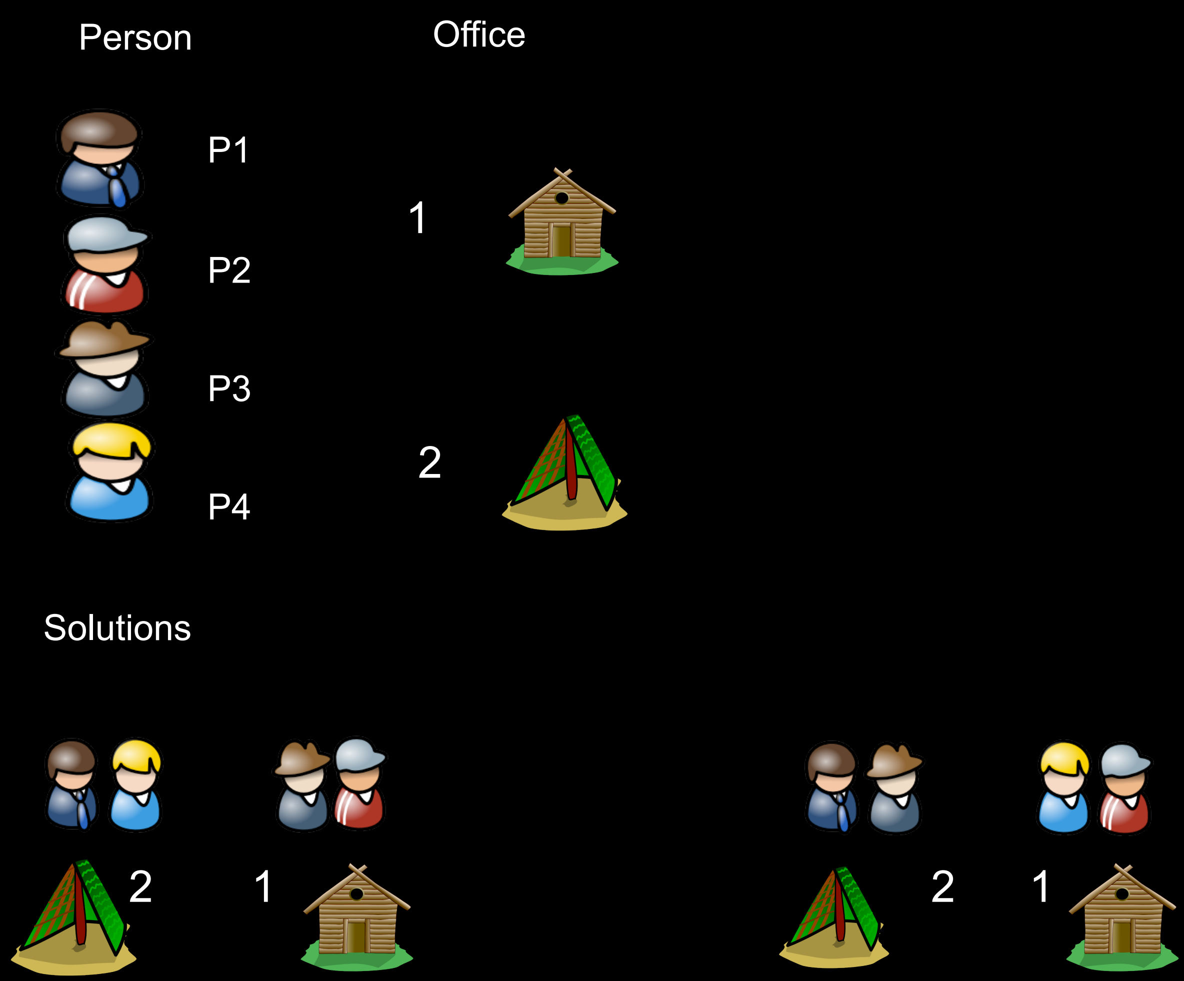
P1 != 1

Solutions



choose P3 = 1

Search



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

P1 != P2

P3 != P4

P1 != 1

Search

Person

P1

P2

P3

P4

Office

1

2

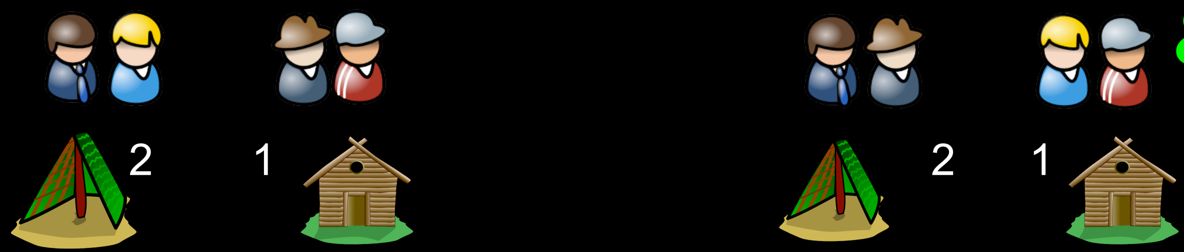
D(P1) = {   2}
D(P2) = {1   }
D(P3) = {1   }
D(P4) = {   2}

P1 != P2

P3 != P4

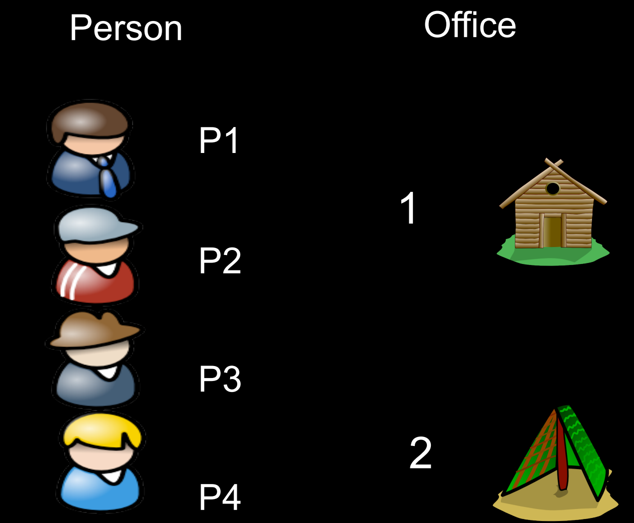
P1 != 1

Solutions

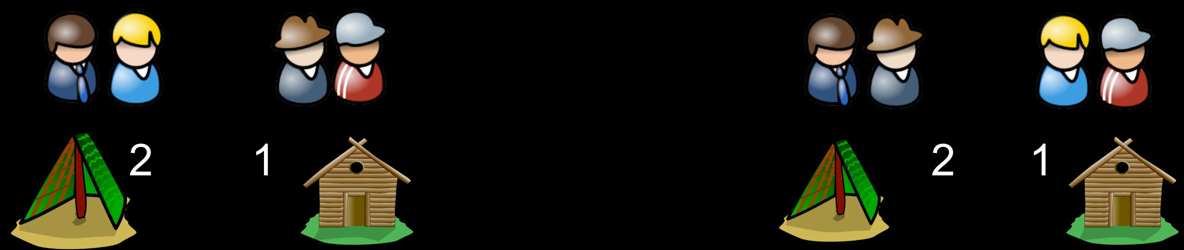


& backtrack

Search



Solutions



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

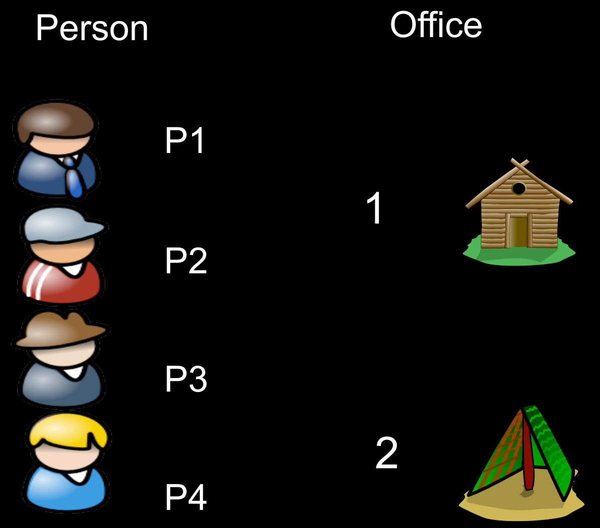
P1 != P2

P3 != P4

P1 != 1

choose P3 = 2

Search



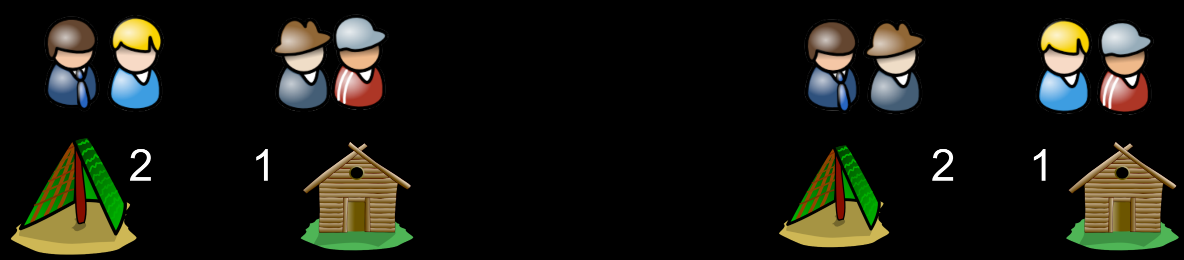
D(P1) = {   2}
D(P2) = {1   }
D(P3) = {   2}
D(P4) = {1,2}

P1 != P2

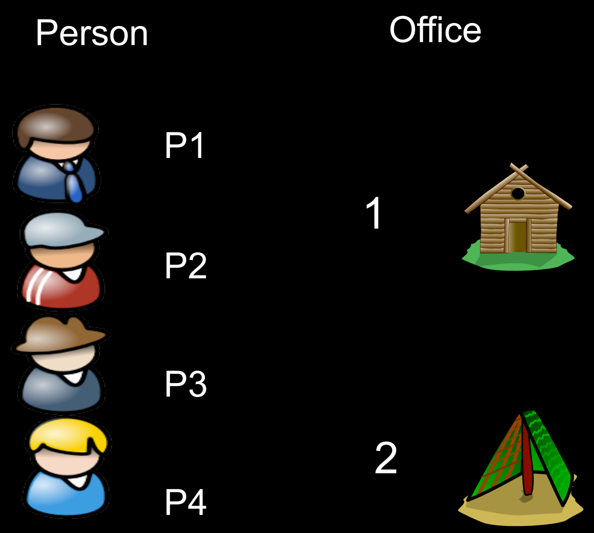
P3 != P4

P1 != 1

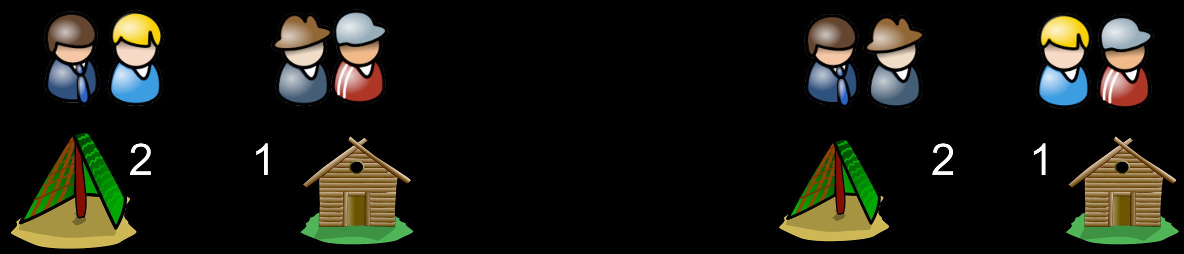
Solutions



Search



Solutions



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

What about ML/DM ?

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

[Mannila and Toivonen, 96]

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 → Y
- a hypothesis space L containing functions X → Y
- a dataset of examples E = { (x, f(x)) | x ∈ X }
- a loss function *loss*(h,E) → ℝ

Find h ∈ L that minimizes *loss*(h,E)

supervised

Observation 1

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

well-known in ML and DM
good news

Observation 2

Use of solvers is very common in

statistical learning (and SVMs)

- convex optimization and mathematical programming solvers

graphical models

- knowledge compilation packages and belief propagation

An important factor for their success

Observation 2

There has been a paradigm shift in the field of AI from programming to solving (Hector Geffner at ECAI 2012)

Today AI uses solvers for crisp computational problems

- SAT, ASP, CSP, CP, maxSAT, weighted model counting, ...
- many problems are reduced to these basic problems ... and solved efficiently

Still less common in other areas of DM/ML

Observation 3

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

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

- specify the WHAT -- not the HOW

Examples include

- ZINC, Essence, Comet, OPL, FO(.), ...

Very flexible approach ...

Still less common in DM/ML (except Matlab ?)

Long standing open questions

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

Can we design programming languages containing machine learning primitives?

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

... some subroutines are hand-coded while others are specified as "to be learned." ... the programmer declares the inputs and outputs of each "to be learned" subroutine, then selects a learning algorithm ...

Questions remain open

Though 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] and kLog [Frasconi et al.]

Can we obtain programming languages for ML / DM

by applying the principles of constraint programming ?

Evidence
The case of Pattern mining

Pattern Mining

A. frequent pattern

- which patterns are frequent ?

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

B. Correlated pattern mining = subgroup discovery

- which patterns are significant w.r.t. classes ? all patterns ? k-best patterns ?

$$Th(\mathcal{L}, Q, \mathcal{D}) = \arg_{p \in \mathcal{L}} \max_k \phi(p, \mathcal{D})$$

C. pattern set mining

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

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

Pattern Mining

A. frequent pattern

KDD 08

- which patterns are frequent ?

$Th(\mathcal{L}, Q, \mathcal{D}) = \{$

B. Correlated pattern

- which p… k-best
  pat…

KDD 09

$_k\ \phi(p, \mathcal{D})$

C. patte…

- whi… concept-description for the actives ?
  for th…

IEEE TKDE 11

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

We have been using off-the-shelf CP SOLVERS for these tasks, cf. Guns, Nijssen, De Raedt [AAAI 10, AIJ 11]

One solver for all of these

Easy to combine different constraints

Now looking at modeling level

A. Frequent Pattern Mining

A. Frequent Itemset Mining

Given

- $\mathcal{I} = \{1, \cdots, NrI\}$
  set of items
- $\mathcal{T} = \{1, \cdots, NrT\}$
  set of transactions identifiers
- $\mathcal{D} = \{(t, I) | t \in \mathcal{T}, I \subseteq \mathcal{I}\}$
  Dataset
- $Items \subseteq \mathcal{I}$ and $Trans \subseteq \mathcal{T}$

Find $Items$ such that
$|covers(Items, \mathcal{D})| > freq$

where $covers(Items, \mathcal{D}) =$
$\{t \in \mathcal{T} | (t, I) \in \mathcal{D}$ and $Items \subseteq I\}$

A.Frequent Itemset Mining

Given

- $\mathcal{I} = \{1, \cdots, NrI\}$
  set of items
- $\mathcal{T} = \{1, \cdots, NrT\}$
  set of transactions identifiers
- $\mathcal{D} = \{(t, I) | t \in \mathcal{T}, I \subseteq \mathcal{I}\}$
  Dataset
- $Items \subseteq \mathcal{I}$ and $Trans \subseteq \mathcal{T}$

Find $Items$ such that
$|covers(Items, \mathcal{D})| > freq$

where $covers(Items, \mathcal{D}) =$
$\{t \in \mathcal{T} | (t, I) \in \mathcal{D}$ and $Items \subseteq I\}$

```
int: Freq;
int: NrI;
int: NrT;

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

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

constraint card(Trans) > Freq;
constraint Trans = covers(Items, D);

solve satisfy;
```

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

Frequent Itemset Mining

math like notation

user defined functions and constraints

solver independent (standardized)

efficiently solvable

```
int: Freq;
int: NrI;
int: NrT;

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

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

constraint card(Trans) > Freq;
constraint Trans = covers(Items, D);

solve satisfy;
```

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

Closed Itemset Mining

```
int: Freq;
int: NrI;
int: NrT;

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

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

constraint card(Trans) > Freq;
constraint Trans = covers(Items, D);
constraint Items = cover_inv(Trans, D);
solve satisfy;
```

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

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

Further Constraints

* exact coverage :
  t in Trans <-> Items subset D[t]

* freq:
  i in Items -> card(Trans intersect D'[i]) >= Freq

* maximal:
  i in Items <-> card(Trans intersect D'[i]) >= Freq

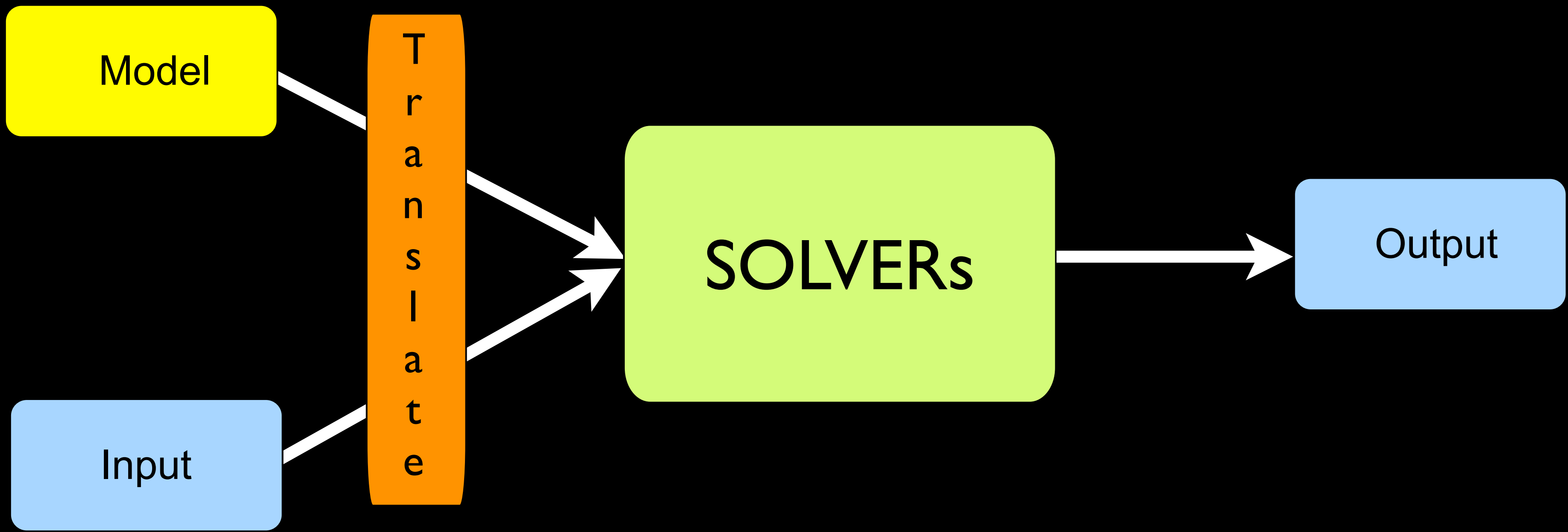
* closed:
  i in Items <-> Trans subset D'[i]

* delta-closed:
  i in Items <-> card(Trans intersect D'[i]) <= Delta*card(Trans)

easy to model

How does it work ?

MODEL specifies task = constraints
+ optimization criterion



Only state WHAT the problem is

Data = Input

Solver 1

- CP based
- Map to standard Solvers offered by Zinc
- Like Gecode and Comet
  - Gecode -- sound and complete
  - Comet -- local search ...
- CHALLENGE
  - how to encode this efficiently?

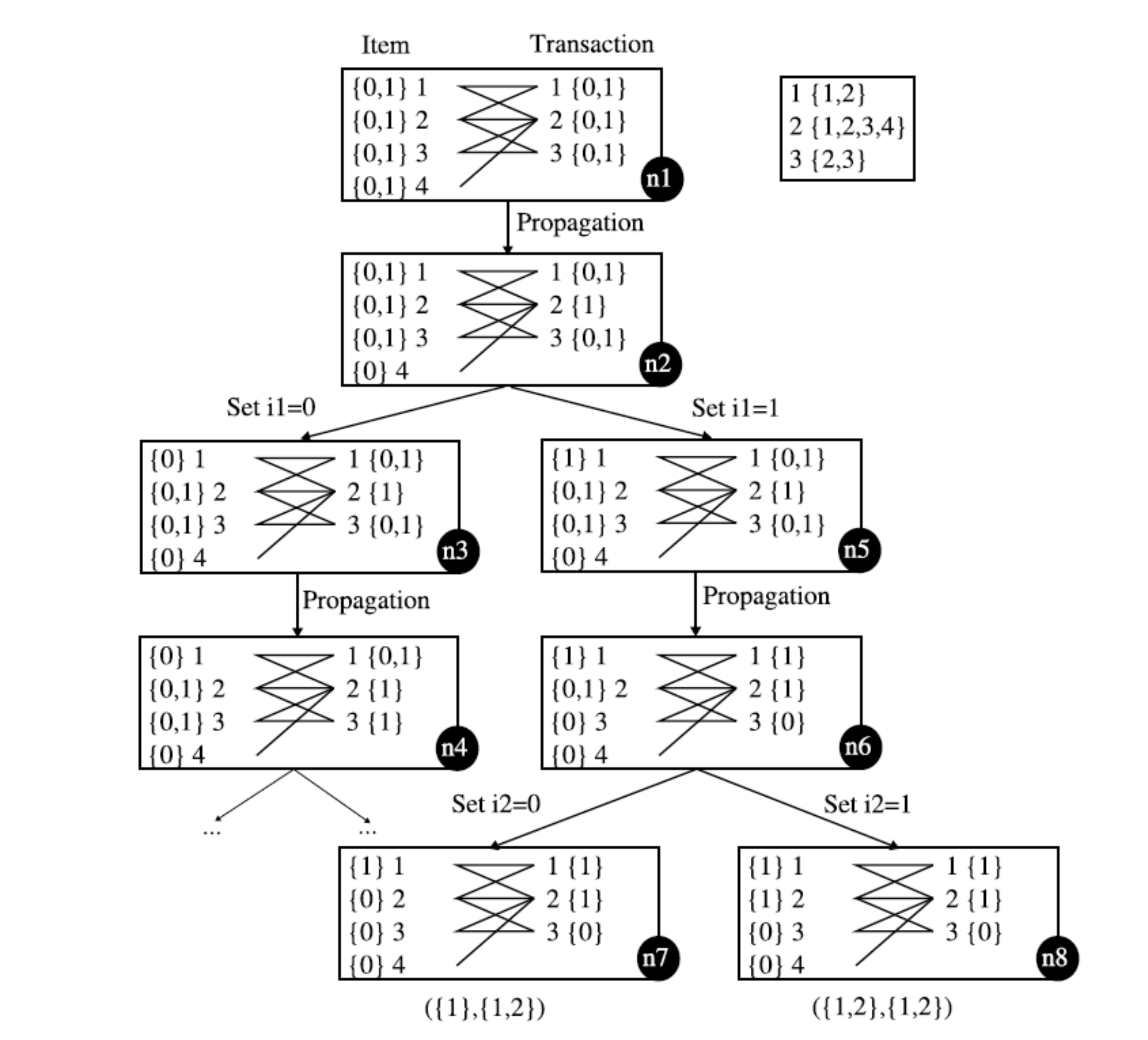
Encoding in Zinc

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

array [1..NrI] of var bool: Items;
array [1..NrT] of var bool: Trans;

constraint % encode D: every Trans complement has no supported Items
  forall(t in 1..NrT) (
     Trans[t] <-> sum(i in 1..NrI) ( Items[i]*(1 - (i in D[t])) ) <= 0
  );

constraint % frequency: every Item is supported by sufficently many Trans
  forall(i in 1..NrI) (
     Items[i] -> sum(t in 1..NrT) ( Trans[t]*(i in D[t]) ) >= Freq
  );

solve satisfy;
```

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

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

Resulting Search Strategy akin to Zaki's Eclat [KDD 97]



see
Guns et al AIJ 11

Solver 2

- Use a Data Mining System as solver
- Results with LCM [Uno et al.] within Zinc
- CHALLENGE
  - how to recognize that DM system applies ?
  - possibly add post-processing ...

B. Correlated Pattern Mining
= Subgroup Discovery
= Discriminative patterns

Top-k Correlated Pattern Mining
Subgroup Discovery

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

Modeling perspective

```
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: Items;
var set of 1..NrT: Trans;

constraint Trans = cover(Items, D);
constraint Items = 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)
```

Correlation function

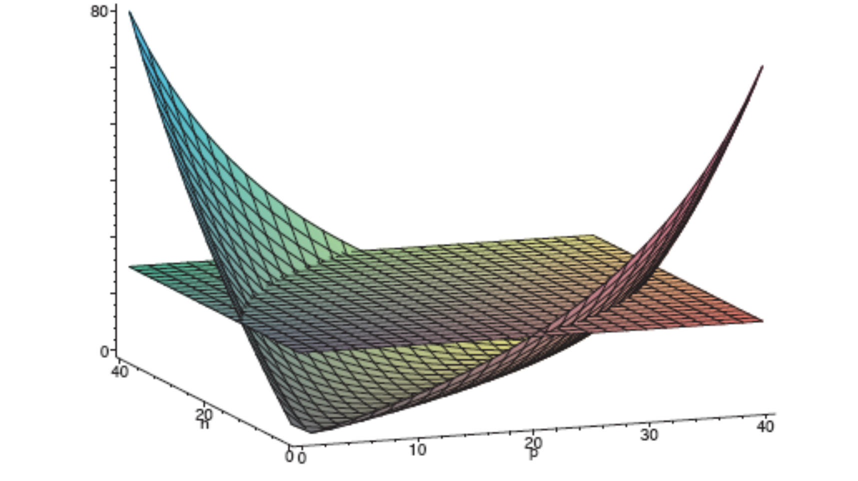
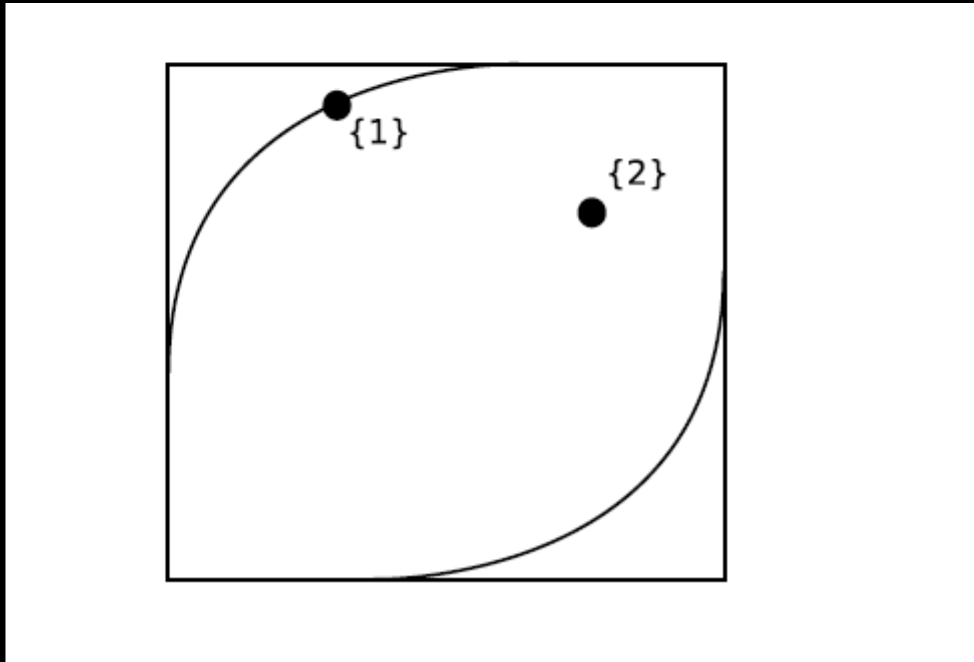


Figure 1: A plot of the χ^2 scoring function, and a threshold on χ^2.



Projection on PN-space
Nijssen KDID

Monotonicity

$$freq(S) \geq freq(S \cup T) \geq freq(S \cup Dom(S))$$

Traditional pruning/propagation employs **upper bound**:

remove d from $Dom(S)$ when $freq(S) \geq t$ and $freq(S \cup \{d\}) < t$

Other propagation – unavoidable item sets also possible – **lower bound**

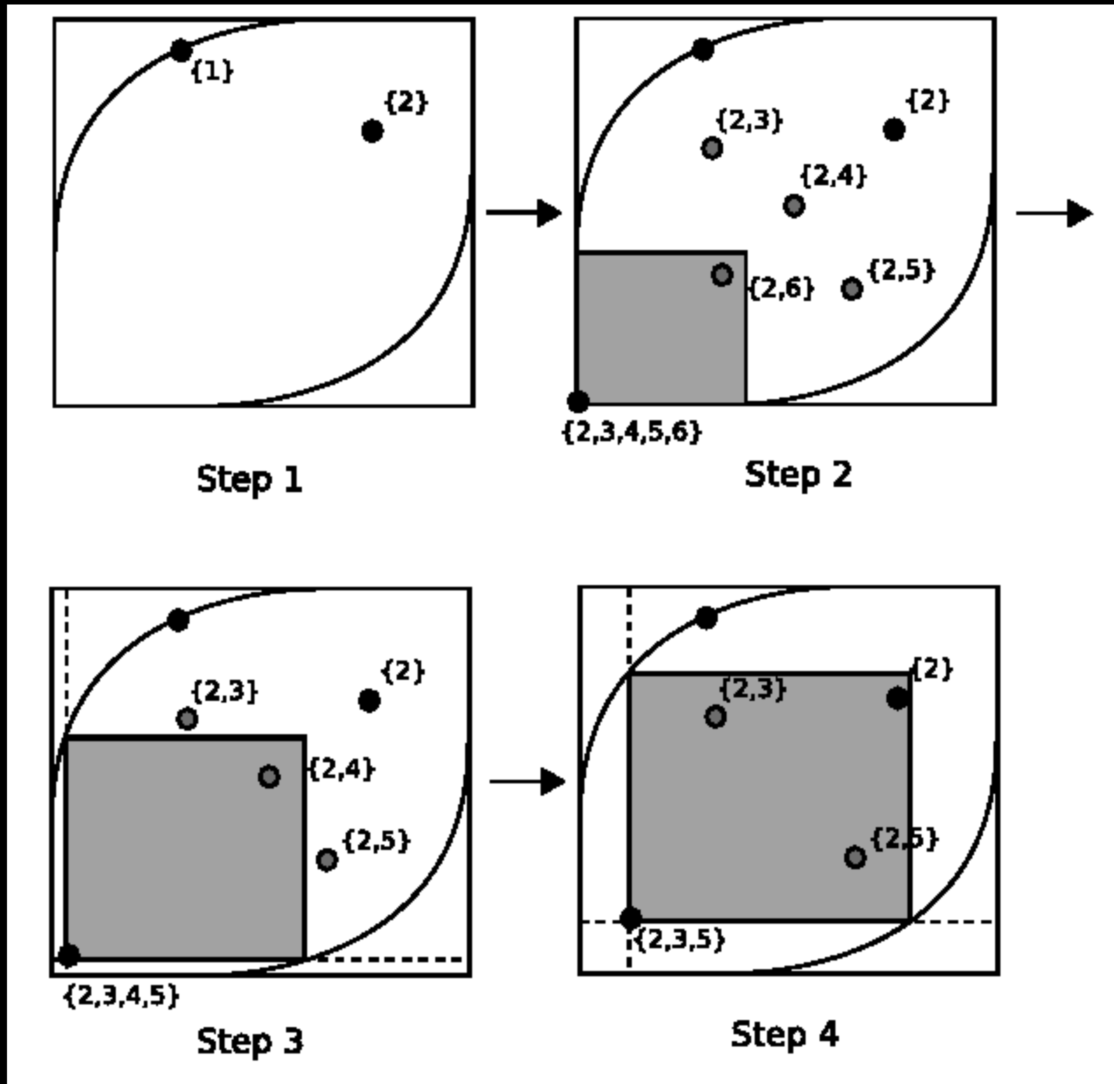
$$freq(S) \geq freq(S \cup T) \geq freq(S \cup Dom(S)) = a > 0$$

then a is a lower bound on $freq(S)$, that is $freq(S) \geq a$

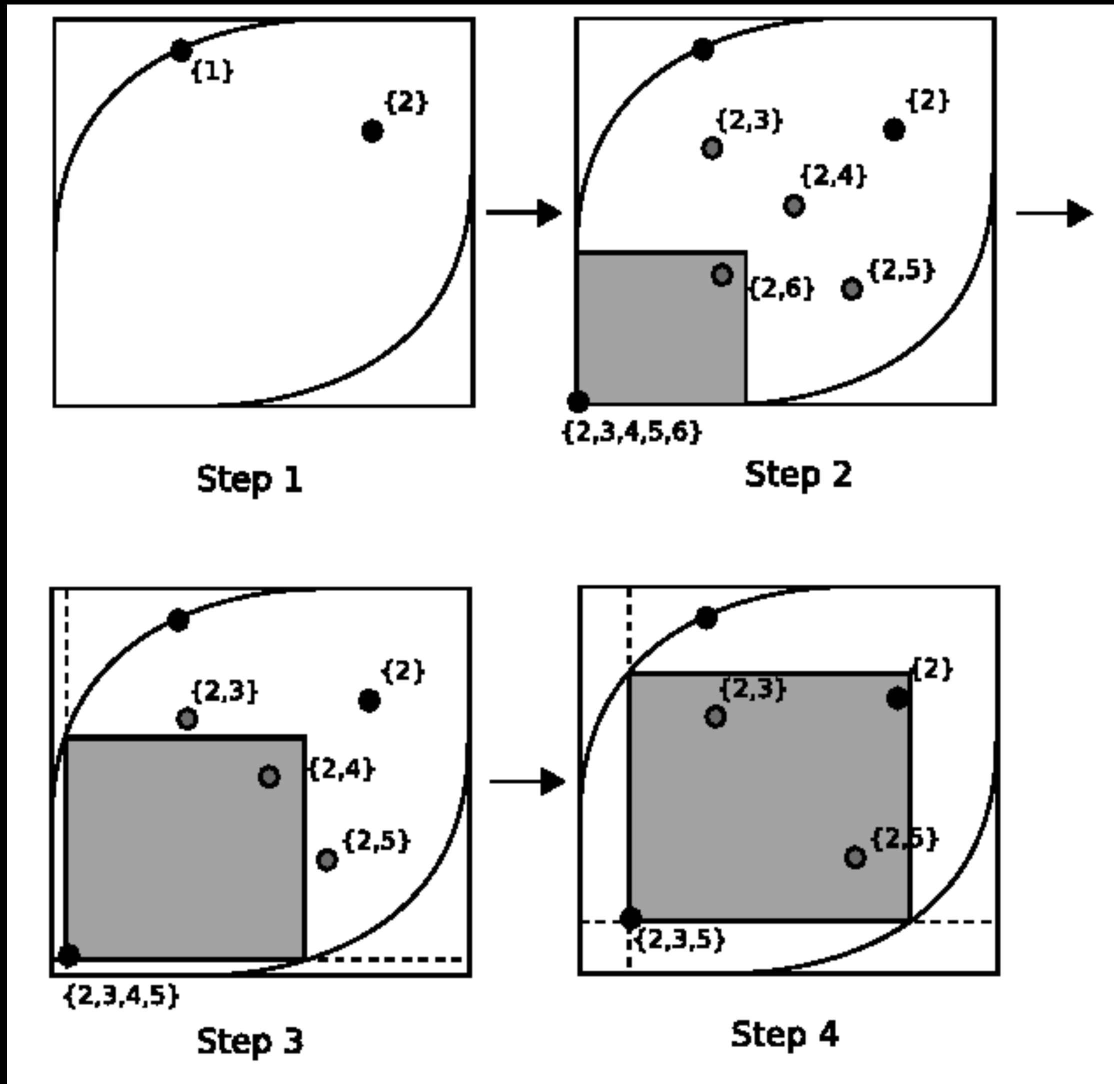
Solving using CP can be extremely effective

Illustration

Dom({2})=
{2,3,4,5,6}

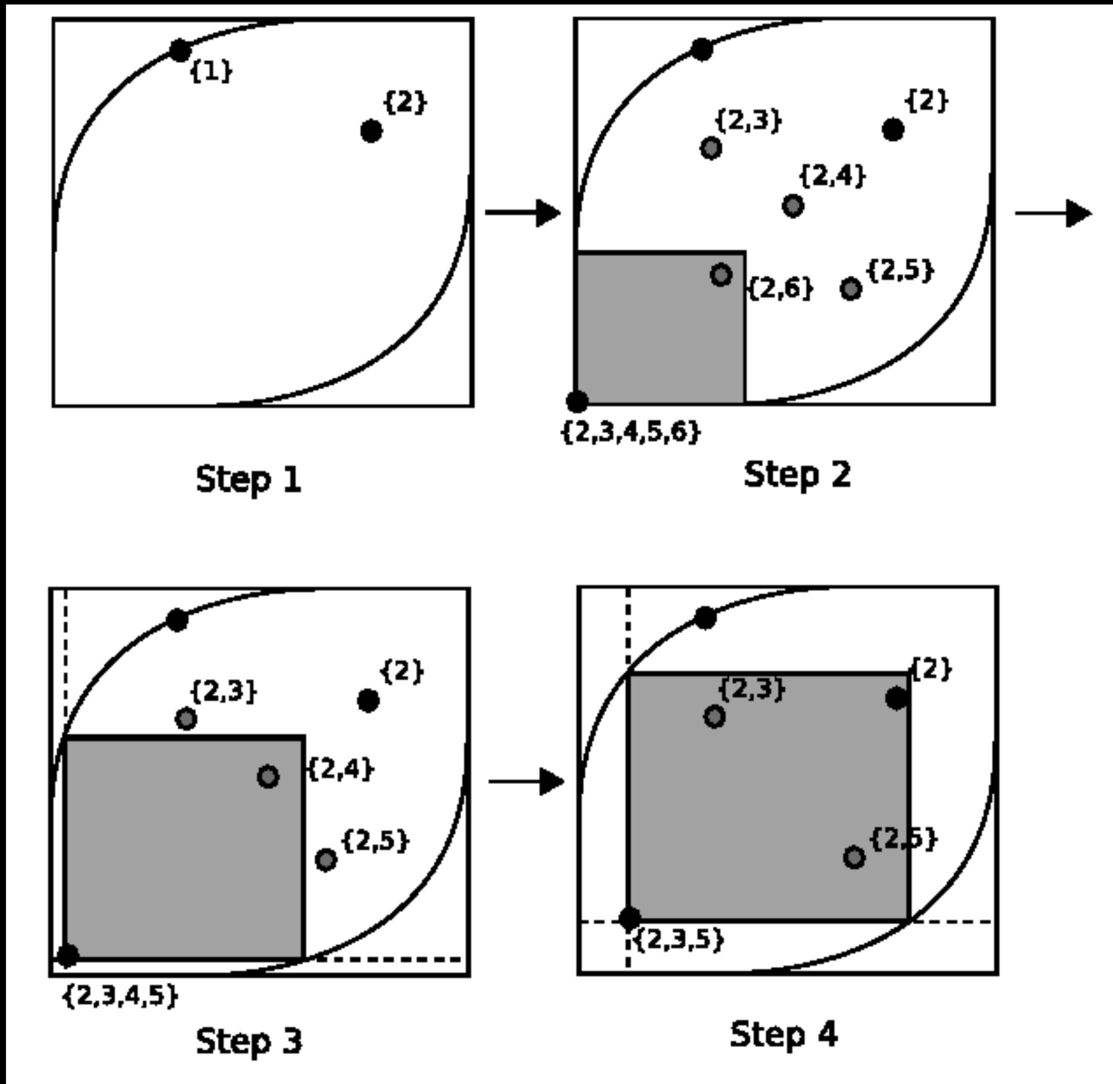


Illustration



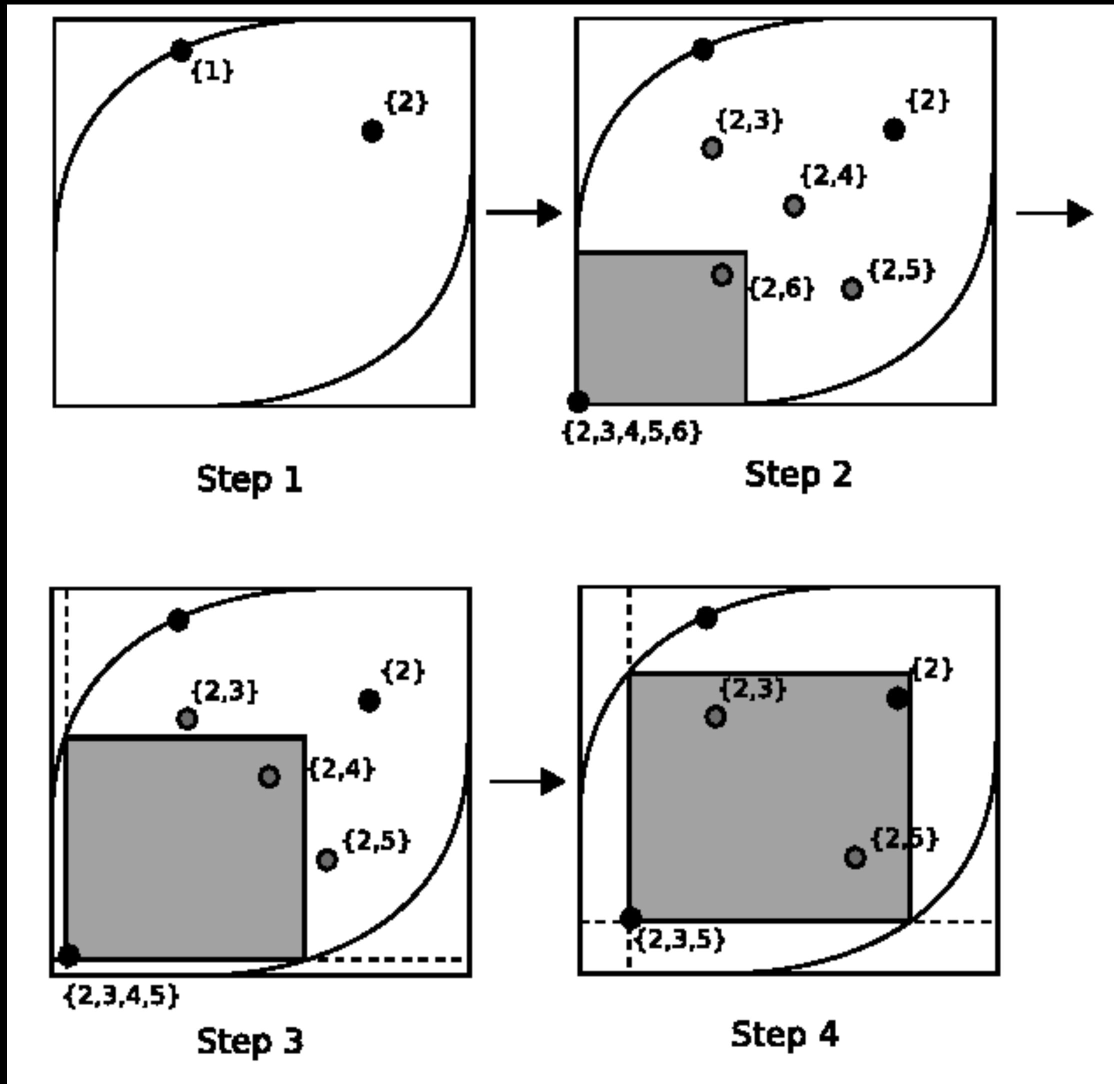
Dom({2})=
{2,3,4,5}

Illustration



Dom({2})=
{2,3,5}

Illustration



Dom({2})=
{}

4-support bound

Nijssen et al. KDD 09 AIJ 11

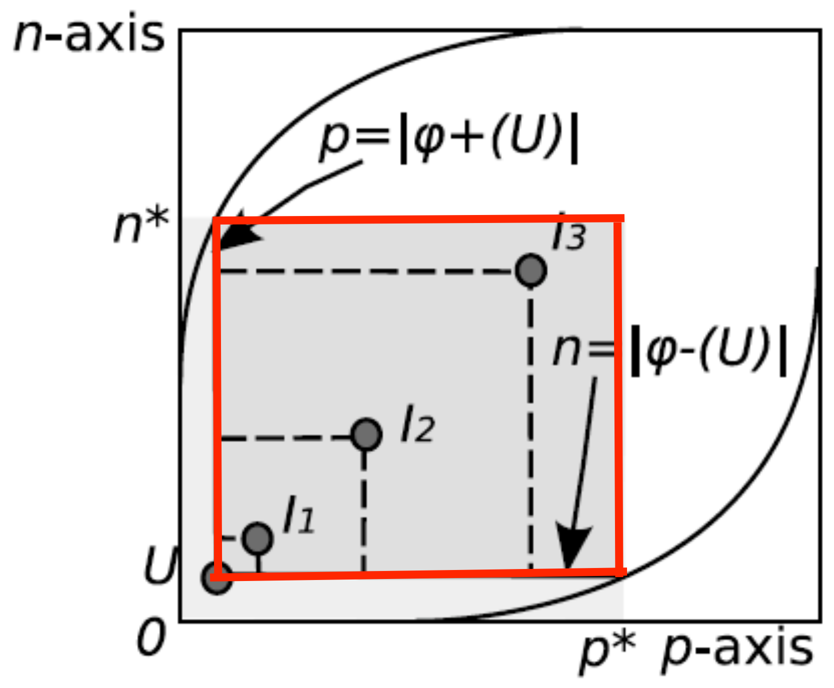


Figure 4: The 4-support bound in PN-space.

2-support bound

Morishita &
Sese
SIGMOD98

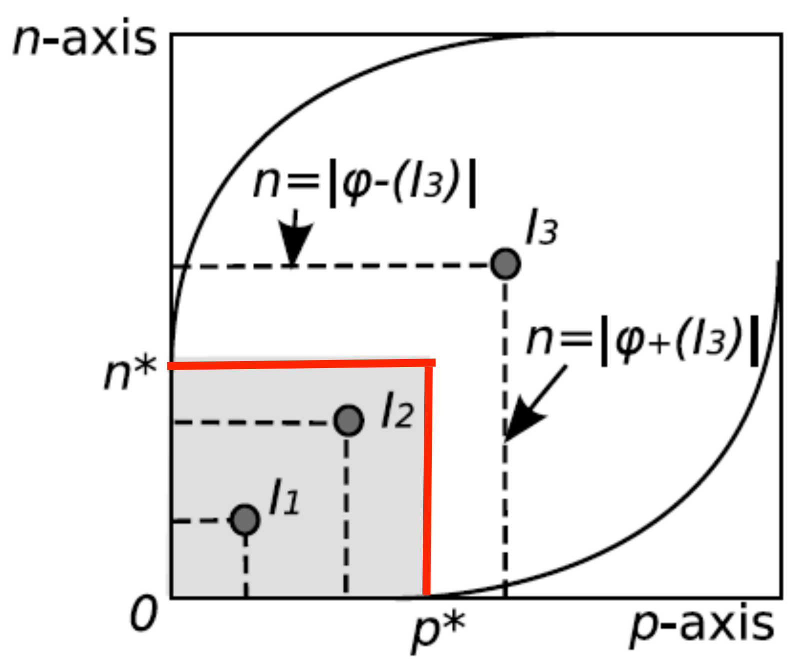


Figure 3: The 2-support bound in PN-space.

1-support bound

Han et al.
ICDM 08

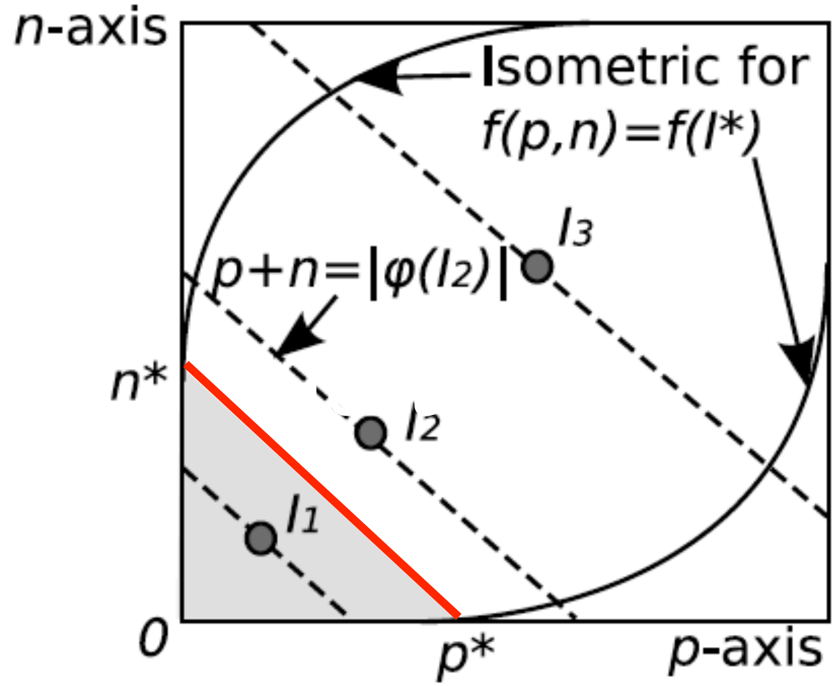


Figure 2: The 1-support bound in PN-space.

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

Solving using CP can be extremely effective

C. Pattern Set Mining

Pattern Sets

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

One is not interested in all solutions to a pattern mining task, typically post-processing needed

So, why not apply constraint based mining to pattern sets directly ? [Zimmermann 09] [Guns et al, IEEE TKDE 11]

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

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

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, ...

k-Pattern *Set* Mining (|P|=k)

```
int: NrI; int: NrT;        int K;
array[1..NrT] of set of int: D;
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 { Dp = [D[t] | t in pos] } in
        forall (d in 1..K) (
                Items[d] = cover_inv(cover(Items[d], Dp), Dp));

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

Generality

Can model instantiations/versions of:

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

Pattern Mining

A. frequent pattern

KDD 08

- which patterns are frequent ?

$$Th(\mathcal{L}, Q, \mathcal{D}) = \{$$

B. Correlated pattern

- which p … k-best pat…

KDD 09

$$\ldots_k \; \phi(p, \mathcal{D})$$

C. patte…

- whi… t concept-description for the actives ?
for th…

IEEE TKDE 11

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

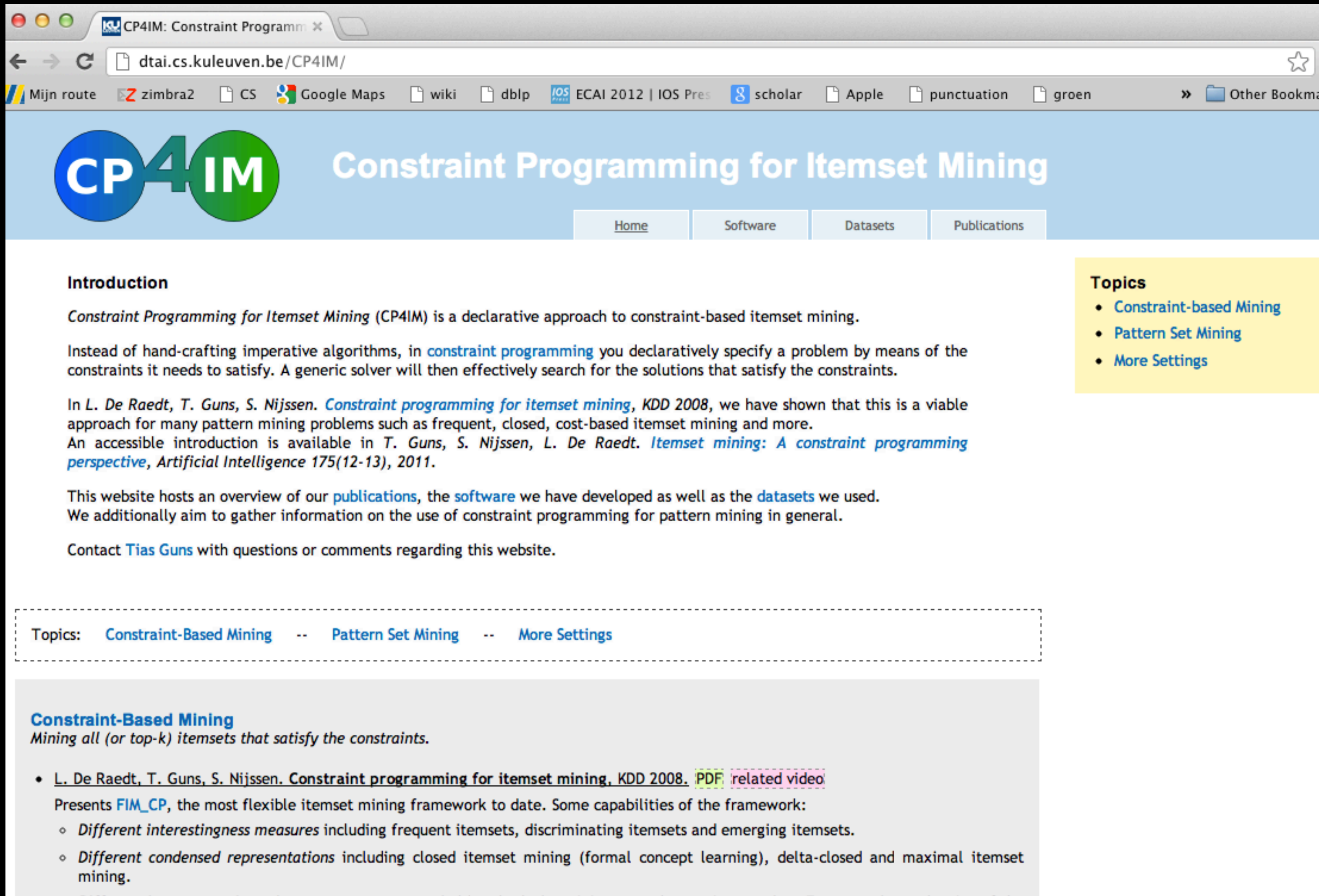
We have been using off-the-shelf CP SOLVERS for these tasks, cf. Guns, Nijssen, De Raedt [AAAI 10, AIJ 11]

One solver for all of these

Easy to combine different constraints

Now looking at modeling level

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



Perspective

All this is fine but...

what about

- efficiency and scalability ?
- other types of data and patterns (sequences, trees, graphs ...) ? relational
- other DM/ML tasks ? probabilistic, statistical learning, kernels / distances ...

Efficiency / Scalability

- Trade-off efficiency / generality
- Current experiments (with ONE solver)
  - Often a constant factor slower
  - Some cases much faster (correlated)
  - Avoid with specialized solvers [Nijssen and Guns, ECMLPKDD 10]
- Feature of Declarative Modeling
  - many solvers available (complete, approximate, ...)
  - one can even work with portfolio's (Satzilla)
- Challenge is to build efficient solvers and translations

The new role of DM/ML scientists if we succeed ?

Task / Representation

Rich representations ~ relational, graphs ?

Task level ~ unsupervised, regression, clustering, probabilistic... ?

Let us have a look at Statistical Relational Learning

Markov Logic [Domingos et al.]

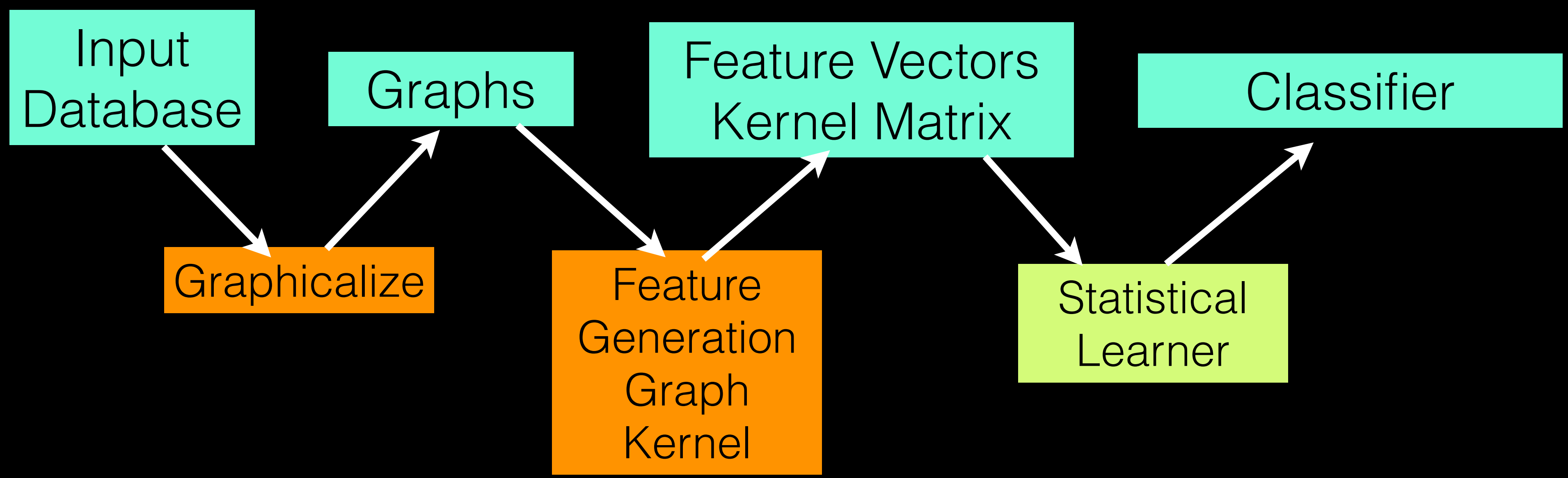
probabilistic

ProbLog [De Raedt et al.]

...

kLog [Frasconi et al.] kernel based

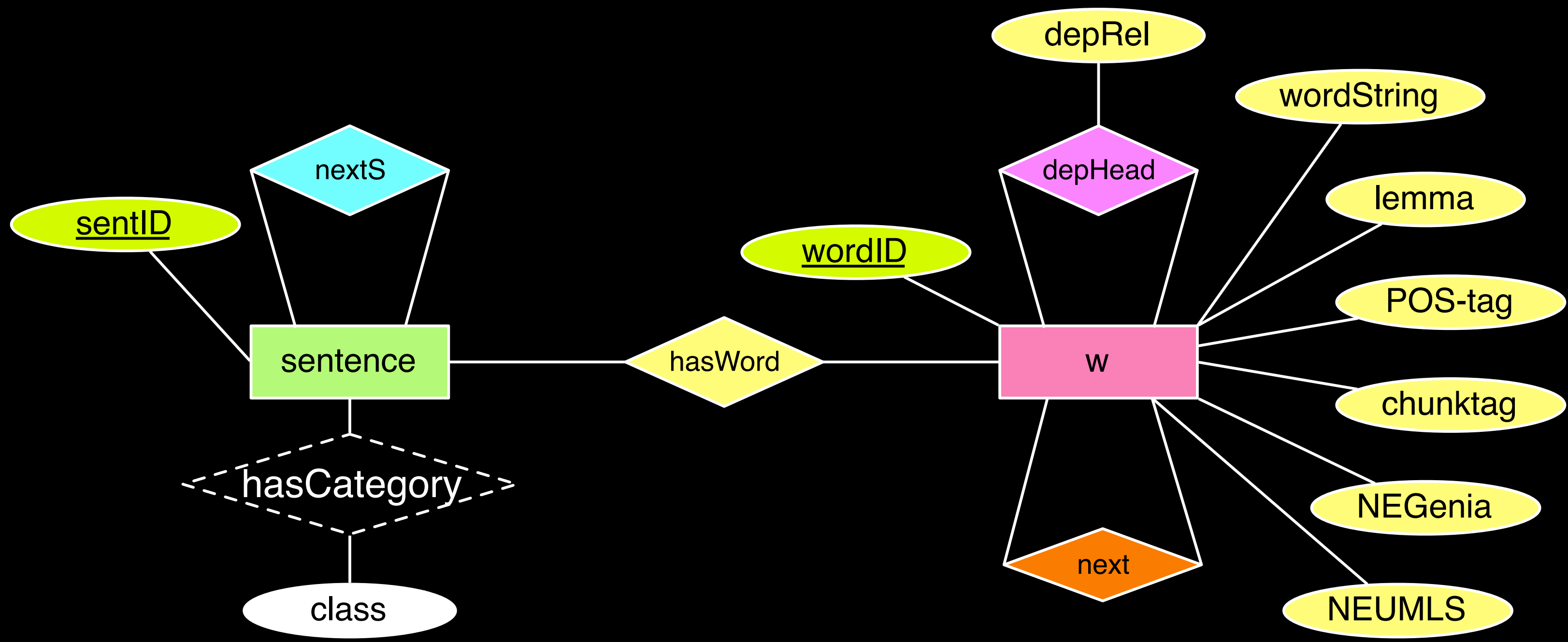
kLOG [Frasconi, Costa, DR, De Grave 12]



A biomedical NLP task [Verbeke et al. EMLNP 12]

Surgical excision of CNV may allow stabilisation or improvement of vision .

E/R-MODEL



[Verbeke et al. EMNLP 12]

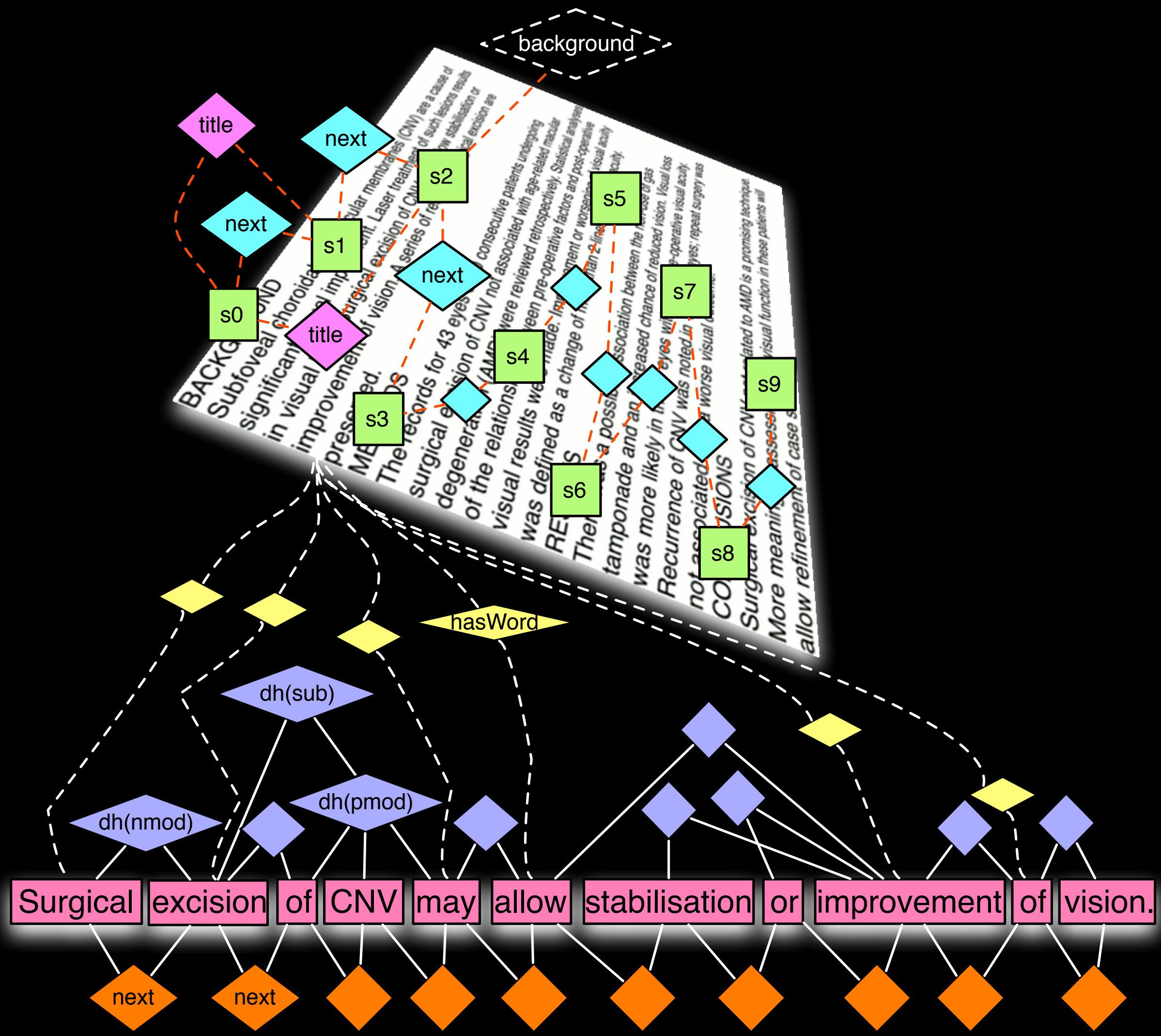
Relational ....

```
sentence(s4,4).
hasCategory(s4,'background').
w(w4_1,'Surgical','Surgical',b-np,jj,'O','O').
hasWord(s4,w4_1).
dh(w4_1,w4_2,nmod).
nextW(w4_2,w4_1).
w(w4_2,'excision','excision',i-np,nn,'O','O').
hasWord(s4,w4_2).
dh(w4_2,w4_5,sub).
nextW(w4_3,w4_2).
w(w4_3,'of','of',b-pp,in,'O','O').
hasWord(s4,w4_3).
dh(w4_3,w4_2,nmod).
nextW(w4_4,w4_3).
w(w4_4,'CNV','CNV',b-np,nn,'B-protein','O').
hasWord(s4,w4_4).
dh(w4_4,w4_3,pmod).
nextW(w4_5,w4_4).
w(w4_5,'may','may',b-vp,md,'O','O').
hasWord(s4,w4_5).
dh(w4_5,w4_0,root).
nextW(w4_6,w4_5).
```

```
lemmaRoot(S,L) :-
    hasWord(S, I), w(I,_,L,_,_,_,_), dh(I,_,root).

isHeaderSentence(S):-
  hasHeaderWord(S,_).

hasSectionHeader(S,X):-
  nextS(S1,S),
  hasHeaderWord(S1,X),!.
hasSectionHeader(S,X):-
  nextS(S1,S),
  \+isHeaderSentence(S),
  once(hasSectionHeader(S1,X)),!.
```

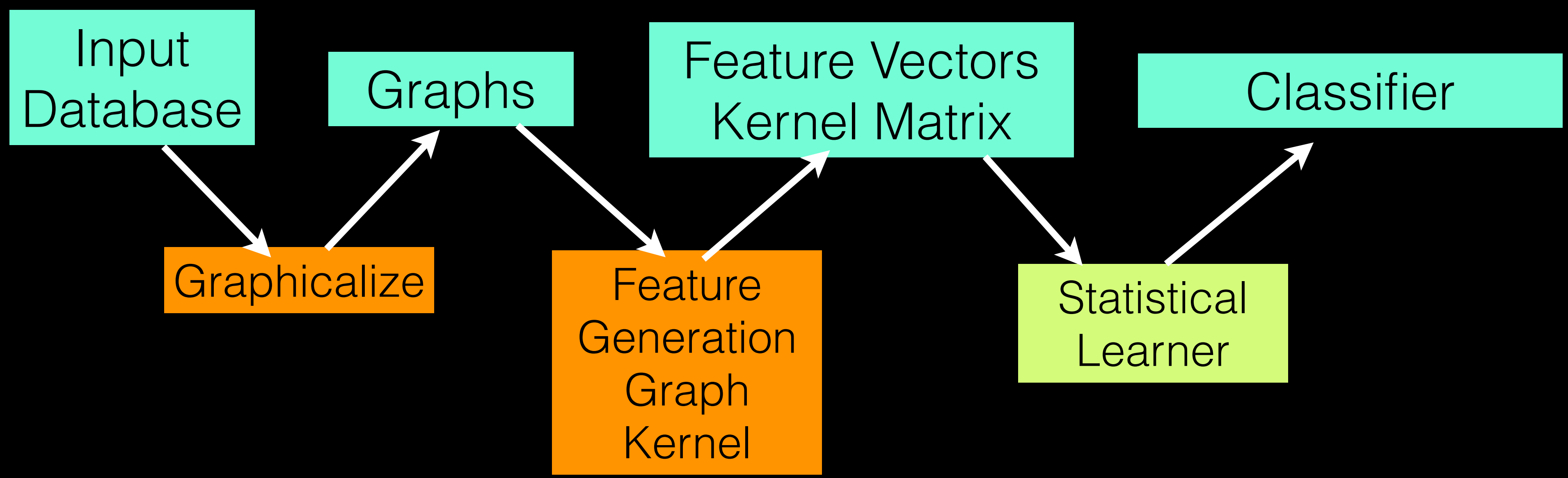
...

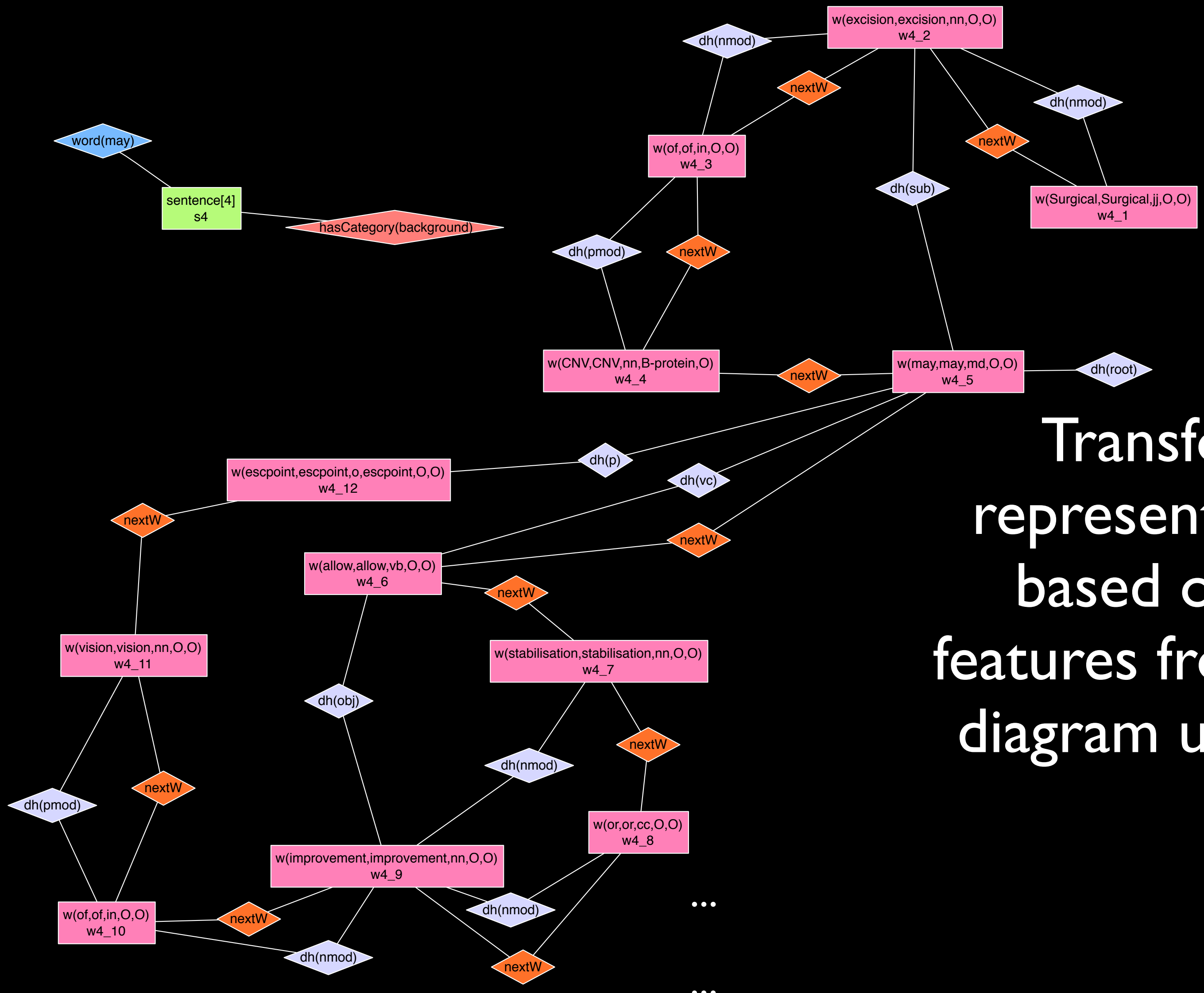
relational database ... Prolog



Graphicalization

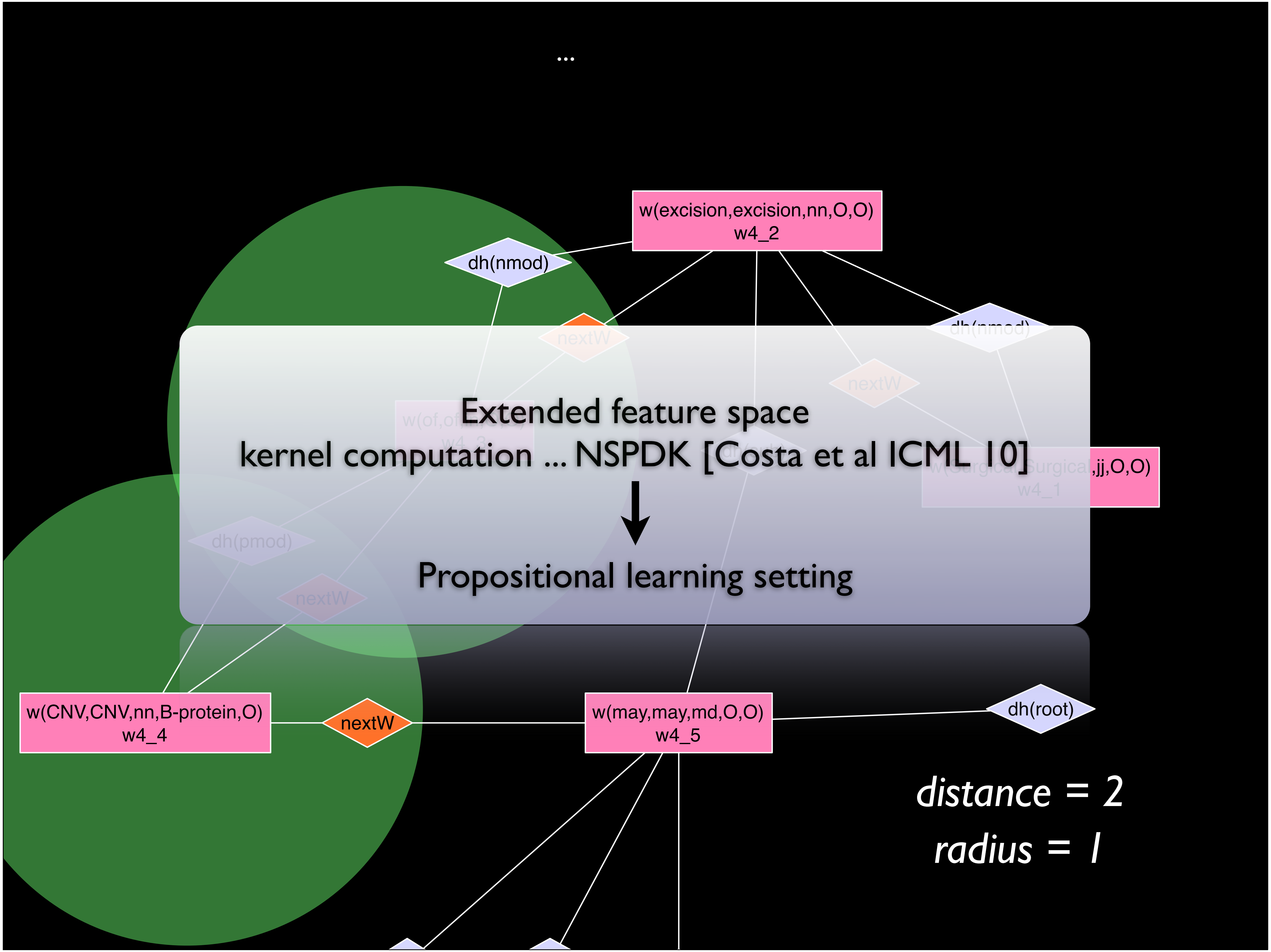
kLOG


Input Database
Graphs
Feature Vectors Kernel Matrix
Classifier
Graphicalize
Feature Generation Graph Kernel
Statistical Learner

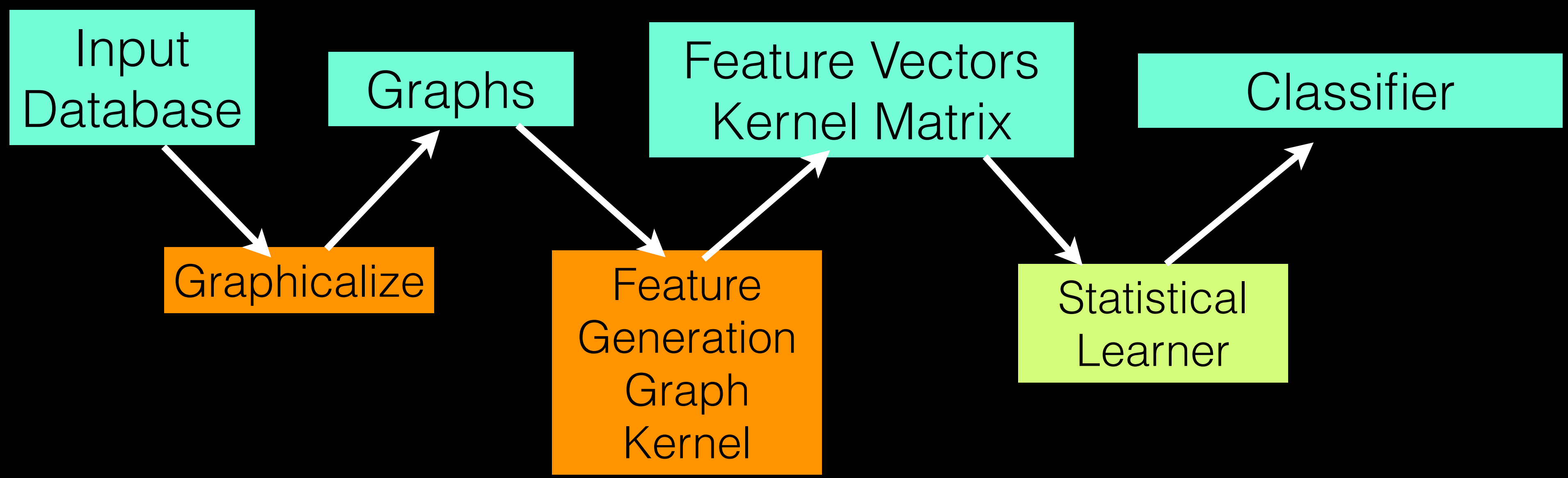


Transforms relational representations into graph based ones and derives features from a grounded E/R diagram using graph kernels

...


...
w(excision,excision,nn,O,O)
w4_2
dh(nmod)
Extended feature space
kernel computation ... NSPDK [Costa et al ICML 10]
Propositional learning setting
w(CNV,CNV,nn,B-protein,O)
w4_4
nextW
w(may,may,md,O,O)
w4_5
dh(root)
distance = 2
radius = 1

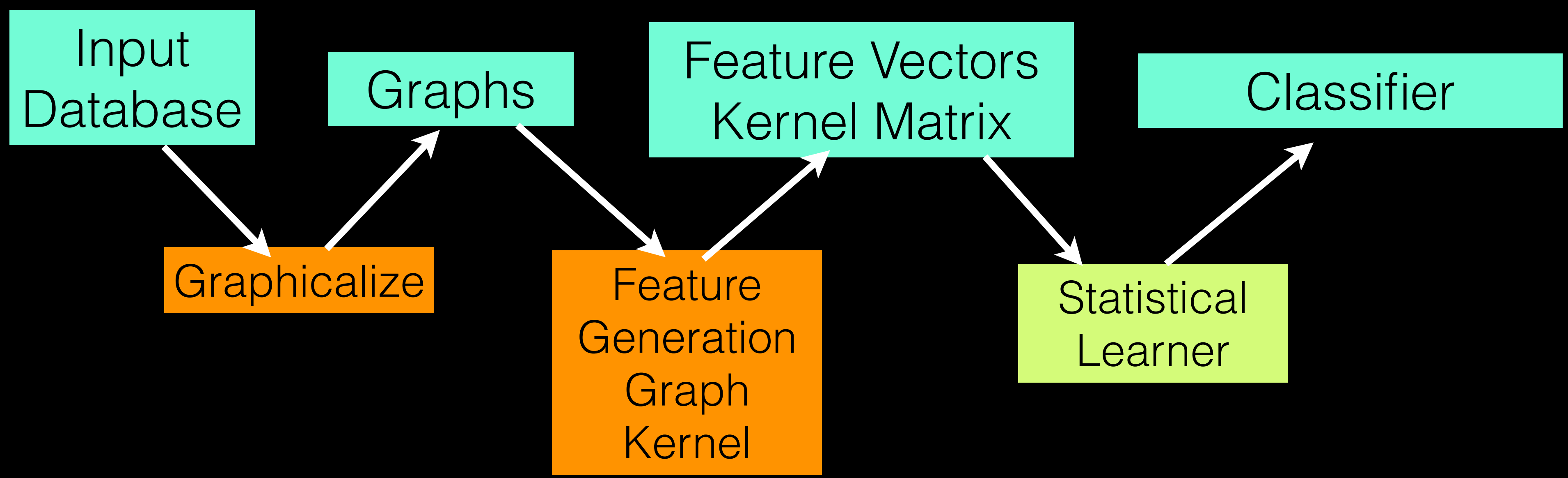
kLOG


Input Database
Graphs
Feature Vectors Kernel Matrix
Classifier
Graphicalize
Feature Generation Graph Kernel
Statistical Learner

kLOG

Task

Task MODEL specifies task =
constraints + optimization criterion



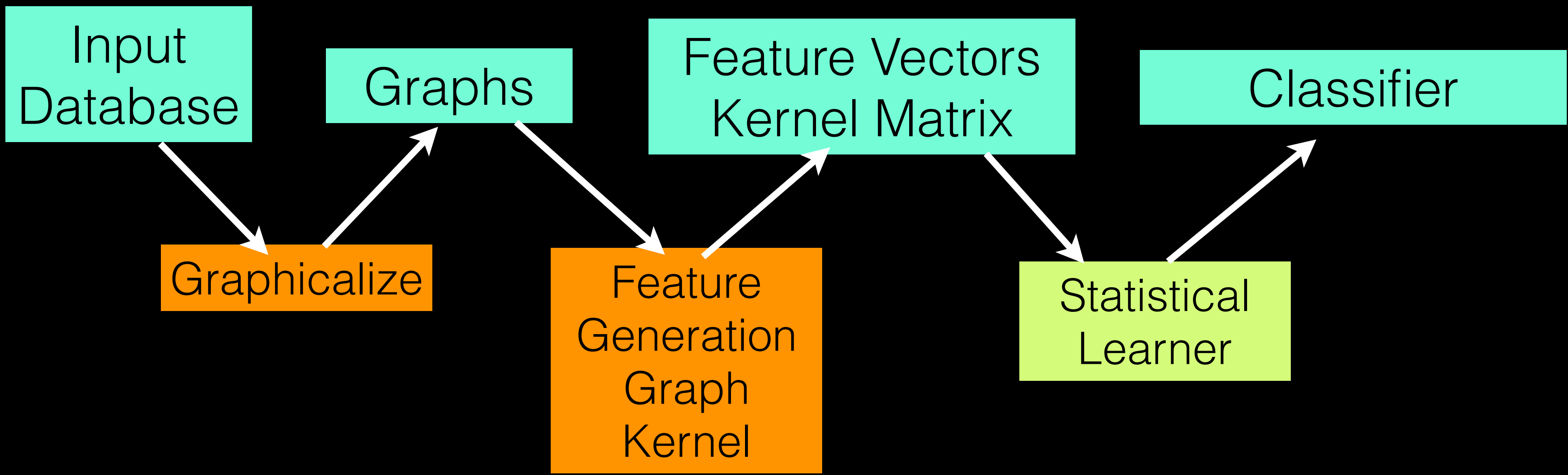
kLOG

Task

CHALLENGE

Define the KERNEL declaratively

Define the LOSS function ... and SOLVE



As for many other ML/DM systems ?

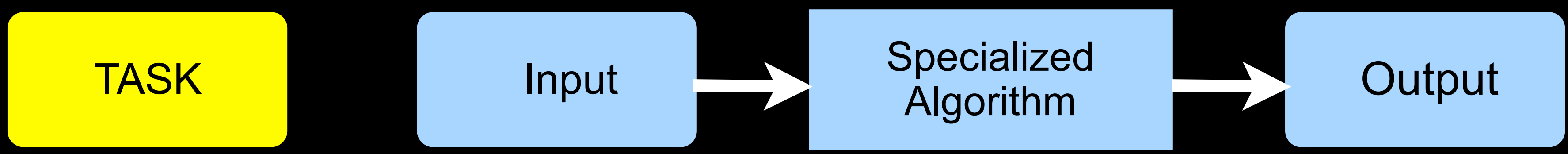
What if we succeed ?

Our work today ...

We typically ...

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

hard



Our work tomorrow ...

The user/application perspective...

1. Formalize learning / mining task
2. Model the problem
3. Select the right solvers
4. Use and distribute the software

easy

More opportunities for re-use ...

A de facto standard language for DM / ML as Zinc ?

Our work tomorrow ...

The scientist's perspective...

Designing modeling languages

Studying task properties

more fun

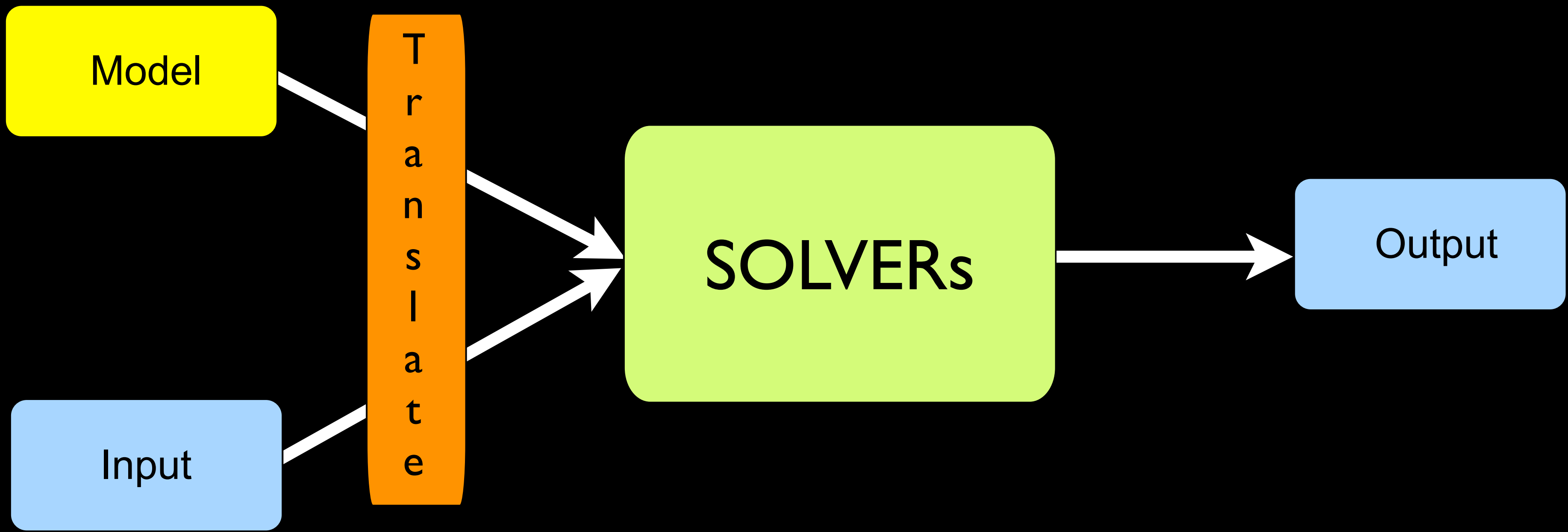
Studying translations

Producing and adapting solvers

Larger impact of results in larger framework ?

Conclusions

Declarative modeling languages for ML / DM has high potential



Embedding in programming languages may provide an answer to Mitchell's question

Conclusions

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

- machine learning & data mining
- declarative modeling and constraint programming
- programming language technology
- should work for unsupervised, clustering ... as well

So let's do it ...

Questions ?

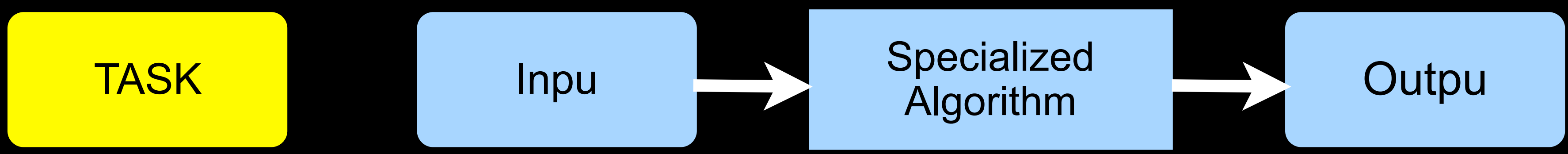
What if we succeed?

Our work today ...

We typically have to

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

Focus on HOW to solve problem



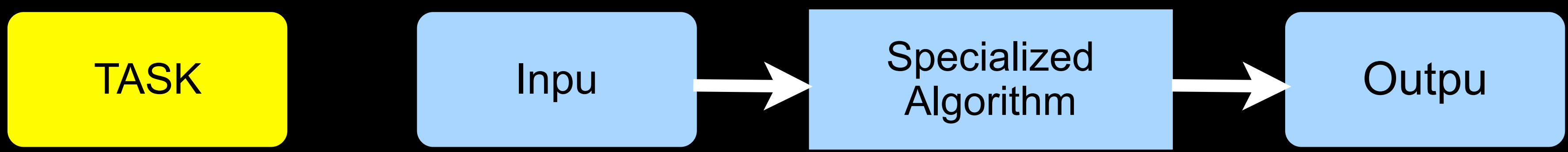
Our work tomorrow ...

We typically have to

1. Formalize learning / mining task
2. Build a declarative mode

More user perspective

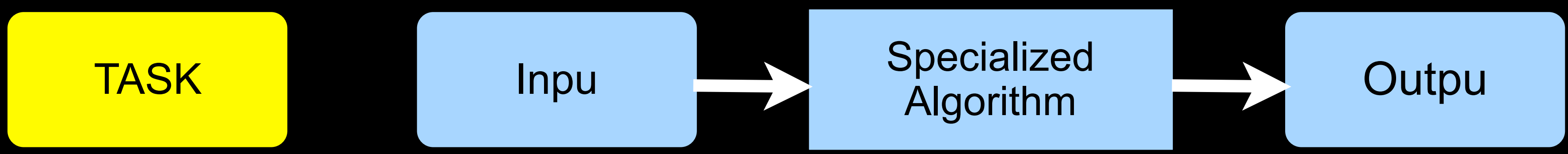
4. Use and distribute the software



Our work tomorrow ...

1. Work on Engines
2. Solvers translations ...

4. Use and distribute the software



Our work tomorrow ?

Work on modeling

Theory ...

Work on translations

Work on solvers

Much more reuse of systems and techniques

Questions ?