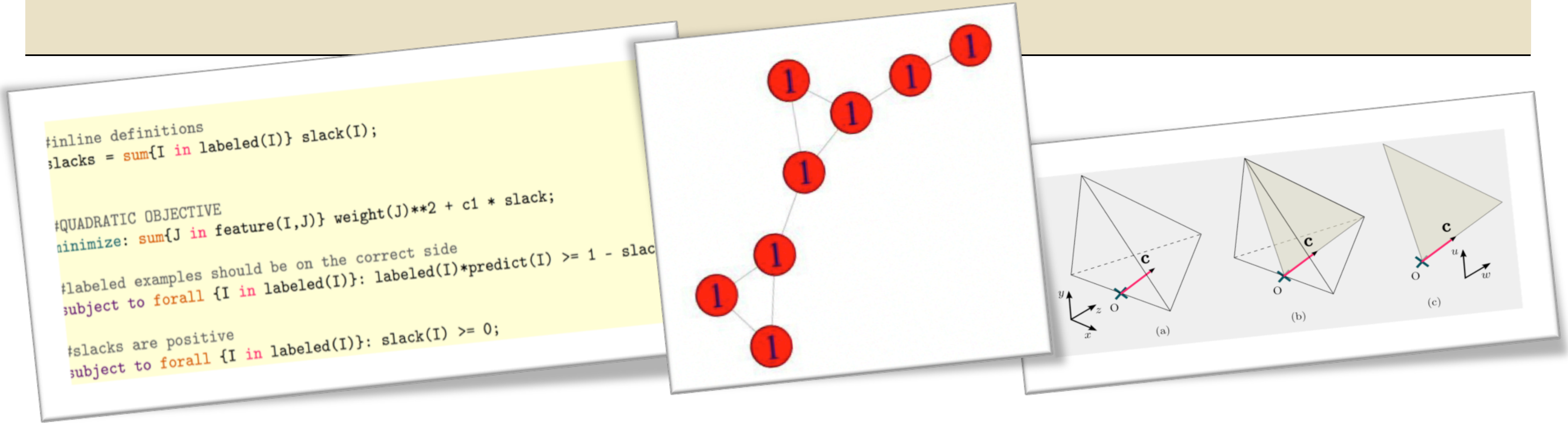
Relational QPs

Exploiting Symmetries for Modelling and Solving QPs



Kristian Kersting

Martin Mladenov
TUD, Google

Sriraam Natarajan
U. Indiana

Amir Globerson
HUJI

Babak Ahmadi
PicoEgo

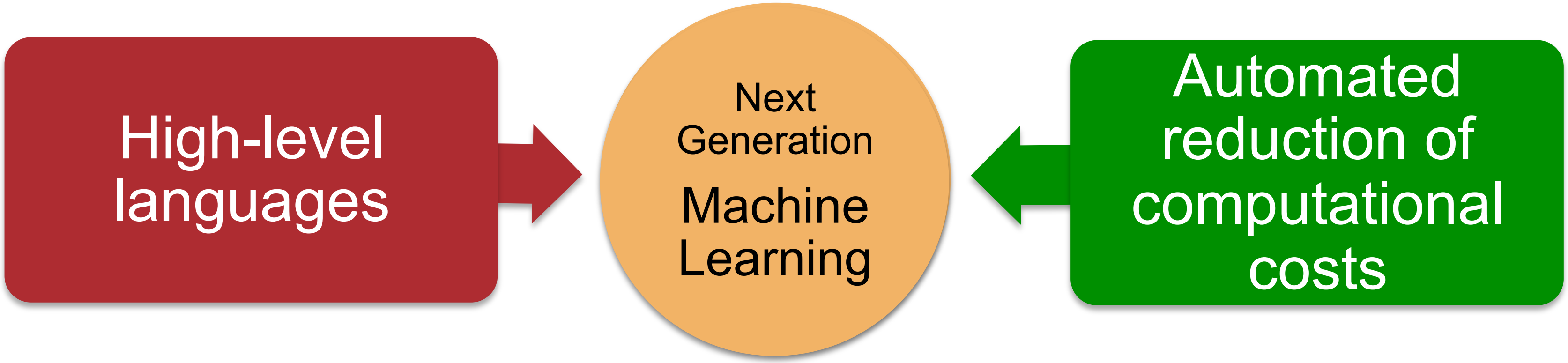
Christopher Re
Stanford

Martin Grohe
RWTH Aachen

Pavel Tokmakov
INRIA Grenoble

Take-away message

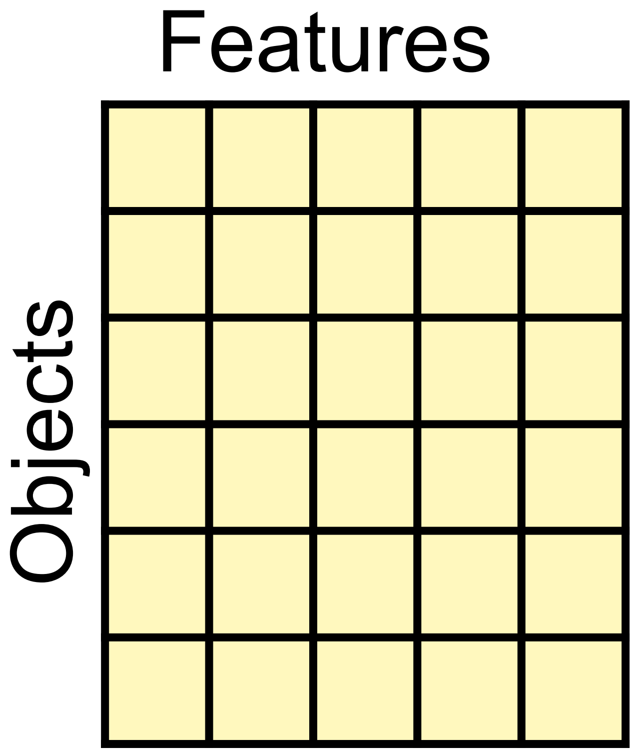
Statistical Machine Learning (ML) needs a crossover with data and programming abstractions



- ML high-level languages increase the number of people who can successfully build ML applications and make experts more effective
- To deal with the computational complexity, we need ways to automatically reduce the solver costs

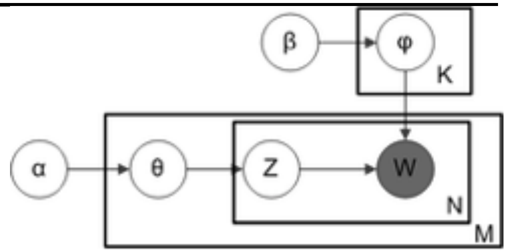
Arms race to deeply understand data

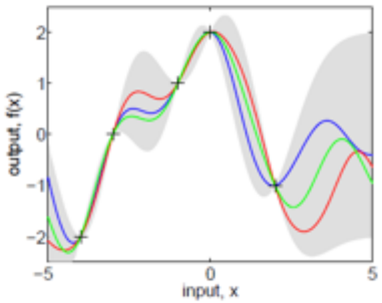
Bottom line:
Take your data spreadsheet …



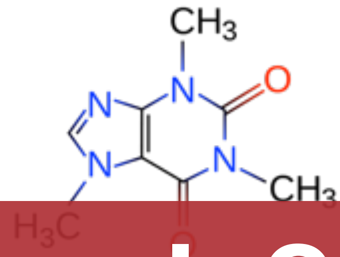
… and apply machine learning

Graphical models



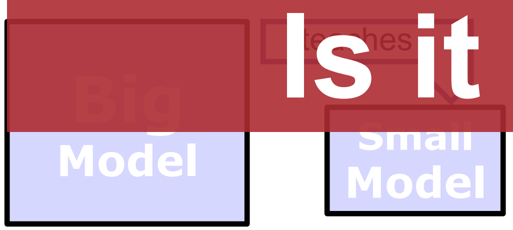


Gaussian Processes



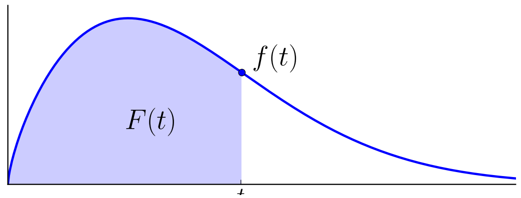
Graph Mining

Is it really that simple?

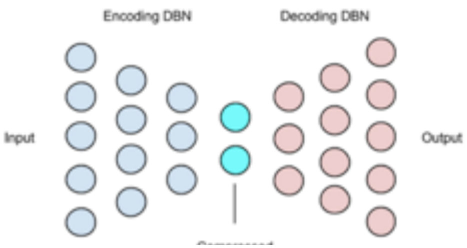


Distillation/LUPI

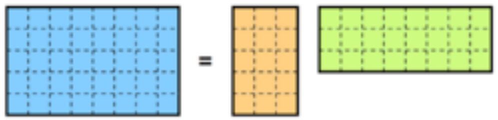
Boosting



Diffusion Models



Autoencoder,
Deep Learning

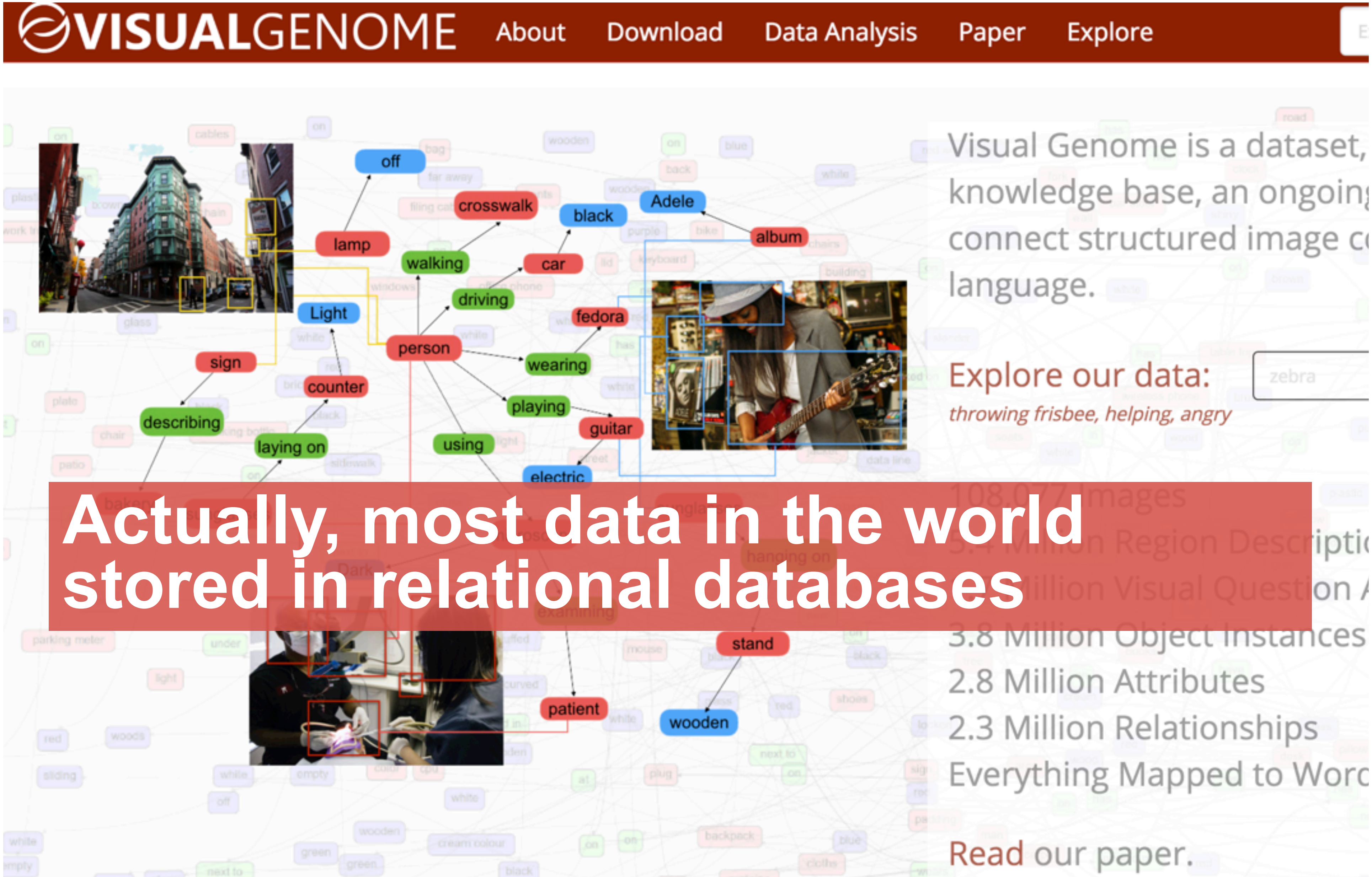


Big Data Matrix Factorization

and many more …

Complex data networks abound

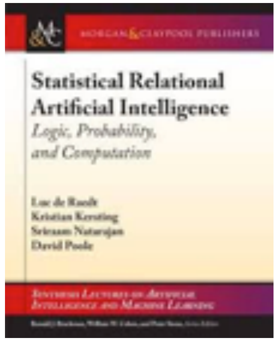
[Lu, Krishna, Bernstein, Fei-Fei „Visual Relationship Detection“ CVPR 2016]



Actually, most data in the world stored in relational databases

De Raedt, Kersting, Natarajan, Poole, Statistical Relational Artificial Intelligence: Logic, Probability, and Computation. Morgan and Claypool Publishers, ISBN: 9781627058414, 2016.

TECHNISCHE UNIVERSITÄT DARMSTADT



Punshline: Two trends that drive ML

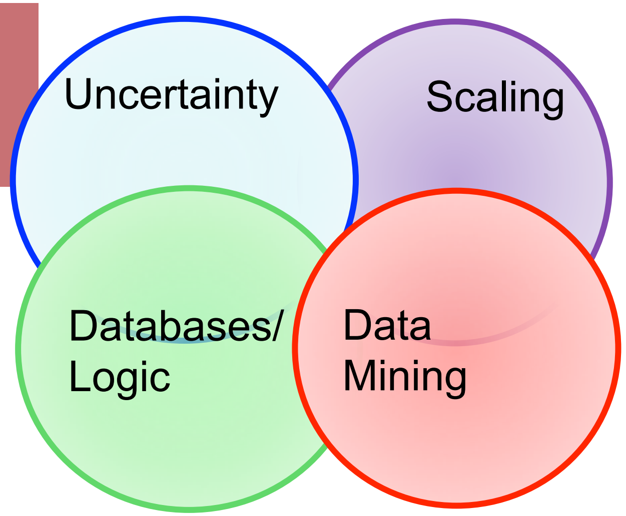
1. Arms race to deeply understand data
2. Data networks of a large number of formats

It costs considerable human effort to develop, for a given dataset and task, a good ML algorithm

Crossover of ML with data & programming abstractions

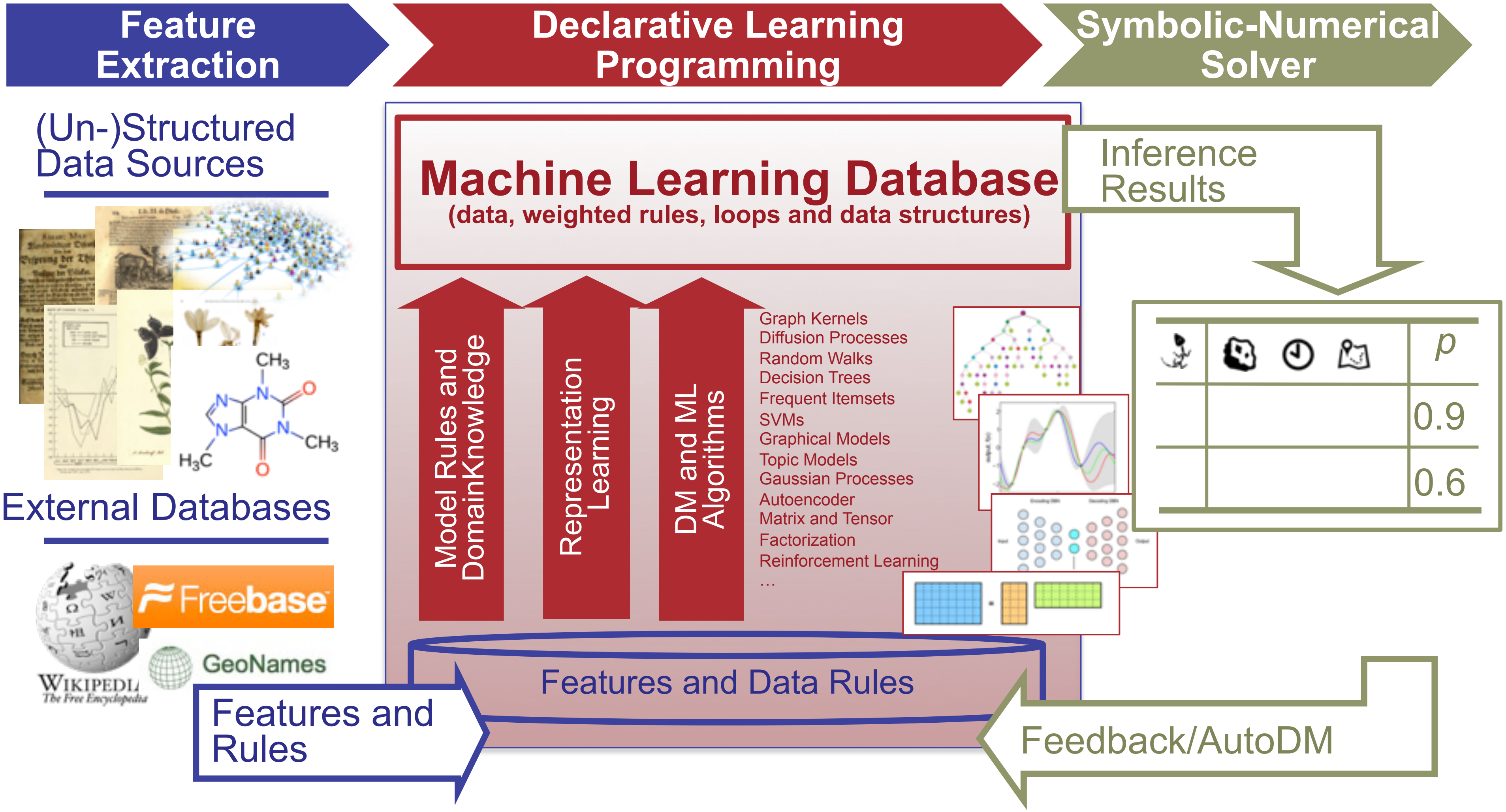
make the ML expert more effective

increases the number of people who can successfully build ML applications



Thinking Machine Learning

TECHNISCHE UNIVERSITÄT DARMSTADT



[Ré et al. IEEE Data Eng. Bull.'14; Natarajan, Picado, Khot, Kersting, Ré, Shavlik ILP'14; Natarajan, Soni, Wazalwar, Viswanathan, Kersting Solving Large Scale Learning Tasks'16, Mladenov, Heinrich, Kleinhans, Gonsior, Kersting DeLBP'16, …]

This connects the CS communities

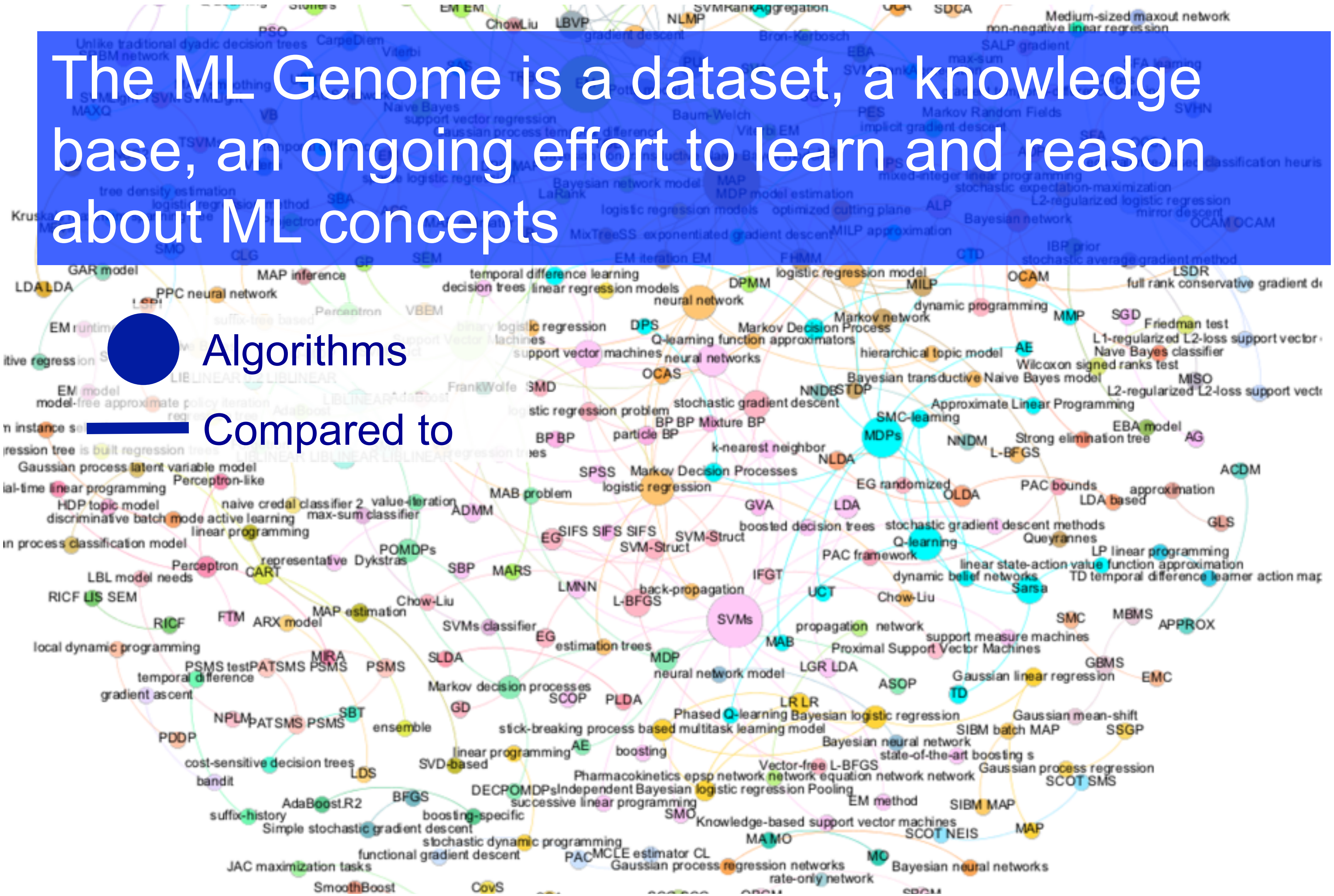
Jim Gray Turing Award 1998
“Automated Programming”

Mike Stonebraker Turing Award 2014
“One size does not fit all”

Data Mining/Machine Learning, Databases, AI, Model Checking, Software Engineering, Optimization, Knowledge Representation, Constraint Programming, Operation Research, … !

The Machine Learning Genome

[Bratzadeh 2016; Bratzadeh, Molina, Kersting „The Machine Learning Genome“ 2017]



The ML Genome is a dataset, a knowledge base, an ongoing effort to learn and reason about ML concepts

Algorithms

Compared to

A simple example

Guy van den Broeck
UCLA

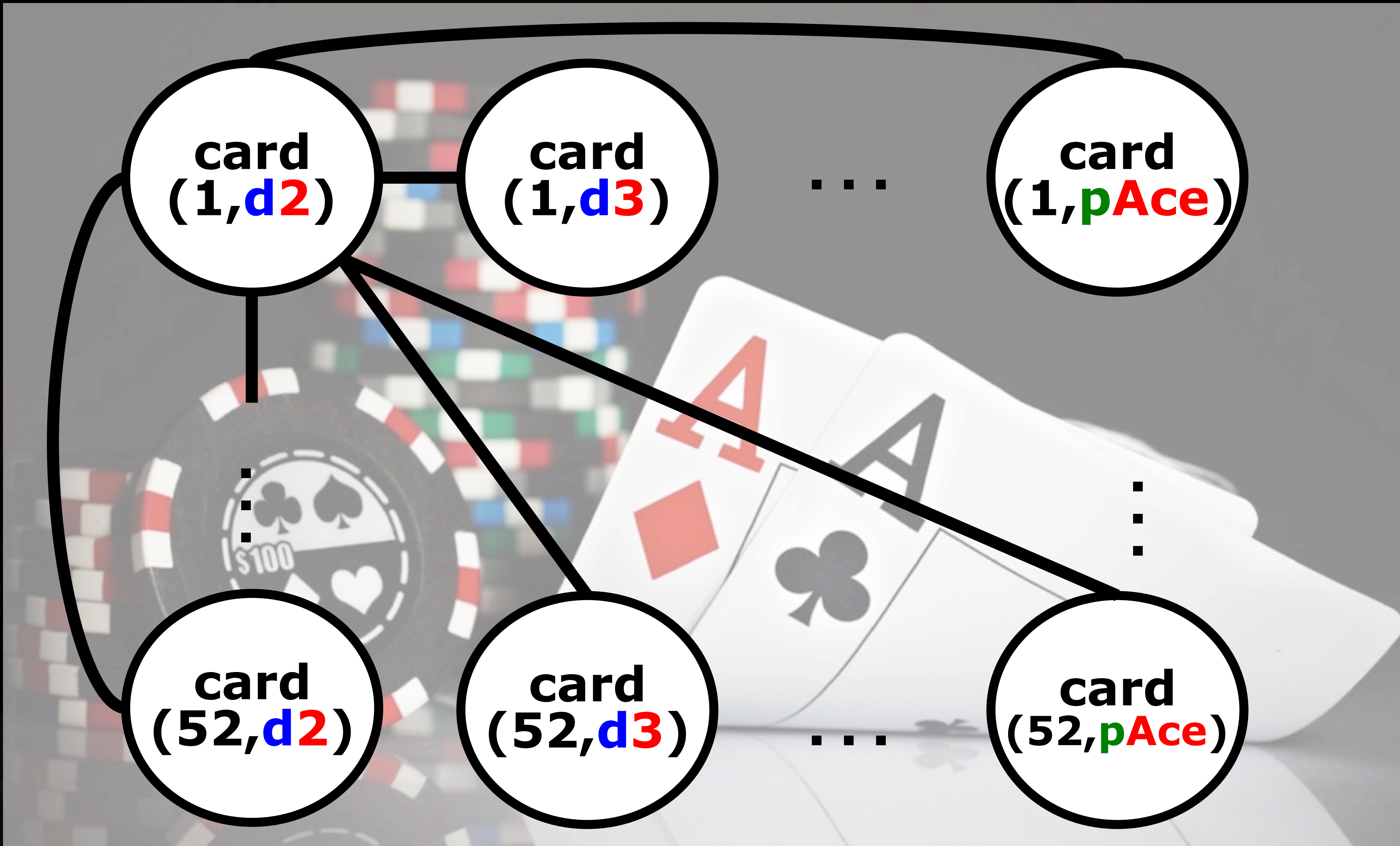
What is the problem that the first card of a randomly shuffled deck with 52 cards is an Ace?

How would a machine solve this?

One option is to treat this as an inference problem within in a graphical model, solved approximately using some mathematical program!

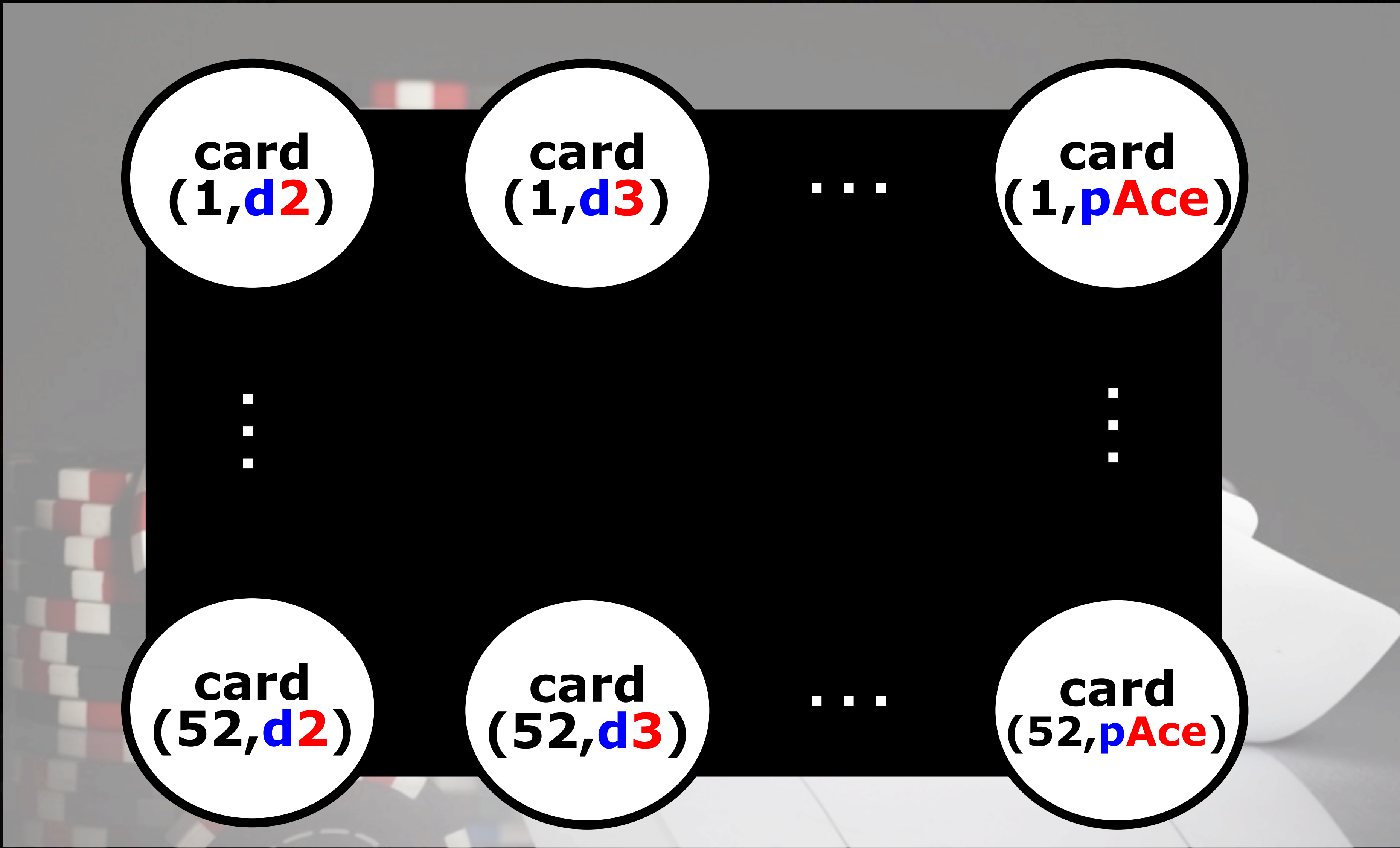
A simple example

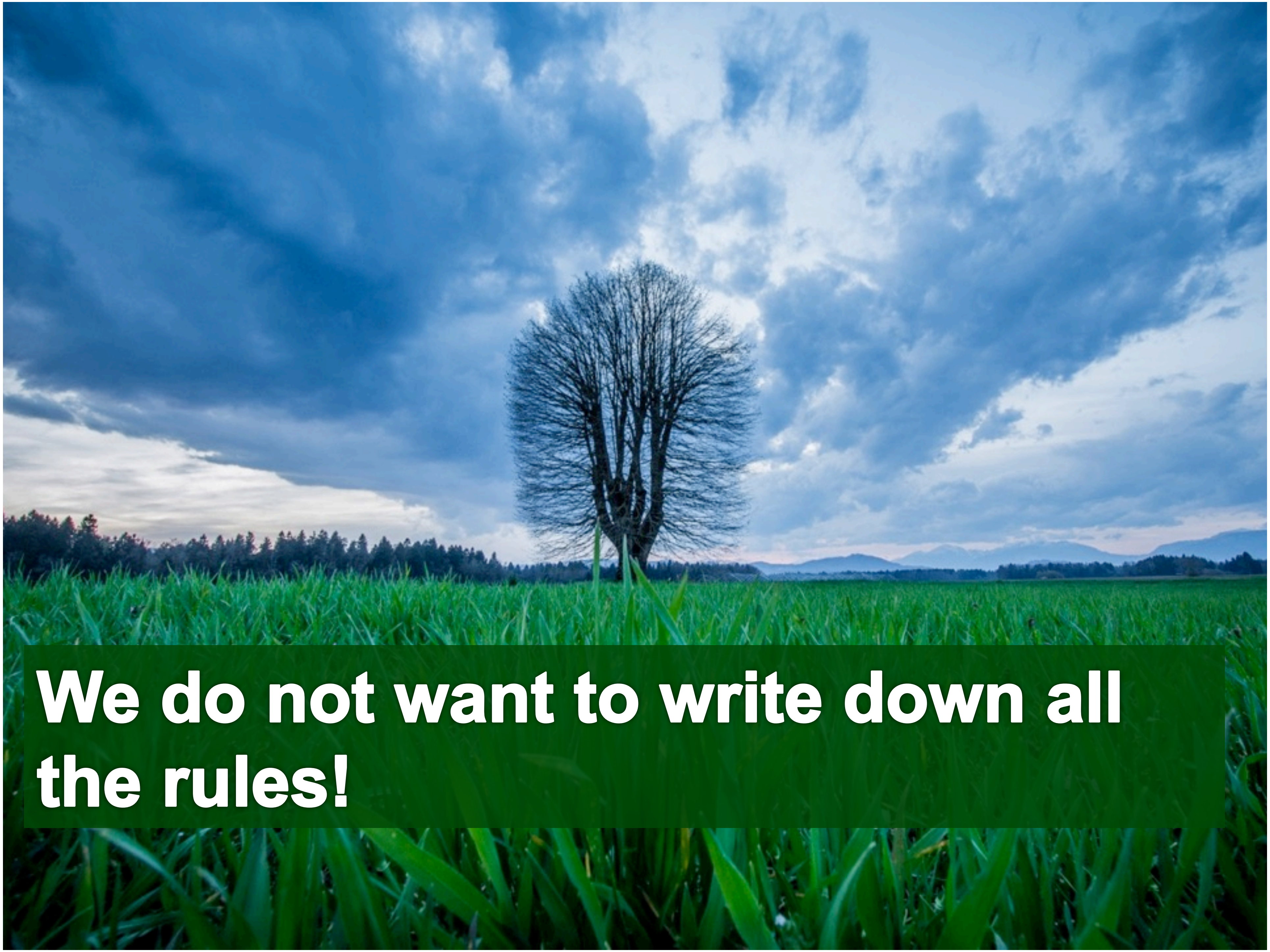
Guy van den Broeck
UCLA



A simple example

Guy van den Broeck
UCLA




We do not want to write down all the rules!

Faster modelling

Let's use programming abstractions such as e.g.

$$w1: \forall p,x,y: card(P,X),card(P,Y) \Rightarrow x=y$$

$$w2: \forall c,x,y: card(X,C),card(Y,C) \Rightarrow x=y$$

We do not want to write down all the rules!

A simple example

Guy van den Broeck
UCLA

card(1,d2) card(1,d3) … card(1,pAce)

⋮ What about inference? ⋮

card(52,d2) card(52,d3) … card(52,pAce)

A simple example

Guy van den Broeck
UCLA

card (1,d2) card (1,d3) … card (1,pAce)

⋮ **No independencies.** ⋮

Fully connected.

2^{2704} states

card (52,d2) card (52,d3) … card (52,pAce)

A simple example

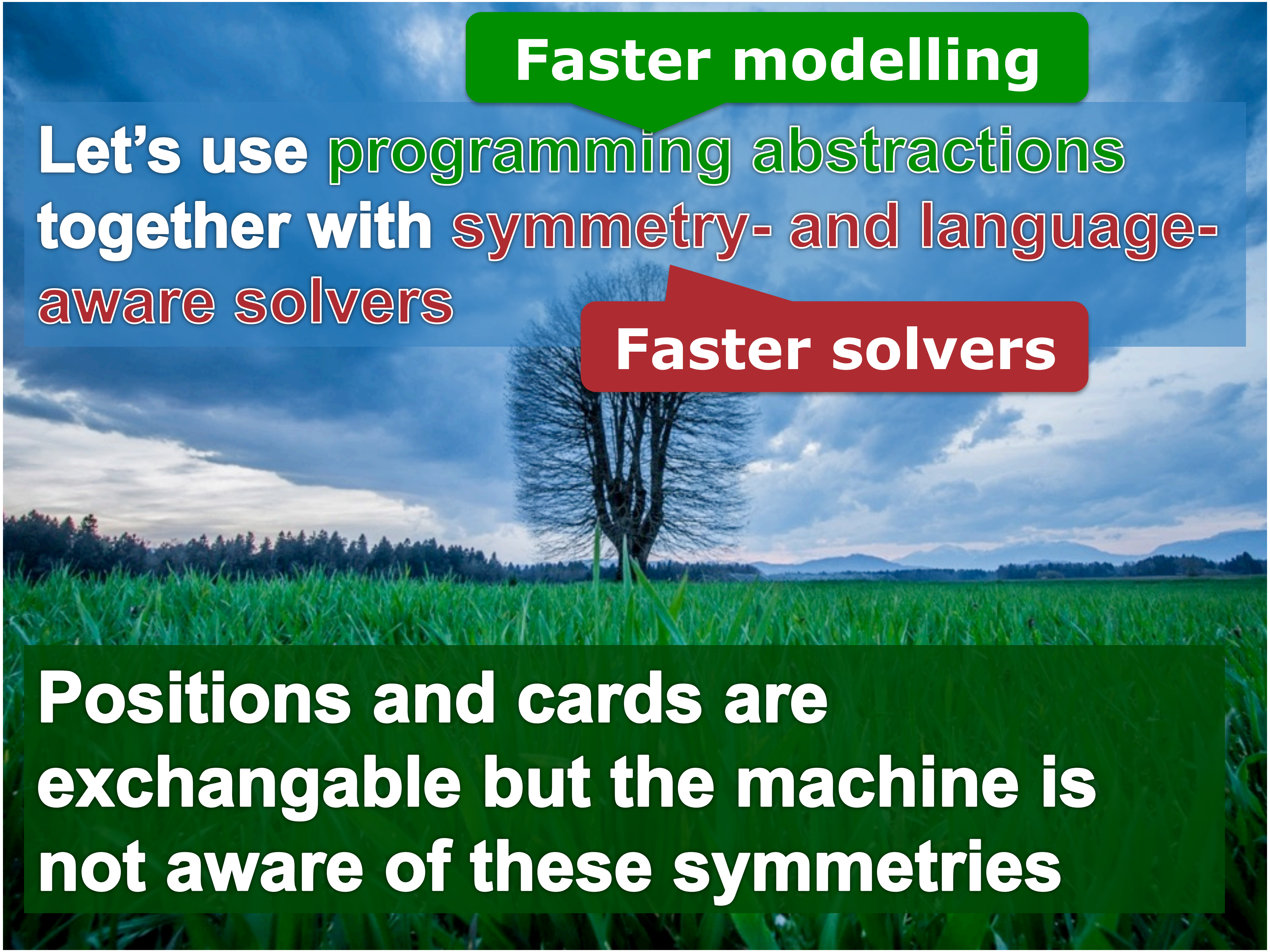
Guy van den Broeck
UCLA

card (1,d2) card (1,d3) … card (1,pAce)

⋮ **A machine will not solve the problem** ⋮

card (52,d2) card (52,d3) … card (52,pAce)


What are we missing?
Positions and cards are exchangable but the machine is not aware of these symmetries

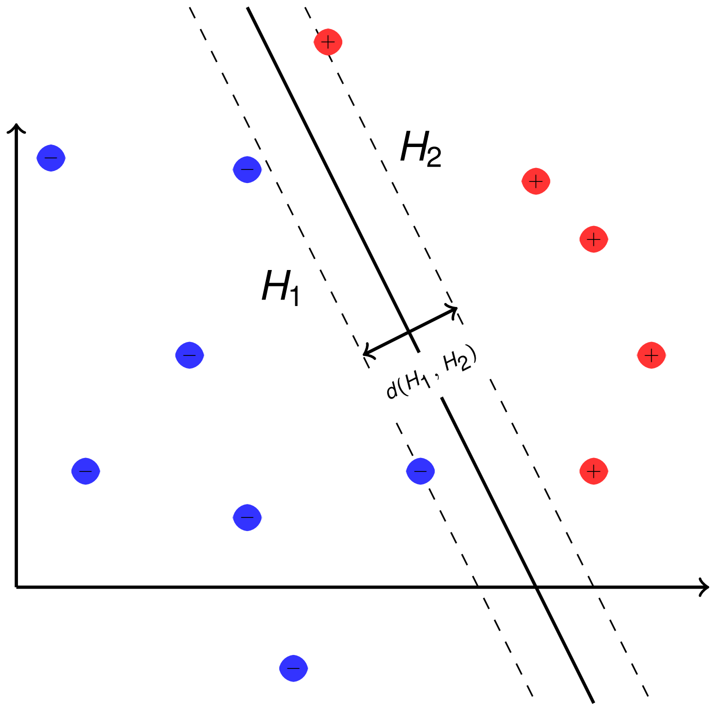

Faster modelling
Let's use programming abstractions together with symmetry- and language-aware solvers
Faster solvers
Positions and cards are exchangable but the machine is not aware of these symmetries

Let’s say we want to classify publications into scientific disciplines

Classification using LP SVMs

[Bennett´99; Mangasarian´99; Zhou, Zhang, Jiao´02, ...]

$$H^* = \left\{ \vec{x} \mid \left\langle \vec{x}, \vec{\beta} \right\rangle + \beta_0 = 0 \right\}$$



$$d(H_1, H_2) = \frac{2}{||\vec{\beta}||}$$

Replace l_2- by l_1-,l_∞-norm in the standard SVM prog.

Relational Data and Program Abstractions

[Kersting, Mladenov, Tokmakov AIJ´15, Mladenov, Heinrich, Kleinhans, Gonsio, Kersting DeLBP´16]

Lifted LP-SVM

```
var pred/1;        #predicted label for unlabeled instances
var slack/1;       #the slacks
var coslack/2;      #sl
var weight/1;      #the
var b/0;           #the
var r/0;           #marg

slack   = sum{label(I)} slack(I);
coslack = sum{cite(I1,I2),label(I1),query(I2)} slack(I1,I2)
                                                ck(I1,I2)

#find        gest margin. here the C's encode trade-off parameters
minimize: -r + C(1) * slack + C(2) * coslack;
```

Logically parameterized LP variable (set of ground LP variables)

Logically parameterized LP objective

Write down the LP-SVM in „paper form“.
The machine compiles it into solver form.

reloop

http://www-ai.cs.uni-dortmund.de/weblab/static/RLP/html/

RELOOP: A Toolkit for Relational Convex Optimization

Embedded within Python s.t. loops and rules can be used

```
#examples should be on the correct side of the hyperplane
subject to forall {I in label(I)}:
    label(I)*(innerProd(I) + b) + slack(I) >= r;
#weights are between -1 and 1
subject to for
subject to : r
subject to forall {I in label(I)}: slack(I) >= 0;       #slacks are positive
```

Logically parameterized LP constraint

But wait, publications are citing each other. OMG, I have to use graph kernels!

REALLY?

Relational Data and Program Abstractions

[Kersting, Mladenov, Tokmakov AIJ´15, Mladenov, Heinrich, Kleinhans, Gonsio, Kersting DeLBP´16]

Lifted LP-SVM

```
var pred/1;        #predicted label for unlabeled instances
var slack/1;       #the slacks
var coslack/2;       #slack between neighboring instances
var weight/1;      #the slope of the hyperplane
var b/0;           #the intercept of the hyperplane
var r/0;           #margin

slack   = sum{label(I)} slack(I);
coslack = sum{cite(I1,I2),label(I1),query(I2)} slack(I1,I2)
```

Logical query defines scope of abstract constraint

```
#find the ... ere the C's encode trade-off parameters
minimize: -r + ... + C(2) * coslack;
```

Citing papers share topics

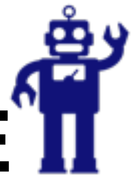
Collective constraints

```
subject to forall {I in q...)}: pred(I) = innerProd(I) + b;
#related instances should hav... the same labels.
subject to forall {I1, I2 in cite(I1, I2), label(I1), query(I2)}:
  label(I1) * pred(I2) +  slack(I1, I2) >= r;
#the symmetric case
subject to forall {I1, I2 in cite(I1, I2), label(I2), query(I1)}:
  label(I2) * pred(I1) + slack(I1, I2) >= r;
```

No kernel, the structure is expressed within the constraints!

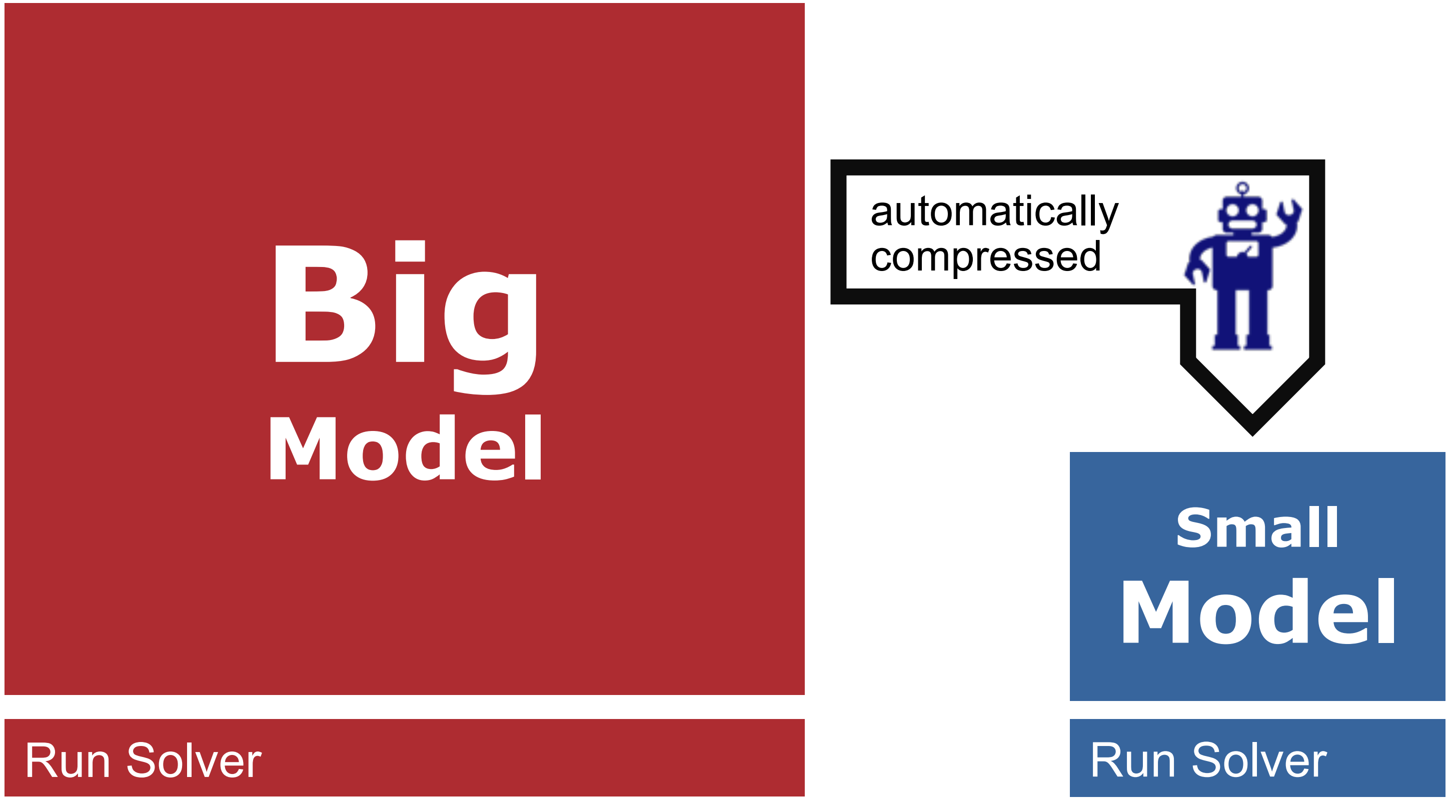
```
subject to forall {I in label(I)}:
    label(I)*(innerProd(I) + b) + slack(I) >= r;
#weights are between -1 and 1
subject to forall {J in attribute(_, J)}: -1 <= weight(J) <= 1;
subject to : r >= 0;          #the margin is positive
subject to forall {I in label(I)}: slack(I) >= 0;    #slacks are positive
```

OK, we have now a high-level, declarative language for mathematical programming.

HOW CAN THE MACHINE  NOW HELP TO REDUCE THE SOLVER COSTS?

Lifted Mathematical Programming
Exploiting computational symmetries

[Mladenov, Ahmadi, Kersting AISTATS´12, Grohe, Kersting, Mladenov, Selman ESA´14, Kersting, Mladenov, Tokmatov AIJ´17]

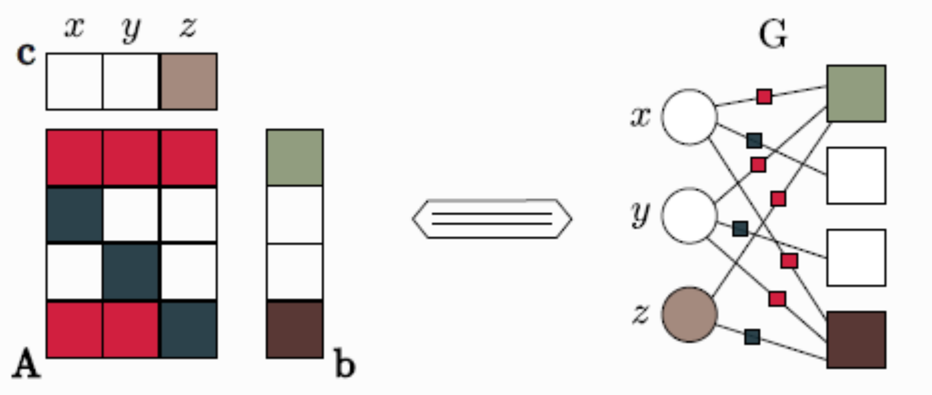


Lifted Mathematical Programming
Exploiting computational symmetries

[Mladenov, Ahmadi, Kersting AISTATS´12, Grohe, Kersting, Mladenov, Selman ESA´14, Kersting, Mladenov, Tokmatov AIJ´17]

$$\max_{[x,y,z]^T \in \mathbb{R}^3} \quad 0x + 0y + 1z$$

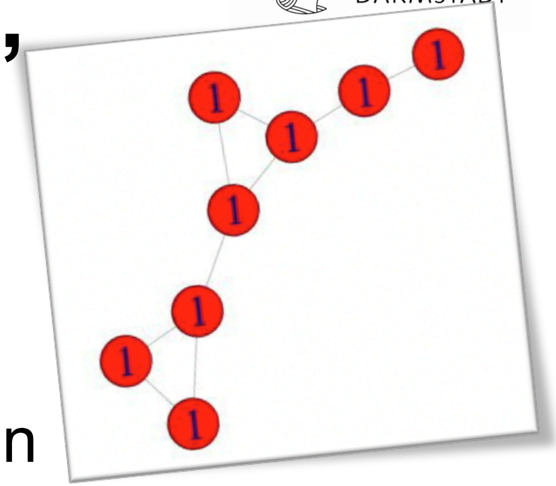
$$\text{s.t.} \quad \begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \leq \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$



View the mathematical program as a colored graph

Reduce the MP by running Weisfeiler-Lehman on the MP-Graph

Weisfeiler-Lehman (WL) aka "naive vertex classification"



Basic subroutine for GI testing

Computes LP-relaxations of GA-ILP, **fractional automorphisms**

Quasi-linear running time O((n+m)log(n)) when using asynchronous updates [Berkholz, Bonsma, Grohe ESA´13]

Part of graph tool SAUCY [See e.g. Darga, Sakallah, Markov DAC´08]

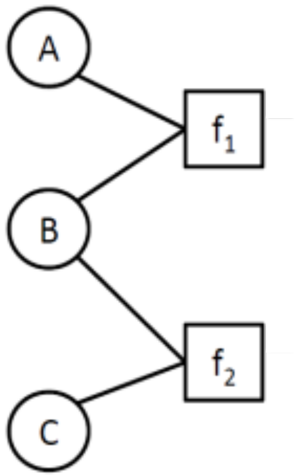
Has lead to highly performant graph kernels [Shervashidze, Schweitzer, van Leeuwen, Mehlhorn, Borgwardt JMLR 12:2539-2561 ´11]

Can be extended to weighted graphs/real-valued matrices [Grohe, Kersting, Mladenov, Selman ESA´14]

Actually a Frank-Wolfe optimizer and can be viewed as recursive spectral clustering [Kersting, Mladenov, Garnett, Grohe AAAI´14]

Compression: Coloring the graph

[Kersting, Ahmadi, Natarajan UAI’09; Ahmadi, Kersting, Mladenov, Natarajan MLJ’13, Mladenov, Ahmadi, Kersting AISTATS´12, Grohe, Kersting, Mladenov, Selman ESA´14, Kersting, Mladenov, Tokmatov AIJ´17]

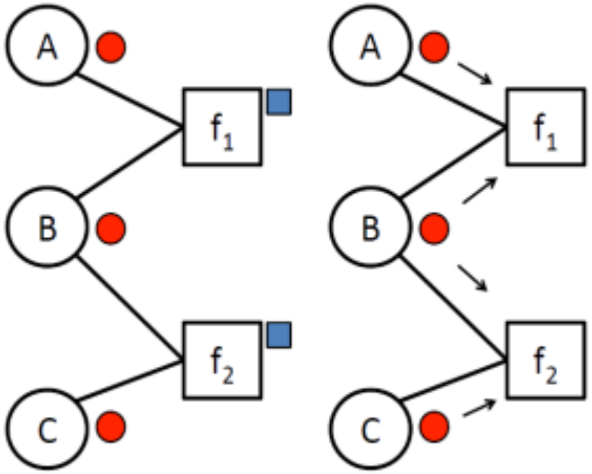


Color nodes initially with the same color, say red

Color factors distinctively according to their equivalences. For instance, assuming f_1 and f_2 to be identical and B appears at the second position within both, say **blue**

Compression: Pass colors around

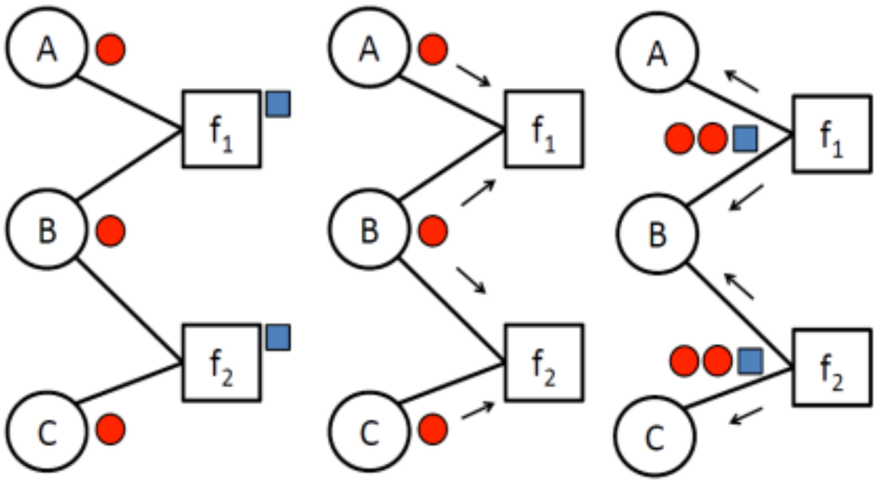
[Kersting, Ahmadi, Natarajan UAI'09; Ahmadi, Kersting, Mladenov, Natarajan MLJ'13, Mladenov, Ahmadi, Kersting AISTATS´12, Grohe, Kersting, Mladenov, Selman ESA´14, Kersting, Mladenov, Tokmatov AIJ´17]



1. Each factor collects the colors of its neighboring nodes

Compression: Pass colors around

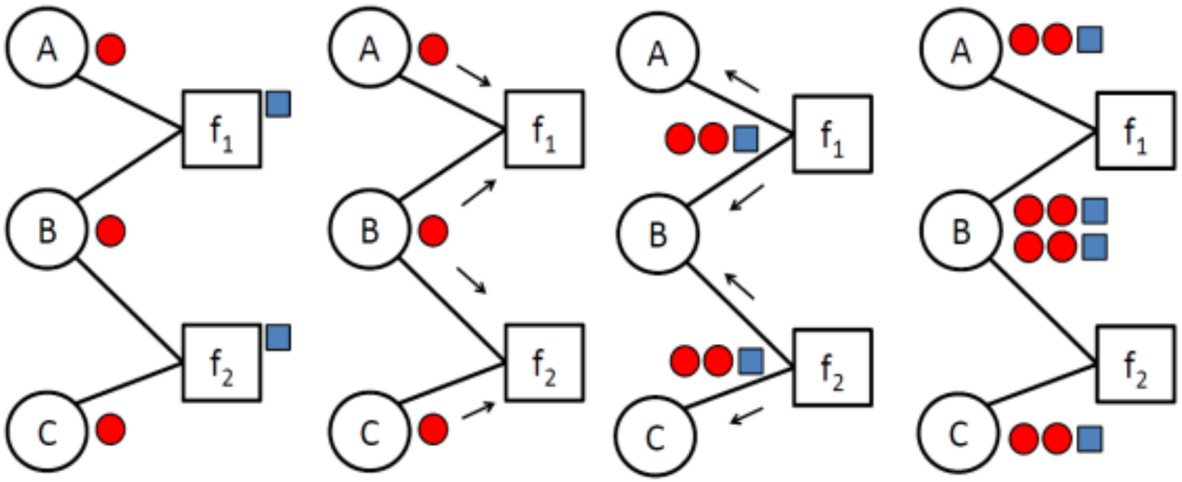
[Kersting, Ahmadi, Natarajan UAI'09; Ahmadi, Kersting, Mladenov, Natarajan MLJ'13, Mladenov, Ahmadi, Kersting AISTATS´12, Grohe, Kersting, Mladenov, Selman ESA´14, Kersting, Mladenov, Tokmatov AIJ´17]



1. Each factor collects the colors of its neighboring nodes
2. Each factor „signs" its color signature with its own color

Compression: Pass colors around

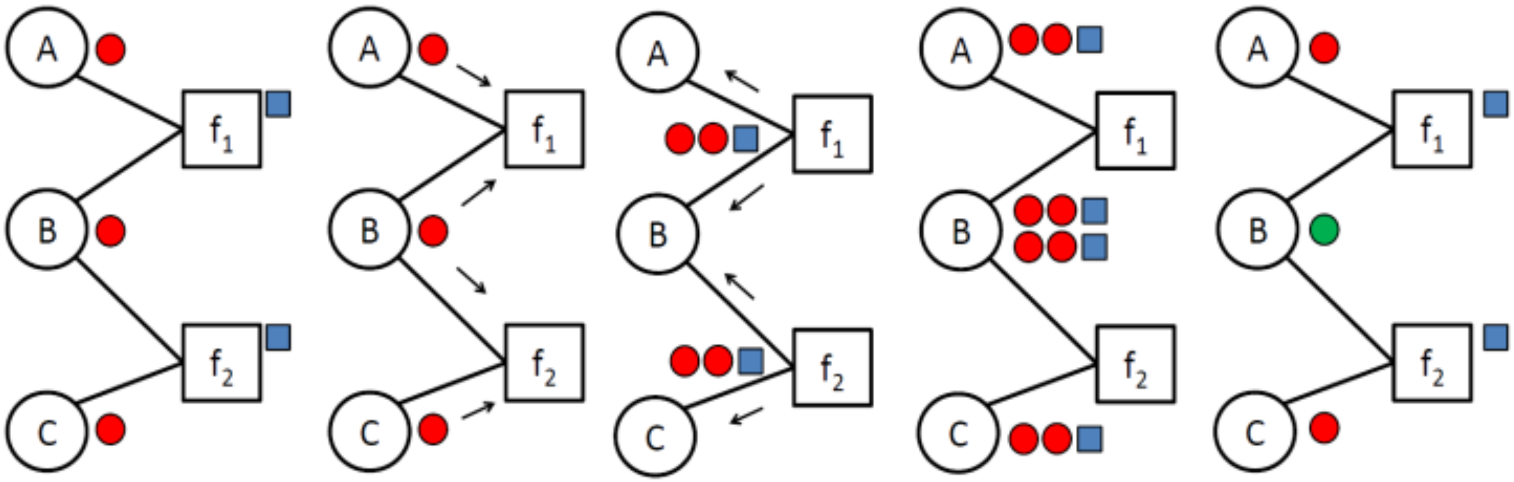
[Kersting, Ahmadi, Natarajan UAI'09; Ahmadi, Kersting, Mladenov, Natarajan MLJ'13, Mladenov, Ahmadi, Kersting AISTATS´12, Grohe, Kersting, Mladenov, Selman ESA´14, Kersting, Mladenov, Tokmatov AIJ´17]



1. Each factor collects the colors of its neighboring nodes
2. Each factor „signs“ its color signature with its own color
3. Each node collects the signatures of its neighboring factors

Compression: Pass colors around

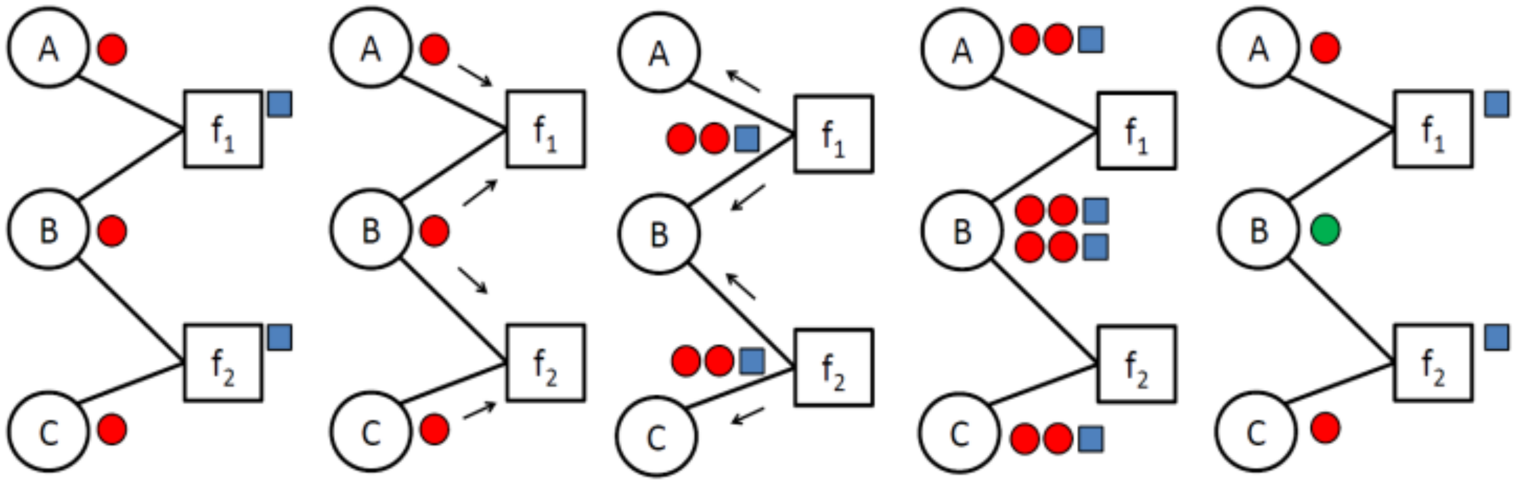
[Kersting, Ahmadi, Natarajan UAI'09; Ahmadi, Kersting, Mladenov, Natarajan MLJ'13, Mladenov, Ahmadi, Kersting AISTATS´12, Grohe, Kersting, Mladenov, Selman ESA´14, Kersting, Mladenov, Tokmatov AIJ´17]



1. Each factor collects the colors of its neighboring nodes
2. Each factor „signs“ its color signature with its own color
3. Each node collects the signatures of its neighboring factors
4. Nodes are recolored according to the collected signatures

Compression: Pass colors around

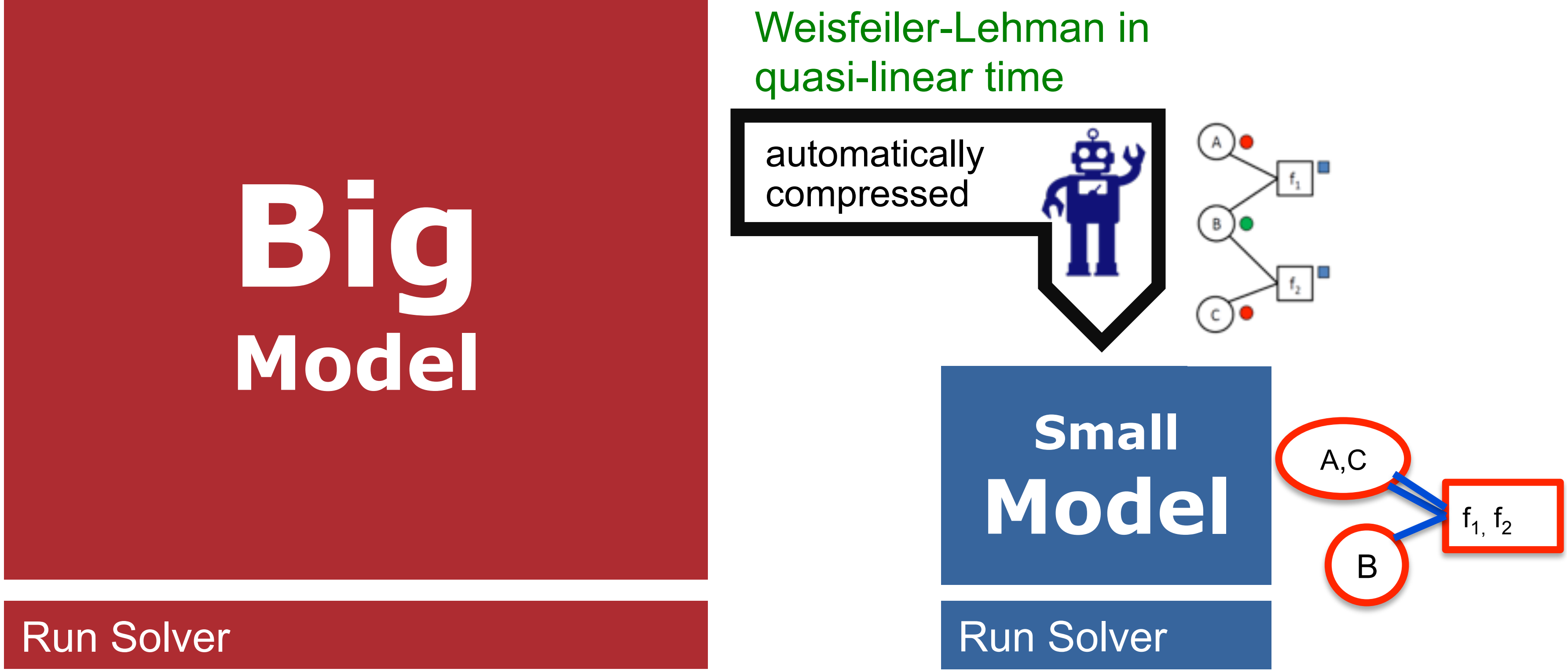
[Kersting, Ahmadi, Natarajan UAI'09; Ahmadi, Kersting, Mladenov, Natarajan MLJ'13, Mladenov, Ahmadi, Kersting AISTATS´12, Grohe, Kersting, Mladenov, Selman ESA´14, Kersting, Mladenov, Tokmatov AIJ´17]

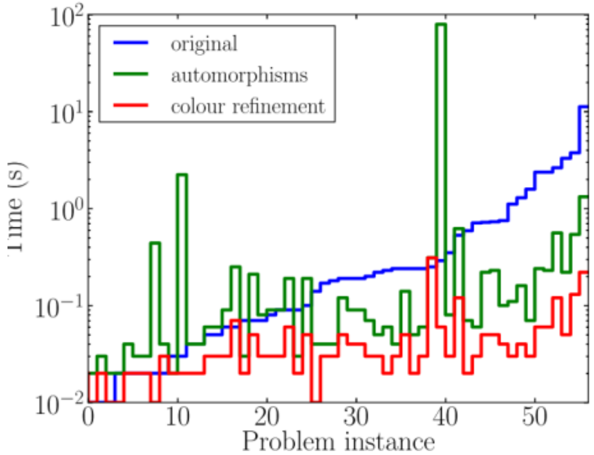


1. Each factor collects the colors of its neighboring nodes
2. Each factor „signs" its color signature with its own color
3. Each node collects the signatures of its neighboring factors
4. Nodes are recolored according to the collected signatures
5. If no new color is created stop, otherwise go back to 1

Lifted Mathematical Programming
Exploiting computational symmetries

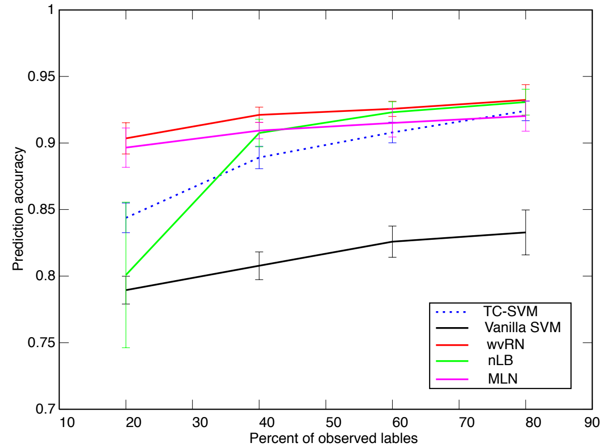
[Mladenov, Ahmadi, Kersting AISTATS´12, Grohe, Kersting, Mladenov, Selman ESA´14, Kersting, Mladenov, Tokmatov AIJ´17]





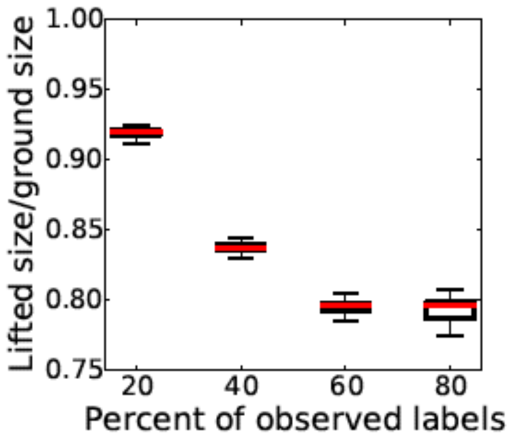
Margout‘s ILPs with symmetries (relaxed)

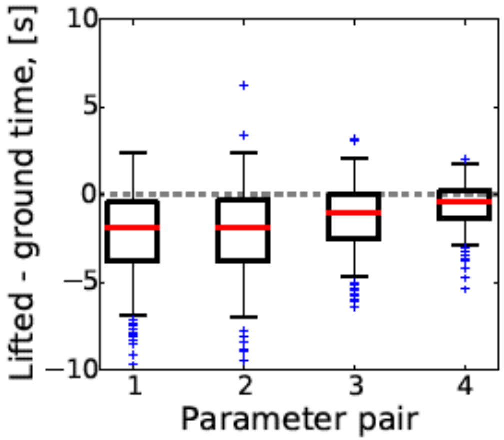
TECHNISCHE UNIVERSITÄT DARMSTADT

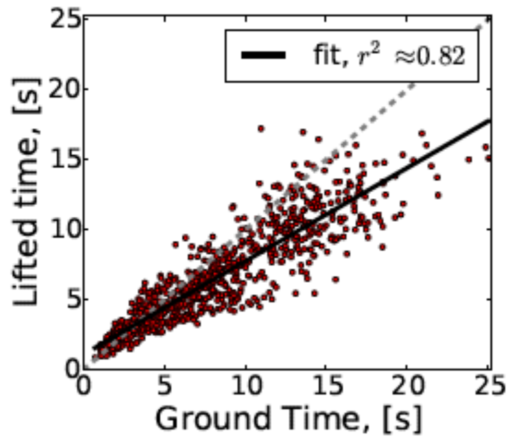


Collective Classification

Cora (most common vs. rest)





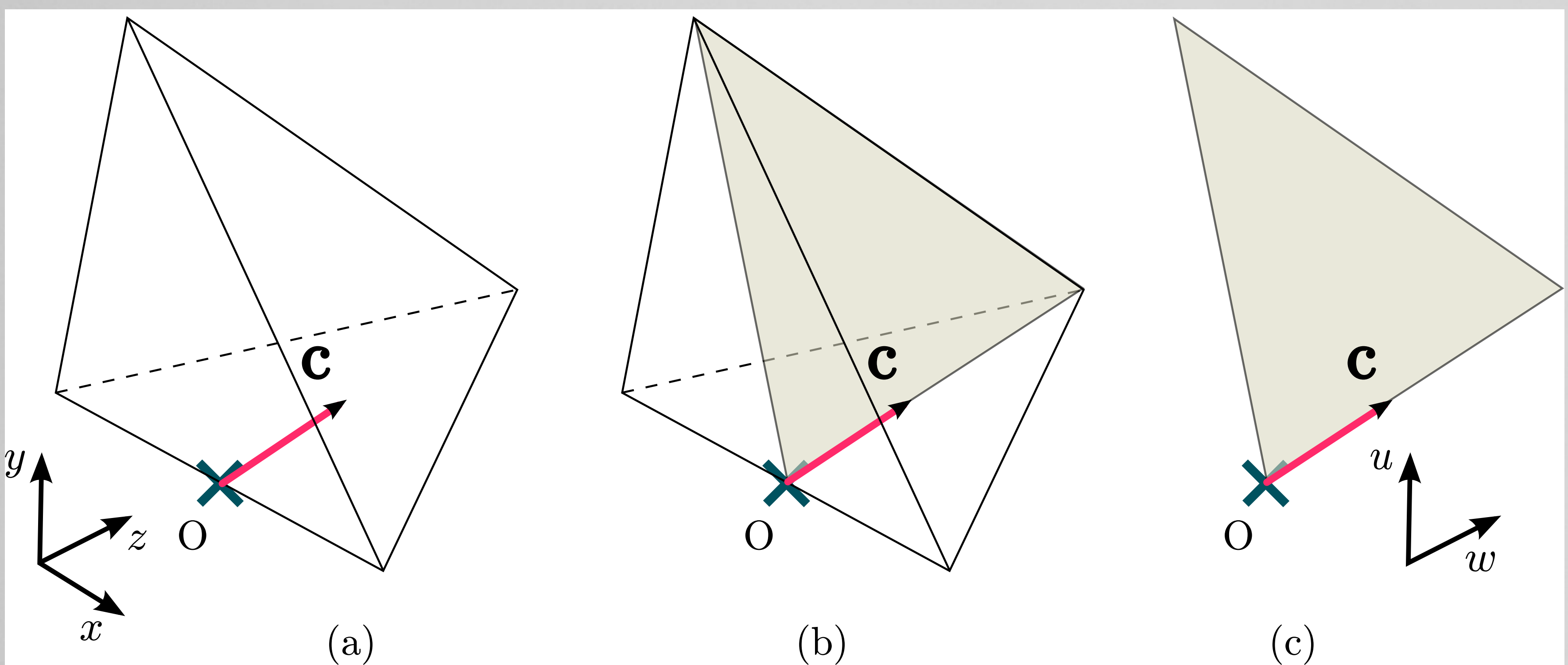


The more observed the more lifting
Faster end-to-end even in the light of Gurobi‘s fast pre-solving heuristics

[Boyd, Diaconis, Parrilo, Xiao: Internet Mathematics 2(1):31-71´05]

As also noted by Stephen Boyd

Dense vs. sparse is not enough, solvers need to be aware of symmetries



Feasible region of LP and the objective vectors

Span of the fractional auto-morpishm of the LP

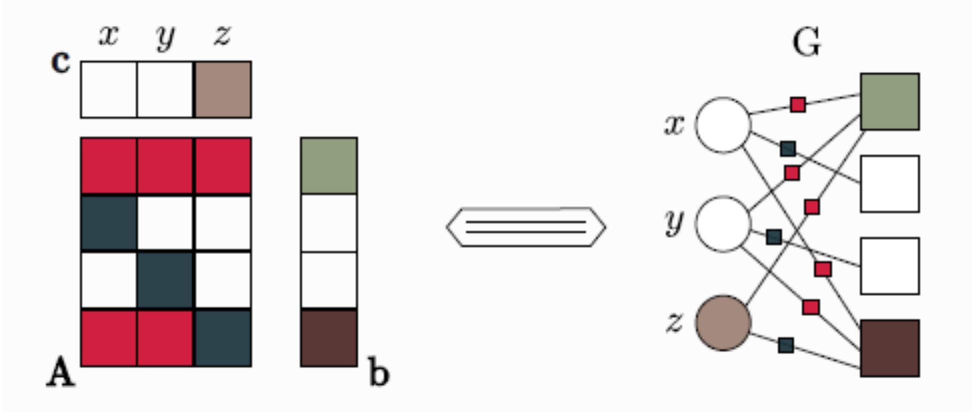
Projections of the feasible region onto the span of the fractional auto-morphism

Why does this work?

[Mladenov, Ahmadi, Kersting AISTATS´12, Grohe, Kersting, Mladenov, Selman ESA´14, Kersting, Mladenov, Tokmatov AIJ´15]

TECHNISCHE UNIVERSITÄT DARMSTADT

Compute Equitable Partition (EP) of the LP using WL



$$\mathcal{P} = \{\underbrace{P_1, \ldots, P_p}_{\text{Partition of LP variables}}; \underbrace{Q_1, \ldots, Q_q}_{\text{Partition of LP constraints}}\}$$

Partition of LP variables **Partition of LP constraints**

Intuitively, we group together variables resp. constraints that interact in the very same way in the LP.

Fractional Automorphisms of LPs

The EP induces a fractional automorphism of the coefficient matrix **A**

$$\mathbf{X}_Q \mathbf{A} = \mathbf{A} \mathbf{X}_P$$

where $\mathbf{X}_Q$ and $\mathbf{X}_p$ are doubly-stochastic matrixes (relaxed form of automorphism)

$$(\mathbf{X}_P)_{ij} = \begin{cases} 1/|P| & \text{if both vertices } i,\ j \text{ are in the same } P, \\ 0 & \text{otherwise.} \end{cases}$$

$$(\mathbf{X}_Q)_{ij} = \begin{cases} 1/|Q| & \text{if both vertices } i,\ j \text{ are in the same } Q, \\ 0 & \text{otherwise} \end{cases}$$

Fractional Automorphisms Preserve Solutions

If **x** is feasible, then $\mathbf{X}_p\mathbf{x}$ is feasible, too.

By induction, one can show that left-multiplying with a double-stochastic matrix preserves directions of inequalities; they are averagers. Hence,

$$\mathbf{Ax} \leq \mathbf{b} \Rightarrow \mathbf{X}_Q\mathbf{Ax} \leq \mathbf{X}_Q\mathbf{b} \Leftrightarrow \mathbf{AX}_P\mathbf{x} \leq \mathbf{b}$$

Fractional Automorphisms Preserve Solutions

TECHNISCHE UNIVERSITÄT DARMSTADT

If $\mathbf{x}^*$ is optimal, then $\mathbf{X}_p\mathbf{x}^*$ is optimal, too.

Since by construnction $\mathbf{c}^T\mathbf{X}_P = \mathbf{c}^T$ and hence $\mathbf{c}^T(\mathbf{X}_P\mathbf{x}) = \mathbf{c}^T\mathbf{x}$

What have we established so far?

Instead of considering the original LP

$$(\mathbf{A}, \mathbf{b}, \mathbf{c})$$

It is sufficient to consider

$$(\mathbf{A}\mathbf{X}_{P}, \mathbf{b}, \mathbf{X}_{P}{}^{T}\mathbf{c})$$

i.e. we “average” parts of the polytope.

But why is this dimensionality reduction?

Dimensionality Reduction

TECHNISCHE UNIVERSITÄT DARMSTADT

The doubly-stochastic matrix $\mathbf{X}_P$ can be written as

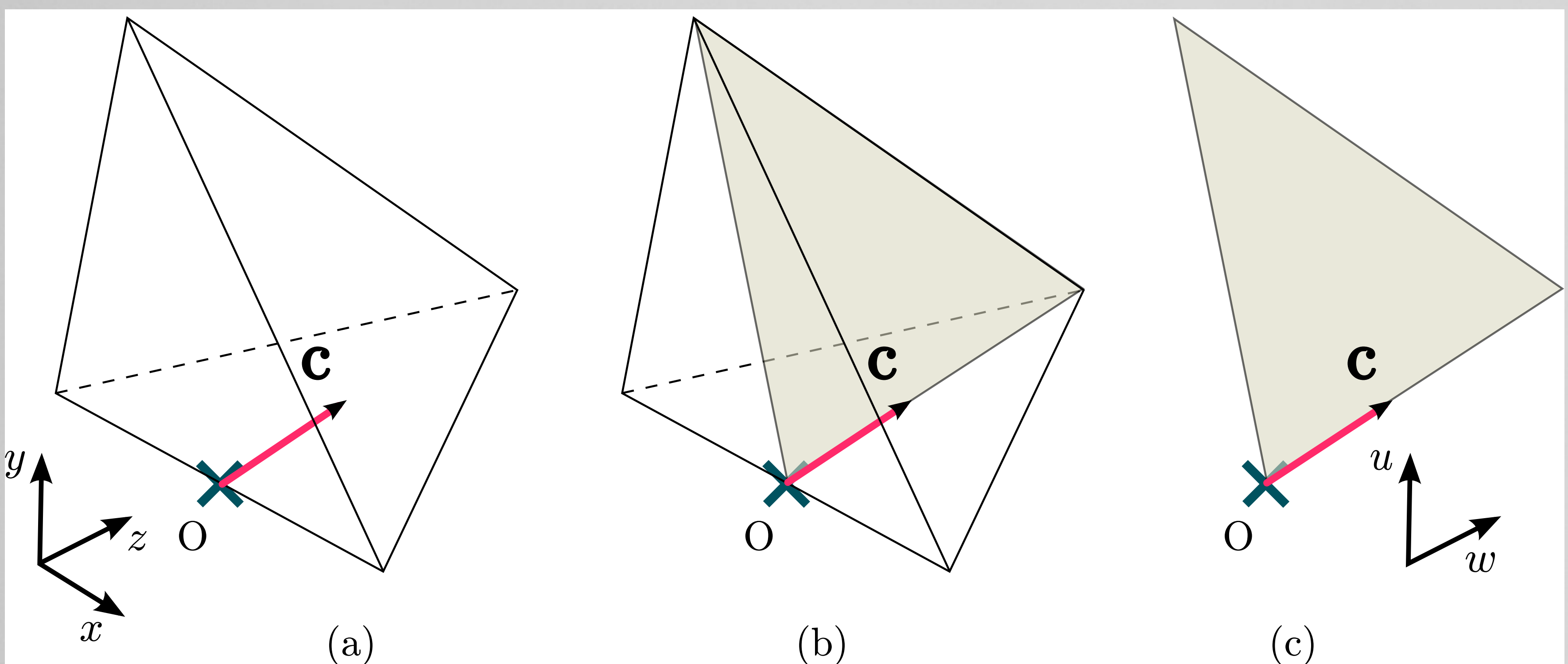
$$\mathbf{X}_P = \mathbf{B}\mathbf{B}^T$$

$$\mathbf{B}_{iP} = \begin{cases} \frac{1}{\sqrt{|P|}} & \text{if vertex } i \text{ belongs to part } P, \\ 0 & \text{otherwise.} \end{cases}$$

Since the column space of B is equivalent to the span of $\mathbf{X}_P$, it is actually sufficient to consider only

$$(\mathbf{A}\mathbf{B}_P, \mathbf{b}, \mathbf{B}_P^T\mathbf{c})$$

This is of reduced size, and actually we can also drop any constraint that becomes identical



Feasible region of LP and the objective vectors

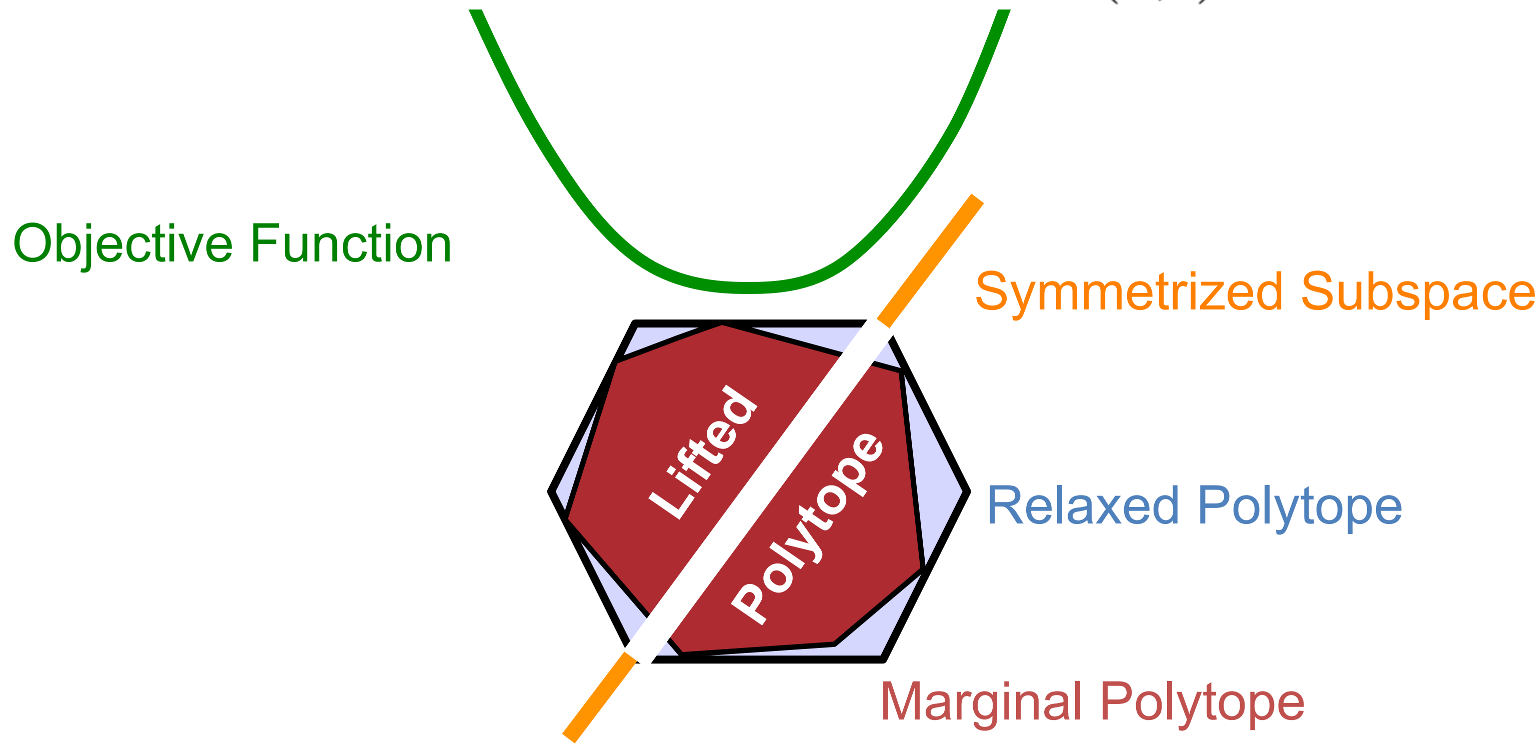
Span of the fractional auto-morpishm of the LP

Projections of the feasible region onto the span of the fractional auto-morphism

WL induces a Fractional Automorphism of the LP

Approximate probabilistic inference closely connected to LPs

$$\hat{\mathbf{x}} \in \arg\max_{\mathbf{x}\in\mathcal{X}^N} \left\{ \sum_{s\in V} \theta_s(x_s) + \sum_{(s,t)\in E} \theta_{st}(x_s, x_t) \right\}$$

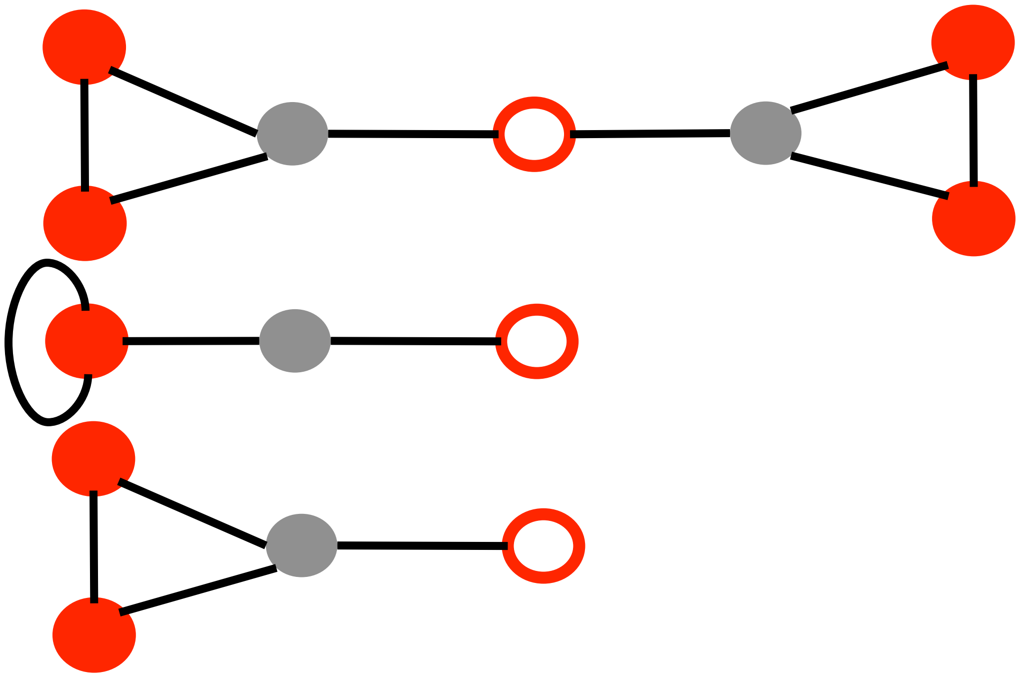


Lifted Optimization

TECHNISCHE UNIVERSITÄT DARMSTADT

Attention: For special-purpose solvers such as message-passing (coordinate descent,) for probabilistic inference we may have to reparameterize the lifted model

[Mladenov, Globerson, Kersting UAI 2014; Mladnov, Kersting UAI 2015]

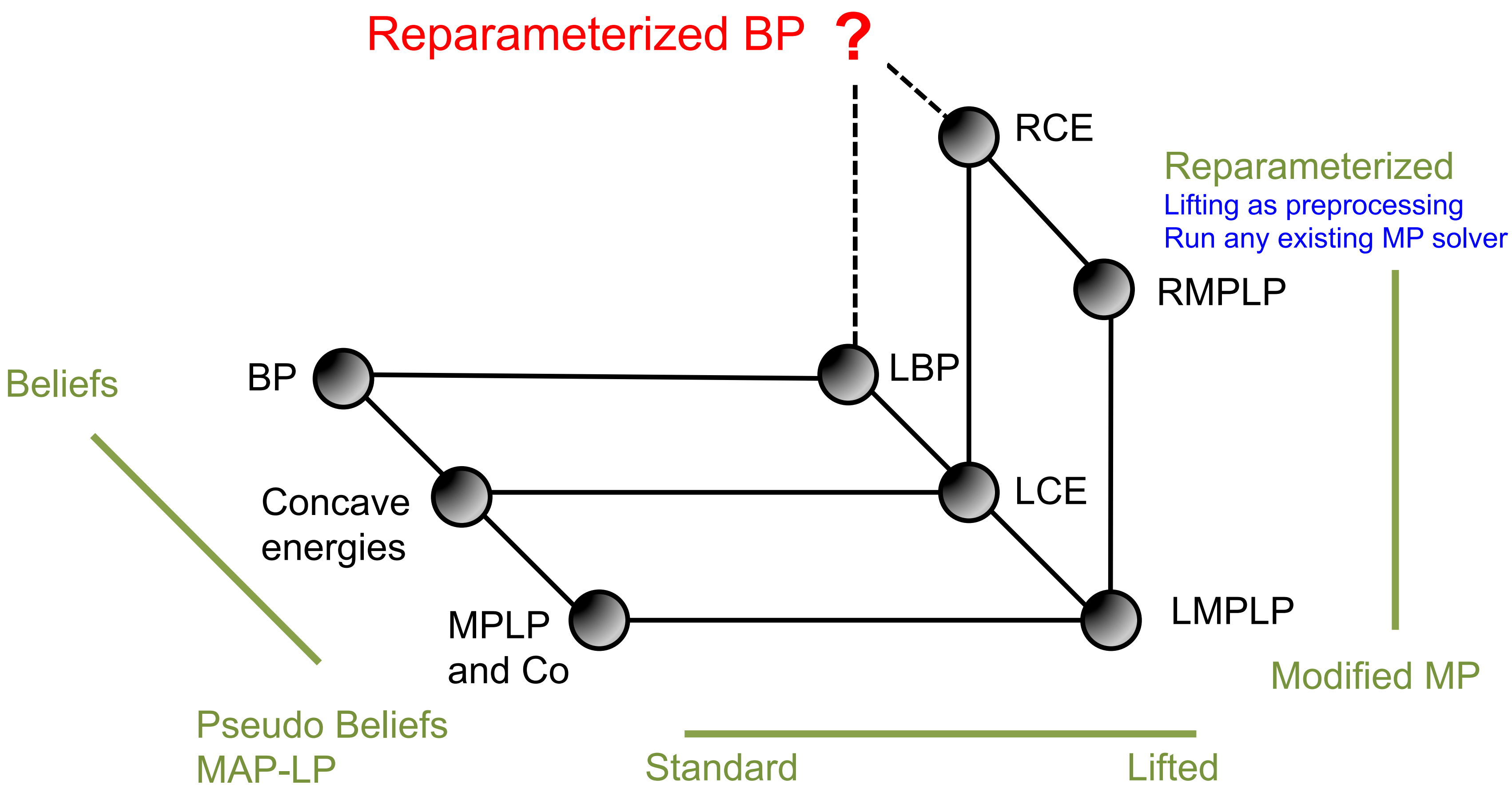


lifting

refine

Lifted probabilistic inference = Inference in a smaller, reparameterized model

[Mladenov, Globerson, Kersting UAI 2014; Mladnov, Kersting UAI 2015]



Holds also for Convex QPs

$$x^* = \arg\min_{x \in \mathcal{D}} J(x)$$
$$J(x) = x^T Q x + c^T x$$
$$\mathcal{D} = \{x : Ax \leqslant b\}$$

Mladenov, Kleinhans, Kersting AAAI´17

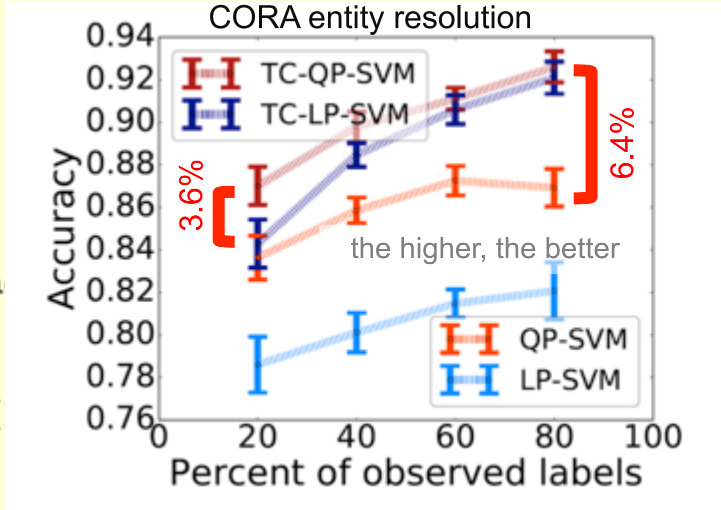
```
#query for the transductive constraint
linked(I1, I2) = label(I1) & query(I2) & (cite(I1, I2) | cite(I2, I1))

#inline definitions
slacks = sum{I in labeled(I)} slack(I);
coslacks = sum{I1, I2 in linked(I1, I2)} slack(I1,I2)

#QUADRATIC OBJECTIVE
minimize: sum{J in feature(I,J)} weight(J)**2 + c1 * sla

#labeled examples should be on the correct side
subject to forall {I in labeled(I)}: labeled(I)*predict(

#slacks are positive
```



On par with state-of-the-art by just four lines of code

```
#TRANSDUCTIVE PART
#cited instances should have the same labels.
subject to forall {I1, I2 in linked(I1, I2)}: labeled(I1) * predict(I2) >= 1 - slack(I1, I2);
subject to forall {I1, I2 in linked(I1, I2)}: coslack(I1, I2) >= 0; #coslacks are positive
```

Papers that cite each other should be on the same side of the hyperplane

A geometric interpretation

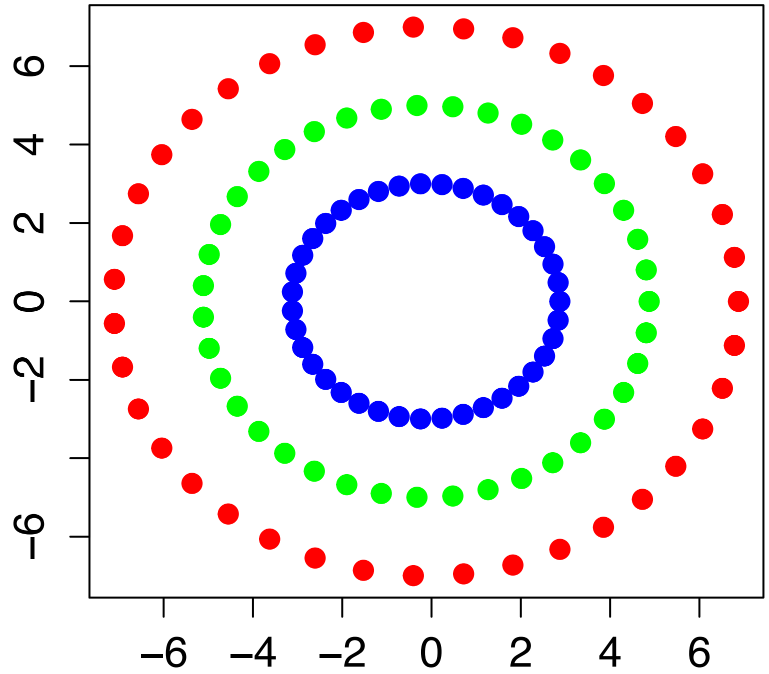
Mladenov, Kleinhans, Kersting AAAI´17

For QPs, a fractional automorphism is a rotation and scaling (of the semidefinite factors B of the Gram matrix)

fractional automorphism

automorphism

Relaxed by scaling



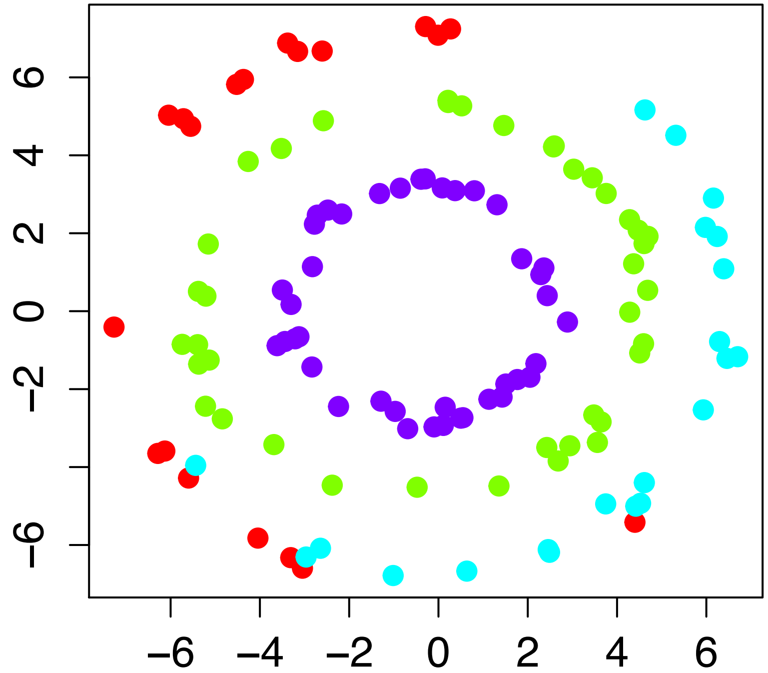
No symmetry-based ML?

Indeed, one may argue that the (rotational) automorphism group of most Euclidean datasets consists of the identity transformation alone: symmetries of a given dataset B can easily be destroyed by slightly perturbing the body.

No, we can have approximate fractional automorphisms (for SVMs)

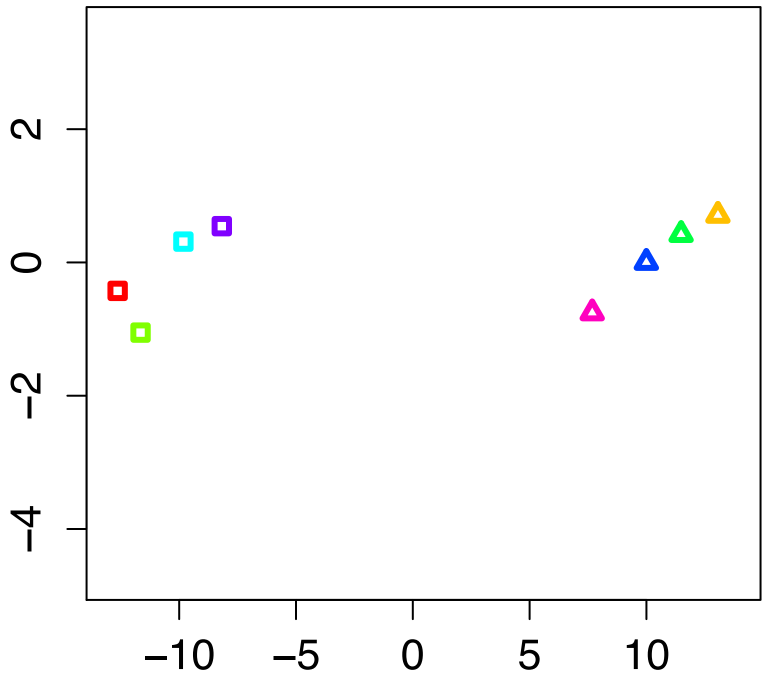
Mladenov, Kleinhans, Kersting AAAI´17

Whitening
+ K-means
of sorted
distance
vectors



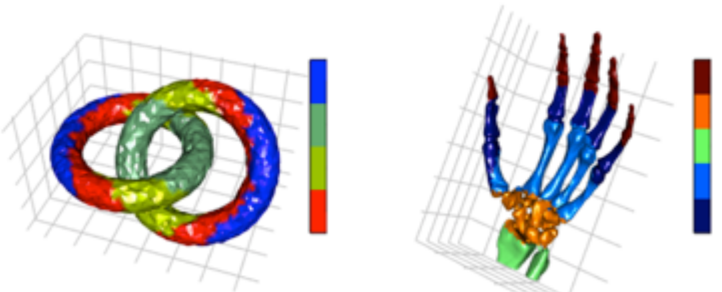
This provides a symmetry argument for known data reduction methods used for SVMs

Mladenov, Kleinhans, Kersting AAAI´17

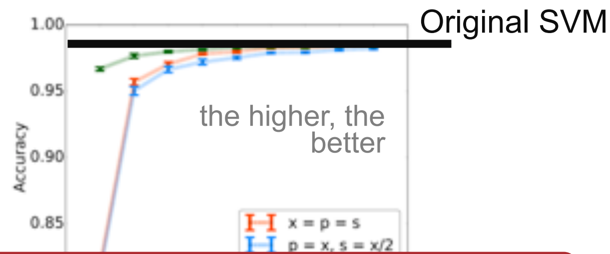


TECHNISCHE UNIVERSITÄT DARMSTADT

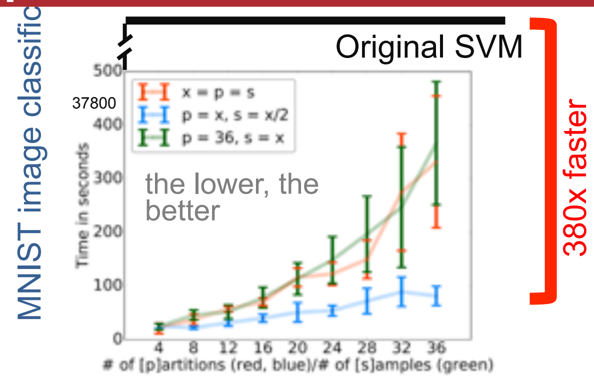
Approximately Lifted SVM: Cluster data points via K-means using sorted distance vectors. Solve SVM on cluster representatives only



Symmetry-based Data Programming: fractional autom. of label-preserving data transformations



Same should work for deep networks



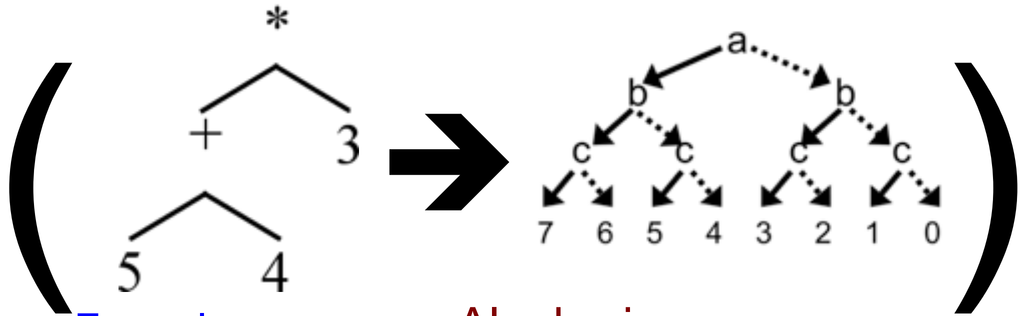
(d) Learning time: Translation

PAC-style generalization bound:

the approximately lifted SVM will very likely have a small expected error rate if it has a small empirical loss over the original dataset.

[Mladenov, Belle, Kersting AAAI´17]

And, there are other “-02”, “-03”, … flags, e.g symbolic-numerical interior point solvers

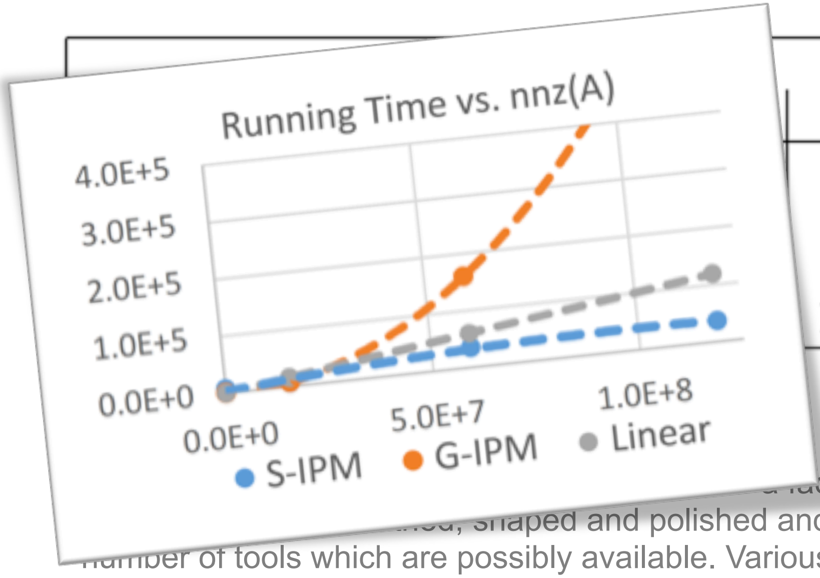


$$+\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Formulae parse trees

Algebraic Decision Diagrams

Matrix Free Optimization



All this opens the general machine learning toolbox for declarative machines:
feature selection, least-squares regression, label propagation, ranking, collaborative filtering, community detection, deep learning, …

…shaped and polished and possibly drilled before painting, each of which actions require a number of tools which are possibly available. Various painting and connection methods are represented, each having an effect on the quality of the job, and each requiring tools. Rewards (required quality) range from 0 to 10 and a discounting factor of 0. 9 was used used

Relations and (fractional) automorphisms are a natural foundation for

SYMMETRY-BASED ML AND DATA PROGRAMMING

[GENS, DOMINGOS NIPS 2014; RATNER ET AL. NIPS 2016]

- Learning (rich) representations is a central problem of machine learning
- (Fractional) symmetry / group theory provide a natural foundation for learning representations
- Symmetries = “unimportant” variants of data (graphs, relational structures, …)
- “Unimportant” variants programmed via declarative rules
- Let’s move beyond QPs: CSPs, SDPs, Deep Networks, …

Together with high-level languages

THINKING MACHINE LEARNING

- Shortens data science code to make ML techniques faster to write and easier to understand
- Reduces the level of expertise necessary to build ML applications
- Facilitates the construction of more sophisticated ML that incorporate rich domain knowledge and separate queries from underlying code
- Supports the construction of integrated ML machines thank think across a wide variety of domains and tool types
- Accelerates ML machines by exploiting language properties, compression, and compilation

Thanks for your attention!

TECHNISCHE UNIVERSITÄT DARMSTADT