Explainable Artificial Intelligence. Model Discovery with Constraint Programming

Antoni Ligęza, Paweł Jemioło, Weronika T. Adrian,
Mateusz Ślażyński, Marek Adrian, Krystian Jobczyk,
Krzysztof Kluza, Bernadetta Stachura-Terlecka, and Piotr Wiśniewski

AGH University of Science and Technology
KRaKEn Research Group
kraken.edu.pl
Kraków, Poland

ISMIS'2020 Industry Session
Graz, Austria – September 23-25, 2020

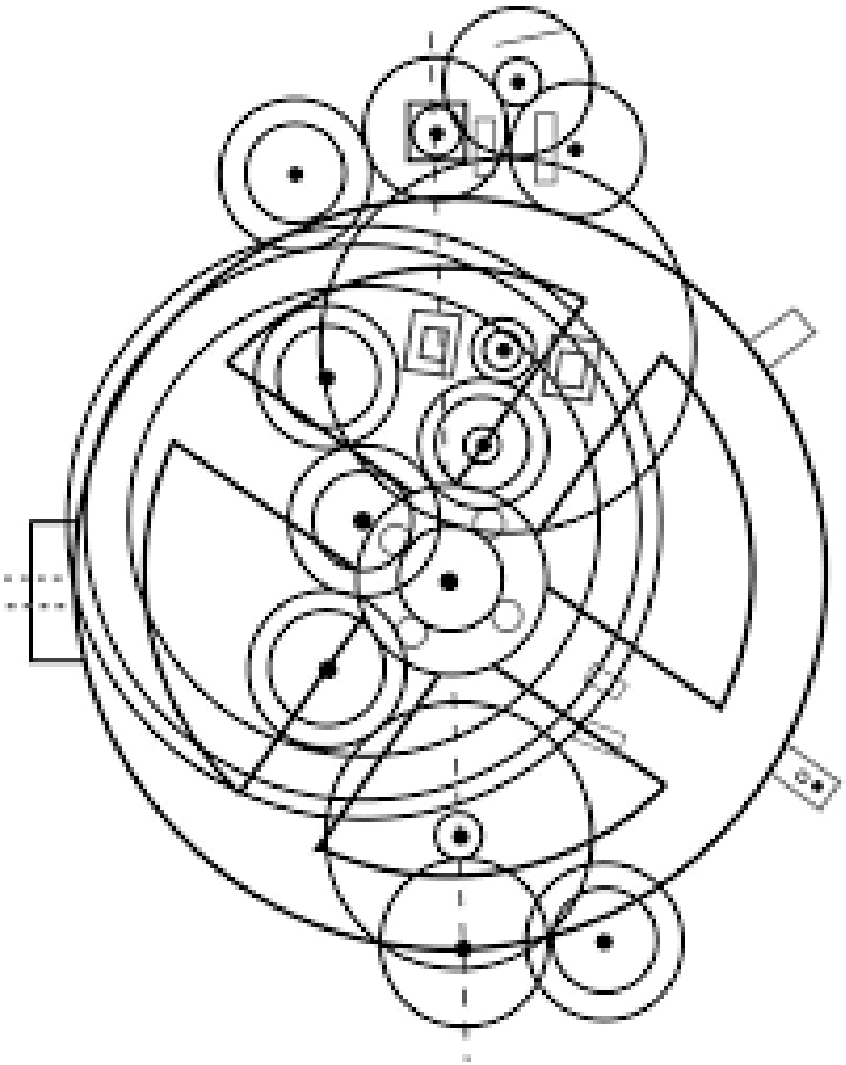
Outline

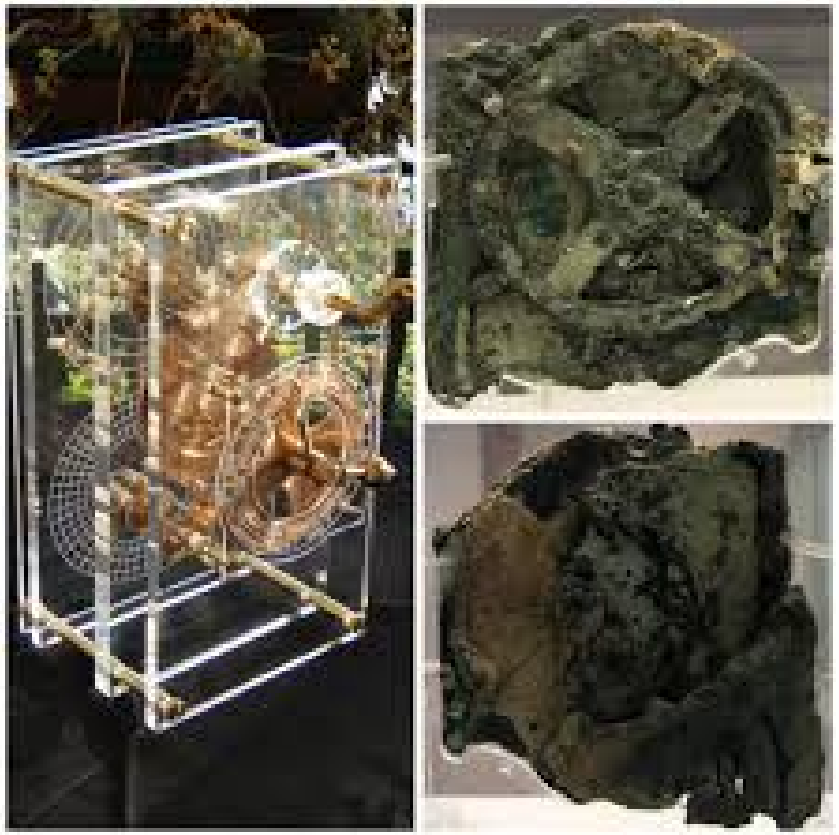
Presentation Outline

An Eternal Question: How Does it Work?

Figure: The Antikythera mechanism; recovered on May 17, 1901. The instrument has been variously dated to about 87 BC, or between 150 and 100 BC, or in 205 BC
`https://en.wikipedia.org/wiki/Antikythera_mechanism`

How Does it Work? Model-Based Reasoning





Components + Connections + Causality = Operation

XAI: Explainable Artificial Intelligence – WHY?

Why do we care about **Explainability in Artificial Intelligence?**

- readable and interpretable models,
- reduction of models (knowledge is more concise than data),
- well-defind domain of application,
- predictable behaviour,
- Reliable Artificial Intelligence,
- safe AI solutions,
- easy adaptation and modification,
- Trustable Artificial Intelligence.

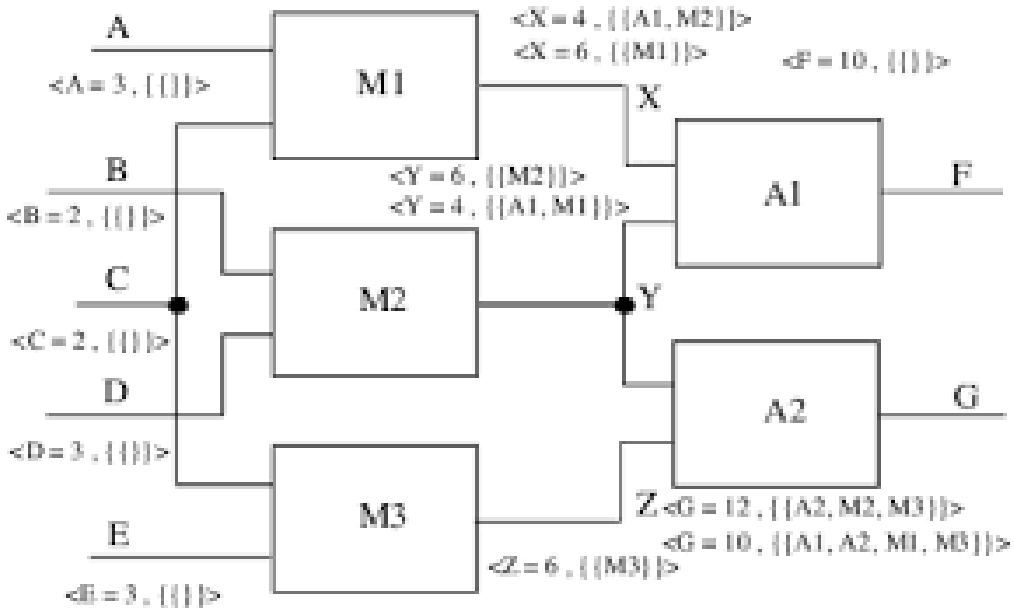
Presentation Outline

Model-Based Reasoning – A Perfect XAI Paradigm?

Discovering Causal Structure: Motivation:

- majority of ML models cover *shallow* knowledge only,
- most of them are on decision/classification type; no *functional* output,
- often: fuzzy/rough/probabilitic output,
- no investigation of the *guts* — what is inside?
- starting point: explanation of behaviour; diagnostic reasoning.

- variables, values, signals,
- components, links,
- internal structure,
- input – internal state – output,
- operation, constraints,
- functionality.



Level of Details in Structure Discovery

Values of variables:

- binary 0/1; true/false, ternary -/0/+, qualitative, integer numbers.

Connections:

- existence, direction, breaks, shortcuts, complex.

Components:

- parametric identification, selection one-of, function discovery.

Overall Structure:

- causality, structure - causal graph, logical and functional dependencies.

Towards Exact Model-Based Reasoning

CSP statement

- $X = \{X_1, X_2, \ldots, X_k\}$ — variables, $D = \{D_1, D_2, \ldots, D_k\}$ — their domains,
- $C = \{(S_i, R_i) : i = 1, 2, \ldots, n\}$ — constraints; S_i — scope; R_i — relation.

CSP solution

A solution to CSP: (X, D, C) — any assignment of values to variables of X:

$$\{X_1 = d_1, X_2 = d_2, \ldots, X_k = d_k\},$$

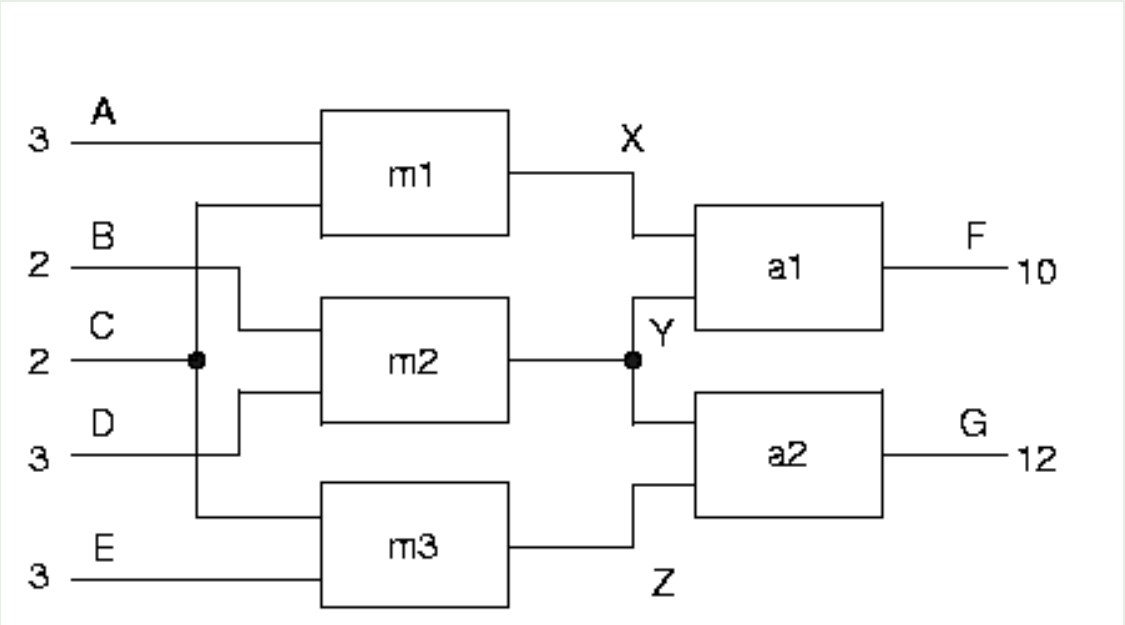
where $d_i \in D_i$, and for any constraint in $(S_i, R_i) \in C$, R_i is satisfied.

A CSP Example

Presentation Outline

Abductive Consistency-based Diagnosis

The multiplier-adder system to be diagnosed

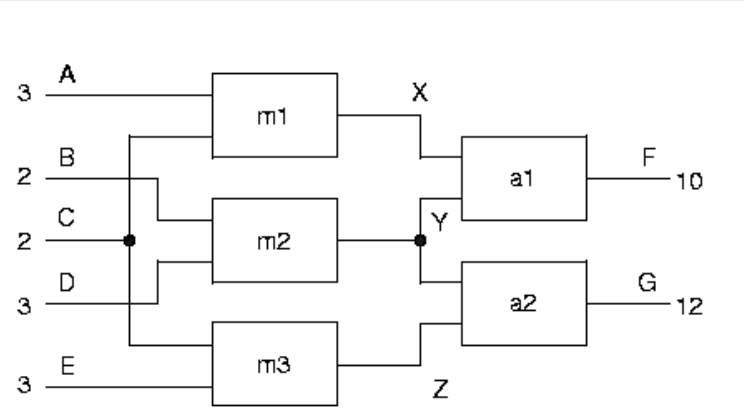


Consistency-Based Diagnosis

- MISBEHAVIOR: F=10 (should be 12); note that G=12 is O.K.
- ABDUCTION — CONFLICTS: $\{a1, m1, m2\}$,$\{a1, m1, a2, m3\}$,
- REPAIR — DIAGNOSES: $\{a1\}, \{m1\}, \{m2, m3\}, \{a2, m2\}$.

An example problem continued: case of {m1}

Model for diagnosis: m1



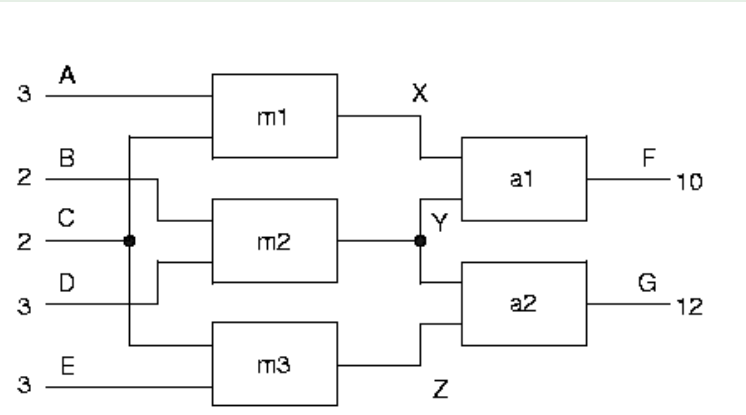

```
% K1/M1 = multiplier error

A * C * K1 #= X *M1,
B * D #= Y,
C * E #= Z,
X + Y #= F,
Y + Z #= G,
K1 #> 0, M1 #> 0.
```

Solution:

```
X=4, Y=6, Z=6, K1=2, M1=3
```

An example problem continued: case of {a1}

Model for diagnosis: a1



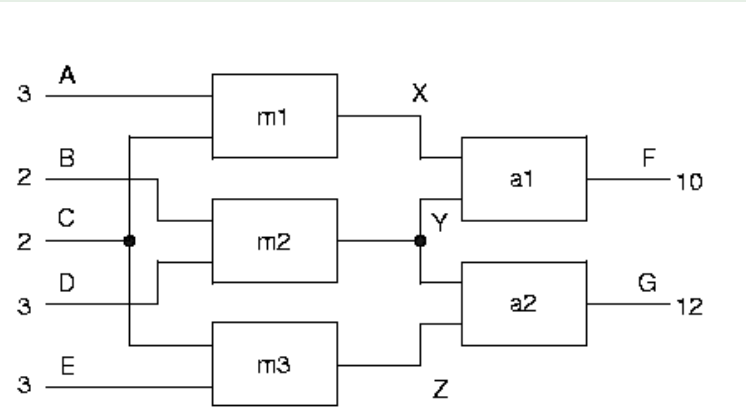

```
% A1 = addition  error
A * C #= X,
B * D #= Y,
C * E #= Z,
X + Y - A1 #= F,
Y + Z #= G.
```

Solution:

```
X=6, Y=6, Z=6, A1=2
```

An example problem continued: case of {a2,m2}

Model for diagnosis: $\{a2, m2\}$




```
% A2 = addition  error
% K2/M2 = multiplier error

A * C #= X,
B * D * K2 #= Y * M2,
C * E  #= Z,
X + Y  #= F,
Y + Z +A2 #= G,
K2 #> 0, M2 #> 0.
```

Solution:

```
X=6, Y=4, Z=6, K2=2, M2=3, A2=2
```

Constraints for parametric identification problems

A generic procedure:

1. take a minimal diagnosis for detailed examination;
2. for any component define the variable(s) to capture its misbehavior,
3. define the domains of the variables
4. and the constraints imposed on these variables for the analyzed case,
5. and constraints modeling the possible faulty work of all the components
6. run the constraint solver and analyze the results.

Results: detailed numerical explanations – numerical models of faulty behavior – for all the diagnoses.

Presentation Outline

Example: The 7-Segment Display

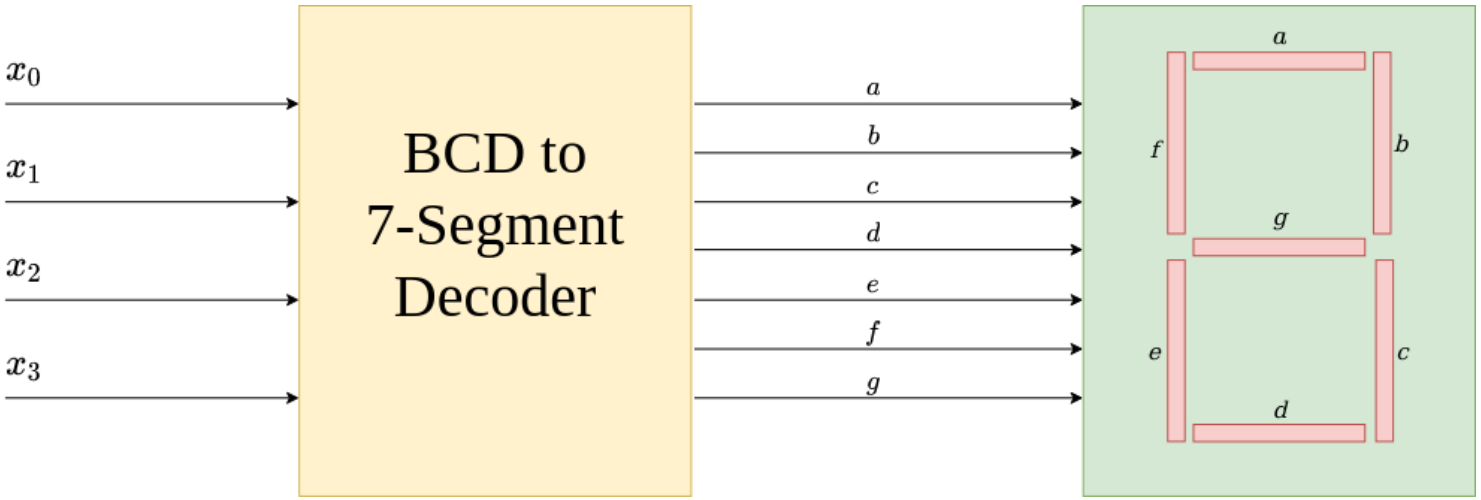


Figure: An example scheme presenting the input and output of the controller and connection to the 7-segment display

The inputs of the controller are the binary signals $\mathtt{x}_3, \mathtt{x}_2, \mathtt{x}_1, \mathtt{x}_0$. The digit $\mathtt{D} \in [0, 1, \ldots, 9]$ to be displayed is calculated as:

$$\mathtt{D} = \mathtt{x}_3 * 2^3 + \mathtt{x}_2 * 2^2 + \mathtt{x}_1 * 2 + \mathtt{x}_0$$

Input-Output Data

Inputs and Outputs Specification

```
INP = [|0,0,0,0|                    OUT = [|1,1,1,1,1,1,0|
       0,0,0,1|                           0,1,1,0,0,0,0|
       0,0,1,0|                           1,1,0,1,1,0,1|
       0,0,1,1|                           1,1,1,1,0,0,1|
       0,1,0,0|                           0,1,1,0,0,1,1|
       0,1,0,1|                           1,0,1,1,0,1,1|
       0,1,1,0|                           1,0,1,1,1,1,1|
       0,1,1,1|                           1,1,1,0,0,0,0|
       1,0,0,0|                           1,1,1,1,1,1,1|
       1,0,0,1|];                         1,1,1,1,0,1,1|];
```

The input table `INP` has 4 columns representing the values of x_3, x_2, x_1, x_0, respectively, and 10 rows, each of them representing a single digit of `0,1,2,3,4,5,6,7,8,9`. The output table `OUT` has 7 columns, each of them corresponding an appropriate display segment `a,b,c,d,e,f,g`, and exactly 10 rows, again each of them representing a single digit of `0,1,2,3,4,5,6,7,8,9` .

Assumptions for Model Discovery and CP

- we shall look for a structure in **Disjunctive Normal Form** (DNF),
- there will be two levels of components, the first one with AND gates (binary multiplication) and the second one with OR-gates (binary addition); the negations of input signals are assumed to be available,
- the number of the AND-gates should be minimal,
- the number of the OR-gates is 7; this is the consequence of the fact that there are 7 independent outputs a,b,c,d,e,f,g,
- the function of the controller must reconstruct the specified input-output behavior in an exact way.

The search areas open for the Constraint Programming:

- search for **connections**, from the input signals x_3, x_2, x_1, x_0 and their negations to the inputs of the AND-gates; note that the case of the lack of the connection (if the signal is not used) must also be represented,
- the **number of connections should be minimal**; this is so to keep the transparency and uniqueness of the solution,
- the number of the AND-gates should be as small as possible,
- for repeated AND-gates only one physical realization is necessary.

Encoding Search for Connections

The key element here is the array `W` encoding the existence (or not) of the input connections for each of the AND-gate. We applied the following encoding:

- `W[i,j] = 1` codes a connection of (positive occurrence) of x_j to an input of the `i`-th AND-gate,
- `W[i,j] = -1` codes a connection of negative occurrence (logical negation) of x_j to an input of the `i`-th AND-gate,
- `W[i,j] = 0` codes a lack of connection of of x_j as an input of the `i`-th AND-gate; in fact, the value of x_j is replaced with a constant `1` (a logical trick for assuring the correct operation of the AND-gate).

The corresponding function is a function taking as the input the appropriate x_j value and depending on the value of `W[i,j]` returns the exact or negated value of x_j, or just `1` in case of lack of the connection.

Encoding connections in MiniZinc

```
function var int: foo(var -1..1: w, var 0..1: x) =
                      if w==1 then x
                      elseif  w==-1
                      then abs(x-1)
                      else 1 endif;
```

Encoding AND and OR gates and the Goal

Encoding the AND-gates for signals x[3], x[2], x[1], x[0]

```
function var int: and4(array[1..4] of var  0..1:x,
                       array[1..4] of var -1..1:w) =
                       product(i in 1..4)(foo(w[i],x[i]));
```

Encoding the OR-gate output

```
function var int: sumM(array[1..M] of var 0..1: m) =
                  if sum(j in 1..M)(m[j]) > 0 then 1 else 0 endif;
```

The number of input connection to be minimized; M = number of inputs:

```
Y = sum(i in 1..M, j in 1..4)(abs(W[i,j]));
solve minimize Y;
```

For every input the appropriate output must be rendered:

```
N = 10;  % N= number of input cases to be covered
constraint forall(i in 1..N) (OUT[i,S] =
              sumM(j in 1..M) (and4(INP[i,..],W[j,..])));
```

Identification results for segment d

An example of identification results for segment d function.
Function d:
M = 5 (number of AND-gates discovered),
Y = 10 (minimal number of connections).

The weights of 5 AND gates

```
W    =  = [0,  1,  -1,  1,
           0, -1,   1,  0,
           1,  0,   0,  0,
           0,  0,   1, -1,
           0, -1,   0, -1]
Y = 10
```

The function is defined as:

$$d = x_2\bar{x}_1x_0 + \bar{x}_2x_1 + x_3 + x_1\bar{x}_0 + \bar{x}_2\bar{x}_0.$$

Summary of the Results

NoTS = the Number of Total Solutions (admissible solutions that are not necessary optimal ones),
TT = the total time of search in case we search for all solutions.

Table: Selected numerical results for the 7-segment display controller reconstruction

Segment	M (No. of AND-gates)	Y (No. of connections)	NoTS	TT [s]
a	4	6	4608	3.53
b	3	5	96	0.07
c	3	5	504	0.11
d	5	10	80640	83.56
e	2	4	12	0.00
f	4	7	8064	1.93
g	4	7	3840	2.31

The presented examples shows that Constraint Programming can be helpful for solving the problem of structure discovery in case of finite-values discrete systems and in presence of auxiliary, partial structural and functional knowledge.

Presentation Outline

Concluding Remarks

Conclusions:

- Exploring mutual interplay of Model-Based Reasoning, and Constraint Programming seems inspiring, especially in modeling Structure Discovery,
- Both Boolean and exact numerical models can be investigated,
- Structural knowledge can be discovered with Constraint Programming,
- Signals + Connections + Functions = System Model.

Further Issues and Work Directions:

- development of a parameterized set of block functions – potential components of larger systems,
- exploring imprecise cases with `solve minimize` of fitness evaluation,
- exploring methods for Causal Graphs discovery.
- typical ML-repository data is insufficient for causal/structural investigation,
- possible data extensions: causal, logical, functional and temporal aspects/knowledge,
- knowledge extensions: components, connections, causality, constraints,...

The End

Thank you for your attention!
Questions? Remarks? Constructive critics?

Powered by LaTeX