REASON WITH SAT FOR RULE LEARNING

Martina Seidl
Institute for Symbolic Artificial Intelligence
LIT AI Lab
Johannes Kepler University Linz

JOHANNES KEPLER UNIVERSITY LINZ

BilAI
Bilateral AI

SAT SOLVING

Propositional Logic: Syntax and Semantics

We consider formulas in negation normal form:

- negation occurs only directly in front of variables
- conjunction and disjunction are the only binary connectives
- special NNFs:
  - **literal**: variable or negated variable
  - **clause / cube**: disjunction of literals / conjunction of literals
  - **formula in CNF (conjunctive normal form)**: conjunction of clauses
  - **formula in DNF (disjunctive normal form)**: disjunction of cubes

Example

$$(\neg u \vee z) \wedge (y \vee u \vee \neg z) \wedge (x \vee \neg u \vee \neg z)$$

Propositional Logic: Syntax and Semantics

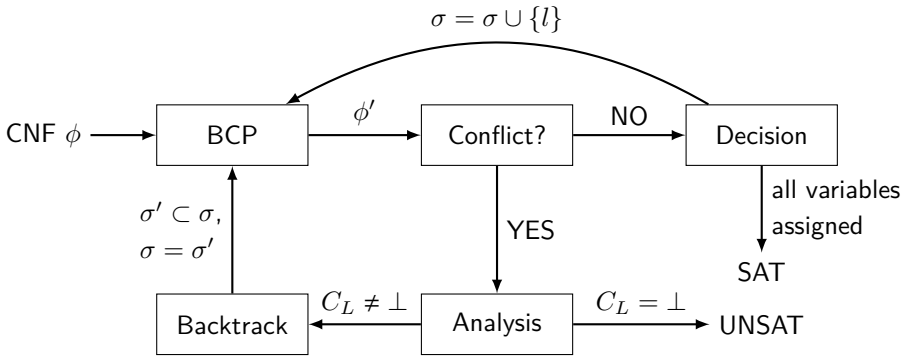
We consider formulas in negation normal form:

- negation occurs only directly in front of variables
- conjunction and disjunction are the only binary connectives
- special NNFs:
  - **literal**: variable or negated variable
  - **clause / cube**: disjunction of literals / conjunction of literals
  - **formula in CNF (conjunctive normal form)**: conjunction of clauses
  - **formula in DNF (disjunctive normal form)**: disjunction of cubes

Example

$$(\neg u \vee z) \wedge (y \vee u \vee \neg z) \wedge (x \vee \neg u \vee \neg z)$$

Semantics: A formula is satisfiable if there is some assignment under which it evaluates to true.


$\sigma = \sigma \cup \{l\}$
CNF ϕ
BCP
ϕ'
Conflict?
NO
Decision
all variables
assigned
SAT
YES
$\sigma' \subset \sigma$,
$\sigma = \sigma'$
Backtrack
$C_L \neq \bot$
Analysis
$C_L = \bot$
UNSAT

Machine Learning for SAT

- **Branching heuristics:** Predict which variable should be assigned next
- **Restart strategies:** Learn when restarting the search is beneficial
- **Clause management:** Predict which learned clauses to keep or delete
- **Algorithm selection:** Choose the most suitable solver for a given instance
- **Parameter tuning:** Automatically optimize solver hyperparameters

Sudoko: SAT Encoding

- dimension d of the Sudoku is given (example: 3)
- size n is defined as $n = d^2$ (example: 9)
- the board consists of n^2 cells (example: 81)
- each cell can take one of the values $1, \ldots, n$ (example: $1, \ldots, 9$)
- hints: predefined values of some cells

Sudoko: Constraints (1/2)

The encoding has $n^2 * n$ variables $f@r@c@v$ which is true iff field (r, c) has value v

The following constraints encode that all fields of the same row, column, subgrid have different values.

- each field has at least one value $1 \ldots 9$ (81 clauses)

$$\bigwedge_{c,r \in [1,\ldots,9]} (f@r@c@1 \vee \ldots \vee f@r@c@9)$$

- each field has at most one value $1 \ldots 9$ (2856 clauses)

$$\bigwedge_{c,r,v,v' \in [1,\ldots,9], v<v'} (\neg f@r@c@v \vee \neg f@r@c@v')$$

Sudoko: Constraints (2/2)

- every value occurs in every row

$$\bigwedge_{v,r \in [1,\ldots,9]} (f@r@1@v \vee \ldots \vee f@r@9@v)$$

- every value occurs in every column

$$\bigwedge_{v,c \in [1,\ldots,9]} (f@1@c@v \vee \ldots \vee f@1@c@v)$$

- every value occurs in every subgrid

$$\bigwedge_{c \in [1,\ldots,9]} (\bigwedge_{r,c \in \{1,4,7\}} (\bigvee_{r',c' \in \{0,1,2\}} f@r + r'@c + c'@v))$$

Can we use SAT solvers for learning how to play Sudoko?

(DEEP) RULE LEARNING

Towards SAT-Based Learning of NNF Networks

Paul Seip, Florian Beck, Johannes Fürnkranz
Institute for Application-Oriented Knowledge Processing
Johannes Kepler University Linz, Austria
{first.last}@jku.at

Clemens Hofstadler, Peter Pfeiffer, Martina Seidl
Institute for Symbolic Artificial Intelligence
Johannes Kepler University Linz, Austria
{first.last}@jku.at

Robert Peharz
Institute of Machine Learning and Neural Computation
TU Graz, Austria
Robert.peharz@tugraz.at

Stefan Szeider
Algorithms and Complexity Group
TU Wien, Austria
sz@ac.tuwien.ac.at

***Abstract*—To solve classification problems, we present a novel SAT-based framework for learning NNF networks directly from binary input-output data. In analogy to deep neural networks, negation normal form (NNF) networks exhibit a deep structure with learnable Boolean weights. By encoding this learning problem as a propositional satisfiability instance, our method leverages modern SAT solvers to construct logically correct, compact, and interpretable Boolean models of given datasets. Unlike statistical learners or heuristic rule induction algorithms, our approach guarantees logical fidelity on the training data. Evaluations on synthetic benchmarks and real-world datasets demonstrate that it outperforms state-of-the-art rule-based algorithms as well as a greedy NNF learning algorithm in terms of accuracy.**

to neural networks. We then express its behavior symbolically and use a SAT solver to learn the corresponding weights of the network. This allows us to find deep rule sets that exactly encode a dataset within the bounds of a given structure (if such a representation exists). Our results show that SAT solvers can be effectively harnessed for constructing logical structures from data. This opens new possibilities for combining symbolic reasoning with data-driven learning, and contributes to the broader goal of interpretable and verifiable AI systems [4].

II. Related Work

Rule Learning

Given a set of training examples, find a set of classification rules that can be used for prediction or classification of new instances.

(Fürnkranz et al., 2012)

Example of a rule:

$$\textbf{IF } \text{Age} > 60 \wedge \text{Blood Pressure} > 140 \quad \textbf{THEN } \text{High Risk}$$

Motivation

- Easy to interpret and explain
- Produces transparent decision models
- Can incorporate domain knowledge

Data and Samples

- We assume a tabular dataset $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$
- Each example $\mathbf{x}_i$ is characterized with d Boolean features $\mathcal{F} = \{f_1, \ldots, f_d\}$.
- 0 and 1 denote *false* and *true*.
- Each sample $\mathbf{x}_i$ is labeled by some Boolean value y_i.

Data and Samples

- We assume a tabular dataset $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$
- Each example $\mathbf{x}_i$ is characterized with d Boolean features $\mathcal{F} = \{f_1, \ldots, f_d\}$.
- 0 and 1 denote *false* and *true*.
- Each sample $\mathbf{x}_i$ is labeled by some Boolean value y_i.

Example

Example	a	b	c	d	e	f	g	y
$\mathbf{x}_1$	1	1	1	1	1	1	1	1
$\mathbf{x}_2$	1	0	1	0	0	1	1	1
$\mathbf{x}_3$	0	0	0	0	0	1	0	0
$\mathbf{x}_4$	1	0	1	0	0	0	1	0

Some Terminology

A **rule** $\mathbf{r}$ is of the form $B \to H$, where

- $B(\mathbf{r}) \subset \mathcal{F} \cup \overline{\mathcal{F}}$ is called the **body** of the rule,
- $H(\mathbf{r}) \in \{0, 1\}$ is its **head**, and
- $l(\mathbf{r}) = |B|$ is the **length** of the rule.

Classifiers can be obtained by forming a **rule set** R.

Rules in Prolog-Style Notation

- Flat rules

```
+ :- a, c, e.          + :- b, c, e.
+ :- a, c, f, g.       + :- b, c, f, g.
+ :- a, d, e.          + :- b, d, e.
+ :- a, d, f, g.       + :- b, d, f, g.
```

- Deep rules

```
+ :- h21, h22, h23.

h21 :- a.    h22 :- c.    h23 :- e.
h21 :- b.    h22 :- d.    h23 :- f, g.
```

(Deep) Rules as Propositional Formulas

- Formula in Negation Normal Form (NNF):

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

(Deep) Rules as Propositional Formulas

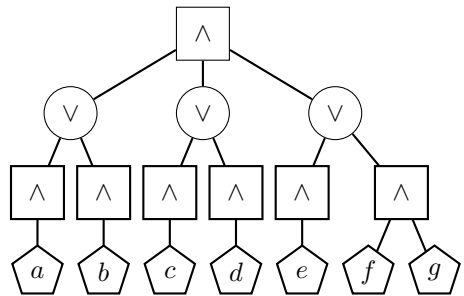
- Formula in Negation Normal Form (NNF):

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

- Equivalent formula in Disjunctive Normal Form (DNF):

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

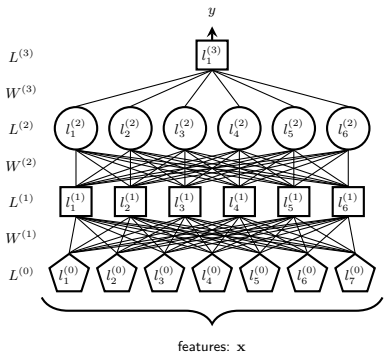
NNF:



NNF Network

An **NNF network** consists of multiple **layers** $\langle L^{(0)}, \ldots, L^{(h+1)} \rangle$, where

- $L^{(0)} = \{l_1^{(0)}, \ldots, l_d^{(0)}\}$ is the d-dimensional input layer, i.e., $|L^{(0)}| = d^{(0)} = d$
- $L^{(h+1)}$ is the 1-dimensional output layer $(d^{(h+1)} = 1)$
- h hidden layers with varying dimensions $d^{(i)}$.



SAT Encoding (1)

- **Encoding of Input Activations**

$$a_{i,k}^{(1)} = \bigwedge_{j=1}^{d} (f_{i,j} \vee \neg w_{j,k}^{(1)})$$

SAT Encoding (1)

- **Encoding of Input Activations**

$$a_{i,k}^{(1)} = \bigwedge_{j=1}^{d} (f_{i,j} \vee \neg w_{j,k}^{(1)})$$

- Example:

$$a_{1,k}^{(1)} = 1 \qquad \text{example 1}$$
$$a_{2,k}^{(1)} = (\neg w_{2,k}^{(1)} \wedge \neg w_{4,k}^{(1)} \wedge \neg w_{5,k}^{(1)}) \qquad \text{example 2}$$
$$a_{3,k}^{(1)} = (\neg w_{1,k}^{(1)} \wedge \ldots \wedge \neg w_{5,k}^{(1)} \wedge \neg w_{7,k}^{(1)}) \qquad \text{example 3}$$
$$a_{4,k}^{(1)} = (\neg w_{2,k}^{(1)} \wedge \neg w_{4,k}^{(1)} \wedge \neg w_{5,k}^{(1)}) \wedge \neg w_{6,k}^{(1)} \qquad \text{example 4}$$
$$\ldots$$

SAT Encoding (2)

- **Output Activation:**
  - number of hidden layers h is even:

$$y_i = \bigwedge_{j=1}^{m} (a_{i,j}^{(h)} \vee \neg w_j^{(h+1)})$$

  - number of hidden layers h is odd:

$$y_i = \bigvee_{j=1}^{m} (a_{i,j}^{(h)} \wedge w_j^{(h+1)})$$

SAT Encoding (2)

- **Output Activation:**
  - number of hidden layers h is even:

$$y_i = \bigwedge_{j=1}^{m} (a_{i,j}^{(h)} \vee \neg w_j^{(h+1)})$$

  - number of hidden layers h is odd:

$$y_i = \bigvee_{j=1}^{m} (a_{i,j}^{(h)} \wedge w_j^{(h+1)})$$

- Example: The output activation is a conjunctive layer, and for $i \in \{1, 2\}$, we get

$$1 = \bigwedge_{j=1}^{3} (a_{i,j}^{(2)} \vee \neg w_j^{(3)})$$

and for $i \in \{3, 4\}$, we get

$$0 = \bigwedge_{j=1}^{3} (a_{i,j}^{(2)} \vee \neg w_j^{(3)})$$

SAT Encoding (3)

- **Disjunctive Activation:** For a disjunctive layer l (l is even), each node k must be activated for example i if at least one of the nodes j in the conjunctive layer $l-1$ for which the connection is enabled ($w_{j,k}^{(l)} = 1$) is activated:

$$a_{i,k}^{(l)} = \bigvee_{j=1}^{m} (a_{i,j}^{(l-1)} \wedge w_{j,k}^{(l)}) \tag{1}$$

- Example: disjunctive layer with size three ($1 \leq k \leq 3$):

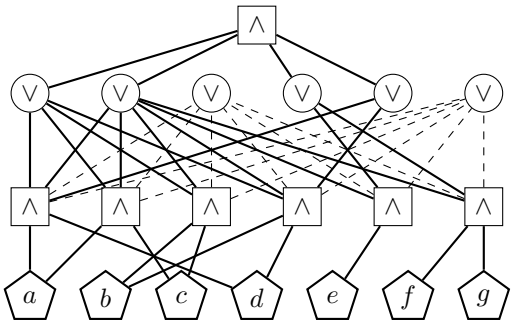
$$a_{i,k}^{(2)} = \bigvee_{j=1}^{6} (a_{i,j}^{(1)} \wedge w_{j,k}^{(2)})$$

SAT Encoding (4)

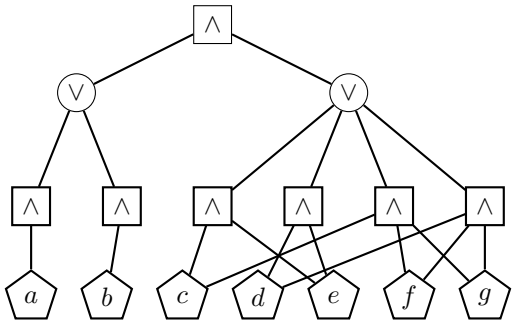
- **Conjunctive Activation:** For a conjunctive layer l ($l > 1$ is odd), each node k must be activated for example i if all of the nodes j in the conjunctive layer $l-1$ for which the connection is enabled ($w_{j,k}^{(l)} = 1$) is activated:

$$a_{i,k}^{(l)} = \bigwedge_{j=1}^{m} (a_{i,j}^{(l-1)} \vee \neg w_{j,k}^{(l)})$$

Found NNF Network with Two Hidden Layers



Another NNF Network



First Experiments

	NNF(5)		NNF(3)		NNF(1)		RIPPER	CART
	SAT	SHS	SAT	SHS	SAT	SHS		
Artificial Datasets								
Ø acc.	**0.9857**	0.9467	**0.9852**	0.9502	***0.9939***	0.9386	0.9591	0.9644
Ø rank	2.7	5.9	2.7	6.05	1.45	7	4.3	4.4
UCI Datasets								
Ø acc.	**0.9682**	0.9229	**0.9722**	0.9190	***0.9807***	0.9153	0.9290	0.9289
Ø rank	4.1429	4.4286	3.1429	5.7143	2.1429	5.8571	5.5714	4.2857

Ongoing Work

- minimizing the circuit
  - different structures
  - structure sharing
  - MaxSAT
- making approach faster
  - separate & conquer
  - incremental solving
- exploiting symmetries on
  - circuit
  - data
- other operators / structures

Why?

- ML with neural networks is not so good for
  - tabular data
  - permutations

Why?

- ML with neural networks is not so good for
  - tabular data
  - permutations
- Full control over the structure of the NNF network

Why?

- ML with neural networks is not so good for
  - tabular data
  - permutations
- Full control over the structure of the NNF network
- Combine different well explored fields
  - Rule learning
  - SAT
  - Circuit complexity
  - Knowledge Compilation

Why?

- ML with neural networks is not so good for
  - tabular data
  - permutations
- Full control over the structure of the NNF network
- Combine different well explored fields
  - Rule learning
  - SAT
  - Circuit complexity
  - Knowledge Compilation
- **Understandability and Explainability**

Related Directions

- **Boolean Functional Synthesis**
  Given a specification $\varphi(\mathbf{X}, \mathbf{Y})$, where
  - $\mathbf{X}$ are input variables,
  - $\mathbf{Y}$ are output variables,

  the goal is to construct Boolean functions

  $$f_i(\mathbf{X}) \quad (1 \leq i \leq |\mathbf{Y}|)$$

  such that

  $$\varphi\big(\mathbf{X}, f_1(\mathbf{X}), \ldots, f_{|\mathbf{Y}|}(\mathbf{X})\big)$$

  holds for every input assignment $\mathbf{X}$.

- **Quantified Boolean Formulas**

Collaborations

- Project: Bilateral AI
  - research module 3: Knowledge and Learning
  - research module 4: Reasoning
- Key researchers:
  - Johannes Fürnkranz
  - Martina Seidl
  - Robert Peharz
  - Stefan Szeider
  - Agata Ciabattoni
- Researchers: David Kattermann, Paul Seip, Peter Pfeiffer, Thomas Pammer, Mykhailo Barabash, Clemens Hofstadler, Florian Beck, ...