University of Udine

Learning from Answer Sets via ASP Encodings

LAUREA MAGISTRALE IN INFORMATICA

CANDIDATE
Roberto Borelli

SUPERVISOR
Prof. Agostino Dovier

July 17, 2025

Motivations

- Learning plays a central role in modern AI
- *Machine Learning* and *Deep Learning* achieve impressive performance, but often lack interpretability
- Symbolic methods, such as *Inductive Logic Programming* (ILP), aim to provide *Explainable AI*
- *Learning from Answer Sets* (LAS) is a state-of-the-art ILP framework successfully applied in domains like:
  - Explainable weather predictions
  - Explainable legal decisions
  - Explainable neural networks
  - Learning LTL formulas

Objective

- *ILASP* is a state-of-the-art ILP system based on the LAS framework
- In this work, we explore a new approach to LAS based on ASP encodings, offering competitive performance on several benchmarks
- The main idea is that a mature ASP solver can compute the solution of an encoded task more efficiently than ILASP can directly solve the corresponding LAS task

Outline

1 Introduction

2 Encodings

2.1 Exponential encoding

2.2 Polynomial encoding

3 Grounding

4 Implementation and Experiments

5 Conclusions

INTRODUCTION

$$\underbrace{H_1 \vee \ldots \vee H_k}_{\textit{head}} \leftarrow \underbrace{A_1 \wedge \cdots \wedge A_n \wedge \texttt{not}\ B_1 \wedge \cdots \wedge \texttt{not}\ B_m}_{\textit{body}}$$

"if the body holds, then the head must also hold"

Answer Sets Programming (ASP)

$$\underbrace{H_1 \vee \ldots \vee H_k}_{\textit{head}} \leftarrow \underbrace{A_1 \wedge \cdots \wedge A_n \wedge \texttt{not}\ B_1 \wedge \cdots \wedge \texttt{not}\ B_m}_{\textit{body}}$$

"if the body holds, then the head must also hold"

- $\mathsf{ASP}^{\mathsf{D}}$ Class of ASP programs with disjunctive heads
- $\mathsf{ASP}^{\mathsf{N}}$ Class of programs with $|\textit{head}| = 1$

Answer Sets Programming (ASP)

$$\underbrace{H_1 \vee \ldots \vee H_k}_{head} \leftarrow \underbrace{A_1 \wedge \cdots \wedge A_n \wedge \texttt{not}\ B_1 \wedge \cdots \wedge \texttt{not}\ B_m}_{body}$$

"if the body holds, then the head must also hold"

ASP$^{\mathsf{D}}$ Class of ASP programs with disjunctive heads

ASP$^{\mathsf{N}}$ Class of programs with $|head| = 1$

An *answer set* is a solution of an ASP program.

Answer Sets Programming (ASP)

$$\underbrace{H_1 \vee \ldots \vee H_k}_{head} \leftarrow \underbrace{A_1 \wedge \cdots \wedge A_n \wedge \texttt{not}\ B_1 \wedge \cdots \wedge \texttt{not}\ B_m}_{body}$$

"if the body holds, then the head must also hold"

ASP$^{\mathsf{D}}$ Class of ASP programs with disjunctive heads

ASP$^{\mathsf{N}}$ Class of programs with $|head| = 1$

An *answer set* is a solution of an ASP program.

Theorem

Deciding if a program P admits an answer set is:
NP-complete *if* $P \in$ ASP$^{\mathsf{N}}$ Σ_2^P-complete *if* $P \in$ ASP$^{\mathsf{D}}$

$$\textit{Learning Task } T : \langle B, S, E \rangle$$

- B Background Knowledge
- S Hypothesis Space
- E Examples

$$\textit{Learning Task } T : \langle B, S, E \rangle$$

- B Background Knowledge
- S Hypothesis Space
- E Examples

Solution

$H \subseteq S$ such that $B \cup H$ satisfies every example in E

$$\textit{Learning Task } T : \langle B, S, E^{+}, E^{-} \rangle$$

- B, S are $\mathsf{ASP}^{\mathsf{N}}$ programs

Learning from Answer Sets (LAS)

$$\textit{Learning Task } T : \langle B, S, E^{+}, E^{-} \rangle$$

- B, S are $\mathsf{ASP}^{\mathsf{N}}$ programs

A solution $H \subseteq S$ must satisfy:

- $\forall e \in E^{+} \exists A \in AS(B \cup H)$ A covers e brave
- $\forall e \in E^{-} \forall A \in AS(B \cup H)$ A does not cover e cautious

$$\textit{Learning Task } T : \langle B, S, E^+, E^- \rangle$$

- B, S are ASP^N programs

A solution $H \subseteq S$ must satisfy:

- $\forall e \in E^+ \exists A \in AS(B \cup H)$ A covers e brave
- $\forall e \in E^- \forall A \in AS(B \cup H)$ A does not cover e cautious

Theorem

Deciding if a task T admits a solution is Σ_2^P-complete

ILASP

- State-of-the art algorithm for ILP_{LAS}
- Iterative algorithm: at each step, an $\mathsf{ASP}^{\mathsf{N}}$ program is solved
- Many versions of ILASP: 1,2,2i,3,4

Our Contributions

- For ground tasks, we introduce single-shot ASP encodings
  - P_{exp} Exponential $\mathsf{ASP}^{\mathsf{N}}$ encoding
  - P_{dis} Linear $\mathsf{ASP}^{\mathsf{D}}$ encoding (if $B \cup S$ has no loops)
- We introduce a notion of grounding for LAS tasks
- We introduce LASCO, a LAS solver based on these ideas

ILASP: multiple calls to an ASP solver
LASCO: single call to an ASP solver

ENCODINGS

Exponential ASP$^{\mathsf{N}}$ encoding (P_{exp})

- Almost "direct" encoding of B and S
- Choice rule to encode the choice of $H \subseteq S$
- Enumeration of all the possible interpretations of $B \cup S$
- For a given H, we have rules that check if an interpretation I is also an answer set of $B \cup H$
- We force brave covering for E^+ and cautious covering for E^-

Exponential ASP$^{\mathsf{N}}$ encoding (P_{exp})

- Almost “direct” encoding of B and S
- Choice rule to encode the choice of $H \subseteq S$
- Enumeration of all the possible interpretations of $B \cup S$
- For a given H, we have rules that check if an interpretation I is also an answer set of $B \cup H$
- We force brave covering for E^+ and cautious covering for E^-

Problem

The number of interpretation of $B \cup S$ is exponential in the number of atoms

Polynomial ASP$^{\mathsf{D}}$ encoding (P_{dis})

$$P_{dis}(T) ::= P^{+}_{dis}(T) \cup P^{-}_{dis}(T)$$

- $P^{+}_{dis}(T)$ ensures brave covering of positive examples
- $P^{-}_{dis}(T)$ ensures cautious covering of negative examples

The program P^{-}_{dis} is built into two stages:

1. T is translated into a Q.B.F. formula $\phi(T)$
2. $\phi(T)$ is translated into an ASP$^{\mathsf{D}}$ program

From the task T we build the following Q.B.F. formula:

$$\phi(T) \coloneqq \exists H \, \forall I \;\; \underbrace{\mathsf{NNF}(\phi^*_{as}(P'(T)) \rightarrow \phi_{neg}(E^-))}_{\psi(T)}$$

From the task T we build the following Q.B.F. formula:

$$\phi(T) := \boxed{\exists H\ \forall I}\ \underbrace{\mathsf{NNF}(\phi^*_{as}(P'(T)) \rightarrow \phi_{neg}(E^-))}_{\psi(T)}$$

$\phi(T)$ is satisfied if:

There exists $H \subseteq S$ s.t. for every I interpretation of $B \cup H$ s.t.

From the task T we build the following Q.B.F. formula:

$$\phi(T) := \exists H \, \forall I \;\; \underbrace{\mathsf{NNF}(\boxed{\phi^*_{as}(P'(T))} \to \phi_{neg}(E^-))}_{\psi(T)}$$

$\phi(T)$ is satisfied if:

There exists $H \subseteq S$ s.t. for every I interpretation of $B \cup H$ s.t.
If I is an answer set of $B \cup H$

From the task T we build the following Q.B.F. formula:

$$\phi(T) := \exists H\, \forall I \;\; \underbrace{\mathsf{NNF}(\phi^*_{as}(P'(T)) \rightarrow \boxed{\phi_{neg}(E^-)})}_{\psi(T)}$$

$\phi(T)$ is satisfied if:

There exists $H \subseteq S$ s.t. for every I interpretation of $B \cup H$ s.t.
If I is an answer set of $B \cup H$, it must satisfy negative examples

From the formula $\phi(T)$ we build an ASP$^{\mathsf{D}}$ program:

$$\begin{aligned}
&\mathtt{h} \vee \mathtt{nh}. && \forall h \in H\\
&\mathtt{i} \vee \mathtt{ni}. \quad \mathtt{i} \leftarrow \mathtt{w}. \quad \mathtt{ni} \leftarrow \mathtt{w}. && \forall i \in I\\
&\mathtt{w} \leftarrow \mathtt{formula}_{\psi(T)}. &&\\
&\mathit{expansion}(\psi(T)) &&\\
&\mathtt{w} \leftarrow \mathtt{not}\ \mathtt{w}. &&
\end{aligned}$$

From the formula $\phi(T)$ we build an ASP$^{\mathsf{D}}$ program:

$$\mathtt{h} \vee \mathtt{nh}. \qquad \forall h \in H$$

$$\mathtt{i} \vee \mathtt{ni}. \quad \mathtt{i} \leftarrow \mathtt{w}. \quad \mathtt{ni} \leftarrow \mathtt{w}. \qquad \forall i \in I$$

$$\mathtt{w} \leftarrow \mathtt{formula}_{\psi(T)}.$$

$$\mathit{expansion}(\psi(T))$$

$$\boxed{\mathtt{w} \leftarrow \mathtt{not}\ \mathtt{w}.}$$

w must be included in a solution.

From the formula $\phi(T)$ we build an ASP$^{\mathsf{D}}$ program:

$$\texttt{h} \vee \texttt{nh}. \qquad \forall h \in H$$

$$\texttt{i} \vee \texttt{ni}. \quad \texttt{i} \leftarrow \texttt{w}. \quad \texttt{ni} \leftarrow \texttt{w}. \qquad \forall i \in I$$

$$\boxed{\texttt{w} \leftarrow \texttt{formula}_{\psi(T)}.}$$

$$\mathit{expansion}(\psi(T))$$

$$\texttt{w} \leftarrow \texttt{not w}.$$

To be included, w must be supported, hence $\psi(T)$ must be satisfied by the current interpretation.

From the formula $\phi(T)$ we build an ASP$^{\mathsf{D}}$ program:

$$\mathtt{h} \vee \mathtt{nh}. \qquad \forall h \in H$$

$$\mathtt{i} \vee \mathtt{ni}. \quad \boxed{\mathtt{i} \leftarrow \mathtt{w}. \quad \mathtt{ni} \leftarrow \mathtt{w}.} \qquad \forall i \in I$$

$$\mathtt{w} \leftarrow \mathtt{formula}_{\psi(T)}.$$

$$\mathit{expansion}(\psi(T))$$

$$\mathtt{w} \leftarrow \mathtt{not\ w}.$$

Since w is true, $\psi(T)$ must be satisfied for every I.

Theorem

The encoding $P_{dis}(T)$ is correct and complete w.r.t. solutions of T

Theorem

If $B \cup S$ is tight, $|P_{dis}(T)| \in \mathcal{O}(|T|)$

GROUNDING

Task grounding

- To deal with non-ground task, we define a notion of single-shot grounding
- Given a task $T = \langle B, S, E^+, E^- \rangle$, we define $ground(T)$ as:

$$\langle \bigcup_{R_i \in B} ground_U(R_i), \{ground_U(h_i) \mid h_i \in S\}, E^+, E^- \rangle$$

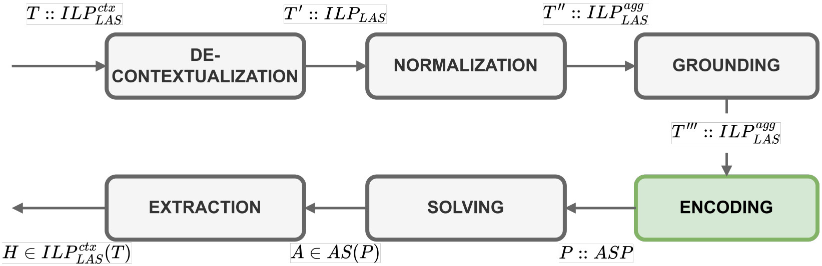
- Technically, $ground(T)$ is a task of the ILP_{LAS}^{agg} framework, where each element in the hypothesis space is an aggregation of rules.

Theorem

If T is safe, then T and $ground(T)$ are equivalent.

IMPLEMENTATION AND EXPERIMENTS

Previous ideas have been implemented in a tool named LASCO, based on the following 6 steps.



Benchmark 1: Learning the < relation

Configuration	$T_{comp}^{sat,tight}$	$T_{comp}^{unsat,tight}$	$T_{comp}^{sat,loop}$	$T_{comp}^{unsat,loop}$
lasco + clingo first	**0.19**	**0.22**	1.99	19.73
lasco + clingo opt	0.20	0.31	4.97	**13.96**
lasco + dlv first	0.44	0.61	42.62	46.77
lasco + dlv opt	e	0.49	e	58.26
ILASP2	0.43	4.72	**1.29**	85.63
ILASP2i	0.63	5.06	1.98	97.49
ILASP3	46.17	45.11	e	e
ILASP4	2.38	7.67	38.27	e

Benchmark 2: Learning logic puzzles

- LASCO is competitive on *nQueens*
- LASCO suffers from scalability issues on *FlowLines*, *StarBattle*, and *15Puzzle* which have non-tight hypothesis spaces causing exponential blow-up in the encoding size

Benchmark 3: Learning automata

Configuration	$T_{(ab)^*}$	$T_{pattern}$
lasco + clingo first	1.26	**0.88**
lasco + clingo opt	1.43	28.17
lasco + dlv first	30.12	e
lasco + dlv opt	33.79	83.44
ILASP2	**0.15**	e
ILASP2i	0.55	e
ILASP3	e	e
ILASP4	e	e

CONCLUSIONS

Summary of the results and future works

Summary of the results:

- We proposed a new LAS solver based on a single-shot, linear, disjunctive ASP encoding

Open question:

- Can we design a polynomial-size disjunctive encoding for every ground LAS task? (Conjecture: no)

Future works:

- Investigate new encodings like *Quantified ASP* (QASP), *Integer Linear Programming* (ILP), *Quadratic Programming* (QP) and *Constraint Programming* (CP)

- Incorporate preprocessing stages into the LASCO pipeline to optimize the input task before encoding

THANK YOU!

REFERENCES

Bibliography I

Michael Gelfond and Vladimir Lifschitz (1988). "The Stable Model Semantics for Logic Programming". In: *Proceedings of International Logic Programming Conference and Symposium*. Ed. by Robert Kowalski, Bowen, and Kenneth. MIT Press, pp. 1070–1080. URL: http://www.cs.utexas.edu/users/ai-lab?gel88.

Stephen Muggleton (1991). "Inductive logic programming". In: *New generation computing* 8, pp. 295–318.

Thomas Eiter and Georg Gottlob (1995). "On the computational cost of disjunctive logic programming: Propositional case". In: *Annals of Mathematics and Artificial Intelligence* 15, pp. 289–323.

Fangzhen Lin and Yuting Zhao (2004). "ASSAT: Computing answer sets of a logic program by SAT solvers". In: *Artificial Intelligence* 157.1-2, pp. 115–137.

Bibliography II

Martin Gebser, Benjamin Kaufmann, et al. (2011). "Potassco: The Potsdam answer set solving collection". In: *Ai Communications* 24.2, pp. 107–124.

Mark Law, Alessandra Russo, and Krysia Broda (2020). "The ilasp system for inductive learning of answer set programs". In: *arXiv preprint arXiv:2005.00904*.

Fredrik Heintz, Michela Milano, and Barry O'Sullivan (2021). *Trustworthy AI-integrating learning, optimization and reasoning*. Springer.

Daniele Fossemò et al. (2022). "Using Inductive Logic Programming to globally approximate Neural Networks for preference learning: challenges and preliminary results". In: *CEUR WORKSHOP PROCEEDINGS*, pp. 67–83.

Martin Gebser, Roland Kaminski, et al. (2022). *Answer set solving in practice*. Springer Nature.

Bibliography III

Talissa Dreossi et al. (2023). "Exploring ILASP through logic puzzles modelling". In: *CEUR Workshop Proceedings*. Vol. 3428. CEUR-WS.

Antonio Ielo et al. (2023). "Towards ILP-based LTL f passive learning". In: *International Conference on Inductive Logic Programming*. Springer, pp. 30–45.

Agostino Dovier, Talissa Dreossi, Andrea Formisano, et al. (2024). "XAI-LAW Towards a logic programming tool for taking and explaining legal decisions". In: *CEUR WORKSHOP PROCEEDINGS*. Vol. 3733. CEUR-WS.

Talissa Dreossi et al. (2024). "Towards Explainable Weather Forecasting Through FastLAS". In: *International Conference on Logic Programming and Nonmonotonic Reasoning*. Springer, pp. 262–275.