Verbalizing Answer Sets: the Map–Summarize Paradigm

Mario Alviano[1], Matteo Capalbo[1] and Sebastiano A. Piccolo[1,*]

[1]University of Calabria, Department of Mathematics and Computer Science (DeMaCS)

Abstract

While Answer Set Programming (ASP) is a powerful tool for symbolic reasoning, its outputs (i.e., sets of ground atoms) are often impenetrable to non-experts. We propose a formal map–summarize framework that decouples the translation of symbolic facts into text from the linguistic refinement of the final explanation. By evaluating five distinct strategies, ranging from raw LLM prompting to deterministic template-based mapping, we highlight the trade-offs between faithfulness, readability, and computational cost.

Keywords

ASP, Answer Set Verbalization, Map–Summarize, Explainability, Logic Programming for the End User

1. Introduction

Answer Set Programming (ASP) is a cornerstone of symbolic AI, particularly for knowledge representation and combinatorial problem solving [1]. Yet, its practical adoption is often limited by a readability gap. ASP solvers [2, 3] produce answer sets: collections of ground atoms describing models of a program. While answer sets are logically precise, they are difficult to parse for non-expert users.

Current ASP explanation tools, such as XCLINGO [4], XASP [5, 6], and UCOREXPLAIN [7], enrich solver outputs with symbolic justifications in the form of explanation graphs. As the demand for explainable AI (XAI) grows, translating these symbolic outputs into natural language becomes essential. However, most existing work focuses on translating natural language inputs into formal representations [8, 9]. In this work, we address the complementary challenge of making solver outputs intelligible to humans, a task we term *Answer Set Verbalization*.

To achieve this, we leverage Large Language Models (LLMs) for their ability to produce fluent text. However, feeding raw atoms directly into an LLM often leads to hallucinations or a loss of predicate-level semantics. We propose a structured *map–summarize* pipeline (Fig. 1) that first transforms atoms into textual fragments (mapping) and then aggregates them into a coherent narrative (summarization). Our contributions include a formalization of this framework, a publicly available *ASP Chef* recipe[1], and an empirical comparison of verbalization strategies.

2. The Map–Summarize Framework

The core of our approach is to treat verbalization as a structured transformation rather than a single-step black box translation.

The Mapping Phase. This phase converts an ASP database D into a list of textual fragments. The system utilizes Predicate Verbalization Resources (V): metadata defining predicate semantics (e.g., for `assign(Cel, Val)`, V might provide the template "Cell { `Cel` } is assigned value { `Val` }"). Atoms are processed according to a grouping strategy G (either predicate-wise G_{pw} or all-at-once G_{all}) using one of three Interpreters (I):

Joint Proceedings of the Workshops and Doctoral Consortium of the 42nd International Conference on Logic Programming (ICLP 2026), co-located with the Federated Logic Conference (FLoC 2026), July 18–19 and 24–25, 2026, Lisbon, Portugal

*Corresponding author.

mario.alviano@unical.it (M. Alviano); matteo.capalbo@unical.it (M. Capalbo); sebastiano.piccolo@unical.it (S. A. Piccolo)

0000-0002-2052-2063 (M. Alviano); 0009-0001-0114-9304 (M. Capalbo); 0000-0002-6986-3344 (S. A. Piccolo)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

[1]https://asp-chef.netlify.app/s/KR2026/verbalizing-asp

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-03

CEUR-WS.org/Vol-4311/EULP_short_04.pdf

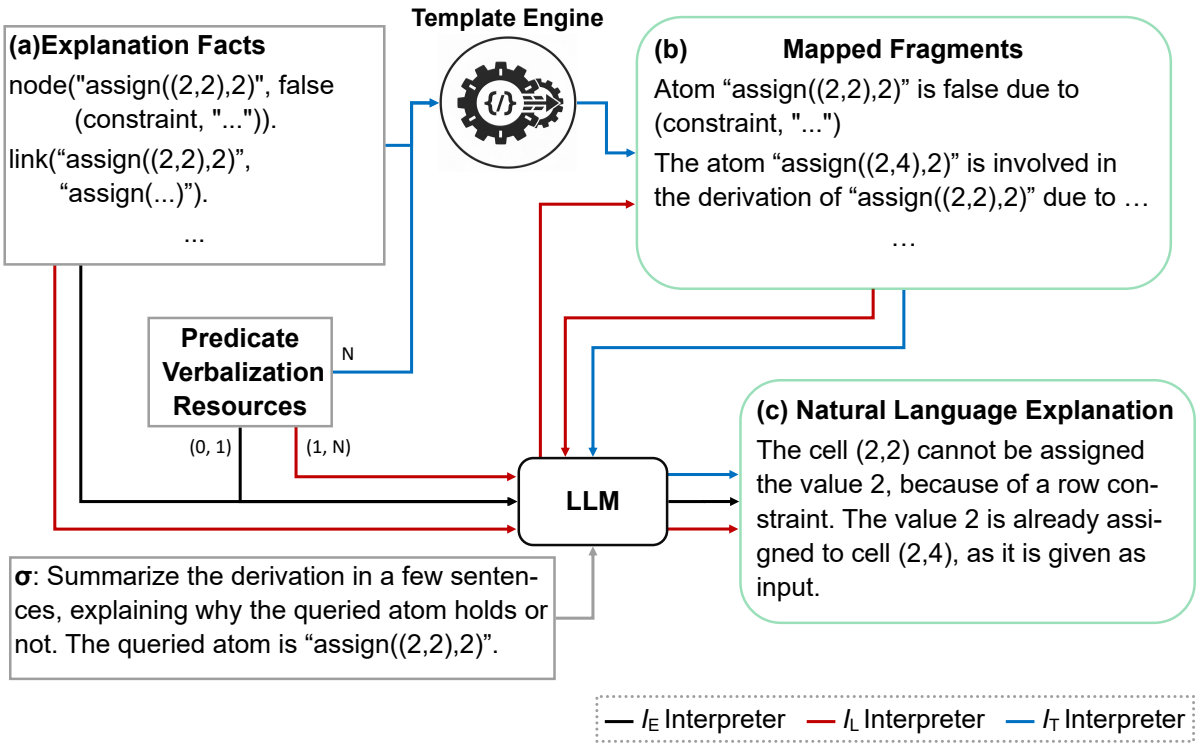


Figure 1: From solver explanation facts to a short natural-language explanation. (a) UCOREXPLAIN exports an ASP database encoding a derivation graph. (b) **Mapping Phase:** In modular strategies, atoms are independently transformed into text fragments via an interpreter: I_L (LLM-based) or I_T (Template-based). (c) **Summarization Phase:** An LLM aggregates these fragments (or raw atoms via I_E, or via I_L—in this case with schema descriptions) into a concise explanation.

- **Echo (I_E):** Passes raw atoms and schema descriptions directly to the next stage.
- **LLM-based (I_L):** Uses an LLM to generate an initial, unpolished description of a specific atom group.
- **Template-based (I_T):** A deterministic approach that injects solver constants into predefined strings, ensuring logical faithfulness.

The Summarization Phase. Mapped fragments can be repetitive or disjointed. This phase uses an LLM as a linguistic editor to aggregate fragments into a fluent description, controlled by an instruction prompt (σ). The complete verbalizer is represented as:

$$f_{I,G,V,\sigma}(D) = \mathbf{Sum}_{\sigma}\big(\mathbf{Map}_{I,G}(D, V)\big)$$

A key strength of the Map-Summarize paradigm is its versatility in handling two distinct verbalization modalities under a unified architecture: Data-to-Text (verbalizing flat facts and answer set models) and Derivation-to-Text (verbalizing complex proof trees and justification graphs produced by solvers like UCOREXPLAIN). In both cases, ground atoms, whether domain-specific facts or derivation nodes/links, are mapped into intermediate fragments before being aggregated into natural language narratives.

3. Evaluation

We instantiated our framework into five distinct strategies (S1–S5) to assess the impact of different configurations of I, G, and V (Table 2). We conducted three complementary studies: (i) a quantitative benchmark on structured Answer Set Programming (ASP) databases to assess core verbalization performance; (ii) a human evaluation of each model paired with its best-performing strategy, formatted as a

Table 1
Semantic preservation (BERTScore) and syntactic overlap (BLEU, ROUGE-L, chrF) across strategies, by models.

Model	BERTScore	ROUGE$_L$	BLEU	chrF	Runtime (s)
LLaMA 3 8b	0.902	0.365	0.099	0.543	84.383
LLaMA 4 16x17b	0.921	0.502	0.172	0.622	367.700
Ministral 3 14b	0.913	0.547	0.178	0.629	106.018
Mixtral 8x22b	0.927	0.505	0.194	0.634	2695.087
Deepseek R1 32b	0.927	0.553	0.206	0.643	1845.587
LLaMA 3 70b	0.926	0.521	0.207	0.649	339.017
Qwen 3 32b	0.934	0.588	0.247	0.668	2049.478
Mistral 3 24b	0.937	0.599	0.257	0.670	136.645

Table 2
Concrete verbalization strategies and their performance using Mistral 3 24b.

Strategy	Interpreter	Grouping	Mapping	BERTScore	ROUGE$_L$	BLEU	chrF	Runtime (s)
S1	I_{E}	G_{all}	Prompt	0.924	0.47	0.17	0.63	*65.05*
S2	I_{E}	G_{all}	Schema	*0.951*	*0.70*	*0.34*	*0.71*	**64.79**
S3	I_{L}	G_{all}	LLM	0.925	0.51	0.20	0.65	241.23
S4	I_{L}	G_{pw}	LLM	0.949	*0.70*	*0.34*	0.70	309.34
S5	I_{T}	G_{pw}	Template	**0.952**	**0.71**	**0.35**	**0.72**	67.09

head-to-head LLM competition; and (iii) a human evaluation of solver-generated explanation graphs to assess their utility for Explainable AI (XAI).

Study I: Quantitative Benchmark. We evaluated our framework on a dataset of fashion retail assistant requests in which ASP databases are paired with ground-truth natural language. We evaluated strategies S1–S5 across eight open-weight LLMs: Deepseek R1 32b, LLaMA 3 8b and 70b, LLaMA 4 16x17b, Ministral 3 14b, Mistral 3 24b, Mixtral 8x22b, and Qwen 3 32b. To evaluate both semantic preservation and syntactic overlap, we computed four standard NLP metrics: BERTScore, ROUGE$_L$, BLEU, and chrF. Results indicate that Mistral 3 24b is the most effective model overall (Table 1), significantly outperforming larger architectures and reasoning-focused models. Among the strategies, S5 (Template-based) and S2 (Schema-guided) performed best, underlining the importance of structured mapping for truthful and faithful ASP verbalization.

As shown in Table 2, strategies S2 and S5 achieve higher syntactic overlap with the ground truth compared to other configurations (captured by ROUGE$_L$, BLEU, and chrF). In contrast, strategies such as S1 and S3 frequently omit critical constants or substitute them with synonyms, leading to lower scores. However, faithful ASP verbalization requires the exact preservation of domain constants alongside overall semantic accuracy. Strategy S4, which executes multiple LLM calls (one per predicate group), also achieves strong performance (comparable to S2 and S5), but at a significantly higher computational cost (reflected in its runtime).

While strategies S2 (schema-guided) and S5 (template-based) require Predicate Verbalization Resources (V), the engineering effort involved is minimal. Because the number of predicate signatures in an ASP program is finite and significantly smaller in scale than the resulting ground derivation graphs, defining a single translation stencil per predicate type imposes low overhead. As shown by our results, this one-time specification effort yields substantial payoffs, dramatically reducing computational runtime while guaranteeing high semantic accuracy and grounding stability.

Study II: Human Assessment of LLMs Paired with Their Best-Performing Strategy. The quantitative benchmark highlighted that strategies S2 and S5 perform best across models, and that Mistral 3 24b is the top-performing model overall. However, qualitative inspection revealed nuanced

Table 3
Human evaluation results for all models paired with their best-performing strategy.

Model	Strategy	Elo Rating	# Wins	# Losses	# Ties
LLaMA 3 8b	S4	802	119	711	220
Mixtral 8x22b	S2	917	222	492	336
LLaMA 3 70b	S4	1000	332	337	381
Ministral 3 14b	S5	1001	321	324	405
Deepseek R1 32b	S2	1047	409	263	378
LLaMA 4 16x17b	S4	1048	398	247	405
Qwen 3 32b	S2	1085	462	194	394
Mistral 3 24b	S5	1097	467	162	421

trade-offs: reasoning models such as Deepseek R1 32b and Qwen 3 32b performed slightly better when paired with strategy S2 than with S5, whereas LLaMA models performed slightly better when paired with strategy S4.

These observations suggest that models and strategies interact to influence performance, and that standard automatic metrics may not capture all nuance in ASP verbalization. Because ASP verbalization targets end-user interpretability, we conducted a human evaluation pairing each LLM with its optimal strategy. Configured in an LLM Arena–style blind preference test, human annotators evaluated paired verbalizations generated from the same input answer set. Model identities were hidden, and model-verbalization pairs were presented in randomized order. We conducted 1,050 pairwise comparisons per model to compute Elo ratings alongside win, loss, and tie counts (Table 3).

The human evaluation largely corroborates our quantitative findings. Mistral 3 24b outperforms all other models, followed closely by Qwen 3 32b, a reasoning-oriented model. LLaMA 4 16x17b and Deepseek R1 32b perform comparably, trailing Mistral 3 24b by approximately 50 Elo points. Notably, Ministral 3 14b achieved a rating comparable to the significantly larger LLaMA 3 70b—further emphasizing the efficacy of verbalization strategy S5.

Study III: Human Evaluation for XAI. Having established Mistral 3 24b as the top-performing model and strategies S2 and S5 as the most effective overall, our final study investigates whether S2 or S5 is better suited for complex verbalization scenarios using Mistral 3 24b. Seven ASP experts conducted a blind, paired comparison of 15 explanation sets for Sudoku puzzles, evaluating faithfulness (fact support), truthfulness (logical correctness), and clarity. Faithfulness was assessed via a binary decision (faithful vs. non-faithful), truthfulness via a ternary categorization (correct, unclear, or incorrect), and clarity using a 5-point Likert scale. As shown in Table 4, S5 outperforms S2 across all metrics and is preferred in 46% of evaluation cases. Experts highlighted S5's superior truthfulness (92%) and consistent grounding stability.

To evaluate inter-annotator reliability, we computed Gwet's AC1 for categorical metrics (Truthfulness and Faithfulness) and Gwet's AC2 (with quadratic weights) for the ordinal Likert scale (Clarity). Gwet's coefficients were chosen because our annotators reached a high overall agreement (82%), yielding a high-prevalence distribution where Cohen's Kappa tends to be overly conservative. Inter-annotator agreement was notably high for correctness (Gwet's AC1: 0.803), confirming S5's advantage in producing logically sound outputs. Agreement on Faithfulness was moderate (AC1 = 0.591), while Clarity exhibited lower agreement (AC2 = 0.341), consistent with the subjective nature of stylistic fluency relative to logical correctness.

Together, these findings suggest that while schema guidance (S2) is effective for standard data-to-text conversion, deterministic mapping (S5) is critical for reliable XAI, as it eliminates semantic drift when verbalizing complex logical derivations.

Table 4
Summary of human evaluation comparing S2 and S5. Truthfulness indicates the proportion of correct (C), unclear (U), or incorrect (I) outputs. Clarity is rated on a 1–5 Likert scale.

Strategy	Faithful	Truthfulness			Clarity	Preferred
		C	U	I		
S2	0.73	0.82	0.11	0.07	3.41	28%
S5	0.79	0.92	0.06	0.02	3.78	46%

4. Conclusion

We presented a unified *map-summarize* framework for verbalizing ASP answer sets and derivation graphs. Our findings demonstrate that lightweight, predicate-aware mappings (S2, S5) allow mid-sized models like Mistral 3 24b to outperform significantly larger models by enforcing exact constant preservation and preventing semantic drift. Crucially, because predicate signatures are fixed and small in number relative to ground derivation trees, the minimal effort required to define verbalization resources pays off in both zero-hallucination accuracy and low computational cost.

Moving forward, this work opens several avenues for future research. First, we plan to investigate verbalization-aware explanation generation, revising upstream solver engines (such as UCOREXPLAIN) to produce proof trees optimized specifically for downstream natural language synthesis. Second, we aim to extend the framework into interactive, conversational XAI, allowing users to dynamically explore specific derivation branches via multi-turn natural language dialogue. Finally, to eliminate remaining manual overhead, we will explore automated template induction, leveraging LLM agents to infer predicate verbalization resources directly from logic program signatures and domain documentation.

Acknowledgments

This work was supported by the Italian Ministry of Health (MSAL) under POS projects CAL.HUB.RIA (CUP H53C22000800006) and RADIOAMICA (CUP H53C22000650006); by the Italian Ministry of Enterprises and Made in Italy under project STROKE 5.0 (CUP B29J23000430005); under PN RIC project ASVIN "Assistente Virtuale Intelligente di Negozio" (CUP B29J24000200005); by Regione Calabria NextGenGuides "Innovazione Tecnologica per le Guide Turistiche del Futuro tramite l'ausilio di tecnologie innovative AI e sistemi di geolocalizzazione avanzata" (CUP J39I24002040005); by MOZART "Modelli e tecniche di mOdernizzazione di applicaZioni e data silos a supporto di clienti esterni, field force e impiegati di backoffice basati su AI GeneRativa e apprendimento auTomatico" (CUP J49I24001740005); by the LAIA lab (part of the SILA labs). Mario Alviano is a member of Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM).

Declaration on Generative AI

During the preparation of this work, the authors used Gemini in order to perform grammar and spelling check. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

[1] G. Brewka, T. Eiter, M. Truszczynski, Answer set programming at a glance, Commun. ACM 54 (2011) 92–103. URL: https://doi.org/10.1145/2043174.2043195. doi:10.1145/2043174.2043195.

[2] M. Alviano, C. Dodaro, S. Fiorentino, A. Previti, F. Ricca, ASP and subset minimality: Enumeration, cautious reasoning and muses, Artif. Intell. 320 (2023) 103931. URL: https://doi.org/10.1016/j.artint.2023.103931. doi:10.1016/J.ARTINT.2023.103931.

[3] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Multi-shot ASP solving with clingo, Theory Pract. Log. Program. 19 (2019) 27–82. URL: https://doi.org/10.1017/S1471068418000054. doi:10.1017/S1471068418000054.

[4] P. Cabalar, B. Muñiz, Model explanation via support graphs, Theory Pract. Log. Program. 24 (2024) 1109–1122. URL: https://doi.org/10.1017/s1471068424000048. doi:10.1017/S1471068424000048.

[5] L. L. T. Trieu, T. C. Son, M. Balduccini, xasp: An explanation generation system for answer set programming, in: G. Gottlob, D. Inclezan, M. Maratea (Eds.), Logic Programming and Nonmonotonic Reasoning - 16th International Conference, LPNMR 2022, Genova, Italy, September 5-9, 2022, Proceedings, volume 13416 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 363–369. URL: https://doi.org/10.1007/978-3-031-15707-3_28. doi:10.1007/978-3-031-15707-3_28.

[6] M. Alviano, L. L. T. Trieu, T. C. Son, M. Balduccini, The XAI system for answer set programming xasp2, J. Log. Comput. 34 (2024) 1500–1525. URL: https://doi.org/10.1093/logcom/exae036. doi:10.1093/LOGCOM/EXAE036.

[7] M. Alviano, S. Hahn, O. Sabuncu, H. Weichelt, Answer set explanations via preferred unit-provable unsatisfiable subsets, in: C. Dodaro, G. Gupta, M. V. Martinez (Eds.), Logic Programming and Nonmonotonic Reasoning - 17th International Conference, LPNMR 2024, Dallas, TX, USA, October 11-14, 2024, Proceedings, volume 15245 of *Lecture Notes in Computer Science*, Springer, 2024, pp. 187–199. doi:10.1007/978-3-031-74209-5_15.

[8] M. A. B. Santana, I. Kareem, F. Ricca, Towards automatic composition of ASP programs from natural language specifications, in: Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024, ijcai.org, 2024, pp. 6198–6206. URL: https://www.ijcai.org/proceedings/2024/685.

[9] S. Szeider, Cp-agent: Agentic constraint programming, CoRR abs/2508.07468 (2025). URL: https://doi.org/10.48550/arXiv.2508.07468. doi:10.48550/ARXIV.2508.07468. arXiv:2508.07468.