Advancing Knowledge-Based Reasoning for Automated Workflow Composition

Nikolas Bertrand

University of Potsdam, Department of Computer Science, An der Bahn 2, 14476 Potsdam, Germany

Abstract

Many scientific disciplines generate increasingly large datasets that require complex Data Analysis Workflows (DAWs) for processing. As the number of available tools grows, manually constructing valid workflows becomes challenging. Existing automated synthesis tools, such as the Automated Pipeline Explorer (APE), successfully compose workflow candidates using SAT-based encodings but face significant scalability and extendability limits. This doctoral research frames automated workflow synthesis as a search for workflow candidates, representing them as stable models using Answer Set Programming (ASP). To support this research, a prototype framework was developed utilizing clingo's multi-shot incremental solving architecture that finds the same workflow candidates as composed by APE using identical domain model inputs and workflow specification. Preliminary results indicate that ASP provides significant performance and extensibility benefits. A primary challenge in transitioning to ASP was that dense domain models often triggered combinatorial explosions and reached clasp identifier limits during grounding. To mitigate these bottlenecks, an optimized backend featuring a compressed-candidate representation was introduced. This approach shifts computationally heavy compatibility and bindability reasoning from the ASP grounder to a Python-based pre-computation layer. Consequently, for small-to-medium domain models, ASP demonstrates a clear advantage in both runtime and memory usage. Large and dense domain models remain the main open scalability challenge. Depending on the specific domain model and time horizon, ASP may consume more memory or experience runtime spikes exceeding those of APE, even when grounding remains faster. Building upon this foundation, future work will focus on optimization, particularly for longer time horizons. My doctoral thesis, specifically, will explore the integration of user preferences through both hard and soft constraints. This approach will allow us to prune the search space and rank workflow candidates based on metrics provided by domain scientists.

Keywords

Scientific Data Analysis Workflows, Answer Set Programming, Workflow Synthesis

CEUR-WS.org/Vol-4311/DC_short_06.pdf

1. Introduction and problem description

The increasing volumes of data generated by research across all disciplines require increasingly complex computational analysis processes. These processes are often implemented as *scientific workflows* (also called data-analysis workflows or computational pipelines) [1]. A scientific workflow describes how data move through a set of usually preexisting computational tools: the output produced by one step becomes input to subsequent steps, collectively transforming initial data into a desired scientific result [2]. As registries that manage tool definitions such as `bio.tools` [3] grow into the tens of thousands of entries, manually authoring valid workflows has become impractical. The *Automated Pipeline Explorer* (APE) [4, 5, 6] attempts to solve this issue by synthesizing workflow candidates from a semantic domain model and constraints expressed in SLTL^x. Two limitations motivate my research: APE's SAT-based finite-automaton encoding strains on medium and larger domains, and its one-shot mode returns all valid candidates until a configured number is reached, rerunning the complete process from the start after the query changes. Depending on the domain model, this can amount to thousands of candidates, turning both composition and scientific evaluation into time-consuming tasks. My doctoral research re-frames workflow synthesis as a search for stable models in *Answer Set Programming* (ASP) [7], executed by clingo [8]. In addition to performance benefits, this approach, thanks to its incremental

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.

nikolas.bertrand@uni-potsdam.de (N. Bertrand)

0000-0001-7259-2111 (N. Bertrand)



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

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

published 2026-11-03

mode, would allow scientists to construct workflows manually within an interface such as a workflow engine, receiving at each step a choice of tools that remain satisfiable under the ASP framework running in the background. The research aims to retain parity with APE's composed candidates while moving from enumeration toward preference-based, *ranked* recommendations. Since registries such as `bio.tools` rarely contain complete metadata and tool annotations, missing values need sensible defaults or uncertainty-aware predictions to avoid dropping candidates or skewing rankings. A neuro-symbolic or LLM layer can supply such missing data while leaving the symbolic encoding responsible for hard constraint soundness. An initial pizza-baking prototype confirmed viability but did not generalize to realistic, domain models. The resulting prototype, *clindaws*, is dimension-independent: drawing on *Object-Oriented ASP* and smart expansion [9, 10], it attaches attributes to identifiers rather than grounding out a full type cross product.

2. Background

APE encodes workflow synthesis as a temporal finite automaton: a domain model fixes the tools, typed ports, and ontology terms, while a workflow specification adds typed inputs, goal outputs, and constraints. APE translates constraints into CNF, delegates to MiniSAT [11], and iterates over workflow lengths until a solution, length, or timeout cap is reached. It has been deployed in proteomics [12], GIS [13], and computational materials science [14], and is part of workflomics [15]. The motivating real-world setting combines pyiron [16], which wraps simulation tasks into job objects. Using an additional tool such as semantikon allows then to annotate Python functions with ontology URIs and physical units. These annotations become an `rdflib` graph interpreted against ontologies such as PMDco [17], which is based on Basic Formal Ontology (BFO) [18]. turning numerical artifacts into typed objects matchable against input requirements. A pipeline connected APE, pyiron and semantikon was demonstrated in [14]. Answer set planning [19] and PDDL-to-ASP translations such as Plasp [20] search horizon by horizon for goal-reaching action sequences. Workflow synthesis shares this pattern but reasons over a *typed artifact flow*; a direct PDDL-style encoding would reintroduce the type-ontology cross product that *clindaws* avoids. The same caveat applies to ASP-based configuration [21]. Workflow synthesis deals with similar problems where temporal tool positions, data dependencies, and port bindings have to be constructed during search. *clindaws* draws on equivalence compression and symmetry breaking from SAT planning [22, 23], and on ontology-based type reasoning that compresses signature profiles without an external DL reasoner [24]. clinguin [25] provides a generic UI layer on top of multi-shot clingo, and smart expansion has been applied to ASP-based interactive configuration [9]; both match the per-step API envisioned for *clindaws*. Recent LLM systems suggest workflows from natural-language requests [26, 27], but evaluations report correctness and executability issues [28]; *clindaws* is complementary because it returns only workflows satisfying formal constraints. Telingo [29] is also relevant because SLTL^x constraints currently lower into time-keyed integrity constraints that inflate and partially invalidate the grounded program after edits. Existing approaches therefore do not jointly provide APE-level semantic soundness, scalable incremental synthesis, interactive recommendation, preference-aware ranking, and robustness to incomplete metadata.

3. Research Questions

The thesis aims to establish multi-shot clingo as a scalable and extensible foundation for workflow composition, moving from pure enumeration toward interactive, preference-driven synthesis. The research questions below are deliberately ordered so that RQ5 enters as a precondition of RQ3 under the conditions of RQ4, not as a separate direction.

RQ1 Can workflow synthesis be expressed using ASP without losing semantic parity with APE on shared domain models?

RQ2 Can multi-shot incremental solving make the iterative composition loop interactive for large scale domains?

RQ3 How can hard, weak constraints and preferences be used to *rank* candidates by domain-scientist-supplied metrics, rather than only enumerate them?

RQ4 How can synthesis remain sound when the domain model is incomplete (registry gaps, missing tool annotations, missing ranking metadata)?

RQ5 Can a neuro-symbolic layer supply the facts RQ4 exposes (filling annotations, predicting ranking metadata, learning ASP rules from logs) so that RQ3 becomes feasible on real registries, while also relieving residual scaling bottlenecks?

4. Approach

The work proceeds along three stages, each motivated by the limitations of the previous one.

Stage 1: Viability check. A small pizza-baking ontology confirmed that multi-shot ASP can produce APE-style candidates from an ontology, tool annotations, a specification, and behavioral constraints. However, the encoding required all artifacts to share one fixed dimension set, unlike real domain models.

Stage 2: Real-world integration. Computational materials-science workflows span different size scales whose heterogeneous artifacts cannot share a fixed type set, so the synthesis system must support arbitrary per-port ontology terms while preserving APE's semantic guarantees.

Stage 3: *clindaws* prototype feature parity. *clindaws* is dimension-independent: it attaches per-port metadata to identifiers rather than grounding a full type cross product. The encoding evolved through three generations: a single-shot encoding, a plain multi-shot encoding using clingo's `base/init/step(t)/check(t)` skeleton, and an optimized multi-shot backend. This backend factors input requirements into signature profiles, prunes candidates by reachability, and emits compressed-candidate facts, lifting the heaviest bindability reasoning out of the grounder in the spirit of FastFound [10]. Two-stage horizon filtering defers goal-validation grounding until structural and exact-horizon checks pass [9]. Merged output ports whose consumers diverge are reintroduced through selective decompression (K-cluster approximation or 1:N split, the latter recovering full APE coverage), while assumption-driven horizon control lets the Python layer switch active horizons without re-grounding.

5. Preliminary Evaluation

clindaws was evaluated against APE across GIS, image manipulation, metabolomics, meta-genomics, materials science, and three MassSpectrometry tiers from 20 to 1678 tools, with the largest tier derived from the full `bio.tools` registry. Each domain was run with and without constraints; both systems received identical inputs and were measured for peak memory, grounding time, and solving time. On all small and medium benchmarks, all *clindaws* variants outperform APE, with the gain largest on ontology-heavy cases: Metabolomics (3421 ontology terms, 37 tools) uses ~133 MB peak memory versus 4100–4600 MB for APE, and the small MassSpec tier drops from 7800–12700 MB to 154–242 MB. Grounding typically completes in under 3 s versus tens to hundreds of seconds for APE. The 1:N decomposition matches APE's workflow space exactly, confirming semantic parity (RQ1); K-cluster is a controlled approximation except in high-multiplicity cases. Within the test cases, compressed-only and K-cluster decompression missed one high-multiplicity case with 12 of 763 workflow candidates. At the largest tier (1678 tools), the picture changes: peak memory reaches 47–52 GB versus 12–14 GB for APE because the precomputed bind-ability graph itself becomes large. Grounding still beats APE (197–266 s vs. 314–818 s) and total runtime is lower on three of four full-bio cases; on the fourth, APE finishes in 1714 s while the *default* variant needs 3600 s and *1:N* crashes by exceeding `clasp`'s 32-bit identifier ceiling [30]. These results do not yet establish ASP as uniformly superior for very large dense registries; rather, they show that the current ASP formulation is already advantageous on small and

medium domains, and that the case for very large domains depends on further reducing grounding pressure and memory use.

6. Planned Evaluation and Future Work

Combinatorial explosion and grounding pressure. On large domains the bind-ability graph and per-horizon goal-validation grounding still grow enough to significantly increase memory usage, trigger `clasp`'s 32-bit identifier-range error, and inflate solving time. Technical debt remains in late pruning of integrity constraints, limited output-port symmetry breaking, rebuilding precomputed candidates after configuration changes, and time-keyed behavioral constraints. More specifically, integrity constraints are not yet folded into the backward candidate closure, output-port symmetry canonicalization covers only within-consumer permutations, and behavioral constraints still lower into time-keyed integrity constraints. The next evaluation will therefore compare cached and optimized *clindaws* variants against APE and the current backend for runtime, memory, grounding size, semantic parity, and responsiveness during iterative query changes. The expected achievements are caching grounded atoms and partial candidate sets, stronger symmetry breaking, and evaluating grounding-compilation hybrids such as proASP [31]. These optimizations support integrating *clindaws* into APE, workflow engines such as Nextflow [32], and clinguin-style front ends [25].

The planned evaluation will combine solver-level benchmarks with domain-scientist review of result quality. Technically, I will extend the benchmark suite to larger registry-derived domains and longer horizons, measuring runtime, memory, grounding size, semantic parity, and response time under iterative query changes. On the domain side, ranked result lists and interactive next-step recommendations will be reviewed for whether they reflect relevant scientific preferences and reduce the manual effort needed to identify useful workflow candidates.

Temporal constraints and BFO-aligned domain models. Adopting temporal-logic ASP along the lines of Telingo [29] addresses constraint re-grounding natively and opens a path to BFO-aligned domain models in ASP. This is useful for multi-scale computational materials science workflows [14], which may run in parallel and interact across scales. A later prototype should therefore distinguish temporal compatibility, data-dependency compatibility, and semantic type compatibility instead of encoding all of them as ordinary step-indexed constraints.

Preference-driven synthesis under incomplete domain models and metadata (RQ3–RQ5). The main expected achievement is the shift from enumeration to preference-based *ranking*. Hard constraints continue to prune; preferences rank candidates by domain-scientist metrics such as citation count, OS support, hardware requirements, expected runtime, robustness, and preferred tools. Locally non-preferred tools must not be pruned too early, because they may still appear in the globally top-ranked workflow. Real registries such as `bio.tools` rarely contain complete preference metadata, missing values therefore need sensible defaults and, where possible, confidence-aware predictions. Probabilistic frameworks such as SLASH [33] and NeurASP [34] can express classifier or LLM outputs as probability distributions over facts such as operating-system support, citation/ranking buckets, or probable tool annotations. The ASP encoding would still enforce hard type and workflow constraints, while weak constraints use predicted facts only for ranking. Differentiable stable-model computation, in particular Neural Decision-Propagation [35], and inductive ASP learning via NSIL [36] are additional candidates for scaling and learning parts of this loop.

Acknowledgments

This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 414984028 – SFB 1404 FONDA.

Declaration on Generative AI

During the preparation of this work, the author used Gemini and DeepL for grammar and spell check. After using these services, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] F. Suter, T. Coleman, I. Altintas, et al., A terminology for scientific workflow systems, Future Generation Computer Systems 174 (2026) 107974. doi:10.1016/j.future.2025.107974.

[2] U. Leser, M. Hilbrich, S. Wilkinson, R. Ferreira da Silva, Workflow systems for large-scale scientific data analysis, Berlin Universities Publishing, 2026. URL: https://depositonce.tu-berlin.de/handle/11303/25179. doi:10.14279/depositonce-24000.

[3] J. Ison, H. Ienasescu, P. J. Chmura, et al., The bio.tools registry of software tools and data resources for the life sciences., Genome Biology (2019). doi:10.1186/S13059-019-1772-6.

[4] V. Kasalica, A.-L. Lamprecht, APE: A command-line tool and API for automated workflow composition, in: International Conference on Computational Science, Springer International Publishing Cham, 2020, pp. 464–476.

[5] V. Kasalica, A.-L. Lamprecht, Workflow Discovery with Semantic Constraints: The SAT- Based Implementation of APE, Electronic Communications of the EASST 78 (2020). doi:10.14279/tuj.eceasst.78.1092.

[6] V. Kasalica, Synthesis of Scientific Workflows: Theory and Practice of an Instance-Aware Approach, Ph.D. thesis, Utrecht University, 2022. URL: https://dspace.library.uu.nl/handle/1874/423894.

[7] V. Lifschitz, Answer set programming and plan generation, Artificial Intelligence (2002). doi:10.1016/S0004-3702(02)00186-8.

[8] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Multi-shot asp solving with clingo, Theory and Practice of Logic Programming (2019). doi:10.1017/S1471068418000054.

[9] L. Balážová, R. Comploi-Taupe, S. Hahn, N. Rühling, G. Schenner, Smart expansion techniques for ASP-based interactive configuration, Theory and Practice of Logic Programming (TPLP) (2025). Under consideration for publication.

[10] A. Beiser, M. Gebser, M. Hecher, S. Woltran, FastFound: Easing the ASP bottleneck via predicate-decoupled grounding, in: Proceedings of the 22nd Int. Conference on Principles of Knowledge Representation and Reasoning (KR 2025), 2025, pp. 100–109. doi:10.24963/kr.2025/10.

[11] N. Sörensson, N. Eén, Minisat v1.13 - a sat solver with conflict-clause minimization, in: Proceedings of SAT 2005, 2005, pp. 502–518.

[12] V. Kasalica, V. Schwämmle, M. Palmblad, J. Ison, A.-L. Lamprecht, APE in the Wild: Automated Exploration of Proteomics Workflows in the bio. tools Registry, Journal of proteome research 20 (2021) 2157–2165. doi:10.1021/acs.jproteome.0c00983.

[13] J. F. Kruiger, V. Kasalica, R. Meerlo, A.-L. Lamprecht, E. Nyamsuren, S. Scheider, Loose programming of GIS workflows with geo-analytical concepts, Transactions in GIS 25 (2021) 424–449. doi:10.1111/tgis.12692.

[14] N. Bertrand, C. Grunert, O. Waseda, T. Hickel, A.-L. Lamprecht, Semantic workflow composition for computational materials science, 2026. URL: https://doi.org/10.5281/zenodo.18822541. doi:10.5281/zenodo.18822541.

[15] V. Kasalica, F. Morato, A. Lamprecht, M. Palmblad, Benchmarks for bioinformatics workflow bake offs, 2024. doi:10.37044/OSF.IO/RQ2DA.

[16] J. Janssen, S. Surendralal, Y. Lysogorskiy, M. Todorova, T. Hickel, R. Drautz, J. Neugebauer, pyiron: An integrated development environment for computational materials science, Computational Materials Science 163 (2019) 24–36. doi:10.1016/j.commatsci.2018.07.043.

[17] B. Bayerlein, M. Schilling, H. Birkholz, M. Jung, J. Waitelonis, L. Mädler, H. Sack, Pmd core

ontology: Achieving semantic interoperability in materials science, Materials & Design (2023). URL: https://api.semanticscholar.org/CorpusID:266646971.
[18] J. N. Otte, J. Beverley, A. Ruttenberg, Bfo: Basic formal ontology, Applied Ontology 17 (2022) 17 – 43. URL: https://api.semanticscholar.org/CorpusID:247354631.
[19] C. Tran, E. Pontelli, M. Balduccini, T. Schaub, Answer set planning: A survey, Theory and Practice of Logic Programming 23 (2022) 226–298. doi:10.1017/S1471068422000072.
[20] Y. Dimopoulos, M. Gebser, P. Lühne, J. Romero, T. Schaub, plasp 3: Towards effective ASP planning, Theory and Practice of Logic Programming 19 (2019) 477–504. doi:10.1017/S1471068418000583.
[21] J. Baumeister, S. Hahn, K. Herud, M. Ostrowski, J. Reutelshöfer, N. Rühling, T. Schaub, P. Wanko, Towards industrial-scale product configuration, Theory and Practice of Logic Programming (2026). FirstView.
[22] J. Rintanen, Symmetry reduction for sat representations of transition systems, in: Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS 2003), 2003, pp. 32–41. URL: https://api.semanticscholar.org/CorpusID:335203.
[23] J. M. Crawford, M. L. Ginsberg, E. M. Luks, A. Roy, Symmetry-breaking predicates for search problems, KR (1996).
[24] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider, The description logic handbook: Theory, implementation, and applications, Description Logic Handbook (2003). doi:10.1108/K.2003.06732IAE.006.
[25] A. Beiser, S. Hahn, T. Schaub, ASP-driven user-interaction with clinguin, in: Proceedings of the 40th International Conference on Logic Programming (ICLP 2024), volume 416 of *Electronic Proceedings in Theoretical Computer Science*, 2025, pp. 215–228. doi:10.4204/EPTCS.416.19.
[26] M. Esmailoghli, M. Weidlich, FlowPilot: A suggestion system for designing scientific workflows, Proceedings of the ACM on Management of Data 4 (2026) 1–27. doi:10.1145/3786653.
[27] J. Xu, W. Du, X. Liu, X. Li, LLM4Workflow: An LLM-based automated workflow model generation tool, in: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering (ASE 2024), Tool Demonstrations, 2024, pp. 2394–2398. doi:10.1145/3691620.3695360.
[28] K. Alam, B. Roy, From prompt to pipeline: Large language models for scientific workflow development in bioinformatics, arXiv preprint arXiv:2507.20122 (2025).
[29] P. Cabalar, R. Kaminski, T. Schaub, A. Schuhmann, Temporal answer set programming on finite traces, Theory and Practice of Logic Programming 18 (2018) 406–420. doi:10.1017/S1471068418000297.
[30] M. Gebser, B. Kaufmann, A. Neumann, T. Schaub, clasp : A conflict-driven answer set solver, International Conference on Logic Programming and Non-Monotonic Reasoning (2007). doi:10.1007/978-3-540-72200-7_23.
[31] C. Dodaro, G. Mazzotta, F. Ricca, Blending grounding and compilation for efficient ASP solving, International Conference on Principles of Knowledge Representation and Reasoning (2024).
[32] P. D. Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, C. Notredame, Nextflow enables reproducible computational workflows, Nature Biotechnology 35 (2017) 316–319. doi:https://doi.org/10.1038/nbt.3820.
[33] A. Skryagin, W. Stammer, D. Ochs, D. S. Dhami, K. Kersting, Neural-probabilistic answer set programming, International Conference on Principles of Knowledge Representation and Reasoning (2022). doi:10.24963/KR.2022/48.
[34] Z. Yang, A. Ishay, J. Lee, Neurasp: Embracing neural networks into answer set programming, International Joint Conference on Artificial Intelligence (2020). doi:10.24963/IJCAI.2020/243.
[35] T. Eiter, K. Inoue, S. Moriyama, Neural decision-propagation for answer set programming, arXiv preprint arXiv:2605.01797 (2026).
[36] D. Cunnington, M. E. Law, J. Lobo, A. Russo, Neuro-symbolic learning of answer set programs from raw data, International Joint Conference on Artificial Intelligence (2023). doi:10.24963/IJCAI.2023/399.