Richer Oracle Interactions in Constraint Acquisition: Theory and Application to Program Verification

Sarra Djebour[a,b,*]

[a]Université Paris-Saclay, CEA, List, Palaiseau, France
[b]Université Paris-Saclay, LISN, Gif-sur-Yvette, France

Abstract. Constraint Acquisition (CA) is a learning-based approach that aims to automate the construction of constraint models by interacting with a user or an oracle. Traditionally, CA assumes a minimally informative oracle that can only classify simple examples as valid or invalid. This limited feedback often results in long acquisition processes, especially for complex concepts. In my PhD research, I explore how relaxing the assumptions about the oracle can lead to more efficient acquisition. By considering oracles capable of providing richer answers—beyond mere classification—we can design more expressive queries and significantly accelerate learning. I propose a new class of queries that, given an example, requests both its classification and an explanation justifying that classification. This allows the learner to extract more information from each interaction, especially in domains with large or structured hypothesis spaces. I formalize the assumptions required on the oracle to support these queries and analyze their theoretical and practical impact on learning efficiency. My objective is to demonstrate how my approach can be applied to the problem of precondition inference in program verification, where the verification engine acts as a powerful oracle capable of providing informative feedback.

1 Introduction

Over the past decades, constraint programming has made significant progress in modeling and solving combinatorial problems. However, building the corresponding model for a given problem often remains challenging and typically requires expert insight. To address this, active CA emerged as a method to automate the modeling process. A learner generates queries—instantiations of problem variables—and the user classifies each as a solution or non-solution to the target concept i.e., the concept we want to learn. Numerous CA approaches have been proposed over the years, including the use of membership queries [7], partial queries [4], generalization queries [5], recommendation queries [8], an argument-based approach [13], agent-based matchmaking to identify user preferences [9]. Some require greater effort from the user to answer, aiming to reduce the number of user-learner interactions without overburdening the user. In my PhD work, I explore how to enhance the scalability of constraint acquisition (CA) in fully automated contexts, with a focus on precondition inference. The recent PRECA framework [11, 12] applied CA [7] to this task by replacing the user with an automated oracle, which demonstrates that delegating the query classification task to an oracle can be highly effective, as it can classify thousands of queries in seconds, enabling faster interaction. However, despite the PRECA oracle's efficiency, the query generation process remains a bottleneck that limits the number of queries processed in practice, thus impacting scalability. As part of my research, I investigate richer oracle interactions in CA by introducing new query types that better exploit the oracle's capabilities and reduce the number of queries that need to be generated. In line with previous CA work [5, 4, 8, 6, 2, 14, 9, 13], I specifically aim to develop queries suited for fully automated environments, as it is still unclear how existing query types can be effectively used with a fully automated oracle.

Contribution. In this paper we propose the following contributions:

- We introduce a new type of queries *explanatory query* that requires the oracle to provide both a classification (yes/no) and an explanation justifying that classification.
- We propose XCONACQ, an extension to the state-of-the-art CA algorithm CONACQ leveraging our new explanatory queries.
- We show how to apply XCONACQ for precondition inference, using *path predicates* to efficiently answer explanatory queries.

2 Background

We now present the necessary background on CA and how it can be used for precondition inference.

Constraint acquisition can be viewed as an interaction between user and learner. As input, the learner is given a bias B, which consists of a collection of constraints over the problem variables X. The goal is for the learning process to output a subset of constraints, called a constraint network, that exactly models the target concept T. They represent the constraints that must be satisfied for a given CSP. To do so, the learner generates membership queries [1] — complete assignments of the variables X over their respective domains D — and asks the user to label them as either solutions (*yes*) or non-solutions (*no*) of T. Based on the user's answer, the learner filters accordingly the candidate space —i.e., the set of all possible constraint networks that can be constructed from the bias—. The acquisition process ends when no other informative query can be generated. By informative query, we refer to a query that filters out concepts from the candidate space regardless of their classification (yes/no). Various other types of queries have been used in CA over the past years [7, 4, 5, 8]. They are usually designed to minimize effort for both the user and the learner, but mostly prioritize the user, who often has limited knowledge.

* Email: sarra.djebour@cea.fr

Precondition inference using CA. Recent work [11, 12] adapted CA for precondition inference which refers to identifying the conditions P on a function's inputs for it to successfully execute and satisfy a given postcondition Q [10]. It led to a new framework, dubbed PreCA, where the user is replaced by an oracle that executes the learner's queries — now interpreted as inputs to the function under analysis. The oracle runs the test case and returns *yes* if it terminates, does not crash, and satisfies the postcondition Q; *no* if it crashes, or terminates but does not satisfy Q; and *ukn* for unknown if the execution does not terminate given a timeout. At the end of the process, the learner will return the concept as a set of constraints that represent the precondition. PreCA considers only membership queries as it remains unclear how to automate an oracle for the other query types previously discussed.

3 Motivation

Membership-based CA techniques have been used for precondition inference. However, in some cases, the learning process takes too long to reach convergence and ultimately timeouts. This is due to the size of the bias, which becomes too large to be handled efficiently. Interestingly, our experiments show that, in these cases, the precondition can often be expressed using only 7% of the bias. This observation highlights the importance of efficiently filtering out irrelevant constraints, especially as the candidate space expands with the bias, since it represent all possible constraint networks that can be constructed from it. We propose leveraging the automated oracle's ability to answer more complex queries than simple membership ones, without affecting response time, and designing query types that exploit this capability to improve acquisition scalability.

4 Acquisition with Explanatory Queries

We define explanatory queries and show how they can be used in CA.

Definition 1 (Explanatory Query). *An* explanatory query *is a complete assignment e over X submitted to the oracle. In addition to answering whether e satisfies the target concept (*yes *or* no*), the oracle also returns a (not necessarily minimal) set of constraints I that explains the classification: We note $sol(I)$ and $sol(T)$ the sets of solutions corresponding to each constraint set, respectively.*

- *If the answer is* yes*, I contains sufficient constraints to justify why e satisfies the target concept, i.e., $e \in sol(I) \subseteq sol(T)$;*
- *If the answer is* no*, I contains sufficient constraints to justify why e violate the target concept, i.e., $e \in sol(I) \subseteq sol(\neg T)$.*

Explanatory queries offer guidance to the learner by narrowing down the search space. The main difference from [13] is that their argumentation queries are used in a passive learning setting (queries are given as input and not generated by the learner), and while argumentation queries must belong to the bias B, explanatory queries are not subject to this restriction.

Explanatory queries for Conacq. We extend the Conacq algorithm [7] to handle *explanatory queries*, which we refer to as XConacq. The core idea is retained, but the candidate space is updated not with membership queries, but directly from a query's explanation I. We define a new approach for processing positive and negative explanations. Since we now deal with a set of solutions rather than one, concepts that fail to classify all solutions in I are filtered out. As the candidate space is updated differently depending on the classification, we define how each case should be processed. We respectively note I^+ and I^- positive and negative explanatory query results. They are handled as follows:

- *Positive Explanation.* In classical CA methods, constraints violating a positively classified query are discarded, as they cannot be part of the final concept—otherwise, the concept would incorrectly reject a positively labeled example. Similarly, for a positively answered explanation query, we remove any constraint that violates any solution in the provided explanation I^+, since we know that all its solutions are classified as positive ones. The discarded constraints can be computed as follows:

$$\mathcal{K}(I^+) = \{c \in B \mid sol(I^+ \cup \neg c) \neq \varnothing\}$$

- *Negative Explanation.* For Negative explanation, we ensure that for each class of query — i.e., queries violating same constraints — covered by I^-, at least one constraint violating all such classes is included in the target concept. Updating the concept space requires extracting these classes from I^- which is non-trivial. Therefore, we define a *Randomized strategy* to build such a set. The goal is to extract as many class of queries as possible from the explanation by generating a fixed number of solutions and, for each solution, using its set of violated constraints to update the candidate space. This optimistic strategy aims to obtain as many informative queries as possible from I^-. We note that some of them may not be extracted from the explanation, and queries sharing the same violated constraints may be generated. While it preserves the correctness of the acquisition, it remains non-optimal.

5 Explanatory Queries for Precondition inference

We now describe how XConacq can be adapted for precondition inference. We use the *path predicate* associated with a given explanatory query as an explanation.

Definition 2 (Path predicate). *Given an execution path in a software, its path predicate π is a set of constraints on the software inputs necessary and sufficient to follow the exact same execution path.*

Given a full inputs assignment e, the path predicate π can be constructed by gathering all the branch conditions taken during the execution of the program on e. It will include all satisfied constraints from conditionals (if and loop conditions) and conditions associated with error handling (e.g., division-by-zero, null pointer dereference). Hence, every solution in π share the same classification. The extraction of path predicate is a common task in program analysis and can be done efficiently [3]. Once extracted by the oracle, the execution path is provided to XConacq as the explanation.

6 Conclusion

We propose a new type of queries that extract a maximum of information from the user during the CA process. This approach is particularly interesting for applications where we have a powerful oracle rather than a user whose effort needs to be minimized. We define and show how to include this new type of queries in Conacq, leading to XConacq, and explain how it can be instantiated for code analysis to find a function's precondition.

For future work, we aim to evaluate our method, compare it to the state of the art, and explore various strategies to extract information from negative explanations or having the oracle perform symbolic execution [3] based on execution paths to produce more general explanations for each query.

References

[1] D. Angluin. Queries and concept learning. *Mach. Learn.*, 2(4):319–342, 1987. doi: 10.1007/BF00116828. URL https://doi.org/10.1007/BF00116828.

[2] R. Arcangioli and N. Lazaar. Multiple constraint acquisition. In D. Jannach, J. Mengin, B. Mobasher, A. Passerini, and P. Viappiani, editors, *Proceedings of the IJCAI 2015 Joint Workshop on Constraints and Preferences for Configuration and Recommendation and Intelligent Techniques for Web Personalization co-located with the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015), Buenos Aires, Argentina, July 27, 2015*, volume 1440 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015. URL https://ceur-ws.org/Vol-1440/Paper4.pdf.

[3] R. Baldoni, E. Coppa, D. C. D'Elia, C. Demetrescu, and I. Finocchi. A survey of symbolic execution techniques. *ACM Comput. Surv.*, 51(3): 50:1–50:39, 2018. doi: 10.1145/3182657. URL https://doi.org/10.1145/3182657.

[4] C. Bessiere, R. Coletta, E. Hebrard, G. Katsirelos, N. Lazaar, N. Narodytska, C. Quimper, and T. Walsh. Constraint acquisition via partial queries. In F. Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 475–481. IJCAI/AAAI, 2013. URL http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6659.

[5] C. Bessiere, R. Coletta, A. Daoudi, N. Lazaar, Y. Mechqrane, and E. Bouyakhf. Boosting constraint acquisition via generalization queries. In T. Schaub, G. Friedrich, and B. O'Sullivan, editors, *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 99–104. IOS Press, 2014. doi: 10.3233/978-1-61499-419-0-99. URL https://doi.org/10.3233/978-1-61499-419-0-99.

[6] C. Bessiere, R. Coletta, and N. Lazaar. Solve a constraint problem without modeling it. In *26th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2014, Limassol, Cyprus, November 10-12, 2014*, pages 1–7. IEEE Computer Society, 2014. doi: 10.1109/ICTAI.2014.12. URL https://doi.org/10.1109/ICTAI.2014.12.

[7] C. Bessiere, F. Koriche, N. Lazaar, and B. O'Sullivan. Constraint acquisition. *Artif. Intell.*, 244:315–342, 2017. doi: 10.1016/J.ARTINT.2015.08.001. URL https://doi.org/10.1016/j.artint.2015.08.001.

[8] A. Daoudi, Y. Mechqrane, C. Bessiere, N. Lazaar, and E. Bouyakhf. Constraint acquisition with recommendation queries. In S. Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 720–726. IJCAI/AAAI Press, 2016. URL http://www.ijcai.org/Abstract/16/108.

[9] E. C. Freuder and R. J. Wallace. Suggestion strategies for constraint-based matchmaker agents. In E. C. Freuder, editor, *Constraints & Agents: Collected Papers from the 1997 AAAI Workshop, Providence, RI, USA, July 27, 1997*, pages 105–111. AAAI Press, 1997.

[10] C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, 1969. doi: 10.1145/363235.363259. URL https://doi.org/10.1145/363235.363259.

[11] G. Menguy, S. Bardin, N. Lazaar, and A. Gotlieb. Automated program analysis: Revisiting precondition inference through constraint acquisition. In L. D. Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 1873–1879. ijcai.org, 2022. doi: 10.24963/IJCAI.2022/260. URL https://doi.org/10.24963/ijcai.2022/260.

[12] G. Menguy, S. Bardin, N. Lazaar, and A. Gotlieb. Active disjunctive constraint acquisition. In P. Marquis, T. C. Son, and G. Kern-Isberner, editors, *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Rhodes, Greece, September 2-8, 2023*, pages 512–520, 2023. doi: 10.24963/KR.2023/50. URL https://doi.org/10.24963/kr.2023/50.

[13] K. M. Shchekotykhin and G. Friedrich. Argumentation based constraint acquisition. In W. Wang, H. Kargupta, S. Ranka, P. S. Yu, and X. Wu, editors, *ICDM 2009, The Ninth IEEE International Conference on Data Mining, Miami, Florida, USA, 6-9 December 2009*, pages 476–482. IEEE Computer Society, 2009. doi: 10.1109/ICDM.2009.62. URL https://doi.org/10.1109/ICDM.2009.62.

[14] D. C. Tsouros, S. Berden, and T. Guns. Guided bottom-up interactive constraint acquisition. *CoRR*, abs/2307.06126, 2023. doi: 10.48550/ARXIV.2307.06126. URL https://doi.org/10.48550/arXiv.2307.06126.