Toward Practical Constraint Acquisition: An Anytime Learning Approach

Aicha BOUKHARI*

Université Paris-Saclay, CEA, List, Palaiseau, France
Université Paris-Saclay, LISN, Gif-sur-Yvette, France

Abstract. Constraint Acquisition (CA) is an interactive learning framework where the goal is to infer a target concept, typically a set of constraints, by interacting with a user through queries. Most existing CA approaches aim to converge to the exact target concept. However, convergence in CA has been shown to be coNP-complete, making it computationally hard and often impractical when time or query budgets are limited. In my PhD research, I focus on anytime Constraint Acquisition, a learning paradigm that relaxes the convergence requirement and instead seeks to produce the best possible approximation of the target concept within a given resource budget — e.g., time, number of queries. The key idea is to ensure that, even if the acquisition process is interrupted prematurely, the returned hypothesis remains as close as possible to the target. To this end, we propose a novel query generation strategy, called ΔGEN, which prioritizes queries based on a distance metric Δ, designed to accelerate the inference of the target concept approximation. This strategy is compatible with standard CA learners such as CONACQ and DCA. We implemented it within DCA and evaluated its impact on a practical usecase: precondition inference for program verification. Preliminary results show that ΔGEN significantly improves the quality of learned concepts, generating up to three times more useful queries compared to default heuristics. This suggests that ΔGEN offers a promising path for scalable and robust CA in real-world applications where convergence is often out of reach.

1 Introduction

Many CA methods have been proposed in the literature to assist users in modeling problems using a constraint language. These methods rely on queries classified by the user or an oracle to guide the acquisition process. In active learning CA, queries are automatically generated by the algorithm itself. In this work, we focus exclusively on active learning methods that use membership queries. Specifically, we consider CONACQ [2] for learning concepts representable as a conjunction of constraints from a given constraint language, and DCA [6] for concepts expressible as a disjunction of such constraints.

Problem. Despite significant progress, CA still struggles with complex problems. My PhD work focuses on improving CA for broader domains like program analysis, specifically precondition inference [5, 6], where accurate and timely models are key. Indeed, in hard cases, CONACQ and DCA often fail to converge due to time or query budgets. Still, in some contexts like program analysis, convergence is not mandatory, and extracting the Maximally Specific concept (MS) after some iterations is acceptable. Unfortunately, even here, the extracted concept tends to lack precision. Indeed, current CA methods do not generate enough informative positive queries (i.e., set of solutions), which is essential for learning precise MS.

Contribution. This paper makes the following contributions:

1. We propose ΔGEN, the first extension in CA to improve MS under a time budget constraint, by guiding query generation over potential positive queries. ΔGEN is a new query generation method that enables fast anytime constraint acquisition.
2. We instantiated ΔGEN within the DCA algorithm leading to ΔDCAGEN. Section 5 evaluates ΔDCAGEN on precondition inference problems. Results show that, thanks to our guidance, ΔDCAGEN generates 5 times more positive queries than the baseline, hence generating more precise MS.

2 Background

Constraint Acquisition (CA). Modeling combinatorial problems can be challenging and requires a significant effort from users. CA automates this process by learning a constrained-based model through interactions with the user. It relies on a predefined bias, which is a set of candidate constraints. The goal is to learn the target concept, expressing it with constraints from the bias. Throughout the inference CA refines the version space, which contains all concepts consistent with the information obtained from the interactions with the user. Initially, the version space is bounded by two extremes: the Maximally General (MG) concept, which accepts all possible assignments (i.e., $MG = true$), and the Maximally Specific (MS) concept, which rejects all assignments (i.e., $MS = false$). The acquisition process aims to generalize MS and specify MG until both boundaries converge, resulting in the learned model. This refinement is guided by interactions with a user (or an oracle), who is asked to classify membership queries, i.e., a complete assignments to the variables. A query is classified as positive if it satisfies the target concept, and negative otherwise. Based on these classifications, the learner updates the version space by inferring which constraints in the bias belong or not to the target concept. Especially, if a query is positive, MS is generalized and MG is specified otherwise. The queries used are informative, meaning that each one helps reduce the version space by eliminating inconsistent concepts, i.e., those that contradict the user's answers.

* Email: aicha.boukhari@cea.fr

CA for Precondition Inference. Given a function under analysis F, and a postcondition Q, the precondition inference problem aims to infer a condition P on F inputs, called the precondition, ensuring that the execution terminates, does not crash and returns an output verifying the postcondition Q [3]. Recently, the PRECA framework proposed to use CA to automatically infer preconditions [5, 6, 7]. It replaces the user by an oracle that executes the binary code of the program under analysis — thus operating in an automated and black-box manner. PRECA infers the weakest precondition (WP), i.e., the most general one, accepting the most models. Interestingly, when applied on precondition inference, the Most Specific concept (MS) from CA algorithms corresponds to a correct precondition, i.e., one implying the WP. In program verification, recovering a precise precondition is acceptable as the weakest precondition is often too complex to infer or even to understand by a human-user.

3 Motivation

While CA made significant progress, convergence is still out-of-reach on complex problems. For instance, over a benchmark gathering functions from the libc, the mbedtls cryptographic library, and the tinyUSB library, PRECA [5] timeouts for 16 / 70 functions within a 1h time budget. Even worst, on some cases like the $mbedtls\_md\_hmac\_starts$ function from the mbedtls cryptography library, the extracted MS after 1h of acquisition equals *false* — i.e., the worst possible precondition. This is due to the complete absence of positive queries during the acquisition.

To effectively generalize the Most Specific concept (MS), it is necessary to maximize the number of positive queries generated. Indeed, only positive queries help generalizing the MS.

Our proposal. In the following, we propose ΔGEN to quickly generalize the MS. It relies on the following intuition:

Intuition 1. The more two queries satisfy common constraints, the more likely they will share the same classification.

The observation is supported by statistics collected on 40 precondition inference problems where PRECA converges and could generate all the needed queries. The analysis shows that, on average, 84% of the queries that share the maximum number of constraints with a given positive query e are also classified as positive. Especially, over the $mbedtls\_md\_hmac\_starts$ function, ΔGEN enables to generate 2194 positive queries.

4 ΔGEN approach

We propose a new query generation heuristic based on Intuition 1. It aims to generate queries satisfying the maximum number of constraints in common with already generated positive queries. To do so, we solve an optimization problem based on a dedicated metric Δ.

Definition 1 (the Δ metric)**.** *Given two membership queries e and e', and a bias B, we define $\Delta(e, e')$ as the number of constraints in B that are satisfied by one query and violated by the other :*

$$\Delta(e, e') = |\{c \in B \mid (e \models c) \Leftrightarrow (e' \not\models c)\}|$$

Given a set of already generated queries E and a positive query $e \in E$, the optimization problem consists in generating a new informative query e' that minimizes $\Delta(e, e')$. In practice, e' is generated through a maxsat formula that aims to maximize the number of constraints satisfied by e while ensuring the informativeness of e'.

Since we aim to generalize the MS, we need to minimize Δ with respect to a positive query. Unfortunately, finding an initial positive query automatically is a hard problem — e.g., PRECA often timeouts after 1h without producing any positive query. Hence, we assume the user can provide it as input for the algorithm. While it may be a strong hypothesis in some contexts, in program analysis, this is a legitimate assumption since developpers often provide such inputs either in the documentation or unit tests.

Table 1. ΔDCAGEN against DCAGEN: number of positive queries out of the total number of queries generated (TO=1h)

Task	DCAGEN	ΔDCAGEN
starts	144/1241	**1105**/1214
cbc	480/889	**1545**/1838
process	62/601	**881**/1326
finish	22/451	**179**/187
reset	59/1245	**264**/522
update	78/564	**553**/578
strtok_r	405/1452	**607**/2282
setup	1014/4797	**4871**/5176

5 Experimental Evaluation

Experimental setup. We evaluate ΔGEN in the context of DCA acquisition, which was shown more efficient for precondition inference [6]. Still, in its basic shape DCA cannot generate MS concepts. For fair comparison, we applied a simple extension of DCA to generate MS, leading to DCAGEN. We then implemented our approach within DCAGEN, which we refer to as ΔDCAGEN. DCAGEN and ΔDCAGEN are written in Java and use SAT4J [1] as SAT and MaxSAT solver, and Choco-solver [8] as CP solver. Evaluation is performed on precondition inference tasks (functions from the libc and mbedtls cryptography library [6]) where the original DCA fails to converge within 1h.

Results. Table 1 compares the number of positive and total queries generated by DCAGEN and ΔDCAGEN. For a timeout of 1h, ΔDCAGEN significantly outperforms DCAGEN, generating on average five times more positive queries. Further analysis showed that ΔDCAGEN consistently produces more positive queries for time budgets $\geq$ 15 mins. Still, for timeouts around 5 mins, there are cases where DCAGEN generates more positive queries than ΔDCAGEN.

Conclusion. On complex tasks with enough time budgets, typically more than 5 mins, ΔDCAGEN significantly outperforms DCAGEN.

6 Conclusion

I propose ΔGEN, a guided query generation based on distance between queries, that prioritizes the generation of positive queries. It enables the acquisition to infer a precise enough concept at anytime. The results are encouraging, ΔGEN generating 5 times more positive queries than the state-of-the-art. As future work, we plan to extend the evaluation of ΔGEN. Especially, even if our approach significantly improves positive query generation, it remains difficult to compare the precision of the resulting concepts. Indeed, the queries generated by ΔGEN and competitors differ significantly and logical implication does not occur. Hence, we plan to use model counting techniques [4] to estimate the number of solutions in acquisition results and better evaluate precision. We also plan to apply ΔGEN within the CONACQ algorithm.

References

[1] D. L. Berre and A. Parrain. The sat4j library, release 2.2. *J. Satisf. Boolean Model. Comput.*, 7(2-3):59–6, 2010. doi: 10.3233/SAT190075. URL https://doi.org/10.3233/sat190075.

[2] 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.

[3] 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.

[4] L. Luu, S. Shinde, P. Saxena, and B. Demsky. A model counter for constraints over unbounded strings. In M. F. P. O'Boyle and K. Pingali, editors, *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '14, Edinburgh, United Kingdom - June 09 - 11, 2014*, pages 565–576. ACM, 2014. doi: 10.1145/2594291.2594331. URL https://doi.org/10.1145/2594291.2594331.

[5] 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.

[6] 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.

[7] G. Menguy, S. Bardin, A. Gotlieb, and N. Lazaar. A query-based constraint acquisition approach for enhanced precision in program precondition inference. *J. Artif. Intell. Res.*, 82:901–936, 2025. doi: 10.1613/JAIR.1.16206. URL https://doi.org/10.1613/jair.1.16206.

[8] C. Prud'homme and J. Fages. Choco-solver: A java library for constraint programming. *J. Open Source Softw.*, 7(78):4708, 2022. doi: 10.21105/JOSS.04708. URL https://doi.org/10.21105/joss.04708.