A Non-monotonic Theory of Oracle-guided Inductive Synthesis

Dalal Alrajeh[1], Susmit Jha[2], and Sanjit Seshia[3]

[1] Imperial College London, UK
dalal.alrajeh@ic.ac.uk
[2] Computer Science Laboratory, SRI International, USA
jha@csl.sri.com
[3] EECS, UC Berkeley, USA
sseshia@eecs.berkeley.edu

Specifications provide significant aid in the formal analysis of software supporting tasks such as consistency management, system verification, program synthesis, program repair and software maintenance. However writing such specifications is difficult and time-consuming. Several approaches have been proposed for automatically generating complete specifications from abstract descriptions (such as UML diagrams and user requirements) which mainly differ in their method of computation, e.g., refinement operators [7], patterns library [6]. Recent years have seen the emergence of techniques based on inductive learning, for instance [4]. The input to these techniques are samples classified as either positive or negative examples. The aim is to learn a specification that is consistent with all positive examples and inconsistent with all the negative ones. However, the quality of a learnt specification (and its proximity to the target specification) heavily depends on the quality of the samples provided to the learner.

Oracle-guided inductive synthesis (OGIS) is a class of approaches that restrict the set of samples for learning to those directly relevant to some target specification [5]. The learner can query an oracle (e.g., a user or verifier) for examples. The oracle in return responds with positive or negative example that is intended to guide the learner's search for candidate specifications. The oracle is also tasked with determining whether the learner has found the correct specification. Each time a negative (resp. positive) example that is consistent (resp. inconsistent) with the current candidate specification is given, the learner proceeds in one of the following directions: (1) the candidate specification is discarded and a new one is synthesized from the set of examples accumulated thus far; or (2) an additional specification is synthesized from the new example and added to the previous ones; we say in the second case a specification has been refined. Both directions may lead to learning a correct specification (consistent with all positive examples but none of the negatives). We argue that the former, from a practical perspective, requires users to abandon any development activities they may have started based on the earlier specification. From a conceptual view, it violates the principle of elaboration tolerance [2, 8]. The applicability of the latter, however, depends on the kind of inference allowed by the learner. Typical OGIS instances that follow this direction assume monotonicity of the synthesis procedure. By this, it is guaranteed that candidate specifications

generated from new examples are consistent with those of previous iterations. However, this guarantee does not hold in the case of non-monotonic learners.

In this work, we conduct a formal investigation into properties of oracle-guided inductive synthesis for specification refinement by examining the impact of using two types of learners: monotonic and non-monotonic [3]. In monotonic learning, inferences cannot be invalidated simply by adding new expressions to a specification. Non-monotonic learning, on the other hand, allows inferences to be made provisionally which can be retracted as new information becomes available and thus specifications are extended. Our investigation seeks to answer the following questions: *Does the quality of a specification improve with a non-monotonic learner? What are the termination guarantees with non-monotonic learner? In cases where termination is guaranteed, does the use of a non-monotonic learner improve the speed of termination?* In answering these, we assume an oracle that defines a fixed ordering over the examples generated and seek to understand the influence of the following factors: (*i*) the types of queries submitted to an oracle (e.g., correctness and whether they provide both positive and negative examples or positive witness only); and (*ii*) the resources available to the learner: finite versus infinite memory. We direct our analysis to a particular instance of OGIS for synthesizing target specifications in Linear Temporal Logic (LTL) [9]. The quality of a specification is measured in terms of: the size of the formulas, and the size of the language defined by the formulas. For monotonic learning, we have developed a monotonic learner that can compute properties in tight Signal Temporal Logic (a flavour of LTL over continuous signals) from positive examples only. As for non-monotonic learning, we consider the approach described in [1] which provides a transformation function from a subclass of LTL to a non-monotonic logic and vice-versa.

References

1. Alrajeh, D., Ray, O., Russo, A., Uchitel, S.: Using abduction and induction for operational requirements elaboration. J. Applied Logic 7(3), 275–288 (2009)
2. Baral, C., Zhao, J.: Non-monotonic temporal logics for goal specification. In: Proceedings of IJCAI07,. pp. 236–242 (2007)
3. Frankish, K.: Non-monotonic inference. In: Barber, A. (ed.) Encyclopedia of Language and Linguistics. Elsevier (2005)
4. Gehr, T., Dimitrov, D., Vechev, M.: Learning commutativity specifications. In: Proceedings of CAV15. pp. 307–323 (2015)
5. Jha, S., Seshia, S.: A theory of formal synthesis via inductive learning. Acta Informatica (2016)
6. van Lamsweerde, A.: Requirements Engineering - From System Goals to UML Models to Software Specifications. Wiley (2009)
7. Li, F.L., Horkoff, J., Borgida, A., Guizzardi, G., Liu, L., Mylopoulos, J.: From stakeholder requirements to formal specifications through refinement. In: Proceedings of REFSQ15 (2015)
8. McCarthy, J.: The artificial intelligence debate: False starts, real foundations. chap. Mathematical Logic in Artificial Intelligence, pp. 297–311. MIT Press (1988)
9. Pnueli, A.: The temporal logic of programs. In: Proceedings of SFCS77. pp. 46–57 (1977)