Heuristic-Based GR(1) Assumptions Refinement

Davide G. Cavezza

Imperial College London, United Kingdom
d.cavezza15@imperial.ac.uk

Abstract. In order to synthesize automatically a controller satisfying a specification given in GR(1) (a subset of linear temporal logic), the environment, where the controller is expected to operate, needs to be characterized by a sufficient set of GR(1) assumptions. Assumptions refinement procedures identify alternative sets of assumptions that make controller synthesis possible. However, since assumptions spaces are intractably large, techniques to explore a subset of them in a guided fashion are needed. In particular, it is important to identify weakest assumptions refinements to avoid overconstraining the environments and hence deeming the controller to be inadequate.
The objective of my research is to devise a heuristic search approach that uses estimates of goodness of explored assumptions to direct the search towards better solutions. The work involves defining computable metrics that capture quality features of assumptions (such as their weakness), and automated ways to select a good subset of refinements in the search procedure.

Background

Generalized reactivity of rank 1 (GR(1) for short) is a subset of linear temporal logic (LTL) largely used in specifying software systems, allowing to describe safety and liveness properties of autonomous systems [8] and on-chip communication protocols [2]. Specifications in this subset are of particular interest in the reactive synthesis community for the feasibility of automated controller synthesis: given a GR(1) specification, it is possible to produce automatically an implementation that satisfies it [2].

A model of a GR(1) specification consists of two interacting agents, an environment and a controller, playing an adversarial game: the controller acts so as to satisfy a set of guarantees, and the environment aims at violating one of the guarantees whilst satisfying a set of assumptions. Both guarantees and assumptions are conjunctions of initial conditions (pure Boolean formulae, like $\neg grant$, where $grant$ is a Boolean variable), invariants (LTL formulae like $\mathbf{G}(request \rightarrow \mathbf{X}grant)$), and fairness conditions (e.g. $\mathbf{GF}valid$). Automated controller synthesis consists in computing a winning strategy for the controller.

In order to make synthesis possible, a winning controller strategy must exist; if not, the GR(1) specification is said to be *unrealizable*. Unrealizability is generally related with an insufficient characterization of environment assumptions. *Assumption refinement* [5,7,1] consists in computing one or more sets

of assumptions that constitute a sufficient restriction to the environment's allowed strategies: this restriction aims at excluding those behaviors that make the game unwinnable by the controller, called *counterstrategies*. A simple counterstrategy of the specification with assumption $\mathbf{GF}\neg req$ and guarantees $\mathbf{G}(cl \rightarrow \neg val) \wedge \mathbf{GF}(gr \wedge val)$ is given in Fig. 1.

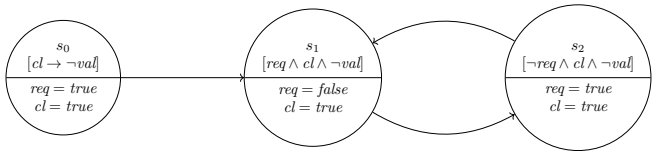

Fig. 1. A counterstrategy. The top half of the circle contains the state's name and a Boolean expression representing a condition being true at that state; the bottom half contains the valuations chosen by the environment at the next step of the computation.

Existing refinement approaches rely on automated counterstrategy computation [6] for guiding the search towards assumptions that make the specification realizable. Given an unrealizable specification, a counterstrategy is computed and a set of alternative assumptions that rule out such counterstrategy is generated; for each new assumption, realizability is checked again and in case the specification is still unrealizable, a new counterstrategy is computed in an iterative fashion. The process is iterated until one or more alternative realizable refinements are computed, or some computation budget is exhausted. The goal is to obtain a set of *weakest* (that is, most permissive) refinements that ensure the realizability of a specification [10].

Motivation

One of the most challenging problems in this field is dealing with the size of the assumptions space. Given a GR(1) specification over n Boolean variables, there are $O(2^{2^n})$ different assumptions (modulo logical equivalence) that may need to be searched in order to find the weakest ones. Since a full exploration of this space is infeasible, only a subset of it is explored. In general, this is tackled by requesting the user to provide one or more assumption templates, which are then instantiated and only those instances get explored.

The main problem with these templates is that they rely on the user's prior knowledge of what causes unrealizability in a specification. A user-defined template may force the procedure to overconstrain the environment by missing weaker solutions, or the procedure may fail to find solutions at all. Hence there is need for automated approaches that explore the search space in an intelligent fashion, by leveraging additional information besides counterstrategies.

Contribution

Our goal is to devise a search strategy that learns upon which regions of the search space should focus while looking for a solution. The work proceeds along

these three directions: *(i)* devising refinement techniques that select candidate assumptions in a fully automated fashion, relieving users from the need of prior knowledge; *(ii)* defining domain-independent quality metrics for assumption refinements, so as to allow a fair comparison between alternative assumptions generated by some technique or between techniques themselves; *(iii)* using quality metrics to redefine the GR(1) assumptions refinement problem as a heuristic-based search of the assumption space.

User-independent refinement techniques. Our work in [3] aims at making the refinement procedure independent of user knowledge. It exploits Craig interpolation to bypass the template selection step.

Craig interpolation [9] is an automated procedure that, given two logical expressions ϕ_1 and ϕ_2 unsatisfiable together, returns a third expression ϕ_i, called *interpolant*, that is implied by ϕ_1, implies $\neg\phi_2$, and only uses Boolean variables that are common between ϕ_1 and ϕ_2. An interpolant is usually interpreted as a logical expression explaining the reason why ϕ_1 causes the violation of ϕ_2. In our approach, ϕ_1 is a description of a counterstrategy leading to a guarantee violation, and ϕ_2 describes the violated guarantee; interpolants between them are used as a basis to generate assumption refinements that eliminate undesired environment behaviors. As an example, for the counterstrategy in Fig. 1 a Boolean description of it is interpolated with the negation of the guarantees given above, yielding the interpolant $cl(s_1) \wedge cl(s_2)$; this says that the cause of unrealizability lies in keeping the cl variable always $true$ in the looping states; a way to prevent that is assuming $\mathbf{GF}\neg cl$, which is the inferred refinement.

Domain-independent quality metrics. The state of the art lacks metrics to quantify the quality of produced specifications. This hinders a fair comparison between approaches when their output differs with each other. Therefore, part of our work has been focused on identifying reasonable domain-independent quality metrics for GR(1) assumptions.

Since the weakness of assumptions is a common concern in assumptions refinement, we looked into ways for measuring this feature. In general, weakness is defined as a non-quantitative feature connected to whether or not some formula implies another: that is, ϕ_1 is weaker than ϕ_2 if ϕ_2 implies ϕ_1; no weakness notion can be defined when the two formulae do not imply each other. Despite that, there are cases in which a formula can be considered less restrictive than another even if implication does not hold. To justify that, we consider the relationship between linear temporal logic and infinite-word automata [12]: an infinite-word automaton summarizes all behaviors that satisfy a given LTL assumption; in some sense, the more paths in the automaton, the weaker the LTL assumption.

In recent work [4], we give theoretical ground to a new weakness measure based on Hausdorff dimension [11]. We study the relationship between implication and Hausdorff dimension, and identify cases when Hausdorff dimension is more or less discriminative than implication for comparing assumptions. We observe on a benchmark that our measure discriminates more than implication, providing a ranking of alternative assumptions even when implication does not hold.

We are investigating other features for comparing assumptions, such as their readability. For this we are considering metrics such as formula length and maximum logical operator nesting.

Heuristic search. Existing refinement procedures are incremental [7,1,3]: given an initial unrealizable specification, containing an initial set of assumptions (possibly empty) and a set of guarantees, it first finds a set of alternative assumptions that refine the initial ones, and then for each refinement it checks whether realizability is achieved; if not, the new assumption is further refined. In this way, a *refinement tree* is built, which contains partial refinements as internal nodes and realizable refinements as leaves.

Our investigation is focused on finding a good heuristic for internal nodes of the tree, in order to explore them in decreasing order of goodness. The heuristic needs to take into account the tradeoff between the final goal (finding weakest assumptions that achieve realizability) and the expected number of further steps needed to achieve a realizable solution: in general, the weaker a partial refinement, the higher this number. In order to account for weakness, the heuristic may embed the weakness measure defined as above. We are investigating ways to estimate distance from realizability accurately.

Acknowledgments The support of the EPSRC HiPEDS CDT (EP/L016796/1) is gratefully acknowledged.

References

1. Alur, R., Moarref, S., Topcu, U.: Counter-Strategy Guided Refinement of GR(1) Temporal Logic Specifications. In: FMCAD. pp. 26–33. No. 1, IEEE (2013)
2. Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A., Sa'Ar, Y.: Synthesis of Reactive(1) designs. Journal of Computer and System Sciences 78(3), 911–938 (2012)
3. Cavezza, D.G., Alrajeh, D.: Interpolation-Based GR(1) Assumptions Refinement. In: TACAS. pp. 281–297. Springer Berlin Heidelberg (2017)
4. Cavezza, D.G., Alrajeh, D., György, A.: A weakness measure for gr(1) formulae To appear in FM 2018
5. Chatterjee, K., Henzinger, T.A., Jobstmann, B.: Environment Assumptions for Synthesis. In: CONCUR, pp. 147–161. Springer Berlin Heidelberg (2008)
6. Konighofer, R., Hofferek, G., Bloem, R.: Debugging formal specifications using simple counterstrategies. In: FMCAD. pp. 152–159. IEEE (2009)
7. Li, W., Dworkin, L., Seshia, S.A.: Mining assumptions for synthesis. In: MEMOCODE. pp. 43–50. ACM/IEEE (2011)
8. Li, W., Sadigh, D., Sastry, S.S., Seshia, S.A.: Synthesis for Human-in-the-Loop Control Systems. In: TACAS. pp. 470–484. Springer Berlin Heidelberg (2014)
9. McMillan, K.L.: Interpolation and SAT-Based Model Checking. In: CAV. pp. 1–13 (2003)
10. Seshia, S.A.: Combining Induction, Deduction, and Structure for Verification and Synthesis. Proceedings of the IEEE 103(11), 2036–2051 (2015)
11. Staiger, L.: On the Hausdorff measure of regular omega-languages in Cantor space. Tech. Rep. 1 (2015)
12. Vardi, M.Y.: An automata-theoretic approach to linear temporal logic. Logics for concurrency pp. 238–266 (1996)