Proceedings of the Sixteenth International Conference on
Principles of Knowledge Representation and Reasoning (KR 2018)

Synthesis under Assumptions

Benjamin Aminof
Technische Univ. Wien
Vienna, Austria
aminof@forsyte.at

Giuseppe De Giacomo
Sapienza Univ. Roma
Rome, Italy
degiacomo@dis.uniroma1.it

Aniello Murano
Univ. Federico II
Naples, Italy
murano@unina.it

Sasha Rubin
Univ. Federico II
Naples, Italy
rubin@unina.it

Abstract

We consider the following elusive question which has generated a lot of ad hoc research: *In synthesis/planning, which constraints on traces are environment assumptions?* We propose to view assumptions as sets of strategies and not, as is typical, as sets of traces, even for assumptions expressed as linear-temporal formulas on traces. This shift in perspective allows us to give a principled and conceptually clear answer to this question, as well as to reuse results and techniques for synthesis in order to solve synthesis under assumptions.

Context. Reasoning about actions and planning concerns the representation of a dynamic system (Reiter 2001). Such a representation corresponds to knowledge that the agent has of the environment. In devising plans, the agent takes advantage of such a representation of the world. In other words, the agent assumes that the environment works in a certain way, and exploits such an assumption in devising its plans. A question immediately comes to mind:

What is an environment assumption?

Obviously the planning domain itself (including the initial state) with its preconditions and effects is an environment assumption. That is, as long as the agent sticks to its preconditions, the environment acts as described by the domain. So, the agent can exploit the effect of its actions in order to reach a certain goal (state of affairs). It is also common to assume the domain is *fair*, cf. (D'Ippolito, Rodríguez, and Sardiña 2018), or more generally that the environment's behavior is restricted by trajectory constraints, often expressed in LTL, cf. (De Giacomo et al. 2016; Bonet et al. 2017). But, is any kind of formula on the fluents and actions of the domain a possible trajectory constraint for the environment? The answer is obviously "no" — indeed, consider a formula expressing that eventually a certain possible action must actually be performed — the agent may decide not to do it. But then we ask:

Which linear-time specifications are environment assumptions?

For instance, a common formalization is to let the assumption be any LTL formula φ, the goal be any LTL formula γ, and to have the agent synthesize a strategy for the implication $\varphi \supset \gamma$. A well-known problem synthesizing for such an implication is that the agent may devise strategies that make φ false. This is undesirable since, by doing this, the agent loses its model of the environment and need not fulfill its goal γ. Overcoming this problem has received considerable attention in the formal methods literature, e.g., (Bloem et al. 2014; Brenguier, Raskin, and Sankur 2017). This problem, and many of the approaches for tackling it, are completely avoided by our definition of environment assumption. Indeed, we propose to view environment assumptions, even when expressed by linear-temporal formulas on traces, as sets of *strategies* the environment can enact, and an agent that achieves its goal only needs to do so against all the environment strategies from this set. We observe that typical environment assumptions, such as the domain specifications themselves or fairness, can be thought of as sets of environment strategies.

We instantiate this idea in the context of Reactive Synthesis, i.e., the problem of producing a module that satisfies a given temporal property no matter how the environment behaves (Pnueli and Rosner 1989). Synthesis can be thought of as a game between two players, the agent and the environment. Each player controls its own set of Boolean variables, i.e., A is controlled by the agent, and E is controlled by the environment. In each phase of the game, both players assign values to their variables, with the environment going first. Thus, in every phase, the environment picks an element from $\mathcal{E} = 2^E$, the agent picks an element from $\mathcal{A} = 2^A$.[1] The game consists of infinitely many phases, and results in a trace, i.e., an infinite string over alphabet $\mathcal{E} \cup \mathcal{A}$.

The object that captures which assignment an agent plays in which situation is called a strategy (these are similar to policies in planning): an agent strategy is a function $\sigma_{\mathsf{ag}} : \mathcal{E}^+ \to \mathcal{A}$ associating an agent move to each nonempty finite sequence of environment moves, and an environment strategy is a function $\sigma_{\mathsf{env}} : \mathcal{A}^* \to \mathcal{E}$ associating an environment move to each possibly empty finite sequence of agent moves. The trace generated by these strategies is denoted $\pi_{\sigma_{\mathsf{ag}},\sigma_{\mathsf{env}}}$. In classic synthesis the agent is trying to ensure that the produced trace satisfies a given linear-time property φ, expressed, say, in LTL. Formally, σ_{ag} *re-*

[1]One may think of E as a set of fluents, and $\mathcal{A}$ as a set of actions that are compactly represented as assignments of the variables in A.

alizes φ if $\forall\sigma_{\mathsf{env}}(\pi_{\sigma_{\mathsf{ag}},\sigma_{\mathsf{env}}} \models \varphi)$. Similarly, we say that σ_{env} *realizes* φ if $\forall\sigma_{\mathsf{ag}}(\pi_{\sigma_{\mathsf{ag}},\sigma_{\mathsf{env}}} \models \varphi)$. *Solving environment- (resp. agent-) synthesis* asks, given φ, to decide if φ is environment- (resp. agent-) realizable, and to return such a finite-state strategy, if one exists. In other words, realizability is the recognition problem associated to synthesis. If φ is expressed in LTL or LTLf, such problems are known to be 2EXPTIME-complete (Pnueli and Rosner 1989; De Giacomo and Vardi 2015).

Using the notions of environment and agent strategies, we give definitions of environment assumptions and synthesis under such assumptions as follows.

Definition 1 (Environment Assumptions)**.** *An* environment assumption *is a non-empty set* Ω *of environment strategies.*

Given an *agent goal*, i.e., a set Γ of desirable traces, the agent considers that the environment will follow a strategy from Ω (but does not know which one), and is trying to ensure the trace is in the set Γ.

Definition 2 (Synthesis under assumptions)**.** *An agent strategy* σ_{ag} realizes Γ under the assumption Ω *if*

$$\forall\sigma_{\mathsf{env}} \in \Omega.\, \pi_{\sigma_{\mathsf{ag}},\sigma_{\mathsf{env}}} \in \Gamma.$$

Note that the requirement that Ω be non-empty is a consistency requirement; if it were empty then there would be no $\pi_{\sigma_{\mathsf{ag}},\sigma_{\mathsf{env}}}$ to test for membership in Γ and so synthesis under assumptions would trivialize and all agent strategies would realize all goals. These definitions are purposefully abstract. They beg the questions: *How should one represent environment assumptions and agent goals?* and *Given such a representation, how can one solve synthesis under assumptions?* To answer these questions we observe that strategies have natural codings as trees, and thus one may use any declarative specification whose models are sets of trees, e.g., CTL^*, FOL, MSOL, tree-automata. We leave this for future work.

Here, instead, we turn to *linear-time specification languages* such as LTL. Every linear-time formula φ induces the set Ω_φ consisting of all environment strategies σ_{env} such σ_{env} realizes φ. In particular, for the consistency requirement of Ω_φ being a non-empty set of strategies we must have that φ is *environment realizable* (itself a decidable property, as mentioned above). For example, in robot-action planning problems, typical environment assumptions encode the physical space, e.g., "if robot is in Room 1 and does action $Move$ then in the next step it can only be in Rooms 1 or 4". The set Ω of environment strategies that realize these properties is an environment assumption, definable in LTL by a conjunction of formulas of the form $\mathsf{G}(R_1 \wedge Move \supset \mathsf{X}(R_1 \vee R_4))$. More generally, the set of environment strategies in a planning domain D can be viewed as an environment assumption definable in LTL. One can also formalize fairness as an environment assumption expressible in LTL.

Hence, we concretize the abstract definitions above to assumptions and goals expressible in LTL (similar definitions can be given for other declarative formalisms of linear-temporal properties, such as LTLf) as follows.

Definition 3 (LTL assumptions)**.** *A formula* φ *is an environment assumption (or simply, an assumption) if is environment realizable.*

Definition 4 (LTL synthesis under assumptions)**.** *An agent strategy* σ_{ag} realizes LTL goal γ under the LTL assumption φ *if for all* σ_{env} *that realize* φ *we have that* $\pi_{\sigma_{\mathsf{ag}},\sigma_{\mathsf{env}}} \models \gamma$. *The* LTL synthesis under assumptions problem *is finding one such agent strategy if it exists.*

One may now ask if LTL synthesis under assumptions amounts to solving agent-synthesis for the formula $\varphi \supset \gamma$. The answer is surprisingly subtle. Although every agent strategy that realizes $\varphi \supset \gamma$ also realizes γ assuming φ, the converse is not true. That is, there are agent strategies that realize γ assuming φ but do not realize the formula $\varphi \supset \gamma$. Intuitively, the reason is that an agent that synthesizes for the implication is too pessimistic: the agent, having chosen a candidate agent strategy, considers as possible all environment strategies that satisfy φ against the specific candidate strategy it is analyzing. But, in this way the agent gives too much power to the environment, since, in fact, the environment does not know the agent's chosen strategy. On the other hand, if there exists an agent strategy realizing γ assuming φ then there exists (a possibly different) agent strategy realizing $\varphi \supset \gamma$. Thus, synthesizing for the implication *can be used to solve* the problem of synthesis under assumptions.

Conclusion. The fact that synthesis under assumptions amounts to synthesis of an implication holds only at the level of strategy existence (and not on a per-strategy basis). This fact hinges on subtle reasons, i.e., determinacy of turn-based two-player zero-sum game with LTL objectives, and opens up interesting questions, such as what happens in case determinacy no longer holds, e.g., in synchronous games.

Our proposal to view assumptions as non-empty sets of environment strategies also applies to planning, i.e., one can give a similar definition and treatment of fully-observable nondeterministic (FOND) planning under assumptions.

References

Bloem, R.; Ehlers, R.; Jacobs, S.; and Könighofer, R. 2014. How to handle assumptions in synthesis. In *SYNT*, 34–50.

Bonet, B.; De Giacomo, G.; Geffner, H.; and Rubin, S. 2017. Generalized planning: Non-deterministic abstractions and trajectory constraints. In *IJCAI*, 873–879.

Brenguier, R.; Raskin, J.; and Sankur, O. 2017. Assume-admissible synthesis. *Acta Inf.* 54(1):41–83.

De Giacomo, G., and Vardi, M. Y. 2015. Synthesis for LTL and LDL on finite traces. In *IJCAI*, 1558–1564.

De Giacomo, G.; Murano, A.; Rubin, S.; and Stasio, A. D. 2016. Imperfect-information games and generalized planning. In *IJCAI*, 1037–1043.

D'Ippolito, N.; Rodríguez, N.; and Sardiña, S. 2018. Fully observable non-deterministic planning as assumption-based reactive synthesis. *J. Artif. Intell. Res.* 61:593–621.

Pnueli, A., and Rosner, R. 1989. On the synthesis of a reactive module. In *POPL*, 179–190.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press.