Incorporating Monitors in Reactive Synthesis without Paying the Price (Extended Abstract)

Shaun Azzopardi[0000−0002−2165−3698]

University of Gothenburg, Gothenburg, Sweden
{name.surname}@gu.se

Reactive synthesis has a high complexity, 2EXPTIME, which can make the synthesis of real systems with large complex declarative specifications difficult. However parts of a system may already exist or may be specified in a lower-level modelling language. Synthesis can then instead be applied on the remaining part of the specification. An interesting problem to consider is what level of knowledge of the existing components is required for the new component's synthesis.

We consider pre-existing components specified as rich symbolic automata environment monitors (a lower-level specification language common in the runtime verification community). These automata have a finite-set of states, and a finite-set of variables, and can observe the environment (but not control it). The symbolic transition function is defined between the states, but parametrised by a *guard* on the variable values, and an action that transforms the variable values. The guard also can predicate on propositional environment events. Choosing variable types that are infinite makes the representation exponentially more succinct than finite-state automata, at a cost. These automata have one initial state, and possibly many final states. We call final states *flagging*, since we use them to flag the point at which an LTL formula should be satisfied.

We consider their combination with LTL formulas in a sequential fashion (a monitor triggers the component corresponding to the formula), and/or with repetition (when the synthesised component finishes the monitoring starts again). An interesting observation is that in the case of repetition we require the LTL formula to be co-safety, otherwise it is unclear at which point, if any, should the repetition occur. We then define the language for these combinations as follows, where M is a monitor, ϕ is an LTL formula, and φ is a co-safety LTL formula:

$$\pi = M{:}\phi \mid (M; \varphi)^*.$$

Note that the monitor and LTL formula only share the part of the trace when the monitor flags (and the LTL formula starts holding).

To well-define repetition, we introduce the notion of tight satisfaction of an LTL formula: a co-safety LTL formula is tightly satisfied by a finite trace that satisfies the formula, but does not have a satisfying prefix. We also take care to require, for example, that $X\mathit{true}$ is satisfied only by traces of length two. When considering synthesis of these combinations, we define tight realisability and tight controllers for co-safety LTL formulas, allowing the controllers to signal immediately when a formula has been fulfilled for any given trace.

We analyse these combinations to identify when synthesis of LTL formulas can be used without any knowledge of the monitors. This requires that the

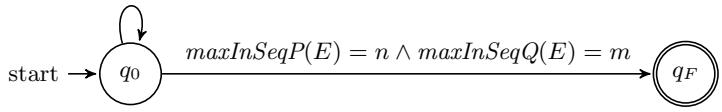

Fig. 1. Event ordering in two buses, where *maxInSeqX* is a function that when $xCount = i$ returns i if E does not contain x_{i+1}, and otherwise returns the maximal j such $x_{i+1}, x_{i+2}, ...$, and x_j are all in E.

underlying assumptions about the environment are stateless, and we then limit them to simple invariants, transition invariants, and recurrence properties. If we allow for transition assumptions one would need to keep track of the current environment state while the monitor is running. We show how any realisable (in general) LTL formula constrained in this way can be synthesised independently of the monitor. Knowledge of the monitor however increases the space of realisable specifications, e.g. given an LTL specification $(b \implies \mathit{false}) \wedge (a \implies c)$ (where a and b are input events, while c is an output event) this LTL specification is realisable in the context of a monitor that only flags upon the event set $\{a\}$.

We give a symbolic procedure to produce a Mealy machine that satisfies the required combination. This procedure can be represented as a symbolic automaton to avoid enumerating the whole state space, allowing for on-the-fly controlling of the particular environment. We have implemented this in a tool syMTri [2], that also allows model checking monitors using nuXmv [1].

This approach has been validated using case studies involving counting and constraints over arbitrarily long sequence of events. For example, given a specification of the form (for $n = 1$) $F(p_0 \wedge F(p_1)) \wedge F(q_0 \wedge F(q_1)) \iff GF\phi$, in SYNTCOMP20 [3] all the tools timed out for $n, m \geq 11$. However, in our approach the practitioner can simply model the problematic part (the left-hand side) for a general n quite concisely using a symbolic automaton (see Fig. 1), and leave the right-hand side for standard LTL synthesis.

This work illustrates how LTL synthesis can be combined with richer more imperative contexts, increasing its scope. It has been carried out with Nir Piterman and Gerardo Schneider, and is to appear in ATVA 2021 [4] (a full version with proofs can be found here [5]).

References

1. nuxmv, https://es-static.fbk.eu/tools/nuxmv/
2. symtri. http://github.com/dSynMa/syMTri
3. Syntcomp20, http://www.syntcomp.org/syntcomp-2020-results/
4. Azzopardi, S., Piterman, N., Schneider, G.: Incorporating monitors in reactive synthesis without paying the price. In: ATVA 2021 (to appear)
5. Azzopardi, S., Piterman, N., Schneider, G.: Incorporating Monitors in Reactive Synthesis without Paying the Price. arXiv e-prints arXiv:2107.00929 (Jul 2021)