Explanations for Unrealizability of Infinite-State Safety Shields

Andoni Rodríguez[1,2] , **Irfansha Shaik**[3,4] , **Davide Corsi**[5] , **Roy Fox**[5] , **César Sánchez**[1]
[1]IMDEA Software Institute, Spain
[2]Universidad Politécnica de Madrid, Spain
[3]Kvantify Aps, Denmark
[4]Department of Computer Science, Aarhus University, Denmark
[6]University of California Irvine, US

Abstract

Safe Reinforcement Learning focuses on developing optimal policies while ensuring safety. A popular method to address such task is *shielding*, in which a correct-by-construction safety component is synthesized from logical specifications. Recently, shield synthesis has been extended to infinite-state domains, such as continuous environments. This makes shielding more applicable to realistic scenarios. However, often shields might be unrealizable because the specification is inconsistent (e.g., contradictory). In order to address this gap, we present a method to obtain simple unconditional and conditional explanations that witness unrealizability, which goes by temporal formula unrolling. In this paper, we show different variants of the technique and its applicability.

1 Introduction

Deep Reinforcement Learning (DRL) has been shown to successfully control reactive systems of high complexity (Marchesini and Farinelli 2020). However, despite their success, DRL controllers can cause even state-of-the-art agents to react unexpectedly (Goodfellow, Shlens, and Szegedy 2014). This issue raises severe concerns regarding the deployment of DRL-based agents in safety-critical reactive systems. Therefore, techniques that rigorously ensure the safe behaviour of DRL-controlled reactive systems have recently been proposed. One of them is *shielding* (Bloem et al. 2015; Alshiekh et al. 2018; Corsi et al. 2025), which incorporates an external component (a "shield") that *enforces* an agent to behave safely according to a given specification φ specified in temporal logic. In this paper, we use post-shielding, where the shield does not interrupt the agent unless it violates a safety constraint.

Regularly, shields are built from specifications using reactive synthesis (Pnueli and Rosner 1989b; Pnueli and Rosner 1989a), in which, given a specification φ, a system is crafted that is guaranteed to satisfy φ for all possible behaviors of its environment. Realizability is the related decision problem of deciding whether such a system exists. This problem is well-studied for specifications written in Linear temporal logic (LTL) (Pnueli 1977; Jacobs et al. 2017). However, many realistic specifications use complex data, whereas LTL is inherently propositional. Alternatively, such realistic specifications can be expressed in LTL modulo theories (LTL$_\mathcal{T}$), which replaces atomic propositions with literals from a first-order theory $\mathcal{T}$ (Geatti, Gianola, and Gigante 2022), whose domain might be infinite (e.g., numbers). In (Rodríguez and Sánchez 2023; Rodríguez and Sánchez 2024b) an LTL$_\mathcal{T}$ specification is translated into an equi-realizable Boolean LTL specification by (1) substituting theory literals by fresh Boolean variables, and (2) computing, using theory reasoning, an additional Boolean formula that captures the dependencies between the new Boolean variables imposed by the literals. This approach is called *Boolean abstraction* and paves the way to produce infinite-state shields, in which the input and output of the shield is no longer Boolean, but belonging to infinite data like numbers (Wu et al. 2019; Corsi et al. 2024; Rodriguez et al. 2025; Kim et al. 2025).

However, it is usually the case that the shield is not realizable, which means that the uncontrollable environment has a way to violate the specification and the shield no longer provides safety guarantees. Thus, we want to understand why this happens: i.e, have an explanation. Recently, the problem of explainability in AI has gained traction, which has had a major impact on the interest in explaining the intricacies of reactive systems (Schewe and Finkbeiner 2007; Baier et al. 2021; Bassan et al. 2023). These approaches focus on either building simple reactive systems, tuning their behaviour or explaining, in each timestep, why the system produces an output to an environment input. However, these approaches (1) have not been used for infinite-state systems and (2) have not addressed the particular problem of understanding shields that are to be synthetised from specifications. Thus, in this paper, we want to open an alternative approach to address explainability: to analyze the LTL$_\mathcal{T}$ specification φ of a shield itself in order to discover why φ is unrealizable (in case it is) via producing a simple witness of an environment strategy such that φ is violated.

Example 1. *Consider the following* LTL$_\mathcal{T}$ *specification:*

$$\varphi = \square[(x < 2) \rightarrow \bigcirc(y > 1)] \wedge [(x \geq 2) \rightarrow (y < x)],$$

where $x, y \in \mathbb{Z}$ and where x is uncontrollable (i.e., belongs to an environment player) and y is controllable (i.e., belongs to an environment player).

Specification φ is unrealizable, which means that the environment player has a strategy to assign valuations to x such that φ is violated. Therefore, it can be argued that we can synthetise (e.g., using (Rodriguez and Sánchez 2024a;

Rodríguez, Gorostiaga, and Sánchez 2024)) the environment strategy for φ and analyze it in order to understand why φ is unrealizable. However, this strategy might be complex to interpret and extract simple explanations (see difficulty in Fig. 3 of App. E). Moreover, this extraction is not automatic and the synthesis procedure might not scale.

In this paper, we address these problems. We propose a general, reusable and an automatic framework to find unrealizability explanations: a bounded unrealizability checking via generating finite *unconditional* and *conditional strategies* as explanations for unrealizable LTL$_\mathcal{T}$ specifications. The idea with (1) unconditional explanations is to provide a finite set of environment assignments that shows why a specification is unrealizable. Alternatively, (2) conditional explanations mean that the environment observes the stateful memory in order to take a decision, which means that the explanation is a function over previous moves. Our general solution works as follows: (1) given φ, we construct an *unrolled* formula φ^k of length k that creates k copies of variables and instances of φ; and (2) we use φ^k to find witnesses of plays that make the environment player reach a violation of φ. This search is described as a logical formula with different quantifier alternations, depending on the conditionality of the strategy sought (meaning that the environment is *conditioned* to a greater or lesser extent on previous moves). If the strategy is completely unconditional, then the formula is quantified with the form $\exists^*\forall^*.\ \varphi$, whereas more conditional strategies involve some quantifier alternations such as $\exists^*\forall^*\exists^*\forall^*.\ \varphi$, which can be understood as split points where the environment *observes* some prefix of the play. Our method explores all possible paths of length k and finds whether there is at least one violation of $\varphi_\mathcal{T}$ that can be achieved by the environment.

The resulting approach is sound, incomplete for unconditional explanations and complete up to k for conditional ones. In summary, the contributions of this paper are as follows:

1. An approach to solve unconditional strategy search of LTL$_\mathcal{T}$ specifications using satisfiability solvers for quantified first-order theory formulae.
2. An alternative method that is based on a combination of a Boolean abstraction, followed by quantified Boolean formulae and a technique to produce values in $\mathcal{T}$. Unlike the first method, this alternative provides termination guarantees in each iteration of method. Both techniques perform an unrolling similar to bounded model checking (Clarke et al. 2001) where we use a k-length unrolling of the original specification.
3. An analogous technique to 1-2 to produce conditional explanations.
4. A case study to discuss the tradeoffs between different explanations and techniques to obtain them.
5. Empirical evaluation showcasing the scalability of the approach.

To the best of our knowledge, this is the first work that focuses on explaining LTL$_\mathcal{T}$ unrealizability and (infinite-state) shield unrealizability.

2 Preliminaries

2.1 Temporal Logic and Synthesis

LTL. We consider LTL (Pnueli 1977; Manna and Pnueli 1995), whose formulae contain atomic propositions, $\wedge$ and $\neg$ (the usual Boolean conjunction and negation[1], respectively), and $\bigcirc$ and $\mathcal{U}$ (the *next* and *until* temporal operators). The semantics of LTL formulae associates traces $\sigma \in \Sigma^\omega$ with LTL formulae as follows:

$$\begin{array}{lll}
\sigma \models \top & & \text{always holds} \\
\sigma \models a & \text{iff} & a \in \sigma(0) \\
\sigma \models \varphi_1 \vee \varphi_2 & \text{iff} & \sigma \models \varphi_1 \text{ or } \sigma \models \varphi_2 \\
\sigma \models \bigcirc\varphi & \text{iff} & \sigma^1 \models \varphi \\
\sigma \models \varphi_1 \,\mathcal{U}\, \varphi_2 & \text{iff} & \text{for some } i \geq 0 \;\; \sigma^i \models \varphi_2, \text{ and} \\
& & \qquad \text{for all } 0 \leq j < i, \sigma^j \models \varphi_1
\end{array}$$

where $\sigma \models \top$ always holds and from which we can also derive common operators like $\wedge$ and $\Box$ (which means *always*), etc. A safety formula φ is such that for every failing trace $\sigma \not\models \varphi$ there is a finite prefix u of σ, such that all σ' extending u also falsify φ (i.e. $\sigma' \not\models \varphi$).

Synthesis. Reactive synthesis is the problem of producing a system from an LTL specification, where the atomic propositions are split into propositions that are controlled by the environment and those that are controlled by the system. Realizability is the related decision problem of deciding whether such a system exists. Realizability corresponds to a turn-based game in a finite arena where, in each turn, the environment produces values of its variables (inputs) and the system responds with values of its variables (outputs).

We revise now conventions of reactive synthesis research: (1) A play is an infinite sequence of turns. (2) The system player wins a play according to an LTL formula φ if the trace of the play satisfies φ. (3) A strategy ρ of a player is a map from positions into a move for the player. (4) A play is played according to ρ if all the moves of the corresponding player are played according to ρ. Also, (5) ρ is winning for a player if all the possible plays played according to ρ are winning. If (6) ρ is winning for the system, the specification is said to be realizable (resp. unrealizable otherwise).

LTL Modulo Theories. A first-order theory $\mathcal{T}$ (Bradley and Manna 2007) consists of a finite set of functions and constants, a set of variables and a domain (which is the sort of its variables). Popular first-order theories are e.g., Presburger arithmetic or real arithmetic.

LTL Modulo Theories (LTL$_\mathcal{T}$) is the extension of LTL where propositions are replaced by literals from a given first-order theory $\mathcal{T}$ (a finite-trace version is given in (Geatti, Gianola, and Gigante 2022) and a infinite-trace version is given in (Rodríguez and Sánchez 2023)). The semantics of LTL$_\mathcal{T}$ associate traces $\sigma \in \Sigma^\omega_\mathcal{T}$ with formulae, where for atomic propositions $\sigma \models l$ holds iff $\sigma(0) \vDash_\mathcal{T} l$, that is, if the valuation $\sigma(0)$ makes the literal l true.

LTL$_\mathcal{T}$ realizability is analogous to LTL realizability, but it corresponds to a game in an arena where positions may have infinitely many successors if ranges of variables are infinite.

[1] We will use $\neg\varphi$ to represent negation of a formula and $\overline{a}$ to represent negation of an atom a.

2.2 Boolean Abstraction for LTL$_\mathcal{T}$

Recently, the field of reactive synthesis beyond Booleans gained traction, and solutions were proposed for different fragments of LTL$_\mathcal{T}$; e.g., very new works like (Heim and Dimitrova 2025; Rodríguez, Gorostiaga, and Sánchez 2025; Azzopardi et al. 2025). In this paper, we want our solution to be complete (up to a bound) and sound.

Therefore, we build upon (Rodríguez and Sánchez 2023), which showed that some fragments of reactive LTL$_\mathcal{T}$ specifications can be translated into equi-realizable purely Boolean LTL specifications via a procedure. The procedure is called Boolean abstraction or Booleanization, and it works as follows: given φ with literals l_i, we get a new specification $\varphi_\mathbb{B} = \varphi[l_i \leftarrow s_i] \wedge \varphi^{\textit{extra}}$, where s_i are fresh Boolean variables, controlled by the system, that replace the literals. The additional subformula $\varphi^{\textit{extra}}$ uses s_i as well as additional Boolean variables e_k controlled by the environment, and captures that, for each possible e_k, the system has the *power* to select a response among a collection of choices, where each choice is a truth valuation of each the variables s_i that represent the literals. That is, a *choice* is a concrete valuation of the s_i variables (and hence of the literals l_i), and a *reaction* is a collection of choices. Pairs $(e_k, \bigvee_i(c_i))$ denote that a decision e_k of the environment can be responded by choosing one of the choices c_i in the disjunction. Then, the set of reactions captures precisely the finite collection of decisions of the environment and the resulting finite responses of the system. The set of valid reactions is determined by the literals in a specific theory $\mathcal{T}$.

Example 2. *Consider φ from Ex. 1, for which the abstraction is as follows: $\varphi_\mathbb{B} = (\varphi'' \wedge \Box[(e_0 \leftrightarrow \overline{e_1}) \wedge (e_0 \leftrightarrow \overline{e_2}) \wedge (e_1 \leftrightarrow \overline{e_2}) \rightarrow \varphi^{\textit{Extra}}])$, where $\varphi'' = (s_0 \rightarrow \bigcirc s_1) \wedge (\overline{s_0} \rightarrow s_2)$ is a direct translation of φ (s_0 abstracts $(x < 2)$, s_1 abstracts $(y > 1)$ and s_2 abstracts $(y < x)$) that over-approximates the power of the system. The additional subformula $\varphi^{\textit{Extra}}$ corrects the over-approximation and makes $\varphi_\mathbb{B}$ preserve the original decision power of each player:*

$$\begin{array}{lll} e_0 & \rightarrow & (s_0 \wedge s_1 \wedge \overline{s_2}) \vee (s_0 \wedge \overline{s_1} \wedge s_2) \vee (s_0 \wedge \overline{s_1} \wedge \overline{s_2}) \\ & \wedge & \\ e_1 & \rightarrow & (\overline{s_0} \wedge s_1 \wedge \overline{s_2}) \vee (\overline{s_0} \wedge \overline{s_1} \wedge s_2) \\ & \wedge & \\ e_2 & \rightarrow & (\overline{s_0} \wedge s_1 \wedge s_2) \vee (\overline{s_0} \wedge s_1 \wedge \overline{s_2}) \vee (\overline{s_0} \wedge \overline{s_1} \wedge s_2) \end{array}$$

where e_0, e_1, e_2 belong to the environment and s_0, s_1 belong to the system (and all of them are Boolean). Intuitively, e_0 represents $(x < 2)$, e_1 represents $(x = 2)$ and e_2 represents $(x > 2)$. Choices are playable valuations in control of the system: $c_0 = \{s_0 \wedge s_1 \wedge s_2\}$, $c_0 = \{s_0 \wedge s_1 \wedge \overline{s_2}\}$, $\dots c_7 = \{\overline{s_0} \wedge \overline{s_1} \wedge \overline{s_2}\}$. In other words, the system can respond to e_0 with either c_1, c_2 or c_3; to e_1 with either c_5 or c_6; and to e_2 with either c_4, c_5 or c_6.

Note that e_1 results in a strictly more restrictive set of choices for the system than e_2, which allows the system to choose one more valuation, specificaly $c_4 = (\overline{s_0} \wedge s_1 \wedge s_2)$. Thus, a "clever" environment will never play e_2 and it will play e_1 instead. Therefore, for simplicity in the paper, we will consider the simplified (equi-realizable) specification $\varphi_\mathbb{B} = [(s_0 \rightarrow s_1) \wedge (\overline{s_0} \rightarrow s_2)] \wedge \Box[(e_0 \leftrightarrow \overline{e_1}) \rightarrow \varphi_{\textit{Extra}'}]$, where $\varphi_{\textit{Extra}'}$ is a version of $\varphi_{\textit{Extra}}$ where e_2 is ignored.

3 Unconditional Explanations

3.1 Unconditional Encoding

The goal of this paper is to generate simple and automatic explanations for unrealizability in safety LTL$_\mathcal{T}$ specifications. We soon explain how we perform this automatization, but first introduce a notion of *simplicity*.

Since, in a reactive system, the interplay between the players can be very complex, it is appealing to find a strategy that reduces the noise of this interplay as far as possible: in other words, ignoring some moves of the (potentially infinite) interplay in order to get an explanation that is as close as possible to a prefix. This is particularly interesting in shields, because we provide a simple explanation regardless of how sophisticated the shielded policy is, and also regardless of how complex the environment is. In this paper, we designed these explanations in the form of *unconditional* and *conditional* strategies. Let us begin with the first ones:

Definition 1. *Given a specification φ and a length k, we call unconditional strategies ρ_k to strategies in which environment can reach a violation of φ in i timesteps with a sequence of moves that is independent of the system.*

More formally, ρ_k is a constant function that assigns valuations to the environment variables up to length k.

Example 3. *In φ of Ex. 1 it suffices for the environment to (1) play a value for x such that $(x < 2)$, for example $x : 1$, in timestep $i = 0$; and (2) play $x : 2$ in $i = 1$. This is a winning strategy for the environment in two steps, no matter what the system plays. Thus, we say that the environment has a two-step unconditional strategy (or explanation) in φ and we denote it $\rho_{k:2} = \{x^0 : 1, x^1 : 2\}$.*

Note that, from this point onwards, we use the terms *explanation* and *strategy* interchangeably, whenever the meaning is clear from the context. Now, in order to generate unconditional explanations/strategies from φ of a length up to k, we propose to use *unrollings* of φ to formalize the existence of this statement in some logic. Unconditional properties are of the form *there is* a sequence of moves of the environment such that, *for all* moves of the system, the formula is violated, which corresponds to a prefix $\exists^*\forall^*$ formula.

Definition 2. *Given a specification φ and a length k, we call unconditional unrolling formula to an encoding $\psi = \exists a^0, a^1, ..., a^{k-1}.\forall b^0, b^1, ..., b^{k-1}.\neg[\varphi_0 \wedge \varphi_1 \wedge ... \wedge \varphi_{k-1}]$, where variables a_i are controlled by the environment and b_j belong to the system, and formulae φ_i correspond to instantiations of the specification φ at instant i.*

Note that we negate $[\varphi_0 ...]$ in our query, because the objective of the environment is to find a witness of the negation of φ. Also, note that copies φ_i resemble classical bounded model checking (Biere et al. 1999; Clarke et al. 2001).

Remark 1. *Before the encoding, φ is transformed into negation normal form, and then specialized as $\varphi^{i,k}$ for every step i (and the maximum unrolling k), using the fix-point expansions of the temporal operators. When an appropriate solver for ψ is searching to make a formula $\top$, then every attempt to expand a sub-formula beyond k is replaced by $\bot$ (and vice-versa when seeking to make a formula $\bot$).*

3.2 Method #1: A Q-SMT Encoding

For a specification φ in LTL$_\mathcal{T}$, ψ of Def. 2 is a quantified formula in some first-order theory $\mathcal{T}$, which is a natural encoding that can be solved using quantified satisfiability modulo theories (Q-SMT) procedures; for instance, (Cooper 1972) for integers or (Collins 1975) for reals.

Now, we detail our method and we start with $k = 1$ in order to find whether there is an environment one-step unconditional winning strategy $\rho_{k=1}$ for $\varphi_\mathcal{T}$.

Example 4. *The* 1*-unrolling for* φ *in Ex. 1 is:* $\exists x^0.\forall y^0.\neg\varphi^{0,1}$*, where*

$$\varphi^{0,1} = [((x^0 < 2) \rightarrow \top) \wedge ((x^0 \geq 2) \rightarrow (y^0 < x^0))],$$

and where $x^0, y^0 \in \mathbb{Z}$ *are variables* x *and* y *instantiated in timestep* $i = 0$*. Note that the sub-formula* $\bigcirc(y > 1)$ *is replaced by* $\top$ *because it falls beyond the end of the unrolling, as in standard bounded model checking (as described in remark 1). Thus, the definitive encoding is:*

$$\exists x^0.\forall y^0.\neg(((x^0 < 2) \rightarrow \top) \wedge ((x^0 \geq 2) \rightarrow (y^0 < x^0))),$$

which is unsatisfiable (i.e., `unsat`*). Thus, there is no strategy* $\rho_{k=1}$*.*

However, as seen in Ex. 3, there exists a two-step unconditional strategy $\rho_{k=2}$*. Moreover, the following Q-SMT encoding (for unrollings depth* $k = 2$*) finds this witness.*

$$\exists x^0, x^1.\forall y^0, y^1.\neg[\varphi^{0,2} \wedge \varphi^{1,2}], \text{ where}$$

$$\varphi^{0,2} = [((x^0 < 2) \rightarrow (y^1 > 1)) \wedge ((x^0 \geq 2) \rightarrow (y^0 < x^0))]$$

and

$$\varphi^{1,2} = [((x^1 < 2) \rightarrow \top) \wedge ((x^1 \geq 2) \rightarrow (y^1 < x^1))]$$

In this case, a Q-SMT solver will respond that the formula is satisfiable and will output a model such as $m = \{x^0 := 1, x^1 := 2\}$ *as an assignment that satisfies it. We can see that* m *is precisely* $\rho_{k=2}$*. Most importantly, note how easily the user can understand this explanation comparing to visually inspecting the environment strategy (of Fig. 3).*

In summary, this approach relies on incremental calls to Alg. 1, which receives the LTL$_\mathcal{T}$ specification $\varphi_\mathcal{T}$, an unrolling depth $k = max$, environment variable set X and system variable set Y. Alg. 1 also uses sub-procedures: (1) *copies*(A, n), which performs n timestep copies of the set A of variables; (2) *unroll*(φ, n), which performs m unrollings of φ; (3) *QElim*(X, φ), which performs quantifier-elimination (QE) of set of variables A of variables from formula φ (which must be the inner set of variables); and (4) *witness*(φ), which returns a model of a satisfiable formula φ. Note that the unrolling formula F (line 5) has $\{x_0, \ldots, x_n, y_0 \ldots, y_n\}$ as free variables, G (line 6) quantifies universally over $\{y_0 \ldots, y_m\}$ so it has $\{x_0, \ldots, x_n\}$ as free variables, and therefore $\varphi_\mathcal{T}^{\text{smt}}$ (line 8) is quantifier-free with $\{x_0, \ldots, x_n\}$ as free variables. Also, note that, for each of the Q-SMT queries to terminate, we require $\mathcal{T}$ to be decidable in the $\exists^*\forall^*$-fragment.

Soundness of Alg. 1 is due to the following:

Theorem 1. *If there is some unrolling depth* k *such that* $\varphi_\mathcal{T}^{\text{smt}}$ *is* `sat`*, then* $\varphi_\mathcal{T}$ *is unrealizable.*

Algorithm 1 Unconditional bounded unrealizability check

```
Require: $\varphi_\mathcal{T}$, $max$, $X$, $Y$
 1: for $n = 1$ to $max$ do
 2:     $[y_0, \ldots, y_n] \leftarrow copies(Y, n)$
 3:     $[x_0, \ldots, x_n] \leftarrow copies(X, n)$
 4:     $F \leftarrow unroll(\varphi_\mathcal{T}, n)$
 5:     $G \leftarrow \forall y_0 \forall y_1 \ldots \forall y_n.F$
 6:     $\varphi_\mathcal{T}^{\text{smt}} \leftarrow QElim([y_0, \ldots, y_n], G)$
 7:     if $\neg\varphi_\mathcal{T}^{\text{smt}}$ is SAT then
 8:         return $(\texttt{true}, witness(\varphi_\mathcal{T}^{\text{smt}}))$
 9:     end if
10: end for
11: return uncertain
```

Remark 2. *The Q-SMT encoding described in Ex. 4 and Alg. 1 is also suitable for theories whose satisfiability problems are semi-decidable: if* m *is obtained, then it is a legal witness of the desired strategy. For instance, we can use general-purpose SMT solvers such as Z3 (de Moura and Bjørner 2008) and encode unrollings for the theory of non-linear integer arithmetic.*

Remark 3. *Since* $\mathcal{T}$ *might have many models satisfying a same valuation of the literals, one may be not only interested in finding a single* ρ*, but instead in obtaining the best strategy with respect to some soft criteria.*

Example 5. *Recall the* ρ *in Ex. 4 is* $\rho = \{(x^0 : 1), (x^1 : 2)\}$*. If the user wants* x_0 *to be as closest as possible to* 0*; then an alternative* $\rho' = \{(x^0 : 0), (x^1 : 2)\}$ *is a better unrealizability witness. A different criteria is to prioritize dynamic realism, e.g., if the designers are interested in smooth solutions of the environment in Ex. 2, then they prefer all the environment-controlled valuations to be as similar as possible to each other. For instance, we would prefer* ρ *to* ρ'*, since* $x^0 : 1$ *is closer to* $x^1 : 2$ *than* $x^0 : 0$*. This way, designers would like to see, for example, if the environment has a way to violate the shield specification and without the need of abrupt movements, but instead more realistic ones. This provides engineers with more understandable explanations, but heavily depends on the domain of usage (further research on this is out of the scope of the paper).*

Algorithm 2 Uncond. bounded unrealiz. check for LTL.

```
Require: $\varphi_\mathbb{B}$, $max$, $E$, $S$
for $n = 1$ to $max$ do
    $[s_0, \ldots, s_n] \leftarrow copies(S, n)$
    $[e_0, \ldots, e_n] \leftarrow copies(E, n)$
    $F \leftarrow unroll(\varphi_\mathbb{B}, n)$
    $G \leftarrow \forall s_0 \forall s_1 \ldots \forall s_n.F$
    $\varphi_\mathbb{B}^{\text{qbf}} \leftarrow QElim([s_0, \ldots, s_n], G)$
    if $\neg\varphi_\mathbb{B}^{\text{qbf}}$ is SAT then
        return $(\texttt{true}, witness(\varphi_\mathbb{B}^{\text{qbf}}))$
    end if
end for
return uncertain
```

3.3 Method #2: Boolean Abstraction to QBF

Although Alg. 1 is sound (find proofs of theorems in App. B), each Q-SMT query lacks termination guarantees (Bjørner and Janota 2015). Therefore, we propose an alternative method based on Boolean abstractions for which the evaluation of each unrolling is guaranteed to terminate. In this method, we generate unrolled formulae in QBF. This second method follows these steps: (1) we compute an equi-realizable purely Boolean LTL $\varphi_{\mathbb{B}}$ from φ following the Boolean abstraction method (Rodríguez and Sánchez 2023); (2) find an unconditional strategy $\rho^{\mathbb{B}}$ in $\varphi_{\mathbb{B}}$ using QBF solvers; and (3) translate $\rho^{\mathbb{B}}$ from Booleans to $\mathcal{T}$ using existential theory queries, generating a strategy for φ.

Example 6. *Consider the search for an unconditional $\rho^{\mathbb{B}}$ for $\varphi_{\mathbb{B}}$ in Ex. 2. Let a_i^j, where i refers to the variable and j to the timestep. Again, k is the unrolling depth. We now encode the problem in QBF with unrolling $k = 1$:*

$$\exists e_0^0, e_1^0 . \forall s_0^0, s_1^0, s_2^0 . \neg \varphi_{\mathbb{B}}^{0,1},$$

where $\varphi_{\mathbb{B}}^{0,1} = [(s_0^0 \rightarrow \top) \wedge (\overline{s_0^0} \rightarrow s_2^0)] \rightarrow [(e_0^0 \leftrightarrow e_1^0) \rightarrow \varphi_{Extra}^{0,1}]$ and where $\varphi_{Extra}^{0,1}$:

$$\begin{array}{rcl} e_0^0 & \rightarrow & \big(s_0^0 \wedge s_1^0 \wedge \overline{s_2^0}\big) \vee \big(s_0^0 \wedge \overline{s_1^0} \wedge s_2^0\big) \vee \big(s_0^0 \wedge \overline{s_1^0} \wedge \overline{s_2^0}\big) \\ & \wedge & \\ e_1^0 & \rightarrow & \big(\overline{s_0^0} \wedge s_1^0 \wedge \overline{s_2^0}\big) \vee \big(\overline{s_0^0} \wedge \overline{s_1^0} \wedge s_2^0\big) \end{array}$$

Note, again like in remark 1, that since $k = 1$, every attempt to access a proposition at time step greater than or equal k (for example, s_1^1) is replaced by $\top$. Hence, the formula $(s_0^0 \rightarrow \top)$ does not impose an actual constraint. As we saw in Subsec. 3.2, there is no strategy ρ for φ. Similarly, the QBF unrolling cannot find a strategy $\rho^{\mathbb{B}}$ for $\varphi_{\mathbb{B}}$, as the system only has to satisfy $(\overline{s_0^0} \rightarrow s_2^0)$, which happens if environment plays e_0^0. This is as expected, since φ and $\varphi_{\mathbb{B}}$ should be equi-realizable and, thus, strategies should be mutually imitable (in this case, have the same length).

Analogously, since there is $\rho_{k=2}$ for φ, then there is a corresponding $\rho_{k=2}^{\mathbb{B}}$ for $\varphi_{\mathbb{B}}$ via a $k = 2$ QBF unrolling:

$$\exists e_0^0, e_1^0, e_0^1, e_1^1 . \forall s_0^0, s_1^0, s_2^0, s_0^1, s_1^1, s_2^1 . \neg(\varphi_{\mathbb{B}}^{0,2} \wedge \varphi_{\mathbb{B}}^{1,2}),$$

where $\varphi_{\mathbb{B}}^{0,1} = [(s_0^0 \rightarrow s_1^0) \wedge (\overline{s_0^0} \rightarrow s_2^0)] \rightarrow [(e_0^0 \leftrightarrow \neg e_0^1) \rightarrow \varphi_{Extra}^{0,1}]$, with $\varphi_{Extra}^{0,1} = \varphi_{Extra}^{0,2}$ and where $\varphi_{\mathbb{B}}^{1,2} = [(s_0^1 \rightarrow \top) \wedge (\overline{s_0^1} \rightarrow s_2^1)] \rightarrow [(e_0^1 \leftrightarrow \overline{e_1^1}) \rightarrow \varphi_{Extra}^{1,2}]$, with $\varphi_{Extra}^{1,2}$:

$$\begin{array}{rcl} e_0^1 & \rightarrow & \big(s_0^1 \wedge s_1^1 \wedge \overline{s_2^1}\big) \vee \big(s_0^1 \wedge \overline{s_1^1} \wedge s_2^1\big) \vee \big(s_0^1 \wedge \overline{s_1^1} \wedge \overline{s_2^1}\big) \\ & \wedge & \\ e_1^1 & \rightarrow & \big(\overline{s_0^1} \wedge s_1^1 \wedge \overline{s_2^1}\big) \vee \big(\overline{s_0^1} \wedge \overline{s_1^1} \wedge s_2^1\big) \end{array}$$

Note again that an attempt to generate s_1^2 is replaced by $\top$, so the formula $(s_0^1 \rightarrow \top)$ again imposes no constraint. This time, the environment has a winning strategy if it plays e_0^0 and e_1^1: playing e_0^0 in timestep $i = 0$ forces s_1^1 to hold in timestep $i = 1$; and e_1^1 in $i = 1$ forces s_2^1 in $i = 1$. Then, only $(\overline{s_0^1} \wedge s_1^1 \wedge \overline{s_2^1})$ and $(\overline{s_0^1} \wedge \overline{s_1^1} \wedge s_2^1)$ are valid responses of the system, but none of them satisfied both s_2^1 forced in timestep 1 and s_1^1 forced by the previous timestep, so the specification is inevitably violated. Thus, in $\varphi_{\mathbb{B}}$, it exists $\rho_{k=2}^{\mathbb{B}} = \{e_0^0, e_1^1\}$. Last, if we recall that e_0 represents $(x < 2)$ and e_1 represents $(x = 2)$, the strategy $\rho^{\mathbb{B}}$ in $\varphi_{\mathbb{B}}$ obtained with QBF encoding is coherent with $\rho_{k=2} = \{x^0 : 1, x^1 : 2\}$ obtained with Q-SMT $\varphi_{\mathcal{T}}$ in Ex. 4.

In summary, this approach starts by φ, performs abstraction $\varphi_{\mathbb{B}}$, and then again relies on incremental calls to an algorithm that is identical to Alg. 1, except for the fact that it receives $\varphi_{\mathbb{B}}$, Boolean environment variable set E and Boolean system variable set S, and does not solve an SMT query, but a QBF query (see Alg. 2). Soundness of this method follows analogously to Thm. 1, given that the Boolean abstraction method in use ensures equi-realizability of φ and $\varphi_{\mathbb{B}}$, which holds with (Rodríguez and Sánchez 2023).

Theorem 2. *If there is some unrolling depth k such that $\varphi_{\mathbb{B}}^{qbf}$ is* `sat`*, then $\varphi_{\mathbb{B}}$ is unrealizable.*

Remark 4. *Even though the performance of QBF solvers degrades with quantifier alternations, modern solvers scale efficiently even for large formulae with $\exists^*\forall^*$ prefixes. Although scalability is not the goal of methods presented in this paper, QBF is a more mature technology than Q-SMT, which suggests that performance might also be gained regularly (see Sec. 6 for experiments in scalability).*

Remark 5. *Similar to remark 2, note that our method is agnostic to the abstraction method in use (even semi-decidable methods), as long as it preserves equi-realizability.*

Remark 6. *Similar to remark 3, note that $\rho^{\mathbb{B}}$ of Ex. 6, could also be related to other strategies in $\varphi_{\mathcal{T}}$ rather than ρ: indeed, $\rho' = \{x^0 : 0, x^1 : 2\}$, to $\rho'' = \{x^0 : -1, x^1 : 2\}$, to $\rho''' = \{x^0 : -2, x^1 : 2\}$ and an infinite amount of strategies that make the literal $(x^0 < 2)$ true.*

Remark 7. *One might wonder when is a Q-SMT encoding preferable to the QBF encoding. We believe there are at least three situations to consider: (1) If the abstraction is not terminating, then we can try a Q-SMT encoding. (2) If the theory $\mathcal{T}$ is undecidable, then still Q-SMT solvers have heuristics for semi-decidability that may produce explanations (which are correct by construction). (3) In very small instances, abstraction might consume most of the time of a* `abstraction+QBF` *query, so Q-SMT might be faster.*

Algorithm 3 Conditional bounded unrealizability check

```
Require: $\varphi_{\mathcal{T}}$, max, $X$, $Y$
for $n = 1$ to max do
    $[y_0, \ldots, y_n] \leftarrow copies(Y, n)$
    $[x_0, \ldots, x_n] \leftarrow copies(X, n)$
    $F \leftarrow unroll(\varphi_{\mathcal{T}}, n)$
    $G \leftarrow \texttt{alternats}([y_0, \ldots, y_n], [x_0, \ldots, x_n]).F$
    $\varphi_{\mathcal{T}}^{\text{smt}} \leftarrow QElim([y_0, \ldots, y_n], G)$
    if $\neg\varphi_{\mathcal{T}}^{\text{smt}}$ is SAT then
        return $(\texttt{true}, witness(\varphi_{\mathcal{T}}^{\text{smt}}))$
    end if
end for
return uncertain
```

3.4 Strategy Deabstraction

Since the explanation of Alg. 2 is given in terms of Boolean variables, we need a technique to produce proper values in the domain of the theory $\mathcal{T}$. We describe now how to craft a strategy ρ from $\rho^{\mathbb{B}}$.

Definition 3. *Consider an* LTL$_{\mathcal{T}}$ *specification* φ *and its equi-realizable abstraction* $\varphi_{\mathbb{B}}$. *Then, if* $\rho^{\mathbb{B}}$ *is winning for the environment in* $\varphi_{\mathbb{B}}$, *we call a strategy deabstraction function to a function* $d : \rho^{\mathbb{B}} \to \rho$ *such that if* $\rho^{\mathbb{B}}$ *is winning in* $\varphi_{\mathbb{B}}$ *then* ρ *is winning in* φ.

In the case of unconditional strategies, these deabstracted strategies can be obtained leveraging the partitions used during the Boolean abstraction.

Example 7. *Consider Ex. 2 and the two environment variables* e_0 *and* e_1, *which are discrete partitions of the infinite input space of for variables of the environment. Recall from Sec. 2 that this partition comes from the fact that the abstraction algorithm found reactions* $r_0 = (x < 2)$ *and* $r_1 = (x = 2)$ *to be valid.*

In other words, in order to get a literal in $\mathcal{T}$ from an environment variable e_k in $\varphi_{\mathbb{B}}$, it suffices to associate each e_k to the reaction formula in $\mathcal{T}$ that is valid.

Definition 4. *Let function* conv *produce a formula in* $\mathcal{T}$ *from each* e_k. *Then, our deabstraction procedure is:*

$$d_{\rho^{\mathbb{B}} \text{ to } \rho} : \bigwedge_{e_k^i \in \rho^{\mathbb{B}}}^{i \leq k} (\mathit{conv}(e_k^i))$$

This way, we only need to produce a $\mathcal{T}$-valuation of x that satisfies the reaction associated to e_k.

Example 8. *For the reactions in Ex 7, assignments* $\exists x.r_0(x) \leftarrow 1$ *and* $\exists x.r_1(x) \leftarrow 2$ *mean that* e_0, *which represents,* $(x < 2)$ *has a witness* $x : 1$*; and* e_1, *which represents* $(x = 2)$, *has the (only possible) witness* $x : 2$. *This allows to characterize the strategy of Ex. 2 as expected. Since the Boolean strategy of the environment is* $\rho^{\mathbb{B}} = \{e_0^0, e_1^1\}$, *then a possible* $\mathcal{T}$*-strategy is* $\rho = \{(x^0 : 1), (x^1 : 2)\}$, *as predicted in Ex. 6.*

We now formalize that any Boolean strategy $\rho_{\mathbb{B}} = \{e_i^0, e_j^1, ...\}$ of the environment is related to a $\rho = \{(x^0 : a), (x^1 : b), ...\}$, where $a, b \in \mathcal{T}$, in the sense that the outcomes of $\varphi_{\mathbb{B}}$ and $\varphi_{\mathcal{T}}$ are related in terms of literals they satisfy.

Theorem 3. *Let* φ *and* $\varphi_{\mathbb{B}}$ *it Boolean abstraction. Then, an unconditional strategy* ρ *of length* k *exists if and only if* $\rho^{\mathbb{B}}$ *of length* k *exists.*

Remark 8. *Thm. 3 is different to correctness theorems of (Rodríguez and Sánchez 2023), because we prove that a strategy of* ***same length*** *exists in both* φ *and* $\varphi_{\mathbb{B}}$.

Remark 9. *Soundness of deabstraction (and thus Thm. 3) is preserved with any model that satisfies a given choice, meaning that, like we did in remark 3 of the Q-SMT method of Sec. 3.2, we can compute the witness that maximizes a certain soft criteria.*

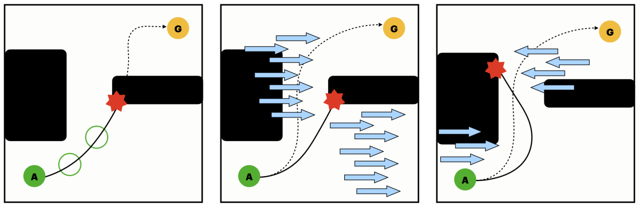

Figure 1: Illustration of agent behavior under different conditions.

4 Conditional Explanations

In Sec. 3 we showed how to discover unconditional explanations for unrealizability, which are simple and preferrable for the solvers (because of the lack of quantifier alternations). However, they carry two fundamental problems: (1) incompleteness (unsatisfiability of a k-step unrolling does not mean the formula is unrealizable in k timesteps); and (2) lack of practical applicability, due to the fact that often an unrealizable formula does not contain an unconditional strategy that explains it. Both problems require that the environment movements are no longer independent to the system, but instead has the ability to look at them in order to choose the move most promising to win.

Definition 5. *Given a specification* φ *and a length k, we call conditional strategies* ρ_k *to strategies in which environment can reach a violation of* φ *in* i *timesteps with a sequence of moves that depends on previous moves of the system.*

Example 9. *We illustrate conditionality with* φ*':*

$$\begin{array}{rr} (x < 5) \to & \bigcirc^3(y > 9) \\ \wedge & \\ (5 \leq x < 10) \to & [((y < 0) \to \bigcirc(y > 9)) \\ & \wedge((y \geq 0) \to \bigcirc(y \leq 9))] \\ \wedge & \\ (10 \leq x \leq 15) \to & (y < x) \\ (x > 15) \to & (y > x), \end{array}$$

which is unrealizable. Again, we can synthetise an environment automata for φ' *and again the corresponding automata is not easy to interpret (see Fig. 4 in App. E). Instead, we can get both unconditional and conditional explanations.*

First, φ' *is unconditionally unrealizable in 4 timesteps: if the environment plays* $x : (x < 5)$ *in* t_k, *then* $y > 9$ *has to hold in* t_{k+3}*; and if* $x : 10$ *in* t_{k+3}, *then* $(y < x)$ *has to hold (i.e.,* $y < 10$*), which contradicts* $(y > 9)$. *Note that, if the first line of* φ' *did not exist, then there would not exists an unconditional strategy.*

Additionally, φ *does contain a shorter strategy of the environment to win, but this strategy is conditional (i.e., looks at the previous play of the system): concretely, the environment can play* $x : (5 \leq x < 10)$ *in* t_k, *then looks at whether the system played* $(y < 0)$ *or* $(y \geq 0)$ *in* t_k, *which imposes* $(y > 0)$ *or* $(y \leq 9)$ *respectively in* t_{k+1} *and thus the environment reacts accordingly in* $k + 1$ *by playing* $(10 \leq x \leq 15)$ *or* $(x > 15)$ *respectively in order to violate* φ. *Note how this strategy is shorter, but more complex.*

In order to automatically produce these explanations, we now present *conditional* strategy search, for which we need to solve $(\exists^*\forall^*)^*$ formulae.

Definition 6. *Given a specification φ and a length k, we call conditional unrolling formula to an encoding*

$$\exists a^0.\forall b^0.\exists a^1.\forall b^1, ..., \exists a^{k-1}.\forall = b^{k-1}.\neg[\varphi_0 \wedge ... \wedge \varphi_{k-1}],$$

where again variables a_i are controlled by the environment and b_j belong to the system and formulae φ_i correspond to instantiations of φ at instant i.

Example 10. *Consider φ' from Ex. 9. Unfortunately, due to space limitations, we cannot show the 1-step unrolling (only note that its verdict is* `unsat` *and that both conditional and unconditional formulae are the same). Let us denote with Q an arbitrary quantification. Then, the 2-step unrolling is $Q.\neg[\varphi^{0,1} \wedge \varphi^{1,2}]$, where $\varphi^{0,1}$:*

$$\begin{aligned}
(x^0 < 5) \rightarrow & \qquad \top \\
(5 \leq x^0 < 10) \rightarrow & \qquad [((y^0 < 0) \rightarrow (y^1 > 9)) \\
& \qquad \wedge((y^0 \geq 0) \rightarrow (y^1 \leq 9))] \\
(10 \leq x^0 \leq 15) \rightarrow & \qquad (y^0 < x) \\
(x^0 > 15) \rightarrow & \qquad (y^0 > x),
\end{aligned}$$

and where $\varphi^{1,2}$:

$$\begin{aligned}
(x^1 < 5) \rightarrow & \qquad \top \\
(5 \leq x^1 < 10) \rightarrow & \qquad \top \\
(10 \leq x^1 \leq 15) \rightarrow & \qquad (y^1 < x) \\
(x^1 > 15) \rightarrow & \qquad (y^1 > x)
\end{aligned}$$

As we can see, an unconditional instantiation, $Q = \exists x^0.\forall y^0.\exists x^1.\forall y^1$, results in an `unsat` *verdict (because there is no way the environment can win in 2 steps unconditionally, although it can do it in 3 steps as shown in Ex. 9). However, if the environment is allowed to observe the prefix of the play, then the conditional instantiation $Q' = \exists x^0.\forall y^0.\exists x^1.\forall y^1$ is* `sat` *with a model m'.*

Moreover, since a conditional formula captures the environment's ability to to look at the trace and adapt the behaviour, a model m' is no longer an assignment of environment variables to valuations; instead, this only happens in instant $i = 0$, whereas for any $i > 0$ the valuation of x^i is decided using a Skolem function that depends on the previous value of the system. For instance, $\{x^0 : 7, x^1 : f_{x^1}(y^0)\}$, where:

$$f_{x^1}(y^0) = \begin{cases} 10 & \textit{if } (y^0 < 5) \\ 18 & \textit{otherwise} \end{cases}$$

In summary, we can construct Alg. 3 where changes with respect to Alg. 1 are that (1) G from line 6 now represents quantifier alternations, (2) `witness` from line 8 does not produce constant Skolem functions and (3) line 11 returns `unreal` (see Thm. 5 below). Soundness of Alg. 3 for safety is due to the following:

Theorem 4. *If there is some unrolling depth k such that $\varphi_{\mathcal{T}}^{smt}$ of Alg. 3 is* `sat`*, then $\varphi_{\mathcal{T}}$ is unrealizable.*

Moreover, for arbitrary k, Alg. 3 is guaranteed to find a conditional strategy of length k, whenever φ is an unrealizable safety specifications. This provides k-completeness.

Theorem 5. *If φ is an unrealizable safety formula in* LTL$_{\mathcal{T}}$*, then there is an unrolling k such that $\varphi_{\mathcal{T}}^{smt}$ of Alg. 3 is* `sat`*.*

However, as in Sec. 3, extracting simple conditional explanations is a challenge and using Q-SMT for $(\exists^*\forall^*)^*$ can be intractable or even undecidable for some theories. On the other side, Boolean abstraction for LTL$_{\mathcal{T}}$ is decidable for $\exists^*\forall^*$ decidable theories, so whenever the abstraction is obtained, we can perform a QBF encoding that is analogous to the one in Subsec. 3.3: this approach (see Alg. 4 in App. E) starts by $\varphi_{\mathcal{T}}$, performs abstraction $\varphi_{\mathbb{B}}$ then again relies on incremental calls to an algorithm that is identical to Alg. 3, except for the fact that it receives $\varphi_{\mathbb{B}}$, Boolean environment variable set E and Boolean system variable set S, and does not solve an SMT query, but a QBF query. Soundness of and completeness of this method follow analogously.

Theorem 6. *If there is some unrolling depth k such that conditional $\varphi_{\mathbb{B}}^{qbf}$ is* `sat`*, then $\varphi_{\mathbb{B}}$ is unrealizable.*

Theorem 7. *If φ is an unrealizable safety formula in* LTL$_{\mathcal{T}}$*, then there is an unrolling k such that cond. $\varphi_{\mathbb{B}}^{qbf}$ is* `sat`*.*

However, again we need a deabstraction procedure, because the explanations that we will get will be made up of Boolean Skolem functions. Moreover, leaning on Subsec. 3.4, the deabstraction procedure for the Boolean Skolem function is trivial: (1) in timestep $i = 0$ it substitutes the environment assignments by reactions (exactly as in the unconditional case); whereas (2) in $i >$ it takes each Boolean Skolem function and performs a term substitution by replacing the domain (variables s^i by their literals in $\mathcal{T}$) and the co-domain (variables e^i by the reactions). Thus:

Theorem 8. *Let φ and abstraction $\varphi_{\mathbb{B}}$. Then, an conditional strategy ρ of length k exists iff $\rho^{\mathbb{B}}$ of length k exists.*

Remark 10. *We can construct arbitrary versions of the algorithms presented in this paper if we consider semi-conditional strategies that are between conditional and unconditional. For instance, for $k = 10$, it can be the case that an unconditional unrolling (which has $a = 1$ quantifier alternations) is* `unsat`*, a conditional encoding (for which $a = 10$) is* `sat` *and some semi-conditional encoding (for which $1 < a < 10$) is* `sat`*.*

Remark 11. *It is very important to note that a conditional encoding is not necessarily harder to solve than unconditional encodings for QBF solvers. This is not surprising, since the number of quantifier alternations are not the sole deciding factors for hardness of a problem. For instance, in this paper, we presented Ex. 1 which can be solved using unconditional strategy. However, one could easily encode the same problem using a conditional strategy, but this does not increase the intrinsic hardness of the problem.*

Indeed, every problem has its intrinsic hardness and it is not trivial to figure out at which layer such a problem belongs to in the polynomial hierarchy.

5 Discussion: What is a *better* explanation?

Since the unrealizability explanations that we find are the shortest possible (because the bounded search ends as soon as a satisfiable assignment is found) it is easy to see that a shorter explanation is better than a longer one. However, there are other criteria for measuring quality of explanations: e.g., if we consider *conditionality*, then finding a strategy that is less conditional is better than a more conditional one, because the less conditional the strategy, more of its Skolem functions will be constants. We now formalize this intuition.

Definition 7. *We measure the quality of an explanation as a function $q(k, o)$, where k is the length of the explanation and o is the amount of times the environment observes some prefix of the play (note $o = 0$ corresponds to unconditional).*

Since the goal is to minimize q, then, given k and o for strategy ρ and k' and o' for explanation ρ', where $k \leq j'$ and $o \leq o'$, then it is easy to see that $q(k, o) \leq q(k', o')$, which means q is a *better* explanation. However, what if $k \leq k'$ but $o > o'$, or vice-versa? For instance, in Ex. 10, is it better to have a longer unconditional explanation or a shorter conditional one? This is not always an easy choice and its automatization will heavily depend on our definition of q. Moreover, both kinds of explanations can be complementary, allowing us to find different unrealizability sources.

Example 11. *Consider a particle scenario where a drone wants to reach a certain goal while avoiding dangerous zones (please, refer to App. D for a complete description). Also, there is an uncontrollable (environment) wind-turbulence that can affect the position of the drone. Now, the developers (1) train the drone using RL until they reach a desired success rate in satisfying the goal, and afterwards (2) design the specification φ of an infinite-state safety shield that would achieve collision avoidance. However, they find that φ is unrealizable. Moreover, the environment strategy is not easy to interpret (the automata is too big), so they use methods of Sec. 3 and Sec. 4. In particular, they find that there are both conditional and unconditional explanations.*

We illustrate this using Fig. 1, where the green dot (A) represents the agent, and the yellow dot (G) represents the goal. Also, the dotted line indicates the agent's intended trajectory, while the solid line represents the actual trajectory executed. Now, in Fig. 1 ***(Left)*** *the agent collides without external influences.* ***(Middle)*** *The agent attempts to follow a safe trajectory, but unavoidable external perturbations (the wind represented by blue arrows) push it towards the obstacle, resulting in a collision.* ***(Right)*** *The agent initially follows a safe trajectory but is influenced by the wind, causing a deviation from its intended path. Despite this disturbance, the agent successfully recovers and continues towards the goal. However, later, a stronger perturbation pushes the agent in the opposite direction, resulting in a collision. As we can see, both the unconditional and conditional strategies (i.e.,* ***Middle*** *and* ***Right*** *resp.) allow us to discover information about φ to restrict the power of the environment towards a realizable version of φ:* ***Middle*** *shows that no matter what the agent does, the turbulence is too strong; whereas* ***Right*** *shows that φ is allowing an unrealistic dynamic such as changing the turbulence direction from timestep to timestep.*

6 Empirical Evaluation

Experimental setting. The main contribution of this paper is the method for obtaining simple explanations in reactive systems specified in LTL$_\mathcal{T}$. In addition, Sec. 5 showcases in a qualitative manner how the method helps finding such explanations for shielding. Now, we perform a scalability evaluation with a prototype `unrealExplainerT` that takes an LTL$_\mathcal{T}$ formula and a bound k, produces the unrolling for both Q-SMT and QBF approaches, and returns the unrealizability explanation (in case it exists).

Concretely, we created two versions for each benchmark in (Rodríguez and Sánchez 2023): (1) if the specification does not have a strategy of length k (e.g., it is realizable), we introduced minimal modifications to have such a strategy; (2) if the specification did contain such strategy, then we made minimal modifications not to have the strategy. We tested both the original and modified versions, both QBF and Q-SMT and both conditional and unconditional cases.

Unconditional encoding. We can see results in Tab. 1, under the block *unconditional*. The first column corresponds to the name of the benchmarks (*nm.*) The two next columns shows variables (*vr.*) and literals (*lt.*) per benchmark, where gray colour indicates that the specification is unrealizable and white colour means the opposite.; The following two groups of columns show results of the execution of QBF and Q-SMT techniques for a different (at most) number of unrollings of the formula: 10, 50 and 100 [2]. We show the time needed (in seconds) for each execution, where Preprocessing time (*Pre.*) in the QBF column is the time needed for computing the Boolean abstraction.

Results show that, even if QBF has an initial Boolean abstraction cost, it quickly begins to to perform better than Q-SMT in time and also reaches higher limits. Note that we can encounter false negatives (i.e., the specification is unrealizable but because there is no unconditional strategy), but all the results are sound. Also, note that the time necessary for constructing the unrollings of both QBF and Q-SMT formulae is negigible for these experiments.

Conditional encoding. Similar experiments were conducted for conditional versions of the benchmarks, noting that this method offers completeness, but at the cost of an expected dramatical decrease in scalability. The *conditional* block of Tab. 1 confirms these results, were times increased a lot and Q-SMT usually does not go beyond 5 whereas QBF does not go beyond 20 (usually not beyond 10). Therefore, we can argue that, in case conditional explanations are seeked, paying the price for the abstraction is absolutely worth it. Moreover, usually Q-SMT could not provide an answer (underlying SMT-solver responded `N/A`).

[2] Note that an interval of 100 timesteps in industrial benchmarks happens because time of the original specifications is dense: this means that if there is a requirement that must be satisfied in 0.2 seconds and another one in 1 second, this imposes a discrete representation with 5 timestep horizon. Thus, if another one must hold for 20 secs, this already has to be represented with 100 timesteps.

Bn.	Cls.	Unconditional							Conditional			
		QBF				Q-SMT			QBF		Q-SMT	
(nm.)	(v, l)	Pre.	10	50	100	10	50	100	10	20	10	20
Li.	(5, 16)	4.41	0.02	0.44	9.85	0.32	0.64	14.08	422	-	611	-
	(5, 16)	3.71	0.02	0.36	8.00	0.26	0.52	11.44	341	-	493	-
Tr.	(19, 36)	5.13	0.01	0.53	11.70	0.38	0.76	16.72	507	-	-	-
	(19, 36)	5.01	0.01	0.57	12.62	0.41	0.82	18.04	544	-	-	-
Con.	(2, 2)	4.37	0.02	0.47	10.52	0.34	0.68	15.04	452	490	654	7029
	(2, 2)	4.34	0.02	0.51	11.39	0.37	0.74	16.28	498	536	741	-
Tn.	(8, 8)	7.04	0.02	0.43	9.54	0.31	0.62	13.64	411	4454	-	-
	(8, 8)	7.38	0.02	0.61	13.55	0.44	0.88	19.36	588	6563	824	-
Air.	(13, 14)	3.29	0.01	0.50	11.14	0.36	0.72	15.92	477	-	6838	-
	(13, 14)	4.51	0.01	0.40	8.93	0.29	0.58	12.76	397	-	5124	-
Coo.	(3, 5)	3.64	0.03	0.46	10.16	0.33	0.66	14.52	443	475	622	681
	(3, 5)	3.56	0.03	0.54	12.01	0.39	0.78	17.16	520	556	754	811
Usb	(5, 8)	3.93	0.01	0.49	10.78	0.35	0.70	15.40	466	5151	6557	-
	(5, 8)	3.93	0.01	0.56	12.32	0.40	0.80	17.6	5343	5702	7837	-
St.	(11, 14)	6.06	0.02	0.39	8.62	0.28	0.56	12.32	3417	-	-	-
	(11, 14)	2.86	0.02	0.51	11.39	0.37	0.74	16.28	5162	-	-	-

Table 1: Results using both Q-SMT and QBF unrollings to find unconditional and conditional environment strategies of industrial benchmarks.

7 Final Remarks

Related Work. We classify our work in classic incremental SAT/SMT/QBF solving methods like CEGAR-loops (Clarke et al. 2000), which repeatedly check satisfiability while incrementally growing the formula/constraint set and bounded unroll constraints each iteration until the proof is found. Some concrete works are intended to find counter examples and counter traces for SMT solvers (e.g., (Chehida et al. 2021; Reynolds et al. 2015)). Also, QBF solvers have been used for such tasks (e.g., (Balabanov et al. 2015; Janota et al. 2016)) and (Hecking-Harbusch and Tentrup 2018) encodes Petri games in QBF and provides strategies of the environment. QBF has also been used for planning (Shaik and van de Pol 2022; Shaik et al. 2023).

Note that our fully conditional method, is not the same as bounded synthesis (Schewe and Finkbeiner 2007), because the latter bound strategies by size of the controller, not by a temporal size. However, our technique is similar to bounded model checking, BMC, (Biere et al. 1999; Clarke et al. 2001) in the sense that BMC algorithms unroll a finite-state automata for a fixed number of steps k, and check whether a property violation can occur in k or fewer steps. This typically involves encoding the restricted model as an instance of SAT. The process can be repeated with larger and larger values of k. Recently, QBF solving has been used in bounded model checking for hyperproperties (Hsu, Sánchez, and Bonakdarpour 2021), which opens the door for the same study over $\text{LTL}_{\mathcal{T}}$ properties (and also with loop conditions (Hsu, César Sánchez, and Bonakdarpour 2023)). Last, we find similar research directions in other temporal logics.; e.g., in STL (Maler and Nickovic 2004), the problem of falsification (i.e., achieving the violation of the proposed requirements) is considered, via different approaches (e.g., (Peltomäki and Porres 2022)).

Limitations and opportunities. In this work, we are restricted to the fragment of $\text{LTL}_{\mathcal{T}}$ for which synthesis is decidable. Therefore, future work includes proposing similar methods to temporal logics that allow to transfer richer data across time (e.g., infinite-trace (Geatti, Gianola, and Gigante 2022)). Also, we want to study usability of semi-conditional explanations (see remark 10), because they offer a trade-off between explainability, completeness and performance (e.g., to solve failures of Tab. 1) that seems promising.

Last, we consider that the discussion of Sec. 5 about optimizing different explanation criteria has to be widely made (and note that neurosymbolic approaches seem to be a key direction in order to optimize an eventual explainability function q). Thus, we want to integrate our solution under unified views of explainability. One example is comparing our work with conditional conformant planning (Smith and Weld 1998; Cimatti and Roveri 2000), where the objective is to find a sequence of actions that will guide the system to the desired state, regardless of the nondeterminism.

Conclusion. In this paper, we showed methods to find simple explanations of unrealizable $\text{LTL}_{\mathcal{T}}$ safety specifications, mostly designed for shields. We obtain such witnesses via unrollings of the formula up to a certain number k in a semi-complete spirit. We first showed that this method can be designed following an SMT-with-quantifiers (Q-SMT) encoding, but that this method lacks some termination guarantees and might not scale. Then, we showed a second method that uses an QBF encoding preceded by an abstraction process and a posterior deabstraction process. In both cases, we proposed a method to generate unconditional explanations (simpler, but less common) and conditional explanations (more complex, but complete). The paper is the basis for exciting work in other aforementioned directions.

References

Alshiekh, M.; Bloem, R.; Ehlers, R.; Könighofer, B.; Niekum, S.; and Topcu, U. 2018. Safe Reinforcement Learning via Shielding. In *Proc. of the 32nd AAAI Conference on Artificial Intelligence*, 2669–2678.

Azzopardi, S.; Stefano, L. D.; Piterman, N.; and Schneider, G. 2025. Full ltl synthesis over infinite-state arenas. In *Proc. of the 37th International Conference on Computer Aided Verification (CAV'25)*, LNCS.

Baier, C.; Coenen, N.; Finkbeiner, B.; Funke, F.; Jantsch, S.; and Siber, J. 2021. Causality-based game solving. In *Proc. of the 33rd International Conference in Computer Aided Verification (CAV'21), Part I*, volume 12759 of *LNCS*, 894–917. Springer.

Balabanov, V.; Jiang, J. R.; Janota, M.; and Widl, M. 2015. Efficient extraction of QBF (counter)models from long-distance resolution proofs. In *Proc. of the 29th Conference on Artificial Intelligence (AAAI 2015), January 25-30, 2015, Austin, Texas, USA*, 3694–3701. AAAI Press.

Barrett, C. W., and Tinelli, C. 2018. Satisfiability modulo theories. In *Handbook of Model Checking*. Springer. 305–343.

Bassan, S.; Amir, G.; Corsi, D.; Refaeli, I.; and Katz, G. 2023. Formally explaining neural networks within reactive systems. In *Formal Methods in Computer-Aided Design (FMCAD 2023)*, 1–13. IEEE.

Biere, A.; Cimatti, A.; Clarke, E. M.; and Zhu, Y. 1999. Symbolic model checking without BDDs. In *Proc. of the 5th Int'l Confe. on Tools and Algorithms for Construction and Analysis of Systems (TACAS'99)*, volume 1579 of *LNCS*, 193–207. Springer.

Bjørner, N. S., and Janota, M. 2015. Playing with quantified satisfaction. In *Proc. of the 20th International Conferences on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2015), Short Presentations, Suva, Fiji, November 24-28, 2015*, volume 35 of *EPiC Series in Computing*, 15–27. EasyChair.

Bloem, R.; Könighofer, B.; Könighofer, R.; and Wang, C. 2015. Shield Synthesis: - Runtime Enforcement for Reactive Systems. In *Proc. of the 21st Int. Conf. in Tools and Algorithms for the Construction and Analysis of Systems, (TACAS)*, volume 9035, 533–548.

Bradley, A. R., and Manna, Z. 2007. *The Calculus of Computation*. Springer-Verlag.

Chehida, S.; Ledru, Y.; Blein, Y.; and Vega, G. 2021. An SMT-based approach for generating trace examples and counter-examples of parametric properties. *Int. J. Crit. Comput. Based Syst.* 10(2):143–183.

Cimatti, A., and Roveri, M. 2000. Conformant planning via symbolic model checking. *J. Artif. Intell. Res.* 13:305–338.

Clarke, E. M.; Grumberg, O.; Jha, S.; Lu, Y.; and Veith, H. 2000. Counterexample-guided abstraction refinement. In *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*, volume 1855 of *Lecture Notes in Computer Science*, 154–169. Springer.

Clarke, E. M.; Biere, A.; Raimi, R.; and Zhu, Y. 2001. Bounded model checking using satisfiability solving. *Formal Methods Syst. Des.* 19(1):7–34.

Collins, G. E. 1975. Quantifier elimination for real closed fields by cylindrical algebraic decompostion. In *Automata Theory and Formal Languages*, volume 33 of *LNCS*, 134–183. Berlin, Heidelberg: Springer.

Cooper, D. W. 1972. Theorem proving in arithmetic without multiplication. *Machine Intelligence* 7(2):91–100.

Corsi, D.; Amir, G.; Rodríguez, A.; Katz, G.; Sánchez, C.; and Fox, R. 2024. Verification-guided shielding for deep reinforcement learning. *RLJ* 4:1759–1780.

Corsi, D.; Mallik, K.; Rodríguez, A.; and Sánchez, C. 2025. Efficient dynamic shielding for parametric safety specifications. In *Proc. of the 23rd International Symposium on Automated Technology for Verification and Analysis (ATVA 2025),*, LNCS. Springer.

de Moura, L. M., and Bjørner, N. S. 2008. Z3: an efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, 337–340. Springer.

Geatti, L.; Gianola, A.; and Gigante, N. 2022. Linear temporal logic modulo theories over finite traces. In *Proc. of the 31st International Joint Conference on Artificial Intelligence, (IJCAI 2022)*, 2641–2647. ijcai.org.

Goodfellow, I.; Shlens, J.; and Szegedy, C. 2014. Explaining and Harnessing Adversarial Examples. Technical Report. http://arxiv.org/abs/1412.6572.

Hecking-Harbusch, J., and Tentrup, L. 2018. Solving QBF by abstraction. In *Proc. of the 9th International Symposium on Games, Automata, Logics, and Formal Verification, (GandALF 2018), Saarbrücken, Germany, 26-28th September 2018*, volume 277 of *EPTCS*, 88–102.

Heim, P., and Dimitrova, R. 2025. Issy: A comprehensive tool for specification and synthesis of infinite-state reactive systems. In *Proc. of the 37th International Conference on Computer Aided Verification (CAV'25)*, LNCS.

Hsu, T.; César Sánchez, S. S.; and Bonakdarpour, B. 2023. Efficient loop conditions for bounded model checking hyperproperties. In *Proc. of the 29th International Conference in Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2023) , Held as Part of the European Joint Conferences on Theory and Practice of Software (ETAPS) 2023, Paris, France, April 23 - April 27, 2023*, volume ?? of *Lecture Notes in Computer Science*, ?? Springer.

Hsu, T.; Sánchez, C.; and Bonakdarpour, B. 2021. Bounded model checking for hyperproperties. In *Proc. of the 27th International Conference in Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2021)*, volume 12651 of *Lecture Notes in Computer Science*, 94–112. Springer.

Jacobs, S.; Basset, N.; Bloem, R.; Brenguier, R.; Colange, M.; Faymonville, P.; Finkbeiner, B.; Khalimov, A.; Klein, F.; Michaud, T.; Pérez, G. A.; Raskin, J.; Sankur, O.; and Tentrup, L. 2017. The 4th reactive synthesis competition (SYNTCOMP 2017): Benchmarks, participants & results. In *Proc. of the 6th Workshop on Synthesis (SYNT@CAV 2017)*, volume 260 of *EPTCS*, 116–143.

Janota, M.; Klieber, W.; Marques-Silva, J.; and Clarke, E. M. 2016. Solving QBF with counterexample guided refinement. *Artif. Intell.* 234:1–25.

Kim, K.; Corsi, D.; Rodríguez, A.; Lanier, J.; Parellada, B.; Baldi, P.; Sánchez, C.; and Fox, R. 2025. Realizable continuous-space shields for safe reinforcement learning. In *Proc. of the 7th Annual Learning for Dynamics & Control Conference (L4DC'25)*, PMLR.

Maler, O., and Nickovic, D. 2004. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Joint International Conferences on Formal Modelling and Analysis of Timed Systems, (FORMATS 2004) and Formal Techniques in Real-Time and Fault-Tolerant Systems, (FTRTFT 2004), Grenoble, France, September 22-24, 2004, Proceedings*, volume 3253 of *Lecture Notes in Computer Science*, 152–166. Springer.

Manna, Z., and Pnueli, A. 1995. *Temporal verification of reactive systems - safety*. Springer.

Marchesini, E., and Farinelli, A. 2020. Discrete deep reinforcement learning for mapless navigation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*.

Peltomäki, J., and Porres, I. 2022. Falsification of multiple requirements for cyber-physical systems using online generative adversarial networks and multi-armed bandits. In *Proc. of the 15th IEEE International Conference on Software Testing, Verification and Validation Workshops, (ICST Workshops 2022), Valencia, Spain, April 4-13, 2022*, 21–28. IEEE.

Pnueli, A., and Rosner, R. 1989a. On the synthesis of a reactive module. In *Proc. of the 16th Annual ACM Symp. on Principles of Programming Languages (POPL'89)*, 179–190. ACM Press.

Pnueli, A., and Rosner, R. 1989b. On the synthesis of an asynchronous reactive module. In *Proc. of the 16th Int'l Colloqium on Automata, Languages and Programming (ICALP'89)*, volume 372 of *LNCS*, 652–671. Springer.

Pnueli, A. 1977. The temporal logic of programs. *Proc. of the 18th Annual Symposium on Foundations of Computer Science (FOCS 1977)* 46–57.

Reynolds, A.; Deters, M.; Kuncak, V.; Tinelli, C.; and Barrett, C. W. 2015. Counterexample-guided quantifier instantiation for synthesis in SMT. In *Proc. of the 27th International Conference on Computer Aided Verification (CAV 2015), San Francisco, CA, USA, July 18-24, 2015*, volume 9207 of *Lecture Notes in Computer Science*, 198–216. Springer.

Rodríguez, A., and Sánchez, C. 2023. Boolean Abstractions for Realizability Modulo Theories. In *Proc. of the 35th International Conference on Computer Aided Verification (CAV'23)*, volume 13966 of *LNCS*. Springer, Cham.

Rodriguez, A., and Sánchez, C. 2024a. Adaptive reactive synthesis for LTL and LTLf modulo theories. In *Proc. of the 38th AAAI Conf. on Artificial Intelligence (AAAI'24)*. AAAI Press.

Rodríguez, A., and Sánchez, C. 2024b. Realizability modulo theories. *J. Log. Algebraic Methods Program. (JLAMP)* 140:100971.

Rodriguez, A.; Amir, G.; Corsi, D.; Sánchez, C.; and Katz, G. 2025. Shield synthesis for LTL modulo theories. In *Proc. of the 39th AAAI Conf. on Artificial Intelligence (AAAI'25)*. AAAI Press.

Rodríguez, A.; Gorostiaga, F.; and Sánchez, C. 2024. Predictable and performant reactive synthesis modulo theories via functional synthesis. In *Proc. of the 22nd International Symposium on Automated Technology for Verification and Analysis (ATVA 2024), Part II*, volume 15055 of *LNCS*, 28–50. Springer.

Rodríguez, A.; Gorostiaga, F.; and Sánchez, C. 2025. Counter Example Guided Reactive Synthesis for LTL Modulo Theories. In *Proc. of the 37th International Conference on Computer Aided Verification (CAV'25)*, LNCS.

Schewe, S., and Finkbeiner, B. 2007. Bounded synthesis. In *Proc. of the 5th International Symposium in Automated Technology for Verification and Analysis (ATVA 2007)*, volume 4762 of *LNCS*, 474–488. Springer.

Shaik, I., and van de Pol, J. 2022. Classical planning as QBF without grounding. In *Proc. of the 32nd International Conference on Automated Planning and Scheduling, (ICAPS 2022)*, 329–337. AAAI Press.

Shaik, I.; Heisinger, M.; Seidl, M.; and van de Pol, J. 2023. Validation of QBF encodings with winning strategies. In *26th International Conference on Theory and Applications of Satisfiability Testing, SAT 2023, July 4-8, 2023, Alghero, Italy*, volume 271 of *LIPIcs*, 24:1–24:10. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Smith, D. E., and Weld, D. S. 1998. Conformant graphplan. In *Proc. of the 15th National Conference on Artificial Intelligence (AAAI 98)*, 889–896. AAAI Press / The MIT Press.

Wu, M.; Wang, J.; Deshmukh, J.; and Wang, C. 2019. Shield synthesis for real: Enforcing safety in cyber-physical systems. In *Proc. of 19th Formal Methods in Computer Aided Design, (FMCAD'19)*, 129–137. IEEE.

A First-Order Logic Background

A.1 Satisfiability

The (NP-complete) Boolean Satisfiability Problem (SAT) is the decision problems of determining whether a propositional formula has a satisfying assignment. Satisfiability Modulo Theories (SMT) (Barrett and Tinelli 2018) consists on determining whether a first-order formula is satisfiable using literals from background theories. Background theories of interest include arithmetic, arrays, bit-vectors, inductive data types, uninterpreted functions and combinations of these. SMT solving typically deals with existentially quantifier theory variables, but it has also been extended with capabilities to handle universal quantifiers, but typically at the expense of semi-decidability. In this paper, we call Q-SMT to this quantified SMT extension and we consider theories for which satisfiability in the $\exists^*\forall^*$ fragment is decidable (which includes e.g., linear integer arithmetic and nonlinear real arithmetic), because these are the ones for which Boolean abstraction guarantees termination.

Quantified Boolean formulae (QBF) extend propositional formulae by allowing arbitrary quantification over the propositional variables. Unlike SAT and (standard) SMT solving, QBF solving deals with both existentially and universally quantified Boolean variables. Unlike SMT, QBF does not consider theory predicates.

A.2 Skolem Functions

A Skolem function is a concept that plays a crucial role in the elimination of quantifier alternation in logical formulas.

Definition 8. *Let $A(x_1, \ldots, x_n, y)$ be a predicate formula with individual variables $x_1, \ldots, x_n, y$ whose domains are sets $X_1, \ldots, X_n, Y$, respectively. A function*

$$f : X_1 \times \cdots \times X_n \to Y$$

is called a Skolem function for the formula

$$\exists y \, A(x_1, \ldots, x_n, y)$$

if, for all $x_1 \in X_1, \ldots, x_n \in X_n$, the following holds:

$$\exists y \, A(x_1, \ldots, x_n, y) \Rightarrow A(x_1, \ldots, x_n, f(x_1, \ldots, x_n)).$$

Skolem functions are utilized to transform formulas into a form free from the alternation of the quantifiers $\forall$ and $\exists$. For any formula A in the language of restricted predicate calculus, one can construct a formula in the Skolem normal form:

$$\exists x_1, \ldots, x_n \, \forall y_1, \ldots, y_m \, C,$$

where C does not contain new quantifiers but includes new function symbols. The original formula A is deducible in predicate calculus if and only if its Skolem normal form is.

B Proofs

We now show proof sketched of the theorems in the paper. Proof of Thm. 1 is as follows:

Proof. Let m be such that $\varphi_{\mathcal{T}}^{\text{smt}}$ is SAT and let $e_0, \ldots, e_m$ be a model of $\varphi_{\mathcal{T}}^{\text{smt}}$. The unconditional strategy $\rho^{\mathbb{E}}$ that plays first e_0, then e_1, etc up to e_m is winning for the environment because it falsifies $\varphi_{\mathcal{T}}$. Note that the universal quantifiers for $y_0, \ldots, y_m$ guarantee that all moves of the system for the first m steps are considered and in all cases $\varphi_{\mathcal{T}}$ is falsified. Since $\rho^{\mathbb{E}}$ is winning, then $\varphi_{\mathcal{T}}$ is unrealizable. □

Note that Thm. 2, Thm. 4 and Thm. 6 follow analogously.

Proof of Thm. 3 is as follows, with an auxiliary lemma:

Lemma 1 (From e_k to x)**.** *Let $X = [x_0, x_1, ...]$, $Y = [y_0, y_1, ...]$, $E = [e_0, e_1, ...]$ and $S = [e_0, e_1, ...]$. Consider $\varphi_{\mathcal{T}}(X, Y)$ be an $\text{LTL}_{\mathcal{T}}$ formula and $\varphi_{\mathbb{B}}(E, S)$ be its Boolean abstraction. Let us denote with $v_a : val(A)$, where A is a set of variables. For every $e \in E$ there is a computable satisfiable predicate $r_e(X)$ such that, for every valuation $v_x \in val(X)$ with $r_e(v_x)$,*

- *let $v_y : val(A)$ be arbitrary, then the valuation $v_s : val(S)$ such that $s(s_i)$ is true if and only if $l_i(v_x, v_y)$ holds $\varphi^{extra}(E, S)$.*
- *let $v_s : val(S)$ be such that $\varphi^{extra}(E, S)$ holds, then there is a valuation $v_y : val(Y)$ that makes $l_i(v_x, v_y)$ hold if and only if $s(s_i)$.*

Moreover, for every $v_x : val(X)$ there is exactly one $e \in E$ such that $r_e(v_x)$ holds.

Proof. The main idea is to create the corresponding sequence using Lemma 1. It follows that if the system in $\varphi_{\mathcal{T}}$ can make a collection of literals at some point then the system can make the corresponding s_i hold at the same point (and viceversa). By structural induction, if the atoms have the same valuation then all sub-formulae have the same valuation. Therefore, given an arbitrary length l, if the unconditional strategy $\rho_{k=l}$ in $\varphi_{\mathcal{T}}$ is winning for a player the strategy $\rho_{k=l}$ in $\varphi_{\mathbb{B}}$ is winning for a player as well, and viceversa. □

Note that Thm. 8 follows analogously.

Proof of Thm. 5 is as follows:

Proof. Since $\varphi_{\mathcal{T}}$ is unrealizable, there is some reachability goal that the environment satisfies. Moreover, since $\varphi_{\mathcal{T}}$ is in safety, then this goal is reached in a finite number of steps. □

Note that Thm. 7 follows analogously.

C QBF Encoding

We now show the QBF encoding for our running example.

C.1 Unconditional Encoding

The formula is as follows:

$$
\begin{array}{lr}
\exists e_0^0, e_1^0, e_2^0, e_3^0 & \\
\exists e_0^1, e_1^1, e_2^1, e_3^1 & \\
\exists e_0^2, e_1^2, e_2^2, e_3^2 & \\
\forall s^0, s^1, s^2 & \\
\neg. & \\
\varphi_0 : & e_0^0 \rightarrow s^2 \\
 & e_1^0 \rightarrow \neg s^0 \\
 & e_2^0 \rightarrow (s^0 \leftrightarrow s^1) \\
 & e_3^0 \rightarrow s^0 \\
\wedge & \\
\varphi_1 : & e_0^1 \rightarrow \top \\
 & e_1^1 \rightarrow \neg s^1 \\
 & e_2^1 \rightarrow (s^1 \leftrightarrow s^2) \\
 & e_3^1 \rightarrow s^1 \\
\wedge & \\
\varphi_2 : & e_0^2 \rightarrow \top \\
 & e_1^2 \rightarrow \neg s^2 \\
 & e_2^2 \rightarrow (s^2 \leftrightarrow \top) \\
 & e_3^2 \rightarrow s^2
\end{array}
$$

In order to solve it, we encode it in the QCIR format:

```
exists(1, 2, 3, 4)
exists(5, 6, 7, 8)
exists(9, 10, 11, 12)
forall(13, 14, 15)
output(-40)
#\varphi_0
16 = or(-1 , 15)
17 = or(-2 , - 13)
18 = or(13, -14)
19 = or(-13, 14)
20 = and(18, 19)
21 = or(-3, 20)
22 = or(-4 , 13)
23 = and(16, 17, 18, 19, 20, 21, 22)
#\varphi_1:
24 = and()
25 = or(-6 , - 14)
26 = or(14, -15)
27 = or(-14, 15)
28 = and(26,27)
29 = or(-7, 28)
30 = or(-8 , 14)
31 = and(24, 25, 26, 27, 28, 29, 30)
#\varphi_2:
32 = and()
33 = or(-10 , - 15)
34 = or(15)
35 = and()
36 = and(34, 35)
37 = or(-11, 36)
38 = or(-12 , 15)
39 = and(32, 33, 34, 35, 36, 37, 38)
# conjunction:
40 = and(23, 31, 39)
```

C.2 Conditional Encoding

The formula is as follows:

$$
\begin{array}{lr}
\exists e_0^0, e_1^0, e_2^0, e_3^0 & \\
\forall s^0, \exists s_c^0 & \\
\exists e_0^1, e_1^1, e_2^1, e_3^1 & \\
\forall s^1, \exists s_c^1 & \\
s^0 \leftrightarrow s_c^0 & \\
s^1 \leftrightarrow s_c^1 & \\
\neg. & \\
\varphi_0 : & e_0^0 \rightarrow \top \\
 & e_1^0 \rightarrow \neg s_c^0 \\
 & e_2^0 \rightarrow (s_c^0 \leftrightarrow s_c^1) \\
 & e_3^0 \rightarrow s_c^0 \\
\wedge & \\
\varphi_1 : & e_0^1 \rightarrow \top \\
 & e_1^1 \rightarrow \neg s^1 \\
 & e_2^1 \rightarrow (s_c^1 \leftrightarrow \top) \\
 & e_3^1 \rightarrow s^1
\end{array}
$$

Again, we encode it in QCIR:

```
exists(1, 2, 3, 4)
forall(13)
exists(9)
exists(5, 6, 7, 8)
forall(14)
exists(10)
output(55)
#\varphi_0
17 = or(-2 , -13)
18 = or(13, -14)
19 = or(-13, 14)
20 = and(18, 19)
21 = or(-3, 20)
22 = or(-4 , 13)
23 = and(17, 21, 22)
#\varphi_1:
25 = or(-6 , -14)
```

```
29 = or(-7, 14)
30 = or(-8 , 14)
31 = and(25, 29, 30)
# conjunction:
32 = and(23, 31)
# mutual exclusion, 1,2,3,4:
33 = or(1, 2, 3, 4)
34 = or(-1, -2)
35 = or(-1, -3)
36 = or(-1 , -4)
37 = or(-2, -3)
38 = or(-2, -4)
39 = or(-3, -4)
40 = and(33, 34, 35, 36, 37, 38, 39)
# mutual exclusion, 5,6,7,8:
41 = or(5, 6, 7, 8)
42 = or(-5, -6)
43 = or(-5, -7)
44 = or(-5 , -8)
45 = or(-6, -7)
46 = or(-6, -8)
47 = or(-7, -8)
48 = and(41, 42, 43, 44, 45, 46, 47)
# equality:
49 = or(-9, 13)
50 = or(-13, 9)
51 = and(49, 50)
52 = or(-10, 14)
53 = or(-14, 10)
54 = and(52, 53)
#output:
55 = and(-32, 40, 48, 51, 54)
```

C.3 Handling System Moves with existential variables

Unrealizability can be encoded as a 2-player turn-based game where environment is the first (existential) player and system is the second (universal) player. A 2-player game can be elegantly encoded as a QBF. For encoding environment, one can use existential variables since we only need a single winning move in each turn. For System, on the otherhand, we need to encode all possible moves. Thus, we can use universal variables to represent all the moves. The goal of the environment player is then to *win the game* i.e., make the formula true regardless of the opponent moves. Now, when translating in a QBF encoding, one cannot disable any turn of universal player. For example, consider the following formula where $0 \leq i \leq n$:

$$\exists e_0, \forall s_0, ...\exists e_n, \forall e_n \neg(\varphi \land (e_i \rightarrow \neg s_i))$$

Regardless of the formula φ, setting e_i to True makes the formula True. Essentially, the above formula encodes that *If environment plays e_i move, then do not allow the s_i move for the system.* However, a QBF solver (by construction) will search the complete search space i.e., all possible moves of the universal player. Disallowing or forcing a move of the universal player simply makes the formula false (or true if negated). While one can construct a custom QBF solver to disallow or force moves of universal player, we cannot use an off-the-shelf QBF solver.

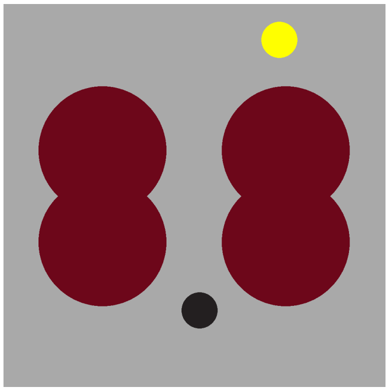

Figure 2: Screenshot of the configuration employed for the main set of experiments.

Alternatively, one can use existential variables for each universal variable to encode the force or disallow the moves on the universal player. For example, let us extend our previous formula $s_0^c, ..., s_n^c$ existential variables as follows:

$$\exists e_0, \forall s_0, ...\exists e_n, \forall e_n \tag{1}$$

$$\exists s_0^c, ..., s_n^c \tag{2}$$

$$\bigwedge_{k=0}^{n} (s_k \leftrightarrow s_k^c) \tag{3}$$

$$\neg(\varphi \land (e_i \rightarrow \neg s_i^c)) \tag{4}$$

Equations 2 and 3 defines existential variables and forces equalities between corresponding universal and existential system variables. Unlike previous encoding, we allow all possible moves of the universal player. At the same time, when environment plays e_i, we encode that s_i is an invalid move and thus the formula becomes True (by negation) only in that particular move but not others. The new formula is not trivially true, it now depends on the φ. While this technique adds one additional quantifier layer, notices that the search space is not increased. Due do our equality clauses in the equation 3 by unit clause propagation the search space remains same. It is also possible to push existential system variables to the layers directly below corresponding universal variables as follows:

$$\exists e_0, \forall s_0, \exists s_0^c ...\exists e_n, \forall e_n, \exists s_n^c \tag{5}$$

$$\bigwedge_{k=0}^{n} (s_k \leftrightarrow s_k^c) \tag{6}$$

$$\neg(\varphi \land (e_i \rightarrow \neg s_i^c)) \tag{7}$$

Notice that the number of quantifier layers still only increased by 1. Similar techniques are used in encoding two-player games as QBF, for instance handling illegal moves (Shaik et al. 2023).

D The Drone2D Environment

The **Drone2D Environment** (Fig.2) is a lightweight continuous control environment where a drone navigates a bounded 2D space to reach a randomly assigned goal position while avoiding collisions with obstacles. The agent's state includes its current position (x, y), velocity (v_x, v_y), and the goal's coordinates $(x_{\text{goal}}, y_{\text{goal}})$. The agent's actions represent accelerations (a_x, a_y) in the x- and y-directions. The environment simulates realistic dynamics with position and velocity updates, and the agent is rewarded for minimizing its distance to the goal.

- **Unsafe regions** (red circles): The drone cannot cross these zones. A collision with these obstacles results in episode termination with failure.
- **Goal** (yellow circle): The target the agent must reach.
- **Drone** (black circle): The visual representation of the drone.

D.1 State Space

The state of the drone at time t is represented as:

$$s_t = [x_t, y_t, x_{\text{goal}}, y_{\text{goal}}, w_x, w_y]$$

where:

- x_t, y_t: Current position of the drone.
- $x_{\text{goal}}, y_{\text{goal}}$: Coordinates of the goal position.
- w_x, w_y: Turbulence force affecting the drone's motion along both axes.

The state space is defined as:

$$x_t, y_t, x_{\text{goal}}, y_{\text{goal}} \in [0, 21], \quad w_x, w_y \in \mathbb{R}$$

D.2 Action Space

The action at time t is represented as:

$$\mathbf{a}_t = [a_t^x, a_t^y]$$

where a_t^x and a_t^y are accelerations applied to the drone in the x- and y-directions, respectively. The actions are continuous and bounded as:

$$a_t^x, a_t^y \in [-1, 1]$$

D.3 Dynamics

The drone's position and velocity evolve over discrete time steps according to the following equations:

$$x_{t+1} = x_t + a_{x,t} \cdot \Delta t + w_x$$

$$y_{t+1} = y_t + a_{y,t} \cdot \Delta t + w_y$$

where the time step is fixed at:

$$\Delta t = 0.5$$

D.4 Sampling Turbulence (Environment Action)

In this environment, turbulence is represented as an external force affecting the agent's velocity. The turbulence at each time step is defined as a vector $\mathbf{w}_t = [w_x, w_y]$, where w_x and w_y represent the force along the x- and y-axes, respectively.

The turbulence is sampled at each time step using the following rule:

$$w_x, w_y \sim \mathcal{U}(-\sigma, \sigma)$$

where:

- $\mathcal{U}(-\sigma, \sigma)$ denotes a uniform distribution.
- σ is the maximum magnitude of turbulence along each axis (default $\sigma = 0.6$).

D.5 Reward Function

The reward function encourages the drone to minimize its distance to the target while applying a small constant penalty to incentivize faster completion of the task. The reward at time t, denoted as r_t, is defined as:

$$r_t = -(\alpha \cdot d_t) - \beta$$

where d_t is the Euclidean distance between the current position of the drone (x_t, y_t) and the target position $(x_{\text{goal}}, y_{\text{goal}})$, calculated as:

$$d_t = \sqrt{(x_t - x_{\text{goal}})^2 + (y_t - y_{\text{goal}})^2}$$

finally, α is a scaling factor for the distance penalty (set to $\alpha = 0.0005$ by default) and β is a small constant penalty to encourage the agent to reach the target quickly (set to $\beta = 0.0001$ by default). This reward function ensures that:

1. The agent receives larger penalties for being far from the target (d_t).
2. The small constant penalty (β) encourages the agent to minimize the number of time steps taken to reach the target.

D.6 Safety Constraint

For the main set of experiments, the environment contains four predefined unsafe zones:

1. A circle centered at $(5.5, 8.0)$ with a radius of 3.5.
2. A circle centered at $(5.5, 13.0)$ with a radius of 3.5.
3. A circle centered at $(15.5, 8.0)$ with a radius of 3.5.
4. A circle centered at $(15.5, 13.0)$ with a radius of 3.5.

The union of these regions forms the total unsafe zone $\mathcal{Z}$, defined as $\mathcal{Z} = \{(x, y) \mid \sqrt{(x - 5.5)^2 + (y - 8.0)^2} \leq 3.5 \wedge ...\}$[3]

Let $\mathbf{1}_{\mathcal{Z}}(x_t, y_t)$ be an indicator function that is 1 if the agent is in the unsafe zone at time t, and 0 otherwise:

$$\mathbf{1}_{\mathcal{Z}}(x_t, y_t) = \begin{cases} 1 & \text{if } (x_t, y_t) \in \mathcal{Z}, \\ 0 & \text{otherwise.} \end{cases}$$

[3] Repeated for each obstacle (omitted for calrity).

The agent must satisfy the following constraint:

$$\sum_{t'=t}^{t+4} \mathbf{1}_{\mathcal{Z}}(x_{t'}, y_{t'}) \leq 1$$

which ensures that the agent does not remain in an unsafe region for more than one consecutive time steps.

E Supplement for the Main Text

E.1 Figures of Automatas.

We now show automatas for specifications of Ex. 2 and Ex. 9.

Moreover, the explainability problem from synthetised automata quickly becomes worse as we increase the complexity of the specifications. For instance, consider the following slightly more complex specification:

$$\varphi_{\mathcal{T}} : \Box \left(\begin{array}{ccc} \Diamond(x < 10) & \rightarrow & \bigcirc^{2}(y > 9) \\ & \wedge & \\ (x \geq 10) & \rightarrow & \Diamond(y \leq x), \end{array} \right)$$

where the green colour denotes the changes with respect to the specification of Ex. 2.

The strategy for this specification is depicted in Fig. 5, where we can see that the difficulty of extracting explanations from Mealy machines gets impossible even with reasonable specifications.

E.2 Missing Algorithms

Find below Alg. 4.

Algorithm 4 Cond. bounded unrealiz. check for LTL.

```
Require: $\varphi_{\mathbb{B}}$, $max$, $E$, $S$
$T \leftarrow \emptyset$
for $n = 1$ to $max$ do
    $[s_0, \dots, s_n] \leftarrow copies(S, n)$
    $[e_0, \dots, e_n] \leftarrow copies(E, n)$
    $F \leftarrow unroll(\varphi_{\mathbb{B}}, n)$
    $G \leftarrow \texttt{alternations}([s_0, \dots, s_n], [e_0, \dots, e_n]).F$
    $\varphi_{\mathbb{B}}^{\text{qbf}} \leftarrow QElim([s_0, \dots, s_n], G)$
    if $\neg\varphi_{\mathbb{B}}^{\text{qbf}}$ is SAT then
        return $(\texttt{true}, witness(\varphi_{\mathbb{B}}^{\text{qbf}}))$
    end if
end for
return unreal
```

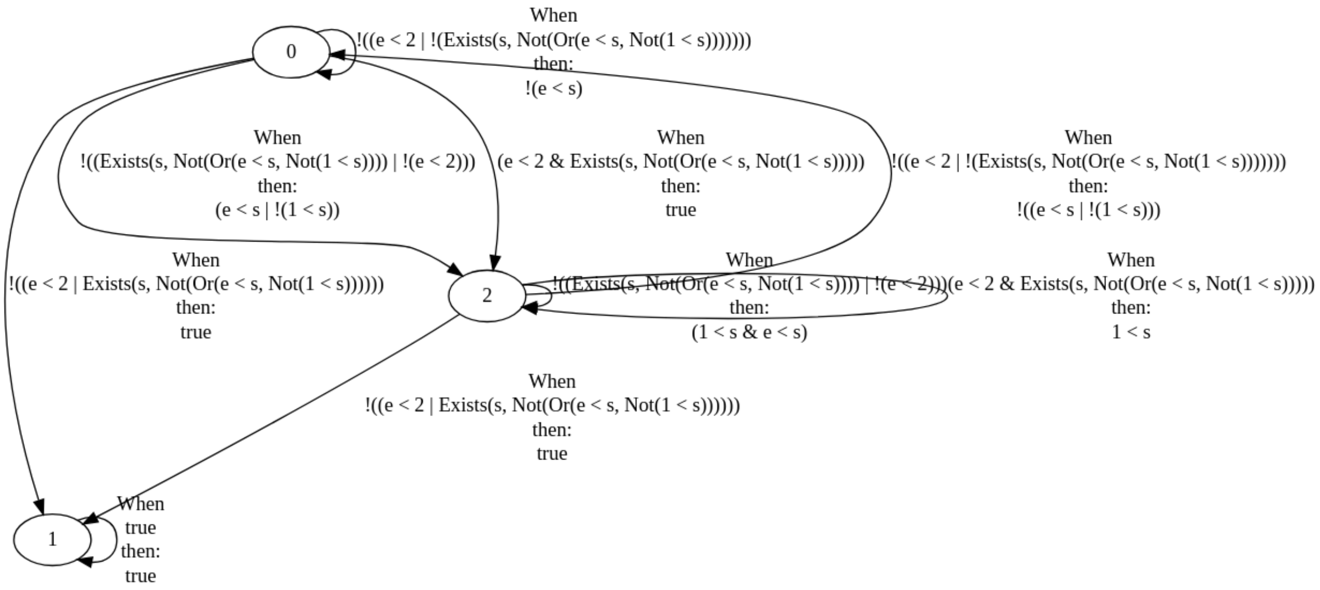
0
When
!((e < 2 | !(Exists(s, Not(Or(e < s, Not(1 < s)))))))
then:
!(e < s)
When
!((Exists(s, Not(Or(e < s, Not(1 < s)))) | !(e < 2)))
then:
(e < s | !(1 < s))
When
(e < 2 & Exists(s, Not(Or(e < s, Not(1 < s)))))
then:
true
When
!((e < 2 | !(Exists(s, Not(Or(e < s, Not(1 < s)))))))
then:
!((e < s | !(1 < s)))
When
!((e < 2 | Exists(s, Not(Or(e < s, Not(1 < s))))))
then:
true
2
When
!((Exists(s, Not(Or(e < s, Not(1 < s)))) | !(e < 2)))
then:
(1 < s & e < s)
When
(e < 2 & Exists(s, Not(Or(e < s, Not(1 < s)))))
then:
1 < s
When
!((e < 2 | Exists(s, Not(Or(e < s, Not(1 < s))))))
then:
true
1
When
true
then:
true

Figure 3: Synthetised strategy for Ex. 1. Note that x of Ex. 1 is here e and y is s.

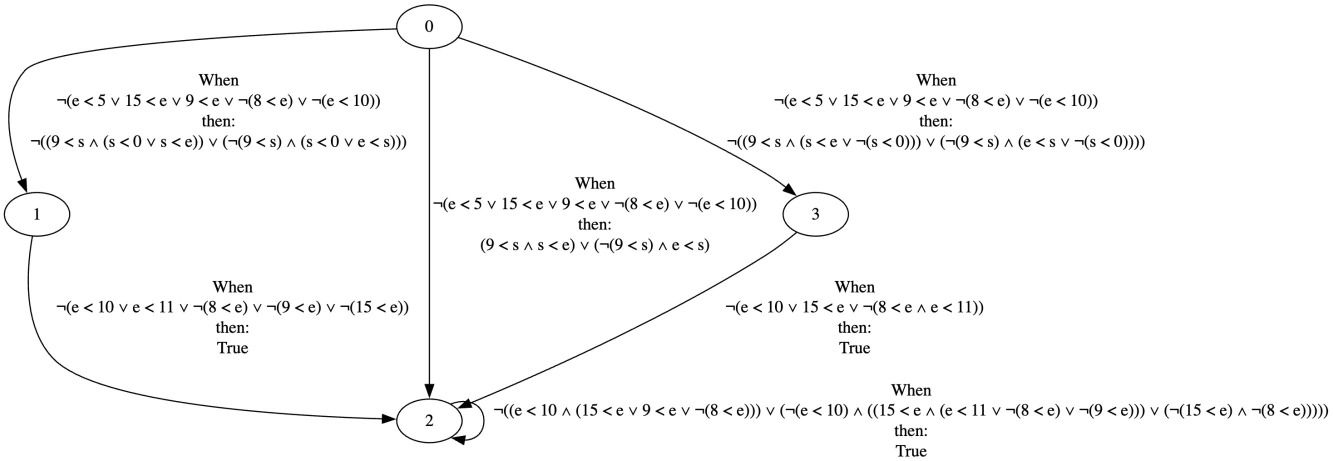
0
When
¬(e < 5 ∨ 15 < e ∨ 9 < e ∨ ¬(8 < e) ∨ ¬(e < 10))
then:
¬((9 < s ∧ (s < 0 ∨ s < e)) ∨ (¬(9 < s) ∧ (s < 0 ∨ e < s)))
When
¬(e < 5 ∨ 15 < e ∨ 9 < e ∨ ¬(8 < e) ∨ ¬(e < 10))
then:
¬((9 < s ∧ (s < e ∨ ¬(s < 0))) ∨ (¬(9 < s) ∧ (e < s ∨ ¬(s < 0))))
1
When
¬(e < 5 ∨ 15 < e ∨ 9 < e ∨ ¬(8 < e) ∨ ¬(e < 10))
then:
(9 < s ∧ s < e) ∨ (¬(9 < s) ∧ e < s)
3
When
¬(e < 10 ∨ e < 11 ∨ ¬(8 < e) ∨ ¬(9 < e) ∨ ¬(15 < e))
then:
True
When
¬(e < 10 ∨ 15 < e ∨ ¬(8 < e ∧ e < 11))
then:
True
When
¬((e < 10 ∧ (15 < e ∨ 9 < e ∨ ¬(8 < e))) ∨ (¬(e < 10) ∧ ((15 < e ∧ (e < 11 ∨ ¬(8 < e) ∨ ¬(9 < e))) ∨ (¬(15 < e) ∧ ¬(8 < e)))))
then:
True
2

Figure 4: Synthetised strategy for Ex. 9

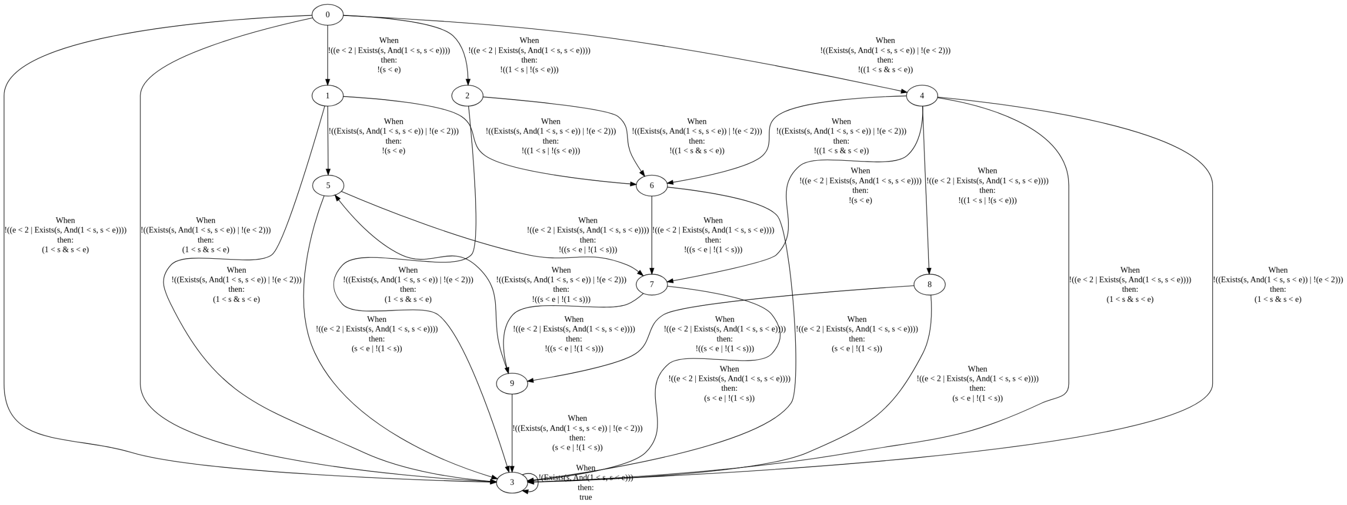
0
When
!((e < 2 | Exists(s, And(1 < s, s < e))))
then:
!(s < e)
When
!((e < 2 | Exists(s, And(1 < s, s < e))))
then:
!((1 < s | !(s < e)))
When
!((Exists(s, And(1 < s, s < e)) | !(e < 2)))
then:
!((1 < s & s < e))
1
2
4
When
!((Exists(s, And(1 < s, s < e)) | !(e < 2)))
then:
!(s < e)
When
!((Exists(s, And(1 < s, s < e)) | !(e < 2)))
then:
!((1 < s | !(s < e)))
When
!((Exists(s, And(1 < s, s < e)) | !(e < 2)))
then:
!((1 < s & s < e))
When
!((Exists(s, And(1 < s, s < e)) | !(e < 2)))
then:
!((1 < s & s < e))
When
!((e < 2 | Exists(s, And(1 < s, s < e))))
then:
!(s < e)
When
!((e < 2 | Exists(s, And(1 < s, s < e))))
then:
!((1 < s | !(s < e)))
5
6
When
!((e < 2 | Exists(s, And(1 < s, s < e))))
then:
(1 < s & s < e)
When
!((Exists(s, And(1 < s, s < e)) | !(e < 2)))
then:
(1 < s & s < e)
When
!((e < 2 | Exists(s, And(1 < s, s < e))))
then:
!((s < e | !(1 < s)))
When
!((e < 2 | Exists(s, And(1 < s, s < e))))
then:
!((s < e | !(1 < s)))
When
!((Exists(s, And(1 < s, s < e)) | !(e < 2)))
then:
(1 < s & s < e)
When
!((Exists(s, And(1 < s, s < e)) | !(e < 2)))
then:
(1 < s & s < e)
When
!((Exists(s, And(1 < s, s < e)) | !(e < 2)))
then:
!((s < e | !(1 < s)))
7
8
When
!((e < 2 | Exists(s, And(1 < s, s < e))))
then:
(1 < s & s < e)
When
!((Exists(s, And(1 < s, s < e)) | !(e < 2)))
then:
(1 < s & s < e)
When
!((e < 2 | Exists(s, And(1 < s, s < e))))
then:
(s < e | !(1 < s))
When
!((e < 2 | Exists(s, And(1 < s, s < e))))
then:
!((s < e | !(1 < s)))
When
!((e < 2 | Exists(s, And(1 < s, s < e))))
then:
!((s < e | !(1 < s)))
When
!((e < 2 | Exists(s, And(1 < s, s < e))))
then:
(s < e | !(1 < s))
When
!((e < 2 | Exists(s, And(1 < s, s < e))))
then:
(s < e | !(1 < s))
When
!((e < 2 | Exists(s, And(1 < s, s < e))))
then:
(s < e | !(1 < s))
9
When
!((Exists(s, And(1 < s, s < e)) | !(e < 2)))
then:
(s < e | !(1 < s))
3
When
!(Exists(s, And(1 < s, s < e)))
then:
true

Figure 5: A large automata (note that it is supposed to be unintelligible).