




Tableaux for Realizability of Safety Specifications*

Montserrat Hermo¹, Paqui Lucio¹, and César Sánchez²

¹ University of the Basque Country, San Sebastián, Spain

² IMDEA Software Institute, Madrid, Spain

Abstract. We introduce a tableau decision method for deciding realizability of specifications expressed in a safety fragment of LTL that includes bounded future temporal operators. Tableau decision procedures for temporal and modal logics have been thoroughly studied for satisfiability and for translating temporal formulae into equivalent Büchi automata, and also for model checking, where a specification and system are provided. However, to the best of our knowledge no tableau method has been studied for the reactive synthesis problem.

Reactive synthesis starts from a specification where propositional variables are split into those controlled by the environment and those controlled by the system, and consists on automatically producing a system that guarantees the specification for all environments. Realizability is the decision problem of whether there is one such system.

In this paper, we present a method to decide realizability of safety specifications, from which we can also extract (i.e., synthesize) a correct system (in case the specification is realizable). The main novelty of a tableau method is that it can be easily extended to handle richer domains (integers, etc) and bounds in the temporal operators in ways that automata approaches for synthesis cannot.

1 Introduction

Linear Temporal Logic (LTL) [27] is modal logic for expressing correctness properties of reactive systems. Verification is the problem of deciding, given a system S and an LTL specification φ , whether S models φ . Reactive synthesis, first studied by Pnueli and Rosner in 1989 [29,28], is the problem of automatically producing S from φ with the guarantee that S models φ . In the reactive synthesis problem, the atomic variables are split into those variables controlled by the environment and the rest, controlled by the system.

* This work was funded in part by the European Union (ERDF funds) under grant PID2020-112581GB-C22, European COST Action CA20111 EuroProofNet (European Research Network on Formal Proofs), by the University of the Basque Country under project LoRea GIU21/044, by the Madrid Regional Government under project S2018/TCS-4339 (BLOQUES-CM) and by a research grant from Nomadic Labs and the Tezos Foundation.

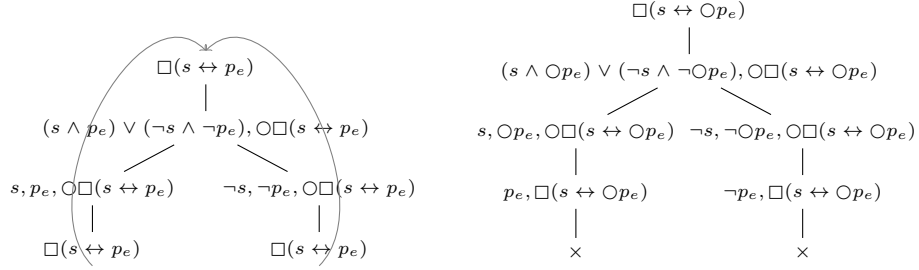


Fig. 1. Tableaux for $\Box\psi_1$ and $\Box\psi_2$.

In the last two decades, the reactive synthesis problem has received lot of attention (e.g., [5,10,15,16,21]). The approaches can be classified into three categories: (1) game-based [7], (2) approaches that cover a strict fragment of LTL, like GR(1) specifications [26,3]; (3) bounded synthesis [30], which explores the problem up to a fixed bound on the size of the system. In all these cases, the state space of the game arena is either captured by an automaton or explored explicitly or symbolically. In this paper, we study a deductive alternative: a tableau method for the realizability and synthesis for the class of safety specifications.

Tableau methods were originally created [2,33] as intuitive deduction procedures for classical propositional and first-order logic. A tableau is a tree that performs symbolic handling of formulas according to simple rules based on semantics, model-theory and proof-theory. Classical tableaux correspond to deductive proofs in Gentzen's sequent calculus. Tableaux have been evolving for years to decide the satisfiability problem of many other non-classical logics (modal, multi-valued, temporal, etc.), in some cases combined with other formal structures, such as different kinds of automata.

Traditional tableau techniques for satisfiability do not directly work for realizability, where tableaux have only been used for auxiliary steps in automata-based methods [6]. We present in this paper, a tableau-based method for the realizability of reactive safety specifications. To illustrate the problem, consider the following formulas where p_e is an environment variable and s is a system variable: $\psi_1 = s \leftrightarrow p_e$, $\psi_2 = s \leftrightarrow Op_e$ and $\psi_3 = Os \leftrightarrow Op_e$. Symbols \bigcirc and \Box are temporal operators which refer to the next instant and to all instants of time respectively. The safety specifications $\Box\psi_1$ and $\Box\psi_3$ are realizable: consider the system that mimics in s the value observed in e .

A temporal tableau for $\Box\psi_1$ (shown in Fig. 1 (left)) first uses the semantics of the \Box operator, which states that $\Box\psi_1 = \psi_1 \wedge \bigcirc\Box\psi_1$. Then, it decomposes the formula into $s \leftrightarrow p_e, \bigcirc\Box\psi_1$ and splits two branches for the two cases: $s, p_e, \bigcirc\Box\psi_1$ and $\neg s, \neg p_e, \bigcirc\Box\psi_1$. Both nodes then jump to the next temporal state, so both branches generate a loop to the root $\Box\psi_1$. Each branch represents a model of the initial formula. It is tempting to interpret this tableau as a winning strategy for the system that witnesses the realizability of $\Box\psi_1$. On the other hand, $\Box\psi_2$ is not realizable, as the system is required to guess the next value of p_e , and the environment can later emit the opposite value. The tableau for $\Box\psi_2$ is shown in

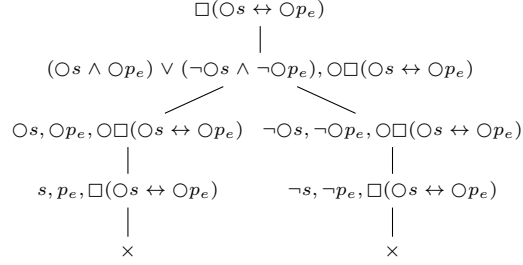


Fig. 2. Tableau for $\Box\psi_3$.

Fig. 1 (right). The left branch in the tableau corresponds to the system choosing s hoping for the environment to play p_e in the next step. Since the environment can choose $\neg p_e$, this branch must close at node $p_e, \Box\psi_2$ (the right branch is similar). A branch closing condition typical of tableaux closes this branch as the environment wins by forcing a contradiction. However, this closing condition fails to capture the realizability of $\Box\psi_3$, since the resulting tableau for $\Box\psi_3$ would be as shown in Fig. 2.

The previous closing condition would close the left branch (choosing $\neg p_e$) and the right branch (choosing p_e), incorrectly concluding that $\Box\psi_3$ is unrealizable. The problem here is in the splitting of the two cases $\circ s, \circ p_e$ and $\neg \circ s, \neg \circ p_e$, which reveals too early the future move of the system given the power (incorrectly) to the environment to create a contradiction. To overcome this problem, we introduce in this paper the *terse normal form* of formulas which prevents these incorrect splittings on formulas that reveal future choices too early. Intuitively, at the second temporal state, our tableau will just have one node $n : (s \wedge p_e) \vee (\neg s \wedge \neg p_e), \Box\psi_3$. Node n has two children (one for each choice of the environment):

$$p_e, s, \circ((s \wedge p_e) \vee (\neg s \wedge \neg p_e)), \circ \Box\psi_3 \mid \neg p_e, \neg s, \circ((s \wedge p_e) \vee (\neg s \wedge \neg p_e)), \circ \Box\psi_3$$

Then, the next state from both nodes produces again node n . This tableau encodes the proof that $\Box\psi_3$ is realizable (see Ex. 4).

We introduce in this paper *realizability tableaux* to fix classical temporal tableau rules to obtain a correct decision procedure for realizability. Our tableau method solves the realizability decision problem for a fragment of LTL, which includes temporal operators of the form $\Box_{[n,m]}$ and $\Diamond_{[n,m]}$ (for $n, m \in \mathbb{N}$). These operators are very common in industrial critical specifications where the system is supposed to respond within a predefined amount of time. Although these operators can be seen as a short-hand for a Boolean combination of formulas using only \circ , the compact notation is effectively exploited in our tableau deductions in a more efficient way that prevents exponential unfoldings. Consider for example the formula $\psi_4 = p_e \rightarrow \Box_{[0,2^{100}]}s$. Our tableau for $\Box\psi_4$ splits two branches for the two cases $(\neg p_e \wedge \circ \Box\psi_4)$ and $(p_e \wedge s \wedge \circ \Box_{[0,2^{100}-1]}s \wedge \circ \Box\psi_4)$. The first branch jumps to the next state, which loops to the root $\Box\psi_4$. The second branch

jumps to $(\Box_{[0,2^{100}-1]}s \wedge \Box\psi_4)$ which in turn spawns two new branches, both of which loop immediately to their previous state. This very small tableaux encodes the $\Box\psi_4$ is realizable. This example illustrates a crucial difference between automata and tableaux: the deductive power of the tableau, after checking two successive states, is able to decide the realizability of $\Box\psi_4$, whereas automata techniques require an explicit upfront elimination of the intervals. As far as we know, this is the first temporal tableaux for solving realizability of safety LTL specifications. Although this paper focuses on realizability, our tableaux provide a procedures for both kinds of certificates: the realizability strategy (i.e. the synthesis of a system) and the counterexample in the case of unrealizability.

In summary, our contributions are: (1) The introduction of the novel terse normal form that captures in a logical form the timely choices of the environment and the responses by the system. (2) A tableau method including all the deductive rules to build the tableau graph and rules to close the branches, with success and with failure. (3) Sound and completeness proofs for our tableau method.

Related Work. Current approaches to reactive synthesis [5,10,15,16,21] are either (1) based on games [7], which create a mathematical structure—like an automaton—that capture the game arena and then explore this structure, or (2) rely on bounded synthesis [30], which produce a set of constraints that characterizes all correct systems up to fixed bound. Modern game approaches use a symbolic representation [21], or SAT or QBF decision procedures [4]. Existing tools for full LTL synthesis, including Unbeast [10] and Acacia+ [5] are based on bounded synthesis. Different encoding of the constraint for a given bound have been proposed [14,30,15,12,22,32,13,11]. Since 2014, the reactive synthesis competition (SYNTCOMP) [1,20] compares the performance of synthesis tools against different benchmark problems.

Reactive synthesis for full LTL is 2EXPTIME-complete [29], so LTL fragments with better complexity have been identified. For example, GR(1) (general reactivity with rank 1)—enjoy an efficient (polynomial) symbolic synthesis algorithm [26,3], with practical applications [23,9]. Translating GR(1) specifications into the safety language that we consider in this paper involves at least an exponential blow-up in the worst case. All methods listed above perform an algorithmic exhaustive exploration of the game arena. In contrast, our deductive tableau method is deductive. Even though some game-based tools, like Strix [25,24], perform some on-the-fly construction of the game arena the deductive nature of tableaux allows to skip larger portions of the state space. An explicit comparison of the performance between methods requires a polished implementation, which is out of the scope of this paper. We focus here the foundations³ of the realizability tableau, emphasizing its power to handle richer settings and prevent explicit blow-ups.

The first tableau method [35] for the satisfiability of LTL is not purely tree-shape but builds a graph that is explored in a second pass. This inspired a connection with Büchi automata [35,34], on which many decision procedures [8]

³ The full proof of correctness, including all intermediate lemmas can be found in the extended version [19], which also includes several realizability tableaux examples.

for LTL satisfiability and model checking are based. The use of an auxiliary structure raised two difficulties: one is the size and another is the loss of the original correspondence with sequent proofs that could certify the result. Some alternative ideas (e.g., [31,18]) have been developed to explore on-the-fly the graph (or automaton) not requiring a second pass, and also for constructing one-pass tableaux that preserve the correspondence with sequent proofs (cf. [17]).

2 Preliminaries. Safety Specifications and Games

Given a set R , R^* denotes the set of finite strings over R and R^k the set of strings over R of length k . R^ω is the set of infinite sequences over R . We sometimes use \bar{x} to remark that string \bar{x} is a sequence of elements, and use $|\bar{x}|$ for its length and $\bar{x} \cdot v$ for the concatenation of \bar{x} with v . We use ϵ for the empty string. Given $r = r_0, r_1, r_2 \dots \in R^\omega$, and r^i for $r_i, r_{i+1} \dots$, we use $r^{<i}$ for the finite sequence r_0, \dots, r_{i-1} and $r^{i..j}$ for the finite sequence $r_i \dots r_{j-1}$. LTL extends propositional logic with temporal operators \bigcirc (next) and \mathcal{U} (until). Given a set \mathcal{V} of propositional variables, a valuation v is a map $\mathcal{V} \rightarrow \mathbb{B}$ (where \mathbb{B} is a Boolean domain). We denote by $\text{Val}(\mathcal{V})$ the set of all valuations of \mathcal{V} . A trace σ is an infinite sequence $\sigma_0, \sigma_1, \sigma_2, \dots$ of valuations of \mathcal{V} . The semantics of LTL relate formulas with traces as follows:

$$\begin{aligned} \sigma \models p & \quad \text{iff } \sigma_0(p) & \quad \sigma \models \neg\varphi & \quad \text{iff } \sigma \not\models \varphi \\ \sigma \models \bigcirc\varphi & \quad \text{iff } \sigma^1 \models \varphi & \quad \sigma \models \varphi \wedge \psi & \quad \text{iff } \sigma \models \varphi \text{ and } \sigma \models \psi \\ \sigma \models \varphi \mathcal{U} \psi & \quad \text{iff } \sigma^j \models \psi \text{ for some } 0 \leq j & \quad \text{and } \sigma^i \models \varphi \text{ for all } i & \quad \text{such that } 0 \leq i < j \end{aligned}$$

We use standard abbreviations, like \mathbf{T} for truth and \mathbf{F} for falsehood, \vee, \rightarrow and \leftrightarrow , and $\Diamond\varphi$ for $\mathbf{T}\mathcal{U}\varphi$ and $\Box\varphi$ for $\neg(\mathbf{T}\mathcal{U}\neg\varphi)$. A set of formulas is (syntactically) consistent if and only if it does not contain a formula and its negation. If $\sigma \models \varphi$ then we say that σ is a model of φ and we use $\text{Mod}(\varphi)$ to denote the set of all models of φ . We interpret a finite set of formulas as the conjunction of all its members, and use $\sigma \models \Phi$ to denote the set of traces that are models of all $\varphi \in \Phi$. A set of formulas Φ is satisfiable if and only if there exists at least one σ such that $\sigma \models \Phi$. Two formulas φ and ψ are logically equivalent, denoted $\varphi \equiv \psi$, if and only if $\text{Mod}(\varphi) = \text{Mod}(\psi)$. A set of traces L is a *safety* language whenever for any trace $\sigma \notin L$ there exists some $i > 0$ such that $\sigma^{<i} \cdot \sigma' \notin L$ for any trace σ' . We call $\sigma^{<i}$ a *witness of the violation* of σ .

Safety Specifications. We split the set of propositions in a formula φ into two disjoint subsets: \mathcal{X}_e , controlled by the environment and \mathcal{Y} , controlled by the system. We use a subscript e (e.g., *sensor_e* or *p_e*) for the elements of \mathcal{X}_e .

We use a fragment of LTL for safety specifications. To illustrate the power of our tableau technique to handle richer types, we do not restrict ourselves to Boolean variables, but also consider enumerated variables and atoms $x = c$ where x is a variable of an enumerated type T and c is a constant value of type T . Boolean formulas are built from atoms (Boolean variables or enumerated atoms) using Boolean connectives. *The fragment of safety LTL specifications consists of*

formulas $\alpha \wedge \Box\psi$, where α , called the *initial formula*, is a Boolean constraint that captures the initial states. The formula $\Box\psi$, called the safety constraint, restricts the transition relation by means of the following temporal operators:

$$\eta ::= p \mid x = c \mid \neg\eta \mid \bigcirc\eta \mid \Box_I\eta \mid \Diamond_I\eta \mid \eta \vee \eta \mid \eta \wedge \eta$$

where $I = [n, m]$ for some $n, m \in \mathbb{N}$ such that $n \leq m$. The semantics is:

$$\begin{aligned} \sigma \models \Box_{[n,m]}\eta & \text{ iff } \sigma^j \models \eta \text{ for all } j \text{ such that } n \leq j \leq m. \\ \sigma \models \Diamond_{[n,m]}\eta & \text{ iff there exists } j \text{ such that } n \leq j \leq m \text{ such that } \sigma^j \models \eta. \end{aligned}$$

Note that \Diamond_I and \Box_I can be de-sugared using \bigcirc , but with an exponential unfolding in terms m . A trace σ models $\alpha \wedge \Box\psi$ whenever $\sigma_0(\alpha)$ holds and $\sigma^k \models \psi$ for all $k \geq 0$.

It is easy to see that any safety formula is logically equivalent to a formula in Negation Normal Form (NNF) by pushing negation to the propositional level (using equivalences $\neg\bigcirc\eta \equiv \bigcirc\neg\eta$, $\neg\Diamond_I\eta \equiv \Box_I\neg\eta$ and $\neg\Box_I\eta \equiv \Diamond_I\neg\eta$):

$$\ell ::= p \mid \neg p \mid x = c \mid \neg(x = c) \mid \mathbf{T} \mid \mathbf{F} \quad \eta ::= \ell \mid \bigcirc\eta \mid \Box_I\eta \mid \Diamond_I\eta \mid \eta \vee \eta \mid \eta \wedge \eta.$$

We assume that formulas are translated to NNF, ℓ stands for a literal, and, for $i \in \mathbb{N}$, \bigcirc^i abbreviates a sequence of operators \bigcirc of length i . The temporal depth of φ is the maximum number of nested \bigcirc operators, where \Box_I and \Diamond_I are interpreted in terms of \bigcirc . It is easy to see that the truth value of a formula (at position i) of depth d only requires to inspect d positions of the trace (after i). We define a semantics \models^{fin} of our safety fragment of LTL on finite traces $\lambda = \lambda_0 \cdots \lambda_{d-1}$ where $d \geq 1$ by:

$$\begin{aligned} \lambda \models^{fin} \ell & \text{ iff } \lambda_0(\ell) = 1 \\ \lambda \models^{fin} \eta_1 \wedge \eta_2 & \text{ iff } \lambda \models^{fin} \eta_1 \text{ and } \lambda \models^{fin} \eta_2 \\ \lambda \models^{fin} \eta_1 \vee \eta_2 & \text{ iff } \lambda \models^{fin} \eta_1 \text{ or } \lambda \models^{fin} \eta_2 \\ \lambda \models^{fin} \bigcirc\eta & \text{ iff if } d > 1 \text{ then } \lambda^{1..d} \models^{fin} \eta \text{ (remember that } \lambda^{1..d} \text{ denotes } \lambda_1 \cdots \lambda_d) \\ \lambda \models^{fin} \Box_{[n,m]}\eta & \text{ iff } \lambda^j \models^{fin} \eta \text{ for all } n \leq j \leq \min(m, d) \\ \lambda \models^{fin} \Diamond_{[n,m]}\eta & \text{ iff if } n \leq m < d \text{ then } \lambda^j \models^{fin} \eta \text{ for some } n \leq j \leq m \end{aligned}$$

Note that a witness of the violation of a safety formula η is a finite sequence $\lambda = \lambda_0 \cdots \lambda_{d-1}$ such that $\lambda \not\models^{fin} \eta$.

Given a set of formulas Δ , we denote by $\text{Val}_\Delta(\mathcal{V})$ the set of all valuations $v \in \text{Val}(\mathcal{V})$ such that $v(x)$ for every Boolean variable $x \in \Delta$, $\neg v(x)$ for every Boolean variable $\neg x \in \Delta$, $v(x) = c$ for every x of enumerated type such that $x = c \in \Delta$, and $v(x) \neq c$ for every x of enumerated type such that $\neg(x = c) \in \Delta$. Note that if x does not occur in Δ , there are many $v \in \text{Val}_\Delta(\mathcal{V})$ with different values for $v(x)$. If Δ is a set of literals then $\lambda_0 \models^{fin} \Delta$ if and only if $\lambda_0 \in \text{Val}_\Delta(\mathcal{V})$. Given $v \in \text{Val}(\mathcal{X}_e)$ and $w \in \text{Val}(\mathcal{Y})$, we denote by $v + w$ the valuation in $z \in \text{Val}(\mathcal{X}_e \cup \mathcal{Y})$ such that $z(p) = v(p)$ if $z \in \mathcal{X}_e$ and $z(p) = w(p)$ if $z \in \mathcal{Y}$. This notation is extended to pairs of finite traces λ on \mathcal{X}_e and λ' on \mathcal{Y} of the same length d , i.e., $\lambda + \lambda'$ denotes the trace $(\lambda_0 + \lambda'_0) \cdots (\lambda_{d-1} + \lambda'_{d-1})$. It is easy to see that our fragment of safety specifications can only describe safety languages.

Lemma 1. *Given a safety spec. $\varphi = \alpha \wedge \Box \psi$ and a trace σ , $\sigma \not\models \varphi$ iff either*

- (i) σ_0 is a witness of the violation of $\alpha \wedge \psi$, or
- (ii) for some i and $d \leq \text{depth}(\psi)$ $\sigma^{i..i+(d+1)}$ is a witness of the violation of ψ .

Safety Games. A safety game $\langle I, P, P_E, P_S, T, B \rangle$ is played by two players E (the environment) and S (the system), where (1) P is the set of positions, partitioned into $P = P_E \cup P_S$; (2) $I \subseteq P$ is the initial positions; (3) $T \subseteq (P \times P)$ is the set of moves; and (4) $B \subseteq P$ is the safety winning condition. E moves at positions P_E and S moves at P_S , choosing a successor. A play $\pi : v_0 v_1 v_2 \dots$ is an infinite sequence of positions, related by moves. We assume that every position has a successor so we do not have to deal with finite plays. A play π is winning for S if for all i , $\pi(i) \notin B$. A memoryless strategy ρ_S for S is a map $\rho_S : P_S \rightarrow P$, such that $(p, \rho_S(p)) \in T$ is a move for all $p \in P_S$.

Strategies for E are defined analogously. A play π is played according to a strategy ρ_S if for every i , if $\pi(i) \in P_S$ then $\pi(i+1) = \rho_S(\pi(i))$. A strategy ρ_S of S is winning if every initial play π played according to ρ_S is winning for S . It is well-known that safety games are memoryless determined (either S or E have a memoryless winning strategy). We now construct a safety game from a specification φ over \mathcal{X}_e and \mathcal{Y} :

- $P_E = \{\text{Val}(\mathcal{X}_e)^k \times \text{Val}(\mathcal{Y})^k \mid k \in \mathbb{N}\}$. We use $P_E^k = \{(\bar{x}, \bar{y}) \mid |x| = |y| = k\}$.
- $P_S = \{\text{Val}(\mathcal{X}_e)^{k+1} \times \text{Val}(\mathcal{Y})^k \mid k \in \mathbb{N}\}$. We use $P_S^{k+1} = \{(\bar{x}, \bar{y}) \mid |x| = k + 1 \text{ and } |y| = k\}$.
- T contains two types of edges $T = T_E \cup T_S$ defined as follows for each $k \in \mathbb{N}$:
 - $T_E \subseteq (P_E^k, P_S^{k+1})$ such that $((\bar{x}, \bar{y}), (\bar{x} \cdot v, \bar{y})) \in T_E$ iff $v \in \text{Val}(\mathcal{X}_e)$.
 - $T_S \subseteq (P_S^{k+1}, P_E^{k+1})$ such that $((\bar{x} \cdot v, \bar{y}), (\bar{x} \cdot v, \bar{y} \cdot w)) \in T_S$ iff $w \in \text{Val}(\mathcal{Y})$.
- $I = \{(\epsilon, \epsilon)\}$.

Note that E and S alternate playing. Given a position $p \in P_E \setminus I$ of the form $(\bar{x} \cdot v, \bar{y} \cdot w)$ we use $\text{move}(p) = (v + w)$ for the valuation of the variables of $\mathcal{X}_e \cup \mathcal{Y}$ according to v and w . Given a play π we use $\text{trace}(\pi)$ for the trace σ such that $\sigma(i) = \text{move}(\pi(2i+1))$, which corresponds to the sequence of valuations that E and S pick. This arena is essentially an infinite tree that records the valuations chosen. We define the set of bad states as the safety winning condition:

$$B_\varphi = \{(\bar{x}, \bar{y}) \mid \text{there is } v \in \text{Val}(\mathcal{X}_e), \text{ for all } w \in \text{Val}(\mathcal{Y}) : \bar{x} \cdot v + \bar{y} \cdot w \not\models^{\text{fin}} \varphi\}.$$

We use $\mathcal{G}(\varphi) : \langle P, P_E, P_S, I, T, B_\varphi \rangle$ for the safety specification game for φ .

Lemma 2. *A safety spec. φ is realizable if and only if $\mathcal{G}(\varphi)$ is winning for S .*

3 Realizability Tableaux

We introduce now the main technical contribution of this paper, a tableau method for deciding the realizability of a safety specifications, which also allows to synthesize a winning strategy for realizable specifications.

3.1 Terse Normal Form

Our tableau for φ will cover the plays of $\mathcal{G}(\varphi)$, where the environment chooses a move on its variables \mathcal{X}_e and, then, the system responds with a move on \mathcal{Y} . In order for branches to represent real plays, the formula in a node should determine the true strict-future possibilities at the current position. Consider that the formula $\varphi_2 = (\bigcirc\neg s) \vee (p_e \wedge \bigcirc\bigcirc s)$ represents the possible moves at some position in a game. Satisfying $(\bigcirc\neg s)$ would fulfill the specification. Also, if the environment moves p_e both $\bigcirc\bigcirc s$ and $\bigcirc\neg s$ would satisfy φ_2 . However, a classical tableau-style analysis would split φ_2 into two branches such that the one containing p_e requires $\bigcirc\bigcirc s$ to satisfy the specification, precluding the possibility of $\bigcirc\neg s$. Note also that the formula $\varphi_3 = (p_e \wedge (\bigcirc\neg s \vee \bigcirc\bigcirc s)) \vee (\neg p_e \wedge \bigcirc\neg s)$ is logically equivalent to φ_2 , but suitable for a tableau-style analysis of realizability. We now introduce the *Terse Normal Form* (TNF) for safety formulas that associates moves with formulas that capture the condition that any trace must satisfy in the (strict) future to be coherent with the current safety specification. The formula φ_3 above is in TNF.

Basic (sub)formulas of a safety formula are of the form ℓ , $\bigcirc^n \eta$, $\diamond_I \eta$ or $\square_I \eta$. We classify these into *from-now* formulas: ℓ , $\diamond_{[0,m]} \eta$, $\square_{[0,m]} \eta$ and *from-next* formulas: $\bigcirc \eta$, $\diamond_{[n,m]} \eta$ and $\square_{[n,m]} \eta$ (for any $m \geq n \geq 1$).

Definition 1 (Strict-future and separated). *A strict-future formula is a DNF combination of from-next formulas. A separated formula is the conjunction of a set of Boolean literals (possibly empty) and (at most) one strict-future formula. If π is a separated formula, then $\mathcal{L}(\pi)$ denotes the set of literals in π and $\mathcal{F}(\pi)$ denotes the strict-future formula in π .*

Definition 2 (TNF). *A safety formula η in Terse Normal Form (TNF) is a disjunction $\bigvee_{i=1}^n \pi_i$ such that each π_i is a separated formula, and for all $1 \leq i \neq j \leq n$ there is at least one literal ℓ such that $\ell \in \mathcal{L}(\pi_i)$ and $\neg \ell \in \mathcal{L}(\pi_j)$.*

Proposition 1. *For any safety formula η there is a logically equivalent safety formula, called $\text{TNF}(\eta)$, that is in TNF.*

Example 1. The TNF for $p_e \leftrightarrow \bigcirc s$ and $\bigcirc p_e \leftrightarrow \bigcirc s$ from Section 1 are $\text{TNF}(p_e \leftrightarrow \bigcirc s) \equiv (p_e \wedge \bigcirc s) \vee (\neg p_e \wedge \bigcirc\neg s)$ and $\text{TNF}(\bigcirc p_e \leftrightarrow \bigcirc s) \equiv (\bigcirc p_e \wedge \bigcirc s) \vee (\bigcirc\neg p_e \wedge \bigcirc\neg s)$. Finally, for $\eta = c \wedge (\neg p_e \rightarrow \square_{[0,9]}\neg c) \wedge (\square_{[0,9]}c \vee \diamond_{[0,2]}\neg c)$: $\text{TNF}(\eta) \equiv (p_e \wedge c \wedge (\bigcirc\diamond_{[0,1]}\neg c \vee \bigcirc\square_{[0,8]}c)) \vee (\neg p_e \wedge c \wedge \bigcirc\square_{[0,8]}c)$.

Definition 3 (Moves). *Given $\bigvee_{i=1}^n \pi_i$ in TNF we call each π_i a move.*

Note that $\text{Val}_{\pi_i} = \text{Val}_{\mathcal{L}(\pi_i)}$ for any move π_i of any formula in TNF. In Ex. 1, $\text{TNF}(p_e \leftrightarrow \bigcirc s)$ contains two moves, each having a literal and a strict-future formula, but $\text{TNF}(\bigcirc p_e \leftrightarrow \bigcirc s)$ has only one move (the empty set of literals) with one future-strict formula (which is a disjunction).

Proposition 2. *Let η be a safety formula and let $\text{TNF}(\eta) = \bigvee_{i=1}^n \pi_i$. Then,*
 (a) *For any trace σ , $\sigma \models \eta$ iff $\sigma \models \pi_i$ for exactly one $1 \leq i \leq n$.*

- (b) For any finite trace λ , $\lambda \models^{fin} \eta$ iff $\lambda \models^{fin} \pi_i$ for exactly one $1 \leq i \leq n$.
- (c) Let σ be such that $\sigma \models \eta$ and let $1 \leq i \leq n$. Then, $\sigma \models \mathcal{L}(\pi_i) \rightarrow \mathcal{F}(\pi_i)$.

We define now a special subset of moves in a TNF that are called \mathcal{X}_e -coverings.

Definition 4. A formula $\bigvee_{i=1}^n \pi_i$ in TNF with $\bigcup_{i=1}^n \text{Val}_{\pi_i}(\mathcal{X}_e) = \text{Val}(\mathcal{X}_e)$ is called an \mathcal{X}_e -covering. An \mathcal{X}_e -covering is minimal if $\bigvee_{i=1, i \neq j}^n \pi_i$ is not an \mathcal{X}_e -covering for any $1 \leq j \leq n$.

Intuitively, a minimal \mathcal{X}_e -covering represents a system strategy from the current position. Therefore, the collection of all minimal coverings represents all possible strategies. Moreover, each move in a strategy contains all the strict-future possibilities for this move.

Example 2. Let $\text{TNF}(\eta) = (p_e \wedge c \wedge \eta_1) \vee (\neg p_e \wedge c \wedge \eta_2) \vee (\neg c \wedge \eta_3)$ where η_1, η_2, η_3 are strict-future formulas and $\mathcal{X}_e = \{p_e\}$. It is a non-minimal \mathcal{X}_e -covering, but the third move $(\neg c \wedge \eta_3)$ is a minimal one. The two first moves together also provide a minimal \mathcal{X}_e -covering.

We say that a set of indices I is a (minimal) \mathcal{X}_e -covering when $\bigvee_{i \in I} \pi_i$ is a (minimal) \mathcal{X}_e -covering.

Proposition 3. Let Φ be a set of safety formulas and $\text{TNF}(\Phi \wedge \psi) = \bigvee_{i \in I} \pi_i$.

- (a) If I is not an \mathcal{X}_e -covering, then for some $v \in \text{Val}(\mathcal{X}_e)$, $v \not\models^{fin} \Phi \wedge \psi$.
- (b) If I is a minimal \mathcal{X}_e -covering, then for all $i \in I$ and all $v \in \text{Val}_{\pi_i}(\mathcal{X}_e)$, there exists some $v' \in \text{Val}_{\pi_i}(\mathcal{Y})$ such that $v + v' \in \text{Val}_{\pi_i}(\mathcal{X}_e \cup \mathcal{Y})$.
- (c) If for each $v \in \text{Val}(\mathcal{X}_e)$ there exists $v' \in \text{Val}(\mathcal{Y})$ such that $v + v' \models^{fin} \Phi \wedge \psi$, then there exists some minimal \mathcal{X}_e -covering $J \subseteq I$.

To handle strict-future formulas $\mathcal{F}(\pi)$ in the tableau rules we introduce the symbol $\ddot{\vee}$ which is semantically equivalent to \vee , but our tableau rules deal differently with both disjunctive operators. More precisely, strict-future subformulas $\mathcal{F}(\pi)$ (inside moves of TNF formulas) will be written as $\ddot{\vee}_{i=1}^m \delta_i$.

3.2 Tableaux

Realizability tableaux are AND-OR trees, where each node is labelled by a set of formulas⁴. A node is said to be the parent of its successors nodes. The root of the tree is labelled with the input safety specification. The tableau is constructed using the set of tableau rules shown in Fig. 3. Each rule determines the labels on the children of a node and the kind (AND or OR) of its successors. A tableau is completed when no further rule can be applied. Rules apply only to nodes in branches that are neither failed nor successful. A node is called a leaf when no rule can be applied to it. There are two kinds of leaves. Failure leaves are labelled by (syntactically) inconsistent sets of formulas, which indicates that the

⁴ We graphically represent AND-nodes with an arc embracing all the edges to the AND-successors of a node.

branch from the root to the leaf is failed. Successful leaves are labelled by sets of formulas that are subsumed (in the sense we will make precise in Def. 6) by those previous node in the branch from the root to the leaf.

Before we introduce the tableau rules, we define the finite set of all formulas that could appear in the construction of a tableau for φ , denoted as $\text{Clo}(\varphi)$.

Definition 5. *Given a formula β , we denote by $\text{SubFm}(\beta)$ the set of all subformulas of β . In particular, $\text{SubFm}(\bigcirc^i \beta) = \{\bigcirc^j \beta \mid 0 \leq j \leq i\} \cup \text{SubFm}(\beta)$. For a given safety formula ψ , we define $\text{Varnt}(\psi)$ to be the union of the following four sets that collect all the variants of subformulas \diamond_I and \square_I that the tableau rules could introduce.*

$$\begin{aligned} & \{\diamond_{[n,m']}\beta, \bigcirc \diamond_{[n,m']}\beta \mid \diamond_{[n,m]}\beta \in \text{SubFm}(\psi), n \leq m' < m\} \cup \\ & \{\square_{[n,m']}\beta, \bigcirc \square_{[n,m']}\beta \mid \square_{[n,m]}\beta \in \text{SubFm}(\psi), n \leq m' < m\} \cup \\ & \{\text{SubFm}(\bigcirc^i \beta) \mid \diamond_{[n,m]}\beta \in \text{SubFm}(\psi), 0 \leq i \leq n\} \cup \\ & \{\text{SubFm}(\bigcirc^i \beta) \mid \square_{[n,m]}\beta \in \text{SubFm}(\psi), 0 \leq i \leq n\} \end{aligned}$$

The set $\text{Ordnf}(\psi)$ consists of all formulas of the form $\check{\vee}_{i=1}^n \bigwedge_{j=1}^m \beta_{i,j}$ where each $\beta_{i,j}$ is in $\text{Varnt}(\psi)$. Then, the closure of a safety specification $\varphi = \alpha \wedge \square \psi$ is the finite set $\text{Clo}(\varphi) = \text{Preclo}(\varphi) \cup \{\square \psi, \bigcirc \square \psi\}$ where $\text{Preclo}(\varphi) = \text{SubFm}(\alpha \wedge \psi) \cup \text{Varnt}(\psi) \cup \text{Ordnf}(\psi)$.

Realizability Tableaux. A tableau for a safety specification $\varphi = \alpha \wedge \square \psi$ is a labelled tree $\text{Tab}(\varphi) = (N, \tau, R)$, where N is a set of nodes, τ is a map from N to $\text{Clo}(\varphi)$ and $R \subseteq N \times N$, such that the following conditions hold:

- The root is labelled by $\{\alpha, \square \psi\}$.
- For any $(n, n') \in R$, $\tau(n')$ is the set of formulas obtained as the result of the application of one of the tableau rules (in Fig. 3) to $\tau(n)$. If the applied rule is ρ , we say that n' is a ρ -successor of n .
- For every success or failure leaf n there is no $n' \in N$ s.t. $(n, n') \in R$ where:
 - A failure leaf is a node $n \in N$ s.t. $\text{Incst}(\tau(n))$ (see Def. 7).
 - A success leaf is a node $n \in N$ such that $\square \psi \in \tau(n)$ and there exists $k \geq 0$, $n_0, \dots, n_k \in N$ such that $(n_i, n_{i+1}) \in R$ for all $0 \leq i < k$, $(n_k, n) \in R$ and $\tau(n_0) \prec \tau(n)$ (see Def. 8).

3.3 Subsumption and Syntactical Inconsistency

Subsumption rules allow to control the potential set of labellings of the tableau nodes. We use $\beta \sqsubseteq \gamma$ to denote that β subsumes γ or that γ is subsumed by β . Subsumption is related to logical implication, if $\beta \sqsubseteq \gamma$, then $\text{Mod}(\beta) \subseteq \text{Mod}(\gamma)$. Classical subsumption rules include $\beta \sqsubseteq \beta$, $\beta \wedge \gamma \sqsubseteq \beta$, and $\beta \sqsubseteq \beta \vee \gamma$. The set of formulas used to label our tableau nodes are subsumption-free with respect to classical subsumption on Boolean formulas and the following subsumption rules for temporal operators.

Definition 6. *The subsumption rules for temporal formulas are:*

- For all $n \leq n'$ and $m' \leq m$,
$$\diamond_{[n',m']}\beta \sqsubseteq \diamond_{[n,m]}\beta, \square_{[n,m]}\beta \sqsubseteq \square_{[n',m']}\beta, \text{ and } \square_{[n',m']}\beta \sqsubseteq \diamond_{[n,m]}\beta.$$

– For all $n \leq k \leq m$: $\bigcirc^k \beta \sqsubseteq \diamond_{[n,m]} \beta$ and $\square_{[n,m]} \beta \sqsubseteq \bigcirc^k \beta$.

The following result easily follows from Def. 6 and semantics.

Proposition 4. *Let $\beta \sqsubseteq \gamma$ be a pair of formulas. For any trace σ , if $\sigma \models \beta$ then $\sigma \models \gamma$. For any finite trace λ , if $\lambda \models^{fin} \beta$ then $\lambda \models^{fin} \gamma$. Consequently, $\sigma \not\models \beta \wedge \tilde{\gamma}$ and $\lambda \not\models^{fin} \beta \wedge \tilde{\gamma}$ for any σ and λ , where $\tilde{\gamma}$ is the NNF of $\neg\gamma$.*

Definition 7. *A set of formulas Φ is (syntactically) inconsistent (denoted by $\text{Incnst}(\Phi)$) whenever one of the following four conditions hold:*

- (a) $\mathbf{F} \in \Phi$
- (b) $\{\beta, \tilde{\gamma}\} \subseteq \Phi$ for some β, γ such that $\beta \sqsubseteq \gamma$
- (c) $\{x = c_1, x = c_2\} \subseteq \Phi$ for some $c_1 \neq c_2$
- (d) $\{\neg(x = c) \mid c \in T\} \subseteq \Phi$ for some enumerated type T .

Otherwise, Φ is (syntactically) consistent, denoted $\text{Cnst}(\Phi)$.

A node that is labelled by an inconsistent set is a failure leaf and no rule is applied to it. We now define a subsumption-based order relation on sets of formulas to detect successful leaves.

Definition 8. *For two given set of formulas Φ and Φ' , we say that $\Phi < \Phi'$ if and only if for every formula $\beta \in \Phi$ there exists some $\beta' \in \Phi'$ such that $\beta \sqsubseteq \beta'$. For two given strict-future formulas, $\check{\vee}_{i=1}^n \Delta_i \sqsubseteq \check{\vee}_{j=1}^m \Gamma_j$ if and only if for all $1 \leq i \leq n$ there exists $1 \leq j \leq m$ such that $\Delta_i < \Gamma_j$.*

The following result follows from Def. 8 and Prop. 4.

Proposition 5. *For any finite trace λ and any pair of set of formulas Φ and Φ' such that $\Phi < \Phi'$, if $\lambda \models^{fin} \Phi$ then $\lambda \models^{fin} \Phi'$.*

No rule is applied to a node that is labelled by a set Φ' such that $\Phi < \Phi'$ for some previous label Φ in the same branch, because it is a successful leaf.

3.4 Tableau Rules

First, the *Always Rules* in Fig.3(a) provides a non-deterministic procedure for analyzing the minimal \mathcal{X}_e -coverings in $\text{TNF}(\Phi \wedge \psi)$ (see Def. 4 and Prop. 3). Rule $(\square \wedge)$ is the only rule in our system that produces AND-successors, by splitting the cases of each minimal \mathcal{X}_e -covering. We introduce the rules that decompose formulas into their constituents, using saturation as usual in tableau methods. The decomposing of formulas inside the conjunctions connected by $\check{\vee}$ is just an unfolding in the formula. The *Saturation Rules* in Fig. 3(b) saturate with respect to \wedge and \vee (including $\check{\vee}$) and temporal operators \diamond_I and \square_I . The following property of saturation rules is proved by routinely applying semantics.

Proposition 6. *For any saturation rule $\frac{\Phi}{\Phi_1 | \dots | \Phi_k}$, it holds that $\sigma \models \Phi$ if and only if $\sigma \models \Phi_i$ for some $1 \leq i \leq k$.*

$(\Box_F) \frac{\Phi, \Box\psi}{F, \Box\psi}$	if $\text{TNF}(\Phi \wedge \psi)$ is not an \mathcal{X}_e -covering
$(\Box\vee) \frac{\Phi, \Box\psi}{\bigvee_{i \in J_1} \pi_i, \Box\psi \mid \cdots \mid \bigvee_{i \in J_m} \pi_i, \Box\psi}$	if J_1, \dots, J_m is the collection of all minimal \mathcal{X}_e -covering of $\text{TNF}(\Phi \wedge \psi)$
$(\Box\wedge) \frac{\bigvee_{i \in I} \pi_i, \Box\psi}{\pi_1, \Box\psi \ \& \ \dots \ \& \ \pi_n, \Box\psi}$	if I is a minimal \mathcal{X}_e -covering

(a) Always Rules (where τ denotes $\text{TNF}(\Phi \wedge \psi)$)

$(\vee) \frac{\Phi, \beta \vee \gamma}{\Phi, \beta \mid \Phi, \gamma}$	$(\wedge) \frac{\Phi, \beta \wedge \gamma}{\Phi, \beta, \gamma}$	$(\ddot{\vee} \wedge) \frac{\Phi, (\eta \wedge (\beta \vee \gamma)) \ddot{\vee} \delta}{\Phi, (\eta \wedge \beta) \ddot{\vee} (\eta \wedge \gamma) \ddot{\vee} \delta}$
$(\diamond <) \frac{\Phi, \diamond_{[n,m]}\beta}{\Phi, \circ^n \beta \mid \Phi, \circ \diamond_{[n,m-1]}\beta}$ if $n < m$		
$(\ddot{\vee} \diamond <) \frac{\Phi, (\eta \wedge \diamond_{[n,m]}\beta) \ddot{\vee} \delta}{\Phi, (\eta \wedge \circ^n \beta) \ddot{\vee} (\eta \wedge \circ \diamond_{[n,m-1]}\beta) \ddot{\vee} \delta}$ if $n < m$		
$(\diamond =) \frac{\Phi, \diamond_{[n,n]}\beta}{\Phi, \circ^n \beta}$	$(\square =) \frac{\Phi, \square_{[n,n]}\beta}{\Phi, \circ^n \beta}$	$(\ddot{\vee} \diamond =) \frac{\Phi, (\eta \wedge \diamond_{[n,n]}\beta) \ddot{\vee} \delta}{\Phi, (\eta \wedge \circ^n \beta) \ddot{\vee} \delta}$
$(\square <) \frac{\Phi, \square_{[n,m]}\beta}{\Phi, \circ^n \beta, \circ \square_{[n,m-1]}\beta}$ if $n < m$	$(\ddot{\vee} \square =) \frac{\Phi, (\eta \wedge \square_{[n,n]}\beta) \ddot{\vee} \delta}{\Phi, (\eta \wedge \circ^n \beta) \ddot{\vee} \delta}$	
$(\ddot{\vee} \square <) \frac{\Phi, (\eta \wedge \square_{[n,m]}\beta) \ddot{\vee} \delta}{\Phi, (\eta \wedge \circ^n \beta \wedge \circ \square_{[n,m-1]}\beta) \ddot{\vee} \delta}$ if $n < m$		

(b) Saturation Rules

$(\circ) \frac{\Phi, \eta, \circ \Box\psi}{\eta^\downarrow, \Box\psi}$ if $\Phi \cup \{\eta\}$ is elementary and η is strict-future
--

(c) Next-state Rule

Fig. 3. Realizability Tableau Rules

Definition 9. A next-formula is a formula whose first symbol is \circ . A strict-future formula $\ddot{\vee}_{i=1}^n \Delta_i$ is elementary if every formula in the set $\bigcup_{i=1}^n \Delta_i$ is a next-formula.

The successive application of the rules $(\ddot{\vee} \wedge)$, $(\ddot{\vee} \diamond <)$, $(\ddot{\vee} \diamond =)$, $(\ddot{\vee} \square <)$ and $(\ddot{\vee} \square =)$ ensures the following proposition.

Proposition 7. Given a strict-future formula δ , there is an elementary formula δ^E such that $\delta \equiv \delta^E$ and δ^E is in DNF.

Definition 10. A set Δ is saturated whenever for all $\delta \in \Delta$ the following hold:

- If $\delta = \beta \wedge \gamma$, then $\{\beta, \gamma\} \in \Delta$. If $\delta = \beta \vee \gamma$, then $\beta \in \Delta$ or $\gamma \in \Delta$.
- If $\delta = \circ_{[n,m]}\beta$ and $n < m$, then $\{\circ^n \beta, \circ \square_{[n,m-1]}\beta\} \subseteq \Delta$.

- If $\delta = \diamond_{[n,m]}\beta$ and $n < m$, then either $\circ^n\beta \in \Delta$ or $\circ \diamond_{[n,m-1]}\beta \in \Delta$
- If $\delta = \square_{[n,n]}\beta$ or $\gamma = \diamond_{[n,n]}\beta$, then $\circ^n\beta \in \Delta$.
- If δ is a strict-future formula, then $\delta^E \in \Delta$

We use $\text{Stt}(\Delta)$ to denote the set of all (minimal) saturated sets that contains Δ .

Proposition 8. Let Δ be a set of formulas, σ a trace and λ a finite trace.

- $\sigma \models \Delta$ if and only if $\sigma \models \Phi$ for some $\Phi \in \text{Stt}(\Delta)$.
- $\lambda \models^{\text{fin}} \Delta$ if and only if $\lambda \models^{\text{fin}} \Phi$ for some $\Phi \in \text{Stt}(\Delta)$.

By Prop. 2 and 8, we obtain the next result.

Proposition 9. Let Φ be a set of safety formulas and let J_1, \dots, J_m be the collection of all minimal \mathcal{X}_e -coverings in $\text{TNF}(\Phi \wedge \psi) = \bigvee_{i \in I} \pi_i$. Then

- (a) For any trace σ , $\sigma \models \Phi, \square\psi$ iff $\sigma \models \pi_i, \circ\square\psi$ holds for some $i \in J_k$ for each $1 \leq k \leq m$. Let λ be finite trace, $\lambda \models^{\text{fin}} \Phi \wedge \psi$ iff $\lambda \models^{\text{fin}} \pi_i$ for some $i \in J_k$ for each $1 \leq k \leq m$.
- (b) For any $1 \leq k \leq m$ and any $i \in J_k$ the following two facts hold:
 - (i) If $\text{Incnst}(\Delta)$ for all $\Delta \in \text{Stt}(\Phi \cup \{\pi_i\})$, then every $\lambda_0 \in \text{Val}_\Delta(\mathcal{X}_e \cup \mathcal{Y})$ is a witness of the violation of $\Phi \wedge \square\psi$.
 - (ii) Let $\Delta \in \text{Stt}(\Phi \cup \{\pi_i\})$ be s.t. $\text{Cnst}(\Delta)$. Then, $\lambda_0 \models^{\text{fin}} \Phi \wedge \psi$ for every $\lambda_0 \in \text{Val}_\Delta(\mathcal{X}_e \cup \mathcal{Y})$.

Prop. 10 follows from the fact that, by Def. 4, there is some $v \in \text{Val}(\mathcal{X}_e) \setminus \text{Val}_{\Phi \wedge \psi}(\mathcal{X}_e)$.

Proposition 10. Let Φ be a set of formulas. If $\text{TNF}(\Phi \wedge \psi)$ is not an \mathcal{X}_e -covering, then there is a $v \in \text{Val}(\mathcal{X}_e)$ s.t. for all $v' \in \text{Val}(\mathcal{Y})$, $v + v' \not\models^{\text{fin}} \Phi \wedge \psi$.

Proposition 11. Let Φ be a set of formulas, $\text{TNF}(\Phi \wedge \psi) = \bigvee_{i \in I} \pi_i$ be an \mathcal{X}_e -covering and J_1, \dots, J_m be the collection of all minimal \mathcal{X}_e -coverings in I . If for every $1 \leq k \leq m$ there exists some $i \in J_k$ such that $\text{Incnst}(\Delta)$ for all $\Delta \in \text{Stt}(\pi_i)$, then there exists some $v \in \text{Val}(\mathcal{X}_e)$ such that for all $v' \in \text{Val}(\mathcal{Y})$, $v + v' \not\models^{\text{fin}} \Phi \wedge \psi$.

Finally, the tableau rules also include the *Next-state rule* in Fig. 3(c). This rule is used to generate a new tableau node, that is to jump from a temporal position to the next. Its formalization is based on the following definitions.

Definition 11. Given an elementary strict-future formula $\eta = \ddot{\bigvee}_{i=1}^n \bigwedge_{j=1}^m \circ\beta_{i,j}$, the formula η^\downarrow is $\ddot{\bigvee}_{i=1}^n \bigwedge_{j=1}^m \beta_{i,j}$.

Example 3. Consider the strict-future formula $\delta = \diamond_{[1,2]}a \ddot{\bigvee} \square_{[1,3]}b$. Then, $\delta^E = \circ a \ddot{\bigvee} \circ \diamond_{[1,1]}a \ddot{\bigvee} (\circ b \wedge \circ \square_{[1,2]}b)$ and $\delta^{E\downarrow} = a \ddot{\bigvee} \diamond_{[1,1]}a \ddot{\bigvee} (b \wedge \square_{[1,2]}b)$. Note that the only effect of \downarrow is to remove the \circ -operators in each term of δ^E .

Definition 12. A set of formulas Φ is elementary if it consists of a set of literals and one elementary strict-future formula.

Basically, the application of the *Next-state rule* to an elementary set that labels a node, removes all literals and removes the \circ -operators (in each term) from the single elementary strict-future formula.

Proposition 12. *Let $\Phi \cup \{\eta\}$ be a consistent and elementary set of formulas with strict-future formula η . Then, (a) For any trace σ , if $\sigma \models \Phi, \eta, \bigcirc \square \psi$ then $\sigma^1 \models \eta^\downarrow, \square \psi$; (b) let $\lambda = \lambda_0, \dots, \lambda_{k-1}$ be a pre-witness of $\alpha \wedge \square \psi$ s.t. $\lambda_{k-1} \models^{fin} \Phi$, and let $\lambda_k \in \text{Val}(\mathcal{X}_e \cup \mathcal{Y})$. Then, $\lambda \cdot \lambda_k$ is a pre-witness of $\alpha \wedge \square \psi$ iff $\lambda_k \models^{fin} \eta^\downarrow \wedge \psi$.*

3.5 A Tableau Algorithm for Realizability

Alg. 1 provides a decision procedure for realizability. The algorithm constructs completed tableau by expanding the minimal \mathcal{X}_e -coverings produced by the moves (and allowed by the input safety specification) at successive positions. Alg. 1 uses recursion to explore in-depth the branches of the tree. The formal parameter is given as the union of a set of formulas Φ and a formula χ that ranges in $\{\square \psi, \bigcirc \square \psi\}$. For deciding realizability of a safety specification $\varphi = \alpha \wedge \square \psi$, the initial call $\text{Tab}(\varphi)$ is really $\text{Tab}(\{\alpha, \square \psi\})$. Intuitively, player E moves when $\chi = \square \psi$ (including at the start), whereas S moves when $\chi = \bigcirc \square \psi$.

Algorithm 1: $\text{Tab}(\Phi \cup \{\chi\})$ returns *is_open*: Boolean

```

1 if  $\Phi$  is inconsistent then is_open := False;
2 else if  $\chi = \square \psi$  then
3   if  $\Phi_0 \leq \Phi$  for some  $\Phi_0$  in the branch of  $\Phi$  then
4     is_open := True
5   else if  $\text{TNF}(\Phi \wedge \psi)$  is not an  $\mathcal{X}_e$ -covering then
6     is_open :=  $\text{Tab}(\{\mathbf{F}, \square \psi\})$ ;
7   else if  $\text{TNF}(\Phi \wedge \psi)$  is a non-minimal  $\mathcal{X}_e$ -covering then
8     Let  $J_1, \dots, J_m$  be all the minimal  $\mathcal{X}_e$ -coverings of  $\text{TNF}(\Phi \wedge \psi)$ ;
9      $i, \text{is\_open} := 0, \text{False}$ ;
10    while  $\neg \text{is\_open} \wedge i < m$  do
11      i, is_open :=  $i + 1, \text{Tab}(J_i \cup \{\square \psi\})$ ;
12  else //  $\text{TNF}(\Phi \wedge \psi) = \bigvee_{i=1}^n \pi_i$  is a minimal  $\mathcal{X}_e$ -covering
13     $i, \text{is\_open} := 0, \text{True}$ ;
14    while is_open  $\wedge i < n$  do
15      i, is_open :=  $i + 1, \text{Tab}(\{\pi_i, \bigcirc \square \psi\})$ ;
16 else if  $\Phi = \Lambda \cup \{\eta\}$  is elementary ( $\eta$  is strict-future) then
17   is_open :=  $\text{Tab}(\{\eta^\downarrow, \square \psi\})$ ;
18 else
19    $\rho := \text{select\_saturation\_rule}(\Phi)$ ;
20   Let  $1 \leq k \leq 2$  and  $\Phi_1, \dots, \Phi_k$  the set of all  $\rho$ -children;
21   is_open :=  $\text{Tab}(\Phi_1 \cup \{\bigcirc \square \psi\})$ ;
22   if  $k = 2 \wedge \neg \text{is\_open}$  then
23     is_open :=  $\text{Tab}(\Phi_2 \cup \{\bigcirc \square \psi\})$ 

```

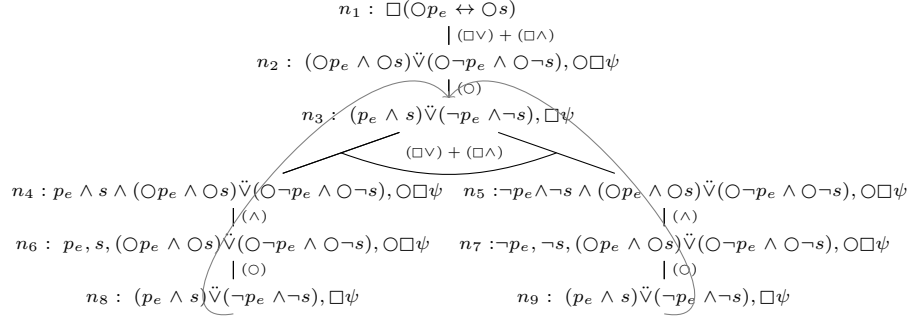


Fig. 4. Open tableau for $\Box(\bigcirc p_e \leftrightarrow \bigcirc s)$.

Definition 13. A branch b of a tableau is a sequence of nodes n_0, \dots, n_k such that n_0 is the root and $(n_i, n_{i+1}) \in R$ for $0 \leq i < k - 1$. If n_k is a successful leaf, then b is called a successful branch. If n_k is a failure leaf, then b is called a failure branch.

Alg. 1 returns the Boolean variable is_open , which corresponds to whether the completed tableau for the call parameter $\Phi \cup \{\chi\}$ is open or closed. Lines 1–4 deal with the simple cases of the recursion. Line 6 produces a recursive call that immediately returns failure. A tableau is called *completed* when all its branches contain a terminal node, i.e., all its branches are failure or successful. Recursive calls in Alg. 1 and the notions of open and closed tableaux are related to AND-nodes, for which we introduce the following definition.

Definition 14. A set of branches H of a completed tableau is called a bunch whenever for every $b \in H$, every AND-node $n \in b$, and every n' that is an $(\Box\wedge)$ -successor of n , there is $b' \in H$ such that $n' \in b'$. A completed tableau is open when it contains at least one bunch with all its branches successful. Otherwise, the tableau is closed.

Alg. 1 looks for bunches of successful branches as follows. Lines 7-11 of Alg. 1 invoke a recursive call for each minimal \mathcal{X}_e -covering, according to rule $(\Box\vee)$. When some of these calls return is_open for a minimal \mathcal{X}_e -covering J_i , which is an OR-node, the iteration is finished with this result for the previous call. The construction of the tableau for each J_k , by the rule $(\Box\wedge)$ and according to lines 12-15, produces a call for each move π_i in J_k . Moves are AND-children, hence all the calls should give is_open to obtain truth for J_k . Finally, lines 16-17 perform the application of (\bigcirc) , and lines 18-23 apply the saturation rules. When one rule is applied, the second child is expanded only if the first child returns not is_open .

Proposition 13. Alg. 1 terminates and $\text{Tab}(\varphi)$ builds a completed tableau.

Example 4. We revisit the specification $\Box\psi_3$ with $\psi_3 : (\bigcirc p_e \leftrightarrow \bigcirc s)$ discussed in Section 1, for which $\text{TNF}(\bigcirc p_e \leftrightarrow \bigcirc s) = (\bigcirc p_e \wedge \bigcirc s) \vee (\bigcirc \neg p_e \wedge \bigcirc \neg s)$ is the only minimal \mathcal{X}_e -covering. Fig. 4 shows an open tableau for this formula.

The only child of the root, n_2 , is obtained by rule $(\Box\vee)$ and then $(\Box\wedge)$. When the (\bigcirc) applies to n_2 , the label of node n_3 is obtained, which is $\{(p_e \wedge s) \check{\vee} (\neg p_e \wedge \neg s), \Box\psi\}$. Then, $\text{TNF}((p_e \wedge s) \vee (\neg p_e \wedge \neg s)) \wedge \psi$ yields a minimal \mathcal{X}_e -covering with two moves: $(p_e \wedge s \wedge (\bigcirc p_e \wedge \bigcirc s) \check{\vee} (\bigcirc \neg p_e \wedge \bigcirc \neg s))$ and $(\neg p_e \wedge \neg s \wedge (\bigcirc p_e \wedge \bigcirc s) \check{\vee} (\bigcirc \neg p_e \wedge \bigcirc \neg s))$. Hence, the rule $(\Box\vee)$ is applied, and after it, the rule $(\Box\wedge)$ produces one AND-node with two children, one for each move. In both branches, after saturation and application of (\bigcirc) , a node already in the branch is obtained. Therefore, the completed tableau has an open bunch and the specification is realizable. More examples can be found in [19].

Correctness. For any given specification φ , it holds that φ is realizable if and only if the completed tableau $\text{Tab}(\varphi)$ is open. We formally prove this statement by defining a new class of games (a variation of safety games) called a safety tableau-game $\mathcal{T}(\varphi)$ where players E and S play with game rules that correspond to the tableau rules. Then we connect winning strategies for S in $\text{Tab}(\varphi)$ with winning strategies for S in $\mathcal{T}(\varphi)$. The full proof is in [19].

4 Conclusions

We have introduced the first tableau method to decide realizability of temporal safety formulas. Our tableau method allows to synthesize a system when the specification is realizable because a (memoryless) winning strategy for the system can be extracted from an open tableau (the technical details of synthesizing the system is out of the scope of this paper and how to efficiently extract and encode this strategy is ongoing work).

Our tableau method is based on the novel notion of terse normal form (TNF) of formulas that is crucial in the formulation of the realizability tableau. The tableau rules make use of the terse normal form to precisely capture the information that each player (environment and system) has to reveal at each step. We have proved soundness and completeness of the proposed method.

Future work includes the implementation of the method presented in this paper and to experiment with the resulting prototype in a collection of benchmarks. We would ultimately like to compare an efficient implementation of the realizability tableau with mature tools from the SYNTCOMP competition.

We also plan to extend the method to more expressive languages, including the handling of richer propositions (like numeric variables and expressions) by combining realizability tableau rules with tableau reasoning capabilities for these domains. We have illustrated this path in this paper by the introduction of enumerated types. Another interesting extension is a deeper analysis, including new rules, to handle upper and lower bounds of intervals in temporal operators, for example to accelerate a branch to reach the lower bound a of an $\Box_{[a,b]}$ operator. We would like to ultimately extend our tableau method to richer fragments of LTL.

Finally, future work includes a precise analysis of the complexity of the realizability tableau and its different instances.

References

1. <https://syntcomp.org>.
2. Beth. *The Foundation of Mathematics*. North-Holland, 1959.
3. Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive(1) designs. *J. Comput. Syst. Sci.*, 78(3):911–938, 2012.
4. Roderick Bloem, Bettina Könighofer, and Martina Seidl. SAT-based synthesis methods for safety specs. In *Proc. of VMCAI'14*, volume 8318 of *LNCS*, pages 1–20, 2014.
5. Aaron Bohy, Véronique Bruyère, Emmanuel Filiot, Naiyong Jin, and cois Raskin Jean-Fran' Acacia+, a tool for LTL synthesis. In *Proc. of CAV'12*, volume 7358 of *LNCS*, pages 652–657. Springer, 2012.
6. Romain Brenguier, Guillermo A. Pérez, Jean-François Raskin, and Ocan Sankur. AbsSynthe: abstract synthesis from succinct safety specifications. In *Proc. of the 3rd Workshop in Synthesis (SYNT'14)*, volume 157 of *EPTCS*, pages 100–116, 2014.
7. J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138, 1969.
8. Martin De Wulf, Laurent Doyen, Nicolas Maquet, and Jean-François Raskin. Alaska: Antichains for logic, automata and symbolic kripke structures analysis. In *Proc. of the 6th Int'l Symp. on Automated Technology for Verification and Analysis (ATVA'08)*, volume 5311 of *LNCS*, pages 240–245. Springer, 2008.
9. Nicolás D'Ippolito, Victor A. Braberman, Nir Piterman, and Sebastian Uchitel. Synthesizing nonanomalous event-based controllers for liveness goals. *ACM Trans. Softw. Eng. Methodol.*, 22(1), 2013.
10. Rüdiger Ehlers. Unbeast: Symbolic bounded synthesis. In *Proc. of TACAS'11*, volume 6605 of *LNCS*, pages 272–275. Springer, 2011.
11. Bernd Finkbeiner. Bounded synthesis for Petri games. In *Proc. of the Symp. in Honor of Ernst-Rüdiger Olderog on the Occasion of His 60th Birthday*, volume 9360, pages 223–237. Springer, 2015.
12. Bernd Finkbeiner and Swen Jacobs. Lazy synthesis. In *Proc. of VMCAI'12*, volume 7148 of *LNCS*, pages 219–234. Springer, 2012.
13. Bernd Finkbeiner and F. Klein. Bounded cycle synthesis. In *Proc. of CAV'16*, volume 9779 of *LNCS*, pages 118–135. Springer, 2016.
14. Bernd Finkbeiner and Sven Schewe. SMT-based synthesis of distributed systems. In *Proc. of the 2nd Workshop on Automated Formal Methods (AFM'07)*, pages 69–76. ACM, 2007.
15. Bernd Finkbeiner and Sven Schewe. Bounded synthesis. *Int. J. Softw. Tools Technol. Transf.*, 15(5-6):519–539, 2013.
16. Bernd Finkbeiner and Leander Tentrup. Detecting unrealizable specifications of distributed systems. In *Proc. of TACAS'14*, volume 8413 of *LNCS*, pages 78–92. Springer, 2014.
17. Jose Gaintzarain, Montserrat Hermo, Paqui Lucio, Marisa Navarro, and Fernando Orejas. Dual systems of tableaux and sequents for PLTL. *Journal of Logic and Algebraic Programming*, 78(8):701–722, 2009.
18. Rajeev Goré and Florian Widmann. An optimal on-the-fly tableau-based decision procedure for PDL-satisfiability. In *Proc. of the 22nd Int. Conf. on Automated Deduction (CADE'09)*, volume 5663 of *LNCS*, pages 437–452. Springer, 2009.

19. Montserrat Hermo, Paqui Lucio, and César Sánchez. A tableau method for the realizability and synthesis of reactive safety specifications. arXiv, 2022.
20. Swen Jacobs, Nicolas Basset, Roderick Bloem, Romain Brenguier, Maximilien Colange, Peter Faymonville, Bernd Finkbeiner, Ayrat Khalimov, Felix Klein, Thibaud Michaud, Guillermo A. Pérez, Jean-François Raskin, Ocan Sankur, and Leander Tentrup. 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*, pages 116–143, 2017.
21. Barbara Jobstmann, Stefan Galler, Martin Weiglhofer, and Roderick Bloem. Anzu: A tool for property synthesis. In *Proc. of CAV'07*, volume 4590, pages 258–262. Springer, 2007.
22. Ayrat Khalimov, Swen Jacobs, and Roderick Bloem. Towards efficient parameterized synthesis. In *Proc. of VMCAI'13*, volume 7737 of *LNCS*, pages 108–123. Springer, 2013.
23. Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics*, 25:1370–1381, 2009.
24. Michael Luttenberger, Philipp J. Meyer, and Salomon Sickert. Practical synthesis of reactive systems from LTL specifications via parity games. *Acta Informatica*, 57(1-2):3–36, 2020.
25. Philipp J. Meyer, Salomon Sickert, and Michael Luttenberger. Strix: Explicit reactive synthesis strikes back! In *Proc. of 30th Int'l Conf. on Computer Aided Verification (CAV'18)*, volume 10981 of *LNCS*, pages 578–586. Springer, 2018.
26. Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive(1) designs. In *Proc. of VMCAI'06*, volume 3855 of *LNCS*, pages 364–380. Springer, 2006.
27. Amir Pnueli. The temporal logic of programs. In *Proc. of the 18th IEEE Symp. on Foundations of Computer Science (FOCS'77)*, pages 46–67. IEEE CS Press, 1977.
28. Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *Proc. of POPL'89*, pages 179–190. ACM, 1989.
29. Amir Pnueli and Roni Rosner. On the synthesis of an asynchronous reactive module. In *Proc. of ICALP'89*, volume 372 of *LNCS*, pages 652–671. Springer, 1989.
30. Sven Schewe and Bernd Finkbeiner. Bounded synthesis. In *Proc. of ATVA'07*, volume 4762 of *LNCS*. Springer, 2007.
31. Stefan Schwendimann. A new one-pass tableau calculus for PLTL. In *International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 277–291. Springer, 1998.
32. Masaya Shimakawa, Shigeki Hagihara, and Naoki Yonezaki. Reducing bounded realizability analysis to reachability checking. In *Proc. of RP'15*, volume 9328 of *LNCS*, pages 140–152. Springer, 2015.
33. Raymond M. Smullyan. *First-Order Logic*. Springer-Verlag, 1968.
34. Moshe Y. Vardi and Pierre Wolper. Reasoning about infinite computations. *Information and computation*, 115(1):1–37, 1994.
35. Pierre Wolper. The tableau method for temporal logic: An overview. *Logique et Analyse*, 28:119–136, 1985.