This version has been accepted for publication, after peer review (when applicable) and is subject to Springer Nature's AM terms of use, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: https://doi.org/10.1007/978-3-031-27481-7_28 [In: Chechik, M., Katoen, JP., Leucker, M. (eds) Formal Methods. FM 2023. Lecture Notes in Computer Science, vol 14000. Springer, Cham]

A Tableau Method for the Realizability and Synthesis of Reactive Safety Specifications

Montserrat Hermo[1], Paqui Lucio[1], and César Sánchez[2]

[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). Our method can easily be 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) [24] 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 [26,25], first studied by Pnueli and Rosner in 1989, is the problem of automatically producing a system S from a given LTL specification φ which guarantees 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.

In the last two decades the reactive synthesis problem has been studied extensively (see e.g. [4,9,14,15,20]). The approaches can be classified into three categories: (1) approaches based on games [6], (2) approaches that cover a strict fragment of LTL, like GR(1) specifications [23,2]; (3) bounded synthesis [27] explore the problem 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 realizability and synthesis for the class of safety specifications.

Tableaux methods were originally created [30] as very intuitive deduction procedures for classical propositional and first-order logic. A tableau is a tree that performs a symbolic handling of formulas according to very simple and intuitive rules based on semantics, model-theory and proof-theory. Indeed, classical tableaux corresponds to deductive proofs in Gentzen's sequent calculus. Tableaux has been evolving for years to decide the satisfiability problem of many other non-classical logics (modal, multi-valued, temporal, etc.), in some cases combining with other formal structures, such as e.g. different kinds of automata.

Traditional tableau techniques for satisfiability do not directly work for realizability. As far as we know, tableau techniques has not been yet applied for solving the realizability problem of temporal formulas, beyond its auxiliary use in automata-based methods [5]. We present in this paper a tableau based methods for the realizability of reactive safety specifications. To illustrate the problem, consider the following three safety formulas where e is an environment variable e and s is a system variable:

$$\psi_1 = s \leftrightarrow e$$
$$\psi_2 = s \leftrightarrow \bigcirc e$$
$$\psi_3 = \bigcirc s \leftrightarrow \bigcirc e$$

The safety specifications $\square\psi_1$ and $\square\psi_3$ are realizable. To see this, consider the strategy where the system simply mimics in s the value observed in e. On the other hand $\square\psi_2$ is not realizable, as the system is required to guess the next value of e which requires clairvoyance. A temporal tableau for $\square\psi_1$, first decomposes the formula into $s \leftrightarrow e, \bigcirc\square(s \leftrightarrow e)$ and, second, splits two branches for the two cases $s, e, \bigcirc\square(s \leftrightarrow e)$ and $\neg s, \neg e, \bigcirc\square(s \leftrightarrow e)$. In both nodes there is not more to do than 'jumping' to the next state. Hence in both branches we get a loop to the root $\square(s \leftrightarrow e)$. Therefore, each branch represents a model of the initial formula. Tableau structure with two leaves pointing to the root can be interpreted as the winning strategy of the system that witnesses the realizability of the safety specification. However, following the same steps $\square\psi_2$ fails to capture the unrealizability of this specification. Indeed, the formulas that successively appear in just one branch, where new-line represents 'jump' to next state, are

$$\square(s \leftrightarrow \bigcirc e), s \leftrightarrow \bigcirc e, \bigcirc\square(s \leftrightarrow \bigcirc e), s, \bigcirc e$$
$$e, \square(s \leftrightarrow \bigcirc e), s \leftrightarrow \bigcirc e, \bigcirc\square(s \leftrightarrow \bigcirc e), s, \bigcirc e$$
$$e, \square(s \leftrightarrow \bigcirc e)$$

The other branches are similar. The branch above should be seen as a strategy that, no matter what the environment does at the start, the system chooses to satisfy s, which in turn forces the environment to satisfy e in the next-step. Obviously, since the environment could freely choose $\neg e$ this tableau branch should be closed at node $e, \square(s \leftrightarrow \bigcirc e)$. A branch closing condition would successfully solve this example: whenever it is the environment turn, if there is a literal on an environment variable, the branch is closed (the environment is the winner by choosing a contradiction). This closing condition however fails to capture the

realizability of safety specifications such as $\Box\psi_3$. The tableau for $\Box(\bigcirc s \leftrightarrow \bigcirc e)$ would be formed by the following two closed branches:

$$\Box(\bigcirc s \leftrightarrow \bigcirc e), \bigcirc s \leftrightarrow \bigcirc e, \bigcirc\Box(\bigcirc s \leftrightarrow \bigcirc e), \bigcirc s, \bigcirc e$$
$$s, e, \Box(\bigcirc s \leftrightarrow \bigcirc e)$$

$$\Box(\bigcirc s \leftrightarrow \bigcirc e), \bigcirc s \leftrightarrow \bigcirc e, \bigcirc\Box(\bigcirc s \leftrightarrow \bigcirc e), \bigcirc\neg s, \bigcirc\neg e$$
$$\neg s, \neg e, \Box(\bigcirc s \leftrightarrow \bigcirc e)$$

According to our previous closing condition, the first branch is closed by e and the second by $\neg e$, whereas the safety specification $\Box(\bigcirc s \leftrightarrow \bigcirc e)$ is realizable. The problem here is in the splitting of the two cases $\bigcirc s, \bigcirc e$ and $\bigcirc\neg s, \bigcirc\neg e$. Intuitively, the environment should be free to choose at the second state (really, at any state) either e or $\neg e$, but in the second state of the two branches above the choice is already taken, because the splitting of the 'strict' future formula $\bigcirc s \leftrightarrow \bigcirc e$ has been already made at the first state. 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 state, our tableau will have just one node:

$$(s \wedge e) \vee (\neg s \wedge \neg e), \Box(\bigcirc s \leftrightarrow \bigcirc e) \tag{1}$$

This node has two children (one for each choice of the environment):

$$e, s, \bigcirc((s \wedge e) \vee (\neg s \wedge \neg e)), \bigcirc\Box(\bigcirc s \leftrightarrow \bigcirc e)$$

$$\neg e, \neg s, \bigcirc((s \wedge e) \vee (\neg s \wedge \neg e)), \bigcirc\Box(\bigcirc s \leftrightarrow \bigcirc e)$$

Then, 'jumping' to the next-state from both nodes produces again node (1). This tableau witnesses (see Ex. 7) that $\Box\psi_3$ is realizable. In summary, classical tableau rules for the logical operators does not provide a correct decision procedure for realizability.

In this paper we present a tableau method that decides the realizability problem for a fragment of LTL which includes temporal operators of the form $\Box_{[n,m]}$ and $\Diamond_{[n,m]}$ (for $n, m \in \mathbb{N}$ such that $n \leq m$). Although, from the semantic point of view, these operators can be seen as a short-hand for a boolean combinations of formulas using only the "next time" operator ($\bigcirc$), the compact notation is effectively exploited in tableau deductions in a more efficient way that prevents their unfoldings. As an example of such benefits consider the safety formula $\psi_4 = e \rightarrow \Box_{[0,2^{100}]}s$. A tableau for $\Box\psi_4$ according to the rules we propose, splits two branches for the two cases $(\neg e \wedge \bigcirc\Box\psi_4)$ and $(e \wedge s \wedge \bigcirc\Box_{[0,2^{100}-1]}s \wedge \bigcirc\Box\psi_4)$. The first branch ends up jumping to the next state, which loops to the root $\Box\psi_4$. The second branch jumps to the next state with the formula $(\Box_{[0,2^{100}-1]}s \wedge \Box\psi_4)$ and again two new branches start from here. Each of these branches ends up with a loop to its previous state and the tableau finishes certifying that $\Box\psi_4$ is realizable.

This example illustrates a crucial difference between automata and tableaux: the tableau "deduction", after checking two successive states, is able to decide the

realizability of $\Box\psi_4$, whereas automata techniques require an explicit blasting. Throughout the paper we provide more examples remarking this point. As far as we know, there is no previous temporal tableaux for solving the realizability problem of this logic, so no technique for reactive realizability/synthesis enjoys the benefits of deductive temporal tableaux. Though this paper is mainly dedicated to the realizability problem, our tableaux provides an easy way for synthesis of both kind of certificates: the realizability strategy and the counterexample in the case of unrealizability. Indeed, the constructed tableau is really a deductive proof of the un/realizability of the input formula.

In summary, the contributions of this paper are:

- The introduction of a new normal form, called terse normal form that captures precisely, in a logical form, the timely choices of the environment and the responses by the system.
- A tableau method including all the deductive rules to build the tableau graph and rules to close the branches, with success and with failure. Some of these rules are non-deterministic, which enable heuristics for exploring the graph in different forms (depth-first, breadth first, etc).
- Sound and completeness proofs for the tableau method.

Related Work. Current approaches to reactive synthesis [4,9,14,15,20] can be classified into three categories: (1) approaches based on games [6] that translate φ into a deterministic automaton and determine the winner of the game played on the state graph of the automaton, (2) approaches that cover a strict fragment of LTL, like GR(1) specifications (the simpler fragment of general reactivity of rank 1) [23,2]; (3) bounded synthesis [27], which process a set of constraints that characterizes all correct systems up to fixed bound on the size. Modern implementations of the the game approach use a symbolic representation [20] of the arena of the game, or use decision procedures based on SAT or QBF [3]. Existing tools for full LTL synthesis, including Unbeast [9] and Acacia+ [4] are based on the bounded synthesis idea, where different encoding of the (decidable) constraint for a given bound have been proposed [13,27,14] (first-order logic module finite integer arithmetic), using lazy constraint generation [11], rewriting and modular solving [21], SAT encodings [29,12], QBF encoding [10], etc. Since 2014, the reactive synthesis competition (SYNTCOMP) [1,19] tries to compare the performance of synthesis tools against different benchmark problems.

Reactive synthesis for full LTL specifications is 2EXPTIME-complete [26], so synthesis for more restricted fragments of LTL that exhibit better complexity has been identified. For example, synthesis of GR(1) specifications—the simpler fragment of general reactivity of rank 1—has an efficient polynomial time symbolic synthesis algorithm [23,2]. GR(1) synthesis has been used in various application domains and contexts, including robotics [22], event-based behavior models [8], etc. It is easy to see that even though GR(1) covers the safety fragment of LTL, translating specifications into the language that we consider in this paper involves at least an exponential blow-up in the worst case. All the methods listed above correspond to an algorithmic exhaustive exploration of the

game arena. In contrast, we introduce in this paper a deductive tableau method for realizability.

The first tableau method [32] for the satisfiability problem of LTL is not purely tree-shape, instead they require the constructions of a graph that should be explored in a second pass. This inspired a connection with Büchi automata [32,31] that is in the origin of many automata-based decision procedures [7] for LTL satisfiability and model-checking problems. The use of an auxiliary structure raised two difficulties: One is the construction of a big structure and the other is the lost of the original correspondence with sequent proofs that could certify the result. Some alternative ideas (e.g. [28,17]) has been developed to explore on-the-fly the graph (or automaton) thorough its construction instead in a second pass, and also for constructing one-pass tableaux that preserves the correspondence with sequent proofs (cf. [16]).

Outline of the paper. The rest of the paper is structured as follows. Section 2 presents the fragment of LTL for safety specifications. Section 3 introduces the terse normal form. which is allows the realizability tableaux, presented in Section 4. Several examples are shown in Section 5. Section 6 includes and proofs of correctness and finally, Section 7 concludes.

2 Safety Specifications and Games

In this section, we first define a sublanguage of LTL to express safety properties. Our language is aimed at industrial specifications of reactive systems. We present a method to decide realizability of these specifications in Section 4. We will introduce safety games and games for safety specifications, which will serve as method to prove correctness of our approach. We start by revisiting LTL.

2.1 Linear Temporal Logic

Linear Temporal Logic (LTL) [24], extends propositional logic by introducing temporal operators $\bigcirc$ (next) and $\mathcal{U}$ (until). LTL formulas are interpreted in traces on their set of (occurring) variables $\mathcal{V}$. A *trace* σ is a denumerable sequence of states $\sigma_0, \sigma_1, \sigma_2, \ldots$ where each state σ_i is a valuation from $\mathcal{V}$ to $\{0,1\}$. A valuation v on a set of variables $\mathcal{V}$ is a function that assigns to each $p \in \mathcal{V}$ a boolean value $\{0,1\}$. We denote by $\mathsf{Val}(\mathcal{V})$ the set of all valuations on $\mathcal{V}$. For any $i \geq 0$, σ^i denotes the trace $\sigma_i, \sigma_{i+1}, \ldots$ and $\sigma^{<i}$ denotes the finite sequence $\sigma_0, \ldots, \sigma_{i-1}$. Therefore, $\sigma = \sigma^{<i} \cdot \sigma^i$. We also use $\sigma^{i..j}$ for the finite sequence between i and j (including σ_i but not σ_j), which can be defined as $(\sigma^i)^{<j}$. Given a trace σ and an LTL formula ϕ the meaning of $\sigma \models \phi$ is inductively defined as follows:

$$
\begin{array}{lll}
\sigma \models p & \text{iff} & \sigma_0(p) = 1 \\
\sigma \models \neg\phi & \text{iff} & \sigma \not\models \phi \\
\sigma \models \phi \wedge \psi & \text{iff} & \sigma \models \phi \text{ and } \sigma \models \psi \\
\sigma \models \bigcirc\phi & \text{iff} & \sigma^1 \models \phi
\end{array}
$$

$$\sigma \models \phi \mathcal{U} \psi \text{ iff } \sigma^j \models \psi \text{ for some } j \text{ such that } 0 \leq j \text{ and } \sigma^i \models \phi \text{ for all } i \text{ such that } 0 \leq i < j.$$

There are standard abbreviations of constants propositions ($\mathbf{T}$ for truth and $\mathbf{F}$ for falsehood), classical connectives (such as $\vee, \rightarrow$ and $\leftrightarrow$) and also of temporal operators such as 'eventually': $\Diamond\phi$ for $\mathbf{T}\,\mathcal{U}\,\phi$ and 'always': $\Box\phi$ for $\neg(\mathbf{T}\,\mathcal{U}\,\neg\phi)$.

A set of formulas is (syntactically) consistent if and only if it does not contain a formula and its negation. If $\sigma \models \phi$ then we say that σ models (or is a model of) ϕ. We denote $\mathsf{Mod}(\phi)$ to the set of all traces that are models of ϕ. A finite set of formulas is intentionally confused with the conjunction of all it members. Therefore, for a set of formulas Φ, $\sigma \models \Phi$ if and only if $\sigma \models \phi$ for all $\phi \in \Phi$ and $\mathsf{Mod}(\Phi)$ denotes the set of traces that are models of all $\phi \in \Phi$. A set of formulas Φ is satisfiable if and only if there exists at least one σ such that $\sigma \models \Phi$, otherwise Φ is unsatisfiable. Two formulas ϕ and ψ are logically equivalent, denoted $\phi \equiv \psi$, if and only if $\mathsf{Mod}(\phi) = \mathsf{Mod}(\psi)$.

Any sublanguage $\mathcal{G}$ of LTL is *safety* whenever for any formula $\phi \in \mathcal{G}$ and for any trace σ, if $\sigma \not\models \phi$ then there exists some $i > 0$ such that $\sigma^{<i} \cdot \sigma' \not\models \phi$ for any trace σ'. In words, any trace with prefix $\sigma^{<i}$ is not a model of ϕ. In such case, $\sigma^{<i}$ is called a *witness of the violation* of ϕ (a.k.a. a *bad prefix* for ϕ).

2.2 Safety Specifications

Our safety specifications are constructed over some set of variables $\mathcal{X} \cup \mathcal{Y}$ that is partitioned into two subsets: $\mathcal{X}$ is the set of variables that are controlled by the environment and $\mathcal{Y}$ is the set of variables controlled by the system. In what follows, each variable in $\mathcal{X}$ is marked with e as a subscript (e.g. $sensor_e$ or p_e). Variables in $\mathcal{Y}$ do not have any subscript.

Boolean formulas are constructed from atoms that could be either propositional variables or equations $x = c$ where x is a variable of an enumerated type T and c is a constant value of type T. Compound formulas are constructed using the classical boolean connectives ($\neg, \wedge, \vee, \rightarrow, \leftrightarrow$). We also use the boolean constants $\mathbf{T}$ (for truthness) and $\mathbf{F}$ (for falsehood) as atomic formulas. More precisely, the grammar for any boolean formula η is:

$$a ::= p \mid x = c \mid \mathbf{T} \mid \mathbf{F}$$
$$\beta ::= a \mid \neg\beta \mid \beta \wedge \beta \mid \beta \vee \beta \mid \beta \rightarrow \beta \mid \beta \leftrightarrow \beta$$

where p is a boolean variable, x has enumerated type T and $c \in T$. The formulas of the form $a, \neg a$, $x = c$ and $\neg(x = c)$ are called literals. We consider that the valuations in traces assign to each variable x of enumerated type T a value in T, which gives the atom $x = c$ a truth value.

We consider a fragment of LTL specifications of the form $\alpha \wedge \Box\psi$ where α is an initial formula and ψ is a safety formula. The formula α, called the *initial formula*, is a boolean constraint that captures the initial states The formula $\Box\psi$ *safety constraint* that restricts the transition relation by means of temporal operators. Here, ψ is a temporal formula over $\mathcal{X} \cup \mathcal{Y}$, called the *safety formula*,

whose grammar is defined as follows:

$$\eta ::= \beta \mid \neg\eta \mid \bigcirc\eta \mid \Box_I\eta \mid \Diamond_I\eta \mid \eta \vee \eta \mid \eta \wedge \eta \mid \eta \rightarrow \eta \mid \eta \leftrightarrow \eta$$

where $I = [n, m]$ for Therefore, safety formulas are constructed adding to the boolean connectives the temporal operators $\Diamond_I$ and $\Box_I$. The interpretation of this operators in traces are

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

Note that $\Diamond_I$ (resp. $\Box_I$) can be expressed as a disjunction (resp. conjunction) of formulas that only use $\bigcirc$ as temporal operator. A trace σ models $\alpha \wedge \Box\psi$ whenever $\sigma_0(\alpha) = 1$ 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), i. e. a formula in the following grammar:

$$\begin{array}{l} \ell ::= p \mid \neg p \mid x = c \mid \neg(x = c) \mid \mathbf{T} \mid \mathbf{F} \\ \eta ::= \ell \mid \bigcirc\eta \mid \Box_I\eta \mid \Diamond_I\eta \mid \eta \vee \eta \mid \eta \wedge \eta \end{array}$$

It suffices to use, besides classical logical equivalences on boolean connectives, the following equivalences on temporal connectives:

$$\neg\bigcirc\eta \equiv \bigcirc\neg\eta \;\; \neg\Diamond_I\eta \equiv \Box_I\neg\eta \;\; \neg\Box_I\eta \equiv \Diamond_I\neg\eta$$

In what follows, we assume that safety formulas are translated to their equivalent formulas in 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 a safety formula is the maximum number of nested $\bigcirc$ operators in the formula, where $\Box_I$ and $\Diamond_I$ are interpreted as being expressed in terms of $\bigcirc$. Formally,

$$\mathsf{depth}(\eta) = \begin{cases} 0 & \eta \text{ is a literal} \\ max\{\mathsf{depth}(\eta_1), \mathsf{depth}(\eta_2)\} & \eta = \eta_1 \vee \eta_2 \text{ or } \eta = \eta_1 \wedge \eta_2 \\ 1 + \mathsf{depth}(\eta_1) & \eta = \bigcirc\eta_1 \\ m + \mathsf{depth}(\eta_1) & \eta = \Box_{[n,m]}\eta_1 \text{ or } \eta = \Diamond_{[n,m]}\eta_1 \end{cases}$$

The following result states that the language of our safety formulas is a safety sublanguage of LTL.

Lemma 1. *For any safety formula η and any trace σ, $\sigma \not\models \eta$ if and only if exists $d \leq \mathsf{depth}(\eta)$ such that $\sigma^{<d+1}$ is a witness of the violation of η.*

Proof. The backward direction holds by the definition of witness of the violation of η. The proof of the forward direction is done by structural induction on η. □

Corollary 1. *For any safety specification $\varphi = \alpha \wedge \Box\psi$ and any trace σ, $\sigma \not\models \varphi$ if and only if one of the following holds:*

(i) σ_0 *is a witness of the violation of* $\alpha \wedge \psi$*, or*
(ii) there exists some i *and some* $d \leq \mathsf{depth}(\psi)$ *such that* $\sigma^{i..d+1}$ *is a witness of the violation of* ψ*.*

We next introduce the concept of *witness of a safety specification*, which allows us to work with a finite semantics for safety specifications.

Definition 1. *We define the relation* $\models^{fin}$ *on finite sequences* $\lambda = \lambda_0 \cdots \lambda_{d-1}$ *where* $d \geq 1$ *and formulas as follows:*

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

A pre-witness of a safety specification $\alpha \wedge \Box\psi$ *is a finite sequence* $\lambda = \lambda_0 \cdots \lambda_{d-1}$ *such that* $\lambda_0 \models^{fin} \alpha \wedge \psi$ *and* $\lambda^{<i} \models^{fin} \psi$ *for every* $1 \leq i \leq d-1$*. A witness of a safety specification* $\alpha \wedge \Box\psi$ *is a pre-witness* $\lambda_0, \cdots, \lambda_{d-1}$ *such that for some* $0 \leq j < d$ *it holds that* $\lambda_0 \cdots (\lambda_j \cdots \lambda_{d-1})^\omega \models \alpha \wedge \Box\psi$*.*

To construct traces, we usually deal with valuations that satisfies a set of literals.

Definition 2. *Let* Δ *be any set of formulas, we denote by* $\mathsf{Val}_\Delta(\mathcal{V})$ *the set of all valuations* $v \in \mathsf{Val}(\mathcal{V})$ *such that* $v(x) = 1$ *for every boolean variable* $x \in \Delta$*,* $v(x) = 0$ *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 for each variable x that does not occur in Δ, there are many $v \in \mathsf{Val}_\Delta(\mathcal{V})$ with different values for $v(x)$ and also that if Δ is a set of literals then $\lambda_0 \models^{fin} \Delta$ if and only if $\lambda_0 \in \mathsf{Val}_\Delta(\mathcal{V})$.

In safety specifications, variables come from two disjoint sets of variables $\mathcal{X}$ and $\mathcal{Y}$. Given $v \in \mathsf{Val}(\mathcal{X})$ and $w \in \mathsf{Val}(\mathcal{Y})$, we denote by $v + w$ the valuation in $z \in \mathsf{Val}(\mathcal{X} \cup \mathcal{Y})$ such that $z(p) = v(p)$ if $z \in \mathcal{X}$ and $z(p) = w(p)$ if $z \in \mathcal{Y}$. This notation is extended to pairs of finite traces λ on $\mathcal{X}$ and λ' on $\mathcal{Y}$ of the same length d, i.e. $\lambda + \lambda'$ denotes the trace $(\lambda_0 + \lambda'_0) \cdots (\lambda_{d-1} + \lambda'_{d-1})$.

2.3 Safety Games, Games for Safety Specifications

We now give an introduction to safety games and games for safety specifications.

Safety Games. A *safety game* is a game played by two players: Eve (the environment) denoted by E, and Sally (the system), denoted by S. The set of *positions* of the game is partitioned into those position where Eve plays and

those where Sally plays. A *move* consists on the player that owns the current position choosing a successor position. A *play* is a infinite sequence of moves starting from some positions within a predetermined set of initial positions. The outcome of a play of the game is determined as follows. Eve wins if some move during the play reaches some bad positions for a predetermined subset of bad positions. Otherwise, Sally wins. In other words, Sally wins a play if she avoids bad positions at all times during the play.

Formally, an *arena* is a tuple $\mathcal{A} : \langle I, P, P_E, P_S, T\rangle$

- P is the set of positions, partitioned into $P = P_E \cup P_S$ and $P_E \cap P_S = \emptyset$
- $I \subseteq P$ is the set of initial positions;
- $T \subseteq (P \times P)$ is the set of moves; and

A play $\pi : v_0 v_1 v_2 \dots$ is an infinite sequence of positions $v_i \in P$. A game equips an arena with a winning condition $W \subseteq P^w$. A play π is winning for Sally if $\pi \in W$. Safety games are defined by a set of bad states $B \subseteq P$ that encodes the following winning condition $W_B = \{\pi \mid \text{for all } i\ , \pi(i) \notin B\}$ Therefore, Sally wins a safety game $\mathcal{G} : (\mathcal{A}, B)$ in those infinite plays where bad positions B are avoided.

We assume that every state has a successor so we do not have to deal with finite plays. This does not limit the class of games because we can extend every arena with a sink state that is either good (resp. bad) if the halting state is good (resp. bad), generating an equivalent arena that admits only infinite plays. Given a play and a natural number i, we use $\pi(i)$ to denote the i-th state of π, given two natural numbers $i < j$, $\pi[i, j]$ denotes the finite sequence $\pi(i), \dots, \pi(j)$, and $\pi(i..)$ denotes the infinite sequence $\pi(i)\pi(i+1)\dots$ A play is initial if $\pi(0) \in I$.

A *strategy* ρ_S for Sally is a map $\rho_S : P^* \times P_S \to P$, such that for every sequence of positions $s \in P^*$ followed by a position of Sally $v \in P_S$, $T(v, \rho_S(s, v))$ is a legal move in the game. A strategy is memory-less if for all $s_1, s_2 \in P^*$ and $v \in P_S$, $\rho_S(s_1, v) = \rho_S(s_2, v)$, that is, if the move that the strategy represents is determined solely by the last position. In this case we represent ρ_S as a function $\rho_S : P_S \to P$ and the move $\rho_S(v)$ is determined by v. Similarly, a strategy for Eve is a map $\rho_E : P^* \times P_E \to P$, such that for every $s \in P^*$ and $v \in P_E$, $T(v, \rho_E(s, v))$ is a legal move for Eve in the game. A play π is played according to a strategy ρ_S if for every i, if $\pi(i) \in P_S$ then $\pi(i+1) = \rho_S(\pi[0, i], \pi(i))$ (if ρ_S is memory-less then $\pi(i+1) = \rho_S(\pi(i))$). The definition is analogous for ρ_E. A strategy ρ_S of Sally is winning if every initial play π played according to ρ_S is winning for Sally. It is well-known that safety games are determined (either a game has a winning strategy for Sally or it has a winning strategy for Eve), and that safety games are memory-less determined (either a game has a memory-less winning strategy for Sally or it has a memory-less winning strategy for Eve).

Games for Safety Specifications. Consider a safety specification φ over $\mathcal{X}$ and $\mathcal{Y}$. Given a set S, S^* denotes the set of finite strings over S and S^k the set of strings over S of length k. The build an arena $\mathcal{A}(\mathcal{X}, \mathcal{Y})$ of a safety game as follows:

- $P_E = \{\mathsf{Val}(\mathcal{X})^k \times \mathsf{Val}(\mathcal{Y})^k \mid k \in \mathbb{N}\}$. We use $P_E^k = \{(\overline{x}, \overline{y}) \mid |x| = |y| = k\} \subseteq P_E$.
- $P_S = \{\mathsf{Val}(\mathcal{X})^{k+1} \times \mathsf{Val}(\mathcal{Y})^k \mid k \in \mathbb{N}\}$. We use P_S^{k+1} for the set $\{(\overline{x}, \overline{y}) \mid |x| = k+1 \text{ and } |y| = k\} \subseteq P_S$.
- 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 $((\overline{x}, \overline{y}), (\overline{x} \cdot v, \overline{y})) \in T_E$ if $v \in \mathsf{Val}(\mathcal{X})$.
  - $T_S \subseteq (P_S^{k+1}, P_E^{k+1})$ such that $((\overline{x} \cdot v, \overline{y}), (\overline{x} \cdot v, \overline{y} \cdot w)) \in T_S$ if $w \in \mathsf{Val}(\mathcal{Y})$.
- $I = \{(\epsilon, \epsilon)\}$.

The arena described above considers all sets of possibles moves by Eve and Sally. Note that Eve and Sally alternate playing. Given a position $p \in P_E \setminus I$ of the form $(\overline{x} \cdot v, \overline{y} \cdot w)$ we use $\mathit{move}(p) = (v + w)$ for the valuation of the variables of $\mathcal{X} \cup \mathcal{Y}$ according to v and w. Given a play π we use $\mathit{trace}(\pi)$ for the trace σ such that $\sigma(i) = \mathit{move}(\pi(2i+1))$. Note that $\mathit{trace}(\pi)$ corresponds to the sequence of valuations that Eve and Sally pick. The arena is essentially an infinite tree that records the valuations chosen in turns by Eve and Sally. Given a specification φ (for example, a safety specification), the set of winning plays is $W_\varphi = \{\pi \mid \mathit{trace}(\pi) \models \varphi\}$.

Definition 3 (Realizability). *Given a temporal formula φ over propositions $\mathcal{X}$ and $\mathcal{Y}$, the formula is called realizable if Sally wins the game $(\mathcal{A}(\mathcal{X}, \mathcal{Y}), W_\varphi)$.*

We now create a safety game defining a set of bad positions B of the arena, and show in Lemma 2 that the resulting safety game solves the realizability problem.

$$B = \{(\overline{x}, \overline{y}) \mid \text{there exists } v \in \mathsf{Val}(\mathcal{X}) \text{ such that for all } v' \in \mathsf{Val}(\mathcal{Y}) : \overline{x} \cdot v + \overline{y} \cdot v' \not\models^{fin} \alpha \wedge \Box\psi\}.$$

Even though we do not show it here, it is easy to see that in order to determine whether a state $(\overline{x}, \overline{y})$ of P_E is bad, it is sufficient to remember (1) whether some prefix was bad (for which only one bit is necessary) and (2) the last m elements of $(\overline{x}, \overline{y})$ where m is the maximum number of nested $\bigcirc$ operators in the specification. Hence, the arena of a safety specification can be turned into a finite arena. Therefore the realizability of safety specifications is decidable (by reduction into whether Sally has a winning strategy in a finite state safety game). Given a specification φ we call $\mathcal{G}(\varphi)$ for the safety specification game for φ.

Lemma 2. *Given a safety specification φ, φ is realizable if and only if $\mathcal{G}(\varphi)$ is winning for Sally.*

Proof. Assume that φ is realizable, and let ρ_S be a strategy for Sally in the specification game. Therefore any play π played according to ρ_S is winning for Sally, so $\pi \in W_\varphi$ i.e. $\mathit{trace}(\pi) \models \varphi$. Hence, by Corollary 1 there is no finite witness of the violation of $\mathit{trace}(\pi)$, consequently ρ_S is a winning strategy for Sally in $\mathcal{G}(\varphi)$. Similarly, assume that Sally wins $\mathcal{G}(\varphi)$ and let π be an arbitrary play played according to a winning strategy ρ_S. The play π in the specification game is winning for Sally as well, due to Corollary 1. □

3 Terse Normal Form

Our tableaux for a safety specification $\varphi = \alpha \wedge \Box\psi$ analyze its realizability on the basis of a play where the environment chooses a move on its variables $\mathcal{X}$ and, then, the system chooses its move on its variables $\mathcal{Y}$ according to the safety specification. In order to the branches of the tableau to represent the real play, any formula in a node should determine the true strict-future possibilities of the game. For example, considered that following formula in *disjunctive normal form* (DNF) is the representation of all possible moves at some state of the game:

$$(\bigcirc\neg s) \vee (p_e \wedge s \wedge \bigcirc\bigcirc s) \tag{2}$$

This means that, after any choice of the environment and the system, satisfying $(\bigcirc\neg s)$ would fulfill the specification. Moreover, after the environment moves p_e followed by the system moving s, not only $\bigcirc\bigcirc s$ would be coherent with the specification, but also $\bigcirc\neg s$. However, a classical tableau-style analysis would split (2) into two branches such that the one containing $p_e \wedge s$ requires $\bigcirc\bigcirc s$ to satisfy the specification, precluding the possibility of $\bigcirc\neg s$. Note also that the formula

$$(p_e \wedge s \wedge (\bigcirc\neg s \vee \bigcirc\bigcirc s)) \vee (\neg p_e \wedge \bigcirc\neg s) \vee (\neg s \wedge \bigcirc\neg s) \tag{3}$$

is logically equivalent to (2), but suitable for a tableau-style analysis of realizability. In this section, we define the *Terse Normal Form* (TNF) for safety formulas that allows us to associate to any move the formula that any trace must satisfy in the (strict) future to be coherent with the current safety specification. The formula (3) is in TNF.

We consider a partition of all the *basic* (sub)formulas that occurs in a safety formula—that is all the formulas of the forms ℓ, $\bigcirc^n\eta$, $\Diamond_I\eta$ or $\Box_I\eta$—into two classes of formulas:

- the class *from-now*: $\ell, \Diamond_{[0,m]}\eta, \Box_{[0,m]}\eta$
- the class *from-next*: $\bigcirc\eta$, $\Diamond_{[n,m]}\eta$ and $\Box_{[n,m]}\eta$ (for any $n \geq 1$).

Definition 4 (Strict-future and separated). *A safety formula is a strict-future formula if and only if it is a* DNF *combination of from-next formulas. A safety formula is a separated formula if and only if it is the (possibly empty) conjunction of a set of Boolean literals, denoted as $\mathcal{L}(\pi)$, and (at most) a strict-future formula, denoted as $\mathcal{F}(\pi)$.*

Definition 5 (TNF). *A safety formula η is in Terse Normal Form* (TNF) *if and only if it 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* TNF(η)*, that is in* TNF.

Proof. First, any safety formula η (for simplicity, suppose that η is in NNF) can be converted into a disjunctive normal form-like formula, that we call $\mathsf{DNF}(\eta)$, whose "literals" are either classical literals (p or $\neg p$), or from-next formulas. For constructing $\mathsf{DNF}(\eta)$, it suffices to use, besides classical logical equivalences on boolean connectives, the following equivalences on temporal connectives:

$$\Diamond_{[n,n]}\beta \equiv \bigcirc^n \beta \qquad \Box_{[n,n]}\beta \equiv \bigcirc^n \beta$$
$$\Diamond_{[n,m]}\beta \equiv \bigcirc^n \beta \vee \bigcirc\Diamond_{[n..m-1]}\beta \qquad \Box_{[n,m]}\beta \equiv \bigcirc^n \beta \wedge \bigcirc\Box_{[n..m-1]}\beta$$

Then, we transform each pair of disjuncts in $\mathsf{DNF}(\eta)$ with indexes $1 \leq i \neq j \leq n$ such that for all literal $\ell \in \mathcal{L}(\pi_i)$ it holds that $\neg\ell \notin \mathcal{L}(\pi_j)$ as follows. Let $\delta = \mathcal{L}(\pi_i) \cap \mathcal{L}(\pi_j)$, $\delta_1 = \mathcal{L}(\pi_i) \setminus \delta$ and $\delta_2 = \mathcal{L}(\pi_j) \setminus \delta$. Then, we apply

$$\begin{aligned}(\delta \wedge \delta_1 \wedge \eta_1) \vee (\delta \wedge \delta_2 \wedge \eta_2) \equiv\ & (\delta \wedge \delta_1 \wedge \delta_2 \wedge (\eta_1 \vee \eta_2)) \\ & \vee\ \mathsf{DNF}(\delta \wedge \delta_1 \wedge \neg\delta_2 \wedge \eta_1) \\ & \vee\ \mathsf{DNF}(\delta \wedge \neg\delta_1 \wedge \delta_2 \wedge \eta_2)\end{aligned} \tag{4}$$

where $\mathcal{F}(\pi_1) = \eta_1$ and $\mathcal{F}(\pi_j) = \eta_2$. Note that, in particular cases, δ, δ_1 or δ_2 could be empty (equivalently $\mathbf{T}$), hence their negation are equivalent to $\mathbf{F}$. In this cases, the equivalence (4) could be simplified. For example, if $\mathcal{L}(\pi_i) = \mathcal{L}(\pi_j)$, then $\delta = \mathcal{L}(\pi_i) \cap \mathcal{L}(\pi_j) = \mathcal{L}(\pi_i) = \mathcal{L}(\pi_j)$. Consequently, (4) is simplified to $(\delta \wedge \eta_1) \vee (\delta \wedge \eta_2) \equiv \delta \wedge (\eta_1 \vee \eta_2)$.

This transformation is repeatedly applied until every pair (π_i, π_j) satisfies the required condition. It is easy to see that the above process produces a formula in TNF. Moreover, since we only apply logical equivalences to subformulas, by substitutivity, the resulting formula $\mathsf{TNF}(\eta)$ is logically equivalent to η. □

Example 1. The TNF for $p_e \leftrightarrow \bigcirc s$ and $\bigcirc p_e \leftrightarrow \bigcirc s$ from Section 1 are:

$$\mathsf{TNF}(p_e \leftrightarrow \bigcirc s) = (p_e \wedge \bigcirc s) \vee (\neg p_e \wedge \bigcirc \neg s)$$
$$\mathsf{TNF}(\bigcirc p_e \leftrightarrow \bigcirc s) = (\bigcirc p_e \wedge \bigcirc s) \vee (\bigcirc \neg p_e \wedge \bigcirc \neg s)$$

Note that $\mathsf{TNF}(p_e \leftrightarrow \bigcirc s)$ is composed by two disjuncts, each having a literal and a strict-future formula, but $\mathsf{TNF}(\bigcirc p_e \leftrightarrow \bigcirc s)$ has only one move (the empty set of literals) with one future-strict formula (which is a disjunction). Finally, for $\eta = c \wedge (\neg p_e \rightarrow \Box_{[0,9]}\neg c) \wedge (\Box_{[0,9]} c \vee \Diamond_{[0,2]}\neg c)$:

$$\mathsf{TNF}(\eta) = (p_e \wedge c \wedge (\bigcirc\Diamond_{[0,1]}\neg c \vee \bigcirc\Box_{[0,8]} c)) \vee (\neg p_e \wedge c \wedge \bigcirc\Box_{[0,8]} c).$$

For any formula $\bigvee_{i=1}^{n} \pi_i$ in TNF and any $1 \leq i \leq n$, we denote by $\mathcal{L}(\pi_i)$ the conjunction (or set) of literals in π_i and by $\mathcal{F}(\pi_i)$ the unique strict-future formula in π_i. Abusing language, we say that each π_i is a *move*, though it could really represent a collection of pairs of moves (one of the environment followed by one of the system) because the variables that do not appear in $\mathcal{L}(\pi_i)$ could be freely chosen. Note that $\mathsf{Val}_{\pi_i} = \mathsf{Val}_{\mathcal{L}(\pi_i)}$ for any move π_i of any formula in TNF.

Proposition 2. *Let η be a safety formula and $\mathsf{TNF}(\eta) = \bigvee_{i=1}^{n} \pi_i$*

(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,* $\sigma \models \mathcal{L}(\pi_i) \rightarrow \mathcal{F}(\pi_i)$.

Proof. Items (*a*) and (*b*) are easy consequences of Definition 5 and the fact that $\eta \equiv \mathsf{TNF}(\eta)$ (Proposition 1). For item (*c*), consider any trace σ and any $1 \leq i \leq n$ such that $\sigma \models \eta$ and $\sigma \models \mathcal{L}(\pi_i)$. Then, $\sigma_0 \in \mathsf{Val}_{\mathcal{L}(\pi_i)}(\mathcal{X} \cup \mathcal{Y})$. Let us suppose that $\sigma \not\models \mathcal{F}(\pi_i)$ then, by (*a*), it should exists $1 \leq j \leq n$ such that $i \neq j$, $\sigma \models \pi_j$. Then, $\sigma \models \mathcal{L}(\pi_j)$ and $\sigma_0 \in \mathsf{Val}_{\mathcal{L}(\pi_j)}(\mathcal{X} \cup \mathcal{Y})$. The latter contradicts $\sigma_0 \in \mathsf{Val}_{\mathcal{L}(\pi_i)}(\mathcal{X} \cup \mathcal{Y})$, because there is a literal ℓ such that $\ell \in \mathcal{L}(\pi_i)$ and $\neg\ell \in \mathcal{L}(\pi_j)$. □

Many moves in formulas in TNF could be discharged for realizability test purposes. In a practical implementation of our method, we will discharge them. However, for simplicity of the presentation and the proofs of soundness and completeness, we consider all the moves in the TNF of a formula.

From now on we fix a safety specification on variables $\mathcal{X} \cup \mathcal{Y}$ whose safety formula is ψ. Consider any formula $\bigvee_{i=1}^{n} \pi_i$ in TNF. By Proposition 2, the collection $\{\mathsf{Val}_{\pi_i}(\mathcal{X} \cup \mathcal{Y}) \mid 1 \leq i \leq n\}$ is pairwise disjoint. Next, we define properties of the collection $\{\mathsf{Val}_{\pi_i}(\mathcal{X}) \mid 1 \leq i \leq n\}$.

Definition 6. *A formula* $\bigvee_{i=1}^{n} \pi_i$ *in* TNF *is an* $\mathcal{X}$*-covering if and only if*

$$\bigcup_{i=1}^{n} \mathsf{Val}_{\pi_i}(\mathcal{X}) = \mathsf{Val}(\mathcal{X}).$$

A formula $\bigvee_{i=1}^{n} \pi_i$ *in* TNF *is a minimal* $\mathcal{X}$*-covering if it is a* $\mathcal{X}$*-covering and* $\bigvee_{i=1, i\neq j}^{n} \pi_i$ *is not an* $\mathcal{X}$*-covering for every* $1 \leq j \leq n$.

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

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

Example 3. Let $\mathsf{TNF}(\eta) = (p_e \wedge c \wedge \eta_1) \vee (\neg p_e \wedge q_e \wedge c \wedge \eta_2) \vee (\neg c \wedge \eta_3)$ where η_1, η_2, η_3 are strict-future formulas and $\mathcal{X} = \{p_e, q_e\}$. $\mathsf{TNF}(\eta)$ is a non-minimal $\mathcal{X}$-covering, but the third move $\neg c \wedge \eta_3$ is a minimal $\mathcal{X}$-covering, However, the disjunction of the two first moves is not an $\mathcal{X}$-covering, because the valuation that makes both environment variables to be false is not included there.

Example 4. The following two TNF formulas (where $\eta_1, \eta_2, \eta_3, \eta_4$ are strict-future formulas and $\mathcal{X} = \{p_e\}$) are each composed by four different minimal $\mathcal{X}$-coverings:

$$(p_e \wedge c \wedge \eta_1) \vee (\neg p_e \wedge c \wedge \eta_2) \vee (p_e \wedge \neg c \wedge \eta_3) \vee (\neg p_e \wedge \neg c \wedge \eta_4)$$

$$(p_e \wedge c \wedge \eta_1) \vee (\neg p_e \wedge \neg c \wedge \eta_2) \vee (p_e \wedge \neg c \wedge d \wedge \eta_3) \vee (\neg p_e \wedge c \wedge \neg d \wedge \eta_4).$$

Abusing language, we say that a set of indices I is a (minimal) $\mathcal{X}$-covering when really the formula $\bigvee_{i\in I}\pi_i$ is a (minimal) $\mathcal{X}$-covering.

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

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

Proof. For item (a), there exists $v\in\mathsf{Val}(\mathcal{X})$ such that $v\not\models^{fin}\pi_i$ for all $1\leq i\leq n$. By Proposition 2, $v\not\models^{fin}\Phi\wedge\psi$. Items (b) and (c), are easy consequences of Definition 6. □

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 $\ddot{\bigvee}_{i=1}^{m}\delta_i$.

4 Realizability Tableaux

Our tableaux are AND-OR trees of nodes, each labelled by a set of formulas. Hence, each node has 0, 1 or more AND-successors or OR-successors. A node is said to be the parent of its successors nodes. In the examples below we mark AND-nodes with a semicircle embracing all the edges to the AND-successors of a node. A tableau is constructed for an input safety specification that is the root of the tree. The set of tableau rules (see Figures 1, 2 and 3) determine how the tableau can be constructed. Each rule determines the labels on the children of a node and the kind (AND or OR) of its successors. A tableau is completed (or finished) when no further rule can be applied. Rules can be applied 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 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 precise below) by some previous node in the branch from the root to the leaf.

The following definition formalizes our notion of tableau in terms of many concepts that will be precised in the next subsections.

Definition 7. *A tableau for a safety specification $\varphi=\alpha\wedge\Box\psi$ is a labelled tree $\mathsf{Tab}(\varphi)=(N,\tau,R)$, where N is a set of nodes, τ is a mapping of the nodes of T to formulas in $\mathsf{Clo}(\varphi)$ and $R\subseteq N\times N$, such that the following conditions hold:*

– *The root is labelled by the set $\{\alpha,\Box\psi\}$.*

- *For any pair of nodes $(n, n') \in R$, $\tau(n')$ is the set of formulas obtained as the result of the application of one of the tableau rules (in Figures 1, 2 and 3) to $\tau(n)$. Given the applied rule is ρ, we term n' a ρ-successor of n*
- *For every success or failure leaf n there is no $n' \in N$ such that $(n, n') \in R$ where:*
  - *A failure leaf is a node such that $n \in N$ such that $\mathsf{Incnst}(\tau(n))$ (see Definition 9).*
  - *A success leaf is a node $n \in N$ such that $\Box\psi \in \tau(n)$ and there exists $k \geq 0$, $n_0, \ldots, n_k \in N$ such that $(n_i, n_{i+1}) \in R$ for all $0 \leq i < k$, $(n_k, n) \in R$ and $\tau(n_0) \lessdot \tau(n)$ (see Definition 10).*

For any tableau rule ρ, the set of ρ-successors of the node to which ρ is applied is the set of children generated by ρ. Note that the ρ-successors of a node are OR-siblings for any rule ρ except for the rule ($\Box\&$) which are AND-siblings. In examples, we use an arc for linking the edges of the nodes that are AND-siblings, and no mark in the edges of OR-siblings.

4.1 Subsumption and Syntactical Inconsistency

The set of formulas used to label our tableaux nodes are subsumption-free with respect to the collection of subsumption rules between temporal formulas given in Definition 8 and classical subsumption on boolean formulas. Let $\beta \sqsubseteq \gamma$ denote the fact that β *subsumes* γ or that γ *is subsumed by* β. Subsumption is related to logical implication or logical consequence in the sense that, if $\beta \sqsubseteq \gamma$, then $\models \beta \rightarrow \gamma$ or equivalently $\beta \models \gamma$. Note that the converse is not necessarily true. Classical subsumption rules includes $\beta \sqsubseteq \beta$, $\beta \wedge \gamma \sqsubseteq \beta$, and $\beta \sqsubseteq \beta \vee \gamma$.

Definition 8. *The subsumption rules for temporal formulas that apply in our tableau method are:*

- *For all $n \leq n'$ and $m' \leq m$[3],*
  $\Diamond_{[n',m']}\beta \sqsubseteq \Diamond_{[n,m]}\beta$, $\Box_{[n,m]}\beta \sqsubseteq \Box_{[n',m']}\beta$, and $\Box_{[n',m']}\beta \sqsubseteq \Diamond_{[n,m]}\beta$.
- *For all $n \leq k \leq m$: $\bigcirc^k\beta \sqsubseteq \Diamond_{[n,m]}\beta$ and $\Box_{[n,m]}\beta \sqsubseteq \bigcirc^k\beta$.*

The following result easily follows from Def. 8 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, $\sigma \not\models \beta \wedge \widetilde{\gamma}$ and $\lambda \not\models^{fin} \beta \wedge \widetilde{\gamma}$ for any σ and λ. (Note that $\widetilde{\gamma}$ means the negation normal form of $\neg\gamma$.)*

According to Proposition 4, subsumption is used to detect (syntactical) inconsistencies (see Definition 9(b)). Inconsistencies are used to close tableau branches by a failure leaf. No rule is applied to a node labelled by an inconsistent set.

[3] By construction of our tableaux, we always generate from formulas with starting interval at n new formulas were $n' = n$. Note also that $\bigcirc^n\beta = \Diamond_{[n,n]}\beta = \Box_{[n,n]}\beta$.

Definition 9. *A set of formulas Φ is* (syntactically) inconsistent *(denoted* $\mathsf{Incnst}(\Phi)$*) if and only if one of the following four conditions hold:*

(a) $\mathbf{F} \in \Phi$
(b) $\{\beta, \widetilde{\gamma}\} \subseteq \Phi$ *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* $\mathsf{Cnst}(\Phi)$*.*

Example 5. The set $\{\Box_{[2,8]}c, \bigcirc^5 \neg c\}$ is inconsistent because $\Box_{[2,8]}c \sqsubseteq \bigcirc^5 c$ and $\bigcirc^5 \neg c = \widetilde{\bigcirc^5} c$.

Inconsistencies are used to close tableau branches. No rule is applied to a node labelled by an inconsistent set, and this node is called a failure leaf.

We define the following subsumption-based order relation between sets of formulas for detecting successful leaves.

Definition 10. *For two given set of formulas Φ and Φ', we say that $\Phi \lessdot \Phi'$ 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, $\ddot{\bigvee}_{i=1}^{n} \Delta_i \sqsubseteq \ddot{\bigvee}_{j=1}^{m} \Gamma_j$ if and only if for all $1 \leq i \leq n$ there exists $1 \leq j \leq m$ such that $\Delta_i \lessdot \Gamma_j$.*

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

Proof. By Definition 10 and Proposition 4. □

No rule is applied to a node that is labelled by a set Φ', such that for some previous label Φ in the same branch (some path from the root) it holds that $\Phi \lessdot \Phi'$ (see Definition 7).

4.2 Tableau Rules

In this section, we introduce the tableau rules, along with the concepts and properties related with the sets of formulas generated from a safety specification according to this set of rules.

First, the *Always Rules* in Figure 1 provides a non-deterministic procedure of analyzing the minimal $\mathcal{X}$-coverings in the $\mathsf{TNF}(\Phi \wedge \psi)$ (see Definition 6 and Proposition 3). The rule ($\Box\&$) is the only rule in our system that produces AND-successors for splitting the cases of each minimal $\mathcal{X}$-covering.

Next, we introduce the set of rules that are used to decompose formulas into its constituents, in the usual way that tableau methods perform the so-called *saturation*. In our method, decomposition of formulas inside the conjunction (or sets) connected by the operator $\ddot{\vee}$ just performs an unfolding in the formula. The *Saturation Rules* in Figure 2 are used to saturate with respect to classical connectives $\wedge$ and $\vee$ (including $\ddot{\vee}$) and temporal operators $\Diamond_I$ and $\Box_I$.

$$(\Box\mathbf{F})\ \frac{\Phi, \Box\psi}{\mathbf{F}, \Box\psi} \quad \text{if } \tau \text{ is not an } \mathcal{X}\text{-covering}$$

$$(\Box\|)\ \frac{\Phi, \Box\psi}{\bigvee_{i\in J_1} \pi_i, \Box\psi \ \mid \cdots \mid \ \bigvee_{i\in J_m} \pi_i, \Box\psi} \quad \text{if } J_1, \ldots, J_m \text{ is the collection of all minimal } \mathcal{X}\text{-covering of } \tau$$

$$(\Box\&)\ \frac{\bigvee_{i\in I} \pi_i, \Box\psi}{\pi_1, \bigcirc\Box\psi \ \ \& \ \ldots \ \ \& \ \pi_n, \bigcirc\Box\psi} \quad \text{if } I \text{ is a minimal } \mathcal{X}\text{-covering}$$

Fig. 1. Always Rules (where τ denotes $\mathsf{TNF}(\Phi \wedge \psi)$)

$$(\vee)\ \frac{\Phi, \beta \vee \gamma}{\Phi, \beta \ \mid \ \Phi, \gamma} \qquad (\ddot{\vee}\ \vee)\ \frac{\Phi, (\eta \wedge (\beta \vee \gamma))\ddot{\vee}\delta}{\Phi, (\eta \wedge \beta)\ddot{\vee}(\eta \wedge \gamma)\ddot{\vee}\delta}$$

$$(\wedge)\ \frac{\Phi, \beta \wedge \gamma}{\Phi, \beta, \gamma}$$

$$(\Diamond <)\ \frac{\Phi, \ \Diamond_{[n,m]}\beta}{\Phi, \ \bigcirc^n\beta \ \mid \ \Phi, \bigcirc\Diamond_{[n,m-1]}\beta} \ \text{if } n < m$$

$$(\ddot{\vee}\Diamond <)\ \frac{\Phi, (\eta \wedge \Diamond_{[n,m]}\beta)\ddot{\vee}\delta}{\Phi, (\eta \wedge \bigcirc^n\beta)\ddot{\vee}(\eta \wedge \bigcirc\Diamond_{[n,m-1]}\beta\}\ddot{\vee}\delta} \ \text{if } n < m$$

$$(\Diamond =)\ \frac{\Phi, \ \Diamond_{[n,n]}\beta}{\Phi, \ \bigcirc^n\beta} \qquad (\ddot{\vee}\Diamond =)\ \frac{\Phi, (\eta \wedge \Diamond_{[n,n]}\beta)\ddot{\vee}\delta}{\Phi, (\eta \wedge \bigcirc^n\beta)\ddot{\vee}\delta}$$

$$(\Box <)\ \frac{\Phi, \ \Box_{[n,m]}\beta}{\Phi, \bigcirc^n\beta, \bigcirc\Box_{[n,m-1]}\beta} \ \text{if } n < m \qquad (\Box =)\ \frac{\Phi, \ \Box_{[n,n]}\beta}{\Phi, \bigcirc^n\beta}$$

$$(\ddot{\vee}\Box <)\ \frac{\Phi, (\eta \wedge \Box_{[n,m]}\beta)\ddot{\vee}\delta}{\Phi, (\eta \wedge \bigcirc^n\beta \wedge \bigcirc\Box_{[n,m-1]}\beta)\ddot{\vee}\delta} \ \text{if } n < m$$

$$(\ddot{\vee}\Box =)\ \frac{\Phi, (\eta \wedge \Box_{[n,n]}\beta)\ddot{\vee}\delta}{\Phi, (\eta \wedge \bigcirc^n\beta)\ddot{\vee}\delta}$$

Fig. 2. Saturation Rules

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

Proof. By routinely applying semantics. □

Definition 11. *A next-formula is a formula whose first symbol is* $\bigcirc$*. A strict-future formula* $\ddot{\bigvee}_{i=1}^{n}\Delta_i$ *is elementary if every formula in the set* $\bigcup_{i=1}^{n}\Delta_i$ *is a next-formula.*

Proposition 7. *For any given strict-future formula* δ*, there exists an elementary formula that we call* δ^E *such that* $\delta \equiv \delta^E$ *and* δ^E *is in* DNF*.*

Proof. Repeatedly apply to δ the rules $(\ddot{\vee}\ \wedge)$, $(\ddot{\vee}\Diamond <)$, $(\ddot{\vee}\Diamond =)$, $(\ddot{\vee}\Box <)$ and $(\ddot{\vee}\Box =)$, until no one can be applied. □

Definition 12. *A set of formulas Δ is* saturated *if and only if for all $\delta \in \Delta$ the following conditions holds:*
- *If $\delta = \beta \wedge \gamma$, then $\{\beta, \gamma\} \in \Delta$.*
- *If $\delta = \beta \vee \gamma$, then $\beta \in \Delta$ or $\gamma \in \Delta$.*
- *If $\delta = \Box_{[n,m]}\beta$ and $n < m$, then $\{\bigcirc^n \beta, \bigcirc\Box_{[n,m-1]}\beta\} \subseteq \Delta$.*
- *If $\delta = \Diamond_{[n,m]}\beta$ and $n < m$, then either $\bigcirc^n \beta \in \Delta$. or $\bigcirc\Diamond_{[n,m-1]}\beta \in \Delta$*
- *If $\delta = \Box_{[n,n]}\beta$ or $\gamma = \Diamond_{[n,n]}\beta$, then $\bigcirc^n \beta \in \Delta$.*
- *If δ is a strict-future formula, then $\delta^E \in \Delta$*

We use $\mathsf{Stt}(\Delta)$ 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 $\Phi \in \mathsf{Stt}(\Delta)$.*
- *$\lambda \models^{fin} \Delta$ if and only if $\lambda \models^{fin} \Phi$ for some $\Phi \in \mathsf{Stt}(\Delta)$.*

Proof. By induction on the construction of $\Phi \in \mathsf{Stt}(\Delta)$. □

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

(a) For any trace σ, $\sigma \models \Phi, \Box\psi$ iff $\sigma \models \pi_i, \bigcirc\Box\psi$ holds for some $i \in J_k$ for each $1 \leq k \leq m$. Let λ be finite trace, $\lambda \models^{fin} \Phi \wedge \psi$ iff $\lambda \models^{fin} \pi_i$ 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 $\mathsf{Incnst}(\Delta)$ for all $\Delta \in \mathsf{Stt}(\Phi \cup \{\pi_i\})$, then every $\lambda_0 \in \mathsf{Val}_\Delta(\mathcal{X} \cup \mathcal{Y})$ is a witness of the violation of the safety specification $\Phi \wedge \Box\psi$.*
- *(ii) For any $\Delta \in \mathsf{Stt}(\Phi \cup \{\pi_i\})$ such that $\mathsf{Cnst}(\Delta)$, it holds that $\lambda_0 \models^{fin} \Phi \wedge \psi$ for every $\lambda_0 \in \mathsf{Val}_\Delta(\mathcal{X} \cup \mathcal{Y})$.*

Proof. Item (a) follows from Proposition 2. Item (b) follows from item (a) and Proposition 8. □

In the following two results, under certain conditions on $\mathsf{TNF}(\Phi \wedge \psi)$, we get some valuation $v \in \mathsf{Val}(\mathcal{X})$ such that $v + v' \not\models^{fin} \Phi \wedge \psi$ holds for all $v' \in \mathsf{Val}(\mathcal{Y})$.

Proposition 10. *Let Φ be any set of safety formulas. If $\mathsf{TNF}(\Phi \wedge \psi)$ is not an $\mathcal{X}$-covering, then there exists some $v \in \mathsf{Val}(\mathcal{X})$ such that for all $v' \in \mathsf{Val}(\mathcal{Y})$, $v + v' \not\models^{fin} \Phi \wedge \psi$*

Proof. By Definition 6, there is some $v \in \mathsf{Val}(\mathcal{X}) \setminus \mathsf{Val}_{\Phi \wedge \psi}(\mathcal{X})$. □

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

Proof. Suppose that for all $v \in \mathsf{Val}(\mathcal{X})$ there is some $v' \in \mathsf{Val}(\mathcal{Y})$ such that $v + v' \models^{fin} \Phi \wedge \psi$. Then, by Proposition 3(c), there exists some $1 \leq k \leq m$ such that the minimal $\mathcal{X}$-covering $J_k \subseteq I$. Therefore, by Proposition 3(b), for all $v \in \mathsf{Val}(\mathcal{X})$ there is some $j \in J_k$ and some $v' \in \mathsf{Val}(\mathcal{Y})$ such that $v + v' \models^{fin} \pi_j$. According to Proposition 8, for each $j \in J_k$ there exists some $\Delta \in \mathsf{Stt}(\pi_j)$ such that $\mathsf{Cnst}(\Delta)$, which contradicts the hypothesis. □

Next, we introduce the *next-state rule* that (roughly speaking) allows us to jump from one state to the next one. For that, we introduce some preliminary concepts and notation.

Definition 13. *A set of formulas Φ is elementary if it consists of a set of literals and one elementary strict-future formula.*

Definition 14. *For any set Φ of next-formulas, $\Phi^{\downarrow} = \{\beta \mid \bigcirc\beta \in \Phi\}$. Given an elementary strict-future formula $\eta = \ddot{\bigvee}_{i=1}^{n} \bigwedge_{j=1}^{m} \bigcirc\beta_{i,j}$, the formula $\eta^{\downarrow}$ is defined to be $\ddot{\bigvee}_{i=1}^{n} \bigwedge_{j=1}^{m} \beta_{i,j}$.*

Example 6. Consider the strict-future formula $\eta = \Diamond_{[1,2]} a \,\ddot{\vee}\, \Box_{[1,3]} b$. Then, $\eta^E = \bigcirc a \,\ddot{\vee}\, \bigcirc\Diamond_{[1,1]} a \,\ddot{\vee}\, (\bigcirc b \wedge \bigcirc\Box_{[1,2]} b)$ and $\eta^{\downarrow} = a \ddot{\vee} \Diamond_{[1,1]} a \ddot{\vee} (b \wedge \Box_{[1,2]} b)$.

$$(\bigcirc)\ \frac{\Phi, \eta, \bigcirc\Box\psi}{\eta^{\downarrow}, \Box\psi} \quad \text{if } \Phi \cup \{\eta\} \text{ is elementary and } \eta \text{ is strict-future.}$$

Fig. 3. Next-state Rule

The *Next-state Rule* (in Figure 3) is applied whenever the target set of formulas is elementary. Note that, if there is not an strict-future formula η, the children of the above rule $(\bigcirc)$ is just $\Box\psi$.

Proposition 12. *Let $\Phi \cup \{\eta\}$ be any consistent and elementary set of formulas with strict-future formula η. Then*

(a) For any trace σ, if $\sigma \models \Phi, \eta, \bigcirc\Box\psi$ then $\sigma^1 \models \eta^{\downarrow}, \Box\psi$.
(b) For any (non-empty) finite trace $\lambda = \lambda_0, \ldots \lambda_{k-1}$ that is a pre-witness of $\alpha \wedge \Box\psi$ such that $\lambda_{k-1} \models^{fin} \Phi$ and any $\lambda_k \in \mathsf{Val}(\mathcal{X} \cup \mathcal{Y})$, the finite trace $\lambda \cdot \lambda_k$ is a pre-witness of $\alpha \wedge \Box\psi$ if and only if $\lambda_k \models^{fin} \eta^{\downarrow} \wedge \psi$.

Proof. Both items follows from Definitions 11 and 14, by routine application of semantic definition of $\bigcirc$ operator. □

According to the previously introduced set of tableau rules, we can define the set of all formulas that could appear in the construction of a tableau for a given safety specification, which is known as the closure of the given specification.

Definition 15. *We denote by $\mathsf{SubFm}(\beta)$ the set of all subformulas of any given formula β. In particular, $\mathsf{SubFm}(\bigcirc^i\beta) = \{\bigcirc^j\beta \mid 0 \leq j \leq i\} \cup \mathsf{SubFm}(\beta)$. For a given safety formula ψ, we define $\mathsf{Varnt}(\psi)$ to be the union of the following four sets that collects all the variants of subformulas $\Diamond_I$ and $\Box_I$ that the tableau rules could introduce.*

$\{\Diamond_{[n,m']}\beta, \bigcirc\Diamond_{[n,m']}\beta \mid \Diamond_{[n,m]}\beta \in \mathsf{SubFm}(\psi), n \leq m' < m\}$
$\{\Box_{[n,m']}\beta, \bigcirc\Box_{[n,m']}\beta \mid \Box_{[n,m]}\beta \in \mathsf{SubFm}(\psi), n \leq m' < m\}$
$\{\mathsf{SubFm}(\bigcirc^i\beta) \mid \Diamond_{[n,m]}\beta \in \mathsf{SubFm}(\psi), 0 \leq i \leq n\}$
$\{\mathsf{SubFm}(\bigcirc^i\beta) \mid \Box_{[n,m]}\beta \in \mathsf{SubFm}(\psi), 0 \leq i \leq n\}$

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

4.3 A Tableau Algorithm for Realizability

In this section we present a non-deterministic algorithm (see Alg. 1) for deciding whether a given safety specification is realizable. Alg. 1 constructs a completed tableau that analyzes the minimal $\mathcal{X}$-coverings produced by the moves (and allowed by the input safety specification) at the successive states of the game. Alg. 1 uses recursion to explore in-depth the branches of the tree. The formal parameter of Alg. 1 is given as the union of a set of formulas Φ and a formula χ that ranges in $\{\Box\psi, \bigcirc\Box\psi\}$. For deciding realizability of a safety specification $\varphi = \alpha \wedge \Box\psi$, the initial call $\mathsf{Tab}(\varphi)$ is really $\mathsf{Tab}(\{\alpha\} \cup \{\Box\psi\})$. Intuitively, Eve moves when $\chi = \Box\psi$ (at the start), whereas Sally moves when $\chi = \bigcirc\Box\psi$.

Definition 16. *A branch* b *of a tableau is any finite sequence of nodes* $n_0, \ldots, n_k$ *such that* n_0 *is the root and* $(n_i, n_{i+1}) \in R$ *for all* $0 \leq i < k-1$*. If* n_k *is a successful leaf, we say that* b *is a successful branch. If* n_k *is a failure leaf, we say that* b *is a failure branch.*

For the sake of clarity, we omit in Alg. 1 the details for loading the current branch B and for performing subsumption in nodes. The result of Alg. 1 is returned in the boolean variable *is_ open*. Lines 1-5 deal with the two types of terminal nodes to which no rule is applied,no recursive call is produced, and returns success (*is_ open* := *True*) or failure (*is_ open* := *False*). Line 7 produces a recursive call that immediately returns failure. The value returned in *is_ open* corresponds to whether the completed tableau for the call parameter $\Phi \cup \{\chi\}$ is open or not (i.e is closed). Recursive calls in Alg. 1 and the notions of open and closed tableau, are related with AND-nodes, for which we introduce the following definition. For that we next introduce the notion of bunch.

Definition 17. *Given a set of branches* H *of a completed tableau, we say that* H *is a* bunch *if and only if for every* $b \in H$ *and every AND-node* $n \in b$*, and every* n' *that is an* $(\Box\&)$*-successor of* n*, there is* $b' \in H$ *such that* $n' \in b'$*. A completed tableau is* open *if and only if it contains at least one* bunch *such that all its branches are successful. Otherwise, when all possible bunches of a completed tableau contains a failure branch, the tableau is* closed.

Definition 18. *A tableau is* completed *when all its branches contains a terminal node, i.e. all its branches are failure or successful.*

Algorithm 1: $\mathsf{Tab}(\Phi \cup \{\chi\})$ returns *is_ open*: Boolean

```
1  if Φ is inconsistent then
2  |  is_ open := False
3  else if χ = □ψ then
4  |  if Φ0 ◁ Φ for some Φ0 in the branch of Φ then
5  |  |  is_ open := True
6  |  else if TNF(Φ ∧ ψ) is not an X-covering then
7  |  |  is_ open := Tab({F, □ψ});
8  |  else if TNF(Φ ∧ ψ) is a non-minimal X-covering then
9  |  |  Let J1, ..., Jm be all the minimal X-coverings of TNF(Φ ∧ ψ);
10 |  |  i := 0; is_ open := False ;
11 |  |  while ¬is_ open ∧ i < m do
12 |  |  |  i := i + 1 ;
13 |  |  |  is_ open := Tab(Ji ∪ {□ψ});
14 |  |  end
15 |  else                                  // TNF(Φ ∧ ψ) = ⋁_{i=1}^{n} πi is a minimal X-covering
16 |  |  i := 0; is_ open := True ;
17 |  |  while is_ open ∧ i < n do
18 |  |  |  i := i + 1 ;
19 |  |  |  is_ open := Tab({πi, ○□ψ});
20 |  |  end
21 |  end
22 else if Φ = Λ ∪ {η} is elementary (η is strict-future) then
23 |  is_ open := Tab({η↓, □ψ});
24 else
25 |  ρ := select_ saturation_ rule(Φ);
26 |  Let 1 ≤ k ≤ 2 and Φ1, ..., Φk the set of all ρ-children;
27 |  is_ open := Tab(Φ1 ∪ {○□ψ});
28 |  if k = 2 ∧ ¬is_ open then is_ open := Tab(Φ2 ∪ {○□ψ});
29 end
```

Alg. 1 looks for bunches of successful branches as follows. Lines 8-14 of Alg. 1 invoke a recursive call for each minimal $\mathcal{X}$-covering, according to rule ($\Box\|$). When some of these calls return *is_ open* for a minimal $\mathcal{X}$-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 rule ($\Box\&$) and according to lines 15-20, 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 22-23 perform the application of ($\bigcirc$), and lines 25-28 apply the saturation rules, when the applied rule split in two children the second is expanded only if the first one returns that *is_ open* is false.

Proposition 13. *Alg. 1 terminates and* $\mathsf{Tab}(\varphi)$ *is a completed tableau.*

Proof. It is easy to see that any node in $\mathsf{Tab}(\varphi)$ is labelled by a finite subset of $\mathsf{Clo}(\varphi)$. Therefore, supposing that there exists an infinite branch, we could get the

infinite sequence of its node labels, namely $\Phi_0, \Phi_1, \ldots$. Then, every $\Phi_i \in 2^{\mathsf{Clo}(\varphi)}$ and for every $0 \leq i < j$ it does not hold that $\Phi_i \lessdot \Phi_j$. In particular, all them must be pairwise different. Hence, we have infinitely many subsets of $\mathsf{Clo}(\varphi)$ that are pairwise different, which contradicts the finiteness of $\mathsf{Clo}(\phi)$. Therefore, any branch of $\mathsf{Tab}(\varphi)$ is finite. Since the number of branches of $\mathsf{Tab}(\varphi)$ is finite (by construction), termination is ensured. Lines 1-5 ensures that $\mathsf{Tab}(\varphi)$ is a completed tableau. □

Note that our algorithm assumes a procedure that given a formula calculates its TNF. For the implementation, we plan to use BICA ([18]), a Boolean simplifier for non-clausal formulas that computes a minimal (size) prime cover of the input formula.

5 Examples

In this section, we present some representative examples that illustrate how our tableau method works.

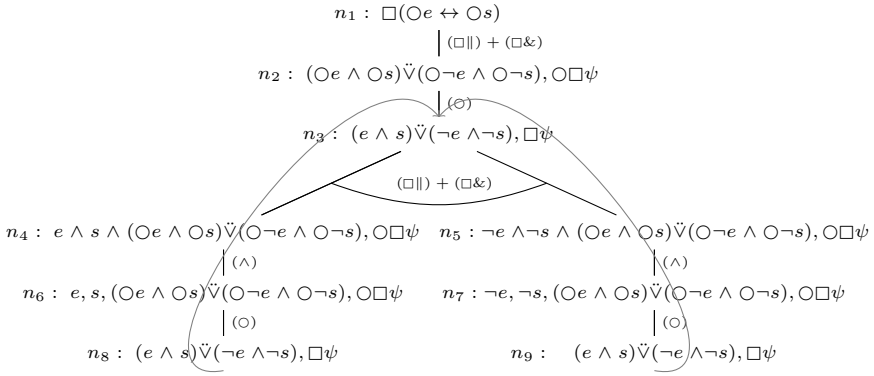

Fig. 4. A tableau that proves the realizability of $\Box(\bigcirc e \leftrightarrow \bigcirc s)$.

Example 7. We revisit the specification $\Box(\bigcirc e \leftrightarrow \bigcirc s)$ discussed in Section 1, for which $\mathsf{TNF}(\bigcirc e \leftrightarrow \bigcirc s) = (\bigcirc p_e \wedge \bigcirc s) \vee (\bigcirc\neg p_e \wedge \bigcirc\neg s)$, which is a minimal $\mathcal{X}$-covering with just one move without literals. Hence, the only child of the root in Figure 4 is obtained by rule $(\Box\|)$ and then $(\Box\&)$. According to Alg. 1, in the node n_3, $\mathsf{TNF}((c \wedge s) \vee (\neg c \wedge \neg s)) \wedge \psi)$ yields a minimal $\mathcal{X}$-covering with two moves, hence the rule $(\Box\|)$ is applied and, after it, the rule $(\Box\&)$ yields two AND-nodes, one for each move. The first move contains $e \wedge s$ and the second one contains $\neg e \wedge \neg s$. 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 that shows that the input is realizable.

Let us remark that, in practice, to decide realizability, it suffices to construct just one of the two branches in the tableau of Figure 4. Since the set of strict-future formulas in the label of nodes n_4 and n_5 are exactly the same, their expansion after the application of the next-state rule, in the example nodes n_8 and n_9, are also identical.

Example 8. Let ψ be the second safety formula η of Example 1 where $\mathcal{X} = \{p_e\}$. In Figure 5 we depict the tableau for $\Box\psi$. The $\mathsf{TNF}(\psi)$ (see $\mathsf{TNF}(\eta)$ in Example 1) is a minimal $\mathcal{X}$-covering hence, by applying rule $(\Box\|)$ and then $(\Box\&)$, nodes n_2 and n_3 are generated. According to Alg. 1, they are AND-nodes, one for each move in $\mathsf{TNF}(\psi)$. The label of node n_4 is an elementary formula and the $(\bigcirc)$-rule yields node n_5. Since it is not true that $\tau(n_5) \lessdot \tau(n_1)$, the tableau goes on applying rules $(\Box\|)$ and $(\Box\&)$ to node n_5. Then,

$$\mathsf{TNF}(\Box_{[0,8]}c \wedge \psi) = (p_e \wedge c \wedge \bigcirc\Box_{[0,8]}c) \vee (\neg p_e \wedge c \wedge \bigcirc\Box_{[0,8]}c)$$

is a minimal $\mathcal{X}$-covering with two moves corresponding to nodes n_6 and n_7. As both nodes contain an elementary formula, by $(\bigcirc)$-rule, nodes n_8 and n_9 are obtained. Finally, relations $\tau(n_8) \lessdot \tau(n_5)$ and $\tau(n_9) \lessdot \tau(n_5)$ hold and both branches are successful.

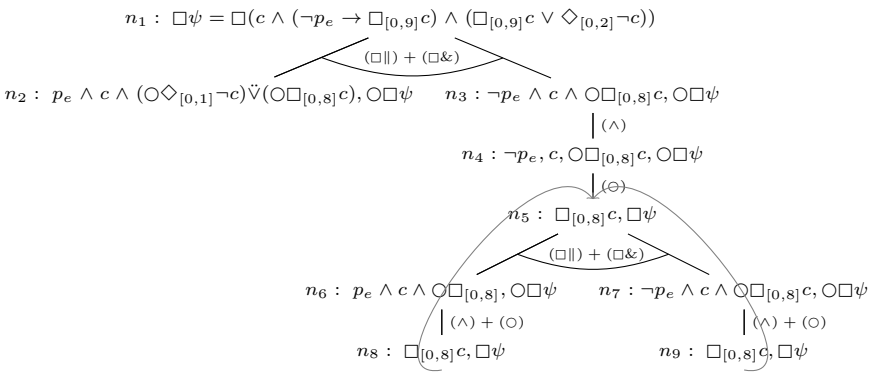

Fig. 5. Tableau for $\Box\psi$ where $\psi = c \wedge (\neg p_e \rightarrow \Box_{[0,9]}c) \wedge (\Box_{[0,9]}c \vee \Diamond_{[0,2]}\neg c)$.

In previous Example 7 we noticed that one of the two branches would have not been expanded to decide that the input is realizable because the identical strict-future formulas in the label of two AND-siblings. In Figure 5 we do not depict the expansion of node n_2, because its AND-sibling n_3 contains a stronger strict-future formula. Note that $\bigcirc\Box_{[0,8]}c$ is stronger than $(\bigcirc\Diamond_{[0,1]}\neg c\ddot{\vee}\bigcirc\Box_{[0,8]}c)$. Then, if the expansion of node n_3 gets an open tableau, that should be also the case for node n_2. Moreover, when the strongest AND-child of a node, n_3 in this case, has a closed tableau, the tableau for the parent node is closed independently of the tableau for the remaining AND-children. It can be also observed that in nodes n_6 and n_7 we have the same situation as in Example 7 both have identical strict-future formulas.

Example 9. Consider the safety specification

$$\Box\psi = \Box(a \to c, \bigcirc p_e \to \Diamond_{[1,2]} a, \bigcirc\neg p_e \to \Diamond_{[1,10]}\neg c).$$

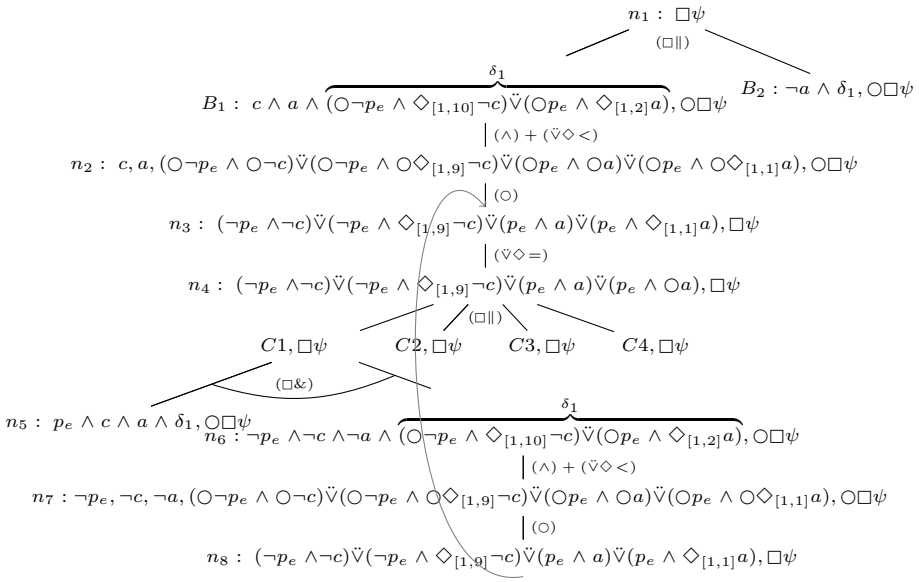

Fig. 6. Tableau for $\Box(a \to c, \bigcirc p_e \to \Diamond_{[1,2]} a, \bigcirc\neg p_e \to \Diamond_{[1,10]}\neg c)$

The tableau in Figure 6 starts with $\mathsf{TNF}(\psi) = (c \wedge a \wedge \delta_1) \vee (\neg a \wedge \delta_1)$ where $\delta_1 = (\bigcirc\neg p_e \wedge \Diamond_{[1,10]}\neg c)\ddot{\vee}(\bigcirc p_e \wedge \Diamond_{[1,2]} a)$. There are two minimal $\mathcal{X}$-covering B_1 and B_2 with one move each. The tableau chooses to expand the covering B_1. The application of various saturation rules from B_1 on, in particular $(\ddot{\vee}\Diamond <)$ and $(\ddot{\vee}\Diamond =)$, leads to node n_4. Then, $(\Box\|)$ is applied to n_4 resulting in four OR-siblings corresponding to the four minimal $\mathcal{X}$-coverings in $\mathsf{TNF}(((\neg p_e \wedge\neg c) \vee (\neg p_e \wedge \Diamond_{[1,9]}\neg c) \vee (p_e \wedge a) \vee (p_e \wedge \bigcirc a)) \wedge \psi)$ which are:

$$\begin{aligned}
C1 &= (p_e \wedge c \wedge a \wedge \delta_1) \vee (\neg p_e \wedge\neg c \wedge\neg a \wedge \delta_1)\\
C2 &= (p_e \wedge c \wedge a \wedge \delta_1) \vee (\neg p_e \wedge c \wedge \delta_3)\\
C3 &= (p_e \wedge\neg a \wedge \delta_2) \vee (\neg p_e \wedge\neg c \wedge\neg a \wedge \delta_1)\\
C4 &= (p_e \wedge\neg a \wedge \delta_2) \vee (\neg p_e \wedge c \wedge \delta_3)
\end{aligned}$$

where

$$\begin{aligned}
\delta_1 &= (\bigcirc\neg p_e \wedge \Diamond_{[1,10]}\neg c) \vee (\bigcirc p_e \wedge \Diamond_{[1,2]} a)\\
\delta_2 &= (\bigcirc\neg p_e \wedge \Diamond_{[1,10]}\neg c \wedge \bigcirc a)\ddot{\vee}(\bigcirc p_e \wedge \bigcirc a)\\
\delta_3 &= (\bigcirc\neg p_e \wedge \Diamond_{[1,9]}\neg c)\ddot{\vee}(\bigcirc p_e \wedge \Diamond_{[1,2]} a \wedge \Diamond_{[1,9]}\neg c)
\end{aligned}$$

In this case, δ_1 is weaker than both δ_2 and δ_3. We expand only the minimal $\mathcal{X}$-covering C_1 that contains δ_1 in its two moves. This creates two successful branches. Note that the branch starting at node n_5 (that we do not depicted) would be also successful. Indeed it would be finished by a node identical to n_8. Hence, the completed tableau is open.

In node n_4, we choose to expand the node of the minimal $\mathcal{X}$-covering $C1$ with the weakest strict-future formulas. This is really convenient because if the tableau is closed for this option, we immediately know that the tableau will also be closed for any of the other minimal $\mathcal{X}$-covering. Moreover, if the expansion of C_1 leads to an open tableau, then n_4 (whose children are OR-siblings) has an open tableau and there is no need to expand the remaining OR-siblings corresponding to $\mathcal{X}$-coverings C_2, C_3 and C_4.

It is important to highlight that it is possible to synthesise a winning strategy from any open tableau. For instance, Figure 7 represents a winning strategy by means of a finite state machine extracted from the tableau in Figure 6.

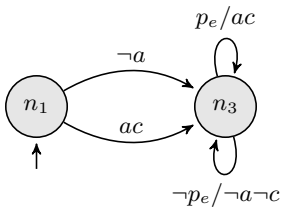

Fig. 7. Finite state machine that represents the strategy synthesised from the open tableau in Figure 6

Example 10. Let $a \wedge \Box\psi$ be a safety specification where

$$\psi = (a \rightarrow c) \wedge (p_e \rightarrow \Diamond_{[0,100]}\neg c) \wedge (\neg p_e \rightarrow \Diamond_{[0,100]} a).$$

Figure 8 shows the open completed tableau for $a \wedge \Box\psi$. Nodes n_2 and n_3 comes from

$$\mathsf{TNF}(a \wedge \psi) = (p_e \wedge a \wedge c \wedge \bigcirc\Diamond_{[0,99]}\neg c) \vee (\neg p_e \wedge a \wedge c)$$

which is a minimal $\mathcal{X}$-covering. In node n_4, the $\mathsf{TNF}(\Diamond_{[0,99]}\neg c \wedge \psi)$ is

$$(p_e \wedge c \wedge \bigcirc\Diamond_{[0,98]}\neg c) \vee (p_e \wedge \neg a \wedge \neg c) \vee (p_e \wedge \neg a \wedge c \wedge \bigcirc\Diamond_{[0,98]}\neg c) \vee$$
$$(\neg p_e \wedge c \wedge a \wedge \bigcirc\Diamond_{[0,98]}\neg c) \vee (\neg p_e \wedge c \wedge \neg a \wedge \bigcirc\Diamond_{[0,99]} a \wedge \bigcirc\Diamond_{[0,98]}\neg c) \vee$$
$$(\neg p_e \wedge \neg a \wedge \neg c \wedge \bigcirc\Diamond_{[0,99]} a)$$

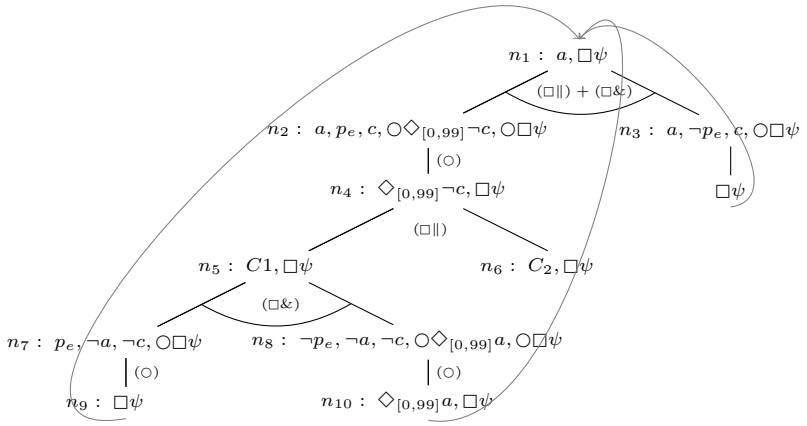

Fig. 8. Tableau for $a \wedge \Box((a \rightarrow c) \wedge (p_e \rightarrow \Diamond_{[0,100]}\neg c) \wedge (\neg p_e \rightarrow \Diamond_{[0,100]}a))$.

The weakest option for the environment variable p_e is $(p_e \wedge \neg a \wedge \neg c)$ because its strict-future formula is $True$. For the environment variable $\neg p_e$ the weakest options are two: $(\neg p_e \wedge \neg a \wedge \neg c \wedge \bigcirc\Diamond_{[0,99]}a)$ or $(\neg p_e \wedge c \wedge a \wedge \bigcirc\Diamond_{[0,98]}\neg c)$. So, from node n_4 the tableau goes on with two possible minimal $\mathcal{X}$-coverings.

$$C1 = (p_e \wedge \neg a \wedge \neg c) \vee (\neg p_e \wedge \neg a \wedge \neg c \wedge \bigcirc\Diamond_{[0,99]}a)$$
$$C2 = (p_e \wedge \neg a \wedge \neg c) \vee (\neg p_e \wedge c \wedge a \wedge \bigcirc\Diamond_{[0,98]}\neg c)$$

As explained in Example 9, for deciding realizability, it suffices to expand this two $\mathcal{X}$-coverings. However, since $\mathcal{X}$-covering C_1 gives an open tableau, we do not have to expand C_2.

It is worthy to note that formulas $\Diamond_{[0,100]}\neg c$ and $\Diamond_{[0,100]}a$ are fulfilled in at most two steps in Figure 8. In general, our method does not force the complete unfolding of eventualities, nor formulas of the form $\Box_I\eta$, unless its fulfillment check requires it.

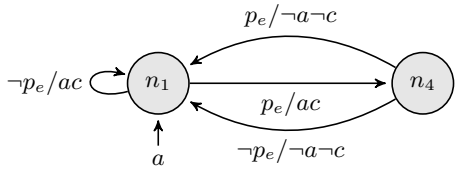

Fig. 9. Finite state machine that represents the strategy synthesised from the open tableau in Figure 8

From the open tableau, we can easily synthesise a winning strategy, which is the one represented by the finite state machine of Figure 9.

Example 11. Let $a \wedge \Box\psi$ be a safety specification where $\psi = (a \rightarrow c) \wedge (p_e \rightarrow \bigcirc a) \wedge (\neg p_e \rightarrow \Box_{[2,10]}\neg c)$. Figure 10 is a closed tableau that proves that $a \wedge \Box\psi$ is unrealizable. To start the tableau construction, we have that $\mathsf{TNF}(a \wedge \psi) = (p_e \wedge a \wedge c \wedge \bigcirc a) \vee (\neg p_e \wedge a \wedge c \wedge \Box_{[2,10]}\neg c)$.

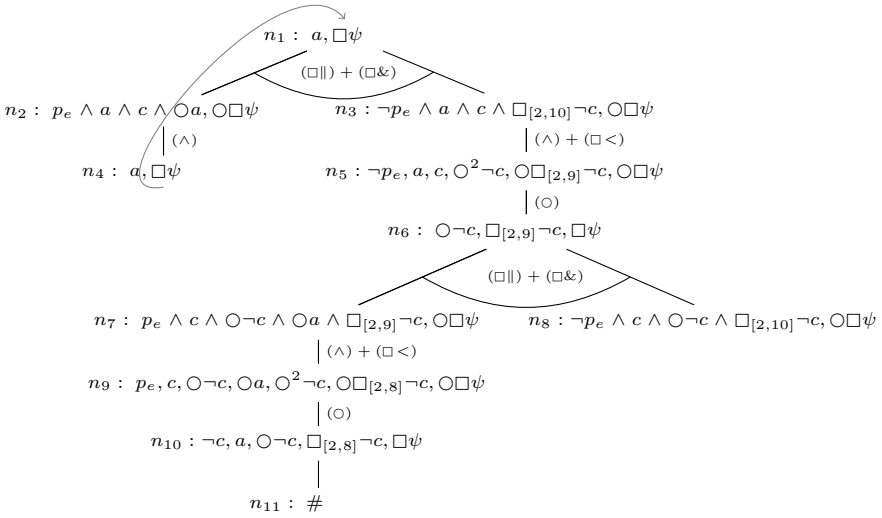

Fig. 10. Closed tableau for $a \wedge \Box((a \rightarrow c) \wedge (p_e \rightarrow \bigcirc a) \wedge (\neg p_e \rightarrow \Box_{[2,10]}\neg c))$.

The realizability result depends on the success of the AND-nodes n_2 and n_3. Once the success of node n_2 is ensured, the tableau goes on with the expansion of node n_3. At node n_6, we have that $\mathsf{TNF}(\bigcirc\neg c \wedge \Box_{[2,9]}\neg c \wedge \Box\psi) =$

$$(p_e \wedge c \wedge \bigcirc\neg c \wedge \bigcirc a \wedge \Box_{[2,9]}\neg c) \vee (p_e \wedge\neg a \wedge \bigcirc\neg c \wedge \bigcirc a \wedge \Box_{[2,9]}\neg c) \vee$$
$$(\neg p_e \wedge c \wedge \bigcirc\neg c \wedge \Box_{[2,10]}\neg c) \vee (\neg p_e \wedge\neg a \wedge \bigcirc\neg c \wedge \Box_{[2,10]}\neg c)$$

There are 4 possible minimal $\mathcal{X}$-coverings, but it is enough to choose any of them to decide that the tableau is closed. The reason is that they have the same strict-future formula. Therefore, the tableau goes on with the AND-nodes n_7 and n_8, which correspond to the following minimal $\mathcal{X}$-covering.

$$(p_e \wedge c \wedge \bigcirc\neg c \wedge \bigcirc a \wedge \bigcirc^2\neg c \wedge \bigcirc\Box_{[2,8]}\neg c) \vee$$
$$(\neg p_e \wedge c \wedge \bigcirc\neg c \wedge \bigcirc^2\neg c \wedge \bigcirc\Box_{[2,9]}\neg c)$$

As the TNF at node n_{10} is $False$, this node is a failure leaf. This fact completes the tableau, since n_7 and n_8 are AND-siblings.

6 Correctness

In this section, we connect properties of the completed tableau $\mathsf{Tab}(\varphi)$ with the existence of winning strategies for Sally or Eve in the safety formula game $\mathcal{T}(\varphi)$. We will show that a closed tableau $\mathsf{Tab}(\varphi)$ represents a wining strategy for Eve. However, an open tableau $\mathsf{Tab}(\varphi)$ represents a wining strategy for Sally which ensures (or proves) that the safety specification in the root is realizable. Indeed, the tableau construction could construct that winning strategy and returns it as output for the user. To carry out this proof, we first introduce tableau games and connect them with safety games (and therefore with realizability). Then we will connect complete tableau with the winning strategy of the corresponding player in the tableau game.

6.1 Safety Tableau Games

In this subsection we define a class of games $\mathcal{T}(\varphi)$ where positions are labeled by sets of formulas from the closure $\mathsf{Clo}(\varphi)$ of the given specification $\varphi = \alpha \wedge \Box\psi$. First, we define the components of the game $\mathcal{T}(\varphi)$. The set $P = P_E \cup P_S$ of positions is formed by subsets of the closure of the specification φ where

$$P_E = \{\Phi \cup \{\Box\psi\} \mid \Phi \in \mathsf{Preclo}(\varphi) \text{ and } \mathsf{Cnst}(\Phi)\}$$
$$P_S = \{\Phi \cup \{\bigcirc\Box\psi\} \mid \Phi \in \mathsf{Preclo}(\varphi) \text{ and } \mathsf{Cnst}(\Phi)\}.$$

The set of bad positions $B \subseteq P_E$ is defined as follows: $\Phi \cup \{\Box\psi\} \in P_E$ is in B whenever one of the following conditions holds

(i) $\mathsf{TNF}(\Phi \wedge \psi)$ is not an $\mathcal{X}$-covering.
(ii) Let $J_1, \ldots, J_m$ the collection of all minimal $\mathcal{X}$-coverings in $\mathsf{TNF}(\Phi \wedge \psi)$. For every $1 \leq k \leq m$ there exists some $i \in J_k$ such that $\mathsf{Incnst}(\Delta)$ for all $\Delta \in \mathsf{Stt}(\pi_i)$.

Next, we define the moves of the arena. In every position $p = \Phi \cup \{\Box\psi\} \in P_E$ (in particular, in the initial one where $\Phi = \alpha$) Eve, if $p \notin B$, chooses one of the minimal $\mathcal{X}$-coverings $J \subseteq I$ in $\mathsf{TNF}(\Phi \wedge \psi) = \bigvee_{i \in I} \pi_i$ and then choose some $j \in J$. The resulting position $\{\pi_j, \bigcirc\Box\psi\}$ is in P_S, therefore it is the Sally's turn. Then, Sally, in each of her turns, annotates the state with an exhaustive application of choices for the system, where each choice is a saturated set. If all possible choices for Sally (according to the specification) are inconsistent, then the position is marked as bad, and otherwise, the position is good. Then, Sally must get the next-state formulas from the previous position of Sally to pass the turn to Eve. Formally,

- $(\Phi \cup \{\Box\psi\}, \{\pi, \bigcirc\Box\psi\}) \in T_E$ if π is a move in $\mathsf{TNF}(\Phi \wedge \psi)$.
- $(\{\pi, \bigcirc\Box\psi\}, \Delta^{\downarrow} \cup \{\Box\psi\}) \in T_S$ if $\Delta \in \mathsf{Stt}(\pi)$

Note that the description above defines a finite state safety game, which we call the tableau game $\mathcal{T}(\varphi)$ for specification φ. We will then prove the following fact.

Lemma 3. *$\mathcal{G}(\varphi)$ is winning for Sally if and only if $\mathcal{T}(\varphi)$ is winning for Sally.*

Proof. We use $\mathcal{T}$ and $\mathcal{G}$ as superscripts to identify the components of the two games. We first associate to each position $\Phi \in P^{\mathcal{T}}$ a set of positions $\mathsf{Pos}(\Phi) \subseteq P^{\mathcal{G}}$. For simplicity, in the argument of Pos, we will omit the formula $\Box\psi$ in every position in $P_E^{\mathcal{T}}$ and also $\bigcirc\Box\psi$ in every position in $P_S^{\mathcal{T}}$, we just refer the set of the remaining formulas in each position.

The set of initial states $I^{\mathcal{T}} = \{\{\alpha\}\}$ and $\mathsf{Pos}(\alpha) = \{(\epsilon, \epsilon)\}$.

For any given $\Phi \in P_E^{\mathcal{T}}$ we define the set of positions of its T_S and T_E-related positions as follows. For any move π in $\mathsf{TNF}(\Phi \wedge \psi)$ (note that $\{\pi, \bigcirc\Box\psi\} \in P_S^{\mathcal{T}}$ and $(\Phi \cup \{\Box\psi\}, \{\pi, \bigcirc\Box\psi\}) \in T_E$), we define

$$\mathsf{Pos}(\pi) = \{(\overline{x} \cdot v, \overline{y}) \in P_S^{\mathcal{G}} \mid (\overline{x}, \overline{y}) \in \mathsf{Pos}(\Phi) \text{ and } v \in \mathsf{Val}_\pi(\mathcal{X})\}$$

and for any $\Delta \in \mathsf{Stt}(\pi)$ such that $\mathsf{Cnst}(\Delta)$ (note that $\Delta^{\downarrow} \cup \{\Box\psi\} \in P_E^{\mathcal{T}}$ and $(\{\pi, \bigcirc\Box\psi\}, \Delta^{\downarrow} \cup \{\Box\psi\}) \in T_S$), we define

$$\mathsf{Pos}(\Delta^{\downarrow}) = \{(\overline{x} \cdot v, \overline{y} \cdot v') \in P_E^{\mathcal{G}} \mid (\overline{x}, \overline{y}) \in \mathsf{Pos}(\Phi), v \in \mathsf{Val}_\Delta(\mathcal{X}) \text{ and } v' \in \mathsf{Val}_\Delta(\mathcal{Y})\}.$$

For any $\Phi \in P_E^{\mathcal{T}}$, we say that $(\overline{x}, \overline{y}) \in \mathsf{Pos}(\Phi)$ is a *play*, whenever $|\overline{x}| = |\overline{y}| \geq 1$ and $(x_1, y_1) \in \mathsf{Pos}(\{\alpha\})$. Moreover, by Proposition 12(b), for any play $(\overline{x}, \overline{y}) \in \mathsf{Pos}(\Phi)$ of length k:

$$\overline{x} + \overline{y} \models^{\mathit{fin}} \alpha \wedge \Box\psi \text{ if and only if } x_k + y_k \models^{\mathit{fin}} \Phi \wedge \psi \tag{5}$$

Next, we prove that for all $\Phi \in P_E^{\mathcal{T}}$ and all play $(\overline{x}, \overline{y}) \in \mathsf{Pos}(\Phi)$: $\Phi \in B^{\mathcal{T}}$ if and only if $(\overline{x}, \overline{y}) \in B^{\mathcal{G}}$.

First, consider any $\Phi \in B^{\mathcal{T}}$ and any play $(\overline{x}, \overline{y}) \in \mathsf{Pos}(\Phi)$. If (i) $\mathsf{TNF}(\Phi \wedge \psi)$ is not an $\mathcal{X}$-covering then, by Proposition 10, there exists some $v \in \mathsf{Val}(\mathcal{X})$ such that for all $v' \in \mathsf{Val}(\mathcal{Y})$, $v + v' \not\models^{\mathit{fin}} \Phi \wedge \psi$. Otherwise, if (ii) for every minimal $\mathcal{X}$-covering J in $\mathsf{TNF}(\Phi \wedge \psi)$, there exists some $i \in J$ such that $\mathsf{Incnst}(\Delta)$ for all $\Delta \in \mathsf{Stt}(\pi_i)$. Then, by Proposition 11, there exists some $v \in \mathsf{Val}(\mathcal{X})$ such that for all $v' \in \mathsf{Val}(\mathcal{Y})$, $v + v' \not\models^{\mathit{fin}} \Phi \wedge \psi$. Therefore, by (5), $\overline{x} \cdot v + \overline{y} \cdot v' \not\models^{\mathit{fin}} \alpha \wedge \Box\psi$.

Second, consider any $\Phi \notin B^{\mathcal{T}}$ and any play $(\overline{x}, \overline{y}) \in \mathsf{Pos}(\Phi)$. Then $\mathsf{TNF}(\Phi \wedge \psi)$ is an $\mathcal{X}$-covering and there exists at least one minimal $\mathcal{X}$-covering $\bigvee_{j \in J} \pi_j$ in $\mathsf{TNF}(\Phi \wedge \psi)$ such that for all $j \in J$ there exists some $\Delta_j \in \mathsf{Stt}(\pi_j)$ such that $\mathsf{Cnst}(\Delta_j)$. According to Proposition 8, for each $j \in J$ there exists $\lambda_j \in \mathsf{Val}_{\Delta_j}(\mathcal{X} \cup \mathcal{Y})$ such that $\lambda_j \models^{\mathit{fin}} \pi_j$ (hence $\lambda_j \models^{\mathit{fin}} \Phi \wedge \psi$). Since J is an $\mathcal{X}$-covering, for all $v \in \mathsf{Val}(\mathcal{X})$ there exists $v' \in \mathsf{Val}(\mathcal{Y})$ such that $v + v' \models^{\mathit{fin}} \Phi \wedge \psi$. Therefore, since $(\overline{x}, \overline{y})$ is a play, by (5), we have that $\overline{x} \cdot v + \overline{y} \cdot v' \models^{\mathit{fin}} \alpha \wedge \psi$ Therefore, $(\overline{x}, \overline{y}) \notin B^{\mathcal{G}}$.

Consequently, from any winning strategy for Sally in $\mathcal{G}(\varphi)$ we can construct a winning strategy for Sally in $\mathcal{T}(\varphi)$ and vice versa. □

Since $\mathcal{G}(\varphi)$ is winning for Sally if and only if φ is realizable, the following holds.

Corollary 2. *A safety specification φ is realizable if and only if $\mathcal{T}(\varphi)$ is winning for Sally.*

6.2 Soundness and Completeness

Let us first recall that for every AND-node n that occurs in a branch of a bunch H and every children n' of n, there is a branch in H that includes n'. On the other hand for OR-nodes there is a possible bunch for each children. Consequently, when a tableau $\mathsf{Tab}(\Phi \cup \{\Box\psi\})$ is closed (i.e. any possible bunch contains a failure branch) at least one child of every AND-node produces a closed tableau.

Proposition 14. *Let Φ be any consistent set of safety formulas such that $\mathsf{TNF}(\Phi \wedge \psi)$ is an $\mathcal{X}$-covering. If $\mathsf{Tab}(\Phi \cup \{\Box\psi\})$ is closed, then for any AND-node $\{\bigvee_{i\in I} \pi_i, \Box\psi\}$ (where I is a minimal $\mathcal{X}$-covering) there exists at least one child $i \in I$ such that $\mathsf{Tab}(\{\pi_i, \bigcirc\Box\psi\})$ is also closed.*

Proof. Suppose that there exists an AND-node $\{\bigvee_{i\in I} \pi_i, \Box\psi\}$ such that every $\mathsf{Tab}(\{\pi_i, \bigcirc\Box\psi\})$ is open for all $i \in I$. Then, each $\mathsf{Tab}(\{\pi_i, \bigcirc\Box\psi\})$ has a bunch of successful branches, hence $\mathsf{Tab}(\Phi\cup\{\Box\psi\})$ has also a bunch of successful branches, which contradicts our hypothesis. □

Theorem 1. *For any given safety specification φ, the completed tableau $\mathsf{Tab}(\varphi)$ is closed if and only if the game $\mathcal{T}(\varphi)$ is winning for Eve.*

Proof. To prove the left-to-right implication of the theorem, suppose that $\mathsf{Tab}(\varphi)$ is closed. Therefore, any bunch of $\mathsf{Tab}(\varphi)$ contains at least one failure branch.

We define a winning strategy for Eve $\rho : P_E \to P_S$ in the game $\mathcal{T}(\varphi)$. For each position $\Phi \cup \{\Box\psi\} \in P_E$, if $\mathsf{TNF}(\Phi \wedge \psi)$ is not and $\mathcal{X}$-covering, then $\Phi \cup \{\Box\psi\} \in B^{\mathcal{T}}$ and then Eve is winning. Otherwise, by Proposition 14, for any minimal $\mathcal{X}$-covering there must exists some move π in $\mathsf{TNF}(\Phi \wedge \psi)$ such that $\mathsf{Tab}(\{\pi, \bigcirc\Box\psi\})$ is also closed. Hence, we select an arbitrary minimal $\mathcal{X}$-covering and one such move π and define $\rho_E(\Phi \cup \{\Box\psi\}) = \{\pi, \bigcirc\Box\psi\}$. Then, for every $\Delta \in \mathsf{Stt}(\pi)$, either $\Delta^{\downarrow} \cup \{\Box\psi\} \in B^{\mathcal{T}}$ or $\Delta^{\downarrow} \cup \{\Box\psi\} \in P_E$ and every $\mathsf{Tab}(\Delta^{\downarrow} \cup \{\Box\psi\})$ is closed. Therefore, any bunch of all these sub-tableaux has also a failure branch.

In order to prove that ρ_E is winning for Eve, it is enough to show that for every initial play δ played according to ρ_E is winning for Eve. It is easy to see that every initial play $\delta = \delta(0), \delta(1), \ldots$ such that $\delta(i+1) = \rho_E(\delta(i))$ for all $\delta(i) \in P_E$ corresponds to a failure branch. Indeed, $\delta(0) = \{\alpha, \Box\psi\}$ is in $B^{\mathcal{T}}$ or in P_E, and $\mathsf{Tab}(\delta(i))$ is closed for all $i \geq 0$. Then, every initial play δ played according to ρ_E is finite and $\delta = \delta(0), \delta(1), \ldots, \delta(k)$ for some $k \geq 0$ such that $\delta(k) \in B^{\mathcal{T}}$.

Conversely, suppose that $\mathsf{Tab}(\varphi)$ is open, then there exists at least one bunch H such that all the leaves in H are successful terminal nodes in $\mathsf{Tab}(\{\alpha, \Box\psi\})$. Note that, for every AND-node $\{\bigvee_{i\in I} \pi_i, \Box\psi\} \in H$ and every $i \in I$, the node $\{\pi_i, \bigcirc\Box\psi\} \in H$ and $\mathsf{Tab}(\{\pi_i, \bigcirc\Box\psi\})$ is open. Moreover, every branch of H is ended by a node labelled by a Σ such that $\Sigma_0 \lessdot \Sigma$ for some $\Sigma_0 \in H$. Therefore, for each $\{\pi_i, \bigcirc\Box\psi\} \in H$ there exists at least one $\Delta \in \mathsf{Stt}(\pi_i)$ such that $\mathsf{Cnst}(\Delta)$ and one of the following two cases holds:

- $\Delta^{\downarrow} \cup \{\Box\psi\} \in H$, or

– $\Phi_0 \lessdot \Phi$ for some $\Phi_0 \in H$ and some $\Phi \subseteq \Delta$. In this case, there also exists some $\Delta_0 \in \mathsf{Stt}(\Phi_0)$ such that $\Delta_0^{\downarrow} \cup \{\Box\psi\} \in H$.

Then, according to H, we can define a winning strategy for Sally $\rho_S : P_S \to P_E$ in the game $\mathcal{T}(\varphi)$ as follows. For each position $\{\pi, \bigcirc\Box\psi\} \in P_S$, we define $\rho_S(\{\pi, \bigcirc\Box\psi\}) = \Delta^{\downarrow} \cup \{\Box\psi\}$ for some chosen Δ such that $\Delta^{\downarrow} \cup \{\Box\psi\} \in H$. Since H and $B^{\mathcal{T}}$ are trivially disjoint, it is obvious that $\delta(i) \in P_S \setminus B^{\mathcal{T}}$ for every initial play $\delta = \delta(0), \delta(1), \ldots$ played according to ρ_S and all $i \in \mathbb{N}$. Therefore, ρ_S is winning for Sally. □

The following follows immediately

Corollary 3. *A safety specification φ is realizable if and only if the completed tableau for $\mathsf{Tab}(\varphi)$ is open. Moreover, any bunch in $\mathsf{Tab}(\varphi)$ such that all its leaves are successful represents a winning strategy for Sally.*

In the following example we illustrate Corollary 3 providing a tableau from which the winning strategy can be extracted.

Example 12. We consider a variant of a synthesis problem about a simple arbiter presented in [14]. The arbiter receives requests from two clients, represented by two environment variables $\mathcal{X} = \{r_1, r_2\}$, and responds by assigning grants, represented by two system variables $\mathcal{Y} = \{g_1, g_2\}$. We specify that each request should eventually be followed by a grant in at most three second, and that the two grants should never be assigned simultaneously.

We assume that initially there are not requests and impose an additional requirement to hinder the winning strategy. The safety specification is as follows.

$$\Box\psi = \Box((r_1 \to \Diamond_{[0,3]} g_1) \wedge (r_2 \to \Diamond_{[0,3]} g_2) \wedge \neg(g_1 \wedge g_2) \wedge ((\neg r_1 \wedge \neg r_2) \to \bigcirc\neg g_2))$$

In Figure 11 we show an open tableau for $\Box\psi$, whose construction starts with C_1, the weakest minimal $\mathcal{X}$-covering in $\mathsf{TNF}(\psi)$, composed by the labels of nodes n_2, n_3, n_4 and n_5. At node n_8, the weakest minimal $\mathcal{X}$-covering in $\mathsf{TNF}(\Diamond_{[0,2]}\, g_1 \wedge \psi)$ is C_2, which has the following four moves:

$$(r_1 \wedge \neg r_2 \wedge g_1 \wedge \neg g_2) \vee (\neg r_1 \wedge r_2 \wedge g_1 \wedge \neg g_2 \wedge \bigcirc\Diamond_{[0,2]}\, g_2) \vee$$
$$(r_1 \wedge r_2 \wedge g_1 \wedge \neg g_2 \wedge \bigcirc\Diamond_{[0,2]}\, g_2) \vee (\neg r_1 \wedge \neg r_2 \wedge g_1 \wedge \neg g_2 \wedge \bigcirc\neg g_2)$$

that we name by m_1, m_2, m_3, m_4 in the given order. They are represented in the nodes of the tableau with the same name (in abuse of notation). Note that, for simplicity, we group m_2 and m_3 in the same node that omits the value of r_1, which is the only difference between both moves. At node n_{10}, C_3, is the weakest minimal $\mathcal{X}$-covering in $\mathsf{TNF}(\Diamond_{[0,2]}\, g_2 \wedge \psi)$. It has four moves m_1', m_2', m_3', m_4' but m_2' and m_3' has been grouped. And similarly, at node n_{13}, where $\mathsf{TNF}(\neg g_2 \wedge \psi)$ provides C_4, with the moves $m_1'', m_2'', m_3'', m_4''$. Note that nodes m_4', m_4 and n_5 share the same strict-future formula. Hence, to save space, we do not depict the expansion of nodes m_4 and n_5 since it repeats the tableau behind node m_4'. All in all, the completed tableau for the input specification is open. Moreover, the

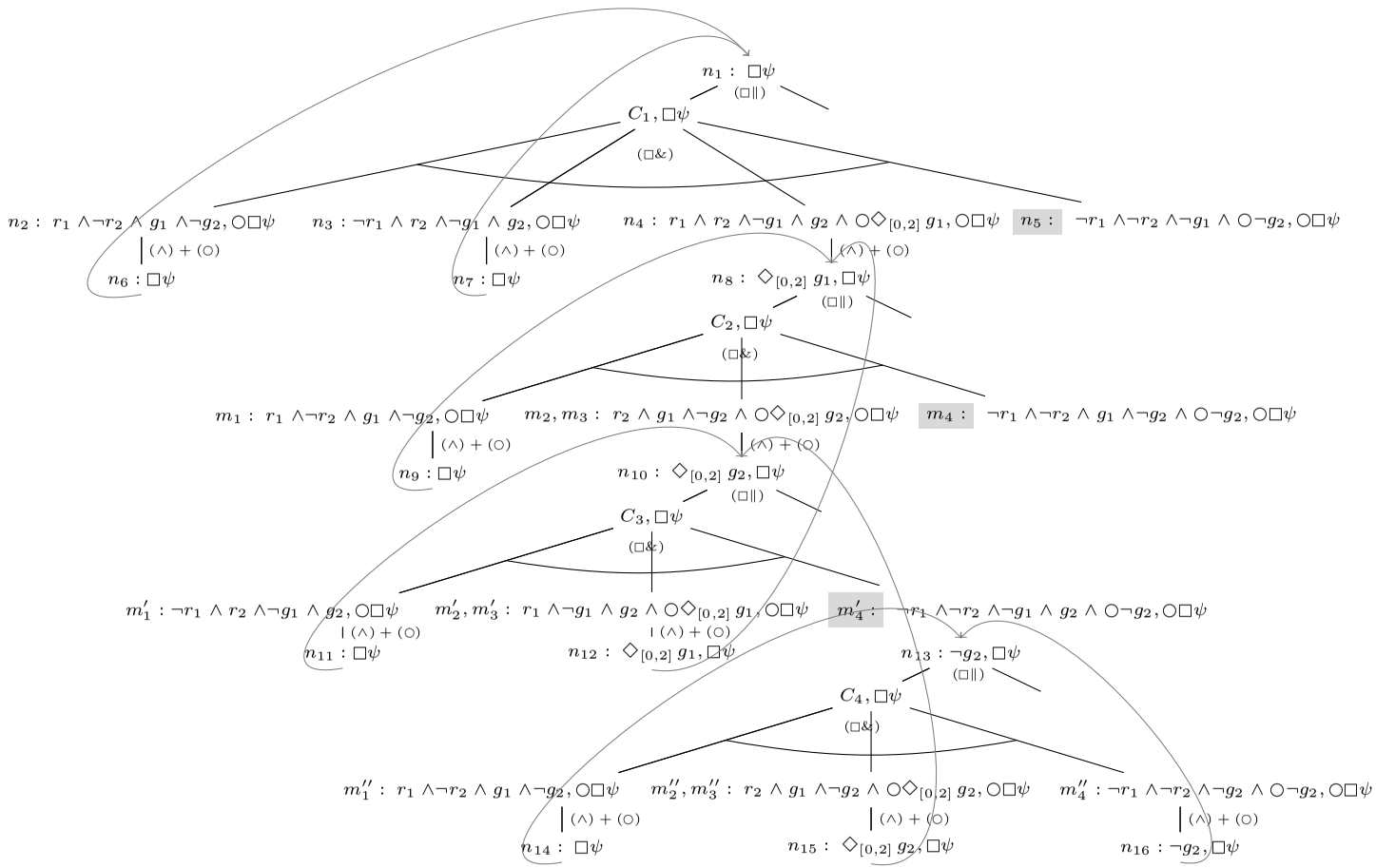

Fig. 11. Open tableau for $\Box((r_1 \rightarrow \Diamond_{[0,3]}\, g_1) \wedge (r_2 \rightarrow \Diamond_{[0,3]}\, g_2) \wedge \neg(g_1 \wedge g_2) \wedge ((\neg r_1 \wedge\neg r_2) \rightarrow \bigcirc\neg g_2))$.

winning strategy represented by the finite state machine in Figure 12 can be synthesised from this tableau.

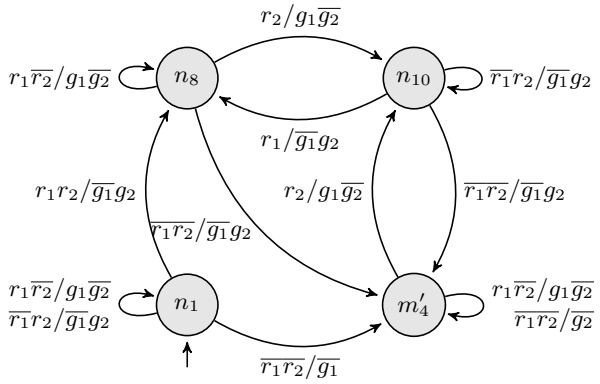

Fig. 12. Finite state machine that represents the strategy synthesised from the open tableau in Figure 11

In Figure 12, the machine nodes has been intentionally named as some of the tableau nodes, to make easier to see the correspondence between the transition edges labels of the form Eve/Sally and the moves in the tableau. The forward slash separates environment variables from system variables. The negation operator is represented with a top line and the $\wedge$-operator has been omitted.

7 Conclusions

We have introduced the first tableau method to decide realizability of temporal formulas. Our tableau method allows to synthesize a system when the specification is realizable. Our realizability tableau method is based on the novel notion terse normal form (TNF) of formulas that is crucial in the tableau formulation. Our realizability 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. Our proofs imply a method to synthesize a correct system for a realizable specification, as illustrated that in Example 12.

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 also plan to extend the method to more expressive languages, including the handling of richer propositional languages (like numeric variables) by combining realizability tableau rules with tableau reasoning capabilities for these domains. This has been illustrated by the handling of enumerated types in this paper. 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.

References

1. `https://syntcomp.org`.
2. 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.
3. 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.
4. 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.
5. 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 Syntehsis (SYNT'14)*, volume 157 of *EPTCS*, pages 100–116, 2014.
6. J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138, 1969.
7. Martin De Wulf, Laurent Doyen, Nicolas Maquet, and Jean-FrançoisRaskin. 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.
8. 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.
9. Rüdiger Ehlers. Unbeast: Symbolic bounded synthesis. In *Proc. of TACAS'11*, volume 6605 of *LNCS*, pages 272–275. Springer, 2011.
10. 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.
11. Bernd Finkbeiner and Swen Jacobs. Lazy synthesis. In *Proc. of VMCAI'12*, volume 7148 of *LNCS*, pages 219–234. Springer, 2012.
12. Bernd Finkbeiner and F. Klein. Bounded cycle synthesis. In *Proc. of CAV'16*, volume 9779 of *LNCS*, pages 118–135. Springer, 2016.
13. 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.
14. Bernd Finkbeiner and Sven Schewe. Bounded synthesis. *Int. J. Softw. Tools Technol. Transf.*, 15(5-6):519–539, 2013.
15. 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.
16. 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.

17. Rajeev Goré and Florian Widmann. An optimal on-the-fly tableau-based decision procedure for PDL-satisfiability. In *Proc. of the 22nd Int'l Conf. on Automated Deduction (CADE'09)*, volume 5663 of *LNCS*, pages 437–452. Springer, 2009.
18. Alexey Ignatiev, Alessandro Previti, and João Marques-Silva. Sat-based formula simplification. In *Theory and Applications of Satisfiability Testing - SAT 2015 - 18th International Conference, Austin, TX, USA, September 24-27, 2015, Proceedings*, volume 9340 of *Lecture Notes in Computer Science*, pages 287–298. Springer, 2015.
19. 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.
20. 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.
21. 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.
22. 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.
23. 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.
24. 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.
25. Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *Proc. of POPL'89*, pages 179–190. ACM, 1989.
26. 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.
27. Sven Schewe and Bernd Finkbeiner. Bounded synthesis. In *Proc. of ATVA'07*, volume 4762 of *LNCS*. Springer, 2007.
28. 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.
29. 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.
30. Raymond M. Smullyan. A unifying principal in quantification theory. *Proceedings of the National Academy of Sciences*, 49(6):828–832, 1963.
31. Moshe Y. Vardi and Pierre Wolper. Reasoning about infinite computations. *Information and computation*, 115(1):1–37, 1994.
32. Pierre Wolper. The tableau method for temporal logic: An overview. *Logique et Analyse*, 28:119–136, 1985.