LTLf Synthesis with Fairness and Stability Assumptions

Shufang Zhu,1 Giuseppe De Giacomo,2 Geguang Pu,1 Moshe Y. Vardi3
1 Each China Normal University, 2 Sapienza Università di Roma, 3 Rice University
shufangzhu.szu@gmail.com, degiacomo@diag.uniroma1.it, ggpu@sei.ecnu.edu.cn, vardi@cs.rice.edu

Abstract

In synthesis, assumptions are constraints on the environment that rule out certain environment behaviors. A key observation here is that even if we consider systems with LTLf goals on finite traces, environment assumptions need to be expressed over infinite traces, since accomplishing the agent goals may require an unbounded number of environment action. To solve synthesis with respect to finite-trace LTLf goals under infinite-trace assumptions, we could reduce the problem to LTL synthesis. Unfortunately, while synthesis in LTLf and in LTL have the same worst-case complexity (both 2EXPTIME-complete), the algorithms available for LTL synthesis are much more difficult in practice than those for LTLf synthesis. In this work we show that in interesting cases we can avoid such a detour to LTL synthesis and keep the simplicity of LTLf synthesis. Specifically, we develop a BDD-based fixpoint-based technique for handling basic forms of fairness and of stability assumptions. We show, empirically, that this technique performs much better than standard LTL synthesis.

Introduction

In many situations we are interested in expressing properties over an unbounded but finite sequence of successive states. Linear-time Temporal Logic over finite traces (LTLf) and its variants have been thoroughly investigated for doing so. There has been broad research for logical reasoning (De Giacomo and Vardi 2013; Li et al. 2019), synthesis (De Giacomo and Vardi 2015; Camacho et al. 2018), and planning (Camacho et al. 2017; De Giacomo and Rubin 2018).

Recently synthesis under assumptions in LTLf has attracted specific interest (De Giacomo and Rubin 2018; Camacho, Bienvenu, and McIraith 2018). First, planning for LTLf goals can be considered as a form of LTLf synthesis under assumptions, where the assumptions are the dynamics of the environment encoded in the planning domain (Green 1969; Camacho, Bienvenu, and McIraith 2018; Aminof et al. 2018; Aminof et al. 2019). However, more generally, assumptions can be arbitrary constraints on the environment that can be exploited by the agent in devising a strategy to fulfill its goal.

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Synthesis under assumptions has been extensively studied in LTL, where environment assumptions are expressed as LTL formulas (Chatterjee and Henzinger 2007; Chatterjee, Henzinger, and Jobstmann 2008; D’Ippolito et al. 2013; Bloem, Ehlers, and Könighofer 2015; Brenguier, Raskin, and Sankur 2017). In fact, LTL formulas can be used as assumptions as long as it is guaranteed that the environment is able to behave so as to keep the assumptions true, i.e., assumptions are environment realizable. Under these circumstances, it is possible to reduce synthesis for LTL goal \( \psi_G \) under assumptions as \( \psi_A \) to standard synthesis for \( \psi_A \rightarrow \psi_G \). Note that because of the guarantee of \( \psi_A \) being environment realizable, no agent strategy can win \( \psi_A \rightarrow \psi_G \) by falsifying \( \psi_A \). See (Aminof et al. 2019) for a discussion.

When we turn to LTLf, a key observation is that even if we consider (finite-trace) LTLf goals for the agent, assumptions need to be expressed considering infinite traces, since accomplishing the agent goals may require an unbounded number of environment action. So we have an assumption \( \psi_A \) expressed in LTL and a goal \( \phi_G \) expressed in LTLf. To solve synthesis under assumptions in LTLf, we could translate \( \phi_G \) into LTL getting \( \psi_G \), by applying the translation of LTLf into LTL in (De Giacomo and Vardi 2013), and then do LTL synthesis for \( \psi_A \rightarrow \psi_G \), see e.g. (Camacho, Bienvenu, and McIraith 2018).

Unfortunately, while synthesis in LTLf and in LTL have the same worst-case complexity, being both 2EXPTIME-complete (Pnueli and Rosner 1989; De Giacomo and Vardi 2015), the algorithms available for LTL synthesis are much harder in practice than those for LTLf synthesis. In particular, the lack of efficient algorithms for the crucial step of automata determinization is prohibitive for finding scalable implementations (Fogarty et al. 2013; Finkbeiner 2016). In spite of recent advancement in synthesis such as reducing to parity games (Meyer, Sickert, and Luttenberger 2018), bounded synthesis based on solving iterated safety games (Kupferman and Vardi 2005; Finkbeiner and Schewe 2013), Iterated FOND planning (Camacho et al. 2018), LTL synthesis remains very challenging. In contrast, LTLf synthesis is based on a translation to Deterministic Finite Automaton (DFA) (Rabin and Scott 1959), which...
can be seen as a game arena where environment and agent make their own moves. On this arena, the agent wins if a simple fixpoint condition (reachability of the DFA accepting states) is satisfied.

Hence, when we introduce assumptions, an important question arises: can we retain the simplicity of LTL synthesis? In particular, we are thinking about algorithms based on devising some sort of arena and then extracting winning strategies by relying on computing a small number of nested fixpoints (note that the reduction of LTL synthesis to parity games may generate exponentially many nested fixpoints (Grädel, Thomas, and Wilke 2002)).

We consider here two different basic, but quite significant, forms of assumptions: a basic form of fairness GFα (always eventually α), and a basic form of stability FGα (eventually always α), where in both cases the truth value of α is under the control of the environment, and hence the assumptions are trivially realizable by the environment. Note that due to the existence of LTL, goals, synthesis under both kinds of assumptions does not fall under known easy forms of synthesis, such as GR(1) formulas (Bloem et al. 2012). For these kinds of assumptions, we devise a specific algorithm based on using the DFA for the LTL, goal as the arena and then computing 2-nested fixpoint properties over such arena. It should be noted that the kind of nested fixpoint that we compute for fairness GFα is similar to the one in (De Giacomo and Rubin 2018), but it is clear that the “fairness” stated there is different from what we claim in this paper. The “fairness” in (De Giacomo and Rubin 2018) is interpreted as all effects happening fairly, therefore the assumption is hardcoded in the arena itself. Here, instead, we only require that a selected condition α happens fairly, and our technique extends to deal with stability assumptions as well. We compare the new algorithm with standard LTL synthesis (Meyer, Sickert, and Luttenberger 2018) and show empirically that this algorithm performs significantly better, in the sense that solving more cases with less time cost. Some proofs have been removed due to the lack of space.

Preliminaries

Linear-time Temporal Logic over finite traces (LTLf) has the same syntax as LTL over infinite traces introduced in [Pnueli 1977]. Given a set of propositions P, the syntax of LTLf formulas is defined as φ ::= a | ¬φ | φ1 & φ2 | Xφ | φ1Uφ2. Every a ∈ P is an atom. A literal l is an atom or the negation of an atom. X for “Next”, and U for “Until”, are temporal operators. We make use of the standard Boolean abbreviations, such as ∨ (or) and → (implies), true and false. Additionally, we define the following abbreviations “Weak Next” Xa, φ ≡ ¬X¬φ, “Eventually” Fφ ≡ trueUφ and “Always” Gφ ≡ falseRφ, where R is for “Release”.

A trace ρ = ρ[0], ρ[1], . . . is a sequence of propositional interpretations (sets), where ρ[m] ∈ 2P (m ≥ 0) is the m-th interpretation of ρ, and |ρ| represents the length of ρ. Intuitively, ρ[m] is interpreted as the set of propositions which are true at instant m. Trace ρ is an infinite trace if |ρ| = ∞, which is formally denoted as ρ ∈ (2P)ω; otherwise ρ is a finite trace, denoted as ρ ∈ (2P)+. LTLf formulas are interpreted over finite, nonempty traces. Given a finite trace ρ and an LTLf formula φ, we inductively define when φ is true on ρ at step i (0 ≤ i < |ρ|), written ρ, i |= φ, as follows:

• ρ, i |= a iff a ∈ ρ[i];
• ρ, i |= ¬φ iff ρ, i |= φ;
• ρ, i |= φ1 & φ2 iff ρ, i |= φ1 and ρ, i |= φ2;
• ρ, i |= Xφ iff i + 1 < |ρ| and ρ, i + 1 |= φ;
• ρ, i |= φ1Uφ2 iff there exists j such that i ≤ j < |ρ| and ρ, j |= φ2, and for all k, i ≤ k < j, we have ρ, k |= φ1.

A fair or stable trace can then be defined in terms of the corresponding assumption (fairness or stability).

Definition 1 (Environment Constraint). An environment constraint α is a Boolean formula over X.

In particular, we define here two different basic, but common forms of assumptions.

Definition 2 (Fairness and Stability Assumptions). An LTL formula ψ is considered as a fairness assumption if it is of the form GFα, and a stability assumption if of the form FGα, where in both cases α is an environment constraint.

A fair or stable trace can then be defined in terms of the corresponding assumption (fairness or stability).

Definition 3 (Fair and Stable Traces). A trace ρ ∈ (2U∪Γ)ω is α-fair if ρ |= GFα and it is α-stable if ρ |= FGα.

Intuitively, α holds infinitely often on an α-fair trace, while eventually holds forever on an α-stable trace. Note that, if trace ρ is not α-fair, i.e., ρ ⊈ GFα, then ρ |= FG(¬α) such that ρ is α-stable. Similarly, if trace ρ is not α-stable, i.e., ρ ⊈ FGα, then ρ |= GF(¬α) such that ρ is α-fair.

An LTLf specification can be viewed as a game of two players, the environment and the agent, contrasting each other. The aim is to synthesize a strategy for the agent such that no matter how the environment behaves, the combined behavior of both players satisfy the logical specification expressed in LTLf (De Giacomo and Vardi 2015).

Fair and Stable LTLf Synthesis

In this paper, we focus on LTLf synthesis under assumptions in two different basic forms: fairness and stability, which we call in the following fair LTLf synthesis and stable LTLf synthesis, respectively. In such synthesis problems, both players (environment and agent) have Boolean variables under their respective control. Here, we use X to denote the set of environment variables that are uncontrollable for the agent, and Y the set of agent variables that are controllable for the agent. Therefore, X and Y are disjoint.

In general, assumptions are specific forms of constraints.

Definition 1 (Environment Constraint). An environment constraint α is a Boolean formula over X.

In particular, we define here two different basic, but common forms of assumptions.

Definition 2 (Fairness and Stability Assumptions). An LTL formula ψ is considered as a fairness assumption if it is of the form GFα, and a stability assumption if of the form FGα, where in both cases α is an environment constraint.

A fair or stable trace can then be defined in terms of the corresponding assumption (fairness or stability).

Definition 3 (Fair and Stable Traces). A trace ρ ∈ (2U∪Γ)ω is α-fair if ρ |= GFα and it is α-stable if ρ |= FGα.

Intuitively, α holds infinitely often on an α-fair trace, while eventually holds forever on an α-stable trace. Note that, if trace ρ is not α-fair, i.e., ρ ⊈ GFα, then ρ |= FG(¬α) such that ρ is α-stable. Similarly, if trace ρ is not α-stable, i.e., ρ ⊈ FGα, then ρ |= GF(¬α) such that ρ is α-fair. Although there is a duality between fairness and stability, such duality breaks when applying these environment assumptions to the problem of LTLf synthesis. This is because in addition to the assumptions, the synthesis problems also require the LTLf specification to be satisfied.

We now define fair and stable LTLf synthesis by making use of fair and stable traces.
Definition 4 (Fair (Stable) \( \text{LTL}_f \) Synthesis). \( \text{LTL}_f \) formula \( \phi \), defined over \( \mathcal{X} \cup \mathcal{Y} \), is \( \alpha \)-fair (resp., \( \alpha \)-stable) realizable if there exists a strategy \( g : (2^X)^+ \rightarrow 2^Y \), such that for an arbitrary environment trace \( \lambda = X_0, X_1, \ldots \in (2^X)^\omega \), if \( \lambda \) is \( \alpha \)-fair (resp., \( \alpha \)-stable), then we can find \( k \geq 0 \) such that \( \phi \) is true in the finite trace \( \rho_k = (X_0 \cup g(X_0)), (X_1 \cup g(X_0, X_1)), \ldots, (X_k \cup g(X_0, X_1, \ldots, X_k)) \).

A fair (resp., stable) \( \text{LTL}_f \) synthesis problem, described as a tuple \( (\mathcal{X}, \mathcal{Y}, \alpha, \phi) \), consists in checking whether \( \phi \), defined over \( \mathcal{X} \cup \mathcal{Y} \), is \( \alpha \)-fair (resp., \( \alpha \)-stable) realizable. The synthesis procedure aims to computing a strategy if realizable.

Intuitively speaking, \( \phi \) describes the desired goal when the environment behaviors satisfy the assumption. An agent strategy \( g : (2^X)^+ \rightarrow 2^Y \) for fair (resp., stable) synthesis problem \( (\mathcal{X}, \mathcal{Y}, \alpha, \phi) \) is \emph{winning} if it guarantees the satisfaction of the objective \( \phi \) under the condition that the environment behaves in a way that \( \alpha \) holds infinitely often (resp., \( \alpha \) eventually holds forever). The realizability procedure of \( (\mathcal{X}, \mathcal{Y}, \alpha, \phi) \) aims to answer the existence of a winning strategy \( g \) and the synthesis procedure amounts for computing \( g \) if it exists. In fact one can consider two variants of the synthesis problem, depending on the player who moves first, in the sense of assigning values to variables under its control first. Here we consider the environment as the first-player (as in planning), but a version where the agent moves first can be obtained by a small modification.

Since every \( \text{LTL}_f \) formula \( \phi \) can be translated to a Deterministic Finite Automaton (DFA) \( G_\phi \) that accepts exactly the same language as \( \phi \) (De Giacomo and Vardi 2013), we are able to reduce the problems of fair \( \text{LTL}_f \) synthesis and stable \( \text{LTL}_f \) synthesis to specific two-player DFA games, in particular, fair DFA game and stable DFA game, respectively. We start with introducing DFA games.

Games over DFA

Two-player games on DFA are games consisting of two players, the \emph{environment} and the \emph{agent}. \( \mathcal{X} \) and \( \mathcal{Y} \) are disjoint sets of environment Boolean variables and agent Boolean variables, respectively. The specifications of the game arena is given by a DFA \( G = (2^{X \cup Y}, S, s_0, \delta, \text{Acc}) \), where \( 2^{X \cup Y} \) is the alphabet, \( S \) is a set of states, \( s_0 \in S \) is an initial state, \( \delta : S \times 2^{X \cup Y} \rightarrow S \) is a transition function and \( \text{Acc} \subseteq S \) is a set of accepting states.

A \emph{round} of the game consists of both players setting the values of variables under their respective control. A \emph{play} \( \rho \) over \( G \) records how two players set the values at each round and how the DFA proceeds according to the values. Formally, a play \( \rho \) from state \( s_0 \) is an infinite trace \( (s_{i_0}, X_0 \cup Y_0), (s_{i_1}, X_1 \cup Y_1), \ldots \in (S \times 2^{X \cup Y})^\omega \) such that \( s_{i+1} = \delta(s_i, X_j \cup Y_j) \). Moreover, we also assign the \emph{environment} as the first-player, which sets values first.

A play \( \rho \) is considered as a winning play if it follows a certain winning condition. Different winning conditions lead to different games. In this paper, we consider two specific two-player games, fair DFA game and stable DFA game, both of which are described as \( (G, \alpha) \), where \( G \) is the game arena and \( \alpha \) is the environment constraint.

Fair DFA Game. Although the ultimate goal for solving a fair DFA game is to perform winning plays for the agent, since it is more straightforward to formulate the game considering the environment as the protagonist, we first define the winning condition of the environment over a play. A play \( \rho = (s_{i_0}, X_0 \cup Y_0), (s_{i_1}, X_1 \cup Y_1), \ldots \) over \( G \) is \emph{winning} for the environment with respect to a fair DFA game \( (G, \alpha) \) if the following two conditions hold:

- **Recurrence**: \( \rho \) is \( \alpha \)-fair (that is, \( \rho \models LTL(\alpha) \)),
- **Safety**: \( s_i \notin \text{Acc} \) for all \( j \geq 0 \) (\( \text{Acc} \) is avoided).

Consequently, a play \( \rho \) is \emph{winning} for the agent if one of the following conditions holds:

- **Stability**: \( \rho \) is not \( \alpha \)-fair (that is, \( \rho \models \neg LTL(\alpha) \)),
- **Reachability**: \( s_i \in \text{Acc} \) for some \( j \geq 0 \) (\( \text{Acc} \) is reached).

Stable DFA Game. As for a stable DFA game \( (G, \alpha) \), a play \( \rho = (s_{i_0}, X_0 \cup Y_0), (s_{i_1}, X_1 \cup Y_1), \ldots \) over \( G \) is \emph{winning} for the environment if the following two conditions hold:

- **Stability**: \( \rho \) is \( \alpha \)-stable (that is, \( \rho \models \neg LTL(\alpha) \)),
- **Safety**: \( s_i \notin \text{Acc} \) for all \( j \geq 0 \) (\( \text{Acc} \) is avoided).

Consequently, a play \( \rho \) is \emph{winning} for the agent if one of the following conditions holds:

- **Recurrence**: \( \rho \) is not \( \alpha \)-stable (that is, \( \rho \models LTL(\alpha) \)),
- **Reachability**: \( s_i \in \text{Acc} \) for some \( j \geq 0 \) (\( \text{Acc} \) is reached).

Since we consider here the environment as the first-player, a strategy for the agent is a function \( g : (2^X)^+ \rightarrow 2^Y \), deciding the values of the controllable variables for every possible history of the uncontrollable variables. Respectively, an environment strategy is a function \( h : (2^X)^+ \rightarrow 2^X \). A play \( \rho = (s_{i_0}, X_0 \cup Y_0), (s_{i_1}, X_1 \cup Y_1), \ldots \in (S \times 2^{X \cup Y})^\omega \) follows a strategy \( g \) (resp., a strategy \( h \)), if \( Y_j = g(X_0, \ldots, X_j) \) for all \( j \geq 0 \) (resp., \( X_j = h(Y_0, \ldots, Y_{j-1}) \) for all \( j > 0 \)).

We can now define winning states and winning strategies.

Definition 5 (Winning State and Winning Strategy). In the game \( (G, \alpha) \) described above, \( s \in S \) is a \emph{winning state} for the agent (resp., environment) if there exists strategy \( g \) (resp., \( h \)) s.t. every play \( \rho \) from \( s \) that follows \( g \) (resp., \( h \)) is a \emph{winning strategy} for the agent (resp., environment) from \( s \).

As shown in (Martin 1975), both of the fair DFA game and stable DFA game described above are determined, that is, a state \( s \in S \) is a winning state for the agent if and only if \( s \) is not a winning state for the environment. The realizability of the game consists of checking whether there exists a winning strategy for the agent from initial state \( s_0 \). The synthesis procedure aims to computing such a strategy.

We then show how to reduce the problems of fair \( \text{LTL}_f \) synthesis and stable \( \text{LTL}_f \) synthesis to fair DFA game and stable DFA game, respectively. Hence we can solve the DFA game, thus settling the corresponding synthesis problem.

Solution to Fair \( \text{LTL}_f \) Synthesis

In order to perform fair synthesis on \( \text{LTL}_f \), given problem \( (\mathcal{X}, \mathcal{Y}, \alpha, \phi) \), we first translate the \( \text{LTL}_f \) specification \( \phi \) into a DFA \( G_\phi \). We then view \( (G_\phi, \alpha) \) as a fair DFA game, and consider exactly the separation between environment and agent variables as in the original synthesis problem. Specifically, we assign \( \mathcal{X} \) as the environment variables and \( \mathcal{Y} \) as the agent variables. Finally, we solve the fair DFA game, thus settling the fair \( \text{LTL}_f \) synthesis problem. The following theorem assesses the correctness of this technique.
Theorem 1. Fair LTL synthesis problem \( (X, Y, \alpha, \phi) \) is realizable iff fair DFA game \( (G_\phi, \alpha) \) is realizable.

Proof. We prove the theorem in both directions.

\( \leftarrow \): Since \( (G_\phi, \alpha) \) is realizable for the agent, the initial state \( s_0 \) is an agent winning state with winning strategy \( g : (2^X)^+ \rightarrow 2^Y \). Therefore, a play \( \rho = (s_0, X_0 \cup g(X_0)), (s_1, X_1 \cup g(X_0, X_1)), \ldots \) follows from \( G_\phi \) such that \( g \) is a winning play for the agent. Moreover, for every such play \( \rho \) from \( s_0 \), either of the following conditions holds:

- \( \rho \not\models GF\alpha \) such that \( \rho \) is not \( \alpha \)-fair.
- \( \rho \models GF\alpha \) such that \( \rho \) is \( \alpha \)-fair. Since \( \rho \) is winning for the agent, the environment exists, and \( s_j \in Acc \). This implies that \( \rho^j \models \phi \) holds, where \( \rho^j = (s_0, X_0 \cup g(X_0)), (s_1, X_1 \cup g(X_0, X_1)), \ldots, (s_j, X_j \cup g(X_0, X_1, \ldots, X_j)) \).

Consequently, the strategy \( g \) assures that for an arbitrary environment trace \( \lambda = X_0, X_1, \ldots \in (2^X)^\omega \), if \( \lambda \) is \( \alpha \)-fair, then there is \( j \geq 0 \) such that \( s_j \in Acc \). We conclude that \( (X, Y, \alpha, \phi) \) is realizable.

\( \rightarrow \): For this direction, we assume that \( (X, Y, \alpha, \phi) \) is realizable, then there exists a strategy \( g : (2^X)^+ \rightarrow 2^Y \) that realizes \( \phi \). Thus consider an arbitrary environment trace \( \lambda = X_0, X_1, \ldots \in (2^X)^\omega \), if \( \lambda \) is \( \alpha \)-fair, then there is \( j \geq 0 \) such that \( s_j \in Acc \). We conclude that \( (X, Y, \alpha, \phi) \) is realizable for the agent.

Fair DFA Game Solving

Winning fair DFA games means that the agent can eventually reach an “agent wins” region from which if the constraint \( \alpha \) holds, then it is possible to reach an accepting state. Given a fair DFA game \( (G, \alpha) \), we proceed as follows: (1) Compute “agent wins” region in fair DFA game \( (G, \alpha) \); (2) Check realizability; (3) Return an agent winning winning strategy if realizable.

Since the environment winning condition is more intuitive, in order to show the solution to fair DFA game, we start by solving the Recurrence-Safety game, which considers the environment as the protagonist. The idea for winning such game is that the environment should remain in an “environment wins” region from which the constraint \( \alpha \) holds infinitely often referring to Recurrence game, meanwhile the accepting states are forever avoidable referring to Safety game. Therefore, in order to have both of Recurrence such that having GF\alpha holds and Safety such that avoiding accepting states \( s \in Acc \), the “environment wins” region computation is defined as:

\[
\text{Env}_f = \nu Z.\mu Z.(\exists X.\forall Y.(X \models \alpha \land \delta(s, X \cup Y) \in Z \setminus Acc) \lor (s, X \cup Y) \in Z \setminus Acc),
\]

where \( X \) ranges over \( 2^X \) and \( Y \) over \( 2^Y \).

The fixpoint stages for \( Z \) (note \( Z_{i+1} \subseteq Z_i \), for \( i \geq 0 \), by monotonicity) are:

- \( Z_0 = S \).
- \( Z_{i+1} = \mu Z.(\exists X.\forall Y.(X \models \alpha \land \delta(s, X \cup Y) \in Z_i \setminus Acc) \lor (s, X \cup Y) \in Z \setminus Acc)). \)

Eventually, \( \text{Env}_f = Z_k \) for some \( k \) such that \( Z_{k+1} = Z_k \).

The fixpoint stages for \( \hat{Z} \) with respect to \( \hat{Z}_i \) (note \( \hat{Z}_i \subseteq \hat{Z}_{i+1} \), for \( j \geq 0 \), by monotonicity) are:

- \( \hat{Z}_0 = \emptyset \).
- \( \hat{Z}_{i+1} = \exists X.\forall Y.((X \models \alpha \land \delta(s, X \cup Y) \in Z_i \setminus Acc) \lor (s, X \cup Y) \in \hat{Z}_{i+1} \setminus Acc)). \)

Finally, \( \hat{Z} = \hat{Z}_{i,k} \) for some \( k \) such that \( \hat{Z}_{i+k+1} = \hat{Z}_{i,k} \).

The following theorem assures that the nested fixpoint computation of \( \text{Env}_f \) collects exactly all environment winning states in fair DFA game.

Theorem 2. For a fair DFA game \( (G, \alpha) \) and a state \( s \in S \), we have \( s \in \text{Env}_f \) iff \( s \) is an environment winning state.

Proof. We prove the two directions separately.

\( \leftarrow \): We prove by showing the contrapositive. If a state \( s \notin \text{Env}_f \), then \( s \) must be removed from \( \text{Env}_f \) at stage \( i+1 \), therefore, \( s \in Z_i \setminus Z_{i+1} \). Then \( s \notin \mu Z.(\exists X.\forall Y.((X \models \alpha \land \delta(s, X \cup Y) \in Z_i \setminus Acc) \lor (s, X \cup Y) \in \hat{Z}_{i+1} \setminus Acc)). \) That is, no matter what the (environment) strategy \( h \) is, traces from \( s \) satisfy neither of the following conditions:

- \( \alpha \) holds and the trace gets trapped in \( Z \) without visiting accepting states such that \( X \models \alpha \land \delta(s, X \cup Y) \in Z \setminus Acc \), in which case \( s \) is a new environment winning state;
- \( \alpha \) eventually gets hold and from there we can have \( \alpha \) as true infinitely often without visiting accepting states such that \( \delta(s, X \cup Y) \in \hat{Z} \setminus Acc \), in which case \( s \) is a new environment winning state.

Therefore, \( s \) is not an environment winning state. So if \( s \) is an environment winning state then \( s \in \text{Env}_f \) holds.

\( \rightarrow \): If a state \( s \in \text{Env}_f \), then \( s \in \mu Z.(\exists X.\forall Y.((X \models \alpha \land \delta(s, X \cup Y) \in Z \setminus Acc) \lor (s, X \cup Y) \in \hat{Z} \setminus Acc)). \) That is, no matter what the (agent) strategy \( g \) is, traces from \( s \) satisfy either of the following conditions:

- \( \alpha \) holds and the trace gets trapped in \( Z \) without visiting accepting states such that \( X \models \alpha \land \delta(s, X \cup Y) \in Z \setminus Acc \), in which case \( s \) is a new environment winning state;
- \( \alpha \) eventually gets hold and from there we can have \( \alpha \) as true infinitely often without visiting accepting states such that \( \delta(s, X \cup Y) \in \hat{Z} \setminus Acc \), in which case \( s \) is a new environment winning state.

Thus \( s \) is a winning state for the environment.

Strategy Extraction

Having completed the realizability checking procedure, this section deals with the agent winning strategy generation if \( (G, \alpha) \) is realizable. It is known that if some strategy that
realizes $\phi$ exists, then there also exists a finite-state strategy generated by a finite-state transducer that realizes $\phi$ [Büchi and Landweber 1990]. Formally, the agent winning strategy $g : (2^X)^+ \to 2^Y$ can be represented as a deterministic finite transducer based on the set $\text{Sys}_f$, described as below.

**Definition 6 (Deterministic Finite Transducer).** Given a fair DFA game $\langle G, \alpha \rangle$, where $G = (2^X, Y, S, s_0, \delta, Acc)$, a deterministic finite transducer $T = (2^X, 2^Y, Q, s_0, \omega, \omega_f)$ of such game is defined as follows:

- $Q \subseteq S$ is the set of agent winning states s.t. $Q = \text{Sys}_f$;
- $\omega : Q \times 2^X \to Q$ is the transition function such that $\omega(q, X) = \delta(q, X \cup Y)$ and $Y = \omega_f(q, X)$;
- $\omega_f : Q \times 2^X \to 2^Y$ is the output function such that at an agent winning state $q$ with assignment $X$, $\omega_f(q, X)$ returns an assignment $Y$ leading to an agent winning play.

The transducer $T$ generates $g$ in the sense that for every $\lambda \in (2^X)^\omega$, we have $g(\lambda) = \omega_f(\omega(\lambda))$, with the usual extension of $\delta$ to words over $2^X$ from $s_0$. Note that there are many possible choices for the output function $\omega_f$. The transducer $T$ defines a winning strategy by restricting $\omega_f$ to return only one possible setting of $Y$.

We extract the output function $\omega_f : Q \times 2^X \to 2^Y$ for the game from the approximates for $Z$ approximating to $\text{Sys}_f$, from where no matter what the environment strategy is, traces have to always get $\alpha$-stable. Thus, we consider the following theorem guarantees that $T$ generates an agent winning strategy $g$.

**Theorem 4.** Strategy $g$ with $g(\lambda) = \omega_f(\omega(\lambda))$ is a winning strategy for the agent.

**Solution to Stable LTL$_f$ Synthesis**

Solving stable LTL$_f$ synthesis problem $\langle X, Y, \alpha, \phi \rangle$ relies on solving the stable DFA game $\langle G_\phi, \alpha \rangle$, where $G_\phi$ is the corresponding DFA of $\phi$. The following theorem guarantees the correctness of such reduction.

**Theorem 5.** Stable LTL$_f$ synthesis problem $\langle X, Y, \alpha, \phi \rangle$ is realizable iff stable DFA game $\langle G_\phi, \alpha \rangle$ is realizable.

**Proof.** We prove the theorem in both directions.

$(\Rightarrow)$ Since $\langle G_\phi, \alpha \rangle$ is realizable for the agent, the initial state $s_0$ is an agent winning state with winning strategy $g : (2^X)^+ \to 2^Y$. Therefore, a play $\rho = (s_0, X_0 \cup g(X_0)), (s_1, X_1 \cup g(X_0, X_1)), \ldots$ over $G_\phi$ from $s_0$ following $g$ is a winning play for the agent. Moreover, for every such play $\rho$ from $s$, either of the following conditions holds:

- $\rho \not\models FG\alpha$ such that $\rho$ is not $\alpha$-stable.
- $\rho \models FG\alpha$ such that $\rho$ is $\alpha$-stable. Since $\rho$ is winning for the agent, there exists $j \geq 0$ such that $s_j \in Acc$. Therefore, $\rho^j \models \phi$ holds, where $\rho^j = (s_0, X_0 \cup g(X_0)), (s_1, X_1 \cup g(X_0, X_1)), \ldots$ satisfying $s_j \in Acc$.

Consequently, the strategy $g$ assures that for an arbitrary environment trace $\lambda = X_0, X_1, \ldots \in (2^X)^\omega$, if $\lambda$ is $\alpha$-stable, then there exists $j \geq 0$ such that $\phi$ is true in finite trace $\rho^j$. Thus $\langle X, Y, \alpha, \phi \rangle$ is realizable.

$(\Leftarrow)$ For this direction, we assume that $\langle X, Y, \alpha, \phi \rangle$ is realizable, then there exists a strategy $g : (2^X)^+ \to 2^Y$ that realizes $\phi$. Thus consider an arbitrary environment trace $\lambda \in (2^X)^\omega$, either of the following conditions holds:

- $\lambda$ is not $\alpha$-stable, then the induced play $\rho = (s_0, X_0 \cup g(X_0)), (s_1, X_1 \cup g(X_0, X_1)), \ldots$ over $G_\phi$ from $s_0$ follows $g$ is winning for the agent by default.
- $\lambda$ is $\alpha$-stable, then on the induced play $\rho = (s_0, X_0 \cup g(X_0)), (s_1, X_1 \cup g(X_0, X_1)), \ldots$ over $G_\phi$ from $s_0$, there exists $j \geq 0$ such that $\phi$ is true in the finite trace $\rho^j = (s_0, X_0 \cup g(X_0)), (s_1, X_1 \cup g(X_0, X_1)), \ldots$ in which case $s_j \in Acc$. Therefore, $\rho$ is winning for the agent.

Consequently, we conclude that stable DFA game $\langle G_\phi, \alpha \rangle$ is realizable for the agent.

**Stable DFA Game Solving**

Despite the duality between fairness and stability, solving the stable DFA game here cannot directly dualize the solution to fair DFA game. This is because the computation here involves a Stability-Safety game, which is not dual to the Recurrence-Safety game in fair DFA game solving. In order to deal with stable DFA game, we again first consider the environment as the protagonist. We compute the set of environment winning states as follows:

$\text{Env}_{st} = \mu Z. \nu \hat{Z}. \forall X. \forall Y. ((X \models \alpha \land \delta(s, X \cup Y) \in Z \land Acc) \land \delta(s, X \cup Y) \in \text{Sys}_{st} \land Acc)$

Define an output function $\omega_f : \text{Sys}_{st} \times 2^X \to 2^Y$ as follows: for $s \in Z_{i+1} \setminus Z_i$, for all possible values $X \in 2^X$, set $Y$ to be such that $(X \models \neg \alpha \land \delta(s, X \cup Y) \in Z_i \land Acc) \land \delta(s, X \cup Y) \in \text{Sys}_{st} \land Acc$ holds for $s \not\in Acc$. Consider a deterministic finite transducer $T$ defined in the sense that constructing $\omega_f$ as described above, the following theorem guarantees that $T$ generates an agent winning strategy $g$.

**Theorem 6.** For a stable DFA game $\langle G, \alpha \rangle$ and a state $s \in S$, we have $s \in \text{Env}_{st}$ iff $s$ is an environment winning state.

Correspondingly, since stable DFA game is determined, the set of agent winning states can be computed as follows:

$\text{Sys}_{st} = \nu Z. \mu \hat{Z}. \forall X. \forall Y. ((X \models \alpha \land \delta(s, X \cup Y) \in Z \land Acc) \land \delta(s, X \cup Y) \in Z \land Acc)$. 

**Theorem 7.** A stable DFA game $\langle G, \alpha \rangle$ has an agent winning strategy if and only if $s_0 \in \text{Sys}_{st}$.

**Strategy Extraction**

Here, the agent winning strategy $g : (2^X)^+ \to 2^Y$ can also be represented as a deterministic finite transducer $T = (2^X, 2^Y, Q, s_0, \rho, \omega_f)$ in terms of the set of agent winning states such that $Q = \text{Sys}_{st}$.

We extract the output function $\omega_f : Q \times 2^X \to 2^Y$ for the game from the approximates for $Z$ approximating $\hat{Z}$ to be $\text{Sys}_{st}$, from where no matter what the environment strategy is, traces cannot always get $\alpha$ hold. Thus, we consider the fixpoint computation as follows:
We observe that a straightforward approach to \(\text{strategy for the agent.}
\)

LTL standard synthesis under assumptions can be obtained by a reduction to standard lemma. In this section, we first revisit the reduction to standard synthesis, which allows us to utilize tools for LTL synthesis to solve the fair (or stable) LTL synthesis problem. In this section, we first revisit the reduction to standard LTL synthesis, and then show an experimental comparison with the approach proposed earlier in this paper.

**Reduction to LTL Synthesis.** The insight of reducing LTL synthesis under assumptions comes from the reduction in (Zhu et al. 2017b) for general LTL synthesis, and in (Camacho, Bienvenu, and McIlraith 2018) for constraint LTL synthesis, where the constraint describes the desired environment behaviors, under which the goal is to satisfy the given LTL specification. Both reductions adopt the translation rules in (De Giacomo and Vardi 2013) to polynomially transform an LTL formula \(\phi\) over \(X \cup Y\) into an LTL formula \(\psi\) over \(X' \cup Y' \cup \{\text{alive}\}\), retaining the satisfiability equivalence, where proposition \(\text{alive}\) indicates the last instance of the finite trace. Such translation bridges the gap between LTL over finite traces and LTL over infinite traces. Based on the translation from LTL to LTL, we then reduce fair (resp., stable) LTL synthesis problem \(\langle X, Y, \alpha, \phi \rangle\) to LTL synthesis problem \(\langle X', Y' \cup \{\text{alive}\}, G\alpha \rightarrow \psi \rangle\) (resp., \(\langle X, Y \cup \{\text{alive}\}, F\alpha \rightarrow \psi \rangle\)).

**Implementation.** Based on the LTL synthesis tool Syft, we implemented our fixpoint-based techniques for solving fair LTL synthesis and stable LTL synthesis in two tools called FSyft and StSyft, respectively (name after Syft). Both frameworks consist of two steps: the symbolic DFA construction and the respective DFA game solving. In the first step, we based on the code of Syft, to construct the symbolic DFA represented in Binary Decision Diagrams (BDDs). The implementation of the nested fixpoint computation for solving DFA games over such symbolic DFA, borrows techniques from (Zhu et al. 2017a) for greatest fixpoint computation and from Syft for least fixpoint computation. The construction of the transducer for generating the winning strategy utilizes the boolean-synthesis procedure introduced in (Fried, Tabajara, and Vardi 2016) for realizeable formulas. The implementation makes use of the BDD library CUDD-3.0.0 (Somenzi 2016). In order to evaluate the performance of FSyft and StSyft, we compared it against the solution of reducing to standard LTL synthesis shown above. For such comparison, we employed the LTL-to-LTL translator implemented in SPOT (Duret-Lutz et al. 2016) and chose

\[\nu Z. (\forall X. \exists Y. ((X = \neg \alpha \lor \delta(s, X \cup Y) \in Sys_{st} \cup Acc) \land \delta(s, X \cup Y) \in Z \cup Acc)).\]

Define an output function \(\omega_{st} : Sys_{st} \times 2^X \rightarrow 2^X\) s.t. for \(s \in Z_{i+1} \cap Z_i\), for all possible values \(X \in 2^X\), set \(Y\) to be s.t. \((X = \neg \alpha \lor \delta(s, X \cup Y) \in Sys_{st} \cup Acc) \land \delta(s, X \cup Y) \in Z_i \cup Acc\) holds for \(s \notin Acc\). The following theorem guarantees that \(T\) generates an agent winning strategy \(g\).

**Theorem 8.** Strategy \(g\) with \(g(\lambda) = \omega_{st}(g(\lambda))\) is a winning strategy for the agent.

**Evaluation**

We observe that a straightforward approach to LTL synthesis under assumptions can be obtained by a reduction to standard LTL synthesis. In this section, we first revisit the reduction to standard LTL synthesis, and then show an experimental comparison with the approach proposed earlier in this paper.

**Experimental Methodology**

**Benchmarks.** We collected 1200 formulas consisting of two classes of benchmarks: 1000 randomly conjuncted LTL formulas over 100 basic cases, generated in the style described in (Zhu et al. 2017b), the length of which, indicating the number of conjuncts, ranges from 1 to 5. The assumption (either fairness or stability) is assigned by randomly selecting one variable from all environment variables; 200 LTL synthesis benchmarks with assumptions generated from a scalable counter game, described as follows:

- There is an \(n\)-bit binary counter. At each round, the environment chooses whether to increment the counter or not. The agent can choose to grant the request or ignore it.
- The goal is to get the counter having all bits set to 1, so the counter reaches the maximal value.
- The fairness assumption is to have the environment infinitely request the counter to be incremented.
- The stability assumption is to have the environment eventually keep requesting the counter to be incremented.

We reduce solving the counter game above to solving LTL synthesis with assumptions. First, we have \(n\) agent variables \(\{b_{n-1}, b_{n-2}, \ldots, b_0\}\) denoting the value of \(n\) counter bits. We also introduce another \(n + 1\) agent variables \(\{c_n, c_{n-1}, \ldots, c_0\}\) representing the carry bits. In addition, we have an environment variable \(\text{add}\) representing the environment making an increment request or not, and \(c_0\) as true is considered as the agent granting the request. We then formulate the counter game into LTL formula as follows:

\[
\begin{align*}
\text{Init} &= ((\neg c_0) \land \ldots \land (\neg c_{n-1}) \land (\neg b_0) \land \ldots \land (\neg b_{n-1})), \\
\text{Goal} &= F(b_0 \land \ldots \land b_{n-1}), \\
B &= G((\neg \text{add} \rightarrow X_w(\neg c_0)), \\
&\quad \left\{\begin{array}{l}
((c_i \land \neg b_i) \rightarrow X_w(b_i \land \neg c_{i+1})) \\
((\neg c_i) \land b_i) \rightarrow X_w(b_i \land \neg c_{i+1}))
\end{array}\right.
\end{align*}
\]

The LTL formula \(\phi\) is then \((\text{Init} \land B \land \bigwedge_{0 \leq i < n} G(B_i)) \land \text{Goal}\), and the constraint \(\alpha\) is \text{add}. Obviously, such counter game only returns realizable cases, since a winning strategy for the agent is to grant all increment requests.

In order to get unrealizable cases, we can make some modifications on the counter game above. One possibility is to have the counter increment by 2 if the agent chooses to grant the request sent by the environment. Such modification leads to no winning strategy for the agent, since the maximal counter value of having each bit as 1 is odd. However, incrementing by 2 at each time will never reach an odd value. Therefore, for bit \(B_i\) such that \(i > 0\), we keep the same formulation. While for bit \(B_0\), we change as follows:

\[\text{Init} = ((\neg c_0) \land \ldots \land (\neg c_{n-1}) \land (\neg b_0) \land \ldots \land (\neg b_{n-1})), \]

\[\text{Goal} = F(b_0 \land \ldots \land b_{n-1}), \]

\[B = G((\neg \text{add} \rightarrow X_w(\neg c_0)), \]

\[\left\{\begin{array}{l}
((c_i \land \neg b_i) \rightarrow X_w(b_i \land \neg c_{i+1})) \\
((\neg c_i) \land b_i) \rightarrow X_w(b_i \land \neg c_{i+1}))
\end{array}\right.\]
Figure 1: Fair LTLₙ synthesis. Comparison of the number of solved cases with limited time between FSyft and Strix over random conjunction benchmarks.

Figure 2: Stable LTLₙ synthesis. Comparison of the number of solved cases with limited time between StSyft and Strix over random conjunction benchmarks.

![Graph showing the number of solved cases vs time for FSyft and Strix](image1)

![Graph showing the number of solved cases vs time for StSyft and Strix](image2)

Figure 3: Fair LTLₙ synthesis. Comparison of running time between FSyft and Strix, in log scale. Bars of the maximum height indicate cases timed out.

Figure 4: Stable LTLₙ synthesis. Comparison of running time between StSyft and Strix, in log scale. Bars of the maximum height indicate cases timed out.

In this paper we presented a fixpoint-based technique for LTLₙ synthesis with assumptions for basic forms of fairness and stability, which is quite effective, as our experiment shows. Our technique can be summarized as follows: use the DFA for the LTLₙ formula as the arena to play a game for the environment whose winning condition is to avoid reaching the accepting states while making the assumption true. Note that for a general LTL assumption (see (Aminof et al. 2019)), we can transform such an assumption into a parity automaton, take the Cartesian product with the DFA and play the parity/reachability game over the resulting arena. Comparing this possible approach to the reduction to LTL synthesis is a subject for future work.

Acknowledgments. Work supported in part by European
Figure 4: Stable LTLf synthesis. Comparison of running time between StSyft and Strix, in log scale. Bars of the maximum height indicate cases timed out.

Research Council under the European Unions Horizon 2020 Programme through the ERC Advanced Grant WhiteMec (No. 834228), NSF grants IIS-1527668, CCF-1704883, and IIS-1830549, NSFC Projects No. 61572197, No. 61632005 and No. 61532019.

References
[Aminof et al. 2018] Aminof, B.; De Giacomo, G.; Murano, A.; and Rubin, S. 2018. Synthesis under Assumptions. In KR, 615–616.
[Aminof et al. 2019] Aminof, B.; De Giacomo, G.; Murano, A.; and Rubin, S. 2019. Planning under LTL Environment Specifications. In ICAPS.
[Bloom et al. 2012] Bloom, R.; Jobstmann, B.; Piterman, N.; Puelu, A.; and Sa’ar, Y. 2012. Synthesis of Reactive(1) Designs. J. Comput. Syst. Sci. 78(3):911–938.
[Bloom, Ehlers, and Könighofer 2015] Bloom, R.; Ehlers, R.; and Könighofer, R. 2015. Cooperative Reactive Synthesis. In ATVA, volume 9364 of Lecture Notes in Computer Science, 394–410. Springer.
[Brenguier, Raskin, and Sankur 2017] Brenguier, R.; Raskin, J.; and Sankur, O. 2017. Assume-admissible synthesis. Acta Inf. 54(1):41–83.
[Buchi and Landweber 1990] Buchi, J. R., and Landweber, L. H. 1990. Solving Sequential Conditions by Finite-State Strategies.
[Camacho et al. 2017] Camacho, A.; Triantafillou, E.; Muise, C.; Baier, J. A.; and McIrlraith, S. 2017. Non-Deterministic Planning with Temporally Extended Goals: LTL over Finite and Infinite Traces. In AAAI.
[Camacho et al. 2018] Camacho, A.; Baier, J. A.; Muise, C. J.; and McIrlraith, S. A. 2018. Finite LTL Synthesis as Planning. In ICAPS, 29–38.
[Camacho, Bienvenu, and McIrlraith 2018] Camacho, A.; Bienvenu, M.; and McIrlraith, S. A. 2018. Finite LTL Synthesis with Environment Assumptions and Quality Measures. In KR, 454–463.
[Chatterjee and Henzinger 2007] Chatterjee, K., and Henzinger, T. A. 2007. Assume-guarantee synthesis. In TACAS, 261–275.
[Chatterjee, Henzinger, and Jobstmann 2008] Chatterjee, K.; Henzinger, T. A.; and Jobstmann, B. 2008. Environment Assumptions for Synthesis. In CONCUR, 147–161.
[De Giacomo and Rubin 2018] De Giacomo, G., and Rubin, S. 2018. Automata-Theoretic Foundations of FOND Planning for LTL/F/ LDL Goals. In IJCAI, 4729–4735.
[De Giacomo and Vardi 2013] De Giacomo, G., and Vardi, M. 2013. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In IJCAI, 854–860.
[De Giacomo and Vardi 2015] De Giacomo, G., and Vardi, M. Y. 2015. Synthesis for LTL and LDL on Finite Traces. In IJCAI, 1558–1564.
[D’Ippolito et al. 2013] D’Ippolito, N.; Braberman, V. A.; Piterman, N.; and Uchitel, S. 2013. Synthesizing nonanomalous event-based controllers for liveness goals. ACM Trans. Softw. Eng. Methodol. 22(1):9:1–9:36.
[Duret-Lutz et al. 2016] Duret-Lutz, A.; Lewkowicz, A.; Fauchille, A.; Michaud, T.; Renault, E.; and Xu, L. 2016. Spot 2.0 — a framework for LTL and ω-automata manipulation. In ATVA, 122–129.
[Finkbeiner and Schewe 2013] Finkbeiner, B., and Schewe, S. 2013. Bounded Synthesis. STTT 15(5-6):519–539.
[Finkbeiner 2016] Finkbeiner, B. 2016. Synthesis of Reactive Systems. Dependable Software Systems Eng. 45:72–98.
[Fogarty et al. 2013] Fogarty, S.; Kupferman, O.; Vardi, M. Y.; and Wilke, T. 2013. Profile Trees for Büchi Word Automata, with Application to Determinization. In GandALF.
[Fried, Tabajara, and Vardi 2016] Fried, D.; Tabajara, L. M.; and Vardi, M. Y. 2016. BDD-Based Boolean Functional Synthesis. In CAV.
[Gerstacker, Klein, and Finkbeiner 2018] Gerstacker, C.; Klein, F.; and Finkbeiner, B. 2018. Bounded Synthesis of Reactive Programs. In ATVA, 441–457.
[Gradel, Thomas, and Wilke 2002] Grädel, E.; Thomas, W.; and Wilke, T., eds. 2002. Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001], volume 2500 of Lecture Notes in Computer Science. Springer.
[Green 1969] Green, C. 1969. Theorem Proving by Resolution as Basis for Question-Answering Systems. In Machine Intelligence, volume 4. American Elsevier. 183–205.
[Grädel, Thomas, and Wilke 2002] Grädel, E.; Thomas, W.; and Wilke, T., eds. 2002. Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001], volume 2500 of Lecture Notes in Computer Science. Springer.
[Green 1969] Green, C. 1969. Theorem Proving by Resolution as Basis for Question-Answering Systems. In Machine Intelligence, volume 4. American Elsevier. 183–205.
[Kupferman and Vardi 2005] Kupferman, O., and Vardi, M. Y. 2005. Safraless Decision Procedures. In FOCS, 531–542. IEEE Computer Society.
[Li et al. 2019] Li, J.; Rozier, K. Y.; Pu, G.; Zhang, Y.; and Vardi, M. Y. 2019. Sat-based explicit ltlf satisfiability checking. In AAAI, 2946–2953.
[Martin 1975] Martin, D. 1975. Borel Determinacy. Annals of Mathematics 65:363–371.
[Meyer, Sickert, and Luttenberger 2018] Meyer, P. J.; Sickert, S.; and Luttenberger, M. 2018. Strix: Explicit Reactive Synthesis Strikes Back! In CAV, 578–586.
Appendix

Due to the lack of space, we move some proofs and the details of the reduction from fair LTL$_f$ synthesis and stable LTL$_f$ synthesis to standard LTL synthesis in this appendix.

For better readability, we redefine two-players DFA games here. Two-player games on DFA are games consisting of two players, the environment and the agent. $X$ and $\gamma$ are disjoint sets of environment Boolean variables and agent Boolean variables, respectively. The specification of the game arena is given by a DFA $\mathcal{G} = (2^{X \cup \gamma}, S, s_0, \delta, Acc)$, where

- $2^{X \cup \gamma}$ is the alphabet;
- $S$ is a set of states;
- $s_0 \in S$ is an initial state;
- $\delta : S \times 2^{X \cup \gamma} \rightarrow \gamma$ is a transition function;
- $Acc \subseteq S$ is a set of accepting states.

Here, we consider two specific two-player games, fair DFA game and stable DFA game, both of which are described as $\langle \mathcal{G}, \alpha \rangle$, where $\mathcal{G}$ is the game arena and $\alpha$ is the environment constraint, which is a Boolean formula over $X$.

Fair LTL$_f$ Synthesis

Due to the determinacy of fair DFA game, the set of agent winning states $\text{Sys}_f$ can be computed by negating $\text{Env}_f$:

$$\text{Sys}_f = \mu Z. \nu \hat{Z}. (\forall X. \exists Y. (X \models \alpha \land \delta(s, X \cup Y) \in \hat{Z} \land Acc) \lor \delta(s, X \cup Y) \in \hat{Z} \land Acc)).$$

The following theorem guarantees the correctness of the set of agent winning states computation $\text{Sys}_f$.

**Theorem 11.** A fair DFA game $\langle \mathcal{G}, \alpha \rangle$ has an agent winning strategy if and only if $s_0 \in \text{Sys}_f$.

**Proof.** Since $\text{Sys}_f$ is the dual formula of $\text{Env}_f$, for a state $s \in S$, we have $s \in \text{Sys}_f$ if and only if $s \notin \text{Env}_f$ such that $s$ is not a winning state for the environment, in which case $s$ is an agent winning state with winning strategy $g : (2^X)^+ \rightarrow 2^Y$. Therefore, for a state $s \in S$, we have $s \in \text{Sys}_f$ if and only if $s$ is an agent winning state. Moreover, fair DFA game is realizable if and only if the initial state $s_0$ is an agent winning state. Consequently, we conclude that fair DFA game $\langle \mathcal{G}, \alpha \rangle$ is realizable with agent winning strategy $g$ if and only if $s_0 \in \text{Sys}_f$.

Define an output function $\omega_f : \text{Sys}_f \times 2^X \rightarrow 2^Y$ as follows: for $s \in Z_{i+1} \setminus Z_i$, for all possible values $X \in 2^X$, set $Y$ to be such that $(X \models \neg \alpha \lor \delta(s, X \cup Y) \in Z_i \land Acc) \land \delta(s, X \cup Y) \in \text{Sys}_f \land Acc$ holds for $s \notin Acc$. Consider a deterministic finite transducer $T$ defined in the sense that constructing $\omega_f$ so as described above, the following theorem guarantees that $T$ generates an agent winning strategy $g$. The following theorem guarantees that deterministic finite transducer $T$ is able to generate a winning strategy $g$ for the agent.

**Theorem 12.** Strategy $g$ with $g(\lambda) = \omega_f(g(\lambda))$ is a winning strategy for the agent.

**Proof.** Consider an arbitrary environment trace $\lambda = (X_0, X_1, \ldots) \in (2^X)^\omega$, the corresponding play over $\gamma$ that follows $\rho$ is $\rho = (s_0, X_0 \lor g(X_0)), (s_1, X_1 \lor g(X_0, X_1)), \ldots$. We now prove that $\rho$ is a winning play for the agent. For every state $s$ along the play $\rho$, the construction of $\omega_f$ ensures that, no matter how the environment sets $X$, $\omega_f$ returns $Y$ such that $(X \models \neg \alpha \lor \delta(s, X \cup Y) \in Z_i \land Acc) \lor \delta(s, X \cup Y) \in \text{Sys}_f \land Acc$ holds. Thus we either have $\omega_f$ holds, or $\rho$ visits $Z_i \land Acc$. At the same time, $\rho$ keeps in $\text{Sys}_f \land Acc$. The first possibility keeps the stability condition and the latter one retains the reachability condition, by inductive hypothesis, both of them give $\rho$ a winning play. Therefore, $g$ is a winning strategy for the agent.

Stable LTL$_f$ Synthesis

In stable DFA game $\langle \mathcal{G}, \alpha \rangle$, we compute the set of environment winning states as follows:

$$\text{Env}_{st} = \mu Z. \nu \hat{Z}. (\exists X. \forall Y. ((X \models \alpha \land \delta(s, X \cup Y) \in \hat{Z} \land Acc) \lor \delta(s, X \cup Y) \in \hat{Z} \land Acc),$$

where $X$ ranges over $2^X$ and $Y$ over $2^Y$.

The fixpoint stages for $\hat{Z}$ (note $\hat{Z}_i \subseteq \hat{Z}_{i+1}$, for $i \geq 0$, by monotonicity) are:

- $Z_0 = \emptyset$,
- $Z_{i+1} = \nu \hat{Z}. (\exists X. \forall Y. ((X \models \alpha \land \delta(s, X \cup Y) \in \hat{Z} \land Acc) \lor \delta(s, X \cup Y) \in Z_{i+1} \land Acc))$.

Eventually, $\text{Env}_{st} = Z_k$ for some $k$ such that $Z_{k+1} = Z_k$.

The fixpoint stages for $\hat{Z}$ with respect to $Z_i$ (note $\hat{Z}_{i+1} \subseteq \hat{Z}_i$, for $j \geq 0$, by monotonicity) are:

- $\hat{Z}_{i,0} = S$,
- $\hat{Z}_{i,j+1} = \exists X. \forall Y. ((X \models \alpha \land \delta(s, X \cup Y) \in \hat{Z}_{i,j} \land Acc) \lor \delta(s, X \cup Y) \in Z_{i+1} \land Acc)$.

Finally, $\hat{Z}_i = \hat{Z}_{i,k}$ for some $k$ such that $\hat{Z}_{i,k+1} = \hat{Z}_{i,k}$.

The following theorem asserts that the nested fixpoint computation of $\text{Env}_{st}$ collects exactly all environment winning states in stable DFA game.

**Theorem 14.** For a stable DFA game $\langle \mathcal{G}, \alpha \rangle$ and a state $s \in S$, we have $s \in \text{Env}_{st}$ if and only if $s$ is an environment winning state.

**Proof.** We prove the theorem in both directions.

$\leftarrow$: We proceed the proof by showing the contrapositive. A state $s \notin \text{Env}_{st}$ indicates that $s$ cannot be added to $Z_{i+1}$ at stage $i + 1$ for all $i \geq 0$. Then $s \notin \nu \hat{Z}. (\exists X. \forall Y. ((X \models \alpha \land \delta(s, X \cup Y) \in \hat{Z}_{i+1} \land Acc) \lor \delta(s, X \cup Y) \in Z_{i+1} \land Acc))$.
Thus \( \alpha \land \delta(s, X \cup Y) \in \hat{Z} \setminus \text{Acc} \lor \delta(s, X \cup Y) \in Z \setminus \text{Acc} \). That is, no matter what the (environment) strategy \( \beta \) is, traces from \( s \) satisfy \textbf{neither} of the following conditions:

- \( \alpha \) holds and the trace gets trapped in \( \hat{Z} \) without visiting any accepting states such that \( X \models \alpha \land \delta(s, X \cup Y) \in \hat{Z} \setminus \text{Acc} \), in which case, \( S \) is a new environment winning state;
- one already defined environment winning state gets visited such that \( \delta(s, X \cup Y) \in Z \setminus \text{Acc} \), in which case, \( S \) is a new environment winning state.

Therefore, \( S \) is not an environment winning state. So if \( S \) is an environment winning state then \( s \in \mathcal{E}_{\text{env}} \) holds.

\( \Rightarrow \) If a state \( s \in \mathcal{E}_{\text{env}} \), then \( s \in \nu \hat{Z}.(\exists X.\forall Y .((X \models \alpha \land \exists Y .(X \models \alpha \land \delta(s, X \cup Y))) \lor \delta(s, X \cup Y) \in Z \setminus \text{Acc})) \). That is, no matter what the (system) strategy \( g \) is, traces from \( s \) satisfy either of the following conditions:

- \( \alpha \) holds and the trace gets trapped in \( \hat{Z} \) without visiting any accepting states such that \( X \models \alpha \land \delta(s, X \cup Y) \in \hat{Z} \setminus \text{Acc} \), in which case, \( S \) is a new environment winning state;
- one already defined environment winning state gets visited such that \( \delta(s, X \cup Y) \in Z \setminus \text{Acc} \), in which case, \( S \) is a new environment winning state.

Thus \( S \) is a winning state for the environment. \( \square \)

In stable DFA game \( \langle G, \alpha \rangle \), the set of agent winning states can be computed as follows:

\[
\text{Sys}_S = \nu Z.\mu \hat{Z}.(\forall X.\exists Y .((X \models \neg \alpha \lor \exists Y .(X \models \neg \alpha \lor \delta(s, X \cup Y))) \land \delta(s, X \cup Y) \in Z \cup \text{Acc}))
\]

**Theorem 15.** A stable DFA game \( \langle G, \alpha \rangle \) has an agent winning strategy if and only if \( s_0 \in \text{Sys}_S \).

**Proof.** Since \( \text{Sys}_S \) is the dual formula of \( \mathcal{E}_{\text{env}} \), for a state \( s \in S \), we have \( s \in \text{Sys}_S \) if and only if \( s \notin \mathcal{E}_{\text{env}} \) such that \( s \) is not a winning state for the environment, in which case \( s \) is an agent winning state with winning strategy \( g : (2^X)^+ \rightarrow 2^Y \). Therefore, for a state \( s \in S \), we have \( s \in \text{Sys}_S \) if and only if \( s \) is an agent winning state. Moreover, stable DFA game is realizable if and only if the initial state \( s_0 \) is an agent winning state. Consequently, we conclude that stable DFA game \( \langle G, \alpha \rangle \) is realizable with agent winning strategy \( g \) if and only if \( s_0 \in \text{Sys}_S \). \( \square \)

We extract the output function \( \omega_\alpha : S \times 2^X \rightarrow 2^Y \) for the game from the approximates for \( Z \) assuming \( \hat{Z} \) to be \( \text{Sys}_S \), from where no matter what the environment strategy is, traces cannot always get \( \alpha \) hold. Thus, we consider the fixpoint computation as follows:

\[
\nu Z.(\forall X.\exists Y .((X \models \neg \alpha \lor \exists Y .(X \models \neg \alpha \lor \delta(s, X \cup Y) \in S \cup \text{Acc}) \land \delta(s, X \cup Y) \in Z \cup \text{Acc}))
\]

with approximates defined as:

- \( Z_0 = S \)
- \( Z_{i+1} = \forall X.\exists Y .((X \models \neg \alpha \lor \delta(s, X \cup Y) \in S \cup \text{Acc}) \land \delta(s, X \cup Y) \in Z_i \cup \text{Acc}) \)

Define an output function \( \omega_\alpha : S \times 2^X \rightarrow 2^Y \) as follows: for \( s \in Z_{i+1} \cap Z_i \), for all possible values \( X \in 2^X \), set \( Y \) to be such that \( (X \models \neg \alpha \lor \delta(s, X \cup Y) \in S \cup \text{Acc}) \lor \delta(s, X \cup Y) \in Z \cup \text{Acc} \) holds for \( s \notin \text{Acc} \). Consider a deterministic finite transducer \( \mathcal{T} \) defined in the sense that constructing \( \omega_\alpha \) so as described above, the following theorem guarantees that \( \mathcal{T} \) generates an agent winning strategy \( g \).

**Theorem 16.** Strategy \( g \) with \( g(\lambda) = \omega_\alpha(\rho(\lambda)) \) is a winning strategy for the agent.

**Proof.** Consider an arbitrary environment trace \( \lambda = X_0, X_1, \ldots \in (2^X)^\omega \), the corresponding play over \( \mathcal{G} \) that follows \( g \) is \( \rho = (s_0, X_0 \cup g(X_0)), (s_1, X_1 \cup g(X_0, X_1)), \ldots \)

We now prove that \( \rho \) is a winning play for the agent. For every state \( s \) along the play \( \rho \), the construction of \( \omega_\alpha \) ensures that, no matter how the environment sets \( X \), \( \omega_\alpha \) returns \( Y \) such that \( s \in (X \models \neg \alpha \lor \delta(s, X \cup Y) \in S \cup \text{Acc}) \lor \delta(s, X \cup Y) \in Z \cup \text{Acc} \) holds. Thus we either have \( \neg \alpha \) holds, or \( \rho \) stays in \( S \cup \text{Acc} \). At the same time, \( \rho \) visits \( Z \cup \text{Acc} \). The first possibility keeps the recurrence condition and the latter one remains the reachability condition, by inductive hypothesis, both of them give \( \rho \) a winning play. Therefore, \( g \) is a winning strategy for the agent. \( \square \)

**Reduction to LTL Synthesis**

In addition to the fixpoint-based automata theoretical solution, an alternative approach to fair LTLf synthesis and stable LTLf synthesis can be obtained by a reduction to standard LTL synthesis.

**Definition 13 (LTL Synthesis).** Let \( \psi \) be an LTL formula over an alphabet \( P \) and \( X, Y \) be two disjoint atom sets such that \( X \cup Y = P \), \( \psi \) is realizable with respect to \( X, Y \) if there exists a strategy \( f : (2^X)^+ \rightarrow 2^Y \), such that for an arbitrary infinite sequence \( \lambda = X_0, X_1, \ldots \in (2^X)^\omega \), \( \psi \) is true in the infinite trace \( \rho = (X_0 \cup f(X_0)), (X_1 \cup f(X_0, X_1)), (X_2 \cup f(X_0, X_1, X_2)), \ldots \). The synthesis procedure is to compute such a strategy if \( \psi \) is realizable.

Reducing fair or stable LTLf synthesis to LTL synthesis allows tools for general LTL synthesis to be used in solving fair LTLf synthesis and stable LTLf synthesis. The reduction adopts the translation rules in [De Giacomo and Vardi 2013] to polynomially transform an LTLf formula \( \phi \) over propositions \( X \cup Y \) to LTL formula \( \psi \) over \( X \cup Y \cup \{\text{alive}\} \) by introducing a new variable \( \text{alive} \). As such, we have \( \phi \) is satisfiable if and only if \( \psi \) is satisfiable. The translation requires a function \( t \) that reads an LTLf formula and returns an LTL formula, which is defined as follows:

- \( t(a) = a \)
- \( t(\neg \phi_1) = \neg t(\phi_1) \)
- \( t(\phi_1 \land \phi_2) = t(\phi_1) \land t(\phi_2) \)
- \( t(X \phi) = X(\text{alive} \land t(\phi)) \)
- \( t(\phi_1 U \phi_2) = t(\phi_1)U(\text{alive} \land t(\phi_2)) \)

Finally, \( \psi = t(\phi) \land \text{alive} \land (\text{alive} \ U \text{(G-alive)}) \). Since the proof of the satisfiability equivalence between \( \phi \) and \( \psi \) is implicit in [De Giacomo and Vardi 2013], we show here the
Lemma 1. Let $\phi$ be an LTL$_f$ formula, $\psi$ be the corresponding translated LTL formula, and $\tau$ be a finite trace, $\tau'$ be an infinite trace with $\tau \approx \tau'$. Then $\tau \models \phi$ iff $\tau' \models \psi$ is true.

Proof. Since $|\tau| > 0$, it is straightforward to show that $\tau' \models alive \land (alive U (G\neg alive))$ since $\tau \approx \tau'$. Now we prove the lemma by a constructive induction of $\phi$.

- If $\phi = a$, then $\psi = a \land alive \land (alive U (G\neg alive))$, $\tau \models a$ such that $\tau[0] = a$, in which case $\tau'[0] = a$ such that $\tau' \models a$. Therefore, $\tau' \models \psi$.

- If $\phi = \neg \phi_1$, then $\psi = \neg \tau(\phi_1) \land alive \land (alive U (G\neg alive))$, $\tau \models \phi$ such that $\tau \models \phi_1$ holds. By induction hypothesis, $\tau' \not\models t(\phi_1)$ such that $\tau' \models \neg t(\phi_1)$. Therefore, $\tau' \models \psi$.

- If $\phi = \phi_1 \land \phi_2$, then $\psi = t(\phi_1) \land t(\phi_2) \land alive \land (alive U (G\neg alive))$, $\tau \models \phi_1 \land \phi_2$ such that $\tau \models \phi_1$ and $\tau \models \phi_2$ hold. By induction hypothesis, $\tau' \models t(\phi_1)$ and $\tau' \models t(\phi_2)$ hold such that $\tau' \models t(\phi_1) \land t(\phi_2)$. Therefore, $\tau' \models \psi$.

- If $\phi = X\phi_1$, then $\psi = X(alive \land t(\phi_1)) \land alive \land (alive U (G\neg alive))$, $\tau \models alive \land t(\phi_1)$ such that $\tau \models \phi_1$. By induction hypothesis, $\tau' \models alive \land t(\phi_1)$ such that $\tau' \models X(alive \land t(\phi_1))$. Therefore, $\tau' \models \psi$.

- If $\phi = \phi_1 U \phi_2$, then $\psi = t(\phi_1) U (alive \land t(\phi_2)) \land alive \land (alive U (G\neg alive)) \land alive \land (alive U (G\neg alive))$, $\tau \models \phi$ such that there exists $j \geq 0$ such that $\tau_j = \phi_2$, and for $0 \leq i < j$, we have $\tau_i = \phi_1$. By induction hypothesis, $\tau'_i \models alive \land t(\phi_2)$ and $\tau'_i \models t(\phi_1)$ hold for $0 \leq i < j$, this $\tau' \models t(\phi_1) U (alive \land t(\phi_2))$. Therefore, $\tau' \models \psi$.

To show the reduction from fair LTL$_f$ synthesis to LTL synthesis, we start by assigning the environment and agent variables. Intuitively, alive is a signal whose failure indicates the end of the finite trace. Therefore, alive is assigned as an agent variable such that the agent can keep setting alive as true until alive is set to false when $\phi$ is satisfied. The environment constraint $\alpha$ over environment variables in both of fair LTL$_f$ synthesis and stable LTL$_f$ synthesis is a condition for the satisfaction of the desired goal $\phi$. Since being realizable requires $\phi$ to be satisfied under the condition such that $\alpha$ holds infinitely often for fair LTL$_f$ synthesis and eventually holds forever for stable LTL$_f$ synthesis, we obtain the LTL goal $G\phi \to \psi$ and $FG\alpha \to \psi$, respectively. In both cases, $\psi$ is the corresponding translated LTL formula of $\phi$. Thus solving the problem $\langle X, Y, \alpha, \phi \rangle$ is reduced to solving the LTL synthesis problem $\langle X, Y \cup \{alive\}, G\phi \to \psi \rangle$ for fair LTL$_f$ synthesis, and to $\langle X, Y \cup \{alive\}, FG\alpha \to \psi \rangle$ for stable LTL$_f$ synthesis. The following theorems guarantee the correctness of this reduction respectively.

Theorem 17. Let $\phi$ be an LTL$_f$ formula, $\psi$ be the corresponding translated LTL formula, then fair LTL$_f$ synthesis problem $\langle X, Y, \alpha, \phi \rangle$ is realizable if and only if LTL formula $G\alpha \to \psi$ is realizable with respect to $\langle X, Y \cup \{alive\} \rangle$.

Proof. We prove the two directions separately.

- $\leftarrow$: Since $G\phi \to \psi$ is realizable with respect to $\langle X, Y \cup \{alive\} \rangle$, there exists a winning strategy $g'$ : $(2^{2^{|\lambda|} \cup \{alive\}})^+ \to 2^{2^{|\lambda|} \cup \{alive\}}$ such that every trace $\rho'$ that follows $g'$ gets $G\alpha \to \psi$ hold, therefore, enabling either of the following situations:
  
  - $G\alpha$ is true such that the environment behaves such as having a hold infinitely often, and $\psi$ holds. Therefore, $\rho' \models alive \land (alive U (G\neg alive))$, and there exists a position $e$ such that $\rho'[i] = alive$ for $0 \leq i \leq e$ and $\rho'[i] = \neg alive$ for $i > e$. Thus we have a finite trace $\rho$ such that $\rho \approx \rho'$ with $e$. Therefore, $\rho \models \phi$ holds by Lemma 1.

  - $G\phi$ is falsified such that the environment behaves such as only having $\alpha$ hold for finite times, in which case the fairness assumption is violated, we conclude that $\rho \models \phi$ holds by default.

Finally, in order to obtain the winning strategy $g$, we have $g(\lambda) = g'(\lambda)|_\lambda$, where $\lambda \in (2^{X})^{\omega}$.

$\to$: Since $\langle X, Y, \alpha, \phi \rangle$ is realizable, there is a winning strategy $g : (2^{X})^{+} \to 2^{X}$ such that for every trace $\rho$ that follows $g$, either of the following situations happens:

  - The environment behaves such as having $\alpha$ hold infinitely often such that $G\phi$ is true, then there is $k \geq 0$ such that $\rho^k \models \phi$. Since $alive$ is assigned as an agent variable, we can construct a play $\rho'$ such that $\rho'[i] = \rho[i] \land \neg alive$ for $0 \leq i \leq k$ and $\rho'[i] = \rho[i] \land alive$ for $i > k$. Thus we have $\rho' \approx \rho$ such that $\rho' \models \psi$ by Lemma 1. Therefore, we conclude that $\rho' \models G\alpha \to \psi$ holds.

  - The environment behaves such as violating the fairness assumption such that only having $\alpha$ hold for finite times, in which case $G\phi$ does not hold. Therefore, $G\phi \to \psi$ is true.

Finally, in order to obtain the winning strategy $g'$, we have $g'(\lambda) = g(\lambda) \land alive$ if $\phi$ has not been satisfied and $g'(\lambda) = g(\lambda) \land \neg alive$ since $\phi$ has been satisfied, where $\lambda \in (2^{X})^{\omega}$.

Theorem 18. Let $\phi$ be an LTL$_f$ formula, $\psi$ be the corresponding translated LTL formula, then stable LTL$_f$ synthesis problem $\langle X, Y, \alpha, \phi \rangle$ is realizable if and only if LTL formula $G\alpha \to \psi$ is realizable with respect to $\langle X, Y \cup \{alive\} \rangle$.

Proof. We prove the two directions separately.

- $\leftarrow$: Since $G\phi \to \psi$ is realizable with respect to $\langle X, Y \cup \{alive\} \rangle$, there exists a winning strategy $g'$ : $(2^{X})^{+} \to 2^{X}$ such that every trace $\rho'$ that follows $g'$ gets
Let $\psi$ be an LTL$_f$ formula, $\psi$ be the corresponding translated LTL formula, then $\psi$ is realizable with respect to $\langle X, Y \rangle$ with assumption $\psi_A$ if and only if LTL formula $\psi_A \rightarrow \psi$ is realizable with respect to $\langle X, Y \cup \{\text{alive}\} \rangle$.

**Proof.** We prove the two directions separately.

- $\rightarrow$: Since $\psi_A \rightarrow \psi$ is realizable with respect to $\langle X, Y \cup \{\text{alive}\} \rangle$, there exists a winning strategy $g' : (2^X)^+ \rightarrow 2^{Y \cup \{\text{alive}\}}$ such that every trace $\rho'$ that follows $g'$ gets $\psi_A \rightarrow \psi$ hold, therefore, enabling either of the following situations:
  - $\psi_A$ is true such that the environment behaves such as having $\psi_A$ hold, and $\psi$ holds. Therefore, $\rho' \models \text{alive} \wedge (\text{alive} U (G\neg\text{alive}))$, and there exists a position $e$ such that $\rho'[e] \models \text{alive}$ for $e > e$. Thus we have a finite trace $\rho$ such that $\rho \models \phi$ with $e$. Therefore, $\rho \models \phi$ holds by Lemma 1.
  - $\psi_A$ is falsified such that the environment behaves such as having assumption $\psi_A$ get violated, we conclude that $\rho \models \phi$ holds by default.

Finally, in order to obtain the winning strategy $g$, we have $g(\lambda) = g'(\lambda)|_{\lambda} = (2^X)^\omega$.

- $\leftarrow$: Since $\phi$ is realizable with respect to $\langle X, Y \rangle$ under assumption $\psi_A$, there is a winning strategy $g : (2^X)^+ \rightarrow 2^Y$ such that for every trace $\rho$ that follows $g$, either of the following situations happens:
  - The environment behaves such as having $\psi_A$ hold, then there is $k \geq 0$ such that $\rho^k \models \phi$. Since $\text{alive}$ is assigned as an agent variable, we can construct a trace $\rho'$ such that $\rho'[i] = \rho[i] \wedge \text{alive}$ for $i \leq k$ and $\rho'[i] = \rho[i] \wedge \neg\text{alive}$ for $i > k$. Thus we have $\rho' \models \phi$ by Lemma 1. Therefore, we conclude that $\rho' \models F \alpha \rightarrow \psi$. holds.
  - The environment behaves such as violating the assumption $\psi_A$, in which case $\psi_A \not\rightarrow \psi$ is true.

Finally, in order to obtain the winning strategy $g'$, we have $g'(\lambda) = g(\lambda) \wedge \text{alive}$ if $\phi$ has not been satisfied and $g'(\lambda) = g(\lambda) \wedge \neg\text{alive}$ since $\phi$ has been satisfied, where $\lambda \in (2^X)^\omega$.