Realizability Problem for Constraint LTL

Ashwin Bhaskar
Chennai Mathematical Institute, India

M. Praveen
Chennai Mathematical Institute, India
CNRS IRL ReLaX, Chennai, India

Abstract

Constraint linear-time temporal logic (CLTL) is an extension of LTL that is interpreted on sequences of valuations of variables over an infinite domain. The atomic formulas are interpreted as constraints on the valuations. The atomic formulas can constrain valuations over a range of positions along a sequence, with the range being bounded by a parameter depending on the formula. The satisfiability and model checking problems for CLTL have been studied by Demri and D'Souza. We consider the realizability problem for CLTL. The set of variables is partitioned into two parts, with each part controlled by a player. Players take turns to choose valuations for their variables, generating a sequence of valuations. The winning condition is specified by a CLTL formula – the first player wins if the sequence of valuations satisfies the specified formula. We study the decidability of checking whether the first player has a winning strategy in the realizability game for a given CLTL formula. We prove that it is decidable in the case where the domain satisfies the completion property, a property introduced by Balbiani and Condotta in the context of satisfiability. We prove that it is undecidable over $(\mathbb{Z}, <, =)$, the domain of integers with order and equality. We prove that over $(\mathbb{Z}, <, =)$, it is decidable if the atomic constraints in the formula can only constrain the current valuations of variables belonging to the second player, but there are no such restrictions for the variables belonging to the first player. We call this single-sided games.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Modal and temporal logics; Theory of computation → Verification by model checking; Theory of computation → Automata over infinite objects; Theory of computation → Tree languages

Keywords and phrases Realizability, constraint LTL, Strategy trees, Tree automata

Digital Object Identifier 10.4230/LIPIcs.TIME.2022.8

Related Version *Full Version*: `https://arxiv.org/abs/2207.06708`

Funding *M. Praveen*: This author is partially supported by the Infosys foundation.

1 Introduction

Propositional linear temporal logic (LTL) and related automata theoretic models have been extended in various ways to make it more expressive. Prompt-LTL [18], Constraint LTL [13], LTL with freeze operators [12], temporal logic of repeating values [11, 24], finite memory automata [16], data automata [8] are all examples of this. Prompt-LTL is concerned with bounding wait times for formulas that are intended to become true eventually, while other extensions are concerned with using variables that range over infinite domains in place of Boolean propositions used in propositional LTL. Variables ranging over infinite domains are a natural choice for writing specifications for systems that deal with infinite domains. For example, constraint LTL has been used for specifications of cloud based elastic systems [6], where the domain of natural numbers is used to reason about the number of resources that are being used by cloud based systems.

An orthogonal development in formal verification is synthesis, that is concerned with automatically synthesizing programs from logical specifications. The problem was identified by Church [10] and one way to solve it is by viewing it as the solution of a two person game. For specifications written in propositional LTL, the worst case complexity of the realizability problem is doubly exponential [23]. However, efficient algorithms exist for fragments of LTL. The algorithms are efficient enough and the fragments are expressive enough to be used in practice, for example to synthesize robot controllers [17], data buffers and data buses [22].

This paper is in an area that combines both developments mentioned in the above paragraphs. We consider constraint LTL (CLTL) and partition the set of variables into two parts, each being owned by a player in a two player game. The players take turns to choose a valuation for their variables over an infinite domain. The game is played forever and results in a sequence of valuations. The first player tries to ensure that the resulting sequence satisfies a specified CLTL formula (which is the winning condition) and the second player tries to foil this. We study the decidability of checking whether the first player has a winning strategy, called the realizability problem in the sequel. CLTL is parameterized by a constraint system, that can have various relations over the infinite domain. The atomic formulas of CLTL can compare values of variables in different positions along a range of positions, using the relations present in the constraint system. The range of positions is bounded and depends on the formula. E.g., an atomic formula can say that the value of x at a position is less than the value of y in the next position, in the domain of integers or real numbers with linear order. Decidability of the CLTL realizability problem depends on the constraint system. It also depends on whether the atomic formulas can compare values at different positions of the input, as opposed to comparing values of different variables at the same position of the input. If the former is allowed only for variables belonging to one of the players, they are called single-sided games. This is illustrated next.

For instance in cloud based elastic systems [6], the number of resources allocated and the number of virtual machines running are tracked. One desirable property is that if the number of virtual machines increases, the number of resources allocated also increase. Typically the number of resources allocated is controlled by the system and the number of virtual machines is controlled by the environment. Let x be a variable that keeps track of the number of resources allocated and let y denote the number of virtual machines. Specifying this property will require comparing the value of x at the current position with the value of x at the next position. We may also compare the current value of y with its value at the next position, but this will need both the system and the environment to be able to compare the values of their variables at different positions. Instead, if we restrict the environment to only decide whether a new virtual machine request is raised at the current position, the environment need not compare the value of y with its value at the next position. Hence only system compares the values of its variables across different positions and thus, the game will be single-sided.

Contributions. We prove that the realizability problem for CLTL is

1. 2EXPTIME-complete for constraint systems that satisfy a so-called completion property,
2. undecidable for integers with linear order and equality and
3. 2EXPTIME-complete for single-sided games on integers with linear order and equality.

The third result above is the main one and is inspired by concepts used in satisfiability [13]. In satisfiability, this technique is based on patterns that repeat in ultimately periodic words. It requires new insights to make it work in trees that we use to represent strategies here.

Related works. Two player games on automata models and logics dealing with infinite domains have been studied before [28, 14]. The techniques involved are similar to those used here in the sense that instead of reasoning about sequences of values from an infinite domain, sequences of elements from a finite abstraction are considered. Single-sided games are considered in [14], like we do here, but for register automata specifications. Their result subsumes ours, since register automata are more expressive than CLTL. In register automata, values can be compared even if they occur far apart in the input sequence, but in CLTL, values can only be compared if they occur within a bounded distance. For this reason, CLTL can be handled with simpler arguments, resulting in some differences in technical details, which we will highlight later in this paper. This can potentially speed up procedures in case the specifications only need CLTL and not the full power of register automata[1]. Similar single-sided games are also considered in [25], for an extension of LTL incomparable with CLTL. There, single-sided games are reduced to energy games [2] to get decidability.

Church Synthesis problem for a restriction of First Order Logic over parametrized alphabets has been studied in [5]. The parametrized alphabet reflects the number of processes. Similar to the results in our paper, the synthesis problem in [5] is undecidable in the general case but turns decidable when the number of processes that are controllable by the environment is bounded, while the number of system processes remains unbounded. In [26], a parametrized extension of the Church Synthesis Problem of MSO Logic over $(\mathbb{N}, <)$ is considered. The decidability result in this paper extensively uses the idea of patterns repeating in ultimately periodic words [26, Proposition 3.4] as is the case in our work too.

2 Preliminaries

Let $\mathbb{Z}$ be the set of integers and $\mathbb{N}$ be the set of non-negative integers. We denote by $\lceil i \rceil_k$ the number i ceiled at k: $\lceil i \rceil_k = i$ if $i \leq k$ and $\lceil i \rceil_k = k$ otherwise. If m is any mapping and S is a subset of the domain of m, we denote by $m \restriction S$ the mapping m restricted to the domain S. For a sequence of mappings $m_1 \cdot m_2 \cdots$, we write $m_1 \cdot m_2 \cdots \restriction S$ for $m_1 \restriction S \cdot m_2 \restriction S \cdots$. For integers n_1, n_2, we denote by $[n_1, n_2]$ the set $\{n \in \mathbb{Z} \mid n_1 \leq n \leq n_2\}$.

We recall the definitions of constraint systems and constraint LTL (CLTL) from [13]. A constraint system $\mathcal{D}$ is of the form $(D, R_1, \ldots, R_n, \mathcal{I})$, where D is a non-empty set called the domain. Each R_i is a predicate symbol of arity a_i, with $\mathcal{I}(R_i) \subseteq D^{a_i}$ being its interpretation.

Let V be a set of variables, partitioned into the sets V^a, V^b of look-**a**head and future-**b**lind variables. A look-ahead term is of the form $X^i y$, where y is a look-ahead variable, $i \geq 0$ and X is a symbol intended to denote "next". For $k \geq 0$, we denote by $T^a[k]$ the set of all look-ahead terms of the form $X^i y$, where $i \in [0, k]$ and y is a look-ahead variable. A constraint c is of the form $R(t_1, \ldots, t_n)$, where R is a predicate symbol of arity n and $t_1, \ldots, t_n$ are all future-blind variables or they are all look-ahead terms. The syntax of CLTL is given by the following grammar, where c is a constraint as defined above.

$$\phi ::= c \mid \neg\phi \mid \phi \vee \phi \mid X\phi \mid \phi U \phi$$

The semantics of CLTL is defined over sequences σ (also called concrete models in the following); for every $i \geq 0$, $\sigma(i) \colon V \to D$ is a mapping of the variables. Given, $x_1, \ldots, x_n \in V^a$ and $i_1, \ldots, i_n \in \mathbb{N}$, the i^{th} position of a concrete model σ satisfies the constraint

[1] This does need a detailed study, which we defer to future work.

$R(X^{i_1}x_1, \ldots, X^{i_n}x_n)$ (written as $\sigma, i \models R(X^{i_1}x_1, \ldots, X^{i_n}x_n)$) if $(\sigma(i+i_1)(x_1), \ldots, \sigma(i+i_n)(x_n)) \in \mathcal{I}(R)$. If the constraint is of the form $R(x_1, \ldots, x_n)$ where $x_1, \ldots, x_n \in V^b$, then $\sigma, i \models R(x_1, \ldots, x_n)$ if $(\sigma(i)(x_1), \ldots, \sigma(i)(x_n)) \in \mathcal{I}(R)$. The semantics is extended to the rest of the syntax similar to the usual propositional LTL. We use the standard abbreviations $F\phi$ (resp. $G\phi$) to mean that ϕ is true at some position (resp. all positions) in the future. The X-length of a look-ahead term $X^i y$ is i. We say that a formula is of X-length k if it uses look-ahead terms of X-length at most k. The constraint system $(\mathbb{Z}, <, =)$ (resp. $(\mathbb{N}, <, =)$) has the domain $\mathbb{Z}$ (resp. $\mathbb{N}$) and $<, =$ are interpreted as the usual linear order and equality relations. The formula $G(x < Xy)$ will be true in the first position of a concrete model if in all positions, the value of x is less than the value of y in the next position. Recall the example of cloud based elastic systems described in the introduction. Variable x denoted the number of resources allocated and it was possible to compare its values across different positions, hence making it a look-ahead variable. Whereas, under the restrictions on the environment, the value of the variable y was not allowed to be compared with the values of variables at other positions. This makes y a future-blind variable.

We adapt the concept of realizability games [23] to CLTL. There are two players `system` and `environment`. The set of variables V is partitioned into two parts SV, EV owned by `system`, `environment`, respectively. The `environment` begins by choosing a mapping $em_0 \colon EV \to D$, to which `system` responds by choosing a mapping $sm_0 \colon SV \to D$. This first round results in the mapping $em_0 \oplus sm_0$. This notation is used to define the function such that $em_0 \oplus sm_0(x) = em_0(x)$ if $x \in EV$ and $em_0 \oplus sm_0(x) = sm_0(x)$ if $x \in SV$. In the next round, the two players chose mappings em_1, sm_1. Both players continue to play forever and the play results in a concrete model $\sigma = (em_0 \oplus sm_0)(em_1 \oplus sm_1) \cdots$. The winning condition is specified by a CLTL formula ϕ. `System` wins this play of the game if $\sigma, 0 \models \phi$.

Let M (resp. EM,SM) be the set of all mappings of the form $V \to D$ (resp. $EV \to D$, $SV \to D$). For a concrete model σ and $i \geq 0$, let $\sigma \restriction i$ denote the prefix of σ of length i (for $i = 0$, $\sigma \restriction i$ is the empty sequence ϵ). An `environment` strategy is a function $et \colon M^* \to EM$ and a `system` strategy is a function $st \colon M^* \cdot EM \to SM$. We say that the `environment` plays according to the strategy et if the resulting model $\sigma = (em_0 \oplus sm_0)(em_1 \oplus sm_1) \cdots$ is such that $em_i = et(\sigma \restriction i)$ for all $i \geq 0$. `System` plays according to the strategy st if the resulting model $\sigma = (em_0 \oplus sm_0)(em_1 \oplus sm_1) \cdots$ is such that $sm_i = st(\sigma \restriction i \cdot em_i)$ for all $i \geq 0$. We say that st is a winning strategy for `system` if she wins all plays of the game played according to st, irrespective of the strategy used by `environment`. For example, let us consider a CLTL game with $V = V^a = \{x, y\}, EV = \{x\}, SV = \{y\}$, over the constraint system $(\mathbb{Z}, <, =)$ with winning condition $G((y > Xy) \wedge \neg((X^2x > y) \wedge (X^2x < Xy)))$. For `system` to win, the sequence of valuations for y should form a descending chain, and at any position, the value of x should be outside the interval defined by the previous two values of y. `System` has a winning strategy in this game: it can choose y to be $-i$ in the i^{th} round and the `environment` cannot choose its x to be strictly between the previous two values of y in any round. `System` does not have a winning strategy in the same game when it is considered over $(\mathbb{N}, <, =)$, as there is no infinite descending sequence of natural numbers. `System` does not have a winning strategy over dense domains, since `environment` can choose the third value of x to be strictly between the first two values of y, violating the winning condition. Given a CLTL formula ϕ, the **realizability problem** is to check whether `system` has a winning strategy in the CLTL game whose winning condition is ϕ.

We now state an important result.

▶ **Theorem 1.** *The realizability problem for CLTL over $(\mathbb{Z}, <, =)$ and $(\mathbb{N}, <, =)$ is undecidable.*

This can be proved by a reduction from the repeated control state reachability problem for 2-counter machines, which is known to be undecidable [3]. The main idea of the reduction is that one of the players simulates the counter machine and the other player catches mistakes, like other similar reductions for games [1]. For a detailed proof of this result, please refer to the arXiv version of this paper [7].

Some proofs and technical details in the subsequent sections are moved to the appendix due to space constraints.

3 Symbolic Models

The models of CLTL are infinite sequences over infinite alphabets. Frames, introduced in [13], abstract them to finite alphabets. We adapt frames to constraint systems of the form $(D, <, =)$. Conceptually, frames and symbolic models as we will define here are almost the same as introduced in [13], where the authors used these notions to solve the satisfiability problem for CLTL. For the purpose of CLTL games, we use slightly different definitions and notations, as this makes it easier to present game-theoretic arguments. For the rest of the paper, we shall assume that the set of variables V is finite. Also, unless mentioned otherwise, we shall assume that D is $\mathbb{Z}$, $\mathbb{N}$ or a domain that satisfies a so-called completion property.

Suppose that the first player owns the variables x, z. The second player owns y and wants to ensure that $x < y \ \wedge \ y < z$ over the domain of integers. It depends on whether the gap between the values assigned by the first player to x and to z, is large enough for the second player to push y in between.

▶ **Definition 2** (gap functions). *Given a mapping $m\colon V^b \to D$, we associate with it a gap function $gp\colon V^b \to \mathbb{N}$ as follows. Arrange V^b as $x_0, x_1, \ldots$ such that $m(x_0) \leq m(x_1) \leq \cdots$. Define the function gp such that $gp(x_0) = 0$ and $gp(x_{l+1}) = gp(x_l) + \lceil m(x_{l+1}) - m(x_l) \rceil_{|V^b|-1}$ for all $l < |V^b| - 1$.*

The left hand side of the above equation denotes the gap between x_l and x_{l+1} according to the gp function. The right hand side denotes the gap between the same variables according to the mapping m, ceiled at $|V^b| - 1$. Since V^b is finite, the set of gap functions is also finite. We use gap functions only for future-blind variables V^b, only for the domains $\mathbb{Z}$ or $\mathbb{N}$. Hence, the minus sign '$-$' in the definition of gap functions is interpreted as the usual subtraction over $\mathbb{Z}$ or $\mathbb{N}$.

The following definition formalizes how a frame captures information about orders and gaps for s successive positions.

▶ **Definition 3** (Frames). *Given a number $s \geq 1$, an s-frame f is a pair $(\leq_f, gp_f)$, where $\leq_f$ is a total pre-order[2] on the set of look-ahead terms $T^a[s-1]$ and $gp_f\colon V^b \times [0, s-1] \to \mathbb{N}$ is a function such that for all $i \in [0, s-1]$, $\lambda x.gp_f(x, i)$[3] is a gap function.*

In the notation s-frame, s is intended to denote the size of the frame – the number of successive positions about which information is captured. The current position and the following $(s-1)$ positions are considered, for which the look-ahead terms in $T^a[s-1]$ are needed. We denote by $<_f$ and $\equiv_f$ the strict order and equivalence relation induced by $\leq_f$: $x <_f y$ iff $x \leq_f y$ and $y \not\leq_f x$ and $x \equiv_f y$ iff $x \leq_f y$ and $y \leq_f x$.

[2] a reflexive and transitive relation such that for all x, y, either $x \leq_f y$ or $y \leq_f x$

[3] Note that we could have used a function $h_i(x) = gp_f(x, i)$ instead of using the lambda notation. But this introduces a new notation – the function h_i, which will not be used anywhere else.

We will deal with symbolic models that constitute sequences of frames. An s-frame will capture information about the first s positions of a model. If this is followed by a $(s+1)$-frame, it will capture information about the first $(s+1)$ positions of the model. Both frames capture information about the first s positions, so they must be consistent about the information they have about the shared positions. Similarly, an s-frame meant for positions i to $i+s-1$ may be followed by another s-frame meant for positions $i+1$ to $i+s$. The two frames must be consistent about the positions $i+1$ to $i+s-1$ that they share. The following definition formalizes these requirements.

▶ **Definition 4** (One-step compatibility). *For $s \geq 1$, an s-frame f and an $(s+1)$-frame g, the pair (f,g) is one-step compatible if the following conditions are true.*

- *For all terms $t_1, t_2 \in T^a[s-1]$, $t_1 \leq_f t_2$ iff $t_1 \leq_g t_2$.*
- *For all $j \in [0, s-1]$ and all variables $x \in V^b$, $gp_f(x,j) = gp_g(x,j)$.*

For $s \geq 2$ and s-frames f, g, the pair (f,g) is one-step compatible if:

- *For all terms $t_1, t_2 \in T^a[s-2]$, $Xt_1 \leq_f Xt_2$ iff $t_1 \leq_g t_2$ and*
- *for all $j \in [0, s-2]$ and all variables $x \in V^b$, $gp_f(x, j+1) = gp_g(x,j)$.*

Fix a number $k \geq 0$ and consider formulas of X-length k. A symbolic model is a sequence ρ of frames such that for all $i \geq 0$, $\rho(i)$ is an $\lceil i+1 \rceil_{k+1}$-frame and $(\rho(i), \rho(i+1))$ is one-step compatible. CLTL formulas can be interpreted on symbolic models, using symbolic semantics $\models_s$ as explained next. To check if the i^{th} position of ρ symbolically satisfies the atomic constraint $t_1 < t_2$ (where t_1, t_2 are look-ahead terms), we check whether $t_1 < t_2$ according to the i^{th} frame $\rho(i)$. In formal notation, this is written as $\rho, i \models_s t_1 < t_2$ if $t_1 <_{\rho(i)} t_2$. For future-blind variables x, y, $\rho, i \models_s x < y$ if $gp_{\rho(i)}(x,0) < gp_{\rho(i)}(y,0)$. The symbolic satisfaction relation $\models_s$ is extended to all CLTL formulas of X-length k by induction on the structure of the formula, as done for propositional LTL. To check whether $\rho, i \models_s t_1 < t_2$ in this symbolic semantics, we only need to check $\rho(i)$, the i^{th} frame in ρ, unlike the CLTL semantics, where we may need to check other positions also. In this sense, the symbolic semantics lets us treat CLTL formulas as if they were formulas in propositional LTL and employ techniques that have been developed for propositional LTL. But to complete that task, we need a way to go back and forth between symbolic and concrete models.

Given a concrete model σ, we associate with it a symbolic model $\mu(\sigma)$ as follows. Imagine we are looking at the concrete model through a narrow aperture that only allows us to view $k+1$ positions of the concrete model, and we can slide the aperture to view different portions. The i^{th} frame of $\mu(\sigma)$ will capture information about the portion of the concrete model visible when the right tip of the aperture is at position i of the concrete model (so the left tip will be at $i - \lceil i \rceil_k$). Formally, the total pre-order of the i^{th} frame is the one induced by the valuations along the positions $i - \lceil i \rceil_k$ to i of the concrete model. For every $j \in [0, \lceil i \rceil_k]$, the function $\lambda x.gp_f(x,j)$ of the i^{th} frame is the gap function associated with the mapping $\sigma(i - \lceil i \rceil_k + j) \restriction V^b$.

For every concrete model, there is an associated symbolic model, but the converse is not true. E.g., if every frame in a symbolic model requires $Xx < x$, the corresponding concrete model needs to have an infinite descending chain, which is not possible in the constraint system $(\mathbb{N}, <, =)$. We say that a symbolic model ρ admits a concrete model if there exists a concrete model σ such that $\rho = \mu(\sigma)$.

▶ **Lemma 5** ([13, Lemma 3.1]). *Let ϕ be a CLTL formula of X-length k. Let σ be a concrete model and $\rho = \mu(\sigma)$. Then $\sigma, 0 \models \phi$ iff $\rho, k \models_s \phi$.*

4 Decidability Over Domains Satisfying the Completion Property

In this section, we prove that the CLTL realizability problem is decidable if the domain satisfies a so called completion property. Let C be a set of constraints over a constraint system $\mathcal{D}$. We call C satisfiable if there is a valuation satisfying all the constraints in C. For a subset $U \subseteq V$ of variables, $C \restriction U$ is the subset of C consisting of those constraints that only use terms built with variables in U. A partial valuation v' is a valuation for the terms occurring in $C \restriction U$. We say $\mathcal{D}$ has the completion property if for every satisfiable set of constraints C and every subset $U \subseteq V$, every partial valuation v' satisfying $C \restriction U$ can be extended to a valuation v satisfying C. An example of a constraint system which does not satisfy the completion property is $(\mathbb{Z}, <, =)$, since for the set of constraints $C = \{x < y, x < z, z < y\}$ over the set of variables $V = \{x, y, z\}$, the partial valuation $v\colon x \mapsto 0, y \mapsto 1$ satisfies the constraints in C involving x and y, but cannot be extended to a valuation which satisfies the constraints $x < z$ and $z < y$ in C. The constraint systems $(\mathbb{Q}, <, =)$ and $(\mathbb{R}, <, =)$ satisfy the completion property. Also, one can easily see that for every infinite domain D, the constraint system $(D, =)$ always satisfies the completion property.

It is known that CLTL satisfiability is decidable for constraint systems that satisfy the completion property [13, 4]. The completion property of a constraint system is closely related to the denseness of the underlying domain. A constraint system satisfies the completion property if and only if the underlying domain is dense and open [13, Lemma 5.3].

Consider an example of a controller system that controls the temperature of water in a water tank. Let x be a variable that denotes the temperature of the water. The controller may be required to guarantee, for instance, that $20 \leq x \leq 100$. In principle, the temperature of water can be any real number. Hence x comes from a dense domain. So the domain over which properties of such a system are specified satisfies the completion property (refer to [27] for a detailed explanation of such a controller). In contrast, consider cloud-based elastic systems [6], which we briefly described in the introduction. It is clear that both–the number of resources allocated and the number of virtual machines running are natural numbers. As the domain of natural numbers is not dense, we can conclude that constraint systems used to model these cloud-based elastic systems do not satisfy the completion property.

Now we prove that for constraint systems of the form $(D, <, =)$ that satisfy the completion property, the CLTL realizability problem is decidable. This holds even when both players have look-ahead variables, so we don't need to treat future-blind variables separately. Hence, we set V^b to be empty and ignore gap functions in frames.

We reduce CLTL games to parity games on finite graphs, which are known to be decidable (see, e.g., [19]). In a CLTL game, `environment` chooses a valuation for EV, which we track in our finite graph by storing the positions of the new values relative to the values chosen in the previous rounds. We do this with partial frames, which we define next.

▶ **Definition 6** (Partial frames and compatibility). *For $s \geq 1$, a partial s-frame pf is a total pre-order $\leq_{pf}$ on the set of terms $T^a[s-2] \cup \{X^{s-1}y \mid y \in EV\}$. For $s \geq 0$, an s-frame f and an $(s+1)$-partial frame pf, the pair (f, pf) is one step compatible if for all $t_1, t_2 \in T^a[s-1]$, $t_1 \leq_f t_2$ iff $t_1 \leq_{pf} t_2$. For $s \geq 2$, an s-frame f and an s-partial frame pf, the pair (f, pf) is one-step compatible if for all $t_1, t_2 \in T^a[s-2]$, $Xt_1 \leq_f Xt_2$ iff $t_1 \leq_{pf} t_2$. For $s \geq 2$, an s-partial frame pf and an s-frame f, (pf, f) is one step compatible if for all $t_1, t_2 \in T^a[s-2] \cup \{X^{s-1}y \mid y \in EV\}$, $t_1 \leq_{pf} t_2$ iff $t_1 \leq_f t_2$.*

In the set of terms $T^a[s-2] \cup \{X^{s-1}y \mid y \in EV\}$ used in partial frames, the terms in the first set represent values chosen in the previous rounds and the terms in the second set represent values chosen by `environment` for EV in the current round.

Note that a partial s-frame is a total pre-order on the set of terms $T^a[s-2] \cup \{X^{s-1}y \mid y \in EV\}$ and an s-frame is a total pre-order on the set of terms $T^a[s-1]$. Let pf be an s-partial frame and let f be an s-frame such that (pf, f) is one-step compatible. Suppose $C_1 = \{t_1 = t_2 \mid t_1 \equiv_f t_2\} \cup \{t_1 < t_2 \mid t_1 <_f t_2\}$ and $C_2 = \{t_1 = t_2 \mid t_1 \equiv_{pf} t_2\} \cup \{t_1 < t_2 \mid t_1 <_{pf} t_2\}$. Clearly, C_2 is a subset of C_1 skipping all those constraints that contain `system` variables corresponding to the s^{th} position. If a finite sequence of mappings $(em_1 \oplus sm_1)...(em_{s-1} \oplus sm_{s-1})em_s$ satisfies the pre-order $\leq_{pf}$ then it satisfies the constraints in C_2. Since the constraint system satisfies the completion property, there must exist a `system` mapping sm_s for the `system` variables at position s such that the sequence of mappings $(em_1 \oplus sm_1)...(em_s \oplus sm_s)$ satisfies the constraints in C_1 and hence, also satisfies the pre-order $\leq_f$. Thus, we have the following proposition:

▶ **Proposition 7.** *Given $s \geq 1$, suppose $(em_1 \oplus sm_1)...(em_i \oplus sm_i)em$ is a sequence of mappings, where $em_1, \ldots, em_i, em \in EM$, $sm_1, \ldots, sm_i \in SM$, pf is the s-partial frame induced by em and the previous $(s-1)$ mappings in the sequence, and f is an s-frame such that (pf, f) is one-step compatible (where $i \geq s$). If the constraint system satisfies the completion property, then em can be extended to a mapping $em \oplus sm$ such that f is the s-frame associated with $em \oplus sm$ and the previous $(s-1)$ mappings in the sequence.*

We know that any LTL formula ϕ can be converted to an equivalent non-deterministic Büchi automaton with an exponential number of states in the size of ϕ in EXPTIME [30]. Now, every non-deterministic Büchi automaton B with n states can be converted to a deterministic parity automaton [15, Chapter 1] with number of states exponential in n and number of colours polynomial in n [21, Theorem 3.10]. Using these results, it is easy to see that given a CLTL formula ϕ, we can construct a deterministic parity automaton A_ϕ with set of states Q and with number of colours d, accepting the set of all sequences of frames that symbolically satisfy ϕ, such that $|Q|$ is double exponential in the size of ϕ and d is exponential in the size of ϕ. Now we design parity games to simulate CLTL games.

▶ **Definition 8.** *Let ϕ be the CLTL formula defining the winning condition for a CLTL game and k be its X-length. Let $\mathcal{F}$ denote the set of all s-frames for $s \in [0, k]$. Let A_ϕ be a deterministic parity automaton accepting the set of all sequences of frames that symbolically satisfy ϕ, with Q being the set of states, $q_I \in Q$ being the initial state and d being the number of colours. We define a parity game with* `environment` *vertices $V_e = \{(f, q_I) \mid f$ is an s-frame, $0 \leq s \leq k\} \cup \{(f, q) \mid f$ is a $(k+1)$-frame, $q \in Q\}$. The set of* `system` *vertices is $V_s = \{(f, q_I, pf) \mid f$ is an s-frame, $0 \leq s \leq k$, pf is an $(s+1)$-partial frame$\} \cup \{(f, q, pf) \mid f$ is a $(k+1)$-frame, pf is a $(k+1)$-partial frame$\}$. There is an edge from (f, q) to (f, q, pf) if (f, pf) is one-step compatible, f is an s-frame for some s and pf is a partial $\lceil s+1 \rceil_{(k+1)}$-frame. There is an edge from (f, q_I, pf) to (g, q_I) if (pf, g) is one step compatible and g is an s-frame for $s \in [1, k]$. There is an edge from (f, q, pf) to (g, q') if (pf, g) is one-step compatible, g is a $(k+1)$-frame and A_ϕ goes from q to q' on reading g. Vertices (f, q) and (f, q, pf) get the same colour as q in the parity automaton A_ϕ. The initial vertex is $(\bot, q_I)$, where $\bot$ is the trivial 0-frame.*

The edges of the parity game above are from V_s to V_e or vice-versa. They are designed such that q_I is the only state used for the first k rounds, where the frames will be of size at most k (this is because for the `system` to win in a play of the parity game generating a frame sequence ρ, we only require that the sequence $\boldsymbol{\rho[k, \infty)}$ symbolically satisfy ϕ, according to Lemma 5). For the first $(k+1)$ frame, an edge from a `system` vertex of the form (f, q_I, pf)

to an `environment` vertex of the form (g, q') is taken and from then on, we track the state of the parity automaton as it reads the sequence of frames contained in the sequence of vertices that are chosen by the players in the game.

▶ **Lemma 9.** *For a CLTL game over a constraint system satisfying the completion property with winning condition given by a formula ϕ,* `system` *has a winning strategy iff she has a positional winning strategy in the parity game given in Definition 8.*

Proof idea. For every play in the CLTL game, there is a corresponding play in the parity game, but the converse is not true in general, since only the order of terms are tracked in the parity game and not the actual values. For constraint systems satisfying the completion property, Proposition 7 implies that there exist valuations corresponding to all possible orderings of terms, so the converse is also true. ◀

▶ **Theorem 10.** *The CLTL realizability problem over constraint systems that satisfy the completion property is 2EXPTIME-complete.*

Proof. From Lemma 9, this is effectively equivalent to checking the existence of a winning strategy for `system` in a game. Now, checking if `system` has a winning strategy in the parity game (constructed using A_ϕ) can be achieved in $O(n^{\log d})$ time where n is the number of states in the game graph [9]. Now, by our construction, $n = |Q| \times |\mathcal{F}|$. We know, $|\mathcal{F}|$ is the number of total pre-orders on V, for which $2^{(k.|V|)^2}$ is a crude upper bound. This means that $|\mathcal{F}|$ is exponential in the size of ϕ and hence, overall we get a 2EXPTIME upper bound for our realizability problem. We also know that the realizability problem for LTL is complete for 2EXPTIME [23]. Thus, the CLTL realizability problem over constraint systems satisfying the completion property is also 2EXPTIME-complete. ◀

We know that a positional winning strategy in the parity game for a player, if it exists, can be implemented by a deterministic finite state transducer. Since $\mathcal{D}$ satisfies the completion property, consider a resource-bounded Turing machine M, which can, given an environment mapping em as described in Proposition 7, extend it to a mapping $em \oplus sm$ such that the order f imposed by the $em \oplus sm$ and the previous $s-1$ mappings over the set of all terms extends the order pf imposed by em and the previous $s-1$ mappings. Now, for implementing the winning strategy for a player in a CLTL game, we use the deterministic finite state transducer corresponding to the parity game given in Definition 8. For every input of a partial frame pf by `environment` in a round, the transducer returns a frame f for `system` that extends pf. The transducer along with the machine M implements the winning strategy for `system` in a given CLTL game, if it exists.

Note that as we saw above, the constraint systems $(\mathbb{N}, =)$ and $(\mathbb{Z}, =)$ (with just equality and no linear order) also satisfy the completion property. So, it follows that the CLTL realizability problem over these constraint systems is also decidable.

5 Decidability of single-sided CLTL games over $(\mathbb{Z}, <, =)$

We consider games where `environment` has only future-blind variables, while the `system` has both future-blind and look-ahead variables. We call this single-sided CLTL games. So, in a single-sided game, $EV = EV^b$ and $SV = SV^b \cup SV^a$. Given a CLTL formula ϕ, the **single-sided realizability problem** is to check whether `system` has a winning strategy in the single-sided CLTL game whose winning condition is ϕ. We only consider the constraint system $(\mathbb{Z}, <, =)$ and show that the single-sided realizability problem is decidable over $(\mathbb{Z}, <, =)$. We do this in two stages. In the first stage, we reduce it to the problem

of checking the non-emptiness of a set of trees satisfying certain properties. These trees represent `system` strategies. In the second stage, we show that non-emptiness can be checked using tree automata techniques.

Let G be the set of gap functions associated with mappings of the form $EV^b \to \mathbb{Z}$. For $s \geq 1$, an s-frame g and a function $gp \in G$, the pair (gp, g) is gap compatible if for all $x, y \in EV^b$, $gp(x) - gp(y) = gp_g(x, s-1) - gp_g(y, s-1)$. Intuitively, the gaps that frame g imposes between EV^b variables in its last position are the same as the gaps imposed by gp. We now have the following proposition (refer to the arXiv version of the paper for the proof).

▶ **Proposition 11** (gap compatibility). *For $s \geq 1$, an s-frame g and a function $gp \in G$, suppose the pair (gp, g) is gap compatible. If gp is the gap function associated with a mapping $em\colon EV^b \to \mathbb{Z}$, it can be extended to a mapping $em \oplus sm\colon V^b \to \mathbb{Z}$ such that $\lambda x.gp_g(x, s-1)$ is the gap function associated with $em \oplus sm$.*

Let ϕ be the CLTL formula defining the winning condition of a single-sided CLTL game and let k be its X-length. Let $\mathcal{F}$ be the set of all s-frames for $s \in [0, k]$. For technical convenience, we let $\mathcal{F}$ include the trivial 0-frame $\perp = (\leq_\perp, gp_\perp)$, where $\leq_\perp$ is the trivial total pre-order on the empty set and $gp_\perp$ is the trivial function on the empty domain.

▶ **Definition 12** (Winning strategy trees). *A strategy tree is a function $T\colon G^* \to \mathcal{F}$ such that for every node $\eta \in G^*$, $T(\eta)$ is a $\lceil |\eta| \rceil_{k+1}$-frame and for every $gp \in G$, $(T(\eta), T(\eta \cdot gp))$ is one-step compatible and $(gp, T(\eta \cdot gp))$ is gap compatible. A function L is said to be a labeling function if for every node $\eta \in G^*$, $L(\eta)\colon V \to \mathbb{Z}$ is a mapping of the variables in V. For an infinite path π in T, let $T(\pi)$ (resp. $L(\pi)$) denote the infinite sequence of frames (resp. mappings) labeling the nodes in π, except the root node ϵ. A winning strategy tree is a pair (T, L) such that T is a strategy tree and L is a labelling function satisfying the condition that for every infinite path π, $T(\pi) = \mu(L(\pi))$ and $T(\pi), k \models_s \phi$.*

The last condition above means that $T(\pi)$ is the symbolic model associated with the concrete model $L(\pi)$ and that it symbolically satisfies the formula ϕ.

Two concrete models may have the same symbolic model associated with them, if they differ only slightly, as explained next. Two concrete models σ_1, σ_2 are said to coincide on V^a if $\sigma_1(i) \restriction V^a = \sigma_2(i) \restriction V^a$ for all $i \geq 0$. They are said to coincide on V^b up to gap functions if for every $i \geq 0$, the same gap function is associated with $\sigma_1(i) \restriction V^b$ and $\sigma_2(i) \restriction V^b$. The following result follows directly from definitions.

▶ **Proposition 13** (similar concrete models have the same symbolic model). *If two concrete models coincide on V^a and they coincide on V^b up to gap functions, then they have the same symbolic model associated with them.*

The following result accomplishes the first stage of the decidability proof, reducing the existence of winning strategies to non-emptiness of a set of trees. A detailed proof of this result can be found on the arXiv version of the paper with the same title.

▶ **Lemma 14** (strategy to tree). `System` *has a winning strategy in the single-sided CLTL game with wining condition ϕ iff there exists a winning strategy tree.*

Proof idea. If `environment` chooses a mapping $em\colon EV^b \to \mathbb{Z}$ in the CLTL game, the corresponding choice in the tree T is to go to the child gp, the gap function associated with em. `System` responds with the mapping $L(gp) \restriction SV^a$ for the look-ahead variables. For the future-blind variables SV^b, `system` chooses a mapping that ensures compatibility with the

frame $T(gp)$. This will ensure that `system`'s response and L coincide on V^a and coincide on V^b up to gap functions, so Proposition 13 ensures that both have the same symbolic model. The symbolic model symbolically satisfies ϕ by definition of wining strategy trees and Lemma 5 implies that the concrete model satisfies ϕ. ◀

Given a tree $G^* \to \mathcal{F}$, a tree automaton over finite alphabets can check whether it is a strategy tree or not, by allowing transitions only between one-step and gap compatible frames. However, to check whether it is a winning strategy tree, we need to check whether there exists a labeling function L, which is harder. One way to check the existence of such a labeling function is to start labeling at the root and inductively extend to children. Suppose there are two variables x, y at some node and we have to label them with integers. There may be many variables in other nodes whose labels should be strictly between those of x, y in the current node. So our labels for x, y in the current node should leave a gap large enough to accommodate others that are supposed to be in between. Next we introduce some orderings we use to formalize this.

A node variable in a strategy tree T is a pair (η, x) where η is a node and $x \in V^a$ is a look-ahead variable. The tree induces an order on node variables as follows. Suppose η is a node, $T(\eta)$ is an s-frame for some s and η_a is an ancestor of η such that the difference in height $h = |\eta| - |\eta_a|$ between the descendant and ancestor is at most $s-1$. For look-ahead variables $x, y \in V^a$, recall that the term $X^{s-1}x$ represents the variable x in the last position of the frame $T(\eta)$, and $X^{s-1-h}y$ represents the variable y at h positions before the last one. We say $(\eta, x) \sqsubseteq_T (\eta_a, y)$ (resp. $(\eta_a, y) \sqsubseteq_T (\eta, x)$) if $X^{s-1}x \leq_{T(\eta)} X^{s-1-h}y$ (resp. $X^{s-1-h}y \leq_{T(\eta)} X^{s-1}x$). In other words, for the variables and positions captured in the frame $T(\eta)$, $\sqsubseteq_T$ is same as the total pre-order $\leq_{T(\eta)}$. We define $(\eta, x) \sqsubset_T (\eta_a, y)$ (resp. $(\eta_a, y) \sqsubset_T (\eta, x)$) if $(\eta, x) \sqsubseteq_T (\eta_a, y)$ and $(\eta_a, y) \not\sqsubseteq_T (\eta, x)$ (resp. $(\eta_a, y) \sqsubseteq_T (\eta, x)$ and $(\eta, x) \not\sqsubseteq_T (\eta_a, y)$). We define $\sqsubset_T^*$ to be the reflexive transitive closure of $\sqsubset_T$ and $\sqsubset_T^+$ to be the transitive closure of $\sqsubset_T$. Note that $\sqsubset_T^*$ and $\sqsubset_T^+$ can compare node variables that are in different branches of the tree also, though they are not total orders. We write $(\eta_1, x) \sqsubset_T^* (\eta_2, y)$ (resp, $(\eta_1, x) \sqsubset_T^+ (\eta_2, y)$) equivalently as $(\eta_2, y) \sqsupset_T^* (\eta_1, x)$ (resp. $(\eta_1, x) \sqsupset_T^+ (\eta_2, y)$). By definition, $(\eta_1, x) \sqsubset_T^+ (\eta_2, y)$ (resp.$(\eta_2, y) \sqsubset_T^+ (\eta_1, x)$) if $(\eta_1, x) \sqsubset_T^* (\eta_2, y)$ and $(\eta_2, y) \not\sqsubset_T^* (\eta_1, x)$ (resp. $(\eta_2, y) \sqsubset_T^* (\eta_1, x)$ and $(\eta_1, x) \not\sqsubset_T^* (\eta_2, y)$). $\sqsubset^+$ is irreflexive and transitive.

▶ **Definition 15** (Bounded chain strategy trees). *Suppose T is a strategy tree, η, η' are two nodes and $x, y \in V^a$ are look-ahead variables such that $(\eta, x) \sqsubset_T^+ (\eta', y)$. A chain between (η, x) and (η', y) is a sequence $(\eta_1, x_1)(\eta_2, x_2) \cdots (\eta_r, x_r)$ such that $(\eta, x) \sqsubset_T^+ (\eta_1, x_1) \sqsubset_T^+ (\eta_2, x_2) \sqsubset_T^+ \cdots \sqsubset_T^+ (\eta_r, x_r) \sqsubset_T^+ (\eta', y)$. We say r is the length of the chain. The strategy tree T is said to have bounded chains if for any two node variables (η, x) and (η', y), there is a bound N such that any chain between (η, x) and (η', y) is of length at most N.*

▶ **Lemma 16.** *A strategy tree T has a labeling function L such that (T, L) is a winning strategy tree iff T has bounded chains.*

The above lemma characterizes those strategy trees that are winning strategy trees. This is the main technical difference between CLTL games and games with register automata specifications [28, 14]. Since register automata can compare values that are arbitrarily far apart, the corresponding characterization of symbolic structures that have associated concrete structures is more involved compared to Lemma 16 above.

Detecting unbounded chains is still difficult for tree automata – to find longer chains, we may have to examine longer paths. This difficulty can be overcome if we can show that longer chains can be obtained by repeatedly joining shorter ones. We now introduce some

notation and results to formalize this. For a node η and an ancestor η_a, $\hat{T}(\eta_a, \eta)$ is the sequence of frames $T(\eta_a) \cdots T(\eta)$ labeling the path from η_a to η. A node η_1 is said to occur within the influence of (η_a, η) if η_1 occurs between η_a and η or η_1 is an ancestor of η_a and $|\eta_a| - |\eta_1| \leq s-1$, where s is the size of the frame $T(\eta_a)$. The following result follows directly from definitions.

▶ **Proposition 17** (Identical paths induce identical orders)**.** *Suppose nodes η, η' and their ancestors η_a, η'_a respectively are such that $\hat{T}(\eta_a, \eta) = \hat{T}(\eta'_a, \eta')$. Suppose η_1, η_2 occur within the influence of (η_a, η) and η'_1, η'_2 occur within the influence of (η'_a, η') such that $|\eta| - |\eta_1| = |\eta'| - |\eta'_1|$ and $|\eta| - |\eta_2| = |\eta'| - |\eta'_2|$. For any look-ahead variables x, y, $(\eta_1, x) \sqsubset^*_T (\eta_2, y)$ (resp. $(\eta_1, x) \sqsubseteq_T (\eta_2, y)$) iff $(\eta'_1, x) \sqsubset^*_T (\eta'_2, y)$ (resp. $(\eta'_1, x) \sqsubseteq_T (\eta'_2, y)$).*

For a node η, the subtree T_η rooted at η is such that for all η', $T_\eta(\eta') = T(\eta \cdot \eta')$. A tree T is called *regular* if the set $\{T_\eta \mid \eta \in G^*\}$ is finite, i.e., there are only finitely many subtrees up to isomorphism. Two nodes η, η' are said to be isomorphic if $T_\eta = T_{\eta'}$.

▶ **Lemma 18** (Pumping chains in regular trees)**.** *Suppose T is a regular tree. Then T has unbounded chains iff there exists an infinite path containing two infinite sequences $(\eta_1, x), (\eta_2, x), (\eta_3, x) \ldots$ and $(\eta'_1, y), (\eta'_2, y), (\eta'_3, y) \ldots$ such that η_{i+1} (resp. η'_{i+1}) is a descendant of η_i (resp. η'_i) for all $i \geq 1$ and satisfy one of the following conditions.*

$$\begin{array}{ccccccc} (\eta_1,x) & \sqsubset^+_T & (\eta_2,x) & \sqsubset^+_T & (\eta_3,x) \sqsubset^+_T \cdots \\ \sqsubseteq_T & & \sqsubseteq_T & & \sqsubseteq_T \\ (\eta'_1,y) & \sqsupset^*_T & (\eta'_2,y) & \sqsupset^*_T & (\eta'_3,y) \sqsupset^*_T \cdots \end{array} \quad \textit{or} \quad \begin{array}{ccccccc} (\eta_1,x) & \sqsupset^+_T & (\eta_2,x) & \sqsupset^+_T & (\eta_3,x) \sqsupset^+_T \cdots \\ \sqsupseteq_T & & \sqsupseteq_T & & \sqsupseteq_T \\ (\eta'_1,y) & \sqsubset^*_T & (\eta'_2,y) & \sqsubset^*_T & (\eta'_3,y) \sqsubset^*_T \cdots \end{array}$$

Proof idea. We can choose a chain that is long enough to contain two isomorphic nodes. The path between them can be repeated infinitely. Proposition 17 will imply that this infinite path contains an infinite chain as required. ◀

Lemma 18 says that if a regular tree has unbounded chains, it will have an infinite path containing an infinite chain. The infinite sequence of the first (resp. second) kind given in Lemma 18 is called an infinite forward (resp. backward) chain. Now we design a tree automaton $\mathcal{A}_\phi$ whose language $\mathcal{L}(\mathcal{A}_\phi)$ is an approximation of the set $\mathcal{T} = \{T \mid \exists L, (T, L)$ is a winning strategy tree$\}$ such that $\mathcal{L}(\mathcal{A}_\phi)$ is non-empty iff $\mathcal{T}$ is. Hence, the single-sided CLTL realizability problem is equivalent to checking the non-emptiness of $\mathcal{L}(\mathcal{A}_\phi)$. The tree automaton $\mathcal{A}_\phi$ is the intersection of three automata $\mathcal{A}^{\text{str}}_\phi$, $\mathcal{A}^{\text{symb}}_\phi$ and $\mathcal{A}^{\text{chain}}_\phi$, all of which read $|G|$-ary trees labeled with letters from $\mathcal{F}$. The automaton $\mathcal{A}^{\text{str}}_\phi$ accepts the set of all strategy trees, $\mathcal{A}^{\text{symb}}_\phi$ accepts the set of all trees each of whose paths symbolically satisfies the formula ϕ and $\mathcal{A}^{\text{chain}}_\phi$ accepts the set of all trees that do not have any infinite forward or backward chains. Construction of these automata are explained in detail in Appendix B.

▶ **Lemma 19.** *The* `system` *player has a winning strategy in the single-sided CLTL$(\mathbb{Z}, <, =)$ game with winning condition ϕ iff $\mathcal{L}(\mathcal{A}_\phi)$ is non-empty.*

Proof. Suppose there is a winning strategy for the `system` player in single-sided CLTL$(\mathbb{Z}, <, =)$ game with winning condition ϕ. By Lemma 14, there exists a winning strategy tree, say (T, L). Since, T is a strategy tree, $T \in \mathcal{L}(\mathcal{A}^{\text{str}})$. We know that every branch of T must symbolically satisfy ϕ and hence, $T \in \mathcal{L}(\mathcal{A}^{\text{symb}}_\phi)$. Further, since T has the labelling function L, Lemma 16 implies that T has bounded chains and thus, it cannot have any infinite forward or backward chains. So $T \in \mathcal{L}(\mathcal{A}^{\text{chain}})$. Thus, $T \in \mathcal{L}(\mathcal{A}_\phi)$.

Conversely, suppose $\mathcal{A}_\phi$ accepts a tree T. It is known that if the language of a tree automaton is non-empty, it contains a regular tree [20, Corollary 8.20]. Although this result

holds for tree automata that read infinite binary trees as inputs, the proofs can be suitably modified to work for tree automata that read $|G|$-ary trees. Hence we can conclude that $\mathcal{A}_\phi$ must accept a regular tree T'. Since, $T' \in \mathcal{L}(\mathcal{A}_\phi)$, every branch of T' must symbolically satisfy ϕ, T' must be a strategy tree and it cannot have any infinite forward or backward chains. Thus, by Lemma 18, T' must have bounded chains and hence by Lemma 16, T' must have a labelling function L' such that (T', L') is a winning strategy tree. Hence, by Lemma 14 the `system` player has a winning strategy in the single-sided CLTL$(\mathbb{Z}, <, =)$ game. ◀

Note that $\mathcal{L}(\mathcal{A}_\phi)$ is not equal to the set $\mathcal{T} = \{T \mid \exists L, (T, L)$ is a winning strategy tree$\}$ in general. As seen in the above proof, we can only guarantee that the regular trees in $\mathcal{L}(\mathcal{A}_\phi)$ are in $\mathcal{T}$. The non-regular trees in $\mathcal{L}(\mathcal{A}_\phi)$ need not be in $\mathcal{T}$.

Using the previous lemma, we get the following decidability result.

▶ **Theorem 20.** *The single-sided realizability problem for CLTL over* $(\mathbb{Z}, <, =)$ *is 2EXPTIME -complete.*

Proof. Given a formula ϕ, Lemma 19 implies that it is enough to construct the tree automaton $\mathcal{A}_\phi$ and check it for non-emptiness. From the description of the construction in Appendix B, we can see that $\mathcal{A}_\phi^{\text{str}}$, $\mathcal{A}_\phi^{\text{symb}}$ and $\mathcal{A}_\phi^{\text{chain}}$ can be constructed in 2EXPTIME in the size of ϕ. Thus, the automaton $\mathcal{A}_\phi$ can be constructed in 2EXPTIME. Now, checking non-emptiness of a parity tree automaton is decidable and the upper bound stated in [20, Corollary 8.22 (1)] implies that the single-sided realizability problem for CLTL over $(\mathbb{Z}, <, =)$ is in 2EXPTIME. Now, the realizability problem for LTL is 2EXPTIME-complete [23] and hence, the single-sided realizability problem for CLTL over $(\mathbb{Z}, <, =)$ must also be 2EXPTIME-complete. ◀

6 Discussion and Future Work

We have seen in this paper that the CLTL realizability problem is decidable over domains satisfying completion property and that the single-sided CLTL realizability problem is decidable over integers with linear order and equality. But both these problems have a high complexity (both are 2EXPTIME-complete). It would be interesting to see if there are expressive fragments of CLTL with lower complexity, like the fragments of LTL studied in [22], which work on practical examples.

We believe that single-sided CLTL games over the domain of natural numbers $(\mathbb{N}, <, =)$ are also decidable. In [13], the authors extend the automata-characterization for the satisfiability problem for CLTL over the integer domain to the domain of natural numbers. A similar extension of the tree-automata characterization for the single-sided games over integers to one for single-sided games over the naturals seems possible, although the details need to be worked out.

Despite the decidability result that we have for the single-sided CLTL games over integers, the language of the tree automaton that we construct in this paper is an approximation of the set of all winning strategy trees. We do not have a machine-theoretic representation for winning strategies yet, and this is an interesting direction for future exploration.

References

1 Parosh Aziz Abdulla, Ahmed Bouajjani, and Julien d'Orso. Deciding monotonic games. In *International Workshop on Computer Science Logic*, pages 1–14. Springer, 2003.

2 Parosh Aziz Abdulla, Richard Mayr, Arnaud Sangnier, and Jeremy Sproston. Solving parity games on integer vectors. In *International Conference on Concurrency Theory*, pages 106–120. Springer, 2013.

3 Rajeev Alur and Thomas A Henzinger. A really temporal logic. *Journal of the ACM (JACM)*, 41(1):181–203, 1994.

4 Philippe Balbiani and Condotta Jean-François. Computational complexity of propositional linear temporal logics based on qualitative spatial or temporal reasoning. In *International Workshop on Frontiers of Combining Systems*, pages 162–176. Springer, 2002.

5 Béatrice Bérard, Benedikt Bollig, Mathieu Lehaut, and Nathalie Sznajder. Parameterized synthesis for fragments of first-order logic over data words. In *FoSSaCS*, pages 97–118, 2020.

6 Marcello M Bersani, Domenico Bianculli, Schahram Dustdar, Alessio Gambi, Carlo Ghezzi, and Srđan Krstić. Towards the formalization of properties of cloud-based elastic systems. In *proceedings of the 6th international workshop on principles of engineering service-oriented and cloud systems*, pages 38–47, 2014.

7 Ashwin Bhaskar and M. Praveen. Realizability problem for constraint ltl, 2022. doi:10.48550/ARXIV.2207.06708.

8 Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data words. *ACM Transactions on Computational Logic (TOCL)*, 12(4):1–26, 2011.

9 Cristian S Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasi-polynomial time. *SIAM Journal on Computing*, 51(2):STOC17–152, 2020.

10 Alonzo Church. Logic, arithmetic, and automata. *Journal of Symbolic Logic*, 29(4), 1964.

11 Stephane Demri, Deepak D'Souza, and Régis Gascon. Temporal logics of repeating values. *Journal of Logic and Computation*, 22, October 2012. doi:10.1093/logcom/exr013.

12 Stéphane Demri and Ranko Lazić. Ltl with the freeze quantifier and register automata. *ACM Trans. Comput. Logic*, 10(3), April 2009. doi:10.1145/1507244.1507246.

13 Stéphane Demri and Deepak D'Souza. An automata-theoretic approach to constraint ltl. *Information and Computation*, 205(3):380–415, 2007. doi:10.1016/j.ic.2006.09.006.

14 Léo Exibard, Emmanuel Filiot, and Ayrat Khalimov. Church synthesis on register automata over linearly ordered data domains. *arXiv preprint arXiv:2004.12141*, 2020.

15 Erich Gradel and Wolfgang Thomas. *Automata, logics, and infinite games: a guide to current research*, volume 2500. Springer Science & Business Media, 2002.

16 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theoretical Computer Science*, 134(2):329–363, 1994.

17 Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE transactions on robotics*, 25(6):1370–1381, 2009.

18 Orna Kupferman, Nir Piterman, and Moshe Vardi. From liveness to promptness. *Formal Methods in System Design*, 34, April 2009. doi:10.1007/s10703-009-0067-z.

19 René Mazala. *Infinite Games*, pages 23–38. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002. doi:10.1007/3-540-36387-4_2.

20 Frank Nießner. Nondeterministic tree automata. In *Automata Logics, and Infinite games*, pages 135–152. Springer, 2002.

21 Nir Piterman. From nondeterministic büchi and streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3, 2007.

22 Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive (1) designs. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, pages 364–380. Springer, 2006.

23 Amir Pnueli and Roni Rosner. On the synthesis of an asynchronous reactive module. In *International Colloquium on Automata, Languages, and Programming*, pages 652–671. Springer, 1989.

24 M Praveen, Diego Figueira, and Stephane Demri. Reasoning about data repetitions with counter systems. *Logical Methods in Computer Science*, 12, 2016.

25 M Praveen, Anirban Majumdar, and Diego Figueira. Playing with repetitions in data words using energy games. *Logical Methods in Computer Science*, 16, 2020.

26 Alexander Rabinovich. Decidable extensions of church's problem. In *International Workshop on Computer Science Logic*, pages 424–439. Springer, 2009.

27 Jean-François Raskin. An introduction to hybrid automata. In *Handbook of networked and embedded control systems*, pages 491–517. Springer, 2005.

28 Pierre-Alain Reynier, Emmanuel Filiot, and Léo Exibard. Synthesis of data word transducers. *Logical Methods in Computer Science*, 17, 2021.

29 A Prasad Sistla, Moshe Y Vardi, and Pierre Wolper. The complementation problem for büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49(2-3):217–237, 1987.

30 Moshe Y Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification. In *Proceedings of the First Symposium on Logic in Computer Science*, pages 322–331. IEEE Computer Society, 1986.

A Details of Section 4

Proof of Lemma 9. ($\Rightarrow$) Suppose `system` has a winning strategy st in the CLTL game. We show that `system` has a winning strategy in the parity game. Plays in the parity game are of the form $(\bot, q_I)(\bot, q_I, pf_1)(f_1, q_1)(f_1, q_1, pf_2)(f_2, q_2)\cdots(f_i, q_i, pf_{i+1})(f_{i+1}, q_{i+1})\cdots$, where (f_i, pf_{i+1}) and (pf_{i+1}, f_{i+i}) are one-step compatible for all i. For any such play π, let $\pi \restriction i$ be $(\bot, q_I)(\bot, q_I, pf_1)(f_1, q_1)(f_1, q_1, pf_2)(f_2, q_2)\cdots(f_i, q_i, pf_{i+1})$. Let $\Pi = \{\pi \restriction i \mid \pi \text{ is a play in the parity game}, i \geq 0\}$. We will show the existence of a function $st_p \colon \Pi \to V_e \times EM \times SM$ satisfying some properties. Such a function can be used as a strategy by `system` in the parity game: for a play $\pi \restriction i$, `system`'s response (f_{i+1}, q_{i+1}) is given by st_p, i.e., $st_p(\pi \restriction i) = ((f_{i+1}, q_{i+1}), em_{i+1}, sm_{i+1})$. For such plays that `system` plays according st_p, let frames$(\pi \restriction i)$ be the symbolic model $f_1 f_2 \cdots f_{i+1}$ and let maps$(\pi \restriction i)$ be the concrete model $(em_1 \oplus sm_1)(em_2 \oplus sm_2)\cdots(em_{i+1} \oplus sm_{i+1})$.

We will show that there is a function st_p such that for all plays π that `system` plays according to st_p and all $i \geq 0$, maps$(\pi \restriction i)$ is a concrete model resulting from `system` playing the CLTL game according to st and frames$(\pi \restriction i) = \mu(\text{maps}(\pi \restriction i))$. We will define such a function st_p by induction on i. We assume this has been done for i and show how to extend to $i+1$. We have $\pi \restriction (i+1) = (\bot, q_I)(\bot, q_I, pf_1)(f_1, q_1)(f_1, q_1, pf_2)(f_2, q_2)\cdots(f_i, q_i, pf_{i+1})(f_{i+1}, q_{i+1})$ $(f_{i+1}, q_{i+1}, pf_{i+2})$. By induction hypothesis, $f_1 f_2 \cdots f_{i+1} = \mu(\text{maps}(\pi \restriction i))$. Since the constraint system satisfies the completion property and (f_{i+1}, pf_{i+2}) is one-step compatible, by Proposition 7, there is a mapping $em \colon EV \to D$ such that the symbolic model induced by maps$(\pi \restriction i) \cdot em$ is $f_1 f_2 \cdots f_{i+1} \cdot pf_{i+2}$. Let $sm \colon SV \to D = st(\text{maps}(\pi \restriction i) \cdot em)$ be `system`'s response in the CLTL game according to st. Let f_{i+2} be the frame such that $f_1 f_2 \cdots f_{i+1} f_{i+2} = \mu(\text{maps}(\pi \restriction i) \cdot (em \oplus sm))$. Set $st_p(\pi \restriction (i+1))$ to be $((f_{i+2}, q_{i+2}), em, sm)$, where q_{i+2} is the state A_ϕ reaches after reading f_{i+2} in state q_{i+1}. Now, maps$(\pi \restriction (i+1))$ is a concrete model resulting from `system` playing the CLTL game according to st and frames$(\pi \restriction (i+1)) = \mu(\text{maps}(\pi \restriction (i+1)))$, as required for the inductive construction.

Let π be any infinite play in the parity game that `system` plays according to st_p. Then maps(π) is a concrete model resulting from `system` playing the CLTL game according to st and frames$(\pi) = \mu(\text{maps}(\pi))$. Since st is a winning strategy for `system`, maps$(\pi), 0 \models \phi$. We infer from Lemma 5 that frames(maps$(\pi)), k \models_s \phi$. Hence, the sequence of states $q_I, q_1, q_2, \ldots$ contained in the sequence of vertices that are visited in π satisfy the parity condition of A_ϕ. Hence, π itself satisfies the parity condition and hence `system` wins π. Hence, st_p is a winning strategy for `system` in the parity game.

($\Leftarrow$) Suppose st_p is a positional strategy for `system` in the parity game. We will show that `system` has a winning strategy st in the CLTL game. We will define st by induction on the number of rounds played. For the base case, suppose `environment` starts by choosing a mapping $em_1 \colon EV \to D$. In the parity game, let `environment` go to the vertex $(\bot, q_I, pf_1)$ in the first round, where pf_1 is the 1-partial frame associated with em_1. Let $(f_1, q_1) = st_p((\bot, q_I, pf_1))$ be `system`'s response according to st_p. Since (pf_1, f_1) is one-step compatible and the constraint system satisfies the completion property, by Proposition 7, em_1 can be extended to a mapping $em_1 \oplus sm_1 \colon V \to D$ such that f_1 is the frame associated with $em_1 \oplus sm_1$. Set $st(em_1)$ to be sm_1. After i rounds of the CLTL game, suppose $(em_1 \oplus sm_1) \cdots (em_i \oplus sm_i)$ is the resulting concrete model and let $(\bot, q_I)(\bot, q_I, pf_1)(f_1, q_1) \cdots (f_i, q_i)$ be the corresponding play in the parity game. Suppose `environment` chooses em_{i+1} in the next round. Let $pf_{i+1}, f_{i+1}, q_{i+1}, sm_{i+1}$ be obtained similarly as in the base case. Set $st((em_1 \oplus sm_1) \cdots (em_i \oplus sm_i) \cdot em_{i+1})$ to be sm_{i+1}.

Suppose $(em_1 \oplus sm_1)(em_2 \oplus sm_2) \cdots$ is an infinite play in the CLTL game that `system` plays according to st. There is a play $(\bot, q_I)(\bot, q_I, pf_1)(f_1, q_1)(f_1, q_1, pf_2)(f_2, q_2) \cdots$ in the parity game that is winning for `system`. This satisfies the parity condition, hence A_ϕ accepts the symbolic model $f_1 f_2 \cdots$. The symbolic model $f_1 f_2 \cdots$ is the one associated with $(em_1 \oplus sm_1)(em_2 \oplus sm_2) \cdots$ by construction of st, so Lemma 5 implies that $(em_1 \oplus sm_1)(em_2 \oplus sm_2) \cdots, 0 \models \phi$. Hence, st is a winning strategy for `system` in the CLTL game. ◀

B Details of Section 5

Proof of Lemma 16. ($\Rightarrow$) Suppose T has a labeling function L such that (T, L) is a winning strategy tree. Since for every infinite path π, $T(\pi) = \mu(L(\pi))$, L should respect the relation $\sqsubset_T^+$, i.e., if $(\eta, x) \sqsubset_T^+ (\eta', y)$, then $L(\eta)(x) < L(\eta')(y)$. Hence, any chain between (η, x) and (η', y) cannot be longer than $L(\eta')(y) - L(\eta)(x)$.

($\Leftarrow$) Suppose T has bounded chains. We construct a labeling function L such that (T, L) is a winning strategy tree. At every node η, we choose mappings for future-blind variables V^b such that the gap function associated with $L(\eta) \restriction V^b$ is $gp_{T(\eta)}$. These choices can be done independently for every node. For look-ahead variables, we construct L for every node by induction on depth of the node such that for any node variables $(\eta, x), (\eta', y)$ such that $(\eta, x) \sqsubset_T^+ (\eta', y)$ and $L(\eta), L(\eta')$ have been constructed, $L(\eta')(y) - L(\eta)(x)$ is at least as large as the length of the longest chain between (η, x) and (η', y). For the base case $\eta = \epsilon$, let $L(\eta)$ be the trivial mapping on the empty domain.

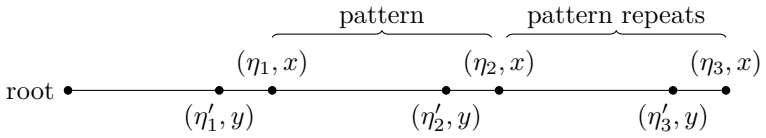
For the induction step, consider a node η. Let $(\eta, x_0), (\eta, x_1), \ldots$ be the node variables from η and let $(\eta_1, y_1), (\eta_2, y_2), \ldots$ be the node variables from all the ancestors of η. Arrange them in ascending order according to $\sqsubset_T^*$. In this arrangement, suppose $(\eta_i, y_i)(\eta, x_j)(\eta, x_{j+1}) \cdots (\eta, x_l)(\eta_{i+1}, y_{i+1})$ is a contiguous sequence of node variables from η surrounded by ancestor node variables (η_i, y_i) and (η_{i+1}, y_{i+1}). Set $L(\eta)(x_j)$ to be the sum of $L(\eta_i)(y_i)$ and the length of the longest chain between (η_i, y_i) and (η, x_j). Set $L(\eta)(x_{j+1})$ to be the sum of $L(\eta)(x_j)$ and the length of the longest chain between (η, x_j) and (η, x_{j+1}). Continue this way till (η, x_l). The value set for $L(\eta)(x_l)$ will be less than $L(\eta_{i+1})(y_{i+1})$ minus the length of the longest chain between $L(\eta)(x_l)$ and (η_{i+1}, y_{i+1}), since by induction hypothesis, $L(\eta_{i+1})(y_{i+1}) - L(\eta_i)(y_i)$ is large enough to accommodate the longest chain between (η_i, y_i) and (η_{i+1}, y_{i+1}) (note that any chain between (η, x_j) and (η, x_{j+1}) can be concatenated with any chain between (η, x_{j+1}) and (η, x_{j+2}) and so on to form a chain between (η_i, y_i) and (η_{i+1}, y_{i+1})). This way, all contiguous sequence of node variables from η can be mapped satisfactorily. This completes the induction step and hence the proof. ◀

Proof of Lemma 18. ($\Leftarrow$) We consider the first case; the other case is similar. Since $(\eta_i, x) \sqsubseteq_T (\eta'_i, y) \sqsubset^*_T (\eta'_{i-1}, y) \sqsubset^*_T \cdots \sqsubset^*_T (\eta'_1, y)$ for all $i \geq 1$, we have $(\eta_i, x) \sqsubset^*_T (\eta'_1, y)$. Hence, $(\eta_1, x) \sqsubset^+_T (\eta_2, x) \sqsubset^+_T \cdots \sqsubset^+_T (\eta_i, x) \sqsubset^*_T (\eta'_1, y)$ for all $i \geq 1$, demonstrating that there are chains of unbounded lengths between (η_1, x) and (η'_1, y).

($\Rightarrow$) We show the existence of a short segment that can be repeated arbitrarily many times to get the required infinite path. We show that there are node variables along a path satisfying the following conditions:

1. $\begin{array}{ccc} (\eta_1,x) & \sqsubset^+_T & (\eta_2,x) \\ \sqcap\!\!| & & \sqcap\!\!| \\ (\eta'_1,y) & \sqsupset^*_T & (\eta'_2,y) \end{array}$ or $\begin{array}{ccc} (\eta_1,x) & \sqsupset^+_T & (\eta_2,x) \\ |\!\!\sqcup & & |\!\!\sqcup \\ (\eta'_1,y) & \sqsubset^*_T & (\eta'_2,y) \end{array}$,
2. the nodes are arranged as $\eta'_1, \eta_1, \eta'_2, \eta_2$ in ascending order of depth, $|\eta'_1| > k$,
3. η_1, η_2 are isomorphic, η'_1, η'_2 are isomorphic and $|\eta_1| - |\eta'_1| = |\eta_2| - |\eta'_2| \leq k$.

The node variables mentioned above are as shown below.

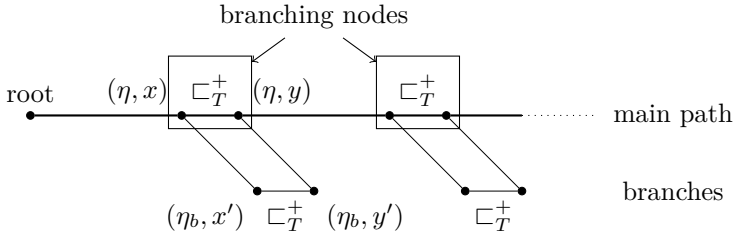


We first prove that the existence of such nodes is sufficient. Since η_1, η_2 are isomorphic, for any sequence of frames starting from η_1, the same sequence also starts from η_2. Hence there is a descendant η_3 of η_2 such that η_2, η_3 are isomorphic and $\hat{T}(\eta_1, \eta_2) = \hat{T}(\eta_2, \eta_3)$. The nodes $\eta'_1, \eta_1, \eta'_2, \eta_2$ occur within the influence of (η_1, η_2) and the nodes $\eta'_2, \eta_2, \eta'_3, \eta_3$ occur within the influence of (η_2, η_3). In the first case in the first condition above, $(\eta_1, x) \sqsubset^+_T (\eta_2, x) \sqsubseteq_T (\eta'_2, y) \sqsubset^*_T (\eta'_1, y)$ and Proposition 17 implies that $(\eta_2, x) \sqsubset^+_T (\eta_3, x) \sqsubseteq_T (\eta'_3, y) \sqsubset^*_T (\eta'_2, y)$. This pattern can be repeated arbitrarily many times, proving that there are node variables as stated in the first case of the lemma. The other case is similar.

Now we will show the existence of the short segment as claimed above. Since T is regular, the number of non-isomorphic subtrees of T is finite, say κ. Let $N = \kappa^2 |V^a|^2$. We will show subsequently that there is a chain of the form $(\eta, x_1) \sqsubset^+_T (\eta_1, y_1) \sqsubset^+_T (\eta_2, y_2) \sqsubset^+_T \cdots \sqsubset^+_T (\eta_{N+2}, y_{N+2}) \sqsubset^*_T (\eta', x_2)$ or $(\eta, x_1) \sqsupset^+_T (\eta_1, y_1) \sqsupset^+_T (\eta_2, y_2) \sqsupset^+_T \cdots \sqsupset^+_T (\eta_{N+2}, y_{N+2}) \sqsupset^*_T (\eta', x_2)$, where η_1 is a descendant of both η and η' of depth at least $(k+1)$ more than both η and η' and η_{i+1} is a descendant of η_i of depth at least $(k+1)$ more than η_i for all $i \in [1, N+1]$ (we call such chains straight segments). We will only consider the first case here; the other case is similar. Now $(\eta_{N+2}, y_{N+2}) \sqsubset^*_T (\eta', x_2)$ and η_{N+2} is a deep descendant of η' with $\eta_1, \ldots, \eta_{N+1}$ (which are themselves at least $(k+1)$ positions apart from each other) in between. Recall that $\sqsubset^*_T$ is the transitive closure of $\sqsubseteq_T$ and $\sqsubseteq_T$ holds only between node variables that are at most k positions apart. Hence, there must be intermediate node variables between $(\eta_{N+2}, y_{N+2}), (\eta', x_2)$ so that $(\eta_{N+2}, y_{N+2}) \sqsubset^*_T (\eta', x_2)$. For every $i \in [1, N+1]$, there must be some intermediate node variable (η'_i, y'_i) such that η'_i is an ancestor of η_i, $|\eta_i| - |\eta'_i| \leq k$ and $(\eta_{N+2}, y_{N+2}) \sqsubset^*_T (\eta'_i, y'_i) \sqsubset^*_T (\eta', x_2)$. Since $|\eta_i| - |\eta'_i| \leq k$, either $(\eta_i, y_i) \sqsubseteq_T (\eta'_i, y'_i)$ or $(\eta'_i, y'_i) \sqsubseteq_T (\eta_i, y_i)$ (the frame $T(\eta_i)$ spans η'_i also; hence the frame imposes an order between the node variables). If $(\eta'_i, y'_i) \sqsubseteq_T (\eta_i, y_i)$, then $(\eta_i, y_i) \sqsubset^+_T (\eta_{N+2}, y_{N+2}) \sqsubset^*_T (\eta'_i, y'_i) \sqsubseteq_T (\eta_i, y_i)$ implies that $(\eta_i, y_i) \sqsubset^+_T (\eta_i, y_i)$, contradicting the fact that $\sqsubset^+_T$ is irreflexive. Hence, $(\eta_i, y_i) \sqsubseteq_T (\eta'_i, y'_i)$. Consider the sequence $(\eta_1, y_1), (\eta'_1, y'_1), (\eta_2, y_2), (\eta'_2, y'_2), \ldots, (\eta_{N+1}, y_{N+1}), (\eta'_{N+1}, y'_{N+1})$. Since $N = \kappa^2 |V^a|^2$, there are i, j such that η_i (resp. η'_i) is isomorphic to η_j (resp. η'_j), $y_i = y_j$ and $y'_i = y'_j$. The node variables $(\eta_i, y_i), (\eta_j, y_i), (\eta'_i, y'_i), (\eta'_j, y'_i)$ satisfy the conditions required for $(\eta_1, x), (\eta_2, x), (\eta'_1, y), (\eta'_2, y)$ respectively in our claim about the existence of a short segment.

Next we will show that there are chains that go arbitrarily deep in a single branch. Suppose there are chains of unbounded lengths between (η_1, x_1) and (η_2, x_2). All such chains must pass through the least common ancestor (say η_a) of η_1, η_2. For some variable x_a, there must be chains of unbounded lengths between either (η_1, x_1) and (η_a, x_a) or between (η_a, x_a) and (η_2, x_2). Say there are unbounded chains between (η_1, x_1) and (η_a, x_a); the other case is similar. There is only one path between η_1 and η_a, so there must be chains of unbounded lengths that go beyond this path and come back. There must be node variables $(\eta_1, y_1), (\eta_1, y_2)$ or $(\eta_a, y_1), (\eta_a, y_2)$ such that there are chains of unbounded lengths between them. We will consider $(\eta_1, y_1), (\eta_1, y_2)$; the other case is similar. For the chains of unbounded lengths starting from (η_1, y_1) and ending at (η_2, y_2), let η be the highest node (nearest to the root) visited. There must be $(\eta, z_1), (\eta, z_2)$ such that there are chains of unbounded lengths between them that only visit descendants of η. If there is a bound (say B) on how deep the chains go below η and come back, the number of nodes that can be visited is bounded by the number of node variables that occur in the subtree of height B rooted at η (a node can occur at most once in a chain; otherwise, it will contradict the fact that $\sqsubset_T^+$ is irreflexive). Hence, for any bound B, there are chains that go deeper than B and come back.

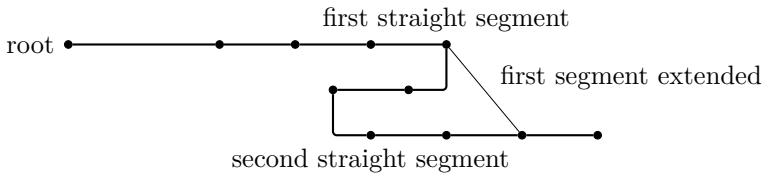
Next we prove that there is no bound on the number of node variables in a single path that belong to a chain. For this, first suppose that there is a node η and a chain goes down one child of η starting from (η, x), comes back to η via (η, y) and goes down another child. Then we have $(\eta, x) \sqsubset_T^+ (\eta, y)$ or $(\eta, y) \sqsubset_T^+ (\eta, x)$ (see the illustration below; if $(\eta, x) \sqsubset_T^* (\eta_b, x') \sqsubset_T^+ (\eta_b, y') \sqsubset_T^* (\eta, y)$ in the branch, we have $(\eta, x) \sqsubset_T^+ (\eta, y)$ in the main path by transitivity). Hence, every such node contributes a node variable in a chain.



So if there is no bound on the number of such branching nodes along a path, then there is no bound on the number of node variables in a single path that belong to a chain, as required. Suppose for the sake of contradiction that the number of such branching nodes along any path is bounded (by say B_1) and the number of node variables in a chain along any one path is also bounded (say by B_2). Then any chain is in a subtree with at most $|G|^{B_1}$ leaves (and hence at most as many paths) and at most B_2 node variables along any path, so the length of such chains is bounded. Hence, either the number of branching nodes along a path is unbounded or the number of node variables in a chain along a path is unbounded. Both of these imply that the number of node variables in a chain along a path is unbounded, as required.

A chain that goes deep down a path may make u-turns (first descend through descendants and then go to an ascendant or vice-versa) multiple times within the branch. We would like to prove that there is no bound on the length of chain segments that don't have u-turns (these are the straight segments that we need). Suppose for the sake of contradiction that there is a bound on the length of straight segments. Then there is no bound on the number of straight segments in a path, since we have already shown that the number of node variables in a chain along a path is unbounded. There can be only boundedly many distinct straight segments in a path of bounded depth, so the straight segments go deeper without any bound.

If there is a straight segment and another one occurs below the first one, the first straight segment can be extended by appending node variables of the second one, as can be seen in the illustration below.



This contradicts the hypothesis that length of straight segments is bounded. This shows that there are unboundedly long straight segments, completing the proof. ◀

B.1 Construction of $\mathcal{A}_\phi$

The automaton $\mathcal{A}_\phi^{\text{str}}$ has set of states $\mathcal{F}$. In state f, it can read the input label f and go to states $f_1, \ldots, f_{|G|}$ in its children, provided (f, f_i) is one-step compatible and (gp_i, f_i) is gap-compatible for all $i \in [1, |G|]$. All states are accepting in this Büchi automaton. This automaton just checks that every pair of consecutive frames along every branch of the tree is one-step compatible and gap-compatible and hence verifies that the tree accepted is a strategy tree. Now, the size of the set of states of $\mathcal{A}_\phi^{\text{str}}$ is $|\mathcal{F}|$, and the size of the transition set is $|\mathcal{F}| \times |\Sigma| \times |\mathcal{F}|^{|G|}$ where the input alphabet $\Sigma = \mathcal{F}$. Since, G is the set of all gap functions associated with mappings of the form $EV^b \to \mathbb{Z}$, by definition of G its range must be $\{0, \ldots, |EV^b|^2\}$ implying $|G| \leq |EV^b|^{(|EV^b|^2)}$. Also, from the definition of $\mathcal{F}$, we get $|\mathcal{F}| \leq 2^{(k.|V^a|)^2} \times (|V^b|^{|V^b|^2})^k$ where k is the X-length of ϕ. Thus, the size of $\mathcal{A}_\phi^{\text{str}}$ is double exponential in the size of ϕ.

The automaton $\mathcal{A}_\phi^{\text{symb}}$ checks that every path in the input tree is accepted by a Büchi automaton $\mathcal{B}_\phi^{\text{symb}}$, which ensures that the input sequence symbolically satisfies the formula ϕ. Given the Büchi automaton $\mathcal{B}_\phi^{\text{symb}}$, we first convert it to some deterministic parity automaton $\mathcal{C}_\phi^{\text{symb}}$ in exponential time in the size of $\mathcal{B}_\phi^{\text{symb}}$ and from that, it is easy to construct the parity tree automaton $\mathcal{A}_\phi^{\text{symb}}$ with the same size as $\mathcal{C}_\phi^{\text{symb}}$. The Büchi automaton $\mathcal{B}_\phi^{\text{symb}}$ needs to check symbolic satisfiability – whether an atomic formula is satisfied at a position can be decided by checking just the current frame, just like propositional LTL. Hence the standard Büchi automaton construction for LTL can be used to construct $\mathcal{B}_\phi^{\text{symb}}$ in EXPTIME [30]. Thus, the parity tree automaton $\mathcal{A}_\phi^{\text{symb}}$ can be constructed in 2EXPTIME in the size of ϕ.

Next, we describe the construction of the parity tree automaton $\mathcal{A}_\phi^{\text{chain}}$. It needs to check that there are no infinite forward or backward chains in any of the paths. For this we will first construct a Büchi word automaton that accepts all words not having an infinite forward or backward chain, convert it into a deterministic parity automaton $\mathcal{C}_\phi^{\text{chain}}$ and then as before, construct $\mathcal{A}_\phi^{\text{chain}}$ with the same size as $\mathcal{C}_\phi^{\text{chain}}$. This Büchi word automaton can be constructed by complementing the Büchi automaton $\mathcal{B}^{\text{chain}}$ which accepts all words that contain an infinite forward chain or an infinite backward chain in EXPTIME in the size of $\mathcal{B}^{\text{chain}}$ [29]. The construction of such a Büchi automaton $\mathcal{B}^{\text{chain}}$ is already described in [13]. The size of $\mathcal{B}^{\text{chain}}$ (as described in [13]) is polynomial in the size of the CLTL formula ϕ and hence, the size of $\mathcal{A}_\phi^{\text{chain}}$ is double exponential in the size of ϕ.