Complexity Bounds for Muller Games [1]

Paul Hunter [a], Anuj Dawar [b]

[a] *Oxford University Computing Laboratory, UK*

[b] *University of Cambridge Computer Laboratory, UK*

Abstract

We consider the complexity of infinite games played on finite graphs. We establish a framework in which the expressiveness and succinctness of different types of winning conditions can be compared. We show that the problem of deciding the winner in Muller games is PSPACE-complete. This is then used to establish PSPACE-completeness for Emerson-Lei games and for games described by Zielonka DAGs. Adaptations of the proof show PSPACE-completeness for the emptiness problem for Muller automata as well as the model-checking problem for such automata on regular trees. We also show CO-NP-completeness for two classes of union-closed games: games specified by a basis and superset Muller games.

1 Introduction

Recent years have seen an increasing use of two-player infinite games as a means of modelling reactive and concurrent systems. Games have emerged as essential tools for the analysis, synthesis and verification of such systems with a close connection to logic and to automata on infinite objects. The general framework consists of games played on finite or infinite graphs (whose vertices represent a state space) with players moving a token along the edges of the graph. The (possibly infinite) sequence of vertices that is visited constitutes a *play* of the game with the winner of a play being defined by some predetermined condition.

When we are concerned with algorithmic issues surrounding such games, we need to restrict ourselves to games that can be described in a finite fashion.

Email addresses: paul.hunter@comlab.ox.ac.uk (Paul Hunter), anuj.dawar@cl.cam.ac.uk (Anuj Dawar).

[1] An extended abstract of this paper appeared at MFCS 2005 [7]. Some material presented here first appeared in [6].

This does not mean that the graph on which the game is played is necessarily finite as it is possible to finitely describe an infinite graph. Nor does having a finite game graph by itself guarantee that the game can be finitely described. Even with two nodes in a graph, the number of distinct plays can be uncountable and there are more possible winning conditions than one could possibly describe. In this paper, we are concerned with Muller games played on finite graphs. These are games in which the graph is finite and the winner of a play is determined by the set of vertices of the graph that are visited infinitely often in the play (see Section 2 for formal definitions). This category of games is wide enough to include most kinds of game winning conditions that are considered in the literature, including Streett, Rabin, Büchi and parity games.

Specifically, we are concerned with the problem of deciding, given a game and a starting position, which player has a strategy for winning the game. It is well-known that Muller games are determined, i.e. one of the players has a winning strategy and the problem of determining which player has such a strategy is decidable [1]. We are interested in the computational complexity of deciding the winner. Since the complexity is measured as a function of the length of the description, this in turn depends on how exactly the game is described. In general, a Muller game is defined by an arena (V, V_0, V_1, E, v_I), and a winning condition $\mathcal{F} \subseteq \mathcal{P}(V)$ consisting of a set of subsets of V. One could specify $\mathcal{F}$ by listing all its elements explicitly (we call this an *explicit* presentation) but one could also adopt a formalism which allows one to specify $\mathcal{F}$ more succinctly. In the latter case, there are two possibilities. Either the formalism is general enough that any winning condition $\mathcal{F} \subseteq \mathcal{P}(V)$ can be expressed in it or there is only a restricted class of winning conditions that can be expressed. An example of the first case are Muller games presented by sets of colours, while Rabin, Streett, Büchi and parity games are all examples of the second case. Since the number of possible winning conditions $\mathcal{F}$ is $2^{2^{|V|}}$, if the formalism is general enough to describe any winning condition then the description of the game must, in general, be exponential in the size of the game graph. However, some presentations may still be more succinct than the explicit presentation. On the other hand, if the formalism is restricted in its expressive power, it may be possible that the length of a description of the game is always bounded by a polynomial in the size of the graph. We investigate these two dimensions of variation in the description of games – the expressive power of the formalism on the one hand and its succinctness on the other – in the results we establish.

As an example, consider a min-parity winning condition. Here, the winning condition is specified by a priority function $\Omega : V \to \{0, \ldots, d\}$. This is treated as a specification of the set $\mathcal{F}$ consisting of those sets $I \subseteq V$ such that the smallest number in $\Omega(I)$ is even. It is clear that the description of Ω is bounded in length by a polynomial (indeed, linear) function of $|V|$. It is also clear that not every set $\mathcal{F} \subseteq \mathcal{P}(V)$ can be described in this way. On the other hand,

there are such sets $\mathcal{F}$ for which the description using a priority function is exponentially more succinct than an explicit presentation.

The exact computational complexity of deciding the winner of a parity game is a central open question in the theory of graph games. It is known to be in NP$\cap$ co-NP [3] and conjectured by some to be in Ptime. However, lower bounds on the complexity of any class of games are hard to come by. It is known that deciding games specified by the Rabin condition is NP-complete [3] and for the Streett condition the problem is co-NP-complete. Both of these are condition types that are restricted in that they cannot express all Muller games. No lower bounds are previously known for formalisms that are expressive enough to specify all Muller games, though algorithms for such games have been studied which establish, for instance, that the games are decidable in Pspace.

We consider six general-purpose formalisms. Our main result is that the problem of deciding the winner of a Muller game is Pspace-complete. We then use this to establish Pspace-completeness for three further general-purpose representations: Emerson-Lei games, where the winning condition is presented as a Boolean formula over the vertices of the graph; games with a winning condition presented as a circuit; and the case where the winning condition is represented as a Zielonka DAG. The latter is a data structure (defined in Definition 3.7) based on the Zielonka trees of [17]. We define a notion of polynomial-time *translatability* between formalisms. A formalism is translatable into another if the representation of a game in the first can be transformed into a representation *of the same game* in the second. This is stronger than polynomial-time reducibility of the corresponding decision problems. We show that games with a Muller winning condition are translatable to Zielonka DAGs and Emerson-Lei games, both of which are in turn translatable to circuit games, but the reverse translations do not hold. Our hardness result for Muller games is based on the presentation of these games which includes a colouring of the vertices. This allows for more succinct descriptions than the explicit presentation of sets. Indeed, we show that there is a translation in one direction but not the other. The complexity of deciding the winner of the games where the sets are explicitly presented remains an open question. As an aside, we also show that the Pspace-completeness result for Muller games holds even when the game arenas are restricted to small tree-width.

We also consider the restriction to games where the winning condition $\mathcal{F}$ is closed under unions. The question of lower-bounds for *union-closed* games was posed by Khoussainov (see [8]). It is known that deciding whether or not Player 0 wins such a game is decidable in co-NP. The precise formalism used to describe the set $\mathcal{F}$ is not relevant to this upper bound as the non-deterministic algorithm runs in time polynomial in the size of the game graph. We show, for two particular formalisms that the problem of deciding the winner is co-NP-complete. One such formalism is what we call *Basis*

games while the other is the *superset Muller* games defined in [10]. The former is expressive enough to define all union-closed games while the latter is restricted to expressing sets $\mathcal{F}$ that are upward-closed. Both are, as we show, more succinct than an explicit representation of $\mathcal{F}$.

An adaptation of the PSPACE-completeness result shows that two important problems related to Muller *automata* are also PSPACE-complete. These are the emptiness problem and the model-checking problem on regular trees.

2 Preliminaries

In this section we present the definitions of arenas, games and strategies that we use throughout the paper. The definitions we use follow [5].

An arena is a generalization of a transition system where two entities or *players* control the transitions. More precisely, an *arena* is a tuple $\mathcal{A} := (V, V_0, V_1, E, v_I)$ where:

- (V, E) is a directed graph,
- V_0, the set of *Player 0 vertices*, and V_1, the set of *Player 1 vertices*, form a partition of V, and
- $v_I \in V$ is the *initial vertex*.

Viewing arenas as directed graphs with some additional structure, we define the notions of *subarena* and *induced subarena* in the obvious way.

Given an arena, $\mathcal{A}$, we consider the following set of interactions between two players: Player 0 and Player 1.[2] A token, or pebble, is placed on $v_I(\mathcal{A})$. Whenever the pebble is on a vertex $v \in V_0(\mathcal{A})$, Player 0 chooses a successor of v and moves the pebble to that vertex, and similarly when the pebble is on a vertex $v \in V_1(\mathcal{A})$, Player 1 chooses the move. This results in a (possibly infinite) sequence of vertices visited by the pebble. We call such a sequence a *play*. More formally, a *play in $\mathcal{A}$ (from v)* is a (possibly infinite) sequence of vertices $v_1 v_2 \cdots$ such that $v_1 = v$ and for all $i \geq 1$, $(v_i, v_{i+1}) \in E(\mathcal{A})$. If v is not specified, we assume the play is from $v_I(\mathcal{A})$. The set of all plays in $\mathcal{A}$ from $v_I(\mathcal{A})$ is denoted by $\mathrm{Plays}(\mathcal{A})$.

Arenas and plays establish the interactions that we are concerned with. We now use these to define games by imposing outcomes for plays. The games we are interested in are zero-sum games, that is, if one player wins then the other player loses. We can therefore define a winning condition as a set of plays

[2] For convenience we use the feminine pronoun for Player 0 and the masculine pronoun for Player 1.

that are winning for one player, say Player 0, working on the premise that if a play is not in that set then it is winning for Player 1. A *game* is a pair $\mathbb{G} := (\mathcal{A}, \mathrm{Win})$ where $\mathcal{A}$ is an arena and $\mathrm{Win} \subseteq \mathrm{Plays}(\mathcal{A})$. For $\pi \in \mathrm{Plays}(\mathcal{A})$ if $\pi \in \mathrm{Win}$, we say π is *winning for Player 0*, otherwise π is *winning for Player 1*.

As we mentioned earlier, to consider algorithmic aspects of these games we need to assume that they can be finitely presented. Muller games are an important example of a class of finitely presentable games. With a Muller game, if a player cannot move then he or she loses, otherwise the outcome of an infinite play is dependent on the set of vertices visited infinitely often.

Definition 2.1 (Muller game)**.** A game $\mathbb{G} = (\mathcal{A}, \mathrm{Win})$ is a *Muller game* if $\mathcal{A}$ is finite and there exists $\mathcal{F} \subseteq \mathcal{P}(V(\mathcal{A}))$ such that for all $\pi \in \mathrm{Plays}(\mathcal{A})$:

$$\pi \in \mathrm{Win} \iff \begin{cases} \pi \text{ is finite and ends with a vertex from } V_1(\mathcal{A}), \text{ or} \\ \pi \text{ is infinite and } \{v : v \text{ occurs infinitely often in } \pi\} \in \mathcal{F}. \end{cases}$$

If $\mathbb{G}$ is a Muller game, witnessed by $\mathcal{F} \subseteq \mathcal{P}(V(\mathcal{A}))$, we write $\mathbb{G} = (\mathcal{A}, \mathcal{F})$.

The games used in the literature in the study of logics and automata are generally Muller games. In these games, the set $\mathcal{F}$ is often not explicitly given but is specified by means of a *condition*. Different types of condition lead to various different types of games. We explore this in more detail in Section 3.

Two important subclasses of Muller games which we consider are union-closed and upward-closed games. A Muller game $\mathbb{G} = (\mathcal{A}, \mathcal{F})$ is *union-closed* if for all $X, Y \in \mathcal{F}$, $X \cup Y \in \mathcal{F}$. $\mathbb{G}$ is *upward-closed* if for all $X \in \mathcal{F}$ and $Y \supseteq X$, $Y \in \mathcal{F}$.

Remark. Union-closed games are often called Streett-Rabin games in the literature, as Player 0's winning set can be specified by a set of Streett pairs and Player 1's winning set can be specified by a set of Rabin pairs.

Given a game on an arena $\mathcal{A}$ we can define a restricted game on a subarena $\mathcal{A}'$ by restricting the winning condition to valid plays in the subarena. That is, given a game $\mathbb{G} = (\mathcal{A}, \mathrm{Win})$, and a subarena $\mathcal{A}'$ of $\mathcal{A}$. The *subgame induced by* $\mathcal{A}'$ is the game $\mathbb{G}' = (\mathcal{A}', \mathrm{Win}')$ where $\mathrm{Win}' = \mathrm{Win} \cap \mathrm{Plays}(\mathcal{A}')$.

2.1 Strategies

As with most games we are less interested in outcomes of single plays in the game and more interested in the existence of strategies that ensure one player wins against any choice of moves from the other player.

Definition 2.2 (Strategy)**.** Let $\mathcal{A} = (V, V_0, V_1, E, v_I)$ be an arena. A *strat-*

egy (for Player i) in $\mathcal{A}$ is a partial function $\sigma : V^*V_i \to V$ such that if $\sigma(v_1v_2\cdots v_n) = v'$ then $(v_n, v') \in E$. A play $\pi = v_1v_2\cdots$ is *consistent* with a strategy σ if for all $j < |\pi|$ such that $v_j \in V_i$, $\sigma(v_1v_2\cdots v_j) = v_{j+1}$. Given a game $\mathbb{G} = (\mathcal{A}, \mathrm{Win})$, σ is *winning* if all plays consistent with σ are winning for Player i.

Given a sequence of vertices visited, ending with a vertex in V_i, a strategy for Player i gives the vertex to which Player i should then play.

If there exists a winning strategy for Player i in a game $\mathbb{G}$, then we say Player i *wins* $\mathbb{G}$.

A useful class of strategies are those that can be defined from a fixed number of previously visited vertices. If a strategy σ has the property that for some fixed m, $\sigma(w) = \sigma(w')$ if w and w' agree on their last m letters, then we say that the strategy requires *finite memory* (of size $m-1$). If $m = 1$, we say the strategy is *memoryless* or *positional*. An important property of Muller games is that they only require winning strategies with finite memory.

Theorem 2.3 ([1])**.** *Let* $\mathbb{G} = (\mathcal{A}, \mathcal{F})$ *be a Muller game. One player has a winning strategy on* $\mathbb{G}$ *with finite memory of size at most* $|V(\mathcal{A})|!$*.*

An immediate corollary of this is that Muller games are decidable: we can check all possible strategies for both players that use at most $|V(\mathcal{A})|!$ memory, and see if the corresponding defined plays are winning. However, the complexity bounds on such an algorithm are enormous. In [12] McNaughton provided an algorithm with considerably better space and time bounds.

Theorem 2.4 ([12])**.** *Let* $\mathbb{G} = (\mathcal{A}, \mathcal{F})$ *be a Muller game with* $\mathcal{A} = (V, V_0, V_1, E, v_I)$*. Whether Player 0 has a winning strategy from* v_I *can be decided in time* $O(|V|^2|E||V|!)$ *and space* $O(|V|^2)$*.*

For union-closed games we can reduce the memory requirement for a winning strategy.

Theorem 2.5 ([9])**.** *Let* $\mathbb{G} = (\mathcal{A}, \mathcal{F})$ *be a Muller game. If* $\mathcal{F}$ *is closed under unions and Player 1 has a winning strategy, then Player 1 has a memoryless winning strategy.*

Finally, two useful tools for constructing decidability algorithms are *force-sets* and *avoid-sets*. Let $\mathcal{A}$ be an arena, and $X, Y \subseteq V(\mathcal{A})$. The set $Force^i_X(Y)$ is the set of vertices from which Player i has a strategy σ such that any play consistent with σ reaches some vertex in Y without leaving X. The set $Avoid^i_X(Y)$ is the set of vertices from which Player i has a strategy σ such that any play consistent with σ that remains in X avoids all vertices in Y.

We observe from the definitions that $Force^i_X(Y) = X \setminus Avoid^{1-i}_X(Y)$. We also observe that we may assume the strategies σ are memoryless: if Player i can force the play from v to some vertex of Y, the play to v is irrelevant.

Computing a force-set is an instance of the well-known alternating reachability problem. Nerode, Remmel and Yakhnis [13] provide an implementation of this algorithm which runs in time $O(|E(\mathcal{A})|)$, giving us the following:

Lemma 2.6 ([13]). *Let $\mathcal{A}$ be an arena. For any sets $X, Y \subseteq V(\mathcal{A})$, $Force^0_X(Y)$ can be computed in time $O(|E(\mathcal{A})|)$*

3 Winning condition presentations

As we discussed above, if we are interested in investigating the complexity of the problem of deciding Muller games, we need to consider the manner in which the winning condition is presented. As we see in Section 3.1, for many games that occur in the literature relating to logics and automata the winning condition can be expressed in a more efficient manner than simply listing the elements of $\mathcal{F}$. To formally describe such specifications, we introduce the concept of a *condition type*.

Definition 3.1 (Condition type). A *condition type* is a function $\mathfrak{A}$ which maps an arena $\mathcal{A}$ to a pair $(\mathcal{I}^{\mathcal{A}}, \models^{\mathcal{A}})$ where $\mathcal{I}^{\mathcal{A}}$ is a set and $\models^{\mathcal{A}} \subseteq \text{Plays}(\mathcal{A}) \times \mathcal{I}^{\mathcal{A}}$ is the *acceptance relation*. We call elements of $\mathcal{I}^{\mathcal{A}}$ *condition instances* (or simply, *conditions*).

A condition type is called *regular* if, for all $\pi_1, \pi_2 \in \text{Plays}(\mathcal{A})$ and $\Omega \in \mathcal{I}^{\mathcal{A}}$, if the set of elements of $V(\mathcal{A})$ occuring infintely often in π_1 and π_2 are the same then $\pi_1 \models^{\mathcal{A}} \Omega$ if, and only if, $\pi_2 \models^{\mathcal{A}} \Omega$.

Remark. (1) In the sequel we will generally regard the relation $\models^{\mathcal{A}}$ as intrinsically defined, and associate $\mathfrak{A}(\mathcal{A})$ with the set $\mathcal{I}^{\mathcal{A}}$. That is, we will use $\Omega \in \mathfrak{A}(\mathcal{A})$ to indicate $\Omega \in \mathcal{I}^{\mathcal{A}}$.

(2) For a regular condition type, we can identify the relation $\models^{\mathcal{A}}$ with a subset of $\mathcal{P}(V(\mathcal{A})) \times \mathcal{I}^{\mathcal{A}}$. We implicitly make this identification in the sequel.

A condition type defines a family of games in the following manner. Let $\mathfrak{A}$ be a condition type, $\mathcal{A}$ an arena, and $\mathfrak{A}(\mathcal{A}) = (\mathcal{I}^{\mathcal{A}}, \models^{\mathcal{A}})$. For $\Omega \in \mathcal{I}^{\mathcal{A}}$, the game $(\mathcal{A}, \Omega)$ is the game $(\mathcal{A}, \text{Win})$ where $\text{Win} = \{\pi \in \text{Plays}(\mathcal{A}) : \pi \models^{\mathcal{A}} \Omega\}$. That is, we treat each condition instance as a specification of a set of plays. Clearly, a regular condition type defines a family of Muller games. In the sequel we are only concerned with regular condition types, and we generally drop the qualifier "regular".

We generally call a game where the winning condition is specified by a condition of type $\mathfrak{A}$ an $\mathfrak{A}$-*game*, for example a *parity game* is a game where the winning condition is specified by a *parity condition*. We can now state precisely the decision problem we are interested in investigating.

$\mathfrak{A}$-Game	
Instance:	A game $\mathbb{G} = (\mathcal{A}, \Omega)$ where $\Omega \in \mathfrak{A}(\mathcal{A})$.
Problem:	Does Player 0 have a winning strategy in $\mathbb{G}$?

3.1 Examples

We now give some examples of condition types that occur in the literature. First we observe that an instance $\Omega \in \mathfrak{A}(\mathcal{A})$ of a regular condition type $\mathfrak{A}$ defines a family of subsets of $V(\mathcal{A})$:

$$\mathcal{F}_\Omega := \{I \subseteq V(\mathcal{A}) : I \models^{\mathcal{A}} \Omega\}.$$

We call this the *set specified by the condition* Ω. In the examples below, we describe the set specified by a condition to define the acceptance relation $\models^{\mathcal{A}}$.

3.1.1 General purpose condition types

The first examples we consider are general purpose formalisms in that they may be used to specify any family of sets, and therefore any Muller game.

The most straightforward presentation of the winning condition of a Muller game $(\mathcal{A}, \mathcal{F})$ is given by explicitly listing all elements of $\mathcal{F}$. We call this an *explicit presentation*. We can view such a formalism in our framework as follows:

Definition 3.2 (Explicit condition type)**.** An instance of the *explicit condition type* is a set $\mathcal{F} \subseteq \mathcal{P}(V(\mathcal{A}))$. The set specified by an instance is the set itself.

In the literature an explicit presentation is sometimes called a *Muller condition*. However, we reserve that term for the more commonly used presentation for Muller games in terms of colours given next.

Definition 3.3 (Muller condition type)**.** An instance of the *Muller condition type* is a pair $(\chi, \mathcal{C})$ where, for some set C, $\chi : V(\mathcal{A}) \to C$ and $\mathcal{C} \subseteq \mathcal{P}(C)$. The set $\mathcal{F}_{(\chi,\mathcal{C})}$ specified by a Muller condition $(\chi, \mathcal{C})$ is the set $\{I \subseteq V(\mathcal{A}) : \chi(I) \in \mathcal{C}\}$.

To distinguish Muller games from games with a winning condition specified by a Muller condition, we explicitly state the nature of the presentation of the winning condition if it is critical.

From a more practical perspective, when considering applications of these types of games it may be the case that there are vertices whose appearance in any infinite run is irrelevant. This leads to the definition of a *win-set condition*.

Definition 3.4 (Win-set condition type)**.** An instance of the *win-set condition type* is a pair $(W, \mathcal{W})$ where $W \subseteq V(\mathcal{A})$ and $\mathcal{W} \subseteq \mathcal{P}(W)$. The set $\mathcal{F}_{(W,\mathcal{W})}$ specified by a win-set condition $(W, \mathcal{W})$ is the set $\{I \subseteq V(\mathcal{A}) : W \cap I \in \mathcal{W}\}$.

Another way to describe a winning condition is as a boolean formula. Such a formalism is somewhat closer in nature than the specifications we have so far considered to the motivating problem of verifying reactive systems: requirements of such systems are more readily expressed as logical formulas. Winning conditions of this kind were considered by Emerson and Lei [4].

Definition 3.5 (Emerson-Lei condition type)**.** An instance of the *Emerson-Lei condition type* is a boolean formula φ with variables from the set $V(\mathcal{A})$. The set $\mathcal{F}_\varphi$ specified by an Emerson-Lei condition φ is the collection of sets $I \subseteq V(\mathcal{A})$ such that the truth assignment that maps each element of I to true and each element of $V(\mathcal{A}) \setminus I$ to false satisfies φ.

A boolean formula can contain a lot of repetition, so it may be more efficient to consider *boolean circuits* rather than formulas. This motivates one of the most succinct types of winning condition we consider.

Definition 3.6 (Circuit condition type)**.** An instance of the *circuit condition type* is a boolean circuit C with input nodes from the set $V(\mathcal{A})$ and one output node. The set $\mathcal{F}_C$ specified by a circuit condition C is the collection of sets $I \subseteq V(\mathcal{A})$ such that C outputs true when each input corresponding to a vertex in I is set to true and all other inputs are set to false.

The final general purpose formalisms we consider are somewhat more exotic. In [17], Zielonka introduced a representation for a family of subsets of a set V, $\mathcal{F} \subseteq \mathcal{P}(V)$, in terms of a labelled tree where the labels on the nodes are subsets of V.

Definition 3.7 (Zielonka tree and Zielonka DAG)**.** Let V be a set and $\mathcal{F} \subseteq \mathcal{P}(V)$. The *Zielonka tree* (also called a *split tree*) of the set $\mathcal{F}$, $\mathcal{Z}_{\mathcal{F},V}$, is defined inductively as:

(1) If $V \notin \mathcal{F}$ then $\mathcal{Z}_{\mathcal{F},V} = \mathcal{Z}_{\overline{\mathcal{F}},V}$, where $\overline{\mathcal{F}} = \mathcal{P}(V) \setminus \mathcal{F}$.
(2) If $V \in \mathcal{F}$ then the root of $\mathcal{Z}_{\mathcal{F},V}$ is labelled with V. Let $M_1, M_2, \ldots, M_k$ be the $\subseteq$-maximal sets in $\overline{\mathcal{F}}$, and let $\mathcal{F}|_{M_i} = \mathcal{F} \cap \mathcal{P}(M_i)$. The successors

of the root are the subtrees $\mathcal{Z}_{\mathcal{F}|_{M_i},M_i}$, for $1 \leq i \leq k$.

A *Zielonka DAG* is constructed as a Zielonka tree except nodes labelled by the same set are identified, making it a directed acyclic graph. Nodes of $\mathcal{Z}_{\mathcal{F},V}$ labelled by elements of $\mathcal{F}$ are called *0-level nodes*, and other nodes are *1-level nodes*.

Zielonka trees are intimately related to Muller games. In particular they identify the size of memory required for a winning strategy: the "amount" of branching of 0-level nodes indicates the maximum amount of memory required for a winning strategy for Player 0, and similarly for 1-level nodes and Player 1 [2]. For example, the 1-level nodes of a Zielonka tree of a union-closed family of sets have at most one successor, indicating that if Player 1 has a winning strategy then he has a memoryless winning strategy. Thus we also consider games where the winning condition is specified as a Zielonka tree (or the more succinct Zielonka DAG).

Definition 3.8 (Zielonka tree and Zielonka DAG condition types)**.** An instance of the *Zielonka tree (DAG) condition type* is a Zielonka tree (DAG) $\mathcal{Z}_{\mathcal{F},V(\mathcal{A})}$ for some $\mathcal{F} \subseteq \mathcal{P}(V(\mathcal{A}))$. The set specified by an instance is the set $\mathcal{F}$ used to define the instance.

3.1.2 Union-closed condition types

We now consider formalisms that can only specify families of sets that are closed under union. The first of these, the Streett condition type, introduced in [16], is useful for describing fairness conditions such as those considered by Emerson and Lei in [4].

Definition 3.9 (Streett condition type)**.** An instance of the *Streett condition type* is a set of pairs of sets of vertices $\Omega = \{(L_i, R_i) : 1 \leq i \leq m\}$. The set $\mathcal{F}_\Omega$ specified by a Streett condition Ω is the collection of sets $I \subseteq V(\mathcal{A})$ such that for all i, $1 \leq i \leq m$, either $I \cap L_i \neq \emptyset$ or $I \cap R_i = \emptyset$.

If we are interested in specifying union-closed families of sets efficiently, we can consider the closure under union of a given set. This motivates the following definition:

Definition 3.10 (Basis condition type)**.** An instance of the *basis condition type* is a set $\mathcal{B} \subseteq \mathcal{P}(V(\mathcal{A}))$. The set $\mathcal{F}_\mathcal{B}$ specified by a basis condition $\mathcal{B}$ is the collection of sets $I \subseteq V(\mathcal{A})$ such that there are $B_1, \ldots, B_n \in \mathcal{B}$ with $I = \bigcup_{1 \leq i \leq n} B_i$.

In a similar manner to the basis condition type, if we are interested in efficiently specifying an upward-closed family of sets, we can explicitly list the

$\subseteq$-minimal elements of the family. This gives us the *superset condition type*, also called a *superset Muller condition* in [10].

Definition 3.11 (Superset condition type)**.** An instance of the *superset condition type* is a set $\mathcal{M} \subseteq \mathcal{P}(V(\mathcal{A}))$. The set $\mathcal{F}_{\mathcal{M}}$ specified by a superset condition $\mathcal{M}$ is the set $\{I \subseteq V(\mathcal{A}) : M \subseteq I \text{ for some } M \in \mathcal{M}\}$.

3.2 Translations

We now present a framework in which we can compare the expressiveness and succinctness of condition types by considering transformations between games which keep the arena the same. More precisely, we define what it means for a condition type to be *translatable* to another condition type as follows.

Definition 3.12 (Translatable)**.** Given two condition types $\mathfrak{A}$ and $\mathfrak{B}$, we say that $\mathfrak{A}$ is *polynomially translatable* to $\mathfrak{B}$ if for any arena $\mathcal{A}$, with $\mathfrak{A}(\mathcal{A}) = (\mathcal{I}^{\mathcal{A}}_{\mathfrak{A}}, \models^{\mathcal{A}}_{\mathfrak{A}})$ and $\mathfrak{B}(\mathcal{A}) = (\mathcal{I}^{\mathcal{A}}_{\mathfrak{B}}, \models^{\mathcal{A}}_{\mathfrak{B}})$, there is a function $f : \mathcal{I}^{\mathcal{A}}_{\mathfrak{A}} \to \mathcal{I}^{\mathcal{A}}_{\mathfrak{B}}$ such that for all $\Omega \in \mathcal{I}^{\mathcal{A}}_{\mathfrak{A}}$:

- $f(\Omega)$ is computed in time polynomial in $|\mathcal{A}| + |\Omega|$, and
- For all $\pi \in \text{Plays}(\mathcal{A})$, $\pi \models^{\mathcal{A}}_{\mathfrak{A}} \Omega \iff \pi \models^{\mathcal{A}}_{\mathfrak{B}} f(\Omega)$.

As we are only interested in polynomial translations, we simply say $\mathfrak{A}$ is *translatable* to $\mathfrak{B}$ to mean that it is polynomially translatable. Clearly, if condition type $\mathfrak{A}$ is translatable to $\mathfrak{B}$ then the problem of deciding the winner for games of type $\mathfrak{A}$ is reducible in polynomial time to the corresponding problem for games of type $\mathfrak{B}$. That is,

Lemma 3.13. *Let $\mathfrak{A}$ and $\mathfrak{B}$ be condition types such that $\mathfrak{A}$ is translatable to $\mathfrak{B}$. Then there is a polynomial time reduction from* $\mathfrak{A}$-Game *to* $\mathfrak{B}$-Game.

If condition type $\mathfrak{A}$ is not translatable to $\mathfrak{B}$ this may be for one of three reasons. Either $\mathfrak{A}$ is more expressive than $\mathfrak{B}$ in that there are sets $\mathcal{F}$ that can be expressed using conditions from $\mathfrak{A}$ but no condition from $\mathfrak{B}$ can specify $\mathcal{F}$; or there are some sets for which the representation of type $\mathfrak{A}$ is necessarily more succinct; or the translation, while not size-increasing, can not be computed in polynomial time. We are primarily interested in the second situation. Formally, we say

Definition 3.14 (Succinctness)**.** $\mathfrak{A}$ is *more succinct* than $\mathfrak{B}$ if $\mathfrak{B}$ is translatable to $\mathfrak{A}$ but $\mathfrak{A}$ is not translatable to $\mathfrak{B}$.

We now consider translations between some of the condition types we defined in Section 3.1.

3.2.1 Translations between general purpose condition types

It is straightforward to show that win-set conditions are more succinct than explicit presentations. To translate an explicitly presented game $(\mathcal{A}, \mathcal{F})$ to a win-set condition, simply take $W = V(\mathcal{A})$ and $\mathcal{W} = \mathcal{F}$. To show that win-set conditions are not translatable to explicit presentations, consider a game where $W = \emptyset$ and $\mathcal{W} = \{\emptyset\}$. The set $\mathcal{F}_{(W,\mathcal{W})}$ specified by this condition consists of all subsets of $V(\mathcal{A})$ and thus an explicit presentation must be exponential in length.

Proposition 3.15. *The win-set condition type is more succinct than an explicit presentation.*

Similarly, there is a trivial translation from the Emerson-Lei condition type to the circuit condition type. However, the question of whether there is a translation in the other direction is an important open problem in the field of circuit complexity (see [15, Problem 15.5.4]).

We now show, through the next theorems, that circuit presentations are more succinct than Zielonka DAG presentations, which, along with Emerson-Lei presentations, are more succinct than Muller presentations, which are in turn more succinct than win-set presentations.

Theorem 3.16. *The Muller condition type is more succinct than the win-set condition type.*

Proof. Given a win-set game $\Big(\mathcal{A}, (W, \mathcal{W})\Big)$, we construct a Muller condition describing the same set of subsets as $(W, \mathcal{W})$. For the set of colours we use $C = W \cup \{c\}$, where c is distinct from any element of W. The colouring function $\chi : V(\mathcal{A}) \to C$ is then defined as:

- $\chi(w) = w$ for $w \in W$,
- $\chi(v) = c$ for $v \notin W$.

The family $\mathcal{C}$ of subsets of C is the set $\Big\{X, X \cup \{c\} : X \in \mathcal{W}\Big\}$. For $I \subseteq V$, if $I \subseteq W$, then $\chi(I) = I$ otherwise $\chi(I) = \{c\} \cup I$. Either way, $I \cap W$ is in $\mathcal{W}$ if, and only if, $\chi(I) \in \mathcal{C}$.

To show that there is no translation in the other direction, consider a Muller game on $\mathcal{A}$, where half of $V(\mathcal{A})$, V_r, is coloured red, the other half coloured blue, and the family of sets of colours is $\Big\{\{\mathsf{red}\}\Big\}$. The family $\mathcal{F}$ described by this condition consists of the $2^{|V(\mathcal{A})|/2} - 1$ non-empty subsets of V_r. Now consider trying to describe this family using a win-set condition. In general, for the set $\mathcal{F}'$ specified by the win-set condition $(W, \mathcal{W})$, if $v \notin W$ and $X \subseteq V(\mathcal{A})$ we have $\{v\} \cup X \in \mathcal{F}' \Leftrightarrow X \in \mathcal{F}'$. Observe that in our game there is no vertex

v that has this property: if $v \in V_r$, then $\{v\} \in \mathcal{F}$, but $\emptyset \notin \mathcal{F}$; and if $v \notin V_r$ then $\{v\} \cup V_r \notin \mathcal{F}$, but $V_r \in \mathcal{F}$. Thus our win-set, W must be equal to $V(\mathcal{A})$, and $\mathcal{W}$ is the explicit listing of the $2^{|V(\mathcal{A})|/2} - 1$ subsets of V_r. Thus $(W, \mathcal{W})$ cannot be produced in polynomial time. □

Theorem 3.17. *The Zielonka DAG condition type is more succinct than the Muller condition type.*

Proof. Given a Muller game consisting of an arena $\mathcal{A} = (V, V_0, V_1, E, v_I)$, a colouring $\chi : V \to C$ and a family $\mathcal{C}$ of subsets of C, we construct a Zielonka DAG $\mathcal{Z}_{\mathcal{F},V}$ which describes the same set of subsets of $V(\mathcal{A})$ as the Muller condition $(\chi, \mathcal{C})$. Consider the Zielonka DAG $\mathcal{Z}_{\mathcal{C},C}$, whose nodes are labelled by sets of colours. If we replace a label $L \subseteq C$ in this tree with the set $\{v \in V : \chi(v) \in L\}$ then we obtain a Zielonka DAG $\mathcal{Z}_{\mathcal{F},V}$ over the set of vertices. We argue that $\mathcal{F}$ is, in fact, the set specified by the Muller condition $(\chi, \mathcal{C})$ and then show that $\mathcal{Z}_{\mathcal{C},C}$ can be constructed in polynomial time. Since the translation from $\mathcal{Z}_{\mathcal{C},C}$ to $\mathcal{Z}_{\mathcal{F},V}$ involves an increase in size by at most a factor of $|V|$, this establishes that Muller games are translatable to Zielonka DAGs.

Let $I \subseteq V$ be a set of vertices. If $I \in \mathcal{F}$ then, by the definition of Zielonka DAGs, I is a subset of a label X of a 0-level node t of $\mathcal{Z}_{\mathcal{F},V}$ and is not contained in any of the labels of the 1-level successors of t. That is, for each 1-level successor u of t, there is a vertex $v \in I$ such that $\chi(v) \notin \chi(L_u)$ where L_u is the label of u. Moreover, $\chi(I) \subseteq \chi(X)$. Now $\chi(X)$ is, by construction, the label of a 0-level node of $\mathcal{Z}_{\mathcal{C},C}$ and we have established that $\chi(I)$ is contained in this label and is not contained in any of the labels of the 1-level successors of that node. Therefore, $\chi(I) \in \mathcal{C}$. Similarly, by interchanging 0-level and 1-level nodes, $\chi(I) \notin \mathcal{C}$ if $I \notin \mathcal{F}$.

To show that we can construct $\mathcal{Z}_{\mathcal{C},C}$ in polynomial time, observe first that every subset $X \subseteq C$ has at most $|C|$ maximal subsets. Note further that the label of any node in $\mathcal{Z}_{\mathcal{C},C}$ is either C, some element of $\mathcal{C}$ or a maximal (proper) subset of an element of $\mathcal{C}$. Thus, $\mathcal{Z}_{\mathcal{C},C}$ is no larger than $1 + |\mathcal{C}| + |C||\mathcal{C}|$. This bound on the size of the DAG is easily turned into a bound on the time required to construct it, using the inductive definition of Zielonka trees. Thus, we have shown that the Muller condition type is translatable into the Zielonka DAG condition type.

To show there is no translation in the other direction, consider the family $\mathcal{F}$ of subsets of $V(\mathcal{A})$ which consist of 2 or more elements. The Zielonka DAG which describes this family consists of $|V(\mathcal{A})| + 1$ nodes – one 0-level node labelled by $V(\mathcal{A})$, and $|V(\mathcal{A})|$ 1-level nodes labelled by the singleton subsets of $V(\mathcal{A})$. However, to express this as a Muller condition, each vertex must have a distinct colour since for any pair of vertices there is a set in $\mathcal{F}$ that

contains one but not the other. Thus, $|\mathcal{C}| = |\mathcal{F}| = 2^{|V(\mathcal{A})|} - |V(\mathcal{A})| - 1$. It follows that the translation from Zielonka DAGs to Muller conditions cannot be done in polynomial time. □

To show the remaining results, we use the following observation:

Lemma 3.18. *There is no translation from the Emerson-Lei condition type to the Zielonka DAG condition type.*

Proof. Let $V(\mathcal{A}) = V = \{x_1, \ldots, x_{2k}\}$, and consider the family of sets $\mathcal{F}$ described by the formula

$$\varphi := \bigvee_{1 \leq i \leq k} (x_{2i-1} \wedge x_{2i}).$$

Clearly $|\varphi| = O(|V(\mathcal{A})|)$. Now consider the Zielonka DAG $\mathcal{Z}_{\mathcal{F},V}$ describing $\mathcal{F}$. As $V \in \mathcal{F}$, the root of $\mathcal{Z}_{\mathcal{F},V}$ is a 0-level node labelled by V. The maximal subsets of V not in $\mathcal{F}$ are the 2^k subsets containing exactly one of $\{x_{2i-1}, x_{2i}\}$ for $1 \leq i \leq k$. Thus $\mathcal{Z}_{\mathcal{F},V}$ must have at least this number of nodes, and is therefore not constructible in polynomial time. □

Theorem 3.19. *The Emerson-Lei condition type is more succinct than the Muller condition type.*

Proof. Given a Muller game consisting of an arena $\mathcal{A}$, a colouring $\chi : V(\mathcal{A}) \to C$ and a family $\mathcal{C}$ of subsets of C, let φ be the boolean formula defined as:

$$\varphi := \bigvee_{X \in \mathcal{C}} \Big(\bigwedge_{c \in X} \Big(\bigvee_{\chi(v)=c} v \Big) \wedge \bigwedge_{c \notin X} \Big(\bigwedge_{\chi(v)=c} \neg v \Big) \Big).$$

It is easy to see that a subset $I \subseteq V(\mathcal{A})$ satisfies φ if, and only if, there is some set $X \in \mathcal{C}$ such that for all colours $c \in X$ there is some $v \in I$ such that $\chi(v) = c$ and for all colours $c' \notin X$ there is no $v \in I$ such that $\chi(v) = c'$. Since φ can clearly be constructed in time polynomial in $|\mathcal{C}| + |V(\mathcal{A})|$, it follows that there is a translation from the Muller condition type to the Emerson-Lei condition type.

For the reverse direction, we observe that as there is a translation from the Muller condition type to the Zielonka DAG condition type, if there were a translation from the Emerson-Lei condition type to the Muller condition type, this would contradict Lemma 3.18 as "translatability" is transitive. □

Theorem 3.20. *The circuit condition type is more succinct than the Zielonka DAG condition type.*

Proof. Given a Zielonka DAG game $(\mathcal{A}, \mathcal{Z}_{\mathcal{F},V})$ where $V = V(\mathcal{A})$, we define, for each node t in $\mathcal{Z}_{\mathcal{F},V}$ a boolean circuit C_t. This circuit is defined by induction on the height of t. For convenience, we associate each circuit with its output node. Suppose the label of t is X. We have the following cases:

(i) t is a 0-level ($X \in \mathcal{F}$) leaf: In this case, let $C_t = \bigwedge_{x \notin X} \neg x$.
(ii) t is a 1-level ($X \notin \mathcal{F}$) leaf: In this case, let $C_t = \bigvee_{x \notin X} x$.
(iii) t is a 0-level node with k successors $t_1, \ldots, t_k$: In this case, let $C_t = \bigwedge_{x \notin X} \neg x \wedge \bigwedge_{i=1}^{k} C_{t_i}$.
(iv) t is a 1-level node with k successors $t_1, \ldots, t_k$: In this case, let $C_t = \bigvee_{x \notin X} x \vee \bigvee_{i=1}^{k} C_{t_i}$.

We claim that the condition $\mathcal{F}$ is specified by the circuit C_r where r is the root of $\mathcal{Z}_{\mathcal{F},V}$. This formula has size at most $|V(\mathcal{A})||\mathcal{Z}_{\mathcal{F},V}|$ and is constructed in polynomial time. To show its correctness we argue by induction on the height of any node t with label X that C_t defines the restriction of $\mathcal{F}$ to X. We consider the following cases:

(i) t is a 0-level leaf. In this case any subset of X is in $\mathcal{F}$. $I \subseteq V(\mathcal{A})$ satisfies C_t if, and only if, no variable that is not in X appears in I, that is $I \subseteq X$.
(ii) t is a 1-level leaf. In this case any subset of X is not in $\mathcal{F}$. Here $I \subseteq V(\mathcal{A})$ satisfies C_t if, and only if, there is some element in I which is not in X, that is $I \not\subseteq X$.
(iii) t is a 0-level node with k successors labelled by $X_1, \ldots, X_k$. In this case any subset of X is in $\mathcal{F}$ unless it is a subset of X_i for some i, in which case whether it is in $\mathcal{F}$ is determined by nodes lower in the DAG. Here $I \subseteq V(\mathcal{A})$ satisfies C_t if, and only if, I is a subset of X and I satisfies C_{t_i} for all successors.
(iv) t is a 1-level node with k successors labelled by $X_1, \ldots, X_k$. In this case any subset of X is not in $\mathcal{F}$ unless it is a subset of X_i for some i. Here $I \subseteq V$ satisfies C_t if, and only if, either I is not contained in X, or there is some successor t_i such that I satisfies C_{t_i}.

We observe that as there is a translation from the Emerson-Lei condition type to the circuit condition type, Lemma 3.18 implies there is no translation from the circuit condition type to the Zielonka DAG condition type. □

Figure 1 summarizes the succinctness results we have so far shown, with the more succinct types towards the top. The dashed edge indicates that there is a translation but it is not known whether there is a translation in the opposite direction.

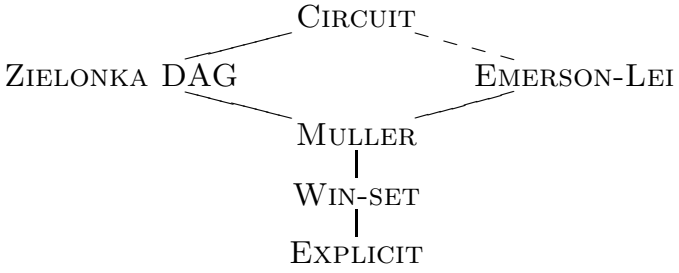

Fig. 1. Summary of the succinctness results

3.2.2 Translations between union-closed condition types

Turning to union-closed condition types, we observe that the basis condition type is a succinct way of describing union-closed sets. It is not even known if it is translatable to the circuit condition type, the most succinct type considered above. In Section 4.2 we show that the problem of deciding basis games is co-NP-complete. As we mentioned earlier, deciding Streett games is also co-NP-complete. The following result implies that we cannot use translatability to obtain upper or lower bounds on the complexity of basis games based on the known bounds for Streett games.

Theorem 3.21. *The basis and Streett condition types are incomparable with respect to translatability.*

Proof. To show there is no translation from Streett games to basis games, let $V(\mathcal{A}) = \{x_1, \ldots, x_{2k}\}$, and consider the Streett game with winning condition described by the pairs $\Big\{(L_i, \emptyset) : 1 \leq i \leq k\Big\}$, where $L_i = \{x_{2i-1}, x_{2i}\}$. Note that the family of sets described by this condition is $\mathcal{F} = \Big\{X \subseteq V(\mathcal{A}) : \forall i\ X \not\subseteq V(\mathcal{A}) \backslash L_i\Big\}$. Any basis for $\mathcal{F}$ must include the minimal elements of $\mathcal{F}$. However, the minimal elements include

$$\mathcal{M} = \Big\{\{v_1, \ldots, v_k\} : v_i \in \{x_{2i-1}, x_{2i}\}\Big\},$$

and $|\mathcal{M}| = 2^k$. Thus $\mathcal{F}$ cannot be represented by a basis constructible in polynomial time.

To show there is no translation in the other direction, let $V(\mathcal{A}) = \{x_1, \ldots, x_{2k}\}$, and consider the family $\mathcal{F}$ of sets formed by closing

$$\mathcal{B} = \Big\{\{x_{2i-1}, x_{2i}\} : 1 \leq i \leq k\Big\}$$

under union. Note that this is the same construction as for the proof of Lemma 3.18. Observe that $\mathcal{F}$ contains $2^k - 1$ sets, each with an even number of elements. Any Streett condition which describes the same family must contain at least this number of pairs in order to exclude the sets of odd cardinality.

Thus $\mathcal{F}$ cannot be represented by a Streett condition which is constructible in polynomial time. □

It should be clear that the superset condition type is translatable to the basis condition type. We include the result for completeness.

Proposition 3.22. *The superset condition type is translatable to the basis condition type.*

Proof. Let $(\mathcal{A}, \{A_1, A_2, \ldots, A_k\})$ be a superset game. The following basis:

$$\bigcup_{i=1}^{k} \left(\left\{ A_i \right\} \cup \left\{ A_i \cup \{x\} : x \notin A_i \right\} \right)$$

specifies the same family of sets and has size at most $k|V(\mathcal{A})| + 1$. □

We conclude these results with the following two observations regarding translations between explicit presentations and the basis and superset condition types.

Proposition 3.23. *The superset condition type is more succinct than an explicit presentation of an upward-closed set.*

Proof. Given an explicitly presented upward-closed game $(\mathcal{A}, \mathcal{F})$, the set $\mathcal{F}$, viewed as a superset condition, clearly describes the same set of subsets of $V(\mathcal{A})$. Conversely, for the superset game $\left(\mathcal{A}, \left\{ \{v\} : v \in V(\mathcal{A}) \right\}\right)$, the set described by the winning condition is of size $2^{|V(\mathcal{A})|} - 1$, and therefore cannot be explicitly presented in polynomial time. □

Corollary 3.24. *The basis condition type is more succinct than an explicit presentation of a union-closed set.*

Proof. The fact that the basis condition type is not translatable to an explicit presentation follows from Proposition 3.23 and Proposition 3.22 as translatability is transitive. The other direction is straightforward, the explicit presentation itself suffices as a basis. □

3.3 Extendibility

We now introduce a property of condition types that allows us to make simplifying assumptions about the arena. We say a condition type is *extendible* if it can "ignore" a set of added vertices. More precisely,

Definition 3.25 (Extendible condition type)**.** Let $\mathfrak{A}$ be a condition type. We say $\mathfrak{A}$ is *extendible* if for any arenas $\mathcal{A}$ and $\mathcal{A}'$ such that $V(\mathcal{A}) \subseteq V(\mathcal{A}')$, and any instance $\Omega \in \mathfrak{A}(\mathcal{A})$, there is an instance $\Omega' \in \mathfrak{A}(\mathcal{A}')$, computable in time polynomial in $|\Omega| + |V(\mathcal{A}')|$, such that $\mathcal{F}_{\Omega'} = \{I \subseteq V(\mathcal{A}') : I \cap V(\mathcal{A}) \in \mathcal{F}_\Omega\}$.

We observe that if $|V(\mathcal{A}')| - |V(\mathcal{A})| = m$, then $|\mathcal{F}_{\Omega'}| = 2^m |\mathcal{F}_\Omega|$, so in particular, the explicit condition type is not extendible. However, all the other condition types we have so far considered are extendible.

Proposition 3.26. *The following condition types are extendible: Muller, circuit, Emerson-Lei, Zielonka tree/DAG, win-set, Streett, parity, basis, and superset.*

Proof. Let us fix arenas $\mathcal{A}$ and $\mathcal{A}'$ such that $V(\mathcal{A}) \subseteq V(\mathcal{A}')$. We show for each condition type above how to compute the required instance Ω' from a given Ω. It follows from the definitions that for the circuit, Emerson-Lei, win-set, Streett and superset conditions taking $\Omega' = \Omega$ suffices. So let us consider the other condition types.

Suppose $\Omega = (\chi, \mathcal{C})$ is a Muller condition instance with $\chi : V(\mathcal{A}) \to C$. We define $\Omega' = (\chi', \mathcal{C}')$ as follows. Let $C' = C \cup \{c\}$ where c is not an element of C. We define

$$\chi'(v) := \begin{cases} \chi(v) \text{ if } v \in V(\mathcal{A}) \\ c \text{ otherwise} \end{cases}$$

and we define $\mathcal{C}' := \mathcal{C} \cup \{I \cup \{c\} : I \in \mathcal{C}\}$. The condition $(\chi', \mathcal{C}')$ is clearly computable in time polynomial in $|\Omega| + |V(\mathcal{A}')|$, and for every $I \subseteq V(\mathcal{A}')$ we have $\chi'(I) \in \mathcal{C}'$ if, and only if, $\chi(I \cap V(\mathcal{A})) \in \mathcal{C}$. Thus Ω' is as required.

Similarly, if $\Omega = (\chi, \mathsf{P})$ is a parity condition, we let $\mathsf{P}' = \mathsf{P} \cup \{p\}$ for some odd $p < \min\{\chi(v) : v \in V(\mathcal{A})\}$ and define $\chi'(v) = p$ for $v \notin V(\mathcal{A})$, and $\chi(v) = v$ otherwise. For any set $I \subseteq V(\mathcal{A}')$, if $I \cap V(\mathcal{A}) \neq \emptyset$ then $\max\{\chi'(v) : v \in I\} = \max\{\chi(v) : v \in I \cap V(\mathcal{A})\}$, so $I \in \mathcal{F}_{\Omega'}$ if, and only if, $I \cap V(\mathcal{A}) \in \mathcal{F}_\Omega$. Otherwise, if $I \cap V(\mathcal{A}) = \emptyset$, then $\min\{\chi'(v) : v \in I\} = p$, and as $\emptyset \notin \mathcal{F}_\Omega$ and p is odd, we have $I \notin \mathcal{F}_{\Omega'}$ and $I \cap V(\mathcal{A}) \notin \mathcal{F}_\Omega$. Thus Ω' is as required.

Given a Zielonka structure $\mathcal{Z}_{\mathcal{F},V}$ where $V = V(\mathcal{A})$, consider the Zielonka structure $\Omega' = \mathcal{Z}_{\mathcal{F}',V'}$, where $V' = V(\mathcal{A}')$, defined by adding $V(\mathcal{A}') \setminus V(\mathcal{A})$ to each label. That is, if t is a node in $\mathcal{Z}_{\mathcal{F},V}$, labelled by $X \subseteq V$, then t is a node in $\mathcal{Z}_{\mathcal{F}',V'}$ labelled by $X \cup (V(\mathcal{A}') \setminus V(\mathcal{A}))$. Now consider $I \in \mathcal{F}'$. From the definition of a Zielonka structure, I is a subset of a label of a 0-level node t and not a subset of a label of any of the successors of t. Suppose t is labelled, in $\mathcal{Z}_{\mathcal{F},V}$, by X, so $I \subseteq X \cup (V' \setminus V)$. Thus $I \cap V(\mathcal{A}) \subseteq X$. Now suppose $I \cap V(\mathcal{A})$ is a subset of Y, a label (in $\mathcal{Z}_{\mathcal{F},V}$) of a successor of t. It follows that $I \subseteq Y \cup (V' \setminus V)$, and so I is a subset of a label (in $\mathcal{Z}_{\mathcal{F}',V'}$) of a successor of t, contradicting the choice of t. So $I \cap V(\mathcal{A}) \in \mathcal{F}$. Interchanging the roles of

0-level nodes and 1-level nodes establishes that if $I \notin \mathcal{F}'$ then $I \cap V(\mathcal{A}) \notin \mathcal{F}$. Thus Ω' is as required.

Finally, given an instance of a basis condition type $\Omega = \mathcal{B}$, we define $\Omega' = \mathcal{B}'$ as follows:

$$\mathcal{B}' = \mathcal{B} \cup \Big\{ B \cup \{v\} : B \in \mathcal{B} \text{ and } v \in V(\mathcal{A}') \setminus V(\mathcal{A}) \Big\}.$$

Suppose $I = \bigcup_{i=1}^{n} B_i$ for sets $B_1, \ldots, B_n \in \mathcal{B}'$. Now, by definition of $\mathcal{B}'$, for each i, $B_i \cap V(\mathcal{A}) \in \mathcal{B}$. Thus, $I \cap V(\mathcal{A}) = \bigcup_{i=1}^{n} (B_i \cap V(\mathcal{A})) \in \mathcal{F}_\Omega$. Conversely, if $I \cap V(\mathcal{A}) \in \mathcal{F}_\Omega$, let $I \cap V(\mathcal{A}) = \bigcup_{i=1}^{n} B_i$, for $B_i \in \mathcal{B}$. Then $I = \bigcup_{v \in V(\mathcal{A}') \setminus V(\mathcal{A})} (B_1 \cup \{v\}) \cup \bigcup_{i=1}^{n} B_i$. Hence, $I \in \mathcal{F}_{\Omega'}$. □

Given a game with a winning condition specified by an extendible condition type, we can add vertices to the arena without significantly changing the size of the instance. This enables us to assume that the arena has a very simple structure.

Theorem 3.27. *Let $\mathfrak{A}$ be an extendible condition type and $\mathbb{G} = (\mathcal{A}, \Omega)$ be a Muller game with $\Omega \in \mathfrak{A}(\mathcal{A})$. Then there exists a Muller game $(\mathcal{A}', \Omega')$ with $\Omega' \in \mathfrak{A}(\mathcal{A}')$, computable in time polynomial in $||\mathbb{G}||$, such that:*

(i) $\mathcal{A}'$ is a bipartite graph with $E(\mathcal{A}') \subseteq (V_0(\mathcal{A}') \times V_1(\mathcal{A}')) \cup (V_1(\mathcal{A}') \times V_0(\mathcal{A}'))$,
(ii) All vertices in $V_0(\mathcal{A}')$ have out-degree at most 2, and
(iii) Player 0 wins $\mathbb{G}$ if, and only if, she wins $\mathbb{G}'$.

Proof. We construct $\mathcal{A}'$ from $\mathcal{A}$ in a series of stages by adding vertices and adding and replacing edges, so $V(\mathcal{A}) \subseteq V(\mathcal{A}')$. We observe that the resulting arena has size polynomial in $|\mathcal{A}|$, so it can be constructed in polynomial time. We then use the definition of extendible condition type to obtain the winning condition Ω' from Ω. Since the size of $\mathcal{A}'$ is polynomial in the size of $\mathcal{A}$, we can compute Ω' in time polynomial in $|\Omega| + |\mathcal{A}|$. It is clear from the definition of extendible condition types that in the resulting game Player 0 wins from $v_I(\mathcal{A})$ if, and only if, she wins from $v_I(\mathcal{A}')$. Thus it remains to show the first two conditions may be met with at most a polynomial increase in the size of the arena.

First we ensure all vertices in $V_0(\mathcal{A}')$ have out-degree at most 2. If $v \in V_0(\mathcal{A})$ has out-degree $m > 2$, we replace the m outgoing edges from v with a binary branching tree, rooted at v, with m leaves – the successors of v. We observe that this requires adding at most m vertices and m edges. Each of the newly added vertices are added to $V_1(\mathcal{A})$. After repeating this for all vertices in $V_0(\mathcal{A})$, the resulting arena $\mathcal{A}'$ has at most $|V(\mathcal{A})| + |E(\mathcal{A})|$ vertices, and $2|E(\mathcal{A})|$ edges, and every vertex in $V_0(\mathcal{A}')$ has out-degree at most 2.

Now suppose all vertices in $\mathcal{A}$ have out-degree at most 2. For each edge $e = (u, v) \in E(\mathcal{A})$ such that $u, v \in V_0(\mathcal{A})$ ($u, v \in V_1(\mathcal{A})$), add a vertex v_e to $V_1(\mathcal{A})$ ($V_0(\mathcal{A})$) and replace the edge e with edges (u, v_e) and (v_e, v). After repeating this for all edges in $E(\mathcal{A})$, the resulting arena $\mathcal{A}'$ has at most $|V(\mathcal{A})| + |E(\mathcal{A})|$ vertices, and $2|E(\mathcal{A})|$ edges, and $E(\mathcal{A}') \subseteq V_0(\mathcal{A}') \times V_1(\mathcal{A}') \cup V_1(\mathcal{A}') \times V_0(\mathcal{A}')$. □

4 Complexity results

In this section we consider the complexity of deciding whether Player 0 has a winning strategy in a Muller game when the winning condition is specified using some of the formalisms we have considered. We show that the problem of deciding Muller games in which the winning condition is specified by a win-set condition is PSPACE-complete. It follows from our results on translatability that the decision problems for Muller games with winning condition specified by a Muller condition, Zielonka DAG, Emerson-Lei condition or a circuit condition, are all also PSPACE-complete. We also show that the decision problems for basis and superset games are CO-NP-complete.

4.1 PSPACE-*completeness*

As we saw in Theorem 2.4, McNaughton [12] presented an algorithm for deciding Muller games in space $O(|V(\mathcal{A})|^2)$. In fact, the games he considered were win-set games. However, the algorithm is easily adapted to the case where the winning condition is presented explicitly, or as a Muller condition, a Zielonka DAG, an Emerson-Lei condition, or a circuit condition without significant increase in the space requirements. Thus, each of these classes of games is decidable in PSPACE.

We now show corresponding lower bounds. By the results of the previous section, it suffices to establish the hardness result for the win-set condition type.

Theorem 4.1. *Deciding win-set games is* PSPACE*-complete.*

Proof. By the above comments, we only need to show PSPACE-hardness. For this, we reduce the problem of QSAT (satisfiability of a quantified boolean formula [QBF]) to the problem of deciding the winner of a win-set game.

We assume, without loss of generality a QBF, $\Phi = Q_{k-1}x_{k-1} \ldots \forall x_1 \exists x_0 \varphi$ is given in which quantifiers are strictly alternating and φ is in disjunctive

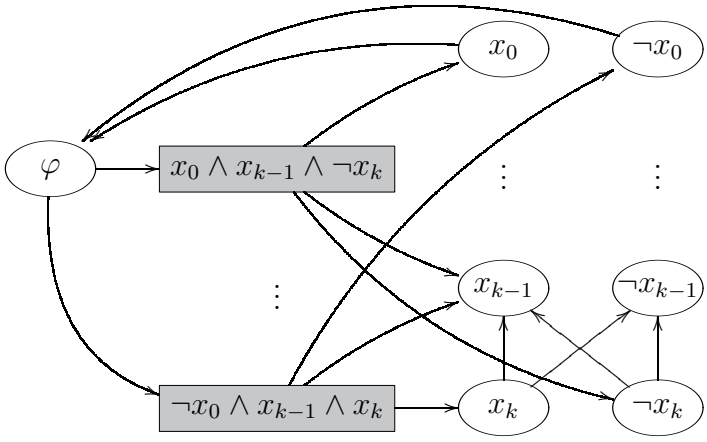

Fig. 2. Arena of $\mathbb{G}_\Phi$ for $\varphi = (x_0 \wedge x_{k-1} \wedge \neg x_k) \vee \ldots \vee (\neg x_0 \wedge x_{k-1} \wedge x_k)$

normal form with at most 3 literals per term. We then define a win-set game $\mathbb{G}_\Phi = (\mathcal{A}, \Omega)$, where $\Omega = (W, \mathcal{W})$, as follows:

- $V_0(\mathcal{A}) = \{\varphi\} \cup \{x, \neg x : \text{for all variables } x\}$,
- $V_1(\mathcal{A}) = \{t_0, \ldots, t_{m-1}\}$, the set of terms in φ,
- $E(\mathcal{A})$ given by:
  - $(\varphi, t_j) \in E(\mathcal{A})$ for $0 \leq j < m$;
  - If $t_j = (l_0 \wedge l_1 \wedge l_2)$, then $(t_j, l_0), (t_j, l_1), (t_j, l_2) \in E(\mathcal{A})$;
  - $(x_i, x_{i-1}), (x_i, \neg x_{i-1}) \in E(\mathcal{A})$ for $0 < i < k$;
  - $(\neg x_i, x_{i-1}), (\neg x_i, \neg x_{i-1}) \in E(\mathcal{A})$ for $0 < i < k$; and
  - $(x_0, \varphi), (\neg x_0, \varphi) \in E(\mathcal{A})$,
- $v_I(\mathcal{A}) = \varphi$,
- $W = V_0(\mathcal{A}) \setminus \{\varphi\}$, and $\mathcal{W}$ is

$$\mathcal{W} = \Big\{S_i, S_i \cup \{x_i\}, S_i \cup \{\neg x_i\} : 0 \leq i < k,\ i \text{ even}\Big\}$$

  where $S_0 = \emptyset$ and for $i > 0$, $S_i = \{x_j, \neg x_j : 0 \leq j < i\}$.

Figure 2 illustrates how the arena of $\mathbb{G}_\Phi$ would look if φ contained the terms $(x_0 \wedge x_{k-1} \wedge \neg x_k)$ and $(\neg x_0 \wedge x_{k-1} \wedge x_k)$.

Note that as this is a win-set game, we are only interested in vertices of W that are visited infinitely often. Observe that the winning condition ensures that Player 0 can win if, and only if, the minimum i such that at most one of x_i and $\neg x_i$ is visited infinitely often is even. The idea behind the strategy for Player 0 is to perpetually verify φ. The choice of strategies by both players then dictates the choices of the truth values for each of the variables, and the winning condition guarantees a winning strategy for Player 0 if, and only if, Φ is true. To formally show that Player 0 has a winning strategy if, and only if, Φ is true, we proceed by induction on k, the number of quantifiers of Φ.

Base case: $k = 1$ By the idempotence of $\wedge$ and $\vee$ and assuming Φ has no free variables, Φ is logically equivalent to one of the following forms.

- $\Phi = \exists x_0.x_0$ or $\exists x_0.\neg x_0$. In this case the arena consists of four vertices, $\{\varphi, t_0, x_0, \neg x_0\}$. Player 0 wins by always returning to φ from whichever of x_0 and $\neg x_0$ Player 1 is forced to play to, and Φ is clearly true.
- $\Phi = \exists x_0.(x_0 \vee \neg x_0)$. Here Φ is also true. The arena consists of five vertices $\{\varphi, t_0, t_1, x_0, \neg x_0\}$ and Player 0 has the only choice (at φ). A winning strategy is to always play from φ to t_0.
- $\Phi = \exists x_0.(x_0 \wedge \neg x_0)$. Here Φ is false. The arena consists of four vertices $\{\varphi, t_0, x_0, \neg x_0\}$ and Player 1 can force the play to visit both x_0 and $\neg x_0$ infinitely often by alternately choosing each from t_0. Note that this strategy requires memory to remember which vertex was visited last time.

Note that if x_0 does not appear in φ, we can add the term $(x_0 \wedge \neg x_0)$ without changing the truth value of Φ.

Inductive case: The inductive hypothesis asserts that if Φ has $k-1$ quantifiers and has no free variables, then Player 0 has a winning strategy if, and only if, Φ is true. To show that this implies the case for k quantifiers, we use the following lemma which shows how subgames correspond to restricted subformulas. First we introduce some notation. If x is free in Φ' and v is either true or false, we write $\Phi'[x \mapsto \mathsf{v}]$ to denote the formula obtained by substituting v for x in Φ' and simplifying. Note that if $\Phi'[x \mapsto \mathsf{true}]$ simplifies to true then Φ' must have at least one term containing the single literal x, and if it simplifies to false, then all terms contain $\neg x$. The crucial lemma can now be stated as

Lemma 4.2. *If $\Phi = Qx\Phi'$ ($Q \in \{\exists, \forall\}$) and $\Phi'[x \mapsto \mathsf{true}]$ does not simplify to true or false, then $\mathbb{G}_{\Phi'[x \mapsto \mathsf{true}]}$ is isomorphic to the subgame of $\mathbb{G}_\Phi = (\mathcal{A}, \Omega)$ induced by the set $Avoid^1_{Avoid^0_{\mathcal{A}}(\neg x)}(x)$. Dually, if $\Phi'[x \mapsto \mathsf{false}]$ does not simplify to true or false, then $\mathbb{G}_{\Phi'[x \mapsto \mathsf{false}]}$ is isomorphic to the subgame of $\mathbb{G}_\Phi$ induced by the set $Avoid^1_{Avoid^0_{\mathcal{A}}(x)}(\neg x)$.*

Proof. Let $\Phi = Qx \ldots \exists x_0 \varphi$ and $\Phi'[x \mapsto \mathsf{true}] = Q'y \ldots \exists x_0 \varphi'$. Then φ' consists of the terms of φ that do not contain $\neg x$, with all occurrences of x removed. The assumption that $\Phi'[x \mapsto \mathsf{true}]$ does not simplify to true or false implies that there is at least one such term. Thus the vertices of the arena for the game $\mathbb{G}_{\Phi'[x \mapsto \mathsf{true}]}$ have a natural embedding in the vertices of the arena for the game $\mathbb{G}_\Phi$. Furthermore, the edges of the arena for $\mathbb{G}_{\Phi'[x \mapsto \mathsf{true}]}$ are the same as those for $\mathbb{G}_\Phi$ restricted to this vertex set. We show that the subarena of $\mathbb{G}_\Phi$ induced by $Avoid^1_{Avoid^0_{\mathcal{A}}(\neg x)}(x)$ is identical. As the winning condition only depends on vertices corresponding to variables, it follows that the winning conditions are also identical.

In $\mathbb{G}_\Phi = (\mathcal{A}, \Omega)$, the set $Avoid^0_{\mathcal{A}}(\neg x)$ consists of the vertices from which Player 0 can avoid $\neg x$. As Player 1 chooses the play from vertices corresponding to terms, the set of vertices from which Player 1 can reach $\neg x$ is

$\{\neg x\} \cup \{t : \neg x \in t\}$. As there is at least one term that does not contain $\neg x$, Player 0 can play to that term to avoid $\neg x$ from φ. As x is the outermost variable of Φ, there are no other vertices that can reach $\neg x$, so

$$Avoid^0_{\mathcal{A}}(\neg x) = V(\mathcal{A}) \setminus \Big(\{\neg x\} \cup \{t : \neg x \in t\}\Big).$$

Next we consider $Avoid^1_{V'}(x)$ for $V' = Avoid^0_{\mathcal{A}}(\neg x)$. As φ does not contain a term containing x by itself, Player 0 cannot force the play to x from φ, as Player 1 can always choose to play to another literal. Furthermore, as x is the outermost variable in Φ, the only edges to x are from vertices associated with terms. Thus x is the only vertex from which Player 0 can force the play to visit x, so

$$Avoid^1_{V'}(x) = V' \setminus \{x\}.$$

Thus $Avoid^1_{Avoid^0_{\mathcal{A}}(\neg x)}(x) = V(\mathcal{A}) \setminus \Big(\{x, \neg x\} \cup \{t : \neg x \in t\}\Big)$, which is precisely the vertex set corresponding to $\mathbb{G}_{\Phi'[x \mapsto \mathsf{true}]}$. The edges for both arenas are those of $\mathbb{G}_{\Phi}$ restricted to these vertices, as are the winning conditions. Thus the two games are identical. $\dashv$

To complete the inductive step, we consider two cases.

- $\Phi = \exists x_{k-1}.\Phi'$. If Φ is true, then there is a value of v such that $\Phi'[x_{k-1} \mapsto \mathsf{v}]$ is true. Assume that $\mathsf{v} = \mathsf{true}$, the case for $\mathsf{v} = \mathsf{false}$ being similar. The winning strategy for Player 0 is then to avoid $\neg x_{k-1}$ and try to play to x_{k-1}, playing through each vertex in S_{k-1} when the latter vertex is reached. Note that to play through each vertex in S_{k-1} requires at least two visits to x_{k-1} – Player 0 must remember (the parity of) the number of times she has visited that vertex. If $\Phi'[x_{k-1} \mapsto \mathsf{v}]$ simplifies to true, then Player 0 can force the play to visit x_{k-1}, by playing to the term that only contains x_{k-1}. Otherwise Player 1 can play to avoid x_{k-1}, restricting the play to $Avoid^1_{Avoid^0_{\mathcal{A}}(\neg x_{k-1})}(x_{k-1})$. From the above lemma, this subgame is equivalent to $\mathbb{G}_{\Phi'[x_{k-1} \mapsto \mathsf{true}]}$, and from the inductive hypothesis, Player 0 has a winning strategy on this game. Thus the strategy of Player 0 is to play her winning strategy on the smaller game. If Φ is false, then Player 1 plays a strategy similar to the strategy of Player 0 in the case below.
- $\Phi = \forall x_{k-1}.\Phi'$. In this case, if Φ is true, then for both choices of truth value $\mathsf{v} \in \{\mathsf{true}, \mathsf{false}\}$, $\Phi'[x_{k-1} \mapsto \mathsf{v}]$ is true. The winning strategy for Player 0 is to alternately attempt to play to each of x_{k-1} and $\neg x_{k-1}$ (and then through all vertices in S_{k-1}), avoiding the other at the same time. If, at any point, Player 1 plays to avoid the vertex Player 0 is attempting to reach, Player 0 plays her winning strategy on the reduced game (which exists from the lemma and the inductive hypothesis). Again, if Φ is false, Player 1 plays a strategy similar to the strategy of Player 0 in the previous case. Note that in this case Player 0 cannot force the play to visit both x_{k-1} and $\neg x_{k-1}$.

$\square$

From the results on translatability in Section 3 and our observation regarding the PSPACE solvability of these games, we obtain completeness results for Muller games when the winning condition is presented as a Muller condition, Zielonka DAG, Emerson-Lei condition or a circuit condition.

Corollary 4.3. *The following problems are* PSPACE*-complete: Deciding Muller games with winning condition specified by a Muller condition, deciding Zielonka DAG games, deciding Emerson-Lei games, and deciding circuit games.*

It can be verified that an explicit presentation of the winning condition constructed in the proof of Theorem 4.1 would be exponentially larger than the presentation using a win-set. Thus, the proof cannot be used to provide a PSPACE-hardness result for the explicitly presented games. The exact complexity of deciding the winner of such games remains open. Indeed, it is conceivable (though it appears unlikely) that the problem is in PTIME. However, if the explicitly presented winning condition is an anti-chain with respect to the subset relation (that is, $X \not\subseteq Y$ for all $X, Y \in \mathcal{F}$), determining the winner is tractable. The following result generalizes a similar result for the *fully separated* games of Ishihara and Khoussainov [8].

Theorem 4.4. *Let $\mathbb{G} = (\mathcal{A}, \mathcal{F})$ be an explicitly presented Muller game such that $\mathcal{F}$ is an anti-chain. Whether Player 0 wins $\mathbb{G}$ can be decided in time $O(|\mathcal{F}||V(\mathcal{A})|^2|E(\mathcal{A})|)$.*

Proof. Consider the algorithm ANTICHAIN$(\mathcal{A}, \mathcal{F})$ in Algorithm 1. We show that it is correct and runs in time $O(|\mathcal{F}||V(\mathcal{A})|^2|E(\mathcal{A})|)$.

Algorithm 1 ANTICHAIN$(\mathcal{A}, \mathcal{F})$

Returns: true if, and only if, Player 0 has a winning strategy from $v_I(\mathcal{A})$ in $(\mathcal{A}, \mathcal{F})$ when $\mathcal{F}$ is an anti-chain.

```
for each X ∈ F do
    N_X = {v : Player 0 has a winning strategy from v in the game (A, {X})}
let N = Force^0_A(⋃_{X∈F} N_X)
if v_I(A) ∈ N then
    return true
else if N = ∅ then
    return false
else
    let F' = {X ∈ F : X ∩ N = ∅}
    return ANTICHAIN(A \ N, F')
```

We first show that ANTICHAIN$(\mathcal{A}, \mathcal{F})$ returns true if, and only if, Player 0 has a winning strategy in $\mathbb{G} = (\mathcal{A}, \mathcal{F})$. Let us suppose N has been computed as above. We consider three cases:

(i) $v_I(\mathcal{A}) \in N$. From the definition of N, there exists $v \in V(\mathcal{A})$ and $X \in \mathcal{F}$ such that Player 0 can force the play to v from $v_I(\mathcal{A})$ and Player 0 has a winning strategy from v which visits every vertex in X, and only vertices in X, infinitely often. The winning strategy for Player 0 is then to force the play to v and play this strategy. Since $X \in \mathcal{F}$, this is a winning strategy.

(ii) $N = \emptyset$. In this case, for every $X \in \mathcal{F}$, Player 1 has a strategy τ_X from every vertex in $\mathcal{A}$ which can ensure either not all vertices of X are visited infinitely often, or some vertices not in X are visited infinitely often. The strategy for Player 1 on $(\mathcal{A}, \mathcal{F})$ is now defined as follows. Play anything until the play enters some $X \in \mathcal{F}$, then play the strategy τ_X until the play leaves X. Clearly if there is no $X \in \mathcal{F}$ such that the play remains forever in X, Player 1 wins the play. So let us suppose the play remains indefinitely in X for some $X \in \mathcal{F}$. From the definition of τ_X, the set I of vertices visited infinitely often is properly contained in X. Since $\mathcal{F}$ is an anti-chain, it follows that $I \notin \mathcal{F}$. Thus Player 1 wins the play.

(iii) $N \neq \emptyset$ and $v_I(\mathcal{A}) \notin N$. In this case, Player 1 can force the play to remain in $\mathcal{A} \setminus N$ and it follows from case (i) above that Player 0 has a winning strategy from every vertex in N. Clearly, if Player 0 has a winning strategy in $(\mathcal{A} \setminus N, \mathcal{F}')$ then she has a winning strategy in the larger game: if Player 1 chooses to keep the play in $\mathcal{A} \setminus N$ then Player 0 can play her winning strategy on the subgame, otherwise if Player 1 chooses to move to a vertex in N, Player 0 can play her winning strategy from N. Conversely, if Player 1 has a winning strategy in $(\mathcal{A} \setminus N, \mathcal{F}')$ then, as he can force the play to remain in $\mathcal{A} \setminus N$, he can play his winning strategy on the subgame.

Thus, ANTICHAIN$(\mathcal{A}, \mathcal{F})$ returns true if, and only if, Player 0 has a winning strategy in $\mathbb{G} = (\mathcal{A}, \mathcal{F})$.

To show the algorithm returns in time $O(|\mathcal{F}||V(\mathcal{A})|^2|E(\mathcal{A})|)$, we require the following result from [8]:

Lemma 4.5 ([8]). *Let $\mathbb{G} = (\mathcal{A}, \mathcal{F})$ be an explicitly presented Muller game with $\mathcal{F} = \{X\}$. The set $\{v \in V(\mathcal{A}) : \text{Player 0 has a winning strategy from } v \text{ in } \mathbb{G}\}$ can be computed in time $O(|V(\mathcal{A})||E(\mathcal{A})|)$.*

It follows that at each stage of the recursion, it takes $O(|\mathcal{F}||V(\mathcal{A})||E(\mathcal{A})|)$ time to compute N. Furthermore, since $|N| \geq 1$ whenever ANTICHAIN$(\mathcal{A}, \mathcal{F})$ is recursively called, it follows that the algorithm has recursion depth at most $|V(\mathcal{A})|$. Thus the algorithm runs in time $O(|\mathcal{F}||V(\mathcal{A})|^2|E(\mathcal{A})|)$ as required.

□

4.1.1 Bounded tree-width arenas

Tree-width is an important graph parameter which measures how closely a graph resembles a tree. It has proved useful in the design of algorithms as many problems that are intractable on general graphs are known to have polynomial time solutions when restricted to graphs of bounded tree-width. In the context of Muller games, Obdržálek [14] exhibited a polynomial-time algorithm for deciding the winner in parity games on arenas of bounded tree-width. We show that this is not the case for games presented by a Muller condition type (and neither, therefore, for Zielonka DAG games, Emerson-Lei games, and circuit games). The proof of Theorem 4.1 can be modified so that the arenas constructed all have tree-width two provided we allow ourselves to specify the winning condition as a Muller condition rather than a win-set.

Theorem 4.6. *Deciding Muller games specified by a Muller condition on arenas of tree-width 2 is* Pspace*-complete.*

Proof. Membership of Pspace follows from the fact that deciding general Muller games specified by a Muller condition is in Pspace.

The construction to show Pspace-hardness is similar to that of Theorem 4.1. The reduction is also from QSAT, and the proof that it is in fact a reduction is similar. Given a QBF $\Phi = Q_{k-1}x_{k-1}\ldots\forall x_1\exists x_0\varphi$ where φ is in DNF with three literals per term, the Muller game we construct is:

- $V_1(\mathcal{A}) = T$ where T is the set of terms.
- $V_0(\mathcal{A}) = \{\varphi\} \cup \Big(T \times \{1,2,3\} \times \{x, \neg x : x \text{ is a variable}\}\Big)$.
- We have the following edges in $E(\mathcal{A})$ for all $t \in T$:
  - (φ, t),
  - $\Big(t, (t,n,l)\Big)$ if l is the n-th literal in t,
  - $\Big((t,n,x_i), (t,n,x_{i-1})\Big)$ if the n-th literal of t is x_i $(i > 0)$
  - $\Big((t,n,x_0), \varphi\Big)$ if the n-th literal of t is x_0
  - $\Big((t,n,x_i), (t,n,\neg x_i)\Big)$ for all i less than the index of the n-th literal of t
  - $\Big((t,n,\neg x_i), (t,n,x_{i-1})\Big)$ for all i less than or equal to the index of the n-th literal of t
  - $\Big((t,n,\neg x_0), \varphi\Big)$ for all n.
- $C = \{\varphi\} \cup \{x, \neg x : x \text{ is a variable}\}$ is the set of colours,
- $\chi : V(\mathcal{A}) \to C$ defined as:
  - $\chi(\varphi) = \chi(t) = \varphi$ for all $t \in T$
  - $\chi\Big((t,n,l)\Big) = l$.
- $\mathcal{C} = \Big\{S_i, S_i \cup \{x_i\}, S_i \cup \{\neg x_i\} : 0 \le i < k,\ i \text{ even}\Big\}$ where $S_0 = \{\varphi\}$ and for $i > 0$, $S_i = \{\varphi\} \cup \{x_j, \neg x_j : 0 \le j < i\}$.

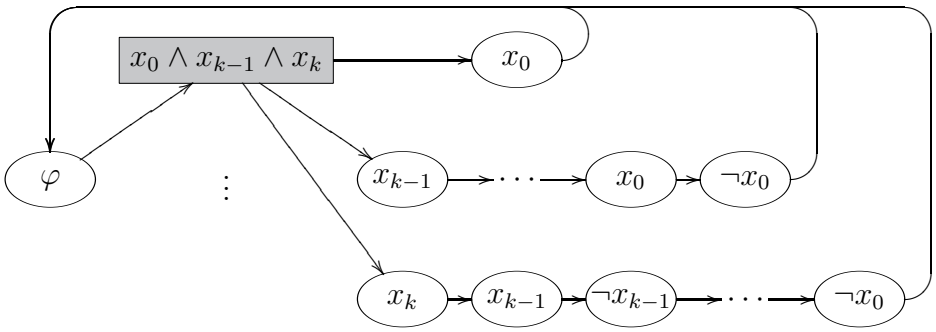

Fig. 3. Arena with bounded tree-width

Figure 3 illustrates how this arena differs from that of Theorem 4.1.

The resulting arena has tree-width 2, and the proof that Player 0 has a winning strategy if, and only if, Φ is true is similar to that of Theorem 4.1. □

4.2 Complexity of union-closed games

We now turn our attention to Muller games where the winning condition $\mathcal{F}$ is a union-closed set. Among games studied in the literature, Streett games and parity games are examples of condition types that can only specify union-closed games. Union-closed games were also studied as a class in [8]. One consideration that makes them an interesting case to study is that they admit memoryless strategies for Player 1 [9]. That is, on a game with a union-closed winning condition, if Player 1 has a winning strategy then he has a strategy which is a function only of the current position. One consequence of this fact is that, for explicitly presented union-closed games, the problem of deciding whether Player 0 wins such a game is in co-NP. This is because once a memoryless strategy for Player 1 is fixed, the problem of deciding whether Player 0 wins against that fixed strategy is in Ptime. Indeed, it is a version of alternating reachability. Thus, to decide whether Player 1 has a winning strategy we can nondeterministically guess such a strategy and then verify that Player 0 cannot defeat it. Hence, determining whether Player 1 wins is in NP and therefore deciding whether Player 0 wins is in co-NP. In this section, we aim to establish a corresponding lower bound for two condition types that can only represent union-closed games, namely the basis and superset condition types.

We saw with Theorem 3.21 that we cannot use the known complexity bounds on Streett games to easily establish similar bounds for basis games. Nevertheless, deciding basis games is still in co-NP.

Proposition 4.7. *Deciding basis games is in* co-NP.

Proof. From the comments above, it suffices to show that if we fix a memoryless strategy for Player 1 then we can decide the resulting single player basis game in polynomial time.

The algorithm is as follows. Let $\mathcal{B}$ be the basis for the winning condition. Initially let $\mathcal{B}_0 = \mathcal{B}$, and repeat the following:

(1) Let $X_i = \bigcup_{B \in \mathcal{B}_i} B$.
(2) Partition X_i into strongly connected components (SCCs).
(3) Remove any element of $\mathcal{B}_i$ which is not wholly contained in a SCC to obtain $\mathcal{B}_{i+1}$,

until $\mathcal{B}_i = \mathcal{B}_{i-1}$, at which point, let $X = X_i$. This takes at most $O\Big(|\mathcal{B}|(|V(\mathcal{A})|+|E(\mathcal{A})|)\Big)$ time using a standard SCC-partitioning algorithm. At this point, every SCC of X is a union of basis elements – all x in X are members of basis elements, and any basis element not contained in any SCC of X is removed at step 3. Furthermore, any strongly connected set of $V(\mathcal{A})$ which is a union of basis elements is a subset (of an SCC) of X, because the algorithm preserves such sets. Thus, Player 0 can win from any node from which she can reach X (play to X and then visit every node within an SCC of X forever); and Player 0 cannot win if she cannot reach X (there is no union of basis elements for which Player 0 can visit every vertex infinitely often). Thus the set of nodes from which Player 0 wins can be computed in $O\Big(|\mathcal{B}|(|V(\mathcal{A})| + |E(\mathcal{A})|) + |E(\mathcal{A})|\Big)$ time. □

We now obtain the lower bounds we seek on superset games.

Theorem 4.8. *Deciding superset games is* co-NP*-complete.*

Proof. Membership of co-NP follows from Propositions 3.22 and 4.7. To show co-NP-hardness, we use a reduction from validity of DNF formulas.

Given a formula $\varphi(x_0, x_1, \ldots, x_{k-1})$ in DNF, consider the superset game defined as follows:

- for every variable x_i we include three vertices, $x_i, \neg x_i \in V_0(\mathcal{A})$ and $x'_i \in V_1(\mathcal{A})$;
- for each i we have the edges $(x'_i, x_i), (x'_i, \neg x_i), (x_i, x'_{i+1}), (\neg x_i, x'_{i+1})$, where addition is taken modulo k;
- $v_I(\mathcal{A}) = x_0$; and
- the winning condition is specified by the set

$$\mathcal{M} = \Big\{\{l_i \in V_0(\mathcal{A}) : l_i \text{ is a literal of } t\} \text{ for every term } t \text{ of } \varphi\Big\},$$

As the superset condition is closed under union, if Player 1 has a winning strategy he has a memoryless winning strategy. Note that any memoryless strategy for Player 1 effectively chooses a truth value for each variable. The set of vertices visited infinitely often is a superset of an element of $\mathcal{M}$ if, and only if, the truth assignment chosen by Player 1 makes one term of φ (and hence all of φ) true. Thus Player 0 wins this game if, and only if, there is no truth assignment which makes φ false. □

Corollary 4.9. *Deciding basis games is* co-NP*-complete.*

We note in conclusion that the exact complexity of deciding union-closed games when they are explicitly presented remains an open problem. It is clearly in co-NP but the above arguments do not establish lower bounds for it.

5 Infinite tree automata

One of the original motivations for studying Muller and related games was to establish decidability results for problems such as non-emptiness and model checking for infinite tree automata [11]. A reduction to non-emptiness of infinite tree automata is used in some of the most effective algorithms for deciding satisfiability of formulas in logics such as $S2S$, μ-calculus, CTL*, and other logics useful for reasoning about non-terminating, branching computation. Furthermore, determining if a structure satisfies a formula in any of these logics reduces to determining if a certain automaton accepts a particular tree. In this section we show that the non-emptiness and model-checking problems (for regular trees) are Pspace-complete for Muller automata. We first present the definitions of infinite trees and infinite tree automata.

Definition 5.1 (Infinite tree)**.** For $k \in \mathbb{N}$, let $[k] = \{1, 2, \ldots, k\}$. An infinite, k-ary branching tree labelled by elements of Σ is a function $t : [k]^* \to \Sigma$. *Nodes* of an infinite tree are elements of its domain, the *root* of an infinite tree is the empty string.

Definition 5.2 (Regular tree)**.** A *subtree* of tree t rooted at $u \in [k]^*$ is the tree t_u defined as $t_u(v) = t(u \cdot v)$ for all $v \in [k]^*$. A tree t is *regular* if it has finitely many distinct subtrees, or equivalently, if there are finitely many equivalence classes under the equivalence relation

$$u \sim v \iff t(u \cdot w) = t(v \cdot w) \quad \forall w \in [k]^*.$$

Note that if a tree is regular it can be represented by a finite transition system, with the equivalence classes of the above equivalence relation as states,

the equivalence class containing the root as the initial state, and k distinct transition relations.

Definition 5.3 (Infinite tree automaton)**.** An infinite (Muller) (k-ary) tree automaton is a tuple $\mathbb{A} = (Q, \Sigma, \delta, q_0, \mathcal{F})$ where

- Q is a finite set of states,
- Σ is a finite alphabet,
- $\delta \subseteq Q \times \Sigma \times Q^k$ is the transition relation,
- q_0 is the initial state, and
- $\mathcal{F} \subseteq \mathcal{P}(Q)$ is the acceptance condition.

Given an infinite, k-ary branching tree t labelled by elements of Σ, a run of $\mathbb{A}$ on t is an infinite, k-ary branching tree r labelled by elements of Q satisfying the following two conditions.

- The root of r is labelled by q_0 ($r(\varepsilon) = q_0$).
- For all $w \in [k]^*$, if $r(w) = q$, $r(w \cdot 1) = q_1$, $r(w \cdot 2) = q_2$, $\ldots$, $r(w \cdot k) = q_k$, and $t(w) = a$, then $(q, a, q_1, q_2, \ldots, q_k) \in \delta$.

We say a run r is successful if for every (infinite) path, the set I of states visited infinitely often is an element of $\mathcal{F}$. We say $\mathbb{A}$ *accepts* t if there is a successful run of $\mathbb{A}$ on t. Given an automaton $\mathbb{A}$, the *language* of $\mathbb{A}$ is the set of trees

$$\mathcal{L}(\mathbb{A}) := \{t : \mathbb{A} \text{ accepts } t\}.$$

Two important decision problems in automata theory are non-emptiness and model-checking.

Non-emptiness of Muller tree automata	
Instance:	A Muller automaton $\mathbb{A}$
Problem:	Is $\mathcal{L}(\mathbb{A}) \neq \emptyset$?

Model-checking for Muller tree automata	
Instance:	A Muller automaton $\mathbb{A}$, and a regular infinite tree t
Problem:	Is $t \in \mathcal{L}(\mathbb{A})$?

The close connection between automata and games can be established by considering the game where the moves of Player 0 consist of choosing a transition in δ to make from a current state, and the moves of Player 1 consist of choosing which branch of the tree to descend. With this translation in mind, the

non-emptiness problem reduces to the problem of finding the winner in the win-set game $\Big(\mathcal{A}, (W, \mathcal{W})\Big)$ with

- $V_0(\mathcal{A}) = W = Q$,
- $V_1(\mathcal{A}) = Q^k$,
- $\mathcal{W} = \mathcal{F}$,
- edges from $V_0(\mathcal{A})$ to $V_1(\mathcal{A})$ determined by δ: an edge from q to $(q_1, q_2, \ldots, q_k)$ if there is $a \in \Sigma$ such that $(q, a, q_1, \ldots q_k) \in \delta$, and
- edges from $V_1(\mathcal{A})$ to $V_0(\mathcal{A})$ being projections: an edge from $(q_1, \ldots, q_k)$ to q_i for all $i \in [k]$.

Clearly if Player 0 has a winning strategy in this game, it is possible to construct a tree which the automaton accepts. Conversely, if Player 1 has a winning strategy, no such tree exists.

By adapting the proof of Theorem 4.1 we are able to show that the non-emptiness problem for Muller automata as well as the problem of determining whether a given automaton accepts a given regular tree are both PSPACE-complete.

Theorem 5.4. *The non-emptiness problem for Muller tree automata is* PSPACE*-complete.*

Proof. Membership in PSPACE is established by the above polynomial time reduction from the non-emptiness problem of Muller automata to win-set games. Here we show PSPACE hardness through a reduction from QSAT.

Given a QBF $\Phi = Q_{k-1}x_{k-1} \ldots \forall x_1 \exists x_0 \varphi$, where φ is in disjunctive normal form with 3 literals per term, we construct the following Muller automaton $\mathbb{A}_\Phi = (Q, \Sigma, q_I, \delta, \mathcal{F})$ that accepts infinite ternary trees:

- $Q = \{q_\varphi\} \cup \{q_x, q_{\neg x} : \text{for all variables } x\}$
- $\Sigma = \{a\}$ [3]
- $q_I = q_\varphi$
- $\delta \subseteq Q \times Q^3$ given by:
  - for each term $(l_0 \wedge l_1 \wedge l_2)$ of φ, $(q_\varphi, q_{l_0}, q_{l_1}, q_{l_2}) \in \delta$;
  - $(q_{x_i}, q_{x_{i-1}}, q_{x_{i-1}}, q_{x_{i-1}}), (q_{x_i}, q_{\neg x_{i-1}}, q_{\neg x_{i-1}}, q_{\neg x_{i-1}}) \in \delta$ for $0 < i < k$;
  - $(q_{\neg x_i}, q_{x_{i-1}}, q_{x_{i-1}}, q_{x_{i-1}}), (q_{\neg x_i}, q_{\neg x_{i-1}}, q_{\neg x_{i-1}}, q_{\neg x_{i-1}}) \in \delta$ for $0 < i < k$; and
  - $(q_{x_0}, q_\varphi, q_\varphi, q_\varphi), (q_{\neg x_0}, q_\varphi, q_\varphi, q_\varphi) \in \delta$.
- $\mathcal{F} = \Big\{S_i, S_i \cup \{q_{x_i}\}, S_i \cup \{q_{\neg x_i}\} : 0 \leq i < k, i \text{ even}\Big\}$ where $S_i = \{q_\varphi\} \cup \{q_{x_j}, q_{\neg x_j} : 0 \leq j < i\}$.

Now by using the reduction to win-set games outlined above, asking if $\mathbb{A}_\Phi$

[3] as Σ is a singleton, for ease of reading we omit a from the description of δ.

accepts any tree is equivalent to asking if Player 0 has a winning strategy (from q_φ) on the win-set game used in Theorem 4.1. □

The model checking problem also reduces to deciding which player wins an infinite game. However, depending on how the tree is presented, the resulting arena may be of infinite size. If the tree is presented as a finite transition system, a game with finite arena can be constructed, and we can apply Theorem 5.4 to obtain the following corollary.

Corollary 5.5. *Given a regular, infinite, k-ary branching tree t (represented as a transition system) and a Muller automaton $\mathbb{A} = (Q, \Sigma, \delta, q_I, \mathcal{F})$, asking if $\mathbb{A}$ accepts t is* PSPACE*-complete.*

Proof. PSPACE hardness follows from the proof of Theorem 5.4, as the automata constructed there accept at most one tree – the ternary branching tree with all nodes labelled by a.

To show that the problem is in PSPACE, we reduce it to the problem of deciding a Muller game with winning condition specified by a Muller condition. Let $(S, s_0, t_1, \ldots, t_k)$ denote the transition system representing the tree t. The required Muller game, $\Big(\mathcal{A}, (\chi, \mathcal{C})\Big)$, is given by the following.

- $V_0(\mathcal{A}) = Q \times S$.
- $V_1(\mathcal{A}) = Q \times S \times Q^k$.
- There is an edge from $(q, s) \in V_0(\mathcal{A})$ to $(q, s, q_1, \ldots q_k) \in V_1(\mathcal{A})$ whenever $(q, a, q_1, \ldots, q_k) \in \delta$ where a is the label of s.
- There is an edge from $(q, s, q_1, \ldots, q_k) \in V_1(\mathcal{A})$ to $(q_i, t_i(s)) \in V_0(\mathcal{A})$ for $1 \leq i \leq k$.
- $v_I(\mathcal{A}) = (q_I, s_0)$,
- Q is the set of colours,
- $\chi : V(\mathcal{A}) \to Q$ is defined by taking the first component of the vertex.
- $\mathcal{C} = \mathcal{F}$.

It is clear from the definitions that Player 0 has a winning strategy from (q_I, s_0) in this game if, and only if, $\mathbb{A}$ accepts t. □

6 Conclusion

We have considered the complexity of deciding the winner in a variety of Muller games. We establish a framework, through the notion of polynomial translatability, within which the expressive power and the succinctness of types

of winning conditions can be considered. We used this, along with an encoding of QBF in win-set conditions to establish PSPACE-completeness for five different condition types that can be used to describe Muller games and to establish the PSPACE-completeness of the non-emptiness and model-checking problems for Muller automata. We also showed CO-NP-completeness results for two different condition types describing union-closed games.

References

[1] J. R. Büchi and L. H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.

[2] S. Dziembowski, M. Jurdziński, and I. Walukiewicz. How much memory is needed to win infinite games? In *Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science*, pages 99–110, 1997.

[3] E. A. Emerson and C. S. Jutla. The complexity of tree automata and logics of programs (extended abstract). In *Proceedings for the 29th IEEE Symposium on Foundations of Computer Science*, pages 328–337, 1988.

[4] E. A. Emerson and C.-L. Lei. Modalities for model checking: Branching time strikes back. In *Proceedings of the 12th Annual ACM Symposium on Principles of Programming Languages*, pages 84–96, 1985.

[5] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

[6] P. Hunter. *Complexity and Infinite Games on Finite Graphs*. PhD thesis, University of Cambridge, 2007.

[7] P. Hunter and A. Dawar. Complexity bounds for regular games. In *Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science*, volume 3618 of *Lecture Notes in Computer Science*, pages 495–506. Springer, 2005.

[8] H. Ishihara and B. Khoussainov. Complexity of some infinite games played on finite graphs. In *Proceedings of the 28th International Workshop on Graph Theoretical Concepts in Computer Science*, volume 2573 of *Lecture Notes in Computer Science*. Springer, 2002.

[9] N. Klarlund. Progress measures, immediate determinacy, and a subset construction for tree automata. *Annals of Pure and Applied Logic*, 69(2-3):243–268, 1994.

[10] S. La Torre, A. Murano, and M. Napoli. Weak Muller acceptance conditions for tree automata. In *Proceedings of the 3rd International Workshop on Verification, Model Checking and Abstract Interpretation*, volume 2294 of *Lecture Notes in Computer Science*, pages 240–254. Springer, 2002.

[11] R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9(5):521–530, 1966.

[12] R. McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993.

[13] A. Nerode, J. B. Remmel, and A. Yakhnis. McNaughton games and extracting strategies for concurrent programs. *Annals of Pure and Applied Logic*, 78(1-3):203–242, 1996.

[14] J. Obdržálek. Fast mu-calculus model checking when tree-width is bounded. In *Proceedings of 15th International Conference on Computer Aided Verification*, volume 2725 of *Lecture Notes in Computer Science*, pages 80–92. Springer, 2003.

[15] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1995.

[16] R. S. Streett. Propositional dynamic logic of looping and converse is elementarily decidable. *Information and Control*, 54(1-2):121–141, 1982.

[17] W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200:135–183, 1998.