# Complexity Bounds for Regular Games

Paul Hunter and Anuj Dawar

University of Cambridge Computer Laboratory, Cambridge CB3 0FD, UK.
paul.hunter@cl.cam.ac.uk, anuj.dawar@cl.cam.ac.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. 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 *regular* 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 Muller, 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 regular 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 regular game is defined by a graph $(V, E)$, where $E \subseteq V \times V$, and a winning condition $\mathcal{F} \subseteq \mathscr{P}(V)$ 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 $\mathcal{F} \subseteq \mathscr{P}(V)$ can be expressed in it or there is only a restricted class of winning conditions that can be expressed. Muller games are an example of the first case 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 regular 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 $\Omega$ is bounded in length by a polynomial (indeed, linear) function of $|V|$. It is also clear that not every set $\mathcal{F} \subseteq \mathscr{P}(V)$ 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 regular games. It is known to be in NP $\cap$ co-NP [3] and conjectured by some to be in P. 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 regular games. No lower bounds are previously known for formalisms that are expressive enough to specify all regular games, though algorithms for such games have been studied which establish, for instance, that the games are decidable in Pspace.

We consider five 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 two further general-purpose representations: Emerson-Lei games, where the winning condition is presented as a Boolean formula over the vertices of the graph; and the case where the winning condition is represented as a Zielonka DAG. The latter is a data structure (defined in Section 2) based on the Zielonka trees of [12]. 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 Muller games are translatable to Zielonka DAGs which are in turn translatable to Emerson-Lei 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 [6]). 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 [7]. 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 Background and Definitions

In this section, we present the basic definitions of games as well as particular winning conditions. Many of the definitions presented here are standard. Where this is the case, we follow terminology and notation from [5].

An *arena* $\mathcal{A}$ is a directed graph on a set of vertices $V$ which is partitioned into two sets $V_0$ and $V_1$, i.e. $\mathcal{A} = (V, E)$ where $V = V_0 \cup V_1$, $E \subseteq V \times V$ and $V_0$ and $V_1$ are disjoint. For the results we establish in this paper, there is no loss of generality in assuming that the graph is bipartite in the sense that $E \subseteq (V_0 \times V_1) \cup (V_1 \times V_0)$ and that for each $v \in V$, there is a $v' \in V$ such that $(v, v') \in E$. For instance, there is an easy transformation that maps any game to a bipartite game by inserting a new $V_0$ (resp. $V_1$) vertex in every edge that connects two $V_1$ (resp. $V_0$) vertices. This transformation does not change the existence of winning strategies for either player from any of the original vertices. Thus, wherever it is convenient, we will assume that the arena satisfies the above assumptions.

A *game* $G$ is an arena $\mathcal{A}$ together with a winning set of sequences $\text{Win} \subseteq V^\omega$. Informally, we think of the game as played between two players, Player 0 and Player 1, with a token that sits on a vertex $v$ in $V$. If $v \in V_0$, it is Player 0's turn to move and she[1] moves it to some $v'$ such that there is an edge $(v, v')$ in $E$ and similarly for Player 1 when the token is on a vertex in $V_1$. The infinite sequence of such moves determines a *play* $\pi$ which is the sequence in $V^\omega$ of vertices visited. We say Player 0 wins the play if $\pi \in \text{Win}$ and Player 1 wins otherwise.

A *strategy* (for Player $i$) is a function $f$ from $V^* V_i$ to $V$ with $f(v_0 v_1 \ldots v_n) \in v_n E$. Given a sequence of vertices visited, ending with a vertex in $V_i$, a strategy gives the vertex that Player $i$ should then play to. A play is *consistent* with a strategy if every move made by Player $i$ is determined by the strategy, and a strategy is *winning* if every play consistent with it is winning for Player $i$. If a strategy $f$ has the property that for some fixed $m$, $f(w) = f(w')$ 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*.

A game $(V, E, \text{Win})$ is *regular* if there is a set $\mathcal{F} \subseteq \mathscr{P}(V)$ such that for any $\pi \in V^\omega$, $\pi \in \text{Win}$ if, and only if, the set $\{v : v \text{ occurs infinitely often in } \pi\}$ is in $\mathcal{F}$. In the remainder of the paper, we are concerned with games that are finite (i.e. $V$ is a finite set) and regular. Regular games are known to be determined, that is, for each game and each initial vertex $v$, either Player 0 or Player 1 has a winning strategy.

We say that a regular game $(V, E, \mathcal{F})$ is *union-closed* if whenever $I, J \in \mathcal{F}$, then $I \cup J \in \mathcal{F}$. A regular game is *upward-closed* if for any $I \in \mathcal{F}$ and $I \subseteq J$, we have $J \in \mathcal{F}$. Clearly any upward-closed game is also union-closed.

The games used in the literature in the study of logics and automata are generally regular games (though not necessarily finite). 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 do not give a formal definition of a *condition type* but we will define specific instances of such types.

The most straightforward presentation of a regular game $(V, E, \mathcal{F})$ is given by listing all elements of $\mathcal{F}$. We call this an *explicit condition*. Games specified by such a condition type are sometimes called Muller games in the literature, but we reserve that term for the more commonly used presentation in terms of colours given next.

A *Muller condition* on an arena $(V, E)$ is given by a set of colours $C$, a colouring function $\chi : V \to C$ and a set $\mathcal{C} \subseteq \mathscr{P}(C)$. The set $\mathcal{F}$ specified by such a condition is the set $\{I \subseteq V : \chi(I) \in \mathcal{C}\}$.

An *Emerson-Lei condition* [4] on an arena $(V, E)$ is given by a Boolean formula $\varphi$ with variables from the set $V$. The set $\mathcal{F}$ specified is the collection of

[1] For ease of reference we use the feminine pronoun for Player 0 and the masculine for Player 1.

sets $I \subseteq V$ such that the truth assignment that maps each element of $I$ to true and each element of $V \setminus I$ to false satisfies $\varphi$.

In [12], Zielonka introduced a representation of a winning set $\mathcal{F} \subseteq \mathscr{P}(V)$ in terms of a labelled tree, where the labels on the nodes are subsets of $V$. The *Zielonka 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 $\overline{\mathcal{F}} = \mathscr{P}(V) \setminus \mathcal{F}$.
2. If $V \in \mathcal{F}$ then the root of $\mathcal{Z}_{\mathcal{F},V}$ is labelled with $V$. Let $M_0, M_1, \ldots, M_{k-1}$ be the maximal sets in $\overline{\mathcal{F}}$, and let $\mathcal{F}|_{M_i} = \mathcal{F} \cap \mathscr{P}(M_i)$. The children of the root are the subtrees $\mathcal{Z}_{\mathcal{F}|_{M_i},M_i}$, for $0 \leq i \leq k-1$.

A *Zielonka DAG* is constructed as a Zielonka tree except nodes labelled by the same set are identified, making it a directed acyclic graph. A Zielonka tree (DAG) condition is one which uses a Zielonka tree (respectively, DAG) presentation. 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*. In the sequel, we use terms such as *children* and *leaves* when referring to DAGs as well as trees, where the meaning is clear.

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*, which is given by $W \subseteq V$ and $\mathcal{W} \subseteq \mathscr{P}(W)$. The sets described by this condition are $\{I \subseteq V : I \cap W \in \mathcal{W}\}$. Win-set games are the type of games considered by McNaughton in [9] where he presents an algorithm to decide the winner of such games.

The five condition types defined above are general purpose in that any regular game can be specified by any one of the condition types. We now look at some less general types of conditions.

A *basis condition* on an arena $(V, E)$ is given by a set $\mathcal{B} \subseteq \mathscr{P}(V)$. This specifies the collection $\mathcal{F}$ of sets $I \subseteq V$ such that there are $B_1, \ldots, B_n \in \mathcal{B}$ with $I = \bigcup_{1 \leq i \leq n} B_i$. It is clear that a regular game can be specified using a Basis condition if, and only if, it is union-closed.

A *superset condition* (also called a superset Muller condition in [7]) is given by a set $\mathcal{M} \subseteq \mathscr{P}(V)$ which specifies the set $\mathcal{F} = \{I \subseteq V : J \subseteq I \text{ for some } J \in \mathcal{M}\}$. Only upward-closed games can be specified in this way.

Our main concern is with the complexity of the following decision problem for a fixed condition type: given a game consisting of a finite arena, a condition of the given type and an initial vertex, which of the two players has a winning strategy? We often refer to this as the problem of deciding the winner of a game. This problem has been investigated for condition types other than the ones considered here. For example, in [3] it was shown that deciding games with a winning condition expressed in Rabin form is NP-complete; and the complexity of deciding Parity games is a question that has been the focus of intensive research. However, lower bound proofs for any games are hard to come by.

## 3 Translations

We begin by considering the five ways we have defined of specifying a winning condition that are *general purpose*, i.e. expressive enough to describe any regular game. These are the explicit presentation, the win-set condition, the Muller condition, the Zielonka DAG and the Emerson-Lei condition. We show that this list is strictly increasing in order of succinctness. That is, any game specified using a condition of one of these types can also be specified using a type later in the list with only a polynomial increase in size. However, for each type, there are specifications of games for which any description of a type earlier in the list is necessarily exponentially longer. We formalise the notion of succinctness through the following definition. Note that this definition is somewhat informal as we have not given a formal definition of a "condition type". It suffices for our present purposes if we take $A$ and $B$ in this definition to range over the types defined in the previous section.

**Definition 3.1.** *Given two condition types $A$ and $B$, we say that $A$ is* polynomially translatable *to $B$ if there is an algorithm, running in polynomial time which, given a game with condition of type $A$ produces a condition of type $B$ which describes the same game.*

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

If condition type $A$ is not translatable to $B$ this may be for one of three reasons. Either $A$ is more expressive than $B$ in that there are sets $\mathcal{F}$ that can be expressed using $A$ but not $B$; or there are some sets for which the representation of type $A$ 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 that $A$ is *more succinct* than $B$ if $B$ is translatable to $A$ but $A$ is not translatable to $B$.

It is straightforward to show that win-set conditions are more succinct than explicit presentations. To translate an explicitly presented game $(V, E, \mathcal{F})$ to a win-set condition, simply take $W = V$ 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 $\mathcal{F}$ described consists of all subsets of $V$ and an explicit presentation must be exponential in length.

We now show, through the next three theorems, that Emerson-Lei games are more succinct than Zielonka DAG games, which are in turn more succinct than Muller games, which are more succinct than win-set games. In Section 5 we also show that basis and superset games are more succinct than explicit presentations of union-closed and upward-closed games respectively.

**Theorem 3.2.** *The Muller condition type is more succinct than the win-set condition type.*

*Proof.* Given a win-set game $(V, E, W, \mathcal{W})$, 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 $\chi : V \to C$ 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 $\{X, X \cup \{c\} : X \in \mathcal{W}\}$. 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 $(V, E)$, where half of $V$, $V_r$, is coloured red, the other half coloured blue, and the family of sets of colours is $\mathcal{C} = \{\{\text{red}\}\}$. The family $\mathcal{F}$ described by this condition consists of the $2^{|V|/2} - 1$ non-empty subsets of $V_r$. Now consider trying to describe this family using a win-set condition. In general, if $\mathcal{G}$ is the family of subsets of $V$ described by the win-set condition $(W, \mathcal{W})$, then for any $v \notin W$ and $X \subseteq V$ we have $\{v\} \cup X \in \mathcal{G} \Leftrightarrow X \in \mathcal{G}$. Observe that in our game no vertex has this latter 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 must be $V$, and $\mathcal{W}$ is the explicit listing of the $2^{|V|/2} - 1$ subsets of $V_r$. Thus $(W, \mathcal{W})$ cannot be produced in polynomial time.

**Theorem 3.3.** *The Zielonka DAG condition type is more succinct than the Muller condition type.*

*Proof.* Given a Muller game consisting of an arena $(V, E)$, 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 $V$ 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 children of $t$. That is, for each 1-level child $u$ 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 children of that node. Therefore, $\chi(I) \in \mathcal{C}$. Similarly, by changing 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 $V$ which consist of 2 or more elements. The Zielonka DAG which describes this family consists of $|V| + 1$ nodes – one 0-level node labelled by $V$, and $|V|$ 1-level nodes labelled by the singleton subsets of $V$. 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, $|\mathcal{C}| = |\mathcal{F}| = 2^{|V|} - |V| - 1$. It follows that the translation from Zielonka DAGs to Muller conditions cannot be done in polynomial time.

**Theorem 3.4.** *The Emerson-Lei condition type is more succinct than the Zielonka DAG condition type.*

*Proof.* Given a Zielonka DAG game $(V, E, \mathcal{Z}_{\mathcal{F},V})$, we define, for each node $t$ in $\mathcal{Z}_{\mathcal{F},V}$ a boolean formula $\varphi_t$. This formula is defined by induction on the height of $t$. Suppose the label of $t$ is $X$. We have the following cases:

1. $t$ is a 0-level ($X \in \mathcal{F}$) leaf: In this case, let $\varphi_t = \bigwedge_{x \notin X} \neg x$.
2. $t$ is a 1-level ($X \notin \mathcal{F}$) leaf: In this case, let $\varphi_t = \bigvee_{x \notin X} x$.
3. $t$ is a 0-level node with $k$ children $t_0, t_1, \ldots, t_{k-1}$: In this case, let $\varphi_t = \bigwedge_{x \notin X} \neg x \wedge \bigwedge_{i=0}^{k-1} \varphi_{t_i}$.
4. $t$ is a 1-level node with $k$ children $t_0, t_1, \ldots, t_{k-1}$: In this case, let $\varphi_t = \bigvee_{x \notin X} x \vee \bigvee_{i=0}^{k-1} \varphi_{t_i}$.

We claim that the condition $\mathcal{F}$ is specified by the formula $\varphi_r$ where $r$ is the root of $\mathcal{Z}_{\mathcal{F},V}$. This formula has size at most $|V||\mathcal{Z}_{\mathcal{F},V}|$ 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 $\varphi_t$ defines the restriction of $\mathcal{F}$ to $X$. We consider the following cases:

1. $t$ is a 0-level leaf. In this case any subset of $X$ is in $\mathcal{F}$. $I \subseteq V$ satisfies $\varphi_t$ if, and only if, no variable that is not in $X$ appears in $I$, i.e. $I \subseteq X$.
2. $t$ is a 1-level leaf. In this case any subset of $X$ is not in $\mathcal{F}$. Here $I \subseteq V$ satisfies $\varphi_t$ if, and only if, there is some element in $I$ which is not in $X$, i.e. $I \nsubseteq X$.
3. $t$ is a 0-level node with $k$ children labelled by $X_0, X_1, \ldots, X_{k-1}$. 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 $I \subseteq V$ satisfies $\varphi_t$ if, and only if, $I$ is a subset of $X$ and $I$ satisfies $\varphi_{t_i}$ for all children.
4. $t$ is a 1-level node with $k$ children labelled by $X_0, X_1, \ldots, X_{k-1}$. 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 $\varphi_t$ if, and only if, either $I$ is not contained in $X$, or there is some child $t_i$ such that $I$ satisfies $\varphi_{t_i}$.

To show there is no translation in the other direction, let $V = \{x_0, x_1, \ldots, x_{2k-1}\}$, and consider the family of sets $\mathcal{F}$ described by the formula

$$\bigvee_{0 \leq i < k} (x_{2i} \wedge x_{2i+1}).$$

Clearly $|\varphi| = O(|V|)$. 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 $2^{|V|/2}$ subsets containing exactly one of $\{x_{2i}, x_{2i+1}\}$ for $0 \leq i < k$. Thus $\mathcal{Z}_{\mathcal{F},V}$ must have at least this number of nodes, and is therefore not constructible in polynomial time.

## 4 PSPACE-completeness

In this section we establish the complexity of deciding the winner for the four main condition types considered in the previous section. McNaughton [9], and later Nerode, Remmel and Yakhnis [10] describe an algorithm for deciding win-set games. An analysis of this algorithm shows it requires space $O(|V|^2)$. Moreover, the algorithm is easily adapted to the case where the winning condition is presented explicitly, or as a Muller condition, a Zielonka DAG or an Emerson-Lei 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.*

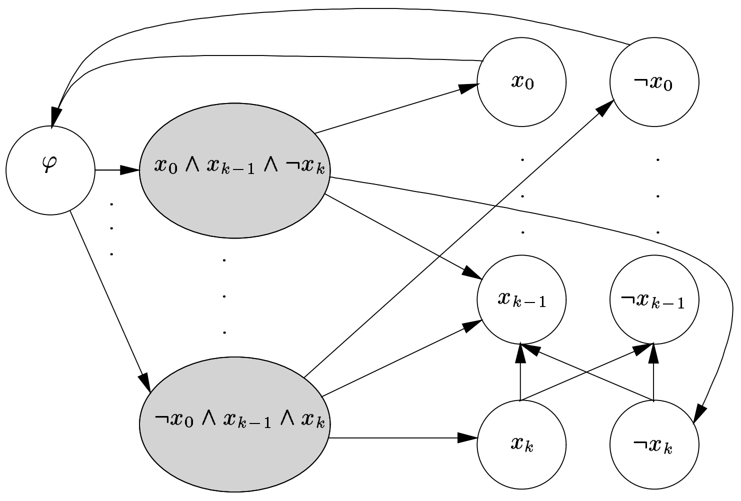
*Proof.* By the above comments, we only need to show PSPACE-hardness. For this, we reduce the problem of QSAT (satisfiability of a quantified boolean formula [QBF]) to the problem of deciding the winner of a win-set game.

We assume, without loss of generality that we are given a QBF, $\Phi = Q_{k-1}x_{k-1} \ldots \forall x_1 \exists x_0 \varphi$ in which quantifiers are strictly alternating and $\varphi$ is in disjunctive normal form with 3 literals per clause. We then define a win-set game $\mathcal{G}_\Phi$ as follows:

- $V_0 = \{\varphi\} \cup \{x, \neg x \ : \ \text{for all variables } x\}$
- $V_1 = \{C_0, \ldots, C_{m-1}\}$, the set of clauses in $\varphi$.
- $E$ given by:
  - $(\varphi, C_j) \in E$ for $0 \leq j < m$;
  - If $C_j = (l_0 \wedge l_1 \wedge l_2)$, then $(C_j, l_0), (C_j, l_1), (C_j, l_2) \in E$;
  - $(x_i, x_{i-1}), (x_i, \neg x_{i-1}) \in E$ for $0 < i < k$;
  - $(\neg x_i, x_{i-1}), (\neg x_i, \neg x_{i-1}) \in E$ for $0 < i < k$; and
  - $(x_0, \varphi), (\neg x_0, \varphi) \in E$.
- $W = V_0 \setminus \{\varphi\}$, 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 $S_i = \{x_j, \neg x_j \ : \ 0 \leq j < i\}$.



$\varphi$
$x_0 \wedge x_{k-1} \wedge \neg x_k$
$\neg x_0 \wedge x_{k-1} \wedge x_k$
$x_0$
$\neg x_0$
$x_{k-1}$
$\neg x_{k-1}$
$x_k$
$\neg x_k$

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

Figure 1 illustrates how the arena of $\mathcal{G}_\Phi$ would look if $\varphi$ contained the clauses $(x_0 \wedge x_{k-1} \wedge \neg x_k)$ 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 Player 0's strategy is to perpetually verify $\varphi$. 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, $\Phi$ is true. To formally show that Player 0 has a winning strategy if, and only if, $\Phi$ is true, we proceed by induction on $k$, the number of quantifiers of $\Phi$.

**Base case:** $k = 1$ By the idempotence of $\wedge$ and $\vee$ and assuming $\Phi$ is closed, $\Phi$ 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, $\{\varphi, C_0, x_0, \neg x_0\}$. Player 0 wins by always returning to $\varphi$ from whichever of $x_0$ and $\neg x_0$ Player 1 is forced to play to, and $\Phi$ is clearly true.
- $\Phi = \exists x_0 . (x_0 \vee \neg x_0)$. Here $\Phi$ is also true. The arena consists of five vertices $\{\varphi, C_0, C_1, x_0, \neg x_0\}$ and Player 0 has the only choice (at $\varphi$ and $x_0$). A winning strategy is to always play from $\varphi$ to $C_0$, and to return immediately to $\varphi$ from $x_0$.
- $\Phi = \exists x_0 . (x_0 \wedge \neg x_0)$. Here $\Phi$ is false. The arena consists of four vertices $\{\varphi, C_0, x_0, \neg x_0\}$ and Player 1 can force the play to visit both $x_0$ and $\neg x_0$ infinitely often by alternately choosing each from $C_0$. Note that this strategy requires memory to remember which vertex was visited last time.

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

**Inductive case:** The inductive hypothesis asserts that if $\Phi$ has $k-1$ quantifiers and is closed, then Player 0 has a winning strategy if, and only if, $\Phi$ 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 $\varphi$ and $\mathsf{v}$ is either $\mathsf{true}$ or $\mathsf{false}$, we write $\varphi[x \mapsto \mathsf{v}]$ to denote the formula obtained by substituting $\mathsf{v}$ for $x$ in $\varphi$ and simplifying. Note that if $\varphi[x \mapsto \mathsf{true}]$ simplifies to $\mathsf{true}$ then $\varphi$ must have at least one clause containing the single literal $x$, and if it simplifies to $\mathsf{false}$, then all clauses contain $\neg x$. For a set $U$ of vertices of an arena, we write $\mathrm{Avoid}_i(U, v)$ to denote the subset of $U$ from which Player $i$ has a strategy to avoid the vertex $v$ without leaving $U$. This can be constructed in the following way:

1. Start with $k = 0$ and $R_0 = \{v\}$
2. For every $v \in V_i \cap U$ with all out-going edges ending in $R_k \cup (V \setminus U)$, add $v$ to $R_{k+1}$
3. For every $v \in V_{1-i} \cap U$ with an out-going edge ending in $R_k \cup (V \setminus U)$, add $v$ to $R_{k+1}$
4. Increase $k$ by 1 and repeat steps 2 and 3 until $R_k = R_{k+1}$
5. $\mathrm{Avoid}_i(U, v)$ is then defined as $V \setminus R_k$.

The crucial lemma can now be stated as

**Lemma 4.2.** *If $\Phi = Qx\varphi$ ($Q \in \{\exists, \forall\}$) and $\varphi[x \mapsto \mathsf{true}]$ does not simplify to* $\mathsf{true}$ *or* $\mathsf{false}$*, then $\mathcal{G}_{\varphi[x\mapsto\mathsf{true}]}$ is isomorphic to the subgame of $\mathcal{G}_\Phi = (V, E, \Omega)$ induced by the set $\mathit{Avoid}_1(\mathit{Avoid}_0(V, \neg x), x)$. Dually, if $\varphi[x \mapsto \mathsf{false}]$ does not simplify to* $\mathsf{true}$ *or* $\mathsf{false}$*, then $\mathcal{G}_{\varphi[x\mapsto\mathsf{false}]}$ is isomorphic to the subgame of $\mathcal{G}_\Phi$ induced by the set $\mathit{Avoid}_1(\mathit{Avoid}_0(V, x), \neg x)$.*

*Proof.* $\varphi[x \mapsto \mathsf{true}]$ consists of the clauses of $\varphi$ that do not contain $\neg x$, with all occurrences of $x$ removed. The assumption that $\varphi[x \mapsto \mathsf{true}]$ does not simplify to $\mathsf{true}$ or $\mathsf{false}$ implies that there is at least one such clause. The arena for the game $\mathcal{G}_{\varphi[x\mapsto\mathsf{true}]}$ thus consists of vertices for $\varphi[x \mapsto \mathsf{true}]$, the clauses, and the variables (and their negations) of $\varphi$, excluding $x$ and $\neg x$. The edges are the same as those for $\mathcal{G}_\Phi$ restricted to this vertex set. We show that the subarena of $\mathcal{G}_\Phi$ induced by $\mathrm{Avoid}_1(\mathrm{Avoid}_0(V, \neg x), x)$ is identical. As the winning condition only depends on vertices corresponding to variables, it follows that the winning conditions are also identical.

In $\mathcal{G}_\Phi = (V, E, \Omega)$, the set $\mathrm{Avoid}_0(V, \neg x)$ consists of the vertices from which Player 0 can avoid $\neg x$. As Player 1 chooses the play from vertices corresponding to clauses, the set of vertices from which Player 1 can reach $\neg x$ is $\{\neg x\} \cup \{C : \neg x \in C\}$. As there is at least one clause that does not contain $\neg x$, Player 0 can play to that clause to avoid $\neg x$ from $\varphi$. The only other vertex from which it is possible to reach $\neg x$ is $x$ (as $x$ is the outermost variable in $\Phi$), and from there Player 0 can play to either $y$ (for the next outermost variable $y$) or $\varphi$ (if no such variable exists). Thus

$$\mathrm{Avoid}_0(V, \neg x) = V \setminus \{\neg x\} \cup \{C : \neg x \in C\}.$$

Next we consider $\mathrm{Avoid}_1(V', x)$ for $V' = \mathrm{Avoid}_0(V, \neg x)$. As $\varphi$ does not contain a clause containing $x$ by itself, Player 0 cannot force the play to $x$ from $\varphi$, as Player 1 can always choose to play to another literal. Furthermore, as $x$ is the outermost variable in $\Phi$, the only edges to $x$ are from vertices associated with clauses. Thus $x$ is the only vertex from which Player 0 can force the play to visit $x$, so

$$\mathrm{Avoid}_1(V', x) = V' \setminus \{x\}.$$

Thus $\mathrm{Avoid}_1(\mathrm{Avoid}_0(V, \neg x), x) = V \setminus \big(\{x, \neg x\} \cup \{C : \neg x \in C\}\big)$, which is precisely the vertex set of $\mathcal{G}_{\varphi[x \mapsto \mathsf{true}]}$. The edges for both arenas are those of $\mathcal{G}_\Phi$ restricted to these vertices, as are the winning conditions. Thus the two games are identical.

Note that due to the structure of the games, the Avoid operators commute.

To complete the inductive step, we consider two cases.

- $\Phi = \exists x_{k-1}.\varphi$. If $\Phi$ is true, then there is a truth value $\mathsf{v}$ such that $\varphi[x_{k-1} \mapsto \mathsf{v}]$ is true. Assume that $\mathsf{v} = \mathsf{true}$, the case for $\mathsf{v} = \mathsf{false}$ being similar. Player 0's winning strategy 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 $\varphi[x_{k-1} \mapsto \mathsf{v}]$ simplifies to $\mathsf{true}$, then Player 0 can force the play to visit $x_{k-1}$, by playing to the clause that only contains $x_{k-1}$. Otherwise Player 1 can play to avoid $x_{k-1}$, restricting the play to $\mathrm{Avoid}_1(\mathrm{Avoid}_0(V, \neg x_{k-1}), x_{k-1})$. From the above lemma, this subgame is equivalent to $\mathcal{G}_{\varphi[x_{k-1} \mapsto \mathsf{true}]}$, and from the inductive hypothesis, Player 0 has a winning strategy on this game. Thus Player 0's strategy is to play her winning strategy on the smaller game. If $\Phi$ is false, then Player 1 plays a strategy similar to Player 0's strategy in the case below.
- $\Phi = \forall x_{k-1}.\varphi$. In this case, if $\Phi$ is true, then for both choices of truth value $\mathsf{v} \in \{\mathsf{true}, \mathsf{false}\}$, $\varphi[x_{k-1} \mapsto \mathsf{v}]$ is true. Player 0's winning strategy 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 $\Phi$ is false, Player 1 plays a strategy similar to Player 0's strategy in the previous case. Note that in this case Player 0 cannot force the play to visit both $x_{k-1}$ and $\neg x_{k-1}$.

**Corollary 4.3.** *Deciding Muller games is* PSPACE*-complete.*

*Proof.* We have already indicated that the problem is in PSPACE. PSPACE-hardness follows from Theorem 3.2 and Theorem 4.1.

**Corollary 4.4.** *Deciding Zielonka DAG games is* PSPACE*-complete.*

*Proof.* From Theorem 3.3 and Theorem 4.1.

**Corollary 4.5.** *Deciding Emerson-Lei games is* PSPACE*-complete.*

*Proof.* From Theorem 3.4 and Theorem 4.1.

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 P.

*Bounded tree-width arenas.* Tree-width is a measure of 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 regular games, Obdržálek [11] 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 Muller games (and neither, therefore, for Zielonka DAG games and Emerson-Lei 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 on arenas of tree-width 2 is* PSPACE*-complete.*

*Proof.* Membership of PSPACE follows from the fact that deciding Muller games in general 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 $\varphi$ is in DNF with three literals per clause, the Muller game we construct is:

- $V_1 = D$ where $D$ is the set of clauses.
- $V_0 = \{\varphi\} \cup D \times \{1,2,3\} \times \{x, \neg x \,:\, x \text{ is a variable}\}$.
- We have the following edges for all $c \in D$:
  - $(\varphi, c)$,
  - $\big(c, (c, n, l)\big)$ if $l$ is the $n$-th literal in $c$,
  - $\big((c, n, x_i), (c, n, x_{i-1})\big)$ if the $n$-th literal of $c$ is $x_i$ $(i > 0)$
  - $\big((c, n, x_0), \varphi\big)$ if the $n$-th literal of $c$ is $x_0$
  - $\big((c, n, x_i), (c, n, \neg x_i)\big)$ for all $i$ less than the index of the $n$-th literal of $c$
  - $\big((c, n, \neg x_i), (c, n, x_{i-1})\big)$ for all $i$ less than or equal to the index of the $n$-th literal of $c$
  - $\big((c, n, \neg x_0), \varphi\big)$ for all $n$.
- $C = \{\varphi\} \cup \{x, \neg x \,:\, x \text{ is a variable}\}$ is the set of colours,
- $\chi : V \to C$ defined as:
  - $\chi(\varphi) = \chi(c) = \varphi$ for all $c \in D$
  - $\chi\big((c, n, l)\big) = l$.

– $\mathcal{C} = \{S_i, S_i \cup \{x_i\}, S_i \cup \{\neg x_i\} : 0 \leq i < k,\ i \text{ even}\}$ where $S_i = \{\varphi\} \cup \{x_j, \neg x_j : 0 \leq j < i\}$.

Figure 2 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, $\Phi$ is true is similar to that of Theorem 4.1.

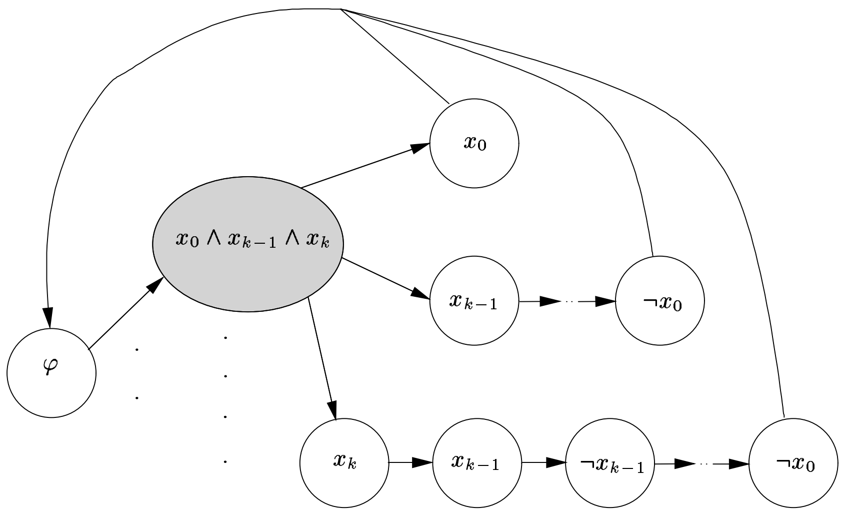


**Fig. 2.** Arena with bounded tree-width

## 5 Complexity Bounds for Union-Closed Games

We now turn our attention to 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 [6]. One consideration that makes them an interesting case to study is that they admit memoryless strategies for Player 1 [2]. 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 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 P. Indeed, it is a version of the alternating reachability problem. 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.

The Basis condition type is a succinct way of describing union-closed types. It is not even known if it is translatable to the Emerson-Lei condition type, the most succinct type considered above. However, the following result shows that the bounds obtained cannot easily be derived from the known completeness results of Streett games.

**Theorem 5.1.** *The Basis and Streett condition types are incomparable with respect to translatability. That is, neither is translatable to the other.*

*Proof.* To show there is no translation from Streett games to Basis games, let $V = \{x_0, x_1, \dots, x_{2k-1}\}$, and consider the Streett game with winning condition described by the pairs $\big\{(L_i, \emptyset) : 0 \le i < k\big\}$, where $L_i = \{x_{2i}, x_{2i+1}\}$. Note that the family of sets described by this condition is $\mathcal{F} = \big\{X \subseteq V : \forall i\, X \not\subseteq V \setminus L_i\big\}$. Any basis for $\mathcal{F}$ must include the minimal elements of $\mathcal{F}$. However, the minimal elements include

$$\mathcal{M} = \big\{\{v_0, v_1, \dots, v_{k-1}\} : v_i \in \{x_{2i}, x_{2i+1}\}\big\},$$

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 $V = \{x_0, x_1, \dots, x_{2k-1}\}$, and consider the family $\mathcal{F}$ of sets formed by closing

$$\mathcal{B} = \big\{\{x_{2i}, x_{2i+1}\} : 0 \le i < k\big\}$$

under union. Note that this is the same construction as for the proof of Theorem 3.4. 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.

Nevertheless, deciding Basis games is still in co-NP.

**Proposition 5.2.** *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 $O\big(|\mathcal{B}|(|V|+|E|)\big)$ 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 elements not contained in any SCC of $X$ is removed at step 3). Furthermore, any strongly connected set of $V$ 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 $O\big(|\mathcal{B}|(|V| + |E|) + |E|\big)$ time.

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

**Proposition 5.3.** *The Superset condition type is translatable to the Basis condition type.*

*Proof.* Let $(V, E, \{A_0, A_1, \ldots, A_{k-1}\})$ be a Superset game. The following basis:

$$\bigcup_{i=0}^{k-1} \Big( \{A_i\} \cup \{A_i \cup \{x\} \;:\; x \notin A_i\} \Big)$$

describes the same family of sets and has size at most $k|V| + 1$.

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

**Theorem 5.4.** *Deciding Superset games is* co-NP*-complete.*

*Proof.* Membership of co-NP follows from the previous two propositions. 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, $x_i, \neg x_i \in V_0$ and $x'_i \in V_1$;
- 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 $k$; and
- the winning condition is specified by the set

$$\mathcal{M} = \big\{\{l_i \in V_0 \;:\; l_i \text{ is a literal of } C\} \text{ for every clause } C \text{ of } \varphi\big\},$$

Take $x_0$ to be the initial vertex.

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 clause of $\varphi$ (and hence $\varphi$) true. Thus Player 0 wins this game if, and only if, there is no truth assignment which makes $\varphi$ false.

**Corollary 5.5.** *Deciding Basis games is* co-NP*-complete.*

*Succinctness Results* We finish this section with two succinctness results.

**Proposition 5.6.** *The Superset condition type is more succinct than an explicit presentation of an upward-closed set.*

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

**Corollary 5.7.** *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 5.6 and Proposition 5.3 (as "translatable" is transitive). The other direction is straightforward, the explicit presentation itself suffices as a basis.

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.

## 6 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 [8]. 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 $S2S$, $\mu$-calculus and $CTL^*$ – 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.

**Definition 6.1.** *Let* $[k] = \{1, 2, \ldots, k\}$*. An infinite, $k$-ary branching tree labelled by elements of $\Sigma$ is a function* $t : [k]^* \to \Sigma$*.* Nodes *of a tree are elements of its domain, the* root *of a tree is the empty string.*

**Definition 6.2.** *A* subtree *of tree $t$ rooted at $u \in [k]^*$ is the tree $t_u$ defined as $t_u(v) = t(u.v)$ 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*

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

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 and $k$ distinct transition relations.

**Definition 6.3.** *An infinite (Muller) (k-ary) tree automaton is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, \mathcal{F})$ where*

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

Given an infinite, $k$-ary branching tree $t$ labelled by elements of $\Sigma$, a run of $\mathcal{A}$ 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 $r(w) = q$, $r(w.1) = q_1$, $r(w.2) = q_2$, $\dots$, $r(w.k) = q_k$, and $t(w) = a$, then $(q, a, q_1, q_2, \dots, q_k) \in \delta$.

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 $\mathcal{A}$ *accepts* $t$ if there is a successful run of $\mathcal{A}$ on $t$.

The close connection between automata and games can be established by considering the game where Player 0's moves consist in choosing a transition in $\delta$ to make from a current state, and Player 1's moves consist in choosing which branch of the tree to descend. With this translation in mind, the non-emptiness problem for infinite tree automata (given an automaton, determine whether there is a tree it accepts) reduces to the win-set game with

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

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 6.4.** *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 (satisfiability of a quantified boolean formula [QBF]).

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

- $Q = \{q_\varphi\} \cup \{q_x, q_{\neg x} \,:\, \text{for all variables } x\}$
- $\Sigma = \{a\}$ (as $\Sigma$ is a singleton, for ease of reading we omit $a$ from the description of $\delta$)
- $q_I = q_\varphi$
- $\delta \subseteq Q \times Q^3$ given by:
  - for each clause $(l_0 \wedge l_1 \wedge l_2) \in \varphi$, $(q_\varphi, q_{l_0}, q_{l_1}, q_{l_2}) \in \delta$;
  - $(q_{x_i}, q_{x_{i-1}}, q_{x_{i-1}}, q_{x_{i-1}}) \in \delta$ for $0 < i < k$;
  - $(q_{\neg x_i}, q_{x_{i-1}}, q_{x_{i-1}}, q_{x_{i-1}}) \in \delta$ for $0 < i < k$;
  - $(q_{x_0}, q_\varphi, q_\varphi, q_\varphi) \in \delta$; and
  - $(q_{\neg x_0}, q_\varphi, q_\varphi, q_\varphi) \in \delta$.
- $\mathcal{F} = \big\{S_i, S_i \cup \{q_{x_i}\}, S_i \cup \{q_{\neg x_i}\} \;:\; 0 \leq i < k,\, i \text{ even}\big\}$ 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 $\mathcal{A}_\Phi$ accepts any tree is equivalent to asking if Player 0 has a winning strategy (from $q_\varphi$) on the win-set game used in Theorem 4.1.

The model checking problem (does a given automaton accept a given tree?) 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 regular, a game with finite arena can be constructed, and we can apply Theorem 6.4 to obtain the following corollary.

**Corollary 6.5.** *Given a regular, infinite, $k$-ary branching tree $t$ (represented as a transition system) and a Muller automaton $\mathcal{A} = (Q, \Sigma, \delta, q_I, \mathcal{F})$, asking if $\mathcal{A}$ accepts $t$ is* PSPACE*-complete.*

*Proof.* PSPACE hardness follows from the proof of Theorem 6.4, 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. Let $(S, t_1, t_2, \ldots, t_k)$ denote the transition system representing the tree $t$. The required Muller game is given by the following.

- $V_0 = Q \times S$.
- $V_1 = Q \times S \times Q^k$.
- There is an edge from $(q, s) \in V_0$ to $(q, s, q_1, \ldots q_k) \in V_1$ if $(q, a, q_1, \ldots, q_k) \in \delta$ where $a$ is the label of $s$.
- There is an edge from $(q, s, q_1, \ldots, q_k) \in V_1$ to $(q_i, t_i(s)) \in V_0$ for $1 \leq i \leq k$.
- $Q$ is the set of colours,
- $\chi : V \to Q$ 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 $(q_I, s_0)$ – where $s_0$ is the state corresponding to the root of $t$) in this game if, and only if, $\mathcal{A}$ accepts $t$.

## 7 Conclusion

We have considered the complexity of deciding the winner in a variety of regular 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 four different condition types that can be used to describe regular 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. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Trans. Amer. Math. Soc.*, 138:295–311, 1969.
2. Stefan Dziembowski, Marcin Jurdziński, and Igor 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. Allen Emerson and Charanjit 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. Allen Emerson and Chin-Laung Lei. Modalities for model checking: Branching time strikes back. In *Proceedings of the 12th Annual ACM Symposium on Principles of Porgramming Languages*, pages 84–96, 1985.
5. Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science.* Springer, 2002.
6. Hajime Ishihara and Bakhadyr 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.
7. Salvatore La Torre, Aniello Murano, and Margherita Napoli. Weak Muller acceptance conditions for tree automata. In Agostino Cortesi, editor, *3rd International Workshop on Verification, Model Checking and Abstract Interpretation*, volume 2294 of *Lecture Notes in Computer Science*, pages 240–254. Springer, 2002.
8. Robert McNaughton. Finite-state infinite games. Technical report, Project MAC, Massachusetts Institute of Technology, USA, 1965.
9. Robert McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993.
10. Anil Nerode, Jeffery B. Remmel, and Alexander Yakhnis. McNaughton games and extracting strategies for concurrent programs. *Annals of Pure and Applied Logic*, 78(1-3):203–242, 1996.
11. Jan Obdržálek. Fast mu-calculus model checking when tree-width is bounded. In Warren A. Hunt Jr. and Fabio Somenzi, editors, *Proceedings of 15th International Conference on Computer Aided Verification*, volume 2725 of *Lecture Notes in Computer Science*, pages 80–92. Springer, 2003.
12. Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200:135–183, 1998.