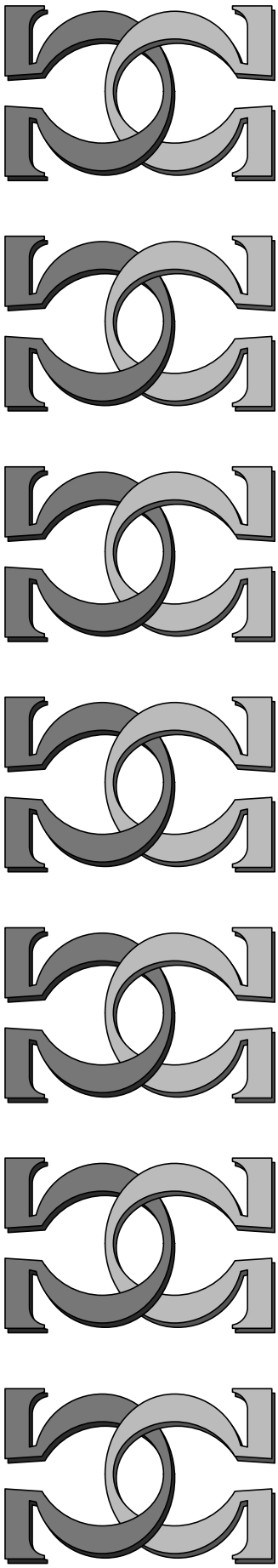

CDMTCS
Research
Report
Series

Complexity of Some Infinite Games Played on Finite Graphs

Hajime Ishihara
Japan Advanced Institute of Science and Technology, Japan

Bakhadyr Khoussainov
Department of Computer Science
University of Auckland

CDMTCS-178
February 2002

Centre for Discrete Mathematics and
Theoretical Computer Science

Complexity of Some Infinite Games Played on Finite Graphs

Hajime Ishihara
Japan Advanced Institute of Science and Technology, Japan
ishihara@jaist.ac.jp

Bakhadyr Khoussainov
Computer Science Department, The University of Auckland, New Zealand
bmk@cs.auckland.ac.nz

1 Introduction and Basic Concepts

Games played on finite graphs were first introduced by McNaughton in [6]. McNaughton, using the ideas of the paper by Gurevich and Harrington [3], proved that winners in his games have finite state winning strategies. Later based on McNaughton games, Nerode, Remmel and Yakhnis in a series of papers (see [7] and [8], for example) developed foundations of concurrent programming by identifying distributed concurrent programs with finite state strategies and studied complexities of finding winners in McNaughton games. Dinneen and Khoussainov use McNaughton games for modelling and studying structural and complexity-theoretical properties of update networks (see [1]). Later in [2] Bodlaender, Dinneen and Khoussainov generalize the study of update networks by introducing the concept of relaxed update network. They proved that it is possible to detect in polynomial time whether or not a given game represents a relaxed update network. In this paper we continue the line of research of the above mentioned work and begin with the following definition from [6]:

Definition 1 *A* **game** Γ *is a tuple* $(S \bigcup A, E, W, \Omega)$*, where:*

1. *The sets* S *and* A *are disjoint and finite, where* S *is the set of positions for Survivor and* A *is the set of positions for Adversary,*
2. *The set* E *of edges is such that* $E \subseteq A \times S \bigcup S \times A$ *and for all* $s \in S$ *and* $a \in A$ *there are* $a' \in A$ *and* $s' \in S$ *for which* $(s, a'), (a, s') \in E$*,*
3. *The set* W *is a subset of* $S \bigcup A$ *and* $\Omega \subseteq 2^W$*.*

The graph $\mathcal{G} = (V, E)$*, where* $V = S \cup A$*, is called the* **system** *or the* **graph of the game***, the pair* (W, Ω) *is the* **specification***, and each set* $U \in \Omega$ *is a* **winning set***.*

In game Γ, a **play (from p_0)** is an infinite sequence $\pi = p_0, p_1, \ldots$ such that $(p_i, p_{i+1}) \in E$, $i \in \omega$. Consider the set $Inf(\pi)$ consisting of those positions $p \in V$ that appear infinitely often in play π. Survivor **wins** the play if $Inf(\pi) \in \Omega$; otherwise, Adversary wins. The **histories** of the play π are its finite prefixes. The set $H(S)$ consists of all histories of all plays whose last positions are in S. The set $H(A)$ is defined similarly. A **strategy** for *Survivor* is a function $f : H(S) \to A$ such that $(q_n, f(u)) \in E$ for all $u = q_0 \ldots q_n \in H(S)$. A **strategy** for *Adversary* is defined similarly.

Let f be a strategy for a player and p be a position. Consider all the plays that begin from p which are played when the player follows the strategy f. We call these **plays consistent with f from p.**

Definition 2 *The strategy f of a player is a* **winning strategy** *from position p if all plays consistent with f from p are won by the player. In this case we say that the player* **wins the game** *from p.* **To decide game** Γ *means to find all the positions q in the game from which Survivor wins. We denote this set of position by $Win(S)$. The set $Win(A)$ is defined similarly*[1]*.*

In [6] McNaughton proved that there is an algorithm that decides any given McNaughton game. McNaughton's algorithm is quite inefficient, however. In [7] Nerode, Remmel and Yakhnis improved MacNaughton algorithm by showing that it takes $O(|W|!2^{|W|}|W||E|)$ time to decide a given game Γ. Thus, a natural question arises as to under which conditions, put either on the specifications or the systems, the games can be decided more effectively. Here is a list of some results and definitions related to this question.

A natural specification is to require Survivor to update every node of the system infinitely often. This is formalized as follows.

Definition 3 *A game Γ is an* **update game** *if $W = V$ and $\Omega = \{V\}$. If Survivor wins this game, then we call this game an* **update network**.

Update games can be decided in polynomial time as shown in [1]:

Theorem 1 *There exists an algorithm that given a game decides in $O(|V||E|)$ time whether or not the game is an update network.*

We use this theorem (most of the time without a reference to it) in obtaining several results of this paper. The proof of this theorem is based on finding certain structural properties of update networks. In [2] Theorem 1 has been generalized.

Definition 4 *A game Γ is a* **relaxed update game** *if $U \bigcap V = \emptyset$ for all distinct $U, V \in \Omega$. If Survivor wins this game, then we call Γ a* **relaxed update network**.

In [2], similar to the notion of rank defined in Gurevich-Harrington [3], certain natural concepts (such as forcing) are introduced for the investigation of relaxed update games, and the following theorem is proved:

[1] Any McNaughton game Γ is a Borel game. Hence, by the known result of Martin, Γ is determined (see [5]). Therefore $Win(S) \bigcup Win(A) = S \bigcup A$.

Theorem 2 *There exists an algorithm that given a game decides in $O(|V|^2|E|)$-time whether or not the game is a relaxed update network.*

In this paper our goal is three fold. Firstly, we generalize the theorems above by greater exploiting the ideas of the proofs of Theorem 1 and Theorem 2. Secondly, we give other types of natural specifications when the winners can be determined in polynomial time with a parameter. Finally, we study the interactions between efficent winning strategies, complexity of finding such strategies, and the structural properties of the underlying graphs for games with an emphasis to update games. We also briefly consider the relationship between McNaughton games and temporal logic.

2 Preliminary Results

Given a game Γ and a subset $X \subseteq V$, a node v is in the set REACH(S, X) if *Survivor* can force every play starting at v into X after a finite number of steps.

Lemma 1 *[2] The set REACH(S, X) can be computed in $O(|V| + |E|)$ time.*

Proof. We build a set R, that will eventually be REACH(S, X). Initially, $R = X$. If a node $x \in S$ has an edge to a node in R, then x is added to R. If a node $x \in A$ has only edges to nodes in R, then x is added to R. ¿From every node in R *Survivor* can force plays to go to a node in X. When no nodes can be added to R anymore, then REACH(S, X)=R. *Adversary* has a strategy to stay inside $V \backslash$REACH(S, X) when game begins in a node from $V \backslash$REACH(S, X). The procedure of constructing REACH(S, X) can be implemented in $O(|V| + |E|)$ time, by giving each node not in X a counter, that is initially 1 for nodes owned by *Survivor* and its outdegree for nodes owned by *Adversary*. Whenever we add a node v to R, we subtract 1 from the counters of each node with an edge to v; when a counter becomes 0 then the node is also added to R. □

Let $v \notin$ REACH(S, X). Iteratively define the set AVOID(v, A, X): initially, AVOID(v, A, X)=$\{v\}$. For $x \in A \cap$AVOID(v, A, X) we add a neighbor y of x into AVOID(v, A, X) if $(x, y) \in E$ and $y \notin$ REACH(S, X). For $x \in S \cap$AVOID(v, A, X) we add all neighbors of x into AVOID(v, A, X). From Lemma 1 we obtain the following:

Lemma 2 *The set AVOID(v, A, X) has the following properties:*

1. *AVOID(v, A, X) can be constructed in $O(|V| + |E|)$ time.*
2. *AVOID$(v, A, X) \cap$ REACH$(S, X) = \emptyset$.*
3. Adversary *has a strategy such that when a play visits a node in AVOID(v, A, X) then all nodes visited afterwards are in AVOID(v, A, X).*
4. *For all s in AVOID$(v, A, X) \cap S$ and all $a \in A$ if $(s, a) \in E$ then a is in AVOID(v, A, X).*

Note that the sets REACH(A, X) and AVOID(v, S, X) can be defined in a similar matter. We will now need some notations that will be used later. The strategy for a player P to force the plays into X from a node v will be denoted by $Force_X^{P,v}$. Similarly, the strategy that keeps out all the plays from v to enter the set X will be denoted $Avoid_X^{P,v}$.

3 Games With Separable Winning Sets

Given a system we would not like the system to enter useless states during a computation. This naturally suggests that the specification (W, Ω) of the game to be such that $W = S \bigcup A$. Another natural assumption on the specification is that each winning condition U to be distinguishable from all other winning condition U' in Ω. We formalize this as follows:

Definition 5 *Game Γ is* **fully separated** *if $W = S \bigcup A$ and for each $U \in \Omega$ there is a s_U, called* **separator**, *such that $s_U \in U$ but $s_U \notin U'$ for all $U'\Omega$ distinct from U.*

Thus, the separator s_U for $U \in \Omega$ can be thought as a certificate of the winning set U. Now our goal is to provide a polynomial time algorithm that decides fully separted games. We use techniques developed in [1] and [2].

Definition 6 *[2] A winning set $U \in \Omega$ is* **S-closed** *if it satisfies the following conditions:*

1. *For each $s \in U \cap S$ there is an $a \in U \cap A$ such that $(s, a) \in E$.*
2. *For each $a \in U \cap A$ and all s such that $(a, s) \in E$ we have $s \in U$.*

Thus, this definition informally tells us the following. If a play arrives to a node v in an S-closed set U then Survivor is able to always keep all the plays after v inside U no matter what the oppenent does. Here is a lemma which is true for any McNaughton game and therefore is of independent interest:

Lemma 3 *Let Γ be a McNaughton game. Then if Survivor wins Γ from a position p then one of the winning sets U in Ω must be S-closed.*

Proof. In order to prove the lemma, we assume the opposite and then construct a winning strategy for Adversary thus contradicting the assumption of the lemma.

Since each $U \in \Omega$ is not S-closed, there is a $p_U \in U$ that satisfies one of the following conditions:

1. $p_U \in S$ and for all $a \in A$ if $(p_U, a) \in E$ then $a \notin U$.
2. $p_U \in A$ and there is an s such that $(p_U, s) \in E$ and $s \notin U$.

We call p_U a **witness**. Here is a strategy for Adversray. Let $p_0, \ldots, p_n$ be a finite play. If p_n is not a witness and $p_n \in A$ then Adversray moves to any node s such that $(p_n, s) \in E$. Assume that p_n is a witness. Let $U_0, \ldots, U_{k-1}$ be *all* winning sets in Ω for which p_n is a witness. Note that if $p_n \in S$ then for every a such that $(p_n, a) \in E$ we have $a \notin U_0 \cup \ldots \cup U_{k-1}$. Assume now that $p_n \in A$. Let i be the number of times p_n appears in the finite play $p_0, \ldots, p_n$. Then Adversray moves to any s that does not belong to $U_{i+1(mod(k))}$. The basic idea is that, once p_n is reached, Adversary leaves the sets $U_0, \ldots, U_{k-1}$ turn by turn making in a cyclic manner.

We claim that the strategy described is a winning strategy for Adversary. Indeed let $\pi = p_0, p_1, \ldots$ be a play consistent with the strategy. Assume that $In(\pi) = U$ and $U \in \Omega$.

Let n be the first position after which no nodes outside of U appear in π. Then since a witness p_U appeares in $In(\pi)$ infinitely many times it must be the case that p_U appears in π after position n. Hence, by the definition of the strategy we described, there is a position $j > n$ such that $p_j \notin U$. This contradicts with the choice of n. The lemma is proved. □

We now need the following lemma that characterizes update networks (see Definition 3) in terms of the sets REACH(S, X).

Lemma 4 *Survivor wins an update game* Γ *if and only if* $x \in$ *REACH*$(S, \{y\})$ *for all* $x, y \in V$.

Proof. Assume that $x \in \text{REACH}(\{y\})$ for all $x, y \in V$. List all the nodes $v_0, \ldots, v_n$. Survivor cycles by forcing the plays to visit v_0, then v_1, etc. This shows that Survivor wins the game Γ.

Assume that there exists x, y such that $x \notin \text{REACH}(S, \{y\})$. Then Adversray uses the strategy $Avoid_y^{A,x}$ as soon as a play is at the node x. By by Part 3 of Lemma 2 Adversary wins the game. The lemma is proved. □

The next lemma gives another sufficient condition for Adversary to win a fully separated game.

Lemma 5 *Let* Γ *be a fully separated game* Γ *such that every* $U \in \Omega$ *satisfies one of the following properties:*

1. *U is not S-closed,*

2. *U is S-closed and Adversary wins the update game $(U, \{U\})$.*

Then Adversray wins the game Γ.

Proof. By the lemma above, if $U \in \Omega$ is an S-closed set and $(U, \{U\})$ is not an update network then there exists a pair x_U, y_U such that $x_U \notin \text{REACH}(S, \{y_U\})$. If U is not S-closed then we take a witness p_U for U as in the proof of Lemma 3. Now we construct the following strategy g for Adversray. Let $p_0, \ldots, p_n$ be a finite play such that $p_n \in A$. Let p_i be the last separator seen in the finite play and $p_i = s_U$ for some $U \in \Omega$. There are three cases to consider.

Case 1. The set $\{p_i, \ldots, p_n\}$ is not a subset of U then Adversary chooses any p_{n+1} for which $(p_n, p_{n+1}) \in E$.

Case 2. The set $\{p_i, \ldots, p_n\}$ is a subset of U and U is S-closed. In this case Adversary follows $Avoid_{y_U}^{A,x_U}$ strategy.

Case 3. The set $\{p_i, \ldots, p_n\}$ is a subset of U and U is not S-closed. Then if $p_n = p_U$ then Adversray chooses p_{n+1} such that $p_{n+1} \notin U$ and $(p_n, p_{n+1}) \in E$.

Let $\pi = p_0, p_1, \ldots$ be a play consistent with g. Assume that $Inf(\pi) = U$ and $U \in \Omega$. Let n be the first position after which no nodes outside of U appear in π. Then since the separator s_U appeares infinitely many times in $In(\pi)$ it must be the case that s_U appears in π after position n. Hence, either *Case 2* or *Case 3* is applied. In *Case 2*, Adversary wins as the node y_U will not be visited infinitely often, and hence $Inf(\pi) \neq U$ which is a

contradiction. In *Case 3*, we will have a contradiction with the choice of position n. The lemma is proved. □

We are now ready to prove the following theorem whose proof uses the lemmas proved above as well as Theorem 1.

Theorem 3 *There exists an algorithm that decides any given fully separated game* Γ *in* $O(|V|^2|E|)$ *running time.*

Proof. Let p be the node from which all the plays begin. We describe the following algorithm $Procedure(\Gamma, p)$:

1. For each $U \in \Omega$, find whether or not $U \in \Omega$ is S-closed. If each $U \in \Omega$ is not S-closed then Adversary wins the game from p.

2. For each S-closed $U \in \Omega$ find whether or not the game $(U, \{U\})$ is an update network.

3. Let X be the union of all $U \in \Omega$ so that $(U, \{U\})$ is an update network and U is S-closed. If $X = \emptyset$ then Adversary wins the game from p.

4. If $p \in$ REACH (S, X) then Survivor wins the game from p.

5. If $p \notin$ REACH (S, X) then construct the game $\Gamma_1 = (V_1, E_1, W_1, \Omega_1)$ as follows. The set V_1 of nodes is AVOID(p, A, X), the set E_1 is the restriction of E to V_1, the set W_1 is V_1, and Ω_1 consists of all $U \in \Omega$ such that $U \subset V_1$. Note that $|V_1| < |V|$. Run $Procedure(\Gamma_1, p)$.

It is not hard to see that the algorithm runs in $O(|V|^2|E|)$ time. This proves the theorem. □

4 Games with Linear Winning Conditions

Let Γ be a game. The winning conditions set Ω is a partially ordered set in which the partial order is defined by means of the set-theoretic inclusion. We call it the **partial order associated with** the game Γ. The previous section deals with those winning conditions whose associated partial orders form antichains, that is $U \not\subseteq V$ for all distinct $U, V \in \Omega$. In this section, we investigate a dual case by considering games whose associated partial orders are linearly ordered.

Definition 7 *A game* Γ *is a* **linear game** *if the set* Ω *forms a linear order* $U_1 \subset U_2 \subset \ldots \subset U_n$ *and* $W = V$. *We call* n *the* **length of the winning conditions**.

Our goal is to show that linear games can be decided in polynomial time if the length of the winning conditions is fixed. We begin with an example.

Example 1 *Consider the game* $\Gamma = (S \cup A, E, W, \Omega)$, *where:*

1. $S = \{s_1, s_2, \ldots, s_n\}$, $A = \{a_1, a_2 \ldots, a_n\}$, $W = S \cup A$,

2. $E = \{(s_i, a_i) \mid 1 \leq i \leq n\} \cup \{(a_i, s_{i+1}) \mid 1 \leq i \leq n-1\} \cup \{(a_i, s_1) \mid 1 \leq i \leq n\}$,

3. $\Omega = \{U_i \mid 1 \leq i \leq n\}$, *where* $U_i = \{s_1, a_1, \ldots, s_i, a_i\}$.

In this game Survivor wins the whole game. It is worth to note that Adversary wins each of the game whose winning conditions set is a proper subset of Ω. The basic reason for this is that Survivor has no choice at any given node $s_i \in S$ but to move to a_i.

We need some notations, definitions, and lemmas for our next theorem. Let X, Y be a subset of nodes in a given game. Then REACH(S, X, Y) is the set of all nodes v such that *Survivor* can force every play starting at v into X after a finite number of steps and staying inside Y. As in Lemma 1 it can be shown that the set REACH(S, X, Y) can be computed in $O(|V| + |E|)$ time.

Let $\Gamma = (V, E, \Omega)$ be a game with the winning conditions $\Omega = \{U_1 \subset U_2 \subset \ldots U_n\}$. Assume that $n > 1$. For each $i < n$ consider the linear game $\Gamma_i = (V, E, \Omega_i)$, where $\Omega_i = \{U_1 \subset U_2 \subset \ldots U_i\}$. Here is a simple lemma whose proof is left to the reader.

Lemma 6 *1. Survivor wins the game* Γ_1 *from position* p *if and only if* $p \in$ *REACH*(S, U_1)*, the set* U_1 *is* S*-closed, and Survivor wins the update game* $(U_1, \{U_1\})$.

2. If Survivor wins the game Γ_i *from* p *then Survivor wins games* Γ_j *from position* p *for all* $j \geq i$. □

For each $i < n$ consider the set X_i consisting of all positions p from which Survivor wins the game Γ_i. By the second part of the lemma above we have the sequence $X_1 \subseteq X_2 \subseteq \ldots \subseteq X_{n-1}$. Assume that $X_{n-1} \neq \emptyset$. For every position $p \notin$ REACH(S, X_{n-1}), consider the set AVOID(p, A, X_{n-1}). Now note that since $U_1 \subseteq U_2 \subseteq \ldots \subseteq U_n$ and $X_{n-1} \neq \emptyset$ it must be the case that $U_1 \subseteq X_{n-1}$. Therefore $Inf(\pi) \notin \Omega$ for any play π inside AVOID(p, A, X_{n-1}). We conclude that the following lemme is true:

Lemma 7 *Assume that* $X_{n-1} \neq \emptyset$*. Then Survivor wins the game* Γ *from position* p *if and only if* $p \in$ *REACH*(S, X_{n-1}). □

An important point of this lemma is that the original game can be reduced to a smaller linear game in case $X_{n-1} \neq \emptyset$.

Next we consider the case when $X_{n-1} = \emptyset$.

Lemma 8 *Assume that* $X_{n-1} = \emptyset$*. Then if Survivor wins the game* Γ *from position* p *then* $p \in$ *REACH*(S, U_n) *and* U_n *is* S*-closed.*

Proof. Clearly $p \in$ REACH(S, U_n). Assume that U_n is not S-closed. There is a $x \in U_n$ that satisfies one of the following conditions:

1. $x \in S$ and for all $a \in A$ if $(x, a) \in E$ then $a \notin U$.

2. $x \in A$ and there is an s such that $(x, s) \in E$ and $s \notin U$.

We describe a strategy for Adversary. Let $h = p_0 \dots p_n$ with $p_0 = p$ be a finite play with $p_n \in A$. We consider several cases.

Case 1. $p_n = x$. Then Adversary moves outside of U_n.

Case 2. $p_i \neq x$ for all $i = 1, \dots, n$. In this case Adversary plays his winning strategy in game Γ_{n-1} from p_0.

Case 3. Let $p_i = x$ and $x \notin \{p_{i+1}, \dots, p_n\}$. In this case Adversary plays his winning strategy in game Γ_{n-1} from p_{i+1}.

Assume that π is play consistent with the strategy described above. If x occurs in π finitely many times then there is a position p_i in the play after which the play becomes consistent with Adversray's winning strategy in game Γ_{n-1} from position p_i. Hence $Inf(\pi) \notin \Omega$ since $x \notin Inf(\pi)$ and $X_{n-1} = \emptyset$. If x occurs infinitely often then $Inf(\pi)$ has an element outside of U_n. Thus, the strategy is a winning strategy for Adversray. □

Assume that $X_{n-1} = \emptyset$. Let $x, y \in U_n$ be such that $x \notin \text{REACH}(S, \{y\}, U_n)$. Consider the set $\text{AVOID}(x, A, \{y\})$. We can define a new game denoted by $\Gamma(x, y)$ such that the graph of the game is $\text{AVOID}(x, A, \{y\})$, and the set $\Omega(x, y)$ of winning conditions are those $U \in \Omega$ which are subsets of $\text{AVOID}(x, A, \{y\})$. Note that if $\Omega(x, y)$ is the empty set then x is a winning position of Adversary in the original game. Also, $\Gamma(x, y)$ is a linear game and the length of its winning conditions is strictly less than n. Here is our next lemma.

Lemma 9 *Assume that $X_{n-1} = \emptyset$. Survivor wins the linear game Γ from position p if and only if p belongs to REACH(S, U_n), U_n is S-closed and one of the following two conditions is satisfied:*

1. *Survivor wins the update game $(U_n, \{U_n\})$.*
2. *For any pair $x, y \in U_n$ of nodes if $x \notin$ REACH$(S, \{y\}, U_n)$ then Survivor wins the game $\Gamma(x, y)$ from p while staying inside U_n.*

Proof. Assume that Survivor wins the game Γ from position p. Clearly it must be the case that $p \in \text{REACH}(S, U_n)$. Now assume that none of the conditions is true. We need to describe a winning strategy for Adversary. By the assumption, there must exist $x_0, y_0 \in U_n$ such that x_0 does not belong to the set $\text{REACH}((S, \{y_0\})$ and Adversray wins the game $\Gamma(x_0, y_0)$ while staying inside U_n. We fix x_0 and y_0. Here is now a strategy for Adversary. Let $h = p_0, \dots, p_m$ be a finite play from p.

Case 1. $p_i = x_0$ for some $i \leq m$ and all nodes in h after p_i are in U_n. In this case Adversary follows his winning strategy (from position p_i) in game $\Gamma(x_0, y_0)$.

Case 2. Suppose that *Case 1* does not hold and all nodes in h are in U_n. In this case Adversary follows his winning strategy (from position p) in game $(V, W, \{U_1, \dots, U_{n-1}\})$.

Case 3. $p_i \notin U_n$ for some $i \leq m$ and no node in h after p_i is x_0. In this case Adversary follows his winning strategy (from position p_i) in game $(V, W, \{U_1, \dots, U_{n-1}\})$.

We need to show that thus described strategy is a winning strategy for Adversary. Let $\pi = p_0, p_1, p_2, \dots$ be a play, where $p = p_0$, consistent with the strategy. Assume that $p_i = x_0$ for some i so that $p_j \in U_n$ for all $j > i$. Then Adversary follows his winning strategy in $\Gamma(x_0, y_0)$. Note that $y_0 \notin Inf(\pi)$. We conclude that $Inf(\pi) \notin \Omega$. Assume that $p_i \neq x_0$

for all p_i after some $p_j \notin U_n$. Then the play is consistent with the Adversary's winning strategy in game $(V, W, \{U_1, \ldots, U_{n-1})$ after p_j. Hence $Inf(\pi) \notin \{U_1, \ldots, U_{n-1}\}$. Since $x_0 \notin Inf(\pi)$ we conclude that $Inf(\pi) \neq U_n$. Assume that x_0 and y_0 appears infinitely often in π. This means π contains infinitely many nodes outside of U_n. Hence $Inf(\pi) \notin \Omega$. Thus, the strategy is winning strategy for Adversary. This is contradiction.

Now assume that one of the two conditions is satisfied and $p \in$ REACH (S, U_n). Clearly if the first conditions is true then Survivor wins the game. Assume that the first condition is not satisfied. Let $p = x_0, x_1, \ldots, x_k$ be the list of all nodes in U_n. Survivor's strategy is as follows. Initially $i = 0$ and the current position is p. If the current position q of a play is in REACH$((S, \{x_{i+1(mod(k))}, U_n)$ then Survivor forces the play into $x_{i+1(mod(k))}$. As soon as $x_{i+1(mod(k))}$ is reached i is set to $i + 1(mod(k))$ and the current position is $x_{i+1(mod(k))}$. If the current position q is not in REACH$((S, \{x_{i+1(mod(k))}, U_n\})$ then Survivor plays his winning strategy inside $\Gamma(x_i, x_{i+1(mod(k))})$ while staying in U_n. We need to show that this is a winning strategy for Survivor. Let $\pi = p_0 p_1 p_2 \ldots$ (with $p_0 = p$) be a play consistent with the strategy. Let $U = Inf(\pi)$ be the infinity set of the play. Assume that $x_{i+1(mod(k))} \notin U$ for some i. This means that there is a position n in the play π such that all x_j with $j > n$ belong to AVOID$(x_i, A, x_{i+1(mod(k))})$. Since Survivor plays his winning strategy inside the game $\Gamma(x_i, x_{i+1(mod(k))})$ the set U must belong to Ω. □

¿From the lemmas above and Theorem 1 we now can derive the following result about complexity of deciding linear games.

Theorem 4 *There exists an algorithm that decides any linear game $\mathcal{G}$ with winning conditions $\{U_1, \ldots, U_n\}$ in $O(|V|^{2 \cdot n-1}|E|)$ running time. In particular, if n is fixed then deciding linear games with n winning conditions can be done in a polynomial time.*

Proof. We analyze the case when $n = 2$. We use the lemmas above. Constructing X_1 and checking if $p \in$ REACH(S, X_1) takes at most $O(|V||E|)$-time.

Assume that $X_1 = \emptyset$. Cheking that $p \in$ REACH(S, U_2) and $(U_2, \{U_2\})$ is an update network takes at most $(|V||E|)$-time.

Let us now compute the time needed to check the second condition in the lemma above. It is not hard to see that for each $y \in U_2$ the set AVOID$(A, \{y\})$=$V \setminus$ REACH$(S, \{y\})$ can be constructed in $O(|E| + |V|)$-time (see Lemma 2). For each $x \in$ AVOID$(A, \{y\})$ checking if Survivor wins $\Gamma(x, y)$ inside U_2 takes at most $O(|\text{AVOID}(x, A, \{y\})||E|)$-time. Therefore, by varying x, y we see that the total time does not exceed $O(|V|^2 \times |V||E|)$. This proves the theorem for $n = 2$.

The rest can be done by using recursion and the use of the previous lemmas. The theorem is proved. □

5 No-Memory Strategies, Complexity, and Structure

The goal in this section is twofold. On the one hand we show that finding efficient strategies in McNaughton games, even in a simple case such as update games, is an untractible problem. On the other hand, we show how efficient winning strategies can be used to

extract structural properties of the underlying graphs. In this section we consider update games only.

Argueably the most simple strategies are the ones that depend on the current node of a play and not any other part of its history. We single out such strategies in the following definition.

Definition 8 *A strategy for Survivor is a* **no-memory strategy** *if it is induced by a function $f : S \to A$ such that $(s, f(s)) \in E$ for all $s \in S$.*

Thus if f is a no-memory strategy and h is a finite play whose last symbol $last(h)$ is in S then Survivor's next move is $f(last(h))$. The next definition gives us a tool to analyze the structure of graph games.

Definition 9 *A cycle $a_0, s_0 \ldots, a_n, s_n$ in game graph $\mathcal{G}$ is a* **forced cycle** *if $(a_i, s) \in E$ implies $s = s_i$ for all $0 = 1, \ldots, n$.*

Here is our theorem that shows the interaction between no-memory winning strategies in update games and the structural properties of the underlying graphs.

Theorem 5 *Let Γ be an update game whose graph is $\mathcal{G}$. Then Survivor has a no-memory winning strategy if and only if the graph $\mathcal{G}$ forms a forced cycle.*

Proof. Assume that the graph $\mathcal{G}$ forms a forced cycle $a_0, s_0 \ldots, a_n, s_n$. Then the mapping $s_i \to a_{i+1(mod(n+1))}$ establishes a no-memory winning strategy for Survivor.

Assume that in game Γ Survivor has a no-memory winning strategy f. Consider a play $\pi = s_0, a_0, s_1, a_1, \ldots$ consistent with f. Thus $f(s_i) = a_i$ for all i. Since f is a no-memory winning strategy we have $Inf(\pi) = V$. In this play there exist positions i and $i + m$ such that $s_i = s_{i+m}$, $m > 0$, and no two Survivors nodes between positions i and $i + m$ coincide. It is not hard to see that $s_i, a_i, \ldots, a_{i+m}$ is the list of *all* the nodes of the graph as otherwise f would not be a winning strategy. Moreover, in this list if $k \neq t$ then $a_k \neq a_t$. Indeed, say $k < t$ and $a_k = a_t$. Then Adversary by always moving from a_k into s_{t+1} would win against strategy f. This would contradict the assumption that f is a winning strategy. Thus, $s_i, a_i, \ldots, a_{i+m-1}, s_{i+m}$ is in fact a forced cycle. The theorem is proved.

Corollary 1 *The problem of finding whether or not Survivor has a no-memory winning strategy in a given update game is NP-hard.*

Proof. We reduce the problem of finding a Hamiltonian path in a directed graph to the problem of interest. Let $\mathcal{G} = (V', E')$ be a directed graph. Construct an update game $\Gamma(\mathcal{G}) = (S \cup A, E)$ as follows:

1. $S = V'$, $A = \{a_{(v,w)} \mid (v, w) \in E'\}$.
2. $E = \{(s, a_{(s,w)}) \mid s \in S, a_{(s,w)} \in A\} \cup \{(a_{(v,s)}, s) \mid a_{(v,s)} \in A, s \in S\}$.

Basically, we subdevide each edge (v, w) of the original graph $\mathcal{G}$ by introducing new Adversary's node $a_{(v,w)}$ that is connected to v and w. Clearly, the construction of $\Gamma(\mathcal{G})$ is linear on the size of the graph $\mathcal{G}$. Moreover, it is easy to see that $\mathcal{G}$ has a Hamiltonian cycle if and only if Survivor has a non-memory winning strategy in $\Gamma(\mathcal{G})$. The corollary is proved. □

6 Games and Temporal Logic

We now say a few words in relation to connections with temporal logic. There are several ways to think about these games using the language of temporal logic. For example, one way to think about the specifications (W, Ω) is to identify them with classes of formulas of temporal logic. Formally, this can be established as follows as it is done in [4]. Given a system $(S \bigcup A, E)$, form propositions of temporal logic by identifying each $p \in S \cup A$ as an atomic proposition. In the inductive step, if ϕ and ψ are propositions then their Boolean combinations and the expressions $G\phi$ and $F\phi$ are also propositions. Semantics for these propsitions are the runs of the system $(S \bigcup A, E)$. Let $\pi = p_0, p_1, p_2, \ldots$ be a run and ϕ be formula. Let π^i be the sequence $p_i, p_{i+1}, \ldots$. One now can define what it means ϕ to be true on π, denoted by $\pi \models \phi$, by induction as follows. If $p_0 = p$ then $\pi \models p$. The case for Boolean connectives is defined naturally. For $\phi = F\psi$, $\pi \models \phi$ if $\pi^i \models \psi$ for some i. For $\phi = G\psi$, $\pi \models \phi$ if $\pi^i \models \psi$ for all i. Thinking of G as "globally" and of F as "future", we can represent specifications (W, Ω) in the language of temporal logic. Thus, given a system $(S \bigcup A, E)$ and a specification ϕ we can now ask whether or not the system satisfies ϕ. The satisfaction can be expressed in terms of winning. Namely, the system **satisfies** ϕ if Survivor has a strategy so that in every play π consistent with the strategy the formula ϕ is true. For example, for the system $(S \bigcup A, E)$ with nodes $\{p_0, \ldots, p_{n-1}\}$, the specification $(p_0 \vee \ldots \vee p_n) \& \&_{0 \leq i \leq n-1} G(p_i \rightarrow F p_{i+1(mod\ n)})$ tells us that Survivor must visit every node infinitely often. This is in fact a specification of update networks in terms of temporal logic. Thus, one can study the following natural questions:

1. (Model checking complexity) Given a system, what is the time complexity of finding whether or not a given temporal formula is staisfied in the system?

2. (Implementation complexity) Given a formula ϕ, what is the complexity of finding whether or not a given system satisfies ϕ?

3. (Combined complexity) What is the complexity of finding, given a formula ϕ and the system $\mathcal{A}$, that $\mathcal{A}$ satisfies ϕ?

Fore details on research on the relationship between verification and specifications of systems, temporal logic and games see Vardi [9].

7 Conclusion

The results of this paper can be generalized. For example, in fully separated games or in linear games the condition $W = S \bigcup A$ can be removed. Techniques for a such generalization can be found in [2]. We expect that it is possible to decide linear games more efficiently than the time bound presented in Theorem 4. We think that the methods and techniques developed in this paper, papers [1] and [2] give sufficient tools for a deep study of games from computational, algebraic and logical points of view. One can hope to provide fast algorithms for deciding those games in which the winning configurations can be decided efficiently, e.g. when the number of winning conditions is fixed. Section 5 shows

that interesting results can be obtained in relation to implementing winning strategies by finite automata. This is a topic of our future papers. Note that apart from Corolloary 1 neither in this nor in any of the previous papers [1] [2] the topic on complexity of extracting winning strategies has been discussed. To our knowledge the only paper that deals with this issue explicetly is one by Nerode, Remmel, and Yakhnis [7]. A fruitful direction is related to the study of connections with temporal logic briefly described in the last section. One can also study the effect of the topology of the systems on finding the winners and winning strategies. As it is seen, much more is needed to be done.

References

[1] M. J. Dinneen and B. Khoussainov. Update networks and their routing strategies. In *Proceedings of the 26th International Workshop on Graph-Theoretic Concepts in Computer Science, WG2000*, volume 1928 of *Lecture Notes on Computer Science*, pages 127–136. Springer-Verlag, June 2000.

[2] H.L. Bodlaender, M.J. Dinneen and B. Khoussainov. On Game-Theoretic Models of Networks, in Algorithms and Computation (ISAAC 2001 proceedings), LNCS 2223, P. Eades and T. Takaoka (Eds.), p. 550-561, Springer-Verlag Berlin Heidelberg 2001.

[3] Y. Gurevich and L. Harrington. Trees, Automata, and Games, STOCS, 1982, pages 60–65.

[4] J. F. Knight and B. Luense. Control Theory, Modal Logic, and Games, In *Hybrid Systems IV*. Panos J. Antsaklis, Wolf Kohn, Anil Nerode, Shankar Sastry (Eds.), volume 1273 of *Lecture Notes in Computer Science*, pages 160–173. Springer, 1997.

[5] D. Martin. Borel Determinacy. Ann. Math. Vol 102, 363-375, 1975.

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

[7] A. Nerode, J. Remmel, and A. Yakhnis. McNaughton games and extracting strategies for concurrent programs. *Annals of Pure and Applied Logic*, 78:203–242, 1996.

[8] A. Nerode, A. Yakhnis, V. Yakhnis. Distributed concurrent programs as strategies in games. Logical methods (Ithaca, NY, 1992), pages 624–653, *Progr. Comput. Sci. Appl. Logic*, 12, Birkhauser Boston, Boston, MA, 1993.

[9] M. Vardi. An automata-theoretic approach to linear temporal logic. Proceedings of the VIII Banff Higher Order Workshop. Springer Workshops in Computing Series, Banff, 1994.