CDMTCS
Research
Report
Series

Games Played on Finite Graphs and Temporal Logic

Bakhadyr Khoussainov
Department of Computer Science
University of Auckland
Auckland, New Zealand

CDMTCS-174
January 2002

Centre for Discrete Mathematics and
Theoretical Computer Science

Games Played on Finite Graphs and Temporal Logic

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

Our aim is to study reactive finite state systems (e.g. communication networks, banking systems, airtraffic control systems) by means of game-theoretic methods. A reactive system acts upon the inputs from environment by changing its states. The goal of the system is to satisfy given specifications no matter how environment behaves. We model this situation using games played on finite graphs first introduced by McNaughton [6].

Informally, an interaction between the system and environment (which is assumed to be a finite state system) is a sequence, called **run**, $\pi = s_0, a_0, s_1, a_1, \ldots$ of states of the system and the environment. The success of the run depends upon whether or not the system satisfies a given specification. Since both systems are finite, some states in the run appear infinitely often. We denote the set of these states by $Inf(\pi)$. An idea suggested by this observation is that specifications can be expressed in terms of infinity sets $Inf(\pi)$ of the runs. The run $\pi = s_0, a_0, s_1, a_1, \ldots$ can be thought as a play between two players, Survivor and Adversary. Survivor moves to a_i from s_i, and Adversary responds by moving to s_{i+1} from a_i for $i \in \omega$. The goal of Survivor is to satisfy the specification given while the goal of Adversary is not to allow the specification to be satisfied. We are now ready to give the following definition.

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

1. *S and A are disjoint finite sets called the sets of* **positions** *for Survivor and Adversary, respectively;*
2. *E is the subset of $A \times S \bigcup S \times A$ so that for all $s \in S$ there is an $a \in A$ such that $(s, a) \in E$ and for all $a \in A$ there is an $s \in S$ such that $(a, s) \in E$,;*
3. *W is a subset of $S \bigcup A$; and*
4. *Ω is a subset of 2^W.*

The graph (V, E), where $V = S \cup A$, is called the **system** *(or the graph of the game), and the pair (W, Ω) is a* **specification**.

Given a game Γ, a play (from p_0) is an infinite sequence $\pi = p_0, p_1, \ldots$ such that $(p_i, p_{i+1}) \in E$ for each $i \in \omega$. Survivor wins the play if $Inf(\pi) \in \Omega$; otherwise Adversary wins. We say that a player wins the game if the player has a winning strategy in the game. **To decide game** Γ means to find all positions q in the game from which Survivor wins. We denote this set of position by $Win(S)$. Similarly, $Win(A)$ is the set of all winning positions for Adversary. It is not hard to see that the games defined are Borel games, and hence each game is determined by the known result of Martin [5]. Therefore $Win(S) \bigcup Win(A) = S \bigcup A$. In [6] McNaughton proved that there is an algorithm that given a game finds the sets $Win(S)$ and $Win(A)$. In fact, the proof implicitly tells us that the winner of any of these games has a finite state winnng strategy. 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 under which conditions (which can be put either on the specifications or systems) the games can be decided more efficiently. Here is a list of some results related to this question.

One of the natural specifications consists of requiring Survivor to update every node of the system infinitely often. This is formalized as follows.

*This is a short abstract of the talk presented by the author at the Mathematical Logic Group Meeting in Japan held in Echigo-Yuzawa on January 9-11, 2002.

Definition 2 *A game Γ is an* **update game** *if $W = S \bigcup A$ and $\Omega = \{S \bigcup A\}$. If Survivor wins this game, we then call the game an* **update network**.

It turns out that update games can be decided in a polynomial time as shown in [1]:

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

The proof of this theorem is based on finding certain natural structural properties of update networks. See [1] for details.

In [2] Theorem 1 has been generalized as follows. We need a definition.

Definition 3 *A game Γ is a* **relaxed update game** *if Ω consists of pairwise disjoint subsets of W, that is for distinct $V, U \in \Omega$ we have $V \cap U = \emptyset$. If Survivor wins this game, we then call the game a* **relaxed update network**.

Using certain forcing techniques it is possible to prove the following theorem [2]:

Theorem 2 *There exists an algorithm that given a game decides in $O(|V|^3)$ time whether or not the game is a relaxed update network.*

The proof of this theorem does not use the techniques provided in the proof of Theorem 1. Instead, based on the notion of rank defined in Gurevich-Harrington [3], certain natural concepts (such as forcing) are introduced for the investigation of relaxed update games.

Recently we have been able to prove the following theorem whose proof consists of generalizing the ideas exploited in Theorem 1 and Theorem 2. We first give the following definition.

Definition 4 *We say that the game Γ is* **fully separated** *if $W = S \bigcup A$ and for each $V \in \Omega$ there is a w such that $w \in V$ but w belongs to no other V' from Ω.*

Here is our theorem.

Theorem 3 *(Ishihara, Khoussainov: 2001) There exists an algorithm that given a fully separated game Γ decides in $O(|V|^3)$ time whether or not Survivor wins Γ.*

The proof uses techniques developed in both [1] and [2]. A technical notion that is used in the proof is the notion of S-closedness which is defined in [2]:

Definition 5 *A winning set $V \in \Omega$ is* **S-closed** *if it satisfies the following conditions:*

1. *For each $s \in V \cap S$ there is an $a \in V \cap A$ such that $(s, a) \in E$.*

2. *For each $a \in V \cap A$ and all s such that $(a, s) \in E$ we have $s \in V$.*

Thus, informally this definition tells us that Survivor is able to always keep plays inside V no matter what the oppenent does. The proof of the theorem uses the idea of reducing the game Γ to subgames of the type $(V, \{V\})$ for S-closed $V \in \Omega$. These are update games, and hence Theorem 1 can be applied in the line of the proof.

Now we would like to 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. More formally, this can be established as follows as it is done in [4]. Given a system $(S \bigcup A, E)$ we can 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 $F\phi$ also propositions. Now semantics for these propsitions are runs of the system $(S \bigcup A, E)$. Let π be a run and ϕ be formula. One now can define what it means ϕ to be true on π. Thinking of G as "globally" or "always", and of F as "future", we can actually 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 that Survivor must visit every node infinitely often. This is in fact a specification of an update network game in terms of temporal logic. Thus, one can study the following natural questions:

1. (Model checking) Given a (fixed) 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 (fixed) 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 more details on topics of research in the line of verification and specifications of systems by means of temporal logic see Vardi [8].

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] 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.