Timed games with branching-time winning conditions

Aniello Murano
Dipartimento di Informatica ed Applicazioni,
Università degli Studi di Salerno,
Via S. Allende, 84081 Baronissi (SA)
{murano}@dia.unisa.it

The theory of games, traditionally related to the economic-theoretic environment (see for instance [19]), has been attracting the interest of many researchers in both computer science and control theory. The notion of a game naturally arises in the verification of reactive systems and program synthesis [4, 21]. In the compositional approach, a reactive system is seen as a set of interacting components, each of them is modelled as an *open system* (that is, a system whose behaviour depends on the current state as well as the behaviour of the environment in which it is embedded). The interaction between a component and the rest of the system, or the interaction between a controller and the related plant, both can be modeled as a two-player game. For verifying *closed systems*, where a system behaviour is completely determined by the current state, model-checking is a very successful technique [6, 7, 8]. The equivalent of model-checking for open systems is checking for the existence of a winning strategy, according to a given winning condition, in a two-player game [15].

In this paper we focus on the verification of open real-time systems, modeled as nondeterministic *timed automata* [2]: finite automata augmented with a finite set of real-valued *clocks*. The transitions of a timed automaton are enabled according to the current state, that is, the current *location* and the current clock values. In a transition, clocks can be instantaneously reset. The value of a clock is exactly the time elapsed since the last time it was reset. A clock constraint (*guard*) is associated to each transition with the meaning that a transition can be taken only if the associated guard is enabled. A clock constraint (*invariant*) can also be associated to each location with the meaning that the automaton can stay in a location as long as the corresponding invariant remains true.

When interpreting a nondeterministic timed automaton as a *(timed) game graph*, we capture the choices of the *protagonist* by the symbols associated with the transitions and nondeterminism is used to model the possible choices of the *antagonist*. To model the case that the protagonist stays *idle* and the antagonist is moving, we use a special symbol denoted by ξ. The case that both players stay idle is captured by letting time elapse in a location. A *play* of a timed game is thus constructed in the following way. At each time, a player declares the time it will wait idling, along with its next choice. At the time one of the players or both move, both players are allowed to redeclare their next move and the time they will issue it. Technically, a play is a run of the automaton modeling the game. A game is given by a game graph along with a winning condition that establishes which computations are winning for the protagonist. In this paper we solve Ctl and Tctl games, that is, timed games with a winning condition given, respectively, by a formula of the branching-time temporal logic Ctl [6], or its real-time extension Tctl [1].

Ctl Timed Games. A way to decide discrete games is to reduce them to the emptiness problem for tree automata (see, for example [22]). Here, we extend the automata-theoretic approach to solve Ctl timed games. Given a timed game (G, φ), where G is a game graph and φ a Ctl formula, we construct a tree automaton A_G that accepts all the ω–trees corresponding to a strategy of the protagonist. We construct this automaton exploiting the clock region relation [2]. Then, we construct another tree automaton A_φ accepting all the trees satisfying the property expressed by φ.

Since strategies of the protagonist correspond to trees accepted by A_G, to construct A_φ we only need to capture models with arity bounded above by the arity of A_G (i.e. the maximum branching degree of A_G transitions). Thus, there exists a winning strategy of the protagonist in the game (G, φ) if and only if the intersection between the languages accepted by A_G and A_φ is not empty. Since the size of A_φ is exponential in the size of φ [25], checking the emptiness for a Büchi tree automaton is polynomial in the number of its states [26], and the intersection of the languages accepted by A_G and A_φ is accepted by a Büchi tree automaton whose number of states is exponential in the number of states of A_G and A_φ [23], we obtain an algorithm that takes exponential time. Since timed reachability games are known to be Exptime-hard [14], we have that our result is complete.

Tctl Timed Games. The syntax of Tctl is given by augmenting the temporal operators of Ctl (except for the "next", which is discarded since it does not have a clear meaning in a dense time domain) with a timing constraint of type $\approx c$, where $\approx$ is one among $<$, $\leq$, $>$, $\geq$, and $=$, and c is a natural number. While Ctl allows to express qualitative temporal requirements on the occurrence of events, with Tctl it is also possible to express quantitative requirements. For example, using Ctl we can express assertion like "p_1 is true until p_2 is true", but we cannot express assertion like "p_1 is true until p_2 is true within time 2", which can be expressed in Tctl. In Tctl timed games, thus, we allow dense real-time in both the game graph and the winning condition.

Model-checking of Tctl-formulas is Pspace-complete while checking for their satisfiability is undecidable [1]. In a recent paper, the satisfiability of Tctl is proved to be Exptime-complete if equality is not allowed in timing constraints of the formulas [16]. Here we obtain the same complexity/decidability result for games. We prove that Tctl games are Exptime-complete, if equality is not allowed in timing constraints. Our approach to solve Tctl games relies on a reduction to the emptiness problem of timed Büchi tree automata. Given a timed automaton G and a Tctl–formula φ, we solve the game (G, φ) by constructing two timed Büchi tree automata: A_G and A_φ. The automaton A_G accepts, for each strategy $\mathcal{F}$ of the protagonist, a timed ω-tree which is "embedded" into $\mathcal{F}$. The automaton A_φ accepts instead all the models of φ with arity bounded above by the arity of G. By the known results on timed Büchi tree automata [17], the intersection of A_G and A_φ is still a timed Büchi tree automaton, and accepts all the trees that satisfy φ and are embedded into a strategy. Thus, a timed game G admits a winning strategy with respect to a formula φ if and only if the language accepted by the intersection automaton is not empty. Denoting the product of the sizes of φ and G by K, we have that the number of states of the intersection automaton is exponential in K while the size of its clock constraints is linear in K. Since checking the emptiness for a timed Büchi tree automaton is polynomial in the number of states and exponential in clock size [17], we obtain an algorithm that takes exponential time. The result we obtain is tight, in the sense that Tctl games are EXPTIME-complete (hardness is a consequence of the EXPTIME-hardness of the reachability timed games [14]).

Finally, by allowing equality in the timing constraints, TCTL games turns out to be undecidable. It is possible to prove this result by reducing the satisfiability problem of TCTL–formulas, which is known to be undecidable [1] to our problem. Another approach to prove this result is to introduce the *module-checking* problem of TCTL–formulas in dense real-time, and prove that this problem is undecidable when equality is allowed in the timing constraints [18]. Here, the proof is obtained by reducing the problem of deciding whether a nondeterministic two-counter machine has a recurring computation (this problem is known to be Σ_1^1-*hard* [12]).

Games have been studied in the context of discrete, timed, and hybrid systems (see [5, 13, 21, 24]. In literature, different formulations of games with winning conditions expressed by temporal logics have been considered. In [3], alternating temporal logic is introduced. In [15] the model checking of open systems (*module checking*) is studied. This problem turns out to be EXPTIME-compete for specification given by CTL-formulas and 2EXPTIME-complete for specification given by CTL*-formulas. Rectangular hybrid games with winning conditions expressed by LTL formulas were solved in [13]. In [9], the controller synthesis of timed automata with linear-time specifications given also as timed automata is addressed. Finally, in [10] an automata-theoretic approach to solve LTL timed games is also considered.

Acknowledgments. This abstract contains results obtained during my Ph.D. studies and have been obtained in collaboration with other researchers that I would like to thank: Moshe Vardi, Marherita Napoli, Salvatore La Torre, Marco Faella.

References

[1] R. Alur, C. Courcoubetis, and D.L. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2 – 34, 1993.

[2] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183 – 235, 1994.

[3] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proc. of the 38th IEEE Symposium on Foundations of Computer Science*, pages 100 – 109, 1997.

[4] M. Abadi, L. Lamport, and P. Wolper. Realizable and unrealizable specifications of reactive systems. In *Proc. of the 16th Intern. Colloquium on Automata, Languages and Programming, ICALP'89*, LNCS 372, pages 1–17, 1989.

[5] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *Proc. IFAC Symposium on System Structure and Control*, pages 469 – 474. Elsevier, 1998.

[6] E.A. Emerson and E.M. Clarke. Using branching-time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2:241 – 266, 1982.

[7] E. M. Clarke , E.A. Emerson , A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal-logic specifications. *ACM Transactions on Programming Languages and Systems*, v. 8, 2 : 244-263, 1986

[8] E. M. Clarke , R. P. Kurshan. Computer-aided verification. *IEEE Spectrum*, v. 33, 6: 61–67, 1996

[9] D. D'Souza, P. Madhusudan. Timed control synthesis for external specifications. In *Proc. of the 19th Annual Symposium on Theoretical Aspects of Computer Science*, (STACS'02), LNCS 2285: 571 – 582, 2002.

[10] M. Faella, S. La Torre, and A. Murano. Automata-theoretic decision of timed games. In *Proc. of the 3rd International Workshop on Verification, Model Checking, and Abstract Interpretation, VMCAI 2002*, LNCS 2294, pages 94 – 108. Springer-Verlag, 2000.

[11] M. Faella, S. La Torre and A. Murano. Dense Real-Time Games. *Proc. of the Seventeenth Annual IEEE Symposium on Logic in Computer Science (LICS02)*, pages 167–176, 2002.

[12] D. Harel, A. Pnueli, J. Stavi. Propositional Dynamic Logic of Regular Programs", In *Journal of Computer and System Sciences*, 26: 222-243, 1983

[13] T.A. Henzinger, B. Horowitz, and R. Majumdar. Rectangular hybrid games. In *Proc. of the 10th International Conference on Concurrency Theory, CONCUR'99*, LNCS 1664, pages 320 – 335, 1999.

[14] T. Henzinger and P. Kopke. Discrete-time control for rectangular hybrid automata. *Theoretical Computer Science*, 221(1–2):369–392, 1999

[15] O. Kupferman and M.Y. Vardi. Module checking. In *Computer Aided Verification, Proc. Eighth Int. Workshop*, LNCS 1102, pages 75 – 86. Springer-Verlag, 1996.

[16] S. La Torre and M. Napoli. A decidable dense branching-time temporal logic. In *Proc. of the 20th Conference on the Foundations of Software Technology and Theoretical Computer Science, FSTTCS'00*, LNCS 1974, pages 139 – 150. Springer-Verlag, 2000.

[17] S. La Torre and M. Napoli. Timed tree automata with an application to temporal logic. *Acta Informatica*, 38(2):89–116, 2001.

[18] A. Murano and M. Vardi. *Real-time module checking.* Work in progress.

[19] J. Von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press, 1944.

[20] A. Pnueli. The temporal logic of programs. In *Proc. of the 18th IEEE Symposium on Foundations of Computer Science*, pages 46 – 77, 1977.

[21] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. of the 16th ACM Symposium on Principles of Programming Languages*, pages 179 – 190, 1989.

[22] M.O. Rabin. Automata on infinite objects and Church's problem. *Trans. Amer. Math. Soc.*, 1972.

[23] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 133 – 191. Elsevier Science Publishers, 1990.

[24] W. Thomas. On the synthesis of strategies in infinite games. In Ernst W. Mayr and Claude Puech, editors, *12th Annual Symposium on Theoretical Aspects of Computer Science, STACS'95*, LNCS 900, pages 1 – 13. Springer-Verlag, 1995.

[25] M.Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 32:182 – 211, 1986.

[26] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115:1 – 37, 1994.