Games, Time, and Probability: Graph Models for System Design and Analysis*

Thomas A. Henzinger[1]

EPFL, Switzerland
tah@epfl.ch

Abstract. Digital technology is increasingly deployed in safety-critical situations. This calls for systematic design and verification methodologies that can cope with three major sources of system complexity: concurrency, real time, and uncertainty. We advocate a two-step process: formal modeling followed by algorithmic analysis (or, "model building" followed by "model checking"). We model the concurrent components of a reactive system as potential collaborators or adversaries in a multi-player game with temporal objectives, such as system safety. The real-time aspect of embedded systems requires models that combine discrete state transitions and continuous state evolutions. Uncertainty in the environment is naturally modeled by probabilistic state changes. As a result, we obtain three orthogonal extensions of the basic state-transition graph model for reactive systems —game graphs, timed graphs, and stochastic graphs— as well as combinations thereof. In this short text, we provide a uniform exposition of the underlying definitions. For verification algorithms, we refer the reader to the literature.

1 Graph Models

In graph models of reactive systems, the vertices of a graph represent system states, and the edges represent state transitions. Formally, a *graph model* $G = (Q, \Gamma, \delta)$ consists of a set Q of *states*, a set Γ of *choices*, and a *transition* function $\delta\colon Q \times \Gamma \to Q$. Given a state $q \in Q$, each choice $\gamma \in \Gamma$ determines a successor state $\delta(q, \gamma) \in Q$. The choices represent nondeterminism in state transitions. In the following, we fix the graph model G. A *Boolean valuation* $v\colon Q \to \{0, 1\}$ assigns to each state a truth value; a *Markovian valuation* $p\colon Q \to [0, 1]$ assigns to each state a real value denoting a probability.

If a reactive system is modeled as a graph, then paths in the graph represent possible behaviors of the system. A finite or infinite sequence of states is called a *trace*. We write $q_{0..i}$ short for the finite trace $q_0, q_1, \ldots, q_i$. Traces are generated by schedulers. A *scheduler* $\sigma\colon Q^+ \to \Gamma$ maps every nonempty finite trace, representing the past behavior of the system, to a choice, which determines the next state of the system. We write Σ for the set of schedulers. Given a scheduler

* This research was supported in part by the Swiss National Science Foundation, and by the NSF ITR grant CCR-0225610.

$\sigma \in \Sigma$ and a start state $q \in Q$, the infinite trace $\sigma^*(q) = q_0, q_1, q_2, \ldots$ *generated* by σ from q is defined by (1) $q_0 = q$ and (2) $q_{i+1} = \delta(q_i, \sigma(q_{0..i}))$ for all $i \geq 0$.

Consider the Cantor topology on infinite traces, whose basic open sets have the form $q_{0..i}Q^\omega$; that is, each basic open set contains all infinite extensions of some finite trace $q_{0..i} \in Q^*$. A *property* $\Phi \subseteq Q^\omega$ is a Borel set of infinite traces. We write $\mathcal{F}$ for the set of properties. In our usage (which is dual to the standard usage), a reactive system satisfies the property Φ iff all possible behaviors of the system lie outside Φ; in other words, the set Φ specifies the *un*desirable traces. An example of a safety property is $Q^*q_{err}Q^\omega$, where $q_{err} \in Q$ represents an error state: this property is violated iff some scheduler causes the error state to be visited. An example of a liveness property is $Q^*q_{req}(Q \backslash \{q_{grnt}\})^\omega$, where q_{req} represents a state where a request is made, and q_{grnt} represents a state where the request is granted: this property is violated iff some scheduler causes a request not to be followed by a grant. Formally, given a start state $q \in Q$ and a scheduler $\sigma \in \Sigma$, the *payoff* function $P(q, \sigma)$: $\mathcal{F} \to \{0, 1\}$ assigns to each property a truth value, namely, $P(q, \sigma)(\Phi) = 1$ iff $\sigma^*(q) \in \Phi$. For a property $\Phi \in \mathcal{F}$, the *value* function $\mathrm{val}(\Phi)$: $Q \to \{0, 1\}$ is the Boolean valuation defined by $\mathrm{val}(\Phi)(q) = \sup_{\sigma \in \Sigma} P(q, \sigma)(\Phi)$; that is, the value of Φ at a state q is 1 if there exists a scheduler that generates from q a trace in Φ, and otherwise the value of Φ at q is 0. If $\mathrm{val}(\Phi)(q) = 0$, then we say that the state q *satisfies* the property Φ; if $\mathrm{val}(\Phi)(q) = 1$, then q *violates* Φ. In the latter case, the witnessing scheduler σ is called a *counterexample* to Φ at q. If the graph model is not fixed, then we write $\mathrm{val}^G(\Phi)$ for the value function of a property Φ on a graph model G.

A scheduler σ is *memoryfree* if $q_i = q'_j$ implies $\sigma(q_{0..i}) = \sigma(q'_{0..j})$; that is, each choice of the scheduler depends only on the current state, and not on the past behavior of the system. It should be noted that counterexamples may need memory. For instance, the state q_0 of the graph model with $Q = \{q_0, q_1, q_2\}$, $\Gamma = \{1, 2\}$, and $\delta = \{(q_0, 1, q_1), (q_0, 2, q_2), (q_1, \cdot, q_0), (q_2, \cdot, q_0)\}$ violates the generalized Büchi property $(Q^*q_1Q^*q_2)^\omega$, but there is no memoryfree counterexample.

Graph models are often very large (even infinite) objects, and properties are usually infinite sets. In practice, graph models and properties are specified by succinct, finite descriptions given in a particular syntax. For example, the terms of a process algebra, or a class of programs written in a programming language, may be used to specify graph models; the formulae of a temporal logic, or a class of finite automata over infinite words, may be used to specify properties. Let L_1 be a *system description language*, whose expressions specify graph models, and let L_2 be a *property description language*, whose expressions specify properties. For an expression $e \in L$, we write $[\![e]\!]$ for the graph model (resp. property) specified by e. The *model-checking* problem for L_1 and L_2 asks, given two expressions $e_1 \in L_1$ and $e_2 \in L_2$, to compute the Boolean valuation $\mathrm{val}^{[\![e_1]\!]}([\![e_2]\!])$. Moreover, for each violating state q of the graph model $[\![e_1]\!]$, a counterexample to the property $[\![e_2]\!]$ at q is desired.

For algorithms that solve important instances of the model-checking problem and compute counterexamples, start with [CGP99].

2 Game Models

Game models of reactive systems are extensions of graph models, where the players in the game represent different components of the system (or the environment). Formally, a two-player *game model* $(Q, \Gamma_1, \Gamma_2, \delta)$ consists of a set Q of states, two sets Γ_1 and Γ_2 of choices for the two players, and a transition function $\delta\colon Q \times \Gamma_1 \times \Gamma_2 \to Q$. Given a state $q \in Q$, if player 1 chooses $\gamma_1 \in \Gamma_1$ and player 2 chooses $\gamma_2 \in \Gamma_2$, then the successor state is $\delta(q, \gamma_1, \gamma_2) \in Q$. A generalization to more than two players is possible, but not discussed here. These graph games are called *concurrent* [AHK02]; the special case of *turn-based* games occurs if every state $q \in Q$ is either a player-1 state or a player-2 state. The state q is a *player-1 state* if for all choices $\gamma_1 \in \Gamma_1$ and $\gamma_2, \gamma_2' \in \Gamma_2$, we have $\delta(q, \gamma_1, \gamma_2) = \delta(q, \gamma_1, \gamma_2')$; that is, player 1 determines the successor state. The *player-2 states* are defined symmetrically.

In games, schedulers are called strategies, properties are called objectives, and counterexamples are called winning strategies. For $k \in \{1, 2\}$, a *player-k strategy* $\sigma_k\colon Q^+ \to \Gamma_k$ maps every nonempty finite trace to a choice for player k. We write Σ_k for the set of player-k strategies. Given a start state $q \in Q$ and strategies $\sigma_1 \in \Sigma_1$ and $\sigma_2 \in \Sigma_2$ for the two players, the generated trace $(\sigma_1, \sigma_2)^*(q) = q_0, q_1, q_2, \ldots$ is defined by (1) $q_0 = q$ and (2) $q_{i+1} = \delta(q_i, \sigma_1(q_{0..i}), \sigma_2(q_{0..i}))$ for all $i \geq 0$. The *payoff* function $P(q, \sigma_1, \sigma_2)\colon \mathcal{F} \to \{0, 1\}$ is defined by $P(q, \sigma_1, \sigma_2)(\Phi) = 1$ iff $(\sigma_1, \sigma_2)^*(q) \in \Phi$. For an objective $\Phi \in \mathcal{F}$, the *player-1 value* function $\mathrm{val}_1(\Phi)\colon Q \to \{0, 1\}$ is the Boolean valuation defined by $\mathrm{val}_1(\Phi)(q) = \sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} P(q, \sigma_1, \sigma_2)(\Phi)$; that is, the value of Φ at a state q is 1 iff there exists a player-1 strategy such that for all player-2 strategies, the trace generated from q lies in Φ. If $\mathrm{val}_1(\Phi)(q) = 1$, then the state q is *player-1 winning* for the objective Φ; in this case, the witnessing player-1 strategy σ_1 is called a *winning strategy* for Φ at q.

The *player-2 value* function $\mathrm{val}_2(\Phi)$ is defined symmetrically, by $\mathrm{val}_2(\Phi)(q) = \sup_{\sigma_2 \in \Sigma_2} \inf_{\sigma_1 \in \Sigma_1} P(q, \sigma_1, \sigma_2)(\Phi)$. The game is *zero-sum* if the objectives of the two players are complementary. Turn-based graph games with zero-sum Borel objectives are determined: each state of a turn-based game model is either player-1 winning for the objective Φ, or player-2 winning for the complementary objective $Q^\omega \backslash \Phi$ [Mar75].

Let L_1 be a system description language whose expressions specify game models, and let L_2 be an objective description language. The *game-solving* problem for L_1 and L_2 asks, given two expressions $e_1 \in L_1$ and $e_2 \in L_2$, to compute the Boolean valuation $\mathrm{val}_1^{[\![e_1]\!]}([\![e_2]\!])$. Moreover, for each player-1 winning state q of the game model $[\![e_1]\!]$, a winning strategy for the objective $[\![e_1]\!]$ at q is desired.

For algorithms that solve turn-based games and compute winning strategies, start with [Tho95,GTW03]; for concurrent games, see [AHK02].

3 Stochastic Models

Stochastic models allow probabilistic transition functions and probabilistic schedulers. We write $\mathcal{D}(S)$ for the probability distributions on a set S. A *stochastic*

graph model (a.k.a. *Markov decision process*) (Q, Γ, δ) consists of a set Q of states, a set Γ of choices, and a transition function $\delta\colon Q \times \Gamma \to \mathcal{D}(Q)$ which assigns to each state $q \in Q$ and choice $\gamma \in \Gamma$ a probability distribution $\delta(q,\gamma)\colon Q \to [0,1]$ on the successor state.

A *probabilistic scheduler* $\sigma\colon Q^+ \to \mathcal{D}(\Gamma)$ maps every nonempty finite trace to a probability distribution on the next choice. We write Σ^p for the set of probabilistic schedulers. To distinguish the nonprobabilistic schedulers in Σ, we call them *pure*. The outcome of applying a probabilistic scheduler to a state of a graph model, or of applying a pure scheduler to a state of a stochastic model, is not a single infinite trace, but a probability distribution on the generated infinite trace. Formally, given a start state $q \in Q$ of a stochastic model and a probabilistic scheduler $\sigma \in \Sigma^p$, the *payoff* function $P(q,\sigma)\colon \mathcal{F} \to [0,1]$ is a probability measure that assigns to each property $\Phi \in \mathcal{F}$ the probability that the generated infinite trace lies in Φ. In order to define the function $P(q,\sigma)$, it suffices to define its value on the basic open sets. This is done inductively: (1) $P(q,\sigma)(q_0 Q^\omega)$ is 1 if $q_0 = q$, and 0 otherwise; (2) $P(q,\sigma)(q_{0..i} q_{i+1} Q^\omega) = P(q,\sigma)(q_{0..i} Q^\omega) \cdot \sum_{\gamma \in \Gamma} (\sigma(q_{0..i})(\gamma) \cdot \delta(q_i,\gamma)(q_{i+1}))$. For a property $\Phi \in \mathcal{F}$, the *value* function $\mathrm{val}(\Phi)\colon Q \to [0,1]$ is the Markovian (rather than Boolean) valuation defined by $\mathrm{val}(\Phi)(q) = \sup_{\sigma \in \Sigma^p} P(q,\sigma)(\Phi)$; that is, the value of Φ at a state q is the maximal probability with which any scheduler can ensure that an infinite trace generated from q lies in Φ. The witnessing scheduler σ is called an *optimal* scheduler for Φ at q.

A *stochastic game model* $(Q, \Gamma_1, \Gamma_2, \delta)$ is a game model with a probabilistic transition function $\delta\colon Q \times \Gamma_1 \times \Gamma_2 \to \mathcal{D}(Q)$. In such games, the two players are allowed probabilistic strategies $\sigma_1 \in \Sigma_1^p$ and $\sigma_2 \in \Sigma_2^p$. In the definition of payoffs, let $P(q,\sigma_1,\sigma_2)(q_{0..i} q_{i+1} Q^\omega) = P(q,\sigma_1,\sigma_2)(q_{0..i} Q^\omega) \cdot \sum_{\gamma_1 \in \Gamma_1} \sum_{\gamma_2 \in \Gamma_2} (\sigma_1(q_{0..i})(\gamma_1) \cdot \sigma_2(q_{0..i})(\gamma_2) \cdot \delta(q_i,\gamma_1,\gamma_2)(q_{i+1}))$. For an objective $\Phi \in \mathcal{F}$, the *player-1 value* function $\mathrm{val}_1(\Phi)\colon Q \to \{0,1\}$ is the Markovian valuation defined by $\mathrm{val}_1(\Phi)(q) = \sup_{\sigma_1 \in \Sigma_1^p} \inf_{\sigma_2 \in \Sigma_2^p} P(q,\sigma_1,\sigma_2)(\Phi)$; that is, the value of Φ at a state q is the maximal probability with which player 1 can ensure, against any player-2 strategy, that an infinite trace generated from q lies in Φ.

Stochastic graph games with zero-sum Borel objectives are determined: for all objectives $\Phi \in \mathcal{F}$ and all states $q \in Q$, we have $\mathrm{val}_1(\Phi)(q) + \mathrm{val}_2(Q^\omega \backslash \Phi)(q) = 1$ [Mar98]. It should be noted that probabilistic strategies are required for determinacy: even for nonstochastic game models, optimal strategies may need to be probabilistic. For instance, the objective $Q^* q_1 Q^\omega$ has the player-1 value 1 at the state q_0 of the game model with $Q = \{q_0, q_1\}$, $\Gamma_1 = \{1,2\}$, $\Gamma_2 = \{1,2\}$, and $\delta = \{(q_0,1,2,q_0), (q_0,2,1,q_0), (q_0,1,1,q_1), (q_0,2,2,q_1), (q_1,\cdot,\cdot,q_1)\}$. To see this, observe that any memoryfree player-1 strategy σ_1 with $\sigma_1(q_0)(1) > 0$ and $\sigma_1(q_0)(2) > 0$ is optimal. However, there is no optimal pure strategy for player 1.

Let L_1 be a system description language whose expressions specify stochastic graph or game models, and let L_2 be a property description language. The *quantitative-solution* problem for L_1 and L_2 asks, given two expressions $e_1 \in L_1$ and $e_2 \in L_2$ and a desired degree of numerical precision, to compute the Markovian valuation $\mathrm{val}^{[\![e_1]\!]}([\![e_2]\!])$ (resp. $\mathrm{val}_1^{[\![e_1]\!]}([\![e_2]\!])$). For a Markovian valuation

$v\colon Q \to [0,1]$, let $\lfloor v \rfloor\colon Q \to \{0,1\}$ be the Boolean valuation defined by $\lfloor v \rfloor(q) = 1$ if $v(q) = 1$, and $\lfloor v \rfloor(q) = 0$ if $v(q) < 1$. The *qualitative-solution* problem asks to compute the Boolean valuation $\lfloor v \rfloor$, where v is the answer to the quantitative problem. Many qualitative problems can be solved in a 3-valued probability model, which distinguishes only the three probability values 0, (0,1), and 1.

For an introduction to stochastic graph games, start with [FV97]. For algorithms that solve stochastic turn-based games (including Markov decision processes) and compute optimal strategies, see [Con92,Con93,CY95,dAlf97] and, more recently, [CJH03,CJH04,CdAH05,CH06a,CH06b]. For the qualitative solution of concurrent games, see [dAHK98,dAH00,CdAH06b]; for the quantitative solution of concurrent games, see [dAM04,CdAH06a].

4 Timed Models

We model time by the real numbers, although other choices are possible. In a timed graph model, each state $q \in Q$ is annotated with a real-valued *timeout* $u(q) \in \mathbb{R}_{\geq 0} \cup \{\infty\}$, which provides an upper bound on the amount of time that may be spent in the state. A *timed graph model* (Q, Γ, u, δ) consists of a set Q of states, a set Γ of choices, a timeout function $u\colon Q \to \mathbb{R}_{\geq 0}$, and a transition function $\delta\colon Q \times \mathbb{R}_{\geq 0} \times \Gamma \to Q$. If in state $q \in Q$, choice $\gamma \in \Gamma$ is scheduled after a time delay of $t \leq u(q)$, then the successor state is $\delta(q, t, \gamma)$.

The behaviors of a real-time system are represented as timed traces. A *timed trace* $\tau = q_0, t_0, q_1, t_1, q_2, \ldots$ is a finite or infinite sequence of alternating states $q_i \in Q$ and time delays $t_i \in \mathbb{R}_{\geq 0}$. The timed trace τ *diverges* if $\sum_{i \geq 0} t_i = \infty$. Let $T = Q \times \mathbb{R}_{\geq 0}$. We write $T^{div} \subseteq T^\omega$ for the set of divergent timed traces. A *timed scheduler* $\sigma\colon T^*Q \to \mathbb{R}_{\geq 0} \times \Gamma$ maps every finite timed trace that ends in a state, to both a time delay and a choice. We require that if $\sigma(\tau q) = (t, \cdot)$, then $t \leq u(q)$. We write Σ^t for the set of timed schedulers. Given a timed scheduler $\sigma \in \Sigma^t$ and a start state $q \in Q$, the infinite timed trace $\sigma^*(q) = q_0, t_0, q_1, t_1, q_2, \ldots$ *generated* by σ from q is defined by (1) $q_0 = q$ and (2) for all $i \geq 0$, if $\sigma(q_0, \ldots, q_i) = (t, \gamma)$, then $t_i = t$ and $q_{i+1} = \delta(q_i, t, \gamma)$.

A *timed property* $\Phi \subseteq T^\omega$ is a set of infinite timed traces. We write $\mathcal{F}^t$ for the set of timed properties. Given a start state $q \in Q$ and a timed scheduler $\sigma \in \Sigma^t$, the *payoff* function $P(q, \sigma)\colon \mathcal{F}^t \to \{0,1\}$ is defined by $P(q,\sigma)(\Phi) = 1$ iff $\sigma^*(q) \in (\Phi \cap T^{div})$. Note that for payoff 1 we require that the generated trace diverges; this is because a physically realizable scheduler must not enforce a property by preventing time from diverging. For a timed property $\Phi \in \mathcal{F}^t$, the *value* function $\mathrm{val}(\Phi)\colon Q \to \{0,1\}$ is defined as usual, by $\mathrm{val}(\Phi)(q) = \sup_{\sigma \in \Sigma^t} P(q,\sigma)(\Phi)$.

Some timed graph models are not well-formed, in the sense that they contain states from which no scheduler can ensure the divergence of time. Formally, a timed graph model is *nonzeno* if $\mathrm{val}(T^\omega)(q) = 1$ for all states $q \in Q$. Thus, checking nonzenoness is a special instance of the model-checking problem for timed graph models. Only nonzeno models represent physical systems.

A *timed game model* $(Q, \Gamma_1, \Gamma_2, u, \delta)$ has a timeout function $u\colon Q \to \mathbb{R}_{\geq 0}$ and a transition function of type $\delta\colon Q \times \mathbb{R}_{\geq 0} \times (\Gamma_1 \cup \Gamma_2) \to Q$. If in state $q \in Q$,

player 1 schedules choice $\gamma_1 \in \Gamma_1$ after time delay $t_1 \leq u(q)$, and player 2 schedules choice $\gamma_2 \in \Gamma_2$ after time delay $t_2 \leq u(q)$, then the successor state is $\delta(q, t_1, \gamma_1)$ if $t_1 < t_2$; otherwise the successor state is $\delta(q, t_2, \gamma_2)$. In other words, both players propose time delays, and the proposal for the shorter delay "wins." Note that the definition breaks ties ($t_1 = t_2$) arbitrarily in favor of player 2; a symmetric resolution, such as defining concurrent outcomes for simultaneous choices, would be more desirable, but the issue altogether disappears in a suitable stochastic model, where the time delays are chosen at random. However, we do not define such a model here.

We wish to restrict both players to use physically realizable strategies, which do not prevent time from diverging. This restriction is surprisingly subtle, because any one player cannot ensure the divergence of time if the other player chooses to block the advance of time [dAFH$^+$03]. In order to decide which player is to "blame" for a convergent trace, we need to slightly generalize the notion of timed trace: we now consider a *timed trace* $\tau = q_0, t_0, b_0, q_1, t_1, b_1, q_2, \ldots$ to be a sequence of alternating states $q_i \in Q$, time delays $t_i \in \mathbb{R}_{\geq 0}$, and player names $b_i \in \{1, 2\}$. The bit $b_i = k$ indicates that the i-th time delay t_i is chosen by player k. Player k cannot be held responsible for causing the convergence of τ if $b_i = k$ for only finitely many $i \geq 0$. In the following, let $T = Q \times \mathbb{R}_{\geq 0} \times \{1, 2\}$. For $k \in \{1, 2\}$, let $T_k^{noblame}$ be the set of timed traces $\tau = q_0, t_0, b_0, q_1, \ldots$ such that (1) $\sum_{i \geq 0} t_i < \infty$ (that is, τ converges), and (2) there exists a $j \geq 0$ such that $b_i \neq k$ for all $i \geq j$ (that is, player k is not to blame for the convergence of τ). Given a start state $q \in Q$ and two timed schedulers $\sigma_1 \in \Sigma_1^t$ and $\sigma_2 \in \Sigma_2^t$, the *generated* timed trace $(\sigma_1, \sigma_2)^*(q) = q_0, t_0, b_0, q_1, \ldots$ is defined by (1) $q_0 = q$ and (2) for all $i \geq 0$, if $\sigma_1(q_0, \ldots, q_i) = (t_1, \gamma_1)$ and $\sigma_2(q_0, \ldots, q_i) = (t_2, \gamma_2)$, then (2a) if $t_1 < t_2$, then $t_i = t_1$ and $b_i = 1$ and $q_{i+1} = \delta(q_i, t_1, \gamma_1)$; and (2b) if $t_2 \leq t_1$, then $t_i = t_2$ and $b_i = 2$ and $q_{i+1} = \delta(q_i, t_2, \gamma_2)$.

The restriction to strategies that do not prevent time from diverging leads to nonzero-sum games: for player $k \in \{1, 2\}$, start state $q \in Q$, and strategies $\sigma_1 \in \Sigma_1$ and $\sigma_2 \in \Sigma_2$, the *player-k payoff* function $P_k(q, \sigma_1, \sigma_2)$: $\mathcal{F}^t \to \{0, 1\}$ is defined by $P_k(q, \sigma_1, \sigma_2)(\Phi) = 1$ iff $(\sigma_1, \sigma_2)^*(q) \in ((\Phi \cap T^{div}) \cup T_k^{noblame})$. Note that player k receives payoff 1 for objective Φ on a convergent trace if she cannot be held responsible for the convergence of time, irrespective of whether or not the trace lies in Φ. For an objective $\Phi \in \mathcal{F}^t$, the *player-1 values* are defined by $\mathrm{val}_1(\Phi)(q) = \sup_{\sigma_1 \in \Sigma_1^t} \inf_{\sigma_2 \in \Sigma_2^t} P_1(q, \sigma_1, \sigma_2)(\Phi)$; that is, player 1 must ensure that either the generated trace diverges and lies in Φ, or that she is not held responsible for the convergence of time. The *player-2 values* are defined symmetrically. Even if both players have complementary objectives Φ and $T^\omega \backslash \Phi$, both may receive payoff 0 on a convergent trace; indeed, there are states q with both $\mathrm{val}_1(\Phi)(q) = 0$ and $\mathrm{val}_2(T^\omega \backslash \Phi)(q) = 0$ [dAFH$^+$03]. However, for a timed game to be well-formed, each player must be able to win for the trivial objective $\Phi = T^\omega$: a timed game model is *nonzeno* if both $\mathrm{val}_1(T^\omega)(q) = 1$ and $\mathrm{val}_2(T^\omega)(q) = 1$ for all states $q \in Q$ [HP06].

A popular system description language whose expressions specify timed graph or timed game models is the formalism of *timed automata* [AD94]. This language

has the advantage that, even over infinite state spaces, important model-checking and game-solving problems can be decided. For model checking timed graphs that are specified by timed automata, see [ACD93,HNSY94]; for the solution of timed games that are specified by timed automata, see [WH91,MPS95,dAFH$^+$03].

References

[ACD93] R. Alur, C. Courcoubetis, and D.L. Dill. Model checking in dense real time. *Information and Computation*, 104:2–34, 1993.

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[AHK02] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49:672–713, 2002.

[CdAH05] K. Chatterjee, L. de Alfaro, and T.A. Henzinger. The complexity of stochastic Rabin and Streett games. In *ICALP: Automata, Languages, and Programming*, Lecture Notes in Computer Science 3580, pages 878–890. Springer, 2005.

[CdAH06a] K. Chatterjee, L. de Alfaro, and T.A. Henzinger. The complexity of quantitative concurrent parity games. In *Proceedings of the 17th Annual Symposium on Discrete Algorithms*, pages 678–687. ACM Press, 2006.

[CdAH06b] K. Chatterjee, L. de Alfaro, and T.A. Henzinger. Strategy improvement for concurrent reachability games. In *Proceedings of the Third Annual Conference on Quantitative Evaluation of Systems*. IEEE Computer Society Press, 2006.

[CH06a] K. Chatterjee and T.A. Henzinger. Strategy improvement and randomized subexponential algorithms for stochastic parity games. In *STACS: Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 3884, pages 512–523. Springer, 2006.

[CH06b] K. Chatterjee and T.A. Henzinger. Strategy improvement for stochastic Rabin and Streett games. In *CONCUR: Concurrency Theory*, Lecture Notes in Computer Science 4137, pages 375–389. Springer, 2006.

[CJH03] K. Chatterjee, M. Jurdziński, and T.A. Henzinger. Simple stochastic parity games. In *CSL: Computer Science Logic*, Lecture Notes in Computer Science 2803, pages 100–113. Springer, 2003.

[CJH04] K. Chatterjee, M. Jurdziński, and T.A. Henzinger. Quantitative stochastic parity games. In *Proceedings of the 15th Annual Symposium on Discrete Algorithms*, pages 114–123. ACM Press, 2004.

[CGP99] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[Con92] A. Condon. The complexity of stochastic games. *Information and Computation*, 96:203–224, 1992.

[Con93] A. Condon. On algorithms for simple stochastic games. In *Advances in Computational Complexity Theory*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 13, pages 51–73. AMS, 1993.

[CY95] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42:857–907, 1995.

[dAlf97] L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997.

[dAFH$^+$03] L. de Alfaro, M. Faella, T.A. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. In *CONCUR: Concurrency Theory*, Lecture Notes in Computer Science 2761, pages 144–158. Springer, 2003.

[dAH00] L. de Alfaro and T.A. Henzinger. Concurrent omega-regular games. In *Proceedings of the 15th Annual Symposium on Logic in Computer Science*, pages 141–154. IEEE Computer Society Press, 2000.

[dAHK98] L. de Alfaro, T.A. Henzinger, and O. Kupferman. Concurrent reachability games. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, pages 564–575. IEEE Computer Society Press, 1998.

[dAM04] L. de Alfaro and R. Majumdar. Quantitative solution of omega-regular games. *Journal of Computer and System Sciences*, 68:374–397, 2004.

[FV97] J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer, 1997.

[GTW03] E. Gräedel, W. Thomas, and T. Wilke (eds.). *Automata, Logics, and Infinite Games*. Lecture Notes in Computer Science 2500. Springer, 2003.

[HNSY94] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111:193–244, 1994.

[HP06] T.A. Henzinger and V. Prabhu. Timed alternating-time temporal logic. In *FORMATS: Formal Modeling and Analysis of Timed Systems*, Lecture Notes in Computer Science 4202, pages 1–17. Springer, 2006.

[MPS95] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *STACS: Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 900, pages 229–242. Springer, 1995.

[Mar75] D.A. Martin. Borel determinacy. *Annals of Mathematics*, 102:363–371, 1975.

[Mar98] D.A. Martin. The determinacy of Blackwell games. *The Journal of Symbolic Logic*, 63:1565–1581, 1998.

[Tho95] W. Thomas. On the synthesis of strategies in infinite games. In *STACS: Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 900, pages 1–13. Springer, 1995.

[WH91] H. Wong-Toi and G. Hoffmann. The control of dense real-time discrete event systems. In *Proceedings of the 30th Annual Conference on Decision and Control*, pages 1527–1528. IEEE Press, 1991.