

Long-Run Average Rewards
for
Priced Timed Games

UNIVERSITY OF CALIFORNIA
SANTA CRUZ

LONG-RUN AVERAGE REWARDS FOR PRICED TIMED GAMES

A thesis submitted in partial satisfaction of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

B. Thomas Adler

June 2007

The Thesis of B. Thomas Adler
is approved:

Professor Luca de Alfaro, Chair

Professor E. Jim Whitehead

Professor Cormac Flanagan

Lisa C. Sloan
Vice Provost and Dean of Graduate Studies

Copyright © by
B. Thomas Adler
2007

Contents

Contents	iii
List of Figures	v
List of Tables	vi
Acknowledgements	vii
1 Introduction	1
1.1 Control Problems as Games	1
A Priced Timed Game	3
1.2 Related Work	6
Contributions	7
2 Defining the Game	8
2.1 The Playing Field	8
2.2 Rules of the Game	9
2.3 Describing Plays of the Game	10
2.4 Player Goals	11
Long-Run Average Reward of a Run	12
2.5 Specifying the Game	12
2.6 A Turn-based Timed Game	14
An Example Automata	15

3	Well-Formedness and Value	17
3.1	Well-Formedness	18
3.2	The Value of the Game	22
	Determinacy	23
4	Some Restrictions Apply	26
4.1	The Finite Game	27
4.2	Value Proofs	28
4.3	Memoryless Strategies	33
5	Conclusions	37
5.1	Future Work	37
	Removing the Restriction	37
	Other Directions	38
5.2	Summary	39
A	Additional Well-Formedness Proofs	41
	Bibliography	46

List of Figures

1.1	Modeling a control system	2
1.2	Modeling resources	5
2.1	A timed automaton specification	15
2.2	Turn-based equivalent automaton	16
3.1	Why time divergence is important	18
3.2	An undetermined game	24
4.1	Alternating moves	27
4.2	An example loop	29
5.1	A 2-jackpot	38

List of Tables

3.1	Potential well-formedness conditions	19
-----	--	----

Acknowledgements

This work is based on joint research conducted with Luca de Alfaro and Marco Faella. I am indebted to their guidance and instruction in understanding the setting of the problem and then developing the rigorous proofs found within this thesis. I also wish to thank my family, whose continual support buoys me through the difficult moments.

Chapter 1

Introduction

1.1 Control Problems as Games

Control systems are at the heart of the technology that we use in our daily lives. These systems run our cell phones, microwaves, and cars, and direct the devices to respond to the press of a button. When our dependence on such devices is critical, as in medical and military devices, much more effort must be expended to ensure correct operation, usually at significant expense. There is growing evidence that, even for less critical applications, failures have a substantial cumulative cost (in 2002, this was estimated at \$59 billion in the United States [12]); in effect, many devices are becoming “mission critical” to our economy. Thus, a better understanding of the theory of control systems will have widespread implications.

The actual mechanics of a control system in an embedded device takes the form of a state machine. Sensors gather data about the environment, which the controller combines with internal state variables to determine an appropriate response. Mathematically, we can then model the system as a graph where nodes represent the state of the sensors and internal variables, and directed edges represent *actions* by the controller or environment that change the state. Note that an action might represent multiple variables or sensors changing simultaneously; the start and end nodes of the action indicate how the system state has changed. We assume that that sensors report their data digitally, resulting in a finite and discrete state space for the entire system.

In Figure 1.1, we show a graph representation of a very simple controller for the “automatic

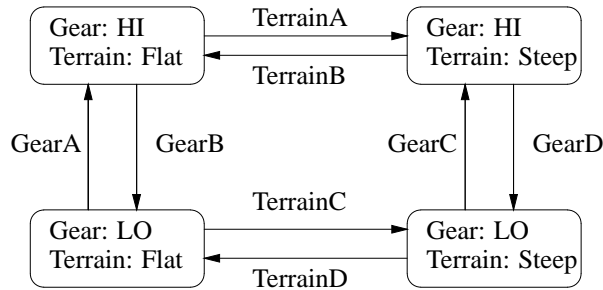


Figure 1.1: An example graph modeling a simplistic controller for the “automatic transmission” of an automobile. The controller can set the transmission to HI or LO gear, and a sensor indicates whether the terrain is flat or steep. Edges indicate changes in state that our model allows.

transmission” of an automobile; the problem the controller is trying to solve is how to control the gear shifting of the vehicle in response to the environment. The controller has one sensor, which indicates whether the terrain being driven on is flat or steep, and it controls one internal variable which sets the gear shift into HI or LO mode. When the controller indicates a gear change, it does so via one of the actions `GearA`, `GearB`, `GearC`, or `GearD` — moving the current state of the system between the top half of the graph and the bottom half. Changes in the environment are indicated by a corresponding set of four actions that move the current state of the system between the left and right halves of the graph. An important point is that our model doesn’t necessarily indicate every possible state change. For instance, it is possible that the terrain might change from flat to steep in the same instant that the controller changes the gear from HI to LO. There is no edge in Figure 1.1 that allows such a state change, but our model accommodates the possibility by allowing the system to traverse through one of the intermediate states for an instant; e.g., by a play of `TerrainA` followed by `GearD` with no advancement of time. The partitioning of actions into those which can be controlled and those which are uncontrolled, and the ability to treat concurrent actions as the composition of two actions with zero delay between them, simplifies both the modeling and the analysis of the system.

A useful paradigm in the study of control systems has been to view such systems as a game between the controller and the environment; the controller acts to ensure some goal, and the environment (in the worst case) acts to prevent the goal. Edges in the state graph represent moves that one player or the other can take. (This is why we chose to leave out the edge connecting the diagonal states in Figure 1.1, as discussed above. There is no action that either the controller or environment

can take independently that forces such a state change.) This casting of the problem allows us to draw on results and techniques from game theory to define strategies for the controller to achieve its goal [7, 17, 16]. Although this works in the simplest cases, real-time systems have additional deadline requirements; for instance, shutting down a nuclear reactor when the emergency stop is pressed. *Timed games* incorporate time variables into the state space [2], and allow the specification of *guards* on moves of the game to model the deadlines; incrementing of time variables can be accomplished (as in our framework) through an explicit *tick* action. Work on timed games has primarily focused on safety, reachability, and other ω -regular goals [14, 3, 4, 11, 8].

Even with the addition of a time component, our model from Figure 1.1 is only partially helpful. Suppose that the system is in the state where the car is in LO gear while the terrain is flat (the lower left node of the graph); by appealing to intuition (or possibly some analysis of the physical system), we know that the controller should change the gear shift to HI gear. There is not actually enough information in the model for us to decide this, however. By adding costs to the actions, we can model resources and ask whether it is possible for the controller to make optimal use of them. The goal of *efficiency* in *priced timed games* [6, 1] is one that has only recently been studied, primarily in the context of efficient reachability. In this paper, we consider the problem of a priced timed game which is played forever by two players. Their opposing goals are to maximize and minimize the long-run average cost of the game.

A Priced Timed Game

In our formulation of the controller problem, there are two players, the *controller* and the *environment*, moving a token around a finite graph. At each round, they concurrently choose a move from those available from the current location of the token. Moving the token from one position in the graph to another we call an *immediate move*, modeling the actions of the environment and responses by the controller. There are also two *timed moves* available for modeling the passage of time which influence the game clock without moving the game token: Δ_0 and Δ_1 . The move Δ_1 signals the desire by a player to advance the clock variables by one time step, while move Δ_0 does not advance the game clock, which we call a *stuttering step*. (Time steps are discrete units of time; they might represent seconds or minute or other arbitrary time increment. There might be multiple clock variables that

measure the elapsed time since they were reset; all of them advance on an increment of the game clock.) The outcome of each round is determined as follows: immediate moves take precedence over timed moves, the game clock only advances when both players play Δ_1 , and if both players choose an immediate move then one is nondeterministically chosen to move the token.

We associate rewards with each move as well. The goal of the controller is to maximize the long-run average reward (the total reward accumulated so far, divided by the elapsed time on the game clock, as the game becomes infinitely long), while the environment plays to minimize the long-run average reward. We present published results for the case where reward is linked to the advancement of time.

The game structure, as we have defined it, can be specified by a variation of the notation for timed automata. Each location is labeled by two invariants (one for each player), which specify how long the controller and environment can stay at the location. The actions labeling the edges correspond to immediate moves, and each one of them belongs to one of the two players. Each location is labeled with the reward that is accumulated if time advances one time step while in that state. Figure 1.2 shows an expanded model of the “automatic transmission” controller previously discussed, using a priced timed game as the basis.

Engines have a heat budget which they must operate within, before they fail catastrophically; we model this constraint as two deadlines. The first deadline (modeled by an invariant on the controller) requires that the automobile is to operate in HI gear on a steep grade for no more than ten minutes, before it must down-shift. The second deadline (modeled by a guard on the `Reset` action) requires a cool-down period (simplistically modeled using the same clock variable) before the engine can attempt to go into HI gear again. We have also added rewards to each state, indicating the speed that the automobile travels at when it is in that state for any amount of time. The goal of the controller in this new model is to maximize the overall average speed of the automobile, while respecting the timing constraints of the engine.

In the context of this example, this work poses the question, “what is the best average speed that the automobile can achieve?” To answer this question, we treat the environment as an adversary who strives to minimize the average speed. The example of Figure 1.2 is simple enough that we can appeal to intuition to arrive at a strategy for the controller: stay in HI gear as long as allowed. The environment’s best strategy for minimizing the performance is to always have steep terrain.

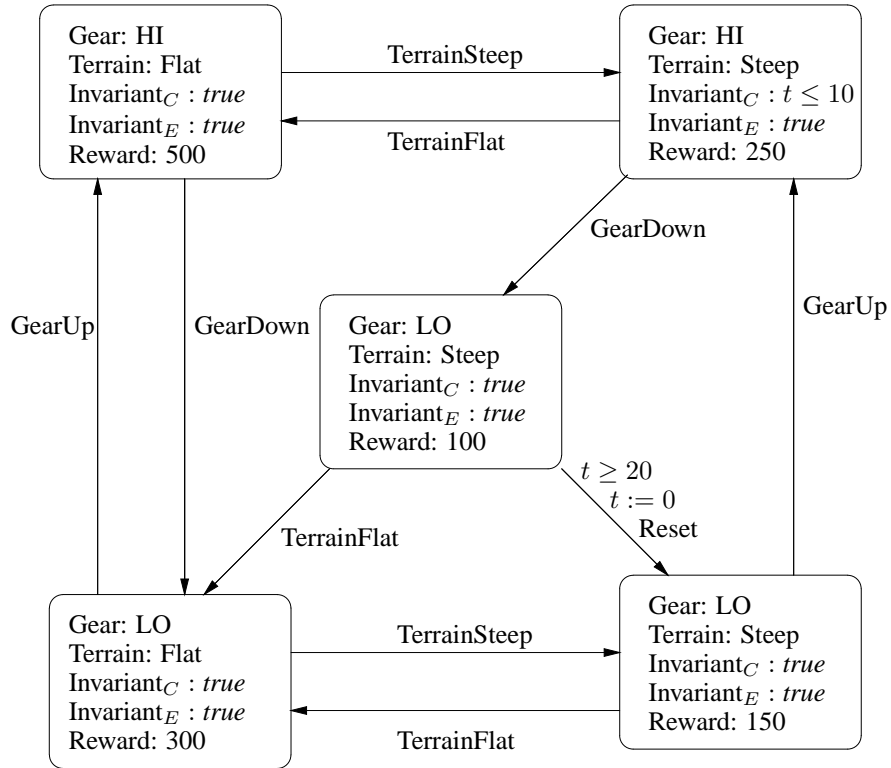


Figure 1.2: This example expands upon the control system of Figure 1.1 by modeling two resources: the heat budget of the engine and speed of the automobile. The heat budget is tracked by clock variable t , allowing the engine to be in HI gear on a steep grade for at most ten minutes before requiring a cool-down period. The speed in any system state is given by the reward in that state. The goal of the controller is to maximize the long-run average reward in an arbitrary terrain.

Although this particular example has a rather simple solution, it does not take much more complexity to make analysis intractable. For instance, if we were to model deadlines for the environment (e.g., that a steep grade must last at least three minutes) or add some additional states (e.g., uphill versus downhill), the number of nodes and cycles in the graph quickly grows beyond what can be analyzed through intuition. By treating the system as a game with a performance goal, we can answer the question of what the best performance is (and precompute a strategy to achieve this performance) by assuming that the environment is an adversary intent on minimizing the performance, subject to its own constraints.

1.2 Related Work

Game theory is a branch of mathematics studying choices of players, with the objective of achieving some goal. Many control problems can be described as a *two-player game*, between the controller and the environment, each with their own goals, and are easily represented as a decision tree where each node requires that a choice be made a player. There is a correspondence between the decision tree view of a game and automata specifications, allowing many results from game theory to be applied to computer science problems [7, 17, 16].

The deadline requirements of real-time systems has necessitated the development of *timed games* played on timed automata [2], where the advancement of time can be either an implicit action or an explicit action to be made by the players. Any representation of a timed game with discrete-time can be transformed into an untimed equivalent, and is thus equivalent to a classic two-player game. Work on timed games has primarily focused on safety, reachability, and other ω -regular goals [14, 3, 4, 11, 8].

Although timed games can be used to model and solve many kinds of problems, certain efficiency goals can't be expressed. For instance, Figure 1.1 is not able to express a goal about efficiently using gasoline, or minimizing damage to the underlying equipment. Problems relating to the use of resources require additional information; this leads to *priced timed games*, where weights are specified when time advances.

Previous work on priced timed games [6, 1] does not restrict the framework to discrete-time increments. Instead, they treat time as a continuous variable; players can choose arbitrary delays, potentially leading to a "Zeno's Paradox" situation where delays become smaller and smaller so that time never actually advances beyond a limit point. Earlier works address this by including a restriction such that the timed automaton are constructed in a way which forces time to advance. Although our framework is less expressive for being discrete time, we don't have any such structural constraints.

The previous work in priced timed games is also different in that it has been concerned with efficient reachability [6, 1] as a goal for the players. That is, given a starting state and a set of target states, what strategy incurs the least accumulated cost to reach any target state, if cost is accrued only when time advances. We consider a different goal for the player, that of achieving an efficient long-run average, which might be a more appropriate performance metric for reactive systems which are "always

on.” This is most like the work done by Bouyer, et al. [5], in which they examine double priced timed automata (including both costs *and* rewards, with restrictions) and show how to determine the optimal cost/reward ratio. Our work differs from this previous work in that we examine two-player priced timed games with non-determinism, as opposed to a deterministic single player game.

The long-run average reward in a priced timed game is equivalent to research on *mean payoff* goals in classic two player-games [10, 18, 13]. Our particular definition of long-run average reward depends on the specific *tick* action to advance time, and also has concurrent actions by players, which is a generalization of the case first considered in [10]. The player control over time potentially introduces undesired strategies which defy our notions of realistic physical systems, which we address by including a goal of not receiving blame for stopping time.

Contributions

The unique contributions made in this thesis are:

- A generalization of mean payoff games [10] where the advancement of time is an action that can be chosen by the players. We adapt and expand upon details of the earlier proof that there is a finite game with the same value as the infinite game, and that there is a memoryless strategy to achieve that value. We add one structural requirement to address physically impossible strategies that arise due to the player control of time.
- A discussion of well-formedness and value, and argument against some possibilities.
- A counter-example of determinacy in our formalism.
- An example to demonstrate the need for restricting costs to timed moves.

Chapter 2

Defining the Game

In this chapter, we present a discrete-time game structure that serves as the formalism for our proofs. We also define the goal of long-run average games played on the game structure, and introduce terms to describe plays. We then introduce priced timed automata for specifying the discrete-time game structures, and show how to create a turn-based version of the game.

2.1 The Playing Field

We define *discrete-time game structures* as a discrete-time variation of the timed game structures of [8]. A discrete-time game structure represents a game between two players, which we denote by 1, 2; we indicate by $\sim i$ the opponent of $i \in \{1, 2\}$ (that is, player $3 - i$). A *discrete-time game structure* is a tuple $\mathcal{G} = (S, Acts_1, Acts_2, \Gamma_1, \Gamma_2, \delta, r)$, where:

- S is a finite set of states.
- $Acts_1$ and $Acts_2$ are two disjoint sets of actions for player 1 and player 2, respectively. We assume that $\Delta_0, \Delta_1 \notin Acts_i$ and write $M_i = Acts_i \cup \{\Delta_0, \Delta_1\}$ for the sets of moves of player $i \in \{1, 2\}$.
- For $i = 1, 2$, the function $\Gamma_i : S \mapsto 2^{M_i} \setminus \emptyset$ is an enabling condition, which assigns to each state s a set $\Gamma_i(s)$ of moves available to player i in that state.
- $\delta : S \times (M_1 \cup M_2) \mapsto S$ is a destination function that, given a state and a move of either player, determines the next state in the game.

- $r : S \times (M_1 \cup M_2) \mapsto \mathbb{Z}$ is a function that associates a *reward* with each state $s \in S$ and move of either player: this is the reward that is accumulated to the game total when the given move is played at state s .

The move Δ_0 represents a stuttering move that takes zero amount of time, and allows a player to “pass” for a round to see what action the opponent will take. We require that Δ_0 is always enabled: for $s \in S$ and $i \in \{1, 2\}$, we always have $\Delta_0 \in \Gamma_i(s)$. When taken, the move Δ_0 does not cause a state change: for all $s \in S$, we have $\delta(s, \Delta_0) = s$. The moves in $\{\Delta_0\} \cup Acts_1 \cup Acts_2$ are known as the *zero-time moves*. The move Δ_1 represents the decision of waiting for 1 time unit. We do not require that Δ_1 be always enabled: if we have $\Delta_1 \notin \Gamma_i(s)$ for player $i \in \{1, 2\}$ at a state $s \in S$, then player i cannot wait, but must immediately play a zero-time move. We define the *size* of a discrete-time game structure by $|\mathcal{G}| = \sum_{s \in S} (|\Gamma_1(s)| + |\Gamma_2(s)|)$.

2.2 Rules of the Game

A timed game proceeds as follows. At each state $s \in S$, player 1 chooses a move $a^1 \in \Gamma_1(s)$, and simultaneously and independently, player 2 chooses a move $a^2 \in \Gamma_2(s)$. The successor state $\tilde{\delta}(s, a^1, a^2)$ is then determined according to the following rules.

- *Actions take precedence over stutter steps and time steps.* If $a^1 \in Acts_1$ or $a^2 \in Acts_2$, then the game takes an action a^T selected nondeterministically from $\{a^1, a^2\} \cap (Acts_1 \cup Acts_2)$, and the game proceeds to location $\tilde{\delta}(s, a^1, a^2) \triangleq \delta(s, a^T)$.
- *Stutter steps take precedence over time steps.* If $a^1, a^2 \in \{\Delta_0, \Delta_1\}$, there are two cases.
 - If $a^1 = \Delta_0$ or $a^2 = \Delta_0$, the game performs a stutter step, and $\tilde{\delta}(s, a^1, a^2) \triangleq s$.
 - If $a^1 = a^2 = \Delta_1$, then the game performs a time step of duration 1, and the game proceeds to $\tilde{\delta}(s, \Delta_1, \Delta_1) \triangleq \delta(s, \Delta_1) \triangleq s$.

Fitting a real-world situation to this framework has an issue that might seem strange at first glance. The awkwardness appears in the nondeterminism if both players choose an action in the same round; only one action is selected and the other is discarded, but actions by the environment seem “sacred” in the sense that we cannot just decide that they did not happen. It is the constructed model that must

resolve this, by allowing the action of the environment from any state where it is possible to occur. The action by the controller might be chosen, but in subsequent rounds the environment can again play the same action until it is selected. (Recall that actions have precedence over timed moves and take zero time themselves, so there might be multiple rounds that occur in the same instant of time.) This turns out to be a modeling detail that is specific to designing control systems; the mathematics of our proofs does not actually require that actions of the environment always be enabled.

2.3 Describing Plays of the Game

An *infinite run* (or simply *run*) of the discrete-time game structure \mathcal{G} is a sequence

$$s_0, \langle a_0^1, a_0^2, a_0^T \rangle, s_1, \langle a_1^1, a_1^2, a_1^T \rangle, s_2, \dots$$

describing a play of the game starting from state s_0 such that, for all $k \geq 0$, we have:

$$\begin{aligned} s_k &\in S, \\ a_k^1 &\in \Gamma_1(s_k), \\ a_k^2 &\in \Gamma_2(s_k), \\ a_k^T &\in \{a_k^1, a_k^2\} \\ s_{k+1} &\in \tilde{\delta}(s_k, a_k^1, a_k^2), \\ s_{k+1} &= \delta(s_k, a_k^T). \end{aligned}$$

where a_k^T is the action that was actually taken from state s_k to state s_{k+1} . A *finite run* σ is a finite prefix of a run that terminates at a state s_n ; we then set $last(\sigma) = s_n$.

We denote by $FRuns$ the set of all finite runs of the game structure, and by $Runs$ the set of its infinite runs. For a finite or infinite run σ , and a number $k < |\sigma|$, we denote by $\sigma_{\leq k}$ the prefix of σ up to and including state σ_k . That is,

$$\sigma_{\leq k} = s_0, \langle a_0^1, a_0^2, a_0^T \rangle, s_1, \langle a_1^1, a_1^2, a_1^T \rangle, s_2, \dots, s_{k-1}, \langle a_{k-1}^1, a_{k-1}^2, a_{k-1}^T \rangle, s_k$$

A state s' is *reachable* from another state s if there exists a finite run

$$s_0, \langle a_0^1, a_0^2, a_0^T \rangle, s_1, \dots, s_n$$

such that $s_0 = s$ and $s_n = s'$.

A *strategy*, π_i , for player $i \in \{1, 2\}$ is a mapping $\pi_i : FRuns \mapsto M_i$ that associates with each finite run $s_0, \langle a_0^1, a_0^2, a_0^T \rangle, s_1, \dots, s_n$ the move to be played at s_n , denoted by

$$\pi_i(s_0, \langle a_0^1, a_0^2, a_0^T \rangle, s_1, \dots, s_n)$$

We require that the strategy only selects enabled moves, that is, $\pi_i(\sigma) \in \Gamma_i(last(\sigma))$ for all $\sigma \in FRuns$. For $i \in \{1, 2\}$, let Π_i denote the set of all player i strategies. For strategies $\pi_1 \in \Pi_1$ and $\pi_2 \in \Pi_2$, we say that a run $s_0, \langle a_0^1, a_0^2, a_0^T \rangle, s_1, \langle a_1^1, a_1^2, a_1^T \rangle, s_2, \dots$ is *consistent* with π_1 and π_2 if, for all $n \geq 0$ and $i = 1, 2$, we have $\pi_i(s_0, \langle a_0^1, a_0^2, a_0^T \rangle, s_1, \dots, s_n) = a_n^i$. We denote by $Outcomes(s, \pi_1, \pi_2)$ the set of all infinite runs that start in s and are consistent with π_1, π_2 . Note that in our timed game, two strategies and a start state yield a *set* of outcomes, because if the players both propose actions, a nondeterministic choice between the two moves is made. According to this definition, strategies can base their choices on the entire history of the game, consisting of both past states and moves.

2.4 Player Goals

We consider games where the goal for player 1 consists in maximizing the average reward per time unit obtained along a game outcome. The goal for player 2 is symmetrical, and it consists in minimizing the average reward per time unit obtained along a game outcome. It bears mentioning that we assume *time divergence* as part of the goal; that is, that players don't play in such a way that causes time to stop progressing.

Long-Run Average Reward of a Run

To define the long-run average reward of an infinite run, we must first define the average reward for finite runs, and then extend this to the infinite case. Consider a finite run

$$\sigma = s_0, \langle a_0^1, a_0^2, a_0^T \rangle, s_1, \dots, s_n$$

and for all $k = 0 \dots n$, denote by σ_k its k -th state s_k , and by σ_k^1, σ_k^2 the moves a_k^1 and a_k^2 played by players 1 and 2 at the k -th position of σ . We also let σ_k^T represent the move which was actually taken, leading to state σ_{k+1} . The time elapsed, T_k , at step k of the run is defined by $T_k(\sigma) = 1$ if $\sigma_k^1 = \sigma_k^2 = \Delta_1$, and $T_k(\sigma) = 0$ otherwise; the reward R_k accrued at step k of the run is given by $R_k(\sigma) = r(\sigma_k, \sigma_k^T)$. The time elapsed during σ and the reward achieved during σ are defined in the obvious way, by $T(\sigma) = \sum_{k=0}^n T_k(\sigma)$ and $R(\sigma) = \sum_{k=0}^n R_k(\sigma)$.

As a game is played, the finite run representing the game so far grows with each round. Since rewards may accumulate at different positions than where time elapses, the average reward may not approach a single value. For this reason, we define the long-run average reward of an infinite run, σ' , to be the infimum of the average reward, as given by

$$\bar{r}(\sigma') = \liminf_{n \rightarrow \infty} \frac{R(\sigma'_{\leq n})}{T(\sigma'_{\leq n})}.$$

2.5 Specifying the Game

We specify discrete-time game structures via *discrete-time game automata*, which are a discrete-time version of the *timed automaton games* of [8]; both models are two-player versions of timed automata [2]. A *clock condition* over a set C of clocks is a boolean combination of formulas of the form $x \preceq c$ or $x - y \preceq c$, where c is an integer, $x, y \in C$, and \preceq is either $<$ or \leq . We denote the set of all clock conditions over C by $\text{ClkConds}(C)$. A *clock valuation* is a function $\kappa : C \mapsto \mathbb{R}_{\geq 0}$, and we denote by $K(C)$ the set of all clock valuations for C .

A *discrete-time game automaton* is a tuple

$$\mathcal{A} = (Q, C, \text{Acts}_1, \text{Acts}_2, E, \theta, \rho, \text{Inv}_1, \text{Inv}_2, \text{SRew}, \text{ERew})$$

where:

- Q is a finite set of locations.
- C is a finite set of clocks.
- $Acts_1$ and $Acts_2$ are two disjoint, finite sets of actions for player 1 and player 2, respectively.
- $E \subseteq Q \times (Acts_1 \cup Acts_2) \times Q$ is an edge relation.
- $\theta : E \mapsto ClkConds(C)$ is a mapping that associates with each edge a clock condition that specifies when the edge can be traversed. We require that for all $(q, a, q_1), (q, a, q_2) \in E$ with $q_1 \neq q_2$, the conjunction $\theta(q, a, q_1) \wedge \theta(q, a, q_2)$ is unsatisfiable. In other words, the game move and clock values determine uniquely the successor location.
- $\rho : E \mapsto 2^C$ is a mapping that associates with each edge the set of clocks to be reset when the edge is traversed.
- $Inv_1, Inv_2 : Q \mapsto ClkConds(C)$ are two functions that associate with each location an invariant for player 1 and 2, respectively.
- $SRew : Q \mapsto \mathbb{Z}$ is a function that assigns a reward $SRew(q) \in \mathbb{Z}$ with each $q \in Q$, accumulated when time advances at location q .
- $ERew : E \mapsto \mathbb{Z}$ is a function that assigns a reward $ERew(e) \in \mathbb{Z}$ to each edge $e \in E$, accumulated when that edge is traversed as part of a run.

Given a clock valuation $\kappa : C \mapsto \mathbb{R}_{\geq 0}$, we denote by $\kappa + 1$ the valuation defined by $(\kappa + 1)(x) = \kappa(x) + 1$ for all clocks $x \in C$. The clock valuation $\kappa : C \mapsto \mathbb{R}_{\geq 0}$ *satisfies* the clock constraint $\alpha \in ClkConds(C)$, written $\kappa \models \alpha$, if α holds when the clocks have the values specified by κ . For a subset $C' \subseteq C$ of clocks, $\kappa[C' := 0]$ denotes the valuation defined by $\kappa[C' := 0](x) = 0$ if $x \in C'$, and by $\kappa[C' := 0](x) = \kappa(x)$ otherwise.

The discrete-time game automaton \mathcal{A} induces a discrete-time game structure $\llbracket \mathcal{A} \rrbracket$, whose states consist of a location of \mathcal{A} and a clock valuation over C . The idea is the following. The move Δ_0 is always enabled at all states $\langle q, \kappa \rangle$, and leads again to $\langle q, \kappa \rangle$. The move Δ_1 is enabled for player $i \in \{1, 2\}$ at state $\langle q, \kappa \rangle$ if $\kappa + 1 \models Inv_i(q)$; the move leads to state $\langle q, \kappa + 1 \rangle$. For player $i \in \{1, 2\}$

and $a \in Acts_i$, the move a is enabled at a state $\langle q, \kappa \rangle$ if there is a transition (q, a, q') in E which is enabled at $\langle q, \kappa \rangle$, and if the invariant $Inv_i(q')$ holds for the destination state $\langle q', \kappa[\rho(q, a, q') := 0] \rangle$. If the values of the clocks can grow without bound, this translation would yield an infinite-state discrete-time game structure. However, we can define *clock regions* similarly to timed automata [2], and we can include in the discrete-time game structure only one state per clock region; as usual, this leads to a finite state space.

2.6 A Turn-based Timed Game

As a further refinement, we now describe a *turn-based* version of the timed game, which can be played on the same discrete-time game structure. Turn-based games are easier to reason about, and we will later discover that this one has an equivalent value to the concurrent version. In the turn-based game, player 1 chooses his move before player 2 at each round. Player 2 can thus use her knowledge of player 1's move to choose her own. Moreover, when both players choose an action, the action chosen by player 2 is carried out. Notice that if player 2 prefers to carry out the action chosen by player 1, she can reply with the stuttering move Δ_0 . Definitions pertaining this game have a “ t_∞ ” superscript that stands for “turn-based infinite”. We define the *turn-based joint destination function*, $\tilde{\delta}^t : S \times M_1 \times M_2 \mapsto S$, by

$$\tilde{\delta}^t(s, a^1, a^2) = \begin{cases} \delta(s, \Delta_1) & \text{if } a^1 = a^2 = \Delta_1 \\ \delta(s, \Delta_0) & \text{if } \{a^1, a^2\} \subseteq \{\Delta_0, \Delta_1\} \text{ and } a^1 = \Delta_0 \text{ or } a^2 = \Delta_0 \\ \delta(s, a^1) & \text{if } a^1 \in Acts_1 \text{ and } a^2 \in \{\Delta_0, \Delta_1\} \\ \delta(s, a^2) & \text{if } a^2 \in Acts_2 \end{cases}$$

As before, a run is an infinite sequence $s_0, \langle a_0^1, a_0^2, a_0^T \rangle, s_1, \langle a_1^1, a_1^2, a_1^T \rangle, s_2, \dots$ such that $s_k \in S$, $a_k^1 \in \Gamma_1(s_k)$, $a_k^2 \in \Gamma_2(s_k)$, and $s_{k+1} \in \tilde{\delta}^t(s_k, a_k^1, a_k^2)$ for all $k \geq 0$. A *1-run* is a finite prefix of a run ending in a state s_k , while a *2-run* is a finite prefix of run ending in a move $a^1 \in M_1$. For a 2-run $\sigma = s_0, \langle a_1^1, a_1^2 \rangle, s_1, \dots, s_n, \langle a_n^1 \rangle$, we set $last(s_0, \langle a_1^1, a_1^2 \rangle, s_1, \dots, s_n, \langle a_n^1 \rangle) = s_n$ and $lastact(s_0, \langle a_1^1, a_1^2 \rangle, s_1, \dots, s_n, \langle a_n^1 \rangle) = a_n^1$. For $i \in \{1, 2\}$, we denote by $FRuns_i$ the set of all i -runs. Intuitively, i -runs are runs where it is player i 's turn to move. In the turn-based game, a *strategy*,

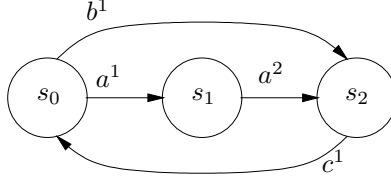


Figure 2.1: A sample timed automaton is shown, where time is allowed to diverge in every state. All state invariants and transition guards are *true*. All rewards are set to zero.

π_i , for player $i \in \{1, 2\}$ is a mapping $\pi_i : FRuns_i \mapsto M_i$ such that $\pi_i(\sigma) \in \Gamma_i(last(\sigma))$ for all $\sigma \in FRuns_i$. For $i \in \{1, 2\}$, let Π_i^t denote the set of all player i strategies; notice that $\Pi_1^t = \Pi_1$. Player 1 memoryless strategies are defined as usual. We say that a player 2 strategy $\pi_2 \in \Pi_2^t$ is *memoryless* iff, for all $\sigma, \sigma' \in FRuns_2$, $last(\sigma) = last(\sigma')$ and $lastact(\sigma) = lastact(\sigma')$ imply $\pi_2(\sigma) = \pi_2(\sigma')$.

For strategies $\pi_1 \in \Pi_1^t$ and $\pi_2 \in \Pi_2^t$, we say that a run $s_0, \langle a_0^1, a_0^2, a_0^T \rangle, s_1, \langle a_1^1, a_1^2, a_1^T \rangle, s_2, \dots$ is *consistent* with π_1 and π_2 if, for all $n \geq 0$ and $i \in \{1, 2\}$, we have $\pi_1(s_0, \langle a_0^1, a_0^2, a_0^T \rangle, s_1, \dots, s_n) = a_n^1$ and $\pi_2(s_0, \langle a_1^1, a_1^2 \rangle, s_1, \dots, s_n, \langle a_n^1 \rangle) = a_n^2$. Since $\tilde{\delta}^t$ is deterministic, for all $s \in S$, there is a unique run that starts in s and is consistent with π_1 and π_2 ; we call this run $Outcomes^{t\infty}(s, \pi_1, \pi_2)$.

An Example Automata

It can be helpful to imagine that the discrete-time game structure has “ghost” states, representing positions where it is player 2’s turn to move (one for each move of player 1, in fact). As an example, consider a timed automata (shown in Figure 2.1), which has no explicit clocks or invariants; time is allowed to diverge in every state. The expansion of the Figure 2.1 automaton into a turn-based game, shown in Figure 2.2, results in several additional “ghost” states representing moves by player 2 (one for every possible move of player 1, in fact). From player 2 states, the moves Δ_0 and Δ_1 always have the same destination in this particular expansion, and are shown together as a single edge.

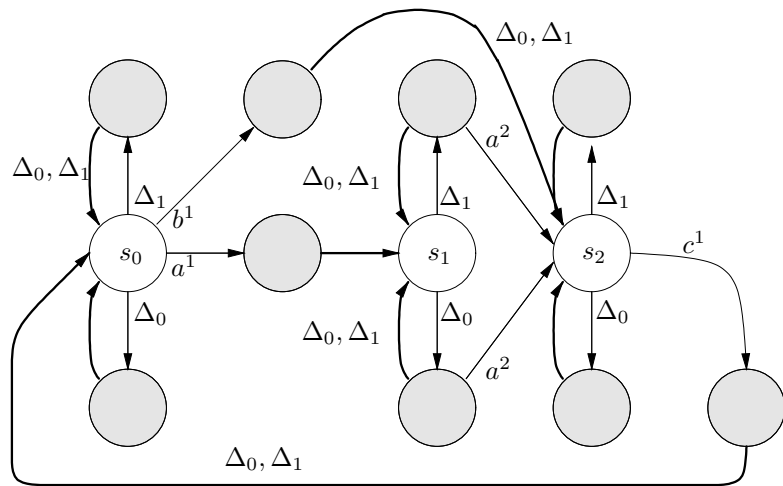


Figure 2.2: This figure shows the turn-based equivalent of Figure 2.1, where “ghost” states have been added to represent positions where player 2 must play a move. Notice that any play by player 1 from state s_2 leads to a state where player 2 can choose to play a^2 and override player 1’s choice. Bold arrows represent moves Δ_0 and Δ_1 made by player 2, which always have the same destination in this particular automaton.

Chapter 3

Well-Formedness and Value

When devising control systems, it is often useful to consider “worst-case scenarios” for how the overall system will behave; casting the problem as a game between the controller and the environment is a natural way to analyze the system for the worst case. In particular, we can define the *value* of a game structure to each player as being the maximum reward that a player can ensure (or *secure*), no matter how his opponent plays. This forces us to analyze the system with the assumption that the environment is clever and will implement a strategy to thwart the controller. A formal definition of the value, for player 1, of a game structure, when starting from state s , is simply:

$$\tilde{v}_1(\mathcal{G}, s) = \sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} \inf \{ \bar{r}(\sigma) \mid \sigma \in \text{Outcomes}(s, \pi_1, \pi_2) \}.$$

A short-coming of modeling real systems, however, is that spurious solutions might arise. Our model, as presented, is simplistic in its handling of time: player actions take zero time, and time only advances when both players agree to advance it (by playing Δ_1). In embedded software, this isn’t an unrealistic view; the idea is that actions are typically so fast as compared to a global timer that all actions can be completed before the timer expires.

The game as we have defined it so far, however, does nothing to prohibit such strategies. For instance, nothing about our game restricts both players from always playing Δ_0 if the game structure allows it, preventing time from ever advancing. As a more concrete example, consider the game of Figure 3.1 (on page 18). We have $\tilde{v}_1(\langle q_0, [x := 0] \rangle) = 4$, where the optimal strategy for player 1

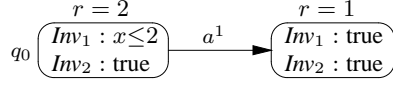


Figure 3.1: A game automaton where player 1 can freeze time to achieve a higher average reward. The labeled rewards are accumulated only when time advances in each state.

consists in staying at q_0 forever, never playing the move a^1 . Due to the invariant $x \leq 2$, such a strategy blocks the progress of time: once $x = 2$, the only move player 1 can play is Δ_0 . It is easy to see that the only strategies of player 1 that do not block time eventually play move a^1 , and have value 1. Note that the game does not contain any blocked states, i.e., from every reachable state there is a run which is time-divergent: the lack of time progress of the above-mentioned strategy is due to the fact that player 1 values obtaining a high average reward more than letting time progress.

Obviously, we need to forbid players from “stopping time”. There are two typical ways to do this: define a well-formedness condition to forbid runs which allow time to freeze, or redefine the reward to punish strategies which do not advance time. We first explore well-formedness conditions, and then present an alternative formulation of the game reward to address time divergence.

3.1 Well-Formedness

Well-formedness is a technique for defining which runs are “legal” for a player; that is, we specify the conditions that must hold along a run for it to be meaningful “in the real world.” The game structure is described as *well-formed* if both players have strategies such that every outcome of the game meets the well-formedness conditions of each player. In the case of the game we are analyzing, we would like to ensure that either time diverges in a run, or the player is not responsible for stopping time. Formalizing this requirement has been previously explored in timed games [3, 9, 8], and we borrow the same notions here.

We first define two predicates on states and the moves played from the states. A state represents a *tick event* if both players agree to advance time: $tick(s_j, \langle a_j^1, a_j^2, a_j^T \rangle)$ is true iff $a_j^T = \Delta_1$. Additionally, we describe a player i as receiving blame at a state if he does not try to advance time: $bl_i(s_j, \langle a_j^1, a_j^2, a_j^T \rangle)$ is true iff $a_j^i \neq \Delta_1$. Using these predicates, we can specify the well-formedness conditions for each player using linear temporal logic [15].

We proposed five different formulations for how to assign responsibility for stopping time in a run, documented in Table 3.1. When considering them informally, each proposal seems somewhat reasonable, highlighting the general difficulty of formalizing specifications. In practice, we can immediately disqualify (a) and (e) by considering a timed automaton where invariants forbid the advancement of time by either player; clearly, no run is valid, but the well-formedness conditions of both players allow all runs. Proposal (d) can also be disqualified, by considering a run where the players alternate playing Δ_1 , but never play it at the same time; again, time does not advance, but each player repeatedly satisfies the predicate $\neg bl_i$.

The two proposals left are both agreeable to our intended specification. Proposal (b) has the nice property that $\diamond \square \neg bl_1$ and $\square \diamond bl_1$ are logical negations of each other, so any run where time does not diverge clearly fails one of the two conditions. What is less obvious is that when time doesn't diverge, eventually it must hold that $\neg bl_2 \implies bl_1$, leading us to infer:

$$\diamond \square \neg bl_2 \implies \diamond \square bl_1 \implies \square \diamond bl_1$$

Thus, all runs where time does not diverge will fail one or both well-formedness conditions in proposals (b) and (c). Given the choice, we then prefer proposal (c) for its symmetry, and being the discrete-time equivalent of the winning condition used in [8].

We can now formally define that runs satisfy the well-formedness condition for player i if time

(a)	$WFC_1(\sigma) \triangleq \sigma \models \square \diamond tick \vee \sigma \models \diamond \square bl_2$
	$WFC_2(\sigma) \triangleq \sigma \models \square \diamond tick \vee \sigma \models \diamond \square bl_1$
(b)	$WFC_1(\sigma) \triangleq \sigma \models \square \diamond tick \vee \sigma \models \diamond \square \neg bl_1$
	$WFC_2(\sigma) \triangleq \sigma \models \square \diamond tick \vee \sigma \models \square \diamond bl_1$
(c)	$WFC_1(\sigma) \triangleq \sigma \models \square \diamond tick \vee \sigma \models \diamond \square \neg bl_1$
	$WFC_2(\sigma) \triangleq \sigma \models \square \diamond tick \vee \sigma \models \diamond \square \neg bl_2$
(d)	$WFC_1(\sigma) \triangleq \sigma \models \square \diamond tick \vee \sigma \models \square \diamond \neg bl_1$
	$WFC_2(\sigma) \triangleq \sigma \models \square \diamond tick \vee \sigma \models \square \diamond \neg bl_2$
(e)	$WFC_1(\sigma) \triangleq \sigma \models \square \diamond tick \vee \sigma \models \square \diamond bl_2$
	$WFC_2(\sigma) \triangleq \sigma \models \square \diamond tick \vee \sigma \models \square \diamond bl_1$

Table 3.1: Five proposals for well-formedness conditions that prescribe valid runs for each player.

diverges or player i is eventually always blameless for time not diverging:

$$\text{WFC}_i(\sigma) \triangleq \sigma \models \square \diamond \text{tick} \vee \sigma \models \diamond \square \neg \text{bl}_i$$

Additionally, the well-formedness of the game structure for each player i is defined as:

$$\text{WF}_i(\mathcal{G}) \triangleq \exists \pi_i \in \Pi_i^t. \forall \pi_{\sim i} \in \Pi_{\sim i}^t. \forall \sigma \in \text{Outcomes}^{t\infty}(s, \pi_i, \pi_{\sim i}) \mid \text{WFC}_i(\sigma)$$

Theorem 1. *A run of the game satisfies the well-formedness condition for both players iff time diverges in the run.*

Formally:

$$\forall \pi_1 \in \Pi_1^t. \forall \pi_2 \in \Pi_2^t. \forall \sigma \in \text{Outcomes}^{t\infty}(s, \pi_1, \pi_2) \mid \text{WFC}_1(\sigma) \wedge \text{WFC}_2(\sigma) \iff \sigma \models \square \diamond \text{tick}$$

Proof. SKETCH: The reverse direction trivially satisfies the definition of the well-formedness condition for both players. In the forward direction, we show that satisfying the well-formedness condition for both players leads to a run where both players are blameless, which is equivalent to time diverging.

(1)1. The reverse direction of the theorem is trivially true. The definition of well-formedness for each player is:

$$\text{WFC}_1(\sigma) \iff \sigma \models \square \diamond \text{tick} \vee \sigma \models \diamond \square \neg \text{bl}_1$$

$$\text{WFC}_2(\sigma) \iff \sigma \models \square \diamond \text{tick} \vee \sigma \models \diamond \square \neg \text{bl}_2$$

Time divergence satisfies the first clause of the disjunction for both definitions, \therefore both WFC_1 and WFC_2 are true and thus all runs where time is divergent satisfy the well-formedness conditions.

(1)2. For all runs which satisfy the well-formedness condition for both players, time either diverges or both players are eventually forever blameless.

Proof. By hypothesis of the forward direction of the theorem, we are considering all runs which satisfy the well-formedness condition for both players when they are playing some strategies

$\pi_1 \in \Pi_1^t$ and $\pi_2 \in \Pi_2^t$, respectively. Then, for all $\sigma \in \text{Outcomes}^{t\infty}(s, \pi_1, \pi_2)$, we have:

$$\begin{aligned}
& \text{WFC}_1(\sigma) \wedge \text{WFC}_2(\sigma) \\
& \iff (\sigma \models \Box \Diamond \text{tick} \vee \sigma \models \Box \Diamond \neg \text{bl}_1) \wedge (\sigma \models \Box \Diamond \text{tick} \vee \sigma \models \Box \Diamond \neg \text{bl}_2) \\
& \iff \sigma \models \Box \Diamond \text{tick} \vee (\sigma \models \Box \Diamond \neg \text{bl}_1 \wedge \sigma \models \Box \Diamond \neg \text{bl}_2) \\
& \iff \sigma \models \Box \Diamond \text{tick} \vee \sigma \models \Box \Diamond (\neg \text{bl}_1 \wedge \neg \text{bl}_2)
\end{aligned}$$

(1)3. Time must diverge for all run which satisfy the well-formedness condition for both players.

Proof. SKETCH: Both players being always blameless implies that time diverges.

(2)1. If both players are eventually always blameless in a run of the game, then time must diverge in that run.

SKETCH: The *tick* event is equivalent to both players making a blameless move. Thus, an infinite number of blameless move is equivalent to an infinite number of *tick* events.

(3)1. By definition, time divergence is the property that there is always a *tick* event at some time in the future: $\sigma \models \Box \Diamond \text{tick}$

(3)2. By definition, a *tick* event occurs when both players choose the Δ_1 move in the same round, and are thus both without blame for the round: $\sigma \models \Box \Diamond \text{tick} \iff \sigma \models \Box \Diamond (\neg \text{bl}_1 \wedge \neg \text{bl}_2)$

(3)3. If both players are eventually always blameless, they are also always eventually blameless. This follows from the fact that always being blameless is a stronger condition: $\sigma \models \Box \Diamond (\neg \text{bl}_1 \wedge \neg \text{bl}_2) \implies \sigma \models \Box \Diamond (\neg \text{bl}_1 \wedge \neg \text{bl}_2)$

(3)4. From (3)2 and (3)3, we then have: $\sigma \models \Box \Diamond (\neg \text{bl}_1 \wedge \neg \text{bl}_2) \implies \sigma \models \Box \Diamond \text{tick}$

(2)2. The result of (1)2 then implies more simply that $\sigma \models \Box \Diamond \text{tick}$, therefore:

$$\forall \sigma \in \text{Outcomes}^{t\infty}(s, \pi_1, \pi_2). (\text{WFC}_1(\sigma) \wedge \text{WFC}_2(\sigma) \implies \sigma \models \Box \Diamond \text{tick})$$

since the right term of the disjunction implies time divergence.

(1)4. Result (1)3 shows the forward direction of the theorem, and result (1)1 shows the reverse.

Therefore, the theorem is true as stated. □

Corollary 2. *If time cannot advance in any run of a game, then that game is not well-formed.*

Formally:

$$\forall \pi_1 \in \Pi_1^t. \forall \pi_2 \in \Pi_2^t. \forall \sigma \in \text{Outcomes}^{t\infty}(s, \pi_1, \pi_2) \mid \sigma \models \neg \square \diamond \text{tick} \implies \neg \text{WF}_1(\sigma) \vee \neg \text{WF}_2(\sigma)$$

Proof. Directly follows from the contrapositive of the previous theorem. □

Additional proofs relating to well-formedness can be found in Appendix A.

3.2 The Value of the Game

An alternative to having a well-formedness condition is to redefine the rewards of the game to punish undesirable behaviors. Although both techniques are essentially equivalent for the proofs in this report, we find the implementation to be simpler when only rewards must be considered.

To ensure that winning strategies do not block the progress of time, we modify the definition of value of a run, so that ensuring time divergence has higher priority than maximizing the average reward. Following [8], we introduce the following predicates:

- For $i \in \{1, 2\}$, we denote by $\text{blameless}_i(\sigma)$ (“blameless i ”) the predicate defined by $\exists n \geq 0. \forall k \geq n. \sigma_k^i = \Delta_1$. Intuitively, $\text{blameless}_i(\sigma)$ holds if, along σ , player i beyond a certain point does not play any moves that block the progress of time. (This corresponds to the LTL expression “ $\sigma \models \diamond \square \neg \text{bl}_i$ ” which we explored in the previous section.)
- We denote by $\text{td}(\sigma)$ (“time-divergence”) the predicate defined by $\forall n \geq 0. \exists k \geq n. [(\sigma_k^1 = \Delta_1) \wedge (\sigma_k^2 = \Delta_1)]$. (This is exactly “ $\sigma \models \square \diamond \text{tick}$ ”, as explored in the previous section.)

We define the value of a run $\sigma \in \text{Runs}$ for player $i \in \{1, 2\}$ by:

$$w_i(\sigma) = \begin{cases} +\infty & \text{if } \text{blameless}_i(\sigma) \wedge \neg \text{td}(\sigma); \\ (-1)^{(i+1)} \bar{r}(\sigma) & \text{if } \text{td}(\sigma); \\ -\infty & \text{if } \neg \text{blameless}_i(\sigma) \wedge \neg \text{td}(\sigma). \end{cases} \quad (3.1)$$

It is easy to check that, for each run, exactly one of the three cases of the above definition applies. Also, notice that if $td(\sigma)$ holds, then $w_1(\sigma) = -w_2(\sigma)$, so that the value of time-divergent runs is defined in a zero-sum fashion. We define the value of the game for player i at $s \in S$ as follows:

$$v_i(\mathcal{G}, s) = \sup_{\pi_i \in \Pi_i} \inf_{\pi_{\sim i} \in \Pi_{\sim i}} \inf \{w_i(\sigma) \mid \sigma \in \text{Outcomes}(s, \pi_1, \pi_2)\}. \quad (3.2)$$

We omit the argument \mathcal{G} from $v_i(\mathcal{G}, s)$ when clear from the context.

Note that equation (3.2) resolves nondeterminism in favor of player 2; intuitively, player 1 must be able to secure a value even if “luck” should go against her. Since our construction of the turn-based game gives preference to player 2 as well, we have:

Theorem 3. *The value of the turn-based game is the same as the value of the concurrent game:*

$$v_1(\mathcal{G}, s) = v_1^{\dagger\infty}(\mathcal{G}, s)$$

Proof. Consider any $\pi_1 \in \Pi_1$. Since $\Pi_1 = \Pi_1^{\dagger}$, player 1 can employ π_1 in both the original timed game and in the turn-based game. We show that, in the turn-based game, π_1 can achieve at least the same value as in the original timed game. Consider any $\pi_2 \in \Pi_2^{\dagger}$ and $\sigma \in \text{Outcomes}^{\dagger\infty}(s, \pi_1, \pi_2)$. Player 2 cannot directly use π_2 in the original timed game, because in that game she cannot base her decisions on the current move of player 1. However, since π_1 is fixed, we can find $\pi_2' \in \Pi_2$ that *guesses* the move of player 1 at each step, effectively simulating the behavior of π_2 when played against π_1 . It then holds that $\sigma \in \text{Outcomes}(s, \pi_1, \pi_2')$. Since this holds for any π_1 , we have that $v_1(s) \leq v_1^{\dagger\infty}(s)$.

The inequality $v_1(s) \geq v_1^{\dagger\infty}(s)$ is immediate, since it is clearly an advantage for player 1 to conceal his move from player 2 at each round.

□

Determinacy

A game is *determined* if, for all $s \in S$, we have $v_1(s) + v_2(s) = 0$: this means that if player $i \in \{1, 2\}$ cannot enforce a reward $c \in \mathbb{R}$, then player $\sim i$ can enforce at least reward $-c$. The following theorem provides a strong non-determinacy result for average-reward discrete-time games.

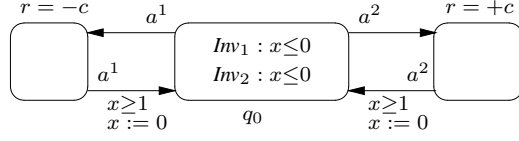


Figure 3.2: A game automaton. Unspecified guards and invariants are “true”.

Theorem 4. (non-determinacy) *For all $c > 0$, there exists a game structure*

$$\mathcal{G} = (S, Acts_1, Acts_2, \Gamma_1, \Gamma_2, \delta, r)$$

with a state $s \in S$, and two “spoiling” strategies $\pi_1^* \in \Pi_1$, $\pi_2^* \in \Pi_2$, such that the following holds:

$$\begin{aligned} \sup_{\pi_1 \in \Pi_1} \sup \{w_1(\sigma) \mid \sigma \in Outcomes(s, \pi_1, \pi_2^*)\} &\leq -c \\ \sup_{\pi_2 \in \Pi_2} \sup \{w_2(\sigma) \mid \sigma \in Outcomes(s, \pi_1^*, \pi_2)\} &\leq -c. \end{aligned}$$

As a consequence, $v_1(s) \leq -c$ and $v_2(s) \leq -c$.

Proof. Consider the game of Figure 3.2. We take for $\pi_1^* \in \Pi_1$ and $\pi_2^* \in \Pi_2$ the strategies that always play Δ_0 at q_0 . Let $s_0 = \langle q_0, [x := 0] \rangle$, and consider the value

$$\widehat{v}_1(s_0) = \sup_{\pi_1 \in \Pi_1} \sup \{w_1(\sigma) \mid \sigma \in Outcomes(s_0, \pi_1, \pi_2^*)\}.$$

There are two cases. If eventually player 1 plays forever Δ_0 in s_0 , player 1 obtains the value $-\infty$, as time does not progress, and player 1 is not blameless. If player 1, whenever at s_0 , eventually plays a^1 , then the value of the game to player 1 is $-c$. Hence, we have $\widehat{v}_1(s_0) = -c$. The analysis for player 2 is symmetrical. \square

Note that in the theorem we take sup, rather than inf as in (3.2), over the set of outcomes arising from the strategies. Hence, the theorem states that, even if the choice among actions is resolved in favor of the player trying to achieve the value, there is a game with a state s where $v_1(s) + v_2(s) \leq -2c < 0$. Moreover, in the theorem, the adversary strategies are fixed, again providing an advantage to the player trying to achieve the value.

The example of Figure 3.2, together with the above analysis, indicates that we cannot define the value of an average reward discrete-time game in a way that leads to determinacy, and enforces time progress. In fact, consider again the case in which player 2 plays always Δ_0 at s_0 . If, beyond some point, player 1 plays forever Δ_0 in s_0 , time does not progress, and the situation is symmetrical wrt. players 1 and 2: they both play forever Δ_0 . Hence, we must rule out this combination of strategies (either by assigning value $-\infty$ to the outcome, as we do, or by some other device). Once this is ruled out, the other possibility is that player 1, whenever in s_0 , eventually plays a^1 . In this case, time diverges, and the average value to player 1 is $-c$. As the analysis is symmetrical, the value to both players is $-c$, contradicting determinacy.

Chapter 4

Some Restrictions Apply

Our approach to finding the value of the infinite game is to recycle the technique used by Ehrenfeucht and Mycielski [10] in a game similar to the one under our consideration. Their insight was to recognize that an infinite path on a finite graph is necessarily composed of a series of loops. By creating a related finite game, they were able to show that there is a strategy in the infinite game that can achieve the same reward which is enforceable in the finite game.

The key to the relation between the infinite game and the finite game is the series of loops which make up the path in the infinite game. The finite game is played on the same game structure, but terminates as soon as a loop is closed (i.e., a state is repeated). Back in the infinite game, we can play using the strategy from the finite game, with one modification: every time a loop is closed, pretend that the loop portion never happened.

As an example, consider the graph shown in Figure 4.1. One possible outcome of the infinite game starts by visiting the states $ABCB$. At this point, the finite game would have terminated; by removing the loop (namely, CB), we are left with a path visiting AB . By moving to D , the path leads to the finite game $ABDEA$, and a new loop is closed. Note that the true path in the infinite game is now $ABCBDEA$, but is composed of two loops and represent two finite games.

In the remainder of this chapter, we will present proofs similar to those presented by Ehrenfeucht and Mycielski [10]. However, to complete the proof of Theorem 7, we must require that rewards be non-zero only when time advances; more specifically, simple loops with non-zero reward accumulated along their transitions and no *tick* events confound the proof. This was not a problem in the game

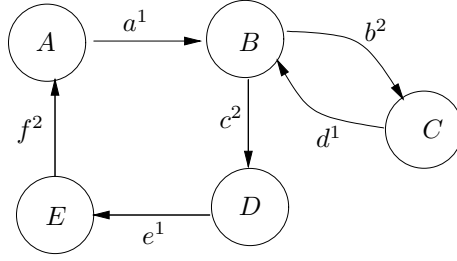


Figure 4.1: An example graph where the players take turns making plays. Actions are marked with a superscript indicating which player may play the action.

analyzed in [10] because their equivalent of “time” advanced on every move.

4.1 The Finite Game

The turn-based finite game is played on the same discrete-time game structure, and terminated as soon as a loop is closed; that is, when a state is first repeated. The termination condition assures a finite game, because the game is played over finitely many states; eventually, one must be repeated.

The final payoff received by player 1 (and lost by player 2) is equal to the sum of the rewards found in the loop, divided by the number of *tick* events found in the loop. If there are no *tick* events in the terminating loop, then the payoff is determined according to blame. If player 1 is blameless in the loop, then a payoff of $+\infty$ is assigned; if player 1 receives any blame, then a payoff of $-\infty$ is assigned.

Definitions pertaining this game have a “tf” superscript that stands for “turn-based finite”. A *maximal run* in the finite game is a 1-run $\sigma = s_0, \langle a_0^1, a_0^2, a_0^T \rangle, s_1, \dots, s_n$ such that s_n is the first state that is repeated in σ . Formally, n is the least number such that $s_n = s_j$, for some $j < n$. We set $loop(\sigma)$ to be the suffix of σ : $s_j, \langle a_j^1, a_j^2, a_j^T \rangle, \dots, s_n$. For $\pi_1 \in \Pi_1^t$, $\pi_2 \in \Pi_2^t$, and $s \in S$, we denote by $Outcomes^{tf}(s, \pi_1, \pi_2)$ the unique maximal run that starts in s and is consistent with π_1 and π_2 .

In the finite game, a maximal run σ ending with the loop λ is assigned the value of the infinite run obtained by repeating λ forever. Formally, $w_1^{tf}(\sigma) = w_1(\sigma \cdot \lambda^\omega)$, where λ^ω denotes the concatenation of numerably many copies of λ (and common states are included only once). The value assigned to a

strategy $\pi_1 \in \Pi_1^i$ and the value assigned to the whole game are defined as follows.

$$v_1^{\text{tf}}(s, \pi_1) = \inf_{\pi_2 \in \Pi_2^i} w_1^{\text{tf}}(\text{Outcomes}^{\text{tf}}(s, \pi_1, \pi_2))$$

$$v_1^{\text{tf}}(s) = \sup_{\pi_1 \in \Pi_1^i} v_1^{\text{tf}}(s, \pi_1).$$

4.2 Value Proofs

We now introduce definitions to show how to construct the loop-cutting strategy (called the “forgetful strategy” in [10]) for the infinite game. For a 1-run $\sigma = s_0, \langle a_0^1, a_0^2, a_0^T \rangle, s_1, \dots, s_n$, let $\text{firstloop}(\sigma)$ be the operator that returns the first simple loop (if any) occurring in σ . Similarly, let $\text{loopcut}(\sigma)$ be the operator that removes the first simple loop (if any) from σ . Formally, if σ is a simple run (i.e., it contains no loops) we set $\text{firstloop}(\sigma) = \varepsilon$ (the empty sequence), and $\text{loopcut}(\sigma) = \sigma$. Otherwise, let $k \geq 0$ be the smallest number such that $\sigma_j = \sigma_k$ for some $j < k$; we then set

$$\text{firstloop}(\sigma) = \sigma_j, \langle a_j^1, a_j^2, a_j^T \rangle, \dots, \sigma_{k-1}, \langle a_{k-1}^1, a_{k-1}^2, a_{k-1}^T \rangle, \sigma_k$$

$$\text{loopcut}(\sigma) = \sigma_0, \langle a_0^1, a_0^2, a_0^T \rangle, \dots, \sigma_j, \langle a_k^1, a_k^2, a_k^T \rangle, \dots, \sigma_n$$

We define the *quasi-segmentation* $QSeg(\sigma)$ to be the sequence of simple loops obtained by applying firstloop repeatedly to σ .

$$QSeg(\sigma) = \begin{cases} \varepsilon & \text{if } \text{firstloop}(\sigma) = \varepsilon \\ \text{firstloop}(\sigma) \cdot QSeg(\text{loopcut}(\sigma)) & \text{otherwise} \end{cases}$$

For an infinite run σ , we set $QSeg(\sigma) = \lim_{n \rightarrow \infty} QSeg(\sigma_{\leq n})$. Given a finite run σ , loopcut can only be applied a finite number of times before it converges to a fixpoint. We call this fixpoint the residual of the loop-cutting process, $\text{residual}(\sigma)$. Notice that for all runs σ , $\text{residual}(\sigma)$ is a simple path and therefore its length is bounded by $|S|$.

For simplicity, we developed the above definitions for 1-runs. The corresponding definitions of $\text{residual}(\sigma)$ and $QSeg(\sigma)$ for 2-runs σ are similar.

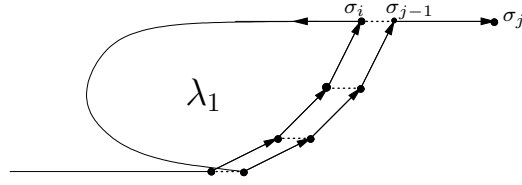


Figure 4.2: Nodes linked by dashed lines represent the same state of the game. This example closes the loop, then repeats some states of the loop before leaving the loop.

For all $i \in \{1, 2\}$ and all strategies $\pi \in \Pi_i^t$, we define the loop-cutting strategy $\tilde{\pi}$ as $\tilde{\pi}(\sigma) = \pi(\text{residual}(\sigma))$ for all $\sigma \in \text{FRuns}_i$. Intuitively, $\tilde{\pi}$ behaves like π until a loop is formed. At that point, $\tilde{\pi}$ forgets the loop, behaving as if the whole loop had not occurred.

Lemma 5. *Given a strategy, π_i , for player i in the finite game, then the residual of every run of an infinite game played with the loop-cutting strategy, $\tilde{\pi}_i$, is the prefix of some finite run consistent with π_i :*

Formally stated for player 1:

$$\forall \pi_1 \in \Pi_1^t. \forall \pi_2 \in \Pi_2^t. \forall \sigma \in \text{Outcomes}^{t\infty}(s, \tilde{\pi}_1, \pi_2). \forall k \geq 0$$

$$. \exists \pi'_2 \in \Pi_2^t. \exists \sigma' \in \text{Outcomes}^{tf}(s, \pi_1, \pi'_2). \exists \rho \mid \sigma' = \text{residual}(\sigma_{\leq k}) \cdot \rho$$

Proof. We prove the statement for player 1, as the proof is similar for player 2. The proof is by induction on the length of $QSeg(\sigma_{\leq k})$. If $QSeg(\sigma_{\leq k})$ is the empty sequence (i.e. $\sigma_{\leq k}$ contains no loops), the result is easily obtained, as $\tilde{\pi}_1$ coincides with π_1 until a loop is formed. So, we can take $\pi'_2 = \pi_2$ and obtain the conclusion.

On the other hand, suppose $QSeg(\sigma_{\leq k}) = \lambda_1, \dots, \lambda_n$. For simplicity, suppose $\lambda_1 \neq \lambda_2$. As illustrated in Figure 4.2, let σ_j be the first state after λ_1 that does not belong to λ_1 . Then, σ_{j-1} belongs to λ_1 and there is another index $i < j - 1$ such that $\sigma_i = \sigma_{j-1}$. So, the game went twice through σ_{j-1} and two different successors were taken. However, player 1 must have chosen the same move in σ_i and σ_{j-1} , as by construction $\tilde{\pi}_1(\sigma_{\leq i}) = \tilde{\pi}_1(\sigma_{\leq j-1})$. Therefore, the change must be due to a different choice of π_2 . It is easy to devise π'_2 that coincides with π_2 , except that λ_1 may be skipped, and at σ_i , the successor σ_j is chosen. We can then obtain a run $\rho = \text{outcome}^{t\infty}(s, \tilde{\pi}_1, \pi'_2)$ and an integer $k' \geq 0$ such that $QSeg(\rho_{\leq k'}) = \lambda_2, \dots, \lambda_n$ and $\text{residual}(\rho_{\leq k'}) = \text{residual}(\rho)$. The thesis is

obtained by applying the inductive hypothesis to ρ and k' . \square

Lemma 6. *Given a strategy, π_i , for player i in the finite game, and the corresponding loop-cutting strategy, $\tilde{\pi}_i$, all loops appearing in the quasi-segmentation of a run of the infinite game played with $\tilde{\pi}_i$ are terminating loops from some maximal run of the finite game played with π_i .*

Formally stated for player 1:

$$\begin{aligned} \forall \pi_1 \in \Pi_1^t. \forall \pi_2 \in \Pi_2^t. \forall \sigma \in \text{Outcomes}^{t\infty}(s, \tilde{\pi}_1, \pi_2) \\ \forall \lambda \in \text{QSeg}(\sigma). \exists \pi_2' \in \Pi_2^t. \exists \sigma' \in \text{Outcomes}^{tf}(s, \pi_1, \pi_2') \mid \lambda = \text{loop}(\sigma') \end{aligned}$$

Proof. We prove the statement for player 1, as the proof is similar for player 2. Let $\text{QSeg}(\sigma) = \lambda_1, \lambda_2, \dots$, and $\lambda = \lambda_j$ for any j . Let k be the largest integer such that $\text{QSeg}(\sigma_{\leq k}) = \lambda_1, \dots, \lambda_{j-1}$. Clearly, this means that the last edge of λ_j is $\sigma_k, \langle a_k^1, a_k^2, a_k^T \rangle, \sigma_{k+1}$. Also, at the end of $\text{residual}(\sigma_{\leq k})$ are all the edges of λ_j , except the last one. By applying Lemma 5 to σ and k , we obtain $\pi_2' \in \Pi_2^t$ and $\sigma' = \text{Outcomes}^{tf}(s, \pi_1, \pi_2')$ such that $\sigma' = \text{residual}(\sigma_{\leq k}) \cdot \rho$. To get the specific maximal run so that $\text{loop}(\sigma') = \lambda$, we define π_2'' to coincide with π_2' , except for $\pi_2''(\text{residual}(\sigma_{\leq k}) \cdot \langle a_{k+1}^1 \rangle) = a_{k+1}^2$. \square

Theorem 7. *Let π_1 be a strategy for player 1 in the finite game, so that $\tilde{\pi}_1$ is the corresponding loop-cutting strategy in the infinite game. Then $\tilde{\pi}_1$ is able to secure a value in the infinite game greater than or equal to the value which π_1 secures in the finite game.*

Formally:

$$\forall s \in S. \forall \pi_1 \in \Pi_1^t \mid v_1^{t\infty}(s, \tilde{\pi}_1) \geq v_1^{tf}(s, \pi_1).$$

Proof. Clearly, the theorem is trivially true when $v_1^{tf}(s, \pi_1) = -\infty$, and we therefore only consider the case where $v_1^{tf}(s, \pi_1) > -\infty$.

Let $\nu = v_1^{tf}(s, \pi_1)$. Fix a player 2 strategy $\pi_2 \in \Pi_2^t$, and let $\sigma = \text{Outcomes}^{t\infty}(s, \tilde{\pi}_1, \pi_2)$. Apply quasi-segmentation to σ , and let $\text{QSeg}(\sigma) = (\lambda_1, \lambda_2, \dots)$. Since, by Lemma 6, each of these loops must be a terminating loop possible under π_1 in the finite game, they each must have a reward of ν or greater.

Consider the case where an infinite number of loops from the list $(\lambda_1, \lambda_2, \dots)$ contain a *tick* event, indicating that time diverges in the infinite trace. Recall that, for this chapter, rewards are non-zero

only at moves that represent a *tick* event, so any terminating loops of the finite game that have no *tick* event must also have no reward. Therefore, we can write the following inequality which must hold for every loop occurring in the quasi-segmentation:

$$R(\lambda_i) \geq \nu \cdot T(\lambda_i)$$

since loops with no *tick* event evaluate to zero on both sides of the inequality, and loops with *tick* events are defined to have a value of $R(\lambda_i)/T(\lambda_i)$ (which must be greater or equal to the securable value, ν). and then sum this inequality over every loop. If, for all $n \geq 0$, we let $m_n = |QSeg(\sigma_{\leq n})|$, then we can write the sum as:

$$\liminf_{n \rightarrow \infty} \sum_{j=1}^{m_n} R(\lambda_j) \geq \nu \cdot \liminf_{n \rightarrow \infty} \sum_{j=1}^{m_n} T(\lambda_j)$$

To calculate the value over the entire infinite trace, we must also account for the reward and time accumulated in the residual portion of the trace. These are finite values, however, being counted as part of an infinite sum, and do not contribute to the final sum:

$$\begin{aligned} w_1^{\dagger \infty}(\sigma) &= \liminf_{n \rightarrow \infty} \frac{R(\sigma_{\leq n})}{T(\sigma_{\leq n})} \\ &= \liminf_{n \rightarrow \infty} \frac{R(\text{residual}(\sigma_{\leq n})) + \sum_{j=1}^{m_n} R(\lambda_j)}{T(\text{residual}(\sigma_{\leq n})) + \sum_{j=1}^{m_n} T(\lambda_j)} \\ &= \liminf_{n \rightarrow \infty} \frac{\sum_{j=1}^{m_n} R(\lambda_j)}{\sum_{j=1}^{m_n} T(\lambda_j)} \\ &\geq \nu \end{aligned}$$

Thus, when time diverges in the infinite game, every run has a value at least as large as the value of the finite strategy π_1 . Therefore, $v_1^{\dagger \infty}(s, \tilde{\pi}_1) \geq \nu$.

Now consider the case where only a finite number of loops from the list $(\lambda_1, \lambda_2, \dots)$ contain a *tick* event. There must then exist some point, $k \geq 0$, in the run (and some corresponding loop in the quasi-segmentation), so that there are no *tick* events beyond σ_k . Consider a loop λ_j entirely occurring after σ_k . Obviously λ_j contains no time steps. Moreover, by Lemma 6, λ_j is a terminating loop for a maximal run ρ in the finite game under π_1 . Since $v_1^{\text{tf}}(s, \pi_1) > -\infty$, it must be the case that

$w_1^{\text{tf}}(\rho) = +\infty$. Consequently, $\text{blameless}_1(\rho)$ must be true, and therefore player 1 is blameless in all edges in λ_j .

Now, let $k' \geq 0$ be such that each state (and edge) after $\sigma_{k'}$ will eventually be part of a loop of $QSeg(\sigma)$. Let $k'' = \max\{k, k'\}$. Then, all edges that occur after k'' will eventually be part of a loop where player 1 is blameless. Consequently, k'' is a witness to the fact that $\text{blameless}_1(\sigma)$, and therefore $w_1^{\text{t}\infty}(\sigma) = +\infty \geq \nu$.

□

Theorem 8. For all $s \in S$ and $\pi_2 \in \Pi_2^{\text{t}}$, it holds $v_1^{\text{t}\infty}(s, \tilde{\pi}_2) \leq v_1^{\text{tf}}(s, \pi_2)$.

Proof. Let $\nu = v_1^{\text{tf}}(s, \pi_2)$. Similarly to Theorem 7, we can rule out the case $\nu = +\infty$ as trivial. Fix a player 1 strategy π_1 , and let $\sigma = \text{Outcomes}^{\text{t}\infty}(s, \pi_1, \tilde{\pi}_2)$. We show that $w_1^{\text{t}\infty}(\sigma) \leq \nu$. If time diverges on σ , the proof is similar to the analogous case in Theorem 7. Otherwise, let $k \geq 0$ be such that no time steps occur in σ after σ_k . Consider a loop $\lambda \in QSeg(\sigma)$, entirely occurring after σ_k . Obviously λ contains no time steps. Moreover, by Lemma 6, λ is a terminating loop for a maximal run ρ in the finite game under π_1 . Since $v_1^{\text{tf}}(s, \pi_1) < +\infty$, it must be $w_1^{\text{tf}}(\rho) = -\infty$. Consequently, it holds $\neg \text{blameless}_1(\rho)$ and in particular player 1 is blamed in some edge of λ . This shows that $\neg \text{blameless}_1(\sigma)$, and consequently $w_1^{\text{t}\infty}(\sigma) = -\infty \leq \nu$.

□

Theorem 9. For all $s \in S$, $v_1^{\text{t}\infty}(s) = v_1^{\text{tf}}(s)$.

Proof. Theorems 7 and 8 show that the infinite game is no harder than the finite one, for both players. Since this game is finite and turn-based, the minmax principle tells us that, for all $s \in S$, it holds:

$$\sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} w_1^{\text{tf}}(\text{Outcomes}^{\text{tf}}(s, \pi_1, \pi_2)) = \inf_{\pi_2 \in \Pi_2} \sup_{\pi_1 \in \Pi_1} w_1^{\text{tf}}(\text{Outcomes}^{\text{tf}}(s, \pi_1, \pi_2)).$$

and we see that

$$v_1^{\text{tf}}(s) = \sup_{\pi_1 \in \Pi_1^{\text{t}}} v_1^{\text{tf}}(s, \pi_1) = \inf_{\pi_2 \in \Pi_2^{\text{t}}} v_1^{\text{tf}}(s, \pi_2)$$

Referring back to Theorem 7 and the definition of value in the turn-based infinite game, we see

$$v_1^{\text{t}\infty}(s) = \sup_{\pi_1 \in \Pi_1^{\text{t}}} v_1^{\text{t}\infty}(s, \pi_1) \geq \sup_{\pi_1 \in \Pi_1^{\text{t}}} v_1^{\text{t}\infty}(s, \tilde{\pi}_1) \geq \sup_{\pi_1 \in \Pi_1^{\text{t}}} v_1^{\text{tf}}(s, \pi_1) = v_1^{\text{tf}}(s)$$

but Theorem 8 tells us

$$\inf_{\pi_2 \in \Pi_2^t} v_1^{t\infty}(s, \tilde{\pi}_2) \leq \inf_{\pi_2 \in \Pi_2^t} v_1^{tf}(s, \pi_2) = v_1^{tf}(s)$$

and clearly,

$$\inf_{\pi_2 \in \Pi_2^t} v_1^{t\infty}(s, \pi_2) \leq \inf_{\pi_2 \in \Pi_2^t} v_1^{t\infty}(s, \tilde{\pi}_2)$$

Again appealing to the definition of value in the turn-based game, and doing several substitutions, we find:

$$\begin{aligned} v_1^{t\infty}(s) &= \sup_{\pi_1 \in \Pi_1^t} v_1^{t\infty}(s, \pi_1) = \sup_{\pi_1 \in \Pi_1^t} \inf_{\pi_2 \in \Pi_2^t} w_1^{t\infty}(Outcomes^{t\infty}(s, \pi_1, \pi_2)) \\ &\leq \inf_{\pi_2 \in \Pi_2^t} \sup_{\pi_1 \in \Pi_1^t} w_1^{t\infty}(Outcomes^{t\infty}(s, \pi_1, \pi_2)) \\ &\leq \inf_{\pi_2 \in \Pi_2^t} v_1^{t\infty}(s, \pi_2) \leq v_1^{tf}(s) \end{aligned}$$

Since we have shown both $v_1^{t\infty}(s) \geq v_1^{tf}(s)$ and $v_1^{t\infty}(s) \leq v_1^{tf}(s)$, the two values must be equal. □

4.3 Memoryless Strategies

By following the “forgetful game” construction and proofs used by [10], we can derive a similar result on the existence of memoryless strategies for both players. The proof depends on the fact that the value of forgetful game is the same as the turn-based finite game (and hence, the same as the infinite game, from Theorems 9 and 3), and follows the same inductive steps as provided in [10].

To this end, following [10], we define a modified version of the finite game, which is still played

on the same discrete-time game structure. For a state $s \in S$, the s -forgetful game is played as the finite game, except that the first time s is encountered while playing, the history up to s is forgotten for the purpose of checking game termination. Formally, a maximal run in the s -forgetful game is a finite run $\sigma = s_0, \langle a_0^1, a_0^2, a_0^T \rangle, s_1, \dots, s_n$ such that:

- either s does not occur in σ and s_n is the first state that is repeated in σ ,
- or σ_k is the first occurrence of s in σ , and s_n is the first state that is repeated in $\sigma_{\geq k}$.

We define $loop(\sigma)$ to be the final loop λ of σ , and we set $w_1^{\text{tf},s}(\sigma) = w_1^{\text{tf}}(\sigma \cdot \lambda^\omega)$. We denote by $Outcomes^{\text{tf},s}(t, \pi_1, \pi_2)$ the unique maximal run in the s -forgetful game that starts in t and is consistent with π_1 and π_2 . The value assigned to a strategy $\pi_1 \in \Pi_1^t$ and the value assigned to the whole game are defined as usual:

$$v_1^{\text{tf},s}(t, \pi_1) = \inf_{\pi_2 \in \Pi_2^t} w_1^{\text{tf},s}(Outcomes^{\text{tf},s}(t, \pi_1, \pi_2)); \quad v_1^{\text{tf},s}(t) = \sup_{\pi_1 \in \Pi_1^t} v_1^{\text{tf},s}(t, \pi_1).$$

Lemma 10. *For all $s, t \in S$, $v_1^{\text{tf}}(t) = v_1^{\text{tf},s}(t)$.*

Proof. SKETCH: By considering whether a run goes through state s or not, we show that $w_1^{\text{tf},s}(\sigma)$ is always greater than or equal to $v_1^{\text{tf}}(t)$. But $v_1^{\text{tf},s}(t)$ is the least enforceable $w_1^{\text{tf},s}(\sigma)$, and must then also be greater than or equal to $v_1^{\text{tf}}(t)$.

- ⟨1⟩1. Let $\pi_1 \in \Pi_1^t$ be the optimal strategy for player 1 in the finite game, starting at state t .
- ⟨1⟩2. Let $\nu = v_1^{\text{tf}}(t)$, which is the value that optimal π_1 can enforce.
- ⟨1⟩3. By Theorems 3 and 9, we also have $\nu = v_1^{t,\infty}(t) = v_1(t)$.
- ⟨1⟩4. Fix an arbitrary strategy $\pi_2 \in \Pi_2^t$ for player 2, and let $\sigma = Outcomes^{\text{tf},s}(t, \tilde{\pi}_1, \pi_2)$. Note that we use the loop-cutting strategy $\tilde{\pi}_1$ in the s -forgetful game.
- ⟨1⟩5. We then have $w_1^{\text{tf},s}(\sigma) \geq v_1^{\text{tf}}(t)$.
 - ⟨2⟩1. There are two cases to consider: when σ passes through state s , and when it does not.
 - ⟨2⟩2. First consider the case where σ never passes through s .

$$w_1^{\text{tf},s}(\sigma) = w_1^{\text{tf}}(\sigma) \quad \text{rewards are the same when } s \text{ does not occur.}$$

$$w_1^{\text{tf}}(\sigma) \geq \nu \quad \text{since } \pi_1 \text{ is optimal}$$

$$w_1^{\text{tf},s}(\sigma) \geq v_1^{\text{tf}}(t) \quad \text{by transitivity and substitution}$$

⟨2⟩3. The other possible case is where σ passes through s . In this case, we observe that $w_1^{\text{tf},s}(\sigma)$ is determined by the final loop $\lambda = \text{loop}(\sigma)$.

⟨2⟩4. There is a $\pi_2' \in \Pi_2^t$ such that $\sigma' = \text{Outcomes}^{t,\infty}(t, \tilde{\pi}_1, \pi_2')$ ends with the sequence λ^ω . This is because the $\tilde{\pi}_1$ forgets every loop closed, and tries to repeat it. If entering the loop is the decision of player 2, we choose π_2' to repeatedly take the loop.

⟨2⟩5. Now we can analyze $w_1^{\text{tf},s}(\sigma)$:

$$\begin{aligned} w_1^{\text{tf},s}(\sigma) &= w_1^{t,\infty}(\sigma \cdot \lambda^\omega) && \text{definition of reward in the } s\text{-forgetful game.} \\ &= w_1^{t,\infty}(\sigma') && \text{Definition of } \sigma'. \\ &\geq v_1^{t,\infty}(t, \tilde{\pi}_1) && \text{definition of } v_1^{t,\infty} \\ &\geq v_1^{\text{tf}}(t, \pi_1) && \text{by Theorem 7} \\ &\geq v_1^{\text{tf}}(t) && \text{since } \pi_1 \text{ is the optimal strategy from } t \text{ in the finite game} \end{aligned}$$

⟨1⟩6. Since $w_1^{\text{tf},s}(\sigma) \geq v_1^{\text{tf}}(t)$ for an arbitrary π_2 ,

$$v_1^{\text{tf},s}(t, \tilde{\pi}_1) = \inf_{\pi_2 \in \Pi_2^t} w_1^{\text{tf},s}(\text{Outcomes}^{t,\text{tf},s}(t, \tilde{\pi}_1, \pi_2)) \geq v_1^{\text{tf}}(t)$$

⟨1⟩7. Strategy $\tilde{\pi}_1$ might not be the optimal strategy in the s -forgetful game, so

$$v_1^{\text{tf},s}(t) = \sup_{\pi' \in \Pi_1^t} v_1^{\text{tf},s}(t, \pi') \geq v_1^{\text{tf},s}(t, \tilde{\pi}_1) \geq v_1^{\text{tf}}(t)$$

⟨1⟩8. Conversely, we can prove $v_1^{\text{tf},s}(t) \leq v_1^{\text{tf}}(t)$ by a symmetrical argument which exchanges the roles of the two players.

⟨1⟩9. Therefore, we conclude $v_1^{\text{tf},s}(t) = v_1^{\text{tf}}(t)$.

□

Theorem 11. *For all $i \in \{1, 2\}$, and $t \in S$, there exists a memoryless optimal strategy for player i . Formally, there exists $\pi_i \in \Pi_i$ such that $v_1(t, \pi_i) = v_1(t)$ and π_i is memoryless.*

Proof. We prove the statement for $i = 1$, as the other case is symmetrical. We develop our arguments for the finite turn-based game, as the conclusion for the original timed game follows from Theorem 7. We proceed by complete induction on $n = \sum_{s \in S} |M_1(s)| - |S|$. When $n = 0$, only one move is available to player 1 at each state and there is obviously a memoryless optimal strategy.

Suppose that the statement is true for all integers up to $n \geq 0$, and consider the situation where $\sum_{s \in S} |M_1(s)| - |S| = n + 1$; in this case, there is at least one state, s , where $|M_1(s)| > 1$. In this game structure (call it \mathcal{G}), we can play the s -forgetful game, and Lemma 10 states that the game value will be the same as that of the normal turn-based finite game, $v_1^{\text{tf},s}(\mathcal{G}, t) = v_1^{\text{tf}}(\mathcal{G}, t)$, which can be ensured by some strategy, π_1 . By Theorems 3 and 9, $v_1^{\text{tf}}(\mathcal{G}, t) = v_1^{\text{t}\infty}(\mathcal{G}, t) = v_1(\mathcal{G}, t)$, so strategy π_1 is able to ensure $v_1(\mathcal{G}, t)$ in the s -forgetful game. Let $\nu = v_1(\mathcal{G}, t)$.

Clearly, π_1 only plays one move at state s in the s -forgetful game (call this move $a \in M_1(s)$), because any return to s would end the game. Therefore, we can set $\Gamma'_1(s) = \{a\}$ (and leave the enabled moves for all other states the same) as part of a new discrete-time game structure

$$\mathcal{G}' = (S, Acts_1, Acts_2, \Gamma'_1, \Gamma_2, \delta, r)$$

and still be consistent with how π_1 desires to play the s -forgetful game.

Since \mathcal{G}' is consistent with π_1 in the s -forgetful game, π_1 is able to ensure at least the same value as in \mathcal{G} , namely ν . Moreover, since \mathcal{G}' contains less choices for player 1, no player 1 strategy can achieve a value greater than ν . So, $v_1^{\text{tf},s}(\mathcal{G}', t) = \nu$. By again appealing to Lemma 10 and Theorems 3 and 9, we find that $\nu = v_1^{\text{tf},s}(\mathcal{G}', t) = v_1^{\text{tf}}(\mathcal{G}', t) = v_1^{\text{t}\infty}(\mathcal{G}', t) = v_1(\mathcal{G}', t)$.

To summarize, we have constructed a reduced game structure, \mathcal{G}' , with value $v_1(\mathcal{G}', t) = \nu$. Notice that our inductive hypothesis applies to \mathcal{G}' , since $\sum_{s \in S} |M'_1(s)| - |S| \leq n$. Therefore, there exists some memoryless strategy, π_1^* , which is able to ensure ν when played on \mathcal{G}' . But then π_1^* is also able to ensure ν in the original \mathcal{G} since all moves that π_1^* might play exist in \mathcal{G} . This demonstrates the desired conclusion for player 1. □

Chapter 5

Conclusions

5.1 Future Work

We have presented a generalization of the long-run average work done by Ehrenfeucht and Mycielski [10], where time is advanced by a specific action taken by the players in unison. A notable restriction in our proof is that non-zero rewards for actions are only permitted when the action is Δ_1 (that is, non-zero rewards are accumulated only when time is advancing).

Removing the Restriction

Jackpots are a feature of some graphs, where there is a reward for an immediate action, which result in a counter-intuitive strategy for winning. In Figure 5.1, we see a timed automaton where reward can be accrued in only two ways: as time advances in state s_0 , or when the transition from s_0 to s_1 is taken. The best strategy, according to the finite game of the last chapter, is for both players to advance time at s_0 . Player 2 has a better strategy, though — sometimes advancing time at s_0 and sometimes playing action a^2 . By repeatedly playing a^2 , player 2 can drive the total accumulated reward downward to any level she selects. Judicious alternation between advancing time and ever decreasing accumulated rewards leads to a reward of ∞ for player 2, since the definition of the long-run average reward includes a “lim inf” in the definition:

$$\bar{r}(\sigma') = \liminf_{n \rightarrow \infty} \frac{R(\sigma'_{\leq n})}{T(\sigma'_{\leq n})}.$$

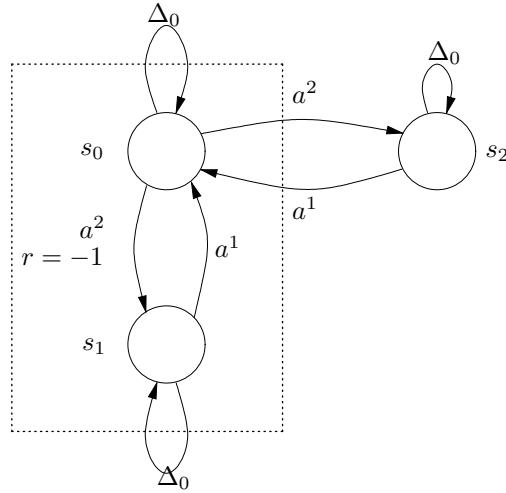


Figure 5.1: This graph shows a game which is well-formed, but player 2 can get an arbitrary amount of reward between *tick* events. Unspecified guards and invariants are taken to be *true* and always satisfied; unspecified rewards are assumed to be zero. The boxed region on the left is a 2-jackpot which can generate this large reward.

A jackpot, then, is any loop without a *tick* event that has a non-zero reward, and one of the players does not have a strategy to avoid the loop. It is only useful, however, if the player that benefits can also cause a *tick* event to occur, and then return to the jackpot loop.

As in [10], we solved for the long-run average reward in our framework by showing a mapping to a simpler finite game where the game terminates as soon as a cycle is formed in the moves of the players. This same finite game does not work for cases such as Figure 5.1 because a second cycle is necessary to determine whether one of the opponents can periodically force time to advance in a way that gives them the advantage in the “lim inf” calculation. We believe that a different finite game might exist which has equivalent value to the infinite game, but more research is required to discover that finite game and create valid proofs.

Other Directions

Many works in priced timed games assume that time is a continuous variable, rather than discrete. Due to the finite nature of the timed automata specification, it might be possible to allow continuous time in our own framework. This would allow each player to specify the amount of time they would

like to wait, and allows more complex interactions between the players.

Another interesting variation on the framework would be to incorporate probabilistic weighting on actions. This would allow us to calculate a “worst expected performance”, rather than a “worst case performance.” Such modeling would be extremely useful in hybrid electric cars, which currently always charge batteries when the internal combustion engine is on; on an uphill which will be immediately followed by a downhill (as in a mountain pass), any “extra” charge that is generated on the downhill is wasted as soon as the batteries are at full capacity. Through the use GPS and topographic maps, a car which could predict a likely downhill would be able to expend all the battery energy on the uphill and then recover the charge on the downhill, reducing gas consumption.

5.2 Summary

Real-time embedded systems often have strict performance goals that must be adhered to under any arbitrary environment. We have constructed a discrete-time version of priced timed games that is suitable for modeling many problems in this domain, and discuss some variations on well-formedness criteria. We present proofs that a finite game, which terminates when a cycle is formed, has the same securable value to each player as in the infinite game, and that there is a positional strategy which achieves the value. These proofs represent a generalization of the long-run average games of Ehrenfeucht and Mycielski [10] to solve for strategies that achieve the worst-case optimal performance in this framework.

Appendices

Appendix A

Additional Well-Formedness Proofs

These are some proofs that use the well-formedness condition of chapter 3 to show the correspondence between the finite and infinite games. These are analogous to the proofs based on the value of the finite game to a player, since a well-formed graph implies that there is a strategy to achieve a value greater than $-\infty$ for both players.

Lemma 12. *Given a strategy, π_i^* , which is a witness for well-formedness in the finite game for player i , we consider the infinite strategy, $\tilde{\pi}_i^*$. In all loops created by $\tilde{\pi}_i^*$ when playing against some opponent strategy, $\pi_{\sim i}$, either time must diverge, or player i is always not to blame.*

Proof. SKETCH: We show that the lemma must be true for the first loop made in an infinite game. By the construction of $\tilde{\pi}_i^*$, there are alternative traces where the second loop of the original trace becomes the first loop of the new trace. This recursive application of the lemma continues until all possible loops are covered. We make the case for player 1, but the arguments apply equally well to player 2.

(1)1. We are given a well-formed finite game, and π_1^* and $\tilde{\pi}_1^*$ strategies for the game, satisfying the following:

$$\text{WF}_1(\mathcal{G}) \implies \exists \pi_1^* \in \Pi_1^t. \forall \pi_2 \in \Pi_2^t. \forall \sigma \in \text{Outcomes}^{\text{tf}}(s, \pi_1^*, \pi_2)$$

$$| \sigma \models \square \diamond \text{tick} \vee \sigma \models \diamond \square \neg \text{bl}_1$$

$$\tilde{\pi}_1^*(s_0, \langle a_0^1, a_0^2, a_0^T \rangle, \dots, s_k) = \pi_1^*(\text{residual}(s_0, \langle a_0^1, a_0^2, a_0^T \rangle, \dots, s_k))$$

- ⟨1⟩2. Let $\sigma_{\leq r-1} = s_0, \langle a_0^1, a_0^2, a_0^T \rangle, s_1, \dots, s_{r-1}$ be a partial play of the infinite game where player 1 has been using strategy π_1^* and player 2 has been using strategy π_2 , and the first loop to occur has just been closed.
- ⟨1⟩3. Since a loop has just been closed, then we know that there is a $j < r - 1$ such that $s_j = s_{r-1}$, and the run of the game played so far can now be written as:

$$\sigma_{\leq r-1} = s_0, \langle a_0^1, a_0^2, a_0^T \rangle, s_1, \dots, s_{j-1}, \langle a_{j-1}^1, a_{j-1}^2, a_{j-1}^T \rangle, s_j, \dots, s_{r-1}$$

- ⟨1⟩4. Up until this point, $\tilde{\pi}_1^*(\sigma_{\leq i}) = \pi_1^*(\sigma_{\leq i})$, for all i up to and including $i = r - 1$. Therefore, $\sigma_{\leq r-1}$ can be described as a play of the finite game, using strategy π_1^* .
- ⟨1⟩5. Since π_1^* is a witness to the well-formedness condition for player 1 in the finite game, we know that:

$$(s_0, \langle a_0^1, a_0^2, a_0^T \rangle, \dots, s_{j-1}, \langle a_{j-1}^1, a_{j-1}^2, a_{j-1}^T \rangle)(s_j, \dots, s_{r-1})^\omega \models \square \diamond tick \vee \diamond \square \neg bl_1$$

- ⟨1⟩6. Both the time divergence and blameless conditions describe conditions that apply to the infinite path in a way which allows us to discard any finite prefix and still satisfy the conditions:

$$(s_j, \dots, s_{r-1})^\omega \models \square \diamond tick \vee \diamond \square \neg bl_1$$

Thus, we have demonstrated that in the first loop created by a play of $\tilde{\pi}_1^*$ against π_2 , it must be the case that there is a *tick* event within the loop, or player 1 is not to blame anywhere within the loop.

- ⟨1⟩7. To extend this result to further loops in the run, set us now consider the case where one further move has been played: $\sigma_{\leq r} = s_0, \langle a_0^1, a_0^2, a_0^T \rangle, \dots, s_r$. The two cases to consider are whether it is player 1's play that dictates the move to s_r , or whether it is player 2's play.

- ⟨2⟩1. Case “player 1's choice”: In this case, we know that the finite strategy must have provided

the move $a_{r-1}^1 = a_{r-1}^T$, so:

$$\begin{aligned} a_{r-1}^T &= \tilde{\pi}_1^*(\sigma_{\leq r-1}) \\ &= \pi_1^*(\text{residual}(\sigma_{\leq r-1})) \\ &= \pi_1^*(s_0, \langle a_0^1, a_0^2, a_0^T \rangle, \dots, s_j) \end{aligned}$$

But this means that $s_0, \langle a_0^1, a_0^2, a_0^T \rangle, \dots, s_j, s_r$ is a possible trace of the finite game with at most one loop. Thus, we can reapply steps $\langle 1 \rangle 2$ through $\langle 1 \rangle 6$ to

$$s_0, \langle a_0^1, a_0^2, a_0^T \rangle, \dots, s_j, s_r$$

to show that the first loop that appears in this new run must also satisfy the well-formedness condition for player 1.

$\langle 2 \rangle 2$. Case “player 2’s choice”: By player 2’s choice, we mean that $a_{r-1}^1 \neq a_{r-1}^T$; the move to s_r is due to a play of player 2. Using a similar set of equalities as in the other case, we can conclude

$$\pi_1^*(s_0, \langle a_0^1, a_0^2, a_0^T \rangle, \dots, s_j) = a_j^1 = a_{r-1}^1 \neq a_{r-1}^T$$

In fact, we already know that a_j^1 must be the move describing the edge $s_j \langle a_j^1 \rangle s_{j+1}$, so that we have:

$$\pi_2(s_0, \langle a_0^1, a_0^2, a_0^T \rangle, \dots, s_{r-1} \langle a_{r-1}^1 \rangle) = a_{r-1}^2$$

where a_{r-1}^2 is the move that defines the edge $s_{r-1} \langle a_{r-1}^2 \rangle s_r$. Since a_{r-1}^2 is a valid move for player 2 at the current state of the game, it must have also been a valid move when the game was at the same state earlier — namely, s_j . Therefore, there must exist some strategy, π'_2 , such that it is the same as π_2 except that

$$\pi'_2(s_0, \langle a_0^1, a_0^2, a_0^T \rangle, \dots, s_j \langle a_j^1 \rangle) = a_{r-1}^2$$

The existence of this alternative strategy for player 2 then admits the existence of a possible run of the finite game which looks like

$$\sigma' = s_0, \langle a_0^1, a_0^2, a_0^T \rangle, \dots, s_j \langle a_j^1, a_{r-1}^2, a_{r-1}^2 \rangle, s_r$$

where there is only at most one loop in the run (terminating at s_r). As in the other case, we can reapply steps $\langle 1 \rangle 2$ through $\langle 1 \rangle 6$ to σ' to show that the first loop that appears in this new run must also satisfy the well-formedness condition for player 1.

$\langle 1 \rangle 8$. By repeating this argument for each possible trace and every loop in each trace, we end up at the result that all loops must satisfy the well-formedness condition for player 1.

□

Theorem 13. *Let π_1 be a strategy for player 1 in the finite game, and $\tilde{\pi}_1$ be an infinite strategy for player 1 which is formed using the loop-cutting method. If π_1 is a witness to well-formedness of the finite game, then $\tilde{\pi}_1$ is a witness to well-formedness in the infinite game.*

Proof. SKETCH: Every loop in the quasi-segmentation of an infinite run is a possible terminating loop from the finite game. By reasoning about the properties of these loops, we obtain the result.

$\langle 1 \rangle 1$. By hypothesis, strategy π_1 (and by extension, strategy $\tilde{\pi}_1$) generates only loops which individually satisfy the well-formedness condition (follows from lemma 12), no matter what strategy, $\pi_2 \in \Pi_2^t$, that player 2 may employ.

$\langle 1 \rangle 2$. For all possible runs resulting from $\tilde{\pi}_1$ playing against π_2 , we can describe the run as a set of loops which make it up:

$$\forall \sigma \in \text{Outcomes}^{t \infty}(s, \tilde{\pi}_1, \pi_2) \mid QSeg(\sigma) = (\lambda_1, \lambda_2, \dots)$$

$\langle 1 \rangle 3$. Consider the set of loops in the quasi-segmentation. Either there are infinitely many loops in which time diverges, or there are a finite number of such loops.

$\langle 2 \rangle 1$. Case I: If there are infinitely many loops in which time diverges, then it must be the case that $\sigma \models \Box \diamond tick$ as well, and therefore $WFC_1(\sigma)$ is true.

⟨2⟩2. Case II: Since each loop satisfies the well-formedness condition, but only a finite number let time diverge, then there must exist some point beyond which time never diverges and instead each loop must have player 1 be blameless (because they are well-formed loops): $\exists k. \forall i > k \mid \lambda_i \models \Box \neg tick \wedge \Box \neg bl_1$. Since every loop after this point satisfies $\Box \neg bl_1$ and there are only a finite number of steps which might not belong to any loop, then $\sigma \models \Diamond \Box \neg bl_1$, and therefore $WFC_1(\sigma)$ is true.

⟨1⟩4. For every possible run generated by an arbitrary strategy π_2 and the loop-cutting strategy $\tilde{\pi}_1$, we find that the run must satisfy the well-formedness condition. But this means that $\tilde{\pi}_1$ must be a witness to well-formedness in the infinite game.

□

Corollary 14. *If a game structure is well-formed in the finite version of the game, then it is also well-formed in the infinite version of the game.*

Proof. By hypothesis, the game is well-formed, so by definition there must exist a strategy of the finite game, π_1^* , which is a witness to the well-formedness for player 1. Construct the infinite strategy $\tilde{\pi}_1^*$ from π_1^* using the loop-cutting method, and apply Theorem 13. Thus, $\tilde{\pi}_1^*$ is a witness to well-formedness for player 1 in the infinite game. Similarly, there must exist a π_2^* and a constructed $\tilde{\pi}_2^*$ which are witnesses for player 2 of the well-formedness of the finite and infinite versions of the game. By definition, the infinite game must then be well-formed since it is well-formed for both players. □

Bibliography

- [1] R. Alur, M. Bernadsky, and P. Madhusudan. Optimal reachability for weighted timed games. In *Proc. 31st Int. Colloq. Aut. Lang. Prog.*, volume 3142 of *Lect. Notes in Comp. Sci.* Springer-Verlag, 2004.
- [2] R. Alur and D.L. Dill. A theory of timed automata. *Theor. Comp. Sci.*, 126:183–235, 1994.
- [3] R. Alur and T.A. Henzinger. Modularity for timed and hybrid systems. In *CONCUR 97: Concurrency Theory. 8th Int. Conf.*, volume 1243 of *Lect. Notes in Comp. Sci.*, pages 74–88. Springer-Verlag, 1997.
- [4] 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.
- [5] P. Bouyer, E. Brinksma, and K.G. Larsen. Staying alive as cheaply as possible. In *Proc. of 7th Intl. Workshop on Hybrid Systems: Computation and Control (HSCC)*, volume 2993 of *Lect. Notes in Comp. Sci.*, pages 203–218. Springer-Verlag, 2004.
- [6] P. Bouyer, F. Cassez, E. Fleury, and K.G. Larsen. Optimal strategies in priced timed game automata. In *Found. of Software Technology and Theoretical Comp. Sci.*, volume 3328 of *Lect. Notes in Comp. Sci.*, pages 148–160. Springer-Verlag, 2004.
- [7] A. Church. Logic, arithmetics, and automata. In *Proc. International Congress of Mathematicians, 1962*, pages 23–35. Institut Mittag-Leffler, 1963.
- [8] L. de Alfaro, M. Faella, T.A. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. In *CONCUR 03: Concurrency Theory. 14th Int. Conf.*, volume 2761 of *Lect. Notes in Comp. Sci.*, pages 144–158. Springer-Verlag, 2003.
- [9] L. de Alfaro, T.A. Henzinger, and M. Stoelinga. Timed interfaces. In *Proceedings of the Second International Workshop on Embedded Software (EMSOFT 2002)*, volume 2491 of *Lect. Notes in Comp. Sci.*, pages 108–122. Springer-Verlag, 2002.
- [10] A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. *Int. Journal of Game Theory*, 8(2):109–113, 1979.
- [11] T.A. Henzinger, B. Horowitz, and R. Majumdar. Rectangular hybrid games. In *CONCUR 99: Concurrency Theory. 10th Int. Conf.*, volume 1664 of *Lect. Notes in Comp. Sci.*, pages 320–335. Springer-Verlag, 1999.
- [12] Research Triangle Institute. The economic impacts of inadequate infrastructure for software testing. Planning Report 02-3, National Institute of Standards and Technology, June 2002.

- [13] H. Klauck. Algorithms for parity games. In *Automata, Logics, and Infinite Games*, volume 2500 of *Lect. Notes in Comp. Sci.*, pages 107–129. Springer, 2001.
- [14] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *Proc. of 12th Annual Symp. on Theor. Asp. of Comp. Sci.*, volume 900 of *Lect. Notes in Comp. Sci.*, pages 229–242. Springer-Verlag, 1995.
- [15] A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symp. Found. of Comp. Sci.*, pages 46–57. IEEE Computer Society Press, 1977.
- [16] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proceedings of the 16th Annual Symposium on Principles of Programming Languages*, pages 179–190. ACM Press, 1989.
- [17] P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *IEEE Transactions on Control Theory*, 77:81–98, 1989.
- [18] U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theor. Comp. Sci.*, 158:343–359, 1996.