Chapter 5

Shapley's Game and Church's Problem

5.1 Introduction

In this chapter, we study graph games played by two players. The games starts from start position p_0 with Player 1 moving to position p_1 along an edge. Then Player 2 moves to position p_2, followed by Player 1 making a move, and so on. In this way, an infinite play $p = p_0 p_1 p_2 \ldots$ is constructed. At the "end," Player 2 pays to Player 1 the amount $P(p)$ where P is a payoff function. The games are perfect information zero-sum games. So Player 2 tries to minimize the payoff to Player 1, and Player 1 tries to maximize it.

A strategy for a player is a rule which tells him how to play. A game has a value v when Player 1 has a strategy which assures him a payoff of at least v and Player 2 has a strategy which guarantees her a loss no greater than v. The strategy which assures a payoff of v for Player 1 is his optimal strategy, and the strategy which guarantees for Player 2 a loss no greater than v is her optimal strategy. By "solving a game," we mean determining the value of the game, and the optimal strategies for the players. We study two classes of games: the payoff games and games on ω-automata.

Payoff games are a special case of stochastic games introduced by Shapley [38] and extensively studied in the operations research community since then [36]. In the

payoff game, each position w has some reward $r(w)$. The players move from position to position, creating the play $p = p_0 p_1 p_2 \ldots$. The payoff is $P(p) = \sum_k r(p_k)$. Player 1 chooses his moves to maximize the payoff; Player 2 chooses hers to minimize it.

An ω-automaton is a finite state automaton which accepts infinite sequences [40, 23]. In a game on ω-automaton, the objective of Player 1 is to create a play p which is accepted by the automaton; Player 2 tries to create a play which is rejected. Solving a game on an ω-automaton is sometimes referred to as Church's problem who posed the question as a synthesis problem for digital circuits [8]. Games on ω-automata have been extensively studied in the logic and computer science communities [35, 40, 23].

In this chapter, we relate games on ω-automata with the payoff games. We show that a game on an ω-automaton with the chain acceptance condition can be solved as a payoff game. The chain acceptance condition can express any ω-regular language. As a result, any game on an ω-automaton can be solved as a payoff game. It is known that solving the model checking problem for propositional μ-calculus is polynomially equivalent to solving the chain game [13]. Hence, we get a new method for model checking μ-calculus using algorithms for solving payoff games.

In Section 5.2, we introduce games played on graphs. In Section 5.3, we discuss games on ω-automata. We review results on ω-automata games and show that such games do not have a value when restricted to positional strategies. In Section 5.4, we begin discussing payoff games. We study both finite and infinite games. For the infinite game, we study the discounted payoff game and the mean payoff game. We prove that both games have a value and optimal positional strategies. We then introduce the successive approximation and the policy iteration algorithms for solving payoff games. In Section 5.5, we relate the chain games with the payoff games. We show that the results for payoff games also provide answers to classical problems such as Church's solvability and synthesis problem. As a result of the relationship between chain games and payoff games, we can use the policy iteration algorithm to model check μ-calculus formulae.

5.2 Games on Graphs

A game graph is a directed bipartite graph $G = (V, E)$. The positions in the game are the vertices $V = V_1 \cup V_2$ where V_1 and V_2 are disjoint. The edges are $E = E_1 \cup E_2$ where $E_1 \subset V_1 \times V_2$ and $E_2 \subset V_2 \times V_1$. Without loss of generality, we assume $V = \{1, \ldots, 2n\}$. The index set $I = \{1, \ldots, n\}$, and the position set $V_1 = \{2i | i \in I\}$ and $V_2 = \{2i - 1 | i \in I\}$. Furthermore, there are exactly two outgoing edges from each vertex. Hence, we may interpret the set of edges as functions $e_0 : V \longrightarrow V$ and $e_1 : V \longrightarrow V$ where the edges out of vertex v are $(v, e_0(v))$ and $(v, e_1(v))$. The neighbors of a vertex v are $N(v) = \{e_0(v), e_1(v)\}$.

The game is played between two players: Player 1 and Player 2. From a position $v \in V_1$, Player 1 can make a move to $e_0(v)$ or $e_1(v)$. Similarly, Player 2 moves from $w \in V_2$ to $e_0(w)$ or $e_1(w)$. For convenience, we will refer to Player 1 as "him", and Player 2 as "her".

The game begins from the start position $p_0 \in V_1$ with Player 1 making a move. Then it is Player 2's turn to move, and so on. The resulting play of the game is $p = p_0 p_1 p_2 \ldots$ where Player 1 moved from position p_{2i} to p_{2i+1}, and Player 2 moved from position p_{2i+1} to p_{2i+2}. The payoff is a function $P : V^\omega \longrightarrow \mathcal{R}$. The games we are interested in are perfect information zero-sum games. When the play is p, Player 2 pays to Player 1 the amount $P(p)$ (when $P(p)$ is negative, Player 1 pays to Player 2). Player 1 tries to maximize the payoff, and Player 2 tries to minimize it.

A game is a triple $\langle G, p_0, P \rangle$ where $G = (V, E)$ is a game graph, $p_0 \in V$ is the position from which the game starts, and P is the payoff function.

Example 5.2.1 *Consider the game graph shown in Figure 5.1. The positions represented with circles are V_1. Those represented with squares are V_2. There is an arrow pointing to the start position. When the game position is a circle, Player 1 moves; when the game is at a square, Player 2 moves. The edges out of each position represent the moves the players can make. So from position D, Player 2 can make a move to either position C, or to position B. For a play p, let us define the payoff function to be $P(p) = -1$ when $p_i = E$ or $p_i = F$ for some i. Otherwise $P(p) = 1$. So, Player*

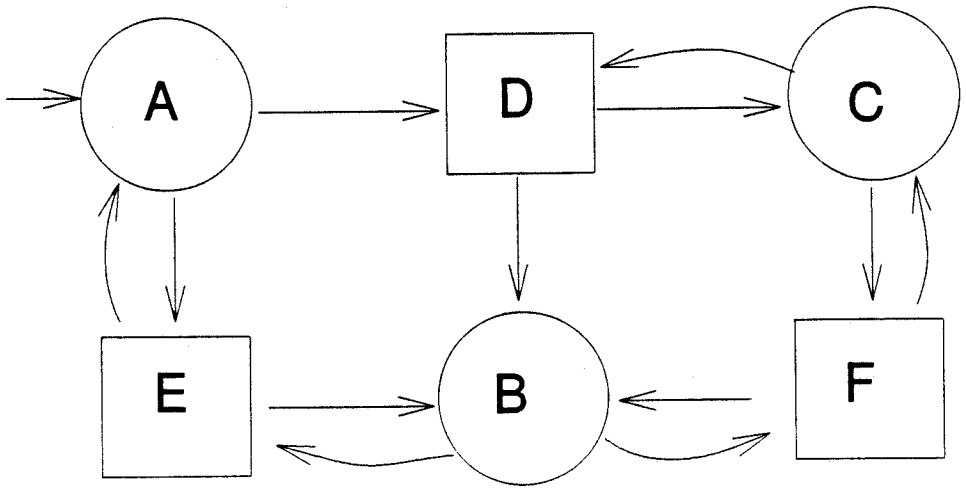

Figure 5.1: An Example Game

2 gets paid 1 dollar provided the play p "hits" the vertices E or F. On the other hand, if those vertices are not hit, Player 2 has to pay 1 dollar to Player 1.

We will say Player 1 wins in the play p provided $P(p) > 0$; otherwise Player 2 wins. We leave it to the reader to find the winner in the game of Figure 5.1.

5.2.1 Strategies for Playing

A strategy for a player is a rule or a set of rules which tells the player how to play. We first discuss history based strategies. The history of the play p at step j is $h_j = p_0 p_1 \dots p_j$. A history based strategy for Player 1 is the set of functions $\tau = \{\tau_{2i}\}$ from the histories of the play to its next move. At step $2i$ when h_{2i} is the history at step $2i$, Player 1 makes a move to position $\tau_{2i}(h_{2i})$. Similarly, a history based strategy for Player 2 is the set of functions $\sigma = \{\sigma_{2i+1}\}$. Let us denote by HD the set of all history based strategies.

In many games, it is possible to play with a simpler class of strategies: the positional strategies. A positional strategy for Player 1 is a function $\tau : V_1 \longrightarrow V_2$ where $(v, \tau(v)) \in E_1$. When in position v, Player 1 always moves to $\tau(v)$, independent of the history of the play. A positional strategy for Player 2 is a function $\sigma : V_2 \longrightarrow V_1$. Positional strategies are also called deterministic Markov strategies. Let us denote by

MD the set of positional strategies. Corresponding to a positional strategy τ is the history based strategy $\{\tau_{2i}\}$ where $\tau_{2i}(h_{2i}) = \tau(p_{2i})$. When convenient, we will also think of positional strategy τ as the set of edges $\{(v, \tau(v))\}$.

5.2.2 Value of the Game

Denote by A the strategy space for Player 1 and by B the strategy space for Player 2. Fixing a strategy $\tau \in A$ for Player 1 and $\sigma \in B$ for Player 2 leads to the play $p = p_0 p_1 \ldots$ where $p_{2i+1} = \tau_{2i}(p_{2i})$ and $p_{2i+2} = \sigma_{2i+1}(p_{2i+1})$.

Definition 5.2.1 *The strategy payoff function is* $\psi : A \times B \longrightarrow \mathcal{R}$ *where* $\psi(\tau, \sigma) = P(p)$. *The play* p *is obtained by fixing the strategy of Player 1 to* τ *and the strategy of Player 2 to* σ.

Definition 5.2.2 *A game is said to have a value in strategy space* (A, B) *provided*

$$\sup_{\tau} \inf_{\sigma} \psi(\tau, \sigma) = \inf_{\sigma} \sup_{\tau} \psi(\tau, \sigma), \tag{5.1}$$

where the strategies for Player 1 are selected from A, and the strategies for Player 2 are selected from B. The value of Equation 5.1 is the value for the game in strategy space (A, B).

The value for the game in strategy space *(HD,HD)* is the *value of the game*. When the value of the game is v, and $v > 0$, we say Player 1 wins the game. Otherwise, Player 2 wins.

It will also be useful to study the class of games $\mathcal{G} = \{\mathcal{G}_j\} = \langle G, P \rangle$ where $\mathcal{G}_j = \langle G, j, P \rangle$ is the game played on graph G starting from position j. The value of $\mathcal{G}$ is a vector u where $u(j)$ is the value of game $\mathcal{G}_j$.

5.2.3 Complete Strategy Spaces and 1-Player Games

Definition 5.2.3 *For Player 1, the strategy space MD is complete for Player 1 provided the value for the game in (MD,HD) is the value of the game. Completeness for Player 2 is defined similarly.*

A player needs to search only among positional strategies when the strategy space MD is complete.

Definition 5.2.4 *The response function for Player 2 is* $r_2 : A \longrightarrow B$ *where*

$$r_2(\tau) = \arg\min_{\sigma} \psi(\tau, \sigma).$$

We note that $r_2(\tau)$ is a set of strategies — each strategy in $r_2(\tau)$ is the optimal strategy for Player 2 in response to Player 1 playing with strategy τ. The response function for Player 1, r_1, is defined similarly. Strategy $\tau \in A$ and $\sigma \in B$ are said to be optimal in (A, B) provided $\tau \in r_1(\sigma)$ and $\sigma \in r_2(\tau)$.

A 1-Player game is obtained from game $\mathcal{G}$ by fixing the strategy of one of the players to a positional strategy.

Definition 5.2.5 *Fixing positional strategy* τ *for Player 1, define the 1-Player game* $\mathcal{G}_\tau = (G_\tau, P)$ *where* $G_\tau = (V, E_2 \cup \tau)$.

In game $\mathcal{G}_\tau$, Player 2's objective is to minimize the payoff when Player 1 has fixed his strategy to τ. The optimal strategy for Player 2 in game $\mathcal{G}_\tau$ is $\sigma \in r_2(\tau)$.

Definition 5.2.6 *We say MD is complete for Player 2 in* $\mathcal{G}_\tau$ *provided* $r_2(\tau)$ *contains a positional strategy.*

We can similarly define 1-Player games and completeness for Player 1.

Lemma 5.2.1 *Suppose* τ *and* σ *are optimal in* (MD, MD) *for game* $\mathcal{G}$*, and MD is complete for both players in 1-Player games. Then* τ *and* σ *are also optimal in* (HD, HD).

5.3 Games on ω-Automata, Church's Problem and μ-Calculus

ω-automata are finite automata which accept infinite sequences. The sequences which are accepted form the language of the ω-automaton [40, 23]. A game on an ω-automaton is played between Player 1 and Player 2. The two players together create

a sequence $p = p_0 p_1 p_2 \ldots$. Player 1 wins provided the sequence is in the language of the ω-automaton; otherwise, Player 2 wins. In Section 5.3.1, we discuss ω-automata. In Section 5.3.2, we discuss games on ω-automata. In Section 5.3.3, we discuss the logic propositional μ-calculus.

5.3.1 ω-Automata

A deterministic ω-automaton over alphabet Σ is $\mathcal{A} = \langle T, \phi \rangle$, where T is the transition structure, and ϕ is the acceptance condition. The transition structure is $T = (Q, q_0, \Sigma, \delta)$ where Q is a finite set of states, $q_0 \in Q$ is the initial state, and $\delta : Q \times \Sigma \longrightarrow Q$ is the transition function.

Definition 5.3.1 *A word $\sigma \in \Sigma^\omega$ has the run $r_\sigma \in Q^\omega$ where $r_\sigma(0) = q_0$ and $\delta(r_\sigma(i), \sigma(i)) = r_\sigma(i+1)$.*

The infinity set of the run r_σ, denoted $\inf(r_\sigma)$, is the set of states that are visited infinitely often in r_σ. The acceptance condition ϕ is a boolean formula, where the boolean variables are the states $Q = \{q_1, \ldots, q_m\}$.

Definition 5.3.2 *A boolean formula is generated by the following rules*
1) $q_i \in Q$ is a boolean formula.
2) If ϕ_1, ϕ_2 are boolean formulae, then $\neg\phi_1$, $\phi_1 \vee \phi_2$, and $\phi_1 \wedge \phi_2$ are boolean formulae.

The truth of the boolean variable $q_i \in Q$ is determined by the run r_σ. For $C \subset Q$, define the boolean assignment $q_i = 1$ provided $q_i \in C$, otherwise $q_i = 0$. Let $\phi[C]$ denote the truth value of ϕ under this assignment.

Definition 5.3.3 *The language generated by the ω-automaton $\mathcal{A} = \langle T, \phi \rangle$ is $\mathcal{L}(\mathcal{A}) = \{\sigma | \sigma \in \Sigma^\omega$ and $\phi[\inf(r_\sigma)] = 1 \}$.*

Definition 5.3.4 *We define various types of boolean formulae which are used for acceptance criterion:*
1) A disjunctive formula (DF) (Buchi formula) is a disjunction of boolean variables, i.e., for $F = \{f_1, \ldots, f_k\} \subset Q$, $\phi = f_1 \vee \ldots \vee f_k$.

2) A Rabin formula is $\phi = \bigvee_{i=1}^{n}(L_i \wedge \neg(\bar{U}_i))$ *where* $L_i, U_i, 1 \leq i \leq n$ *are DF.*
3) A Streett formula is $\phi = \bigwedge_{i=1}^{n}(L_i \vee \neg(\bar{U}_i))$ *where* $L_i, U_i, 1 \leq i \leq n$ *are DF.*
4) Given DF $E_i, F_i, i = 1, \ldots, n$ *where* $E_n \subset F_n \subset E_{n-1} \subset F_{n-1} \subset \ldots E_1 \subset F_1$, *the chain formula is* $\phi = \bigvee_{i=1}^{n}(F_i \wedge \neg E_i) \equiv \bigwedge_{i=1}^{n+1}(\neg E_{i-1} \vee F_i)$ *where* $E_0 = true$ *and* $F_{n+1} = false$.

The complement of a Rabin formula is a Streett formula, and vice versa. The chain formula can be expressed either as a Rabin formula, or as a Streett formula.

The chain acceptance condition is also sometimes written as the parity acceptance condition (B_i, G_i) where $B_i = E_i \setminus F_{i+1}$ and $G_i = F_i \setminus E_i$ (see [23, 13]). The sets B_i and G_i are disjoint. A run r_σ is accepting provided for some i, $\inf(r_\sigma)$ "touches" G_i but not B_j for $j \geq i$.

The Rabin and Streett acceptance conditions are also referred to as a set of pairs (L_i, U_i) of subsets of states. A run r_σ in a DRA is accepting if for some pair, $\inf(r_\sigma)$ "touches" L_i and is contained in U_i.

Remark: Our syntax for the Rabin and Streett condition differs from the standard definition in the literature, where a Rabin formula is $\bigvee_{i=1}^{n}(L_i \wedge \neg U_i)$; a run is accepting (for Rabin acceptance), if for some i, it visits L_i (the GREEN states) infinitely often, but U_i (the RED states) only finitely often. Complementing U_i translates between the standard syntax and ours.

The ω-automaton $\mathcal{A} = \langle T, \phi \rangle$, where ϕ is a Buchi, Rabin, Streett, or Chain formula is called a Buchi, Rabin, Streett, and Chain Automaton respectively. We will need the following two results from the theory of ω-automata.

Theorem 5.3.1 *Determining whether the language of* $\mathcal{A} = \langle T, \phi \rangle$ *is empty for Buchi, Rabin, Streett, and Chain Automata is in polynomial time.*

Theorem 5.3.2 *Given a Rabin Automaton* $\mathcal{A} = \langle T, \phi \rangle$ *with* n *states and* h *pairs, there is a Chain Automata* $\mathcal{B}$ *with* nh^k *states and* k *pairs such that* $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$. *The index* k *is the Rabin Index of the language — the minimum number of pairs in a Rabin automaton required to realize the language.*

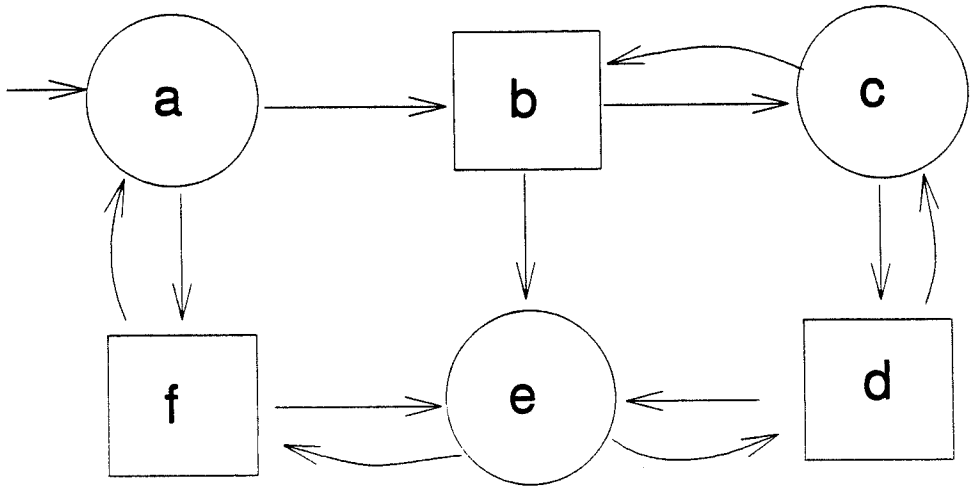

Figure 5.2: An Game on Buchi Automaton

For a more complete discussion of ω-automata and a proof of Theorem 5.3.1, see [40]. Theorem 5.3.2 is the main result of [23].

5.3.2 Games on ω-Automata

A game on an ω-automaton is $\mathcal{G} = (G, P_\phi)$ where $G = (V, E)$ is a game graph and ϕ is an acceptance condition. The acceptance condition ϕ is a boolean formula with boolean variables $V = \{v_1, \ldots, v_{2n}\}$. For a Buchi, Rabin, Streett, or Chain formula ϕ, we say the game is a Buchi, Rabin, Streett, and Chain game respectively.

Definition 5.3.5 *The payoff function for the game is* $P_\phi : V^\omega \longrightarrow \{0,1\}$, *where for a play* p, $P_\phi(p) = \phi[\inf(p)]$.

For a play p, Player 1 is the the winner provided $\phi[\inf(p)] = 1$, and Player 2 is the winner when $\phi[\inf(p)] = 0$. Player 1 tries to maximize the payoff, and Player 2 tries to minimize it.

Example 5.3.1 *Consider the formula* $\phi = v_1 \vee \ldots \vee v_k$ *where* $F = \{v_1, \ldots, v_k\}$ *are called the final states. For Player 1 to win, the play must visit some state in* F *infinitely often. On the other hand, Player 2 tries to limit the number of times a state in* F *is visited. For the game graph in Figure 5.2,* $F = \{b, e\}$ *and the boolean*

formula $\phi = b \vee e$. To win, Player 1 must use a strategy, such that any resulting play visits position b or position e infinitely often. Consider the positional strategy τ for Player 1 where $\tau(a) = b$, $\tau(c) = b$ and $\tau(e) = f$. The reader can check that starting from any position, this is a winning strategy for Player 1 since any resulting play visits a state in F infinitely often.

Games on ω-automata are closely related to synthesis problems. Church in [8] poses two problems about digital circuits and logic: the decision problem and the synthesis problem. The decision problem is the verification question: does the designed circuit meet its specification? The synthesis problem is to automatically synthesize a circuit to meet a specification. He gives various cases of the synthesis and decision problem which had been solved, but points out that the synthesis question for ω-automata was unresolved.

In the synthesis problem, the specification is a game which is played by two players: the "input" (also sometimes called the "disturbance") and the "controller". The idea is that for every input to the circuit, the controller must respond so that the specification is never violated. When the controller can win this game, we say the specification is synthesizable. The winning strategy for the controller is implemented as the circuit. This circuit then meets its specification.

Solving a game on ω-automaton is sometimes called Church's problem. In [40], Church's problem is presented as the following two questions about games on ω-automata:

1. Solvability Problem — Is there an algorithm to determine the winner?

2. Synthesis Problem — Is there a finite-state strategy for the winner?

The problem was resolved by Buchi, Landweber and Rabin (see [35]).

Theorem 5.3.3 *Consider the game $\mathcal{G} = (G, P_\phi)$ where ϕ is a boolean formula. Then*

1. the game $\mathcal{G}$ has a value (i.e., the game has a winner),

2. there is an algorithm to determine the winner,

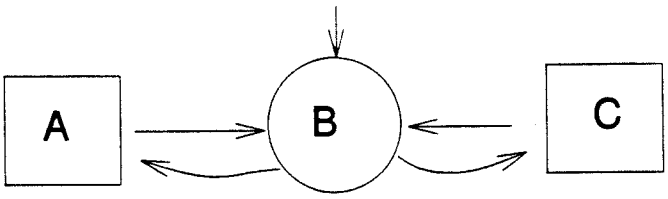

Figure 5.3: The value in (MD,MD) and (HD,HD) are not the same

3. the winner can implement its strategy with a finite amount of memory.

The following is the result of [11].

Theorem 5.3.4 *Strategy space MD is complete for Player 1 in Rabin Games.*

Hence, Player 1 can restrict himself to playing with positional strategies in Rabin Games. Since a Chain formula is a special case of a Rabin formula and a Streett formula, and the complement of a Streett formula is a Rabin formula, we get the following result.

Lemma 5.3.1 *The strategy space MD is complete for both players in Chain Games.*

At this time, one may ask the following questions: Are there games in which a player needs to remember the history? Do games on ω-automata have a value in the strategy space *(MD,MD)*? Does the value in *(MD,MD)* coincide with the value in *(HD,HD)* ?

We first show with an example that history based strategies are required, and that the value in *(MD,MD)* need not coincide with the value in *(HD,HD)*.

Example 5.3.2 *Consider the Streett game in Figure 5.3 where the acceptance condition is $A \wedge C$. We use the convention that when both edges out of a position are going to the same position, we only draw one edge. To win, Player 1 has to visit A and C infinitely often. Consider the positional strategy where Player 2 always plays to B from A and C. There is no positional strategy for Player 1 which can beat this. Hence the value of the game in (MD,MD) is 0. But using a history based strategy Player 1 can win. He just alternately visits A and C. Therefore, the value of the game in (MD,MD) and (HD,HD) need not coincide.*

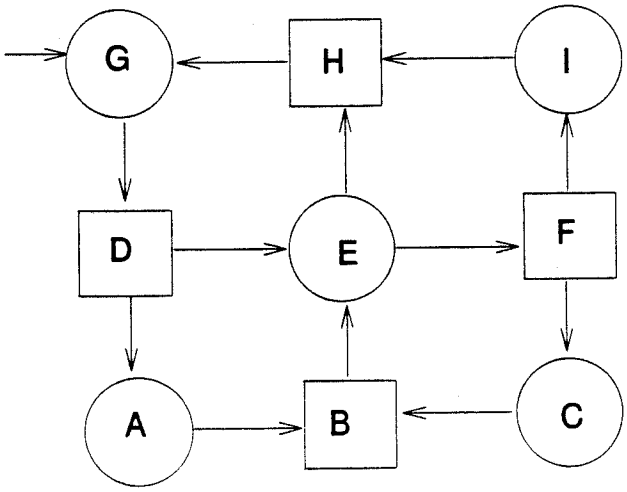

Figure 5.4: The game does not have a value for (MD,MD)

In the next example, we show that a game need not a have a value for the strategy space *(MD,MD)* .

Example 5.3.3 *Consider the Rabin game shown in Figure 5.4 where the acceptance condition is* $\phi = C \vee (I \wedge \neg A) \vee (A \wedge \neg I)$. *Player 1 wins by visiting infinitely often (i.o.) position C, or visiting i.o. position I but not A, or by visiting i.o. position A but not I. We will leave it to the reader to check that any positional strategy of Player 1 can be beaten by Player 2 playing a positional strategy. And any positional strategy of Player 2 can be beaten by a positional strategy of Player 1. Hence the game does not have a value for the strategy space (MD,MD). But the game is won by Player 2 using the following history based strategy: Player 2 always plays to position I from position F. At position D, she looks at what Player 1 did during his last move at E: when Player 1 moved to position F, Player 2 moves from D to A; if Player 1 moved to position H, then Player 2 moves from D to E. The reader can check that this is a winning strategy. We can now see the significance of the first part of Theorem 5.3.3 which assures us that a game on an* ω*-automaton always has a value in strategy space* (HD,HD).

Given an ω-automaton of one type, we can convert it to another kind such that the languages of the two are the same. The same construction can be applied to convert

one type of game into another type such that both games have the same winner, and the winner can use the "same" strategy to win both games.

Let us clarify the meaning of the "same" strategy. The history upto step j is $h_j = p_0 p_1 p_2 \ldots p_j$. A strategy for Player 1 is a set of functions $\tau = \{\tau_{2i}\}$ where at step $2i$, Player 1 moves to the position $\tau_{2i}(h_{2i})$. Corresponding to h_j is the string $\tilde{h}_j = b_0 b_1 \ldots b_{j-1}$ where $b_i \in \Sigma = \{0,1\}$ and $p_{i+1} = e_{b_i}(p_i)$. We may view a strategy as functions $\tilde{\tau} = \{\tilde{\tau}_{2i}\}$ from strings in Σ^* to Σ. That is $w = \tilde{\tau}_{2i}(\tilde{h}_{2i}) \in \Sigma$ and at step $2i$, Player 1 moves to $e_w(p_{2i})$. Given a strategy τ and a game graph G, strategy $\tilde{\tau}$ is uniquely defined, and vice versa. Furthermore $\tilde{\tau}$ does not depend on the graph G. Strategies for Player 2 which are independent of the graph G could be defined similarly. When the game is converted into another type of game, the winner can use same strategy $\tilde{\tau}$ to win both games.

Lemma 5.3.2 *Given a Rabin Game $\mathcal{G} = (G, P_\phi)$ with n positions and h pairs, there is a Chain Game $\mathcal{C} = (C, P_\psi)$ with nh^k positions and k pairs where $k \leq h$, such that the same player wins both games, and can do so by using the same strategy. The index k is the Rabin Index of the Game language.*

Proof: Using Theorem 5.3.2. ∎

But a chain game can be played using positional strategies. Hence we obtain an upper bound on the amount of memory that Player 2 (Player 1) requires to play a Rabin Game (Streett Game).

Theorem 5.3.5 *In a Rabin Game (Streett Game) with n positions and h pairs, the amount of memory required by Player 2 (Player 1) to play is at most nh^k where k is the Rabin Index of the game language.*

1-Player ω-Automata Games

Definition 5.3.6 *Corresponding to the 1-Player game $\mathcal{G}_\tau = (G_\tau, P_\phi)$ with $G = (V, E_2 \cup \tau)$, define the ω-automaton $\mathcal{A} = \langle T, \phi \rangle$ with alphabet $\Sigma = \{0,1\}$ where $T = \langle V, q_0, \delta, \Sigma \rangle$ and for $i \in I$, $\delta(2i-1, 0) = e_0(2i)$, $\delta(2i-1, 1) = e_1(2i)$ and $\delta(2i, 0) = \delta(2i, 1) = \tau(2i)$.*

The ω-automaton $\mathcal{A}$ is obtained by labelling the edges of transition structure $\mathcal{G}_\tau$ with an alphabet. A player can win the game $\mathcal{G}_\tau$ provided he produces a play which satisfies acceptance condition ϕ. But this is exactly the emptiness problem for the ω-automaton $\mathcal{A}$. Hence, a player wins the 1-Player game $\mathcal{G}_\tau$ iff $\mathcal{L}(\mathcal{A}) \neq \emptyset$.

Lemma 5.3.3 *Determining if a player wins the 1-Player Buchi, Rabin, Streett or Parity game is in polynomial time.*

Proof: From Theorem 5.3.1, checking emptiness of the corresponding ω-automaton is in polynomial time. ∎

We also know that the space *MD* is complete for Player 1 in Rabin games, and for both players in Chain Games. From Lemma 5.3.3, it follows that determining whether Player 1 wins a Rabin Game is in NP, a Streett Game in Co-NP, and a Chain Game in $NP \cap Co-NP$. In [12], it is shown that determining whether Player 1 wins a Rabin (Streett) game is NP-hard (Co-NP hard).

5.3.3 Propositional μ-Calculus

Propositional μ-calculus is a propositional modal logic with a least fixed point operator μ [22, 13].

Syntax

The logic has atomic propositions $A = p, q, \ldots$, and propositional variables $x, y, \ldots$. The set of formulas are defined inductively:

1. atomic propositions $p, q, \ldots$,
2. propositional variables $x, y, \ldots$,
3. $f \vee g$, and $\neg f$ where f, g are formulae,
4. $\langle a \rangle f$ where f is a formula,
5. $\mu x.f$ where formula f is positive in variable x.

Scope, occurence of free and bound variables, and closed formulas are defined similar to first-order logic.

Semantics

The model for a μ-calculus formula is a Kripke structure $K = (S, E, L)$ where S is a set of states, $E \subset S \times S$, and $L : S \longrightarrow 2^A$.

A valuation ρ is map which assigns sets of states to free variables. It assigns ρ_i to variable x_i. We will write $f^K(\rho)$ to denote formula f interpreted with valuation ρ. When f does not contain free variables, its interpretation does not depend on ρ, and we write it as f^K. We define $f^K(\rho)$ inductively as follows:

1. $x_i^K(\rho) = \rho_i$
2. $q^K(\rho) = \{s | q \in L(s)\}$
3. $(f \vee g)^K(\rho) = f^K(\rho) \cup g^K(\rho)$
4. $(\neg f)^K(\rho) = S \setminus f^K(\rho)$
5. $(\langle a \rangle f)^K(\rho) = \{s | (s, s') \in E \text{ and } s' \in f^K(\rho)\}$
6. $(\mu x_i . f)^K(\rho) = \bigcap \{X | X = f^K(\rho') \text{ where } \rho'_i = X \text{ and } \rho'_j = \rho_j \text{ for } j \neq i\}$.

We also write $[a]f = \neg\langle a \rangle \neg f$. The greatest fixed point ν is defined as $\nu x.f = \neg\mu x.\neg f$.

The least fixed point can be computed by starting from the emptyset, and iterating until a fixed point is reached. Similarly, the greatest fixed point is computed by starting from the set containing all states and iterating [22]. In this way, the set f^K can be computed inductively.

Given a Kripke structure $K = (S, E, L)$, a state $s \in S$, and a μ-calculus formula f, the *model checking problem* is to determine whether $s \in f^K$.

Chain Games and μ-Calculus

The following two theorems of [13] show that the model checking problem for μ-calculus and solving Chain games are essentially equivalent problems.

Theorem 5.3.6 *Given a Kripke structure $K = (S, E, L)$, state $s_0 \in S$, and μ-calculus formula f, there is a Chain Game $\mathcal{G}$ with $O((|S| + |E|)|f|)$ positions such that Player 1 wins the chain game $\mathcal{G}$ iff $s_0 \in f^K$.*

Theorem 5.3.7 *Given a Chain Game $\mathcal{G} = \langle G, p_0, P_\phi \rangle$, where $G = (V, E)$, there is a Kripke structure K of size $O(|V|)$ and a μ-calculus formula f of size $O(|\phi|)$ such that $p_0 \in f^K$ iff Player 1 wins game $\mathcal{G}$.*

Since we gave an inductive algorithm for model checking μ-calculus, we also obtain an algorithm for solving Chain games. The complexity of this algorithm is $O((|V|k)^k)$ where $|V|$ is the number of positions in the chain game, and k is the number of pairs. In Section 5.5, we will compare this algorithm with a new algorithm which we present in Section 5.4.

5.4 Shapley's Game

In this section, we discuss one of the simplest dynamical games. The game is played on the game graph $G = (V, E)$. There is a reward $r(v)$ at each position v. When the play in the game is $p = p_0 p_1 \ldots p_N$, Player 2 pays to Player 1 the amount $\Sigma_{i=0}^{N} r(p_i)$. Player 1 tries to maximize his payoff, and Player 2 tries to minimize it.

This class of games was originally studied by Shapley under a more general setting. Shapley in [38] introduced the class of stochastic games. A stochastic game is a zero-sum game played between two players: Player 1 who is trying to maximize his payoff, and Player 2 who wants to minimize it. The game has a finite number of positions. In each position, the players choose from a finite number of actions. When the game is in positon v, both players simultaneously choose their actions (say actions a and b), then Player 1 gets a reward r_v^{ab} and the game moves to the position w with probability

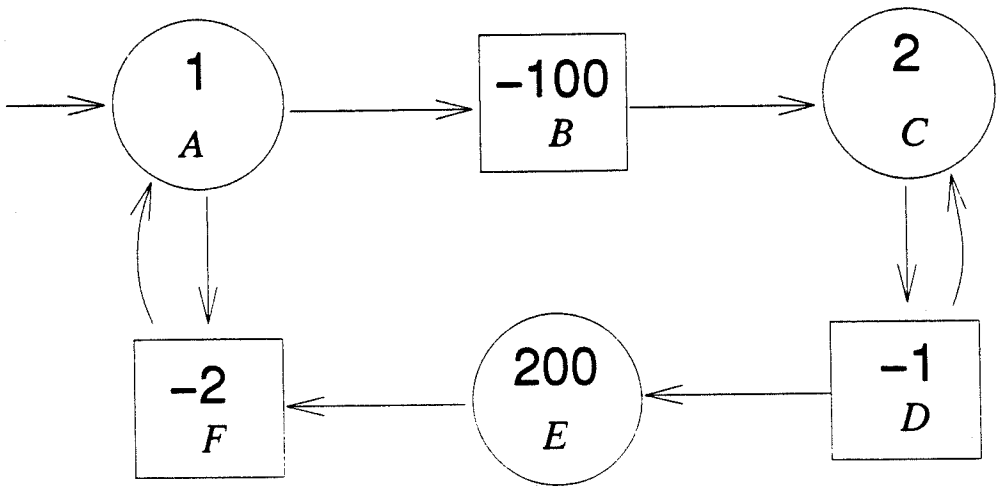

Figure 5.5: A Payoff Game

P_{vw}^{ab}. Both players again choose actions at position w and the game moves on to a new position, and so on.

Shapley studied stochastic games with the discounted payoff $P_\beta(p) = \sum_{i=0}^{\infty} \beta^i r_{p_i}^{a_i b_i}$, where $p = p_0 p_1 \ldots$ is the play of the game and a_i, b_i are actions of the players at step i and $0 < \beta < 1$. He showed that such games have a value, and both players have optimal stationary strategies [38].

We study a subclass of stochastic games — called *Payoff Games* — played on the game graph G. From positions $v \in V_1$, Player 1 chooses an action and moves to $e_0(v)$ or $e_1(v)$. From position $w \in V_2$, Player 2 moves to $e_0(w)$ or $e_1(w)$. The reward function $r : V \longrightarrow \mathcal{Z}$ is independent of actions. At a position v, Player 1 gets the reward $r(v)$. We study both finite and infinite games. Figure 5.5 shows an example of a payoff game.

In a finite N-step game the play is $p = p_0 p_1 \ldots p_N$. The payoff is

$$P_N(p) = \sum_{k=0}^{N-1} \beta^k r(p_k) + \beta^N f(p_N),$$

where f is the terminal reward function, and β is a discount factor.

For the infinite game, the play is $p = p_0 p_1 p_2 \ldots$. Two payoff functions are usually

studied. For a play $p = p_0 p_1 \ldots$, the discounted payoff is

$$P_\beta(p) = \sum_{k=0}^{\infty} \beta^k r(p_k)$$

where the discount factor $\beta \in (0,1)$. The average payoff is

$$\bar{P}(p) = \lim_{N\to\infty} \inf \frac{1}{N} \sum_{k=0}^{N} r(p_k).$$

In this section, we will discuss the class of games $\mathcal{G} = \{\mathcal{G}_j\} = \langle G, P\rangle$ where $\mathcal{G}_j = (G, j, P)$ is the game with start position j. Notice, the payoff function P actually depends on the reward function r for the game. This will be implicit throughout this section.

5.4.1 Notation

For a vector $u \in \mathcal{R}^k$, we say $u < v$ provided $u(i) \leq v(i)$ for each i, and $u(j) < v(j)$ for some j. The norm of u is $\|u\| = \max_i\{|u(i)|\}$. Given a function $r : V \longrightarrow \mathcal{R}$, where V is a finite set, and $W \subset V$, define $|W|$ to be the number of elements in the set W, $r(W) = \sum_{w\in W} r(w)$, and $mean(W) = \frac{r(W)}{|W|}$.

A map $T : \mathcal{R}^k \longrightarrow \mathcal{R}^m$ is said to be a contraction provided there is a $0 \leq \beta < 1$ such that $\|T(x) - T(y)\| \leq \beta\|x - y\|$ for all x, y. A contraction map has a unique fixed point (i.e., unique x such that $T(x) = x$). Furthermore, for any y $\lim_{n\to\infty} T^n(y) = x$ (by T^n, we mean the n fold composition of T). See [24, 37] for a proof of this.

The games are played on the game graph $G = (V, E)$ where $V = \{1, \ldots, 2n\}$. We remind the reader that $I = \{1, \ldots, n\}$, $V_1 = \{2i | i \in I\}$, and $V_2 = \{2i - 1 | i \in I\}$. The edges out of position j are $(j, e_0(j))$ and $(j, e_1(j))$; the set $N(j) = \{e_0(j), e_1(j)\}$. We will usually write τ for a strategy of Player 1 and σ for a strategy of Player 2.

5.4.2 1-Player Payoff Games

A 1-Player payoff game is $\mathcal{G}_\tau = (G_\tau, P)$ where $G_\tau = (V, E_2 \cup \tau)$. The strategy for Player 1 is fixed to τ and the game is played by Player 2. The reward function is $r : V \longrightarrow \mathcal{R}$. The 1-Player payoff games are deterministic analogues of Markov

decision problems [20, 24, 37]. We study the N-step game, and the infinite discounted and mean payoff games. We show that 1-Player discounted and mean payoff games have a value and optimal positional strategies. The value and optimal strategies in 1-Player games can be computed in polynomial time.

N-Step

The payoff is

$$P_N(p) = \sum_{k=0}^{N-1} \beta^k r(p_k) + \beta^N f(p_N)$$

where $f : V \longrightarrow \mathcal{R}$ is the terminal reward function.

Definition 5.4.1 *Define the map $T_\tau : \mathcal{R}^{2n} \longrightarrow \mathcal{R}^{2n}$ where for $i \in I$,*

$$T_\tau(u)(2i) = r(2i) + \beta u(\tau(2i)) \tag{5.2}$$

$$T_\tau(u)(2i-1) = r(2i-1) + \beta \min_{a \in N(2i-1)} \{u(a)\}.$$

Equation 5.2 is the optimality equation in dynamic programming.

Lemma 5.4.1 *The value of the N-step 1-Player game $\mathcal{G}_\tau$ is $u_N = T_\tau^N(f)$.*

Discounted Payoff

The payoff is

$$P_\beta(p) = \sum_{k=0}^{\infty} \beta^k r(p_k)$$

where $\beta \in (0,1)$.

Lemma 5.4.2 *The map $T_\tau : \mathcal{R}^{2n} \longrightarrow \mathcal{R}^{2n}$ in Definition 5.4.1 satisfies*

$$\|T_\tau(u) - T_\tau(v)\| \leq \beta \|u - v\|.$$

Therefore for $\beta \in (0,1)$, T_τ is a contraction and has a unique fixed point.

Lemma 5.4.3 *The value of the game $\mathcal{G}_\tau$ is v, where $v = T_\tau(v)$. Player 2 has an optimal positional strategy σ where for $i \in I$,*

$$\sigma(2i-1) = \arg \min_{a \in N(2i-1)} \{v(a)\}.$$

Mean Payoff

The payoff in the mean payoff game is

$$\bar{P}(p) = \lim_{N\to\infty} \inf \sum_{k=0}^{N} r(p_k).$$

It is easy to see that the value of position i in game $\mathcal{G}_\tau$ is the smallest mean value of a cycle that is reachable from i. Furthermore, Player 2 has an optimal positional strategy which drives the game from position i to this cycle.

5.4.3 A Finite N-Step Game

In a finite N-step Game $\mathcal{G} = (G, P_N)$, the game ends after N steps. The play of the game is $p = p_0 p_1 \dots p_N$. Player 2 pays to Player 1

$$P_N(p) = \sum_{k=0}^{N-1} \beta^k r(p_k) + \beta^N f(p_N)$$

where β is a discount factor, and $f : V \longrightarrow \mathcal{R}$ is the terminal reward function. We want to determine the value v_N of the N-step game, and the optimal strategies of the two players.

Definition 5.4.2 *Define the map $T : \mathcal{R}^{2n} \longrightarrow \mathcal{R}^{2n}$ where for $i \in I$,*

$$T(u)(2i) = r(2i) + \beta \max_{a\in N(2i)} \{u(a)\} \tag{5.3}$$

$$T(u)(2i-1) = r(2i-1) + \beta \min_{a\in N(2i-1)} \{u(a)\}$$

Equation 5.3 is the basic optimality equation for the payoff games. Using dynamic programming, we can determine the value of the game, and the strategies of the players. The strategy for Player 1 is $\{\tau_k\}$ where $\tau_k : V_1 \longrightarrow V_2$. At step k, when the game is in position p_k and it is Player 1's turn to move, he moves to $\tau_k(p_k)$. Strategy for Player 2 is similarly defined.

Lemma 5.4.4 *The value of the N-step game with terminal reward f is $v_N = T^N(f)$. The optimal strategy for Player 1 is $\{\tau_k\}$ where*

$$\tau_{N-k}(2i) = \arg \max_{a\in N(2i)} \{v_{k-1}(a)\}.$$

The optimal strategy for Player 2 is $\{\sigma_k\}$ *where*

$$\sigma_{N-k}(2i-1) = \arg \min_{a \in N(2i-1)} \{v_{k-1}(a)\}.$$

Proof: By induction on N. ∎

Example 5.4.1 *Let's consider the payoff game in Figure 5.5 where the game stops after* $N = 5$ *steps. The discount factor* $\beta = 1$, *and the terminal reward is* 0 *at each position. The strategy for Player 1 is* $\{\tau_k\}$, *and for Player 2,* $\{\sigma_k\}$, *where* $k = 0, \ldots, 4$. *We leave it to the reader to show that for each* k $\tau_k(A) = F$, $\tau_k(C) = D$, *and* $\tau_k(E) = F$; *and* $\sigma_k(B) = C$, $\sigma_k(D) = C$, *and* $\sigma_k(F) = A$. *When the game starts from position A, Player 2 wins the game.*

Next consider the game for a large number of steps, say $N = 1000$. *We leave it to the reader to show that when the game starts from position A, Player 1 wins the game.*

5.4.4 Discounted Payoff Game

In this section, we study the discounted payoff game (DPG) $\mathcal{D} = (G, P_\beta)$, where discount factor $\beta \in (0,1)$. The play is $p = p_0 p_1 p_2 \ldots$, and the payoff is

$$P_\beta(p) = \sum_{k=0}^{\infty} \beta^k r(p_k).$$

We prove Shapley's result that the game has a value, and both players have optimal positional strategies. We also discuss Howard's policy iteration algorithm [20] and its extension to stochastic games [19].

Value of the Game and Successive Approximation

Lemma 5.4.5 and Theorem 5.4.1 are due to Shapley [38].

Lemma 5.4.5 *The map* T *in Definition 5.4.2 satisfies*

$$\|T(u) - T(v)\| \leq \beta \|u - v\|.$$

Proof: For $j \in V_1$,

$$\begin{aligned} |T(u)(j) - T(v)(j)| &= \beta |\max_{a \in N(j)} \{u(a)\} - \max_{a \in N(j)} \{v(a)\}| \\ &\leq \beta \max_{a \in N(j)} |u(a) - v(a)| \\ &\leq \beta \|u - v\| \end{aligned}$$

Similarly for $k \in V_2$, $|T(u)(k) - T(v)(k)| \leq \beta \|u - v\|$. Therefore $\|T(u) - T(v)\| \leq \beta \|u - v\|$. ∎

By Lemma 5.4.5, the map T in Definition 5.4.2 is a contraction, and hence has a unique fixed point v.

Theorem 5.4.1 *The value of the game the DPG $\mathcal{D}$ is v where $v = T(v)$.*

Proof: Define the strategy τ for Player 1 as:

$$\tau(2i) = \arg \max_{a \in N(2i)} \{v(a)\}.$$

Similarly, the strategy σ for Player 2 is:

$$\sigma(2i-1) = \arg \min_{a \in N(2i-1)} \{v(a)\}.$$

From Lemma 5.4.3, σ is the optimal response of Player 2 to τ, and τ is the optimal response of Player 1 to σ. Therefore, strategies τ and σ are optimal in (MD, MD). From Lemma 5.2.1, they are also optimal in (HD, HD). The payoff when Player 1 plays with τ, and Player 2 plays with σ is v. ∎

How do we compute the value of the discounted payoff game? This value can be approximated by computing the value of the N-step game for large N. This is known as the method of successive approximation [38, 20, 37, 24].

Lemma 5.4.6 *Suppose v is the value of the discounted payoff game. Then $\lim_{N \to \infty} v_N = v$ where v_N is the value of the N-step game.*

Proof: Since $v_N = T^N(f)$ and $\lim_{N \to \infty} T^N(f) = v$, where v is the unique fixed point of T. ∎

Theorem 5.4.2 *Consider the discounted payoff game $\mathcal{D} = (G, P_\beta)$ with $G = (V, E)$ and $\beta = \frac{k}{l}$. Then using the successive approximation technique, the value of the DPG $\mathcal{D}$ can be computed in $O(\frac{n^2 log(l) + log M}{log(\frac{l}{k})})$ steps where $|V| = 2n$ and $M = \max_{v \in V} |r(v)|$.*

Proof: First we will show that the value of position i, $v(i)$, is a rational number $\frac{n}{d}$ where n and d are integers. Then we will get a bound on the size of d.

The discounted game has optimal positional strategies for the two players. When the two players play these positional strategies, the play from position i is $p = p_0 p_1 \ldots p_g (p_{g+1} \ldots p_{g+h})^\omega$ where $g + h \leq 2n$. Therefore the value

$$v(i) = \sum_{m=0}^{g} \beta^m r(p_m) + \frac{\beta^g}{1 - \beta^h} (\sum_{j=1}^{h} \beta^j r(p_{g+j})).$$

The value of position i can be written as a rational number $v(i) = \frac{n'}{d'}$ where $d' = l^{g+h}(l^h - k^h)$. But since g and h are not known a priori, we define

$$d = l^{2n} \prod_{j=1}^{2n} (l^j - k^j).$$

For each i, we can write $v(i)$ as a rational number with denominator d.

Since

$$\|T^N(0) - T^\infty(0)\| \leq \frac{\beta^N}{1 - \beta} M,$$

we choose N such that

$$\frac{1}{2d} > \frac{\beta^N}{1 - \beta} M.$$

This gives us N is $O(\frac{n^2 log(l) + log M}{log(\frac{l}{k})})$. The value $v(i)$ in the DPG $\mathcal{D}$ is obtained by computing $v_N(i)$ in the N-step game and rounding it to to the nearest rational with denominator d. ∎

From Theorem 5.4.2, it follows that for a fixed β, the discounted payoff games can be solved in polynomial time. Unfortunately, this method can be very slow in converging when $\beta \approx 1$ (see Example 5.4.1). To overcome the problem with slow convergence, we will study the policy iteration algorithm.

Strategy Improvement using Policy Iteration

In this section, we will show how to use a variant of the policy iteration algorithm [20, 37, 24] to determine the value of the discounted payoff game $\mathcal{D} = (G, P_\beta)$.

Definition 5.4.3 *Let us fix positional strategy τ for Player 1, and positional strategy σ for Player 2 in the DPG $\mathcal{D}$. Define the map $T_{\tau\sigma} : \mathcal{R}^{2n} \longrightarrow \mathcal{R}^{2n}$ by*

$$T_{\tau\sigma}(u)(2i) = r(2i) + \beta u(\tau(2i)) \tag{5.4}$$

$$T_{\tau\sigma}(u)(2i-1) = r(2i-1) + \beta u(\sigma(2i-1)).$$

Lemma 5.4.7 *The map $T_{\tau\sigma}$ in Equation 5.4 has the following properties:*

1. *For $u < v$, $T_{\tau\sigma}(u) < T_{\tau\sigma}(v)$.*
2. *$\|T_{\tau\sigma}(u) - T_{\tau\sigma}(v)\| \leq \beta\|u - v\|$.*

Hence, $v = T_{\tau\sigma}(v)$ is the unique fixed point. The vector v is the payoff when Player 1 plays strategy τ, and Player 2 plays strategy σ in game $\mathcal{D}$. The value v can be computed in polynomial time by solving the system of linear equations in Equation 5.4. We are now ready to describe the strategy improvement algorithm of [19].

Strategy Improvement Algorithm:

1. *$i = 0$; Player 1 picks an initial positional strategy τ_0.*
2. *repeat*
3. *Player 2 responds with positional strategy $\sigma_i \in r_2(\tau_i)$.*
4. *Compute the fixed point $v_i = T_{\tau_i\sigma_i}(v_i)$.*
5. *For $j \in I$, define $\tau_{i+1}(2j) = \tau_i(2j)$ when $\tau_i(2j) \in \arg\max_{a \in N(2j)}\{v_i(a)\}$, else define $\tau_{i+1}(2j) = \arg\max_{a \in N(2j)}\{v_i(a)\}$.*
6. *$i := i + 1$*

7. *until*$(\tau_i = \tau_{i-1})$

8. $\tau^* = \tau_{i-1}$; $\sigma^* = \sigma_{i-1}$.

At each step τ_i and σ_i are positional strategies. Each iteration of the algorithm takes only polynomial time. We next show that the algorithm incrementally improves the strategy for Player 1. That is, strategy τ_{i+1} is a better strategy than τ_i.

Theorem 5.4.3 *Suppose* $v_i = T_{\tau_i\sigma_i}(v_i)$ *and* $v_{i+1} = T_{\tau_{i+1}\sigma_{i+1}}(v_{i+1})$. *Then* $v_i < v_{i+1}$.

Proof: Since

$$\sigma_i(2j-1) = \arg\min_{b \in N(2j-1)} \{v_i(b)\},$$

and

$$\tau_{i+1}(2j) = \arg\max_{a \in N(2j)} \{v_i(a)\},$$

we get $v_i < T_{\tau_{i+1}\sigma_{i+1}}(v_i)$. From Lemma 5.4.7, $v_i < T_{\tau_{i+1}\sigma_{i+1}}(v_i) < T^2_{\tau_{i+1}\sigma_{i+1}}(v_i) < \cdots < T^n_{\tau_{i+1}\sigma_{i+1}}(v_i)$. But $\lim_{n\to\infty} T^n_{\tau_{i+1}\sigma_{i+1}}(v_i) = v_{i+1}$. Hence $v_i < v_{i+1}$. ∎

The strategy for Player 1 is improving at each step. In a game with $|V_1| = n$, Player 1 has atmost 2^n strategies. Therefore the algorithm terminates in at most 2^n steps with optimal strategies τ^* and σ^*.

Example 5.4.2 *Let us look at the payoff game in Figure 5.5 with discounted payoff and discount factor* $\beta = 0.999$. *We only need to look at what Player 1 does at position A, and what Player 2 does at position D since the moves at the other positions are fixed. We will solve the game using the strategy improvement algorithm. The initial strategy for Player 1 is* τ_0 *where* $\tau_0(A) = F$. *Then* $\sigma_0 \in r_2(\tau_0)$ *where* $\sigma_0(D) = E$. *The value is* v_0 *where* $v_0(A) = -499.25$, $v_0(B) = -398.35$, $v_0(C) = -298.65$, $v_0(D) = -300.95$, $v_0(E) = -300.25$ *and* $v_0(F) = -500.75$. *Since* $v_0(B) > v_0(F)$, $\tau_1(A) = B$. *Then* $\sigma_1(D) = C$, $v_1(A) = 400.85$, $v_1(B) = 400.25$, $v_1(C) = 500.75$, $v_1(D) = 499.25$, $v_1(E) = 598.05$, *and* $v_1(F) = 398.45$. *At the next iteration* $\tau_2 = \tau_1$ *and the algorithm stops. The reader can compare this with the successive approximation algorithm from Example 5.4.1.*

5.4.5 Mean Payoff Games

We next study the mean payoff game (MPG) $\mathcal{M} = (G, \bar{P})$ in which the play is $p = p_0 p_1 p_2 \ldots$ and the payoff is

$$\bar{P}(p) = \lim_{N \to \infty} \inf \frac{1}{N} \sum_{k=0}^{N} r(p_k).$$

Ehrenfeucht and Mycielski [10] showed that mean payoff games have a value and both players have optimal positional strategies. To do this, they needed to work with both finite and infinite games. They define a finite game in which the play stops as soon as a cycle is formed. Player 2 then pays to Player 1 the mean value of the cycle. All finite games have a value [25]. Ehrenfeucht and Mycielski show that this is also the value of the infinite game. To show that the game has optimal positional strategies, they needed to use the infinite game to show some results about the finite game. The proof is somewhat involved. Mycielski [28] mentions that no direct proof is known. In contrast, Shapley's proof for discounted payoff games relies on a contraction map, and is simpler. We give a new proof that mean payoff games have optimal positional strategies. In the proof, we make precise the relationship between discounted and mean payoff games.

Theorem 5.4.4 *Consider a game graph $G = (V, E)$ with the reward function $r : V \longrightarrow \mathcal{Z}$. Suppose v_β is the value of the DPG $\mathcal{D} = (G, P_\beta)$. Then*

1. *There is a $0 < \beta^* < 1$ and positional strategies τ^* and σ^* such that for all β where $\beta^* \leq \beta < 1$, τ^* and σ^* are optimal in the DPG $\mathcal{D} = (G, P_\beta)$.*

2. *Strategies τ^* and σ^* are optimal for the MPG $\mathcal{M} = (G, \bar{P})$, and the value of the MPG is $\bar{v} = \lim_{\beta \to 1}(1 - \beta) v_\beta$.*

Proof: 1) Fix a start position. Let τ and σ be positional strategies and $p = fc^\omega = p_0 p_1 \ldots p_g (p_{g+1} \ldots p_{g+h})^\omega$ the resulting play when Player 1 plays τ and Player 2 plays σ. Then $g \leq |V| - 1$, $|h| \leq |V|$,

$$P_\beta(p) = \sum_{m=0}^{g} \beta^m r(p_m) + \frac{\beta^g}{1 - \beta^h} (\sum_{j=1}^{h} \beta^j r(p_{g+j})),$$

and

$$(1-\beta)P_\beta(p) = (1-\beta)\sum_{m=0}^{g}\beta^m r(p_m) + \frac{\beta^g}{\sum_{l=1}^{h}\beta^{l-1}}(\sum_{j=1}^{h}\beta^j r(p_{g+j})).$$

Thus $\lim_{\beta\to 1}(1-\beta)P_\beta(p) = mean(c)$. Let τ' and σ' be some other positional strategies with play $p' = f'(c')^\omega$. Consider

$$poly(\beta) = ((1-\beta)P_\beta(p) - (1-\beta)P_\beta(p'))(\sum_{l=1}^{h}\beta^{l-1})(\sum_{l'=1}^{h'}\beta^{l'-1}),$$

where $h = |c|$ and $h' = |c'|$. Then $poly(\beta)$ has at most $2|V|$ zeroes since it is a polynomial of degree at most $2|V|$. Thus the strategy pair (τ,σ) and (τ',σ') have the same value at most $2|V|$ times. When (τ,σ) is optimal at discount factor β_1, and (τ',σ') is optimal at β_2, it must be the case that two strategy pairs have the same value at some α for $\beta_1 < \alpha < \beta_2$, but not the same value at β_1. Since there are only $2^{|V|}$ different strategy pairs, this can only happen a finite number of times. Therefore there is a β^* where $0 < \beta^* < 1$, and positional strategies τ^* and σ^* such that for all β, where $\beta^* \le \beta < 1$, τ^* and σ^* are optimal in the DPG $\mathcal{D} = (G, P_\beta)$.

2) Let the resulting play from playing τ^* and σ^* be $p = fc^\omega$. We first show that τ^* and σ^* are optimal in *(MD,MD)* for the MPG $\mathcal{M} = (G, \bar{P})$. Suppose σ', not σ^*, is the optimal response of Player 2 to τ^*. Let $p' = f'(c')^\omega$ be the resulting play. Then $mean(c') < mean(c)$. Therefore $\lim_{\beta\to 1}(1-\beta)P_\beta(p') < \lim_{\beta\to 1}(1-\beta)P_\beta(p)$. But then for some $\beta^* < \beta' < 1$, $P_{\beta'}(p') < P_{\beta'}(p)$ and σ' is the optimal response for Player 2 in the DPG $\mathcal{D} = (G, P_{\beta'})$ — a contradiction. Similarly τ^* is the optimal response to σ^*. Thus the MPG has the value

$$\bar{v} = mean(c) = \lim_{\beta\to 1}(1-\beta)v_\beta$$

in *(MD,MD)*. From Lemma 5.2.1 and since *MD* is complete for 1-Player mean payoff games, $\bar{v}$ is also the value in *(HD,HD)*. ∎

As our intuition would suggest, the optimal strategies in the MPG are the same ones which would be used for the DPG for β close to 1. In [43], it is shown how the value of the mean payoff game can be obtained from solving the discounted payoff game. It is also possible to extend the successive approximation and the strategy improvement algorithms directly to mean payoff games.

5.4.6 Complexity

Let us consider the computational complexity of solving payoff games. For a MPG $\mathcal{M} = (G, p_0, \bar{P})$, consider the following question: does Player 1 win $\mathcal{M}$? The problem is in NP because the optimal strategy τ for Player 1 can be "guessed". Then one finds the value of the 1-Player game $\mathcal{M}_\tau$ — a polynomial time problem. The "complement" of the problem — does Player 1 not win the game — is equivalent to showing that Player 2 wins the game. The "complement" problem is also in NP since determining if Player 2 wins is in NP. Therefore the problem is in $NP \cap co-NP$. Notice, to show this, essential use is made of three facts: the game has a value, both players have optimal positional strategies (and therefore the optimal strategies are of polynomial size), and 1-Player games are solvable in polynomial time. By similar reasoning, it can be shown that determining whether Player 1 wins a discounted payoff game is in $NP \cap co-NP$.

At present, no polynomial time algorithm is known for solving payoff games. Our computational experience with the policy iteration algorithm suggests that it is an efficient algorithm for solving mean payoff games and discounted payoff games. At present, it is not known whether the policy iteration algorithm is a polynomial time algorithm.

The complexity question for the problem has also attracted attention in the computer science community [9, 26, 43]. Condon [9] studies a simplified version of stochastic games called *simple stochastic games*. She points out that this problem is in $NP \cap co-NP$. Zwick and Patterson [43] study mean payoff and discounted payoff games. They provide a polynomial reduction from payoff games to simple stochastic games, hence showing that these games are perhaps easier than payoff games. Ludwig [26] gives an interesting randomized algorithm of complexity $O(2^{\sqrt{|V|}})$ for solving simple stochastic games. The same algorithm extends to mean payoff games and discounted payoff games.

5.5 Chain Games and Mean Payoff Games

A chain game has n pairs — (B_i, G_i) — where the sets B_i and G_i are disjoint. Player 1 wins the play p provided $\inf(p)$ "touches" G_i but not B_j for any $j \geq i$.

In this section, we show how to translate a chain game into a mean payoff game, so that both games have the same winner. Using this reduction, it becomes possible to use the successive approximation algorithm and the policy iteration algorithm to solve chain games.

In Section 5.5.1, we will give the method to translate a chain game into a mean payoff game. We will prove that the same player wins both the chain game and the mean payoff game. We then discuss the consequences of this result.

5.5.1 Reducing a Chain Game to a Mean Payoff Game

The new mean payoff game will be played on the same game graph as the chain game. We assign positive rewards to positions in G_i, and negative rewards to positions in B_i. The idea is to assign much larger positive rewards to positions in G_i, than positions in B_j for $j < i$; and much larger negative rewards to positions in B_k for $k \geq i$. So Player 1 wins the mean payoff game iff the play visits a position in G_i, and perhaps some positions in B_j for $j < i$, but no position in B_k for $k \geq i$.

Example 5.5.1 *A Buchi Game has a set $F \subset V$ of final states. Define the reward function where $r(f) = 1$ for $f \in F$, and $r(k) = 0$ for $k \notin F$. Then Player 1 wins the Buchi Game iff the value of the mean payoff game is v where $v > 0$. It is easy to see that the positional strategy which wins one game also wins the other.*

Let us now discuss the general method for translating a chain game into a mean payoff game. The reward function is $r : V \longrightarrow \mathcal{Z}$ where positive rewards are assigned to position in G_k, and negative rewards are assigned to positions in B_k. For $g \in G_k$, the reward is

$$r(g) > |\sum_{j=0}^{k-1} r(B_j)|.$$

For $b \in B_k$, the reward is

$$r(b) < -|\sum_{j=1}^{k} r(G_j)|.$$

When the reward function satisfies the inequalities given above, both the chain game and mean payoff game have the same winner. We next define such a reward function.

Translating a chain game into a mean payoff game:
Given a chain game $\mathcal{G} = (G, P_\phi)$ with N positions and m pairs (B_i, G_i), define the mean payoff game $\mathcal{M}_\mathcal{G} = (G, \bar{P})$ where the reward function is defined as follows:
For $g \in G_k$,

$$r(g) = N^k,$$

and for $b \in B_k$,

$$r(b) = -N^{k+1}.$$

Theorem 5.5.1 *The same player wins the chain game $\mathcal{G}$ and the mean payoff game $\mathcal{M}_\mathcal{G}$. Furthermore, the same positional strategy can be used to win both games.*

Proof: First let us show that for a cycle C, $\phi[C] = 1$ iff $r(C) > 0$. Suppose $\phi[C] = 1$ and the position with the largest index in C is v. Then $v \in G_k$ and when $B_j \cap C \neq \emptyset$, $j < k$. Therefore

$$r(C) = \sum_{c \in C} r(c) = r(v) + \sum_{w \in C \setminus \{v\}} r(w).$$

But

$$\sum_{w \in C \setminus \{v\}} r(w) \geq -(N-1)N^{k-1} > -N^k.$$

Since $r(v) = N^k$, $r(C) > 0$. Similarly, when $r(C) > 0$, $\phi[C] = 1$.

Player 1 wins the chain game $\mathcal{G}$ using positional strategy τ iff for every cycle C in G_τ reachable from start position p_0, $\phi[C] = 1$. But $\phi[C] = 1$ iff $r(C) > 0$. And $r(C) > 0$ for every cycle C in G_τ reachable from p_0 iff Player 1 wins the mean payoff game $\mathcal{M}_\mathcal{G}$ with strategy τ. Therefore Player 1 wins the chain game $\mathcal{G}$ using positional strategy τ iff he wins the mean payoff game $\mathcal{M}_\mathcal{G}$ with the same strategy. ∎

5.5.2 Some Consequences

Theorem 5.5.1 has interesting consequences. It states that a chain game is a special instance of the mean payoff game. But any game on an ω-automaton may be translated to a chain game (Theorem 5.3.2). Therefore any game on an ω-automaton can be translated to a chain game and solved as a a mean payoff game. In particular, the policy iteration algorithm may be used to solve games on ω-automata. We also get a new algorithm for model checking the propositional μ-calculus using the policy iteration algorithm.

Many of the results we discussed in Section 5.3 about games on ω-automata become special cases of the results for mean payoff games. In particular Church's questions regarding solvability and synthesis are special cases of similar questions for mean payoff games. Also the fact that *MD* is complete for chain games follows from the result that *MD* is complete for mean payoff games of which chain games are a special instance.

Bibliography

[1] R. Abraham, J.E. Marsden, and T. Raitu, *Manifolds, Tensor Analysis, and Applications*, Springer-Verlag, 1988.

[2] R. Alur et. al., The Algorithmic Analysis of Hybrid Systems, *Theoretical Computer Science*, Feb. 1995.

[3] R.Alur, C.Courcoubetis, T.A. Henzinger and P.-H. Ho, Hybrid automata: an algorithmic approach to the specification and analysis of hybrid systems,*Hybrid Systems*, LNCS 736, Springer-Verlag 1993.

[4] R. Alur and D. Dill, Automata for modeling real-time systems, *Proc. 17th ICALP*, Lecture Notes in Computer Science 443, Springer-Verlag, 1990.

[5] J.P. Aubin and A. Cellina, *Differential Inclusions*, Springer-Verlag, 1984.

[6] J.P. Aubin, *Viability Theory*, Birkhauser, 1991.

[7] V. Borkar and P. Varaiya, ϵ-Approximation of Differential Inclusion using Rectangular Differential Inclusion, Notes.

[8] A. Church, Logic, arithmetic and automata, *Proc. International Congress of Mathematicians*, 1963.

[9] A. Condon, The complexity of stochastic games, *Information and computation*, 96:203-224, 1992.

[10] A. Ehrenfeucht and J. Mycielski, Positional strategies for mean payoff games, *International Journal of Game Theory*, 1979.

[11] E. A. Emerson, Automata, tableaux, and temporal logics, *Logics of Programs*, LNCS 193, Springer-Verlag, 1985.

[12] E. A. Emerson and C.S. Jutla, The complexity of tree automata and logics of programs, *Proc. of the 29th Ann. IEEE Symposium on Foundations of Computer Science*, 1988.

[13] E.A. Emerson, C.S. Jutla, and A.P. Sistla, On model-checking for fragments of μ-calculus, *Proc. of Fifth Conference on Computer Aided Verification*, LNCS 697, Springer-Verlag, 1993.

[14] J.Frankel, L.Alvarez, R.Horowitz, and P.Li. "Robust Platoon Manuevers for AVHS," UCB-PATH TECH NOTE 94-09, University of California.

[15] D.Godbole and J.Lygeros, Longitudinal Control of the Lead Car of a Platoon. *IEEE Transactions on Vehicular Technology*, 43(4):1125-35, November 1994.

[16] J. Guckenheimer and P. Holmes, *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, Springer-Verlag, 1983.

[17] T. Henzinger, P. Kopke, A. Puri and P. Varaiya, What's Decidable About Hybrid Automata, *Proceedings of the 27th Ann. ACM Symposium on the Theory of Computing*, 1995.

[18] M. W. Hirsh and S. Smale *Differential Equations, Dynamical Systems, and Linear Algebra*, Academic Press, Inc., 1974.

[19] A.J. Hoffman and R.M. Karp, On Non-terminating Stochastic Games, *Management Science*, 12:359-370, 1966.

[20] R.A. Howard, *Dynamic Programming and Markov Processes*, M.I.T. Press, 1960.

[21] A.Hsu, F.Eskafi, S.Sachs, and P.Varaiya. Protocol Design for an Automated Highway System. *Discrete Event Dynamic Systems: Theory and Applications*, vol.2,(no.3-4):183-206, February 1993.

[22] D. Kozen, Results on propositional μ-calculus, *Theoretical Computer Science*, Dec. 1983.

[23] S.C. Krishnan, A. Puri, R.K. Brayton, and P.P. Varaiya, The Rabin index and chain automata, with applications to automata and games, *Proc. of the Seventh Conference on Computer Aided Verification*, LNCS 939, Springer-Verlag, 1995.

[24] P.R. Kumar and P. Varaiya, *Stochastic Systems: Estimation, Identification and Adaptive Control*, Prentice Hall, 1986.

[25] R.D. Luce and H. Raiffa, *Games and Decisions: Introduction and Critical Survey*, Dover Publications, 1957.

[26] W. Ludwig, A subexponential randomized algorithm for the simple stochastic game problem, *Information and Computation*, 117:151-155, 1995.

[27] R. Milner, *Communication and Concurrency*, Prentice Hall, 1989.

[28] J. Mycielski, Games with Perfect Information, *Handbook of Game Theory*, vol. 1, Elsevier Science, 1992.

[29] X.Nicollin, A.Olivero, J. Sifakis and S. Yovine, An Approach to the Description and Analysis of Hybrid Systems, *Hybrid Systems*, LNCS 736, Springer-Verlag 1993.

[30] A. Puri and P. Varaiya, Decidability of Hybrid System with Rectangular Differential Inclusions, *CAV 94: Computer-Aided Verification*, Lecture Notes in Computer Science 818, pages 95-104. Springer-Verlag, 1994.

[31] A. Puri and P. Varaiya, Verification of Hybrid Systems using Abstractions, *Hybrid Systems II*, LNCS 999, Springer-Verlag, 1995.

[32] A. Puri, V. Borkar and P. Varaiya, ϵ-Approximation of Differential Inclusions, *Proceedings of the 34th IEEE Conference on Decision and Control*, 1995.

[33] A. Puri and P. Varaiya, Driving Safely in Smart Cars, California PATH Research Report UCB-ITS-PRR-95-24, July 1995.

[34] A. Puri and P. Varaiya, Decidable Hybrid Systems, To appear in *Computer and Mathematical Modeling.*

[35] M.O. Rabin, *Automata on Infinite Objects and Church's Problem*, volume 13 of *Regional Conf. Series in Mathematics*, 1972.

[36] T.E.S. Raghavan and J.A. Filar, Algorithms for Stochastic Games - A Survey, *ZOR - Methods and Models of Operations Research*, 35:437-472, 1991.

[37] S.M. Ross, *Introduction to Stochastic Dynamic Programming*, Academic Press, 1983.

[38] L.S. Shapley, Stochastic Games, *Proceedings National Academy of Sciences*, vol. 39, 1957.

[39] C. Sparrow, *The Lorenz Equations*, Springer-Verlag, 1982.

[40] W. Thomas, Automata on Infinite Objects, *Formal Models and Semantics*, volume B of Handbook of Theoretical Computer Science, Elsevier Science, 1990.

[41] P.Varaiya. Smart Cars on Smart Roads: Problems of Control. *IEEE Transactions on Automatic Control*, 38(2):195-207, February 1993.

[42] P. P. Varaiya, On the Trajectories of a Differential System, in A.V. Balakrishnan and L.W. Neustadt, editor, *Mathematical Theory of Control*, Academic Press, 1967.

[43] U. Zwick and M. Patterson, The complexity of mean payoff games on graphs, *COCOON '95*, LNCS, Springer-Verlag.