GUILLERMO A. PÉREZ

PLAYING GAMES TO SYNTHESIZE REACTIVE SYSTEMS

Contents

Preliminaries

In this chapter, we will go through most of the mathematical notations which are used in later chapters. We will also derive some intermediate results that will be, sometimes implicitly, used in the sequel.

Notations and Conventions

We assume the reader is familiar with basic discrete mathematics and computational complexity theory. Whenever possible, we adhere to the following notational conventions. Use lowercase letters such as $a, b, \ldots$ for elementary objects; uppercase letters such as $A, B, \ldots$ for sets of elementary objects; uppercase letters in calligraphic font such as $\mathcal{A}, \mathcal{B}, \ldots$ for structures composed of other objects; lowercase Greek letters $\alpha, \beta, \ldots$ for functions and sequences; and, finally, uppercase Greek letters for sets of the latter.

The symbols $\mid$ and $:$ will be used interchangeably to separate elements of a set and the properties these must satisfy, variable quantification and formulas, etc. Both of them should be read as "such that".

Sets and functions. For a set S, we denote by $\mathcal{P}(S)$ the set of subsets of S (also referred to as the *power set* of S).

Let $\alpha : A \to B$ be a function mapping elements from the set A to elements from the set B. Let $\beta : A \nrightarrow B$ be a partial function mapping a strict subset of A to elements from B. We denote by $\mathrm{supp}(\beta)$ the *support* of β: the set $A' \subset A$ for which the function is defined. We may sometimes favour using functions, instead of partial functions, and just add a distinct new element to B which will be the value assigned by it to all $a \in A$ for which it would, otherwise, not be defined. For example, we replace β by $\kappa : A \to (B \cup \{\bot\})$ which is such that $\kappa(a) = \beta(a)$ if β is defined for a and $\kappa(a) = \bot$ otherwise. The support of κ is the set $\{a \in A \mid \kappa(a) \neq \bot\}$.

Sequences and tuples. We will write a sequence of elements as a comma-separated list in parenthesis. For instance, we could list all non-negative integers in order: $(0, 1, 2, \dots)$. Alternatively, we might use angle brackets, *i.e.* $\langle \cdot \rangle$, instead of parentheses for improved readability, (for example, in cases where we have sequences of sequences). As a third possibility, we may omit the parentheses and commas altogether in order to avoid notation saturation.

The usual sets of numbers. We denote by $\mathbb{R}$ the set of *real numbers*; $\mathbb{Q}$ the set of *rational numbers*; $\mathbb{N}$ the set of *natural numbers*—including 0; and $\mathbb{N}_{>0}$ the set of *positive integers*. We will also use the Boolean numbers $\mathbb{B} := \{0, 1\}$.

The usual notation for intervals of real numbers will be sometimes used. That is, we denote by (a, b) the set $\{x \in \mathbb{R} \mid a < x < b\}$; by $[a, b)$, the set $\{x \in \mathbb{R} \mid a \leq x < b\}$; by $(a, b]$, the set $\{x \in \mathbb{R} \mid a < x \leq b\}$; and by $[a, b]$, the set $\{x \in \mathbb{R} \mid a \leq x \leq b\}$.

Directed graphs. A directed graph is a pair (V, E) consisting of a set V of *vertices* and a set $E \subseteq V \times V$ of *edges* with a direction associated to them. That is, an edge $(u, v) \in E$ is considered to "leave" u and "arrive" at v. In these notes, all considered graphs are directed. Thus, henceforth we often omit the adjective "directed" and write digraph or just graph.

Let $\mathcal{G} = (V, E)$ be a directed graph and consider $(u, v) \in E$. The vertex v is said to be a *direct successor* of u and u is a *direct predecessor* of v. A vertex $u \in V$ has a *self-loop* if $(u, u) \in E$. A *path* is a sequence of vertices $v_0 v_1 \dots$ where v_{i+1} is a direct successor of v_i, for all $i \geq 0$; it is said to be *simple* if $v_i \neq v_j$ holds for all $0 \leq i < j$. If a path contains a vertex v, we sometimes say it "visits" or "reaches" v; if it contains the sub-sequence uv then we say it "takes" or "traverses" the edge (u, v). If there is a finite path from $u \in V$ to $v \in V$ in the graph, *i.e.* there is a path $u v_1 \dots v_n v$, then we say v is a successor of u and u is a predecessor of v; alternatively, we may say v is *reachable* from u. If a vertex $v \in V$ has no direct successor, we call it a *sink*; if it has only one direct successor and a self-loop, we say it is *trapping*. A *cycle* is a finite path $\chi = v_0 \dots v_n$ with $v_0 = v_n$; it is said to be simple if $v_i \neq v_j$ holds for all $0 \leq i < j < n$. We refer to a finite path $\lambda = v_0 \dots v_{n-1} v_n \dots v_m$, where $v_0 \dots v_{n-1}$ is a finite path and $v_n \dots v_m$ is a cycle, as a *lasso*.

Exercise 1. What is the computational complexity of determining whether u can reach v in a given graph? Can you describe an efficient algorithm to solve this problem?

Players and their preferences. In these notes we will be interested in games played by two players only. We will refer to the first one (the one we are, in some sense, "supporting") as Eve and to the second one as Adam. In most cases, Eve will be interested in maximizing some value that Adam will try to minimize.

Languages, Automata, and Topology

In these notes we study systems, and thus games, with infinite behaviours. We will repeatedly make use of infinite sequences studied in formal language theory and topology. Hence, we need to introduce some of their notation.

Languages. Consider a (possibly infinite) set A of *symbols*. A *word* on A is a sequence $a_0 a_1 \dots$ of elements from A. The special *empty word* is denoted by ε. We sometimes refer to a set of symbols, such as A, as an *alphabet* and to symbols as *letters*. Given a finite word $\alpha = a_0 \dots a_n$ and a (possibly infinite) word $\beta = a'_0 a'_1 \dots$, we denote their *concatenation*: $a_0 \dots a_n a'_0 \dots$ by $\alpha \cdot \beta$. We denote by A^* the set of all finite words on A, that is to say the set

$$\{a_0 \dots a_n \mid a_i \in A \text{ for all } 0 \leq i \leq n\}$$

of finite sequences of elements from A. The set

$$\{a_0 \dots \mid \forall i \geq 0 : a_i \in A\}$$

of infinite words on A we denote by A^ω.

Given $B \subseteq A^*$ and C a set of (infinite or finite) words on A, we denote by $B \cdot C$ the concatenation of B and C. That is, the set consisting of all words constructed by concatenating a word from B to a word from C:

$$\{\beta \cdot \kappa \mid \beta \in B, \kappa \in C\}.$$

Regular and Omega-regular Languages. A set $L \subseteq A^*$ of finite words on A is a *language* over A. A special set of languages over A, the *regular* languages, is defined recursively as follows:

- The empty language $\varnothing$ and the empty word language $\{\varepsilon\}$ are both regular languages.
- Any singleton language $\{a\}$, for $a \in A$, is a regular language over A.
- For any regular language B over A, B^* is also a regular language over A.
- For any two regular languages B, C over A, their union and concatenation is also a regular language over A.
- No other language over A is regular.

A set $L \subseteq A^\omega$ of infinite words on A is referred to as an *ω-language* over A. An ω-language L over A is ω-regular if any of the following hold:

- $L = B^\omega$, where $B \subseteq A^*$ is a non-empty regular language not containing the empty word.
- $L = B \cdot C$, where $B \subseteq A^*$ is a regular language and $C \subseteq A^\omega$ is an ω-regular language.
- $L = B \cup C$, where $B, C \subseteq A^\omega$ are both ω-regular.

Exercise 2. Is the language $L = \{w \mid \#_a(w) \geq \#_b(w)\}$, where $\#_\sigma(w)$ denotes the number of times the symbol σ occurs in w, an ω-regular language?

Omega-automata. Automata over infinite words are very similar to their finite-word counterparts: they consist of a finite set of states, transitions labelled with symbols, and an initial state. However, since the words accepted by them do not end, *i.e.* they are infinite, there are no "final" states. Thus, more involved acceptance conditions are needed.

Formally, an *automaton* is a tuple (Q, q_0, A, Δ) where Q is a finite set of *states*, $q_0 \in Q$ is the *initial state*, A is a finite alphabet (of actions), and $\Delta \subseteq Q \times A \times Q$ is the *transition relation*. We assume that Δ is *total*, in the following sense: for all $(q, a) \in Q \times A$, there is $q' \in Q$ such that $(q, a, q') \in \Delta$. If Δ is functional, *i.e.* for all $q \in Q$ and all $a \in A$ there is a unique $q' \in Q$ such that $(q, a, q') \in \Delta$, then we say the automaton is *deterministic* and write $\delta(q, a)$ to denote q'. Given a set $S \subseteq Q$ and a letter $a \in A$ we denote by

$$\mathrm{post}_a(S) := \{q \in Q \mid \exists p \in S : (p, a, q) \in \Delta\}$$

the set of a-successors of S. In a slight abuse of notation, we write $\mathrm{post}_a(q)$ instead of $\mathrm{post}_a(\{q\})$ to improve readability.

Consider an automaton $\mathcal{A} = (Q, q_0, A, \Delta)$. A *run* of $\mathcal{A}$ over an infinite word $a_0 a_1 \cdots \in A^\omega$ is a sequence $q_0 a_0 q_1 a_1 \ldots$ such that $(q_i, a_i, q_{i+1}) \in \Delta$ for all $i \geq 0$. A run $\rho = q_0 a_0 q_1 \ldots$ is then said to be *accepting* if some property—regarding the states which appear infinitely often in ρ—is fulfilled. The automaton $\mathcal{A}$ then accepts an infinite word α if it has an accepting run over α. We call the set of all infinite words which are accepted by $\mathcal{A}$ the language of $\mathcal{A}$ and denote it by $\mathcal{L}_\mathcal{A}$. We also say that $\mathcal{A}$ "recognizes" $\mathcal{L}_\mathcal{A}$. Let $\rho = q_0 a_0 \ldots$ be a run of $\mathcal{A}$. The set of states which appear infinitely often in ρ is defined as follows:

$$\mathrm{OccInf}(\rho) := \{p \in Q \mid \forall i \geq 0, \exists j \geq i : q_j = p\}.$$

We write $\rho[i]$ to denote the $(i+1)$-th state, that is q_i, in the sequence. Given indices $i, j \in \mathbb{N}$ such that $i \leq j$, we write $\rho[i..j]$ for the *infix*

$q_i a_i \dots a_{j-1} q_j$, $\rho[..i]$ for the *prefix* $q_0 a_0 \dots a_{i-1} q_i$, and $\rho[j..]$ for the *suffix* $q_j a_j \dots$ A run prefix $\pi = q_0 a_0 \dots a_{n-1} q_n$ ending in state $\text{last}(\pi) = q_n$ is said to have length $n+1$, denoted $|\pi| = n+1$.

Büchi, co-Büchi, parity, and Streett automata. Once more, let us consider an automaton $\mathcal{A} = (Q, q_0, A, \Delta)$. We will now describe several *acceptance conditions* and recall their expressive power. The following Boolean *payoff functions* can be used to define acceptance conditions for automata, *i.e.* to determine whether a run $\rho = q_0 a_0 \dots$ is accepting. One can view a Boolean payoff function as the indicator function of a *payoff set* of infinite runs. For convenience, instead of defining a function $\mathbf{Val} : (Q \cdot A)^\omega \to \mathbb{B}$ we will define $\mathbf{Val}^{-1}(1)$.

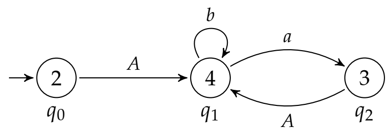

Figure 1: Omega automaton with priorities labelling the states

- The Reachability function is defined for a set $T \subseteq Q$ of *target states* as:
$$\text{Reach}^{-1}(1) := \{q_0 a_0 \dots \mid \exists i \geq 0 : q_i \in T\}.$$
- The Safety function is defined for a set $U \subseteq Q$ of *unsafe states* as follows:
$$\text{Safe}^{-1}(1) := \{q_0 a_0 \dots \mid \forall i \geq 0 : q_i \notin U\}.$$
- The Büchi function is defined for a set of *accepting* or *Büchi* states $B \subseteq Q$ as:
$$\text{Buchi}^{-1}(1) := \{\rho \mid \text{OccInf}(\rho) \cap B \neq \varnothing\}.$$
- The co-Büchi function is defined for a set of *rejecting* or *co-Büchi* states $B \subseteq Q$:
$$\text{coBuchi}^{-1}(1) := \{\rho \mid \text{OccInf}(\rho) \cap B = \varnothing\}.$$
- The parity function is defined for a *priority function* $p : Q \to \mathbb{N}$ as:
$$\text{parity}^{-1}(1) := \left\{ q_0 a_0 \dots \,\middle|\, \liminf_{i \to \infty} p(q_i) \text{ is even} \right\}.$$
- Finally, the Streett function is defined for a finite set of *Streett pairs* $\{(E_i, F_i) \mid i \in I\}$ where $E_i, F_i \subseteq Q$ for all $i \in I$. Its payoff set $\text{Streett}^{-1}(1)$ is equal to
$$\{\rho \mid \forall i \in I : E_i \cap \text{OccInf}(\rho) \neq \varnothing \text{ or } F_i \cap \text{OccInf}(\rho) = \varnothing\}.$$

The are other classical payoff functions such as *Rabin* and *Muller* but we do not make use of them here.

We refer to an automaton with a payoff function **Val**, from the list above, as an **Val** automaton. For instance, the parity automaton $\mathcal{B} = (Q, q_0, A, \Delta, p)$ accepts a word $\alpha = a_0 a_1 \dots$ if and only if it has a

Exercise 3. Are the words a^ω and ab^ω in the language of the automaton from Figure 1 in the following cases?

- with reachability acceptance condition and $T = \{q_0\}$
- with safety acceptance condition and $U = \{q_2\}$
- with Büchi acceptance condition and $B = \{q_2\}$
- with co-Büchi acceptance condition and $B = \{q_2\}$
- with parity acceptance condition

run $\rho = q_0 a_0 \dots$ on α such that $\text{parity}(\rho) = 1$. Incidentally, we refer to the number $\max\{p(q) \mid q \in Q\}$ as the *index* of the parity function p.

Intuitively, a Büchi automaton with alphabet A has as its language the set of infinite words which have some *liveness* property, that is, some event represented by a Büchi state occurs infinitely often. Co-Büchi automata have a dual acceptance condition. This can be thought of as bad events occurring only finitely often (in at least one run of the automaton). Parity and Streett automata capture, in different ways, the idea that (except for a finite number of times) bad events must always be trumped by a good event. In the case of parity, any bad event—namely the occurrence of an odd parity—can either: take place a finite number of times, or a smaller and even priority must also occur an infinite number of times (to trump it). For Streett automata, the acceptance condition can be thought of as imposing a conjunction of "conditioned obligations": for all $i \in I$, if an element from F_i is seen infinitely often, then some element from E_i must also be seen infinitely often.

Parity and Streett automata are known to be more expressive than Büchi and co-Büchi automata when restricted to being deterministic. In fact, the following is well-known about ω-regular languages and automata on infinite words.[1]

[1] Wolfgang Thomas. On the synthesis of strategies in infinite games. In *STACS*, pages 1–13. Springer, 1995; and Dominique Perrin and Jean-Eric Pin. *Infinite words: automata, semigroups, logic and games*, volume 141. Academic Press, 2004

Proposition 1. *For any alphabet A and ω-language L over A, the following are equivalent:*

- *L is ω-regular.*
- *There exists a Büchi automaton $\mathcal{A}$ such that $\mathcal{L}_{\mathcal{A}} = L$.*
- *There exists a deterministic parity automaton $\mathcal{A}$ such that $\mathcal{L}_{\mathcal{A}} = L$.*

Of particular interest to us in the context of these notes is the fact that any non-deterministic Büchi automaton can be *determinized* into a parity automaton.[2] In other words, for a given Büchi automaton, one can construct a deterministic parity automaton with exactly the same language. More formally:

[2] Shmuel Safra. On the complexity of ω-automata. In *FOCS*, pages 319–327. IEEE, 1988; Shmuel Safra. Exponential determinization for omega-automata with strong-fairness acceptance condition (extended abstract). In *STOC*, pages 275–282, 1992. DOI: 10.1145/129712.129739; and Nir Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *LMCS*, 3(3), 2007. DOI: 10.2168/LMCS-3(3:5)2007. URL http://dx.doi.org/10.2168/LMCS-3(3:5)2007

Proposition 2 (Determinization of omega-automata)**.** *Given a Büchi automaton $\mathcal{A} = (Q, q_0, A, \Delta, B)$, there is an algorithm which yields a deterministic parity automaton $\mathcal{A}' = (Q', q_0', A, \Delta', p)$ such that $|Q'|$ is of size $2^{\mathcal{O}(|Q| \log |Q|)}$ and with parity index polynomial with respect to $|Q|$.*

Borel hierarchy. Let A be a (possibly infinite) alphabet. The *Borel hierarchy* of subsets of A^ω is inductively defined as follows.

- $\Sigma_1^0 = \{W \cdot A^\omega \mid W \subseteq A^*\}$ is the set of *open subsets*[3] of A^ω.
- For all $n \geq 1$, $\Pi_n^0 = \{A^\omega \setminus L \mid L \in \Sigma_n^0\}$ consists of the complement of sets in Σ_n^0.

[3] Here, we are extending the familiar notion of open set of numbers, *e.g.* $0 < x < 1$, to sets of infinite words. A set $L \subseteq A^\omega$ is open if every word $w \in L$ has a non-trivial prefix p such that all words $pw' \in A^\omega$ are also in L.

- For all $n \geq 1$, $\Sigma^0_{n+1} = \{\bigcup_{i \in \mathbb{N}} L_i \mid \forall i \in \mathbb{N} : L_i \in \Pi^0_n\}$ is the set obtained by countable unions of sets in Π^0_n.

A map $f : A \to B$ is said to be *Borel measurable* if $f^{-1}(X)$ is Borel for any open subset X of B.

Quantitative Payoff Functions

As the title of these notes implies, we will focus mainly on non-Boolean payoff functions. We will now define some classical functions of the form

$$\mathbb{Q}^\omega \to (\mathbb{R} \cup \{-\infty, +\infty\}).$$

Formally, for an infinite sequence of rationals $\chi = x_0 x_1 \dots$ we define:

- the Inf (Sup) payoff, is the minimum (maximum) rational seen along the sequence:

$$\mathrm{Inf}(\chi) := \inf\{x_i \mid i \geq 0\}$$

  and

$$\mathrm{Sup}(\chi) := \sup\{x_i \mid i \geq 0\};$$

- the LimInf (LimSup) payoff, is the minimum (maximum) rational seen infinitely often:

$$\mathrm{LimInf}(\chi) := \liminf_{i \to \infty} x_i$$

  and, respectively, we have that

$$\mathrm{LimSup}(\chi) := \limsup_{i \to \infty} x_i;$$

- the *mean-payoff* value of a sequence, *i.e.* the limiting average rational, defined using lim inf or lim sup since the running averages might not converge:

$$\underline{\mathrm{MP}}(\chi) := \liminf_{k \to \infty} \frac{1}{k+1} \sum_{i=0}^{k} x_i$$

  and

$$\overline{\mathrm{MP}}(\chi) := \limsup_{k \to \infty} \frac{1}{k+1} \sum_{i=0}^{k} x_i.$$

In words, $\underline{\mathrm{MP}}$ corresponds to the *limit inferior* of the average of increasingly longer prefixes of χ while $\overline{\mathrm{MP}}$ is defined as the *limit superior* of χ.

Exercise 4. Consider the sequence

$$\alpha = 1, -1, 1, 1, -1, 1, 1, 1, 1, \dots$$

where the i-th -1 is followed by 2^i occurrences of 1. What are the values of $\underline{\mathrm{MP}}(\alpha)$ and $\overline{\mathrm{MP}}(\alpha)$?

Another payoff function we consider is the *discounted sum*. Given a sequence of rationals $\chi = x_0 x_1 \dots$ of length $n \in \mathbb{N} \cup \{\infty\}$, the *discounted sum* is defined for a rational discount factor $\lambda \in (0,1)$ as follows:

$$\mathrm{DS}_\lambda(\chi) := \sum_{i=0}^{n} \lambda^i x_i.$$

We remark the above payoff functions together with the Boolean functions—Büchi, parity, etc.—are all Borel measurable. Note that, since they map sequences of rationals to real numbers (or infinity) then it suffices to show that, for all $a \in \mathbb{R}$, the set of sequences with value above a is Borel. Formally, we have that:

Exercise 5. With $\chi = 3^\omega$ and $\lambda = \frac{3}{4}$, what is the value of $\mathrm{DS}_\lambda(\chi)$?

Proposition 3. *Let $\rhd \in \{>, \geq\}$. For all $a \in \mathbb{R}$, for any Boolean or quantitative payoff function* **Val***, the set $\{\chi = x_0 x_1 \dots \mid \mathbf{Val}(\chi) \rhd a\}$ is Borel.*

Indeed, it is easy to use the definition of the payoff functions we have presented thus far, and convince oneself that the above holds. For example, for $a \in \mathbb{R}$ and $\rhd$ set to $\geq$, the payoff function $\underline{\mathrm{MP}}$ yields:

$$\left\{ x_0 x_1 \dots \in \mathbb{Q}^\omega \;\middle|\; \bigcap_{i \in \mathbb{N}_{>0}} \bigcup_{j \in \mathbb{N}} \bigcap_{k \geq j} \frac{1}{k+1} \sum_{\ell=0}^{k} x_\ell \geq a - \frac{1}{i} \right\}$$

Exercise 6. Using the mean-payoff example, prove that the parity and co-Büchi payoff functions are Borel when $\rhd = \geq$ and $a = 1$.

which is clearly Borel. It follows that all Boolean and quantitative payoff functions defined in this chapter are Borel-measurable functions.

Proposition 4. *All Boolean and quantitative payoff functions are Borel measurable.*

Prefix independence. A payoff function **Val** is said to be *prefix independent* if for any two sequences of rationals $\chi = x_0 x_1 \dots, \chi' = x'_0 x'_1 \dots \in \mathbb{Q}^\omega$ the following holds:

$$\left(\exists i \geq 0, \forall j \geq i : x_j = x'_j\right) \implies \mathbf{Val}(\chi) = \mathbf{Val}(\chi').$$

In other words, any two sequences with the same infinite suffix (starting from any point onward) will have the same value. This can be seen as the payoff function "not caring" about the prefix of the sequence. Hence the name, prefix independent.

Computational Complexity

Throughout these notes, we follow classical notation and definitions for concepts regarding computational complexity.[4] We regard algorithms which have polynomial worst-case running time as "efficient". Thus, we shall provide polynomial-time reductions when proving

[4] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979; and Christos H. Papadimitriou. *Computational complexity*. John Wiley and Sons Ltd., 2003

hardness results. Furthermore, we use *big-O* notation—*i.e.* $\mathcal{O}(\cdot)$—to describe the limiting behaviour of functions (as was done in Proposition 2).

As we work with games played on weighted structures, let us comment on the format of the input of their decision problems. Unless explicitly stated otherwise, all weights labelling considered structures are given in binary. Furthermore, parameters for payoff functions—such as the discount factor λ required for the discounted sum function—are also given as input and in binary. Thus, an algorithm with worst-case running time $\mathcal{O}(\lambda)$ is of *pseudo-polynomial* running time. That is, polynomial in the numeric value of λ yet exponential in the size of its representation.

Quantitative Games

The main mathematical object studied throughout these notes is that of *two-player quantitative games* played on finite structures.

In its most general form, these games are played on a finite directed graph with rational numbers labelling the edges: a directed weighted graph. We shall call the structure on which a quantitative game is played a *(weighted) arena*.

Games Played on Graphs

In this section, we shall first present all the definitions required in order to state well-known results on quantitative games played on graphs. Furthermore, we will recall some of these classical results since they will be used in several parts of these notes.

Quantitative games are played by two players: Eve and Adam. We will partition the vertices of an arena into those owned by Eve and those owned by Adam.

Definition (Weighted arena)**.** A *(weighted) arena* (or WA, for short) is a tuple $\mathcal{G} = (V, V_\exists, v_I, E, w)$ where (V, E) is a finite digraph, $V_\exists \subseteq V$ is the set of vertices belonging to Eve, $v_I \in V$ is the initial vertex, and $w : E \to \mathbb{Q}$ is a rational weight function assigning weights to the edges of the graph.

Since we will focus on infinite paths in arenas, we will assume that the underlying digraph of any arena has no sinks. We depict vertices owned by Eve (*i.e.* those in $V_\exists$) with squares and vertices owned by Adam (*i.e.* those in $V \setminus V_\exists$) with circles. We denote the maximum absolute value of a weight in an arena by $w_{\max}$. (See Figure 2 for an example of a weighted arena.)

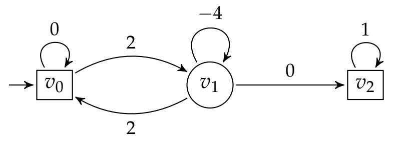

Figure 2: Example weighted arena with an Eve vertex, v_0, on the left and an Adam vertex, v_1, on the right; in this arena, $w_{\max}$ is equal to 4

A game played on a weighted arena by Eve and Adam proceeds in rounds as follows. Initially, the "current vertex" is v_I, that is the initial vertex. From the current vertex u, if $u \in V_\exists$ then Eve chooses a direct successor v of u, otherwise Adam chooses a direct successor v of u. The process is then repeated from v. This interaction determines an infinite path in the arena. We shall call such an infinite path, a *play*.

More formal definitions for these concepts follow.

Definition (Plays and prefixes)**.** A *play* in an arena $(V, V_\exists, v_I, E, w)$ is an infinite path in the digraph (V, E). In other words, a play is an infinite sequence of vertices $\pi = v_0 v_1 \ldots$ where $v_0 = v_I$ and $(v_i, v_{i+1}) \in E$ for all $i \geq 0$; a *play prefix* is then a finite path starting from v_I.

In the weighted arena depicted in Figure 2, examples of plays are v_0^ω, $v_0 \cdot v_1^\omega$, and $(v_0 \cdot v_1)^\omega$. An example of a sequence of vertices which is not a valid play is $v_0 v_1 v_2 \cdot v_0^\omega$ since there is no edge from v_2 to v_0 and therefore the sequence is not a path.

Definition (Strategies)**.** Consider an arena $\mathcal{G} = (V, V_\exists, v_I, E, w)$. A *strategy for* Eve (respectively, Adam) in $\mathcal{G}$ is a function that maps play prefixes ending with a vertex v in $V_\exists$ $(V \setminus V_\exists)$ to a direct successor of v.

Consider once more the arena from Figure 2. A sample strategy $\sigma : V^* \cdot V_\exists \to V$ for Eve is: from v_2 always play to v_2, and for any play prefix $\rho \cdot v_0$ play v_1 if the length of ρ is even and v_0 otherwise. An example of a strategy $\tau : V^* \cdot (V \setminus V_\exists)$ for Adam could be play from v_1 to v_1 always.

Exercise 7. In the game from Figure 2, does Eve have a strategy to ensure her mean-payoff value is non-negative?

We say a strategy for Eve σ has *memory* m if there are: a non-empty set M with $|M| = m$, an element $m_0 \in M$, and functions $\alpha_u : M \times V \to M$ and $\alpha_o : M \times V_\exists \to V$ such that for any play prefix $\rho = v_0 \ldots v_n$, we have $\sigma(\rho) = \alpha_o(m_n, v_n)$, where m_n is defined inductively by $m_{i+1} = \alpha_u(m_i, v_i)$ for $i \geq 0$. The memory of a strategy of Adam is defined analogously. A strategy is said to have *finite memory* if $m \in \mathbb{N}_{>0}$. The tuple $(M, m_0, \alpha_u, \alpha_o)$ is sometimes referred to as the *Mealy machine realizing* σ.[5]

[5] Marcus Gelderie. *Strategy machines: representation and complexity of strategies in infinite games*. PhD thesis, RWTH Aachen University, 2014. URL `http://darwin.bth.rwth-aachen.de/opus3/volltexte/2014/5025`

Definition (Sets of strategies)**.** Consider an arena $\mathcal{G} = (V, V_\exists, v_I, E, w)$. We denote by $\Sigma_\exists(\mathcal{G})$ and $\Sigma_\forall(\mathcal{G})$ the sets of all strategies for Eve and, respectively, for Adam in $\mathcal{G}$; by $\mathrm{FM}^m_\exists(\mathcal{G})$ and $\mathrm{FM}^m_\forall(\mathcal{G})$ the sets of all strategies with memory m for both players.

Exercise 8. Describe the Mealy machine that corresponds to the strategy Eve for the game in Figure 2 which consists in playing from v_0 to v_1 every other time v_0 is visited and to v_0 otherwise.

Also, if $\mathcal{G}$ is clear from the context, we shall omit it.

A strategy for either player with memory 1 is said to be *positional* or *memoryless*. Let $(M, m_0, \alpha_u, \alpha_o)$ realize a strategy for one of the players. Observe that when $|M| = 1$ we then have that $M = \{m_0\}$ and thus α_o does not depend on its first parameter (since it is always m_0). A positional strategy can then be expressed as a function from vertices to vertices as follows:

$$\sigma : V_\exists \to V \text{ (or } \tau : (V \setminus V_\exists) \to V, \text{ respectively).}$$

A play $\pi = v_0 \ldots$ in an arena $\mathcal{G}$ is said to be *consistent* with a strategy σ (respectively, τ) for Eve (Adam) if for all $i \geq 0$ it holds that

$v_i \in V_\exists$ ($v_i \notin V_\exists$) implies that:

$$\sigma(\langle v_j \rangle_0^i) = v_{i+1} \; (\tau(\langle v_j \rangle_0^i) = v_{i+1}).$$

Definition (Outcome of strategies)**.** Consider an arena $(V, V_\exists, v_I, E, w)$. Given strategies σ and τ for Eve and Adam, respectively, we denote by $\pi^v_{\sigma\tau}$ the unique play starting from $v \in V$ that is consistent with σ and τ.

In the $\pi^v_{\sigma\tau}$ notation, if v is omitted we assume it is v_I.

While fixing a strategy for each player results in a unique play induced by them, fixing a strategy for only one of them yields a set of plays consistent with it. One way to represent the latter set is to "combine" the weighted arena with a given strategy. More formally,

Definition (Product of an arena and a strategy)**.** Consider an arena $\mathcal{G} = (V, V_\exists, v_I, E, w)$ and a strategy σ for Eve in $\mathcal{G}$ realized by the Mealy machine $(M, m_0, \alpha_u, \alpha_o)$. We denote by $\mathcal{G} \times \sigma$ their *product*, that is the arena $(V \times M, \varnothing, (v_I, m_0), E', w')$ where:

- E' has an edge $((t, m), (v, n))$ if only if $(t, v) \in E$, $\alpha_u(m, t) = m$, and $t \in V_\exists \implies \alpha_o(m, t) = v$;
- w' maps $((t, m), (v, n))$ to $w(t, v)$ for all $(t, m), (v, n) \in V \times M$.

The product of $\mathcal{G}$ with a strategy for Adam is defined similarly.

Intuitively, the product of an arena and a strategy is a weighted arena in which Eve no longer controls any vertex since her choices have been fixed according to the strategy. If the strategy used finite memory, the resulting weighted arena is finite. Note that there is indeed a one-to-one correspondence between plays consistent with a strategy σ for a player in $\mathcal{G}$ and plays in $\mathcal{G} \times \sigma$.

Exercise 9. Describe the product of the strategy from the previous exercise and the game.

So far, we have defined the type of arena on which the games we study will be played. Additionally, we have formalized the notion of strategy for each of the two players who will take part in the games. Now, we must assign a value to a play, so as to determine how much each player gains from having witnessed it. We shall do so by using *payoff functions* of the form $\mathbb{Q}^\omega \to \mathbb{R}$.

Definition (Value of a play)**.** Consider an arena $\mathcal{G} = (V, V_\exists, v_I, E, w)$ and a payoff function $\mathbf{Val} : \mathbb{Q}^\omega \to \mathbb{R}$. The *value of a play* $\pi = v_0 v_1 \ldots$ in $\mathcal{G}$ is denoted by $\mathbf{Val}(\pi)$ and defined as:

$$\mathbf{Val}(w(v_0, v_1) w(v_1, v_2) \ldots).$$

For simplicity, we denote the value of a play starting from v and consistent with strategies σ and τ, for Eve and Adam respectively, as

$$\mathbf{Val}^v(\sigma, \tau) := \mathbf{Val}(\pi^v_{\sigma\tau}).$$

For the case when $v = v_I$ we write simply $\mathbf{Val}(\sigma, \tau)$, *i.e.* we omit v.

The above definition of the value of a play concludes the set of basic definitions required to determine a quantitative game. Indeed, what we refer to as a "quantitative game" is merely a weighted arena together with a payoff function. Examples of such payoff functions are Büchi, parity, Inf, Sup, LimInf, and LimSup. [6] For simplicity, we refer to the quantitative game consisting of arena $\mathcal{G}$ and payoff function $\mathbf{Val}$ as a $\mathbf{Val}$ game, *e.g.* an "$\underline{\text{MP}}$ game" a.k.a. a mean-payoff game. Whenever the payoff function is understood from the context, we directly speak of an arena as being a game.

[6] Note that for quantitative payoff functions such as LimSup, it is not the case that for any sequence of rationals χ we have $\mathbf{Val}(\chi) \in \mathbb{R}$. Indeed, their values might be $-\infty$ or $+\infty$ for some sequences of rationals. However, when applied to a sequence of rational weights obtained from a weighted arena, they are easily seen to be bounded by a function of $w_{\max}$.

A *Boolean game* is a particular case of quantitative game. The payoff functions for such games are of the form $V^\omega \to \mathbb{B}$ and are obtained by adapting the Boolean payoff functions for automata runs, such as Büchi and parity, to sequences of vertices.

Winning condition

By far the most widely studied *solution concept* for quantitative games used for reactive synthesis is that of *winning strategies* (we shall define winning strategies briefly). In game theory, a solution concept is a rule predicting how a game will or should be played. As we are interested in how Eve should behave, we are looking for a solution concept for Eve. Intuitively, a winning strategy for her is a strategy which ensures some property regardless of what strategy Adam plays. This is an extremely robust concept—hence its popularity. In order to properly define it, we need some more notation.

Let $\mathcal{G}$ be a quantitative game with weighted arena $(V, V_\exists, v_I, E, w)$ and payoff function $\mathbf{Val}$. In order for a strategy for Eve in $\mathcal{G}$ to be declared "winning", we need a *winning condition* (a.k.a. *objective*) for the game: a subset of the plays which are desirable (or winning) for Eve. In quantitative games, a winning condition is determined by a *threshold*. For instance, if we fix $\nu \in \mathbb{R}$ we can then say that a play π in $\mathcal{G}$ is winning for Eve if it has a value of at least ν, that is, if it holds that $\mathbf{Val}(\pi) \geq \nu$. Hence, we have a partition of the set of plays into winning and losing for Eve: the set $\{\pi \mid \mathbf{Val}(\pi) \geq \nu\}$ is winning for her, and the complement is losing for her. Two different, yet interrelated questions, arise from this definition. The first, asks, for a given winning condition, whether a player can ensure to win against any behavior of his adversary. The second, asks what is the maximal or minimal threshold for which he can do the latter. We focus here on the first question and deal with the second question in the following section.

Problem (Deciding a game)**.** Given a quantitative game with weighted arena $(V, V_\exists, v_I, E, w)$, payoff function $\mathbf{Val}$, and threshold ν, determine whether there exists a strategy σ for Eve in $\mathcal{G}$ such that, against any strategy τ for Adam in $\mathcal{G}$ it holds that $\mathbf{Val}(\sigma, \tau) \geq \nu$. If the latter holds, then the strategy of Eve witnessing it, is called a *winning strategy* for her. If Eve has a winning strategy in a game, she wins that game.

Similarly, a winning strategy for Adam is a strategy such that, against any strategy of Eve, the resulting play is not winning for Eve. If Adam has a winning strategy, we say he wins the game.

Note that, even if we can determine whether Eve wins a game, it is not immediate how to obtain a winning strategy for her. Further, determining the winner of a game might be "easier", in terms of computational complexity, than to obtain a winning strategy. This motivates the following search problem.

Problem (Winning strategy synthesis)**.** Given a quantitative game with arena $(V, V_\exists, v_I, E, w)$, payoff function **Val**, and threshold ν, if Eve wins the game output a strategy σ for her such that:

$$\inf_{\tau \in \Sigma_\forall} \mathbf{Val}(\sigma, \tau) \geq \nu.$$

Both problems stated above have been centered on Eve. A useful property of all quantitative games considered in these notes is that: if Eve does not have a winning strategy, then necessarily Adam has one. A game with this property is said to be *determined*. Let us formalize these claims.

Definition (Determinacy)**.** A quantitative game $\mathcal{G}$ is said to be determined if: either Eve has a winning strategy, or Adam has a winning strategy.

A quantitative game being determined can be seen as a kind of "quantifier swap" property which holds for the logic formulas stating the existence of winning strategies for the players.[7] In a determined quantitative game, deciding the game is equivalent to determining the winner of the game.

[7] In general, it does not hold that $\neg(\exists A, \forall B : \varphi)$ implies $\exists B, \forall A : \neg\varphi$ and that is why determined games are so special.

A very general result due to Martin[8] implies that any quantitative game with a Borel winning condition is guaranteed to be determined. All winning conditions studied in these notes can easily be shown to be Borel subsets of the set of all plays (see Proposition 3).

[8] Donald A. Martin. Borel determinacy. *The annals of Mathematics*, 102(2):363–371, 1975

Theorem 1 (Borel determinacy)**.** *For all weighted arenas $\mathcal{G} = (V, V_\exists, v_I, E, w)$, for all payoff functions* **Val**, *for all thresholds $\nu \in \mathbb{R}$, if the corresponding set of winning plays $W \subseteq V^\omega$ is a Borel subset of the set of all plays in $\mathcal{G}$ then the* **Val** *game played on $\mathcal{G}$ is determined.*

The above result will prove to be extremely useful throughout these notes. More specific statements hold for different quantitative games, *e.g.* in mean-payoff games one of the two players has a winning positional strategy for every threshold. We recall these and other results regarding classical games later. In the next section we will be interested in the maximal threshold for which Eve is guaranteed to have a winning strategy.

Values of a game

Previously, we have asked whether a player can enforce outcomes with value of at least a given threshold (or, respectively, at most a threshold). However, instead of fixing a threshold, we could directly define the *value of a game*. Since Eve is attempting to maximize the value of the witnessed play and Adam is trying to do the opposite, we have the following two alternative values of a game in which the players are *antagonistic*:

$$\begin{aligned}\overline{\mathbf{aVal}^v(\mathcal{G})} &:= \sup_{\sigma\in\Sigma_\exists(\mathcal{G})} \inf_{\tau\in\Sigma_\forall(\mathcal{G})} \mathbf{Val}^v(\sigma,\tau), \text{ and}\\ \underline{\mathbf{aVal}^v(\mathcal{G})} &:= \inf_{\tau\in\Sigma_\forall(\mathcal{G})} \sup_{\sigma\in\Sigma_\exists(\mathcal{G})} \mathbf{Val}^v(\sigma,\tau).\end{aligned}$$

It should be clear that the following relation among the two values trivially holds for all $v \in V$:

$$\underline{\mathbf{aVal}^v(\mathcal{G})} \leq \overline{\mathbf{aVal}^v(\mathcal{G})}.$$

Furthermore, one can show—using Borel determinacy or directly applying another result of Martin[9]—that if **Val** is bounded and Borel-measurable then both are equivalent.

[9] Donald A. Martin. The determinacy of blackwell games. *The Journal of Symbolic Logic*, 63(4):1565–1581, 1998

Theorem 2 ((Unique) antagonistic value)**.** *For any quantitative game with arena $\mathcal{G} = (V, V_\exists, v_I, E, w)$ and bounded Borel-measurable payoff function* **Val***, for all $v \in V$ it holds that $\underline{\mathbf{aVal}^v(\mathcal{G})} = \overline{\mathbf{aVal}^v(\mathcal{G})}$.*

Henceforth, we shall refer to

$$\mathbf{aVal}^v(\mathcal{G}) := \underline{\mathbf{aVal}^v(\mathcal{G})} = \overline{\mathbf{aVal}^v(\mathcal{G})}$$

as the *antagonistic value* of a game $\mathcal{G}$ (and we will omit $\mathcal{G}$, as usual, if it is clear from the context, or v if it is assumed to be v_I).

Let us remark that all payoff functions considered in these notes are in fact bounded and Borel-measurable. (Measurability was already argued in Proposition 4. All payoff functions used in these notes can be shown to be bounded by a function of $w_{\max}$ due to the finiteness of the arenas.) We thus consider the problem of computing the unique antagonistic value of a given game.

Problem (Computing the value of a game)**.** Given a quantitative game $\mathcal{G}$, output its antagonistic value $\mathbf{aVal}^{v_I}(\mathcal{G})$.

Following the definitions of the antagonistic value of a game and of the problem of deciding a game, one might be tempted to say that the former (an optimization problem) is harder than the latter. However, even if the antagonistic value of a game turns out to be μ, this does not imply that there is a strategy for Eve which actually achieves at least μ against any strategy for Adam. We will later see that for some classical quantitative games, this intuition does turn out to be correct and *worst-case optimal strategies* do exist for both players.

Definition (Worst-case optimal strategies)**.** In a quantitative game $\mathcal{G}$ with weighted arena $(V, V_\exists, v_I, E, w)$ and payoff function **Val**,

- a strategy σ for Eve is said to be *worst-case optimal (maximizing)* from $v \in V$ if it holds that $\inf_{\tau \in \Sigma_\forall} \mathbf{Val}^v(\sigma, \tau) = \mathbf{aVal}^v(G)$, and
- a strategy τ for Adam is said to be *worst-case optimal (minimizing)* from $v \in V$ if it holds that $\sup_{\sigma \in \Sigma_\exists} \mathbf{Val}^v(\sigma, \tau) = \mathbf{aVal}^v(G)$.

Thus far we have considered the case where Adam attempts to witness a play with a small value. One can also study the case in which both Eve and Adam *co-operatively* maximize the same value. We then have the *co-operative value* of a quantitative game $\mathcal{G}$ played on arena $(V, V_\exists, v_I, E, w)$ with payoff function **Val**:

Exercise 10. Is "staying in v_0 forever" a worst-case optimal strategy for Eve in the game from Figure 2 with the mean-payoff function?

$$\mathbf{cVal}^v(\mathcal{G}) := \sup_{\sigma \in \Sigma_\exists} \sup_{\tau \in \Sigma_\forall} \mathbf{Val}^v(\sigma, \tau)$$

where $v \in V$.

Definition (Best-case optimal strategies)**.** In a quantitative game $\mathcal{G}$ with weighted arena $(V, V_\exists, v_I, E, w)$ and payoff function **Val**, a pair of strategies σ and τ for Eve and Adam, respectively, is said to be *best-case optimal* from $v \in V$ if $\mathbf{Val}^v(\sigma, \tau) = \mathbf{cVal}^v(\mathcal{G})$.

We observe that this scenario can be reduced to a one-player game: a game in which Eve owns all the vertices.

We conclude this section on the values of a quantitative game by observing that, for Boolean games, computing the antagonistic values is not really an interesting problem.[10] The only interesting question in these games is whether Eve wins with threshold 1. Henceforth, we shall implicitly assume, for Boolean games, a threshold $\nu = 1$.

Exercise 11. Describe best-case optimal strategies for both players in the game from Figure 2 with the mean-payoff function. What is the co-operative value of the game?

[10] Indeed, by definition, we have that for any Boolean game $\mathcal{G}$ it holds that $\mathbf{cVal}(\mathcal{G}), \mathbf{aVal}(\mathcal{G}) \in \mathbb{B}$. Also note, for threshold $\nu = 0$ Eve always has a winning strategy and for any threshold $\nu > 1$ Adam always has a winning strategy. Furthermore, Eve wins the game for any threshold $0 < \nu < 1$ if and only if she wins for threshold $\nu = 1$. Thus, the only interesting question in these games is whether Eve wins the game with threshold 1.

Classical games

In this section, we will recall the definitions and properties of several classical quantitative games. First, we will define all the Boolean games used in the following chapters. We then focus on non-Boolean quantitative games.

Boolean games

Reachability, Safety, Büchi, co-Büchi, parity, and Streett games are defined using the corresponding Boolean payoff functions. An interesting property about most of the aforementioned games is that they are positionally determined.[11] More formally:

[11] Krzysztof R. Apt and Erich Grädel. *Lectures in game theory for computer scientists*. Cambridge University Press, 2011

Theorem 3 (Positional determinacy)**.** *Reachability, safety, Büchi, co-Büchi, and parity games are* positionally determined*: either Eve has a positional winning strategy or Adam has a positional winning strategy.*

Additionally, determining the winner of all but parity and Streett games, can be done in polynomial time.[12]

[12] E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs. *SIAM J. Comput.*, 29(1):132–158, 1999

Theorem 4 (Complexity of determining the winner)**.** *Determining the winner of a reachability, safety, Büchi, or co-Büchi game can be done in polynomial time. Determining the winner of a parity game is in* **NP** $\cap$ **coNP***. Determining the winner of a Streett game is* **coNP***-complete.*

There is no known polynomial-time algorithm to determine the winner of a parity game. Membership of the problem in **NP** $\cap$ **coNP** follows from the games being positionally determined, and the fact that one-player parity games are in polynomial time (hence, one can guess a strategy for a player and verify if the opponent can beat it). Its exact complexity is considered to be one of the most important open problems in formal verification. (It is known that the model checking problem for the modal μ-calculus reduces to determining the winner of a parity game.)

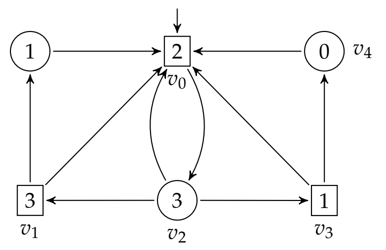

Figure 3: Parity game won by Eve; vertices are labelled with their priorities

Example 1 (A parity-game example)**.** Consider the parity game depicted in Figure 3. From Theorem 3 it follows that if Eve has a strategy to ensure the minimal priority seen infinitely often is even, then she also has a positional strategy which ensures the same property. Note that the only vertices at which players have a choice of successor are v_1, v_2, and v_3. Let us consider the strategy for Eve which corresponds to the mapping $v_1 \mapsto v_0$ and $v_3 \mapsto v_4$. Again, from Theorem 3, we know that if Adam can beat this strategy for Eve (*i.e.* force a play consistent with it for which the minimal priority seen infinitely often is odd) then he has a positional strategy to do so. Clearly, if Adam plays to v_1 from v_2 then the outcome of the strategies will be the play $(v_0v_2v_1)^\omega$ with value 1 since the priorities of the vertices are, respectively, 2, 3, and 3. If instead Adam plays to v_3 from v_1, then the resulting outcome $(v_0v_2v_3v_4)^\omega$ has value 1—since the priority of v_4 is 0 and it is the minimal priority in the arena. Finally, if Adam plays to v_0 from v_3 then the outcome is (v_0v_2) and its value is 1 since their priorities are 2 and 3. We conclude that the described strategy for Eve is winning.

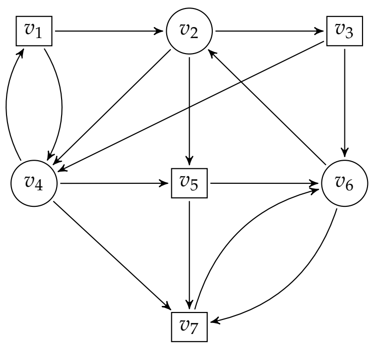

Figure 4: Yet another arena

Exercise 12. Consider the game from Figure 4 and suppose that Adam wants to ensure at most two distinct vertices are visited. From which vertices can he win against Eve? Describe a strategy of his that witnesses the fact. What if we allow at most three vertices?

Quantitative games

Games defined using the quantitative payoff functions are the main focus of these notes. It is not hard to see that Inf games generalize safety games; as Sup does reachability; LimInf does co-Büchi; and LimSup does Büchi. Conversely, for any threshold, one can reduce the quantitative game to the Boolean game it generalizes.[13] It follows that, for any threshold, these four quantitative games are positionally determined and solvable in polynomial time. For Inf, Sup, LimInf, and LimSup, the above already implies that in those games there always are worst-case optimal strategies for both players.[14] For mean payoff and discounted sum, the same property holds.[15] Thus, we have:

[13] For instance, for LimSup and a threshold ν we can mark as Büchi edges all of those with weight at least ν and then play a Büchi game on the resulting arena. Clearly, Eve wins in the new game if and only if she wins in the original one.

[14] Also best-case optimal pairs of strategies! This is actually a sub-case.

[15] Andrzej Ehrenfeucht and Jan Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8:109–113, 1979. ISSN 0020-7276. DOI: 10.1007/BF01768705; and Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *TCS*, 158 (1):343–359, 1996

Theorem 5 (Existence of optimal strategies)**.** *For all quantitative games with payoff function* Inf, Sup, LimInf, LimSup, *mean payoff, or discounted sum, the following hold:*

- *there exists $\sigma \in \Sigma_\exists$ which is positional and worst-case optimal maximizing from all $v \in V$,*
- *there exists $\tau \in \Sigma_\forall$ which is positional and worst-case optimal minimizing from all $v \in V$,*
- *there are $\sigma \in \Sigma_\exists$ and $\tau \in \Sigma_\forall$ which are positional best-case optimal from all $v \in V$.*

In terms of the complexity of solving the games, for all but mean payoff and discounted sum, we have already argued polynomial-time algorithms exist. Based on the above result we then get the following:

Theorem 6 (Complexity of determining the winner)**.** *Determining the winner of a* Inf, Sup, LimInf, *or* LimSup *game can be done in polynomial time. Determining the winner of a mean-payoff or discounted-sum game is in* ***NP*** $\cap$ ***coNP****.*

A reduction from mean-payoff games to discounted-sum games due to Zwick and Paterson tells us that reducing the complexity upper bound of the winner determination problem for discounted-sum games would result in a new upper bound for the complexity of the same problem for mean-payoff games. Additionally, Jurdziński has established a reduction from parity to mean-payoff games[16] which gives an analogue of the latter for these two games. It follows that improving the upper bound for that problem, for mean-payoff or discounted-sum games, would be a major development in the area of verification.

[16] Marcin Jurdziński. Deciding the winner in parity games is in **UP** $\cap$ **coUP**. *Information Processing Letters*, 68(3): 119–124, 1998

Although no polynomial-time algorithm is known to determine the winner of a mean-payoff or discounted-sum game, pseudo-polynomial algorithms have been discovered.[17]

[17] Lubos Brim, Jakub Chaloupka, Laurent Doyen, Raffaella Gentilini, and Jean-François Raskin. Faster algorithms for mean-payoff games. *Formal Methods in System Design*, 38(2): 97–118, 2011. DOI: 10.1007/s10703-010-0105-x. URL http://dx.doi.org/10.1007/s10703-010-0105-x

Theorem 7. *The antagonistic value of a mean-payoff game can be computed in pseudo-polynomial time, i.e. in polynomial time w.r.t.* $|V|$, $|E|$, *and* $w_{\max}$. *The antagonistic value of a discounted-sum game can be computed in pseudo-polynomial time, i.e. polynomial w.r.t.* $|V|$, $|E|$, $\log_2 w_{\max}$, *and* λ.

We have two final remarks on the antagonistic and co-operative values of all games considered above. It follows from Theorem 3 and Theorem 6 that they can be easily represented in binary.[18]

Theorem 8 (Representation of the values of a game)**.** *For all quantitative games, both its co-operative and antagonistic values, i.e.* **cVal** *and* **aVal**, *are representable using a polynomial number of bits.*

Also, the co-operative value of a game is easy to compute.[19]

Theorem 9. *The co-operative value of all quantitative games can be computed in polynomial time.*

We will now study a sample mean-payoff game.

Example 2 (A mean-payoff game)**.** Consider the weighted arena from Figure 2 and let us focus on the mean-payoff function. If Eve controlled all the vertices, then she would be able to force a play of the form $\rho \cdot (v_0 v_1)^\omega$. Note that any such play has value 2—that is, regardless of the prefix ρ since mean payoff is prefix independent. Since there is no other play with a higher value in this game, 2 is its co-operative value. We will now argue that the antagonistic value of the game is 0. Recall that in a mean-payoff game Eve always has memoryless worst-case optimal strategies. Since at v_2 she does not really have a choice, then the only options are for her to stay in v_0 forever or move to v_1 every time the play reaches v_0. If she moves to v_1 then Adam might stay in v_1 forever. The outcome of these two strategies has value -4. If she stays in v_0 then the play v_0^ω has value 0 and Adam cannot change that. Hence the antagonistic value is indeed 0 and a worst-case optimal strategy for Eve in this game is to stay in v_0 always.

[18] Axel Haddad and Benjamin Monmege. Why Value Iteration Runs in Pseudo-Polynomial Time for Discounted-Payoff Games. Technical note, Université libre de Bruxelles, 2015. URL http://pageperso.lif.univ-mrs.fr/~benjamin.monmege/papers/HM15.pdf

[19] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Transactions on Computational Logic*, 11(4), 2010. DOI: 10.1145/1805950.1805953. URL http://doi.acm.org/10.1145/1805950.1805953

Games Played on Automata

In this section we consider an alternative choice of arena in which the games we study can be played. Namely, we describe how Eve and Adam can take turns to build infinite runs in an automaton, and the relation of such games to games played on graphs.

A game played on an automaton (Q, q_0, A, Δ) with *action set* A proceeds in rounds. First, from the current state $p \in Q$ of the game, Eve chooses an action $a \in A$. Then, Adam selects a state q from $\mathrm{post}_a(p)$. The new state of the game then becomes q. This process is repeated *ad infinitum* and its outcome is an infinite run. Formally, a game consists of an automaton and a payoff function (as is the case for games played on directed graphs).

Quantitative games played on automata require we extend our definition of automata.

Definition (Weighted automata)**.** A *weighted automaton* is a tuple $\mathcal{A} = (Q, q_0, A, \Delta, w)$ where (Q, q_0, A, Δ) is a usual automaton and $w : \Delta \to \mathbb{Q}$ is a *weight function* that assigns rational weights to transitions.

Notions of determinism, runs, etc. are inherited from the usual automata for infinite words. As in graph games, let $w_{\max}$ denote the maximum absolute value of a transition weight in the automaton.

A weighted automaton $\mathcal{A} = (Q, q_0, A, \Delta, w)$ and a payoff function **Val** are, together, called a *quantitative automaton*. A quantitative automaton realizes a function from words to real numbers (this generalizes the notion of language for Boolean automata). More formally, for a word $\alpha = a_0 a_1 \dots$, and a run $\rho = q_0 a_0 \dots$ of $\mathcal{A}$ on α, we denote by $\mathbf{Val}(\rho)$ the value of the run, *i.e.*

$$\mathbf{Val}(w(q_0, a_0, q_1) w(q_1, a_1, q_2) \dots).$$

We then write $\mathcal{A}(\alpha)$ for the supremum of the values of all its runs on the word.

In a quantitative game played on an automaton, a play is a run from the automaton and strategies are extended in the natural way from their graph game definitions. We remark that games played in graphs can be transformed into games played on automata.[20] The same is true in the opposite direction, from automata games to graph games.[21]

The usefulness of considering an alternative arena for quantitative games comes from the fact that high-level specifications such as linear-temporal logic usually give rise to automata-based games.

[20] This can be achieved, *e.g.*, by having states modelling vertices of Eve allow choice of successor depending on actions, and states corresponding to vertices of Adam transitioning to all direct successors with all actions.

[21] Here one can, for instance, split transitions into two edges with the same weight as the original transition and adjust the weights to account for the doubling of path lengths.

Solving Games Algorithmically

In this chapter we will refer sometimes refer to Eve and Adam as Players 0 and 1 respectively. This change in notation will allow for simpler notation in presenting an algorithm to solve games.

We begin by presenting a natural way of solving reachability games: by characterizing the sets of vertices from which Eve can force to visit some given set of vertices in at most k turns. This *attractor* will also be used later in the algorithm for parity games.

Game Attractors

Consider an arena $\mathcal{G} = (V, V_\exists, E)$ and a set of vertices $T \subseteq V$. (We have omitted v_I since it is not relevant at this point.) We define $\mathrm{Attr}_0^k(\mathcal{G}, T)$ for all $k \in \mathbb{N}$ inductively as follows. For $k = 0$ we set $\mathrm{Attr}_0^0(\mathcal{G}, T) := T$. For $k > 0$ we set

Some intuition on attractors. Essentially, the set $\mathrm{Attr}_0^k(\mathcal{G}, T)$ contains all vertices u such that: either Eve can choose a direct successor of u and reach the set $\mathrm{Attr}^{k-1}(\mathcal{G}, T)$, Adam has no other option but to choose a direct successor of u which ends up in the same set, or u is already in the set.

$$\begin{aligned}\mathrm{Attr}_0^k(\mathcal{G}, T) := \mathrm{Attr}_0^{k-1}(\mathcal{G}, T)&\\ &\cup\{u \in V_\exists \mid \exists(u,v) \in E : v \in \mathrm{Attr}_0^{k-1}(\mathcal{G}, T)\}\\ &\cup\{u \in V \setminus V_\exists \mid \forall(u,v) \in E : v \in \mathrm{Attr}_0^{k-1}(\mathcal{G}, T)\}.\end{aligned}$$

The attractor sets $\mathrm{Attr}_1^k(\mathcal{G}, T)$ for Adam are defined similarly. For $k = 0$ we set $\mathrm{Attr}_1^0(\mathcal{G}, T) := T$. Then for $k > 0$ we have

$$\begin{aligned}\mathrm{Attr}_1^k(\mathcal{G}, T) := \mathrm{Attr}_1^{k-1}(\mathcal{G}, T)&\\ &\cup\{u \in V \setminus V_\exists \mid \exists(u,v) \in E : v \in \mathrm{Attr}_1^{k-1}(\mathcal{G}, T)\}\\ &\cup\{u \in V_\exists \mid \forall(u,v) \in E : v \in \mathrm{Attr}_1^{k-1}(\mathcal{G}, T)\}.\end{aligned}$$

The main property guaranteed by attractor sets is formalized below.

Theorem 10. *For all $i \in \{0,1\}$, all $k \in \mathbb{N}$, and all $u \in V$ we have that if $u \in \mathrm{Attr}_i^k(\mathcal{G}, T)$ then Player i has a strategy to ensure that, against any strategy of the adversary, the resulting play visits T in at most k turns.*

We further observe that the sequence of attractor sets stabilizes after at most $|V|$ iterations.

Exercise 13. Give a proof by induction of Theorem 10 for Eve based on the definition of the attractor sets.

Theorem 11. *For all $i \in \{0,1\}$ and all $k \geq |V|$ it holds that* $\mathrm{Attr}_i^k(\mathcal{G}, T) = \mathrm{Attr}_i^{k+1}(\mathcal{G}, T)$.

Henceforth, we will denote by $\mathrm{Attr}_i(\mathcal{G}, T)$ this limit of the attractor sets $\mathrm{Attr}_i^0(\mathcal{G}, T), \ldots, \mathrm{Attr}_i^k(\mathcal{G}, T)$.

Exercise 14. Prove Theorem 11.

Note that, for all values of i, the set $\mathrm{Attr}_i^0(\mathcal{G}, S)$ can be computed in polynomial time from $S \subseteq V$ based on an exploration of the graph. It follows from Theorem 11 that we can actually compute all distinct attractor sets for both players in polynomial time.

Before we close this short section we establish two interesting results that already follow from the previous theorems.

Theorem 12. *For all $i \in \{0,1\}$ we have that Player i has a winning strategy for the reachability objective from $u \in V$ if and only if $u \in \mathrm{Attr}_i(\mathcal{G}, T)$.*

It immediately follows from the above results and from determinacy of reachability games that both are solvable in polynomial time.

Exercise 15. Prove Theorem 12 using the fact that positional strategies suffice for both players in reachability games.

Corollary 1. *The winner of all reachability and safety games can be determined in polynomial time.*

A Divide-and-Conquer Algorithm for Parity Games

Remember that we are assuming our arenas have no sinks. In Algorithm 1, we have to verify that removing all vertices R and edges including them does not break this property

Exercise 16. Using the fact that R is exactly the set of vertices from which a player wins a reachability game (and the other one loses a safety game), prove that $\mathcal{G} \setminus R$ contains no sinks. Prove that $\mathcal{G} \setminus S$ also contains no sinks.

Algorithm 1 Zielonka($\mathcal{G} = \langle V, V_\exists, E \rangle, p$)

```
1:  if $V = \varnothing$ then
2:      return $(\varnothing, \varnothing)$
3:  else
4:      $m \leftarrow \min\{p(v) \mid v \in V\}$
5:      $M \leftarrow \{v \in V \mid p(v) = m\}$
6:      $i \leftarrow m \pmod 2$
7:      $R \leftarrow \mathrm{Attr}_i(\mathcal{G}, M)$
8:      $(W'_i, W'_{i-1}) \leftarrow$ Zielonka$(\mathcal{G} \setminus R, p)$
9:      if $W'_{i-1} = \varnothing$ then
10:         $W_i \leftarrow W'_i \cup R$
11:         $W_{i-1} \leftarrow \varnothing$
12:     else
13:         $S \leftarrow \mathrm{Attr}_{i-1}(\mathcal{G}, W'_{i-1})$
14:         $(W''_i, W''_{i-1}) \leftarrow$ Zielonka$(\mathcal{G} \setminus S, p)$
15:         $W_i \leftarrow W''_i$
16:         $W_{i-1} \leftarrow W''_{i-1} \cup S$
17:     end if
18:     return $(W_i, W_{i-1})$
19: end if
```

The algorithm described above was discovered by Zielonka.[22] It computes, in a recursive fashion, the sets of vertices from which each player has a winning strategy for a given parity game.

[22] Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2): 135–183, 1998. DOI: 10.1016/S0304-3975(98)00009-7. URL https://doi.org/10.1016/S0304-3975(98)00009-7

Theorem 13. *Let (W_0, W_1) be the sets computed by Algorithm 1. For all $i \in \{0, 1\}$ and all $u \in V$ we have that $u \in W_i$ if and only if Player i has a winning strategy for the parity objective from u.*

Why does it work?

The main intuition behind why the algorithm works is the following. We start by choosing a player to support based on parity of the minimal priority. We also compute the set of vertices R from which said player, Player i in the algorithm, can ensure to visit a vertex with minimal priority at least once.

The recursive call from line 8 is made on a parity game in which we focus on all vertices from which Player $i - 1$ can "stay safe" of R. The idea is that Player i would win if he can stay within R. However, after a visit to a minimal-priority vertex, he may end up falling into a vertex outside of R. So we must check what happens there. If Player $i - 1$ cannot win from vertices outside of R then we have a winner for the whole arena: Player i. Otherwise, we have to take the opponent's point of view. We compute his attractor set S to the set of vertices from which he can win.[23] Clearly, from the whole of S Player $i - 1$ wins the game: he just plays to reach a vertex from which he has a winning strategy and follows it.[24] This is where a second recursive call is made to compute the winning vertices for both players in the game where Player i can stay safe of S. The winning vertices in this sub-game for Player i are the only ones in the global game from which he can win. The winning vertices for Player $i - 1$ are added to the ones we had previously seen were already winning for him, that is S.

[23] He wins **and** stays safe from R!

[24] Yes, the parity objective is prefix independent. This means that eventually winning is the same as winning.

Complexity

It should not be surprising that this algorithm **does not** run in polynomial time. An easy-to-prove upper bound on its running time is given below.

Proposition 5. *Algorithm 1 always terminates. Additionally, its running time is in $\mathcal{O}(2^n)$, where $n = |V|$ is the size of the given parity game.*

Termination should be obvious: every recursive call is made on a strictly smaller game.

Exercise 17. Prove the upper bound on the running time of the algorithm.

More exercises

We did not yet give an algorithm to solve Büchi and co-Büchi games. Your first task is to convince yourself that Zielonka's algorithm gives you such an algorithm.

Exercise 18. Given a (co-)Büchi game, construct a parity game with the same arena such that both players win from a vertex for the original objective if and only if they win from it for the parity objective. (Tip: you should not use more than three priorities)

Using the fact that we have at most 3 priorities, we can have a finer analysis of Zielonka's algorithm.

Exercise 19. Based on how many recursive calls Zielonka's algorithm makes on (co-)Büchi games, conclude that determining the winner of such games is decidable in polynomial time.

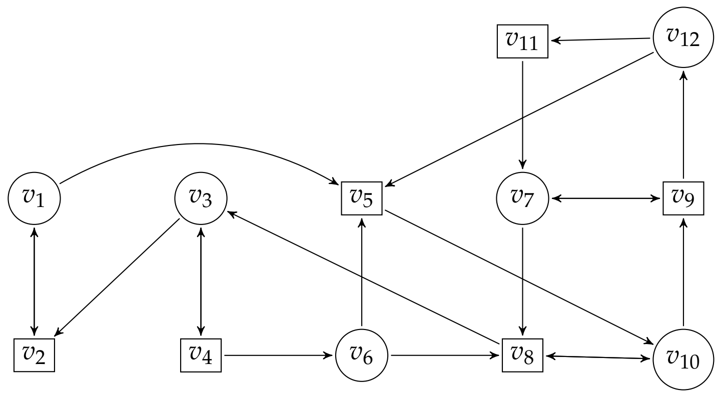

Figure 5: A reachability game with $T = \{v_1, v_2, v_10\}$

Exercise 20. Compute all distinct attractor sets for both players in the game from Figure 5. From which vertices can each player guarantee to win the game?

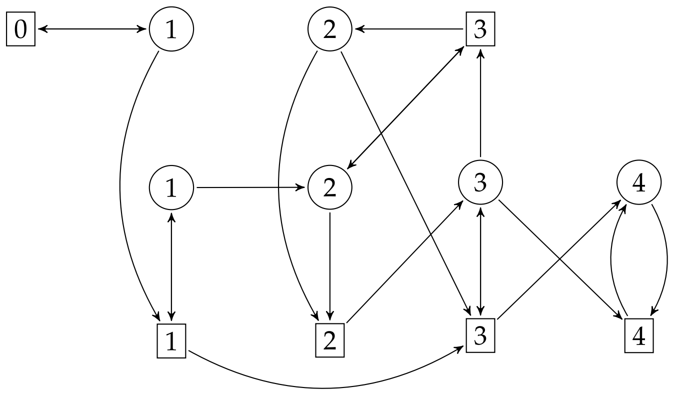

Figure 6: A parity game

Exercise 21. Run Zielonka's algorithm to determine the vertices from which each player can guarantee to win the parity game from Figure 6.

Bibliography

Krzysztof R. Apt and Erich Grädel. *Lectures in game theory for computer scientists*. Cambridge University Press, 2011.

Lubos Brim, Jakub Chaloupka, Laurent Doyen, Raffaella Gentilini, and Jean-François Raskin. Faster algorithms for mean-payoff games. *Formal Methods in System Design*, 38(2):97–118, 2011. DOI: 10.1007/s10703-010-0105-x. URL http://dx.doi.org/10.1007/s10703-010-0105-x.

Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Transactions on Computational Logic*, 11 (4), 2010. DOI: 10.1145/1805950.1805953. URL http://doi.acm.org/10.1145/1805950.1805953.

Andrzej Ehrenfeucht and Jan Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8:109–113, 1979. ISSN 0020-7276. DOI: 10.1007/BF01768705.

E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs. *SIAM J. Comput.*, 29(1):132–158, 1999.

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

Marcus Gelderie. *Strategy machines: representation and complexity of strategies in infinite games*. PhD thesis, RWTH Aachen University, 2014. URL http://darwin.bth.rwth-aachen.de/opus3/volltexte/2014/5025.

Axel Haddad and Benjamin Monmege. Why Value Iteration Runs in Pseudo-Polynomial Time for Discounted-Payoff Games. Technical note, Université libre de Bruxelles, 2015. URL http://pageperso.lif.univ-mrs.fr/~benjamin.monmege/papers/HM15.pdf.

Marcin Jurdziński. Deciding the winner in parity games is in **UP** ∩ **coUP**. *Information Processing Letters*, 68(3):119–124, 1998.

Donald A. Martin. Borel determinacy. *The annals of Mathematics*, 102 (2):363–371, 1975.

Donald A. Martin. The determinacy of blackwell games. *The Journal of Symbolic Logic*, 63(4):1565–1581, 1998.

Christos H. Papadimitriou. *Computational complexity*. John Wiley and Sons Ltd., 2003.

Dominique Perrin and Jean-Eric Pin. *Infinite words: automata, semigroups, logic and games*, volume 141. Academic Press, 2004.

Nir Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *LMCS*, 3(3), 2007. DOI: 10.2168/LMCS-3(3:5)2007. URL http://dx.doi.org/10.2168/LMCS-3(3:5)2007.

Shmuel Safra. On the complexity of ω-automata. In *FOCS*, pages 319–327. IEEE, 1988.

Shmuel Safra. Exponential determinization for omega-automata with strong-fairness acceptance condition (extended abstract). In *STOC*, pages 275–282, 1992. DOI: 10.1145/129712.129739.

Wolfgang Thomas. On the synthesis of strategies in infinite games. In *STACS*, pages 1–13. Springer, 1995.

Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200 (1-2):135–183, 1998. DOI: 10.1016/S0304-3975(98)00009-7. URL https://doi.org/10.1016/S0304-3975(98)00009-7.

Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *TCS*, 158(1):343–359, 1996.