1. Parity Objectives

1.1. Definition. Consider a graph \((S, E)\). Let \(d\) be a non-negative integer and let \(p : S \rightarrow \{0, 1, \ldots, d\}\) be a function which we call the priority function. The corresponding parity objective \(\text{Parity}(p)\) is defined to be the set of (infinite) paths in which the minimal priority of infinitely often visited nodes is even. In a more symbolic notation, this reads

\[
\text{Parity}(p) = \{\pi \in \Pi : \min_{s \in \text{Inf}(\pi)} p(s) \text{ is even}\}
\]

where \(\Pi\) denotes the set of all infinite paths in \((S, E)\) and \(\text{Inf}(\pi)\) is the set of nodes visited infinitely often in path \(\pi\). If \(G = ((S, E), (S_1, S_2))\) is a game graph and \(p : S \rightarrow \{0, 1, \ldots, d\}\) is a priority function, then we call the pair \((G, p)\) a parity game.

1.2. Parity Objectives Generalize Büchi and co-Büchi. It turns out that Büchi and co-Büchi objectives are in fact parity objectives: Given a Büchi set \(B \subseteq S\), the priority function \(p : S \rightarrow \{0, 1\}\)

\[
p(s) = \begin{cases} 
0 & \text{if } s \in B \\
1 & \text{if } s \not\in B
\end{cases}
\]

yields \(\text{Parity}(p) = \Diamond \square \neg B\). Likewise, priority function \(p' : S \rightarrow \{0, 1, 2\}\),

\[
p'(s) = \begin{cases} 
1 & \text{if } s \in B \\
2 & \text{if } s \not\in B
\end{cases}
\]

gives rise to \(\text{Parity}(p') = \Diamond \square \neg B\), which covers co-Büchi objectives.

The passage from \(p\) to \(p'\) is part of a more general principle: Given a priority function \(p\), we can calculate a \(p'\) such that \(\text{Parity}(p')\) is the complement of \(\text{Parity}(p)\); we achieve this by setting \(p'(s) = p(s) + 1\). The property that the complement of a parity objective is again a parity objective is called self-duality.

1.3. \(\omega\)-Regular Languages, Büchi Automata, and Parity Automata. Recall that a regular language is a set of finite words over the alphabet \(\Sigma\) that can be described by one of the following means which are mutually equivalent.

- a non-deterministic finite automaton (NFA)
- a deterministic finite automaton (DFA)

---

\(^1\)The proof of this statement is Lemma 2 in Appendix A.
\(^2\)cf. Lemma 3
a regular expression using the operations union \((r_1 \cup r_2)\), concatenation \((r_1r_2)\), and Kleene star \((r^*)\)

A **finite automaton** is a tuple \(A_F = (Q, \Sigma, \Delta, q_0, F)\) where \(Q\) is a set of states, \(\Sigma\) is an alphabet, \(\Delta \subseteq Q \times \Sigma \times Q\) is a transition relation, \(q_0 \in Q\) is a starting state, and \(F \subseteq Q\) is a set of accepting states. The language **accepted by** \(A\) is defined by

\[
L(A_F) = \{w \in \Sigma^* : \text{there exists a run of } A \text{ on word } w \text{ that ends in } F\}.
\]

We introduce two more types of automata—Büchi automata and parity automata, which can be used to describe \(\omega\)-regular languages. An \(\omega\)-regular language \(L\) over an alphabet \(\Sigma\) is a set of infinite words on \(\Sigma\), i.e., \(L \subseteq \Sigma^\omega\), that can be described in one of the following mutually equivalent ways (Theorem 1).

- a non-deterministic Büchi automaton (NBA)
- a non-deterministic parity automaton (NPA)
- a deterministic\(^3\) parity automaton (DPA)
- an \(\omega\)-regular expression using the operations union, concatenation, Kleene star, and omega iteration \((r^\omega)\)
- a linear-time logic (LTL) formula

A **Büchi automaton** is a tuple \(A_B = (Q, \Sigma, \Delta, q_0, F)\) like in the definition of finite automata. The difference to finite automata is the way in which its **accepted language** is defined, namely by

\[
L(A_B) = \{w \in \Sigma^\omega : \text{some run of } A \text{ on word } w \text{ visits } F \text{ infinitely often}\}.
\]

The expressive power of deterministic Büchi automata (DBA) is strictly weaker than that of NBA (Lemma 1).

A **parity automaton** is a tuple \(A_P = (Q, \Sigma, \Delta, q_0, p)\) where \(Q\), \(\Sigma\), \(\Delta\), \(q_0\) are like in the definition of finite automata, and \(p : Q \rightarrow \{0, 1, \ldots, d\}\) is a priority function. The **accepted language** of \(A\) is defined as

\[
L(A_P) = \left\{ w \in \Sigma^\omega : \text{there exists run } R \text{ of } A \text{ on } w \text{ s.t. } \min_{s \in \text{Inf}(R)} p(s) \text{ is even} \right\}.
\]

**Theorem 1.** The following statements are true:

1. For every NBA \(A_B\) there exists an NPA \(A_P\) such that \(L(A_B) = L(A_P)\).
2. For every NPA \(A_P\) there exists an \(\omega\)-regular expression \(r\) with the property that \(L(A_P) = L(r)\).
3. For every \(\omega\)-regular expression \(r\) there exists an LTL formula \(\varphi\) such that \(L(r) = L(\varphi)\).
4. For every LTL formula \(\varphi\) there exists an NBA \(A_B\) such that \(L(\varphi) = L(A_B)\).

We illustrate Theorem 1 with the help of two examples.

**Example 1.** Consider the alphabet \(\Sigma = \{a, b\}\) and the language \(L_1\) that includes all words in \(\Sigma^\omega\) that contain infinitely many \(a\)’s. It can be described by the \(\omega\)-regular expression \(r = (b^*a)^\omega\). Equivalently, \(L_1\) is expressible as the accepted language of the following NBA:

\[^3\text{The equivalence of NPA and DPA is non-trivial. Its proof can be found in Chapters 1 and 3 of Erich Grädel, Wolfgang Thomas, and Thomas Wilke (eds.), Automata Logics, and Infinite Games: A Guide to Current Research, Lecture Notes in Computer Science 2500, Springer, 2002.}\]
Example 2. Consider the alphabet of Example 1 and the language $L_2 = \Sigma^\omega \setminus L_1$ that is the complement of $L_1$. This language includes all words in $\Sigma^\omega$ that are eventually constantly equal to $b$. It can be described by the $\omega$-regular expression $r = (a \cup b)^* b^\omega$. Equivalently, $L_2$ is expressible as the accepted language of the following NBA:

Lemma 1. There exists an $\omega$-regular language that is not expressible by a DBA.

Proof. Consider language $L_2$ of Example 2. We claim that no DBA accepts $L_2$. Let $n$ be the number of $A$’s states.

Step 1. We can assume without loss of generality that all states $q \in Q$ are reachable from $q_0$, i.e., for every $q \in Q$ there exists a finite word $w_q \in \Sigma^*$ such that the application of $w_q$ to $A$ ends in state $q$. We could otherwise remove the non-reachable states and arrive at a DBA that accepts the same language as $A$ and possesses the above property.

Step 2. For every $q \in Q$, the run of $A$ on the infinite word $w_q b^\omega$ contains a cycle that is traversed infinitely often and that contains a state in $F$. Otherwise $A$ would not accept language $L_2$, which contains $w_q b^\omega$. Of course, this cycle is completely traversed after $n$ steps starting from $q$.

Step 3. For every $j \geq n$, the infinite word $w = (b^j a)^\omega \notin L_2$ is accepted by $A$. For every positive integer $m$, set $w_m = (b^j a)^m$. It is $w = \lim_{m \to \infty} w_m$. By Step 2, the run on $w_m b^j a$ contains at least one state in $F$. Hence the run on $w = (b^j a)$ contains some state in $F$ infinitely often, which causes $A$ to accept $w$; a contradiction to the fact that $A$ describes language $L_2$ because $w \notin L_2$. \hfill $\Box$

1.4. Single-Player Parity Games are PTIME. For single-player game graphs $G = (S, E)$ and parity objectives $\text{Parity}(p)$ on $G$, the task of calculating the winning set (i.e., the set of states starting from which a winning strategy for $\text{Parity}(p)$ exists) is solvable in polynomial time. To show this, we present a polynomial-time algorithm (Algorithm 1 on page 4).

Algorithm 1 is in PTIME, more specifically, its running time is in $O(d \cdot |E|)$, because computing maximal strongly connected components can be done in $O(|E|)$.

Algorithm 1 is correct: For a state $s$ to be in the winning set, it is necessary and sufficient that either (a) $s$ is contained in a cycle whose minimal priority is even or (b) $s$ can reach a state that satisfies (a). This equivalence is a consequence of memoryless determinacy (Theorem 2). Code lines 3–8 identify exactly those states that are contained in a cycle whose minimal priority is equal to $2k$. Hence lines 1–10 determine all nodes that satisfy (a). Line 11 adds those nodes satisfying (b).

---

4That is, all states are player 1 states.
2. The Memoryless Determinacy Theorem for Parity Objectives

**Theorem 2.** For all game graphs $G$ and all parity objectives $\text{Parity}(p)$ holds:

1. Determinacy: $\langle\langle 1 \rangle\rangle(\text{Parity}(p)) = \Pi \setminus \langle\langle 2 \rangle\rangle(\neg \text{Parity}(p))$
2. Existence of memoryless strategies:

\[
\langle\langle 1 \rangle\rangle(\text{Parity}(p)) = \{ s \in S : \exists \sigma_1 \in \Sigma_1^M \forall \sigma_2 \in \Sigma_2 : \pi(s, \sigma_1, \sigma_2) \in \text{Parity}(p) \} \\
\langle\langle 2 \rangle\rangle(\neg \text{Parity}(p)) = \{ s \in S : \exists \sigma_2 \in \Sigma_2^M \forall \sigma_1 \in \Sigma_1 : \pi(s, \sigma_1, \sigma_2) \in \neg \text{Parity}(p) \}
\]

**Proof sketch.** We prove both (1) and (2) by induction on $d$. The case $d = 0$ is trivial, for then $\langle\langle 1 \rangle\rangle(\text{Parity}(p)) = S$ with any strategy.

For the induction step, we assume without loss of generality that at least one state has priority $0$ or $1$. There are two cases: (a) some state has priority $0$ and (b) no state has priority $0$. We show how to handle case (a). For the following exposition, it might be helpful to refer to Figure 1 on page 5.

Let $W_2 = \langle\langle 2 \rangle\rangle(\neg \text{Parity}(p))$ (not necessarily with memoryless strategies) and $W_1 = S \setminus W_2$. Denote by $Z$ the set of states in $W_1$ with priority $0$ and define $A = \langle\langle 1 \rangle\rangle(\diamondsuit Z)$. It is $A \subseteq W_1$, because if player 1 can force to reach $Z$ (hence $W_1$) from state $s$, then player 2 does not have a winning strategy starting from $s$. Moreover, set $B = W_1 \setminus A$. The subgraph induced by $B$ is a game graph (i.e., every state has a successor in $B$) because otherwise, there exists a state $s \in B$ such that $E(s) \cap B = \emptyset$; but this is not possible:

- If $s \in B$ is a player 1 state, clearly, $E(s) \cap W_1 \neq \emptyset$. But also $E(s) \cap A = \emptyset$, which implies $E(s) \cap B = E(s) \cap A^c \cap W_1 = E(s) \cap W_1 \neq \emptyset$.
- If $s \in B$ is a player 2 state, then $E(s) \subseteq W_1$ and $E(s) \not\subseteq A$, which implies $E(s) \cap B = E(s) \cap W_1 \cap A_c = E(s) \cap A^c \neq \emptyset$.

By induction hypothesis (note that $B$ does not contain priority $0$), $B$ can be partitioned into $(B_1, B_2)$ such that player 1 has memoryless winning strategies in $B$’s subgame for states in $B_1$ and player 2 has memoryless winning strategies (in $B$’s subgame) for states in $B_2$. In fact, $B_2 = \emptyset$ since otherwise player 2 would have a winning strategy in the original game for states in $B_2$.

---

5Here, $A^c = S \setminus A$ denotes the complement of $A$. 

---

**Algorithm 1** PTIME algorithm for calculating the winning set of $\text{Parity}(p)$

1. $W \leftarrow \emptyset$
2. for $k = 0$ to $\lfloor d/2 \rfloor$ do
3. Compute maximal strongly connected components $C_j$ of $G$.
4. for all components $C_j$ do
5. if $C_j$ contains a state with priority $2k$ then
6. $W \leftarrow W \cup C_j$
7. end if
8. end for
9. Remove all states with priority $\leq 2k + 1$ from $G$.
10. end for
11. $W \leftarrow$ attractor set of $W$
We show that player 1 has a memoryless winning strategy on $W_1$. Denote by $\sigma_A$ player 1’s memoryless winning strategy$^6$ for reaching $Z$ starting from $A$. Let $\sigma_B$ player 1’s memoryless winning strategy for satisfying $\text{Parity}(p)$ starting from $B$. Finally, choose $\sigma^*_1 \in \Sigma^M_1$ equal to $\sigma_A$ on $A$ and equal to $\sigma_B$ on $B$. We claim that $\sigma^*_1$ is winning for player 1 on $W_1$.

Let $\sigma_2 \in \Sigma_2$ be any player 2 strategy and let $s \in W_1$. The claim is that $\pi = \pi(s, \sigma^*_1, \sigma_2) \in \text{Parity}(p)$. We distinguish two cases:

(A) $\pi$ visits $A$ infinitely often. Then, because $\sigma^*_1$ is equal to $\sigma_A$ on $A$, $\pi$ visits $Z$ infinitely often; hence $\min_{s \in \text{Inf}(\pi)} p(s) = 0$.

(B) $\pi$ visits $A$ only finitely often. Then, from some time on, player 2 chooses not to enter $A$ anymore. Hence after ignoring a finite prefix, play $\pi$ is in fact a play of the subgame induced by $B$, in which case $\sigma_B$ is a winning strategy for player 1.

\[ \square \]

3. Time Complexity

3.1. Problem Definition. We consider the decision problem for parity objectives:

**Input:** game graph $G = ((S, E), (S_1, S_2))$, priority function $p$, state $s \in S$

**Output:** “Yes” if and only if $s \in \langle 1 \rangle \langle \text{Parity}(p) \rangle$

3.2. Parity Decision is in $\text{NP} \cap \text{coNP}$. In this section, we show that the decision problem for parity objectives is in both $\text{NP}$ and $\text{coNP}$. It is not known whether it is solvable in polynomial time or not. Because of self-duality of parity objectives, it suffices to show the problem’s inclusion in $\text{NP}$. To show inclusion in $\text{NP}$, we need two things:

- a representation of a “Yes”-witness that takes polynomial memory space (i.e., it takes polynomial time to write it onto the string)

---

$^6$It is a general fact that there exists a single strategy $\sigma_1 \in \Sigma_1$ that wins against all player 2 strategies for plays starting from player 1’s winning set. When restricting the choice of $\sigma_1$ to memoryless strategies in $\Sigma^M_1$ and when considering reachability games, this also holds. We show that it holds when considering parity games.
a polynomial-time procedure to check the witness.

We choose the witness to be a memoryless winning strategy for player 1, i.e., a function \( \sigma_1 : S_1 \to E \).

The procedure to check \( \sigma_1 \) works as follows: First, modify the game graph \( G \) to graph \( G' \) by fixing the choices of \( \sigma_1 \); that is, for every \( s \in S_1 \), remove all edges in \( E(s) \) except for \( \sigma_1(s) \). After that, define all states in \( G' \) to belong to the same player and apply Algorithm 1 from Section 1.4. Strategy \( \sigma_1 \) is a correct witness if and only if \( s \) is not contained in the winning set output by Algorithm 1.

### 3.3. Classical \textsc{EXPTIME} Algorithm

We give an exponential-time algorithm (Algorithm 2) for calculating the winning set \( W_1 = \ll 1 \rr \langle \text{Parity}(p) \rangle \), hence for solving the decision problem, of arbitrary parity games. The intuition is to mimic the proof of memoryless determinacy. Refer to Figure 2 for a pictorial representation of a loop iteration in the algorithm.

**Algorithm 2** Classical Parity Algorithm

1: \( W'_2 \leftarrow \emptyset \)
2: repeat
3: \( W_2 \leftarrow W'_2 \)
4: \( Z \leftarrow \{ s \in S : p(s) = 0 \} \)
5: \( B \leftarrow S \setminus \ll 1 \rr \langle \Diamond Z \rangle \)
6: Recursively calculate \( W^B_2 \), player 2’s winning set in the subgame induced by the subset \( B \).
7: \( W'_2 \leftarrow W_2 \cup \ll 2 \rr \langle \Diamond W^B_2 \rangle \)
8: Remove \( W'_2 \) from the game graph \( (S, E) \).
9: until \( W_2 = W'_2 \)
10: \( W_1 \leftarrow \) the complement of \( W_2 \) in the original game graph.

Algorithm 2 is correct: Variable \( W_2 \) increases and is a subset of \( \ll 2 \rr \langle \neg \text{Parity}(p) \rangle \) at any time, because \( W^B_2 \subseteq \ll 2 \rr \langle \neg \text{Parity}(p) \rangle \) in any iteration. Also, variable \( W_1 \) satisfies \( W_1 \subseteq \ll 1 \rr \langle \text{Parity}(p) \rangle \) at the end of the algorithm, because player 1 wins in this subgame and removing the sets \( \ll 2 \rr \langle \Diamond W^B_2 \rangle \) from the game graph does not limit player 2’s choices.

![Figure 2. A single iteration in Algorithm 2](image)

Let’s analyze the running time of Algorithm 2. Denote by \( T(n, m, d) \) the worst-case time complexity of the algorithm when run on a game graph with \( n \) states,
m edges, and priorities 0 through d. It is $T(n, m, 0) = O(1)$ and $T(n, m, d) \leq n \cdot (O(m) + T(n, m, d - 1))$, because the loop is iterates at most $n$ times and each iteration takes $O(m) + T(n, m, d - 1)$ time. We deduce $T(n, m, d) = O(n^{d-1} \cdot m)$ by induction.

**Appendix A. Proofs Left Out in the Main Text**

**Lemma 2.** For $B \subseteq S$ and priority function $p$ defined in equation (2) it holds that $\text{Parity}(p) = \Box B$.

*Proof.* Let $\pi \in \text{Parity}(p)$. It follows from the definition that $\text{Inf}(\pi)$ contains at least one state $s$ with even $p(s)$, hence $p(s) = 0$. But this means that $s \in B \cap \text{Inf}(\pi)$; therefore this intersection is not empty and we have shown $\text{Parity}(p) \subseteq \Box B$.

Let now $\pi \in \Box B$. By definition, $B \cap \text{Inf}(\pi) \neq \emptyset$. Hence there exists a state $s$ that is contained in $\text{Inf}(\pi)$ and also contained in $B$, which means $p(s) = 0$. But then $\min_{s \in \text{Inf}(\pi)} p(s) = 0$ is an even number. $\Box$

**Lemma 3.** Let $G = (S, E)$ be a graph and $p : S \to \{0, 1, \ldots, d\}$ be a priority function. Then $p'$ defined as $p'(s) = p(s) + 1$ satisfies $\text{Parity}(p') = \Pi \setminus \text{Parity}(p)$.

*Proof.* The set $\Pi \setminus \text{Parity}(p)$ contains exactly those paths $\pi$ for which $\min_{s \in \text{Inf}(\pi)} p(s)$ is odd. But this is equivalent to $\min_{s \in \text{Inf}(\pi)} p(s) + 1$ being even. $\Box$

*E-mail address: tnowak@ecs.tuwien.ac.at*