Ranking Automata and Games for Prioritized Requirements

Rajeev Alur, Aditya Kanade, and Gera Weiss

University of Pennsylvania

Abstract. Requirements of reactive systems are usually specified by classifying system executions as desirable and undesirable. To specify *prioritized* requirements, we propose to associate a rank with each execution. This leads to optimization analogs of verification and synthesis problems in which we compute the "best" requirement that can be satisfied or enforced from a given state. The classical definitions of acceptance criteria for automata can be generalized to ranking conditions. In particular, given a mapping of states to colors, the *Büchi ranking* condition maps an execution to the highest color visited infinitely often by the execution, and the *cyclic ranking* condition with cycle k maps an execution to the modulo-k value of the highest color repeating infinitely often. The well-studied parity acceptance condition is a special case of cyclic ranking with cycle 2, and we show that the cyclic ranking condition can specify all ω-regular ranking functions. We show that the classical characterizations of acceptance conditions by fixpoints over sets generalize to characterizations of ranking conditions by fixpoints over an appropriately chosen lattice of coloring functions. This immediately leads to symbolic algorithms for solving verification and synthesis problems. Furthermore, the precise complexity of a decision problem for ranking conditions is no more than the corresponding acceptance version, and in particular, we show how to solve Büchi ranking games in quadratic time.

1 Introduction

A requirement φ of a reactive system M can be formally described as a set L_φ of finite or infinite words over system states (or observations) [14]. Verification of the system M with respect to the requirement φ corresponds to checking whether there exists an execution of M that does not belong to L_φ. When the system has only finitely many states, and the requirement can be captured as an ω-regular language, the verification problem can be solved algorithmically using decision procedures for ω-automata [18]. When the choices within the system M are partitioned into controllable and uncontrollable, then synthesis with respect to the requirement φ corresponds to checking whether there exists a strategy to resolve the controllable choices to ensure that the resulting execution belongs to L_φ. For the finite-state case, the synthesis question can be solved by algorithms for solving games with ω-regular winning conditions [15, 17].

In this paper, we propose a framework for specification, verification, and synthesis of *prioritized* requirements. Given a sequence $\varphi_0, \ldots \varphi_k$ of requirements,

listed in increasing order of importance, we want to find the *best* requirement that can be satisfied. For example, in Section 2, we describe a scheduling case study which employs several identical buffers to ensure uninterrupted flow of input messages, and using large buffers is expensive. We can specify prioritized requirements $\varphi_0, \ldots \varphi_k$, where φ_i states that the maximal buffer size required infinitely often during the execution is i or less. There are other potential scenarios where words can be naturally associated with ranks. For example, given multiple requirements, the priority of a word can be the number of requirements it satisfies. For software verification, we can associate different costs with different types of bugs, and have the analysis tool compute a classification of program statements grouped according to the worst bug that can manifest from them.

The optimization questions concerning prioritized requirements $\varphi_0, \ldots \varphi_k$ can, of course, be answered by solving k verification (or synthesis) questions separately, one for each requirement φ_i, using known techniques. However, as we establish in this paper, there is a better way of formulating and solving such questions. We formalize a prioritized requirement as a *ranking function* r that maps each word to a rank in an ordered set. We will focus only on *ω-regular ranking functions*, namely, functions that use only finitely many ranks such that the set of words with the same rank is an ω-regular language. For the case of two ranks, these notions coincide with the classical definitions of ω-regular sets.

Automata with different types of acceptance conditions (such as final-state, Büchi, Rabin, parity) are commonly used to specify ω-regular sets. To generalize acceptance to ranking, we consider deterministic automata in which each state is assigned a color. Then, given a run ρ of the automaton over a word w, the *reachability ranking* condition maps w to the highest color appearing in ρ. It is well-known that the set of states from which a target set can be reached can be computed as a least-fixpoint computation over sets of states starting with the target set. We show that this computation can be naturally generalized to fixpoints over functions that assign a color to each state (coloring functions). If the number of states is n and the number of colors (which captures the number of disjoint prioritized reachability requirements) is k, then, even though the number of iterations of the fixpoint computation is nk, we show that existing algorithm for solving reachability games (see [5]) can be adopted to solve the reachability ranking games in linear time.

Given a mapping of states to colors, the *Büchi ranking* condition maps a word to the highest color that repeats infinitely often in the corresponding run. The classical nested fixpoint characterization of Büchi acceptance [6] can now be generalized to an analogous fixpoint over coloring functions, and this allows us to compute the value of the corresponding verification/synthesis question at every state. We show that the number of iterations of the outer greatest-fixpoint loop is *independent* of the number of ranks. This gives us a quadratic-time algorithm for solving games with respect to Büchi ranking conditions.

Our final set of results concerns generalizing *parity* acceptance to *cyclic ranking* condition. Given an assignment of colors to states, the cyclic ranking condition with cycle k maps a word to the *modulo-k* value of the highest color

that repeats infinitely often in the corresponding run. The well-studied parity acceptance is a cyclic ranking condition with cycle 2. It is known that parity acceptance can specify all ω-regular sets. We prove an analogous result: every ω-regular ranking condition can be captured by a cyclic ranking condition. We also show that the winning strategy in games with cyclic ranking condition is memoryless, and the corresponding decision problem is in NP $\cap$ coNP.

Related Work: In scheduling literature, priorities are usually assigned with tasks, and are used to make local decisions [3]. Our definition allows associating priorities with global executions, and can potentially be used to capture high-level quality-of-service goals. In verification literature, various quantitative generalizations of verification and synthesis questions have been studied, typically involving real-time and/or probabilities, and are orthogonal to our notion of rankings. Parametric temporal logic allows capturing some versions of optimization problems, but the corresponding model checking problem has very different technical flavor [1]. Lattice automata [12, 13] generalize the notions of initialization, transitions, and acceptance from the Boolean case to a lattice, and can be used to associate ranks with words. However, it does not consider the problem of computing the optimum values of games, central to our motivation. While the decision procedures studied in [12, 13] can be used for optimization, we propose a faster algorithm. Similar to our result for reachability ranking games, [4] proposes a linear-time algorithm for weak parity games. Finally, the games literature considers models with costs associated with each state or transition. The most widely studied ranking function for runs is the *mean-payoff* cost [20], which is not ω-regular, and does not capture prioritization of requirements. The work in [9] considers real valued ranking functions for stochastic games.

2 A motivating example

Our example is based on a case study in which scheduling policies are determined for a signal processing board [8, 19]. We show that the problem of synthesizing an optimal scheduler is naturally modeled as a ranking question.

A block diagram of the signal processing board is depicted in Fig. 1. The board processes the two input streams shown at the top left and produces the two output streams shown at the bottom left. As depicted in the figure, data is stored in memory and is brought back when needed. The memory can only be accessed via a shared bus that transports data in quantities of 128 bits. Nine identical buffers are designed to allow uninterrupted flow of data when the bus is not available. The main objective of the case study is to analyze and to design an arbiter that schedules the use of the shared bus. A valid schedule must guarantee that, after a finite initialization phase, none of the data streams are interrupted. The optimal schedule is the one which minimizes the buffer size (the number of bits per buffer).

A schedule for bus arbitration is represented by a word over the alphabet $\Sigma = \{0, 1, \ldots, 9\}$. If the ith letter of the word is B, the bus is used to transfer

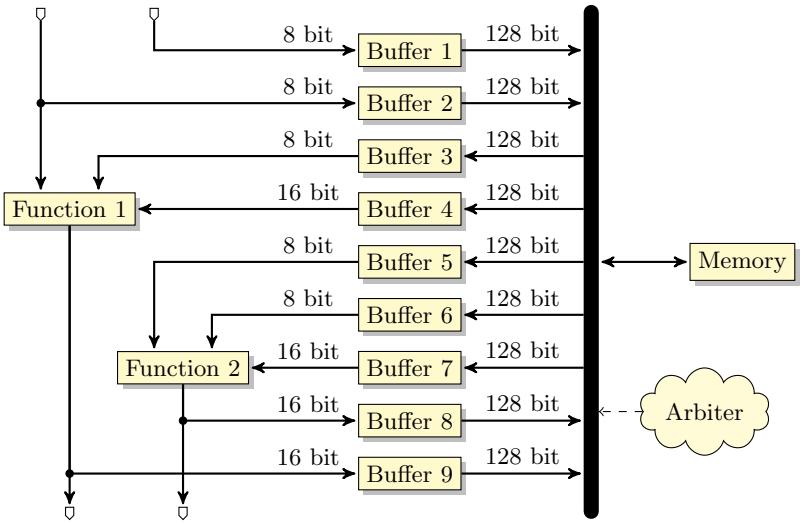


Fig. 1. A signal processing board.

data to or from the Bth buffer at the ith clock cycle. Zero at the ith letter of the word means that the bus is idle at the ith clock cycle. Our goal is to form an automaton that accepts a word if and only if it represents a valid schedule (no overflows or underflows). The automaton will also be used to assign ranks (payoffs) to schedules and analyzed to determine the optimal schedule.

The automaton for the language of valid schedules is constructed as a composition of nine automata, one for each buffer. The automaton for a buffer B is as follows. Let M be an upper bound on the size of the buffer and $r \in \{-16, -8, 8, 16\}$ denote the inflow to the buffer (negative if B buffers a stream from memory to a functional block). These parameters can be read from Fig. 1. The states of the automaton are the numbers $Q(B) = \{0, |r|, 2|r|, \ldots, m\}$ where $m \leq M$ and $m + |r| > M$ for the quantum of flow r associated with buffer B.

We define two types of transitions. First, if the buffer is not scheduled to use the bus, it gets or loses a quantum of flow. This is modeled by $\delta(q, b) = q + r$ if $q + r \in Q(B)$ and $b \in \Sigma \setminus \{B\}$. Second, if the buffer is scheduled to use the bus, it loses or gets 128 bits. This is modeled by $\delta(q, B) = q - 128\,\mathrm{sgn}(r) + r$ if $q - 128\,\mathrm{sgn}(r) + r \in Q(B)$ (where $\mathrm{sgn}(r) := r/|r|$). All other transitions go to an implicit sink which is the only non-accepting state. The initial state is $q_0 = 0$.

Now, we use the product construction to get an automaton for the intersection of the languages of the nine automata. Clearly, a word is accepted by the resulting automaton if and only if it is a valid schedule. Note that the intersection may be empty, in which case we would take a larger M and recompute.

To analyze the schedules, we assign with a state $q = (q_1, \ldots, q_9) \in Q(1) \times \cdots \times Q(9)$ a number $c(q) = \max\{q_1, \ldots, q_9\}$. We call this number the color of the state and c the coloring function. Let the *rank* of an infinite word $w \in \Sigma^\omega$ be

the maximal color visited infinitely often when the automaton reads the word. The rank of a word is thus the maximal buffer size needed for valid execution of the bus arbitration schedule identified by the word. The optimal buffer size thus identified could be smaller than the buffer size required during the initialization phase of the board (a finite prefix of a valid schedule). The potential loss of data (overflow) during the initialization phase is acceptable in the case study.

To get an optimal schedule, we need to search for a word with the minimal rank. A naïve approach is to solve this (quantitative) problem by solving a series of (qualitative) decision problems. In particular, let L_i be the set of valid schedules where no state with color higher than i is visited infinitely often. Then, one can check non-emptiness of L_i to identify whether the buffer size i is sufficient; until the minimal bound is found. In this paper we first study this iterative approach and then present a direct and more efficient algorithm.

We also analyze the more general setting of games. In games, nondeterminism is split into controllable and uncontrollable choices. The synthesis of an optimal schedule for the signal processing board, in its full generality, requires game based analysis. The memory needs to be refreshed periodically and this is not in the control of the arbiter. The chip decides the refresh timings and, during a memory refresh, the memory is not available. Specifically, the time between consecutive refreshes varies nondeterministically between 100 and 200 clocks and each refresh takes 10 clocks. Using the algorithms presented in this paper, we can find an optimal schedule even in the presence of memory refreshes, by considering the refresh mechanism as adversarial and finding the best strategy against its worst-case behavior. Note that games are essential here since the refresh mechanism is nondeterministic and the worst possible refresh (from the perspective of the arbiter) cannot be identified independently.

3 Ranking functions and games

We introduce a generalization of ω-regular languages called ω-regular ranking functions. Intuitively, languages specify qualitative properties of words by giving the set of accepted words whereas rankings specify quantitative properties by assigning numbers to words.

Definition 1. An ω-regular ranking function over a finite alphabet Σ is a function $r\colon \Sigma^\omega \to \mathbb{N}$ with a finite range and $\{w\colon r(w) = n\}$ is an ω-regular language for every n in the range.

Note that the characteristic function of an ω-regular language is an ω-regular ranking function with range $\{0, 1\}$.

A *(deterministic) finite state automaton* is a tuple (Q, Σ, δ, q_0) where Q is a set of states, Σ is an alphabet, $\delta\colon Q \times \Sigma \to Q$ is a transition function, and $q_0 \in Q$ is an initial state. Define $\delta^*\colon \Sigma^* \to Q$, inductively, by $\delta^*(\sigma_1 \cdots \sigma_l) = \delta(\delta^*(\sigma_1 \cdots \sigma_{l-1}), \sigma_l)$ and $\delta^*(\epsilon) = q_0$ (where ϵ is the empty word). For $w \in \Sigma^\omega$, define $\mathrm{Inf}_{\mathcal{A}}(w) := \{q \in Q\colon \delta^*(w') = q$ for infinitely many prefixes w' of $w\}$.

We introduce cyclic ranking conditions. A cyclic ranking condition with cycle k maps a word to the modulo-k value of the highest color repeating infinitely often. A parity acceptance condition is a cyclic ranking condition with $k = 2$.

Definition 2. A cyclic ranking condition for a finite automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0)$ is a pair (c, k) where $c\colon Q \to \mathbb{N}$ is a coloring function and $k \in \mathbb{N}$ is the number of ranks. The corresponding ranking function is given by $\mathrm{Cyclic}_{\mathcal{A},c,k}(w) := \max\{c(q)\colon q \in \mathrm{Inf}_{\mathcal{A}}(w)\} \operatorname{modulo} k$.

Analogous to the known result that parity acceptance conditions can capture all ω-regular languages, we show that cyclic ranking conditions can capture all ω-regular ranking functions.

Proposition 3 (Expressive completeness of cyclic ranking conditions)**.** *A ranking function $r\colon \Sigma^\omega \to \mathbb{N}$ is an ω-regular ranking function if and only if there is a deterministic finite automaton $\mathcal{A}$ and a cyclic ranking condition (c, k) such that $\mathrm{Cyclic}_{\mathcal{A},c,k}(w) = r(w)$, for all $w \in \Sigma^\omega$.*

Proof. The if direction is straightforward. For the only-if direction, let the ω-regular language $\{w\colon r(w) = s\}$ be specified by a deterministic Muller automaton $\mathcal{A}_s = (Q_s, \Sigma, \delta_s, q_{0_s}, \mathcal{F}_s)$ where $s \in S$ and S is the range of r. For simplicity, let $S = \{0, \cdots, k-1\}$. Let $\mathcal{A}_M = (Q_M, \Sigma, \delta_M, q_{0_M})$ be the product of the automata $\mathcal{A}_s$, $s \in S$. For $F \subseteq Q_M$, let $\mathrm{proj}_s(F) = \{q_s \in Q_s\colon \exists q = \langle \ldots, q_s, \ldots\rangle \in F\}$. Consider a Muller ranking function $\mathrm{Mul}_{\mathcal{A}_M}\colon 2^{Q_M} \to \mathbb{N}$ defined as $\mathrm{Mul}_{\mathcal{A}_M}(F) := \max(\{s\colon \mathrm{proj}_s(F) \in \mathcal{F}_s\} \cup \{0\})$.

Consider a ranking function $r_M\colon \Sigma^\omega \to \mathbb{N}$ where $r_M(w) = \mathrm{Mul}_{\mathcal{A}_M}(\mathrm{Inf}_{\mathcal{A}_M}(w))$. Suppose for an infinite word $w \in \Sigma^\omega$, $r(w) = i$. Clearly, $\mathrm{Inf}_{\mathcal{A}_i}(w) \in \mathcal{F}_i$ and for $j \neq i$, $\mathrm{Inf}_{\mathcal{A}_j}(w) \notin \mathcal{F}_j$. Thus $r_M(w) = \mathrm{Mul}_{\mathcal{A}_M}(\mathrm{Inf}_{\mathcal{A}_M}(w)) = i$.

Using the latest appearance record (LAR) [10, 11] we construct an automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0)$ which simulates $\mathcal{A}_M$ as follows:

$Q = Q_M! \times |Q_M|$ where $Q_M!$ is the set of all permutations of Q_M

$q_0 = ((p_1, \cdots, p_n), 1)$ for some permutation $(p_1, \cdots, p_n) \in Q_M!$ for $p_1 = q_{0_M}$

$\delta(((p_1, \cdots, p_n), h), \sigma) = ((\delta_M(p_1, \sigma), p_1, \cdots, p_{h'-1}, p_{h'+1}, \cdots, p_n), h')$
with h' as the index for $(p_1, \cdots, p_n)$ called *hit* position s.t. $\delta_M(p_1, \sigma) = p_{h'}$, if $\delta_M(p_1, \sigma)$ is defined.

Consider a cyclic ranking condition (c, k) where $k = |S|$ and a coloring function $c\colon Q \to \mathbb{N}$ defined as follows:

$$c(((p_1, \cdots, p_n), h)) := \begin{cases} kh & \text{if } \{p_1, \cdots, p_h\} \in \mathrm{Mul}^{-1}_{\mathcal{A}_M}(0) \\ kh+1 & \text{if } \{p_1, \cdots, p_h\} \in \mathrm{Mul}^{-1}_{\mathcal{A}_M}(1) \\ \vdots & \\ kh+k-1 & \text{if } \{p_1, \cdots, p_h\} \in \mathrm{Mul}^{-1}_{\mathcal{A}_M}(k-1) \end{cases}$$

Let h_{max} be the maximal hit position occurring infinitely often on the run of $\mathcal{A}$ on w. Eventually for any state $((p_1, \cdots, p_n), h)$, $h \leq h_{max}$ and $\{p_1, \cdots, p_{h_{max}}\} =$

$\mathrm{Inf}_{\mathcal{A}_M}(w)$. If $r_M(w) = i$ then $\{p_1, \cdots, p_{h_{max}}\} \in \mathrm{Mul}^{-1}_{\mathcal{A}_M}(i)$. Thus the maximal color occurring infinitely often on the run of $\mathcal{A}$ on w is $kh_{max} + i$. Hence $\mathrm{Cyclic}_{\mathcal{A},c,k}(w) = i = r_M(w) = r(w)$. □

Similarly, we generalize reachability, Büchi, and coBüchi acceptance conditions to reachability, Büchi, and coBüchi ranking conditions as follows.

Definition 4. Reachability, Büchi, and coBüchi ranking conditions for a finite automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0)$ are expressed by coloring functions $c\colon Q \to \mathbb{N}$. The corresponding ranking functions are defined as follows:

$$\begin{aligned}
\mathrm{Reach}_{\mathcal{A},c}(w) &:= \max\{c(\delta^*(w'))\colon\ w' \text{ is a prefix of } w\} \\
\mathrm{Buchi}_{\mathcal{A},c}(w) &:= \max\{c(q)\colon q \in \mathrm{Inf}_{\mathcal{A}}(w)\} \\
\mathrm{coBuchi}_{\mathcal{A},c}(w) &:= \min\{c(q)\colon q \in \mathrm{Inf}_{\mathcal{A}}(w)\}
\end{aligned}$$

Next, we extend ω-win-lose games to ω-ranking games. A *game automaton* is a triplet $(\mathcal{A}, Q_0, Q_1)$ where $\mathcal{A} = (Q, \Sigma, \delta, q_0)$ is a finite state automaton and (Q_0, Q_1) is a partition of Q into Player 0 and Player 1 states, respectively. In figures, following the usual drawing convention, we use ◯ to denote states in Q_0 and □ to denote states in Q_1.

Definition 5. An ω-regular ranking game is a pair $\mathcal{G} = (\mathcal{A}, r)$ where $\mathcal{A}$ is a game automaton and $r\colon \Sigma^\omega \to \mathbb{N}$ is an ω-regular ranking function (rewards for Player 0, penalties for Player 1).

A strategy for a player $p \in \{0, 1\}$ in an ω-regular ranking game is a function $s_p\colon \{w \in \Sigma^*\colon \delta^*(w) \in Q_p\} \to \Sigma$. The letter $s_p(w)$ models the move of player p after observing w. Let $\mathcal{S}_p(\mathcal{G})$ be the set of all strategies for player $p \in \{0, 1\}$.

A pair of strategies $(s_0, s_1) \in \mathcal{S}_0(\mathcal{G}) \times \mathcal{S}_1(\mathcal{G})$ induces an infinite word w whose $(i+1)$th letter is given by $w_{i+1} := s_0(w_{1..i})$ if $\delta^*(w_{1..i}) \in Q_0$ and $s_1(w_{1..i})$ if $\delta^*(w_{1..i}) \in Q_1$, where $w_{1..i}$ is the prefix of length i of w and $w_{1..0} = \epsilon$. We denote this word by $w_{\mathcal{G}}(s_0, s_1)$. The outcome of a pair of strategies is defined to be the rank of this word and the value of the game is defined accordingly, as follows.

Definition 6. The value of a strategy $s_0 \in \mathcal{S}_0(\mathcal{G})$ in a game $\mathcal{G} = (\mathcal{A}, r)$ is defined by $\mathrm{val}_{\mathcal{G}}(s_0) = \min\{r(w_{\mathcal{G}}(s_0, s_1))\colon s_1 \in \mathcal{S}_1(\mathcal{G})\}$. The value of the game is defined by $\mathrm{val}(\mathcal{G}) = \max\{\mathrm{val}_{\mathcal{G}}(s_0)\colon s_0 \in \mathcal{S}_0(\mathcal{G})\}$.

If we think of Player 0 as the system and Player 1 as the environment, the above definition captures the objective of Player 0 which is to maximize the outcome against the worst-case behavior of Player 1.

4 Algorithms for ranking games

We analyze algorithms for solving ω-regular ranking games. In our context, solving means for each state of the automaton, determining the value of the game starting at it and synthesizing a strategy that achieves these values.

4.1 Solving ranking games as a series of win-lose games

In this section we propose a simple scheme for solving ranking games as a series of appropriately defined win-lose games. In lattice games (cf. [2, 13]), a similar decomposition to join-irreducible elements is used but it only gives a sufficiency condition. We show that for Büchi games such decomposition is also necessary. We focus on Büchi games for simplicity. Generalization to other games is straightforward.

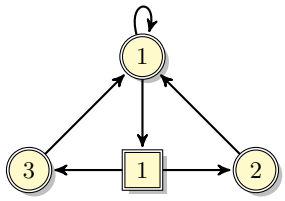
For a game automaton $\mathcal{A}$ and a set of states B, let Buchi$(\mathcal{A}, B)$ be an algorithm for win-lose Büchi games [11] which computes the states from which Player 0 can force visiting B infinitely often (the winning region for Player 0).

Consider a game automaton whose set of states is Q, and a coloring function $c\colon Q \to \mathbb{N}$ that maps the states to colors. Algorithm 1 computes the function $r\colon Q \to \mathbb{N}$ where $r(q)$ is the maximal color such that Player 0 can force infinitely many visits to states with color $r(q)$, starting at q. In the first iteration, the algorithm computes all the states from which Player 0 can force infinitely many visits to the highest color. Then, it removes these nodes from the graph and proceeds to the second highest color and the process is repeated.

Algorithm 1: IteratedMaxBuchi$(\mathcal{A}, c)$

```
foreach q ∈ Q do mark r(q) as undefined
foreach γ ∈ {c(q): q ∈ Q} in decreasing order do
    Q_γ := {q ∈ Q: r(q) is undefined}
    A_γ := the automaton A restricted to the states in Q_γ
    B_γ := {q ∈ Q_γ : c(q) ≥ γ}
    foreach q ∈ Buchi(A_γ, B_γ) do r(q) := γ
return r
```

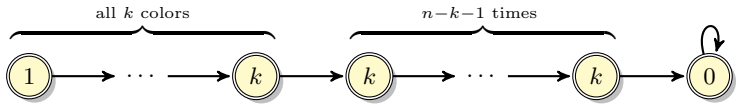
Remark 7. For a color γ, it is necessary to also include the states (if any) with color $> \gamma$ from the set Q_γ in the Büchi set B_γ. For example, consider the following game automaton:



Player 0 cannot force infinite visits to color 3. Player 0 also cannot force infinite visits to color 2. However, the objective of Player 1 is to minimize the value and hence Player 1 will not choose color 3 over color 2. Hence the value that Player 0 can achieve starting from any state is 2. In terms of Büchi win-lose games, the

winning set corresponding to color 2 is identified correctly only if the Büchi set, B_2, contains states with colors 2 as well as 3.

Complexity analysis. For $k, n \in \mathbb{N}$, consider the game automaton:



As the value for each state is 0, Algorithm 1 takes k iterations. Assuming the standard implementation of Buchi, it can be verified that each iteration runs in time $O(|B_\gamma| \cdot |\delta|)$ (Buchi$(\mathcal{A}_\gamma, B_\gamma)$ starts with B_γ and, in its ith iteration, removes the states from which Player 0 cannot force at least i visits to B_γ; each iteration takes $O(|\delta|)$). Further, from Remark 7, $|B_\gamma| = \sum_{i=\gamma}^{k} |c^{-1}(i)|$ and $\sum_{\gamma=k}^{1} |B_\gamma|$ is $O(k \cdot |Q|)$. Hence the worst-case execution time of Algorithm 1 is $O(k \cdot |Q| \cdot |\delta|)$ which is k times the worst-case execution time for solving a Büchi win-lose game.

Remark 8. Denote the leftmost state in the above automaton by q_0. If we only need to compute the value of q_0, we can use binary search as follows. Check if $q_0 \in$ Buchi$(\mathcal{A}_{k/2}, B_{k/2})$ then if $q_0 \in$ Buchi$(\mathcal{A}_{k/4}, B_{k/4})$ and so on until we find that $q_0 \in$ Buchi$(\mathcal{A}_0, B_0)$. As only the last query is answered positively, the algorithm takes $\log k$ steps. The total execution time is $O(\log k \cdot |Q| \cdot |\delta|)$.

Remark 7 shows that in case of Büchi games, the decomposition to join-irreducible elements (with respect to the order $1 \leq \ldots \leq k$) is necessary but the complexity of the resulting algorithm is not optimal as shown by the example above. In Section 4.4, we generalize the algorithm for solving Büchi win-lose games to solve Büchi ranking games without any increase in complexity. The advantage of the above algorithm, however, is that existing algorithms for Büchi games can be directly used for solving Büchi ranking games.

4.2 Fixpoints over coloring functions

The solutions of win-lose games are typically defined as fixpoints of functions over the lattice of the power-set of the set of states of the game automaton, ordered by inclusion. For ranking games, we use coloring functions as a generalization of sets of states and define the following lattice.

Definition 9. For a set Q, $\mathbb{N}^Q$ is the set of all functions that map elements of Q to natural numbers (coloring functions). Consider the lattice $(\mathbb{N}^Q, \sqsubseteq)$ where $r_1 \sqsubseteq r_2$ if $r_1(q) \leq r_2(q)$ for all $q \in Q$. The join and meet operations of the lattice are $(r_1 \sqcup r_2)(q) = \max\{r_1(q), r_2(q)\}$ and $(r_1 \sqcap r_2)(q) = \min\{r_1(q), r_2(q)\}$, respectively.

The lattice of coloring functions is infinite. However, the range of a coloring function used for defining a ranking condition is a finite subset of the set of natural numbers. We therefore identify a finite sub-lattice defined below.

Definition 10. For a coloring function $c\colon Q \to \mathbb{N}$, let $\mathrm{Lat}(c)$ be the lattice $(\mathcal{R}(c)^Q, \sqsubseteq)$ where $\mathcal{R}(c) = \{c(q)\colon q \in Q\}$. It is easy to verify that $\mathrm{Lat}(c)$ is closed under join and meet, so it is a complete finite sub-lattice of $(\mathbb{N}^Q, \sqsubseteq)$. The bottom of the lattice is $\bot = Q \times \{\min \mathcal{R}(c)\}$ and the top is $\top = Q \times \{\max \mathcal{R}(c)\}$.

In the following subsections, we give fixpoint characterizations for ranking games. It can be easily verified that the functions whose extremal fixpoints determine the solutions of the games are monotonic and closed over the lattice defined above. By finiteness of the lattice and by the Knaster–Tarski fixpoint theorem, we know that the extremal fixpoints of the functions can be computed.

4.3 Solving reachability ranking games in linear time

For a game automaton $\mathcal{A}$ and a coloring function c, let the solution of the reachability ranking game be given by the function $r^{\mathcal{A},c}_{\mathrm{Reach}}$ that maps each state q to the maximal color i such that Player 0 can force a visit to a state in $\{q' \in Q\colon c(q') \geq i\}$, starting from q. Let $\mathrm{succ}(q) := \{\delta(q,\sigma)\colon \sigma \in \Sigma\}$ and $\mathrm{pred}(q) := \{q'\colon q \in \mathrm{succ}(q')\}$.

The fixpoint formulation of $r^{\mathcal{A},c}_{\mathrm{Reach}}$ is given in Proposition 11 as the least fixpoint (LFP) of the function f which assigns to a state q the highest color that Player 0 can force in one step (or less) from q. Let $\mathrm{apred}(r)(q) := \max\{r(q')\colon q' \in \mathrm{succ}(q)\}$ for $q \in Q_0$ and $\mathrm{apred}(r)(q) := \min\{r(q')\colon q' \in \mathrm{succ}(q)\}$ for $q \in Q_1$.

Proposition 11. $r^{\mathcal{A},c}_{\mathrm{Reach}} = \mathrm{LFP}(f)$ *where* $f\colon \mathrm{Lat}(c) \to \mathrm{Lat}(c)$ *is given by* $f(r) := r \sqcup \mathrm{apred}(r) \sqcup c$.

The function apred can be computed in time $O(|\delta|)$, where δ is the transition function of the game automaton. Hence, the function f can also be computed in time $O(|\delta|)$. Further, f is defined such that if, starting at a state q, Player 0 can force a visit to a color $\geq \gamma$ (with respect to a coloring function r) within i steps, it can do so within $\max\{0, i-1\}$ steps with respect to the coloring function $f(r)$. Since the length of an acyclic path in $\mathcal{A}$ can be at most $|Q|-1$, the effective height of the lattice $\mathrm{Lat}(c)$ for f is $|Q|-1$. The overall complexity of computing the fixpoint appears to be quadratic. However, we now specify a suitable traversal of the automaton and also propose to keep a record of the transitions already processed. This ensures that any transition of the automaton is processed only once and we get a linear-time algorithm.

With each state $q \in Q_1$ associate a number $\mathrm{count}(q) = |\mathrm{succ}(q)|$. Also let all states be marked as not visited. Consider the following evaluation order: Starting with the highest color, for each color γ, perform a preorder backwards traversal starting with the states with color γ that are not marked as visited. Let q be the state being processed. If $q \in Q_0$ and not marked as visited then its color is set to γ and is marked as visited. Let $q \in Q_1$. If $\mathrm{count}(q) = 1$ then all other outgoing transitions of q have been explored during processing of colors $> \gamma$. Hence the minimal color that Player 1 can force is γ. Set the color of q to γ and mark it as visited. Otherwise, set $\mathrm{count}(q)$ to $\mathrm{count}(q) - 1$.

Theorem 12. *Reachability ranking games can be solved in $O(|\delta|)$ time, where δ is the transition function of the game automaton.*

Remark 13. The memoryless optimal winning strategy for Player 0 can be identified during the computation of values described above. If $q \in Q_0$ has color γ and is not marked as visited until the processing of color γ then the value at q is γ, that is, its own color. Thus the strategy for Player 0 at q is to select the label of any outgoing transition. Otherwise, the strategy is to select the label of the outgoing transition of q that lead to q being marked as visited (note that the game automaton is deterministic).

4.4 An efficient quadratic-time algorithm for Büchi ranking games

For a game automaton $\mathcal{A}$ and a coloring function c, the solution to the Büchi ranking problem is a function $r_{\text{Buchi}}^{\mathcal{A},c}$ that maps each state q to the maximal color i such that Player 0 can force infinitely many visits to $\{q' \in Q \colon c(q') \geq i\}$, starting from q. We present a fixpoint formulation of the solution function and show that it can be computed in quadratic-time. Its complexity is independent of the number of colors, as opposed to Algorithm 1.

The fixpoint formulation of $r_{\text{Buchi}}^{\mathcal{A},c}$ is given in Proposition 14. The function g identifies for each state q the maximal color that Player 0 can force to visit at least once, starting at q. The greatest fixpoint (GFP) of g computes for each state q the maximal color (less than or equal to $c(q)$) that Player 0 can force to visit infinitely many times. Finally, the solution of the reachability ranking for the coloring function given by $\text{GFP}(g)$ determines the solution $r_{\text{Buchi}}^{\mathcal{A},c}$.

Proposition 14. $r_{\text{Buchi}}^{\mathcal{A},c} = r_{\text{Reach}}^{\mathcal{A},\text{GFP}(g)}$ *where* $g \colon \text{Lat}(c) \to \text{Lat}(c)$ *is defined by* $g(r) = r \sqcap c \sqcap \text{LFP}(f_r)$ *and* $f_r \colon \text{Lat}(r) \to \text{Lat}(r)$ *is defined by* $f_r(r') = r' \sqcup \text{apred}(r' \sqcup r)$.

Note that the function f_r is defined analogously to the function f given in Proposition 11 but considers states reachable in one or more steps instead of zero or more steps. From Theorem 12, we can deduce that $\text{LFP}(f_r)$ can be computed in time $O(|\delta|)$. Consequently, the function g can be computed in time $O(|\delta|)$.

Complexity analysis. We now show that in each iteration of the fixpoint computation $\text{GFP}(g)$, at least one additional state gets its final color. Thus, the fixpoint computation $\text{GFP}(g)$ takes no more than $|Q|$ steps. This gives quadratic-time complexity for solving Büchi ranking games.

Let $r_0 = c$ and $r_0 \sqsupseteq r_1 \sqsupseteq \cdots \sqsupseteq r_l$ be the sequence of functions computed in the fixpoint computation $\text{GFP}(g)$. Let $W_j(r_i) := \{q \in Q \colon r_i(q) = j > r_l(q)\}$ be the set of states whose color in the ith iteration (given by coloring function r_i) is j which is larger than their final color given by coloring function r_l.

In Lemma 15, we show that if $W_j(r_i) \neq \emptyset$ then, in the next iteration, at least one state from $\bigcup_{j'=j}^{k} W_{j'}(r_i)$ gets a color smaller than j or is assigned to its final color; and this is true for each color j.

Lemma 15. *If* $W_j(r_i) \neq \emptyset$ *then* $\bigcup_{j'=j}^{k} W_{j'}(r_i) \setminus \bigcup_{j'=j}^{k} W_{j'}(r_{i+1}) \neq \emptyset$.

Proof. Since $W_j(r_i) \neq \emptyset$ there exists a state q such that $r_i(q) = j > r_l(q)$. Because $r_l(q)$ is the final color of q and $r_0 = c$, Player 0 cannot force infinitely many visits to $\{q' \colon r_0(q') > r_l(q)\}$ starting from q. Since $r_0 \sqsupseteq r_i$ and $r_i(q) > r_l(q)$, we have $r_0(q) \geq r_i(q) > r_l(q)$. Thus, Player 0 cannot force infinitely many visits to $S := \{q' \colon r_i(q') \geq r_i(q)\}$, starting at q. Because $q \in S$, there exists at least one 'exit' state $q' \in S$ from which Player 1 cannot be forced to visit S again. In particular, Player 0 cannot force visiting S starting at q' which means that $\mathrm{LFP}(f_{r_i})(q') < j \leq r_i(q')$. Since $r_{i+1}(q') = \min\{r_i(q'), \mathrm{LFP}(f_{r_i})(q')\}$ we get that $r_{i+1}(q') < j \leq r_i(q')$. □

In Lemma 16, we show that if $W_j(r_i) = \emptyset$ then in the next iteration, no state gets color j unless it is its final color.

Lemma 16. *If* $W_j(r_i) = \emptyset$ *then* $W_j(r_{i+1}) = \emptyset$.

Proof. Assume that $W_j(r_{i+1}) \neq \emptyset$, that is there exists a state q such that $r_{i+1}(q) = j > r_l(q)$. Let $S := \{q' \in Q \colon r_i(q') = r_{i+1}(q)\}$. By the definition of $\mathrm{LFP}(f_{r_i})$, there is $S' \subseteq S$ that Player 0 can force visiting, starting from q. Consider also the set $S'' = \{q' \in S' \colon r_l(q') = r_{i+1}(q)\}$. If $S'' = S'$, starting at q, Player 0 can force infinite visits to S' which contradicts our assumption that $r_{i+1}(q) > r_l(q)$. Therefore, there exists $q' \in S' \setminus S''$. As $q' \in \{q'' \colon r_i(q'') > r_l(q'') = j\}$, we can infer that $W_j(r_i) \neq \emptyset$. □

Theorem 17. *Büchi ranking games can be solved in* $O(|Q| \cdot |\delta|)$ *time, where* Q *is the set of game states and* δ *is the transition function.*

Proof. The function g can be computed in time $O(|\delta|)$. We show that the number of iterations of the fixpoint computation $\mathrm{GFP}(g)$ is bounded by $|Q|$ by proving that in each iteration at least one additional state gets its final color.

For an iteration i, let $j = \min\{j' \colon W_{j'}(r_i) \neq \emptyset\}$. By Lemma 15, $\bigcup_{j'=j}^{k} W_{j'}(r_i) \setminus \bigcup_{j'=j}^{k} W_{j'}(r_{i+1}) \neq \emptyset$. By minimality of j, $\bigcup_{j'=0}^{j-1} W_{j'}(r_i) = \emptyset$. By Lemma 16, $\bigcup_{j'=0}^{j-1} W_{j'}(r_{i+1}) = \emptyset$. Therefore we have $\bigcup_{j'=j}^{k} W_{j'}(r_i) \setminus \bigcup_{j'=0}^{k} W_{j'}(r_{i+1}) \neq \emptyset$. This means that at least one state whose color in r_i was not its final color gets its final color in r_{i+1}. □

Remark 18. The memoryless optimal winning strategy for Player 0 can be identified during the computation of values described above. In the final reachability computation with the coloring function as $\mathrm{GFP}(g)$, if $q \in Q_0$ has color γ and is not marked as visited until the processing of color γ then the strategy for Player 0 at q is same as the strategy determined in the last $\mathrm{LFP}(f_r)$ step (a one or more step reachability computation). Otherwise, the strategy is to select the label of the outgoing transition of q that lead to q being marked as visited in the reachability computation for the coloring function $\mathrm{GFP}(g)$.

4.5 Cyclic ranking games

For a cyclic ranking game, consider the following decision problem. Given a number $i \in \mathbb{N}$, determine if Player 0 can force a word with rank $\geq i$.

Proposition 19. *The decision problem is in* $\mathrm{NP} \cap \mathrm{coNP}$.

Proof. Assume that the game is defined by the automaton $\mathcal{A}$ and the pair (c, k) where $c\colon Q \to \mathbb{N}$ is a coloring function and $k \in \mathbb{N}$ is the number of ranks. Consider a parity game over $\mathcal{A}$ using the coloring function

$$c'(q) = 2 \cdot \lfloor c(q)/k \rfloor + \begin{cases} 1 \text{ if } c(q) \geq i (\text{modulo } k) \\ 0 \text{ if } c(q) < i (\text{modulo } k). \end{cases}$$

Player 0 wins if the maximal color appearing infinitely often is odd. Clearly, Player 0 can force a win in this game if and only if the answer to the decision problem is affirmative. The proof follows from the known result that deciding the winner of a parity game is in $\mathrm{NP} \cap \mathrm{coNP}$ [7]. □

Using the above reduction, we can determine the value of a game with a cyclic ranking condition by repeated queries. Once we know the value, any strategy that wins the parity game also assures the value in the cyclic game. Since parity games have memoryless optimal strategies, we get the following proposition.

Proposition 20. *Cyclic ranking games have memoryless optimal strategies.*

5 Conclusions

We have proposed a framework for specifying prioritized requirements by associating ranks with executions and shown how to generalize classical automata-theoretic notions of acceptance to rankings. The resulting optimization analogs of verification and synthesis problems can naturally be solved by adopting symbolic fixpoint algorithms to an appropriately chosen lattice of coloring functions. In particular, we have identified the cyclic ranking condition as a means of specifying all ω-regular ranking functions, and shown that Büchi ranking games can be solved in quadratic time. Implementation using binary decision diagrams (BDDs) and algebraic decision diagrams (ADDs) [16] is planned for future work.

Acknowledgments. This research was partially supported by NSF grants 0541149 and 0524059, and by General Motors India Science Lab.

References

1. R. Alur, K. Etessami, S. L. Torre, and D. Peled. Parametric temporal logic for "model measuring". *ACM Trans. Comput. Log*, 2(3):388–407, 2001.
2. G. Bruns and P. Godefroid. Model checking with multi-valued logics. In J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella, editors, *ICALP*, volume 3142 of *Lecture Notes in Computer Science*, pages 281–293. Springer, 2004.

3. G. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Kluwer Academic Publishers, Boston, USA, 2000.
4. K. Chatterjee. A linear-time algorithm for weak parity games. Technical Report UCB/EECS-2006-153, University of California, Berkeley, 2006.
5. R. Cleaveland and B. Steffen. A linear-time model-checking algorithm for the alternation-free modal mu-calculus. In *Proc. 3rd Conference on Computer Aided Verification*, pages 48–58, 1991.
6. E. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 995–1072. Elsevier Science Publishers, 1990.
7. E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model checking for the μ-calculus and its fragments. *Theor. Comput. Sci*, 258(1-2):491–522, 2001.
8. J. P. Ernits. Memory arbiter synthesis and verification for a radar memory interface card. *Nord. J. Comput*, 12(2):68–88, 2005.
9. H. Gimbert and W. Zielonka. Perfect information stochastic priority games. In L. Arge, C. Cachin, T. Jurdzinski, and A. Tarlecki, editors, *ICALP*, volume 4596 of *Lecture Notes in Computer Science*, pages 850–861. Springer, 2007.
10. Y. Gurevich and L. Harrington. Trees, automata, and games. In *Proc. 14th Annual ACM Symposium on Theory of Computing*, pages 60–65, 1982.
11. J.R. Büchi. State-strategies for games in $F_{\sigma\delta} \cap G_{\sigma\delta}$. *Journal of Symbolic Logic*, 48:1171–1198, 1983.
12. O. Kupferman and Y. Lustig. Lattice automata. In *Proc. 8th Int. Conf. on Verification, Model Checking, and Abstract Interpretation*, pages 199 – 213, 2007.
13. O. Kupferman and Y. Lustig. Latticed simulation relations and games. In *Proc. 5th symp. on Aut. Technology for Verification and Analysis*, pages 316–330, 2007.
14. A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symposium on Foundations of Computer Science*, pages 46–77, 1977.
15. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symposium on Principles of Programming Languages*, 1989.
16. R.I. Bahar, E.A. Frohm, C.M. Gaona, G.D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic Decision Diagrams and Their Applications. In *Proc. 9th International Conference on CAD*, pages 188–191, 1993.
17. W. Thomas. On the synthesis of strategies in infinite games. In *Proc. 12th Symp. on Theoretical Aspects of Computer Science*, pages 1–13, 1995.
18. M. Y. Vardi and P. Wolper. Reasoning about infinite computations. *Inf. Comput.*, 115(1):1–37, 1994.
19. G. Weiss. Optimal scheduler for a memory card. Research report, Dep. of Computer Science and Applied Mathematics, The Weizmann Institue of Science, 2002.
20. U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158(1-2):343–359, 1996.