Synthesis from Satisficing and Temporal Goals

Suguman Bansal, Lydia Kavraki, Moshe Y. Vardi, Andrew Wells

1 University of Pennsylvania
2 Rice University
3 Tesla

Abstract

Reactive synthesis from high-level specifications that combine hard constraints expressed in Linear Temporal Logic (LTL) with soft constraints expressed by discounted-sum (DS) rewards has applications in planning and reinforcement learning. An existing approach combines techniques from LTL synthesis with optimization for the DS rewards but has failed to yield a sound algorithm. An alternative approach combining LTL synthesis with satisficing DS rewards (rewards that achieve a threshold) is sound and complete for integer discount factors, but, in practice, a fractional discount factor is desired. This work extends the existing satisficing approach, presenting the first sound algorithm for synthesis from LTL and DS rewards with fractional discount factors. The utility of our algorithm is demonstrated on robotic planning domains.

1 Introduction

Reactive synthesis is the automated construction, from a high-level description of its desired behavior, of a reactive system that continuously interacts with an uncontrollable external environment (Church 1957). Recent applications of reactive synthesis have emerged in AI for planning and robotics tasks (Camacho, Bienvenu, and McIlraith 2019; He et al. 2019; Kress-Gazit, Lahijanian, and Raman 2018). These applications can be formulated as a deterministic turn-based interaction between a controllable system player and an uncontrollable environment player. Given a specification, the synthesis task is to generate a system strategy such that all resulting interactions with the environment satisfy the specification. A large focus in this line of work has been on synthesis from Linear Temporal Logic (LTL) specifications (Pnueli 1977; Pnueli and Rosner 1989).

Yet, several desired specifications either cannot be expressed using LTL or doing is cumbersome. Examples include specifications about the quantitative properties of systems, such as rewards, costs, degrees of satisfaction, and so on. In fact, the combination of LTL with quantitative properties is used to express more nuanced and complex specifications (see Figure 1). Subsequently, synthesis algorithms from combination specifications have followed (Ding et al. 2014; He et al. 2017; Lahijanian et al. 2015).

This work investigates the problem of reactive synthesis from specifications that combine hard qualitative constraints expressed by LTL with soft quantitative constraints expressed by discounted-sum rewards. Discounted-sum rewards are well-suited for infinite-horizon executions because the discounted-sum is guaranteed to converge on infinite-sequence of costs whereas other aggregation functions such as limit-average may not. Discounted-sum encodes diminishing returns. As a result, the combination of LTL with discounted-sum rewards frequently appears in the automated construction of systems using planning and reinforcement learning (Bozkurt et al. 2020; Camacho et al. 2017, 2019; Hasanbeig et al. 2019; Kalagarla, Jain, and Nuzzo 2021; Kwiatkowska, Parker, and Wiltsche 2017). Note, however, these works only deal with a single player case (controllable system agent) while reactive synthesis also assumes the presence of an uncontrollable environment.

Broadly speaking, there are two approaches to reactive synthesis from LTL and discounted-sum rewards. The first approach is based on optimization of the discounted-sum reward. A strategy that optimizes the discounted-sum reward alone is guaranteed to exist in deterministic, turn-based settings (Shapley 1953). This existence guarantee, however, is lost upon combination with LTL constraints. For example, consider a two-state game where state $s_0$ gives negative reward and state $s_1$ positive. Each state can transition to all other states. Our LTL objective is $(\text{Globally Eventually } s_0)$. Clearly, there exists no strategy that simultaneously maximizes the discounted-sum reward and satisfies the LTL objective (Chatterjee et al. 2017). To
Due to the use of approximation, our algorithm is no longer complete. To this end, we evaluate the practical utility of our algorithm on case studies from robotics planning. Our evaluation demonstrates that our sound but incomplete procedure succeeds in efficiently constructing high-quality strategies in complex domains from nuanced constraints.

2 Preliminaries

2.1 Automata and Formal Specifications

Büchi Automata and Co-Safety Automata. A Büchi automaton is a tuple $\mathcal{A} = (S, \Sigma, \delta, s_0, \mathcal{F})$, where $S$ is a finite set of states, $\Sigma$ is a finite input alphabet, $\delta \subseteq (S \times \Sigma \times S)$ is the transition relation, state $s_T \in S$ is the initial state, and $\mathcal{F} \subseteq S$ is the set of accepting states. A Büchi automaton is deterministic if for all states $s$ and inputs $a$, $\{s'|\langle s, a, s'\rangle \in \delta \text{ for some } s'\} \leq 1$. For a word $w = w_0w_1 \cdots \in \Sigma^\omega$, a run $\rho$ of $w$ is a sequence of states $s_0s_1 \cdots$ s.t. $s_0 = s_T$, and $t_i = (s_i, w_i, s_{i+1}) \in \delta$ for all $i$. Let $\inf(\rho)$ denote the set of states that occur infinitely often in run $\rho$. A run $\rho$ is an accepting run if $\inf(\rho) \cap \mathcal{F} \neq \emptyset$. A word $w$ is an accepting word if it has an accepting run. Büchi automata are closed under set-theoretic union, intersection, and complementation (Thomas, Wilke et al. 2002).

A co-safety automaton is a deterministic Büchi automaton with a single accepting state. Additionally, the accepting state is a sink state (Kupferman and Vardi 1999).

Comparator Automata. Given an aggregate function $f : \mathbb{Z}^\omega \rightarrow \mathbb{R}$, equality or inequality relation $R \in \{\leq, \geq, =, \neq\}$, and a threshold value $v \in \mathbb{Q}$, the comparator automaton for $f$ with upper bound $\mu$, relation $R$, and threshold $v \in \mathbb{Q}$ is an automaton that accepts an infinite word $w$ over the alphabet $\Sigma = \{-\mu, -\mu + 1, \cdots \mu\}$ iff $f(A)Rv$ holds (Bansal, Chaudhuri, and Vardi 2018a,b,c).

The discounted-sum of an infinite-length weight-sequence $W = w_0w_1 \cdots$ with discount factor $d > 1$ is given by $DS(W, d) = \sum_{i=0}^\infty w_i d^{-i}$. The comparator automata for the discounted-sum has been shown to be a safety or co-safety automata when the discount factor $d > 1$ is an integer, for all values of $R$ and $v$. It is further known to not form a Büchi automata for non-integer discount factors $d > 1$, for all values of $R$, $\mu$ and $v$ (Bansal and Vardi 2019; Bansal, Chatterjee, and Vardi 2021).

Linear Temporal Logic. Linear Temporal Logic (LTL) extends propositional logic with infinite-horizon temporal operators. The syntax of LTL is defined as $\varphi := a \in AP | \neg \varphi | \varphi \land \varphi | \varphi \lor \varphi | \varphi \rightarrow \varphi | X \varphi | F \varphi | G \varphi$. Here $X$ (Next), $U$ (Until), $F$ (Eventually), $G$ (Always) are temporal operators. The semantics of LTL can be found in (Pnueli 1977).

2.2 Two-Player Graph Games

Reachability Games. A reachability game $G = (V, v_{init}, E, \mathcal{F})$ consists of a directed graph $(V, E)$, initial state $v_{init}$, and non-empty set of accepting states $\mathcal{F} \subseteq V$. The set $V$ is partitioned into $V_0$ and $V_1$. For convenience, we assume every state has at least one successor. A game is played between two players $P_0$ and $P_1$.  

this end, an alternate synthesis task is to compute an optimal strategy from those that satisfy the LTL constraint (Wen, Ehlers, and Topcu 2015). Unfortunately, even here existing synthesis algorithms may generate a sub-optimal strategy. Overall, synthesis algorithms from LTL and discounted-sum rewards that optimize the discounted-sum reward, in one way or another, have hitherto failed to provide guarantees of correctness or completeness.

The second approach to synthesis from LTL and discounted-sum rewards is based on satisfying the discounted-sum reward. A strategy is satisficing with respect to a given threshold value $v \in \mathbb{Q}$ if it guarantees the discounted-sum reward of all executions will exceed $v$. The synthesis task, therefore, is to compute a strategy that satisfies the LTL specification and is satisficing w.r.t. the threshold value. The advantage of this approach is that when the discount factor is an integer, an existing synthesis algorithm is both sound and complete (Bansal, Chaudhuri, and Vardi 2018a,b). The central contribution of this work is a theoretically correct and is practically applicable. We show that comparators for approximations of discounted-sum goals for the case when the discount factor ranges between 1 and 2 (specifically of the form $\mu = 2^{-k}$ for positive integer values of $k$). To the best of our knowledge, this is the first synthesis algorithm from LTL and discounted-sum rewards, and thus preserves soundness.

Due to the use of approximation, our algorithm is no longer complete. To this end, we evaluate the practical utility of our algorithm on case studies from robotics planning. Our evaluation demonstrates that our sound but incomplete procedure succeeds in efficiently constructing high-quality strategies in complex domains from nuanced constraints.
A play in the game is created by the players moving a token along the edges as follows: at the beginning, the token is at the initial state. If the token’s current position \( v \) belongs to \( V_i \), then \( P_i \) chooses the next position from the successors of \( v \). Formally, a play \( \rho = v_0v_1v_2 \ldots \) is an infinite sequence of states such that \( v_0 = v_{\text{init}} \) and \( (v_k, v_{k+1}) \in E \) for all \( k \geq 0 \). A play is winning for player \( P_i \) in the game if it visits an accepting state, and winning for player \( P_o \) otherwise.

A strategy for a player is a recipe that guides the player on which state to go next to based on the history of a play. A strategy is winning for a player \( P_i \) if for all strategies of the opponent player \( P_{o} \), all resulting plays are winning for \( P_i \). To solve a graph game is to determine whether there exists a winning strategy for player \( P_i \). Reachability games are solved in \( O(|V| + |E|) \) (Thomas, Wilke et al. 2002).

**Quantitative Graph Games.** A quantitative graph game (quantitative game, in short) is given by \( G = (V = V_o \cup V_i, \nu_{\text{init}}, E, \gamma, \mathcal{L}, d) \) where \( V, V_o, V_i, \nu_{\text{init}}, E, \) plays, and strategies are defined as earlier. Each edge is associated with a cost determined by the cost function \( \gamma : E \rightarrow \mathbb{R} \), and \( d > 1 \) is the discount factor. The cost-sequence of a play \( \rho \) is the sequence \( w_0w_1w_2 \ldots \) where \( w_k = \gamma((v_k, v_{k+1})) \) for all \( i \geq 0 \). The cost of play \( \rho \) is the discounted-sum of its cost sequence with discount factor \( d > 1 \). A labelling function \( \mathcal{L} : V \rightarrow 2^{\mathcal{AP}} \) maps states to propositions from the set \( \mathcal{AP} \). The label sequence of a play \( \rho \) is given by \( \mathcal{L}(v_0)\mathcal{L}(v_1)\ldots \).

### 3 Problem Formulation and Overview

The two players, the controllable system and uncontrollable environment, interact in a domain described by a quantitative game \( G \). The specification of the system player is a combination of hard and soft constraints.

The hard constraint is given as by an LTL formula \( \varphi \). A play in \( G \) satisfies formula \( \varphi \) if its labelled sequence satisfies the formula. We say, a strategy for the system player satisfies a formula \( \varphi \) if it guarantees that all resulting plays will satisfy the formula. We call such a strategy \( \varphi \)-satisfying.

The soft constraints are given by satisficing goals. W.l.o.g., the system and environment players maximize and minimize the cost of plays, respectively. Given a threshold value \( v \in \mathbb{Q} \), a play \( \rho \) is \( v \)-satisficing for the system (maximizing) player if its cost is greater than or equal to \( v \). Conversely, a play is \( v \)-satisficing for the environment (minimizing) player if its cost is less than \( v \). A strategy is \( v \)-satisficing for a player if it guarantees all resulting plays are \( v \)-satisficing for the player.

We are interested in solving the following problem:

**Problem (Reactive Synthesis from Satisficing and Temporal Goals).** Given a quantitative game \( G \), a threshold value \( v \in \mathbb{Q} \), and an LTL formula \( \varphi \), the problem of reactive synthesis from satisficing and temporal goals is to compute a strategy for the system player that is \( \varphi \)-satisfying and \( v \)-satisficing for the player, if such a strategy exists.

The problem is solved for integer discount factors (Bansal, Chatterjee, and Vardi 2021).

**Algorithm Overview.** In this paper, we extend to fractional discount factors \( 1 < d < 2 \), yielding practical applications of the synthesis problem. In particular, we solve the problem for \( d = 1 + 2^{-k} \) where \( k > 0 \) is an integer. Since the comparator for discounted-sum with fractional discount factors are not representable by Büchi automata, we construct a comparator automata for lower approximations of discounted-sum. This comparator soundly captures the criteria for \( \psi \)-satisficing for system player. In particular, if the comparator accepts the weight sequence of a play, then the play must be \( v \)-satisficing for the player. Therefore, just like LTL goals, the satisficing goal is also soundly captured by an automaton. Thus, we can reduce the synthesis problem to parity games via appropriate synchronized product constructions of both the automata-based goals.

The comparator construction for approximation of discounted-sum is presented in Section 4 and the reduction to games on graphs is presented in Section 5.

### 4 Comparator Construction

This section develops the key machinery required to design our theoretically sound algorithm for synthesis from temporal and satisficing goals with fractional discount factors. We construct comparator automata for a lower approximation of discounted-sum for fractional discount factors of the form \( d = 1 + 2^{-k} \) where \( k > 0 \) is an integer. We show these comparators are represented by co-safety automata.

This section is divided in two parts. Section 4.1 defines an aggregate function that approximates the discounted-sum. Section 4.2 constructs a comparator for this function.

Unless stated otherwise, we assume the approximation factor is of the form \( \varepsilon = 2^{-p} \) where \( p > 0 \) is an integer. Please refer to the Appendix for missing proofs and details.

#### 4.1 Approximation of Discounted-Sum

Given parameters \( k, p > 0 \) of the discount factor and the approximation factor, respectively, let \( \text{roundLow}(x, k, p) \) be the largest integer multiple of \( 2^{-(p+k)} \) that is less than or equal to \( x \), where \( x \in \mathbb{R} \). Let \( W[[n]] \) denote the \( n \)-length prefix of a weight-sequence \( W \).

Then, the lower approximation of discounted-sum of an infinite-length weight-sequence \( W \) with discount factor \( d > 1 \) and approximation factor \( \varepsilon > 0 \) is defined as

\[
\text{DSLow}(W, k, p) = \lim_{n \to \infty} \frac{\text{gapLow}(W[[n]], k, p)}{d^{n-1}}
\]

where the lower gap value of a finite-length weight-sequence \( U \) is defined as

\[
\text{gapLow}(U, k, p) = \begin{cases} 0, & \text{if } |U| = 0 \\ \text{roundLow}(d \cdot \text{gapLow}(V, k, p) + v, k, p), & \text{if } U = V \cdot v \end{cases}
\]

Finally, the definition of \( \text{DSLow} \) is completed by proving \( \text{DSLow} \) approximates the discounted-sum of sequences within an additive factor of \( d \cdot \varepsilon \):

**Theorem 1.** Let \( d = 1 + 2^{-k} \) be the discount factor and \( \varepsilon = 2^{-p} \) be the approximation factor; for rational parameters \( p, k > 0 \). Let \( W \) be an infinite-length weight sequence. Then, \( 0 \leq \text{DS}(W, d) - \text{DSLow}(W, k, p) < d \cdot \varepsilon \).
Proof Sketch. While the definition of the lower approximation of discounted-sum may look notationally dense, it is inspired by an alternate definition of discounted-sum:

\[ DS(W, d) = \lim_{n \to \infty} \frac{\text{gap}(W[\ldots n], d)}{d^{n-1}} \]

where \( \text{gap}(U, d) = 0 \) if \(|U| = 0 \) and \( \text{gap}(U, d) = d \cdot \text{gap}(V, d) + v \) if \( U = V \cdot v \).

Intuitively, the lower gap value approximates gap. Subsequently, \( \text{DSLow} \) approximates the discounted-sum. \( \square \)

4.2 Comparator for Approximation of DS

This section presents the construction of a comparator for lower approximation of discounted-sum defined above.

**Definition 1** (Comparator automata for lower approximation of DS). Let \( \mu > 0 \) be an integer bound, and \( k, p > 0 \) be integers. The comparator automata for lower approximation of discounted sum with discount factor \( d = 1 + 2^{-k} \), approximation factor \( \epsilon = 2^{-p} \), upper bound \( \mu \), threshold value \( v \in \mathbb{Q} \), and inequality relation \( R \in \{ \leq, \geq \} \) is an automaton over infinite weight sequences \( W \) over the alphabet \( \Sigma = \{-\mu, \ldots, \mu\} \) that accepts \( W \) iff \( \text{DSLow}(W(k, p)) R v \).

**Construction Sketch.** We sketch the construction of the comparator for lower approximation of discounted-sum. For sake of exposition, we begin the construction for threshold value \( v = 0 \). W.l.o.g., we present for the relation \( \geq \). Notations \( \mu, d = 1 + 2^{-k}, \epsilon = 2^{-p} \), and \( W \) are from Definition 1.

The lower gap value of prefixes of a weight-sequence can be used as a proxy for acceptance of a weight-sequence in the comparator for the following two observations:

1. \( \text{DSLow}(W(k, p)) \geq 0 \) for an infinite-length weight sequence \( W \) iff there exists a finite prefix \( A \) of \( W \) such that \( \text{gapLow}(A, k, p) \geq \mu \cdot 2^k + 2^{-p} \). Let us denote \( \mu \cdot 2^k + 2^{-p} \) by \( \text{upperLimit} \).

2. \( \text{DSLow}(W(k, p)) \) cannot be greater than or equal to \( 0 \) iff there exists a finite prefix \( A \) of \( W \) such that \( \text{gapLow}(A, k, p) \leq -\mu \cdot 2^k \). Let us denote \( -\mu \cdot 2^k \) by \( \text{lowerLimit} \).

So, the core idea behind our construction is two fold: (a) use states of the comparator to record the lower gap value of finite-length prefixes, and (b) assign transitions so that the final state of finite-prefix corresponds to its lower gap value.

To this end, we set the initial state to \( 0 \) as the lower gap value of the 0-length prefix is \( 0 \). The transition relation mimics the inductive definition of lower gap value, i.e. there is a transition from a state \( s \) on the alphabet (weight) \( a \) to state \( t \) if \( t = \text{roundLow}(a \cdot s + a, k, p) \). These ensure that the lower gap value of a finite-state word (finite-length weight-sequence) is detected from the final state in its run. Clearly, the transition relation is deterministic.

The final piece of the construction is to restrict the automata to finitely many states and to determine its accepting states. Note that due to the enumerated observations it is sufficient to track the lower gap value for only as long as it lies between \( \text{lowerLimit} \) and \( \text{upperLimit} \). Observe that there are only finitely many such values of interest since lower gap value is always an integer multiple of \( 2^{-(p+k)} \). Thus, we have obtained a finite number of states. By the first observation, state \( \text{upperLimit} \) is made an accepting sink since every weight-sequence that visits \( \text{upperLimit} \) must be accepted by the comparator. Similarly, by the second observation, the state \( \text{lowerLimit} \) is made a non-accepting sink. This completes the construction for threshold value \( v = 0 \).

To extend the construction to a non-zero threshold value \( v \in \mathbb{Q} \), let \( V \) be a lasso weight-sequence s.t. \( DS(V, d) = v \), the comparator incorporates \( V \) into its construction. Specifically, we construct a comparator that accepts \( W \) iff \( \text{DSLow}(W - V, k, p) \geq 0 \). So, when \( W \) is accepted then \( DS(W, d) \geq v \), otherwise \( DS(W, d) \leq v + d \cdot \epsilon \).

As an example, Figure 2 illustrates a snippet of the comparator with discount factor \( d = 1 + 2^{-1} \), approximation factor \( \epsilon = 2^{-1} \), upper bound \( \mu = 1 \), threshold value \( v = 0 \), and relation \( \geq \). As one can see, weight sequence \( A = -1, 0, 1^\omega \) with discounted-sum \( \frac{1}{3} \) is accepting and weight sequence \( B = -1, -1, 1^\omega \) with discounted-sum \( \frac{1}{3} \) is non-accepting.

**Theorem 2.** The comparator automata for lower approximation of discounted sum with discount factor \( d = 1 + 2^{-k} \), approximation factor \( \epsilon = 2^{-p} \), upper bound \( \mu \), threshold value \( v \), and inequality relation \( R \in \{ \leq, \geq \} \) is a co-safety automata with \( O(2^{-k}) \) states, where \( k, p > 0 \) are integers.

Therefore, \( DS(W, d) \geq 0 \) if a weight-sequence \( W \) is accepted by the comparator constructed above, and \( DS(W, d) < d \cdot \epsilon \) otherwise (Theorem 1-2).

5 Reactive Synthesis from Satisficing and Temporal Goals

This section presents the central contribution of this work. We present a theoretically sound algorithm for reactive synthesis from LTL and satisficing discounted-sum goals for the case when the discount factor ranges between 1 and 2, referred to as fractional discount factors hereon.

Our algorithm utilizes the comparator automata for the lower approximation of discounted-sum for fractional discount factors constructed in Section 4. For ease of exposition, we present our solution in two parts. First, we present an algorithm for reactive synthesis from satisficing goals only in Section 5.1. Next, we extend this algorithm to solve our original problem in Section 5.2.
5.1 Satisficing Goals

We describe an automata-based solution for reactive synthesis from satisficing goals with fractional discount factors. Our solution reduces to reachability games using the comparator for lower approximation of discounted sum.

The key idea behind our solution is that the said comparator can be treated as a sufficient criteria to compute a satisficing strategy for the system player. We explain this further. Take a comparator for the lower approximation for discounted-sum with discount factor $\varepsilon$, threshold value $v \in \mathbb{Q}$, and relation $\geq$. Then, a play in the quantitative game is $v$-satisficing for the system player if the comparator accepts the cost sequence of the play. This can be derived directly from Theorem 1-2. So, a strategy is satisficing for the system player if it is winning with respect to the comparator. To this end, we construct a synchronized product of the quantitative game with the comparator. The resulting product game is a reachability game since the comparator is represented by a co-safety automata. Formally,

**Theorem 3.** Let $G$ be a quantitative game with discount factor $d = 1 + 2^{-k}$, for integer $k > 0$. Let $v \in \mathbb{Q}$ be the threshold value and $\varepsilon = 2^{-p}$ be the approximation factor. There exists a reachability game $GA$ such that

- If the system has a winning strategy in $GA$, then the system has a $v$-satisficing strategy in $G$.
- If the environment has a winning strategy in $GA$, then the environment has a $v + d \cdot \varepsilon$-satisficing strategy in $G$.

**Proof.** The product game synchronizes costs along edges in the quantitative game with the alphabet of the co-safety comparator. Let $G = (V = V_0 \uplus V_1, v_{\text{init}}, E, \gamma)$ be a quantitative game. Let $\mu > 0$ be the maximum absolute value of costs along transitions in $G$. Then, let $A = (S, s_I, \Sigma, \delta, F)$ be the co-safety comparator for lower approximation of discounted-sum with upper bound $\mu$, discount factor $d = 1 + 2^{-k}$, approximation factor $\varepsilon = 2^{-p}$, threshold value $v$, and relation $\geq$. Then, the reachability game is $GA = (W = W_0 \uplus W_1, s_0 \times \text{init}, \delta_W, F_W)$. Here, $W = V \times S$, $W_0 = V_0 \times S$, and $W_1 = V_1 \times S$. Clearly, $W_0$ and $W_1$ partition $W$. The edge relation $\delta_W \subseteq W \times W$ is defined such that edge $((v, s), (v', s')) \in \delta_W$ synchronizes between transitions $(v, v') \in E$ and $(s, a, s') \in \delta$ if $a = \gamma((v, v'))$ is the cost of transition $(v, v')$ in $G$. State $s_0 \times \text{init}$ is the initial state and $F_W = V \times F$.

It suffices to prove that a play is winning for the system in $GA$ iff its cost sequence $A$ in $G$ satisfies $\text{DSLow}(A, k, p) \geq 0$. This is ensured by the standard synchronized product construction and Theorem 2. The reachability game $GA$ is linear in size of the quantitative graph and the comparator.

Theorem 3 describes a sound algorithm for reactive synthesis from satisficing goals when the discount factor is fractional. The algorithm is not complete since it is possible that there is a $v + d \cdot \varepsilon$-satisficing strategy for the environment even when the system has a $v$-satisficing strategy.

5.2 Satisficing and Temporal Goals

Finally, we present our theoretically sound algorithm for synthesis from LTL and discounted-sum satisficing goals for fractional discount factors.

The algorithm is essentially a sum of two parts. The algorithm combines the automata-based solution for satisficing goals (presented in Section 5.1) with the classical automata-based solutions for LTL goals (Pnueli and Rosner 1989). Solving satisficing goals forms a reachability game while solving LTL goals forms a parity game. Thus, the final game which combines both of the goals will be a parity game. Lastly, the algorithm will inherit the soundness guarantees from both of its parts.

**Theorem 4.** Let $G$ be a quantitative game with discount factor $d = 1 + 2^{-k}$, for integer $k > 0$. Let $\varphi$ be an LTL formula and $v \in \mathbb{Q}$ be a threshold value. Let $\varepsilon = 2^{-p}$ be the approximation factor. There exists a parity game $GA$ such that

- If the system has a winning strategy in $GA$, then the system has a $v$-satisficing and $\varphi$-satisfying strategy in $G$.
- If the environment has a winning strategy in $GA$, then there exists a $v + d \cdot \varepsilon$-satisficing strategy or it has a winning strategy w.r.t. LTL formula in $G$.

**Proof Sketch.** The reduction consists of two steps of synchronized products: first with the comparator to fulfill the $v$-satisficing goal and then with the automaton corresponding to the LTL goal. The first step conducts the reduction from Theorem 3 while lifting the labelling function from the quantitative game to the reachability game: If a state $s$ is labeled by $l$ in the quantitative game, the all states of the form $(s, q)$ will be labeled by $l$ in the reachability game. The second product synchronizes between the atomic propositions in the reachability game (with a labelling function) and the deterministic parity automaton (DPA) corresponding to the LTL specification, thus combining their winning conditions.

Observe that the product construction is commutative, i.e., one can first construct the product of $G$ with the DPA of the LTL goal and then with the comparator automata.

In either case, we generate a parity game of size linear in $|G|$, DPA of the LTL specification, and the comparator. A winning strategy for the system player in this game is also $v$-satisficing and $\varphi$-satisfying for the same player in $G$.

A salient feature of our algorithm is that the complexity to solve the final product game is primarily governed by the temporal goal and not the satisficing goal. What we mean is that if the temporal goal is given by a fragment of LTL, such as co-safe LTL (Lahijanian et al. 2015), then the final product game would be reachability game. This is because co-safe LTL formulas are represented by co-safe automata and thus their combination with comparators would also be a co-safe automata. More generally, if the temporal goal is a conjunction of safety and reachability goals, the resulting game would be a weak-Büchi game, which are also solved in linear time in size of the game (Chatterjee 2008). This demonstrates that even though the comparator contributes to growing the state-space of the game linearly,
whether the game is solved using efficient linear-time algorithms or higher complexity algorithms for parity games is determined by the temporal goal.

This feature has implications on the practicality of our algorithm. In practice, it has been observed that wide-ranging temporal goals in robotics domains can be expressed in simpler fragments and variants of LTL, such as co-safe LTL (Lahijanian et al. 2015) and LTLf (He et al. 2017). These fragments can be expressed as conjunctions of safety and reachability goals. For this fragment synthesis from temporal and satisficing goals can be solved in linear-time.

6 Case Studies

The objective of our case studies is to demonstrate the utility of reactive synthesis from LTL and satisficing goals in realistic domains inspired from robotics and planning. Since ours is the first algorithm to offer theoretical guarantees with fractional discount factors, there are not any baselines to compare to. So, we focus on our scalability trends and identify future scalability opportunities.

6.1 Design and Set-up

We examine our algorithm on two challenging domains inspired from robotic navigation and manipulation problems. Source code and benchmarks are open source1.

Grid World. The robot-human interaction is based on a classic \( n \times n \) grid world domain (see Fig 1). The grid simulates a grocery store with static obstacles, e.g., placements of aisles. Each agent controls its own location and is only permitted to move in the cardinal directions. The robot’s LTL constraint is to reach the locations of all items on its grocery list without colliding with the walls (in grey) or the dynamic human, thus combining safety and reachability goals. The robot’s soft constraints are modelled to achieve two behaviors. The first one is to distance itself from the human. The second is to encode promptness in fulfilling its reachability goal reach\(_{banana}\). We model distancing with quantitative rewards using the Manhattan distance between the two agents. Suppose, the locations of the players are \((x_0, y_0)\) and \((x_1, y_1)\), then the reward received by the robot is given by \(\text{negative_reward} = \frac{|x_0 - x_1| + |y_0 - y_1|}{1}\), where negative\(_{reward} < 0\) is an integer parameter. We model promptness with an integer positive\(_{reward} > 0\) which the robot receives only when it reaches a location of each item on its grocery list for the first time. The rewards are additive, i.e., the robot receives the sum of both rewards in every grid configuration. Then, it is reasonable to say that a play accomplishes these two behaviors if the discounted-sum reward of the robot is greater than or equal to 0, i.e., 0-satisficing plays/strategies are good for the robot.

Conveyor Belt. Our second case study is inspired by cutting-edge applications in manipulation tasks (Wells et al. 2021). A robot must operate along a \( r \times c \) conveyor belt with \( r \) rows and \( c \) columns across from a human. When out of reach of the human, the robot can move quickly. Otherwise, it must proceed more slowly. The blocks move down the conveyor belt at a constant speed. Each agent controls the location of its arm. The human also controls the placement of new objects. New blocks are placed whenever a block is removed to maintain a constant number of blocks on the belt at all times.

The robot’s LTL goal is to avoid interfering with the human. As soft constraints, the robot gains a positive\(_{reward}\) for every object it grasps and a negative\(_{reward}\) for every object that falls off the belt. The rewards are additive in every belt configuration. The robot’s goal is to ensure its total discounted-sum reward exceeds 0.

On grid world, we take \( n = 4, 6, 8, 10 \). On conveyor belt, we take \( r \times c = 4 \times 3, 5 \times 3 \) with 2 or 3 blocks. The hardness of our benchmarks is illustrated Table 1. The benchmarks have so many states since both scenarios have a large number of unique configurations.

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>Number of States</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid World</td>
<td></td>
</tr>
<tr>
<td>( n = 4 )</td>
<td>397</td>
</tr>
<tr>
<td>( n = 6 )</td>
<td>2407</td>
</tr>
<tr>
<td>( n = 8 )</td>
<td>8093</td>
</tr>
<tr>
<td>( n = 10 )</td>
<td>20572</td>
</tr>
<tr>
<td>Conveyor Belt</td>
<td></td>
</tr>
<tr>
<td>( r \times c = 4 \times 3 ), 2 blocks</td>
<td>9966</td>
</tr>
<tr>
<td>( r \times c = 5 \times 3 ), 2 blocks</td>
<td>31547</td>
</tr>
<tr>
<td>( r \times c = 5 \times 3 ), 3 blocks</td>
<td>60540</td>
</tr>
</tbody>
</table>

Table 1: Complexity of Benchmarks: Number of states in product of the labelled quantitative game with the automata of its LTL specification.

Figure 3: Scalability (Number of states). Plots runtime on Grid World with positive\(_{reward} = 10\) and negative\(_{reward} = -2\). x-axis are \( n = 4, 6, 8, 10 \).
Our evaluation demonstrates that our solution successfully scales to very large benchmarks. Despite their difficulty, we solve almost all of our benchmarks (Figure 5). Runtime examination indicates that our algorithm is linear in size of the game and the comparator, in practice. The scalability trends in size of the game for varying discount factors are shown in Figure 3. Determining the dependence on the comparator automata is more involved since its size depends on several parameters, namely positive_reward, the discount factor, and the approximation factor. Figure 4 suggests the algorithm is linear in positive_reward. The margin between the three discount factor curves on Fig 3-4 suggests a significant blow-up as the discount factor nears 1. Additional experiments (see Appendix) that vary the approximation factor also display a significant blow-up as the approximation factor decreases. These are not alarming since the size of the comparator is in the order of $O(\text{positive_reward})$, $O((d-1)^{-2})$ and $O(\varepsilon^{-1})$. These reflect that our current implementation is faithful to the theoretical analysis of the algorithm.

These are encouraging results as our implementation uses explicit state representation. The overhead of this state representations can be very high. In some cases, we observed that for the large benchmarks about 70% of the total compute time may be spent in constructing the product explicitly. Despite these issues with explicit-state representation, our algorithm efficiently scales to large and challenging benchmarks. This indicates potential for further improvements.

In terms of quality of solutions, the resulting strategies are of better quality. For example, in Figure 1 we observed that as the discount factor becomes smaller the robot is able to reason for a longer horizon and not get "trapped". Another benefit are the soundness guarantees. They are especially valuable in environments such as the Conveyor belt which are so complex that they preclude a manual analysis.

To conclude, our case studies demonstrates the promise of our approach in terms of its ability to scale and utility in practical applications, and encourage future investigations.

## 7 Conclusion

Combining hard constraints (qualitative) with soft constraints (quantitative) is a challenging problem with many applications to automated planning. This paper presents the first sound algorithm for reactive synthesis from LTL constraints with soft discounted-sum rewards when the discount factor is fractional. Our approach uses an automata-based method to solve the soft constraints, which is then elegantly combined with existing automata-based methods for LTL constraints to obtain the sound solution. Case studies on classical and modern domains of robotics planning demonstrate use cases, and also, shed light on recommendations for future work to improve scalability to open up exciting applications in robotics e.g. warehouse robotics, autonomous driving, logistics (supply-chain automation).

## Acknowledgements

We thank anonymous reviewers. This work is supported in part by NSF grant 2030859 to the CRA for the CIFellows Project, NSF grants IIS-1527668, CCF-1704883, IIS-1830549, CNS-2016656, DoD MURI grant N00014-20-1-2787, and an award from the Maryland Procurement Office.

## References


A.1 Definition of lower approximation of DS is well-defined

For an infinite-length weight sequence $W$, let $W[\ldots n]$ denote its $n$-length prefix for $n \geq 0$. Given parameters $k$ and $p$ of the discount factor and the approximation factor, respectively, let the resolution be given by $r = 2^{-p-k}$. For real number $x \in \mathbb{R}$, let $\text{roundLow}(x, k, p)$ denote the largest integer multiple of the resolution that is less than or equal to $x$. Formally, $\text{roundLow}(x, k, p) = \lceil x \cdot 2^{-(p+k)} \rceil$ for an integer $i \in \mathbb{Z}$ such that for all integers $j \in \mathbb{Z}$ for which $j \cdot 2^{-(p+k)} \leq x$, we get that $j \leq i$. Then it is clear that for all real values $x \in \mathbb{R}$, $0 \leq x - \text{roundLow}(x, k, p) < 2^{-(p+k)}$. Then, the lower approximation of discounted-sum is defined as follows:

**Definition 2 (Lower Approximation of Discounted-Sum).**

Given discount factor $d = 1 + 2^{-k}$ and approximation factor $\varepsilon = 2^{-p}$ with rational-valued parameters $k, p \in \mathbb{Q}$. The lower gap of a finite-length weight sequence $U$, denoted $\text{gapLow}(U, k, p)$ is 0 if $|U| = 0$ and $\text{roundLow}(\text{gapLow}(V, k, p) + v, k, p)$ if $U = V \cdot v$. Then, the lower approximation of discounted sum of an infinite-length weight sequence $W$ with discount factor $d$ and approximation factor $\varepsilon$ is denoted by and defined as follows:

$$\text{DSLow}(W, k, p) = \lim_{n \to \infty} \frac{\text{gapLow}(W[\ldots n], k, p)}{d^{n-1}}$$

Our goal is to show that $\text{DSLow}(W, k, p) = \lim_{n \to \infty} \frac{\text{gapLow}(W[\ldots n], k, p)}{d^{n-1}}$ is well-defined, i.e., the limit of $\frac{\text{gapLow}(W[\ldots n], k, p)}{d^{n-1}}$ exists as $n \to \infty$ (Theorem 5). Next,
we need to show that Definition 2 indeed computes a value that approximates the discounted-sum of a sequence (Theorem 1).

We begin with some additional notation. Let \( d > 1 \) be a rational valued discount factor. The recoverable gap of a finite, bounded, weight-sequence \( U \) and discount factor \( d \), denoted by \( \text{gap}(U, d) \), is 0 if \( |U| = 0 \) and \( \text{gap}(V, d) = d \cdot \text{gap}(V, d) + v \) if \( U = V \cdot v \). Intuitively, the recoverable gap of a finite weight-sequence is the normalized discounted-sum of the finite weight-sequence. Then, it is known that for an infinite-length weight sequence \( W \)

\[
\lim_{n \to \infty} \frac{\text{gap}(W[\ldots n], d)}{d^{n+1}} \to DS(W, d).
\]

The following holds:

**Lemma 1.** Let \( d = 1 + 2^{-k} \) and \( 2^{-p} \) be the discount-factor and precision, for rational numbers \( k, p > 0 \). Let \( \mu > 0 \) be the upper-bound. Let \( W \) be an infinite and bounded weight-sequence. Then, there exists an infinite and bounded rational number weight-sequence \( U \) such that for all \( n > 0 \),

\[
\text{gap}(W[\ldots n], k, p) = \text{gap}(U[\ldots n], d).
\]

**Proof.** For sake of simplicity, we assume \( W \) is an integer weight sequence. The proof extends to rational weight sequences as well. Let \( W = w_0w_1w_2\ldots \) such that for all \( i > 0 \), \( |w_i| \in \mathbb{Z} \) and \( |w_i| < \mu \). We will construct the desired infinite-length weight sequence \( U \) inductively.

**Base Case.** Consider the 1-length prefix of \( W \), \( W[1] = (w_0) \). By definition, \( \text{gap}(W[1], k, p) = \text{roundLow}(w_0, k, p) = w_0 \). So, we set \( u_0 \), the 0-th element of \( U \), to be \( w_0 \). Clearly, \( \text{gap}(U[1], d) = w_0 = \text{gap}(W[1], k, p) \).

**Inductive Hypothesis.** For an \( n > 0 \), let there exist an \( n \)-length rational-number sequence \( (u_0u_1\ldots u_{n-1}) \) bounded by \( \mu \) such that for all \( m \leq n \) it holds that \( \text{gap}(W[\ldots m], k, p) = \text{gap}(u_0u_1\ldots u_{n-1}, d) \).

**Induction Step.** It suffices to prove that the \( n \)-length weight-sequence \( (u_0u_1\ldots u_{n-1}) \) can be extended by appending a rational-number \( u_n \) bounded by \( \mu \) such that

\[
\text{gap}(W[\ldots (n+1)], k, p) = \text{gap}(u_0u_1\ldots u_{n-1}, d).
\]

By definition, \( \text{gap}(W[\ldots (n+1)], k, p) = \text{roundLow}(d \cdot \text{gap}(W[\ldots n], k, p) + w_n, k, p) \). By definition of roundLow, there exists a \( 0 \leq \varepsilon_n < 2^{-(p+k)} \) such that \( \text{roundLow}(d \cdot \text{gap}(W[\ldots n], k, p) + w_n, k, p) = d \cdot \text{gap}(W[\ldots n], k, p) + w_n - \varepsilon_n \). Therefore, we obtain \( \text{gap}(W[\ldots (n+1)], k, p) = d \cdot \text{gap}(W[\ldots n], k, p) + w_n - \varepsilon_n \). By I.H., we see that \( \text{gap}(W[\ldots (n+1)], k, p) = d \cdot \text{gap}(u_0u_1\ldots u_{n-1}, d) + w_n - \varepsilon_n \). Set \( u_n = w_n - \varepsilon_n \). Then, we obtain that \( |u_n| \leq \mu \).

**Lemma 2.** Let \( d = 1 + 2^{-k} \) and \( 2^{-p} \) be the discount factor and approximation factor, for rational numbers \( k, p > 0 \). Let \( \mu > 0 \) be the upper-bound. Let \( W \) be a non-empty, finite-length, and bounded weight sequence. Then,

\[
0 \leq \text{gap}(W, d) - \text{gap}(W, k, p) < \text{gap}(R, d)
\]

where \( R \) is \(|W|\)-length resolution sequence.

**Proof.** The proof proceeds by induction on the length of the weight sequence.

**Base Case.** When \( |W| = 1 \). Let \( W = w_0 \) where \( w_0 \in \mathbb{Z} \) and \( |w_0| \leq \mu \). Then \( \text{gap}(W, d) = \text{gap}(W, k, p) = w_0 \). Then \( \text{gap}(W, d) = w_0 \) and \( \text{gap}(W, k, p) = \text{roundLow}(w_0, k, p) \). Thus, trivially, \( 0 \leq \text{gap}(W, d) - \text{gap}(W, k, p) < 2^{-(p+k)} = \text{gap}(R, d) \), where \( R \) is the resolution sequence of length 1.

**Inductive Hypothesis.** For all weight-sequences \( W \) of length \( n \geq 1 \), it is true that there exists a \( 0 \leq \varepsilon_n < 2^{-(p+k)} \) such that

\[
\text{gap}(W[\ldots n], k, p) = \text{gap}(W[\ldots (n-1)], k, p) + w_n - \varepsilon_n.
\]

**Induction Step.** We extend this result to weight-sequences of length \( n+1 \). Let \( W \) be an \( n+1 \)-length weight-sequence. Let \( W = W[\ldots n] \cdot w_n \) where \( w_n \in \mathbb{Z} \) such that \( |w_n| < \mu \).

First, we show that \( \text{gap}(W, d) - \text{gap}(W, k, p) \geq 0 \):

\[
\text{gap}(W, d) - \text{gap}(W, k, p) = d \cdot \text{gap}(W[\ldots n], d) + w_n - \text{roundLow}(d \cdot \text{gap}(W[\ldots n], k, p) + w_n, k, p)
\]

From the I.H. we get

\[
\geq d \cdot \text{gap}(W[\ldots n], d) + w_n - \text{roundLow}(d \cdot \text{gap}(W[\ldots n], d) + w_n, k, p)
\]
Since \( \text{gap}(a, d) - \text{gapLow}(a, k, p) \geq 0 \), we obtain the desired result that \( \text{gap}(W, d) - \text{gapLow}(W, k, p) \geq 0 \).

Next, we show that \( \text{gap}(W, d) - \text{gapLow}(W, k, p) < \text{gap}(R, d) \), where \( R \) is the \(|W|\)-length resolution sequence.

\[
\text{gap}(W, d) - \text{gapLow}(W, k, p) \\
= d \cdot \text{gap}(W[\ldots, n], d) + w_n \\
- \text{round}(d \cdot \text{gapLow}(W[\ldots, n], k, p) + w_n, k, p)
\]

Since \( \text{gap}(a, d) - \text{gapLow}(a, k, p) < 2^{-(p+k)} \), we get
\[
< d \cdot \text{gap}(W[\ldots, n], d) + w_n \\
- (d \cdot \text{gapLow}(W[\ldots, n], k, p) + w_n) + 2^{-(p+k)}
\]

\[
= d \cdot \text{gap}(W[\ldots, n], d) - d \cdot \text{gapLow}(W[\ldots, n], k, p) \\
+ 2^{-(p+k)}
\]

From the I.H. we obtain
\[
< d \cdot \text{gap}(R', d) + 2^{-(p+k)}
\]

where \( R' \) is the \( n \)-length resolution sequence

\[
= \text{gap}(R, d) \text{ where } R \text{ is the } (n+1) \text{-length resolution sequence}
\]

This concludes our proof. \( \square \)

A.2 Proof of Theorem 1

**Theorem 1.** Let \( d = 1 + 2^{-k} \) be the discount factor and \( \varepsilon = 2^{-p} \) be the approximation factor, for positive rational parameters \( p, k > 0 \). Let \( W \) be an infinite-weight length sequence. Then,

\[
0 \leq DS(W, d) - DS\text{Low}(W, k, p) < d \cdot \varepsilon
\]

**Proof.** Let \( R_n \) denote the \( n \)-length resolution sequence, and \( R \) be infinite-length resolution sequence. From Lemma 2, we know that for all \( n > 0 \),

\[
0 \leq \text{gap}(W[n], d) - \text{gapLow}(W[n], k, p) \\
< \text{gap}(R_n, d)
\]

\[
\Leftrightarrow 0 \leq \frac{(\text{gap}(W[n], d) - \text{gapLow}(W[n], k, p))}{d^{n-1}} \\
< \frac{\text{gap}(R_n, d)}{d^{n-1}}
\]

\[
\Leftrightarrow 0 \leq \frac{(\text{gap}(W[n], d) - \text{gapLow}(W[n], k, p))}{d^{n-1}} \\
< DS(R, d)
\]

By taking the limit and by further simplification, we get

\[
0 \leq DS(W, d) - DS\text{Low}(W, k, p) < d \cdot \varepsilon
\]

\( \square \)

A.3 Comparator Automata Construction

**Theorem 2.** Let \( \mu > 0 \) be and integer upper bound. Let \( k, p > 0 \) be integer parameters s.t. \( d = 1 + 2^{-k} \) is the discount factor and \( \varepsilon = 2^{-p} \) is the approximation parameter. Then, the comparator automata for lower approximation of discounted sum with discount factor \( d = 1 + 2^{-k} \), approximation factor \( \varepsilon = 2^{-p} \), upper bound \( \mu \), threshold 0 and inequality relation \( R \in \{\leq, \geq\} \) is \( \omega \)-regular.

**Proof.** The proof presents the construction of a co-safety automaton for the said comparator, thus proving the comparator is \( \omega \)-regular. Recall, the parameters are integer upper bound \( \mu > 0 \), discount factor \( d = 1 + 2^{-k} \), and approximation factor \( \varepsilon = 2^{-p} \). We are left with the main proof that \( A_{\mu,\varepsilon} \) accepts an infinite-weight sequence \( W \) iff \( DS\text{Low}(W, k, p) \geq 0 \). For this, we explain the key ideas behind the construction. A state \( s \) is interpreted to have a lower gap value of \( s \cdot 2^{-(p+k)} \). Since the automaton is deterministic, every weight sequence, finite-or-infinite length, has a unique run in the automaton. So, the idea is to ensure that for any finite-length weight sequence \( A \) if state \( s \) is the final state in its run in the automaton, then (a) if \( s \) is \( T_u \), \( \text{gapLow}(A, k, p) \geq T_u \cdot 2^{-(p+k)} \), (b) if \( s \) is \( T_l \), \( \text{gapLow}(A, k, p) \leq T_u \cdot 2^{-(p+k)} \), and (c) \( \text{gap}(A, k, p) = s \cdot 2^{-(p+k)} \) otherwise.

In summary, the critical observation here is that Item 3a ensures that the transition function follows the inductive definition of lower gap from Definition 2. This uses a proof by induction on the length of weight sequence \( A \). If \( |A| = 0 \), the final state of its run is the initial state 0, i.e., \( \text{gapLow}(A, k, p) = 0 \). Suppose the hypothesis holds for weight-sequences of length \( n \), we prove it holds for weight sequences of length \( n + 1 \). Let \( A = B \cdot b \) and \( A \) be of length \( n + 1 \). Then, suppose the final state in the run of \( B \) is \( s \). Suppose, \( b \in S \setminus \{T_u, T_l\} \). Then, I.H., \( \text{gap}(B, k, p) = s \cdot 2^{-(p+k)} \). Let the automaton transition to state \( t \) on reading alphabet \( b \) from state \( s \). Then, from definition of lower gap value, \( \text{gapLow}(A, k, p) = \text{round}(d \cdot \text{gapLow}(B, k, p) + b, k, p) \). In other words, \( \text{gap}(A, k, p) = \text{round}(d \cdot \text{gapLow}(B, k, p) + b, k, p) \). This is precisely the criteria used in the transition function to determine the state \( t \) in Eq. 3. Thus, suppose \( \text{gapLow}(A, k, p) = i \cdot 2^{-(p+k)} \), then (a) if \( T_l < i < T_u \), then \( t = i \) and \( \text{gap}(A, k, p) = t \cdot 2^{-(p+k)} \), (b) if \( i > T_u \) then \( t = T_u \) and \( \text{gap}(A, k, p) = i \cdot 2^{-(p+k)} > T_u \cdot 2^{-(p+k)} \), and (c) if \( i < T_l \) then \( t = T_l \) and \( \text{gap}(A, k, p) = i \cdot 2^{-(p+k)} < T_l \cdot 2^{-(p+k)} \). For the state \( T_u \), one can prove
that if $\text{gapLow}(A, k, p) \geq T_u \cdot 2^{-(p+k)}$ then for all $a \in \Sigma$, $\text{gapLow}(A \cdot a, k, p) \geq T_u \cdot 2^{-(p+k)}$. Conversely, for the state $T_t$, one can prove that if $\text{gapLow}(A, k, p) \leq T_t \cdot 2^{-(p+k)}$ then for all $a \in \Sigma$, $\text{gapLow}(A \cdot a, k, p) \leq T_t \cdot 2^{-(p+k)}$. This completes the proof of the claim.

Finally, to prove correctness it is sufficient to show that for all sequences $W$, $\text{DSLow}(W, k, p) \geq 0$ iff there exists a finite prefix $A$ of $W$ such that $\text{gapLow}(A, k, p) \geq T_u \cdot 2^{-(p+k)}$. This is why state $T_u$ is an accepting sink state.

\section{Case Study I: Grid World}

The human-robot interaction from is based off a classic $n \times n$ grid world domain. The human and robot correspond to the environment and system player. Initially, the two agents are present at diagonally opposite corners of the grid. Two bananas have been placed on the grid, one at each of the remaining corners. There are static obstacles of different configurations on these grids, e.g., placements of aisles (Fig 1) and an obstacle block in the center. Each agent controls its own location and is allowed to move in the cardinal directions only. The agents take turns to change their location. We assume the human makes the first move. We say a collision occurs between the robot and an object/agent if the robot is in the same location as the object/agent. In this case, a strategy for the robot tells in which location to move to next based on the history of previous configurations.

The robot’s hard (qualitative) constraint is to reach the location of at least one of the bananas without colliding into the static obstacles or the (moving) human. Thus, this constraint combines safety and reachability goals. It can be expressed as an LTL formula using atomic propositions reach_banana, collision_obstacle, and collision_human. Proposition reach_banana holds on those configurations of the grid in which the robot reaches the location of the banana. Proposition collision_obstacle holds on those configurations in which the robot collides with the wall. Similarly, proposition collision_human holds on those configurations in which the robot collides with the human. Then, the LTL formula $\varphi$ is

$$\varphi := G(\neg\text{collision_obstacle}) \land G(\neg\text{collision_human}) \land F(\text{reach_banana}).$$

The robot’s soft constraints are modelled to achieve two behaviors. The first one is to distance itself from the human. This could alternately be represented using temporal logic, however the representation will be cumbersome. Quantitative rewards can easily express this behavior. Given a negative integer parameter negative_reward, a negative reward is assigned to the robot if it comes too close to the human. This is modelled using the Manhattan distance between the two agents. Suppose, the locations of the agents are $(x_0, y_0)$ and $(x_1, y_1)$, then given negative_reward $< 0$, the reward received by the robot is

$$\text{negative_reward} = |x_0 - x_1| + |y_0 - y_1|.$$

The second behavior expressed by soft constraints is to encode promptness to fulfil reach_banana. Temporal logics are good at specifying what should be done (using the $F$ operator) but, to the best of our knowledge, they cannot nicely specify measures such as promptness. One could attempt using several X (Next operator) but that puts a hard bound on the number of steps within which the constraint must be satisfied. With quantitative constraints, one can encode promptness more naturally and softly (giving the robot more flexibility in deciding when to accomplish the constraint). In our case, we model promptness with a positive integer parameter positive_reward $> 0$ which the robot receives only when it reaches a location of the banana for the first time. This is necessary since otherwise the robot’s strategy could be to remain at the location of a banana, thus flouting the consideration to distance itself from the human.

These two rewards are additive, i.e., if both the positive and negative rewards are non-zero in a configuration of the grid world, the robot receives the sum of both rewards in that configuration. Then, it is reasonable to say that a play accomplishes these two behaviors if the total discounted-sum reward of the robot is greater than or equal to 0, i.e., 0-satisficing plays/strategies are good for the robot. Observe that if the discount factor were an integer, then robot would be prompted to pick up the banana too soon. Then in Fig 1 the robot would pick up the closer banana and would be unable to maintain sufficient distance from the human. With fractional discount factors, the robot recognizes it can plan for a longer term and will opt to reach the farther banana. This will also ensure it maintains distance form the human. This is exactly why fractional discount factors are preferred: they allow for planning on a longer term than what conservative integer factors would permit.

The output of the algorithm is either a strategy for the robot which satisfies the LTL formula $\varphi$ and is 0-satisficing for the robot in this scenario. The input to the algorithm will be a quantitative game $(G, \varphi, 0)$ where $G$ is a quantitative graph which formalizes the grid world, assigns its configurations (states) labels from the atomic propositions reach_banana, collision_obstacle, collision_human, and costs to transitions based on assignments from negative_reward and positive_reward as described above.

The output of the algorithm is either a strategy for the robot which satisfies the LTL formula $\varphi$ and is 0-satisficing for the robot in the grid, or it is a strategy for the environment which either satisfies $\neg\varphi$ or is $d \cdot \varepsilon$-satisficing for the environment where $d$ and $\varepsilon$ are the discount factor and approximation factor, respectively.

\section{Empirical Analysis}

In the experiments on grid world, we take $n = 4, 6, 8, 10$. We choose values of positive and negative rewards (positive_reward, negative_reward) from the set $\{(5, -1), (10, -1), (10, -2), (20, -2), (20, -5)\}$, creating 20 grid world benchmarks.

\section{Observations and Inferences}

Our experiments demonstrate that our algorithm facilitates the design of provably correct strategies for the robot with respect to given the soft and hard constraints. This way we are able to soundly generate a strategy for the robot, from high-level specifications, which not only satisfies a temporal objective but also take into softer consideration social-distancing and promptness.
No other known approach is able to accomplish this task soundly.

Our algorithm solves all *all but one benchmark* within the timeout. The benchmark our algorithm failed on the largest grid of size $10 \times 10$ when $d = 1.125$ ($k = 3$), positive\_reward $= 20$, and negative\_reward $= -2$. The scalability trends of our algorithm on the grid world with a $2 \times 2$ obstacle in the center of the grid on the $10 \times 10$ grid with negative\_reward $= -2$ have been summarized in Fig ??.

The runtime trends with other grid sizes and negative values are similar. This shows that the performance of the algorithm is faithful to the size of the parity game which, in turn, is linear in the size of the comparator automata (Theorem 4).

A thorough analysis of our experiments reveals avenues for improvement of the scalability of our algorithm. The one benchmark for which our algorithm failed to terminate within the timeout, we observed that the number of states in the product was high, the positive reward was high, and the discount factor was low ($10 \times 10$ grid with $d = 1.125$, positive\_reward $= 20$, negative\_reward $= -2$). Each one of these parameters contributes significantly to increasing the size of the comparator (Theorem 2) and subsequently the parity game (Theorem 4). In this case, we observed that the algorithm ran out of memory on our machine. This suggests to focus on succinct representations of the comparator and the game in future work.

Another observation has to do with the percentage of time spent in each step of the algorithm. Currently, our algorithm implements an explicit construction of the parity game. We observed that on most benchmarks, the algorithm spent around 70-80% of its time constructing the game in future work.

The runtime trends with other grid sizes and negative values are similar. This shows that the performance of the algorithm is faithful to the size of the comparator automata (Theorem 4).

In our second case study, we consider a significantly more challenging set of scenarios. A robot must operate along a $r \times c$ conveyor belt with $r$ rows and $c$ columns across from a human, see Fig 6. Both agents are restricted to not reach fully across the conveyor belt. When out of reach of the human, the robot can move quickly. Otherwise, it must proceed more slowly. The blocks move down the conveyor belt at a constant speed.

The human controls the location of its arm and the placement of new objects. New blocks of identical type (color) are placed whenever a block is removed from the belt so that a constant number and proportion of types of blocks are maintained on the belt. The human controls the placement of new blocks, except that it must place green blocks near the robot (to ensure the game is winnable).

2 blocks. In the two block scenario, the robot’s LTL goal is to ensure it doesn’t interfere with the human grasping objects. We define proposition collision as in the previous example. Proposition block\_human holds in a state if the robot and human are adjacent to the human’s object and the human

![Image](363x601 to 514x738)

Figure 6: Example conveyor belt scenario with three blocks.

simultaneously. Then, the robot’s LTL goal in the 2-block scenario is

$$
\varphi_2 := G(\neg\text{collision}) \land G(\neg\text{block\_human})
$$

The robot’s soft constraint is designed to encourage it to pick up as many blocks as possible. The robot receives a positive reward for every block it picks up and a negative reward for every block that falls off the belt.

3 blocks. In the three block scenario (Fig 6), the green blocks are “critical” and the robot must grab one. The blue blocks are “desired” and the robot should retrieve as many of them as possible. The red blocks are “the human’s” and the robot should ensure it never blocks the human from reaching them. The robot’s LTL goal is to ensure it grasps all green objects and avoids the human grasping red objects. We define Propositions collision and block\_human as in the previous example. Proposition dropped\_critical holds if a critical object has been dropped prior to or in the current state.

$$
\varphi_3 := G(\neg\text{collision}) \land G(\neg\text{block\_human}) \land G(\neg\text{dropped\_critical})
$$

The robot’s soft constraint is to maximize the number of blue objects it grasps. Each arm is modeled as grid cells emanating from either side of the conveyor belt. The robot controls the location of its arm. Every desired object retrieved gives positive reward. Every desired object that falls off the end of the belt gives negative reward. If both positive and negative reward are achieved in the same step, the rewards are added.

C.1 Empirical Evaluation

In the experiments on conveyor belt, $r \times c = 4 \times 3, 5 \times 3$ with 2 or 3 blocks. We choose negative\_reward $= -1$. With 2 blocks, we choose positive\_reward $= 2, 3, 4$ and with 1 block positive\_reward $= 5$, creating 7 conveyor belt benchmarks.

Observations and Inferences For the two block scenario, our algorithm solves all but one benchmark. The failure here is a $5 \times 3$ conveyor belt when the positive reward is 4 and the discount factor is 1.125 ($k = 3$). As earlier, the runtime
trends are consistent with the theoretical analysis on size of
the parity game and the comparator.

In the solved cases, we see that the algorithm generates
a strategy for the robot in all games (which we engineer so
that the robot can win). We note that the robot quickly ob-
tains its rewards, suggesting its policy is of high-quality. Un-
fortunately, the complexity of the game makes it intractable
to hand-compute an optimal policy and compare it to the
robot’s policy generated by the algorithm. The inability to
manually or algorithmically check the correctness of a pol-
cy w.r.t. optimality is a reason why one would want sound
algorithms like ours to solve complex scenarios like this.

On the three block scenario, we performed experiments
on the $5 \times 3$ conveyor belt. Our algorithm terminates on the
belts when the discount factor is $d = 1.5, 1.25 (k = 1, k = 2)$ but it struggled with discount factor $d = 1.125(k = 3)$. As a representative case. Further, none of our experiments
terminated at $d = 1.125$. This is not surprising since the
product game is large (60K states) and the discount factor
is low. Again, we see that future work will require improved
scalability. This will open up new applications for robotic

<table>
<thead>
<tr>
<th>Rewards</th>
<th>Discount factor</th>
<th>Total time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>Negative</td>
<td>1.25</td>
</tr>
<tr>
<td>Grid World $n = 4$ with 397 states</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-1</td>
<td>0.015</td>
</tr>
<tr>
<td>10</td>
<td>-1</td>
<td>0.012</td>
</tr>
<tr>
<td>10</td>
<td>-2</td>
<td>2.084</td>
</tr>
<tr>
<td>20</td>
<td>-2</td>
<td>11.770</td>
</tr>
<tr>
<td>20</td>
<td>-5</td>
<td>24.503</td>
</tr>
</tbody>
</table>

| Grid World $n = 6$ with 2407 states |
| 5 | -1 | 0.050 | 0.158 |
| 10 | -1 | 0.050 | 0.150 |
| 10 | -2 | 6.856 | 46.199 |
| 20 | -2 | 13.987 | 94.525 |
| 20 | -5 | 19.424 | 106.136 |

| Grid World $n = 8$ with 8093 states |
| 5 | -1 | 0.159 | 0.444 |
| 10 | -1 | 0.158 | 0.419 |
| 10 | -2 | 19.932 | 138.201 |
| 20 | -2 | 38.293 | 279.519 |
| 20 | -5 | 56.544 | 330.451 |

| Grid World $n = 10$ with 20572 states |
| 5 | -1 | 0.416 | 0.972 |
| 10 | -1 | 0.413 | 0.914 |
| 10 | -2 | 39.102 | 315.604 |
| 20 | -2 | 78.329 | Timeout |
| 20 | -5 | 122.792 | Timeout |

Table 2: Analysis of Grid World Domain. Table does not
record $d = 1.5$ to improve readability of table. All runs with
$d = 1.5$ terminated within less than 1 sec. Timeout = 750 sec
### Table 3: Analysis of Conveyor Belt domain. Table does not record $d = 1.5$ to improve readability of table. All runs with $d = 1.5$ terminated within less than 10 sec. Timeout = 750 sec.

<table>
<thead>
<tr>
<th>Rewards</th>
<th>Positive</th>
<th>Negative</th>
<th>Discount factor</th>
<th>Total time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Conveyor Belt $r \times c = 4 \times 3$ with 2 blocks (9966 states)</strong></td>
<td>2</td>
<td>-1</td>
<td>1.25</td>
<td>20.121</td>
</tr>
<tr>
<td>1.125</td>
<td>102.815</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td>1.25</td>
<td>1.125</td>
<td>29.922</td>
</tr>
<tr>
<td>1.125</td>
<td>152.274</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-1</td>
<td>1.25</td>
<td>1.125</td>
<td>40.216</td>
</tr>
<tr>
<td>1.125</td>
<td>208.748</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Conveyor Belt $r \times c = 5 \times 3$ with 2 blocks (31547 states)</strong></td>
<td>2</td>
<td>-1</td>
<td>1.25</td>
<td>64.782</td>
</tr>
<tr>
<td>1.125</td>
<td>332.764</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td>1.25</td>
<td>1.125</td>
<td>98.520</td>
</tr>
<tr>
<td>1.125</td>
<td>677.558</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-1</td>
<td>1.25</td>
<td>1.125</td>
<td>127.422</td>
</tr>
<tr>
<td>1.125</td>
<td>Timeout</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Conveyor Belt $r \times c = 5 \times 3$ with 3 blocks (60540 states)</strong></td>
<td>5</td>
<td>-1</td>
<td>1.25</td>
<td>712.941</td>
</tr>
<tr>
<td>1.125</td>
<td>Timeout</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 4: 10x10 social dist with varied approximation factor

<table>
<thead>
<tr>
<th>Rewards</th>
<th>Positive</th>
<th>Negative</th>
<th>Discount factor</th>
<th>Approx factor</th>
<th>Total time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>-1</td>
<td>1.5</td>
<td>1.25</td>
<td>0.385</td>
<td></td>
</tr>
<tr>
<td>1.125</td>
<td>0.394</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.25</td>
<td>0.415</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.125</td>
<td>0.421</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.125</td>
<td>0.915</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.125</td>
<td>0.917</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-2</td>
<td>1.5</td>
<td>1.25</td>
<td>1.642</td>
<td></td>
</tr>
<tr>
<td>1.125</td>
<td>6.407</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.25</td>
<td>39.018</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.125</td>
<td>82.824</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.125</td>
<td>319.686</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.125</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>-1</td>
<td>1.5</td>
<td>1.25</td>
<td>0.383</td>
<td></td>
</tr>
<tr>
<td>1.125</td>
<td>0.443</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.25</td>
<td>0.418</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.125</td>
<td>0.428</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.125</td>
<td>0.908</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.125</td>
<td>0.917</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>-2</td>
<td>1.5</td>
<td>1.25</td>
<td>4.001</td>
<td></td>
</tr>
<tr>
<td>1.125</td>
<td>11.082</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.25</td>
<td>80.069</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.125</td>
<td>160.947</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.125</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.125</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>