Acta Informatica manuscript No.
(will be inserted by the editor)

Reactive Synthesis with Maximum Realizability of Linear Temporal Logic Specifications

Rayna Dimitrova* · **Mahsa Ghasemi*** · **Ufuk Topcu**

Abstract A challenging problem for autonomous systems is to synthesize a reactive controller that conforms to a set of given correctness properties. Linear temporal logic (LTL) provides a formal language to specify the desired behavioral properties of systems. In applications in which the specifications originate from various aspects of the system design, or consist of a large set of formulas, the overall system specification may be unrealizable. Driven by this fact, we develop an optimization variant of synthesis from LTL formulas, where the goal is to design a controller that satisfies a set of hard specifications and minimally violates a set of soft specifications. To that end, we introduce a value function that, by exploiting the LTL semantics, quantifies the level of violation of properties. Inspired by the idea of bounded synthesis, we fix a bound on the implementation size and search for an implementation that is optimal with respect to the said value function. We propose a novel maximum satisfiability encoding of the search for an optimal implementation (within the given bound on the implementation size). We iteratively increase the bound on the implementation size until a termination criterion, such as a threshold over the value function, is met.

Keywords Maximum realizability · Linear temporal logic · Bounded synthesis · Maximum satisfiability

Part of the results in this paper were presented at the Sixteenth International Symposium on Automated Technology for Verification and Analysis, Los Angeles, California, USA, October 2018 [1].

∗These authors contributed equally to the manuscript.

Rayna Dimitrova
University of Leicester, Leicester, UK
E-mail: rd307@leicester.ac.uk

Mahsa Ghasemi
University of Texas at Austin, Austin, Texas, USA
E-mail: mahsa.ghasemi@utexas.edu

Ufuk Topcu
University of Texas at Austin, Austin, Texas, USA
E-mail: utopcu@utexas.edu

1 Introduction

In an ideal world, a user may specify a set of high-level behavioral characteristics for an autonomous system, and a controller (i.e., implementation) can be synthesized to comply with these specifications. Such automatic synthesis has been a topic for various studies in the domain of formal methods where the goal is to design a hardware or a software system that satisfies a set of formally defined properties. These properties can be formulated in an appropriate language, such as linear temporal logic (LTL) [2]. In conventional synthesis, either an implementation is constructed for a given specification, or the specification is identified as unrealizable. Nevertheless, especially in large systems, specifications may arise from different design perspectives, and if they consist of a large number of individual requirements, it is easy to encounter specifications that are unrealizable. In other scenarios, the user may have several alternative requirements in mind, potentially with some preferences, and want to know the best realizable combination of them with respect to some metric. Such cases usually lead to alternating between specification modification and synthesis procedure and hence, defeating the purpose of facilitating the design process.

The possibility of conflict amongst the provided requirements calls for a more comprehensive synthesis procedure that, in the case of unrealizability, can generate an implementation that minimally violates the specifications. In order to define the notion of minimality, one requires a quantitative metric on the satisfaction of LTL formulas. The approach we pursue relies on multiple levels of relaxations of an LTL formula. We associate each level with a binary variable and form a value function that indicates the levels of relaxations of the formula that the implementation satisfies. The value function respects a lexicographic ordering according to preferences over relaxations. Having defined a value function, one can interpret maximum realizability of a set of LTL formulas as seeking an implementation that maximizes the corresponding value.

In this paper, we consider settings in which the goal is to design a system that satisfies a given hard specification and maximizes the defined value function over a set of (potentially prioritized) soft specifications. We first focus on soft specifications that are safety formulas and later discuss the extension to general formulas. We quantify the compliance with a safety property according to the LTL semantics in a straightforward manner. The highest value is associated with satisfying the formula at all times, and it monotonically decreases if the formula is satisfied from some point on, then satisfied infinitely often, and lastly satisfied only for a finite number of times. Based on this ordering, we define the cumulative value of a conjunction of safety properties according to given priorities or design criteria.

The backbone of our approach toward maximum realizability is bounded synthesis, originally introduced by Schewe and Finkbeiner [3]. Bounded synthesis tackles the computational complexity of reactive synthesis from LTL properties by restricting the size of the search space. More specifically, it searches for a realizable implementation of the size up to a prespecified bound. If no such implementation exists, it increments the bound and repeats the search process. Each instance of bounded search for an implementation can be encoded as a SAT (or QBF, or SMT) problem [4]. The algorithm is complete as a theoretical bound on the maximum size of the implementation exists.

We formulate maximum realizability as iterative MaxSAT solving [5]. In each iteration, we construct a MaxSAT instance that characterizes the existence of an implementation of size within the given bound that not only realizes the hard specification but is also optimal with respect to defined value function for soft specifications. We prove that, for any given finite set of soft specifications, there exists an optimal implementation with a bounded size. Consequently, the proposed algorithm that gradually increases the bound on the implemen-

tation size is complete. On the other hand, the theoretical upper bound on the minimal implementation size is generally impractically large. Therefore, in practice, we also settle for termination criteria such as a problem-specific bound on controller size, a limit on running time, or a desired threshold on the value function.

The proposed encoding of maximum realizability generates partial weighted MaxSAT instances. A partial weighted MaxSAT problem is composed of a set of hard clauses and a set of weighted soft clauses. The hard clauses capture the encodings imposed by bounded synthesis procedure for both hard and soft specifications. The soft clauses determine the level of relaxation of a soft specification that can be satisfied. We design the weights of soft clauses in a way that they correspond to the quantitative semantics of soft specifications. Therefore, adjusting the weights allows our approach to easily adapt to different design criteria.

Recent advances in SAT solving along with the development of novel algorithms such as structural partitioning and search heuristics have made MaxSAT solviers a promising tool. MaxSAT formulations have been effective in solving many real-world problems including most probable explanation in Bayesian networks [6], package management [7], and correlation clustering and causal structure learning [8]. While SAT solving has been an essential part of numerous formulations proposed for reactive synthesis problems, the applicability of MaxSAT solving has not yet been explored in this field. In this paper, we develop the first maximum realizability algorithm that utilizes the power of MaxSAT solvers to deal with the underlying combinatorial nature of the optimization task.

We evaluated the proposed maximum realizability procedure experimentally on reactive synthesis instances from two domains where considering combinations of hard and soft specifications is natural and often unavoidable. The first domain is robotic navigation, where due to the adversarial nature of the environment in which robots operate, safety requirements might prevent a system from achieving its goal, or a large number of tasks of different types might not necessarily be consistent when posed together. The second domain relates to load distribution tasks in power networks. There, generators have limited capacity to power a set of vital and non-vital loads, whose total demand may exceed the capacity of the generators, thus leading to a combination of hard and soft specifications.

The rest of the manuscript is organized as follows. Section 2 discusses related work. Section 3 recalls the necessary background on LTL synthesis, the bounded synthesis approach, and maximum satisfiability. In Section 4, we describe the proposed quantitative semantics for soft specifications and using this semantics, formally state the maximum realizability problem. In Section 5, we detail the proposed bounded maximum realizability algorithm along its encoding into MaxSAT instances. Section 6 presents the experimental settings and the obtained results. Lastly, Section 7 states the concluding remarks and future directions.

This paper is an extension of the conference publication [1]. It contains the complete proofs and presents (Section 5.4) a generalization of the maximum realizability problem to soft specifications in the full class of LTL formulas and to prioritized specifications.

2 Related Work

Maximum realizability and several closely related problems have attracted significant attention in recent years. Tumova et al. [9] studied the problem of planning over a finite horizon with prioritized safety requirements, where the goal is to synthesize a least-violating control strategy. Kim et al. [10] studied a similar problem for the case of infinite-horizon temporal logic planning, which seeks to revise an inconsistent specification, minimizing the cost of revision with respect to costs for atomic propositions provided by the specifier. Lahijanian et

al. [11] describe a method for computing optimal plans for co-safe LTL specifications, where optimality is again with respect to the cost of violating each atomic proposition, which is provided by the user. All of these approaches are developed for the planning setting, where there is no adversarial environment, and thus they are able to reduce the problem to the computation of an optimal path in a graph. Lahijanian and Kwiatkowska [12] considered the case of probabilistic environments. In contrast, the proposed method seeks to maximize the satisfaction of the given specification against the worst-case behavior of the environment. Lahijanian et al. [13] studied the problem of partial satisfaction of guarantees in an unknown environment, where, unlike in our work, no relaxations of the soft specifications are considered, but simply the number of those that are satisfied is maximized.

The problem setting that is the closest to ours is that of Tomita et al. [14]. There, the authors study a maximum realizability problem in which the specification is a conjunction of a *must* (or *hard*, in our terms) LTL specification, and a number of weighted *desirable* (or *soft*, in our terms) specifications of the form $\square\varphi$, where φ is an arbitrary LTL formula. When φ is not a safety property it is first strengthened to a safety formula before applying the synthesis procedure, which then weakens the result to a mean-payoff term. Thus, while Tomita et al. consider a broader class of soft specifications compared to those in this paper, when φ is not a safety property there is no clear relationship between $\square\varphi$ and the resulting mean-payoff term. When applied to multiple soft specifications, the method by Tomita et al. combines the corresponding mean-payoff terms in a weighted sum, and synthesizes an implementation optimizing the value of this sum. Thus, without inspecting the synthesized implementation it is not possible to determine to what extent the individual desirable specifications are satisfied. In contrast, in the proposed maximum realizability procedure each satisfaction value is characterized as an LTL formula, which is useful for explainability and providing feedback to the designer.

To the best of our knowledge, our work is the first to employ MaxSAT in the context of reactive synthesis. MaxSAT has been used for preference-based planning [15] and for computing optimal plans in propositional planning problems with action costs [16]. However, since maximum realizability is concerned with reactive systems, it requires a fundamentally different approach from planning.

Two other main research directions related to maximum realizability are *quantitative synthesis* and *specification debugging*. There are two predominant flavours of quantitative synthesis problems studied in the literature. In the first one (cf. [17]), the goal is to generate an implementation that maximizes the value of a mean-payoff objective, while possibly satisfying some ω-regular specification. In the second setting (cf. [18,19]), the system requirements are formalized in a multi-valued temporal logic. These synthesis methods [19, 17,18], however, do not directly solve the corresponding optimization problem, but instead check for the existence of an implementation whose value is in a given set. The optimization problem can then be reduced to a sequence of such queries.

Alur et al. [20] studied an optimal synthesis problem for an ordered sequence of prioritized ω-regular properties, where the classical fixpoint-based game-solving algorithms are extended to a quantitative setting. The main difference in our work is that we allow for incomparable soft specifications each with a number of prioritized relaxations, for which the equivalent set of preference-ordered combinations would be of size exponential in the number of soft specifications. Our MaxSAT formulation avoids explicitly considering these combinations.

In specification debugging there is a lot of research dedicated to finding good explanations for the unsatisfiability or unrealizability of temporal logic specifications [21–23], and more generally to the analysis of specifications [24,25]. Our approach to maximum real-

izability can prove useful for specification analysis, since instead of simply providing an optimal value, it computes an optimal relaxation of the given specification in the form of another LTL formula.

3 Background

We start by an overview of syntax and semantics of linear temporal logic (LTL) and language-equivalent automata representation. Next, we define finite-state transition systems, and formally state the synthesis problem. Then, we proceed to go over definitions of run graph and annotations to describe the bounded synthesis method and its SAT encoding. Lastly, we provide a brief description of maximum satisfiability (MaxSAT) problem, particularly a class of that called partial weighted MaxSAT.

3.1 Synthesis from LTL Specifications

Linear temporal logic (LTL) is a formal language for specifying behavioral characteristics of reactive systems. Formulas in LTL are constructed according to the following grammar:

$$\varphi := p \mid \mathit{true} \mid \mathit{false} \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \,\mathcal{U}\, \varphi_2 \mid \varphi_1 \,\mathcal{R}\, \varphi_2,$$

where $p \in \mathcal{P}$ is an atomic proposition. *next* ($\bigcirc$), *until* ($\mathcal{U}$), and *release* ($\mathcal{R}$) are temporal operators. The *finally* operator ($\Diamond$) is defined as $\Diamond\varphi \equiv \mathit{true}\,\mathcal{U}\,\varphi$ and the *globally* operator ($\Box$) is defined as $\Box\varphi \equiv \mathit{false}\,\mathcal{R}\,\varphi$. We denote the size of an LTL formula (that is, the number of operators in the formula) with $|\varphi|$, and the set of all its subformulas with $\mathsf{subf}(\varphi)$. A negation normal form (NNF) of an LTL formula is a semantically-equivalent LTL formula in which negations appear only in front of atomic propositions. Without loss of generality, we consider LTL formulas in NNF.

Let $\Sigma = 2^{\mathcal{P}}$ denote the finite alphabet composed of all possible valuations of the propositions. A letter $\sigma \in \Sigma$ is interpreted as the valuation that assigns value true to all $p \in \sigma$ and false to all $p \in \mathcal{P} \setminus \sigma$. An infinite word $w \in \Sigma^\omega$ is an infinite sequence of letters. An LTL formula φ defines a language over infinite words. A word is included in the language if it satisfies the formula, denoted by $w \models \varphi$. The full semantics of LTL can be found in [26].

A *safety LTL formula* is an LTL formula such that every word not in its language has a bad prefix. Formally, φ is a safety LTL formula if for each $w \not\models \varphi, w \in \Sigma^\omega$, there exists a bad prefix $u \in \Sigma^*$ such that $u \cdot v \not\models \varphi, \forall v \in \Sigma^\omega$. *Syntactically safe* LTL formulas are a subclass of safety LTL formulas which do not contain any occurrences of $\mathcal{U}$ when written in NNF.

The language accepted by an LTL formula can equivalently be represented by a nondeterministic (or universal) Büchi (or co-Büchi) automaton. A *Büchi automaton* over a finite alphabet Σ is a tuple $\mathcal{A} = (Q, q_0, \delta, F)$, where Q is a finite set of states, q_0 is the initial state, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation, and $F \subseteq Q$ is a subset of states. A run of $\mathcal{A}$ on an infinite word $w = \sigma_0\sigma_1 \ldots \in \Sigma^\omega$ is an infinite sequence $q_0, q_1, \ldots$ of states, where q_0 is the initial state and for every $i \geq 0$ it holds that $(q_i, \sigma_i, q_{i+1}) \in \delta$.

A run of a Büchi automaton is accepting if it contains infinitely many occurrences of states in F. A *co-Büchi automaton* $\mathcal{A} = (Q, q_0, \delta, F)$ differs from a Büchi automaton in the accepting condition: a run of a co-Büchi automaton is accepting if it contains only *finitely many* occurrences of states in F. For a Büchi automaton the states in F are called *accepting states*, while for a co-Büchi automaton they are called *rejecting states*. A *nondeterministic*

automaton $\mathcal{A}$ accepts a word $w \in \Sigma^\omega$ if *some* run of $\mathcal{A}$ on w is accepting. A *universal* automaton $\mathcal{A}$ accepts a word $w \in \Sigma^\omega$ if *every* run of $\mathcal{A}$ on w is accepting.

For a reactive system, the set of atomic propositions is $\mathcal{P} = \mathcal{I} \cup \mathcal{O}$, where $\mathcal{I}$ and $\mathcal{O}$ are disjoint sets, denoting *input* propositions controlled by the environment and *output* propositions controlled by the system, respectively. A *transition system* over a set of input propositions $\mathcal{I}$ and a set of output propositions $\mathcal{O}$ is a tuple $\mathcal{T} = (S, s_0, \tau)$, where S is a set of states, s_0 is the initial state, and the transition function $\tau : S \times 2^{\mathcal{I}} \to S \times 2^{\mathcal{O}}$ maps a state s and a valuation $\sigma_I \in 2^{\mathcal{I}}$ of the input propositions to a successor state s' and a valuation $\sigma_O \in 2^{\mathcal{O}}$ of the output propositions. For any letter σ, we consider the projection to input propositions by $\sigma_I \stackrel{\text{def}}{=} \sigma \cap \mathcal{I}$ and to output propositions by $\sigma_O \stackrel{\text{def}}{=} \sigma \cap \mathcal{O}$. If the set S is finite, $\mathcal{T}$ is a finite-state transition system and its size is defined by $|\mathcal{T}| \stackrel{\text{def}}{=} |S|$.

An *execution* of $\mathcal{T}$ is an infinite sequence $s_0, (\sigma_{I0} \cup \sigma_{O0}), s_1, (\sigma_{I1} \cup \sigma_{O1}), s_2 \ldots$ such that s_0 is the initial state, and $(s_{i+1}, \sigma_{O\,i}) = \tau(s_i, \sigma_{I\,i})$ for every $i \geq 0$. The corresponding sequence $(\sigma_{I0} \cup \sigma_{O0}), (\sigma_{I1} \cup \sigma_{O1}), \ldots \in \Sigma^\omega$ is called a trace. We denote with $\mathsf{Traces}(\mathcal{T})$ the set of all traces of a transition system $\mathcal{T}$. A transition system $\mathcal{T}$ satisfies an LTL formula φ over atomic propositions $\mathcal{P} = \mathcal{I} \cup \mathcal{O}$, denoted by $\mathcal{T} \models \varphi$, if for every $w \in \mathsf{Traces}(\mathcal{T})$, it holds that $w \models \varphi$.

The decision problem of determining whether there exists a transition system that satisfies an LTL formula is called the *realizability problem for LTL*. If an LTL formula φ is realizable, the goal of *LTL synthesis problem* is to construct a transition system $\mathcal{T}$ such that $\mathcal{T} \models \varphi$.

3.2 Bounded Synthesis Approach

The *run graph* of a universal automaton $\mathcal{A} = (Q, q_0, \delta, F)$ on a transition system $\mathcal{T} = (S, s_0, \tau)$ is the unique graph $G = (V, E)$ with a set of nodes $V = S \times Q$ and a set of labeled edges $E \subseteq V \times \Sigma \times V$ such that $((s, q), \sigma, (s', q')) \in E$ iff $(q, \sigma, q') \in \delta$ and $\tau(s, \sigma \cap \mathcal{I}) = (s', \sigma \cap \mathcal{O})$. That is, G is the product of $\mathcal{A}$ and $\mathcal{T}$.

A run graph of a universal Büchi (resp. co-Büchi) automaton is accepting if every infinite path $(s_0, q_0), (s_1, q_1), \ldots$ contains infinitely (resp. finitely) many occurrences of states q_i in F. A transition system $\mathcal{T}$ is accepted by a universal automaton $\mathcal{A}$ if the unique run graph of $\mathcal{A}$ on $\mathcal{T}$ is accepting. We denote with $\mathcal{L}(\mathcal{A})$ the set of transition systems accepted by $\mathcal{A}$.

The bounded synthesis approach is based on the fact that for every LTL formula φ one can construct a universal co-Büchi automaton $\mathcal{A}_\varphi$ with at most $2^{O(|\varphi|)}$ states such that $\mathcal{T} \in \mathcal{L}(\mathcal{A}_\varphi)$ iff $\mathcal{T} \models \varphi$, for every transition system $\mathcal{T}$ [27].

An *annotation* of a transition system $\mathcal{T} = (S, s_0, \tau)$ with respect to a universal co-Büchi automaton $\mathcal{A} = (Q, q_0, \delta, F)$ is a function $\lambda : S \times Q \to \mathbb{N} \cup \{\bot\}$ that maps nodes of the run graph of $\mathcal{A}$ on $\mathcal{T}$ to the set $\mathbb{N} \cup \{\bot\}$. Intuitively, such an annotation is valid if every node (s, q) that is reachable from the node (s_0, q_0) is annotated with a natural number, which is an upper bound on the number of rejecting states visited on any path from (s_0, q_0) to (s, q). Formally, an annotation $\lambda : S \times Q \to \mathbb{N} \cup \{\bot\}$ is *valid* if

- $\lambda(s_0, q_0) \neq \bot$, i.e., the pair of initial states is labeled with a number, and
- whenever $\lambda(s, q) \neq \bot$, then for every edge $((s, q), \sigma, (s', q'))$ in the run graph of $\mathcal{A}$ on $\mathcal{T}$ we have that (s', q') is annotated with a number (i.e., $\lambda(s', q') \neq \bot$), such that $\lambda(s', q') \geq \lambda(s, q)$, and if $q' \in F$, then $\lambda(s', q') > \lambda(s, q)$.

Valid annotations of finite-state transition systems correspond to accepting run graphs. An annotation λ is c-bounded if $\lambda(s, q) \in \{0, \ldots, c\} \cup \{\bot\}$ for all $s \in S$ and $q \in Q$.

The synthesis method proposed in [3,28] employs the following result in order to reduce the bounded synthesis problem to checking the satisfiability of propositional formulas: a transition system $\mathcal{T}$ is accepted by a universal co-Büchi automaton $\mathcal{A} = (Q, q_0, \delta, F)$ iff there exists a $(|\mathcal{T}| \cdot |F|)$-bounded valid annotation for $\mathcal{T}$ and $\mathcal{A}$. One can estimate a bound on the size of the transition system, which allows to reduce the synthesis problem to its bounded version. Namely, if there exists a transition system that satisfies an LTL formula φ, then there exists a transition system satisfying φ with at most $\left(2^{(|\mathsf{subf}(\varphi)|+\log|\varphi|)}\right)!^2$ states.

Let $\mathcal{A} = (Q, q_0, \delta, F)$ be a universal co-Büchi automaton for the LTL formula φ. Given a bound b on the size of the desired transition system $\mathcal{T}$, the bounded synthesis problem can be encoded as a satisfiability problem with the following sets of propositional variables and constraints.

Variables: The variables represent the desired transition system $\mathcal{T}$, and the desired valid annotation λ of the run graph of $\mathcal{A}$ on $\mathcal{T}$. A transition system with b states $S = \{1, \ldots, b\}$ is represented by Boolean variables $\tau_{s,\sigma_I,s'}$ and o_{s,σ_I} for every $s, s' \in S$, $\sigma_I \in 2^{\mathcal{I}}$, and $o \in \mathcal{O}$. The variable $\tau_{s,\sigma_I,s'}$ encodes the existence of transition from s to s' on input σ_I, and the variable o_{s,σ_I} encodes o being true in the output from state s on input σ_I.

The annotation λ is represented by the following variables. For each $s \in S$ and $q \in Q$, there is a Boolean variable $\lambda^{\mathbb{B}}_{s,q}$ and a vector $\lambda^{\mathbb{N}}_{s,q}$ of $\log(b \cdot |F|)$ Boolean variables: the variable $\lambda^{\mathbb{B}}_{s,q}$ encodes the reachability of (s, q) from the initial node (s_0, q_0) in the corresponding run graph, and the vector of variables $\lambda^{\mathbb{N}}_{s,q}$ represents the bound for the node (s, q). The constraints are as follows [28].

Constraints for input-enabled $\mathcal{T}$: $C_\tau \stackrel{\text{def}}{=} \bigwedge_{s\in S} \bigwedge_{\sigma_I \in 2^{\mathcal{I}}} \bigvee_{s' \in S} \tau_{s,\sigma_I,s'}$.

Constraints for valid annotation:

$$C_\lambda \stackrel{\text{def}}{=} \lambda^{\mathbb{B}}_{s_0,q_0} \wedge \bigwedge_{q,q' \in Q} \bigwedge_{s,s' \in S} \bigwedge_{\sigma_I \in 2^{\mathcal{I}}} \Big(\big(\lambda^{\mathbb{B}}_{s,q} \wedge \delta_{s,q,\sigma_I,q'} \wedge \tau_{s,\sigma_I,s'}\big) \rightarrow \mathsf{succ}_\lambda(s, q, s', q') \Big),$$

where $\delta_{s,q,\sigma_I,q'}$ is a formula over the variables o_{s,σ_I} that characterizes the transitions in $\mathcal{A}$ between q and q' on labels consistent with σ_I, and $\mathsf{succ}_\lambda(s, q, s', q')$ is a formula over the annotation variables such that

$$\mathsf{succ}_\lambda(s, q, s', q') \stackrel{\text{def}}{=} \begin{cases} \lambda^{\mathbb{B}}_{s',q'} \wedge (\lambda^{\mathbb{N}}_{s',q'} > \lambda^{\mathbb{N}}_{s,q}) & \text{if } q' \in F, \\ \lambda^{\mathbb{B}}_{s',q'} \wedge (\lambda^{\mathbb{N}}_{s',q'} \geq \lambda^{\mathbb{N}}_{s,q}) & \text{if } q' \notin F. \end{cases}$$

3.3 Maximum Satisfiability

The procedure proposed by Finkbeiner and Schewe [28] and recalled in the previous section provides a SAT encoding of synthesis when the size of the implementation is bounded. Maximum realizability is an optimization variant of synthesis while MaxSAT is an optimization variant of SAT. We show that, for a proposed value function, the maximum realizability problem under a bounded implementation size can be reduced to a *partial weighted MaxSAT* instance.

Consider a propositional logic formula in conjunctive normal form (CNF), i.e., a formula that is a conjunction of disjunction of literals, where a literal is a Boolean variable or its negation and a disjunction of literals is called a clause. *MaxSAT* is the problem of assigning truth values to a set of Boolean variables such that the number of clauses of a propositional logic formula in CNF that are made true, is maximized [5]. A *partial MaxSAT* is a variant of MaxSAT problem where the clauses are categorized as hard and soft clauses. In this case, the

goal is to find a truth assignment to the variables such that all the hard clauses are made true and the number of soft clauses that become true is maximized. A more general problem is that of *partial weighted MaxSAT* where each of the soft clauses is associated with a positive numerical weight. There, the objective is to find a truth assignment to the variables that not only makes all the hard clauses true but also maximizes the sum of the weights of the soft clauses that become true.

We exploit the separation of the hard and soft clauses in partial weighted MaxSAT to capture the hard and soft constraints that arise in the encoding of the maximum realizability problem. Furthermore, we design the weights of the soft clauses in a way to promote the quantitative objective associated with the conjunction of the given soft specifications.

4 Maximum Realizability

Often, the specifications required from a system are a combination of multiple requirements, which might not be realizable in conjunction. In such a case, in addition to reporting the unrealizability to the system designer, we would like the synthesis procedure to construct an implementation that satisfies the specifications "as much as possible". Such implementation is particularly useful in the case where some of the requirements describe desirable but not necessarily essential properties of the system. To determine what "as much as possible" formally means, a quantitative semantics of the specification language is necessary. In the following, we provide such semantics for a fragment of LTL. The quantitative interpretation is based on the standard semantics of LTL formulas of the form $\square\varphi$.

4.1 Quantitative Semantics of Soft Safety Specifications

Let $\square\varphi_1, \ldots, \square\varphi_n$ be a set of LTL specifications, where each φ_i is a safety LTL formula. In order to formalize the maximal satisfaction of $\square\varphi_1 \wedge \ldots \wedge \square\varphi_n$, we first give a quantitative semantics of formulas of the form $\square\varphi$.

Quantitative semantics of safety specifications. For an LTL formula of the form $\square\varphi$ and a transition system $\mathcal{T}$, we define *the value* $val(\mathcal{T}, \square\varphi)$ *of* $\square\varphi$ *in* $\mathcal{T}$ as

$$val(\mathcal{T}, \square\varphi) \stackrel{\text{def}}{=} \begin{cases} (1,1,1) & \text{if } \mathcal{T} \models \square\varphi, \\ (1,1,0) & \text{if } \mathcal{T} \not\models \square\varphi \text{ and } \mathcal{T} \models \diamondsuit\square\varphi, \\ (1,0,0) & \text{if } \mathcal{T} \not\models \square\varphi \text{ and } \mathcal{T} \not\models \diamondsuit\square\varphi \text{ and } \mathcal{T} \models \square\diamondsuit\varphi, \\ (0,0,0) & \text{if } \mathcal{T} \not\models \square\varphi \text{ and } \mathcal{T} \not\models \diamondsuit\square\varphi \text{ and } \mathcal{T} \not\models \square\diamondsuit\varphi. \end{cases}$$

Thus, the value of $\square\varphi$ in a transition system $\mathcal{T}$ is a vector $(v_1, v_2, v_3) \in \{0,1\}^3$, where the value $(1,1,1)$ corresponds to the $true$ value in the classical semantics of LTL. When $\mathcal{T} \not\models \square\varphi$, the values $(1,1,0)$, $(1,0,0)$ and $(0,0,0)$ capture the extent to which φ holds or not along the traces of $\mathcal{T}$. For example, if $val(\mathcal{T}, \square\varphi) = (1,0,0)$, then φ holds infinitely often on each trace of $\mathcal{T}$, but there exists a trace of $\mathcal{T}$ on which φ is violated infinitely often. When $val(\mathcal{T}, \square\varphi) = (0,0,0)$, then on some trace of $\mathcal{T}$, φ holds for at most finitely many positions. Note that by the definition of val, if $val(\mathcal{T}, \square\varphi) = (v_1, v_2, v_3)$, then

- $v_1 = 1$ if and only if $\mathcal{T} \models \square\diamondsuit\varphi$,
- $v_2 = 1$ if and only if $\mathcal{T} \models \diamondsuit\square\varphi$,

– $v_3 = 1$ if and only if $\mathcal{T} \models \square\varphi$.

Thus, the lexicographic ordering on $\{0,1\}^3$ captures the preference of one transition system over another with respect to the quantitative satisfaction of $\square\varphi$.

Example 1 Suppose that we want to synthesize a transition system representing a navigation strategy for a robot working at a restaurant. We require that the robot serves the VIP area infinitely often, formalized in LTL as $\square\diamondsuit\, vip\_area$. We also desire that the robot never enters the staff's office, formalized as $\square\neg office$. Now, suppose that initially the key to the VIP area is in the office. Thus, in order to satisfy $\square\diamondsuit\, vip\_area$, the robot must violate $\square\neg office$. A strategy in which the office is entered only once, and satisfies $\diamondsuit\square\neg office$, is preferable to one which enters the office over and over again, and only satisfies $\square\diamondsuit\neg office$. Thus, we want to synthesize a strategy $\mathcal{T}$ maximizing $val(\mathcal{T}, \square\neg office)$.

In order to compare implementations with respect to their satisfaction of a conjunction of several safety specifications $\square\varphi_1 \wedge \ldots \wedge \square\varphi_n$, we will extend the above definition. We first consider the case where the specifier has not expressed any preference for the individual conjuncts and later on, extend that to the case with a given priority ordering. Consider the following example.

Example 2 We consider again the restaurant robot, now with two soft specifications. The soft specification $\square(req1 \rightarrow \bigcirc table1)$ requires that each request by table 1 is served immediately at the next time instance. Similarly, $\square(req2 \rightarrow \bigcirc table2)$, requires the same for table number 2. Since the robot cannot be at both tables simultaneously, formalized as the hard specification $\square(\neg table1 \vee \neg table2)$, the conjunction of these requirements is unrealizable. Unless the two tables have priorities, it is preferable to satisfy each of $req1 \rightarrow \bigcirc table1$ and $req2 \rightarrow \bigcirc table2$ infinitely often, rather than serve one and the same table all the time.

Quantitative semantics of conjunctions. To capture the idea illustrated in Example 2, we define a value function, which intuitively gives higher values to transition systems in which a fewer number of soft specifications have low values. Formally, let *the value of* $\square\varphi_1 \wedge \ldots \wedge \square\varphi_n$ *in* $\mathcal{T}$ be

$$val(\mathcal{T}, \square\varphi_1 \wedge \ldots \wedge \square\varphi_n) \stackrel{\text{def}}{=} \big(\sum_{i=1}^{n} v_{i,1}, \sum_{i=1}^{n} v_{i,2}, \sum_{i=1}^{n} v_{i,3}\big),$$

where $val(\mathcal{T}, \square\varphi_i) = (v_{i,1}, v_{i,2}, v_{i,3})$ for $i \in \{1, \ldots, n\}$. To compare transition systems according to these values, we use lexicographic ordering on $\{0, \ldots, n\}^3$.

Example 3 For the specifications in Example 2, the defined value function assigns value $(2,0,0)$ to a system satisfying $\square\diamondsuit(req1 \rightarrow \bigcirc table1)$ and $\square\diamondsuit(req2 \rightarrow \bigcirc table2)$, but neither of $\diamondsuit\square(req1 \rightarrow \bigcirc table1)$ and $\diamondsuit\square(req2 \rightarrow \bigcirc table2)$. It assigns the smaller value $(1,1,1)$ to an implementation that gives priority to table 1 and satisfies $\square(req1 \rightarrow \bigcirc table1)$ but not $\square\diamondsuit(req2 \rightarrow \bigcirc table2)$.

According to the definition above, a transition system that satisfies all soft requirements to some extent is considered better in the lexicographic ordering than a transition system that satisfies one of them exactly and violates all the others. We could instead inverse the order of the sums in the triple, thus giving preference to satisfying some soft specification exactly, over having some lower level of satisfaction over all of them. The next example illustrates the differences between the two variations.

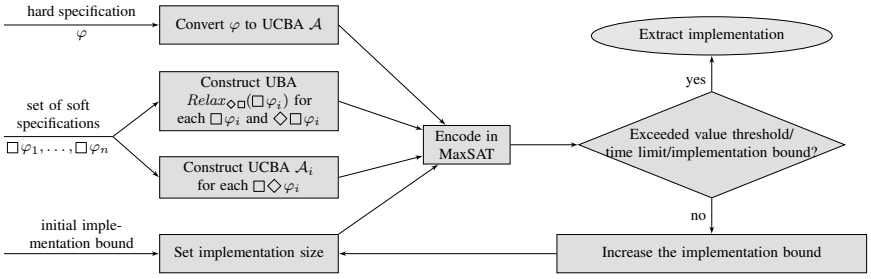


Figure 1: Outline of the maximum realizability procedure.

Example 4 For the two soft specifications from Example 2, reversing the order of the sums in the definition of $val(\mathcal{T}, \square\varphi_1 \wedge \ldots \wedge \square\varphi_n)$ results in giving the higher value $(1, 1, 1)$ to a transition system that satisfies $\square(req1 \rightarrow \bigcirc table1)$ but not $\square\Diamond(req2 \rightarrow \bigcirc table2)$, and the lower value $(0, 0, 2)$ to the one that only guarantees $\square\Diamond(req1 \rightarrow \bigcirc table1)$ and $\square\Diamond(req2 \rightarrow \bigcirc table2)$. The most suitable ordering usually depends on the specific application.

4.2 Problem Formulation

Using the definition of quantitative satisfaction of soft safety specifications, we now define the maximum realizability problem, which asks to synthesize a transition system that satisfies a given *hard* LTL specification, and is optimal with respect to the satisfaction of a conjunction of *soft* safety specifications.

Maximum realizability problem: Given an LTL formula φ and formulas $\square\varphi_1, \ldots, \square\varphi_n$, where each φ_i is a safety LTL formula, the maximum realizability problem asks to determine if there exists a transition system $\mathcal{T}$ such that $\mathcal{T} \models \varphi$, and if the answer is positive, to synthesize a transition system $\mathcal{T}$ such that $\mathcal{T} \models \varphi$, and for every transition system $\mathcal{T}'$ with $\mathcal{T}' \models \varphi$ it holds that $val(\mathcal{T}, \square\varphi_1 \wedge \ldots \wedge \square\varphi_n) \geq val(\mathcal{T}', \square\varphi_1 \wedge \ldots \wedge \square\varphi_n)$.

Bounded maximum realizability problem: Given an LTL formula φ and formulas $\square\varphi_1, \ldots, \square\varphi_n$, where each φ_i is a safety LTL formula, and a bound $b \in \mathbb{N}_{>0}$, the bounded maximum realizability problem asks to determine if there exists a transition system $\mathcal{T}$ with $|\mathcal{T}| \leq b$ such that $\mathcal{T} \models \varphi$, and if the answer is positive, to synthesize a transition system $\mathcal{T}$ such that $\mathcal{T} \models \varphi$, $|\mathcal{T}| \leq b$ and for every transition system $\mathcal{T}'$ with $\mathcal{T}' \models \varphi$ and $|\mathcal{T}'| \leq b$, it holds that $val(\mathcal{T}, \square\varphi_1 \wedge \ldots \wedge \square\varphi_n) \geq val(\mathcal{T}', \square\varphi_1 \wedge \ldots \wedge \square\varphi_n)$.

5 Maximum Realizability as Iterative MaxSAT Solving

We now describe the proposed MaxSAT-based approach to maximum realizability. First, we establish an upper bound on the minimal size of an implementation that satisfies a given LTL specification φ and maximizes the satisfaction of a conjunction of the soft specifications $\square\varphi_1, \ldots, \square\varphi_n$, according to the value function defined in Section 4.1. This bound can be used to reduce the maximum realizability problem to its bounded version, which we encode as a MaxSAT problem.

5.1 Bounded Maximum Realizability

To establish an upper bound on the minimal (in terms of size) optimal implementation, we make use of an important property of the function val defined in Section 4.1. Namely, the property that for each of the possible values of $\square\varphi_1 \wedge \ldots \wedge \square\varphi_n$ there is a corresponding LTL formula that encodes this value in the classical LTL semantics, as we formally state in the next lemma.

Lemma 1 *For every transition system $\mathcal{T}$ and soft safety specifications $\square\varphi_1, \ldots, \square\varphi_n$, if $val(\mathcal{T}, \square\varphi_1 \wedge \ldots \wedge \square\varphi_n) = v$, then there exists an LTL formula ψ_v where*
(1) $\psi_v = \varphi'_1 \wedge \ldots \wedge \varphi'_n$, such that $\varphi'_i \in \{\square\varphi_i, \diamondsuit\square\varphi_i, \square\diamondsuit\varphi_i, true\}$ for $i = 1, \ldots, n$,
(2) $\mathcal{T} \models \psi_v$, and for every $\mathcal{T}'$, if $\mathcal{T}' \models \psi_v$, then $val(\mathcal{T}', \square\varphi_1 \wedge \ldots \wedge \square\varphi_n) \geq v$.

Proof For each $i \in \{1, \ldots, n\}$, let $(v_{i,1}, v_{i,2}, v_{i,3}) = val(\mathcal{T}, \square\varphi_i)$, and let

$$\psi_v^i \stackrel{\text{def}}{=} \begin{cases} \square\varphi_i & \text{if } v_{i,3} = 1, \\ \diamondsuit\square\varphi_i & \text{if } v_{i,3} = 0 \text{ and } v_{i,2} = 1, \\ \square\diamondsuit\varphi_i & \text{if } v_{i,2} = 0 \text{ and } v_{i,1} = 1, \\ true & \text{if } v_{i,1} = 0. \end{cases}$$

We define $\psi_v = \bigwedge_{i=1}^n \psi_v^i$. By the definition of $val(\mathcal{T}, \square\varphi_i)$ and ψ_v^i, we have that $\mathcal{T} \models \psi_v^i$ for all $i \in \{1, \ldots, n\}$. Thus, we conclude that $\mathcal{T} \models \psi_v$. Clearly, ψ_v also satisfies condition *(1)*. Now, consider a transition system $\mathcal{T}'$ with $\mathcal{T}' \models \psi_v$.

Let $(v'_1, v'_2, v'_3) = val(\mathcal{T}', \square\varphi_1 \wedge \ldots \wedge \square\varphi_n)$. We will show that $v'_1 \geq v_1$, $v'_2 \geq v_2$ and $v'_3 \geq v_3$, where $(v_1, v_2, v_3) = v$. Fix some $i \in \{1, \ldots, n\}$.

Let $(v'_{i,1}, v'_{i,2}, v'_{i,3}) = val(\mathcal{T}', \square\varphi_i)$. Since $\mathcal{T}' \models \psi_v$ we have that $\mathcal{T}' \models \psi_v^i$. Thus by the definition of ψ_v^i, we have that if $v_{i,3} = 1$, then $\mathcal{T}' \models \square\varphi_i$, and thus $v'_{i,3} = 1$. Similarly, if $v_{i,2} = 1$ we can conclude that $v'_{i,2} = 1$, and if $v_{i,1} = 1$, then we have $v'_{i,1} = 1$. Since $i \in \{1, \ldots, n\}$ was arbitrary, and since

$$\begin{aligned} (v_1, v_2, v_3) &= \left(\textstyle\sum_{i=1}^n v_{i,1}, \sum_{i=1}^n v_{i,2}, \sum_{i=1}^n v_{i,3}\right) \text{ and} \\ (v'_1, v'_2, v'_3) &= \left(\textstyle\sum_{i=1}^n v'_{i,1}, \sum_{i=1}^n v'_{i,2}, \sum_{i=1}^n v'_{i,3}\right), \end{aligned}$$

we can conclude that $v'_1 \geq v_1$, $v'_2 \geq v_2$ and $v'_3 \geq v_3$. This implies that $(v'_1, v'_2, v'_3) \geq (v_1, v_2, v_3)$ also according to the lexicographic ordering, which proves *(2)*. □

The following theorem is a consequence of Lemma 1.

Theorem 1 *Given an LTL specification φ and soft safety specifications $\square\varphi_1, \ldots, \square\varphi_n$, if there exists a transition system $\mathcal{T} \models \varphi$, then there exists $\mathcal{T}^*$ such that*
(1) $val(\mathcal{T}^*, \square\varphi_1 \wedge \ldots \wedge \square\varphi_n) \geq val(\mathcal{T}, \square\varphi_1 \wedge \ldots \wedge \square\varphi_n)$ for all $\mathcal{T}$ with $\mathcal{T} \models \varphi$,
(2) $\mathcal{T}^* \models \varphi$ and $|\mathcal{T}^*| \leq \left((2^{(b+\log b)})!\right)^2$,
where $b = \max\{|\mathsf{subf}(\varphi \wedge \varphi'_1 \wedge \ldots \wedge \varphi'_n)| \mid \forall i : \varphi'_i \in \{\square\varphi_i, \diamondsuit\square\varphi_i, \square\diamondsuit\varphi_i\}\}$.

Proof Let $v^* = \max\left\{v \in \{0, \ldots, n\}^3 \mid \exists \mathcal{T} : \mathcal{T} \models \varphi \text{ and } val(\mathcal{T}, \square\varphi_1 \wedge \ldots \wedge \square\varphi_n) = v\right\}$. Let $\mathcal{T}$ be a transition system such that $\mathcal{T} \models \varphi$ and $val(\mathcal{T}, \square\varphi_1 \wedge \ldots \wedge \square\varphi_n) = v^*$. According to Lemma 1, there exists an LTL formula ψ_{v^*} that satisfies the conditions of the lemma. Thus, $\mathcal{T} \models \varphi \wedge \psi_{v^*}$. According to [3], there exists a transition system $\mathcal{T}^*$ such that $\mathcal{T}^* \models \varphi \wedge \psi_{v^*}$ and $|\mathcal{T}^*| \leq \left(\left(2^{|\mathsf{subf}(\varphi \wedge \psi_{v^*})| + \log|\varphi \wedge \psi_{v^*}|}\right)!\right)^2$. Combining this with

the guarantees of Lemma 1, we get that $val(\mathcal{T}^*, \square\varphi_1 \wedge \ldots \wedge \square\varphi_n) \geq v^*$, $\mathcal{T}^* \models \varphi$ and $|\mathcal{T}^*| \leq \left((2^{b+\log b})!\right)^2$. Thus, $\mathcal{T}^*$ satisfies condition *(2)*, and since the value v^* is optimal, we have that condition *(1)* holds as well. □

The bound above is estimated based on the size of the specifications, using a worst-case bound on the size of the corresponding automata. Given the automata for all the specifications $\square\varphi_i, \diamondsuit\square\varphi_i$ and $\square\diamondsuit\varphi_i$, a potentially better bound can be estimated based on the sizes of these automata.

Lemma 1 immediately provides a naive synthesis procedure, which searches for an optimal implementation by enumerating possible ψ_v formulas and solving the corresponding realizability questions. The total number of these formulas is 4^n, where n is the number of soft specifications. The approach that we propose avoids this rapid growth, by reducing the optimization problem to a single MaxSAT instance, making use of the power of the state-of-the-art MaxSAT solvers.

Figure 1 gives an overview of our maximum realizability procedure and the automata constructions it involves. As in the bounded synthesis approach, we construct a universal co-Büchi automaton $\mathcal{A}$ for the hard specification φ. For each soft specification $\square\varphi_j$, we construct a pair of automata corresponding to the relaxations of $\square\varphi_j$. The relaxation $\square\diamondsuit\varphi_j$ is treated as in bounded synthesis. For $\square\varphi_i$ and $\diamondsuit\square\varphi_i$, we construct a single universal Büchi automaton and define a corresponding annotation function as described next.

5.2 Automata and Annotations for Soft Safety Specifications

We present here the reduction to MaxSAT for the case when each soft specification is of the form $\square\psi$ where ψ is a *syntactically safe* LTL formula. In this case, we construct a single automaton for both $\square\psi$ and its relaxation $\diamondsuit\square\psi$, and encode the existence of a single annotation function in the MaxSAT problem. The size of this automaton is at most exponential in the size of $\square\psi$. In the general case, we can treat $\square\psi$ and $\diamondsuit\square\psi$ separately, in the same way that we treat the relaxation $\square\diamondsuit\psi$ of $\square\psi$ in the presented encoding. That would require in total three instead of two annotation functions per soft specification.

We now describe the construction of a universal Büchi automaton $\mathcal{B}_{\square\psi}$ for a syntactically safe soft specification $\square\psi$ and show how we can modify it to obtain an automaton $Relax_{\diamondsuit\square}(\square\psi)$ that incorporates the relaxation of $\square\psi$ to $\diamondsuit\square\psi$.

Proposition 1 *Given an LTL formula $\square\psi$ where ψ is syntactically safe, we can construct a universal Büchi automaton $\mathcal{B}_{\square\psi} = (Q_{\square\psi}, q_0^{\square\psi}, \delta_{\square\psi}, F_{\square\psi})$ such that $\mathcal{L}(\mathcal{B}_{\square\psi}) = \{\mathcal{T} \mid \mathcal{T} \models \square\psi\}$, and $\mathcal{B}_{\square\psi}$ has a unique non-accepting sink state, that is, there exists a unique state $\mathsf{rej}_\psi \in Q_{\square\psi}$ such that $F_{\square\psi} = Q_{\square\psi} \setminus \{\mathsf{rej}_\psi\}$, and for every $\sigma \in \Sigma$ it holds that $\{q \in Q_{\square\psi} \mid (\mathsf{rej}_\psi, \sigma, q) \in \delta_{\square\psi}\} = \{\mathsf{rej}_\psi\}$.*

Proof We first describe the construction of the automaton $\mathcal{B}_{\square\psi}$ of the desired form, and then proceed to prove its correctness.

Construction. First we construct a universal Büchi automaton $\mathcal{B}_\psi$ for the formula ψ such that, for every word $w \in \Sigma^\omega$, it holds that w is accepted by $\mathcal{B}_\psi$ if and only if $w \models \psi$. To this end, we use the following result from [29]. Given a syntactically safe LTL formula ψ, we can construct a nondeterministic finite automaton $\mathcal{N} = (Q_\mathcal{N}, q_0^\mathcal{N}, \delta_\mathcal{N}, F_\mathcal{N})$ with at most $2^{\mathcal{O}(|\psi|)}$ states, and such that:
– if $v \in \Sigma^*$ is accepted by $\mathcal{N}$, then for all $w' \in \Sigma^\omega$ we have $vw' \not\models \psi$, and

– for every $w \in \Sigma^\omega$, if $w \not\models \psi$, then there exists a prefix v of w accepted by $\mathcal{N}$. Thus, $\mathcal{N}$ accepts at least one bad prefix of each word $w \in \Sigma^\omega$ that violates ψ.

The automaton $\mathcal{B}_\psi = (Q_\psi, q_0^\psi, \delta_\psi, F_\psi)$ is obtained from $\mathcal{N}$ as follows. The set of states of $\mathcal{B}_\psi$ consists of those states of $\mathcal{N}$ that are not accepting, together with a new state $\mathsf{rej}_\psi \notin Q_\mathcal{N}$, that is, $Q_\psi = (Q_\mathcal{N} \setminus F_\mathcal{N}) \cup \{\mathsf{rej}_\psi\}$. We let $q_0^\psi = q_0^\mathcal{N}$ and $F_\psi = Q_\psi \setminus \{\mathsf{rej}_\psi\}$. The transition relation of $\mathcal{B}_\psi$ is obtained from $\delta_\mathcal{N}$ by redirecting all transitions leading to states in $F_\mathcal{N}$ to the new state rej_ψ. Formally,

$$\delta_\psi = (\delta_\mathcal{N} \cap (Q_\psi \times \Sigma \times Q_\psi)) \cup \{(q, \sigma, \mathsf{rej}_\psi) \mid \sigma \in \Sigma \text{ and } \exists q' \in F_\mathcal{N}.\ (q, \sigma, q') \in \delta_\mathcal{N}\} \cup \{(\mathsf{rej}_\psi, \sigma, \mathsf{rej}_\psi) \mid \sigma \in \Sigma\}.$$

We now construct a universal Büchi automaton $\mathcal{B}_{\Box\psi} = (Q_{\Box\psi}, q_0^{\Box\psi}, \delta_{\Box\psi}, F_{\Box\psi})$ such that w is accepted by $\mathcal{B}_{\Box\psi}$ iff $w \models \Box\psi$. We let $Q_{\Box\psi} = Q_\psi$, $q_0^{\Box\psi} = q_0^\psi$, and $F_{\Box\psi} = F_\psi$. The transition relation $\delta_{\Box\psi}$ extends δ_ψ by adding a self-loop at the initial state $q_0^{\Box\psi}$ for all transitions from q_0^ψ in δ_ψ that do not lead to rej_ψ:

$$\delta_{\Box\psi} = \delta_\psi \cup \{(q_0^\psi, \sigma, q_0^\psi) \mid \sigma \in \Sigma \text{ and } \exists q' \in (Q_\psi \setminus \{\mathsf{rej}_\psi\}).\ (q_0^\psi, \sigma, q') \in \delta_\psi\}.$$

Correctness. Let $\mathcal{T} \in \mathcal{L}(\mathcal{B}_{\Box\psi})$. Since $\mathcal{B}_{\Box\psi}$ is a universal Büchi automaton, this means that the unique run graph of $\mathcal{B}_{\Box\psi}$ on $\mathcal{T}$ is accepting, which in turn means that each infinite path contains infinitely many occurrences of states in $F_{\Box\psi}$. Since $F_{\Box\psi}$ contains all states except rej_ψ, and rej_ψ is a sink state, it follows that every infinite path in the run graph contains only states in $F_{\Box\psi}$.

Suppose, for the sake of contradiction, that $\mathcal{T} \not\models \Box\psi$. Thus, there exists $\omega = \sigma_0, \sigma_1, \ldots \in \mathsf{Traces}(\mathcal{T})$ such that $\omega \not\models \Box\psi$. Let $i \geq 0$ be an index such that $\sigma_i, \sigma_{i+1}, \ldots \not\models \psi$. By the choice of the automaton $\mathcal{N}$, there exists a prefix of $\sigma_i, \sigma_{i+1}, \ldots$ accepted by $\mathcal{N}$. Since $\omega \in \mathsf{Traces}(\mathcal{T})$ and $\mathcal{T} \in \mathcal{L}(\mathcal{B}_{\Box\psi})$, every path in the run graph corresponding to ω never visits rej_ψ. Thus, since $\delta_{\Box\psi}$ contains a self-loop at state $q_0^{\Box\psi}$ with letters not leading to rej_ψ, there exists a path in the run graph corresponding to $\sigma_0, \ldots, \sigma_{i-1}$ that ends in $q_0^{\Box\psi}$. By the definition of δ_ψ and the existence of an accepting run of $\mathcal{N}$ on a prefix of $\sigma_i, \sigma_{i+1}, \ldots$ we can conclude that there exists a path in the run graph of $\mathcal{B}_{\Box\psi}$ corresponding to ω that reaches rej_ψ, which is a contradiction.

For the other direction, consider a transition system $\mathcal{T}$ such that $\mathcal{T} \models \Box\psi$, and suppose that $\mathcal{T} \notin \mathcal{L}(\mathcal{B}_{\Box\psi})$. This means that there exists an infinite path in the run graph of $\mathcal{B}_{\Box\psi}$ on $\mathcal{T}$ that visits states in $F_{\Box\psi}$ only finitely many times, which means that this path eventually reaches rej_ψ. Let $\omega = \sigma_0, \sigma_1, \ldots \in \mathsf{Traces}(\mathcal{T})$ be the word corresponding to this path, and $i \geq 0$ be the last occurrence of $q_0^{\Box\psi}$ on this path and $j > i$ be the index of the first occurrence of rej_ψ. Due to the definition of δ_ψ, this implies that there exists an accepting run of $\mathcal{N}$ on the word $\sigma_i, \ldots, \sigma_{j-1}$. Thus, $\sigma_i, \sigma_{i+1}, \ldots \not\models \psi$, which in turn means that $\omega \not\models \Box\psi$. This is a contradiction with $\mathcal{T} \models \Box\psi$, and thus we can conclude that $\mathcal{T} \in \mathcal{L}(\mathcal{B}_{\Box\psi})$. □

From $\mathcal{B}_{\Box\psi}$, which has at most $2^{\mathcal{O}(|\psi|)}$ states, we obtain a universal Büchi automaton $\mathit{Relax}_{\Diamond\Box}(\Box\psi)$ constructed by redirecting all the transitions leading to rej_ψ to the initial state $q_0^{\Box\psi}$. Formally, $\mathit{Relax}_{\Diamond\Box}(\Box\psi) = (Q, q_0, \delta, F)$, where $Q = Q_{\Box\psi} \setminus \{\mathsf{rej}_\psi\}$, $q_0 = q_0^{\Box\psi}$, $F = F_{\Box\psi}$ and $\delta = \big(\delta_{\Box\psi} \setminus \{(q, \sigma, q') \in \delta_{\Box\psi} \mid q' = \mathsf{rej}_\psi\}\big) \cup \{(q, \sigma, q_0) \mid (q, \sigma, \mathsf{rej}_\psi) \in \delta_{\Box\psi}\}$.

Let $\mathit{Rej}(\mathit{Relax}_{\Diamond\Box}(\Box\psi)) = \{(q, \sigma, q_0) \in \delta \mid (q, \sigma, \mathsf{rej}_\psi) \in \delta_\psi\}$ be the set of transitions in $\mathit{Relax}_{\Diamond\Box}(\Box\psi)$ that correspond to transitions in $\mathcal{B}_{\Box\psi}$ leading to rej_ψ. The automaton $\mathit{Relax}_{\Diamond\Box}(\Box\psi)$ has the property that its run graph on a transition system $\mathcal{T}$ does *not* contain

a reachable edge corresponding to a transition in $Rej(Relax_{\Diamond\Box}(\Box\psi))$ iff $\mathcal{T}$ is accepted by the automaton $\mathcal{B}_{\Box\psi}$, (i.e., $\mathcal{T} \models \Box\psi$). Otherwise, if the run graph of $Relax_{\Diamond\Box}(\Box\psi)$ on $\mathcal{T}$ contains a reachable edge that belongs to $Rej(Relax_{\Diamond\Box}(\Box\psi))$, then $\mathcal{T} \not\models \Box\psi$. However, if each infinite path in the run graph contains only a finite number of occurrences of such edges, then $\mathcal{T} \models \Diamond\Box\psi$. Based on these observations, we define an annotation function that annotates each node in the run graph with an upper bound on the number of edges in $Rej(Relax_{\Diamond\Box}(\Box\psi))$ visited on any path reaching the node.

The next proposition formalizes the property that a transition system $\mathcal{T}$ is accepted by $\mathcal{B}_{\Box\psi}$ if and only if the run graph of $Relax_{\Diamond\Box}(\Box\psi)$ on $\mathcal{T}$ does not contain a reachable edge corresponding to a transition in $Rej(Relax_{\Diamond\Box}(\Box\psi))$.

Proposition 2 *Let $\mathcal{T}$ be a transition system and let $G = (V, E)$ be the run graph of $Relax_{\Diamond\Box}(\Box\psi)$ on $\mathcal{T}$. Then, $\mathcal{T} \in \mathcal{L}(\mathcal{B}_{\Box\psi})$ iff for every $((s,q),\sigma,(s',q')) \in E$ with $(q,\sigma,q') \in Rej(Relax_{\Diamond\Box}(\Box\psi))$, (s,q) is not reachable from (s_0,q_0) in G.*

Proof Suppose, for the sake of contradiction, that $\mathcal{T} \in \mathcal{L}(\mathcal{B}_{\Box\psi})$ and let $G' = (V', E')$ be the run graph of $\mathcal{B}_{\Box\psi}$ on $\mathcal{T}$. Suppose that there exists a path $(s_0,q_0),\sigma_0,\dots(s_l,q_l)$ in G such that there exists an edge $((s_l,q_l),\sigma,(s',q')) \in E$ with $(q_l,\sigma,q') \in Rej(Relax_{\Diamond\Box}(\Box\psi))$. Without loss of generality, we assume that $(q_i,\sigma_i,q_{i+1}) \notin Rej(Relax_{\Diamond\Box}(\Box\psi))$ for all $i = 0,\dots,l-1$. Then, the sequence $(s_0,q_0),\sigma_0,\dots(s_l,q_l),\sigma,(s',q')$ corresponds to a path in the run graph G' of $\mathcal{B}_{\Box\psi}$ on $\mathcal{T}$ which enters the state rej_ψ. Since rej_ψ is a non-accepting sink state, we conclude that G' is not accepting. This implies $\mathcal{T} \notin \mathcal{L}(\mathcal{B}_{\Box\psi})$, which is a contradiction.

Suppose now that for every node (s,q) reachable in G from (s_0,q_0) and every edge $((s,q),\sigma,(s',q')) \in E$ we have that $(q,\sigma,q') \notin Rej(Relax_{\Diamond\Box}(\Box\psi))$. Assume that $\mathcal{T} \notin \mathcal{L}(\mathcal{B}_{\Box\psi})$, which means that there exists an infinite path from (s_0,q_0) in the run graph of $\mathcal{B}_{\Box\psi}$ on $\mathcal{T}$ that reaches the state rej_ψ. This path corresponds to a path in G from (s_0,q_0) to some state (s,q) for which there is an edge $((s,q),\sigma,(s',q')) \in E$ with $(q,\sigma,q') \in Rej(Relax_{\Diamond\Box}(\Box\psi))$, which is a contradiction. □

A function $\pi : S \times Q \to \mathbb{N} \cup \{\bot\}$ is $\Diamond\Box$–*valid* annotation for the run graph of the automaton $Relax_{\Diamond\Box}(\Box\psi) = (Q, q_0, \delta, F)$ on the transition system $\mathcal{T} = (S, s_0, \tau)$ if
(1) $\pi(s_0,q_0) \neq \bot$, i.e., the pair of initial states is labeled with a number, and
(2) if $\pi(s,q) \neq \bot$, then for every edge $((s,q),\sigma,(s',q'))$ in the run graph, we have that $\pi(s',q') \neq \bot$, and
- if $(q,\sigma,q') \in Rej(Relax_{\Diamond\Box}(\Box\psi))$, then $\pi(s',q') > \pi(s,q)$, and
- if $(q,\sigma,q') \notin Rej(Relax_{\Diamond\Box}(\Box\psi))$, then $\pi(s',q') \geq \pi(s,q)$.

This guarantees that $\mathcal{T} \models \Diamond\Box\psi$ iff there exists a $\Diamond\Box$–valid $|\mathcal{T}|$-bounded annotation π for $\mathcal{T}$ and $Relax_{\Diamond\Box}(\Box\psi)$. Moreover, if π is $|\mathcal{T}|$-bounded and $\pi(s_0,q_0) = |\mathcal{T}|$, then $\mathcal{T} \models \Box\psi$, as this means that no edge in $Rej(Relax_{\Diamond\Box}(\Box\psi))$ is ever reached.

Proposition 3 *Let $\mathcal{T} = (S, s_0, \tau)$ be a finite-state transition system, and $G = (V, E)$ be the run graph of $Relax_{\Diamond\Box}(\Box\psi)$ on $\mathcal{T}$. Then, $\mathcal{T} \models \Diamond\Box\psi$ if and only if there exists a $\Diamond\Box$–valid $|\mathcal{T}|$-bounded annotation for G.*

Proof Suppose that $\mathcal{T} \models \Diamond\Box\psi$. We will fist show that in every infinite path from (s_0,q_0) in G there are at most $|S|$ occurrences of edges whose corresponding transitions are in $Rej(Relax_{\Diamond\Box}(\Box\psi))$, and then we will use this fact to define a $\Diamond\Box$–valid $|S|$-bounded annotation. Assume, for the sake of contradiction, that there exists an infinite path $(s_0,q_0),\sigma_0,(s_1,q_1),\sigma_1,\dots$ such that for infinitely many positions $i \geq 0$ it holds that

$(q_i, \sigma_i, q_{i+1}) \in Rej(Relax_{\Diamond\Box}(\Box\psi))$. Let $i_1 < i_2 < \ldots$ be a sequence of such positions. By the construction of $Relax_{\Diamond\Box}(\Box\psi)$, we have $q_{i_j+1} = q_0$ for each i_j. Thus, using reasoning similar to that in Proposition 2, we can show that the trace σ_0, σ_1, contains infinitely many positions k such that $\sigma_k, \sigma_{k+1}, \ldots \not\models \psi$. This means that $\sigma_0, \sigma_1, \ldots \not\models \Diamond\Box\psi$. Since $\sigma_0, \sigma_1, \ldots \in \mathsf{Traces}(\mathcal{T})$, we can conclude that $\mathcal{T} \not\models \Diamond\Box\psi$, which is a contradiction.

Thus, each infinite path in G contains only finitely many occurrences of edges in $Rej(Relax_{\Diamond\Box}(\Box\psi))$. Since the number of distinct nodes in G of the form (s, q_0) is $|S|$, we obtain an upper bound of $|S|$ occurrences of transitions from $Rej(Relax_{\Diamond\Box}(\Box\psi))$ on every path in G. Thus, we can construct a $\Diamond\Box$–valid $|S|$-bounded annotation π by mapping each reachable node (s, q) to the maximal number of transitions from $Rej(Relax_{\Diamond\Box}(\Box\psi))$ on a path from (s_0, q_0) to (s, q), and mapping each unreachable node to $\bot$.

For the other direction, suppose that π is a $\Diamond\Box$–valid $|S|$-bounded annotation for $\mathcal{T}$ and $Relax_{\Diamond\Box}(\Box\psi)$. Assume that $\mathcal{T} \not\models \Diamond\Box\psi$. This means that there exists a trace $w = \sigma_0, \sigma_1, \ldots \in \mathsf{Traces}(\mathcal{T})$ such that for every position i it holds that $\sigma_i, \sigma_i + 1, \ldots \not\models \Box\psi$. Let $s_0, \sigma_0, s_1, \sigma_1 \ldots$ be the execution of $\mathcal{T}$ corresponding to w. Since $\sigma_i, \sigma_i + 1, \ldots \not\models \Box\psi$, with reasoning similar to Proposition 2 we can establish that there exists a path in G starting from (s_i, q_0) that eventually takes an edge corresponding to a transition in $Rej(Relax_{\Diamond\Box}(\Box\psi))$, and by the construction of $Relax_{\Diamond\Box}(\Box\psi)$, this transition leads to the node (s_j, q_0) for some $j > i$. Thus, by induction, we can establish the existence of an infinite path $(s_0, q_0), (s_1, q_1), \ldots$ in G that contains infinitely many occurrences of edges whose transitions are in $Rej(Relax_{\Diamond\Box}(\Box\psi))$. Since the annotation π is $\Diamond\Box$–valid, we can show by induction that for each $i \geq 0$ it holds that $\pi(s_i, q_i) \in \mathbb{N}$ and $\pi(s_{i+1}, q_{i+1}) \geq \pi(s_i, q_i)$. Since G is finite, this path contains an edge $((s_i, q_i), \sigma_i, (s_{i+1}, q_{i+1}))$ for which $(q_i, \sigma_i, q_{i+1}) \in Rej(Relax_{\Diamond\Box}(\Box\psi))$, and which is such that there exists $j \leq i$ such that $(s_{i+1}, q_{i+1}) = (s_j, q_j)$. Since the annotation π is $\Diamond\Box$–valid, we have that $\pi(s_j, q_j) \leq \pi(s_i, q_i)$ and $\pi(s_i, q_i) < \pi(s_{i+1}, q_{i+1})$, which contradicts $(s_{i+1}, q_{i+1}) = (s_j, q_j)$. Thus, by contradiction, we conclude that $\mathcal{T} \models \Diamond\Box\psi$. $\square$

In particular, we have that if $\pi(s_0, q_0) \in \mathbb{N}$, then $\mathcal{T} \models \Diamond\Box\psi$, and if π is c-bounded and $\pi(s_0, q_0) = c$, then $\mathcal{T} \models \Box\psi$. This property allows us to capture the satisfaction of $\Box\psi$ and $\Diamond\Box\psi$ with soft clauses for the same annotation function in the MaxSAT formulation.

5.3 MaxSAT Encoding of Bounded Maximum Realizability

Let $\mathcal{A} = (Q, q_0, \delta, F)$ be a universal co-Büchi automaton for the LTL formula φ. For each syntactically safe formula $\Box\varphi_j$, $j \in \{1, \ldots, n\}$, we consider two universal automata: the universal automaton $\mathcal{B}_j = Relax_{\Diamond\Box}(\Box\varphi_j) = (Q_j, q_0^j, \delta_j, F_j)$, constructed as described in Section 5.2, and a universal co-Büchi automaton $\mathcal{A}_j = (\widehat{Q}_j, \widehat{q}_0^j, \widehat{\delta}_j, \widehat{F}_j)$ for the formula $\Box\Diamond\varphi_j$. Given a bound b on the size of the desired transition system, we encode the bounded maximum realizability problem as a MaxSAT problem with the following sets of variables and constraints.

Variables: The MaxSAT formulation includes the variables from the SAT formulation of the bounded synthesis problem, which represent the desired transition system $\mathcal{T}$ and the desired valid annotation of the run graph of $\mathcal{A}$ on $\mathcal{T}$. Additionally, it includes variables for representing the annotations π_j and λ_j for $\mathcal{B}_j$ and $\mathcal{A}_j$, respectively, similarly to λ in the SAT encoding. More precisely, the variables for π_j and λ_j are respectively represented by variables $\pi_{s,q}^{\mathbb{B},j}$ and $\pi_{s,q}^{\mathbb{N},j}$ where $s \in S$ and $q \in Q_j$, and variables $\lambda_{s,q}^{\mathbb{B},j}$ and $\lambda_{s,q}^{\mathbb{N},j}$ where $s \in S$ and $q \in \widehat{Q}_j$.

The set of constraints includes C_τ and C_λ from the SAT formulation as hard constraints, as well as the following constraints for the new annotations.

Hard constraints for valid annotations: For each $j = 1, \ldots, n$, let

$$C^j_\pi \stackrel{\text{def}}{=} \bigwedge_{q,q' \in Q_j} \bigwedge_{s,s' \in S} \bigwedge_{\sigma_I \in 2^{\mathcal{I}}} \Big((\pi^{\mathbb{B},j}_{s,q} \wedge \delta^j_{s,q,\sigma_I,q'} \wedge \tau_{s,\sigma_I,s'}) \rightarrow \mathsf{succ}^j_\pi(s,q,s',q',\sigma_I) \Big),$$

$$C^j_\lambda \stackrel{\text{def}}{=} \bigwedge_{q,q' \in \widehat{Q}_j} \bigwedge_{s,s' \in S} \bigwedge_{\sigma_I \in 2^{\mathcal{I}}} \Big((\lambda^{\mathbb{B},j}_{s,q} \wedge \widehat{\delta}^j_{s,q,\sigma_I,q'} \wedge \tau_{s,\sigma_I,s'}) \rightarrow \mathsf{succ}^j_\lambda(s,q,s',q',\sigma_I) \Big),$$

$$\text{where} \quad \mathsf{succ}^j_\pi(s,q,s',q',\sigma_I) \stackrel{\text{def}}{=} \pi^{\mathbb{B},j}_{s',q'} \wedge \big(\mathsf{rej}^j(s,q,q',\sigma_I) \rightarrow \pi^{\mathbb{N},j}_{s',q'} > \pi^{\mathbb{N},j}_{s,q}\big) \wedge \big(\neg \mathsf{rej}^j(s,q,q',\sigma_I) \rightarrow \pi^{\mathbb{N},j}_{s',q'} \geq \pi^{\mathbb{N},j}_{s,q}\big),$$

and $\mathsf{rej}^j(s,q,q',\sigma_I)$ is a formula over o_{s,σ_I} obtained from $Rej(\mathcal{B}_j)$.The formula $\mathsf{succ}^j_\lambda(s,\widehat{q},s',\widehat{q}',\sigma_I)$ is analogous to $\mathsf{succ}_\lambda(s,q,s',q',\sigma_I)$ defined in Section 3.2.

Soft constraints for valid annotations: Let $b \in \mathbb{N}_{>0}$ be the bound on the size of the transition system. For each $j = 1, \ldots, n$, we define

$$\begin{array}{lll} Soft^j_{\Box} & \stackrel{\text{def}}{=} \pi^{\mathbb{B},j}_{s_0,q_0} \wedge (\pi^{\mathbb{N},j}_{s_0,q_0} = b) & \text{with weight } 1, \\ Soft^j_{\Diamond\Box} & \stackrel{\text{def}}{=} \pi^{\mathbb{B},j}_{s_0,q_0} & \text{with weight } n, \text{ and} \\ Soft^j_{\Box\Diamond} & \stackrel{\text{def}}{=} \pi^{\mathbb{B},j}_{s_0,q_0} \vee \lambda^{\mathbb{B},j}_{s_0,\widehat{q}_0} & \text{with weight } n^2. \end{array}$$

The definition of the soft constraints guarantees that $\mathcal{T} \models \Box\varphi_j$ if and only if there exist corresponding annotations that satisfy all three of the soft constraints for $\Box\varphi_j$. Similarly, if $\mathcal{T} \models \Diamond\Box\varphi_j$, then $Soft^j_{\Diamond\Box}$ and $Soft^j_{\Box\Diamond}$ can be satisfied. The weights of the soft clauses reflect the ordering of transition systems with respect to their satisfaction of $\Box\varphi_1 \wedge \ldots \wedge \Box\varphi_n$, as stated below.

Lemma 2 *Let $\mathcal{T}'$ and $\mathcal{T}''$ be two transition systems such that $\mathcal{T}' \models \varphi$ and $\mathcal{T}'' \models \varphi$. Let a' and a'' be the variable assignments satisfying the constraint system, such that a' is an optimal assignment consistent with $\mathcal{T}'$, and a'' is an optimal assignment consistent with $\mathcal{T}''$. Furthermore, let w' and w'' be the sums of the weights of the soft clauses satisfied by a' and a'', respectively. Then, it holds that $val(\mathcal{T}', \Box\varphi_1 \wedge \ldots \wedge \Box\varphi_n) < val(\mathcal{T}'', \Box\varphi_1 \wedge \ldots \wedge \Box\varphi_n)$ iff $w' < w''$.*

Proof Let $(v'_1, v'_2, v'_3) = val(\mathcal{T}', \Box\varphi_1 \wedge \ldots \wedge \Box\varphi_n)$ and $(v''_1, v''_2, v''_3) = val(\mathcal{T}'', \Box\varphi_1 \wedge \ldots \wedge \Box\varphi_n)$. This means that there are exactly v'_3 distinct indices $i \in \{1, \ldots, n\}$ such that $\mathcal{T}' \models \Box\varphi_i$, v'_2 distinct indices $i \in \{1, \ldots, n\}$ such that $\mathcal{T}' \models \Diamond\Box\varphi_i$ and v'_1 distinct indices $i \in \{1, \ldots, n\}$ such that $\mathcal{T}' \models \Box\Diamond\varphi_i$. Since a' is an optimal satisfying assignment corresponding to $\mathcal{T}'$, we have that a' satisfies exactly v'_3 of the soft clauses $Soft^j_{\Box}$, exactly v'_2 of the soft clauses $Soft^j_{\Diamond\Box}$ and exactly v'_1 of the soft clauses $Soft^j_{\Box\Diamond}$. This means that $w' = v'_3 + v'_2 \cdot n + v'_1 \cdot n^2$. In a similar way we can conclude that $w'' = v''_3 + v''_2 \cdot n + v''_1 \cdot n^2$ holds for $\mathcal{T}''$.

First, suppose that $(v'_1, v'_2, v'_3) < (v''_1, v''_2, v''_3)$. There are three possible cases:
Case 1: $v'_1 = v''_1$, $v'_2 = v''_2$ and $v'_3 < v''_3$. Then, $w'' - w' = (v''_3 - v'_3) > 0$.
Case 2: $v'_1 = v''_1$ and $v'_2 < v''_2$. Then, $w'' - w' = (v''_2 - v'_2) \cdot n + (v''_3 - v'_3)$. Since $\mathcal{T}' \models \Box\varphi_i$ implies $\mathcal{T}' \models \Diamond\Box\varphi_i$, we have that $v'_3 - v''_3 \leq n - 1$, due to the fact that $v''_2 - v'_2 \geq 1$. Thus, we conclude $w'' - w' \geq n - (n-1) = 1 > 0$.

Case 3: $v'_1 < v''_1$. Now, $w'' - w' = (v''_1 - v'_1) \cdot n^2 + (v''_2 - v'_2) \cdot n + (v''_3 - v'_3)$. Again, since $\mathcal{T}' \models \Box\varphi_i$ implies $\mathcal{T}' \models \Box\Diamond\varphi_i$ and $\mathcal{T}' \models \Diamond\Box\varphi_i$ implies $\mathcal{T}' \models \Box\Diamond\varphi_i$, we have that $v'_3 - v''_3 \leq n-1$ and $v'_2 - v''_2 \leq n-1$, both due to the fact that $v''_1 - v'_1 \geq 1$. Thus, we conclude $w'' - w' \geq n^2 - (n-1)\cdot n - (n-1) = 1 > 0$.

In all three cases we showed that $w' < w''$.

For the other direction, suppose that $w' < w''$. If we assume that $(v'_1, v'_2, v'_3) \geq (v''_1, v''_2, v''_3)$, then we can show as above that $w'' \geq w'$, which is a contradiction. Hence, we have that $(v'_1, v'_2, v'_3) < (v''_1, v''_2, v''_3)$, which concludes the proof. □

This in turn guarantees that a transition system extracted from an optimal satisfying assignment for the MaxSAT problem is optimal with respect to the value of $\Box\varphi_1 \wedge \ldots \wedge \Box\varphi_n$, as stated in the following theorem that establishes the correctness of the encoding.

Theorem 2 *Let $\mathcal{A}$ be a given co-Büchi automaton for φ, and for each $j \in \{1, \ldots, n\}$, let $\mathcal{B}_j = Relax_{\Diamond\Box}(\Box\varphi_j)$ be the universal automaton for $\Box\varphi_j$ constructed as in Section 5.2, and let $\mathcal{A}_j$ be a universal co-Büchi automaton for $\Box\Diamond\varphi_j$. The constraint system for bound $b \in \mathbb{N}_{>0}$ is satisfiable if and only if there exists an implementation $\mathcal{T}$ with $|\mathcal{T}| \leq b$ such that $\mathcal{T} \models \varphi$. Furthermore, from the optimal satisfying assignment to the variables $\tau_{s,\sigma_I,s'}$ and o_{s,σ_I}, one can extract a transition system $\mathcal{T}^*$ such that for every transition system $\mathcal{T}$ with $|\mathcal{T}| \leq b$ and $\mathcal{T} \models \varphi$ it holds that $val(\mathcal{T}^*, \Box\varphi_1 \wedge \ldots \wedge \Box\varphi_n) \geq val(\mathcal{T}, \Box\varphi_1 \wedge \ldots \wedge \Box\varphi_n)$.*

Proof The first part of the claim follows from the correctness of the classical bounded synthesis approach. More precisely, if the constraint system is satisfiable, then there exists a satisfying assignment, which in particular, satisfies the constraints asserting the existence of a transition system $\mathcal{T}$ of size less than or equal to b, and the existence of a valid annotation for the run graph of $\mathcal{A}$ on $\mathcal{T}$. If, on the other hand, there exists a transition system $\mathcal{T}$ such that $|\mathcal{T}| \leq b$ and $\mathcal{T} \models \varphi$, then there exists a variable assignment a consistent with $\mathcal{T}$ that satisfies the constraints asserting the existence of a valid annotation for the run graph of $\mathcal{A}$ on $\mathcal{T}$. It remains to show that a can be chosen in a way that satisfies the remaining hard constraints as well. To see that, notice that all the constraints for the annotations π_j and λ_j can be satisfied (not necessarily in an optimal way) by setting all the variables $\pi^{\mathbb{B},j}_{s,q}$ and $\lambda^{\mathbb{B},j}_{s,q}$ to false. This completes the proof of the first statement.

Now, let a^* be an optimal solution to the MaxSAT problem, and $\mathcal{T}^*$ be the transition system extracted from a^*. Consider a transition system $\mathcal{T}$ such that $|\mathcal{T}| \leq b$ and $\mathcal{T} \models \varphi$. Then, as we showed above, there exists a satisfying assignment a consistent with $\mathcal{T}$. Let w^* be the sum of the weights of the soft clauses satisfied by a^*, and w be the sum of the weighs of the soft clauses satisfied by a. Since a^* is an optimal satisfying assignment, we have that $w \leq w^*$. Thus, by applying Lemma 2 we obtain $val(\mathcal{T}^*, \Box\varphi_1 \wedge \ldots \wedge \Box\varphi_n) \geq val(\mathcal{T}, \Box\varphi_1 \wedge \ldots \wedge \Box\varphi_n)$, which concludes the proof of the second claim of the theorem. □

Figure 2 shows a transition system extracted from an optimal satisfying assignment for Example 2 with bound 3 on the implementation size. The transitions depicted in the figure are defined by the values of the variables $\tau_{s,\sigma_I,s'}$. The outputs of the implementation (omitted from the figure) are defined by the values of o_{s,σ_I}. The output in state s_1 when $r1$ is true is $table1 \wedge \neg table2$, and the output in s_2 when $r2$ is true is $\neg table1 \wedge table2$. For all other combinations of state and input, the output is $\neg table1 \wedge \neg table2$.

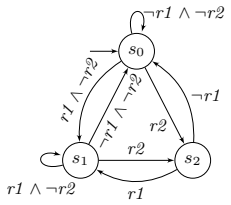


Figure 2: An optimal implementation for Example 2.

The next proposition establishes the size of the MaxSAT encoding.

Proposition 4 *Let $\mathcal{A}$ be a given co-Büchi automaton for φ, and for each $j \in \{1, \ldots, n\}$, let $\mathcal{B}_j = Relax_{\Diamond\Box}(\Box\varphi_j)$ be the universal Büchi automaton for $\Box\varphi_j$ constructed as in Section 5.2, and let $\mathcal{A}_j$ be a universal co-Büchi automaton for $\Box\Diamond\varphi$. The constraint system for bound $b \in \mathbb{N}$ has weights in $\mathcal{O}(n^2)$. It has*

$$\mathcal{O}\Big((b^2 + b \cdot |\mathcal{O}|) \cdot 2^{|\mathcal{I}|} + b \cdot |Q| \cdot (1 + \log(b \cdot |Q|)) + \\ \textstyle\sum_{j=1}^{n} \big(b \cdot |Q_j|(1 + \log(b \cdot |Q_j|))\big) + \sum_{j=1}^{n} \big(b \cdot |\widehat{Q}_j|(1 + \log(b \cdot |\widehat{Q}_j|))\big)\Big)$$

variables, and its size (before conversion to CNF) is

$$\mathcal{O}\Big(|Q|^2 \cdot b^2 \cdot 2^{|\mathcal{I}|} \cdot (d + \log(b \cdot |Q|)) + \\ \textstyle\sum_{j=1}^{n} \big(|Q_j|^2 \cdot b^2 \cdot 2^{|\mathcal{I}|} \cdot (d_j + r_j + \log(b \cdot |Q_j|))\big) + \\ \textstyle\sum_{j=1}^{n} \big(|\widehat{Q}_j|^2 \cdot b^2 \cdot 2^{|\mathcal{I}|} \cdot (\widehat{d}_j + \log(b \cdot |\widehat{Q}_j|))\big)\Big),$$

where $d = \max_{s,q,\sigma_I,q'} |\delta_{s,q,\sigma_I,q'}|$, $\quad d_j = \max_{s,q,\sigma_I,q'} |\delta^j_{s,q,\sigma_I,q'}|$,
$\widehat{d}_j = \max_{s,q,\sigma_I,q'} |\widehat{\delta}^j_{s,q,\sigma_I,q'}|$, *and* $r_j = \max_{s,q,\sigma_I,q'} |\mathsf{rej}^j(s,q,q',\sigma_I)|$.

Proof The constraint system is defined in terms of the following variables:

- Boolean variables $\tau_{s,\sigma_I,s'}$ and o_{s,σ_I} representing the transition system. The total number of these variables is $b^2 \cdot 2^{|\mathcal{I}|} + b \cdot |\mathcal{O}| \cdot 2^{|\mathcal{I}|}$.
- Boolean variables $\lambda^{\mathbb{B}}_{s,q}$ and vectors of Boolean variables $\lambda^{\mathbb{N}}_{s,q}$ representing the annotation λ. The total number of bits is $b \cdot |Q| + b \cdot |Q| \cdot \log(b \cdot |Q|)$.
- Boolean variables $\pi^{\mathbb{B},j}_{s,q}$ and vectors of Boolean variables $\pi^{\mathbb{N},j}_{s,q}$ representing the annotations π_j. The total number of bits is $\sum_{j=1}^{n} \big(b \cdot |Q_j|(1 + \log(b \cdot |Q_j|))\big)$.
- Boolean variables $\lambda^{\mathbb{B},j}_{s,q}$ and vectors of Boolean variables $\lambda^{\mathbb{N},j}_{s,q}$ representing the annotation λ_j. The total number of bits is $\sum_{j=1}^{n} \big(b \cdot |\widehat{Q}_j|(1 + \log(b \cdot |\widehat{Q}_j|))\big)$.

The sum of the above quantities yields the total number of Boolean variables.
The constraint system consists of the following constraints:

- Constraints C_τ encoding input-enabledness, of size $b^2 \cdot 2^{|\mathcal{I}|}$.
- Constraints C_λ for valid annotation λ of size $\mathcal{O}\big(|Q|^2 \cdot b^2 \cdot 2^{|\mathcal{I}|} \cdot (d + \log(b \cdot |Q|))\big)$.
- Hard constraints C^j_π for valid annotations π_j, of size

$$\mathcal{O}\big(\sum_{j=1}^{n} \big(|Q_j|^2 \cdot b^2 \cdot 2^{|\mathcal{I}|} \cdot (d_j + r_j + \log(b \cdot |Q_j|))\big)\big).$$

- Hard constraints C^j_λ for valid annotations λ_j, of size

$$\mathcal{O}\big(\sum_{j=1}^{n} \big(|\widehat{Q}_j|^2 \cdot b^2 \cdot 2^{|\mathcal{I}|} \cdot (\widehat{d}_j + \log(b \cdot |Q_j|))\big)\big).$$

- Soft constraints $Soft^j_\Box$, $Soft^j_{\Diamond\Box}$ and $Soft^j_{\Box\Diamond}$ for valid annotations, of size

$$\mathcal{O}\big(\sum_{j=1}^{n} \big(|\log(b \cdot |Q_j|)|\big)\big).$$

Summing up, we obtain the total size of the constraint system. □

5.4 Generalizations of the Maximum Realizability Problem

5.4.1 Maximum Realizability with Soft LTL Specifications

The first generalization of the maximum realizability problem that we consider is the setting where the soft specifications can be arbitrary LTL formulas, and not just safety properties of specific form. More precisely, each soft specification φ is an LTL formula for which we are also given a vector $Relax(\varphi)$ of LTL formulas that defines the possible relaxations of φ. Formally, $Relax(\varphi) = (\psi_1, \ldots, \psi_m)$, where $\psi_1 = \varphi$, and for every $1 \leq k < m$, it holds that $\mathcal{T} \models \psi_k$ implies $\mathcal{T} \models \psi_{k+1}$ for every transition system $\mathcal{T}$. That is, $\psi_1, \ldots, \psi_m$ are ordered according to strength. In particular, if $\mathcal{T} \models \varphi$, then $\mathcal{T} \models \psi_k$ for each ψ_k in $Relax(\varphi)$. For example, if $\varphi = \Box p$ for some atomic proposition p, we can take $Relax(\varphi) = (\Box p, \Diamond\Box p, \Box\Diamond p)$.

As in Section 4.1, we define the value of φ for given $Relax(\varphi) = (\psi_1, \ldots, \psi_m)$ to be $val(\mathcal{T}, \varphi) = (v_1, \ldots, v_m)$, where $v_k = 1$ if $\mathcal{T} \models \psi_{(m+1)-k}$, and $v_k = 0$ otherwise. For a conjunction $\varphi_1 \wedge \ldots \wedge \varphi_n$ of soft specifications with given $Relax(\varphi_j) = (\psi_{j,1}, \ldots, \psi_{j,m})$ for each $j \in \{1, \ldots, n\}$, we define the value $val(\mathcal{T}, \varphi_1 \wedge \ldots \wedge \varphi_n) = \big(\sum_{i=1}^{n} v_{i,1}, \ldots, \sum_{i=1}^{n} v_{i,m}\big)$, where $val(\mathcal{T}, \varphi_i) = (v_{i,1}, \ldots, v_{i,m})$ for $i \in \{1, \ldots, n\}$.

The maximum realizability problem asks for a given LTL specification φ and soft LTL specifications $\varphi_1, \ldots, \varphi_n$ with given $Relax(\varphi_j) = (\psi_{j,1}, \ldots, \psi_{j,m})$, to determine whether there exists a transition system $\mathcal{T}$ such that $\mathcal{T} \models \varphi$, and if the answer is positive, to construct a transition system $\mathcal{T}^*$ such that $\mathcal{T}^* \models \varphi$, and for every $\mathcal{T}$ with $\mathcal{T} \models \varphi$, it holds that $val(\mathcal{T}, \varphi_1 \wedge \ldots \wedge \varphi_n) \leq val(\mathcal{T}^*, \varphi_1 \wedge \ldots \wedge \varphi_n)$. The bounded maximum realizability problem is defined in the straightforward way.

We can adapt the MaxSAT approach from Section 5.3 to solve the bounded maximum realizability problem in this setting as follows.

First, for each $\psi_{j,k}$ in $Relax(\varphi_j)$, we construct a universal co-Büchi automaton $\mathcal{A}_{j,k} = (Q_{j,k}, q_0^{j,k}, \delta_{j,k}, F_{j,k})$ such that $\mathcal{T} \in \mathcal{L}(\mathcal{A}_{j,k})$ if and only if $\mathcal{T} \models \psi_{j,k}$. In the MaxSAT encoding, the hard constraints for the annotation $\lambda_{j,k}$ are

$$C_{j,k} \stackrel{\text{def}}{=} \bigwedge_{q,q' \in Q_{j,k}} \bigwedge_{s,s' \in S} \bigwedge_{\sigma_I \in 2^{\mathcal{I}}} \left(\left(\lambda_{s,q}^{\mathbb{B},j,k} \wedge \delta_{s,q,\sigma_I,q'}^{j,k} \wedge \tau_{s,\sigma_I,s'}\right) \rightarrow \mathsf{succ}_{\lambda}^{j,k}(s, q, s', q', \sigma_I) \right).$$

Generalizing the encoding, for each $j \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, m\}$, we now have one soft constraint $Soft_{j,k} \stackrel{\text{def}}{=} \bigvee_{l=1}^{k} \lambda_{s_0,q^{j,l}}^{\mathbb{B},j,l}$ with weight n^{k-1}.

5.4.2 Maximum Realizability with Priorities

In the definitions in Section 4.2 and the paragraph above, all soft specifications have the same priority. Now, we extend the maximum realizability setting to the case with priorities for the soft specifications given as part of the input to the problem.

Soft specifications with priority ordering. We begin with a simple setting where soft specifications are simply ordered in decreasing priority, without assigning any numerical weight for the preferences over the formulas. More specifically, we assume that the soft specifications $\varphi_1, \ldots, \varphi_n$ are ordered such that, for every $i \in \{1, \ldots, n\}$, we have that φ_i has higher priority than φ_j for all $j > i$.

Now, given a vector $Relax(\varphi_j) = (\psi_{j,1}, \ldots, \psi_{j,m})$ for φ_j we define the value of φ_j in a transition system $\mathcal{T}$ to be the number of specifications in $Relax(\varphi_j)$ satisfied by $\mathcal{T}$,

i.e., $val(\mathcal{T}, \varphi_j) = |\{k \in \{1, \ldots, m\} \mid \mathcal{T} \models \psi_{j,k}\}|$. The value of $\varphi_1 \wedge \ldots \wedge \varphi_n$ is then defined as $val(\mathcal{T}, \varphi_1 \wedge \ldots \wedge \varphi_n) = (val(\mathcal{T}, \varphi_1), \ldots, val(\mathcal{T}, \varphi_n))$. The values of transition systems are compared according to the lexicographic ordering of vectors in $\{0, \ldots, m\}^n$, thus giving priority to φ_i over φ_j for $i < j$.

The MaxSAT approach can be adapted for this value function in the same way as above. The difference is in the weights of the soft constraints for the annotations $\lambda_{j,k}$: for each $j \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, m\}$ we have a soft constraint $Soft_{j,k} \stackrel{\text{def}}{=} \bigvee_{l=1}^{k} \lambda_{s_0,q^{j,l}}^{\mathbb{B},j,l}$ with weight $w_{j,k}$, where $w_{j,k} = 1$ if $j = n$ or $k < m$, and $w_{j,k} = \sum_{j'=j+1}^{n} \sum_{k=1}^{m} w_{j',k} + 1$ otherwise.

Soft specifications with given weights. We also consider the weighted maximum realizability problem, in which, together with $Relax(\varphi)$ for each soft specification φ, the user also provides numerical weights for the formulas in $Relax(\varphi)$. That is, for each $j \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, m\}$, we are given a weight $w_{j,k}$ for $\psi_{j,k}$. These weights specify the priority of each of the soft specifications.

The MaxSAT formulation is then adapted to incorporate the given weights, by using them for the corresponding soft constraints. Namely, for each j and k, the corresponding soft constraint $Soft_{j,k} \stackrel{\text{def}}{=} \bigvee_{l=1}^{k} \lambda_{s_0,q^{j,l}}^{\mathbb{B},j,l}$ has weight $w_{j,k}$.

Theorem 3 *Given an LTL specification φ and soft specifications $\square\varphi_1, \ldots, \square\varphi_n$ together with a vector of formulas $Relax(\varphi_j) = (\psi_{j,1}, \ldots, \psi_{j,m})$ for each φ_j, if there is a transition system $\mathcal{T}$ with $\mathcal{T} \models \varphi$, then there exists $\mathcal{T}^*$ such that:*

- *$val(\mathcal{T}, \varphi_1 \wedge \ldots \wedge \varphi_n) \leq val(\mathcal{T}^*, \varphi_1 \wedge \ldots \wedge \varphi_n)$ for all $\mathcal{T}$ with $\mathcal{T} \models \varphi$, and*
- *$\mathcal{T}^* \models \varphi$ and $|\mathcal{T}^*| \leq (2^{b+\log b})!^2$,*

where $b = \max\{|\mathsf{subf}(\varphi \wedge \varphi'_1 \wedge \ldots \wedge \varphi'_n)| \mid \varphi'_i \in Relax(\varphi_i)$ for $i = 1, \ldots, n\}$.

Proof The proof is a generalization of the proof of Theorem 1. First, we need to establish the analogue of Lemma 1 for the general case.

Lemma 3 *For every transition system $\mathcal{T}$, soft specifications $\square\varphi_1, \ldots, \square\varphi_n$, and vector of formulas $Relax(\varphi_j) = (\psi_{j,1}, \ldots, \psi_{j,m})$ for each φ_j, if $val(\mathcal{T}, \square\varphi_1 \wedge \ldots \wedge \square\varphi_n) = v$, then there exists an LTL formula ψ_v such that $\mathcal{T} \models \psi_v$ and the following holds:*

(1) $\psi_v = \varphi'_1 \wedge \ldots \wedge \varphi'_n$, where $\varphi'_i \in Relax(\varphi_i) \cup \{true\}$ for $i = 1, \ldots, n$,
(2) for every $\mathcal{T}'$, if $\mathcal{T}' \models \psi_v$, then $val(\mathcal{T}', \square\varphi_1 \wedge \ldots \wedge \square\varphi_n) \geq v$.

The proof of Lemma 3 is analogous to the proof of Lemma 1. Then, with the help of Lemma 3 we can establish Theorem 3 in the same way as Theorem 1. □

6 Experimental Evaluation

We implemented the proposed approach to maximum realizability[1] in Python 2.7. For the LTL to automata translation Spot [30] version 2.2.4 is used. MaxSAT instances are solved by Open-WBO [31] version 2.0. We evaluated our method on instances of two examples. Each experiment was run on a machine with a 2.3 GHz Intel Xeon E5-2686 v4 processor and 16 GiB of memory. While the processor is quad-core, only a single core was used. We set a time-out of 1 hour.

[1] The code is available at https://github.com/MahsaGhasemi/max-realizability

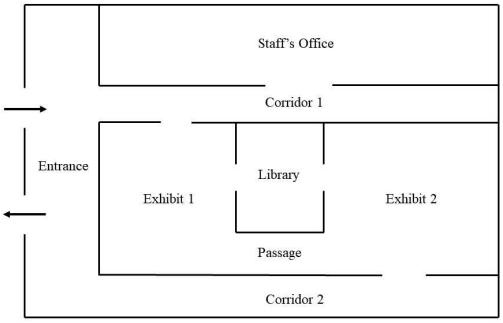


Figure 3: Map of the museum.

6.1 Robotic Navigation.

We applied our method to the strategy synthesis for a robotic museum guide. The map of the museum is shown in Figure 3. The robot has to give a tour of the exhibitions in a specific order, which constitutes the hard specification. The tour starts at the entrance of the museum where the robot picks up newly arrived visitors. The main objective is to take the group through the two exhibitions on that floor and then return to the entrance to pick up a new group of people. Preferably, it also avoids certain locations, such as the library, or the passage when it is occupied. These preferences are encoded in the soft specifications. In particular, on one hand, the robot can only gain access to Exhibition 2 by getting a key from the staff's office. On the other hand, the robot is asked not to disturb the employees in the office. There is a library between Exhibition 1 and Exhibition 2 which can be used to go from one to the other, but it is preferred that visitors do not enter the library. However, it is also desirable that when the other passage between these two exhibitions is occupied, the robot does not go through there.

Clearly, these specifications cannot be realized in conjunction. Given their priorities, we categorize the requirements into hard and soft specifications, and synthesize a strategy which satisfies the hard specifications and maximizes the satisfaction of the soft specifications. We formalize the problem as follows.

Input propositions: The set $\mathcal{I}$ contains a single Boolean variable $occupied$ that indicates whether the passage between the two exhibitions is occupied.

Output propositions: The set of output propositions $\mathcal{O}$ consists of eight Boolean variables corresponding to the eight locations on the map: $entrance$, $corridor_1$, $corridor_2$, $exhibition_1$, $exhibition_2$, $passage$, $office$, $library$.

The hard specification is the conjunction of the following formulas.

– The robot starts at the entrance:

$$entrance.$$

– At each time step, the robot can occupy only one location:

$$\square \bigwedge_{o_1 \in \mathcal{O}} \left(o_1 \rightarrow \bigwedge_{o_2 \in \mathcal{O} \setminus \{o_1\}} \neg o_2 \right).$$

– The admissible actions of the robot are to stay in the current location or move to an adjacent one. This leads to eight requirements describing the map. For instance:

$$\square(corridor_1 \rightarrow \bigcirc(corridor_1 \vee office \vee exhibition_1)).$$

Remark: Due to the requirements above, the robot will always be in exactly one valid location, i.e., in a transition system that satisfies the specifications it is impossible to reach a state where all output variables are false.

– The robot must infinitely often visit both exhibitions:

$$\square\diamondsuit\, exhibition_1,$$
$$\square\diamondsuit\, exhibition_2.$$

– The robot has to respect the order of visits, by starting from Exhibition 1, going to Exhibition 2 and finishing at the entrance:

$$\square(exhibition_1 \rightarrow \bigcirc((\neg entrance \wedge \neg exhibition_1)\,\mathcal{U}\, exhibition_2)),$$
$$\square(exhibition_2 \rightarrow \bigcirc((\neg exhibition_1 \wedge \neg exhibition_2)\,\mathcal{U}\, entrance)),$$
$$\square(entrance \rightarrow \bigcirc((\neg exhibition_2 \wedge \neg entrance)\,\mathcal{U}\, exhibition_1)).$$

– The robot does not have access to Exhibition 2 before it visits the office:

$$\neg exhibition_2\,\mathcal{U}\, office.$$

The set of soft specifications describes the desirable requirements that the robot does not enter the office, the library, or a occupied passage. Formally:

– The robot must not enter the office from corridor 1:

$$\square(corridor_1 \rightarrow \bigcirc \neg office).$$

– The robot must not enter the library from the exhibitions:

$$\square(exhibition_1 \vee exhibition_2 \rightarrow \bigcirc \neg library).$$

– The robot must not enter the passage from the exhibitions when it is occupied:

$$\square((exhibition_1 \vee exhibition_2) \wedge \bigcirc occupied \rightarrow \bigcirc \neg passage).$$

We applied the proposed method described in Section 5 on this example. Table 1 summarizes the results. With implementation bound of 8, the hard specification is realizable and a partial satisfaction of soft specifications is achieved. This strategy always selects the passage to transition from Exhibition 1 to Exhibition 2 and hence, avoids the library. It also violates the requirement of not entering the staff's office, to acquire access to Exhibition 2. For implementation bound 10 the solver times out. Notice that strategies with higher values exists, however, they require larger implementation size.

Table 1: Results of applying synthesis with maximum realizability to the robotic navigation example, with different bounds on implementation size $|\mathcal{T}|$. We report on the number of variables and clauses in the encoding, the satisfiability of hard constraints, the value (and bound) of the MaxSAT objective function, the running times of Spot, Open-WBO, and the time of the solver plus the time for generating the encoding.

	Encoding		Solution		Time (s)		
$\|\mathcal{T}\|$	# vars	# clauses	sat.	$\Sigma weights$	Spot	Open-WBO	enc.+solve
2	4051	25366	UNSAT	0 (39)	0.93	0.011	0.12
4	19965	125224	UNSAT	0 (39)	0.93	0.079	0.57
6	45897	289798	UNSAT	0 (39)	0.93	1.75	2.9
8	95617	596430	SAT	31 (39)	0.93	956	959
10	152949	954532	SAT	- (39)	0.93	time-out	time-out

6.2 Power Distribution Network.

We consider the problem of dynamic reconfiguration of power distribution networks. A power network consists of a set P of power supplies (generators) and a set L of loads (consumers). The network is a bipartite graph with edges between supplies and loads, where each supply is connected to multiple loads and each load is connected to multiple supplies. Each power supply has an associated capacity, which determines how many loads it can power at a given time. For each supply $p \in P$, we denote with E_p^+ the capacity of p, that is, how many loads p can power, and with $Consumers(p)$ the set of loads connected to p in the network graph. Similarly, for a load $l \in L$, let $Suppliers(l)$ be the set of suppliers to which the load l is connected in the network. It is possible that not all loads can be powered all the time. Some loads are critical and must be powered continuously (hard specification), while others are not and should be powered when possible (soft specification). Some loads can be initializing, meaning they must be powered only initially for several steps. Power supplies can become faulty during operation, which necessitates dynamic network reconfiguration. The number of supplies that can be simultaneously faulty is upper bounded by a constant f. Further, we add a soft specification to limit the frequency of switching the relays between power supplies and loads. We model the problem as follows.

Input propositions: The set $\mathcal{I}$ consists of input propositions which form the binary encoding of f integer variables $e_1, e_2, \ldots, e_f$ each with domain $\{0, \ldots, |L|\}$. The values of these variables indicate which power supplies are faulty at a given point in time: if $e_i = p$ for some i and p, then p is faulty at that time.[2]

Output propositions: The set of output propositions $\mathcal{O}$ consists of the Boolean variables $s_{l \to p}$ for all loads $l \in L$ and supplies $p \in P$ where l and p are connected. The meaning of $s_{l \to p}$ being true is that l is powered by p.

The hard specification is the conjunction of the following formulas.

- A critical load must always be powered:

$$\square \left(\bigvee_{p \in Suppliers(l)} s_{l \to p} \right), \quad \forall\, l \in L \text{ where } l \text{ is critical.}$$

[2] There can be different indices i for which $e_i = p$ at the same time. While this will be redundant, it does not affect the encoding.

– An initializing node must be powered during the first two steps:

$$\left(\bigvee_{p\in Suppliers(l)} s_{l\to p}\right) \wedge \bigcirc\left(\bigvee_{p\in Suppliers(l)} s_{l\to p}\right), \quad \forall\, l \in L \text{ where } l \text{ is initializing.}$$

– A load must only be assigned to one power supply:

$$\bigwedge_{l\in L}\ \bigwedge_{p_1\in Suppliers(l)} \Box\left(s_{l\to p_1} \to \bigwedge_{p_2\in Suppliers(l), p_2\neq p_1} \neg s_{l\to p_2}\right).$$

– The capacity of power supplies must not be exceeded:

$$\bigwedge_{p\in P}\ \bigwedge_{\substack{L'\subseteq Consumers(p)\\ |L'|=E_p^+}} \Box\left(\left(\bigwedge_{l\in L'} s_{l\to p}\right) \to \bigwedge_{l\in Consumers(p)\setminus L'} \neg s_{l\to p}\right).$$

– When a power supply becomes faulty, no loads can be powered by it:

$$\bigwedge_{i\in\{1,2,\ldots f\}}\ \bigwedge_{p\in P} \Box\left(e_i = p \to \bigwedge_{l\in Consumers(p)} \neg s_{l\to p}\right).$$

The set of soft specifications consists of the requirements for powering the non-vital loads, and optionally, a restriction on switching supplies too often unless they become faulty. The respective formulas are given below.

– A non-critical load should always be powered:

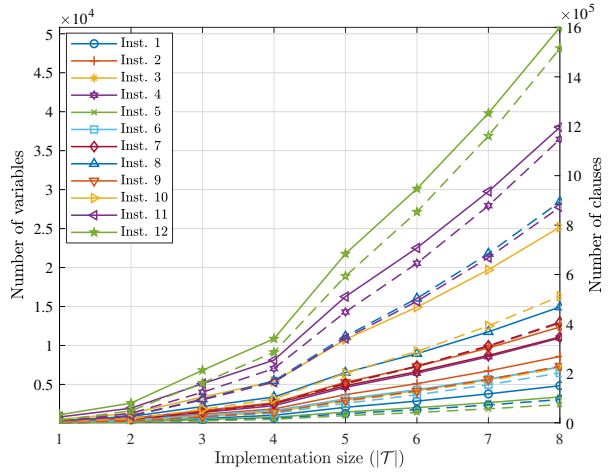
$$\Box\left(\bigvee_{p\in Suppliers(l)} s_{l\to p}\right), \quad \forall\, l \in L \text{ where } l \text{ is non-critical.}$$

– All loads powered by a supply remain powered by it unless it becomes faulty:

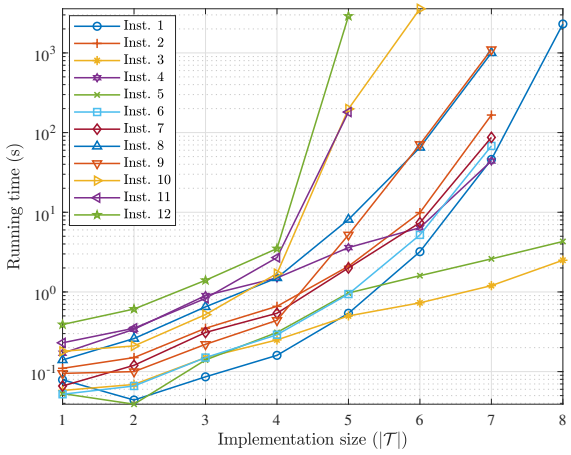
$$\bigwedge_{l\in L}\ \bigwedge_{p\in Suppliers(l)} \Box\left(s_{l\to p} \wedge \bigcirc\left(\neg \bigvee_{i\in\{1,2,\ldots,f\}} e_i = l\right) \to \bigcirc s_{l\to p}\right).$$

We applied our method to the problem of synthesizing a relay-switching strategy from the above LTL specifications. Table 2 describes the instances to which we applied our synthesis method. Power supplies have the same capacity E^+ (number of loads they can power) and at most one can be faulty at each time. We consider three categories of instances, depending on the network connectivity (full or sparse), and whether we restrict frequent switching of supplies.

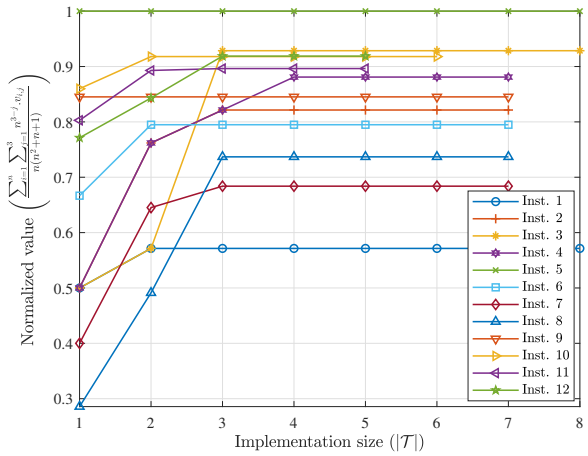
In Figure 4, we show the results for the instances defined in Table 2 (detailed results are reported in Table 3). As expected, the value function is monotonically nondecreasing with respect to the bound on the implementation size. In the first set of instances, the specifications have large number of variables (due to full connectivity), and the bottleneck is the translation to automata. In the third set of instances, the limiting factor is the number of soft specifications, leading to large weights and number of variables in the MaxSAT formulation. We observe that the number of soft specifications is an important factor affecting the scalability of the proposed method. For example, Instance 12, on which the MaxSAT solver reaches time-out for implementation size bound 6 contains 23 soft specifications.



(a) Encoding size



(b) Running time



(c) Normalized value

Figure 4: Results of applying our method to the instances in Table 2, with different bounds on implementation size $|\mathcal{T}|$. (a) shows the size of the MaxSAT encoding as the number of variables (solid lines) and the number of clauses (dashed lines). (b) shows the running time of the MaxSAT solver plus the time for the encoding. (c) shows the level of realizability of soft specifications.

Table 2: Power distribution network instances. An instance is determined by the number supplies $|P|$, the number of loads $|L|$, the capacity of supplies E^+, the number of critical, non-critical and initializing loads. We also show the number of input $|\mathcal{I}|$ and output $|\mathcal{O}|$ propositions and the number of soft specifications.

		Network			Load characterization			Specifications		
	Instance #	$\|P\|$	$\|L\|$	E^+	crit.	non-crit.	init.	$\|\mathcal{I}\|$	$\|\mathcal{O}\|$	# Soft spec.
fully connected, switching allowed	1	3	3	1	1	2	0	2	9	2
	2	3	6	2	2	4	0	2	18	4
	3	3	3	1	0	2	1	2	9	2
	4	3	6	2	1	4	1	2	18	4
sparse, switching allowed	5	4	2	1	1	1	0	3	4	1
	6	4	4	1	1	3	0	3	8	3
	7	4	6	1	1	5	0	3	12	5
	8	4	8	1	1	7	0	3	16	7
sparse, switching restricted	9	4	2	1	1	1	0	3	4	5
	10	4	4	1	1	3	0	3	8	11
	11	4	6	1	1	5	0	3	12	17
	12	4	8	1	1	7	0	3	16	23

7 Conclusion and Future Work

In this paper, we considered settings in which a system's requirements are categorized as hard and soft linear temporal logic (LTL) specifications and the goal is to design a controller that satisfies the hard specifications while maximally realizing the soft specifications. To that end, we introduced relaxations of soft LTL formulas and accordingly defined a value function that captures the level of realizing a conjunction of soft LTL formulas. We further constructed a MaxSAT encoding of maximum realizability that aims to maximize the value function. By incrementing the size of the implementation and generating the induced MaxSAT encodings, we developed a bounded synthesis procedure to find a controller with smallest size that meets a termination criterion. We computed a theoretical bound on the size of the implementation and proved soundness and completeness of the synthesis algorithm. Additionally, we discussed multiple generalizations of the proposed method and provided experimental results in multiple scenarios.

As part of future work, we plan to employ the proposed algorithm to construct controllers from a combination of temporal logic specifications and data in the form of sample demonstration of desired system behavior. In such settings, the system is asked to imitate the sample demonstrations as much as possible while satisfying the given specifications. We are also considering to design a customized search procedure for solving MaxSAT instances generated for maximum realizability that benefits from the knowledge on the specific structure of soft clauses.

Table 3: Results of applying synthesis with maximum realizability on the instances in Table 2, with different bounds on implementation size $|\mathcal{T}|$. We report on the number of variables and clauses in the encoding, the value (and bound) of the objective function in the MaxSAT instance, the running times of Spot, Open-WBO, and the time of the solver plus the time for generating the encoding.

Instance #	$\|\mathcal{T}\|$	Encoding		Solution	Time (s)		
		# vars	# clauses	$\Sigma weights$	Spot	Open-WBO	enc.+solve
1	2	246	3183	8 (14)	0.23	0.0050	0.044
	4	1038	18059	8 (14)	0.23	0.046	0.16
	6	2862	52999	8 (14)	0.23	2.9	3.2
	8	4838	94079	8 (14)	0.23	2305	2306
2	2	452	13429	64 (84)	91	0.015	0.15
	4	1860	77125	69 (84)	91	0.15	0.66
	6	5100	226933	69 (84)	91	8.5	9.9
	8	8588	403165	N/A (84)	91	time-out	time-out
3	2	302	7567	8 (14)	0.23	0.0067	0.069
	4	1446	42891	13 (14)	0.23	0.021	0.25
	6	4206	125287	13 (14)	0.23	0.11	0.73
	8	7206	222591	13 (14)	0.23	1.1	2.5
4	2	508	38165	64 (84)	210	0.028	0.34
	4	2268	219589	74 (84)	210	0.21	1.5
	6	6444	645397	74 (84)	210	2.7	6.4
	8	10956	1147101	N/A (84)	210	time-out	time-out
5	2	203	2476	3 (3)	0.058	0.017	0.039
	4	779	14126	3 (3)	0.058	0.24	0.31
	6	2019	41584	3 (3)	0.058	1.4	1.6
	8	3395	73808	3 (3)	0.058	4.0	4.3
6	2	433	6722	31 (39)	0.17	0.0069	0.066
	4	1649	38472	31 (39)	0.17	0.076	0.29
	6	4329	113422	31 (39)	0.17	4.6	5.2
	8	7241	201326	N/A (39)	0.17	time-out	time-out
7	2	663	13464	100 (155)	1.3	0.011	0.12
	4	2519	77538	106 (155)	1.3	0.11	0.54
	6	6639	229036	106 (155)	1.3	6.3	7.4
	8	11087	406668	N/A (155)	1.3	time-out	time-out
8	2	893	29070	196 (399)	30	0.019	0.26
	4	3389	169596	294 (399)	30	0.47	1.5
	6	8949	503338	294 (399)	30	62	65
	8	14933	894122	N/A (399)	30	time-out	time-out
9	2	631	7352	131 (155)	0.21	0.0069	0.10
	4	2647	43106	131 (155)	0.21	0.15	0.44
	6	7311	129100	131 (155)	0.21	71	71
	8	12367	229004	N/A (155)	0.21	time-out	time-out
10	2	1289	16474	1343 (1463)	0.44	0.012	0.21
	4	5385	96432	1343 (1463)	0.44	1.1	1.7
	6	14913	288454	1343 (1463)	0.44	3579	3581
	8	25185	511718	N/A (1463)	0.44	time-out	time-out
11	2	1947	28092	4660 (5219)	1.9	0.021	0.35
	4	8123	164478	4678 (5219)	1.9	1.7	2.7
	6	22515	491584	N/A (5219)	1.9	time-out	time-out
	8	38003	872256	N/A (5219)	1.9	time-out	time-out
12	2	2605	48574	10724 (12719)	28	0.056	0.61
	4	10861	285516	11686 (12719)	28	1.7	3.5
	6	30117	853402	N/A (12719)	28	time-out	time-out
	8	50821	1514906	N/A (12719)	28	time-out	time-out

Acknowledgements This work was supported in part by AFRL grants UTC 17-S8401-10-C1 and FA8650-15-C-2546, and ONR grant N000141613165.

References

1. R. Dimitrova, M. Ghasemi, and U. Topcu, "Maximum realizability for linear temporal logic specifications," in *Proc. Automated Technology for Verification and Analysis*, pp. 458–475, Springer, 2018.
2. A. Pnueli, "The temporal logic of programs," in *Annual Symposium on Foundations of Computer Science*, pp. 46–57, IEEE, 1977.
3. S. Schewe and B. Finkbeiner, "Bounded synthesis," in *Proc. Automated Technology for Verification and Analysis*, vol. 4762 of *LNCS*, pp. 474–488, 2007.
4. P. Faymonville, B. Finkbeiner, M. N. Rabe, and L. Tentrup, "Encodings of bounded synthesis," in *Proc. International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, vol. 10205 of *LNCS*, pp. 354–370, 2017.
5. A. Biere, M. Heule, and H. van Maaren, *Handbook of satisfiability*, vol. 185. IOS press, 2009.
6. J. D. Park, "Using weighted MAX-SAT engines to solve MPE," in *Proc. American Association for Artificial Intelligence*, pp. 682–687, 2002.
7. M. Janota, I. Lynce, V. Manquinho, and J. Marques-Silva, "PackUp: Tools for package upgradability solving," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 8, pp. 89–94, 2012.
8. J. Berg, A. Hyttinen, and M. Järvisalo, "Applications of MaxSAT in data analysis," *Pragmatics of SAT*, 2015.
9. J. Tumova, G. C. Hall, S. Karaman, E. Frazzoli, and D. Rus, "Least-violating control strategy synthesis with safety rules," in *Proc. ACM International Conference on Hybrid Systems: Computation and Control*, 2013.
10. K. Kim, G. E. Fainekos, and S. Sankaranarayanan, "On the minimal revision problem of specification automata," *International Journal of Robotics Research*, vol. 34, no. 12, 2015.
11. M. Lahijanian, S. Almagor, D. Fried, L. E. Kavraki, and M. Y. Vardi, "This time the robot settles for a cost: A quantitative approach to temporal logic planning with partial satisfaction," in *Proc. Association for the Advancement of Artificial Intelligence*, 2015.
12. M. Lahijanian and M. Z. Kwiatkowska, "Specification revision for Markov decision processes with optimal trade-off," in *Proc. IEEE Conference on Decision and Control*, pp. 7411–7418, 2016.
13. M. Lahijanian, M. R. Maly, D. Fried, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal planning in uncertain environments with partial satisfaction guarantees," *IEEE Trans. Robotics*, vol. 32, no. 3, pp. 583–599, 2016.
14. T. Tomita, A. Ueno, M. Shimakawa, S. Hagihara, and N. Yonezaki, "Safraless LTL synthesis considering maximal realizability," *Acta Informatica*, vol. 54, no. 7, 2017.
15. F. Juma, E. I. Hsu, and S. A. McIlraith, "Preference-based planning via MaxSAT," in *Proc. Advances in Artificial Intelligence*, vol. 7310 of *LNCS*, pp. 109–120, 2012.
16. N. Robinson, C. Gretton, D. N. Pham, and A. Sattar, "Partial weighted MaxSAT for optimal planning," in *Proc. Pacific rim international conference on artificial intelligence*, pp. 231–243, Springer, 2010.
17. R. Bloem, K. Chatterjee, T. A. Henzinger, and B. Jobstmann, "Better quality in synthesis through quantitative objectives," in *Proc. International Conference on Computer-Aided Verification*, vol. 5643 of *LNCS*, pp. 140–156, 2009.
18. S. Almagor, U. Boker, and O. Kupferman, "Formally reasoning about quality," *Journal of the ACM*, vol. 63, no. 3, pp. 24:1–24:56, 2016.
19. P. Tabuada and D. Neider, "Robust linear temporal logic," in *Proc. Computer Science Logic*, vol. 62 of *LIPIcs*, pp. 10:1–10:21, 2016.
20. R. Alur, A. Kanade, and G. Weiss, "Ranking automata and games for prioritized requirements," in *Proc. International Conference on Computer-Aided Verification*, vol. 5123 of *LNCS*, 2008.
21. A. Cimatti, M. Roveri, V. Schuppan, and S. Tonetta, "Boolean abstraction for temporal logic satisfiability," in *Proc. International Conference on Computer-Aided Verification*, vol. 4590 of *LNCS*, pp. 532–546, 2007.
22. V. Schuppan, "Towards a notion of unsatisfiable and unrealizable cores for LTL," *Science of Computer Programming*, vol. 77, no. 7-8, pp. 908–939, 2012.
23. V. Raman and H. Kress-Gazit, "Towards minimal explanations of unsynthesizability for high-level robot behaviors," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 757–762, 2013.
24. A. Cimatti, M. Roveri, V. Schuppan, and A. Tchaltsev, "Diagnostic information for realizability," in *Proc. International Conference on Verification, Model Checking, and Abstract Interpretation*, LNCS, 2008.

25. R. Ehlers and V. Raman, "Low-effort specification debugging and analysis," in *Proc. Workshop on Synthesis*, vol. 157 of *EPTCS*, pp. 117–133, 2014.
26. C. Baier and J. Katoen, *Principles of model checking*. 2008.
27. O. Kupferman and M. Y. Vardi, "Safraless decision procedures," in *Proc. IEEE Annual Symposium on Foundations of Computer Science*, pp. 531–542, 2005.
28. B. Finkbeiner and S. Schewe, "Bounded synthesis," *International Journal on Software Tools for Technology Transfer*, vol. 15, no. 5-6, 2013.
29. O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Methods in System Design*, vol. 19, no. 3, pp. 291–314, 2001.
30. A. Duret-Lutz, A. Lewkowicz, A. Fauchille, T. Michaud, E. Renault, and L. Xu, "Spot 2.0 - A framework for LTL and ω -automata manipulation," in *Proc. Automated Technology for Verification and Analysis*, vol. 9938 of *LNCS*, 2016.
31. R. Martins, V. M. Manquinho, and I. Lynce, "Open-WBO: A modular MaxSAT solver,," in *Proc. SAT'14*, vol. 8561 of *LNCS*, pp. 438–445, 2014.