An SMT-based approach to Weak Controllability for Disjunctive Temporal Problems with Uncertainty☆

Alessandro Cimatti[a], Andrea Micheli[a,b], Marco Roveri[a]

[a] *Fondazione Bruno Kessler, Istituto per la Ricerca Scientifica e Tecnologica*
Via Sommarive 18, 38123 Povo, Trento, Italy
[b] *Doctoral School in Information and Communication Technology, University of Trento*
Via Sommarive 9, 38123 Povo, Trento, Italy

Abstract

The framework of temporal problems with uncertainty (TPU) is useful to express temporal constraints over a set of activities subject to uncertain (and uncontrollable) duration. In this work, we focus on the most general class of TPU, namely disjunctive TPU (DTPU), and consider the case of weak controllability, that allows one to model problems arising in practical scenarios (e.g. on-line scheduling).

We first tackle the decision problem, i.e. whether there exists a schedule of the activities that, depending on the uncertainty, satisfies all the constraints. We propose a logical approach, based on the reduction to a problem of Satisfiability Modulo Theories (SMT), in the theory of Linear Real Arithmetic with Quantifiers. This results in the first implemented solver for weak controllability of DTPUs.

Then, we tackle the problem of synthesizing control strategies for scheduling the activities. We focus on strategies that are amenable for efficient execution. We prove that linear strategies are not always sufficient, even in the sub-case of simple TPU (STPU), while piecewise-linear strategies, that are multiple conditionally-applied linear strategies, are always sufficient. We present several algorithms for the synthesis of linear and piecewise-linear strategies, in case of STPU and of DTPU.

All the algorithms are implemented on top of SMT solvers. We provide experimental evidence of the scalability of the proposed techniques, with dramatic speed-ups in strategy execution compared to on-line reasoning.

Keywords: Weak Controllability, Temporal Problems, Satisfiability Modulo Theory, Strategy Synthesis

1. Introduction

Many practical settings, such as planning and scheduling, require the solution of sets of constraints over time points, that typically represent the time at which activities

☆This paper is an extended version of [10] published at AAAI 2012, Toronto, Canada.

Email addresses: cimatti@fbk.eu (Alessandro Cimatti), amicheli@fbk.eu (Andrea Micheli), roveri@fbk.eu (Marco Roveri)

begin and end. For example, constraints may represent a bound on the overall time span, or lower/upper bounds on the distance between two activities.

The Temporal Problem (TP) [18, 36] is a well studied formalism to model such temporal constraints. In the basic form of TP, also referred to as TP without uncertainty, the durations of activities are assumed to be controllable by the executor. This means that the executor assumes to have the possibility of choosing any duration that it may want. A solution is an assignment to all the time points (i.e., the beginning time and the end time of the activities), that satisfies the constraints. Depending on the structure of the constraints, TPs range from simple temporal problems (STP) [18], to temporal constraint satisfaction problems (TCSP) [18], to disjunctive temporal problems (DTP) [36].

In practice, activities may also have uncertain (and uncontrollable) durations. For example, it may be impossible to know precisely the time taken by a drilling or locomotion procedure; yet, the production of an overall schedule must be able to take this uncertainty into account. The formal framework of TPs has been extended with uncertainty (TPU), thus obtaining STPU, TCSPU, and DTPU [38, 32, 37]. Because of uncertainty, TPUs are much more complicated than TPs without uncertainty. In fact, they can be thought in terms of games, where the scheduler/executor must play against an "adversarial" environment. Intuitively, the variables representing the time points are separated into controllable ones (that are existentially quantified), and uncontrollable ones (universally quantified).

Within this setting, several degrees of solution have been identified for TPUs [38]. In *strong controllability*, a solution is a fixed, unconditioned assignment to each controllable time point, that will satisfy the constraints regardless of the uncontrollable duration of the activities. This corresponds to devising a time-triggered program, where activities are started at fixed times.

In *dynamic controllability*, a solution is a strategy where the values of controllable variables may depend on the values of the uncontrollable ones, as long as they can be observed, i.e. they occur in the past. The corresponding execution must deal with branching, and may interleave the start of activities with the observation of the uncontrollable (but observable) "end of activity" events.

In this paper we focus on *weak controllability*, that is concerned with the existence of a strategy that associates values to the controllable starting points of each activity, as a function of the uncontrollable durations. The values for the uncontrollable durations are not known at the moment of solving the problem; however, the executor is given the actual value of such durations just before the execution starts.

There are several reasons for studying weak controllability. From the temporal problems perspective, weak controllability is a conceptually interesting dual of the strong controllability problem. In addition, deciding whether a given TPU is weakly controllable may serve as a pre-check for more complex problems such as dynamic controllability. In fact, weak controllability is a necessary condition for dynamic controllability [38].

From the practical standpoint, weak controllability allows for the modeling of a setting where a number of tasks is to be repeatedly executed, but with modalities that depend on some environmental parameters that become available just prior to execution. For example, an automated production line may be required to perform a set of

activities, whose duration functionally depends on the measured size of the objects to be manipulated. The duration of the activities is unknown a priori, except for an upper and lower bound, but it becomes precise once the actual objects materialize. Similarly, in a multi-core processor, the power management may dynamically control the actual clock speeds, thus affecting the duration of jobs. An on-line scheduler may be required to decide the appropriate allocation based on information that may be made available by the power management unit. Another example of application is given in the setting of remote systems (such as space exploration rovers or satellites), where the degradation due to use causes many activities to change duration over time. For example, the movement speed of many components may decrease with the age of the system. These domains share the fact that the tasks may be repeated multiple times, on platforms of limited capacity, and in conditions that can be estimated prior to execution. As such, they can be encoded as weak controllability problems.

In this paper, we tackle weak controllability for DTPUs (i.e. in its most general form), making the following contributions. First, we propose a general decision procedure for the problem of weak controllability for DTPUs. Our approach makes use of the framework of Satisfiability Modulo Theory (SMT) [4], a formal framework that allows for the analysis of problems in decidable fragments of First Order Logic. The decision procedure is based on a reduction to an SMT problem for the theory of Quantified Linear Real Arithmetic ($\mathcal{LRA}$). The encoding can be thought as working by refutation: we state the existence of an assignment to uncontrollable time points that cannot be countered by any controllable assignment. This means that the SMT problem is satisfiable if and only if the TPU is not weakly controllable. The problem can thus be directly provided to an efficient SMT solver. This approach accounts for the first implemented decision procedure for weak controllability of DTPUs.

Then, we investigate the problem of on-line strategy execution, i.e. given a weakly controllable DTPU, how to repeatedly produce a suitable schedule for the controllable time points as a function of a valuation to the uncontrollable ones. We propose an approach, referred to as *implicit strategy execution*, based on the run-time execution of a solver for TP without uncertainty: any valuation to the uncontrollable durations removes the uncertainty from the problem, and thus transforms the TPU at hand into a TP. The solver is then invoked to solve the consistency problem yielding an assignment to the controllable time points. Unfortunately, this solution imposes strong requirements on the run-time: most notably, the control platform must support the execution of a solver; in addition, at each iteration it is required to solve an NP-hard problem, i.e. a DTP (without uncertainty).

This motivates the investigation of efficient run-time execution for weakly controllable TPUs. We analyze the spectrum of *explicit strategies*, expressed in a form that does not require reasoning, and can thus be directly evaluated. We consider *linear strategies*, that are strategies in which the values for the controllable time points are a linear function of the uncontrollable ones; and *piecewise-linear strategies*, that are combinations of different linear strategies, each associated with an activation condition defined over the uncontrollable time points. Linear strategies turn out not to be expressive enough in general: we prove that even for the STPU problem class, a weakly controllable instance is not guaranteed to have a linear strategy. We also prove that piecewise-linear strategies are sufficiently expressive: a piecewise-linear strategy

is guaranteed to exist for every weakly controllable DTPU.

Finally, we address the synthesis problem: given a weakly controllable temporal problem, we algorithmically synthesize a function from an assignment to uncontrollable time points to an assignment to the controllable ones. We propose a number of algorithms for the synthesis of a strategy. We start by considering linear strategies, developing two algorithms to produce linear strategies for the STPU and DTPU cases. Then, we generalize to the case of piecewise-linear strategies, and we propose several algorithms for the STPU and DTPU cases.

All the proposed algorithms have been implemented in a tool for solving temporal problems under uncertainty. The tool is developed on top of, and fully leverages, state-of-the-art SMT solvers [17, 7]. To the best of our knowledge, this is the first implementation for weak controllability and strategy extraction. We carried out an extensive experimental evaluation on a comprehensive set of benchmarks. Our implementation, available on-line, demonstrates high scalability, and is able to automatically extract strategies of significant size. The experimental evaluation highlights a dramatic speed-up in the execution of the synthesized explicit strategies.

Structure of the paper. In Section 2, we provide some background and we define the addressed problem. In Section 3 we formally introduce TPUs and weak controllability. In Section 4 we describe our SMT-based decision procedure. In Section 5 we formalize the concept of weak strategy and we prove that linear strategies are not always present for a weakly controllable problem, but piecewise-linear strategies are. In Section 6 we analyze the problem of strategy synthesis for all the different problem classes and strategy types. In Section 7 we present an experimental evaluation of the approach. In Section 8 we summarize the most relevant work. Finally, in Section 9 we draw some conclusions and discuss future work.

2. Background

2.1. Technical Preliminaries

Our setting is standard First Order Logic [24]. The first-order signature is composed of constants, variables, function symbols, Boolean variables, and predicate symbols. A term is either a constant, a variable, or the application of a function symbol of arity n to n terms. A theory constraint (also called a theory atom) is the application of a predicate symbol of arity n to n terms. An atom is either a theory constraint or a Boolean variable. A literal is either an atom or its negation. A clause is a finite disjunction of literals. A formula is either true ($\top$), false ($\bot$), a Boolean variable, a theory constraint, the application of a propositional connective ($\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$) of arity n to n formulae, or the application of a quantifier ($\forall$, $\exists$) to an individual variable and a formula. If t_1 and t_2 are terms, and ϕ is a formula, an if-then-else (ITE) term is $ite(\phi, t_1, t_2)$. The semantics of an ITE term is the usual if-then-else semantics from programming languages. For example, the term $ite(x > y, x, y)$ where x and y are numeric variables, corresponds to the maximum between x and y. An ITE term $ite(\phi, t_1, t_2)$ occurring in a formula ψ can be rewritten by substituting each occurrence with a fresh variable v and by conjoining $(\neg\phi \vee (v = t_1)) \wedge (\phi \vee (v = t_2))$. See [23]

for a thorough discussion. We use $x, y, v, \ldots$ for variables, and $\vec{x}, \vec{y}, \vec{v}, \ldots$ for vectors of individual or Boolean variables. Terms and formulae are referred to as expressions. Formulae are denoted with $\phi, \psi, \ldots$. Let $\vec{x}$ be a vector of variables, we indicate the i-th variable in the vector with x_i. We write $\phi(x)$ to highlight the fact that x is free in ϕ, and $\phi(\vec{x})$ to highlight the fact that the free variables of ϕ are variables in $\vec{x}$. We indicate with $Q\vec{x}.\phi(\vec{x})$ the formula $Qx_1.Qx_2.\ldots.Qx_n.\phi(x_1, \ldots, x_n)$, where $Q \in \{\forall, \exists\}$.

Substitution is defined in the standard way. We write $\phi[s/v]$ for the substitution of every occurrence of the variable v in ϕ with the term s. Let $\vec{v}$ be a vector of variables and $\vec{s}$ be a vector of terms, we write $\phi[\vec{s}/\vec{v}]$ for the parallel substitution of every occurrence of v_i in ϕ with s_i. With a slight abuse of notation, if $\phi(\vec{x}, \vec{y})$ is a formula, we write $\phi(\psi(\vec{t}), \vec{y})$ as a shorthand for $\phi[\psi(\vec{t})/\vec{x}]$.

We use the standard semantic notion of interpretation and satisfiability [24]. We call *model* μ of a formula $\phi(\vec{x})$ a pair composed of an assignment that maps each variable x_i into an element of its domain and an interpretation for the non-logical symbols that satisfies the formula. A formula $\phi(\vec{x})$ is *satisfiable* if and only if it has a model. Thus, the problem of checking the satisfiability of a formula consists in determining whether there exists a model for that formula.

2.2. Satisfiability Modulo Theory

In propositional logic, the satisfiability problem is approached with enhancements of the DPLL algorithm [15]: the formula is converted into an equi-satisfiable one in Conjunctive Normal Form (CNF); then, a satisfying assignment is incrementally built, until either all the clauses are satisfied, or a conflict is found, in which case back-jumping takes place (i.e. certain assignments are undone). Keys to efficiency are heuristics for the variable selection, and learning of conflicts [30].

Given a first-order formula ψ with non-logical symbols interpreted in a decidable background theory T, *Satisfiability Modulo Theory* (SMT) [4] is the problem of deciding whether there exists a satisfying assignment to the free variables in ψ. For example, consider the formula $(x \leq y) \wedge ((x+3 = z) \vee (z \geq y))$ in the theory of real arithmetic. The theory of real arithmetic interprets the constant symbol "3" as the real number 3 and the operators $+, =, <, >, \leq, \geq$ as the usual functions and relations. The formula is satisfiable and a satisfying assignment is $\{x \mapsto 5,\ y \mapsto 6,\ z \mapsto 8\}$.

In this work we primarily concentrate on the theory of linear arithmetic over the real numbers ($\mathcal{LRA}$). A formula in $\mathcal{LRA}$ is an arbitrary Boolean combination, a universal ($\forall$) or an existential ($\exists$) quantification, of atoms in the form $\sum_i a_i x_i \bowtie c$ where $\bowtie \in \{>, <, \leq, \geq, \neq, =\}$, every x_i is a real variable and every a_i and c is a real constant. Difference logic ($\mathcal{RDL}$) is the subset of $\mathcal{LRA}$ such that atoms have the form $x_i - x_j \bowtie c$. If a formula in $\mathcal{RDL}$ is satisfiable, then there exists infinitely many models for such formula [13]. This is because in $\mathcal{RDL}$ we can only express "distances" between variables, thus the absolute value of at least one variable can always be chosen. We denote with $\mathcal{QF}\_\mathcal{LRA}$ and $\mathcal{QF}\_\mathcal{RDL}$ the quantifier-free fragments of $\mathcal{LRA}$ and $\mathcal{RDL}$, respectively. This means that a formula ϕ is in $\mathcal{QF}\_\mathcal{LRA}$ (resp. in $\mathcal{QF}\_\mathcal{RDL}$) if it is in $\mathcal{LRA}$ (resp. in $\mathcal{RDL}$) and no first-order quantifier appears in ϕ.

A conjunction of atoms in $\mathcal{LRA}$ over n variables represents a (non-necessarily closed) convex polyhedron in n dimensions: each point of the polyhedron is a model

of the formula. Similarly, an $\mathcal{LRA}$ formula represents the union of finitely many non-necessarily closed convex polyhedra. If $\phi(\vec{x})$ and $\psi(\vec{x})$ are $\mathcal{LRA}$ formulae, the formula $\phi(\vec{x}) \rightarrow \psi(\vec{x})$ geometrically corresponds to the constraint $\phi(\vec{x}) \subseteq \psi(\vec{x})$. Similarly, the conjunction of two formulae corresponds to intersection, the disjunction to union and the negation to the complement. A model μ of an $\mathcal{LRA}$ formula ϕ ($\mu \models_{\mathcal{LRA}} \phi$) corresponds to a geometric point $\vec{\mu}$ (that is the vector of values assigned by μ to each dimension) that belongs to the region represented by ϕ (that is $\vec{\mu} \in \phi$).

Given an $\mathcal{LRA}$ formula $\phi(\vec{x})$ we denote with $Atoms(\phi(\vec{x}))$ the set of its atoms ($\sum_{x_i \in \vec{x}} c_i x_i \bowtie b$, with c_i and b being rational coefficients). Given an atom $a(\vec{x}) \in Atoms(\phi(\vec{x}))$ ($a(\vec{x}) \doteq \sum_{x_i \in \vec{x}} c_i x_i \bowtie b$[1]), let $Eq(a(\vec{x})) = \sum_{x_i \in \vec{x}} c_i x_i = b$. Let $Equalities(\phi(\vec{x}))$ be the conjunction of all the equalities in a given formula, namely $Equalities(\phi(\vec{x})) \doteq \bigwedge_{a_i(\vec{x}) \in Atoms(\phi(\vec{x}))} Eq(a_i(\vec{x}))$ if $a_i(\vec{x}) = \sum_{x_i \in \vec{x}} c_i x_i \bowtie b$ with $\bowtie$ restricted to $\{\leq, \geq, =\}$.

We write $x - y \in [a, b]$ where x and y are variables and a and b are constants, meaning the formula $(x - y) \geq a \wedge (x - y) \leq b$. If a is $-\infty$ then the first conjunct is omitted and similarly, if b is ∞ then the second conjunct is omitted. In presence of constant bounds, we write the intervals with the usual open-closed notation: $[a, b]$, $[a, \infty)$ or $(-\infty, b]$ for some $a, b \in \mathbb{R}$.

A second theory of interest for this work is the theory of Equality and Uninterpreted Functions ($\mathcal{EUF}$), in which variables range over an unspecified infinite domain and function symbols are introduced. The only interpreted symbol in the theory is $=$, the equality predicate. There is no restriction on the interpretation of function symbols: the only property assumed for a function f is $\forall x. \forall y. (x = y) \rightarrow (f(x) = f(y))$. In addition, we can have formulae defined over the combination of two or more theories. For example, an atom like $x = f(y + z) + w$ combines the $+$ operator of $\mathcal{LRA}$ with function symbols of $\mathcal{EUF}$, therefore this formula is expressed in the combination of $\mathcal{EUF}$ and $\mathcal{LRA}$ (indicated $\mathcal{EUF} \cup \mathcal{LRA}$). Different techniques can be adopted to address the theory combination problem (e.g. Nelson-Oppen [31], Delayed Theory Combination [5], Model-based Theory Combination [16]).

An SMT solver [4] is a decision procedure which solves the satisfiability problem for a formula expressed in a decidable subset of First Order Logic. The most efficient implementations of SMT solvers use the so-called "lazy approach" [34]. In order to decide a formula ϕ expressed in the theory T, a SAT solver is tightly integrated with a T-solver, that is used to decide conjunctions of constraints in the theory T. The role of the SAT solver is to enumerate the truth assignments to the Boolean abstraction of the first-order formula. The Boolean abstraction has the same Boolean structure of the first-order formula, but "replaces" the predicates which contain T information with fresh Boolean variables. The Boolean abstraction of $(x \leq y) \wedge ((x+3 = z) \vee (z \geq y))$ is $P \wedge (Q \vee R)$, where P, Q, R are fresh Boolean variables. The T-solver is invoked when the SAT solver finds a satisfying assignment for the Boolean abstraction: the satisfying assignment to Boolean abstraction maps directly to a conjunction of T atoms, which the T-solver can handle. If the conjunction is satisfiable also the original formula is

[1] We use the symbol $\doteq$ to indicate a definition. We need this symbol to distinguish between the definition from the equality as a logical predicate.

satisfiable. Otherwise the T-solver returns a conflict set which identifies a reason for the unsatisfiability. Then, the negation of the conflict set is learned by the SAT solver in order to prune the search. Examples of solvers based on the "lazy approach" are MATHSAT5 [7], Z3 [17], YICES [19] and OPENSMT [6].

In order to deal with quantifiers in $\mathcal{LRA}$, many techniques have been developed and implemented in SMT solvers. Some solvers (e.g. Z3) natively support quantifiers. However, many others (e.g. MATHSAT5) cannot deal with them. For some theories of interest, it is possible to apply algorithms that remove quantifiers from any given formula in the theory. A theory T is said to admit quantifier elimination, if for every quantified formula ϕ in T, there exists a quantifier-free formula ϕ' that is logically equivalent to ϕ. It was proved that $\mathcal{LRA}$ admits quantifier elimination, and there are techniques (e.g. Fourier-Motzkin [33], Loos-Weispfenning [25, 26]) that transform any $\mathcal{LRA}$ formula containing quantifiers into an *equivalent* $\mathcal{QF\_LRA}$ formula. These techniques, at a cost that is doubly exponential in time and space in the original formula size [33, 26, 25], enable for the use of solvers with no native support for quantifiers.

3. Temporal Problems with Uncertainty

The formalism of TP is used to model temporal constraints over time-valued variables representing time points. This formalism is often used to characterize scheduling problems, where activities must be ordered in time according to some specified constraints. The variables in a TP typically represent the beginning time and the end time of activities. For example, given two activities A_1 and A_2, it is possible to state that the distance between the beginning of A_1 and the end of A_2 is less than or equal to 2. The TP formalism is expressive enough to model Allen's interval algebra [1], and also metric constraints in the form of arbitrary Boolean combination of atoms $(x - y) \in [l, u]$, where x, y are time points and $l, u \in \mathbb{R} \cup \{+\infty, -\infty\}$ [18].

Two families of TPs have been presented in literature over the years: TP without Uncertainty, and TP with Uncertainty (TPU). In the first, simply called TP, all the time points (i.e. both beginning and end points) can be decided by the scheduler/executor [18, 36]. The case of TPU represents the more complex situation where the executor is only able to decide the beginning time of activities, whereas the end of activities is to be decided by an adversarial environment [38]. In this work we focus on TPUs.

Definition 1. *A TPU is a tuple* (X_c, X_u, C_c, C_f)*, where* $X_c \doteq \{b_1, ..., b_n\}$ *is the set of* controllable time points*,* $X_u \doteq \{e_1, ..., e_m\}$ *(*$n \geq m$*) is the set of* uncontrollable time points*,* $C_c \doteq \{cc_1, ..., cc_m\}$ *is the set of* contingent constraints*, and* $C_f \doteq \{cf_1, ..., cf_h\}$ *is the set of* free constraints*.*

$$cc_i \doteq \bigvee_{j=1}^{E_i}(e_i - b_i) \in [l^c_{i,j}, u^c_{i,j}] \qquad cf_i \doteq \bigvee_{j=1}^{D_i}(v_{i,j} - w_{i,j}) \in [l^f_{i,j}, u^f_{i,j}]$$

such that: $l^f_{i,j}, u^f_{i,j} \in \mathbb{R} \cup \{+\infty, -\infty\}$*,* $l^c_{i,j}, u^c_{i,j} \in \mathbb{R}^+$*,* $l^f_{i,j} \leq u^f_{i,j}$*,* $l^c_{i,j} \leq u^c_{i,j}$*,* D_i *is the number of disjuncts for the* i*-th free constraint,* E_i *is the number of disjuncts for the* i*-th contingent constraint,* $v_{i,j}, w_{i,j} \in X_c \cup X_u$ *and* $v_{i,j} \neq w_{i,j}$*.*

Variables in X_c represent time decisions that can be controlled by the executor. Variables in X_u represent time decisions that are under the control of the environment (and are thus uncontrollable from the point of view of the executor). We use the letters b and e as mnemonics for the beginning and end of activities, respectively[2].

The constraints are separated in two sets: free constraints C_f are constraints that the executor is required to fulfill; contingent constraints C_c are the assumptions that the environment will fulfill. Intuitively, free constraints are the requirements of the problem, while contingent constraints are the assumptions under which the problem must be solved. Notice that, differently from free constraints, each contingent constraint is required to be expressed using exactly two variables (b_i and e_i) sharing the same index i. As in [38], we consider only contingent constraints involving exactly one controllable time point and one uncontrollable time point. Thus, each uncontrollable time point e_i is linked by exactly one contingent constraint to a controllable time point b_i and $|C_c| = |X_u|$. In fact, we have a number of controllable time points that is at least as big as the number of uncontrollable ones (hence $n \geq m$) because we allow for controllable time points that are not linked to an uncontrollable via a contingent constraint. This formulation is customary in the literature, and assumes a complete independence between contingent constraints: the framework can express disjunctions in the contingent constraints but assumes the independence between different uncontrollable activities.

An example of TPU is depicted in Figure 1. The example is composed of two activities A_1 and A_2. A_1 starts at time point b_1 and ends in e_1; similarly, A_2 starts at b_2 and ends in e_2. The two activities have uncontrollable duration: A_1 has duration between 0 and 3 time units, while A_2 lasts for at least 1 and at most 2 time units. We require b_1 to be scheduled before b_2 ($b_2 - b_1 \in [0, +\infty)$), b_2 before e_1 ($e_1 - b_2 \in [0, +\infty)$), e_2 to happen at most 1 time unit before e_1 ($e_1 - e_2 \in (-\infty, 1]$) and e_2 at most 2 time units after b_1 ($e_2 - b_1 \in (-\infty, 2]$).

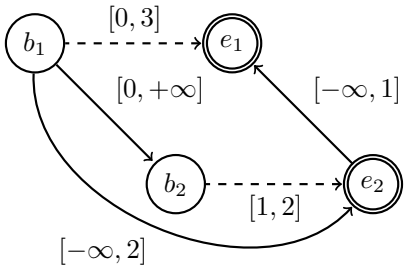
Depending on the form of the constraints in C_c and C_f, we consider three classes of TPUs. Definition 1 captures the most general class, referred to in the literature definition as *Disjunctive Temporal Problem with Uncertainty* (DTPU) [32, 37]. If each disjunction is restricted to refer to a single pair of variables, the resulting problem is a *Temporal Constraint Satisfaction Problem with Uncertainty* (TCSPU) [32, 37]. If disjunctions are disallowed, we obtain a *Simple Temporal Problem with Uncertainty* (STPU) [38, 29, 28, 21, 22]. The problem in Figure 1 is an STPU.

Following the classification of Peintner et al. [32], we also say that a problem is *simple-natured* if the contingent constraints have no disjunctions ($E_i = 1$ for each i).

We define an assignment to the time points as a total function from time points to real values. Given a TP without uncertainty, checking *consistency* corresponds to deciding the existence of an assignment that fulfills all the constraints of the problem. We call such an assignment a *consistent schedule*, and we say that the TP is *consistent*.

Given a TPU, values for controllable time points can be decided, namely they can

[2]The TPU formalism is more general, as it allows for the representation of fully controllable activities and of controllable time points not belonging to any activity. The b_i-e_i notation gives a useful intuition for uncontrollable activities, but it is possible to have the beginning and end time points of a controllable activity marked as b_i and b_j, because they both belong to X_c.



(a)

$$
\begin{aligned}
X_c &= \{b_1, b_2\} \\
X_u &= \{e_1, e_2\} \\
C_c &= \{e_1 - b_1 \in [0, 3], \\
&\qquad e_2 - b_2 \in [1, 2]\} \\
C_f &= \{b_2 - b_1 \in [0, +\infty), \\
&\qquad e_1 - e_2 \in (-\infty, 1], \\
&\qquad e_2 - b_1 \in (-\infty, 2]\}
\end{aligned}
$$

(b)

ma

Figure 1: (a) The running example of a weakly controllable STPU: nodes are time points, double-circled nodes are uncontrollable time points; contingent constraints are depicted as dashed arrows while free constraints are solid. (b) The constraint definitions for the running example.

be scheduled in time by an executor, while an uncontrollable time point e_i just happens after its activation time point b_i has been scheduled. The only assumption is that the i-th contingent constraint will be satisfied by the values of b_i and e_i. Given this intuitive meaning, we rephrased the concept of situation for a TPU [38] for the DTPU problem class.

Definition 2. *Let $P \doteq (X_c, X_u, C_c, C_f)$ be a TPU where $|X_u| = m$. The space of situations for P is $\Omega_P \doteq S_1 \times \cdots \times S_m$, where $S_i \doteq \bigcup_{j=1}^{E_i} [l^c_{i,j}, u^c_{i,j}]$. A situation is an element ω of Ω_P.*

Intuitively, a situation is a choice of the actual duration for each activity with uncontrollable duration. For example, the activity representation of the running example in Figure 2, represents a possible situation in which y_i is the duration of A_i.

For a TPU, three different problems can be addressed [38]: strong controllability, dynamic controllability and weak controllability. In all these problems, the executor is required to find a winning strategy, i.e. an assignment to the controllable time points that "works" in any situation. More specifically, the assignment to the controllable time points must fulfill all the free constraints in any situation for the given problem. The difference depends on the extent to which the executor can use (or, observe) the situation to decide the assignments to the controllable time points.

In strong controllability, the executor is "blind", i.e. it cannot observe the situation at all. Therefore, the solution to a strong controllability problem is a fixed schedule, that assigns a time value to each controllable time point. This schedule must fulfill all the free constraints in any situation. The example problem in Figure 1 (a) is not strongly controllable, as there is no way of statically assign b_2 without knowing in advance the time at which e_2 will happen. In fact, scheduling b_2 too early could violate the $e_1 - e_2 \leq 1$ constraint.

In weak controllability, the executor is allowed to act based on the situation, i.e. knowing in advance the uncontrollable durations. In this setting, the solution to a weak

controllability problem has the form of a strategy that given a situation computes an assignment to the controllable time points.

Dynamic controllability limits the observations of the executor to the past events only. This problem can be seen as a restriction of weak controllability, in which it is possible for the executor to condition the assignments to situations, as long as they have become apparent. In other works, if a choice depends on the duration of an activity, this must have already terminated.

In the rest of this paper, we focus on weak controllability, addressing the decision problem (i.e. determining whether there exists a winning strategy), and the synthesis problem (i.e. extracting such a strategy in an executable form).

We now formally define the concept of weak controllability.

Definition 3. *Let $P\doteq(X_c, X_u, C_c, C_f)$ be a TPU and let $\omega\doteq(\omega_1, \ldots, \omega_{|X_u|})$ be a situation in Ω_P. The projection P_ω of the problem P with respect to the situation ω is the TP $(X_c \cup X_u, \emptyset, \emptyset, C_f \cup C'_c)$, where C'_c is derived from C_c by replacing each contingent constraint $\bigvee_{j=1}^{E_i}(e_i - b_i) \in [l^c_{i,j}, u^c_{i,j}]$ with a free constraint $e_i - b_i \in [\omega_i, \omega_i]$.*

Intuitively, the projection P_ω is the problem without uncertainty in which each uncontrollable duration has been fixed to a given value.

Definition 4. *Let $P\doteq(X_c, X_u, C_c, C_f)$ be a TPU. P is weakly controllable if and only if for each situation $\omega \in \Omega_P$ the projection P_ω is consistent.*

Definition 4 captures the weak controllability concept by requiring the existence of a schedule for each situation. This definition implicitly models a strategy as a function $f : \Omega_P \to \mathbb{R}^{|X_c|}$ that maps each situation ω in a schedule for the controllable time points that fulfills the constraints of the projection P_ω. If such a function exists, the problem is weakly controllable.

Weak controllability is, in terms of games, the dual of strong controllability: in strong controllability, the executor is required to make its move (i.e. all its decisions) without observing the situation (i.e. the move of the environment); in weak controllability, the environment is required to make all its decisions before the executor.

Finally, we note that the constraints of a TPU are essentially $\mathcal{RDL}$ formulae. As such, if there exists a weak strategy that fulfills all the problem constraints, there are infinitely many other strategies obtained by adding the same constant value to each strategy decision. In the example of Figure 1, the value of b_1 can be arbitrarily chosen because it precedes all the other time points, and the chosen value acts only as a shift for all the other time points.

4. Deciding Weak Controllability

In order to logically define weak controllability, we first perform some manipulations on the problem definition. We encode each uncontrollable time point e_i in terms of the time difference with its starting time point b_i by means of an uncontrollable duration variable y_i. Intuitively, if we take an activity view, y_i measures the duration of the i-th activity. For every contingent constraint $cc_i = \bigvee_{j=1}^{E_i}(e_i - b_i) \in [l_{i,j}, u_{i,j}]$, let $y_i \in \mathbb{R}$ be the uncontrollable duration variable associated to e_i such

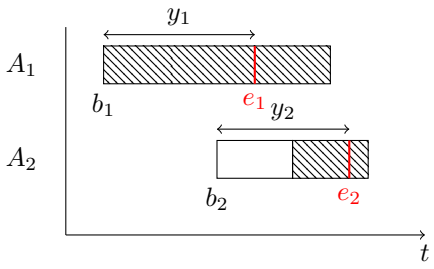


Figure 2: The running example seen from an activities point of view to explain the encoding of the problem. The striped regions are the uncontrollable space, namely where an uncontrollable time point can be scheduled given the decision on the related controllable time point. Each contingent constraint is seen as an activity (A_1 and A_2 in the picture) and the duration y_i is the actual duration of the i-th activity.

that $\bigvee_{j=1}^{E_i}(y_i \in [l_{i,j}, u_{i,j}])$. y_i represents the duration of the interval $[b_i, e_i]$ that is constrained by the i-th contingent constraint. We are thus symbolically encoding a situation $\omega \doteq (\omega_1, \ldots, \omega_{|X_u|})$ in which y_i models the value of ω_i. Figure 2 gives a pictorial representation of this encoding interpreted at the activity level.

Definition 5. *Given a TPU (X_c, X_u, C_c, C_f), let $\vec{Y}_u$ be the vector of uncontrollable duration variables $(y_1, y_2, \ldots, y_m)$, with $m = |X_u|$. We define the encoding of the problem as a tuple $(\vec{X}_c, \vec{Y}_u, \Gamma(\vec{Y}_u), \Psi(\vec{X}_c, \vec{Y}_u))$ where*

$$\Gamma(\vec{Y}_u) \doteq \bigwedge_{i=1}^{m} \bigvee_{j=1}^{E_i} (y_i \in [l_{i,j}, u_{i,j}])$$

and

$$\begin{aligned}\Psi(\vec{X}_c, \vec{Y}_u) &\doteq \bigwedge_{cf_i \in C_f} cf_i[(\vec{X}_c + \vec{Y}_u)/\vec{X}_u](\vec{X}_c, \vec{Y}_u) \\ &\doteq \bigwedge_{cf_i \in C_f} cf_i[(b_1 + y_1)/e_1][(b_2 + y_2)/e_2] \ldots [(b_m + y_m)/e_m](\vec{X}_c, \vec{Y}_u).\end{aligned}$$

Intuitively, $\Gamma(\vec{Y}_u)$ is the formula representing the conjunction of all the contingent constraints after the recoding, and $\Psi(\vec{X}_c, \vec{Y}_u)$ is the conjunction of all the free constraints rewritten in terms of $\vec{X}_c$ and $\vec{Y}_u$.

We remark that the use of this encoding yields two consequences. First, thanks to the redefinition of each e_i in terms of y_i, we managed to encode the contingent constraints in terms of $\vec{Y}_u$ only, therefore they are independent of the values of the controllable time points ($\vec{X}_c$). Intuitively, $\Gamma(\vec{Y}_u)$ encodes the set of all possible situations (Ω_P) for the given problem P: each model of $\Gamma(\vec{Y}_u)$ corresponds to a situation ω. Second, the constraints in this formulation are expressed in the $\mathcal{LRA}$ theory (the original formulation was expressed in the $\mathcal{RDL}$ fragment of $\mathcal{LRA}$). This encoding applied to the STPU problem in Figure 1 is shown in Figure 3.

From here on, we assume an encoded problem $(\vec{X}_c, \vec{Y}_u, \Gamma(\vec{Y}_u), \Psi(\vec{X}_c, \vec{Y}_u))$ is given. Intuitively, a temporal problem is weakly controllable if there exists a strategy that

$$
\begin{aligned}
X_c &= \{b_1, b_2\} \\
Y_u &= \{y_1, y_2\} \\
\Gamma(\vec{Y}_u) &= (y_1 \geq 0) \wedge (y_1 \leq 3) \wedge (y_2 \geq 1) \wedge (y_2 \leq 2) \\
\Psi(\vec{X}_c, \vec{Y}_u) &= (b_2 - b_1 \geq 0) \wedge \\
&\quad ((b_1 + y_1) - (b_2 + y_2) \leq 1) \wedge \\
&\quad ((b_2 + y_2) - b_1 \leq 2)
\end{aligned}
$$

Figure 3: The encoding of the example STPU of Figure 1.

maps every situation to a corresponding assignment to controllable time points, in such a way that all free constraints are satisfied. We can rephrase the concept of weak controllability presented in Definition 4 as a satisfiability problem modulo the $\mathcal{LRA}$ theory as follows.

Proposition 1. *Let $P \doteq (X_c, X_u, C_c, C_f)$ be a TPU and let $(\vec{X}_c, \vec{Y}_u, \Gamma(\vec{Y}_u), \Psi(\vec{X}_c, \vec{Y}_u))$ be its encoding. P is weakly controllable if and only if the following formula is valid in the $\mathcal{LRA}$ theory.*

$$\forall \vec{Y}_u . \exists \vec{X}_c . (\Gamma(\vec{Y}_u) \rightarrow \Psi(\vec{X}_c, \vec{Y}_u)) \tag{1}$$

The formula in Equation 1 is a direct formalization of the intuitive notion of weak controllability, and of the original definition in [38]. The universal quantifier captures the uncertainty in the decision of the duration variables. The implication ensures that free constraints are checked only when $\Gamma(\vec{Y}_u)$ is satisfied, that is only on assignments that encode situations of the original temporal problem. In fact, if $\Gamma(\vec{Y}_u)$ is not satisfied, the implication is automatically satisfied.

If we interpret the vector of time points $\vec{X}_c$ and the vector of durations $\vec{Y}_u$ as vectors of real variables, and the set of constraints $\Gamma(\vec{Y}_u)$ and $\Psi(\vec{X}_c, \vec{Y}_u)$ as formulae in $\mathcal{QF\_LRA}$, Equation 1 becomes a formula in $\mathcal{LRA}$ that is valid if and only if the problem is weakly controllable.

For example, the problem depicted in Figure 1 (a) is weakly controllable if and only if the following formula is valid.

$$
\begin{aligned}
\forall y_1, y_2 . \exists b_1, b_2 . (((y_1 \geq 0) \wedge (y_1 \leq 3) \wedge (y_2 \geq 1) \wedge (y_2 \leq 2)) \rightarrow \\
((b_2 - b_1 \geq 0) \wedge ((b_1 + y_1) - (b_2 + y_2) \leq 1) \wedge \\
((b_2 + y_2) - b_1 \leq 2)))
\end{aligned}
$$

Looking at the weak controllability formal characterization in Proposition 1 from an SMT perspective, it is clear that we are solving the validity problem of an $\mathcal{LRA}$ formula. Any SMT solver supporting $\mathcal{LRA}$ is able to deal with such a formula directly and it can correctly solve the problem. However, due to the high computational cost of directly handling quantifiers, an optimized encoding is required.

We first rewrite the formula encoding weak controllability in Proposition 1 by transforming the external universal quantifier into the negation of an existential one, and we

$$\neg\exists b_1, b_2.((y_1 \geq 0) \wedge (y_1 \leq 3) \wedge (y_2 \geq 1) \wedge (y_2 \leq 2)) \rightarrow ((b_2 - b_1 \geq 0) \wedge ((b_1 + y_1) - (b_2 + y_2) \leq 1) \wedge ((b_2 + y_2) - b_1 \leq 2))$$

(a)

$$(y_1 \geq 0) \wedge (y_1 \leq 3) \wedge (y_2 \geq 1) \wedge (y_2 \leq 2) \wedge \neg\exists b_1, b_2.((b_2 - b_1 \geq 0) \wedge ((b_1 + y_1) - (b_2 + y_2) \leq 1) \wedge ((b_2 + y_2) - b_1 \leq 2))$$

(b)

Figure 4: Inverted encoding (a) and assumption-extraction encoding (b) applied to the running example STPU of Figure 1.

consider the negation of the resulting formula. We call the resulting formula *inverted encoding*.

$$\neg\exists \vec{X}_c.(\Gamma(\vec{Y}_u) \rightarrow \Psi(\vec{X}_c, \vec{Y}_u)) \tag{2}$$

If this formula is unsatisfiable, then the problem is weakly controllable, while if it is satisfiable, then the problem is not weakly controllable. Note that in Equation 2 we dropped the outermost $\neg\exists \vec{Y}_u$ as any SMT problem is inherently an existential quantification and we consider the negation by reversing the interpretation of the result. Intuitively, we are searching for an assignment to the uncontrollable time points that is able to violate the free constraints under any possible strategy (it is a winning strategy for the environment). In fact, if the formula is satisfiable, each model corresponds to a situation for which no strategy of controllable time-point assignment exists. Therefore, differently from Equation 1, this encoding is also helpful for debugging a non-weakly controllable problem. This encoding still requires a solver with full support of $\mathcal{LRA}$, but is able to exploit the searching power of the SMT framework and, in case of non-weak controllability, it allows for the extraction of debug information by providing a model of the formula. An example of this encoding for the running example problem is shown in Figure 4 (a).

A further improvement can be achieved by limiting as much as possible the scope of the existential quantifier. To this extent, we push the existential quantifier over the implication, and thus the quantification is limited to the free constraints only (ref. as *assumption-extraction* encoding):

$$\Gamma(\vec{Y}_u) \wedge \neg\exists \vec{X}_c.\Psi(\vec{X}_c, \vec{Y}_u). \tag{3}$$

The assumption-extraction encoding for the running example problem is reported in Figure 4 (b).

The following proposition states that the inverted and assumption-extraction encodings are logically equivalent.

Proposition 2. *Equation 2 and Equation 3 are logically equivalent.*

Proof. We show how to convert Equation 2 into Equation 3, using logically equivalent

rewritings.

$$
\begin{aligned}
&\neg\exists\vec{X}_c.(\Gamma(\vec{Y}_u) \rightarrow \Psi(\vec{X}_c, \vec{Y}_u)) \\
\Leftrightarrow\ &\neg\exists\vec{X}_c.(\neg\Gamma(\vec{Y}_u) \vee \Psi(\vec{X}_c, \vec{Y}_u)) \\
\Leftrightarrow\ &\neg((\exists\vec{X}_c.\neg\Gamma(\vec{Y}_u)) \vee (\exists\vec{X}_c.\Psi(\vec{X}_c, \vec{Y}_u))) \\
\Leftrightarrow\ &\neg(\neg\Gamma(\vec{Y}_u) \vee (\exists\vec{X}_c.\Psi(\vec{X}_c, \vec{Y}_u))) \\
\Leftrightarrow\ &\Gamma(\vec{Y}_u) \wedge \neg(\exists\vec{X}_c.\Psi(\vec{X}_c, \vec{Y}_u))
\end{aligned}
$$

□

5. Strategies for Weak Controllability

We now consider the problem of actually executing a control strategy that is associated with a given weakly controllable TPU. A TPU is a modeling framework that represents a set of assumptions over the environment and imposes a set of requirements to be fulfilled. We consider the use-case in which a strategy for scheduling the controllable time points is repeatedly executed by reading the inputs from the environment in the form of a situation. Such a situation is generated by reading the parameters on which the uncontrollable durations depends, by means of appropriate sensors and estimators. The strategy computes an assignment to the controllable time points that fulfills the problem constraints and is then deployed to an actuator for execution.

The problem we tackle here is to automatically synthesize such a strategy: we discuss two approaches. First, we use a TP solver to do on-line reasoning, thus executing a control strategy that is *implicitly* defined in the TPU, if solvable. Then, we investigate the idea of *explicit* strategies, that can be readily executed without resorting to on-line reasoning.

5.1. Implicit strategies

A way of obtaining a strategy for a weakly controllable TPU is given by Definition 3 and depicted in Figure 5: when a situation $\bar{S}$ is read[3], we eliminate the uncertainty by substituting the uncontrollable duration variables in the TPU formulation with the values obtained from the situation (obtaining a TP that is the projection of the TPU). Then, we solve the resulting temporal problem, that is now without uncertainty, and return the assignment to the controllable time points $\bar{X}_c$ for execution. Formally, given the encoding of a TPU $(\vec{X}_c, \vec{Y}_u, \Gamma(\vec{Y}_u), \Psi(\vec{X}_c, \vec{Y}_u))$ and an assignment to all the uncontrollable durations $\bar{S}$ fulfilling $\Gamma(\bar{S})$ (a situation), we can find an assignment to the controllable variables $\vec{X}_c$ by finding a model for the formula $\Psi(\vec{X}_c, \bar{S})$. This strategy requires a solver to be executed once the situation $\bar{S}$ is known. In practice, we can implement this idea using any SMT solver by searching for a model for $\Psi(\vec{X}_c, \bar{S})$. However, this approach (called IMPLICIT-SMT) requires one to solve a separate SMT problem for each situation. A more advanced approach is to exploit the incrementality

[3] $\bar{S}$ is a vector of $|\vec{Y}_u|$ rational numbers, one for each uncontrollable duration.

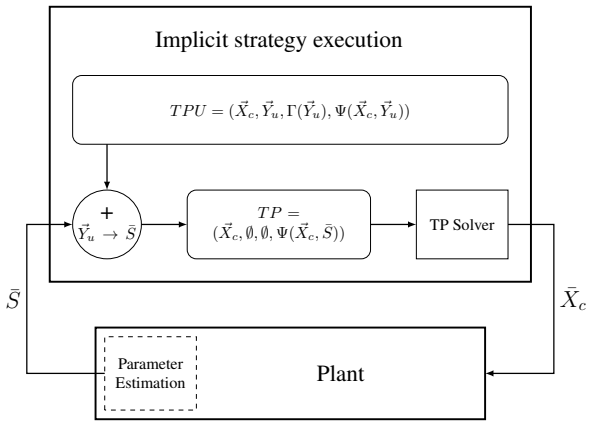


Figure 5: Schematic view of implicit strategy mechanism. The strategy is repeatedly executed once a situation is obtained by estimating the relevant parameters in the Plant. The output of the strategy is a controllable schedule (i.e. an assignment $\bar{X}_c$ to all the controllable time points). The implicit strategy works by "projecting away" the uncertainty in the TPU: the uncontrollable durations $\vec{Y}_u$ are substituted with the actual values of the situation $\bar{S}$. Then, a TP is obtained and is solved using a TP solver, yielding the assignment ($\bar{X}_c$) to the controllable time points.

feature of modern SMT solvers [4], allowing the solver to "recycle" discovered clauses and lemmas among different situations. For this purpose, we designed an incremental approach, described in Algorithm 1. IMPLICIT-SMT-INCREMENTAL takes the encoding of a TPU $(\vec{X}_c, \vec{Y}_u, \Gamma(\vec{Y}_u), \Psi(\vec{X}_c, \vec{Y}_u))$, and initializes the SMT solver by asserting the problem constraints $\Psi(\vec{X}_c, \vec{Y}_u)$. Then, it enters a (possibly infinite) loop, and processes a sequence of situations $\bar{S}^1, \bar{S}^2, \cdots$. The problem description is asserted in the solver once and for all, while the situation is first asserted, and once an assignment is found, it is retracted.

The main drawback of the implicit approach is the requirement of on-line reasoning. In fact, once the situation is known, a solver is invoked to discover the assignment for the controllable time points. Solving the TP resulting from the projection of a TPU is hard in general. If the problem belongs to the STPU problem class the resulting STP can be solved in polynomial time, but for the general case of DTPU, the projection results in a DTP that is, in general, NP-hard [37]. In addition, having a solver as part of the run-time may require much more expensive platforms.

5.2. Explicit strategies

We avoid the burden of on-line reasoning by providing techniques for the synthesis of functions that are simple and fast to execute. Consider the formalization in Proposition 1. Interestingly, we can apply skolemization [24], thus replacing the existential quantifier by means of a fresh function symbol.

Theorem 1. *A TPU (X_c, X_u, C_c, C_f) is weakly controllable if and only if the formula*

$$\forall \vec{Y}_u . \Gamma(\vec{Y}_u) \rightarrow \Psi(f(\vec{Y}_u), \vec{Y}_u). \tag{4}$$

is satisfiable.

Algorithm 1 Implicit strategy execution based on SMT with incrementality

1: **procedure** IMPLICIT-SMT-INCREMENTAL($\Gamma(\vec{Y}_u)$, $\Psi(\vec{X}_c, \vec{Y}_u)$)
2: **for all** $b_i \in \vec{X}_c$ **do**
3: SMT.DECLAREREALVAR(b_i)
4: **for all** $y_j \in \vec{Y}_u$ **do**
5: SMT.DECLAREREALVAR(y_i)
6: SMT.ASSERT($\Psi(\vec{X}_c, \vec{Y}_u)$)
7: **loop**
8: $\bar{S} \leftarrow$ WAITFORSITUATION()
9: SMT.PUSH()
10: **for all** $y_i \in \vec{Y}_u$ **do**
11: SMT.ASSERT($y_i = \bar{S}_i$)
12: **if** SMT.SOLVE $=$ SAT **then**
13: $\mu \leftarrow$ SMT.GETMODEL()
14: EXCECUTETIMEPOINTS(μ)
15: **else**
16: $\bot$ ▷ Unreachable if the problem is WC and the situation S fulfills $\Gamma(\vec{Y}_u)$
17: SMT.POP()
18: **end loop**
19: **end procedure**

Proof. Equation 4 is the result of applying the skolemization [24] procedure to Equation 1. Since skolemization produces an equi-satisfiable formula, Equation 4 is equi-satisfiable to Equation 1. Since Equation 1 has no free variables nor has uninterpreted terms, satisfiability coincides with validity. Therefore Equation 1 is valid if and only if Equation 4 is satisfiable, and Proposition 1 states that the TPU is weakly controllable if and only if Equation 1 is valid. □

We transform the inner existential quantifier into a function f that models the weak strategy for the problem. In fact, in Equation 4, the interpretation of the function f is exactly a strategy that solves the problem. Equation 4 gives a clear vision of what a strategy is: a function that gets in input the uncontrollable durations and returns an assignment to the controllable time points that fulfills all the problem constraints.

In principle, one would like to exploit this formulation to query an SMT solver, and extract, from the model, a closed form for the strategy f. However, Equation 4 is a quantified first-order formula involving uninterpreted functions[4], that is in general undecidable.

In the following, we focus on two types of strategies: *linear strategies*, where each controllable variable is computed as a linear combination of the uncontrollable durations; *piecewise-linear strategies*, where different linear strategies are executed de-

[4]Formally, Equation 4 is a quantified first-order formula expressed in the theory combination of $\mathcal{LRA}$ and the theory of uninterpreted functions ($\mathcal{EUF}$) [4].

pending on the input situation.

From here on, we assume the encoding $(\vec{X}_c, \vec{Y}_u, \Gamma(\vec{Y}_u), \Psi(\vec{X}_c, \vec{Y}_u))$ of a TPU is given. In general, a weak strategy is a function that maps each assignment to uncontrollable durations satisfying $\Gamma(\vec{Y}_u)$ (i.e. each situation) into an assignment to the controllable time points, such that all the free constraints are satisfied.

Definition 6. *A weak strategy for a TPU is a function $f : \mathbb{R}^{|Y_u|} \to \mathbb{R}^{|X_c|}$ defined for every point $\vec{Y}_u$ in $\Gamma(\vec{Y}_u)$ and such that $\Psi(f(\vec{Y}_u), \vec{Y}_u)$ holds for every $\vec{Y}_u$ in $\Gamma(\vec{Y}_u)$.*

Note that, this definition does not impose any constraint (e.g. linearity, continuity) on f other than being a function.

In Definition 6 we modeled a weak strategy as a single function $f : \mathbb{R}^{|Y_u|} \to \mathbb{R}^{|X_c|}$, but we can equivalently consider a set of functions $f_1, \ldots, f_{|X_c|}$ each computing a schedule for a single controllable time point given the situation. The two formalizations are equivalent because if there exists a unique function f, we can obtain the set of function by projection of f and vice-versa.

Let $\bar{f}(\vec{Y}_u) : \mathbb{R}^{|Y_u|} \to \mathbb{R}^{|X_c|}$ be a strategy. The strategy imposes a relation between the controllable time points and the uncontrollable durations: $\vec{X}_c = \bar{f}(\vec{Y})$. If such a relation is expressible as a formula in a theory $\mathcal{T}$ we can check whether $\bar{f}$ is a weak strategy for a given temporal problem by checking the existence of a point in $\Gamma(\vec{Y}_u)$ that violates the free constraints.

$$\Gamma(\vec{Y}_u) \wedge \neg\Psi(\vec{X}_c, \vec{Y}_u) \wedge (\vec{X}_c = \bar{f}(\vec{Y}_u)) \tag{5}$$

If Equation 5 is satisfiable modulo $\mathcal{T} \cup \mathcal{QF}\_\mathcal{LRA}$, then $\bar{f}(\vec{Y}_u)$ is not a valid weak strategy, because there exists a situation for which the strategy violates the free constraints. In this case, $\mathcal{T}$ can be any theory needed to express the relation imposed by the strategy, for example it could be $\mathcal{LRA}$ or even Nonlinear Real Arithmetic. Note that, this check is very useful in practice if $\bar{f}(\vec{Y}_u)$ can be expressed in $\mathcal{QF}\_\mathcal{LRA}$ because the entire check would fit in $\mathcal{QF}\_\mathcal{LRA}$. In the following, we describe two possible shapes of strategies, namely linear and piecewise-linear. Both the shapes can be expressed in $\mathcal{QF}\_\mathcal{LRA}$. Therefore checking if such strategies are weak strategies for a given problem is possible by performing a single call to an SMT solver in $\mathcal{QF}\_\mathcal{LRA}$.

Linear strategies. A linear strategy is such that the value of every controllable time point is obtained as a linear combination of $\vec{Y}_u$. Let $n \doteq |X_c|$ and $m \doteq |X_u|$. A linear strategy can be represented with a matrix A of real coefficients of size $n \times m$ and a vector $\vec{c}$ of size n. Every controllable variable is scheduled according to a linear function of the uncontrollable durations. The strategy $f(\vec{Y}_u)$ can be then expressed as $A \cdot \vec{Y}_u + \vec{c}$ in which each $b_i \in X_c$ can be computed as $A_{i,1}y_1 + \ldots + A_{i,m}y_m + c_i$. Therefore, the matrix A must have one column for every duration and the vector $\vec{c}$ contains the constant additive terms. The problem of synthesizing a linear strategy is then equivalent to the problem of finding a suitable matrix A and vector $\vec{c}$.

Piecewise-linear strategies. A more general form of strategy is the piecewise-linear strategy, that is the composition of a finite number of linear strategies. A piecewise-linear strategy is defined by cases over a finite partition of the situations (a partition

of the region represented by $\Gamma(\vec{Y}_u)$). For each case we have a linear strategy that is a valid weak strategy for that subset of the situations. We can compose these linear strategies by first checking in which element of the partition the observed situation belongs, and then applying the corresponding linear strategy. In this setting, a linear strategy is a particular case of a piecewise-linear strategy in which we have a partition of cardinality one.

Definition 7. *A* piecewise-linear strategy *is a function*

$$f(\vec{Y}_u) \doteq \begin{cases} f^1(\vec{Y}_u) & \textit{if } \eta^1(\vec{Y}_u) \\ f^2(\vec{Y}_u) & \textit{else if } \eta^2(\vec{Y}_u) \\ \dots & \\ f^k(\vec{Y}_u) & \textit{else if } \eta^k(\vec{Y}_u) \end{cases}$$

where each f^i is a linear strategy and $\eta^i(\vec{Y}_u)$ are sub-regions of $\Gamma(\vec{Y}_u)$ such that $\Gamma(\vec{Y}_u) \subseteq (\bigcup_{i=1}^{k} \eta^i(\vec{Y}_u))$.

Note that, even this kind of strategy can be directly encoded in $\mathcal{QF\_LRA}$. We call each pair $(f^i(\vec{Y}_u), \eta^i(\vec{Y}_u))$ a "piece" of the strategy. In order to compactly represent a piecewise-linear strategy in the algorithms we abstract a piecewise-linear strategy $f(\vec{Y}_u)$ as the ordered list of its pieces. For example, the strategy $f(\vec{Y}_u)$ in Definition 7 can be represented as the following list of pieces:

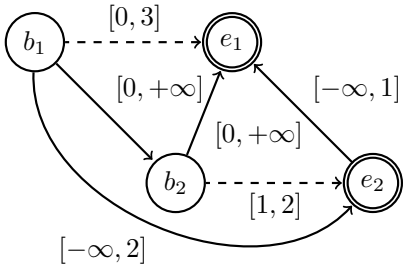
$$[(f^1(\vec{Y}_u), \eta^1(\vec{Y}_u)), (f^2(\vec{Y}_u), \eta^2(\vec{Y}_u)), \dots, (f^k(\vec{Y}_u), \eta^k(\vec{Y}_u))].$$

Following Definition 7, no continuity requirement is imposed on a piecewise-linear strategy. Continuity is not required by the weak controllability definition and is not a useful requirement for our setting, as we assume that the parameters yielding the situation are fully specified before scheduling the problem. Continuity is instead an important issue in dynamic strategies as small variations in the situation should not yield substantial variations in the strategy outcome. However this discussion is out of the scope of this paper.

Linearity is not enough. A linear strategy is very useful in practice: it is compact to represent and easy to evaluate. In fact, it can be represented using just a matrix and a vector; moreover, given an assignment to the uncontrollable duration, we can compute the resulting assignment to the controllable variables by means of a single matrix multiplication. In general, unfortunately, a weakly controllable TPU is not guaranteed to have such a strategy. In fact, this holds even for the STPU class of problems.

Theorem 2. *There exists an STPU that is weakly controllable and does not have any linear strategy.*

Proof. Let us consider the STPU depicted in Figure 6 obtained by adding the constraint $e_1 - b_2 \in [0, +\infty)$ to the running example in Figure 1. In the following we show that this STPU is weakly controllable, but there exists no linear strategy.

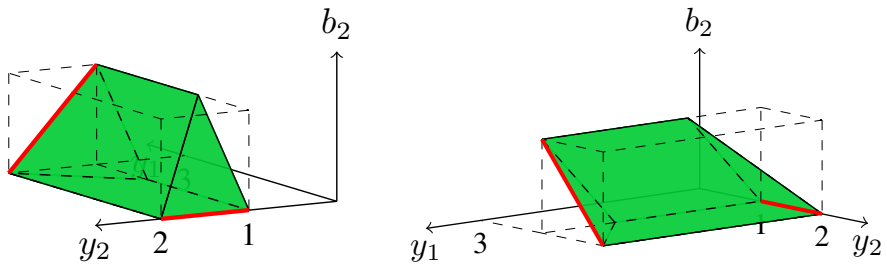


(a)

$$
\begin{aligned}
X_c &= \{b_1, b_2\} \\
X_u &= \{e_1, e_2\} \\
C_c &= \{e_1 - b_1 \in [0, 3], \\
&\qquad e_2 - b_2 \in [1, 2]\} \\
C_f &= \{b_2 - b_1 \in [0, +\infty), \\
&\quad (*)\ e_1 - b_2 \in [0, +\infty), \\
&\qquad e_1 - e_2 \in (-\infty, 1], \\
&\qquad e_2 - b_1 \in (-\infty, 2]\}
\end{aligned}
$$

(b)

Figure 6: (a) The modified running example STPU: the problem is weakly controllable, but does not have any linear strategy. (b) The constraint definitions for the modified running example.



(a) (b)

Figure 7: (a) The region of feasibility of the STPU in Figure 3 with $b_1 = 0$ in the space of b_2, y_1 and y_2, depicted from two different angles.

The problem is weakly controllable, because we can apply the following piecewise-linear weak strategy.

$$
\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = f(y_1, y_2) \doteq \begin{cases} \begin{pmatrix} 0 \\ y_1 - y_2 - 1 \end{pmatrix} & \text{if } (y_2 \leq y_1 - 1) \\ \begin{pmatrix} 0 \\ 0 \end{pmatrix} & \text{otherwise} \end{cases}
$$

This strategy corresponds to the following assignments.

$$
b_1 = 0
$$

$$
b_2 = \begin{cases} y_1 - y_2 - 1 & \text{if } (y_2 \leq y_1 - 1) \\ 0 & \text{otherwise} \end{cases}
$$

This strategy clearly covers the entire uncontrollable space because it is a total function. Given this strategy, the free constraints are always satisfied: assuming $\Gamma(y_1, y_2)$ and $y_2 > y_1 - 1$, the formula $\Psi(\vec{X}_c, \vec{Y}_u)$ reduces to

$$\begin{aligned}\Psi(f(y_1, y_2), y_1, y_2) \doteq\ & (0 - 0 \geq 0) \wedge ((0 + y_1) - 0 \geq 0) \wedge \\ & ((0 + y_1) - (0 + y_2) \leq 1) \wedge \\ & ((0 + y_2) - 0 \leq 2) \\ \Leftrightarrow\ & (y_1 \geq 0) \wedge (y_1 - y_2 \leq 1) \wedge (y_2 \leq 2).\end{aligned} \tag{6}$$

The atoms $(y_1 \geq 0)$ and $(y_2 \leq 2)$ follow from the assumptions of $\Gamma(y_1, y_2)$ while the atom $(y_1 - y_2 \leq 1)$ is entailed by the condition of the piece: $y_2 > y_1 - 1$.

Considering the other piece, namely the case $y_2 \leq y_1 - 1$, we obtain the following.

$$\begin{aligned}\Psi(f(y_1, y_2), y_1, y_2) \doteq\ & (y_1 - y_2 - 1 - 0 \geq 0) \wedge ((0 + y_1) - (y_1 - y_2 - 1) \geq 0) \wedge \\ & ((0 + y_1) - (y_1 - y_2 - 1 - y_2) \leq 1) \wedge \\ & ((y_1 - y_2 - 1 + y_2) - 0 \leq 2) \\ \Leftrightarrow\ & (y_2 \leq y_1 - 1) \wedge (y_2 \geq 1) \wedge (1 \leq 1) \wedge (y_1 \leq 3)\end{aligned} \tag{7}$$

The atoms $(y_2 \geq 1)$ and $(y_1 \leq 3)$ follow from the assumptions of $\Gamma(y_1, y_2)$ while the atom $(y_2 \leq y_1 - 1)$ is exactly the condition of the piece we are considering.

We now show that no linear strategy exists for the given problem. For the sake of contradiction, let us suppose that a linear strategy exists for the problem. Let $\bar{f}(\vec{Y}_u) \doteq A \cdot \vec{Y}_u + \vec{c}$ be such a linear strategy. Then, $b_1 \doteq A_{1,1}y_1 + A_{1,2}y_2 + c_1$ and $b_2 \doteq A_{2,1}y_1 + A_{2,2}y_2 + c_2$. If $\bar{f}(\vec{Y}_u)$ is a valid weak linear strategy, it must fulfill the problem constraints in all the situations. Let us consider four particular situations, namely $\omega_1 = (0, 1)$ (that is, $y_i \doteq 0$ and $y_2 \doteq 1$), $\omega_2 = (0, 2)$, $\omega_3 = (3, 1)$ and $\omega_4 = (3, 2)$.

We can now build the following system obtained by instantiating each constraint of $\Psi(b_1, b_2, y_1, y_2)$ in each of the four picked situations, and by substituting each b_i with

its strategy definition.

$$\begin{cases}
(A_{2,1}\cdot 0 + A_{2,2}\cdot 1 + c_2) - (A_{1,1}\cdot 0 + A_{1,2}\cdot 1 + c_1) \geq 0 \\
(A_{1,1}\cdot 0 + A_{1,2}\cdot 1 + c_1 + 0) - (A_{2,1}\cdot 0 + A_{2,2}\cdot 1 + c_2) \geq 0 \\
(A_{1,1}\cdot 0 + A_{1,2}\cdot 1 + c_1 + 0) - (A_{2,1}\cdot 0 + A_{2,2}\cdot 1 + c_2 + 1) \leq 1 \\
(A_{2,1}\cdot 0 + A_{2,2}\cdot 1 + c_2 + 1) - (A_{1,1}\cdot 0 + A_{1,2}\cdot 1 + c_1) \leq 2 \\
(A_{2,1}\cdot 0 + A_{2,2}\cdot 2 + c_2) - (A_{1,1}\cdot 0 + A_{1,2}\cdot 2 + c_1) \geq 0 \\
(A_{1,1}\cdot 0 + A_{1,2}\cdot 2 + c_1 + 0) - (A_{2,1}\cdot 0 + A_{2,2}\cdot 2 + c_2) \geq 0 \\
(A_{1,1}\cdot 0 + A_{1,2}\cdot 2 + c_1 + 0) - (A_{2,1}\cdot 0 + A_{2,2}\cdot 2 + c_2 + 2) \leq 1 \\
(A_{2,1}\cdot 0 + A_{2,2}\cdot 2 + c_2 + 2) - (A_{1,1}\cdot 0 + A_{1,2}\cdot 2 + c_1) \leq 2 \\
(A_{2,1}\cdot 3 + A_{2,2}\cdot 1 + c_2) - (A_{1,1}\cdot 3 + A_{1,2}\cdot 1 + c_1) \geq 0 \\
(A_{1,1}\cdot 3 + A_{1,2}\cdot 1 + c_1 + 3) - (A_{2,1}\cdot 3 + A_{2,2}\cdot 1 + c_2) \geq 0 \\
(A_{1,1}\cdot 3 + A_{1,2}\cdot 1 + c_1 + 3) - (A_{2,1}\cdot 3 + A_{2,2}\cdot 1 + c_2 + 1) \leq 1 \\
(A_{2,1}\cdot 3 + A_{2,2}\cdot 1 + c_2 + 1) - (A_{1,1}\cdot 3 + A_{1,2}\cdot 1 + c_1) \leq 2 \\
(A_{2,1}\cdot 3 + A_{2,2}\cdot 2 + c_2) - (A_{1,1}\cdot 3 + A_{1,2}\cdot 2 + c_1) \geq 0 \\
(A_{1,1}\cdot 3 + A_{1,2}\cdot 2 + c_1 + 3) - (A_{2,1}\cdot 3 + A_{2,2}\cdot 2 + c_2) \geq 0 \\
(A_{1,1}\cdot 3 + A_{1,2}\cdot 2 + c_1 + 3) - (A_{2,1}\cdot 3 + A_{2,2}\cdot 2 + c_2 + 2) \leq 1 \\
(A_{2,1}\cdot 3 + A_{2,2}\cdot 2 + c_2 + 2) - (A_{1,1}\cdot 3 + A_{1,2}\cdot 2 + c_1) \leq 2
\end{cases}$$

This system can be rewritten as follows.

$$\begin{cases}
-A_{1,2} + A_{2,2} - c_1 + c_2 \geq 0 \\
A_{1,2} - A_{2,2} + c_1 - c_2 \geq 0 \\
-A_{1,2} + A_{2,2} - c_1 + c_2 \geq 0 \\
A_{1,2} - A_{2,2} + c_1 - c_2 \geq -1 \\
-2A_{1,2} + 2A_{2,2} - c_1 + c_2 \geq 0 \\
2A_{1,2} - 2A_{2,2} + c_1 - c_2 \geq 0 \\
-2A_{1,2} + 2A_{2,2} - c_1 + c_2 \geq -3 \\
+2A_{1,2} - 2A_{2,2} + c_1 - c_2 \geq 0 \\
-3A_{1,1} - A_{1,2} + 3A_{2,1} + A_{2,2} - c_1 + c_2 \geq 0 \\
3A_{1,1} + A_{1,2} - 3A_{2,1} - A_{2,2} + c_1 - c_2 \geq -3 \\
-3A_{1,1} - A_{1,2} + 3A_{2,1} + A_{2,2} - c_1 + c_2 \geq 1 \\
3A_{1,1} + A_{1,2} - 3A_{2,1} - A_{2,2} + c_1 - c_2 \geq -1 \\
-3A_{1,1} - 2A_{1,2} + 3A_{2,1} + 2A_{2,2} - c_1 + c_2 \geq 0 \\
3A_{1,1} + 2A_{1,2} - 3A_{2,1} - 2A_{2,2} + c_1 - c_2 \geq -3 \\
-3A_{1,1} - 2A_{1,2} + 3A_{2,1} + 2A_{2,2} - c_1 + c_2 \geq 0 \\
-3A_{2,1} - 2A_{2,2} + 3A_{1,1} + 2A_{1,2} + c_1 - c_2 \geq 0
\end{cases}$$

The system admits no solution in the real numbers. Therefore there exists no linear strategy for the given problem as there exists no assignment to the coefficients that are able to fulfill the four situations at the same time. □

In order to graphically explain the reason why no linear strategy exists for the given problem, we plotted the space of free constraints of the STPU problem in the space (y_1, y_2, b_2) regions in Figures 7 (a) and 7 (b) (without loss of generality, we assigned $b_1 = 0$ as we can always freely assign a reference controllable time point thanks to the $\mathcal{RDL}$ property of shifting solutions). The plot clearly shows that there exists no linear strategy for b_2. Considering the vertex $(0, 1)$ in the space (y_1, y_2), a linear solution must contain point $(0, 1, 0)$ as it is the only feasible point for the vertex $(0, 1)$. Similarly, considering $(0, 2)$ we must include $(0, 2, 0)$; considering $(3, 1)$ we must include $(3, 1, 1)$ and for $(3, 2)$ the linear solution must include the point $(3, 2, 0)$. However, no linear solution can exist because no plane contains all the four points. In fact, the only plane containing $(0, 1, 0)$, $(0, 2, 0)$ and $(3, 2, 0)$ is $b_2 = 0$, but this plane does not contain the point $(3, 1, 1)$.

We can also exploit the encodings for the decision problem to show that the STPU in Figure 6 is weakly controllable. The inverted SMT encoding of Equation 2 for the example problem is as follows.

$$\begin{aligned}\neg\exists b_1, b_2.((y_1 \geq 0) \wedge (y_1 \leq 3) \wedge (y_2 \geq 1) \wedge (y_2 \leq 2)) \rightarrow \\ ((b_2 - b_1 \geq 0) \wedge ((b_1 + y_1) - b_2 \geq 0) \wedge \\ ((b_1 + y_1) - (b_2 + y_2) \leq 1)) \wedge \\ ((b_2 + y_2) - b_1 \leq 2)\end{aligned} \tag{8}$$

This formula can be shown to be unsatisfiable by any $\mathcal{LRA}$ SMT solver. Therefore the problem is indeed weakly controllable. The unsatisfiability of the formula is also shown by Equations 6 and 7 that provide a witness strategy for the existential quantifier, making the formula false. In fact, if $(y_2 > y_1 - 1)$ holds, Equation 8 is unsatisfiable because $b_1 = 0$ and $b_2 = 0$ is a model of Equation 6. Similarly, if $(y_2 \leq y_1 - 1)$ holds, Equation 8 is unsatisfiable because $b_1 = 0$ and $b_2 = y_1 - y_2 - 1$ is a model of Equation 7.

Piecewise-linear strategy is enough. We now prove that a piecewise-linear strategy always exists for any weakly controllable TPU.

Theorem 3. *For any given TPU P, if P is weakly controllable, then P admits a piecewise-linear strategy.*

Proof. Let $(\vec{X}_c, \vec{Y}_u, \Gamma(\vec{Y}_u), \Psi(\vec{X}_c, \vec{Y}_u))$ be the encoding of P. Since P is weakly controllable, we know that

$$\forall \vec{Y}_u.\exists \vec{X}_c.\Gamma(\vec{Y}_u) \rightarrow \Psi(\vec{X}_c, \vec{Y}_u)$$

is valid.

We want to prove that there exists a piecewise-linear strategy f such that

$$\forall \vec{Y}_u.(\Gamma(\vec{Y}_u) \wedge \vec{X}_c = f(\vec{Y}_u)) \rightarrow \Psi(\vec{X}_c, \vec{Y}_u)$$

is valid.

By construction, both $\Gamma(\vec{Y}_u)$ and $\Psi(\vec{X}_c, \vec{Y}_u)$ are formulae in $\mathcal{QF\_LRA}$ and hence they geometrically correspond to the union of finitely many closed convex polyhedra

(the polyhedra are closed because all the inequalities are non-strict by problem definition).

We now show that from each face we can extract a linear strategy that correctly work for a sub-region of $\Gamma(\vec{Y}_u)$. By combining these linear strategies for all the faces of the polyhedron we obtain a weak strategy for P.

Without loss of generality we can assume $\Psi(\vec{X}_c, \vec{Y}_u)$ being a bounded set (meaning that it can be completely contained in a ball of finite radius). This is because we already have bounds for all the uncontrollable variables in $\Gamma(\vec{Y}_u)$ (because of the assumptions in Definition 1) and we can always add upper and lower bounds on controllable variables as follows. Since the problem is weakly controllable, let $g(\vec{Y})$ be any weak strategy. For each variable $x \in \vec{X}_c$, let $u_x \doteq max(\{g(\vec{Y}_u) \mid \vec{Y}_u \models \Gamma(\vec{Y}_u)\})$ and $l_x \doteq min(\{g(\vec{Y}_u) \mid \vec{Y}_u \models \Gamma(\vec{Y}_u)\})$. We can then add the following constraint to the problem without altering its weak controllability: $x \in [l_x, u_x]$.

Let $\phi^1(\vec{X}_c, \vec{Y}_u), \cdots, \phi^w(\vec{X}_c, \vec{Y}_u)$ be the formulae corresponding to the faces of $\Psi(\vec{X}_c, \vec{Y}_u)$. Each face $\phi^z(\vec{X}_c, \vec{Y}_u)$ is a convex polyhedron and can be expressed as a system of inequalities $A(\vec{X}_c|\vec{Y}_u) \leq b$ with at least one inequality satisfied as an equality. From this system is easy to extract a linear strategy $f^z(\vec{Y}_u)$ by reducing the augmented matrix $(A|b)$ into reduced row echelon form and applying substitution to extract the relation between $\vec{X}_c$ and $\vec{Y}_u$ in closed form.

For each face $\phi^z(\vec{X}_c, \vec{Y}_u)$ we define its projection $\chi^z(\vec{Y}_u) \doteq \exists \vec{X}_c.\phi^z(\vec{X}_c, \vec{Y}_u)$. Since $\mathcal{QF\_LRA}$ admits quantifier elimination, also $\chi^z(\vec{Y}_u)$ can be expressed as a $\mathcal{QF\_LRA}$ formula, and geometrically corresponds to a finite union of convex polyhedra.

Therefore, we can build the piecewise-linear weak strategy f defined as follows.

$$f(\vec{Y}_u) \doteq \begin{cases} f^1(\vec{Y}_u) & \text{if } \chi^1(\vec{Y}_u) \\ f^2(\vec{Y}_u) & \text{else if } \chi^2(\vec{Y}_u) \\ \ldots & \\ f^k(\vec{Y}_u) & \text{else if } \chi^k(\vec{Y}_u) \end{cases}$$

Clearly, $\forall \vec{Y}_u.(\bigvee_{i=1}^{w} \chi^i(\vec{Y}_u)) \rightarrow \Gamma(\vec{Y}_u)$ because we know that P is weakly controllable and we assumed $\Psi(\vec{X}_c, \vec{Y}_u)$ to be bounded (being bounded, the projection of all the faces corresponds to the projection of the polyhedral union itself). In addition, each $f^z(\vec{Y}_u)$ applied to a point in $\chi^z(\vec{Y}_u)$ yields a point belonging to a face of the polyhedron, hence belonging to $\Psi(\vec{X}_c, \vec{Y}_u)$. Thus, the strategy is a valid weak strategy for P. □

6. Synthesis of strategies for Weak Controllability

We are interested in generating strategies that can be efficiently executed once the situation is known. Given this requirement, linear strategies are very helpful, because they are compact (the size is quadratic in the number of time points) and can be executed by performing a linear computation in the size of the strategy. Piecewise-linear strategies are also helpful because they can be executed in linear time in the size of the strategy as they require only a case switch before applying the linear executor.

The problem of synthesizing weak strategies can be classified along two dimensions; we distinguish between (i) convex (STPU) vs. disjunctive (DTPU) temporal problems and (ii) linear vs. piecewise-linear strategies. Table 1 summarizes this classification and indicates the algorithms we developed for each problem class.

	Strategy Type	
	Linear	Piecewise-Linear
Convex (STPU)	VERTEXENCODING (Section 6.1.1) INCREMENTALWEAKENING (Section 6.1.2)	SIMPLEXESDECOMPOSITION (Section 6.2.1) LAZYEXPANSION (Section 6.2.2)
Disjunctive (DTPU)	NRA ENCODING (Section 6.3)	SKINCRAWLER (Section 6.4.1) CONVEXREGIONENUMERATOR (Section 6.4.2)

Table 1: Overview of the developed algorithms with references to the section that describes each of them.

All the algorithms assume that the given problem is weakly controllable, but it is not known in advance whether the problem admits a linear strategy. Thus, the algorithms listed in the "Linear" column of Table 1 return $\perp$ in case no linear strategy exists. The others are guaranteed to find a piecewise-linear strategy. In the rest of this section we analyze each combination of temporal problem class and strategy type separately.

SMT notation. In the following, we present algorithms that use different features provided by modern SMT solvers, such as optimization [35]. We indicate with the prefix "SMT." the functions that are related with SMT solving. In particular, the function SMT.DECLARETYPEVAR(v) declares an SMT variable named v with type TYPE (e.g. SMT.DECLAREREALVAR(v) is the function that declares a real-valued SMT variable). SMT.SOLVE($\phi(\vec{x})$) is a function that checks the satisfiability of the formula $\phi(\vec{x})$ and returns "SAT" if and only if the formula is satisfiable, otherwise the function returns "UNSAT". SMT.GETMODEL() returns a satisfying assignment to the formula that was checked using SMT.SOLVE if the answer was "SAT". Finally, the function SMT.SOLVEMAXIMIZING($\phi(\vec{x})$, $h(\vec{x})$) behaves like SMT.SOLVE($\phi(\vec{x})$) but generates the model that maximizes the function $h(\vec{x})$.

In an incremental setting [7, 17], we assume a stateful SMT solver that has the following capabilities. SMT.ASSERT($\phi(\vec{x})$) conjoins the formula $\phi(\vec{x})$ to the state of the SMT solver without performing any solving operation. SMT.PUSH() records a backtrack point in the sate of the SMT solver in a stack. The last recorded state can be restored by calling the SMT.POP() function. Finally, when using incrementality we assume that the SMT.SOLVE function can be called without arguments to check the satisfiability of the conjunction of the currently asserted set of formulae.

6.1. Linear Strategies for STPU

In the following, we discuss two algorithms that are able to synthesize linear strategies for a given STPU problem. They both leverage the convexity in the constraints of the STPU problem class.

6.1.1. Vertex Encoding

If the problem is an STPU, then the free constraints represent a convex space: given any two points in the space of free constraints, any point in the line connecting these two points is also a solution. Following this idea we can generalize the result of weak controllability on bounds in [38] to the search of linear strategies. We consider all the vertexes of the uncontrollable space $\Gamma(\vec{Y}_u)$ that, by definition of $\Gamma(\vec{Y}_u)$, are the elements of the set $V_\Gamma \doteq \{l^c_{1,1}, u^c_{1,1}\} \times \cdots \times \{l^c_{m,1}, u^c_{m,1}\}$.

Theorem 4. *Let $P \doteq (X_c, X_u, C_c, C_f)$ be an STPU, $(\vec{X}_c, \vec{Y}_u, \Gamma(\vec{Y}_u), \Psi(\vec{X}_c, \vec{Y}_u))$ be its encoding and let $\bar{f} : \mathbb{R}^{|Y_u|} \to \mathbb{R}^{|X_c|}$ be a linear strategy. If $\bar{f}$ fulfills $\Psi(\vec{X}_c, \vec{Y}_u)$ in all the vertexes $v_i \in V_\Gamma$, then $\bar{f}$ is a weak linear strategy for P.*

Proof. For the sake of contradiction, let us suppose that there exists a point $\bar{p}$ in the space of $\vec{Y}_u$ such that $\Gamma(\bar{p})$ holds and $\Psi(\bar{f}(\bar{p}), \bar{p})$ does not hold.

Then, there must exist a free constraint c_k that is violated by $\bar{p}$. Since the problem is an STPU, each free constraint is geometrically either a half-space (for example $a - b \in [1, \infty)$) or the intersection of two half-spaces (for example, $a - b \in [10, 15]$ is the intersection of the half-space $a - b \leq 15$ with $a - b \geq 10$). Therefore, $\bar{p}$ does not belong to one of the half-spaces encoded by c_k. Let H be the violating half-space.

However, for each vertex $v_i \in V_\Gamma$, $\bar{f}(v_i)$ must belong to H because the free constraints are fulfilled in all the vertexes.

The point $\bar{f}(\bar{p})$ belongs to the convex hull of the points $\bar{f}(v_i)$, but then it must belong to the half-space H. Hence, we have a contradiction. □

Based on this insight, the idea is to create a single formula that encodes the problem with a symbolic strategy in all the vertexes of the uncontrollable region. The encoding is obtained instantiating the problem constraint in all the vertexes $v_i \in V_\Gamma$ and by enforcing a single hyperplane to contain all of them. If such a hyperplane exists, then it is a valid linear strategy for the entire problem.

Algorithm 2 shows the pseudo-code for extracting a linear strategy with such encoding. We create a matrix A and a vector $\vec{c}$ of real SMT variables representing the coefficients of the linear strategy. The function VERTEXASSIGNMENTS generates all the vertexes of the convex polyhedron corresponding to $\Gamma(\vec{Y}_u)$. In order to achieve this result if $\Gamma(\vec{Y}_u)$ is generated as in Definition 5, it suffices to generate all the possible combinations of assignments of contingent constraints bounds. We remark that each $\bar{p}$ is a vector of constants, and therefore the only variables occurring in the formula $\phi(A, \vec{c})$ are the coefficients of the linear strategy $f(\vec{Y}_u) \doteq A \cdot \vec{Y}_u + \vec{c}$. The function SMT.SOLVE checks the satisfiability of the given formula using an SMT solver, while SMT.GETMODEL returns the produced model in case of SAT answer.

We presented this algorithm for an encoded problem as formalized in Definition 5, but the same idea can be applied when $\Gamma(\vec{Y}_u)$ and $\Psi(\vec{X}_c, \vec{Y}_u)$ are simply conjunctive $\mathcal{QF\_LRA}$ formulae (so that they represent convex polyhedra). The only modification needed for the algorithm is to change the VERTEXASSIGNMENTS so that it is able to produce the vertexes of general formulae. For doing this we can employ well known techniques for enumerating the vertexes of a convex polyhedron [3].

Consider for example the STPU problem in Figure 1. The resulting problem admits a linear strategy. The encoding obtained by the application of Algorithm 2 is as follows.

Algorithm 2 Vertex Encoding

procedure VERTEXENCODING($\Gamma(\vec{Y}_u)$, $\Psi(\vec{X}_c, \vec{Y}_u)$)
  for all $b_i \in \vec{X}_c$ **do**
    SMT.DECLAREREALVAR(c_i)
    for all $y_j \in \vec{Y}_u$ **do**
      SMT.DECLAREREALVAR($A_{i,j}$)
  $\phi(A, \vec{c}) \leftarrow \top$
  for all $\bar{p} \in$ VERTEXASSIGNMENTS($\Gamma(\vec{Y}_u)$) **do**
    $\phi(A, \vec{c}) \leftarrow \phi(A, \vec{c}) \wedge \Psi(A \cdot \bar{p} + \vec{c}, \bar{p})$
  if SMT.SOLVE($\phi(A, \vec{c})$) = SAT **then**
    $(A, \vec{c}) \leftarrow$ SMT.GETMODEL()
    return $f(\vec{Y}_u) \doteq A \cdot \vec{Y}_u + \vec{c}$
  else
    return $\perp$
end procedure

$$
\begin{aligned}
&((A_{1,2}\cdot 0 + A_{2,2}\cdot 1 + c_2) - (A_{1,1}\cdot 0 + A_{2,1}\cdot 1 + c_1) \geq 0) \wedge \\
&(((A_{1,1}\cdot 0 + A_{2,1}\cdot 1 + c_1) + 0) - ((A_{1,2}\cdot 0 + A_{2,2}\cdot 1 + c_2) + 1) \leq 1) \wedge \\
&(((A_{1,2}\cdot 0 + A_{2,2}\cdot 1 + c_2) + 1) - (A_{1,1}\cdot 0 + A_{2,1}\cdot 1 + c_1) \leq 2) \\
\wedge \\
&((A_{1,2}\cdot 0 + A_{2,2}\cdot 2 + c_2) - (A_{1,1}\cdot 0 + A_{2,1}\cdot 2 + c_1) \geq 0) \wedge \\
&(((A_{1,1}\cdot 0 + A_{2,1}\cdot 2 + c_1) + 0) - ((A_{1,2}\cdot 0 + A_{2,2}\cdot 2 + c_2) + 2) \leq 1) \wedge \\
&(((A_{1,2}\cdot 0 + A_{2,2}\cdot 2 + c_2) + 2) - (A_{1,1}\cdot 0 + A_{2,1}\cdot 2 + c_1) \leq 2) \\
\wedge \\
&((A_{1,2}\cdot 3 + A_{2,2}\cdot 1 + c_2) - (A_{1,1}\cdot 3 + A_{2,1}\cdot 1 + c_1) \geq 0) \wedge \\
&(((A_{1,1}\cdot 3 + A_{2,1}\cdot 1 + c_1) + 3) - ((A_{1,2}\cdot 3 + A_{2,2}\cdot 1 + c_2) + 1) \leq 1) \wedge \\
&(((A_{1,2}\cdot 3 + A_{2,2}\cdot 1 + c_2) + 1) - (A_{1,1}\cdot 3 + A_{2,1}\cdot 1 + c_1) \leq 2) \\
\wedge \\
&((A_{1,2}\cdot 3 + A_{2,2}\cdot 2 + c_2) - (A_{1,1}\cdot 3 + A_{2,1}\cdot 2 + c_1) \geq 0) \wedge \\
&(((A_{1,1}\cdot 3 + A_{2,1}\cdot 2 + c_1) + 3) - ((A_{1,2}\cdot 3 + A_{2,2}\cdot 2 + c_2) + 2) \leq 1) \wedge \\
&(((A_{1,2}\cdot 3 + A_{2,2}\cdot 2 + c_2) + 2) - (A_{1,1}\cdot 3 + A_{2,1}\cdot 2 + c_1) \leq 2)
\end{aligned}
$$

The encoding is satisfiable and a possible model (encoding a linear strategy) is reported in Equation 9.

$$A = \begin{pmatrix} 0 & 0 \\ 0 & -1 \end{pmatrix} \quad \vec{c} = \begin{pmatrix} 0 \\ 2 \end{pmatrix} \tag{9}$$

Therefore, the assignments for the controllable time points b_1 and b_2 are $b_1 \doteq 0$ and $b_2 \doteq -y_2 + 2$.

Algorithm 3 Incremental weakening

1: **procedure** INCREMENTALWEAKENING($\Gamma(\vec{Y}_u)$, $\Psi(\vec{X}_c, \vec{Y}_u)$)
2: **repeat**
3: $O \leftarrow$ GETUSABLEDURATIONS($\vec{Y}_u$)
4: $N \leftarrow \{y \in \vec{Y}_u \mid y \notin O\}$
5: $\eta(\vec{X}_c, \vec{O}) \leftarrow$ SC_ENCODE($\Gamma_{|N}(\vec{N})$, $\Psi_{|X_c \cup N}(\vec{X}_c, \vec{N})$)
6: $f(\vec{O}) \leftarrow$ VERTEXENCODING($\Gamma_{|\vec{O}}(\vec{O})$, $\eta(\vec{X}_c, \vec{O})$)
7: **if** $f(\vec{O}) \neq \bot$ **then**
8: **return** ADDNULLCOLUMNS($f(\vec{O})$)
9: **until** $O = Y_u$
10: **return** $\bot$
11: **end procedure**

Note that, this approach leads to an exponential blowup in the size of the SMT problem, caused by the fact that the number of vertexes is $2^{|Y_u|}$.

6.1.2. Incremental weakening

In order to limit the exponential blowup of the previous encoding to the worst case only, we developed another approach called "*incremental weakening*", that tries to limit the number of coefficients to search for and to reduce the amount of variables that are used in the linear strategy. This idea amounts to finding a matrix A in which some (possibly many) columns are null vectors; in fact, if the i-th column is null in A, the strategy does not depend on the actual values of y_i. In the limit case in which A is the null matrix, the strategy degenerates to an assignment of constant values to each controllable time points, and thus to a strong controllability solution.

We start by solving a relaxed problem, in which no uncontrollable duration is observed. This coincides with the definition of a strong controllability problem. If a solution is found, the strong assignment is a valid weak linear strategy for the problem, because strong controllability implies weak controllability. Otherwise, a subset of the uncontrollable durations $\vec{p} \subseteq \vec{Y}_u$ is picked and marked as "usable" by the strategy. The algorithm then tries to build a linear strategy that uses uncontrollable durations in $\vec{p}$ only. In this way, we are limiting the observations available to our strategy. Using the previous algorithm, we build the coefficients for the p-th column of the matrix A and we encode the problem as in the previous algorithm, limiting the exponential explosion only to the durations marked as "usable". If the algorithm fails to find a linear strategy for a particular set of "usable" durations, a different subset of the durations is picked and the approach is iterated, until all the uncontrollable durations are marked as "usable" and the encoding coincides with the previous approach.

The pseudo-code of this method is reported in Algorithm 3. The function GETUSABLEDURATIONS returns an heuristically computed subset of $\vec{Y}_u$ that constitutes the set of "observed" durations. The function is stateful as it is assumed to return a different subset at each call. The termination of the algorithm requires that this function eventually returns the entire $\vec{Y}_u$ that exits the repeat loop fulfilling the condition at line 9. In

this termination condition, the algorithm behaves like the VERTEXENCODING procedure executed on the entire problem. The function SC_ENCODE produces the encoding of a strong controllability problem in SMT (as in [11]). This encoding is used to prevent the observation of non-used durations, leaving the others untouched. The function VERTEXENCODING is the function described in Algorithm 2. If the VERTEXENCODING function returns a strategy that works for a subset of the uncontrollable duration variables, we return the same linear strategy completed by the function ADDNULLCOLUMNS. The function adds columns of 0s in the positions of the durations that were not used. This guarantees that the strategy is independent of the actual values of those durations.

This algorithm tries to abstract the problem by limiting the set of "usable" durations in a strategy, and refines the abstraction if no linear strategy is found. The process is iterated until a strategy is found or the entire set of durations is marked as "usable".

As shown in the previous section, if we consider the running example in Figure 1, we can derive a strategy in which y_1 is never observed. The advantage of INCREMENTALWEAKENING over the previous algorithm is that if we choose to use y_2 but not y_1 in our strategy we can get to the same result reported in Equation 9 with a smaller and simpler encoding.

This algorithm depends on the heuristic used for selecting the "usable" durations. In fact, the number of cycles of the algorithm directly depends on the heuristic.

In our experiments we implemented an heuristic based on a topological sorting of the uncontrollable time points. The heuristic first generates all the singleton subsets and, if the algorithm is not terminated, considers prefixes of the topological order of increasing size until all the durations are marked as "usable" and the algorithm terminates.

6.2. Piecewise-linear Strategies for STPU

In the following, we present two algorithms for extracting a piecewise-linear strategy for a given weakly controllable STPU.

6.2.1. Simplexes decomposition

A direct approach to extract a piecewise-linear strategy consists in partitioning the region of the uncontrollable durations in a set of m-simplexes (hyper-tetrahedra in m dimensions) with $m = |Y_u|$. In geometry, a k-simplex is the generalization of a triangle to k-dimensions. A k-simplex is a k-dimensional polytope which is the convex hull of $k + 1$ linearly independent (i.e. not aligned) vertexes. For example, a 2-simplex is a triangle and a 3-simplex is a tetrahedron. We consider these polyhedra because they can be used to triangulate more complex regions [20]. The following theorem states the existence of a linear strategy in any simplex contained in the uncontrollable space.

Theorem 5. *Let $P \doteq (\vec{X}_c, \vec{Y}_u, \Gamma(\vec{Y}_u), \Psi(\vec{X}_c, \vec{Y}_u))$ be an encoded weakly controllable STPU. For each $|Y_u|$-simplex $\sigma(\vec{Y_u})$ such that $\sigma(\vec{Y_u}) \subseteq \Gamma(\vec{Y})$ there exists a valid weak linear strategy f such that $\forall \vec{Y}_u.((\sigma(\vec{Y}_u) \wedge \vec{X}_c = f(\vec{Y}_u)) \rightarrow \Psi(\vec{X}_c, \vec{Y}_u))$ is valid.*

Proof. Let $m \doteq |\vec{Y_u}|$ and V be the set of $m + 1$ vertexes of $\sigma(\vec{Y_u})$. By definition of simplex, $\sigma(\vec{Y_u})$ is the convex hull of the points in V. P is weakly controllable by

assumption, therefore for each $v_i \in V$, there exists a point t_i that extends v_i in the space of $\vec{X}_c \cup \vec{Y}_u$ such that $t_i \in \Psi(\vec{X}_c, \vec{Y}_u)$. Let T be $\{t_i | v_i \in V\}$.

Let f be a linear strategy $A \cdot \vec{Y}_u + \vec{c}$, such that for each controllable time point $b_i \in X_c$, $b_i = A_{i,1}y_1 + \cdots + A_{i,m}y_m + c_i$ is the hyperplane passing through all the points $t_i \in T$. Such a hyperplane exists and is unique because it is the m-hyperplane containing the $m+1$ not-collinear [14] points $t_i \in T$ (points in T are not collinear as they are the results of an extension of the points in V that are not collinear because are the $m+1$ vertexes of a simplex).

We now prove that f is a strategy such that $\forall \vec{Y}_u.(\sigma(\vec{Y}_u) \wedge \vec{X}_c = f(\vec{Y}_u)) \rightarrow \Psi(\vec{X}_c, \vec{Y}_u)$ is valid by showing that the hyperplane $b_i = A_{i,1}y_1 + \cdots + A_{i,m}y_m + c_i$ is contained in $\Psi(\vec{X}_c, \vec{Y}_u)$ for each $\vec{Y}_u \models \sigma(\vec{Y}_u)$. Since the hyperplane contains all the $t_i \in T$, for each point k in the convex hull of V, the hyperplane computed in k is contained in $\Psi(\vec{X}_c, \vec{Y}_u)$ because of the convexity of $\Psi(\vec{X}_c, \vec{Y}_u)$. This proves the thesis because $\sigma(\vec{Y}_u)$ is the convex hull of the points in V. □

In order to exploit Theorem 5 we need to be able to split the uncontrollable space into simplexes. Doing so would allow us to split the problem of finding a piecewise-linear strategy for the whole problem in the problem of finding linear strategies for each simplex and then combine them.

The uncontrollable region $\Gamma(\vec{Y}_u)$ is a hyper-rectangle, and the minimum number of simplexes needed to cover an hyper-rectangle is an open mathematical problem [20]. However, it is known that any hyper-rectangle in m dimensions can be split in a factorial number of simplexes ($m!$). For example, a rectangle can be split in 2 triangles, and a rectangular cuboid can be covered by 6 tetrahedrons.

Given $\Gamma(\vec{Y}_u)$, we can obtain all the simplexes using the following idea. Suppose that the bounds for all the uncontrollable variables are $[0, 1]$. Then, let R be the region satisfying the sequence of inequalities $y_{p_1} \leq y_{p_2} \leq \cdots \leq y_{p_m}$ where $(p_1, \ldots, p_m)$ is a permutation of $(1, \ldots, m)$ and m is the number of uncontrollable duration variables. It can be shown that R is a simplex and that the simplexes generated for all the permutations form a partition of the uncontrollable space [20]. In the general case, when we can have arbitrary bounds, we apply the very same idea, permuting the variables and considering the inequalities arising from considering the concrete lower/upper bounds.

Using this approach, we have to enumerate all the permutations of the uncontrollable variables. Thus, the number of considered simplexes is factorial in the number of uncontrollable variables (i.e. $|Y_u|!$).

For each simplex it is possible to find a linear strategy separately by enforcing a hyperplane to satisfy the problem constraints in all the simplex vertexes. In Figure 8 we depicted an example of this idea for the running example problem.

Algorithm 4 shows the pseudo-code for extracting a piecewise linear strategy by enumerating all the simplexes and finding a linear strategy for every simplex. The computational complexity of this algorithm is factorial due to the enumeration of all the ($|Y_u|!$) simplexes.

In the pseudo-code, the function GETMAXIMALSIMPLEXES enumerates a sequence of $|Y_u|$-simplexes needed to cover the $\Gamma(\vec{Y}_u)$ polyhedron, while VERTEXENCODING

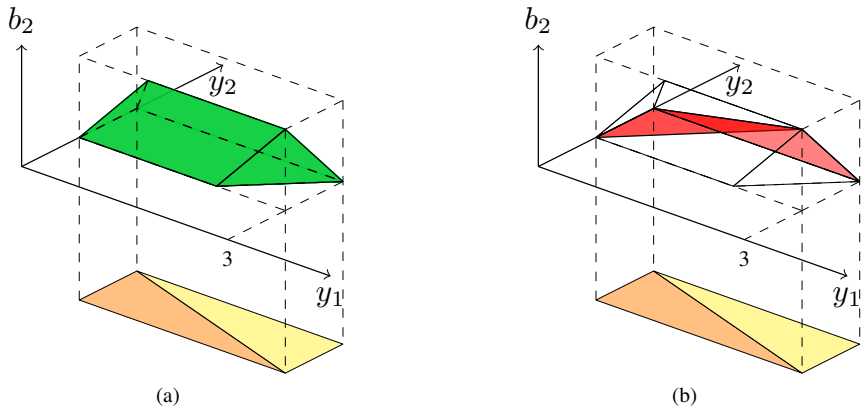


Figure 8: Plot of the running example problem (with b_1 assigned to 0) with a partition of the space of uncontrollable durations. The space of uncontrollable durations is split in two triangles, depicted in yellow and orange. In (a) we plot the space of the solutions, while in (b) we draw in red a possible piecewise-linear strategy obtained by using a linear strategy for each triangle.

Algorithm 4 Simplexes Decomposition strategy extraction algorithm

1: **procedure** SIMPLEXESDECOMPOSITION($\Gamma(\vec{Y}_u)$, $\Psi(\vec{X}_u, \vec{Y}_u)$)
2: $f \leftarrow$ GETEMPTYSTRATEGY()
3: **for all** $\sigma(\vec{Y}_u) \in$ GETMAXIMALSIMPLEXES($\Gamma(\vec{Y}_u)$) **do**
4: $f_{sub} \leftarrow$ VERTEXENCODING($\sigma(\vec{Y}_u)$, $\Psi(\vec{X}_u, \vec{Y}_u)$)
5: $f \leftarrow$ ADDPIECETOSTRATEGY(f, $(\sigma(\vec{Y}_u), f_{sub})$)
6: **return** f
7: **end procedure**

returns a linear strategy suitable for the given simplex[5]. We obtain the resulting piecewise-linear strategy f by adding a piece for each simplex by means of the function ADDPIECETOSTRATEGY.

Consider the problem in Figure 6; the algorithm works as follows. We first consider the simplex with vertexes $\{(0,1),(0,2),(3,1)\}$ in the space of y_1 and y_2. A possible linear strategy in this simplex is $f_1 = A_1 \cdot \vec{Y}_u + \vec{c}_1$ as follows.

$$A_1 = \begin{pmatrix} 0 & 0 \\ \frac{1}{3} & 0 \end{pmatrix} \quad \vec{c}_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

We then consider the second maximal simplex with vertexes $\{(0,2),(3,1),(3,2)\}$. A

[5] $(\vec{X}_c, \vec{Y}_u, \sigma(\vec{Y}_u), \Psi(\vec{X}_u, \vec{Y}_u))$ is not a well formed encoding of an STPU, because $\sigma(\vec{Y}_u)$ is not in the shape prescribed by Definition 5. Nevertheless, the VERTEXENCODING algorithm can deal with any convex uncontrollable region.

possible linear strategy in this simplex is $f_2 = A_2 \cdot \vec{Y}_u + \vec{c}_2$ as follows.

$$A_2 = \begin{pmatrix} 0 & 0 \\ 0 & -1 \end{pmatrix} \quad \vec{c}_2 = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$$

The algorithm combines such strategies in a valid piecewise-linear weak strategy f as follows.

$$\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = f(y_1, y_2) \doteq \begin{cases} f_1 & \text{if } (y_2 \leq -\frac{1}{3} y_1 + 2) \\ f_2 & \text{otherwise} \end{cases}$$

6.2.2. Lazy Expansion

To overcome the complexity limitation of the previous approach we developed a second technique, called *Lazy Expansion*, that first selects a simplex in the uncontrollable region and finds a linear strategy in that simplex. Second, we symbolically compute the region of the uncontrollable durations that is satisfied by the computed strategy. In this way, we perform a "widening" of the portion of the uncontrollable space that can be satisfied using the computed linear strategy. This widened region is guaranteed to cover at least the simplex, but it might be larger. We then associate the computed strategy to the resulting region. Finally, we search a new simplex in the remaining part of the space of uncontrollable durations. The algorithm terminates when the space of uncontrollable durations is completely covered. The idea behind the approach is to generalize the strategy found for a particular simplex to cover a wider potion of the space of the uncontrollable durations. The algorithm lazily picks a simplex from the region of the uncontrollable durations to be covered and gets a strategy that is able to cover that particular simplex. We then generalize the applicability of the returned strategy and proceed until we completely cover the uncontrollable space. The main advantage of this algorithm with respect to the previous one is that it is not forced to enumerate all the possible simplexes, because the computed strategy once found is exploited in all the possible points of the space where it is applicable.

Algorithm 5 shows the pseudo-code for extracting a piecewise linear strategy exploiting lazy expansion. The function GETUNCOVEREDSIMPLEX returns any simplex $\sigma(\vec{Y}_u)$ completely contained in the uncontrollable region $\Gamma(\vec{Y}_u)$. At each step we compute the widening of the simplex $o(\vec{Y}_u)$, that is the region in which the computed linear strategy f_{sub} is applicable. In order to symbolically obtain this region, we substitute the $\mathcal{LRA}$ encoding of the strategy f_{sub} in the free constraints $\Psi(\vec{X}_c, \vec{Y}_u)$. In this way, each controllable time point variable $b_i \; in \vec{X}_c$ is replaced by the linear term that computes it according to f_{sub}, and we are left with a formula defined over $\vec{Y}_u$. Each model of such formula, is a point in the uncontrollable region for which the application of f_{sub} fulfills the free constraints of the problem. We use this procedure to create "bigger" pieces and reduce the number of iterations of the algorithm.

In general, this algorithm is not guaranteed to terminate. In fact, termination can be assured with the following two requirements. First, each region $\sigma(\vec{Y}_u)$ covers a non-empty volume of the space of the uncontrollable durations. This is needed for progression: the piece of strategy computed at each step is guaranteed to cover at least the simplex that originated it (at each step $\sigma(\vec{Y}_u) \models_{\mathcal{LRA}} o(\vec{Y}_u)$). The second requirement is to have progression, that is we disallow infinite decomposition chains

Algorithm 5 Lazy piecewise-linear strategy extraction

procedure LAZYEXPANSION($\Gamma(\vec{Y}_u)$, $\Psi(\vec{X}_c, \vec{Y}_u)$)
  $f \leftarrow$GETEMPTYSTRATEGY()
  $\eta(\vec{Y}_u) \leftarrow \Gamma(\vec{Y}_u)$
  for all $y_j \in \vec{Y}_u$ **do**
    SMT.DECLAREVAR($y_i, \mathbb{R}$)
  while SMT.SOLVE($\eta(\vec{Y}_u)$) **do**
    $\sigma(\vec{Y}_u) \leftarrow$ GETUNCOVEREDSIMPLEX($\eta(\vec{Y}_u)$)
    $f_{sub} \leftarrow$ VERTEXENCODING($\sigma(\vec{Y}_u)$, $\Psi(\vec{X}_c, \vec{Y}_u)$)
    $o(\vec{Y}_u) \leftarrow \Psi(f_{sub}(\vec{Y}_u), \vec{Y}_u)$
    $f \leftarrow$ ADDPIECETOSTRATEGY(f, $(\eta(\vec{Y}_u) \wedge o(\vec{Y}_u), f_{sub})$)
    $\eta(\vec{Y}_u) \leftarrow \eta(\vec{Y}_u) \wedge \neg o(\vec{Y}_u)$
  return f
end procedure

for finite regions. If we avoid empty regions and infinite subdivisions of finite regions, we will eventually get to the empty region, and thus to unsatisfiability and termination.

In our implementation, we do not guarantee termination. However, the algorithm correctly terminated in all our experiments and in many cases it performed much faster than the SIMPLEXESDECOMPOSITION algorithm. One possibility to guarantee the termination would be to hybridize this algorithm with the SIMPLEXESDECOMPOSITION approach by bounding the number of loops of the LAZYEXPANSION procedure or to take a portfolio approach.

One key issue for the efficiency of this approach resides in finding good simplexes to cover a (possibly non-convex) region $\eta(\vec{Y}_u)$. Defining what a "good" simplex is for the algorithm performance is a non-trivial task because the aim is to terminate with a minimum number of iterations, and thus to obtain a decomposition of $\Gamma(\vec{Y}_u)$ in a minimal set of $o_i(\vec{Y}_u)$ regions.

6.3. Linear Strategies for DTPU

We now consider the DTPU problem class and we provide algorithms for strategy synthesis starting from the linear case.

A way of computing the elements of matrix A and vector $\vec{c}$ for a linear strategy is an encoding of the constraints in the theory of Nonlinear Real Arithmetic over Polynomials ($\mathcal{NRA}$). In fact, we can compactly express the properties of each entry of A and $\vec{c}$ by imposing constraints on polynomials. Equation 10 is an encoding into $\mathcal{NRA}$ for extracting a linear strategy for any TPU problem in a single check.

$$\begin{aligned} &\exists A_{1,1}, \ldots, A_{1,m}, c_1, \\ &\quad \ldots \\ &A_{n,1}, \ldots, A_{n,m}, c_n. \forall \vec{Y}_u. \left(\Gamma(\vec{Y}_u) \rightarrow \Psi[A \cdot \vec{Y}_u + \vec{c}/\vec{X}_c](\vec{Y}_u)\right) \end{aligned} \tag{10}$$

The idea is to let the solver search for the $A_{i,j}$ and c_i coefficients of the linear combination of $\vec{Y}_u$ that represent the set of hyper-planes that are strategies for each $b_i \in \vec{X}_c$. If the solver reports unsatisfiable, it means that no linear strategy exists for the given problem. This approach directly follows from the definition of linear strategy and is applicable to the entire spectrum of temporal problems with uncertainty because no assumption on the convexity of the search space is made.

As an example we show the encoding of Equation 10 applied to the TPU in Figure 6, to remark the fact that no linear strategy exists for this problem.

$$
\begin{aligned}
&\exists A_{1,1}, A_{1,2}, A_{2,1}, A_{2,2}, c_1, c_2 . \forall y_1, y_2 . \\
&((y_1 \geq 0) \wedge (y_1 \leq 3) \wedge (y_2 \geq 1) \wedge (y_2 \leq 2)) \rightarrow \\
&(((A_{2,1} y_1 + A_{2,2} y_2 + c_2) - (A_{1,1} y_1 + A_{1,2} y_2 + c_1) \geq 0) \wedge \\
&\quad (((A_{1,1} y_1 + A_{1,2} y_2 + c_1) + y_1) - (A_{2,1} y_1 + A_{2,2} y_2 + c_2) \geq 0) \wedge \\
&\quad (((A_{1,1} y_1 + A_{1,2} y_2 + c_1) + y_1) - ((A_{2,1} y_1 + A_{2,2} y_2 + c_2) + y_2) \leq 1) \wedge \\
&\quad (((A_{2,1} y_1 + A_{2,2} y_2 + c_2) + y_2) - (A_{1,1} y_1 + A_{1,2} y_2 + c_1) \leq 2))
\end{aligned}
$$

The running example problem is an STPU, but the approach presented here is more general. In fact, given a procedure that is able to decide $\mathcal{NRA}$ formulae with arbitrary disjunctions we can deal with DTPU as well. For example, the Cylindrical Algebraic Decomposition (CAD) procedure [12] can deal with this kind of formulae.

6.4. Piecewise-linear Strategies for DTPU

In this section we analyze the synthesis of a piecewise-linear strategy for a DTPU. When dealing with a DTPU, the convexity assumptions holding for the STPU case are not valid anymore. We present two algorithms for the strategy synthesis in the DTPU problem class. The "skin crawler" method searches a strategy by considering the faces of the DTPU solution space considered as a polyhedron. The "convex region enumerator" approach, instead, decomposes the DTPU in a number of convex regions and applies the techniques for the STPU problem class on each of them.

6.4.1. Skin crawler

An intuition that can be exploited to synthesize a weak strategy for the DTPU problem class is obtained from the proof of Theorem 3. The idea is to iterate on the faces of $\Psi(\vec{X}_c, \vec{Y}_u)$ and to project each of them in the space of $\vec{Y}_u$ until the entire $\Gamma(\vec{Y}_u)$ region is covered by the projections. Such an iteration can be done efficiently by exploiting the optimization features of many modern SMT solvers.

The top-level procedure, shown in Algorithm 6, iterates over the faces and extracts a linear strategy for each face, accumulating this result in a piecewise-linear strategy.

The face extraction procedure (Algorithm 7) starts by extracting all the equalities from the free constraints. Since the free constraints are made of non-strict inequalities, we aim at extracting the skin of the free constraints by considering the equality $a - b = k$ derived from $a - b <= k$. The algorithm uses an optimization procedure that maximizes the number of equalities satisfied at each step represented by the variable $satEqualities$. In this way, considering the conjunction of all the satisfied equalities, we first explore the vertexes, then the edges and finally the faces. The conjunction of

Algorithm 6 Skin-based strategy extraction for DTPU

1: **procedure** SKINCRAWLER($\Gamma(\vec{Y_u})$, $\Psi(\vec{X_c}, \vec{Y_u})$)
2: $f \leftarrow$ GETEMPTYSTRATEGY()
3: **for all** $(o(\vec{Y_u}), f_{sub}) \in$ GETFACESTRATEGIES($\Gamma(\vec{Y_u})$, $\Psi(\vec{X_c}, \vec{Y_u})$) **do**
4: $f \leftarrow$ADDPIECETOSTRATEGY(f, $(o(\vec{Y_u}), f_{sub})$)
5: **return** f
6: **end procedure**

equalities is actually a system of linear equalities representing a face. In order to extract a strategy from a face, we transform a conjunction of linear equalities into matrix form. As an optimization we discard systems that have dimension lower than the number of uncontrollable durations. This prevents the creation of pieces representing regions having null volume.

The algorithm termination depends only on the termination of the GETFACESTRATEGIES procedure. The procedure is guaranteed to terminate because at each step we add a new clause to $\chi(\vec{X_c}, \vec{Y_u})$ that forces at least one equality that was positive in the found model to be false. Therefore we can have at most an exponential number of cycles with respect to the number of equalities in $Equalities(\Psi(\vec{X_c}, \vec{Y_u}))$.

6.4.2. Convex region enumerator

We can exploit the possibility of generating a strategy for the convex case by enumerating the convex regions in the space of free constraints.

This idea requires the possibility to deal with a possibly non-convex $\Gamma(\vec{Y_u})$ because the projection of a convex polyhedron space intersected with the non-convex $\Gamma(\vec{Y_u})$ can generate non-convex (and non-rectangular) regions. The LAZYEXPANSION algorithm is able to deal with such constraints, because no constraint is imposed on the shape of $\Gamma(\vec{Y_u})$ during the algorithm execution.

In this case, we need to enumerate a set of convex formulae $\{\mu_i(\vec{X_c}, \vec{Y_u}) | i \in [1, I]\}$ such that $\bigvee_{i=1}^{I} \mu_i \Leftrightarrow \Psi(\vec{X_c}, \vec{Y_u})$. Such formulae can be obtained by computing the Disjunctive Normal Form (DNF) of the formula $\Psi(\vec{X_c}, \vec{Y_u})$. From the practical point of view, each disjunct is either an atom of the original formula or its negation. The DNF can be efficiently computed in the SMT framework using an incremental mechanism.

Algorithm 8 shows the pseudo-code for the Convex Region Enumerator algorithm for the weak strategy synthesis of DTPU problems.

The algorithm works as follows. First it selects any consistent temporal evolution by solving the SMT problem of the conjunction of the contingent and free constraints. Given the consistent model, the algorithm extracts the free constraints atoms that are satisfied and the atoms that are not fulfilled. The region obtained by conjoining all those atoms is a convex (non-necessarily closed) polyhedron. We compute the projection of such a polyhedron in the region of the uncontrollable durations and we compute a strategy for this quasi-STPU problem[6]. The obtained strategy is applied in the covered

[6]The problem is not a proper STPU because the projection can be non-rectangular and the constraints

Algorithm 7 Generates all the faces of $\Psi(\vec{X_c}, \vec{Y_u})$ and converts them in a linear system of equations

1: **procedure** GETFACESTRATEGIES($\Gamma(\vec{Y_u})$, $\Psi(\vec{X_c}, \vec{Y_u})$)
2: **for all** $x_i \in \vec{X_c} \cup \vec{Y_u}$ **do**
3: SMT.DECLAREVAR(x_i, $\mathbb{R}$)
4: $\chi(\vec{X_c}, \vec{Y_u}) \leftarrow \Psi(\vec{X_c}, \vec{Y_u}) \wedge \Gamma(\vec{Y_u})$
5: $satEqualities \leftarrow 0$
6: **for all** $eq_i(\vec{X_c}, \vec{Y_u}) \in Equalities(\Psi(\vec{X_c}, \vec{Y_u}))$ **do**
7: SMT.DECLAREVAR(eqv_i, $\mathbb{R}$)
8: $\chi(\vec{X_c}, \vec{Y_u}) \leftarrow \chi(\vec{X_c}, \vec{Y_u}) \wedge (eq_i(\vec{X_c}, \vec{Y_u}) \rightarrow (eqv_i = 1))$
9: $\chi(\vec{X_c}, \vec{Y_u}) \leftarrow \chi(\vec{X_c}, \vec{Y_u}) \wedge (eq_i(\vec{X_c}, \vec{Y_u}) \vee (eqv_i = 0))$
10: $satEqualities \leftarrow satEqualities + eqv_i$
11: $faces \leftarrow \emptyset$
12: **while** SMT.SOLVEMAXIMIZING($\chi(\vec{X_c}, \vec{Y_u})$, $satEqualities$) = SAT **do**
13: $system \leftarrow \{eq_i(\vec{X_c}, \vec{Y_u}) \in Equalities(\Psi(\vec{X_c}, \vec{Y_u}))|\mu \models eq_i(\vec{X_c}, \vec{Y_u})\}$
14: $(M\vec{t} = \vec{d}) \leftarrow$ CONVERTTOLINEARSYSTEM(system)
15: $bases \leftarrow$ GETBASES($M\vec{t} = 0$)
16: **if** $|bases| \geq |\vec{Y_u}|$ **then**
17: $(A, \vec{c}) \leftarrow$ TOLINEARSTRATEGY(M)
18: $o(\vec{Y}) \leftarrow \Psi(A \cdot \vec{Y_u} + \vec{c}, \vec{Y_u})$
19: $faces \leftarrow faces \cup \{(o(\vec{Y}), A \cdot \vec{Y_u} + \vec{c})\}$
20: $\chi(\vec{X_c}, \vec{Y_u}) \leftarrow \chi(\vec{X_c}, \vec{Y_u}) \wedge (\bigvee_{eq_i(\vec{X_c}, \vec{Y_u}) \in system} \neg eq_i(\vec{X_c}, \vec{Y_u}))$
21: **return** $faces$
22: **end procedure**

region that is removed from the problem together with the polyhedron. This ensures the algorithm termination, because at each step we remove a model of the Boolean abstraction from the $\rho(\vec{X_c}, \vec{Y_u})$ formula.

In the pseudo-code, the function PROJECT performs a quantifier elimination in order to compute the projection of a given polyhedron onto the given space and is used to compute the uncontrollable region that is covered by the selected polyhedron.

Termination is not guaranteed, because we internally use the LAZYEXPANSION algorithm that is incomplete; however, also in this case, the algorithm terminated in all the benchmarks.

7. Experimental Evaluation

In order to empirically test the effectiveness of the proposed approaches, we implemented a tool for deciding weak controllability and synthesizing weak strategies

can contain strict inequalities. However the LAZYEXPANSION algorithm is able to deal even with such degenerated problems.

Algorithm 8 Convex Region Enumeration strategy extraction for DTPU

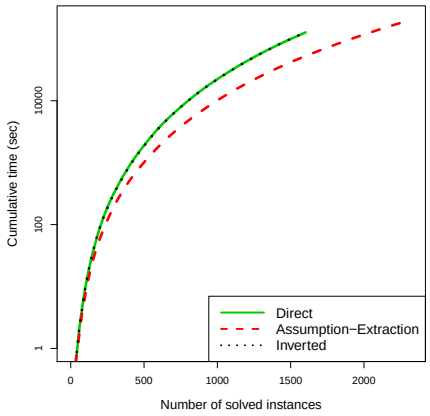
procedure CONVEXREGIONENUMERATOR($\Gamma(\vec{Y_u})$, $\Psi(\vec{X_c}, \vec{Y_u})$)
  for all $x_i \in \vec{X_c} \cup \vec{Y_u}$ **do**
    SMT.DECLAREVAR(x_i, $\mathbb{R}$)
  $f \leftarrow$ GETEMPTYSTRATEGY()
  $\rho(\vec{X_c}, \vec{Y_u}) \leftarrow \Psi(\vec{X_c}, \vec{Y_u}) \wedge \Gamma(\vec{Y_u})$
  while SMT.SOLVE($\rho(\vec{X_c}, \vec{Y_u})$) = SAT **do**
    $\mu \leftarrow$ SMT.GETMODEL()
    $\delta(\vec{X_c}, \vec{Y_u}) \leftarrow \top$
    for all $\alpha(\vec{X_c}, \vec{Y_u}) \in Atoms(\Psi(\vec{X_c}\vec{Y_u}))$ **do**
      if $\mu \models \alpha(\vec{X_c}, \vec{Y_u})$ **then**
        $\delta(\vec{X_c}, \vec{Y_u}) \leftarrow \delta(\vec{X_c}, \vec{Y_u}) \wedge \alpha(\vec{X_c}, \vec{Y_u})$
      else
        $\delta(\vec{X_c}, \vec{Y_u}) \leftarrow \delta(\vec{X_c}, \vec{Y_u}) \wedge \neg(\alpha(\vec{X_c}, \vec{Y_u}))$
    $o(\vec{Y_u}) \leftarrow$ PROJECT($\delta(\vec{X_c}, \vec{Y_u})$, $\vec{Y_u}$)
    $f_{sub} \leftarrow$ LAZYEXPANSION($o(\vec{Y_u})$, $\delta(\vec{X_c}, \vec{Y_u})$)
    $f \leftarrow$ ADDPIECETOSTRATEGY(f, ($o(\vec{Y_u})$, f_{sub}))
    $\rho(\vec{X_c}, \vec{Y_u}) \leftarrow \rho(\vec{X_c}, \vec{Y_u}) \wedge \neg\delta(\vec{X_c}, \vec{Y_u}) \wedge \neg o(\vec{Y_u})$
  return f
end procedure

for a TPU. Our tool is implemented in Python. It reads a TPU problem, and applies to it the portfolio of encodings and algorithms we presented in this paper. The tool can synthesize explicit strategies as C++ functions (taking in input a situation), that can be compiled and linked in any program. We used the Z3 [17] SMT solver for the weak controllability decision problem; we rely on the Python API provided by the MATHSAT5 [7] SMT solver for all the strategy-synthesis techniques.

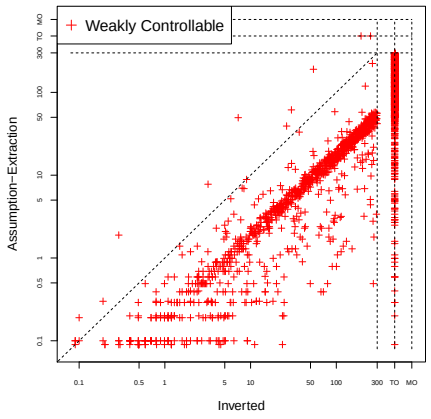
We tested the tool on a set of benchmarks described in detail below. We remark that, as far as our knowledge is concerned, there are no competitor tools or solvers able to deal with the weak controllability decision problem, nor with the synthesis of a weak strategy. Thus, in the experimental evaluation, we do not compare with any other tool or approach. All the experiments have been performed on a Scientific-Linux server equipped with two quad-core Xeon processors @ 2.70GHz. We used a memory limit of 2GB, a time-out of 300 seconds and we used sequential, single-core computation only. The tool, together with all the benchmarks and the results of the evaluation, can be downloaded from `https://es.fbk.eu/people/roveri/tests/aij-weakcontr`.

The randomly-generated benchmarks were obtained by modifying the generator of temporal problems presented in [2] by introducing uncertainty in the problem: each constraint introduced by the consistency problem generator is turned into a contingent constraint with a given probability, and its destination node is considered as uncontrollable. We used random instance generators because they are typically used in the literature (e.g. [2]), and because they can be easily scaled to stress the solvers.

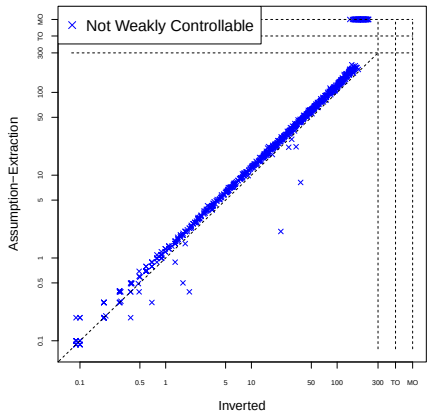
We tested the decision problem encoding over a set of 2442 randomly generated DTPU, TCSPU and STPU instances, with a number of time points ranging from 6 to 20000. For the evaluation of the strategy-extraction techniques, we used 1354 weakly controllable STPU benchmarks and 2112 weakly controllable DTPU instances ranging from 4 to 50 time points.



(a)



(b)



(c)

Figure 9: Results for the decision problem encodings solved using the Z3 SMT solver. (a) reports the cumulative time (in logarithmic scale) cactus plot; (b) and (c) show the scatter plots of INVERTED vs. ASSUMPTION-EXTRACTION encodings divided in weakly controllable (b), and not weakly controllable (c). The TO line denotes the instances that reached the time out, while MO indicates instances that hit the memory limit.

7.1. Decision problem

The results of checking the decision problem over the set of TPUs are plotted in Figure 9. The cactus plot (a) reports, in the horizontal axis, the number of solved instances and, on the vertical axis, the cumulative time, in logarithmic scale, taken by the SMT solver for each encoding. For example, the ASSUMPTION-EXTRACTION encoding takes about 10000 seconds to solve the easiest 750 instances. We compared the formulation of Proposition 1 called DIRECT with the INVERTED and ASSUMPTION-EXTRACTION encodings.

The figure highlights the fact that Z3 performs much better when the ASSUMPTION-EXTRACTION encoding of the problem is considered: in fact, this approach is able to solve, in less time, a higher number of instances with respect to the INVERTED and DIRECT encodings. The DIRECT encoding performs almost identically to the INVERTED one. This behavior is due to the fact that the INVERTED encoding has the same shape as the DIRECT one. The only difference is the negation of the DIRECT encoding, that does not affect the solver performance.

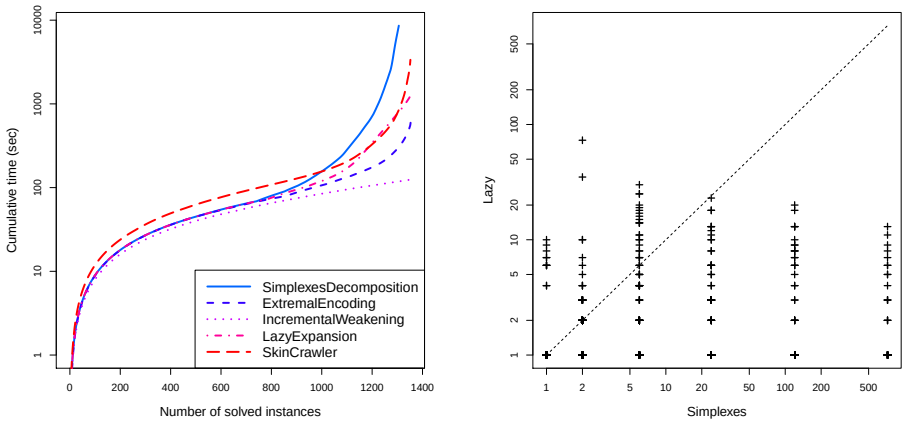
In Figures 9 (b) and 9 (c) we reported the scatter plots comparing the performances of the ASSUMPTION-EXTRACTION with the INVERTED encodings, distinguishing between weakly controllable and non weakly controllable instances. We note that, in the weakly controllable case, the ASSUMPTION-EXTRACTION encoding outperforms the INVERTED encoding in most of the benchmarks. For non weakly controllable instances, the two encodings perform similarly in terms of speed. However, the INVERTED encoding is able to solve 86 instances that are unsolvable by the ASSUMPTION-EXTRACTION encoding due to the imposed memory limit.

7.2. STPU strategy synthesis

The results for the evaluation of the strategy-extraction techniques for the 1354 STPU benchmarks are reported in Figure 10 (a). The plot considers only those benchmarks that admit a linear strategy, and compares the four different approaches. The plot clearly shows that for linear strategies, the INCREMENTALWEAKENING approach outperforms all the others. The SIMPLEXESDECOMPOSITION method quickly explodes due to the factorial complexity of simplexes enumeration. Although the techniques for piecewise-linear strategy extraction are penalized as they are strictly more general than the others, the plot shows that LAZYEXPANSION approach is much faster than the SIMPLEXESDECOMPOSITION. In Figure 10 (b) we plotted the number of "pieces" of the strategies for the LAZYEXPANSION and SIMPLEXESDECOMPOSITION methods. The plot clearly shows that, although for small problems the LAZYEXPANSION approach generates additional, unneeded "pieces", when the problem size increases the number of "pieces" identified by the LAZYEXPANSION method is much smaller than for the SIMPLEXESDECOMPOSITION one. In general, the LAZYEXPANSION approach has a huge gain in performances and in strategy size.

7.3. DTPU strategy synthesis

In Figure 11 we report the results on the DTPU problem class. The plots clearly show that the CONVEXREGIONENUMERATOR algorithm performs better than the SKINCRAWLER one. This is because of two main reasons. First, the SKINCRAWLER approach solves a costly minimization problem and has to traverse all the faces of the



(a) (b)

Figure 10: Results for STPU linear strategy extraction problem. In (a) we plotted the cumulative cactus plot of the strategy extraction time for the different algorithms proposed in this paper, while in (b) we compared the number of pieces for piecewise-linear algorithms expressed as the number of split regions.

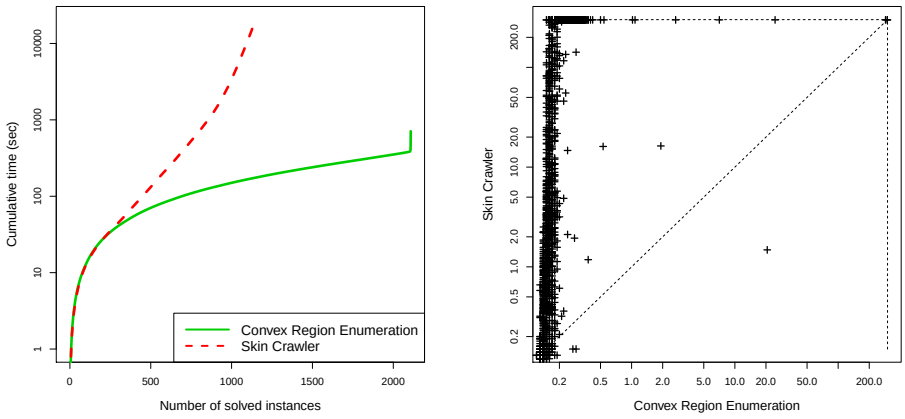
space of free constraints while the CONVEXREGIONENUMERATOR algorithm applies the cheap LAZYEXPANSION approach to each convex region that is generated by a single call to the SMT solver. Second, the linear strategy generated by the LAZYEXPANSION approach is generalized and applied wherever possible, therefore if the problem allows for a linear strategy the CONVEXREGIONENUMERATOR algorithm is able to quickly synthesize it, while the SKINCRAWLER has to enumerate enough faces to cover the entire uncontrollable space.

There is an interesting peak on the rightmost part of the CONVEXREGIONENUMERATOR curve in the plot. This is due to a particular instance that is solved in 287.28 seconds generating a strategy with 3750 pieces (thus using the same number of iterations to terminate). This is one example in which the splitting done by the LAZYEXPANSION approach gets lost in splitting the uncontrollable space in simplexes.

Finally, we did not experimented the effectiveness of the $\mathcal{NRA}$ encoding for two reasons. First, since a linear strategy is not guaranteed to exist for STPUs, it is also not guaranteed to exist in DTPUs. Second, the $\mathcal{NRA}$ approach needs a solver supporting the quantification over the real polynomial arithmetic (the full polynomial $\mathcal{NRA}$ theory), and to the best of our knowledge, no SMT solver fully supports this theory due to its complexity, even if the problem is decidable [12].

7.4. Strategy execution

In this paper we proposed a number of approaches to synthesize weak strategies arguing that their execution is practically more efficient than solving the individual problems without uncertainty obtained by projecting the uncertainty away. In this section we provide experimental evidence supporting this claim on a number of STPU and DTPU instances.



(a) (b)

Figure 11: Results for strategy extraction problem in the DTPU problem class. In (a) we plotted the cumulative cactus plot of the solving time for the CONVEXREGIONENUMERATOR and the SKINCRAWLER algorithms while (b) is a scatter plot of the data.

We conducted the experiment as follows. For each TPU, we randomly generated 1000 situations, represented as complete assignments for the uncontrollable durations. We implemented the IMPLICIT-SMT and IMPLICIT-SMT-INCREMENTAL general strategies described in section 5.1 using the MATHSAT5 SMT solver. Other TP solvers can, in principle, be employed to create implicit strategies, but SMT solvers showed to be effective in dealing with consistency problems [11]. For this reason, and for the lack of publicly available implemented solvers, we limited our experimentation to the SMT based techniques described in section 5.1.

In addition, we considered three ways to compile in machine code the problem-specific strategies generated by the algorithms presented in this paper. We translated the linear or piecewise-linear strategy synthesized by any of our algorithms into a C++ code. The translation for linear strategies is straight-forward: we create a function that takes in input a numeric value for each uncontrollable duration and we compute the output of the strategy by solving the matrix multiplication as described in section 5.2. Given a piecewise-linear strategy, we translate it using a sequence of `if` statements, one for each piece. The condition of each `if` is the transposition in C++ syntax of the piece condition. Each conditional statement returns the value computed by the translation of the linear strategy relative to the particular piece. We used three different datatypes to represent numeric values and perform the arithmetic operations. In particular, we used the (finite-precision) C++ `float` and `double` and the GNU-GMP library for arbitrary precision arithmetic[7]. The `float` and `double` datatypes are a finite-precision representation of rational numbers. As such, they suffer from both nu-

[7]The GNU Multiple Precision Arithmetic Library `https://gmplib.org/`.

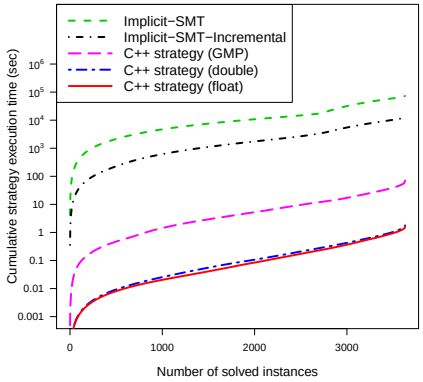


Figure 12: Results for strategy execution: for each problem, the generated strategy is executed on 1000 randomly generated situations. the plot considers all the STPU and DTPU randomly generated problems. The cactus plot shows the number of solved instances on the x axis and the accumulated time to solve them in the y axis.

meric stability and rounding problems that may, in principle, cause unsoundness in the strategy output. On the other hand, GNU-GMP is the same library employed by the MATHSAT5 SMT solver and does not suffer from any kind of numeric stability or rounding problems.

Figure 12 shows the results of the comparison: in our experiments the explicit strategies outperform projection-based implicit strategies. IMPLICIT-SMT-INCREMENTAL performs better than IMPLICIT-SMT, thanks to the incrementality feature of the SMT solver, but the explicit strategies bring a significant speedup on all the instances. Arbitrary-precision arithmetic (that fairly compares with the SMT precision) outperforms projection-based techniques by two orders of magnitude. The compiled strategies with native C++ datatypes perform even better, but the numerical stability problems can, in principle, lead to unsound results. We checked the output of each technique on each situation in order to assess the soundness, but all the results were correct. Nevertheless, studying under which condition we can guarantee that such finite-precision implementations are correct is subject of future work. We highlight that the compilations using native C++ datatypes can be translated to Boolean circuits, and open for the possibility to create very efficient hardware implementations of these strategies.

8. Related Work

The literature in temporal problems is vast. Starting from the seminal paper of Dechter [18] describing STP and TCSP, many authors worked in the field of temporal reasoning without uncertainty [2, 36]. Armando et al. [2] were the first proposing the use of SMT in the field of temporal problems tackling the DTP problem class. This

work focuses on temporal problems without uncertainty, and is thus limited to checking the consistency of a given TP.

Fargier and Vidal introduced the concept of STPU and described the three controllability levels [38]. This work has been extended for TCSPU and DTPU in [32, 37].

Concerning strong controllability, this work already contained a polynomial algorithm for schedule synthesis of STPU. For TCSPU and DTPU, the strong controllability problem has been studied, from a theoretical point of view, by Peintner et al. in [32]; a number of techniques that leverage SMT solvers to obtain a practical solver are presented in our previous work [11].

For the decision problem of weak controllability, in [38] a basic algorithm for the STPU class is presented. In [37], Venable et al. approached the problem of deciding weak controllability of DTPU using an explicit algorithm that enumerates the STPU components of a DTPU. In this work we also tackle the DTPU class, but we exploit symbolic techniques to avoid this explicit enumeration, delegating it to the SMT solver, that can exploit advanced mitigation techniques, such as variable selection heuristics and conflict learning. In addition, both [38] and [37] take a purely theoretical point of view of the problem, while here we formalize and encode the weak controllability decision problem in the SMT framework to obtain practical experimental results.

The only work in the literature for the strategy extraction problem for weak controllability is our previous work [10]. In [10] we presented the algorithms for linear and piecewise-linear strategy extraction for the STPU problem class. In this paper, we cover the more general DTPU problem class, considering the strategy synthesis problem in presence of disjunctions. We provide a more detailed theory discussion, we add the proof of existence of piecewise linear strategies for any weakly controllable TPU, and proofs and explanations that were not included in [10]. Also, the experimental evaluation has been extended and clarified.

There exists another form of controllability for TPUs, namely dynamic controllability. The key difference is that in the case of dynamic controllability the strategy is not allowed to observe the entire situation, but each decision can only depend on past events. The problem of dynamic controllability for STPU is a widely studied problem [29, 28, 27, 21, 22], in fact many algorithms have been devised for deciding the dynamic controllability of a STPU. For the disjunctive problem classes, in [37], Venable et. al. present an algorithm for deciding dynamic controllability for TCSPU, while in [8] we, Hunsberger and Poseanto present a decision procedure for DTPUs based on a reduction of the dynamic controllability problem to the reachability problem in a Timed Game Automaton. Also in the field of dynamic controllability, strategy synthesis is a relevant topic: many authors presented implicit strategies for the runtime scheduling of time points [27, 22] for STPU. Those strategies require on-line reasoning and are based on generating networks that can be dispatched by a runtime algorithm using constraint propagation techniques. Recently, we started a research line aiming at synthesizing explicit strategies for dynamic controllability in the STPU problem class [9] and in the DTPU problem class [8]. In these works, we present encodings of the dynamic controllability problem into reachability games in Timed Gamed Automata for which constructive solution techniques exist. The strategies generated can be converted in implicit strategies for the TPU that are similar to the one we synthesize in this paper, for weak controllability.

9. Conclusions and Future Work

In this paper we presented the first comprehensive approach to the problem of weak controllability for TPUs. We cover the problem in its full generality, in the case of TPUs with disjunctions. We work in a logic-based framework, that relies on SMT techniques to achieve an efficient implementation. We make the following contributions: we provide the first effective procedure for deciding the weak controllability of DTPU; we investigate the problem of repeated execution; we propose various constructive forms of strategy extraction. The experimental evaluation shows the feasibility of the method, and demonstrates dramatic speed ups on explicit over implicit strategy execution.

In the future, we will investigate the following research lines. The "incremental weakening" approach for linear strategy extraction and the "lazy decomposition" approach for piecewise-linear strategy extraction are strongly influenced by the selection heuristics. We will investigate the possibility of using topological information for the generation of effective subsets of $\vec{y}$ in "incremental weakening", and the use of extremal simplexes in the "lazy decomposition" approach. We plan to study the applicability to the strategy construction problem of SMT proof-extraction techniques: the capability of modern solvers to extract proofs of unsatisfiability could provide a way to extract the strategy while proving weak controllability. In addition, determining under which conditions a finite state representation is guaranteed to be applicable, can open the way to more efficient and even hardware-implemented strategies. Finally, we believe that the techniques we described can pave the way to efficient explicit strategies for dynamic controllability, where strategies are required to rely only on the observation of events that have already occurred. In particular, we plan to combine the approaches to strong and weak controllability, to explore the continuum in between.

References

[1] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of ACM*, 26(11):832–843, 1983.

[2] Alessandro Armando, Claudio Castellini, and Enrico Giunchiglia. SAT-based procedures for temporal reasoning. In *ECP*, pages 97–108, 1999.

[3] David Avis and Komei Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete and Computational Geometry*, 8(1):295–313, 1992.

[4] Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of Satisfiability*, pages 825–885. IOS Press, 2009.

[5] Marco Bozzano, Roberto Bruttomesso, Alessandro Cimatti, Tommi Junttila, Silvio Ranise, Peter Van Rossum, and Roberto Sebastiani. Efficient satisfiability modulo theories via delayed theory combination. In *CAV*, pages 335–349, 2005.

[6] Roberto Bruttomesso, Edgar Pek, Natasha Sharygina, and Aliaksei Tsitovich. The OpenSMT solver. In *TACAS*, pages 150–153, 2010.

[7] Alessandro Cimatti, Alberto Griggio, Bastiaan Joost Schaafsma, and Roberto Sebastiani. The MathSAT5 SMT solver. In *TACAS*, pages 93–107, 2013.

[8] Alessandro Cimatti, Luke Hunsberger, Andrea Micheli, Roberto Posenato, and Marco Roveri. Sound-and-complete algorithms for checking the dynamic controllability of temporal networks with uncertainty, disjunction and observation. In *TIME*, 2014.

[9] Alessandro Cimatti, Luke Hunsberger, Andrea Micheli, and Marco Roveri. Using timed game automata to synthesize execution strategies for simple temporal networks with uncertainty. In *AAAI*, pages 2242–2249, 2014.

[10] Alessandro Cimatti, Andrea Micheli, and Marco Roveri. Solving temporal problems using SMT: weak controllability. In *AAAI*, 2012.

[11] Alessandro Cimatti, Andrea Micheli, and Marco Roveri. Solving strong controllability of temporal problems with uncertainty using SMT. *Constraints*, 2014.

[12] George E. Collins and Hoon Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12(3):299–328, 1991.

[13] Scott Cotton, Eugene Asarin, Oded Maler, and Peter Niebert. Some progress in satisfiability checking for difference logic. In *FORMATS/FTRTFT*, pages 263–276, 2004.

[14] H. S. M. Coxeter and Samuel L. Greitzer. Collinearity and concurrence. In *Geometry Revisited*, pages 51–79. Mathematical Association of America Textbooks, 1967.

[15] Martin Davis, George Logemann, and Donald W. Loveland. A machine program for theorem-proving. *Communications of ACM*, 5(7):394–397, 1962.

[16] Leonardo de Moura and Nikolaj Bjørner. Model-based theory combination. *Electronic Notes in Theoretical Computer Science*, 198(2):37–49, 2008.

[17] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *TACAS*, pages 337–340, 2008.

[18] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, 1991.

[19] Bruno Dutertre and Leonardo de Moura. The Yices SMT solver. Tool paper at `http://yices.csl.sri.com/tool-paper.pdf`, 2006.

[20] Robert B. Hughes and Michael R. Anderson. Simplexity of the cube. *Discrete Mathematics*, 158(13):99 – 150, 1996.

[21] Luke Hunsberger. Fixing the semantics for dynamic controllability and providing a more practical characterization of dynamic execution strategies. In *TIME*, pages 155–162, 2009.

[22] Luke Hunsberger. A fast incremental algorithm for managing the execution of dynamically controllable temporal networks. In *TIME*, pages 121–128, 2010.

[23] Hyondeuk Kim, Fabio Somenzi, and HoonSang Jin. Efficient Term-ITE conversion for satisfiability modulo theories. In *SAT*, pages 195–208, 2009.

[24] S.C. Kleene. *Mathematical Logic*. J. Wiley & Sons, 1967.

[25] Rüdiger Loos and Volker Weispfenning. Applying linear quantifier elimination. *Computer Journal*, 36(5):450–462, 1993.

[26] David Monniaux. A quantifier elimination algorithm for linear real arithmetic. In *LPAR*, pages 243–257, 2008.

[27] Paul Morris. A structural characterization of temporal dynamic controllability. In *CP*, pages 375–389, 2006.

[28] Paul Morris and Nicola Muscettola. Temporal dynamic controllability revisited. In *AAAI*, pages 1193–1198, 2005.

[29] Paul Morris, Nicola Muscettola, and Thierry Vidal. Dynamic control of plans with temporal uncertainty. In *IJCAI*, pages 494–502, 2001.

[30] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *DAC*, pages 530–535, 2001.

[31] Greg Nelson and Derek C. Oppen. Simplification by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, 1(2):245–257, 1979.

[32] Bart Peintner, Kristen B. Venable, and Neil Yorke-Smith. Strong controllability of disjunctive temporal problems with uncertainty. In *CP*, pages 856–863, 2007.

[33] Alexander Schrijver. *Theory of Linear and Integer Programming*. J. Wiley & Sons, 1998.

[34] Roberto Sebastiani. Lazy satisability modulo theories. *JSAT*, 3(3-4):141–224, 2007.

[35] Roberto Sebastiani and Silvia Tomasi. Optimization in SMT with $\mathcal{LA}(\mathbb{Q})$ cost functions. In *IJCAR*, pages 484–498, 2012.

[36] Ioannis Tsamardinos and Martha E. Pollack. Efficient solution techniques for disjunctive temporal reasoning problems. *Artificial Intelligence*, 151(12):43 – 89, 2003.

[37] Kristen B. Venable, Michele Volpato, Bart Peintner, and Neil Yorke-Smith. Weak and dynamic controllability of temporal problems with disjunctions and uncertainty. In *COPLAS*, 2010.

[38] Thierry Vidal and Hélène Fargier. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental and Theoretical Artificial Intelligence*, 11(1):23–45, 1999.