Solving Temporal Problems using SMT: Strong Controllability

Alessandro Cimatti, Andrea Micheli, and Marco Roveri
`{cimatti,amicheli,roveri}@fbk.eu`

Fondazione Bruno Kessler — Irst

Abstract. Many applications, such as scheduling and temporal planning, require the solution of Temporal Problems (TP's) representing constraints over the timing of activities. A TP with uncertainty (TPU) is characterized by activities with uncontrollable duration. Depending on the Boolean structure of the constraints, we have simple (STPU), constraint satisfaction (TCSPU), and disjunctive (DTPU) temporal problems with uncertainty.
In this work we tackle the problem of strong controllability, i.e. finding an assignment to all the controllable time points, such that the constraints are fulfilled under any possible assignment of uncontrollable time points. We work in the framework of Satisfiability Modulo Theory (SMT), where uncertainty is expressed by means of universal quantifiers. We obtain the first practical and comprehensive solution for strong controllability: the use of quantifier elimination techniques leads to quantifier-free encodings, which are in turn solved with efficient SMT solvers.
We provide a detailed experimental evaluation of our approach over a large set of benchmarks. The results clearly demonstrate that the proposed approach is feasible, and outperforms the best state-of-the-art competitors, when available.

1 Introduction

Many applications require the scheduling of a set of activities over time, subject to constraints of various nature. Scheduling is often expressed as a *Temporal Problem* (TP), where each activity is associated with two time points, representing the start time and the end time, and with a duration, all subject to constraints. Several kinds of temporal problems have been identified, depending on the nature and structure of the constraints. If the constraints are expressible as a simple conjunction of constraints over distances of time points, then we have the so-called *Simple Temporal Problem* (STP). A more complex class is *Temporal Constraint Satisfaction Problem* (TCSP), where a distance between time points can be constrained to a list of disjoint intervals. Constraints in TCSP's can be seen as a restricted form of Boolean combinations. When arbitrary Boolean combinations are allowed, we have a *Disjunctive Temporal Problem* (DTP). A temporal problem is said to be *consistent* if there exists an assignment for the time points, such that all the constraints are satisfied [1]. Such an assignment is called a *schedule*, and it corresponds to sequential time-triggered programs, that are often used in control of satellites and rovers.

In many practical cases, however, the duration of activities is uncontrollable. TP are thus extended with *uncertainty* in the duration of activities, thus obtaining the classes of STPU, TCSPU and DTPU. As in the case of consistency, we look for a schedule. However, the schedule only determines the start of the activities, and must satisfy the constraints *for all* the uncontrollable durations of the activities. If such a schedule exists, the problem is said to be *strongly controllable* [2].

In this paper, we propose a comprehensive and effective approach to strong controllability of TPU. The approach relies on the Satisfiability Modulo Theory (SMT) framework [3]. This framework provides representation capabilities, based on fragments of first order formulae. Reasoning is carried out within a decidable fragments of first order logic, where interpretations are constrained to satisfy a specific theory of interest (i.e. linear arithmetic). Modern SMT solvers are a tight integration of a Boolean SAT solver, that is highly optimized for the *case split* required by the Boolean combination of constraints, with dedicated constraint solvers for the theories of interest. Moreover, some SMT solvers provide embedded efficient primitives to handle *quantifiers*, and dedicated techniques for quantifier-elimination are available. Several effective SMT solvers are available (e.g. MathSAT [4, 5], Z3 [6], Yices [7], OpenSMT [8]). SMT solving has had increasing applications in many areas including Answer Set Programming (ASP) [9], formal verification [10], and test case generation [11].

We tackle the strong controllability problem of TPUs by reduction to SMT problems, that are then fed into efficient SMT solvers. First, we show how to encode a TPU into the theory of quantified linear Real arithmetic (LRA) and, by leveraging the specific nature of the problem, we optimize the encoding by reducing the scope of quantifiers. The resulting formula can be fed to any SMT solver for (quantified) LRA. Second, we present a general reduction procedure from strong controllability to consistency, based on the application of quantifier elimination techniques upfront. The resulting formulae can be directly fed into any SMT solver for the quantifier-free LRA. This gives the first general comprehensive solver for strong controllability of TPUs. Third, we generalize the results by Vidal and Fargier [2], originally stated for STPU, to the class of TCSPU. In this way, we avoid the use of expensive general purpose quantifier elimination techniques, with significant performance improvements.

The proposed approach has been implemented in a solver based on state-of-the-art SMT techniques. To the best of our knowledge, this is the first solver for strong controllability of TPUs. We carried out a thorough experimental evaluation, over a large set of benchmarks. We analyze the merits of the various encodings, and demonstrate the overall feasibility of the approach. We also compare the proposed approaches with state-of-the-art algorithms on consistency problems. SMT solvers are competitive with, and often outperform, the best known dedicated solving techniques.

Structure of the paper. In sections 2 and 3 we present some technical preliminaries and background about SMT. In section 4 we formally define temporal problems, while in section 5 we present several SMT encodings for consistency. Encodings for strong controllability are described in section 6 and an overview of the related work is given is section 7. We report the results of the performed experimental evaluation in section 8, and in section 9 we draw some conclusions and outline directions for future work.

2 Technical Preliminaries

Our setting is standard first order logic. 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 of arity n to n formulae, or the application of a quantifier to an individual variable and a formula. 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, denoted with $\phi, \psi, \ldots$ We write $\phi(x)$ to highlight the fact that x occurs in ϕ, and $\phi(\vec{x})$ to highlight the fact that each x_i occurs in ϕ.

Substitution is defined in the standard way (see for instance [12]). We write $\phi[t/s]$ for the substitution of every occurrence of term t in ϕ with term s. Let $\vec{t}$ and $\vec{s}$ be vectors of terms, we write $\phi[\vec{t}/\vec{s}]$ for the parallel substitution of every occurrence of t_i (the i-th element of $\vec{t}$) in ϕ with s_i.

We use the standard semantic notion of interpretation and satisfiability. We call *satisfying assignment* or *model* of a formula $\phi(\vec{x})$ a total function μ that assigns to each x_i an element of its domain such that the formula $\phi[\vec{x}/\mu(\vec{x})]$ evaluates to $\top$. A formula $\phi(\vec{x})$ is *satisfiable* if and only if it has a satisfying assignment.

Checking the satisfiability (SAT) of a formula consists in finding a satisfying assignment for the formula. This problem is approached in propositional logic with enhancements of the DPLL algorithm: 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 (see e.g. [13]).

3 Satisfiability Modulo Theories

Given a first-order formula ψ in a decidable background theory T, *Satisfiability Modulo Theory* (SMT) [3] 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 $(x, y, z \in \mathbb{R})$. The formula is satisfiable and a satisfying assignment is $\{x := 5,\ y := 6,\ z := 8\}$. The theory of real arithmetic interprets 3 as a real number and $+, =, <, >, \leq, \geq$ as the corresponding operations and relations over $\mathbb{R}$.

In this work we concentrate on the theory of Linear Arithmetic over the Real numbers (LRA). A formula in LRA is an arbitrary Boolean combination, or universal ($\forall$) and 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. Given two real constants l, u such that $l \leq u$, we denote with $t \in [l, u]$ the formula $l \leq t \wedge t \leq u$. Difference logic (RDL) is the subset of LRA such that atoms have the form $x_i - x_j \bowtie c$. We denote with QF_LRA and QF_RDL the quantifier-free fragments of LRA and RDL, respectively.

An SMT solver [3] 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", where a SAT solver is tightly integrated with a T-solver. 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 $a \wedge (b \vee c)$, where a, b, c 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 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 MathSAT [4] and Z3 [6].

In order to deal with quantifiers in LRA many techniques have been developed and implemented in SMT solvers. Some solvers, like e.g. Z3 [6] natively support quantifiers. However, many SMT solvers cannot deal with them. Several techniques have been developed for removing quantifiers from an LRA formula (e.g. Fourier-Motzkin [14], Loos-Weispfenning [15, 16]): they transform an LRA formula into an *equivalent* QF_LRA formula. These techniques enable for the use of solvers with no native support for quantifiers at a cost that is doubly exponential in time and space in the original formula size [14, 16, 15].

4 Temporal Problems

A Temporal Problem (TP) is a formalism that is used to represent temporal constraints over time-valued variables representing time points. This formalism is expressive enough to express Allen's interval algebra [17] and also quantitative constraints over intervals and time points. Two families of TP's have been presented in literature over the years: TP without uncertainty, in which all the time points can be freely assigned by the solver [1,18]; TP with uncertainty (TPU), in which only some of the time points can be assigned by the solver, while the others are intended to be assigned by an adversary. As such, TPU's can be seen as a form of game between the solver and an adversarial environment [2,19].

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, $X_u \doteq \{e_1, ..., e_m\}$ *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.

$$cc_i \doteq (e_i - b_{j_i}) \in [l_i, u_i] \qquad cf_i \doteq \bigvee_{j=1}^{D_i} (x_{i,j} - y_{i,j}) \in [l_{i,j}, u_{i,j}]$$

such that: $j_i \in [1 \ldots n]$, $l_i, u_i \in \mathbb{R}$, $l_i \leq u_i$, $l_{i,j}, u_{i,j} \in \mathbb{R} \cup \{+\infty, -\infty\}$, $l_{i,j} \leq u_{i,j}$, D_i *is the number of disjuncts for the i-th constraint,* $x_{i,j}, y_{i,j} \in X_c \cup X_u$

Intuitively, time points belonging to X_c are time decisions that can be controlled by the solver, while time points in X_u are under the control of the environment. A similar subdivision is imposed on the constraints: free constraints C_f are constraints that the solver is required to fulfill, while contingent constraints (C_c) are the assumptions that the environment will fulfill. As in [2] we consider only contingent constraints that start with a controllable time point. Thus, each uncontrollable time point is linked by exactly one contingent constraint to a controllable time point. We remark that this assumption does not affect the generality of the formalism, as for each contingent constraint $(e_i - e_j) \in [l, u]$ we can add an artificial new controllable time point b, and add $(b - e_j) \in [0, 0]$ to the free constraints and $(e_i - b)$ to the contingent constraints.

A *TP without uncertainty* is a TPU $(X_c, \emptyset, \emptyset, C_f)$, i.e. the set of uncontrollable time points is empty (from which it also follows that the set of contingent constraints is empty).

Depending on the generality of the constraints in C_c and C_f, three classes of TPU's are identified [19]. Definition 1 in its general form identifies *Disjunctive Temporal Problem with Uncertainty* (DTPU). If each constraint contains at most two time points, the resulting problem is a *Temporal Constraint Satisfaction Problem with Uncertainty* (TCSPU). If each constraint has exactly one disjunct (i.e. $D_i = 1$ for all i), we obtain a *Simple Temporal Problem with Uncertainty* (STPU). Similarly, we can define the corresponding TP without uncertainty (DTP [18], TCSP, and STP [1]).

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.* Checking the consistency of a TPU (X_c, X_u, C_c, C_f) is defined as checking the consistency of the TP without uncertainty $(X_c \cup X_u, \emptyset, \emptyset, C_c \cup C_f)$.

Intuitively, when checking consistency of a TPU, the behavior of the environment is assumed to be "cooperative" with the solver. In this paper, we focus on

Strong Controllability (SC) [2] for a TPU, where the environment is adversarial. SC consists in deciding the existence of a *strong schedule*, i.e. an assignment to controllable time points that fulfills the free constraints under any assignment of uncontrollable time points that satisfies the contingent constraints. A TPU for which there exists a strong schedule is said to be *strongly controllable*.

If a TPU is strongly controllable, it is also consistent. However, the converse does not hold in general. Consider for example the STPU such that $X_c = \{A, B\}$, $X_u = \{C\}$, $C_c = \{(B - A) \in [1, 10]\}$, and $C_f = \{(C - A) \in [0, 12], (C - B) \in [1, 5]\}$. The problem is consistent, and a consistent schedule is $\{A = 0,\ B = 3,\ C = 5\}$. However, the problem is not strongly controllable because if the duration of the interval $B - A = 1$, the window of opportunity for scheduling C is $[2, 6]$, that is disjoint from the window of opportunity when the duration is equal to 10, that is $[11, 12]$. Since $B - A$ is decided by the adversarial environment, there is no strong schedule that allows the solver to win.

5 Encoding of Consistency Problems in SMT

We first focus on the consistency problem, i.e. the case in which there is no uncontrollability. The consistency problem can be reduced to checking the satisfiability of a quantifier-free formula modulo the LRA theory. The temporal problem is consistent if and only if the corresponding SMT formula is satisfiable, and any satisfying assignment for the formula corresponds to a consistent schedule for the problem.

The use of SMT to check the consistency of TP without uncertainty has been investigated in the past (e.g. [20]). Here, consistency checking plays the role of backend for strong controllability. In the following, we present several SMT encodings, that turn out to have different performance in the solvers, depending on the nature of the constraints.

In the following, we assume that a TP $(X_c, \emptyset, \emptyset, C_f)$ is given. The first encoding in SMT of the consistency problem can be directly obtained as follows: for every time point in X_c we introduce a real variable, and we denote with $\vec{X_c}$ the vector of such variables; each constraint in C_f is directly mapped on the corresponding SMT formula; the encoding is the SMT formula shown in Equation 1.

$$\bigwedge_{i=1}^{|C_f|} \bigvee_{j=1}^{D_i} (((x_{i,j} - y_{i,j}) \geq l_{i,j}) \wedge ((x_{i,j} - y_{i,j}) \leq u_{i,j})) \tag{1}$$

This encoding is linear in the size of the original TP, but does not exploit any knowledge on the structure of the problem, and is thus referred to as *naïve encoding*. In particular, we notice that the resulting SMT formula is not in CNF.

In the rest of this section we introduce three optimizations: the switch encoding (applicable to any TP), the switch encoding with mutual exclusion and the hole encoding (both for TCSPs only). The *switch encoding* performs a CNF conversion of the formula in Equation 1 by means of a polarity-based CNF labeling conversion [21]. To this extent, we introduce $\sum_{i=0}^{|C_f|} D_i$ Boolean "switch" variables $s_{i,j}$, and the resulting encoding is the one in Equation 2.

$$\begin{aligned}\textstyle\bigwedge_{i=1}^{|C_f|}((\bigwedge_{j=1}^{D_i}((\neg s_{i,j} \vee ((x_{i,j} - y_{i,j}) \geq l_{i,j})) \wedge \\ (\neg s_{i,j} \vee ((x_{i,j} - y_{i,j}) \leq u_{i,j})))) \wedge (\bigvee_{j=1}^{D_i} s_{i,j}))\end{aligned} \tag{2}$$

This encoding is also linear in the size of the original TP, and it directly produces a CNF formula. We notice that the clauses involving theory atoms are binary; furthermore, if a switch variable is assigned to false, the corresponding clauses are satisfied without any theory reasoning. These factors have a positive impact on the performance of the SMT solver.

If we focus on the TCSP class, we can exploit the problem structure to further improve our encodings. In TCSP each constraint is composed of disjuncts of the form $t \in [l_j, u_j]$, where t is the difference of two variables, and for all j, $l_j \leq u_j$ and $u_j < l_{j+1}$. Clearly, the disjuncts are mutually exclusive. However, with the previous encoding it is left to the solver to discover this property. We strengthen the switch encoding by statically adding mutual exclusion constraints of the form $(\neg s_h \vee \neg s_k)$, with $h \neq k$. Adding this information to the encoding is a form of static learning, and it can guide the Boolean search by pruning branches that are unsatisfiable in the theory. The *switch encoding with mutual exclusion* is presented in Equation 3.

$$\begin{aligned}\textstyle\bigwedge_{i=1}^{|C_f|}(\bigwedge_{j=1}^{D_i}((\neg s_{i,j} \vee ((x_{i,j} - y_{i,j}) \geq l_{i,j})) \wedge \\ (\neg s_{i,j} \vee ((x_{i,j} - y_{i,j}) \leq u_{i,j}))) \wedge \\ (\bigvee_{j=1}^{D_i} s_{i,j}) \wedge (\bigwedge_{j=1}^{D_i} \bigwedge_{k=j+1}^{D_i} (\neg s_{i,j} \vee \neg s_k)))\end{aligned} \tag{3}$$

This encoding is in CNF, but its size is quadratic in the size of the TP— or, more specifically, in $(\max_i D_i)$, i.e. the size of the longest clause.

A different encoding for the TCSP problem class is obtained as follows. For each constraint, we constrain $t \in [l_1, u_D]$, and we exclude the "holes" between intervals, a hole being an open interval (u_j, l_{j+1}). The result is the *hole encoding* reported in Equation 4.

$$\begin{aligned}\textstyle\bigwedge_{i=1}^{|C_f|}((((x_i - y_i) \geq l_{i,1}) \wedge ((x_i - y_i) \leq u_{i,D_i}) \wedge \\ (\bigwedge_{j=1}^{D_i - 1}((x_i - y_i) \leq u_{i,j}) \vee ((x_i - y_i) \geq l_{i,(j+1)})))\end{aligned} \tag{4}$$

This encoding is linear in the size of the original TP, does not introduce any additional variable, and, most importantly, results in a 2-CNF formula. These properties are noteworthy and will be exploited in the following sections.

Finally, we notice that Equation 4 is logically equivalent to Equation 1 (in the applicable case of TCSP), while Equations 2 and 3 are only are equi-satisfiable to it, because of the added switch variables. The solution to the temporal problem is still obtained directly from any satisfying assignment, gathering the values for the variables in $\vec{X_c}$.

6 Encoding of SC Problems in SMT

We now consider the SC problem, in which some time points are not schedulable by the solver, and are considered uncontrollable when looking for a schedule for the controllable time points. We describe the reduction of the SC problem to SMT. We developed a number of encodings that are satisfiable if and only if the temporal problem is strongly controllable, and such that a model of each encoding yields a solution for the original problem.

In the following, we assume that a TPU (X_c, X_u, C_c, C_f) is given.

6.1 Encodings into Quantified LRA

As in the previous section, each time point is associated with an SMT variable. The encoding in Equation 5 is a direct logical mapping of the notion of strong controllability; we call this encoding *direct encoding.*

$$\forall \vec{X}_u . (C_c(\vec{X}_c, \vec{X}_u) \rightarrow C_f(\vec{X}_c, \vec{X}_u)) \tag{5}$$

Equation 5 is satisfiable if and only if there exists an assignment to the controllable variables X_c such that, for all assignments to the uncontrollable variables X_u satisfying the contingent constraints C_c, the free constraints C_f are also satisfied. In the above formula, the controllable variables are implicitly existentially quantified. In case of satisfiability, the SMT solver returns a satisfying assignment to the controllable variables that is exactly a strong schedule.

In order to enable further simplifications, we notice that contingent constraints depend both on controllable and uncontrollable time points, and we re-code the problem as follows. We rewrite each uncontrollable time point e_i in terms of the time difference with its starting time point b_{j_i} by means of an uncontrollable offset variable y_i. For every contingent constraint $cc_i = e_i - b_{j_i} \in [l_i, u_i]$, let $y_i \in \mathbb{R}$ be the uncontrollable offset variable associated to e_i such that: $0 \leq y_i \leq u_i - l_i$ and $e_i = b_{j_i} + u_i - y_i$. Intuitively, y_i represents the offset w.r.t. maximum duration, and can be used to rewrite all the constraints involving e_i in terms of b_{j_i} and y_i only. We formalize this rewriting as a function ρ such that $\rho(e_i) \doteq b_{j_i} + u_i - y_i$. With a small abuse of notation, we denote with $\rho(\vec{X}_u)$ the vector of formulae obtained by the application of ρ to all the elements of X_u. To simplify the notation, we also introduce the vector $\vec{Y}_u$ that is the vector of uncontrollable offset variables $(y_1, ..., y_m)$. Thanks to the redefinition of each e_i in terms of y_i, the rewriting of the contingent constraints depends on $\vec{Y}_u$ only.

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

$$\Gamma(\vec{Y}_u) \doteq \bigwedge_{k=1}^{m} (y_k \in [0, (u_k - l_k)]) \qquad \Psi(\vec{X}_c, \vec{Y}_u) \doteq \bigwedge_{c \in C_f} c[\vec{X}_u / \rho(\vec{X}_u)](\vec{X}_c, \vec{Y}_u)$$

In this setting, the SC consists in finding a value for $\vec{X}_c$ that satisfies the free constraints $\Psi(\vec{X}_c, \vec{Y}_u)$ under any possible value of $\vec{Y}_u$ that satisfies $\Gamma(\vec{Y}_u)$.

The SC encoding in Equation 5 can be recoded as an LRA formula in the free variables $\vec{X}_c$ as follows.

$$\forall \vec{Y}_u . (\Gamma(\vec{Y}_u) \rightarrow \Psi(\vec{X}_c, \vec{Y}_u)) \tag{6}$$

We call this encoding *offset encoding*. This formulation corresponds to a quantified SMT problem in LRA, and still requires a solver that supports quantified formulae, but the part of the encoding representing the contingent constraint is now dependent on $\vec{Y}_u$ only.

The main problem in the previous encodings is the scope of the universal quantifier. Since the computational cost of quantification is very high, we can rewrite the offset encoding in Equation 6 in order to obtain a possibly more efficient encoding. Let us assume that $\Psi(\vec{X}_c, \vec{Y}_u)$ is written as a conjunction of h clauses $\psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h})$, where $X_{c_h} \subseteq X_c$ and $Y_{u_h} \subseteq Y_u$ are the variables used in the clause ψ_h. This assumption can be easily satisfied by converting $\Psi(\vec{X}_c, \vec{Y}_u)$ in CNF. We can rewrite $\neg\Gamma(\vec{Y}_u)$ as $\bar{\Gamma}(\vec{Y}_u) \doteq \bigvee_{k=1}^{m}((y_k < 0) \vee (y_k > (u_k - l_k)))$. Let $\bar{\Gamma}(\vec{Y}_u)|_{Y_{u\,k}} \doteq \bigvee_{y_k \in Y_{u\,k}}((y_k < 0) \vee (y_k > (u_k - l_k)))$.

Assuming the temporal problem is consistent, we have that $\bigwedge_h \forall \vec{Y}_u.(\bar{\Gamma}(\vec{Y}_u) \vee \psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h}))$ if and only if $\bigwedge_h \forall \vec{Y}_{u_h}.(\bar{\Gamma}(\vec{Y}_u)|_{Y_{u_h}} \vee \psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h}))$, and we obtain the *distributed encoding* of Equation 7.

$$\bigwedge_h \forall \vec{Y}_{u_h}.(\bar{\Gamma}(\vec{Y}_u)|_{Y_{u_h}} \vee \psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h})) \tag{7}$$

The size of the produced (quantified) formula is linear with respect to the original TPU. This encoding still requires a solver that supports quantified formulae, and contains as many quantifiers as clauses. However, each quantification is now restricted to the offset variables $Y_{u_h} \subseteq Y_u$ occurring in each clause ψ_h. This encoding also limits the scope of the universal quantifiers, which turns out to be beneficial in practice. Intuitively, this is related to the fact that a number of quantifier eliminations in LRA on smaller formulae may be much cheaper than a single, monolithic quantifier elimination over a large formula.

6.2 Encodings into Quantifier-free LRA

In order to exploit solvers that do not support quantifiers, we propose an encoding of strong controllability into a quantifier-free SMT(LRA) formula. This is obtained by resorting to an external procedure for quantifier elimination.

We rewrite Equation 7 as $\bigwedge_h \neg(\exists \vec{Y}_{u_h}.(\neg\bar{\Gamma}(\vec{Y}_u)|_{Y_{u_h}} \wedge \neg\psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h})))$, in order to apply a procedure for the elimination of existential quantifiers from a conjunction of literals (e.g. Fourier-Motzkin [14]). Notice that both $\bar{\Gamma}(\vec{Y}_u)|_{Y_{u_h}}$ and $\psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h})$ are clauses, and thus their negations are both conjunctions of literals. The result of each quantifier elimination is again a conjunction of literals, which, once negated, yields a clause, in the following referred to as $\psi_h^{\Gamma}(\vec{X}_{c_h})$. The resulting encoding, reported in Equation 8, is called *eager for-all elimination encoding.*

$$\bigwedge_h \psi_h^{\Gamma}(\vec{X}_{c_h}) \tag{8}$$

For the TCSPU class, it is not necessary to apply a general purpose quantifier elimination procedure. Given the specific nature of the constraints, only few cases are possible, and for each of them we use a pattern-based encoding, that in essence precomputes the result of quantifier elimination. This result can be thought of as generalizing to TCSPU the result proposed in [2] for the case of STPU. We start from the distributed encoding of Equation 7, where each (sub)clause ψ_h is generated by the hole encoding. We treat each clause as a separate existential quantification problem, and provide static results for each case. The final result is logically equivalent to the corresponding $\psi_h^{\Gamma}(\vec{X}_{c_h})$ in Equation 8.

Each clause under analysis results from the encoding of a free constraint in the TCSPU over variables v and w, with D intervals. Let t be $v - w$. The encoding results in two unit clauses ($t \geq l_1$ and $t \leq u_D$), and in $D-1$ binary clauses in the form $(t \leq u_i) \vee (t \geq l_{i+1})$.

The static elimination procedure must deal with four possible cases, depending on v and w being controllable or uncontrollable[1]. For the two unit clauses, we proceed as in [2]. Here we show the more complex cases of binary clauses. Let the binary clause have the form $(v - w \leq u) \vee (v - w \geq l)$ (notice that $u < l$ because u is the upper bound of the "lower" interval). When v is uncontrollable, we write x_v for its starting point, y_v for its offset, and L_v and U_v for the lower and upper bound of the contingent constraint relative to v[2]; similarly for w.

1. $v \in X_c$ and $w \in X_c$. The clause does not contain quantified variables, and therefore the quantifier can be simply removed.
2. $v \in X_c$ and $w \in X_u$. The formula $\neg\bar{\Gamma}(\vec{Y}_u)|_{\{y_w\}} \wedge \neg\psi(v, x_w, y_w)$ can be represented by:

$$\begin{aligned}&(0 \leq y_w) \wedge (y_w \leq U_w - L_w) \wedge\\ &(y_w < l - v + x_w + U_w) \wedge (x_w - v + U_w + u < y_w).\end{aligned}$$

   Using quantifier elimination over $\exists y_w.(\neg\bar{\Gamma}(\vec{Y}_u)|_{\{y_w\}} \wedge \neg\psi(v, x_w, y_w))$, we obtain the following formula (given that $(l - u > 0)$ and $(U_w - L_w > 0)$):

$$\begin{aligned}&((l - v + x_w + U_w > 0) \wedge (l - v + x_w + L_w \leq 0)) \vee\\ &((l - v + x_w + L_w \geq 0) \wedge (v - x_w - u - L_w > 0)).\end{aligned}$$

   Since in eager for-all elimination encoding we need the negation of the existential quantification we can rewrite the formula as follows:

$$\begin{aligned}&((l - v + x_w + U_w \leq 0) \vee (l - v + x_w + L_w > 0)) \wedge\\ &((l - v + x_w + L_w < 0) \vee (v - x_w - u - L_w \leq 0)).\end{aligned}$$

3. The case when $v \in X_u$ and $w \in X_c$ is dual:

$$\begin{aligned}&((x_v + U_v - w - u \leq 0) \vee (x_v - w - u + L_v > 0)) \wedge\\ &((x_v - l - w + L_v \geq 0) \vee (x_v - w - u + L_v < 0)).\end{aligned}$$

4. $v \in X_u$ and $w \in X_u$. The formula $\neg\bar{\Gamma}(\vec{Y}_u)|_{\{y_v,y_w\}} \wedge \neg\psi(x_v, x_w, y_v, y_w)$ is thus $\neg\bar{\Gamma}(\vec{Y}_u)|_{\{y_v,y_w\}} \wedge (v - w < l) \wedge (v - w > u)$ which in turn can be rewritten as

$$\begin{aligned}&(x_v + U_v - x_w - U_w + y_w - l < y_v) \wedge\\ &(y_v < x_v + U_v - x_w - U_w + y_w - u) \wedge\\ &(0 \leq y_v) \wedge (y_v \leq U_v - L_v) \wedge (-y_w \leq 0) \wedge (y_w \leq U_w - L_w).\end{aligned}$$

[1] The possible cases are actually eight but $v - w \geq k$ can be rewritten as $w - v \leq -k$

[2] We assume $L_v < U_v$; in the other cases the problem is not interesting.

Using the assumptions detailed above and negating the quantification result, we obtain the following formula:

$$
\begin{aligned}
&((x_v + U_v - x_w - U_w - u > 0) \vee (x_v + U_v - x_w - u - L_w \leq 0)) \wedge \\
&((x_v + U_v - x_w - U_w - u < 0) \vee (x_v - x_w - U_w - u + L_v \geq 0)) \wedge \\
&((x_v - x_w - U_w - l + L_v \geq 0) \vee (x_v - x_w - l + L_v - L_w < 0)) \wedge \\
&((x_v - x_w - l + L_v - L_w > 0) \vee (x_v - x_w - u + L_v - L_j \leq 0)).
\end{aligned}
$$

The construction described above can be used in Equation 8. This specialized quantification technique results in a 2-CNF formula that has size linear in the original TCSPU. This is because the size of the hole encoding is linear, and for each clause, we statically resolve the quantification by creating at most four new binary clauses. As far as encoding time is concerned, for a TCSPU with m free constraints, the encoding can be generated in $O(m * max(D_i) log(max(D_i)))$ time, because of the sorting time needed in the hole encoding. This encoding spares the computational cost of quantifier elimination and produces a highly optimized QF_LRA formula.

7 Related Work

The seminal work on strong controllability is [2]. The problem is tackled for the limited case of STPU problem class. Vidal and Fargier identify a clever, constant time quantification technique for SC reasoning, which is at the core of their procedure. Compared to [2], we propose a comprehensive solution and an implementation for the cases of TCSPU and DTPU. Furthermore, we generalize to the case of TCSPU the specialized quantifier elimination techniques proposed in [2] for STPU.

Strong controllability for the cases beyond STPU have been tackled in [19], where specialized algorithms based on meta-CSP are proposed. The work in [19] tackles the same problem addressed here; however, it is purely theoretical, and to the best of our knowledge no implementation exists. Furthermore, the approach is based on the use of explicit CSP search to deal with case splits, while we rely on the symbolic expressive power of the SMT framework.

The use of SMT techniques to solve temporal problems is not new. The most advanced work is presented in [20], where the *TSAT++* tool is presented. TSAT++ can be seen as a specialized SMT solver for DTP problems. The work does not deal with strong controllability, and is limited to consistency for temporal problems. The performance of TSAT++ relative to more modern SMT solvers is analyzed in the next section, on temporal consistency problems.

As far as the consistency problem of STP is concerned, the work in [22] represents the state-of-the-art. Planken, de Weerdt and van der Krogt presented an efficient algorithm for computing all-pairs shortest paths in a directed graph with possibly negative Real weights. As pointed out by the authors, the proposed algorithm can be used to solve STP (but not TCSP or DTP). We used their tool in our experimental comparison for STP consistency. We remark that the focus of our work is on the strong controllability (and not consistency) problem.

We also mention two other forms of controllability for TPUs: weak controllability (WC) and dynamic controllability (DC). A TPU is said to be WC if, for every possible evolution of the uncontrollable environment, there exists an allocation to the controllable time points that fulfills the free constraints of the problem. This notion is much weaker than SC, because the allocation strategy for the controllable time points is allowed to depend on the allocation of the uncontrollable time points. In this setting, the solver is assumed to be "clairvoyant" and is able to decide its moves based on the past and also the future moves of the opponent. In their seminal paper, Vidal and Fargier [2] address the WC problem for the STPU class. Algorithms for deciding WC for TCSPU and DTPU are provided in [23]. The use of SMT techniques to deal with weak controllability has been recently investigated in [24], addressing both the decision and the strategy extraction problems (i.e. the problem of checking if a TPU is WC, and the problem of building a strategy for the solver). The work presented in this paper, compared to [24], tackles a radically different problem. An important difference between SC and WC is the shape of the solution: while in SC a solution is a static assignment to controllable time points, in WC the strategy requires conditional structures to be expressed. Thus, the use of SMT techniques in [24] is also substantially different from what is done here.

DC is similar to WC, but the choices of the scheduler can be based on past environment decisions only. As pointed out in [2], if a problem is SC then it is also DC and if it is DC then it is also WC, but the implication chain is not reversible. In [25] the authors focus on deciding DC for the STPU problem class, while in [23] the result is extended for TCSPUs. However, no effective solutions to DC exists for the DTPU problem class.

8 Experimental Evaluation

8.1 Implementation

We developed a tool that automatically encodes the various classes of temporal problems in SMT problems. The tool can deal with consistency problems by generating SMT (QF_LRA) encodings. As for strong controllability problems, the tool implements the two reductions to SMT (LRA) (with quantifiers), and can obtain SMT (QF_LRA) by applying quantifier elimination techniques. The quantifier elimination step in the eager for-all elimination encoding is carried out by calling the formula simplifier provide by Z3 [6], and a quantifier elimination functionality built on top of MathSAT5 [5].

The tool is currently connected to three different SMT solvers: namely *MathSAT4* [4], *MathSAT5* [5] and *Z3* [6]. Given that the encodings are written in SMT-LIB2 (and also in SMT-LIB1 format), it would be straightforward to connected it to any SMT solver that is able to parse the SMT-LIB language. We remark however that our purpose is to compare the performance of the encodings we propose, and not to compare the various SMT solvers. Z3 can be seen as a representative for solvers that support quantified theories, and MathSAT as representative for quantifier-free solvers. We expect other solvers (e.g. Yices [7], OpenSMT [8]) to exhibit a similar behavior (see [26]).

8.2 Experimental set-up

In order to experimentally assess the performance of the techniques presented in this paper, we used a set of randomly-generated benchmarks. Consistency problems were generated using the random instance generator presented in [20]; the same generator was extended to introduce random uncertainty, and to generate strong controllability problems. The benchmark set contains 2108 instances for each problem class in TP without uncertainty (STP, TCSP and DTP), and 1054 instances for each TPU class (STPU, TCSPU and DTPU). We used random instance generators because they are typically used in literature (e.g. [20]), and because they can be easily scaled to stress the solvers.

We executed all our experiments on a machine running Scientific Linux 6.0, equipped with two quad-core Xeon processors @ 2.70GHz. We considered a memory limit of 2GB and a time-out of 300 seconds. The benchmarks and the results are available from `https://es.fbk.eu/people/roveri/tests/cp2012`.

For consistency problems, we analyzed the performance of the various solvers on the various encodings. We also compared our encodings with all the available solver for TP without uncertainty (i.e. Snowball for the case of STP, and TSAT++).

For strong controllability problems, to the best of our knowledge there are no solvers available. Thus, we only evaluated the different approaches, to highlight the difference in performance and the respective merits.

8.3 Results for Consistency

The results for consistency problems are reported in Figure 1 (left). We plotted in logarithmic scale the cumulative time in seconds to solve the considered set of benchmarks. For STP problems, we compared the naïve encoding with various algorithms available in the *SnowBall3* [22] tool, and with TSAT++ [20]. (In the case of STP, the other encodings coincide with the naïve encoding.) In TCSP and DTP, we tested all the applicable encodings with all the SMT solvers under analysis and with TSAT++. The plots show that the SMT approach is competitive with dedicated techniques. MathSAT4 implements a dedicated algorithm for the theory of difference logic [27], and is thus faster than MathSAT5, that uses a general purpose algorithm for LRA [28]. Both solvers greatly benefit from the hole encoding, compared to the switch encoding with mutual exclusion (switch me) and the plain switch encoding. This encoding produces a formula that has just one real variable for every time point and has at most two literals per clause: this greatly simplifies the SMT search procedure by augmenting the number of unit propagations and by reducing the size of the search space.

Z3 is extremely efficient, and the attempts to improve the encodings may result in performance degradation. The TSAT++ solver is outperformed by state-

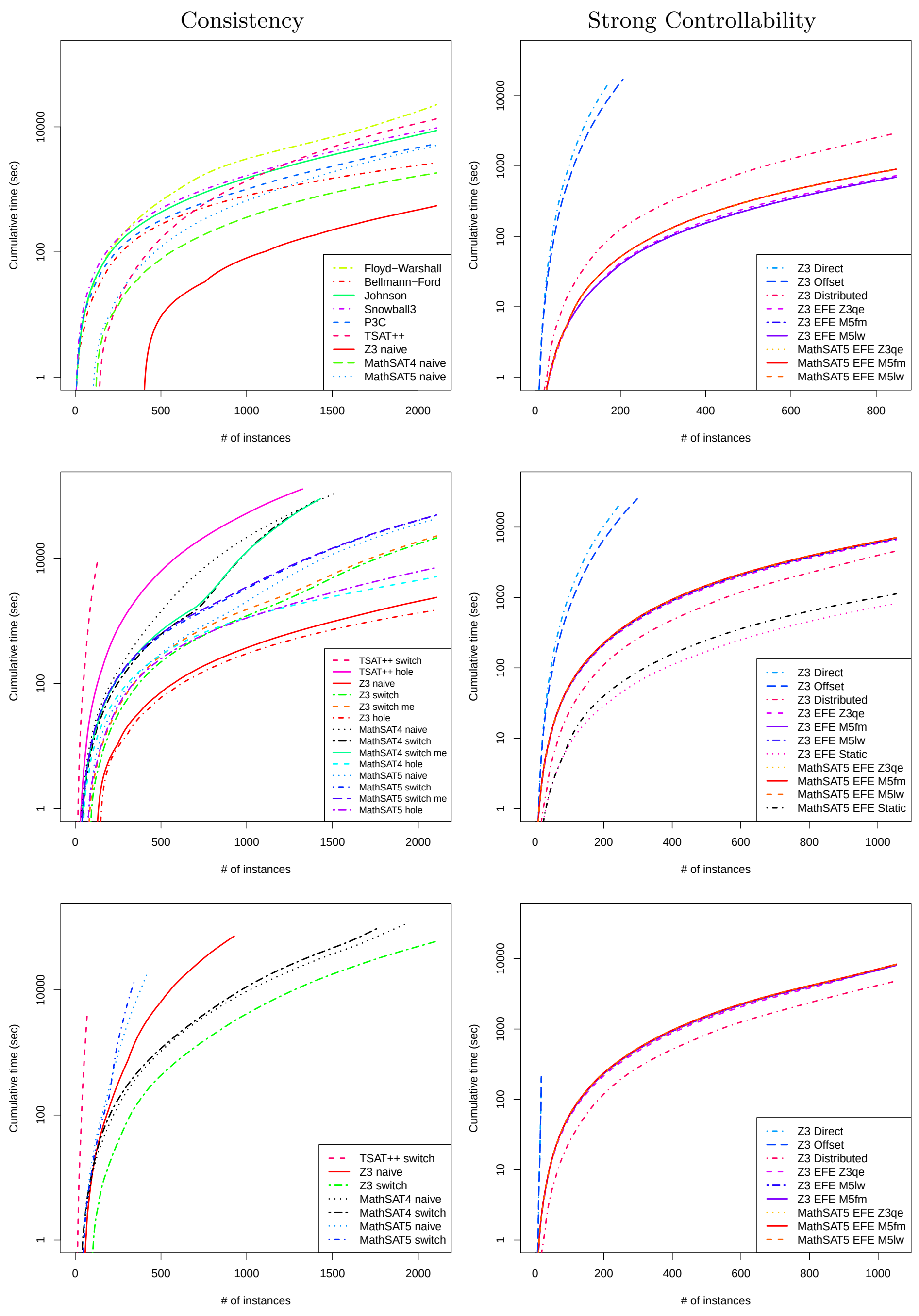

Fig. 1. Results for experimental evaluation: consistency of STP (left-top), TCSP (left-center), and DTP (left-bottom); strong controllability of STPU (right-top), TCSPU (right-center) and DTPU (right-bottom).

of-the-art SMT solvers, but again notice that the hole encoding yields substantial improvements in performance.

8.4 Results for Strong Controllability

The results for strong controllability are reported in Figure 1 (right). We plotted in logarithmic scale the cumulative time in seconds to solve the considered set of benchmarks. Differently from the consistency case, the time here considers also the encoding time which dominates solving time for the case of quantifier-free encodings. The quantified encodings (Direct, Offset and Distributed) are solved with Z3. The quantifier-free encodings resulting from eager for-all elimination are obtained by the application of three quantifier elimination procedures: the internal simplifier of Z3 (EFE Z3qe), and two implementations in MathSAT5 of the Fourier-Motzkin (EFE M5fm) and the Loos-Weispfenning (EFE M5lw) elimination procedures. The resulting encodings are solved using Z3 and MathSAT5 on the quantifier-free theory of Reals.

The plots clearly show that both the offset and direct encodings quickly reach the resource limits, and are unable to solve all the instances. The behavior of the distributed encoding is slightly better than the eager for-all elimination approach. The difference can be explained in purely technological terms: the quantifier elimination modules are called via pipe in our implementation; on the other hand, when Z3 solves the distributed encoding, it can perform quantifier elimination “in-memory”.

We notice the impact of the static quantification techniques (EFE Static), when applicable (i.e. for TCSPU). The substantial difference in performance resides only in the quantification technique, that all produce the same quantifier-free formula.

9 Conclusions

In this paper, we presented a comprehensive approach to strong controllability for temporal problems with uncertainty. The formalization is based on the SMT framework, and encompasses constraints with arbitrary disjunctions. We deal with uncertainty by means of logic-based quantifier elimination techniques. The experiments demonstrate the scalability of the approach, based on the use of efficient SMT solvers.

In the future, we will investigate the problem of searching schedules that optimize a given cost function, and the addition of constraints over resources associated to activities. Finally, within the SMT-based framework, we will investigate the case of dynamic controllability.

References

1. Dechter, R., Meiri, I., Pearl, J.: Temporal constraint networks. Artif. Intell. **49** (1991) 61–95
2. Vidal, T., Fargier, H.: Handling contingency in temporal constraint networks: from consistency to controllabilities. Journal of Experimental Theoretical Artificial Intelligence **11** (1999) 23–45
3. Barrett, C.W., Sebastiani, R., Seshia, S.A., Tinelli, C.: Satisfiability modulo theories. In: Handbook of Satisfiability. IOS Press (2009) 825–885
4. Bruttomesso, R., Cimatti, A., Franzén, A., Griggio, A., Sebastiani, R.: The MathSAT 4SMT Solver. In Gupta, A., Malik, S., eds.: Computer Aided Verification - CAV. Volume 5123 of LNCS., Springer (2008) 299–303
5. Cimatti, A., Griggio, A., Sebastiani, R., Schaafsma, B.: The MathSAT5 SMT solver. `http://mathsat.fbk.eu` (2011)
6. de Moura, L.M., Bjørner, N.: Z3: An Efficient SMT Solver. In Ramakrishnan, C.R., Rehof, J., eds.: Tools and Algorithms for the Construction and Analysis of Systems - TACAS. Volume 4963 of LNCS., Springer (2008) 337–340
7. Dutertre, B., de Moura, L.: The Yices SMT solver. Tool paper at `http://yices.csl.sri.com/tool-paper.pdf` (2006)
8. Bruttomesso, R., Pek, E., Sharygina, N., Tsitovich, A.: The OpenSMT Solver. In Esparza, J., Majumdar, R., eds.: Tools and Algorithms for the Construction and Analysis of Systems - TACAS. Volume 6015 of LNCS., Springer (2010) 150–153
9. Niemelä, I.: Integrating Answer Set Programming and Satisfiability Modulo Theories. In Erdem, E., Lin, F., Schaub, T., eds.: Logic Programming and Nonmonotonic Reasoning, 10th International Conference - LPNMR. Volume 5753 of LNCS., Springer (2009) 3
10. Franzén, A., Cimatti, A., Nadel, A., Sebastiani, R., Shalev, J.: Applying SMT in symbolic execution of microcode. In Bloem, R., Sharygina, N., eds.: Formal Methods in Computer-Aided Design - FMCAD, IEEE (2010) 121–128
11. Godefroid, P., Levin, M.Y., Molnar, D.A.: Automated whitebox fuzz testing. In: Network and Distributed System Security Symposium - NDSS, The Internet Society (2008)
12. Kleene, S.: Mathematical Logic. J. Wiley & Sons (1967)
13. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: Engineering an Efficient SAT Solver. In: Design Automation Conference - DAC, New York, NY, USA, ACM Press (2001) 530–535
14. Schrijver, A.: Theory of Linear and Integer Programming. J. Wiley & Sons (1998)
15. Loos, R., Weispfenning, V.: Applying linear quantifier elimination. Computer Journal **36** (1993) 450–462
16. Monniaux, D.: A Quantifier Elimination Algorithm for Linear Real Arithmetic. In Cervesato, I., Veith, H., Voronkov, A., eds.: Logic for Programming, Artificial Intelligence, and Reasoning - LPAR. Volume 5330 of LNCS., Springer (2008) 243–257
17. Allen, J.F.: Maintaining knowledge about temporal intervals. Communication of the ACM **26** (1983) 832–843
18. Tsamardinos, I., Pollack, M.E.: Efficient solution techniques for disjunctive temporal reasoning problems. Artificial Intelligence **151** (2003) 43 – 89
19. Peintner, B., Venable, K.B., Yorke-Smith, N.: Strong controllability of disjunctive temporal problems with uncertainty. In Bessiere, C., ed.: Principles and Practice of Constraint Programming - CP. Volume 4741 of LNCS., Springer (2007) 856–863

20. Armando, A., Castellini, C., Giunchiglia, E.: SAT-Based Procedures for Temporal Reasoning. In Biundo, S., Fox, M., eds.: European Conference on Planning - ECP. Volume 1809 of LNCS., Springer (1999) 97–108
21. de la Tour, T.: Minimizing the number of clauses by renaming. In Stickel, M., ed.: Conference on Automated Deduction - CADE. Volume 449 of LNCS. Springer (1990) 558–572
22. Planken, L., de Weerdt, M., van der Krogt, R.: Computing all-pairs shortest paths by leveraging low treewidth. Journal of Artificial Intelligence Research (JAIR) **43** (2012) 353–388
23. Venable, K.B., Volpato, M., Peintner, B., Yorke-Smith, N.: Weak and dynamic controllability of temporal problems with disjunctions and uncertainty. In: Workshop on Constraint Satisfaction Techniques for Planning & Scheduling. (2010) 50–59
24. Cimatti, A., Micheli, A., Roveri, M.: Solving Temporal Problems using SMT: Weak Controllability. In Hoffmann, J., Selman, B., eds.: American Association for Artificial Intelligence - AAAI, AAAI Press (2012)
25. Morris, P.H., Muscettola, N., Vidal, T.: Dynamic control of plans with temporal uncertainty. In Nebel, B., ed.: International Joint Conference on Artificial Intelligence - IJCAI, Morgan Kaufmann (2001) 494–502
26. Barrett, C., Deters, M., de Moura, L., Oliveras, A., Stump, A.: 6 Years of SMT-COMP. Journal of Automated Reasoning (2012) To appear.
27. Cotton, S., Maler, O.: Fast and flexible difference constraint propagation for dpll(t). In Biere, A., Gomes, C.P., eds.: Theory and Applications of Satisfiability Testing - SAT. Volume 4121 of LNCS., Springer (2006) 170–183
28. Dutertre, B., de Moura, L.M.: A Fast Linear-Arithmetic Solver for DPLL(T). In Ball, T., Jones, R.B., eds.: Computer Aided Verification - CAV. Volume 4144 of LNCS., Springer (2006) 81–94